DESIGN AND ANALYSIS OF LINEAR SAMPLED-DATA CONTROL SYSTEMS USING DIGITAL SIMULATION TECHNIQUES

A Thesis

Presented to

the Faculty of the Graduate School

The University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

---

by

Raymond Earl Fulghum

May, 1970

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Professor W. P. Schneider for both his excellent lectures and his guidance during the development of this thesis. The helpful comments and cooperation of Dr. E. D. Denman and Dr. C. E. Donaghey must also be acknowledged with gratitude.

A special note of thanks is due Professor C. F. Chen for his invaluable advice and instruction in automatic control theory, and Mr. G. W. Oprea of Houston Lighting and Power Company for his approval of computer usage.

The author finds it impossible to express his full gratitude to his wife, Kathleen, for her encouragement and patience throughout the period of his graduate education and research.

DESIGN AND ANALYSIS OF LINEAR SAMPLED-DATA CONTROL SYSTEMS

USING DIGITAL SIMULATION TECHNIQUES

Abstract of a Thesis

Presented to

the Faculty of the Graduate School

The University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

---

by

Raymond Earl Fulghum

May, 1970

ABSTRACT

The problem of discrete compensator design for acceptable transient response in linear stationary plants modeled in state-variable form is considered in this research. A digital computer algorithm is programmed in FORTRAN for calculation of the feedforward compensator transfer function coefficients. The generalized method for realization of the compensator is demonstrated. System performance with no compensation, classical phase-lead compensation, and discrete compensation is exemplified via a CSMP simulation. An investigation of the effect of saturation nonlinearity in the plant with respect to the discrete compensator design is included.

by

Raymond Earl Fulghum

May, 1970

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

Background

Industrial corporations have recently been confronted with consumer demands for increased economy and reliability. The concurrent dissemination of knowledge in the field of automatic control theory has led to research in automatic control applications.

The electric utility industry is faced with a twofold control problem--the economic dispatch and load frequency requirements. The economic dispatch algorithm must calculate a set of steady-state operating points for the generators such that the power system operating costs and energy loss in transmission are minimized for generator output perturbations. The load frequency control program must calculate and allocate a set of generator demand inputs, altering real power generation in response to system load variations such that integrated system control error is minimized.

A practical solution to the economic dispatch of real power has been developed by Kirchmayer (1958). Research concerning the extension of economic dispatch philosophy to reactive power is in progress (Adibi, 1969).

The load frequency control problem has been widely investigated with classical work by Cohn (1957) and most recent effort by Elgerd and Fosha (1969). This prior load frequency control research was concentrated upon the dynamic characteristics of the power system as a whole, ignoring the dynamics peculiar to single prime mover-turbine-generator combinations. Justification for this approach lay in the fact that the power system time constants associated with frequency and net interchange variations were two orders of magnitude larger than the time constants for output variations in turbine-generator units. Furthermore, the subcritical steam generators and hydro units traditionally used as prime movers could be considered as energy sources. As such, these prime movers had insignificant time constants with respect to load frequency control (Cohn, 1957). Figure 1.1 shows the traditional relationship of a typical prime mover-turbine-generator combination to the load frequency control system.

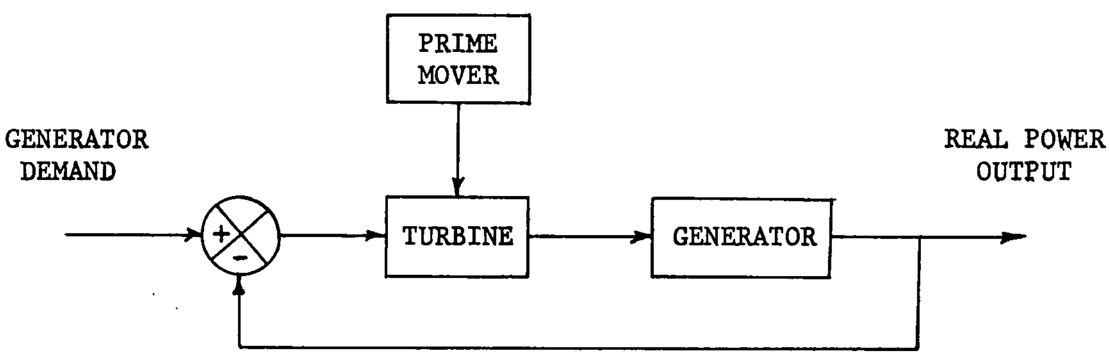

Figure 1.1

TRADITIONAL LOAD FREQUENCY CONTROL
RELATIONSHIP

The need for increased thermodynamic efficiency in the prime mover has recently been served by the installation of supercritical (once-through) steam generators. The major local electric utility, Houston Lighting and Power Company, currently produces fifty-three per cent of its system capability with four supercritical units. Power output of a once-through boiler is a function of fuel and air input. Also, the time constants associated with supercritical boiler dynamics are of the same order of magnitude as the time constants for system frequency and net interchange variations. Therefore, the once-through boiler must be placed in the load frequency control loop, as illustrated in Figure 1.2 (Kenny, 1967).

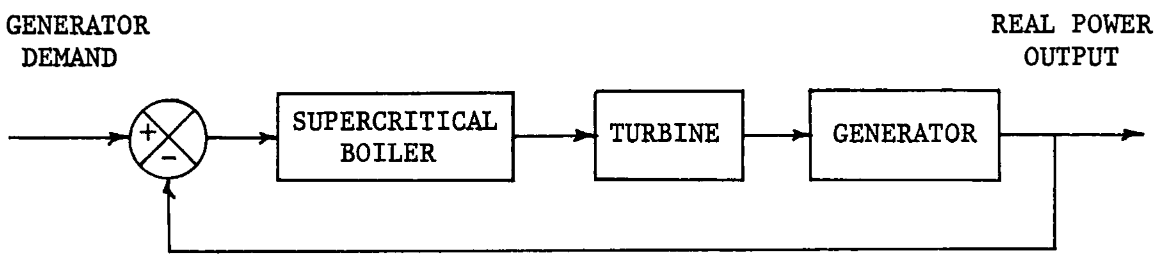

Figure 1.2

SUPERCRITICAL BOILER/LOAD FREQUENCY CONTROL RELATIONSHIP

Problem Definition

The area of investigation in this work is one of compensator design for acceptable transient response of a generalized boiler-turbine-generator unit under load frequency control. As the load frequency control is accomplished in a region of perturbation about a steady-state operating point, a linear plant model of the boiler-turbine-generator will be satisfactory (Schultz and Melsa, 1967:112-118). The time constants for parameter variations in the plant are on the order of months; thus the linear plant will be considered time-invariant with respect to load frequency control.

Houston Lighting and Power Company is currently implementing a typical digital load frequency control program. The algorithm employs sampled-data inputs and generator demand outputs, both of identical constant periodicity. The generator demand signal is conditioned to a step of constant amplitude during the sample period. The step has height directly proportional to the magnitude of the generator demand and sign indicating the direction of change. Also, the load frequency control algorithm treats each boiler-turbine-generator unit as a single-input (generator demand) and single-output (real power) plant.

Within the context of plant real power output response to generator demand input, acceptable transient response may be defined as small rise time, small overshoot, and small settling time. The specific problem considered in this thesis is, therefore, compensator design for approximately deadbeat response of a single-variable, linear, time-invariant plant to a step input.

CHAPTER II

STATE-VARIABLE THEORY

This chapter prepared the reader with the basics of state-variable theory as background for the compensator design procedure developed in Chapter III. The state-variable representation is defined, and the state transition equation for continuous variables is derived.

State-Variable Representation

A state-variable representation is the expression of an n-th order ordinary differential equation as a set of n first order ordinary differential equations (Dorf, 1965). For example, consider the second order ordinary differential equation

$$a_2 \frac{d^2y}{dt^2} + a_1 \frac{dy}{dt} + a_0 y = r \qquad 2.1$$

Letting $x_1 = y$ and $x_2 = \frac{dy}{dt}$, the set of first order differential equations may be written as

$$\frac{dx_1}{dt} = x_2 \qquad 2.2$$

$$\frac{dx_2}{dt} = -\frac{a_0}{a_2} x_1 - \frac{a_1}{a_2} x_2 + \frac{r}{a_2} \qquad 2.3$$

The time derivative $\frac{dx_n}{dt}$ is written in the form $\dot{x}_n$ throughout the remainder of this work. As Equation 2.1 is linear, Equations 2.2 and 2.3 may be written in matrix form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{a_0}{a_2} & -\dfrac{a_1}{a_2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{a_2} \end{bmatrix} r \qquad 2.4$$

Here the dependent variables x_1 and x_2 are defined as the state-variables. The system represented by Equation 2.1 is completely described by Equation 2.4 and the value of the state vector at some initial time t_0

$$\begin{bmatrix} x_1(t_0) \\ x_2(t_0) \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \qquad 2.5$$

Now the word "plant" will be defined as the unaltered set of first order differential equations describing a process, and the word "system" as the total set of equations describing the original process and any attached compensation in either a feedforward or feedback signal path.

State Transition Equation

Given a linear plant with external input and the initial value of the state vector, calculation of the state vector value for an arbitrarily specified later point in time is possible. A mathematically tractable format for this calculation, the state transition equation, will now be derived.

The generalized matrix notation for describing a linear plant with a single input is

$$\underline{\dot{x}} = A\,\underline{x} + \underline{b}\,r \qquad 2.6$$

where, for an n-vector $\underline{x}$ and scalar input r, A is the $n \times n$ plant matrix and $\underline{b}$ is the n-vector relating the input to the state-variables. For a time-invariant system, the elements of A and $\underline{b}$ are constants.

First consider the scalar, time-invariant form of Equation 2.6

$$\dot{x} = a x + b r \qquad 2.7$$

Taking the Laplace transform of Equation 2.7 results in

$$s X(s) - x(0) = a X(s) + b R(s) \qquad 2.8$$

Solving for X (s),

$$X(s) = \frac{x(0)}{s-a} + \frac{bR(s)}{s-a} \qquad 2.9$$

The inverse Laplace transform of Equation 2.9 is (Cheng, 1961)

$$x = \varepsilon^{at} x(0) + \int_0^t \varepsilon^{a(t-\gamma)} b\, r(\gamma)\, d\gamma \qquad 2.10$$

Equation 2.10 is the scalar state transition equation for continuous variables.

Now a similar set of operations is performed on Equation 2.6, assuming time-invariance.

$$\mathcal{L}\{\dot{\underline{x}}\} = \mathcal{L}\{A \underline{x} + \underline{b} r\} \qquad 2.11$$

$$s \underline{X}(s) - \underline{x}(0) = A \underline{X}(s) + \underline{b} R(s) \qquad 2.12$$

Solving for $\underline{X}$ (s),

$$\underline{X}(s) = \{s I - A\}^{-1} \underline{x}(0) + \{s I - A\}^{-1} \underline{b} R(s) \qquad 2.13$$

where I is the identity matrix.

The inverse Laplace transform of Equation 2.13 is (Schultz and Melsa, 1967:129)

$$\underline{x} = \varepsilon^{At}\,\underline{x}(0) + \int_0^t \varepsilon^{A(t-\gamma)}\,\underline{b}\,r(\gamma)\,d\gamma \qquad 2.14$$

The matrix exponential function ε^{At} will now be expressed in terms of the plant matrix A. Consider the input r of Equation 2.14 equal to zero

$$\underline{x} = \varepsilon^{At}\,\underline{x}(0) \qquad 2.15$$

Now assume a Taylor's Series expansion of the vector $\underline{x}$ about the origin of state space

$$\underline{x} = \sum_{n=0}^{\infty} \underline{c}_n\, t^n \qquad 2.16$$

or

$$\underline{x} = \underline{c}_0 + \underline{c}_1 t + \underline{c}_2 t^2 + \underline{c}_3 t^3 + \ldots \qquad 2.17$$

In order to determine the vector coefficients $\underline{c}_n$, Equation 2.17 is successively differentiated and evaluated at t=0 (Dorf, 1965:8). This set of operations gives

$$\underline{c}_n = \frac{A^n}{n!}\,\underline{x}(0) \qquad 2.18$$

Substituting from Equation 2.18 to Equation 2.16,

$$\underline{x} = \sum_{n=0}^{\infty} \frac{A^n\, t^n}{n!}\,\underline{x}(0) \qquad 2.19$$

Now by comparing like terms of Equation 2.19 and Equation 2.15,

$$\varepsilon^{At} = \sum_{n=0}^{\infty} \frac{A^n\, t^n}{n!} \qquad 2.20$$

The infinite series of Equation 2.20 may be shown to be convergent for all square matrices A (Courant and Hilbert, 1965:19).

By defining the transition matrix $\Phi(t)$ as ε^{At} and specifying arbitrary initial time t_0, Equation 2.14 is rewritten

$$\underline{x}(t) = \Phi(t-t_0)\,\underline{x}(t_0) + \int_{t_0}^{t} \Phi(t-\gamma)\,\underline{b}\,r(\gamma)\,d\gamma \qquad 2.21$$

Equation 2.21 is the general form of the matrix state transition equation for continuous variables (Dorf, 1965:12).

Summary

The discrete compensator design procedure developed in Chapter III requires no more from the theory of state-variables than that presented in this chapter. As the system is composed of a continuous signal plant and a discrete signal compensator, some of the derivations must be made in the z-transform domain instead of the Laplace domain, but the principles and the form of the equations are exactly the same.

CHAPTER III

DEVELOPMENT OF DESIGN PROCEDURE

The first section of this chapter documents the design procedure for the discrete compensator chosen to solve the control problem stated in Chapter I. A realization of the discrete compensator is also defined. A classical continuous compensator design is formulated in section two of this chapter in order that the discrete design and the continuous design may be compared in Chapter IV.

Discrete Compensator

Design procedure. The major portion of the theory used in the discrete compensator synthesis is the result of work by Meksawan and Murphy (1963). The identity of the control problem defined in Chapter I and the problem investigated by Meksawan and Murphy served as the basis for selection of their technique.

Consider the configuration of elements in Figure 3.1.

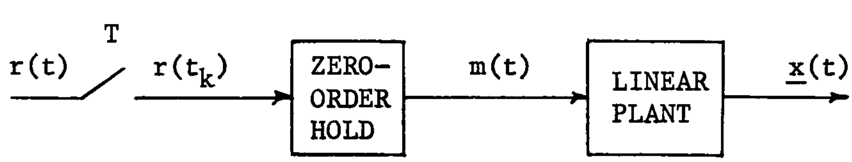

Figure 3.1

OPEN-LOOP SAMPLED SYSTEM

Noting that the linear plant input is m(t), Equation 2.21, page 9, may be rewritten as

$$\underline{x}\,(t) = \Phi\,(t-t_0)\,\underline{x}\,(t_0) + \int_{t_0}^{t} \Phi\,(t-\gamma)\,\underline{b}\,m\,(\gamma)\,d\,\gamma \qquad 3.1$$

The input-output relationship of the zero-order hold is

$$m\ (t_k + \gamma) \ = \ r\ (t_k),\ \text{for}\ 0 < \gamma < t_{k+1} - t_k \qquad 3.2$$

where $r\ (t_k)$ is the instantaneous value of $r(t)$ at corresponding sampling instants t_k, $k = 0, 1, 2, \ldots$. A constant sample period is now chosen, of interval length

$$T \ = \ t_{k+1} - t_k\ . \qquad 3.3$$

With initial time $t_0 = t_k = kT$, Equation 3.1 may be rewritten to give the value of the state vector $\underline{x}(t)$ at $t = t_{k+1} = (k+1)\ T$ as

$$\underline{x}\ [(k+1)T] = \Phi\ (T)\ \underline{x}\ (kT) + \int_{kT}^{(k+1)T} \Phi[(k+1)T-\gamma]\ \underline{b}\ d\ \gamma\ r\ (kT) \qquad 3.4$$

For a time-invariant system, the integral portion of Equation 3.4 is independent of k, so for $k = 0$,

$$\int_{kT}^{(k+1)T} \Phi[(k+1)T-\gamma]\ \underline{b}\ d\ \gamma \ = \ \int_0^T \Phi\ (T-\gamma)\ \underline{b}\ d\ \gamma\ . \qquad 3.5$$

As $\Phi\ (t) \triangleq \varepsilon^{At}$, the value of the integral in Equation 3.5 is independent of the direction of integration; therefore,

$$\int_0^T \Phi\ (T-\gamma)\ \underline{b}\ d\ \gamma = \int_0^T \Phi\ (\gamma)\ \underline{b}\ d\ \gamma \qquad 3.6$$

By definition, let

$$\underline{d}\ (T) \ = \ \int_0^T \Phi\ (\gamma)\ \underline{b}\ d\ \gamma \qquad 3.7$$

Then Equation 3.4 becomes

$$\underline{x}\,[(k+1)T] = \Phi\,(T)\,\underline{x}\,(kT) + \underline{d}\,(T)\,r\,(kT) \qquad 3.8$$

Equation 3.8 may now be used to describe the dynamic behavior of a system containing both continuous and discrete elements, or one about which some behavioral information is available only at discrete instants in time. Such a system is illustrated in Figure 3.2.

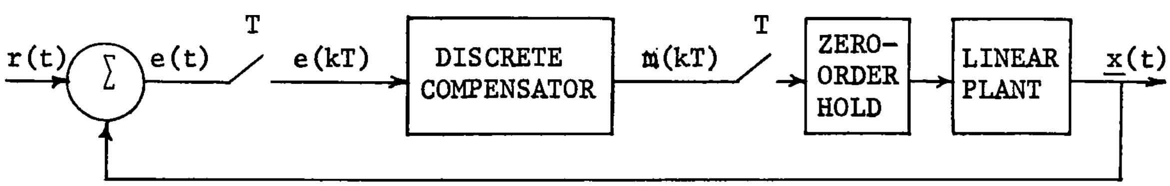

Figure 3.2

CONTROL SYSTEM WITH DISCRETE COMPENSATOR

Rewriting Equation 3.8 in the notation of Figure 3.2,

$$\underline{x}\,[(k+1)T] = \Phi\,(T)\,\underline{x}\,(kT) + \underline{d}\,(T)\,m\,(kT) \qquad 3.9$$

The development of a form of Equation 3.9 suitable for digital computer programming is now performed. Noting that $\Phi(T)$ is a constant matrix and $\underline{d}(T)$ is a constant vector, the z-transform of Equation 3.9 is

$$z\,\underline{X}\,(z) - z\,\underline{x}\,(0) = \Phi\,\underline{X}\,(z) + \underline{d}\,M\,(z) \qquad 3.10$$

Solving for $\underline{X}$ (z),

$$\underline{X}\,(z) = \{z\,I - \Phi\}^{-1}\,z\,\underline{x}\,(0) + \{z\,I - \Phi\}^{-1}\,\underline{d}\,M\,(z) \qquad 3.11$$

From Kuo (1964:136), the inverse z-transform of Equation 3.11 is

$$\underline{x}(kT) = \Phi(kT)\,\underline{x}(0) + \sum_{j=0}^{k-1} \Phi[(k-1-j)T]\,\underline{d}(T)\,m(kT) \qquad 3.12$$

In order to express Equation 3.12 in a more useful form, multiply both sides by $\Phi(-kT)$.

$$\Phi(-kT)\underline{x}(kT) = \Phi(-kT)\Phi(kT)\underline{x}(0) + \sum_{j=0}^{k-1} \Phi(-kT)\Phi[(k-1-j)T]\underline{d}(T)m(kT) \qquad 3.13$$

It may be shown that ε^{At} and ε^{-At} are reciprocal matrices (Schultz and Melsa, 1967:122), so that

$$\Phi(-kT)\Phi(kT) = I \qquad 3.14$$

Also,

$$\Phi(-kT)\Phi[(k-1-j)T] = \varepsilon^{-kAT}\varepsilon^{(k-1-j)AT}$$

$$= \varepsilon^{-(j+1)AT}$$

$$\Phi(-kT)\Phi[(k-1-j)T] = \Phi[-(j+1)T] \qquad 3.15$$

Therefore, Equation 3.12 may be given as

$$\Phi(-kT)\underline{x}(kT) - \underline{x}(0) = \sum_{j=0}^{k-1} \Phi[-(j+1)T]\underline{d}(T)m(jT) \qquad 3.16$$

Now define

$$\underline{p}(l) = \Phi(-lT)\,\underline{d}(T) \qquad 3.17$$

and

$$\underline{q}(k,i) = \Phi[(i-k)T]\underline{x}(kT) - \underline{x}(i) \qquad 3.18$$

Substituting Equations 3.17 and 3.18, Equation 3.16 becomes

$$\underline{q}(k,0) = \sum_{j=0}^{k-1} \underline{p}(j+1)\ \underline{m}\ (jT) \qquad 3.19$$

For a linear plant of order n, Equation 3.19 represents a set of n simultaneous linear equations with k unknowns, where k=n. The sequence of unknowns $\underline{m}(jT)$, j=0, 1, ..., k-1, is the set of control signals required to produce state transition from $\underline{x}(0)$ to $\underline{x}(kT)$ in at least n sampling periods.

Now Equation 3.19 may be expressed as follows:

$$\begin{bmatrix} q_1(k,0) \\ q_2(k,0) \\ \vdots \\ q_n(k,0) \end{bmatrix} = \begin{bmatrix} p_1(1) & p_1(2) & \cdots & p_1(k) \\ p_2(1) & p_2(2) & \cdots & p_2(k) \\ \vdots & \vdots & & \vdots \\ p_n(1) & p_n(2) & \cdots & p_n(k) \end{bmatrix} \begin{bmatrix} m(0) \\ m(T) \\ \vdots \\ m[(k-1)T] \end{bmatrix} \qquad 3.20$$

or

$$\underline{q} = P\ \underline{m} \qquad 3.21$$

If the matrix P is determined to be non-singular, then the control vector $\underline{m}$ may be found by

$$\underline{m} = P^{-1}\ \underline{q} \qquad 3.22$$

The error vector $\underline{e}(kT)$ input to the discrete compensator is now characterized. Defining β_1 as a scalar and β_2 as a row vector relating the error to the system input and the state vector, respectively, the error may be expressed as

$$e(kT) = \beta_1 r(kT) + \beta_2 \underline{x}(kT) \qquad 3.23$$

Equation 3.23 yields an error vector $\underline{e}$, with elements $e(jT)$ for $j = 0, 1, \ldots, k-1$.

The discrete compensator input and output vectors developed in this subsection are used to produce the discrete compensator realization in the following subsection.

Realization. In order to obtain the discrete compensator realization, the following pulse transfer function is formed (Dorf, 1965:136):

$$\frac{M(T)}{E(T)} = \frac{\sum_{j=0}^{k-1} m(jT)\ \delta\ (t-jT)}{\sum_{j=0}^{k-1} e(jT)\ \delta\ (t-jT)} , \tag{3.24}$$

where $\delta(t-jT)$ is a unit impulse occurring at time $t = jT$.

A general linear difference equation which allows for storing and weighting the present and past values of the compensator input and output pulses may be written as

$$m(kT) = \sum_{l=0}^{n} a_l e(kT-lT) - \sum_{l=1}^{n} b_l m(kT-lT) \tag{3.25}$$

The z-transform of Equation 3.25 yields the ratio of polynomials (Dorf, 1965:93)

$$\frac{M(z)}{E(z)} = \frac{\sum_{i=0}^{n} a_i z^{-i}}{1+\sum_{i=1}^{n} b_i z^{-i}} \tag{3.26}$$

As $Z^{-1}\{z^{-m}\} = \delta(t-mT)$, the similarity of Equations 3.26 and 3.24 is evident. E. W. Henry (1960:28) derived a realization of the discrete compensator pulse transfer function, expressed as a ratio of polynomials in z, in the following manner.

Henry states that the generalized discrete compensator shown in Figure 3.3 has the transfer function of Equation 3.26.

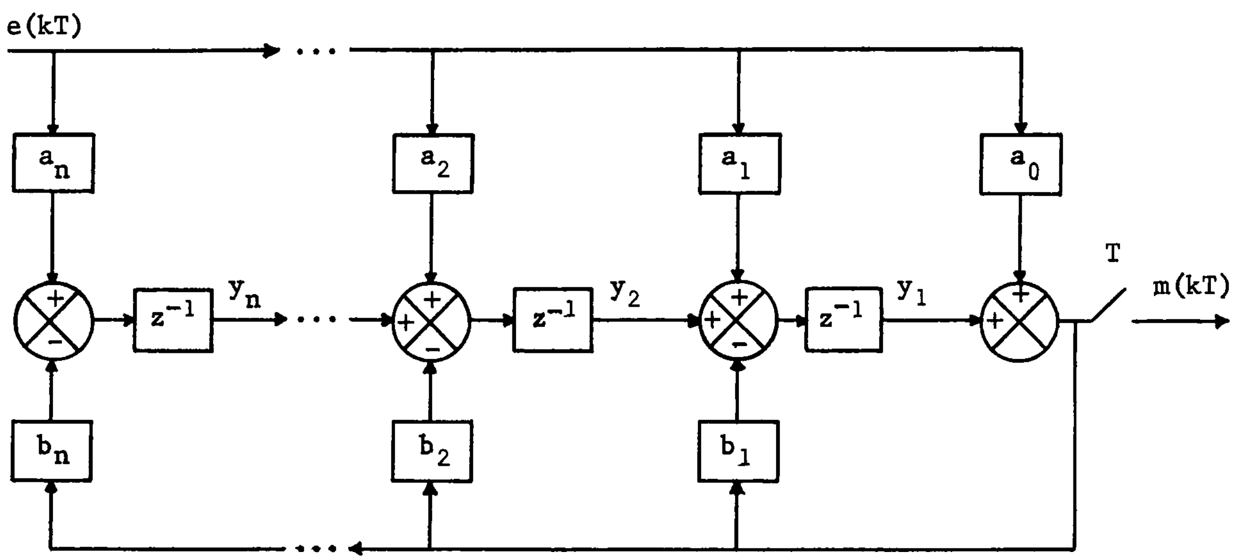

Figure 3.3

HENRY'S FORM OF DISCRETE COMPENSATOR

The special case of n=2 will be used in the derivation. By inspection of Figure 3.3,

$$m = a_0 e + y_1 \qquad 3.27$$

$$y_1 = (a_1 e - b_1 m + y_2)\ z^{-1} \qquad 3.28$$

$$y_2 = (a_2 e - b_2 m)\ z^{-1} \qquad 3.29$$

Substituting Equation 3.29 into Equation 3.28,

$$y_1 = (a_1 e - b_1 m)\ z^{-1} + (a_2 e - b_2 m)\ z^{-2} \qquad 3.30$$

and substituting Equation 3.30 into Equation 3.27,

$$m = a_0 e + (a_1 e - b_1 m)\ z^{-1} + (a_2 e - b_2 m)\ z^{-2} \qquad 3.31$$

Collecting terms of m and e,

$$m(1 + b_1 z^{-1} + b_2 z^{-2}) = e(a_0 + a_1 z^{-1} + a_2 z^{-2}) \qquad 3.32$$

Therefore,

$$\frac{m}{e} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \qquad 3.33$$

As the compensator form shown in Figure 3.3 is completely modular, Equation 3.33 may be extended by inspection to the general transfer function of Equation 3.26. Thus, a realization of the discrete compensator has been effected.

Continuous Compensator

Design procedure. This subsection deals with the design of a classical continuous compensator. In order to facilitate comparison in Chapter IV of the discrete compensator and the continuous compensator, Example 2 in the Meksawan and Murphy (1963) paper was chosen for the continuous compensator development.

Figure 3.4 illustrates the control system configuration to be considered.

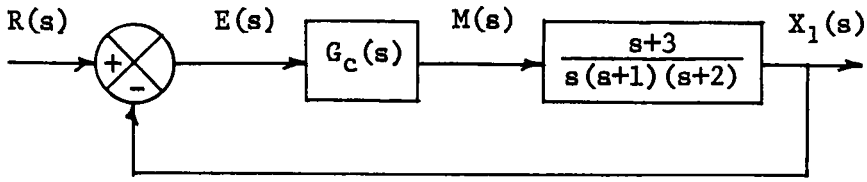

Figure 3.4

CONTINUOUS-COMPENSATED SYSTEM

The linear plant has the transfer function

$$G(s) = \frac{s+3}{s(s+1)(s+2)} \quad , \qquad 3.34$$

and the general compensator transfer function is

$$G_c(s) = \frac{N(s)}{D(s)} \qquad 3.35$$

Now the standard root locus method (D'Azzo and Houpis, 1966) will be used in order to design the compensator G_c for small rise time, small overshoot, and small settling time of the output X_1 in response to a step input R.

The root locus is a plot of the roots of the characteristic equation of the closed-loop system as a function of the gain. Thus the root locus may be used to determine the effect of compensator gain adjustment and compensator pole-zero placement upon the closed-loop system transient response.

In order to identify a compensator of minimum complexity which provides acceptable transient response, first consider only constant-gain compensation. The characteristic equation of the closed-loop system shown in Figure 3.4 is

$$s(s+1)(s+2)\,D(s) + (s+3)\,N(s) = 0 \qquad 3.36$$

For constant-gain compensation, N(s) = K and D(s) = 1; therefore, Equation 3.36 may be rewritten as

$$s^3 + 3s^2 + (2+K)\,s + 3K = 0 \qquad 3.37$$

The roots of Equation 3.37 were calculated for a range of K by the use of a FORTRAN subroutine, POLRT (IBM, 1968a:181). Figure 3.5 illustrates the corresponding root locus plot. Examination of the root locus for increasing values of gain K reveals the fact that constant-gain compensation gives unsatisfactory transient response. The closed-loop system becomes increasingly oscillatory as the complex root pair approaches the $j\omega$-axis.

The lead network is a compensator of the next order of complexity which may produce acceptable transient response in the system configuration of Figure 3.4. The general lead network transfer function is

$$\frac{N(s)}{D(s)} = \frac{K(s+a)}{s+b} \qquad 3.38$$

Therefore, the characteristic equation of the lead-network compensated, closed-loop system becomes

$$s(s+1)(s+2)(s+b) + K(s+a)(s+3) = 0 \qquad 3.39$$

The lead-network compensator design procedure of D'Azzo and Houpis (1966:411) suggests the following placement of the compensator pole and zero. The linear plant G(s) of Equation 3.34 is Type 1 (has a denominator term of the form s^n, where n=1). In such a case, the

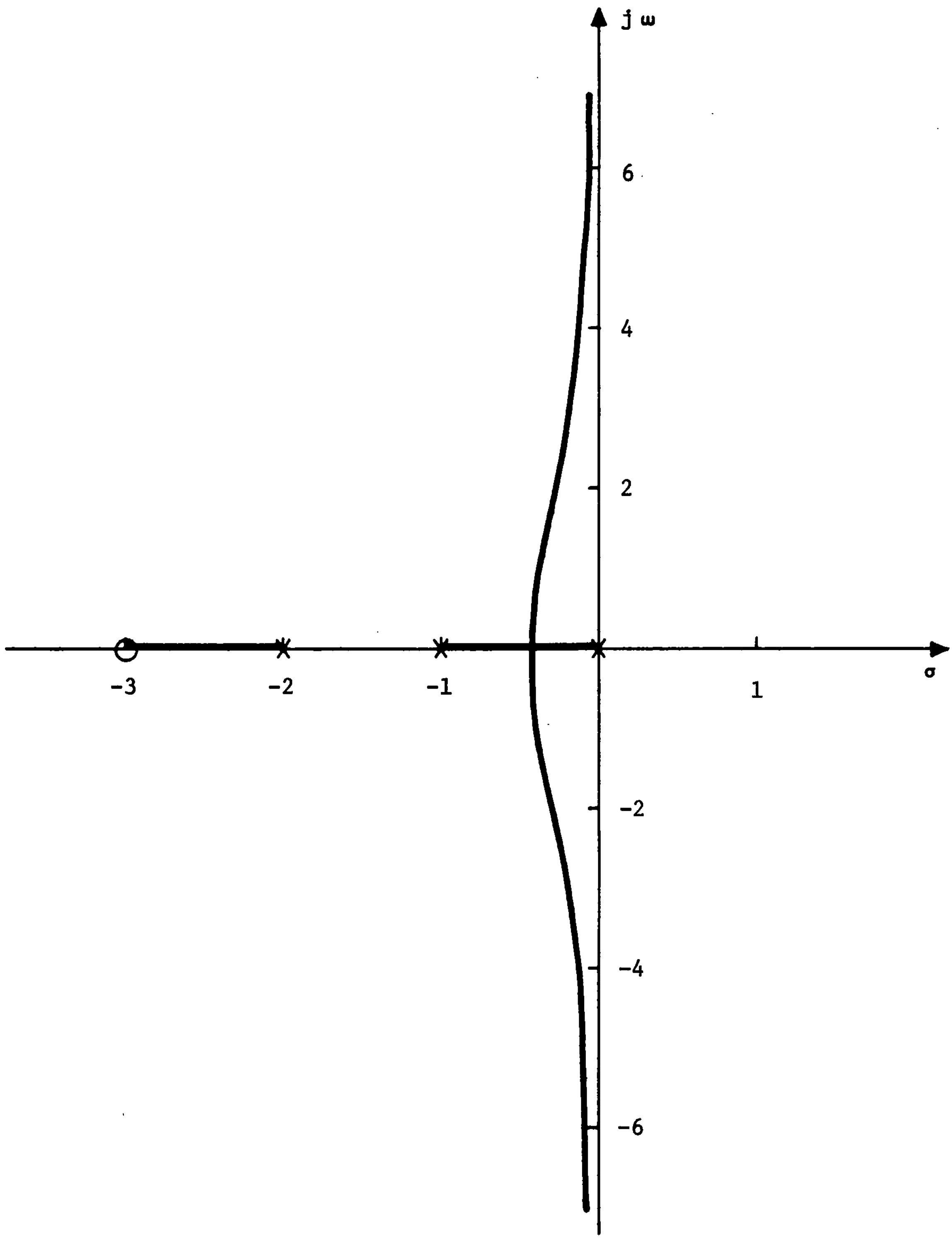

Figure 3.5

ROOT LOCUS FOR CONSTANT-GAIN COMPENSATION compensator zero is chosen to cancel the most positive real pole of G(s), excluding the pole at s=0. The compensator pole is chosen such that its effect upon the transient response is minimized (its corresponding real exponential term decays rapidly with increasing time). Thus, for a=1 and b=10, Equation 3.39 is written as

$$s^4 + 13s^3 + (32+K)s^2 + (20+4K)s + 3K = 0 \qquad 3.40$$

The subroutine POLRT (IBM, 1968a:181) was again employed in order to calculate the roots of Equation 3.40 for varying K. The root locus plot for the lead-network compensated system is shown in Figure 3.6.

In order to specify a desired value of gain K, additional constraints must be placed upon the compensator design. These constraints may be expressed in terms of peak overshoot and settling time, as the lead-network compensated system may be approximated by a second-order system. Referring to the root locus in Figure 3.6, the second-order approximation is possible due to the fact that, for increasing values of gain K, the pole at s = -10 moves toward the zero at s = -3, producing a cancellation effect. Also, the poles at s = 0 and s = -2 break away from the σ-axis, becoming dominant complex poles for increasing values of gain K.

The second-order approximation is of the form (D'Azzo and Houpis, 1966:243)

$$\frac{X_1(s)}{R(s)} = \frac{L}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad 3.41$$

The dominant complex poles of the lead-network compensated system correspond to the roots of the denominator of the right-hand side of Equation

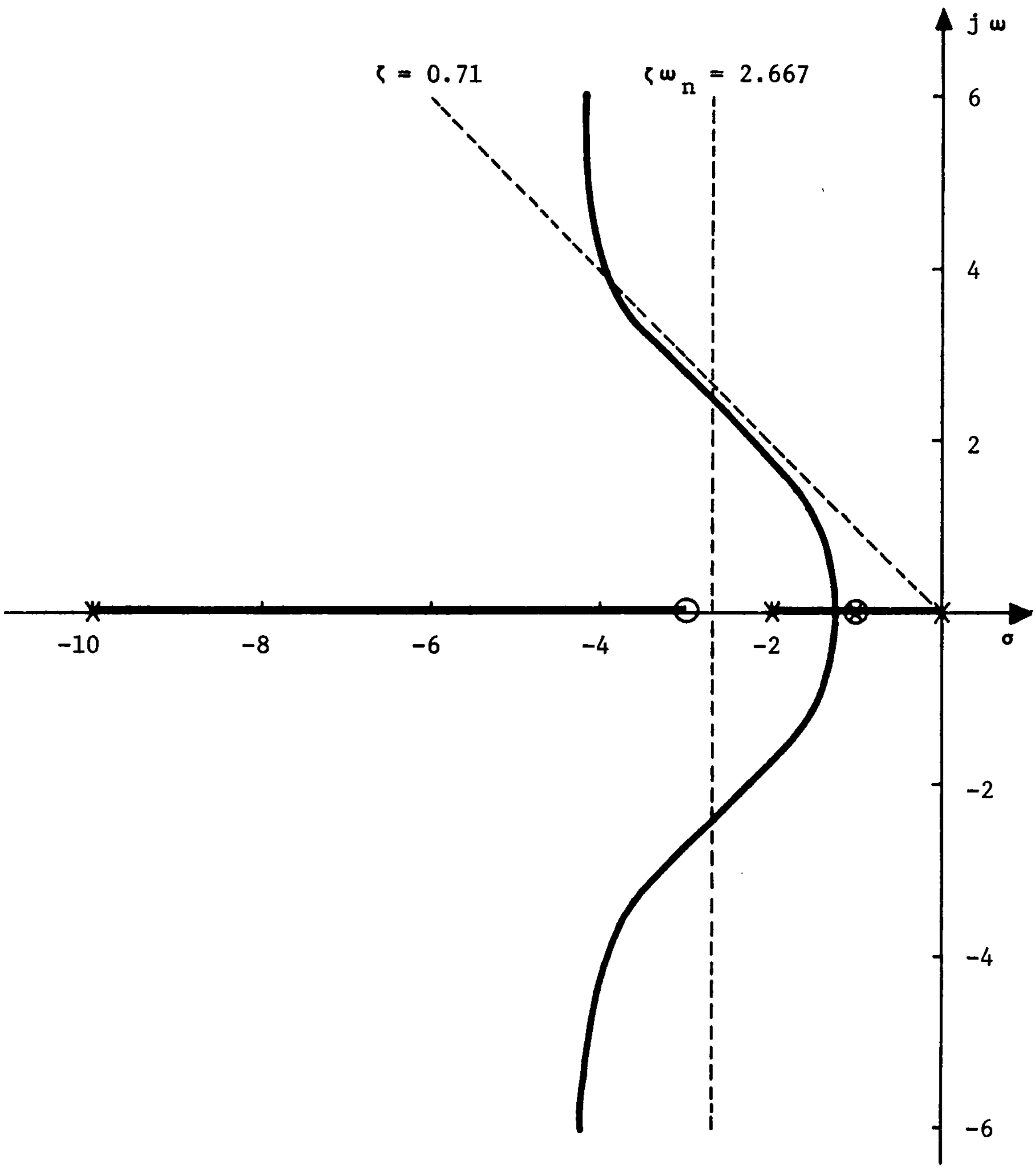

Figure 3.6

ROOT LOCUS FOR LEAD NETWORK COMPENSATION

3.41,

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n \sqrt{1-\zeta^2} \quad , \qquad 3.42$$

where $0 \leq \zeta \leq 1$. Now, the settling time T_s for ±2 per cent error in the second-order system is approximately four time constants (D'Azzo and Houpis, 1966:83). Therefore,

$$T_s = \frac{4}{\zeta\omega_n} \qquad 3.43$$

The peak overshoot of the second-order system in response to a unit step input is given as

$$M_p = 1 + \varepsilon^{-\zeta\pi/\sqrt{1-\zeta^2}} \qquad 3.44$$

In order to obtain small overshoot while maintaining small rise time (D'Azzo and Houpis, 1966:81), the damping ratio is chosen as $\zeta = 0.71$. Substitution of the value $\zeta = 0.71$ into Equation 3.44 yields the value $M_p = 1.042$. The representation of $\zeta = 0.71$ on the root locus plot of Figure 3.6 is a radial line from the origin, forming an angle η with the σ-axis such that

$$\eta = \cos^{-1} \zeta \qquad 3.45$$

Evaluating Equation 3.45 for $\zeta = 0.71$ gives the value $\eta = 44.77$ degrees.

Now Example 2, Meksawan and Murphy (1963) has n=3 state-variables and sample period T = 0.5. From Equation 3.19, the settling time of this discrete-compensated system is

$$T_{sd} = nT \qquad 3.46$$

Evaluating Equation 3.46 for n=3 and T = 0.5 yields the value T_{sd} = 1.5. Assuming that the settling time of the lead-network compensated system is required to be less than or equal to the settling time of the discrete-compensated system,

$$T_s \leq T_{sd} \qquad 3.47$$

Substituting Equations 3.43 and 3.46 into Equation 3.47 and evaluating,

$$\frac{4}{\zeta\omega_n} \leq nT$$

$$\frac{4}{\zeta\omega_n} \leq 1.5$$

Therefore, $\zeta\omega_n \geq 2.667$. This relation is represented on the root locus plot of Figure 3.6 as that area to the left of the vertical line $\sigma = -2.667$.

The intersection of the root locus and the radial line $\zeta = 0.71$ gives K = 42.2. This value of gain K yields $\zeta\omega_n = 3.800$, thus satisfying the requirement $\zeta\omega_n \geq 2.667$.

The lead-network compensator which gives peak overshoot $M_p = 1.042$ and settling time $T_s < 1.5$ for the second-order system approximation is now expressed in the form of Equation 3.38 as

$$G_c(s) = \frac{42.2(s+1)}{s+10} \qquad 3.48$$

<u>Realization</u>. The continuous compensator transfer function of Equation 3.48 will be realized in this subsection using linear, passive network elements. Consider the network shown in Figure 3.7 (D'Azzo and Houpis, 1966:159-160).

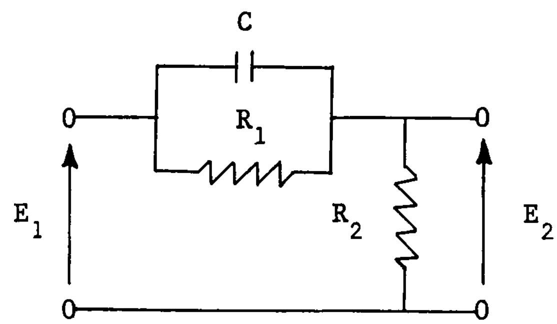

Figure 3.7

LINEAR, PASSIVE NETWORK

The output voltage $E_2(s)$ is

$$E_2(s) = \frac{R_2 E_1(s)}{R_2 + \dfrac{\dfrac{R_1}{Cs}}{R_1 + \dfrac{1}{Cs}}} \qquad 3.49$$

Taking the ratio of voltages and collecting like terms,

$$\frac{E_2(s)}{E_1(s)} = \frac{R_2}{R_1 + R_2} \; \frac{1 + R_1Cs}{1 + \dfrac{R_2}{R_1 + R_2} R_1Cs} \qquad 3.50$$

Let $\alpha = \dfrac{R_2}{R_1 + R_2}$ and $\beta = R_1C.$

Then Equation 3.50 may be rewritten as

$$\frac{E_2(s)}{E_1(s)} = \alpha \frac{1 + \beta s}{1 + \alpha\beta s} \qquad 3.51$$

or, in another form,

$$\frac{E_2(s)}{E_1(s)} \qquad \frac{s + \frac{1}{\beta}}{s + \frac{1}{\alpha\beta}} \tag{3.52}$$

Equation 3.52 is now recognized as the dynamic portion of Equation 3.38 with $a = 1/\beta$ and $b = 1/\alpha\beta$. For $s = 0$, the static gain of the transfer function of Equation 3.52 is α. From Equations 3.48 and 3.52, $\alpha = 0.1$ and $\beta = 1$. Substituting Equation 3.51 into Equation 3.48 and evaluating,

$$G_c(s) = 4.22 \ \frac{1 + s}{1 + .1s} \tag{3.53}$$

Now $\alpha = R_2/(R_1+R_2) = 0.1$ and $\beta = R_1C = 1$. Choose C = 1 microfarad, giving R_1 = 1 megohm. Evaluating $R_2/(1 \times 10^6 + R_2) = 1 \times 10^{-1}$ gives R_2 = 1/9 megohm.

The gain factor of 4.22 in Equation 3.53 may be realized by a linear voltage amplifier. Details of the amplifier design were not considered in this work.

Figure 3.8 illustrates the continuous compensator realization for the transfer function of Equation 3.48.

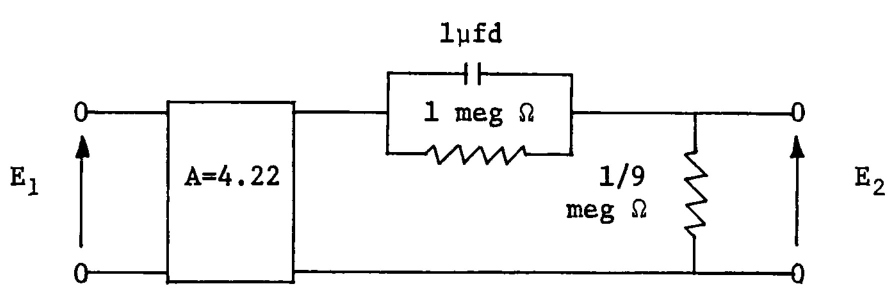

Figure 3.8

CONTINUOUS COMPENSATOR REALIZATION

Summary

Compensator design procedures have been developed for both the discrete case, using the design technique of Meksawan and Murphy and the realization due to Henry, and the continuous case, employing the root locus design method and the lead network compensator realization. The organization of the investigation utilizing these design and synthesis techniques is documented in Chapter IV.

CHAPTER IV

ORGANIZATION OF THE INVESTIGATION

The investigation associated with this work is logically constructed in two phases. The first of these phases, the design phase, contains a digital computer program implementation of the discrete compensator design technique developed in Chapter III. The second, or analysis phase, demonstrates a digital simulation method for verification of both the discrete and the continuous compensator designs. The analysis phase also includes consideration of the effect of saturation nonlinearity in the plant with respect to the discrete compensator design.

Design Phase

As the discrete compensator design method of Meksawan and Murphy is mathematically general and deals primarily with linear operations, a digital computer program implementation of this method is facilitated. FORTRAN (IBM, 1968b) was chosen as the source language because of its problem-oriented nature. Design program inputs and outputs are described in this section, and the internal structure of the program is detailed in the Appendix.

The required design program inputs and corresponding outputs are best described by a specific example. Example 2 from the Meksawan and Murphy (1963) paper is to be considered, as shown in Figure 4.1.

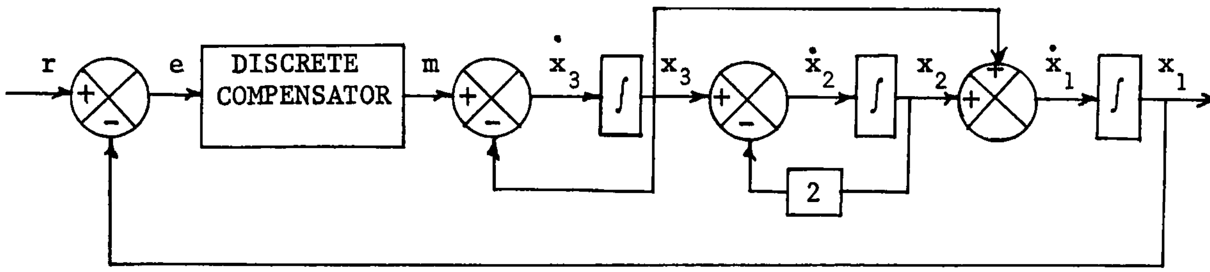

Figure 4.1

G(s) = (s+3)/s(s+1)(s+2), DISCRETE-COMPENSATED

The linear plant in Figure 4.1 is represented in state-variable format. The equivalence of this linear plant to the linear plant in Figure 3.4, page 18, may be shown by block-diagram reduction. In matrix format, the linear plant is described as

$$\dot{\underline{x}} = A\,\underline{x} + \underline{b}\,m,$$

or

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} m \qquad 4.1$$

with initial conditions

$$\underline{x}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad 4.2$$

Now the control problem specifies that the output state-variable x_1 is required to equal the unit step system input r in n sampling periods T, where n is the number of state-variables in the linear plant. Also, x_1 must reach the steady-state in n = 3 sampling periods, therefore $\dot{x}_1(nT) = 0$. For T = 0.5, the constraints on x_1 and $\dot{x}_1$ are expressed as

$$\underline{x}(nT) = \begin{bmatrix} x_1(1.5) \\ x_2(1.5) \\ x_3(1.5) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad 4.3$$

The error input e to the discrete compensator is written, by inspection of Figure 4.1, in the following manner:

$$e = \beta_1 r + \beta_2 \underline{x}$$

or

$$e = (1)r + [-1, 0, 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad 4.4$$

The system under consideration is now sufficiently defined for design program input.

The program accepts input, in the standard NAMELIST format, as follows:

<u>Record 1, SIZE</u>

NDIM - number of state-variables n in the linear plant

NINPUT - number of inputs to the system

<u>Record 2, PARM1</u>

PLANT - plant matrix A

X - initial conditions $\underline{x}(0)$

XQUIES - final conditions $\underline{x}(nT)$

EPSLN - error criterion ε for transition matrix series convergence (set arbitrarily small, 1×10^{-5})

Record 3, PARM2

B - vector $\underline{b}$ relating plant input m to plant state-variables

R - sign and magnitude of unit step system input r

INTOER - scalar β_1 relating system input r to error e

TRNDCR - row vector β_2 relating plant state vector $\underline{x}$ to error e

Record 4, PARM3

TIMPD - sample period T

LASTST - "last set" switch ($0 \Rightarrow$ another case follows, $1 \Rightarrow$ last case)

Output from the design program is routed to the line printer. The printed output of the plant matrix A, input vector $\underline{b}$, and the matrix Φ and vector $\underline{d}$ for the transition equation corresponding to Example 2 of Meksawan and Murphy is shown in Figure 4.2. The elements of the control vector $\underline{m}$ and the error vector $\underline{e}$ are displayed as the open-loop control sequence and the error sequence, respectively, in Figure 4.3. The final program output, upon completion of successful execution, is the message

"***** CONTROL SEQUENCES VALID FOR ANALYSIS PHASE *****".

The design-program calculated values for the matrix Φ and vector $\underline{d}$ agree exactly with the published values of Meksawan and Murphy. Furthermore, the calculated values for the sequences $\underline{m}$ and $\underline{e}$ are equal to the published values within the bounds of computer round-off error (Meksawan and Murphy, 1963:298).

A second example of discrete compensator design is now developed

DISCRETE COMPENSATION OF A STATE-VARIABLE
DEFINED LINEAR PLANT

SAMPLE PERIOD = 0.500

LINEAR PLANT MATRIX 'A'

0.0	1.000000	1.000000
0.0	-2.000000	1.000000
0.0	0.0	-1.000000

VECTOR 'B' RELATING EXTERNAL INPUTS TO PLANT STATES

0.0
0.0
1.000000

****TAYLOR'S SERIES CONVERGED IN 10 ITERATIONS****

TRANSITION MATRIX 'PHI'

1.000000	0.316060	0.470878
0.0	0.367879	0.238651
0.0	0.0	0.606531

VECTOR 'D' RELATING EXTERNAL INPUTS TO PLANT STATES

0.121091
0.077409
0.393469

Figure 4.2

DESIGN PROGRAM OUTPUT, PAGE 1

CONTROL SEQUENCES FOR HENRY'S
FORM OF THE DISCRETE COMPENSATOR

CASE NUMBER 1

THE LINEAR PLANT HAS 3 STATES

SAMPLE PERIOD = 0.500

OPEN-LOOP CONTROL SEQUENCE	ERROR SEQUENCE
5.36078	1.00000
-5.22360	0.35085
1.19615	-0.14099

Figure 4.3

DESIGN PROGRAM OUTPUT, PAGE 2

for the system illustrated in Figure 4.4. The linear plant transfer function,

$$G(s) = \frac{4}{s(s+1)(s+2)} , \qquad 4.5$$

is similar to G(s) for the previous example except for the absence of a zero at s = -3 and the presence of a constant factor of 4 in the numerator.

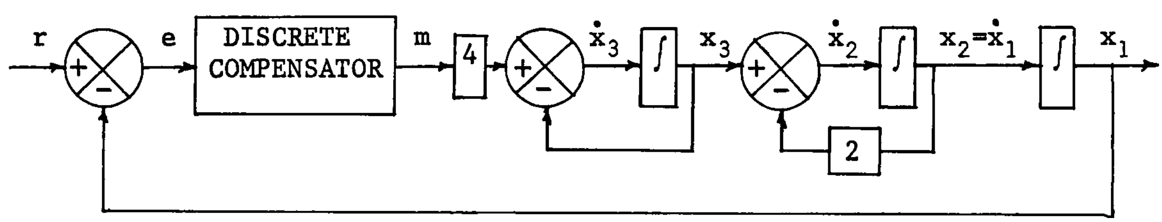

Figure 4.4

G(s) = 4/s(s+1)(s+2), DISCRETE-COMPENSATED

Discrete compensator coefficients were calculated to the specifications of Figure 4.4 for sample periods TIMPD = {1.0, 0.75, 0.5, 0.25, 0.1}. Table 4.1 enumerates the compensator coefficient sequences <u>m</u> and <u>e</u> for all sample periods considered.

Analysis Phase

System response. In the analysis phase of the investigation, both the compensator design procedures and the compensator realizations developed in Chapter III are verified. The demonstration of system performance in the time domain is effected employing Continuous System Modeling Program (CSMP) digital simulation (IBM, 1968c). CSMP allows programming of the linear plant with no compensation, discrete compensation,

Table 4.1

DISCRETE COMPENSATOR COEFFICIENTS FOR $G(s) = 4/s(s+1)(s+2)$

TIMPD = 1.0		TIMPD = 0.75	
m	e	m	e
0.91479	1.00000	1.62640	1.00000
-0.46034	0.69246	-1.13116	0.72947
0.04554	0.06904	0.17142	0.08806

TIMPD = 0.5		TIMPD = 0.25	
m	e	m	e
4.02059	1.00000	22.97933	1.00000
-3.91770	0.76583	-31.83402	0.80071
0.89711	0.11070	10.85469	0.13698

TIMPD = 0.1	
m	e
289.85529	1.00000
-499.58552	0.82060
214.73023	0.15441

or continuous compensation through the use of CSMP function blocks and FORTRAN-type source statements.

Figure 4.5 shows the time response of the output state-variable x_1 for a unit step input r to the system of Figure 4.1 with the compensator by-passed such that $m \equiv e$. The output response of the system of Figure 4.1 with the discrete compensator intact is illustrated in

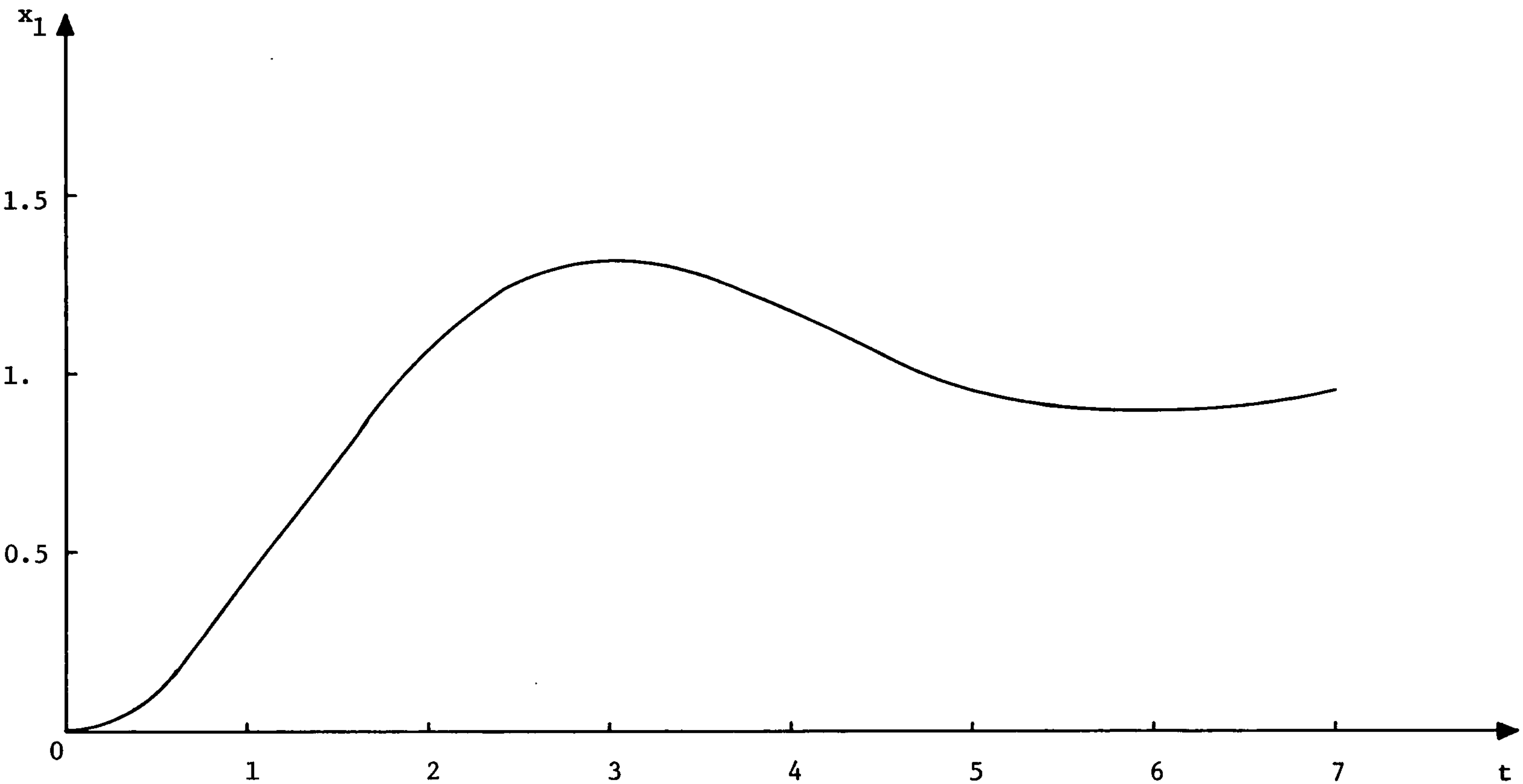

Figure 4.5

UNCOMPENSATED RESPONSE, $G(s) = (s+3)/s(s+1)(s+2)$

Figure 4.6. Figure 4.7 shows the output response of the system of Figure 4.1 with the discrete compensator displaced by the continuous compensator of Figure 3.8, page 26. Therefore, the uncompensated, discrete-compensated, and continuous-compensated versions of Example 2, Meksawan and Murphy may be compared using Figures 4.5, 4.6, and 4.7.

Numerical values of the response in Figure 4.6 are listed in Table 4.2 for sampling instants $t = nT$; $n = 0,1,2,3$; $T = 0.5$. The corresponding values from the Meksawan and Murphy paper are also shown.

Table 4.2

NUMERICAL RESPONSE VALUES FOR $G(s) = (s+3)/s(s+1)(s+2)$, DISCRETE-COMPENSATED

$t = nT$	x_1 CALCULATED	x_1 PUBLISHED
0	0.000	0
0.5	0.649	0.649
1.0	1.138	1.140
1.5	1.001	1

Figure 4.8 illustrates the response of the output x_1 for a unit step input r to the system of Figure 4.4 with the compensator by-passed such that $m \equiv e$. Figures 4.9 to 4.13, inclusive, present output response curves for the system of Figure 4.4 with sample periods TIMPD = {1.0, 0.75, 0.5, 0.25, 0.1}, respectively.

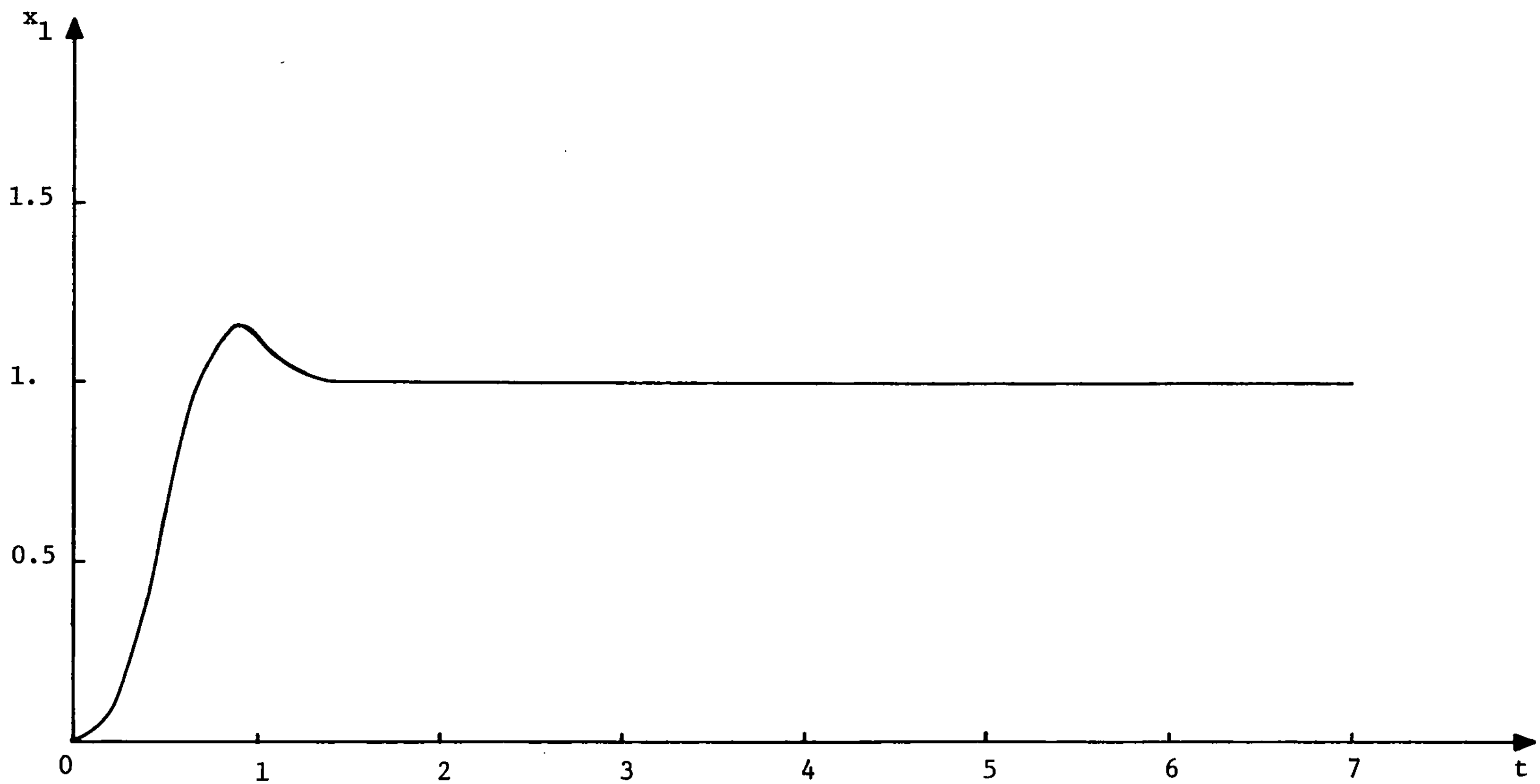

Figure 4.6

DISCRETE-COMPENSATED RESPONSE, $G(s) = (s+3)/s(s+1)(s+2)$

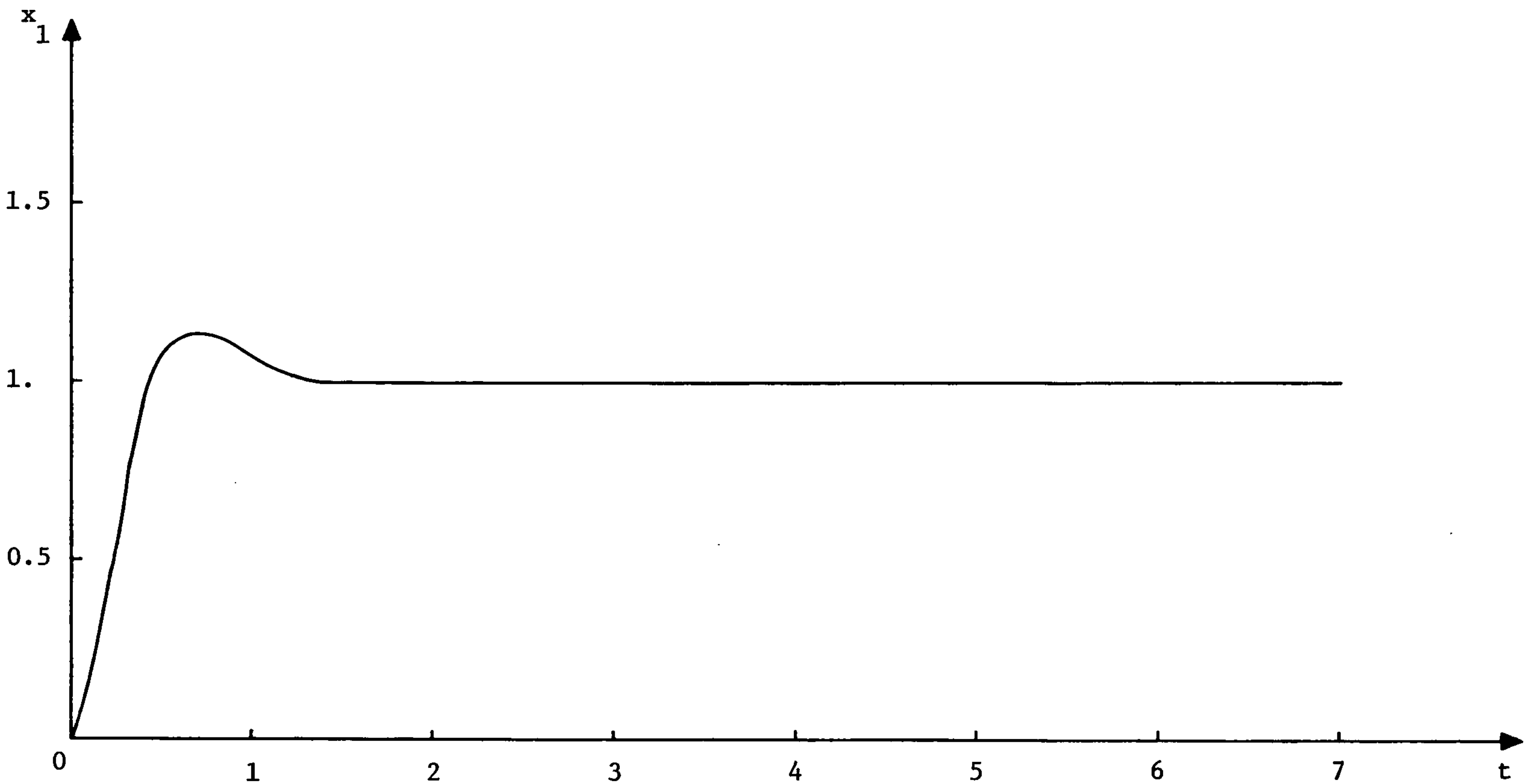

Figure 4.7

CONTINUOUS-COMPENSATED RESPONSE, G(s) = (s+3)/s(s+1)(s+2)

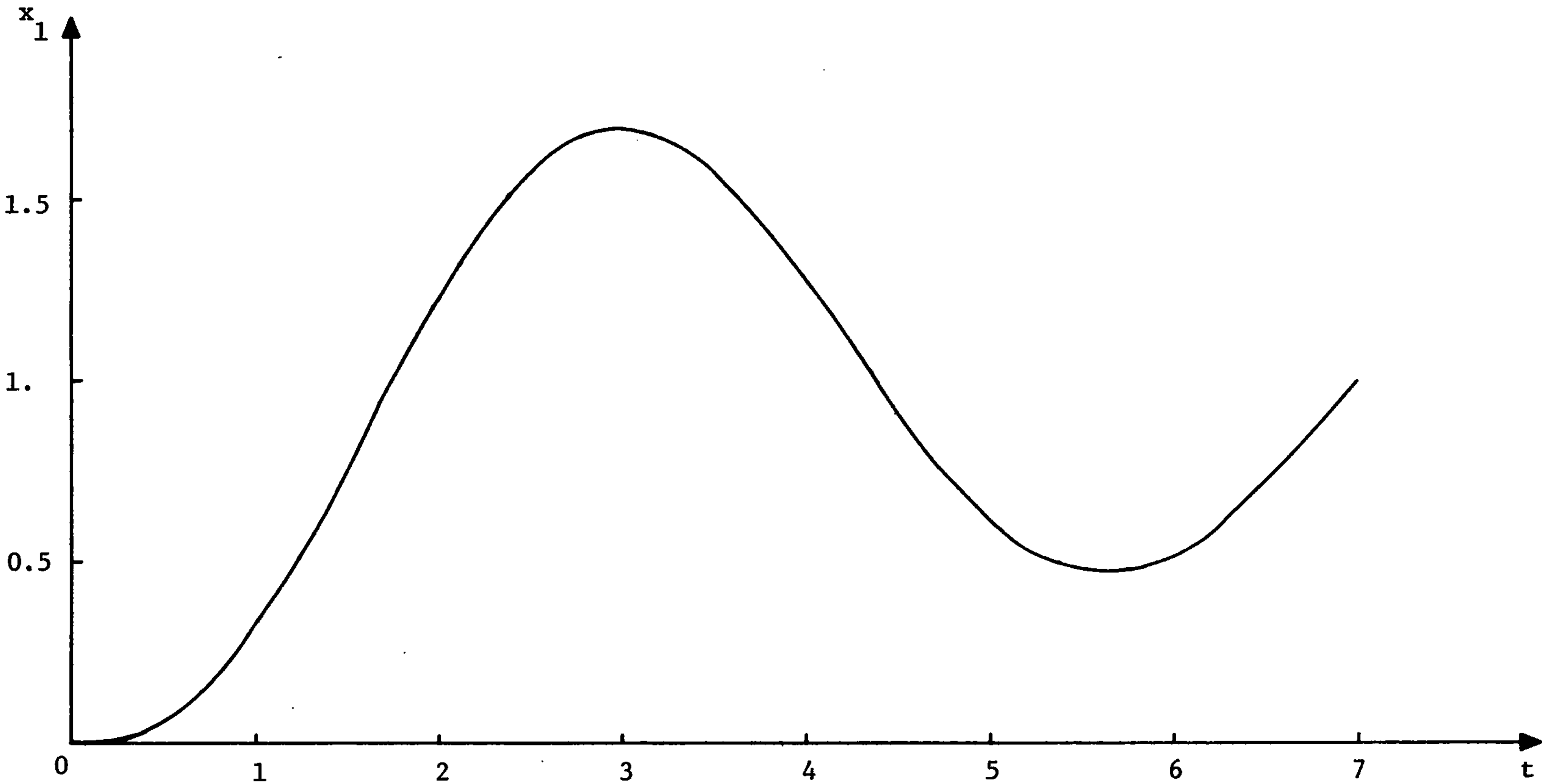

Figure 4.8

UNCOMPENSATED RESPONSE, $G(s) = 4/s(s+1)(s+2)$

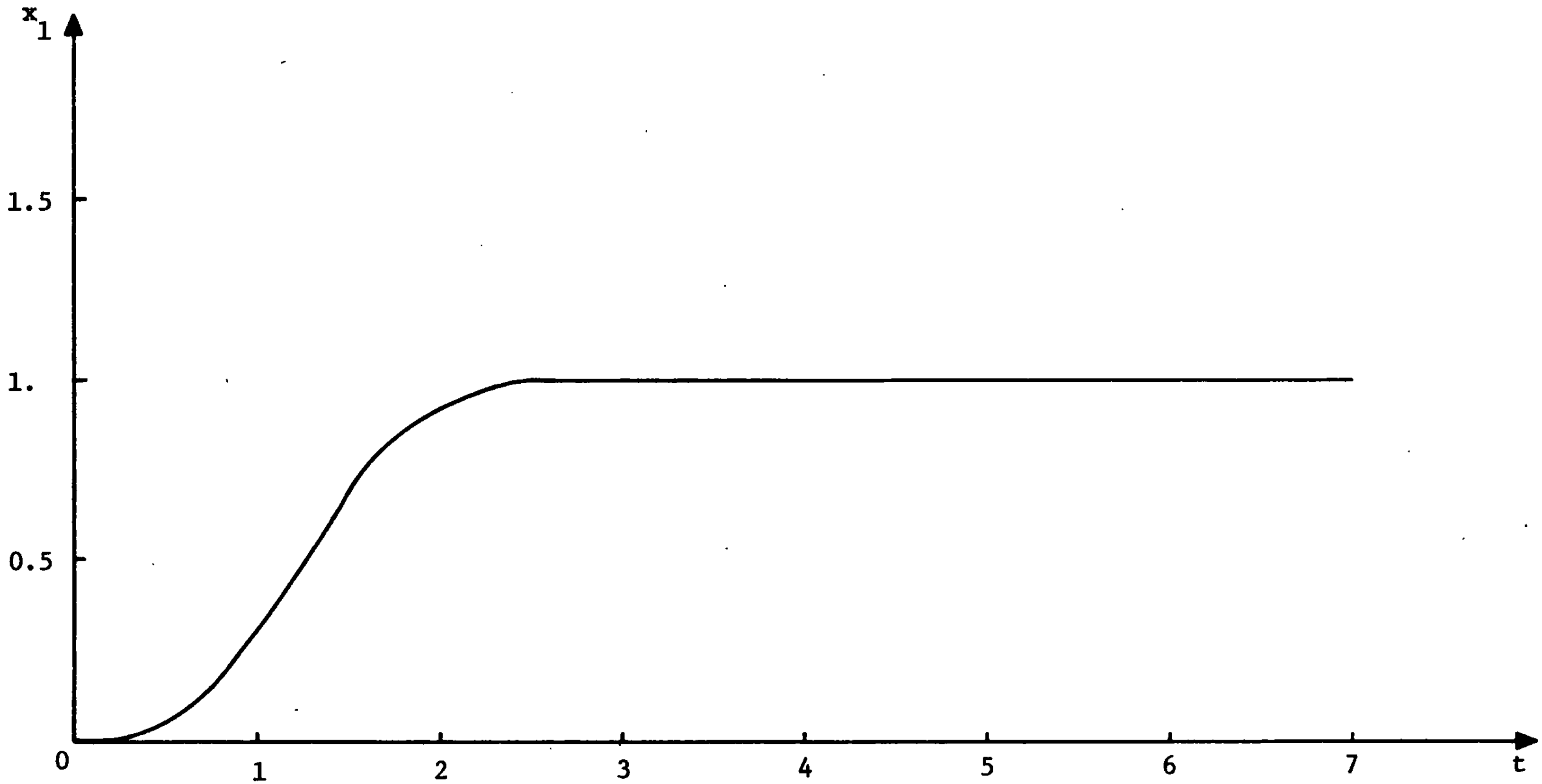

Figure 4.9

COMPENSATED RESPONSE, $G(s) = 4/s(s+1)(s+2)$,
TIMPD = 1.0

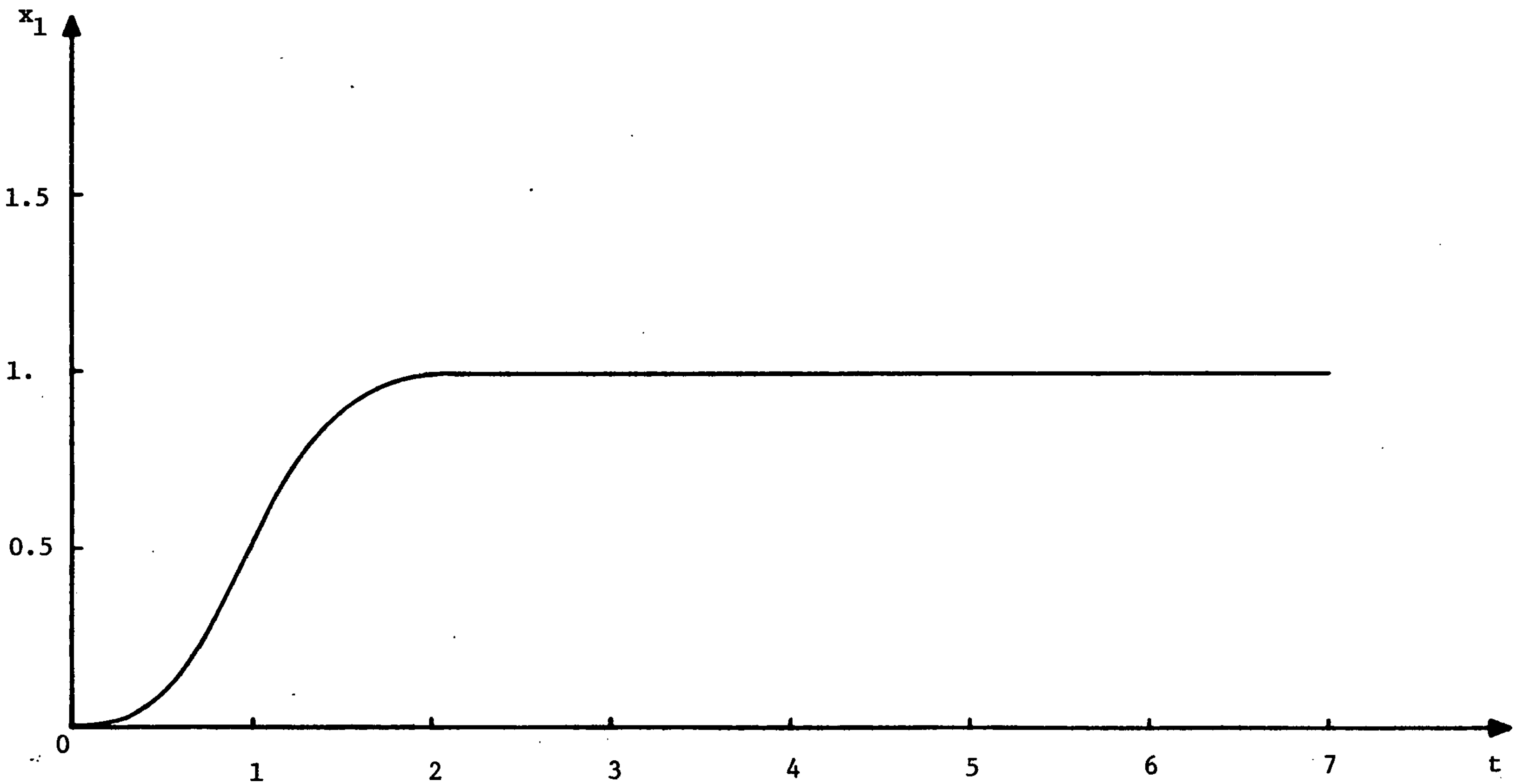

Figure 4.10

COMPENSATED RESPONSE, G(s) = 4/s(s+1)(s+2),
TIMPD = 0.75

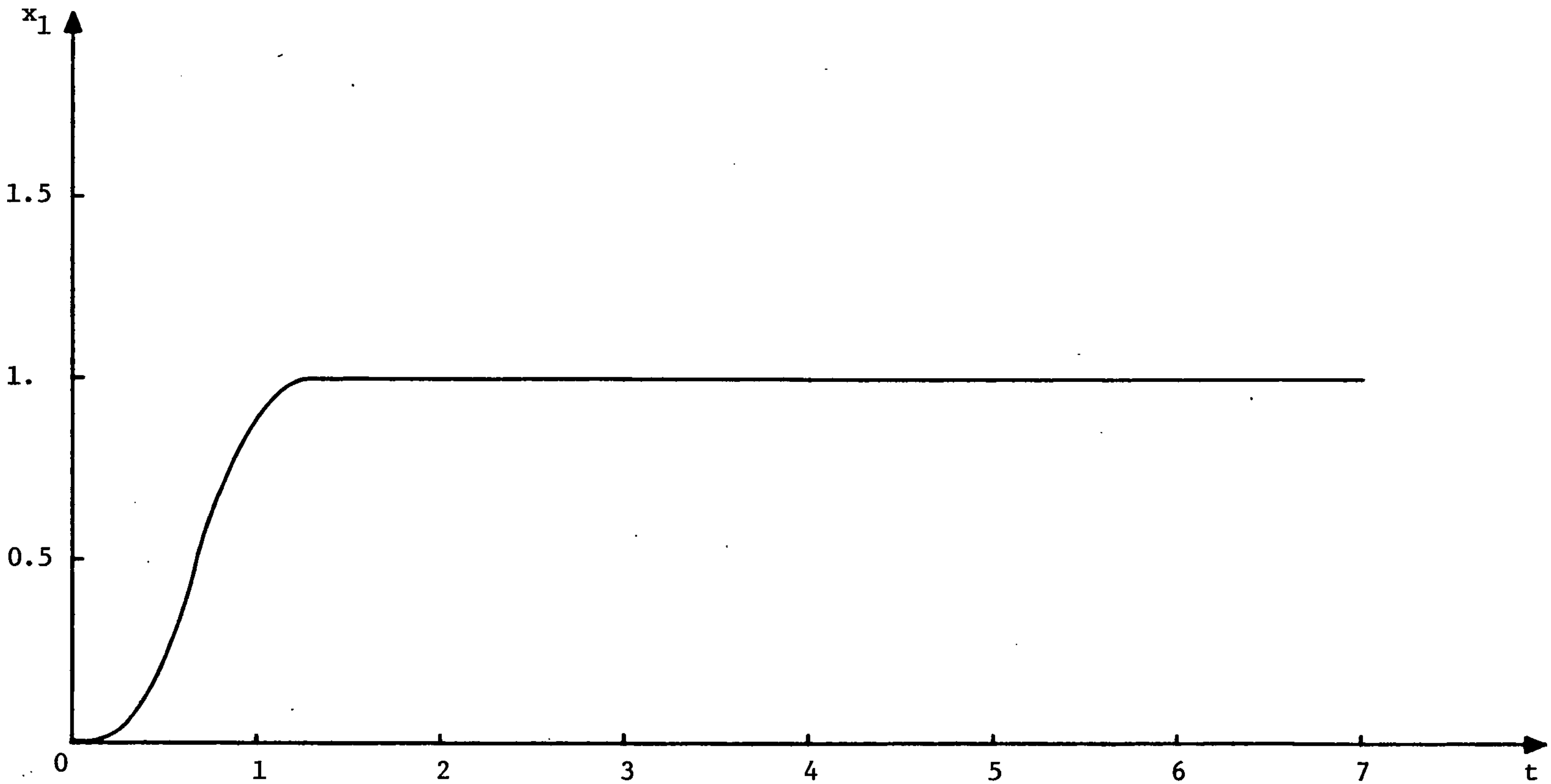

Figure 4.11

COMPENSATED RESPONSE, G(s) = 4/s(s+1)(s+2),
TIMPD = 0.5

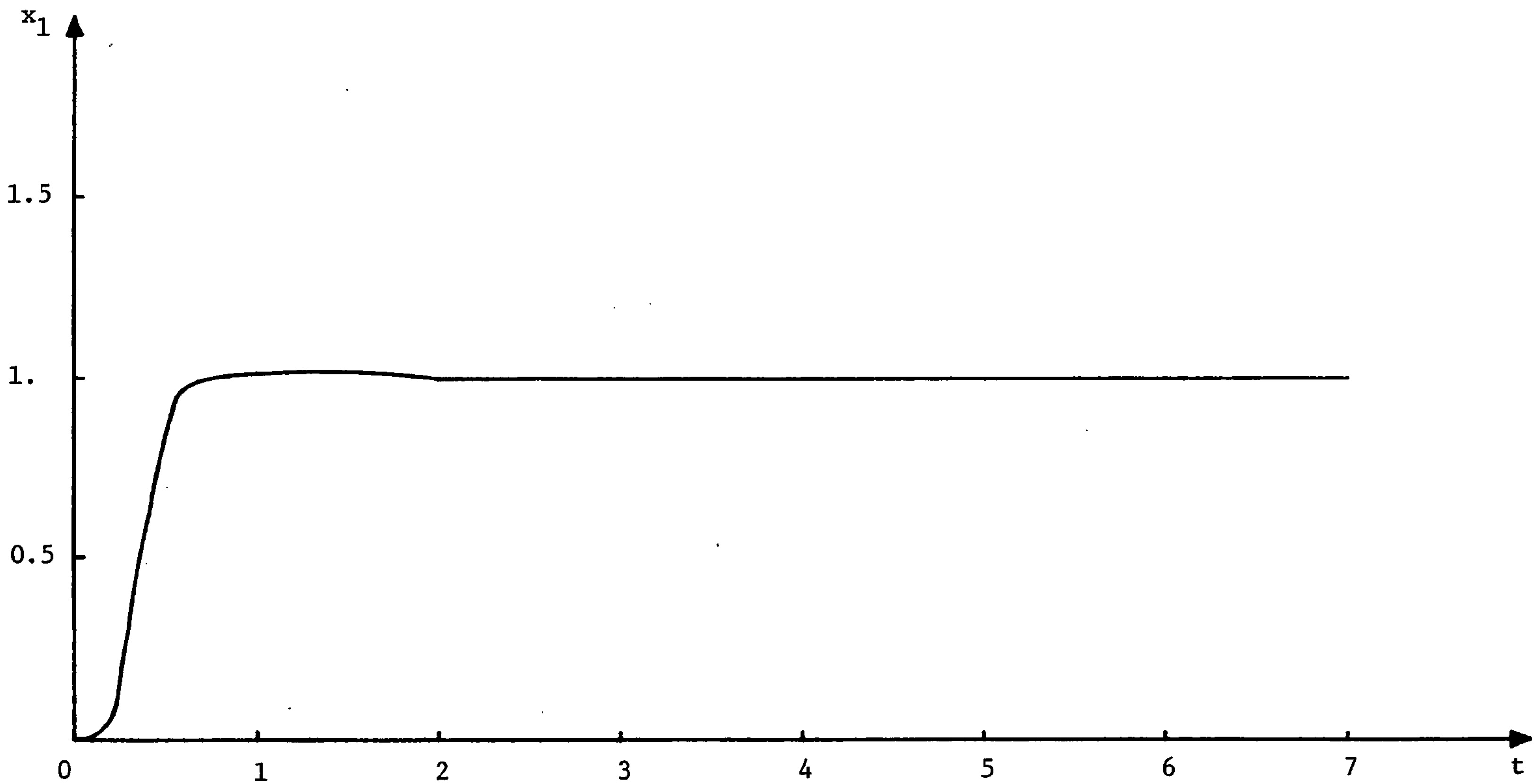

Figure 4.12

COMPENSATED RESPONSE, G(s) = 4/s(s+1)(s+2),
TIMPD = 0.25

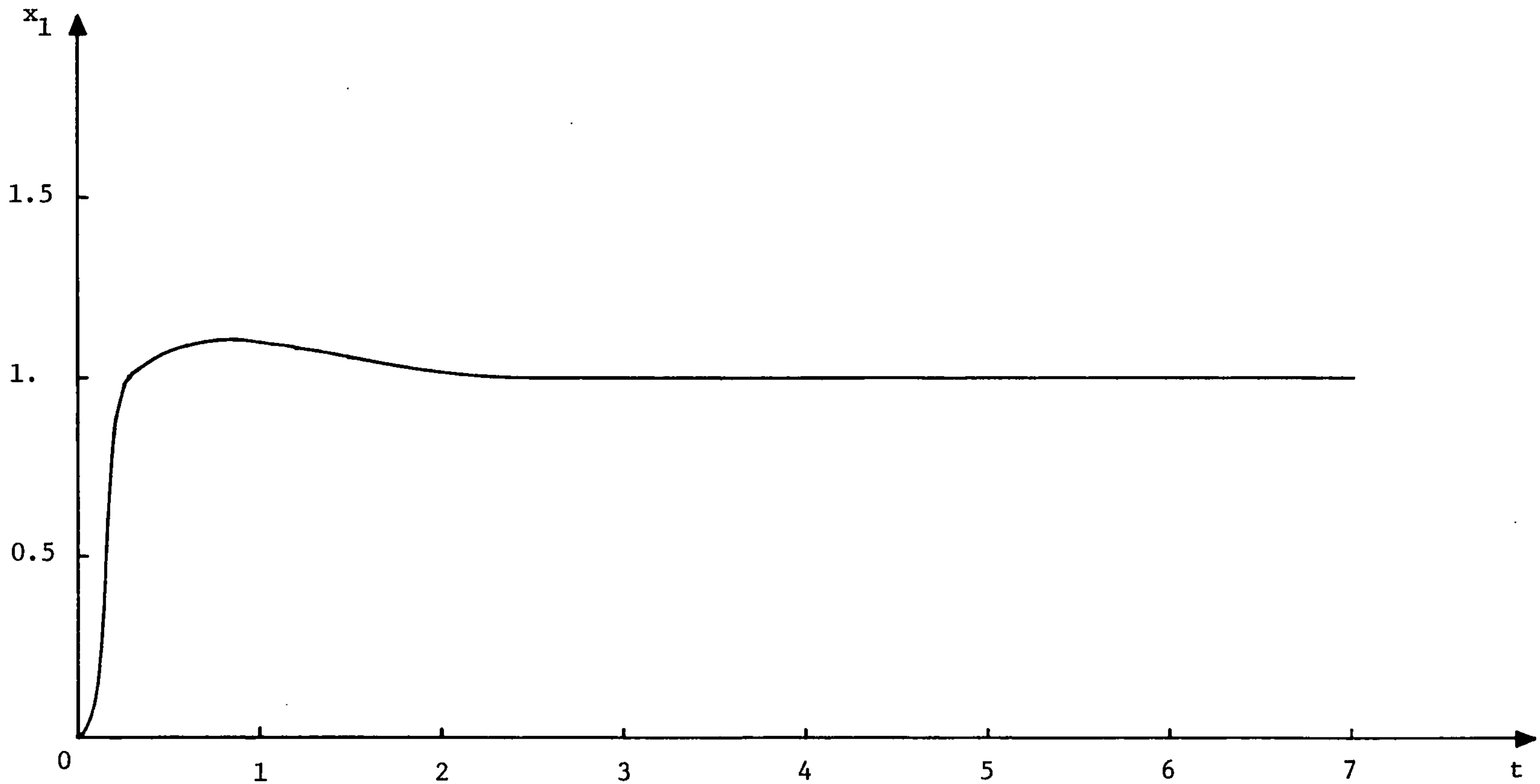

Figure 4.13

COMPENSATED RESPONSE, G(s) = 4/s(s+1)(s+2),
TIMPD = 0.1

Saturation nonlinearity. The design and analysis work has, to this point, been performed on systems containing linear plants. Actual systems such as those boiler-turbine-generator units described in Chapter I may rarely contain significant saturation nonlinearity. This nonlinearity is isolated in the electromechanical or hydraulic actuator for the turbine inlet valve train; therefore, the saturation characteristics may be lumped and placed at the compensator output, in cascade with the compensator and the linear plant (Kenny, 1967).

Consider the system of Figure 4.4, adding an ideal saturation nonlinearity with the transfer characteristic

$$m' = \begin{cases} -L, & m < -L \\ m, & -L \leq m \leq L \\ L, & m > L \end{cases}, \qquad 4.6$$

where m is the discrete compensator output and m′ is the linear plant input. Any term of the control sequence $\underline{m}$ greater than L in magnitude cannot be completely delivered to the linear plant. Examination of the control sequences $\underline{m}$ and corresponding sample periods from Table 4.1 shows that an inverse relationship exists between the value of sample period and the value of the term of greatest magnitude within the respective control sequence $\underline{m}$. This inverse relationship is explained by the fact that the time derivative of the state vector $\dot{\underline{x}}$ is directly proportional to the plant input m, and the value of the vector $\dot{\underline{x}}(t)$ for arbitrary time t_1 is inversely related to the value of the sample period. Therefore, the effect of a saturation nonlinearity as described by Equation 4.6 may be obviated by choice of a sample period of sufficient length.

Summary

The compensator design procedures and compensator realizations formulated in Chapter III have been programmed using FORTRAN and CSMP and have been verified by example from Meksawan and Murphy, as described in this chapter. A second example further exercised the discrete compensator design program and provided data for a range of sample periods, used in the consideration of a system containing saturation nonlinearity. Chapter V contains an evaluation of the investigation forming a portion of this work.

CHAPTER V

CONCLUSIONS

The results of the investigation described in Chapter IV clearly demonstrate the mathematical validity of both the discrete compensator design procedure of Meksawan and Murphy and the discrete compensator realization of Henry. As the discrete compensator design procedure is mathematically general and deals primarily with linear operations, a digital computer implementation using matrix algebra is facilitated. The discrete compensator realization is modular to the extent that delay and weighting sections may be added or subtracted in order to serve linear plants containing more or fewer state-variables, respectively. Another advantage of discrete compensation is the fact that control signal update is required only periodically. As the discrete compensator design procedure is formulated for a control problem identical to the basic load frequency control problem defined in Chapter I, this procedure could be integrated into a practical electric power system control configuration.

An element of the discrete compensator hypothesis worthy of further study is best expressed in the question: What characteristics of the plant matrix A and the input vector $\underline{b}$ would produce singularity in the matrix P? Additional study of the effect of other nonlinearity types upon discrete compensator applications is also recommended.

REFERENCES

Adibi, M. M. 1969. "Power System Computer Feasibility Study for the Department of Public Service, State of New York." Report of IBM Research, San Jose, California.

Cheng, David K. 1961. Analysis of Linear Systems. Reading: Addison-Wesley Publishing Co.

Cohn, N. 1957. "Some Aspects of Tie-Line Bias Control on Interconnected Power Systems," Transactions of the AIEE, LXXII, Part III (February), 1415-1436.

Courant, R., and D. Hilbert. 1965. Methods of Mathematical Physics, I (New York: Interscience Publishers, Inc.)

Crandall, S. H. 1956. Engineering Analysis. New York: McGraw-Hill Book Co.

D'Azzo, J. D., and C. H. Houpis. 1966. Feedback Control System Analysis and Synthesis. New York: McGraw-Hill Book Co.

Dorf, Richard C. 1965. Time Domain Analysis and Design of Control Systems. Reading: Addison-Wesley Publishing Co.

Elgerd, O. I., and C. E. Fosha. 1969. "The Megawatt-Frequency Control Problem--A New Approach Via Optimal Control Theory," Proceedings of the PICA Conference, Denver, Colorado, May 18-21, 123-132.

Frazer, R. A., W. J. Duncan, and A. R. Collar. 1963. Elementary Matrices. New York: Syndic of the Cambridge University Press.

Henry, E. W. 1969. "Logical Scheduling of a Multiplexed Digital Controller." Report of Technical Department 2106-1 on ONR Contract NONR 225(38), NR 049 132, Stanford University, Stanford, California, July 11.

IBM Corp. 1968. "System/360 Scientific Subroutine Package Programmer's Manual," Form H20-0205-3. IBM Technical Publications Dept., White Plains, New York.

IBM Corp. 1968. "System/360 FORTRAN IV Language Manual," File No. S360-25, Form C28-6515-7. IBM Program Publications, New York, New York.

IBM Corp. 1968. "System/360 Continuous System Modeling Program," Form H20-0367-2. IBM Technical Publications Dept., White Plains, New York.

Kenny, P. L. 1967. "Once-Through Boiler Control." Bailey Meter Co. Technical Paper TP 67-13.

Kirchmayer, L. K. 1958. Economic Operation of Power Systems. New York: John Wiley and Sons, Inc.

Kuo, Benjamin C. 1964. Analysis and Synthesis of Sampled-Data Control Systems. Englewood Cliffs: Prentice-Hall, Inc.

Meksawan, T., and G. J. Murphy. 1963. "Optimum Design of Non-Linear Sampled-Data Control Systems," Regelungstechnik, VII (January), 295-299.

Schultz, D. G., and J. L. Melsa. 1967. State Functions and Linear Control Systems. New York: McGraw-Hill Book Co.

APPENDIX

This appendix describes the internal structure of the discrete compensator design program used in the investigation as defined in Chapter IV. The design program is coded as a set of FORTRAN subroutines employing a simple main program for execution initiation.

In order for a program user to obtain a design, it is necessary to supply only the input data as specified in Chapter IV, and to start the procedure with a main program consisting of a single statement,

CALL DESIGN.

The subroutine DESIGN, with flowchart shown in Figure A-1, reads the input data, properly sequences other subroutine calls, and completes output of the discrete compensator solution.

The transition matrix $\Phi(t)$ is developed from the plant matrix A in the subroutine TRANS. As shown in the flowchart of Figure A-2, the transition matrix is evaluated utilizing Equation 2.20, page 8, as the Taylor's series expansion of the matrix exponential function ε^{At}. The criterion used to determine convergence of the Taylor's series is based upon the magnitude of change

$$\Big| |\Phi_{k+1}| - |\Phi_k| \Big| \leq \Big| \text{EPSLN}* \; |\Phi_{k+1}| \Big|$$

where $|\Phi_k|$ is the determinant of the matrix Φ_k consisting of the identity matrix plus k additional terms of the Taylor's series and EPSLN is an arbitrarily small positive constant. Several trials were performed using plant matrices A with known analytic solutions (Dorf, 1965) for ε^{At}. For a value of EPSLN = 1×10^{-5}, the given convergence criterion produced

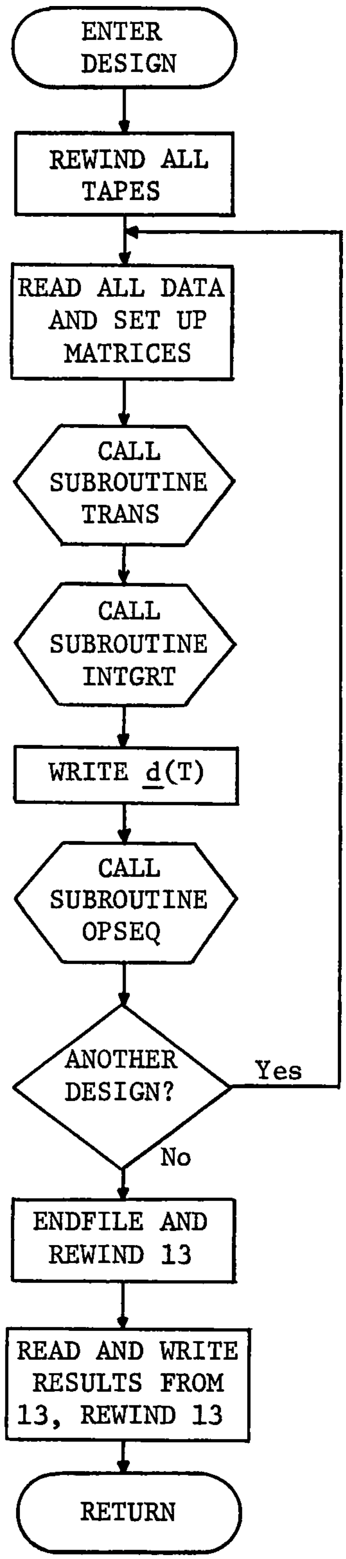

Figure A-1

SUBROUTINE DESIGN

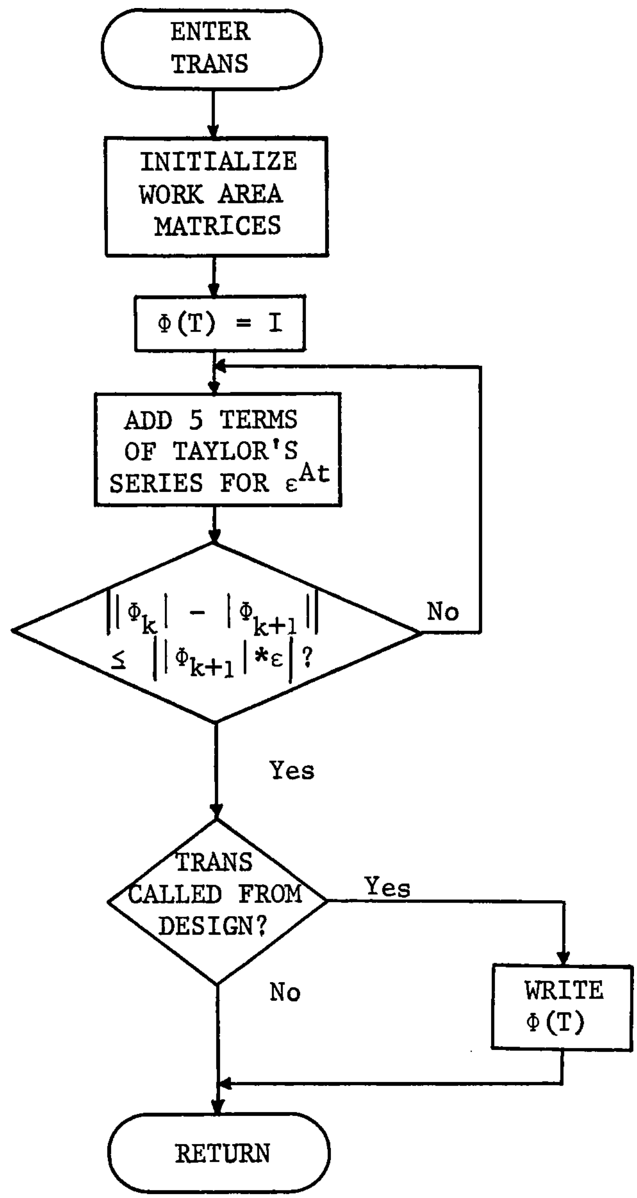

Figure A-2

SUBROUTINE TRANS

accuracy of four significant digits within the range from ten to twenty terms of the Taylor's series.

Figure A-3 illustrates the flowchart for the subroutine INTGRT. This subroutine is used to perform the numerical integration required by Equation 3.7, page 11, in order to develop the vector

$$\underline{d}(T) = \int_0^T \Phi(\gamma)\, \underline{b}\, d\gamma \ .$$

The integration is accomplished using a Simpson's Rule numerical approximation (Crandall, 1956). The integration interval $0 \leq t \leq T$ is partitioned into 100 subintervals of equal length ΔT. This partitioning yields

$$\int_0^T \Phi(\gamma)\underline{b}d\gamma \cong \frac{\Delta T}{3}\left[\underline{d}_0 + 4\sum_{k=1}^{50}\underline{d}_{(2k-1)\Delta T} + 2\sum_{l=1}^{49}\underline{d}_{2l\Delta T} + \underline{d}_{100\Delta T}\right] \ ,$$

where $\underline{d}_m = \Phi(m)\underline{b}$.

The subroutine OPSEQ, flowcharted in Figure A-4, calculates the control sequence $\underline{m}$ and the error sequence $\underline{e}$ utilizing both the discrete compensator design procedure of Meksawan and Murphy, and the discrete compensator realization requirements of Henry, as formulated in Chapter III.

All of the matrix operations required in the design program are performed by additional FORTRAN subroutines, as follow:

ZERO - generates an $m \times n$ null matrix

MADD - forms the sum of two $m \times n$ matrices

MTMUL - forms the scalar product of a $1 \times m$ row vector and a $m \times 1$ column vector

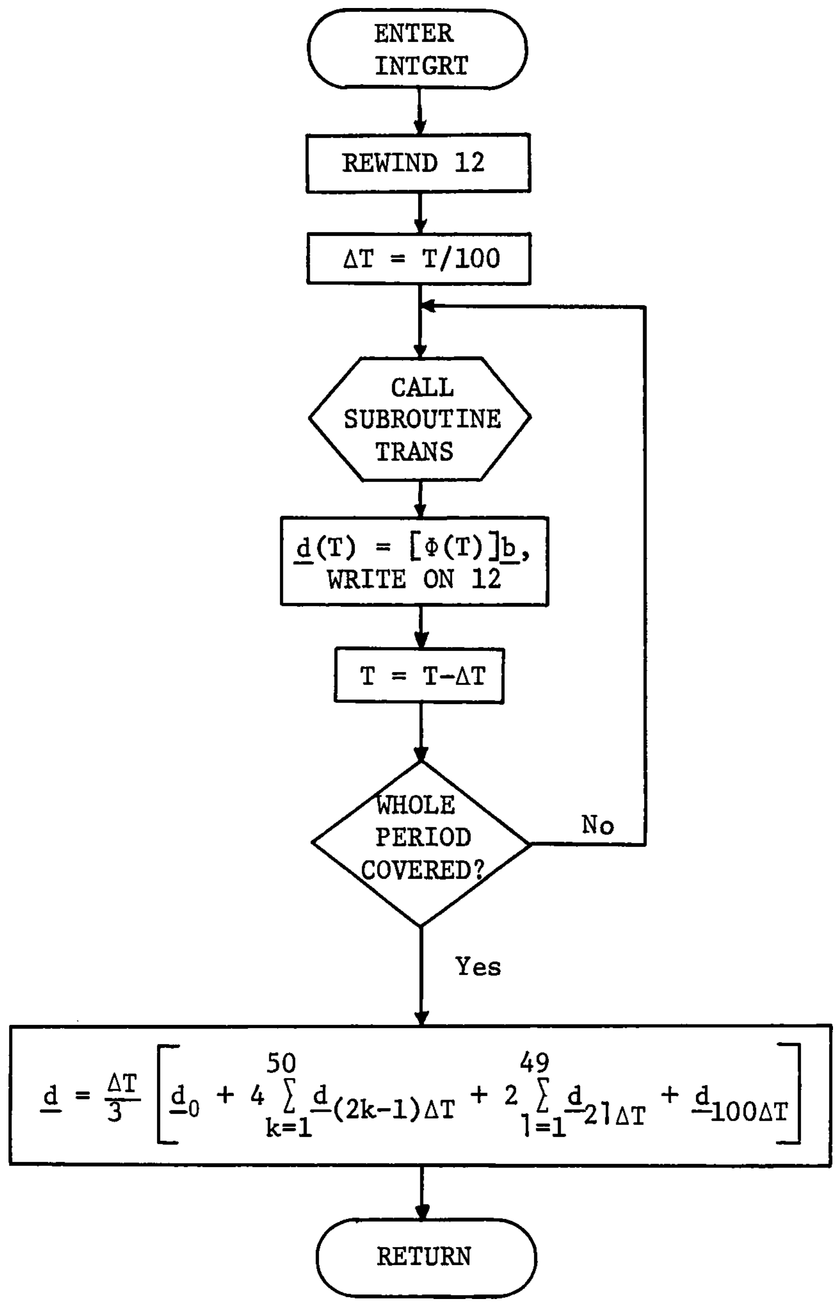

Figure A-3

SUBROUTINE INTGRT

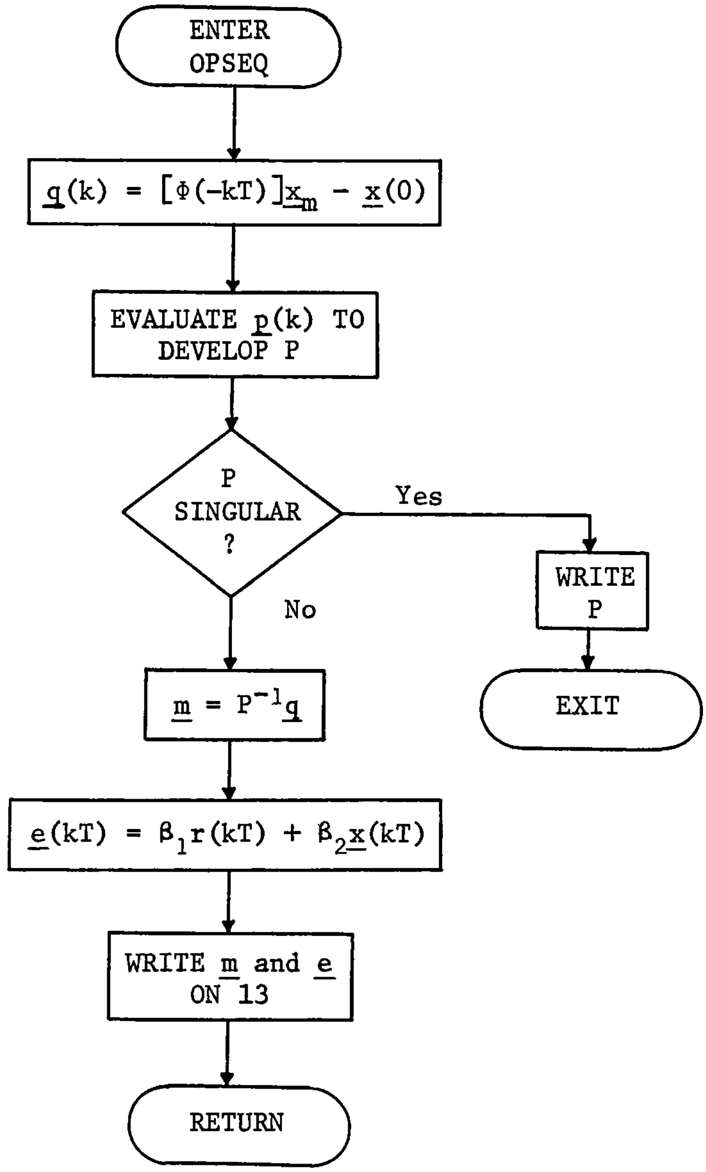

Figure A-4

SUBROUTINE OPSEQ

DET - calculates the determinant of an $m \times m$ matrix using the matrix triangularization method (Frazer, Duncan, and Collar, 1963:106-108)

MINV - calculates the inverse of an $m \times m$ matrix using the method of postmultipliers, with accuracy improvement (Frazer, Duncan, and Collar, 1963:109-112,120-121)

SHIFT - replaces one $m \times n$ matrix with another $m \times n$ matrix

MSUB - forms the difference of two $m \times n$ matrices

MTSCAL - forms the $m \times n$ matrix product of an $m \times n$ matrix and a scalar

IDENT - generates an $m \times m$ identity matrix

MTMPY - forms the $l \times n$ matrix product of an $l \times m$ matrix and an $m \times n$ matrix.

Table A-1 contains FORTRAN source listings of all subroutines in the discrete compensator design program. Arrays are currently dimensioned for a maximum of ten state-variables in the linear plant.

Table A-1

DESIGN PROGRAM SOURCE LISTINGS

MAIN

```
  CALL DESIGN
  WRITE (6,1)
1 FORMAT ('1'////5('*'),'CONTROL SEQUENCES VALID FOR ANALYSIS PHASE'
 1,5('*'))
  STOP
  END
```

Table A-1 (continued)

DESIGN

```
      SUBROUTINE DESIGN
      INTEGER*2 J,NDIMEN,K,ICOL,LASTST,IROW,NPUTS,INCALL,I,NDIM,KOUNT,
     1NINPUT
      REAL*8 INTOER(5,5)
      DOUBLE PRECISION A(10,10),PHI(10,10),PLANT(10,10),B(10,5),D(10,5),
     1X(10,1),R(5,1),XQUIES(10,1),TRNDCR(5,10),FCOMP(10,5),RCOMP(10,5),
     2EPSLN,TIMPD,PERIOD
      COMMON A,PHI,X,XQUIES,B,D,R,EPSLN,PERIOD,NDIM,NINPUT,INCALL
      NAMELIST /SIZE/NDIM,NINPUT /PARM1/PLANT,X,XQUIES,EPSLN /PARM2/
     1B,R,INTOER,TRNDCR /PARM3/TIMPD,LASTST
      REWIND 12
      REWIND 13
      KOUNT = 0
      READ (5,SIZE)
      WRITE (13,1) NDIM,NINPUT
    1 FORMAT (2I2)
      CALL ZERO(B,NDIM,NINPUT)
      CALL ZERO(PLANT,NDIM,NDIM)
      READ (5,PARM1)
      READ (5,PARM2)
    2 INCALL = 0
      READ (5,PARM3)
      PERIOD = TIMPD
      KOUNT = KOUNT+1
      WRITE (6,3)
    3 FORMAT ('1'//14X,'DISCRETE COMPENSATION OF A STATE-VARIABLE'/24X,
     1'DEFINED LINEAR PLANT'//)
      WRITE (6,4) PERIOD
    4 FORMAT (22X,'SAMPLE PERIOD = ',F7.3//)
      CALL TRANS(PLANT)
    5 CALL INTGRT
      WRITE (6,6)
    6 FORMAT (//5X,'VECTOR ''D'' RELATING EXTERNAL INPUTS TO PLANT STATE
     1S'//)
      DO 7 IROW = 1,NDIM
    7 WRITE (6,8) (D(IROW,ICOL), ICOL = 1,NINPUT)
    8 FORMAT (2X,10F10.6)
      CALL OPSEQ(INTOER,TRNDCR)
      IF (LASTST.EQ.0) GO TO 2
      ENDFILE 13
      REWIND 13
      READ (13,1) NDIMEN,NPUTS
      DO 19 K = 1,KOUNT
      WRITE (6,9) K
    9 FORMAT ('1'//15X,'CONTROL SEQUENCES FOR HENRY''S'/14X,'FORM OF THE
     1 DISCRETE COMPENSATOR'//22X,'CASE NUMBER ',I2)
      WRITE (6,10) NDIMEN
   10 FORMAT (/14X,'THE LINEAR PLANT HAS ',I2,' STATES'/)
```

Table A-1 (continued)

DESIGN

```
      READ (13,11) PERIOD
   11 FORMAT (D15.8)
      DO 14 I = 1,NINPUT
      DO 12 J = 1,NDIM
   12 READ (13,11) FCOMP(J,I)
      DO 13 J = 1,NDIM
   13 READ (13,11) RCOMP(J,I)
   14 CONTINUE
      WRITE (6,20) PERIOD
      WRITE (6,15)
   15 FORMAT (//7X,'OPEN-LOOP CONTROL',8X,'ERROR'/11X,'SEQUENCE',12X,
     1'SEQUENCE')
      DO 19 I = 1,NINPUT
      WRITE (6,16)
   16 FORMAT (//)
      DO 17 J = 1,NDIM
   17 WRITE (6,18) FCOMP(J,I),RCOMP(J,I)
   18 FORMAT (8X,F11.5,9X,F11.5)
   19 CONTINUE
   20 FORMAT (18X,'SAMPLE PERIOD = ',F7.3//)
      REWIND 13
      RETURN
      END
```

Table A-1 (continued)

TRANS

```
      SUBROUTINE TRANS(PLANT)
      INTEGER*2 ICOL,IKNT,NDIM,NINPUT,IROW,J,ITER,INCALL
      DOUBLE PRECISION A(10,10),PHI1(10,10),PHI2(10,10),TERM(10,10),
     1PLANT(10,10),B(10,5),D(10,5),X(10,1),R(5,1),XQUIES(10,1),W(10,10),
     2EPSLN,PERIOD,TIME,FCTRL,DET
      COMMON A,PHI1,X,XQUIES,B,D,R,EPSLN,PERIOD,NDIM,NINPUT,INCALL
      IF (INCALL.NE.0) GO TO 6
      CALL ZERO(A,NDIM,NDIM)
      CALL SHIFT(PLANT,A,NDIM,NDIM)
      WRITE (6,1)
    1 FORMAT (5X,'LINEAR PLANT MATRIX ''A'' '//)
      DO  2 IROW = 1,NDIM
    2 WRITE (6,3) (A(IROW,ICOL), ICOL = 1,NDIM)
    3 FORMAT (2X,10F10.6)
      WRITE (6,4)
    4 FORMAT (//5X,'VECTOR ''B'' RELATING EXTERNAL INPUTS TO PLANT STATE
     1S'//)
      DO 5 IROW = 1,NDIM
    5 WRITE (6,3) (B(IROW,ICOL), ICOL = 1,NINPUT)
    6 CALL ZERO(PHI2,NDIM,NDIM)
      CALL IDENT(TERM,NDIM)
      CALL SHIFT(TERM,PHI1,NDIM,NDIM)
      TIME = PERIOD
      FCTRL = 0.0
      IKNT = 1
      DO 10 J = 1,100
      ITER = J
      FCTRL = FCTRL+1.0
      CALL MTMPY(TERM,A,W,NDIM,NDIM,NDIM)
      CALL MTSCAL(W,NDIM,NDIM,(TIME/FCTRL),TERM)
      CALL MADD(PHI1,NDIM,NDIM,TERM,PHI1)
      IF (IKNT-4) 7,8,9
    7 IKNT = IKNT+1
      GO TO 10
    8 CALL SHIFT(PHI1,PHI2,NDIM,NDIM)
      IKNT = IKNT+1
      GO TO 10
    9 IKNT = 1
      IF  (DABS(DET(PHI2,NDIM)-DET(PHI1,NDIM)) .LE. DABS(DET(PHI1,NDIM)*
     1EPSLN))  GO TO 12
   10 CONTINUE
      WRITE (6,11)
   11 FORMAT (//5X,5('*'),'TRANSITION MATRIX ''PHI'' DID NOT CONVERGE IN
     1 100 ITERATIONS',5('*'))
      GO TO 14
   12 IF (INCALL.NE.0) GO TO 17
      WRITE (6,13) ITER
   13 FORMAT (//5X,4('*'),'TAYLOR''S SERIES CONVERGED IN ',I3,' ITERATIO
```

Table A-1 (continued)

TRANS

```
  1NS',4('*'))
14 WRITE (6,15)
15 FORMAT (/5X,'TRANSITION MATRIX ''PHI'' '//)
   DO 16 IROW = 1,NDIM
16 WRITE (6,3) (PHI1(IROW,ICOL), ICOL = 1,NDIM)
17 RETURN
   END
```

Table A-1 (continued)

INTGRT

```
      SUBROUTINE INTGRT
      INTEGER*2 ICOL,NINPUT,IROW,NDIM,I,INCALL
      DOUBLE PRECISION A(10,10),PHI(10,10),PHIHLD(10,10),MAT1(10,5),MAT2
     1(10,5),B(10,5),D(10,5),X(10,1),R(5,1),XQUIES(10,1),EPSLN,PERIOD,
     2HOLD,DELT,ODDCON,EVNCON
      COMMON A,PHI,X,XQUIES,B,D,R,EPSLN,PERIOD,NDIM,NINPUT,INCALL
      REWIND 12
      INCALL = 65535
      CALL SHIFT(PHI,PHIHLD,NDIM,NDIM)
      HOLD = PERIOD
      DELT = 0.01*HOLD
      DO 2 I = 1,101
      CALL TRANS(PHI)
      CALL MTMPY(PHI,B,D,NDIM,NDIM,NINPUT)
      DO 1 IROW = 1,NDIM
    1 WRITE (12) (D(IROW,ICOL), ICOL = 1,NINPUT)
      PERIOD = PERIOD-DELT
    2 CONTINUE
      REWIND 12
      ODDCON = 4.0
      EVNCON = 2.0
      PERIOD = HOLD
      CALL SHIFT(PHIHLD,PHI,NDIM,NDIM)
      CALL ZERO(D,NDIM,NINPUT)
      DO 3 IROW = 1,NDIM
    3 READ (12) (D(IROW,ICOL), ICOL = 1,NINPUT)
      DO 6 I = 1,49
      DO 4 IROW = 1,NDIM
    4 READ (12) (MAT1(IROW,ICOL), ICOL = 1,NINPUT)
      DO 5 IROW = 1,NDIM
    5 READ (12) (MAT2(IROW,ICOL), ICOL = 1,NINPUT)
      CALL MTSCAL(MAT1,NDIM,NINPUT,ODDCON,MAT1)
      CALL MTSCAL(MAT2,NDIM,NINPUT,EVNCON,MAT2)
      CALL MADD(D,NDIM,NINPUT,MAT1,D)
    6 CALL MADD(D,NDIM,NINPUT,MAT2,D)
      DO 7 IROW = 1,NDIM
    7 READ (12) (MAT1(IROW,ICOL), ICOL = 1,NINPUT)
      CALL MTSCAL(MAT1,NDIM,NINPUT,ODDCON,MAT1)
      CALL MADD(D,NDIM,NINPUT,MAT1,D)
      DO 8 IROW = 1,NDIM
    8 READ (12) (MAT1(IROW,ICOL), ICOL = 1,NINPUT)
      REWIND 12
      CALL MADD(D,NDIM,NINPUT,MAT1,D)
      CALL MTSCAL(D,NDIM,NINPUT,(DELT/3.0),D)
      RETURN
      END
```

Table A-1 (continued)

OPSEQ

```
      SUBROUTINE OPSEQ(BETAI,BETA)
      INTEGER*2 J,ICOL,INCALL,K,IROW,NDIM,NINPUT,I
      REAL*8 M(10,1)
      DOUBLE PRECISION A(10,10),PHI(10,10),B(10,5),D(10,5),X(10,1),R(5,1
     1),P(10,10),Q(10,1),VEC(10,1),PHIHLD(10,10),XQUIES(10,1),VECHLD(10,
     21),BETA1(1,5),BETA2(1,10),XVEC(10,1),ERROR(10,1),BETA(5,10),BETAI(
     35,5),DVEC(10,1),WORK(10,1),WORK1(10,10),EPSLN,PERIOD,HOLD,FORDER,
     4FJ,FM,TEMP1,TEMP2,DET
      COMMON A,PHI,X,XQUIES,B,D,R,EPSLN,PERIOD,NDIM,NINPUT,INCALL
      REWIND 12
      INCALL = 255
      HOLD = PERIOD
      CALL SHIFT(PHI,PHIHLD,NDIM,NDIM)
      FORDER = NDIM
      PERIOD = -FORDER*HOLD
      CALL TRANS(PHI)
      CALL MTMPY(PHI,XQUIES,WORK,NDIM,NDIM,1)
      CALL MSUB(WORK,NDIM,1,X,Q)
      PERIOD = HOLD
      WRITE (13,1) PERIOD
    1 FORMAT (D15.8)
      DO 13 I = 1,NINPUT
      DO 2 J = 1,NINPUT
      BETA1(1,J) = BETAI(I,J)
    2 CONTINUE
      DO 3 J = 1,NDIM
      BETA2(1,J) = BETA(I,J)
      VEC(J,1) = D(J,I)
    3 CONTINUE
      CALL SHIFT(VEC,VECHLD,NDIM,1)
      DO 5 J = 1,NDIM
      CALL SHIFT(VECHLD,VEC,NDIM,1)
      FJ = J
      PERIOD = -FJ*HOLD
      CALL TRANS(PHI)
      CALL MTMPY(PHI,VEC,WORK,NDIM,NDIM,1)
      CALL SHIFT(WORK,VEC,NDIM,1)
      DO 4 K = 1,NDIM
    4 P(K,J) = VEC(K,1)
    5 CONTINUE
      IF (DABS(DET(P,NDIM)).LE.0.1D-5) GO TO 14
      CALL MINV(P,NDIM,WORK1)
      CALL MTMPY(WORK1,Q,M,NDIM,NDIM,1)
      DO 10 J = 1,NDIM
      IF (J.GT.1) GO TO 7
      CALL SHIFT(X,XVEC,NDIM,1)
      GO TO 9
    7 DO 8 K = 1,NDIM
```

Table A-1 (continued)

OPSEQ

```
 8 DVEC(K,1) = D(K,I)
   CALL MTMPY(PHIHLD,XVEC,WORK,NDIM,NDIM,1)
   FM = M(J-1,1)
   CALL MTSCAL(DVEC,NDIM,1,FM,WORK1)
   CALL MADD(WORK,NDIM,1,WORK1,XVEC)
 9 CALL MTMUL(BETA1,R,TEMP1,NINPUT)
   CALL MTMUL(BETA2,XVEC,TEMP2,NDIM)
   ERROR(J,1) = TEMP1+TEMP2
10 CONTINUE
   DO 11 J = 1,NDIM
11 WRITE (13,1) M(J,1)
   DO 12 J = 1,NDIM
12 WRITE (13,1) ERROR(J,1)
13 CONTINUE
   PERIOD = HOLD
   CALL SHIFT(PHIHLD,PHI,NDIM,NDIM)
   RETURN
14 WRITE (6,15)
15 FORMAT ('1'//5('*'),'MATRIX ''P'' IS SINGULAR',5('*')//)
   DO 16 IROW = 1,NDIM
16 WRITE (6,17) (P(IROW,ICOL), ICOL = 1,NDIM)
17 FORMAT (10D12.5)
   STOP
   END
```

Table A-1 (continued)

ZERO

```
      SUBROUTINE ZERO(A,M,N)
      INTEGER*2 N,I,J,M
      DOUBLE PRECISION A(10,10)
      DO 1 J = 1,N
      DO 1 I = 1,M
    1 A(I,J) = 0.0
      RETURN
      END
```

Table A-1 (continued)

MADD

```
  SUBROUTINE MADD(A,M,N,B,C)
  INTEGER*2 J,M,N,I
  DOUBLE PRECISION A(10,10),B(10,10),C(10,10)
  DO 1 J = 1,N
  DO 1 I = 1,M
1 C(I,J) = A(I,J)+B(I,J)
  RETURN
  END
```

Table A-1 (continued)

MTMUL

```
      SUBROUTINE MTMUL(A,B,C,M)
      INTEGER*2 M,K
      DOUBLE PRECISION A(1,10),B(10,1),C
      C = 0.0
      DO 1 K = 1,M
    1 C = C+A(1,K)*B(K,1)
      RETURN
      END
```

Table A-1 (continued)

DET

```
      DOUBLE PRECISION FUNCTION DET(A,N)
      INTEGER*2 J,K,MM,M,I,N
      DOUBLE PRECISION A(10,10),AA(10,10),C(10,10),D(10,10)
      MM = N-1
      DO 1 J = 1,N
      DO 1 I = 1,N
    1 AA(I,J) = A(I,J)
      DET = 1.0
      DO 7 M = 1,MM
      DO 3 J = 1,N
      DO 3 I = 1,N
      IF (I.EQ.J) GO TO 2
      C(I,J) = 0.0
      GO TO 3
    2 C(I,J) = 1.0
    3 CONTINUE
      DO 4 J = M,MM
    4 C(M,J+1) = -AA(M,J+1)/AA(M,M)
      DO 5 J = M,N
      DO 5 I = M,N
      D(I,J) = 0.0
      DO 5 K = M,N
    5 D(I,J) = AA(I,K)*C(K,J)+D(I,J)
      DO 10 J = M,N
      DO 10 I = M,N
   10 AA(I,J) = D(I,J)
    7 DET = DET*D(M,M)
      DET = DET*D(N,N)
      RETURN
      END
```

Table A-1 (continued)

MINV

```
      SUBROUTINE MINV(AA,N,AINV)
      DIMENSION ID(10)
      INTEGER*2 J,II,KK,N2,K,IS,NI,NIT,M,IC,IT,NN,I,N,ID,KJ,N1
      DOUBLE PRECISION AA(10,10),AINV(10,10),A(10,20),WORK3(10,10),B,C,W
      NN = N+1
      N2 = 2*N
      DO 1 I = 1,N
      DO 1 J = 1,N
    1 A(I,J) = AA(I,J)
      K = 1
      DO 2 I = 1,N
      DO 2 J = NN,N2
    2 A(I,J) = 0.0
      DO 3 I = 1,N
      NI = N+I
      A(I,NI) = 1.0
    3 ID(I) = I
    4 KK = K+1
      IS = K
      IT = K
      B = DABS(A(K,K))
      DO 5 I = K,N
      DO 5 J = K,N
      IF (DABS(A(I,J)).LE.B) GO TO 5
      IS = I
      IT = J
      B = DABS(A(I,J))
    5 CONTINUE
      IF (IS.LE.K) GO TO 7
      DO 6 J = K,N2
      C = A(IS,J)
      A(IS,J) = A(K,J)
    6 A(K,J) = C
    7 IF (IT.LE.K) GO TO 9
      IC = ID(K)
      ID(K) = ID(IT)
      ID(IT) = IC
      DO 8 I = 1,N
      C = A(I,IT)
      A(I,IT) = A(I,K)
    8 A(I,K) = C
    9 DO 10 J = KK,N2
      A(K,J) = A(K,J)/A(K,K)
      DO 10 I = KK,N
      W = A(I,K)*A(K,J)
      A(I,J) = A(I,J)-W
      IF (DABS(A(I,J)).GE.0.1D-3*DABS(W)) GO TO 10
      A(I,J) = 0.0
```

Table A-1 (continued)

MINV

```
10 CONTINUE
   K = KK
   IF (K.LT.N) GO TO 4
   DO 11 J = NN,N2
11 A(N,J) = A(N,J)/A(N,N)
   N1 = N-1
   DO 12 M = 1,N1
   I = N-M
   II = I+1
   DO 12 K = II,N
   DO 12 J = NN,N2
12 A(I,J) = A(I,J)-A(I,K)*A(K,J)
   DO 14 I = 1,N
   DO 14 J = 1,N
   IF (ID(J).NE.I) GO TO 14
   DO 13 K = NN,N2
   KJ = K-N
13 AINV(I,KJ) = A(J,K)
14 CONTINUE
   DO 15 I = 1,N
   CALL IDENT(A,N)
   CALL MTSCAL(A,N,N,2.0,A)
   CALL MTMPY(AA,AINV,WORK3,N,N,N)
   CALL MSUB(A,N,N,WORK3,A)
   CALL MTMPY(AINV,A,WORK3,N,N,N)
   CALL SHIFT(WORK3,AINV,N,N)
15 CONTINUE
   RETURN
   END
```

Table A-1 (continued)

SHIFT

```
      SUBROUTINE SHIFT(A,B,M,N)
      INTEGER*2 M,N,I,J
      DOUBLE PRECISION A(10,10),B(10,10)
      DO 1 J = 1,N
      DO 1 I = 1,M
    1 B(I,J) = A(I,J)
      RETURN
      END
```

Table A-1 (continued)

MSUB

```
      SUBROUTINE MSUB(A,M,N,B,C)
      INTEGER*2 J,M,N,I
      DOUBLE PRECISION A(10,10),B(10,10),C(10,10)
      DO 1 J = 1,N
      DO 1 I = 1,M
    1 C(I,J) = A(I,J)-B(I,J)
      RETURN
      END
```

Table A-1 (continued)

MTSCAL

```
      SUBROUTINE MTSCAL(A,M,N,S,B)
      INTEGER*2 M,N,I,J
      DOUBLE PRECISION A(10,10),B(10,10),S
      DO 1 J = 1,N
      DO 1 I = 1,M
    1 B(I,J) = A(I,J)*S
      RETURN
      END
```

Table A-1 (continued)

IDENT

```
      SUBROUTINE IDENT(A,M)
      INTEGER*2 I,J,M
      DOUBLE PRECISION A(10,10)
      DO 2 J = 1,M
      DO 2 I = 1,M
      IF (I.EQ.J) GO TO 1
      A(I,J) = 0.0
      GO TO 2
    1 A(I,J) = 1.0
    2 CONTINUE
      RETURN
      END
```

Table A-1 (continued)

MTMPY

```
      SUBROUTINE MTMPY(A,B,C,L,M,N)
      INTEGER*2 J,N,K,L,I,M
      DOUBLE PRECISION A(10,10),B(10,10),C(10,10)
      DO 1 J = 1,N
      DO 1 I = 1,L
      C(I,J) = 0.0
      DO 1 K = 1,M
    1 C(I,J) = C(I,J)+A(I,K)*B(K,J)
      RETURN
      END
```