QUANTIZATION ERRORS IN DIGITAL SIGNAL PROCESSORS OF RADAR SYSTEMS

Final Technical Report

By

Jerry D. Moore
Principal Investigator

Brian P. Holt
Research Assistant

Bhadrayu J. Trivedi
Research Associate

June, 1976

Prepared for

U.S. Army Research Office, Triangle Park, North Carolina

under

Grant DAAG29-76-G-0072

THE UNIVERSITY OF ALABAMA

BER Report No. 205-125

Approved for Public Release:
Distribution Unlimited

UA

DISCLAIMER

THE FINDINGS OF THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

QUANTIZATION ERRORS IN DIGITAL SIGNAL PROCESSORS OF RADAR SYSTEMS.

5. TYPE OF REPORT & PERIOD COVERED

FINAL REPORT.
1 Jan 1976 — 30 June 1976

6. PERFORMING ORG. REPORT NUMBER

205-125

7. AUTHOR(s)

Jerry D. Moore, Brian P. Holt, and Bhadrayu J. Trivedi

8. CONTRACT OR GRANT NUMBER(s)

DAAG29-76-G-0072

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Bureau of Engineering Research
P. O. Box 1968
University of Alabama University, Alabama 35486

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE

June 1976

13. NUMBER OF PAGES

253

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

NA

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

DDC

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Signal Processing, Radar Systems, Moving Target Indicators, Quantization Noise, Quantization Errors, Fixed-Point Signal Processors, Floating-Point Signal Processors, Processor Simulations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Studies were performed in three aspects of radar system digital signal processors, viz., 1) theoretical analysis of the quantization errors in fixed-point and floating-point realizations of a quadrature channel processor, 2) simulation analysis of the processors, and 3) floating-point hardware considerations. These efforts included the effects of 1) the quantization errors of the quadrature channel analog-to-digital converters, 2) the finite word length coefficients used in the moving target indicator, 3) the vector magnitude

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

Continuation of Block 20. ABSTRACT

approximation involved in the residue calculation, and 4) the truncation of products and/or sums. Numerous graphical presentations are given for the average value and variance of the error at the processor output. General conclusions include the following: 1) the vector magnitude algorithm is a predominate source of error and causes a strong dependence on the input doppler signal amplitude, 2) clutter signals do not substantially change the error statistics, 3) processor word lengths can be chosen to minimize hardware while simultaneously meeting specified error limits, 4) the error performance of a floating-point processor is less regular than the fixed-point processor, e.g., the performance curves show peaks and valleys as a function of the signal amplitude whereas the fixed-point curves are smooth, 5) the comparison of the two processor performances depends on many factors, but they generally produce the same order-of-magnitude error statistics, 6) the floating-point processor requires more hardware, and 7) the theoretical bounds have trends similar to the simulation results, but do not bound them tightly.

FOREWORD

The authors appreciate the support of this research by the Department of the Army, U.S. Army Research Office, Durham, North Carolina under Grant DAAG29-76-G-0072. Technical discussions with Messrs. N.B. Lawrence and L.B. Owens and Dr. D.W. Burlage of the Radar Technology Branch, U.S. Army Missile Command, Redstone Arsenal have rendered valuable assistance to this effort. Many of the computer runs, data collections and presentations, and schematic diagrams were prepared by Messrs. James Blackburn and Mark Letson while undergraduate students in Electrical Engineering at The University of Alabama. Their efforts have formed a valuable contribution to this report. The manuscript was typed by Ms. Sandra Lawless and her patience, diligence and skilful work are very much appreciated.

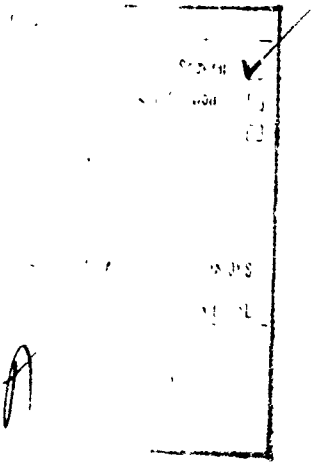


TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 DESCRIPTION OF PROBLEM

Digital signal processors (DSP) are being used to perform the moving-target-indicator (MTI) function in radar systems [1]. There are many advantages obtained by using the DSP approach, e.g., 1) increased flexibility in meeting a specific requirement through the use of adaptive and time-varying system structure, 2) decreased maintenance and fine tuning requirements of analog systems, etc. However, the finite length of data words used in a DSP system produce errors that are not present in an infinite precision case. These quantization errors can be categorized as follows:

1) Analog-to-digital quantization of input signal.
2) Quantization of processor parameters, such as coefficient values, due to finite word length constraint.
3) Quantization of arithmetic operations within the processor.

It would appear that the output quantization error is a monotonically decreasing function for increasing length of the various finite representations used in the processor. There are exceptions to this intuitive rule as shown later in this report. The hardware complexity is an increasing function for increasing word length. The design objective is to produce a DSP with minimum hardware complexity which will give acceptable system error performance. It is not necessary to minimize the quantization error, but it is desirable to keep this error below the other errors present in the system. Random noise is inherently associated with the input signal and this produces a corresponding random error at the DSP output. The performance of the system is not adversely affected by the random quantization errors which are less than the errors induced by the input random noise.

Two methods of implementing the DSP are: 1) performing the arithmetic operations with fixed-point numbers and 2) floating-point numbers. There are various advantages and disadvantages of these two approaches. It is generally recognized that a floating-point structure gives a larger dynamic range, but requires more hardware. Detailed analyses of these methods are presented in this report for a quadrature channel radar system. The specific structure of the radar system DSP considered in this work is different from digital filter systems analyzed in the literature. This uniqueness has made it extremely difficult to apply the results obtained by other authors and reported in the open literature. However, these previous efforts have established analytical procedures that are utilized in this work.

1.2 PREVIOUS RESULTS

Two categories of previous results are considered. First, statistical models associated with the error introduced by roundoff and truncation of two's complement binary numbers are reviewed. Second, the analyses of quantization errors in digital filters are considered for their application to the radar DSP.

Gold and Rader [2] have set the example of statistical models to be used for roundoff, truncation and sign-magnitude truncation. They have also considered the effects of inexact values of filter parameters, A/D conversion quantization and product quantization. The main point of their work is to show that certain recursive digital filter forms are less sensitive to quantization error than are other forms. The results obtained are not applicable to the fixed-window nonrecursive digital filter used in the radar moving-target-indicator.

Oppenheim and Schafer [3] have treated roundoff and truncation for sign magnitude, two's-complement and one's complement number systems for both fixed-point and floating-point configurations. They analyze both infinite impulse response (IIR) and finite impulse response filters. The signal flow graph approach used on pages 439 through 441 is very useful for the work documented in this report. However, the results do not apply to a fixed-window MTI.

Chapter 5 of Rabiner and Gold [4] is similar to the Oppenheim and Schafer presentation but has a slightly different emphasis. This same comment applies to Oppenheim and Weinstein [5].

The works of Sandberg [6] and Liu and Kaneko [7] are primarily devoted to floating-point realizations of recursive filters. The flow graph presentation has been adapted for use in this report.

All of the above referenced literature share the disadvantage that they do not directly apply to the fixed-window MTI structure used in the radar signal processor. Also, the quadrature channel structure, which requires that $\sqrt{I^2 + Q^2}$ be determined, has not been included in the results cited in the literature. The results indirectly apply to digital filters but not to the entire radar digital signal processor. An analysis of a fixed-point DSP was made on the LRCP for the U.S. Army in the summer of 1975. The report [8] submitted on this task has outlined the basic analytical and simulation procedures that were continued on this grant.

1.3 OUTLINE OF RESEARCH TASKS AND REPORT

This project was organized into three areas, viz., 1) Theoretical analyses of fixed-point and floating-point DSP, 2) Simulation program development and analysis of fixed-point and floating-point DSP and 3) Floating-point hardware design considerations. Each of these areas was directed at a quadrature channel radar signal processor with a square-root-of-the-sum-of-the-squares unit and a post residue integrator. The exact configuration will be presented later in this report. The areas were divided into tasks as follows. Moore refined the LRCP analysis [8] for a fixed-point processor as presented in Chapter 2 while Trivedi worked with the fixed-point simulation program documented in Chapter 3.

Holt studied floating-point arithmetic systems and applied these to the design of a floating-point DSP as reported in Chapter 4. Moore followed Holt's effort with a theoretical analysis of the floating-point DSP in Chapter 5 while Holt and Trivedi developed the floating-point simulation program of Chapter 6.

CHAPTER 2

THEORETICAL ANALYSIS OF FIXED-POINT PROCESSOR
by Jerry D. Moore

2.1 DESCRIPTION OF PROCESSOR

There are many configurations that could be used for a digital signal processor in a radar system. This effort is limited to pulsed doppler radars that use quadrature channels in the receiver to extract the moving target information from the noise and clutter signals. A schematic representation of the signal processor is shown in Figure 2.1 and the details of the digital filter configuration are shown in Figure 2.2. This system is capable of processing multiple range bins (e.g., 1000) by utilizing one A/D converter per channel and one digital multiplier per channel. Typical pulse repetition rates of 5000 pulses/sec requires A/D sample rates of 5×10^6 samples/sec to accommodate the 1000 range bins. The filter of Figure 2.2 is a fixed window configuration, i.e., N samples are used in a fixed block size to determine a residue output. While this results in a signal-to-noise ratio loss as compared to the moving-window approach [9,10] it does give a much simpler hardware realization.

The residue calculation indicated by Figure 2.1 is ideally given by

$$\text{Residue} = \sqrt{I^2 + Q^2}\ , \tag{2.1}$$

where I and Q are the outputs of the two quadrature channels. In practice, the square root operation is difficult to implement and various approximations are utilized [1], such as

$$\text{Residue} \approx \begin{cases} L + 3S/16 & 0 \le S/L \le 0.5 \\ 3L/4 + 11S/16 & 0.5 < S/L \le 1.0 \end{cases} \tag{2.2}$$

where $L = \max\{|I|,|Q|\}$ and $S = \min\{|I|,|Q|\}$. This two-sector approximation gives a peak error of -2.16%, an RMS error of 12.65%, and an average error of 0.69% when averaged on all phase angles between I and Q. As will be shown, this approximation is a major contributor to the DSP errors.

The finite word lengths used in the DSP are shown in Figure 2.3 and summarized in Table 2-1. The nomenclature presented here is used in the following section to theoretically analyze the fixed-point DSP.

2.2 OUTLINE OF ANALYSIS

The theoretical analysis of the DSP quantization error is presented in this section. Many symbols, equations, and bounding procedures are necessary and it is difficult to follow the main development when presented in its entire detail. Consequently, a summary of the analysis is presented

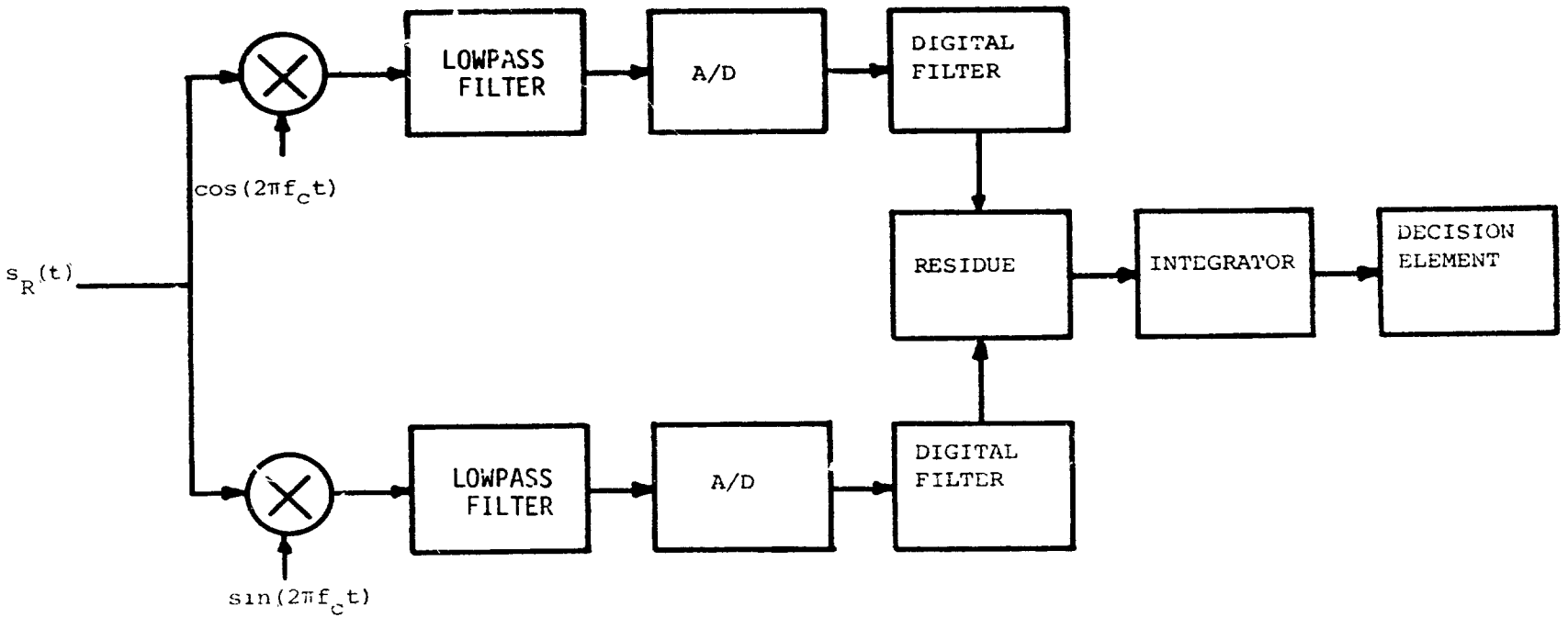


Fig. 2.1 Quadrature Channel Digital Signal Processor Configuration

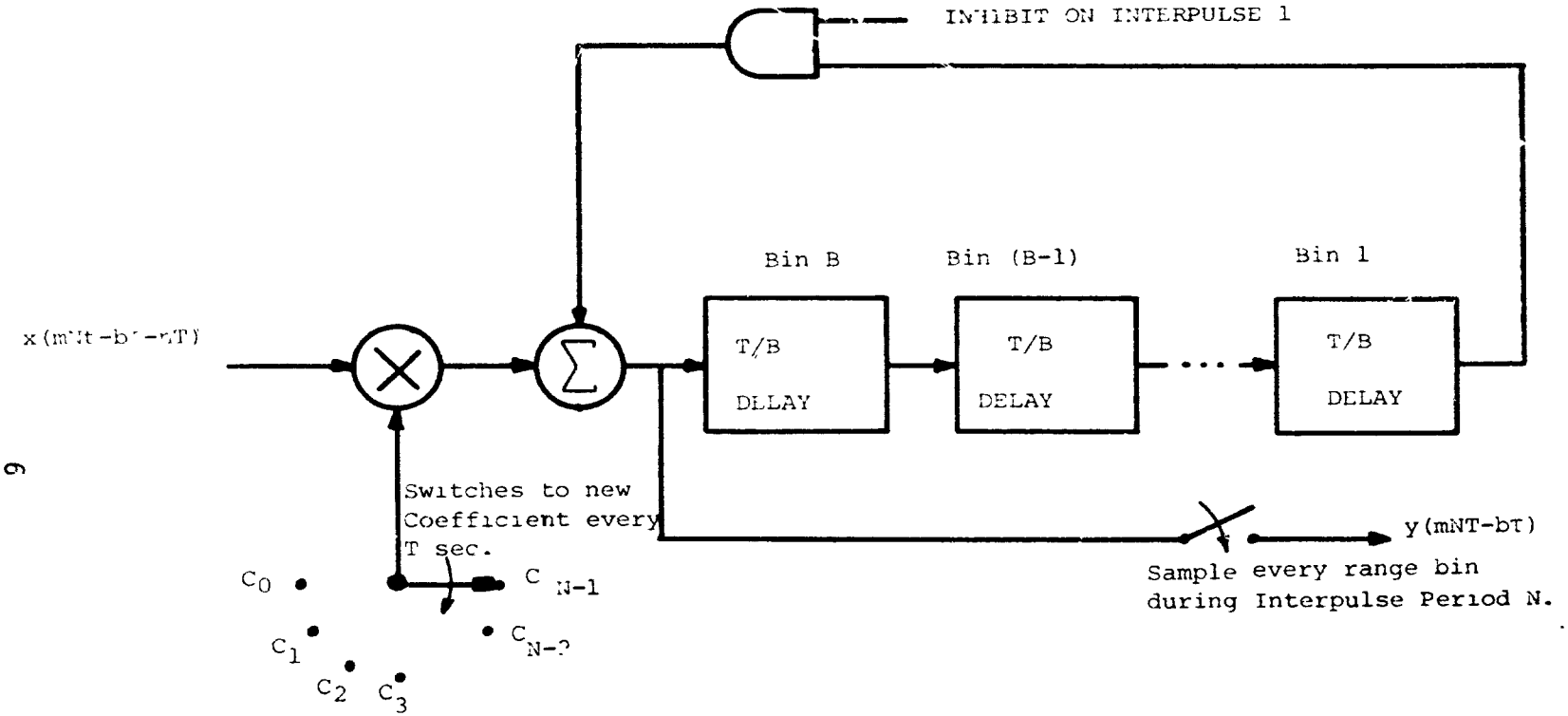
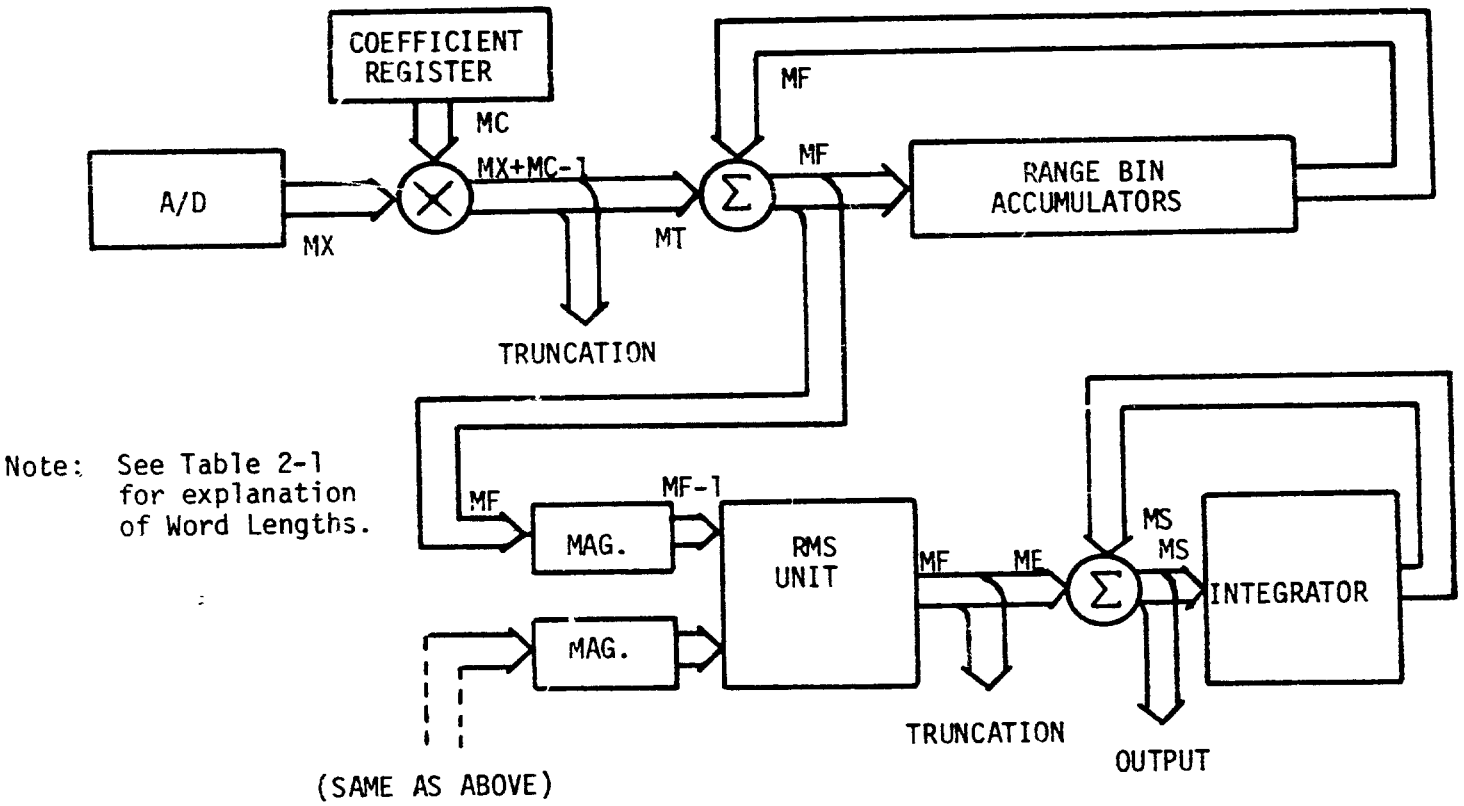


Fig. 2.2 A Fixed-Window Digital Filter for Multiple Range Bins



Note: See Table 2-1 for explanation of Word Lengths.

Fig. 2.3 Fixed-Point Digital Signal Processor with Finite Word Lengths Indicated

TABLE 2-1

FIXED-POINT DSP WORD LENGTH

Symbol	Description
MX	A/D Converter word length. Two's complement form with 1 sign bit and MX - 1 fractional bits.
MC	Coefficient word length. Two's complement form with 1 sign bit and MC - 1 fractional bits.
MX + MC - 1	Product word length. Two's complement form with 1 sign bit and MX + MC - 2 fractional bits.
MT	Truncated product word length. Two's complement form with 1 sign bit and MT - 1 fractional bits.
MF	Range bin accumulator word length. Two's complement form with 1 sign bit, MT - 1 fractional bits and MF - MT integer bits.
ME	Truncated residue word length. Magnitude form with MF - MI + 1 integer bits and ME - MF + MT - 1 fractional bits. No sign bit is used.
MS	Integrator accumulator word length. Magnitude form with MS - ME + MF - MT + 1 integer bits and ME - MF + MT - 1 fractional bits. No sign bit is used.

here and detailed derivations, etc., are left to an appendix.

The binary word outputs from the A/D converter are written as the sum of an errorless term x() and the quantization error e(). The output of the digital filter w() is used every N samples (where N is the number of filter coefficients) and written as a sum of an errorless output y() and the quantization error g(), i.e.,

$$w(mN) = y(mN) + g(mN) \quad , \tag{2.3}$$

where

$$y(mN) = \sum_{n=0}^{N-1} h(n)x(mN-n) \quad , \tag{2.4}$$

and

$$g(mN) = \sum_{n=0}^{N-1} \left[h(n)e(mN-n) + e_n(mN-n) \right] \quad , \tag{2.5}$$

and h(n) represents the filter coefficients. The term $e_n()$ introduced in (2.5) is the representation for the truncation of the n^{th} product used in forming the mN^{th} filter output. The expressions of (2.3), (2.4) and (2.5) are valid for either the I or Q channel by adding the appropriate subscript.

The output of the RMS approximation unit, r() is represented as the sum of errorless output and an error term $e_r()$, i.e.,

$$r(mN) = \sqrt{w_I^2(mN) + w_Q^2(mN)} + e_r(mN) \quad . \tag{2.6}$$

The error term is expressed as a random variable term, $\gamma_w()$ times the errorless output since the error is a percentage of the perfect value. Thus

$$r(mN) = [1 + \gamma_w(mN)] \sqrt{w_I^2(mN) + w_Q^2(mN)} \quad . \tag{2.7}$$

This approximation to the residue is then truncated prior to the integrator, i.e.,

$$b(mN) = r(mN) + e_t(mN) \quad , \tag{2.8}$$

where b() is the input to the integrator and $e_t()$ is the truncation error associated with the residue.

An expression for the integrator output INT(M) is given by the sum of M residues. Taking into account the finite arithmetic effects yields

$$INT(M) = \sum_{m=1}^{M} b(mN) \ . \tag{2.9}$$

An infinite precision system would have an integrator output that depends on the input signal frequency, f, and amplitude, A, and the magnitude of the transfer function associated with the filter, $|H(f)|$. A perfect residue is

$$u(mN) = u = |H(f)| \cdot A \ , \tag{2.10}$$

and the errorless integrator output is

$$INT(M)|_{errorless} = M \cdot u \ . \tag{2.11}$$

The actual integrator output is expressed as the sum of this errorless term and the error term INTE(M), viz.,

$$INT(M) = M \cdot u + INTE(M) \ . \tag{2.12}$$

The primary goal of this analysis is to study the error term INTE(M). This will be accomplished by evaluating the average value and variance of the integrator error. Using the previous results and assuming statistical independence of the error contributions (see details in Appendix A) gives for the average error,

$$\overline{INTE(M)} = M[\bar{r} + \bar{e}_t - u] \ , \tag{2.13}$$

and for the variance

$$\sigma^2_{INTE} = M[\sigma_r{}^2 + \sigma_t{}^2] \ . \tag{2.14}$$

The mean and variance of the truncation term, i.e., $\bar{e}_t$ and σt^2 can be calculated precisely, but the corresponding parameters for r() are not easily obtained. It was necessary to use bounding procedures as presented in Appendix A. A condensed version of the bounding method is obtained by noting r() given by (2.7) is proportional to the vector magnitude of the sum of two vectors U and V, where $U = (y_I, y_Q)$ and $V = (g_I, g_Q)$. If u

and v are the respective magnitudes of these vectors, then it follows that

$$(1 + \gamma_w)\,|u - v| \le r \le (1 + \gamma_w)(u + v)\,. \tag{2.15}$$

The mN arguments have been omitted for simplicity.
The u term can be evaluated as in (2.10) and

$$v(mN) = \sqrt{g_I^2(mN) + g_Q^2(mN)}\,. \tag{2.16}$$

The expected value of r from (2.15) can be expressed with the aid of the Concave/Convex Theorem of Appendix A as

$$(1 + \bar{\gamma}_w)\,|u - \bar{v}| \le (1 + \bar{\gamma}_w)\,\overline{|u - v|} \le \bar{r} \le (1 + \bar{\gamma}_w)(u + \bar{v})\,, \tag{2.17}$$

where

$$\sqrt{2}\,|\bar{g}| \le \bar{v} \le \sqrt{2\,\bar{g^2}}\,. \tag{2.18}$$

It follows from (2.13), (2.17) and (2.18) that

$$\overline{INTE(M)} \le M\left[\,\bar{\gamma}_w u + (1 + \bar{\gamma}_w)\sqrt{2\,\bar{g^2}} + \bar{e}_t\,\right], \tag{2.19}$$

and

$$\overline{INTE(M)} \ge M\left[(1 + \bar{\gamma}_w)\sqrt{u^2 + 2\cdot\bar{g}^2 - 2\cdot\sqrt{2}\cdot u\cdot\sqrt{\bar{g^2}}} + \bar{e}_t - u\right]. \tag{2.20}$$

It is possible to start with (2.15) and use (2.17) to evaluate σ_r^2, i.e.,

$$\sigma_r^2 = \bar{r^2} - \bar{r}^2\,. \tag{2.21}$$

After considerable manipulation the variance term is bounded as follows,

$$\sigma_r^2 \leq 2(1 + 2\bar{\gamma}_w)\sigma_g^2 + \sigma_\gamma^2 u^2 + 2\sqrt{2\,\overline{g^2}}\,[\overline{\gamma_w^2} + \bar{\gamma}_w^2 + 4\bar{\gamma}_w + 2]\,u$$

$$+ 2\,\overline{\gamma_w^2}\,\overline{g^2} - 2\,\bar{\gamma}_w^2\,\bar{g}^2 \,, \qquad (2.22)$$

and

$$\sigma_r^2 \geq (u^2 + 2\,\overline{g^2})\,\sigma_\gamma^2 - 2\sqrt{2\,\overline{g^2}}\,[\overline{\gamma_w^2} + \bar{\gamma}_w^2 + 4\,\bar{\gamma}_w + 2]\,u \,. \qquad (2.23)$$

It follows from (2.14), (2.22) and (2.23) that

$$\sigma_{INTE}^2 \leq M\,[\sigma_r^2{}_{max} + \sigma_t^2] \,, \qquad (2.24)$$

and

$$\sigma_{INTE}^2 \geq M\,[\sigma_r^2{}_{min} + \sigma_t^2] \,. \qquad (2.25)$$

The important bound results of (2.19), (2.20), (2.24) and (2.25) depend on $\bar{e}_t$, σ_t^2, $\bar{g}$ and $\overline{g^2}$. The parameters are evaluated in the following presentation. As stated in Table 2-1, the RMS unit output has MT - 1 fractional bits and MF - MT + 1 integer bits. When this word is truncated to ME bits total with ME - MF + MT - 1 fractional bits then the error e_t is within the range

$$-(2^{-ME+MF-MT+1} - 2^{-MT+1}) < e_t \leq 0 \,. \qquad (2.26)$$

Assuming a uniform distribution over this range gives

$$\bar{e}_t = -(2^{-ME+MF-MT} - 2^{-MT}) \,, \qquad (2.27)$$

$$\sigma_t^2 = \frac{2^{-2MT}\,(2^{-ME+MF} - 1)^2}{3} \,. \qquad (2.28)$$

From (2.5) it follows that

$$\bar{g} = \sum_{n=0}^{N-1} [h(n)\,\bar{e} + \bar{e}_n] , \tag{2.29}$$

but $\bar{e}$ is equal to zero because of the roundoff procedure used in the A/D converter and

$$-(2^{-MT+1} - 2^{-MX-MC+2}) < e_n \leq 0 , \tag{2.30}$$

since it is the truncated version of the product. Thus

$$\bar{e}_n = -(2^{-MT} - 2^{-MX-MC+1}) , \tag{2.31}$$

and

$$\sigma_n^2 = \frac{(2^{-MT} - 2^{-MX-MC+1})^2}{3} . \tag{2.32}$$

Using (2.31) in (2.29) yields

$$\bar{g} = -N\,(2^{-MT} - 2^{-MX-MC+1}) . \tag{2.33}$$

The variance of g is given by

$$\sigma_g^2 = N\sigma_n^2 + \sigma_e^2 \sum_{n=0}^{N-1} h^2(n) , \tag{2.34}$$

where σ_e^2 is the variance of the roundoff error e, i.e.,

$$\sigma_e^2 = \frac{2^{-2MX}}{3} . \tag{2.35}$$

The mean-squared-value of g is

$$\overline{g^2} = \sigma_g^2 + \overline{g}^2 = N\,\sigma_n^2 + \sigma_e^2 \sum_{n=0}^{N-1} h^2(n) + N^2\,\overline{e}_n^2 \quad . \qquad (2.36)$$

This concludes the analysis for the fixed-point processor. The average error at the integrator output can be upper and lower bounded by using (2.19), (2.20); the results below (2.2); (2.27), (2.29) and (2.36). The integrator output variance can be bounded by using (2.24), (2.25), (2.22), (2.23); the results below (2.2); (2.34) and (2.36). A computer program was written to calculate the values for specified word lengths, filter coefficients, etc. Appendix B presents the listing of the program. An additional feature was included into the program, viz., the capability of calculating the minimum and maximum values of the integrator output error. This program was utilized in obtaining the results presented in the next section.

2.3 GRAPHICAL PRESENTATIONS OF THEORETICAL RESULTS

The computer program of Appendix B was utilized to obtain specific values for the integrator output error statistics. The results presented in this section are for an A/D converter word length of MX = 9 bits, coefficient word length MC = 9 [see Ref. 1] and various combinations of truncated product length MT and pre-integrator word length ME. The input signal amplitude was also varied. The range bin accumulator word length MF was chosen to have the same number of fractional bits as the truncated product and 3 integer bits to avoid overflow, i.e., MF = MT + 3. In general the choice is dependent on the number of coefficients being used.

The data presented first are for an input signal amplitude of 0.025 volts with a frequency of 1500 Hz. In Fig. 2.4, the upper bound on the average output error is plotted as a function of the truncated product length MT with a family of curves dependent of the truncated residue word length ME. There are several important observations that can be made from Fig. 2.4, and from the expanded plot of Fig. 2.5.

1) For a specified member of the family of curves, i.e., given ME, then the upper bound on the error goes from a positive value to a negative value as MT is increased.
2) For a given value of ME, the error curve doesn't change very much when MT increases beyond some threshold value, ($\approx$ 11 or 12).
3) For a given value of ME, then in general the minimum magnitude of the error is not obtained by the maximum value of MT. For example, when ME = 10 the smallest magnitude error is -3.35×10^{-3} and occurs when MT = 11.
4) For a given value of MT, then the minimum magnitude of the error is not obtained by the maximum value of ME. For example, when MT = 12 the smallest magnitude error is

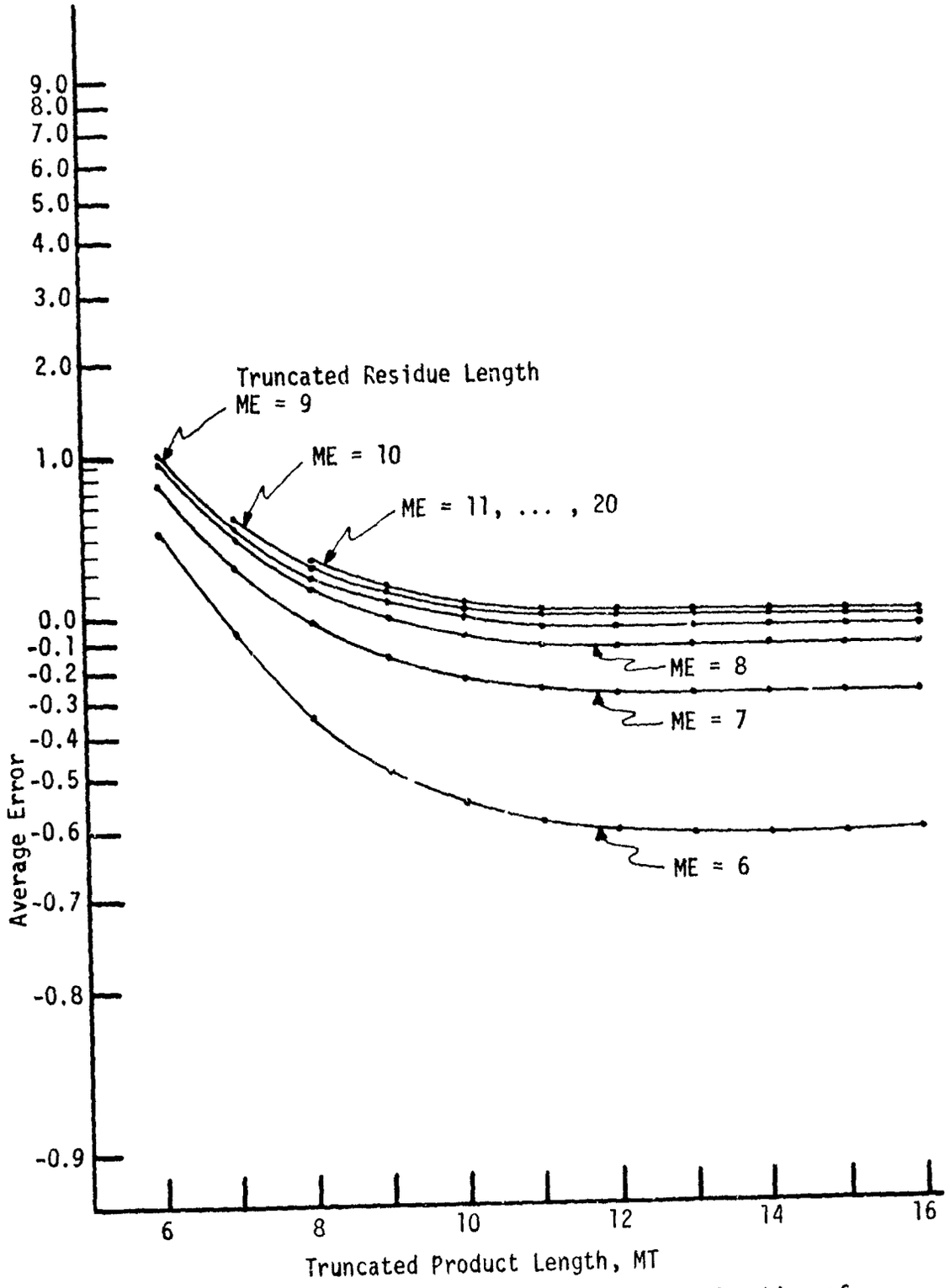


Fig. 2.4 Upper Bound on Average Integrator Error as Function of Truncated Product Length (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

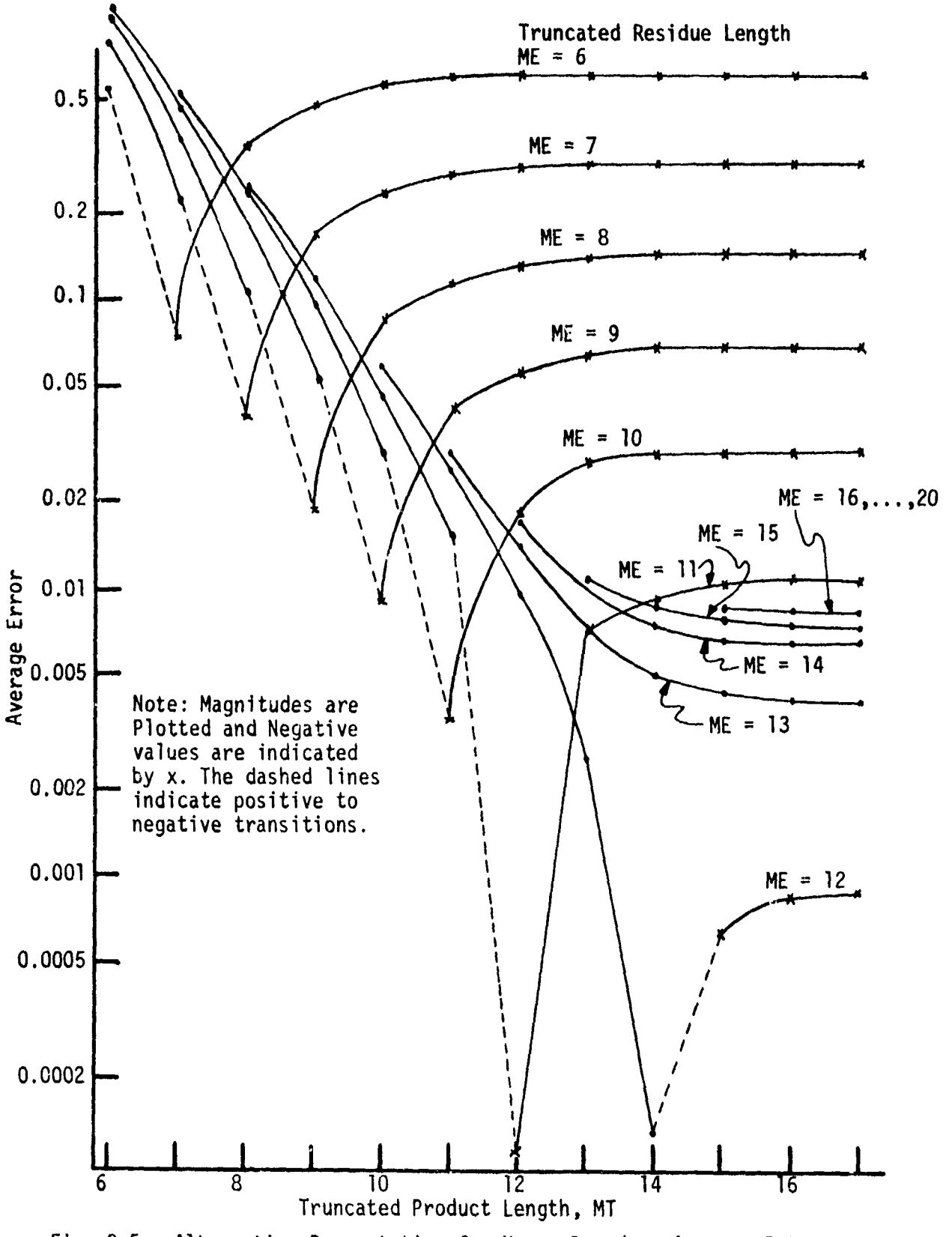


Fig. 2.5 Alternative Presentation for Upper Bound on Average Integrator Error. (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

-1×10^{-4} and occurs when ME = 11. If ME is decreased then a larger magnitude negative error is obtained. If ME is increased then a larger magnitude positive error is obtained.

5) For decreasing values of MT, then the family of curves tend to converge.

These observations are supported by the intuitive reasoning that the RMS unit converts the negative error caused by the product truncation into a positive error. The truncation of the residue will cause a negative error that will make the output error more negative. The relative contributions of these terms will determine the sign of the resultant error.

Lower bound results for 0.025 volts signal amplitude and 1500 Hz frequency are presented in Fig. 2.6 and 2.7. The same observations are noted as above, but the curves have been shifted in a negative direction.

A comparison of the upper and lower bounds reveals two observations. First, when ME is small, e.g., ME = 6, then there is strong convergence. Second,for larger ME values,e.g., ME > 10, the bounds are not tight. A typical comparison is shown in Fig. 2.8 for maximum length of ME = MT + 3. The maximum and minimum error limits are also included for comparison. Note that the middle point of the extremes is in close agreement to the middle point of the bounds.

An alternative method for analyzing the data is to present the average error bounds as a function of ME with MT being fixed. The same amplitude and frequency signal was used to obtain the results of Fig. 2.9. A value of MT = 13 was chosen as a typical value. Note the convergence of the bound curves as ME decreases. The best choice for ME depends on which bound is being considered, i.e., for the upper bound ME = 11 or 12 would be chosen while for the lower bound any ME > 12 would be acceptable.

Figures 2.4 through 2.9 have been concerned with the average error bounds. The integrator output error variance bounds are presented in Fig. 2.10 as a function of MT with ME as a family of curves. The lower bound is zero for values of ME and MT not plotted. The following observations are made:

1) The upper bound curves converge for decreasing values of MT.
2) The bound curves level off for increasing values of MT.
3) The upper and lower bound curves converge for ME = 6 or 7 as MT increases.
4) The variance decreases monotonically as ME increases, but the decrease is small for ME > 10.

The curves of Fig. 2.11A and B present the variance bounds as a function of ME with MT as the family parameter. Similar conclusions are drawn.

The second set of data presented are for a signal amplitude of 0.413 volts and 1500 Hz. The average error upper bounds are presented in Fig. 2.12 through Fig. 2.15 while the variance is presented in Fig. 2.16 through Fig. 2.18. Comparing Fig. 2.12 to Fig. 2 4 reveals a strong similarity for low values of ME. It is difficult to compare the results

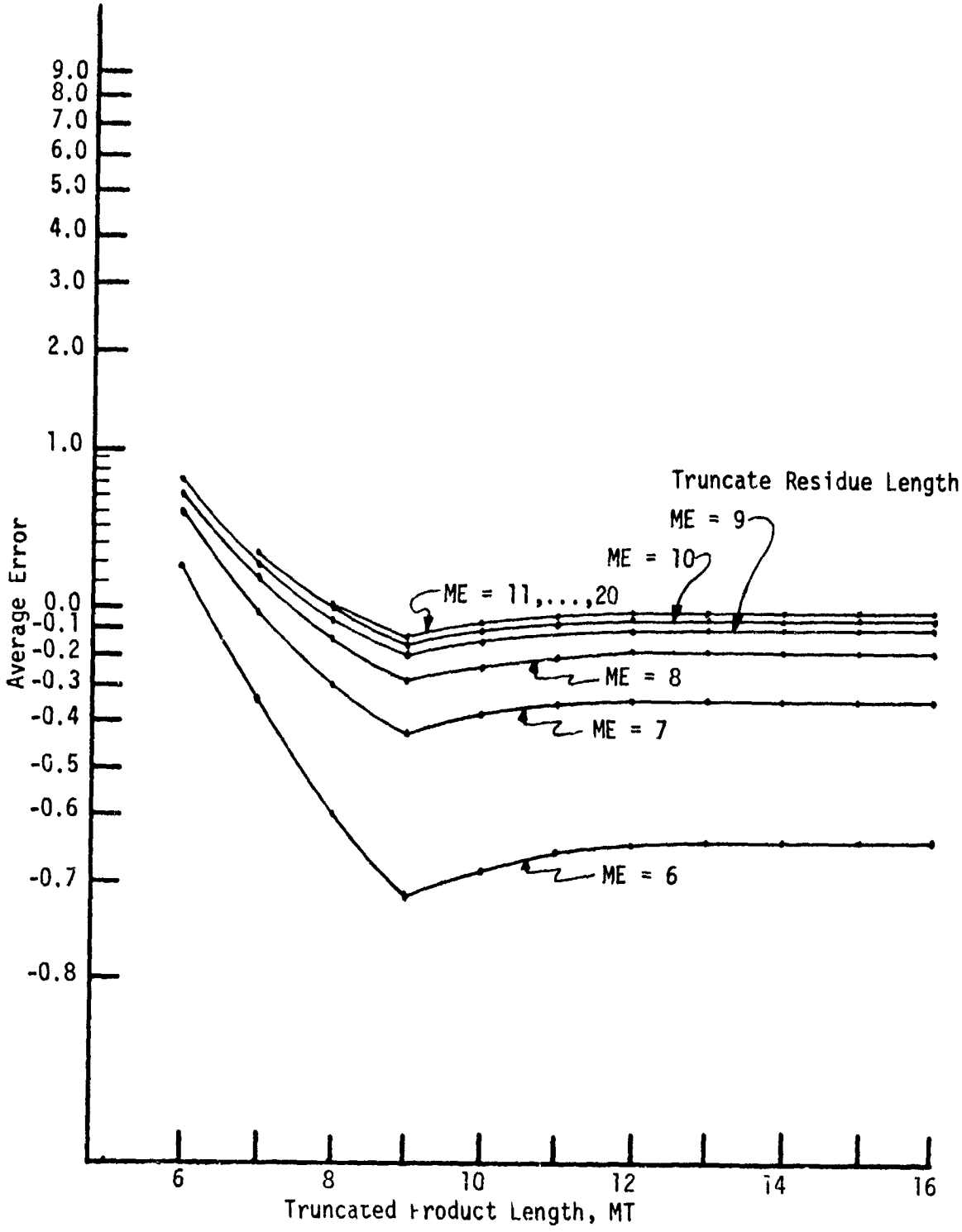


Fig. 2.6 Lower Bound on Average Integrator Error as Function of Truncated Product Length (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

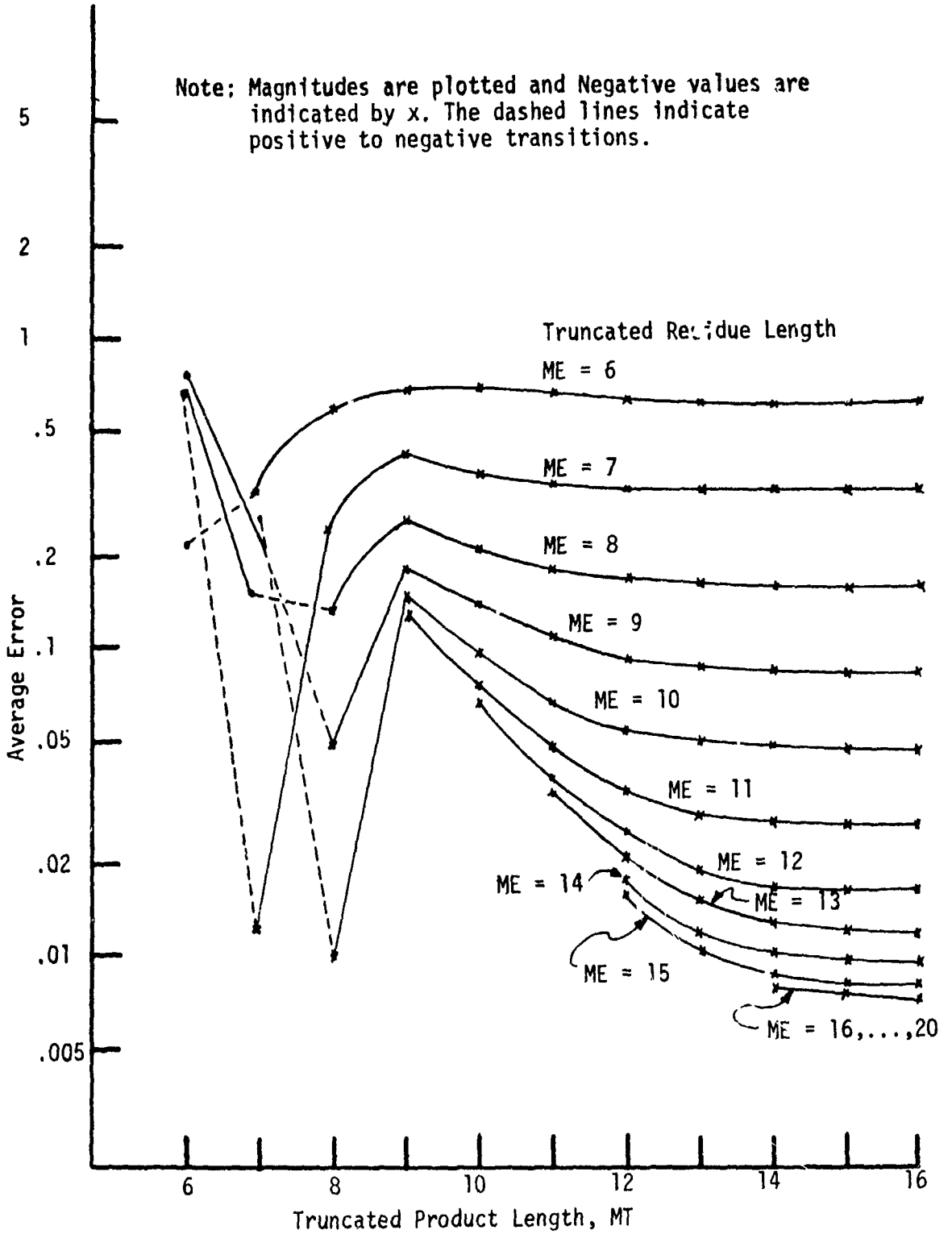


Fig. 2.7 Alternative Presentation of Lower Bound on Average Integrator Error (Signal Amplitudes = 0.025V, Frequency = 1500 Hz)

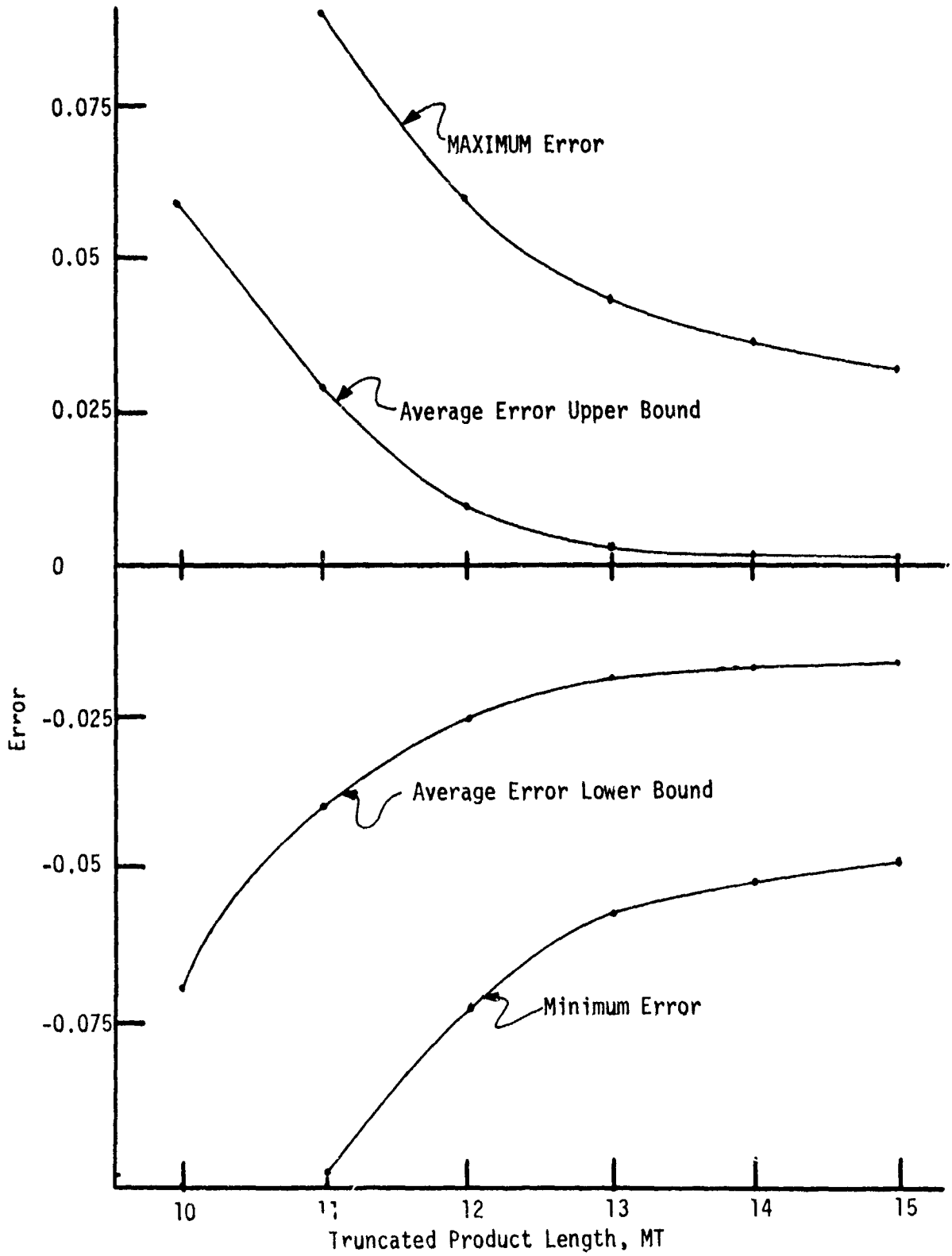


Fig. 2.8 Average Integrator Error Bounds Comparison as Function of Truncated Product Length with Residue Length of ME=MT+3 (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

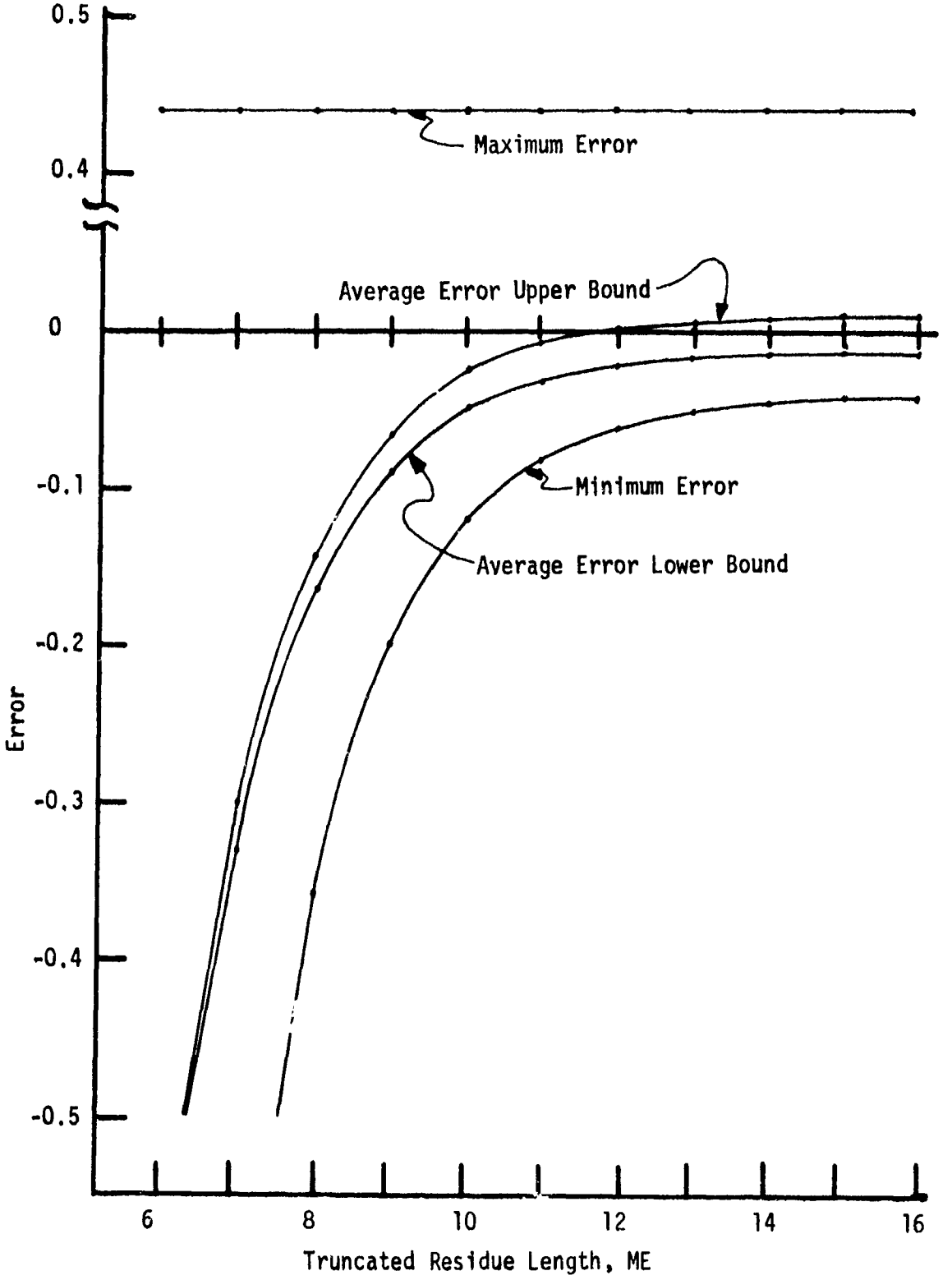


Fig. 2.9 Average Integrator Error Bounds Comparison as Function of Truncated Residue Length with Truncated Product Length MT=13 (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

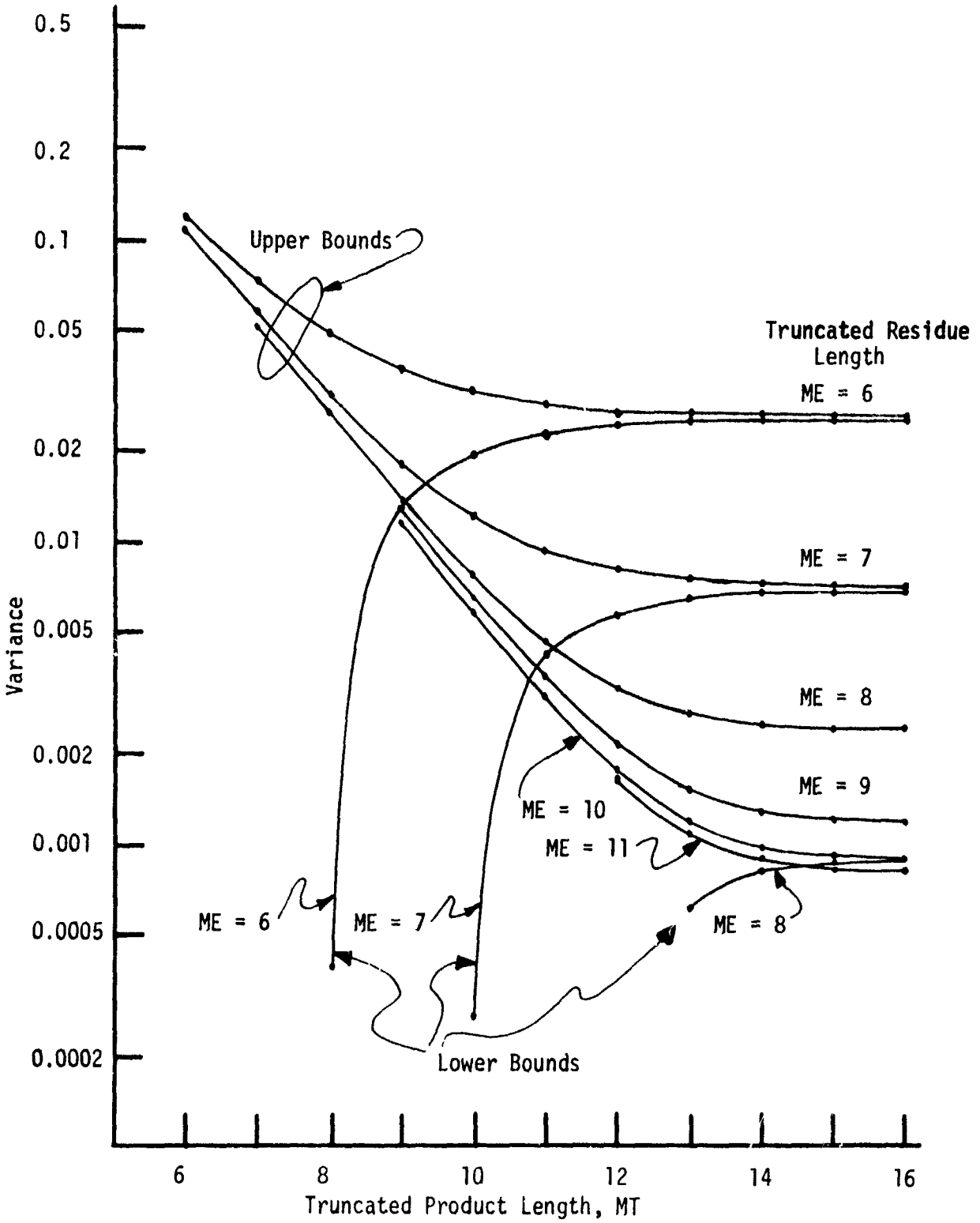


Fig. 2.10 Bounds on Integrator Error Variance as Function of Truncated Product Length (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

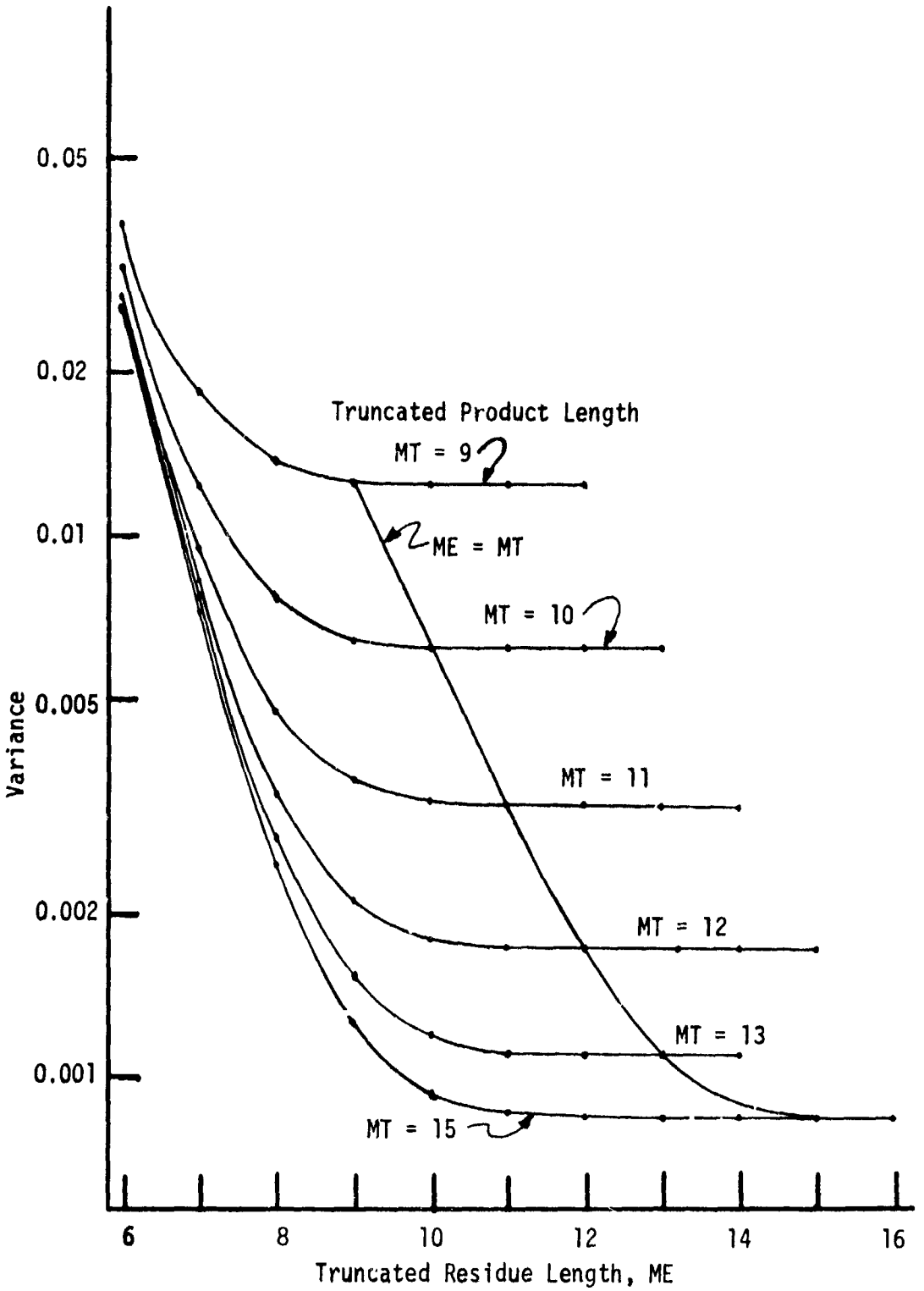


Fig. 2.11A Upper Bound on Integrator Error Variance as Function of Truncated Residue Length (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

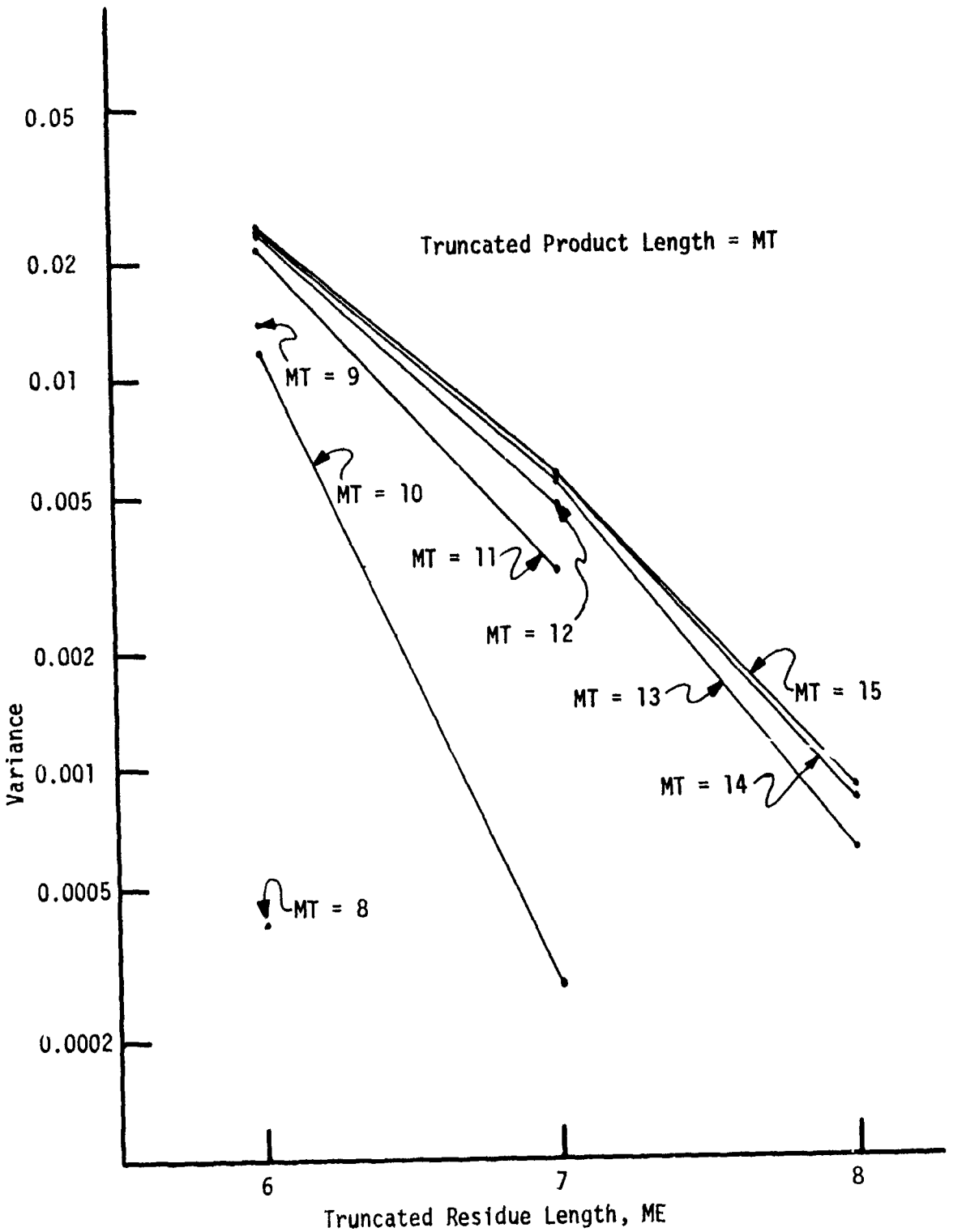


Fig. 2.11B Lower Bound on Integrator Error Variance as Function of Truncated Residue Length (Signal Amplitude = 0.025V, Frequency = 1500 Hz)

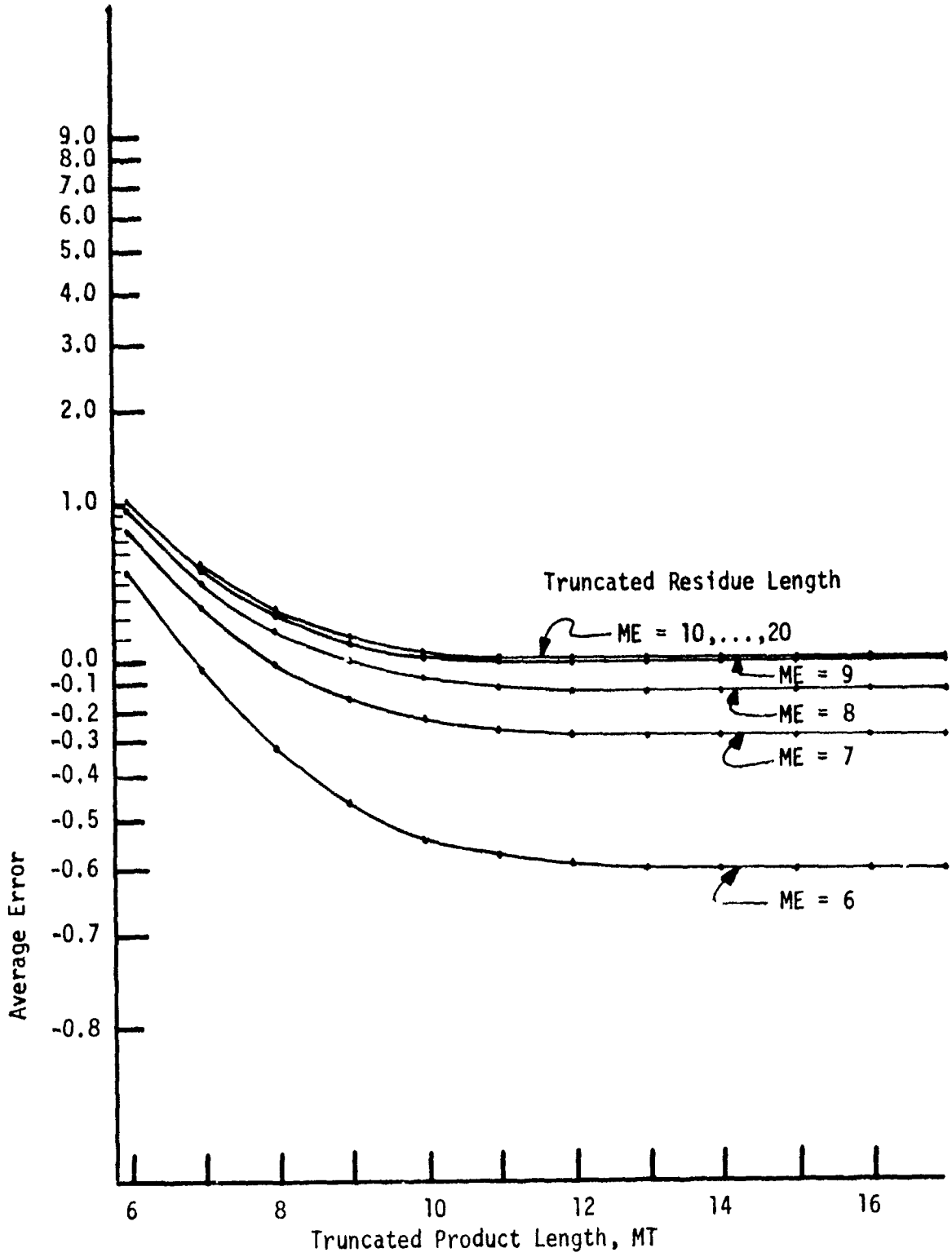


Fig. 2.12 Upper Bound on Average Integrator Error as Function of Truncated Product Length (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

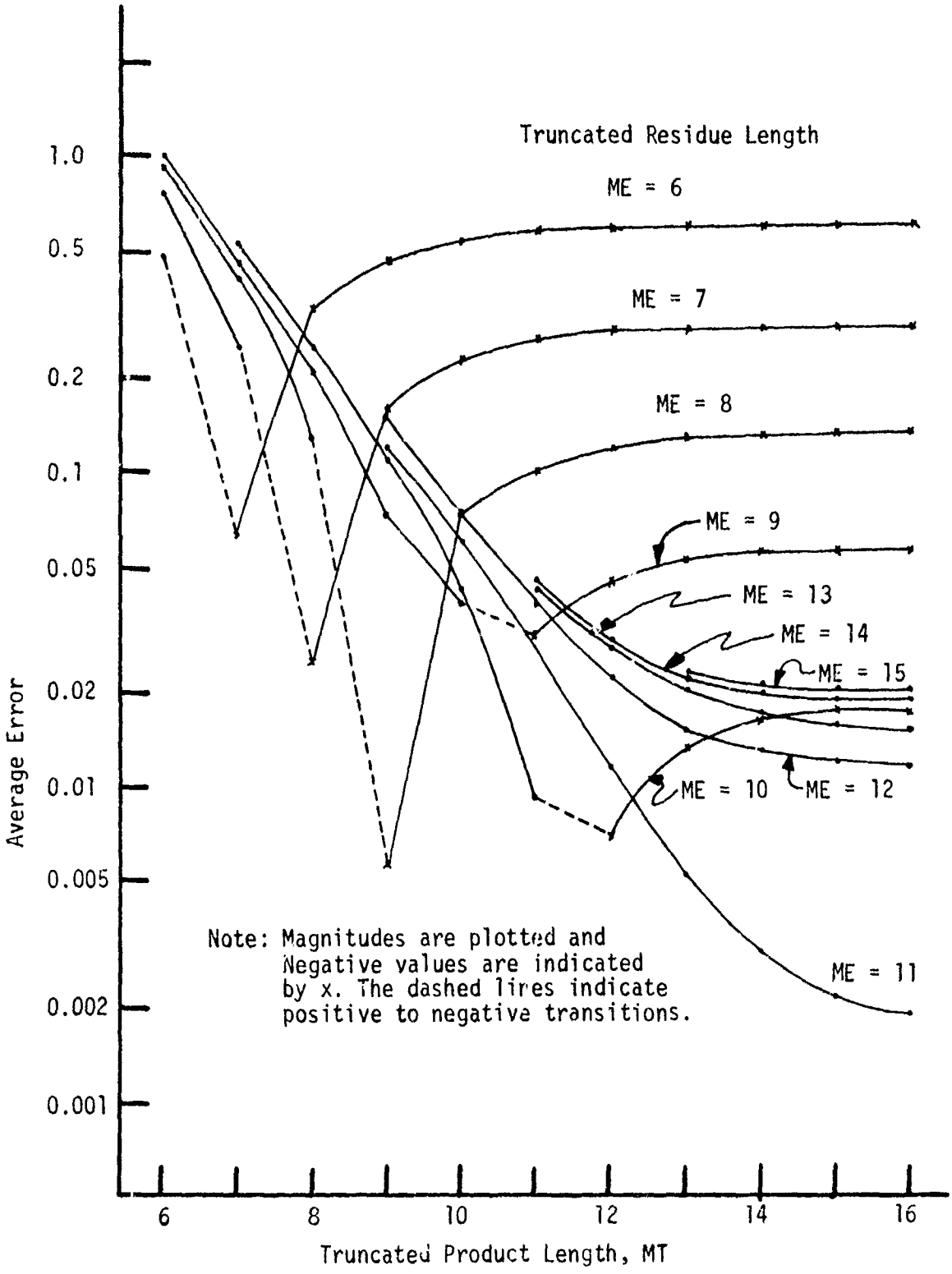


Fig. 2.13 Alternative Presentation for Upper Bound on Average Integrator Error (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

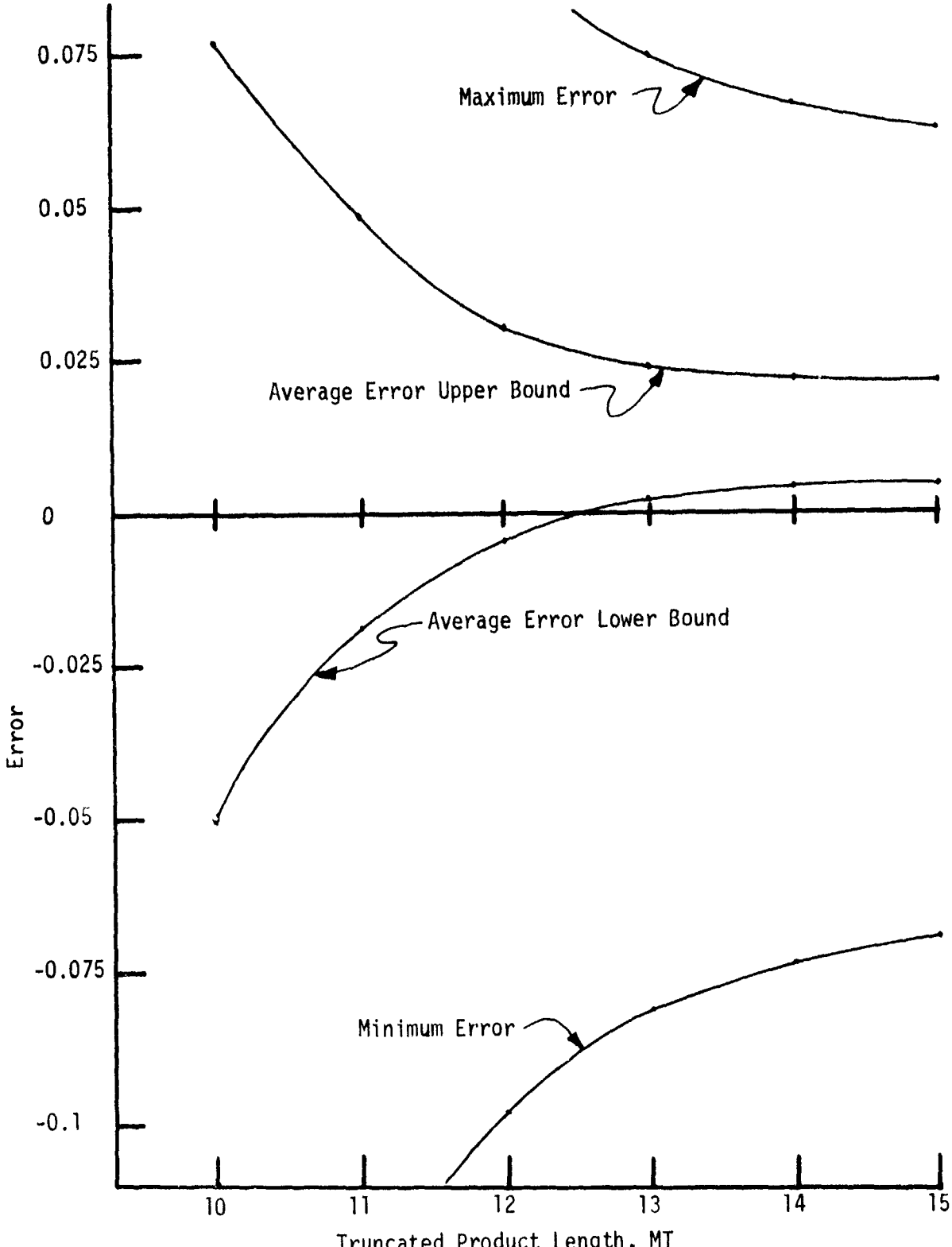


Fig. 2.14 Average Integrator Error Bounds Comparison as Function of Truncated Product Length with Residue Length of ME=MT+3 (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

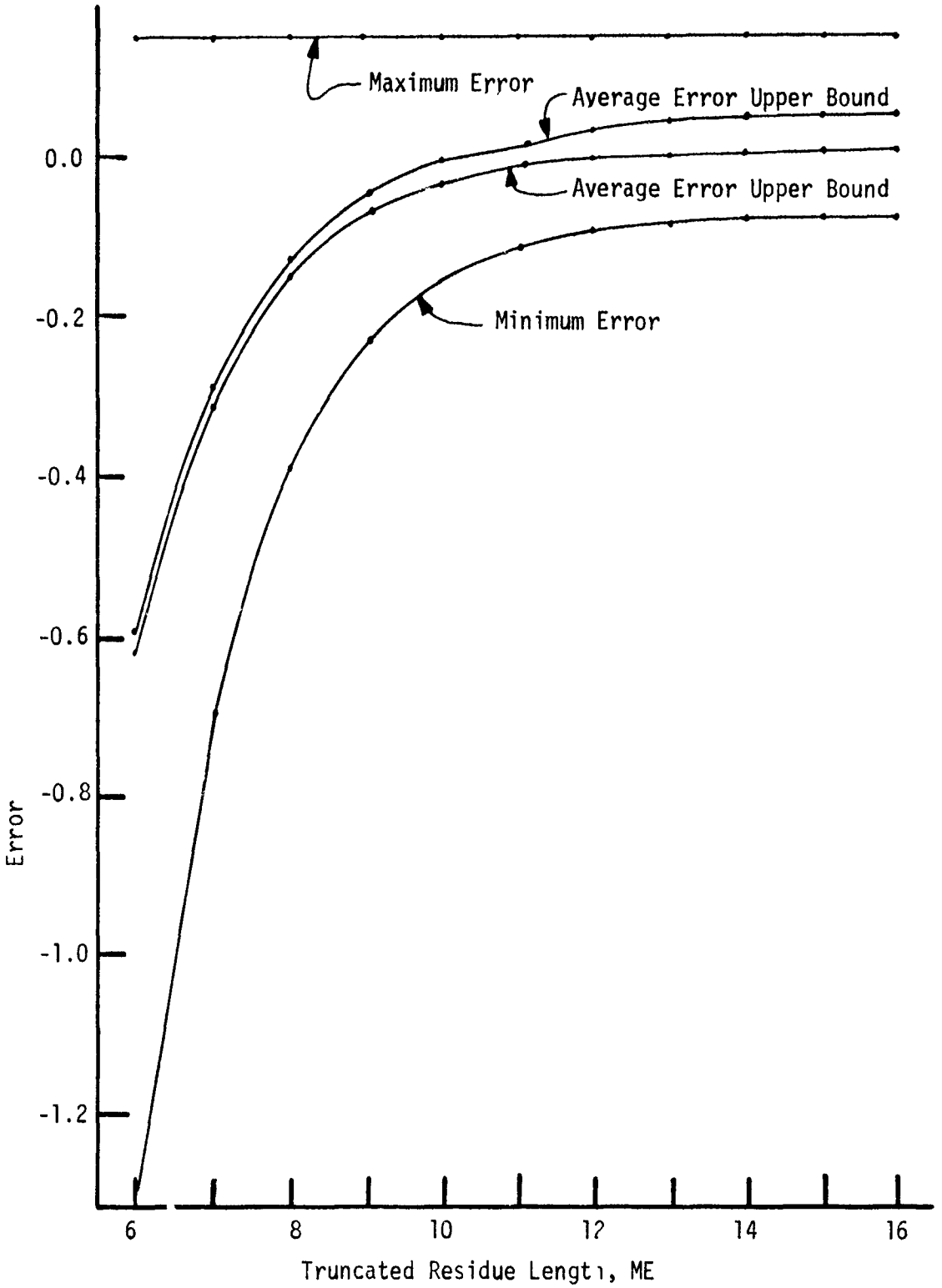


Fig. 2.15 Average Integrator Error Bounds Comparison as Function of Truncated Residue Length with Truncated Product Length MT = 13 (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

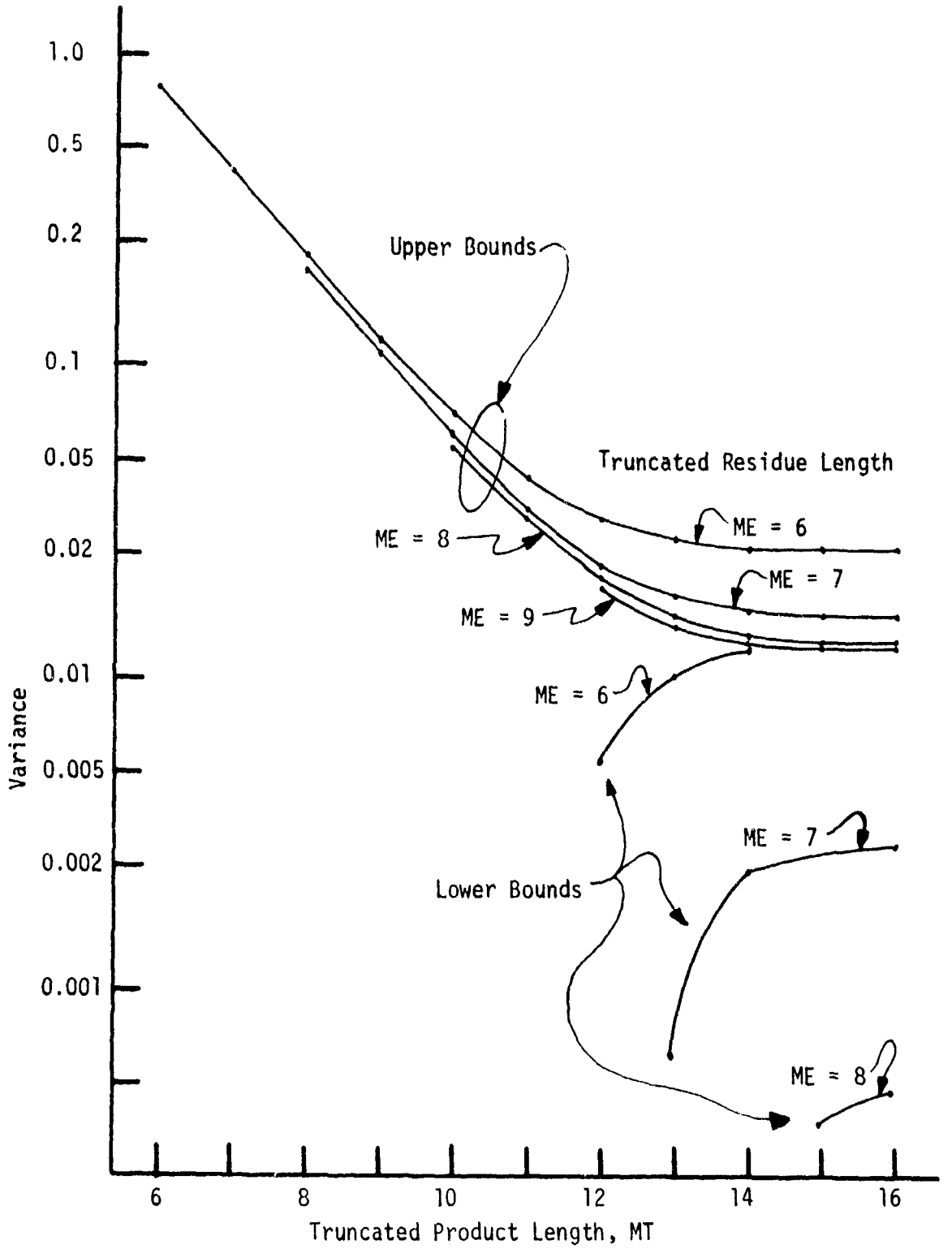


Fig. 2.16 Bounds on Integrator Error Variance as Function of Truncated Product Length (Signal Amplitude = 0.413V, Frequency = 1500 HZ)

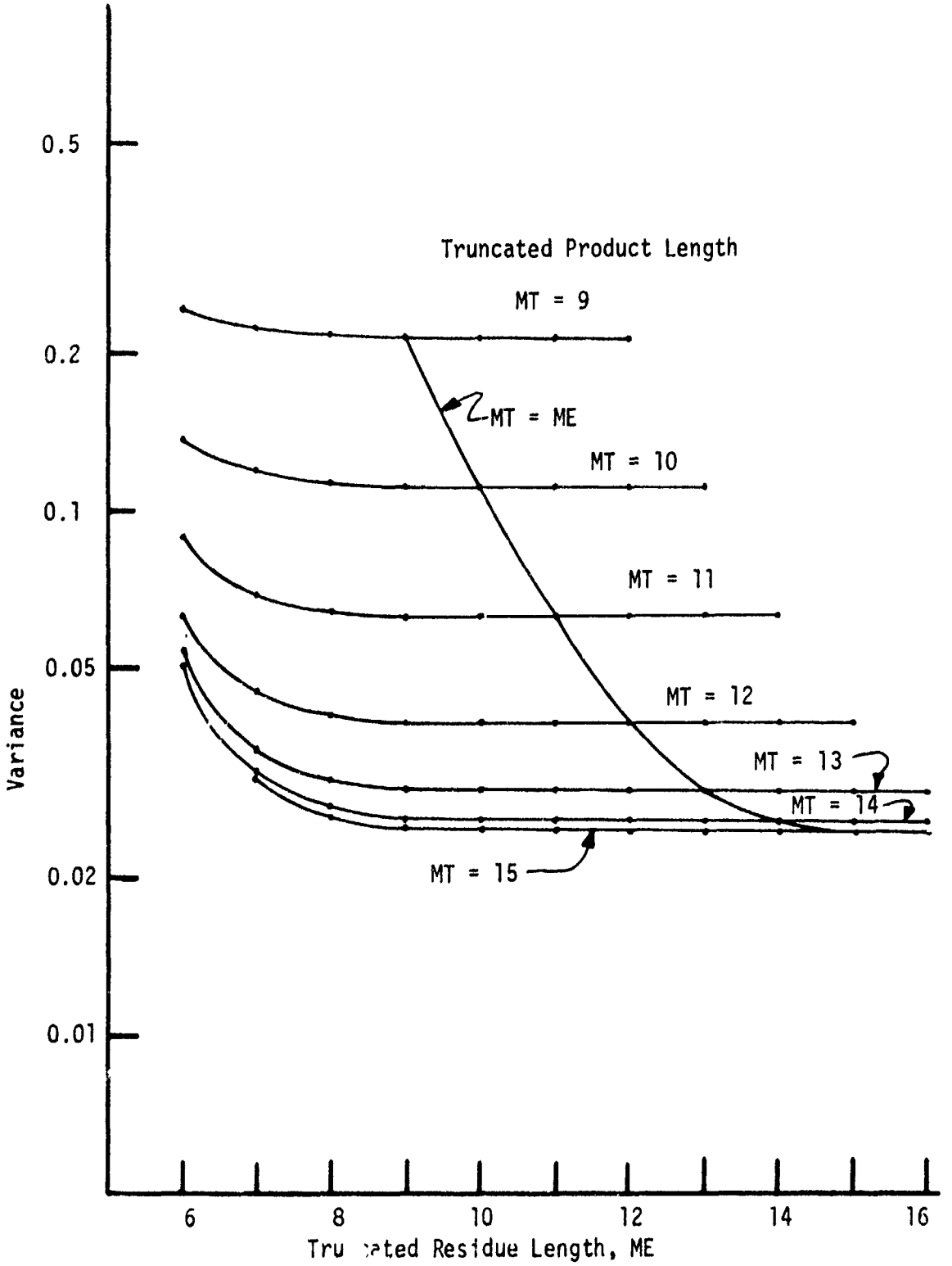


Fig. 2.17 Upper Bound on Integrator Error Variance as Function of Truncated Residue Length (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

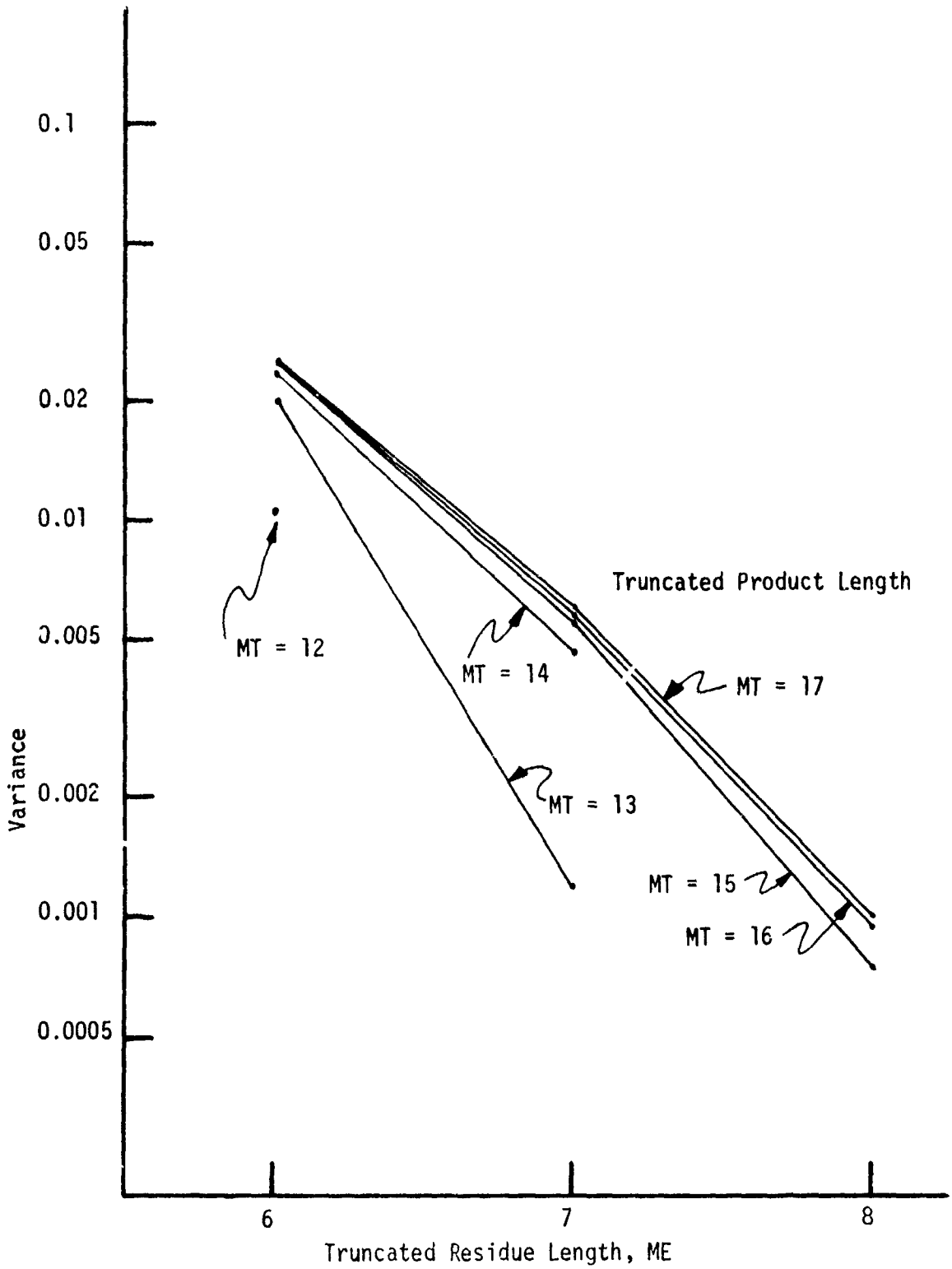


Fig. 2.18 Lower Bound on Integrator Error Variance as Function of Truncated Residue Length (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

for larger values of ME by using these curves. However, a comparison of Fig. 2.13 to Fig. 2.5 reveals that when the truncation error introduced by ME becomes small, then the error caused by other sources tends to dominate. The larger signal amplitude used in Fig. 2.13 will have a larger error contribution due to the 0.69% error of the RMS unit.

A comparison of the upper and lower bounds on the average error is presented in Figures 2.14 and 2.15. As noted previously for the lower amplitude case, there is not an advantage gained by increasing ME > 10 or 11. The middle values of the extreme values in Fig. 2.14 do not fall as closely to the middle of the bound values as they did in Fig. 2.8. This is attributed to the asymmetrical error properties of the RMS unit that become predominant for larger signal amplitudes.

The integrator output error variance bounds are presented in Fig. 2.16 as a function of MT with ME as a parameter family. The lower bound is zero for ME > 8. Similar observations as made for Fig. 2.10 are possible for this case. However, note that the bound curves are not as tight due to the large signal amplitude and the corresponding larger error introduced by the RMS unit. It appears that the upper bound curves have leveled off for MT > 12 or 13.

The variance bounds are presented in Figures 2.17 and 2.18 as a function of ME with MT as a parameter family. Note that the upper bound variance does not decrease appreciably for ME > 7 or 8.

A third technique for analyzing the data is to fix the MT parameter at a value greater than the critical threshold and present the error statistics as a function of signal amplitude. The average error upper and lower bounds are presented in Figures 2.19 and 2.20 with MT = 13. There is close agreement between the bounds for ME $\leq$ 9. The bounds are not strongly dependent on the signal amplitude except for certain isolated values of ME, e.g., the upper bound curves for ME = 11. The variance bounds are presented in Fig. 2.21. Note the convergence of the upper bound curves as the signal amplitude increases. This is a result of the RMS unit induced error becoming predominant. The bounds are not tight and appear to diverge for increasing signal amplitude. The variance upper bound doesn't change much for ME > 9.

The results presented in this section are compared to the simulation results in Chapter 3.

A follow-on study was attempted after the results presented in Figures 2.4 through 2.21 were obtained. It was determined that the error statistics reported in Reference 8 and used in this analysis were not descriptive of the two sector approximation of (2.2). The root-mean-square error should be 0.1265% instead of 12.65%. This new information was incorporated into the computer program and runs made to determine the affect on the error statistics. The average error was unchanged, but the error variance did reflect this parameter change. The change was most pronounced at larger signal amplitudes, e.g., a maximum decrease by a factor of 2 resulted at 0.317 volts amplitude in Fig 2.1 whereas a regligible decrease was noted for amplitudes less than 0.1 volts.

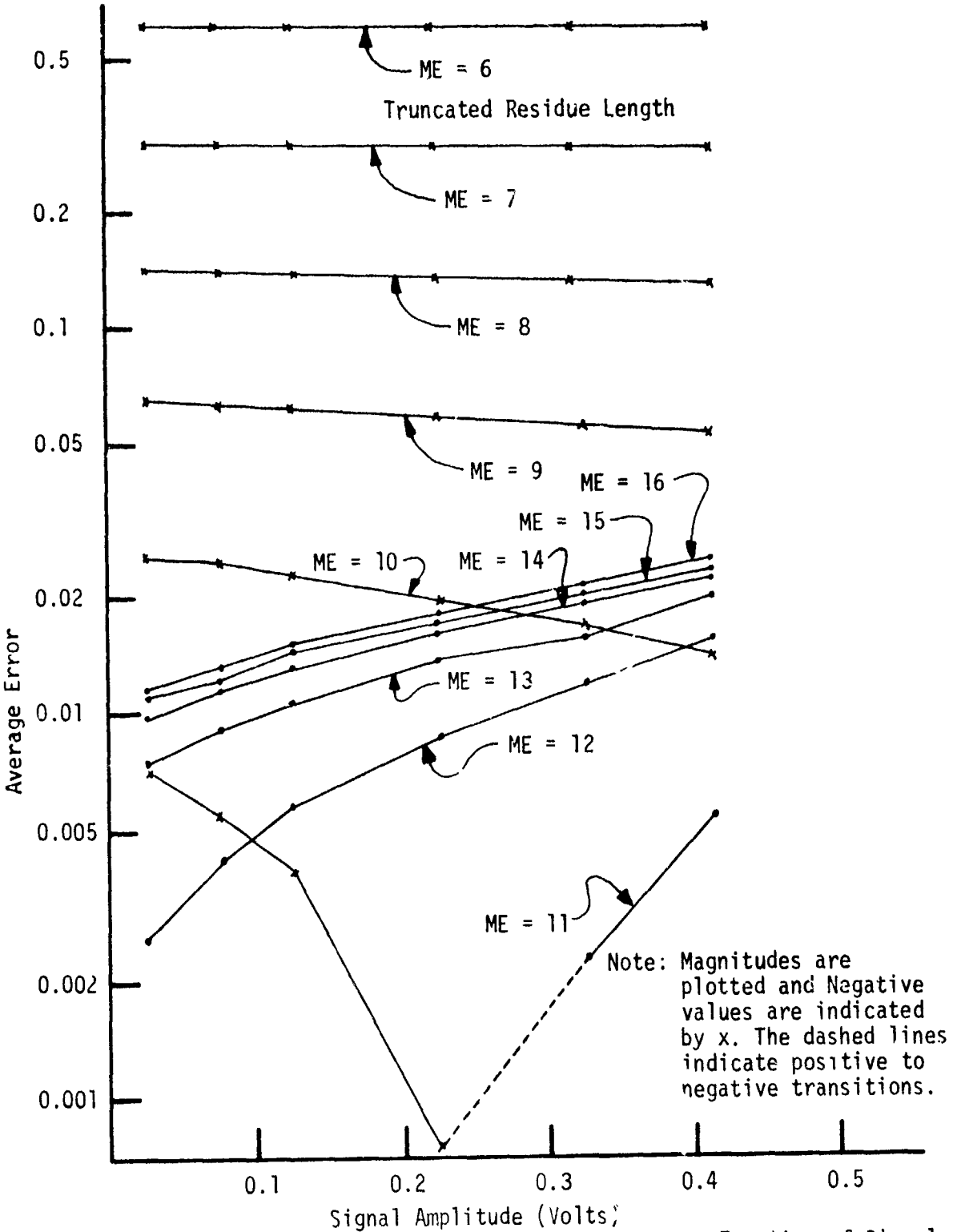


Fig. 2.19 Upper Bound on Average Integrator Error as Function of Signal Amplitude with Truncated Product Length MT = 13 (Frequency = 1500 Hz)

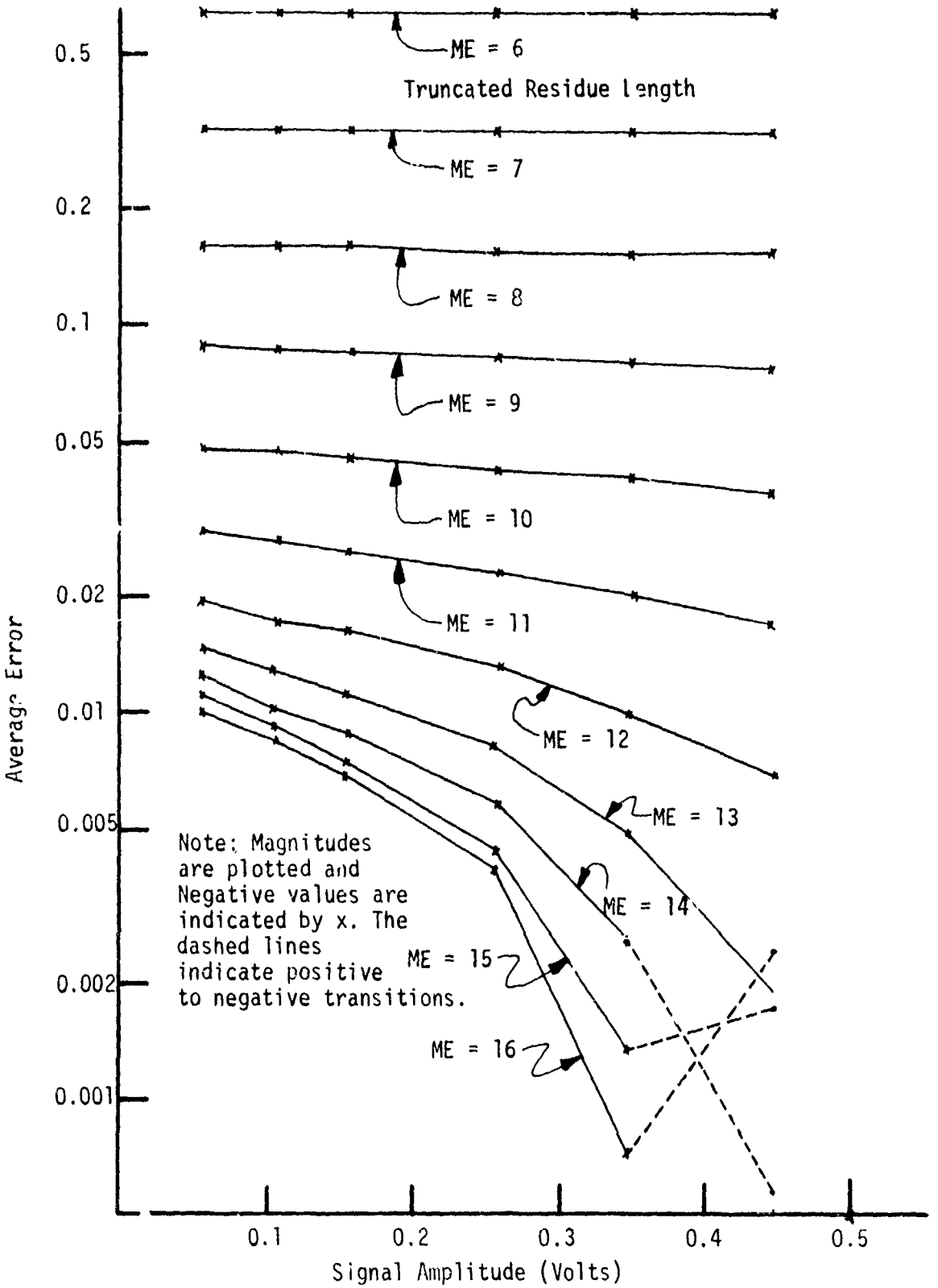


Fig. 2.20 Lower Bound on Average Integrator Error as Function of Signal Amplitude with Truncated Product Length MT = 13 (Frequency = 1500 Hz)

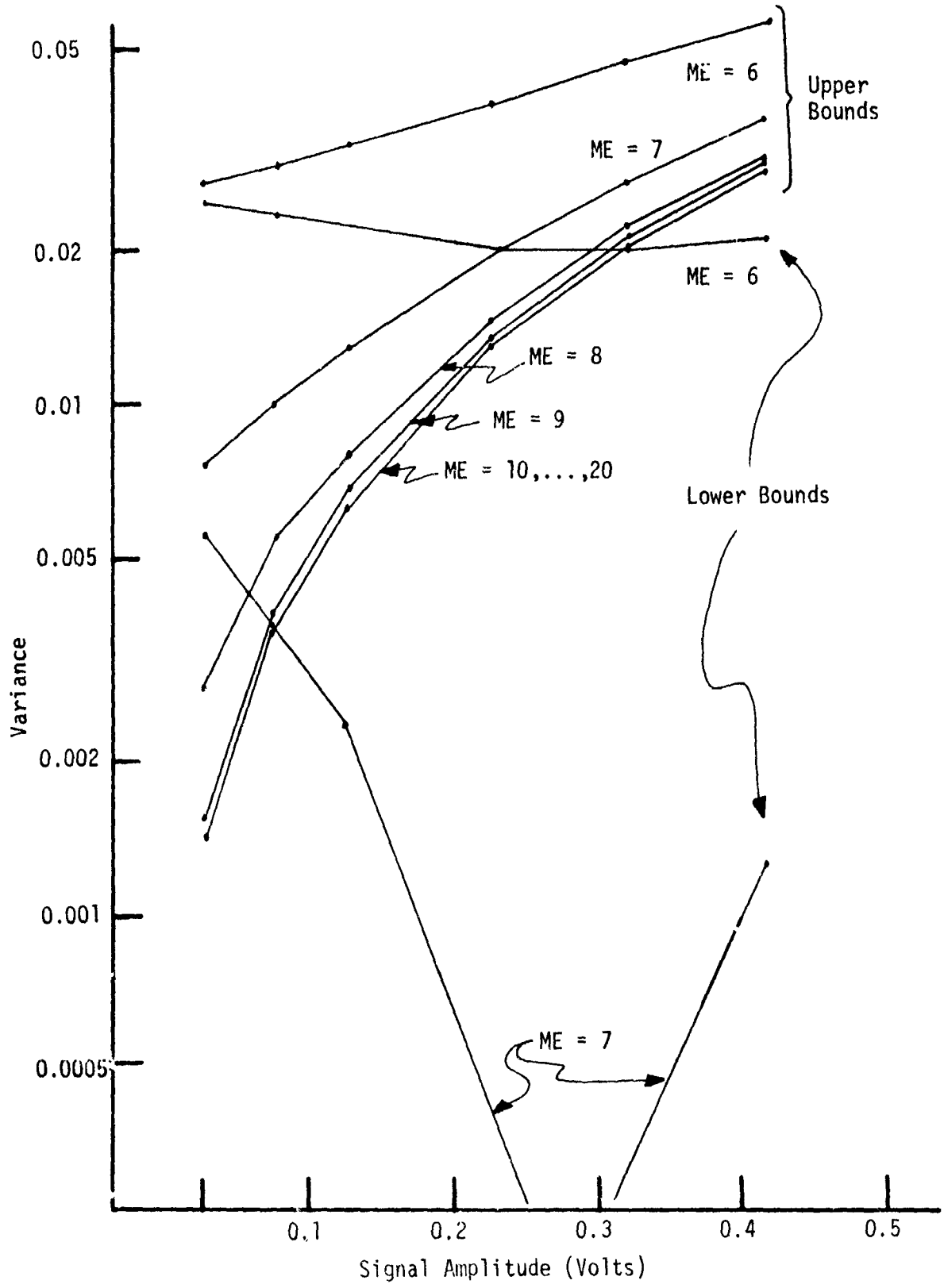


Fig. 2.21 Bounds on Integrator Error Variance as Function of Signal Amplitude with Truncated Product Length MT = 13 (Frequency = 1500 Hz)

CHAPTER 3

SIMULATION ANALYSIS OF FIXED POINT-PROCESSOR
by Bhadrayu J. Trivedi

The simulation study of the fixed-point MTI processor was performed with the aid of a Fortran simulation program developed at The University of Alabama, Huntsville [14]. A description of the simulation program and the modifications made to obtain statistical information and a discussion of the simulation results are presented in this chapter. Section 3.1 deals with the description of the program and Section 3.2 with the discussion of simulation results.

3.1 DESCRIPTION OF SIMULATION PROGRAM

The simulation program uses an integer programming technique which represents all the fixed-point binary numbers in the processor as decimal integer numbers. This technique is described in detail in Section 3.1.1. The routines for simulating basic operations such as addition, multiplication, truncation and/or expansion, finding magnitude, etc., of binary numbers; are based on this technique. The routines which simulate the system blocks such as the coefficient quantizer, input A/D converter, the digital filter, the RMS unit and the integrator also use the same technique as well as the routines which simulate the basic binary arithmetic operations. These routines and the overall program are described in Section 3.1.2 with the help of detailed flow charts. The simulation program as it was used for this project is listed in Appendix E along with a discussion of data card-formats. Section 3.1.3 suggests improvements in the simulation program that could be implemented to achieve higher speed and increased efficiency for a typical program run.

3.1.1 Detailed Integer Programming Technique

Throughout the fixed-point processor the arithmetic operations are performed in a two's complement fixed-point binary scheme. For the purpose of Fortran simulation they are represented as positive decimal integers. For example, suppose that at a particular point in the processor a number is represented with N binary digits including the sign. Its level is defined as

$$\text{LEVEL} = 2^N . \tag{3.1}$$

If the number has no sign bit, a fictitious sign bit should be added to compute the level. Let there be L binary digits for the fractional part of the number without sign, then the quantization interval is defined as

$$\text{QI} = 2^{-L} . \tag{3.2}$$

Therefore,

$$\text{LEVEL}/2 = 2^{N-1} \tag{3.3}$$

represents the sign bit in this two's complement integer notation. Table 3-1 illustrates conversion from a real two's complement notation to a "integer decimal notation." The real two's complement binary number is integerized by shifting the radix point to the right-most position. This binary number then is expressed as an integer decimal number. Note that in this form the number is always expressed as a positive number. If the number is smaller than (LEVEL/2) it is a positive number and otherwise negative. To obtain its real decimal value, LEVEL has to be subtracted from the negative number; the positive number is not altered. The number then has to be multiplied by its quantization interval as defined in Equation (3.2).

The following example illustrates an application of the technique. Example: Suppose the number $(-0.25)_{10}$ is to be added to itself six times to obtain the answer $(-1.5)_{10}$. If $(-0.25)_{10}$ is represented with N=3 bits its representation in integer decimal notation is 7_{10} (i.e., 1.11_2). To avoid an overflow it is necessary to expand the bit-length to N=4 thereby adding an extra integer bit. Now the number is represented as 15_{10} (i.e., 11.11_2). Now if the number is added to itself six times, each time ignoring the carry, then the result is 10_{10} (i.e., 10.10_2). Now the result can be truncated down to a bit-length N=3 by deleting the least significant fractional bit. If the answer is to be expressed in real decimal, then it should be multiplied by (2^{-1}), the new quantization interval after truncation. The original number is represented in the second row and the final answer in the fourth row of Table 3-1.

TABLE 3-1
INTEGER PROGRAMMING TECHNIQUE ILLUSTRATION

Real Decimal	Real Two's Complement	Integer Two's Complement	Integer Decimal	Q.I
+0.25	0.01	001	1	2^{-2}
-0.25	1.11	111	7	2^{-2}
+1.5	01.1	011	3	2^{-1}
-1.5	10.1	101	5	2^{-1}

The above example illustrates that it is essential to use the proper quantization intervals to convert numbers from integer decimal notation to real decimal and vice versa, before and after undergoing an arithmetic operation. If two fractional numbers are multiplied together then the quantization interval of the product is the product of the quantization intervals of the two numbers. The next section will show how the simulation routines keep track and make use of the appropriate quantization intervals and levels.

3.1.2 Flow Chart of Simulation Program

In this section the MAIN program and the subroutine STARTQ are described. The subroutine STARTQ is the first step implemented by the MAIN program to read all input parameters, compute data for clutter generation, and quantize the filter coefficients. It does not perform any special task and normally this would be done in a MAIN program itself, hence the MAIN and STARTQ are described together. Next, the group of routines which simulate the basic binary arithmetic operations such as input and coefficient quantization, addition, multiplication, etc. are described. This is followed by a description of routines which simulate the system functions such as signal and clutter generation, digital filter, RMS unit, etc. An effort is made to emphasize features which have been added to the basic program [14] and those not already apparent from the discussion of the program in Reference [14]. Detailed flow charts were developed for this purpose.

The objective of the program is to simulate the fixed-point processor shown in Fig. 2.1 and Fig. 2.3 and generate results that can be used in a statistical study of quantization errors. The flow charts for the MAIN program and the subroutine STARTQ are shown in Figs. 3.1 and 3.2 respectively. The simulation is performed over NDWEL antenna dwells each containing NPULSE signal-plus-clutter samples per range bin. NPULSE is a product of NDELAY, the number of filter coefficients, and NCYCLE, the number of residues to be integrated. It is a system requirement that NPULSE be less than 48. The MAIN program calls STARTQ to start the simulation by reading in clutter, signal, radar and filter parameters which are subsequently printed out. These parameters are explained in Appendix E with the details of how they are specified on input cards. Next, STARTQ calls the subroutine COEF to quantize the filter coefficients and print the unquantized and quantized values. Then, a set of Gaussian random samples are generated by calling the subroutines ANIT and RANDM. The impulse response of the digital filter used in the clutter generation is calculated. The purpose of this filter is to yield clutter samples with a desired power spectrum from the input Gaussian samples. The clutter filtering is implemented in subroutine UPDATQ. STARTQ also generates scale factors for signal and clutter combination. After this the control reverts back to the MAIN program and all the parameters read and computed by STARTQ are transferred to MAIN. The MAIN program next reads the parameters which control RMS-statistics-print, clutter and theoretical output options. The second option controls whether clutter is to be added to the doppler signal or not. The third option determines whether the theoretical output (infinite precision answer) is to be computed with a quantized or an unquantized set of filter coefficients. If the unquantized set is used then a valid basis for comparison between the fixed-point and floating-point processor statistics can be provided. This follows because the quantization error introduced for the same number of coefficient bits and the same set of coefficients is different for the two processors. The MAIN program simulates only one system block, viz., the integrator. All the other system blocks and functions are delegated to different subprograms. The MAIN program calculates the statistics for the hardware RMS unit output at the end of each residue or cycle. The

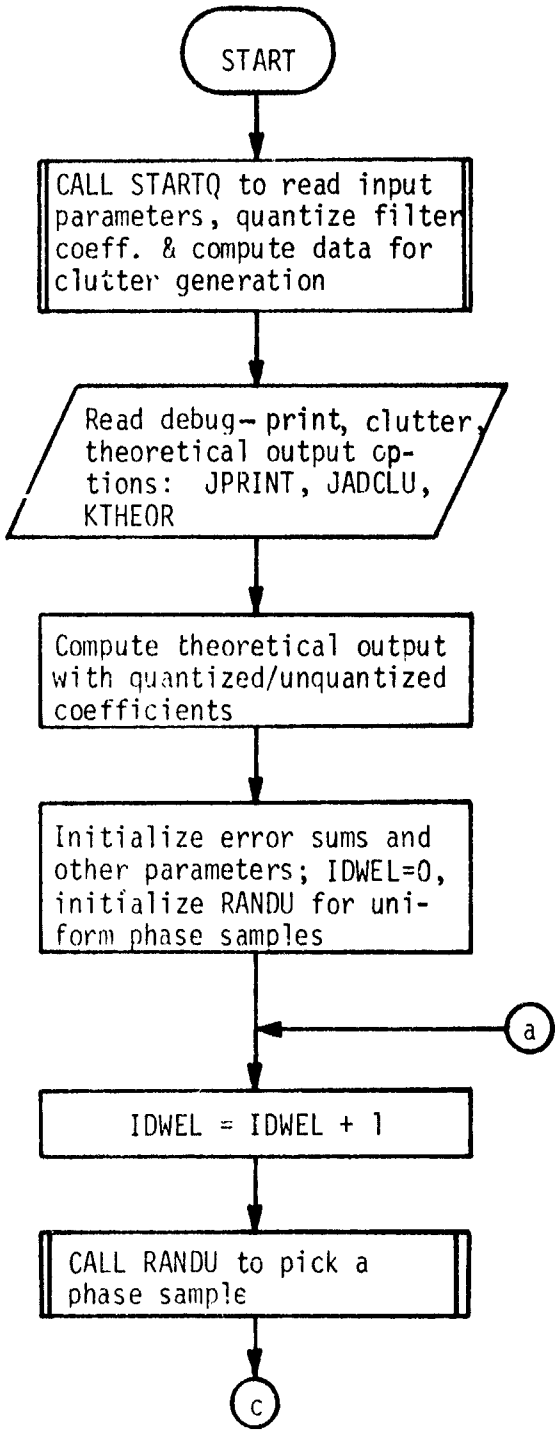


Fig. 3.1 Flow chart for the MAIN program of the Fixed-Point Processor Simulation program

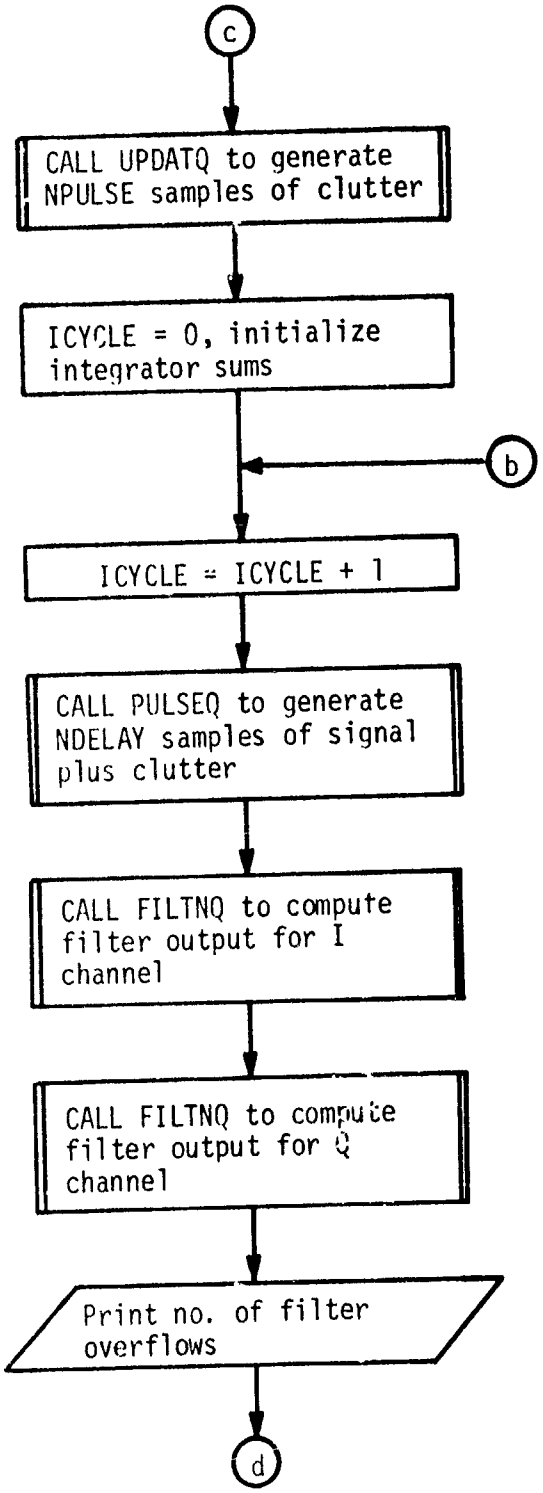


Fig. 3.1 (Continued)

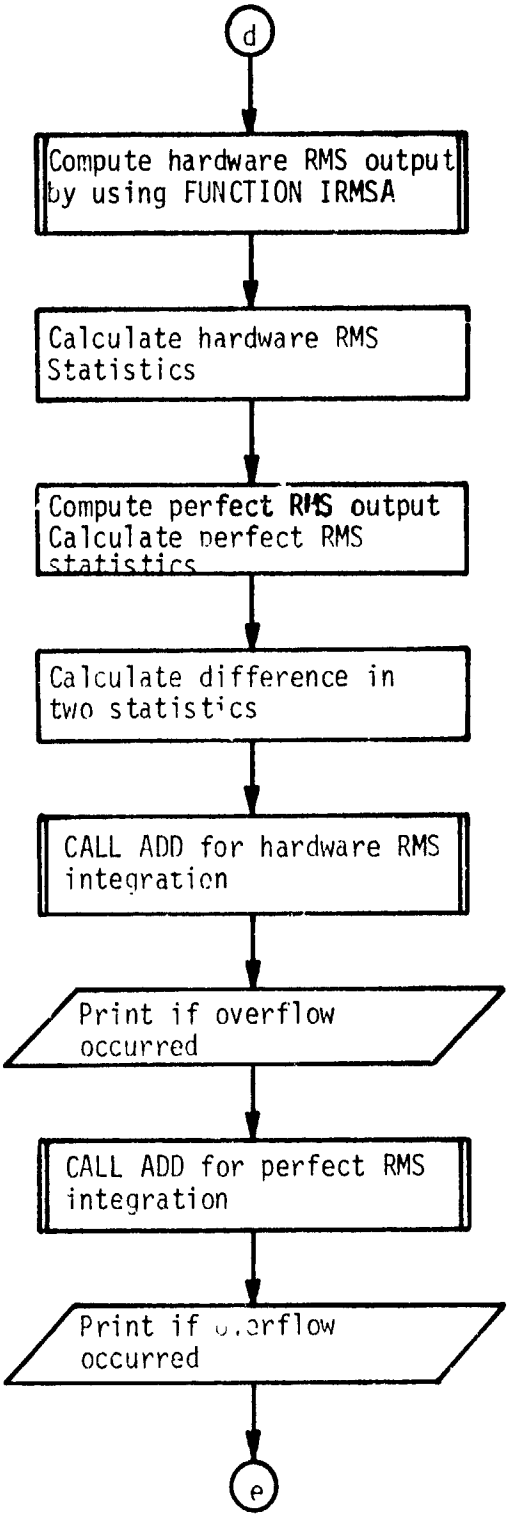


Fig. 3.1 (Continued)

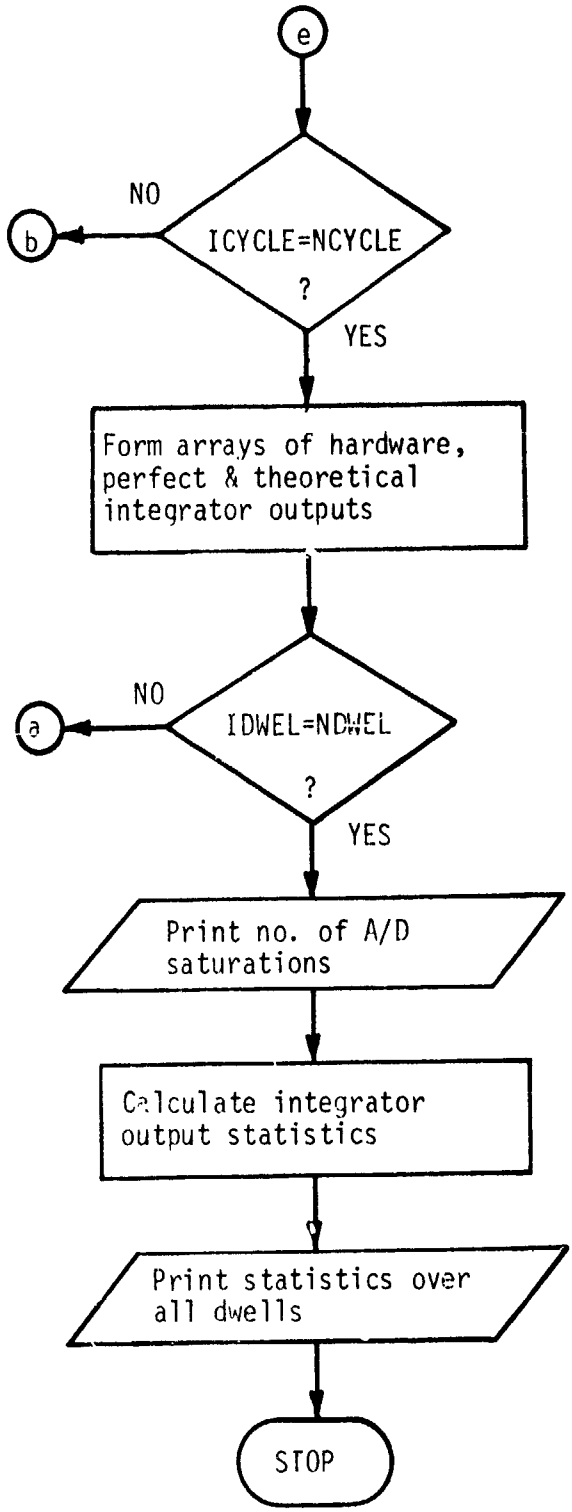


Fig. 3.1 (Continued)

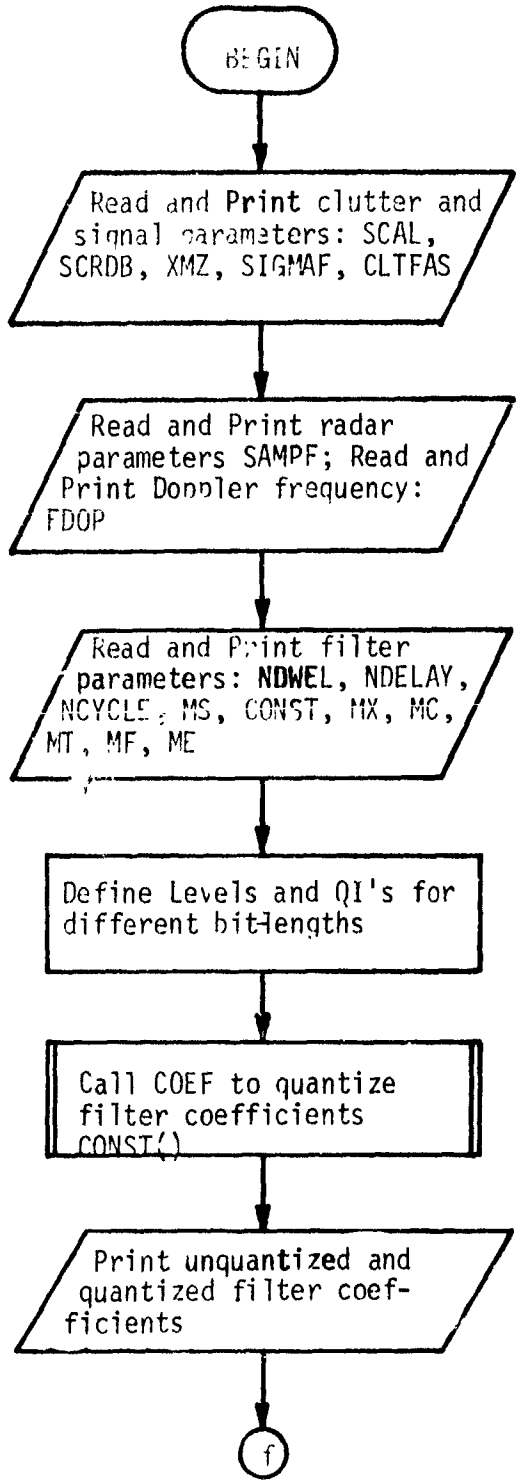


Fig. 3.2 Flow Chart for Subroutine STARTQ

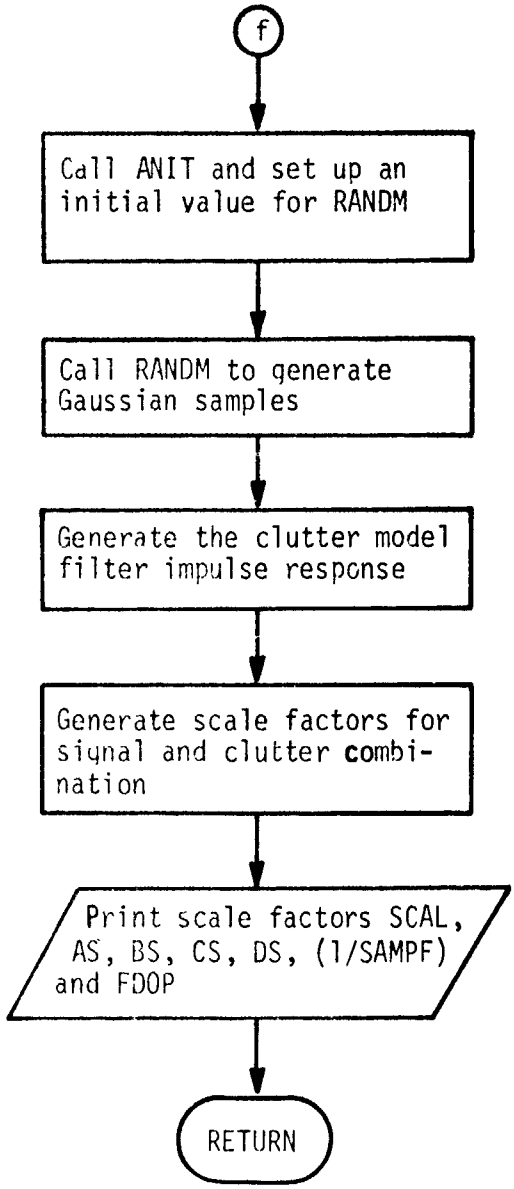


Fig. 3.2 (Continued)

same is also calculated for the output of an imaginary perfect RMS unit. The statistics involve computation of the maximum, the minimum, the mean and the variance of the error in the output. The error is defined as the actual output minus the theoretical output (see Equations 2.3, 2.6, 2.8, 2.12). The statistics for the difference between the hardware and perfect RMS outputs are also computed. A similar statistical analysis is carried out on the integrator outputs, i.e., at the end of each antenna dwell. Note that the simulation results presented in Section 3.2 pertain to the integrator output statistics. The nature of the statistics is biased, i.e., the definition of variance is,

$$\sigma_x^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \qquad (3.4)$$

as given in reference [15]. The MAIN program gives an indication if an overflow occurred in the filter or integrator operation and also the total number of A/D saturations over all dwells. The MAIN program calls the subroutine RANDU for picking a uniformly distributed random phase starting angle for the doppler signal. The subroutines RANDU and RANDM are not described in this report because they are canned programs which depend on the host computer being used.

The subroutine IAD simulates the A/D converter scheme given by Equation 3.1 and Fig. 3.1 of Reference [14]. This scheme represents the actual 'Computer Labs - A/D Converter Model - 5905' being used by the Radar Technology Branch of the U.S. Army Missile Command at Huntsville. Note that the output of the A/D converter is in two's complement binary form with a unique "offset", i.e., the quantization intervals are centered at ±QI/2, ±3QI/2, ±5QI/2 and so on. Normally one would expect the quantization intervals to be centered at 0.0, ±QI, ±2QI, ±3QI and so on [16]. For an "ideal" 3-bit A/D converter equal and opposite analog voltages if added after conversion to two's complement numbers yield a zero result if the carry is ignored, e.g., when QI=0.25,

	+ 0.125	= 001_2		+ 0.375	= 011_2
	- 0.125	= 111_2		- 0.375	= 110_2
sum	0.000	= $1)000_2$	,	0.000	= $1)000_2$

For a 3-bit A/D converter with the actual offset scheme being used,

	+ 0.125	= 000_2		+ 0.375	= 001_2
	- 0.125	= 111_2		- 0.375	= 110_2
sum	0.000	$\neq$ 111_2	,	0.000	$\neq$ 111_2

The sum in this case does not result in a zero. This is a disadvantage of the A/D scheme being used, however, once the output numbers are available they are treated as two's complement numbers. Although these numbers

contain the 'offset', no correction is applied because the actual hardware system [1] is built in this manner. The flow chart for the subroutine IAD is shown in Fig. 3.3. Note that it gives a saturation indication if

$$|x| \geq 1.0 \ , \tag{3.5}$$

where x is the input analog signal sample.

The flow chart for subroutine COEF is shown in Fig. 3.4. COEF quantizes a filter coefficient to a specified bit-length. The routine requires that the quantization interval and the level be specified. Note that it uses a round off procedure which increases the magnitude of the number by one if the remainder of the number after integer quantization is greater than or equal to a half quantization interval (see flow chart for details).

The simulation of two's complement addition is implemented by the subroutine ADD. Two numbers in two's complement 'integer decimal' form with the same level are added, the carry is ignored, and any overflow is detected and a flag is set by this routine. The flow chart is shown in Fig. 3.5.

The function subprogram MUL simulates two's complement multiplication. The flow chart is shown in Fig. 3.6. The two numbers are first converted to sign magnitude representation (in decimal integers), multiplied and then converted back to the two's complement 'integer decimal' form.

For word-length truncation and expansion a function subprogram ITREX was used which truncates the least significant fractional bits and expands at the most significant integer end. The expansion for a positive number involves appending zeros and that for a negative number involves ones at the most significant integer end. The extent of truncation or expansion is specified by levels. If the number to be truncated or expanded does not have a sign bit it is essential to compute the level by adding a fictitous sign bit to the actual bit-length. The flow chart for ITREX is shown in Fig. 3.7.

The function subprogram MAGNF finds the magnitude of a two's complement number and expresses it as a positive decimal integer. Its flow chart is shown in Fig. 3.8. Note that it rounds up a negative number if it is the largest negative number representable in the two's complement scheme (LEVEL/2) to avoid a zero output. As for example, for a 3-bit number (LEVEL/2) = 100_2. This needs to be rounded up to 101_2 to avoid a zero output.

For each antenna dwell the MAIN program calls the subroutine UPDATQ once to generate NPULSE samples of clutter for each of the I and the Q channel filters. UPDATQ uses the 1024 Gaussian random samples generated by STARTQ. It uses INCGAU samples (a subset of the above 1024) for each dwell. INCGAU is calculated by STARTQ and depends on clutter and system parameters. If all the samples generated by STARTQ are exhausted it calls RANDM and generates 1024 more Gaussian samples. It then convolves INCGAU/2 samples with the clutter model filter impulse response (generated by STARTQ) and generates a set of NPULSE clutter samples shaped according to the desired power spectrum. This process is duplicated for the Q channel as can be seen in the flow chart in Fig. 3.9.

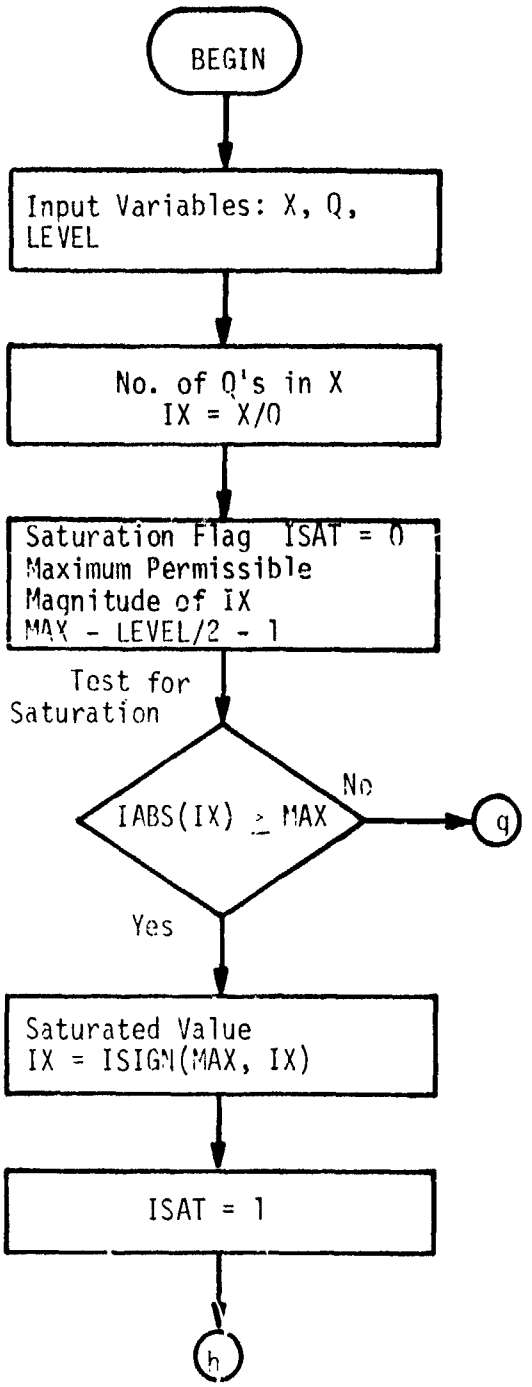


Fig. 3.3 Flow Chart for Subroutine IAD

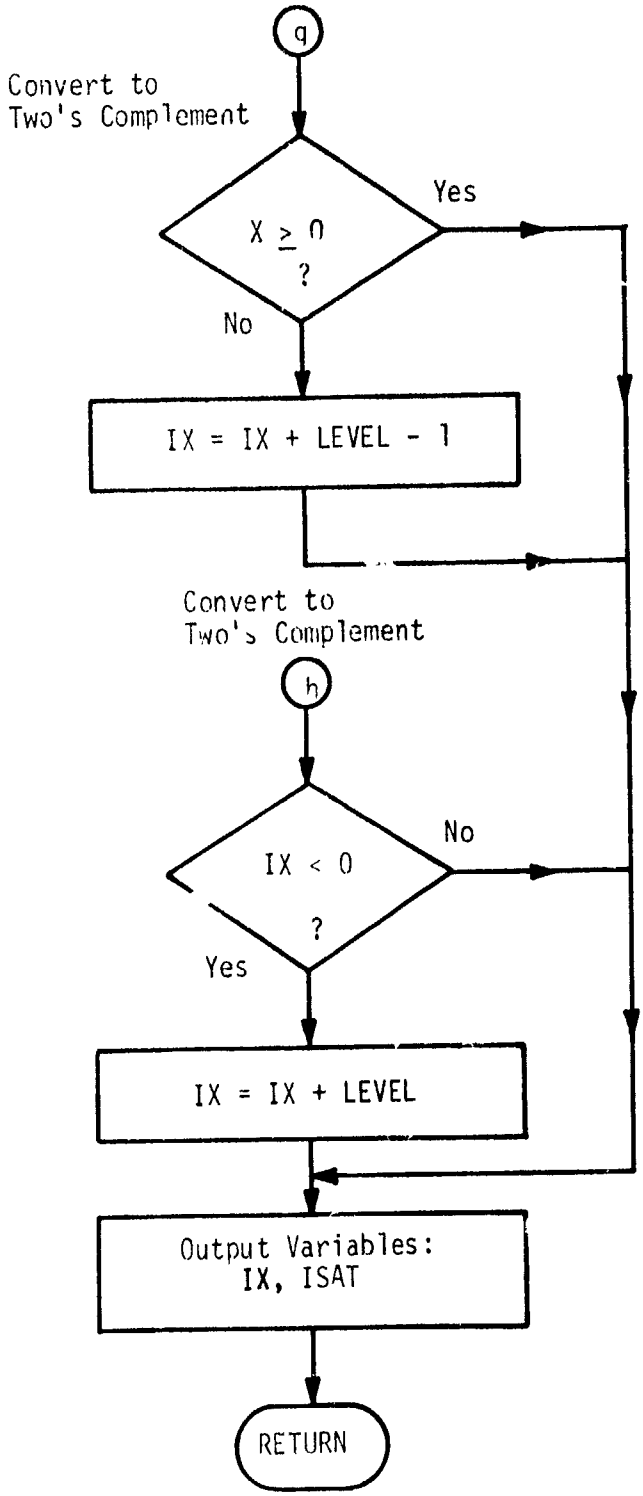


Fig. 3.3 (Continued)

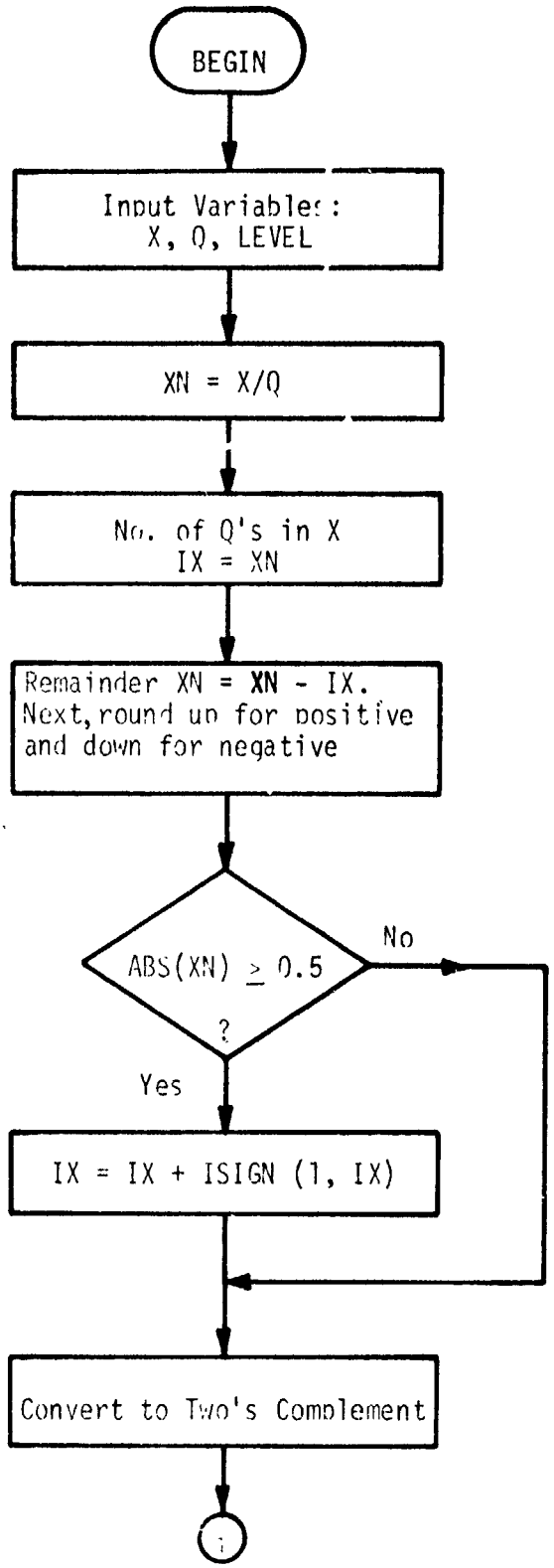


Fig. 3.4 Flow Chart for Subroutine COEF

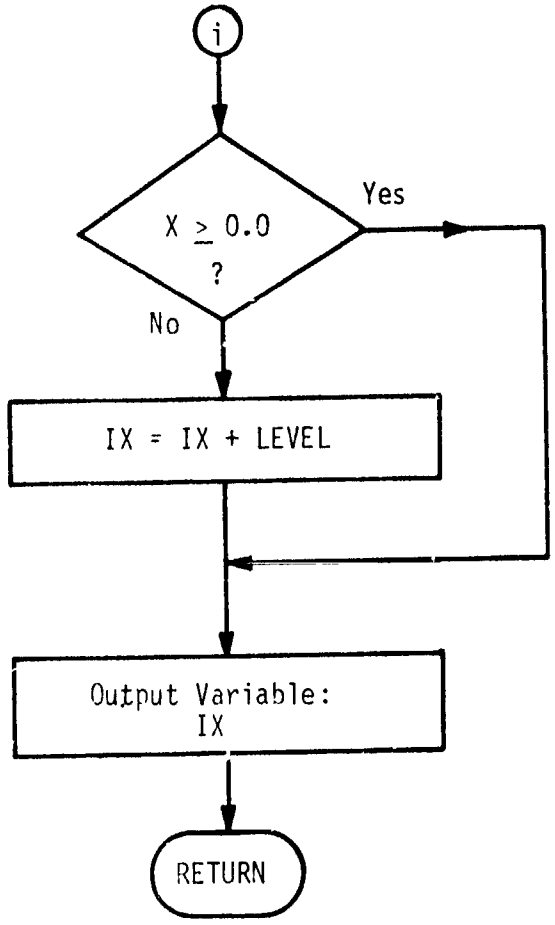


Fig. 3.4 (Continued)

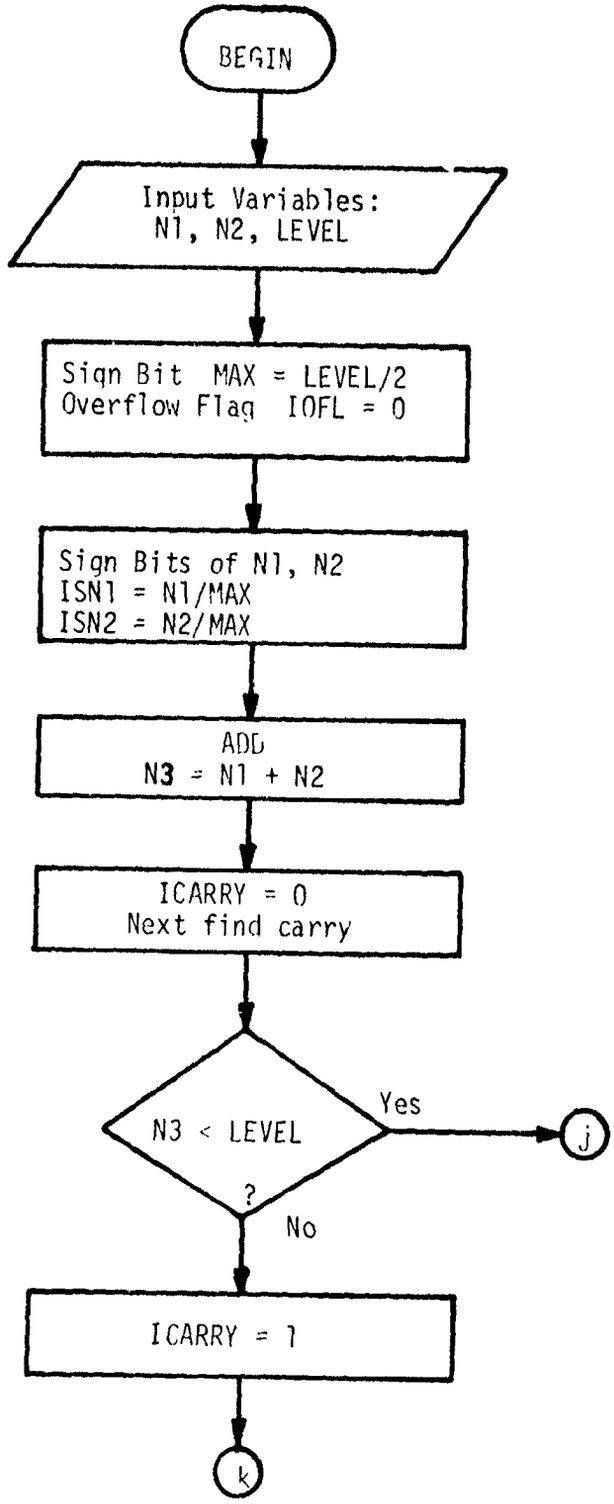


Fig. 3.5 Flow Chart for Subroutine ADD

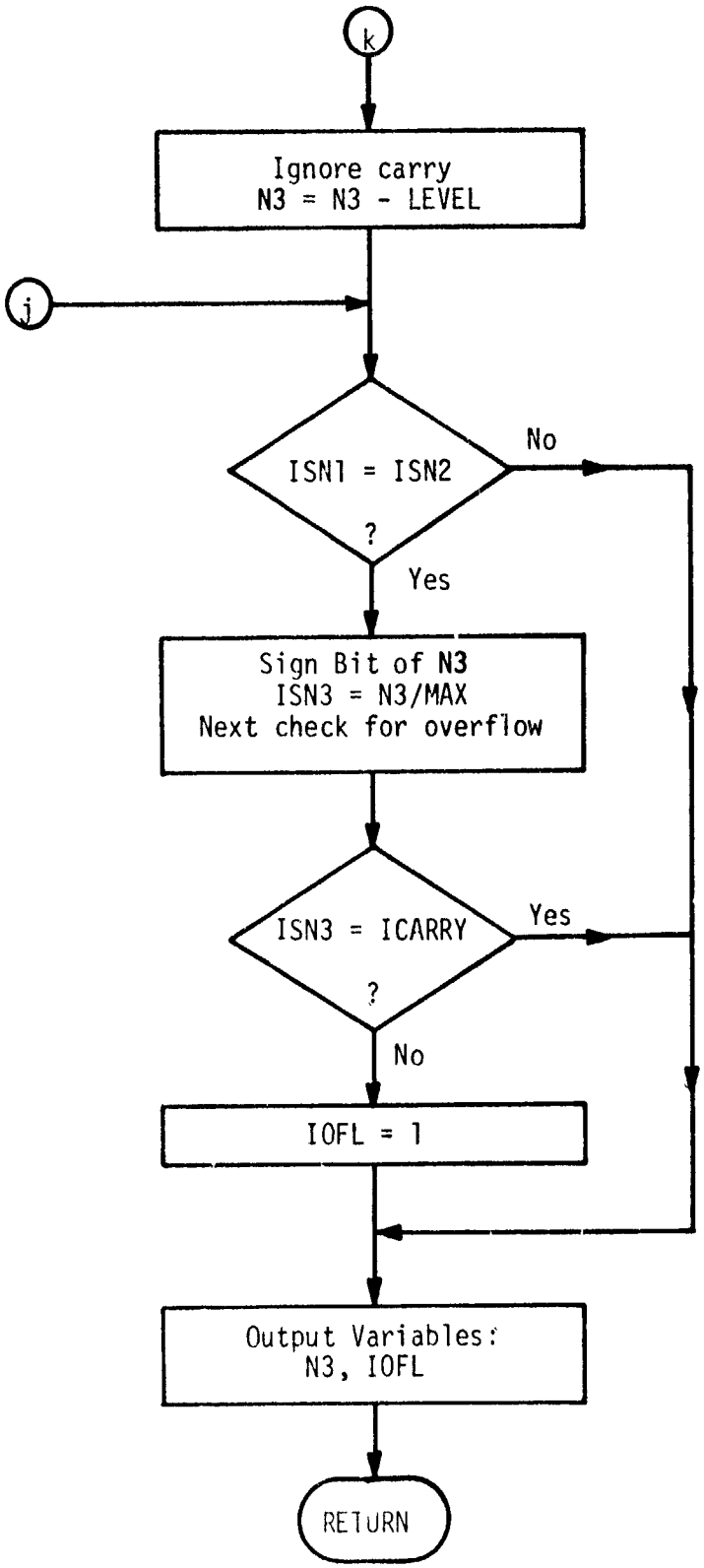


Fig. 3.5 (Continued)

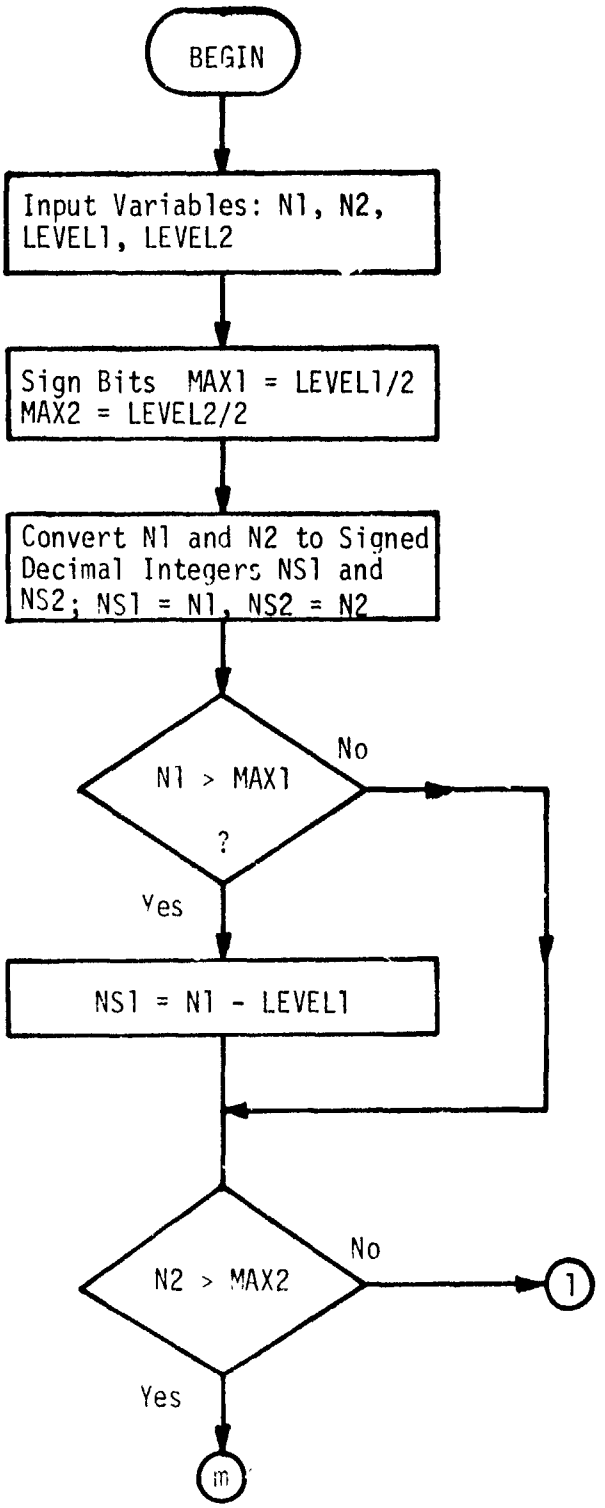


Fig. 3.6 Flow Chart for Function Subprogram MUL

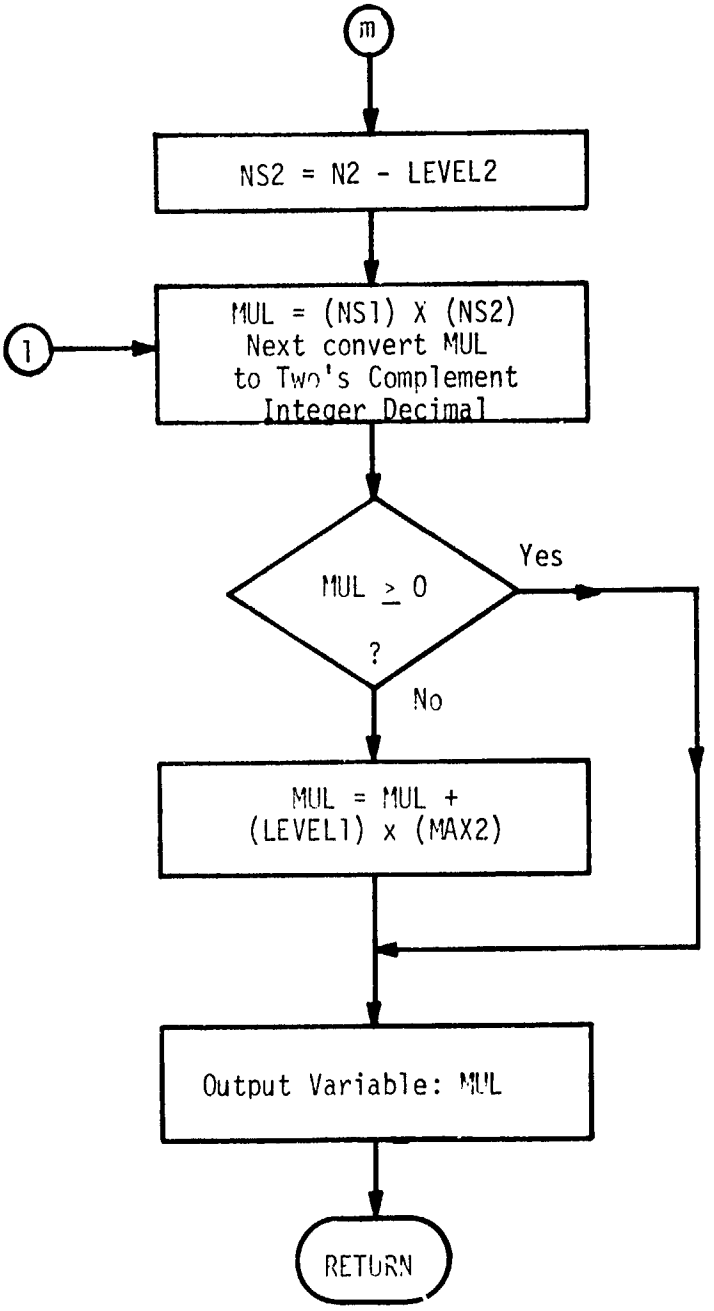


Fig. 3.6 (Continued)

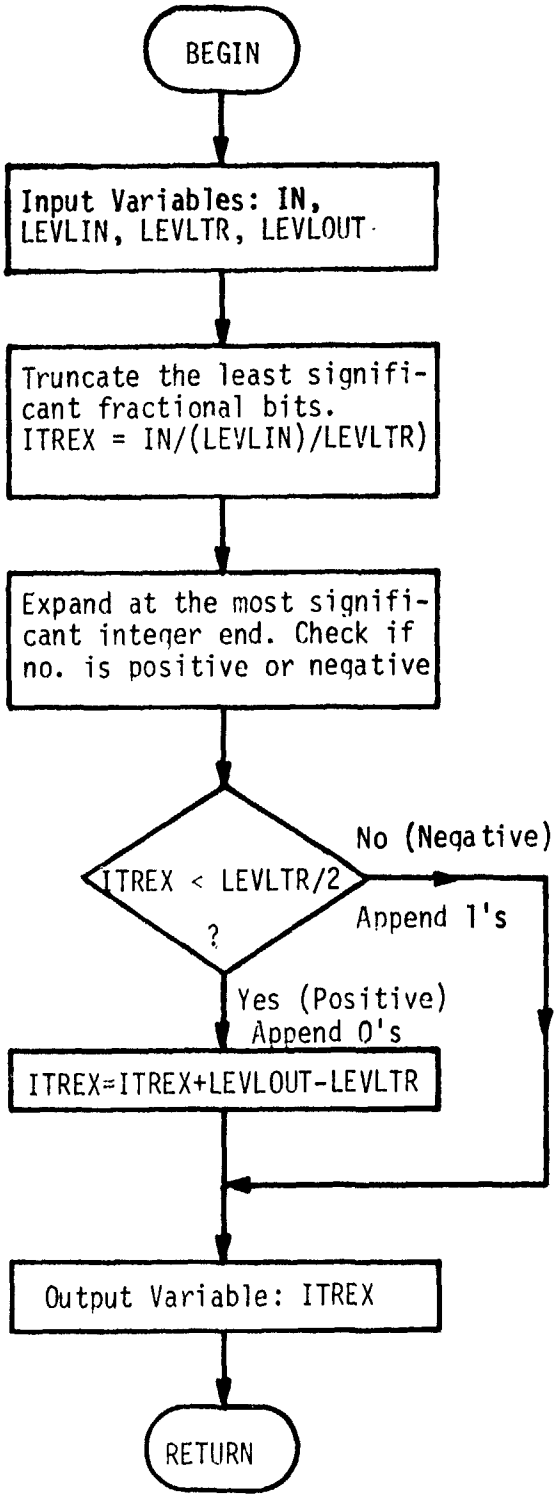


Fig. 3.7 Flow Chart for Function Subprogram ITREX

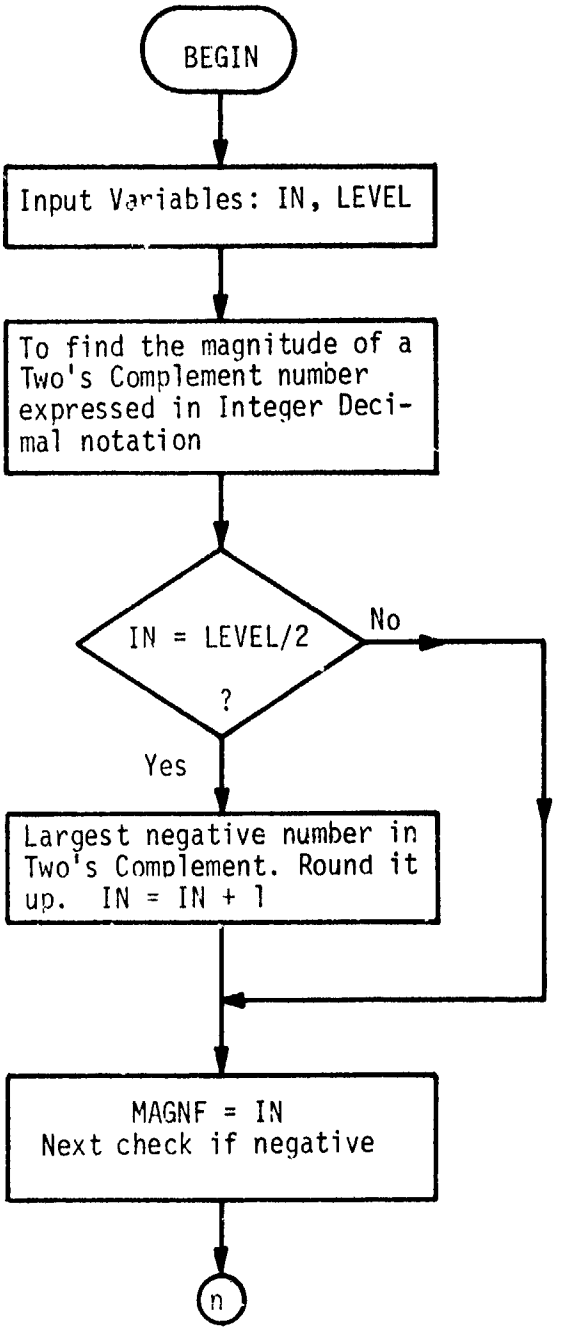


Fig. 3.8 Flow Chart for Function Subprogram MAGNF

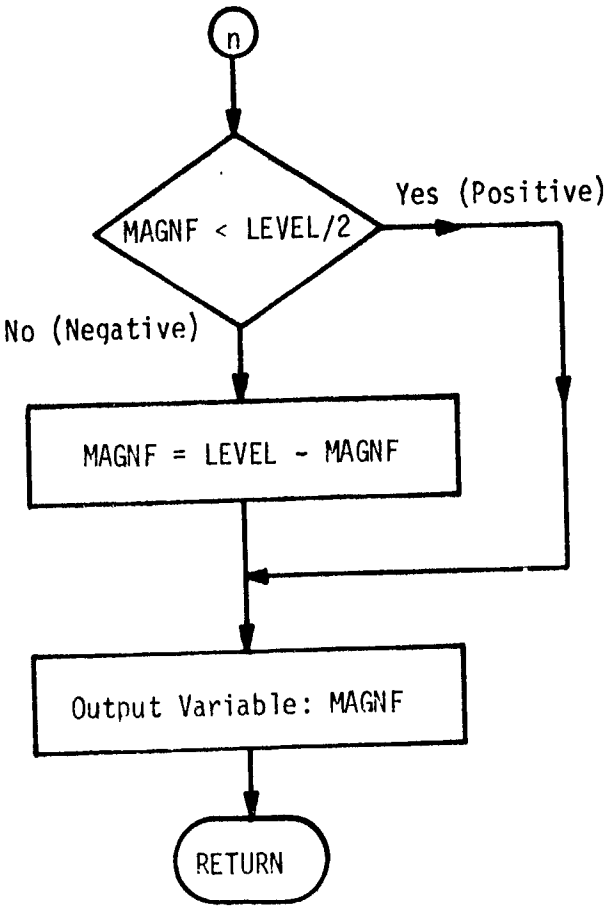


Fig. 3.8 (Continued)

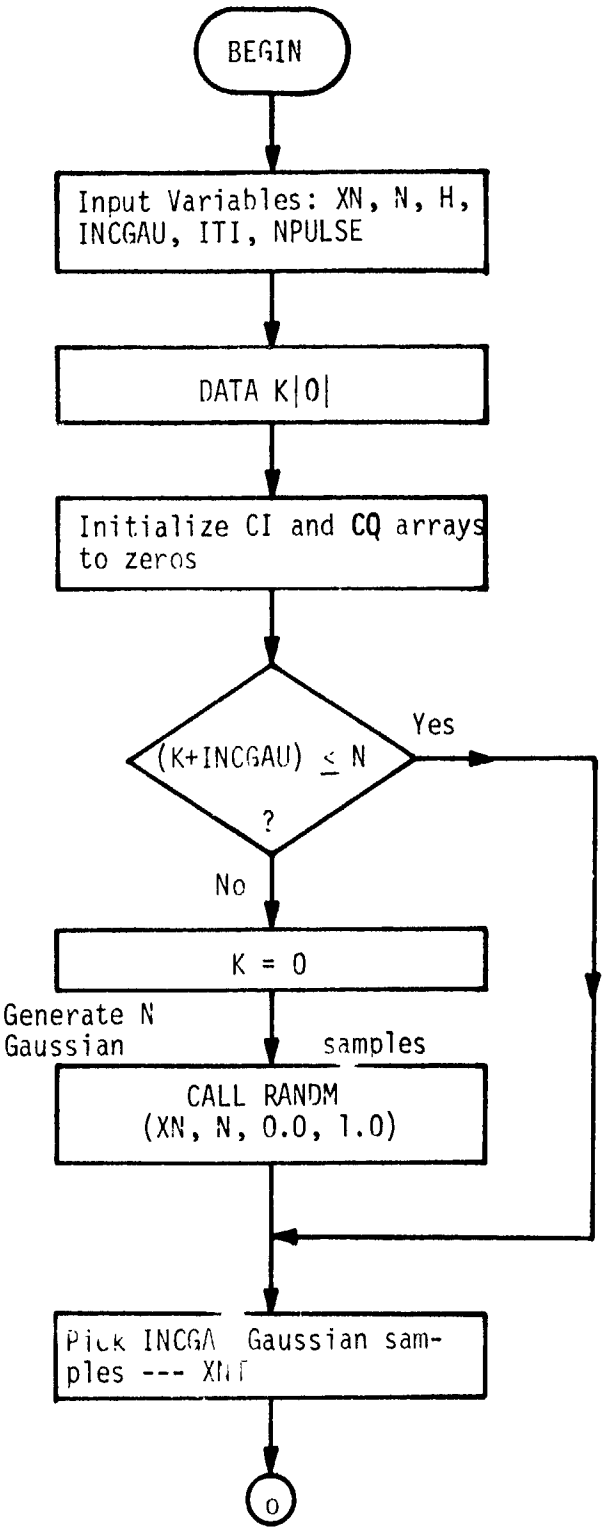


Fig. 3.9 Flow Chart for Subroutine UPDATQ

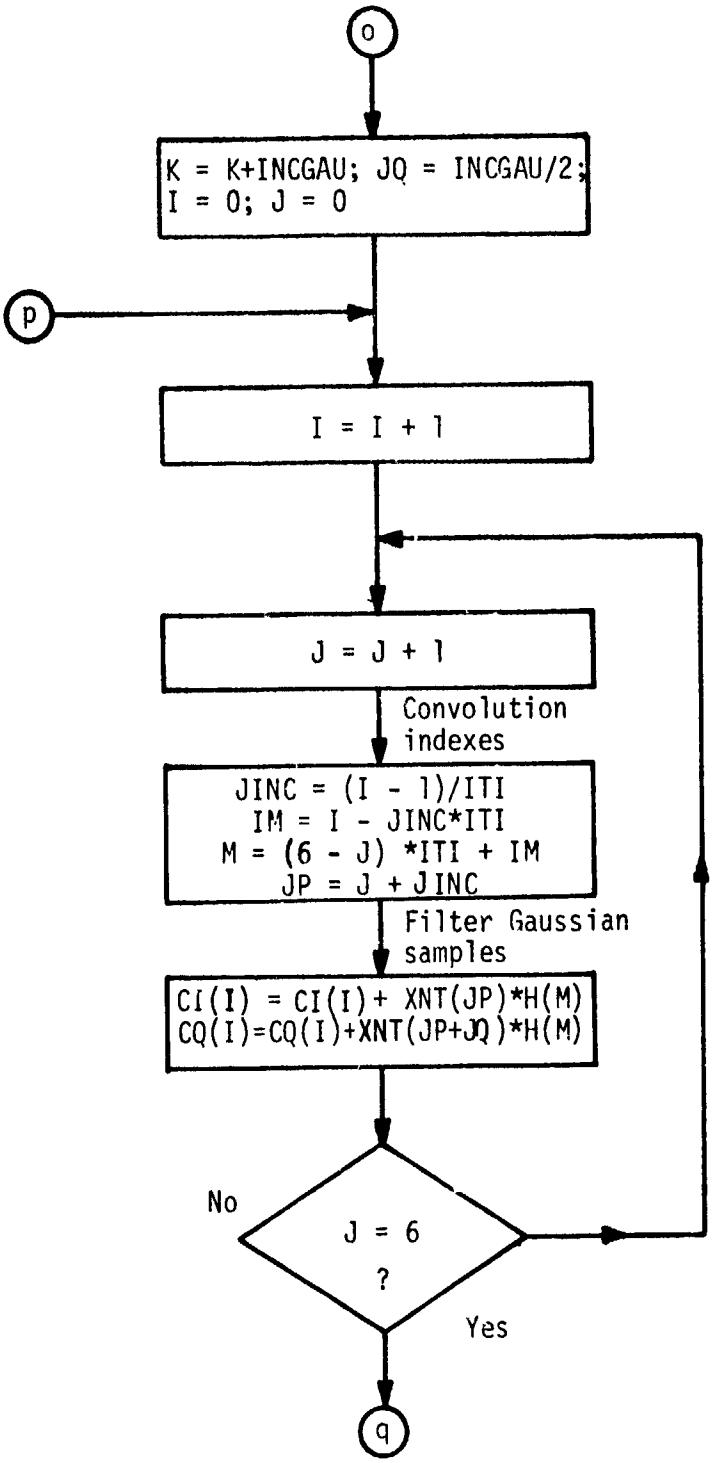


Fig. 3.9 (Continued)

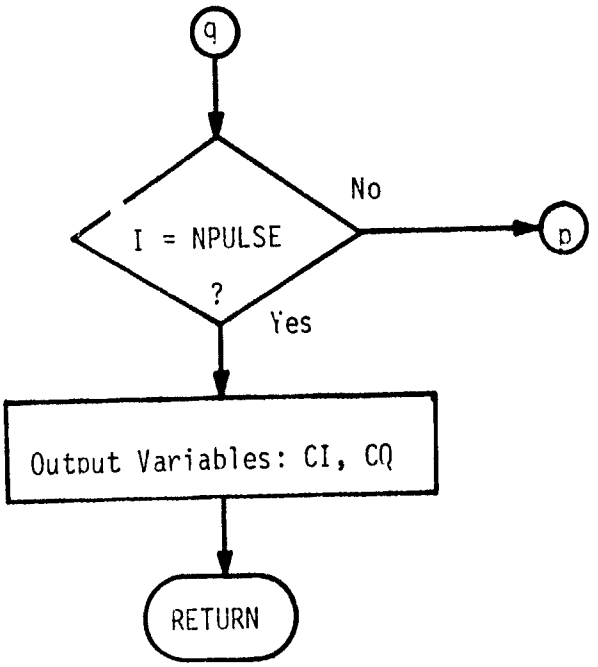


Fig. 3.9 (Continued)

For each residue (or cycle) the MAIN program calls the subroutine PULSEQ once to generate NDELAY samples of signal using the starting phase picked by MAIN for the first sample of the first residue. For each subsequent sample the phase is incremented by

$$2\pi \left(\frac{\text{Doppler frequency}}{\text{Pulse Repetition frequency}} \right) .$$

If clutter is to be added, NDELAY out of NPULSE samples of clutter generated by UPDATQ are utilized. For the next cycle the next consecutive set of NDELAY samples are used. The flow chart for PULSEQ is shown in Fig. 3.10.

The subroutine FILTNQ simulates the fixed-window non-recursive MTI filter being used. FILTNQ is used for both the in-phase and the quadrature channel filtering. As can be seen from the flow chart in Fig. 3.11, it makes use of the subprograms IAD, MUL, ITREX, ADD and MAGNF for all the computations. IAD converts the analog signal sample provided by PULSEQ to a two's complement number, MUL multiplies it by the respective quantized filter coefficient, ITREX truncates and expands the product, ADD accumulates NDELAY such products and finally MAGNF gives the magnitude of the filter accumulator after NDELAY additions. FILTNQ also counts the number of A/D saturations and the number of addition overflows.

The function subprogram IRMSA is flow charted in Fig. 3.12 and implements the hardware RMS approximation algorithm. The algorithm is outlined by Equation 2.2.

The integrator simply accumulates the RMS outputs and is implemented by the MAIN program by using ADD.

3.1.3 Suggestions for Improvement

From the point of view of increasing the speed and efficiency of computation the following improvements in the simulation package are in order:

1) Since STARTQ is used once only in the entire simulation of NDWEL dwells and does not perform a repetitive task it ought to be eliminated as a subroutine. By making it a part of the MAIN program the time lost in calling STARTQ and transferring all the parameters read and computed by STARTQ to MAIN can be saved.
2) STARTQ and UPDATQ generate Gaussian random samples and convert them to clutter samples even when clutter is not to be used in the simulation. The time lost in this can be saved by bypassing these steps if clutter is not to be used.
3) Presently STARTQ generates 1024 Gaussian random samples and for each dwell UPDATQ tests if more are necessary and if so generates 1024 more samples. A better way would be to generate the exact number of samples necessary for each dwell by UPDATQ itself.

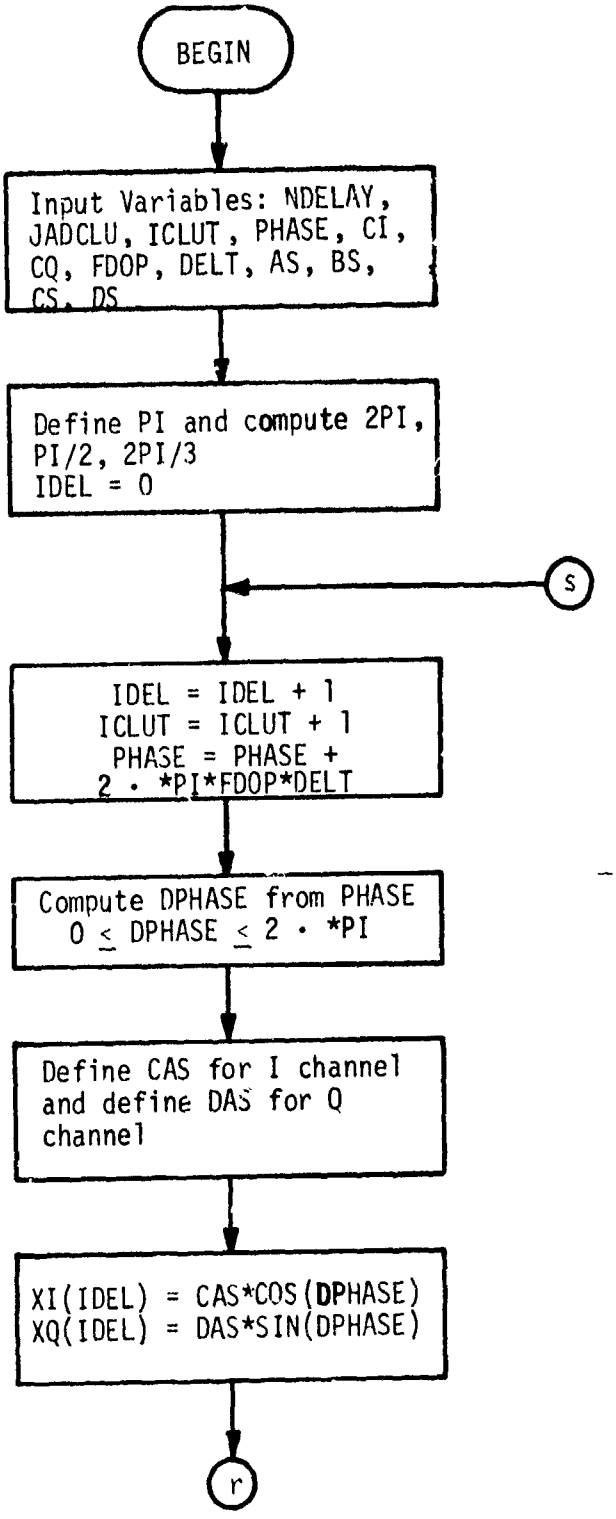


Fig. 3.10 Flow Chart for Subroutine PULSEO

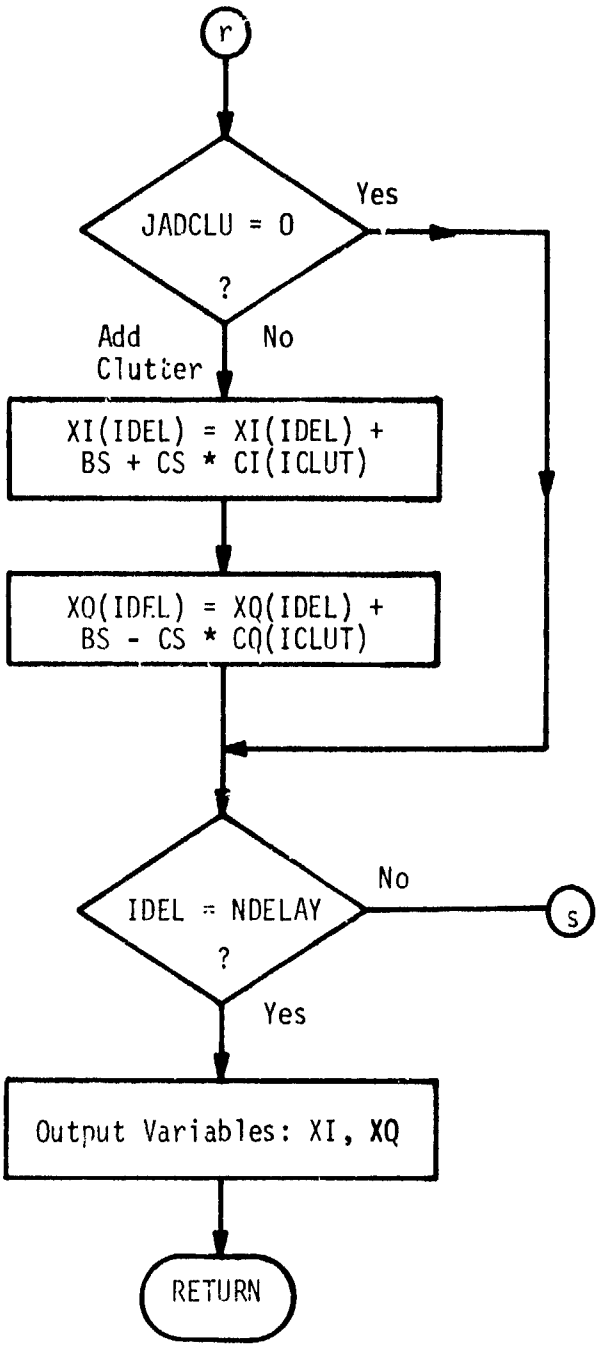


Fig. 3.10 (Continued)

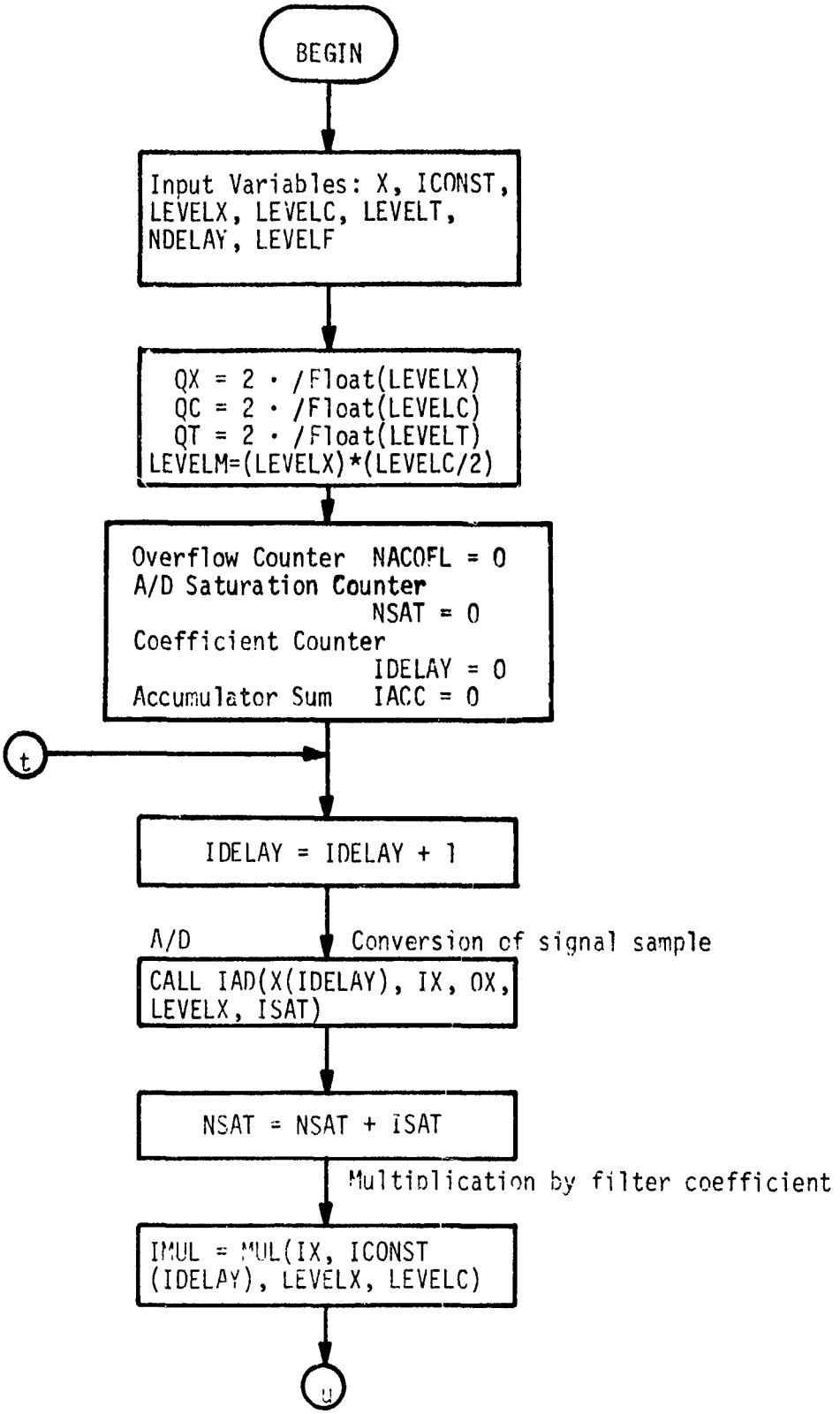


Fig. 3.11 Flow Chart for Subroutine FILTNQ

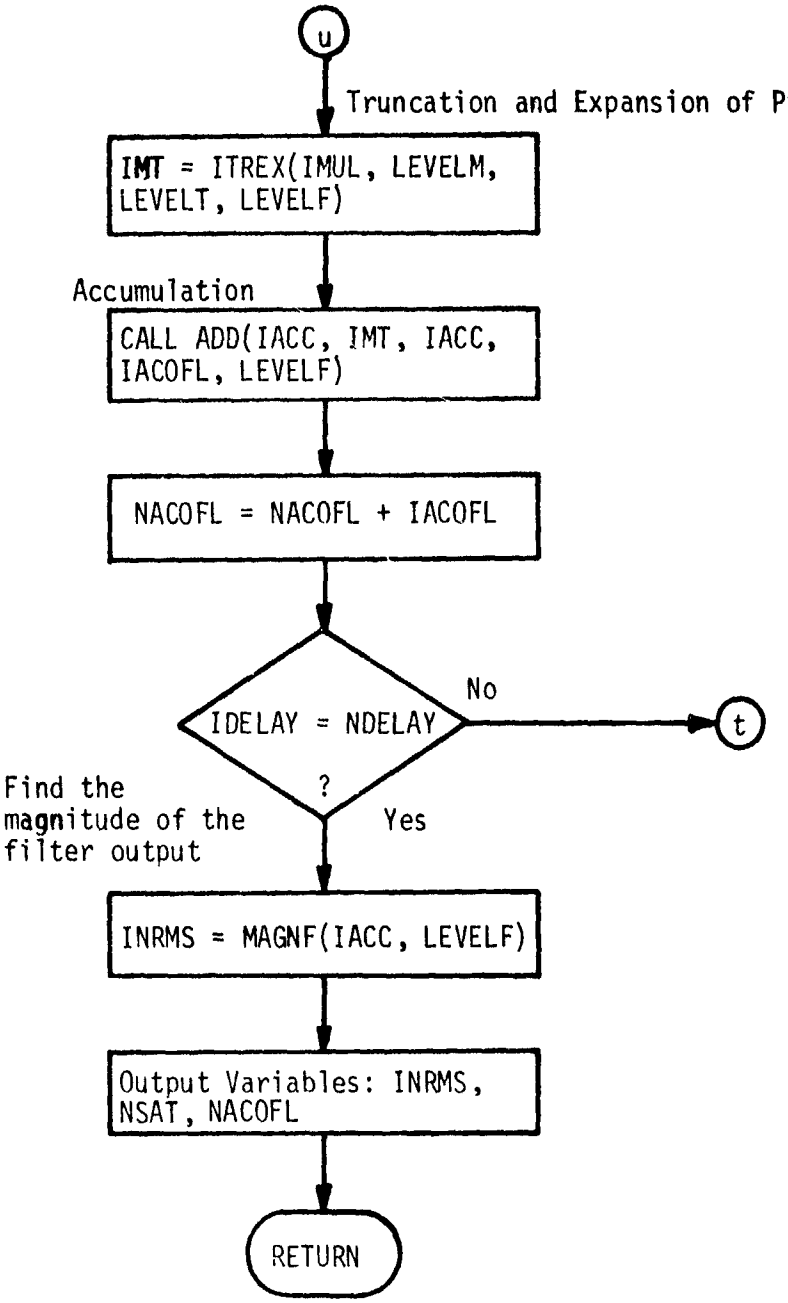


Fig. 3.11 (Continued)

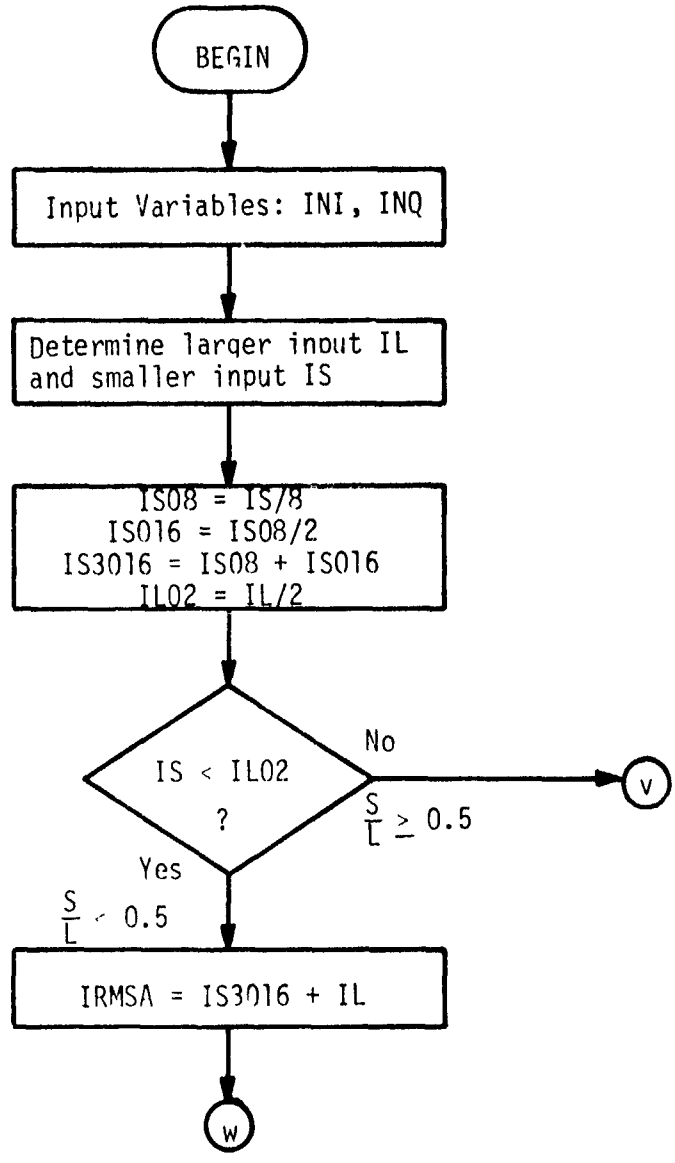


Fig. 3.12 Flow Chart for Function Subprogram IRMSA

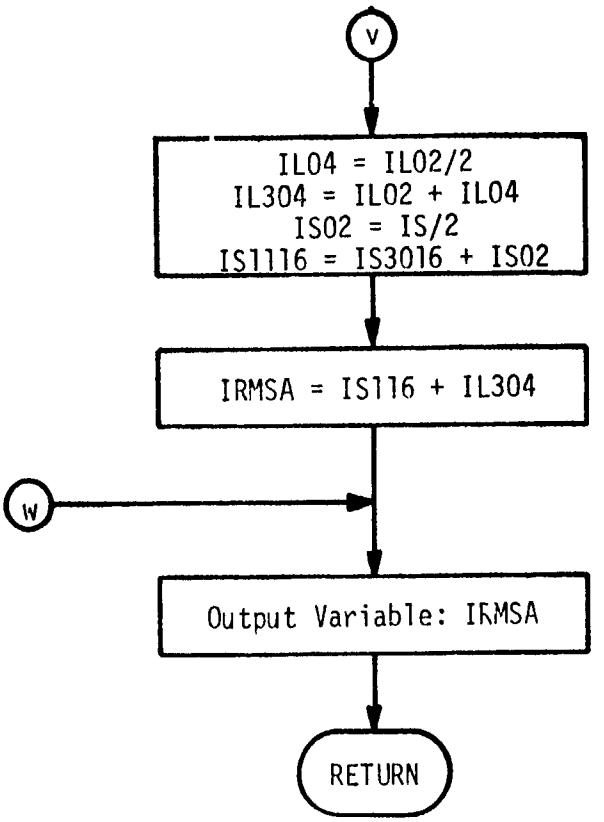


Fig. 3.12 (Continued)

4) The subroutine ANIT is used to set up an initial value when RANDM is called for the first time. There is no need for a special routine for this purpose, it can be incorporated in UPDATQ itself.

The above suggestions were incorporated in a trial program and did result in a considerable saving of computer time for a typical run. They were not made a permanent feature of the fixed-point simulation package but were utilized while constructing the floating-point simulation package.

3.2 DISCUSSION OF SIMULATION RESULTS

The effect of different host computers on the simulation results is discussed briefly in Section 3.2.1. In Section 3.2.2 simulation results obtained with the UNIVAC 1110 system are presented graphically. Finally, a comparison is made between the theoretical results of Chapter 2 and the simulation results presented in this chapter.

3.2.1 Simulation Results from Different Host Computers

The original fixed-point filter simulation program was developed by the UAH Communications Group [14] on an unknown computer system. After that the program was modified to accommodate quantization error study [8] and used to obtain results on a CDC-3600 computer system. On this system the program used a canned routine RANF to generate uniformly distributed random phase values. When the program was made compatible with the UNIVAC 1110 system another canned routine RANDU was used for the same purpose. The starting value and the listing for RANDU appear in Appendix E. Apparantly RANF had a built in starting value but its listing was unavailable, so any differences between the two could not be compared. The effect of this was seen primarily in the number of A/D converter saturations when clutter was added to signal samples. This made for some small but significant changes in the results. Also, it is conjectured that the software algorithms used by the two systems for calculating library functions such as sine, cosine (which are very important for signal generation), log, etc., differ in accuracy and make for some changes in the result. However, the results from the two systems followed the same trends very closely. The minor deviations were observed at

1) extremely low doppler frequencies,
2) doppler frequencies very close to half the pulse repetition frequency, and
3) very small truncated product bit-lengths.

A detailed comparison can be made by comparing the results documented in reference [8] and those in the next section.

3.2.2 Graphical Presentation of Simulation Results

Simulation results were obtained for the sets of nine and five filter coefficients shown in Table 3-2. Results for the nine coefficient case are discussed first.

TABLE 3-2
FILTER COEFFICIENT SETS

9 - Tap	5 - Tap
- 0.159480	- 0.280040
- 0.089430	- 0.163400
- 0.105110	0.888680
- 0.115390	- 0.163400
0.940430	- 0.280040
- 0 115390	
- 0.105110	
- 0.089430	
- 0.159480	

The average error for the hardware RMS and perfect RMS implementations as a function of doppler frequency are plotted in Fig. 3.13, for three different signal amplitudes and no clutter. The ripple effect in the hardware RMS cases is explained by the fact that the MTI digital filter response has local peaks at 750 and 1900 Hz and local minima at 1500 and 2400 Hz. The ripple in the perfect RMS case does not closely correlate with the filter response curve. Also, the perfect RMS has negative average errors for some cases while the hardware RMS has positive average error for all cases. The increase in the magnitude of the average error for extremely low frequencies in the stopband has not been explained.

Fig. 3.14 presents the variance results as a function of frequency for the same cases mentioned above. The hardware RMS cases show much more variation than their counterparts in Fig. 3.13. There is a strong amplitude dependence for the hardware RMS cases while the perfect RMS cases do not show such a strong amplitude dependence. Note that the perfect RMS cases are closely grouped together.

The minimum, maximum and average errors are shown in Figs. 3.15 and 3.16 for signal amplitudes of 0.025 and 0.413 volts respectively, with no clutter and for the hardware RMS algorithm. Note that the middle of the extreme values is in general agreement with the average value. This suggests a symmetrical distribution about the mean for the error probability density function.

Next, the average error and variance as functions of truncated product bit-length MT for the hardware algorithm with a doppler frequency of 1500 Hz, are shown in Figs. 3.17 and 3.18. The cases for five different signal amplitudes without clutter are shown. It is to be expected that

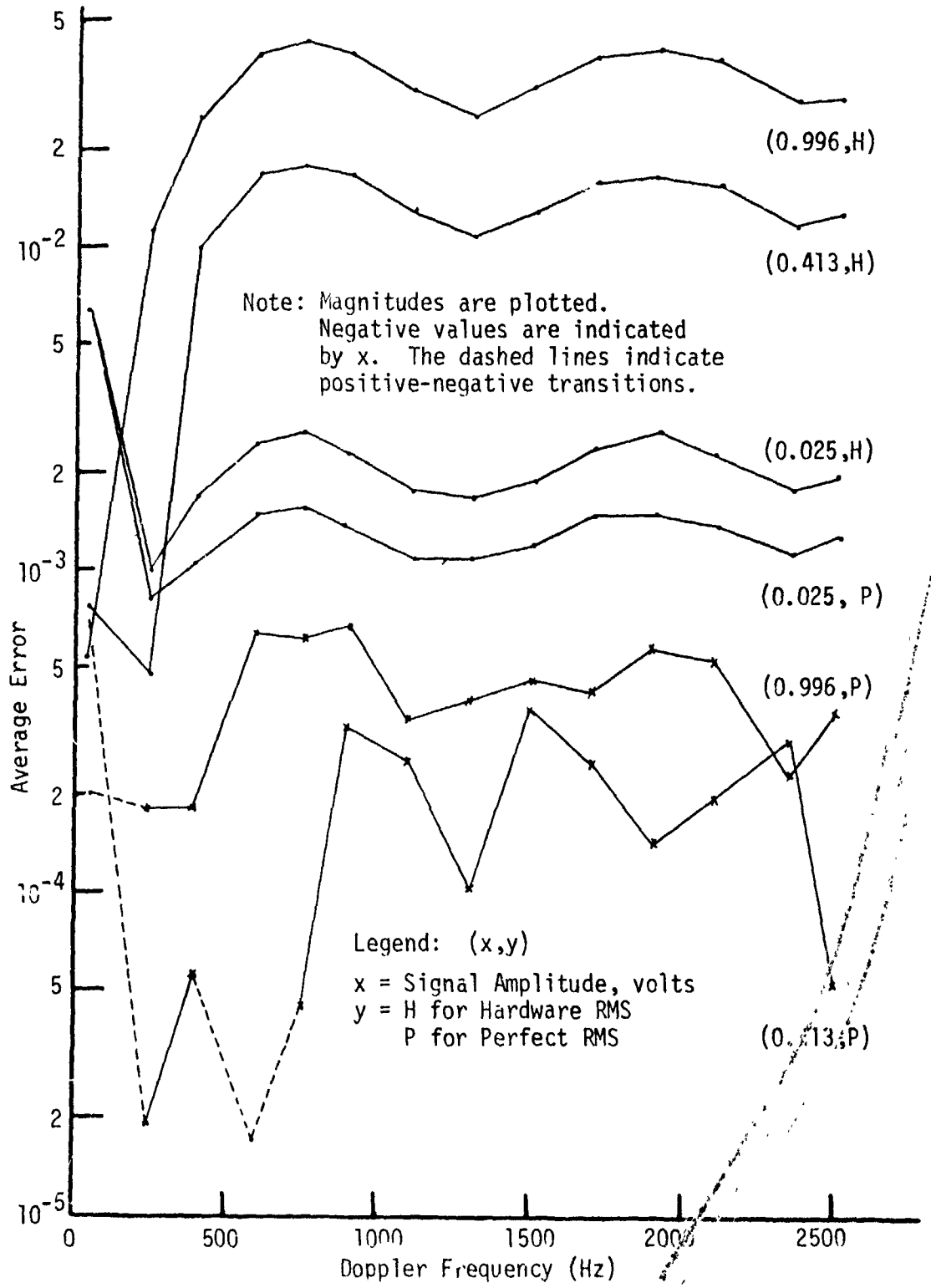


Fig. 3.13 Average Error as Function of Doppler Frequency (9-Tap Fixed-Point Simulation Results, No Clutter, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

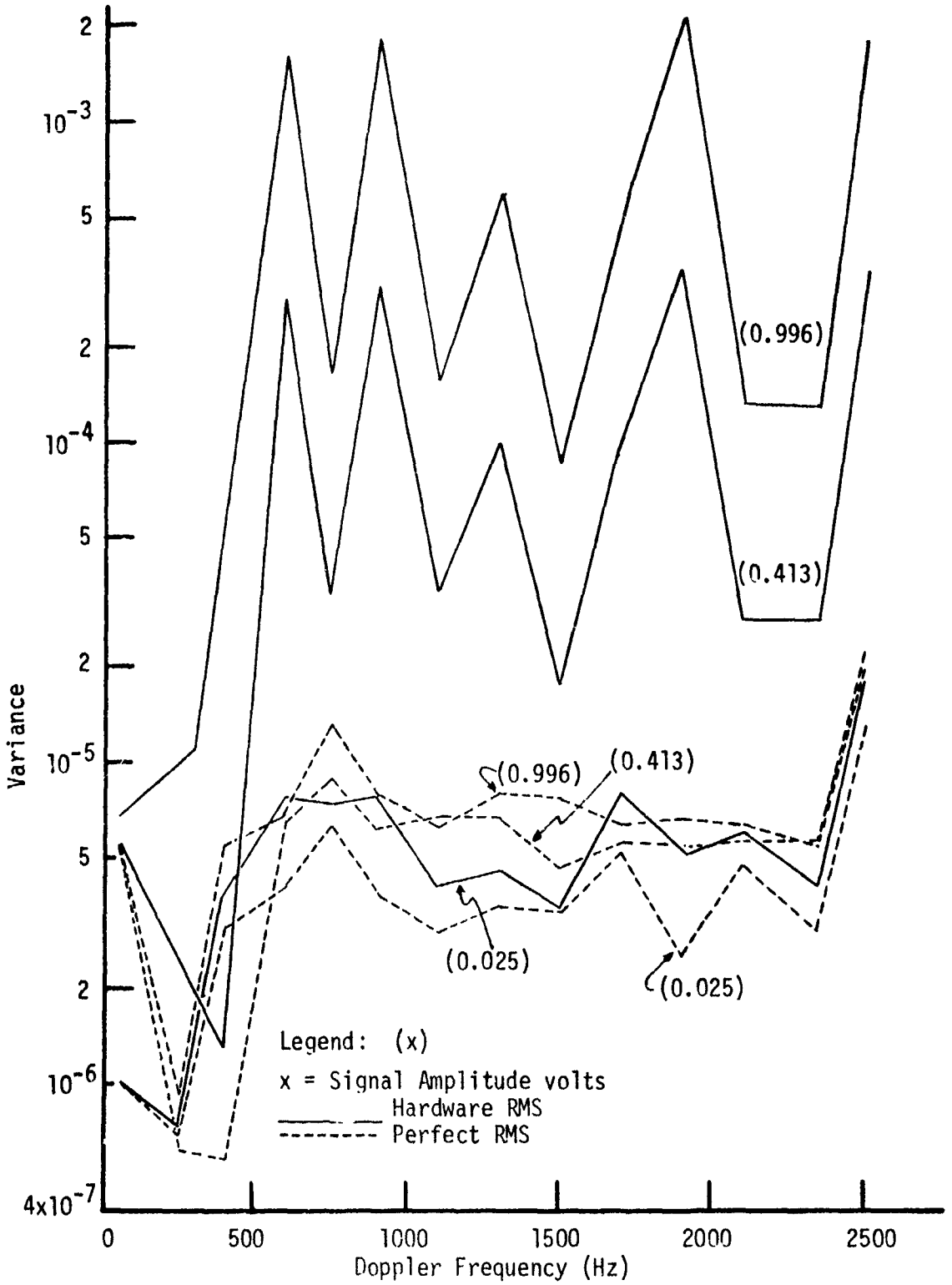


Fig. 3.14 Variance as Function of Doppler Frequency (9-Tap Fixed-Point Simulation Results, No Clutter, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

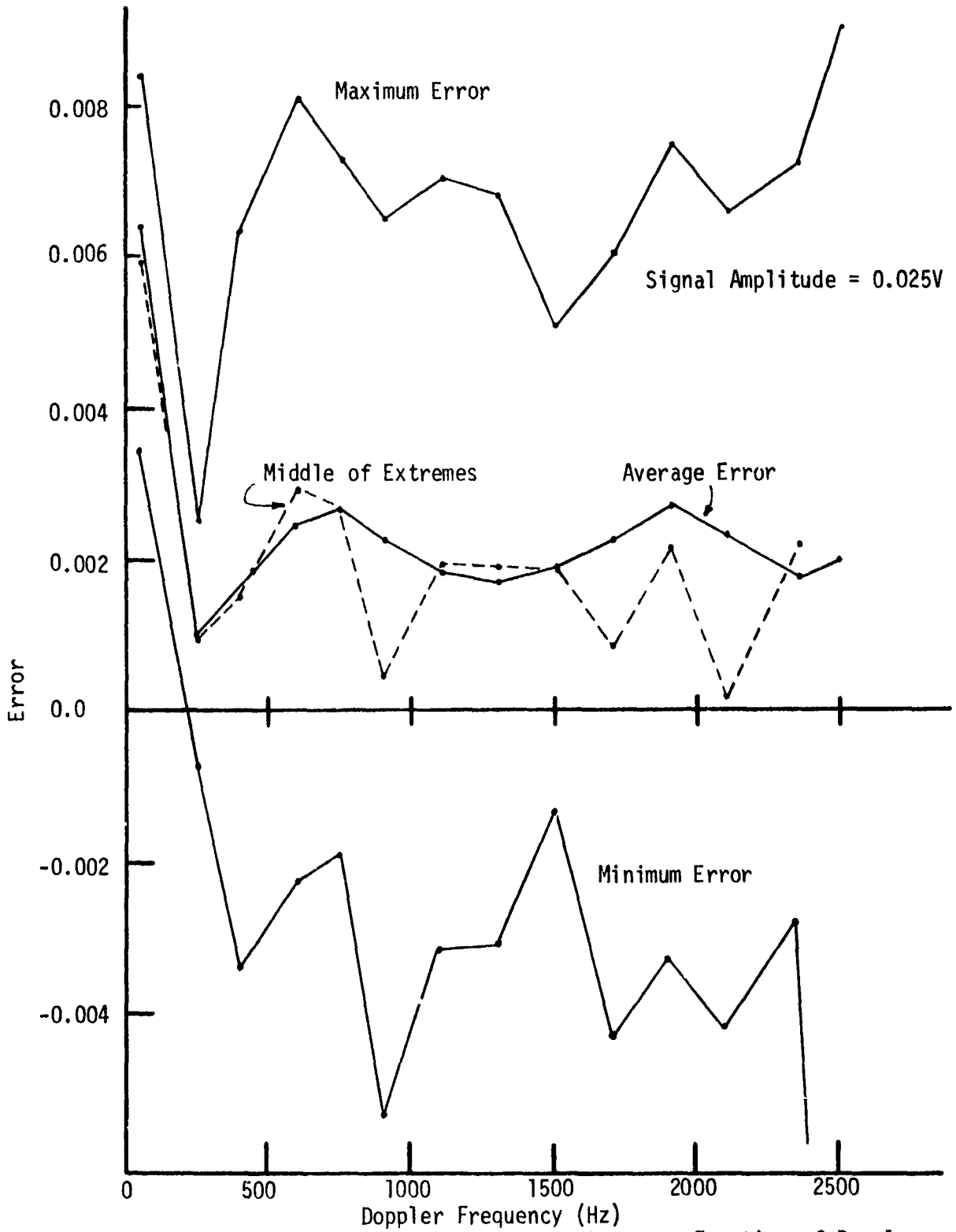


Fig. 3.15 Hardware RMS Extreme and Average Error as Function of Doppler Frequency (9-Tap Fixed-Point Simulation Results, No Clutter, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

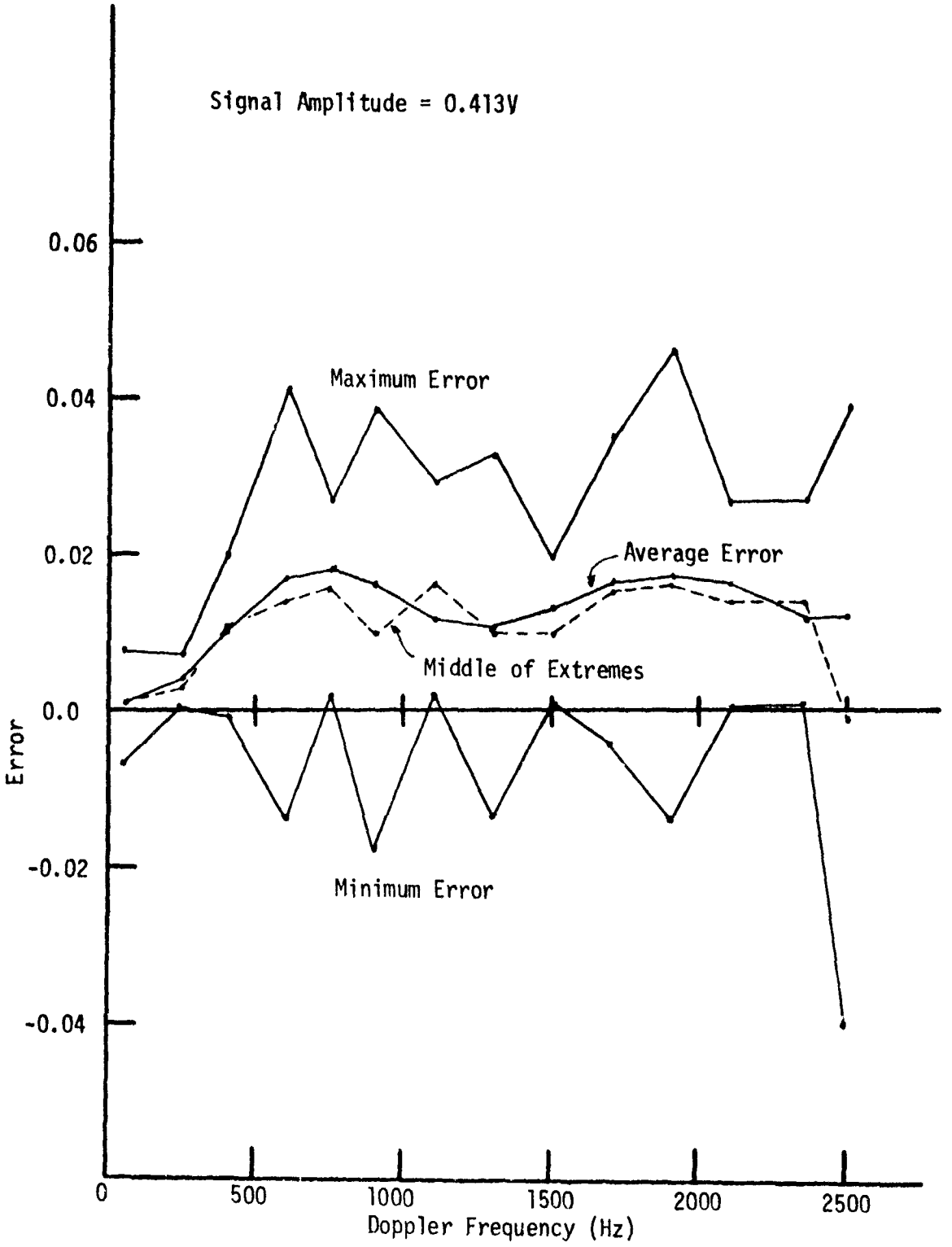


Fig. 3.16 Hardware RMS Extreme and Average Error as Function of Doppler Frequency (9-Tap Fixed-Point Simulation Results, No Clutter, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

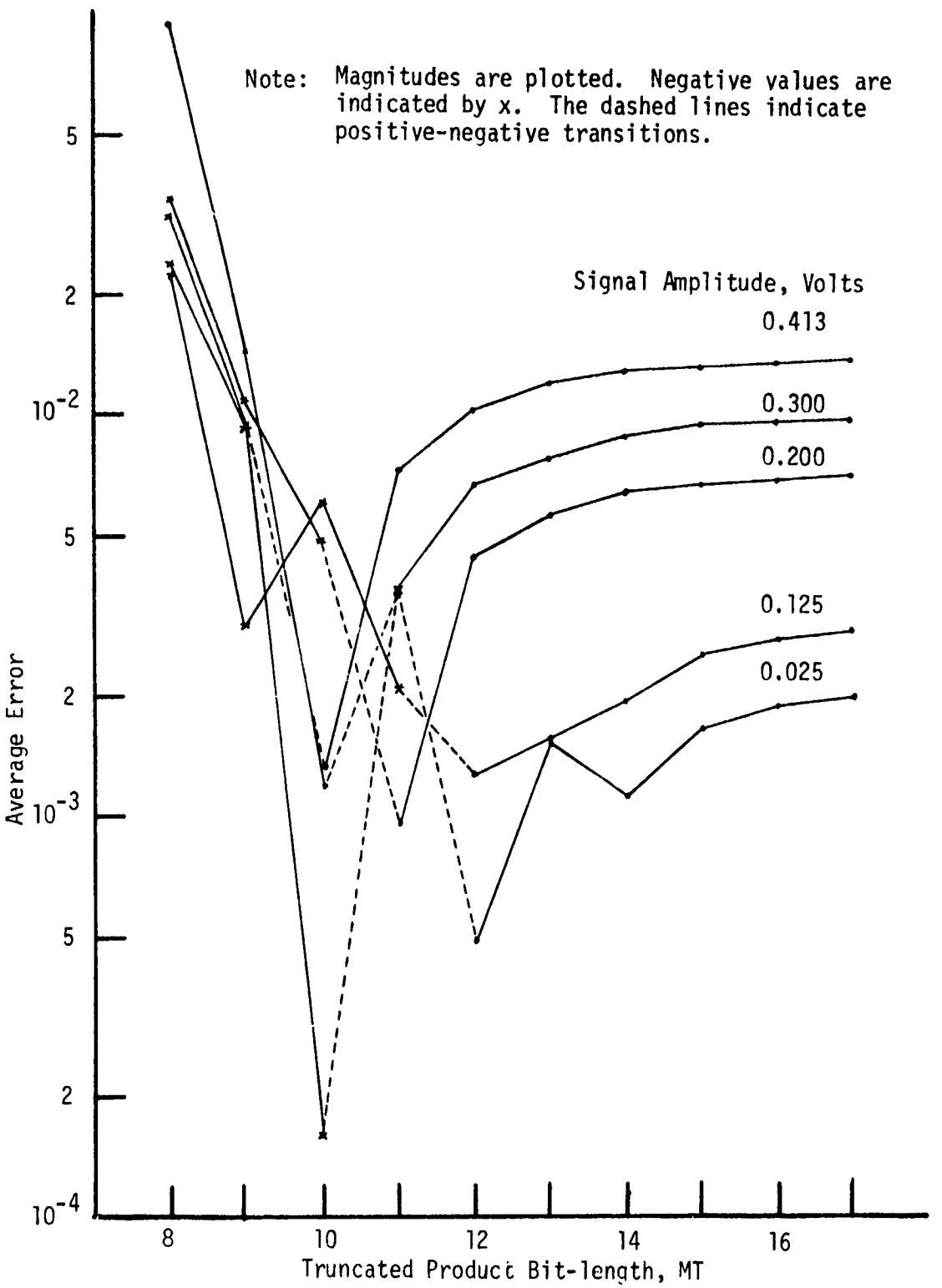


Fig. 3.17 Hardware RMS Average Error as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

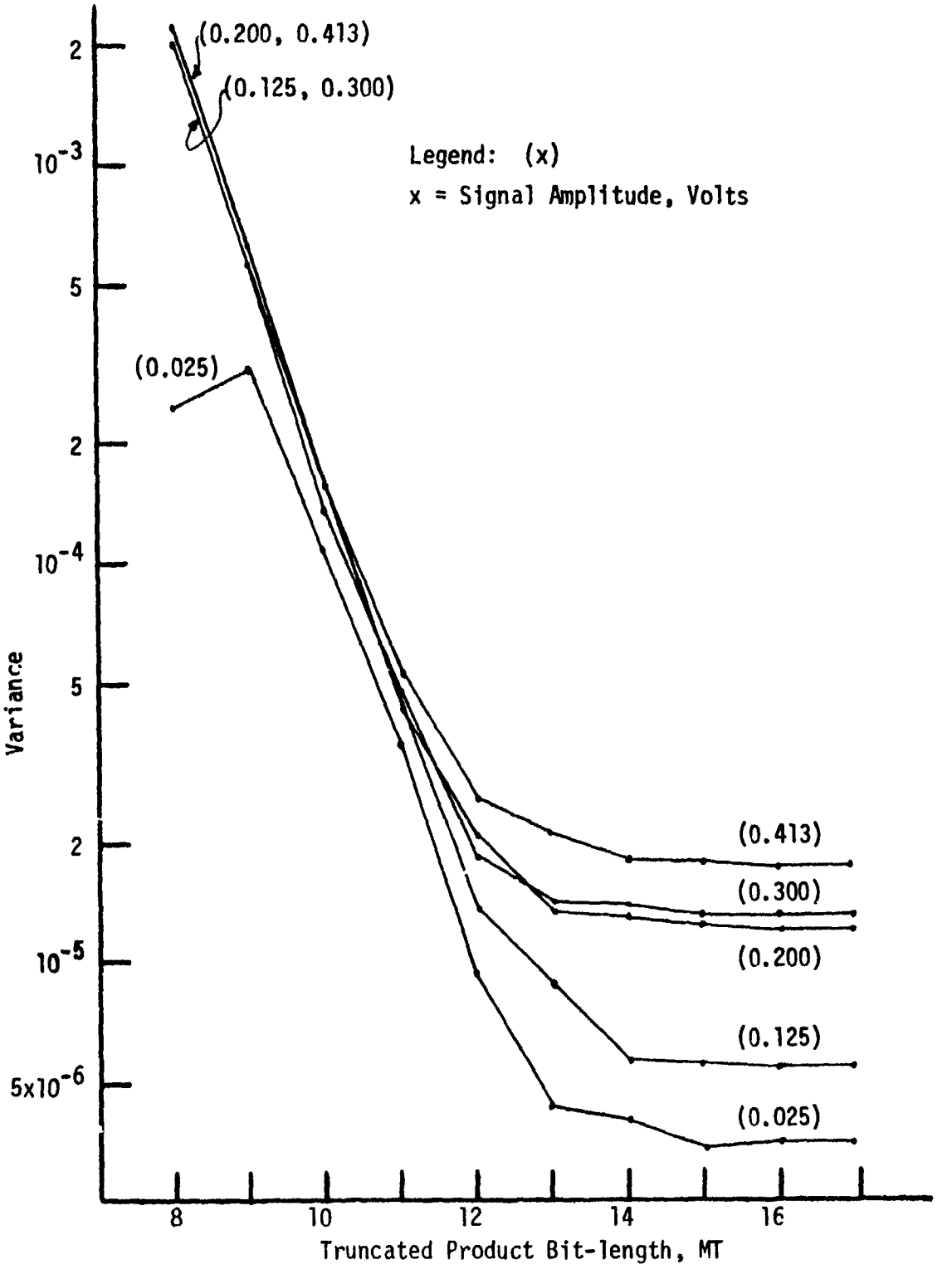


Fig. 3.18 Hardware RMS Variance as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

with increasing amplitude the average error and especially the variance would increase since the predominant contributor to these is the RMS unit. The output of the RMS unit increases with signal amplitude and the error introduced by it is a random multiplicative error which increases with the signal amplitude. Note that both positive and negative values of average error are present and that break points are between product lengths of 9 and 12 bits. For each signal amplitude in Figs. 3.17 and 3.18 the no clutter and extreme clutter (as explained below) cases are presented in Fig. 3.19 to 3.28. Alternately they give the average error and variance for the hardware RMS case. Five different signal-to-clutter power ratios were tried, viz., +3.0 dB, 0.0 dB, -3.0 dB, -10.0 dB and -20.0 dB. For each signal amplitude the most negative ratio that did not show any A/D saturations or a very minimal number of saturations was chosen for presentation. Results from all the other signal-to-clutter ratios higher than this extreme case were bounded between the no clutter and extreme clutter cases. It needs to be borne in mind that a total of 45,000 signal plus clutter samples are used in a simulation with 500 dwells, 9 coefficients and 5 residues. From Fig. 3.19 to Fig. 3.28 it is difficult to generalize about the behavior of the average error curves as a function of signal amplitude and the relation between the no clutter and extreme clutter cases. But, the variance curves show a very distinct trend. For each signal amplitude the variance for the extreme clutter case is higher than the no clutter case. This seems to be due to the added contribution of the clutter variance to the RMS output and other quantization noise variances. The extent of deviation from the no clutter case is approximately of the same order of magnitude except in the cases of signal amplitudes 0.200 and 0.413. But, both these cases exhibit some amount of A/D saturation which is fairly minimal and the curves do follow the same general trend.

For the case of MT=17 the effect of the signal amplitude on hardware RMS algorithm average error and variance are shown in Figs. 3.29 and 3.30. Each figure has the no clutter curve as a reference and the excursions due to different signal-to-clutter ratios are indicated. It is seen that when clutter does not cause excessive A/D saturations, the clutter cases are grouped together and their deviation from the no clutter case decreases as the signal amplitude increases.

Next, the results for the five coefficient case are presented. Figures 3.31 and 3.32 present data similar to the Figures 3.17 and 3.18 which are for the nine coefficient case. The hardware RMS algorithm average error and variance increase with increasing signal amplitude just like the nine coefficient case. The average error again shows both positive and negative values and the break points are between product lengths of 9 and 13. The five coefficient case values are about an order of magnitude larger than the nine coefficient case. This is to be expected since the number of residues in this case is nine as opposed to five in the previous case and this means an increased number of RMS outputs integrated. Also, 1500 Hz represents a relative minimum in the frequency response of the nine coefficient filter whereas for the five coefficient filter it is not a relative minimum and has a higher transfer.

Again Figures 3.33 to 3.42 present results in a manner similar to that of Figures 3.19 to 3.28 of the nine coefficient case. It is diffi-

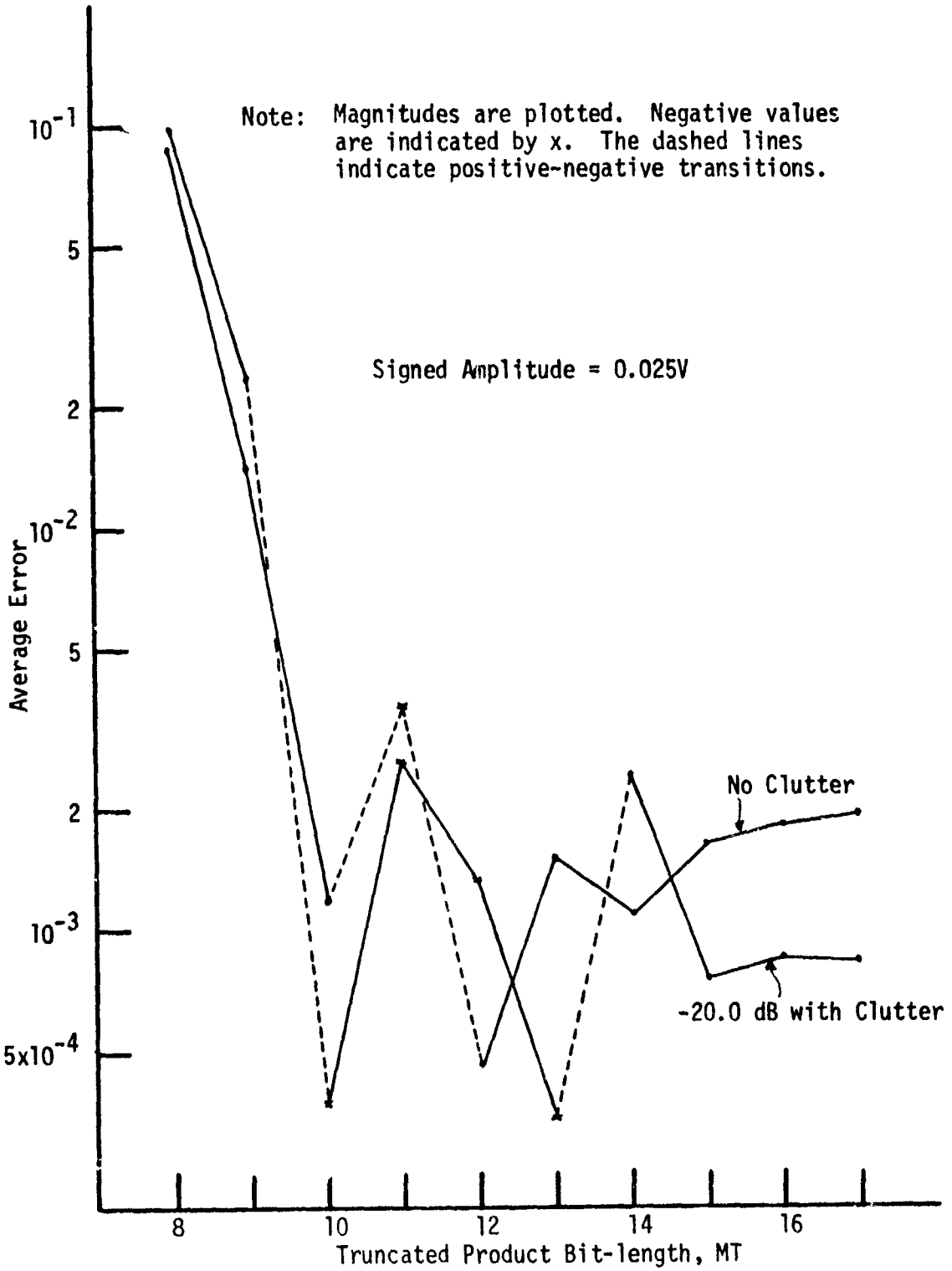


Fig. 3.19 Hardware RMS Average Error as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

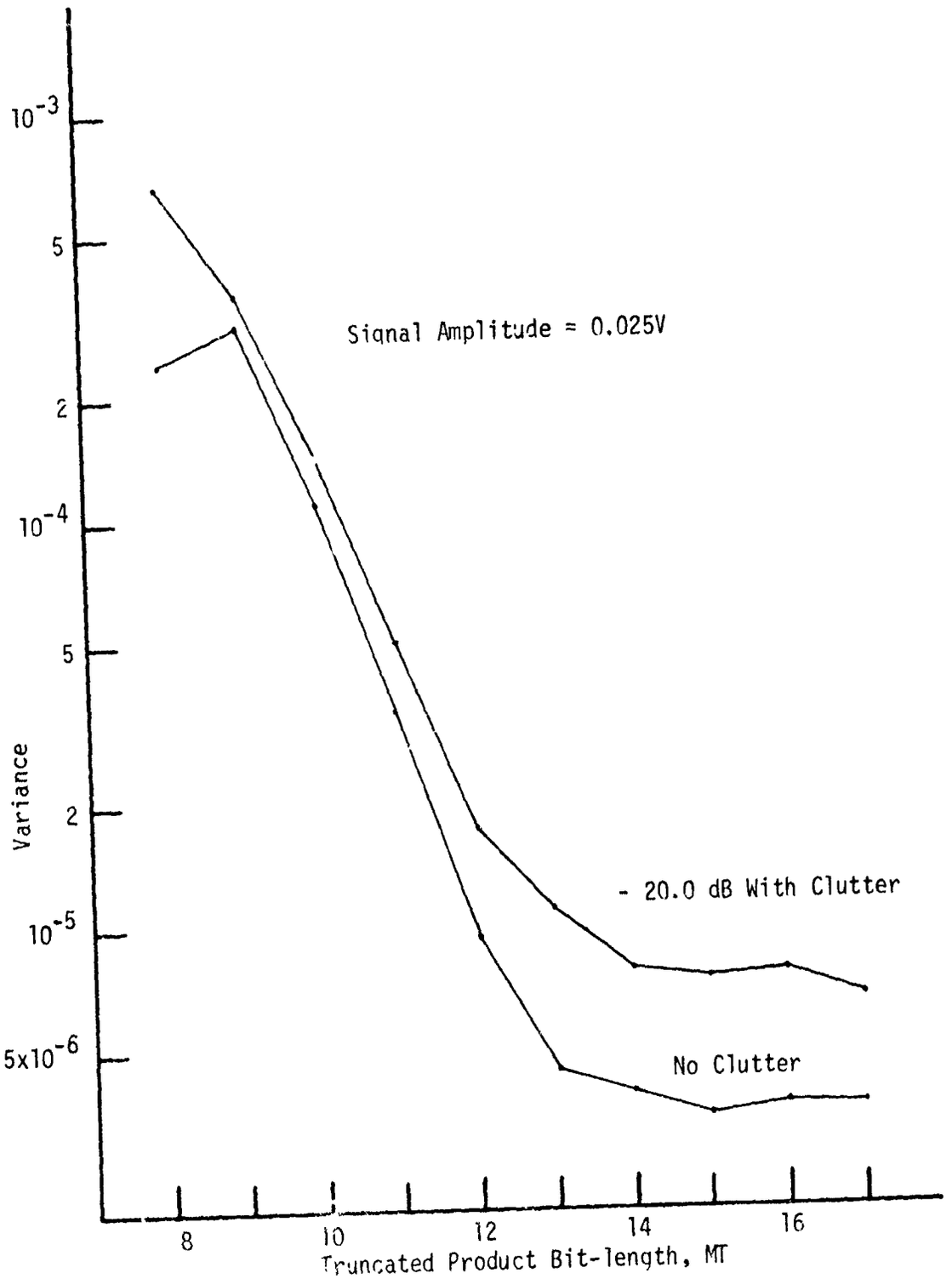


Fig. 3.20 Hardware RMS Variance as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

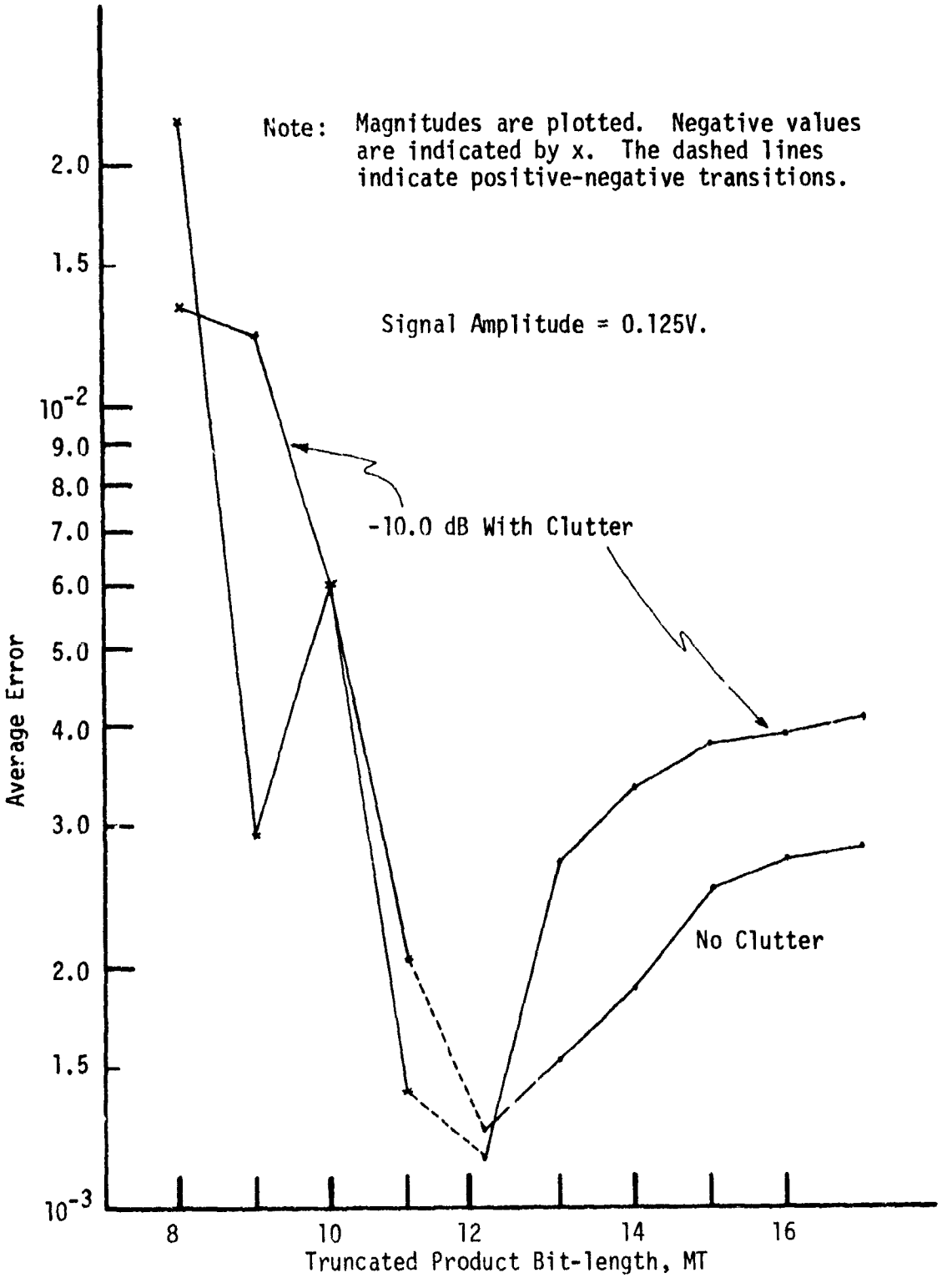


Fig. 3.21 Hardware RMS Average Error as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

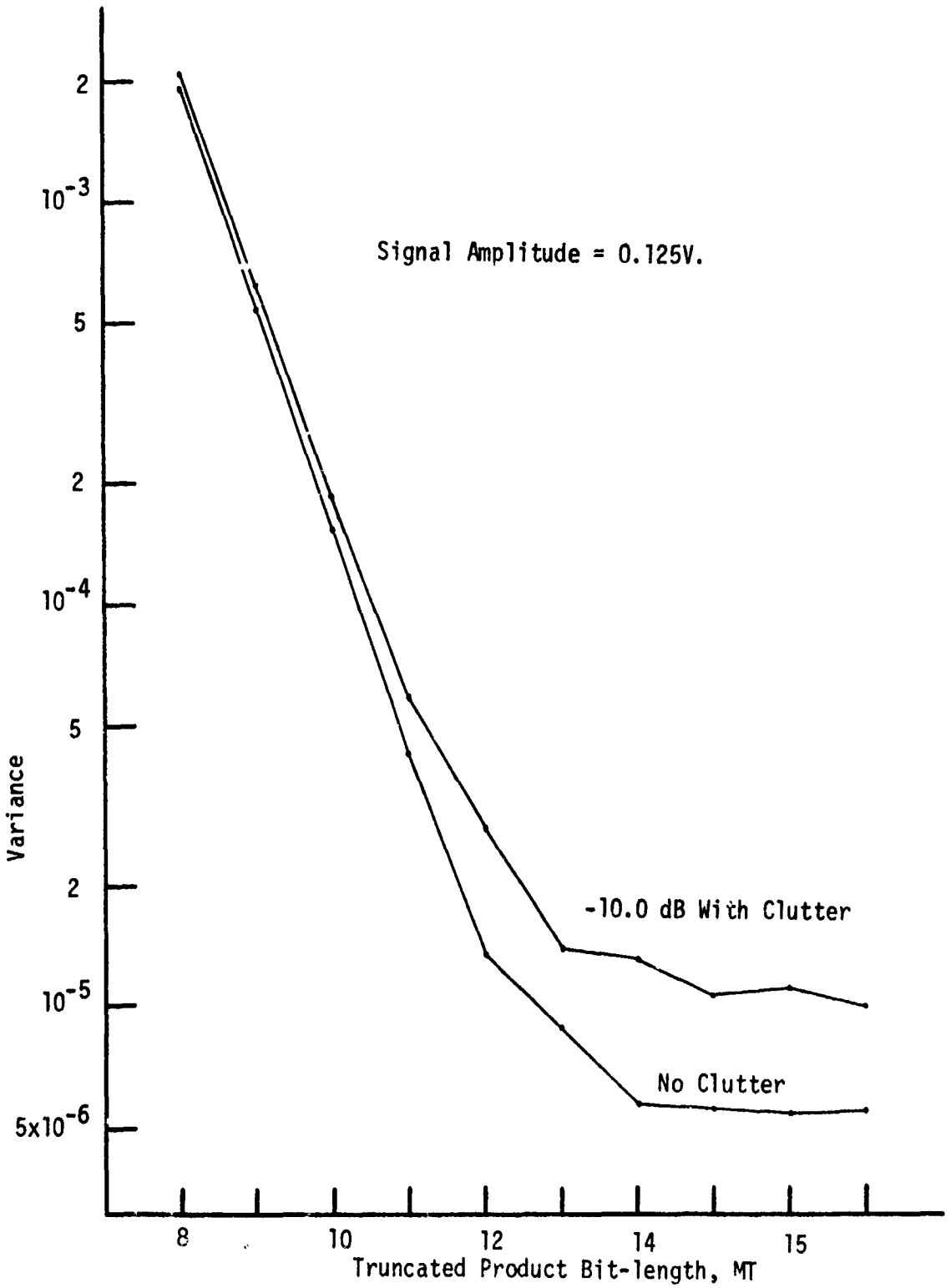


Fig. 3.22 Hardware RMS Variance as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

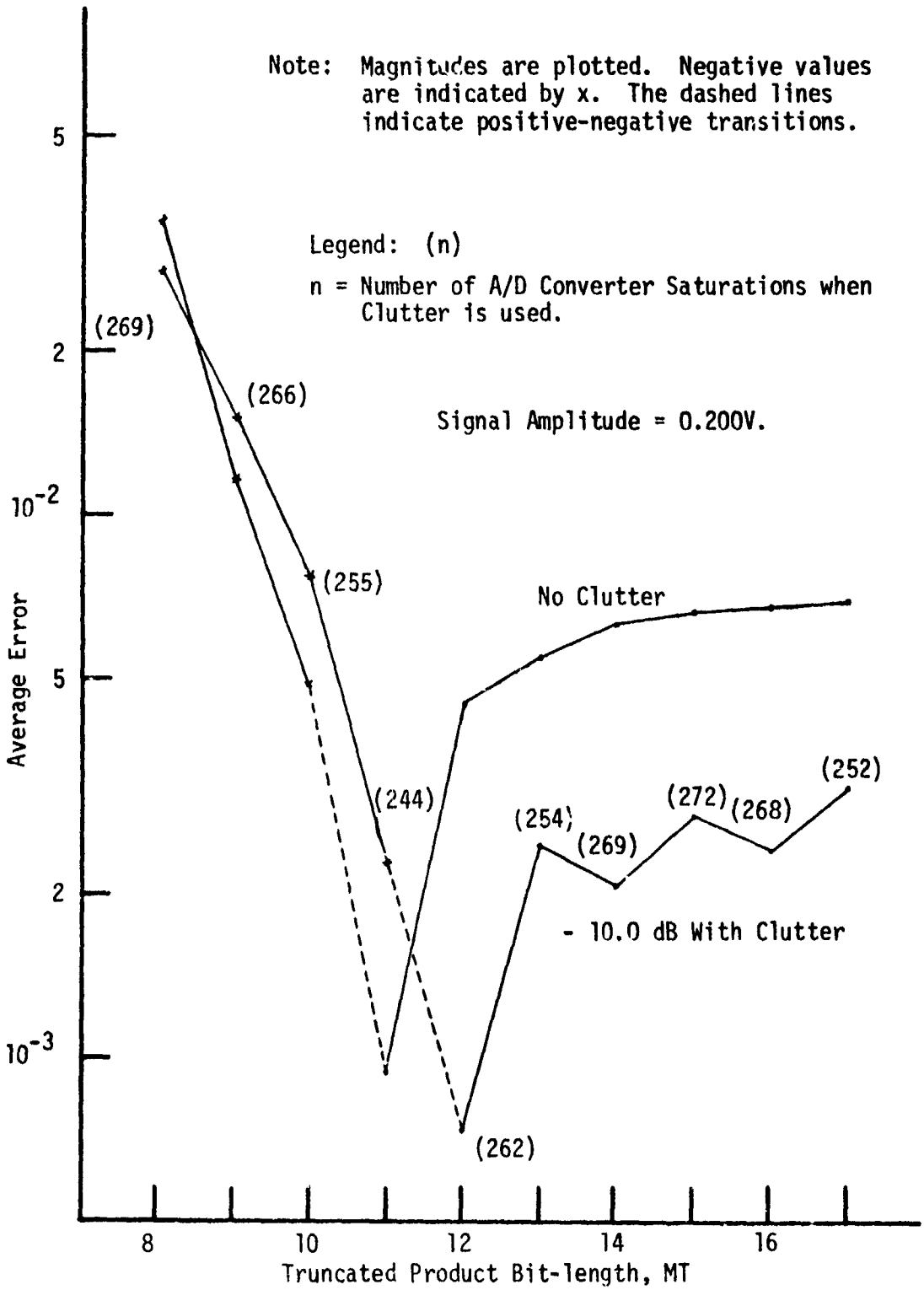


Fig. 3.23 Hardware RMS Average Error as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

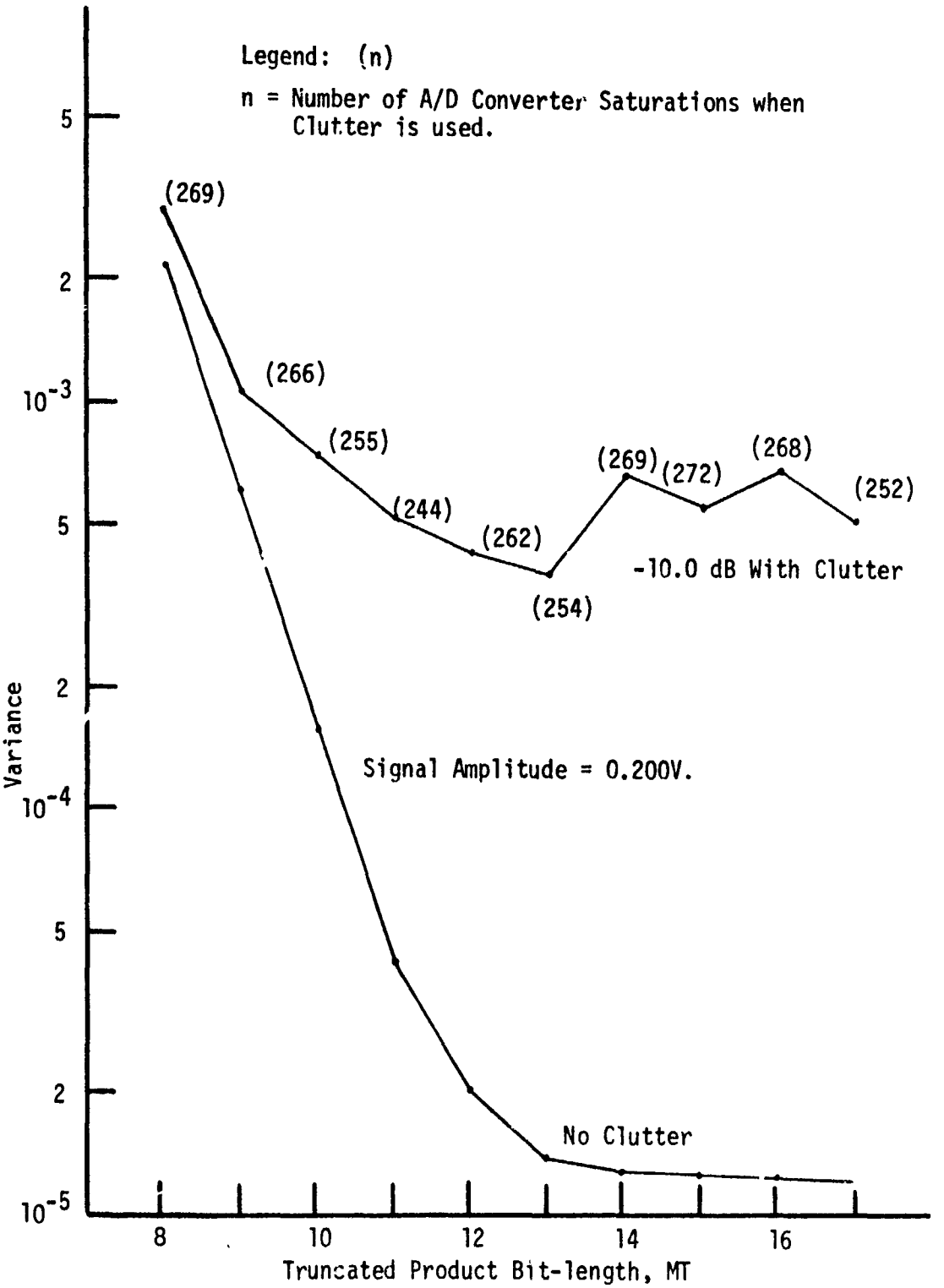


Fig. 3.24 Hardware RMS Variance as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

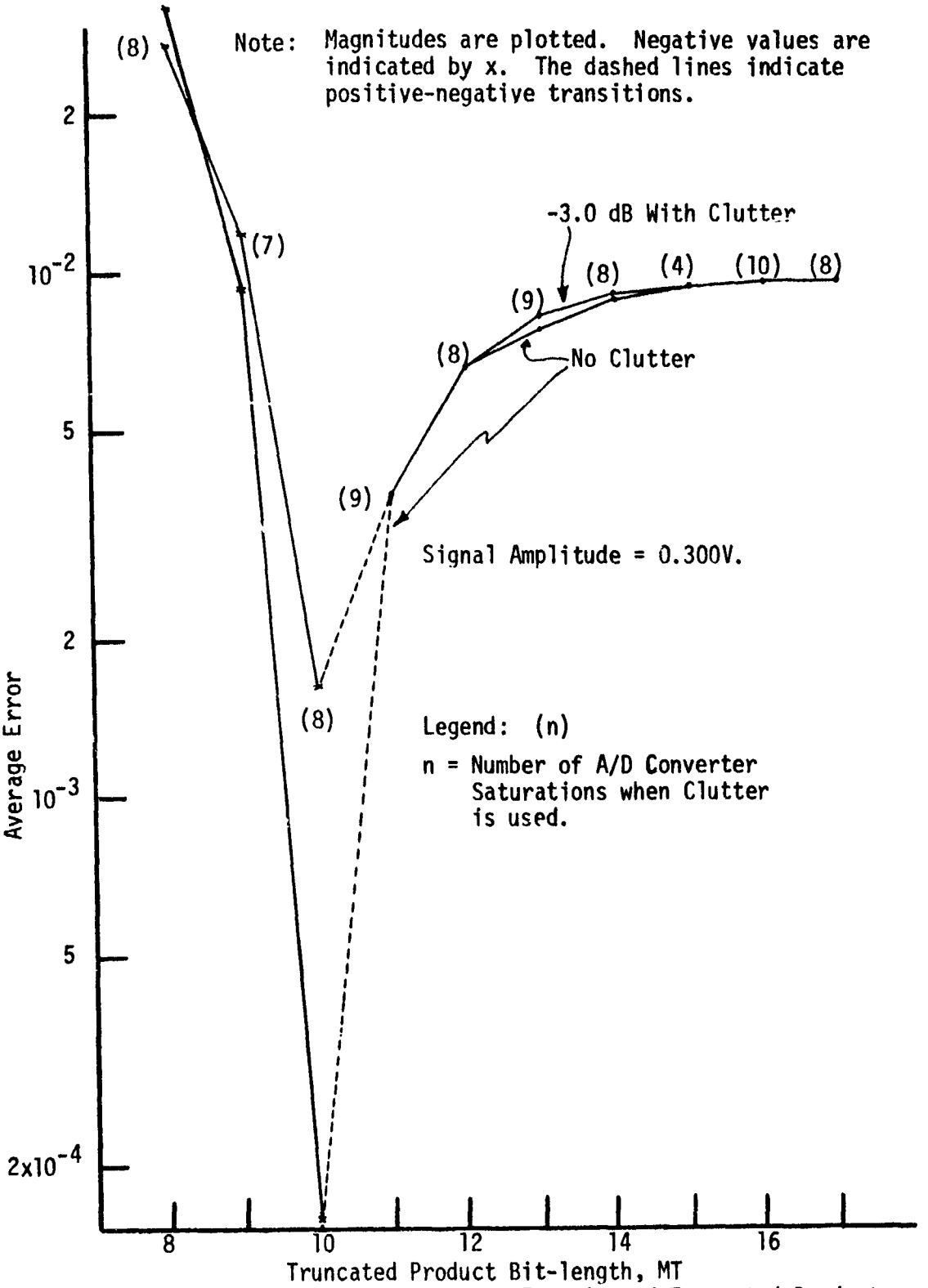


Fig. 3.25 Hardware RMS Average Error as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

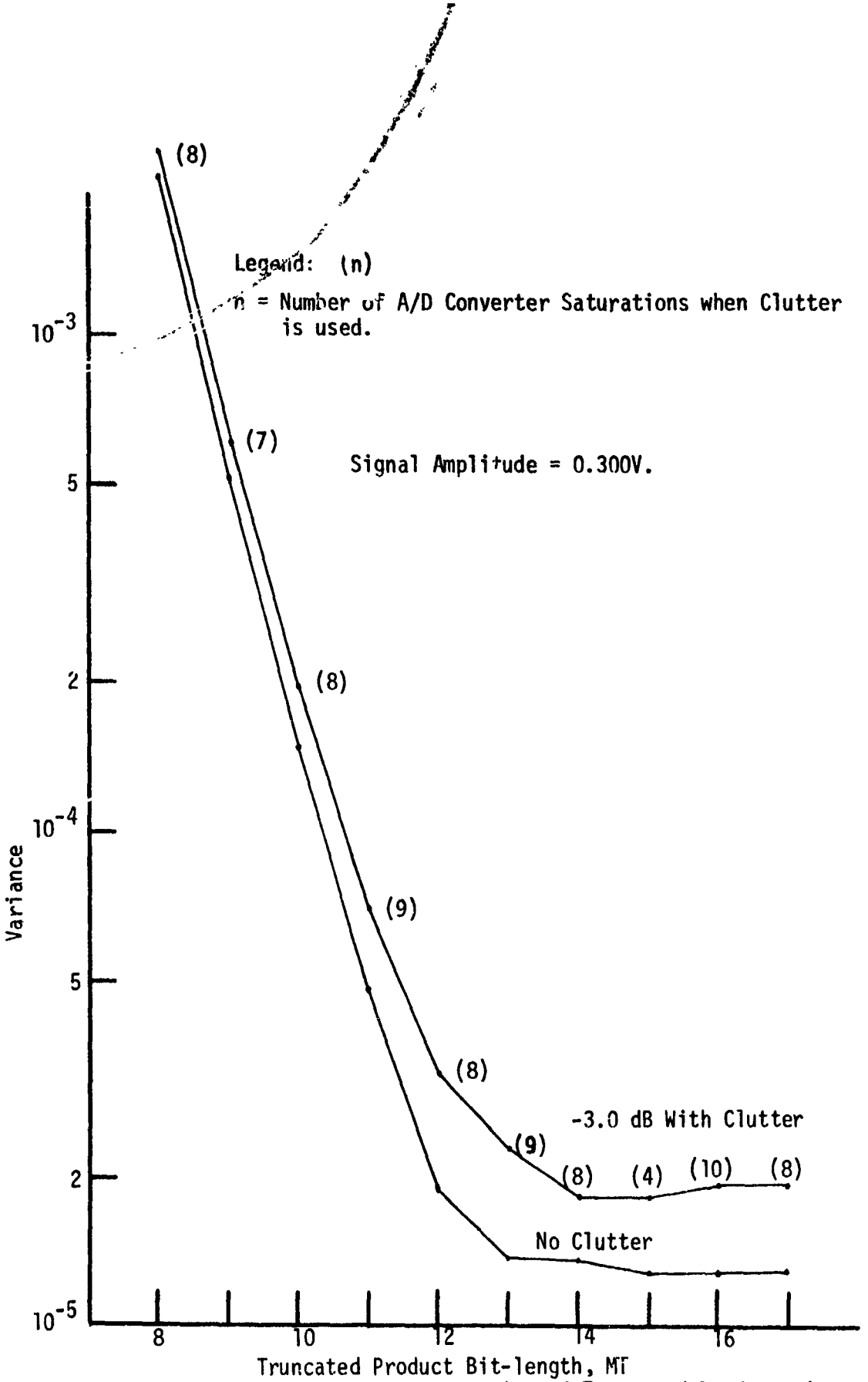


Fig. 3.26 Hardware RMS Variance as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

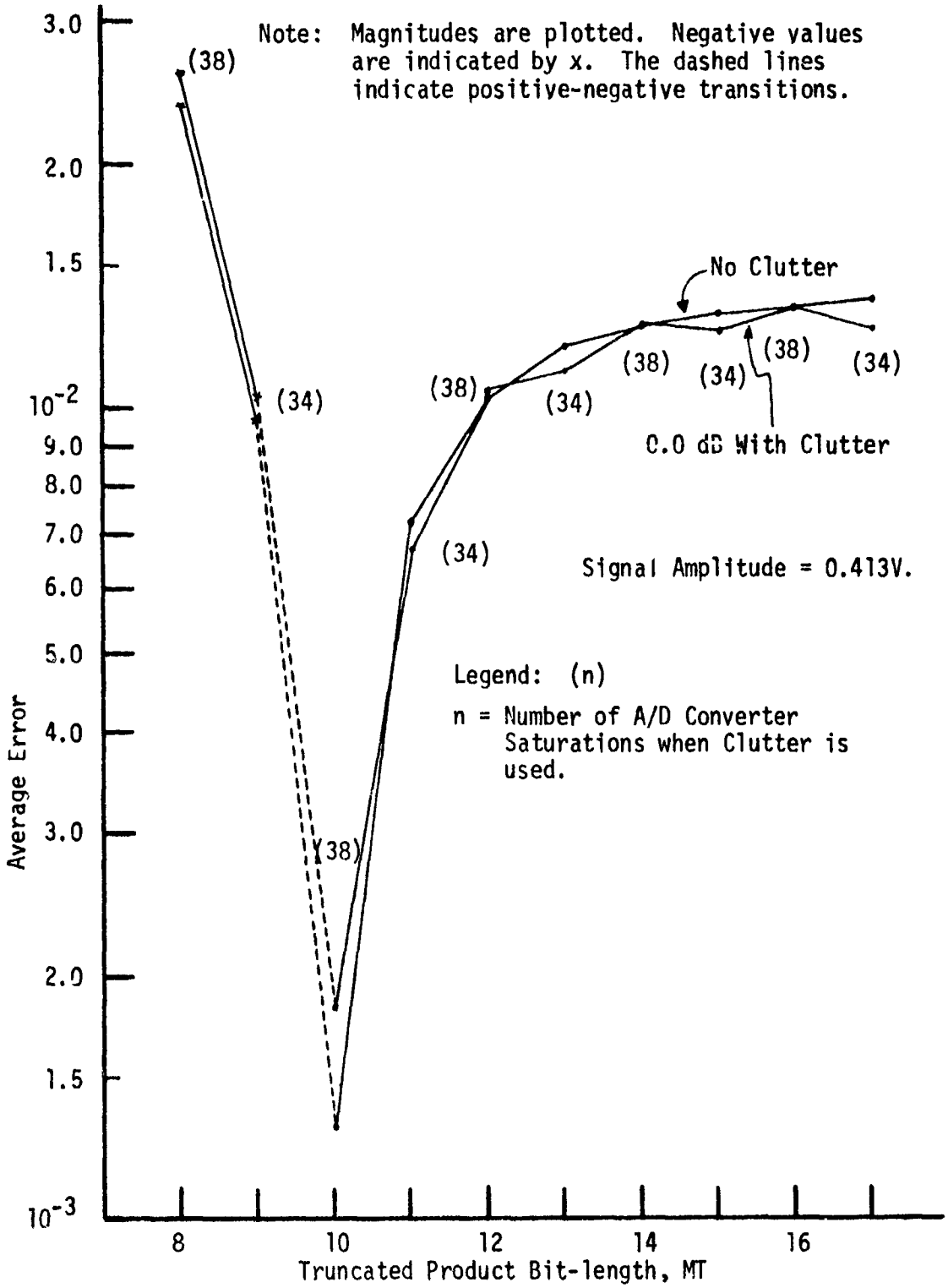


Fig. 3.27 Hardware RMS Average Error as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 hz, MX = MC = 9, MF = ME = 20, MS = 24)

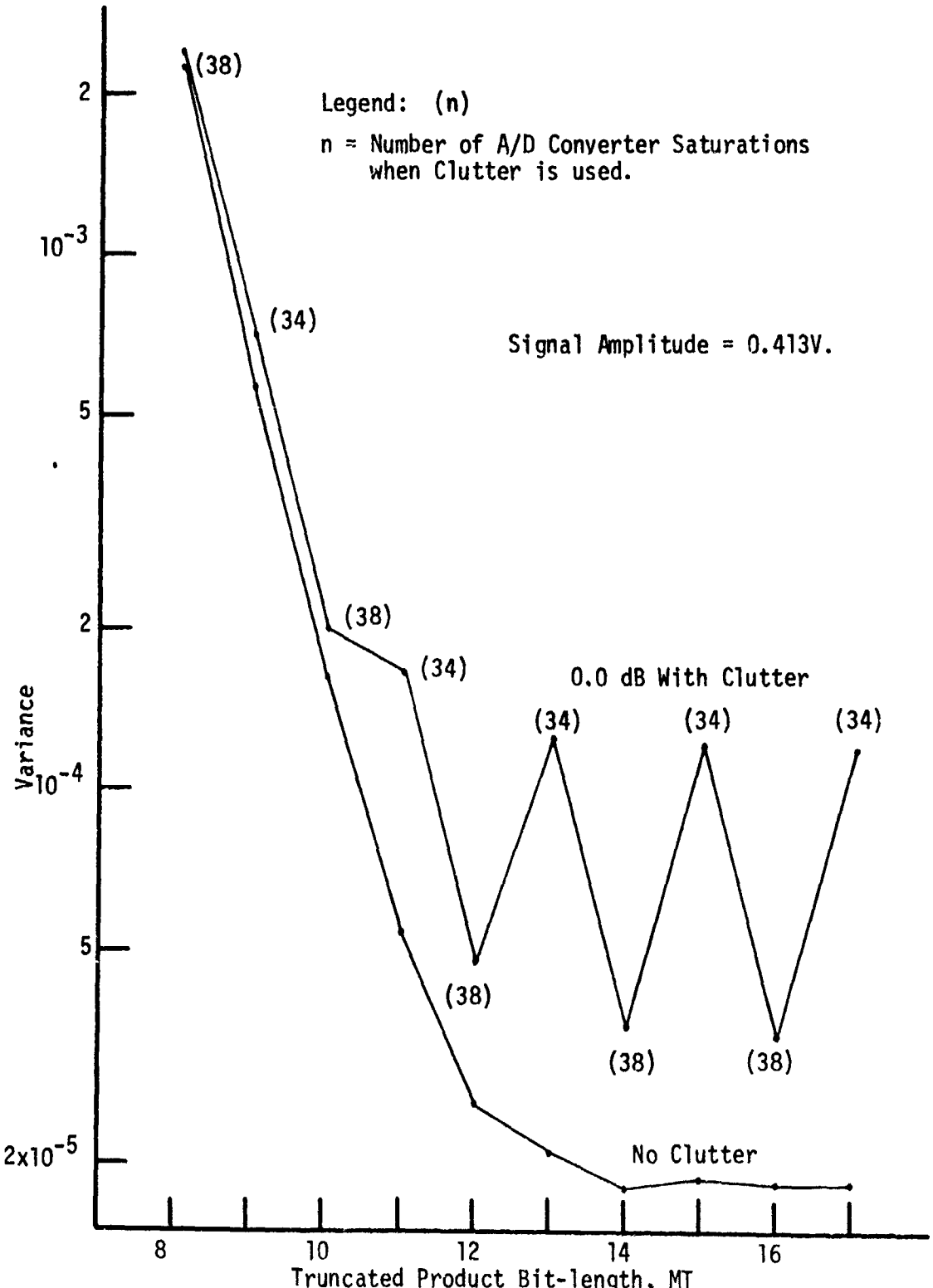


Fig. 3.28 Hardware RMS Variance as Function of Truncated Product Bit-length (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

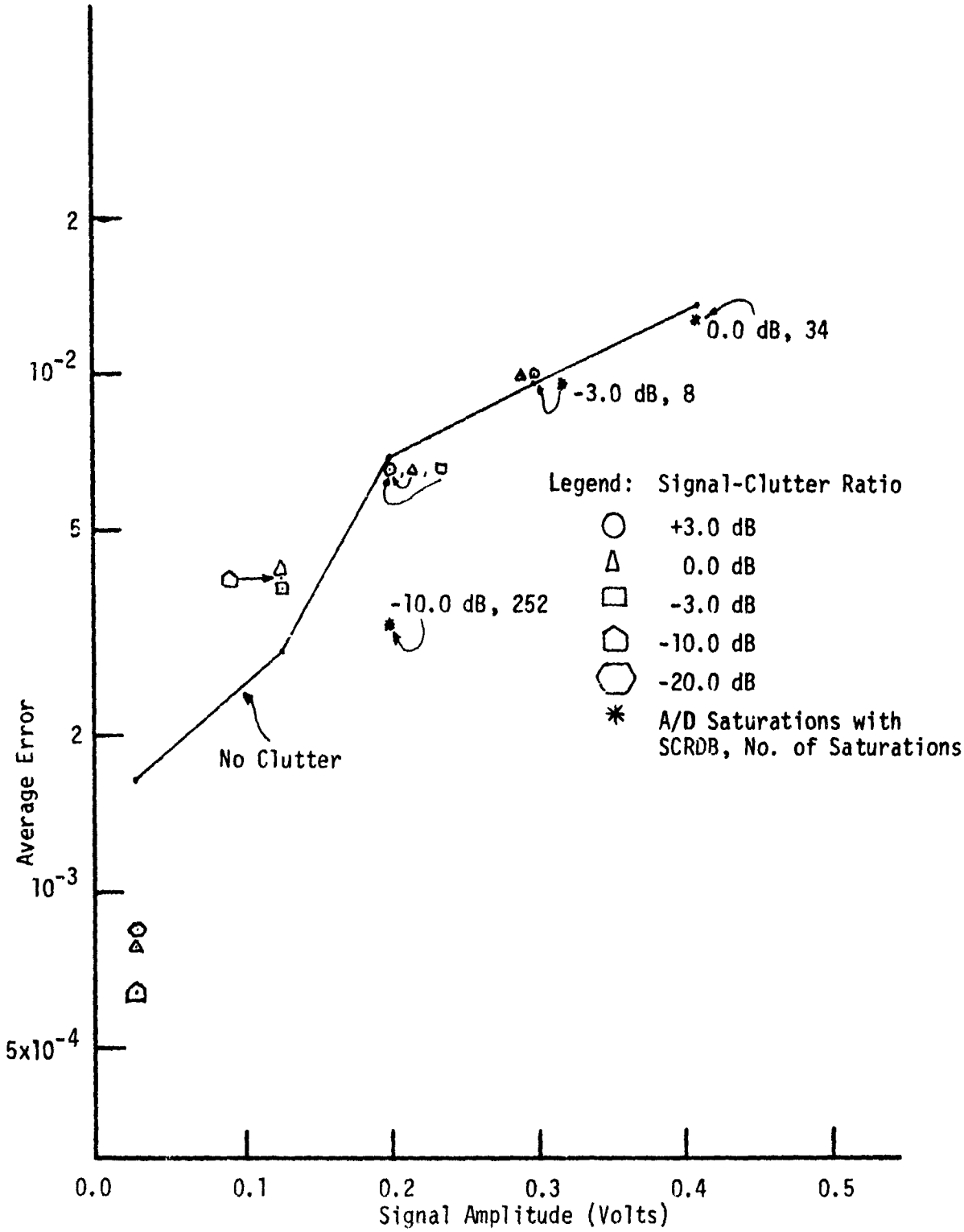


Fig. 3.29 Hardware RMS Average Error as Function of Signal Amplitude (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

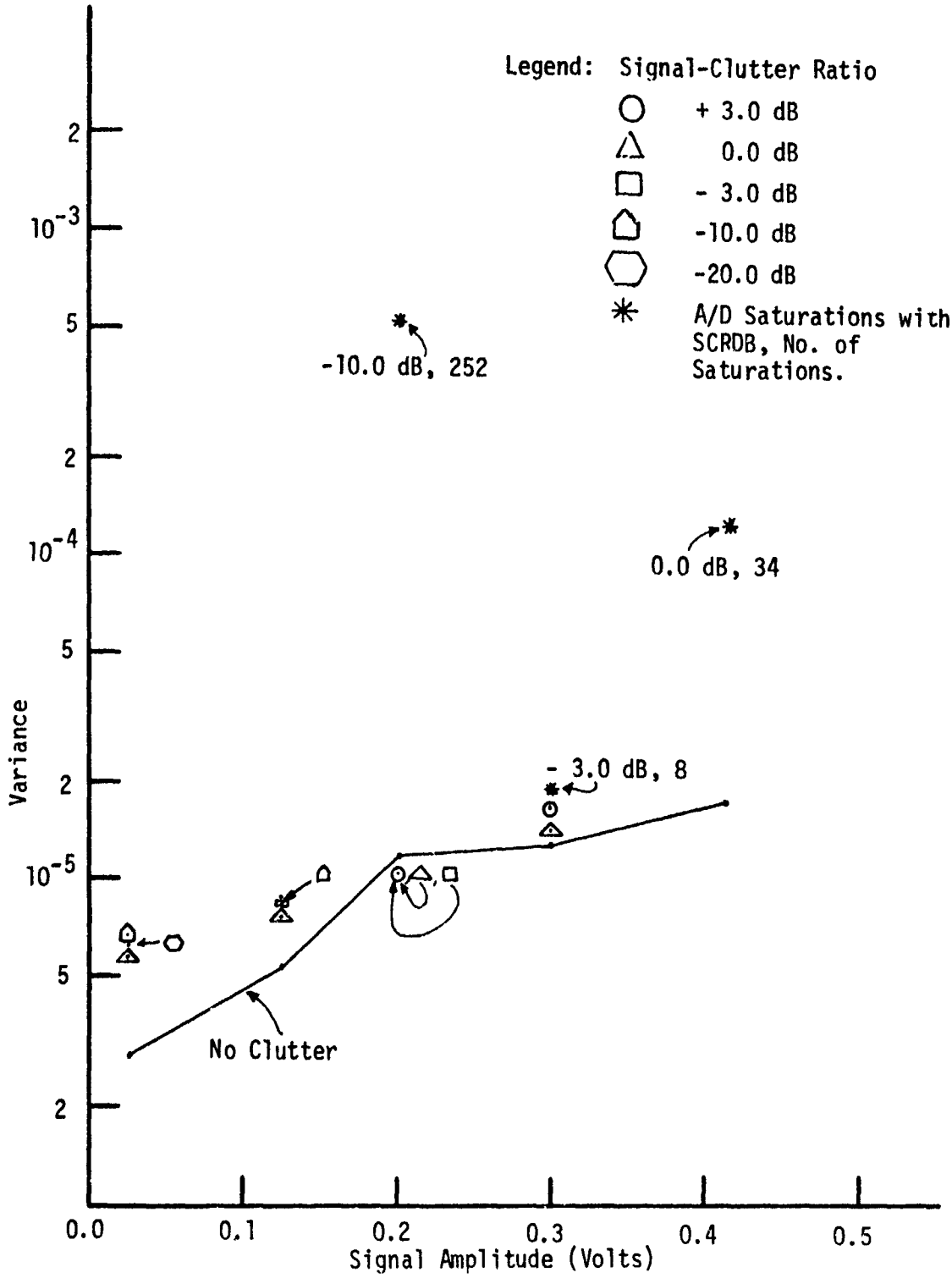


Fig. 3.30 Hardware RMS Variance as Function of Signal Amplitude (9-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

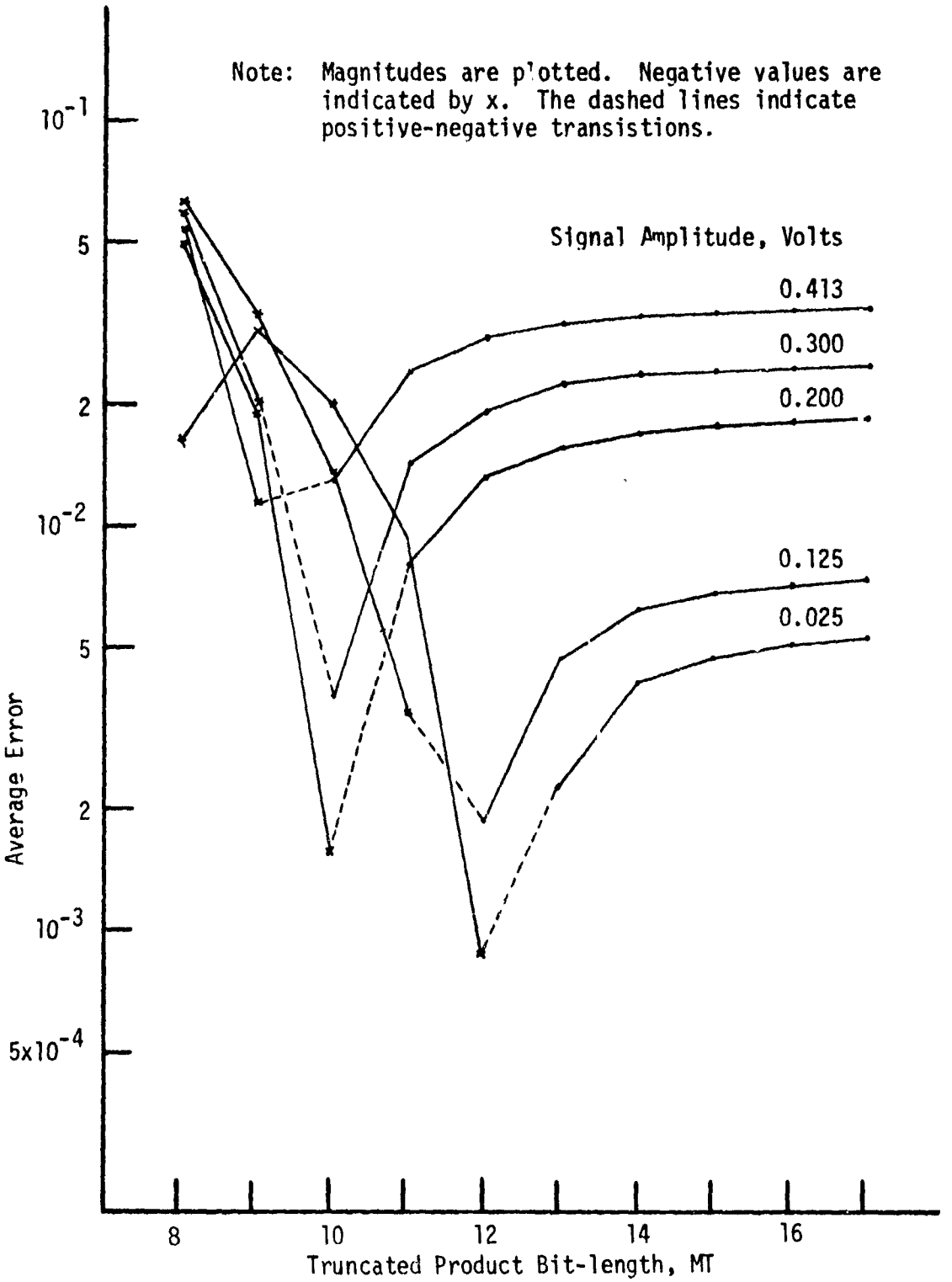


Fig. 3.31 Hardware RMS Average Error as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

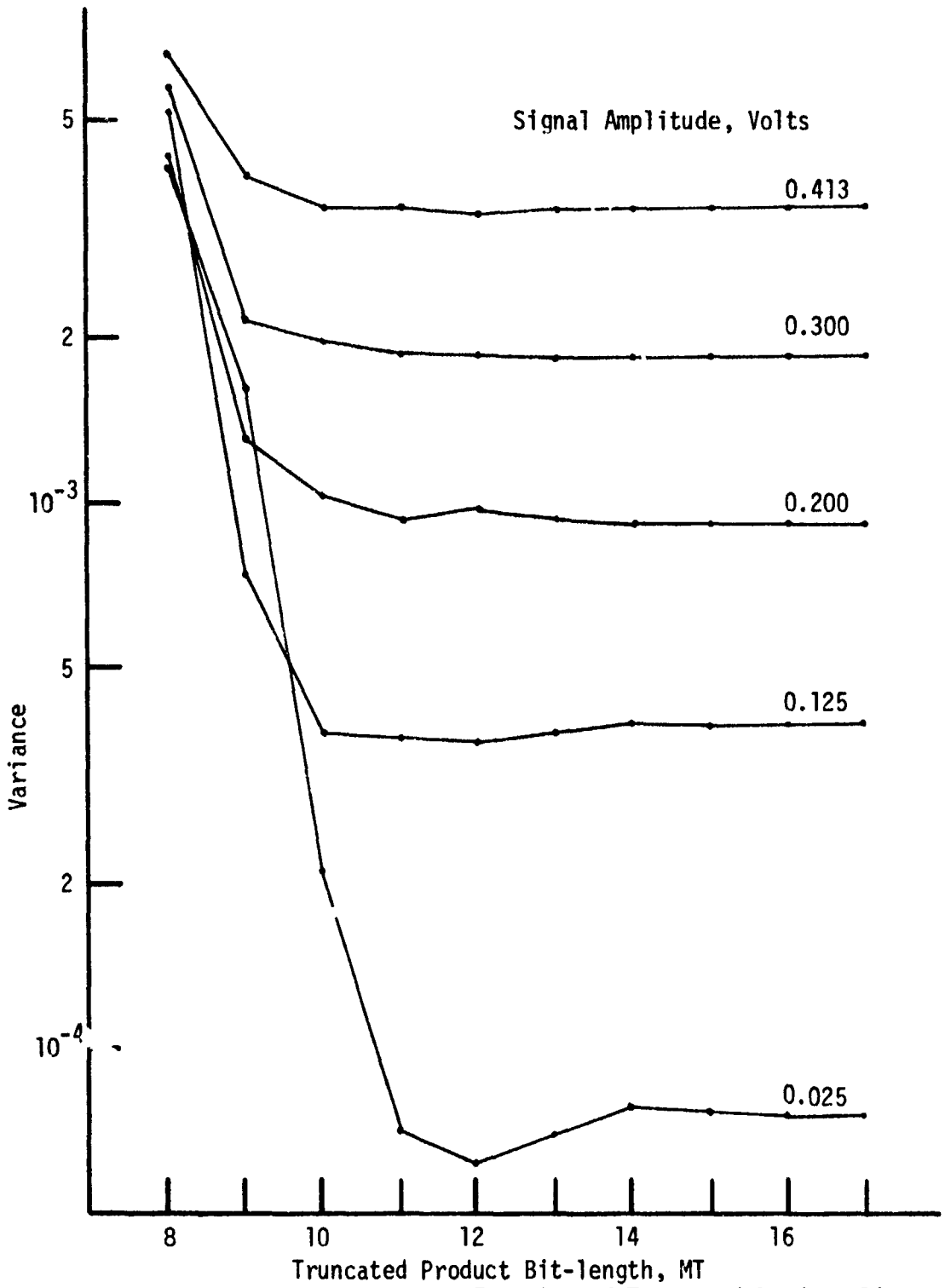


Fig. 3.32 Hardware RMS Variance as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

8798 CIDILKOVSKIJ, I.M. The Nature of the Scattering Process of Carriers in Trivalent-Pentavalent-Type Compounds and Some Ionic Crystals as Found by Thermomagnetic Measurements. In INTERNATIONAL CONFERENCE ON SEMICONDUCTOR PHYSICS, PROCEEDINGS, PRAGUE, 1960. New York, Academic Press, c1961. p. 96-100.

10544 CORNISH, A.J. Arrays of Inorganic Semiconducting Compounds. ELECTROCHEM. SOC., J., v. 106, no. 8, Aug. 1959. p. 685-689.

10904 WRIGHT, R.W. and J.A. BASTIN. The Characteristic Temperature and Effective Electron Mass for Conduction Processes in Cadmium Oxide. PHYS. SOC., PROC., v. 71, pt. 1, Jan. 1958. p. 109-116.

11654 HOGARTH, C.A. Some Conduction Properties of the Oxides of Cadmium and Nickel. PHYS. SOC., PROC., B, v. 64, pt. 8, no. 380, Aug. 1, 1951. p. 691-700.

11868 STUKE, J. Die Optische Absorptionskonstante von Kadmiumoxyd. The Optical Absorption Constant of Cadmium Oxide. Z. FUER PHYS., v. 137, no. 4, May 8, 1954. p. 401-415.

12730 COLIN, Y. and R. TUFEU. Sur les proprietes semi-conductrices de poudres d'oxyde de cadmium. On Semi-Conducting Properties of Cadmium Oxide Powders. ACAD. DES SCI., C.R., v. 256, no. 20, May 13, 1963. p. 4195-4198.

13280 BALCO RES. LAB. The Development of Electrical Conducting Transparent Coatings for Acrylic Plastic Sheet, by DALIN, G.A. and J. RENNERT. WADC TR no. 53-378, pt. 3, Contract no. AF 33-616-111. Apr. 1956. ASTIA AD-99 564.

13331 BYLANDER, E.G. Semiconductor Materials for High Temperatures. ELECTRO-TECHNOL. v. 72, no. 3, Sept. 1963. p. 123-127.

13648 MEISSNER, W. and H. FRANZ. Messungen mit Hilfe von fluessigem Helium. IX. Supraleitfaehigkeit von Carbiden and Nitriden. Measurements by Means of Liquid Helium. IX. Superconductivity of Carbides and Nitrides. Z. FUER PHYS., v. 65, 1930. p. 30-54.

15410 HOLLAND, L. and G. SIDDALL. The Properties of Some Reactively Sputtered Metal Oxide Films. VACUUM, v. 3, no. 4, Oct. 1953. p. 375-391.

17670 FINKENRATH, H. and H. KOEHLER. Temperaturabhaengigkeit der Optischen Konstanten duenner Cadmiumoxydschichten. Temperature Dependence of the Optical Constants of Thin Cadmium Oxide Films. Z. FUER NATURFORSCH., v. 19a, no. 10, Oct. 1964. p. 1236-1237.

20243 LAMB, E.F. and F.C. TOMPKINS. Semi-Conductivity and Thermoelectric Power of Cadmium Oxide. FARADAY SOC., TRANS., v. 58, pt. 7, no. 475. July 1962. p. 1424-1438.

20339 FINKENRATH, H. Die Optischen Konstanten von Kadmiumoxyd im Bereich der Eigenabsorption. The Optical Constants of Cadmium Oxide in the Intrinsic Absorption Range. Z. FUER ANGEW. PHYS., v. 16, no. 6, June 1964. p. 503-510.

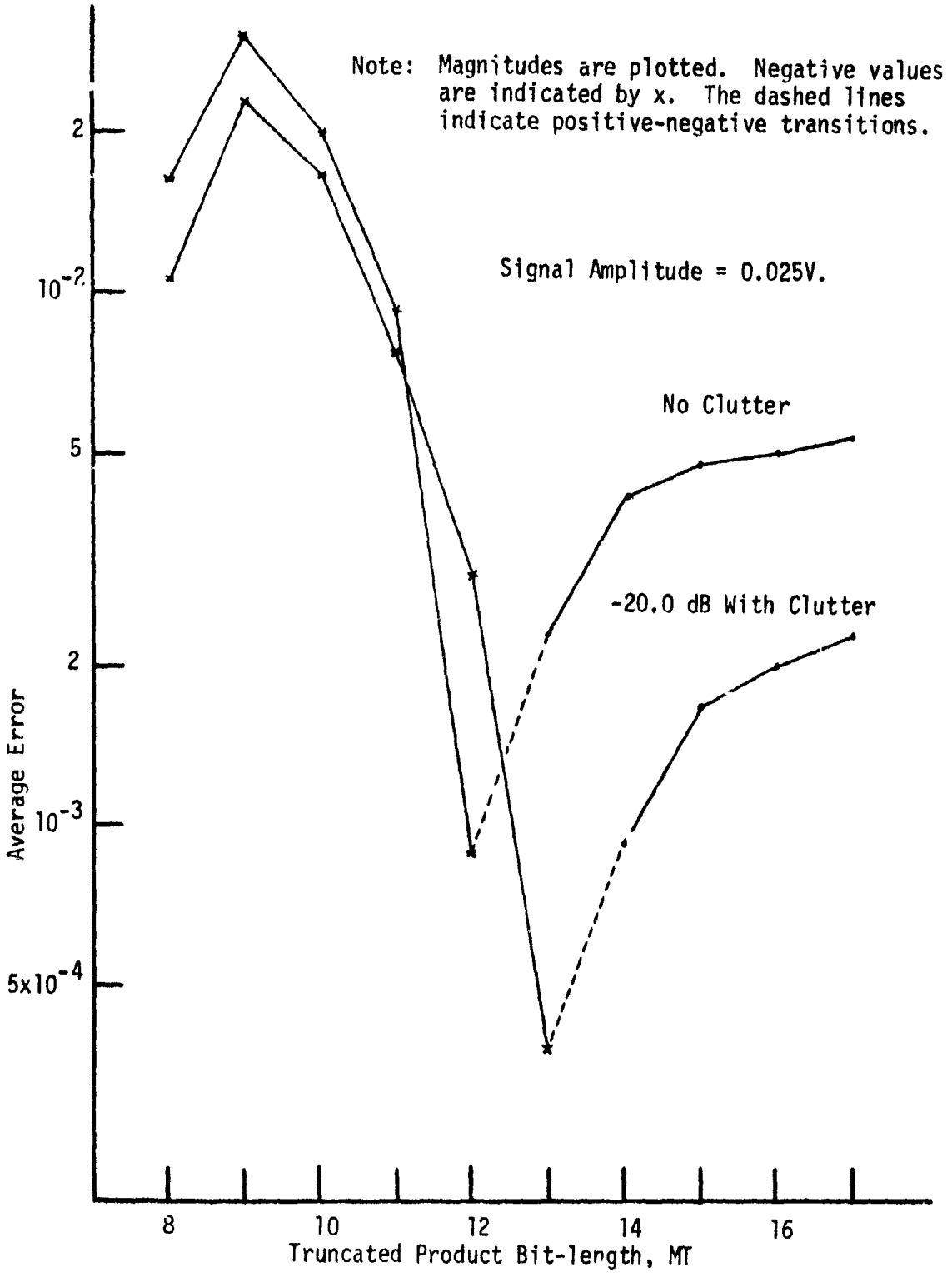


Fig. 3.33 Hardware RMS Average Error as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

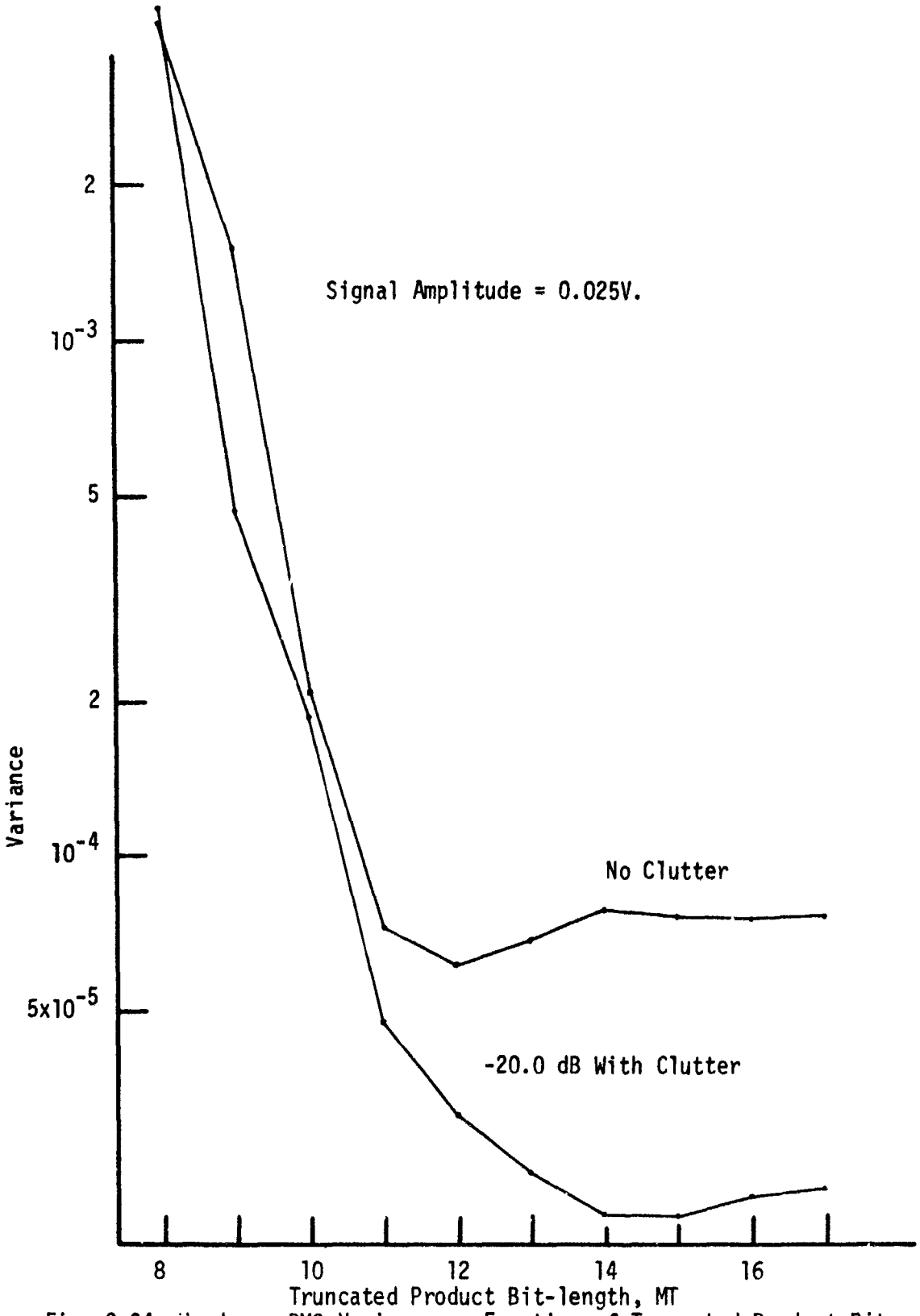


Fig. 3.34 Hardware RMS Variance as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

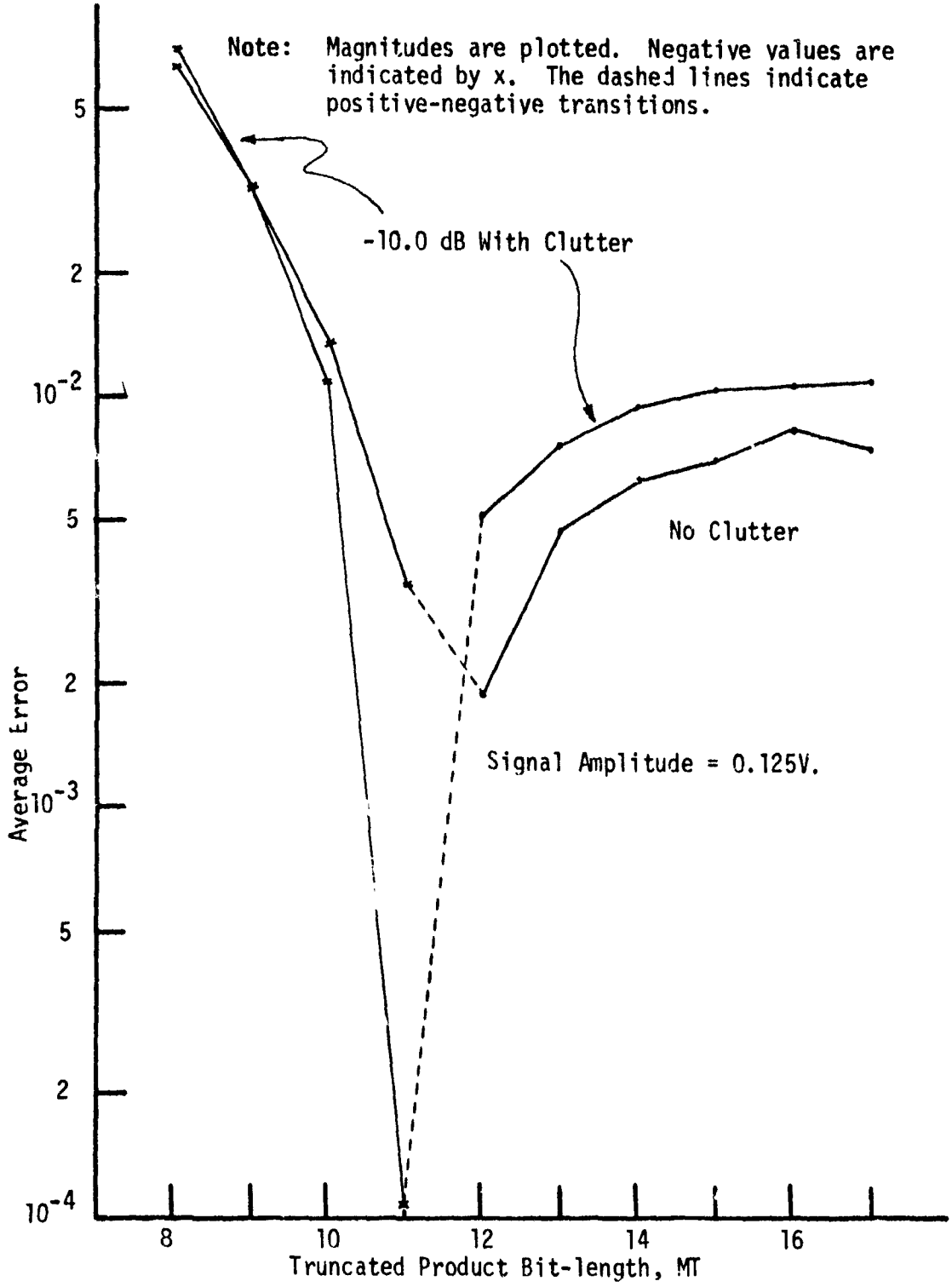


Fig. 3.35 Hardware RMS Average Error as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

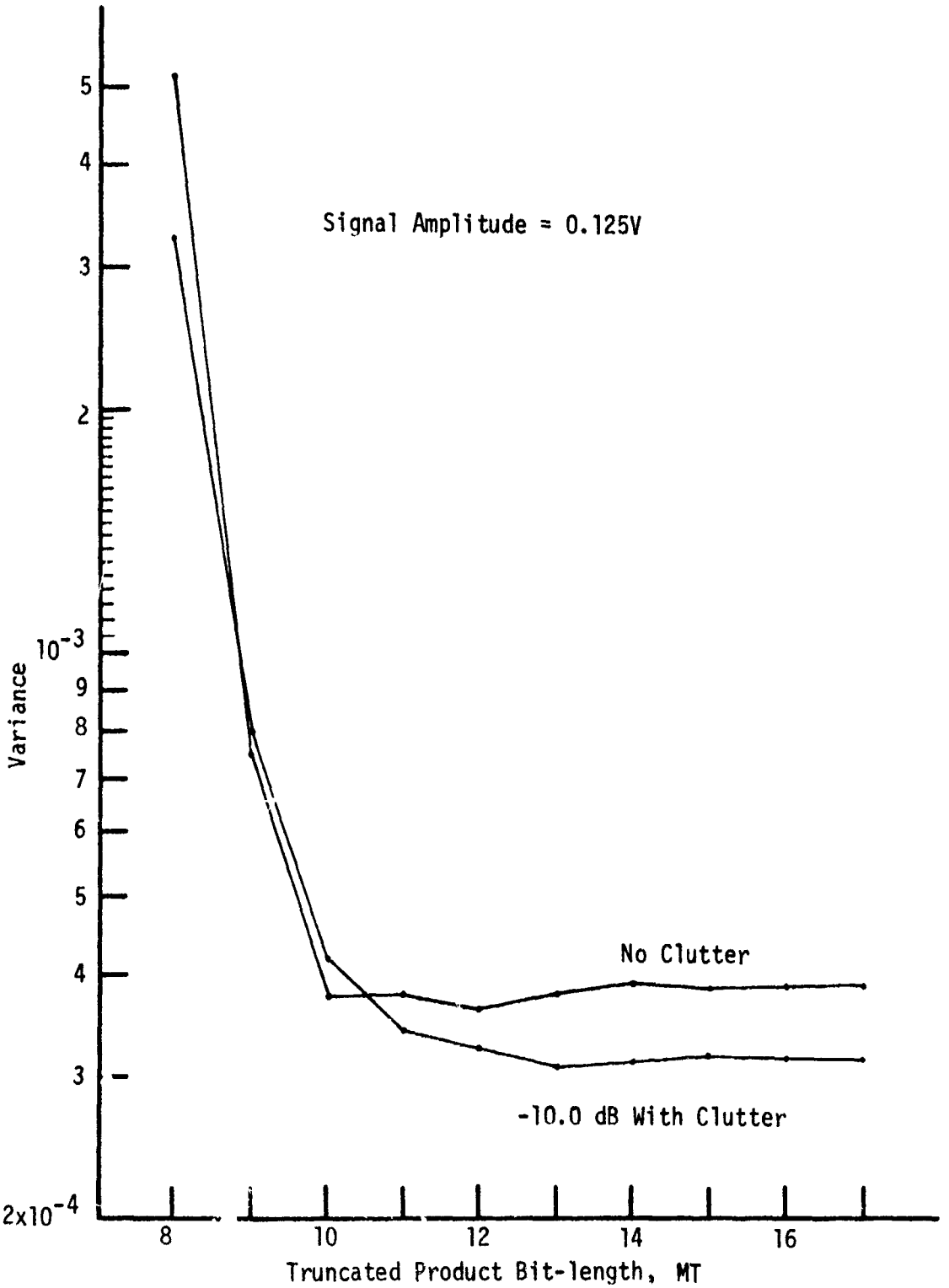


Fig. 3.36 Hardware RMS Variance as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

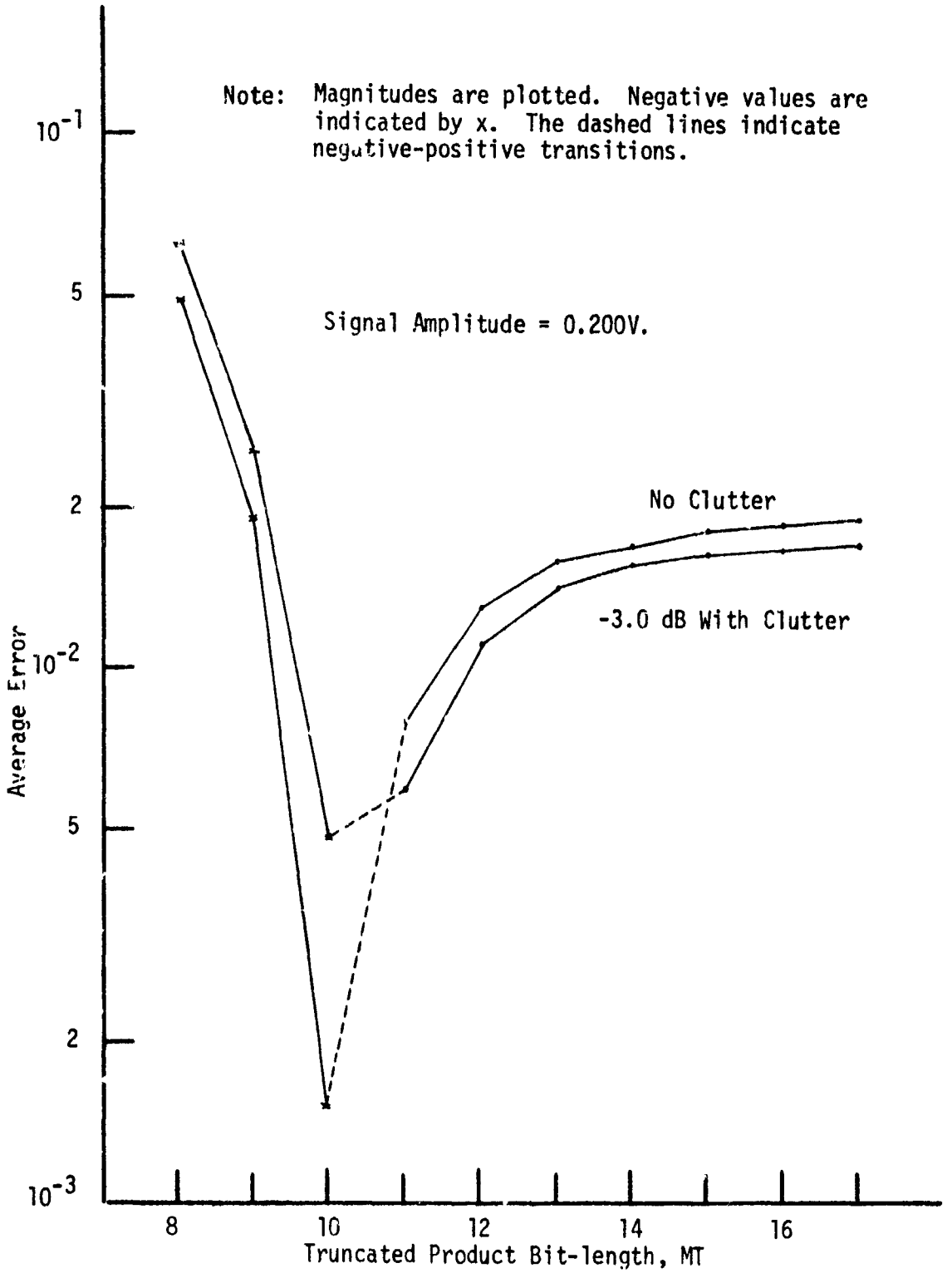


Fig. 3.37 Hardware RMS Average Error as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

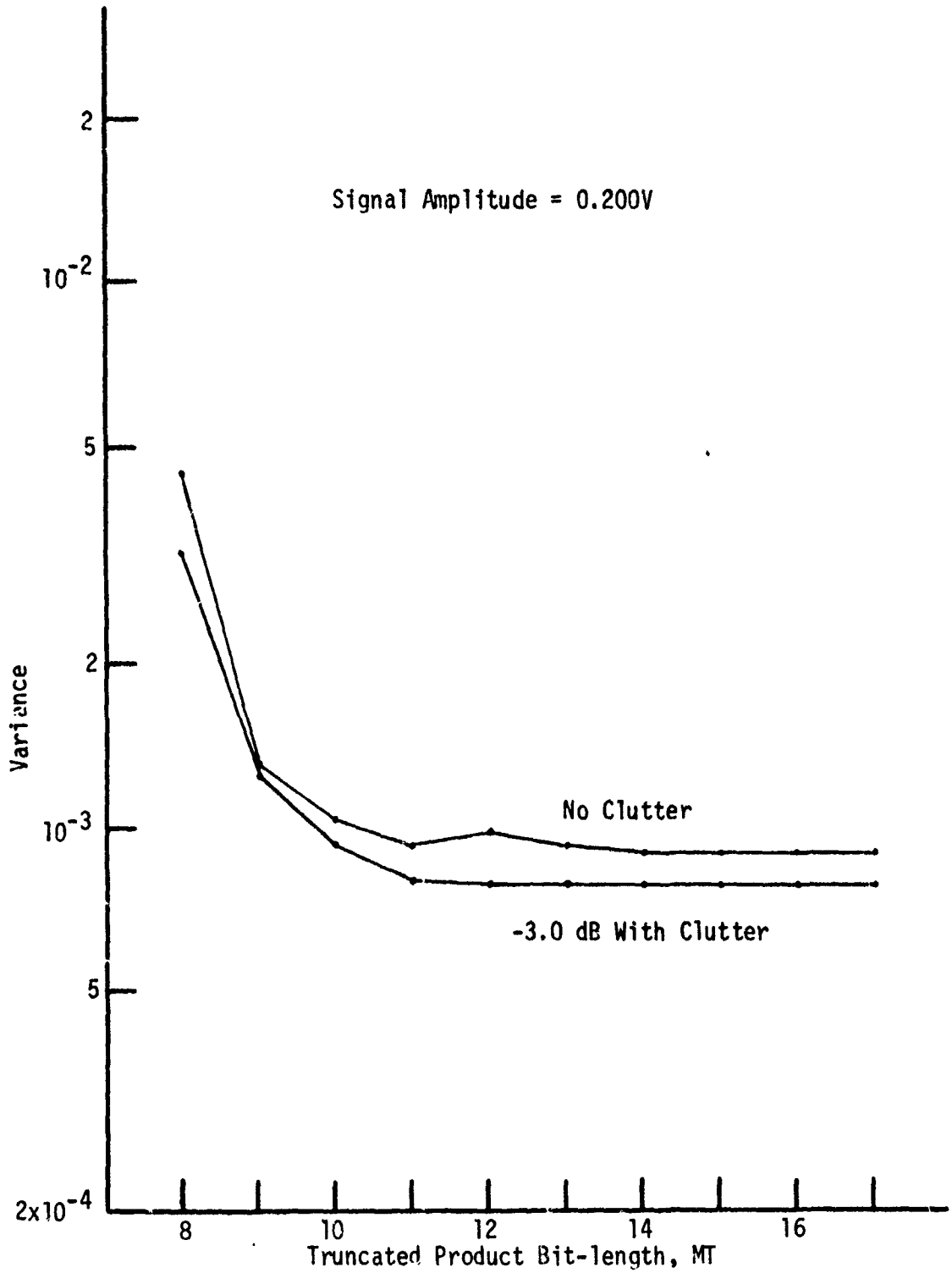


Fig. 3.38 Hardware RMS Variance as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

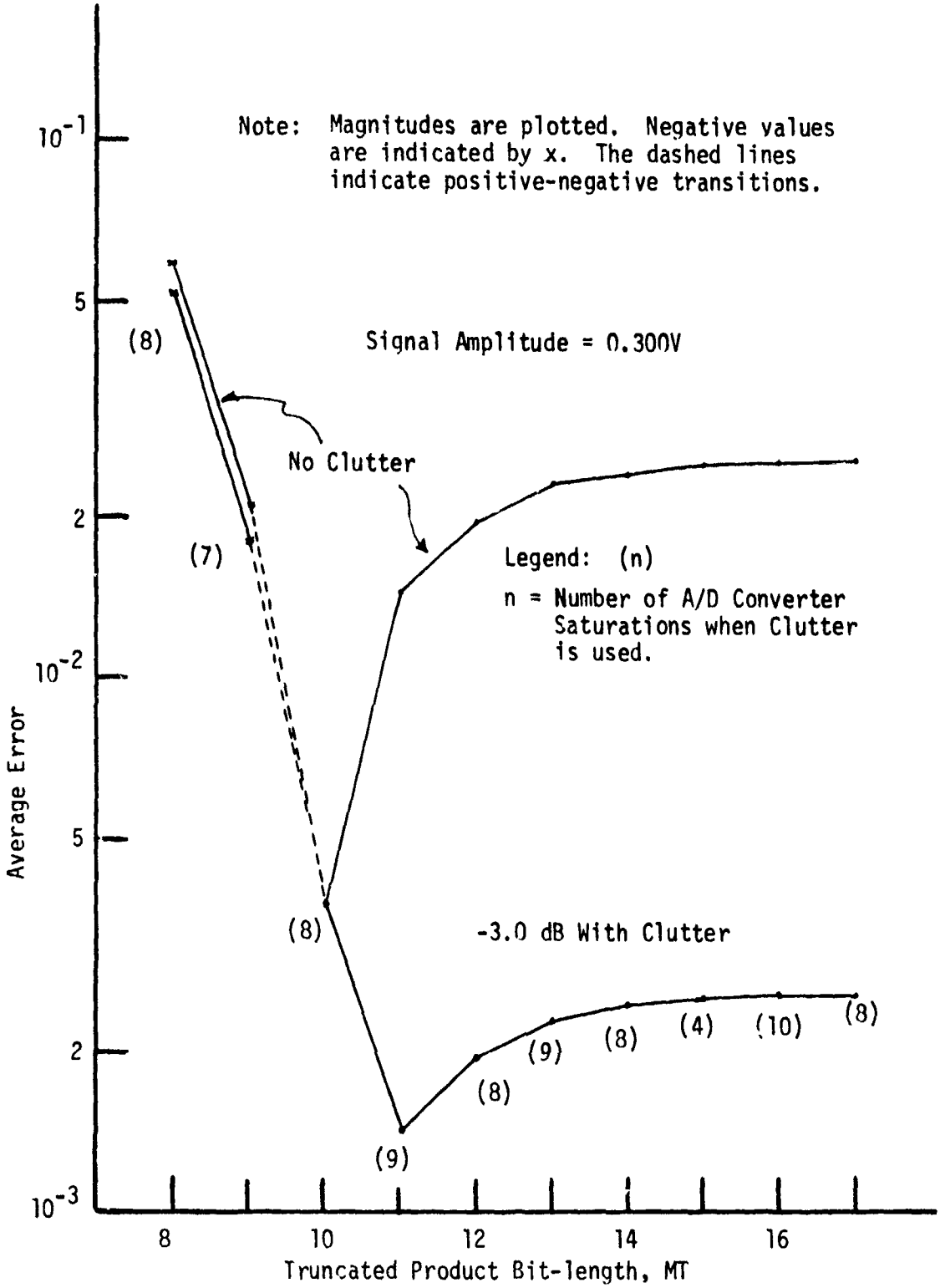


Fig. 3.39 Hardware RMS Average Error as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MT = 24)

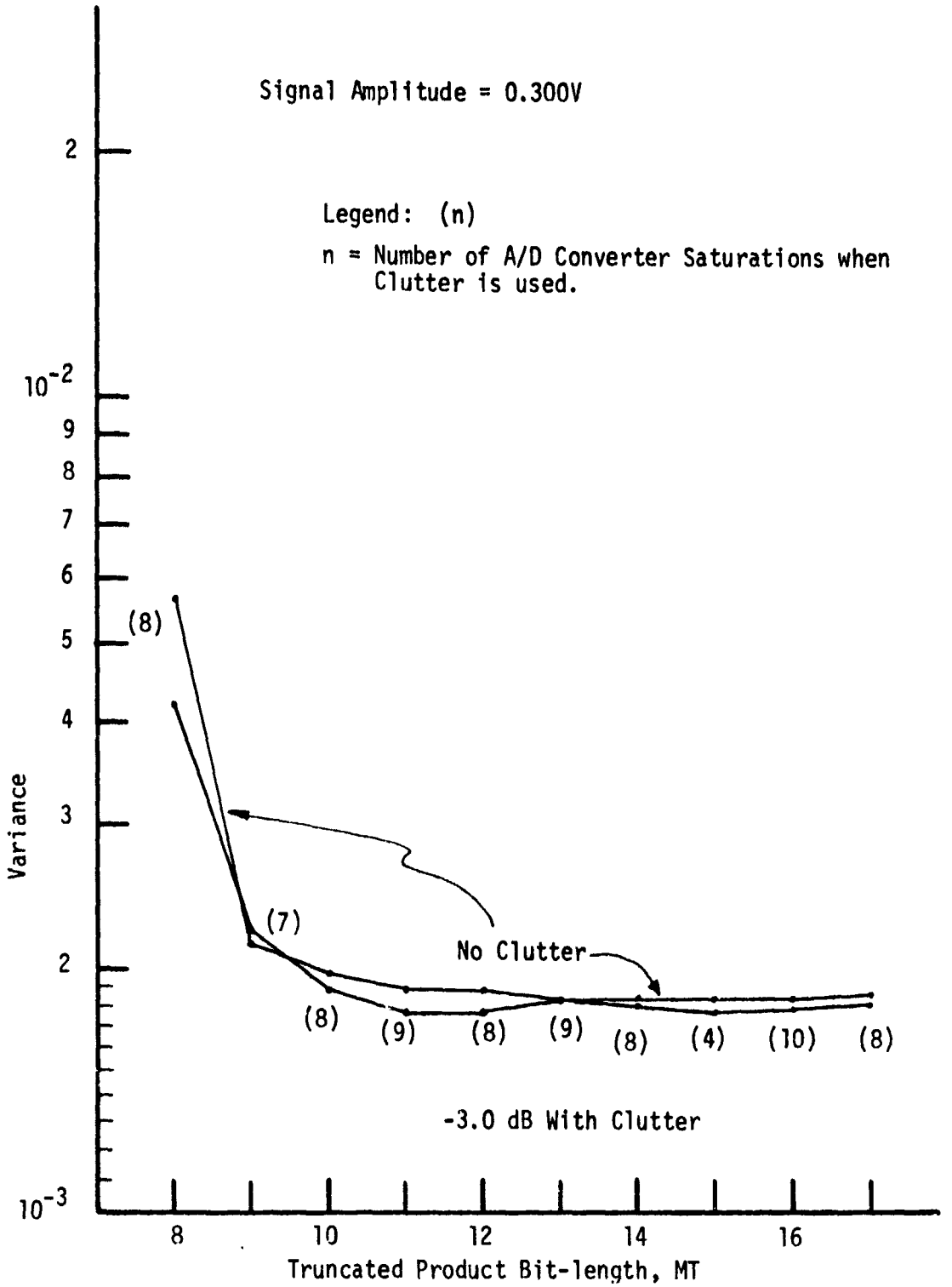


Fig. 3.40 Hardware RMS Variance as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

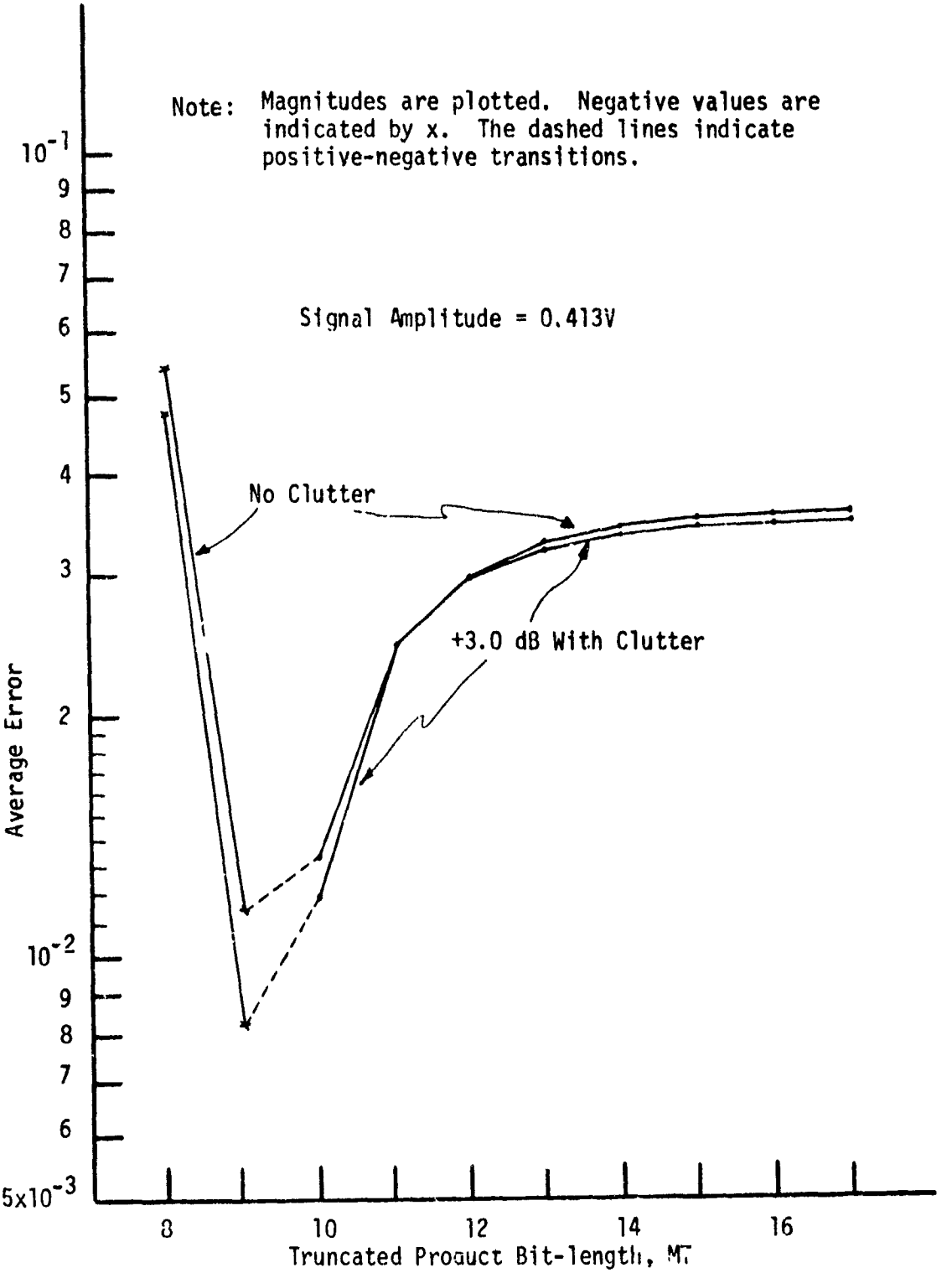


Fig. 3.41 Hardware RMS Average Error as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

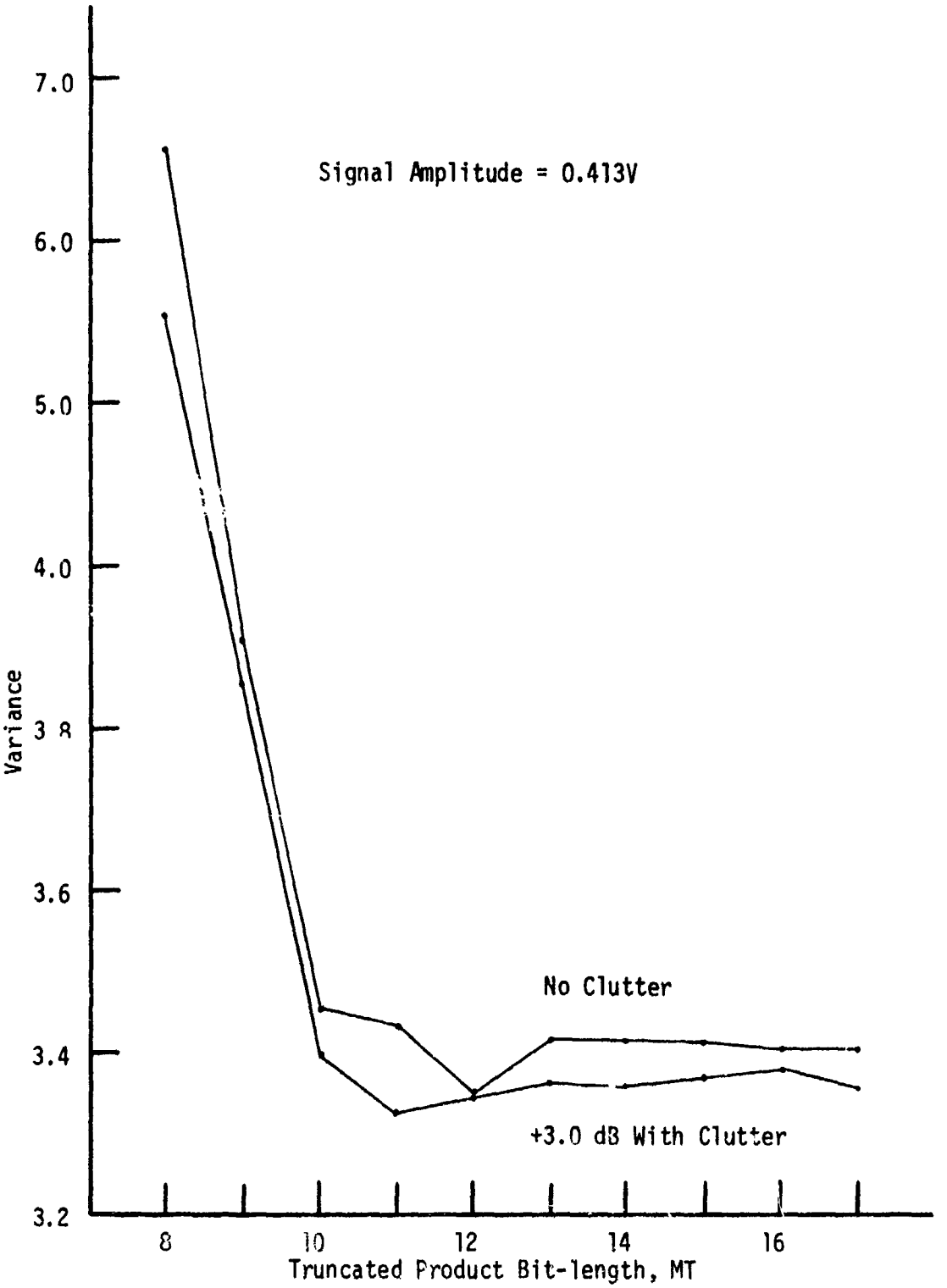


Fig. 3.42 Hardware RMS Variance as Function of Truncated Product Bit-length (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MF = ME = 20, MS = 24)

cult to generalize the trends in the average error but the variance curves show a new trend. For each signal amplitude the variance curve for the extreme clutter case now falls below the no clutter case. This seems contrary to expectations. It may be due to statistical dependences between the clutter, the RMS output and other quantization noise giving rise to some large negative cross-product terms which may cause the clutter case to fall below the no clutter case.

Figures 3.43 and 3.44 present data similar to Figures 3.29 and 3.30 and lead to the same conclusions, i.e., the excursions due to the different signal-to-clutter ratios from the no clutter case decrease as the signal amplitude increases. Also when clutter does not cause excessive A/D saturations, the clutter cases for a specific signal amplitude are grouped together.

3.2.3 Comparison of Theoretical and Simulation Results

A comparison is made between the theoretical and simulation results for the following 9-Tap cases (where the doppler frequency is 1500 Hz).

1) Average error and variance as functions of truncated product bit-length MT for signal amplitudes of 0.025V and 0.413V.
2) Average error and variance as functions of signal amplitude with the product bit-length MT = 17.

First, for the signal amplitude of 0.025V, Fig. 3.19 gives the hardware RMS average error as a function of MT. The theoretical upper bound for this case is shown in Figures 2.4 and 2.5 as the curve corresponding to ME=20. It is seen that for each value of MT in Fig. 3.19 the simulation results are at least an order of magnitude higher than the lower bound curve. The hardware RMS variance is shown in Fig. 3.20 and the related bound curves are obtained from Fig. 2.10. The upper bound curve is essentially the same for all $ME \geq 11$. For the entire range of MT values the upper bound has values greater than 0.0008 whereas the simulation results are all at least two orders of magnitude below this value. The lower bound value is identically zero and hence is below the simulation results.

For the signal amplitude of 0.413V the hardware RMS average error as a function of truncated product bit-length is given in Fig. 3.27. The corresponding upper bound values appear in Fig. 2.12 as the curve for ME=20. Alternatively, in Fig. 2.13 the curve for ME=15 is approximately the same as the curve for ME=20. For values of MT from 8 to 12 the upper bound curve is several orders of magnitude higher than the simulation results but, for values of MT from 12 to 17 the simulation results approach closely the upper bound value but are always smaller than the bound. For MT=17 where they are the closest the upper bound value is approximately 0.03 whereas the simulation value is 0.0135. Figure 2.14 shows a lower bound average error curve for which ME=MT+3. This condition ensures that no residue truncation error is produced at the integrator output. Consequently, this curve can be used to compare to the simulation value of ME=20. It is seen that the lower bound curve is lower than the simulation

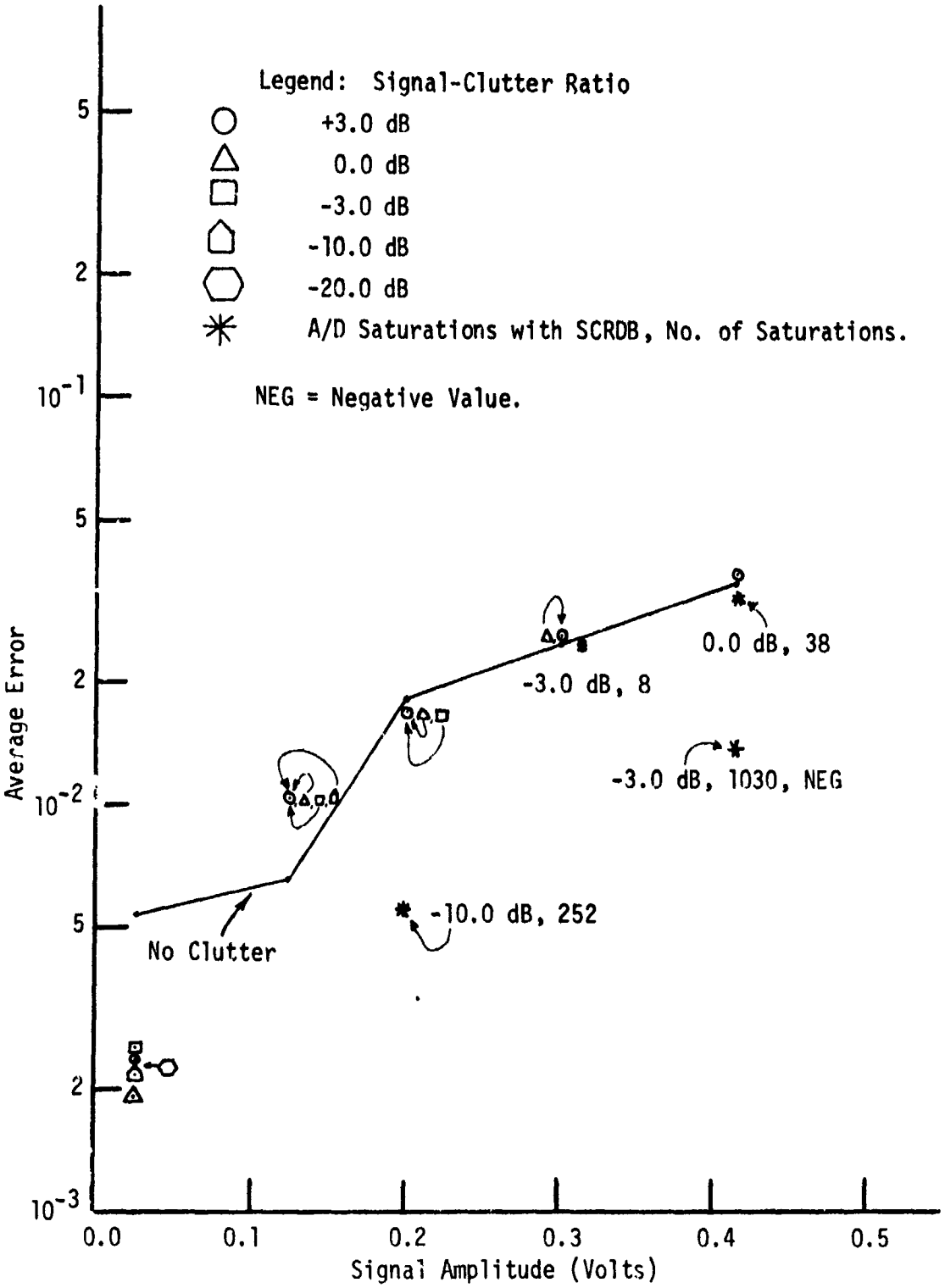


Fig. 3.43 Hardware RMS Average Error as Function of Signal Amplitude (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

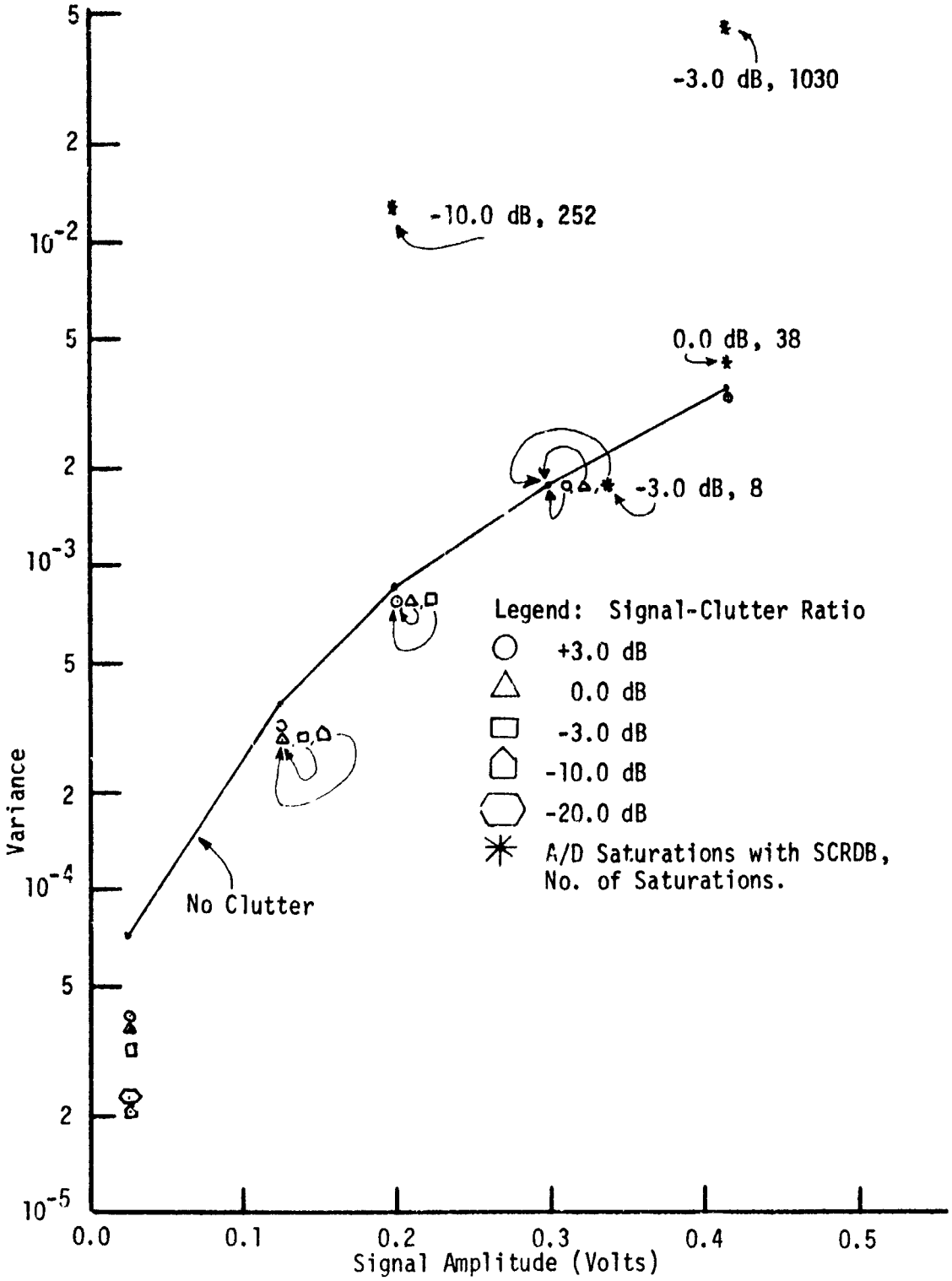


Fig. 3.44 Hardware RMS Variance as Function of Signal Amplitude (5-Tap Fixed-Point Simulation Results, Doppler Frequency = 1500 Hz, MX = MC = 9, MT = 17, MF = ME = 20, MS = 24)

results by a little less than an order of magnitude. The hardware RMS variance is shown in Fig. 3.28 and the related bound curves in Fig. 2.16. The upper bound curve (same as that for ME=9) is about three orders of magnitude higher than the simulation results. The lower bound is identically zero and hence below the simulation results.

Next, the comparison is made for the error results with varying signal amplitudes. The simulation results are for MT=17 and ME=20. The hardware RMS average error is shown in Fig. 3.29. Figures 2.19 and 2.20 show bound curves for MT=13 and ME=16 and are very close to the bound curves for MT=17 and ME=20. It is seen that for low signal amplitudes the simulation results are about an order of magnitude below the upper bound. However, as the signal amplitude increases the simulation results converge towards the upper bounds. At the closest point the upper bound value is approximately 0.03 whereas the simulation value is 0.0135. The lower bound curve stays at least an order of magnitude below the simulation results. These observations hold for the simulation results with and without clutter. A similar comparison for the variance values can be made with the help of the simulation results of Fig. 3.30 and the bound results of Fig. 2.21. The upper bound curve is at least three orders of magnitude higher than the no clutter simulation curve. The simulation results with clutter also fall below the bound curve with the exception of one case, viz., signal amplitude of 0.300V and signal-to-clutter power ratio of -10.0 dB. This is accounted for by the fact that a large number of A/D saturations occurred. The variance lower bound is identically zero, thus it bounds all the simulation results.

CHAPTER 4

FLOATING-POINT AND BLOCK-FLOATING-POINT PROCESSORS
by Brian P. Holt

4.1 GENERAL FLOATING-POINT CONSIDERATIONS

In fixed-point realizations of the moving target indicator (MTI) portion of the DSP system it is convenient to think of the numbers as fractions. The product of two numbers is then also a fraction and the least significant bits of the product can be truncated or rounded to maintain a given word length. The result of an addition need not be truncated or rounded to maintain the word length; however, it is possible that the sum exceeds unity in which case an overflow has occurred. Thus it is essential to scale the inputs to the MTI by an appropriate factor to insure that overflows do not occur or, conversely, to use processor word lengths based on maximum allowable input values.

The dynamic range limitations of a fixed-point MTI processor can be overcome by using floating-point arithmetic. In the following sections floating-point representation of binary numbers is introduced and the operations required to implement floating-point arithmetic are examined. The focus of the discussion is on signed magnitude, one's complement and two's complement schemes. Other notations are discussed only if they may offer significant advantages in performing a particular operation. It is assumed that the reader is familiar with signed magnitude, one's complement and two's complement numbers as discussed in the literature [17,18]. In the following discussion the most significant digit (MSD) of each binary number is the sign bit.

4.1.1 Representing Floating-Point Numbers

A binary number is expressed in floating-point form as

$$M \cdot 2^C$$

where M, the mantissa, is a positive or negative real number and C, the characteristic (often called the exponent), is an integer. Using this broad definition, a given number can have several floating-point representations. For example, the decimal number +0.75 could be represented as follows.

BINARY NUMBER	INTERPRETATION
$0.110_2 \cdot 2^0$	$+3/4 \cdot 1 = +3/4$
$0.011_2 \cdot 2^1$	$+3/8 \cdot 2 = +3/4$
$01.100_2 \cdot 2^{-1}$	$+3/2 \cdot 1/2 = +3/4$

In signal processing applications it is convenient if the mantissas of floating point numbers are fractions. When a multiplication of two numbers is performed (by multiplying the mantissas as fixed-point fractions and adding the characteristics) the mantissa of the result is a fraction (the location of the radix point is fixed). Truncation or rounding of the product is done by truncating or rounding the appropriate number of bits from the mantissa just as if it were a fixed-point fraction. Truncation and rounding introduce errors in the floating-point representation of a number. (For a discussion of the statistics of truncation and rounding errors in floating-point numbers see Appendix C.) The magnitude of the resultant error is minimized if the floating-point number is 'normalized', as described in the following section, before the truncation or rounding takes place.

4.1.2 Normalizing Floating-Point Numbers

A floating-point number is said to be normalized if

$$1/2 \leq |M| < 1 . \qquad (4.1)$$

Thus, a floating-point number is normalized by multiplying or dividing the mantissa by 2 until its magnitude falls within the desired range. By adding 1 to the characteristic for each division by 2 and subtracting 1 for each multiplication, the overall value of the floating-point number is preserved. Table 4-1 shows fixed-point and normalized floating-point representations of fractions in signed magnitude, one's complement and two's complement notation. In the table, both the mantissa and the characteristic of each floating point number are represented in the same notation, i.e., signed magnitude, one's complement, etc. This need not be the case. Often, hardware requirements can be reduced by using one scheme for representing the mantissas and another for representation of the characteristics.

According to the definition given above, the floating-point representation of zero can never be normalized. However, it is often convenient to establish the convention that the floating-point representation of zero is normalized by setting the characteristic to the most negative value it can have. The reason for this is made clear in Section 4.1.4.

Shifting the bits of a binary number left one position relative to the radix point is equivalent to multiplying the number by 2 and shifting them right one position is equivalent to dividing by 2. Thus, normalization can be accomplished by shifting the bits of the mantissa left or right the appropriate number of positions and adjusting the characteristic accordingly. If the mantissas of the numbers used are fractions, a right shift is needed only when the result of an addition overflows into the least significant integer position. Thus, at most one right shift is needed and the sign bit then would occupy the vacated integer bit position.

The algorithm for normalization of positive floating-point numbers in all three notations of Table 4-1 is as follows.

TABLE 4-1 NUMBER SYSTEM REPRESENTATIONS

Fraction	Fixed-Point Representation			Floating-Point Representation					
	Signed Magnitude	1's Complement	2's Complement	Signed Magnitude M	Signed Magnitude C	1's Complement M	1's Complement C	2's Complement M	2's Complement C
7/8	0.111	0.111	0.111	0.111	000	0.111	000	0.111	000
6/8	0.110	0.110	0.110	0.110	000	0.110	000	0.110	000
5/8	0.101	0.101	0.101	0.101	000	0.101	000	0.101	000
4/8	0.100	0.100	0.100	0.100	000	0.100	000	0.100	000
3/8	0.011	0.011	0.011	0.110	101	0.110	110	0.110	111
2/8	0.010	0.010	0.010	0.100	101	0.100	110	0.100	111
1/8	0.001	0.001	0.001	0.100	110	0.100	101	0.100	110
+ 0	0.000	0.000	0.000	0.000	111	0.000	100	0.000	100
- 0	1.000	1.111	-----	1.000	111	1.111	100	-----	---
-1/8	1.001	1.110	1.111	1.100	110	1.011	101	1.100	110
-2/8	1.010	1.101	1.110	1.100	101	1.011	110	1.100	111
-3/8	1.011	1.100	1.101	1.110	101	1.001	110	1.010	111
-4/8	1.100	1.011	1.100	1.100	000	1.011	000	1.100	000
-5/8	1.101	1.010	1.011	1.101	000	1.010	000	1.011	000
-6/8	1.110	1.001	1.010	1.110	000	1.001	000	1.010	000
-7/8	1.111	1.000	1.001	1.111	000	1.000	000	1.001	000
-8/8	-----	-----	1.000	-----	---	-----	---	1.100	001

1. Shift the mantissa right or left until the most significant 1 is to the immediate right of the radix point.
2. Increase the characteristic by 1 for a right shift and decrease it by 1 for each left shift.
3. When the mantissa is shifted left a 0 is brought into the vacated LSB position and any 0 shifted into the integer portion of the mantissa is dropped.

For negative numbers the algorithm used for normalization depends upon the notation being used. Signed magnitude negative numbers are normalized using the same algorithm used for positive numbers if we interpret the phrase "most significant 1" to mean the most significant 1 excluding the sign bit.

The rules for normalizing negative one's complement numbers are summarized as follows.

1. Shift the mantissa right or left until the most significant 0 is to the immediate right of the radix point.
2. Increase the characteristic by 1 for a right shift and decrease it by 1 for each left shift of the mantissa.
3. When the mantissa is shifted left a 1 is brought into the vacated LSB position and the 1 which was moved to the left of the radix point is dropped.

The algorithm for normalizing two's complement negative numbers follows.

1. If the magnitude of the mantissa is a positive or negative integer power of 2, shift the mantissa so that the most significant 1 (excluding the sign bit) which has a 0 to its immediate right is to the right of the radix point. If the mantissa contains only zeros shift right one position and bring a 1 into the vacated MSB position.
2. If the magnitude of the mantissa is not an integer power of 2, shift the mantissa so that the most significant 0 is to the right of the radix point.
3. Increase the characteristic by 1 for a right shift of the mantissa and decrease it by 1 for each left shift.
4. When the mantissa is shifted left, any bit shifted past the radix point is dropped and a 0 is brought into the vacated LSB position.

The hardware for normalizing two's complement negative numbers is complicated by the fact that the algorithm used depends upon the magnitude of the mantissa. By changing the rules for normalization of negative two's complement numbers this complication can be eliminated.

The definition of Equation (4.1) shall be changed for negative two's complement numbers so that

$$1/2 < |M| \leq 1 . \qquad (4.2)$$

This new definition makes use of the fact that the decimal number -1 can be represented as a two's complement fraction (see the fixed-point representations of Table 4-1). Using Equation (4.2) the rules for normalizing negative two's complement numbers are as follows.

1. Shift the mantissa right or left until the most significant 0 is to the immediate right of the radix point.
2. Increase the characteristic by 1 for a right shift and decrease it by 1 for each left shift of the mantissa.
3. When the mantissa is shifted left a 0 is brought into the vacated LSB position and the 1 which was moved to the left of the radix point is dropped.

Table 4-2 shows the two's complement fixed-point and normalized floating-point representation of numbers based on Equation (4.2).

For the remainder of this chapter and in the floating-point simulation described in Chapter 6, negative two's complement numbers are normalized using the alternative algorithm based on Equation (4.2).

4.1.2 Multiplication

Multiplication of floating-point numbers is accomplished by taking the product of the mantissas of the two operands just as if they were fixed-point numbers and summing the characteristics. It is not necessary that the numbers to be multiplied be normalized but, as pointed out previously, it is desirable that the mantissas be fractions. If the two numbers are normalized, however, the mantissa of the product will be a fraction between 1/4 and 1 (except of course when either or both of the operands are zero) and normalization of the result would never require a right shift and at most one left shift of the mantissa.

4.1.3 Addition

Addition of positive and negative numbers in signed magnitude notation is complicated by the fact that both binary adders and subtractors are required. Additional circuitry is also necessary for determining the sign of the result. The use of one's complement or two's complement numbers allows addition and subtraction operations to be carried out

TABLE 4-2 ALTERNATIVE TWO'S COMPLEMENT REPRESENTATION

Fraction	2's Comp. Fixed-Point	2's Comp. Floating-Point	
		M	C
7/8	0.111	0.111	000
6/8	0.110	0.110	000
5/8	0.101	0.101	000
4/8	0.100	0.100	000
3/8	0.011	0.110	111
2/8	0.010	0.100	111
1/8	0.001	0.100	110
+ 0	0.000	0.000	100
- 0	-----	-----	---
-1/8	1.111	1.000	101
-2/8	1.110	1.000	110
-3/8	1.101	1.010	111
-4/8	1.100	1.000	111
-5/8	1.011	1.011	000
-6/8	1.010	1.010	000
-7/8	1.001	1.001	000
-8/8	1.000	1.000	000

using only adder circuitry. Furthermore, the sign of the result is determined as a part of the addition process. Thus the DSP would use a complement system and this section is devoted to explaining floating-point addition using one's complement and two's complement notations.

Floating-point numbers are added by summing the mantissas as if they were fixed-point numbers. Recall that in adding two fixed-point binary numbers, the radix points are first lined up so that bits with equal weights, or positional coefficients, are added together and carries are propagated to the column with the next higher weight. By shifting the mantissa bits of the two floating-point numbers so that the characteristics are made equal, the bits in corresponding columns of the mantissas have equal weights and can be added correctly. This process of shifting the mantissa bits and adjusting the characteristics is called 'aligning' and is explained in Section 4.1.4. Once the two numbers are aligned and the mantissas added, the characteristic of the sum is set equal to the characteristics of the aligned numbers.

When one's complement or two's complement numbers are added, the sign bits are treated as part of the arithmetic portion of the operands as illustrated by the following examples. Notice that in one's complement addition a carry propagated out of the sign bit column is added to the least significant bit of the sum and any carries thereby generated are propagated forward. This peculiarity of one's complement addition is called 'end-around carry'.

Case 1: Addition of a positive and a negative number never results in overflow.

Example 4-1: No overflow

One's Complement	Decimal	Two's Complement
1.1101	-2/16	1.1110
+ 0.0001	+1/16	+ 0.0001
0)1.1110		0)1.1111
└→ 0		
1.1110	-1/16	1.1111

Example 4-2: No overflow

One's Complement	Decimal	Two's Complement
1.1101	-2/16	1.1110
+ 0.1000	8/16	+ 0.1000
1)0.0101		1)0.0110
└→ 1		
0.0110	+6/16	0.0110

Case 2: Addition of two positive numbers results in overflow if the sign bit of the result is 1.

Example 4-3: No overflow

One's Complement	Decimal	Two's Complement
0.1100	+12/16	0.1100
+ 0.0010	+2/16	+ 0.0010
0)0.1110		0)0.1110
└→ 0		
0.1110	+14/16	0.1110

Example 4-4: Overflow

One's Complement	Decimal	Two's Complement
0.1100	+12/16	0.1100
+ 0.1011	+11/16	+ 0.1011
0)1.0111		0)1.0111
└→ 0		
01.0111	+23/16	01.0111

Case 3: Addition of two negative numbers results in overflow if the sign bit of the result is 0.

Example 4-5: No overflow

One's Complement	Decimal	Two's Complement
1.1110	-1/16	1.1111
+ 1.1101	-2/16	+ 1.1110
1)1.1011		1)1.1101
└→ 1		
1.1100	-3/16	1.1101

Example 4-6: Overflow

One's Complement	Decimal	Two's Complement
1.0011	-12/16	1.0100
+ 1.1010	-5/16	1.1011
1)0.1101		1)0.1111
└→ 1		
10.1110	-17/16	10.1111

When an overflow occurs in the addition of two floating-point mantissas as in Examples 4-4 and 4-6, the carry out of the sign column of the two operands then becomes the sign of the result. The result is then normalized by shifting the resultant sum to the right and increasing the characteristic as explained in Section 4.1.2. If overflow did not occur, it is possible that the mantissa will have to be shifted left and the characteristic decreased until the magnitude of the mantissa is in the appropriate range.

Another well known algorithm for detecting overflows in additions involves comparing the carries into and out of the sign bit column [17]. If the two carries are not equal, then an overflow has occurred.

4.1.4 Aligning Floating-Point Numbers

In Section 4.1.3 it was stated that floating-point numbers must be aligned prior to addition. In the following discussion it is assumed that the two numbers to be aligned are in normalized form. (Although this is not strictly necessary, it greatly simplifies the alignment algorithm.)

The rules for aligning two (normalized) floating-point numbers are listed below.

1. The mantissa of the number with the smaller characteristic is shifted right (and its characteristic increased) by the number of bit positions equal to the difference in the two characteristics.
2. For one's and two's complement mantissas the MSB position vacated by a right shift is filled by a bit equal to the sign bit, i.e., a 0 for positive numbers and a 1 for negative numbers.

Table 4-3 gives several examples of the alignment process for both one's complement and two's complement notation. Notice that each time a mantissa is shifted right the LSB is truncated to maintain the original word length. Parts (g) and (h) of Table 4-3 are included to illustrate the motivation for normalizing the floating-point representation of zero by setting the characteristic to the lowest value it can have (see Section 4.1.2). This insures that when two floating-point numbers are to be aligned, and one of them is zero, the mantissa of the nonzero number will not be shifted (causing truncation errors) since its characteristic cannot be the smaller of the two.

	ONE'S COMPLEMENT				TWO'S COMPLEMENT			
	Before		After		Before		After	
	Binary	Decimal	Binary	Decimal	Binary	Decimal	Binary	Decimal
a)					b)			
M_1 =	1.0110	-9/16	1.0110	-9/16	1.0111	-9/16	1.0111	-9/16
C_1 =	101	-2	101	-2	110	-2	110	-2
M_2 =	1.0011	-12/16	1.1001	-6/16	1.0100	-12/16	1.1010	-6/16
C_2 =	100	-3	101	-2	101	-3	110	-2
c)					d)			
M_1 =	1.0110	-9/16	1.1101	-2/16	1.0111	-9/16	1.1101	-3/16
C_1 =	001	1	011	3	001	1	011	3
M_2 =	1.0010	-13/16	1.0010	-13/16	1.0011	-13/16	1.0011	-13/16
C_2 =	011	3	011	3	011	3	011	3
e)					f)			
M_1 =	0.1001	9/16	0.0100	4/16	0.1010	10/16	0.1010	10/16
C_1 =	000	0	001	1	011	3	011	3
M_2 =	1.0111	-8/16	1.0111	-8/16	1.0000	-16/16	1.1111	-1/16
C_2 =	001	1	001	1	101	-3	011	3
g)					h)			
M_1 =	0.1101	13/16	0.1101	13/16	0.1011	11/16	0.1011	11/16
C_1 =	001	1	001	1	010	2	010	2
M_2 =	1.1111	-0	1.1111	-0	0.0000	0	0.0000	0
C_2 =	100	-3	001	1	100	0	010	2

TABLE 4-3 ALIGNING FLOATINC-POINT NUMBERS

4.2 GENERAL BLOCK-FLOATING-POINT CONSIDERATIONS

The motivation for using floating-point arithmetic in realizing the MTI portion of the DSP is to overcome the dynamic range limitations of fixed-point systems. Implementing floating-point arithmetic, however, significantly increases the complexity of the processor hardware. An alternative realization, block-floating-point, is a hybrid approach which has some of the advantages of both fixed-point and floating-point structures. Oppenheim [19] has proposed and evaluated a structure for implementing recursive digital filters using block-floating-point arithmetic. His results, however, are not generally applicable to the fixed-window MTI structure used in the radar signal processor.

In block-floating-point arithmetic all of the intermediate values are jointly normalized, i.e., they all have the same characteristic. This is accomplished by shifting the bits of each number left one position at a time until any one of the numbers is normalized according to the rules of Section 4.1.2. The following table illustrates how several positive numbers are jointly normalized.

	JOINTLY NORMALIZED RESULT	
ORIGINAL FIXED-POINT NUMBER	MANTISSA	CHARACTERISTIC
0.00111	0.01110	-1
0.00011	0.00110	-1
0.01010	0.10100	-1
0.00010	0 00100	-1
0.01100	0.11000	-1

The arithmetic operations of addition and multiplication are performed in fixed-point arithmetic using the jointly normalized mantissas. The characteristic is then used to scale the fixed-point output to give the final fixed-point result.

This joint normalization scheme appears to be particularly applicable in schemes where a large number of intermediate results are stored and fed back simultaneously. However, in the case where only one intermediate result is to be stored, such as in the fixed-window, multiple range bin MTI (where only one intermediate result is stored per range bin) it does not offer any significant advantages over floating-point realization. For this reason, a hardware realization for a block-floating-point MTI processor has not been proposed.

4.3 DESCRIPTION OF PROPOSED FLOATING-POINT PROCESSOR

In this section a floating-point configuration for realizing the digital signal processor described in Section 2.1 is proposed. Block diagrams of the three major system divisions, viz., 1) the I and Q channel filters, 2) the RMS approximation circuit and 3) the post residue integrator are presented and discussed. General requirements for implementing the various subtasks are discussed but a detailed hardware design

is not attempted.

4.3.1 Floating-Point Filter

Figure 4.1 shows a block diagram of the proposed floating-point implementation of the I channel filter of the DSP (the Q channel is identical). The A/D converter outputs are in two's complement fixed-point form as described in Section 3.1.2 and are supplied at a 5 MHz rate. The coefficient mantissas are stored as 9 bit two's complement fractions. The coefficient characteristics are stored in magnitude form (no sign bit is needed) since they will always be zero or negative. This is a result of the requirement that the MTI filters have unity noise gain, i.e., the sum of the squares of the coefficients is one [10]. Thus the coefficient values would all be less than one.

A correction circuit is used in conjunction with an 8 by 8 bit binary multiplier to perform two's complement multiplication. This same multiplier scheme is presently used for multiplication in the fixed-point processor at the Radar Technology Branch, U.S. Army Missile Command [1]. The output of the multiplier is normalized prior to truncation in order to minimize the magnitude of the truncation error.

Throughout the floating-point processor , truncation is used (instead of rounding) when word lengths are reduced. Since rounding the mantissa of a floating-point number could result in an overflow, extra hardware would have to be included to normalize the result.

The product can be normalized in two ways as shown below.

1. Serial Approach
   Load the mantissa into a parallel access serial shift register and compare the most significant fractional bit with the sign bit. If the most significant fractional bit is equal to the sign bit the mantissa bits are shifted left one position and the characteristic is decremented by one. The compare and shift operation is then repeated until the most significant fractional bit does not equal the mantissa sign bit.
2. Parallel Approach
   The mantissa bits are loaded into a position scaler, i.e., a shifting circuit which can shift the bits any number of positions in one operation (for example the Signetics 8243 Position Scaler). The number of bit position that the mantissa is shifted must be determined by a circuit (such as ROM or PLA for example) which counts the number of consecutive leading zeros in the mantissa if the sign is positive and the number of consecutive one's if the sign is negative.

The serial approach would require considerably less hardware than the parallel approach. However, if the number of shifts needed for normalization is large, the serial test and shift would have to be done at high speed. For example, if 8 shift and test operations are required for normalization in the serial approach, and normalization must be accom-

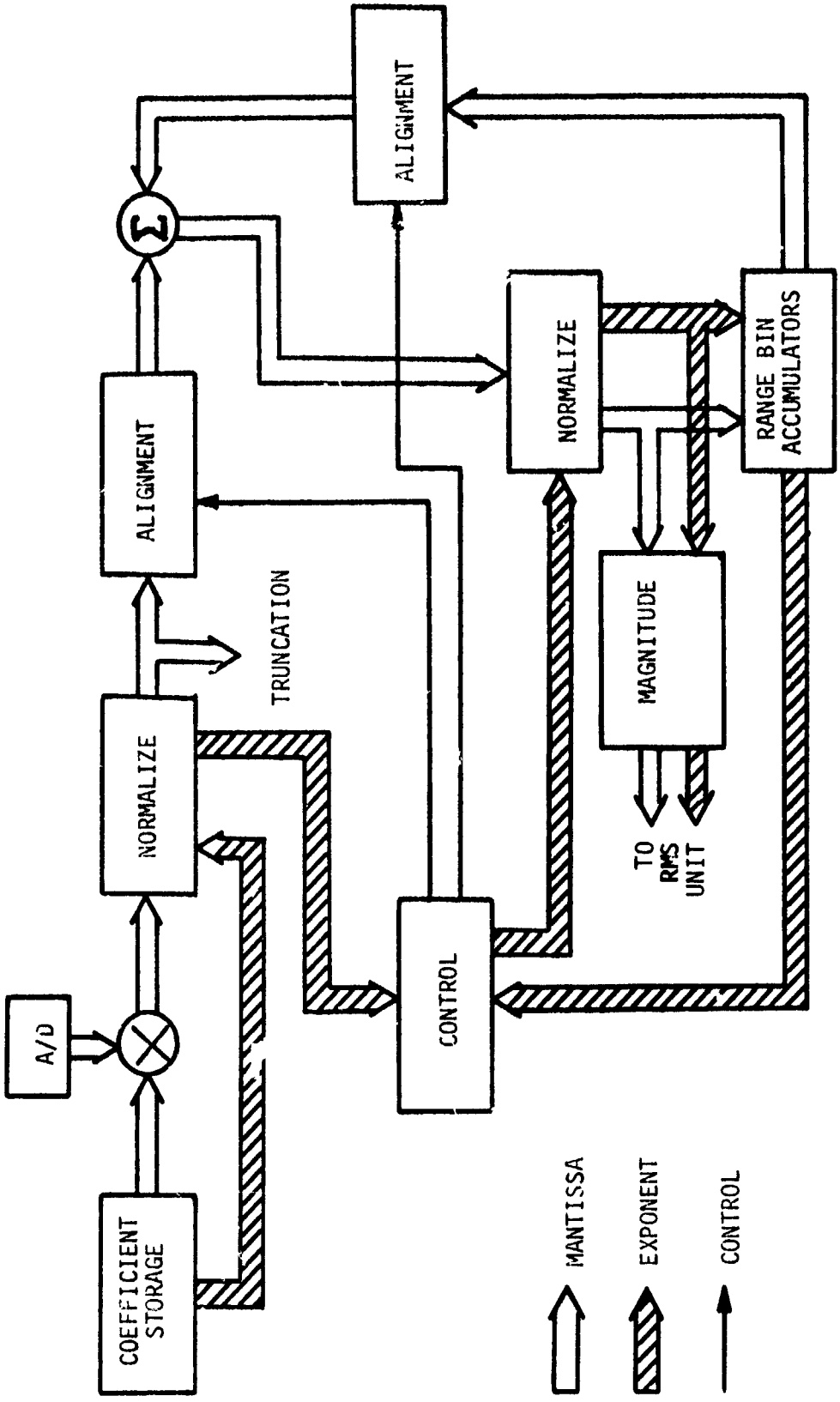


Figure 4.1 Block Diagram of One Channel of Floating-Point Filter.

plished in a maximum of 200 nanoseconds, then the test and shift operation would have to be accomplished at a 40 MHz rate.

The control block of Figure 4.1 calculates the difference in the characteristics of the normalized product and the partial sum fed back from the range bin accumulators. Control signals are then fed to the alignment circuits and the characteristic of the sum is set equal to the larger of the two characteristics. Shifting of the mantissas in the alignment process can be done using a serial shift register or a position scaler. If the serial approach is used, the same comments apply as those concerning the normalization circuit speed.

The sum is then normalized (note that a right shift is required if overflow occurs). The normalized sum is then stored in the range bin accumulators. After the appropriate number of cycles the two's complement filter output mantissa is converted to magnitude form then sent to the RMS circuit described in Section 4.3.2. Conversion of negative two's complement numbers is accomplished by complementing each bit and adding 1 to the result. Special precautions are necessary to insure that the two's complement mantissa representation of -1 is converted correctly. For example, if the mantissa of the filter output was the two's complement number 1.0000 (-1 in decimal), then complementing each bit and adding 1 would give the same two's complement number as a result. The correct magnitude would be obtained by shifting the mantissa bits right one position (just as is done when an addition overflows) and increasing the characteristic.

4.3.2 RMS Approximation Circuit

A block diagram of the proposed floating-point RMS approximation circuit is shown in Figure 4.2. The algorithm implemented is the two sector approximation described in Section 2.1. The nomenclature of Table 4-4 is used in the following discussion.

TABLE 4-4 RMS UNIT NOMENCLATURE

SYMBOL	DESCRIPTION
$\|I\|$	Magnitude of the I channel filter output (normalized)
$\|Q\|$	Magnitude of the Q channel filter output (normalized)
L	The larger of $\|I\|$ and $\|Q\|$
S	The smaller of $\|I\|$ and $\|Q\|$
M_L	Mantissa of L (always positive)
M_S	Mantissa of S (always positive)
C_L	Characteristic of L
C_S	Characteristic of S

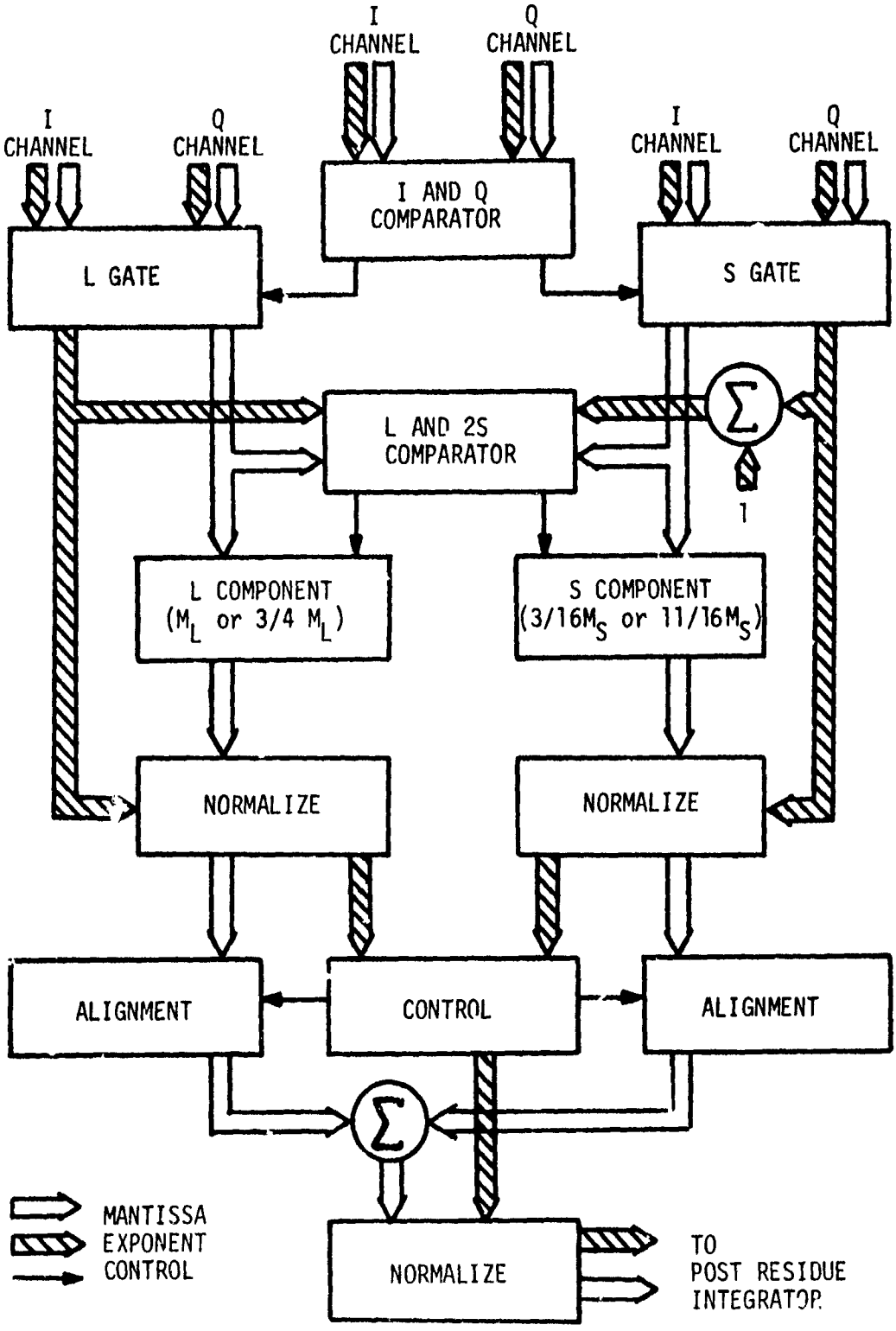


Fig. 4.2 Floating-Point RMS Approximation Circuit.

|I| and |Q| are simultaneously fed to the |I| and |Q| comparator and the L and S gates. The comparator generates signals which cause L and S to pass through the appropriate gates. C_S is then increased by 1 (which in effect multiplies S by 2) and a second comparison is made. The result of the second comparison controls the functioning of the L and S component blocks. If $L \geq 2S$, then M_L is unchanged and 3/16 M_S is generated. If, however, $L < 2S$, then 3/4 M_L and 11/16 M_S are generated.

Two different approaches could have been taken in generating the L and S components as illustrated below.

Approach 1: Generate the components by operating only on the mantissas of L and S.

COMPONENT TO BE GENERATED	METHOD	
	M	C
3/16 S	$(M_S + M_S/2)/8$	C_S
11/16 S	$((M_S + M_S/2)/4 + M_S)/2$	C_S
3/4 L	$(M_L + M_L/2)/2$	C_L

Approach 2: Generate the components by operating on both the mantissas and characteristics of L and S.

COMPONENT TO BE GENERATED	METHOD	
	M	C
3/16 S	$M_S + M_S/2$	$C_S - 3$
11/16 S	$(M_S + M_S/2)/4 + M_S$	$C_S - 1$
3/4 L	$M_L + M_L/2$	$C_L - 1$

The first approach was chosen since it requires less hardware to implement than the second approach. Normalization of the results of the first approach would also be simplified since none of the mantissas could overflow. This is not true of the second approach.

Since |I| and |Q| are in normalized form, normalizing the scaled L component will involve at most one left shift. Similarly, normalization of the scaled S component would take at most three left shifts. Thus, normalization could be done serially in a limited amount of time since only a small number of shifts are involved. The alignment of the scaled L and S components is accomplished in the same manner as the alignment process described in Section 4.3.1.

Since M_L and M_S are always positive, their sum is in normalized form unless an overflow occurs. Consequently, normalization of the sum would involve at most a right shift of one position. After normalization, the RMS output is sent to the post residue integrator.

4.3.3 Post Residue Integrator

The floating-point post residue integrator block diagram is shown in Figure 4.3. The inputs from the RMS unit are normalized positive floating point numbers. The control and alignment blocks function in the same manner as the control and alignment blocks of Figure 4.1. Since the numbers to be added are always positive, the sum needs to be normalized only if an overflow occurs.

After the required number of integrations, the output of the integrator is sent to the threshold detector. The threshold detector indicates the presence of a target if the integrator output exceeds the preset threshold value.

4.4 ALTERNATIVES

Several alternatives are available for realizing the floating-point MTI signal processor described in Section 4.3. Several of these alternatives and their effect on the hardware complexity are considered below:

1. The I an Q channel filter outputs could be converted to fixed-point numbers and the RMS approximation and post residue integration could then be realized using fixed-point arithmetic. This configuration would allow the use of floating-point coefficients (thus reducing the quantization error in the coefficients) but would require less hardware than a complete floating-point system. For example, of the six circuits of Figure 4.3 only the adder and accumulator circuits would be required if fixed-point arithmetic were used for the post residue integration. Furthermore, six of the circuits in Figure 4.2 would be eliminated (three normalize circuits, two align circuits, and the control circuit). This reduction in circuit complexity would outweigh the added circuitry needed for the conversion from floating-point to fixed point numbers.
2. The normalizing and aligning circuits could be combined. Since both the normalizing and aligning processes involve shifting of the mantissa bits of the floating-point numbers, one bidirectional shift circuit could be configured to perform both operations. The complexity of the control circuitry would be increased and would be strongly dependent on the number of bits used in the mantissas and characteristics but a net savings in hardware might be achieved.
3. Fixed-point coefficients could be used. The hardware required for determining the characteristics of the products in the I and Q channel filters would be simplified if fixed-point coefficients were used. In this configuration, the characteristic of the normalized product would be equal to the negative of the number of left shifts performed in the normalization. No sign bit would be required for the characteristic of the product since it would always be ≤ 0.

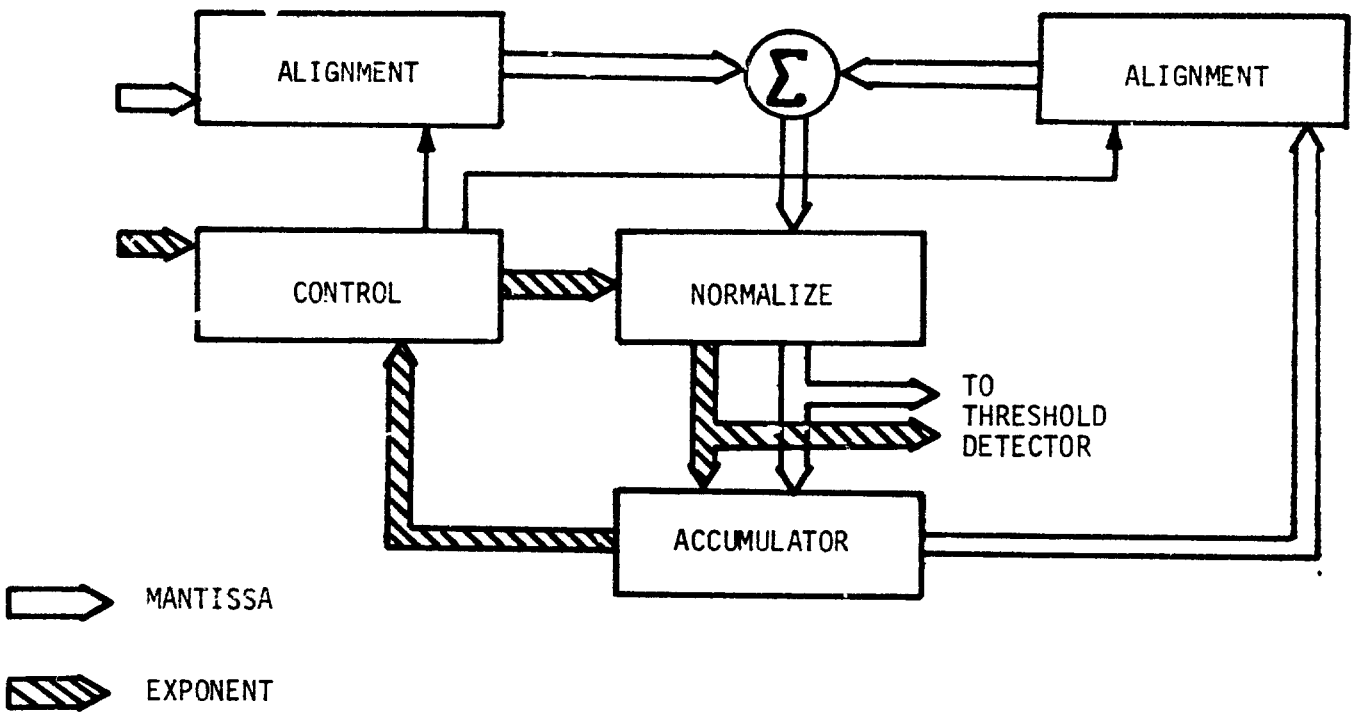


Figure 4.3 Floating-Point Post Residue Integrator Block Diagram.

The floating-point simulation described in Chapter 6 has an option which allows the use of fixed-point coefficients. Therefore, simulation runs could be made to determine the processor error performance for fixed-point and floating-point coefficients. These results would provide a basis for evaluating this alternative configuration.

4. Time multiplexing could be used to reduce the hardware complexity. Since the I and Q channel filters are identical, it would be possible to reduce hardware requirements by time-sharing some circuits. For example, one circuit could perform the magnitude conversion at the output of both the I and Q filters. It might be possible to multiplex other circuits such as the alignment control. Here again, the hardware tradeoffs are strongly dependent on the word lengths necessary.

In most cases, the hardware complexity of the circuits is strongly dependent upon the word lengths necessary to achieve a specified error performance. As the word lengths grow, a threshold is crossed where time constraints no longer allow serial operation. At that point, the complexity of the hardware rises significantly.

4.5 COMPARISON OF HARDWARE COMPLEXITY

The relative complexities of the hardware required to implement the fixed-point and floating-point versions of the MTI portion of the radar signal processor vary with the particular configurations used. However, the following general statements can be made:

1. The hardware required to implement the floating-point processor is approximately 2 to 5 times greater than that required in the fixed-point version.
2. The complexity of the floating-point hardware can be reduced by approximately 1/4 if the mantissa bit lengths are short enough that the aligning and normalizing operations can be done serially.
3. The amount of memory required for the fixed-point and floating-point systems is about equal. Preliminary simulation runs seem to indicate that fixed-point and floating-point processors with similar error performance require approximately equal word lengths (where the floating-point word length is the total bits used in the mantissa and characteristic).

More simulation runs need to be made in order to determine the bit lengths required at various points in the floating-point processor that are necessary to achieve specified error performance. Once these bit lengths are known, an accurate comparison of the actual integrated circuit chip counts necessary for the fixed-point and floating-point systems can be made.

CHAPTER 5

THEORETICAL ANALYSIS OF FLOATING-POINT PROCESSOR
by Jerry D. Moore

5.1 OUTLINE OF ANALYSIS

The theoretical analysis of the floating-point DSP quantization error is presented in this section. A system configuration as described in Chapter 4 is used for the analysis. The pattern established in Chapter 2 for the fixed-point processor is followed here, i.e., an outline is presented with the detailed derivations presented in an appendix.

The finite word lengths used in the DSP are identified in Fig. 5.1 and summarized in Table 5-1. It is important to notice that the A/D converter has a fixed-point output containing MX bits, as was used in the fixed-point analysis of Chapter 2. This choice was based on the availability of fixed-point A/D converters and the limited access to floating point A/D converters.

The output of the digital filter, w() is written as the sum of an errorless output y() and the quantization error g(), i.e.,

$$w(mN) = y(mN) + g(mN) \quad , \tag{5.1}$$

where

$$y(mN) = \sum_{n=0}^{N-1} h(n)\, x(mN - n) \quad . \tag{5.2}$$

The N filter coefficients are represented by h() and the input samples by x(). In Appendix C it is shown that

$$g(mN) = \sum_{n=0}^{N-1} h(n) \left[(\theta_{N-1-n} - 1)\, x(mN - n) + \theta_{N-1-n}\, e(mN - n) \right] \quad . \tag{5.3}$$

The θ_n terms are the statistical parameters used to relate the floating point quantization errors. An expression for this term is given in Appendix C.

An expression for the output of the RMS unit (cf., Chapter 2 Equation (2.7)) is

$$r(mN) = [1 + \gamma_w(mN)] \sqrt{w_I^2(mN) + w_Q^2(mN)} \quad . \tag{5.4}$$

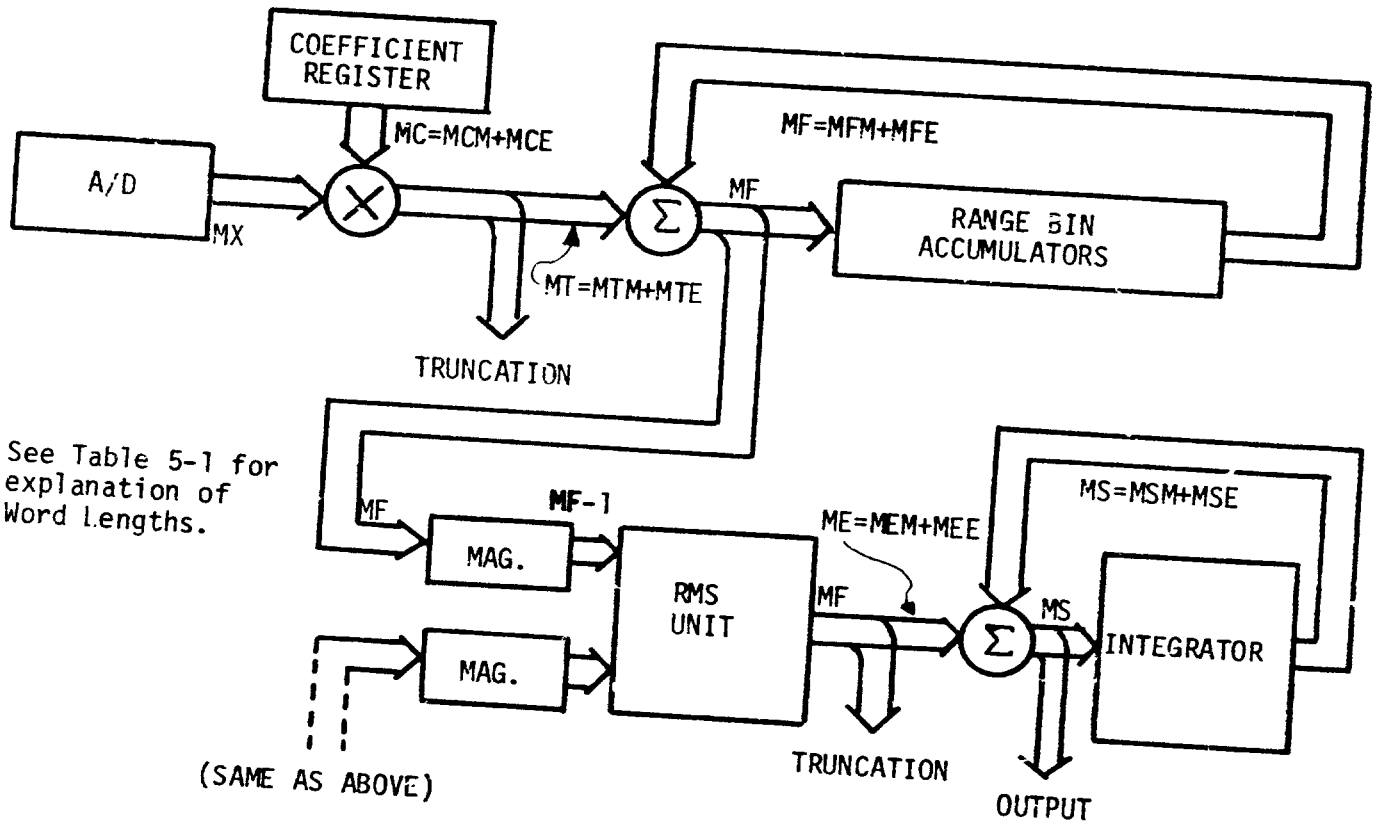


Fig. 5.1 Floating-Point Digital Signal Processor with Finite Word Lengths Indicated.

TABLE 5-1

FLOATING POINT DSP WORD LENGTH

Symbol	Description
MX	A/D Converter fixed-point word length. Two's complement form with 1 sign bit and MX - 1 fractional bits.
MCM	Coefficient mantissa word length. Two's complement form with 1 sign bit and MCM - 1 fractional bits.
MCE	Coefficient exponent word length. The exponent is in integer form and will always be negative, thus no sign bit is required.
MTM	Truncated product mantissa word length. Two's complement form with 1 sign bit and MTM - 1 fractional bits.
MTE	Truncated product exponent word length. Two's complement integer form with 1 sign bit and MTE - 1 integer bits.
MFM	Range bin accumulator mantissa word length. Two's complement form with 1 sign bit and MFM - 1 fractional bits.
MFE	Range bin accumulator exponent word length. Two's complement integer form with 1 sign bit and MFE - 1 integer bits.
MEM	Truncated residue mantissa word length. Magnitude form with MEM fractional bits. No sign bit is used.
MEE	Truncated residue exponent word length. Two's complement integer form with 1 sign bit and MEE - 1 integer bits.
MSM	Integrator accumulator mantissa word length. Magnitude form with MSM fractional bits. No sign bit is used.
MSE	Integrator accumulator exponent word length. Two's complement integer form with 1 sign bit and MSE - 1 integer bits.

This term is truncated and then summed in the integrator to give

$$INT(M) = \sum_{m=1}^{M} r(mN) + e_{int}(M) , \tag{5.5}$$

where the truncation and summing quantization error is (cf., Appendix C)

$$e_{int}(M) = \sum_{m=0}^{M-1} (\theta_{rm} - 1)\, r[(m+1)N] , \tag{5.6}$$

and θ_{rm} represents the floating-point error parameters as shown in Appendix C.

As in Chapter 2, the integrator output can be expressed as an errorless term, m·u, plus an error term, INTE(M), viz.,

$$INT(M) = M\,u + INTE(M) . \tag{5.7}$$

The average integrator output error is

$$\overline{INTE(M)} = \sum_{m=1}^{M} \bar{r} + \bar{e}_{int}(M) - M\,u . \tag{5.8}$$

It has been shown in Appendix C that the variance of INTE() is equal to the variance of INT() and that

$$\sigma^2_{INTE} = [M + 2 \sum_{m=0}^{M-1} (\bar{\theta}_{rm} - 1)]\, \sigma_r^2 + \sigma^2_{int} . \tag{5.9}$$

From (5.6) it follows that

$$\sigma^2_{int} = \sum_{m=0}^{M-1} \{\overline{(\theta_{rm} - 1)^2\, r^2[(m+1)N]} - \overline{(\theta_{rm} - 1)}^2\, \overline{r[(m+1)N]}^2\}$$

$$\bar{e}_{int} = \sum_{m=0}^{M-1} \overline{(\theta_{rm} - 1)\, r[(m+1)N]} = \bar{r} \sum_{m=0}^{M-1} (\bar{\theta}_{rm} - 1) . \tag{5.10}$$

A bounding procedure is necessary to evaluate $\bar{r}$ and $\overline{r^2}$, cf., (2.15), (2.16), (2.17), and (2.18), i.e.,

$$(1 + \bar{\gamma}_w)\,|u - \bar{v}| \leq \bar{r} \leq (1 + \bar{\gamma}_w)(u + \bar{v}) \,, \tag{5.11}$$

where

$$\sqrt{2}\,|\bar{g}| \leq \bar{v} \leq \sqrt{2\,\overline{g^2}} \,, \tag{5.12}$$

and

$$\overline{(1 + \gamma_w)^2}\;\overline{(u - v)^2} \leq \overline{r^2} \leq \overline{(1 + \gamma_w)^2}\;\overline{(u + v)^2} \,. \tag{5.13}$$

After manipulation and use of (5.12)

$$\bar{r} \leq (1 + \bar{\gamma}_w)(u + \sqrt{2\,\overline{g^2}}) \,, \tag{5.14}$$

$$\bar{r} \geq \mathrm{SM}\left[(1 + \bar{\gamma}_w)\,|u - \sqrt{2\,\overline{g^2}}|\,,\ (1 + \bar{\gamma}_w)\,|u - \sqrt{2}\,|\bar{g}|\,|\right] \,, \tag{5.15}$$

$$\overline{r^2} \leq \overline{(1 + \gamma_w)^2}\,(u + \sqrt{2\,\overline{g^2}})^2 \,, \tag{5.16}$$

$$\overline{r^2} \geq \overline{(1 + \gamma_w)^2}\,(u - \sqrt{2\,\overline{g^2}})^2 \,, \tag{5.17}$$

where SM[ ] indicates the smallest of the set.
It follows that

$$\sigma_r^2 \leq \overline{r^2}_{\max} - \bar{r}^2_{\min} \,, \tag{5.18}$$

and

$$\sigma_r^2 \geq \mathrm{LA}[(\overline{r^2}_{\min} - \overline{r^2}_{\max}),\ 0] \,, \tag{5.19}$$

where LA[ ] indicates the largest of the set.
The determination of the average value and variance of the error of (5.8) and (5.9) through the use of (5.10), (5.14) - (5.19) thus depends on the evaluation of $\bar{g}$ and $\overline{g^2}$. It is shown in Appendix C that

$$\overline{g} = \frac{A}{\pi} \sum_{n=0}^{N-1} h(n)\ (\overline{\theta}^{+}_{N-1-n} - \overline{\theta}^{-}_{N-1-n}) , \qquad (5.20)$$

where A is the peak value of the sinusoidal doppler input to the filter, h() represents the filter coefficients, and $\overline{\theta}^{+}$ and $\overline{\theta}^{-}$ represent the floating point error statistics for positive and negative values of x() respectively. Also, it is shown in Appendix C that

$$\overline{g^2} = \frac{A^2}{\pi^2} \sum_{\substack{n=0 \\ n\neq k}}^{N-1} \sum_{k=0}^{N-1} h(n)h(k)(\overline{\theta}^{+}_{N-1-n} - \overline{\theta}^{-}_{N-1-n})(\overline{\theta}^{+}_{N-1-k} - \overline{\theta}^{-}_{N-1-k}) \qquad (5.21)$$

$$+ \sum_{n=0}^{N-1} \left\{ h^2(n) \left[\left(\frac{A^2}{4} + \frac{\overline{e^2}}{2}\right)(\overline{\theta^2}^{+}_{N-1-n} + \overline{\theta^2}^{-}_{N-1-n}) \right.\right.$$

$$\left.\left. + \frac{A^2}{2}\ [1 - (\overline{\theta}^{+}_{N-1-n} + \overline{\theta}^{-}_{N-1-n})] \right] \right\} .$$

The $\overline{e^2}$ is the mean squared value of the quantization error associated with the input signal x().

Evaluations of the floating-point parameters θ_{rm} and θ_n and their statistical values are given in Appendix C This concludes the analysis for the floating-point processor. It is possible to predict and upper and lower bound on the integrator output average error and variance by using the results of this section. A computer program was written to implement the equations of this analysis. The program listing is given in Appendix D and was used to obtain the results presented in the next section.

5.2 GRAPHICAL PRESENTATION OF RESULTS

The computer program of Appendix D was used to obtain specific values for the integrator output error statistics of a floating-point processor. As in Chapter 2, the A/D converter word is a fixed-point word with MX = 9. The coefficients were also treated as fixed-point words with MC = 9. This choice results in simpler hardware requirements for the processor and allows a convenient comparison with the fixed-point processor results. The product truncation length is MT bits with MTM bits for the mantissa. This quantity, MTM, is varied as is the integrator input word length mantissa, MEM. Other word lengths were fixed, viz., MFM = MTM and MSM = MEM Such choices seem to be justified by hardware considerations. Many other choices could be made.

The average error bounds are presented in Fig. 5.2 as a function of MTM with MEM = 10. The bounds are not tight, but do become closer as the signal amplitude is reduced. The bounds are not strongly dependent on MTM for values between 11 and 15. The bounds do not appear to be as dependent on signal amplitude as was the fixed-point processor.

Upper bounds on the error variance are shown in Fig. 5.3 as a function of MTM with MEM = 10. The lower bound values were either zero or very small numbers. The variance bounds are dependent on the signal amplitude, but independent of the MTM values between 11 and 15.

The integrator word length effects are demonstrated in Figures 5.4 and 5.5 for MTM = 13. First the 0.125 volt signal amplitude is used and then 0.413 volts. The variance bounds are almost independent of MEM, but the average error bounds are strongly dependent on MEM for values less than 10. The best values of MEM for the upper bounds are MEM = 8 for 0.125 volts and MEM = 9 for 0.413 volts. The best values of MEM for the lower bounds are any MEM > 12. Fig. 5.6 is an expanded version of the variance upper bound versus MEM with MTM as a parameter family. Note that the smaller the amplitude becomes then the less dependency on MTM, e.g., for 0.413 changes are not noticed for MTM > 12, for 0.125 volts changes are not noticed for MTM > 10, while for 0.025 volts the results do not change for MTM > 9. The reduction in the variance bounds for low MEM values was not expected. Even though the reduction of MEM would appear to produce more variations, the bound has a canceling effect by reducing the dependence of the residue variance σ_r^2. This effect is not anticipated for the simulation results.

The variance bounds are presented as a function of signal amplitude with MTM = 13 in Fig. 5.7. There is not a strong dependence on MEM and the lower bound is zero for values not shown.

The results of this section are compared to the floating-point simulation results in Chapter 6.

A follow-on analysis using a two-sector approximation RMS error of 0.1265% was made. The results of Fig. 5.7 revealed a maximum decrease by a factor of 2 at 0.413 volts and insignificant changes for amplitude less than 0.1 volts.

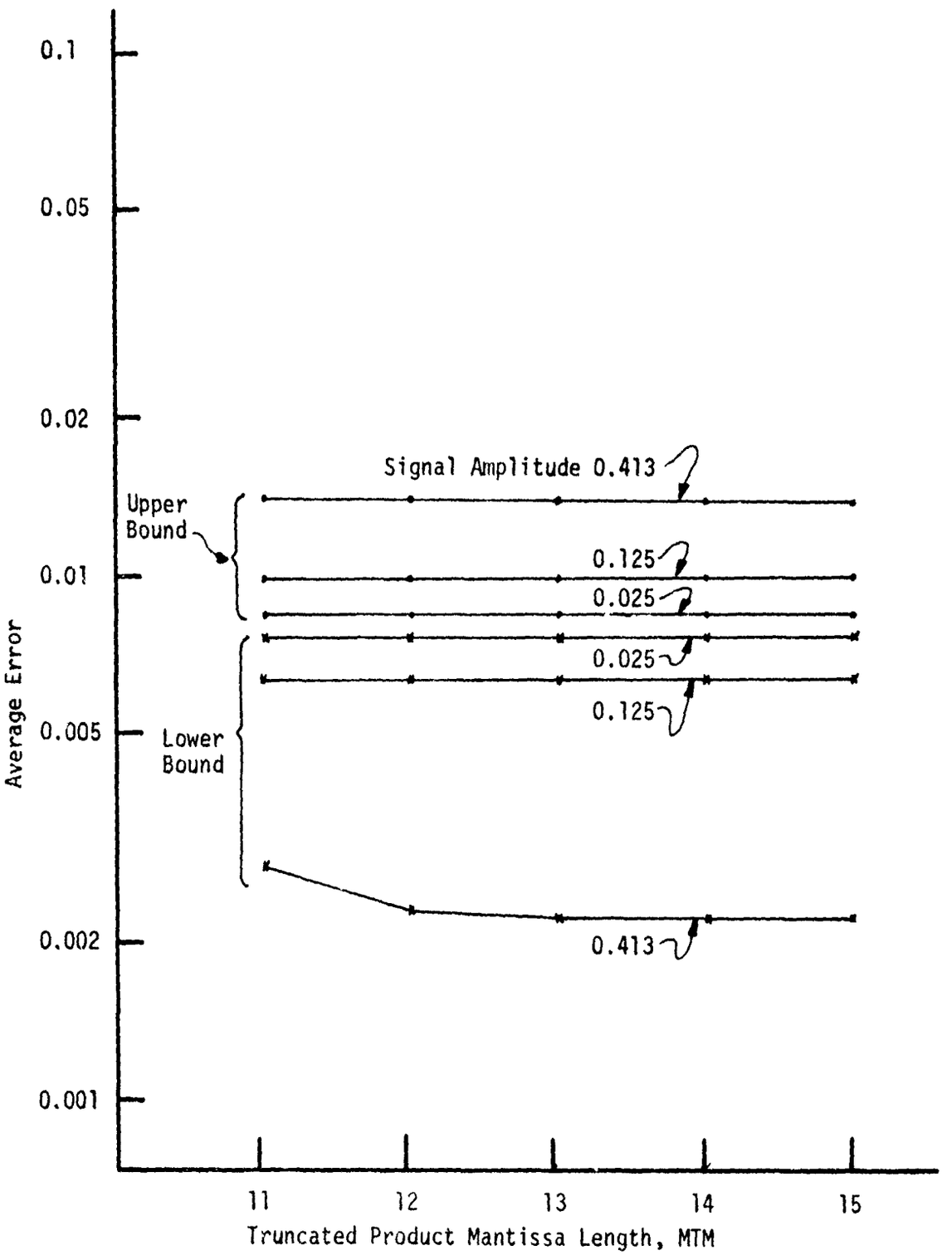


Fig. 5.2 Bounds on Average Integrator Error as Function of Truncated Product Length with Truncated Residue Mantissa Length MEM = 10 (Frequency = 1500 Hz)

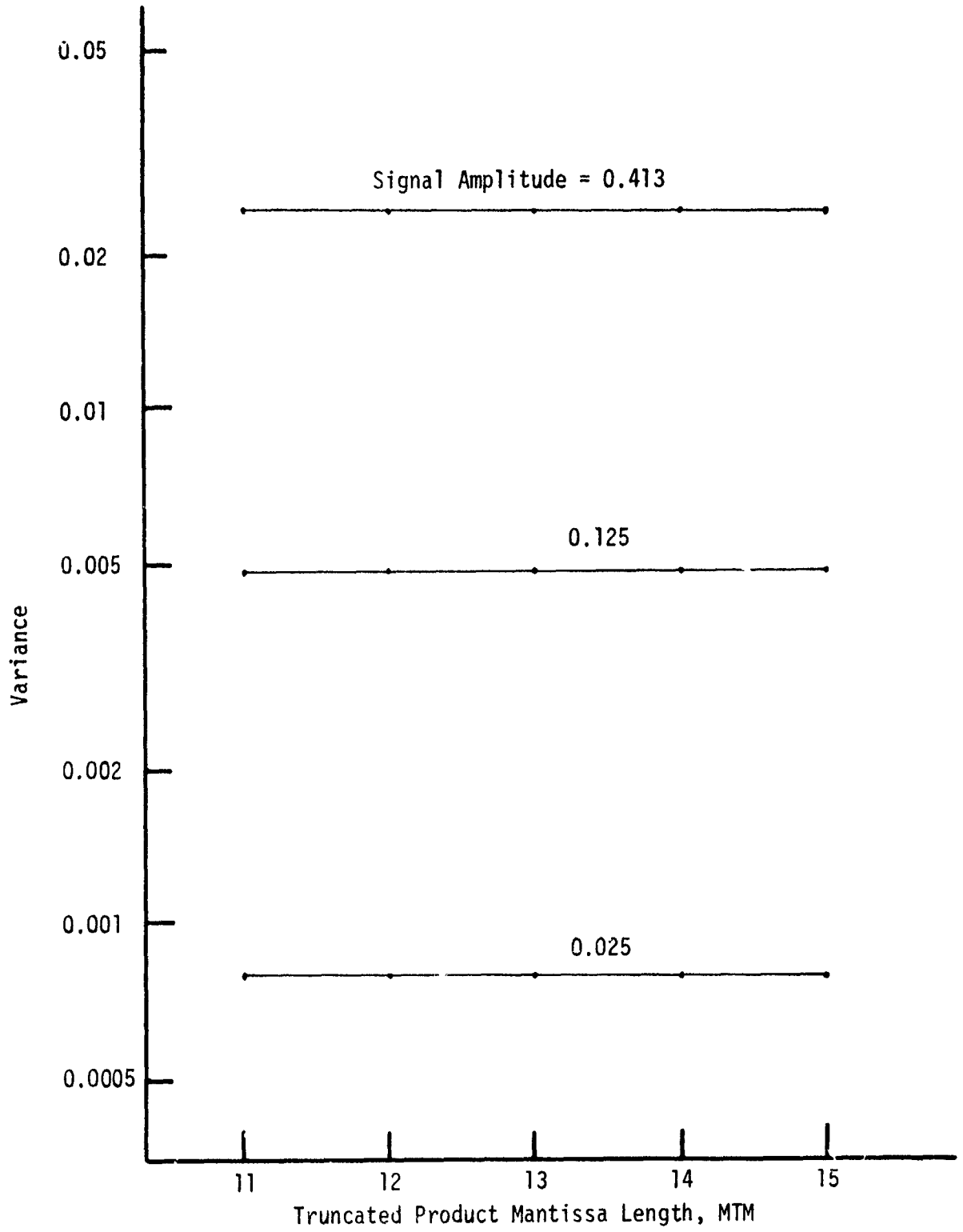


Fig. 5.3 Upper Bound on Integrator Error Variance as Function of Truncated Product Length with Truncated Residue Mantissa Length MEM = 10 (Frequency = 1500 Hz)

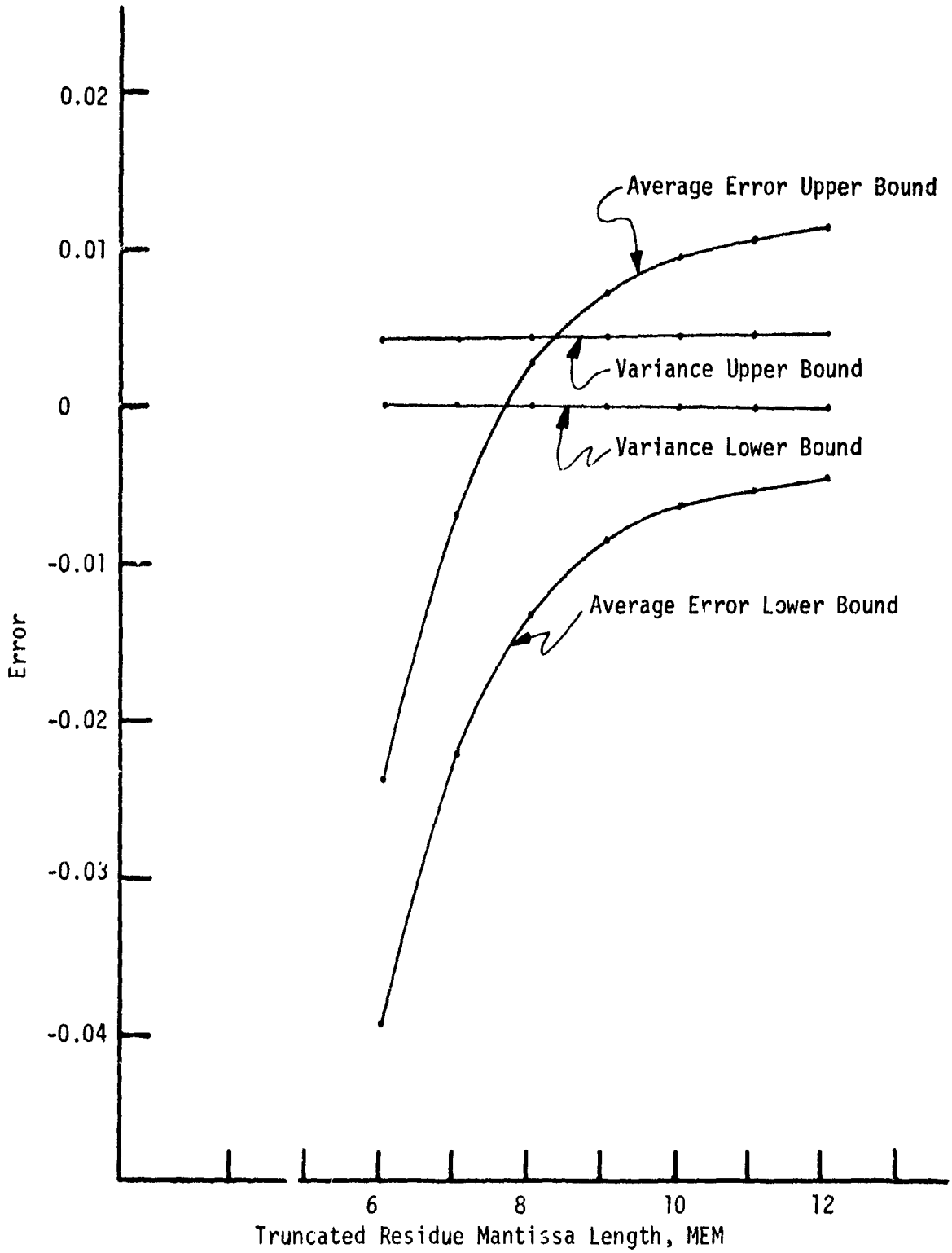


Fig. 5.4 Comparison of Bounds as Function of Truncated Residue Length with Truncated Product Mantissa Length MTM = 13 (Signal Amplitude = 0.125V, Frequency = 1500 Hz)

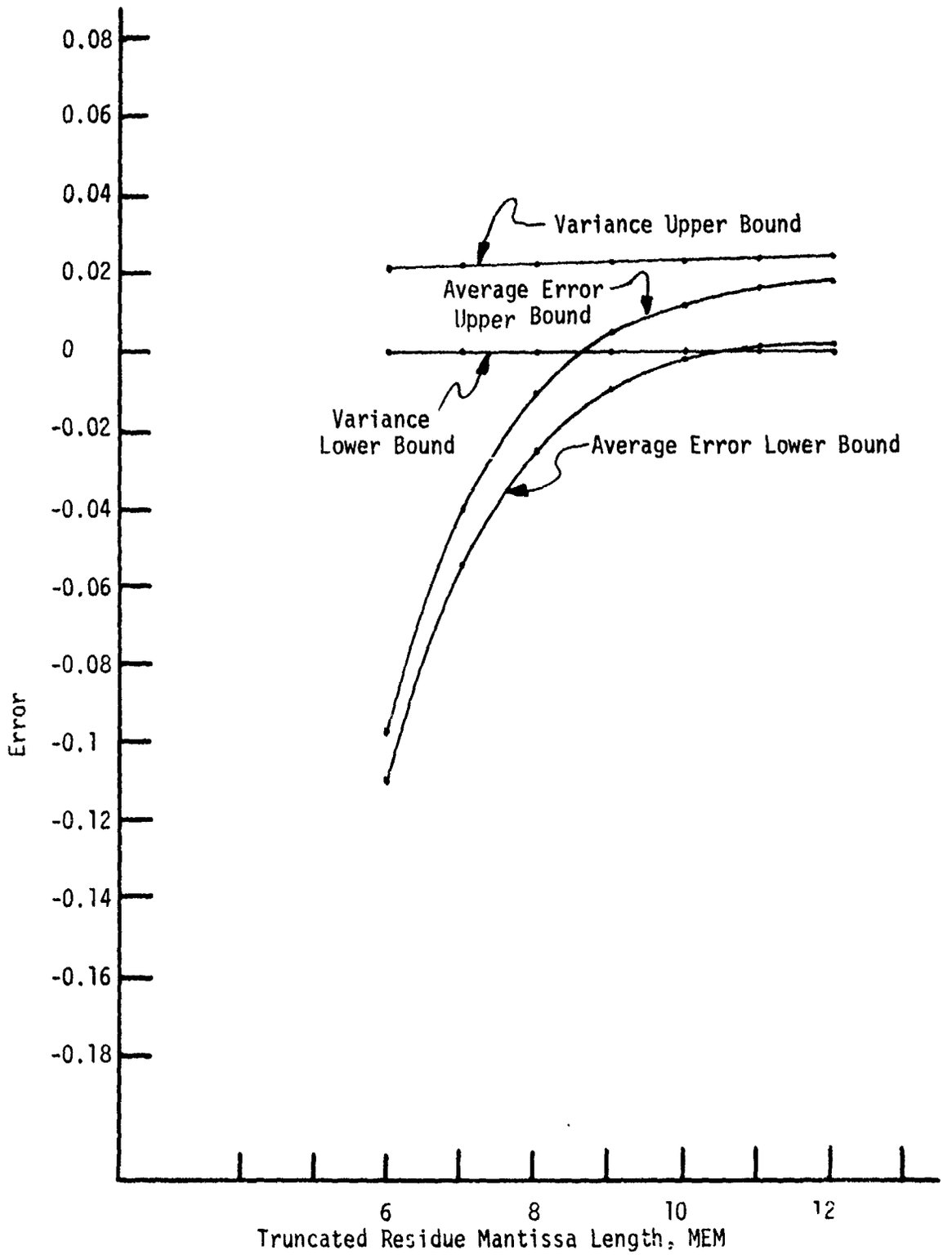


Fig. 5.5 Comparison of Bounds as Function of Truncated Residue Length with Truncated Product Mantissa Length MTM = 13 (Signal Amplitude = 0.413V, Frequency = 1500 Hz)

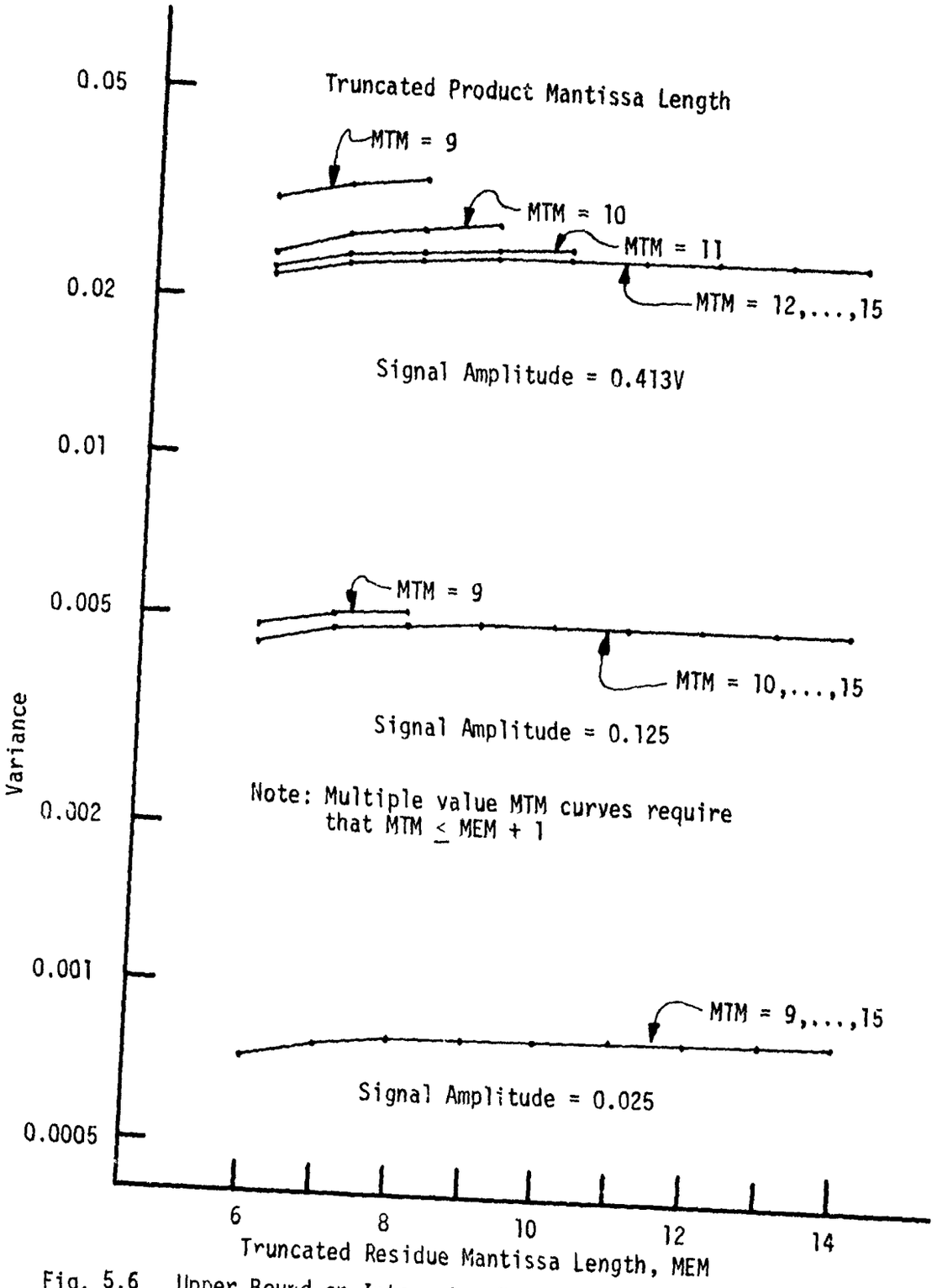


Fig. 5.6 Upper Bound on Integrator Error Variance as Function of Truncated Residue Length (Frequency = 1500 Hz)

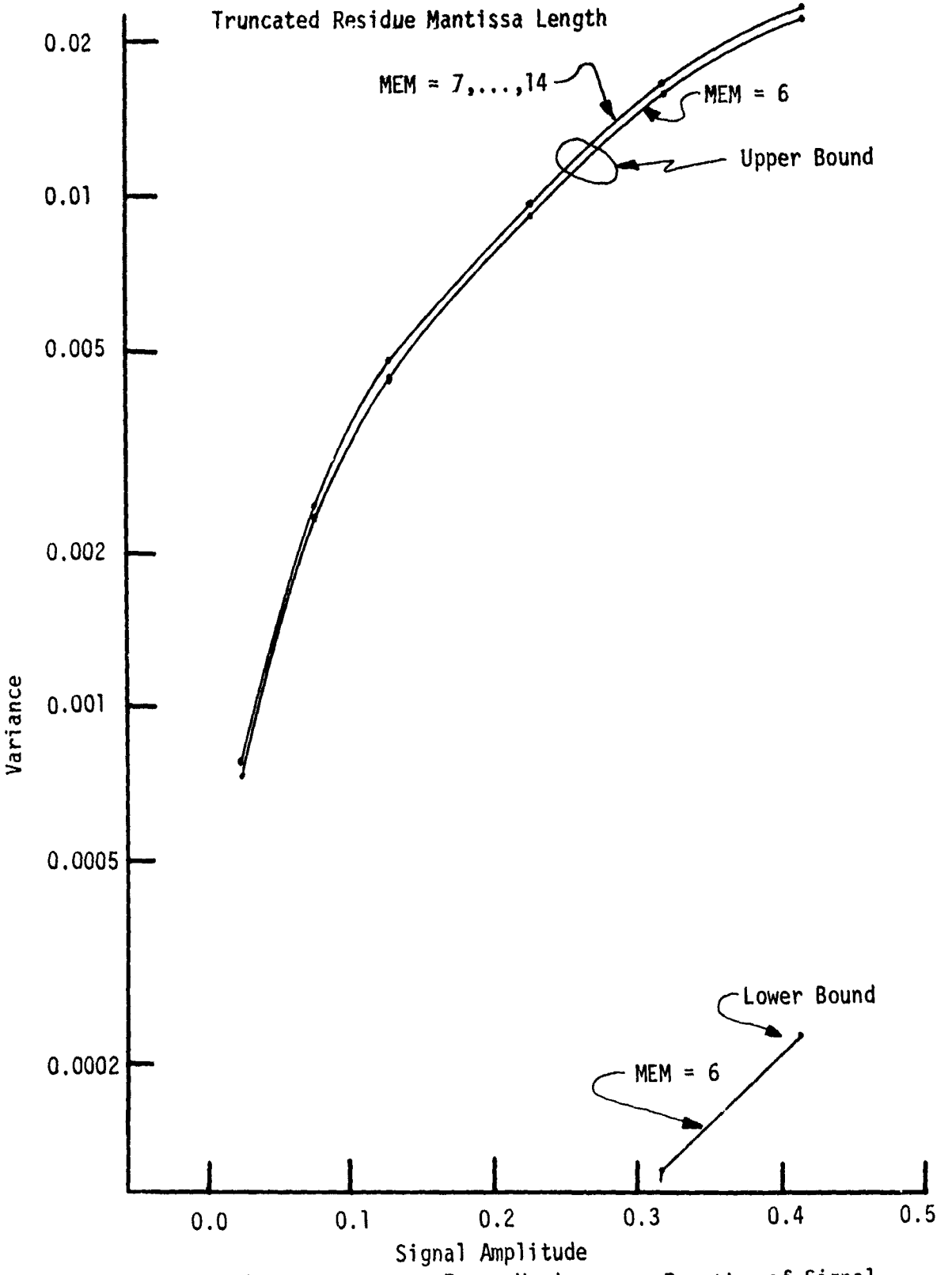


Fig. 5.7 Bounds on Integrator Error Variance as Function of Signal Amplitude with Truncated Product Mantissa Length MTM = 13 (Frequency = 1500 Hz)

CHAPTER 6

FLOATING-POINT SIMULATION PROGRAM
by Bhadrayu J. Trivedi and Brian P. Holt

A FORTRAN program was developed to perform the simulation study of the floating-point processor. A description of the program follows in Section 6.1 and some typical simulation results are discussed in Section 6.2.

6.1 DESCRIPTION OF PROGRAM

The floating-point simulation program was patterned after the structure of the fixed-point simulation program. The suggested improvements of Section 3.1.3 were incorporated in this program. The technique for representing the floating-point binary numbers and simulating the arithmetical operations was markedly different from the fixed-point case. However, it did follow the principle of representing all binary numbers as positive decimal integers. The programming technique is described in Section 6.1.1. The routines to simulate the basic arithmetic operations, other system blocks and functions, and the overall program are described with detailed flow charts in Section 6.1.2. The floating-point simulation program is listed in Appendix F with the details of data card formats.

6.1.1 Programming Techniques

In the floating-point processor discussed in Chapter 4, numbers are represented as

$$\text{NUMBER} = M \cdot 2^{C} \tag{6.1}$$

where M is the mantissa in two's complement binary with sign and C is the characteristic (often called exponent) in signed magnitude form. For the purpose of Fortran simulation the mantissa was represented in the manner identical to the fractional numbers in the fixed-point processor. This technique was described in detail in Section 3.1.1. The characteristic is always an integer and is therefore converted to its decimal equivalent with a sign for the purpose of simulation.

This approach, for the Fortran representation of the mantissa and exponent, permitted the use of some of the fixed-point simulation routines; viz., ADD for addition, MUL for multiplication, ITREX for truncation of the mantissa, and MAGNF for finding the magnitude of the mantissa. It is necessary to align two numbers to be added so that they have the same exponents. Also, it is necessary to normalize the result of an addition, multiplication or truncation operation (as discussed in Section 4.1). Two routines ALIGN and JUSTFY were written for this purpose and are described in the next section. It was also necessary to modify ADD slightly for the reasons discussed in the next section.

6.1.2 Flow Chart of Program

In this section the MAIN program is described first. Next, the routines ADD, JUSTFY and ALIGN are described. These routines are used with function subprograms MUL, MAGNF and ITREX to form the group of routines which simulate basic arithmetic operations. The latter three are described in Section 3.1.2 and are not discussed here. The routines FLCOEF, IAD and SACGEN perform the functions of coefficient quantization, input sample quantization and signal-clutter combination respectively. IAD was described in Section 3.1.2 and so was SACGEN but with a different name PULSEQ. FLCOEF was prepared to quantize the MTI filter coefficients into floating-point numbers. Its counterpart for the fixed-point program was COEF. After FLCOEF the routines FLOFLT and RMSHAL are described. These simulate the hardware MTI digital filter and the RMS unit (also referred to as the vector-magnitude unit) for the floating-point processor.

The overall program simulates the floating-point processor discussed in detail in Section 4.3. Like the fixed-point simulation program it generates results that can be used in a statistical study of quantization errors. The MAIN program flow charted in Fig. 6.1, starts the simulation by reading in signal, clutter, radar and filter parameters which it subsequently prints out. It then reads in the mantissa and characteristic bit-lengths to be maintained at different points in the processor. Next, a set of parameters are read which control several different functions, viz., an option for implementing the hardware RMS approximation algorithm, an option to include clutter in simulation or not, an indicator to print out the integrator output statistics after a fixed number of dwells, start and stop indicators to obtain detailed debugged simulation printouts for all dwells between the specified limits, and an option which controls whether the theoretical output is to be computed with a quantized or unquantized set of filter coefficients. All the parameters read in by the MAIN program are explained in Appendix F with the details of how they are specified on input cards. Then, the clutter filter impulse response is modelled, and scale factors for clutter and signal combination are generated and printed (if clutter is to be used in the simulation). The MAIN program simulates one system block, viz., the post-residue integrator. All the other system blocks are delegated to routines which MAIN calls. It calculates the output statistics for the hardware RMS unit and a hypothetical perfect RMS unit at the end of each residue. The statistics involve computation of the maximum, the minimum, the mean and the variance of the error in the residue output. The error is defined as the actual output minus the theoretical output (see Equations 5.1, 5.5, 5.7). The statistics for the difference between the hardware and perfect RMS outputs are also computed. A similar statistical analysis is carried out on the integrator outputs, i.e., at the end of each antenna dwell. The nature of the statistics is biased as defined in Equation (3.4). Note that the simulation results presented in Section 6.2 pertain to the integrator output statistics. The MAIN program calls RANDU for picking uniformly distributed random phase starting angles for the doppler signal and calls RANDM for generating a set of Gaussian random numbers to be used for simulating clutter. The program also monitors and indicates the total number of A/D converter saturations in a simulation run. The dis-

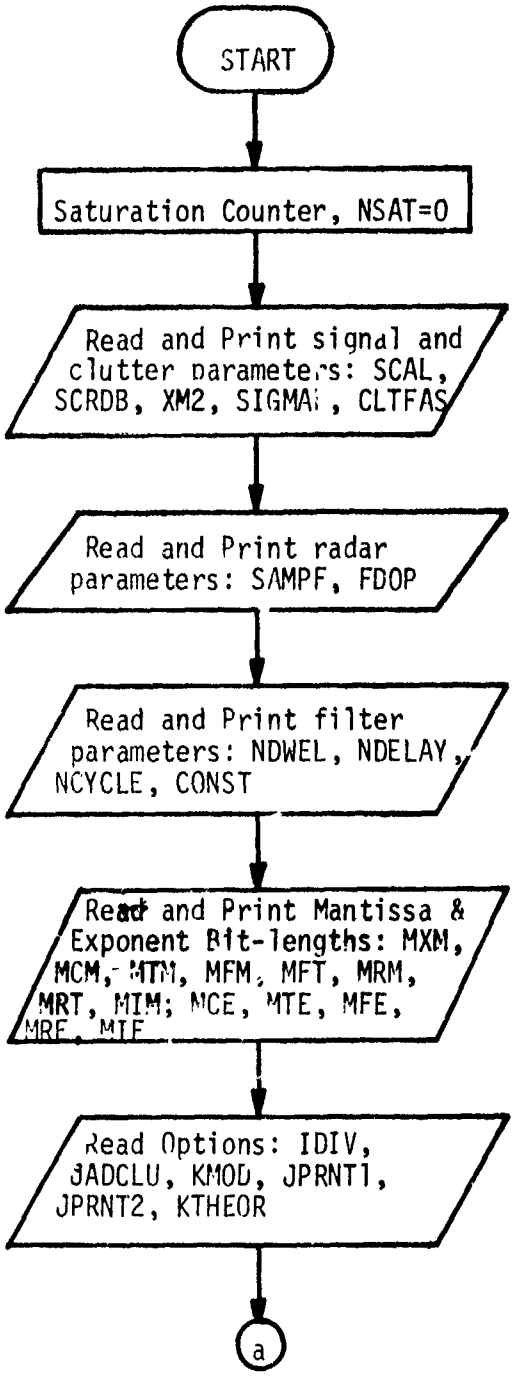


Fig. 6.1 Flow Chart for the MAIN Program of the Floating-Point Processor Simulation Program.

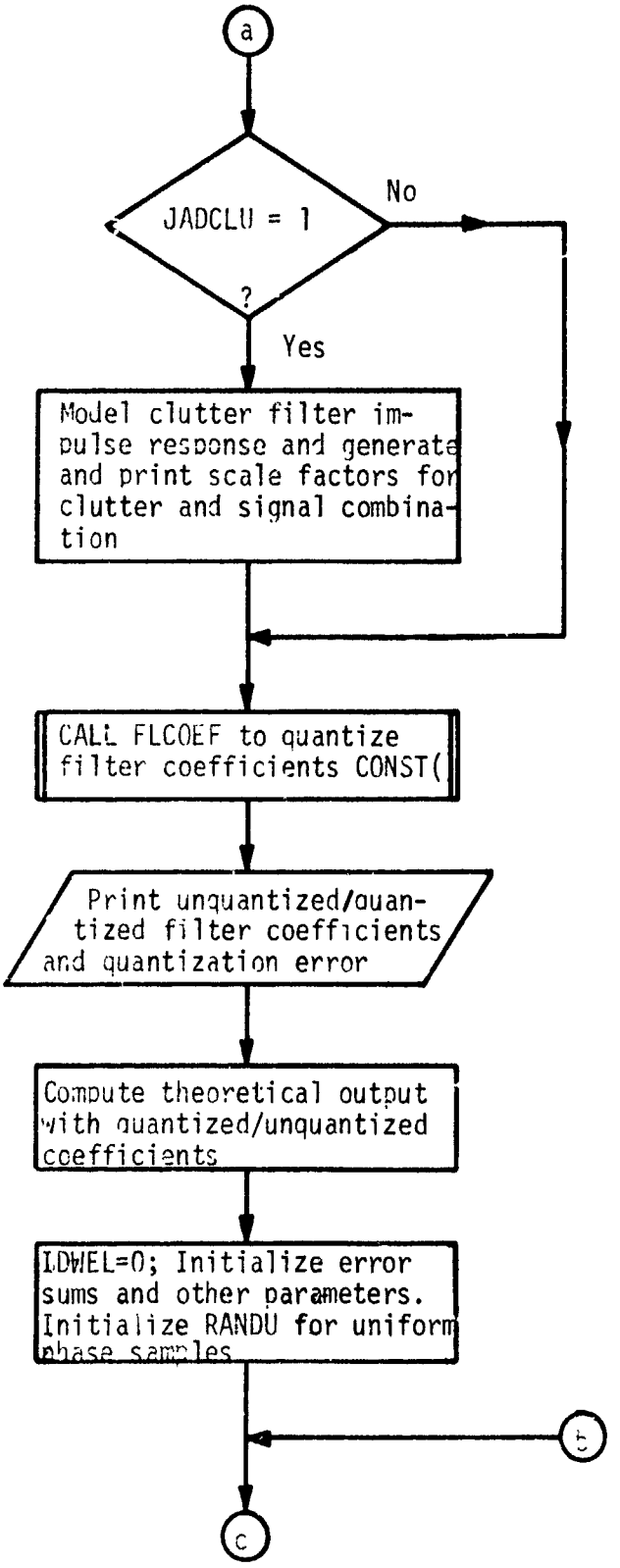


Fig. 6.7 (Continued)

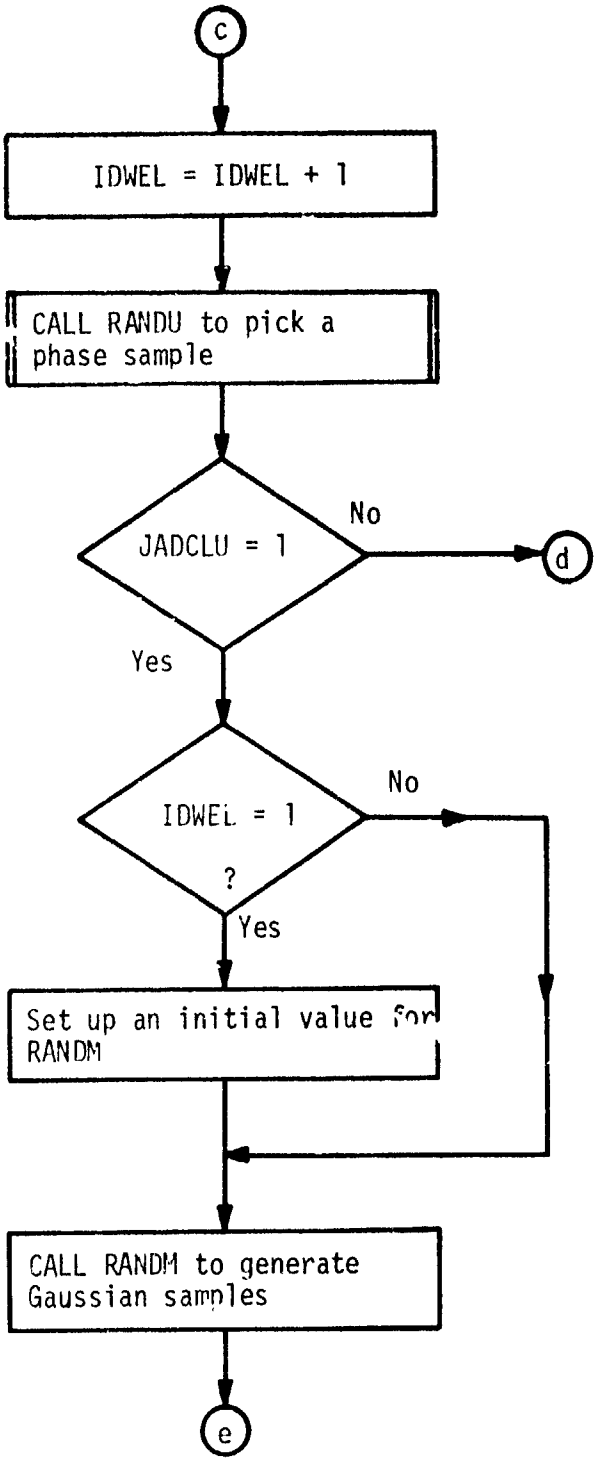


Fig. 6.1 (Continued)

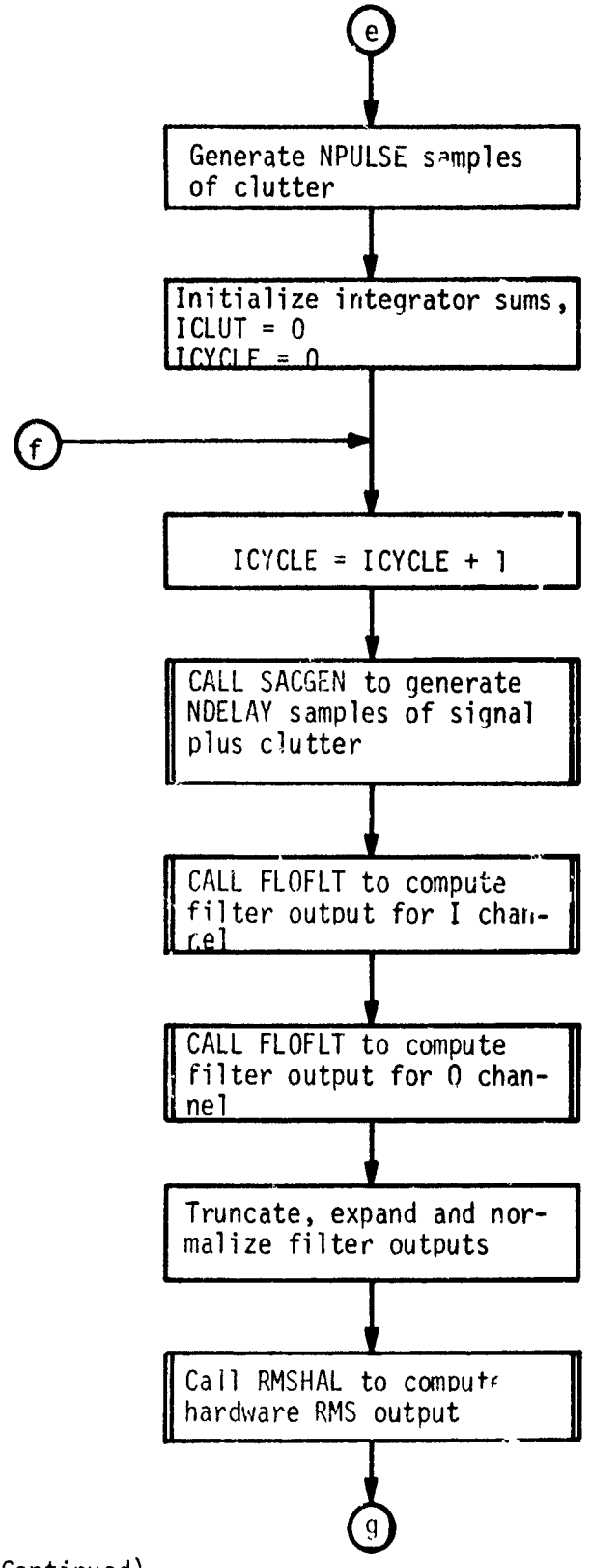


Fig. 6.1 (Continued)

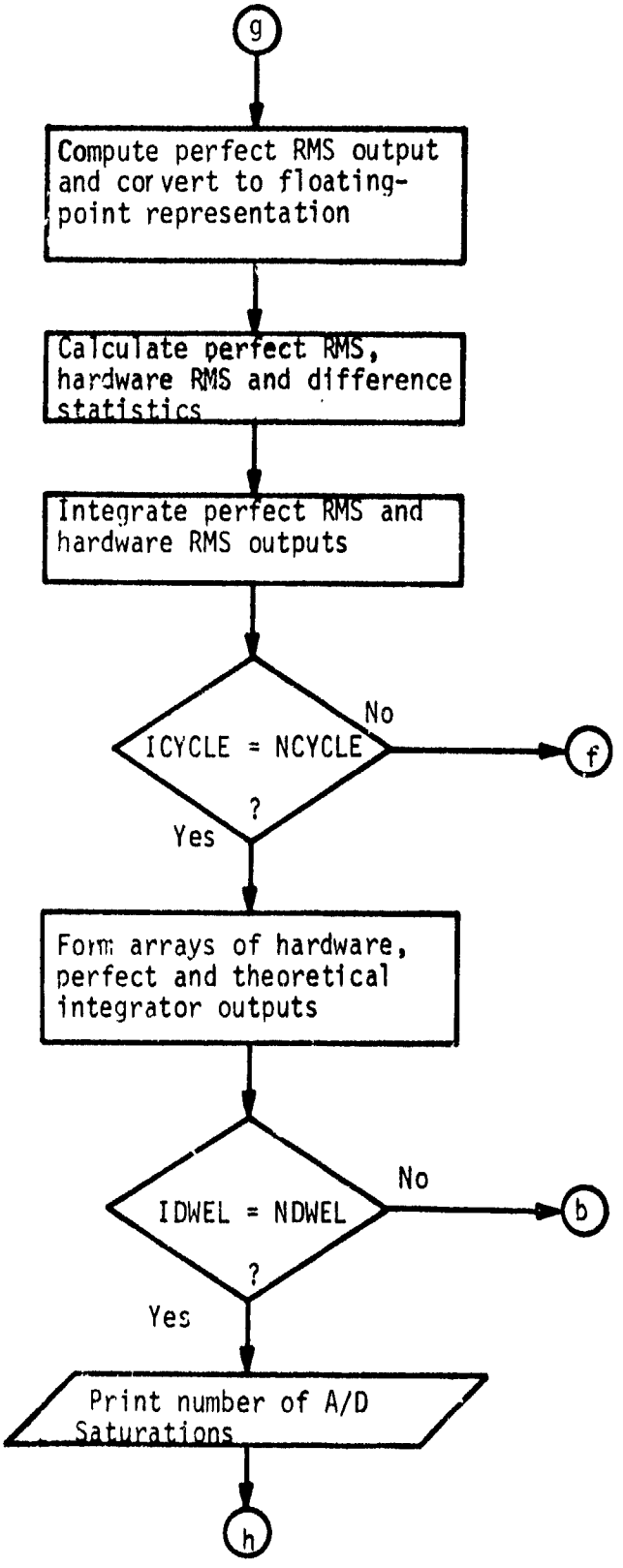


Fig 6.1 (Continued)

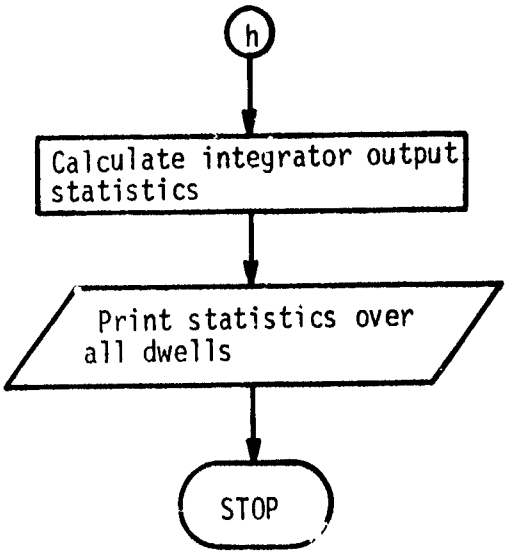


Fig. 6.1 (Continued)

cussion here and the flow chart of Fig. 6.1 present the important features and a brief outline of MAIN since it is a very large sized program. But, when used with the detailed comments of the listing (in Appendix F), the working of MAIN can be easily traced.

The flow chart for subroutine JUSTFY is shown in Fig. 6.2. JUSTFY accepts the mantissa and the exponent of an unjustified or unnormalized number with M < 0.5 and gives back the mantissa and the exponent of the resulting justified number. The mantissa is shifted left until a 1 appears in the most significant bit position of a positive number or a 0 appears in the most significant bit position of a negative number. The exponent is reduced by the number of shifts the mantissa undergoes. If the unjustified number is zero then its exponent is set to the most negative value,EMIN,specified to the routine. Note that JUSTFY does not normalize a number which has overflowed into the integer portion of the mantissa as a result of an addition. Such a number is normalized by correcting for overflow by the program that calls ADD. This involves dividing the mantissa by 2 and incrementing the exponent by 1.

The subroutine ALIGN accepts the mantissas and the exponents of two justified numbers, determines the number with smaller exponent, increments its exponent and divides its mantissa by 2 until the two exponents are equal. The algorithm is illustrated in Fig. 6.3.

The modified subroutine ADD as flow charted in Fig. 6.4 accepts the mantissas of two aligned numbers. The modification implies that the carry of the result is inserted back into the number if an overflow occurred so that the program which called ADD can correct for it and normalize the result. Except for this modification the routine is exactly the same as used for the fixed-point simulation.

The subroutine FLCOEF quantizes filter coefficients in either fixed-point (exponent value set to zero) or floating-point representations by using the concept of quantization interval defined in Section 3.1.1. The floating-point representation requires that the quantization interval be modified depending upon the magnitude of the coefficient and the number of bits available for the exponent. Once the quantization interval and the exponent are determined the mantissa is quantized just like in the fixed-point simulation routine COEF. However, if an overflow occurs in the rounding of the mantissa then it is corrected unlike in COEF. Finally, as shown in the flow chart in Fig. 6.5, FLCOEF calls JUSTFY to normalize the number.

For the floating-point processor, the fixed-window non-recursive MTI digital filter is simulated by the subroutine FLOFLT. It is used for both the in-phase and quadrature channel filtering. Its flow chart appears in Fig. 6.6. It calls IAD to quantize the signal sample obtained from SACGEN. The signal sample is represented as a two's complement fixed-point number. Next, FLOFLT uses MUL to multiply the quantized signal sample by the respective quantized coefficient mantissa. This is followed by the normalization of the product by JUSTFY. Next, the product mantissa is truncated by using ITREX and is again normalized by calling JUSTFY. Then subroutine ALIGN is called to align the product and the accumulator sum for addition. They are added with a call to ADD and then corrected if a mantissa overflow occurred or normalized by a call to JUSTFY. Finally, after NDELAY (total number of coefficients) additions,

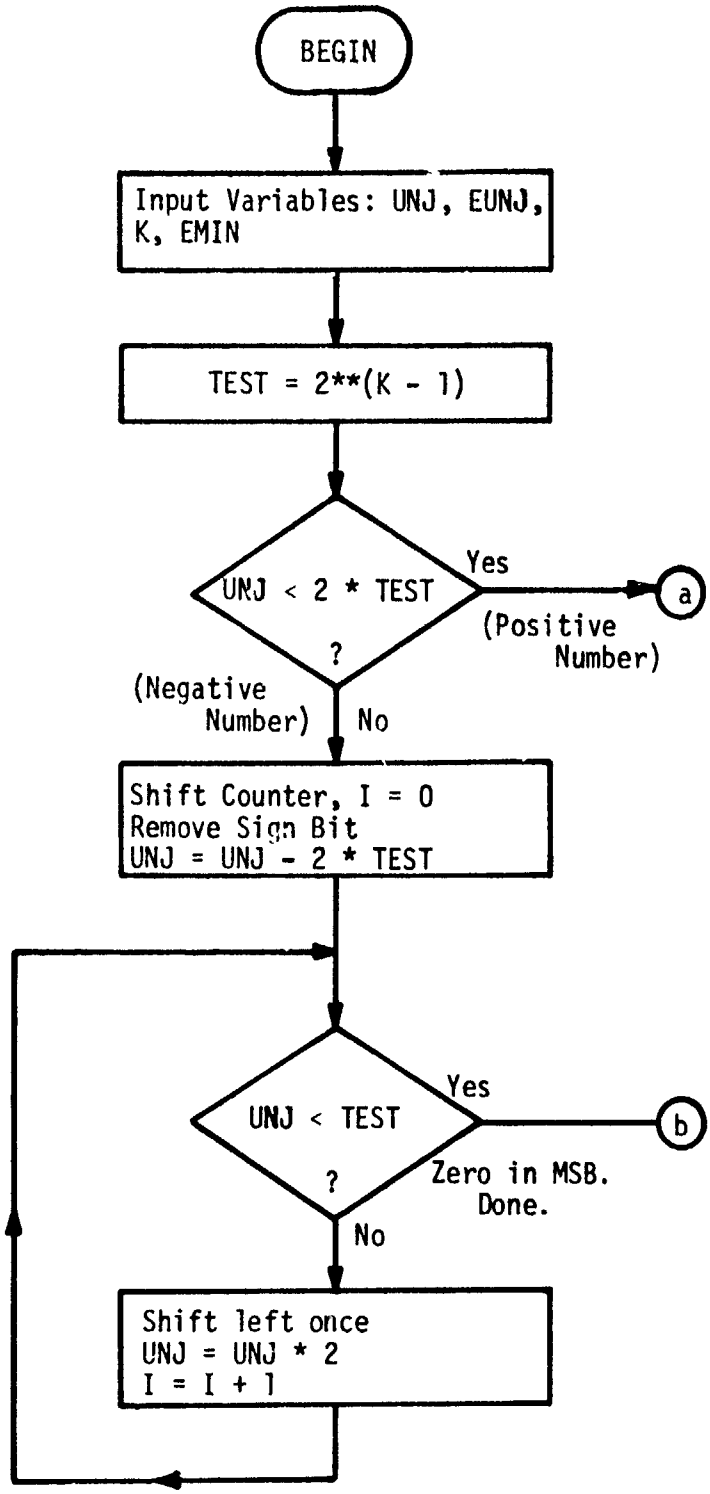


Fig. 6.2 Flow Chart for Subroutine JUSTFY.

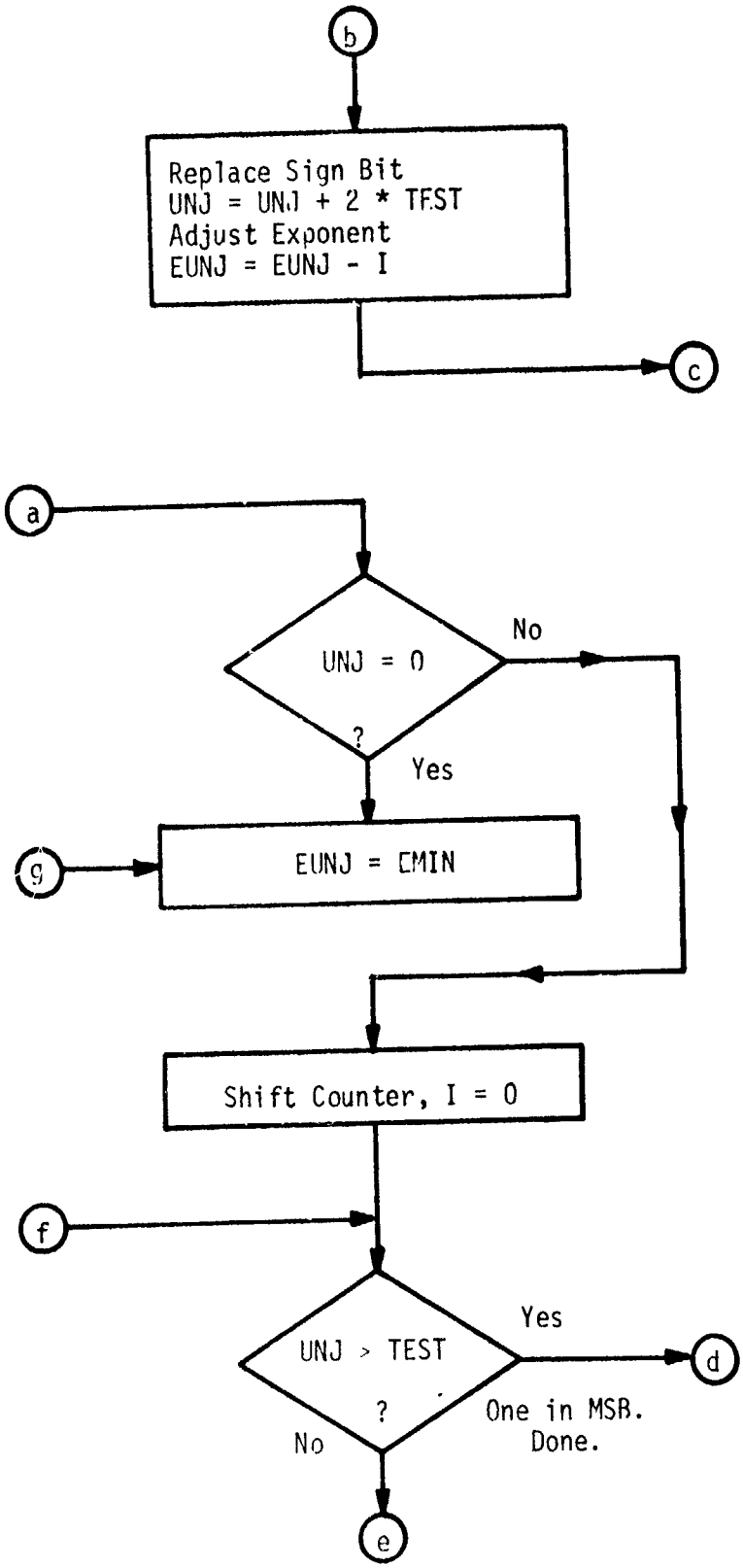


Fig. 6.2 (Continued)

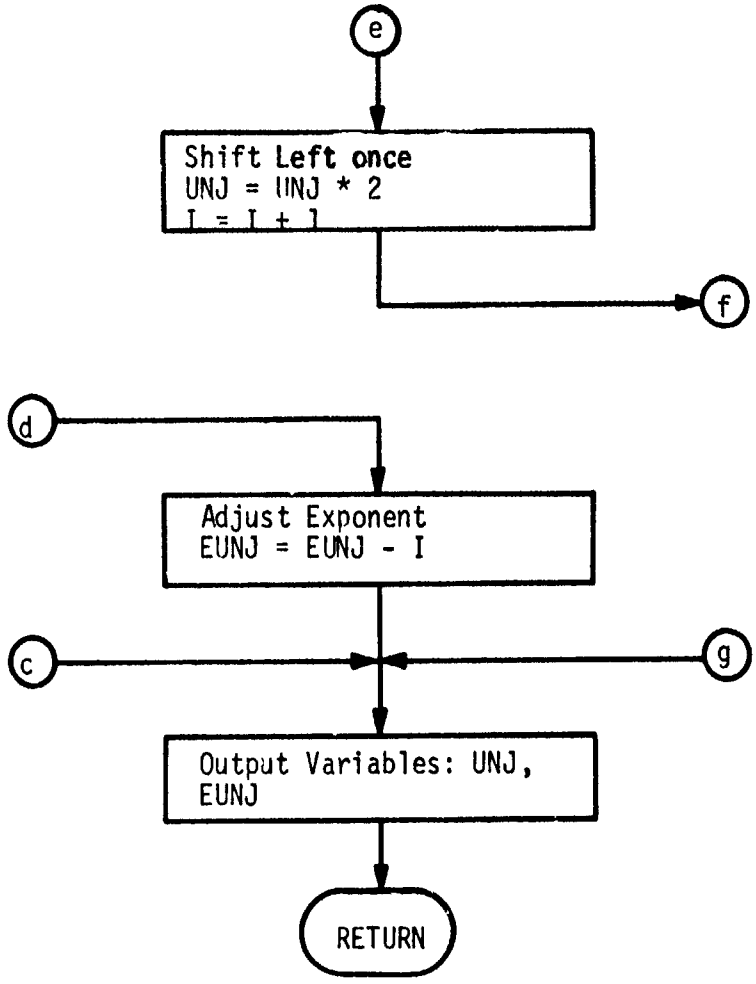


Fig. 6.2 (Continued)

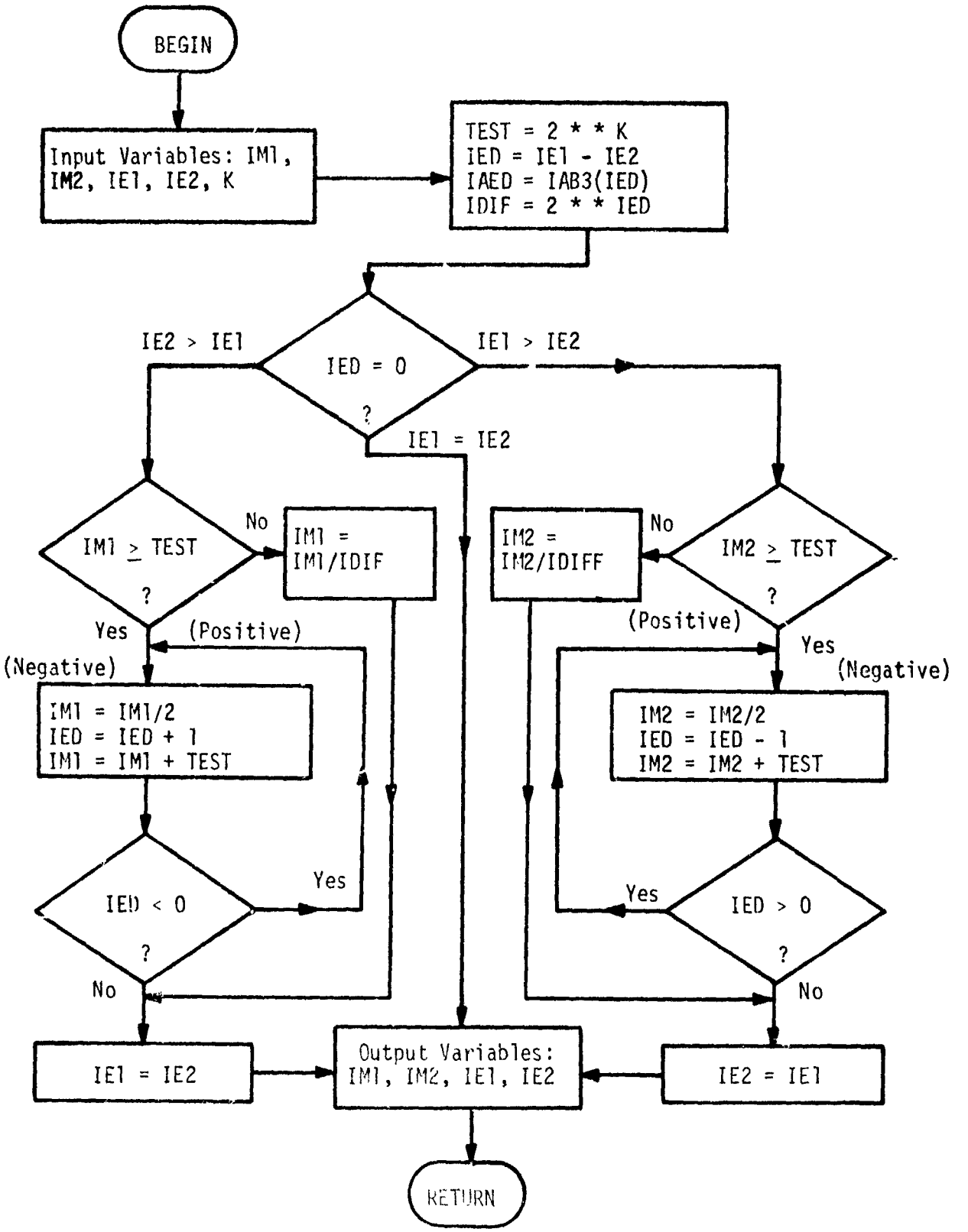


Fig. 6.3 Flow Chart for Subroutine ALIGN.

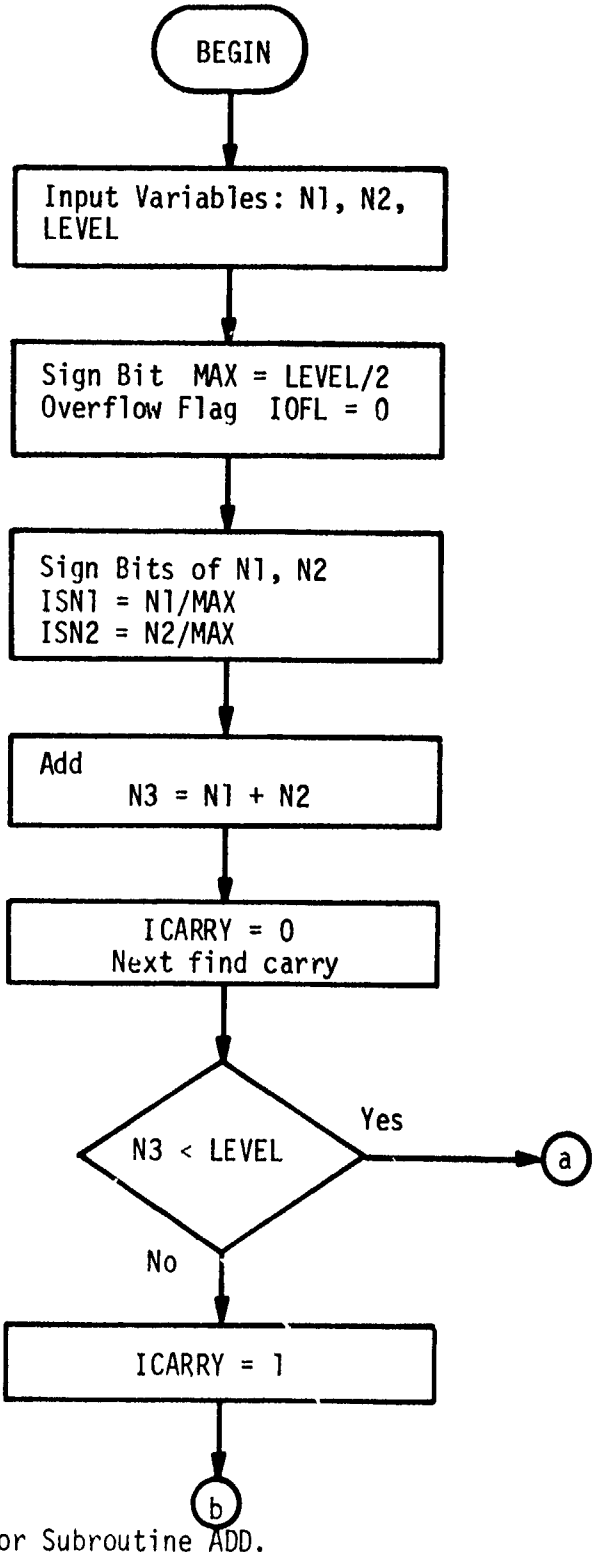


Fig. 6.4 Flow Chart for Subroutine ADD.

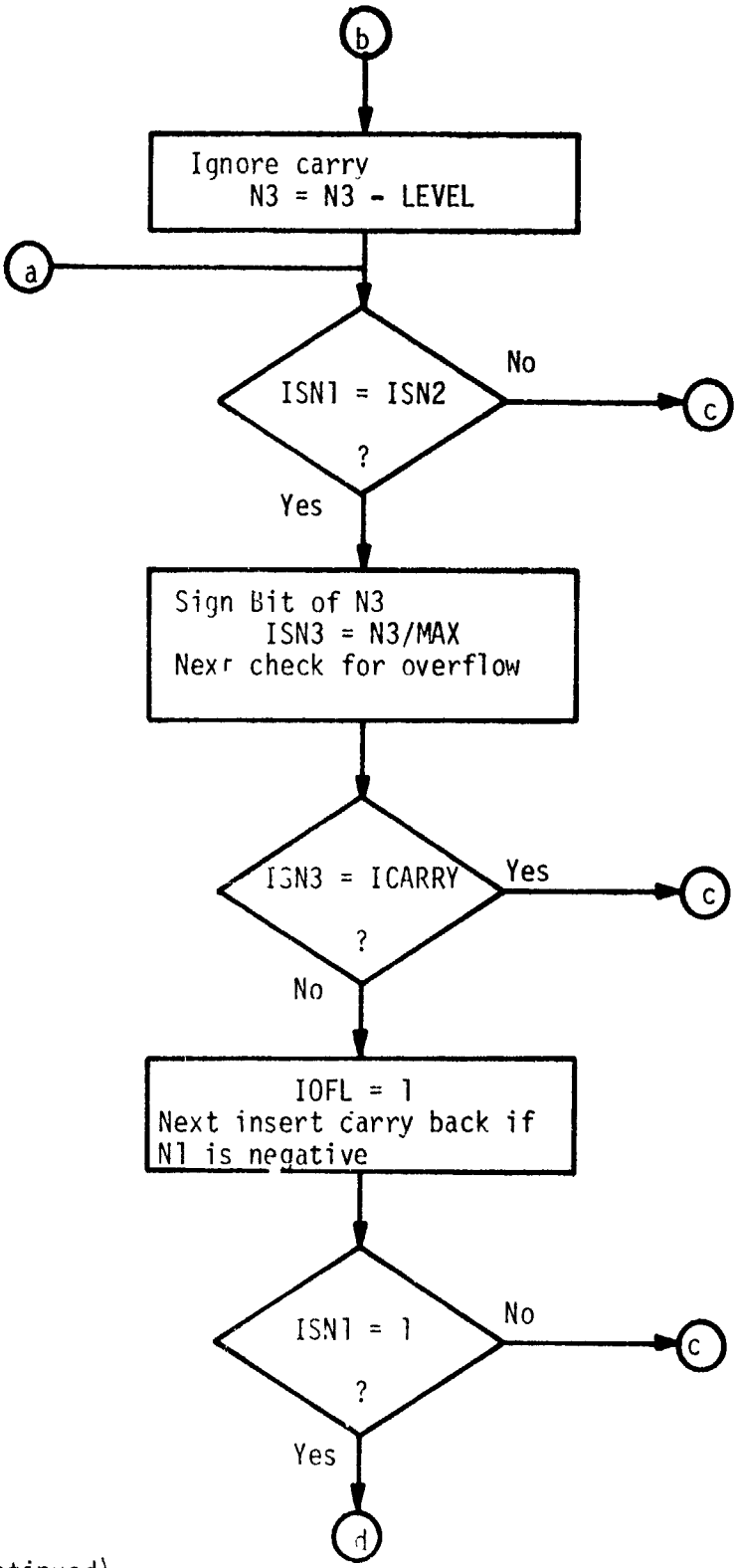


Fig. 6.4 (Continued)

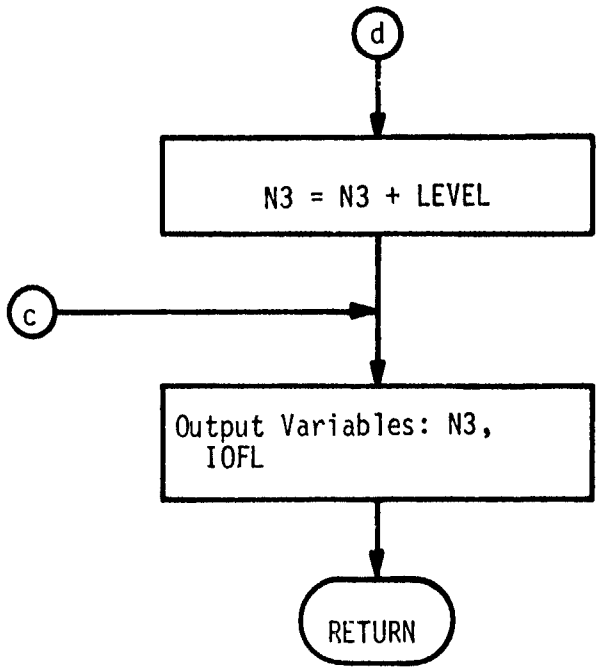


Fig. 6.4 (Continued)

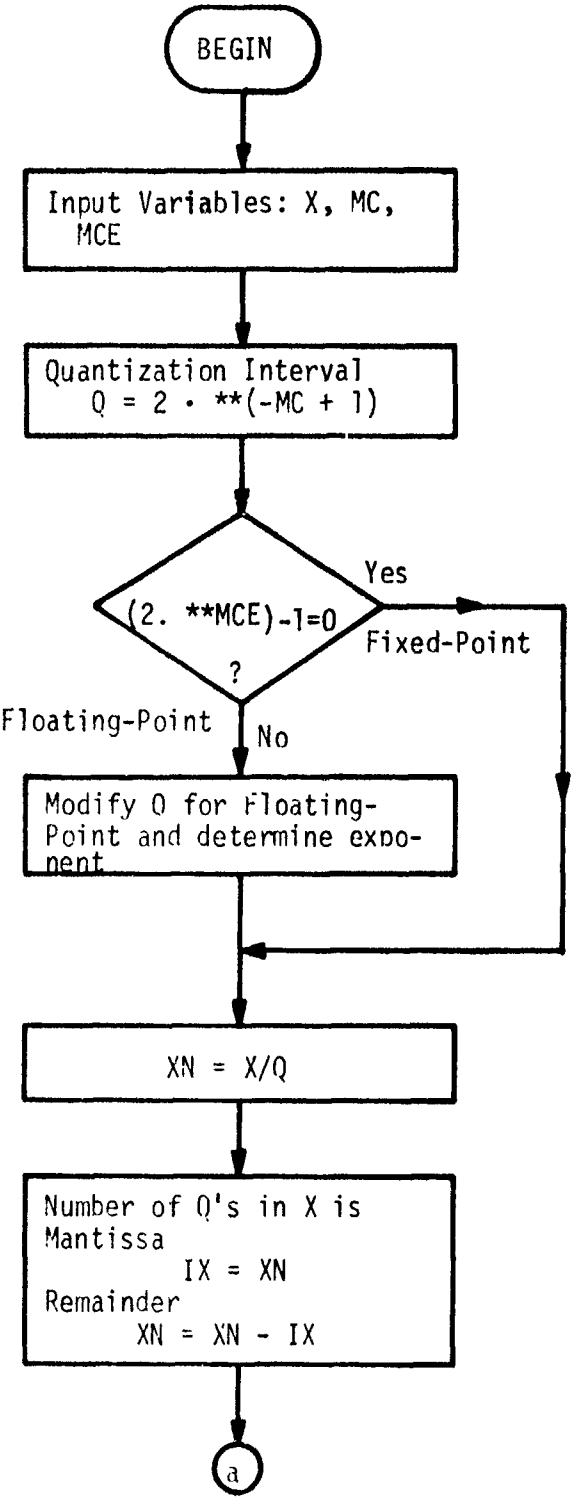


Fig. 6.5 Flow Chart for Subroutine FLCOEF.

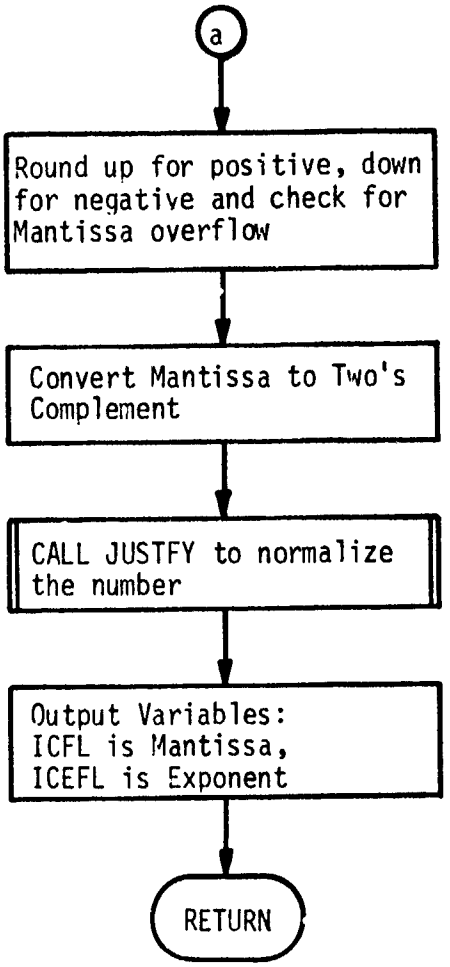


Fig. 6.5 (Continued)

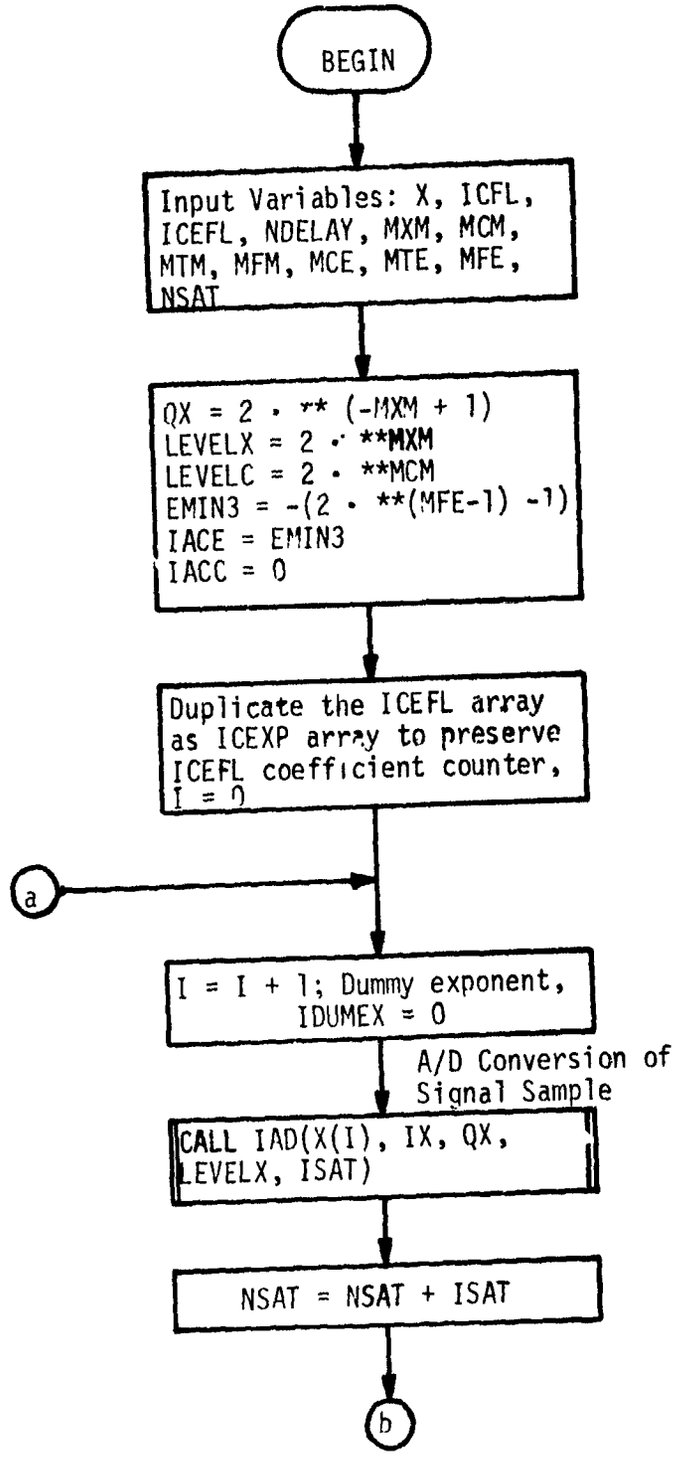


Fig. 6.6 Flow Chart for Subroutine FLOFLT.

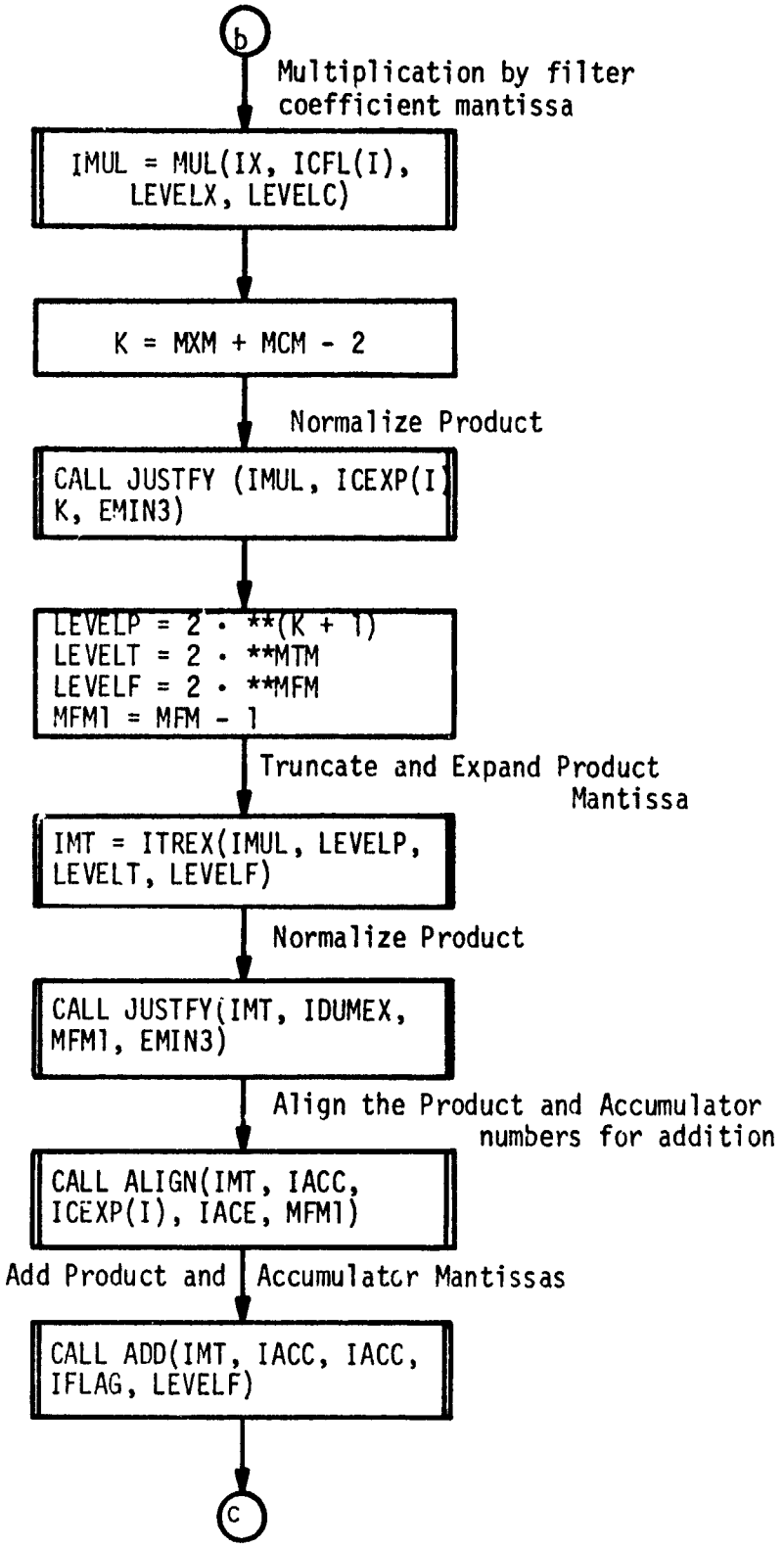


Fig. 6.6 (Continued)

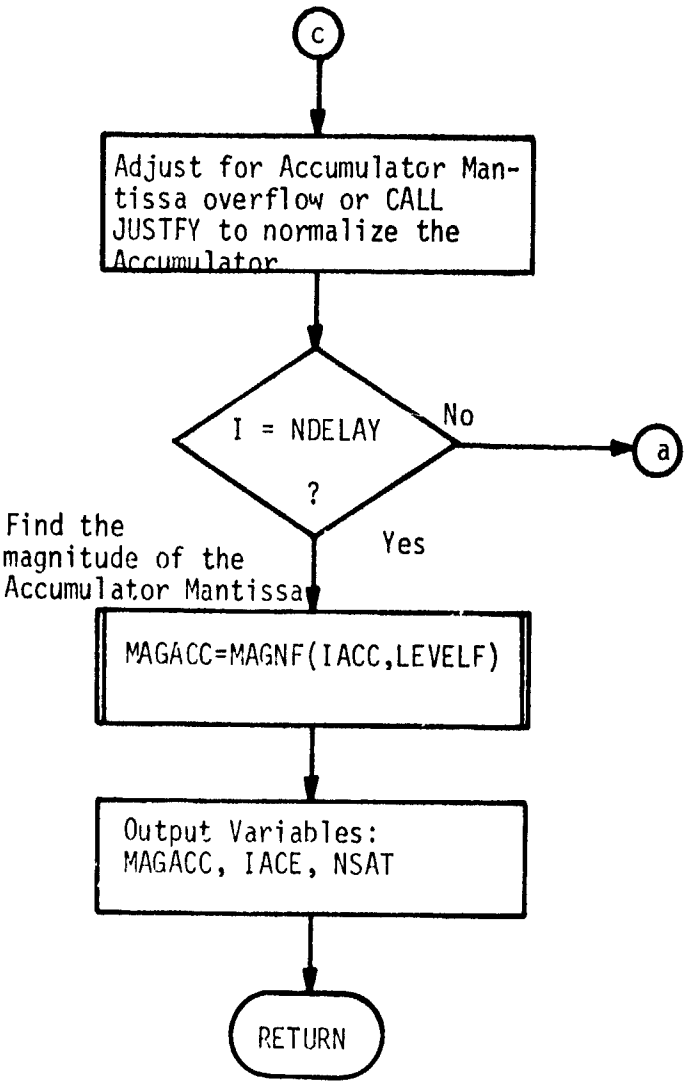


Fig. 6.6 (Continued)

the magnitude of the accumulator mantissa is determined by using MAGNF. Thus, the computations of FLOFLT use all the subprograms which simulate different arithmetic operations.

The subroutine RMSHAL implements the hardware algorithm for the RMS unit and its flow chart is shown in Fig. 6.7. The algorithm is outlined in Equation 2.2. As can be seen from the flow chart the parameter IDIV provides two options for the manipulation of the mantissas and exponents of S and L before they are finally added to implement Equation 2.2. If IDIV = 1, none of the exponents are disturbed and all the manipulations are carried out on the mantissas. If IDIV=0, then wherever possible the mantissa is divided by multiples of 2 and the exponents adjusted accordingly. The hardware implications of these are discussed in Section 4.3. At the time of constructing the simulation package it was deemed worthwhile to investigate the two options. But, from the hardware point of view it is easier to implement the IDIV = 1 option. All of the simulation results discussed in the next section were with IDIV = 0 but it did not have any effect on the quantization errors since the mantissa bit-length used for the RMS unit was sufficiently large.

In addition to the floating-point DSP word lengths defined in Table 5-1, three more word lengths were defined for the purpose of simulation. They appear below as Table 6-1.

TABLE 6-1 ADDITIONAL FLOATING POINT DSP WORD LENGTHS

MFT ---- Truncated filter output mantissa bit-length. Magnitude with MFT fractional bits. No sign bit is used.

MRM ---- RMS unit input mantissa bit-length. Magnitude with MRM fractional bits. No sign bit is used.

MRE --- RMS unit input exponent bit-length. Signed magnitude form with 1 sign bit and MRE - 1 integer bits.

Also note

1) For simulation all the exponents were represented in signed magnitude form instead of the two's complement form indicated in Table 5-1.
2) MX, MEM, MSM, MSE in Table 5-1 correspond to MXM, MRT, MIM, MIE respectively in the simulation program.
3) It was not necessary in the simulations to define new names for the exponent bit-lengths corresponding to MFT and MRT, i.e., MFE and MRE were used.

6.2 TYPICAL RESULTS

Simulation results were obtained for the set of 9 filter coefficients listed in Table 3-2. All the simulation results pertain to integrator outputs using the hardware RMS algorithm.

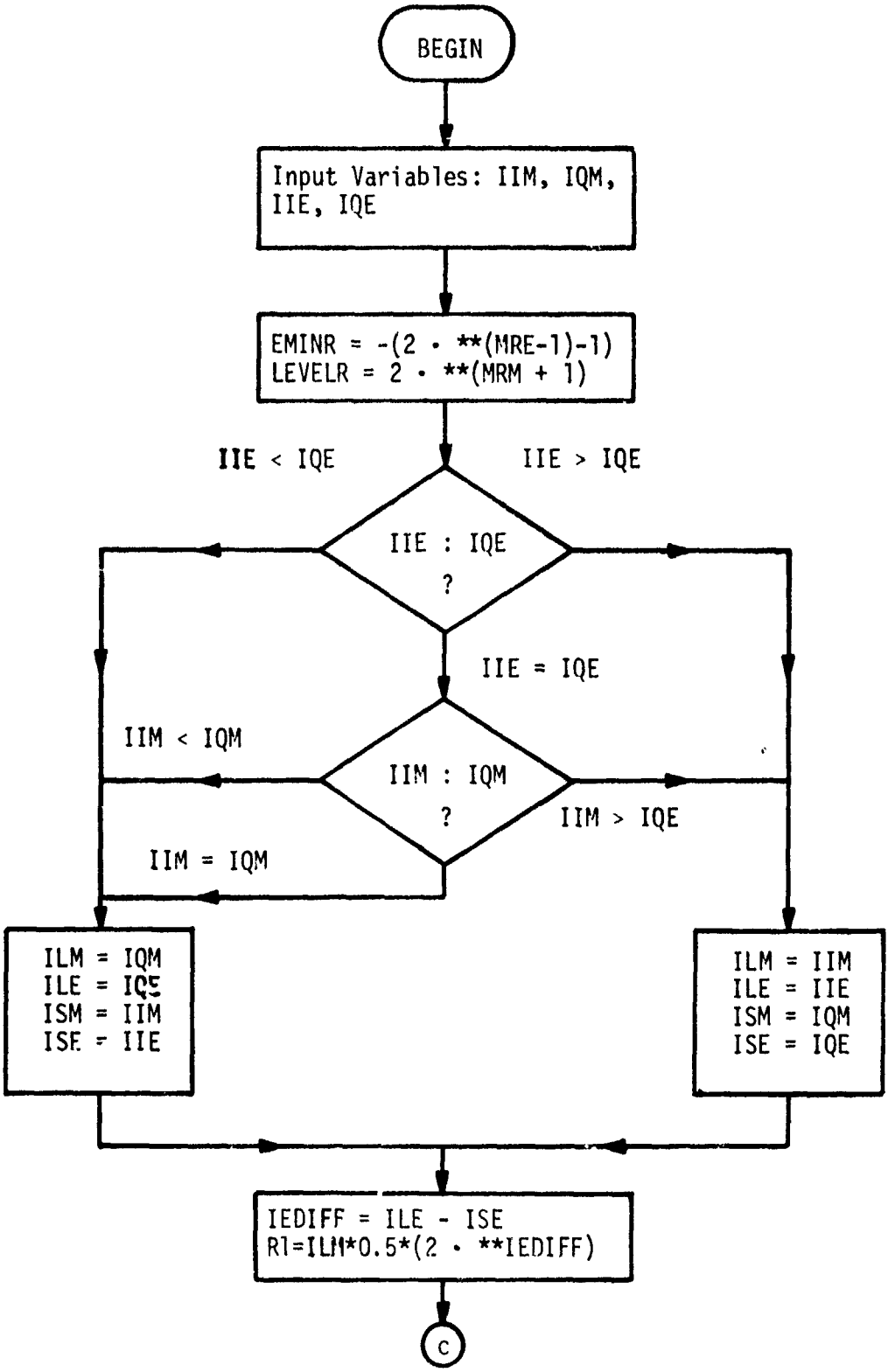


Fig. 6.7 Flow Chart for Subroutine RMSHAL

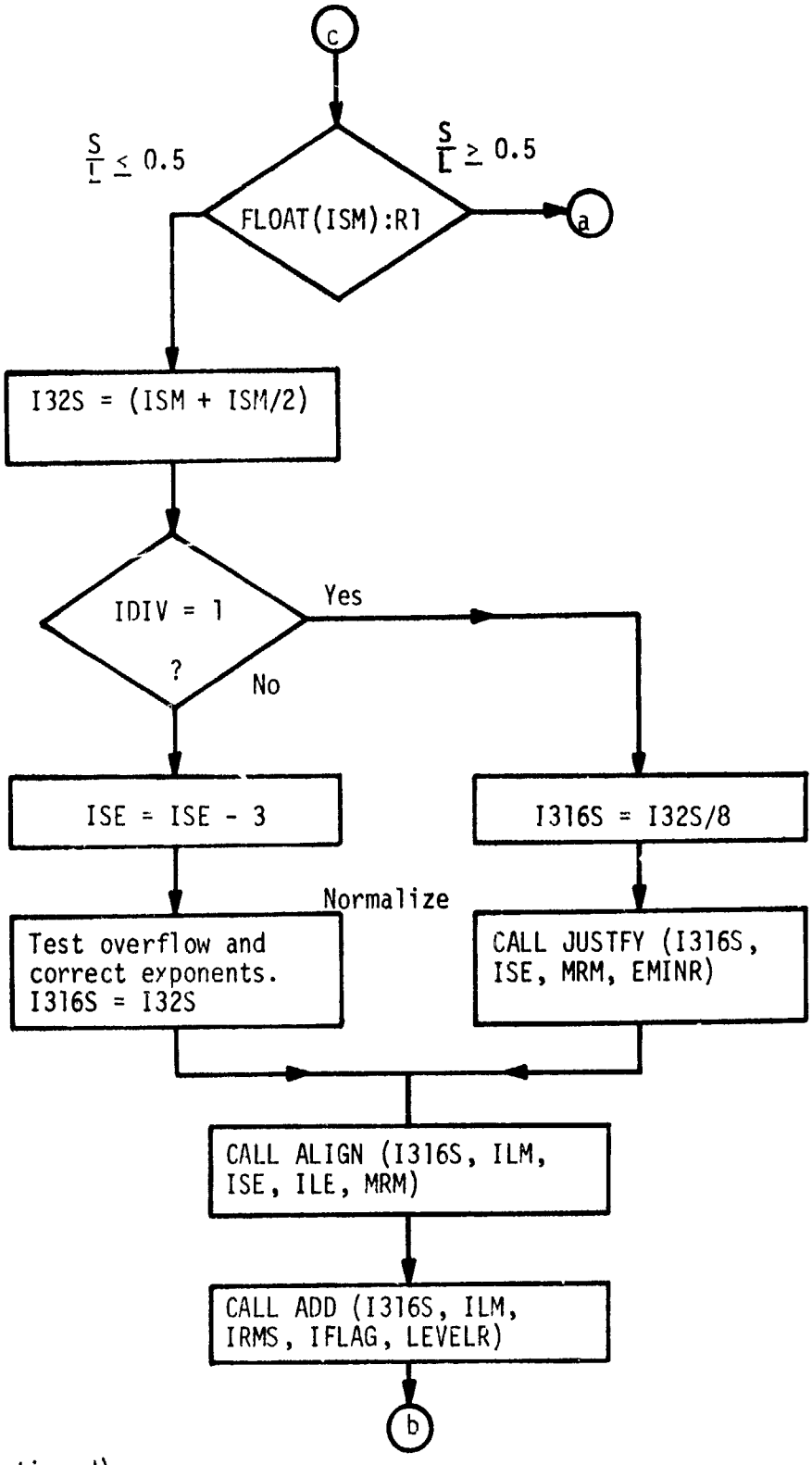


Fig. 6.7 (Continued)

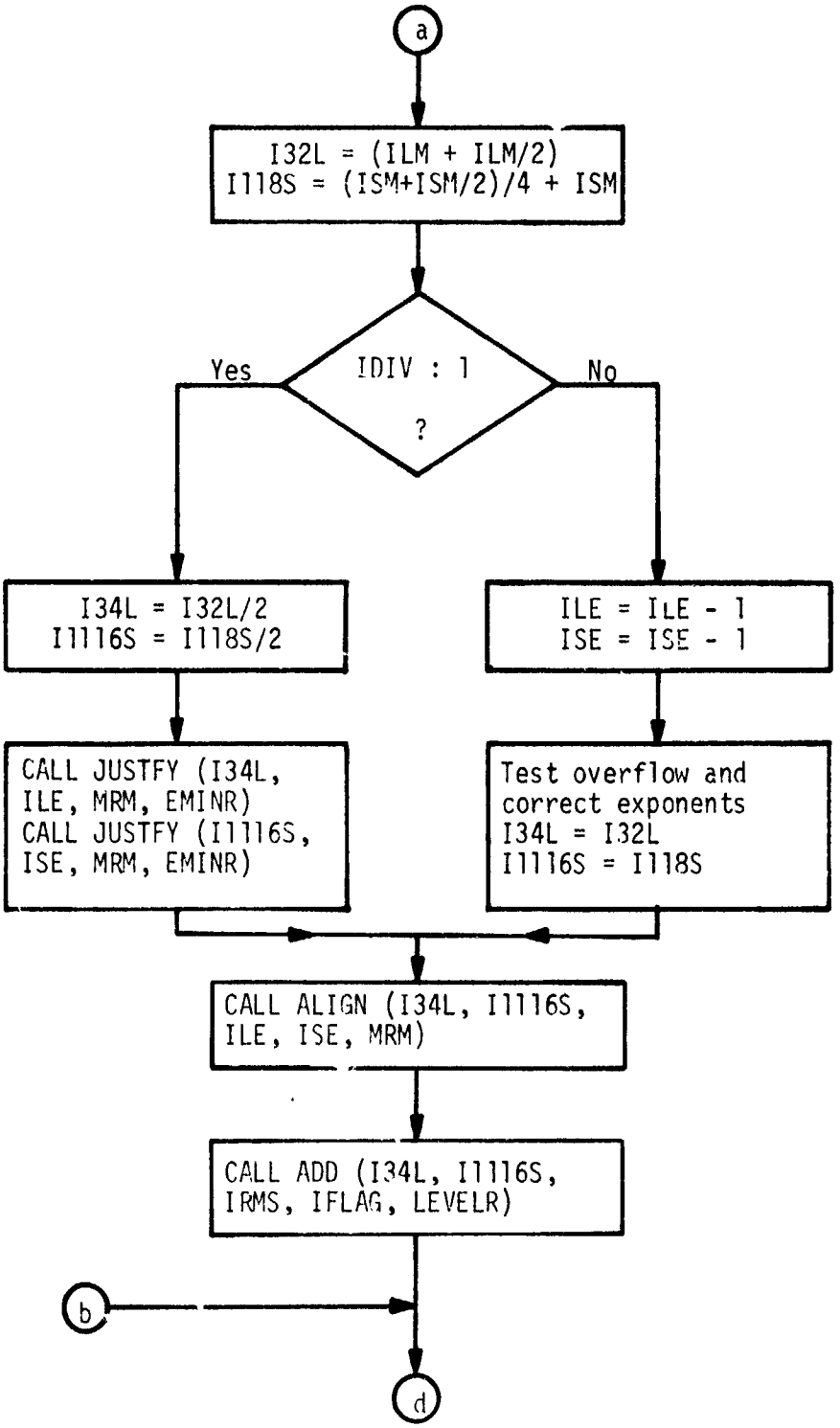


Fig. 6.7 (Continued)

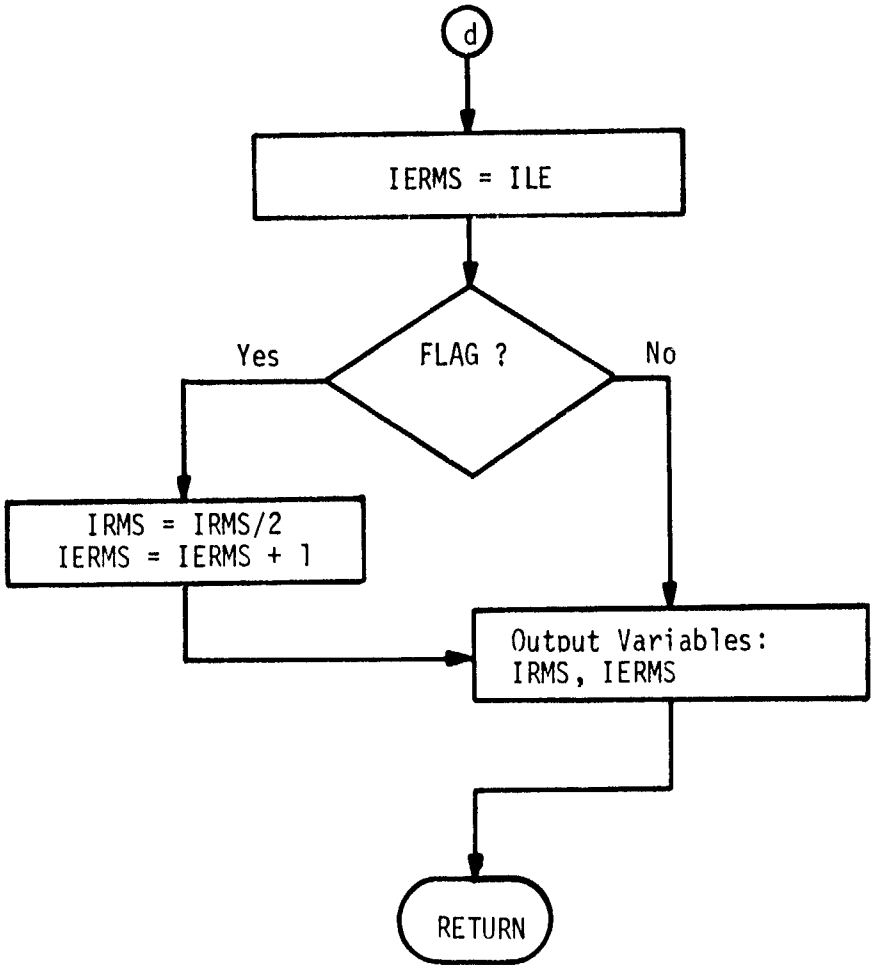


Fig. 6.7 (Continued)

The effect of the truncated product mantissa bit-length MTM and the integrator mantissa bit-length MIM on the average value and variance of the error at the integrator output is presented first. Then, the effect of varying signal amplitude on the same output parameters is shown. For all simulation results MXM = MCM = 9 and MCE = 0 were maintained, making these parameters the same as for the fixed-point processor simulation. The value of MFM was maintained the same as MTM, MFT = MFM - 1, MRM = 20 and MRT was kept the same as MIM. The values for the exponent bit-lengths were chosen as MTE = 4 and MFE = MRE = MIE = 6 to avoid possible exponent underflows and overflows.

The average error as a function of the truncated product bit-length MTM is shown in Fig. 6.8. Here the integrator bit-length MIM is chosen as the family parameter and the results are plotted for three different signal amplitudes. For the signal amplitude of 0.025V the dependence on MTM is very strong for $9 \leq MTM \leq 11$. However, for values of MTM > 11 there is much less dependence. For the signal amplitudes of 0.125V and 0.413V it is seen that the results are not strongly dependent on MTM. For all three cases it is seen that as the value of MIM decreases the magnitude of the average error increases. This effect is most prominent for the 0.025V case. Comparing this with the corresponding theoretical results shown in Fig. 5.2 it is seen that the lower bound for the 0.025 case is at least an order of magnitude lower than the simulation results. For 0.125V of signal amplitude the lower bound is smaller than the simulation results by approximately a factor of 2. However, for the 0.413V case the lower bound seems to be an order of magnitude larger than the simulation results. It is important to note that an exact equivalence does not exist between the two figures because MEM = 10 corresponds to MIM = 10 which lies in between MIM = 9 and MIM = 11 curves. All the upper bound values are positive and hence above the simulation results which are all negative.

The effect on the variance of the same parameters mentioned above is shown in Fig. 6.9. Neither MTM nor MIM strongly affects the variance. As expected the variance increases with signal amplitude just like the average error. This is the only significant generalization that can be made about the variance behavior. The theoretical upper bounds for the corresponding cases are seen in Fig. 5.3, and it is seen that for all signal amplitudes they are about an order of magnitude higher than the simulation results. The lower bounds are almost zero and hence are below the simulation results.

The average error as a function of MIM with MTM as a family parameter is shown in Fig. 6.10 for three different signal amplitudes. In this presentation the effect of MIM is more strongly brought out. But, the effect of MTM is not strongly discernible except in the case of the signal amplitude of 0.025V. The average error increases with increasing signal amplitude. An exact comparison can be made with the theoretical bounds for the cases of 0.125V and 0.413V by using Fig. 5.4 and Fig. 5.5 respectively, for MTM = 13. For 0.125V the average error lower bound is smaller than the simulation results by at least a factor of 4. The upper bound falls below the simulation curve for MIM = 7 but for larger values of MIM always stays positive in contrast to the negative simulation values. For 0.413V the lower bound is smaller than the simulation results

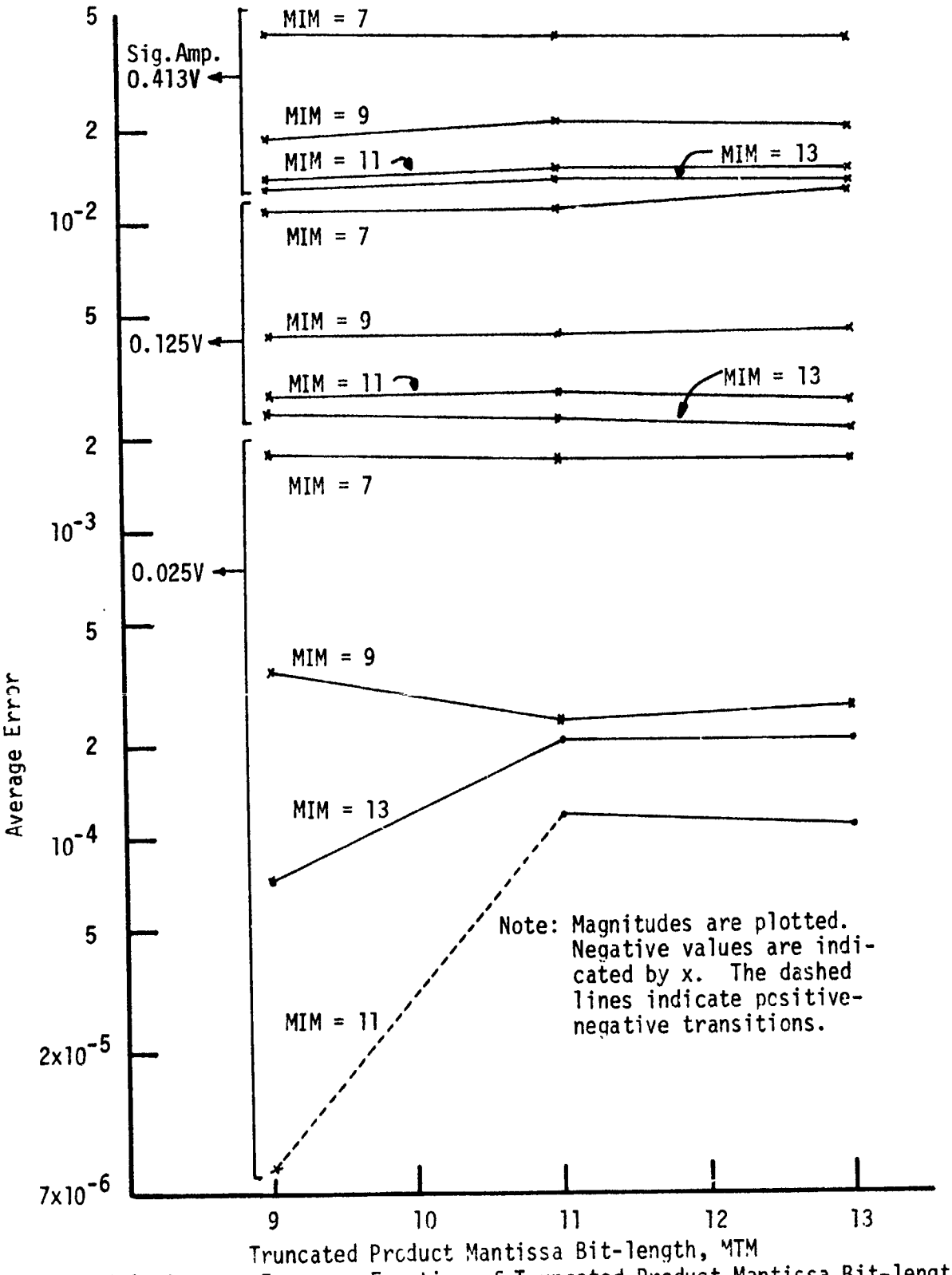


Fig. 6.8 Average Error as Function of Truncated Product Mantissa Bit-length with Integrator Mantissa Bit-length MIM as Family Parameter (9-Tap Floating-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz)

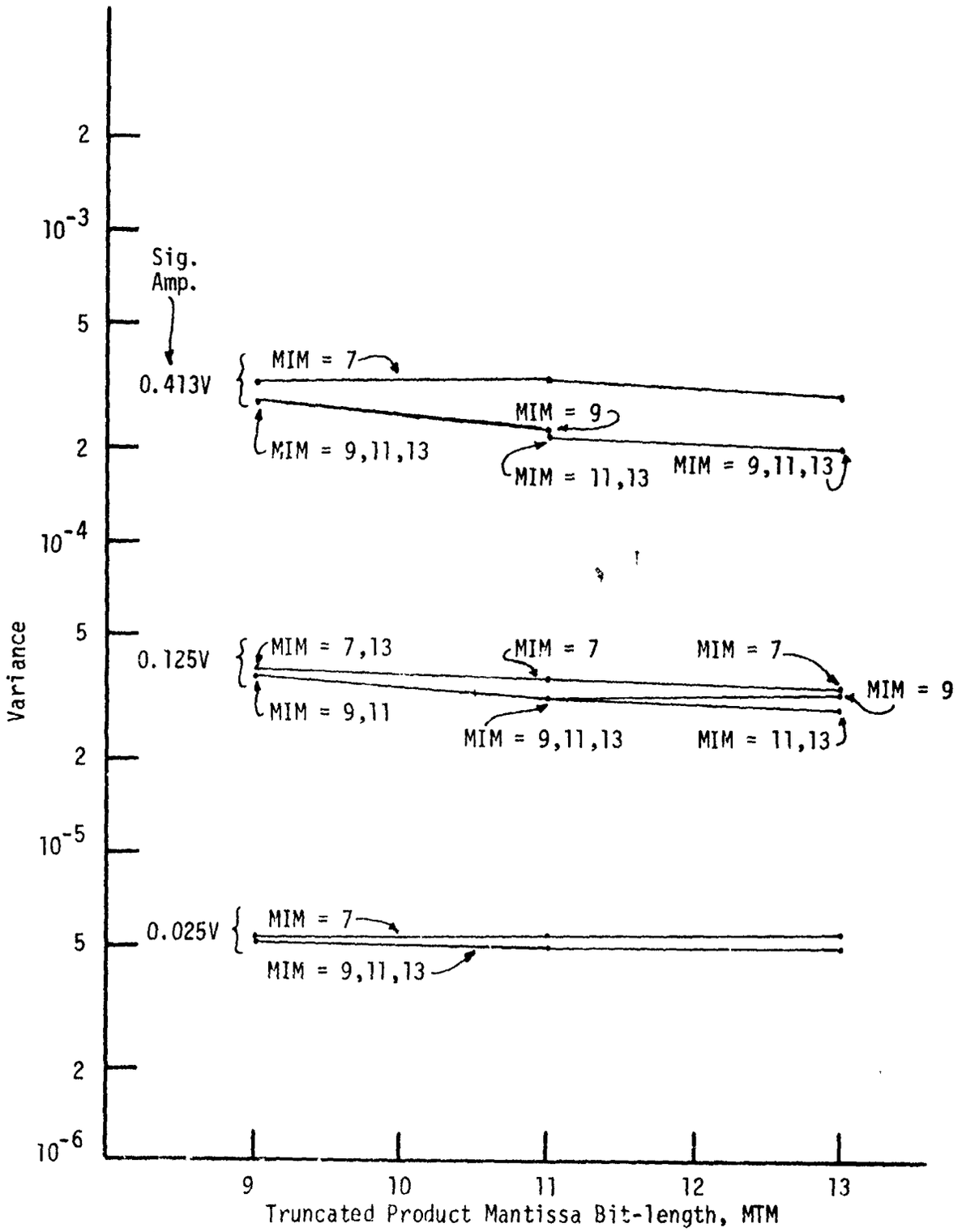


Fig. 6.9 Variance as Function of Truncated Product Mantissa Bit-length with Integrator Mantissa Bit-length as Family Parameter (9-Tap Floating-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz)

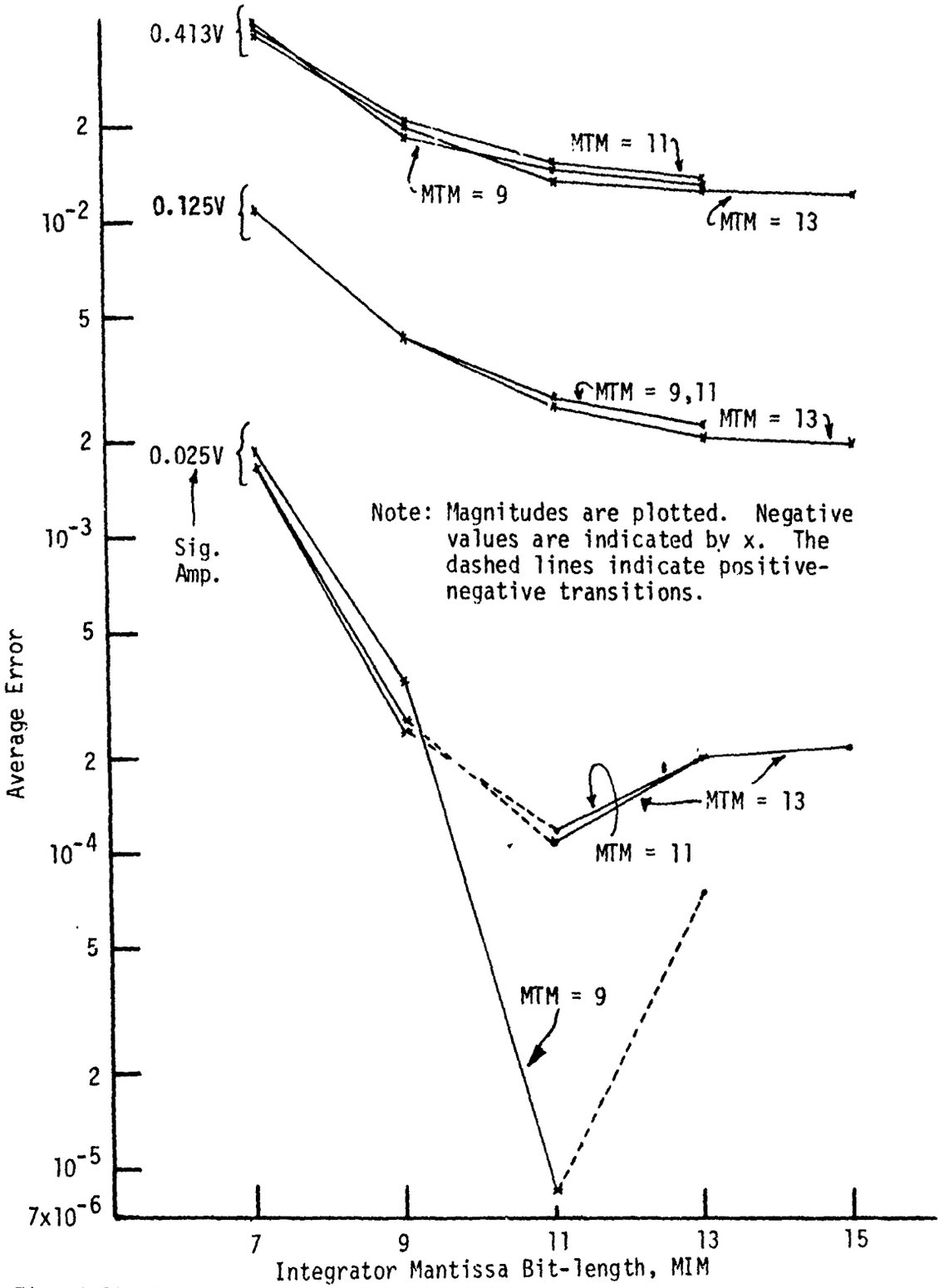


Fig. 6.10 Average Error as Function of Integrator Mantissa Bit-length with Truncated Product Mantissa Bit-length as Family Parameter (9-Tap Floating-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz)

at least by a factor of 3 and the upper bound is greater than the simulation value by a factor of 4 for MIM = 7. For larger values of MIM the upper bound is larger by a factor of 2.

Variance behavior for the same case mentioned above is shown in Fig. 6.11. Again no strong dependences on either MIM or MTM are observed. As the signal amplitude increases, the variance also increases. A comparison with the theoretical upper bounds is obtained by looking at Fig. 5.6. For each signal amplitude the upper bound is about two orders of magnitude larger. The variance lower bound is almost zero and hence below all the simulation results.

In Fig. 6.12 the effect of signal amplitude on average error is seen for MTM = MIM = 13. The magnitude of the average error shows several peaks and valleys with several positive-negative transitions, unlike the fixed-point processor simulation results. The excursions due to clutter are spread out at valleys but are cluttered together at peaks. The effect on variance is seen in Fig. 6.13. It is less erratic than the average error and shows an average positive slope for the curve. The excursions due to different clutter cases are closely grouped together. The theoretical upper bound curve as seen from Fig. 5.7 is an order of magnitude above the simulation results. The lower bound is almost zero and hence below all simulation results.

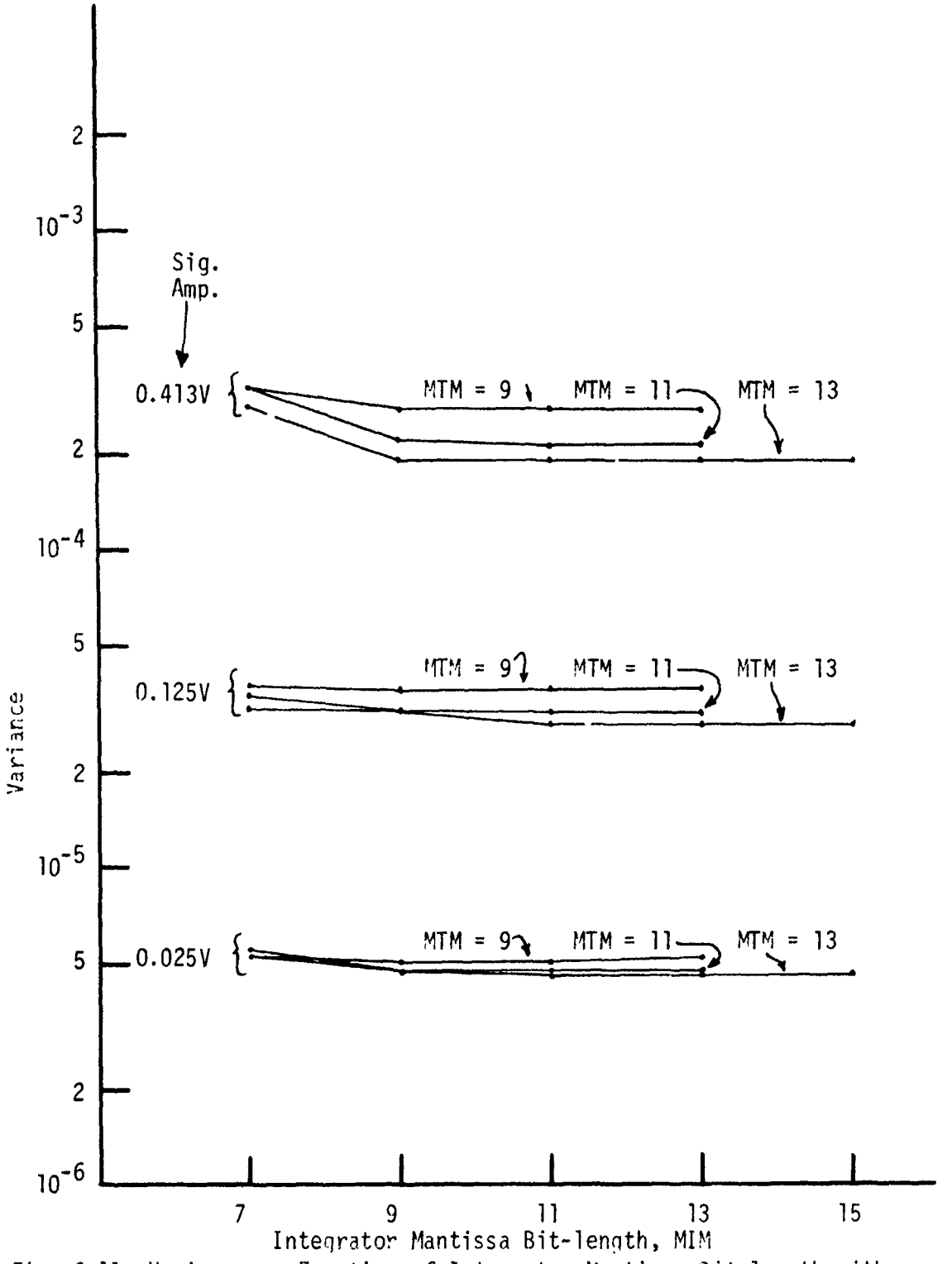


Fig. 6.11 Variance as Function of Integrator Mantissa Bit-length with Truncated Product Mantissa Bit-length as Family Parameter (9-Tap Floating-Point Simulation Results, No Clutter, Doppler Frequency = 1500 Hz)

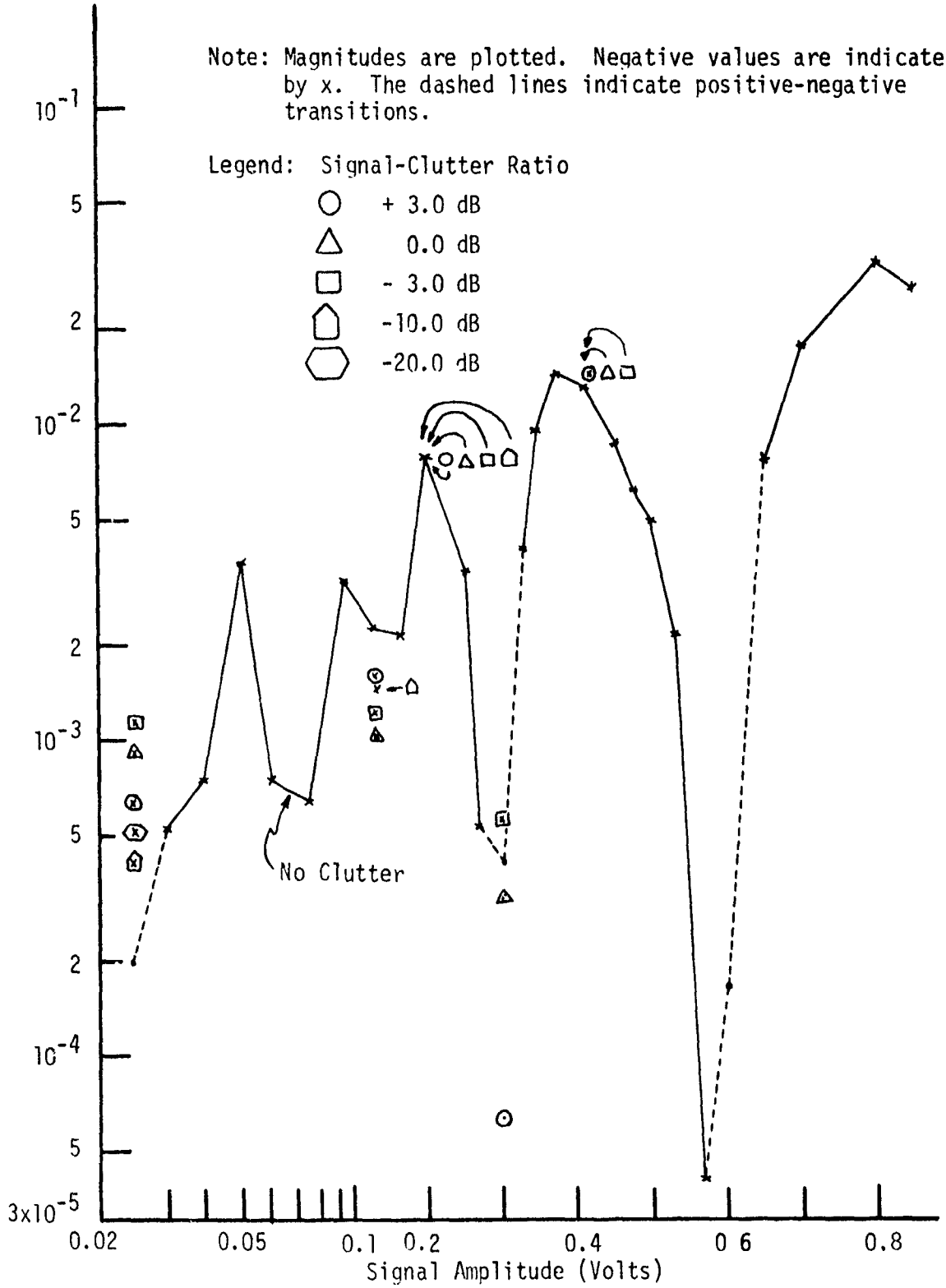


Fig. 6.12 Average Error as Function of Signal Amplitude (9-Tap Floating-Point Simulation Results, Doppler Frequency = 1500 Hz, MTM = MIM = 13)

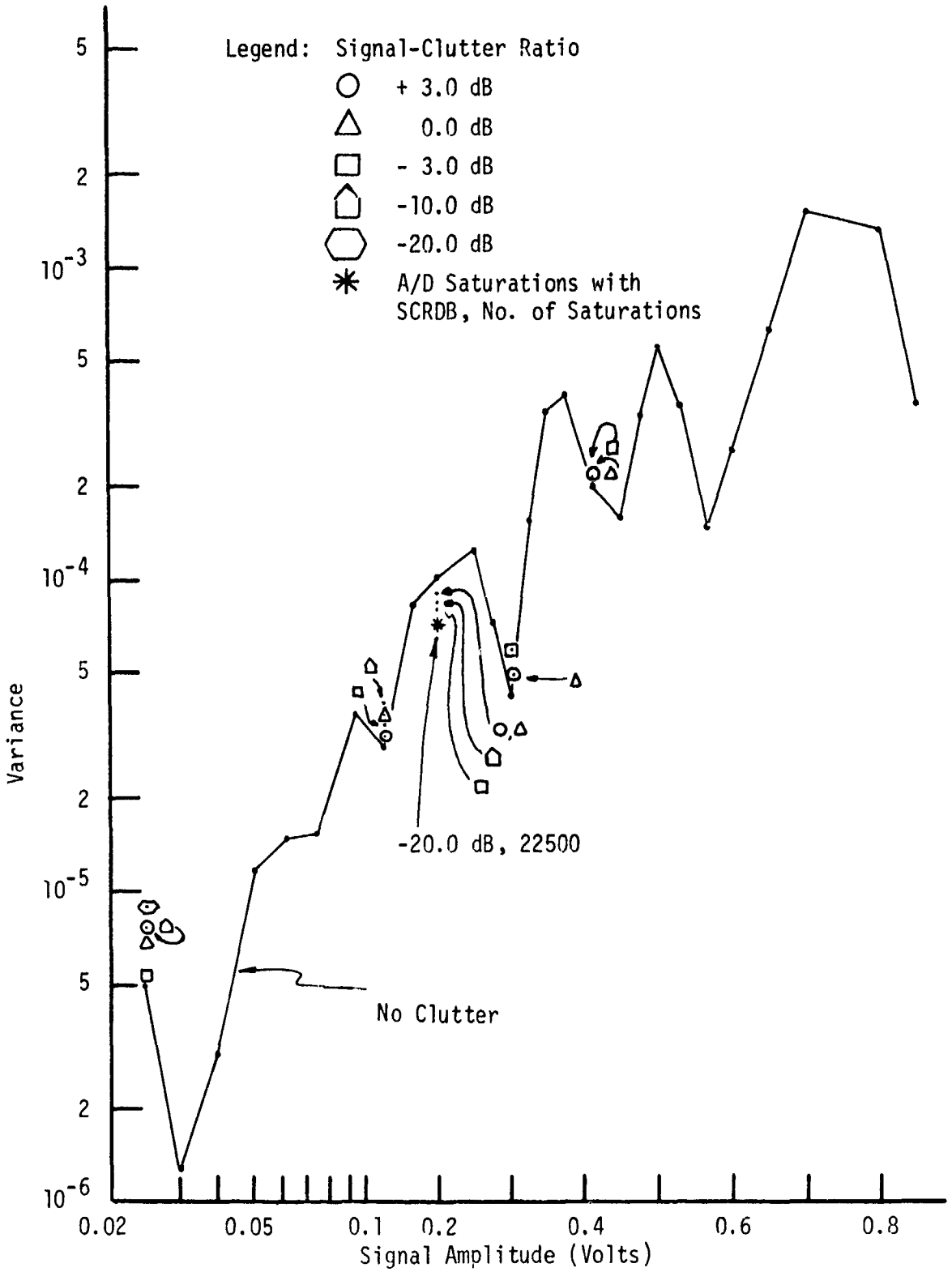


Fig. 6.13 Variance as Function of Signal Amplitude (9-Tap Floating-Point Simulation Results, Doppler Frequency = 1500 Hz, MTM = MIM = 13)

APPENDIX A

DETAILS OF FIXED-POINT ANALYSIS
by Jerry D. Moore

The analysis of the fixed-point DSP described in Chapter 2 is supplemented by this appendix. Equations presented in Chapter 2 are not repeated, but the detailed mathematical steps leading to the summary equations are given.

A.1 FILTER OUTPUT

Signal flow graphs are used to determine the output of the fixed-window MTI digital filter used in the DSP. Fig. A.1 depicts the MTI filter. The input is the sum of an errorless signal x() and the A/D converter quantization error e(). Traditional symbols are used, i.e., z^{-1} represents a one sample period delay, h() represents coefficient values, arrows represents multiplication, unlabeled arrows have unity factors associated with them, and small circles represent summation nodes. The $e_0(n)$, ... , $e_{N-1}(n-N+1)$ terms are the error associated with truncation of the product [x()+e()]· h(). A moving-window MTI, or an FIR digital filter, would produce an output value for each input value. The fixed-window realization is used because of simpler hardware requirements, i.e., the delay/multiplication process of Fig. A.1 can be realized by time multiplexing one multiplier and one storage location. It is the fixed-window approach that dictates that the sampler be present at the output. Sample values are taken from the output every N input samples.

The product truncation error terms can be collected and added to the final filter output as shown in Fig. A.2. Superposition is then applied to the input sequence and the structure shown in Fig. A.3 is obtained. The lower portion of this figure (where e(n) is the input and the product truncation terms are added) represents the total quantization error g() as given by Equation (2.5).

A.2 AVERAGE ERROR AND VARIANCE OF INTEGRATOR OUTPUT

The development in Chapter 2 following (2.5) through (2.12) is complete. The average value of the integrator error is obtained as follows,

$$\overline{INTE(M)} = \sum_{m=1}^{M} \overline{b(mN)} - Mu$$

$$= M[\bar{b} - u]$$

$$= M[\bar{r} + \bar{e}_t - u] \quad , \qquad (A.1)$$

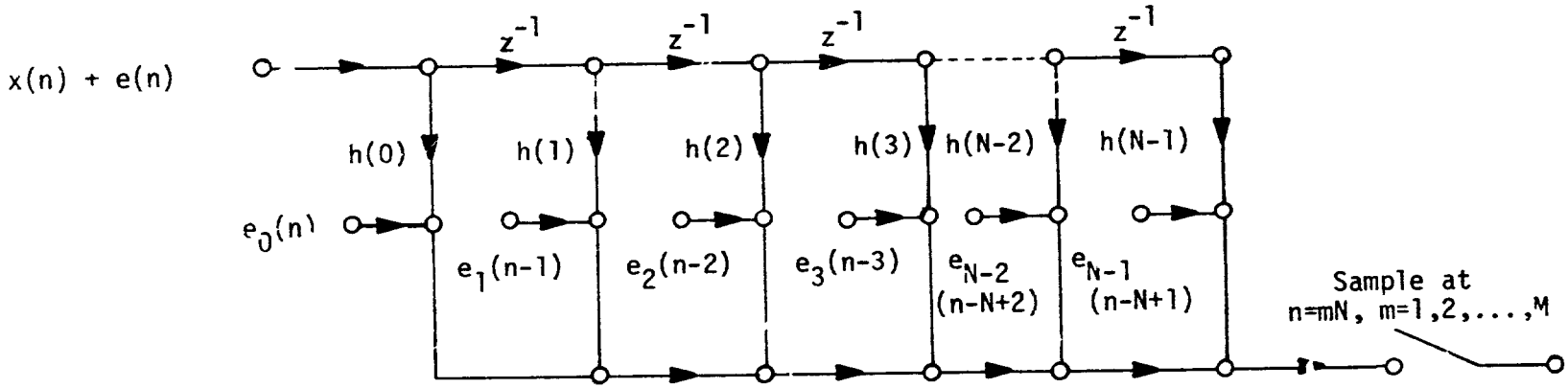


Fig. A.1 Original Signal Flow Graph for Fixed-Point MTI Filter

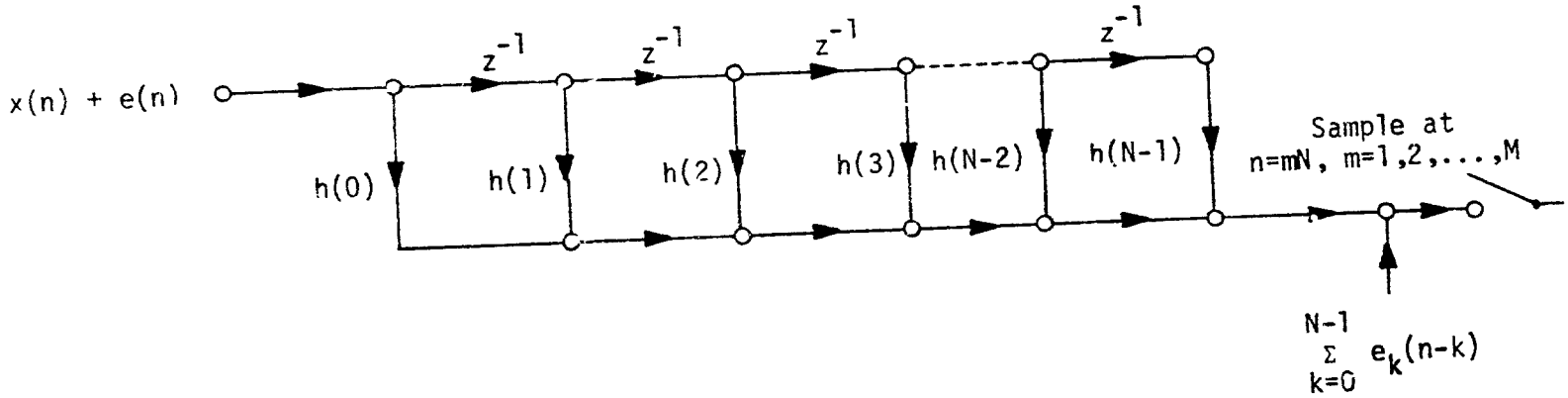


Fig. A.2 Signal Flow Graph for Fixed-Point MTI Filter with Product Truncation Error Separated

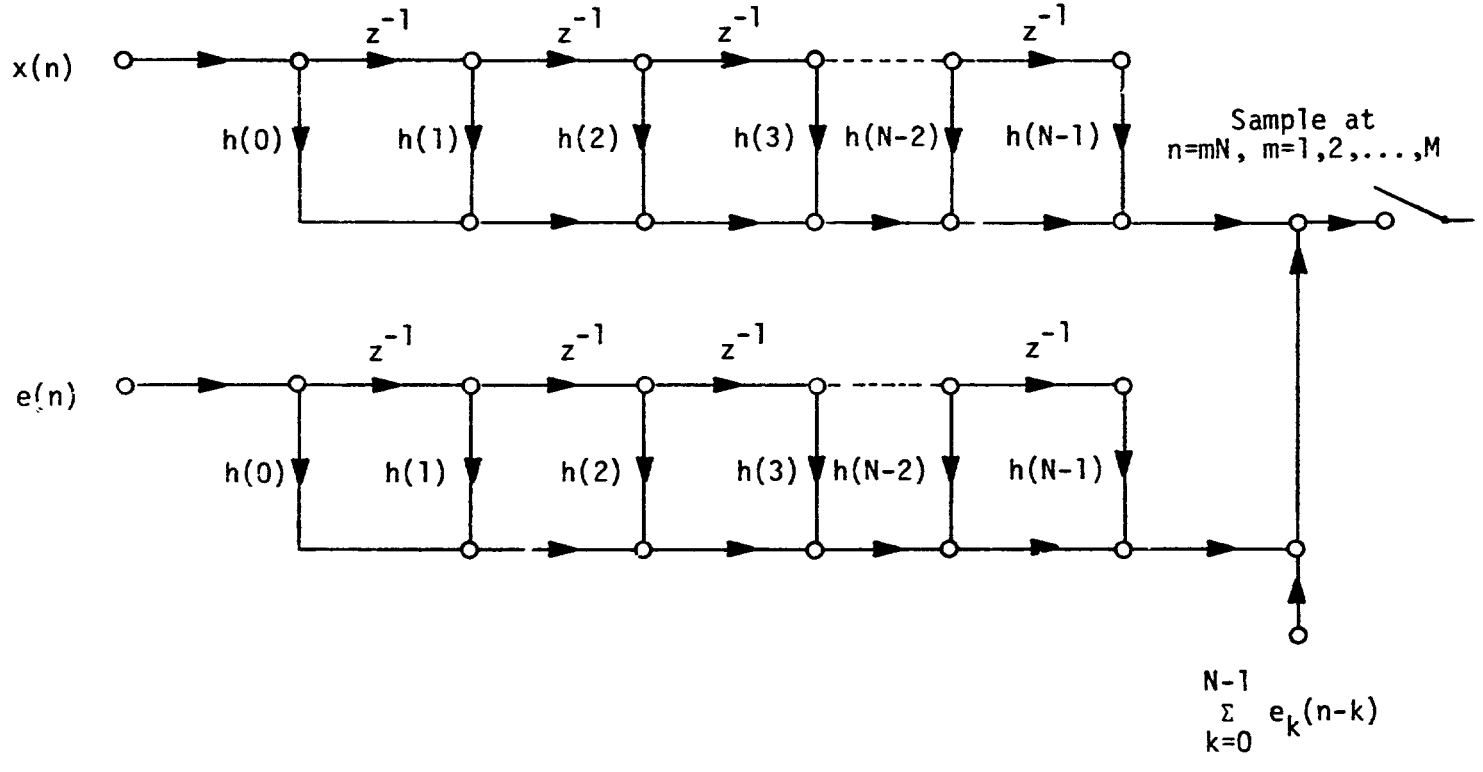


Fig. A.3 Signal Flow Graph for Fixed-Point MTI Filter with All Quantization Errors Separated

where it has been assumed that the average value of b(mN) does not depend on m. This is a logical choice for the signal environment encountered in radar systems, i.e., sinusoidal doppler signal with additive random noise and clutter. It is possible to show cases where the assumption is not valid, e.g., a constant level input, but this is not a practical radar signal.

An expression for the variance is obtained by

$$\sigma^2_{INTE} = \overline{INTE^2(M)} - \overline{INTE(M)} . \tag{A.2}$$

The mean squared value follows from

$$\overline{INTE^2(M)} = \overline{\left(\sum_{m=1}^{M} b(mN) - M u \right)^2}$$

$$= \sum_m \sum_k \overline{b(mN)b(kN)} - 2 M u \sum_m \overline{b}(mN)$$

$$+ M^2u^2 . \tag{A.3}$$

Using (A.1), squaring and subtracting from (A.3),

$$\overline{INTE(M)}^2 = \sum_m \sum_k \overline{b}(mN)\overline{b}(kN) - 2 M u \sum_m \overline{b}(mN)$$

$$+ M^2u^2 ,$$

$$\sigma^2_{INTE} = \sum_m \sum_k [\overline{b(mN)b(kN)} - \overline{b}(mN)\overline{b}(kN)] . \tag{A.4}$$

The bracketed term will equal zero for $m \neq k$ when statistical independence is assumed, thus

$$\sigma^2_{INTE} = \sum_m [\overline{b^2}(mN) - \overline{b}^2(mN)] . \tag{A.5}$$

Results given by (2.8) are used for b() and

$$\overline{b^2(mN)} = \overline{[r(mN) + e_t(mN)]^2}$$

$$= \overline{r^2} + 2\,\overline{r\,e_t} + \overline{e_t^2}$$

$$= \overline{r^2} + \overline{e_t^2} + 2\,\overline{r}\,\overline{e}_t \,, \qquad \text{(A.6)}$$

where r and e_t are statistically independent. Similarly,

$$\overline{b(mN)}^2 = \overline{r}^{\,2} + 2\,\overline{r}\,\overline{e}_t + \overline{e}_t^{\,2} \,. \qquad \text{(A.7)}$$

Subtracting (A.7) from (A.6) and substituting into (A.5) gives

$$\sigma^2_{INTE} = \sum_m [(\overline{r^2} - \overline{r}^{\,2}) + (\overline{e_t^2} - \overline{e}_t^{\,2})] \,, \qquad \text{(A.8)}$$

from which (2.14) is obtained, i.e.,

$$\sigma^2_{INTE} = M[\sigma_r^2 + \sigma_t^2] \,. \qquad \text{(A.9)}$$

A.3 BOUNDING OF RMS UNIT OUTPUT

Equation (2.7) gives the expression for the residue, i.e., the RMS Unit output. Because of the nonlinear relationship, it is necessary to bound this expression prior to determining the average value and variance. Considering the square-root portion and replacing w() by (2.3) gives,

$$\sqrt{w_I^2 + w_Q^2} = \sqrt{(y_I + g_I)^2 + (y_Q + g_Q)^2} \,. \qquad \text{(A.10)}$$

This can be viewed as the magnitude of a resultant vector which is formed by summing two vectors U and V, where

$$U = (y_I, y_Q) \,,$$

$$|U| = u = \sqrt{y_I^2 + y_Q^2} \,,$$

$$V = (g_I, g_Q) \,,$$

$$|V| = v = \sqrt{g_I^2 + g_Q^2} \,. \qquad \text{(A.11)}$$

The magnitude of the sum of two vectors is less than the sum of the magnitude of these vectors. Also, the magnitude of the sum of two vectors is greater than the magnitude of the difference of the mag-

nitudes of these vectors. Stated in equation form this gives (2.15).

A.4 CONCAVE/CONVEX THEOREM

A general form of this theorem is given [cf., pp. 563-567 of [13]) and then applied to the analysis problem of Chapter 2.

Theorem: The expected value of a concave or convex function, f(), of a random variable, x, can be bounded as follows

$$\overline{f(x)} \leq f(\bar{x}) \text{ when } f''(x) \leq 0 : \text{CONVEX} . \tag{A.12}$$

$$\overline{f(x)} \geq f(\bar{x}) \text{ when } f''(x) \geq 0 : \text{CONCAVE} . \tag{A.13}$$

Proof: Expand $f(x)$ in a Taylor's series about the point $\bar{x}$,

$$f(x) = \sum_{k=0}^{\infty} \frac{(x - \bar{x})^k}{k!} f^k(\bar{x}) . \tag{A.14}$$

Express this as a sum of the first two terms and a remainder term,

$$f(x) = f(\bar{x}) + (x - \bar{x}) f'(\bar{x}) + R ,$$

$$R = \frac{(x - \bar{x})^2}{2} f''(x_1) ,$$

$$\bar{x} < x_1 < x \quad \text{for} \quad \bar{x} < x ,$$

$$x < x_1 < \bar{x} \quad \text{for} \quad \bar{x} > x . \tag{A.15}$$

If f"() is less than zero, i.e., the function is convex, then the remainder term is negative and

$$f(x) \leq f(\bar{x}) + (x - \bar{x}) f'(\bar{x}) ,$$

$$\overline{f(x)} \leq \overline{f(\bar{x})} + \overline{(x - \bar{x}) f'(\bar{x})} ,$$

$$\overline{f}(x) \leq f(\bar{x}) : \text{CONVEX} . \tag{A.16}$$

For f"() positive, the function is concave and

$$\overline{f(x)} \geq f(\bar{x}) : \text{CONCAVE, Q.E.D.} \tag{A.17}$$

The first application of the theorem is in (2.17), where $|u - v|$ is a function of the random variable v and is obviously concave. Thus, from (A.17)

$$\overline{|u - v|} \geq |u - \bar{v}| . \tag{A.18}$$

The next application of the theorem is in (2.18). The lower bound is obtained by treating v as a function of g_I and then g_Q.

$$v'' = \frac{d^2 v}{d g_I^2} = \frac{g_Q^2}{v^3} \geq 0 . \tag{A.19}$$

The use of g_I and then g_Q is justified because of the statistical independence and

$$\begin{aligned} \bar{v} &= \int\int \sqrt{g_I^2 + g_Q^2}\, p(g_I, g_Q)\, dg_I dg_Q \\ &= \int \left[\int \sqrt{g_I^2 + g_Q^2}\, p(g_I) dg_I \right] p(g_Q) dg_Q . \end{aligned} \tag{A.20}$$

This gives from the concave theorem,

$$\bar{v} \geq \sqrt{\bar{g}_I^2 + \bar{g}_Q^2} = \sqrt{2\bar{g}^2} = \sqrt{2}\, |\bar{g}| . \tag{A.21}$$

However, if v is treated as a function of g_I^2 and then g_Q^2 then

$$v'' = \frac{d^2 v}{d(g_I^2)^2} = -\frac{1}{4} v^{-3/2} \leq 0 . \tag{A.22}$$

This is convex and thus

$$v \leq \sqrt{\overline{g_I^2} + \overline{g_Q^2}} = \sqrt{2\overline{g^2}} . \tag{A.23}$$

A.5 AVERAGE ERROR BOUND

The upper bound expression of (2.19) follows immediately by substituting (2.18) into the upper bound portion of (2.17) and the results used in (2.13). The lower bound expression of (2.20) involves another simple manipulation, viz.,

$$|u - \bar{v}| = \sqrt{\overline{(u - \bar{v})^2}} = \sqrt{u^2 - 2u\,\bar{v} + \overline{\bar{v}^2}} . \tag{A.24}$$

The smallest value of this expression is obtained by

$$|u - \bar{v}|_{min} = \sqrt{u^2 - 2u\,\bar{v}_{max} + \overline{\bar{v}^2}_{min}} . \tag{A.25}$$

Using (2.18) yields

$$|u - \bar{v}|_{min} = \sqrt{u^2 - 2u\sqrt{2\bar{g}^2} + 2\bar{g}^2} \quad . \tag{A.26}$$

Equation (2.20) follows from (A.26).

A.6 ERROR VARIANCE BOUNDS

The upper bound on σ_r^2 is a consequence of using the maximum mean-square value and the minimum mean value squared, i.e.,

$$\sigma_r^2 \leq \overline{r^2}_{max} - \bar{r}^2_{min} \ ,$$

$$\overline{r^2} \leq \overline{(1 + \gamma_w)^2}\ (u^2 + 2u\,\bar{v} + \overline{v^2})$$

$$\leq \overline{(1 + \gamma_w)^2}\ (u^2 + 2u\sqrt{2\bar{g}^2} + 2\bar{g}^2) \ , \tag{A 27}$$

$$\bar{r}_{min}^2 = (1 + \bar{\gamma}_w)^2\ (u - \bar{v})^2 \ ,$$

$$\sigma_r^2 \leq (1 + 2\bar{\gamma}_w + \overline{\gamma_w^2})(u^2 + 2u\,\bar{v} + \overline{v^2})$$

$$- (1 + 2\bar{\gamma}_w + \bar{\gamma}_w^2)(u^2 - 2u\,\bar{v} + \bar{v}^2)$$

$$= (\overline{\gamma_w^2} - \bar{\gamma}_w^2)\ u^2 + (1 + 2\bar{\gamma}_w)(\overline{v^2} - \bar{v}^2)$$

$$+ \overline{\gamma_w^2}\ \overline{v^2} - \bar{\gamma}_w^2\ \bar{v}^2 + (2 + 4\bar{\gamma}_w + \overline{\gamma_w^2} + \bar{\gamma}_w^2)\ 2u\,\bar{v} \ . \tag{A.28}$$

Collecting terms yields,

$$\sigma_r^2 \leq \sigma_\gamma^2\ u^2 + (1 + 2\bar{\gamma}_w)\ \sigma_v^2 + 2\,u\,\bar{v}\,(2 + 4\bar{\gamma}_w + \overline{\gamma_w^2} + \bar{\gamma}_w^2)$$

$$+ \overline{\gamma_w^2}\ \overline{v^2} - \bar{\gamma}_w^2\ \bar{v}^2 \ . \tag{A.29}$$

The lower bound results from using the minimum mean-square value and the maximum mean value squared, i.e.,

$$\sigma_r^2 \geq \begin{cases} \overline{r^2}_{min} - \overline{r}^2_{max} \\ 0, \text{ if above result is } < 0. \end{cases}$$

$$\overline{r^2} \geq \overline{r^2}_{min} \geq \overline{(1+\gamma_w)^2}\,(u^2 - 2u\,\overline{v} + \overline{v^2})$$

$$= \overline{(1+\gamma_w)^2}\,(u^2 - 2u\sqrt{2\overline{g^2}} + 2\overline{g^2}) \,,$$

$$\overline{r^2} \leq \overline{r^2}_{max} = (1+\overline{\gamma}_w)^2\,(u + 2u\,\overline{v} + \overline{v^2}) \,,$$

$$\sigma_r^2 \geq (1 + 2\overline{\gamma}_w + \overline{\gamma_w^2})(u^2 - 2u\,\overline{v} + \overline{v^2})$$

$$- (1 + 2\overline{\gamma}_w + \overline{\gamma}_w^2)(u^2 + 2u\,\overline{v} + \overline{v^2})$$

$$= (\overline{\gamma_w^2} - \overline{\gamma}_w^2)\,u^2 + (1 + 2\overline{\gamma}_w)(\overline{v^2} - \overline{v^2}) + \overline{\gamma_w^2}\,\overline{v^2} - \overline{\gamma}_w^2\,\overline{v^2})$$

$$- (2 + 4\overline{\gamma}_w + \overline{\gamma_w^2} + \overline{\gamma}_w^2) \cdot 2u\,\overline{v} \,. \qquad \text{(A.30)}$$

Collecting terms yields

$$\sigma_r^2 \geq \sigma_\gamma^2\,u^2 + (1 + 2\overline{\gamma}_w)\,\sigma_v^2 - 2u\,\overline{v}\,(2 + 4\overline{\gamma}_w + \overline{\gamma_w^2} + \overline{\gamma}_w^2)$$

$$+ \overline{\gamma_w^2}\,\overline{v^2} - \overline{\gamma}_w^2\,\overline{v}^2 \,. \qquad \text{(A.31)}$$

The results of (A.29) and (A.30) depend on σ_v^2 which cannot be determined exactly. It can be bounded as

$$\sigma_v^2 = \overline{v^2} - \overline{v}^2$$

$$= 2\overline{g^2} - \overline{v}^2 \,,$$

$$\sigma_v^2 \leq 2\overline{g^2} - 2\overline{g}^2 = 2\,\sigma_g^2, \text{ from (A.21)},$$

$$\sigma_v^2 \geq 2\overline{g^2} - 2\overline{g^2} = 0, \text{ from (A.23)}. \qquad \text{(A.32)}$$

Thus

$$\sigma_r^2 \leq \sigma_\gamma^2\,u^2 + 2(1 + 2\overline{\gamma}_w)\,\sigma_g^2 + 2(\overline{\gamma_w^2} + \overline{\gamma}_w^2 + 4\overline{\gamma}_w + 2)\sqrt{2\overline{g^2}} \cdot u$$

$$+ 2\,\overline{\gamma_w^2}\,\overline{g^2} - 2\,\overline{\gamma}_w^2\,\overline{g}^2 \,, \qquad \text{(A.33)}$$

and

$$\sigma_{\gamma}^2 \geq \sigma_{\gamma}^2 u^2 - 2 (\overline{\gamma_w^2} + \overline{\gamma}_w^2 + 4\overline{\gamma}_w + 2) \sqrt{2\overline{g^2}}\, u + 2\overline{g^2}\, \sigma_{\gamma}^2 . \quad (A.34)$$

The results of (A.33) and (A.34) are (2.22) and (2.23) in Chapter 2. The output error variance is bounded as per (2.24) and (2.25) in Chapter 2.

A.7 TRUNCATION STATISTICS

The presentation by Oppenheim and Schafer [3] on pages 409-413 forms the basis for the truncation statistics used in (2.26)-(2.28) and (2.30)-(2.32). A two's complement arithmetic system is used, i.e., positive numbers have a 0 sign bit and magnitude bits, while negative numbers are represented in two's complement form. Letting x represent the fractional number with n_b bits, (the b subscript is a mnemonic for before and will become obvious later in the presentation), then

$$|x| = \sum_{n=1}^{n_b} a_n 2^{-n} . \quad (A.35)$$

The representation of a number is

$$x = 0 \cdot a_1 a_2 \ldots a_{n_b}, \; x \geq 0 ,$$

$$x = 1 \cdot b_1 b_2 \ldots b_{n_b}, \; x < 0 , \quad (A.36)$$

where the b_n terms are determined for negative numbers by

$$x = 1 - |x| . \quad (A.37)$$

It can be shown [3] that when such numbers are truncated to contain n_a bits, (where the a subscript is a mnemonic for after truncation), then the error e_{tr} is

$$-(2^{-n_b} - 2^{-n_a}) \leq e_{tr} \leq 0 . \quad (A.38)$$

Assuming that the error is uniformly distributed over this range yields

$$\overline{e}_{tr} = -\frac{2^{-n_b} - 2^{-r_a}}{2} , \quad (A.39)$$

and

$$\sigma_{tr}^2 = \frac{(2^{-n_b} - 2^{-n_a})^2}{12} \quad . \tag{A.40}$$

The error statistics for rounding numbers is similar to these results except that $\bar{e}_{round} = 0$. The variance expression is the same as in (A.40)

APPENDIX B

COMPUTER PROGRAM FOR FIXED-POINT THEORETICAL ANALYSIS
by Jerry D. Moore

The computer program for implementing the theoretical analysis of the fixed-point DSP is presented. The flow chart is given in Fig. B.1 and the program listing in Fig. B.2. The program requires data cards to specify the number of filter coefficients, the number of residues to be calculated, the input signal amplitude and frequency, the radar pulse repetition rate, and the filter coefficient values. The format structure is shown in Table B-1.

TABLE B-1

DATA CARD #	CONTENT	COMMENTS	CARD FORMAT
1	N,M	N=Number of filter Coefficients M=Number of Residues	2I2
	SIGAMP, FREQ,FPRF	SIGAMP=Signal Amplitude FREQ=Signal Frequency FPRF=Pulse Repetition Rate	3F15.4
2,3,...	H(I),I=1,N	Coefficient values	4F15.8 per card

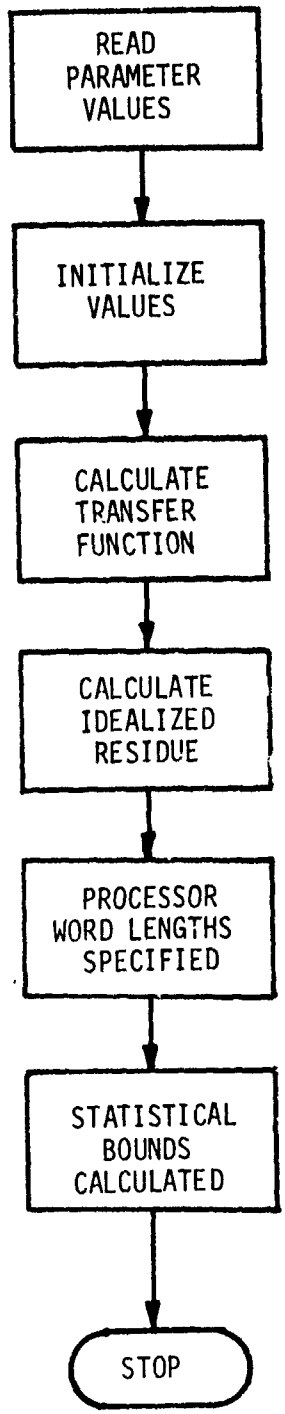


Fig. B.1 Flow Chart for Fixed-Point Processor Theoretical Calculation of Statistical Bounds

```
C  TIOS --- THEORETICAL INTEGRATOR OUTPUT STATISTICS
C
C DEVELOPED ON CONTRACT DAG0072 FOR US ARMY RESEARCH OFFICE
C BY JERRY D. MOORE AT UNIVERSITY OF ALABAMA MARCH 1976
C
C DETERMINES INTEGRATOR OUTPUT AVERAGE VALUE AND VARIANCE
C BOUNDS FOR FIXED POINT RADAR DIGITAL SIGNAL PROCESSOR
C
C SYMBOLS USED
C    SIGAMP     SIGNAL AMPLITUDE AT INPUT
C    U          SIGNAL AMPLITUDE AT RMS OUTPUT
C    FREQ       SIGNAL FREQUENCY
C    FPRF       PULSE RATE FREQUENCY
C    N          NUMBER OF FILTER COEFICIENTS
C    H()        FILTER COEFFICIENTS
C    M          NUMBER OF RESIDUES
C    MX         A/D WORD LENGTH
C    MC         COEF. WORD LENGTH
C    MT         TRUNCATED PRODUCT LENGTH
C    MF         RANGE BIN ACCUMULATOR LENGTH
C    ME         TRUNCATED RMS LENGTH
C    MAXG       MAXIMUM MAGNITUDE OF ERROR AT FILTER OUTPUT
C               DUE TO MX AND MT
C    AVGG       AVERAGE ERROR AT FILTER OUTPUT DUE TO MX AND MT
C    AVGG2      AVERAGE SQUARE ERROR AT FILTER OUTPUT DUE TO
C               MX AND MT
C    SQAG2      SQUARE ROOT OF THE AVERAGE SQUARED VALUE OF
C               THE ERROR AT FILTER OUTPUT DUE TO MX AND MT
C    VARG       VARIANCE AT FILTER OUTPUT DUE TO MX AND MT
C    ETMIN      MINIMUM TRUNCATION ERROR INTRODUCED AT ME
C    AVGET      AVERAGE TRUNCATION ERROR INTRODUCED AT ME
C    VARET      VARIANCE OF ERROR INTRODUCED AT ME
C    ERRLES     ERRORLESS OUTPUT (PERFECT)
C    AVGIO      AVERAGE INTEGRATOR OUTPUT UPPER BOUND
C    AVGER      AVERAGE ERROR AT INTEGRATOR OUTPUT UPPER BOUND
C    VARIO      VARIANCE OF INTEGRATOR OUTPUT UPPER BOUND
C    VARIOL     VARIANCE OF INTEGRATOR OUTPUT LOWER BOUND
C    AVGERL     AVERAGE ERROR AT INTEGRATOR OUTPUT
C               LOWER BOUND
C    ERMIN      MINIMUM INTEGRATOR OUTPUT ERROR
C    ERMAX      MAXIMUM INTEGRATOR OUTPUT ERROR
      REAL MAXG
      DIMENSION H(21)
C    READ IN THE FIXED PARAMETERS
      DO 40 MR=1,4
      READ(5,19) N,M,SIGAMP,FREQ,FPRF,(H(I),I=1,N)
      PRINT 18
      PRINT 20,N,M,SIGAMP,FREQ,FPRF,(H(K),K=1,N)
```

Fig. B.2 Fixed-Point Processor Theoretical Analysis Program

```
      PRINT 23
      PRINT 21
C     INITIALIZE
C     HMSUM = SUM OF MAGNITUDES OF COEFFICIENTS
C     HSQSUM = SUM OF SQUARES OF COEFFICIENTS
C     PI2 = 2*PI
      PI2=6.283185308
      HMSUM=0.
      HSQSUM=0.
      DO 10 I=1,N
      HMSUM=HMSUM+ABS(H(I))
 10   HSQSUM=HSQSUM+H(I)*H(I)
C     INITIALIZATION ENDS
C     CALCULATE TRANSFER FUNCTION AT SIGNAL FREQUENCY
      XREALS =0
      XIMAGS=0
      ARG=PI2*(FREQ/FPRF)
      DO 11 I=1,N
      XREAL=H(I)*COS(ARG*(I-1))
      XIMAG=-H(I)*SIN(ARG*(I-1))
      XREALS=XREALS+XREAL
 11   XIMAGS=XIMAGS+XIMAG
      SUMSQ=XREALS**2+XIMAGS**2
      TRANS=SQRT(SUMSQ)
C     TRANSFER FUNCTION CALCULATION ENDS
C     CALCULATE U, THE SIGNAL AMPLITUDE AT RMS OUTPUT
      U=SIGAMP*TRANS
C     CALCULATION OF U COMPLETE
C     CREATE LOOPS FOR WORD LENGTH VARIATIONS
      DO 14 MX=9,9
      DO 14 MC=9,9
      MDUM=MX+MC-1
      DO 13 MT=6,MDUM
      MF=MT+ALOG10(N)/ALOG10(2.)
      DO 13 ME=6,MF
C     END OF WORD LENGTH SPECIFICATIONS
C     STATISTICAL BOUNDS CALCULATIONS
      DUM1=2.**(-MT)-2.**-(MX+MC-1)
      AVGG=-N*DUM1
      VARG=N*(DUM1**2)/3.+((2.**(-2*MX))/3.)*HSQSUM
      AVGG2=VARG+(AVGG)**2
      MAXG=N*(DUM1*2.)+(2.**-MX)*HMSUM
      SQRAG2=SQRT(AVGG2)
      AVGET=-(2.**(-MT))*((2.**(MF-ME))-1.)
      VARET=(AVGET**2)/3.
      AVG10=M*(1.0069*U+1.42397*SQRAG2+AVGET)
      EROLES=M*U
      AVGER=AVG10-EROLES
```

Fig. B.2 (Continued)

```
      VARIO=M*(2.0276*VARG+0.01595*U**2+5.78031*U*SQRAG2
     1+0.032*AVGG2-9.522*(10.**-5)*(AVGG**2)+VARET)
      ETMIN=AVGET*2.
      DUM2=U**2+2.*AVGG**2-2.82843*U*SQRAG2
      IF (DUM2.LT.0) DUM2=0
      AVGERL=M*(1.0069*SQRT(DUM2)+AVGET-U)
      VARIOL=M*(0.01595*(U**2+2.*AVGG2)-5.7803*U*SQRAG2
     1+VARET)
      IF (VARIOL.LT.0) VARIOL=0
      ERMAX=M*(0.0174*U+1.438821*MAXG)
      IF (U.LE. 1.414*MAXG) GO TO 30
      ERMIN=M*(-0.0216*U-1.38367*MAXG+ETMIN)
      GO TO 31
   30 ERMIN=M*(ETMIN-U)
   31 CONTINUE
C    END OF CALCULATIONS
   13 PRINT 22,MX,MC,MT,MF,ME,EROLES,AVGIO,AVGER,VARIO,
     1AVGERL,VARIOL,ERMIN,ERMAX
   14 CONTINUE
   18 FORMAT(1H1,2X,/,/,/,2X,'BOUNDS ON INTEGRATOR OUTPUT',
     1' ERROR STATISTICS --- DAG0072 1976',/)
   19 FORMAT(2I2,3F15.4,/,5(4F15.8,/))
   20 FORMAT(2X,I3,1X,'COEFFICIENTS',I5,1X,'RESIDUES',5X,
     1'SIGNAL AMPLITUDE = ',F8.6,5X,'SIGNAL FREQUENCY = ',
     1F8.2,5X,'PRF = ',F8.2,/,2X,'COEFFICIENTS',/,2X,10F12.8,
     1/,10F12.8)
  023 FORMAT(2X,/,38X,'-----UPPER BOUND-----',
     123X,'-----LOWER BOUND -----')
   21 FORMAT(2X,/,2X,'MX'  X,'M ',2X,'MT',2X,'MF',2X,'ME',2X,
     1'PERFECT OUTPUT',2X,'AVERAGE OUT',4X,'ERROR OUT',
     14X,'VARIANCE OUT',3X,'ERROR OUT',4X,'VARIANCE OUT',
     15X,'ERMIN',9X,'ERMAX')
   22 FORMAT(5I4,E14.6,E15.6,F14.6,E14.6,E14.6,F14.6,
     1E14.6,E13.6)
   40 CONTINUE
      STOP
      END
```

Fig. B.2 (Continued)

APPENDIX C

DETAILS OF FLOATING-POINT ANALYSIS
by Jerry D. Moore

The analysis of the floating-point DSP described in Chapter 5 is supplemented by this appendix. Equations presented in Chapter 5 are not presented, but the detailed mathematical steps leading to the summary equations are given.

C.1 FILTER OUTPUT

Signal flow graphs are used to determine the output of the fixed-window MTI digital filter used in the DSP. The symbology is similar to that of Appendix A.

Figure C.1 depicts the floating-point filter with an input of $x(n) + e(n)$. The $h(k)(1 + \delta_{N-1-k})$ terms are the transmission factors associated with the filter coefficient multiplication and the truncation of this product. The δ term is the multiplicative statistical factor associated with the truncation. When two of the truncated products are added there is an additional truncation error introduced. This is represented by the ζ terms and is a multiplicative term. Consequently, the terms $(1 + \zeta_j)$, $j = 1, 2, \ldots, N-1$, represent the transmissions of the $(N-1)$ summation processes. A shorthand notation is developed to represent the chain of transmission multiplications that occur, i.e.,

$$\theta_k = (1 + \delta_k) \prod_{i=k}^{N-1} (1 + \zeta_i), \quad k = 1, 2, \ldots, N-1,$$

$$\theta_0 = \theta_1 . \tag{C.1}$$

It is seen that for a given output, $w(mN)$, each input sample that is being used will be multiplied by a filter coefficient and a θ term. Mathematically this is given by

$$w(mN) = \sum_{n=0}^{N-1} h(n)\, \theta_{N-1-n} \left[x(mN - n) + e(mN - n) \right] . \tag{C.2}$$

Separation of the $x()$ and $e()$ terms and manipulating gives

$$w(mN) = \sum_{n=0}^{N-1} h(n) \left[\theta_{N-1-n} + 1 - 1 \right] x(mN - n)$$

$$+ \sum_{n=0}^{N-1} h(n)\, \theta_{N-1-n}\, e(mN - n) , \tag{C.3}$$

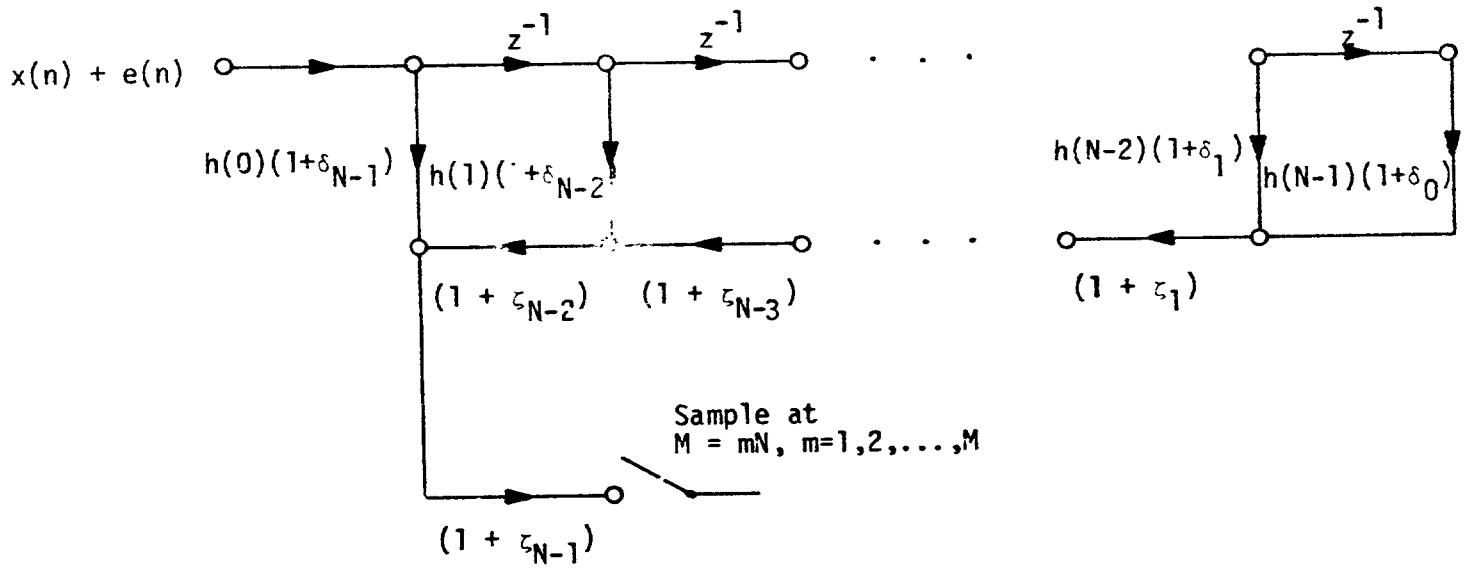


Fig. C.1 Signal Flow Graph for Floating-Point MTI Filter

where 0 has been added by the + 1 - 1 operation. Thus

$$w(mN) = \sum_{n=0}^{N-1} h(n)\, x(mN - n)$$

$$+ \sum_{n=0}^{N-1} [h(n)(\theta_{N-1-n} - 1)\, x(mN - n) + h(n)\, \theta_{N-1-n}\, e(mN - n)]. \qquad (C.4)$$

This yields the equations of (5.1), (5.2) and (5.3).

C.2 INTEGRATOR OUTPUT

The residues are calculated by the RMS Unit and then truncated prior to being summed in the integrator. Since these are floating point operations, there is error introduced by the addition operation, i.e., the sum will be truncated back to the same number of bits used in each input word. A signal flow graph is shown in Fig. C.2. There is a strong similarity to the graph of Fig. C.1, but r subscripts have been included for the multiplicative error parameters δ and ζ. This structure assumes that M residues are used to form an integrator output. The chain of transmission multiplications is written as

$$\theta_{r,k} = (1 + \delta_{r,k}) \prod_{i=k}^{M-1} (1 + \zeta_{r,i}),\ k = 1, 2, \ldots, M-1,$$

$$\theta_{r,0} = \theta_{r,1}. \qquad (C.5)$$

The integrator output can be expressed as

$$INT(M) = \sum_{m=1}^{M} r(mN)\, \theta_{r,m-1}$$

$$= \sum_{m=0}^{M-1} [\theta_{r,m} + 1 - 1]\, r[(m + 1)N]$$

$$= \sum_{m=1}^{M} r(mN) + \sum_{m=0}^{M-1} [\theta_{r,m} - 1]\, r[(m + 1)N]. \qquad (C.6)$$

This yields the equations of (5.5) and (5.6).

C.3 AVERAGE ERROR AND VARIANCE OF INTEGRATOR OUTPUT

A determination of the average error expression of (5.8) follows immediately from (5.5) and (5.7) by assuming statistical independence of

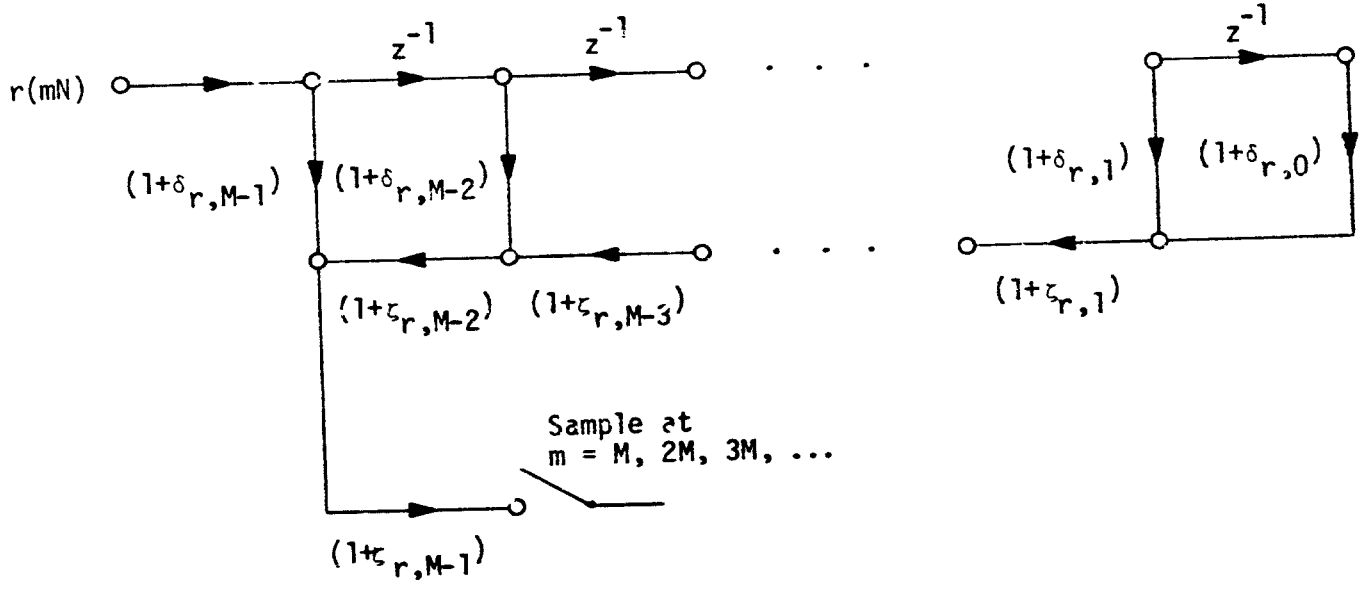


Fig. C.2 Signal Flow Graph for Floating-Point Integrator

the residue terms.

The variance of the integrator error will be equal to the variance of the integrator output, i.e.,

$$\sigma^2_{INTE} = \sigma^2_{INT} = \overline{\left[\sum_m r(mN) + e_{int}(M)\right]^2} - \left[\sum_m \bar{r} + \bar{e}_{int}(M)\right]^2$$

$$= \overline{\sum_m r(mN)^2} + 2\,\overline{e_{int}(M) \sum_m r(mN)} + \overline{e^2_{int}}(M) - \left[\sum_m \bar{r}\right]^2 - 2\,\bar{e}_{int}(M) \sum_m \bar{r} - \overline{e_{int}}^2(M)$$

$$= \sum_m \sum_m \left[\overline{r(mN)r(mN)} - \bar{r}^2\right] + \overline{e^2_{int}}(M) - \bar{e}_{int}^2(M) + 2\left[\overline{e_{int}(M) \sum_m r(mN)} - \bar{e}_{int}(M) \sum_m \bar{r}\right]. \quad (C.7)$$

For $m \neq n$ the term inside the double summation is equal to zero, thus

$$\sigma^2_{INTE} = M\,\sigma_r^2 + \sigma_{int}^2 + 2\left[\overline{e_{int}(M) \sum_m r(mN)} - \bar{e}_{int}(M) \sum_m \bar{r}\right]. \quad (C.8)$$

The bracketed terms are now evaluated

$$\overline{e_{int}(M) \sum_m r(mN)} = \sum_{k=1}^{M} \sum_{m=1}^{M} \overline{(\theta_{r,k-1} - 1)\, r(kN)\, r(mN)}$$

$$= \sum_{k=1}^{M} \sum_{m=1}^{M} (\bar{\theta}_{r,k-1} - 1)\, \overline{r(kN)\, r(mN)}, \quad (C.9)$$

where statistical independence between θ and r is assumed. Also,

$$\bar{e}_{int}(M) \sum_{m=1}^{M} \bar{r} = M\bar{r} \sum_{k=1}^{M} (\bar{\theta}_{r,k-1} - 1)\, \bar{r}$$

$$= M\bar{r}^2 \sum_{k=1}^{M} (\bar{\theta}_{r,k-1} - 1). \quad (C.10)$$

From (C.9) when $k \neq m$ then $\overline{r(kN)\, r(mN)} = \bar{r}^2$, thus (C.9) minus (C.10) can be written as

$$(M-1)\,\bar{r}^{\,2} \sum_{k=1}^{M} (\bar{\theta}_{r,k-1} - 1) + \sum_{k=1}^{M} (\bar{\theta}_{r,k-1} - 1)\,\overline{r^2} - M\bar{r}^{\,2} \sum_{k=1}^{M} (\bar{\theta}_{r,k-1} - 1)$$

$$= (\overline{r^2} - \bar{r}^{\,2}) \sum_{k=1}^{M} (\theta_{r,k-1} - 1)$$

$$= \sigma_r^{\,2} \sum_{k=0}^{M-1} (\bar{\theta}_{r,k} - 1)\,. \qquad (C.11)$$

Using (C.11) in (C.8) gives (5.9).

Determining the variance σ^2_{int} follows from (5.6) by finding

$$\overline{e^2_{int}(M)} = \sum_{k=1}^{M} \sum_{m=1}^{M} \overline{(\theta_{r,k-1} - 1)(\theta_{r,m-1} - 1)\, r(kN)\, r(mN)}\,,$$

$$\overline{e_{int}(M)}^{\,2} = \left[\sum_{m=1}^{M} (\bar{\theta}_{r,m-1} - 1)\,\bar{r} \right]^2 = \bar{r}^{\,2} \sum_{m=1}^{M} \sum_{k=1}^{M} (\bar{\theta}_{r,n-1} - 1)(\bar{\theta}_{r,k-1} - 1)\,,$$

$$\sigma^2_{int} = \sum_{k=1}^{M} \sum_{m=1}^{M} \Big[\overline{(\theta_{r,k-1} - 1)(\theta_{r,m-1} - 1)\, r(kN)\, r(mN)} - (\bar{\theta}_{r,k-1} - 1) \cdot (\bar{\theta}_{r,m-1} - 1)\,\bar{r}^{\,2} \Big]\,. \qquad (C.12)$$

For $k \neq m$ the difference is equal to zero, thus

$$\sigma^2_{int} = \sum_{m=1}^{M} \Big[\overline{(\theta_{r,m-1} - 1)^2}\,\overline{r^2} - (\bar{\theta}_{r,m-1} - 1)^2\,\bar{r}^{\,2} \Big]\,. \qquad (C.13)$$

Changing the index for summation yields the results of (5.10).

The bounding procedure used on the residue calculation is complete in (5.11) through (5.19). The results depend on the filter output statistics $\bar{g}$ and $\overline{g^2}$. These quantities are determined from (5.3), i.e.,

$$\bar{g}(mN) = \sum_{n=0}^{N-1} h(n) \Big[\overline{(\theta_{N-1-n} - 1)\, x(mN - n)} + \bar{\theta}_{N-1-n}\,\bar{e}(mN - n) \Big], \qquad (C.14)$$

where θ and e are assumed statistically independent. Since $\bar{e} = 0$, it follows that

$$\bar{g} = \sum_{n=0}^{N-1} h(n) \overline{(\theta_{N-1-n} - 1)\, x(mN - n)}$$

$$= \sum_{n=0}^{N-1} h(n) \left[\overline{\theta_{N-1-n}\, x(mN - n)} - \bar{x} \right] . \qquad (C.15)$$

Assuming that the x() samples come from a doppler signal that has zero mean, i.e., $\bar{x} = 0$, and that θ is statistically independent from x given that x is positive or given that x is negative yields

$$\bar{g} = \frac{A}{\pi} \sum_{n=0}^{N-1} h(n) \left[\bar{\theta}^{+}_{N-1-n} - \bar{\theta}^{-}_{N-1-n} \right] , \qquad (C.16)$$

where A is the peak value of the sinusoidal signal x() and half-wave average values are used for $\bar{x}^{+}$ and $\bar{x}^{-}$, i.e., $\bar{x}^{+} = -\bar{x}^{-} = 2A/\pi$.

The mean squared value is more complicated, but squaring (5.3) and taking the expected value gives,

$$\overline{g^2} = \sum_{n=0}^{N-1} \sum_{k=0}^{N-1} h(n)\, h(k) \left[\overline{(\theta_{N-1-n} - 1)(\theta_{N-1-k} - 1)\, x(mN - n)\, x(mN - k)} \right.$$

$$\left. + \overline{\theta_{N-1-n}\, \theta_{N-1-k}\, e(mN - n)\, e(mN - k)} \right] , \qquad (C.17)$$

where $\bar{e} = 0$ has been utilized to obtain this result. For $n \neq k$ statistical independence can be assumed between the e() terms and also between the θ and x() terms, thus

$$\bar{g}^2 = \sum_{\substack{n \\ n \neq k}} \sum_{k} h(n)\, h(k)\, \overline{\theta_{N-1-n}\, x(mN - n)}\; \overline{\theta_{N-1-k}\, x(mN - k)}$$

$$+ \sum_{n=0}^{N-1} h^2(n) \left[\overline{(\theta_{N-1-n} - 1)^2\, x^2(mN - n)} + \overline{\theta^2_{N-1-n}}\; \overline{e^2} \right]$$

$$= \frac{A^2}{\pi^2} \sum_{\substack{n \\ n \neq k}} \sum_{k} h(n)\, h(k) \left(\bar{\theta}^{+}_{N-1-n} - \bar{\theta}^{-}_{N-1-n} \right) \left(\bar{\theta}^{+}_{N-1-k} - \bar{\theta}^{-}_{N-1-k} \right)$$

$$+ \sum_{n=0}^{N-1} h^2(n) \left[\theta^2_{N-1-n} \left(\overline{x^2} + \overline{e^2} \right) - \bar{\theta}_{N-1-n}\, \overline{x^2} + \overline{x^2} \right] . \qquad (C.18)$$

The mean squared value of x is the rms value squared, i.e., $\overline{x^2} = A^2/2$, thus simple manipulation yields (5.21) when $\overline{\theta^2} = 1/2[\overline{\theta+^2} + \overline{\theta-^2}]$.

C.3 FLOATING-POINT STATISTICS

Evaluation of the statistics of the floating-point parameters θ_k and $\theta_{r,k}$ is presented in this section. The procedure is identical for each of these, thus a generalization is made from the θ_k derivation. Using (C.1) and taking the average value gives,

$$\overline{\theta+_k} = \overline{(1+\delta+)_k} \prod_{i=k}^{N-1} \overline{(1+\zeta+)_i}$$

$$= (1+\overline{\delta+})_k \prod_{i=k}^{N-1} (1+\overline{\zeta+})_i$$

$$= (1+\overline{\delta+})(1+\overline{\zeta+})^{N-k}, \qquad (C.19)$$

where statistical independence and identical distributions are assumed between index values. The mean value for $\overline{\theta_k}$ is similar except the minus subscripts be used on the δ and ζ values. The mean squared value is

$$\overline{\theta+_k^2} = \overline{(1+\delta+)_k^2} \prod_{i=k}^{N-1} \overline{(1+\zeta+)_i^2}$$

$$= (1+2\overline{\delta+}+\overline{\delta+^2})(1+2\overline{\zeta_+}+\overline{\zeta_+^2})^{N-k}. \qquad (C.20)$$

The negative statistic is obtained by using the appropriate subscript on δ and ζ. Evaluation of the $\theta_{r,k}$ statistics can be accomplished by replacing N by M and using δ_r and ζ_r associated with the integrator word lengths.

Truncation is proposed for the operations in the DSP. A general development for describing the resulting error is presented. A floating-point word containing b fractional bits in the mantissa (after truncation) and C integer bits in the exponent, (sign bits are not included in these counts) is used and ε is the multiplicative error term. The truncated word, x_T is expressed as

$$x_T = x + \varepsilon x . \qquad (C.21)$$

For positive values of x the error is negative, i.e.,

$$-2^{-b} \cdot 2^C < \varepsilon x \leq 0, \; x \geq 0 . \qquad (C.22)$$

The minimum value of x is

$$x_{min} = 0.5 \cdot 2^{c}, \; x \geq 0 \; . \tag{C.23}$$

Thus

$$-2^{-(b-1)} < \varepsilon \leq 0, \; x \geq 0 \; ,$$

$$\bar{\varepsilon} = -2^{-b}, \; x \geq 0 \; ,$$

$$\sigma_{\varepsilon}^{2} = \frac{2^{-2b}}{3}, \; x \geq 0 \; , \tag{C.24}$$

where a uniform distribution has been assumed for ε.
Negative values of x are expressed in two's complement form, i.e.

$$|x| = M \cdot 2^{c} \; ,$$

$$x^{**} = (2 - M) \cdot 2^{c} = M^{**} \cdot 2^{c} \; , \tag{C.25}$$

where M is the mantissa of the magnitude of x. Standard floating-point form dictates that

$$0.5 \leq M < 1 \; . \tag{C.26}$$

Thus,

$$1 < M^{**} \leq 1.5 \; ,$$

$$2^{c} < x^{**} \leq 1.5 \cdot 2^{c} = 3 \cdot 2^{c-1} . \tag{C.27}$$

Truncation of x^{**} will cause a negative error, i.e.,

$$x_{T}^{**} = x^{**} + \varepsilon x^{**} . \tag{C.28}$$

Considering (C.22) gives

$$(2 - 2^{-b})2^{c} < \varepsilon x^{**} \leq (2 - 0) \cdot 2^{c} \; ,$$

$$(2 - 2^{-b}) < \varepsilon M^{**} \leq (2 - 0) \equiv 0 \; ,$$

$$(2 - 2^{-b}) < \varepsilon (2 - M) \leq 0 \; ,$$

$$(2 - 2^{-b}) < \varepsilon (2 - 2^{-1}) \leq 0 \; . \tag{C.29}$$

In two's complement multiplication and division by 2, a negative number is shifted with zeros brought in from the right and ones from the left. Thus, to get the 2^{-1} term to appear in the 2^{-b} position requires division by $2^{(b-1)}$. It follows that

$$0 \leq \varepsilon < 2^{-(b-1)},$$

$$\bar{\varepsilon} = 2^{-b},$$

$$\sigma_\varepsilon^2 = \frac{2^{-2b}}{3}. \qquad \text{(C.30)}$$

Applying the results of (C.24) and (C.30) to the development of the DSP parameters gives

$$\bar{\delta}_+ = -\bar{\delta}_- = -2^{-(MTM-1)},$$

$$\overline{\delta_+^2} = \overline{\delta_-^2} = \frac{16}{3} \cdot 2^{-2MTM},$$

$$\bar{\zeta}_+ = -\bar{\zeta}_- = -2^{-(MFM-1)},$$

$$\overline{\zeta_+^2} = \overline{\zeta_-^2} = \frac{16}{3} \cdot 2^{-2MFM},$$

$$\bar{\delta}_r = -2^{-MEM},$$

$$\overline{\delta_r^2} = \frac{4}{3} \cdot 2^{-2MEM},$$

$$\bar{\zeta}_r = -2^{-MSM},$$

$$\overline{\zeta_r^2} = \frac{4}{3} 2^{-2MSM}. \qquad \text{(C.31)}$$

These results can be used to evaluate the θ statistics of (C.19) and (C.20) and consequently the expressions of (C.16) and (C.18) and the error statistics given in Chapter 5.

APPENDIX D

COMPUTER PROGRAM FOR FLOATING-POINT THEORETICAL ANALYSIS
by Jerry D. Moore

The computer program for implementing the theoretical analysis of the floating-point DSP is presented. The flow chart is given in Fig. D.1 and the program listing in Fig. D.2. The program requires data cards to specify parameters as shown in Table B-1 for the fixed-point processor.

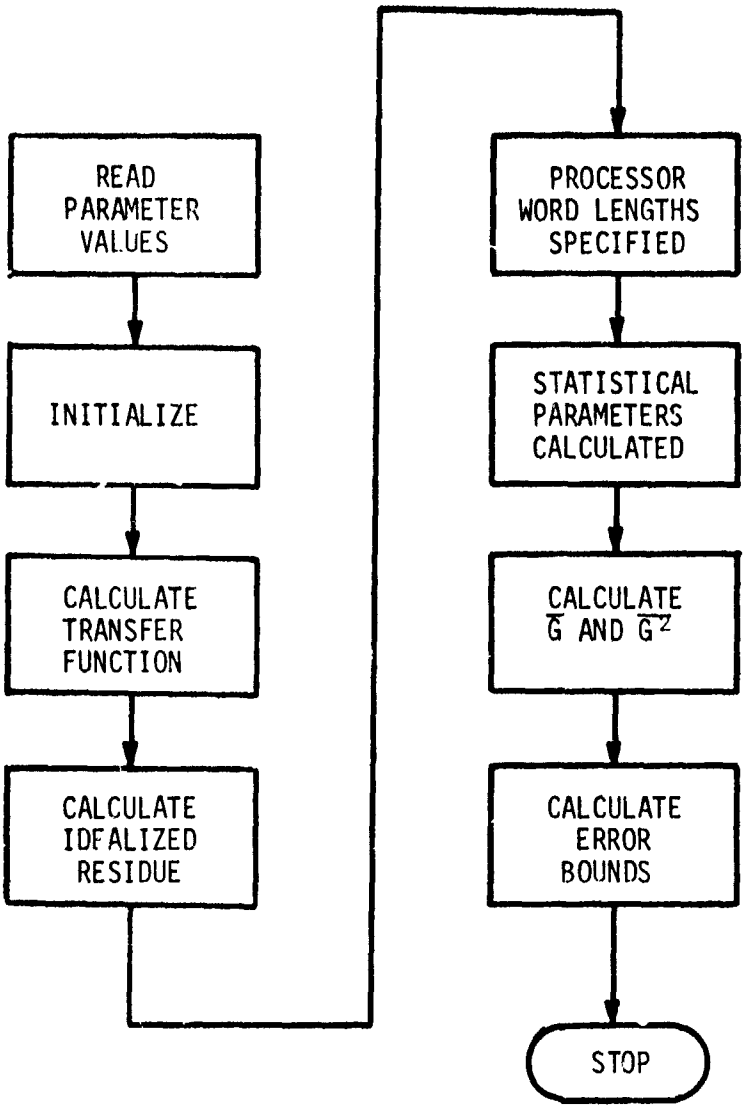


Fig. D.1 Flow Chart for Floating-Point Processor Theoretical Calculation of Statistical Bounds

```
C TIOS2---THEORETICAL INTEGRATOR OUTPUT STATISTICS
C
C DEVELOPED ON CONTRACT DAG0072 FOR US ARMY RESEARCH OFFICE
C BY JERRY D. MOORE AT THE UNIVERSITY OF ALABAMA MAY 1976
C
C DETERMINES INTEGRATOR OUTPUT AVERAGE VALUE AND VARIANCE
C BOUNDS FOR FLOATING-POINT RADAR DIGITAL SIGNAL PROCESSOR
C
C SYMBOLS USED
C      DELTAP      DELTA FOR POSITIVE VALUES
C      ZETAP       ZETA FOR POSTIVE VALUES
C      ALL OF THE FOLLOWING TERMS ARE THE AVERAGE VALUE OF,
C      THE TERM INDICATED:
C      DELTAR      DELTA  FOR THE R VALUE
C      ZETAR       ZETA FOR THE R VALUE
C      A 2 FOLLOWING THE DELTA OR ZETA MEANS SQUARED
C      THETAP      THETA FOR POSITIVE VALUES
C      THETAN      THETA FOR NEGATIVE VALUES
C      THETAR      THETA FOR R VALUES
C      L2          INPUT QUANTIZATISN ERROR SQUARED
C      ERRUP       ERROR AT OUTPUT UPPER BOUND
C      ERRLO       ERROR AT OUTPUT LOWER BOUND
C      VARERU      VARIANCE AT OUTPUT UPPER BOUND
C      VARERL      VARIANCE AT OUTPUT LOWER BOUND
C
      DIMENSION H(20)
      DIMENSION ARTHET(20)
      COMMON/P/ DELTAP,ZETAP,DELTP2,ZETAP2
      COMMON/R/ DELTAR,ZETAR,DELTR2,ZETAR2
C
C READ IN FIXED PARAMETERS
      DO 99 NRUNS=1,1
      READ(5,19)N,M,SIGAMP,FREQ,FPRF,(H(I),I=1,N)
      PRINT 18
      PRINT 20,N,M,SIGAMP,FREQ,FPRF,(H(K),K=1,N)
      PRINT 23
      PRINT 21
C INITIALIZE
      PI=3.141592624
      PI2=6.283185308
C END INITIALIZATION
C
C CALCULATE TRANSFER FUNCTION AT SIGNAL FREQUENCY
      XREALS=0
      XIMGS=0
      ARG=PI2*(FREQ/FPRF)
      DO 11 I=1,N
      XREAL=H(I)*COS(ARG*(I-1))
```

Fig. D.2 Floating-Point Processor Theoretical Analysis Program

```
            XIMG=-H(I)*SIN(ARG*(I-1))
            XREALS=XREAL+XREALS
         11 XIMGS=XIMG+XIMGS
            SUMSQ=XREALS**2+XIMGS**2
            TRANS=SQRT(SUMSQ)
            U=SIGAMP*TRANS
            EROLES=M*U
      C END OF TRANSFER CALCULATION
      C
      C CREATE LOOPS FOR WORD LENGTH  ARIATION
            DO 14 MX=9,9
            DO 13 MTM=9,15
            MFM=MTM
            MFMM1=MFM-1
            DO 13 MEM=6,MFMM1
            MSM=MEM
      C END OF WORD LENGTH SPECIFICATION
      C
      C CALCULATE STATISTICAL PARAMETERS
            DELTAP=-2.**(1-MTM)
            ZETAP=-2.**(1-MFM)
            DELTP2=(16./3.)*2.**-(2*MTM)
            DELTAR=-2.**-MEM
            ZETAR=-2.**-MSM
            DELTR2=(4./3.)*2.**-(2*MEM)
            ZETAR2=(4./3.)*2.**-(2*MSM)
            E2=(2.**-(2*MX))/3.
            GAM=0.0069
            GAM2=0.000256
      C END OF STATISTICAL PARAMETER  ALCULATIONS
      C
      C CALCULATE G
            GDUM=0.
            DO 30 NN=1,N
            NNN=NN-1
            NNNN=N-1-NNN
            DUM1=THETAP(N,NNNN)-THETAN(N,NNNN)
            ARTHET(NN)=DUM1
         30 GDUM=GDUM+H(NN)*DUM1
            G=GDUM*(SIGAMP/PI)
      C END CALCULATION OF G
      C
      C CALCULATE G2
            GDUM1=0.
            NM1=N-1
            DO 31 NN=1,N
            DO 31 KK=1,N
            IF(KK.EQ.NN) GO TO 31
```

Fig. D.2 (Continued)

```
      GDUM1=GDUM1+H(NN)*H(KK)*ARTHET(NN)*ARTHET(KK)
 31   CONTINUE
      G2DUM=((SIGAMP**2)/(PI**2))*GDUM1
      GDUM3=0.
      DO 32 NN=1,N
      NNN=NN-1
      NNNN=N-1-NNN
      DUM2=((SIGAMP**2)/4.+E2/2.)*(THETP2(N,NNNN)+THETN2(N,NNNN))
      DUM3=THETAP(N,NNNN)+THETAN(N,NNNN)
      GDUMX=(1.-DUM3)*((SIGAMP**2)/2.)
      GDUMY=(DUM2+GDUMX)*H(NN)*H(NN)
      GDUM3=GDUM3+GDUMY
 32   CONTINUE
      G2=G2DUM+GDUM3
C END CALCULATION OF G2
C
C CALCULATE ERRUP AND ERRLO
      DUM4=SQRT(2.*G2)
      DUM6=U+DUM4
      DUM10=1.+GAM
      RMAX=DUM10*DUM6
      DUM7=U-DUM4
      DUM11=1.+2.*GAM+GAM2
      R2MAX=DUM11*DUM6**2
      R2MIN=DUM11*DUM7**2
      RDUM=U**2-2.*U*DUM4+2.*G**2
      RDUM1=0.
      IF(RDUM.GT.0.) RDUM1=SQRT(RDUM)
      RMIN1=DUM10*RDUM1
      ADUM7=ABS(DUM7)
      ADUM12=ABS(U-SQRT(2.)*ABS(G))
      RMIN2=DUM10*AMIN1(ADUM7,ADUM12)
      RMIN=AMAX1(RMIN1,RMIN2)
      DUM5=0.
      DO 33 MM=1,M
      MMM=MM-1
 33   DUM5=DUM5+THETAP(M,MMM)-1.
      ERRUP=GAM*M*U+DUM10*(M*DUM4+DUM5*DUM6)
      ERRLO=RMIN*(M+DUM5)-M*U
C END CALCULATION OF ERRUP AND ERRLO
C
C CALCULATE VARERU AND VARERL
      VARRUP=R2MAX-RMIN**2
      DUM13=R2MIN-RMAX**2
      VARRLO=AMAX1(DUM13,0.)
      DUM8=0.
      DUM9=0.
      DO 34 MM=1,M
```

Fig. D.2 (Continued)

```
      MMM=MM-1
      DUM8=DUM8+THETR2(M,MMM)-2.*THETAR(M,MMM)+1
 34   DUM9=DUM9+(THETAR(M,MMM)-1.)**2
      VAREIU=R2MAX*DUM8-(RMIN**2)*DUM9
      EILDUM=R2MIN*DUM8-(RMAX**2)*DUM9
      VAREIL=AMAX1(EILDUM,0.)
      VARERU=(M+2.*DUM5)*VARRUP+VAREIU
      VARERL=(M+2.*DUM5)*VARRLO+VAREIL
C END CALCULATION OF VARERU AND VARERL
C
      WRITE(6,40) MX,MTM,MFM,MEM,MSM,EROLES,ERRUP,VARERU,
     1ERRLO,VARERL
 15   CONTINUE
 14   CONTINUE
 18   FORMAT(1H1,2X,/,/,/,2X,'BOUNDS ON INTEGRATOR OUTPUT',
     1' ERROR STATISTICS ---FLOATING POINT --- DAG0072 1976',/)
 19   FORMAT(2I2,3F12.4,/,5(4E20.13,/))
 20   FORMAT(2X,I3,1X,'COEFFICIENTS',I5,1X,'RESIDUES',5X,
     1'SIGNAL AMPLITUDE = ',F8.6,5X,'SIGNAL FREQUENCY = ',
     1F8.2,5X,'PRF = ',F8.2,/,2X,'COEFFICIENTS',/,2X,7E15.8,
     1/,7E15.8,/,7E15.8)
025  FORMAT(2X,/,56X,'-----UPPER BOUND-----',
     119X,'-----LOWER BOUND-----')
 21   FORMAT(2X,/,5X,'MX',2X,'MTM',2X,'MFM',2X,'MEM',2X,'MSM',
     16X,'PERFECT OUTPUT',11X,'ERROR OUT',8X,'VARIANCE OUT',
     111X,'ERROR OUT',8X,'VARIANCE OUT')
 40   FORMAT(2X,5I5,5E20.6)
 99   CONTINUE
      STOP
      END
```

Fig. D.2 (Continued)

```
      FUNCTION THETAP(N,K)
      COMMON/P/ DELTAP,ZETAP,DELTP2,ZETAP2
      THETAP=(1.+DELTAP)*(1.+ZETAP)**(N-K)
      RETURN
      END
      FUNCTION THETP2(N,K)
      COMMON/P/ DELTAP,ZETAP,DELTP2,ZETAP2
      THETP2=(1.+2.*DELTAP+DELTP2)*(1.+2.*ZETAP+ZETAP2)**(N-K)
      RETURN
      END
      FUNCTION THETAN(N,K)
      COMMON/P/ DELTAP,ZETAP,DELTP2,ZETAP2
      THETAN=(1.-DELTAP)*(1.-ZETAP)**(N-K)
      RETURN
      END
      FUNCTION THETN2(N,K)
      COMMON/P/ DELTAP,ZETAP,DELTP2,ZETAP2
      THETN2=(1.-2.*DELTAP+DELTP2)*(1.-2.*ZETAP+ZETAP2)**(N-K)
      RETURN
      END
      FUNCTION THETAR(M,K)
      COMMON/R/ DELTAR,ZETAR,DELTR2,ZETAR2
      THETAR=(1.+DELTAR)*(1.+ZETAR)**(M-K)
      RETURN
      END
      FUNCTION THETR2(M,K)
      COMMON/R/ DELTAR,ZETAR,DELTR2,ZETAR2
      THETR2=(1.+2.*DELTAR+DELTR2)*(1.+2.*ZETAR+ZETAR2)**(M-K)
      RETURN
      END
```

Fig. D.2 (Continued)

APPENDIX E

COMPUTER PROGRAM FOR SIMULATION OF THE FIXED-POINT PROCESSOR
by Bhadrayu J. Trivedi

This appendix contains the FORTRAN computer program to simulate the fixed-point processor. In the following discussion the input cards are described in the sequence actually used by the program.

Input to MAIN

Card-1 -- NRUN
Explanation -- Number of simulation runs to be made.
Format -- I2

Inputs to STARTQ

Card-2 -- SCAL, SCRDB, XM2, SIGMAF, CLTFAS
Explanation -- SCAL is a scale factor for normalizing signal plus clutter, in order to use all the dynamic range of the A/D converter. SCRDB is the signal-to-clutter power ratio in dB. SCRDB is 0.0 if no clutter is to be used. XM2 is the ratio of dc to ac clutter power. SIGMAF is the standard deviation frequency in Hz of the clutter power spectrum. CLTFAS is the phase of the clutter.
Format -- 5F10.4

Card-3 -- SAMPF
Explanation -- SAMPF is the sampling or pulse repetition frequency in Hz.
Format -- F10.4

Card-4 -- FDOP
Explanation -- Doppler Frequency in Hz.
Format -- F10.4

Card-5 -- NDWEL, NDELAY, NCYCLE
Explanation -- NDWEL is the number of antenna dwells. NDELAY is the number of filter coefficients. NCYCLE is the number of residues.
Format -- 3I3

Card-6 -- (CONST(K), K=1 NDELAY)
Explanation -- CONST(K) is the Kth digital filter coefficient. Note that a maximum of seven coefficients can be specified on a card. For more coefficients, extra cards should be used.
Format -- 7F10.6

Card-7 -- MX, MC, MT, MF, ME, MS
Explanation -- These are the processor word lengths in bits as described in Table 2.1.
Format -- 6I3

Card-8 -- (IPRINT(I), I=1,9)
Explanation -- IPRINT(I) is the Ith print option such that:

IPRINT(I) = 0 No print
= 1 Print

where

I = 1 Input to A/D converter
= 2 Output from A/D converter
= 3 Output from multiplier
= 4 Truncated output from multiplier
= 5 Input to accumulator and accumulator output magnitude
= 6 Final filter output magnitude (input to RMS unit)
= 7 Output of RMS unit
= 8 Truncated output of RMS unit
= 9 Integrator output

Format -- 9I3

Inputs to MAIN

Card-9 -- JPRINT, JADCLU, KTHEOR
Explanation -- JPRINT controls the printing of RMS output statistics.

JPRINT = 0 No print
= 1 Print

JADCLU controls addition of clutter to signal.

JADCLU = 0 No clutter
= 1 Add clutter

KTHEOR controls the type of filter coefficients to be used for computing theoretical output (infinite precision answer)

KTHEOR = 0 Unquantized coefficients
= 1 Quantized coefficients

Format -- 3I1

```
C    MAIN PROGRAM TFTNG
C    SHOWS QUANTIZATION ERROR OF NON-RECURSIVE FILTER
C    ASSUME STATIONARY TARGET WITH DOPPLER
C    MAY REPEAT THE ANTENNA DWELL BECAUSE CLUTTER IS RANDOM
      DIMENSION OUTAR(512),OUTPAR(512),OUTTHR(512)
      COMMON/FLTPAR/NDWEL,NDELAY,NCYCLE,ICONST(512),LEVELX,
     * LEVELF,LEVELS,LEVELE,LEVELC,CONST(512),XCONST(512),
     * LEVELT
      COMMON/PRINT/IPRINT(15)
      COMMON/PULTEL/XI(512),XQ(512),JADCLU
      COMMON/RADPAR/PI2,DELT,NPULSE,FDOP,AS,BS,CS,DS
      COMMON/UPDPUL/TIM,PHASE,ICLUT,CI(512),CQ(512)
      READ (5,1000) NRUN
 1000 FORMAT (I2)
      DO 1001 I=1,NRUN
    1 FORMAT(/1X,'NUMBER OF A/D CONVERTER SATURATIONS = ',I5/)
    2 FORMAT(1X,I4,'  OVERFLOWS IN ACCUMULATION, IN PHASE ',
     * 'PULSE NO.',I5)
    3 FORMAT(1X,I4,'  OVERFLOWS IN ACCUMULATION, QUADRAT ',
     * 'PULSE NO.',I5)
    4 FORMAT(1X,' OVERFLOW IN INTEGRATION DURING CYCLE NO. ',
     * I5)
    5 FORMAT(1X,'INTEGRATOR OUTPUT FOR DWELL NO. ',I5,5X,
     * 'OCTAL = ',O12,5X,'REAL = ',F12.6)
    6 FORMAT(/25X,'ANTENNA DWELL NUMBER ',I5/25X,29(1H*))
    7 FORMAT(/5X,'IN-PHASE CHANNEL, MTI CYCLE NO.',I4/)
    8 FORMAT(/5X,'QUADRATURE CHANNEL, MTI CYCLE NO.',I4/)
    9 FORMAT(/1X,'OUTPUT OF RMS ALGORITHM',2X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
   10 FORMAT(/1X,'TRUNCATED RMS VALUE',6X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
   16 FORMAT(/1X,'INTEGRATED VALUE',9X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
   17 FORMAT(1X,10(1H*),'OVERFLOW IN INTEGRATION',10(1H*))
  980 FORMAT(1H1,T50,'---- RMS OUTPUT STATISTICS ----',//)
  981 FORMAT(T6,'INFINITE',T28,'ACTUAL RMS HARDWARE',T66,
     *'PERFECT RMS REALIZATION',T107,'RMS DIFFERENCE',/,T6,
     *'PRECISION',/,T2,'NUM ANSWER',T19,'OUTPUT',T32,'ERROUT',
     *T45,'VARIANCE',T58,'OUTPUT',T71,'ERROUT',T84,'VARIANCE',
     *T97,'ERRDIF',T110,'AVGDIF',T121,'VARDIF',/,T2,'---',
     *T6,12(1H*),T19,39(1H*),T58,39(1H*),T97,35(1H*))
  982 FORMAT(1H1,T46,'---- INTEGRATOR OUTPUT STATISTICS ----',
     * //)
  983 FORMAT(T2,'---- AVERAGE ERROR',T32,E12.5,T71,E12.5,T97,
     *F11.4,/,T2,'---- MINIMUM ERROR',T32,E12.5,T71,E12.5,T97,
     *F11.4,/,T2,'---- MAXIMUM ERROR',T32,E12.5,T71,E12.5,T97,
     *F11.4)
  984 FORMAT(1X,I2,7E13.5,3E12.4)
```

Fig. E.1 Listing for the Fixed-Point Simulation Program.

```
  956  FORMAT(3I1)
  22   FORMAT(1X,'NO. OF DWELLS',I3.5X,'SUM OF INTEGRATOR ',
      *'OUTPUTS=',F12.5/1X,'AVG. OF INTEGRATOR OUTPUTS=',F12.5)
  971  FORMAT(/,1X,10(1H*),'THIS SIMULATION CONTAINS CLUTTER')
  972  FORMAT(/,1X,10(1H*),'THIS SIMULATION DOES NOT ',
      * 'CONTAIN CLUTTER')
       CALL STARTG
C*****************************************************************
       READ (5,9AA) JPRINT,JADCLU,KTHEOR
       IF (KTHEOR.EQ.0) PRINT 2051
       IF (KTHEOR.EQ.1) PRINT 2052
 2051 FORMAT (1X,10(1H*),'THEORETICAL OUTPUT COMPUTED',
      * ' WITH IDEAL COEFFICIENTS'/)
 2052 FORMAT (1X,10(1H*),'THEORETICAL OUTPUT COMPUTED',
      * ' WITH QUANTIZED COEFFICIENTS'/)
       IF(JADCLU.NE.0) GO TO 972
       PRINT 972
       GO TO 973
  971  PRINT 971
  973  CONTINUE
       NDUMP=(NDELAY+1)/2
       NDUM=(NDELAY-1)/2
       IF (KTHEOR.EQ.0) GO TO 2053
       HOFF=XCONST(NDUMP)
       DO 991 K=1,NDUM
       HOFF=HOFF+2.*(CONST(NDUMP-K)*COS(PI2*FDOP*DELT*K)
  991  CONTINUE
       GO TO 2054
 2053  HOFF=CONST(NDUMP)
       DO 2055 J=1,NDUM
 2055  HOFF=HOFF+2.*CONST(NDUMP-J)*COS(PI2*FDOP*DELT*J)
 2054  THOUT=AS*HOFF
C*****************************************************************
       GX=2./FLOAT(LEVELX)
       GOUT=2.*FLOAT(LEVELF/LEVELT)/FLOAT(LEVELF)
       GT=2./FLOAT(LEVELT)
       SUMINT=0.0
       IF(JPRINT.EQ.0) GO TO 987
       PRINT 980
       PRINT 981
  987  CONTINUE
       IPOUT=0
       QUERSM=0.
       QPERSM=0.
       QTSQSM=0.
       QPESSM=0.
       QDESSM=0.
       QDIFSM=0.
```


Fig. E.1 (Continued)

```
      OUTMAX=-1000.
      OUTMIN=1000.
      OUPMAX=-1000.
      OUPMIN=1000.
      ODFMAX=-1000.
      ODFMIN=1000.
      NSAT=0
      IX=1234567
      DO 15 IDWEL=1,NDWEL
      CALL RANDU (IX,IY,PND)
      PHASE=(PI2)*PND
C*****************************************************************
      ERRSUM=0.
      FSQSUM=0.
      ERRMIN=1000.
      ERRMAX=-1000.
C*****************************************************************
      IF(JPRINT.EQ.0) GO TO 990
      PRINT 6,IDWEL
  990 CONTINUE
      CALL UPDATQ
      IOUT=0
      IPOUT=0
      PERSUM=0.
      PESQSM=0.
      PERMIN=1000.
      PERMAX=-1000.
      DIFSUM=0.
      DIFSQS=0.
      DIFMIN=1000.
      DIFMAX=-1000.
      DO 17 ICYCLE=1,NCYCLE
C     GENERATE NDELAY SAMPLES EACH OF XI AND XQ
      CALL PULSEQ
C     COMPUTE ACCUMULATOR OUTPUT FOR I AND Q CHANNELS
      IPR=IPRINT(1)+IPRINT(2)+IPRINT(3)+
     * IPRINT(4)+IPRINT(5)+IPRINT(6)
      IF(IPR.EQ.0) GO TO 19
      PRINT 7,ICYCLE
   19 CONTINUE
      CALL FILTNQ(XI,INORMS,NOVFLI,NSAT)
      IF(IPR.EQ.0) GO TO 20
      PRINT 8,ICYCLE
   20 CONTINUE
      CALL FILTNQ(XQ,INORMS,NOVFLQ,NSAT)
C     CHECK FOR OVERFLOW
C     NOVFLI AND NOVFLQ ARE NUMBERS OF OVERFLOWS
      IF(NOVFLI.NE.0) PRINT 2,NOVFLI
```

Fig. E.1 (Continued)

```
      IF(NOVFLO.NE.0) PRINT 3,NOVFLO
C    IMPLEMENT RMS ALGORITHM
      IRMS=IRMSA(INIPMS,INORMS)
      IF(IPRINT(7).EQ.0) GO TO 11
      XIRMS=FLOAT(IRMS)*QT
      PRINT 9,IRMS,XIRMS
   11 CONTINUE
C    TRUNCATE TO ME BITS AND EXPAND TO MF BITS
      IRMST=ITREX(IRMS,LEVELF,LEVELC,LEVELS)
C*****************************************************************
      XIRMST=FLOAT(IRMST)*QOUT
      IF(IPRINT(8).EQ.0) GO TO 13
C*****************************************************************
      PRINT 10,IRMST,XIRMST
   13 CONTINUE
C*****************************************************************
      ERROUT=XIRMST-THEOUT
      ERRSUM=ERRSUM+ERROUT
      AVGERR=ERRSUM/ICYCLE
      ESQSUM=ESQSUM+ERROUT**2
      AVGSQV=ESQSUM/ICYCLE
      AVGMSV=SQRT(AVGSQV)
      SIGERR=AVGSQV-AVGERR**2
      IF(ERRMAX.LT.ERROUT) ERRMAX=ERROUT
      IF(ERRMIN.GT.ERROUT) ERRMIN=ERROUT
      IPRMS=SQRT((INTRMS*QT)**2+(INQRMS*QT)**2)/QT
      IPRMST=ITREX(IPRMS,LEVELF,LEVELC,LEVELS)
      XPRMST=FLOAT(IPRMST)*QOUT
      PEROUT=XPRMST-THEOUT
      PERSUM=PERSUM+PEROUT
      PAVERR=PERSUM/ICYCLE
      PESQSM=PESQSM+PEROUT**2
      PAVSQV=PESQSM/ICYCLE
      PAVMSV=SQRT(PAVSQV)
      PSIGER=PAVSQV-PAVERR**2
      IF(PERMAX.LT.PEROUT) PERMAX=PEROUT
      IF(PERMIN.GT.PEROUT) PERMIN=PEROUT
      ERRDIF=ERROUT-PEROUT
      DIFSUM=DIFSUM+ERRDIF
      AVGDIF=DIFSUM/ICYCLE
      DIFSQS=DIFSQS+ERRDIF**2
      DIFASV=DIFSQS/ICYCLE
      DIFMSV=SQRT(DIFASV)
      VARDIF=DIFASV-AVGDIF**2
      IF(DIFMAX.LT.ERRDIF) DIFMAX=ERRDIF
      IF(DIFMIN.GT.ERRDIF) DIFMIN=ERRDIF
      IF(JPRINT.EQ.0) GO TO 988
      PRINT 984,ICYCLE,THEOUT,XIRMST,ERROUT,SIGERR,APRMST,
```

Fig. E.1 (Continued)

```
     *PEROUT,PSIGER,ERRDIF,AVGDIF,VARDIF
 988  CONTINUE
C**************************************************************
C    INTEGRATION
      CALL ADD(IOUT,IRMST,IOUT,INTOFL,LEVELS)
      IF(INTOFL.NE.0)PRINT 17
      XIOUT=FLOAT(IOUT)*GOUT
      IF(IPRINT(2).EQ.0) GO TO 14
      PRINT 16,IOUT,XIOUT
  14  CONTINUE
      CALL ADD(IPOUT,IPRMST,IPOUT,INTOFL,LEVELS)
      IF(INTOFL.NE.0)PRINT 17
      XPIOUT=FLOAT(IPOUT)*GOUT
      IF(INTOFL.EQ.0) GO TO 12
      PRINT 4,ICYCLE
  12  CONTINUE
      OUT=FLOAT(IOUT)*GOUT
      SUMINT=SUMINT+OUT
      IF(JPRINT.EQ.0) GO TO 989
      PRINT 987,AVGERR,PAVERR,AVGDIF,ERRMIN,PERMIN,DIFMIN,
     *ERRMAX,PERMAX,DIFMAX
 989  CONTINUE
      OUTAR(IDWEL)=XIOUT
      OUTPAR(IDWEL)=XPIOUT
      OUTTHE(IDWEL)=NCYCLE*THEOUT
  15  CONTINUE
      IF(NSAT.NE.0) PRINT 1,NSAT
      PRINT 982
      PRINT 981
      DO 985 IIDWEL=1,NDWEL
      OUTERR=OUTAR(IIDWEL)-OUTTHE(IIDWEL)
      OUTPER=OUTPAR(IIDWEL)-OUTTHE(IIDWEL)
      OUTDIF=OUTERR-OUTPER
      OUERSM=OUERSM+OUTERR
      OPERSM=OPERSM+OUTPER
      ODIFSM=ODIFSM+OUTDIF
      OUAVER=OUERSM/IIDWEL
      OPAVER=OPERSM/IIDWEL
      ODAVER=ODIFSM/IIDWEL
      OESQSM=OESQSM+OUTERR**2
      OPESSM=OPESSM+OUTPER**2
      ODFSSM=ODFSSM+OUTDIF**2
      OUTSIG=(OESQSM/IIDWEL)-OUAVER**2
      OUPSIG=(OPESSM/IIDWEL)-OPAVER**2
      ODFSIG=(ODFSSM/IIDWEL)-ODAVER**2
      IF(OUTMAX.LT.OUTERR) OUTMAX=OUTERR
      IF(OUTMIN.GT.OUTERR) OUTMIN=OUTERR
      IF(OUPMAX.LT.OUTPER) OUPMAX=OUTPER
```

Fig. E.1 (Continued)

```
      IF(OUPMIN.GT.OUTPER) OUPMIN=OUTPER
      IF(ODFMAX.LT.OUTDIF) ODFMAX=OUTDIF
      IF(ODFMIN.GT.OUTDIF) ODFMIN=OUTDIF
      IF(MOD(IIDWEL,10).NE.0) GO TO 992
      PRINT 984,IIDWEL,OUTTHF(IIDWEL),OUTAR(IIDWEL),OUTERR,
     *OUTSIG,OUTPAR(IIDWEL),OUTPER,OUPSIG,OUTDIF,ODAVER,ODFSIG
  992 CONTINUE
  985 CONTINUE
      PRINT 987,OUAVER,OPAVER,ODAVER,OUTMIN,OUPMIN,ODFMIN,
     *OUTMAX,OUPMAX,ODFMAX
      AVGSI=SUMINT/FLOAT(NDWEL)
      IF(IPRINT(11).EQ.0) GO TO 21
      PRINT 22,NDWEL,SUMINT,AVGSI
   21 CONTINUE
 1001 CONTINUE
      STOP
      END


      SUBROUTINE STARTQ
      COMMON/CLTRAT/XN(1024),H(1420),N,ITI,INCRAU
      COMMON/FLTRAP/NDWEL,NDELAY,NCYCLE,YCONST(512),LEVELX,
     * LEVELF,LEVELS,LEVELE,LEVELC,CONST(512),XCONST(512),
     * LEVELT
      COMMON/PRINT/IPRINT(15)
      COMMON/RADPAR/PI,DELT,NPULSE,FDOP,AS,BS,CS,DS
    1 FORMAT(7F10.4)
    2 FORMAT(10I7)
   25 FORMAT(7F10.6)
    3 FORMAT(1H1,1X,'SCALE FACTOR = ',16X,F9.2,/
     * 1X,'SIGNAL-TO-CLUTTER RATIO = ',5X,F9.2,' DB',/
     * 1X,'CLUTTER DC-TO-AC POWER RATIO = ',F9.3,/
     * 1X,'SIGMA OF CLUTTER SPECTRUM = ',3X,F9.2,' HZ',/
     * 1X,'CLUTTER PHASE = ',15X,F9.4,' RAD')
    4 FORMAT(1X,'SAMPLING FREQUENCY = ',10X,F9.2,' HZ')
    5 FORMAT(1X,'DOPPLER FREQUENCY = ',11X,F9.2,' HZ')
    6 FORMAT(1X,'NUMBER OF ANTENNA DWELLS = ',4X,I5,/
     * 1X,'NUMBER OF DELAYS = ',12X,I5,/
     * 1X,'NUMBER OF PULSES INTEGRATED = ',1X,I5)
    7 FORMAT(1X,'FILTER WORD LENGTHS ',4X,'MX',4X,'MC',4X,
     * 'MT',4X,'MF',4X,'ME',4X,'MS'/24X,6I6)
    8 FORMAT(/25X,' FILTER COEFFICIENTS '/)
    9 FORMAT(1X,'COEFF. NO. = ',I5,' ACTUAL = ',F9.6,' OCTAL'
     * ' = ',O12,' QUANTIZED = ',F9.6)
  101 FORMAT(//25X,'XTI INPUT COMPONENTS'
     *//20X,'XI=AS.COS(2.PI.FD.DT)+BS+CS.CI(CLUT)'
     * /20X,'XQ=AS.SIN(2.PI.FD.DT)+BS-CS.CQ(CLUT)')
  102 FORMAT(//1X,'K   = SCALE FACTOR =',12X,F12.5
```

Fig. E.1 (Continued)

```
     */1X,"AS = SIGNAL AMPLITUDE , K =",4X,F12.5
     */1X,"BS = I CHAN. DC CLUTTER , K =",2X,F12.5
     */1X,"DS = Q CHAN. DC CLUTTER , K =",2X,F12.5
     */1X,"CS = I AND Q AC CLUTTER , K =",2X,F12.5
     */1X,"DT = 1/PRF IN SECONDS =",8X,F12.5
     */1X,"FD = INITIAL DOPPLER IN HZ =",3X,F12.5)
C    READ CLUTTER AND SIGNAL PARAMETERS
      N=1024
      READ(5,1)SCAL,SCRDB,XM2,SIGMAF,CLTFAS
      PRINT 3,SCAL,SCRDB,XM2,SIGMAF,CLTFAS
C    READ RADAR PARAMETERS
      READ(5,1)SAMPF
      PRINT 4,SAMPF
C    READ DOPPLER FREQUENCY
      READ(5,1) FDOP
      PRINT 5,FDOP
C    READ FILTER PARAMETERS
      READ(5,2)NDWEL,NDELAY,NCYCLE
      PRINT 6,NDWEL,NDELAY,NCYCLE
      READ(5,25)(CONST(K),K=1,NDELAY)
      READ(5,2)MX,MC,MT,MF,ME,MS
      PRINT 7,MX,MC,MT,MF,ME,MS
      READ(5,2)(IPRINT(I),I=1,9)
      LEVELX=2**MX
      LEVELC=2**MC
      LEVELT=2**MT
      LEVELF=2**MF
      LEVELE=2**ME
      LEVELS=2**MS
      SCR=10**(SCRDB/10.)
C    A/D CONVERSION OF THE FILTER COEFFICIENTS
      QC=2./FLOAT(LEVELC)
      DO 13 IDELAY=1,NDELAY
 13   CALL COEF(CONST(IDELAY),ICONST(IDELAY),QC,LEVELC)
      PRINT 8
      DO 14 IDELAY=1,NDELAY
      IS=ICONST(IDELAY)
      IF(IS.GT.LEVELC/2)IS=IS-LEVELC
      QCONST=FLOAT(IS)*QC
      XCONST(IDELAY)=QCONST
 14   PRINT 9,IDELAY,CONST(IDELAY),ICONST(IDELAY),QCONST
      DELT=1./SAMPF
      NPULSE=NCYCLE*NDELAY
      PI=3.14159265358
      PI2=2.*PI
C    GENERATE GAUSSIAN SAMPLES
      CALL ANIT(43472.)
      SIGRAN=1.
```

Fig. E.1 (Continued)

```
      CALL RANDM(XN,N,0.,SIGRAN)
      SIGTAU=1./(PI2*SIGMAF)
C     SET UP CONSTANTS FOR THE CLUTTER MODEL FILTER IMPULSE
C     RESPONSE
      SIGSS=SIGTAU/DELT
      ITI=SQRT(PI/2.)*SIGSS
      TI=ITI
      SIG1=SIGSS/(SQRT(2.))
      ITMAX=6*ITI
      INCGAU=12+2*((NPULSE-1)/ITI)
C     GENERATE FILTER IMPULSE RESPONSE
      DO 12 I=1,ITMAX
      R=I
      H(I)=EXP(-(R-3.*TI)**2/(2.*(SIG1**2)))
   12 CONTINUE
C     GENERATE SCALE FACTORS
C     CHANGE TO AS=SQRT(SCR)*SCAL
      AS=SQRT(SCR)*SCAL
      BS=SQRT(XM2/(1.+XM2))*SCAL*COS(CLTPAS)
      CS=SQRT(1./(2.*(1.+XM2)))*SCAL
      DS=SQRT(XM2/(1.+XM2))*SCAL*SIN(CLTPAS)
      PRINT 101
      PRINT 102,    SCAL,AS,BS,DS,CS,DELT,FDOP
      RETURN
      END


      SUBROUTINE UPDATE
      COMMON/CLTDAT/XN(1024),H(1420),N,ITI,INCGAU
      COMMON/RADPAR/PI2,DELT,NPULSE,FDOP,AS,BS,CS,DS
      COMMON/UPDPUL/TIM,PHASE,ICLUT,CI(512),CQ(512)
      DIMENSION XNT(24)
      DATA K/0/
      ICLUT=0
C     GENERATE NPULSE SAMPLES OF CLUTTER
      DO 11 I=1,NPULSE
      CI(I)=0.
   11 CQ(I)=0.
C     HERE TO 15 IF MORE THAN N GAUSSIAN SAMPLES ARE NEEDED
      IF((K+INCGAU).LE.N) GO TO 15
      K=0
      CALL RANDM(XN,N,0.,1.)
   15 CONTINUE
      DO 13 I=1,INCGAU
   13 XNT(I)=XN(I+K)
      K=K+INCGAU
      JG=INCGAU/2
      DO 14 I=1,NPULSE
```


Fig. E.1 (Continued)

```
      DO 14 J=1,6
      JINC=(I-1)/ITI
      IM=I-JINC*ITI
      M=(6-J)*ITI+IM
      JP=J+JINC
      CI(I)=CI(I)+XNT(JP)*H(M)
      CQ(I)=CQ(I)+XNT(JP+JQ)*H(M)
14    CONTINUE
      RETURN
      END


      SUBROUTINE PULSEQ
      COMMON/FLTPAR/NDWEL,NDELAY,NCYCLE,ICONST(512),LEVELX,
     * LEVELF,LEVELS,LEVELE,LEVELC,CONST(512),XCONST(512),
     * LEVELT
      COMMON/PULTFL/XI(512),XQ(512),JADCLU
      COMMON/RADPAR/PI2,DELT,NPULSE,FDOP,AS,BS,CS,DS
      COMMON/UPDPUL/TIM,PHASE,ICLUT,CI(512),CQ(512)
C     GENERATE NDELAY SAMPLES OF SIGNAL PLUS CLUTTER
      PI=3.141592653
      PID2=PI/2.
      PID2T3=1.5*PI
      DO 11 IDEL=1,NDELAY
      ICLUT=ICLUT+1
      PHASE=PHASE+PI2*FDOP*DELT
      DAS=AS
      CAS=AS
      IF(PHAS .GE.PI2) PHASE=PHASE-PI2
      IF(PHASE.GE.PI) DAS=-AS
      IF(PHASE.GT.PID2.AND.PHASE.LT.PID2T3) CAS=-AS
      DPHASE=PHASE
      IF(DPHASE.GE.PI) DPHASE=DPHASE-PI
      IF(DPHASE.GE.PID2) DPHASE=PI-DPHASE
      XQ(IDEL)=DAS*SIN(DPHASE)
      XI(IDEL)=CAS*COS(DPHASE)
      IF(JADCLU.EQ.0) GO TO 900
      XI(IDEL)=XI(IDEL)+BS+CS*CI(ICLUT)
      XQ(IDEL)=XQ(IDEL)+DS-CS*CQ(ICLUT)
900   CONTINUE
11    CONTINUE
      RETURN
      END


      SUBROUTINE FILTNQ(X,INRMS,NACOFL,NSAT)
      DIMENSION X(1)
      COMMON/FLTPAR/NDWEL,NDELAY,NCYCLE,ICONST(512),LEVELX,
```

Fig. E.1 (Continued)

```
     * LEVELF,LEVELS,LEVELE,LEVELC,CONST(512),XCONST(512),
     * LEVELT
      COMMON/PRINT/IPRINT(15)
    1 FORMAT(/1X,'FILTER INPUT FOR DELAY NO.',I4,' = ',F12.5)
    2 FORMAT(1X,'QUANTIZED INPUT',10X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
    3 FORMAT(1X,'MULTIPLIER OUTPUT',8X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
    4 FORMAT(1X,'TRUNCATED PRODUCT',9X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
    5 FORMAT(1X,'ACCUMULATED VALUE',8X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
    6 FORMAT(1X,'INPUT TO RMS ALGORITHM',3X,'OCTAL = ',O12,
     * 5X,'QUANTIZED = ',F12.5)
    7 FORMAT(1X,10(1H*),'A/D CONVERTER SATURATION',10(1H*))
    8 FORMAT(1X,10(1H*),'OVERFLOW IN ACCUMULATION',10(1H*))
      QX=2./FLOAT(LEVELX)
      LEVELM=LEVELX*(LEVELC/2)
      QT=2./FLOAT(LEVELT)
      QC=2./FLOAT(LEVELC)
      IACC=0
      NACOFL=0
      DO 11 IDELAY=1,NDELAY
      IF(IPRINT(1).NE.0) PRINT 1,IDELAY,X(IDELAY)
C   A/D CONVERSION OF SIGNAL SAMPLE
      CALL IAD(X(IDELAY),IX,QX,LEVELX,ISAT)
      IF(ISAT.NE.0.AND.IPRINT(1).NE.0) PRINT 7
      IF(IPRINT(2).EQ.0) GO TO 12
      IS=IX
      IF(IS.GT.LEVELX/2)IS=IS-LEVELX
      XIX=FLOAT(IS)*QX
      PRINT 2,IX,XIX
   12 CONTINUE
      NSAT=NSAT+ISAT
C     MULTIPLICATION BY FILTER COEFFICIENT
      IMUL=MUL(IX,ICONST(IDELAY),LEVELX,LEVELC)
      IF(IPRINT(3).EQ.0) GO TO 13
      IS=IMUL
      IF(IS.GT.LEVELM/2)IS=IS-LEVELM
      XIMUL=FLOAT(IS)*QC*QX
      PRINT 3,IMUL,XIMUL
   13 CONTINUE
C   TRUNCATION AND EXPANSION
      IMT=ITREX(IMUL,LEVELM,LEVELT,LEVELE)
      IF(IPRINT(4).EQ.0) GO TO 14
      IS=IMT
      IF(IS.GT.LEVELE/2)IS=IS-LEVELE
      XIMT=FLOAT(IS)*QT
```

Fig. E.1 (Continued)

```
      PRINT 4,IMT,XIMT
   14 CONTINUE
C  ACCUMULATION
      CALL ADD(IACC,IMT,IACC,IACOFL,LEVELF)
      IF(IACOFL.NE.0)PRINT 8
      IF(IPRINT(5).EQ.0) GO TO 15
      IS=IACC
      IF(IS.GT.LEVELF/2)IS=IS-LEVELF
      XIACC=FLOAT(IS)*DT
      PRINT 5,IACC,XIACC
   15 CONTINUE
   11 NACOFL=NACOFL+IACOFL
C  FIND THE MAGNITUDE
      INRMS=MAGNF(IACC,LEVELF)
      IF(IPRINT(6).EQ.0) RETURN
      XINRMS=FLOAT(INRMS)*DT
      PRINT 6,INRMS,XINRMS
      RETURN
      END


      FUNCTION IRMSA(INI,INQ)
C  FIND IL AND IS, THE LARGER AND THE SMALLER OF THE INPUTS
      IF(INI.GT.INQ) GO TO 11
      IS=INI
      IL=INQ
      GO TO 12
   11 IL=INI
      IS=INQ
   12 IS08=IS/8
      IS016=IS08/2
      IS8016=IS08+IS016
      IL02=IL/2
      IF(IS.LT.IL02) GO TO 13
C  IF S > 0.5 L
      IL04=IL02/2
      IL804=IL02+IL04
      IS02=IS/2
      IS1116=IS8016+IS02
      IRMSA= IS1116+IL804
      RETURN
C  IF S < 0.5 L
   13 IRMSA=IS8016+IL
      RETURN
      END


      SUBROUTINE COEF(X,IX,G,LEVEL)
```


Fig. E.1 (Continued)

```
      XN=X/Q
      IX=XN
      YN=XN-IX
      IF(ABS(XN).GE.0.5) IX=IX+ISIGN(1,IX)
      IF(X.GE.0) RETURN
      IX=IX+LEVEL
      RETURN
      END


      SUBROUTINE IAD(X,IX,Q,LEVEL,ISAT)
      IX=X/Q
      ISAT=0
      MAX=LEVEL/2-1
      IF(IABS(IX).GE.MAX) GO TO 10
      IF(X.GE.0.) RETURN
      IX=IX+LEVEL-1
      RETURN
   10 IX=ISIGN(MAX,IX)
      ISAT=1
      IF(IX.LT.0) IX=IX+LEVEL
      RETURN
      END


      SUBROUTINE ADD(N1,N2,N3,IOFL,LEVEL)
      MAX=LEVEL/2
      IOFL=0
C     FIND SIGN BITS OF N1 AND N2
      ISN1=N1/MAX
      ISN2=N2/MAX
C     ADD N1 AND N2
      N3=N1+N2
C     FIND THE CARRY BIT
      ICARRY=0
      IF(N3.LT.LEVEL) GO TO 10
      ICARRY=1
C     IGNORE THE CARRY
      N3=N3-LEVEL
   10 CONTINUE
C     IF N1 AND N2 ARE OF DIFFERENT SIGNS, NO OVERFLOW
      IF(ISN1.NE.ISN2) RETURN
C     FIND SIGN BIT OF N3
      ISN3=N3/MAX
C     CHECK FOR OVERFLOW
      IF(ISN3.EQ.ICARRY) RETURN
      IOFL=1
      RETURN
```

Fig. E.1 (Continued)

```
      END


      FUNCTION ITREX(IN,LEVLIN,LEVLTR,LVLOUT)
C   IN HAS MIN BITS (LEVLIN = 2**MIN)
C   TRUNCATE IN TO MTR BITS (LEVLTR = 2**MTR)
      ITREX=IN/(LEVLIN/LEVLTR)
C   EXPAND TO MOUT BITS (LVLOUT = 2**MOUT)
      IF(ITREX .LT.LEVLTR/2) RETURN
      ITREX=ITREX+LVLOUT-LEVLTR
      RETURN
      END


      FUNCTION MUL(N1,N2,LEVEL1,LEVEL2)
C   CONVERT INPUTS TO SIGNED INTEGER
      MAX1=LEVEL1/2
      MAX2=LEVEL2/2
      NS1=N1
      NS2=N2
      IF(N1.GT.MAX1) NS1=N1-LEVEL1
      IF(N2.GT.MAX2) NS2=N2-LEVEL2
      MUL=NS1*NS2
C   CONVERT PRODUCT TO TWO'S COMPLEMENT
      IF(MUL.GE.0) RETURN
      MUL=MUL+(LEVEL1*MAX2)
      RETURN
      END


      FUNCTION MAGNF(IN,LEVEL)
C   MAGNITUDE OF A TWO'S COMPLEMENT NUMBER
      IF(IN.EQ.LEVEL/2) IN=IN+1
      MAGNF=IN
      IF(MAGNF.LT.LEVEL/2) RETURN
      MAGNF=LEVEL-MAGNF
      RETURN
      END


      SUBROUTINE RANDU (IX,IY,RND)
      IY=FLD(5,31,IX*65539)
      RND=IX*0.4656613E-9
      IX=IY
      RETURN
      END
```

Fig. E.1 (Continued)

```
      SUBROUTINE ANIT(START)
      COMMON/NOISE/ARD,RND1,PS
      ARD=START
      RND1=START+0.000001
      RS=47436.
      RETURN
      END


      SUBROUTINE RANDM(X,N,XMEAN,STDEV)
      COMMON/NOISE/ARD,RND1,PS
      DIMENSION X(1)
      DO 3 I=1,N
      ARN=ARD**2+RS**2
      K=ARN/10000000000.
      ARD=(ARN-FLOAT(K)*10000000000.)/10000.
      IF(ARD)2,1,2
    1 ARD=1.
    2 RS=RS+1.
      RND2=ARD*0.000001
      IF(RND1.GE.1.0) PRINT 5,RND1
    5 FORMAT(1X,'RND1=',F30.15/)
      DEVOT=SQRT(-2.*ALOG(RND1))*COS(6.283185*RND2)
      RND1=RND2
      XNR=STDEV*DEVOT
    3 X(I)=XNR+XMEAN
      RETURN
      END
```


Fig. E.1 (Continued)

APPENDIX F

COMPUTER PROGRAM FOR SIMULATION OF THE FLOATING-POINT PROCESSOR
by Brian P. Holt and Bhadrayu J. Trivedi

This appendix contains the FORTRAN computer program to simulate the floating-point processor. In the following discussion the input cards are described in the sequence actually used by the program.

Card-1 -- NRUM
Explanation -- Number of simulation runs to be made.
Format -- I2

Card-2 -- SCAL, SCRDB, XM2, SIGMAF, CLTFAS
Explanation -- SCAL is a scale factor for normalizing signal plus clutter, in order to use all the dynamic range of the A/D converter. SCRDB is the signal-to-clutter power ratio in dB. SCRDB is 0.0 if no clutter is to be used. XM2 is the ratio of dc to ac clutter power. SIGMAF is the standard deviation frequency in Hz of the clutter power spectrum. CLTFAS is the phase of the clutter.
Format -- 5F10.4

Card-3 -- SAMPF
Explanation -- SAMPF is the sampling or pulse repetition frequency in Hz.
Format -- F10.4

Card-4 -- FDOP
Explanation -- Doppler Frequency in Hz.
Format -- F10.4

Card-5 -- NDWEL, NDELAY, NCYCLE
Explanation -- NDWEL is the number of antenna dwells. NDELAY is the number of filter coefficients. NCYCLE is the number of residues.
Format -- 3I3

Card-6 -- (CONST(K), K=1, NDELAY)
Explanation -- CONST(K) is the Kth digital filter coefficient. Note that a maximum of seven coefficients can be specified on a card. For more coefficients, extra cards should be used.
Format -- 7F10.6

Card-7 -- MXM, MCM, MTM, MFM, MFT, MRM, MRT, MIM
Explanation -- These are the processor mantissa bit-lengths as described in Table 5-1, Table 6-1 and Section 6.1.2.
Format -- 8I3

Card-8 -- MCE, MTE, MFE, MRE, MIE
Explanation -- These are the processor exponent bit-lengths as described in Table 5-1, Table 6-1 and Section 6.1.2.
Format -- 5I3

Card-9 -- IDIV, JADCLU, KMOD, JPRNT1, JPRNT2, KTHEOR
Explanation -- IDIV indicates an option for implementing the hardware RMS algorithm.

IDIV = 0 Manipulate L and S by adjusting exponents as far as possible.

= 1 Manipulate L and S by adjusting mantissas only.

JADCLU controls addition of clutter to signal.

JADCLU = 0 No clutter.

= 1 Add clutter.

KMOD, integrator output statistics are printed out at multiples of KMOD antenna dwells.

JPRNT1 is the antenna dwell number at which a detailed debugged printout is to start.

JPRNT2 is the antenna dwell number at which the detailed debugged printout is to stop.

KTHEOR controls the type of filter coefficients to be used for computing theoretical output.

KTHEOR = 0 Unquantized coefficients.

= 1 Quantized coefficients.

FORMAT -- 6I5

```
C     ************************************************************
C     MAIN PROGRAM
C     FLOATING POINT SIMULATION OF MTI RADAR SIGNAL PROCESSOR
C     JANUARY-JUNE 1976 U. OF A. TUSCALOOSA
C     ************************************************************
C     MAGNITUDE IS EXPRESSED IN INTEGERIZED TWO'S COMPLEMENT
C     NOTATION.  EXPONENT IS EXPRESSED IN INTEGERIZED SIGN
C     MAGNITUDE NOTATION.
C     ************************************************************
      INTEGER EMINX,EMINR
      DIMENSION OUTAR(1000),OUTPAR(1000),OUTTHE(1000)
      DIMENSION XN(2000),H(2000)
      COMMON/SATUR/NSAT
      COMMON/SACOUT/XI(512),XQ(512),JADCLU
      COMMON/CLTGEN/PHASE,ICLUT,CI(512),CQ(512)
      COMMON/SAGGEN/PI2,DELT,FDOP,AS,BS,CS,DS
      COMMON/NOISE/ARD,RND1,RS
      COMMON/FILTER/ICFL(512),ICEFL(512),NDELAY,MTM,MXM,
     *MCM,MFM,MCE,MFE,MTE,CONST(512),XICFL(512)
      COMMON/RMSHAL/MRM,LEVELR,EMINR,IDIV,MRE
      COMMON/PRINT/JPRINT
      READ (5,1) NRUN
    1 FORMAT (I2)
      DO2 IRUN=1,NRUN
C     INITIALIZE A/D SATURATION COUNTER
      NSAT=0
C     ************************************************************
C     READ CLUTTER AND SIGNAL PARAMETERS
C     SCAL=SCALE FACTOR FOR SIGNAL AMPLITUDE
C     SCRDB=SIGNAL TO CLUTTER RATIO IN DB
C     NOTE: SCRDB=0. IF CLUTTER IS NOT USED
C     XM2=RATIO OF DC TO AC CLUTTER POWER
C     SIGMAF=STANDARD DEV. OF CLUTTER FREQ. IN HZ.
C     CLTFAS=CLUTTER PHASE
C     ************************************************************
      READ (5,3) SCAL,SCRDB,XM2,SIGMAF,CLTFAS
      PRINT 4, SCAL,SCRDB,XM2,SIGMAF,CLTFAS
    3 FORMAT (7F10.4)
    4 FORMAT(1H1,'SCALE FACTOR = ',16X,F10.4,/
     * 1X,'SIGNAL-TO-CLUTTER RATIO = ',5X,F10.4,' DB',/
     * 1X,'CLUTTER DC-TO-AC POWER RATIO = ',F10.4,/
     * 1X,'SIGMA OF CLUTTER SPECTRUM = ',3X,F10.4,' HZ',/
     * 1X,'CLUTTER PHASE = ',15X,F10.4,' RAD')
C     ************************************************************
C     READ RADAR PARAMETERS
C     SAMPF=SAMPLING (PULSE REP.) FREQ.
C     FDOP=DOPPLER FREQ.
C     ************************************************************
```

Fig. F.1 Listing for the Floating-Point Simulation Program.

```
      READ (5,3) SAMPF
      READ (5,3) FDOP
      PRINT 5, SAMPF
 5    FORMAT(1X,'SAMPLING FREQUENCY = ',9X,F9.2,3X,'HZ')
      PRINT 6, FDOP
 6    FORMAT(1X,'DOPPLER FREQUENCY = ',10X,F9.2,3X,'HZ')
C     ************************************************************
C     READ FILTER PARAMETERS
C     NDWEL=NO. OF ANTENNA DWELLS
C     NDELAY=NO. OF FILTER COEFFICIENTS
C     NCYCLE=NO. OF RESIDUES
C     CONST(K)=K-TH FILTER COEF.
C     ************************************************************
      READ (5,7) NDWEL,NDELAY,NCYCLE
 7    FORMAT (10I3)
      READ (5,8) (CONST(K),K=1,NDELAY)
 8    FORMAT (7F10.6)
      PRINT 9,NDWEL,NDELAY,NCYCLE
 9    FORMAT(1X,'NUMBER OF ANTENNA DWELLS = ',4X,I5,/
     * 1X,'NUMBER OF DELAYS = ',12X,I5,/
     * 1X,'NUMBER OF PULSES INTEGRATED = ',1X,I5//)
C     ************************************************************
C     READ WORD LENGTHS IN BITS
C     MXM=INPUT SIGNAL MANTISSA WITH SIGN
C     MCM=FILTER COEF. MANTISSA WITH SIGN
C     MCE=FILTER COEF. EXPONENT WITHOUT SIGN, .GE.0
C     MTM=TRUNCATED PRODUCT MANTISSA WITH SIGN, .LE.MXM+MCM-1
C     MTE=TRUNCATED PRODUCT EXPONENT WITHOUT SIGN,
C     2**MTE.GE.((2**MCE)+MXM+MCM-2)
C     MFM=FILTER ACCUMULATOR MANTISSA WITH SIGN, .GE.MTM
C     MFE=FILTER ACCUMULATOR EXPONENT WITH SIGN, .GT.MTE
C     MFT=MAG. OF TRUNCATED FILTER OUTPUT MANTISSA
C     WITHOUT SIGN, .LT.MFM
C     MRM=RMS MANTISSA WITHOUT SIGN, .GE.MFT
C     MRE=RMS EXPONENT WITH SIGN, .GE.MFE
C     MRT=TRUNCATED RMS OUTPUT MANTISSA WITHOUT
C     SIGN, .LE.MRM
C     MIM=INTEGRATOR MANTISSA WITHOUT SIGN, .GE.MRT
C     MIE=INTEGRATOR EXPONENT WITH SIGN, .GE.MRE
C     ************************************************************
      READ (5,7) MXM,MCM,MTM,MFM,MFT,MRM,MRT,MIM
      READ (5,7) MCE,MTE,MFE,MRE,MIE
      PRINT 10, MXM,MCM,MTM,MFM,MFT,MRM,MRT,MIM
 10   FORMAT (1X,'PROCESSOR MANTISSA BIT LENGTHS',4X,'MXM',
     * 3X,'MCM',3X,'MTM',3X,'MFM',3X,'MFT',3X,'MRM',3X,
     * 'MRT',3X,'MIM'/31X,916)
      PRINT 11, MCE,MTE,MFE,MRE,MIE
 11   FORMAT (1X,'PROCESSOR EXPONENT BIT LENGTHS',10X,'MCE',
```

Fig. F.1 (Continued)

```
     *3X,'MTE',3X,'MFE',9X,'MRE',9X,'MIE'/37X,3I6,6X,I6,6X,I6)
C     ************************************************************
C     READ OPTIONS
C     IDIV=OPTION USED BY SUBROUTINE RMSHAL
C     JADCLU=1 IMPLIES SIMULATION WITH CLUTTER
C           =0 IMPLIES SIMULATION WITHOUT CLUTTER
C     JPRINT = A PRINT OPTION, CAN BE USED FOR DEBUGGING
C     IF JPRINT = 1 THEN NDWEL = 4 SO THAT THE SIZE OF
C     THE PRINTOUT IS LESS THAN 100 PAGES
C     KTHEOR=0  IMPLIES THEOR. USING IDEAL COEF.
C           =1  IMPLIES QUANTIZED COEF. USED
C     ************************************************************
      READ (5,12) IDIV,JADCLU,KMOD,JPRNT1,JPRNT2,KTHEOR
 12   FORMAT (6I5)
      IF (JADCLU.NE.1) PRINT 17,IDIV
      IF (JADCLU.EQ.1) PRINT 18,IDIV
 17   FORMAT(/,1X,10(1H*),'THIS SIMULATION DOES NOT ',
     *'CONTAIN CLUTTER',10(1H*),'IDIV=',I2/)
 18   FORMAT(/,1X,10(1H*),'THIS SIMULATION CONTAINS CLUTTER',
     * 10(1H*),'IDIV=',I2/)
      IF(KTHEOR.EQ.0) PRINT 2051
      IF(KTHEOR.EQ.1) PRINT 2052
 2051 FORMAT(1X,10(1H*),'THEORETICAL OUTPUT WITH ',
     * 'IDEAL COEFFICIENTS'/)
 2052 FORMAT(1X,10(1H*),'THEORETICAL OUTPUT WITH ',
     * 'QUANTIZED COEFFICIENTS'/)
C     ************************************************************
C     MODEL CLUTTER FILTER AND GENERATE CLUTTER SAMPLES
      SCR=10.**(SCRDB/10.)
      DELT=1./SAMPF
      PI=3.1415926538
      PI2=2.*PI
      NPULSE=NCYCLE*NDELAY
      SIGTAU=1./(PI2*SIGMAF)
      IF (JADCLU.NE.1) GO TO 16
C     ************************************************************
C     SET UP CONSTANTS FOR THE CLUTTER MODEL FILTER
C     IMPULSE RESPONSE
      SIGSS=SIGTAU/DELT
      ITI=SQRT(PI/2.)*SIGSS
      TI=ITI
      SIG1=SIGSS/(SQRT(2.))
      ITMAX=6*ITI
      INCGAU=12+2*((NPULSE-1)/ITI)
C     ************************************************************
C     GENERATE FILTER IMPULSE RESPONSE
      DO112 I=1,ITMAX
      B=I
```

Fig. F.1 (Continued)

```
      H(I)=EXP(-(B-3.*TI)**2/(2.*(SIG1**2)))
  112 CONTINUE
C     ************************************************************
C     GENERATE SCALE FACTORS
C     CHANGE TO AS=SQRT(SCR)*SCAL
      AS=SQRT(SCR)*SCAL
      BS=SQRT(XM2/(1.+XM2))*SCAL*COS(CLTFAS)
      CS=SQRT(1./(2.*(1.+XM2)))*SCAL
      DS=SQRT(XM2/(1.+XM2))*SCAL*SIN(CLTFAS)
      PRINT 101
      PRINT 102,   SCAL,AS,BS,DS,CS,DELT,FDOP
  101 FORMAT(//25X,'MTI INPUT COMPONENTS'
     *//20X,'XI=AS*COS(2*PI*FD*DT)+BS+CS*CI(CLUT)'
     * /20X,'XQ=AS*SIN(2*PI*FD*DT)+DS-CS*CQ(CLUT)')
  102 FORMAT(//1X,'K   = SCALE FACTOR =',12X,F12.5
     */1X,'AS = SIGNAL AMPLITUDE * K =',4X,F12.5
     */1X,'BS = I CHAN. DC CLUTTER * K =',2X,F12.5
     */1X,'DS = Q CHAN. DC CLUTTER * K =',2X,F12.5
     */1X,'CS = I AND Q AC CLUTTER * K =',2X,F12.5
     */1X,'DT = 1/PRF IN SECONDS =',8X,F12.5
     */1X,'FD = INITIAL DOPPLER IN HZ =',3X,F12.5/)
C     ************************************************************
      GO TO 20
   16 AS=SCAL
   20 CONTINUE
C     ************************************************************
C     QUANTIZE THE FILTER COEFFICIENTS AND REPRESENT THE
C     MANTISSA IN INTEGERIZED FORM (TWOS COMP.)
      PRINT 15
   15 FORMAT (11X,'FILTER COEFFICIENTS'//1X,'NUMBER',
     * 4X,'UNQUANTIZED',7X,'QUANTIZED',11X,'ERROR',8X,
     * 'MANTISSA',2X,'EXPONENT')
      DO 13 IDELAY=1,NDELAY
      CALL FLCOEF (CONST(IDELAY),MCM,MCE,ICFL(IDELAY),
     * ICEFL(IDELAY))
C     CONVERT TO SIGN MAG.
      IF (ICFL(IDELAY).GT.2.**(MCM-1))
     * ICQFL=ICFL(IDELAY)-2.**MCM
      IF (ICFL(IDELAY).LE.2.**(MCM-1)) ICQFL=ICFL(IDELAY)
      XICFL(IDELAY)=FLOAT(ICQFL)*(2.**(1-MCM))
     * *(2.**(ICEFL(IDELAY)))
      COEFER=XICFL(IDELAY)-CONST(IDELAY)
      PRINT 14, IDELAY,CONST(IDELAY),XICFL(IDELAY),COEFER,
     * ICFL(IDELAY),ICEFL(IDELAY)
   14 FORMAT (3X,I3,6X,F10.6,6X,F10.6,6X,F10.6,
     * 6X,O10,I10)
   13 CONTINUE
C     ************************************************************
```

Fig. F.1 (Continued)

```
C       CALCULATE THE INFINITE PRECISION (THEORETICAL) OUTPUT
        NDUMP=(NDELAY+1)/2
        NDUM=(NDELAY-1)/2
        IF(KTHEOR.EQ.0) GO TO 2053
        HOFF=XICFL(NDUMP)
        DO 19 IDUM=1,NDUM
 19     HOFF=HOFF+2.*XICFL(NDUMP-IDUM)*COS(PI2*FDOP*IDUM*
       *DELT)
        GO TO 2054
 2053   HOFF=CONST(NDUMP)
        DO 2055 JDUM=1,NDUM
 2055   HOFF=HOFF+2.*CONST(NDUMP-JDUM)*COS(PI2*FDOP*JDUM*
      * DELT)
 2054   THEOUT=AS*HOFF
C       ************************************************************
C       INITIALIZE ERROR SUMS AND OTHER PARAMETERS
        OUERSM=0.
        OPERSM=0.
        OESQSM=0.
        OPESSM=0.
        ODFSSM=0.
        ODIFSM=0.
        OUTMAX=-1000.
        OUTMIN=1000.
        OUPMAX=-1000.
        OUPMIN=1000.
        ODFMAX=-1000.
        ODFMIN=1000.
C       ************************************************************
C       INITIAL VALUE FOR RANDU WHICH GIVES UNIFORMLY
C       DISTRIBUTED RANDOM SAMPLES BETWEEN 0. AND 1.  THEY
C       ARE SCALED TO RANDOM PHASE VALUES BETWEEN 0. AND 2PI.
        IX=1234567
        EMINX=-(2.**(MIE-1)-1)
        DO 21 IDWEL=1,NDWEL
        IF (IDWEL.GE.JPRNT1.AND.IDWEL.LE.JPRNT2) JPRINT=1
        IF (IDWEL.LT.JPRNT1.OR.IDWEL.GT.JPRNT2) JPRINT=0
        IF (JPRINT.EQ.1) PRINT 2001, IDWEL
 2001   FORMAT (/1X,10(1H*),'DWELL NUMBER',I5/)
        CALL RANDU (IX,IY,RND)
        PHASE=(PI2)*RND
C       ************************************************************
        IMHOUT=0
        IMPOUT=0
        IEHOUT=EMINX
        IEPOUT=EMINX
        ERRSUM=0.
        ESQSUM=0.
```

Fig. F.1 (Continued)

```
      ERRMIN=1000.
      ERRMAX=-1000.
      PERSUM=0.
      PESQSM=0.
      PERMIN=1000.
      PERMAX=-1000.
      DIFSUM=0.
      DIFSQS=0.
      DIFMIN=1000.
      DIFMAX=-1000.
C     ************************************************************
      IF (JADCLU.EQ.0) GO TO 115
      IF (IDWEL.NE.1) GO TO 113
C     GENERATE GAUSSIAN SAMPLES
      N=INCGAU
      ARD=63472.0
      RND1=ARD*0.000001
      RS=47436.0
      SIGRAN=1.0
 113  CALL RANDM (XN,N,0.0,SIGRAN)
C     GENERATE NPULSE SAMPLES OF CLUTTER
      DO 111IPULSE=1,NPULSE
      CI(IPULSE)=0.
 111  CQ(IPULSE)=0.
      JQ=INCGAU/2
      DO 114 IPUL=1,NPULSE
      DO 114 J=1,6
      JINC=(IPUL-1)/ITI
      IM=IPUL-JINC*ITI
      M=(6-J)*ITI+IM
      JP=J+JINC
      CI(IPUL)=CI(IPUL)+XN(JP)*H(M)
      CQ(IPUL)=CQ(IPUL)+XN(JP+JQ)*H(M)
 114  CONTINUE
 115  CONTINUE
      ICLUT=0
      DO 22 ICYCLE=1,NCYCLE
      IDUMEX=0
      IF (JPRINT.EQ.1) PRINT 2002, ICYCLE
 2002 FORMAT (/3X,'CYCLE',I4)
C     GENERATE NDELAY SAMPLES OF SIGNAL + CLUTTER FOR EACH
C     OF I AND Q CHANNELS. THEY ARE XI AND XQ.
      CALL SACGEN
C     CALCULATE FILTER OUTPUT FOR I CHANNEL
      IF (JPRINT.EQ.1) PRINT 2003
 2003 FORMAT (3X,'I CHANNEL')
      CALL FLOFLT (XI,JMFLTI,JEFLTI)
C     CALCULATE FILTER OUTPUT FOR Q CHANNEL
```

Fig. F.1 (Continued)

```
      IF (JPRINT.EQ.1) PRINT 2004
 2004 FORMAT (3X,'Q CHANNEL')
      CALL FLOFLT (XQ,JMFLTQ,JEFLTQ)
C     COMPUTE RMS OUTPUT USING HARDWARE ALGORITHM AND CONVERT
C     TO REAL DECIMAL VALUE.
      LEVELF=2.**MFM
      LEVLFT=2.**(MFT+1)
      JMFLTI=ITREX(JMFLTI,LEVELF,LEVLFT,LEVELR)
      CALL JUSTFY (JMFLTI,IDUMEX,MRM,EMINR)
      JMFLTQ=ITREX(JMFLTQ,LEVELF,LEVLFT,LEVELR)
      CALL JUSTFY (JMFLTQ,IDUMEX,MRM,EMINX)
      CALL RMSHAL (JMFLTI,JMFLTQ,JEFLTI,JEFLTQ,JRMS,JERMS)
      HRMS=FLOAT(JRMS)*(2.**(-MRM))*(2.**JERMS)
C     CALCULATE PERFECT RMS OUTPUT (SQRT(I**2+Q**2))
C     CONVERT FILTER OUTPUTS TO REAL DECIMAL NUMBERS
      PFLTI=FLOAT(JMFLTI)*(2.**(-MRM))*(2.**JEFLTI)
      PFLTQ=FLOAT(JMFLTQ)*(2.**(-MRM))*(2.**JEFLTQ)
      PRMS=SQRT(PFLTI**2+PFLTQ**2)
C     CONVERT PERFECT RMS OUTPUT TO FLOATING POINT NOTATION
      IF (PRMS.LT.1.0) GO TO 89
      IEPRMS=0
      R1=PRMS
 88   R1=R1/2.
      IEPRMS=IEPRMS+1
      IF (R1.GE.1.0) GO TO 88
      IMPRMS=R1*(2.**MRM)
      GO TO 90
 89   MRM1=MRM+1
      MRE1=MRE-1
      CALL FLCOEF (PRMS,MRM1,MRE1,IMPRMS,IEPRMS)
 90   CONTINUE
C     ************************************************************
C     CALCULATE PERFECT RMS STATISTICS
C     PEROUT=PERFECT RMS OUTPUT-THEORETICAL OUTPUT
C     PERSUM=SUM OF PEROUT
C     PAVERR=AVERAGE OF PEROUT OVER RESIDUES CALCULATED
C     PESQSM=SUM OF PEROUT SQUARED
C     PAVSQV=AVERAGE OF SQUARES OF PEROUT
C     PAVMSV=ROOT MEAN SQUARE OF PEROUT
C     PSIGER=VARIANCE OF PEROUT
      PEROUT=PRMS-THEOUT
      PERSUM=PERSUM+PEROUT
      PAVERR=PERSUM/ICYCLE
      PESQSM=PESQSM+PEROUT**2
      PAVSQV=PESQSM/ICYCLE
      PAVMSV=SQRT(PAVSQV)
      PSIGER=PAVSQV-PAVERR**2
      IF(PERMAX.LT.PEROUT) PERMAX=PEROUT
```

Fig. F.1 (Continued)

```
      IF(PERMIN.GT.PEROUT) PERMIN=PEROUT
C     CALCULATE HARDWARE RMS STATISTICS
C     ERROUT=HARDWARE RMS OUTPUT-THEORETICAL OUTPUT
C     ERRSUM=SUM OF ERROUT
C     AVGERR=AVERAGE OF ERROUT OVER RESIDUES CALCULATED
C     ESQSUM=SUM OF ERROUT SQUARED
C     AVGSQV=AVERAGE OF SQUARES OF ERROUT
C     AVGMSV=ROOT MEAN SQUARE OF ERROUT
C     SIGERR=VARIANCE OF ERROUT
      ERROUT=HRMS-THEOUT
      ERRSUM=ERRSUM+ERROUT
      AVGERR=ERRSUM/ICYCLE
      ESQSUM=ESQSUM+ERROUT**2
      AVGSQV=ESQSUM/ICYCLE
      AVGMSV=SQRT(AVGSQV)
      SIGERR=AVGSQV-AVGERR**2
      IF(ERRMAX.LT.ERROUT) ERRMAX=ERROUT
      IF(ERRMIN.GT.ERROUT) ERRMIN=ERROUT
C     CALCULATE THE DIFFERENCES BETWEEN HARDWARE AND PERFECT
C     RMS STATISTICS
      ERRDIF=ERROUT-PEROUT
      DIFSUM=DIFSUM+ERRDIF
      AVGDIF=DIFSUM/ICYCLE
      DIFSQS=DIFSQS+ERRDIF**2
      DIFASV=DIFSQS/ICYCLE
      DIFMSV=SQRT(DIFASV)
      VARDIF=DIFASV-AVGDIF**2
      IF(DIFMAX.LT.ERRDIF) DIFMAX=ERRDIF
      IF(DIFMIN.GT.ERRDIF) DIFMIN=ERRDIF
C     ************************************************************
C     INTEGRATION
C     PERFECT RMS OUTPUT INTEGRATION
      LEVLRT=2.**(MRT+1)
      LEVELI=2.**(MIM+1)
      IMPRMS=ITREX(IMPRMS,LEVELR,LEVLRT,LEVELI)
      IF (JPRINT.EQ.1) PRINT 2005,PRMS,IMPRMS,IEPRMS
 2005 FORMAT(/3X,'INTEGRATION'/6X,'PERFECT RMS',
     * 9X,F13.10,4X,O10,3X,I3)
      IF (JPRINT.EQ.1) PRINT 2007, IMPRMS,IEPRMS
 2007 FORMAT (10X,'TRUNCATED OR EXPANDED',
     * 12X,O10,3X,I3)
      CALL JUSTFY (IMPRMS,IEPRMS,MIM,EMINX)
      IF (JPRINT.EQ.1) PRINT 2008, IMPRMS,IDUMEX
 2008 FORMAT (10X,'JUSTIFIED VALUE',18X,O10,3X,I3)
      IF(JPRINT.EQ.1) PRINT 2006, IMPOUT,IEPOUT
 2006 FORMAT(10X,'PREVIOUS INTEGRATOR SUM',
     * 10X,O10,3X,I3)
      CALL ALIGN (IMPOUT,IMPRMS,IEPOUT,IEPRMS,MIM)
```

Fig. F.1 (Continued)

```
      IF (JPRINT.EQ.1) PRINT 2009, IMPRMS,IEPRMS,IMPOUT,
     * IEPOUT
 2009 FORMAT (10X,'ALIGNED VALUES',4X,'RMS OUTPUT',
     * 5X,010,3X,I3/28X,'INTEGRATOR',5X,010,3X,I3)
      CALL ADD(IMPOUT,IMPRMS,IMPOUT,IFLAG,LEVELI)
      IF (JPRINT.EQ.1) PRINT 2010, IMPOUT,IEPOUT
 2010 FORMAT (10X,'SUM',30X,010,3X,I3)
C     CHECK FOR OVERFLOW
      IF (IFLAG.NE.1) GO TO 23
      IMPOUT=IMPOUT/2
      IEPOUT=IEPOUT+1
      GO TO 24
 23   CALL JUSTFY (IMPOUT,IEPOUT,MIM,EMINX)
 24   CONTINUE
      IF (JPRINT.EQ.1) PRINT 2011, IMPOUT,IEPOUT
 2011 FORMAT (10X,'JUSTIFIED VALUE',18X,010,3X,I3)
C     HARDWARE RMS OUTPUT INTEGRATION
      IF(JPRINT.EQ.1) PRINT 2012, HRMS,JRMS,JERMS
 2012 FORMAT (6X,'HARDWARE RMS',8X,F13.10,4X,010,3X,I3)
      JRMS=ITREX(JRMS,LEVELR,LEVLRT,LEVELI)
      IF (JPRINT.EQ.1) PRINT 2007, JRMS,JERMS
      CALL JUSTFY (JRMS,IDUMEX,MIM,EMINX)
      IF (JPRINT.EQ.1) PRINT 2008, JRMS,JERMS
      IF (JPRINT.EQ.1) PRINT 2006,IMHOUT,IEHOUT
      CALL ALIGN (IMHOUT,JRMS,IEHOUT,JERMS,MIM)
      IF (JPRINT.EQ.1) PRINT 2009, JRMS,JERMS,IMHOUT,
     * IEHOUT
      CALL ADD (IMHOUT,JRMS,IMHOUT,IFLAG,LEVELI)
      IF (JPRINT.EQ.1) PRINT 2010, IMHOUT,IEHOUT
      IF (IFLAG.NE.1) GO TO 25
      IMHOUT=IMHOUT/2
      IEHOUT=IEHOUT+1
      GO TO 26
 25   CALL JUSTFY(IMHOUT,IEHOUT,MIM,EMINX)
 26   CONTINUE
      IF (JPRINT.EQ.1) PRINT 2011, IMHOUT,IEHOUT
 22   CONTINUE
C     ************************************************************
C     FORM ARRAYS OF HARDWARE, PERFECT, AND THEORETICAL
C     INTEGRATOR OUTPUTS AS FUNCTIONS OF DWELL NUMBERS.
      POUT=FLOAT(IMPOUT)*(2.**(-MIM))*(2.**IEPOUT)
      OUTPAR(IDWEL)=POUT
      HOUT=FLOAT(IMHOUT)*(2.**(-MIM))*(2.**IEHOUT)
      OUTAR(IDWEL)=HOUT
      OUTTHE(IDWEL)=NCYCLE*THEOUT
      IF (JPRINT.EQ.1) PRINT 888,OUTTHE(IDWEL),OUTPAR(IDWEL),
     * OUTAR(IDWEL)
 888  FORMAT (10X,'THEORETICAL',5X,F13.10/10X,
```

Fig. F.1 (Continued)

```
     * 'PERFECT',9X,F13.10/10X,'HARDWARE',8X,F13.10)
 21   CONTINUE
C     PRINT A/D SATURATION STATISTICS
      PRINT 887, NSAT
 887  FORMAT (//1X,10(1H*),'THE A/D CONVERTER SATURATED',I5,' TIMES'
     * ,10(1H*)//)
C     ************************************************************
C     CALCULATE INTEGRATOR OUTPUT STATISTICS
      PRINT 27
 27   FORMAT(1H1,T46,'---- INTEGRATOR OUTPUT STATISTICS',
     * '----',//)
      PRINT 28
 28   FORMAT(T6,'INFINITE',T28,'ACTUAL RMS HARDWARE',T66,
     *'PERFECT RMS REALIZATION',T107,'RMS DIFFERENCE',/,T6,
     *'PRECISION',/,T2,'NUM ANSWER',T22,'OUTPUT',T35,'ERROUT',
     *T46,'VARIANCE',T61,'OUTPUT',T74,'ERROUT',T85,'VARIANCE',
     *T99,'ERRDIF',T111,'AVGDIF',T123,'VARDIF',/,T2,'---',
     *T6,12(1H*),T19,38(1H*),T58,38(1H*),T97,35(1H*))
C     OUTTHE=THEORETICAL OUTPUT FOR EACH DWELL
C     OUTAR=HARDWARE RMS OUTPUT FOR EACH DWELL
C     OUTERR=ERROR IN HARDWARE RMS OUTPUT FOR EACH DWELL
C     OUTSIG=VARIANCE OF HARDWARE RMS ERROR
C     OUTPAR=PERFECT RMS OUTPUT FOR EACH DWELL
C     OUTPER=ERROR IN PERFECT RMS OUTPUT
C     OUPSIG=VARIANCE OF PERFECT RMS ERROR
C     OUTDIF=HARDWARE RMS ERROR-PERFECT RMS ERROR
C     ODAVER=AVERAGE OUTDIF
C     ODFSIG=VARIANCE OF OUTDIF
      DO 985 IIDWEL=1,NDWEL
      OUTERR=OUTAR(IIDWEL)-OUTTHE(IIDWEL)
      OUTPER=OUTPAR(IIDWEL)-OUTTHE(IIDWEL)
      OUTDIF=OUTERR-OUTPER
      OUERSM=OUERSM+OUTERR
      OPERSM=OPERSM+OUTPER
      ODIFSM=ODIFSM+OUTDIF
      OUAVER=OUERSM/IIDWEL
      OPAVER=OPERSM/IIDWEL
      ODAVER=ODIFSM/IIDWEL
      OESQSM=OESQSM+OUTERR**2
      OPESSM=OPESSM+OUTPER**2
      ODFSSM=ODFSSM+OUTDIF**2
      OUTSIG=(OESQSM/IIDWEL)-OUAVER**2
      OUPSIG=(OPFSSM/IIDWEL)-OPAVER**2
      ODFSIG=(ODFSSM/IIDWEL)-ODAVER**2
      IF(OUTMAX.LT.OUTERR) OUTMAX=OUTERR
      IF(OUTMIN.GT.OUTERR) OUTMIN=OUTERR
      IF(OUPMAX.LT.OUTPER) OUPMAX=OUTPER
      IF(OUPMIN.GT.OUTPER) OUPMIN=OUTPER
```

Fig. F.1 (Continued)

```
      IF(ODFMAX.LT.OUTDIF) ODFMAX=OUTDIF
      IF(ODFMIN.GT.OUTDIF) ODFMIN=OUTDIF
      IF (IIDWEL.GE.JPRNT1.AND.IIDWEL.LE.JPRNT2) JMOD=1
      IF (IIDWEL.LT.JPRNT1.OR.IIDWEL.GT.JPRNT2) JMOD=KMOD
      IF(MOD(IIDWEL,JMOD).NE.0) GO TO 992
      PRINT 984,IIDWEL,OUTTHE(IIDWEL),OUTAR(IIDWEL),OUTERR,
     *OUTSIG,OUTPAR(IIDWEL),OUTPER,OUPSIG,OUTDIF,ODAVER,ODFSIG
 984  FORMAT(1X,I3,7E13.5,3E12.4)
 992  CONTINUE
 985  CONTINUE
C     ************************************************************
C     PRINT STATISTICS OVER ALL DWELLS
C     OUAVER= AVERAGE OUTERR
C     OPAVER=AVERAGE OUTPER
C     OUTMIN=MINIMUM OUTERR
C     OUPMIN=MINIMUM OUTPER
C     ODFMIN=MINIMUM OUTDIF
C     OUTMAX=MAXIMUM OUTERR
C     OUPMAX=MAXIMUM OUTPER
C     ODFMAX=MAXIMUM OUTDIF
      PRINT 983,OUAVER,OPAVER,ODAVER,OUTMIN,OUPMIN,ODFMIN,
     *OUTMAX,OUPMAX,ODFMAX
 983  FORMAT(T2,'---- AVERAGE ERROR',T32,E12.5,T71,E12.5,T97,
     *E11.4,/,T2,'---- MINIMUM ERROR',T32,E12.5,T71,E12.5,T97,
     *E11.4,/,T2,'---- MAXIMUM ERROR',T32,E12.5,T71,E12.5,T97,
     *E11.4)
 2    CONTINUE
      STOP
      END



      SUBROUTINE SACGEN
      COMMON/PRINT/JPRINT
      COMMON/SACGEN/PI2,DELT,FDOP,AS,BS,CS,DS
      COMMON/FILTER/ICFL(512),ICEFL(512),NDELAY,MTM,MXM,
     *MCM,MFM,MCE,MFE,MTE,CONST(512),XICFL(512)
      COMMON/CLTGEN/PHASE,ICLUT,CI(512),CQ(512)
      COMMON/SACOUT/XI(512),XQ(512),JADCLU
C     GENERATE NDELAY SAMPLES OF SIGNAL PLUS CLUTTER
      PI=3.1415926538
      PID2=PI/2.0
      PID2T3=1.5*PI
      DO211 ICEL=1,NDELAY
      ICLUT=ICLUT+1
      PHASE=PHASE+PI2*FDOP*DELT
      DAS=AS
```

Fig. F.1 (Continued)

```
      CAS=AS
      IF(PHASE.GE.PI2) PHASE=PHASE-PI2
      IF(PHASE.GE.PI) DAS=-AS
      IF(PHASE.GT.PID2.AND.PHASE.LT.PID2T3) CAS=-AS
      DPHASE=PHASE
      IF(DPHASE.GE.PI) DPHASE=DPHASE-PI
      IF(DPHASE.GE.PID2) DPHASE=PI-DPHASE
      XQ(IDEL)=DAS*SIN(DPHASE)
      XI(IDEL)=CAS*COS(DPHASE)
      IF(JADCLU.EQ.0) GO TO 900
      XI(IDEL)=XI(IDEL)+BS+CS*CI(ICLUT)
      XQ(IDEL)=XQ(IDEL)+DS-CS*CQ(ICLUT)
  900 CONTINUE
  211 CONTINUE
      RETURN
      END



      SUBROUTINE FLOFLT (X,MAGACC,IACE)
C     ***************************************************************
C     THIS SUBROUTINE ACCEPTS INPUT (SIGNAL + CLUTTER)
C     SAMPLES AND FILTER COEFFICIENTS.  AFTER CONVERTING THE
C     INPUT SAMPLES TO DIGITAL VALUES IT IMPLEMENTS THE
C     FIXED WINDOW MTI FIR FILTER IN FLOATING POINT ARITH.
C
C       IACC=RANGE BIN ACCUMULATOR MANTISSA
C       IACE=EXPONENT OF THE ACCUMULATOR
C
C       MAGACC=MAGNITUDE OF FILTER OUTPUT MANTISSA
C       IACE   = EXPONENT OF FILTER OUTPUT
C     ***************************************************************
      INTEGER EMIN3
      COMMON/SATUR/NSAT
      COMMON/PRINT/JPRINT
      COMMON/FILTER/ICFL(512),ICEFL(512),NDELAY,MTM,MXM,
     *MCM,MFM,MCE,MFE,MTE,CONST(512),XICFL(512)
      DIMENSION X(512),ICEXP(512)
      EMIN3=-(2.**(MFE-1)-1)
      LEVELX=2.**MXM
      LEVELC=2.**MCM
      IACC=0
      IACE=EMIN3
      QX=2.**(-MXM+1)
      DO 5 KDELAY=1,NDELAY
      ICEXP(KDELAY)=ICEFL(KDELAY)
    5 CONTINUE
      DO 10 I=1,NDELAY
```

Fig. F.1 (Continued)

```
C       IDUMEX IS A DUMMY EXPONENT USED FOR CALLING JUSTFY
C       AFTER TRUNCATION OR EXPANSION.  THE ACTUAL EXPONENT
C       NEEDS TO BE SAVED.
        IDUMEX=0
        IF (JPRINT.EQ.1) PRINT 1001, I,X(I)
 1001 FORMAT (/10X,'INPUT SAMPLE ','X(',I2,')=',F13.10)
C       THIS CALLS THE A/D CONVERSION ROUTINE IAD.
        CALL IAD (X(I),IX,QX,LEVELX,ISAT)
        IF (JPRINT.EQ.1) PRINT 1002, IX
 1002 FORMAT (10X,'A/D CONVERTED INPUT',14X,O10,5X,'O')
C       IF ISAT.EQ.1 SATURATION OCCURED.  SATURATION OCCURS IF
C       ABS(SIGNAL+CLUTTER) > 1.
        IF (ISAT.NE.1) GO TO 20
        NSAT=NSAT+1
   20   IMUL=MUL(IX,ICFL(I),LEVELX,LEVELC)
        IF (JPRINT.EQ.1) PRINT 1003, IMUL,ICEXP(I)
 1003 FORMAT (10X,'UNJUSTIFIED PRODUCT',14X,O10,3X,I3)
        K=MXM+MCM-2
        CALL JUSTFY (IMUL,ICEXP(I),K,EMIN3)
        IF (JPRINT.EQ.1) PRINT 1006,IMUL,ICEXP(I)
 1006 FORMAT (10X,'JUSTIFIED PRODUCT',16X,O10,3X,I3)
        LEVELP=2.**(K+1)
        LEVELT=2.**MTM
        LEVELF=2.**MFM
        IMT=ITREX(IMUL,LEVELP,LEVELT,LEVELF)
        IF (JPRINT.EQ.1) PRINT 1007, IMT,ICEXP(I)
 1007 FORMAT(10X,'TRUNCATED OR EXPANDED PRODUCT',4X,O10,3X,I3)
        MFM1=MFM-1
        CALL JUSTFY (IMT,IDUMEX,MFM1,EMIN3)
        IF (JPRINT.EQ.1)  PRINT 1009, IMT,ICEXP(I)
 1009 FORMAT (10X,'JUSTIFIED VALUE',18X,O10,3X,I3)
        IF (JPRINT.EQ.1) PRINT 1010, IACC,IACE
 1010 FORMAT (10X,'PREVIOUS ACCUMULATOR SUM',9X,O10,3X,I3)
        CALL ALIGN (IMT,IACC,ICEXP(I),IACE,MFM1)
        IF(JPRINT.EQ.1) PRINT 1008, IMT,ICEXP(I),IACC,IACE
 1008 FORMAT (10X,'ALIGNED VALUES',5X,'PRODUCT',7X,O10,3X,
     * I3/29X,'ACCUMULATOR',3X,O10,3X,I3)
        CALL ADD (IMT,IACC,IACC,IFLAG,LEVELF)
        IF (JPRINT.EQ.1) PRINT 1004, IACC,IACE
 1004 FORMAT (10X,'SUM',30X,O10,3X,I3)
        IF (IFLAG.NE.1) GO TO 30
        IACC=IACC/2
        IACE=IACE+1
        GO TO 11
   30   CALL JUSTFY (IACC,IACE,MFM1,EMIN3)
   11   IF (JPRINT.EQ.1) PRINT 1011, IACC,IACE
 1011 FORMAT (10X,'JUSTIFIED VALUE',18X,O10,3X,I3)
   10   CONTINUE
```

Fig. F.1 (Continued)

```
      MAGACC=MAGNF(IACC,LEVELF)
C     THE OUTPUT OF MAGNF IS A POSITIVE JUSTIFIED NUMBER
C     SINCE THE INPUT IS A TWO'S COMPLEMENT JUSTIFIED
C     NUMBER.
      IF (JPRINT.EQ.1) PRINT 1005,MAGACC,IACE
 1005 FORMAT (/10X,'FILTER OUTPUT MAGNITUDE',10X,O10,3X,I3)
      RETURN
      END



      SUBROUTINE RMSHAL (IIM,IQM,IIE,IQE,IRMS,IERMS)
C     ***********************************************************
C     THIS ROUTINE TAKES THE I AND Q FILTER OUTPUTS AND
C     CALCULATES THE RMS APPROX. USING L+3/16S IF S/L IS
C     LESS THAN OR EQUAL TO 1/2. OTHERWISE, IT USES 3/4L +
C     11/16S. IIM & IQM = THE I & Q CHANNEL MAGNITUDES,
C     RESPECTIVELY. IIE & IQE ARE THEIR RESPECTIVE EXPONENTS.
C     IRMS & IERMS ARE THE MAGNITUDE & EXPONENT OF THE RESULT.
C     IDIV IS AN OPTION FOR CALCULATING 11/16S, 3/16S, AND
C     3/4L. IF IDIV= 1;
C                          3/16S=(S+S/2)/8          ) ALL
C                          11/16S=((S+S/2)/4+S)/2   ) EXPONENTS
C                          3/4L=(L+L/2)/2           ) UNCHANGED
C           IF IDIV=0;
C                          3/16S=(S+S/2)            ) EXP=EXP-3
C                          11/16S=(S+S/2)/4+S       ) EXP=EXP-1
C                          3/4L=(L+L/2)             ) EXP=EXP-1
C     IDIV IS READ IN BY THE MAIN PROGRAM
      INTEGER EMINR
      COMMON/PRINT/JPRINT
      IF (JPRINT.EQ.1) PRINT 1003, IDIV
 1003 FORMAT (/3X,'HARDWARE RMS ALGORITHM'/10X,'IDIV',I3)
      COMMON/RMSHAL/MRM,LEVELR,EMINR,IDIV,MRE
      EMINR=-(2.**(MRE-1)-1)
      LEVELR=2.**(MRM+1)
      IF (IIE-IQE) 110,120,130
  120 IF (IIM-IQM) 110,110,130
  110 ILM=IQM
      ILE=IQE
      ISM=IIM
      ISE=IIE
      GO TO 140
  130 ILM=IIM
      ILE=IIE
      ISM=IQM
      ISE=IQE
```

Fig. F.1 (Continued)

```
  140 IEDIFF= ILE-ISE
      R1= ILM*0.5*(2.**IEDIFF)
C     THIS TESTS IF (S/L) IS GREATER THAN 0.5 OR NOT
      IF (FLOAT(ISM).LE.R1) GO TO 150
      IF (JPRINT.EQ.1) PRINT 1001
 1001 FORMAT (10X,'(S/L).GT.0.5')
      I32L=ILM+ILM/2
      I118S=(ISM+ISM/2)/4+ISM
      IF (JPRINT.EQ.1) PRINT 1004, I32L,ILE,I118S,ISE
 1004 FORMAT (10X,'(3/2)L',27X,O10,3X,I3/10X,'(11/8)S',
     * 26X,O10,3X,I3)
      IF (IDIV.EQ.1) GO TO 160
      ILE=ILE-1
      ISE=ISE-1
      IF (JPRINT.EQ.1) PRINT 1005, I32L,ILE,I118S,ISE
 1005 FORMAT (10X,'(3/4)L',27X,O10,3X,I3/10X,'(11/16)S',
     * 25X,O10,3X,I3)
      IF (I32L.LT.2.**MRM) GO TO 170
      I32L=I32L/2
      ILE=ILE+1
      IF (JPRINT.EQ.1) PRINT 1006, I32L,ILE
 1006 FORMAT (10X,'OVERFLOW CORRECTION OF (3/4)L',3X,O10,
     * 3X,I3)
  170 IF (I118S.LT.2.**MRM) GO TO 180
      I118S=I118S/2
      ISE=ISE+1
      IF (JPRINT.EQ.1) PRINT 1007, I118S,ISE
 1007 FORMAT (10X,'OVERFLOW CORRECTION OF (11/8)S',1X,
     * O10,3X,I3)
  180 I34L=I32L
      I1116S=I118S
      GO TO 1000
  160 I34L=I32L/2
      IF (JPRINT.EQ.1) PRINT 1005, I34L,ILE,I1116S,ISE
      I1116S=I118S/2
      CALL JUSTFY (I34L,ILE,MRM,EMINR)
      CALL JUSTFY (I1116S,ISE,MRM,EMINR)
      IF (JPRINT.EQ.1) PRINT 1011, I34L,ILE,I1116S,ISE
 1011 FORMAT (10X,'JUSTIFIED VALUES',2X,'(3/4)L',9X,O10,
     * 3X,I3/28X,'(11/16)S',7X,O10,3X,I3)
 1000 CALL ALIGN (I34L,I1116S,ILE,ISE,MRM)
      IF (JPRINT.EQ.1) PRINT 1008, I34L,ILE,I1116S,ISE
 1008 FORMAT (10X,'ALIGNED VALUES',4X,'(3/4)L',9X,O10,
     * 3X,I3/28X,'(11/16)S',7X,O10,3X,I3)
      CALL ADD (I34L,I1116S,IRMS,IFLAG,LEVELR)
      GO TO 2000
  150 IF (JPRINT.EQ.1) PRINT 1002
 1002 FORMAT (10X,'(S/L).LE.0.5')
```

Fig. F.1 (Continued)

```
      I32S=ISM+ISM/2
      IF (JPRINT.EQ.1) PRINT 1012, ILM,ILE,I32S,ISE
 1012 FORMAT (10X,'L',32X,O10,3X,I3/10X,'(3/2)S',27X,O10,
     * 3X,I3)
      IF (IDIV.EQ.1) GO TO 190
      ISE=ISE-3
      IF (JPRINT.EQ.1) PRINT 1013, I32S,ISE
 1013 FORMAT (10X,'(3/16)S',26X,O10,3X,I3)
      IF (I32S.LT.2.**MRM) GO TO 210
      I32S=I32S/2
      ISE=ISE+1
      I316S=I32S
      IF (JPRINT.EQ.1) PRINT 1014, I316S,ISE
 1014 FORMAT (10X,'OVERFLOW CORRECTION OF (3/16)S',3X,
     * O10,3X,I3)
      GO TO 210
  190 I316S=I32S/8
      IF (JPRINT.EQ.1) PRINT 1013, I316S,ISE
      CALL JUSTFY (I316S,ISE,MRM,EMINR)
      IF (JPRINT.EQ.1) PRINT 1016, I316S,ISE
 1016 FORMAT (10X,'JUSTFIED VALUE',2X,'(3/16)S',8X,O10,
     * 3X,I3)
  210 CALL ALIGN (I316S,ILM,ISE,ILE,MRM)
      IF (JPRINT.EQ.1) PRINT 1015, ILM,ILE,I316S,ISE
 1015 FORMAT (10X,'ALIGNED VALUES',5X,'L',13X,O10,3X,
     * I3/29X,'(3/16)S',7X,O10,3X,I3)
      CALL ADD (I316S,ILM,IRMS,IFLAG,LEVELR)
 2000 IERMS=ILE
      IF (JPRINT.EQ.1) PRINT 1009, IRMS,IERMS
 1009 FORMAT (10X,'SUM',30X,O10,3X,I3)
      IF (IFLAG.NE.1) GO TO 220
      IRMS=IRMS/2
      IERMS=IERMS+1
      IF (JPRINT.EQ.1) PRINT 1010, IRMS,IERMS
 1010 FORMAT (10X,'OVERFLOW CORRECTION OF SUM',7X,
     * O10,3X,I3)
  220 RETURN
      END



      SUBROUTINE ALIGN (IM1,IM2,IE1,IE2,K)
C     ************************************************************
C     THIS ROUTINE TAKES TWO FLOATING POINT NUMBERS AND ALIGNS
C     THEM SO THEY CAN BE ADDED. IM1 & IE1 ARE THE MAGNITUDE
C     AND EXPONENT, RESPECTIVELY, OF THE FIRST NO. IM2 & IE2
C     ARE THE MAGNITUDE AND EXPONENT, RESPECTIVELY, OF THE
C     SECOND NO. K IS THE NO. OF BITS USED TO REPRESENT IM1
```

Fig. F.1 (Continued)

```
C     & IM2 IN TWOS COMP. FORM EXCLUDING THE SIGN BITS.
C     THE MAGNITUDE PORTION OF THE SHIFTED NO. IS TRUNCATED
C     AFTER EACH SHIFT.
C     ************************************************************
      COMMON/PRINT/JPRINT
      INTEGER TEST
      TEST=2**K
      IED=IE1-IE2
      IAED=IABS(IED)
      IDIF=2**IAED
      IF (IED) 10,30,20
C     FROM HERE TO 12 IF EXPONENT 2 > EXPONENT 1.
   10 IF (IM1.GE.TEST) GO TO 11
      IM1=IM1/IDIF
      GO TO 12
   11 IM1=IM1/2
      IED=IED+1
      IM1=IM1+TEST
      IF (IED.LT.0) GO TO 11
   12 IE1=IE2
      RETURN
C     FROM HERE TO 22 IF EXPONENT 1 > EXPONENT 2.
   20 IF (IM2.GE.TEST) GO TO 21
      IM2=IM2/IDIF
      GO TO 22
   21 IM2=IM2/2
      IED=IED-1
      IM2=IM2+TEST
      IF (IED.GT.0) GO TO 21
   22 IE2=IE1
   30 RETURN
      END



      SUBROUTINE JUSTFY (UNJ,EUNJ,K,EMIN)
C     ************************************************************
C     THIS ROUTINE TAKES A FLOATING POINT NUMBER THAT IS IN
C     TWOS COMP. FORM, JUSTIFIES IT AND ADJUSTS THE EXPONENT.
C     UNJ= UNJUSTIFIED MANTISSA ON ENTRY AND JUSTIFIED
C     MANTISSA ON RETURN, EUNJ= UNADJUSTED EXPONENT ON ENTRY
C     AND THE ADJUSTED EXPONENT ON RETURN.
C     K= THE NUMBER OF BITS IN UNJ EXCLUDING SIGN,
C     EMIN= THE SMALLEST POSSIBLE EXPONENT VALUE
C     ************************************************************
      INTEGER UNJ, EUNJ, K, EMIN, TEST
      TEST= 2.**(K-1)
C     TEST FOR POSITIVE OR NEGATIVE NO.,IF POS. GO TO 140
```

Fig. F.1 (Continued)

```
      IF (UNJ.LT.TEST*2) GO TO 140
      I=0
C     STRIP OFF SIGN BIT FROM NEGATIVE NUMBER
      UNJ=UNJ-(TEST*2)
C     TEST FOR 0 IN MSB; IF SO, DONE
   70 IF (UNJ.LT.TEST) GO TO 110
C     SHIFT NUMBER LEFT AND INCR COUNT
      UNJ=(UNJ-TEST)*2
      I=I+1
      GO TO 70
C     REPLACE SIGN BIT
  110 UNJ=UNJ+(TEST*2)
C     ADJUST EXPONENT
      EUNJ=EUNJ-I
      RETURN
C     TEST FOR 0
  140 IF (UNJ.EQ.0) GO TO 220
      I=0
C     TEST FOR 1 IN MSB; IF SO, DONE
  160 IF (UNJ.GE.TEST) GO TO 200
C     SHIFT NUMBER LEFT AND INCREASE COUNT
      UNJ=UNJ*2
      I=I+1
      GO TO 160
C     ADJUST EXPONENT
  200 EUNJ=EUNJ-I
      GO TO 230
C     SET EXPONENT TO MINIMUM VALUE
  220 EUNJ=EMIN
  230 CONTINUE
      RETURN
      END



      FUNCTION ITREX(IN,LEVLIN,LEVLTR,LVLOUT)
C     NOTE: THE BITLENGTH USED FOR COMPUTING THE LEVEL
C           INCLUDES THE SIGN BIT.
C     IN HAS MIN BITS (LEVLIN = 2**MIN)
C     TRUNCATE IN TO MTR BITS (LEVLTR = 2**MTR)
      ITREX=IN/(LEVLIN/LEVLTR)
C     EXPAND TO MOUT BITS (LVLOUT = 2**MOUT)
      IF(ITREX .LT.LEVLTR/2) RETURN
      ITREX=ITREX+LVLOUT-LEVLTR
      RETURN
      END
```

Fig. F.1 (Continued)

```
      SUBROUTINE FLCOEF (X,MC,MCE,ICFL,ICEFL)
C     ***********************************************************
C     THIS ROUTINE TAKES A DECIMAL FRACTION AND CONVERTS IT
C     INTO A TWOS COMP. FLOATING POINT FORM.
C     X= THE DECIMAL FRACTION, MC= THE NUMBER OF BITS TO BE
C     USED IN THE MANTISSA OF THE FLOATING POINT NO. (WITH
C     SIGN), MCE= THE NO. OF BITS IN THE EXPONENT (WITHOUT
C     SIGN), ICFL= THE MANTISSA,ICEFL= THE EXPONENT.
C     NOTE AGAIN: ABS(X).LT.1.0
C     ***********************************************************
      INTEGER EMIN
      L=0
      JMAX=(2**(MC-1))-1
      JMIN=-(2**(MC-1))
      Q=2.**(-MC+1)
      IMAX=(2.**MCE)-1
      I=0
C     CHECK IF FIXED OR FLOATING COEFFICIENTS ARE TO BE
C     USED.  FOR FIXED GO TO 11
      IF (IMAX.EQ.0) GO TO 11
      XMAG=0.5
C     NEXT SIX STATEMENTS TO DETERMINE FLOATING POINT
C     QUANTIZATION INTERVAL
 5    IF(ABS(X).GE.XMAG) GO TO 10
      XMAG=XMAG/2.0
      I=I+1
      IF(I.EQ.IMAX) GO TO 10
      GO TO 5
 10   Q=Q/(2.**I)
 11   XN=X/Q
      IX=XN
      XN=XN-IX
      IF (IX.EQ.JMAX.OR.IX.EQ.JMIN) L=1
      IF (ABS(XN).LT.0.5) GO TO 23
      IX=IX+ISIGN(1,IX)
C     IF ROUNDING CAUSED OVERFLOW, DIVIDE THE MANTISSA BY
C     TWO AND ADJUST THE EXPONENT.
      IF (L.NE.1) GO TO 23
      IX=IX/2
      I=I-1
 23   IF(X.GE.0.) GO TO 20
C     CONVERT NEGATIVE MANTISSA TO TWO'S COMP FORM
      IX=IX+(2.**MC)
 20   EMIN=-IMAX
      K=MC-1
      IEX=-I
      IF (IMAX.EQ.0) GO TO 21
```

Fig. F.1 (Continued)

```
      CALL JUSTFY (IX,IEX,K,EMIN)
21    ICFL=IX
      ICEFL=IEX
      RETURN
      END



      SUBROUTINE RANDU (IX,IY,RND)
      IY=FLD(5,31,IX*65539)
      RND=IX*0.4656413E-9
      IX=IY
      RETURN
      END



      SUBROUTINE RANDM(X,N,XMEAN,STDEV)
      COMMON/NOISE/ARD,RND1,RS
      DIMENSION X(1)
      DO 3 I=1,N
      ARN=ARD**2+RS**2
      K=ARN/1000000000.
      ARD=(ARN-FLOAT(K)*1000000000.)/1000.
      IF(ARD)2,1,2
1     ARD=1.
2     RS=RS+1.
      RND2=ARD*0.000001
      DEVOT=SQRT(-2.*ALOG(RND1))*COS(6.283185*RND2)
      RND1=RND2
      XNR=STDEV*DEVOT
3     X(I)=XNR+XMEAN
      RETURN
      END



      FUNCTION MUL(N1,N2,LEVEL1,LEVEL2)
C     NOTE: THE BITLENGTH USED FOR COMPUTING THE LEVEL
C           INCLUDES THE SIGN BIT.
C     CONVERT INPUTS TO SIGNED INTEGER
      MAX1=LEVEL1/2
      MAX2=LEVEL2/2
      NS1=N1
      NS2=N2
      IF(N1.GT.MAX1) NS1=N1-LEVEL1
      IF(N2.GT.MAX2) NS2=N2-LEVEL2
      MUL=NS1*NS2
```

Fig. F.1 (Continued)

```
C       CONVERT PRODUCT TO TWO:S COMPLEMENT
        IF(MUL.GE.0) RETURN
        MUL=MUL+(LEVEL1*MAX2)
        RETURN
        END



        FUNCTION MAGNF(IN,LEVEL)
C       NOTE: THE BITLENGTH USED FOR COMPUTING THE LEVEL
C             INCLUDES THE SIGN BIT.
C       MAGNITUDE OF A TWO:S COMPLEMENT NUMBER
        IF(IN.EQ.LEVEL/2) IN=IN+1
        MAGNF=IN
        IF(MAGNF.LT.LEVEL/2) RETURN
        MAGNF=LEVEL-MAGNF
        RETURN
        END



        SUBROUTINE IAD(X,IX,Q,LEVEL,ISAT)
C       NOTE: THE BITLENGTH USED FOR COMPUTING THE LEVEL
C             INCLUDES THE SIGN BIT.
        IX=X/Q
        ISAT=0
        MAX=LEVEL/2-1
        IF(IABS(IX).GE.MAX) GO TO 10
        IF(X.GE.0.) RETURN
        IX=IX+LEVEL-1
        RETURN
   10   IX=ISIGN(MAX,IX)
        ISAT=1
        IF(IX.LT.0) IX=IX+LEVEL
        RETURN
        END
        SUBROUTINE ADD(N1,N2,N3,IOFL,LEVEL)
C       NOTE: THE BITLENGTH USED FOR COMPUTING THE LEVEL
C             INCLUDES THE SIGN BIT.
        MAX=LEVEL/2
        IOFL=0
C       FIND SIGN BITS OF N1 AND N2
        ISN1=N1/MAX
        ISN2=N2/MAX
C       ADD N1 AND N2
        N3=N1+N2
C       FIND THE CARRY BIT
        ICARRY=0
```

Fig. F.1 (Continued)

```
      IF(N3.LT.LEVEL) GO TO 10
      ICARRY=1
C     IGNORE THE CARRY
      N3=N3-LEVEL
   10 CONTINUE
C     IF N1 AND N2 ARE OF DIFFERENT SIGNS, NO OVERFLOW
      IF(ISN1.NE.ISN2) RETURN
C     FIND SIGN BIT OF N3
      ISN3=N3/MAX
C     CHECK FOR OVERFLOW
      IF(ISN3.EQ.ICARRY) RETURN
      IOFL=1
C     ADD LEVEL BACK IF NUMBER IS NEGATIVE AND AN OVERFLOW
C     OCCURED.  IT CAN THEN BE CORRECTED FOR OVERFLOW BY
C     THE CALLING PROGRAM.
      IF (ISN1.EQ.1) N3=N3+LEVEL
      RETURN
      END
```

Fig. F.1 (Continued)

GRADUATE STUDENTS RECEIVING REMUNERATION

Name	Appointment Period and Description	Degree Status
1. Brian P. Holt	1 January to 15 May 1976, as Half-time Research Assistant, 16 May to 30 June 1976 as 75% Research Assistant	Expects to complete Master of Science degree in May 1977 with thesis in digital signal processing area
2. Bhadrayu J. Trivedi	1 January to 15 May 1976 as Quarter-time Research Associate, 16 May to 30 June as Full-time Research Associate	Beginning work on Ph.D. program

LIST OF REFERENCES

1. L. B. Owen, N. E. Lawrence and D. W. Burlage, "A Programmable Signal Processor for Doppler Filtering in an Experimental-Array Radar," Proceeding of 1976 IEEE SOUTHEASTERN Conference, pp. 95-97.

2. B. Gold and C. M. Rader, Digital Processing of Signals. McGraw-Hill, 1969.

3. A. V. Oppenheim and R. W. Schafer, Digital Signal Processing. Prentice-Hall, 1975.

4. L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing. Prentice-Hall, 1975.

5. A. V. Oppenheim and C. J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform," Preprint JA-4005, May 1972. Also in Proc. of IEEE, August 1972, pp. 957-976.

6. I. W. Sandberg, "Floating-Point-Roundoff Accumulation in Digital-Filter Realizations," Bell System Technical Journal, Oct. 1967, pp. 1775-1791.

7. B. Liu and T. Kaneko, "Error Analysis of Digital Filters Realized with Floating-Point Arithmetic," Proc. of IEEE, October 1969, pp. 1735-1747, and corrections in March 1970, p. 376.

8. J. D. Moore, "Quantization Noise in a Radar MTI Digital Signal Processor," Report submitted on U.S. Army Laboratory Research Cooperative Program, Task number 75-109, August 1975.

9. W. M. Hall and H. R. Ward, "Signal-to-Noise Loss in Moving Target Indicators," Proc. of IEEE, February 1968, pp. 233-234.

10. J. D. Moore, et al., "A Study of Digital Filter Characteristics in Moving Target Indicators of Radar Systems," Final Report to U.S. Army Research Office on Grant DAHC04-75-G-0039, June 1975.

11. H. Lorenzetti, "Error Properties of Complex Number Magnitude Approximations Using a Linear Combination of Components," Raytheon Memo No. RS-74-61 on Contract DA-AH01-72-C0106, October 25, 1974.

12. A. E. Filip, "A Baker's Dozen Magnitude Approximations and Their Detection Statistics," Correspondence in IEEE Trans. AES, January 1976, pp. 86-89.

13. J. M. Wozencraft and I. M. Jacobs, Principles of Communication Engineering, John Wiley & Sons, 1967.

14. R. J. Polge, et al., "Quantization Effects in Digital MTI Filters," Final Technical Report, Volume II, UAH Research Report No. 179, U.S. Army MICOM Report No. RE-CR-76-2, August 1975.

15. J. S. Bendat and A. G. Piersol, Measurement and Analysis of Random Data, John Wiley & Sons, 1966, Chap. 4, p. 124.

16. D. H. Shenigold, Editor, Analog-Digital Conversion Handbook, Analog Devices, Inc., 1972, p. II-8.

17. V. Thomas Rhyne, Fundamentals of Digital Systems Design, Prentice-Hall, Inc., 1973.

18. Thomas C. Bartee, Digital Computer Fundamentals, 2nd Edition, McGraw-Hill, 1966.

19. A. V. Oppenheim, "Realization of Digital Filters Using Block-Floating-Point Arithmetic," IEEE Trans. Audio and Electroacoustics, June 1970, pp. 130-136.