ON REDUCTION OF FINITE-WORD-LENGTH EFFECTS IN STATE-SPACE DIGITAL FILTERS

BY

PAUL ROBERT MOON

A THESIS PRESENTED TO THE
FACULTY OF GRADUATE STUDIES
UNIVERSITY OF MANITOBA

IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE
DOCTOR OF PHILOSOPHY

FEBRUARY 1983
WINNIPEG, MANITOBA

ON REDUCTION OF FINITE-WORD-LENGTH EFFECTS IN STATE-SPACE DIGITAL FILTERS

BY

PAUL ROBERT MOON

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

Title: On Reduction of Finite-Word-Length Effects in State-Space Digital Filters

Author: Paul Robert Moon

ABSTRACT

This thesis deals with the reduction of finite-word-length effects in state-space digital filters. The frequency-domain approximation problem is discussed here only as a preliminary. A basic analysis of finite-word-length effects is given. These effects are classified as follows: frequency-domain errors due to coefficient quantization; roundoff noise, due to time-domain errors inherent in signal quantization; and limit cycles, due to the system non-linearity inherent in signal quantization. Thus, it is clear that state-space realizations possessing low coefficient sensitivity and low sensitivity to time-domain errors will exhibit reduced finite-word-length effects. The elimination of limit cycles is indirectly related to low-sensitivity realizations by means of Lyapunov theory. A survey of some significant literature pertaining to low-sensitivity realizations and Lyapunov theory is given.

Certain classes of low-sensitivity state-space realizations are extended by a further development of the Lyapunov approach. The two main classes considered are wave-digital realizations and minimum-roundoff-noise realizations. For the minimum-roundoff-noise realization, the solutions K and W to the Lyapunov matrix equations $K - AKA^T = BB^T$ and $W - A^TWA = C^TC$ are diagonal, where real-number coefficients are allowed in the matrices A, B, and C of the state-space realization. It is shown that the solutions K and W are also diagonal for the double-input-output state equations which derive from the wave digital filter. Well-known consequences of the diagonal solution W

are: suppression of zero-input limit cycles under conditions of magnitude truncation at the state variables, by means of the Lyapunov function $x^T Wx$; stability of the forced response under conditions of magnitude truncation in a certain time-varying non-linearity. A well-known consequence of diagonal solutions K and W is minimum roundoff noise in the optimal-word-length case, under a dynamic-range constraint which limits the probability of overflow. In the case of the wave-digital realization, it is shown that the minimum-roundoff-noise property extends as follows: the weighted sum of roundoff noises occurring at both outputs is minimum, under a dynamic-range constraint at each state variable on a weighted sum of signals due to both inputs.

It is shown that the above-mentioned consequences of diagonal solutions K and W are essentially retained if K and W exhibit sufficient diagonal dominance. Diagonally dominant solutions K and W will result from the approximation of real-number coefficients in A, B, and C by (binary) quantized coefficients. In particular, separate necessary and sufficient diagonal-dominance conditions on W are given for $x^T Wx$ to be a Lyapunov function in the event of signal overflow and underflow truncations; this excludes zero-input limit cycles. Similar diagonal-dominance conditions are given which guarantee stability of the forced response by use of the same Lyapunov function $x^T Wx$.

A method of coefficient quantization is developed to limit the degradation of frequency response and diagonal dominance. It employs a search for integer coefficients in a constraint region of coefficient space, by a specialized branch-and-bound (integer programming)

algorithm. The constraint region limits the variation of system eigenvalues; adjustment of mode excitations compensates eigenvector variations, to restore transfer-function zeroes. The variation of K and W is limited by a demonstrated relation to the system eigenanalysis. An overall design procedure is implemented and applied to examples, by a battery of interactive computer programs. A distributed-arithmetic digital implementation, based on the author's previous work, is extended to obtain a necessary true saturation overflow characteristic.

ACKNOWLEDGEMENT

The author wishes to express his appreciation to Professor G. O. Martens for the advice and encouragement received during the course of this research work. He also wishes to thank any other staff members and any graduate students at the University of Manitoba who may have contributed to this work directly or indirectly.

Financial support from the following sources is also gratefully acknowledged: the University of Manitoba; the National Research Council of Canada (now NSERC); Gulf Canada Limited; and the Canadian Council of Professional Engineers.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Statement of Problem

The field of digital signal processing, including digital filters, has expanded rapidly in recent years. The latent possibilities of discrete-time data processing have become realities through extensive research. This is due mainly to the advent of the high= speed digital computer around 1960, which made "real-time" implementation practical for certain signal-processing algorithms. Later, compact hardware implementations became possible by means of integrated circuits. The advantages of digital filters are well known: precise specification of critical frequencies; absence of component-tolerance effects due to aging; possible very-high-Q transfer functions; and no limitations due to component size, at low frequencies.

We may regard a discrete-time filter as an operator which transforms a sequence of numbers into another sequence of numbers. This discrete-time operator is linear and time-invariant in many practical cases. It is usually expressed by a definite computational rule which can be simply stated. (The term "sampled-data filter" is equivalent to "discrete-time filter"; it was often used in early literature, since the sequence of numbers usually originates by sampling continuous data.)

We make the following fundamental distinction between a digital filter and a discrete-time filter. In a discrete-time filter the computational rule is carried out exactly, i.e., with infinite

precision. However, the digital filter is essentially an approximation to some prototype discrete-time filter, where all computations and numerical representations are restricted to finite precision. This restriction naturally arises from the finite-word-length limitation in all digital computers. Although the above distinction appears to be a matter of definition, it is the origin of the basic consideration in digital-filter design. This consideration is the deviation of the response of the digital filter from the response of the prototype discrete-time filter. The various deviations which may occur are generally termed "finite-word-length effects" (FWLEs).

The principal problem in digital filter design is the reduction of FWLEs to a tolerable level. This problem lies at the root of a good deal of current research in digital filters. The preliminary problem in digital filter design is the approximation of a desired response in the frequency domain. The frequency-domain approximation problem belongs more properly to the synthesis of the prototype discrete-time filter. This approximation problem has been adequately treated in the literature.

1.2 Scope of Thesis Research

This thesis deals with the reduction of finite-word-length effects (FWLEs) in state-space filters. The state-space approach to digital filters is chosen because it provides a framework which is well suited to the general investigation of FWLEs. The Lyapunov approach is chosen for the reduction of FWLEs. The Lyapunov approach as it appears here is an abstraction of some underlying principles

which are common to certain results found in the literature; it is further developed in this thesis. It is chosen because it represents a coherent approach in an area (FWLEs) where a great many specific, but often isolated, results have been published. The reader may refer to the literature survey in Section 4.6 for more information concerning the background of the Lyapunov approach. A basic analysis of FWLEs is given. FWLEs are classified as follows: frequency-domain errors, due to coefficient variations; roundoff noise, due to time= domain errors inherent in signal quantization; and limit cycles, due to the system non-linearity inherent in signal quantization. Thus, we conclude that state-space realizations possessing low coefficient sensitivity and low sensitivity to time-domain errors will exhibit reduced FWLEs. The elimination of limit cycles, which is a separate issue, is indirectly related to low-sensitivity realizations by means of the Lyapunov theory.

The result of this thesis is to extend certain classes of low-sensitivity state-space realizations, by means of a further development of the Lyapunov approach. A chain of closely related problems is considered, which includes the digital hardware implementation of the state-space realization; some of these problems are amenable to future research, as well. The two main classes of low-sensitivity realizations considered here are wave digital filters [G1-G16] (by the state-space approach) and the minimum-roundoff-noise realization [C10].

As a first step, the Lyapunov theory pertinent to the wave= digital and minimum-roundoff-noise realizations is consolidated in the following way. For the minimum-roundoff-noise realization, the

solutions K and W to the Lyapunov matrix equations $K - AKA^T = BB^T$ and $W - A^TWA = C^TC$ are diagonal, where real-number coefficients are allowed in the matrices A, B, and C of the state-space realization. Here it is shown that the solutions K and W to the Lyapunov matrix equations are also diagonal for the double-input-output state equations which derive from the wave digital filter. Some desirable consequences of the diagonal solutions K and W are: 1) zero-input limit cycles are excluded under conditions of magnitude truncation at the state variables, by means of the Lyapunov function x^TWx; 2) stability of the forced response is obtained under conditions of magnitude truncation in a certain time-varying non-linearity [D10]; 3) minimum roundoff noise is obtained, according to the analysis given in [C10]. In the case of the wave-digital realization, it is shown here that the minimum-roundoff-noise property extends as follows: the weighted sum of roundoff noises at both outputs is minimum, over all realizations obtained by a similarity transformation on the wave= digital realization.

The central result of this thesis is to extend the application of the Lyapunov theory to the case where the above-mentioned solutions K and W are only approximately diagonal, i.e., diagonally dominant. Such diagonally dominant solutions will result from the approximation of real-number coefficients in the matrices A, B, and C by (binary) quantized coefficients. It is shown here that the above-mentioned desirable consequences of diagonal solutions K and W are essentially retained if K and W exhibit sufficient diagonal dominance. In particular, separate necessary and sufficient

diagonal-dominance conditions on W are given for $x^T Wx$ to be a Lyapunov function in the event of signal overflow and underflow truncations; this excludes zero-input limit cycles. Similar diagonal-dominance conditions are given which guarantee stability of the forced response by use of the same Lyapunov function $x^T Wx$. Thus, the obstacle which coefficient quantization poses to the Lyapunov approach is removed.

A method of coefficient quantization is developed to limit the degradation of frequency response and diagonal dominance. It employs a search for integer coefficients in a constraint region of coefficient space, by a specialized branch-and-bound (integer programming) algorithm. The constraint region limits the variation of system eigenvalues and eigenvectors; adjustment of mode excitations compensates eigenvector variations, to restore the transfer-function zeroes. The variation of K and W is limited by a demonstrated relation to the system eigenanalysis. The overall design procedure is implemented and applied to examples, by a battery of interactive computer programs.

A digital implementation by distributed arithmetic, previously given by the author and Prof. G. O. Martens [H3], is applied to the coefficient-quantized state equations. This method of implementation is further developed here; in particular, it is extended to obtain the true saturation overflow characteristic required here.

1.3 Organization of Thesis

This thesis consists of ten chapters. Chapter 2 provides background material relevant to digital filters, in order to establish

a starting point for the present investigation. Chapter 3 contains useful results from the theory of linear state-space realizations; these realizations are the prototypes upon which state-space digital filters are built. Chapter 4 deals with the principal problem in state-space digital filters: finite-word-length effects. A useful non-linear model of a state-space digital filter is given there and finite-word-length effects are broadly classified as roundoff-noise, stability, and (coefficient) sensitivity problems. A survey of recent research in state-space digital filters is given and the state-space approach to wave digital filters is outlined.

The further development of the Lyapunov approach to reducing FWLEs is contained in Chapters 5, 6, and 7. These chapters deal with the roundoff-noise, stability, and (coefficient) sensitivity problems, respectively. In Chapter 5, a basic analysis of roundoff noise is given, followed by a summary of the theory of minimum-roundoff noise realizations. This leads to the essentially new considerations in Sections 5.5, 5.6, and 5.7, respectively: the effects of coefficient quantization on the minimum-roundoff-noise property; the correspondence between Lyapunov equations of wave-digital realizations and minimum-roundoff-noise realizations; and the interpretation of the wave-digital realization as a two-input-two-output minimum-roundoff-noise realization. In Chapter 6, the new diagonal-dominance conditions on W are derived, for $x^T Wx$ to be a Lyapunov function in the event of underflow and overflow signal conditions. This gives suppression of zero-input underflow and overflow limit cycles, under appropriate conditions of magnitude truncation. The same approach is extended there to

obtain diagonal-dominance conditions for stability of the forced response; a detailed analysis of the forced response is given for this purpose. An interpretation of the Lyapunov function is given by means of vector and related matrix norms. In Chapter 7, the method of coefficient quantization is developed, with the aim of retaining an acceptable frequency response and the favorable roundoff-noise and stability properties implied by the diagonal K and W.

The digital implementation of state-space realizations is treated in Chapter 8; this is the final step in obtaining a state=space digital filter. The overall design procedure is outlined and all related computer programs are described in Chapter 9; examples are included there. Concluding remarks are found in Chapter 10.

1.4 Notation and Conventions

Standard mathematical notation is used for the most part; exceptions are defined wherever they are introduced. Vector and scalar functions are denoted by lower case Latin letters. Matrices are denoted by upper case Latin letters. A single subscript to a vector always denotes a component of that vector; a single subscript to a scalar denotes a different "version" of that scalar. Double subscripts to upper or lower case letters always denote elements of a matrix. Superscripts to a vector denote different "versions" of

the same vector, usually for purposes of comparison; e.g., $x^{(1)}$ and $x^{(2)}$. The superscripts T and -1 denote the transpose and inverse of a matrix, respectively. The symbol $\| \ \|$ denotes the norm of the vector enclosed by it; i.e., $\| x \| = \sqrt{x^T x}$. Note that a generalized norm is introduced in Section 6.14. Finally, the choice of letter symbols in Chapter 8 is independent of the symbols in the remainder of the thesis; no confusion should result, since the topic of this Chapter is entirely self-contained.

The reference list is categorized by subject area; there are ten categories, which are assigned letters A through J. This aids in "sorting out" the large number of publications which have appeared in the field of digital filters. Reference numbers, which appear in brackets, consist of a letter and a number, denoting the subject category and the listing under that category, respectively. For example, [C7] refers to reference number 7 in category C.

2. BACKGROUND: DIGITAL FILTERS

2.1 Introduction

This Chapter contains background material to orient the reader to the field of digital filters. It will also bring us to the appropriate point of departure for the present investigation.

2.2 Historical Background

The basic notion associated with discrete-time filters probably occurred quite early in the development of mathematics, perhaps soon after men were able to compute. Consider the following example:

SEQUENCE: Daily tide recordings (height in feet)

OPERATOR: Average tide readings over previous thirty days

Here the operator (discrete-time filter), which transforms the sequence of tide readings into a sequence of averages, is given by a simple computational rule. The result is a crude low-pass filter, which can be shown by well-known methods of analysis. This agrees with our instinctive perception that the averaging process will smooth out localized variations in data.

A more formal development of discrete-time filters began in the seventeenth century. That period saw the development of the classical linear computational algorithms: numerical integration, numerical differentiation, and numerical interpolation. These algorithms were applied to the compilation of mathematical tables and the reduction of astronomical observations. The calculus of finite

differences gradually evolved from these computational algorithms and other operational methods. By the first third of the twentieth century, the calculus of finite differences was highly developed. It provides the basic forerunner to the discrete-time filter: the nth-order linear difference equation.

The advent of World War II intensified research in signal processing, especially related to discrete-time systems. The necessity of dealing with sampled data arose in certain notable cases: the processing of radar signals corrupted by noise and the design of aircraft control and guidance systems. The z-transform theory, which stems from an early period in mathematics, was applied to the analysis of discrete-time systems during and after World War II. The z-transform has since become the primary analytical tool for discrete-time systems.

The status of discrete-time filter theory in the period between World War II and 1960 can be summarized as follows. The starting point is the nth-order difference equation, borrowed from the calculus of finite differences. The application of z-transform theory to the nth-order difference equation gives the z-domain transfer function, an nth-degree rational function in z; this is analogous to the familiar s-domain transfer function in continuous-time systems. The concept of discrete-time "frequency response" also follows quite naturally from the z-domain transfer function. A filter realization is then achieved by a more-or-less direct implementation of the z-domain transfer function. The implementations in the above-mentioned time period were usually carried out by analog means, including analog computers; these were not necessarily "real-time" implementations.

Books and papers such as [A1-A5, B1] reflect the trend during this period.

The introduction of the high-speed digital computer, about 1960, provided new possibilities for the implementation of discrete-time filters. This motivated the development of new algorithms, which made real-time implementations feasible in many cases. A typical example of an early application of the digital computer is given by [B1].

The reduction of FWLEs is a major research problem which arose from the implementation of discrete-time filters by the digital computer. The need to consider this problem followed directly from the essential difference between a digital filter and discrete-time filter, i.e., the finite-word-length limitation, which was mentioned in Section 1.1. Also, the new possibilities provided by the digital computer motivated renewed research related to a second important problem: the approximation of a desired response in the frequency domain. This second problem was thoroughly investigated during the 1960's. An adequate theory was constructed within a relatively short time, due to its close relation to previously established work. On the other hand, the problem of FWLEs was not nearly so well resolved during the same time period. In Sections 2.3 through 2.6, we will summarize the status of research in the 1960's, since it is the logical point of departure for more current research in digital filters.

2.3 Z-Transform Analysis: Transfer Function

The use of the z-domain transfer function for the analysis of a linear time-invariant discrete-time filter was well established by 1960, as mentioned in Section 2.2. The transfer function, $H(z)$ of such a filter is a rational function in z^{-1}, which has the form

$$H(z) = \frac{b_0 z^{-0} + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_q z^{-q}}{a_0 z^{-0} + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \quad . \qquad (2.1)$$

It can be shown [A6] that the steady-state response of the filter to a complex-exponential input sequence, $e^{j\omega k}$, $k = 0, 1, 2, \ldots$, is a complex-exponential output sequence, $Me^{j(\omega k + \theta)}$, $k = 0, 1, 2, \ldots$. Further, the magnitude M and phase θ of the response are given by

$$M(\omega) = |H(z)|_{z = e^{j\omega}} \qquad (2.2)$$

and

$$\theta(\omega) = \text{Arg } H(z)|_{z = e^{j\omega}} \qquad (2.3)$$

respectively; i.e., $H(z)$ is evaluated on the unit circle $z = e^{j\omega}$. The steady-state response to the real input sequence $\cos \omega k$, $k = 0, 1, 2, \ldots$, is the real output sequence $M \cos(\omega k + \theta)$, $k = 0, 1, 2, \ldots$; i.e., the magnitude and phase are again given by (2.2) and (2.3). This follows by linear superposition of steady-state responses due to $e^{j\omega k}$ and $e^{-j\omega k}$, $k = 0, 1, 2, \ldots$.

2.4 Frequency-domain Approximation Problem

The frequency-domain approximation problem, mentioned in Section 2.2, is the following: $M(\omega)$ and $\theta(\omega)$, given by (2.2) and (2.3), must approximate certain objective functions of ω. Typical objective functions for $M(\omega)$ are low-pass, band-pass, and high-pass characteristics. The simultaneous approximation of a second objective function by $\theta(\omega)$ is a more difficult problem, which requires special techniques, due to the interrelation between $\theta(\omega)$ and $M(\omega)$. Thus, the degrees n and q and the coefficients $a_0, a_1, \ldots, a_n$ and $b_0, b_1, \ldots, b_q$ in (2.1) must be determined to give the required accuracy in the function approximation(s). Real-number coefficients are allowed in the discrete-time filter. Later these coefficients will be quantized due to the inherent finite-word-length limitation of digital filters.

The approximation problem received renewed attention during the 1960's, e.g., in [B2]. The impulse-invariant method was already an established design technique, but it suffered from frequency-aliasing effects. Steiglitz [B3] showed an equivalence between the frequency-domain approximation problems for discrete-time and continuous-time systems, by means of the bilinear transformation

$$z = \frac{1 + s}{1 - s} \quad . \tag{2.4}$$

This transformation maps the imaginary axis of the s-plane onto the unit circle of the z-plane, accompanied by a "warping" of the frequency scale. The equivalence makes continuous-time methods applicable to the discrete-time case. This is advantageous since the frequency-domain

approximation problem for continuous-time systems has been the object of research for over sixty years; this approximation problem reduces essentially to the problem of approximating functions by polynomials. Gibbs [B4] considered the above-mentioned equivalence in greater detail; he established additional correspondences between frequency= domain properties of discrete-time and continuous-time systems. Sablatash [B5] has pointed out further applications of the equivalence. In view of the adequate theory which has been developed, we will not consider the frequency-domain approximation problem further in this thesis, except as it relates to coefficient quantization. That is, we will take the transfer function (2.1) as a starting point, where its coefficients have been completely determined.

2.5 Finite-Word-Length Effects

The problem of FWLEs became very apparent during the 1960's, since FWLEs are unavoidable in the implementation of a discrete-time filter by a digital computer. FWLEs are the distinguishing facet of the realization problem for digital filters. Without FWLEs, the realization problem would simply reduce to the frequency-domain approximation problem for the prototype discrete-time filter. FWLEs fall into two categories, which correspond to the two possible causes. These causes are coefficient quantization [E1] and signal quantization [C1], which are both dictated by finite-word-length limitations. ("Coefficient" refers to any fixed multiplier in the realization; "signal" refers to any input-dependent variable in the realization.)

The FWLE due to coefficient quantization is a deterministic change in the frequency response. The magnitude of the change depends on the sensitivity of the poles and zeros of the transfer function to coefficient variations.

The FWLE due to signal quantization is time-domain errors in all signals, including the output. The magnitudes of the time-domain errors depend on the time-domain sensitivities of the realization to signal disturbances. The time-domain errors fall into two categories: round-off noise and limit cycles.

Roundoff noise is a random disturbance due to the discard of signal underflow bits (i.e., excess signal bits in the least significant positions). We may extend the definition of roundoff noise to include occasional random disturbances due to the discard of signal overflow bits (i.e., excess signal bits in the most significant positions). These overflow disturbances should occur rather infrequently, if the numerical range of signal representation is properly matched to the actual signal levels.

Limit cycles are periodic disturbances which the digital filter may support because signal quantization creates a non-linear system. Limit cycles are truly a characteristic of the digital realization alone, since they cannot exist in a stable prototype linear discrete-time filter. Zero-input limit cycles may be excited by either signal underflows or signal overflows; limit cycles in the forced response are usually associated with signal overflows. Underflow limit cycles have the same basic origin as (underflow) roundoff noise, but they may be substantially larger in amplitude because of their non-random

nature. Overflow limit cycles are very large amplitude disturbances, since they occur at the extremes of the signal representation. Overflow limit cycles should be avoided, since they may render a filter unusable.

The reduction of FWLEs associated with a given discrete-time realization is achieved only by increasing the coefficient and signal word lengths. An increase in the coefficient word lengths decreases the deviation of the frequency response from that of the prototype discrete-time filter. An increase in signal word lengths increases the ratio of the allowable signal to roundoff noise [C2] and to potential underflow limit cycles. The cost of these improvements is an increase in the bit-adder product in the digital implementation. Another approach to reducing FWLEs is to determine alternative, less sensitive realizations, if possible.

The elimination of overflow limit cycles requires special consideration. The most apparent method of achieving this would be to eliminate any possibility of signal overflow, by maintaining a sufficiently large ratio between the range of numerical representation and the average signal levels. This would require a large increase in signal word lengths, hence a large increase in the bit-adder product, solely for the purpose of eliminating overflows. The required increase in signal word lengths would not contribute at all to improving the ratio of signal to (underflow) roundoff noise, since this particular increase must not be accompanied by an increase in signal levels. A more desirable approach is to determine alternative realizations with the inherent property of eliminating overflow limit cycles.

2.6 Transfer-Function Realizations

A. Direct-Form Realizations

The direct-form realization [A6-A7] of a discrete-time filter follows immediately from the transfer function H(z) in (2.1). Here a one-to-one correspondence exists between the coefficients of the realization and the coefficients of the transfer function, i.e., the transfer function is not decomposed algebraically. The properties of this realization are closely related to the properties of the nth- and qth-order polynomials which comprise the denominator and numerator of H(z).

The direct-form realization was the first form to be implemented on the digital computer. FWLEs were found to be quite serious in this form. That is, pole-zero sensitivity to coefficient variations is high and time-domain sensitivity to signal disturbances is high. This leads to significant variation in the frequency response and high roundoff noise, respectively. Also, the direct form was found to be capable of supporting underflow and overflow limit cycles.

The coefficient and signal word lengths required in the direct form are quite large, if FWLEs are to be sufficiently small. Also, for suppression of overflow limit cycles, the costly approach mentioned in the previous Section would be required. These disadvantages lead to the development of building-block realizations, described in Part B.

B. Building-Block Realizations

Alternative realizations [A6-A7] of a discrete-time filter may be obtained from the transfer function H(z) in (2.1) by algebraic decomposition. The most common decompositions are by partial fractions and by product of factors, where both are carried as far as first- and second-degree components. These lead, respectively, to parallel-form and cascade-form realizations by first- and second-order sections (or "building blocks").

The parallel-form and cascade-form realizations were implemented by digital computer soon after the direct-form realization, in an attempt to reduce FWLEs. The individual first- and second-order sections of these realizations exhibit reduced sensitivity, hence reduced FWLEs, relative to the direct-form realization of a high-order transfer function. Even the cummulative FWLEs due to all the sections in a complete building-block realization of a high-order transfer function are less severe than the FWLEs in the corresponding direct= form realization. For this reason, building-block realizations have supplanted direct-form realizations; also, the parallel form is preferred to the cascade form. Subsequently, certain restricted second-order sections were found which guarantee the absence of zero= input limit cycles and overflow limit cycles in the forced response. In these sections, it became unnecessary to use the costly approach mentioned in the previous Section; i.e., the approach of eliminating overflow limit cycles by eliminating overflows.

3. LINEAR DISCRETE-TIME STATE-SPACE REALIZATIONS

3.1 Introduction

The state-space approach to discrete- or continuous-time systems is a general approach which combines matrix algebra and the principle of cause and effect. This approach evolved from the study of dynamical systems during the nineteenth century; the cause-and-effect principle followed naturally from Newtonian mechanics. The state-space approach involves the internal structure of a system, whereas the transfer-function approach considers only input-output properties. Lyapunov stability theory, which dates from the last decade of the nineteenth century, is a very useful part of the state-space approach. A recent resurgence of research in the state-space analysis, related to control systems, dates from the late 1950's; see, e.g., [F1-F13].

Linear state-space realizations are the prototypes for approximation by state-space digital filters. We collect here, for reference, some facts concerning linear state-space realizations.

3.2 State Equations

A linear time-invariant (discrete-time) realization has the form

$$\begin{aligned} \bar{x}(k+1) &= A\bar{x}(k) + Bu(k) \\ y(k) &= C\bar{x}(k) + Du(k), \end{aligned} \tag{3.1}$$

where $k = 0, 1, 2, \ldots$ is the discrete-time variable; the state, input, and output vectors are, respectively,

$$\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_n \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix}; \quad (3.2)$$

A, B, C, and D are real constant matrices of dimension n x n, n x m, r x n, and r x m, respectively. The coefficients of the matrices A, B, C, and D and the components of the vectors u, $\bar{x}$, and y (i.e., the signals) will eventually be quantized for a digital-filter realization.

3.3 Transfer-Function Matrix

The (z-domain) transfer-function matrix [F6]

$$G(z) = C(zI - A)^{-1} B + D \quad (3.3)$$

follows from (3.1). We assume that A is a stability matrix, since an unstable system is not useable. The "frequency response" from a given input to a given output is obtained by evaluating the corresponding entry of G(z) on the unit circle $z = e^{j\omega}$. The approximation problem in this frequency domain consists of approximating a desired function of ω. The basic limitation on the accuracy of the approximation is the order of the transfer function. A minimal realization is a realization (3.1) of the lowest possible order n which provides a specified transfer-function matrix G(z). It is well known [F2] that a realization is minimal if, and only if, it is controllable and

observable. Minimal realizations eliminate redundancies in the state equations; hence, they are preferred as prototypes for state-space digital filters.

3.4 Transformation of State Variables

A most useful concept is the linear transformation of state variables

$$\bar{x} = Tx^{o} \tag{3.4}$$

where T is a non-singular matrix. This transformation (3.4) is equivalent to a similarity transformation on the system (3.1); i.e.,

$$\begin{aligned} A_0 &= T^{-1} AT \\ B_0 &= T^{-1} B \\ C_0 &= CT \\ D_0 &= D \end{aligned} \tag{3.5}$$

would replace A, B, C, and D, respectively. The transfer-function matrix G(z) is invariant under the transformation (3.4) and (3.5). Conversely, if two minimal realizations have the same transfer-function matrix, then they are related by a transformation satisfying (3.4) and (3.5). Note that if either of the realizations {A, B, C} or $\{A_0, B_0, C_0\}$ in (3.5) is minimal, then the other realization is also minimal.

3.5 Lyapunov Stability Theory for Linear Realizations

Here we summarize some well-known results from stability theory [F11], which will be used later. Lyapunov stability theory reduces the stability analysis of the realization (3.1) to the determination of a single decreasing positive-definite scalar function (Lyapunov function) of $\bar{x}(k)$, $k = 0, 1, 2, \ldots$. A quadratic Lyapunov function of the system (3.1) is, by definition, a form

$$V(k) = \bar{x}^T(k)\, W\, \bar{x}(k) \tag{3.6}$$

where W is a positive definite matrix and

$$V(k) - V(k+1) \geq 0 \quad , \quad k = 0, 1, 2, \ldots . \tag{3.7}$$

We write (3.7) as

$$V(k) - V(k+1) = \bar{x}^T(k)[W - A^T WA]\, \bar{x}(k) \geq 0, \tag{3.8}$$

which follows from (3.1) under zero-input conditions. Hence, from (3.8), a necessary condition for $x^T Wx$ to be a Lyapunov function is

$$W - A^T WA = Q \tag{3.9}$$

where Q is a positive semi-definite matrix. Equation (3.9) is known as a Lyapunov matrix equation. For any positive semi-definite matrix Q, it is well known in matrix theory [F10] that the solution W in (3.9) is positive definite if and only if A is a stability matrix, i.e., a matrix with all eigenvalues inside the unit circle. Thus, $\bar{x}^T W \bar{x}$ is a Lyapunov function if and only if W satisfies (3.9) and A is a stability matrix. By solving for W and checking for positive

definiteness, (3.9) is often used to test the stability of A.

The linear realizations (3.1) which we will consider in this thesis are characterized in the following way, before coefficient quantization. A is known to be a stability matrix and the solution W in (3.9) has (positive-definite) diagonal form for the specialization

$$Q = C^T C. \tag{3.10}$$

That is, we will deal exclusively with the particular Lyapunov matrix equation

$$W - A^T W A = C^T C; \tag{3.11}$$

note that

$$V(k) - V(k+1) = y^T y \quad , \tag{3.12}$$

by (3.8), (3.11), and (3.1) under zero-input conditions. We will consider the non-linear state-space digital filter resulting from signal quantization in (3.1). Clearly, (3.7) still holds under conditions of magnitude truncation at the state variables. We will show that V(k) remains a decreasing function, i.e., it satisfies (3.7), in the non-linear system, if the solution W in (3.11) retains sufficient diagonal dominance after coefficient quantization in A and C. This result and other results concerning stability are derived in Chapter 6.

We also mention the Lyapunov matrix equation

$$K - AKA^T = BB^T \quad , \tag{3.13}$$

which corresponds to the adjoint of system (3.1). The linear realizations (3.1) which we will consider in this thesis are also characterized by a diagonal solution K in (3.13). The solution K may be used to determine dynamic-range constraints at the state variables, as described in Section 5.4.

The specific linear realizations considered in this thesis, which are characterized by diagonal solutions K and W in (3.13) and (3.11), are the wave-digital realization and the minimum-roundoff-noise realizations [C10]. For wave-digital realizations, the diagonal solutions K and W are shown in Section 5.6. In the minimum-roundoff-noise realization, W is used as the indicator of roundoff noise and K is used to set a dynamic-range constraint. For the optimal-word-length case, diagonal solutions K and W are necessary and sufficient conditions for minimum roundoff noise. We show in Section 5.5 that near-minimum roundoff noise is obtained if K and W retain sufficient diagonal dominance after coefficient quantization in A, B, and C.

4. STATE-SPACE DIGITAL FILTERS

4.1 Introduction

The prevailing interest in the state-space approach to continuous- and discrete-time systems during the 1960s led quite naturally to its application to digital filters. The state-space approach provides a general method for the analysis and synthesis of digital filters. Digital-filter synthesis by this method consists of: (1) synthesis of a linear state-space realization which provides the desired frequency response (i.e., the desired z-domain transfer function); (2) coefficient and signal quantization of the linear realization. Essentially, then, the state-space digital-filter realization is an approximation to the linear realization, where the errors are the finite-word-length effects (FWLEs) due to coefficient and signal quantization. (Here "signal" refers to state variables as well as inputs and outputs.)

The principal objective of the state-space approach is to determine possible low-sensitivity realizations under a given transfer-function constraint; these realizations would exhibit reduced FWLEs due to coefficient and signal quantization. Thus, it was hoped that the state-space approach would provide an alternative to the costly direct method of reducing FWLEs, i.e., an alternative to increasing coefficient and signal word lengths. Also, the possibility of limit-cycle suppression by means of appropriate state-space realizations could not be discounted. Finally, the state-space approach is a useful analytic tool for the study and comparison of FWLEs in various realizations; it includes the direct (transfer-function) realization as a special case.

4.2 Preliminary Realization Problem

The preliminary realization problem for a state-space digital filter is to determine a linear discrete-time state-space realization (3.1) which provides the desired z-domain transfer function (2.1). We assume that the desired z-domain transfer function has been determined by solving the frequency-domain approximation problem discussed in Section 2.4. This problem has been treated extensively in the literature [B1-B5]. A specific state-space realization (3.1), which provides the desired transfer function, follows immediately from the transfer-function coefficients by use of the companion matrix; this corresponds to the direct form mentioned in Section 2.6A. There are an infinite number of realizations which provide a desired transfer function; these may be obtained from a specific realization by the non-singular transformation of state variables discussed in Section 3.4. Also, for low-order (e.g., 2nd-order) realizations, explicit formulae are feasible. Thus, the transfer function alone does not determine a unique realization.

4.3 Digital Realization: Finite-Word-Length Effects

A state-space digital filter is an approximation to a prototype linear discrete-time state-space realization, where the prototype is the solution to the preliminary realization problem discussed in the previous Section. The digital implementation of the prototype introduces errors due to unavoidable finite-word-length limitations on the coefficients and signals. The principal problem is to determine a prototype realization which reduces FWLEs.

In other words, we desire a prototype realization which reacts favourably to both coefficient and signal quantization, i.e., a low-sensitivity prototype realization. The direct-form realization is seldom satisfactory in this regard. Clearly, we must choose a realization from the set of realizations which satisfies the transfer-function requirement.

The nature of finite-word-length effects has been previously described in Section 2.5; that description applies verbatim to state-space digital filters. From that description, we conclude that reduced finite-word-length effects are exhibited by state-space realizations which possess low pole-zero sensitivity of the transfer function to coefficient variations and low sensitivity to time-domain errors at the signals. The elimination of limit cycles will be achieved by means of Lyapunov stability theory. A related discussion appeared earlier in Section 1.2 of the introductory Chapter.

In Sections 4.4 and 4.5, a non-linear model is given for the state-space digital filter which results from signal quantization. The nature of the time-domain errors and of the time-domain sensitivity in the state-space digital filter is rendered somewhat more precise by means of this model. This model will also be utilized in obtaining the results on stability in Chapter 6. It is assumed that coefficient quantization has first been performed while retaining an acceptable frequency response. A more detailed consideration of coefficient quantization is delayed until Chapter 7; this is of no particular concern here, since coefficient quantization does not introduce non-linearities in the state-space realization.

4.4 Signal Quantization

A. General Considerations

A digital realization which approximates the ideal linear system (3.1) requires signal quantization, i.e., a finite-precision signal representation. For definiteness, we assume that m_i significant bits (excluding the sign) are available for the storage and delay of the i^{th} state variable, $i = 1, 2, \ldots, n$. However, the computation of the exact next-state variables by means of (3.1) may generate overflow and underflow bits. Thus, it is essential to establish a well-defined rule for approximating the exact next-state variable x_i by an m_i- significant-bit integer, $i = 1, 2, \ldots, n$.

B. Notation and Conventions

For convenience, we scale all variables to integer form. That is, we assume the present-state variables $\hat{x}_i(k)$ are m_i-significant-bit integers, $i = 1, 2, \ldots, n$, and scale them by a factor 2^N. N is chosen such that the exact next-state variables $x_i(k + 1)$, $i = 1, 2, \ldots, n$, will also be integers. Thus, we require $N = \max_i N_i$, where N_i is the maximum number of bits to the right of the radix point in any coefficient of the i^{th} row of the quantized matrix A. Hence, for $x_i(k + 1)$ we allot $m_i + N$ non-overflow bits; only the first $m_i + N_i$ of these bits may be non-zero, as seen from the definition of N_i. The non-zero bits beyond the first m_i bits are underflow bits.

We allot an additional P_i overflow bits to the left of the $m_i + N$ non-overflow bits in x_i, $i = 1, 2, \ldots, n$. P_i is chosen to accomodate the maximum (scaled) value that can occur at x_i. R_i

denotes this maximum possible value at x_i; for zero-input

$$R_i = 2^N \sum_{j=1}^{n} |a_{ij}| \, (2^{m_j} - 1). \tag{4.1}$$

In view of the above, we define the following sets of integers:

a) X_i is the set of all possible $P_i + m_i + N$-bit integers, $i = 1, 2, \ldots, n$, such that the last $N - N_i$ bits are zero. (The sign bit is not counted here.)

b) $\tilde{X}_i$ is the subset of X_i consisting of those members of X_i where the first P_i bits are zero, $i = 1, 2, \ldots, n$. This is the "non-overflow" set.

c) $\hat{X}_i$ is the subset of $\tilde{X}_i$ consisting of those members of $\tilde{X}_i$ where the last N bits are zero, $i = 1, 2, \ldots, n$. This is the "non-overflow= non-underflow" set.

We denote the members of the sets, X_i, $\tilde{X}_i$, and $\hat{X}_i$ by x_i, $\tilde{x}_i$, and $\hat{x}_i$, respectively. Thus

$$x_i \in X_i, \tag{4.2}$$

$$\tilde{x}_i \in \tilde{X}_i, \tag{4.3}$$

and

$$\hat{x}_i \in \hat{X}_i, \tag{4.4}$$

for $i = 1, 2, \ldots, n$. Note that (4.2) and (4.4) are consistent with our previous definitions of x_i and $\hat{x}_i$.

In lexicographic notation, x_i, $\tilde{x}_i$, and $\hat{x}_i$ will have the forms

$$x_i = \pm \underbrace{ZZ \ldots Z}_{P_i \text{ bits}} \underbrace{ZZ \ldots Z}_{m_i \text{ bits}} \underbrace{ZZ \ldots Z}_{N_i \text{ bits}} 00 \ldots 0_{\Delta} \qquad (4.5)$$

$$\tilde{x}_i = \pm \underbrace{00 \ldots 0}_{P_i \text{ bits}} \underbrace{ZZ \ldots Z}_{m_i \text{ bits}} \underbrace{ZZ \ldots Z}_{N_i \text{ bits}} 00 \ldots 0_{\Delta} \qquad (4.6)$$

and

$$\hat{x}_i = \pm \underbrace{00 \ldots 0}_{P_i \text{ bits}} \underbrace{ZZ \ldots Z}_{m_i \text{ bits}} \underbrace{00 \ldots\ldots 0_{\Delta}}_{N \text{ bits}} \qquad (4.7)$$

Here Z denotes a bit which may be 0 or 1 and Δ denotes the radix point. Note that $\hat{x}_i$ is the m_i-significant-bit integer which will replace x_i.

$\tilde{R}_i$ and $\hat{R}_i$ will denote the range of magnitude in $\tilde{X}_i$ and $\hat{X}_i$, respectively. A simple calculation gives

$$\tilde{R}_i = 2^{m_i + N} - 2^{N - N_i} \qquad (4.8)$$

and

$$\hat{R}_i = 2^{m_i + N} - 2^{N}, \qquad (4.9)$$

for i = 1, 2, ..., n. R_i, which denotes the range of magnitudes <u>actually used</u> in X_i, has already been given in (4.1). Thus we have

$$|x_i| \leq R_i, \qquad (4.10)$$

$$|\tilde{x}_i| \leq \tilde{R}_i, \qquad (4.11)$$

and

$$|\hat{x}_i| \leq \hat{R}_i. \tag{4.12}$$

$\bar{N}_i(x_i)$ will denote the non-linear function which approximates a computed variable x_i by $\hat{x}_i$; i.e.,

$$\hat{x}_i = \bar{N}_i(x_i) \quad , \quad i = 1, 2, \ldots, n. \tag{4.13}$$

We regard $\bar{N}_i(x_i)$, $i = 1, 2, \ldots, n$, as the composite of two functions, the overflow function $\theta_i(x_i)$ and the underflow function $U_i(x_i)$:

$$\hat{x}_i = \bar{N}_i(x_i) = U_i \circ \theta_i(x_i). \tag{4.14}$$

This allows us to treat the overflow and underflow cases separately and introduces no loss of generality. We characterize the overflow function by its domain X_i and its range $\tilde{X}_i$:

$$\tilde{x}_i = \theta_i(x_i) \quad , \quad i = 1, 2, \ldots, n \ , \tag{4.15}$$

i.e., θ_i must produce a non-overflow result for any argument x_i; we also require

$$\tilde{x}_i = \theta_i(\tilde{x}_i) \ , \quad i = 1, 2, \ldots, n \ , \tag{4.16}$$

i.e., θ_i does not alter any non-overflow variable $\tilde{x}_i$.
It is clear from (4.14) and (4.15) that

$$\hat{x}_i = U_i(\tilde{x}_i) \quad , \quad i = 1, 2, \ldots, n \ , \tag{4.17}$$

which characterizes the underflow function U_i by its domain $\tilde{X}_i$ and its range $\hat{X}_i$. Particular underflow and overflow characteristics are discussed in Sections 5.2 and 5.3, respectively.

The extension of the previous notation to vectors is clear:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad ; \quad \tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix} \quad ; \quad \hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_n \end{bmatrix} \quad . \tag{4.18}$$

X, $\tilde{X}$, and $\hat{X}$ are the sets of all possible x, $\tilde{x}$, and $\hat{x}$, respectively.

Also

$$\bar{N}(x) = \begin{bmatrix} \bar{N}_1(x_1) \\ \bar{N}_2(x_2) \\ \vdots \\ \bar{N}_n(x_n) \end{bmatrix} ; \quad \theta(x) = \begin{bmatrix} \theta_1(x_1) \\ \theta_2(x_2) \\ \vdots \\ \theta_n(x_n) \end{bmatrix} ;$$

$$U(\tilde{x}) = \begin{bmatrix} U_1(\tilde{x}_1) \\ U_2(\tilde{x}_2) \\ \vdots \\ U_n(\tilde{x}_n) \end{bmatrix} \tag{4.19}$$

Thus

$$\hat{x} = \bar{N}(x) = U \circ \theta(x), \tag{4.20}$$

where

$$\tilde{x} = \theta(x) \tag{4.21}$$

and

$$\hat{x} = U(\tilde{x}) \, . \tag{4.22}$$

The designation of a variable by x_i, $\tilde{x}_i$, or $\hat{x}_i$ will serve a dual purpose: (1) it will restrict the domain of the variable, as given by (4.2), (4.3), or (4.4); (2) it will impose the appropriate function relation, (4.14), (4.15), or (4.17), on any two of the variables x_i, $\tilde{x}_i$, and $\hat{x}_i$ which appear in the same equation. Similar remarks apply to the vector extensions, x, $\tilde{x}$, and $\hat{x}$, of these variables and to the vector function relations (4.20), (4.21), and (4.22). We will refer to the functions (4.20), (4.21) and (4.22) as the mappings x to $\hat{x}$, x to $\tilde{x}$, and $\tilde{x}$ to $\hat{x}$, respectively.

C. Alternative Signal Representations

The preceding notation has been defined in terms of the sign-magnitude signal representation, which will serve as a vehicle to derive the stability results in Chapter 6 for systems with finite-precision signal representation. These results will be derived under certain assumptions on the overflow and underflow characteristics, i.e., certain assumptions on the mappings x to $\tilde{x}$ and $\tilde{x}$ to $\hat{x}$. However, other signal representations are permissible for actual hardware realization. We may use any new representation by which it is possible to preserve the mappings x to $\tilde{x}$ and $\tilde{x}$ to $\hat{x}$ under a one-to-one correspondence between elements of the new representation and the sign-magnitude representation. In fact, we expect that the two's complement, rather than the sign-magnitude, representation will be used in practice because of its well-known hardware advantages.

4.5 Models of State-Space Digital Filters

A digital filter may now be characterized by the non-linear state equations

$$\begin{aligned} x(k+1) &= A\,\hat{x}(k) + B\,u(k) \\ y(k) &= C\,\hat{x}(k) + D\,u(k) \\ \hat{x}(k) &= \bar{N}\,(x(k)) \\ k &= 0, 1, 2, \ldots, \end{aligned} \tag{4.23}$$

which include the FWLE due to signal quantization. These equations (4.23) follow from the original system (3.1) by the introduction of the non-linear signal quantization function (4.20); the notation here is that given in Section 4.4B. The non-linear function $\bar{N}$ reflects the elimination of overflow and underflow bits in $x(k+1)$, to obtain $\hat{x}(k+1)$. Coefficient quantization is simply reflected by a change in the entries of A, B, C, and D.

We may also model the state-space digital filter by a linear system which is perturbed by a sequence of state-quantization error vectors $p(k)$, $k = 0, 1, 2, \ldots$. That is, (4.23) is equivalent to

$$\begin{aligned} \hat{x}(k+1) &= A\,\hat{x}(k) + p(k) + B\,u(k) \\ y(k) &= C\,\hat{x}(k) + D\,u(k) \\ k &= 0, 1, 2, \ldots, \end{aligned} \tag{4.24}$$

where we define

$$p(k) \triangleq \hat{x}(k+1) - x(k+1). \tag{4.25}$$

This clearly exhibits the FWLE due to signal quantization, because the system (4.24) is identical in form to the original system (3.1),

except for the presence of the perturbing vector $p(k)$, $k = 0, 1, 2, \ldots$. Thus, the FWLE due to signal quantization is a time-domain error which is the response to the error vector $p(k)$, $k = 0, 1, 2, \ldots$. This error response is usually analyzed by a combination of the principle of linear superposition and probability theory, since the components of $p(k)$, $k = 0, 1, 2, \ldots$, are random in nature. The error response can also be bounded by the bounded-input-implies-bounded=output property of linear systems. In this regard, it is useful to decompose $p(k)$ into the sum

$$p(k) = c(k) + d(k) , \tag{4.26}$$

where

$$c(k) \triangleq \tilde{x}(k+1) - x(k+1) \tag{4.27}$$

is the overflow error vector, and

$$d(k) \triangleq \hat{x}(k+1) - \tilde{x}(k+1) \tag{4.28}$$

is the underflow error vector.

Finally, we render certain common notions more precise by means of the model (4.24). The possible error responses may be classified as follows: (1) roundoff noise is the error response corresponding to random components in $p(k)$, in particular to random components in $d(k)$; (2) a limit cycle is the error response corresponding to periodic and self-sustaining components in $p(k)$. Limit cycles may correspond to underflows, overflows, or a combination of both. Most [D3, D5, D7-D9] of the bounds on limit cycles and

roundoff noise which have appeared in the literature are applications of the bounded-input-implies-bounded-output property. Examples of the limit-cycle problem are given in [D12, D13, D16]. The notion of low time-domain sensitivity to signal disturbances is also clarified by examining (4.24). The appropriate interpretation of low time-domain sensitivity is that a unit sample disturbance in a component of $p(k)$ should cause an output response sequence which has as small an effective value as possible. This will lead to reduced roundoff noise at the output.

4.6 Survey of Recent Research in State-Space Digital Filters

The initial application of the state-space approach to digital filters was confined to various second-order sections which were intended as building blocks. These second-order sections began to appear in the literature about 1967. Rader and Gold [A7] proposed the coupled-loop structure with the intention of obtaining reduced sensitivity to coefficient quantization. Sandberg [D1], Ebert et al. [D2], and Wilson [D4, D6] proposed certain restricted second-order sections which excluded the possibility of zero-input cycles; the Lyapunov approach was used in these cases. It was found [D15] much later that the coupled-loop structure also has a useful relation to the Lyapunov theory.

Fettweis [G1-G6, G8-G11, G13] introduced the wave digital filter in 1971. It is helpful to consider the wave digital filter from the state-space viewpoint, although the original disclosure was given in terms of building blocks called adaptors. The wave digital filter

derives from the transformation of an analog prototype filter, described by scattering (i.e., wave) variables, to the discrete-time domain. The transformation (from the s-domain to the z-domain) is performed by using the bilinear transformation. The wave digital filter has two significant features: low sensitivity to coefficient variations is implied by its network origin; suppression of zero-input limit cycles follows from magnitude truncation, due to the existence of a diagonal Lyapunov function.

It is well known that low sensitivity and low roundoff noise are related [C3, C4, C8, C11]. For example, experimental results confirm that the wave digital filter exhibits low roundoff noise as well as low sensitivity [G7, G12].

Mullis and Roberts [C10] introduced minimum-roundoff-noise (fixed-point) realizations in 1976. This is a general analysis= synthesis result, which uses the Lyapunov matrix equations $K - AKA^T = BB^T$ and $W - A^TWA = C^TC$. For the optimal-word-length case, it is shown [C10] that a necessary and sufficient condition for minimum roundoff noise in a realization of any order is that the solutions K and W are both diagonal. The underlying assumption is uniform and uncorrelated probability distributions of the roundoff errors at the state variables; a dynamic-range constraint, achieved by scaling, is also included to limit overflow probabilities. Jackson et al. [C11] have applied the minimum-roundoff-noise result to second-order sections.

Various other results based on Lyapunov theory have also appeared. For example: Barnes and Fam [D14] have introduced minimum= norm digital filters; Mills et al. [D15] have given realizations

which are free of overflow limit cycles.

As seen from the above survey, the Lyapunov approach, in its various forms, has led to some success in obtaining state-space realizations with reduced FWLEs. For example, it has led to the suppression of zero-input limit cycles in wave digital filters and to minimum-roundoff-noise realizations. The relation between low pole-zero sensitivity to coefficient variations and the Lyapunov approach requires further investigation. On the other hand, it is difficult to find in the literature another approach to reducing FWLEs which has as much generality, or has had as much success, as the Lyapunov approach.

4.7 A General Approach to Reducing Finite-Word-Length Effects

A broad outline of the research in this thesis has already been given in Sections 1.2 and 1.3 of the Introduction. Here we elaborate on that outline to present a more complete overview of the research which follows in Chapters 6 through 9.

The importance of low pole-zero and low time-domain sensitivities in a prototype state-space realization has been stressed in the Introduction. Low pole-zero sensitivity makes possible the use of short coefficient word lengths, while maintaining a tolerable error in the frequency response. Low time-domain sensitivity makes possible the use of reduced signal word lengths, while maintaining an acceptable level of roundoff noise; the generation of roundoff noise has been modelled in Section 4.5. If we define a suitable cost measure, such as the total bit-adder product in a digital realization, then low= sensitivity prototype realizations lead to reduced cost in the

corresponding digital realization. It is appropriate to mention that an additional dynamic-range constraint must be imposed on the signals to limit the probability of overflow [C10]; this effectively restricts the improvement in signal-to-roundoff-noise ratio which can be obtained by simple scaling of signal levels. The elimination of limit cycles is the third important problem mentioned in the Introduction. In fact, we may assign the greatest priority to the elimination of overflow limit cycles, since these may render a digital filter unuseable. Unless overflow limit cycles are initially "designed out", the only remedy in this event is to shut down the digital filter and "start over". Such a shut down is very undesirable in terms of signal loss or if the filter is in a remote location. On the other hand, we may always resort to the simple, although costly, remedy of increasing coefficient and signal word lengths after the initial design, to deal with the problems of sensitivity. Also, underflow limit cycles are not nearly as serious a problem as overflow limit cycles, since underflow limit cycles resemble roundoff noise; however, the effective value of this disturbance may be considerably greater than that of roundoff noise.

It would be most desirable to find that low pole-zero sensitivity, low time-domain sensitivity, and elimination of limit cycles are interrelated, i.e., to find that these properties occur simultaneously in a state-space realization. There are some indications that this is indeed the case. Connections between low pole-zero sensitivity and low time-domain sensitivity have been given in the literature, as mentioned in the survey in Section 4.6. However,

a completely general direct analytic connection has not yet been given; hopefully, such a connection will eventually be found. It is also plausible that susceptibility to limit cycles decreases as time-domain sensitivity decreases, since a limit cycle is a self-sustaining time-domain disturbance; this interpretation of a limit cycle was discussed in Section 4.5. As described below, this plausibility argument is given substance by means of the Lyapunov approach, where it is shown that certain realizations with low time-domain sensitivity give suppression of limit cycles, under conditions of magnitude truncation at the signals.

The Lyapunov approach to obtaining realizations with low sensitivity and elimination of limit cycles is chosen here for various reasons mentioned in Section 1.2. If we take the literature survey in Section 4.6 as a point of departure, it is clear that this approach holds some promise. In particular, two of the more significant results in recent literature, i.e., the wave digital filter and minimum-roundouff-noise (MRN) realization, have close ties to the Lyapunov theory. As described in the Introduction, we have consolidated the Lyapunov theory pertaining to wave-digital and minimum-roundoff-noise realizations, by showing that the solutions K and W to the Lyapunov matrix equations are diagonal in both cases. This result and the appropriate interpretation of the wave-digital realization as a two-input-two-output MRN realization are derived in Sections 5.6 and 5.7. Since the wave-digital realization thus fits into the framework of the MRN realization, in the sequel the term MRN realization will be used to include both types of realizations. Consequently, the reader may

interpret as applicable to the wave-digital realization the results obtained in the sequel by extension of the Lyapunov approach in the MRN realization. This includes the results pertaining to roundoff noise, stability, and sensitivity in Chapters 5, 6, and 7, respectively. We also note that the principles involved in the MRN realization are sufficiently general to include most of the other results in the survey in Section 4.6 which pertain to the Lyapunov approach; e.g., the results pertaining to the elimination of zero-input limit cycles.

The desirable consequences of diagonal solutions K and W in the MRN realization have been listed in Section 1.2. However, only the case of pure-diagonal solutions K and W has been considered in the literature; cf. Section 4.6. Since coefficient quantization in the state equations will normally result in the loss of these pure-diagonal solutions, because they derive from state equations with unrestricted (real-number) coefficients, the following question naturally arises. Is it possible to extend the desirable consequences of diagonal solutions to near-diagonal (i.e., diagonally dominant) solutions K and W? This question is answered in the affirmative in Chapters 5, 6, and 7, by what amounts to an approximation theory. This overcomes the obstacle posed by coefficient quantization.

An arbitrarily close approximation to the original pure= diagonal solutions can be obtained in K and W by sufficiently fine coefficient quantization, because the entries of K and W are continuous functions of the coefficients of the realization; this follows from the Lyapunov matrix equations. This implies that an arbitrary degree of diagonal dominance can be obtained in K and W. However, it

is very desirable that K and W be relatively insensitive to coefficient variations in A, B, and C, so that excessive coefficient word lengths can be avoided. For this reason, it is shown by analysis, in Chapter 7, that low pole-zero sensitivity (i.e., low sensitivity of the frequency response) in the state-space realization {A, B, C} implies low sensitivity of the solutions K and W to coefficient variations in {A, B, C}.

It is tempting to conjecture that diagonal K and W imply low pole-zero sensitivity to coefficient quantization. This would be very expedient for obtaining diagonally dominant solutions K and W after coefficient quantization, in view of the discussion in the previous paragraph. It is known that low or minimum time-domain sensitivity and diagonal K and W are equivalent, by the analysis of the MRN realization [C10]. Thus, if the connection between low pole-zero sensitivity and low time-domain sensitivity can be established with sufficient generality, the above conjecture will in fact be valid. The wave-digital realization is an example where diagonal K and W, low time-domain sensitivity, and low pole-zero sensitivity occur simultaneously.

We now summarize the concepts involved in the "approximation theory" which is developed in Chapters 5 through 9. This approximation theory extends the results obtained by diagonal K and W in MRN realizations to the case where K and W are only diagonally dominant due to coefficient quantization. This provides a means of passing directly from a low-sensitivity linear state-space MRN realization containing ideal (i.e., real-number) coefficients to a corresponding

realization containing quantized (binary) coefficients. The criterion for the acceptability of the realization with quantized coefficients is the degree of diagonal dominance in the solutions K and W; a precise measure of diagonal dominance is given in Chapter 6. If the degree of diagonal dominance in K and W is sufficient, the properties of near-minimum roundoff noise, zero-input limit-cycle suppression, and stability of the forced response will be retained. The net result is to relax the inherent restrictions in the MRN realization, for the purpose of hardware implementation.

The carry-over of results from the pure-diagonal to the diagonally dominant case is achieved in the following way. In the case of time-domain sensitivity (i.e., roundoff noise), a relatively simple extension of the analysis [C10] of the MRN realization is carried out in Sections 5.4 and 5.5. It is shown there that near= minimum roundoff noise will be maintained if the change in K and W is sufficiently small. In fact, it may be desirable to accept a further moderate degradation in roundoff-noise performance, in exchange for simplified coefficients (i.e., in exchange for reduced coefficient word lengths). Low coefficient sensitivity, which is desirable at the outset to facilitate coefficient quantization, will be preserved in so much as it is related to low time-domain sensitivity.

The properties of zero-input limit cycle suppression and stability of the forced response are obtained in the diagonally dominant case, by carrying over the Lyapunov function $\bar{x}^T W \bar{x}$ from the prototype linear state-space realization to the corresponding digital realization. It is shown in Chapter 6 that this carry-over of the

Lyapunov function $\bar{x}^T W \bar{x}$ is possible if W exhibits sufficient diagonal dominance, under conditions of magnitude truncation at the state variables. (In terms of the notation of Section 4.4, magnitude truncation is required when replacing $x(k+1)$ by $\hat{x}(k+1)$ in the digital realization.) This is a significant generalization of the pure-diagonal case, which follows from the nature of the finite-precision signal representation. In Chapter 6, separate necessary and sufficient diagonal-dominance conditions are given for $\hat{x}^T W \hat{x}$ to be a Lyapunov function in the event of signal overflow and underflow truncations. The results pertaining to overflow truncations, which include stability of the forced response, rest primarily on the nature of the overflow characteristic. The results pertaining to underflow truncations follow mainly from the granularity of the signal representation.

A state-space digital filter is obtained from the coefficient-quantized linear state-space realization by a digital implementation of the required matrix multiplication. This digital-implementation problem is treated in Chapter 8; it completes the process of state-space digital-filter synthesis, which began with the determination of a low-sensitivity linear state-space realization. The digital implementation requires a suitable computing algorithm. This is a significant problem in itself, since the computing algorithm ultimately affects the magnitude of roundoff noise; i.e., it determines the sources of roundoff error. The computing algorithm used in Chapter 8 is a distributed-arithmetic structure; it is based on some previous work [H3] by the author. In this

structure, the next-state variables are computed exactly (i.e., with full precision) before overflow and underflow truncations to the allowed word length. This exact computation reduces the number of signal-quantization-error sources to a minimum.

The approach to state-space digital filters described above is a "global" approach; i.e., the digital filter is realized and implemented as one complete block. This is in contrast to the building-block approach. It is therefore appropriate to ask if this global approach offers any particular advantages over the building= block approach. At the outset, we must confine our attention to building-block realizations where the individual first- and second= order sections guarantee stability of the forced response and freedom from zero-input limit cycles; such sections have been described in the literature [D11, D14, D15]. This is necessary to obtain these same properties which are already obtained by the above-mentioned global approach. Clearly, the building block realization cannot surpass the MRN global realization in terms of roundoff-noise performance, for a given signal wordlength; in fact, the building block realization may require a significant increase in signal word length to achieve round-off-noise performance equivalent to that of the MRN realization. (Note that a fair comparison of roundoff noise requires that exact next-state variables be computed before truncation to the allowed word length, in either the building-block or global approach.) The improved roundoff-noise performance of the global realization may be offset by the increased number of coefficient multipliers that it requires. However, the question of sensitivity comes into play here;

i.e., the reduced word lengths in the coefficient multipliers will be helpful. For a given transfer function and equivalent roundoff-noise performance, it is thus likely that the building block approach requires a greater total of signal word lengths and the global approach requires a greater total of coefficient word lengths. The common basis of comparison, which includes the effects of these two factors, is the bit-adder product in the final digital implementation. (Note that the transfer functions obtained after coefficient quantizations in the two approaches may be very nearly the same, but not exactly the same, due to the different effects of the quantization processes.)

4.8 State-Space Approach to Wave Digital Filters

Here we will discuss wave digital filters from the state-space viewpoint, in order to establish background information needed in Chapters 5 and 7. The wave digital filter approximates a prototype discrete-time system which imitates the topological structure of an analog (prototype) filter network. The topological structure of the analog network is first incorporated into a set of continuous-time state equations, by the use of the scattering matrix and scattering variables. The bilinear transformation, from the s-domain to the z-domain, renders this network-derived continuous-time system to the above-mentioned prototype discrete-time system. This transformation leaves the scattering matrix invariant, but induces the following conversion of (continuous-time) network elements to discrete-time elements: capacitances to delays; inductances to

delays with sign inversion; and resistive voltage sources to wave sources and wave sinks.

A brief derivation of the state equations for wave digital filters is given in [H3]. These state equations dictate a partition of the scattering matrix S; this matrix S derives from the topology of the analog prototype filter. The edges of the network graph are ordered according to inductive, capacitive, and resistive elements, and the matrix S is partitioned according to energy-storage (i.e., inductive and capacitive) and dissipative (i.e., resistive) elements. We denote this partition by

$$S = \left[\begin{array}{c|c} S_{11} & S_{12} \\ \hline S_{21} & S_{22} \end{array}\right] , \tag{4.29}$$

where the subscripts 1 and 2 correspond to energy-storage and dissipative elements, respectively. As in [H3], we define the "polarity" matrix

$$P = \left[\begin{array}{c|c} -I_1 & 0 \\ \hline 0 & I_2 \end{array}\right] \tag{4.30}$$

where I_1 and I_2 are identity matrices of dimensions equal to the number of inductive and the number of capacitive network elements, respectively. As seen in [H3], the following correspondence exists between the submatrices of S and the matrices A, B, C, D in the standard state equations (3.1):

$$
\begin{aligned}
PS_{11} &\rightarrow A; \\
PS_{12} &\rightarrow B; \\
S_{21} &\rightarrow C; \\
S_{22} &\rightarrow D.
\end{aligned}
\tag{4.31}
$$

The discrete-time state equations for the wave digital filter bear an interesting and useful relation to the two Lyapunov equations (3.11) and (3.13). That is, for the wave-digital state equations the solutions to the Lyapunov equations are $W = G$ and $K = G^{-1}$, respectively, where G is the diagonal matrix of port reference conductances for the scattering matrix. This is shown in Section 5.6. The Lyapunov equation (3.11) guarantees the suppression of zero-input limit cycles in the digital realization if magnitude truncation is used at the state variables. This follows since $\bar{x}^T W \bar{x}$ is a Lyapunov function for the discrete-time realization, where $W = G$ is diagonal. In fact, this is the Lyapunov function which is normally used in the wave digital filter.

The Lyapunov equations (3.11) and (3.13) are instrumental in the minimum-roundoff-noise realizations discussed in Section 5.4. It is mentioned there that necessary and sufficient conditions for minimum roundoff noise in a single-input-single-output system are that the solutions K and W be diagonal. However, in the wave-digital case, these Lyapunov equations apply to a two-input-two-output system. Although the discrete-time state equations of the wave digital filter clearly satisfy the formalism of Lyapunov equations with diagonal solutions K and W, one cannot assume that the original result, i.e., minimum roundoff noise in the single-input-single=

output case, will be valid in the two-input-two-output case. In Section 5.7, the original minimum-roundoff-noise result is extended in a manner which is appropriate to this two-input-two-output case. This extension is obtained by reinterpreting the minimization process in [C10], for the two-input-two-output case. The extension of the original result can be stated as follows for the two-input-two-output case: the roundoff-noise minimization applies to a weighted sum of roundoff noise variances at both outputs; and the dynamic-range constraint at each state variable applies to a weighted sum of signals due to each input.

For wave digital filters, it is well known that we can obtain binary coefficient quantization in the state equations by binary multiplier quantization in Fettweis' adaptors [G13]. However, scaling of state variables is required to give a legitimate evaluation of roundoff noise by limiting the probability of overflow [C2, C10, G12]. Scaling would be implemented by ideal transformers in the analog prototype network, where each requires a multiplication and a division by the corresponding scale factor. Hence, the scale factors would be restricted to powers of two, so that it might be difficult to obtain an overall scaling which realizes the potential roundoff-noise performance. Also, certain more complex networks [G15, G16], which are not representable by combinations of series, parallel, and lattice adaptors, tend to give fairly long coefficient word lengths. Thus, for scaling and the more complex networks, the results mentioned in Section 4.7 concerning diagonally dominant K and W provide a useful alternative approach to coefficient quantization. That is, we may

quantize the coefficients of the wave-digital state equations directly, subject to a diagonal-dominance criterion on the solutions K and W; this bypasses the preliminary quantization by adaptors. On the other hand, it is also possible that some reduction in coefficient word lengths could be obtained, by starting from relatively short coefficient word lengths which have already been determined by adaptors to give a diagonal K and W.

5. ROUNDOFF NOISE

5.1 Introduction

In this Chapter, we outline a general result pertaining to the roundoff-noise problem; this result is the minimum-roundoff-noise (MRN) realization, introduced by Mullis and Roberts [C10]. A preliminary discussion of relevant aspects of underflow and overflow quantization errors is given here. A logical extension of the minimum-roundoff-noise realization to near-minimum roundoff noise is obtained here for the case where the solutions K and W to the Lyapunov matrix equations are only diagonally dominant. This extension is necessary to cover the approximations introduced by coefficient quantization, since the original minimum-roundoff-noise realization [C10] is based on pure-diagonal K and W. It is then shown that the Lyapunov equations of the wave-digital realization correspond to those of the MRN realization for a two-input-two-output system. As a result, the wave digital realization admits an appropriate interpretation as a two-input-two-output MRN realization, by means of the Lyapunov equations. Thus, the roundoff-noise aspect of the Lyapunov approach is sufficiently developed to meet the requirements of coefficient quantization.

5.2 Quantization Errors Due to Underflows

The output error due to underflows is the effect of the underflow error vector $d(k)$, $k = 0, 1, 2, \ldots$, which was defined in (4.28). This clearly follows from (4.24), (4.26), and (4.28). Thus, it is also clear that the mapping $\tilde{x}_i$ to $\hat{x}_i$, $i = 1, 2, \ldots, n$, should be chosen to minimize the magnitudes of the components of $d(k)$, $k = 0, 1, 2, \ldots$, in order to minimize the resulting output disturbances. This is normally done by rounding $\tilde{x}_i$ to the nearest m_i-bit integer $\hat{x}_i$, $i = 1, 2, \ldots, n$.

It will be necessary to impose magnitude truncation on the mapping $\tilde{x}_i$ to $\hat{x}_i$, $i = 1, 2, \ldots, n$, in order to suppress zero-input underflow limit cycles, as shown later. That is, we require $|\hat{x}_i| \leq |\tilde{x}_i|$. Under this restriction, we minimize the magnitudes of the components of $d(k)$, $k = 0, 1, 2, \ldots$, by "rounding down" to the nearest m_i-bit integer, $\hat{x}_i$, having lesser magnitude.

The output error due to underflows is given a probabilistic analysis by making certain assumptions on the error sequence $d(k)$, $k = 0, 1, 2, \ldots$. Usually, the components of $d(k)$, i.e., the disturbances at x_i, $i = 1, 2, \ldots, n$, are assumed to have uniform uncorrelated probability distributions, although there is some evidence [C7] that this is not always true. The probability distribution lies within plus and minus one-half the least-significant bit in $\hat{x}_i$ for the simple rounding characteristic, or between zero and the least-significant bit in $\hat{x}_i$ for the magnitude-truncation characteristic. With magnitude truncation some correlation will exist between $d(k)$ and $x(k)$, so it is more accurate to remove the corre-

lated component of d(k) when using any analysis based on uncorrelated, uniformly distributed roundoff errors [C6, C9]. It is also well known that the variance of the components of d(k) increases by a factor of four for magnitude truncation versus rounding, neglecting correlation with the signal.

Magnitude truncation can be used in digital filters possessing diagonal Lyapunov functions to suppress zero-input underflow limit cycles, e.g., [G11]. However, this increases the variance at d(k) as mentioned above. Thus, such suppression of zero-input underflow limit cycles is only worthwhile if the output error variance due to potential limit cycles is several times greater than the output error variance due to simple rounding errors. This last condition may warrant further investigation, since roundoff noise and underflow limit cycles are both excited by the same "bounded input" d(k), k = 0, 1, 2, ... The essential difference is that d(k) is self-sustaining and periodic in the limit-cycle case. Thus, any increase in the output error variance in a limit cycle is due only to a change in the probability distribution and correlation of the components of d(k).

5.3 Quantization Errors Due to Overflows

The output error due to overflows is the effect of the overflow error vector c(k), k = 0, 1, 2, ..., which was defined in (4.27). This is clear from (4.24), (4.26), and (4.27). Thus, the function $\theta(x)$ in (4.21) should be chosen to minimize the magnitudes of the components of c(k), k = 0, 1, 2, ... This can be done by using the saturation overflow characteristic, which is defined by

$$\tilde{x}_i = \begin{cases} +\tilde{R}_i \text{ if } x_i > \tilde{R}_i \\ \\ -\tilde{R}_i \text{ if } x_i < -\tilde{R}_i \ . \end{cases} \tag{5.1}$$

The reduction of transient output errors due to overflows is an important consideration in digital filters, even if overflow oscillations have been excluded. In order to achieve the maximum improvement in the ratio of signal to (underflow) roundoff noise, we must scale signal levels so that overflows occur with some small, but non-zero, probability [C10]. Thus we should design filters with "high tolerance" to overflows, i.e., with least transient disturbances due to overflows. The saturation overflow characteristic helps to accomplish this by minimizing the magnitude of the components in $c(k)$, $k = 0, 1, 2, \ldots$.

The elimination of overflow oscillations, in both the force-free and forced responses, has utmost priority, since these disturbances may render a digital filter unuseable. In Chapter 6, we derive conditions for the elimination of such overflow oscillations. There it is shown that the saturation overflow characteristic gives the least-stringent sufficient condition of diagonal dominance in W for the exclusion of overflow oscillations by the Lyapunov function $\hat{x}^T W \hat{x}$. This is another advantage of the saturation overflow characteristic.

The actual implementation of the saturation overflow characteristic for state-space structures of higher order has not received much attention in the literature. The saturation

characteristic is not a natural result of two's complement or sign-magnitude arithmetic, since these both produce a modulo type of result. A true saturation characteristic is required, i.e., internal overflows must not lead to false results. The two's complement system may be more amenable to saturation arithmetic, since it is self-correcting for overflows in a sequence of additions. The inclusion of shifts leads to a more difficult problem. In Chapter 8, a practical and simple method is derived to implement a true saturation overflow characteristic in the distributed-arithmetic structure chosen for implementing the state-space realizations.

There is additional evidence that supports the desireability of the saturation overflow characteristic. Various conditions have been derived for the suppression of overflow limit cycles in second-order sections [D1, D2, D4]. These results are based on a saturation overflow characteristic, or a generalization which includes it. Also, Claasen et al. [D10] have shown that, in a realization of any order, suppression of zero-input limit cycle oscillations implies suppression of overflow oscillations in the forced response, under certain conditions on the overflow characteristic. These conditions include the saturation characteristic.

5.4 Minimum-Roundoff-Noise Realizations

Mullis and Roberts [C10] have derived a necessary and sufficient condition for minimum roundoff noise in a state-space digital-filter realization with fixed-point arithmetic. Their result applies to a state-space realization of any order with a

single input and output. A condensed account of their method follows, for reference in Sections 5.5 and 5.6.

The solutions K and W of the Lyapunov matrix equations

$$K = AKA^T + BB^T \tag{5.2}$$

and

$$W = A^TWA + C^TC \tag{5.3}$$

are given by the series

$$K = \sum_{k=0}^{\infty} (A^kB)\,(A^kB)^T \tag{5.4}$$

and

$$W = \sum_{k=0}^{\infty} (CA^k)^T\,(CA^k)\,, \tag{5.5}$$

respectively. It can therefore be shown that

$$k_{ij} = f_i \cdot f_j \tag{5.6}$$

and

$$w_{ij} = g_i \cdot g_j\,, \tag{5.7}$$

where we define the following response sequences:

a) $f_i(k)$, $k = 0, 1, 2, \ldots$, is the response $x_i(k)$ due to a unit pulse at the input;

b) $g_i(k)$, $k = 0, 1, 2, \ldots$, is the response at the output due to a unit pulse at x_i.

In (5.6) and (5.7), k_{ij} and w_{ij}, $i, j = 1, 2, \ldots, n$, denote the elements of K and W, respectively, and the dot ($\cdot$) denotes the inner product of sequences. The diagonal elements of K and W are, from

(5.6) and (5.7),

$$k_{ii} = f_i \cdot f_i = \left|\left| f_i \right|\right|^2 \tag{5.8}$$

and

$$w_{ii} = g_i \cdot g_i = \left|\left| g_i \right|\right|^2. \tag{5.9}$$

It can also be shown that: $\left|\left|f_i\right|\right|^2$ is the variance at x_i due to a unit-variance uncorrelated input; $\left|\left|g_i\right|\right|^2$ is the output variance due to a unit-variance uncorrelated disturbance at x_i. The disturbances at x_i, $i = 1,2,\ldots, n$, are the roundoff errors $p(k)$, $k = 0,1,2,\ldots$. The notation in (5.2) through (5.9) agrees with that in [C10].

An underlying assumption in [C10] is a dynamic-range constraint at x_i, $i = 1,2,\ldots,n$, to limit the probability of overflow. This constraint is essential; no legitimate roundoff-noise comparison between different realizations can be made without it [C2, G12]. The constraint is imposed by requiring $k_{ii} / 2^{m_i}$, $i = 1,2,\ldots,n$, to be a particular constant, i.e., the variance $\left|\left|f_i\right|\right|^2$ is fixed relative to the unscaled range 2^{m_i} of numerical representation. There is also a constraint on the total word length, i.e., $\sum_{i=1}^{n} m_i$ is a constant. In [C10] the initial realization is scaled to meet the dynamic-range constraint by a similarity transformation with a diagonal matrix.

The output roundoff-noise variance is given by $\sum_{i=1}^{n} w_{ii}$,

as seen from (5.9), assuming uncorrelated unit-variance disturbances at x_i, $i = 1,2,\ldots,n$. In [C10], it is shown that $\sum_{i=1}^{n} w_{ii}$ will be minimum in the scaled realization with optimal word lengths m_i if and only if K and W are diagonal in the original, unscaled realization. (This is the previously mentioned necessary and sufficient condition.) This result follows from a comparison of all possible minimal realizations possessing a given transfer function; these realizations are generated by arbitrary similarity transformations. Since determinant (KW) is invariant under a similarity transformation, the proof of the minimum is achieved by applying the Hadamard determinant inequality and the arithmetic-geometric-mean inequality.

The synthesis of a realization with diagonal K and W can always be achieved. It is equivalent to the simultaneous diagonalization of two positive-definite matrices (K and W) by a congruence transformation, which is a well-known classical problem. Due to the use of the arithmetic-geometric-mean inequality in the minimization, all w_{ii} will be equal in the scaled realization under the optimal-word-length condition. It appears that this minimization of $\sum_{i=1}^{n} w_{ii}$ also tends to minimize output error due to random overflows. In an arbitrary scaled realization, the ratio of $\sum_{i=1}^{n} w_{ii}$ to its theoretical minimum is given by $[e(K)e(W)]^{-2/n}$ in the arbitrary realization either before or after scaling. Here

$$e(P) = \left(\det P / \prod_{i=1}^{n} P_{ii}\right)^{1/2} \tag{5.10}$$

for any positive-definite symmetric matrix P; note that $0 \leq e(P) \leq 1$, with $e(P) = 1$ only when P is diagonal.

5.5 Effect of Coefficient Quantization on Minimum-Roundoff-Noise Realizations

We now consider the effect of coefficient quantization on minimum-noise realizations, since the coefficients of these realizations are in general real numbers. We consider coefficient quantization of the scaled minimum-noise realization, since quantization before scaling is overly restrictive.

We expect that a near-minimum roundoff-noise property can be obtained after coefficient quantization in A, B, and C, since quantization is an approximation process. The new solutions K and W, i.e., the solutions in (5.2) and (5.3) after coefficient quantization, will approximate the original diagonal K and W respectively, within an error bound dependent on the coefficient quantization error and the sensitivity of K and W. This follows since the entries of K and W are continuous functions of the entries of A, B, and C, by (5.2) and (5.3). The new K and W will exhibit strong diagonal dominance if they are close approximations to the original K and W.

Near-minimum roundoff noise can be obtained by quantizing coefficients so that w_{ii}, $i = 1, 2, \ldots, n$, does not change significantly. This follows since the output roundoff noise in the scaled

realization is $\sum_{i=1}^{n} w_{ii}$. We ultimately deal with signal-to-noise ratios, so a change of +3 db, i.e., + 30%, may be tolerable in w_{ii}, $i = 1, 2, \ldots, n$. By a sensitivity argument, it also follows that the change in w_{ii} is related inversely to the degree of diagonal dominance in the new solution W. This is useful because our requirements on W for stability, in Chapter 6, will be diagonal-dominance requirements.

Dynamic-range constraints will be maintained very near the original constraints if the change in k_{ii}, $i = 1, 2, \ldots, n$, is not significant. This follows since k_{ii} gives the variance at x_i due to the input. The effect of changes in these constraints is simply that the overflow probabilities will change. We also mention that approximating the optimal word lengths m_i, $i = 1, 2, \ldots, n$, by integers <u>after</u> scaling has the same effect; i.e., it changes the overflow probabilities but does not increase the roundoff noise.

Clearly, the crucial issue in the above discussion is the sensitivity of K and W to the coefficients of A, B, and C. In Chapter 7, it is shown that the sensitivity of K and W is directly related to the transfer-function sensitivity of the realization {A, B, C}. Since we are seeking realizations {A, B, C} with low transfer-function sensitivity, this is a fortunate circumstance.

5.6 Correspondence Between Lyapunov Equations of Wave Digital Realizations and Minimum-Roundoff-Noise Realizations

Here we show that the coefficient matrices A, B, and C of state equations for a wave digital filter give a diagonal K and W in

a pair of Lyapunov matrix equations which are essentially the same as (5.2) and (5.3). In this case, the form of (5.2) and (5.3) must be generalized slightly to include positive diagonal weighting matrices within the terms BB^T and C^TC. Also, the matrices B and C now consist of two column vectors and two row vectors, corresponding to two inputs and two outputs, respectively. The corresponding interpretation of the wave digital realization as a two-input-two-output minimum= roundoff-noise realization is derived in Section 5.7. The stability results of Chapter 6 will also be applicable to the wave-digital realization, since it satisfies a generalized version of (5.3).

The Lyapunov equations which are actually satisfied (with diagonal solutions) by the wave-digital realization are consequences of the properties of pseudolosslessness and reciprocity in S. These properties are given by

$$S^TGS = G \quad . \tag{5.11}$$

and

$$GS = (GS)^T \tag{5.12}$$

respectively; they are due to the network origin of S. Here G is the positive diagonal matrix of port reference conductances. It follows, by substituting (5.12) in (5.11), that $S = S^{-1}$. Matrix inversion of the entire equation (5.11) and substitution of S for S^{-1} gives

$$SG^{-1}S^T = G^{-1} \quad . \tag{5.13}$$

We also define the partition

$$G = \left[\begin{array}{c|c} G_{11} & 0 \\ \hline 0 & G_{22} \end{array}\right] \tag{5.14}$$

conformable to the partition of S in (4.29). Further information concerning S can be found in [G15].

The desired Lyapunov equations follow simply by partitioning (5.11) and (5.13) to conform to (4.29) and (5.14). This gives four separate matrix equations in each case; the equations corresponding to G_{11} and G_{11}^{-1} are

$$S_{11}^T G_{11} S_{11} + S_{21}^T G_{22} S_{21} = G_{11} , \tag{5.15}$$

and

$$S_{11} G_{11}^{-1} S_{11}^T + S_{12} G_{22}^{-1} S_{12}^T = G_{11}^{-1} \tag{5.16}$$

from (5.11) and (5.13), respectively. We may insert the matrix P, defined in (4.30), into (5.15) and (5.16) to obtain

$$(PS_{11})^T G_{11} PS_{11} + S_{21}^T G_{22} S_{21} = G_{11} \tag{5.17}$$

and

$$PS_{11} G_{11}^{-1} (PS_{11})^T + PS_{12} G_{22}^{-1} (PS_{12})^T = G_{11}^{-1} , \tag{5.18}$$

respectively, since $P = P^T = P^{-1}$ and P commutes with G_{11} and G_{11}^{-1}. In view of (4.31), it is clear that (5.17) and (5.18) have the form of the Lyapunov equations (5.3) and (5.2), respectively, if we take

$$W = G_{11} \tag{5.19}$$

and

$$K = G_{11}^{-1} \tag{5.20}$$

and if we admit the generalizations

$$C^T C \rightarrow C^T G_{22} C \tag{5.21}$$

and

$$BB^T \rightarrow B G_{22}^{-1} B^T \ . \tag{5.22}$$

Note that the generalizations (5.21) and (5.22) meet the requirements of the Lyapunov theory in Section 3.5, since $C^T G_{22} C$ and $B G_{22}^{-1} B^T$ are still positive semi-definite.

5.7 Interpretation of Wave-Digital Realization as Two-Input-Two-Output Minimum-Roundoff-Noise Realization

We now interpret the roundoff-noise minimization process in [C10] for the generalization of (5.2) and (5.3) by (5.21) and (5.22) in the two-input-two-output case; i.e., we consider the Lyapunov equations (5.18) and (5.17) which derive from the state equations of the wave digital filter. As we will show: the roundoff-noise minimization in [C10] now applies to the weighted sum of roundoff-noise variances due to both outputs; the dynamic range constraint at each state variable now applies to a weighted sum of the variances due to each input. This is the result of the formal similarity between this two-input-two-output case and the single-input-single-output case treated in [C10]. In the above sense, the (scaled) two-input-two-output wave digital filter is optimal. The results of Section 5.5, concerning deviation from this optimality due to coefficient quantization, can also be applied to the wave-digital realization.

The exposition here will parallel that in Section 5.4. In place of (5.2) and (5.3), we have

$$K = AKA^T + BG_{22}^{-1}B^T \tag{5.23}$$

and

$$W = A^TWA + C^TG_{22}C \ , \tag{5.24}$$

respectively, where

$$C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \tag{5.25}$$

and

$$B = [B_1 \ B_2] \tag{5.26}$$

consist of the row vectors C_1, C_2 and the column vectors B_1, B_2 respectively; also we define

$$G_{22} = \begin{bmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{bmatrix}. \tag{5.27}$$

It can be shown that

$$W = \sum_{k=0}^{\infty} (CA^k)^TG_{22}(CA^k), \tag{5.28}$$

which gives

$$W = \sum_{k=0}^{\infty} (A^k)^TC^TG_{22}C(A^k) = \sum_{k=0}^{\infty} (A^k)^T[\mu_1C_1^TC_1+\mu_2C_2^TC_2] \ A^k \tag{5.29}$$

by substituting (5.25) and (5.27). Since the necessary convergences can be shown, it follows by (5.29) that

$$W = \mu_1 W_1 + \mu_2 W_2 \tag{5.30}$$

where

$$W_1 \triangleq \sum_{k=0}^{\infty} (C_1 A^k)^T (C_1 A^k) \tag{5.31}$$

and

$$W_2 \triangleq \sum_{k=0}^{\infty} (C_2 A^k)^T (C_2 A^k). \tag{5.32}$$

Similarly,

$$K \triangleq \sum_{k=0}^{\infty} (A^k B) G_{22}^{-1} (A^k B)^T \quad , \tag{5.33}$$

which gives

$$K = \frac{1}{\mu_1} K_1 + \frac{1}{\mu_2} K_2 \tag{5.34}$$

where

$$K_1 \triangleq \sum_{k=0}^{\infty} (A^k B_1) (A^k B_1)^T \tag{5.35}$$

and

$$K_2 \triangleq \sum_{k=0}^{\infty} (A^k B_2) (A^k B_2)^T \quad . \tag{5.36}$$

We interpret the elements of W_1, W_2, K_1, and K_2 as the inner products of certain response sequences, analogous to those in Section 5.4. We define the response sequences:

a) $f_i^{(1)}(k)$ ($f_i^{(2)}(k)$), $k = 0, 1, 2, \ldots$, is the response $x_i(k)$ due to a unit pulse at input 1 (2), i.e., at the input corresponding to $B_1(B_2)$;

b) $g_i^{(1)}(k)$ ($g_i^{(2)}(k)$), $k = 0, 1, 2, \ldots$, is the response at output 1 (2), i.e., at the output corresponding to C_1 (C_2), due to a unit pulse at x_i.

These definitions correspond to those in Section 5.4; we have added the superscripts to distinguish $f_i(k)$ due to input 1 or 2 and to distinguish $g_i(k)$ at output 1 or 2. Note that W_1 and W_2, by (5.31) and (5.32), and K_1 and K_2, by (5.35) and (5.36), have the same form as K and W in (5.4) and (5.5). Thus, W_1, W_2, K_1, and K_2 are also the solutions of appropriate Lyapunov equations. We have the following relations, analogous to Section 5.4:

$$w_{ij}^{(1)} = g_i^{(1)} \cdot g_j^{(1)} ; \tag{5.37}$$

$$w_{ij}^{(2)} = g_i^{(2)} \cdot g_j^{(2)} ; \tag{5.38}$$

$$k_{ij}^{(1)} = f_i^{(1)} \cdot f_j^{(1)} ; \tag{5.39}$$

$$k_{ij}^{(2)} = f_i^{(2)} \cdot f_j^{(2)} , \tag{5.40}$$

Here $w_{ij}^{(1)}$, $w_{ij}^{(2)}$, $k_{ij}^{(1)}$, and $k_{ij}^{(2)}$, $i, j = 1, 2, \ldots, n$, denote the elements of W_1, W_2, K_1, and K_2, respectively. By (5.30) with (5.37) and (5.38), and by (5.34) with (5.39) and (5.40), we obtain

$$w_{ij} = \mu_1 \, g_i^{(1)} \cdot g_j^{(1)} + \mu_2 \, g_i^{(2)} \cdot g_j^{(2)} \tag{5.41}$$

and

$$k_{ij} = \frac{1}{\mu_1} f_i^{(1)} \cdot f_j^{(1)} + \frac{1}{\mu_2} f_i^{(2)} \cdot f_j^{(2)} , \tag{5.42}$$

respectively. The diagonal elements of K and W are, by (5.41) and (5.42),

$$\begin{aligned} w_{ii} &= \mu_1 \, g_i^{(1)} \cdot g_i^{(1)} + \mu_2 \, g_i^{(2)} \cdot g_i^{(2)} \\ &= \mu_1 \left|\left| g_i^{(1)} \right|\right|^2 + \mu_2 \left|\left| g_i^{(2)} \right|\right|^2 \end{aligned} \tag{5.43}$$

and

$$\begin{aligned} k_{ii} &= \frac{1}{\mu_1} f_i^{(1)} \cdot f_i^{(1)} + \frac{1}{\mu_2} f_i^{(2)} \cdot f_i^{(2)} \\ &= \frac{1}{\mu_1} \left|\left| f_i^{(1)} \right|\right|^2 + \frac{1}{\mu_2} \left|\left| f_i^{(2)} \right|\right|^2 \end{aligned} \tag{5.44}$$

Note: $\left|\left| g_i^{(1)} \right|\right|^2$ ($\left|\left| g_i^{(2)} \right|\right|^2$) is the variance at output 1 (2) due to an uncorrelated unit-variance disturbance at x_i; $\left|\left| f_i^{(1)} \right|\right|^2$ ($\left|\left| f_i^{(2)} \right|\right|^2$) is the variance at x_i due to an uncorrelated unit-variance signal at input 1 (2).

The interpretation of the roundoff-noise minimization process in [C10] follows from (5.43) and (5.44) in the two-input-two-output case. The original dynamic-range constraint in Section 5.4 consists of requiring $k_{ii}/2^{m_i}$, $i = 1, 2, \ldots, n$, to be a particular constant, since 2^{m_i} is the unscaled range of numerical representation for x_i. In Section 5.4, k_{ii} gave the variance at x_i due to a single input, whereas k_{ii} now gives a weighted sum of variances at x_i due to

both inputs, as seen from (5.44), where the inputs are unit variance and uncorrelated. Thus, the dynamic-range constraint carries over in a reasonable way to the two-input case. For example, if the uncorrelated inputs are weighted by the factors $1/\mu_1$ and $1/\mu_2$, then k_{ii} in (5.44) accurately expresses the variance at x_i due to unit variance at the inputs. The roundoff-noise minimization in Section 5.4 consists of minimizing $\sum_{i=1}^{n} w_{ii}$. There $\sum_{i=1}^{n} w_{ii}$ gave the variance at the single output, whereas $\sum_{i=1}^{n} w_{ii}$ now gives a weighted sum of variance at both outputs, due to unit-variance and uncorrelated disturbances at x_i, $i = 1, 2, \ldots, n$, as seen from (5.43). Thus, minimization of $\sum_{i=1}^{n} w_{ii}$ gives a minimum weighted sum of roundoff-noise variances due to both outputs in the two-output case.

It is not clear if a subsystem in the wave digital filter can satisfy the minimum roundoff noise condition simultaneously with the overall two-input-two-output system. In particular, we mean a subsystem consisting of a single input and single output corresponding to a transfer function, such as input 1 and output 2. This would require a diagonal K_1 and W_2, which would also give a diagonal K_2 and W_1 by (5.34) and (5.30), since K and W are already diagonal. This possibility is under investigation.

6. STABILITY

6.1 Introduction

The stability problem has two distinct aspects which are relevant to a (state-space) digital filter. First, the prototype linear discrete-time system must be stable; i.e., the system eigenvalues must lie within the unit circle. This is closely linked to the approximation problem in the frequency domain. It is also necessary to ensure that stability of the linear system is retained after coefficient quantization. At this stage we do not distinguish between stability under zero-input conditions and stability under non-zero-input conditions, since these stabilities are equivalent in a linear system. In this Chapter, we will take a stable coefficient-quantized linear system as the starting point.

The second aspect of the stability problem is related to the signal quantization which is inherent in a digital filter. Although the digital filter approximates the prototype linear discrete-time system, it is a non-linear system due to signal quantization. Hence, the digital filter may exhibit limit-cycle oscillations. It is customary to consider the zero-input and non-zero-input cases separately with regard to stability against limit cycles, since these cases are not necessarily equivalent in a non-linear system. The conventional term for stability against limit cycles under non-zero-input conditions is "stability of the forced response".

Stability against limit cycles in the zero-input case consists of two categories: (1) stability against underflow limit cycles; (2) stability against overflow limit cycles. We will refer to these categories as zero-input underflow stability and zero-input overflow stability, respectively; they are treated separately in Sections 6.5 through 6.7.

Stability of the forced response of a digital filter is relevant only with regard to possible overflow limit cycles. This follows because normal operation of a digital filter under non-zero= input conditions generates a continuous sequence of underflow errors. Consequently, it is difficult to distinguish between underflow limit cycles and underflow roundoff noise in the forced response, especially if the input signal is non-repetitive in nature. (A possible exception is the constant-input case, but this qualifies as a "repetitive" input.) Therefore, it is customary to restrict the meaning of the term "stability of the forced response" to stability against overflow limit cycles. In Section 6.8, we give a detailed interpretation of stability of the forced response.

In this Chapter, the Lyapunov approach to limit-cycle suppression in digital filters is extended to include the case where W, the solution of the Lyapunov matrix equation (5.3), is diagonally dominant. In particular, separate necessary and sufficient conditions of diagonal dominance in W are given for $\hat{x}^T W \hat{x}$ to be a Lyapunov function of the state-space digital filter, under conditions of magnitude truncation at the state variables. These diagonal= dominance conditions are derived for the following separate cases:

(1) zero-input underflow stability; (2) zero-input overflow stability; and (3) stability of the forced response.

The same underlying formulation is used to deal with all three of the above-mentioned cases; it is given in Sections 6.2 through 6.4. The specifics pertaining to zero-input overflow stability are developed in Section 6.5. The specifics pertaining to zero-input underflow stability are developed in Sections 6.6 and 6.7. Stability of the forced response is treated in Sections 6.8 through 6.13. This case requires: a detailed analysis of the forced response (Section 6.9); the application of Lyapunov stability theory to the error in the forced response (Section 6.10); a dynamic-range constraint (Section 6.11); and an extension of the magnitude-truncation requirement to an appropriate time-varying non-linearity (Section 6.12). The actual diagonal-dominance conditions for this case are given in Section 6.13. Section 6.14 gives an overview of the entire approach in terms of a vector norm. In Section 6.15, some consideration is given to the asymptotic behaviour of underflow roundoff noise in the forced response.

Some further comments regarding stability of the forced response are appropriate here. The approach taken in Sections 6.9 through 6.11 abstracts some basic principles given by Claasen et al. [D10]. In their analysis [D10], the digital-filter model includes overflow non-linearities at the delays, and possible overflow non= linearities internal to the structure as well, but it does not include underflow errors. Here we will assume overflow non-linearities only at the delays (i.e., only at the state variables), since

we intend to implement the state equations by the structure described in Chapter 8. (This structure computes exact next-state variables, i.e., excess digits are eliminated only at the delays before feedback.) We will later specialize to the saturation overflow characteristic for reasons given in Sections 5.3, 6.4, 6.5, and 6.13. However, the analysis given here will include underflow errors at the delays.

The minimum-roundoff-noise realization provides the basic ingredient for zero-input underflow and overflow stability and stability of the forced response by the Lyapunov approach, before coefficient quantization. That is, the solution W in (5.3) is diagonal. Since $\bar{x}^T W \bar{x}$ is a Lyapunov function in the linear realization, the appropriate magnitude truncations in the corresponding digital realization will guarantee the afore-mentioned stabilities, as explained in this Chapter. However, this aspect of the minimum-roundoff-noise realization was not mentioned by Mullis and Roberts [C10]. Perhaps this omission was due to the practical difficulties associated with coefficient quantization; these difficulties are surmounted by the diagonal-dominance conditions developed in this Chapter.

6.2 Lyapunov Stability Theory Extended to Digital Filters

A nonlinear digital filter is modelled by (4.23). The quadratic form $\hat{x}^T(k)\, W\hat{x}(k)$ is by definition a Lyapunov function for (4.23) if and only if W is positive definite and

$$\hat{x}^T(k)\, W\, \hat{x}\,(k) - \hat{x}^T(k+1)\, W\, \hat{x}\,(k+1) \geq 0. \tag{6.1}$$

Here we use the same W as in (3.11) or (5.3), i.e., the W which derives from the system (4.23) if the non-linearity $\bar{N}(x(k))$ is replaced by a direct feedthrough. Hence, W is positive definite. We also have, for the linear part of (4.23) under zero-input,

$$\hat{x}^T(k)\, W\, \hat{x}\,(k) - x^T(k+1)\, W\, x\,(k+1) \geq 0, \qquad (6.2)$$

since (3.11) implies (3.7). It thus remains to establish

$$x^T(k+1)\, W\, x\,(k+1) - \hat{x}^T(k+1)\, W\, \hat{x}\,(k+1) \geq 0, \quad (6.3)$$

which corresponds to the non-linear part of (4.23). The combination of (6.2) and (6.3) will give (6.1) by simple addition.

We will establish (6.3) by deriving necessary and sufficient conditions of diagonal dominance in the matrix W. We assume magnitude truncation in the mapping x to $\hat{x}$, since we will show that it is a necessary condition for (6.3). It will be possible to obtain the necessary and sufficient diagonal-dominance conditions on W due to the finite-precision signal representation. We may regard these diagonal= dominance conditions as a generalization of a well-known method. This well-known method obtains (6.3) by requiring a <u>diagonal</u> W and magnitude truncation in the mapping x to $\hat{x}$ [G11, D11].

The suppression of zero-input limit cycles will follow from (6.3). However, we require asymptotic stability to guarantee the suppression of limit cycles. This means that the <u>strict</u> inequality in (6.1) must hold for <u>at least one</u> state in any potential limit cycle. In that case we can exclude the existence of limit cycles, since the Lyapunov function $\hat{x}^T\, W\, \hat{x}$ must ultimately decrease to zero in

any potential limit cycle. That is, we ultimately obtain $\hat{x} = 0$, since it is the only state that gives $\hat{x}^T W \hat{x} = 0$.

There are two distinct possibilities to obtain strict inequality in (6.1) for at least one state in any potential limit cycle. Clearly, these possibilities are to obtain a strict inequality in either (6.2) or (6.3). The first possibility, (6.2), is linked to the linear part of the digital filter. The second possibility, (6.3), is linked to the occurrence of a strict magnitude truncation in the mapping x to $\hat{x}$. We will discuss these possibilities in some detail, since they have been a source of mild confusion in the literature.

We consider the possibility of a strict inequality in (6.2) by temporarily replacing the non-linearity $\bar{N}(x(k))$ in (4.23) by a direct feedthrough. In this linear system, a strict inequality will occur in (6.2) whenever the output y(k) is non-zero, as seen from (3.12). This implies asymptotic stability at the output, i.e., no oscillation which contains observable states can exist. Further, if the linear system is completely observable, then no oscillation can exist at the state variables. Unfortunately, the property of observability in the linear system does not necessarily carry over to the digital filter which results by restoring the non-linearity $\bar{N}(x(k))$ in (4.23). Hence, from the linear system, i.e., from (3.12), we can only infer asymptotic stability at the output of the digital filter, i.e., suppression of output-observable limit cycles. This situation occurs because we are using Lyapunov equations which have the form of (3.11) where the right hand side is positive semi-definite.

Strict inequality in (6.3) can always be obtained for at least one state in any potential limit cycle, as follows. First, we assume (non-strict) magnitude truncation is imposed on the mapping x to $\hat{x}$. Next, we note that $x(k+1) \neq \hat{x}(k+1)$ for at least one k in any potential limit cycle. Otherwise, there are no overflow or underflow errors to excite a limit cycle; i.e., the system operates the same as its linear part. By the nature of the mapping x to $\hat{x}$, composed of the overflow mapping x to $\tilde{x}$ and the underflow mapping $\tilde{x}$ to $\hat{x}$, strict magnitude truncation takes place whenever $x(k+1) \neq \hat{x}(k+1)$; this is based on the assumptions in Section 5.2 . Strict magnitude truncation in the mapping x to $\hat{x}$ gives strict inequality in (6.3). This is already clear if W is diagonal. It will also hold under the diagonal-dominance conditions which we derive for W, if in the dominance condition itself we replace the nonstrict inequality by strict inequality. The above method of obtaining strict inequality in (6.3) when W is diagonal, in order to guarantee a strict inequality in (6.1), has been used by Fettweis and Meerkötter [G11, D11].

We mention in passing that Mills et al. [D15] have taken an approach which gives a strict inequality in (6.2) for any potential limit cycle. They synthesize a realization where a positive-definite matrix replaces $C^T C$ in the right-hand side of the Lyapunov equation (3.11) and W is diagonal. This is advantageous for the second-order systems which they consider, but sacrifices some generality in (3.11) for higher-order systems. Perhaps this method ensures a greater decay rate under overflow conditions. On the

other hand, strict inequality in (6.3) is always available, as discussed in the previous paragraph. We also note that the extension from a diagonal W to a diagonally dominant W, which we give in this Chapter, may be applicable to the approach in [D15]. Finally, we note a minor error in [D15], which is more in the nature of a conjecture. In the case of wave digital filters it is inferred that one obtains strict inequality in (6.1) due to a positive-definite matrix in the right-hand side of the Lyapunov equation (3.11); this positive-definite matrix would give strict inequality in (6.2). However, in wave digital filters, as shown in Section 5.6, the right-hand side of (3.11) is positive semi-definite. The strict inequality in (6.1) is actually obtained by a strict inequality in (6.3), as shown in [G11] and discussed in the previous paragraph.

We will find diagonal-dominance conditions on W for (6.3) by giving individual consideration to the overflow mapping x to $\tilde{x}$ and the underflow mapping $\tilde{x}$ to $\hat{x}$. That is, we will break (6.3) into two separate relations, since we have chosen to decompose the mapping x to $\hat{x}$ into x to $\tilde{x}$ and $\tilde{x}$ to $\hat{x}$. These separate relations are

$$x^T (k+1)\, W\, x (k+1) - \tilde{x}^T (k+1)\, W\, \tilde{x} (k+1) \geq 0, \quad (6.4)$$

which corresponds to the overflow mapping x to $\tilde{x}$, and

$$\tilde{x}^T (k+1)\, W\, \tilde{x} (k+1) - \hat{x}^T (k+1)\, W\, \hat{x} (k+1) \geq 0, \quad (6.5)$$

which corresponds to the underflow mapping $\tilde{x}$ to $\hat{x}$. Clearly, (6.4) and (6.5) are sufficient conditions for (6.3), as seen by simple addition. We will find diagonal-dominance conditions on W

for (6.4) and (6.5) individually.

The underflow relation (6.5) is a special case of (6.3) which results when $x_i \in \tilde{X}_i$, $i = 1, 2, \ldots, n$. Thus, (6.5) is also a necessary condition for (6.3).

The overflow relation (6.4) is a limiting case of (6.3), which results as the signal granularity becomes increasingly fine. It is useful if we choose to neglect signal granularity. It is also applicable if we are dealing with a sampled-data system where the only non-linearity is due to magnitude limits.

As it turns out, the diagonal-dominance requirement on W is much greater for (6.5) than for (6.4). If we attain (6.5) by the greater diagonal-dominance condition, then (6.4) will be included as well. This gives the suppression of all zero-input limit cycles, since (6.3) follows. On the other hand, we may choose (6.4) alone. Clearly, (6.4) would be a sufficient condition for suppression of overflow limit cycles if the signal granularity were infinitely fine, e.g., if $m_i \to \infty$, $i = 1, 2, \ldots, n$. (Infinitely-fine signal granularity is tacitly assumed, e.g., in [D15].) In that case, (6.5) would be identically zero, so that (6.3) would follow immediately. This implies that the response to an overflow under zero-input conditions is a transient which decays to zero in finite time. However, the introduction of non-zero underflow errors, due to finite word lengths m_i, is akin to introducing an "input". Under this condition, the only immediate observation is that overflow errors cannot increase the energy function in a potential limit cycle, in view of (6.4). It remains to show that the response to an overflow in the presence of

underflow errors will still be a transient which decays to zero in finite time. That is, potential limit cycles would be the response to underflow errors alone. This is actually shown in the analysis in Sections 6.8 through 6.13, related to stability of the forced response. Stability of the forced response is obtained by applying the same Lyapunov function $x^T W x$ to the error due to signal overflows. This more general analysis is postponed to Sections 6.8 through 6.13, in order to concentrate on the basic principles in Sections 6.2 through 6.7.

6.3 Magnitude-Truncation Requirement

Here we show that (non-strict) magnitude truncation is a necessary condition on the mapping x to $\hat{x}$ for (6.3) to hold in general. It then follows that (non-strict) magnitude truncation is necessary in the mapping $\tilde{x}$ to $\hat{x}$ for (6.5) to hold, since this is a special case of (6.3). (Non-strict) magnitude truncation is also necessary in the mapping x to $\tilde{x}$ for (6.4) to hold. This last statement follows by exactly the same technique as in the proof below; thus we omit its separate proof.

The proof for x to $\hat{x}$ is as follows. We take the case

$$x_i \neq 0,$$

$$x_j = 0, \; j \neq i, \; j = 1, 2, \ldots, n, \tag{6.6}$$

for some i. In this case, we have, from (6.3),

$$x^T W x - \hat{x}^T W \hat{x} = w_{ii}(x_i^2 - \hat{x}_i^2) \geq 0 \; , \tag{6.7}$$

where $w_{ii} > 0$ since W is positive definite. Note that (6.7) holds whether W is diagonal or not. Now, for the right half of (6.7) to hold, we must have $|x_i| \geq |\hat{x}_i|$. Taking each i in turn, for i = 1, 2, ..., n, we have

$$|x_i| \geq |\hat{x}_i|, \quad i = 1, 2, \ldots, n. \tag{6.8}$$

The corresponding results for x to $\tilde{x}$ and $\tilde{x}$ to $\hat{x}$ are, respectively,

$$|x_i| \geq |\tilde{x}_i|, \quad i = 1, 2, \ldots, n, \tag{6.9}$$

and

$$|\tilde{x}_i| \geq |\hat{x}_i|, \quad i = 1, 2, \ldots, n. \tag{6.10}$$

6.4 Diagonal-Dominance Conditions on W: Basic Formulation

Here we give the basic formulation by which we will obtain diagonal-dominance conditions on W for the overflow relation (6.4) and the underflow relation (6.5). The only underlying assumptions are: (1) the mappings x to $\tilde{x}$ and $\tilde{x}$ to $\hat{x}$, which compose the non-linear function $\bar{N}$ in (4.20), satisfy the magnitude-truncation requirements (6.9) and (6.10); (2) magnitude truncation is implemented in the underflow mapping $\tilde{x}$ to $\hat{x}$ by rounding down in magnitude from $\tilde{x}_i$ to the nearest m_i-bit integer $\hat{x}_i$, i = 1, 2, ..., n. This second assumption is in agreement with the discussion of Section 5.2; we may summarize it by setting each of the m_i non-zero bits in (4.7) equal to the bit occupying the corresponding place in (4.6).

We do not, at this moment, impose requirements other than (4.15) and (4.16) on the overflow mapping x to $\tilde{x}$. Clearly, by the definitions of x and $\tilde{x}$, any overflow characteristic satisfying (4.15) and (4.16) will meet the magnitude-truncation requirement (6.9). However, in deriving a necessary condition for the overflow relation (6.4), we will find that the saturation overflow characteristic gives the least-severe diagonal-dominance requirement on W. Hence, we will base our final result on the saturation characteristic.

The preliminary step in the derivation consists of showing that

$$x^T W x - \tilde{x}^T W \tilde{x} \equiv (x + \tilde{x})^T W (x - \tilde{x}) \tag{6.11}$$

and that

$$\tilde{x}^T W \tilde{x} - \hat{x}^T W \hat{x} \equiv (\tilde{x} + \hat{x})^T W (\tilde{x} - \hat{x}) \ . \tag{6.12}$$

Here we mean that (6.11) and (6.12) are algebraic identities independent of the mappings x to $\tilde{x}$ and $\tilde{x}$ to $\hat{x}$. Since (6.11) and (6.12) are <u>formally</u> the same, we need only prove one of them. We demonstrate (6.11) by expanding its right-hand side:

$$(x + \tilde{x})^T W (x - \tilde{x}) \equiv x^T W x - x^T W \tilde{x} + \tilde{x}^T W x - \tilde{x}^T W \tilde{x} \ . \tag{6.13}$$

All the terms in (6.13) are "one-by-one" matrices, in particular $\tilde{x}^T W x$, so

$$\tilde{x}^T W x = (\tilde{x}^T W x)^T = x^T W^T \tilde{x}. \tag{6.14}$$

Also

$$W^T = W, \tag{6.15}$$

i.e., W is symmetric because it is the solution of the "symmetric" Lyapunov matrix equation (3.9). Thus, substituting (6.15) in (6.14), we have

$$\tilde{x}^T W x = x^T W \tilde{x}, \tag{6.16}$$

which means that the middle two terms in (6.13) cancel. This gives the desired result (6.11).

We now rewrite the overflow relation (6.4) as

$$(x + \tilde{x})^T W (x - \tilde{x}) \geq 0 \tag{6.17}$$

and the underflow relation (6.5) as

$$(\tilde{x} + \hat{x})^T W (\tilde{x} - \hat{x}) \geq 0 , \tag{6.18}$$

by substituting (6.11) and (6.12) in (6.4) and (6.5), respectively. The advantage of (6.17) and (6.18) is that the desired condition is expressed in terms of the error vector, $x - \tilde{x}$ or $\tilde{x} - \hat{x}$, and the vector $x + \tilde{x}$ or $\tilde{x} + \hat{x}$ which varies roughly with the signal. This simplifies the problem of bounding terms to obtain diagonal-dominance conditions on W.

We expand (6.17) and (6.18) around the columns of W; the manipulation is the same for (6.17) and (6.18), so here we treat only (6.17) in detail. We begin by pre-multiplying the row vector $(x+\tilde{x})^T$ into W, i.e., into the columns of W. The product $\bar{p}^T$ is a row vector

$$\bar{p}^T = (x + \tilde{x})^T W; \tag{6.19}$$

and the j^{th} component of $\bar{p}$ has the form

$$\bar{p}_j = (x + \tilde{x})_1 w_{1j} + (x + \tilde{x})_2 w_{2j} + \ldots + (x + \tilde{x})_n w_{nj} \quad (6.20)$$

for $j = 1, 2, \ldots, n$, where $(x + \tilde{x})_i$ denotes the i^{th} component of $(x + \tilde{x})$, i.e., $(x + \tilde{x})_i = x_i + \tilde{x}_i$, $i = 1, 2, \ldots, n$. The elements of W are denoted by w_{ij}, $i, j = 1, 2, \ldots, n$, in the usual way; note that $\bar{p}_j$ contains all the elements in the j^{th} column of W, but no others. We complete the expansion of (6.17) by post-multiplying $\bar{p}^T$ by $(x - \tilde{x})$:

$$(x + \tilde{x})^T W(x - \tilde{x}) = \bar{p}^T(x - \tilde{x}) = \sum_{j=1}^{n} \bar{p}_j (x - \tilde{x})_j = \sum_{j=1}^{n} \tilde{C}_j \quad (6.21)$$

where we define

$$\tilde{C}_j \triangleq \bar{p}_j (x - \tilde{x})_j = [(x + \tilde{x})_1 w_{1j} + (x + \tilde{x})_2 w_{2j} + \ldots + (x + \tilde{x})_n w_{nj}\,](x - \tilde{x})_j. \quad (6.22)$$

We now rewrite the overflow relation (6.17) as

$$\sum_{j=1}^{n} \tilde{C}_j \geq 0 \quad (6.23)$$

by substituting (6.21) in (6.17); i.e., (6.23) is equivalent to (6.17). The corresponding expansion of (6.18) is

$$(\tilde{x} + \hat{x})^T W(\tilde{x} - \hat{x}) = \sum_{j=1}^{n} \hat{C}_j \quad (6.24)$$

where we define

$$\hat{C}_j \triangleq [(\tilde{x} + \hat{x})_1 w_{1j} + (\tilde{x} + \hat{x})_2 w_{2j} + \ldots + (\tilde{x} + \hat{x})_n w_{nj}](\tilde{x} - \hat{x})_j. \tag{6.25}$$

We rewrite the underflow relation (6.18) as

$$\sum_{j=1}^{n} \hat{C}_j \geq 0 \tag{6.26}$$

by substituting (6.24) in (6.18); i.e., (6.26) is equivalent to (6.18).

6.5 Diagonal-Dominance Conditions on W: Zero-Input Overflow Stability

We will derive a necessary and a sufficient diagonal-dominance condition on W for the overflow relation (6.4) to hold. We will do this by using (6.23) and the definition (6.22), which are together equivalent to (6.4).

A necessary condition follows by noting that the variables x_i are independent for different indices, i.e., x_i is independent of x_j, $i \neq j$, $i = 1, 2, \ldots, n$, for any given j, $j = 1, 2, \ldots, n$. Thus, regardless of x_j, we may have

$$x_i \in \tilde{X}_i, \; i \neq j, \; i = 1, 2, \ldots, n, \tag{6.27}$$

which gives

$$(x - \tilde{x})_i = 0, \; i \neq j, \; i = 1, 2, \ldots, n, \tag{6.28}$$

by the overflow mapping x_i to $\tilde{x}_i$ in (4.15) and (4.16). Note that (6.27) states that an overflow does not exist at the variable x_i, $i \neq j$, $i = 1, 2, \ldots, n$. From (6.28) and (6.22) we have

$$\tilde{C}_i = 0, \; i \neq j, \; i = 1, 2, \ldots, n, \tag{6.29}$$

which, by (6.23), gives the necessary condition

$$\tilde{C}_j \Big|_{\substack{x_i \in \tilde{X}_i \\ i \neq j,\ i = 1, 2, \ldots, n}} \geq 0,\ j = 1, 2, \ldots, n. \qquad (6.30)$$

The sufficient condition

$$\tilde{C}_j \Big|_{\substack{x_i \in X_i \\ i = 1, 2, \ldots, n}} \geq 0,\ j = 1, 2, \ldots, n, \qquad (6.31)$$

follows simply by noting that (6.31) makes (6.23) a valid statement. Here $x_i \in X_i$, $i = 1, 2, \ldots, n$, means that the variables x_i are unrestricted, i.e., overflows may exist at any variable. The usefulness of this sufficient condition, which may be overly demanding in some cases, will depend on the degree of diagonal dominance which it imposes on W.

We now obtain diagonal-dominance conditions on the columns of W from (6.30) and (6.31); these dominance conditions also apply to the rows of W, since W is symmetric. We will convert (6.31) into a form which shows the dominance conditions explicitly; this form will also apply to (6.30), since (6.30) is the special case of (6.31) where the domain of the variable x_i is restricted to the subset $\tilde{X}_i$, rather than X_i, $i \neq j$, $i = 1, 2, \ldots, n$.

There are two possibilities in (6.31):

I. An overflow error does not occur at x_j, i.e.,

$$x_j \in \tilde{X}_j, \qquad (6.32)$$

which means

$$\tilde{x}_j = x_j \tag{6.33}$$

or

$$(x - \tilde{x})_j = 0; \tag{6.34}$$

II. An overflow error does occur at x_j, i.e.,

$$x_j \in X_j, \notin \tilde{X}_j, \tag{6.35}$$

which means

$$\tilde{x}_j \neq x_j \tag{6.36}$$

or

$$(x - \tilde{x})_j \neq 0. \tag{6.37}$$

In the first possibility we have, by (6.34) and (6.22),

$$\tilde{C}_j = 0, \tag{6.38}$$

regardless of the values of $(x + \tilde{x})_i$, $i = 1, 2, \ldots, n$. Since (6.38) satisfies (6.31) with equality, we need only consider the second possibility. In the second possibility we rewrite $\tilde{C}_j$ as

$$\tilde{C}_j = (x + \tilde{x})_1 (x - \tilde{x})_j w_{1j} + (x + \tilde{x})_2 (x - \tilde{x})_j w_{2j} + \ldots + (x + \tilde{x})_n (x - \tilde{x})_j w_{nj} \geq 0, \tag{6.39}$$

from (6.22). Since x_i, $i \neq j$, is independent of x_j, the terms $(x + \tilde{x})_i (x - \tilde{x})_j w_{ij}$, $i \neq j$, $i = 1, 2, \ldots, n$, in (6.39) may be

negative by appropriate choices of sign for $x_i \in \tilde{X}_i$, $i \neq j$, $i = 1, 2, \ldots, n$. Thus we must have $(x + \tilde{x})_j (x - \tilde{x})_j w_{jj} > 0$ to guarantee (6.39), which implies

$$x_j^2 - \tilde{x}_j^2 > 0, \tag{6.40}$$

since $w_{jj} > 0$. We have already established the non-strict magnitude truncation (6.9) as a necessary condition. The strict magnitude truncation (6.40) is equivalent to (6.9) with two additional requirements: $x_j \neq \tilde{x}_j$, which is already guaranteed by (6.36); and

$$x_j \neq -\tilde{x}_j. \tag{6.41}$$

However, our previous requirements (4.15) and (4.16) on the overflow mapping guarantee strict magnitude truncation, and hence (6.41), due to our definitions of X_i and $\tilde{X}_i$; i.e., we have $|x_i| > |\tilde{x}_i|$ for any $x_i \in X_i$, $\notin \tilde{X}_i$ and any $\tilde{x}_i \in \tilde{X}_i$. We have mentioned the requirement (6.41) for the following reason: The use of two's complement arithmetic and its natural overflow characteristic would produce a slight modification in X_i and $\tilde{X}_i$, $i = 1, 2, \ldots, n$, such that (6.41) is violated when x_j exceeds the maximum positive value by one. This is not necessarily a drawback of two's complement arithmetic, since we will prefer to modify the natural overflow characteristic in any representation to obtain the saturation overflow characteristic.

We have reduced (6.31) to the only non-trivial possibility (6.37), where (6.41) must hold. In this case we may rewrite (6.31) as

$$\tilde{C}_j = (x + \tilde{x})_j (x - \tilde{x})_j \left[w_{jj} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(x + \tilde{x})_i}{(x + \tilde{x})_j} w_{ij} \right] \geq 0 \qquad (6.42)$$

by factoring out $(x + \tilde{x})_j$ in (6.22), since

$$(x + \tilde{x})_j \neq 0 \qquad (6.43)$$

by (6.41). Since $(x + \tilde{x})_j (x - \tilde{x})_j > 0$ by (6.40), we recognize that

$$w_{jj} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(x + \tilde{x})_i}{(x + \tilde{x})_j} w_{ij} \geq 0 \qquad (6.44)$$

is equivalent to (6.42), hence equivalent to (6.31). Finally, we rewrite (6.44) in the form

$$w_{jj} \geq - \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(x + \tilde{x})_i}{(x + \tilde{x})_j} w_{ij} \quad , \qquad (6.45)$$

which we interpret as a diagonal-dominance condition on the j^{th} column of W, equivalent to (6.31). It now remains to establish suitable magnitude bounds on the terms $(x + \tilde{x})_i / (x + \tilde{x})_j$ in (6.45).

The two different sets of conditions under which we bound the terms of (6.45) correspond to the necessary condition (6.30) and the sufficient condition (6.31). These sets of conditions are, respectively:

$$x_i \in \tilde{X}_i, \; i \neq j, \; i = 1, 2, \ldots, n; \qquad (6.46)$$

and

$$x_i \in X_i, \; i \neq j, \; 1 = 1, 2, \ldots, n. \qquad (6.47)$$

The additional restriction (6.35) applies to both (6.46) and (6.47). We will also assume the desirable case where the overflow mapping x to $\tilde{x}$ has odd symmetry; this is true in the sign-magnitude representation and very nearly true in the two's complement representation. We denote the right-hand side of (6.45) by the function

$$f_j(x_1, x_2, \ldots, x_n) \triangleq - \frac{1}{(x + \tilde{x})_j} \sum_{\substack{i=1 \\ i \neq j}}^{n} (x + \tilde{x})_i w_{ij}; \quad (6.48)$$

here f_j is the product of two independent functions

$$g_j \triangleq \frac{-1}{(x + \tilde{x})_j} \quad (6.49)$$

and

$$h_j \triangleq \sum_{\substack{i=1 \\ i \neq j}}^{n} (x + \tilde{x})_i \, w_{ij} \, , \quad (6.50)$$

i.e.,

$$f_j \triangleq g_j h_j . \quad (6.51)$$

(Note that the symbols f and g used in this Chapter are not related to the f and g used in Chapter 5.) The function g_j is bounded due to (6.43), since x_j and $\tilde{x}_j$ are both restricted to integers. The function h_j is bounded due to (6.46) or (6.47). The functions g_j and h_j both have odd symmetry with respect to their independent variables, i.e., with respect to x_j and the set x_i, $i \neq j$, $i = 1, 2, \ldots, n$, respectively. Therefore, the positive and negative extremes of g_j have the same magnitude; likewise, the positive and negative extremes of h_j have the same magnitude. The maximum of

f_j in (6.51) is the product of either the positive extremes of g_j and h_j or the negative extremes of g_j and h_j, depending on which gives the larger result. We will evaluate the maximum of f_j as the product of the positive extremes of g_j and h_j, since here it is clear that either of the above choices gives the maximum.

We first consider (6.46), i.e., the case where an overflow occurs only at x_j. Then (6.50) becomes

$$h_j = \sum_{\substack{i=1 \\ i \neq j}}^{n} 2\tilde{\tilde{x}}_i w_{ij}. \tag{6.52}$$

The positive extreme

$$\max h_j = 2 \sum_{\substack{i=1 \\ i \neq j}}^{n} \tilde{R}_i |w_{ij}| \tag{6.53}$$

occurs in (6.52) when

$$\tilde{\tilde{x}}_i = \mathrm{SGN}(w_{ij})\, \tilde{R}_i,\ i \neq j,\ i = 1, 2, \ldots, n. \tag{6.54}$$

This extreme is independent of the overflow characteristic. However, due to (6.35), the extremes of g_j in (6.49) are dependent on the mapping x_j to $\tilde{x}_j$; i.e., $(x + \tilde{x})_j$ is a function of x_j which changes by changing the mapping x_j to $\tilde{x}_j$. We desire the least max g_j over the choices for x_j to $\tilde{x}_j$, because, by (6.51), this will give the least max f_j and hence minimize the diagonal-dominance requirement in (6.45). Since $g_j(x_j)$ has odd symmetry, $g_j(x_j)$ and $-g_j(-x_j)$ are the same. Therefore, we will deal with $-g_j$, i.e., $1/(x + \tilde{x})_j$, to avoid confusion with the negative sign in (6.49).

Clearly, max $1/(x + \tilde{x})_j$ occurs at the positive minimum of $(x + \tilde{x})_j$. The least max $1/(x + \tilde{x})_j$ occurs for that mapping x_j to $\tilde{x}_j$ which gives the greatest positive minimum for $(x + \tilde{x})_j$. Since (6.35) holds, $x_j > \tilde{R}_j$ is necessary and sufficient to obtain $(x + \tilde{x})_j > 0$, i.e., we need not consider $x_j < -\tilde{R}_j$. The saturation overflow characteristic for x to $\tilde{x}$ gives $x_j + \tilde{R}_j$ for the function $(x + \tilde{x})_j$, which is a pointwise upper bound on any other possibility for $(x + \tilde{x})_j$. This last statement follows because

$$x_j + \tilde{R}_j \geq x_j + \tilde{x}_j \tag{6.55}$$

in any case, due to the requirement $\tilde{R}_j \geq \tilde{x}_j \geq -\tilde{R}_j$. Thus, the saturation overflow characteristic gives the greatest positive minimum for $(x + \tilde{x})_j$. For this reason, we will assume the use of the saturation characteristic; the resulting minimum of $(x + \tilde{x})_j$ is $2\tilde{R}_j + 1$, which occurs at $x_j = \tilde{R}_j + 1$. This gives

$$\max g_j = \frac{1}{2\tilde{R}_j + 1} \tag{6.56}$$

Now, combining (6.45) with (6.48), (6.51), (6.53), and (6.56), we have the diagonal-dominance requirement

$$w_{jj} \geq \frac{1}{\tilde{R}_j + \frac{1}{2}} \sum_{\substack{i=1 \\ i \neq j}}^{n} \tilde{R}_i |w_{ij}| . \tag{6.57}$$

Since (6.57) is based on (6.46), it is a <u>necessary</u> condition on W for (6.23) or (6.17), i.e., for (6.4). By (6.30), (6.57) must hold for each column of W, i.e., for j = 1, 2, ..., n. Since normally

$\tilde{R}_j >> \frac{1}{2}$, we safely replace (6.57) by

$$w_{jj} \geq \frac{1}{\tilde{R}_j} \sum_{\substack{i=1 \\ i \neq j}}^{n} \tilde{R}_i |w_{ij}|. \tag{6.58}$$

In the equal-word-length case, i.e., where $\tilde{R}_1 = \tilde{R}_2 = \ldots = \tilde{R}_n$, (6.58) becomes

$$w_{jj} \geq \sum_{\substack{i=1 \\ i \neq j}}^{n} |w_{ij}|. \tag{6.59}$$

We now consider (6.47), i.e., the case where an overflow may occur at any variable x_i, $i = 1, 2, \ldots, n$. Our method here is identical to that in the previous paragraph, except that the bound on h_j becomes larger. We again assume the saturation overflow characteristic, so (6.50) becomes

$$h_j = \sum_{\substack{i=1 \\ i \neq j}}^{n} (x_i + \tilde{R}_i) w_{ij}. \tag{6.60}$$

The maximum of (6.60) is

$$\max h_j = \sum_{\substack{i=1 \\ i \neq j}}^{n} (R_i + \tilde{R}_i)|w_{ij}| \tag{6.61}$$

which occurs when

$$x_i = SGN(w_{ij})\, R_i, \; i \neq j, \; i = 1, 2, \ldots, n. \tag{6.62}$$

Note that $\max h_j$ is now dependent on the overflow characteristic, since (6.47) permits overflows at x_i, $i \neq j$, $i = 1, 2, \ldots, n$. The

maximum of g_j is the same as that found in the previous paragraph, since the conditions on x_j and $\tilde{x}_j$ have not been changed. Thus we use (6.56) for max g_j. Combining (6.45) with (6.48), (6.51), (6.61), and (6.56), as before, we have the diagonal-dominance condition

$$w_{jj} \geq \frac{1}{2} \left(\frac{1}{\tilde{R}_j + \frac{1}{2}}\right) \sum_{\substack{i=1 \\ i \neq j}}^{n} (R_i + \tilde{R}_i) \, | w_{ij} |. \tag{6.63}$$

Since (6.63) is based on (6.47), (6.63) is a <u>sufficient condition</u> on W for (6.23) or (6.17), i.e., for (6.4). By (6.31), (6.63) must hold for each column of W, i.e., for j = 1, 2, ..., n. Since normally $R_j >> \frac{1}{2}$, we safely replace (6.63) by

$$w_{jj} \geq \frac{1}{2} \sum_{\substack{i=1 \\ i \neq j}}^{n} \left(\frac{\tilde{R}_i + R_i}{\tilde{R}_j}\right) | w_{ij} |, \tag{6.64}$$

which becomes

$$w_{jj} \geq \frac{1}{2} \sum_{\substack{i=1 \\ i \neq j}}^{n} \left(1 + \frac{R_i}{\tilde{R}_j}\right) | w_{ij} | \tag{6.65}$$

in the equal-word-length case, i.e., the case $\tilde{R}_1 = \tilde{R}_2 \ldots = \tilde{R}_n$.

We conclude this section with a few remarks concerning the results. First, we observe that the i^{th} coefficient in the sufficient condition (6.64) exceeds the i^{th} coefficient in the necessary condition (6.58) by the factor $(1 + R_i/\tilde{R}_i) / 2$, i = 1, 2, ..., n. In some realizations, $R_i/\tilde{R}_i$ may be 2 or less; this gives a value of 3/2 for the factor, which is quite tolerable. Secondly, from previous discussion, we have seen that overflow characteristics which reduce the error

between x_j and $\tilde{x}_j$ tend to give lesser maximums for g_j; hence we have chosen the saturation characteristic, which gives the minimum error between x_j and $\tilde{x}_j$. On the other hand, those overflow characteristics which allow $\tilde{x}_j$ to approach $-x_j$, such as the two's complement characteristic, would yield small values for $(x + \tilde{x})_j$, which greatly increase max g_j. Hence the coefficients in the diagonal-dominance requirement (6.58) or (6.64) would increase greatly. For this reason, we have assumed that such unfavorable characteristics would be modified to the saturation characteristic.

6.6 Diagonal-Dominance Conditions on W: Zero-Input Underflow Stability

We will derive a necessary and a sufficient diagonal-dominance condition on W for the underflow relation (6.5) to hold. We will do this by using (6.26) and the definition (6.25), which are together equivalent to (6.5). The line of reasoning will be quite similar to that for the overflow relation in Section 6.5. Some of the formal manipulations will be identical to those in Section 6.5, except for a relabelling from x_i, $\tilde{x}_i$, X_i, and $\tilde{X}_i$ to $\tilde{x}_i$, $\hat{x}_i$, $\tilde{X}_i$, and $\hat{X}_i$, respectively. Thus we will occasionally refer to Section 6.5 and assume the appropriate relabelling.

A necessary condition follows by noting that the variables $\tilde{x}_i$ are independent for different indices, i.e., $\tilde{x}_i$ is independent of $\tilde{x}_j$, $i \neq j$, $i = 1, 2, \ldots, n$, for any given j, $j = 1, 2, \ldots, n$. Thus, regardless of $\tilde{x}_j$, we may have

$$\tilde{x}_i \in \hat{X}_i, \; i \neq j, \; i = 1, 2, \ldots, n, \tag{6.66}$$

which gives

$$(\tilde{x} - \hat{x})_i = 0,\ i \neq j,\ i = 1, 2, \ldots, n, \tag{6.67}$$

due to the underflow mapping $\tilde{x}_i$ to $\hat{x}_i$; this mapping is completely defined in the first paragraph of Section 6.4. (6.66) states that an underflow does not exist at the variable $\tilde{x}_i$, $i \neq j$, $i = 1, 2, \ldots, n$. From (6.67) and (6.25) we have

$$\hat{C}_i = 0,\ i \neq j,\ i = 1, 2, \ldots, n, \tag{6.68}$$

which, by (6.26), gives the necessary condition

$$\hat{C}_j \Big|_{\substack{\tilde{x}_i \in \hat{X}_i \\ i \neq j,\ i = 1, 2, \ldots, n}} \geq 0,\ j = 1, 2, \ldots, n. \tag{6.69}$$

The sufficient condition

$$\hat{C}_j \Big|_{\substack{\tilde{x}_i \in \tilde{X}_i \\ i = 1, 2, \ldots, n}} \geq 0,\ j = 1, 2, \ldots, n, \tag{6.70}$$

follows simply by noting that (6.70) makes (6.26) a valid statement. Here $\tilde{x}_i \in \tilde{X}_i$, $i = 1, 2, \ldots, n$, means that the variables x_i are unrestricted, i.e., underflows may exist at any variable. As it turns out, this sufficient condition can be attained by just slightly more diagonal dominance in W than the necessary condition (6.69) requires.

We now obtain diagonal-dominance conditions on the columns of W from (6.69) and (6.70), analogous to Section 6.5; likewise, these dominance conditions also apply to the rows of W. We will

convert (6.70) into a form which shows the dominance conditions explicitly; this form will also apply to (6.69), since (6.69) is the special case of (6.70) where the domain of the variable $\tilde{x}_i$ is restricted to the subset $\hat{X}_i$, rather than $\tilde{X}_i$, $i \neq j$, $i = 1, 2, \ldots, n$.

There are two possibilities in (6.70):

I. An underflow error does not occur at $\tilde{x}_j$, i.e.,

$$\tilde{x}_j \in \hat{X}_j, \tag{6.71}$$

which means

$$\tilde{x}_j = \hat{x}_j \tag{6.72}$$

or

$$(\tilde{x} - \hat{x})_j = 0; \tag{6.73}$$

II. An underflow error does occur at x_j, i.e.,

$$\tilde{x}_j \in \tilde{X}_j, \notin \hat{X}_j \tag{6.74}$$

which means

$$\bar{x}_j \neq \hat{x}_j \tag{6.75}$$

or

$$(\tilde{x} - \hat{x})_j \neq 0 . \tag{6.76}$$

In the first possibility we have, by (6.73) and (6.25),

$$\hat{C}_j = 0, \tag{6.77}$$

regardless of the values of $(\tilde{x} + \hat{x})_i$, $i = 1, 2, \ldots, n$. Since

(6.77) satisfies (6.70) with equality, we need only consider the second possibility. In the second possibility, we conclude that we require the strict magnitude truncation

$$\tilde{x}_j^2 - \hat{x}_j^2 > 0, \tag{6.78}$$

by exactly the same argument which proceeds from (6.39) to (6.40) in Section 6.5. We have already established the non-strict magnitude truncation (6.10) as a necessary condition. The strict magnitude truncation (6.78) is equivalent to (6.10) with two additional requirements: $\tilde{x}_j \neq \hat{x}_j$, which is already guaranteed by (6.75); and

$$\tilde{x}_j \neq -\hat{x}_j. \tag{6.79}$$

The requirement (6.79) is guaranteed by the previously mentioned definition of the underflow characteristic $\tilde{x}$ to $\hat{x}$, as follows. First, by (6.74), $\tilde{x}_j \neq 0$. Thus (6.79) holds if $\hat{x}_j = 0$. If $\hat{x}_j \neq 0$, then (6.79) must again hold because the underflow characteristic produces the same sign for $\tilde{x}_j$ and $\hat{x}_j$. Hence, (6.78) is satisfied, as required.

We have reduced (6.70) to the only non-trivial possibility (6.76), where (6.79) must hold. In this case we rewrite (6.70) as

$$\hat{C}_j = (\tilde{x} + \hat{x})_j (\tilde{x} - \hat{x})_j \left[w_{jj} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(\tilde{x} + \hat{x})_i}{(\tilde{x} + \hat{x})_j} w_{ij} \right] \geq 0 \tag{6.80}$$

by factoring out $(\tilde{x} + \hat{x})_j$ in (6.25), which is permissible since

$$(\tilde{x} + \hat{x})_j \neq 0 \tag{6.81}$$

by (6.79). By (6.78), we obtain

$$w_{jj} \geq - \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(\tilde{x} + \hat{x})_i}{(\tilde{x} + \hat{x})_j} w_{ij} , \tag{6.82}$$

which is equivalent to (6.80), hence equivalent to (6.70). (See Section 6.5, (6.44) - (6.45) for more detail.) We interpret (6.82) as a diagonal-dominance condition on the j^{th} column of W. It remains to establish suitable magnitude bounds on the terms $(\tilde{x} + \hat{x})_i \; / \; (\tilde{x} + \hat{x})_j$ in (6.82).

The two different sets of conditions under which we bound the terms of (6.82) correspond to the necessary condition (6.69) and the sufficient condition (6.70). These sets of conditions are, respectively:

$$\tilde{x}_i \in \hat{X}_i, \; i \neq j, \; i = 1, 2, \ldots, n, \tag{6.83}$$

and

$$\tilde{x}_i \in \tilde{X}_i, \; i \neq j, \; i = 1, 2, \ldots, n. \tag{6.84}$$

The additional restriction (6.74) applies to both (6.83) and (6.84). Note that the underflow mapping $\tilde{x}$ to $\hat{x}$ has odd symmetry. Similar to Section 6.5, (6.48) - (6.51), we denote the right-hand side of (6.82) by the function

$$\hat{f}_j (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n) \triangleq - \frac{1}{(\tilde{x} + \hat{x})_j} \sum_{\substack{i=1 \\ i \neq j}}^{n} (\tilde{x} + \hat{x})_i \, w_{ij} \tag{6.85}$$

and define the two independent functions

$$\hat{g}_j \triangleq \frac{-1}{(\tilde{x} + \hat{x})_j} \tag{6.86}$$

and

$$\hat{h}_j \triangleq \sum_{\substack{i=1 \\ i \neq j}}^{n} (\tilde{x} + \hat{x})_i \, w_{ij} \; ; \tag{6.87}$$

hence

$$\hat{f}_j \triangleq \hat{g}_j \, \hat{h}_j \; . \tag{6.88}$$

By the same argument as that following (6.51) in Section 6.5, we find the maximum of $\hat{f}_j$ as

$$\max \hat{f}_j = (\max \hat{g}_j) \, (\max \hat{h}_j). \tag{6.89}$$

We first consider (6.83), i.e., the case where an underflow occurs only at x_j. Then (6.87) becomes

$$\hat{h}_j = \sum_{\substack{i=1 \\ i \neq j}}^{n} 2\hat{x}_i \, w_{ij}. \tag{6.90}$$

This gives

$$\max \hat{h}_j = 2 \sum_{\substack{i=1 \\ i \neq j}}^{n} \hat{R}_i \, |w_{ij}| \tag{6.91}$$

which occurs when

$$\tilde{x}_i = \text{SGN} \, (w_{ij}) \, \hat{R}_i, \; i \neq j, \; i = 1, 2, \ldots, n. \tag{6.92}$$

The maximum of $\hat{g}_j$ in (6.86) occurs at the negative value of $(\tilde{x} + \hat{x})_j$ which has minimum magnitude under the restriction (6.74). It is not difficult to show that this negative value is $-2^{(N-N_j)}$, which

occurs at $\tilde{x}_j = -2^{(N-N_j)}$, $\hat{x}_j = 0$. Thus,

$$\max \hat{g}_j = 2^{(N_j-N)}. \tag{6.93}$$

Combining (6.82) with (6.85), (6.89), (6.91), and (6.93), we have the diagonal-dominance requirement

$$w_{jj} \geq 2^{(N_j - N + 1)} \sum_{\substack{i=1 \\ i\neq j}}^{n} \hat{R}_i |w_{ij}|$$

which is equivalent to

$$w_{jj} \geq 2^{(N_j + 1)} \sum_{\substack{i=1 \\ i\neq j}}^{n} (2^{m_i} - 1) |w_{ij}| \tag{6.94}$$

by using (4.9) and cancelling the scale factor 2^N. Since (6.94) is based on (6.83), it is a <u>necessary condition</u> on W for (6.26) or (6.18), i.e., for (6.5). By (6.69), (6.94) must hold for every column of W, i.e., for j = 1, 2, ..., n. In the equal-word-length case, i.e., $m = m_1 = m_2 = \ldots = m_n$, (6.94) specializes to

$$w_{jj} \geq 2^{(N_j + 1)} (2^m - 1) \sum_{\substack{i=1 \\ i\neq j}}^{n} |w_{ij}|. \tag{6.95}$$

We now consider (6.84), i.e., the case where an overflow may occur at any variable x_i, i = 1, 2, ..., n. The method is identical to that in the previous paragraph, except for a small increase in max $\hat{n}_j$. The magnitude bound on $(\tilde{x} + \hat{x})_i$ under (6.84) is $\tilde{R}_i + \hat{R}_i$. This gives

$$\max \hat{h}_j = \sum_{\substack{i=1 \\ i \neq j}}^{n} (\tilde{R}_i + \hat{R}_i) | w_{ij} | \tag{6.96}$$

in (6.87) which occurs when

$$\tilde{x}_i = \text{SGN}\,(w_{ij})\, \tilde{R}_i, \; i \neq j, \; i = 1, 2, \ldots, n. \tag{6.97}$$

The maximum of $\hat{g}_j$ is still given by (6.93), since the conditions on $\tilde{x}_j$ have not changed. In the same way as before, we have the diagonal-dominance requirement

$$w_{jj} \geq 2^{(N_j - N)} \sum_{\substack{i=1 \\ i \neq j}}^{n} (\tilde{R}_i + \hat{R}_i) | w_{ij} | ,$$

which is equivalent to

$$w_{jj} \geq 2^{(N_j + 1)} \sum_{\substack{i=1 \\ i \neq j}}^{n} (2^{m_i} - \tfrac{1}{2} - 2^{-N_i - 1}) | w_{ij} | \tag{6.98}$$

by using (4.8) and (4.9) and cancelling the scale factor 2^N. Since (6.98) is based on (6.84), it is a <u>sufficient condition</u> on W for (6.26) or (6.18), i.e., for (6.5). By (6.70), (6.98) must hold for every column of W, i.e., for j = 1, 2, ..., n. In the equal-word-length case, (6.98) specializes to

$$w_{jj} \geq 2^{(N_j + 1)} (2^m - \tfrac{1}{2}) \sum_{\substack{i=1 \\ i \neq j}}^{n} | w_{ij} | , \tag{6.99}$$

where we have discarded the relatively small term $-2^{-N_i - 1}$ in each coefficient.

The coefficients for the sufficient condition (6.98) and the necessary condition (6.94) are very nearly the same on a fractional basis. The ratio of the i^{th} coefficients is $1 + (1 - 2^{-N_i}) / 2 (2^{m_i}-1)$; this is lower bounded by one and upper bounded by $1 + 1/2 (2^{m_i} - 1)$, which is very close to one. For example, if $m_i = 8$ (bits), $i = 1, 2, \ldots, n$, then the ratio of coefficients exceeds one by less than $\frac{1}{510}$. Since the sufficient condition exceeds the necessary condition by so little in the underflow case, it would be pointless to search for a less-stringent sufficient condition.

The reason for this close agreement between sufficient and necessary conditions in the underflow case lies in the respective restrictions of the variable $\tilde{x}_i$, $i \neq j$, $i = 1, 2, \ldots, n$, to the domains $\tilde{X}_i$ and $\hat{X}_i$. That is, the integer elements of $\tilde{X}_i$ are closely interspersed throughout the integer elements of $\hat{X}_i$, $i = 1, 2, \ldots, n$, so that the magnitude bounds on the sets $\tilde{X}_i$ and $\hat{X}_i$ are nearly the same. On the other hand, in the overflow case there is a marked difference between the sufficient and necessary conditions, which follows similarly from the restrictions of the variable x_i, $i \neq j$, $i = 1, 2, \ldots, n$, to the respective domains X_i and $\tilde{X}_i$. Unlike $\tilde{X}_i$ and $\hat{X}_i$, there is a certain region of X_i which lies outside $\tilde{X}_i$ in magnitude by a significant amount.

The coefficients in the necessary condition (6.94) and sufficient condition (6.98) for the underflow relation (6.5) are

quite large. They are roughly $2^{N_j + m_i + 1}$; e.g., if $m_i = 8$ (bits) and $N_j = 4$, corresponding to 4-bit coefficients in the state equations, then the coefficients are 2^{13}, or 8192, which gives a very severe diagonal-dominance requirement. We obtain this severe diagonal-dominance requirement because we have required the underflow relation (6.5) to hold over the entire domain $\tilde{X}_i$, $i = 1, 2, \ldots, n$, in (6.84). In Section 6.7, we will show that it is sufficient for the underflow relation (6.5) to hold over a greatly restricted subset of $\tilde{X}_i$, $i = 1, 2, \ldots, n$, in order to suppress zero-input underflow oscillations. This will greatly reduce the coefficients in the corresponding diagonal-dominance conditions.

6.7 Reduction of Diagonal-Dominance Conditions on W: Zero-Input Underflow Stability

Here we will show that to suppress zero-input underflow limit cycles, it is sufficient that (6.5) holds over a much smaller domain of $\tilde{x}_i$ than in (6.84). This sufficient smaller domain is any hypercube in $\tilde{X}$ which bounds all zero-input underflow limit cycles. By the same method as in Section 6.6, we will obtain a sufficient diagonal-dominance condition on W for (6.5) to hold in this smaller domain. This dominance condition has the form of (6.98), but greatly reduced coefficients.

A hypercube H bounding all zero-input underflow limit cycles can be found by several methods in the literature [D3, D5, D7-D9]. These methods utilize either the "bounded-input-implies-bounded-output" property of linear systems or the Lyapunov approach. In the bounded-input-bounded-output method, the inputs are considered

to be worst case roundoff or truncation errors, so the principle is very similar to that used for bounds on roundoff noise. The Lyapunov approach has not produced very tight bounds, but may admit further development. Here we require a bounding hypercube based on the use of magnitude truncation at the state variables. The specific method of obtaining the hypercube is not important, except that we desire the tightest possible bounds. We define the hypercube in the following way. Let H_i be the set of all $\tilde{x}_i \in \tilde{X}_i$ such that

$$|\tilde{x}_i| \leq 2^N L_i, \; i = 1, 2, \ldots, n. \tag{6.100}$$

Here $2^N L_i$ is a magnitude bound on scaled zero-input underflow limit cycles which may appear in x_i, $i = 1, 2, \ldots, n$. Then the hypercube H consists of all $\tilde{x}$ such that

$$\tilde{x}_i \in H_i \;, \; i = 1, 2, \ldots, n. \tag{6.101}$$

Thus, <u>if</u> an underflow limit cycle exists, then for all $\hat{x}(k)$, $\tilde{x}(k)$, $x(k)$, $k = 0, 1, 2, \ldots$, in the limit cycle we have

$$\hat{x}, \tilde{x}, x \in H. \tag{6.102}$$

We now show that (6.5) is sufficient with the domain (6.102) to suppress zero-input underflow limit cycles. First, if (6.5) holds under condition (6.102), then (6.1) holds under condition (6.102), since we already have (6.2) which we may specialize to $x \in \tilde{X}$. That is, (6.1) follows by addition of (6.5) and (6.2). Consider a potential zero-input underflow limit cycle. By (6.1) under condition (6.102), the energy function $\hat{x}^T W \hat{x}$ decreases; in fact, it decreases

strictly for at least one state in a potential limit cycle, as discussed in Section 6.2. Thus, for any such state trajectory, we ultimately attain $\hat{x}^T W \hat{x} = 0$, i.e., $\hat{x} = 0$. This rules out the existence of a limit cycle.

We now find a sufficient diagonal-dominance condition for (6.5) to hold in the domain (6.102). As in Section 6.6, (6.5) leads to the sufficient condition (6.70) and its equivalent condition (6.82). Since the domain in (6.102) is contained in (6.84), we use the same argument as in Section 6.6 to obtain sufficient conditions. In this case we have

$$\max \hat{h}_j \leq 2 \sum_{\substack{i=1 \\ i \neq j}}^{n} 2^N L_i \, | w_{ij} | \, ; \tag{6.103}$$

$\max \hat{g}_j$ is the same as (6.93). This gives the sufficient diagonal-dominance condition

$$w_{jj} \geq 2^{(N_j - N+1)} \sum_{\substack{i=1 \\ i \neq j}}^{n} 2^N L_i | w_{ij} | \, ,$$

which is equivalent to the unscaled condition

$$w_{jj} \geq 2^{(N_j + 1)} \sum_{\substack{i=1 \\ i \neq j}}^{n} L_i \, | w_{ij} | \, . \tag{6.104}$$

A necessary condition is essentially the same as the sufficient condition (6.104), analogous to Section 6.6.

We observe that the coefficients of (6.104) are greatly reduced by comparison to those in (6.98); (6.104) is a sufficient diagonal-dominance condition for suppression of underflow limit cycles. Significantly, the coefficients in (6.104) do not increase with signal word length, as the coefficients do in (6.98). This is because zero-input underflow limit cycles occupy the first-few least-significant bits in $\hat{x}_i$, $i = 1, 2, \ldots, n$; they are totally unaffected by an increase in the number of bits to the left. In this sense, (6.104) is a more satisfactory condition than (6.98).

6.8 Stability of the Forced Response

In this Section, we define asymptotic stability of the forced response (ASFR) in a digital filter. This definition is based on a standard interpretation of stability; i.e., an appropriately defined error must eventually decay to zero. Also, we show that ASFR implies stability of the forced response against overflow limit cycles. It is not necessary to consider stability of the forced response against underflow limit cycles, as mentioned in Section 6.1. In Sections 6.9 through 6.13, we obtain sufficient conditions for ASFR, as follows. In Section 6.9, we analyze the forced response, to develop a model of the effects of overflow non-linearities. In Section 6.10, we show that the Lyapunov function $\bar{x}^T W \bar{x}$ will carry over from the linear prototype system (3.1) to the model in Section 6.9, if we resolve the effects of a certain time-varying non-linearity. This leads to a dynamic-range constraint and a magnitude truncation requirement, which are given in Sections 6.11 and 6.12, respectively.

In Section 6.13, we then derive diagonal-dominance conditions on the Lyapunov function $x^T Wx$, to completely resolve the effects of the time-varying non-linearity. This gives ASFR.

We will define ASFR in the digital filter (4.23), based on the following assumptions:

(a) the initial state $\hat{x}(o) \in \hat{X}$ is arbitrary in (4.23);

(b) the input sequence is the same in systems (3.1) and (4.23).

ASFR (asymptotic stability of the forced response) is defined as follows: the sequence of state vectors $\hat{x}(k)$, $k = 0, 1, 2, \ldots$, in (4.23) converges to the sequence of state vectors $\bar{x}(k)$, $k = 0, 1, 2, \ldots$, in (3.1), within a deviation due only to the effects of underflow errors. Here convergence is taken in the following sense: for any $\varepsilon > 0$ which is large enough to accomodate the effects of underflow errors, there exists some K such that $||\hat{x}(k) - \bar{x}(k)|| < \varepsilon$ for $k \geq K$. Note that $\bar{x}(k)$ is bounded for bounded input in (3.1).

Clearly, a necessary condition for ASFR is a dynamic-range constraint on $\bar{x}(k)$, $k = 0, 1, 2, \ldots$, in (3.1). That is, the magnitudes of the state variables for $k \geq K$ must fall within the magnitude range of the set of "machine numbers" $\hat{x} \in \hat{X}$. Otherwise the above-mentioned convergence of $\hat{x}(k)$ to $\bar{x}(k)$ is impossible. We will impose a dynamic-range constraint for all k, $k \geq 0$, without significant loss of generality. In Section 6.11, we define this dynamic-range constraint in terms of a "headroom" parameter, $\bar{H}_i$, $i = 1, 2, \ldots, n$; in practice, we satisfy the constraint by restricting the allowable set of initial states $\bar{x}(o)$ and input sequences $u(k)$, $k = 0, 1, 2, \ldots$, in (3.1). In Section 6.13,

the sufficient conditions to obtain the desired Lyapunov function will include the dynamic-range constraint.

A key feature of ASFR is the arbitrary initial state $\hat{x}$ (o) in (4.23), which need not be the same as the initial state $\bar{x}$ (o) in (3.1). That is, we allow an initial error in the state vector. The arbitrary initial state $\hat{x}$ (o) introduces the possibility of overflows in (4.23), despite the above-mentioned dynamic-range constraint on $\bar{x}$ (k), k = 0, 1, 2, ..., in (3.1). However, such overflows would not truly be driven by the input. ASFR is very useful because it guarantees that $\hat{x}$ (k) will converge to $\bar{x}$ (k) in the presence of such overflows, within a deviation due only to underflow errors. That is, the transient due to an initial state error, including the overflows which it may induce, will decay to zero.

We now show that ASFR implies stability of the forced response response against overflow limit cycles. This result is closely related to the operating conditions under which it is conventional to define stability against overflow limit cycles. Under these operating conditions, the input sequence causes the state variables in the system (3.1) to enter the overflow region of θ only occasionally; otherwise the digital filter (4.23) will operate with large errors. For analytic purposes, we may assume an input sequence which drives the linear system (3.1) into the overflow region only prior to a certain discrete-time instant J. Thereafter, the system (3.1) does not enter the overflow region; i.e., from time J onward we have the necessary dynamic-range constraint for ASFR. Further, we recognize that the effect of any possible sequence of overflows in (4.23) prior

to time J is only to generate some (possibly) incorrect state vector $\hat{x}(J)$, since in all events the state vector must lie in $\hat{X}$. At worst, this amounts to an arbitrary state vector at time J. We may translate the discrete-time axis J units to the left, since the system is time-invariant, to apply the definition of ASFR. By ASFR, the sequence of state vectors in (4.23) converges to the sequence of state vectors in (3.1) after time J, within a deviation due only to the effect of underflow errors. This rules out the possibility of an overflow limit cycle. (Note that the type of input sequence used here has also been used by Claasen et al. [D10] and earlier by Wilson [D4].)

We now consider the possibility of recurring time intervals during which the input drives the system (3.1) into the overflow region. We assume sufficiently large intervening time intervals during which the system (3.1) is in the non-overflow region; this is the normal mode of operation, since overflow conditions must not occur too frequently. From the discussion in the previous paragraph, ASFR implies that the response of the digital filter begins to converge to the correct response during any one of these intervening non-overflow time intervals. This is sufficient to rule out the possibility of self-sustaining overflow limit cycles.

6.9 Analysis of the Forced Response: Effect of Overflow Non-Linearities

We now analyze the forced response of the digital filter (4.23), to obtain a model of the effects of overflow non-linearities. We do this by decomposing the state vector $\hat{x}(k)$, $k = 0, 1, 2, \ldots,$

in (4.23) into three components: (a) $s(k)$, $k = 0, 1, 2, \ldots$, is the forced response of the prototype linear system (3.1) to the same input sequence as in (4.23); (b) $r(k)$, $k = 0, 1, 2, \ldots$, is the response of the system (3.1) to the sequence of underflow errors $d(k)$, $k = 0, 1, 2, \ldots$, which occurs in (4.23); (c) $e(k)$, $k = 0, 1, 2, \ldots$, is the sequence of state error vectors which corresponds to the initial state error $e(o)$. Thus

$$\hat{x}(k) = s(k) + r(k) + e(k), \quad k = 0, 1, 2, \ldots \quad . \tag{6.105}$$

Clearly, ASFR is equivalent to

$$\lim_{k \to \infty} e_i(k) = 0 \ , \quad i = 1, 2, \ldots, n; \tag{6.106}$$

hence, we will investigate the behavior of $e(k)$, $k = 0, 1, 2, \ldots$.

Our immediate objective is to show how $e(k+1)$ evolves from $e(k)$. To this end, we note from (4.23) that $\hat{x}(k+1)$ evolves from $\hat{x}(k)$ in three stages: (a) $\hat{x}(k)$ to $x(k+1)$; (b) $x(k+1)$ to $\theta(x(k+1))$; (c) $\theta(x(k+1))$ to $\hat{x}(k+1) = u \circ \theta(x(k+1))$. In the first stage we have, from (4.23), (6.105), and (3.1),

$$\begin{aligned} x(k+1) &= A\,\hat{x}(k) + B\,u(k) \\ &= A\,s(k) + B\,u(k) + A\,r(k) + A\,e(k) \\ &= s(k+1) + A\,r(k) + A\,e(k); \end{aligned} \tag{6.107}$$

here we have identified $s(k+1)$. In the second stage we have

$$\theta(x(k+1)) = \theta(s(k+1) + A\,r(k) + A\,e(k)) \quad . \tag{6.108}$$

Here we assign the effects of the overflow characteristic θ only to the error term $A\,e(k)$ in (6.108), since the terms $s(k+1)$ and $A\,r(k)$ already represent the response of the linear system (3.1). In other words, we set

$$\theta(x(k+1)) = s(k+1) + A\,r(k) + e(k+1); \tag{6.109}$$

from (6.109) and (6.108) it follows that

$$e(k+1) = \theta(s(k+1) + A\,r(k) + A\,e(k)) - s(k+1) - A\,r(k)\,. \tag{6.110}$$

In the final stage we take

$$\hat{x}(k+1) = u \circ \theta(x(k+1)) = s(k+1) + A\,r(k) + d(k) + e(k+1) \tag{6.111}$$

where $d(k)$ is the underflow error. Since

$$A\,r(k) + d(k) = r(k+1)\,, \tag{6.112}$$

we may rewrite (6.111) as

$$\hat{x}(k+1) = s(k+1) + r(k+1) + e(k+1)\,, \tag{6.113}$$

which is consistent with (6.105). Note that we obtain from (6.112) the following explicit formulation of underflow error:

$$r(k) = \sum_{i=0}^{k} A^{k-i}\,d(i)\,, \tag{6.114}$$

where d (i), i = 1, 2, ..., k, is the sequence of underflow errors.

The behavior of the error term e (k), k = 0, 1, 2, ..., is completely specified by (6.110). For clarity, we rewrite (6.110) as

$$e\ (k+1) \ = \theta\ (v(k) + A\ e\ (k)) - v\ (k) \quad , \tag{6.115}$$

where we have set

$$v\ (k) \ \triangleq \ s\ (k+1) + A\ r\ (k) \ . \tag{6.116}$$

The non-linear system (6.115) has initial state e (o) with no forcing function and consists of two distinct parts: (a) a linear operation on e (k) by the transition matrix A; (b) a time-varying non-linearity in the feedback path. We will denote the time-varying non-linearity by F (w(k), k), i.e.,

$$e\ (k+1) = F\ (w(k),\ k) \triangleq \theta\,(v(k) + w\ (k)) - v\ (k) \tag{6.117}$$

where we set

$$w\ (k) \ \triangleq \ A\ e\ (k) \quad ; \tag{6.118}$$

note that (6.117) and (6.118) are together equivalent to (6.115). If F (w(k), k) were replaced by a direct feed-through, the system (6.117) and (6.118) would be stable, since A is the stability matrix from the original system (3.1). However, the presence of F (w(k), k) produces a more difficult problem. (The above formulation of e (k), k = 0, 1, 2, ..., which culminates in (6.117) and (6.118), is essentially the same <u>in principle</u> as the approach of Claasen et al. [D10]. As they mention, the principle remains valid if we replace the overflow

characteristic θ by a more general non-linearity; however, there is no need to do that here. On the other hand, they do not include the underflow response r (k), k = 0, 1, 2, ..., in their analysis.)

Claasen et al. [D10] consider a stability problem similar to that for (6.117) and (6.118), with certain slope restrictions on the overflow characteristic θ. These slope restrictions give a well= defined region for the time-varying non-linearity F (w(k), k). They mention various methods in the literature which are applicable to the stability analysis of the system (6.117) - (6.118), assuming these well-defined regions for F (w(k), k). In particular, they emphasize the Lyapunov approach. They point out that Wilson [D4] has previously applied the Lyapunov approach to obtain sufficient conditions for stability of the forced response in certain restricted second-order sections. They also point out that the Lyapunov approach yields a proof of the stability of the forced response in wave digital filters; however, they do not give a detailed proof.

6.10 Application of Lyapunov Stability Theory to the Forced Response

We will show that it is possible to obtain stability of the system (6.117) - (6.118), i.e., to obtain (6.106), by the Lyapunov approach, under a certain condition on the linear prototype system (3.1). This condition on (3.1) is the existence of a Lyapunov function $\bar{x}^T W \bar{x}$ for (3.1), where the matrix W exhibits a sufficient degree of diagonal dominance. In particular, we consider linear prototype realizations (3.1) where the (positive-definite) solution W in the Lyapunov matrix equation (3.11) or (5.3) is diagonal,

before coefficient quantization in A and C. We expect that W will remain diagonally dominant after coefficient quantization, as discussed in Section 4.7. We have chosen the particular Lyapunov matrix equation (3.11) because, as discussed in Section 5.4, its solution W is diagonal for MRN realizations before coefficient quantization. Since we have shown in Section 5.7 that the wave-digital realization fits into the framework of the two-input-two-output MRN realization, our derivation will include stability of the forced response in the wave-digital realization as a special case; stability of the forced response in the wave-digital realization was previously pointed out by Claasen et al. [D10]. The actual method consists of carrying over the Lyapunov function $\bar{x}^T W \bar{x}$ from the linear system (3.1). For this purpose, it will be necessary to impose certain magnitude-truncation requirements on the time-varying non-linearity F (w(k), k) which derives from (4.23).

We outline the application of the Lyapunov approach to the system (6.117) - (6.118), as follows. From (3.11), it follows that

$$V(k) = \bar{x}^T(k)\, W\, \bar{x}(k) \tag{6.119}$$

is a Lyapunov function of the linear system (3.1); i.e.,

$$V(k) - V(k+1) \geq 0, \quad k = 0, 1, 2, \ldots \ . \tag{6.120}$$

By applying this Lyapunov function of the linear system (3.1) to the linear part (6.118), we have

$$e^T(k)\, W\, e(k) - w^T(k)\, W\, w(k) \geq 0,\ k=0,1,2,\ldots, \quad (6.121)$$

If we can establish

$$w^T(k)\, W\, w(k) - e^T(k+1)\, W\, e(k+1) \geq 0,\ k=0,1,2,\ldots, \quad (6.122)$$

for the non-linear part (6.117), then we will have

$$e^T(k)\, W\, e(k) - e^T(k+1)\, W\, e(k+1) \geq 0,\ k=0,1,2,\ldots, \quad (6.123)$$

by addition of (6.121) and (6.122). This will establish that $V(k)$ is also a Lyapunov function of the system (6.117) - (6.118), i.e., (6.120) holds <u>if we take $e(k)$ in place of $\bar{x}(k)$ in (6.119).</u>

In Section 6.13, we will establish sufficient conditions for (6.122), hence (6.123) or (6.120). These sufficient conditions will be the previously mentioned diagonal-dominance conditions on W and magnitude-truncation conditions on $F(w(k), k)$. Exceeding the sufficient conditions slightly will guarantee that (6.122), hence (6.123) or (6.120), are bounded away from zero whenever $v(k) + w(k)$ in (6.117) enters the overflow region of θ; by "bounded away from zero" we mean

$$w^T(k)\, W\, w(k) - e^T(k+1)\, W\, e(k+1) \geq \delta \quad (6.124)$$

or

$$V(k) - V(k+1) \geq \delta, \quad (6.125)$$

where δ is a small positive number. The sufficient conditions will

also include the necessary dynamic-range constraint mentioned in Section 6.8.

We now outline certain results concerning the behavior of $e(k)$, $k = 0, 1, 2, \ldots$, in (6.117) - (6.118), which will follow from establishing (6.122) and (6.124), hence (6.120) and (6.125), in Section 6.13. First, the system (6.117) and (6.118) must eventually operate entirely within the non-overflow region of θ; i.e., for some K

$$\theta\,(v(k) + w(k)) = v(k) + w(k),\ k \geq K\ , \tag{6.126}$$

which gives

$$F(w(k), k) = w(k),\quad k \geq K. \tag{6.127}$$

Thus, the system (6.117) and (6.118) becomes the force-free linear system

$$e(k+1) = A\,e(k),\quad k \geq K. \tag{6.128}$$

We show (6.126) by contradiction, as follows. If there is no K such that $v(k) + w(k)$ lies in the non-overflow region of θ for $k \geq K$, then (6.125) would hold for an infinite number of indices k. Thus $V(k)$ would decrease below zero, which is a contradiction. Note that $w^T(k)\,W\,w(k)$ would likewise decrease below zero, since $V(k) \geq w^T(k)\,W\,w(k) \geq V(k+1)$. Thus, in actual fact, $w^T(k)\,W\,w(k)$ decreases to the point where $v(k) + w(k)$, $k \geq K$, lies in the non-overflow region of θ for some K. This is always possible due to the dynamic-range constraint on $v(k)$, $k = 0, 1, 2, \ldots$.

We complete the discussion of the behavior of e (k), k = 0, 1, 2, ..., by considering (6.128), which is the asymptotic characterization of the system (6.117) and (6.118). Since the Lyapunov matrix equation (3.11) applies to the linear system (6.128), the output observed from the sequence e (k), $k \geq K$, must converge to zero. If the system (3.1) is completely observable, then the sequence e (k), $k \geq K$, must converge to zero as well. Recalling that e (o) represents the error in the state vector, we may summarize our results as follows: (1) the output error (exclusive of underflow effects) converges to zero; (2) the state error (exclusive of underflow effects) converges to zero if the system (3.1) is observable. That is, we have output-ASFR and ASFR, respectively.

6.11 Dynamic-Range Constraint

We now set forth the dynamic-range constraint at the state variables; as mentioned in Section 6.8, this constraint is necessary for ASFR. The dynamic-range constraint is a set of magnitude bounds which we impose on the components of v (k), k = 0, 1, 2, ..., defined in (6.116). We have chosen to apply the constraints to v (k), rather than to s (k) alone, to include the effect of underflow errors represented by r (k). We will constrain the magnitude of each component of v (k) to remain within the non-overflow region of θ. This guarantees that the digital system (4.23), which includes underflow errors, will not enter the overflow region, if the initial state error e (o) is zero.

The dynamic-range constraint on $v(k)$ is also a necessary condition for magnitude truncation in the time-varying non-linearity $F(w(k), k)$, defined in (6.117), if we consider all possible values for the variables $v_i(k)$ and $w_i(k)$, $i = 1, 2, \ldots, n$. This follows since the magnitude truncation requirement is equivalent to

$$-w_i(k) \leq \theta_i(v_i(k) + w_i(k)) - v_i(k) \leq w_i(k), \quad i = 1, 2, \ldots, n,$$

or

$$v_i(k) - w_i(k) \leq \theta_i(v_i(k) + w_i(k)) \leq v_i(k) + w_i(k), \quad i = 1, 2, \ldots, n. \tag{6.129}$$

Recalling the basic requirement on the set of "non-overflow" numbers $\tilde{X}$, i.e.,

$$-\tilde{R}_i \leq \theta_i(v_i(k) + w_i(k)) \leq \tilde{R}_i, \tag{6.130}$$

we must have

$$v_i(k) - w_i(k) \leq \tilde{R}_i \tag{6.131}$$

and

$$v_i(k) + w_i(k) \geq -\tilde{R}_i, \tag{6.132}$$

which follow from (6.129) and (6.130). Since (6.131) and (6.132) must hold for any $w_i(k)$ in the interval

$$-R_i \leq w_i(k) \leq R_i, \tag{6.133}$$

in particular for w_i (k) near zero, we obtain the necessary dynamic-range constraint

$$- \tilde{R}_i \leq v_i \ (k) \leq \tilde{R}_i \quad , \quad k = 0, 1, 2, \ldots, \tag{6.134}$$

for i = 1, 2, ..., n. As mentioned in Section 6.10, magnitude truncation in F (w (k), k) will be a necessary condition for (6.122).

We will strengthen the dynamic-range constraint by introducing a parameter $\bar{H}_i$, i = 1, 2, ..., n, in (6.134). That is, in place of (6.134) we will require

$$- \tilde{R}_i + \bar{H}_i \leq v_i \ (k) \leq \tilde{R}_i - \bar{H}_i, \ k = 0, 1, 2, \ldots, \tag{6.135}$$

for i = 1, 2, ..., n, where

$$0 \leq \bar{H}_i \leq \tilde{R}_i \quad . \tag{6.136}$$

The purpose of this parameter is to provide a margin for the signal v_i (k), k = 0, 1, 2, ..., (i.e., some "headroom") below the overflow region of θ_i. The severity of the diagonal-dominance conditions to be given in Section 6.13 will be a function of the parameter $\bar{H}_i$, i = 1, 2, ..., n.

Sufficient magnitude constraints on the components of s (k), k = 0, 1, 2, ..., can be deduced from (6.135), since v (k), k = 0, 1, 2, ..., is the sum of s (k) and r (k). This sufficient constraint on s (k) follows by bounding the components of r (k) (i.e., the roundoff noise), which are the effects of underflow errors. Subtracting these bounds on r (k) from the limits in (6.135) gives the sufficient magnitude constraints on s (k). Bounds on

r (k), k = 0, 1, 2, ..., can be obtained by several methods in the literature. It is shown by (6.114) that r (k) is the response of the system (3.1) to the sequence of underflow error vectors d (k), k = 0, 1, 2, ..., which occurs in (4.23). (See also the related discussion in Sections 4.5 and 5.2.) The bound on r (k) follows by the bounded-input-implies-bounded output property of linear systems. We expect that this bound will be relatively small, since the components of the underflow error vectors are small in magnitude.

The bounds deduced on s (k) can be satisfied by suitable magnitude bounds on the input sequences in (3.1). However, in practice, the magnitude restrictions on the input need only be severe enough to ensure that the bounds on s (k) are satisfied for sufficiently long intervals in discrete time. ASFR guarantees that the digital-filter response will converge toward the correct response during these long intervals, as discussed in Section 6.8. The analysis of the magnitudes which occur at the state variables, due to various types of inputs, is well suited to the application of probability theory.

6.12 Magnitude Truncation in Time-Varying Non-Linearity

Magnitude truncation in the time-varying non-linearity F (w (k), k), defined in (6.117), will be a necessary condition to establish (6.122) by the diagonal-dominance conditions in Section 6.13. Therefore, we will consider necessary and sufficient conditions on the overflow characteristic θ for magnitude truncation in F (w (k), k). We have already shown that (non-strict) magnitude

truncation in $F(w(k), k)$ is equivalent to (6.129); from (6.129) we deduced the dynamic-range constraint (6.134) on $v_i(k)$, $i = 1, 2, \ldots, n$.

We now deduce necessary and sufficient conditions on θ_i, $i = 1, 2, \ldots, n$, for (6.129), under the dynamic-range constraint (6.134). We fix $v_i(k)$ at any value in the interval (6.134) and consider θ_i as a function of its argument, $v_i(k) + w_i(k)$, in (6.129). This defines a wedge-shaped sector enclosed by two line segments, of slope +1 and -1, emanating from a vertex at the point $(v_i(k), v_i(k))$; the sector lies to the right (left) of the vertex if we fix $v_i(k)$ as positive (negative). The only sector for positive (negative) $v_i(k)$ which lies within all the other sectors generated by positive (negative) $v_i(k)$ is that generated by the limiting value $v_i(k) = +\tilde{R}_i$ ($v_i(k) = -\tilde{R}_i$). Hence, the positive (negative) segment of the overflow characteristic θ_i must fall within the sector generated by the positive (negative) limiting value; it must also satisfy the magnitude constraint (6.130). Thus, it is necessary and sufficient that θ_i, $i = 1, 2, \ldots, n$, lie within the hatched region shown in Fig. 1 (Cf. Fig. 6 in [D10]). The corresponding $F_i(w_i(k), k)$, $i = 1, 2, \ldots, n$, will lie within the hatched region in Fig. 2 (Cf. Fig. 5 in [D10]). Note that excluding the boundary segments of slope -1 from the hatched region gives strict magnitude truncation in $F_i(w_i(k), k)$ in the event of an overflow condition.

We will restrict the overflow characteristic θ_i, $i = 1, 2, \ldots, n$, to the saturation characteristic, for various reasons given in Section 5.3. A more immediate reason for choosing the saturation

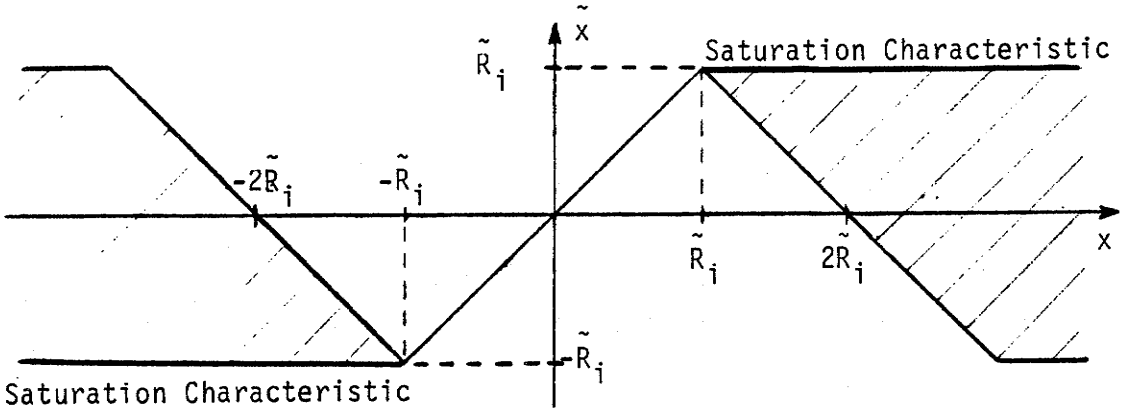


Figure 1. θ_i Restricted for Magnitude Truncation in $F_i(w_i(K),K)$

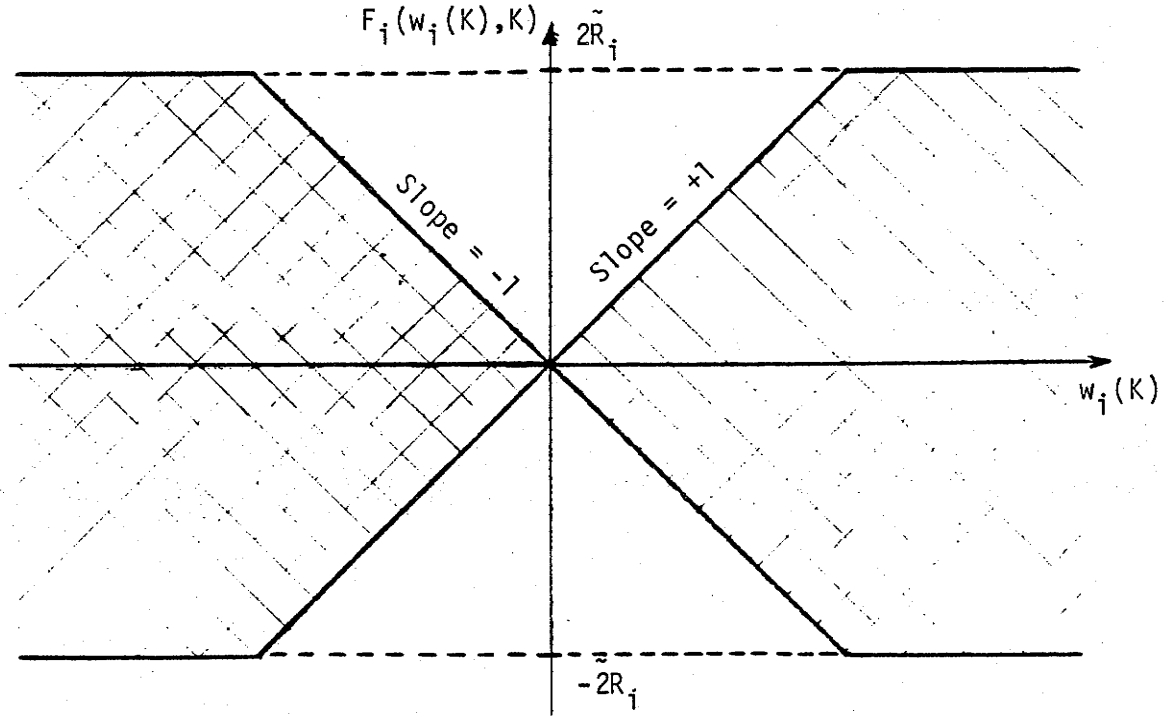


Figure 2. $F_i(w_i(K),K)$ Corresponding to θ_i

characteristic is to obtain the least-severe diagonal-dominance conditions on W for (6.122). We show this in Section 6.13. The true saturation characteristic can be implemented without any difficulty in the distributed-arithmetic structure, as described in Chapter 8. The saturation characteristic corresponds to the upper (lower) bound on the right-hand (left-hand) hatched region in Fig. 1. The corresponding time-varying non-linearity F_i (w_i (k), k) is shown in Fig. 3; here the magnitude-truncated result has the same sign as the argument w_i (k). If, in place of (6.134), we use the strengthened dynamic-range constraint (6.135), which includes the margin $\bar{H}_i$, the region of F_i (w_i (k), k) is further restricted; this is shown in Fig. 4.

6.13 Diagonal-Dominance Conditions on W: Stability of the Forced Response

Here we establish diagonal-dominance conditions on W for (6.122), hence (6.123) or (6.120). This will establish e^T (k) W e (k) as a Lyapunov function of the system (6.117) - (6.118) and hence establish ASFR for the system (4.23). We show separate necessary conditions and sufficient conditions of diagonal dominance for (6.122); the sufficient conditions will be somewhat more severe than the necessary conditions. In many respects, the development in this section is analogous to that in Sections 6.3 through 6.5; thus, we will refer to those Sections for subsidiary details wherever possible.

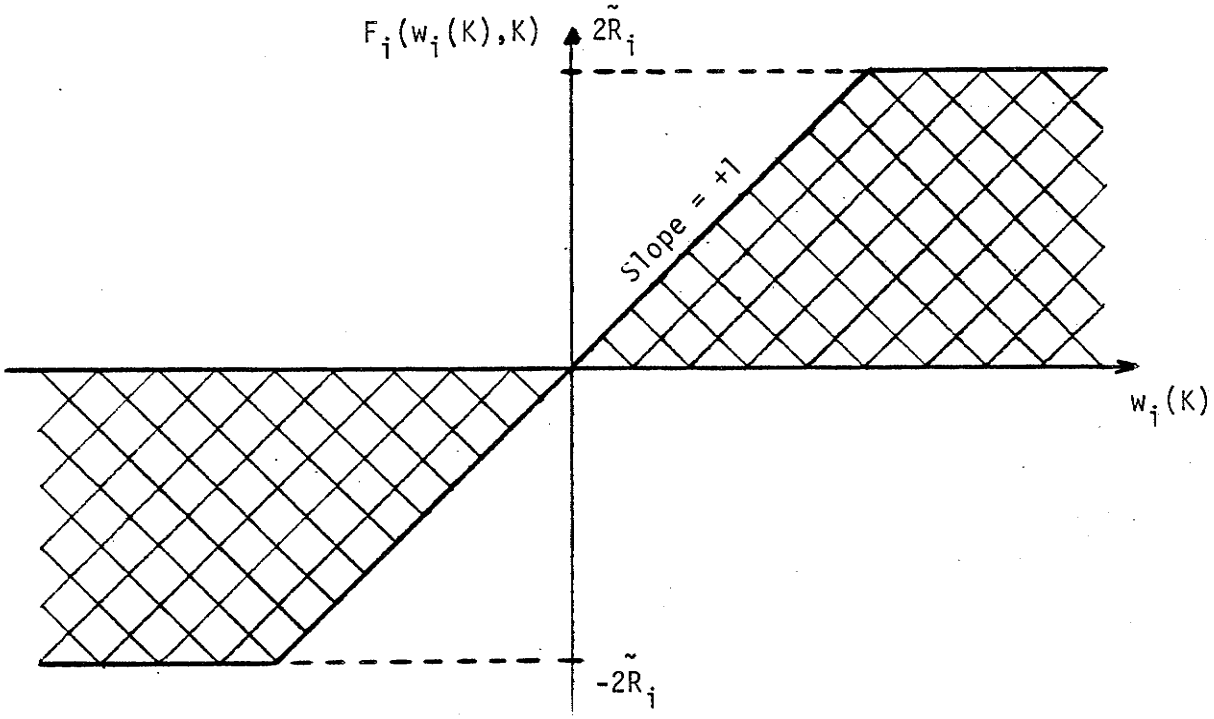


Figure 3. $F_i(w_i(K),K)$ Corresponding to Saturation Characteristic for θ_i

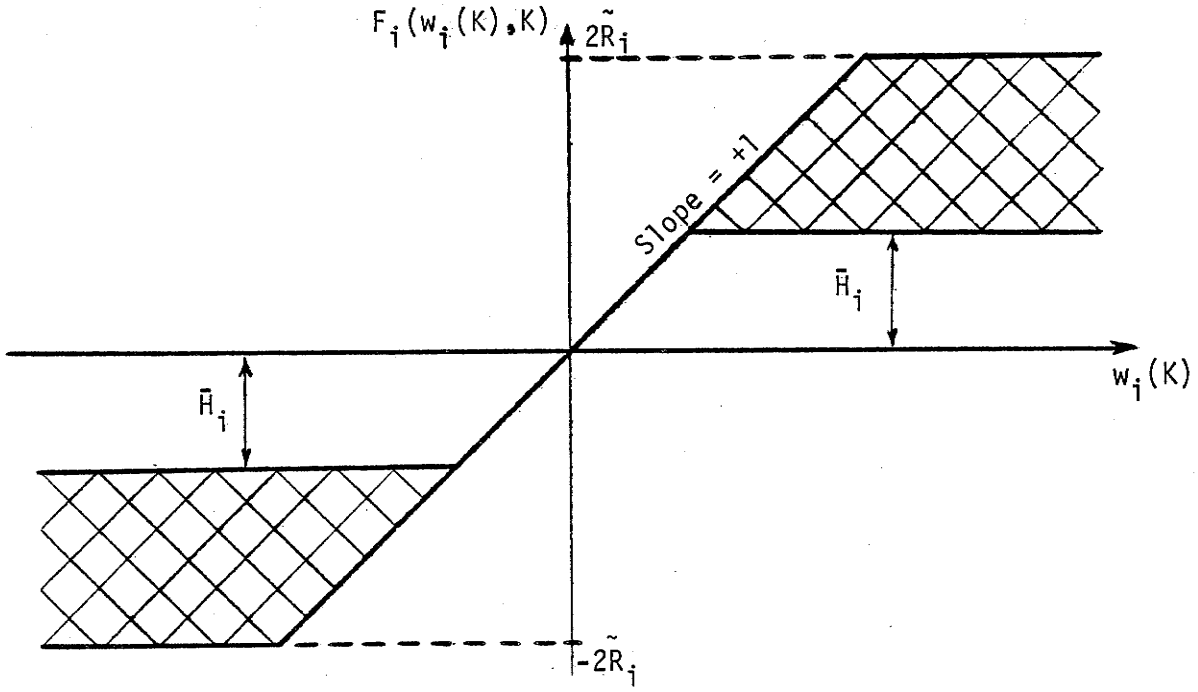


Figure 4. $F_i(w_i(K),K)$ Corresponding to Saturation Characteristic For θ_i with Margin $\bar{H}_i$ on Dynamic-Range Constraint

We will first show that (non-strict) magnitude truncation in the time-varying non-linearity F (w (k), k) is a necessary condition for (6.122). The proof of this is identical to the proof in Section 6.3 for the same condition on the mapping x to $\hat{x}$. For the present purpose, it is only necessary to replace x and $\hat{x}$ in the proof in Section 6.3 by w (k) and e (k+1), respectively. The diagonal-dominance conditions which we will derive here are based on magnitude truncation in F (w (k), k).

The preliminary algebraic formulation, which we require to obtain the diagonal-dominance conditions, is analogous to the formulation in Section 6.4. First, we define

$$\bar{w}\ (k) \ \stackrel{\Delta}{=}\ e\ (k+1) \tag{6.137}$$

strictly for convenience in notation and rewrite (6.122) as

$$w^T\ (k)\ W\ w\ (k) - \bar{w}^T\ (k)\ W\ \bar{w}\ (k) \geq 0,\ k = 0,1,2,\ldots\ . \tag{6.138}$$

Note that $\bar{w}_i\ (k)$ is a (non-strictly) magnitude-truncated version of $w_i\ (k)$, i = 1, 2, ..., n; hereafter we delete the index k wherever w (k) and $\bar{w}$ (k) appear in the same equation. We now obtain

$$w^T\ W\ w - \bar{w}^T\ W\ \bar{w}\ \equiv (w + \bar{w})\ W\ (w - \bar{w}) \tag{6.139}$$

from (6.11) in Section 6.4, by replacing x and $\tilde{x}$ by w and $\bar{w}$, respectively, in (6.11). From (6.138) and (6.139), we have

$$(w + \bar{w})\ W\ (w - \bar{w})\ \geq\ 0, \tag{6.140}$$

which is equivalent to (6.122); this is analogous to (6.17) in

Section 6.4. Following (6.19) to (6.23) in Section 6.4, we establish

$$\sum_{j=1}^{n} C_j \geq 0 \tag{6.141}$$

as equivalent to (6.140), where we define

$$C_j \triangleq [(w+\bar{w})_1 w_{1j} + (w+\bar{w})_2 w_{2j} + \ldots + (w+\bar{w})_n w_{nj}] (w-\bar{w})_j \tag{6.142}$$

for $j = 1, 2, \ldots, n$. Here $(\ )_i$ means the i^{th} component of the vector $w + \bar{w}$ or $w - \bar{w}$ which it may enclose. Also, $w_{1j}, w_{2j}, \ldots, w_{nj}$ are elements in the matrix W, not to be confused with the vectors w and $\bar{w}$.

We will derive separate necessary and sufficient conditions for (6.122) in terms of C_j, $j = 1, 2, \ldots, n$, since (6.141) is equivalent to (6.122). The method which we will use here is analogous to that in Section 6.5. The conditions will follow from the values which the variables $(w \pm \bar{w})_i$, $i = 1, 2, \ldots, n$, in (6.142), can assume. First, we note from (6.107), (6.116), and (6.118) that

$$x(k+1) = v(k) + w(k) . \tag{6.143}$$

Recall that $v_i(k)$, $i = 1, 2, \ldots, n$, is subject to the dynamic-range constraint (6.135); so the condition at $x_i(k+1)$ with regard to overflows is dependent on the value which $w_i(k)$ assumes. The existence of a non-overflow condition in $x_i(k+1)$, which corresponds to

$$x_i(k+1) \in \tilde{X}_i , \tag{6.144}$$

gives

$$\bar{w}_i = w_i \tag{6.145}$$

from the time-varying non-linearity (6.117), in view of (6.143) and (6.137). The existence of an overflow condition in $x_i\ (k+1)$ means

$$x_i\ (k+1) \in X_i' , \notin \tilde{X}_i , \tag{6.146}$$

where X_i' is the set of overflow numbers appropriate to the forced response under the dynamic-range constraint. This gives

$$-w_i \leq \bar{w}_i < w_i \tag{6.147}$$

from the previously discussed magnitude truncation requirement on (6.117), in view of (6.143) and (6.137). (The strict inequality in the right-hand side of (6.147) follows from the hatched region in Fig. 1.) Note that the variables $x_i\ (k+1)$, $i = 1, 2, \ldots, n$, are essentially independent for different indices i, so any combination of overflow and non-overflow conditions may exist in this set of variables.

The <u>necessary condition</u>

$$C_j \Big|_{\substack{x_i\ (k+1) \in \tilde{X}_i \\ i \neq j,\ i = 1, 2, \ldots, n}} \geq 0, \tag{6.148}$$

for $j = 1, 2, \ldots, n$, follows by (6.141) and (6.142), since the conditions on $x_i\ (k+1)$ in (6.148) give (6.145) for $i \neq j$, $i = 1, 2, \ldots, n$, which implies $C_i = 0$ for $i \neq j$, $i = 1, 2, \ldots, n$. Note that in (6.148) the variables x_i, $i \neq j$, $i = 1, 2, \ldots, n$, are

restricted to the non-overflow region. The sufficient condition

$$C_j \Big|_{\substack{x_i\,(k+1)\,\varepsilon\,X'_i \\ i = 1, 2, \ldots, n}} \geq 0, \tag{6.149}$$

for $j = 1, 2, \ldots, n$, follows since (6.149) makes (6.141) a valid statement. In (6.149) the variables x_i are unrestricted with regard to the overflow region. Diagonal-dominance conditions on the columns of W will follow from (6.149) and (6.148) by converting them to a more explicit form. We carry out the development in terms of (6.149), since (6.148) is simply a special case of (6.149) where additional restrictions apply to the variables x_i. Note that the diagonal= dominance conditions will also apply to the rows of W, since W is symmetric.

There are two mutually exclusive possibilities in (6.149). The first possibility is that an overflow error does not occur at x_j, i.e., (6.144) holds for $i = j$, which gives $C_j = 0$ by (6.145) and (6.142). We need not consider the first possibility further, since $C_j = 0$ satisfies (6.149) with equality. The second possibility is that an overflow error does occur at x_j, i.e., (6.146) holds for $i = j$, which gives (6.147), i.e.,

$$(w - \bar{w})_j \neq 0 . \tag{6.150}$$

We now establish that under (6.146) strict magnitude truncation, i.e., (6.147) and

$$\bar{w}_j \neq -w_j , \tag{6.151}$$

is a necessary condition for (6.149), by rewriting C_j as

$$C_j = (w + \bar{w})_1 (w - \bar{w})_j w_{1j} + (w + \bar{w})_2 (w - \bar{w})_j w_{2j} + \ldots + (w + \bar{w})_n (w - \bar{w})_j w_{nj} \geq 0 \qquad (6.152)$$

from (6.142). For a given sign of $(w - \bar{w})_j$, it is possible to choose w_i, $i \neq j$, $i = 1, 2, \ldots, n$, such that the terms $(w + \bar{w})_i (w - \bar{w})_j w_{ij}$, $i \neq j$, $i = 1, 2, \ldots, n$, in (6.152), are all negative. The remaining term $(w + \bar{w})_j (w - \bar{w})_j w_{jj}$ must be greater than zero to guarantee the inequality in (6.152). Thus, since $w_{jj} > 0$, we must have the strict magnitude truncation

$$w_j^2 - \bar{w}_j^2 > 0 , \qquad (6.153)$$

which is equivalent to (6.147) and (6.151). Strict magnitude truncation in F (w (k), k) can be obtained by a slight additional restriction on the overflow characteristic θ, as described in Section 6.12.

We have established that the only non-trivial possibility in (6.149) is (6.150), where (6.147) and (6.151) must hold. We rewrite (6.149) as

$$C_j = (w + \bar{w})_j (w - \bar{w})_j \left[w_{jj} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(w + \bar{w})_i}{(w + \bar{w})_j} w_{ij} \right] \geq 0 \qquad (6.154)$$

by factoring out $(w + \bar{w})_j$ in (6.142); this is permissible since

$$(w + \bar{w})_j \neq 0 \qquad (6.155)$$

by (6.151). In view of (6.153),

$$w_{jj} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(w + \bar{w})_i}{(w + \bar{w})_j} w_{ij} \geq 0 \qquad (6.156)$$

is equivalent to (6.154). We rewrite (6.156) as

$$w_{jj} \geq - \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{(w + \bar{w})_i}{(w + \bar{w})_j} w_{ij} \quad , \qquad (6.157)$$

which is a diagonal-dominance condition on the j^{th} column of W, equivalent to (6.149).

It remains to establish suitable magnitude bounds on the terms $(w + \bar{w})_i/(w + \bar{w})_j$ in (6.157). We bound these terms under two different sets of restrictions, which correspond to the necessary condition (6.148) and the sufficient condition (6.149). These restrictions are, respectively,

$$x_i \in \tilde{X}_i \quad , \; i \neq j, \; i = 1, 2, \ldots, n \; , \qquad (6.158)$$

and

$$x_i \in X_i' \quad , \; i \neq j, \; i = 1, 2, \ldots, n \; . \qquad (6.159)$$

In both cases (6.158) and (6.159), we have the additional restriction

$$x_j \in X_j' \quad , \notin \tilde{X}_j \; , \qquad (6.160)$$

which was previously established. We have stated the above restrictions in terms of the variables x_i, i = 1, 2, ..., n, similar to Section 6.5. However, here the terms to be bounded are functions of the errors w_i and $\bar{w}_i$, i = 1, 2, ..., n, in the state variables,

whereas in Section 6.5 the terms to be bounded were functions of the state variables themselves. As in Section 6.5, we assume that the overflow characteristic θ_i, i = 1, 2, ..., n, has odd symmetry. This is a reasonable assumption for two reasons: the region to which θ_i is restricted in Fig. 1 exhibits odd symmetry; in any case, the natural overflow characteristic of the two's complement or sign-magnitude representation will require modification to conform to Fig. 1. Note that the time-varying non-linearity F (w (k), k) will also exhibit odd symmetry, with respect to the variables w (k) and v (k), if θ has odd symmetry.

In order to establish magnitude bounds for (6.157), we denote its right-hand side by the function

$$f_j\,(w_1, w_2, \ldots, w_n) \triangleq -\frac{1}{(w+\bar{w})_j} \sum_{\substack{i=1 \\ i\neq j}}^{n} (w+\bar{w})_i\, w_{ij}, \quad (6.161)$$

where we assume F (w (k), k) in (6.117) is fixed for the moment by fixing v (k). Further, we define the independent functions

$$g_j\,(w_j) \triangleq \frac{-1}{(w+\bar{w})_j} \quad (6.162)$$

and

$$h_j\,(w_1, w_2, \ldots, w_{j-1}, w_{j+1}, \ldots, w_n) \triangleq \sum_{\substack{i=1 \\ i\neq j}}^{n} (w+\bar{w})_i\, w_{ij}, \quad (6.163)$$

so that

$$f_j = g_j \; h_j \; . \tag{6.164}$$

We will bound g_j and h_j separately as functions of w_j and w_i, $i \neq j$, $i = 1, 2, \ldots, n$, respectively, for all values of the parameter $v(k)$ which fixes $F(w(k), k)$. This will bound f_j, by (6.164). By a symmetry argument similar to that in Section 6.5, it will be sufficient to use the positive extremes of g_j and h_j for bounds, since $F(w(k), k)$ has odd symmetry.

We first bound g_j as a function of w for all possible parameter values $v(k)$ in $F(w(k), k)$. Due to the odd symmetry in g_j, we may ignore the negative sign in (6.162). Thus, we seek the minimum positive value of $(w + \bar{w})_j$, which gives the maximum value for its reciprocal $-g_j$. This minimum positive value of $(w + \bar{w})_j$ is dependent on two elements: (1) the parameter $\bar{H}_j$ which specifies the headroom in the dynamic range constraint (6.135); (2) the overflow characteristic θ_j which determines $F_j(w_j(k), k)$. We desire to maximize the minimum positive value of $(w + \bar{w})_j$, because this minimizes the magnitude bound on g_j and hence minimizes the diagonal-dominance requirement by (6.157), (6.161), and (6.164). For an overflow to exist, i.e., for (6.160) to hold, we must have

$$w_j > \bar{H}_j \; , \tag{6.165}$$

in view of (6.135); this gives

$$(w + \bar{w})_j > \bar{H}_j + \bar{w}_j \; . \tag{6.166}$$

The parameter $\bar{H}_j$ will be chosen in practice by compromising between two conflicting requirements: the need to increase the lower positive bound on $(w + \bar{w})_j$ in (6.166); the need to increase the ratio of signal to roundoff noise. The remaining possibility for increasing the lower bound in (6.166) lies in the overflow characteristic θ_j. For a given w_j satisfying (6.165) and a given parameter v (k) in (6.117), we desire the overflow characteristic θ_j which gives the greatest value of $\bar{w}_j$ by F_j (w_j (k), k). It is not difficult to show that the saturation overflow characteristic gives an F_j (w_j (k), k) which is a pointwise upper bound on all other possibilities, regardless of the particular value of the parameter v (k). Thus, we assume the saturation overflow characteristic. In this case, we have

$$\bar{w}_j > \bar{H}_j , \tag{6.167}$$

which gives

$$(w + \bar{w})_j > 2 \bar{H}_j$$

from (6.166); the corresponding bound on g_j is

$$g_j < \frac{1}{2 \bar{H}_j} . \tag{6.168}$$

We now bound h_j in (6.163) for all possible values w_i, $i \neq j$, $i = 1, 2, \ldots, n$, where we assume the saturation overflow characteristic. We begin by bounding the terms $(w + \bar{w})_i$ in (6.163) under the separate restrictions on x_i in the non-overflow case (6.158) and the overflow case (6.159), which imply

$$- \tilde{R}_i \leq x_i \leq \tilde{R}_i \tag{6.169}$$

and

$$- R_i' \leq x_i \leq R_i' \quad , \tag{6.170}$$

respectively. R_i' is the magnitude bound on the set X_i'. In both cases, by (6.112), (6.113), and (6.116), we have

$$\hat{x}\,(k+1) = v(k) + d(k) + e(k+1) \ , \tag{6.171}$$

which implies by (6.135) that

$$- 2\,\tilde{R}_i + \bar{H}_i \leq e_i(k) \leq 2\,\tilde{R}_i - \bar{H}_i \quad , \ k = 0,1,2,\ldots, \tag{6.172}$$

for i = 1, 2, ..., n, where the relatively small overflow term d(k) has been neglected. From (6.137) and (6.172) it follows that in both cases

$$- (2-\alpha)\,\tilde{R}_i \leq \bar{w}_i \leq (2-\alpha)\,\tilde{R}_i, \ i = 1,2,..,\ n, \tag{6.173}$$

where

$$\alpha \overset{\Delta}{=} \frac{\bar{H}_j}{\tilde{R}_j} \quad , \quad 0 \leq \alpha \leq 1 \ . \tag{6.174}$$

For the non-overflow case, (6.145) and (6.173) give

$$- 2\,(2-\alpha)\,\tilde{R}_i \leq (w + \bar{w})_i \leq 2\,(2-\alpha)\,\tilde{R}_i, \ i=1,2,\ldots,n. \tag{6.175}$$

In the overflow case, it follows from (6.172) and (6.118) that

$$- \sum_{j=1}^{n} |a_{ij}|\ (2\tilde{R}_j - \bar{H}_j) \leq w_i \leq \sum_{j=1}^{n} |a_{ij}|\ (2\tilde{R}_j - \bar{H}_j), \tag{6.176}$$

i = 1, 2, ..., n, which gives

$$- (2 - \alpha)\, R_i \leq w_i \leq (2 - \alpha)\, R_i \tag{6.177}$$

by (4.1) and (6.174). From (6.173) and (6.177), for the overflow case

$$- (2-\alpha)\, (R_i + \tilde{R}_i) \leq (w+\tilde{w})_i \leq (2-\alpha)\, (R_i + \tilde{R}_i),\ i=1,2,\ldots,n. \tag{6.178}$$

The bounds on h_j corresponding to the necessary condition and the sufficient condition follow by using (6.175) and (6.178), respectively, in (6.163). For the necessary condition we have

$$h_j \leq 2\, (2-\alpha) \sum_{\substack{i=1 \\ i \neq j}}^{n} \tilde{R}_i\, |w_{ij}| \ ; \tag{6.179}$$

the bound is achieved when

$$w_i = \mathrm{SGN}\, (w_{ij})\, (2 - \alpha)\, \tilde{R}_i \ . \tag{6.180}$$

For the sufficient condition we have

$$h_j \leq (2-\alpha) \sum_{\substack{i=1 \\ i \neq j}}^{n} (\tilde{R}_i + R_i)\, |w_{ij}| \ ; \tag{6.181}$$

the bound is achieved when

$$w_i = \mathrm{SGN}\, (w_{ij})\, (2-\alpha)\, R_i \ . \tag{6.182}$$

The diagonal-dominance conditions follow from (6.157), (6.161), (6.164), and (6.168), combined with (6.179) for necessity or (6.181) for sufficiency. The resulting necessary condition is

$$w_{jj} \geq \frac{2-\alpha}{\alpha} \sum_{\substack{i=1 \\ i\neq j}}^{n} \frac{\tilde{R}_i}{\tilde{R}_j} |w_{ij}| \ . \tag{6.183}$$

The resulting sufficient condition is

$$w_{jj} \geq \frac{2-\alpha}{2\alpha} \sum_{\substack{i=1 \\ i\neq j}}^{n} \left(\frac{\tilde{R}_i + R_i}{\tilde{R}_j}\right) |w_{ij}| \tag{6.184}$$

Both (6.183) and (6.184) must hold for each j, j = 1, 2, ..., n, i.e., for each column (or row) of W. Note that the ratio of the i^{th} coefficients in (6.184) and (6.183) is $(1 + R_i/\tilde{R}_i)/2$. If we take $R_i = 2\tilde{R}_i$ (or less), which is typical in some realizations, then this ratio is 3/2. This is a tolerable increase over the necessary condition.

We now show that (6.122) will be bounded away from zero, i.e., (6.124) will hold, in the event of an overflow condition, if the dominance condition (6.184) is exceeded by some margin. It is sufficient to show that (6.153) and (6.156) will be slightly positive, since C_j in (6.154) is the product of (6.153) and (6.156). This will bound C_j away from zero, and hence bound (6.122) away from zero since (6.141), (6.140), and (6.122) are equivalent. In the event of an overflow condition at x_j, (6.153) will exceed zero by a fixed amount because $F_j(w_j(k), k)$ is actually granular in nature; i.e., w_j will exceed $\bar{w}_j$ by some number of quantization steps. The granularity of $F(w(k), k)$ follows from the granularity of the overflow characteristic θ. Replacing the zero in the right-hand side of (6.156) by a small positive constant gives this same constant in the right-hand side of (6.157), hence (6.184). Thus, (6.156)

exceeds zero by a fixed amount if (6.184) is exceeded by some margin.

We conclude this Section with remarks concerning the specialization of (6.183) and (6.184) in various directions. Clearly, a pure-diagonal matrix W satisfies (6.183) and (6.184) with some margin, for any allowable $\alpha \neq 0$. In this regard, consider, e.g., the wave digital filter. Secondly, note that some simplification of the expressions occurs in (6.183) and (6.184) for the equal-word-length case, where we take all $\tilde{R}_i$ as equal and all R_i as equal, $i = 1, 2, \ldots, n$. Finally, it is possible to specialize (6.183) and (6.184) to the zero-input case developed in Section 6.5. We do this by taking $\alpha = 1$, which constrains v_i to zero by (6.135) and (6.174); this imposes the zero-input condition. For this particular value of α, (6.183) and (6.184) become identical to (6.58) and (6.64), respectively, in Section 6.5.

6.14 Summary and Consequences in Terms of Vector Norm

The purpose of this Section is to summarize ASFR and deduce some of its consequences, in terms of the vector norm

$$||\bar{v}|| = (\bar{v}^T W \bar{v})^{1/2} . \tag{6.185}$$

Note that (6.185) satisfies the mathematical definition of a norm for a vector (see, e.g., [I1]); here we have used the vector $\bar{v}$ as a dummy variable. In Sections 6.9 through 6.13, we have established diagonal-dominance conditions on W which guarantee

$$\lim_{k \to \infty} ||e(k)||^2 = 0 . \tag{6.186}$$

Since all norms are equivalent in a finite dimensional space, by (6.186)

$$\lim_{k \to \infty} e(k) = 0 ; \tag{6.187}$$

this gives ASFR, from which we deduced stability of the forced response against overflow limit cycles. As an intermediate consequence of (6.186), we showed that the system (4.23) will eventually operate entirely within the non-overflow region.

We first apply the norm (6.185) to the asymptotic comparison of the response sequences $\hat{x}(k)$ in (4.23) and $s(k)$ in (3.1). We assume the same input sequence in (3.1) and (4.23), but a non-zero error in the initial state of (4.23). From (6.113), we take

$$||\hat{x}(k) - s(k)|| = ||r(k) + e(k)|| \tag{6.188}$$

by a simple re-arrangement. Applying the triangle inequality, we obtain

$$||\hat{x}(k) - s(k)|| \overset{<}{=} ||r(k)|| + ||e(k)|| \tag{6.189}$$

from (6.188). In view of (6.186), for an arbitrarily small $\varepsilon > 0$ we have, from (6.189),

$$||\hat{x}(k) - s(k)|| \overset{<}{=} ||r(k)|| + \varepsilon, \quad k \overset{>}{=} K, \tag{6.190}$$

for a sufficiently large K. This simply states that the essential difference between the "digital" response $\hat{x}(k)$ and the desired response $s(k)$ is the roundoff noise $r(k)$.

We now apply the norm (6.185) to the asymptotic comparison of possible response sequences in the digital system (4.23), assuming different initial state errors but the same input sequence. Here we mean different initial state errors in (4.23) relative to system (3.1), or simply different initial states in (4.23). For this comparison, we denote the different initial state errors by $e^1(o)$ and $e^2(o)$. Likewise, we denote the roundoff error, state error, roundoff noise, and state-vector sequences which correspond to these initial errors by $d^1(k)$, $e^1(k)$, $r^1(k)$, $\hat{x}^1(k)$, $k = 0, 1, 2, \ldots$, and $d^2(k)$, $e^2(k)$, $r^2(k)$, $\hat{x}^2(k)$, $k = 0, 1, 2, \ldots$, respectively. By (6.113), we have

$$\hat{x}^1(k) = s(k) + r^1(k) + e^1(k) \tag{6.191}$$

and

$$\hat{x}^2(k) = s(k) + r^2(k) + e^2(k) , \tag{6.192}$$

$k = 0, 1, 2, \ldots$. The difference

$$\hat{x}^1(k) - \hat{x}^2(k) = r^1(k) - r^2(k) + e^1(k) - e^2(k) \tag{6.193}$$

is dependent only on the roundoff noise and the state error sequences generated by the initial state errors, since $s(k)$ is the same for (6.191) and (6.192). Clearly, by (6.187), the difference (6.193) is essentially due to differences in roundoff noise as k becomes sufficiently large. Note that the roundoff-noise sequences $r^1(k)$ and $r^2(k)$, given by (6.114), will differ, because the roundoff-error sequences $d^1(k)$ and $d^2(k)$ will differ due to different initial state errors. In terms of norms, we have

$$||\hat{x}^1(k) - \hat{x}^2(k)|| \leq ||r^1(k) - r^2(k)|| + ||e^1(k) - e^2(k)|| \quad (6.194)$$

from (6.193) by the triangle inequality. Applying the triangle inequality also to the right-hand term in (6.194), we obtain

$$\lim_{k \to \infty} ||e^1(k) - e^2(k)|| = 0 \quad (6.195)$$

since (6.186) holds for both $e^1(k)$ and $e^2(k)$. In view of (6.195), for arbitrarily small $\varepsilon > 0$

$$||\hat{x}^1(k) - \hat{x}^2(k)|| \leq ||r^1(k) - r^2(k)|| + \varepsilon, \; k \geq K, \quad (6.196)$$

if K is chosen sufficiently large. This shows that the "closeness" of $\hat{x}^1(k)$ and $\hat{x}^2(k)$ is dependent on the difference in the roundoff noises, where we use the norm as the measure of distance.

6.15 Underflow Roundoff Noise

In this Section, we mention certain aspects of underflow roundoff noise which require further research. This is motivated by the fact that we have been able to separate the analyses of overflow and underflow effects by the preceeding work. Thus, it only remains to deal with underflow effects if stability of the forced response has been obtained by meeting diagonal-dominance conditions on W.

The analysis of underflow roundoff noise has received much attention in the literature. The generally accepted approach is a combination of probability theory and the principle of linear super-

position; see Section 5.2 for some remarks pertaining to this approach. The most questionable area of the analysis is the assumptions which must be made on the probability distributions of the components of the underflow error vector $d(k)$, $k = 0, 1, 2, \ldots$. It is common to assume uniform, uncorrelated probability distributions in $d(k)$. This assumption is most likely to be valid for input sequences which are sufficiently dynamic, i.e., rapidly changing and uncorrelated. On the other hand, the assumption may not be valid for certain types of input sequences (e.g., periodic sequences) or for underflow limit cycles. The probability distribution of roundoff noise, $r(k)$, $k = 0, 1, 2, \ldots$, defined in (6.114), follows from the probability distribution of $d(k)$, $k = 0, 1, 2, \ldots$, by linear superposition.

Another aspect of the underflow-roundoff-noise problem arises from the asymptotic comparison of response sequences in the digital system (4.23). This comparison is described in Section 6.14, where we assume different initial state errors $e^1(o)$ and $e^2(o)$ but the same input sequence. Here it is natural to ask how small $||r^1(k) - r^2(k)||$ can be. If $||r^1(k) - r^2(k)||$ is sufficiently small for some k, say k_o, then we must conclude by (6.196) that $\hat{x}^1(k_o) = \hat{x}^2(k_o)$, because $\hat{x}^1(k)$ and $\hat{x}^2(k)$ are quantized variables. From this, we would have $\hat{x}^1(k) = \hat{x}^2(k)$, $k \geq k_o$, by the state-space concept; it would then follow that $d^1(k) = d^2(k)$, $k \geq k_o$. In summary, we are asking if the asymptotic part of the underflow error sequence $d(k)$ is unique. There does not appear to be any simple reason to assume that it is unique in every case. In

fact, we may even entertain the possibility of distinct underflow limit cycles.

The difference $r^1(k) - r^2(k)$ must actually assume discrete values dictated by the possible quantized values of $\hat{x}^1(k)$ and $\hat{x}^2(k)$ in (6.193), if k is sufficiently large for $e^1(k) - e^2(k)$ to be negligible. This indicates a cross-correlation between the sequences $r^1(k)$ and $r^2(k)$. Thus, it is well to regard with caution any simplifying assumptions on the probability distributions of $d^1(k)$ and $d^2(k)$, $k = 0, 1, 2, \ldots$.

We now show that $r^1(k)$ and $r^2(k)$ will converge to the same sequence, if $d^1(k) = d^2(k)$ for $k \geq k_0$, i.e., if $d^1(k)$ and $d^2(k)$ differ only in their leading parts. This is intuitively clear, but we show it more rigorously by the use of the norm as follows. By (6.114), we have

$$r^1(k) = \sum_{i=0}^{k} A^{k-i} d^1(i) \tag{6.197}$$

and

$$r^2(k) = \sum_{i=0}^{k} A^{k-i} d^2(i) , \tag{6.198}$$

which gives

$$r^1(k) - r^2(k) = \sum_{i=0}^{k} A^{k-i} (d^1(i) - d^2(i)) . \tag{6.199}$$

Since

$$d^1(i) - d^2(i) = 0, \; i \geq k_0 , \tag{6.200}$$

we obtain from (6.199)

$$r^1(k) - r^2(k) = \sum_{i=0}^{k_o} A^{k-i}\,(d^1(i) - d^2(i))$$

$$= A^{k-k_o} \sum_{i=0}^{k_o} A^{k_o-i}\,(d^1(i) - d^2(i))$$

$$= A^{k-k_o}\,(r^1(k_o) - r^2(k_o)) \qquad (6.201)$$

For simplicity of notation, we define

$$b(k) \triangleq r^1(k) - r^2(k) \qquad (6.202)$$

and note that

$$b(k) = A^{k-k_o}\,b(k_o)\;,\quad k \triangleq k_o\;. \qquad (6.203)$$

We now show that

$$||b(k)||^2 = b^T(k)\,W\,b(k) \qquad (6.204)$$

decreases to zero. Since

$$||b(k+1)||^2 = b^T(k)\,A^T W A\,b(k)\;, \qquad (6.205)$$

we have

$$||b(k)||^2 - ||b(k+1)||^2 = b^T(k)[W - A^T W A]\,b(k)$$

$$= b^T(k)\,C^T C\,b(k) \qquad (6.206)$$

from (6.204), (6.205), and the Lyapunov matrix equation (3.11). It follows by Lyapunov theory [F13] that the system (6.203) is

asymptotically stable for observable sequences b(k); that is,

$$\lim_{k \to \infty} ||b(k)||^2 = 0 \quad , \tag{6.207}$$

which implies

$$\lim_{k \to \infty} b_i(k) = 0 \quad , \; i = 1, 2, \ldots, n. \tag{6.208}$$

If the system (3.1) is completely observable, (6.208) will hold for all sequences b (k), $k = k_o, k_o + 1, k_o + 2, \ldots$. This completes the demonstration that r^1 (k) and r^2 (k) will converge.

6.16 Application of Diagonal-Dominance Conditions

Necessary diagonal-dominance conditions and sufficient diagonal-dominance conditions on the matrix W have been given for each of the following: zero-input underflow stability; zero-input overflow stability; and stability of the forced response. The necessary conditions appear in (6.94), (6.57), and (6.183), respectively; the sufficient conditions appear in (6.98), (6.63), and (6.184), respectively. We apply these conditions to specific realizations by checking if W, the solution of the Lyapunov matrix equation, meets the desired diagonal dominance condition, after coefficient quantization in A and C. That is, we begin with a linear prototype realization (3.1) where W is purely diagonal; e.g., minimum= roundoff-noise (MRN) realizations and state-space realizations of wave digital filters. W will become only approximately diagonal after the coefficients of the realization are quantized. If W does

not then meet the desired diagonal-dominance conditions, we correct this situation by improving the accuracy of coefficient quantization. It is always possible to meet the dominance conditions by sufficiently accurate coefficient quantization, because the elements of W are continuous functions of the coefficients in A and C.

7. SENSITIVITY AND COEFFICIENT QUANTIZATION

7.1 Introduction

Coefficient quantization is unavoidable in passing from a discrete-time state-space realization to the corresponding digital realization, as discussed in Sections 2.4 and 2.5. However, it is desirable to restrict the variation in frequency response and to retain the favorable properties of the MRN realization after coefficient quantization, as explained in Sections 4.3 and 4.7 These favorable properties (limit-cycle suppression and low roundoff noise) were related to diagonal dominance in K and W, in Chapters 5 and 6. Therefore, coefficient quantization should be performed under a simultaneous restriction on the degradations in frequency response and diagonal dominance.

In this Chapter, a method of coefficient quantization is developed which is generally applicable to state-space realizations of any order and well suited to the above-mentioned problem. The central idea is to restrict the variation of the system eigenvalues and to compensate for reasonable variations in eigenvectors by adjusting the magnitude of fundamental modes which appear at the output. This adjustment of fundamental modes is used to restore the original zeroes of the transfer function. The method is effective because the eigenanalysis of the state-space realization completely determines its frequency response and the solutions K and W to the Lyapunov equations, as shown in Sections 7.3 through 7.5.

The method developed here leads to the formulation of a constrained optimization problem. The first set of constraints follows from

the restrictions on the variation in eigenvalues, as explained in Section 7.6. Another constraint is a set of "box bounds" which encloses the infinite precision "center point" provided by the original realization, for reasons given in Section 7.7. Since quantization is equivalent to a set of integer constraints, the problem specializes to an integer programming problem. The objective function to be minimized (or at least reduced) is the cost of the digital realization (e.g., the total bit-adder product). However, this "objective" is interpreted rather loosely in Section 7.7, to expedite matters.

The above-mentioned integer programming problem is formidable; it was approached by means of a branch-and-bound algorithm, after an extensive survey of integer programming methods; the survey is summarized in Section 7.8. A detailed study of the branch-and-bound algorithm, summarized in Section 7.9, leads to an appropriate specialization to fit the optimization method of Section 7.7. The specialization is described in Section 7.10; it was programmed and successfully applied, as described in Chapter 9.

It is appropriate to mention other attempts at the coefficient quantization problem before proceeding. Many techniques have appeared in the literature, which are usually restricted to second-order sections or appear to use integer programming methods less powerful than the branch-and-bound algorithm. Many combinations of constraints and objective functions have been used. The reader may consult papers by Suk and Mitra [J2], Avenhaus [J1], Brglez [J5], Wegener [J6], Bandler et al. [J3], and Charalambous and Best [J4] for information on previous optimi-

zation techniques. An integer programming problem is inherent in any optimization technique. The first four papers mentioned use random search, modified univariate search, sequential search, and pattern search, respectively, to approach the integer programming problem. Bandler et al. [J3] and Charalambous and Best [J4] have used the branch-and-bound algorithm to approach the integer programming problem; their work involves a very specific formulation for second-order sections. However, it appears that no other work on coefficient optimization using the branch-and-bound algorithm has been published.

The computational procedures which result from the analysis in this Chapter are programmed and verified in Chapter 9.

7.2 Sensitivity

The sensitivity of a system to parameter variations is a significant question in many situations [E2-E4]. However, there is no single approach to sensitivity analysis which is appropriate to every situation. Rather, the accepted philosophy is to define an error criterion that appears most suitable to the problem at hand, i.e., a criterion that "works".

The two major classifications are sensitivity of the time-domain trajectory and sensitivity of the frequency-domain response [E4]. Trajectory sensitivity is generally dependent on the particular input, which is a disadvantage for most analysis. However, in the special case of a unit-sample input, trajectory sensitivity is directly related to frequency-domain sensitivity by the Fourier transform. Therefore, a constraint on the variation in frequency response is roughly equivalent

to a constraint on the variation in the unit-sample response. (In all discussions of coefficient sensitivity, it is assumed that signals are represented by infinite precision.) There are numerous ways to obtain a constraint on the frequency response. For example, we may impose bounds on: the variation in response at several discrete points in the frequency domain; the integral-square error in the frequency domain; the variation in each component of the unit-sample response; or the sum of squared errors in the unit-sample response. Ultimately, we choose a bound on the variation in each eigenvalue because it is analytically tractable, very fundamental, and provides the desired constraints on frequency response and K and W.

The following interpretation of any set of constraints will be very helpful in the sequel. Consider the coefficient hyperspace obtained by enumerating all system coefficients (conventionally row by row) into a vector; e.g., if we consider only the coefficients of the A matrix, then we obtain a vector space of dimension N^2. The vector obtained in this manner from the unquantized discrete-time state-space realization will be called the (infinite-precision) center point in coefficient space. If we define suitable error criteria for system performance, then imposing bounds on the magnitude variations in these error criteria is equivalent to defining a "constraint region" R in the hyperspace. That is, any <u>acceptable</u> vector of <u>continuous</u> coefficients must lie within R. R contains the center point, where all errors are zero, and it is a continuous (connected) region lying around the center point if the error criteria are continuous functions. It will be bounded if a sufficient number of

error criteria are used. Finally, the quantization of coefficients introduces an integer constraint. The problem then specializes to finding one or more all-integer points embedded in the region R originally defined on continuous parameter space. (Here all-integer point means a point such that all components are integers.) We may also view the problem in the reverse context if we recognize that the number system is constructed by imbedding rational numbers among integers and then irrational cuts among the rationals, to obtain the continuum.

The constraint region R is a useful abstract concept which must be given a more concrete meaning. This may be done by representing each error function by a multivariable Taylor series [J7], which has the general form:

$$\Delta f = g \cdot \Delta\hat{v} + \Delta\hat{v}^T H \Delta\hat{v} + \ldots , \qquad (7.1)$$

where g is the gradient vector, H is the Hessian matrix of (mixed) second-partial derivatives, and $\Delta\hat{v}$ is the displacement vector in coefficient space. Bounding each function of the form (7.1) defines a region in coefficient space; the intersection of all such regions yields R. However, it is totally impractical to use more than a few terms from the Taylor series (7.1), especially due to the potentially large number of variables in $\Delta\hat{v}$. Nevertheless, the Taylor series provides a firm theoretical justification for using a first- or second-order approximation to a multivariable function; this is often done in gradient-derived continuous optimization algorithms, where quadratic forms play an important role [J7]. Analysis by the Taylor series also reveals the following significant feature of the constrained optimization problem.

Namely, if a point P lies within R, it is not necessarily true that another point closer to the center point than P will also lie in R. This follows from the observation of first- and second-order terms whose magnitude is dependent on "direction", as well as magnitude, of displacement $\Delta\hat{v}$ in coefficient space. In particular, $\Delta\hat{v}$ may be orthogonal to the gradient vector or it may lie parallel to the eigenvector of H which has a corresponding eigenvalue of minimum magnitude. In practical terms, this means that the "best" quantized point is not necessarily the point nearest to the center point. This nearest point would be obtained simply by rounding each coefficient from its infinite-precision value. The constraints actually used in this Chapter will be first order (i.e., linear); this choice is made because efficient optimization algorithms exist to solve the corresponding linear continuous subproblems which occur in the branch-and-bound algorithm.

Low sensitivity to coefficient quantization is a property which is investigated after a given type of realization has been synthesized. There is no known method of explicitly controlling sensitivity during the synthesis process; indeed, natural tradeoffs which prevent this may exist. This "existential" approach to sensitivity means that it must be investigated by generating numerical examples for various realizations, such as the wave digital and MRN realizations. In general terms, low-sensitivity realizations will allow a greater volume in the constraint region R, thus increasing the likelihood that simple (i.e., low word length) all-integer points will lie within R.

The possible strategies for allocation of bits (i.e., precision) to each coefficient are numerous and worthy of a separate investigation. However, it has been empirically observed in computer examples that the assumption of nearly uniform relative sensitivity works well. That is, rather than truncate each coefficient to the same absolute precision, we retain a constant number of bits counting to the right from the most significant bit in each coefficient. This yields a span of active (i.e., potentially non-zero) bits which is the same for all coefficients, although the position of the most signficiant bit may vary. Truncation to uniform absolute precision was observed to produce generally poorer results than the method of uniform relative precision.

7.3 Unit-Sample Response

The unit-sample sequence is

$$\delta(k) \triangleq \begin{cases} 1 & , \quad k = 0 \\ 0 & , \quad k \neq 0 \end{cases} \tag{7.2}$$

For the unit-sample input

$$u(k) = \delta(k) \quad , \tag{7.3}$$

we have the unit-sample response $h(k) = y(k)$ in the discrete-time state-space realization (3.1). It is clear from (3.1) that

$$h(k) = \begin{cases} C\,A^{k-1}\,B & ; \quad k \geq 1 \\ D & ; \quad k = 0. \end{cases} \tag{7.4}$$

The eigenanalysis of matrix A can be used to display the modes in $h(k)$. We assume

$$AT = T\tilde{D} \quad , \tag{7.5}$$

where

$$\tilde{D} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} \tag{7.5a}$$

is the diagnonal matrix of eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and T is the modal matrix (i.e., the columns of T are eigenvectors of A). We assume the simpler case where the eigenvectors of A are linearly independent; i.e., T^{-1} exists; otherwise the Jordan canonical form is required. A sufficient condition for linearly independent eigenvectors is distinct eigenvalues; distinct eigenvalues will occur in nearly all cases encountered in filter design, because nearly all transfer functions of interest have distinct poles. In particular, if we begin with distinct poles in the s-domain, the bilinear transformation yields distinct poles in the z-domain. From (7.5),

$$A = T\tilde{D}\,T^{-1} \quad , \tag{7.6}$$

which gives

$$A^k = T\tilde{D}^k\,T^{-1} \quad . \tag{7.7}$$

By (7.4) and (7.7),

$$h(k) = \begin{cases} CT\tilde{D}^{k-1}\,T^{-1}\,B & ; \quad k \geq 1 \\ D & ; \quad k = 0 \, . \end{cases} \tag{7.8}$$

We view this system as a reduced system by defining the row vector

$$\tilde{C} \triangleq C\,T \quad , \tag{7.9}$$

with components $\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n$, and the column vector

$$\tilde{B} \triangleq T^{-1} B \tag{7.10}$$

with components $\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_n$. Thus, by (7.8), (7.9), and (7.10),

$$h(k) = \begin{cases} \tilde{b}_1\tilde{c}_1\tilde{\lambda}_1^{k-1} + \tilde{b}_2\tilde{c}_2\tilde{\lambda}_2^{k-1} + \ldots + \tilde{b}_n\tilde{c}_n\tilde{\lambda}_n^{k-1}; & k \geq 1; \\ D \quad , & k = 0 \, . \end{cases} \tag{7.11}$$

A strategy for quantizing A, B, C, and D can be developed by examining (7.11), (7.10), and (7.9). From (7.11), it is seen that the eigenvalues of A assume primary importance in h(k). Therefore, we choose to quantize A first, with the intention of minimizing the variation in eigenvalues and accepting any reasonable variation in eigenvectors. After this, it is possible to adjust the coefficients $\tilde{b}_i\ \tilde{c}_i$, $i = 1, \ldots, n$, to compensate for variations in eigenvectors (i.e., variations in T). This can be done by specifying the desired $\tilde{b}_i, \tilde{c}_i$, $i = 1, \ldots, n$, i.e., the desired $\tilde{B}$, $\tilde{C}$, and computing a <u>new</u> B and C from (7.9) and (7.10), respectively, as follows:

$$C = T^{-1}\tilde{C} \, ; \tag{7.12}$$

$$B = T\tilde{B} \, . \tag{7.13}$$

With regard to the frequency response, a very effective criterion for

adjusting $\tilde{b}_i$, $\tilde{c}_i$, $i = 1, \ldots, n$, is the restoration of the zeroes of the transfer functions to their original positions; other criteria may be appropriate in other situations. The adjustment procedure is formulated in Section 7.4. After the new (unquantized) B and C have been computed, the final step is to quantize B, C, and D, with the intention of minimizing the variation in the newly restored zeroes.

7.4 Frequency-Domain Analysis

We now relate the transfer-function matrix (3.3) to the reduced form (7.11) of the unit-sample response; a single-input-single-output system is assumed for simplicity. Substituting (7.6) into (3.3), we have

$$H(z) = C(T[zI-\tilde{D}]T^{-1})^{-1} B + D \quad , \tag{7.14}$$

which gives

$$H(z) = CT\,(zI-\tilde{D})^{-1}\,T^{-1} B + D \quad . \tag{7.15}$$

Recognizing the inverse of the diagonal matrix $zI-\tilde{D}$ and substituting (7.9) and (7.10), we arrive at

$$H(z) = \tilde{C}\begin{bmatrix} \frac{1}{z-\lambda_1} & & 0 \\ & \frac{1}{z-\lambda_2} & \\ & & \ddots \\ 0 & & \frac{1}{z-\lambda_n} \end{bmatrix} \tilde{B} + D \tag{7.16}$$

or

$$H(z) = \frac{\tilde{b}_1 \tilde{c}_1}{z-\lambda_1} + \frac{\tilde{b}_2 \tilde{c}_2}{z-\lambda_2} + \ldots + \frac{\tilde{b}_n \tilde{c}_n}{z-\lambda_n} + D \quad . \tag{7.17}$$

It should be emphasized that $\tilde{b}_i$, $\tilde{c}_i$ are the same in (7.11) and (7.17); the result (7.17) can also be obtained by applying the z-transform directly to (7.11) and using the relation for the sum of a geometric series. The expression (7.17) is the partial-fraction expansion of (2.1).

The partial-fraction expansion (7.17) is most useful for analyzing or formulating the effects of variation of the eigenvalues and for computing $\tilde{b}_i$, $\tilde{c}_i$, $i = 1, \ldots, n$, to adjust the location of zeroes. We may formulate the variation in $H(e^{j\omega})$ at several discrete frequencies ω_i by setting $z = e^{j\omega_i}$ and taking partial derivatives with respect to λ_i, $i = 1, \ldots, n$; many refinements and variations on this method are possible. However, in Chapter 9 bounds are chosen on the variation in eigenvalues simply by examining computer-generated data for the frequency response and corresponding eigenvalue shifts.

The adjustment of $\tilde{b}_i$ $\tilde{c}_i$, $i = 1, \ldots, n$, to obtain specified zeroes proceeds as follows. Assume that $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_n$ are the eigenvalues of A after quantization, that $z_1, z_2, \ldots, z_n$ are the specified zeroes and that $H(e^{j0})$ is the desired zero-frequency gain. Then we require

$$H(z) = \frac{K_a(z-z_1)(z-z_2)\ \ldots\ (z-z_n)}{(z-\hat{\lambda}_1)\ (z-\hat{\lambda}_2)\ \ldots\ (z-\hat{\lambda}_n)} = \frac{\hat{K}_1}{z-\hat{\lambda}_1} + \frac{\hat{K}_2}{z-\hat{\lambda}_2} + \ldots + \frac{\hat{K}_n}{z-\hat{\lambda}_n} + D, \tag{7.18}$$

where K_a is the scale factor and we have used the notation

$$\hat{K}_i \triangleq \hat{b}_i \hat{c}_i \ , \ i = 1, \ldots, n \ , \tag{7.19}$$

for simplicity. Here $\hat{b}_i$ and $\hat{c}_i$ are the new values which will replace $\tilde{b}_i$ and $\tilde{c}_i$, respectively. It follows (by long division) that

$$D = K_a \ . \tag{7.20}$$

The first step is to determine K_a, hence D, by evaluating the left-hand expression in (7.18) at $z = e^{j\omega} = 1$; this gives

$$K_a = \frac{(1-\hat{\lambda}_1)\ (1-\hat{\lambda}_2) \ \ldots \ (1-\hat{\lambda}_n)\ H\ (e^{j\omega})}{(1-z_1)\ (1-z_2) \ \ldots \ (1-z_n)} \ . \tag{7.21}$$

The desired coefficients $\hat{K}_i$ follow by the well-known classical method:

$$\hat{K}_i = \left[\frac{(z-\hat{\lambda}_i)\ K_a\ (z-z_1)\ (z-z_2) \ \ldots \ (z-z_n)}{(z-\hat{\lambda}_1)\ (z-\hat{\lambda}_2) \ \ldots \ (z-\hat{\lambda}_n)} \right]_{z=\lambda_i}, \ i=1,\ldots,n. \tag{7.22}$$

In (7.22), H(z) consists of all but the leading factor $(z - \hat{\lambda}_i)$, where $(z - \hat{\lambda}_i)$ will cancel the same term in the denominator. Many, if not all, of the zeroes $z_1, z_2, \ldots, z_n$ will be at -1, because the bilinear transformation maps each zero at infinity in the s-domain to -1 in the z-domain.

It remains to determine the (unquantized) $\hat{b}_i$ and $\hat{c}_i$ from (7.19); clearly, some latitude exists here because (7.19) specifies only the product $\hat{b}_i\ \hat{c}_i$. In order to avoid excessive variation in the response sequences f_i and g_i, $i = 1, \ldots, n$, which determine K and W, it is desirable that $\hat{b}_i$ be near to $\tilde{b}_i$ and that $\hat{c}_i$ be near to $\tilde{c}_i$. Assuming that

$$K_i \triangleq \tilde{b}_i \tilde{c}_i \quad , \quad i = 1, 2, \ldots, n \quad , \tag{7.23}$$

represents the coefficients of the original partial fraction expansion (7.17) after coefficient quantization in A, we may proceed as follows. Determine the ratio

$$\hat{r}_i = \frac{\hat{K}_i}{K_i} \quad , \quad i = 1, \ldots, n \quad , \tag{7.24}$$

a complex number, which indicates the relative change in K_i. Distribute the relative change between $\hat{b}_i$ and $\hat{c}_i$ by taking the geometric mean $\sqrt{\hat{r}_i}$ between 1 and $\hat{r}_i$; i.e., let

$$\hat{b}_i = \sqrt{\hat{r}_i} \; \tilde{b}_i \tag{7.25}$$

and

$$\hat{c}_i = \sqrt{\hat{r}_i} \; \tilde{c}_i \quad , \tag{7.26}$$

which clearly satisfy (7.19). If we define the diagonal matrix

$$\hat{r} = \begin{bmatrix} \hat{r}_1 & & & \\ & \hat{r}_2 & & 0 \\ & & \ddots & \\ 0 & & & \hat{r}_n \end{bmatrix} \tag{7.27}$$

then we may write

$$\hat{C} \, T = C \, T \, \hat{r}^{1/2} \tag{7.28}$$

and

$$T^{-1} \hat{B} = \hat{r}^{1/2} \, T^{-1} \, B \quad , \tag{7.29}$$

where T is the modal matrix of the quantized A matrix. Simplifying (7.28) and (7.29), we obtain the formulas

$$\hat{C} = C M \tag{7.30}$$

and

$$\hat{B} = M B \ , \tag{7.31}$$

which determine the "new" C and B, where

$$M = T \hat{r}^{1/2} T^{-1} \ . \tag{7.32}$$

The quantization of $\hat{b}_i$, $\hat{c}_i$, i = 1, ..., and D should be performed with the intention of minimizing the variation in the restored zeroes and in the zero-frequency gain. The method used in Chapter 9 is simply to provide a sufficient word length for these quantities. However, it is worth mentioning that the specialized branch-and-bound algorithm can also be applied here because the problem fits the framework developed for the eigenvalue problem. It is only necessary to derive an appropriate sensitivity matrix with respect to $\hat{b}_i$, $\hat{c}_i$, i = 1, ..., n, and D, in order to define a constraint region. Such a sensitivity matrix can be found by chain differentiation: first, we determine the coefficients of the numerator of H(z), which will be linear combinations of $\hat{K}_i$, i = 1, ..., n; then we apply available results for the sensitivity of the roots of a polynomial to variations in its coefficients.

7.5 Lyapunov Equations

The solutions W and K to the Lyapunov matrix equations (3.11) and (3.13), respectively, are closely related to the unit-sample response

h(k) of the realization (3.1) and to the eigenanalysis of the matrix A. The response sequences f_i and g_i, i = 1, ..., n, defined in Section 5.4, determine the elements of K and W, respectively, as shown in (5.6) and (5.7). It follows from the definitions of these response sequences that

$$h(k) = C\, f(k) \quad , \quad k \geq 1, \tag{7.33}$$

and

$$h(k) = g(k - 1)\, B, \quad k \geq 1 \quad ; \tag{7.34}$$

where f and g are the vectors:

$$\underset{\sim}{f}(k) = \begin{bmatrix} f_1(k) \\ f_2(k) \\ \vdots \\ f_n(k) \end{bmatrix} \quad ; \quad g(k) = \begin{bmatrix} g_1(k) \\ g_2(k) \\ \vdots \\ g_n(k) \end{bmatrix} \quad . \tag{7.35}$$

We may also arrive at (7.33) and (7.34) by using (7.4) and the explicit formulations

$$f(k) = A^{k-1} B \tag{7.36}$$

and

$$g(k) = CA^k \quad , \tag{7.37}$$

which were tacitly assumed in (5.4) through (5.7). It follows from (7.4) that for a fixed B, C, and D, we cannot obtain the same h(k), k = 0, 1, 2, ..., with two different A matrices; i.e., A must be unique.

By (7.36) and (7.37), f(k) and g(k) are also unique. If the eigenvectors and eigenvalues of A vary little during quantization, then h(k) varies little by (7.11) and f(k) and g(k) vary little by (7.36) and (7.37). A small variation in B and C during the second stage of quantization will also give a small variation in f(k) and g(k), by (7.36) and (7.37). Thus, it follows by (5.6) and (5.7) that the variation in K and W will be small.

We now derive an explicit formulation of K and W in terms of the eigenvalues and eigenvectors of A and the vectors B and C. This formulation confirms the above assertion that the variation in K and W is closely related to the variation in the frequency response. The first step is to substitute the decomposition (7.6) into (5.2) and (5.3):

$$K - (T\hat{D}T^{-1})\, K\, ([T^{-1}]^T \tilde{D}^T T^T) = BB^T \quad ; \tag{7.38}$$

$$W - (T^{-1})^T \tilde{D}^T T^T W T \tilde{D} T^{-1} = C^T C \quad . \tag{7.39}$$

Pre-multiplying by T^{-1} and post-multiplying by $(T^T)^{-1}$ in (7.38) gives

$$\hat{K} - \tilde{D} \hat{K} \tilde{D} = \tilde{B} \tilde{B}^T \quad , \tag{7.40}$$

where we define,

$$\hat{K} \triangleq T^{-1} K (T^{-1})^T \tag{7.41}$$

for simplicity. Reversing the pre-/post-multiplication procedure in (7.39) gives

$$\hat{W} - \tilde{D} \hat{W} \tilde{D} = \tilde{C}^T \tilde{C} \quad , \tag{7.42}$$

where we define

$$\hat{W} \overset{\Delta}{=} T^T W T \quad . \tag{7.43}$$

On an element-by-element basis in (7.40) and (7.42), we obtain

$$\hat{k}_{ij} = \frac{\tilde{b}_i \tilde{b}_j}{1 - \lambda_i \lambda_j} \quad , \quad i, j = 1, \ldots, n \quad , \tag{7.44}$$

and

$$\hat{w}_{ij} = \frac{\tilde{c}_i \tilde{c}_j}{1 - \lambda_i \lambda_j} \quad , \quad i, j = 1, \ldots, n \quad , \tag{7.45}$$

respectively, due to the simple operations implied by the diagonal matrices $\tilde{D}$. K and W can be obtained from (7.44) and (7.45), respectively, by reversing the transformations (7.41) and (7.43):

$$K = T \hat{K} T^T \quad ; \tag{7.46}$$

$$W = (T^{-1})^T \hat{W} T^{-1} \quad . \tag{7.47}$$

7.6 Eigenvalue Variations

A compact formulation for the first-order variation $\delta\lambda_j$ in the eigenvalues λ_j, $j = 1, \ldots, n$, of a matrix A is derived in [I2]. For a real matrix A, it specializes to

$$\delta\lambda_j = \frac{((\delta A)\, u^j \, , \, t^j)}{(u^j \, , \, t^j)} \quad , \quad j = 1, \ldots, n \quad ; \tag{7.48}$$

where δA is the variation in A, u^j is the eigenvector of A corresponding to λ_j, t^j is the eigenvector of A^T corresponding to λ_j, and $(\, , \,)$ indicates the inner product of the vectors separated by the comma. The relation (7.48) was computer-tested on several examples and found to be quite accurate for variations as large as five to ten percent in the

real and imaginary parts of eigenvalues.

It is necessary to convert (7.48) to the standard form of a gradient (row) vector g^j, to provide useable constraints in the coefficient hyperspace. That is, we require

$$\delta\lambda_j = g^j \cdot \Delta \hat{v} \tag{7.49}$$

where the vector $\Delta\hat{v}$ contains the elements of δA enumerated row by row. The necessary form is simply obtained by grouping the coefficients which multiply each element $\delta A_{\ell k}$, ℓ, $k = 1, \ldots, n$, of A in (7.48). In this regard, it follows from (7.48) that

$$\delta\lambda_j = \frac{(\bar{u}^j)^T \, \delta A^T \, t^j}{(u^j \, , \, t^j)} = \sum_{k,\ell=1}^{n} \frac{(\bar{u}^j)_K \, (t^j)_\ell \, \delta A_{\ell k}}{(u^j \, , \, t^j)} \quad , \tag{7.50}$$

$j = 1, \ldots, n$. If we define

$$S^j_{\ell k} \overset{\Delta}{=} \frac{(\bar{u}^j)_k \, (t^j)_\ell}{(u^j \, , \, t^j)} \quad , \tag{7.51}$$

the sensitivity coefficient which multiplies $\delta A_{\ell k}$, then the matrix of these coefficients is

$$S^j = \left[s^j_{\ell k} \right] = \frac{t^j \, (\bar{u}^j)^T}{(u^j \, , \, t^j)} . \tag{7.52}$$

The matrix S^j, which is amenable to computer computation, should be enumerated row by row into g^j. Then we obtain

$$\delta\lambda = S\Delta\hat{v} \tag{7.53}$$

where row j of the overall sensitivity matrix S is g^j, $j = 1$,

..., n, and

$$\delta\lambda = \begin{bmatrix} \delta\lambda_1 \\ \delta\lambda_2 \\ \vdots \\ \delta\lambda_n \end{bmatrix} . \qquad (7.54)$$

Finally, for each pair of rows in S corresponding to complex-conjugate eigenvalues, the first (second) row of the pair is replaced by its real (imaginary) part. This replaces the variations in λ by the separate real- and imaginary-part variations, respectively, which is necessary to accomodate the real-arithmetic requirement of the continuous linear optimization program in the branch-and-bound algorithm. However, no information is lost from S, because the replaced rows are complex conjugates.

7.7 Optimization Method

A general optimization method will now be formulated, based on the previous background material and the limitations of mathematical programming algorithms. The object is to find an all-integer point within the constraint region R; if such a point exists, it is said to be feasible. A feasible point is necessarily close to the center point only in the sense that it lies within R. The integer constraint leads to an integer programming problem, which is a very difficult class of mathematical programming problem. It has been said that integer programming problems are an order of magnitude more difficult than continuous problems. The main source of difficulty is the discrete

nature of the problem, which rules out the direct application of the well-known gradient methods of continuous optimizations. Although the mathematical programming literature is extensive, only a few basic methods are known for the solution of integer programming problems; these methods are surveyed in Section 7.8.

A key feature of the present method is the introduction of a box search region (hypercube) enclosing the center point and having integer points as its corners. At least some portion of R will lie within the box, since R also includes the center point. The box will be searched for feasible all-integer points. The introduction of the box-like search region stems from several practical considerations. The foremost consideration is the need to reduce the number of worthwhile search possibilities. The knowledge of the center point, combined with the box surrounding it, has this effect. Also, the constraint region R determined by setting bounds on the eigenvalue variations (7.53) is not bounded in every direction. This means that other unconstrained characteristics, such as eigenvectors, may vary excessively without some localization of the search region around the center point. Although there may be points in coefficient space very distant from the center point which give the same frequency response (e.g., by a similarity transformation), the region around the center point is the only locality where all other characteristics (e.g., K and W) will meet the necessary requirements.

The importance of reducing the number of search possibilities can be seen immediately from a medium-size example. Consider a 5 x 5

matrix where each coefficient is assigned 4 bits. Without localizing the search region, there are 2^{100} possibilities due to the 100 zero-one variables (bits). Exhaustive explicit enumeration, i.e., the listing and comparison of all possibilities, at the rate of 100 million per second would require 4×10^{12} centuries! Since exhaustive explicit enumeration is clearly impossible, we must resort to various forms of implicit enumeration, i.e., the elimination or "fathoming" of certain subsets of possibilities by strictly logical arguments. The first step is the implicit enumeration of possibilities which are very unlikely to meet all system performance requirements, by localizing the search to the box region. The next step is the implicit enumeration of the remaining very large number of possibilities within the box, because exhaustive explicit enumeration may still be impossible. This will be done by means of a specialized integer programming algorithm described in Sections 7.9 and 7.10.

A corner of the box-like search region is determined by (computerized) truncation of the canonical-signed-digit-code (CSDC) representing each coefficient of the A matrix. These truncations are all done at a uniform bit span, in accordance with the idea of uniform relative sensitivity discussed in the last paragraph of Section 7.2. It is interesting to note that truncation of the CSDC automatically produces the closest number to an infinite precision coefficient for a given bit span; this may be helpful in reducing system performance errors associated with truncation. The domain of each coefficient within the box lies between its truncated value and its truncated value plus a

change of positive or negative one in its least significant bit (LSB). The sign of the change in each coefficient is chosen so that the infinite precision value of the coefficient lies within the resulting domain. The corners of the box correspond to all possible combinations of the extremes of the domains.

The bit-span used to determine the initial corner of the box is chosen to be a few bits less than adequate to meet the eigenvalue constraints, in order to reduce all initial coefficient word lengths. A (computerized) check on eigenvalue variations aids in finding such a value for the bit span. The bit span also determines the length of each side of the box, since it (temporarily) fixes the LSB of each coefficient. The intention is to avoid a premature assignment of an excessive number of bits, where the addition of a few bits to a limited number of coefficients may be adequate to meet the eigenvalue constraints. The box is then searched by uniformly appending a specified number of less significant bits to each truncated coefficient, with a sign which assures that each coefficient variation falls within its previously defined domain. The effect of these appended bits is to subdivide the box by a discrete grid. Clearly, only the appended bits may be varied during a search which remains within the box. The corner of the box determined by the original truncations plays the role of a starting point in the search. Since the appended bits may be subtracted from a coefficient in some cases, it is possible that some of the more significant bits in the starting point may vary.

It is advantageous here to assign secondary importance to strictly defining an objective function of the type usually associated with an optimization problem. The primary goal of meeting the eigenvalue constraints is assured if any feasible all-integer point is found. In addition, the limitations on the bit spans of both the starting-point coefficients and the appended bits during the box search places a limit on the cost of the hardware realization. A cost definition which fits the role of objective function is the sum of nonzero bits in the conventional binary representation of all coefficients. However, for small bit spans, the use of the CSDC guarantees a relatively small number of non-zero bits, which diminishes the importance of such a cost function. It turns out that this cost function is very useful as a projection criterion for partial solutions during the integer programming algorithm, as described in Section 7.10.

7.8 Survey of Integer Programming

All known methods of attacking integer programming problems amount to either explicit or implicit enumeration. For example, random search, pattern search, and univariate search employ explicit enumeration. Since exhaustive explicit enumeration is impossible in large problems, the above-mentioned methods terminate the search when the solution found appears to be close to optimal. Here we will consider only the more efficient method of implicit enumeration.

Implicit enumeration may be successful in completely fathoming a problem where exhaustive explicit enumeration is out of the question. Fathoming may involve appeals to violated constraints

(infeasibility), known solutions which must be more optimal, etc., which can be very effective. For example, if one fixes a single zero-one variable in the example in Section 7.7, then 2^{99} possibilities are fathomed. Many implicit enumeration algorithms have been devised for very specific problems. However, the two methods which have been most generally successful and widely applied are the cutting-plane algorithms and the branch-and-bound algorithms. The branch-and-bound algorithms, described in Section 7.9, have largely superseded cutting-plane algorithms for reasons given below.

Cutting-plane algorithms, introduced by Gomory during the 1960s [J8], have been successful in certain classes of problems. The principle of the algorithm is to eliminate parts of the constraint region R which contain no integers, by passing "cutting" planes through the extreme integer points in R. Each plane generates an additional constraint, but does not eliminate any integers because the planes bound a convex region. It is well known that the optimal continuous point lies at some intersection of boundary planes (constraints), if the objective function is linear. If a sufficient number of planes are introduced, the intersections will be only integer points, in which case a continuous optimization method (e.g., simplex algorithm) will also arrive at the integer optimum. Experience has shown that cutting-plane algorithms frequently require very large convergence times, i.e., a great number of cutting planes is often required; in this respect the method is unpredictable.

7.9 Branch-and-Bound Algorithm

The earliest use (1960) of the branch-and-bound algorithm in integer programming is credited to Land and Doig [J9]. Five years later, Dakin [J10] made a simple, but fundamental modification in the branching process to obtain a more efficient algorithm. Since that time, numerous workers have refined and improved the algorithm. Although the principles involved in the branch-and-bound algorithm are not particularly profound, it works very well in practice. It is significant to note that most computer-software codes available for large-scale mixed-integer-programming problems are based on branch-and-bound algorithms. Such software codes include APEX by Control Data Corporation, MPSX by IBM, UMPIRE by Scientific Control Systems Ltd., and LINDO by Linus Schrage (University of Chicago). The commercial branch-and-bound codes are used mainly for management decisions, especially by oil companies.

The typical commercial branch-and-bound code is a general purpose package designed to handle problems with several hundred or thousand variables. Many user options are provided to control or modify the search strategy. The programming investment is very large: several thousand FORTRAN statements are not uncommon; out-of-core storage is often necessary to satisfy memory requirements during the search; large problems may require several hours of computer time. Fortunately, the specialization of the branch-and-bound algorithm described in Section 7.10, to fit the optimization method in Section 7.7, leads to many simplifications. Thus, it was practical to program the

specialized branch-and-bound algorithm, as described in Chapter 9. The following description of the branch-and-bound algorithm provides background for the specialization; it consolidates material from many sources, including [J9-J14]. The modelling aspect of integer programming problems is discussed in [J15].

The starting point in the branch-and-bound algorithm is a temporary relaxation of the integer requirement on all integer variables, i.e., the integer variables are allowed to become continuous. The principal idea is to re-impose the integer constraints one-by-one, by means of a decision process designed to support a very efficient implicit enumeration. The record of decisions is a tree consisting of nodes and branches; some version of this tree must be stored in computer memory as it develops. Each node represents a certain set of variables which has been fixed at definite integer values, while the remaining set of variables is so far free to assume any continuous values within an appropriate set of bounds. Each branch, which emanates from a node and leads to another node, represents the introduction of a definite integer-value constraint on a previously free variable. All nodes which can be reached from a given node by travelling down branches of the tree (i.e., by introducing more integer constraints) are called descendants of the given node; conversely, the given node is called the parent node. Clearly, the top node of the tree must represent the initial total relaxation of the problem, i.e., the case where all variables are free; moving down the tree increases the number of integer variables, until reaching the terminal nodes where all variables are integers. We will consider only 0-1 integer constraints, which are the common case in

practice, with very little loss in generality. In this case, each node has one branch entering it and two branches leaving it. Also, the relaxation of the integer requirement on a 0-1 variable means that it may vary continuously between 0 and 1. General integer programming problems are usually reduced to 0-1 programming problems simply by using the binary representation for integers; this approach is a natural choice for the problem at hand.

Implicit enumeration is carried out in the branch-and-bound algorithm by applying the "bound principle" of constrained optimization. The principle states that the introduction of additional constraints cannot enlarge the feasible region for the variables and cannot improve the optimum value of the objective function. That is, the less constrained problem "bounds" the more constrained problem in both respects. To apply the principle, each node is considered to be a continuous subproblem as it is developed in the tree. Thus, an efficient continuous linear programming (LP) package (e.g., the simplex algorithm) is used to determine feasibility and find the optimum value of the objective as a function of the free variables associated with the node; the node is then "tagged" with this value.

There are four possible results for the continuous subproblem at a node: (1) the node is infeasible (i.e., the original constraint equations cannot be satisfied with the associated combination of integer and free variables); (2) the optimal objective at the node is poorer than a previously determined objective at an all-integer (terminal) node; (3) the optimal objective at the node happens to yield all integer variables; (4) the optimal solution is better than

any previously determined solution at an all-integer node. By the bound principle, it follows in cases (1) or (2), respectively, that all descendants of the node are infeasible or poorer than a previously determined solution at an all-integer node. In case (3), it is unnecessary to branch further from the node. Thus, in the first three cases the node and all its descendants are eliminated from further consideration; i.e., they are implicitly enumerated or fathomed. Note that case (2) requires a previously determined solution at an all-integer node for comparison; this is normally found in an earlier mode of operation which is designed to proceed rapidly to a good, but most-likely not optimal solution. Clearly, a better known solution allows nodes to be fathomed earlier in the tree. The user may supply a reasonable cutoff for the objective from his specific knowledge of the problem or he may set the cutoff as a certain percentage of the optimum for the initial totally relaxed (i.e., continuous) problem.

In case (4), the node remains open for exploration of its descendants. However, even in this case exploration from this particular node may be temporarily suspended in favor of exploring a more promising (unfathomed) node which was previously developed. If the more promising node leads to a sufficiently good solution, the abandoned node may revert to the fathomed category by the criterion used in case (2). The best projection criterion, for choosing the most promising node to explore, is described later.

The success of the branch-and-bound algorithm in a given problem depends critically on the efficiency of the implicit enumeration.

If the implicit enumeration failed completely, the result would be a tree which explicitly enumerates all possibilities. For example, in the case of 100 0-1 variables, the fully enumerated tree would contain rows of 1, 2, 2^2, 2^3, 2^4, ..., 2^{100} nodes in progression. This is a clear impossibility for storage or time. The possible number of distinct trees is also enormous, but it is only necessary and desirable to develop one tree efficiently. Therefore, it is important to develop a tree where only a very small fraction of the potential number of nodes needs examination. The character of such a tree is as follows: a relatively small number of paths extend to all-integer nodes at the full depth of the tree; the remaining nodes are successfully terminated at a much earlier stage, by the criteria in cases (1) and (2). If all outstanding nodes are brought to such conclusions, then optimality is actually established. On the other hand, if the computing time becomes too great, the search may be terminated and the user may accept the most optimal solution found thus far. The storage of the partial tree under development is accomplished by a listing procedure. In particular, a parent node may be eliminated as its two direct descendants are added to the list. Also, all fathomed nodes can be eliminated from such a list without danger of later redundant search. For reference purposes, it is necessary to store the most optimal all-integer node found so far.

The tree-search strategy is the major consideration in the branch-and-bound algorithm, since it determines the efficiency of implicit enumeration. Any strategy consists of an alternating sequence of two types of decisions: (1) the choice of node or subproblem

to pursue (sometimes called choice of branch); (2) the choice of free variable on which to branch from the node (i.e., on which to impose an integer constraint), as well as the choice of direction in which to arbitrate the free variable. Node selection methods have evolved from simple last-in-first-out (LIFO) procedures to the best-projection (BP) method. The BP method is an open selection from all outstanding nodes, which requires random-access memory, whereas the LIFO procedure was well adapted to a linear list on tape. In the BP method, the most promising node is selected on the basis of its (optimal) objective value and the expected degradation of the objective due to the sum of integer infeasibilities at its free variables. (The integer infeasibility at a free variable is its deviation from the nearest integer when the optimal objective is attained.) This tends to postpone poor nodes until late in the search. Initially, the choice of branching variable and its direction of arbitration was made entirely by use of penalties. (A penalty is an estimate of the degradation of the objective function due to the arbitration of a free variable; it was usually computed from information contained in the continuous (LP) optimization performed at a node.) Later, priority orderings and pseudo-costs (a more refined version of penalties) superseded penalties, because penalties were found to be unreliable in large problems. Priority orderings stem from the idea of branching on more important variables first (e.g., a major investment or a more significant bit); they may be user-initiated. All tree-search strategies are basically heuristic, but the refinements described here have produced dramatic improvements in practice.

7.10 Specialization of Branch-and-Bound Algorithm

The result of specializing the branch-and-bound algorithm to fit the optimization method described in Section 7.7 can be described as a (truncated) multivariable binary expansion which converges to a generalized point. The "generalized point" is the region R which encloses the center point. The "multivariables" are the set of coefficients which form the hyperspace. "Binary expansion" refers to the only sensible priority ordering of the (appended) zero-one variables (bits); i.e., from most-significant to least-significant bit. "Truncated" means that the convergence process must be terminated prematurely because of the limitation on bit spans. Lest the reader lose patience with the lengthy background discussion in the preceeding Sections, it should be pointed out that this simple conclusion was reached only by the tortuous path in the preceeding Sections; i.e., the abstract follows the concrete, not vice versa. The details which lead to this conclusion follow.

The initial choice of node is obviously restricted to only one possibility, the top node, where all variables are free. The choice of branching variable (bit) is subject to two considerations: the coefficient to which it belongs; the significance of the bit within that coefficient. It was found unnecessary to impose a priority ordering on the coefficients themselves, although such an ordering based on some sensitivity measure might be helpful. Thus, the ordering of coefficients is simply fixed as their natural row order in the matrix A. However, a strict priority of arbitration is imposed from most-significant bit to least-significant bit; also, the most-significant free

bits are arbitrated in a rotational order through all coefficients before proceeding to the next-most-significant bits. The reason for this procedure is to maximize the volume of the region in coefficient hyperspace which can be fathomed by a result corresponding to case (1) or case (2) in Section 7.9. This priority ordering also increases the chance of success as we proceed deeper into the tree, since the changes in arbitrated variables become progressively smaller. In fact, any reverse ordering (from an LSB to an MSB) is likely to lead to a meaningless situation.

Returning to the choice of node, we follow a best projection (BP) criterion applied only to the two nodes generated by the most recent arbitration of a variable, i.e., the two direct descendants of the node presently under consideration. We choose the descendant with the smaller value of the BP criterion. A BP criterion which was found to be effective is the sum of integer bits fixed to one and the objective of the continuous LP, at a given node. The first component of the sum is a measure of "known" hardware cost and the second component indicates the sum of coordinate distances from the nearest point in R. (This second result is obtained by choosing the (minimized) cost function of the LP to be the sum of free 0-1 variables.) This narrow use of the BP criterion leads to a "same-branch" mode of operation that pursues nodes sequentially toward a solution. This mode of operation was successful in reaching all-integer solutions, most likely because the convergence properties of the problem increase the likelihood of remaining within R. In the event that a node is fathomed, the search

backtracks to the nearest hanging node. The same-branch mode of operation also minimizes node-storage requirements. However, the computer program in Chapter 9 can be easily modified to accomodate a global BP mode.

The general characteristics of this specialization are as follows. The feasibility check by the LP is the major force in fathoming nodes and directing the search; in this respect, the continuous LP is the "heart" of the method. This fathoming was found to be so effective that fathoming by case (2) in Section 7.7 was not undertaken. The result of the fathoming process is the uniform tightening of a discrete net around R.

The main theoretical aspects of the required programming are now summarized. We assume a sensitivity matrix is supplied by another program, from which R is determined by setting bounds on real/imaginary eigenvalue variations as described in Section 7.4. A starting point (corner of the box search region) is also supplied by another program. We specify the bit span which subdivides the search region. At each node, a set of algebraic constraints on the free variables is computed from the eigenvalue constraints and the fixed variables associated with the node. These algebraic constraints are supplied to the LP. This process is facilitated by a compact matrix notation for certain required binary expansions; details can be obtained from the program listings in Appendix A, as described in Chapter 9.

Final comments are relevant to the definition of the constraint region R and the objective of the optimization program for the

continuous subproblems. We have used linear equations to define R and a linear objective function. This means that an LP can be used to solve the continuous subproblem at each node; this same choice is made in the commercial codes described in Section 7.9, because of the obvious desirability of the highly efficient LP. The limitations of linear constraints with respect to defining a region of arbitrary shape can be mitigated by an approximation process, similar to finding the area under a curve in calculus by using a sequence of linear approximations. We may use a sufficient number of line segments, plane segments, or hyperplane segments, respectively, to enclose a curve, an ellipsoid, or a hyperellipsoid. The simplest form of this approach is standard practice among the suppliers of commercial branch-and-bound codes. On the other hand, nonlinear constraints (e.g., quadratic) and objective could be used if an appropriate algorithm is available to solve the resulting nonlinear continuous subproblem. In any event, a finite algebraic representation for the constraint region is an approximation. However, the advantages of the constraint region relative to some of the methods mentioned in Section 7.1 are: it lends itself to rapid verification by computer; it may reflect bounds on as many error criteria as desired.

8. DIGITAL IMPLEMENTATION OF STATE-SPACE REALIZATIONS: DISTRIBUTED-ARITHMETIC STRUCTURE

8.1 Introduction

The final stage in the synthesis of a state-space digital filter is the digital implementation of the discrete-time state equations which have been subjected to coefficient quantization. Conceptually, this implementation consists of two parts: (1) a matrix multiplication which computes the next state and output from the present state and input; (2) feedback of the next-state variables to the present state variables through delays. The matrix multiplication requires some type of computing algorithm or structure; this structure is usually expressed by a digital flow graph consisting of multipliers, adders, sign inverters, nodes, etc. Certain types of state-space realizations have led to specifically related flow graphs; e.g., consider direct-form realizations, coupled-loop realizations, and wave-digital realizations by adaptors.

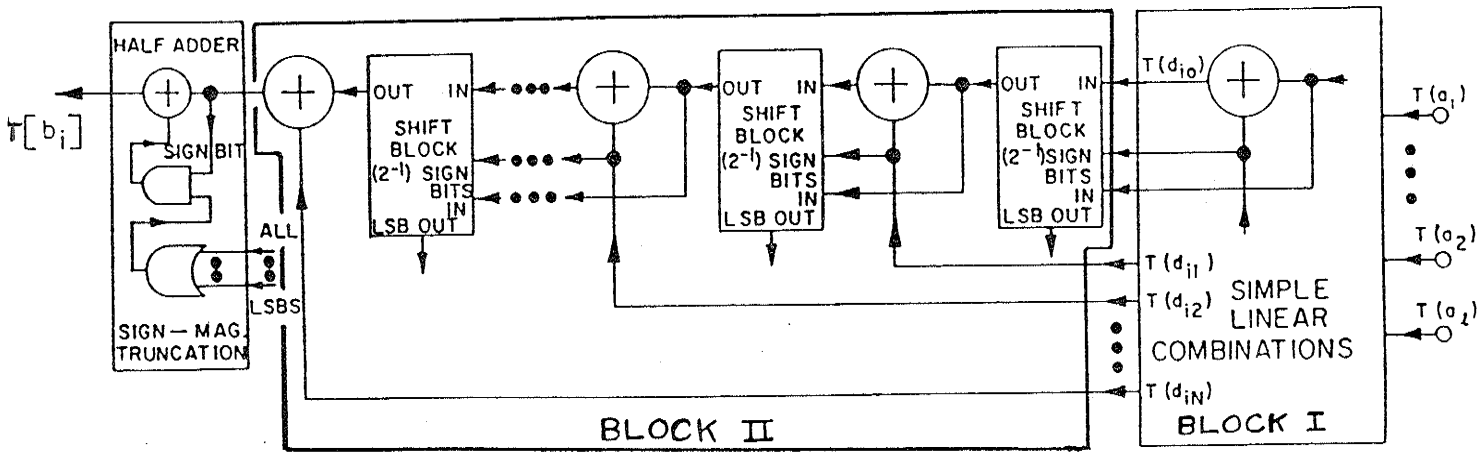
The author and Prof. G. O. Martens [H3] have described a computing algorithm which is essentially numerical in character, for the matrix multiplication; i.e., the algorithm is not related to any specific type of state-space realization. This numerical algorithm can always be applied to the matrix of finite-word-length binary (integer) coefficients which is obtained by coefficient quantization. We will use this algorithm for digital implementations in this thesis; it results in a distributed-arithmetic structure, which we will call the "shift-add structure" hereafter. In this Chapter, we

present two results of some additional research pertaining to the shift-add structure. The first result is a solution to the overflow problem, which makes it possible to implement a true saturation overflow characteristic in the shift-add structure; this is exactly what is required to utilize the results on stability in Chapter 6. The other result is a conceptual comparison with a distributed-arithmetic structure of Peled and Liu [H1].

8.2 Basic Principles in Shift-Add Structure

The shift-add structure is briefly described as follows, for reference purposes; a complete description is found in [H3]. A finite-word-length binary (integer) system matrix is obtained by coefficient quantization; multiplication by this system matrix determines the next-state and output variables from the present-state and input variables. Hereafter, we will use the terms "product variables" and "multiplicand variables" to refer to next-state/output and present=state/input variables, respectively. The integer matrix is decomposed into a binary power series with matrix coefficients, i.e., into a sequence of shifts and adds applied to the matrix coefficients; the canonical-signed-digit code is used to simplify this decomposition. The matrix coefficients contain only 1's, -1's, and 0's, which indicate simple sums and differences of the state variables; the above-mentioned sequence of shifts and adds is overlaid on these simple sums and differences.

The shift-add structure is shown in Fig. 5, where it is decomposed into two basic sections. These sections are: (1) Block



Multiplicand Variables: $a_1, a_2, \ldots, a_\ell$

Two's Complement Forms: $T(\ )$

Simple Sums and Differences: $d_{i1}, d_{i2}, \ldots, d_{iN}$

Underflow Bits: All LSB Out

Product Variable: b_i, $i = 1, 2, \ldots, \ell$

Figure 5. Shift-Add Structure

I, which forms combinations (i.e., simple sums and differences) of the multiplicand variables; (2) Block II, which forms a sequence of partial sums alternating with shift operations. These partial sums are formed by iterating the process of adding a combination to the shifted previous partial sum; the initial partial sum is simply one of the combinations. Note that the structure does not contain any individual coefficient multipliers.

The application of two's complement (2's C) arithmetic to the signals within the shift-add structure described in [H3] leads to the following advantage. The results of the matrix multiplications are exact, where the required adder word lengths in the structure are the same as the multiplicand word lengths, if we exclude the possibility of adder overflows. That is, all bits (including underflow bits) of the next-state and output variables are correctly computed, but the word lengths of the adders, shifts, and sign inverters are the same as the word lengths of the multiplicand variables. Other basic hardware advantages of two's complement arithmetic are already well known.

The exact computation of next-state variables is a significant advantage of the above structure with respect to FWLEs. This exact computation gives the minimum possible roundoff error at each state variable; this error is only that dictated by the elimination of underflow bits to meet the word-length constraint at the feedback registers. Hence, this structure fully exploits state-space realizations which are initially designed to reduce or minimize roundoff noise by means of low time-domain sensitivity to

signal disturbances. In most other structures there are usually internal underflow roundoffs at coefficient multipliers, which increase the magnitude of roundoff error at the state variables. The internal roundoffs in these structures are performed to avoid the increase in adder and multiplier word lengths which would be necessary for exact computation.

The combinations computed in Block I are m+1-bit 2's C forms, where $m = \max m_i$; m_i was defined in Section 4.4. However, the results computed in Block II are m+N+1-bit 2's C forms, since each result includes N underflow bits which occupy the far-right positions; N was also defined in Section 4.4. We regard these m+N+1-bit 2's C forms as integers representing the actual product variables multiplied by 2^N. We will use the term "scaled product variables" to refer to the actual product variables multiplied by 2^N.

The underflow problem has been adequately resolved in [H3]. The N underflow bits in each scaled product variable must be discarded to meet the fixed-word-length constraint at the feedback registers. Normally, we choose to implement either a true rounding characteristic or a true magnitude-truncation characteristic when the underflow bits are discarded. This will be done by the use of subsidiary logic which generates an appropriate correction bit to be added to the least significant bit of the remaining m+1 bits. These remaining m+1 bits comprise an m+1-bit 2's C form, which represents the actual, or "unscaled", product variable. This m+1-bit 2's C form will represent the unscaled product variable with the minimum error possible under the constraint of either

magnitude truncation or rounding. This minimum error follows because all bits were correctly computed in the scaled product variable.

8.3 Statement of Overflow Problem

The exact product variables may require overflow as well as underflow bits, due to the nature of the system matrix. Thus, adder overflows within the shift-add structure described in [H3] may introduce errors in the computed values of the scaled product variables. As we will show in Section 8.4, these errors will consist of overflow conditions and/or incorrect bits in the scaled product variables. Adder overflows may occur at any adder in Block I or II. However, any adder overflows in Block II will be corrected automatically by means of sign-bit correction logic and shifting operations, as explained in [H3]. Thus, only the overflow conditions in Block I will ultimately produce errors in the scaled product variables.

8.4 Analysis of Errors Due to Overflows

The errors in the scaled product variables, produced by overflow conditions in Block I, can be analyzed by the principle of linear superposition. This is possible because the property of linearity holds for the entire network of shifts and adders which comprise the shift-add structure. Consider the possible set of overflow errors which may be present in the combinations which are output from Block I in Fig. 5. The possible overflow error in each combination equals the modulus (2^{m+1}) of the 2's C representa-

tion multiplied by a (positive or negative) integer. The integer multiplier equals the number of net overflows which occurred in computing that combination. (The number of net overflows is defined as the number of positive overflows minus the number of negative overflows.) By the linearity property, the set of errors in the scaled product variables is the response of Block II to the particular set of overflow errors in the combinations which are output from Block I. This error response of Block II is a linear combination of a set of shifted overflow errors. Thus, it is clear that the error response may include overflow conditions as well as incorrect bits in the scaled product variables.

8.5 Solution to Overflow Problem

The most straightforward way to solve the overflow problem is simply to "correct" any possible overflows in Block I. This requires the appendage of a small number of front bits to each adder in Block I. The same number of front bits must also be appended to each adder in Block II, to carry through the results of Block I. It is doubtful that there is any less costly method of solving the problem, since the previous analysis shows that the errors due to overflows may be very general in nature. In fact, the separate calculation of error-correction terms for the scaled product variables would require a duplication of Block II, using the overflow errors of Block I as inputs. The above-mentioned method, consisting of appending extra front bits, in effect calculates the error-correction terms for overflows and simultaneously adds them

to the result. We elaborate on this method below.

The purpose of appending the same number of front bits to each adder in the shift-add structure is to increase the modulus of the 2's C representation. We let m_o represent the number of bits appended to each adder. In this case, the modulus of the 2's C forms computed in Block I will increase from 2^{m+1} to 2^{m_o+m+1}. It will also be necessary to append one more front bit to the final adder in each sequence of shifts and additions in Block II. This extra front bit is necessary to allow for the possibility of overflow at each final adder, since there is no shifting operation following this adder. The extra front bit is determined by the same sign-bit= correction logic described in [H3]. Hence, the modulus of each scaled product variable computed in Block II will increase from 2^{m+N+1} to $2^{m_o+m+N+2}$. (See Section 8.6 for more detail on 2's C arithmetic, including the extra front bit.)

We will use the term "augmented structure" to refer to the original structure with appended front bits. The key point is to restrict the multiplicand variables in the augmented structure to the original set of integers. Here the original set of integers means those integers which are representable with modulus 2^{m+1} in 2's C arithmetic; this is the set used for the multiplicand variables in the original structure. With the above-mentioned restriction, it is possible to determine m_o such that no overflows can occur in Block I of the augmented structure, as described later. Consequently, the scaled product variables in the augmented

structure will give the exact result of the matrix multiplication. That is, no overflow conditions will exist and all bits will be correct in the scaled product variables of the augmented structure, regardless of overflow conditions in the original structure.

The scaled product variables, computed as $m_o+m+N+2$-bit 2's C forms in the augmented structure, must finally be reduced to m+1-bit 2's C forms. These m+1-bit forms should be magnitude-truncated or rounded versions of the unscaled product variables. This is necessary to meet the word-length constraint at the feedback registers in the original structure. The reduction of each scaled product variable (in 2's C form) consists of the following steps: (1) elimination of the m_o+1 front (i.e., overflow) bits; (2) elimination of the N underflow bits. The procedure for elimination of the N underflow bits will be exactly the same as described in Section 8.2 for the original structure. Thus, it is only necessary to consider here the elimination of the m_o+1 overflow bits.

The elimination of the m_o+1 overflow bits in the 2's C form of each scaled product variable implies that the modulus of the 2's C form must decrease from $2^{m_o+m+N+2}$ to 2^{m+N+1}. However, the scaled product variable may or may not lie within the domain which is representable in 2's C form by using the modulus 2^{m+N+1}. If the scaled product variable does lie within the domain, the m_o+1 overflow bits can be eliminated without introducing any error in the 2's C representation. This is the non-overflow condition. If the scaled product variable in the augmented structure does not lie within the

above-mentioned domain, then a true overflow condition must exist at the same scaled product variable in the original structure. Under the overflow condition, some error will be introduced by eliminating the m_o+1 overflow bits. In Section 8.6, we explain the simple procedure for: (1) detecting the true overflow or non-overflow condition from the $m_o+m+N+2$-bit 2's C form; (2) eliminating the m_o+1 overflow bits under the overflow or non-overflow condition.

In the event of the true overflow condition, the $m+N+1$-bit 2's C form, which remains after the elimination of the m_o+1 overflow bits, should be altered to conform to the desired overflow characteristic, for reasons discussed in Section 5.3 and Chapter 6. The saturation overflow characteristic is obtained by altering the $m+N+1$-bit 2's C form to represent the positive (negative) limit of its domain in the event of a positive (negative) overflow.

We now determine m_o, the number of appended front bits, sufficient to exclude overflow conditions in Block I of the shift-add structure. Block I computes combinations of the multiplicand variables. Each combination is a sum formed from the ℓ distinct multiplicand variables, using only the coefficients +1, -1, and 0. We let q represent the maximum number of non-zero coefficients in any combination in Block I, for a given shift-add structure; clearly, $q \leq \ell$. Then the magnitude of all possible partial and final sums in Block I will be bounded by $q2^m$, provided that the multiplicand variables are restricted to the original set of integers. Hence, it is only necessary to determine the number of bits required to

represent the numbers $q2^m$ and $-q2^m$ in 2's C form, in order to exclude the possibility of overflow conditions in Block I. We set m_o+m+1 equal to the required number of bits, so by the rules of 2's C arithmetic we must have $2^{m_o+m} - 1 \geq q2^m$. This gives

$$2^{m_o} \geq q + 1/2^m \quad . \tag{8.1}$$

which we may safely approximate by

$$2^{m_o} \geq q + 1/2 \tag{8.2}$$

for m = 1, 2, ... Thus

$$m_o \geq \log_2 (q + 1/2) \quad ; \tag{8.3}$$

m_o should be chosen as the smallest integer which satisfies (8.3). We may also use the more-conservative general result

$$m_o \geq \log_2 (\ell + 1/2) \quad , \tag{8.4}$$

in place of (8.3). The logarithmic variation of m_o with ℓ implies that m_o will not be excessive for high-order systems. For example, consider a sixth-order system with one input; here $\ell = 7$. In this case, we may use $m_o = 3$ bits to satisfy the conservative requirement (8.4).

8.6 Subsidiary Details Pertaining to Two's Complement Arithmetic

Here we elaborate on some subsidiary matters, pertaining to 2's C arithmetic, which were deferred in the previous discussions. We consider the operation of changing the modulus, hence the number of bits, in the 2's C representation of a fixed integer. This change of modulus is required in two places in the augmented structure: (1) for the multiplicand variables, the modulus must be increased from 2^{m+1}, which corresponds to the original structure, to 2^{m_o+m+1}, which corresponds to the augmented structure; (2) for the scaled product variables, the modulus must be decreased from $2^{m_o+m+N+2}$ to 2^{m+N+1}, to eliminate the m_o+1 front bits, prior to the elimination of the N underflow bits. We also discuss the determination of the extra front bit required at each of the final adders in Block II.

We will follow the summary of 2's C arithmetic given in Section V of [H3]. The 2's C representation $T_{(m)}(X)$ of a signed m-bit integer X is defined by

$$T_{(m)}(X) = \begin{cases} X & \text{if } 0 \leq X < 2^m \\ 2^{m+1} + X & \text{if } -2^m \leq X < 0. \end{cases} \tag{8.5}$$

Hence, $T_{(m)}(X)$ is an m+1-bit positive integer; we call 2^{m+1} the modulus of $T_{(m)}(X)$. The lexicographic notation for the bits of $T_{(m)}(X)$ is:

$$T_{(m)}(X) = X_0 X_1 \ \ldots \ X_m \ . \tag{8.6}$$

The rule for increasing the modulus from 2^{m+1} to 2^{m+m_o+1}, in the 2's C representation of a fixed integer X, is given by

$$T_{(m+m_o)}(X) = \underbrace{X_0\ X_0 \cdots X_0}_{m_o \text{ times}}\ X_0\ X_1\ X_2 \cdots X_m\ , \qquad (8.7)$$

where $T_{(m)}(X)$ is given in (8.6). That is, to $T_{(m)}(X)$ we append m_o front bits which are the same as the sign bit X_0. This rule can be derived from a sequence of two operations: (1) scaling up $T_{(m)}(X)$ by 2^{m_o}, which gives $T_{(m+m_o)}(2^{m_o}X)$; (2) shifting (i.e., multiplying $2^{m_o}X$ by 2^{-m_o}), which gives $T_{(m+m_o)}(X)$. The scaling and shifting operations are detailed in Section V of [H3].

The decrease of the modulus from 2^{m+m_o+N+2} to 2^{m+N+1} in the 2's C representation of a fixed integer Y is possible if and only if $-2^{m+N} \leq Y < 2^{m+N}$. If $-2^{m+N} \leq Y < 2^{m+N}$, then $T_{(m+N)}(Y)$ exists, which we denote by

$$T_{(m+N)}(Y) = Y_0\ Y_1\ Y_2 \cdots Y_{m+1}\ Y_{m+2} \cdots Y_{m+N}\ . \qquad (8.8)$$

By the previously described rule for increasing the modulus, we also have

$$T_{(m_o+m+N+1)}(Y) = \underbrace{Y_0\ Y_0 \cdots Y_0}_{m_o+1 \text{ times}}\ Y_0\ Y_1\ Y_2 \cdots$$

$$\cdots\ Y_{m+1}\ Y_{m+2} \cdots Y_{m+N}\ . \qquad (8.9)$$

Conversely, if $T_{(m_o+m+N+1)}(Y)$ has the form given by (8.9), then the corresponding form (8.8) specifies the same integer, as seen from the rule for increasing the modulus; hence $-2^{m+N} \leq Y < 2^{m+N}$. In summary, $-2^{m+N} \leq Y < 2^{m+N}$ if and only if $T_{(m+m_o+N+1)}(Y)$ has the form given in (8.9), i.e., the form where the first m_o+2 bits are the same. In this case, we obtain $T_{(m+N)}(Y)$ by dropping the first m_o+1 bits; this decreases the modulus of the 2's C representation from 2^{m+m_o+N+2} to 2^{m+N+1}.

Y lies outside the interval $[-2^{m+N}, 2^{m+N})$ if and only if $T_{(m+m_o+N+1)}(Y)$ does not have the form in (8.9). In this event, an overflow condition would exist with respect to $T_{(m+N)}(Y)$. Clearly, this overflow condition can be detected by checking to see if the first m_o+2 bits of $T_{(m+m_o+N+1)}(Y)$ are not all alike. The sign of the overflow would be indicated by the sign bit in $T_{(m+m_o+N+1)}(Y)$. As previously mentioned, the saturation overflow characteristic is implemented by using the limit $Y = 2^{m+N} - 1$ ($Y = -2^{m+N}$) to determine $T_{(m+N)}(Y)$ in the event of a positive (negative) overflow. Thus, in the event of a positive (negative) overflow, $T_{(m+N)}(Y)$ will have a zero (one) as the sign bit, followed by an unbroken string of ones (zeroes).

We summarize the application of the above principles, for the feedback of scaled next-state (i.e., product) variables to unscaled present-state (i.e., multiplicand) variables. Each scaled next-state variable in the augmented structure is given by a 2's C representation with modulus 2^{m+m_o+N+2}. An overflow condition, with respect to the modulus 2^{m+N+1} in the original structure, does not

exist at a scaled next-state variable if the first m_o+2 bits are the same; otherwise, an overflow condition does exist. If an overflow condition does not exist, then it is only necessary to delete the first m_o+1 bits before elimination of underflow bits and subsequent feedback. If an overflow condition does exist, then its sign is determined by the sign bit of the 2's C representation with modulus $2^{m_o+m+N+2}$. In the 2's C representation with modulus 2^{m+N+1}, the limit with the same sign as the overflow is substituted before elimination of underflow bits and subsequent feedback. After feedback, in either case, m_o front bits are restored; they are the same as the leading bit of the m+1 bits which have been fedback. Thus, the storage of the m_o+1 front bits in the feedback registers is avoided.

It was mentioned in Section 8.5 that an extra front bit must be appended to each final adder in Block II, to avoid a possible overflow. The determination of the extra front bit does not actually require an extra single-bit adder. It can be determined by the sign= bit correction logic described in [H3]. We demonstrate the principle involved, as follows. Assume that two integers U and W are represented by P+1-bit 2's C forms, i.e.,

$$T_{(P)}(U) = U_0\ U_1\ U_2\ \ldots\ U_P\ , \tag{8.10}$$

$$T_{(P)}(W) = W_0\ W_1\ W_2\ \ldots\ W_P\ . \tag{8.11}$$

Let

$$T_{(P)}(U) + T_{(P)}(W) = T_0\ T_1\ T_2\ \ldots\ T_P \pmod{2^{P+1}} \tag{8.12}$$

If an overflow occurs in (8.12), then the sign bit T_o is incorrect, since an overflow is equivalent to $T_o \neq U_o = W_o$; otherwise, T_o is correct. Now consider the P+2-bit 2's C forms for the same integers U and W, obtained by appending an extra sign bit to the front of (8.10) and (8.11):

$$T_{(P+1)}(U) = U_o\ U_0\ U_1\ U_2\ \ldots\ U_P\ ; \tag{8.13}$$

$$T_{(P+1)}(W) = W_o\ W_0\ W_1\ W_2\ \ldots\ W_P\ . \tag{8.14}$$

Then

$$T_{(P+1)}(U) + T_{(P+1)}(W) = T_*\ T_0\ T_1\ T_2\ \ldots\ T_P \pmod{2^{P+2}}; \tag{8.15}$$

i.e., T_0, T_1, T_2, ..., T_P in (8.15) are already correctly computed in (8.12). Also, an overflow cannot occur in (8.15), since $T_* \neq U_o = W_o$ cannot occur; i.e., the sign bit T_* is always correct. Thus, if there is an (no) overflow in (8.12), then $T_* = \bar{T}_o$ ($T_* = T_o$), since T_o is incorrect (correct). The sign-bit correction logic described in [H3] implements this last-stated rule. That is, it will determine the appended front bit T_* following the final adder, by inverting (not inverting) the output sign bit T_o at that adder if there is (is not) an overflow.

8.7 Comparison of Shift-Add Structure to the Structure of Peled/Liu

It is useful to compare the shift-add structure given in [H3] to the structure given by Peled and Liu in [H1]. Hereafter, we

will refer to these structures as Structure I and Structure II, respectively. There are certain similarities between these structures: (1) both structures are based on the use of distributed arithmetic; (2) both structures contain a sequence of shift-add operations to determine the final result. The essential distinction between the two structures is due to different decompositions of the required arithmetic operations. In a certain sense, these two different decompositions of arithmetic operations are complementary. It is also possible that other decompositions can be developed.

We may regard both structures as digital realizations which compute the linear combination

$$y = a_1 x^{(1)} + a_2 x^{(2)} + \ldots + a_n x^{(n)} \tag{8.16}$$

Here $a_1, a_2, \ldots, a_n$ and $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ are finite-word-length binary fixed coefficients and independent variables, respectively. In both structures, the 2's C representation is used for the variables, i.e.,

$$T(x^{(k)}) = X_0^{(k)} \; X_1^{(k)} \; X_2^{(k)} \; \ldots \; X_m^{(k)},$$
$$k = 1, 2, \ldots, n, \tag{8.17}$$

in lexicographic notation. It is well known (see, e.g., [H3]) that the actual value $x^{(k)}$ can be computed from the 2's C representation in the following way:

$$x^{(k)} = - X_0^{(k)} 2^m + \sum_{i=1}^{m} X_i^{(k)} 2^{m-i},$$
$$k = 1, 2, \ldots, n. \tag{8.18}$$

It is clear from equation (5) in [H1] that the basic function of Structure II is to compute a form like (8.16). That is, equation (5) in [H1] has the same form as (8.16), except that the number of independent variables is fixed at five, because Structure II is applied only to second-order digital-filter sections in [H1]. However, it is a simple matter to generalize Structure II to allow an arbitrary number of independent variables. The computation of a form like (8.16) is also the basic function of Structure I, which multiplies a matrix into a vector. This follows since matrix multiplication into a vector must determine the set of inner products between the matrix row vectors and the multiplicand vector. Each inner product has the form (8.16). In this discussion, we will restrict Structure I to the realization of a single inner product, which is equivalent to (8.16). It is fairly straightforward to develop matrix generalizations of (8.16) for both Structures I and II.

We now abstract the principle of Structure II, from [H1]. The basic definition of this structure is contained entirely within Section II of [H1], specifically in equations (5) through (9) of [H1]. The remainder of that paper concerns applications and performance of the new structure. We will use a somewhat different notation from [H1], since we are not restricted to second-order sections. However, the essential content of Section II of [H1], concerning arithmetic operations, will be preserved here.

The principle of Structure II follows by substituting (8.18) into (8.16), which gives

$$y = \sum_{k=1}^{n} a_k \left(- X_o^{(k)} 2^m + \sum_{i=1}^{m} X_i^{(k)} 2^{m-i} \right) . \qquad (8.19)$$

The central idea is to interchange the order of summations in (8.19), as follows:

$$y = - 2^m \sum_{k=1}^{n} a_k X_o^{(k)} + \sum_{i=1}^{m} 2^{m-i} \sum_{k=0}^{n} a_k X_i^{(k)} . \qquad (8.20)$$

Following [H1], we consolidate matters by defining the function F, with n binary arguments, $B^{(1)}$, $B^{(2)}$, ..., $B^{(n)}$, as follows:

$$F (B^{(1)}, B^{(2)}, \ldots, B^{(n)}) = \sum_{k=1}^{n} a_k B^{(k)} . \qquad (8.21)$$

We may rewrite (8.20), by using (8.21), to obtain

$$y = - 2^m F (X_o^{(1)}, X_o^{(2)}, \ldots, X_o^{(n)}) + \sum_{i=1}^{m} 2^{m-i} F (X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(n)}) \qquad (8.22)$$

which summarizes the concepts of Structure II. Note that F consists of simple combinations of the coefficients a_k, k = 1, 2, ..., n. The function F can be realized by a combinatorial network or stored in a ready-only memory, since it can take on only 2^n distinct values which depend on its binary arguments. Such a realization of F offers advantages with respect to speed and power consumption; these advantages are exploited in [H1]. However, as mentioned in [H1], fixing F sacrifices the flexibility of changing the coefficients a_1, a_2, ..., a_n; i.e., a separate realization of F is required for each set

of coefficients. A final sequence of shift-add operations is required to compute y in (8.22); this sequence of shift-add operations applies to the functions F of the different sets of binary arguments.

We now derive Structure I from (8.16), for comparison to Structure II. The principle of Structure I follows by decomposing each of the coefficients, $a_1, a_2, \ldots, a_n$ in (8.16), into a binary series. Here it is practical to use the canonical signed-digit code (CSDC), rather than the 2's C or sign-magnitude representation, because the coefficients are fixed. The main advantage of the CSDC is economy; i.e., it requires the minimum number of non-zero bits to represent a coefficient. We will denote the CSDC for each coefficient by

$$C\,(a_k) = A_k^{(N)}\, A_k^{(N-1)}\, A_k^{(N-2)} \ldots A_k^{(0)},\ k = 1, 2, \ldots, n, \tag{8.23}$$

in lexicographic notation. Each bit $A_k^{(i)}$, $i = 1, 2, \ldots, N$, may be either 1, -1, or 0, but there will be no adjacent non-zero bits in $C\,(a_k)$. The value of a_k is simply

$$a_k = \sum_{i=1}^{N} 2^i\, A_k^{(i)},\ k = 1, 2, \ldots, n, \tag{8.24}$$

where we have scaled the coefficients to integers with no loss of generality. We substitute (8.24) in (8.16), which gives

$$y = \sum_{k=1}^{n} \left(\sum_{i=1}^{N} 2^i\, A_k^{(i)} \right) x^{(k)}\ . \tag{8.25}$$

The central idea, as in Structure II, is to interchange the order of summations; here the interchange is in (8.25), to obtain

$$y = \sum_{i=1}^{N} 2^i \sum_{k=1}^{n} A_k^{(i)} x^{(k)} \quad . \tag{8.26}$$

To consolidate matters, we define

$$D_i \triangleq \sum_{k=1}^{n} A_k^{(i)} x^{(k)} \quad , \; i = 1, 2, \ldots, N \; ; \tag{8.27}$$

substituting (8.27) into (8.26) gives

$$y = \sum_{i=1}^{N} 2^i D_i \quad , \tag{8.28}$$

which summarizes the concepts of Structure I.

Thus, Structure I consists of: (1) computation of the simple combinations D_i, $i = 1, 2, \ldots, N$, of the variables $x^{(1)}$, $x^{(2)}$, $\ldots$, $x^{(n)}$, given by (8.27); (2) a sequence of shift-add operations applied to these simple combinations D_i, $i = 1, 2, \ldots, N$, given by (8.28). Note that the combinations D_i, $i = 1, 2, \ldots, N$, have only the coefficients 1, -1, and 0. The original intention in [H3] is to realize these combinations by a simple network of interconnected adders; no shifts are required here. In some cases, a reduction in the number of adders in this network can be achieved by exploiting the occurrence of common sums in the D_i, $i = 1, 2, \ldots, N$. Also, the network of adders is somewhat flexible, since it can be re-wired to accomodate changes in the coefficients. An exact method of performing the final sequence of shift-add operations in 2's C form is shown in [H3], where all adders require only the same word length as the combinations D_i, $i = 1, 2, \ldots, N$. This method is also

directly applicable to the sequence of shift-add operations required by (8.22) in Structure II; however, this was not mentioned in [H1].

We are now in a position to make some comparisons between Structure I and Structure II. The different arithmetic decomposition used in each of the structures can be summarized as follows:

(1) a binary series expansion is used in both structures;

(2) in Structure I, the expansion is around the (CSDC) digits of the coefficients;

(3) in Structure II, the expansion is around the (2's C) digits of the variables;

(4) simple combinations (sums) are formed by grouping like terms from the expansion, in both structures;

(5) in Structure I, the simple combinations are sums of the variables, with fixed coefficients 1, -1, 0;

(6) in Structure II, the simple combinations are sums of the fixed original coefficients, with coefficients 1 and 0 which depend on the digits of the variables;

(7) a final sequence of shift-add operations on the simple combinations is required in both structures.

The number of elementary shift-add operations required in each structure would be the same, if the decomposition in each structure were extended as far as products between individual bits of the original coefficients and signals. However, certain advantages accrue from the inherent grouping of terms in each structure. Structure II is well suited to combinational or memory-storage techniques for "computing" the simple combinations. Note that the

CSDC cannot be applied to any particular advantage in this structure, since the CSDC does not appear to be useful for the representation of variables. The original intention in Structure I is to compute the simple combinations by a network of adders. The problem of reducing or minimizing the number of adders in this network requires further investigation; however, certain methods have been found which work well in practice. The generalization of Structure I to matrix multiplication may lead to simple combinations which contain additional common sums; this is unlikely in Structure II.

We also mention that it is possible to apply combinatorial or memory-storage techniques in Structure I. Here we will only outline this approach to Structure I; a more detailed investigation should be undertaken. The application of these techniques rests on decomposing the simple combinations, D_i, $i = 1, 2, \ldots, N$, which were defined in (8.27). That is, we substitute (8.18) into (8.27), which gives

$$D_i = \sum_{k=1}^{n} A_k^{(i)} \left(- X_o^{(k)} 2^m + \sum_{j=1}^{m} X_j^{(k)} 2^{m-j} \right) , \qquad (8.29)$$

$i = 1, 2, \ldots, N$, and interchange the order of summations in (8.29), to obtain

$$D_i = - 2^m \sum_{k=1}^{n} A_k^{(i)} X_o^{(k)} + \sum_{j=1}^{m} 2^{m-j} \sum_{k=1}^{n} A_k^{(i)} X_j^{(k)} . \qquad (8.30)$$

We define the function G_i, with n binary arguments, $B^{(1)}$, $B^{(2)}$, ..., $B^{(n)}$, as follows:

$$G_i\,(B^{(1)}, B^{(2)}, \ldots, B^{(n)}) = \sum_{k=1}^{n} A_k^{(i)}\, B^{(k)}\,, \qquad (8.31)$$

$i = 1, 2, \ldots, N$. Then we have

$$D_i = -\,2^m\, G_i\,(X_o^{(1)}, X_o^{(2)}, \ldots, X_o^{(n)})$$

$$+ \sum_{j=1}^{m} 2^{m-j}\, G_i\,(X_j^{(1)}, X_j^{(2)}, \ldots, X_j^{(n)})\,, \qquad (8.32)$$

$i = 1, 2, \ldots, N$. The decomposition in (8.32) of the combinations D_i, $i = 1, 2, \ldots, N$, is analogous to the decomposition used in Structure II. However, the coefficients $A_k^{(i)}$, $k = 1, 2, \ldots, n$, in D_i and G_i are only single bits; also, many of them will be zero due to the CSDC. Hence, the functions G_i, $i = 1, 2, \ldots, N$, which are to be stored in memory or realized combinatorially, will be quite simple; only a few bits will be required to represent the outcome of each G_i. This offsets the fact that N different functions G_i are required. Various design techniques for logic functions may be used to reduce the cost of combinatorial realization or memory-storage of the G_i, $i = 1, 2, \ldots, N$. The shift-add sequence in (8.32) may be realized by the same technique associated with Structure I; note that only a few bits are required in each adder in this sequence.

9. DESIGN PROCEDURE, COMPUTER PROGRAMS, AND EXAMPLES

9.1 Introduction

In this Chapter, various results from the previous Chapters are combined into an overall design procedure for state-space digital filters. The design procedure and a few subsidiary derivations are given in Section 9.2. The entire procedure is implemented by a battery of interactive computer programs, described in Sections 9.3 and 9.4, and listed in Appendix A. By means of these programs, it is possible to design, test, modify, and optimize state-space digital filters in a single interactive computer session. A series of examples (low-pass filters) is presented and interpreted in Sections 9.5 and 9.6. In Section 9.7, hardware considerations related to the implementation method of Chapter 8 are discussed.

9.2 Design Procedure

A. Outline

The steps leading from an initial approximation in the frequency-domain to the desired state-space digital filter may be outlined as follows:

1) Choose a suitable rational approximation to the desired characteristic in the s-domain, e.g., Butterworth, Chebyshev,

etc. for a low-pass characteristic ; a table of coefficients, possibly with pole-zero locations, will be used.

2) Apply frequency-scaling in the s-domain, if necessary.

3) Generate the z-domain transfer function from the s-domain approximation, by using the bilinear transformation (2.4); tables or explicit formulas for the transformed coefficients can be used to reduce the labor here.

4) Generate the companion-form state-space realization from the coefficients of the z-domain transfer function.

5) Transform the companion-form to the MRN state-space realization, by using the appropriate similarity transformation to diagonalize K and W.

6) Scale the signal levels in the MRN realization, to meet constraints on overflow probability.

7) Quantize the entries in the A matrix, subject to constraints on eigenvalue variation.

8) Re-compute B, C, and D, to restore the original zeroes and zero-frequency gain in the transfer function.

9) Quantize B, C, and D, subject to constraints on variation of zeroes and zero-frequency gain.

10) Check performance: frequency-response; diagonal dominance in K and W; and roundoff noise.

11) Repeat steps 7) through 10) if necessary.

12) Determine a hardware implementation of quantized state equations by using the shift-add structure.

Steps 1) through 3) are the well-known preliminaries to realizing a direct-form or building-block-form digital filter [A6]. We now consider certain details relevant to steps 4) through 12); in cases where tables of appropriate z-domain transfer functions are available, the design procedure itself may begin at step 4).

B. Transformation of Companion-Form to MRN Realization

The companion form of the state-space realization (3.1) corresponds to

$$A = \begin{bmatrix} 0 & 1 & 0 & & \\ 0 & 0 & 1 & 0 & \\ & & 0 & \ddots & \\ & & & & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \end{bmatrix}, \qquad (9.1)$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \qquad D = b_o, \qquad (9.2)$$

and

$$
\begin{aligned}
c_n &= b_1 - b_o a_1 \\
c_{n-1} &= b_2 - b_o a_2 \\
&\vdots \\
c_2 &= b_{n-1} - b_o a_{n-1} \\
c_1 &= b_n - b_o a_n
\end{aligned}
\tag{9.3}
$$

for the components of C. The coefficients $a_1, a_2, \ldots, a_n$ and $b_o, b_1, \ldots, b_n$ in (9.1) through (9.3) are those of the transfer function H(z) in (2.1), where it is assumed that $a_o = 1$. The matrix T for the similarity transformation (3.5) needed in step 5) is computed as follows, by the straightforward procedure given in [C10]. The solutions of the Lyapunov equations (5.2) and (5.3) corresponding to A_o, B_o, and C_o in (3.5) are

$$K_o = T^{-1} K (T^{-1})^T \tag{9.4}$$

and

$$W_o = T^T W T \quad , \tag{9.5}$$

respectively. Diagonalizing the symmetric matrix W gives

$$W = U_1 \bar{D} U_1^T \quad , \tag{9.6}$$

where U_1 is a unitary matrix. Assuming a transformation matrix

$$T_1 = U_1 \bar{D}^{-1/2} \tag{9.7}$$

in place of T in (3.5) gives

$$W_o = I \tag{9.8}$$

by (9.5) and (9.6) and

$$K_o = T_1^{-1} K (T_1^{-1})^T \tag{9.9}$$

by (9.4), where I is the identity matrix. Computing the diagonal form of the symmetric matrix K_o in (9.9) gives

$$K_o = U_2 \bar{D}_2 U_2^T \quad , \tag{9.10}$$

where U_2 is a unitary matrix. Therefore, the composite transformation matrix

$$T = T_1 U_2 \tag{9.11}$$

is chosen in (3.5) because it produces

$$K_o = \bar{D}_2 \tag{9.12}$$

and

$$W_o = I \quad , \tag{9.13}$$

by reapplying (9.4) and (9.5), respectively. For wave digital filters, steps 3) through 5) are unnecessary. Instead, the chosen s-domain transfer function is realized by an analog prototype network which ultimately provides the discrete-time state equations. The diagonal solutions K and W are inherent in the wave digital filter.

C. Scaling

It is necessary to develop further certain basic ideas in [C10], to obtain a specific procedure for the scaling required in step 6). First, we set down some basic definitions:

$$\tilde{\sigma}_{in} \triangleq \text{standard deviation of the unquantized input signal;} \tag{9.14}$$

$$\varepsilon_{in} \triangleq \text{input quantization step size.} \tag{9.15}$$

The word length m_i assigned to the i^{th} state variable, $i = 1, \ldots, n$, in Section 4.4A is assumed to be the optimal word length given by [C10]

$$m_i - m = \frac{1}{2}\left[\log_2 K_{ii} W_{ii} - \frac{1}{n} \sum_{j=1}^{n} \log_2 K_{jj} W_{jj}\right] , \tag{9.16}$$

where m is the chosen average word length; since (9.16) generally produces non-integer values, m_i will eventually be rounded to the nearest integer. The roundoff-noise invariant for a given transfer function is [C10]

$$M_g = [\det (KW)]^{1/2n} , \tag{9.17}$$

which specializes to

$$M_g = \left[\prod_{j=1}^{n} (K_{jj} W_{jj})\right]^{1/2n} \tag{9.18}$$

for diagonal matrices K and W. By using (9.18), we may rewrite (9.16) as

$$m_i - m = 1/2 \log_2 K_{ii} W_{ii} - \log_2 M_g . \tag{9.19}$$

$m_i - m$ is invariant under a simple scaling transformation if K and W are diagonal, because $K_{ii} W_{ii}$ will be invariant.

The effect of input quantization may be broken into two parts: scaling by the factor $1/\varepsilon_{in}$; discretization to integer values. Here we ignore the discretization, which simply means that input-quantization noise is set aside. The standard deviation of the scaled input is

$$\sigma_{in} = \left(\frac{1}{\varepsilon_{in}}\right) \tilde{\sigma}_{in} \quad ; \tag{9.20}$$

this scaled input is u(k) in (3.1). We define a parameter δ_{in} which determines the probability of amplitude-limiting (i.e., overload or saturation) in the input quantizer:

$$\delta_{in} \triangleq \frac{2^m}{\sigma_{in}} \quad , \tag{9.21}$$

since 2^m bounds the magnitude of the input representation. The actual choice of ε_{in} is a compromise between reducing input quantization noise (small ε_{in}) and reducing the probability of amplitude limiting (large ε_{in}; hence large δ_{in}, by (9.20) and (9.21)). The standard deviation σ_i at the i^{th} state variable is [C10]

$$\sigma_i = \left(\sqrt{K_{ii}}\right) \sigma_{in} \quad , \quad i = 1, \ldots, n \quad . \tag{9.22}$$

A parameter δ has been defined [C10] to determine the probability of overflow at each state variable:

$$\delta \triangleq \frac{2^{m_i}}{\sigma_i} \quad , \quad i = 1, \ldots, n \quad , \tag{9.23}$$

since 2^{m_i} bounds the magnitude of the i^{th} state-variable representation.

We now find the scaled K_{ii}, $i = 1, \ldots, n$, to satisfy (9.23) for a given δ and m_i. Substituting (9.22) and (9.20) into (9.23) gives

$$\delta = \frac{2^{m_i} \varepsilon_{in}}{\sqrt{K_{ii}}\, \tilde{\sigma}_{in}} \tag{9.24}$$

and substituting (9.20) into (9.21) gives

$$\delta_{in} = \frac{2^m \varepsilon_{in}}{\tilde{\sigma}_{in}} \quad . \tag{9.25}$$

Solving (9.25) for $\varepsilon_{in}/\tilde{\sigma}_{in}$, substituting $\varepsilon_{in}/\tilde{\sigma}_{in}$ into (9.24), and solving for K_{ii} gives

$$\sqrt{K_{ii}} = \left(\frac{\delta_{in}}{\delta}\right) 2^{(m_i - m)} \quad . \tag{9.26}$$

Since $m_i - m$, $i = 1, \ldots, n$, is fixed by (9.19), the result (9.26) indicates that K_{ii} is dependent only on the ratio δ_{in}/δ. If this ratio is fixed at the outset, scaling of the realization need be performed only once. A reasonable choice (used in the examples) is

$$\delta = \delta_{in} \quad , \tag{9.27}$$

since δ and δ_{in} represent constraints on similar quantities. However, it is still possible to vary δ and δ_{in} by the common factor $\varepsilon_{in}/\tilde{\sigma}_{in}$, as seen from (9.24) and (9.25). In other words, we adjust the common probabilities of overflow and overload in the fixed realization by varying the input level to the quantizer or by varying the quantization-step size.

The elements of the required scaling matrix T follow by interpreting (9.4) for diagonal K and T. Assuming that K'_{ii}, $i = 1, \ldots, n$, are the elements of a matrix K' resulting from the initial MRN transformation, we must have

$$\sqrt{K_{ii}} = \frac{\sqrt{K'_{ii}}}{T_{ii}} \quad , \quad i = 1, \ldots, n \, . \tag{9.28}$$

Solving (9.26) and (9.28) for T_{ii} gives

$$T_{ii} = (\frac{\delta}{\delta_{in}}) \sqrt{K'_{ii}}\, 2^{(m-m_i)}, \quad i = 1, \ldots, n. \tag{9.29}$$

Using (9.19) in (9.29), we have

$$T_{ii} = (\frac{\delta}{\delta_{in}}) \; M_g \, W'^{-1/2}_{ii} \quad . \tag{9.30}$$

Therefore, for W the result of scaling will be

$$W_{ii} = M_g^2 \, (\frac{\delta}{\delta_{in}})^2, \quad i = 1, \ldots, n \quad , \tag{9.31}$$

by (9.5). The fact (9.31) that the diagonal elements of W are all equal has a favorable interpretation. The sequences g_i, i = 1, ..., n, which determine w_{ij} by inner products in (5.7), have equal norms by (5.9) and (9.31). This eliminates the following possibility: A small relative variation in a sequence g_i with large norm produces a large variation in an off-diagonal inner product $g_i \cdot g_j$ relative to a small inner product $g_j \cdot g_j$ on the diagonal. In other words, it appears that the diagonal dominance in W will generally suffer less degradation by coefficient quantization of A and C if the diagonal elements of W are at least roughly equal. This hypothesis was confirmed by numerical examples.

D. Quantization/Optimization of A, B, C, and D

The quantization of the matrix A in step 7) consists of two parts, as described in Chapter 7: selection of a roughly quantized starting point; adjustment to meet constraints on eigenvalue variations by

adding a few bits to the starting point. The adjustment uses the branch-and-bound algorithm of integer programming. The use of a CSDC truncation program is effective in obtaining a simple starting point. For example, if a bit span of four bits is allowed, then each coefficient will initially be represented by a form $2^{-p} \pm 2^{-q}$, where $p \neq q$ are integers. On the other hand, if too large a bit span is chosen, all performance criteria will be met immediately at the expense of increased hardware. The adjustment and quantization of B, C, and D to reduce the variation of the zeroes of the transfer function was described in Section 7.4. The simple expedient of greater word lengths in B, C, and D was used where necessary, since this part of the quantization problem has not yet been incorporated into the integer programming approach. Binary numbers are represented in all program computations by decimal equivalents with very high precision; the resulting errors are insignificant.

E. Performance Checks

Frequency response and diagonal dominance are checked in step 10) by direct evaluation using computer programs. The frequency response is found from the Fourier transform of the unit-sample response. The Lyapunov equations (5.2) and (5.3) are solved by programming the method described in Section 7.5.

The output roundoff noise variance σ^2 is given by [C10]

$$\sigma^2 = \frac{nv}{3} \left(\frac{\delta}{2^m}\right)^2 [e\,(K)\, e\,(W)]^{-2/n}\, M_g^2 \quad , \tag{9.32}$$

where all variables except v have been defined previously; see (5.10) for an explanation of e(K) and e(W). v is the potential number of roundoff-noise sources per state variable. For the shift-add structure v = 1 (the minimum), because all roundoff error is concentrated in the final stage, as explained in Chapter 8. Note that Mullis and Roberts [C10] assume the worst case v = n+1 in certain examples. Thus, it may be worthwhile to reassess those examples if the shift-add structure is used. For a given transfer function, average word length, and overflow probability, all variables in (9.32) except e(K) and e(W) are determined. Hence, roundoff noise performance may be assessed by computing the noise multiplication factor

$$F = [e(K)\ \ e(W)\ ]^{-2/n} \quad ; \tag{9.33}$$

the minimum possible value is F = 1. This factor can be used to evaluate any type of state-space realization.

9.3 Computer Language

All programs were written in the Speakeasy computer language, a software product of Speakeasy Computing Corporation, Chicago, Illinois. This is a higher-level language, written on a FORTRAN base, with a very large vocabulary providing functions in many areas of mathematics. Speakeasy has been continually enlarged and upgraded over about fifteen years; the user may also add his own function modules. It runs in either batch or interactive mode on IBM TSO or CMS operating systems; it is avilable at many computer installations.

Some advantages of Speakeasy for the present application are: ease of use; extensive error-detection aids; natural notation for mathematical problems, simplified access to stored information; and processing of matrices and arrays as entities. Any need for the user to think about the computer, rather than his problem, is viewed as a failure in this language. Although it appears to be oriented toward the novice, Speakeasy incorporates many advanced concepts in computer usage.

A complete set of commands and functions is provided for matrix algebra. A few sample statements from Speakeasy illustrate its efficiency: A = MATRIX (N,N:) defines an NxN matrix A (to be subsequently loaded with data); A*B gives the product of two matrices A and B; EIGENVALS (A) gives the set of eigenvalues of A.

9.4 Program Descriptions

A complete listing of computer programs, which implement the design procedure of Section 9.2, appears in Appendix A. These programs are basically modular, to clearly exhibit strategy and allow easy revisions. Speakeasy supports the modular concept by allowing simple, direct calls to other programs during execution and by a common name= referencing of all data objects defined during an operating session. The algorithms of particular significance to the design procedure which were mentioned in Section 9.2 are used in the programs.

Two major programs (named FILTER and COOP) are totally modular. Each consists entirely of GET statements, which retrieve the needed

programs from disk storage, followed by a call to the program which functions as the operating system. The operating system coordinates the execution of all other retrieved programs and interacts with the user. It allows the user to control the decision process at certain points during execution. This is essential to avoid inefficient manual transfers during the design and verification process. Certain large data objects generated during execution, such as bit matrices for CSDC expansions, are code-named and stored on the disk for reuse.

The listed programs have generally descriptive names which appear after the word "EDITING". Many detailed comments are included (following the character $), which explain function, input, and output. A brief description of each program is given below; additional help is provided by the program comments.

1) FILTER (totally modular): runs state-space realization; transforms to MRN realization; makes performance checks.
    a) FILOPSYS: operating system for FILTER.
    b) FILDATA: acquires job specification and system matrix from user.
    c) ITERATE: iterates state equation, generating sequence of outputs and state vectors.
    d) FOURIER: computes frequency response by Fourier transform of unit-sample response.
    e) LYAP: solves Lyapunov matrix equations; checks diagonal dominance; checks roundoff noise multiplier; transforms and scales realization for MRN realization, if requested.

2) TRUNC2: computes bit matrices for CSDC expansion of system matrix; supplies decimal equivalents to truncated CSDC expansions.

3) EIGENAL (partially modular): computes eigenvalues of A; computes variation in eigenvalues due to truncated CSDC expansion; computes sensitivity matrix for eigenvalues.

    a) ORDEREIG: reorders/partitions eigenvalues/vectors into real/complex sets.

    b) MATCHEIG: reorders/matches complex eigenvalues for two nearly equal matrices.

    c) SENSITIV: computes eigenvalue sensitivity matrix.

4) COOP (totally modular): performs coefficient optimization of quantized A by branch-and-bound algorithm (integer programming); requires starting point, eigenvalue sensitivity matrix, bounds on eigenvalue variation, and miscellaneous specifications.

    a) OPERATE: operating system for COOP.

    b) DATAIN: acquires job specification and needed inputs from user.

    c) NEWSPEC: acquires change in job specification from user in the event that solution is found or search exceeds limits.

    d) PRIORITY: dummy program for future development; allows priority ordering of coefficients of A.

    e) SETSTORE: defines needed storage areas for tree and computes needed data for search.

f) NODESEL: selects node to pursue in tree.

g) NODEARBI: branches both ways from selected node; stores new feasible nodes only; stores LP cost.

h) FEASLP: executes LP to determine feasibility and cost of new node; computes constraints for LP based on fixed and free variables of new node.

i) INTERACT: interacts with user regarding deposition of solution or change of search limits, in the event that either is required.

j) CONVERT: converts optimized solution from bit form to decimal equivalent.

5) MODEOP: adjusts (infinite-precision) B, C, and D to restore original zeroes and zero-frequency gain of transfer function.

6) PARTS (service program): separates (partitions) A, B, C, and D from system matrix S.

7) ASSEMBLE (service program): inserts (new) A, B, C, and D into system matrix S.

A simple algorithm was developed for the computation of the CSDC of a real number and used in the program TRUNC2. This algorithm proceeds directly from the MSB to the LSB as it generates the CSDC in a single pass. A common and somewhat inefficient previous approach was: generate the conventional binary code (from the MSB to the LSB); use a logic table on this code in the reverse direction (from the LSB to the MSB) to determine the CSDC bit by bit. Prof. G. O. Martens and M. R. Jarmasz have developed an algorithm to produce the generalized

CSDC representation of an integer in a single pass. However, when specialized to base two, their algorithm is basically different from the procedure given below. In particular, their algorithm proceeds from the LSB to the MSB and uses modulo arithmetic.

In the following algorithm, X is a real number and A(n) is the n^{th} bit of its CSDC representation:

1) Choose the smallest n such that $4/3\ (2^n) > |X|$.

2) If $(4/3)(2^n) > |X| > (2/3)(2^n)$, then A(n) = SGN X.
   Otherwise A(n) = 0.

3) Replace X by $X - A(n)\ (2^n)$.

4) If $A(n) \neq 0$, replace n by n - 2.
   Otherwise replace n by n - 1.

5) Return to step 2).

Step 2) is justified because if $A(n) \neq 0$, the extreme contribution of all bits lying to the right is $\pm\ (1/3)\ 2^n$, under the restriction to no adjacent non-zero bits. If $A(n) \neq 0$, the replacement in step 3) guarantees that the inequality in 2) will not hold for n - 1; this gives A(n-1) = 0, i.e., a code with no adjacent non-zero bits. The algorithm converges since |X| decreases after each replacement in step 3). The resulting code must be the CSDC, because the CSDC has no adjacent non-zero bits and the CSDC is unique.

9.5 Examples

Three examples, 2^{nd}-, 3^{rd}-, and 4^{th}-order Butterworth lowpass filters, were worked by means of the programs described in Section 9.4, to verify the design procedure in Section 9.2. The key results are tabulated here; much supporting data was also generated. Since the objective was to investigate the FWLEs associated with the design procedure, no preference existed for a particular type of frequency-domain approximation. The design procedure imposes no restriction in this respect and types other than Butterworth can be considered. In fact, the roundoff-noise property of the MRN realization shows to greater advantage for transfer functions with very sharp cutoff [C10].

The initial designs for the examples are listed in Tables 1, 2, and 3. Although all coefficients are shown with five significant digits to avoid clutter, ten significant digits were actually retained in the transfer-function calculations. All matrix coefficients were computed and stored in double precision, since Speakeasy automatically performs all calculations in double precision. The off-diagonal terms in K and W are shown as zero, because no such computed term exceeded 2.2 (10^{-15}) in magnitude. The system matrices S give A, B, C, and D by the standard partition

$$S = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] . \qquad (9.34)$$

Transfer Functions:

$$H(s) = \frac{1}{s^2 + 1.41421s + 1}$$

$$H(z) = \frac{.29289 + .58579z^{-1} + .29289z^{-2}}{1 + 0z^{-1} + .17157z^{-2}}$$

Companion-Form Realization:

$$S = \begin{bmatrix} 0 & 1 & 0 \\ -.17157 & 0 & 1 \\ .24264 & .58579 & .29289 \end{bmatrix}$$

$$F = 1.414$$

MRN Realization:

$$S = \begin{bmatrix} .23915 & -.92399 & 1.8138 \\ .24758 & -.23915 & -.15447 \\ .33195 & .10551 & .29289 \end{bmatrix}$$

$$F = 1$$

Solutions to Lyapunov Matrix Equations:

$$K = \begin{bmatrix} 3.7321 & 0 \\ 0 & .26795 \end{bmatrix} \qquad W = \begin{bmatrix} .125 & 0 \\ 0 & .125 \end{bmatrix}$$

Optimal Word Lengths:

i	$m_i - m$
1	.94999
2	-.94988

Table 1. Initial Design: 2^{nd}-Order Butterworth

Transfer Functions:

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

$$H(z) = \frac{.16667 + .5z^{-1} + .5z^{-2} + .16667z^{-3}}{1 + 0z^{-1} + .33333z^{-2} + 0z^{-3}}$$

Companion Form Realization:

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -.33333 & 0 & 1 \\ .16667 & .44444 & .5 & .16667 \end{bmatrix}$$

$F = 2.781$

MRN Realization:

$$S = \begin{bmatrix} .37905 & -.87387 & -.24741 & -2.7849 \\ .37905 & -.13164 & .88975 & .64276 \\ -.015878 & -.13164 & -.24741 & .062072 \\ -.20288 & -.10795 & .070461 & .16667 \end{bmatrix}$$

$F = 1$

Solutions to Lyapunov Matrix Equations:

$$K = \begin{bmatrix} 10.887 & 0 & 0 \\ 0 & 2.0484 & 0 \\ 0 & 0 & .044841 \end{bmatrix} \quad W = \begin{bmatrix} .05778 & 0 & 0 \\ 0 & .05778 & 0 \\ 0 & 0 & .05778 \end{bmatrix}$$

Optimal Word Lengths:

i	$m_i - m$
1	1.7223
2	.51725
3	-2.2395

Table 2. Initial Design: 3rd-Order Butterworth

Transfer Functions:

$$H(s) = \frac{1}{s^4 + 2.61313s^3 + 3.41421s^2 + 2.61313s + 1}$$

$$H(z) = \frac{.093980 + .37592z^{-1} + .56389z^{-2} + .37592z^{-3} + .093980z^{-4}}{1 + 0z^{-1} + .48603\,z^{-2} + 0z^{-3} + .017665z^{-4}}$$

Companion-Form Realization:

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -.017665 & 0 & -.48603 & 0 & 1 \\ .092321 & .37592 & .51821 & .37592 & .093981 \end{bmatrix}$$

$$F = 6.121$$

MRN Realization:

$$S = \begin{bmatrix} .47436 & -.81772 & -.23232 & .21573 & -4.0481 \\ .45607 & -.033872 & .86757 & -.15482 & 1.5773 \\ -.03472 & -.23246 & -.18734 & -.92629 & .31191 \\ -.003085 & -.0039697 & .088639 & -.25315 & -.0066168 \\ -.12102 & -.08455 & .062397 & .013833 & .093981 \end{bmatrix}$$

$$F = 1$$

Solutions to Lyapunov Matrix Equations:

$$K = \begin{bmatrix} 28.965 & 0 & 0 & 0 \\ 0 & 9.01 & 0 & 0 \\ 0 & 0 & .64689 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .0059235 \end{bmatrix} \quad W = \begin{bmatrix} .025888 & 0 & 0 & 0 \\ 0 & .025888 & 0 & 0 \\ 0 & 0 & .025888 & 0 \\ 0 & 0 & 0 & .025888 \end{bmatrix}$$

Optimal Word Lengths:

i	$m_i - m$
1	2.4281
2	1.5858
3	-.31421
4	-3.6997

Table 3. Initial Design: 4^{th}-Order Butterworth

Within the limits of machine accuracy, the matrices S for the MRN realization in Tables 1, 2, and 3 represent the infinite-precision center point for the coefficient optimization program.

The frequency responses for the examples appear in Tables 4, 5, and 6: attenuation in dB versus normalized frequency (i.e., cycles per sample). Attenuation less than .002 dB in magnitude is listed as zero. Each table represents the progression (by column) of one example through the steps of the design process. The center point (column 2) is the target response corresponding to the infinite-precision center-point realization; it is the response of either the infinite-precision companion-form or MRN realization. The starting point (column 3) is the response corresponding to the roughly quantized S matrix of the MRN realization; here a coefficient bit-span of four bits was used, which gives only two non-zero bits per coefficient. The result (column 4) of optimizing A by the branch-and-bound method, to reduce eigenvalue variation, cannot be interpreted properly without adjusting B, C, and D to restore the zeroes of the transfer function and the zero-frequency gain. This is particularly true because B, C, and D for column 4 have been quantized roughly and without regard to the optimization of A. The response (column 5) corresponding to "complete optimization", i.e., A optimized and B, C, and D adjusted to infinite precision values as described above, is a proper indication of the potential effect of optimizing A. Another indication is the reduction of eigenvalue variations, discussed in the next paragraph. The response (column 6) corresponding to the optimization of A and the truncation of the adjusted

Normalized Freq.	ATTENUATION (dB) Center Point	Starting Point (4-bit)	A Optimized	A Opt., B,C,D Adj.	A Opt., B,C,D Quant.	Comparison (8-bit)
0	0	.366	.313	0	0	.0082
.025	0	.366	.316	0	0	.00829
.05	.00273	.370	.328	0	.00285	.0106
.075	.0144	.383	.356	.012	.00780	.0218
.1	.0841	.419	.414	.044	.0403	.0549
.125	.126	.502	.525	.120	.117	.132
.15	.283	.670	.729	.276	.275	.288
.175	.573	.980	1.08	.565	.564	.577
.2	1.07	1.52	1.66	1.06	1.06	1.07
.225	1.85	2.39	2.56	1.85	1.85	1.85
.25	3.01	3.71	3.88	3.01	3.02	3.01
.275	4.59	5.55	5.68	4.61	4.62	4.59
.3	6.62	7.96	8.03	6.64	6.66	6.62
.325	9.08	11.0	10.97	9.12	9.14	9.08
.35	12.0	14.9	14.68	12.0	12.1	12.0
.375	15.4	20.2	19.6	15.5	15.5	15.4
.4	19.58	28.9	27.2	19.6	19.7	19.6
.425	24.8	37.7	38.7	24.9	25.0	24.8
.45	32.0	27.4	28.8	32.1	32.5	31.9
.475	44.2	24.3	25.3	44.2	46.0	43.8
.5	> 100	23.5	24.5	> 100	59.0	72.3

Table 4. Frequency Responses: 2^{nd}-Order Example

Normalized Freq.	ATTENUATION (dB) Center Point	Starting Point (4-bit)	A Optimized	A Opt., B,C,D Adj.	A Opt. B,C,D Quant.	Comparison (8-bit)
0	0	-.114	-.124	0	.0421	-.00384
.025	0	-.114	-.122	0	.0422	-.00375
.05	0	-.111	-.114	0	.0431	-.00340
.075	0	-.105	-.101	.00228	.0449	-.00218
.1	.00511	-.091	-.0786	.00794	.0512	.00274
.125	.0219	-.057	-.0364	.0264	.0704	.0203
.15	.075	.027	.0514	.082	.127	.0746
.175	.224	.227	.245	.234	.280	.224
.2	.596	.681	.677	.609	.658	.597
.225	1.42	1.63	1.58	1.44	1.49	1.43
.25	3.01	3.35	3.27	3.03	3.08	3.01
.275	5.53	6.02	5.94	5.55	5.61	5.54
.3	8.92	9.56	9.53	8.93	8.99	8.94
.325	13.0	13.9	13.9	13.0	13.1	13.0
.35	17.6	18.9	19.0	17.7	17.7	17.7
.375	23.0	24.8	25.2	23.0	23.1	23.1
.4	29.3	31.4	32.0	29.3	29.4	29.5
.425	37.2	35.0	35.6	37.2	37.4	37.6
.45	48.0	34.4	34.9	48.0	47.8	48.9
.475	66.2	33.9	34.5	66.2	55.9	66.3
.5	> 100	33.8	34.4	> 100	56.3	67.5

Table 5. Frequency Responses: 3rd-Order Example

Normalized Freq.	Center Point	ATTENUATION (dB) Starting Point (4-bit)	A Optimized	A Opt. B,C,D Adj.	A Opt. B,C,D Quant.	Comparison (8-bit)
0	0	-.666	-.279	0	.0279	-.0413
.025	0	-.646	-.279	0	.0275	-.0417
.05	0	-.585	-.281	-.002	.0263	-.0429
.075	0	-.483	-.285	-.005	.0241	-.0448
.1	0	-.338	-.291	-.009	.0207	-.0473
.125	.00376	-.149	-.299	-.0139	.0170	-.0479
.15	.0197	.092	-.301	-.0120	.0203	-.0369
.175	.0855	.411	-.268	.0291	.0631	.0237
.2	.325	.893	-.082	.228	.264	.259
.225	1.08	1.78	.606	.936	.974	1.02
.25	3.01	3.62	2.49	2.85	2.89	2.96
.275	6.56	7.00	6.05	6.47	6.52	6.53
.3	11.4	11.9	10.9	11.4	11.5	11.4
.325	17.1	18.0	16.4	17.2	17.2	17.1
.35	23.4	25.2	22.3	23.6	23.6	23.6
.375	30.6	34.3	28.5	30.8	30.7	31.0
.4	39.1	50.6	34.5	39.3	38.9	40.0
.425	49.6	48.7	39.3	49.8	48.2	54.0
.45	64.0	45.4	41.9	64.3	57.4	59.9
.475	88.3	45.4	42.6	88.6	61.9	56.2
.5	> 100	45.6	42.7	> 100	62.3	56.0

Table 6. Frequency Responses: 4^{th}-Order Example

infinite-precision B, C, and D represents the completed design. Here B, C, and D have simply been truncated to a sufficiently large bit-span (8 bits), because this part of the optimization problem has not yet been incorporated into the branch-and-bound method. The response (column 7) corresponding to the direct truncation of the center point realization to a bit-span of eight bits is given for comparison. The frequency responses (column 6) of the completed designs are graphed in Figures 6, 7, and 8. The phase responses and the unit-sample responses of the completed designs are tabulated in Appendices B and C, respectively.

The eigenanalysis of A, which is relevant to the optimization of A, is given in Table 7 for each example. The eigenvalues of the center-point A (column 1) are broken into real and imaginary parts. The variation in each part is listed for the starting point A (column 2) and the optimized A (column 3). It is not necessary to list this information for the conjugate of each eigenvalue, since its content is redundant. It is evident that the optimization has greatly reduced the variations on an overall basis; we are not concerned with a slight increase in variation where, by good fortune, one part of an eigenvalue of the starting point lies extremely close to the corresponding part of the center-point eigenvalue. The important feature is that all variations are sufficiently small.

The change in A due to optimization is seen in Tables 8, 9, and 10, which list the CSDC bit matrices for the final design. A — below a particular bit indicates that it changed during optimiza-

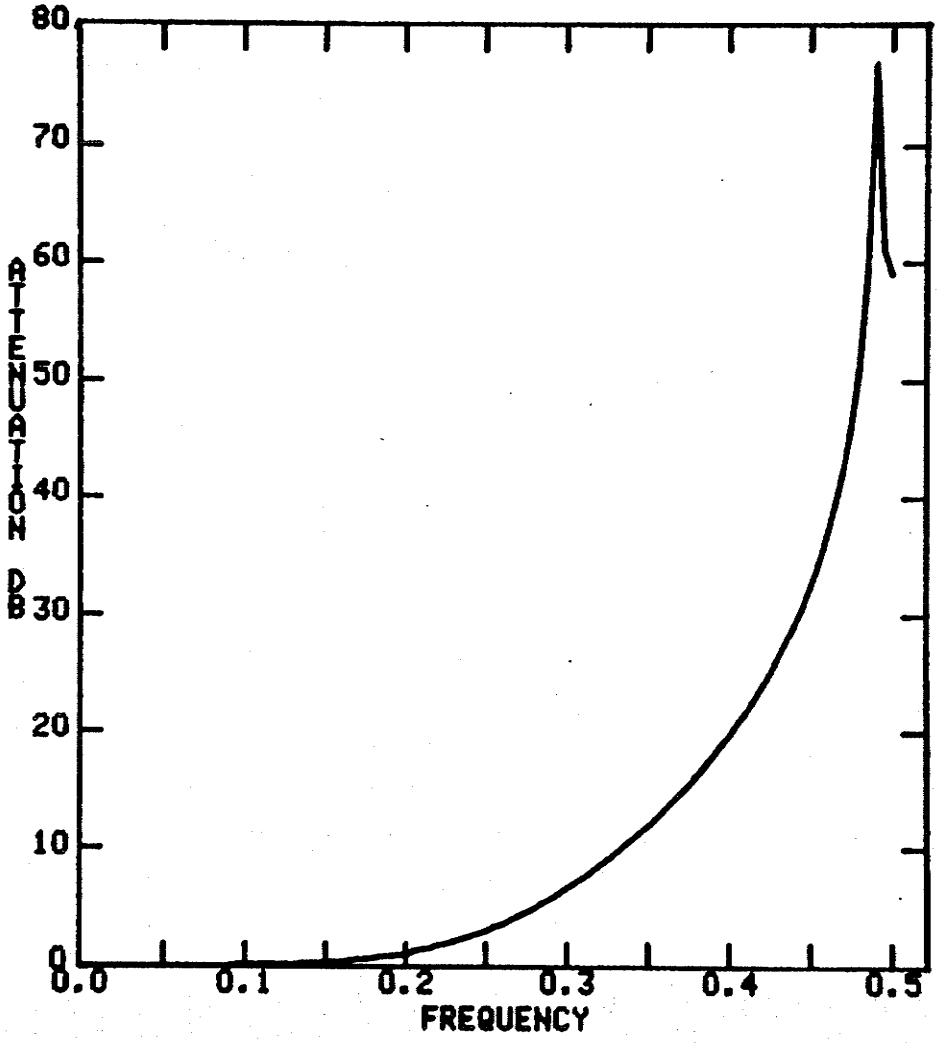


Figure 6. Frequency Response; 2nd-Order Example.

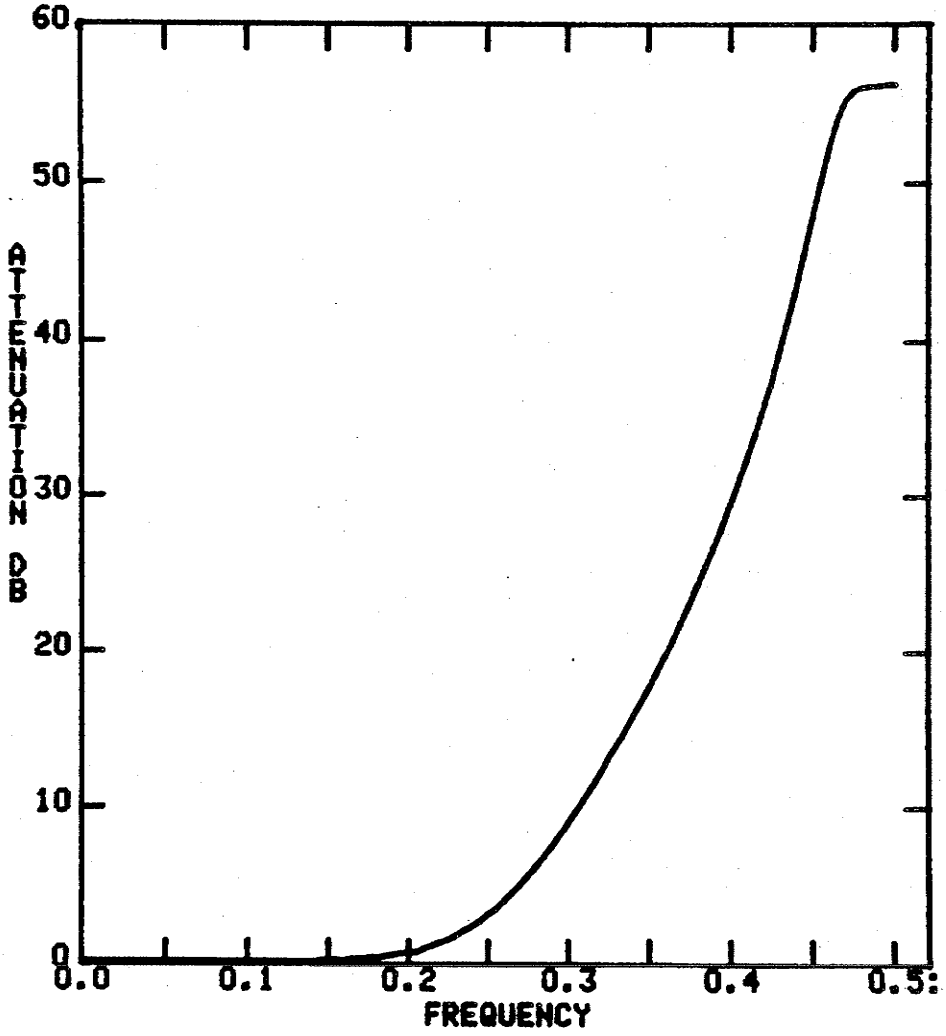


Figure 7. Frequency Response: 3rd-Order Example.

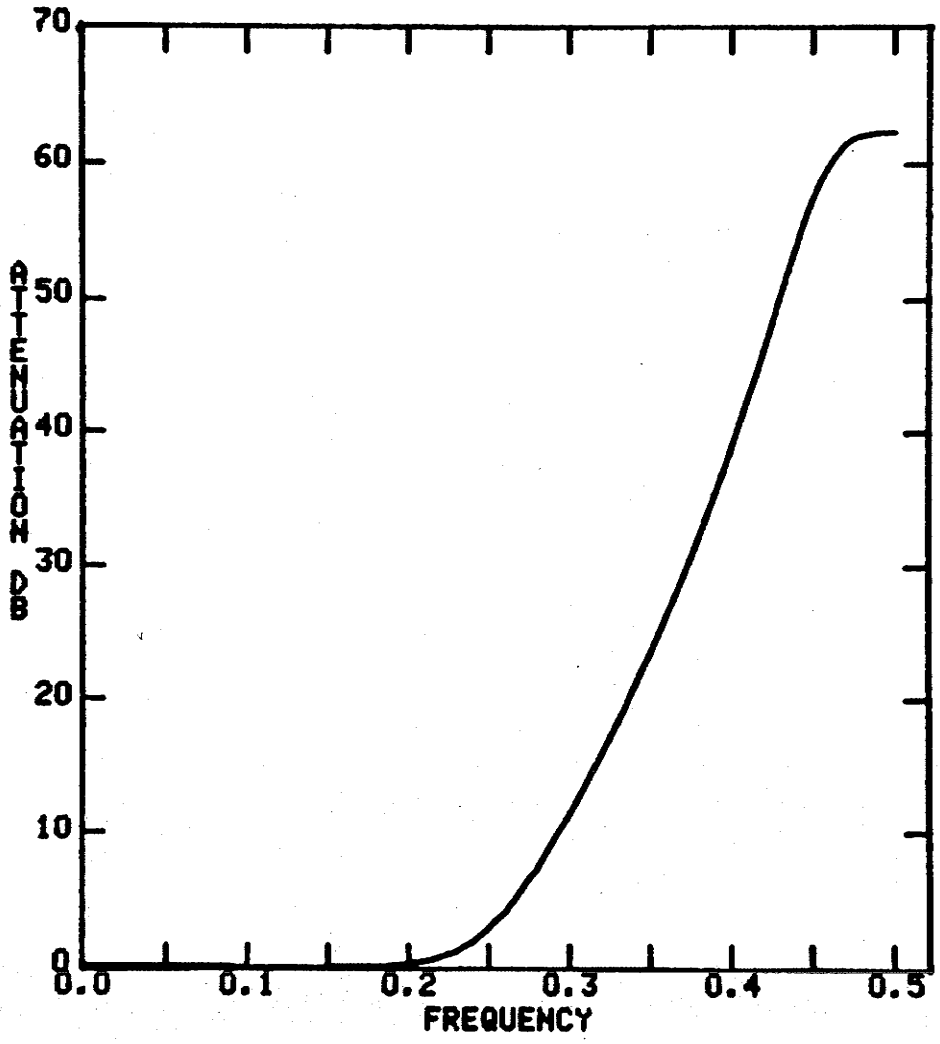


Figure 8. Frequency Response: 4th-Order Example.

		Eigenvalue	Eigenvalue Variation	
		Center Point A	Starting Point A	A Optimized
2^{nd}-Order Example	Real Part	$1.3878(10^{-17})$	0	.0019531
	Imaginary Part	.41421	.018799	.001536
3^{rd}-Order Example	Real	$-4.4357(10^{-17})$	-.015801	.00039581
	Real Part	$-4.1633(10^{-17})$	.0079005	-.0001979
	Imaginary Part	.57735	-.0077838	-.00071833
4^{th}-Order Example	Real Part	$-3.3307(10^{-16})$	.050152	.0071935
	Imaginary Part	.19891	-.037264	.0038138
	Real Part	$9.7145(10^{-17})$	-.03648	.000619
	Imaginary Part	.66818	-.0085301	.0087985

Table 7. Eigenanalysis of A

K_{-1}

$$\begin{vmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

K_0

$$\begin{vmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

K_1

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

K_2

$$\begin{vmatrix} 1 & 0 & -1 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \end{vmatrix}$$

K_3

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix}$$

K_4

$$\begin{vmatrix} 0 & \frac{1}{0} & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{vmatrix}$$

K_5

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{vmatrix}$$

K_6

$$\begin{vmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & -1 \end{vmatrix}$$

K_7

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix}$$

K_8

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{0} & -1 \\ 0 & 0 & -1 \end{vmatrix}$$

K_9

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

K_{10}

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix}$$

(Note: Starting Point Bit Listed Below — if it differs from final design)

Final Design:

$$S = \begin{bmatrix} .25 & -.9375 & 1.8281 \\ .25 & -.24609 & -.14355 \\ .32812 & .10156 & .29297 \end{bmatrix}$$

Starting Point:

$$A = \begin{bmatrix} .25 & -1 \\ .25 & -.25 \end{bmatrix}$$

Table 8. CSDC Bit Matrices with Decimal Equivalent: 2^{nd}-Order Example

$$K_{-2} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad K_{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad K_{0} = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad K_{2} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \quad K_{3} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & -1 & \frac{0}{-1} & 1 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$K_{4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{0} & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad K_{5} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad K_{6} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$K_{7} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad K_{8} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{-1}{0} & 0 & -1 \\ 0 & 0 & \frac{1}{0} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad K_{9} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$K_{10} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad K_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Final Design:

$$S = \begin{bmatrix} .375 & -.875 & -.25 & -3 \\ .375 & -.12891 & .9375 & .63672 \\ -.015625 & -.125 & -.24609 & .0625 \\ -.1875 & -.11133 & .070801 & .16602 \end{bmatrix}$$

Starting Point:

$$A = \begin{bmatrix} .375 & -.875 & -.25 \\ .375 & -.125 & .875 \\ -.015625 & -.125 & -.25 \end{bmatrix}$$

Table 9. CSDC Bit Matrices with Decimal Equivalent: 3rd-Order Example

K_{-2}

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_{-1}

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

K_0

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_1

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_2

$$\begin{bmatrix} 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_3

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 \end{bmatrix}$$

K_4

$$\begin{bmatrix} 0 & \frac{-1}{0} & 0 & 0 & 1 \\ \frac{0}{-1} & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & \frac{1}{0} & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

K_5

$$\begin{bmatrix} 0 & 0 & 0 & -1 & 0 \\ \frac{-1}{0} & -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

K_6

$$\begin{bmatrix} \frac{-1}{0} & 0 & \frac{1}{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{0} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_7

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{0} & 0 \\ 0 & 0 & \frac{1}{0} & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_8

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & \frac{-1}{0} & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

K_9

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Table 10. CSDC Bit Matrices with Decimal Equivalent: 4^{th}-Order Example

K_{10}

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

K_{11}

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_{12}

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

K_{13}

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_{14}

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_{15}

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

K_{16}

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

K_{17}

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

K_{18}

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

Final Design:

$$S = \begin{bmatrix} .48437 & -.8125 & -.23437 & .21875 & -3.9375 \\ .46875 & -.035156 & .875 & -.14844 & 1.4375 \\ -.035156 & -.23437 & -.17969 & -.9375 & .30469 \\ -.0029297 & -.0039062 & .09375 & -.25391 & -.00028992 \\ -.12305 & -.089844 & .060547 & -.00020981 & .09375 \end{bmatrix}$$

Starting Point:

$$A = \begin{bmatrix} .5 & -.75 & -.25 & .21875 \\ .4375 & -.035156 & .875 & -.15625 \\ -.035156 & -.25 & -.1875 & -1 \\ -.0029297 & -.0039062 & .09375 & -.25 \end{bmatrix}$$

Table 10. (Continued)

tion; the value corresponding to the starting point is under the -. Here we should confine our attention to the A portion of the system matrix S, since the B, C, and D matrices were subjected to a totally different adjustment process. For comparison, the decimal equivalent to the truncated CSDC expansions for the final design and for the starting point A is shown in Tables 8, 9, and 10. The CSDC expansion follows from the bit matrices in the following way:

$$S = \sum_i K_i \, 2^{-i} \quad , \tag{9.35}$$

where K_i is the i^{th} bit matrix.

The final performance checks involve diagonal dominance in K and W and roundoff-noise performance. The solutions K, W, and F corresponding to the final designs are listed in Table 11.

9.6 Interpretation of Examples

We now discuss the extent to which the design procedure achieves its original objectives and the possible means of improving it.

It is evident from Tables 4, 5, and 6 that the optimization process produces the desired effect with respect to the frequency response. The starting point (bit-span of 4 bits) represents a 16= fold decrease in precision from the comparison realization (bit-span of 8 bits). However, the appendage of a limited number of non-zero bits to the A matrix, followed by the adjustment and quantization (to 8 bits) of the B, C, and D matrices, produces a frequency response

2^{nd}-Order Example:

$$K = \begin{bmatrix} 3.8045 & .029259 \\ .029259 & .27121 \end{bmatrix} \quad W = \begin{bmatrix} .12284 & -.0024207 \\ -.0024207 & .12471 \end{bmatrix}$$

$$F = 1.0006$$

3^{rd}-Order Example:

$$K = \begin{bmatrix} 12.403 & .053732 & -.017902 \\ .053732 & 2.2086 & -.0072518 \\ -.017902 & -.0072518 & .043718 \end{bmatrix} \quad W = \begin{bmatrix} .050113 & .0014723 & .0014193 \\ .0014723 & .053138 & -.002156 \\ .0014193 & -.002156 & .058894 \end{bmatrix}$$

$$F = 1.0014$$

4^{th}-Order Example:

$$K = \begin{bmatrix} 27.102 & .45404 & -.054541 & -.016995 \\ .45402 & 8.4554 & -.061679 & .013586 \\ -.054541 & -.061679 & .62406 & .0037351 \\ -.016995 & .013586 & .0037351 & .006152 7 \end{bmatrix} \quad F = 1.0063$$

$$W = \begin{bmatrix} .027807 & -.00019642 & .0011921 & .00080074 \\ -.00019642 & .028491 & .00042558 & .0024891 \\ .0011921 & .00042558 & .028652 & -.0016 \\ .00080074 & .0024891 & -.0016 & .02792 \end{bmatrix}$$

Table 11. K, W, and F for Final Design

which is close to that of the comparison realization. For the 4^{th}-order example, the final attenuation (62.3 dB) of the optimized realization is actually greater than that (56.0 dB) of the comparison realization. From Tables 8, 9, and 10, it is seen that 2, 3, and 8 non-zero bits were appended in the 2^{nd}, 3^{rd}, and 4^{th}-order examples, respectively. It appears that applying the integer programming algorithm to the B, C, and D matrices, as discussed in Section 7.4, will provide a further advantage by reducing their bit-spans. Perhaps the most reliable indication of the effect of the optimization algorithm is the extent to which it reduced eigenvalue variations (Table 7).

The deterioration of roundoff-noise performance due to coefficient quantization is negligible. From Table 11, it is seen that the noise multiplication factor F increases from 1 to 1.006, 1.0014, and 1.0063, for the 2^{nd}, 3^{rd}, and 4^{th}-order examples, respectively.

A high degree of diagonal dominance is retained in the matrix W after coefficient quantization, as seen in Table 11. The highest and lowest magnitude ratios between a diagonal element and an element in the same row were 141.6 and 11.21, respectively, from all three examples. This was obtained without any specific constraint on the variation in any element of W; i.e., it followed simply from the constraint on eigenvalue variation and the localized search region which tends to reduce eigenvector variations. On the other hand, the matrix K must have extremely large ratios of magnitudes

on the diagonal (e.g., 4404.9 to 1 in the 4^{th}-order example) because of scaling. The relatively poor diagonal dominance which results for the K matrices in Table 11 appears to be related to these large magnitude ratios, as explained near the end of Section 9.2C.

The application of the diagonal-dominance conditions on W to obtain stability against overflow limit cycles is now considered. We consider the sufficient condition (6.184) for stability of the forced response, which specializes to the sufficient condition (6.64) for zero-input overflow stability if $\alpha = 1$. It is clear that the condition (6.184) can be met by the matrices W in Table 11 for the equal-word-length case and reasonable values of α (e.g., $\alpha = 1/2$). However, the examples in Table 11 derive from the assumption of optimal word lengths, which will determine the weighting factors $(\tilde{R}_i + R_i)/\tilde{R}_j$ in (6.184).

The weighting factors can be analyzed by determining

$$\beta_i \stackrel{\Delta}{=} \frac{R_i}{\tilde{R}_i} , \quad i = 1, 2, \ldots, n , \tag{9.36}$$

which is an indication of the tendency toward overflow under zero-input conditions. It follows from (4.1) and (4.8) that

$$\tilde{R}_i = \sum_{j=1}^{n} |a_{ij}| \tilde{R}_j \tag{9.37}$$

and we will use the close approximation

$$\tilde{R}_i \tilde{=} 2^N 2^{m_i} \tag{9.38}$$

for (4.8). From (9.36), (9.37), and (9.38), we obtain

$$\beta_i = \sum_{j=1}^{n} |a_{ij}| \, 2^{(m_j - m_i)} , \quad i = 1, 2, \ldots, n , \tag{9.39}$$

which is computed for each example and listed in Table 12; the optimal word lengths have been approximated by integers in Table 12, as explained below. By (9.36) and (9.38), we express the weighting factors as

$$\left(\frac{\tilde{R}_i + R_i}{R_j}\right) = (1 + \beta_i) \, 2^{(m_i - m_j)} , \tag{9.40}$$

which gives

$$w_{jj} \geq \frac{2-\alpha}{2\alpha} \sum_{\substack{i=1 \\ i \neq j}}^{n} (1 + \beta_i) \, 2^{(m_i - m_j)} \, |w_{ij}| , \tag{9.41}$$

$j = 1, 2, \ldots, n$, in place of the diagonal-dominance conditions (6.184). The differences $m_i - m_j$ between optimal word lengths follow from the initial design data in Tables 1, 2, and 3, where $m_i - m$ is listed. Clearly,

$$m_i - m_j = (m_i - m) - (m_j - m) \quad ; \tag{9.42}$$

$m_i - m$, $i = 1, 2, \ldots, n$, have been replaced by integers in Table 12, with the intention of maintaining the differences $m_i - m_j$ near their original values. This tends to maintain uniform overflow probabilities at the state variables.

	i	m_i-m	β_i
2^{nd}-Order Example	1	1	.484
	2	-1	1.246
3^{rd}-Order Example	1	2	.8282
	2	1	.9931
	3	-2	1.50
4^{th}-Order Example	1	2	.954
	2	1	1.4154
	3	0	.848
	4	-4	2.071

Table 12. Optimal-Word-Length Approximations and Overflow Parameters

The principal difficulty in satisfying (9.41) is that certain differences $m_i - m_j$ between optimal word lengths may be large enough to produce large weighting factors in (9.40). If $m_i - m$, $i = 1, 2, \ldots, n$, is listed from greatest to least, as in Table 12, then the largest difference $m_i - m_j$ occurs at $i = 1$ and $j = n$ and other large differences may occur in this vicinity. The other extreme, $i = n$ and $j = 1$, or vicinity, gives negative values for $m_i - m_j$, which give weighting factors less than one in (9.40). If we lay out the factors $2^{(m_i - m_j)}$ in an array corresponding to the elements $|w_{ij}|$, then the lower-left-corner region is the area to be examined for potential difficulties.

We now consider the examples in view of the above discussion. Calculations show that the 2^{nd}- and 3^{rd}-order examples will satisfy (9.41) with values $\alpha = .11$ and $\alpha = .75$, respectively. For the 4^{th}-order example, (9.41) is satisfied for $j = 1$, 2, and 3; but (9.41) is not satisfied for $j = 4$ (fourth row), even if $\alpha = 1$ (zero-input case). The multipliers $2^{(m_i - m_j)}$ for the elements of the fourth row are 2^6, 2^5, and 2^4, respectively, which cause (9.41) to be violated by a factor of 18. However, there are several solutions to this problem, which is basically due to the disparity in signal levels at the state variables.

A general remedy for excessive magnitude of $2^{(m_i - m_j)}$ is either to increase m_j or decrease w_{ij}. Increasing m_j increases the headroom for x_j, such that it may also encompass the effects of overflows at other state variables which are input-driven within their dynamic-range constraints. Under this condition, an increase of 2

bits in m_4 of the 4^{th}-order example prevents overflows at x_4, for $\alpha \geq .620$. (This follows by using (6.135) and (6.176) to bound v_i and w_i, respectively; for sufficiently small bounds, (6.143) indicates a non-overflow condition at x_i.) The diagonal-dominance condition (9.41) on row 4 of W then becomes a matter of indifference, because the diagonal-dominance condition on a row of W is active only for an overflow in the corresponding state variable. This is advantageous because it appears that a 4-bit increase in m_4 would otherwise be required to meet the dominance condition (9.41). Decreasing $|w_{ij}|$ may also be feasible, because it is only necessary to concentrate on reducing a few selected $|w_{ij}|$s, while the other $|w_{ij}|$s allow considerable leeway. This could be done by incorporating additional constraints in the optimization algorithm and/or increasing the bit-spans of the coefficients. In this regard, it is necessary to determine if the additional bits are better invested in the signals (i.e., to increase m_j) or the coefficients. Also, there is a certain amount of flexibility in the choice of $\hat{c}_i$ in (7.19), which may be exploited to reduce $|w_{ij}|$.

The source of the problem, i.e., the small magnitude of certain state variables, may also lead to a partial solution. It appears that overflow truncations of these small variables will cause only small increases in the Lyapunov function. Thus, a further analysis by Lyapunov theory might show that any possible overflow limit cycles must be of correspondingly small amplitude. Hence, such limit cycles might be confined to the smaller state variables.

In the case of the wave-digital realization, the optimal word lengths are all equal. This follows by (5.19) and (5.20), which give $K_{ii}\ W_{ii} = 1$, $i = 1, 2, \ldots, n$; inserting this in (9.19) verifies the above claim. Thus, in general, the above weighting problem will be avoided in wave digital filters.

9.7 Hardware Considerations

The examples in Section 9.5 can be implemented digitally by applying the shift-add structure described in Chapter 8. Certain features of the implementations are evident from the pertinent bit matrices in Tables 8, 9, and 10.

The bit matrices are generally sparse in terms of non-zero coefficients. This desirable property is partly due to the reduction of bit-spans in the coefficients of the A-matrices and to the scaling of state variables. Under the sparse conditions, fewer opportunities arise for common sums. Thus, a simple and effective estimate of the required number of adders is the number of non-zero coefficients, which is actually an upper bound on the required number of adders. Table 13 summarizes this; a breakdown is also given for the A-matrix versus the B, C, and D-matrices. This breakdown is appropriate because it demonstrates the result of optimizing the A matrices and the potential for improvement in the unoptimized B, C, and D-matrices. The listed numbers of reductions by obvious common sums is very small. It is expected that the number of sign inversions appearing in the bit matrices will average at about one-half the number of non-zero

	A-Matrix	B, C, D-Matrices	Total	Obvious Reductions by Common Sums
2^{nd}-Order Example	5	19	24	1
3^{rd}-Order Example	15	19	34	3
4^{th}-Order Example	35	22	57	3

Table 13. Number of Non-Zero Coefficients for Final Design

coefficients, due to the random nature of the coefficient-signs and the CSDC. This being the case (cf. Tables 8, 9, and 10), the most generally efficient approach is simply to provide the inverted and non-inverted forms of each state variable and the input; this requires only n + 1 sign inverters. This rule may be violated if the number of common sums becomes significant.

Normally, a separate realization of Block II of the shift-add structure (cf. Fig. 5) is required to realize each row of the S matrix. The number of corresponding non-zero rows in the bit matrices determines the total length of the shift-add chain in Block II. Generally, this number is somewhat less than the total number of coefficient matrices needed to represent all rows of S, because of the scaling effect. The chain may be "padded" by shift blocks alone where coefficient rows are zero, at no expense. The adder word lengths for each Block II should equal the optimal word length for the particular state variable which it computes. Deviation from these word lengths will alter the assumed overflow probabilities. Although this non-uniformity is undesirable when using pre-packaged TTL arithmetic blocks, it should not be a serious problem in LSI, where it would amount to changing the number of mask repetitions. The simplest approach in Block I is to use an adder word length equal to the maximum optimal word length, although various savings in word length may be possible there.

A few general comments conclude this discussion. It is probably best to avoid a pre-occupation with reducing the number of adders under a given performance constraint, since there appears to

be a natural limit on this. Also, adders are relatively inexpensive in LSI. However, since Block II is redundant for each state variable and contains the majority of adders, we may consider the multiplex operation of a single Block II. The maximum repetition rate will decrease by the factor n + 1, but this may be tolerable because the original shift-add structure permits high repetition rates. Also, the reduced coefficient and signal word lengths permit higher repetition rates.

10. CONCLUSION

A viable approach has been given for the design and implementation of state-space digital filters. The emphasis has been on filter performance, i.e., reduction of finite-word-length effects. Although the direct cost in terms of number of adders cannot be avoided, these filters may be desirable in applications requiring high quality.

Many opportunities for additional research have been pointed out, which indicate that the topic lends itself to further growth. The original objectives of low roundoff noise, limit-cycle suppression, and reduced variation in the frequency response have been met. As pointed out in Section 9.6, it is intended to refine the quantization process to obtain greater diagonal dominance over certain off-diagonal elements in W. A summary of the work in this thesis can be obtained by re-reading Sections 1.2, 1.3, 2.5, and 4.7.

APPENDIX A

PROGRAM LISTINGS

```
EDITING FILTER
  1 PROGRAM
  2 $ ** BYPASS GETS, IF DONE IN PREVIOUS ENTRY
  3 ONERROR CONTINUE
  4 TERMINAL OFF
  5 IF (FILGOT.EQ.1) GO TO TERMINAL
  6 $ ** GET REQUIRED SUBPROGRAMS
  7 GET FILOPSYS
  8 GET ITERATE
  9 GET FOURIER
 10 GET LYAP
 11 GET PARTS
 12 GET ASSEMBLE
 13 GET FILDATA
 14 FILGOT=1
 15 TERMINAL: TERMINAL ON
*16 EXECUTE FILOPSYS

EDITING FILOPSYS
  1 PROGRAM
  2 $ ************   OPERATING SYSTEM     **************
  3 $ ***  QUERY USER FOR OPERATING CONDITIONS   *****
  4 CHANS="Y"
  5 NEWJOB="Y"
  6 ASKCHAR "INITIAL ENTRY (NEW S/JOB SPEC.) ?","INITENT="
  7 IF (INITENT.EQ."Y") GO TO FILDATA
  8 CHANGES:
  9 ASKCHAR "CHANGE S ?", "CHANS="
 10 ASKCHAR "NEW JOB SPEC. ?","NEWJOB="
 11 FILDATA: EXECUTE FILDATA
 12 SS=S; L=100
 13 EXECUTE PARTS
 14 IF (PRINSYS.EQ."Y") PRINT A,B,C,D
 15 IF (PRINOUT.EQ."Y".OR.PRINFREQ.EQ."Y") EXECUTE ITERATE
 16 IF (PRINFREQ.EQ."Y") EXECUTE FOURIER
 17 IF (PRINKW.EQ."Y") EXECUTE LYAP
 18 ASKCHAR "RE-RUN PROG. W/CHANGES ?","RERUN="
*19 IF (RERUN.EQ."Y") GO TO CHANGES

EDITING FILDATA
  1 PROGRAM
  2 IF (CHANS.NE."Y") GO TO JOBSPEC
  3 PRINT "NAME SYSTEM MATRIX"
  4 REQUEST S
  5 N=NOROWS(S)-1
  6 JOBSPEC: IF (NEWJOB.NE."Y") GO TO FINISH
  7 ASKCHAR "PRINT A,B,C,D ?", "PRINSYS="
  8 ASKCHAR "PRINT OUTPUT SEQUENCE ?", "PRINOUT="
  9 ASKCHAR "PRINT FREQ. RESP. ?","PRINFREQ="
```

```
 10 ASKCHAR "PRINT/ANALYZE K,W ?","PRINKW="
*11 FINISH:

EDITING ITERATE
  1 PROGRAM
  2 $ ** ITERATE STATE EQUATIONS FOR SPECIFIED A, B, C, D
  3 $ ** AND INPUT U. CREATES: INDEXED OUTPUTS (Y) AND
  4 $ ** STATE VECTORS (X).
  5 PRINT "SPECIFY NO. OF ITERATIONS (KMAX)"
  6 REQUEST KMAX
  7 X = MATRIX (N, KMAX + 2:)
  8 STATINIT=VECTOR(N:)
  9 U = VECTOR (KMAX + 1:1)
 10 Y=VECTOR(KMAX+1)
 11 ASKCHAR "RETAIN UNIT-SAMPLE INPUT ?", "CS = "
 12 IF (CS .EQ. "Y") GO TO CONTINUE
 13 REQUEST U
 14 CONTINUE: ASKCHAR "RETAIN ZERO INITIAL STATE ?", "CS = "
 15 IF (CS .EQ. "Y") GO TO ITERATE
 16 REQUEST STATINIT
 17 X( ,1 ) = STATINIT
 18 ITERATE: $ ITERATE STATE EQUATIONS
 19 FOR K = 1, KMAX + 1, 1
 20 X( , K+1 ) = A * X( , K ) + B * U( K )
 21 Y( K ) = C * X( , K ) + D * U( K )
 22 ENDLOOP K
 23 K = INTEGERS ( 0, KMAX )
*24 IF (PRINOUT.EQ."Y") TABULATE K, Y

EDITING FOURIER
  1 PROGRAM
  2 $ ** COMPUTES FOURIER TRANSFORM H(EXP(JW)) OF SEQUENCE
  3 $ ** Y(K), K=1,...,KMAX. FREQ.: F=CYCLES/SAMPLE (I.E.,
  4 $ ** NORMALIZED RELATIVE TO SAMPLING FREQ.) W/ RANGE OF
  5 $ ** 0 TO .5 IN NO. OF INCREMENTS SPECIFIED BY USER.
  6 $ ** COMPUTED VARIABLES (W.R.T. H): HR/HI=REAL/IMAG. PARTS,
  7 $ ** RESPEC.; MAG = ABS MAG.; ATTENDB= DB MAG. OF RECIPROCAL;
  8 $ ** PHASE= INV. TAN (HI/HR).
  9 PRINT "SPECIFY NO. OF INCREMENTS ON FREQ. SCALE"
 10 REQUEST NUMBER
 11 INC = NUMBER
 12 $ DEFINE TRANSFORM STORAGE ARRAYS
 13 HI = VECTOR ( INC + 1: )
 14 HR = VECTOR ( INC + 1: )
 15 MAG = VECTOR ( INC +1: )
 16 PHASE = VECTOR ( INC + 1: )
 17 ATTENDB = VECTOR ( INC + 1: )
 18 $ INDEX FREQUENCY
 19 FOR I = 0, INC
 20 PI = 3.1415926
 21 F = I * ( .5/INC )
 22 $ DEFINE RADIAN FREQUENCY (PER SAMPLE)
 23 OMEGA = 2 * PI * F
 24 $ DEFINE DISCRETE TIME RANGE
```

```
 25 KRANGE = INTEGERS ( 0, KMAX )
 26 $ DEFINE BASIS FUNCTIONS ON KRANGE
 27 BACOS = COS ( OMEGA * KRANGE )
 28 BASIN = -SIN ( OMEGA * KRANGE )
 29 BACOS=VFAM(BACOS)
 30 BASIN=VFAM(BASIN)
 31 $ COMPUTE COMPONENTS OF SEQUENCE Y , IN BASES
 32 HR ( I + 1 ) = INNER ( Y, BACOS )
 33 HI ( I + 1 ) = INNER ( Y, BASIN )
 34 $ COMPUTE MAGNITUDE
 35 MAG ( I + 1 ) = ( (HR(I+1))**2 + (HI(I+1))**2 ) ** .5
 36 IF (MAG(I+1).LT.(.00001)) GO TO BAILOUT
 37 ATTENDB ( I + 1 ) = (-20) * LOG10 (MAG(I+1))
 38 GO TO PHASE
 39 BAILOUT: ATTENDB(I+1) = 100
 40 PHASE: PHASE ( I + 1 ) = ATAN2 ( HI(I+1),HR(I+1) )
 41 $ CONVERT RADIANS TO DEGREES
 42 PHASE(I+1) = PHASE(I+1) * 57.296
 43 IF (PHASE(I+1).GT.0) PHASE(I+1) = PHASE(I+1) - 360
 44 ENDLOOP I
 45 FREQ = GRID ( 0, .5, .5/INC )
 46 TABULATE FREQ, ATTENDB, PHASE
 47 ASKCHAR "PRINT REAL/IMAG./ABS. FREQ. RESP. ?","TAB="
*48 IF (TAB.EQ."Y") TABULATE FREQ, HR, HI, MAG

EDITING LYAP
  1 PROGRAM
  2 $ ** SOLVES FOR K,W IN LYAPUNOV MATRIX EQUATIONS. PERFORMS
  3 $ ** ROUNDOFF NOISE ANALYSIS. OPTION: APPLY SIMILARITY TRANS-
  4 $ ** FORMATION TO SYSTEM TO DIAGONALIZE K,W; SCALE SYSTEM.
  5 DOMAIN COMPLEX
  6 $ W - A'WA = C'C
  7 $ COMPUTE W
  8 AA = A
  9 CC = C
 10 BRANCH = 1
 11 GO TO COMPUTE
 12 W: W = M
 13 $ K - AKA' = BB'
 14 $ COMPUTE K
 15 AA = TRANSPOSE (A)
 16 CC = TRANSPOSE (B)
 17 BRANCH = 2
 18 GO TO COMPUTE
 19 K: K = M
 20 PRINT K,W
 21 $ COMPUTE STORAGE EFFICIENCY
 22 DK = DET (K)
 23 DEK = DIAGELS (K)
 24 PDEK = PROD (DEK)
 25 EK = (DK/PDEK)**.5
 26 $ COMPUTE QUANTIZATION EFFICIENCY
 27 DW = DET (W)
 28 DEW = DIAGELS (W)
```

```
29 PDEW = PROD (DEW)
30 EW = (DW/PDEW)**.5
31 $ COMPUTE NOISE MULTIPLIER (NM)
32 NM = (EK*EW)**(-2/N)
33 $ COMPUTE ROUNDOFF NOISE INVARIANT
34 MG = (DK*DW)**(.5/N)
35 $ OUTPUT RESULTS
36 PRINT "STORAGE EFFICIENCY"
37 PRINT EK
38 PRINT "QUANTIZATION EFFICIENCY"
39 PRINT EW
40 PRINT "NOISE MULTIPLIER"
41 PRINT NM
42 PRINT "ROUNDOFF-NOISE INVARIANT"
43 PRINT MG
44 ASKCHAR "DIAGONALIZE K AND W ?" , "CS="
45 IF (CS.EQ."N") GO TO RETURN
46 $ SYSTEM TRANSFORMATIONS
47 EIGW = EIGENSYSTEM(W,EVECW)
48 TERMINAL OFF
49 EIGWROOT = SQRT(EIGW)
50 TERMINAL ON
51 $ CONVERT VECTOR EIGWROOT TO DIAGONAL MATRIX DD
52 DD = MATRIX (N,N:)
53 DD = DD + EIGWROOT
54 DD = INVERSE (DD)
55 $ COMPUTE FIRST TRANSFORMATION MATRIX (T)
56 T = EVECW * DD
57 $ EXECUTE FIRST TRANSFORMATION (MATRIX T)
58 WW = (TRANSPOSE(T)) * W * T
59 KK = (INVERSE(T))* K * TRANSPOSE (INVERSE(T))
60 $ COMPUTE COMPOSITE TRANSFORMATION (MATRIX TT)
61 EIGK = EIGENSYSTEM (KK, EVECKK)
62 TT = T * EVECKK
63 $ CHECK TRANSFORMATION
64 K = (INVERSE(TT)) * K * TRANSPOSE(INVERSE(TT))
65 W = TRANSPOSE (TT) * W * TT
66 $ SYSTEM TRANSFORMATION
67 A = (INVERSE(TT)) * A * TT
68 B = (INVERSE(TT)) * B
69 C = C * TT
70 D = D
71 $ OUTPUT
72 PRINT "TRANSFORMED SYSTEM"
73 PRINT A,B,C,D
74 PRINT K,W
75 PRINT "DEVIATION BETWEEN OPTIMAL AND AVERAGE WORDLENGTH"
76 PRINT "LISTED (IN ORDER) FOR STATE VARIABLES 1 THROUGH N"
77 DEV = VECTOR (N:)
78 FOR I = 1 ,N
79 DEV(I) = (-LOG10 (MG) + .5 *LOG10(K(I,I)*W(I,I)))/LOG10(2)
80 ENDLOOP I
81 TABULATE DEV
82 $ COMPUTE SCALING TRANSFORMATION (MATRIX SS)
```

```
  83 SSS = DIAGELS (W)
  84 SSS = MFAM(SSS)
  85 SS = MATRIX(N,N:)
  86 SS = SS + SSS
  87 SS = INVERSE (SS)
  88 TERMINAL OFF
  89 SS = SQRT(SS)
  90 TERMINAL ON
  91 SS = MG * SS
  92 $ SCALE: SYSTEM TRANSFORMATION
  93 A = (INVERSE(SS)) * A * SS
  94 B = (INVERSE(SS)) * B
  95 C = C * SS
  96 D = D
  97 K = (INVERSE(SS)) * K * TRANSPOSE(INVERSE(SS))
  98 W = TRANSPOSE(SS) * W * SS
  99 $ OUTPUT
 100 PRINT "SCALED TRANSFORMED SYSTEM"
 101 PRINT A,B,C,D
 102 PRINT K,W
 103 $ RELOAD S
 104 EXECUTE ASSEMBLE; S=SS
 105 PRINT " "
 106 PRINT "** REMEMBER ** IF YOU WANT TO SAVE NEW S: "
 107 PRINT "               GIVE IT A (CODED) 'NAME' "
 108 PRINT "               THEN USE COMMAND KEEP 'NAME' "
 109 RETURN: RETURN
 110 COMPUTE:
 111 EIGVAL = EIGENVAL (AA, T)
 112 CCC = CC * T
 113 QQ = CCC ** CCC
 114 MM = MATRIX(N,N:)
 115 FOR I = 1,N
 116 FOR J = 1,N
 117 MM(I,J) = QQ(I,J)/(1-EIGVAL(I)*EIGVAL(J))
 118 ENDLOOP J
 119 ENDLOOP I
 120 M = TRANSPOSE(INVERSE(T)) * MM * INVERSE(T)
 121 M = REAL (M)
 122 IF (BRANCH.EQ.1) GO TO W
*123 IF (BRANCH.EQ.2) GO TO K

EDITING TRUNC2
   1 PROGRAM
   2 ASKCHAR "ARE DESIRED CSDC BIT MATRICES NOW IN PROGRAM ?","REENT="
   3 IF (REENT.EQ."Y") GO TO REENT
   4 ASKCHAR "WILL YOU SUPPLY CSDC BIT MATRICES DIRECTLY ?","SUPP="
   5 IF (SUPP.NE."Y") GO TO REQUESTS
   6 REQUEST BITMAT
   7 N = NOROWS (BITMAT) -1
   8 BITS = (NOCOLS(BITMAT))/(N+1)-3
   9 $ GENERATE C-3,...,C0,C1,C2,... FROM BITMAT
  10 INT = INTEGERS (1,N+1)
  11 FOR I=-3,BITS-1
```

```
  12 OBJECT ("C",I) = BITMAT (INT,INT+(N+1)*(I+3))
  13 ENDLOOP I
  14 GO TO REENT
  15 REQUESTS: PRINT "SYSTEM MATRIX?"
  16 REQUEST S
  17 N = NOROWS(S) -1
  18 PRINT "SPECIFY MAXIMUM NUMBER OF BITS IN CSDC EXPANSION"
  19 REQUEST BITS
  20 $  ***************************************************************
  21 $ COMPUTE CSDC BIT-COEFFICIENT MATRICES (C-3,...,C0,C1,C2,...)
  22 M = S
  23 TERMINAL OFF
  24 TEST = ABS(M).GE.(32/3)
  25 TERMINAL ON
  26 IF (MAX(TEST).NE.0) PRINT "EXCESSIVE INPUT MAGNITUDE"
  27 IF (MAX(TEST).NE.0) GO TO FINISH
  28 BITMAT = MATRIX (N+1,(N+1)*(BITS+3))
  29 OP = MATRIX(N+1,N+1:1)
  30 OP = CUMSUM (OP)
  31 FOR I = -3, BITS - 1
  32 OBJECT ("C",I) = MATRIX (N+1,N+1:)
  33 WHERE ((M-OP*2**(-I).LE.OP*2**(-I)/3).AND.(M-OP*2**(-I).GE.-OP*2**(
-I)/3)) OBJECT ("C",I)=1
  34 WHERE ((M+OP*2**(-I).LE.OP*2**(-I)/3).AND.(M+OP*2**(-I).GE.-OP*2**(
-I)/3)) OBJECT ("C",I)=-1
  35 M = M - (2**(-I)) * OBJECT ("C",I)
  36 BITMAT (1,1+(N+1)*(I+3)) = OBJECT ("C",I)
  37 ENDLOOP I
  38 $  ***************************************************************
  39 REENT:
  40 $ SPECIFY, BUILD, AND APPLY MASKS AND ASSEMBLE BITMATM
  41 $ GENERATE MATRIX (LSBLOC) OF LSB ADDRESSES
  42 ASKCHAR "CREATE OR MODIFY MASKED CSDC BIT MATRICES ?","MASK="
  43 IF (MASK.NE."Y") GO TO COMP
  44 PRINT "SPECIFY MASK LENGTH PER COEFFICIENT"
  45 REQUEST LENGTH
  46 BITMATM = MATRIX (N+1,(N+1)*(BITS+3))
  47 OBJECT ("MASCUM",-4)= MATRIX (N+1,N+1:)
  48 MSBLOCS=MATRIX(N+1,N+1:-1*(BITS+1))
  49 MSBLOCS=CUMSUM(MSBLOCS)
  50 FOR I =-3, BITS - 1
  51 OBJECT ("CM",I) = OBJECT ("C",I)
  52 $ GENERATE PRESENT AND CUMULATIVE MASKS
  53 OBJECT ("MASK",I) = MATRIX (N+1,N+1:)
  54 WHERE (OBJECT ("C",I)) OBJECT ("MASK",I) = 1
  55 WHERE(OBJECT("MASK",I).AND.MSBLOCS.EQ.-1*(BITS+1))MSBLOCS=-I
  56 OBJECT ("MASCUM", I) = OBJECT ("MASCUM",I-1).OR.OBJECT ("MASK",I)
  57 L = I - LENGTH
  58 IF (L.LT.-4) L=-4
  59 WHERE (OBJECT ("MASCUM",L))OBJECT ("CM",I) = 0
  60 BITMATM(1,1+(N+1)*(I+3)) = OBJECT ("CM",I)
  61 ENDLOOP I
  62 SPANMAT=MATRIX(N+1,N+1:LENGTH)
  63 SPANMAT=CUMSUM(SPANMAT)
```

```
  64 LSBLOC=MSBLOCS-SPANMAT
  65 $  ************************************************************
  66 COMP: ASKCHAR "DESIRE REAL-NO. EQUIVS. TO CSDC EXPANSION ?","COMP="
  67 IF (COMP.NE."Y") GO TO PRINT
  68 ASKCHAR "COMPUTE FROM MASKED CSDC MATRICES ?","SMASC="
  69 IF (SMASC.NE."Y") GO TO TRUNC
  70 SMASK = MATRIX (N+1,N+1:)
  71 FOR I = -3, BITS - 1
  72 SMASK =SMASK + (2**(-I))*OBJECT ("CM",I)
  73 ENDLOOP I
  74 TRUNC: ASKCHAR "COMPUTE FROM UNMASKED CSDC MATRICES ?","SUNMAS="
  75 IF (SUNMAS.NE."Y") GO TO SELECT
  76 SUNMASK = MATRIX (N+1,N+1:)
  77 PRINT "SPECIFY NUMBER OF BIT MATRICES TO USE"
  78 REQUEST NUMBER
  79 FOR I = -3, NUMBER - 1
  80 SUNMASK = SUNMASK + (2**(-I))*OBJECT ("C",I)
  81 ENDLOOP I
  82 $  ************************************************************
  83 SELECT: ASKCHAR "INSERT ORIGINAL B,C,D MATRICES IN REAL-NO. EQUIV ?",
"INSERT="
  84 IF (INSERT.NE."Y") GO TO PRINT
  85 INT = INTEGERS (1,N+1)
  86 SMASK (N+1,INT) = S(N+1,INT)
  87 SMASK (INT, N+1) = S(INT,N+1)
  88 SUNMASK (N+1,INT) = S(N+1,INT)
  89 SUNMASK (INT,N+1) = S(INT,N+1)
  90 $  ************************************************************
  91 PRINT:
  92 ASKCHAR "PRINT UNMASKED CSDC BIT MATRICES ?","PRINT1="
  93 IF (PRINT1.NE."Y") GO TO PRINT2
  94 PRINT "SUNMASK=SUMMATION (2**-I)CI FOR I=-3,...,NUMBER-1"
  95 FOR I = -3, BITS-1
  96 PRINT "****************************"
  97 PRINT OBJECT ("C",I)
  98 ENDLOOP I
  99 PRINT2: ASKCHAR "PRINT  MASKED CSDC BIT MATRICES ?","PRINT2="
 100 IF (PRINT2.NE."Y") GO TO MESSAGE
 101 PRINT "SMASK=SUMMATION (2**-I)CMI FOR I=-3,...,BITS-1"
 102 FOR I = -3, BITS-1
 103 PRINT "****************************"
 104 PRINT OBJECT ("CM",I)
 105 ENDLOOP I
 106 $  ************************************************************
 107 MESSAGE:
 108 ASKCHAR "DO YOU WANT OPERATIONS MESSAGE ?", "MESS="
 109 IF (MESS.NE."Y") GO TO FINISH
 110 PRINT "PROGRAM TRUNC CREATED FOLLOWING OBJECTS IN TEMPORARY STORAGE
"
 111 PRINT " SMASK/SUNMASK- MAY BE INPUT TO PROGRAMS FILTER/COOP "
 112 PRINT "                MAY BE (CODE) NAMED AND SAVED (KEPT)"
 113 PRINT " BITMAT/BITMATM- MAY BE SAVED (KEPT)"
 114 PRINT " LSBLOC-LSB ADDRESSES FOR PROGRAM COOP "
*115 FINISH:
```

```
EDITING EIGENAL
  1 PROGRAM
  2 $ ** BYPASS GETS, IF DONE IN PREVIOUS ENTRY  **
  3 ONERROR CONTINUE
  4 TERMINAL OFF
  5 IF (EIGGOT.EQ.1) GO TO TERMINAL
  6 TERMINAL ON
  7 GET PARTS
  8 GET ORDEREIG
  9 GET MATCHEIG
 10 GET SENSITIV
 11 EIGGOT=1
 12 TERMINAL: TERMINAL ON
 13 PRINT "NAME SYSTEM MATRIX (INF. PREC. CENTER POINT)"
 14 REQUEST S
 15 L=0; SS=S; EXECUTE PARTS
 16 PRINT "NAME SYSTEM MATRIX (FIN. PREC. STARTING POINT)"
 17 REQUEST START
 18 L=1; SS=START; EXECUTE PARTS
 19 N=NOROWS(S)-1
 20 I=INTEGERS(1,N)
 21 $ ********* COMPUTE EIGENVALUES/VECTORS ***************
 22 DOMAIN COMPLEX
 23 EIGA0=EIGENVALS(A0:U)
 24 EIGAT=EIGENVALS(TRANSPOSE(A0):V)
 25 EIGA1=EIGENVALS(A1:U1)
 26 $ ********* ORDER EIGENVALUES/VECTORS *****************
 27 EIG=EIGA0; UU=U
 28 EXECUTE ORDEREIG
 29 EIGA0=EIG; U=UU; NRA0=NREAL
 30 EIG=EIGAT; UU=V
 31 EXECUTE ORDEREIG
 32 EIGAT=EIG; V=UU; NRAT=NREAL
 33 EIG=EIGA1; UU=U1
 34 EXECUTE ORDEREIG
 35 EIGA1=EIG; U1=UU; NRA1=NREAL
 36 $ ********* MATCH EIGENVALUES/VECTORS ****************
 37 EIG1=EIGA0; UU1=U; NR1=NRA0
 38 EIG2=EIGAT; UU2=V; NR2=NRAT
 39 EXECUTE MATCHEIG; EIGAT=EIG2; V=UU2
 40 EIG2=EIGA1; UU2=U1; NR2=NRA1
 41 EXECUTE MATCHEIG; EIGA1=EIG2; U1=UU2
 42 $ ********* COMPUTE CHANGES IN EIGENVALUES ***********
 43 CHANGE= EIGA1-EIGA0
 44 IF (NRA0.EQ.0) GO TO COMPLEX
 45 IREAL1=INTEGERS(1,NRA0)
 46 CHANGE(IREAL1)=REAL(CHANGE(IREAL1))
 47 COMPLEX:
 48 IREAL2=INTEGERS(NRA0+1,N,2)
 49 IIMAG=1+IREAL2
 50 CHANGE(IREAL2)=REAL(CHANGE(IREAL2))
 51 CHANGE(IIMAG)=-1*IMAG(CHANGE(IIMAG))
 52 ONE=VECTOR(N:1)
```

```
 53 ONE=CUMSUM(ONE)
 54 TYPE=VECTOR(N:)
 55 TYPE(IIMAG)=ONE(IIMAG)
 56 PRINT "CHANGES IN EIGENVALUES"
 57 PRINT "FOR COMPLEX EIGVALS: REAL/IMAG. CHANGES LISTED"
 58 PRINT "TYPE 0 = REAL CHANGE/ TYPE 1 = IMAG. CHANGE"
 59 TABULATE EIGA0, CHANGE, TYPE
 60 ASKCHAR "COMPUTE SENSITIVITY MATRIX ?", "ANS="
 61 IF (ANS.EQ."N") GO TO FINISH
 62 $ *********COMPUTE SENSITIVITY MATRIX **************
 63 U=U; V=V; NREAL=NRA0
 64 EXECUTE SENSITIV; SENMAT=SENMAT
*65 FINISH:

EDITING ORDEREIG
  1 PROGRAM
  2 $ INPUTS: VECTOR OF EIGENVALUES (EIG)
  3 $         MATRIX OF EIGENVECTORS (UU)
  4 $ OUTPUTS:EIG,UU ORDERED BY REAL/COMPLEX EIGVALS
  5 $        COMP PAIR: ORDER IS +/- IMAG PART
  6 $          NREAL= NO. REAL EIGENVALUES
  7 $ INITIALIZE
  8 DOMAIN COMPLEX
  9 N=NOELS(EIG)
 10 I=INTEGERS(1,N); IR=I; IC=I
 11 EIGI=IMAG(EIG)
 12 TERMINAL OFF
 13 CONSTRAIN(IR:ABS(EIGI).LT.1E-20)
 14 CONSTRAIN(IC:ABS(EIGI).GE.1E-20)
 15 TERMINAL ON
 16 NREAL=NOELS(IR)
 17 NCOMP=NOELS(IC)
 18 IF (NREAL.EQ.0) GO TO COMPLEX1
 19 EIGR=REAL(EIG(IR))
 20 VECR=UU(I,IR)
 21 EIGR=RANKED(EIGR:IR)
 22 VECR=VECR(I,IR)
 23 COMPLEX1: IF (NCOMP.EQ.0) GO TO REAL
 24 EIGC=EIG(IC)
 25 VECC=UU(I,IC)
 26 REAL: IF (NREAL.EQ.0) GO TO COMPLEX2
 27 EIG(1)=EIGR
 28 UU(1,1)=VECR
 29 COMPLEX2: IF (NCOMP.EQ.0) GO TO FINISH
 30 IF (MODULO(NCOMP,2).NE.0) PRINT "NO CONJ OF COMP EIG!"
 31 ALLROWS=INTEGERS(1,N)
 32 ODDINT=INTEGERS(1,NCOMP,2)
 33 POSIMAG=INTEGERS(1,NCOMP)
 34 TERMINAL OFF
 35 CONSTRAIN(POSIMAG:IMAG(EIGC(POSIMAG)).GE.1E-20)
 36 TERMINAL ON
 37 EIGC(ODDINT)=EIGC(POSIMAG)
 38 EIGC(1+ODDINT)=CONJUGATE(EIGC(ODDINT))
 39 VECC(ALLROWS,ODDINT)=VECC(ALLROWS,POSIMAG)
```

```
 40 VECC(ALLROWS,1+ODDINT)=CONJUGATE(VECC(ALLROWS,ODDINT))
 41 EIG(NREAL+1)=EIGC
 42 UU(1,NREAL+1)=VECC
*43 FINISH:

EDITING MATCHEIG
  1 PROGRAM
  2 $ INPUTS: EIG1(VALS)/UU1(VECS)/NR1(NO. REAL);EIG2/UU2/NR2;
  3 $         ASSUMED REAL EIGVALS LISTED FIRST AND MATCHED
  4 $        ASSUMED COMP PAIR: ORDER IS +/- IMAG PART
  5 $ OUTPUTS: SAME EIGSETS; BUT SET 2 HAS COMPLEX PART REORD-
  6 $         RED TO MATCH SET 1; ERROR MESSAGE IF NR1.NE.NR2
  7 N1=NOELS(EIG1); N2=NOELS(EIG2)
  8 IF (NR1.NE.NR2.OR.N1.NE.N2) PRINT "CAN'T MATCH NO. REAL EIG"
  9 I=INTEGERS(1,N1)
 10 DOMAIN COMPLEX
 11 $ ** TEST ONLY COMP. EIGS W/POS. IMAG. PART (USE INDEX IC) **
 12 IC=INTEGERS(NR1+1,N1,2)
 13 NC=(N1-NR1)/2
 14 IND=INTEGERS(1,NC)
 15 EIG1C=EIG1(IC); EIG2C=EIG2(IC)
 16 EIG1CC=EIG1C; EIG2CC=EIG2C
 17 MELD (EIG1CC,EIG2CC)
 18 TERMINAL OFF
 19 COMP=ABS(EIG1CC-EIG2CC)
 20 TERMINAL ON
 21 COMP=RANKED(COMP:LOC)
 22 LOC=LOC(IND)
 23 $ ********  GET EIG1CC INTO ORIGINAL ORDER ********
 24 LOC=RANKED(LOC)
 25 EIG1CC=EIG1CC(LOC)
 26 TERMINAL OFF
 27 DEV=MAX(ABS(EIG1CC-EIG1C))
 28 TERMINAL ON
 29 IF (DEV.GT.1E-12) PRINT "PROBLEM MATCHING COMPLEX EIGS"
 30 $ ********* FIND REORDERING OF EIG2CC  **************
 31 LOC=1+MOD(LOC-1,NC)
 32 EIG2C=EIG2C(LOC)
 33 UU2C=UU2(I,IC)
 34 UU2C=UU2C(I,LOC)
 35 EIG2(IC)=EIG2C
 36 EIG2(1+IC)=CONJ(EIG2C)
 37 UU2(I,IC)=UU2C
*38 UU2(I,1+IC)=CONJ(UU2C)

EDITING SENSITIV
  1 PROGRAM
  2 $ COMPUTES SENSITIVITY MATRIX (SENMAT)
  3 $ INPUTS: EIGENVEC MATRICES U/V FOR A/TRANSPOSE A
  4 $         NREAL; NOTE U/V ORDERED BY REAL/COMPLEX EIGVALS
  5 N = NOROWS(U)
  6 SENMAT=MATRIX(N,N**2:)
  7 $ **** COMPUTE SENS MATRIX FOR EIGENVALUE J ****************
  8 FOR J=1,N
```

```
  9 IF (J.GT.NREAL.AND.MOD(J-NREAL,2).EQ.0) GO TO ENDLOOPJ
 10 SENS=CONJ(V(I,J))**CONJ(U(I,J))/INNER(U(I,J),CONJ(V(I,J)))
 11 $ LOAD ROW J(J+1)OF SENMAT WITH ALL ROWS OF SENSREAL(SENSIMAG)
 12 SENSR=REAL(SENS)
 13 SENSI=IMAG(SENS)
 14 FOR K=1,N
 15 SENMAT(J,1+(K-1)*N)=SENSR(K,I)
 16 IF (J.LE.NREAL) GO TO ENDLOOPK
 17 SENMAT(J+1,1+(K-1)*N)=SENSI(K,I)
 18 ENDLOOPK: ENDLOOP K
*19 ENDLOOPJ: ENDLOOP J

EDITING COOP
  1 PROGRAM
  2 $ ** BYPASS GETS, IF DONE IN PREVIOUS ENTRY **
  3 ONERROR CONTINUE
  4 TERMINAL OFF
  5 IF (COOPGOT.EQ.1) GO TO TERMINAL
  6 TERMINAL ON
  7 GET OPERATE
  8 GET PARTS
  9 GET DATAIN
 10 GET NEWSPEC
 11 GET PRIORITY
 12 GET SETSTORE
 13 GET NODESEL
 14 GET NODEARBI
 15 GET FEASLP
 16 GET INTERACT
 17 GET CONVERT
 18 GET ASSEMBLE
 19 COOPGOT =1
 20 TERMINAL: TERMINAL ON
*21 EXECUTE OPERATE

EDITING OPERATE
  1 PROGRAM
  2 $ ********   OPERATING SYSTEM    ********
  3 $ ** QUERY USER FOR PROGRAM ENTRY CONDITIONS  ***
  4 PRINT "SPECIFY PROGRAM-ENTRY CONDITIONS:"
  5 PRINT "0:  INITIAL ENTRY (ALL-NEW DATA/SPECS)"
  6 PRINT "1:  RE-ENTRY (SOME NEW SPECS)"
  7 PRINT "2:  RE-ENTRY (NO NEW SPECS)"
  8 REQUEST ENTRY
  9 $ *** INPUT/STORAGE EXECUTIONS CONDITIONAL ON ENTRY ***
 10 IF (ENTRY.EQ.2) GO TO NODESEL
 11 IF (ENTRY.EQ.0) EXECUTE DATAIN
 12 IF (ENTRY.EQ.0) EXECUTE PRIORITY
 13 IF (ENTRY.EQ.1) EXECUTE NEWSPEC
 14 $ ***********  TREE SEARCH     ***************
 15 NODESEL:
 16 EXECUTE NODESEL
 17 NEWSOLVE=VECTOR(2:)
 18 FOR BIT=0,1
```

```
 19 EXECUTE NODEARBI
 20 ENDLOOP BIT
 21 $ ***** UPDATE DEPTH/MAXSIZE  *****
 22 DEPTH=LOCS (BINSIZE(NOSOL).GT.0:LAST)
 23 MAXSIZE= MAX(BINSIZE)
 24 LIMIT = ARBS.GT.ARBLIM.OR.MAXSIZE.GT.BINLIM
 25 QUITT=0
 26 IF (LIMIT.OR.SUM(NEWSOLVE)) EXECUTE INTERACT
 27 IF (QUITT.EQ.1) GO TO EMPTY
 28 GO TO NODESEL
 29 EMPTY:
 30 PRINT "SOLUTION BIN CONTAINS" BINSIZE((N**2)*SSS) "SOLUTIONS"
 31 PRINT " NUMBER OF SOLUTIONS TO BE LABELLED/OUTPUT?"
 32 REQUEST NUMBER
 33 IF (NUMBER.EQ.0) GO TO FINISH
 34 FOR PICK=1,NUMBER
 35 EXECUTE CONVERT
 36 ENDLOOP PICK
*37 FINISH:

EDITING DATAIN
  1 PROGRAM
  2 $ **********   INPUTS   ***********
  3 PRINT "THE FOLLOWING DATA INPUTS ARE REQUIRED:"
  4 PRINT "NAME SYSTEM MATRIX (INF. PREC. CENTER POINT)"
  5 REQUEST S
  6 PRINT "SPECIFY SENSITIVITY MATRIX"
  7 REQUEST SEN
  8 PRINT "NAME SYSTEM MATRIX (FIN. PREC. STARTING POINT)"
  9 REQUEST START
 10 PRINT "MATRIX OF LSB ADDRESSES FOR START"
 11 REQUEST ADDR
 12 PRINT "VECTOR OF MAG. BOUNDS ON EIGENVALUE VARIATION?"
 13 REQUEST BOUND
 14 PRINT "FOLLOWING SPECIFICATIONS ARE REQUIRED FOR SEARCH:"
 15 PRINT "NODE SELECTION CRITERION?"
 16 PRINT "  0=SAME BRANCH/BEST PROJ.  1=GLOBAL BEST PROJ."
 17 REQUEST CRITER
 18 PRINT "BIT SPAN FOR COEFFICIENT VARIATIONS?"
 19 REQUEST SSS
 20 PRINT "NO. UNDERFLOW BITS FOR COEFFICIENT VARIATIONS?"
 21 REQUEST UUU
 22 PRINT "LIMIT ON NUMBER OF NODE ARBITRATIONS?"
 23 REQUEST ARBLIM
 24 PRINT "LIMIT ON BIN SIZE?"
 25 REQUEST BINLIM
 26 L=0; SS=S; EXECUTE PARTS
 27 L=1; SS=START; EXECUTE PARTS
*28 EXECUTE SETSTORE

EDITING NEWSPEC
  1 PROGRAM
  2 $ ********  CHANGE SEARCH SPECIFICATINS   *********
  3 $ ** OTHER DATA (INPUT) CANNOT BE CHANGED
```

```
  4 $ ** WITHOUT CHANGING BASIC PROBLEM
  5 PRINT "PRESENT SEARCH LIMIT SPECS. ARE:"
  6 PRINT ARBLIM  BINLIM
  7 PRINT "ENTER NEW SPECS.:"
  8 REQUEST ARBLIM
  9 REQUEST BINLIM
 10 PRINT "CHANGE BIT VARIATION SPECS? (0=NO,1=YES)"
 11 REQUEST CHANGE
 12 IF (CHANGE.EQ.0) GO TO FINISH
 13 PRINT "PRESENT BIT SPAN" SSS
 14 REQUEST SSS
 15 PRINT "PRESENT NO. UNDERFLOW BITS" UUU
 16 REQUEST UUU
 17 $ FUTURE: CHANGE NODE SELECTION CRITERION(CRITER)
 18 IF (CHANGE.EQ.1) EXECUTE SETSTORE
*19 FINISH:

EDITING PRIORITY
  1 PROGRAM
*2 $ *** FIND ARBITRATION PRIORITY: REORDER SEN/LPVECTOR

EDITING SETSTORE
  1 PROGRAM
  2 $ ******** DEFINE BINS: NODE STORAGE MATRICES   *******
  3 $ BIN NO. T  STORES ALL NODES WITH T FIXED (BINARY) VARIABLES.
  4 $ ONE ROW ALLOCATED TO EACH NODE: NO. COLS.=T ; NO. ROWS=NO. NODES
  5 $ CONSIDER INTERVALS OF N**2 COLUMNS IN A ROW.
  6 $ INTERVAL NO. K CONTAINS KTH MSB OF EACH LP VECTOR COMPONENT.
  7 $ ORDERING WITHIN INTERVAL IS PRESET ARBITRATION PRIORITY.
  8 $ ********* DEFINE BINRECS: BIN RECORD MATRICES    ********
  9 $ BINREC NO. T CONTAINS INDIVIDUAL NODE DATA FOR BIN NO. T
 10 $ ONE-TO-ONE CORRESPONDENCE EXISTS BETWEEN ROWS OF :
 11 $ BINREC NO. T AND BIN NO. T.  NODE DATA: COL. 1- OBJ.+COST;
 12 $ COL. 2- COST P (FROM LP); COL. 3- OBJ. (NO. OF ONES)
 13 FOR J=1,(N**2)*SSS
 14 OBJECT ("BIN",J)=MATRIX(1,J:)
 15 OBJECT ("BINREC",J)=MATRIX(1,3:)
 16 ENDLOOP J
 17 $ ****** DEFINE BINSIZE: (VECTOR) LIST OF NO. NODES PER BIN
 18 $ ****** ELEMENT NO. T CORRESPONDS TO BIN NO. T
 19 BINSIZE=VECTOR((N**2)*SSS:)
 20 NOSOL=INTEGERS(1,(N**2)*SSS-1)
 21 $ ** DEFINE BIN (BINSOLVA) TO STORE ANY COMPLETE SOLUTIONS
 22 BINSOLVA=MATRIX(1,N**2:)
 23 $ MAXSIZE: SIZE OF LARGEST BIN
 24 MAXSIZE=0
 25 $ ARBS: NO.  NODES ARBITRATED
 26 ARBS=0
 27 $ DEPTH: MAXIMUM DEPTH IN TREE OF NON-EMPTY BIN
 28 DEPTH=0
 29 $ *******  COMPUTE DATA FOR FEASLP  *********
 30 DELTAA=A1-A0
 31 $ **** CONVERT DELTAA FROM MATRIX TO COLUMN VECTOR
 32 DELAVEC=VECTOR(N**2:)
```

```
 33 ALLCOLS=INTEGERS(1,N)
 34 FOR K=1,N
 35 DELAVEC(1+(K-1)*N)=DELTAA(K,ALLCOLS)
 36 ENDLOOP K
 37 $ **** INITIAL BOUND    ***
 38 BOUND1POS=BOUND-SEN*DELAVEC
 39 BOUND1NEG=BOUND+SEN*DELAVEC
 40 $ **** BIT ADDRESS CONVERSION ***
 41 ADDRLSB=ADDR-UUU
 42 ADDRMSB=ADDRLSB+SSS-1
 43 L=8; SS=ADDRMSB;EXECUTE PARTS
 44 HENCEFORTH AADRMSB IS A8
 45 COLAADRMSB=VECTOR(N**2:)
 46 FOR K=1,N
 47 COLAADRMSB(1+(K-1)*N)=AADRMSB(K,ALLCOLS)
 48 ENDLOOP K
 49 $ **** COMPUTE FIRST WEIGHTING MATRIX (W1)
 50 $ **** AND INTERMEDIATE SENSITIVITY MATRIX (MINTP)
 51 W1=2**COLAADRMSB
 52 W1=DIAGMAT(N**2:W1)
 53 MINT=SEN*W1
 54 $ ** COMPUTE POLARITY (SIGN) OF VARIATIONS  ***
 55 TERMINAL OFF
 56 POL=(-1)*SIGN(DELAVEC)
 57 TERMINAL ON
 58 WHERE (POL.EQ.0) POL=-1
 59 POL=DIAGMAT(N**2:POL)
*60 MINTP=MINT*POL

EDITING NODESEL
  1 PROGRAM
  2 $ *********   NODE SELECTION   **********
  3 $ SELECTION CRITERION(CRITER):0=SAME BRANCH;1=BEST PROJ(GLOB)
  4 IF (CRITER.EQ.0) GO TO SAMEBRAN
  5 $ FUTURE:IF(CRITER.EQ.1) GO TO BESTPROJ
  6 SAMEBRAN: $ CHOOSE NON-EMPTY BIN WITH MAX TREE DEPTH.
  7 DEEP=DEPTH
  8 $ ** INITIALIZE ONLY FOR DEEP=0
  9 IF (DEEP.EQ.0) NODE = VECTOR(1:)
 10 IF (DEEP.EQ.0) GO TO FINISH
 11 HENCEFORTH  BIN IS OBJECT("BIN",DEEP)
 12 $ * IN EACH BIN, NODES (ROWS) ARE STORED IN ORDER OF DECREAS-
 13 $ * ING VALUE FOR SELECTION CRITERION. COST P (FROM LP) IS
 14 $ * PRESENT CRITERION. SELECT NODE IN LAST ROW:
 15 ROW=BINSIZE(DEEP)
 16 ALLCOL=INTEGERS(1,DEEP)
 17 $ *** PLACE SELECTED NODE IN TEMP STORAGE:NODE
 18 NODE=BIN(ROW,ALLCOL)
 19 $ *** ELIMINATE NODE FROM BIN  *****
 20 IF (ROW.NE.1) BIN = ELIMROWS(BIN,ROW)
 21 IF (ROW.EQ.1) BIN(ROW,ALLCOL)=VECTOR(DEEP:)
 22 $ (FUTURE) BPGLOBAL: CHOOSE NODE WITH BEST CRITER/ALL BINS
 23 $ *****  UPDATE BINREC
 24 HENCEFORTH BINREC IS OBJECT ("BINREC",DEEP)
```

```
 25 COLS=INTEGERS(1,3)
 26 IF (ROW.NE.1)BINREC=ELIMROWS(BINREC,ROW)
 27 IF (ROW.EQ.1)BINREC(ROW,COLS)=VECTOR(3:)
 28 BINSIZE(DEEP)=ROW-1
*29 FINISH:

EDITING NODEARBI
  1 PROGRAM
  2 $ *********   NODE ARBITRATION    ***************
  3 $ ** ADJOIN BIT (PASSED FROM OPERATE) TO SELECTED NODE
  4 $ ** (PASSED FROM NODESEL) IN THE LSB. A NEW NODE (NODEARB)
  5 $ ** RESULTS. FOR NODEARB: GENERATE CONSTRAINT MATRIX (ON
  6 $ ** FREE VARIABLES); RUN LP TO DETERMINE FEAS./MIN COST P.
  7 NODEARB=NODE
  8 NODEARB(DEEP+1)=BIT
  9 P=10000
 10 EXECUTE FEASLP
 11 $ *** FEASIBILTIY CHECK:
 12 IF (P.EQ.10000) GO TO FINISH
 13 $ ** FEASIBILTY OK (P.NE.10000):
 14 $ ** INSERT NODEARB INTO PROPER ROW OF BIN;
 15 $ ** BIN IS ORDERED BY DECREASING COST P.
 16 HENCEFORTH BIN IS OBJECT("BIN",DEEP+1)
 17 HENCEFORTH BINREC IS OBJECT("BINREC",DEEP+1)
 18 ALLCOL=INTEGERS(1,DEEP+1)
 19 LASTROW=BINSIZE(DEEP+1)
 20 ALLROW=INTEGERS(1,LASTROW+1)
 21 COLS=INTEGERS(1,3)
 22 $ ** ADD EXTRA ROW TO BIN,BINREC
 23 BIN(LASTROW+1,ALLCOL)=VECTOR(DEEP+1:)
 24 BINREC(LASTROW+1,COLS)=VECTOR(3:)
 25 $ ** FIND CORRECT INSERTION ROW FOR NODEARB  ***
 26 ROW=LOCS(BINREC(ALLROW,2).LE.P:FIRST)
 27 $ ** MOVE DOWN ROWS FROM INSERTION ROW; INSERT NODEARB **
 28 PARTROW=INTEGERS(ROW,LASTROW+1)
 29 BIN(PARTROW,ALLCOL)=SHIFTDOWN(BIN(PARTROW,ALLCOL),1)
 30 BIN (ROW,ALLCOL)=NODEARB
 31 $ ** UPDATE BINREC
 32 BINREC(PARTROW,COLS)=SHIFTDOWN(BINREC(PARTROW,COLS),1)
 33 BINREC(ROW,2)=P
 34 BINREC(ROW,3)=SUM(NODEARB)
 35 BINREC(ROW,1)=P+BINREC(ROW,3)
 36 $ ** UPDATE OTHER RECORDS
 38 BINSIZE(DEEP+1)=BINSIZE(DEEP+1)+1
 39 IF ((DEEP+1).NE.(N**2)*SSS) GO TO FINISH
 40 $ ** EVENT RECORD ****
 41 NEWSOLVE(BIT+1)=1
 42 $ *** STORE NEW SOLUTION (FIXED VARIATION) IN REAL NO. FORM:
 43 SOLUTA= W1* POL * BIEXA
 44 BINSOLVA(LASTROW+1,1)=VECTOR(N**2:)
 45 BINSOLVA=SHIFTDOWN(BINSOLVA,1)
 46 BINSOLVA(1,1)=SOLUTA
 47 $ * NOTE: BIT FORMS OF NEW SOLUTIONS ENTER LAST BIN IN ORDER
 48 $ * CORRESPONDING TO REAL NO. FORM (I.E., AT TOP OF PILE)
```

```
*49 FINISH: ARBS=ARBS+1

EDITING FEASLP
  1 PROGRAM
  2 $ ** FOR PASSED NODEARB/DEEP+1: GENERATE CONSTRAINT MATRIX
  3 $ ** (CONALL) FROM SENSITIVITY MATRIX (SEN). RUN LP.
  4 $ ********** COMPUTE FINAL WEIGHTING MATRIX (W2)  *****
  5 FIXVAR=VECTOR(N**2:)
  6 ALLFIX=INTPART((DEEP+1)/N**2)
  7 FIXVAR(1)=ALLFIX
  8 FIXVAR=CUMSUM(FIXVAR)
  9 INCROW=MODULO(DEEP+1,N**2)
 10 IF (INCROW.EQ.0) GO TO W2
 11 INCROWS=INTEGERS(1,INCROW)
 12 FIXVAR(INCROWS)=FIXVAR(INCROWS)+1
 13 W2: W2=2**(-1*FIXVAR)
 14 W2=DIAGMAT(N**2:W2)
 15 $ ***WEIGHTED SENSITIVITY MATRIX FOR LP ***
 16 MFIN=MINTP*W2
 17 $ ** FOR COLUMNS OF MFIN CORRESPONDING TO COMPLETELY
 18 $ ** FIXED VARIABLE, MASK TO ZERO
 19 MASK=VECTOR(N**2:)
 20 WHERE (FIXVAR.NE.SSS) MASK=1
 21 MASK=DIAGMAT(N**2:MASK)
 22 CON=MFIN*MASK
 23 $ ** RANGE CONSTRAINT MATRIX (RANGECON) ***
 24 RANGECON=VECTOR(N**2:1)
 25 RANGECON=CUMSUM(RANGECON)
 26 RANGECON=DIAGMAT(N**2:RANGECON)
 27 $ **  TOTAL CONSTRAINT MATRIX (CONALL) FOR LP  ***
 28 CONALL=MATRIX(2*N+N**2,N**2:)
 29 CONALL(1,1)=CON
 30 CONALL(N+1,1)=(-1)*CON
 31 CONALL(2*N+1,1)=RANGECON
 32 $ ** BINARY EXPANSION (BIEXA) FOR FIXED PART OF VEC A *
 33 POWERS=(-1)*INTEGERS(0,FIXVAR(1)-1)
 34 BINAR=VECTOR(:2**POWERS)
 35 AVECS=MATRIX(FIXVAR(1),N**2:)
 36 ALLELS=INTEGERS(1,N**2)
 37 IF (ALLFIX.EQ.0) GO TO INC
 38 FOR K=1,ALLFIX
 39 AVECS(K,1)=NODEARB(ALLELS+(K-1)*N**2)
 40 ENDLOOP K
 41 INC: IF (INCROW.EQ.0) GO TO AVECS
 42 AVECS(FIXVAR(1),1)=NODEARB((FIXVAR(1)-1)*N**2+INCROWS)
 43 AVECS: AVECS=TRANSPOSE(AVECS)
 44 BIEXA=AVECS*BINAR
 45 $ *** BOUND FOR LP FROM BIEXA  ****
 46 BOUND2POS=BOUND1POS-MINTP*BIEXA
 47 BOUND2NEG=BOUND1NEG+MINTP*BIEXA
 48 BOUNDMAG=VECTOR(N**2:2)
 49 BOUNDMAG=CUMSUM(BOUNDMAG)
 50 BOUNDALL=VECTOR(2*N+N**2:)
 51 BOUNDALL(1)=BOUND2POS
```

```
 52 BOUNDALL(N+1)=BOUND2NEG
 53 BOUNDALL(2*N+1)=BOUNDMAG
 54 $ ******  RUN LP    **********
 55 COSTVEC=VECTOR(N**2:1)
 56 COSTVEC=CUMSUM(COSTVEC)
 57 ONERROR CONTINUE, NOMESSAGE
 58 TERMINAL OFF
 59 P = LPMIN (COSTVEC, CONALL, BOUNDALL, -1*(2*N+N**2))
*60 TERMINAL ON

EDITING INTERACT
  1 PROGRAM
  2 $ ** INTERACTION WITH USER REGARDING SEARCH EVENTS ***
  3 IF (LIMIT.EQ.0) GO TO NEWSOLVE
  4 PRINT "SEARCH LIMITS EXCEEDED!!"
  5 PRINT BINSIZE, ARBS
  6 PRINT BINLIM, ARBLIM
  7 PRINT "OPTIONS: 0=INCREASE LIMITS/CHANGE SPECS."
  8 PRINT "          1= QUIT (BUT OUTPUT SOLUTION BIN)"
  9 REQUEST OPTION
 10 QUITT=OPTION
 11 IF (QUITT.EQ.1) GO TO FINISH
 12 EXECUTE NEWSPEC
 13 NEWSOLVE:
 14 IF (PROD(NEWSOLVE).EQ.0) GO TO FINISH
 15 PRINT "NEW SOLUTION(S) FOUND!!"
 16 PRINT "OPTIONS: 0=CONTINUE SEARCH"
 17 PRINT "         1=OUTPUT SOLUTION BIN/QUIT"
 18 PRINT "         2=OUTPUT NEW SOLUTION/CONTINUE"
 19 REQUEST OPTION
 20 QUITT=OPTION
 21 IF (OPTION.EQ.0.OR.OPTION.EQ.1) GO TO FINISH
 22 FOR K=1,2
 23 IF (NEWSOLVE(K).EQ.0) GO TO ENDLOOPK
 24 PICK=K
 25 IF (K.EQ.2) PICK=SUM(NEWSOLVE)
 26 EXECUTE CONVERT
 27 ENDLOOPK: ENDLOOP K
*28 FINISH:

EDITING CONVERT
  1 PROGRAM
  2 $ ** GET REAL NO. SOLUTION FROM BINSOLVA; ROW
  3 $ ** SPECIFIED  BY PICK. ADD TO A0, REASSEMBLE
  4 $ ** S MATRIX, LABEL, AND OUTPUT.
  5 ALLCOLS=INTEGERS(1,N**2)
  6 SOLUTA=BINSOLVA(PICK,ALLCOLS)
  7 $ EXECUTE REORDER IF PRIORITY IS EFFECTIVE
  8 $ *** CONVERT TO MATRIX  ***
  9 ALLCOLS=INTEGERS(1,N)
 10 ASOL=MATRIX(N,N:)
 11 FOR J=1,N
 12 ASOL(J,1)=SOLUTA(ALLCOLS+(J-1)*N)
 13 ENDLOOP J
```

```
 14 AFIN=A1+ASOL
 15 A=AFIN; B=B1; C=C1; D=D1
 16 EXECUTE ASSEMBLE
 17 OBJECT("SCOOP",PICK)=SS
*18 PRINT OBJECT("SCOOP",PICK)

EDITING MODEOP
  1 PROGRAM
  2 $ ** ADJUST MATRICES B,C,D TO COMPENSATE FOR MODE SHIFT
  3 $ ** (I.E., EIGENVECTOR SHIFT) DURING QUANTIZATION OF A.
  4 $ ** INPUT: SYSTEM MATRIX S; SPECS FOR ZEROES OF H(Z).
  5 $ ** OUTPUT: SYSTEM MATRIX : D CHANGED TO GIVE UNITY
  6 $ ** ZERO-FREQ. GAIN; B,C CHANGED TO GIVE SPEC'D ZEROES
  7 $ ** BYPASS GETS, IF DONE IN PREVIOUS ENTRY **
  8 ONERROR CONTINUE
  9 TERMINAL OFF
 10 IF (MODEGOT.EQ.1) GO TO TERMINAL
 11 TERMINAL ON
 12 GET PARTS
 13 GET ORDEREIG
 14 GET ASSEMBLE
 15 MODEGOT=1
 16 TERMINAL: TERMINAL ON
 17 PRINT "INPUT SYSTEM MATRIX"
 18 REQUEST S
 19 L=3; SS=S; EXECUTE PARTS
 20 PRINT "NUMBER OF ZEROES TO BE -1 ??"
 21 REQUEST NUMBER
 22 IF ((N-NUMBER).EQ.0) GO TO DOMAIN
 23 PRINT "SPECIFY OTHER ZEROES (IN VECTOR)"
 24 REQUEST ZZEROES
 25 IF (NOELS(ZZEROES).NE.(N-NUMBER)) PRINT "ERROR:NO. ZEROES"
 26 DOMAIN: DOMAIN COMPLEX
 27 EIGA3=EIGENVALS(A3:U3)
 28 EIG=EIGA3; UU=U3; EXECUTE ORDEREIG
 29 EIGA3=EIG; U3=UU
 30 K=VECTOR (N:)
 31 ZEROES=VECTOR(N:)
 32 ZNEGONE=VECTOR(NUMBER:-1)
 33 ZNEGONE=CUMSUM(ZNEGONE)
 34 ZEROES(1)=ZNEGONE
 35 IF (NUMBER.EQ.N) GO TO ONES
 36 ZEROES(NUMBER+1)=ZZEROES
 37 ONES: ONES=VECTOR(N:1)
 38 ONES=CUMSUM(ONES)
 39 DNEW= PROD(ONES-EIGA3)/PROD(ONES-ZEROES)
 40 ALLROWS=INTEGERS(1,N)
 41 FOR J=1,N
 42 Z=VECTOR(N:)
 43 Z(1)=EIGA3(J)
 44 Z=CUMSUM(Z)
 45 NUM=Z-ZEROES
 46 DENOM=Z-EIGA3
 47 CONSTRAIN (DENOM:ALLROWS.NE.J)
```

```
 48 K(J)=DNEW*PROD(NUM)/PROD(DENOM)
 49 ENDLOOP J
 50 CC=C3*U3
 51 BB=INVERSE(U3)*B3
 52 RR=VECTOR(N:)
 53 FOR J=1,N
 54 RR(J)=K(J)/(BB(J)*CC(J))
 55 RR(J)=SQRT(RR(J))
 56 ENDLOOP J
 57 RRR=DIAGMAT(N:RR)
 58 M=U3*RRR*INVERSE(U3)
 59 CNEW=C3*M
 60 BNEW=M*B3
 61 A=A3; B=BNEW; C=CNEW; D=DNEW
 62 EXECUTE ASSEMBLE
 63 STEST=SS
*64 STEST=REAL(STEST)

EDITING PARTS
  1 PROGRAM
  2 $ PARTITIONS SYSTEM MATRIX SS INTO A,B,C,D WITH INDEX L
  3 $ ** IF L=100, INDEX IS DELETED
  4 N=NOROWS(SS)-1
  5 I=INTEGERS(1,N)
  6 IF (L.EQ.100) GO TO NOINDEX
  7 OBJECT("A",L)=SS(I,I)
  8 OBJECT("B",L)=SS(I,N+1)
  9 OBJECT("C",L)=SS(N+1,I)
 10 OBJECT("D",L)=SS(N+1,N+1)
 11 GO TO FINISH
 12 NOINDEX:
 13 A=SS(I,I)
 14 B=SS(I,N+1)
 15 C=SS(N+1,I)
 16 D=SS(N+1,N+1)
*17 FINISH:

EDITING ASSEMBLE
  1 PROGRAM
  2 $ ** CREATE SYSTEM MATRIX SS FROM MATRICES
  3 $ ** A,B,C,D SUPPLIED. A IS N X N MATRIX.
  4 N=NOROWS(A)
  5 SS=MATRIX(N+1,N+1:)
  6 SS(1,1)=A
  7 SS(N+1,1)=C
  8 TEMPSS=TRANSPOSE(SS)
  9 TEMPSS(N+1,1)=B
 10 SS=TRANSPOSE(TEMPSS)
*11 SS(N+1,N+1)=D
```

APPENDIX B

PHASE RESPONSES OF COMPLETED DESIGNS

Angle in Degrees

Normalized Freq.	2^{nd}-Order Example	3^{rd}-Order Example	4^{th}-Order Example
0	0	0	0
.025	-6.39	-9.0	-11.8
.05	-12.9	-18.3	-23.8
.075	-19.8	-27.8	-36.2
.1	-27.2	-38.0	-49.4
.125	-35.3	-49.2	-63.6
.15	-44.3	-61.6	-79.5
.175	-54.3	-76.1	-97.9
.2	-65.5	-93.1	-120.3
.225	-77.6	-113.1	-148.0
.25	-90.2	-135.2	-181.1
.275	-102.9	-157.3	-213.9
.3	-114.9	-177.3	-241.7
.325	-126.0	-194.5	-264.1
.35	-136.0	-209.2	-282.6
.375	-144.8	-222.3	-299.0
.4	-152.8	-234.8	-314.5
.425	-160.1	-249.0	-331.0
.45	-166.8	-274.4	-349.4
.475	-172.9	-337.1	-359.1
.5	-360	-360	-360.0

APPENDIX C

UNIT-SAMPLE RESPONSES OF COMPLETED DESIGNS

Sample Index	2^{nd}-Order Example	3^{rd}-Order Example	4^{th}-Order Example
0	.29297	.16602	.09375
1	.58527	.49604	.37379
2	.24413	.44278	.51827
3	-.10021	9.9206E-4	.19474
4	-.04289	-.14716	-.16189
5	.017155	-2.7159E-4	-.1034
6	.0074287	.048932	.070763
7	-.0029363	7.0937E-5	.048002
8	-.0012955	-.01627	-.032228
9	5.0249E-4	-1.7147E-5	-.022059
10	2.259E-4	.0054099	.014736
11	-8.5973E-5	3.5602E-6	.010128
12	-3.9382E-5	-.0017988	-.0067409
13	1.4707E-5	-4.7181E-7	-.0046502
14	6.8648E-6	5.9811E-4	.0030836
15	-2.5153E-6	-7.986E-8	.002135
16	-1.1964E-6	-1.9888E-4	-.0014105
17	4.301E-7	1.0527E-7	9.8021E-4
18	2.0848E-7	6.6127E-5	6.4524E-4
19	-7.3528E-8	-6.1176E-8	4.5003E-4
20	-3.6324E-8	-2.1988E-5	-2.9515E-4

REFERENCES

A. BACKGROUND

[1] N. Weiner, Extrapolation, Interpolation, and Smoothing of Stationary Time Series, Cambridge, Mass.: Technology Press (M.I.T.), 1964.

[2] W. R. Bennett, "Spectra of Quantized Signals," Bell System Tech. Journ., vol. 27, pp. 446-472, July 1948.

[3] W. K. Linvill, "Sampled Data Control Systems Studied Through Comparison of Sampling with Amplitude Modulation," Trans. AIEE, vol. 70, part II, pp. 1779-1788, 1951.

[4] B. Widrow, "A Study of Rough Amplitude Quantization by Means of Nyquist Sampling Theory," IRE Trans. Circuit Theory, vol. CT-3, pp. 266-276, Dec. 1956.

[5] J. R. Ragazinni and G. F. Franklin, Sampled Data Control Systems, New York: McGraw-Hill, 1958.

[6] A. V. Oppenheim and R. W. Schafer, Digital Signal Processing, Englewood Cliffs, N.J.: Prentice-Hall, 1975.

[7] B. Gold and C. M. Rader, Digital Processing of Signals, New York: McGraw-Hill, 1969.

B. APPROXIMATION PROBLEM

[1] J. F. Ormsby, "Design of Numerical Filters with Application to Missile Data Processing," J. Assoc. Computing Machinery (ACM), vol. 8, pp. 440-466, July 1961.

[2] J. F. Kaiser, "Digital Filters," System Analysis by Digital Computer, F. F. Kuo and J. F. Kaiser eds., New York: Wiley, pp. 218-285, 1966.

[3] K. Steiglitz, "The Equivalence of Digital and Analog Signal Processing," Information and Control, vol. 8, pp. 455-467, Oct. 1965.

[4] A. J. Gibbs, "On the Frequency-Domain Response of Causal Digital Filters," Ph.D. Thesis, Univ. of Wisconsin, 1969.

[5] M. Sablatash, "Approximation Theory for Digital Filters," IEEE Trans. Circuit Theory, vol. CT-18, pp. 741-743, Nov. 1971.

C. ROUNDOFF NOISE

[1] J. E. Bertram, "Effect of Quantization in Sampled Data Feedback Systems," Trans. AIEE, vol. 77, pp. 177-182, Sept. 1958.

[2] L. B. Jackson, "On the Interaction of Roundoff Noise and Dynamic Range in Digital Filters," Bell System Tech. Jour., vol. 49, pp. 159-184, Feb. 1970.

[3] A. Fettweis, "On the Connection Between Multiplier Word Length and Roundoff Noise in Digital Filters," IEEE Trans. Circuit Theory, vol. CT-19, pp. 486-491, Sept. 1972.

[4] A. Fettweis, "Roundoff Noise and Attenuation Sensitivity in Digital Filters with Fixed-Point Arithmetic," IEEE Trans. Circuit Theory, vol. CT-20, pp. 174-175, March 1973.

[5] A. Fettweis, "On Sensitivity and Roundoff Noise in Wave Digital Filters," IEEE Trans. Accoust., Speech, Signal Processing, vol. ASSP-22, pp. 383-384, Oct. 1974.

[6] A. Fettweis, "On the Evaluation of Roundoff Noise in Digital Filters," IEEE Trans. Circuits Syst., vol. CAS-22, p. 896, Nov. 1975.

[7] S. R. Parker and P. E. Girard, "Correlated Noise Due to Roundoff in Fixed Point Digital Filters," IEEE Trans. Circuits Syst., vol. CAS-23, pp. 204-211, Apr. 1976.

[8] L. B. Jackson, "Roundoff Noise Bounds Derived from Coefficient Sensitivities for Digital Filters," IEEE Trans. Circuits Syst., vol. CAS-23, pp. 481-485, Aug. 1976.

[9] A. Anuff and C. Y. Kao, "Comments on 'On the Evaluation of Roundoff Noise in Digital Filters'", IEEE Trans. Circuits Syst., vol. CAS-23, p. 573, Sept. 1976.

[10] C. T. Mullis and R. A. Roberts, "Synthesis of Minimum Roundoff Noise Fixed Point Digital Filters," IEEE Trans. Circuits Syst., pp. 551-562, Sept. 1976.

[11] L. B. Jackson, A. G. Lindgren and Y. Kim, "Synthesis of State-Space Digital Filters with Low Roundoff Noise and Coefficient Sensitivity," Proc. IEEE Int. Symp. Circuits and Systems, pp. 41-44, Phoenix, Ariz., 1977.

D. LIMIT CYCLES

[1] I. W. Sandberg, "A Theorem Concerning Limit Cycles in Digital Filters," Proc. of the 7th. Annual Allerton Conf. on Circuit and System Theory, pp. 63-68, Oct. 1969.

[2] P. M. Ebert, J. E. Mazo and M. G. Taylor, "Overflow Oscillations in Digital Filters," Bell System Tech. Jour., vol. 48, pp. 2999-3020, Nov. 1969.

[3] S. R. Parker and S. F. Hess, "Limit Cycle Oscillations in Digital Filters," IEEE Trans. Circuit Theory, vol. CT-18, pp. 687-697, Nov. 1971.

[4] A. N. Willson, Jr., "Some Effects of Quantization and Adder Overflow on the Forced Response of Digital Filters," Bell System Tech. Jour., vol. 51, pp. 863-887, April 1972.

[5] I. W. Sandberg and J. F. Kaiser, "A Bound on Limit Cycles in Fixed-Point Implementation of Digital Filters," IEEE Trans. Audio Electroacoust., vol. AU-20, pp. 110-112, June 1972.

[6] A. N. Wilson, Jr., "Limit Cycles Due to Adder Overflow in Digital Filters," IEEE Trans. Circuit Theory, vol. CT-19, pp. 342-354, July 1972.

[7] J. L. Long and T. N. Trick, "An Absolute Bound on Limit Cycles Due to Roundoff Errors in Digital Filters," IEEE Trans. Audio Electroacoust., vol. AU-21, pp. 27-30, Feb. 1973.

[8] J. L. Long and T. N. Trick, "A Note on Absolute Bounds on Quantization Errors in Fixed Point Implementations of Digital Filters," IEEE Trans. Circuit Syst., vol. CAS-22, pp. 567-570, June 1975.

[9] S. R. Parker and S. Yakowitz, "A General Method for Calculating Quantization Error Bounds Due to Roundoff in Multivariable Digital Filters," IEEE Trans. Circuits Syst., vol. CAS-22, pp. 570-572, June 1975.

[10] T. Claasen, W. Mecklenbräuker, and J. Peek, "On the Stability of the Forced Response of Digital Filters with Overflow Non-Linearities," IEEE Trans. Circuits Syst., vol. CAS-22, pp. 692-696, Aug. 1975.

[11] K. Meerkötter, "Realization of Limit-Cycle-Free Second-Order Digital Filters," Proc. IEEE Int. Symp. Circuits and Systems, pp. 295-298, Munich, 1976.

[12] V. B. Lawrence and K. V. Mina, "A New and Interesting Class of Limit Cycles in Recursive Digital Filters," Proc. IEEE Int. Symp. Circuits and Systems, pp. 191-194, Phoenix, Ariz., 1977.

[13] D. Mitra, "Summary of Some Results on Large Amplitude, Self-Sustained Oscillations in High Order Digital Filter Sections Using Saturation Arithmetic," Proc. IEEE Int. Symp. Circuits and Systems, pp. 195-198, Phoenix, Ariz., 1977.

[14] C. W. Barnes and A. T. Fam, "Minimum Norm Recursive Digital Filters that are Free of Overflow Limit Cycles," IEEE Trans. Circuits Syst., vol. CAS-24, pp. 569-574, Oct. 1977.

[15] W. L. Mills, C. T. Mullis and R. A. Roberts, "Digital Filter Realizations Without Overflow Oscillations," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-26, pp. 334-338, Aug. 1978.

[16] L. B. Jackson, "Limit Cycles in State-Space Structures for Digital Filters," IEEE Trans. Circuit Syst., vol. CAS-26, pp. 67-68, Jan. 1979.

E. COEFFICIENT SENSITIVITY

[1] J. B. Knowles and E. M. Olcayto, "Coefficient Accuracy and Digital Filter Response," IEEE Trans. Circuit Theory, vol. CT-15, pp. 31-41, March 1968.

[2] R. Tomovic and M. Vukobratovic, General Sensitivity Theory, New York: Elsevier, 1972.

[3] J. B. Cruz, System Sensitivity Analysis, Stroudsberg, Pa.: Dowden, Hutchinson, and Ross, Inc., 1973.

[4] P. M. Frank, Introduction to System Sensitivity Theory, New York: Academic Press, 1978.

F. STATE-SPACE APPROACH

[1] R. E. Kalman and J. E. Bertram, "Control System Analysis and Design Via the 'Second Method' of Lyapunov," Journal Basic Engineering, Trans. ASME, pp. 371-400, June 1960.

[2] R. E. Kalman, "Mathematical Description of Linear Dynamical Systems," SIAM Jr. Control, vol. 1, pp. 152-192, 1963.

[3] R. E. Kalman, "On a New Characterization of Linear Passive Systems," Proc. of the 1st. Annual Allerton Conf. on Circuits and System Theory, pp. 456-470, 1963.

[4] R. J. Duffin and D. Hazony, "The Degree of a Rational Matrix Function," J. SIAM, Appl. Math., vol. 11, pp. 645-658, Sept. 1963.

[5] E. G. Gilbert, "Controllability and Observability in Multivariable Systems," SIAM J. Control, vol. 1, pp. 128-151, 1963.

[6] L. Zadeh and C. A. Desoer, Linear System Theory: The State Space Approach. New York: McGraw-Hill, 1963.

[7] R. E. Kalman, "Irreducible Realizations and the Degree of a Rational Matrix," SIAM J. Control, vol. 13, pp. 520-544, 1965.

[8] B. L. Ho and R. E. Kalman, "Effective Construction of Linear State Variable Models from Input-Output Data," Proc. of the 3rd. Annual Allerton Conf. on Circuit and System Theory, pp. 449-459, 1965.

[9] R. E. Kalman, "Algebraic Theory of Linear Systems," Proc. of the 3rd. Annual Allerton Conf. on Circuit and System Theory, pp. 563-577, 1965.

[10] K. Ogata, State Space Analysis of Control Systems. Englewood Cliffs, N.J.: Prentice-Hall, 1967.

[11] J. L. Willems, Stability Theory of Dynamical Systems, New York: Wiley, 1970.

[12] S. Barnett, Matrices in Control Theory, New York: Van Nostrand Reinhold Co., 1971.

[13] A. S. Willsky, Digital Signal Processing and Control and Estimation Theory, Cambridge, Mass.: MIT Press, 1979.

G. WAVE DIGITAL FILTERS

[1] A. Fettweis, "Digital Filter Structures Related to Classical Filter Networks," Arch. Elek. Ubertragung., vol. 25, pp. 79-89, Feb. 1971.

[2] A. Fettweis, "Some Principles of Designing Digital Filters Imitating Classical Filter Structures," IEEE Trans. Circuit Theory, vol. CT-18, pp. 314-316, March 1971.

[3] A. Fettweis, "Pseudopassivity, Sensitivity, and Stability of Wave Digital Filters," IEEE Trans. Circuit Theory, vol. CT-19, pp. 668-673, Nov. 1972.

[4] A. Fettweis and A. Sedlmyer, "Digital Filters with True Ladder Configuration," Int. J. Circuit Theory Appl., vol. 1, pp. 1-10, March 1973.

[5] A. Fettweis, "Reciprocity, Inter-Reciprocity, and Transposition in Wave Digital Filters," Int. J. Circuit Theory Appl., vol. 1, pp. 323-337, Dec. 1973.

[6] A. Fettweis, H. Levin and A. Sedlmyer, "Wave Digital Lattice Filters," Int. J. Circuit Theory Appl., vol. 2, pp. 203-211, March 1974.

[7] K. Renner and S. Gupta, "Reduction of Roundoff Noise in Wave Digital Filters," IEEE Trans. Circuits Syst., vol. CAS-21, pp. 305-311, March 1974.

[8] A. Fettweis, "On Sensitivity and Roundoff Noise in Wave Digital Filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-22, pp. 383-384, Oct. 1974.

[9] A. Fettweis, "Wave Digital Filters with Reduced Number of Delays," Int. J. Circuit Theory Appl., vol. 2, pp. 319-330, Dec. 1974.

[10] A. Fettweis, "Canonic Realization of Ladder Wave Digital Filters," Int. J. Circuit Theory Appl., vol. 3, pp. 321-332, March 1975.

[11] A. Fettweis and K. Meerkötter, "Suppression of Parasitic Oscillations in Wave Digital Filters," IEEE Trans. Circuits Syst., vol. CAS-22, pp. 239-246, March 1975.

[12] S. Y. Hwang, 'Comments on "Reduction of Roundoff Noise in Wave Digital Filters"', IEEE Trans. Circuits Syst., vol. CAS-22, p. 764, Sept. 1975.

[13] A. Fettweis and K. Meerkötter, "On Adaptors for Wave Digital Filters," IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-23, pp. 516-525, Dec. 1975.

[14] W. Wegener, "Design of Wave Digital Filters with Very Short Coefficient Word Lengths," Proc. IEEE Int. Symp. Circuits and Systems, Munich, 1976.

[15] G. O. Martens and K. Meerkötter, "On n-Port Adaptors for Wave Digital Filters with Application to Bridged-Tee Filters," Proc. IEEE Int. Symp. Circuits and Systems, pp. 514-517, Munich, 1976.

[16] G. O. Martens and H. H. Le, "Wave Digital Adaptors for Reciprocal Second Order Sections," IEEE Trans. Circuits Syst., vol. CAS-25, pp. 1077-1083, Dec. 1978.

H. DIGITAL IMPLEMENTATION (STRUCTURE)

[1] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-22, pp. 456-462, Dec. 1974.

[2] A. Peled, "On the Hardware Implementation of Digital Signal Processors," IEEE Trans. Acoust., Speech, and Signal Processing, vol. ASSP-24, pp. 76-86, Feb. 1976.

[3] P. R. Moon and G. O. Martens, "A Digital Filter Structure Requiring Only m-Bit Delays, Shifters, Inverters and m-Bit Adders Plus Simple Logic Circuitry," IEEE Trans. Circuits Syst., vol. CAS-27, pp. 901-908, Oct. 1980.

I. MATRIX THEORY

[1] F. R. Gantmacher, The Theory of Matrices, New York: Chelsea, 1964.

[2] J. N. Franklin, Matrix Theory, Englewood Cliffs, N.J.: Prentice-Hall, 1968.

J. OPTIMIZATION AND INTEGER PROGRAMMING

[1] E. Avenhaus, "On the Design of Digital Filters with Coefficients of Limited Word Length," IEEE Trans. Audio Electroacoust., Vol. AU-20, pp. 206-212, Aug. 1972.

[2] M. Suk and S. K. Mitra, "Computer-Aided Design of Digital Filters with Finite Word Lengths," IEEE Trans. Audio Electro-Acoust., Vol. AU-20, pp. 356-363, Dec. 1972.

[3] J. W. Bandler, B. L. Bardakjian, and J. H. K. Chen, "Design of Recursive Digital Filters with Optimum Word Length Coefficients," Proc. of the Eighth Annual Princeton Conference on Information Science and Systems, Dept. of Elec. Engr., Princeton, N.J., 1974.

[4] C. Charalambous and M. J. Best, "Optimization of Recursive Digital Filters with Finite Word Lengths," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-22, pp. 424-431, Dec. 1974.

[5] F. Brglez, "Digital Filter Design with Short Word-Length Coefficients," IEEE Trans. Circuits Syst., Vol. CAS-25, pp. 1044-1050, Dec. 1978.

[6] W. Wegener, "On the Design of Wave Digital Lattice Filters with Short Coefficient Word Lengths and Optimal Dynamic Range, IEEE Trans. Circuits Syst., Vol. CAS-25, pp. 1091-1098, Dec. 1978.

[7] P. R. Adby and M. A. H. Dempster, Introduction to Optimization Methods, London: Chapman and Hall, 1974.

[8] Stanley Zionts, Linear and Integer Programming, Englewood Cliffs, N.J.: Prentice Hall, 1974.

[9] A. H. Land and A. G. Doig, "An Automatic Method for Solving Discrete Programming Problems," Econometrica, Vol. 28, pp. 497-520, 1960.

[10] R. J. Dakin, "A Tree-Search for Mixed Integer Programming Problems," Computer Journal, Vol. 8, pp. 250-255, 1965.

[11] E. L. Lawler and D. E. Wood, "Branch and Bound Methods - A Survey," Operations Research, 14, pp. 699-719, 1966.

[12] B. Roy, R. Benayoun and J. Tergny, "From S.E.P. Procedure to the Mixed OPHELIE Program," J. Abadie, ed., Integer and Non-linear Programming, New York: Amer. Elsevier Publ. Co., 1970.

[13] A. M. Geoffrion and R. E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science, Vol. 18, pp. 465-491, May, 1972.

[14] J. J. H. Forest, J. P. H. Hirst, and J. A. Tomlin, "Practical Solution of Large Mixed Integer Programming Problems with UMPIRE," Management Science, 20, pp. 736-773, 1974.

[15] H. P. Williams, Model Building in Mathematical Programming, New York: John Wiley & Sons, 1978.