

INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.
2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.
3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.
4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.
5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

**University
Microfilms
International**

300 N. Zeeb Road
Ann Arbor, MI 48106

AZIZ, IRFAN

FIXED POINT DIGITAL FILTER SIMULATION.

THE UNIVERSITY OF ARIZONA,

M.A., 1982

University
Microfilms
International

300 N. Zeeb Road, Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark .

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other _____

**University
Microfilms
International**

FIXED POINT DIGITAL FILTER
SIMULATION

by

Irfan Aziz

A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
In the Graduate College
THE UNIVERSITY OF ARIZONA

1 9 8 2

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Stefan Aziz

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

John V. Wait
J. V. WAIT
Professor of Electrical Engineering

May 13, 1982
Date

ACKNOWLEDGMENTS

I wish to acknowledge the help of Dr. John V. Wait, my advisor, whose insight, encouragement, guidance and many valuable comments made this investigation possible. I am also grateful to Dr. L. C. Schooley and Dr. M. K. Sundareshan for their valuable comments and useful suggestions.

Throughout my academic career my parents have been a source of strength and unending support. To them I owe the most.

TABLE OF CONTENTS

	Page
LIST OF ILLUSTRATIONS	vi
LIST OF TABLES	viii
ABSTRACT	ix
CHAPTER	
1. INTRODUCTION	1
2. DETAILS ON STRUCTURES IMPLEMENTED	4
Finite Word-Length Considerations	4
The Digital Filter Transfer Function	6
Implementation Considerations	6
Structures Implemented	7
Cascade Form	10
Parallel Form	12
Normal Form	14
3. SIMULATION SOFTWARE	20
Parameter Evaluation	20
Cascade (Direct-II) Form	21
Parallel Form	22
Normal Form	25
Filter Implementation	26
Function SHADD	27
Function SMULT	27
Implementing the Delay	28
The Register Overflow Problem	32
4. TEST EXAMPLES	38
Summary and Conclusions	88
APPENDIX A: CARD DECK SETUP FOR A TYPICAL RUN ON THE CDC CYBER 175	90
APPENDIX B: SIMULATION SOFTWARE LISTING	95

TABLE OF CONTENTS -- Continued

	Page
REFERENCES	130

LIST OF ILLUSTRATIONS

Figure		Page
2.1	Three structures implemented	9
2.2	Second-order Cascade realization	11
2.3	Second-order Parallel realization	13
2.4	Parallel connection of second-order	13
2.5	Second-order Normal realization	15
3.1	Symbolic representation of an adder and a multiplier	28
3.2	Simulating a tandem connection of two delays	29
3.3	First-order section	30
3.4	Second-order section in Cascade form based on (3.11)	33
3.5	Second-order section in Cascade form with additional adders to prevent overflow	34
4.1	The second-order, 1.0 dB equal-ripple magnitude low-pass digital filter function (4.2) is subjected to a step- input of magnitude 0.5 for a Cascade form	40
4.2	A section of the Cascade form	46
4.3	Response and Error function plots for simulations of the digital filter function (4.2); a Cascade form is used	47
4.4	Response and Error function plots for simulations of the digital filter function (4.2); Parallel form is used	52

LIST OF ILLUSTRATIONS -- Continued

Figure	Page
4.5	Response and Error function plots for simulations of the digital filter function (4.2); a Normal form is used 58
4.6	The sixth-order, low-pass filter function (4.6) is simulated using the Cascade, Parallel, and Normal forms and driven by a step input $u(n) = 0.5$ 67
4.7	The second-order, band-pass digital filter function (4.8) is simulated using the Normal, Parallel, and Cascade forms and driven respectively by a sinusoidal input $u(n) = 0.5 \sin(n)$, a random broad-band input, and a step input, $u(n) = 0.5$ 81

LIST OF TABLES

Table		Page
3.1	Table with maximum allowable values for coefficients in a second-order section for the three structures	36
4.1	A comparison of the three structures based upon the maximum percentage relative error, $ e(n)_{\max} $, for the low-pass filter of 4.2	65
4.2	A comparison of the three structures based upon the maximum percentage relative error, $ e(n)_{\max} $, for the sixth-order low-pass filter function 4.6	79
4.3	A comparison of the three structures based upon the maximum percentage relative error, $ e(n)_{\max} $, for the band-pass filter of 4.5	87

ABSTRACT

Linear digital filtering systems, when implemented in hardware or on a micro-computer, must use a short word-length, typically 8-16 bits. Before implementing the filter, simulation of its behavior on a general purpose computer can provide insight into the effects of finite word-length.

Using FORTRAN-based fixed-point simulation techniques, this thesis explores the effects of finite word-length on the time response of three recursive filter structures.

CHAPTER 1

INTRODUCTION

The cost reductions and speed improvements achieved in the digital signal processing of discrete-time (sampled-data) signals have been given a great impetus through the availability of MSI and VLSI integrated-circuit technology. Indeed, conventional continuous-time (analog) signal processing or filtering systems have become less advantageous, since digital systems feature potentially lower costs, higher accuracy and stability, and often comparable speed.

Of the methods and techniques used in digital signal processing, linear digital filtering is one of the most important. High-speed digital filters almost invariably utilize fixed-point digital (usually two's complement) operations, and it is desirable to use the shortest word-length (number of bits) to reduce the complexity of the required digital filter.

Currently, there is considerable interest in using available micro-processors, for special-purpose digital filter design. Commercial micro-processors usually have a limited set of choices of word lengths, normally multiples of 4 bits. Obviously, if the digital filter designer can use

8 or 12 bits instead of, say, 16 bits, he has reduced cost and complexity.

The input and output of a digital filter are number sequences, i.e., the input is $x(t) = x(nT)$ and the output is $y(t) = y(nT)$, where T is a constant sampling time interval, i.e., $t = nT$, n an integer. In implementing a digital filter on a digital computer, the input-output relation may be represented by a computational algorithm:

$$y(nT) = \sum_{k=0}^M a_k x(nT-kT) - \sum_{k=1}^N b_k y(nT-kT) \quad (1.1)$$

The algorithm for implementing the filter is then defined by a structure or network consisting of an interconnection of such basic operational elements as delays, adders and multipliers. There exist a number of structures that will result in the same relationship between the input $x(nT)$, and the output $y(nT)$. By structures we mean a choice of state-variables implying a particular circuit topology or choice of operations.

It must be noted that although we tend to say that the algorithm (1.1) is 'linear,' the signal processing when using finite word-lengths (typically 8-16 bits) is no longer linear. It is the effect of these non-linearities that we wish to study by simulation. "Precise prediction of these effects is often difficult and perhaps the best

approach is to simulate the filter behavior on a general purpose computer and study the effect of shortening the word-length empirically" (Wait, 1970, p. 270).

Using the approach outlined above, this thesis will use time-domain digital simulations to explore the effect of finite word-length on filter performance.

CHAPTER 2

DETAILS ON STRUCTURES IMPLEMENTED

Initially the parameters of a recursive digital filter designed for one of the many available structures (Antoniou, 1979, pp. 70-83) are obtained with high accuracy. In the implementation of digital filters with finite word-lengths, however, it is necessary to introduce some finite precision, not only in the coefficients of the filters, but also in the input samples and in the results of arithmetic operations within the filters. The effect of using finite word-length is to introduce error in the filter performance. "The problem of error behavior in digital filtering operations is very involved and is essentially a non-linear problem, because such quantizers as needed in the above description are obviously non-linear devices" (Cappellini, Constantinides and Emiliani, 1978, p. 161).

Finite Word-Length Considerations

Three major sources of error arise from a finite word-length implementation of digital filters. These are:

1. Coefficient quantization errors.
2. Input quantization errors.
3. Product quantization errors.

Most books on digital signal processing provide details of these types of error (e.g., Oppenheim and Schaffer, 1975).

In digital signal processing, as in most numerical algorithms, it is necessary to deal with signed numbers. In the implementation of digital filters, fixed-point two's complement arithmetic is almost always used (Oppenheim and Schaffer, 1975, p. 406).

Number representations, especially in micro-processors, use fixed-point arithmetic. Following addition or multiplication, words are reduced in word-length, usually by rounding (Antoniou, 1979, pp. 282-283). Rounding is a non-linear process and its effect on the realization of a digital filter is to introduce non-linear elements in its structure (Cappellini et al., 1978, p. 166).

Yet another undesirable effect stemming from the use of finite register-length arithmetic is known as zero-input limit-cycle or the "dead-band" effect. It is a consequence of the non-linear quantizers in the feed-back loop of the filter (Antoniou, 1979, pp. 296-303).

The obvious way to avoid finite word-length effects is to use longer word-length, i.e., a large number of bits in the hardware system. This, of course, increases the final implementation cost and, in serial arithmetic implementations, may lower the achievable speed. Thus it is important to use only as many bits as are required (Wait, 1970, p. 269).

There are a number of subtle ramifications of the round-off error problem, which affect the entire filter

design: for example, increasing filter order or decreasing the sampling time, T , makes the filter behavior more sensitive to round-off errors (Wait, 1970).

The Digital Filter Transfer Function

The transfer function of a digital filter, $H(z)$, may be defined as the ratio of the z -transform of the output, $Y(z)$, to the z -transform of the input, $X(z)$, or:

$$H(z) = \frac{Y(z)}{X(z)} \quad (2.1)$$

The exact form of $H(z)$ can be readily derived from the difference equation characterizing the filter, from the filter network, or from the state-space characterization.

Taking the z -transform of (1.1) we have:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M a_k z^{-kT}}{1 + \sum_{k=1}^N b_k z^{-kT}} \quad (2.2)$$

Implementation Considerations

The implementation of a digital filter based upon the system transfer function, $H(z)$, can assume a variety of network configurations or structures.

When implementing a digital filter on a general purpose floating-point computer, e.g., the CDC Cyber 175, it is usually not necessary to pay close attention to the

type of structure used. This is not true when the same filter is to be implemented on a mini- or micro-computer, or using dedicated hardware. Under such circumstances, factors like speed, cost and word-length become closely related to the network structure used.

One consideration in the choice between the different realizations is computational speed. That is, networks with the fewest constant multipliers and the fewest delay operations are often most desirable, since multiplication is a time-consuming operation, and each delay element involves the use of a fetch-store operation. A reduction in the number of constant multipliers means a significant increase in speed. A reduction in the number of delays also means a reduction in memory requirements. On the other hand, the effects of finite register length in the case of micro-computer realizations of digital filters "depend on the structure and it is sometimes desirable to use a structure that does not have the minimum number of multipliers and delays but is less sensitive to finite register length effects" (Oppenheim and Schafer, 1975, p. 149).

Structures Implemented

In this thesis three structures will be studied.

These are:

1. Cascade Form (Direct-II realization).

2. Parallel Form.
3. Normal Form.

Figure 2.1 depicts a block diagram representation of each of the three structures for a second-order filter.

The Cascade and Parallel structures are among the three well-known forms (the third one being the Direct form) that provide a useful measure of comparison with other structures. Also, these three forms require fewer multipliers than any of the other forms. The parameters that define these forms are obtained directly from the system transfer function, $H(z)$, and the computational complexity is minimal.

The Normal realization implementation is described by Mills, Mullis and Roberts (1981, pp. 893-903). Implemented also as second-order sections connected in tandem (called Sectional Optimal Structures), it offers a number of advantages over the Cascade form. Normal structures are also known as "Coupled Resonators" or "Uniform-grid structures" (Gold and Rader, 1969, p. 37).

Cascade forms typically suffer from such drawbacks as large round-off noise, finite register length effects, e.g., overflow oscillations and large coefficient sensitivities. The use of Normal forms greatly reduces such effects. The price for these improvements in performance

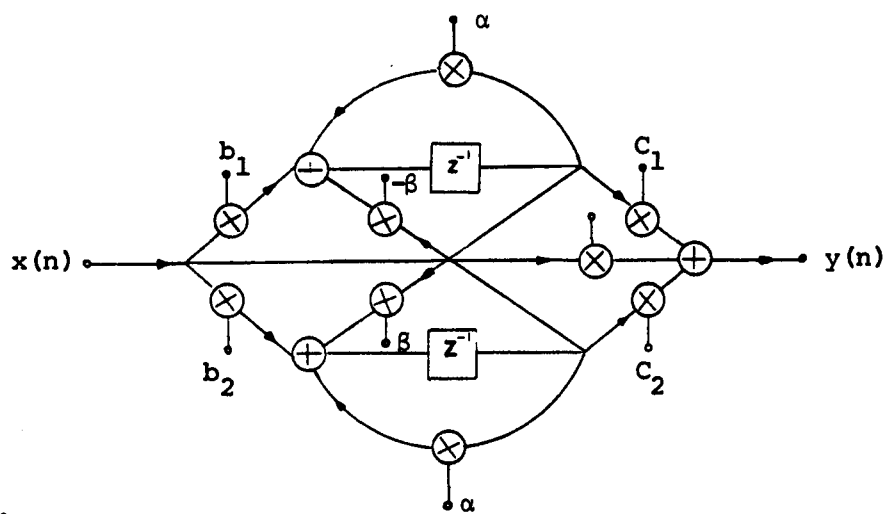
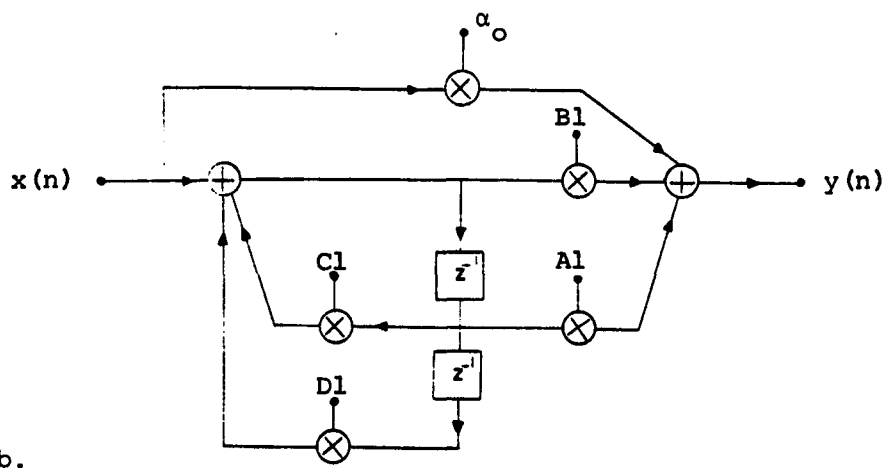
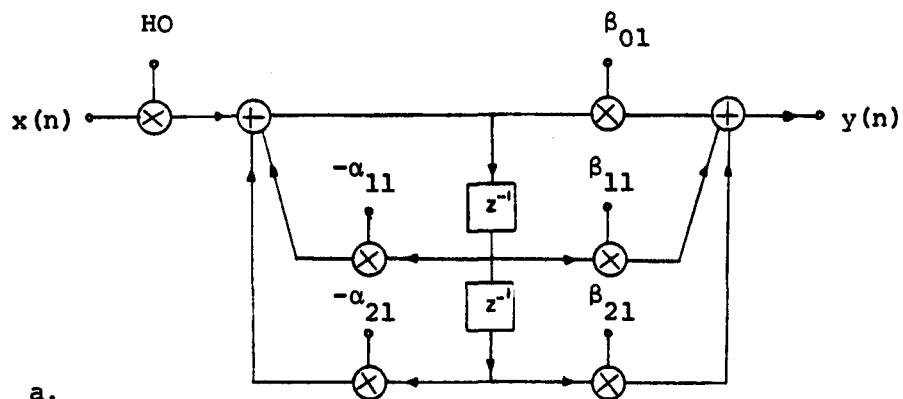


Figure 2.1. Three structures implemented. -- (a) Cascade Form; (b) Parallel Form; (c) Normal Form.

is an increased number of multiplications. A tandem connection of cascade forms requires $3N$ multiplications, where N is the filter order; the Normal form, on the other hand, requires $(N+1)^2$ multiplications (Mills et al., 1981, p. 893).

We assume that a straight-forward design approach will contain second-order sections as building blocks. Such sections can be used to implement any rational function by connecting them in Cascade or in Parallel. The advantage of using second-order sections lies in the relatively low-sensitivity of their transfer properties to element variations. Also they provide flexibility in the choice of composition of the sub-systems and in the order in which the sub-systems are cascaded. This ordering of the second-order sections in the case of Cascade forms becomes important when finite word-length effects are a consideration.

Cascade Form

The system transfer function (2.2) can be expressed as a product of second-order functions of the form:

$$H(z) = H_0 \prod_{k=1}^{[\frac{N+1}{2}]} \frac{\beta_{0k} + \beta_{1k}z^{-1} + \beta_{2k}z^{-2}}{1 + \alpha_{1k}z^{-1} + \alpha_{2k}z^{-2}} \quad (2.3)$$

where $[\frac{N+1}{2}]$ means the largest integer contained in $(\frac{N+1}{2})$.

In writing $H(z)$ in this form the real poles and real zeros have been combined in pairs. If there is an odd

number of real zeros, one of the coefficients β_{2k} will be zero. Likewise, if there are an odd number of real poles, one of the coefficients α_{2k} will be zero.

For $N = 2$ and $\beta_{0k} = 1$, we have:

$$H(z) = HO \frac{1 + \beta_{11}z^{-1} + \beta_{21}z^{-2}}{1 + \alpha_{11}z^{-1} + \alpha_{21}z^{-2}} \quad (2.4a)$$

or

$$H(z) = HO \frac{z^2 + \beta_{11}z + \beta_{21}}{z^2 + \alpha_{11}z + \alpha_{21}} \quad (2.4b)$$

where the gain factor HO represents the gain at a desired frequency.

The block diagram for a Cascade section, Figure 2.1a, is repeated below as Figure 2.2

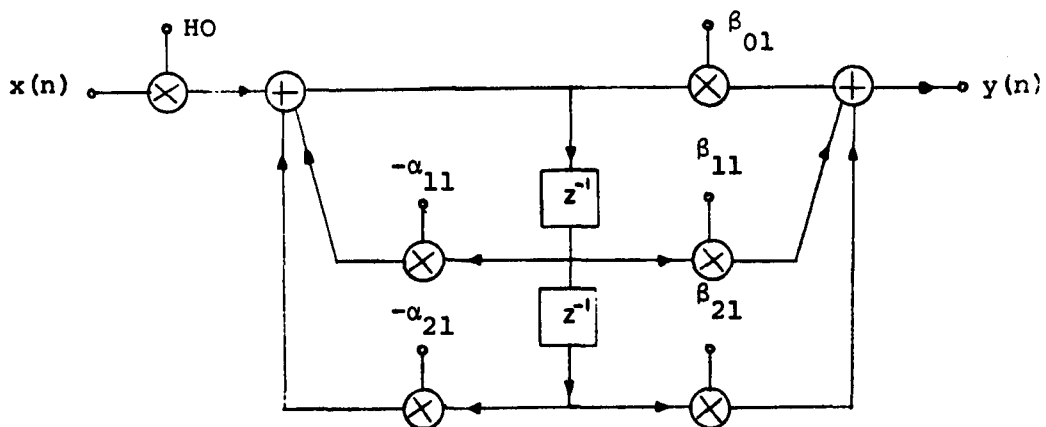


Figure 2.2. Second-Order Cascade Realization.

The ease of implementation of this structure is apparent when the second-order system function is compared with the overall network itself.

Parallel Form

As an alternative to factoring (2.2) as a product of second-order sections, the system function $H(z)$ can be expressed as a partial fraction expansion in the form:

$$H(z) = \gamma_0 + \sum_{j=1}^M \frac{\gamma_{0j} + \gamma_{1j}z^{-1}}{1 + \beta_{1j}z^{-1} + \beta_{2j}z^{-2}} \quad (2.5)$$

where γ_0 is obtained by dividing the numerator and denominator if they are of the same order. Note that (2.5) admits a real pole, by setting $\beta_{2j} = 0$ or $\gamma_{1j} = \beta_{2j} = 0$.

An alternative form of the system function is a partial-fraction expansion of subfunctions of the form:

$$H'(z) = \frac{H(z)}{z} = \gamma_0 + \sum_{k=1}^N \frac{A_k z + B_k}{z^2 + C_k z + D_k} \quad (2.6)$$

where it has been assumed that the coefficient of the denominator quadratic term is unity. Note that for a first-order section with real poles, we have the form:

$$H'(z) = \gamma_0 + \frac{B_k}{z + D_k} \quad (2.7)$$

For a second-order section in parallel form:

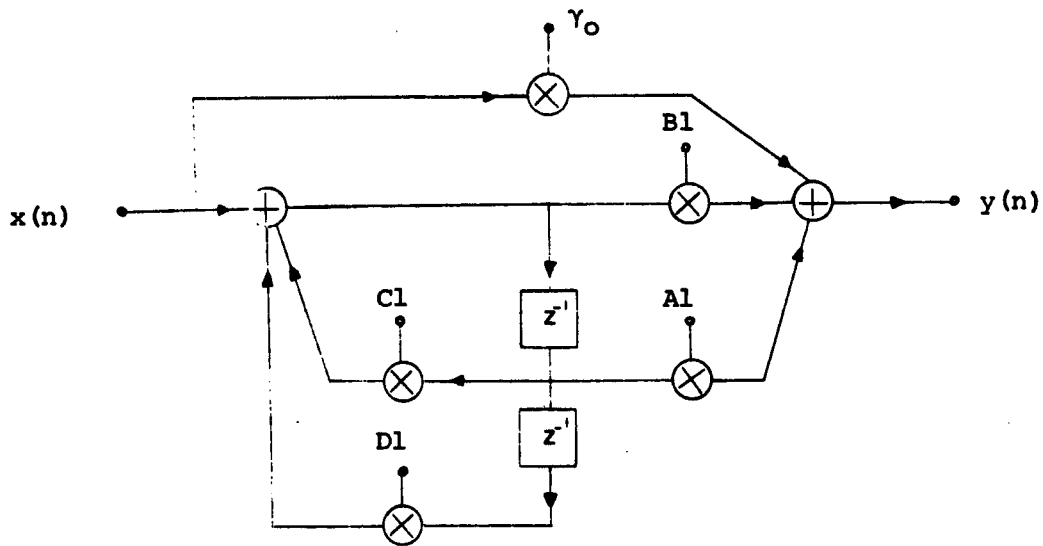


Figure 2.3. Second-Order Parallel Realization.

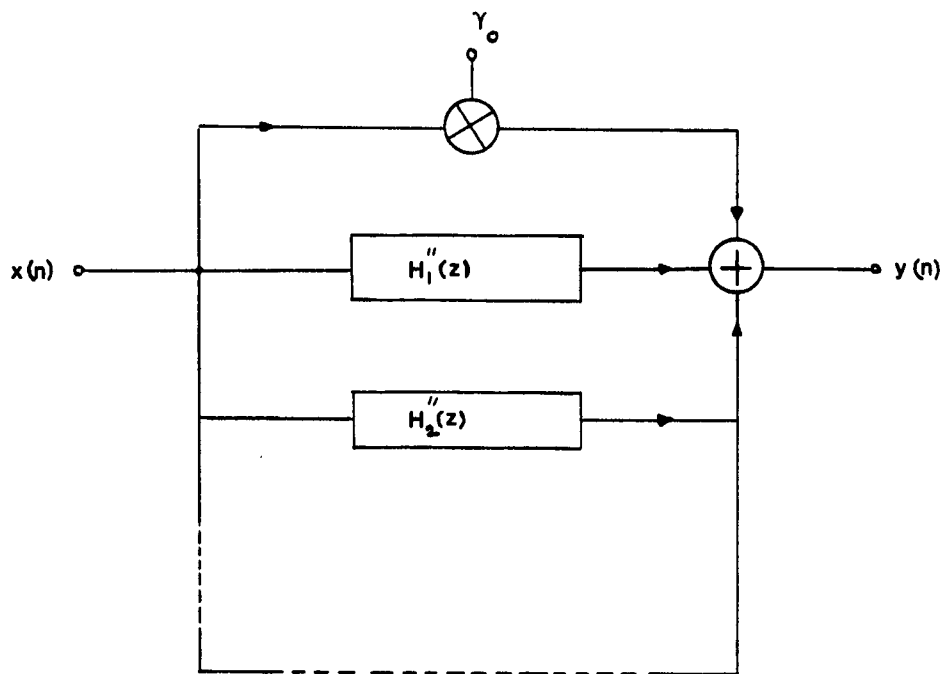


Figure 2.4. Parallel Connection of Second-Order Sections.

$$H_1'(z) = \gamma_0 + \frac{A_1 z + B_1}{z^2 + C_1 z + D_1} \quad (2.8a)$$

$$= \gamma_0 + H_1''(z) \quad (2.8b)$$

which can be implemented as shown in Figure 2.3.

Higher-order system functions can be implemented by connecting each second-order section in Parallel, as shown in Figure 2.4.

Normal Form

The synthesis procedure for Normal structures is based on a specific state-space description. It has been shown (Mills et al., 1981, pp. 895-896; Barnes, 1980, pp. 154-159) that explicit expressions for the Normal structure parameters in terms of the coefficients of the system transfer function can be obtained. Note that ℓ_2 scaling (Jackson, 1970, pp. 176-181) is included as an integral part of the design procedure for dynamic range control.

The block-diagram representation of a general second-order digital filter is shown in Figure 2.1c, and is repeated as Figure 2.5 below. The corresponding state equations are:

$$\left. \begin{aligned} \underline{q}(nT + T) &= \underline{A} \underline{q}(nT) + \underline{b} u(nT) \\ y(nT) &= \underline{C} \underline{q}(nT) + d u(nT) \end{aligned} \right\} \quad (2.9)$$

The state-vector $q(nT)$ is second-order, $y(nT)$ and $u(nT)$ are scalars.

The matrix	\underline{A}	is	2×2
	\underline{b}	is	2×1
	\underline{c}	is	1×2
	\underline{d}	is	1×1

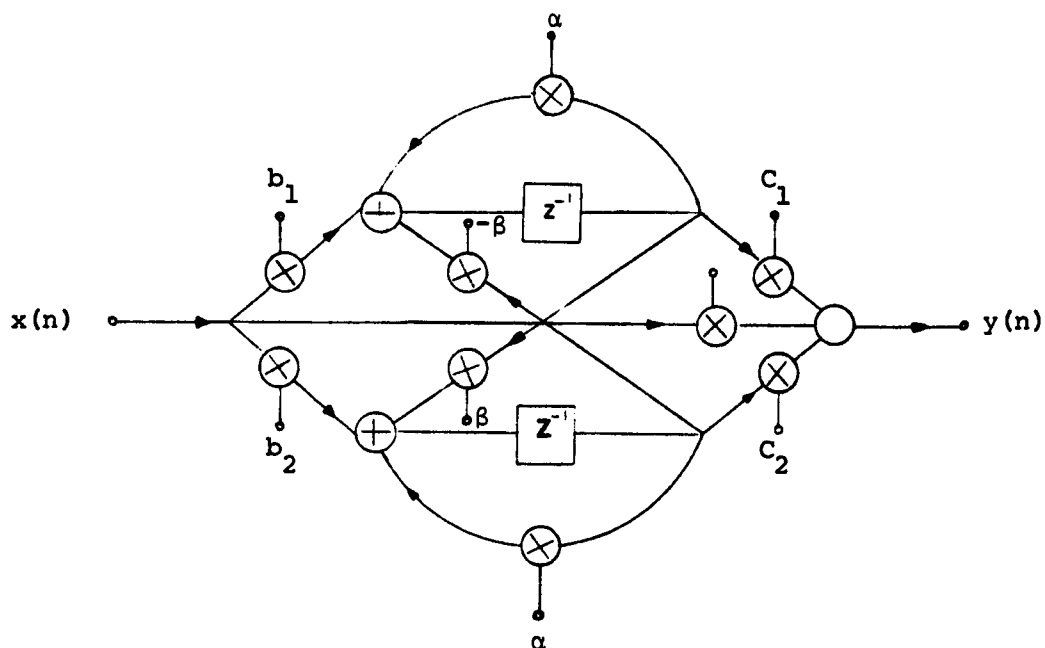


Figure 2.5. Second-Order Normal Realization.

The design of Normal digital filters shall be restricted to the case where the filter poles are complex-conjugates given by $\lambda = \alpha + j\beta$, $\lambda^* = \alpha - j\beta$.

A normal filter with realization $\{\underline{A}, \underline{b}, \underline{c}, \underline{d}\}$ satisfies

$$\underline{\underline{A}}\underline{\underline{A}}^T = \underline{\underline{A}}^T\underline{\underline{A}} \quad (2.10)$$

All Normal second-order filters have the state matrix $\underline{\underline{A}}$ given by:

$$\underline{\underline{A}} = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix} \quad (2.11a)$$

The ℓ_2 scaling defines $\underline{\underline{b}}$ and $\underline{\underline{c}}$.

The design process uses the Normal structure (unscaled) as an intermediate step.

The unscaled realization is denoted as:

$$\{\underline{\underline{A}}_0, \underline{\underline{b}}_0, \underline{\underline{c}}_0, d\}$$

For complex poles $\underline{\underline{A}}_0$ is given by (2.11a). The vectors $\underline{\underline{b}}_0$ and $\underline{\underline{c}}_0$ may be arbitrarily chosen as:

$$\underline{\underline{b}}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \underline{\underline{c}}_0 = r[\cos\phi \quad \sin\phi] \quad (2.11b)$$

Consider the second-order transfer function:

$$H(z) = \frac{q_2 z^2 + q_1 z + q_0}{z^2 - 2\alpha z + \alpha^2 + \beta^2} \quad (2.12)$$

If (2.12) is expressed as:

$$H(z) = d + \frac{q_2 z + q_1}{z^2 + p_1 z + p_2} \quad (2.13)$$

explicit expressions for the Q_2 -scaled Normal realization (denoted with the subscript N) can be derived (Mills et al., 1981, p. 895). For a second-order section with a pair of complex conjugate poles, $\lambda_{1,2} = \alpha \pm j\beta$, these expressions take the form:

$$\underline{A}_N = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}; \quad \underline{b}_N = \psi \begin{bmatrix} \sin(\theta + \frac{\pi}{4}) \\ \cos(\theta + \frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\underline{c}_N = r/\psi \begin{bmatrix} \cos(\phi - \theta - \pi/4) & \sin(\phi - \theta - \pi/4) \end{bmatrix} \left. \vphantom{\underline{c}_N} \right\} (2.14a)$$

$$= \begin{bmatrix} c_1 & c_2 \end{bmatrix}$$

where

$$\psi = 1/\delta \sqrt{2(1-|\lambda|)^2}$$

$$\theta = 1/2 \arg(1 - \lambda^2)$$

$$\phi = \tan^{-1} \left[\frac{q_2}{-q_1 - \alpha q_2} \right]$$

$$r^2 = q_2^2 + \left(\frac{q_1 + \alpha q_2}{\beta} \right)^2 \left. \vphantom{\psi} \right\} (2.14b)$$

More general expressions for higher-order filter transfer functions, implemented as cascaded second-order sections are also available (Mills et al., 1981, p. 900). Normally the gain constant H_0 is distributed equally between each of the second-order sections. δ is a scaling factor. A value of 5 is assumed here.

From the above, we conclude that the computational complexity involved in the evaluation of the respective feed-forward and feed-back coefficients is relatively straightforward in the case of the Cascade form, and fairly complex for the Normal form. The discussion has been restricted to the case of second-order structures, for by cascading or paralleling these second-order sections, we can construct higher-order designs.

The choice of a particular realization depends on a number of factors, some of them application-oriented. Considerable interest has been shown by investigators over the years to arrive at conclusions concerning the advantages and disadvantages of the various structures. Considerable literature exists on the subject (Liu, 1971; Crochiere, 1975). There appears to be consensus among the investigators that although the Cascade form has the least number of multipliers of any realization, and its implementation is straight-forward, it is very sensitive to coefficient word-length, is extremely prone to overflow oscillations, and has relatively large round-off noise. The Normal form, on the other hand, requires more multipliers $((N+1)^2$ compared to $3N$, for the Cascade form, where N is the filter order), but has very low sensitivity to coefficient word-length and the problem of overflows is almost eliminated. Normal forms

have been found to be most suitable for narrow-band low-pass or high-pass filters, whereas Cascade forms are better suited for wide-band filters (Mills et al., 1981, p. 902).

CHAPTER 3

SIMULATION SOFTWARE

This chapter describes the software developed for implementing the three digital filter structures described in the previous chapter.

Starting with the transfer function $H(z)$, the implementation process can be divided into two parts:

1. Evaluate the appropriate parameters that relate directly to the structure under consideration (e.g., the feed-forward, feed-back and direct feed-through coefficients).
2. Implement the filter structure using the parameters evaluated above.

The evaluation of these filter parameters must necessarily precede the implementation of the filter structure itself; therefore, algorithms that outline their evaluation will be treated first.

Parameter Evaluation

The software developed in this work assumes the system transfer function $H(z)$ is in rational form:

$$H(z) = HO \frac{1 + F_1 \cdot z + F_2 \cdot z^2 + \dots + F_M \cdot z^M}{1 + E_1 \cdot z + E_2 \cdot z^2 + \dots + E_N \cdot z^N} \quad (3.1)$$

where $M \leq N$.

An algorithm describing the evaluation of parameters for each of the three different structures is outlined below. The input data to each of the algorithms are the coefficients of the numerator and denominator polynomials, i.e., the F_i and E_j and the gain constant HO in (3.1).

Cascade (Direct-II) Form

The Cascade (Direct-II) form of Figure 2.1a presents the most straight-forward method for obtaining the filter parameters. Once the numerator and denominator polynomials have been factored into second-order terms, the feed-forward and feed-back coefficients of the filter are the coefficients of these terms (Oppenheim and Schaffer, 1975, p. 152).

The algorithm works as follows:

Step 1: Separately find all roots of the numerator and denominator polynomials.

Step 2: Form second-order sections, for both the numerator and denominator polynomials using the roots evaluated in Step 1. Two cases arise:

- a. A pair of complex conjugate roots: $\lambda_{1,2} = \alpha \pm j\beta$
- b. A pair of real roots $\lambda_1 = \alpha_1$ and $\lambda_2 = \alpha_2$

In either case a quadratic of the form

$$z^2 + Pz + Q$$

can be formed where:

$$\text{For case (a): } P = 2\alpha \quad \text{and} \quad Q = \alpha^2 + \beta^2 \quad (3.2a)$$

$$\text{For case (b): } P = \alpha_1 + \alpha_2 \quad \text{and} \quad Q = \alpha_1\alpha_2 \quad (3.2b)$$

Any second-order section in the Cascade (Direct-II)

form is expressed as:

$$H(z) = HO \frac{z^2 + \beta_1 z + \beta_2}{z^2 + \alpha_1 z + \alpha_2} \quad (3.3)$$

where β_1 and β_2 are the feed-forward coefficients and α_1 and α_2 are the feed-back coefficients. HO is a constant that scales the input.

Subroutine DSETUP accomplishes the task outlined in Steps 1 and 2 above. Two supplementary routines, SETUP1 and SETUP2 are used to separately factor the numerator and denominator polynomials. Subroutine PROOT (Shapiro, 1965) finds the roots of the polynomials. Listing of these routines are available in Appendix B.

Parallel Form

To implement the Parallel form of Figure 2.1b, the transfer function $H(z)$ must be expressed as a partial fraction:

$$H(z) = \gamma_0 + \sum_{k=1}^N \frac{\gamma_{0k} z + \gamma_{1k}}{z^2 + \beta_{1k} z + \beta_{2k}} \quad (3.4)$$

The term γ_0 is obtained by dividing the numerator and denominator polynomials of $H(z)$ if $M = N$ in (3.1) (Peled and Liu, 1974, p. 456).

In (3.4), the numerator terms γ_{0k} and γ_{1k} are the feed-forward coefficients, β_{1k} and β_{2k} are the feed-back coefficients.

The algorithm used is:

Step 1: If the order of the numerator and denominator is the same (i.e., $M = N$), divide them out, to obtain:

$$H(z) = \gamma_0 + \frac{G_1 + G_2 \cdot z + G_3 \cdot z^2 + \dots + G_M z^{M-1}}{E_1 + E_2 \cdot z + E_3 \cdot z^2 + \dots + E_{N+1} z^N}$$

$$H(z) = \gamma_0 + H_1(z) \quad (3.5)$$

Step 1: Make a partial-fraction expansion of $H_1(z)$.

Two cases arise:

- a. A pair of complex-conjugate roots of the denominator polynomial ($\lambda_{1,2} = \alpha \pm j\beta$) with a pair of conjugate residues ($R_{1,2} = x \pm jy$).

One such second-order factor would be:

$$H_{11}(z) = \frac{N_{11}(z)}{D_{11}(z)} = \frac{Az + B}{z^2 + 2\alpha z + (\alpha^2 + \beta^2)} \quad (3.6a)$$

where

$$A = 2X \quad \text{and} \quad B = -2\alpha x - 2\beta Y \quad (3.6b)$$

- b. A pair of real roots of the denominator polynomial ($\lambda_1 = \alpha_1$, $\lambda_2 = \alpha_2$) with real distinct residues ($R_1 = X_1$, $R_2 = X_2$). A second-order factor would be:

$$H_{12}(z) = \frac{N_{12}(z)}{D_{12}(z)} = \frac{Az + B}{z^2 + (\alpha_1 + \alpha_2)z + \alpha_1\alpha_2} \quad (3.7a)$$

where

$$A = X_1 + X_2 \quad \text{and} \quad B = \alpha_2 X_1 + \alpha_1 X_2 \quad (3.7b)$$

Comparing one section of (3.4) with either (3.6) or (3.7), a direct correspondence between the coefficients is clear.

Subroutine PSETUP accomplishes the task of forming the filter parameters. Subroutine PFXSD (Huelsman, 1972, pp. 602-604) is used for making the partial-fraction expansion. Root-finding routines ROOT (Huelsman, 1972, pp. 870-871) and PROOT (Shapiro, 1965) are also used. A listing of these routines is available in Appendix B.

Normal Form

The Normal form, Figure 2.1c, is restricted to the case of complex-conjugate pole-pairs, with quadratic numerator terms. A general second-order section in Normal form would have a transfer function:

$$H(z) = \frac{q_2 z^2 + q_1 z + q_0}{z^2 - 2\alpha z + \alpha^2 + \beta^2} \quad (3.8)$$

where α and β are the real and imaginary parts of the complex conjugate pole-pair (Mills et al., 1981).

This procedure assumes a system transfer function $H(z)$ is available as a product of second-order sections.

The algorithm for evaluating the parameters is:

Step 1: Find the roots of the numerator and denominator polynomials.

Step 2: Form quadratic terms of the form shown in (3.8).

Step 3: Divide the numerator and denominator polynomials of each second-order section to obtain:

$$H(z) = d + \frac{q_2 z + q_1}{z^2 + p_1 z + p_2} \quad (3.9)$$

Step 4: Form the coefficients that define the structure, as contained in the state-space description, \underline{A}_N , \underline{b}_N and \underline{c}_N in (2.14).

Subroutine NSETUP accomplishes the task of forming the parameters in this package. It uses routine PROOT to find the roots of the numerator and denominator polynomials, and thus to find the filter parameters. A listing of this routine is available in Appendix B.

Filter Implementation

Once the filter parameters have been evaluated, the next step is to implement the appropriate filter structure.

The basic arithmetic operations involved in the implementation of digital filters are addition, multiplication, and delay. These operations, therefore, form the basic building blocks for implementing filter structures.

Explicit routines developed by Dr. J. V. Wait (TAD Users Manual, in preparation) for simulating fixed-point arithmetic were used in the simulations. These routines, called SHADD, SMULT, OVER, QUAN, IQUAN and FRAC are capable of adding or multiplying two problem variables, checking for register overflow, and quantizing numbers to a specific register length, using rounding. They also simulate one's- and two's-complement and signed-magnitude implementations, and binary (left-right shift) scaling.

These routines are in the form of function subprograms to be directly called by the user.

Function SHADD

SHADD is the summing routine. It has four arguments, X, Y, NSHIFT and I. X and Y are the two problem variables to be added, NSHIFT is a right shifting factor (not used in the simulation process), and I is an index on overflow in the quantizer. NBA is the number of bits used by the quantizer for rounding. If NBA is positive, the sum of X and Y is rounded to NBA bits. NBA has been set to NBITS.

For example, to add two problem variables A and B and return a sum S1, the specific command is:

$$S1 = \text{SHADD} (A, B, 0, 1)$$

Function SMULT

SMULT is the multiplication routine. It has three arguments: X, Y, and I, where X and Y are the two problem variables to be multiplied, I is an index on the quantizer to keep track of overflow. Each problem variable is rounded to NBM bits before being multiplied together to give a product with 2NBM bits. During the quantizing process the problem variable is checked to insure that its value lies within the range $-1 < X < (1 - \frac{1}{2^{NBM-1}})$, otherwise an overflow message is printed out. NBM has been set to NBITS.

For example, to multiply two numbers A and B and return a product P1, the specific command would be:

$$P1 = \text{SMULT} (A, B, I)$$

Symbolically the operations of addition and multiplication can be represented as shown in Figure 3.1.

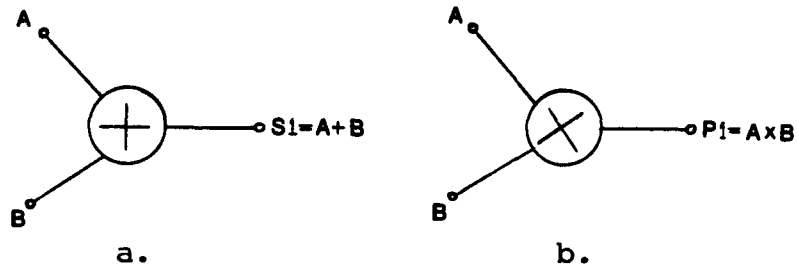


Figure 3.1. Symbolic representation of an Adder and a multiplier. -- (a) An Adder; (b) A Multiplier.

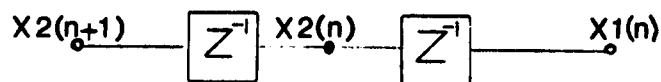
Implementing the Delay

Another basic building-block in the implementation of digital filters is the delay element. A unit delay can be implemented very easily in FORTRAN. A memory register whose entire contents are shifted on command (e.g., a time increment) acts like a delay. Two such shift-registers connected in tandem and working on a "shift and store" principle can be used to implement a pair of delays.

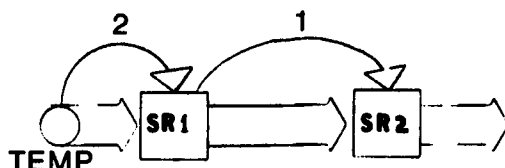
The principle of operation of a pair of delays is demonstrated below in Figure 3.2.

Figure 3.2a shows 2 delays in Cascade with the 2 associated state variables, $X1$ and $X2$.

In a recursive system, the next state of $X2$ or $X2(n+1)$ is a function of the present state $X2(n)$ and/or $X1(n)$ along with possibly the input. Similarly, the output of a recursive system depends on at least one or more of



a.



b.

Figure 3.2. Simulating a tandem connection of two delays.
 -- (a) Symbolic diagram; (b) Flow diagram.

these states. Once these computations have been performed, the state $X1(n)$ is not needed.

Figure 3.2b shows how two memory registers SR1 and SR2 can be used to simulate delay elements. SR2 contains the state $X1(n)$ and SR1 the state $X2(n)$. TEMP is a temporary location for retaining $X2(n+1)$.

Prior to incrementing the time index n , the contents of SR1 and SR2 need to be changed. A two-step procedure, outlined in Figure 3.2b, accomplishes this:

Step 1: Shift the contents of SR1 to SR2.

Step 2: Transfer contents of temporary location TEMP to SR1.

Before incrementing the time index n , the procedure outlined in Steps 1 and 2 is repeated.

Example

The operation of the addition and multiplication routines, SHADD and SMULT, and a delay element are shown below for a first-order filter section.

Suppose the filter has a transfer function $H(z)$ given by:

$$H(z) = HO \frac{z + \alpha}{z + \beta}, \quad HO, \alpha \text{ and } \beta < 1 \quad (3.10)$$

A block diagram description of this section is as shown in Figure 3.3 below:

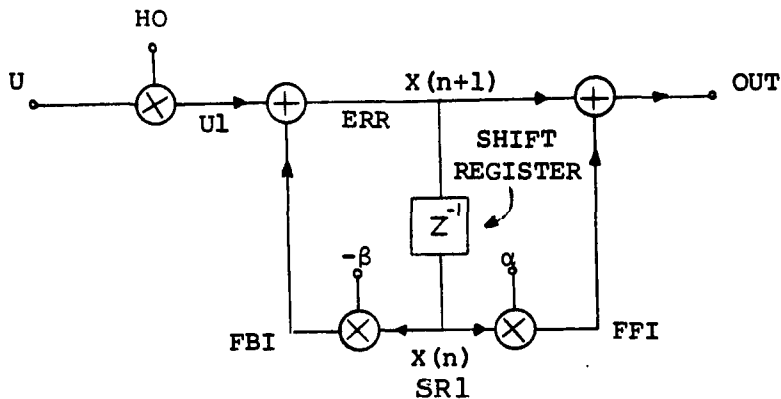


Figure 3.3. First-Order Section.

The outputs of the adders, multipliers and delay are indicated in Figure 3.3, and are defined below:

$$U1 = \text{SMULT}(U, HO, 1) \quad [\text{Input scaled by } HO]$$

$$FBI = \text{SMULT}(SR1, -\beta, 2) \quad [\text{Feed-back coefficient multiplied by present system-state}]$$

$FF1 = SMULT (SRI, \alpha, 3)$ [Feed-forward coefficient
multiplied by present
system-state]

$ERR = SHADD (U1, FBI, 0, 1)$ [Error term or the next
system-state]

$OUT = SHADD (ERR, FF1, 0, 2)$ [Output based upon present
and next state]

$SRI = ERR$ [Transfer ERR content to
SRI. This effectively
delays the next state to
obtain the present state]

No explicit time dependence has been shown above. A time index becomes necessary in an actual filter simulation, for computing the output OUT. If more than one section is connected together in Cascade or in Parallel, an additional index is required.

Listings of the simulation routines for the three structures implemented are available in Appendix B (sub-routines CASCAD, PARLEL and NORMAL).

In the simulation of the Cascade and Parallel forms no scaling of the filter parameters is required to prevent overflows. The input to the first second-order section in the case of these two realizations is scaled down by the gain term HO, which forms part of the system transfer

function. An additional scaling factor given by $2^{-\text{ISHIFT}}$, where ISHIFT is a positive integer, is used to further scale down the input. Note, in typical FORTRAN compilers, multiplying by a power of 2.0 is effectively a binary left-shift and eliminates conversions of variable type from fixed to floating.

Because the Normal realization has ℓ_2 scaling included as part of the parameter evaluation process, no input scaling is used in its implementation.

The Register Overflow Problem

In spite of the precautions listed above, the numerical values of some of the feed-forward and feed-back coefficients in all three simulations will occasionally cause the multipliers to overflow. Since each such coefficient is the input to a multiplier, and must be less than one in magnitude, some way to handle such coefficients needs to be devised. The method adopted here is straight-forward.

Example:

Suppose a filter has a z-domain transfer function:

$$H(z) = 0.02475 \frac{z^2 + 2z + 1}{z^2 - 1.887z + 0.986} = HO \frac{z^2 + A_1z + A_2}{z^2 + B_1z + B_2} \quad (3.11)$$

If this transfer function were implemented in Cascade form, it becomes immediately obvious that the coefficients

A_1 , A_2 and B_1 may not be directly used as inputs to multipliers.

A block diagram of such a filter would look as shown in Figure 3.4 below:

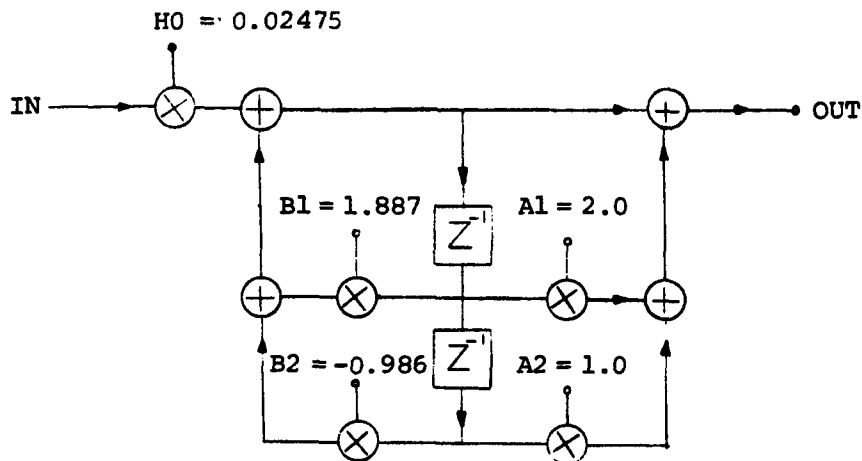


Figure 3.4. Second-Order Section in Cascade Form Based on (3.11).

If each of the coefficients with magnitudes greater than or equal to one are divided by an integer number large enough to reduce their magnitude to a value less than one, the filter implementation becomes possible.

If the coefficients A_2 and B_1 are divided by 2 and A_1 by 3 and additional adders provided so that the final result remains unaffected, the structure would look as shown in Figure 3.5.

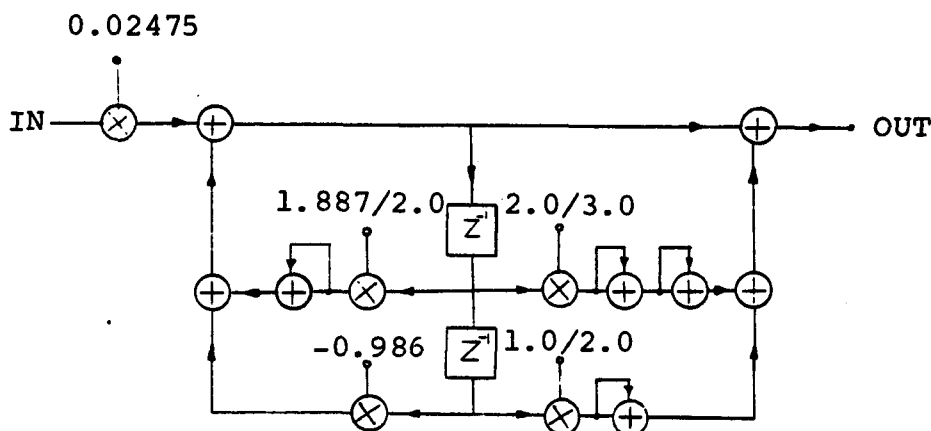


Figure 3.5. Second-Order Section in Cascade Form with Additional Adders to Prevent Overflow.

Note that the above method suggests:

$$3X = (1 + 2)X \quad \text{where } X \text{ is any multiplying factor}$$

or one addition and a shift are needed.

The price paid is a two-fold increase in the number of adders as compared to the unrealizable structure shown in Figure 3.4.

In general, if a coefficient α has magnitude in excess of unity and is divided by an integer I , then $I-1$ additional adders will need to be included in the structure to enable the coefficient α to be accommodated within the structure and prevent overflows at multiplier inputs.

The operation of addition compared to that of multiplication requires a fraction of the time and therefore does not significantly increase the computational time.

The same technique has been used in the implementation of the Parallel and Normal forms. It must be pointed out that the method outlined will prevent overflows only if the filter coefficients lie within the limits for which the structure has been designed. Table 3.1 gives the maximum allowable values for the coefficients of the three structures. These allowable limits on coefficient values were obtained during this study by simulating a variety of narrow- and wide-band digital filter transfer functions of up to sixth-order.

To investigate the system response, three inputs will be used. These are:

1. A step input, $u(n) = A$, of magnitude A to be specified by the user.
2. A sinusoidal input, $u(n) = A \sin(n\omega T)$, where A and ωT are under user control.
3. A broad-band random input generated as follows:

A random number generating function RANF (which returns a real random number between 0 and 1) and a user supplied argument X , are combined in routine RANDOM such that:

$$\text{Random}(X) = X \quad \text{if} \quad \text{RANF} \geq 0.5$$

$$\text{Random}(X) = -X \quad \text{if} \quad \text{RANF} < 0.5$$

Table 3.1. Table with maximum allowable values for coefficients in a second-order section for the three structures.

Structure Type	Coefficients	Magnitude of Maximum Allowable Value
Cascade	A0	0.9
	A1	2.9
	A2	1.9
	B1	2.9
	B2	0.9
Parallel	γ_0	0.9
	γ_{01}	1.9
	γ_{11}	0.9
	β_{11}	2.9
	β_{21}	0.9
Normal	SIGMA	0.9
	OMEGA	0.9
	B1	0.9
	B2	0.9
	C1	5.9
	C2	5.9

The simulation software described here is by no means unique. Various programs for designing digital filters that meet frequency and time-domain specifications are available in the literature (e.g., FAD Users Manual, Version 1, 1981; Antoniou, 1979, Appendix B; Programs for Digital Signal Processing, 1979, Sec. 6.4). Of the three sources cited, Antoniou lists a time-domain analysis program for which the system specifications are made in a way similar to those adopted here. However, the program is implemented specifically for a 16 bit mini-computer (HP 9825A) using HPL language and no control over the word-length is provided to the user. It implements only the Cascade and Parallel forms. The simulation software developed in this thesis, on the other hand, uses standard FORTRAN, making it portable. Also, it gives control to the user over the word-length used in the simulations, and includes a third structure, the Normal form, allowing a more effective comparison between structures.

In this chapter, details of the various software components, how they relate to the implementation of a filter, and a simple method of avoiding overflows at inputs to multipliers have been presented. In the following chapter the software will be tested using simulated fixed-point filter transfer functions and the effect of round-off on the time response will be investigated.

CHAPTER 4

TEST EXAMPLES

To illustrate the usefulness of the software, three examples will be considered in this chapter. The first example is a second-order, equal ripple, low-pass Chebyshev filter; the second is a sixth-order, low-pass filter; and the third example is a second-order band-pass filter.

Example 1

A second-order, 1.0 dB, equal-ripple magnitude (Chebyshev) low-pass filter with a passband of 0 to 1 rad/s has a s-domain transfer function (Huelsman, 1980, pp. 29-39) given by:

$$H(s) = \frac{1}{s^2 + 1.097734s + 1.102510} \quad (4.1)$$

Using the Matched-z transformation (Antoniou, 1979, p. 175), with a sampling time $T = 1$ sec., the corresponding z-domain transfer function is:

$$H(z) = \frac{0.15278491 (z^2 + 2z + 1)}{z^2 - 0.72248469z + 0.33362422} \quad (4.2)$$

where the gain term is chosen for a D.C. gain of unity.

Using a Cascade form, Figure 2.1a, the filter

response to a step-input of magnitude 0.5 is investigated. Figure 4.1 shows a typical output.

The first simulation run, which uses a word-length (NBITS) of 32 bits, will be used as a benchmark. Subsequent simulations use 24, 16, 12, and 8 bits.

Figure 4.1a shows a listing of the input parameters; Figure 4.1b shows the feed-forward and feed-back coefficient for the Cascade form. Note that these coefficients have been obtained using high precision and no quantization is involved. Figure 4.1c gives a listing of the filter response to a step-input of magnitude 0.5; Figure 4.1d plots the response, and Figure 4.1e plots the error-function defined as:

$$\text{Error}(n) = y(n)_{32 \text{ bits}} - y(n)_{\text{NBITS}} \quad (4.3)$$

The results show that as the word-length NBITS is reduced, the response no longer approaches a correct steady-state. For NBITS = 16 and 12, the final values of the output show errors of 0.036% and 0.879%, respectively. For NBITS = 8, the final output becomes oscillatory, with a maximum error of 3.1%. These errors in response can be explained by noting the effect of rounding on the filter input. For the Cascade form, the 0.5 step-input is scaled down by the gain factor $H_0 = 0.15278490$. Before 0.5 and 0.15278490 and multiplied by function SMULT, they are

Figure 4.1. The second-order, 1.0 dB equal-ripple magnitude low-pass digital filter function (4.2) is subjected to a step-input of magnitude 0.5 for a Cascade form.

- a. Simulation parameters.
- b. Feed-forward and feed-back coefficients. These have been obtained using high precision.
- c. A listing of the response as the word-length is reduced.
- d. A plot of the response versus time for the simulations using varying word-length. Note that for NBITS = 8, curve marked 1, the response oscillates about the final value.
- e. A plot of the error function (4.3) versus time as the word-length is changed. Graphs marked 1 are for 24 and 16 bits; graph marked 2 is for 12 bits; and graph marked 3 is for 8 bits.

SIMULATION PARAMETERS

TMAX= 50.0 M= 2 N= 2 NBITS=32 ISHIFT= 0
U=1.0 ARG1= .5 ARG2= 1.0 HQ= .152784900 GRAF=1.0

COEFFICIENTS OF NUMERATOR POLYNOMIAL

F(1)= 1.00000000
F(2)= 2.00000000
F(3)= 1.00000000

COEFFICIENTS OF DENOMINATOR POLYNOMIAL

E(1)= .33362420
E(2)= -.72248460
E(3)= 1.00000000

NUMBER OF SECOND ORDER SECTIONS= 1

STRUCTURE TYPE=DIRECT

NUMBER OF BITS (NBITS) FOR 2ND. RUN= 24

NUMBER OF BITS (NBITS) FOR 4TH. RUN= 12

NUMBER OF BITS (NBITS) FOR 3RD. RUN= 16

NUMBER OF BITS (NBITS) FOR 5TH. RUN= 8

Figure 4.1a.

FEED-FORWARD AND FEED-BACK COEFFICIENTS

A0	A1	A2	B1	B2
1.0000000	2.0000000	1.0000000	-.7224846	.3336242

Figure 4.1b.

TIME	32 bits	24 bits	16 bits	12 bits	8 bits
1.00	.0763924	.0763924	.0763550	.0761719	.0781250
2.00	.2843697	.2843693	.2843933	.2841797	.2890625
3.00	.4855362	.4855357	.4855042	.4853516	.4765625
4.00	.5614896	.5614889	.5614014	.5610352	.5703125
5.00	.5492508	.5492499	.5491028	.5478516	.5859375
6.00	.5150685	.5150677	.5149231	.5131836	.5468750
7.00	.4944555	.4944545	.4943542	.4936523	.5234375
8.00	.4909670	.4909661	.4909973	.4897461	.5078125
9.00	.4953236	.4953226	.4952393	.4921875	.5078125
10.00	.4996350	.4996343	.4994812	.4960938	.5156250
11.00	.5012964	.5012960	.5010986	.4970703	.5078125
12.00	.5010584	.5010582	.5010071	.4960938	.5078125
13.00	.5003322	.5003321	.5003967	.4956055	.5078125
14.00	.4998869	.4998868	.4999390	.4956055	.5156250
15.00	.4998074	.4998070	.4997559	.4956055	.5078125
16.00	.4998986	.4998986	.4998169	.4956055	.5078125
17.00	.4999910	.4999907	.4998169	.4956055	.5078125
18.00	.5000273	.5000268	.4998169	.4956055	.5156250
19.00	.5000227	.5000218	.4998169	.4956055	.5078125
20.00	.5000073	.5000064	.4998169	.4956055	.5078125
21.00	.4999977	.4999971	.4998169	.4956055	.5078125
22.00	.4999959	.4999954	.4998169	.4956055	.5156250
23.00	.4999978	.4999971	.4998169	.4956055	.5078125
24.00	.4999998	.4999990	.4998169	.4956055	.5078125
25.00	.5000006	.4999999	.4998169	.4956055	.5078125
26.00	.5000005	.5000001	.4998169	.4956055	.5156250
27.00	.5000002	.5000001	.4998169	.4956055	.5078125
28.00	.5000000	.5000001	.4998169	.4956055	.5078125
29.00	.4999999	.5000001	.4998169	.4956055	.5078125
30.00	.5000000	.5000001	.4998169	.4956055	.5156250
31.00	.5000000	.5000001	.4998169	.4956055	.5078125
32.00	.5000000	.5000001	.4998169	.4956055	.5078125
33.00	.5000000	.5000001	.4998169	.4956055	.5078125
34.00	.5000000	.5000001	.4998169	.4956055	.5156250
35.00	.5000000	.5000001	.4998169	.4956055	.5078125
36.00	.5000000	.5000001	.4998169	.4956055	.5078125
37.00	.5000000	.5000001	.4998169	.4956055	.5078125
38.00	.5000000	.5000001	.4998169	.4956055	.5156250
39.00	.5000000	.5000001	.4998169	.4956055	.5078125
40.00	.5000000	.5000001	.4998169	.4956055	.5078125
41.00	.5000000	.5000001	.4998169	.4956055	.5078125
42.00	.5000000	.5000001	.4998169	.4956055	.5156250
43.00	.5000000	.5000001	.4998169	.4956055	.5078125
44.00	.5000000	.5000001	.4998169	.4956055	.5078125
45.00	.5000000	.5000001	.4998169	.4956055	.5078125
46.00	.5000000	.5000001	.4998169	.4956055	.5156250
47.00	.5000000	.5000001	.4998169	.4956055	.5078125
48.00	.5000000	.5000001	.4998169	.4956055	.5078125
49.00	.5000000	.5000001	.4998169	.4956055	.5078125
50.00	.5000000	.5000001	.4998169	.4956055	.5156250

Figure 4.1c.

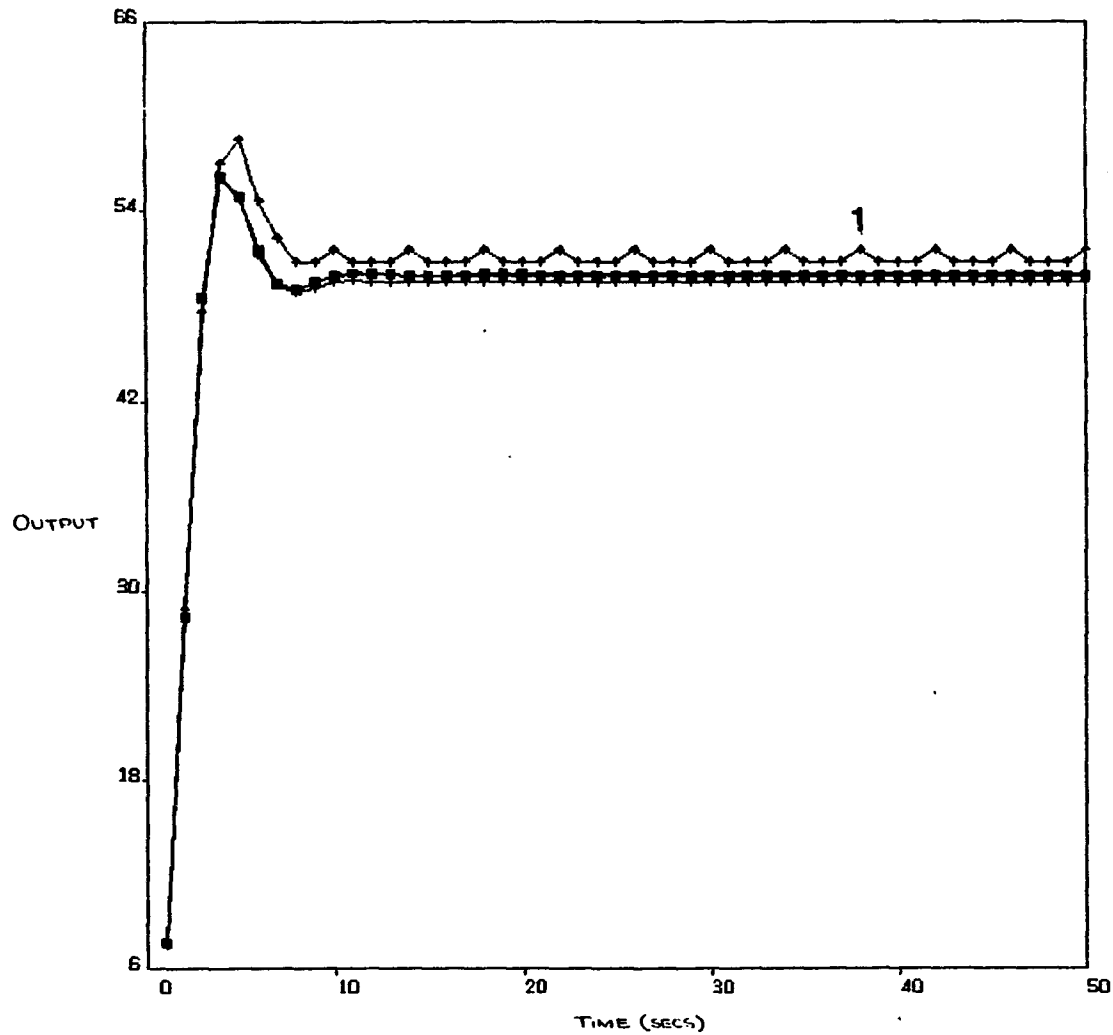


Figure 4.1d.

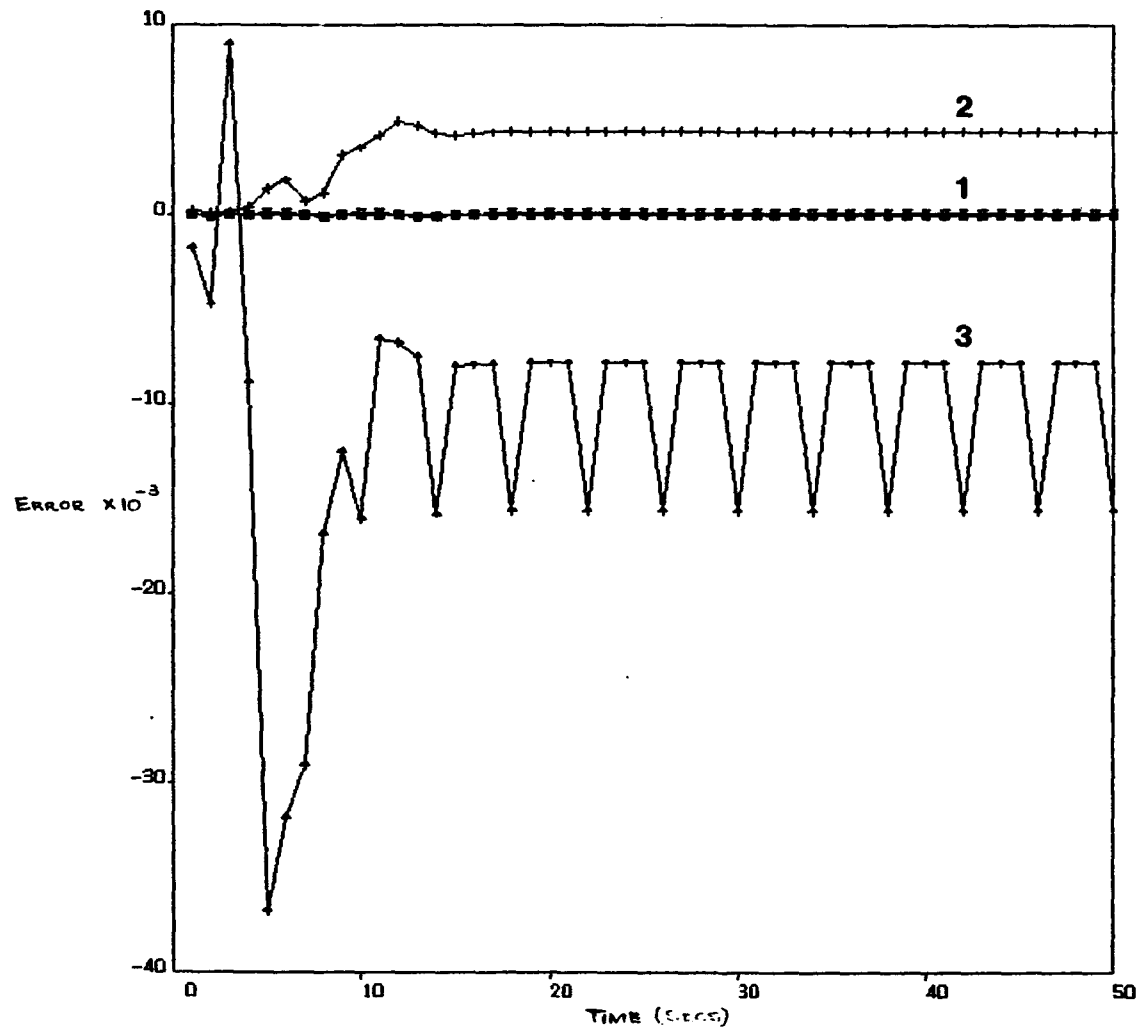


Figure 4.le.

quantized to the respective word-length. Figure 4.2 shows the section of the Cascade form, Figure 2.1a, being referred to above.

Quantized values of HO for 16, 12, and 8 bits are:

$$Q[HO]_{16 \text{ bits}} = 0.1527709961$$

$$Q[HO]_{12 \text{ bits}} = 0.1525878906$$

$$Q[HO]_{8 \text{ bits}} = 0.15234375$$

where $Q[.]$ is the quantized value of HO for the indicated number of bits. The step-input of 0.5 is not affected by quantization.

From the above, it becomes clear that the input to the filter at the point marked X on Figure 4.2 will be different for each word-length. The values of the product, P_X , at point X for 16, 12, and 8 bits are:

$$P_X \text{ 16 bits} = 0.076385498$$

$$P_X \text{ 12 bits} = 0.076416015$$

$$P_X \text{ 8 bits} = 0.078125$$

As a result, the filter output, for 16, 12, or 8 bits, cannot be expected to reach a final value of 0.5; hence the percentage errors noted earlier are obtained. The oscillations obtained for 8 bits, Figure 4.1d, are a consequence of

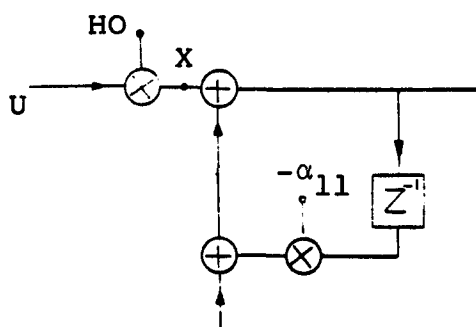


Figure 4.2. A section of the Cascade form.

using quantization in fixed-point arithmetic. The effect of such quantizations is to introduce non-linearities in the filter structure, causing a periodic error at the output. Such limit-cycle effects arise because quantization forces the poles of the system to move to the unit circle (Oppenheim and Schaffer, 1975, pp. 418-423).

The digital filter function (4.2) was driven by two additional inputs: a sinusoidal input, $u(n) = 0.5 \sin(n)$ and the random broad-band input described in Chapter 4. Figure 4.3 shows the response to these inputs.

The Parallel and Normal forms were also tested for the same three inputs. Figure 4.4 shows the responses for the Parallel form, and Figure 4.5 for the Normal form. To illustrate better the effect of reducing the word-length, plots of the error function (4.3) are also included.

An analysis of the three structures in terms of the maximum percentage of relative error, $|e(n)_{\max}|$, based upon the three inputs, is made. The relative error function, $|e(n)_{\max}|$, is defined as:

Figure 4.3. Response and Error function plots for simulations of the digital filter function (4.2); a Cascade form is used. -- Each response plot has results of simulations for 32, 24, 16, 12, and 8 bits. Each error function plot is according to (4.3). Driving functions used are a sinusoidal input $u(n) = 0.5 \sin(n)$ and a random broad-band input (see text).

- a. Response to a sinusoidal input.
- b. Error function versus time for the response plotted in Figure 4a above.
- c. Broad-band random function used as input.
- d. Response to a broad-band random input.
- e. Error function versus time for the response plotted in Figure 4c above.

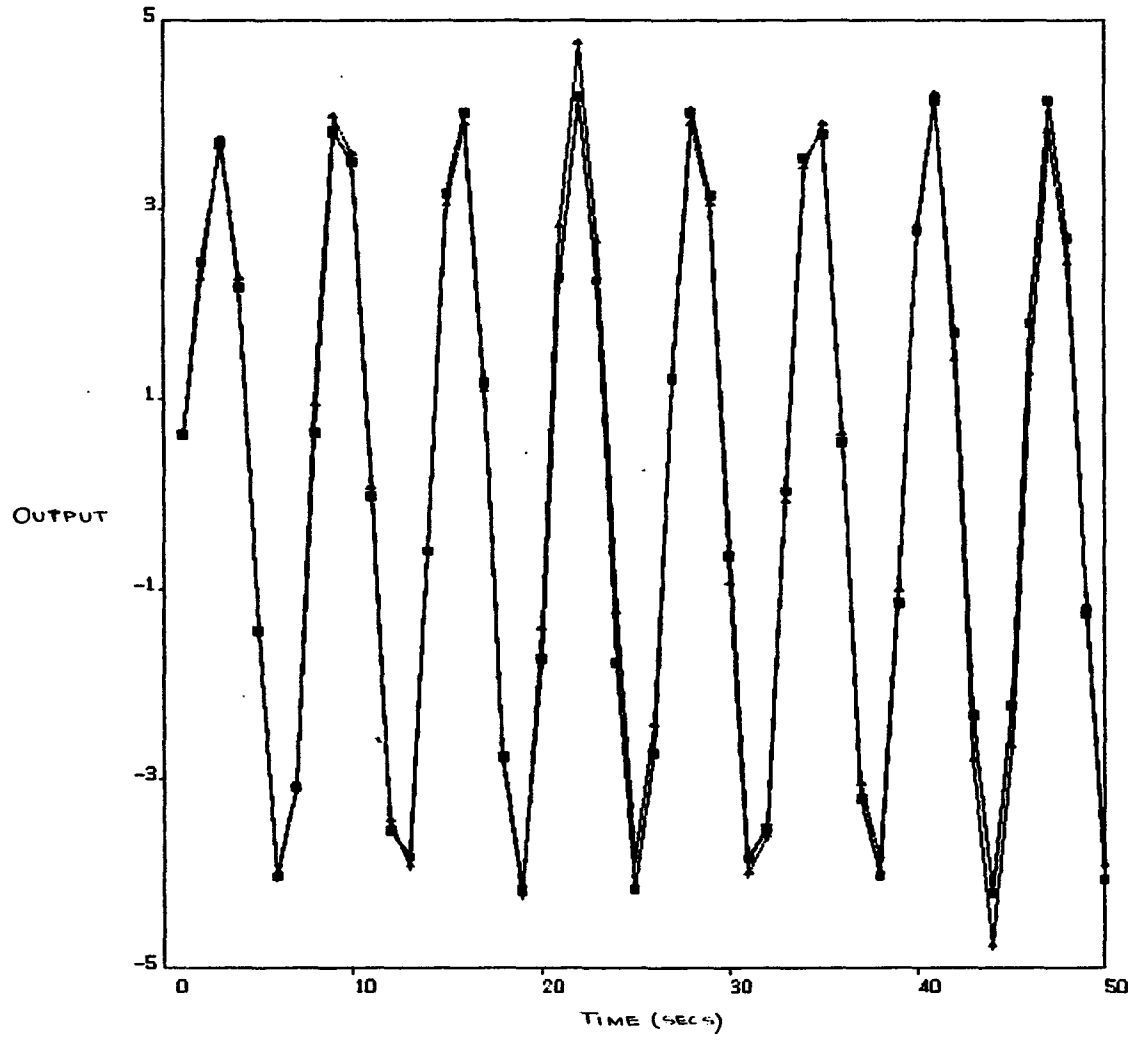


Figure 4.3a.

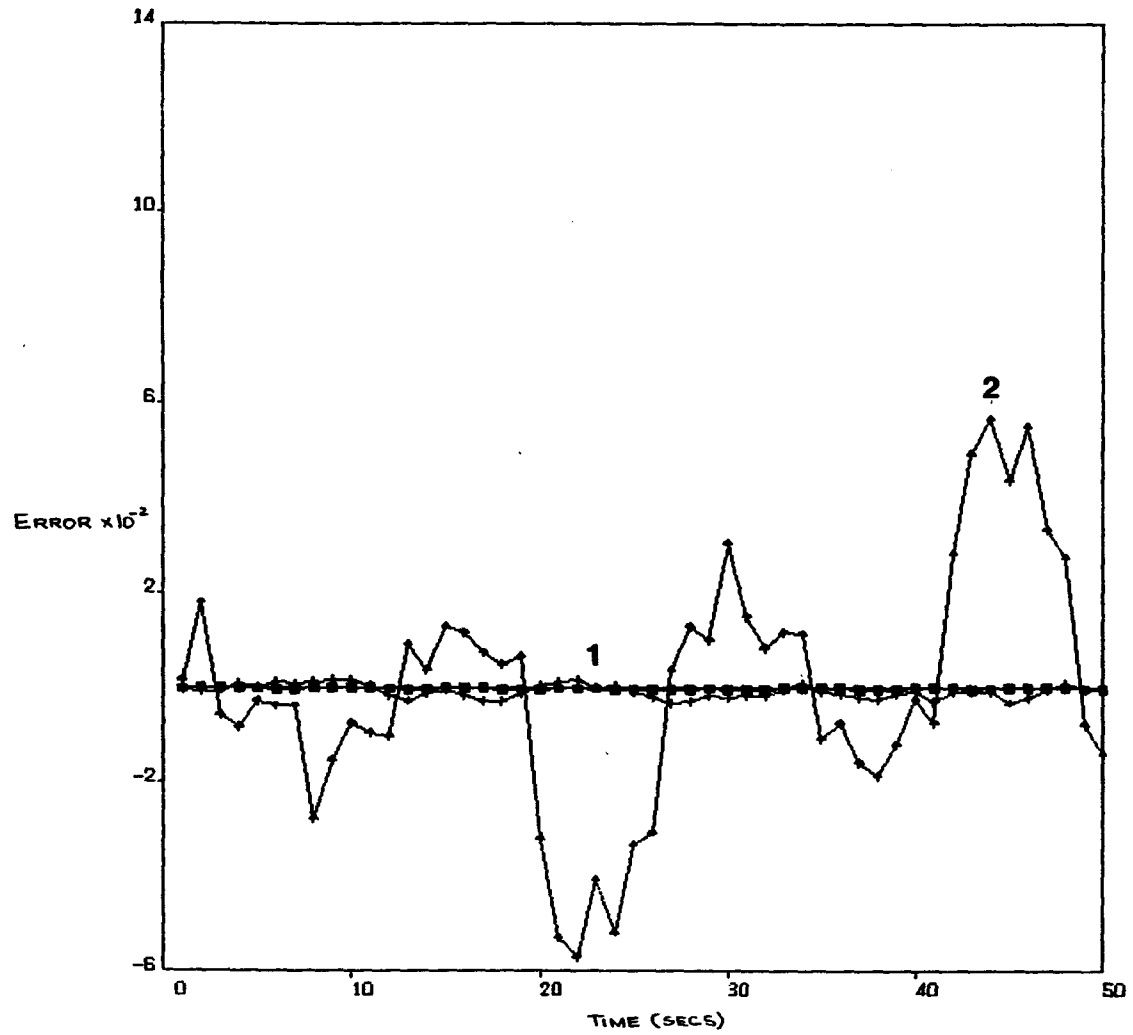


Figure 4.3b.

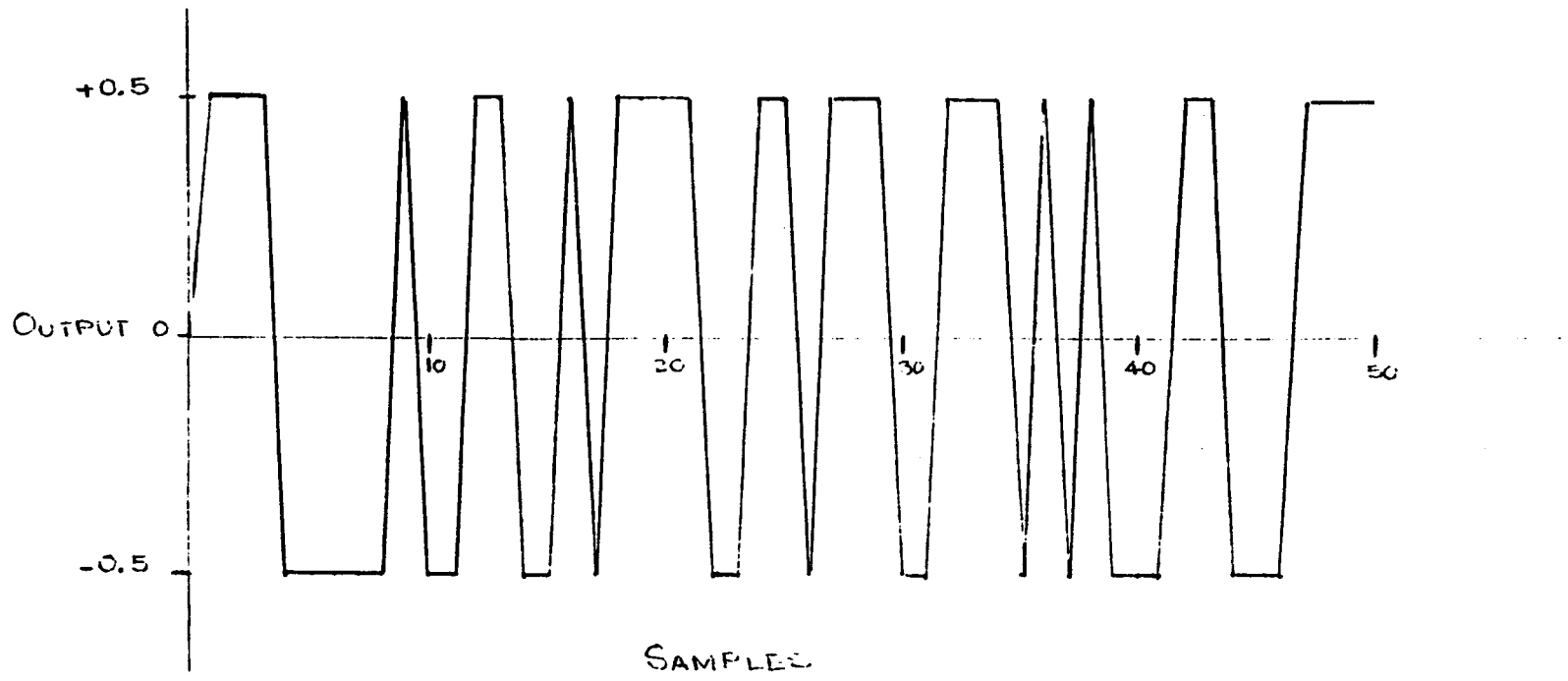


Figure 4.3c.

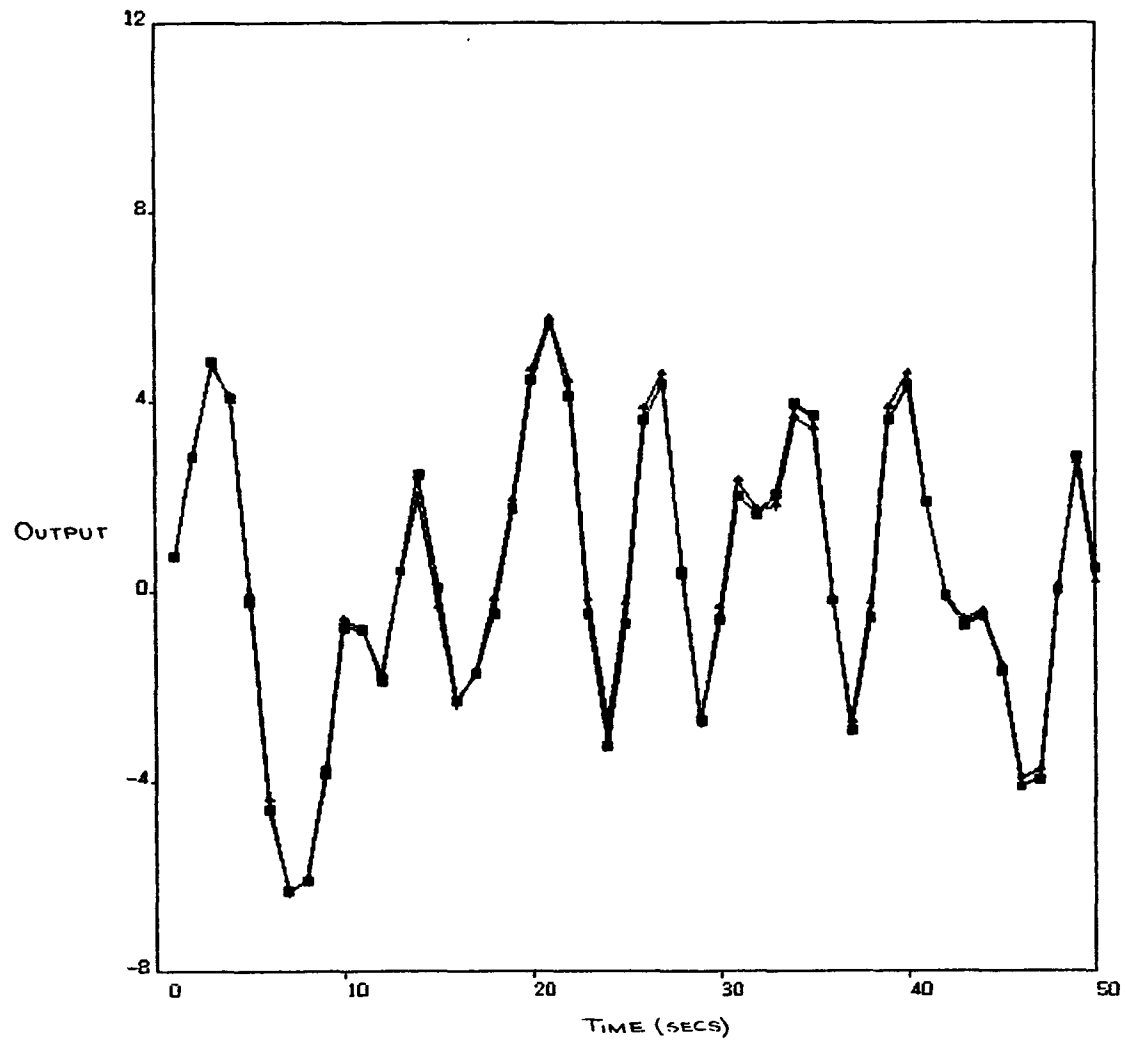


Figure 4.3d.

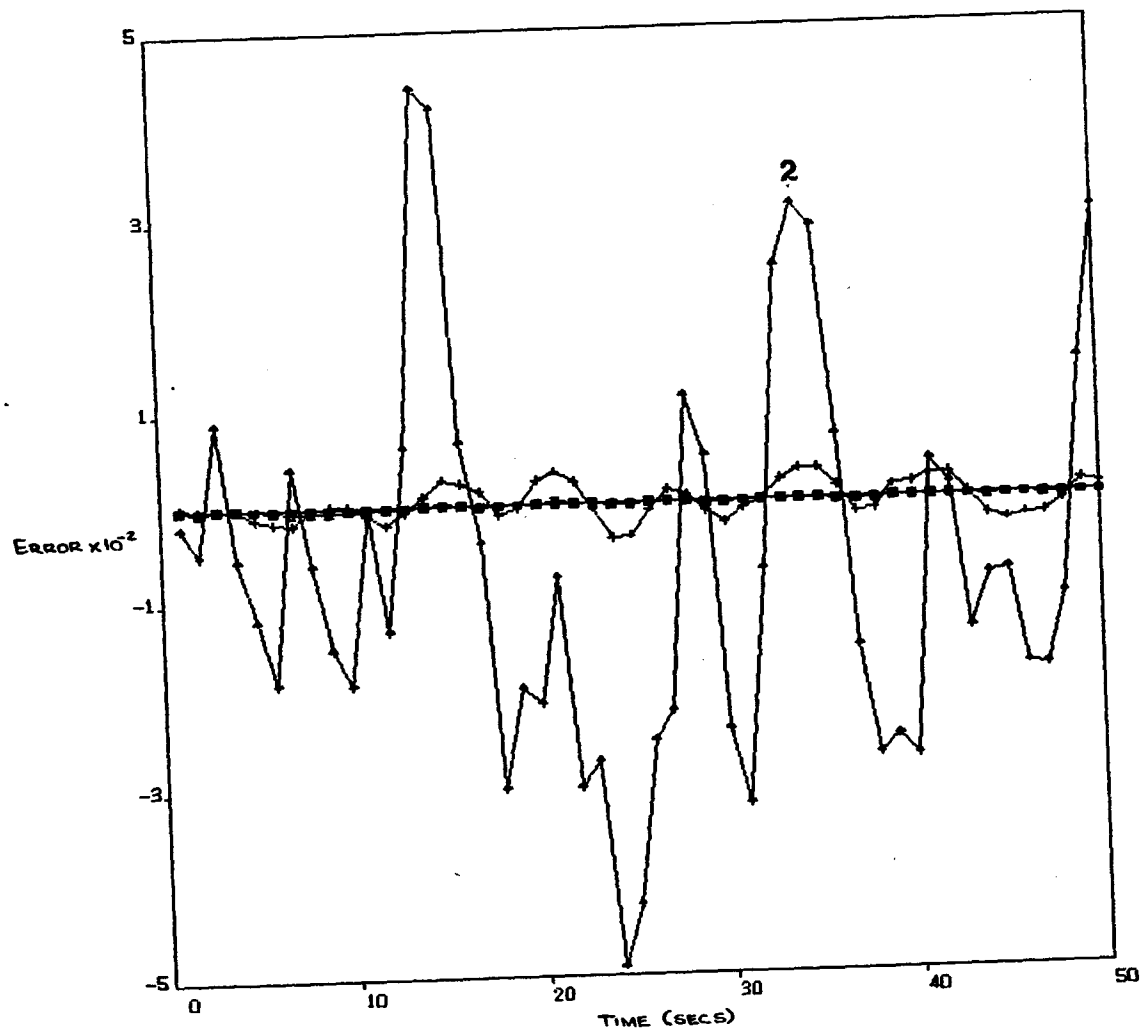


Figure 4.3e.

Figure 4.4. Response and Error function plots for simulations of the digital filter function (4.2); Parallel form is used. -- Each response plot has results of simulations for 32, 24, 16, 12, and 8 bits. Each error function plot is according to (4.3).

- a. Response to a step-input. Curve marked 1 is for 32, 24, 16, and 12 bits. Curve marked 2 is for 8 bits.
- b. Error function versus time for the response plotted in 4.4a above. Dark line (marked 1) is for 32, 24, 16, and 12 bits. Curve marked 2 is for 8 bits.
- c. Response to a sinusoidal input.
- d. Error function versus time for the response plotted in 4.4c above. Curve marked 1 is for 32, 24, 16, and 12 bits. Curve marked 2 is for 8 bits.
- e. Response to a broad-band random input.
- f. Error function versus time for the response plotted in 4.4e above. Curve marked 2 is for 3 bits.

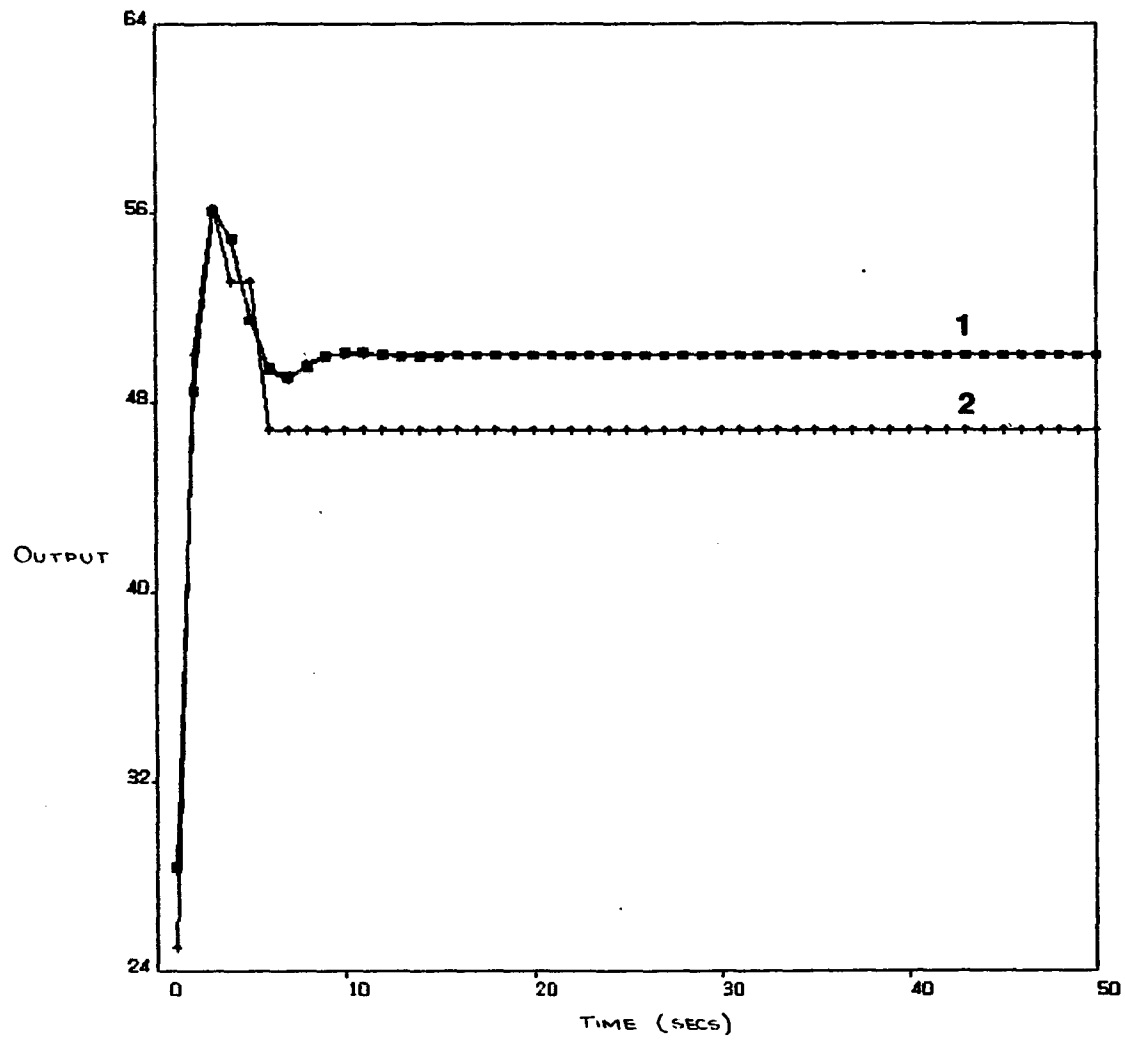


Figure 4.4a.

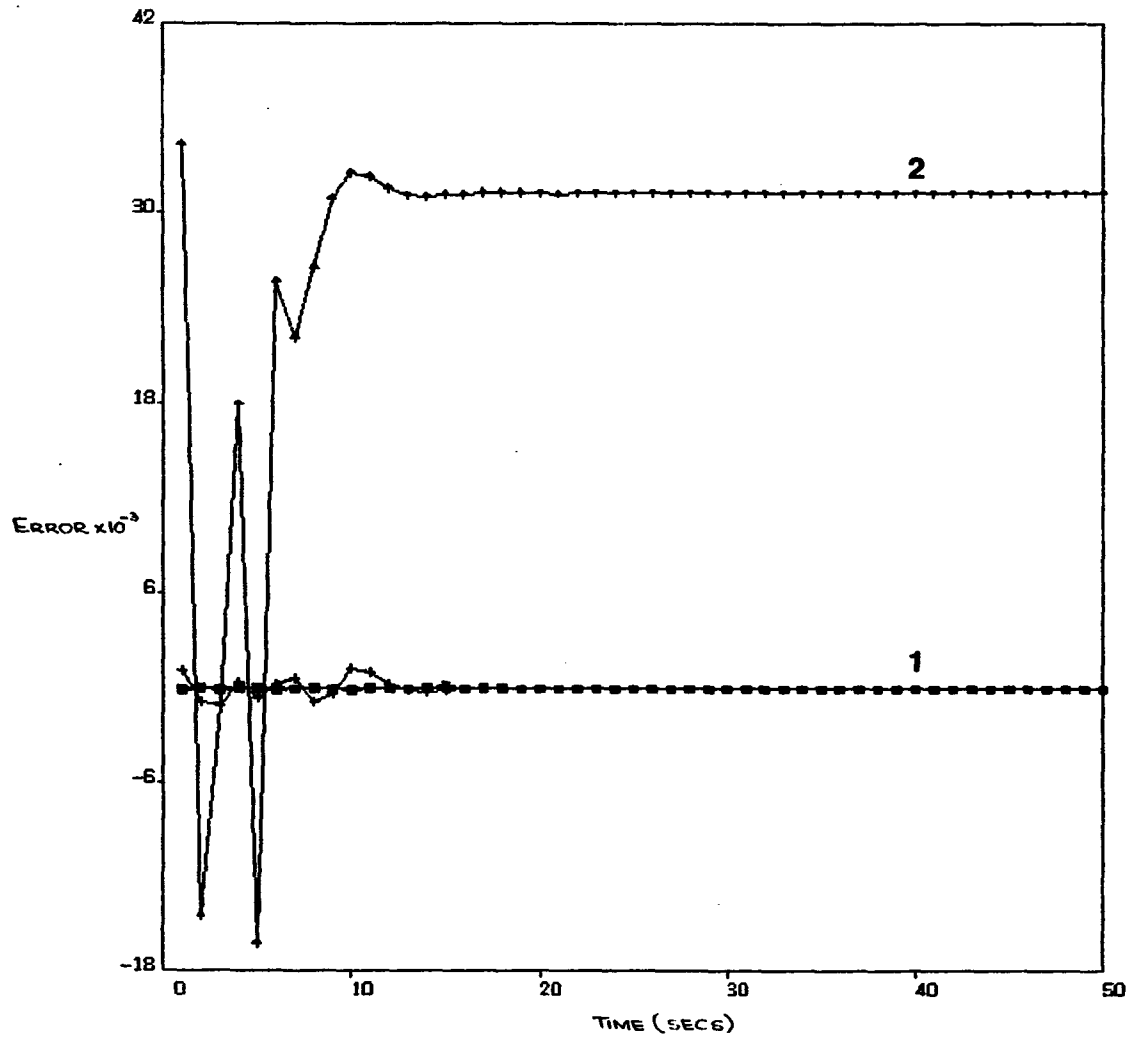


Figure 4.4b.

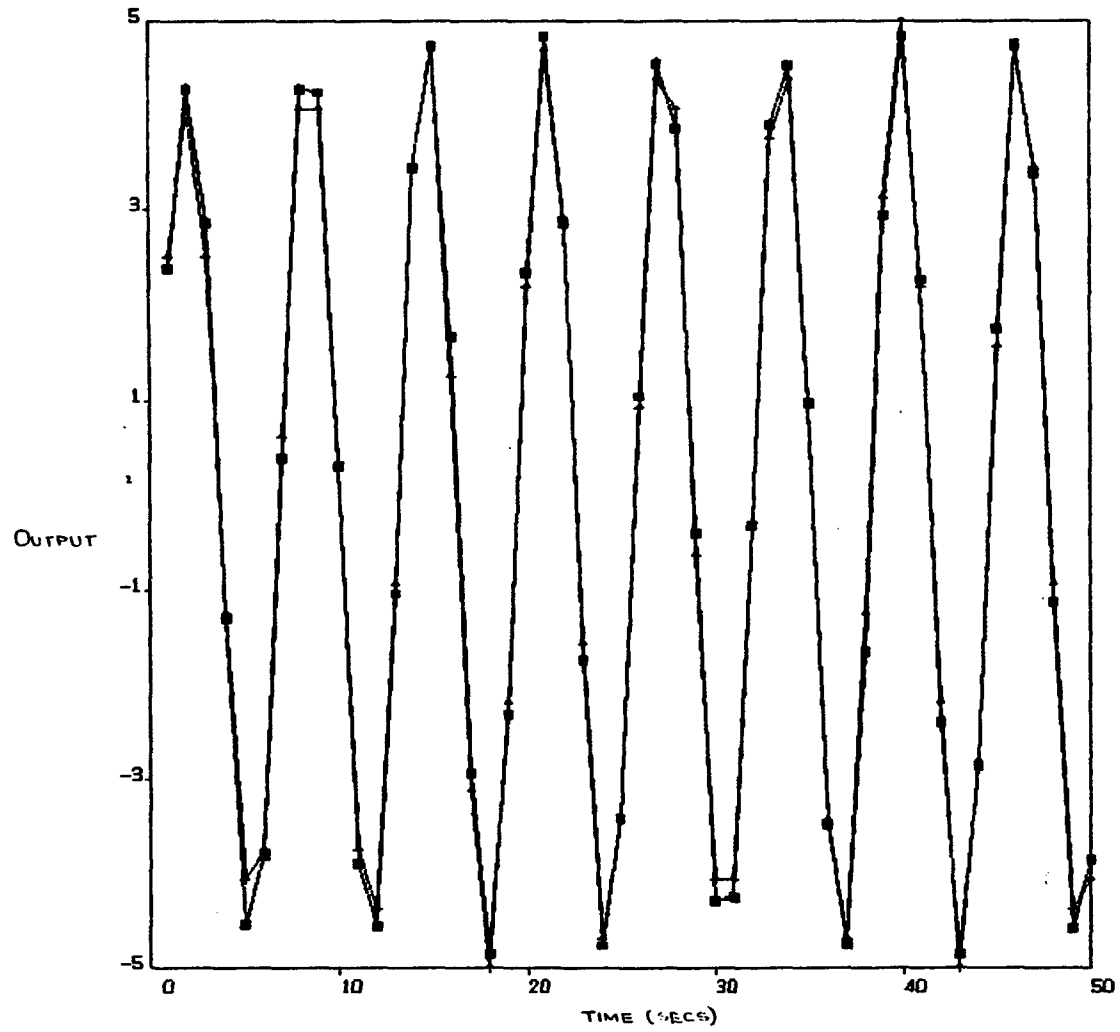


Figure 4.4c.

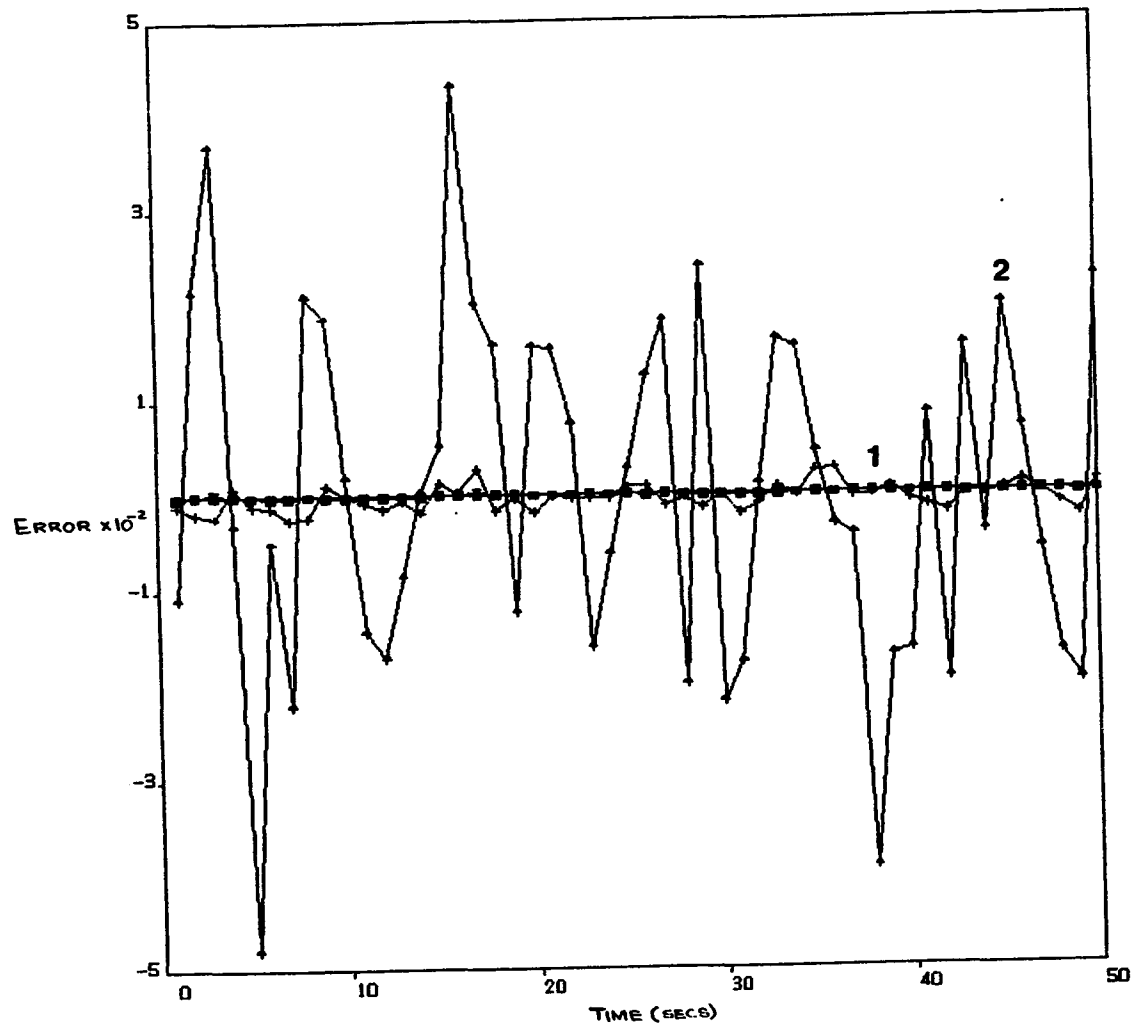


Figure 4.4d.

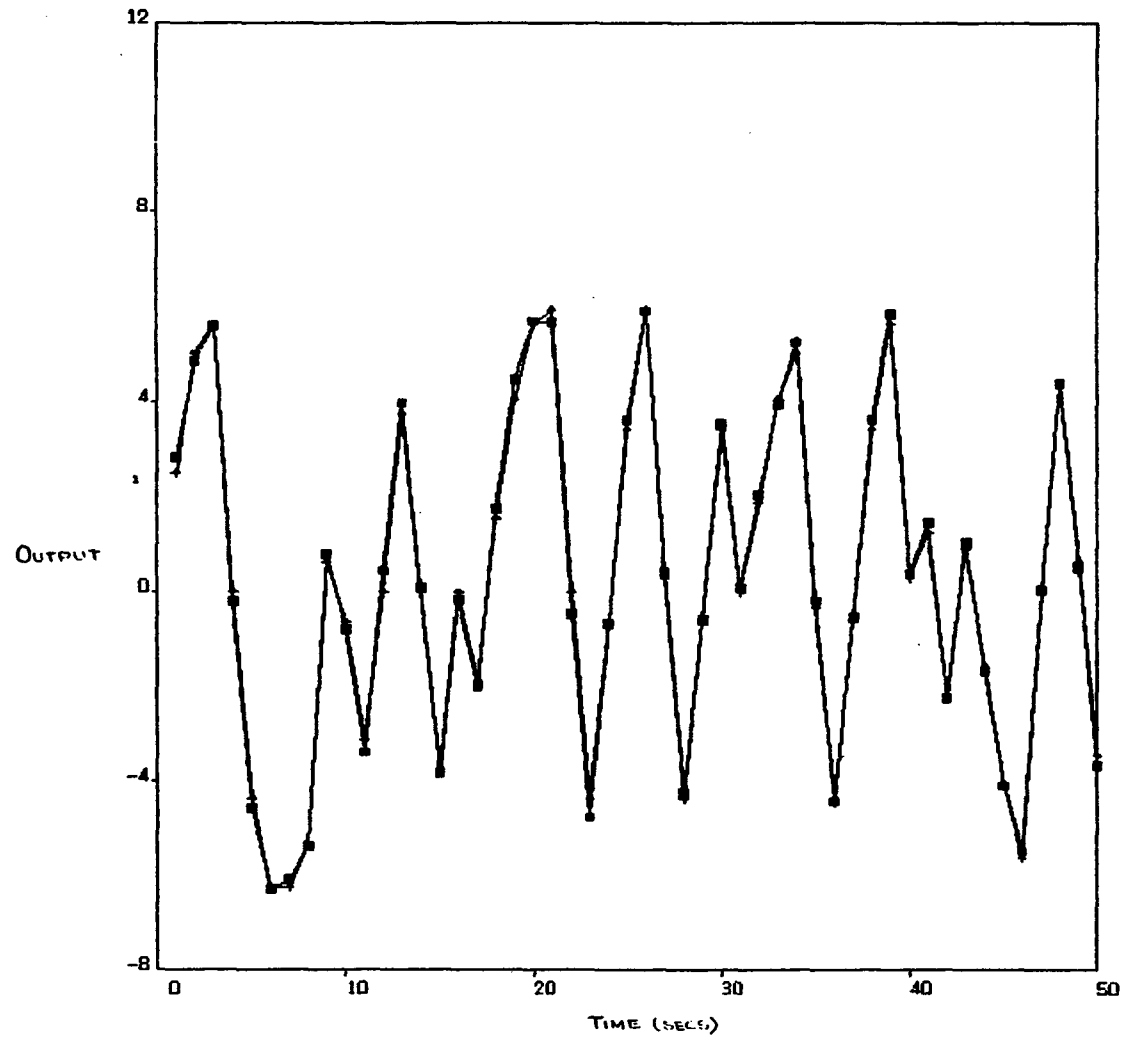


Figure 4.4e.

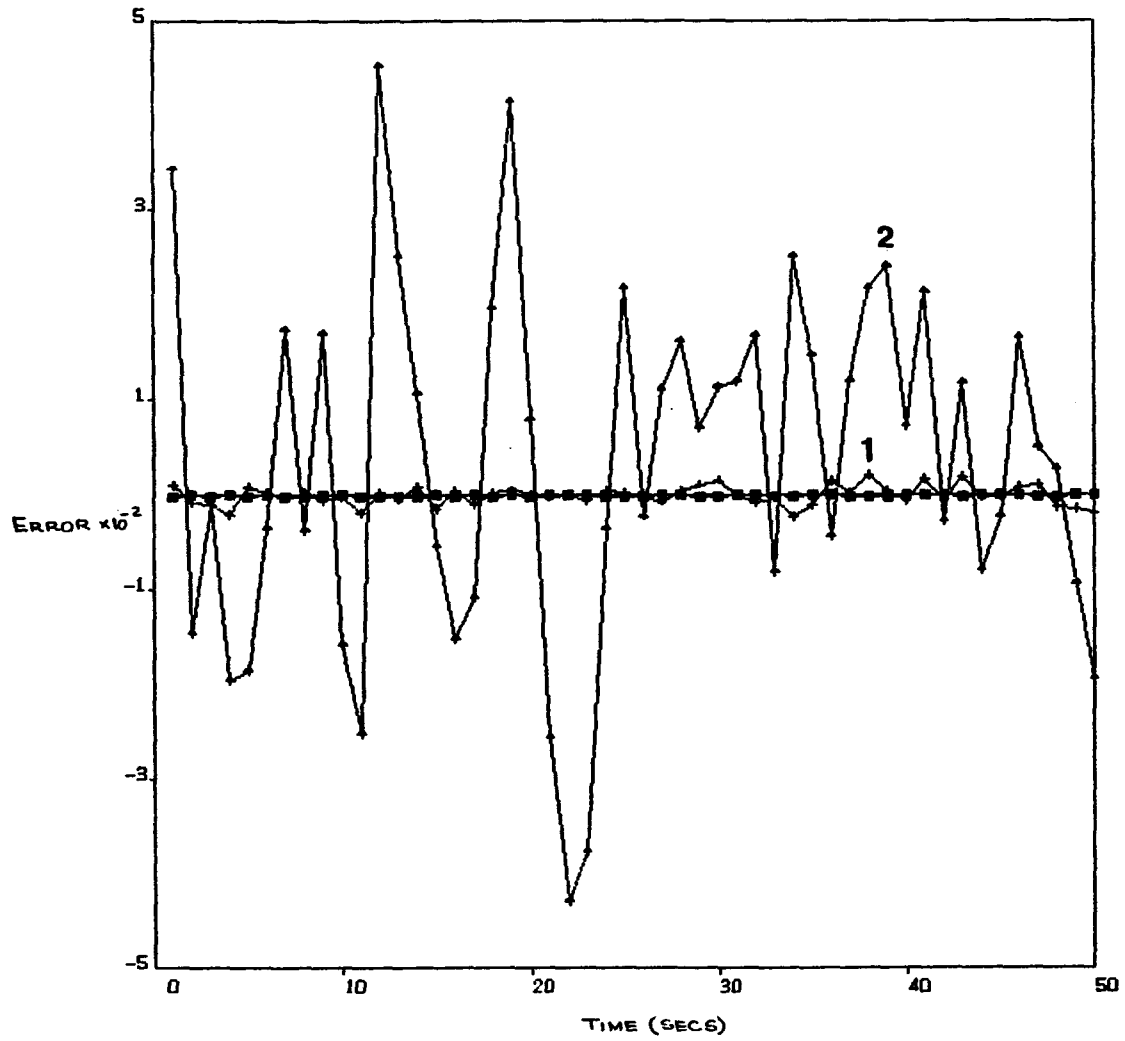


Figure 4.4f.

Figure 4.5. Response and Error function plots for simulations of the digital filter function (4.2); a Normal form is used. -- Each response plot has results of simulations for 32, 24, 16, 12, and 8 bits. Each error function plot is according to (4.3).

- a. Response to a step-input. Curve marked 1 is for 8 bits.
- b. Error function versus time for the response plotted in 4.5a above. Curve marked 1 is for 8 bits.
- c. Response to a sinusoidal input.
- d. Error function versus time for the response plotted in 4.5c above. Curve marked 1 is for the word-lengths up to 12 bits. Curve marked 2 is for 8 bits.
- e. Response to a random input.
- f. Error function versus time for the response plotted in 4.5e above. Curve marked 1 is for the word-lengths up to 12 bits. Curve marked 2 is for 8 bits.

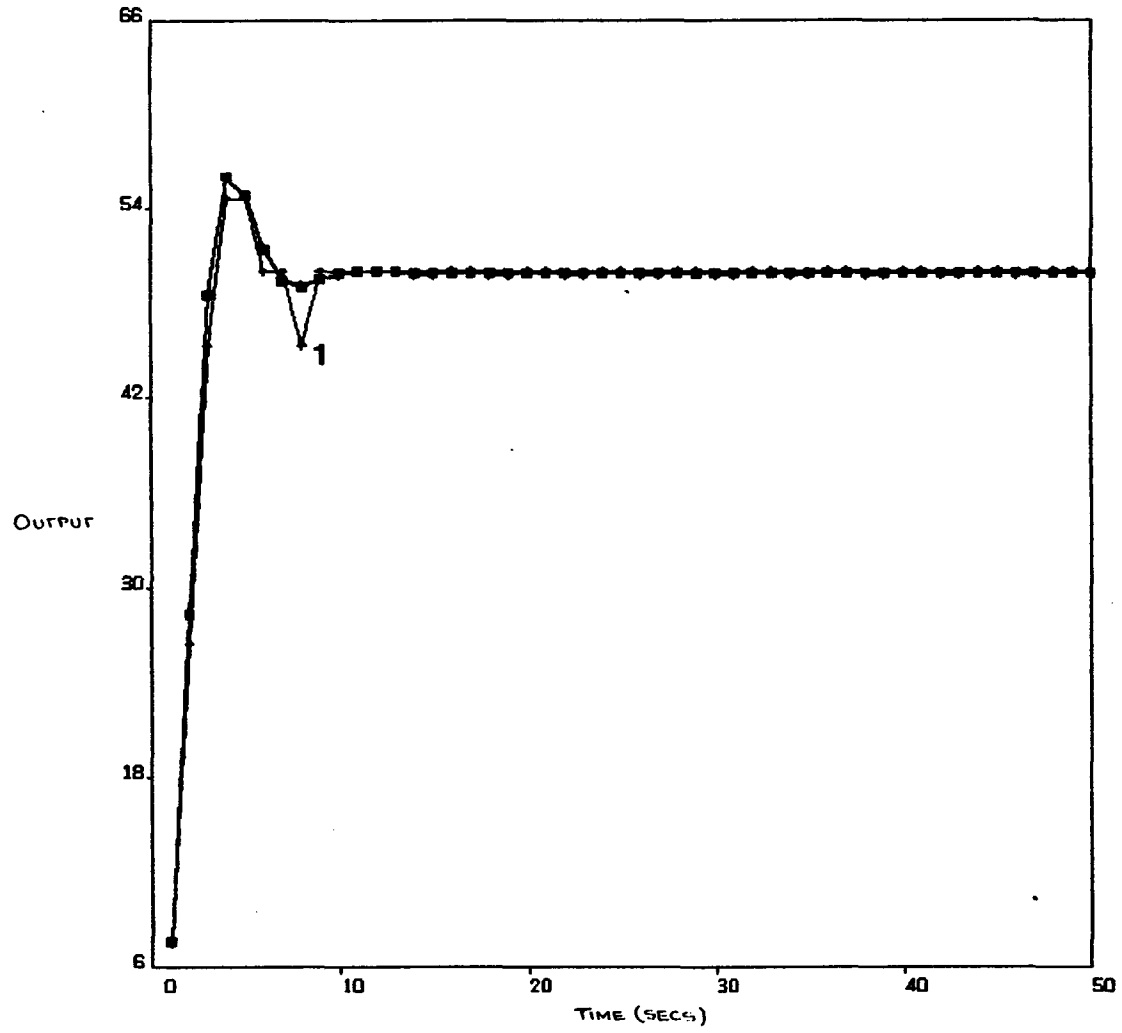


Figure 4.5a.

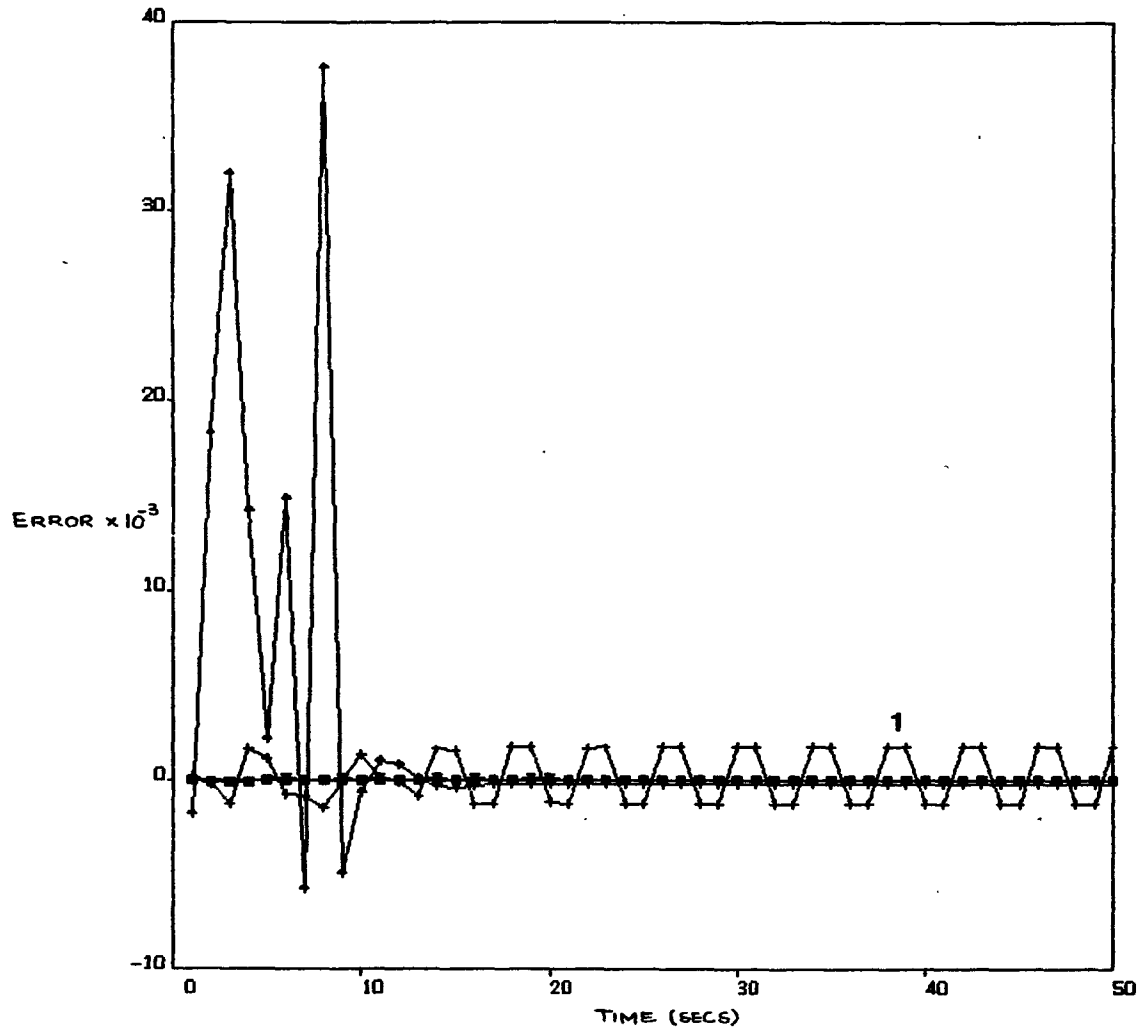


Figure 4.5b.

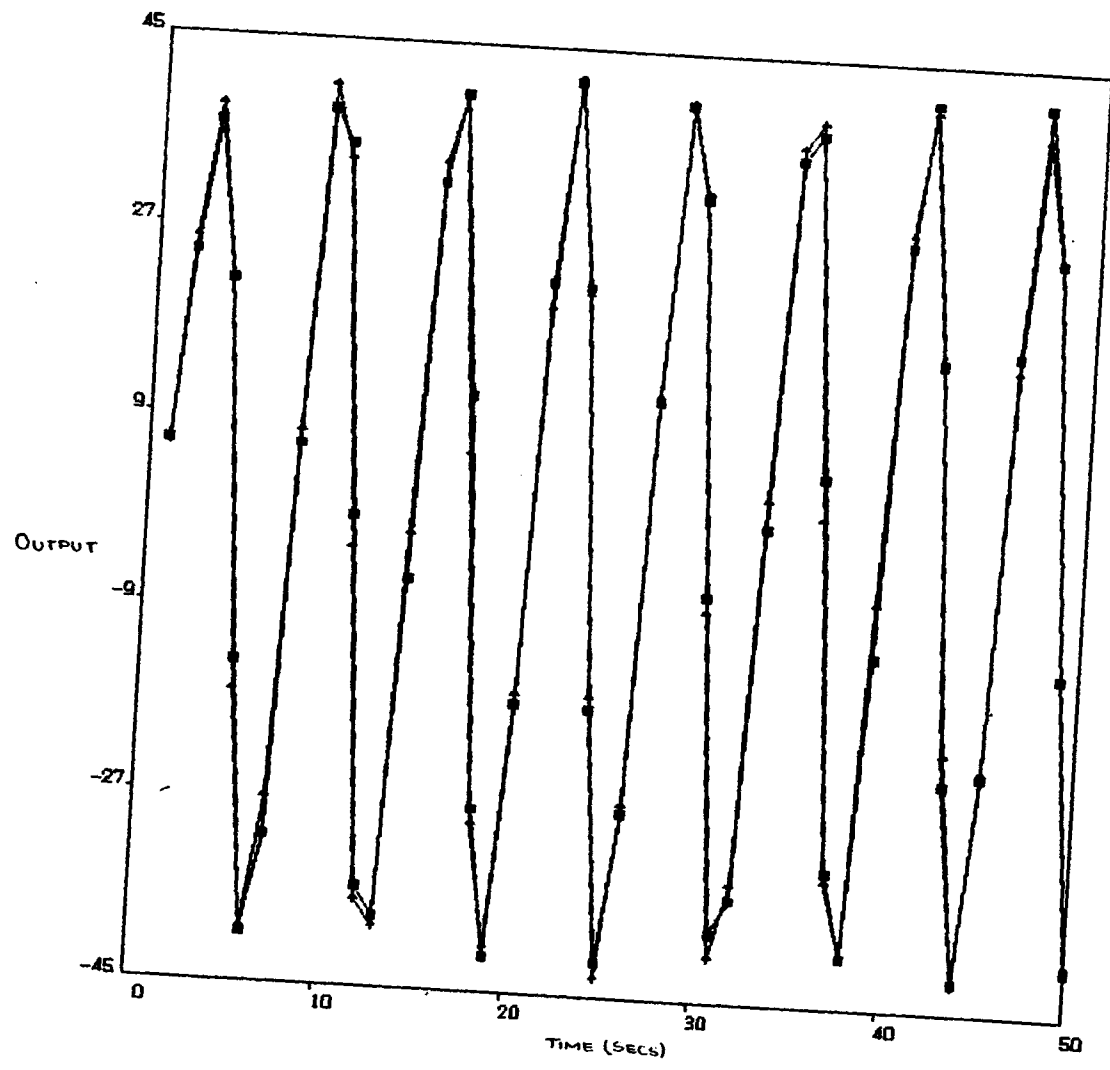


Figure 4.5c.

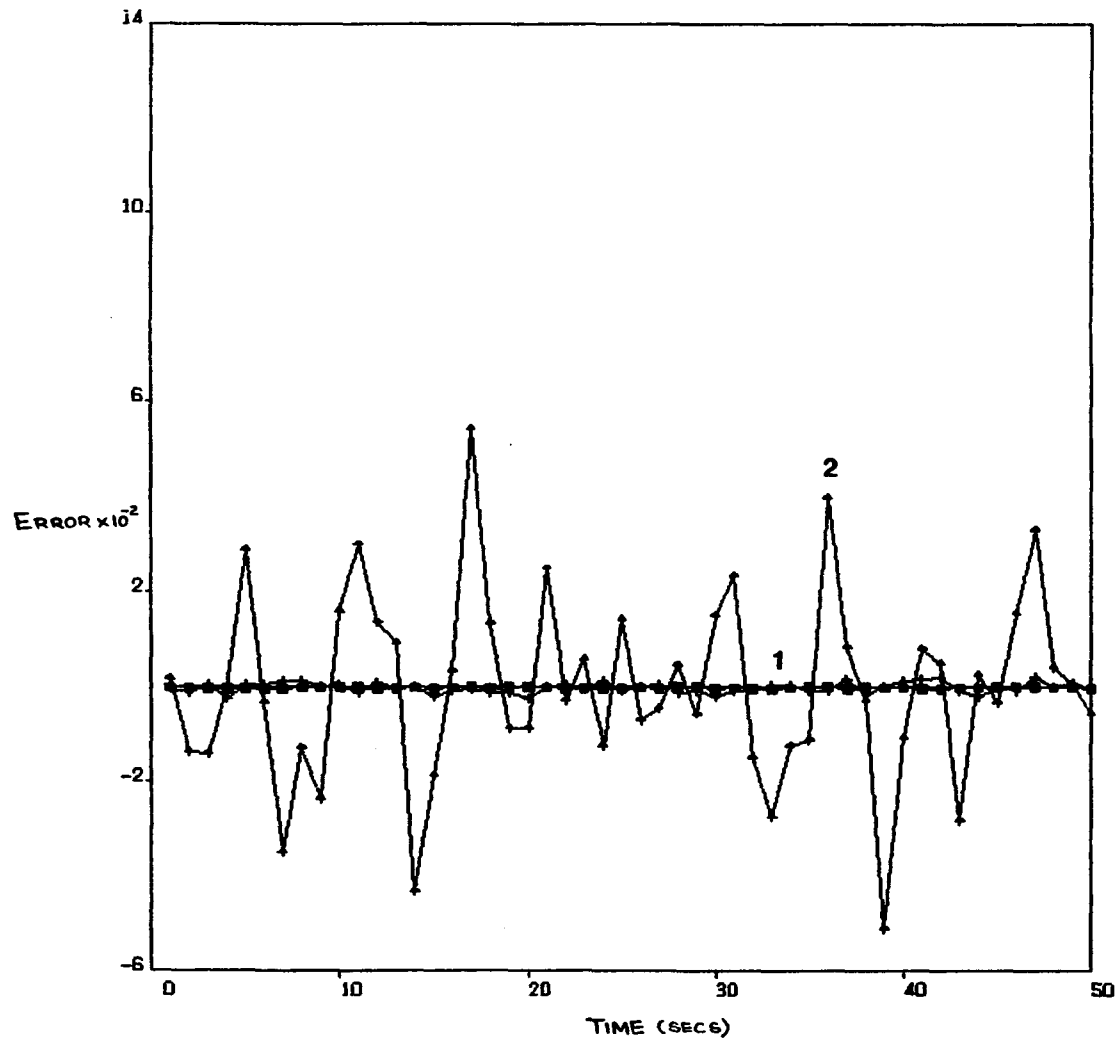


Figure 4.5d.

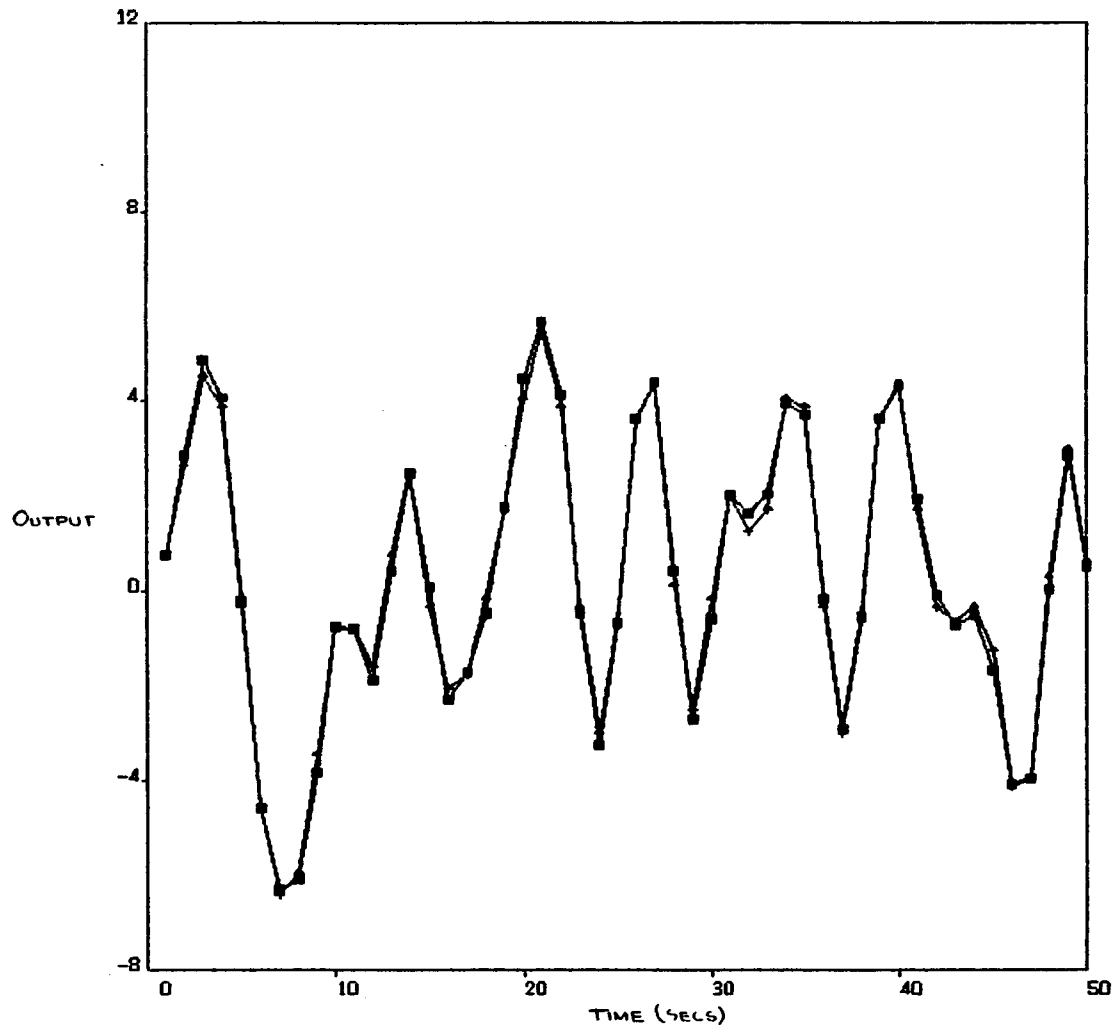


Figure 4.5e.

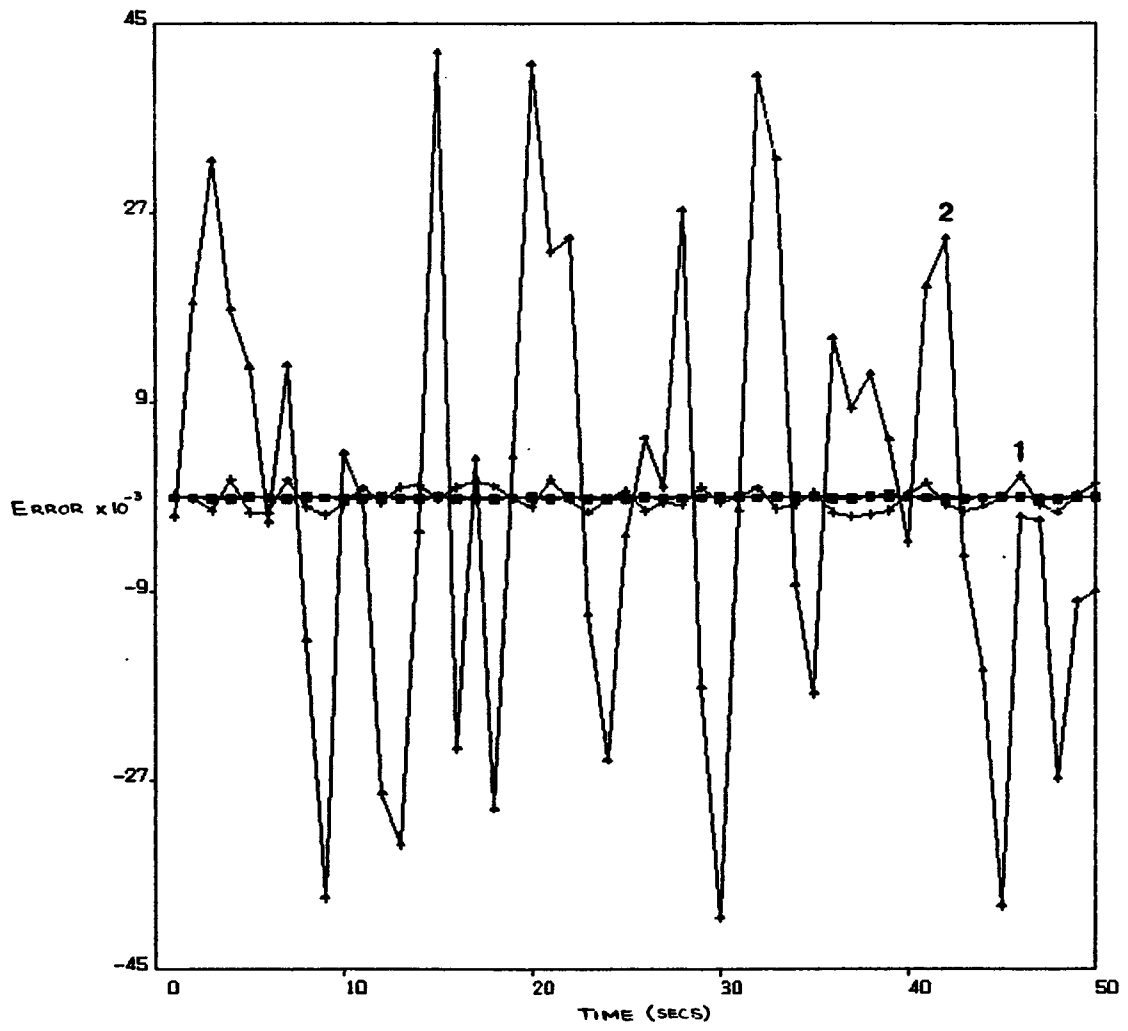


Figure 4.5f.

$$|e(n)_{\max}| = \frac{|y(n)_{32 \text{ bits}} - y(n)_{\text{NBITS}}|}{y(n)_{32 \text{ bits}}} \times 100 \text{ percent}, \quad (4.4)$$

the value of n corresponding to the largest absolute value of the numerator factor. Table 4.1 summarizes the results for the three structures.

Results of the analysis show that for the digital filter function (4.2), the Parallel form gives least error for up to 8 bits, followed by the Normal form, with the Cascade form usually showing maximum sensitivity to the effects of using finite word-length.

Example 2

A sixth-order, low-pass filter will be considered here. This example will be used to demonstrate how a high-order digital filter function can be simulated by using second-order sections connected in cascade (for the Cascade and Normal forms) or in parallel (for the Parallel form). It will also be used to compare the Cascade, Parallel and Normal forms in terms of their sensitivity to finite word-length effects.

A sixth-order, low-pass Butterworth filter has a s -domain transfer function (Oppenheim and Schaffer, 1975, p. 215):

$$H(s) = \frac{0.12903}{(s^2 + 0.3640s + 0.4945)(s^2 + 0.9945s + 0.4945)(s^2 + 1.3585s + 0.4945)} \quad (4.5)$$

Table 4.1. A comparison of the three structures based upon the maximum percentage relative error, $|e(n)_{\max}|$, for the low-pass filter of 4.2. -- Results are based on the effect of changing the word-length (NBITS) as the filter is subjected to three standard inputs.

$$|e(n)_{\max}| = \frac{|y(n)_{32 \text{ bits}} - y(n)_{\text{NBITS}}|}{y(n)_{32 \text{ bits}}} \times 100 \%$$

Structure	Type of Input	$ e(n)_{\max} $			
		Number of Bits			
		24	16	12	8
Cascade	Step	0.002	0.042	0.99	6.7
	Sine	0.0002	0.067	1.33	13.5
	Random	0.0017	0.064	1.10	15.2
Parallel	Step	0.0	0.028	0.26	12.1
	Sine	0.0002	0.076	0.95	10.5
	Random	0.0002	0.0004	0.61	10.0
Normal	Step	0.0	0.03	0.36	7.6
	Sine	0.0002	0.05	1.35	45
	Random	0.0	0.053	0.537	12.3

Using the Matched-z transformation (Antoniou, 1979, p. 175), with a sampling time, $T = 1$ sec., the corresponding z-domain transfer is:

$$H(z) = \frac{HO(z^6 + a_1z^5 + a_2z^4 + a_3z^3 + a_4z^2 + a_5z + a_6)}{(z^6 + b_1z^5 + b_2z^4 + b_3z^3 + b_4z^2 + b_5z + b_6)} \quad (4.6)$$

where

$$HO = 0.000485498$$

$a_1 = 6$	$b_1 = -3.3636809$
$a_2 = 15$	$b_2 = 5.0688387$
$a_3 = 20$	$b_3 = -4.27633882$
$a_4 = 15$	$b_4 = 2.10690555$
$a_5 = 6$	$b_5 = -0.57073848$
$a_6 = 1$	$b_6 = 0.06608585$

Figures 4.6c, 4.6f and 4.6i, respectively, show second-order sections of the Cascade, Parallel, and Normal forms connected to implement (4.6).

In this example only the system response to a step-input will be investigated. Figures 4.6a through 4.6f show the simulation results for the Cascade, Parallel, and Normal networks and a 0.5 step-input.

Figures 4.6c and 4.6d show a listing and a plot of the response. For NBITS = 12 and 8, a zero or "null" response is obtained. Using an analysis similar to that of

Figure 4.6. The sixth-order, low-pass filter function (4.6) is simulated using the Cascade, Parallel, and Normal forms and driven by a step input $u(n) = 0.5$. -- Response plots shown in the following pages and described below are for simulations using 32, 24, 16, 12, and 8 bits. Error function plots according to (4.3) are also included.

- a. Simulation parameters.
- b. Feed-forward and feed-back coefficients for a tandem connection of the Cascade second-order sections.
- c. Cascade network.
- d. Response plot for the Cascade network. Curves marked 1 are for 32, 24, and 16 bits. "Null" responses, curves marked 2, are obtained for 8 and 12 bits.
- e. Error function versus time for the response plotted in 4.6d above. Curves marked 1 are for up to 16 bits. Curves marked 2 are for 12 and 8 bits.
- f. Parallel network.
- g. Response plot for the Parallel network. Curves marked 1 are for 32, 24, 16, and 12 bits. Curve marked 2 is for 8 bits.
- h. Error function versus time for the response plotted in 4.6g above. Curve marked 1 is for 8 bits.
- i. Normal network.
- j. Response plot for the Normal network. Curves marked 1 are for 32, 24, 16, and 12 bits. Curve marked 2 is for 8 bits.
- k. Error function versus time for the response plotted in 4.6j above. Curve marked 1 is for a word-length of up to 12 bits. Curve marked 2 is for 8 bits.

SIMULATION PARAMETERS

TMAX= 50.0 M= 6 N= 6 NBITS=32 ISHIFT= 0
U=1.0 ARG1= .5 ARG2= 1.0 H0= .000485498 GRAF=1.0

COEFFICIENTS OF NUMERATOR POLYNOMIAL

F(1)= 1.00000000
F(2)= 6.00000000
F(3)= 15.00000000
F(4)= 20.00000000
F(5)= 15.00000000
F(6)= 6.00000000
F(7)= 1.00000000

COEFFICIENTS OF DENOMINATOR POLYNOMIAL

E(1)= .06608585
E(2)= -.57073848
E(3)= 2.10690555
E(4)= -4.27633882
E(5)= 5.06883873
E(6)= -3.36368090
E(7)= 1.00000000

NUMBER OF SECOND ORDER SECTIONS= 3

STRUCTURE TYPE=DIRECT

NUMBER OF BITS (NBITS) FOR 2ND. RUN= 24

NUMBER OF BITS (NBITS) FOR 3RD. RUN= 16

NUMBER OF BITS (NBITS) FOR 4TH. RUN= 12

NUMBER OF BITS (NBITS) FOR 5TH. RUN= 8

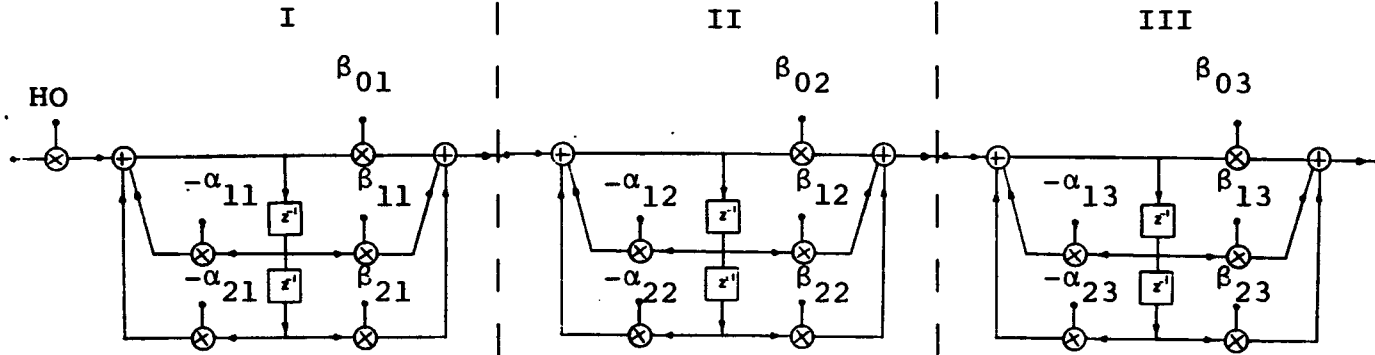
Figure 4.6a.

FEED-FORWARD AND FEED-BACK COEFFICIENTS

A0	A1	A2	B1	B2
1.0000000	1.9946698	.9946980	-.9972958	.2570716
1.0000000	2.0053086	1.0053370	-1.0691760	.3699456
1.0000000	2.0000216	.9999933	-1.2972092	.6948908

Figure 4.6b.

CASCADE NETWORK



HO = 0.000485498

I	II	III
$\beta_{01} = 1.0$	$\beta_{02} = 1.0$	$\beta_{03} = 1.0$
$\beta_{11} = 2.0$	$\beta_{12} = 2.0$	$\beta_{13} = 2.0$
$\beta_{21} = 1.0$	$\beta_{22} = 1.0$	$\beta_{23} = 1.0$
$\alpha_{11} = -0.9973$	$\alpha_{12} = -1.0691$	$\alpha_{13} = -1.2972$
$\alpha_{21} = 0.2571$	$\alpha_{22} = 0.3699$	$\alpha_{23} = 0.6948$

Figure 4.6c.

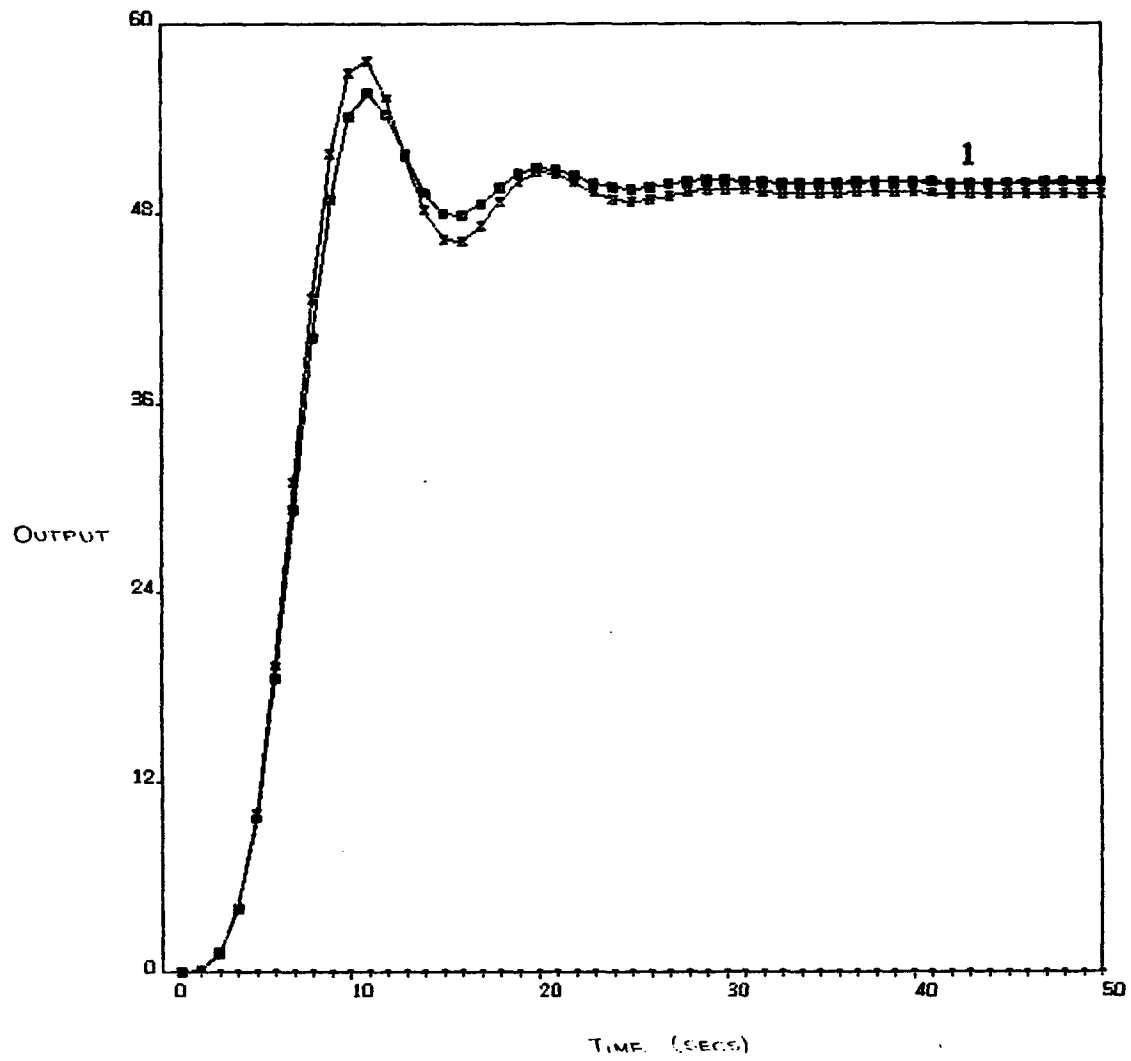


Figure 4.6d.

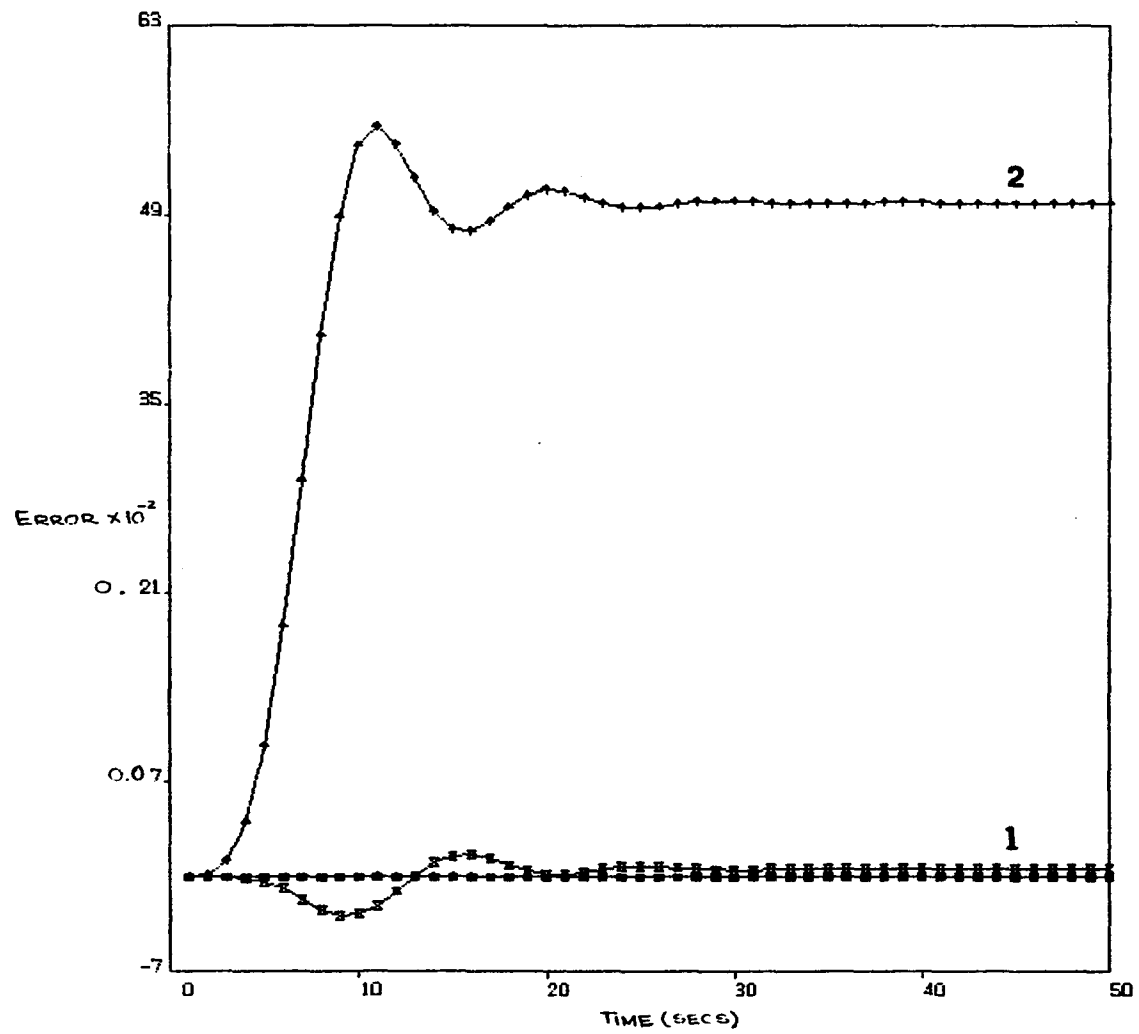
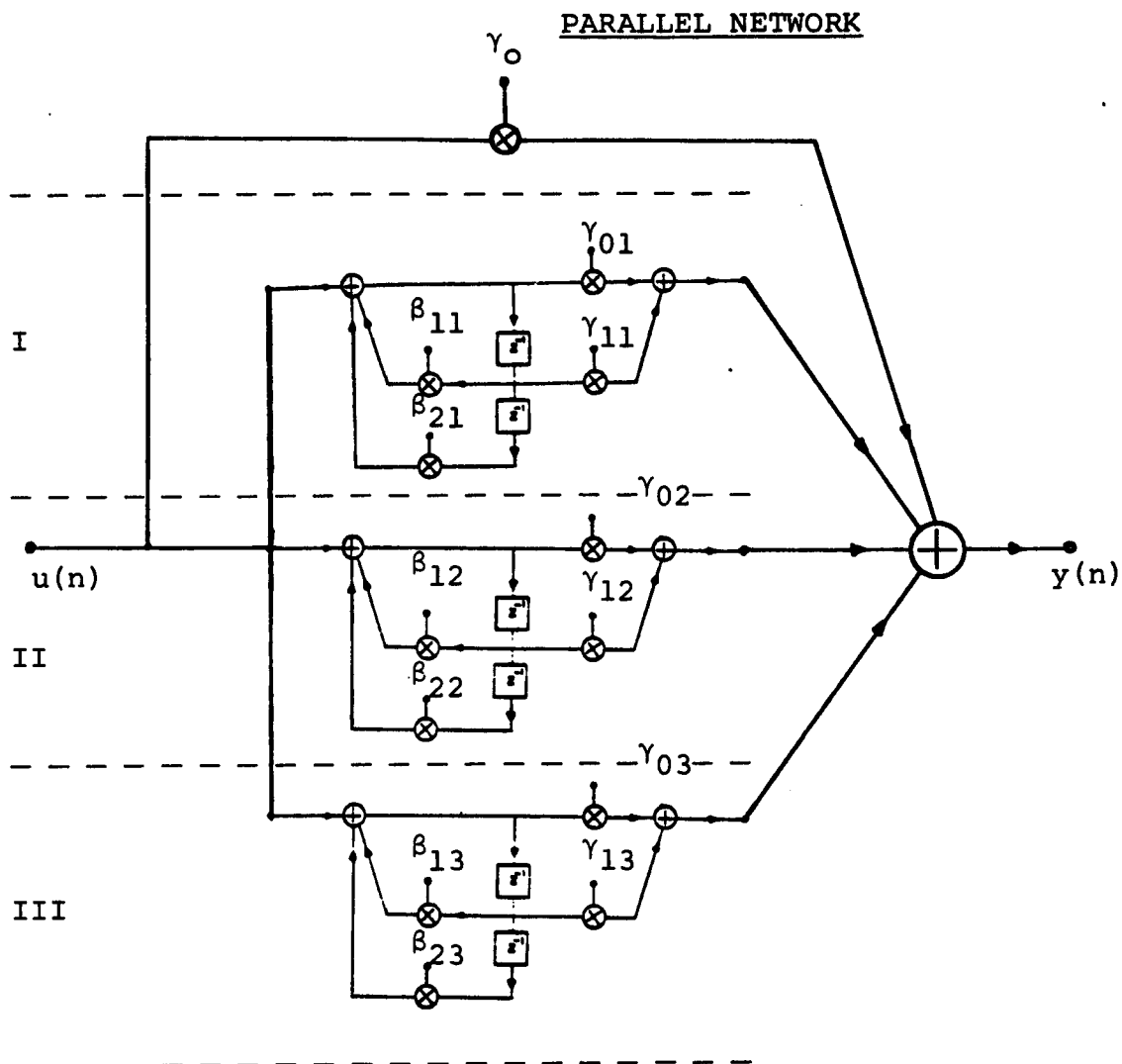


Figure 4.6e.



$$\gamma_0 = 0.000485498$$

I	II	III
$\gamma_{01} = 1.4095$	$\gamma_{02} = -1.4568$	$\gamma_{03} = 0.0519$
$\gamma_{11} = -0.4907$	$\gamma_{12} = 0.8376$	$\gamma_{13} = -0.2422$
$\beta_{11} = -0.9973$	$\beta_{12} = -1.0692$	$\beta_{13} = -1.2972$
$\beta_{21} = 0.2571$	$\beta_{22} = 0.3699$	$\beta_{23} = 0.6949$

Figure 4.6f.

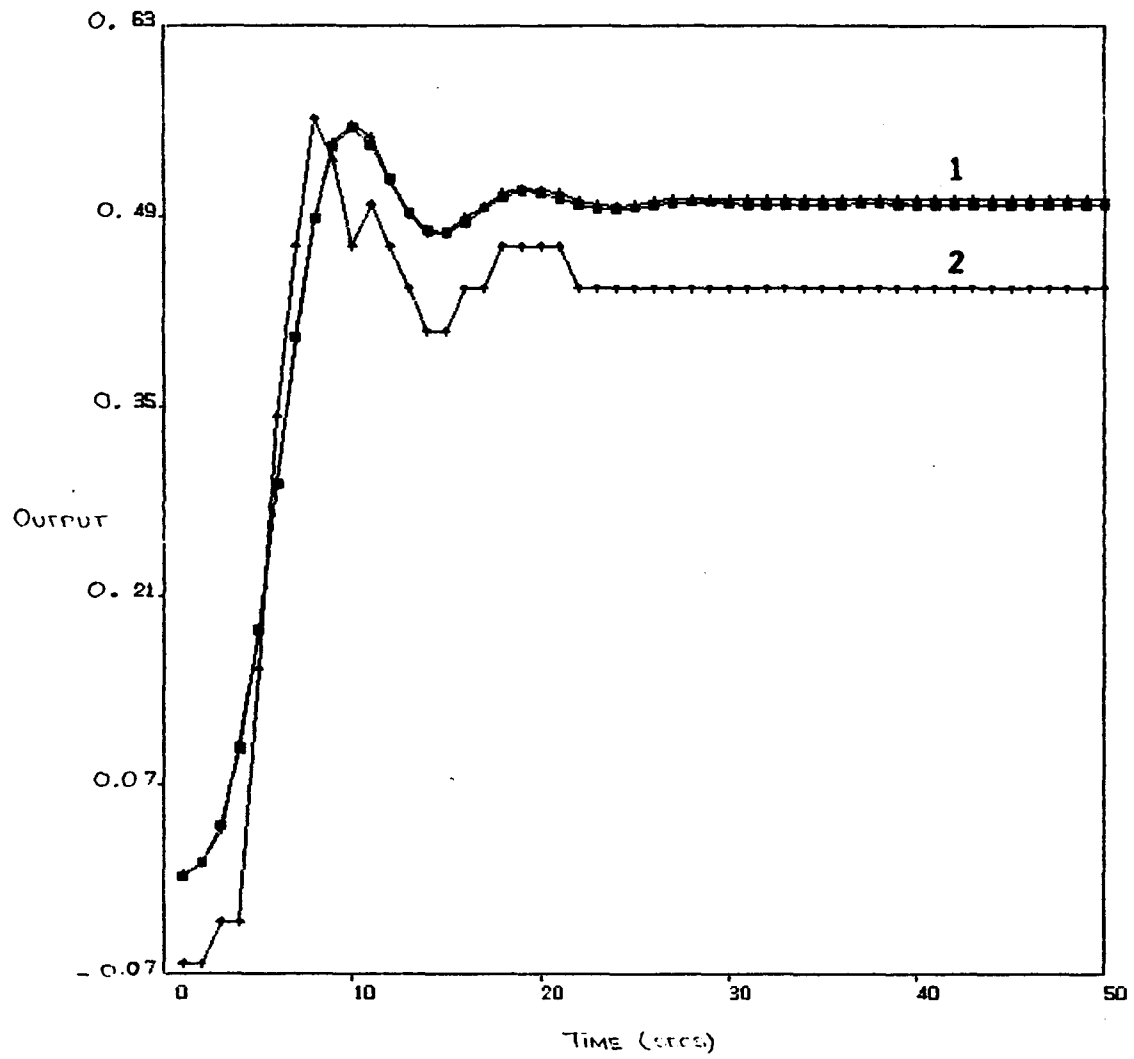


Figure 4.6g.

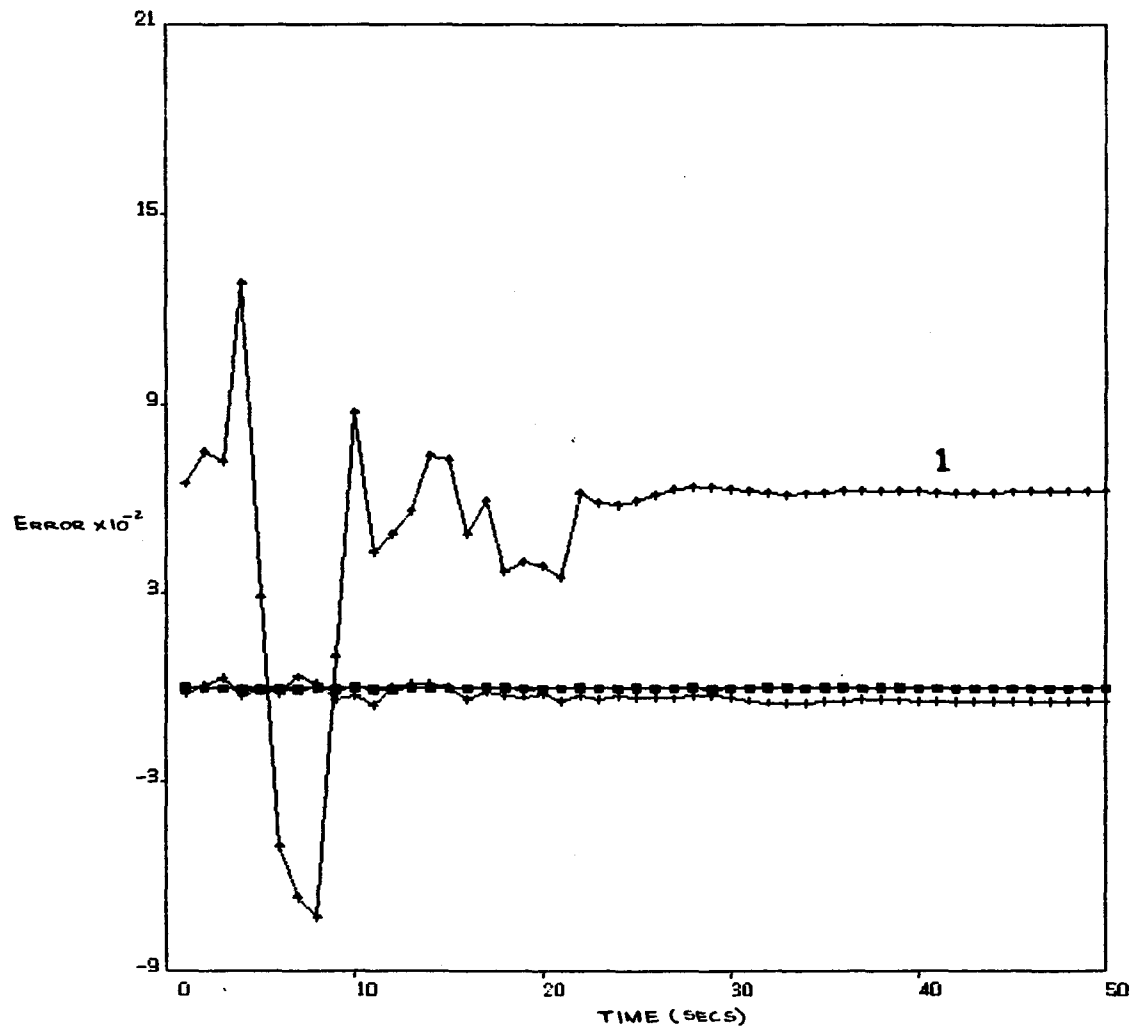
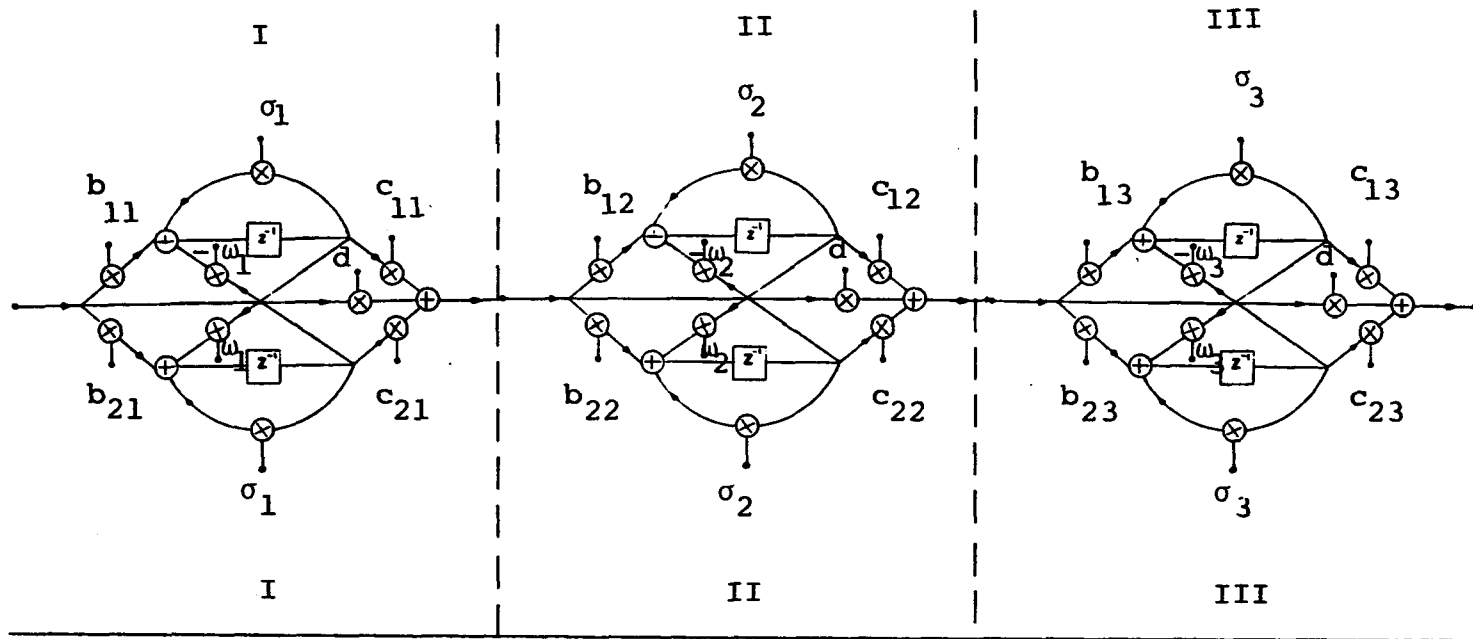


Figure 4.6h.

NORMAL NETWORK



$$\begin{aligned}
 \sigma_1 &= 0.4986 \\
 \omega_1 &= 0.09177 \\
 b_{11} &= 0.18245 \\
 b_{21} &= 0.16169 \\
 c_{11} &= -3.7189 \\
 c_{21} &= 5.78111 \\
 d &= 0.078595
 \end{aligned}$$

$$\begin{aligned}
 \sigma_2 &= 0.53458 \\
 \omega_2 &= 0.29011 \\
 b_{12} &= 0.18532 \\
 b_{22} &= 0.12673 \\
 c_{12} &= -0.6675 \\
 c_{22} &= 2.8767 \\
 d &= 0.078595
 \end{aligned}$$

$$\begin{aligned}
 \sigma_3 &= 0.64860 \\
 \omega_3 &= 0.52364 \\
 b_{13} &= 0.140749 \\
 b_{23} &= 0.06781 \\
 c_{13} &= 0.23544 \\
 c_{23} &= 2.97155 \\
 d &= 0.078595
 \end{aligned}$$

Figure 4.6i.

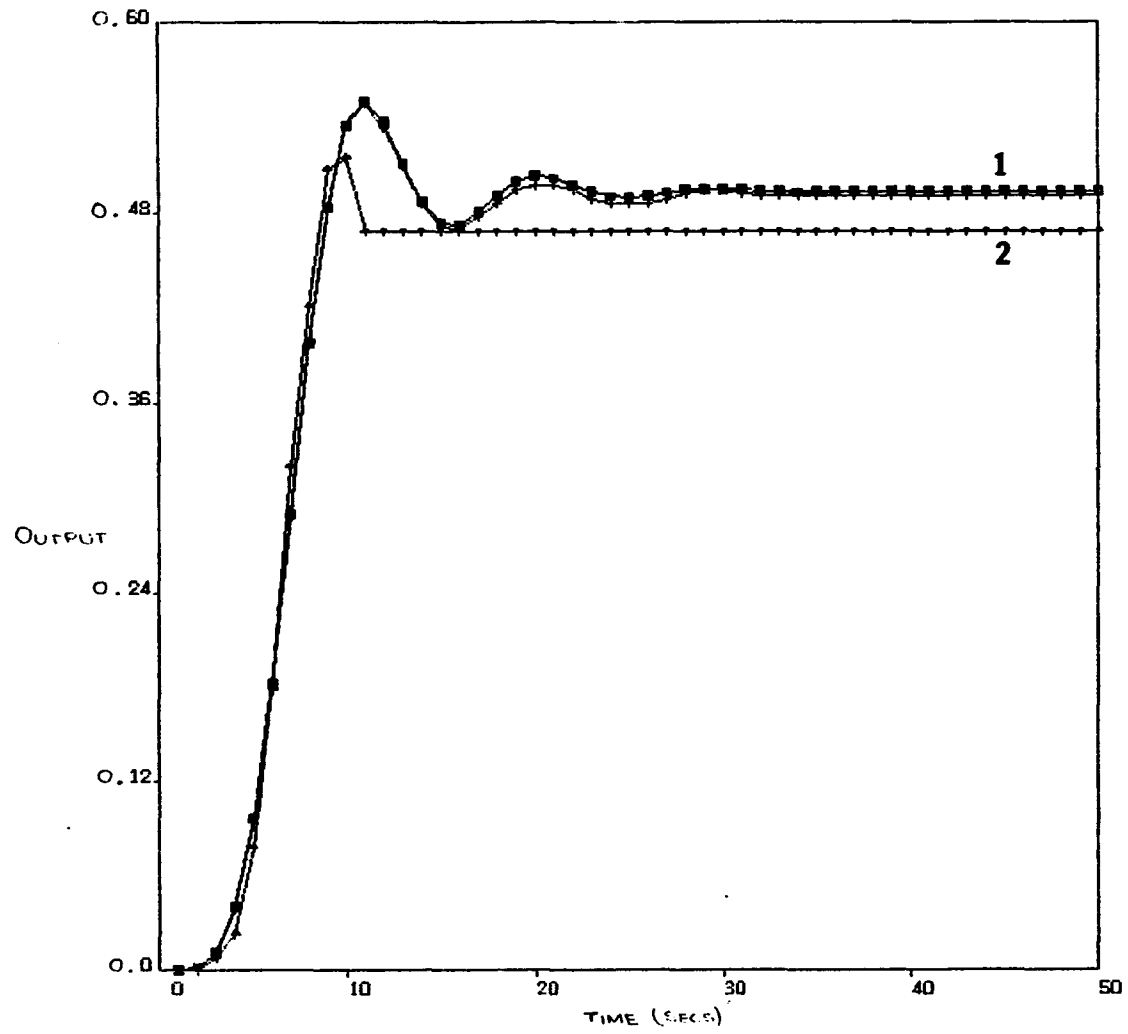


Figure 4.6j.

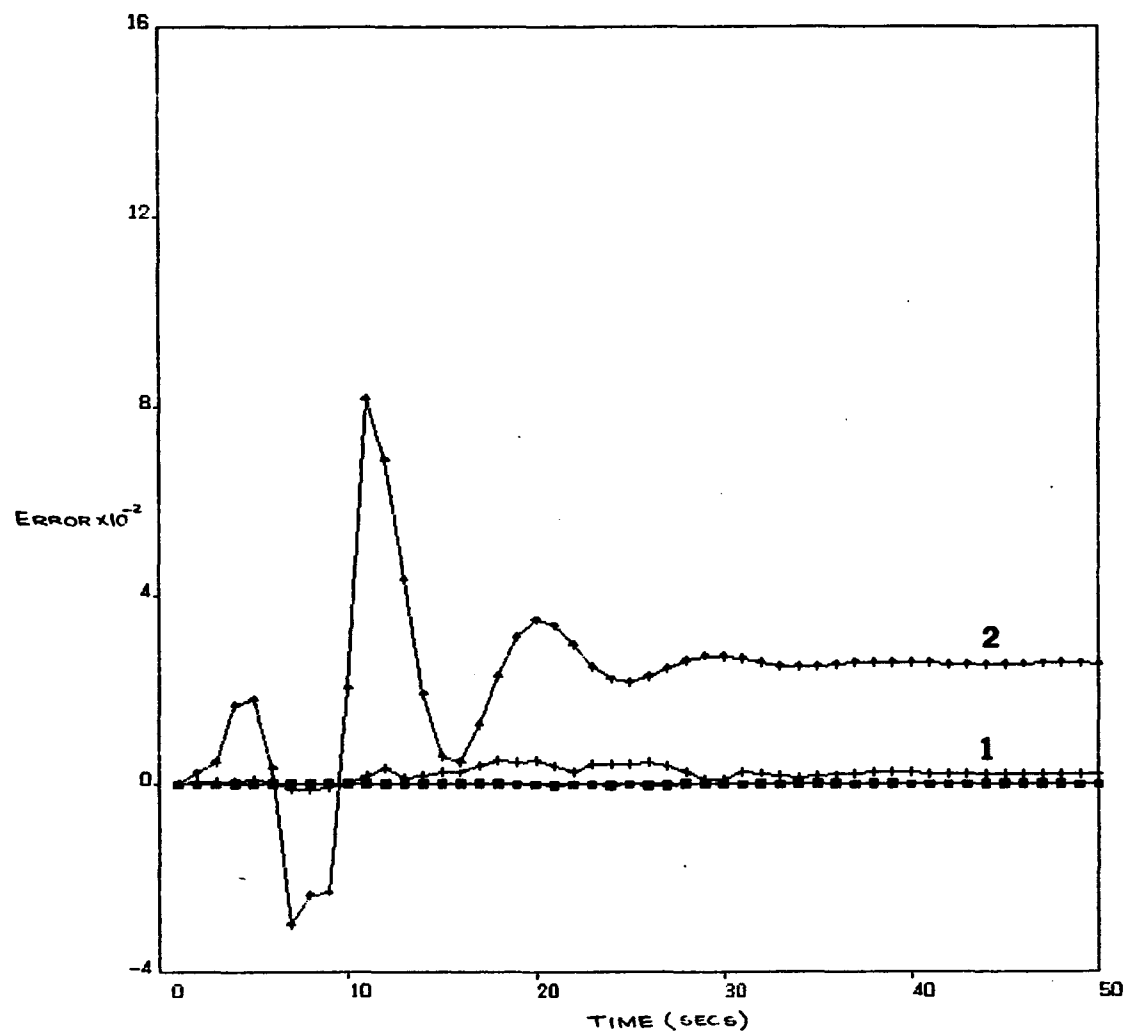


Figure 4.6k.

Example 1, these results will be shown to be a consequence of using a short word-length in forming the system input. Referring once again to Figure 4.2, the quantized values of HO for 12 and 8 bits are:

$$Q[HO]_{12 \text{ bits}} = 0.0$$

$$Q[HO]_{8 \text{ bits}} = 0.0$$

As a result, the product P_x , which is the input to the system for these two word-lengths, is zero and therefore a zero system response is obtained.

For 16 bits, the quantized value of HO is:

$$Q[HO]_{16 \text{ bits}} = 0.000473023$$

which has been rounded down from the high-precision value of 0.000485498. As a result, the steady-state response shows an error of 1.3%.

For the Parallel and Normal forms, the effects of finite word-length were less severe. No zero or "null" responses were obtained. Table 4.2 summarizes the results in terms of $|e(n)_{\max}|$ at steady-state.

Results of this example have demonstrated the usefulness of the simulation software for studying a high-order digital filter. The Cascade form has once again been shown to be extremely prone to finite word-length effects, and its response highly dependent on the value of the gain term, HO.

Table 4.2. A comparison of the three structures based upon the maximum percentage relative error, $|e(n)_{\max}|$, for the sixth-order low-pass filter function 4.6. -- Results are based on the effect of changing the word-length (NBITS) as the filter is subjected to a step-input of magnitude 0.5. Note the blank entries correspond to zero response.

Structure	Type of Input	$ e(n)_{\max} $			
		Number of Bits			
		24	16	12	8
Cascade	Step	0.025	1.32	-	-
Parallel	Step	0.0003	0.021	0.91	12.5
Normal	Step	0.0001	0.059	0.29	5.2

The Parallel and Normal forms perform much better, and in most cases the latter is less sensitive to the effects of finite word-length. Also, compared to the second-order system of Example 1, the sixth-order system is more susceptible to the effects of rounding.

Example 3

A second-order band-pass filter with a Quality Factor (Q) of 10, a center frequency of ω_0 of 1 rad/sec has a s-domain transfer function:

$$H(s) = \frac{s}{s^2 + 0.1s + 1} \quad (4.7)$$

Using the Bilinear transformation (Antoniou, 1979, p. 178) and a sampling time $T = 0.1$ sec., the corresponding z-domain transfer function is:

$$H(z) = \frac{0.0497509 (z^2 - 1)}{z^2 - 1.98010779z + 0.9900498} \quad (4.8)$$

where the constant gain is chosen such that at $\omega T = 0.1$, the overall gain is 10.

The filter function (4.8) is exposed to the three inputs discussed earlier in Example 1. Figures 4.7a and 4.7b show the pertinent outputs for the Normal form and a sinusoidal input. Figures 4.7c through 4.7f show plots of the response and the error function (4.3) for the Parallel form, and a broad-band random input, and also for the Cascade form and a step input. Table 4.3

Figure 4.7. The second-order, band-pass digital filter function (4.8) is simulated using the Normal, Parallel, and Cascade forms and driven respectively by a sinusoidal input $u(n) = 0.5 \sin(n)$, a random broad-band input, and a step input, $u(n) = 0.5$. -- Response plots shown in the following pages and described below are for simulations using 32, 24, 16, 12, and 8 bits. Error function plots according to (4.3) are also included.

- a. Response plot for a Normal form and a sinusoidal input. Curves marked 1 are for 32, 24, 16, and 12 bits. Curve marked 2 is for 8 bits.
- b. Error function versus time for the response plotted in 4.7a above. Dark base line is for 24 and 16 bits. Curve marked 1 is for 12 bits. Curve marked 2 is for 8 bits.
- c. Response plot for a Parallel form and a broad-band random input. Curves marked 1 are for 32, 24, 16, and 12 bits. A zero or "null" response was obtained for 8 bits (see text).
- d. Error function versus time for the response plotted in 4.7c above. Dark base line is for 24 and 16 bits. Curve marked 1 is for 12 bits. Curve marked 2 is for 8 bits.
- e. Response plot for a Cascade form and a step input. Curves marked 1 are for 32, 24, 16, and 12 bits. A null response was obtained for 8 bits.
- f. Error function versus time for the response plotted in 4.7e above. Dark base line and curve marked 1 are for 24, 16, and 12 bits. Curve marked 2 is for 8 bits.

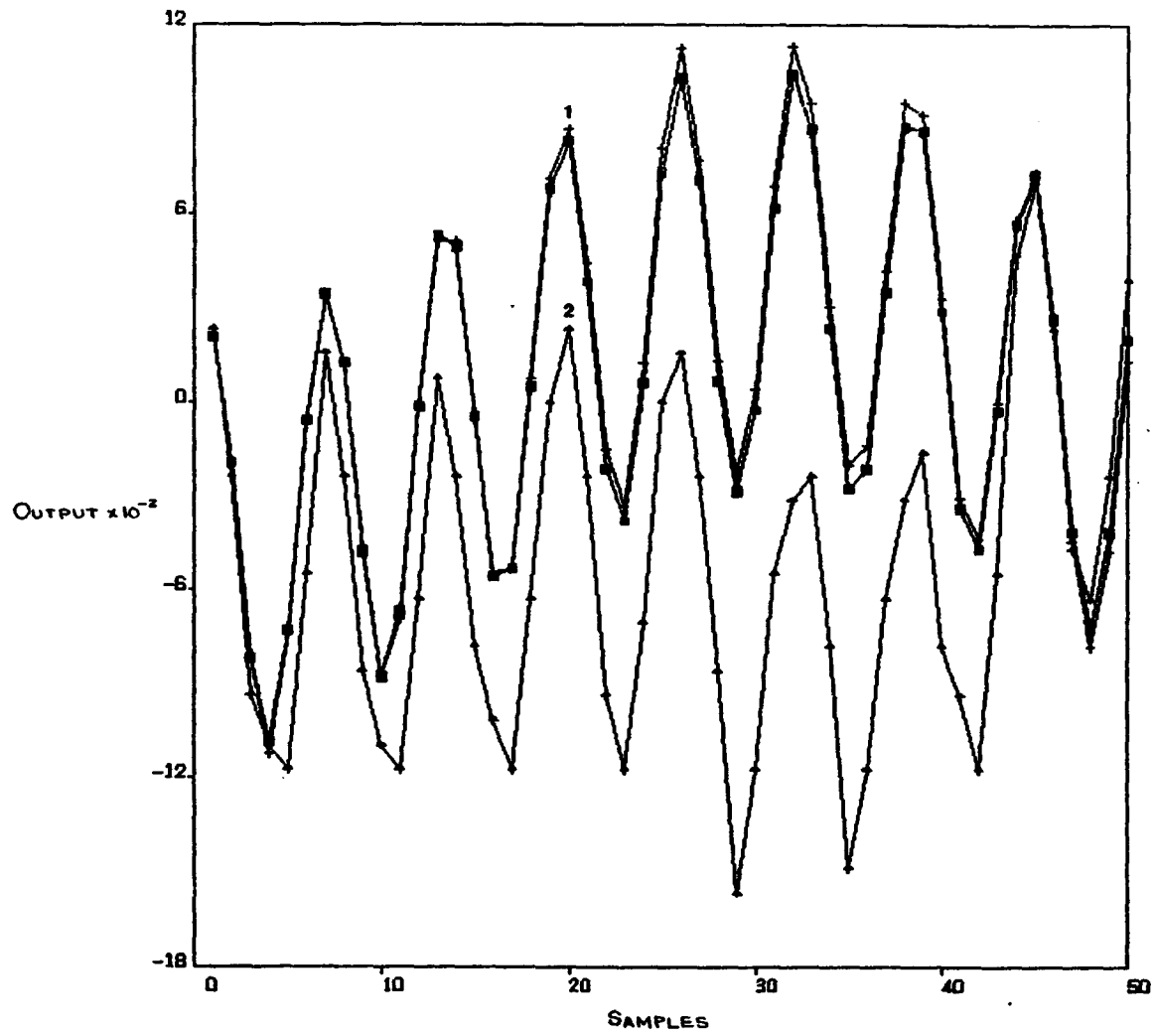


Figure 4.7a.

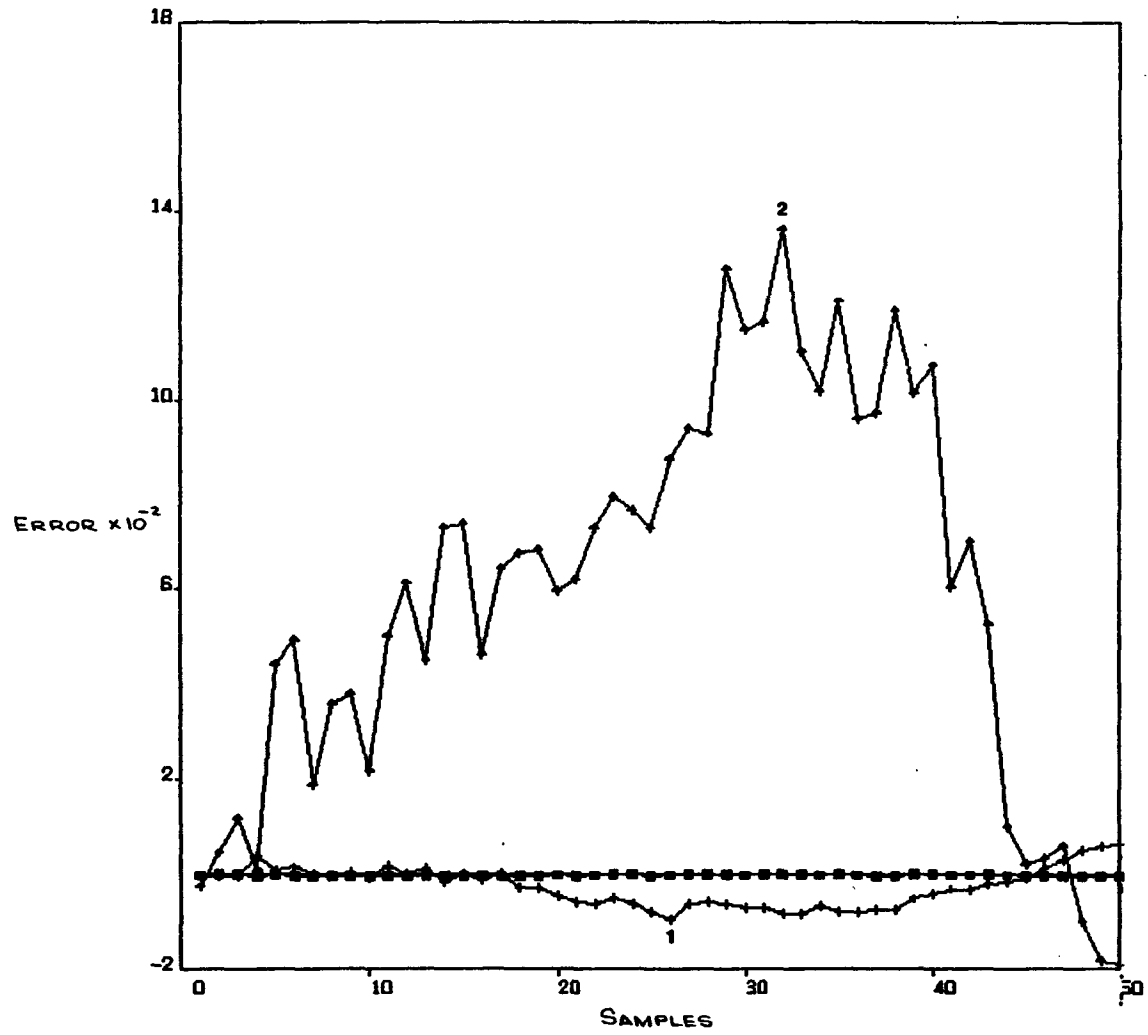


Figure 4.7b.

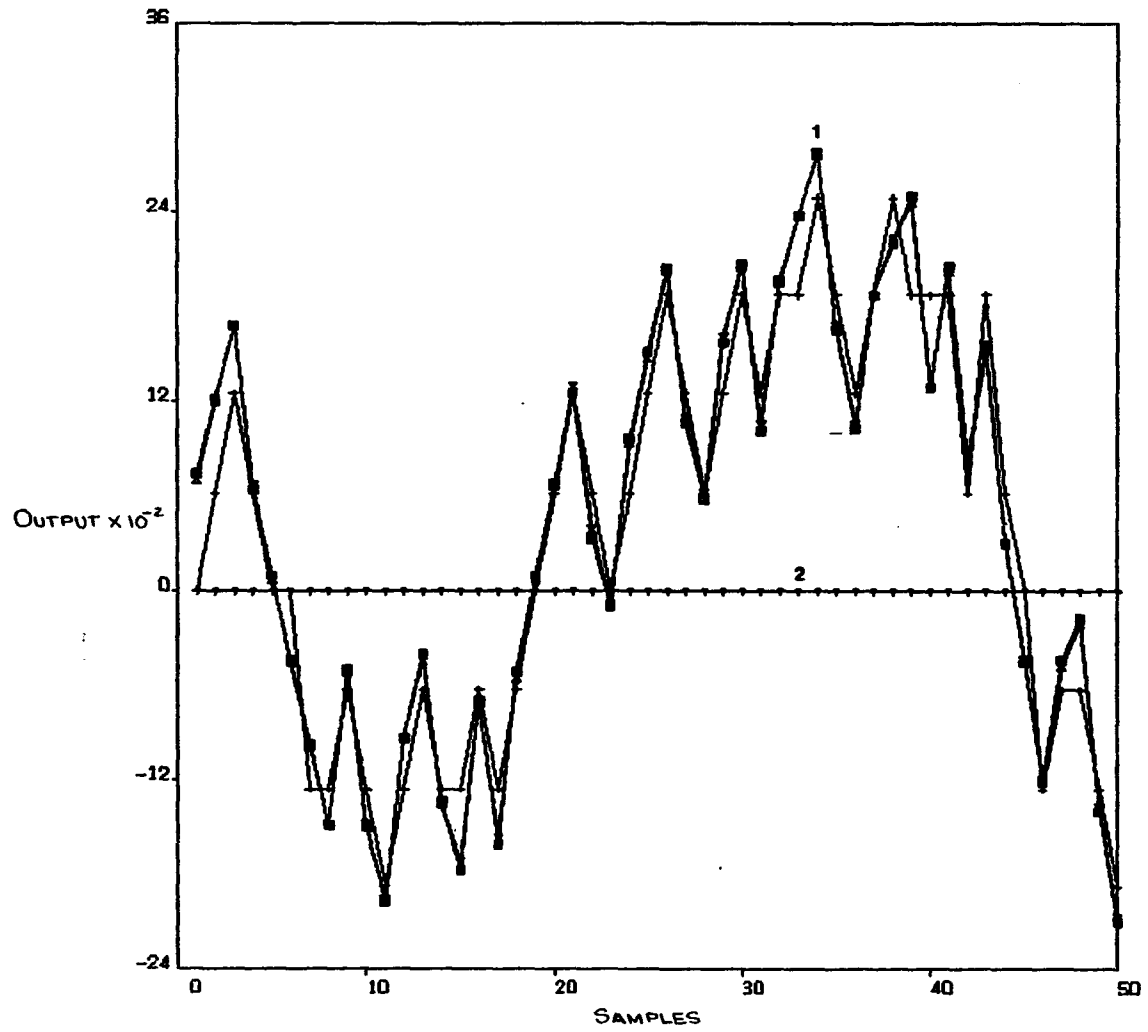


Figure 4.7c.

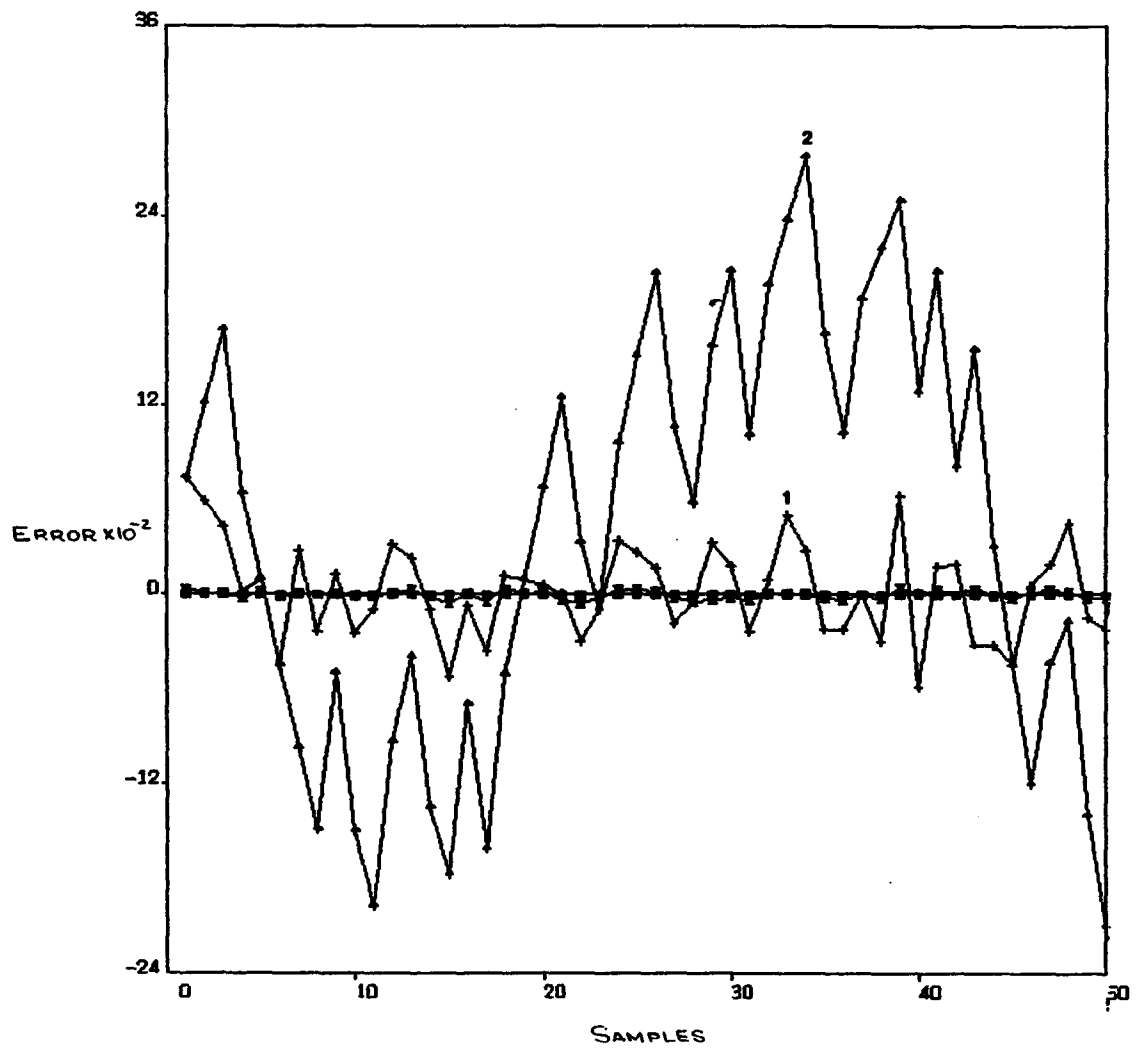


Figure 4.7d.

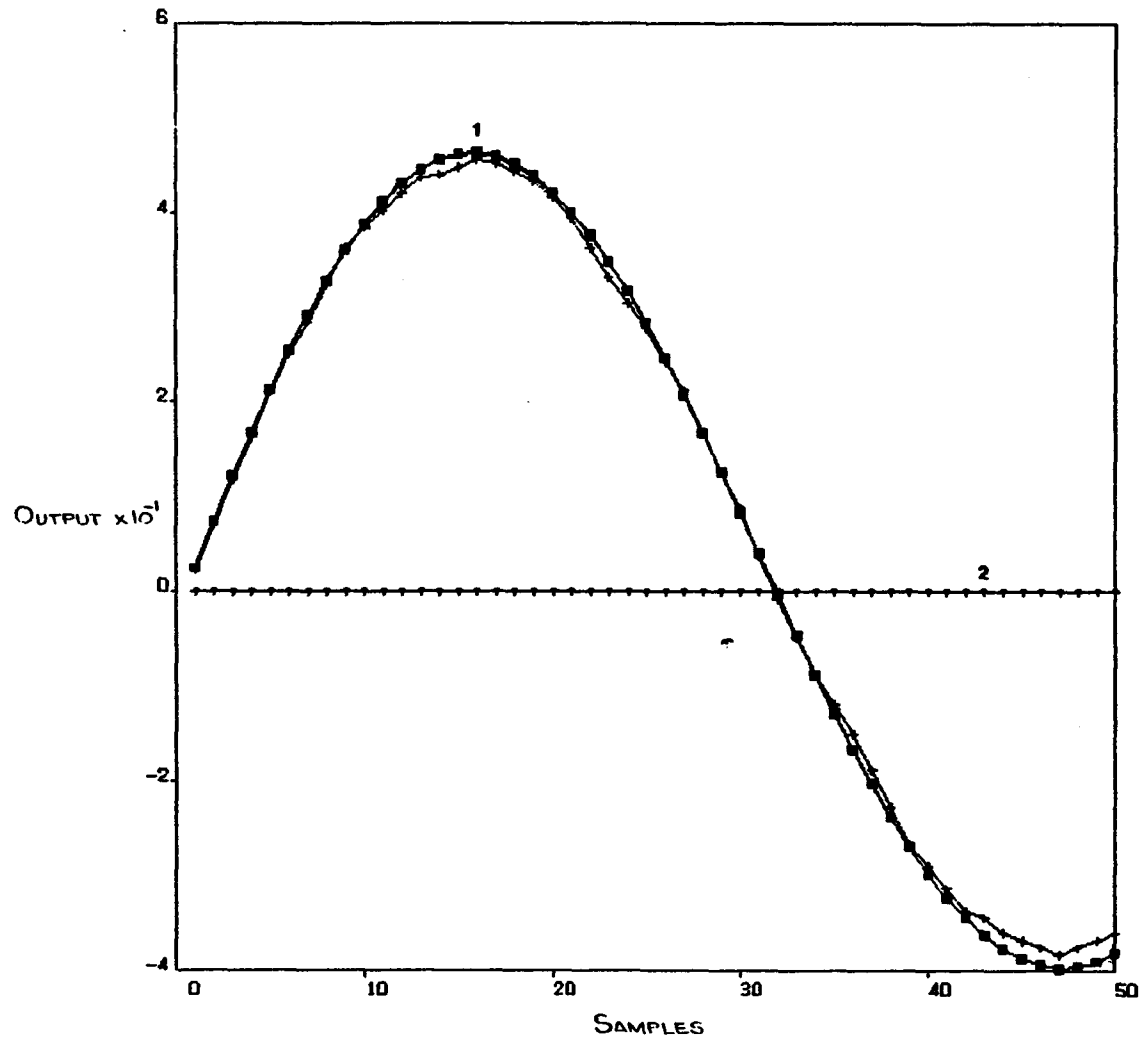


Figure 4.7e.

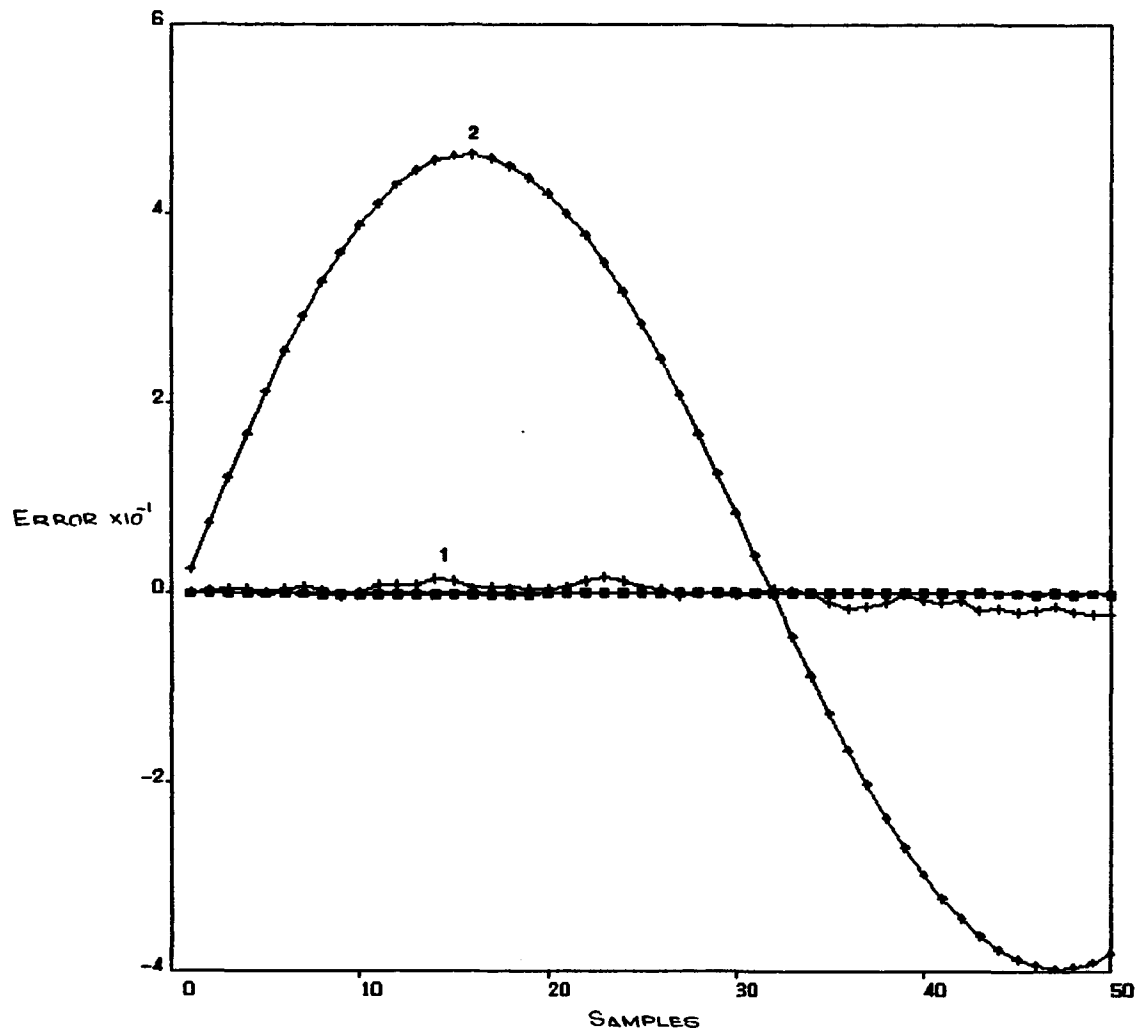


Figure 4.7f.

Table 4.3. A comparison of the three structures based upon the maximum percentage relative error, $|e(n)_{\max}|$, for the band-pass filter of 4.5. -- Results are based on the effect of changing the word-length (NBITS) as the filter is subjected to three standard inputs. Blank entries correspond to responses that suffer from 'dead-band' effects.

Structure	Type of Input	$ e(n)_{\max} $			
		Number of Bits			
		24	16	12	8
Cascade	Step	0.01	1.8	5.8	-
	Sine	0.76	8.3	>100	-
	Random	0.011	1.6	>100	-
Parallel	Step	0.004	1.3	21	-
	Sine	0.048	11.5	>100	-
	Random	0.019	17.9	25	-
Normal	Step	0.003	0.25	1.9	70
	Sine	0.004	0.83	9.1	>100
	Random	0.0009	0.56	7.3	48

summarizes results of the simulations as reflected by the maximum percentage of relative error, $|e(n)_{\max}|$.

Results show that the Normal form performs satisfactorily for a word-length of up to 12 bits, showing an error of less than 10%. The Cascade and Parallel forms, on the other hand, become unstable or show errors in excess of 20% for 12 bits.

For both the Cascade and Parallel forms, the inputs had to be scaled down by 2^{-3} and 2^{-7} , respectively, to prevent overflows. This explains the zero or "null" responses for 8 bits for these two structures. In summary, it can be said that for narrow-band filter functions the Normal form should be preferred over the Cascade and Parallel forms.

Appendix A shows the card deck setup for a typical simulation run on the CDC Cyber 175.

Summary and Conclusions

Results of the simulations tend to support the comments made earlier in Chapter 2 regarding the sensitivity of the Cascade form to finite word-length effects (Example 3). These effects become even more pronounced for higher-order filter functions (Example 2). For wide-band filter functions (Example 1), no significant difference in performance was observed between the three structures. While the Parallel form generally performs better than the Cascade form, it is also not suitable for narrow-band filter functions.

From the examples studied, it appears that the use of input scaling for dynamic range control is a primary cause for deterioration in performance, as the word-length is reduced, for both the Cascade and Parallel forms. Since the Normal form has ℓ_2 scaling included in its design, it is least susceptible to dynamic range problems.

Besides the three structures investigated in this report, other structures such as the Wave Digital filter (Fettweis, 1971) and the Gray-Markel Lattice filter (Gray and Markel, 1973) have also been shown to have low sensitivity to finite word-length effects (Taylor and Marshall, 1981).

This report has shown the advantages of the simulation of finite word-length effects, and also the effectiveness of simulation as a tool in the analysis and design of fixed-point digital filters.

APPENDIX A

CARD DECK SETUP FOR A TYPICAL
RUN ON THE CDC CYBER 175

1

The user may only specify data in a prescribed fashion to be outlined below. This data could be on separate cards or as a DEC-10 file.

Suppose a second-order, low-pass Butterworth filter with a transfer function of the form:

$$H(z) = \frac{N(z)}{D(z)} = \frac{0.0023293288 (z^2 + 2z + 1)}{z^2 - 1.85880613z + 0.86812344} =$$

$$\frac{HO (E3.z^2 + E2.z + E1)}{(F3.z^2 + F2.z + F1)}$$

is to be implemented as a Cascade structure and the response to a step input of magnitude 0.5 is to be investigated.

NBITS is initially set to 32 bits; subsequent runs use 24, 16, 12 and 8 bits, respectively.

The data cards/data file will be of the form:

```

22
50.0
2.0
2.0
32.0
4.0
1.0
0.5
0.0
0.0023293288
0.0
1.0
2.0
1.0

```

0.8681234454
-0.85880613
1.0
20.0
16.0
12.0
8.0
DIRECT

A couple of things need to be noted:

1. Except for the first and last data cards, all the other numbers are real.
2. The information on CARD 1 must be in columns 1 and 2. Except for the structure type (last card), all the other data must remain within the first 20 columns.

A breakdown of each card and the information it contains is as shown below:

- CARD 1: Total number of cards in data deck/data file.
- CARD 2: TMAX. Simulation time for each run. [120 is the upper limit on TMAX.]
- CARD 3: Order of Numerator Polynomial $N(z)$ [Max. = 10]
- CARD 4: Order of Denominator Polynomial $D(z)$ [Max. = 10]
- CARD 5: Number of bits (NBITS) for first run [Cyber 175 is limited to a maximum of 60 bits].
- CARD 6: Shifting factor for input. [Used only for Direct and Parallel forms.]

- CARD 7: Value for type of input:
1.0 = Step; 2.0 = Sine; 3.0 = Random
- CARD 8: Magnitude of Max. Value of Input [should remain less than 1.0].
- CARD 9: Angular frequency in radians (Only for Sine input. Maybe left equal to 0.0 otherwise).
- CARD 10: Gain for zero passband loss (HO).
- CARD 11: Flag for Calcomp/Printronic Plotter
0.0 = Line Printer Plot
1.0 = Line Printer Plot and Calcomp/Printronic
(TMAX must be set to 80)

The next three cards contain the coefficients of the Numerator polynomial in ascending order. Maximum number of coefficients is 11.

- CARD 12: Coefficient of z^0 (or constant term) = E1
CARD 13: Coefficient of $z = E2$
CARD 14: Coefficient of $z^2 = E3$

The following three cards contain the coefficients F(.) of the denominator polynomial in ascending order.

[Maximum number of coefficients is 11.]

- CARD 15: Coefficient of z^0 (or constant term) = F1
CARD 16: Coefficient of $z = F2$
CARD 17: Coefficient of $z^2 = F3$

CARD 18: Number of bits for second run [if set to 0, only
1 simulation run will be made].

CARD 19: Number of bits for third run.

CARD 20: Number of bits for fourth run.

CARD 21: Number of bits for fifth run.

CARD 22: Structure type.

Other possibilities are PARALLEL or NORMAL.

APPENDIX B

SIMULATION SOFTWARE LISTING

1 PROGRAM ASSIGN(INPUT,OUTPUT,PLOT,TAPE5=INPUT,TAPE6=OUTPUT,
 TAPE99=PLOT)
 THIS PROGRAM READS THE USER PROVIDED DATA AS AN ARRAY XX,
 AND ASSIGNS THE VARIOUS SIMULATION PARAMETERS TO THE ARRAY
 ELEMENTS. IT CONTROLS THE SIMULATION OF THE RELEVANT
 SUBROUTINES DEPENDING ON THE PARAMETERS PROVIDED BY THE USER.

NUM--TOTAL NUMBER OF CARDS IN THE DATA DECK
 XX(.)--ARRAY FORMED WITH INPUT DATA
 TYPE--TYPE OF FILTER STRUCTURE
 TMAX--SIMULATION RUN TIME
 M--ORDER OF NUMERATOR POLYNOMIAL OF TRANSFER FUNCTION
 N--ORDER OF DENOMINATOR POLYNOMIAL OF TRANSFER FUNCTION
 NBITS--NUMBER OF BITS FOR COEFFICIENT WORDLENGTH
 ISHIFT--SHIFTING PARAMETER(USED ONLY FOR SCALING INPUT
 IN DIRECT AND PARALLEL FORMS)
 U--TYPE OF INPUT
 U=1 FOR STEP INPUT
 U=2 FOR SINE INPUT
 U=3 FOR RANDOM INPUT
 ARG1 AND ARG2--ARGUMENTS USED IN CALLING TYPE OF INPUT
 HJ--GAIN FOR ZERO PASSBAND LOSS
 GRAF--FLAG FOR RASTER/CALCOMP PLOTTER
 F(J)--ARRAY OF NUMERATOR POLYNOMIAL COEFFICIENTS
 E(J)--ARRAY OF DENOMINATOR POLYNOMIAL COEFFICIENTS
 KK-- NUMBER OF SECOND ORDER SECTIONS
 JMAX--NUMBER OF POSSIBLE SIMULATION RUNS WITH DIFFERENT
 NBITS
 NUBITS(.)--NUMBER OF BITS FOR NEW RUN
 KMAX--COUNTER FOR NUMBER OF RUNS ACTUALLY MADE

COMMON/SYSVAR/EXIT,RLDONE,IFILE,IRUNNO,T,NTMAX,TNEXT
 COMMON/SYSVAR/DT,DTMAX,JTMIN,EMAX,EMIN,SY(35)
 COMMON/SYSVAR/NBITS,NBU,NBM,NBA,NY(40)
 COMMON/CASCADE/AO(10),A1(10),A2(10),B1(10),B2(10)
 COMMON/CASCADE/SHIFT1
 COMMON/PARLLL/FINDUT(101),FK1
 COMMON/PARLLL/GAMAO,GAMMAO(10),GAMMA1(10),BETA1(10),BETA2(10)
 COMMON/NORMAL/SIGMA(10),OMEGA(10),BB1(10),BB2(10),C1(10)
 COMMON/NORMAL/C2(10)
 COMMON/SETUP/M,N,F(10),E(10),U1(10),U2(10),V1(10),V2(10)
 COMMON/SETUP/D(10),DO(10)
 COMMON/PANODI/HO,ISHIFT
 COMMON/OUT/OUTPTT(120,5),ERROR(120,5),JMAX,KMAX
 COMMON/LIST/TMAX,KK
 COMMON/PLOT/IO,DDT,NXDIM,MN,NM,XXMIN,XXMAX

```

COMMON/INPUT/FINPT(120)
REAL INPUT,MAG
COMPLEX P1(10),R1(10),R2(10)
DIMENSION XX(50),NUBITS(10),JUTPUT(80,5),errand(80,5)
DATA XX/50*0.0/
DATA JUTPT,OUTPUT,ERROR,errand/600*0.0,400*0.0,600*0.0,400*0.0/

```

```

C
C
10 READ(5,10)NUM
   FJRMAT(12)
   K=NUM-2
20 READ(5,20)(XX(I),I=1,K)
   FORMAT(F20.0)
30 READ(5,30)IYPE
   FURMAT(A10)

```

```

C
C
   TMAX=XX(1)
   M=IFIX(XX(2))
   N=IFIX(XX(3))
   NBITS=IFIX(XX(4))
   ISHIFT=IFIX(XX(5))
   U=XX(6)
   ARG1=XX(7)
   ARG2=XX(8)
   HQ=XX(9)
   GRAF=XX(10)
   L=11
   MM=M+1
   DO 15 J=1,MM
   F(J)=XX(L)
   L=L+1
15 CONTINUE
   LL=11+MM
   NN=N+1
   DO 16 JJ=1,NN
   E(JJ)=XX(LL)
   LL=LL+1
16 CONTINUE
   KK=N/2

```

```

C
   JMAX=5
   KMAX=0
   LM=NUM-5
   NUBITS(1)=IFIX(XX(LM))
   LM=LM+1
   NUBITS(2)=IFIX(XX(LM))
   LM=LM+1
   NUBITS(3)=IFIX(XX(LM))

```

```

LM=LM+1
NUBITS(4)=IFIX(XX(LM))

IF(NUBITS(3).EQ.0)JMAX=4
IF(NUBITS(2).EQ.0)JMAX=3
IF(NUBITS(1).EQ.0)JMAX=2
IF(U.EQ.1.0)INPUT=STEP(ARG1)
IF(U.EQ.2.0)INPUT=SINE(ARG1,ARG2)
IF(U.EQ.3.0)INPUT=RANDOM(ARG1)

WRITE(6,35)
FURMAT('1',50X,'SIMULATION PARAMETERS')
WRITE(6,36)
FORMAT(50X,'*****')
WRITE(6,40)TMAX,M,N,NUBITS,SHIFT,U,ARG1,ARG2,H0,GRAF
FURMAT('/',10X,'TMAX=',F6.1,10X,'M=',I2,10X,'N=',I2,10X,
'NUBITS=',I2,10X,'SHIFT=',I2,10X,'U=',F3.1,10X,
'ARG1=',F4.1,10X,'ARG2=',F4.1,10X,'H0=',F14.9,10X,'GRAF=',
F3.1)
WRITE(6,41)
FURMAT('/',5X,'COEFFICIENTS OF NUMERATOR POLYNOMIAL')
WRITE(6,42)
FURMAT(5X,36('**'))
WRITE(6,50)(J,F(J),J=1,MM)
FURMAT('/',5X,'F(',I2,')=',F12.8)
WRITE(6,56)
FURMAT('/',5X,'COEFFICIENTS OF DENOMINATOR POLYNOMIAL')
WRITE(6,55)(K,E(K),K=1,NN)
FURMAT('/',5X,'E(',I2,')=',F12.8)
WRITE(6,57)KK
FURMAT('/',5X,'NUMBER OF SECOND ORDER SECTIONS=',I2)
WRITE(6,60)TYPE
FURMAT('/',5X,'STRUCTURE TYPE=',A10)
WRITE(6,91)NUBITS(1),NUBITS(2)
FURMAT('/',5X,'NUMBER OF BITS (NBITS) FOR 2ND. RUN=',I2,
5X,'NUMBER OF BITS (NBITS) FOR 3RD. RUN=',I2)
WRITE(6,92)NUBITS(3),NUBITS(4)
FURMAT('/',5X,'NUMBER OF BITS (NBITS) FOR 4TH. RUN=',I2,
5X,'NUMBER OF BITS (NBITS) FOR 5TH. RUN=',I2)

DO 80 JI=1,JMAX
KMAX=KMAX+1
IF(JI.NE.1)NUBITS=NUBITS(JI-1)
IF(TYPE.EQ.'DIRECT ')GO TO 25

```

33

35

36
40
1
1

41

42

50

56

55

57

60

91
1

92
1

33

35


```

10 STEP=MAG
   FINPT(I)=STEP
   RETURN
   END
C *****
C FUNCTION SINE
C *****
C
C FUNCTION SINE(MAG,W)
C THIS FUNCTION FORMS A SINUSOIDAL INPUT OF PEAK VALUE SPECIFIED
C BY MAG AND ANGULAR FREQUENCY W.
C COMMON/LIST/TMAX, KK
C COMMON/INPUT/FINPT(120)
C REAL INPUT, MAG
C
C IF (MAG.EQ.0.0)MAG=0.5
C IF (W.EQ.0.0)W=0.2
C ITMAX=IFIX(TMAX)
C DO 10 I=1, ITMAX
C T=FLOAT(I)
C SINE=MAG*SIN(W*T)
10 FINPT(I)=SINE
   RETURN
   END
C *****
C FUNCTION RANDOM
C *****
C
C FUNCTION RANDOM(DUMMY)
C THIS FUNCTION FORMS A BROAD-BAND INPUT BASED ON A NUMBER GENERATED
C BY THE RANDOM FUNCTION GENERATOR ,RANF.
C
C COMMON/LIST/TMAX, KK
C COMMON/INPUT/FINPT(120)
C REAL INPUT, MAG
C
C ITMAX=IFIX(TMAX)
C DO 10 I=1, ITMAX
C RANDOM=RANF(A)
C FINPT(I)=RANDOM
C IF (FINPT(I).GE.0.5)FINPT(I)=DUMMY
C IF (FINPT(I).LT.0.5)FINPT(I)=-DUMMY
10 CONTINUE
   RETURN
   END
C

```

```

C*****
C SUBROUTINE DSETUP
C*****
C SUBROUTINE DSETUP
C THIS SUBROUTINE USES A ROOT EXTRACTION ROUTINE TO FIND
C THE ROOTS OF THE DENOMINATOR AND NUMERATOR POLYNOMIALS
C OF A Z-DOMAIN TRANSFER FUNCTION.
C IT THEN CALCULATES THE FEED-FORWARD AND FEED-BACK
C COEFFICIENTS OF 2ND. ORDER SECTIONS FOR A CASCADE (DIRECT-2) FORM
C
C DIMENSION H(10),B(10),C(10)
C COMMON/CASCAD/AO(10),A1(10),A2(10),B1(10),B2(10)
C COMMON/SETUP/M,N,F(10),E(10),U1(10),U2(10),V1(10)
C COMMON/SETUP/V2(10),D(10),DO(10)
C COMMON/CASCAD/SHIFT1
C COMMON/PANOD1/HO,ISHIFT
C DATA U1,U2,V1,V2/10*0.0,10*0.0,10*0.0,10*0./
C
C
C NPLUS2=M+2
C CALL PRDUT(M,F,U1,V1,H,B,C,CONV,NPLUS2)
C
C CALL SETUP1(M,U1,V1,AO,A1,A2)
C
C
C NPLUS2=N+2
C CALL PRDUT(N,E,U2,V2,H,B,C,CONV,NPLUS2)
C
C CALL SETUP2(N,U2,V2,B1,B2)
C RETURN
C END
C
C*****
C SUBROUTINE SETUP1
C*****
C SUBROUTINE SETUP1(M,U1,V1,AO,A1,A2)
C THIS SUBROUTINE FORMS THE FEED-FORWARD COEFFICIENTS FOR THE
C CASCADE FORM
C DIMENSION F(10),E(10),U1(10),V1(10),H(10),C(10),ABSVALL(10)
C DIMENSION AO(10),A1(10),A2(10)
C
C J=0
C DO 50 I=1,M
C IF (V1(I).NE.0)GO TO 11
C IF (U1(I+1).NE.0 .AND. V1(I+1) .EQ. 0)GO TO 12
C AA1=FLOAT(M)
C REM=AMOD(AA1,2.0)
C IF (REM .NE. 0 .AND. U1(I+1) .EQ. 0)GO TO 13

```

```

13  GO TO 50
    J=J+1
    A1(J)=-U1(I)
    A2(J)=0.0
    A0(J)=1.0
    GO TO 50
12  J=J+1
    A2(J)=U1(I)*U1(I+1)
    A1(J)=- (U1(I)+U1(I+1))
    A0(J)=1.0
    GO TO 50
11  IF (I.EQ.1)GO TO 50
    ABSVAL1(I-1)=U1(I-1)*U1(I-1) + V1(I-1)*V1(I-1)
    ABSVAL1(I)=U1(I)*U1(I)+V1(I)*V1(I)
    IF (ABSVAL1(I).NE.ABSVAL1(I-1))GO TO 50
    J=J+1
    A2(J)=U1(I)*U1(I) +V1(I)*V1(I)
    A1(J)=-2.0*U1(I)
    A0(J)=1.0
50  CONTINUE
    RETURN
    END

```

```

C *****
C SUBROUTINE SETUP2
C *****
C
C SUBROUTINE SETUP2(N,U2,V2,B1,B2)
C THIS SUBROUTINE FORMS THE FEEDBACK COEFFICIENTS FOR A CASCADE
C FORM.
C DIMENSION F(10),E(10),U2(10),V2(10),N(10),C(10),ABSVAL2(10)
C DIMENSION B1(10),B2(10)
C
C J=0
C DO 70 I=1,N
C IF (V2(I).NE.0)GO TO 21
C IF (U2(I+1).NE. 0. .AND. V2(I+1).EQ.0.)GO TO 22
C A2=FLOAT(N)
C REM=AMOD(A2,2.0)
C IF (REM.NE. 0. .AND. U2(I+1).EQ. 0.)J=J+1
C B2(J)=0.0
C B1(J)=U2(I)
C GO TO 70
22 J=J+1
C B2(J)=U2(I)*U2(I+1)
C B1(J)=- (U2(I) +U2(I+1))
C GO TO 70
21 IF (I.EQ.1)GO TO 70
C ABSVAL2(I-1)=U2(I-1)*U2(I-1) + V2(I-1)*V2(I-1)
C ABSVAL2(I)=U2(I)*U2(I) + V2(I)*V2(I)

```

```

IF(ABSVAL2(I) .NE. ABSVAL2(I-1))GO TO 70
J=J+1
B1(J)=-2.0*U2(I)
B2(J)=U2(I)*U2(I) + V2(I)*V2(I)
70 CONTINUE
RETURN
END

C *****
C SUBROUTINE PSETUP
C *****
C SUBROUTINE PSETUP
C THIS SUBROUTINE FORMS THE FEED-FORWARD AND FEED-BACK
C COEFFICIENTS FOR A PARALLEL STRUCTURE.
C
C DIMENSION C(10),BC(10),CD(10),H(10)
C COMMON/PARLLL/FINDOUT(10),FK1
C COMMON/PARLLL/GAMMA0,GAMMA0(10),GAMMA1(10),BETA1(10),BETA2(10)
C COMMON/SETUP/M,N,F(10),E(10),U1(10),U2(10),V1(10)
C COMMON/SETUP/V2(10),U(10),DO(10)
C COMMON/PANQUI/H0,ISHIFT
C COMPLEX P1(10),R1(10),R2(10)
C
C NPLUS2=N+2
C J=0
C
C IF (M.NE. N) GO TO 22
C DO 33 I=1,M
C C(I)=(F(I)-E(I))*H0
33 CONTINUE
C MM=M-1
C GO TO 77
C
C 22 DO 66 L=1,M
C C(L)=F(L)
66 CONTINUE
C MM=M
C
C 77 CALL PROOT(N,E,U1,V1,H,BC,CD,CONV,NPLUS2)
C DO 44 I=1,N
C P1(I)=CHPLX(U1(I),V1(I))
44 CONTINUE
C CALL PFEXSD(MM,C,N,E,P1,R1,R2)
C
C J=0
C DO 55 I=1,N
C IF(AIMAG(P1(I)) .NE. 0)GO TO 11
C J=J+1
C GAMMA0(J)=R1(I)
C GAMMA1(J)=AIMAG(R1(I))

```

```

BETA1(J)=-REAL(PI(I))
BETA2(J)=0.0
GO TO 55
IF (I.EQ. 1)GO TO 25
J=J+1
NE=ABS(AIMAG(PI(I))).NE=ABS(AIMAG(PI(I-1)))GO TO 55
GAMMAO(J)=2.0*REAL(R1(I))
GAMMA1(J)=-2.0*REAL(R1(I))*REAL(PI(I))-2.0*AIMAG(R1(I))
AIMAG(PI(I))
BETA1(J)=-2.0*(REAL(PI(I)))
BETA2(J)=REAL(PI(I))*2.0+AIMAG(PI(I))*2.0
FORNAT('1',//,15X,4(F12.7))
CONTINUE
RETURN
END
C*****
C SUBROUTINE NSETUP
C *****
C SUBROUTINE NSETUP
C THIS SUBROUTINE FORMS THE FEED - FORWARD AND FEED-BACK
C COEFFICIENTS FOR A GENERAL NTH. ORDER NORMAL REALIZATION.
C THE FORMULATION OF THESE COEFFICIENTS IS BASED ON RELATIONS
C DERIVED BY HILLS, MULLIS AND ROBERTS IN THEIR PAPER ON
C "LOW ROUND-OFF NOISE AND NORMAL REALIZATIONS FOR FIXED POINT
C IIR DIGITAL FILTERS", IEEE TRANS. ON ASSP, AUG. 1981.
C *****
C COMMON/NORMAL/SIGMA(10),OMEGA(10),BB1(10),BB2(10),C1(10)
C COMMON/NUKHAL/C2(10)
C COMMON/SETUP/M,N,F(10),E(10),U1(10),U2(10),V1(10)
C COMMON/SETUP/V2(10),D(10),DD(10)
C COMMON/PANJDI/HO,ISHIFT
C *****
C COMMON/LIST/FIIMAX, KK
C DIMENSION P(10),RHU(10),X(10),Y(10),Q1(10),Q2(10),
C RAD(10),THEFA(10),RHOSQ(10),RHOFOR(10),ABSLAM(10),F1(10),
C PHIRAD(10),PHIDEG(10),PHI(10),R(10),THECAP(10),ARG1(10),
C ,ARG2(10),H(10),B(10),Z(10),F2(10),F3(10),F4(10),F5(10),
C BB(10),CC(10),DD(10),EE(10)
C *****
C DO I=1, KK
C JO(I)=HU*(1.0/FLOAT(KK))
C OPERATE ON NUMERATOR POLYNOMIAL
C NPLUS2=M+2
C CALL PROOT(M,F,U1,V1,H,B,C,CONV,NPLUS2)
C NPLUS2=N+2
C CALL PROUT(N,E,U2,V2,H,B,C,CONV,NPLUS2)
C FORM SECOND ORDER SECTIONS
C DO BB L=1,N/2
C FORM NUMERATOR SECOND SECTIONS

```

```

IF(V1(L) .NE. 0)GO TO 66
BB(L)=- (U1(N-L)+U1(N-L+1))
CC(L)=U1(N-L)*U1(N-L+1)
GO TO 77
66 BB(L)=-2.0*U1(N-L)
C CC(L)=U1(N-L)*U1(N-L) +V1(N-L)*V1(N-L)
77 FORM DENOMINATOR SECOND ORDER SECTIONS
DD(L)=-2.0*U2(N-L)
EE(L)=U2(N-L)*U2(N-L)+V2(N-L)*V2(N-L)
J=-1
IF(L.EQ.1)J=L
IF(L.EQ.5)J=L-2
IF(L.EQ.7)J=L-3
IF(L.EQ.9)J=L-4
Q1(J)=CC(L)-EE(L)
Q2(J)=BB(L)-DD(L)
88 CONTINUE
C C OPERATE ON DENOMINATOR POLYNOMIAL.

DO 70 I=1,N,2
P(I)=SQRT(U2(I)*U2(I) + V2(I)*V2(I))
P(I+1)=SQRT(U2(I+1)*U2(I+1) +V2(I+1)*V2(I+1))
IF(P(I) .NE. P(I+1))GO TO 70
SIGMA(I)=U2(I)
OMEGA(I)=ABS(V2(I))
RHU(I)=P(I)
X(I)=OMEGA(I)/SIGMA(I)
RAD(I)=ATAN(X(I))
PI=22.0/7.0
THETA(I)=RAD(I)*180.0/PI
RHUSQ(I)=RHU(I)*RHO(I)
RHOFOR(I)=RHOSQ(I)*RHOSQ(I)
ABSLAM(I)=RHO(I)
F1(I)=1.0-ABSLAM(I)*ABSLAM(I)
F2(I)=RHOSQ(I)*SIN(2.0*RAD(I))/(1.0-RHOSQ(I)+COS(2.0*RAD(I)))
THECAP(I)=-ATAN(-F2(I))/2.0
ZETA=5.0
J=I-1
IF(I.EQ.1)J=I
IF(I.EQ.5)J=I-2
F3(I)=SQRT(2.0*F1(I))
F4(I)=F3(I)/ZETA
F5(I)=((Q1(J) +SIGMA(I)*Q2(J))*2.0)/(OMEGA(I)*OMEGA(I))
Y(I)=(Q2(J)*OMEGA(I))/(-Q1(J)-SIGMA(I)*Q2(J))
PHIRAD(I)=ATAN(Y(I))
PHIDEG(I)=PHIRAD(I)*180.0/PI
PHI(I)=PHIDEG(I) +180.0
R(I)=SQRT(Q2(J)*Q2(J) +F5(I))
ARG1(I)=THECAP(I)+PI/4.0

```

```

ARG2(I)=(PHI(I)*PI/180.0 - THECAP(I) -PI/4.0)
BB1(I)=F4(I)*SIN(ARG1(I))
BB2(I)=F4(I)*COS(ARG1(I))
C1(I)=DO(J)*R(I)*COS(ARG2(I))/F4(I)
70 C2(I)=DO(J)*R(I)*SIN(ARG2(I))/F4(I)
CONTINUE
RETURN
END

C*****
C SUBROUTINE PROOT
C*****
C SUBROUTINE PROOT(N,A,U,V,H,B,C,CJNV,NPLUS2)
C THIS SUBROUTINE FORMS ALL THE ROOTS OF A POLYNOMIAL WITH REAL
C COEFFICIENTS OF THE FORM A1 +A2*S + .....+A(N+1)*S**N=0
C BY MIRIAM SHAPIRO (ADAPTED FROM LOS ALAMOS ROUTINE LA-PROOT
C BY T.L VORDEN)
C
DIMENSION A(NPLUS2),U(NPLUS2),V(NPLUS2),H(NPLUS2),B(NPLUS2),
* C(NPLUS2)
CONV=1.E-35
NC=N+1
CSEND COEFFICIENTS TO REDUCED COEFFICIENT STORAGE
DO1=1,NC
1 H(I)=A(I)
C INITIALIZE GUESSES AND SET REVERSAL INDICATOR NORMAL
P=0.
Q=0.
R=0.
IREV=1
C SCALING TO BE DONE AT THIS POINT
CREMOVE ALL ZERO ROOTS
3 IF(H(1))4,2,4
2 NC=NC-1
V(NC)=0.
U(NC)=0.
DO1002I=1,NC
1002 H(I)=H(I+1)
GOTO3
CTEST FOR VARIOUS DEGREES
4 IF(NC-1)5,100,5
5 IF(NC-2)7,0,7
6 R=-H(1)/H(2)
GOTO50
7 IF(NC-3)9,8,9
8 P=H(2)/H(3)
Q=H(1)/H(3)
GOTO70
CTEST TO REVERSE COEFFICIENTS AND DO SO IF TEST SUCCEEDS

```

```

9 IF (ABS(H(NC-1)/H(NC))-ABS(H(2)/H(1)))10,19,19
10 IREV=-IREV
M=NC/2
DO11 I=1,M
NL=NC+1-I
F=H(NL)
H(NL)=H(I)
11 IF(Q)13,12,13
12 P=0.
13 GOTU15
Q=1./Q
15 IF(R)16,19,16
16 R=1./R
CNEWTON, CALCULATE F(R) AND TEST FOR ROOT
E=5 E=20
3(NC)=H(NC)
C(NC)=H(NC)
B(NC+1)=0.
C(NC+1)=0.
NP=NC-1
20 DD49J=1,1000
DO21 I1=1,NP
I=NC-I1
B(I)=H(I)+R*B(I+1)
C(I)=B(I)+R*C(I+1)
21 IF(ABS(B(I)/H(I))-E)50,50,24
24 IF(C(2))23,22,23
22 R=R+1.
23 GOTU30
C MAKE A BAIRSTON REDUCTION AND CORRECT
DO37 I1=1,NP
I=NC-I1
B(I)=H(I)-P*B(I+1)-Q*B(I+2)
C(I)=B(I)-P*C(I+1)-Q*C(I+2)
37 C FOR CONVERGENCE OF BAIRSTON PROCESS.
TEST IF(H(2))32,31,32
31 IF(ABS(B(2)/H(1))-E)33,33,34
32 IF(ABS(B(2)/H(2))-E)33,33,34
33 IF(ABS(B(1)/H(1))-E)70,70,34
34 CBAR=C(2)-B(2)
D=C(3)+*2-CBAR+C(4)
35 P=P-2.
Q=Q+(Q+1.)
GOTO49.
36 P=P+(B(2)*C(3)-B(1)*C(4))/D
J=Q+(-B(2)+CBAR+B(1)*C(3))/D

```

```

49 CONTINUE
   E=E#10.
   IF(E-CONV)20,20,40
40 CONV=E
   GOTO20
CLINEAR. COMPUTE AND STORE LINEAR ROOTS
50 NC=NC-1
   V(NC)=0.
   IF(IREV)51,52,52
51 U(NC)=1./R
   GOTO53
52 U(NC)=R
53 DO54I=1,NC
54 H(I)=B(I+1)
   GOTO4
CQUADRATIC. SOLVE QUADRATIC AND STORE ROOTS
70 NC=NC-2
   IF(IREV)71,72,72
71 QP=1./Q
   PP=P/(Q+2.0)
   GOTO73
72 QP=0
   PP=P/2.0
73 F=(PP)**2-QP
   IF(F)74,75,75
CCASE OF IMAGINARY ROOTS.
74 U(NC+1)=-PP
   U(NC)=-PP
   V(NC+1)=SQRT(-F)
   V(NC)=-V(NC+1)
   GOTO76
CCASE OF REAL ROOTS.
75 U(NC+1)=-SIGN(ABS(PP)+SQRT(F),PP)
   V(NC+1)=0.
   U(NC)=QP/U(NC+1)
   V(NC)=0.
CFORM NEW REDUCED COEFFICIENTS
76 DO77I=1,NC
77 H(I)=B(I+2)
   GOTO4
100 RETURN
END

C
C*****
C      SUBROUTINE PFEXSD
C*****
C      SUBROUTINE PFEXSD(M,A,N,B,P,R1,R2)
C      SUBROUTINE FOR FINDING THE COEFFICIENTS IN THE PARTIAL
C      FRACTION EXPANSION FOR SIMPLE AND DOUBLE PLUES OF A
C      RATIONAL FUNCTION F(S)=A(S)/B(S) WHERE THE POLYNOMIALS

```



```

DIMENSION A(10),d(10),c(10),D(10),E(10)
COMPLEX P(10)
M=N
MP=M+1
DO 4 I=1,MP
  A(I)=R(I)
  IF(ABS(A(I)).GT.1.E-06)GO TO 13
  DO 7 I=1,M
    A(I)=A(I+1)
  P(M)=(0.,0.)
  M=M-1
  IF(M.EQ.0)RETURN
  MP=M+1
  GOTO 5
  IF(M-2)67,64,14
  ITER=0
  Q1=A(1)/A(3)
  Q2=A(2)/A(3)
  ITER=ITER+1
  C(MP)=A(MP)
  D(MP)=0.
  C(M)=A(M)-Q2*C(MP)
  D(M)=-C(MP)
  E(M)=0.
  IF(M.LT.4)GO TO 33
  MM=M-3
  DO 32 I=1,MM
    MI=M-I+1
    MI2=MI+2
    C(MI)=A(MI)-Q2*C(MI1)-Q1*C(MI2)
    D(MI)=-C(MI2)-Q2*E(MI1)-Q1*D(MI2)
    E(MI)=-C(MI2)-Q1*E(MI1)-Q1*E(MI2)
    Q1A=A(1)-Q1*C(3)-Q2*C(3)
    Q2A=A(2)-Q1*D(4)-Q2*D(3)
    G11=-C(4)-Q1*E(4)-Q2*E(3)
    G12=-C(4)*D(3)
    G21=-C(3)-Q1*E(3)
    G22=-C(3)*D(2)+G21
    DQ1=(-G11*Q1A+G21*Q2A)/DET
    DQ2=(G12*Q1A-G22*Q2A)/DET
    Q1=Q1+DQ1
    Q2=Q2+DQ2
  IF(ABS(Q2).LT.1.E-6)GO TO 47
  IF(ITER.GT.30)GO TO 69
  GO TO 17
  IF(ABS(DQ1).GT.1.E-06)GO TO 45
  DISC=J2+J2-4.*Q1

```

4

5

7

13

14

17

32

33

45

47

48


```

      VALUE=B(MC)
      IF(M.LE.0)RETURN
C
      ACCUMULATE EFFECT OF OTHER COEFFICIENTS
      DO 8 I=1,M
      K=MC-I
      8 VALUE=VALUE*P+B(K)
      RETURN
      END

C
C*****
C SUBROUTINE DIFPL
C*****
      SUBROUTINE DIFPL(N,A,M,B)
      HUELSHAN,"BASIC CIRCUIT THEORY WITH DIGITAL COMPUTATIONS",1972
      SUBROUTINE FOR DIFFERENTIATING A POLYNOMIAL OF THE FORM
      A(1)+A(2)*P+A(3)*P**2+ ... +A(M+1)*P**N
      TO OBTAIN THE RESULTING DIFFERENTIATED POLYNOMIAL
      B(1)+B(2)*P+B(3)*P**2+ ... +B(M+1)*P**M
      N-DEGREE OF INPUT POLYNOMIAL
      A-ARRAY OF COEFFICIENTS OF INPUT POLYNOMIAL
      M-DEGREE OF DIFFERENTIATED POLYNOMIAL (=N-1)
      B-OUTPUT ARRAY OF COEFFICIENTS OF DIFFERENTIATED
      POLYNOMIAL
      DIMENSION A(10),B(10)
      DO 3 I=1,N
      C=I
      3 B(I)=A(I+1)*C
      M=N-1
      RETURN
      END

C
C*****
C SUBROUTINE CASCAD
C*****
      SUBROUTINE CASCAD
      THIS SUBROUTINE SIMULATES A CASCADE (DIRECT 2) FORM USING
      SECOND ORDER SECTIONS.
      VALUE OF COEFF. B1 MAY NOT EXCEED 2.9 IN MAGNITUDE.
      VALUE OF COEFF. A1 MAY NOT EXCEED 2.9 IN MAGNITUDE.
      VALUE OF COEFF. A2 MAY NOT EXCEED 1.9 IN MAGNITUDE.
      ALL OTHER COEFFS. MAY NOT EXCEED 0.9 IN MAGNITUDE.

      DIMENSION SUM1(10),SUM2(10),ERR(10),SR1IN(10),SR1OUT(10)
      1,SR2OUT(10),FOBAK1(10),FOBAK2(10),FDFWD1(10),FDFWD2(10)

```

```

2,SUM3(10),SUM4(10),ERR1(10),ERR2(10),SR2IN(10)
COMMON/SYSVAR/EXIT,RLDONE,IFILE,IRUNNO,I,NITMAX,INEXT
COMMON/SYSVAR/DI,DI MAX,DTMIN,EMAX,EMIN,SY(35)
COMMON/SYSVAR/NBITS,NBU,NBM,NBA,NY(40)
COMMON/CASCAD/AO(10),A1(10),A2(10),B1(10),B2(10)
COMMON/CASCAD/SHIFT
COMMON/PANODI/HO,ISHIFT
COMMON/OUT/OUTPT(120,5),ERROR(120,5),JMAX,KMAX
COMMON/INPUT/FINPT(120)
COMMON/LIST/TMAX,KK

INITIALIZATION
SHIFTI=HO/2.0*ISHIFT
NBA=NBITS
NBM=NBITS
DO 10 I=1,KK
SR1OUT(I)=0.0
SR2OUT(I)=0.0
SUM1(I)=0.0
SUM4(I)=0.0
SUM3(I)=0.0

START SIMULATION
ITMAX=IFIX(ITMAX)
DO 100 N=1,ITMAX
FINPUT=FINPT(N)
OUTPTJ=SMULT(SHIFTI,FINPUT,1)
DO 200 J=1,KK
FDBAK1(J)=SMULT(SR1OUT(J),-B1(J)/3.0,2)
FDFWD1(J)=SMULT(SR1OUT(J),A1(J)/3.0,3)
FDBAK2(J)=SMULT(SR2OUT(J),-B2(J),4)
FDFWD2(J)=SMULT(SR2OUT(J),A2(J)/2.0,5)
TEMP1=SHADD(FDBAK1(J),TEMP1,0,2)
TEMP2=SHADD(FDBAK1(J),TEMP2,0,3)
SUM1(J)=SHADD(FDFWD1(J),FDFWD2(J),0,4)
TEMP3=SHADD(TEMP3,FDFWD1(J),0,5)
TEMP4=SHADD(TEMP4,FDFWD2(J),0,6)
SUM2(J)=SHADD(TEMP5,FDFWD2(J),0,7)
ERR1(J)=SHADD(OUTPTJ,SUM1(J),0,8)
ERR2(J)=SHADD(ERR1(J),SUM2(J),0,9)
OUTPTJ=SHADD(ERR1(J),TEMP6,0,10)
FORMAT(5X,10(2X,F10.7),/,/,5X,2(2X,F10.7))
SR1OUT(J)=ERR(J)
CONTINUE
101
200

```

```

      SHIFT2=2.0**ISHIFT
      FINPUT=SHIFT2*OUTPTJ
      OUTPTT(N,KMAX)=FINPUT
100  CONTINUE
      RETURN
      END

C*****
C      SUBROUTINE PARLEL
C*****
C      SUBROUTINE PARLEL
C      THIS SUB-ROUTINE SIMULATES A PARALLEL STRUCTURE AND CONNECTS
C      KK OF THEM IN PARALLEL.
C      VALUE OF COEFF. BETA1 MAY NOT EXCEED 2.9 IN MAGNITUDE.
C      VALUE OF COEFF. GAMMA0 MAY NOT EXCEED 1.9 IN MAGNITUDE.
C      ALL OTHER COEFFS. MAY NOT EXCEED 0.9 IN MAGNITUDE.
C
      DIMENSION SUM1(10),SUM2(10),ERR(10),SR1OUT(10),SR2OUT(10),
      1FDBAK1(10),FDBAK2(10),FDFWD1(10),FWD1(10),OUT1(10)
      COMMON/SYSVAR/EXIT,RLDUNE,IFILE,IRUNNO,T,NTMAX,TNEXT
      COMMON/SYSVAR/DT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
      COMMON/SYSVAR/NBITS,NBU,NBM,NBA,NY(40)
      COMMON/PARLLL/FINDOUT(101),FK1
      COMMON/PARLLL/GAMAO,GAMMA0(10),GAMMA1(10),BETA1(10),BETA2(10)
      COMMON/PANOD1/HO,ISHIFT
      COMMON/OUT/OUTPTT(120,5),ERROR(120,5),JMAX,KMAX
      COMMON/INPUT/FINPT(120)
      COMMON/LIST/TMAX,KK

C      INITIALIZATION
      FK1=1.0/2.0**ISHIFT
      DOUT=0.0
      NBA=NBITS
      NBM=NBITS
      DO 10 I=1,KK
      SR1OUT(I)=0.0
      GAMAO=HO
      SR2OUT(I)=0.0
      SUM1(I)=0.0
      SUM2(I)=0.0
      FINDOUT(I)=0.0
10  CONTINUE

C      START SIMULATION
      ITMAX=IFIX(TMAX)
      DO 100 N=1,ITMAX
      OUTPT=0.0
      FINPTJ=SMULT(FK1,FINPT(N),1)
      DOUT=SMULT(FINPTJ,GAMAO,7)
      DO 200 J=1,KK

```

```

FDBAK1(J)=SMULT(SR1OUT(J),-BETA1(J)/3.0,2)
FDFWD1(J)=SMULT(SR1OUT(J),GAMMA1(J),3)
FDBAK2(J)=SMULT(SR2OUT(J),-BETA2(J),4)
TEMP1=SHADD(FDBAK1(J),FDBAK2(J),0,1)
TEMP2=SHADD(FDBAK1(J),TEMP1,0,6)
SUM1(J)=SHADD(FDBAK1(J),TEMP2,0,2)
ERR(J)=SHADD(FINPTJ,SUM1(J),0,3)
FWD1(J)=SMULT(ERR(J),GAMMA0(J)/2.0,5)
TEMP3=SHADD(FDFWD1(J),FWD1(J),0,4)

```

```

SUM2(J)=SHADD(TEMP3,FWD1(J),0,6)
OUT1(J)=SUM2(J)
SR2OUT(J)=SR1OUT(J)
SR1OUT(J)=ERR(J)
OUTPT=OUTPT+OUT1(J)
CONTINUE
FINOUT(N)=SHADD(OUTPT,DOUT,0,5)
OUTPTI(N,KMAX)=FINOUT(N)/FK1
CONTINUE
RETURN
END

```

200
100

```

C*****
C SUBROUTINE NORMAL
C*****
C SUBROUTINE NORMAL
C THIS SUBROUTINE SIMULATES A 2ND. ORDER NORMAL STRUCTURE
C AND CASCADES KK OF THEM.

```

VALUES OF COEFFS. C1 AND C2 MAY NOT EXCEED 5.9 IN MAGNITUDE.
ALL OTHER COEFFS. MAY NOT EXCEED 0.9 IN MAGNITUDE.

```

1 DIMENSION SR1OUT(10),SR2OUT(10),SUM1(10),SUM2(10),
1 SUM3(10),FNUIN1(10),FNUIN2(10),FDBAK1(10),FDBAK2(10),
1 FDBAK3(10),ERR1(10),ERR2(10),FDBAK3(10),OUT(10),OUT1(10)
COMMON/SYSVAR/EXIT,RLDONE,IFILE,IRUNNO,I,NTMAX,INEXT
COMMON/SYSVAR/DI,DI MAX,DTMIN,EMAX,EMIN,S(35)
COMMON/SYSVAR/NBITS,NBU,NBH,NBA,NY(40)
COMMON/NORMAL/SIGMA(10),OMEGA(10),BB1(10),BB2(10),C1(10)
COMMON/NORMAL/C2(10)
COMMON/SETUP/H,N,F(10),E(10),U1(10),U2(10),V1(10)
COMMON/SETUP/V2(10),D(10),D0(10)
COMMON/PANODI/H0,I SHIFT
COMMON/OUT/OUTPTI(120,5),ERRDR(120,5),JMAX,KMAX
COMMON/LIST/TMAX,KK
COMMON/INPUT/FINPT(120)
INITIALIZATION

```

C
C

NBA=NBITS
NBM=NBITS

```

DD 15 K=2, KK
IF(K.EQ.2) L=K+1
IF(K.EQ.3) L=K+2
BB1(K)=BB I(L)
BB2(K)=BB2(L)
C1(K)=C1(L)
C2(K)=C2(L)
SIGMA(K)=SIGMA(L)
OMEGA(K)=OMEGA(L)
CONTINUE
DD 10 I=1, KK
SR1OUT(I)=0.0
SR2OUT(I)=0.0
SUM1(I)=0.0
SUM2(I)=0.0
CONTINUE
START SIMULATION
ITMAX=ITMAX
DO 100 K=1, IIMAX
FINPUT=FINPT(K)
DO 200 J=1, KK
DOUT=SMULT(DO(J), FINPUT, 9)
FNUIN1(J)=SMULT(BB1(J), FINPUT, 1)
FNUIN2(J)=SMULT(BB2(J), FINPUT, 2)
FDBAK1(J)=SMULT(SIGMA(J), SR1OUT(J), 3)
FDBAK2(J)=SMULT(SIGMA(J), SR2OUT(J), 4)
FDBAK3(J)=SMULT(OMEGA(J), SR1OUT(J), 5)
FDBAK4(J)=SMULT(OMEGA(J), SR1OUT(J), 6)
SUM1(J)=SHADD(FNUIN1(J), FDBAK3(J), 0, 1)
SUM2(J)=SHADD(FNUIN2(J), FDBAK4(J), 0, 2)
ERR1(J)=SHADD(SUM1(J), FDBAK1(J), 0, 3)
ERR2(J)=SHADD(SUM2(J), FDBAK2(J), 0, 4)
OUTPI1=SMULT(SR1OUT(J), C1(J)/6.0, 7)
OUTPI2=SMULT(SR2OUT(J), C2(J)/6.0, 8)
TEMP1=SHADD(OUTPI1, OUTPI1, 0, 5)
TEMP2=SHADD(TEMP1, OUTPI1, 0, 6)
TEMP3=SHADD(TEMP2, OUTPI1, 0, 7)
TEMP4=SHADD(TEMP3, OUTPI1, 0, 8)
TEMP5=SHADD(TEMP4, OUTPI1, 0, 9)
TEMP6=SHADD(TEMP5, OUTPI1, 0, 10)
TEMP7=SHADD(TEMP6, OUTPI2, 0, 11)
TEMP8=SHADD(TEMP7, OUTPI2, 0, 12)
TEMP9=SHADD(TEMP8, OUTPI2, 0, 13)
TEMP10=SHADD(TEMP9, OUTPI2, 0, 14)
OUTJ=SHADD(TEMP10, OUTPI2, 0, 15)
FINPUT=SHADD(OUTJ, DOUT, 0, 16)
SR1OUT(J)=ERR1(J)
SR2OUT(J)=ERR2(J)
CONTINUE
200

```

```

100      OUTPIT(K,KMAX)=SHADD(OUT,OUTJ,0,13)
        CONTINUE
        RETURN
        END
C*****
C      FUNCTION SHADD
C*****
C      FUNCTION SHADD(X,Y,NSHIFT,I)
C*****
C      J.V.WAIT,FEB.1980.
C      SUBROUTINE TO ADD TWO PROBLEM VARIABLES AND SHIFT
C      TO RIGHT,I.E. DIVIDE BY 2**NSHIFT
C      WHERE
C      X AND Y ARE TWO VARIABLES TO BE ADDED
C      NSHIFT IS THE SHIFTING FACTOR
C      I IS INDEX UN OVERFLOW IN QUANTIZERXXXXXXXXXX
C      SHADD IS THE SUM ROUNDED TO NBA BITS.
C      LOGICAL EXIT,RLDONE,OVER
C      COMMON/SYSVAR/EXIT,DTMAX,DTMIN,EMAX,EMIN,SY(35)
C      COMMON/SYSVAR/NBITS,NBU,NBM,NBA,NY(40)
C*****
C      ADD TWO INPUTS.
C      SHADD=X+Y
C      DIVIDE BY 2**NSHIFT
C      SHADD=SHADD/(2.0**NSHIFT)
C      QUANTIZE
C      IF(NBA.GE.0)SHADD=QUAN(SHADD,NBA,I)
C      IF(OVER(SHADD,NBITS))WRITE(6,1000)I,T
C      FORMAT(6HSHADD(,I4,17H)OVERFLOWED AT T=,E15.4)
C      RETURN
C      END
1000
C*****
C      FUNCTION SMULT
C*****
C      FUNCTION SMULT(X,Y,I)
C*****
C      J.V.WAIT,FEB. 1980.
C      SUBROUTINE TO SIMULATE A FIXED-POINT MULTIPLIER
C      GENERATES SMULT=X*Y,BUT IN FIXED POINT
C      IF NBM.GT.0,IT ROUNDS X AND Y TO NBM BITS,AND THEN GENERATES
C      A PRODUCT WITH 2*NBM BIT QUANTIZATION.
C      WHERE
C      X AND Y ARE INPUT FACTORS

```

```

C C SMULT IS PRODUCT
C C I IS MULTIPLIER INDEX, TO TELL WHERE OVERFLOW OCCURS
C C LOGICAL EXIT, RLDOONE
COMMON/SYSVAR/EXIT,RLDOONE,IFILE,IRUNNO,T,NTMAX,TNEXT
COMMON/SYSVAR/DI,DTMAX,DTMIN,EMAX,EMIN,SY(35)
COMMON/SYSVAR/NBITS,NBU,NBH,NBA,NY(40)
XX=X
YY=Y
IF(NBH.LE.0)GO TO 100
ROUND TO NBH BITS
XX=QUAN(XX,NBH,I)
YY=QUAN(YY,NBH,I)
SMULT=XX*YY
RETURN
100 END

C C *****
C C FUNCTION OVER *****
C C LOGICAL FUNCTION OVER(X,NB) *****
C C J.V.WAIT FEB 1980 *****
C C CHECK FOR OVERFLOW OF NORMALIZED REAL VARIABLE *****
C C NB IS NO. OF BITS TWO'S COMPLEMENT *****
M=NB-1
TOP=1.0-(1.0/LOAT(2**M))
OVER=.FALSE.

IF(X.LT.-1.0)OVER=.TRUE.
IF(X.GT.TOP)OVER=.TRUE.
RETURN
END

C C *****
C C FUNCTION QUAN *****
C C *****
C C FUNCTION QUAN(X,NB,I) *****
C C J.V.WAIT, FEB. 1980. *****

LOGICAL EXIT,RLDOONE
COMMON/SYSVAR/EXIT,RLDOONE,IFILE,IRUNNO,T,NTMAX,TNEXT
COMMON/SYSVAR/DI,DTMAX,DTMIN,EMAX,EMIN,SY(35)
COMMON/SYSVAR/NBITS,NBU,NBH,NBA,NY(40)
LOGICAL OVER,OVLW
SIMULATES A NBIT QUANTIZER (ROUNDING)
ASSUMES NB=ND. JF BITS
USE A DIFFERENT I FOR EACH QUANTIZER

```

```

M=NB-1
FACT=FLOAT(2**M)
INTEG=IQUAN(FACT*X)
STORE=FRAC(INTEG,NB)
OVFLW=OVER(STORE,NB)
IF(OVFLW)WRITE(6,100)I,T,X
100  FORMAT(/,11H QUANTIZER,I3,18H OVERFLOWED AT T= ,E15.4,
1E15.4,8H INPUT = ,E15.4)
TOP=1.0-(1.0/FACT)
IF(STORE.LT.-1.0)STORE=-1.0
IF(STORE.GT. TJP)STORE=TOP
200  QUAN=STORE
RETURN
END

C
C*****
C      FUNCTION FRAC
C*****
C      FUNCTION FRAC(I,NB)
C      J.V. WAIT FEB. 1980
C      CONVERTS INTEGER TO FRACTION
C      NB IS NO. OF BITS
C      FACT=FLOAT(2**(NB-1))
C      FRAC=FLOAT(1)/FACT
C      RETURN
C      END

C
C*****
C      FUNCTION IQUAN
C*****
C      FUNCTION IQUAN(X)
C      IQUAN IS NEAREST INTEGER TO X
C      J.V. WAIT , MARCH 1980
C      IX=IFIX(X)
C      XA=ABS(X)
C      INTEG=IFIX(XA)
C      DLO=XA-FLOAT(INTEG)
C      DHI=1.0-DLO
C      IF(DHI.LT.DLO)INTEG=INTEG+1
C      IQUAN=ISIGN(INTEG,IX)
C      RETURN
C      END

C
C*****
C      SUBROUTINE LISTO
C*****
C      SUBROUTINE LISTO
C      THIS SUBROUTINE IS USED TO LIST THE FEED-BACK AND FEED-FORWARD

```

```

C      COEFFICIENTS AND OUTPUT ALL THE ARRAYS OF OUTPUT VALUES.
COMMON/CASCAD/A0(10),A1(10),A2(10),B1(10),B2(10)
COMMON/CASCAD/SHIFT1
COMMON/OUT/OUTPTT(120,5),ERROR(120,5),JMAX,KMAX
COMMON/LIST/TMAX, KK

C
C      PRINT THE FEED-FORWARD AND FEED-BACK COEFFICIENT VALUES

18  WRITE(6,18)
    FORMAT("1",40X,"FEED-FORWARD AND FEED-BACK COEFFICIENTS")
17  WRITE(6,17)
    FORMAT(40X,40("-"))
19  WRITE(6,19)
    FORMAT(/,15X,"A0",12X,"A1",12X,"A2",12X,"B1",12X,"B2")
    DO 50 J=1, KK
    WRITE(6,60)A0(J),A1(J),A2(J),B1(J),B2(J)
60  FORMAT(/,10X,3(F12.7),5X,2(F12.7))
50  CONTINUE

C
    ITMAX=IFIX(TMAX)
    WRITE(6,70)
70  FORMAT("1",2X,"TIME",10X,"OUTPUT")
    DO 22 N=1,ITMAX
    WRITE(6,10)FLOAT(N),(OUTPTT(N,JJ),JJ=1,JMAX)
10  FORMAT(2X,F5.2,5X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7,5X,
1  F12.7)
22  CONTINUE
    WRITE(6,71)
71  FORMAT("1",2X,"TIME",10X,"ERROR")
    DO 20 JI=1,ITMAX
    DO 30 JK=2,JMAX
    ERROR(JI,JK)=OUTPTT(JI,1)-OUTPTT(JI,JK)
30  CONTINUE
20  CONTINUE
    DO 23 NK=1,ITMAX
    WRITE(6,11)FLOAT(NK),(ERROR(NK,NJ),NJ=2,JMAX)
11  FORMAT(2X,F5.2,5X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7)
23  CONTINUE
    RETURN
    END
C *****
C      SUBROUTINE LIST
C *****
C      SUBROUTINE LIST
C      THIS SUBROUTINE LISTS THE FEED-FORWARD AND FEED-BACK
C      COEFFICIENTS OF A PARALLEL STRUCTURE. IT ALSO PRINTS THE
C      ARRAY OF OUTPUT VALUES.

```

```

COMMON/LIST/TMAX, KK
COMMON/PARLLL/FINDOUT(101), FK1
CJMMUN/PARLLL/GAMAO, GAMMAO(10), GAMMA1(10), BETA1(10), BETA2(10)
COMMON/OUT/OUTPTT(120,5), ERROR(120,5), JMAX, KMAX

```

C

```

14 WRITE(6,14)
   FORMAT("1",40X,"FEED-BACK AND FEED-FORWARD COEFFICIENTS")
   WRITE(6,15)
15   FORMAT(40X,40("-"))
   WRITE(6,18)
18   FORMAT(//,12X,"GAMMAO",25X,"GAMMA1",20X,"BETA1",30X,"BETA2")
   WRITE(6,19)(GAMMAO(N),GAMMA1(N),BETA1(N),BETA2(N),N=1, KK)
19   FORMAT(//,4(10X,F12.9,11X)/)
   ITMAX=TMAX
   WRITE(6,21)
21   FORMAT("1",2X,"TIME",10X,"OUTPUT")
   DO 22 J=1,ITMAX
   WRITE(6,10)FLOAT(J),(OUTPTT(J, JJ), JJ=1, JMAX)
10   FORMAT(2X,F5.2,5X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7,5X,
1   F12.7)
22   CONTINUE
   WRITE(6,71)
71   FORMAT("1",2X,"TIME",10X,"ERROR")
   DO 20 JI=1,ITMAX
   DO 30 JK=2,JMAX
   ERROR(JI, JK)=OUTPTT(JI,1)-OUTPTT(JI, JK)
30   CONTINUE
20   CONTINUE
   DO 23 NK=1,ITMAX
   WRITE(6,11)FLOAT(NK),(ERROR(NK, NJ), NJ=2, JMAX)
11   FORMAT(2X,F5.2,5X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7)
23   CONTINUE
   RETURN
   END

```

C
C
C
C
C

```

*****
SUBROUTINE LIST
*****

SUBROUTINE LIST
THIS SUBROUTINE LISTS THE FEED-FORWARD AND FEED-BACK
COEFFICIENTS OF A NORMAL STRUCTURE. IT ALSO PRINTS THE ARRAY OF
OUTPUT VALUES.

COMMON/LIST/TMAX, KK
COMMON/SETUP/M, N, F(10), E(10), J1(10), U2(10), V1(10)
COMMON/SETUP/V2(10), D(10), DO(10)

```

```

COMMON/NORMAL/SIGMA(10),OMEGA(10),BB1(10),BB2(10),C1(10)
COMMON/NORMAL/C2(10)
COMMON/OUT/OUTPTI(120,5),ERROR(120,5),JMAX,KMAX

```

```

C
18 WRITE(6,18)
19 FORMAT(40X,"FEED-FORWARD AND FEED-BACK COEFFICIENTS")
17 WRITE(6,17)
18 WRITE(6,17)
19 WRITE(6,17)
20 WRITE(6,17)
21 WRITE(6,17)
22 WRITE(6,17)
23 WRITE(6,17)
24 WRITE(6,17)
25 WRITE(6,17)
26 WRITE(6,17)
27 WRITE(6,17)
28 WRITE(6,17)
29 WRITE(6,17)
30 WRITE(6,17)
31 WRITE(6,17)
32 WRITE(6,17)
33 WRITE(6,17)
34 WRITE(6,17)
35 WRITE(6,17)
36 WRITE(6,17)
37 WRITE(6,17)
38 WRITE(6,17)
39 WRITE(6,17)
40 WRITE(6,17)
41 WRITE(6,17)
42 WRITE(6,17)
43 WRITE(6,17)
44 WRITE(6,17)
45 WRITE(6,17)
46 WRITE(6,17)
47 WRITE(6,17)
48 WRITE(6,17)
49 WRITE(6,17)
50 WRITE(6,17)
51 WRITE(6,17)
52 WRITE(6,17)
53 WRITE(6,17)
54 WRITE(6,17)
55 WRITE(6,17)
56 WRITE(6,17)
57 WRITE(6,17)
58 WRITE(6,17)
59 WRITE(6,17)
60 WRITE(6,17)
61 WRITE(6,17)
62 WRITE(6,17)
63 WRITE(6,17)
64 WRITE(6,17)
65 WRITE(6,17)
66 WRITE(6,17)
67 WRITE(6,17)
68 WRITE(6,17)
69 WRITE(6,17)
70 WRITE(6,17)
71 WRITE(6,17)
72 WRITE(6,17)
73 WRITE(6,17)
74 WRITE(6,17)
75 WRITE(6,17)
76 WRITE(6,17)
77 WRITE(6,17)
78 WRITE(6,17)
79 WRITE(6,17)
80 WRITE(6,17)
81 WRITE(6,17)
82 WRITE(6,17)
83 WRITE(6,17)
84 WRITE(6,17)
85 WRITE(6,17)
86 WRITE(6,17)
87 WRITE(6,17)
88 WRITE(6,17)
89 WRITE(6,17)
90 WRITE(6,17)
91 WRITE(6,17)
92 WRITE(6,17)
93 WRITE(6,17)
94 WRITE(6,17)
95 WRITE(6,17)
96 WRITE(6,17)
97 WRITE(6,17)
98 WRITE(6,17)
99 WRITE(6,17)
100 WRITE(6,17)
101 WRITE(6,17)
102 WRITE(6,17)
103 WRITE(6,17)
104 WRITE(6,17)
105 WRITE(6,17)
106 WRITE(6,17)
107 WRITE(6,17)
108 WRITE(6,17)
109 WRITE(6,17)
110 WRITE(6,17)
111 WRITE(6,17)
112 WRITE(6,17)
113 WRITE(6,17)
114 WRITE(6,17)
115 WRITE(6,17)
116 WRITE(6,17)
117 WRITE(6,17)
118 WRITE(6,17)
119 WRITE(6,17)
120 WRITE(6,17)

```

```

C
ITMAX=IFLX(ITMAX)
WRITE(6,21)
FORMAT(40X,"FEED-FORWARD AND FEED-BACK COEFFICIENTS")
DO 22 J=1,ITMAX
WRITE(6,20)(SIGMA(I),OMEGA(I),BB1(I),BB2(I),C1(I),C2(I),I=1,KK)
FORMAT(//,10X,F12.7,10X,F12.7,10X,F12.7,10X,F12.7,10X,
1 F12.7)
22 CONTINUE
WRITE(6,71)
FORMAT(40X,"FEED-FORWARD AND FEED-BACK COEFFICIENTS")
DO 23 JI=1,ITMAX
DO 30 JK=2,JMAX
ERROR(JI,JK)=OUTPTI(JI,1)-OUTPTI(JI,JK)
CONTINUE
DO 23 NK=1,ITMAX
WRITE(6,11)(FLOAT(NK),ERROR(NK,NJ),NJ=2,JMAX)
FORMAT(2X,F5.2,2X,F12.7,5X,F12.7,5X,F12.7,5X,F12.7,5X,
11 23)
CONTINUE
RETURN
END

```

```

C *****
C SUBROUTINE LPLOT
C *****
C SUBROUTINE LPLOT
C THIS SUBROUTINE PROVIDES FOR LINE PRINTER PLOTTING.
C
COMMON/PLOT/IO,DDI,NXDIM,MN,NM,XXMIN,XXMAX
COMMON/LIST/TMAX,KK

```



```

C/ LINE(JZERO)=IZERO
C/ START SEARCH THRU X ARRAY FUNCTIONS AND DETERMINE FOR EACH WHERE
C/ THE GRAPH MARKER SHOULD BE, AND LOAD IT INTO THE GRAPH LINE BUFFER
C/
C/ DO 20 I=1,M
C/ IJ=(X(LINES,I)-XMIN)/XSCALE+1.5
C/ IF(IJ.LE.0)IJ=1
C/ IF(IJ.GT.101)IJ=101
C/ LINE(IJ)=IDIGIT(I)
C/
C/ PRINT ONE LINE OF THE GRAPH.
C/
C/ PRINT 21, T,(LINE(I),I=1,101),LINES
C/ FORMAT(2X, E20.8,3X,101A1,3X,13)
C/
C/ IF GRAPH NOT COMPLETE, RETURN AND START A NEW LINE
C/
C/ CONTINUE
C/
C/ PRINT LOWER BORDER FOR GRAPH
C/
C/ PRINT 23, (XREF(1),I=1,11)
C/ FORMAT(25X,101(1H=),/25X,1H+,10(9X,1H+)/22X,11( E9.2,1X)//)
C/ RETURN
C/ END
C/ LIST,NONE
C/ SUBROUTINE SCALE(X,NXDIM,N,M,XMAX,XMIN)
C/
C/ SCALE FINDS THE MAXIMUM AND MINIMUM VALUES OF THE ARRAY X(N,M)
C/
C/ DIMENSION X(NXDIM,M)
C/ LOGICAL ERR
C/ COMMON /FLAGS/ ERR
C/ TEST FOR AUTOSCALING
C/ IF(XMIN.NE.XMAX) GO TO 2
C/ XMAX=X(1,1)
C/ XMIN=XMAX
C/ DO 1 I=1,N
C/ DO 1 J=1,M
C/ XMAX=AMAX1(XMAX,X(I,J))
C/ XMIN=AMIN1(XMIN,X(I,J))
C/ 1 IF(XMAX-XMIN.GT.AMAX1(ABS(XMAX),ABS(XMIN))/10000.) RETURN
C/ 2 PRINT 3
C/ 3 FORMAT(45H *** BAD PLOT/GRAPH DATA ... MAX .LE. MIN ***)
C/ ERR=.TRUE.
C/ RETURN
C/ END
C/ LIST,NONE
C/ SUBROUTINE FIXUP(X,NXDIM,LEN,NFUNC,XMAX,XMIN,FSCALE)

```



```
FMIN=FMAX-DIFFER  
FSCALE=DIFFER/100.  
IF(FMIN.LE.XMIN) RETURN
```

500

5. DIFFER TOO SMALL, ENLARGE AND REPEAT

```
DIFFER=DIFFER*POWER  
GO TO 4  
END
```

REFERENCES

- Antoniou, A., "Digital Filters: Analysis and Design," McGraw-Hill, 1979.
- Barnes, C. W., "Roundoff Noise and Overflow in Normal Digital Filters," IEEE Trans. Circuits Syst., Vol. CAS-26, No. 3, pp. 154-159, March 1979.
- Cappellini, V., A. G. Constantinides and P. Emiliani, "Digital Filters and Their Applications," Academic Press, 1978.
- Crochiere, R. E., "Comparison of Digital Filter Structures on the Basis of Coefficient Wordlength," Mass. Inst. of Tech., Research Report PR115, 1975.
- Fettweis, A., "Digital Filter Structures Related to Classical Filter Networks," Arch. Elektron. Vebtr., Vol. 25, pp. 79-89, Feb. 1971.
- Gold, B. and C. Rader, "Digital Processing of Signals," McGraw-Hill, 1969.
- Gray, A. H. and J. D. Markel, "Digital Lattice and Ladder Filter Synthesis," IEEE Trans. Audio Electroacoust., AU-21, pp. 491-500, Dec. 1973.
- Huelsman, L. P., "Basic Circuit Theory with Digital Computations," Prentice-Hall, 1972.
- Huelsman, L. P., "Introduction to the Theory and Design of Active Filters," McGraw-Hill, 1980.
- Jackson, L. B., "On the Interaction of Roundoff Noise and Dynamic Range in Digital Filters," The Bell System Technical Journal, Vol. 49, No. 2, pp. 159-184, February 1970.
- Liu, B., "Effect of Finite Wordlength on the Accuracy of Digital Filters--A Review," IEEE Trans. Circuit Theory, Vol. CT-18, pp. 670-677, Nov. 1971.

- Mills, W. L., C. T. Mullis and R. A. Roberts, "Low Roundoff Noise and Normal Realizations of Fixed Point IIR Digital Filters," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-29, No. 4, pp. 893-903, Aug. 1981.
- Oppenheim, A. V. and R. L. Schaffer, "Digital Signal Processing," Prentice-Hall, 1975.
- Peled, A. and B. Liu, "Digital Signal Processing," Wiley, 1976.
- Shapiro, M., "PROOT--Polynomial Solver," 1965, University of Arizona, Computer Center, CDC Cyber Program Library.
- Steiglitz, K., and B. Ladendorf, "A Program for Designing Finite Word-Length IIR Digital Filters," in "Programs for Digital Signal Processing," IEEE Press, 1979.
- Taylor, F. J. and J. Marshall, "Computer Aided Design and Analysis of Standard IIR Architectures," IEEE Circuits and Systems Magazine, Vol. 3, No. 4, 1981.
- Wait, J. V., "Digital Filters," in "Active Filters: Lumped, Distributed, Integrated, Digital and Parametric," ed. L. P. Huelsman, McGraw-Hill, pp. 200-279, 1970.
- Wait, J. V., "FAD--(Frequency Analysis--Discrete), Users Manual, CSRL, Department of Electrical Engineering, University of Arizona, 1981.
- Wait, J. V., "TAD--(Time Analysis--Discrete), Users Manual, Unpublished Manuscript, 1982.