Roundoff error sequences in fixed-point arithmetic digital processors

Andrew J. McWilliam

ROUNDOFF ERROR SEQUENCES IN FIXED-POINT ARITHMETIC DIGITAL PROCESSORS

ANDREW J. McWILLIAM

B.Sc. (Dunelm).

Department of Applied Physics and Electronics

University of Durham.

July 1976

A thesis submitted for the degree of

Doctor of Philosophy of the University of Durham.

ACKNOWLEDGEMENT.

Various people and institutions merit thanks for assistance proffered during my research and the production of this thesis.

Dr. B. J. Stanier supervised this project in a way which allowed me substantial freedom in determining the course of the investigation, whilst always being ready to guide and advise. Professor D. A. Wright made available to me the facilities of the Department of Applied Physics and Electronics; I wish him well in his retirement.

My thanks go to the Science Research Council without whose financial support this study would have been impossible.

Work on this thesis was facilitated by the generosity of Rev. A. Casurella Jnr., who lent me his typewriter, and by the practical help and constant encouragement afforded by my wife., Ann.

ABSTRACT.

Three assumptions are normally made about roundoff error sequences in fixed-point arithmetic digital processors: all values of error, within the limits set by the roundoff process, have an equal probability of occurrence; they are random; and they are not correlated with any other signal or error sequence. The validity of these assumptions for digital filter realisations employing a signal wordlength in the range of 4 to 3 bits is examined. The investigation begins by using a fast Fourier transform algorithm to perform spectral analysis on roundoff error sequences when the test signal is a quantised sinusoid. It continues by comparing the variance of the error sequence measured at the output of various filter realisations with that theoretically predicted using the normal assumptions; this is done using random test sequences. It is shown that under certain conditions, for the signal wordlengths under consideration, each of the three assumptions can become invalid. In conclusion this work reports attempts to develop more accurate predictive noise models which only incorporate assumptions of greater validity.

TABLE OF CONTENTS.

CHAPTER 1 - INTRODUCTION.

The processing of electrical signals is a key activity in almost every sphere of contemporary scientific and technological enterprise. Indeed, some of the most important fields of investigation rely on signal processing for their very existence. Speech communications and synthesis, radar, sonar, pattern recognition, forensic science, physiology, and seismology are all activities which could not continue in their present form without processing information borne by electrical signals. Herein lies one of the primary reasons for the great importance of electronics.

Before about 1960 all signal processing was performed using the circuit techniques of analogue electronics. However, the variety of tasks which could be executed in this manner was severely restricted by such factors as the limited range of practical component values, and the lack of any convenient means of delaying or storing signals. It is not surprising, therefore, that when digital electronic hardware, particularly the general purpose computer, became readily available there was great interest in the application of digital techniques to signal processing. Many aspects of signal processing theory which had hitherto been of purely academic interest came to be of great practical importance as the application of the digital computer became feasible. The development of digital hardware has progressed rapidly over the last fifteen years so that the processing resources available have increased tremendously. This has continually prompted and motivated the development of signal processing theory in a constant attempt to take full advantage of the computing power available at any time. Hence the practical capability of signal processing has increased very greatly and this has brought a corresponding growth in its application and importance.

Digital signal processing can be categorised into two main

activities, spectrum analysis and filtering. Both fundamentally involve the production of a sequence of numbers by some arithmetic operation on what may be termed an input sequence of numbers. It is usual to think of the input sequence as representing a quantity which is a function of time. If this is the case then spectrum analysis results in the production of a number sequence representing a function of frequency; this function is the frequency spectrum of the time-domain input sequence. Spectral analysis, therefore, involves a transformation from the time-domain to the frequency-domain. This transformation is reversible so if the input sequence to the processor is regarded as being a frequency function then the output is the time-varying sequence with the given frequency spectrum. Filtering does not involve a transformation between domains, but is a convolution process. If the input sequence is a function of time then so too is the output sequence. The two sequences have different frequency spectra, the characteristics of this modification being determined by the properties of the filter. Filtering can be further subdivided into linear and non-linear processes. Of the three signal processing categories mentioned only linear filtering is considered in the ensuing work. It is hoped, however, that the results and conclusions presented will prove to be of more general significance.

The majority of the research into digital signal processing has been geared to the exploitation of the large main-frame computer. The use of such a machine is often not feasible, however, especially when dedicated hardware is required as in real-time signal processing. The minicomputer is often employed in this kind of application, but its use is restricted by the expense of implementing such a system. It is necessary, therefore, to consider means by which digital signal processing may be performed at greatly reduced cost. Two options available to the designer are to construct a hard-wired purpose-built digital system, or to program a microprocessor to perform the task. The

former approach has the advantage that the design can be optimised to perform one function alone with the result that a relatively high processing speed may be achieved. The microprocessor, on the other hand, offers the flexibility to modify the process easily and the versatility to perform many different tasks just by loading an appropriate program. But the addition of these features must, in general, result in some loss of processing speed. Whichever approach is used the quest for high processing speed and low cost requires three system parameters to be minimised: the complexity of the arithmetic unit; the quantity of data which has to be stored; and finally the number of binary digits (bits) used to represent data items, that is, the wordlength. An increase in any of these three must cause either a degradation in processing speed or a rise in price.

Linear digital filtering is achieved by circuits consisting of two fundamental elements. The first is the time delay, and the second is the constant multiplier. Time delays can easily be practically implemented on a digital processor and are of no great interest. The other process is the multiplication of the numbers of which the signal sequences consist by constant, usually non-integral, coefficients which define the ideal response of the filter. In general, no digital processor can perform such multiplications to infinite precision, but can only return an approximation to the accurate result. If a large main-frame processor is employed the degree of such approximation need be negligible so the practical filter response is virtually ideal. On the other hand, if a processor with a relatively short data wordlength is being used, for reasons of cost or speed, significant errors are created in performing the signal by coefficient multiplications. Such errors, defined as the difference between the accurate result of a multiplication and the approximation returned by the processor, are termed roundoff errors. The generation of these roundoff errors causes the practical behaviour of a

filter to deviate from the ideal response. As the data wordlength employed is shortened this discrepancy between the ideal and practical behaviour must increase.

Because of the speed and cost considerations already mentioned, there is often great purpose in determining the minimum data wordlength required to allow the performance of a given signal processor to come within the limits of the design specification. For this reason there has been considerable interest in producing theoretical models which permit a prediction of the effect of roundoff errors on the behaviour of a given filter. However, it would appear firstly, that much of the work has been of a predominately theoretical nature, and secondly, that the very short data wordlengths encountered, for example, when using a microprocessor have not received particular attention. The object of this project is to investigate, both experimentally and theoretically, some of the effects of roundoff errors on filters realised using a very short data wordlength.

CHAPTER 2 - DISCRETE-TIME LINEAR FILTERING.

2.1 Discrete-time signals.

A discrete-time signal is one which is formed by repeatedly sampling a continuously varying signal at instants in time. If any of the frequency information contained in the continuously varying signal is to be retained in the resulting discrete-time signal, the duration between consecutive samples must be known. The simplest case is when the duration is constant and each sample is separated from the next by the time T. This system of sampling, the only one in common usage, is depicted in Figure 2.1. It can be seen that the continuous signal is converted into a sequence of numbers which represent the amplitude of a sample. No information is retained concerning the behaviour of the continuous signal in between samples, which leads directly to a phenomenon which pervades the whole subject of discrete-time signal processing: that of aliasing.

2.1.1 Aliasing.

Consider Figure 2.3 in which the sampling of a continuous sinusoid is shown[1]. Both the sinusoids depicted, and infinitely many more of higher frequency, would yield the same sequence of numbers when sampled at constant intervals of T. Hence it is impossible to decide which sinusoid the sequence of samples is meant to represent. This effect in which one frequency 'impersonates' others is called aliasing. At this stage it may be stated without proof that the infinite series of frequencies which are indistinguishable from one another is given by

$$\omega_n = \frac{2\pi n}{T} \pm \omega \qquad \text{for any integer } n. \qquad (2.1.1)$$

(Further extension and application of Figure 2.3 would substantiate this

Figure 2.1 Sampling a continuously varying signal

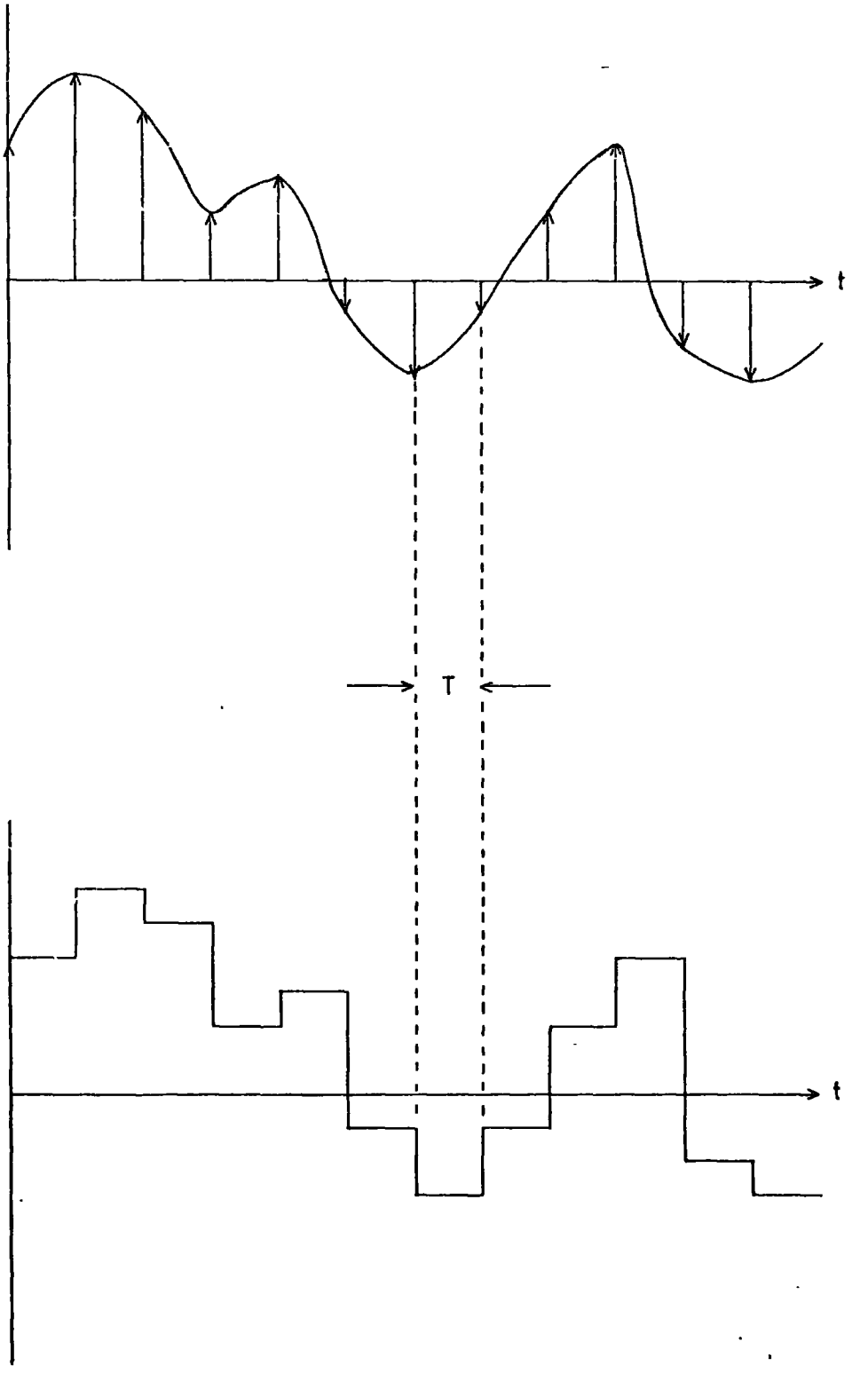


Figure 2.2 A continuous-time signal reconstructed.

Figure 2.3 An example of aliasing.

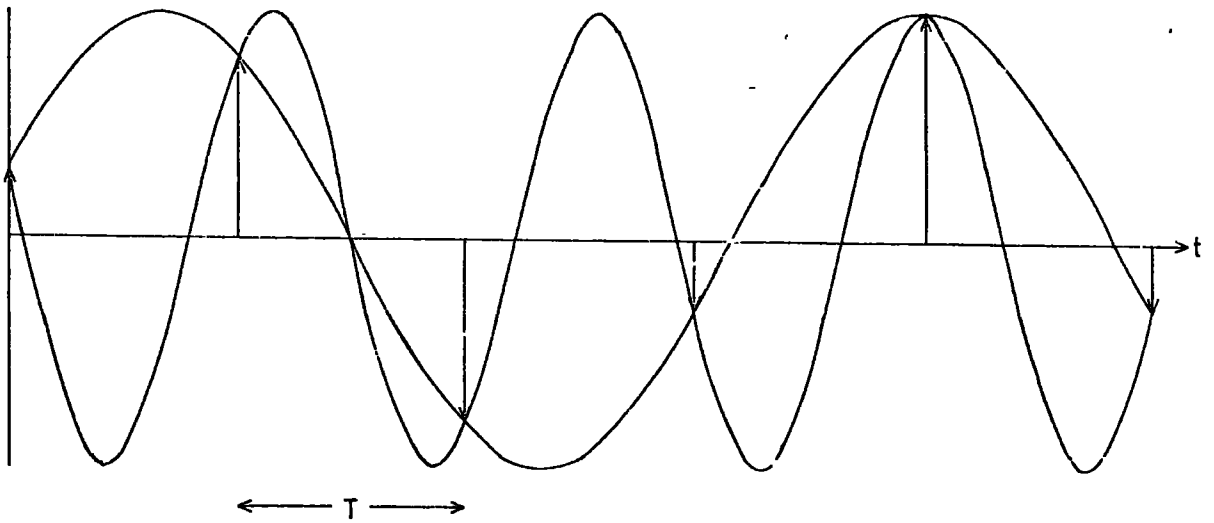
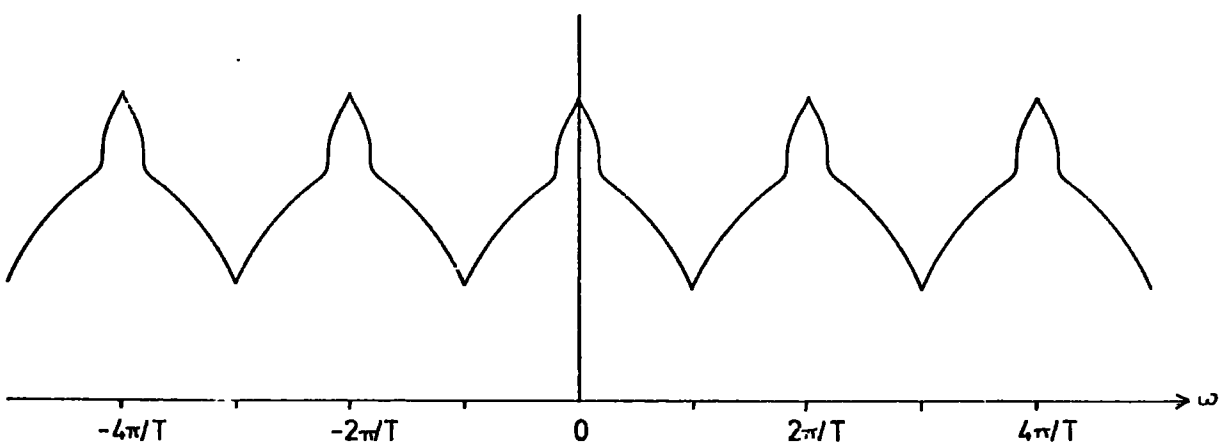


Figure 2.4 The periodic frequency spectrum of a discrete-time signal.



statement). It may be inferred from equation (2.1.1) that the frequency spectrum of a general discrete-time signal will have the form shown in Figure 2.4. That is, the frequency spectrum is periodic with a period equal to a radian frequency of $2\pi/T$, the frequency of the sampling process, and is symmetrical about $\omega = 0$.

A discrete-time sequence has no unique representation, but has an infinite variety of accurate interpretations. This is a concept which is both unnecessarily sophisticated and difficult to work with. It is usual, therefore, to restrict the representation of discrete-time sequences to one bandwidth of π/T, which is half the frequency of the sampling process. In principle this segment may be chosen anywhere in the frequency spectrum, but it is most usual to consider the frequency interval 0 to π/T radians per second. This is because, at the stage of reconversion from a discrete-time sequence to a continuous signal, there is no information available on the required behaviour of the continuous signal in between samples, so that the signal is held constant for the duration of the sampling period. Hence the reconstructed continuous-time signal will have the general shape shown in Figure 2.2. A lowpass analogue filter may be used to remove the frequencies above π/T contained in this waveform, yielding the original continuous waveform as in Figure 2.1.

The important conclusion is that only continuous-time signals with all frequency components within the range 0 to π/T may be sampled and reconstructed. The radian frequency π/T, the half-sampling or Nyquist frequency, is therefore of the utmost significance in the field of discrete-time signal processing.

2.2 Discrete-time linear filtering.

Having made the preliminary, general remarks above which appertain to the whole field of discrete-time signal processing, the theory of

filtering discrete-time signals may now be developed. It is important to understand the breadth of the term filtering in this context and hence the relevance of studying the process. The term filtering is applied to any linear process whereby a single discrete signal, x(nT), is processed to give a single discrete output signal, y(nT). The signal x(nT) is the only stimulus to the system so that the operator relating the input and output sequences is time independent[2,3].

2.2.1 Properties of discrete-time filters.

For a linear discrete system, such as a filter, the input and output sequences may be related by linear difference equations with constant coefficients of the form[2]

$$y(nT) = \sum_{i=0}^{k} a_i x(nT-iT) - \sum_{i=1}^{m} b_i y(nT-iT) \qquad (2.2.1)$$

Such difference equations, whilst they lead directly to a practical filtering algorithm, do not facilitate the determination of the macroscopic properties of a filter. This problem is also encountered in continuous-time filtering where the differential equation relating the input and output signals does not present information on the overall filter characteristics in a readily comprehensible form. In order to overcome this difficulty the Laplace transformation is commonly employed in continuous-time circuit analysis. In discrete-time analysis the Z-transform is used for the same purpose. This is really only an extension of the Laplace transform so a brief consideration should be given to the latter at this stage.

2.2.1.1 The Laplace transformation.

In continuous-time circuit analysis the Laplace transformation is used both to solve the differential equations and also to express the

behaviour of a filter in terms of a transfer function. If the operation of Laplace transformation of a continuous-time function f(t) is denoted by $L\{f(t)\}$, then the filter output y(t) can be related to the filter input x(t) by[4]

$$L\{y(t)\} = (\text{Transfer function}).(L\{x(t)\}) \qquad (2.2.2)$$

The Laplace transformation $L\{f(t)\}$ is defined to be[2]

$$L\{f(t)\} = \int_0^\infty f(t)e^{-st}dt = F(s) \qquad (2.2.3)$$

The resulting function F(s) is seen to be a function of the variable s which has been introduced in the transformation. The term st must be a pure number so the variable s has the units 1/t, that is, it is a frequency variable. In fact, s is a complex frequency variable which may be written

$$s = \sigma + j\omega \qquad (2.2.4)$$

It is perhaps necessary to consider briefly the concept of complex frequency and hence the physical significance of the s-plane[4]. Firstly consider the steady-state sinusoid defined by

$$a(t) = A.\cos\omega_o t \qquad (2.2.5)$$

This real-time function may be rewritten in terms of two rotating phasors.

$$a(t) = \frac{A}{2}(e^{j\omega_o t} + e^{-j\omega_o t}) \qquad (2.2.6)$$

That is, it may be considered as the summation of two phasors rotating

with complex frequencies $j\omega_o$ and $-j\omega_o$ respectively each having amplitude A/2. Hence the sinusoid can be represented in the s-plane by impulse functions of real amplitude A/2 at $s = \pm j\omega_o$.

Consider now that the sinusoid is no longer of constant amplitude but is decaying exponentially for $t > 0$, that is

$$a(t) = A.e^{-\sigma_o t}\cos\omega_o t \qquad (2.2.7)$$

which again may be written in terms of rotating phasors.

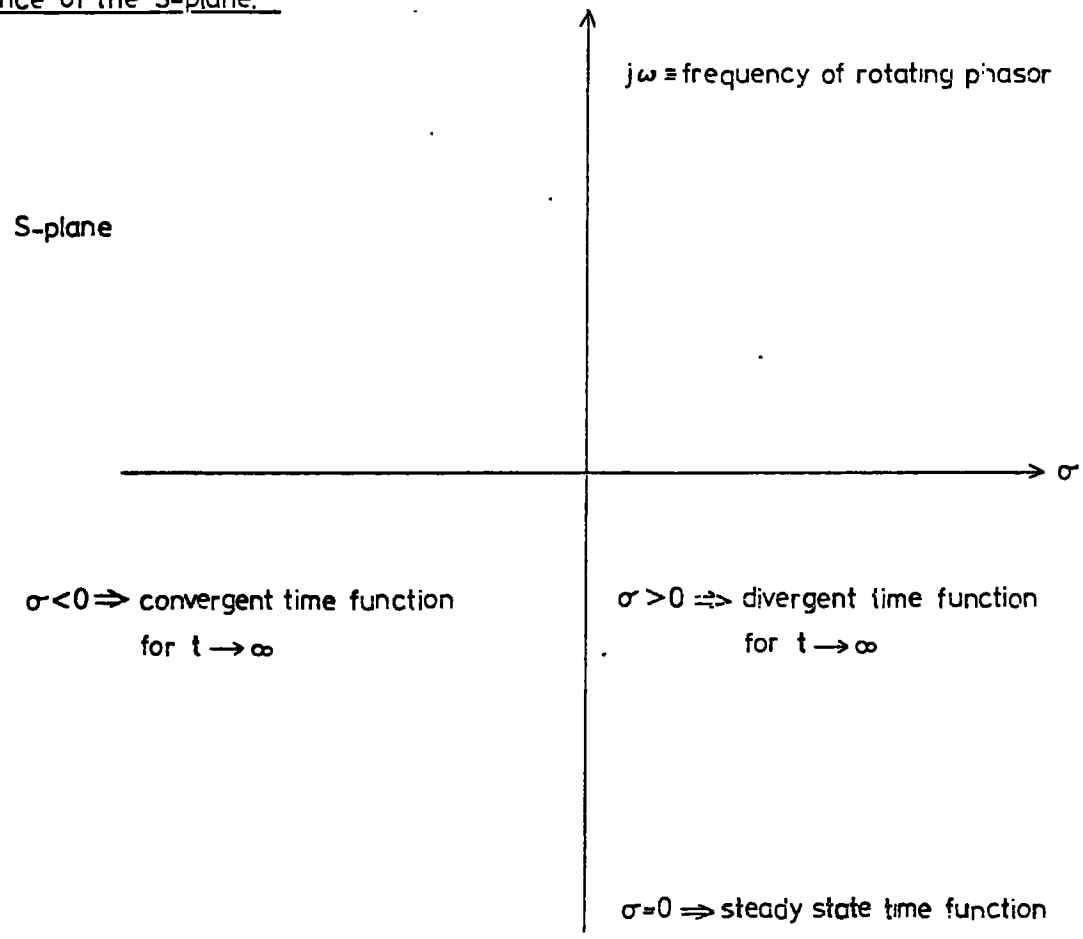
$$a(t) = \frac{A}{2}\{e^{(-\sigma_o + j\omega_o)t} + e^{(-\sigma_o - j\omega_o)t}\} \qquad (2.2.8)$$

It can be seen that the exponentially decaying oscillation may be represented by the sum of two phasors each with real amplitude A/2 and with frequencies $-\sigma_o + j\omega_o$ and $-\sigma_o - j\omega_o$. Hence this function is represented in the s-plane by impulse functions of real amplitude A/2 at $s = -\sigma_o \pm j\omega_o$.

The above examples suggest the following interpretation of the variable $s = \sigma + j\omega$. Periodic functions of constant amplitude have $\sigma = 0$ and are therefore represented on the imaginary axis of the s-plane. Convergent functions for t tending to infinity have $\sigma < 0$ and so map onto the left-hand half of the s-plane. Conversely divergent functions have $\sigma > 0$ and map onto the right-hand half of the s-plane. This is summarised in Figure 2.5.

The s-plane may be used to represent any time function which can be written in the form of a summation of phasors each with a constant complex frequency and a constant complex amplitude. The resulting function is therefore 4-dimensional. It should also be noted that a real-time function is always represented by complex conjugate pairs of frequencies[4].

Figure 2.5 The significance of the S-plane.



Having now gained some insight into the complex frequency variable, it is possible to continue the study of the Laplace transformation and particularly the nature of the transfer function introduced in equation (2.2.2). The Laplace transform of an impulse function at t = 0, $L\{\delta(t)\}$, is equal to 1, from substitution in equation (2.2.3). Therefore if such an impulse function is applied to a circuit as input then the Laplace transform of the resulting output function is equal to the transfer function, which may be termed K(s). This is clear from equation (2.2.2) which may now be rewritten

$$Y(s) = K(s).X(s) \qquad (2.2.9)$$

where $Y(s) = L\{y(t)\}$ and $X(s) = L\{x(t)\}$. The transfer function may be fully factorised and hence expressed in terms of its discontinuities: its poles and zeros

$$K(s) = \frac{(s-z_1)(s-z_2)(s-z_3)\ldots}{(s-p_1)(s-p_2)(s-p_3)\ldots} \qquad (2.2.10)$$

z_1, z_2, z_3 .... are positions of zeros and p_1, p_2, p_3 ....are pole positions. It is worth noting here that a pole at a site in the s-plane for which $\sigma > 0$ indicates that the impulse response as t tends to infinity is divergent, which in turn indicates that the circuit response is inherently unstable.

2.2.1.2 The Z-transformation.

Naturally the Laplace transformation cannot be employed in a discrete-time environment, but an analogous transformation exists for discrete-time analysis. This is called the Z-transformation[5]. A direct discrete-time analogue of the Laplace transformation could be written

$$F(s) = \sum_{n=0}^{\infty} f(nT)e^{-snT} \qquad (2.2.11)$$

However a further change of variable is desirable in order to produce a complex plane with a more convenient coordinate system in view of the periodicity of the frequency variable previously mentioned. This change of variable is defined as

$$z = e^{sT} \qquad (2.2.12)$$

where T is the sampling period and z is a pure complex number variable. Equation (2.2.11) may therefore be rewritten[2]

$$F(z) = \sum_{n=0}^{\infty} f(nT)z^{-n} \qquad (2.2.13)$$

which is the definition of the Z-transformation.

It is instructive to consider the relationship of the z-domain to the s-domain. From equation (2.2.12) z may be rewritten

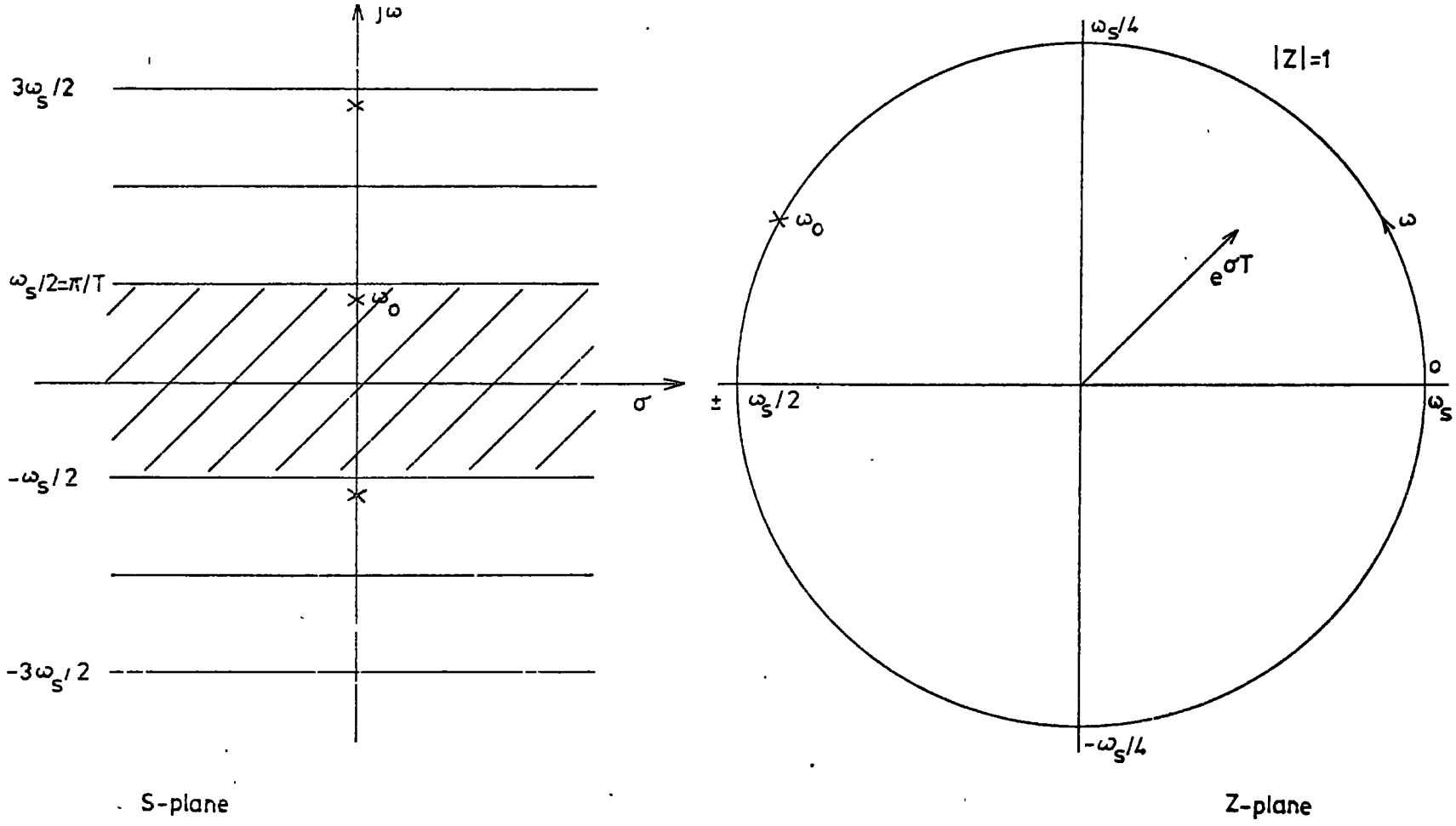
$$z = e^{\sigma T}.e^{j\omega T} \qquad (2.2.14)$$

so that the magnitude of z is given by

$$|z| = e^{\sigma T} \qquad (2.2.15)$$

It is clear that for any constant value of σ, z is a function of the frequency ω and is a periodic function with a period of $2\pi/T$, which exactly follows the phenomenon of aliasing. Hence the infinite series of aliasing frequencies resulting from sampling a single sinusoid all map onto a single point in the z-plane, yielding great clarity and conciseness. (This is demonstrated in Figure 2.6). It is for just this

Figure 2.6 The correspondence between the Z-plane and the S-plane.



reason that the variable change $z = e^{sT}$ is employed.

From Figure 2.6 it can be seen that

(i) $\sigma < 0 \Rightarrow |z| < 1$. The entire left half of the s-plane can be mapped into the interior of the unit circle in the z-plane.

(ii) $\sigma = 0 \Rightarrow |z| = 1$. The imaginary axis of the s-plane can be mapped onto the circumference of the unit circle in the z-plane.

(iii) $\sigma > 0 \Rightarrow |z| > 1$. The entire right half of the s-plane can be mapped onto the exterior of the unit circle in the z-plane.

In the same way as the equation of a continuous filter was stated concisely in the s-domain in equation (2.2.9) the operation of a discrete filter may be thus expressed in the z-domain[2]

$$Y(z) = H(z).X(z) \tag{2.2.16}$$

where Y(z) is the Z-transformation of the output sequence y(nT), X(z) is the Z-transformation of the input sequence x(nT), and H(z) is the z-domain transfer function of the filter. The Z-transformation of the discrete impulse function, $\delta(nT) = \{1 \text{ for } n=0,\ 0 \text{ for } n\neq 0\}$, is clearly equal to 1, so the transfer function H(z) again has the significance of being the transformation of the impulse response. Transforming the general difference equation (2.2.1) into the z-domain yields

$$Y(z)\left(1 + \sum_{i=1}^{m} b_i z^{-i}\right) = X(z) \sum_{i=0}^{k} a_i z^{-i} \tag{2.2.17}$$

so that

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{k} a_i z^{-i}}{1 + \sum_{i=1}^{m} b_i z^{-i}} \tag{2.2.18}$$

where H(z) is a proper rational function of z^{-1}.

2.2.2 Characteristics of a discrete-time filter.

Just as the characteristics of a continuous-time filter may be completely determined from the pole and zero positions of the transfer function in the s-domain, so for the discrete-time filter the pole and zero positions of the transfer function in the z-domain define its characteristics. The steady-state frequency response of a continuous filter is determined by evaluating the transfer function along the imaginary axis in the s-domain. Likewise for a discrete-time filter the steady-state frequency response is obtained by evaluating the transfer function H(z) around the unit circle $|z| = 1$.

In general the transfer function H(z) may be factorised and written[2]

$$H(z) = \frac{(z-z_1)(z-z_2)(z-z_3).....}{(z-p_1)(z-p_2)(z-p_3).....} \qquad (2.2.19)$$

z_1, z_2, z_3... are the zeros and p_1, p_2, p_3... are the poles. The transfer function will have k zeros and m poles where k and m have the same significance as in equations (2.2.1), (2.2.17), and (2.2.18). The order of the filter is said to be equal to the number of zeros or the number of poles, whichever is the greater. As in the case of continuous circuit analysis, where a transfer function pole having $\sigma > 0$ in the s-domain indicates circuit instability, the same inference may be drawn when a pole in the z-domain has $|z| = e^{\sigma T} > 1$. That is, a pole outside the unit circle in the z-plane is prohibited in stable filter design.

The frequency response of the filter may be found by evaluating equation (2.2.19) for $z = e^{j\omega T}$, that is, around the unit circle. The magnitude of the frequency response is given by[6]

$$|H(e^{j\omega T})| = \left|\frac{(e^{j\omega T}-z_1)(e^{j\omega T}-z_2)(e^{j\omega T}-z_3)...}{(e^{j\omega T}-p_1)(e^{j\omega T}-p_2)(e^{j\omega T}-p_3)...}\right| \qquad (2.2.20)$$

Equation (2.2.20) is expressed graphically in Figure 2.7 for a second order filter. $\bar{z}_1$, $\bar{z}_2$, $\bar{p}_1$, and $\bar{p}_2$ are the vectors connecting their respective discontinuity to the frequency ω_r at which the response magnitude is to be evaluated. For this example of a second order filter, equation (2.2.20) is applied thus

$$|H(e^{j\omega_r T})| = \frac{|\bar{z}_1| \cdot |\bar{z}_2|}{|\bar{p}_1| \cdot |\bar{p}_2|} \qquad (2.2.21)$$

An equation of similar form may be applied to a filter of any order.

The phase response at frequency ω_r in Figure 2.7 is given by[6]

$$\psi = (\theta_1 + \theta_2) - (\phi_1 + \phi_2) \qquad (2.2.22)$$

where θ_1, θ_2, ϕ_1, and ϕ_2 are the angles which the vectors $\bar{z}_1$, $\bar{z}_2$, $\bar{p}_1$, and $\bar{p}_2$ make with the positive real axis. Again an equation of similar form is valid for any discrete-time filter. In general the phase response is given by

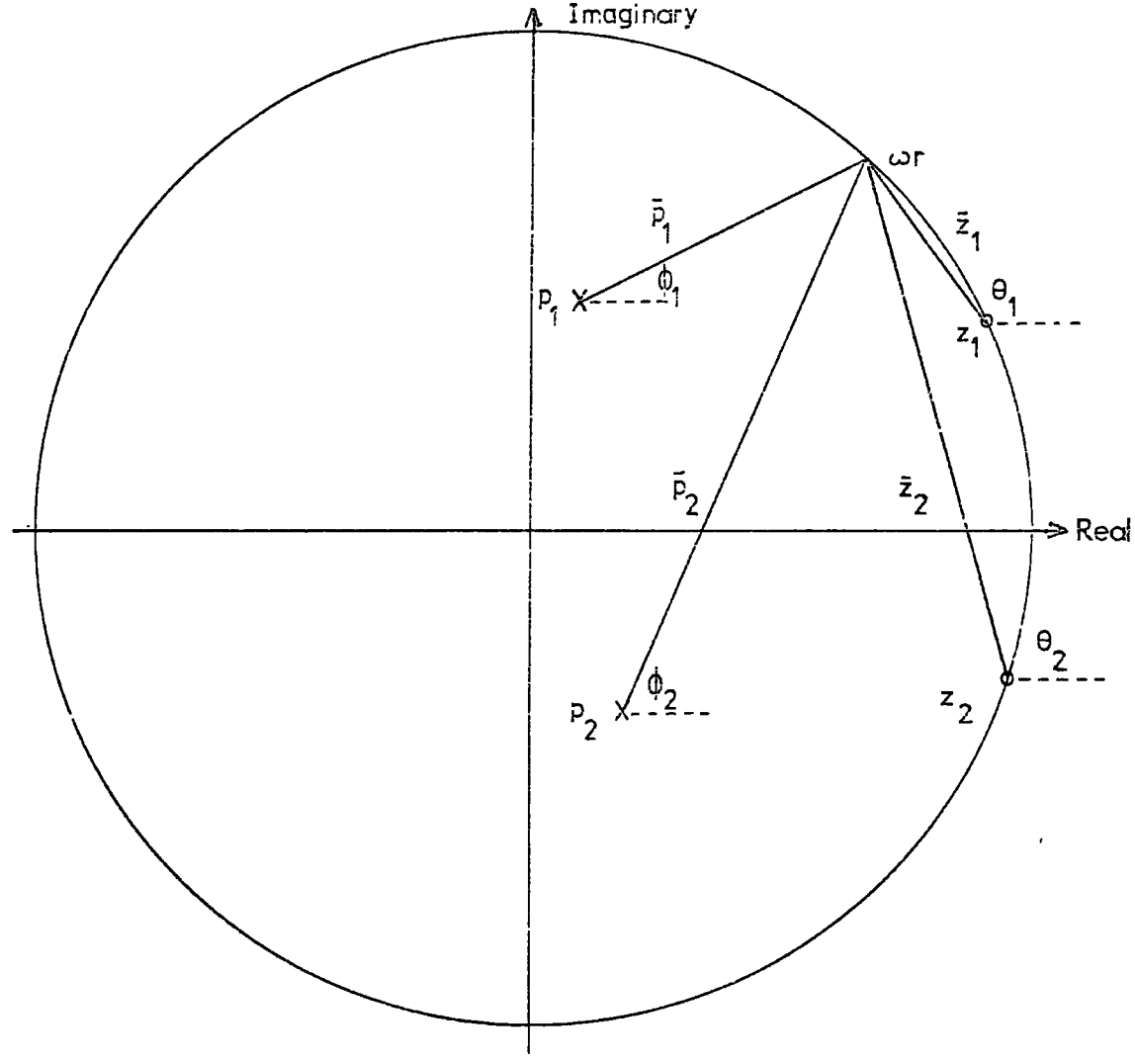
$$\psi(\omega) = \tan^{-1}\left[\frac{\mathrm{Im}\{H(e^{j\omega T})\}}{\mathrm{Re}\{H(e^{j\omega T})\}}\right] \qquad (2.2.23)$$

2.3 Filter realisation.

Having seen the way in which the properties of a filter can be determined via the transfer function H(z) in the z-domain, it is necessary to be able to derive the difference equation corresponding to any required transfer function. In the case of H(z) being in the form of the ratio of two polynomials as in equation (2.2.18), this is a trivial process.

$$H(z) = \frac{\sum_{i=0}^{k} a_i z^{-i}}{1 + \sum_{i=1}^{m} b_i z^{-i}} \qquad (2.2.18)$$

Figure 2.7 A typical pole - zero diagram for the transfer function of a second order filter.



Z-plane

transforms back to the difference equation (2.2.1)

$$y(nT) = \sum_{i=0}^{k} a_i x(nT-iT) - \sum_{i=1}^{m} b_i y(nT-iT) \qquad (2.2.1)$$

2.3.1 Realisation of discrete equivalents of continuous-time filters.

The system function H(z) may not, however, always be directly available. The requirement may be to produce the discrete equivalent of a continuous filter whose transfer function is defined in the s-domain. If this function K(s) is fully factorised, that is, expressed in terms of its poles and zeros, then the corresponding z-domain function H(z) is found by mapping these poles and zeros into the z-domain by the change of variable $z = e^{sT}$. So

$$K(s) = \frac{(s-a_1)(s-a_2)(s-a_3).....}{(s-b_1)(s-b_2)(s-b_3).....} \qquad (2.3.1)$$

gives

$$H(z) = \frac{(z-e^{a_1T})(z-e^{a_2T})(z-e^{a_3T}).....}{(z-e^{b_1T})(z-e^{b_2T})(z-e^{b_3T}).....} \qquad (2.3.2)$$

It is rather more common for K(s) to be given in a less convenient form such that it is not possible to map poles and zeros from the s-plane into the z-plane directly. One method which can be used in this situation is called impulse invariance[6]. This yields a discrete filter whose response to an impulse is identical to the sampled impulse response of the continuous filter. The first step is to determine the impulse response, k(t), of the continuous filter by taking the inverse Laplace transform of the transfer function K(s). So

$$k(t) = L^{-1} \{ K(s) \} \qquad (2.3.3)$$

The effect of sampling k(t) is to produce a discrete sequence h(nT) so that

$$h(nT) = k(t) \quad \text{for } t=nT \text{ and } n \text{ integral.} \tag{2.3.4}$$

The z-domain transfer function may be found by taking the Z-transform of the sequence h(nT). That is,

$$H(z) = Z\{ h(nT) \} = \sum_{n=0}^{\infty} h(nT)z^{-n} \tag{2.3.5}$$

To take the general example of

$$K(s) = \sum_{i=1}^{m} \frac{A_i}{s + s_i} \tag{2.3.6}$$

$$k(t) = L^{-1}\{ K(s) \} = \sum_{i=1}^{m} A_i e^{-s_i t} \tag{2.3.7}$$

Therefore

$$h(nT) = \sum_{i=1}^{m} A_i e^{-s_i nT} \tag{2.3.8}$$

and

$$H(z) = \sum_{i=1}^{m} A_i \sum_{n=0}^{\infty} e^{-s_i nT}.z^{-n} \tag{2.3.9}$$

which may be rewritten

$$H(z) = \sum_{i=1}^{m} \frac{A_i}{1 - e^{-s_i T}.z^{-1}} \tag{2.3.10}$$

Hence the transformation from the s-domain to the z-domain can be

written[6]

$$K(s) = \sum_{i=1}^{m} \frac{A_i}{s + s_i} \Rightarrow \sum_{i=1}^{m} \frac{A_i}{1 - e^{-s_i T}.z^{-1}} = H(z) \qquad (2.3.11)$$

Once H(z) has been determined some further manipulation is usually required in order to ~~state~~ express the function in a form which can be conveniently transformed to give the appropriate difference equation. Other methods of determining the transfer function have not been found necessary during the course of this work and have therefore not been described here[2,6-12].

2.3.2 High order filter realisation.

It is often not desirable, mainly for reasons of circuit stability, to realise high order filters[2] by the implementation of a single transfer function H(z). Hence the total filtering operation is often subdivided into several processes which synthesise to give the required overall transfer function. There are two basic ways in which this subdivision may be achieved, yielding, in one case, a cascade of subfilters, and in the other case, a parallel configuration of subfilters.

If the transfer function H(z) is fully factorised then the subfilters, $H_i(z)$, may be formed by choosing small sub-sets of poles and zeros from the total function. The subfilters are then arranged in a cascade to achieve the overall transfer function. This may be expressed mathematically by

$$H(z) = H_1(z) \times H_2(z) \times ..... \times H_\ell(z) \qquad (2.3.12)$$

This realisation is shown in Figure 2.8 from which it can be seen that the output from one subfilter forms the input to the adjacent subfilter in the cascade. These subfilters may theoretically be arranged in any

Figure 28 Series cascading of low order subfilters

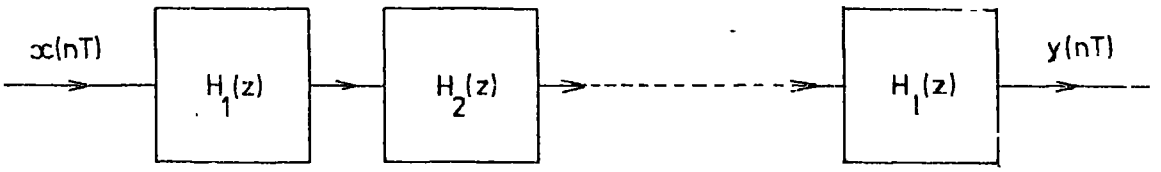
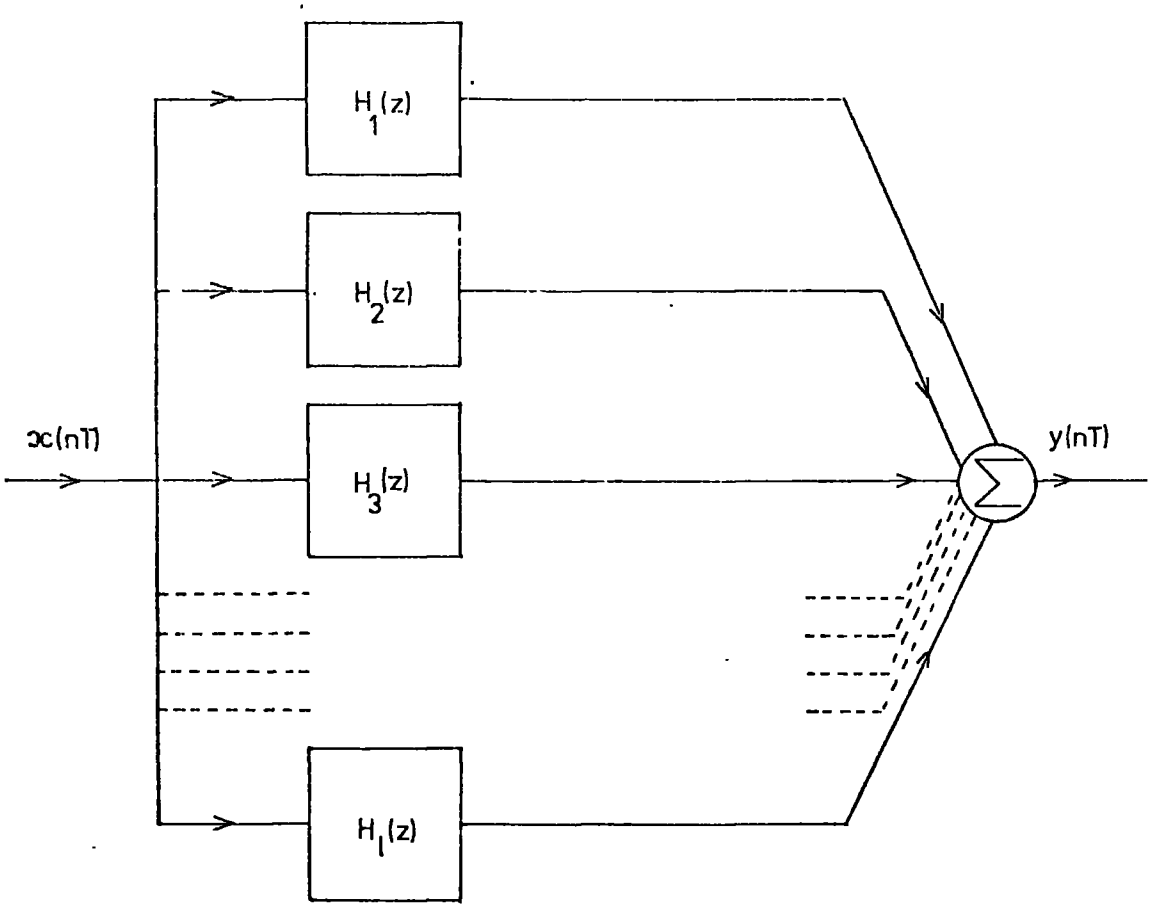


Figure 29 The parallel form of a high order filter



order in the cascade, but in practice there may well be one particular ordering which yields optimum performance[13,14].

The parallel configuration shown in Figure 2.9 may be defined mathematically by

$$H(z) = \sum_{i=1}^{\ell} H_i(z) \qquad (2.3.13)$$

where ℓ is an appropriate constant. Figure 2.9 indicates that the separate outputs from the subfilters are summed sample by sample to yield the overall output y(nT). That is,

$$y(nT) = \sum_{i=1}^{\ell} y_i(nT) \qquad (2.3.14)$$

This realisation has practical advantages over the cascade form, but the mathematical manipulation required to determine the functions $H_i(z)$ from H(z) is often a major disadvantage, whether H(z) is in the form of a ratio of two polynomials in z^{-1} or in fully factorised form as in equation (2.2.19). In practice the subfilters in either of the configurations need be of no greater complexity than second order.

2.3.3 Circuit flow-diagrams of difference equation realisations.

Once the difference equation has been determined for a required subfilter or filter, there are several circuit configurations which may be employed in its realisation. These are presented below using the terminology suggested by Rabiner[15] for use in the circuit flow-diagrams of discrete filters, which is reproduced in Figure 2.10.

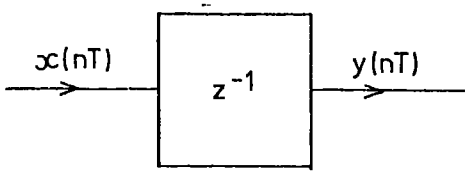
2.3.3.1 Realisation of recursive difference equations.

If, in the difference equation of a filter (equation (2.2.1)), the constant coefficients b_i are not all equal to zero, then the filter is

Figure 2.10 The recommended terminology used in discrete filtering.

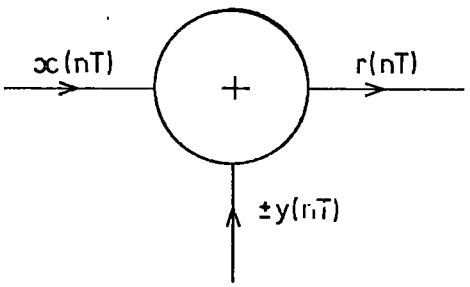
Unit delay

$y(nT) = x((n-1)T)$



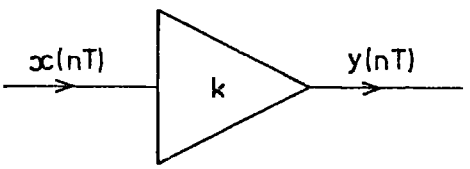
Adder / subtractor.

$r(nT) = x(nT) \pm y(nT)$

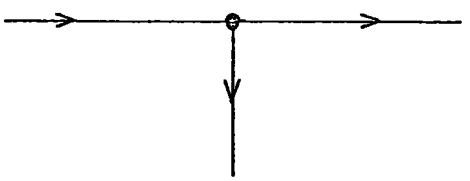


Constant multiplier.

$y(nT) = k \cdot x(nT)$



Branching operation.



said to be recursive, that is, the current output y(nT) is dependent on previous outputs y((n-i)T). (Such filters have an infinite impulse response, IIR). A recursive difference equation may be realised in ~~one of two general~~ various forms, such as

(i) Direct form[2].

Equation (2.2.1) can be realised directly as in Figure 2.11.

(ii) Canonic form.

An intermediate state may be defined in the z-domain as

$$W(z) = \frac{X(z)}{1 + \sum_{i=1}^{m} b_i z^{-i}} \qquad (2.3.15)$$

which enables equation (2.2.18) to be rewritten

$$Y(z) = \left(\sum_{i=0}^{k} a_i z^{-i} \right) . . W(z) \qquad (2.3.16)$$

Transforming these two equations into the time-domain gives

$$w(nT) = x(nT) - \sum_{i=1}^{m} b_i w(nT-iT) \qquad (2.3.17)$$

and

$$y(nT) = \sum_{i=0}^{k} a_i w(nT-iT) \qquad (2.3.18)$$

which may be represented[2] by Figure 2.12.

Comparing Figures 2.11 and 2.12, it can be seen that both forms comprise two basic summation stages, one realising the transfer function zeros and the other the poles. The difference between them lies in the reversal of the order in which the two steps are executed. Figure 2.12 may be realised more concisely yielding a saving in the number of delay elements required and hence a reduction in data storage. The resulting

Figure 2.11 Circuit flow diagram of the direct form for a kth. order filter.

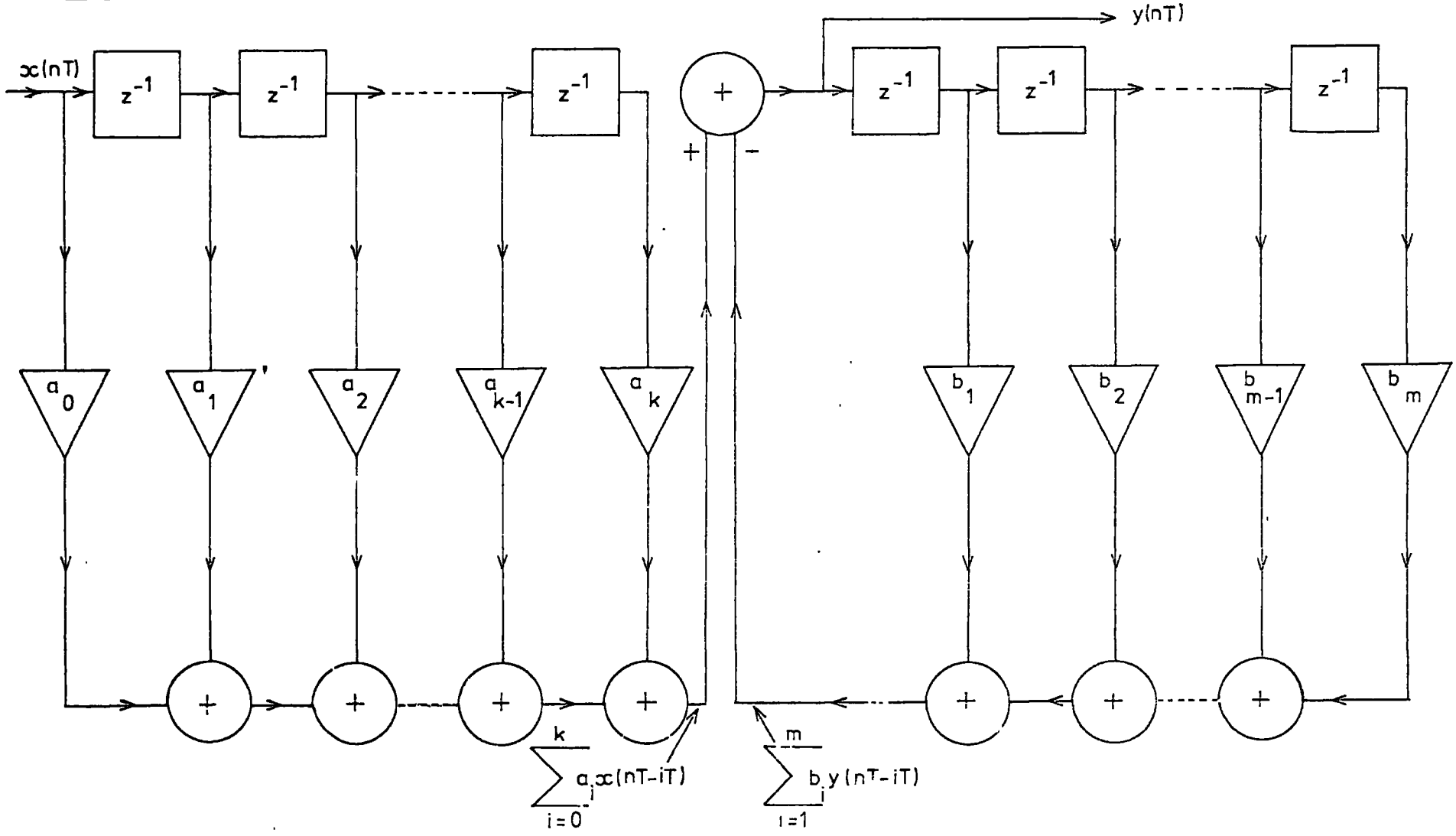


Figure 2.12 A direct filter realisation from which the Canonic form is derived.

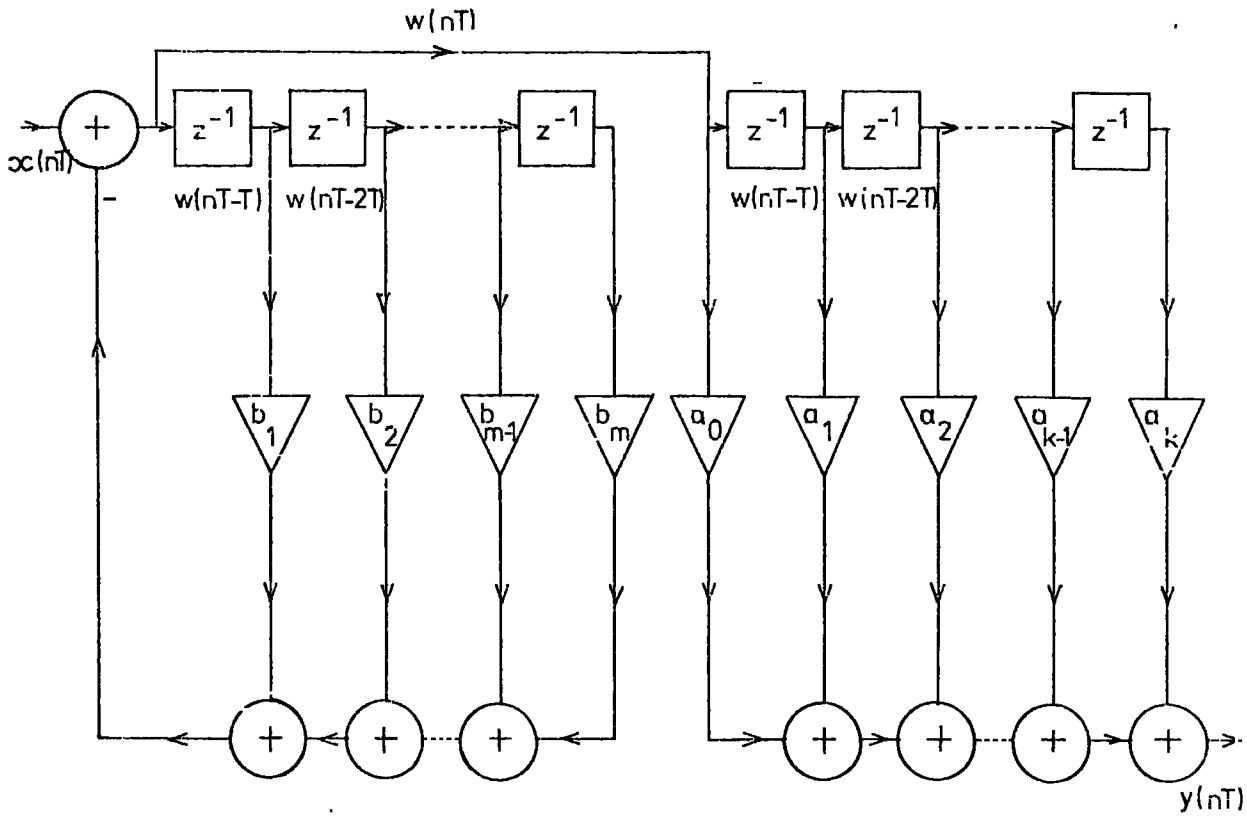


Figure 2.13 Circuit flow diagram of the canonic form of a kth order filter

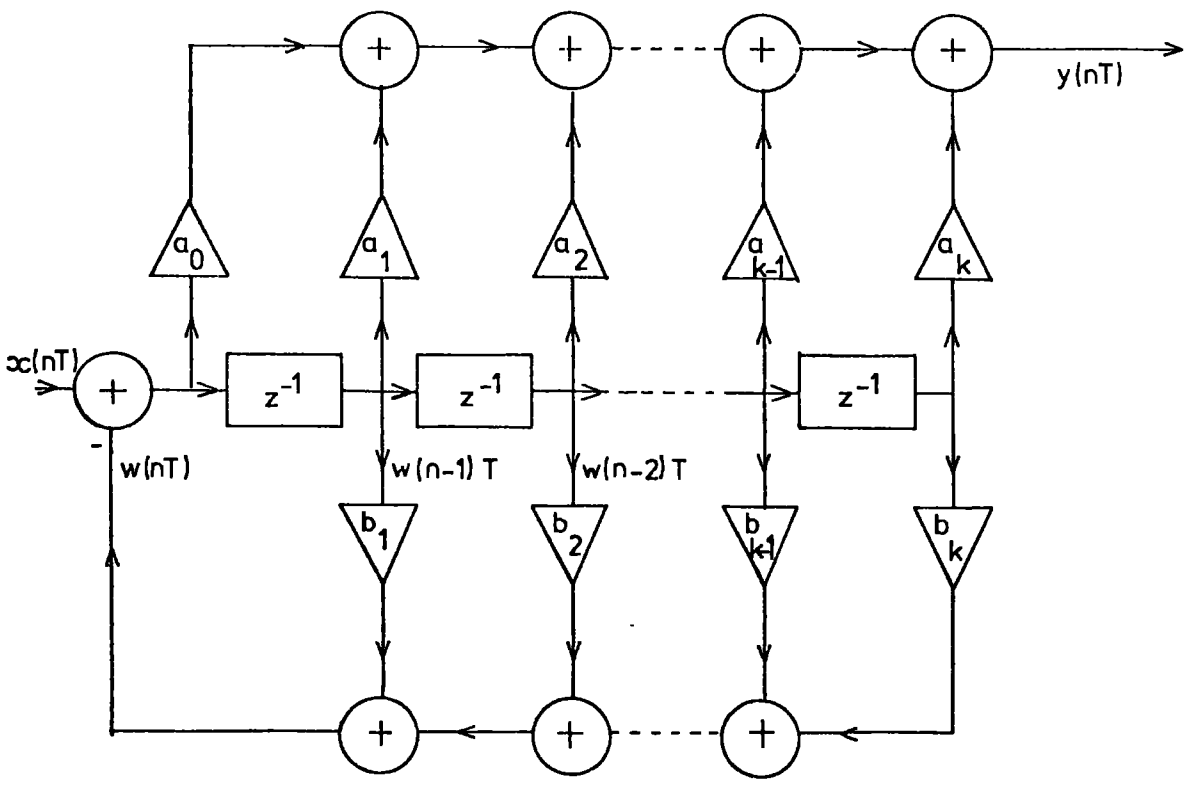
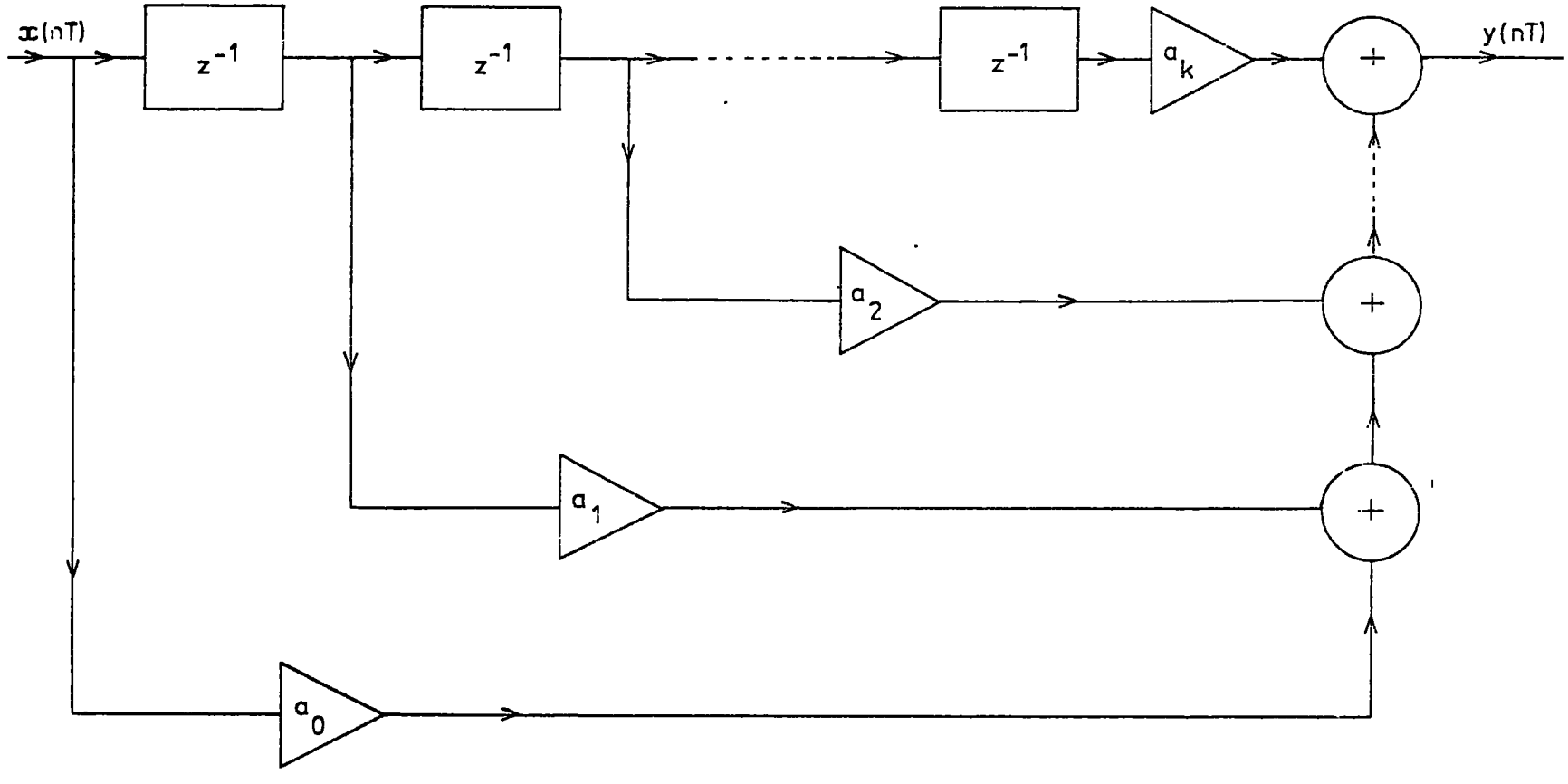


Figure 2.14 Circuit flow diagram of a kth. order non-recursive filter



canonic form[2] is depicted in Figure 2.13.

Theoretically both these forms have identical performance in every respect, but in practice one of the forms will probably more nearly achieve the required specification than the other[10].

2.3.3.2 Realisation of non-recursive difference equations.

If the constant coefficients b_i in the difference equation (2.2.1) are all equal to zero, the resulting filter is said to be non-recursive, that is, the current output y(nT) is only dependent on present and past input samples; there is no feedback from the output. (These filters have a finite impulse response, FIR.) Non-recursive filters are usually of a higher order than is common in recursive filters. Figure 2.14 shows the only common form applicable to a non-recursive filter[2].

2.4 Summary.

The above discussion has attempted to set out the fundamental theory of discrete-time linear filtering. All the common structures for implementing filters by means of a linear difference equation algorithm have been introduced. Attention must now be turned to the practical realisation of such discrete-time linear filters.

CHAPTER 3 - FINITE WORDLENGTH EFFECTS IN DIGITAL FILTERS.

3.1 Introduction.

Hitherto the discussion has been of 'discrete-time' signal processors, placing the emphasis on the consequences of sampling a continuous signal at instants in time. This emphasis must now be shifted to coincide with the main concern of this work. The purpose of converting continuous signals to discrete sequences of numbers is to permit signal processing by digital electronic techniques using either a programmable digital computer or a digital circuit designed to execute a particular operation on a signal. An immediate consequence of using such devices is that, not only is it impossible to represent the time variable continuously, it is similarly impossible to represent the amplitude variable continuously. That is, there is a finite set of numbers which may be represented in a digital electronic device. Hence it is impossible to carry out the arithmetic operations, described previously, with absolute precision.

3.2 Number representation.

A number in a digital device is represented by one or more binary digits (bits) which are either logical 1 or 0. An array, or word, of n bits can represent 2^n distinct numbers and, in principle, the decision on the meaning given to any particular combination of 1's and 0's is purely arbitrary. However, it is clearly advantagecus to adopt a systematic scheme of representation which permits arithmetic operations without complicated programming or circuitry. Four such systems are described below.

3.2.1 Fixed-point arithmetic.

The simplest system of number representation is called fixed-point

arithmetic. If the wordlength of the digital device is taken to be (B+1) bits then all these bits are taken to represent one of a set of numbers defined by

$$x_n = \left(-x_{n,B}.2^B + \sum_{j=0}^{B-1} x_{n,j}.2^j\right).\Delta \qquad (3.2.1)$$

where $x_{n,0}$ is the least significant bit of the word representing x_n, $x_{n,B-1}$ is the most significant bit and $x_{n,B}$ is the sign bit. The quantity Δ is the separation between consecutive allowed numbers and is equal to the value attributed to the least significant bit of the word; it may be termed the quantisation width. Equation (3.2.1) defines a system of fixed-point number representation called two's complement notation. Other forms of fixed-point representation are occasionally encountered but there is no need to discuss them here. Using equation (3.2.1) it is possible to represent 2^{B+1} different numbers in the range $-2^B.\Delta$ to $(2^B-1).\Delta$ inclusive. The quantisation width Δ is constant over the whole range. It is often convenient to consider Δ equal to unity, so that the numbers represented are integers.

3.2.2 Floating-point arithmetic.

The second main system of number representation is floating-point arithmetic. A sub-set of the bits in a word are interpreted as a fixed-point two's complement mantissa, whilst the remaining bits represent a two's complement exponent.

For a given wordlength, (B+1), floating-point arithmetic allows a greater range of numbers to be represented than is possible using fixed-point arithmetic, (that is, the ratio of the largest:smallest allowable numbers is greater). The separation between adjacent numbers is no longer constant but depends on the magnitude of the numbers. For example, if an 8-bit wordlength is used, 4 bits may express a two's complement

fractional mantissa whilst the other 4 bits represent a two's complement integral exponent of 2. The separation, Δ, between two adjacent numbers having the same exponent m is therefore $2^{-3}.2^{m}$.

It will be appreciated that arithmetic operation using such number representation is much more complicated than the fixed-point mode, although in general greater accuracy is possible. Whereas it is perfectly possible to implement floating-point arithmetic on a device such as a microprocessor, it is seldom desirable. Hardware floating-point arithmetic units are not at present available on microprocessors so software must be used to realise this arithmetic mode. This both increases the memory requirement and reduces the speed of the processor, factors which weigh heavily against the use of floating-point arithmetic. For this reason the implementation of digital signal processors using floating-point arithmetic has not been considered further.

3.2.3 Block-floating-point arithmetic.

Oppenheim[16] has suggested a scheme of number representation called block-floating-point arithmetic, which is intended to exhibit some of the advantages of floating-point arithmetic without incurring the same level of programming complexity in a fixed-point device.

All the signal data items held in a device during any particular signal sampling period are scaled up or down by the same power of 2. The exponent of 2 used is held as an additional data item. All the arithmetic operations required are performed in fixed-point arithmetic on the scaled data items, making no reference to the stored exponent. A scaled result is thus produced which requires to be rescaled by the power of 2 indicated by the stored exponent.

The purpose of using this method is to make better use of the available wordlength than is possible using fixed-point arithmetic. For example, in fixed-point arithmetic, the signal data items at one

particular instant may only require a maximum of 4 out of an available 8 bits for representation. The result of an arithmetic operation on these data items may require only 5 bits. If the data items are scaled up by 2^3, then the result before rescaling will occupy the full 8 bits. The rescaled 5-bit result may well be more accurate than that produced by simple fixed-point arithmetic. The experimental results of testing this expectation of increased accuracy are presented in Chapter 5.

3.2.4 Modular arithmetic.

Interest has recently been developing in the utilisation of what may be regarded as a fourth mode of number representation; this may be termed modular arithmetic. Only positive integers are required to be stored in the computer, which makes this form of representation equivalent in complexity to two's complement fixed-point arithmetic. Agarwal & Burrus[17] have recently published a tutorial paper in which they indicate the advantages of using modular arithmetic for transformations used to implement digital convolution. Use of block filtering methods proposed by Burrus[18] allows recursive, or infinite impulse response (IIR), filters to be realised via such transformations. Digital filters realised in this manner require only the addition of data items; no multiplication is necessary. This often leads to increased speed and accuracy[19]. There are, however, detailed problems encountered in this form of filter realisation[20], but these are gradually being solved[21].

3.3 Error sources.

In a practical digital signal processor, such as a filter, there are three basic sources of deviation from the theoretical performance. (i) When a signal is originally converted from a continuous signal to a discrete sequence by an analogue to digital converter the digital output has a finite wordlength so that the amplitude of the continuous input

signal cannot be represented with absolute precision. This error source may be called input quantisation. The effects of input quantisation have been studied quite extensively[22-30] and appear to be well understood, so that little research interest remains.

(ii) Any general digital signal processor requires arithmetic operation involving constant coefficients. Again, because of the finite wordlength of a practical device, only a discrete set of coefficients can be stored. Hence the required coefficients determined by theoretical design can only be approximated. This source of error is termed coefficient quantisation. The effects of coefficient quantisation have also received close attention[8,10,31-46]. The emphasis of such research has been on the development of design techniques which optimise the characteristics of a signal processor implemented using allowed coefficients.

(iii) One of the major operations in a general signal processor is the multiplication of a signal data item by a constant coefficient. If, for example, the signal data items are stored as two's complement, fixed-point integers and the coefficients are held as fixed-point fractions, then the result of a multiplication between the two would, in general, be a number with both an integral and fractional part. Such a number cannot be held and so must be approximated to an integer. The error source exemplified may be called multiplication roundoff, and is perhaps the most interesting of the three sources described. It is with this area in particular that this current work is concerned. The standard theories on the subject will be presented in the following sections. The purpose of this work is to test the veracity of some of these theories for short wordlengths of 8 bits or less.

3.4 Roundoff processes in fixed-point arithmetic.

As mentioned above, the result of a multiplication must in general be approximated to a number which can be stored. There are three basic

mechanisms which may be employed to execute this approximation: namely truncation, sign-magnitude truncation and rounding. Each yields characteristic properties and has certain advantages and disadvantages.

3.4.1 Truncation.

If a two's complement integer of wordlength n is multiplied by an unsigned, fixed-point fraction of wordlength m, the resulting number requires (m+n) bits to represent it accurately, but only n bits are available. If the m least significant bits of the accurate product are simply discarded leaving the n most significant bits, the process is termed truncation. This is the easiest form of approximation to implement. The characteristics of this approximation method are displayed in Figure 3.1 for two's complement fixed-point arithmetic.

It will be observed from Figure 3.1 that the approximate product is always the allowed number just less than the accurate product. Hence if the error ε incurred is defined by

$$\varepsilon = \text{approximate answer} - \text{accurate answer} \qquad (3.4.1)$$

then ε must lie in the range $0 \geqslant \varepsilon > -\Delta$. If a sequence which is symmetrical about zero is multiplied by a constant coefficient, then truncated, the resulting sequence will no longer be symmetrical about zero, but a negative shift will have been applied to the mean level. This is a disadvantage of the simple truncation process.

3.4.2 Sign-magnitude truncation.

Because of the non-symmetrical nature of the simple truncation process, a variant called sign-magnitude truncation is sometimes preferred, despite the increased complexity of implementation. Figure 3.2 depicts the characteristics of this process.

Figure 3.1 The truncation process

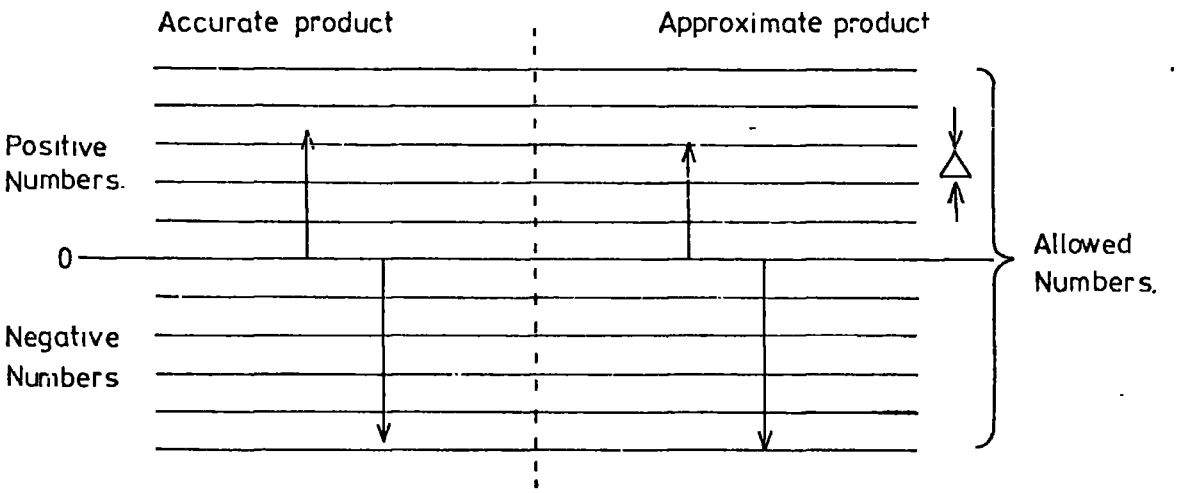


Figure 3.2 The sign-magnitude truncation process

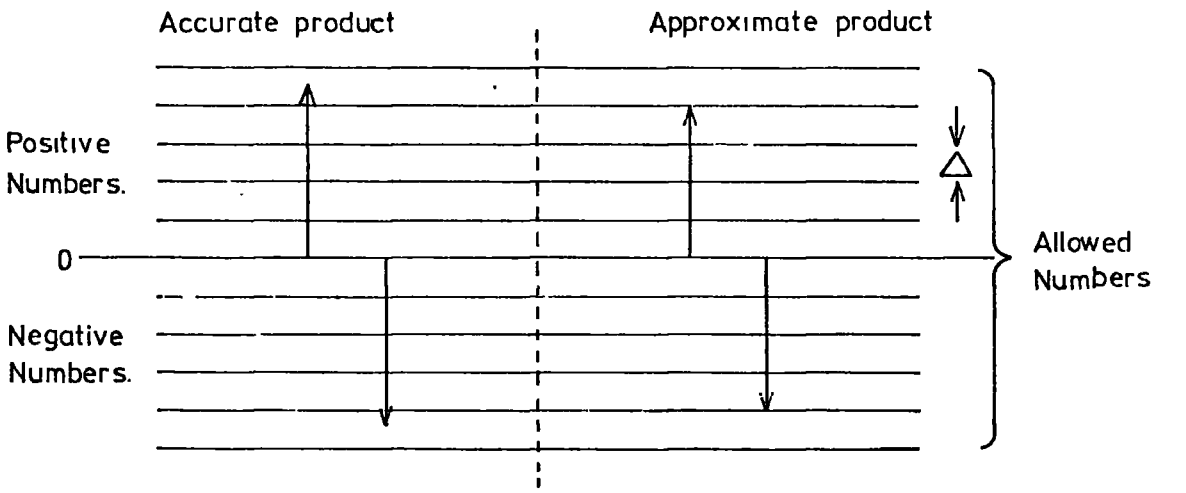


Figure 3.3 The rounding process.

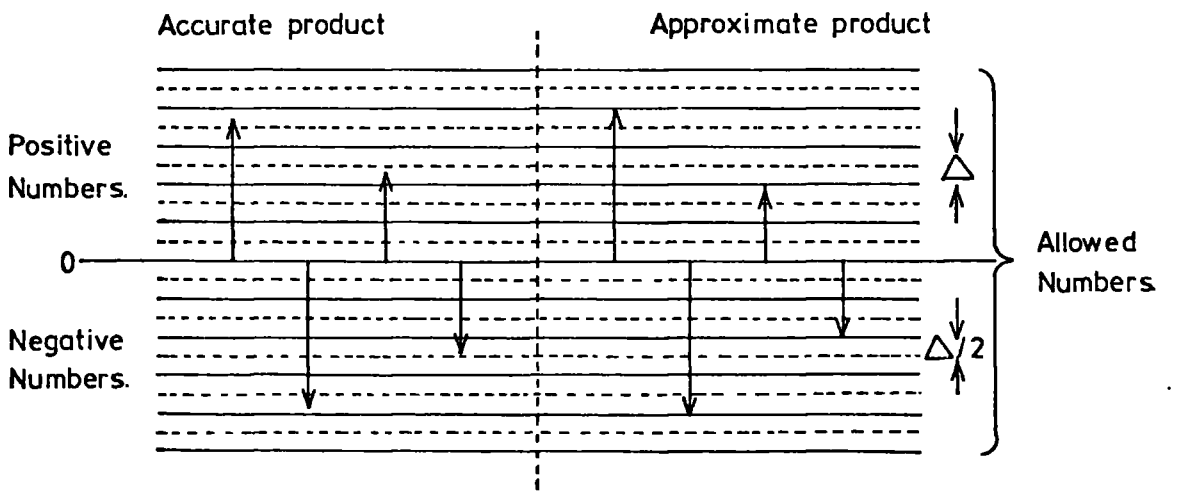


Figure 3.2 indicates that the approximate answer is always the allowed number whose magnitude is just less than the magnitude of the accurate answer. Such a process is symmetrical. A positive number yields a negative error ε in the range $0 \geqslant \varepsilon > -\Delta$, whilst a negative number results in the production of a positive error ε in the range $\Delta > \varepsilon \geqslant 0$. This correlation between the sign of the result and the sign of the error is disadvantageous in many signal processing applications.

3.4.3 Rounding.

In order to retain the symmetry of sign-magnitude truncation without incurring the problems of correlation between the sign of the error and the sign of the product, it is necessary to employ the process of rounding. This is the most difficult of the three processes to implement, but in every other respect is the most advantageous. Figure 3.3 describes this approximation method.

In each example depicted in Figure 3.3 the allowed number is chosen whose value most nearly approximates the accurate result. The process is symmetrical and generates an error with minimum magnitude. The error must lie in the interval $-\Delta/2 < \varepsilon \leqslant \Delta/2$. The magnitude of the error must be less than or equal to $\Delta/2$ whereas in the other two methods the error magnitude is bounded by Δ. For this reason rounding is nearly always employed to minimise finite wordlength roundoff errors in a digital signal processor.

3.5 A theoretical model for roundoff error processes.

Having described the sample by sample behaviour of each of the three approximation methods, it is important to attempt to formulate a theoretical model of the effects of such approximation on a complete system. This has been achieved by considering the effect of multiplying a sequence by a constant coefficient and making certain assumptions

about the resulting sequence of roundoff errors. The input sequence is assumed to be wide-sense-stationary. The theory has been developed for the rounding process as this is the best behaved and most widely used form of roundoff.

Knowles and Edwards[29], making reference to earlier publications[24,25,28 ~~23,24,27~~], state the two following assumptions for the rounding error process in fixed-point arithmetic realisations, which have formed the foundation of all ensuing error analyses[13,16,47-57].

(i) All error values within the allowed interval have equal probability density.

(ii) Rounding errors form a random sequence which is uncorrelated with either the input or output sequence at a multiplier, so that the error sequence may be regarded as having a uniform spectral density, that is, it may be termed white noise and treated as such.

Knowles and Edwards preface these assumptions with one prior restriction, that the separation between allowed numbers, Δ, must be small, presumably in comparison to the signal level. The implication is that the above assumptions become invalid at some stage as Δ increases relative to the signal level, that is, as the effective wordlength decreases. Liu[50] suggests that these assumptions are 'quite satisfactory at least down to a wordlength of 8 bits.'

It is important to note that all of the works[24,25,28 ~~23,24,27~~] cited by Knowles and Edwards[29] consider the situation of a signal whose amplitude can be continuously variable being quantised to a discrete amplitude system. This is an inherently different problem from the one discussed by Knowles and Edwards, where a signal which is already quantised is multiplied, and the product must also be quantised.

Following the first of the two assumptions, the probability density for the errors incurred at a single multiplication employing rounding may be depicted as in Figure 3.4. The error sequence arising from such

Figure 3.4 Probability density for rounding errors

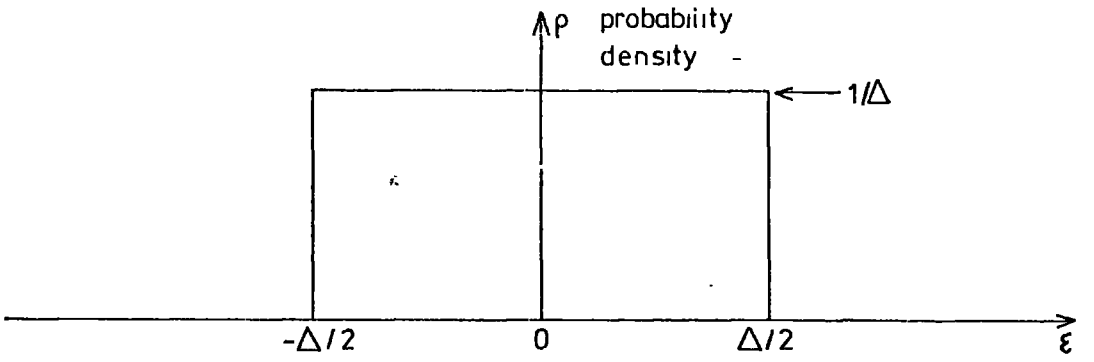


Figure 3.5 Probability density for truncation errors

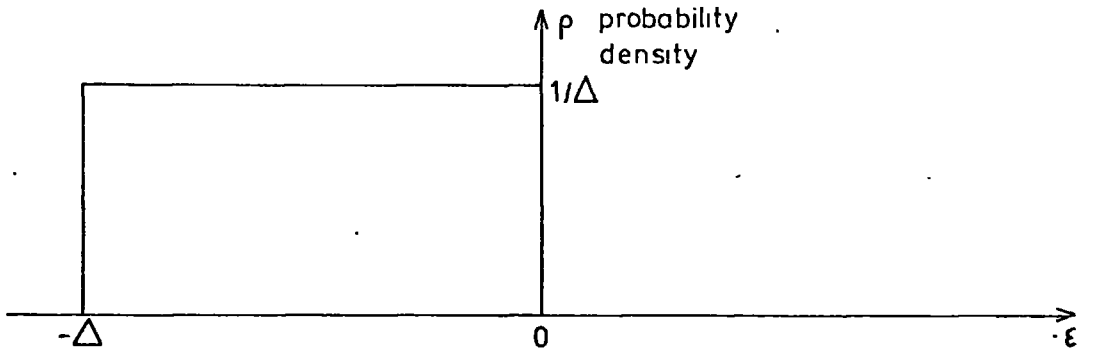
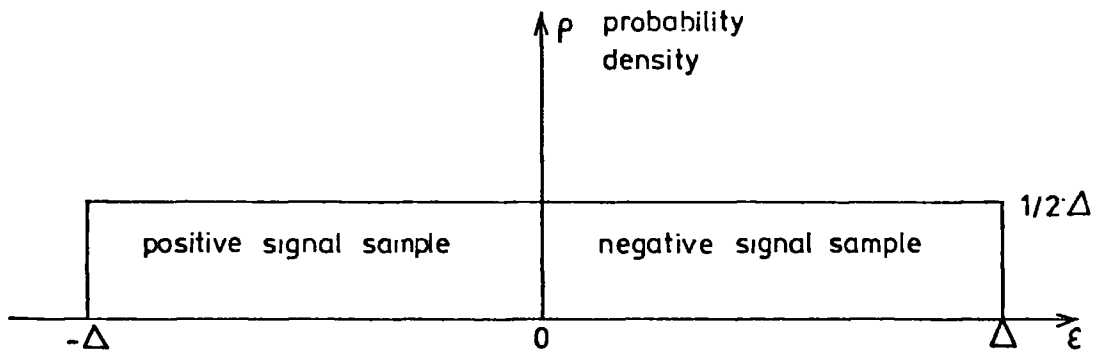


Figure 3.6 Probability density for sign-magnitude truncation errors.



a probability density obviously has zero mean. The variance of the error sequence from its mean, σ^2, is calculated to be[29]

$$\sigma^2 = \frac{\Delta^2}{12} \qquad (3.5.1)$$

Applying the same assumption to the simple truncation process gives a mean error of $-\Delta/2$ and a variance about the mean again of $\Delta^2/12$. The assumed probability density is depicted in Figure 3.5.

In order to apply the same kind of theory to sign-magnitude truncation, it is necessary to add the constraint that an input sample to the multiplier has an equal probability of being either positive or negative. In this case the probability density may be depicted as in Figure 3.6. The mean error is zero, but the variance of the error sequence about its mean is given by

$$\sigma^2 = \frac{\Delta^2}{3} \qquad (3.5.2)$$

It should be noted that such an error is highly correlated with the signal, so that sign-magnitude truncation cannot be treated by the Knowles-Edwards model. This problem has been considered by Liu and Van Valkenburg[52] and more recently by Claasen, Mecklenbräuker and Peek[58].

3.6 Error analysis of some filter realisations.

Before performing error analysis on particular forms of filter, it is necessary to establish the rules of such analysis which arise out of the theoretical model formulated in §3.5:

(i) If an error sequence having variance σ^2 is multiplied by a constant k, then, neglecting any further error caused at the multiplication, the variance of the resulting error sequence[10] is $k^2.\sigma^2$.

(ii) As all error sequences are assumed to be purely random, when a

number of simultaneous error samples from different error sources are summed $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 + \ldots$, the resulting error sequence is also random and has variance[10] given by $\sigma^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_4^2 + \ldots$.

3.6.1 Direct realisation in fixed-point arithmetic.

Consider the direct realisation of a second order filter shown in Figure 3.7. The input sequence, $\{x_n\}$, is assumed to be free from error so that only the effect of errors committed in the filter is under examination. Each of the five multipliers will create an error sequence which is assumed to be of the kind indicated by the theoretical model. Hence, for rounding, each of the error sequences $\varepsilon_1 \ldots \varepsilon_5$ has zero mean and a variance about the mean of $\Delta^2/12$, which is termed σ_ε^2.

The actual filter output may be written

$$v_n = a_0x_n + a_1x_{n-1} + a_2x_{n-2} - b_1v_{n-1} - b_2v_{n-2} + (\varepsilon_{1,n} + \varepsilon_{2,n} + \varepsilon_{3,n} - \varepsilon_{4,n} - \varepsilon_{5,n}) \qquad (3.6.1)$$

The sequences $\varepsilon_1 \ldots \varepsilon_5$ may be combined to give the sequence ε_ξ so that equation (3.6.1) may be rewritten

$$v_n = a_0x_n + a_1x_{n-1} + a_2x_{n-2} - b_1v_{n-1} - b_2v_{n-2} + \varepsilon_{\xi,n} \qquad (3.6.2)$$

The error sequence at the filter output is defined by

$$\varepsilon_{\eta,n} = v_n - y_n \qquad (3.6.3)$$

where $\{y_n\}$ is the ideal signal output sequence. Hence equation (3.6.2) can be written

Figure 3.7 An equivalent circuit for a practical direct realisation of a second order filter

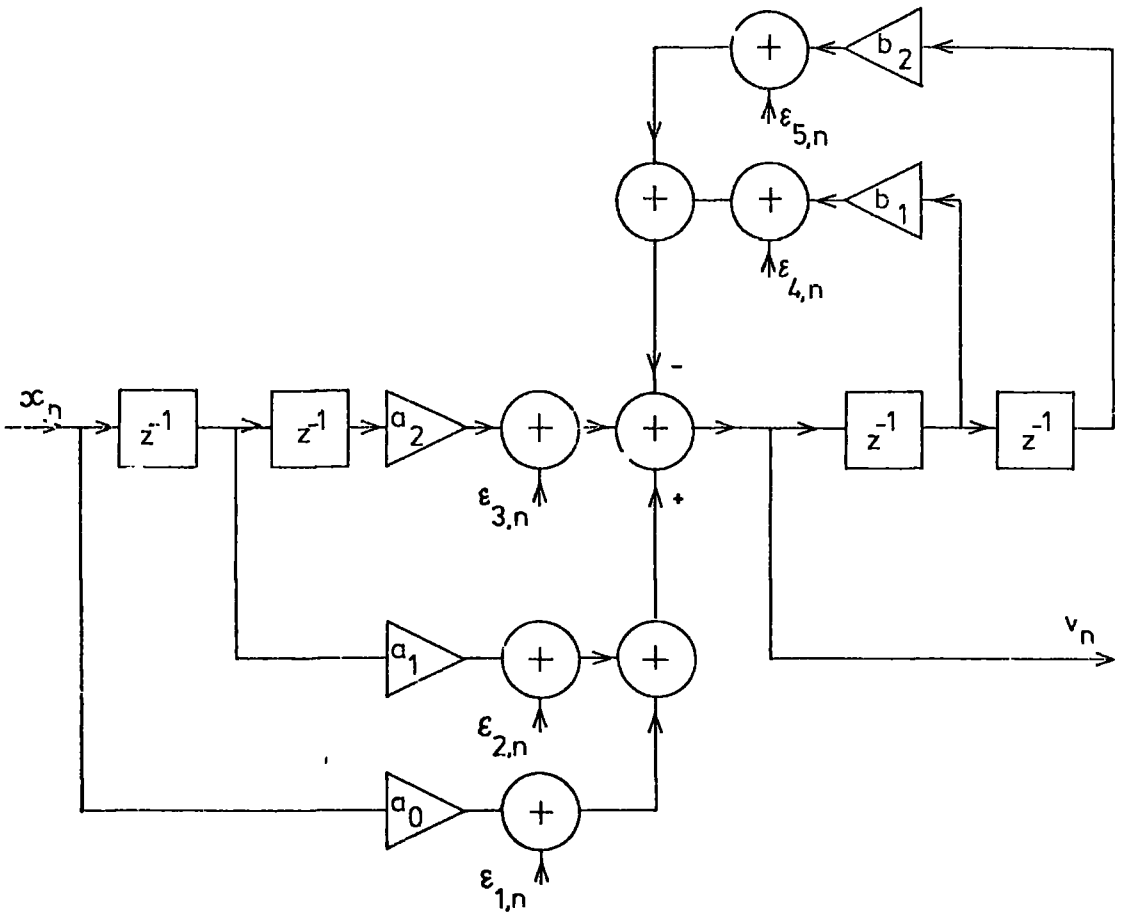
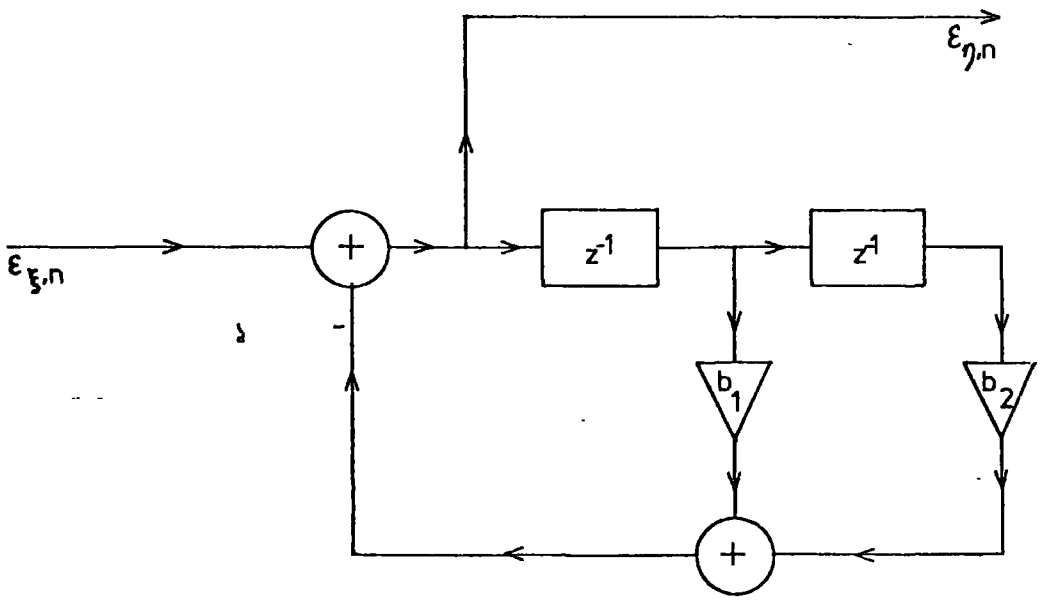


Figure 3.8 Equivalent noise model for a direct realisation of a second order filter



$$y_n + \varepsilon_{\eta,n} = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} - b_1 y_{n-1} - b_2 y_{n-2} - b_1 \varepsilon_{\eta,n-1} - b_2 \varepsilon_{\eta,n-2} + \varepsilon_{\xi,n} \quad (3.6.4)$$

which simplifies to

$$\varepsilon_{\eta,n} = \varepsilon_{\xi,n} - b_1 \varepsilon_{\eta,n-1} - b_2 \varepsilon_{\eta,n-2} \quad (3.6.5)$$

This equation can be represented by the equivalent noise model shown in Figure 3.8.

Transforming equation (3.6.5) into the z-domain gives

$$E_\eta(z) = \frac{E_\xi(z)}{(1 + b_1 z^{-1} + b_2 z^{-2})} \quad (3.6.6)$$

which may be rewritten in general as

$$E_\eta(z) = \frac{E_\xi(z)}{B(z^{-1})} \quad (3.6.7)$$

where $B(z^{-1})$ is the denominator of the filter transfer function.

The variance of the random output sequence ε_η may be related to the variance of the equivalent random input sequence ε_ξ by[10]

$$\sigma_\eta^2 = \sigma_\xi^2 \left(\frac{1}{2\pi j} \cdot \oint_{|z|=1} \frac{dz}{B(z).B(z^{-1}).z} \right) \quad (3.6.8)$$

where, in this example, $B(z^{-1}) = 1 + b_1 z^{-1} + b_2 z^{-2}$ so that $B(z) = 1 + b_1 z + b_2 z^2$. Equation (3.6.8) may be evaluated by contour integration or by numerical methods[59] for any particular filter coefficients.

In general for a filter of any order the variance of the equivalent input error sequence is given by

$$\sigma_\xi^2 = \sigma_\varepsilon^2 (m + p) \qquad (3.6.9)$$

where m is the number of non-unity coefficients defining the zero positions of H(z) and p is the number of non-unity coefficients defining the positions of the poles. Equation (3.6.8) may be applied to any direct realisation filter by substituting the appropriate σ_ξ^2 and $B(z^{-1})$. It will be seen from this equation that for given m and p the variance of the output error sequence is dependent only on the pole positions of H(z).

3.6.2 Canonic realisation in fixed-point arithmetic.

A similar form of analysis can be applied to the canonic realisation of a filter, as exemplified by the practical second order filter shown in Figure 3.9. To prevent overflow in the intermediate stage of the filter, it is usually necessary to scale-down the input sequence. This is performed by the multiplier a_a in Figure 3.9. Rescaling is effected by appropriately increasing the coefficients a_0, a_1 and a_2. The equivalent noise model is shown in Figure 3.10 where the operation marked H(z) indicates the ideal filtering process. From Figure 3.9 the variance of the sequence $\{\varepsilon_{\xi,n}\}$ is given by

$$\sigma_\xi^2 = (p + 1).\sigma_\varepsilon^2 \qquad (3.6.10)$$

and the variance of the sequence $\{\varepsilon_{o,n}\}$ by

$$\sigma_o^2 = m.\sigma_\varepsilon^2 \qquad (3.6.11)$$

where m and p have the same significance as before. Following equation (3.6.8) the variance of the output error sequence $\{\varepsilon_{\eta,n}\}$ is given by

Figure 3 9 An equivalent circuit for a practical canonic realisation of a second order filter.

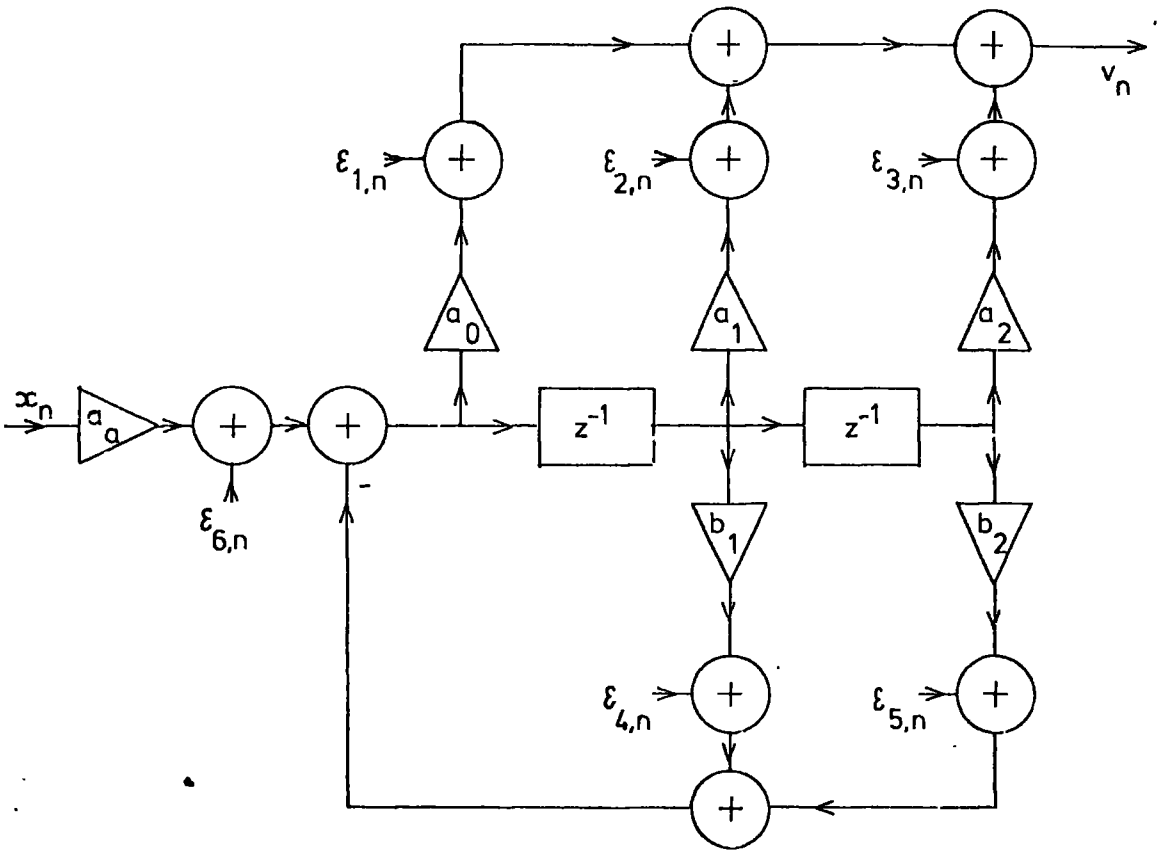
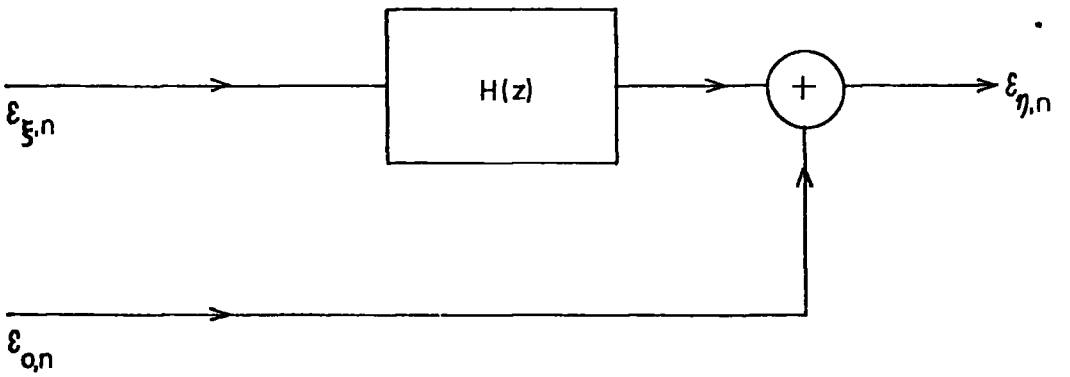


Figure 3.10 Equivalent noise model for a canonic filter.



$$\sigma_\eta^2 = \sigma_o^2 + \sigma_\xi^2 \left(\frac{1}{2\pi j} \cdot \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z} \right) \qquad (3.6.12)$$

It ~~should be~~ is inferred from this equation that, for given m and p, both the zero and pole positions of H(z) affect the variance of the output error sequence.

3.6.3 Filter realisations in block-floating-point arithmetic.

Oppenheim[16] has presented a roundoff noise analysis for a direct realisation of a filter with two poles but no zeros, employing one form of block-floating-point arithmetic. This analysis is extended below to both direct and canonic realisations of a general filter of any order.

3.6.3.1 Direct realisation.

Figure 3.11 shows a circuit flow-diagram of a second order filter using block-floating-point arithmetic. The difference equation describing the filter can be seen to be

$$y_n = \frac{1}{A_n} \left(a_0.\delta_n.\hat{x}_n + a_1.\delta_n.\hat{u}_{n-1} + a_2.\delta_n.\hat{u}_{n-2} - b_1.\delta_n.\hat{w}_{n-1} - b_2.\delta_n.\hat{w}_{n-2} \right) \qquad (3.6.13)$$

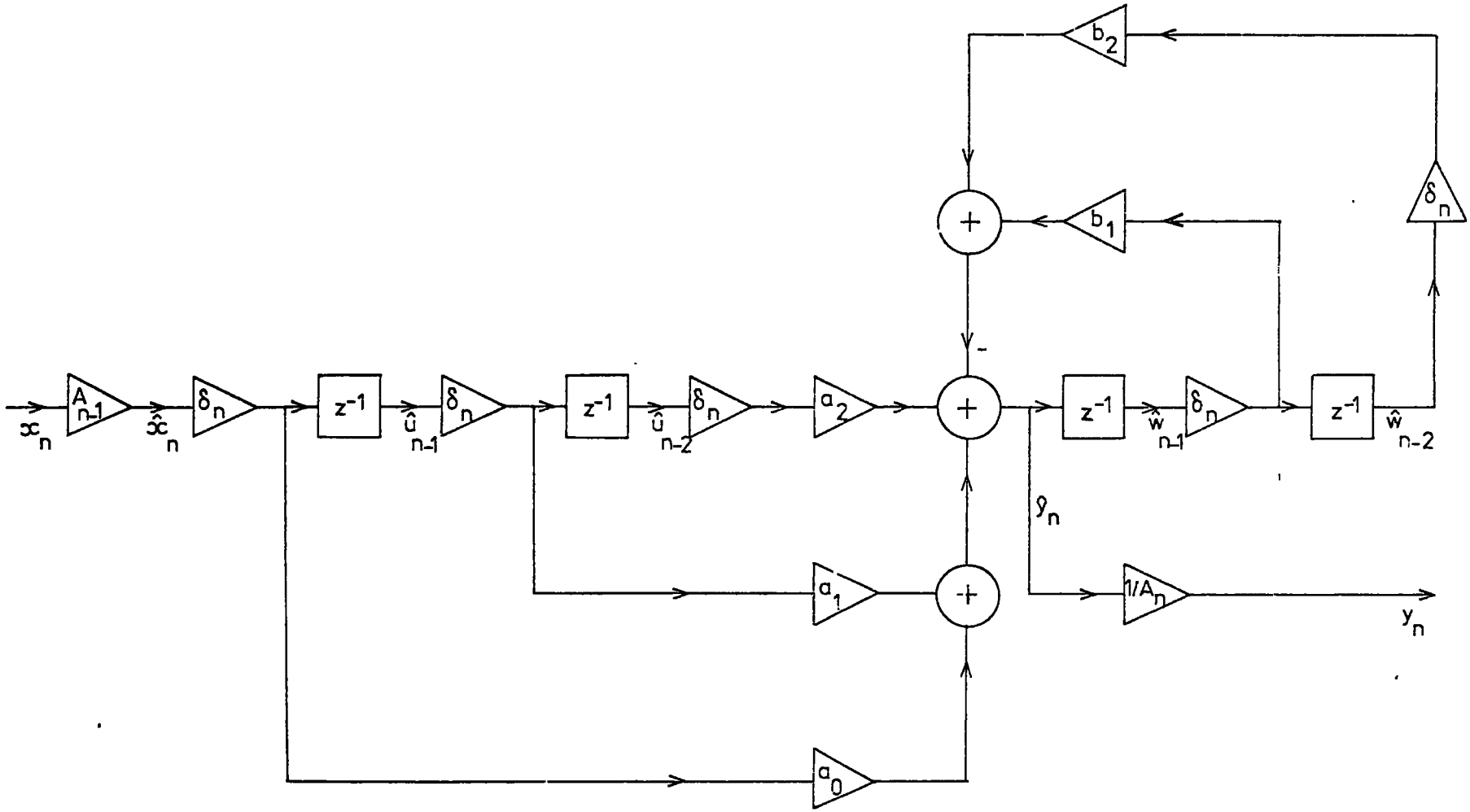
where A_n is the current scaling factor and is a power of 2. This scaling factor is related to the one used during the previous cycle of the filter by

$$A_n = \delta_n.A_{n-1} \qquad (3.6.14)$$

The five data items which are used to compute the output are

$$\hat{x}_n = A_{n-1}.x_n \qquad (a) \qquad (3.6.15)$$

Figure 3.11 A direct realisation of a second order filter using block-floating-point arithmetic.



$$\hat{u}_{n-1} = A_{n-1} \cdot x_{n-1} \qquad \text{(b) (cont.)}$$

$$\hat{u}_{n-2} = A_{n-1} \cdot x_{n-2} \qquad \text{(c)}$$

$$\hat{w}_{n-1} = A_{n-1} \cdot y_{n-1} \qquad \text{(d)}$$

$$\hat{w}_{n-2} = A_{n-1} \cdot y_{n-2} \qquad \text{(e)}$$

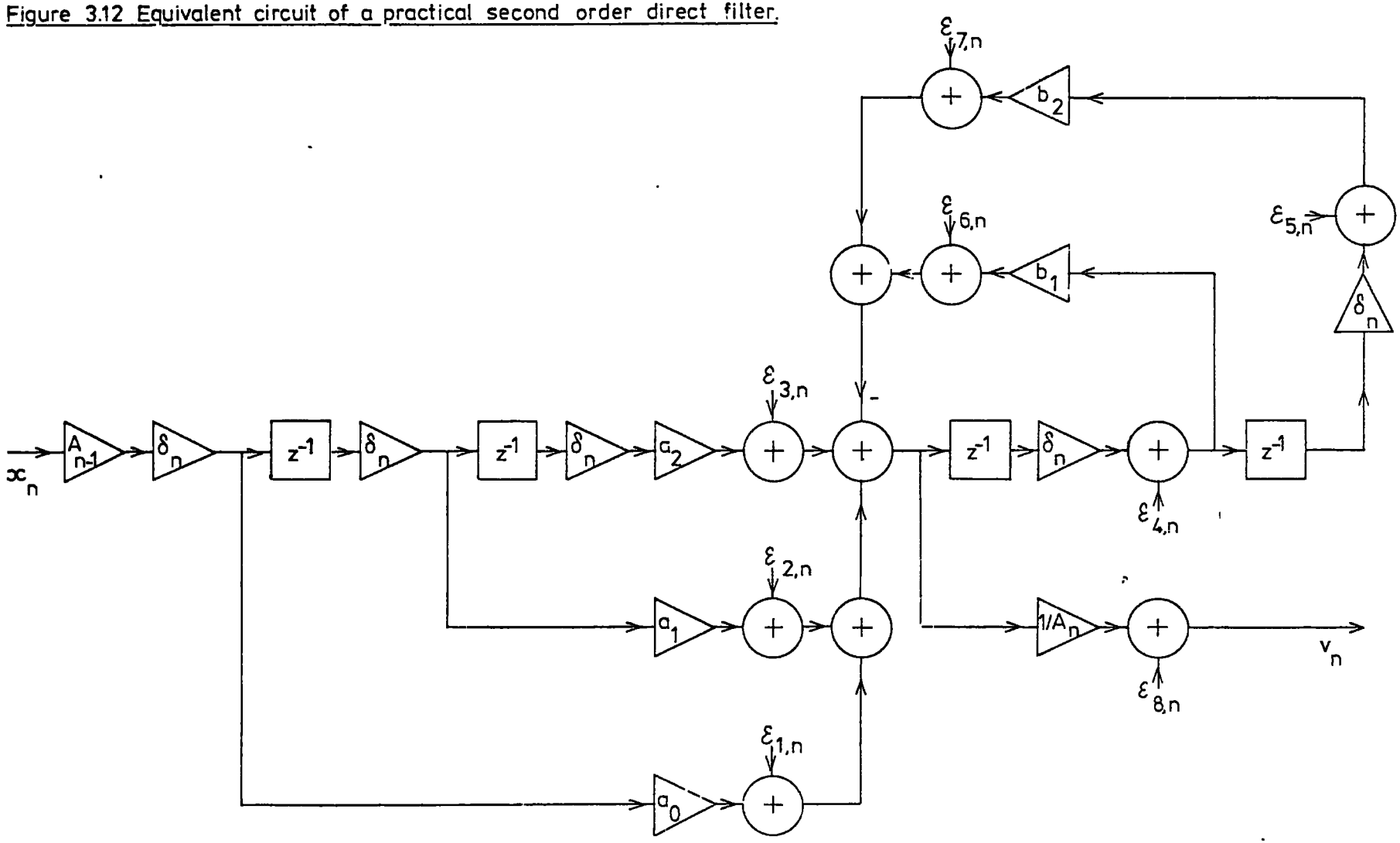
These data items are then rescaled by the factor δ_n, which may be a positive or negative power of two, so that the overall scaling factor becomes A_n. To make best use of this mode of arithmetic the scaling factor should be kept as large as possible without causing data items to exceed the storage capability of the digital device in use. Oppenheim discusses the relative advantages of several variant methods of determining the most appropriate value of the factor δ_n.

In order to produce a noise analysis which involves only apparently reasonable assumptions Oppenheim has imposed the constraint on the scaling factor A_n that it must be a non-negative power of two. It is possible to design a system without this limitation, so that A_n could scale either up or down. However, the theoretical noise model could not be applied to such a system with any reasonable degree of validity. The consequences of this constraint on A_n are that there is no error in the actual values of $\delta_n.x_n$, $\delta_n.u_{n-1}$ or $\delta_n.u_{n-2}$ as $|\hat{x}_n| \geq |x_n| \leq |\delta_n.\hat{x}_n|$, $|\hat{u}_{n-1}| \geq |x_{n-1}| \leq |\delta_n.\hat{u}_{n-1}|$ and $|\hat{u}_{n-2}| > |x_{n-2}| \leq |\delta_n.\hat{u}_{n-2}|$, and that $|y_n| \leq |\hat{y}_n|$, so that normal roundoff occurs at the final rescaling of the output.

Figure 3.12 shows the equivalent circuit for a practical filter with error sources $\varepsilon_1 \ldots . \varepsilon_8$ marked. For simplicity these error sources are taken to arise from rounding processes. Hence according to the model, each of these sequences has a variance of $\Delta^2/12$, which is termed σ_ε^2, and is uncorrelated.

Following the kind of discussion used in §3.6.1 for a fixed-point arithmetic direct realisation of a second order filter, an equivalent

Figure 3.12 Equivalent circuit of a practical second order direct filter.



input error sequence ε_ξ can be determined. The variance of this sequence is given by

$$\sigma_\xi^2 = \sigma_\varepsilon^2.k^2.(5 + b_1^2 + 2b_2^2) \qquad (3.6.16)$$

where k^2 is the mean value of $(1/A_n)^2$ and the mean value of $(\delta_n)^2$ has been assumed to be unity. Following equation (3.6.8) and adding in the effect of the error sequence ε_8, the variance of the output error sequence ε_η may be stated as

$$\sigma_\eta^2 = \sigma_\varepsilon^2 + \sigma_\xi^2 \left(\frac{1}{2\pi j} \cdot \oint_{|z|=1} \frac{dz}{B(z).B(z^{-1}).z} \right) \qquad (3.6.17)$$

where $B(z^{-1}) = 1 + b_1 z^{-1} + b_2 z^{-2}$ for the second order case.

Equation (3.6.17) may be applied to a filter of any order by substituting appropriate expressions for σ_ξ^2 and $B(z^{-1})$. The variance of the appropriate input error sequence ε_ξ for a filter with m non-unity coefficients defining the zero positions and p non-unity coefficients defining the pole positions of the filter transfer function, is given by

$$\sigma_\xi^2 = \sigma_\varepsilon^2.k^2.\left((m + p) + \sum_{i=1}^{p} i b_i^2 \right) \qquad (3.6.18)$$

The value of the factor k^2 is dependent both on the input signal and on the filter characteristics, and it would appear that its only source lies in experimental measurement, a fact which restricts the usefulness and significance of this model.

3.6.3.2 Canonic realisation.

Figure 3.13 shows the equivalent circuit for a practical canonic realisation of a second order filter employing block-floating-point arithmetic. The equivalent noise model for this circuit is shown in

Figure 3.13 Equivalent circuit of a practical canonic realisation of a second order filter.

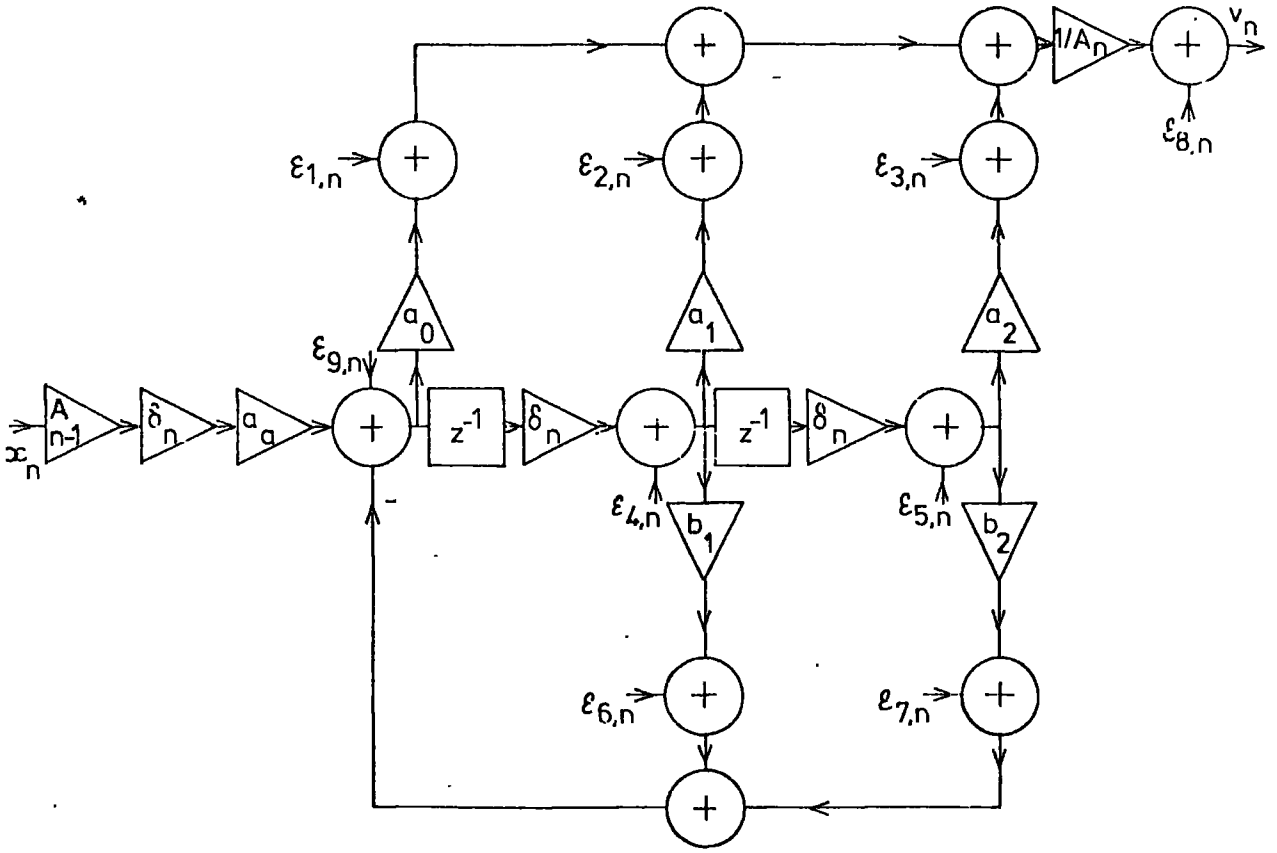


Figure 3.14 An equivalent noise model for a canonic realisation of a second order filter using block-floating-point arithmetic.

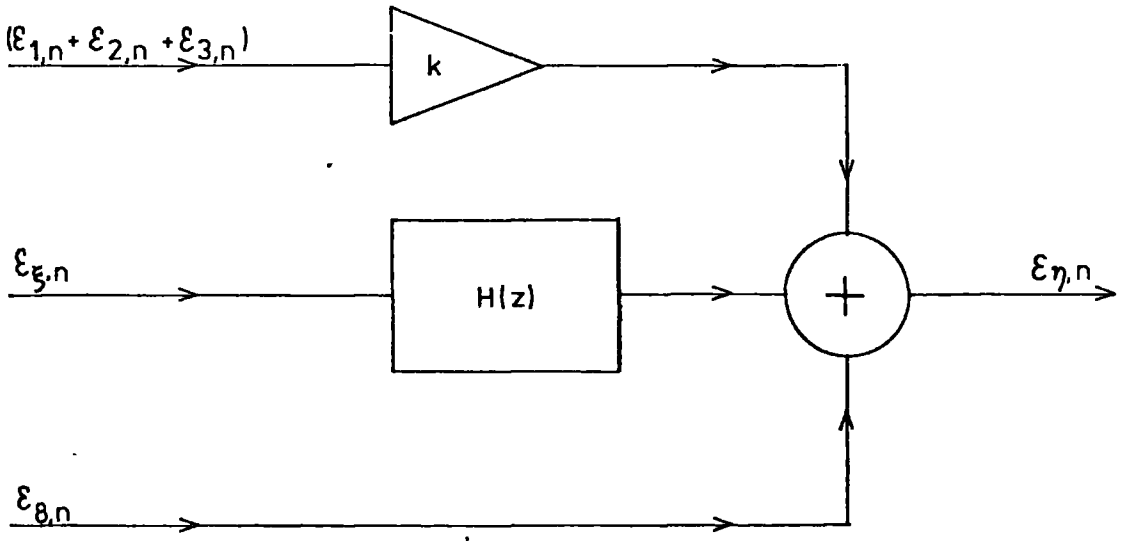


Figure 3.14. The equivalent input error sequence ε_ξ has variance given by

$$\sigma_\xi^2 = \sigma_\varepsilon^2.k^2.(3 + b_1^2 + 2b_2^2) \qquad (3.6.19)$$

From Figure 3.13 the variance of the output error sequence ε_η is given by

$$\sigma_\eta^2 = \sigma_\varepsilon^2.(1 + 3k^2) + \sigma_\xi^2.\left(\frac{1}{2\pi j} \cdot \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z}\right) \qquad (3.6.20)$$

Hence in general for a filter of any order

$$\sigma_\eta^2 = \sigma_\varepsilon^2.(1 + mk^2) + \sigma_\xi^2.\left(\frac{1}{2\pi j} \cdot \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z}\right) \qquad (3.6.21)$$

where

$$\sigma_\xi^2 = \sigma_\varepsilon^2.k^2.\left((p + 1) + \sum_{i=1}^{p} ib_i^2\right) \qquad (3.6.22)$$

and the integers m and p have the same significance as before.

3.7 Filter realisation by a look-up table.

Peled and Liu[60] have proposed a filter realisation which does not comprise distinct multipliers, but which uses the current input sample to the filter together with other delayed signal samples to address locations in a table, which in practice would probably be a read-only memory. The contents of the addressed locations in the table are combined to form the current output sample of the filter. This realisation provides an interesting alternative to the conventional forms. This design philosophy has recently been applied to the fast Fourier transform and to general signal processors[61,62].

The realisation described below may be termed a direct look-up table realisation as it is developed directly from a single difference

equation. It is possible to conceive of a canonic look-up table realisation developed from a pair of difference equations. However, this would require two separate look-up tables and would be much more complex than the direct version. Hence this possible form has not been pursued further.

Consider the example of a second order difference equation

$$y_n = a_0x_n + a_1x_{n-1} + a_2x_{n-2} - b_1y_{n-1} - b_2y_{n-2} \qquad (3.7.1)$$

Assuming that all signal samples are represented in two's complement fixed-point arithmetic, having quantisation width Δ between allowed numbers, and that the wordlength is (B+1) bits, then, for example, following equation (3.2.1), x_n may be written

$$x_n = \left(-x_{n,B}.2^B + \sum_{j=0}^{B-1} x_{n,j}.2^j \right).\Delta \qquad (3.7.2)$$

Hence equation (3.7.1) may be rewritten

$$\begin{aligned} y_n = {} & -2^B.\Delta.(a_0x_{n,B} + a_1x_{n-1,B} + a_2x_{n-2,B} \\ & - b_1y_{n-1,B} - b_2y_{n-2,B}) \\ & + \Delta \sum_{j=0}^{B-1} 2^j.(a_0x_{n,j} + a_1x_{n-1,j} + a_2x_{n-2,j} \\ & - b_1y_{n-1,j} - b_2y_{n-2,j}) \end{aligned} \qquad (3.7.3)$$

Defining a function ψ of five binary arguments r_1, r_2, r_3, r_4, r_5, as

$$\psi(r_1,r_2,r_3,r_4,r_5) = \Delta.2^B(a_0r_1 + a_1r_2 + a_2r_3 - b_1r_4 - b_2r_5) \qquad (3.7.4)$$

where r_1 ... r_5 are each either 0 or 1, equation (3.7.3) may be written

$$y_n = \sum_{j=0}^{B-1} 2^{j-B}.\psi(x_{n,j}, x_{n-1,j}, x_{n-2,j}, y_{n-1,j}, y_{n-2,j}) - \psi(x_{n,B}, x_{n-1,B}, x_{n-2,B}, y_{n-1,B}, y_{n-2,B}) \quad (3.7.5)$$

As the function ψ arising from the difference equation (3.7.1) has five parameters each of which may have one of two values, ψ has $2^5 = 32$ distinct values defined by (3.7.4). Equation (3.7.5) indicates that the filter output y_n may be formed by an additive combination of (B+1) values of ψ. Figure 3.15 shows a block diagram for this realisation, in which the table permanently stores 2^5 distinct values of ψ.

It can be seen from Figure 3.15 that each of the data words x_n, x_{n-1}, x_{n-2}, y_{n-1} and y_{n-2} is converted into a serial stream of length (B+1) bits consisting of logical 1's and 0's. Thus (B+1) distinct addresses are presented to the look-up table during each filter cycle. The first B addressed data words ψ_j are progressively summed and the running total is held in the accumulator. The content of the accumulator is divided by two before each successive addition. The final data word ψ_B is subtracted from the contents of the accumulator and the value of y_n results. (No division by two is performed).

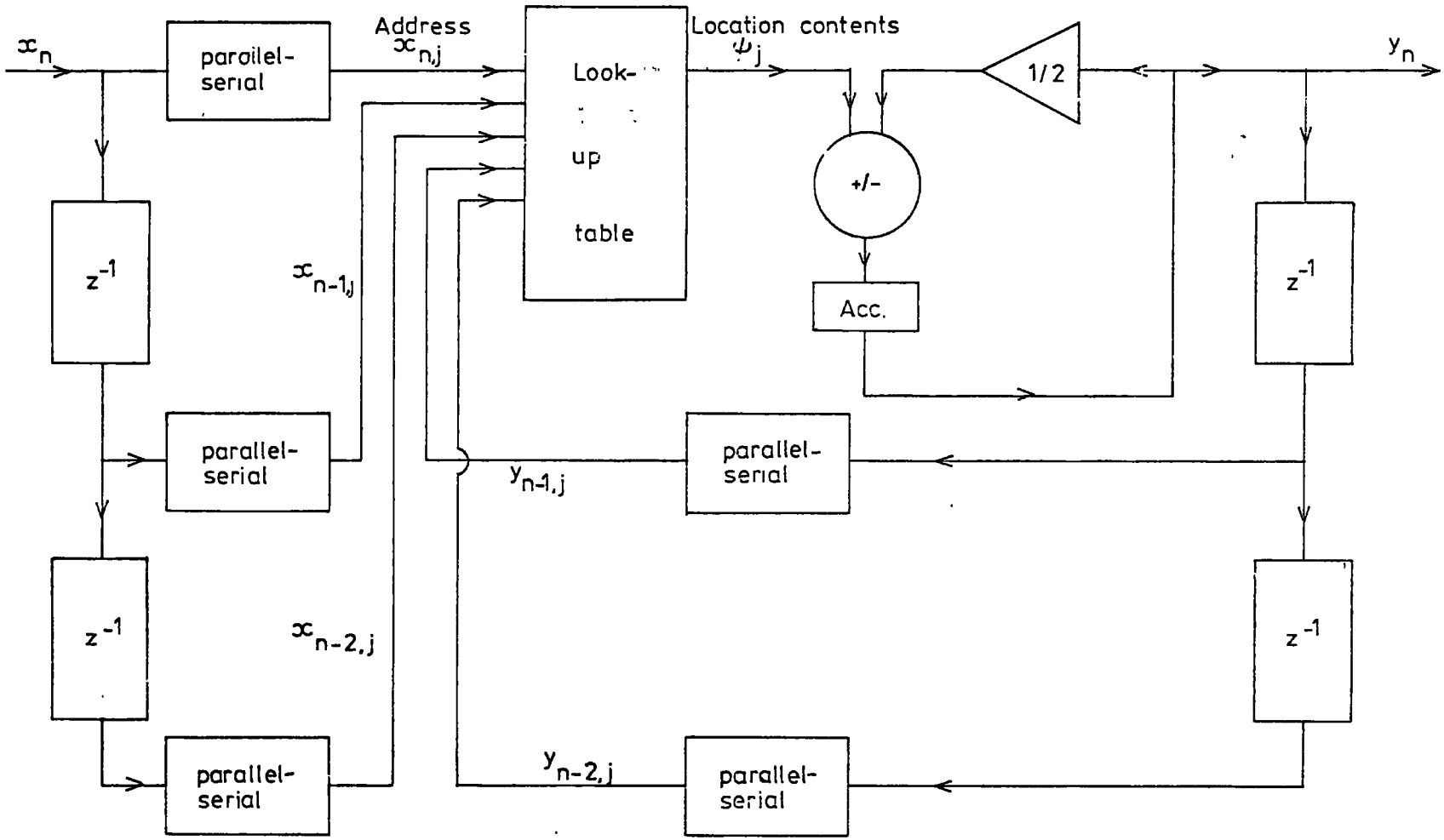
3.7.1 Error analysis.

As the range of values which can be stored in the addressable memory is bounded by $-2^B.\Delta$ and $(2^B-1).\Delta$, it is often necessary to scale down the stored values of ψ to bring them within this interval. Hence the scaled values of ψ, ψ', may be defined by

$$\psi'(r_1,r_2,r_3,r_4,r_5) = 2^{-q}.\psi(r_1,r_2,r_3,r_4,r_5) \quad (3.7.6)$$

so that equation (3.7.5) becomes

Figure 3.15 Filter realisation using a look-up table.



$$y_n = 2^q\Big(\sum_{j=0}^{B-1} 2^{j-B}.\psi'(x_{n,j},\ x_{n-1,j},\ x_{n-2,j},\ y_{n-1,j},\ y_{n-2,j}) - \psi'(x_{n,B},\ x_{n-1,B},\ x_{n-2,B},\ y_{n-1,B},\ y_{n-2,B}) \qquad (3.7.7)$$

ψ' are the theoretical scaled values, but in practice a roundoff error occurs as these are approximated to the actual stored values $\bar{\psi}$, so that

$$\bar{\psi}(r_1,r_2,r_3,r_4,r_5) = \psi'(r_1,r_2,r_3,r_4,r_5) + \varepsilon''_{n,j} \qquad (3.7.8)$$

where $\varepsilon''_{n,j}$ is the roundoff error.

Referring to the equivalent circuit diagram shown in Figure 3.16,

$$w'_n = \sum_{j=0}^{B-1} 2^{j-B}.\bar{\psi}(x_{n,j},\ x_{n-1,j},\ x_{n-2,j},\ v_{n-1,j},\ v_{n-2,j}) - \bar{\psi}(x_{n,B},\ x_{n-1,B},\ x_{n-2,B},\ v_{n-1,B},\ v_{n-2,B}) \qquad (3.7.9)$$

Let w'_n be approximated to $\bar{w}_n$, so that

$$\bar{w}_n = w'_n + \varepsilon_{3,n} \qquad (3.7.10)$$

where $\varepsilon_{3,n}$ is the roundoff error. So on rescaling

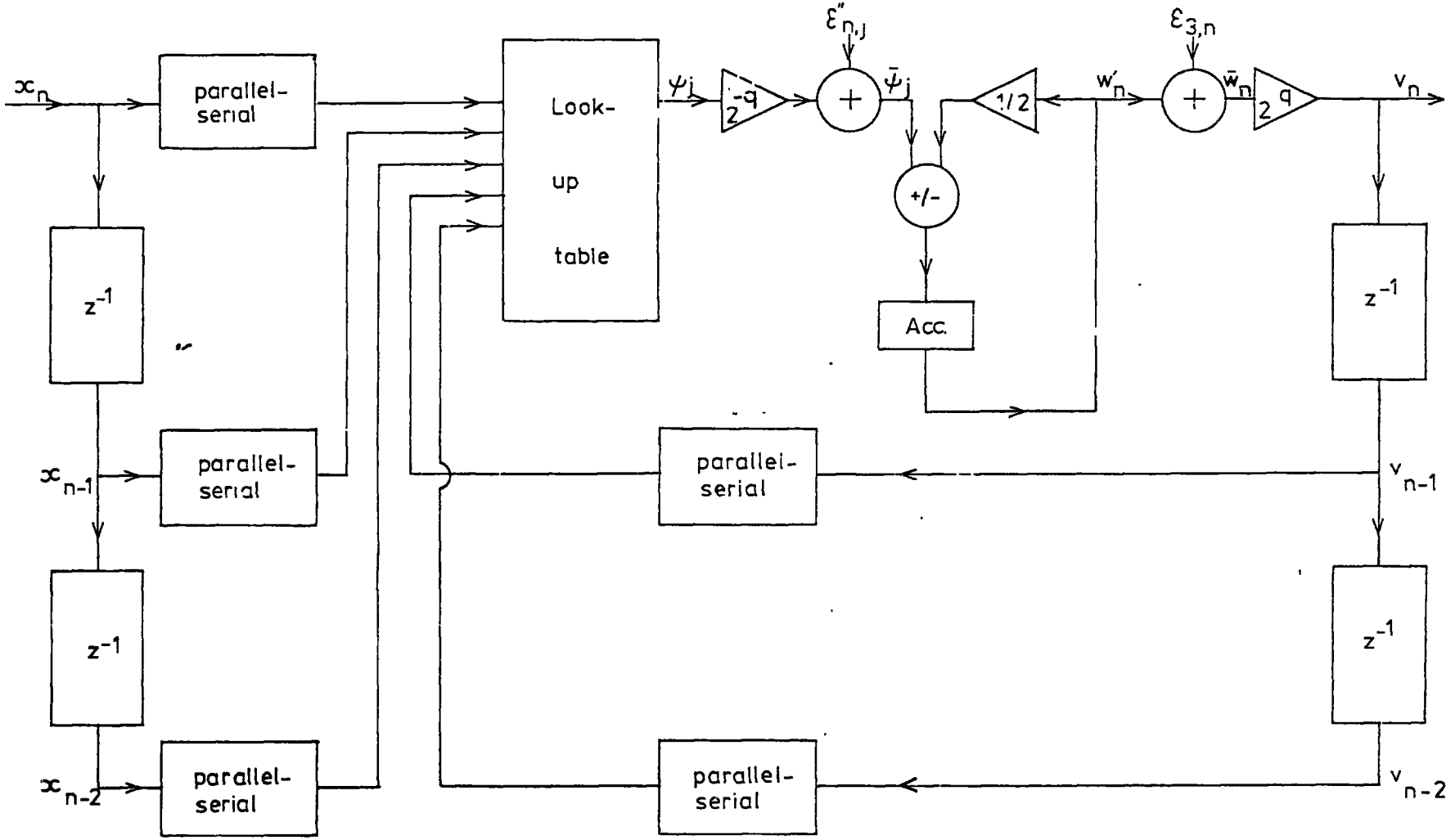
$$v_n = 2^q.\bar{w}_n \qquad (3.7.11)$$

The error at the filter output is therefore given by

$$\varepsilon_{n,n} = v_n - y_n = 2^q(\varepsilon_{3,n} + \sum_{j=0}^{B-1} 2^{j-B}\varepsilon''_{n,j} - \varepsilon''_{n,B}) - b_1\varepsilon_{n,n-1} - b_2\varepsilon_{n,n-2} \qquad (3.7.12)$$

Comparison of this equation with Peled and Liu's expression for the output error will reveal a difference. They have taken account of

Figure 3.16 Equivalent circuit of a practical look-up table filter.



quantisation errors in the input sequence $\{x_n\}$, which is not strictly necessary in an error analysis of the filter alone. They also appear to have omitted the effect of rescaling the filter output by 2^q from their analysis.

Following the assumptions set out in §3.5 and taking the error processes to be rounding, the error sequences $\{\varepsilon''_{n,j}\}$ and $\{\varepsilon_{3,n}\}$ are considered to have variances equal to σ_ε^2 $(\Delta^2/12)$. An equivalent input error sequence can be defined by

$$\varepsilon_{\xi,n} = 2^q(\varepsilon_{3,n} + \sum_{j=0}^{B-1} 2^{j-B}.\varepsilon''_{n,j} - \varepsilon''_{n,B}) \qquad (3.7.13)$$

whose variance is given by

$$\sigma_\eta^2 = 2^{2q}.\sigma_\varepsilon^2\left(\sum_{j=0}^{B} 2^{j-B} + 1\right) \qquad (3.7.14)$$

As in the case of a normal direct realisation the variance of the output error sequence $\{\varepsilon_{\eta,n}\}$ is given by

$$\sigma_\eta^2 = \sigma_\xi^2.\left(\frac{1}{2\pi j}.\oint_{|z|=1} \frac{dz}{B(z).B(z^{-1}).z}\right) \qquad (3.7.15)$$

The variance of the equivalent input error sequence $\{\varepsilon_{\xi,n}\}$ may be reduced by choosing filter coefficients so that no roundoff errors are committed in forming the stored scaled values $\bar{\psi}$ from the theoretical values ψ. If this is the case then $\varepsilon''_{n,j}$ is zero for all allowable values of j. Hence equation (3.7.13) reduces to

$$\varepsilon_{\xi,n} = 2^q.\varepsilon_{3,n} \qquad (3.7.16)$$

indicating that all the noise is generated by the final rounding of the filter output. The variance of $\{\varepsilon_{\xi,n}\}$ is then

$$\sigma_{\xi}^2 = 2^{2q}.\sigma_c^2 \qquad (3.7.17)$$

The variance of the output error sequence may then be found using equation (3.7.15). The variance of the equivalent input noise sequence is independent of the order of the filter, so this analysis may be extended to a filter of any order simply by using the appropriate function for $B(z^{-1})$ in equation (3.7.15).

The experimental results presented in Chapter 5 indicate that a significant reduction in noise can be achieved by this modification. The disadvantage of this form is that there is a finite, discrete set of filter coefficients which can be accurately implemented. The population of this set is reduced as the filter wordlength is decreased. A similar restriction, however, is also imposed in a conventional filter if, realistically, the coefficients are limited to the same wordlength as the signal.

3.8 Summary.

The sources of errors arising in finite wordlength digital filters have been described, and the effect of roundoff errors in particular has been discussed. The assumptions involved in the formulation of Knowles and Edwards'[29] theoretical model for error analysis have been stated and error analyses performed on several digital filter realisations. This includes consideration and correction of the analysis for the look-up table form proposed by Peled and Liu[60]. The results of these analyses must be tested experimentally in order to assess the validity of the theoretical model, particularly as the effective wordlength is decreased below 8 bits.

4.1 Introduction.

One of the assumptions of the model for error analysis is that the roundoff error sequences are truly random and may therefore be regarded as white noise[29]. The validity of this should clearly be tested. One method which may be used is the spectral analysis of error sequences.

A signal, which is normally considered as a real function of time, can equally well be regarded as a superposition of sinusoids of different frequencies. Spectral analysis yields the amplitude and relative phase of each component sinusoid as a function of frequency. Hence spectral analysis is the transformation of the signal from the time-domain to the frequency-domain.

When a white noise sequence undergoes spectral analysis the amplitude of the resulting signal should be constant and therefore independent of frequency. Any variation of amplitude with frequency indicates that the time-domain signal is not truly random.

First of all, the theory of spectral analysis as applied to discrete sequences must be discussed, and then the results of the use of this technique on some roundoff error sequences will be presented.

4.2 The Fourier transform.

The mathematical tool used to transform a signal from the time-domain into the frequency-domain is the Fourier transform. Hence spectral analysis of a signal is also called Fourier analysis. The Fourier transform of a time-domain signal f(t) is defined as[10]

$$F(\omega) = \mathcal{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t).e^{-j\omega t}dt. \qquad (4.2.1)$$

$F(\omega)$ is the frequency-domain function and both this and f(t) may be

complex functions of a real variable. The ability of the Fourier transform to analyse a signal into its component sinusoids can be expressed mathematically thus[10]

$$\mathcal{F}\{\sum_i a_i.e^{j\omega_i t}\} = 2\pi.\sum_i a_i.\delta(\omega - \omega_i) \qquad (4.2.2)$$

where $\delta(\omega - \omega_i)$ denotes a unit impulse at $\omega=\omega_i$. That is, a signal composed of several overlapping sine waves in the time-domain is transformed into a sum of impulses in the frequency-domain which are by definition non-overlapping.

The inverse Fourier transform used for transformation from the frequency-domain to the time-domain is defined by[10]

$$f(t) = \mathcal{F}^{-1}\{F(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega).e^{jt\omega}d\omega. \qquad (4.2.3)$$

The similarity of form between the Fourier transform and its inverse should be noted as this fact increases the usefulness of the tool.

4.2.1 The discrete Fourier transform.

Naturally the transformation defined above, which is the continuous Fourier transform (CFT), cannot be applied to the discrete-time signals with which this work is concerned. However, it is possible to define an analogous transformation (and its inverse) which has very similar properties of spectral analysis. If the discrete-function in the time-domain is represented by f(nT), where n is an integer and T is the duration between consecutive samples of the signal, the discrete Fourier transform (DFT) is defined by

$$F(\omega) = \sum_{n=-\infty}^{\infty} f(nT).e^{-j\omega nT} \qquad (4.2.4)$$

which follows directly from equation (4.2.1) assuming that f(t)=0 for t≠nT where n is an integer. However, practical use of this transformation dictates a finite sequence length for f(nT). If f(nT) is a sequence of N samples then its DFT is defined by[10]

$$F(k.\Delta\omega) = \sum_{n=0}^{N-1} f(nT).e^{-jk\Delta\omega nT} \qquad (4.2.5)$$

where $F(k.\Delta\omega)$ is a discrete function in the frequency-domain, k is an integer, and $\Delta\omega$ is the separation in frequency between two consecutive samples in the frequency-domain.

$$\Delta\omega = \frac{2\pi}{NT} \qquad (4.2.6)$$

It is clear from equation (4.2.5) that the frequency function $F(k.\Delta\omega)$ is periodic with period $2\pi/T$. Hence there can only be N distinct values of $F(k.\Delta\omega)$ computable, which correspond to integral values of k from 0 to N-1. It is further shown in Appendix 1 that for any real sequence in the time-domain, the DFT is symmetrical about $\omega=(2n-1)\pi/T$ for any integer n. Considering the N distinct values of k=0 to N-1 this point of symmetry is at $\omega=\pi/T$. This is a restatement of the phenomenon of aliasing. In fact the finite DFT may be thought of as an evaluation of the Z-transform of the discrete-time function f(nT) at the N points in the z-plane, all equally spaced along the unit circle at angles of $(k.\Delta\omega)$ radians.

Although the DFT has very similar properties to the continuous Fourier transform, it is important to understand the differences. If the continuous function c(t) has been sampled for a finite duration to yield the discrete-time sequence f(nT), then f(nT) is defined by

$$f(nT) = c(t).g(t).s(t) = h(t) \qquad (4.2.7)$$

where $$g(t) = \begin{cases} 0 \text{ for } t < 0 \text{ and } t \geq NT \\ 1 \text{ for } 0 \leq t < NT \end{cases}$$

and $$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

It is a general property of the Fourier transform that a product of two sequences in the time-domain yields the convolution of the Fourier transforms of the two sequences in the frequency-domain. Hence the Fourier transform of equation (4.2.7) is

$$\mathcal{F}\{h(t)\} = \mathcal{F}\{c(t)\} * \mathcal{F}\{g(t)\} * \mathcal{F}\{s(t)\}$$

that is, $$H(\omega) = C(\omega) * G(\omega) * S(\omega) \qquad (4.2.8)$$

It can be shown that[10]

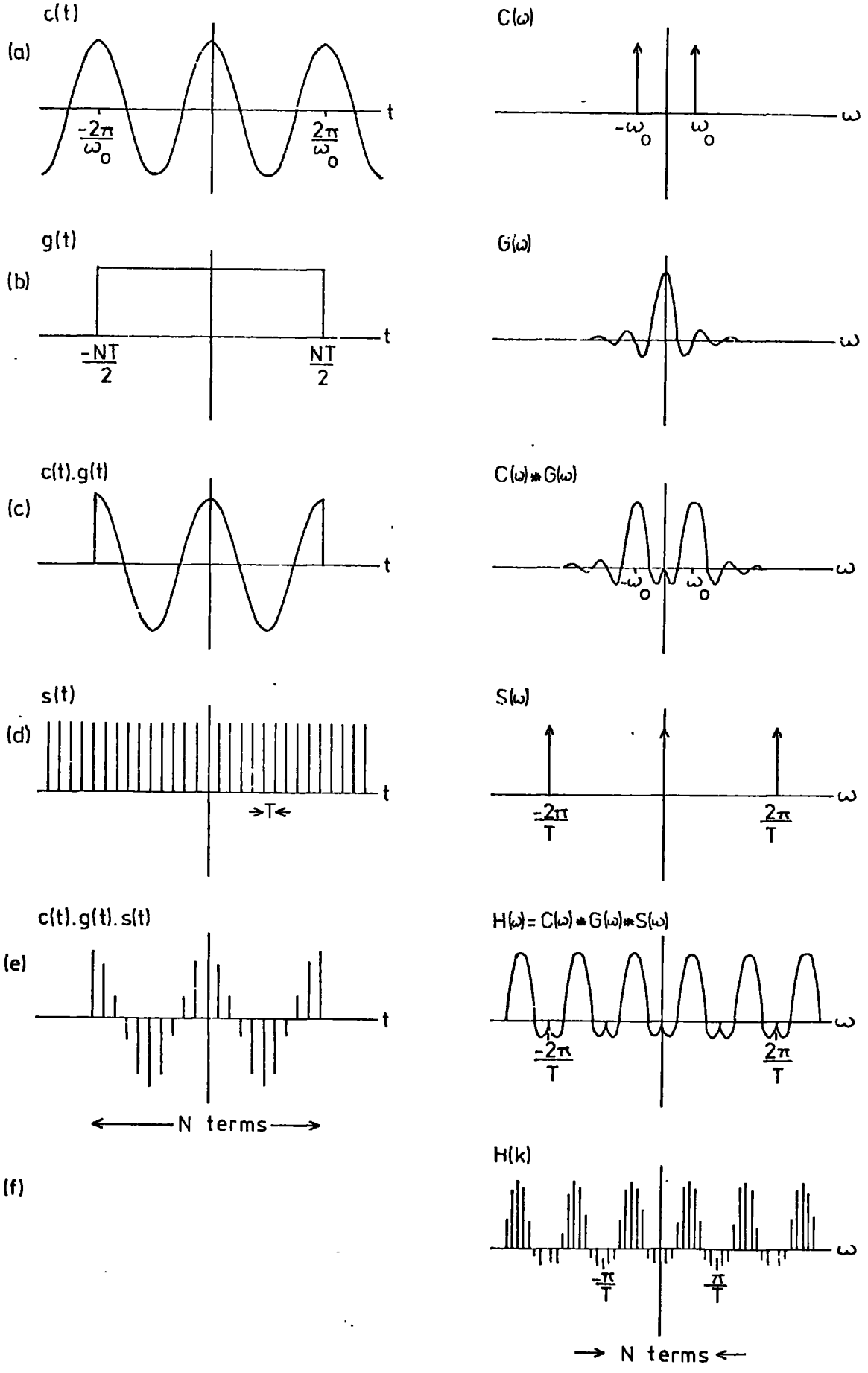
$$G(\omega) = \frac{NT.\sin(\omega NT/2)}{\omega NT/2} \qquad (4.2.9)$$

and that

$$S(\omega) = \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T}) \qquad (4.2.10)$$

The significance of these equations can most easily be demonstrated by an example[1]. Let c(t) be a cosine wave of frequency ω_o. Since c(t) is a real, even function, its Fourier transform is also real and even, and as c(t) is monochromatic C(ω) consists of two impulse functions at $\omega=\pm\omega_o$ (see Figure 4.1(a)). Now consider the effect of making c(t) a finite function by multiplying it with the 'window' function g(t), (see Figure 4.1(b)). Hence the Fourier transform becomes C(ω)*G(ω), whose form is shown in Figure 4.1(c). Next consider sampling this finite continuous signal. The Fourier transform now becomes C(ω)*G(ω)*S(ω) a section of

Figure 4.1 The effect of using the discrete Fourier transform



which is shown in Figure 4.1(e). Finally this continuous Fourier transform must be sampled to give the discrete Fourier transform H(k) defined by

$$H(k) = \sum_{k=-\infty}^{\infty} \delta(\omega - \frac{2\pi k}{NT}) \left(C(\omega) * \{\frac{NT.\sin(\omega NT/2)}{\omega NT/2}\} * \{\frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T})\}\right) \qquad (4.2.11)$$

As H(k) is periodic with period equal to $2\pi/T$ all the information is available in the one cycle $-N/2 \leq k < N/2$. Hence equation (4.2.11) may be simplified to give

$$H(k) = \sum_{k=-N/2}^{N/2-1} \delta(\omega - \frac{2\pi k}{NT}) \left(C(\omega) * \{\frac{NT.\sin(\omega NT/2)}{\omega NT/2}\}\right) \qquad (4.2.12)$$

The important conclusion which must be drawn both from these equations and from Figure 4.1 is that the function g(t) in the time-domain causes a very significant reduction in the resolution of the discrete Fourier transform. What should ideally be an impulse function has become a (sin x)/x curve.

4.2.1.1 Window functions.

The function g(t) above is often termed a rectangular 'window' function. The Fourier transform of this function is the source of the (sin x)/x curve found in the DFT of a finite sequence of a sampled sinusoid. Various modified 'window' functions have been proposed in an attempt to improve the resolution of the discrete Fourier transform. The defining equations of these 'window' functions are set out in Appendix 2.

As shown in Figure 4.1(b) the (sin x)/x curve has one main-lobe and a series of side-lobes which decay in amplitude with frequency deviation from the centre of the main-lobe. All the frequency-domain functions listed in Appendix 2 have the same general features: a main-lobe and a series of lower level side-lobes. A good measure of the spectral resolution permitted by a particular 'window' function is the ratio of

the main-lobe width : side-lobe amplitude of the frequency-domain function.

A decision on the most appropriate 'window' function to use was based on the results of the experiment described in Appendix 3. The convoluted Hamming 'window' suggested by Richards[68] was chosen and has been used for all succeeding power spectra measurements.

4.2.2 The fast Fourier transform.

The direct evaluation of the discrete Fourier transform equation (4.2.5) for k=0 to N-1 requires N^2 complex multiplications and additions[10]. For a sequence length N, say, greater than 1000, this represents an impractical length of computing time. Cooley and Tukey[70] proposed a technique in 1965 which permits much greater efficiency in the evaluation of the DFT. This work was followed by that of Stockham[71] and later by Rader[72] and many others. The many related algorithms which permit the fast evaluation of the DFT are given the generic title of the fast Fourier transform (FFT). Several useful review papers detail the properties of the FFT and consider the most significant algorithms[1,73,74]. These algorithms are at their most efficient when the sequence length N is a power of two[71], and for this reason the value of N used throughout this work was chosen to be 2^{10}=1024. (This is long enough to give reasonable resolution in the frequency-domain when the convoluted Hamming[68] 'window' is used). When N is a power of two the FFT algorithms require only of the order of $N.\log_2 N$ computations rather than N^2, which for N=1024 represents only 1% of the original computing time. The details of the fast Fourier transform algorithm used are presented in Appendix 4.

There is one final property of the fast Fourier transform which needs to be mentioned. Equation (4.2.5) may be rewritten in the form

$$F_k = \sum_{n=0}^{N-1} f_n . W^{nk} \qquad (4.2.13)$$

where

$$W = e^{-j(2\pi/N)} \tag{4.2.14}$$

Excluding normalising factors, the inverse discrete Fourier transform is defined by[74]

$$f_n = \sum_{k=0}^{N-1} F_k . e^{j(2\pi/N)nk} \tag{4.2.15}$$

Hence it is only necessary to redefine the value of W to be

$$W = e^{j(2\pi/N)} \tag{4.2.16}$$

so that the inverse transform can be written

$$f_n = \sum_{k=0}^{N-1} F_k . W^{nk} \tag{4.2.17}$$

As this has exactly the same form as equation (4.2.13) the same algorithm may be used for both the normal and inverse transformations.

4.2.3 Number theoretic transforms.

Agarwal and Burrus[17] have recently reviewed the properties of transformations performed using modular arithmetic; such operations are termed number theoretic transforms (NTT). They propose a transformation which may be regarded as an alternative to the normal FFT. This transformation requires only word shifts and additions, but no multiplications; no storage of complex number data is needed, and there is no generation of roundoff errors[51]. There are also disadvantages encountered in employing this technique[19,75]. There is an interaction between the available data wordlength and the length of sequence which may be transformed. There is also a problem of overflow. Rader[20] has

suggested that these problems are eased by two-dimensional processing. McClellan[21] has recently produced a hardware implementation of one of these transforms. Number theoretic transforms are clearly of the utmost significance for the future of digital signal processing.

4.3 Spectral analysis of error sequences.

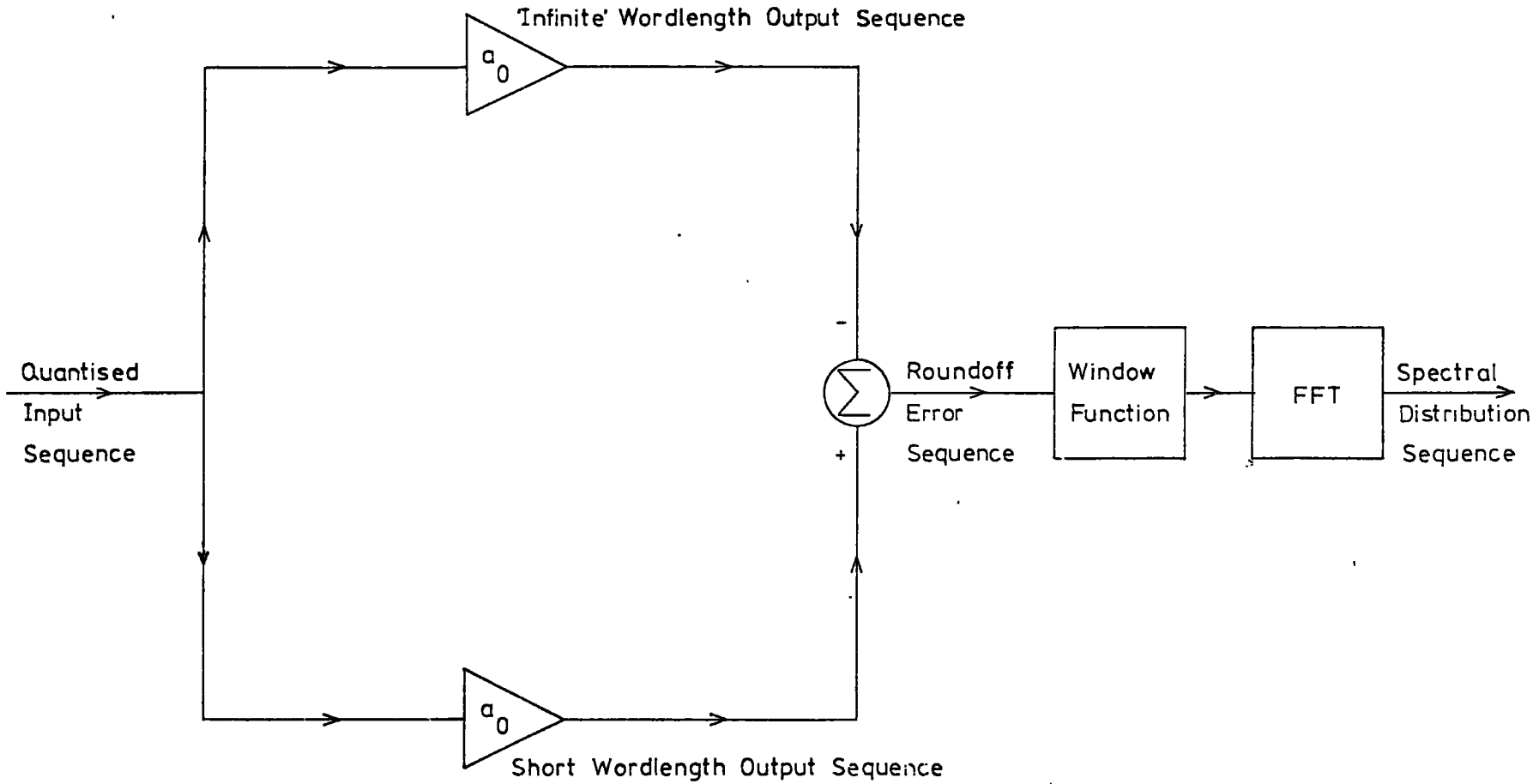
Having discussed the theory of the discrete Fourier transform, chosen a 'window' function to optimise the spectral resolution, and developed a fast Fourier transform algorithm and program, the spectral analysis of some error sequences can now be undertaken.

4.3.1 Experimental technique.

The above consideration of the Fourier transform has tacitly assumed that spectral analysis is to be carried out using a large off-line computer. This is the only feasible approach as the use of a sufficiently powerful on-line computer cannot be countenanced, and purpose-built instrumentation is not to hand. Having decided on this approach, it is a short step to the conclusion that the error sequences should also be produced by the off-line processor. It is possible to conceive of a hybrid approach whereby the error sequences are produced in real-time, stored in some form, then loaded as data into the large computer. However, it is far more advantageous to perform the entire experiment by simulation on the large processor. The location of the boundary between simulation and real experiment is, in any case, rather debatable when the operation of a logical device, such as a microprocessor, is under investigation.

Figure 4.2 depicts the method for performing spectral analysis on the roundoff error sequence produced at a single multiplier. The single multiplier represents either the simplest entire digital filter or the fundamental element of a more complex signal processor. Figure 4.2

Figure 4.2 Spectral analysis of an error sequence.



indicates that the same quantised input sequence is processed by two multipliers in parallel. The quantisation and amplitude of the input sequence are determined by the constraints imposed by the short wordlength and arithmetic mode under consideration. That is, using a wordlength of 8 bits and fixed-point arithmetic, the input sequence is restricted to integers in the range -128 to 127. The two multipliers have the same value, a_0, which is normally less than unity. The difference between the two parallel processing channels occurs in the output sequences. The short wordlength channel has the same quantisation and amplitude restrictions as the input channels. Hence roundoff occurs at the multiplier. The so-called 'infinite' wordlength channel is, however, capable of much greater precision of number representation, such that the output sequence can be considered to be free from roundoff error. In practice, 64-bit floating-point arithmetic (56-bit mantissa, 8-bit exponent) is used in this channel. A sample by sample subtraction of the two output sequences therefore produces a roundoff error sequence, which must also be represented in 64-bit floating-point arithmetic. This sequence is then transmitted through the 'window' function and processed by the fast Fourier transform algorithm.

Any non-uniformity in the spectral distribution of an error sequence will probably be caused by a degree of correlation between the input sequence at a multiplier and the corresponding roundoff error sequence. If any such correlation exists then the frequency spectrum of the error sequence will have some affinity to that of the input sequence. In order to make any resemblance between the two spectral distributions as clear as possible, it is desirable to use an input sequence with a simple, known spectrum. This reasoning leads to the use of a sampled sinusoid as the input sequence.

Some care is required in choosing the frequency of the sinusoid used as the input sequence. The DFT samples the frequency-domain function

at intervals of $2\pi/NT$ and it is clearly desirable to choose one of these allowed frequencies. In other words, the finite sequence must represent an integral number of whole cycles of the input frequency. (1024 is the sequence length used). This, however, is not a sufficient constraint. A 1024 sample sequence representing an even number of whole cycles may be subdivided into two or maybe more identical sequences. This introduces unwanted redundancy into the experiment. In general, redundancy exists if the integral number of cycles represented and the sequence length have a common factor. The sequence length employed, 1024, has the single prime factor 2, so to avoid redundancy any input frequency may be chosen which yields an odd number of whole cycles in this sequence length.

It is always desirable to take more than one set of results for an experiment, so that they can be averaged and the error in the mean determined. If 100 observations are made, then the error in the mean may always be quoted to one significant figure as its fractional error is 10%. This appears to be a reasonable level of precision, and 100 is therefore taken as the standard number of independent observations of a result used throughout the experimental work. This figure would be unrealistically high if the experiments were not performed by computer simulation.

In an experiment simulated on a computer it should always be possible to reproduce a run exactly by setting up the same initial conditions. The problem arises, therefore, of producing numerous sets of results, for the purpose of averaging and error calculation, which are not identical. A solution has been found in varying, from run to run, the phase offset at which the sampling of the sine wave commences. This initial offset is incremented by $\pi/100$ for each of the 100 runs, so if it starts at zero it will be 0.99π on the final run. Hence each of the runs will use a different input sequence.

The results of spectral analysis on some error sequences are

presented in the next section. In all cases rounding, which has the best behaved characteristics of the three roundoff forms, is the error process employed. Various wordlengths, multiplier values and input frequencies are considered, all for fixed-point arithmetic. In order to gain a fair impression of the effect of varying wordlength, the amplitude of the input sequence is set to the maximum permitted by the wordlength. The Fortran IV program used to effect these simulations is listed in Appendix 6; the main program is entitled SPEC.

4.3.2 Experimental results.

Figures 4.3 to 4.12 depict the results of spectral analysis on some roundoff error sequences. The details of coefficient value, signal wordlength and input frequency relative to the sampling frequency are noted on the individual graphs. The graphs are plots of relative amplitude as a function of frequency. The relative amplitude is recorded in decibels, where 0dB refers to the mean amplitude over all frequencies. The frequency axis covers the range from zero to the Nyquist frequency. This axis is graduated in terms of the harmonics of the frequency of the input sequence; the reason for which should be obvious on inspecting the results.

The plots do not carry error bars as these have been omitted for clarity. Calculation of the errors indicates that most points with a relative amplitude of around 25dB have an error of about 0.01dB, while points around -10dB have errors in the region of 1dB. In general, the error is found to increase with decreasing relative amplitude. The magnitude of these errors indicates that all the important structural features depicted in the plots are in no way created by the statistical fluctuations in the results from run to run.

Although the discrete Fourier transform yields a discrete frequency function, the presentation of continuous line plots can be justified.

Figure 4.3 Spectral analysis of an error sequence.

a_0 = 11/1024, Wordlength = 8 bits, Relative input frequency = 9/1024.

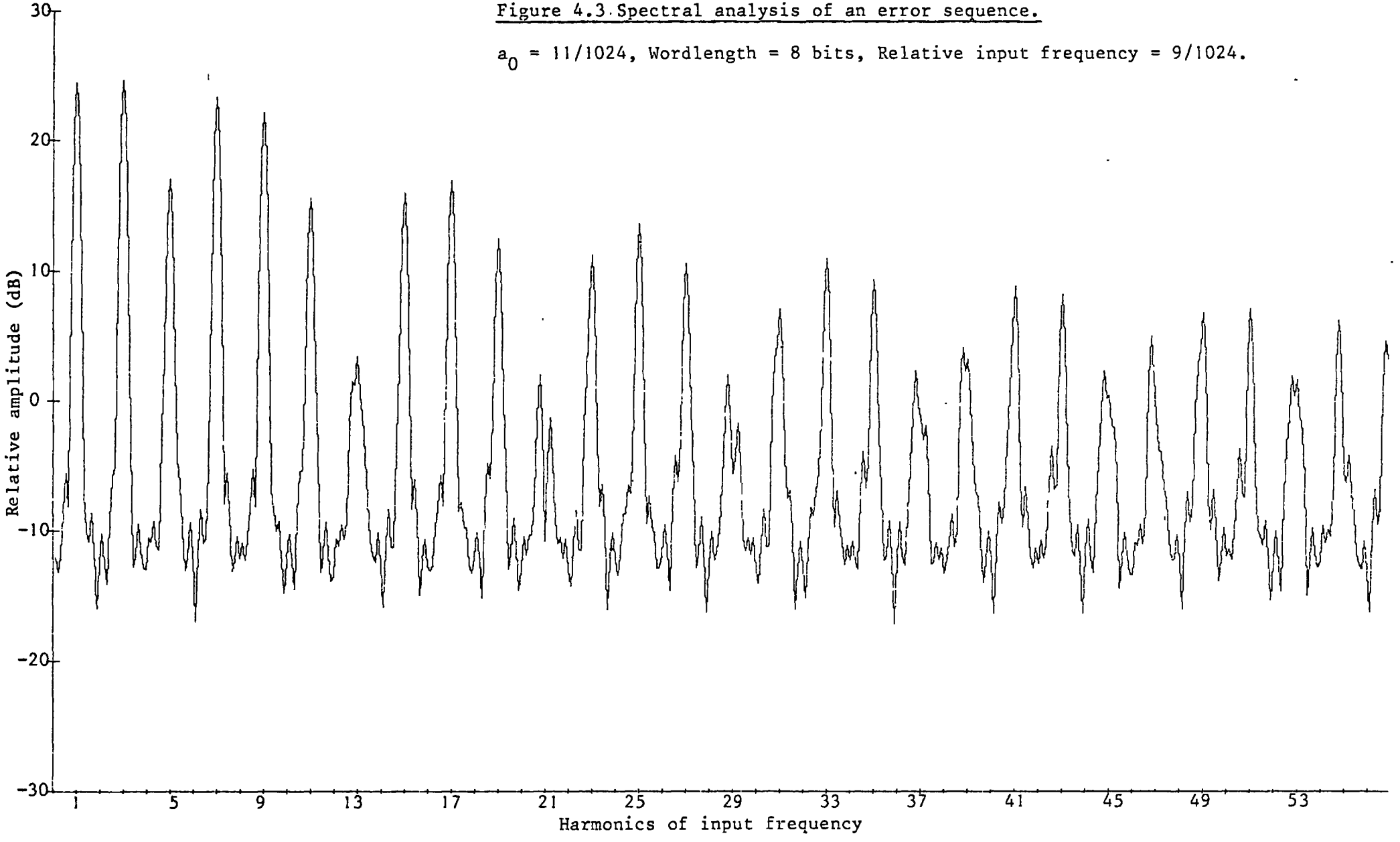


Figure 4.4 Spectral analysis of an error sequence.

a_0 = 111/1024, Wordlength = 4 bits, Relative input frequency = 9/1024.

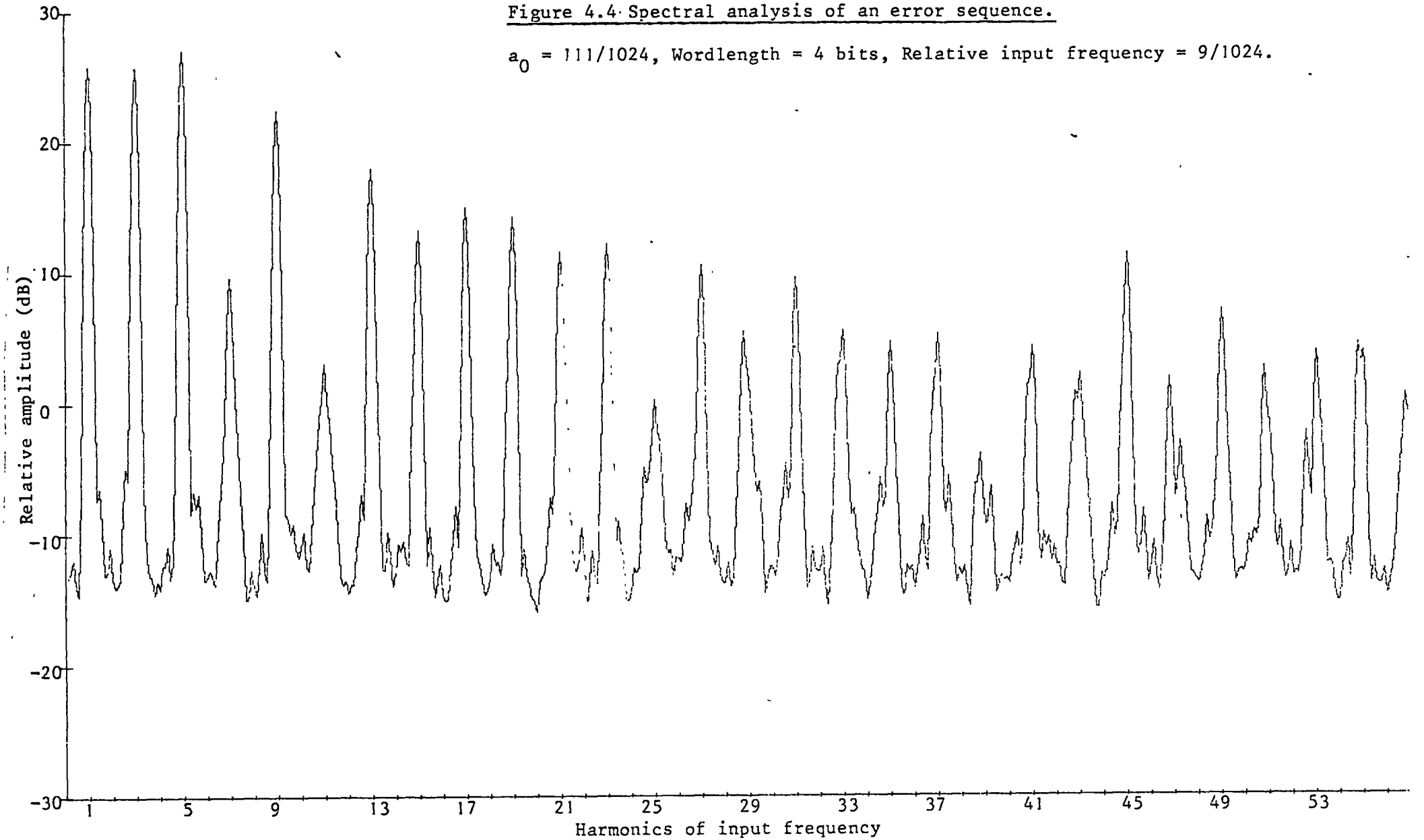


Figure 4.5 Spectral analysis of an error sequence.

a_0 = 111/1024, Wordlength = 5 bits, Relative input frequency = 9/1024.

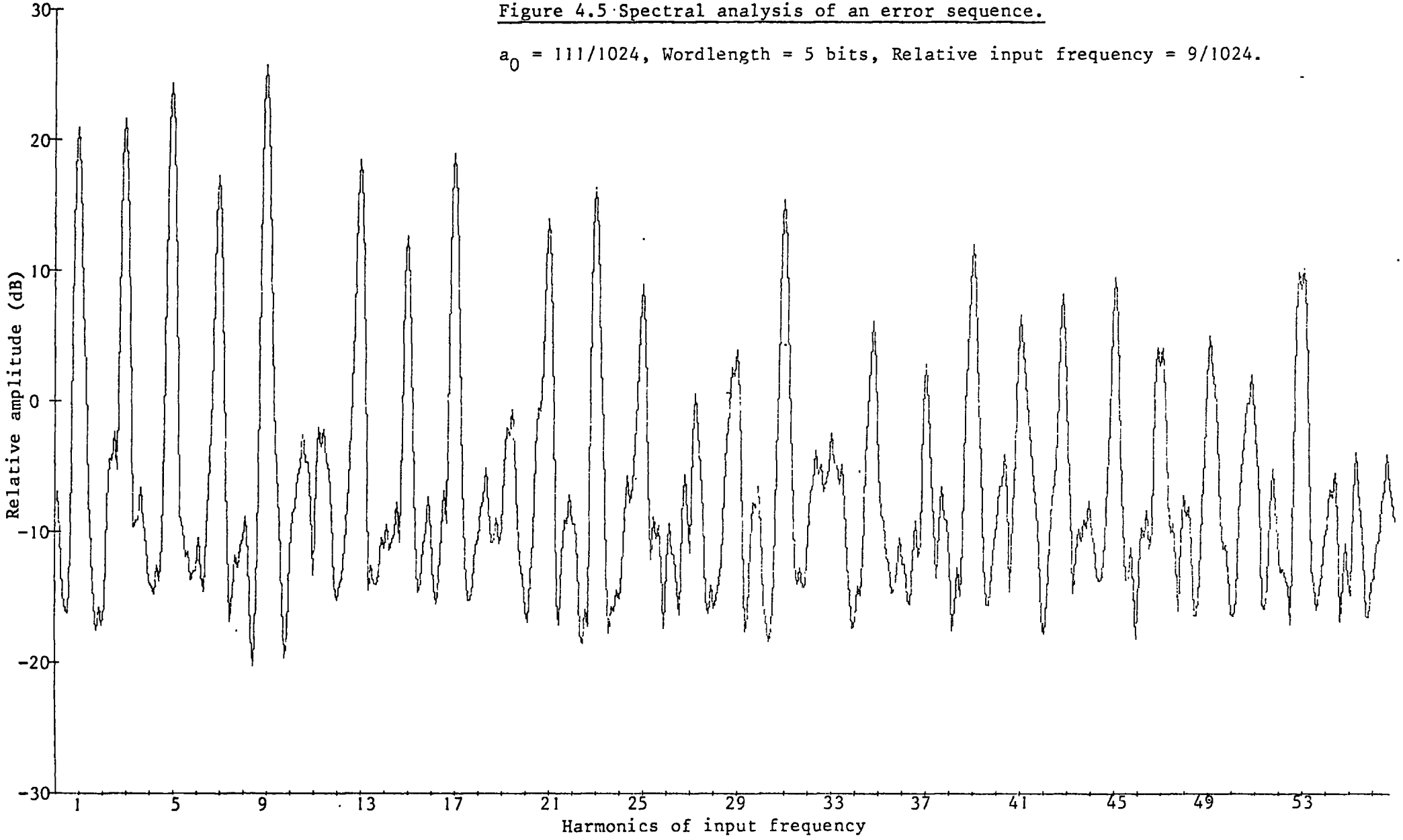


Figure 4.6 Spectral analysis of an error sequence.

a_0 = 111/1024, Wordlength = 6 bits, Relative input frequency = 9/1024.

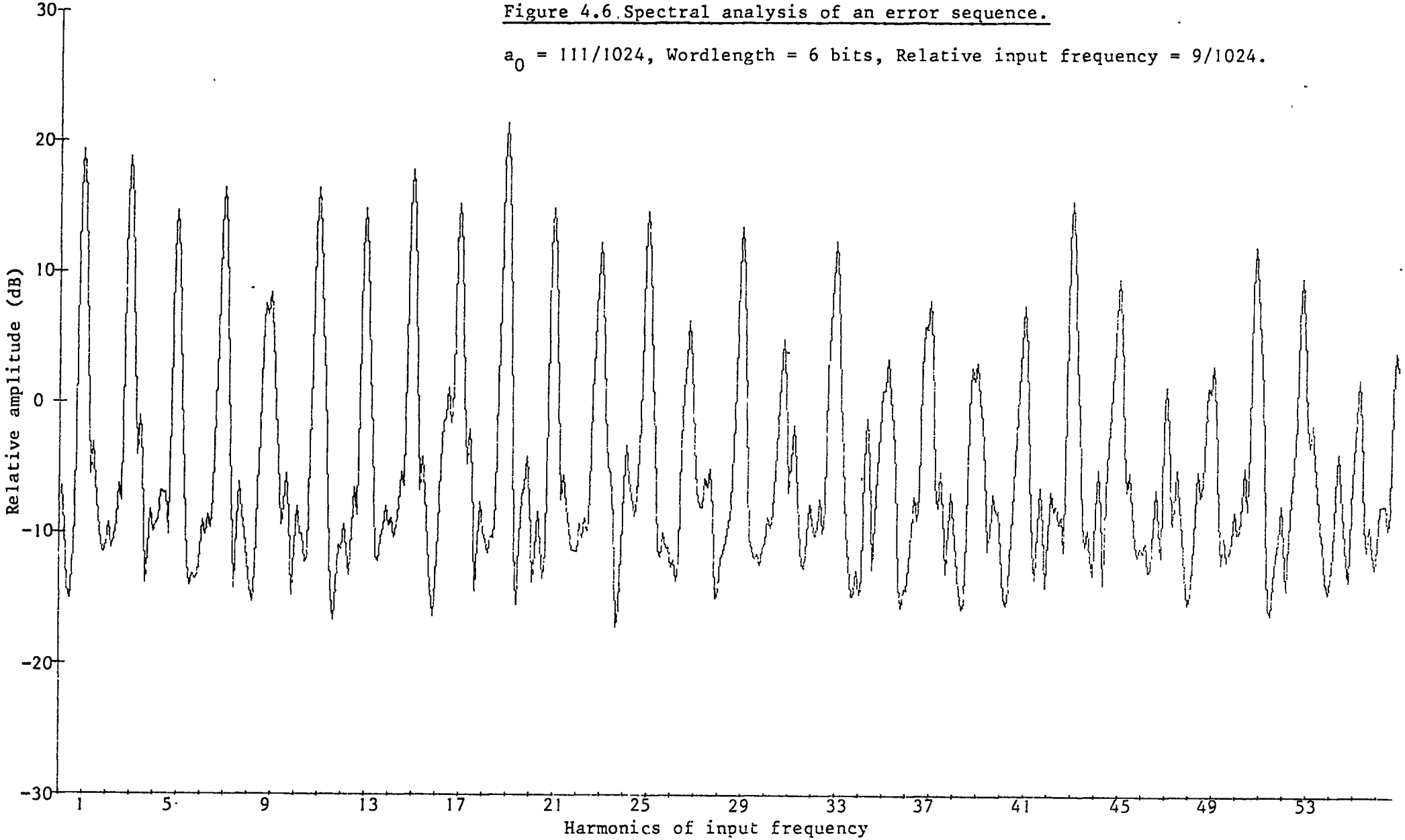


Figure 4.7 Spectral analysis of an error sequence.

a_0 = 111/1024, Wordlength = 7 bits, Relative input frequency = 9/1024.

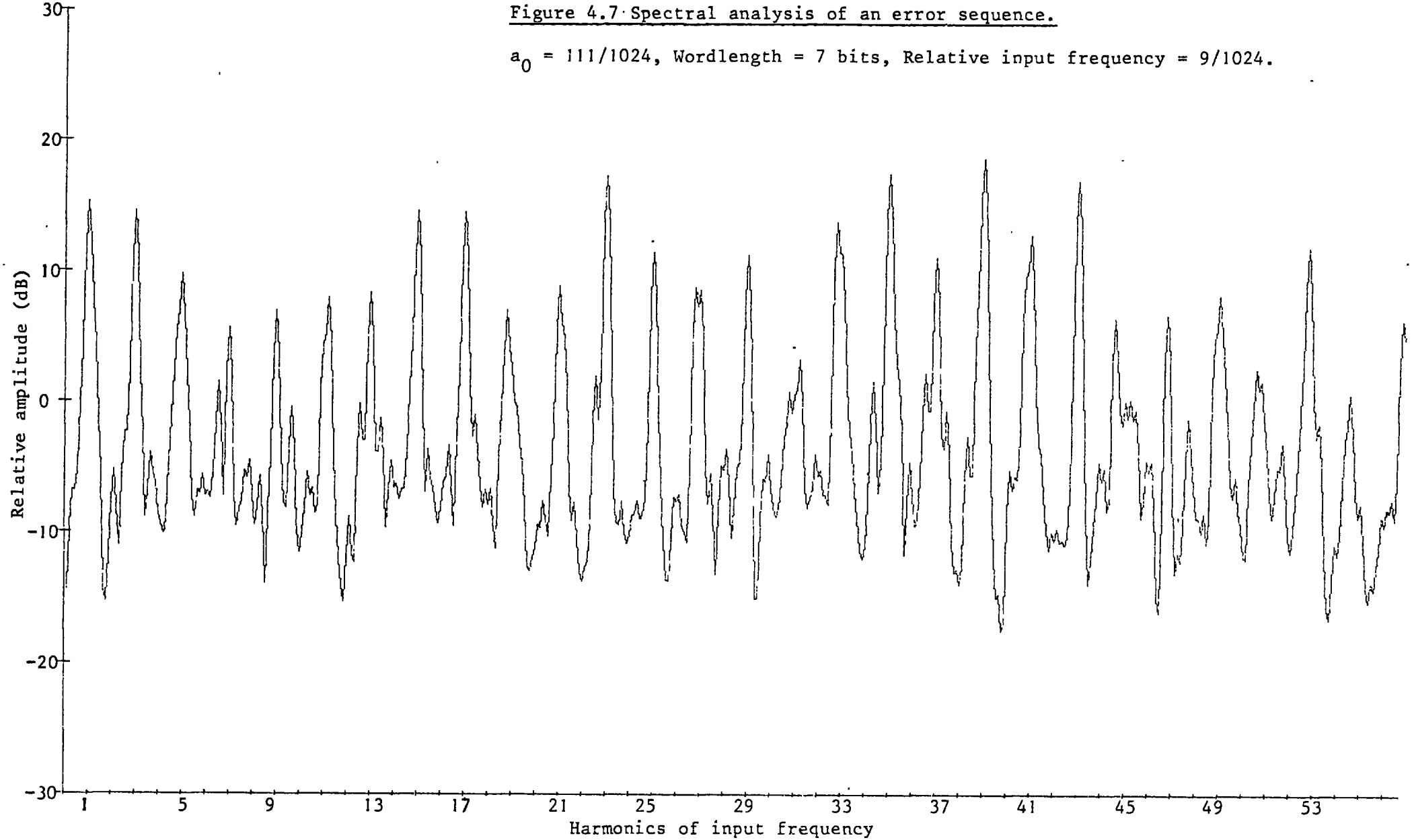


Figure 4.8 Spectral analysis of an error sequence.

a_0 = 111/1024, Wordlength = 8 bits, Relative input frequency = 9/1024.

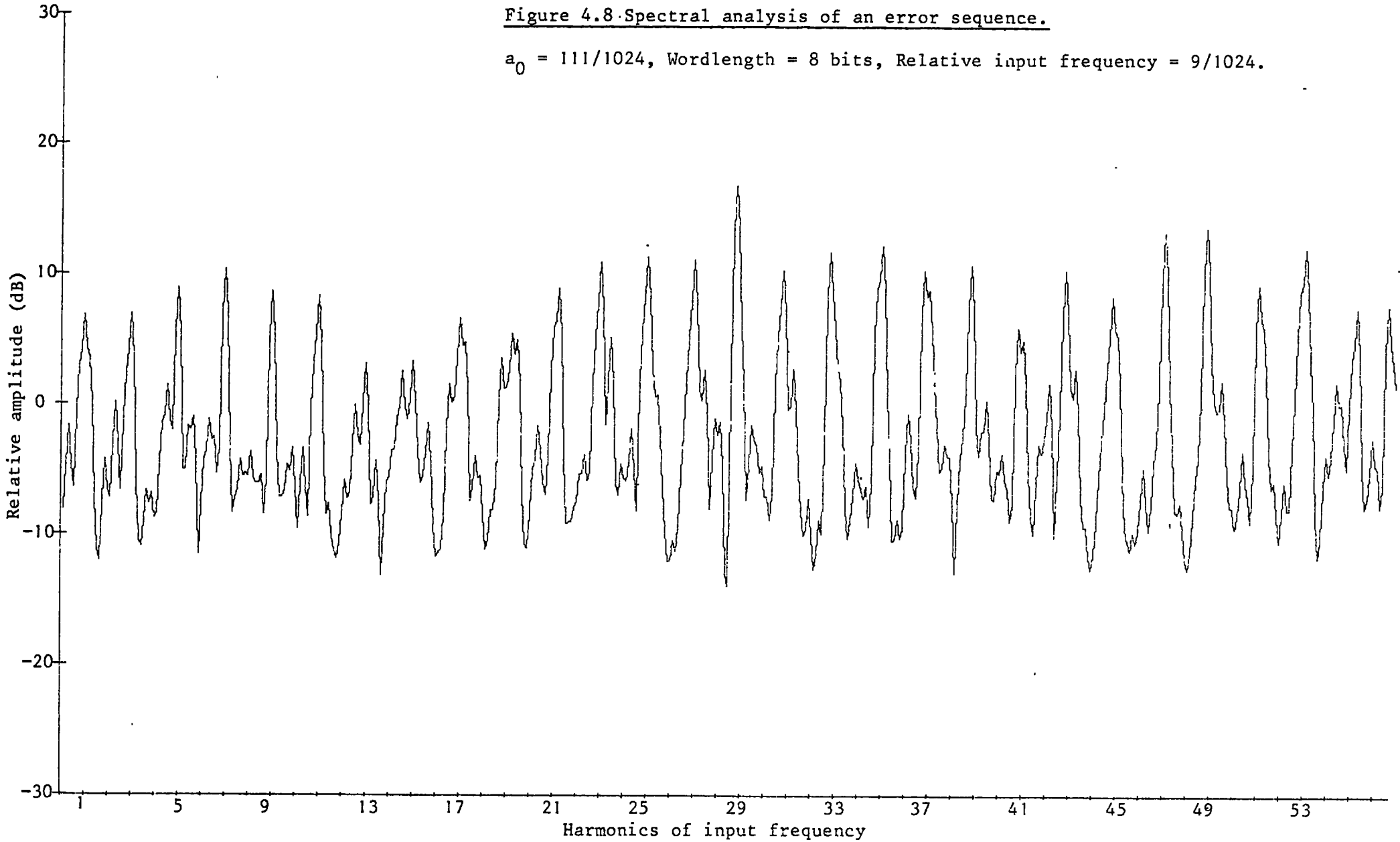


Figure 4.9 Spectral analysis of an error sequence.

a_0 = 411/1024, Wordlength = 8 bits, Relative input frequency = 9/1024.

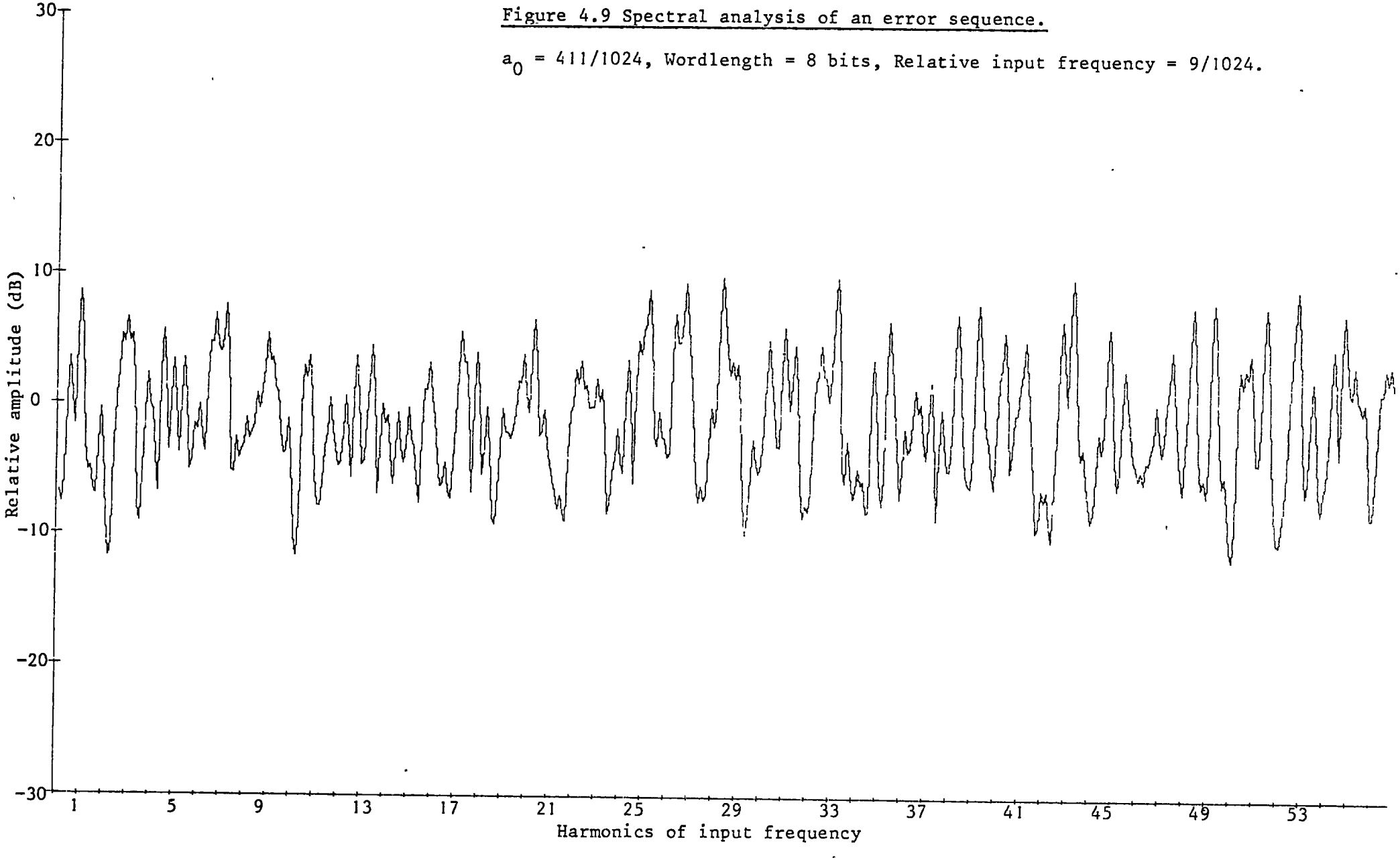


Figure 4.10 Spectral analysis of an error sequence.

a_0 = 11/1024, Wordlength = 8 bits, Relative input frequency = 45/1024.

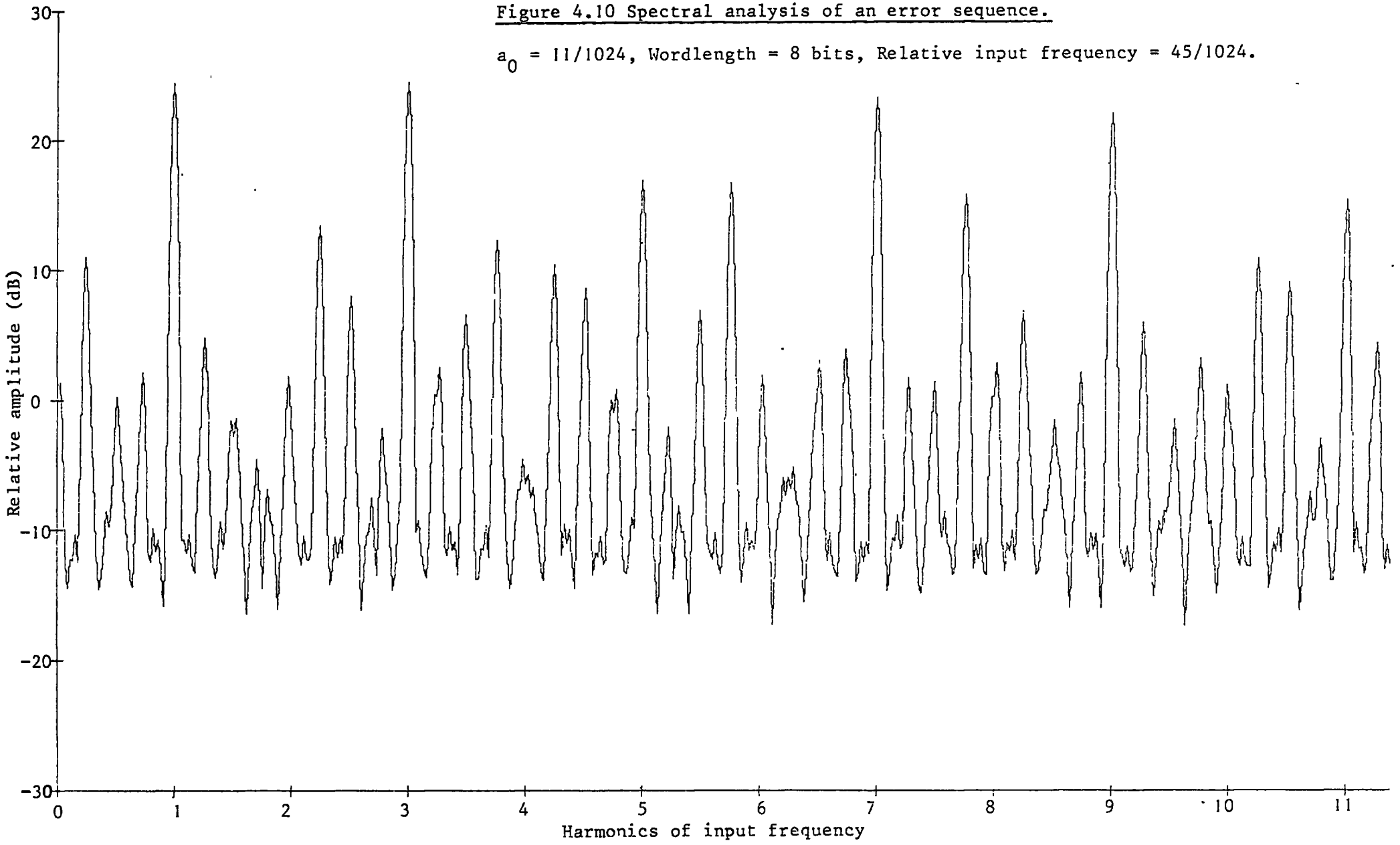


Figure 4.11 Spectral analysis of an error sequence.

a_0 = 111/1024, Wordlength = 8 bits, Relative input frequency = 45/1024.

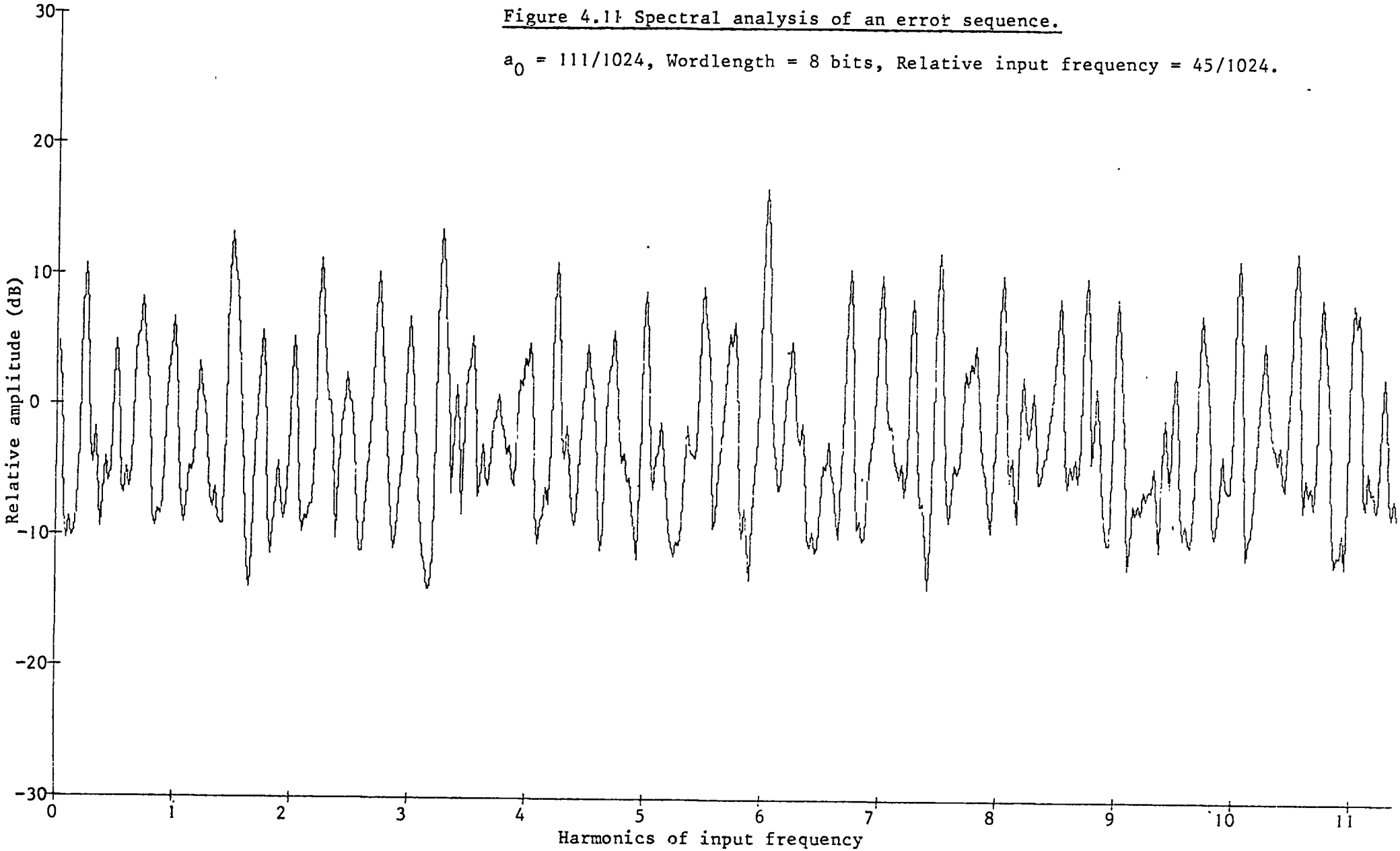
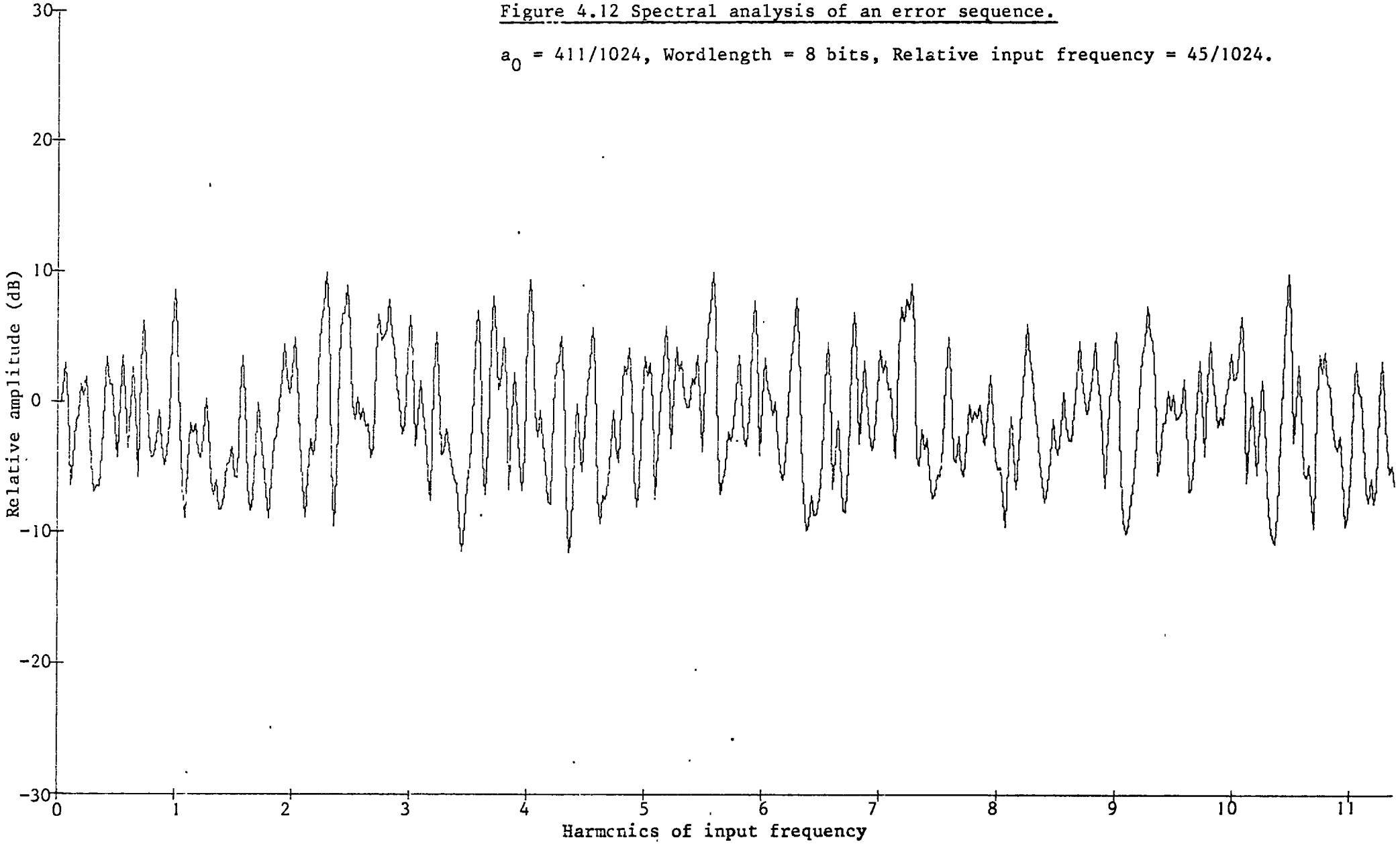


Figure 4.12 Spectral analysis of an error sequence.

a_0 = 411/1024, Wordlength = 8 bits, Relative input frequency = 45/1024.



Firstly, as indicated in Figure 4.1, the DFT samples an underlying continuous frequency function. Secondly, although the choice of 'window' function has optimised the spectral resolution of the DFT, there is still a certain degree of 'blurring', which makes the linear interpolation, inherent in drawing a continuous line plot, a more valid operation. As the plots are produced from 512 points linearly distributed along the frequency axis, the degree of interpolation is not great. Above all, the plots present their information clearly.

A cursory glance at Figures 4.3 to 4.12 reveals clear non-uniformity in the spectral distributions. A slightly more prolonged inspection indicates a definite structure in the spectral responses. This can be seen clearly in Figure 4.3, where the relative amplitude peaks at the odd harmonics of the input frequency.

A comparison of Figures 4.3, 4.8 and 4.9 shows the effect of the coefficient value on the harmonic content, with the input frequency and the signal wordlength kept constant. The harmonic peaks become less strong and well-defined as the multiplier value increases from 11/1024 to 411/1024. Figures 4.10 to 4.12 permit a similar examination of the effect of coefficient value, this time employing a higher relative input frequency. These distributions contain some strong peaks which do not appear to coincide with harmonics of the input frequency. However, detailed inspection indicates that these are harmonics above the Nyquist frequency which are aliasing back into the sub-Nyquist range. These three figures again show the decline in harmonic structure with increasing coefficient value.

The effect of varying input frequency can be observed by comparing the three pairs of graphs, Figures 4.3 and 4.10, 4.8 and 4.11, 4.9 and 4.12. Both members of a given pair of spectral distributions appear to have about the same level of harmonic structure. This implies that the input frequency has little or no effect on the degree of harmonic content

found in a roundoff error sequence.

Finally, Figures 4.4 to 4.8 allow a consideration of the effect of signal wordlength at a constant input frequency and coefficient value. A dependence of the level of harmonic content on wordlength is clearly discernable, the peaks becoming stronger with decreasing wordlength; a relationship which intuitively appears very reasonable.

In summary, it would appear from Figures 4.3 to 4.12 that the spectral distributions of the roundoff error sequences considered are clearly non-uniform and contain strong harmonics of the input frequency. The harmonic structure seems to decline as the coefficient value is increased in the range 11/1024 to 411/1024. There is a less marked decline as the signal wordlength increases from 4 to 8 bits. The frequency of the input sequence, however, appears to have no effect on the level of harmonic structure present in a roundoff error sequence.

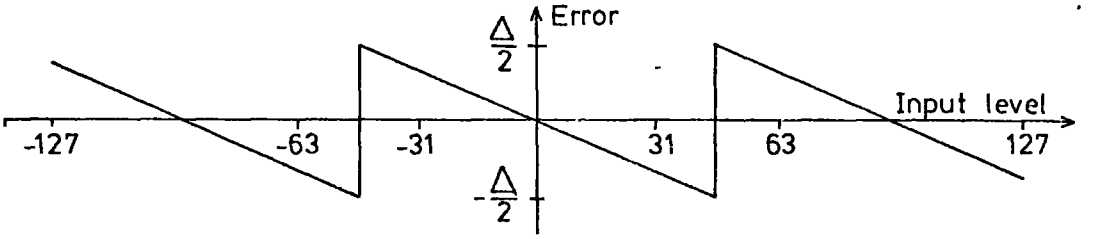
4.3.3 Harmonic generation.

It is not within the scope of this present work to attempt to give a full explanation of the form of the spectral distributions shown in Figures 4.3 to 4.12. The clear non-uniformity of response is the important conclusion to be drawn. However, it may be of some interest to make a brief, qualitative indication of the kind of mechanism involved in creating the observed harmonic structure.

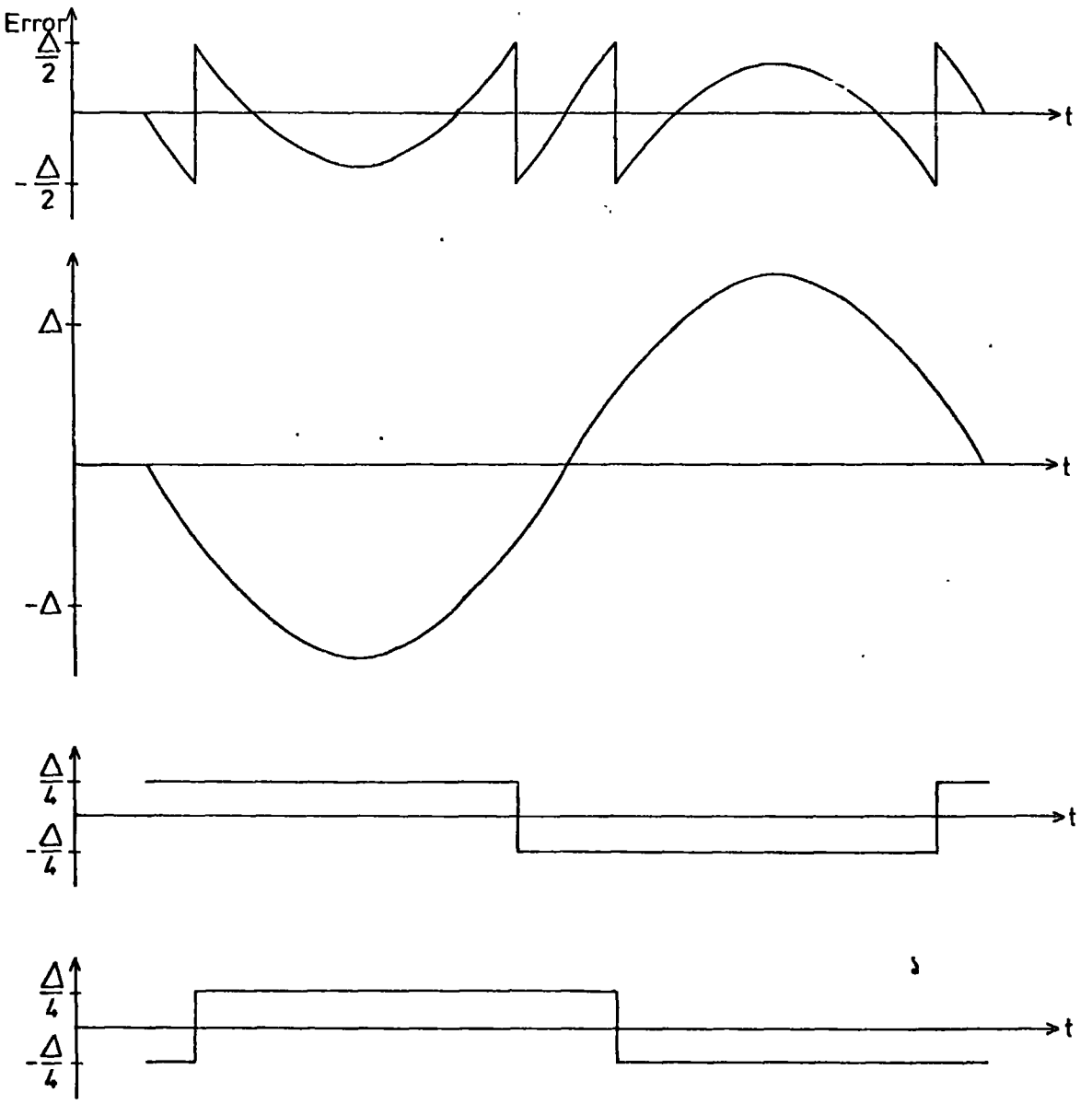
Figure 4.13(a) displays the error function for the multiplier 11/1024. The input range covered is -127 to +127: the range allowed by an 8-bit wordlength. The curve possesses two discontinuities where it encounters the error bounds $\pm\Delta/2$. Any input signal with an amplitude in excess of 46 will traverse this pair of discontinuities in the error curve. Figure 4.13(b) shows the error signal produced at such a multiplier when a sampled sine wave of amplitude 127 is processed. The continuous error waveform shown is that which must underlie the discrete

Figure 4.13 Harmonic generation

a) Error Function. Multiplier = 11/1024.



b) Error waveform from a sampled sinusoid input of amplitude 127.



c) Analysis of error waveform.

sequences actually produced. The shape, and therefore the spectral content, of this error waveform is dependent on the amplitude of the sine wave input, and the value of the multiplier, but the frequency of the input only alters the time-scale of the waveform, not the shape. Figure 4.13(c) indicates how the error waveform may be analysed into three components: a sine wave at the input frequency but in antiphase, and a pair of square waves of amplitude $\Delta/4$. It may be inferred from this diagram that the error waveform will contain all the odd harmonics of the input frequency. The amplitudes of the harmonics will generally decrease with rising harmonic number, that is, frequency. This accords well with the observed results depicted in Figure 4.3.

Certain general principles may be drawn from Figure 4.13. Any error waveform produced when a sampled periodic signal is processed by a multiplier may be analysed into an amplitude-scaled, and perhaps inverted, replica of the input waveform and several pairs of square waves with fundamentals coinciding with that of the input sequence. The amplitudes of the square waves sum to $\Delta/2$. The number of pairs of square waves equals the number of pairs of discontinuities in the error function traversed by the input sequence in one period. This implies that any error waveform produced by a sampled sinusoid input is entirely composed of odd harmonics of the input frequency. Any peak in Figures 4.3 to 4.12 which appears to occupy the position of an even harmonic must in fact be a super-Nyquist odd harmonic which has aliased.

The principles revealed by Figure 4.13 allow an assessment of which situations will be the most prone to give error sequences with strong harmonic structure. If an error waveform contains many pairs of square waves, then the amplitude of these square waves will be correspondingly small. Moreover it is probable that there will be a high degree of partial or total cancellation of the harmonics. It is the number of discontinuities in the error function encountered by the signal

which therefore determines the degree of harmonic content in the error sequence. As this number is increased, so the assumption of a uniform spectral distribution becomes more valid. As discontinuities occur at uniform intervals on the input level axis of the curve, the number encountered may clearly be raised by increasing the amplitude of the input sequence, which, in turn, is permitted by raising the signal wordlength. Increasing the multiplier value in the range 0 to $\frac{1}{2}$ exclusive has the effect of increasing the gradient magnitude of the error function and therefore decreasing the spacing between discontinuities. In the coefficient range $\frac{1}{2}$ to 1 exclusive, increasing the value yields a decreasing magnitude of gradient (now positive) and an increase in the separation of the discontinuities. (For a given input sequence, the multiplier x yields the same error waveform as (1-x) but with phase reversal). The harmonic content of error sequences may therefore be expected to increase with decreasing wordlength[13] and also as the multiplier approaches an integral value, usually zero or unity. The frequency of the input sequence should not affect the degree of harmonic content in an error sequence. These conclusions correlate well with the observed results presented in the previous section.

4.4 Conclusions.

Three important conclusions should be drawn from the observations reported.

(a) Any roundoff error sequence produced by processing a sampled sinusoid at a single multiplier does not possess the uniform spectral distribution expected of white noise, but consists of odd harmonics of the input frequency. To regard roundoff error sequences as white noise sources as does the Knowles - Edwards model[29] is clearly invalid in many situations where the input sequence has a periodic content.

(b) The high first harmonic content found in the roundoff error sequences indicates a significant correlation between the signal and error sequences at the multipliers. Once again an assumption of this model is seen to be highly suspect in some situations.

(c) It would clearly be unwise to test any of the other model assumptions using a periodic input sequence. The only alternative is to use a random input sequence, that is, white noise, so that the modelling of roundoff error sequences as white noise sources should be valid.

CHAPTER 5 - THE VARIANCE OF ROUNDOFF ERROR SEQUENCES.

5.1 Introduction.

In Chapter 3 equations were presented for calculating the mean variance of the roundoff error sequence at a given filter output. The worth of such equations lies in their ability to enable a designer to predict which filter form - direct, canonic, or look-up table - will give the best performance in his particular application, and what wordlength is required to give, for example, a specified signal to noise ratio. However, if the equations are inaccurate so that they represent the options falsely and lead the designer to make a poor selection, they become a hindrance not an asset.

Their formulation relies fundamentally on the assumption that roundoff error sequences can be treated as white noise[29]. The results of the experiment reported in Chapter 4, however, indicate that, given a signal with periodic content and a short wordlength, this assumption is invalid. This immediately limits the proper use of the equations to random signal situations for the wordlengths under consideration: no slight restriction. It is clearly desirable to discover experimentally whether this random signal constraint is sufficient to guarantee the accuracy of the predictive equations, or whether other model assumptions, like the uniform amplitude distributions accredited to roundoff error sequences[29], become invalid at short wordlengths.

This chapter records the nature and results of an experiment which measures the output error variance of a variety of first and second order filter realisations, all under random signal conditions. This permits an assessment of the accuracy of the predictive equations and of the validity of the underlying model assumptions. A consideration of the sensitivity of equation accuracy to filter form or wordlength is also possible, as is an evaluation of the importance of block-floating-point

arithmetic[16] which, as stated in Chapter 3, cannot in general be theoretically assessed. The chapter concludes with a theoretical consideration of the experimental findings.

5.2 Error variance measurements on filters.

As with the spectral analysis experiment of Chapter 4, so too this investigation is carried out by simulation on an IBM 370/168 computer. This approach allows absolute control and reproduceability of experimental conditions, particularly test signals, which could not be achieved in a real situation. In consequence the results of the experiment can be interpreted with greater certainty than would otherwise be permissible.

Table 5.1 indicates the various permutations of filter order, form, arithmetic mode, and roundoff process which are examined. These realisations are tested over the wordlength range of 4 to 8 bits. Four first and four second order filter characteristics have been chosen for implementation. There are two lowpass and two highpass first order filters, whilst the second order responses represent lowpass, highpass, bandpass, and bandstop filters. All have gains which closely approach unity at their frequency of maximum response. Appendix 5 contains tables displaying the filter coefficients used and the characteristics thus obtained as defined by pole and zero positions in the z-plane.

The coefficients employed for the direct and canonic forms require, in general, 10 bits for accurate representation. This is considered to be long enough, relative to the signal wordlengths being used, to free the filters from any effects of a short coefficient wordlength. A comparison may be undertaken of realisations of the same set of ideal filter coefficients for different signal wordlengths, thus permitting a study of those effects alone which are attributable to a short signal wordlength. The modification of the look-up table form suggested to yield a noise reduction, however, relies on using filter coefficients

Table 5.1 Filter realisations tested.

Filter Order	Arithmetic Mode	Direct	Canonic	Look-up Table	Look-up Table, Reduced-noise
First Order	Fixed-point Arithmetic	Rounding & Truncation		Rounding	Rounding
	Block-floating-point Arithmetic	Rounding			
Second Order	Fixed-point Arithmetic	Rounding & Truncation	Rounding & Truncation	Rounding	Rounding
	Block-floating-point Arithmetic	Rounding	Rounding		

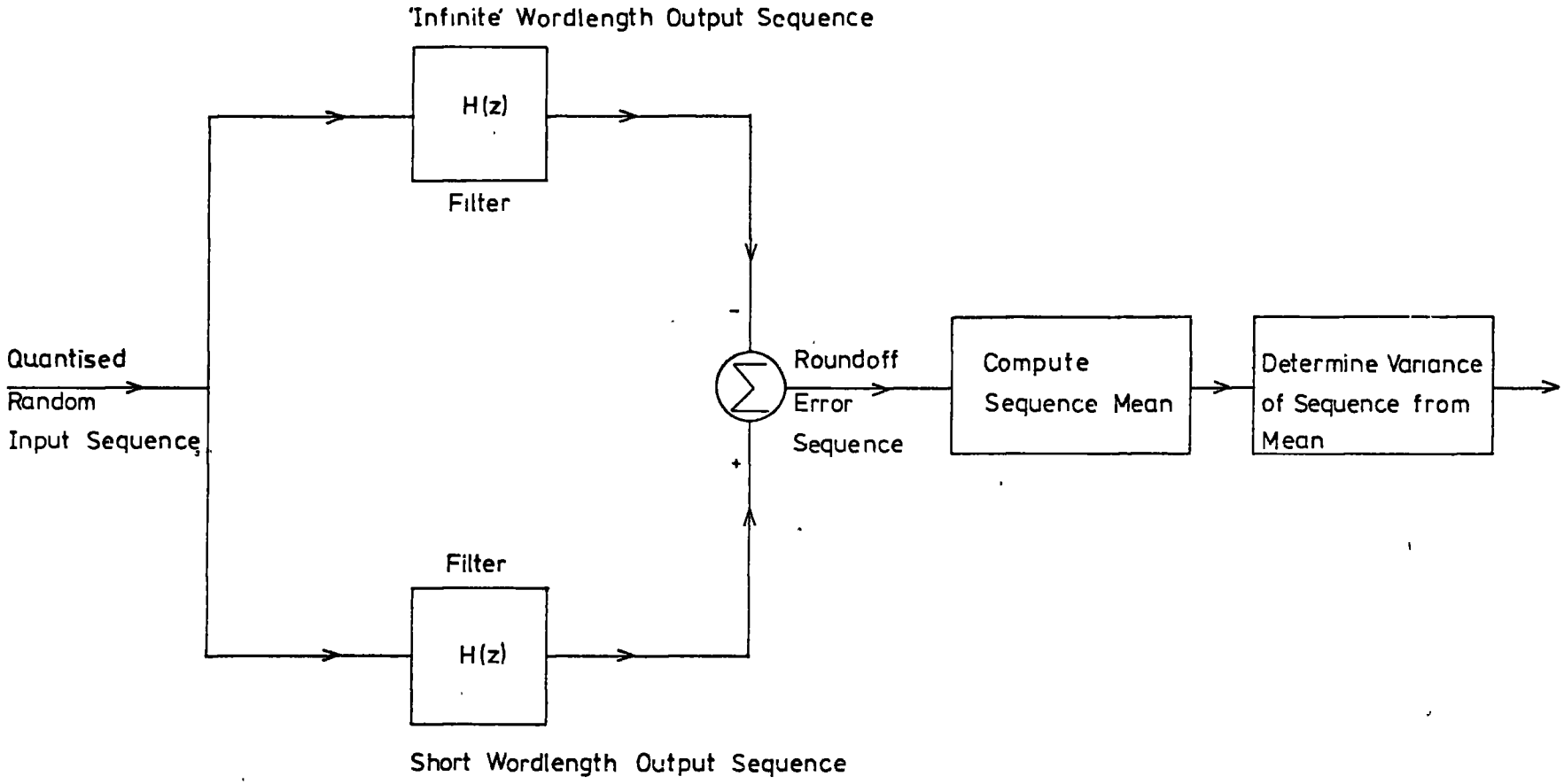
which can be accurately represented within the available signal wordlength. Hence the coefficients used for these forms lead to pole and zero positions which are only approximations of the ideal, as shown in Appendix 5. The omissions from the tables in Appendix 5 are caused because this approximation becomes degraded as the wordlength is decreased, and becomes unacceptably poor for some of the filters at the shorter wordlengths.

5.2.1 Experimental method.

The technique employed to measure roundoff error variance is broadly similar to that used for the spectral analysis of error sequences. That is, an error sequence is formed by the parallel processing of a given input signal in a limited wordlength channel and in a so-called 'infinite' wordlength channel, and the subsequent sample by sample subtraction of the two output sequences. There are, however, several important differences from the previously reported experiment; the two processing channels no longer consist of single multipliers, but now contain the filter realisations under test; the input sequence, as previously mentioned, consists of random integers; and the error sequences formed are processed by firstly computing the mean of the 1024 error samples, and then calculating the mean squared deviation, or the variance, of these samples from their mean. This is schematically represented in Figure 5.1, and Appendix 6 contains a listing of the Fortran IV main program, EXEC, used to implement the experiment. The results of 100 runs processing different input sequences are averaged so that a mean variance can be returned along with its standard error.

One problem which arises with filters is that of their transient response. In general, filters determine a particular output sample not only from the current input sample but from previous input and output samples. Hence when the first input sample is applied to a filter its

Figure 5 1 Error variance measurements on filters.



output depends on the signal data which happens to have been stored beforehand at the appropriate locations in the filter structure. The two filters which perform the parallel processing in this experiment have all data locations corresponding to previous inputs or outputs set to zero before processing the initial sample of the first input sequence.

The measurement of the variance of an error sequence is strictly only meaningful when the sequence is wide-sense-stationary, that is, when neither its mean nor the variance about that mean change with time. As the filter signal data is set up so that there is no error in the delayed output samples in the short wordlength filter, it could be argued that stationarity of the output error sequence will not be obtained immediately filtering commences. Such an argument has particular weight when using the asymmetrical truncation process so that the mean error stabilises at a significant deviation from zero. Assuming that stationarity is achieved within the first run of 1024 samples, the error variance is not measured during this run, but the signal data stored in the filters at the end of it permits the commencement of the second run without any loss of stationarity. Similarly all subsequent runs employ the signal data remaining in the filters from the previous sequence processed. Hence 101 batches of 1024 input samples are processed, the results of the first being discarded, whilst the final 100 are used to compute the mean error variance and its standard error.

5.2.1.1 Generation of the random input sequence.

Consideration must now be given to the generation of suitable random input sequences with which to test the filter realisations. On the grounds of experimental reproduceability and control, and of consistency of approach, the use of a hardware white noise generator is excluded, and a software technique must be used. One of two basic methods may be chosen; a standard table of random numbers may be stored in the

computer memory and read as required; or a mathematical algorithm may be employed to generate what can only be a pseudo-random number sequence. The former approach is impractical because of the large number (over 10^5) of sequence samples required. A technique in the latter category is therefore chosen even though a truly random sequence cannot be produced because each element is determined mathematically from another, usually that preceding it. Much care is needed in the choice of algorithm to produce a sequence which is most nearly truly random. MacLaren and Marsaglia[76] have presented a paper in which they discuss the various algorithm types and also apply tests of sequence randomness which they suggest are more stringent than those commonly employed. The algorithm which performed best in their tests forms the basis of the gencrator used for the error variance measurement experiment.

To improve upon the properties of a sequence generated by the common multiplicative congruential algorithm[76], such a sequence is shuffled into an order determined by a second pseudo-random sequence also produced by a congruential method. The two sequences they suggest may be represented by

$$U_{k+1} = (2^{17} + 3).U_k \qquad \text{mod } 2^{35} \qquad (5.2.1)$$

and

$$V_{k+1} = (2^{7} + 1).V_k + 1 \qquad \text{mod } 2^{35} \qquad (5.2.2)$$

where $U_0 = 1$ and $V_0 = 0$ and k indicates the position of the element in the sequence. Firstly the numbers $U_1 \ldots U_{128}$ are written into a table in the computer memory, then the least significant 7 bits of V_k are used to address this table and access the kth random number. This store location is then refilled with the next element in the main sequence U. Equations (5.2.1) and (5.2.2) cannot, however, be implemented on an IBM 370/168 which has a wordlength of 32 bits and therefore permits

arithmetic modulo 2^{31} but not 2^{35} as stated in the equations. This modification should have little effect on the properties of the random sequence thus produced, apart from reducing the available sequence length before repetition to something in excess of 2^{31}, which is much longer than the experimental requirement of less than 2^{17}.

The method described produces a random sequence whose elements ideally are uniformly distributed in the region 0 to $(2^{31} - 1)$ inclusive. The requirement for the experiment, however, is for sequences which are symmetrical about zero, and the maximum amplitude ever needed is 127. This calls for appropriately shifting and scaling the original sequence. The scaled-down sequence must contain only integers and so roundoff occurs in their formation; rounding is always the method used to achieve this. Such scaling and roundoff should produce sequences with characteristics of randomness which do not deviate significantly from those of the fundamental sequence. Appendix 6 contains listings of the two Fortran IV subroutines, PRAND and RAND, which together are used to generate the random input sequences required throughout this experiment.

The amplitude of the random input sequence fed to a particular filter realisation under examination is always the maximum which does not cause the signal level, at any point in the filter in the short wordlength processing channel, to exceed the limits imposed by the signal wordlength under consideration; this allows the greatest advantage to be taken of the available wordlength. For all the first order filter realisations tested, subroutines have been written which calculate this amplitude for the given filter coefficients, form, and wordlength, and these are listed in Appendix 6; LIMSI, BOUND, and LIMPSI are used for the direct fixed-point arithmetic, direct block-floating-point arithmetic, and the look-up table forms respectively. No suitable algorithm has been discovered, however, to provide the same information in the case of second order filters. This means that the appropriate

amplitude for the input sequence has to be found by trial and error. The amplitude is initialised to the maximum allowed by the signal wordlength. Thereafter if an excessive signal level is detected at any point in the filter, the input amplitude is reduced by one and the entire experiment must be recommenced. This clearly becomes very wasteful of computer time when the final amplitude is significantly lower than the initial value. The failure to solve this problem arises from an inability to predict the allowed signal conditions which yield a maximum signal level at the output of a second order filter.

5.2.2 Experimental results.

The results of this experiment are presented both graphically in Figures 5.2 to 5.9 and in tabular form in Tables 5.2 to 5.9. As all the signals consist only of integers, the quantisation width is always unity and is not dependent on the wordlength. For this reason the presentation of absolute error variance measurements permits the most clear assessment of performance. Each graph and table pair exhibits the measurements obtained from all the various realisations tested of a given ideal filter response; note that the graphical information relating to each second order filter is presented on two separate plots, denoted (a) and (b). The short horizontal bars shown on the graphs represent the error variance theoretically predicted by the appropriate equation as presented in Chapter 3; this information is also given in the tables. For reasons of clarity no attempt is made to include error bars on the graphs. However, each mean experimental result presented in the tables bears its calculated standard error.

The reason for not testing some of the reduced-noise look-up table realisations has already been given. There are, however, other gaps in the results which require explanation. Care has been taken throughout the execution of this experiment to detect situations where the output

sequence from a filter multiplier can only consist of zeros. In such a case the particular multiplier has an effective coefficient value of zero, and so the filter can no longer be deemed to be an adequate realisation of the ideal response; the results, therefore, are not presented. There are a few cases, most notably the canonic block-floating-point arithmetic realisation of the second order bandpass filter, where the particular combination of filter coefficients, form, wordlength, and arithmetic mode requires that the amplitude of the input sequence be reduced to zero before signal overflow is avoided at all points in the filter; results in such a situation are clearly meaningless.

5.2.2.1 First order highpass(i) filter.

The results of the tests carried out on this filter are presented in Figure 5.2 and Table 5.2. Figure 5.2 shows a general lack of agreement between experiment and theory which is revealed more clearly still when the experimental errors indicated in Table 5.2 are taken into consideration. The degree of discrepancy varies quite markedly with the realisation. For instance, the experimental values of the normal look-up table form deviate widely from the theory, whereas the results obtained from the reduced-noise look-up table form are very close indeed to those theoretically predicted. The error variances yielded by the direct, block-floating-point arithmetic implementations also show a significant deviation from the theory. It should be noted that whereas the theoretical equations predict that the use of block-floating-point arithmetic will yield a slightly higher error variance than the use of fixed-point arithmetic, the opposite is in fact the case. The two direct fixed-point arithmetic forms show good agreement at the higher wordlengths, but this deteriorates as the wordlength is decreased. In practice all three direct forms yield a similar level of error variance. In general, the look-up table forms show a lower degree of deviation from the ideal; this is

Figure 5.2 Results of error variance measurements.

First order highpass(i) filter.

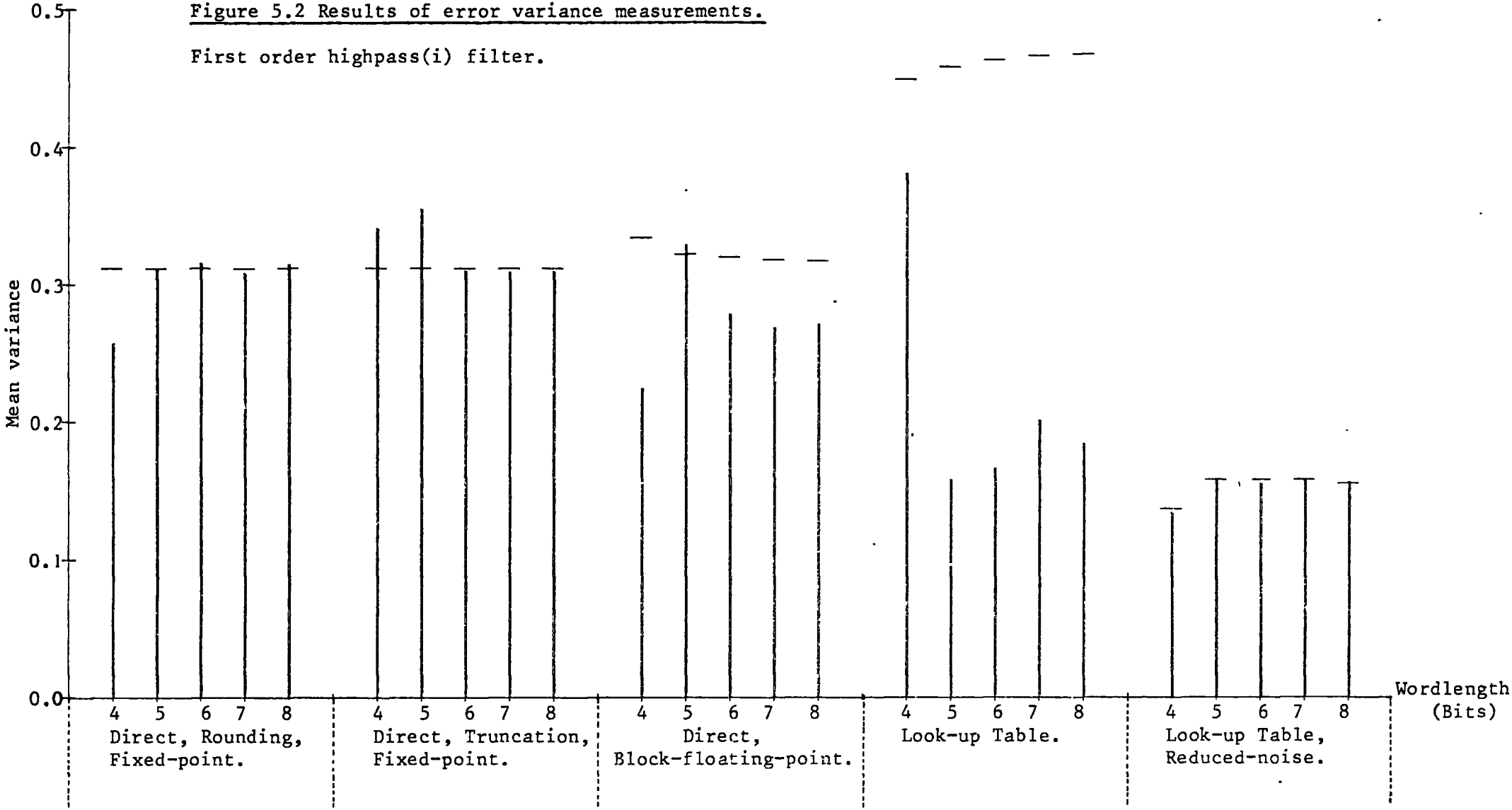


Table 5.2 Error variance measurements.

First order highpass(i) filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.	0.257 ± 0.002	0.311 ± 0.003	0.316 ± 0.003	0.308 ± 0.002	0.315 ± 0.003
Direct, Truncation, Fixed-point.	0.341 ± 0.003	0.355 ± 0.003	0.310 ± 0.003	0.309 ± 0.002	0.309 ± 0.003
Direct, Block-floating-point, Experimental.	0.224 ± 0.002	0.329 ± 0.003	0.278 ± 0.002	0.268 ± 0.002	0.271 ± 0.002
Theory.	0.33354	0.32250	0.31995	0.31798	0.31717
Look-up Table, Experimental.	0.381 ± 0.004	0.158 ± 0.002	0.166 ± 0.002	0.201 ± 0.002	0.184 ± 0.002
Theory.	0.44863	0.45838	0.46326	0.46570	0.46692
Look-up Table, Reduced-noise, Experimental.	0.1337 ± 0.0008	0.157 ± 0.002	0.155 ± 0.001	0.157 ± 0.001	0.156 ± 0.001
Theory.	0.13675	0.15802	0.15802	0.15802	0.15489

Theory: Direct, Fixed-point 0.31209

particularly true for the reduced-noise look-up table realisations.

5.2.2.2 First order lowpass(i) filter.

Figure 5.3 and Table 5.3 indicate that the results obtained for this filter possess many of the characteristics mentioned in the previous paragraph. Once again the normal look-up table form shows the most marked discrepancy between theory and practice, and moreover theory again predicts a worse performance than is actually achieved. The increase in the noise produced by the 4- and 5-bit look-up table forms, when compared with the longer wordlength realisations of this structure, is caused by the need to double the rescaling factor at the filter output. Theory predicts that the use of block-floating-point arithmetic will produce a minor degradation in performance, but this does not result. The reduced-noise look-up table form exhibits both the best performance and the highest degree of agreement with the theory.

5.2.2.3 First order highpass(ii) filter.

The results obtained for this filter, as shown in Figure 5.4 and Table 5.4, display many of the properties already mentioned. The well-behaved nature of the reduced-noise look-up table form is once more to be remarked upon. The ordinary look-up table form also gives a relatively low noise level despite the poor expectations produced by the theory. The direct forms all yield similar performances with block-floating-point arithmetic again proving to be more useful than theoretically anticipated. The results obtained for the direct fixed-point arithmetic implementations display a trend for the discrepancy between theory and experiment to become diminished as the wordlength is increased.

5.2.2.4 First order lowpass(ii) filter.

This filter has its pole positioned extremely close to the unit

Figure 5.3 Results of error variance measurements.

First order lowpass(i) filter.

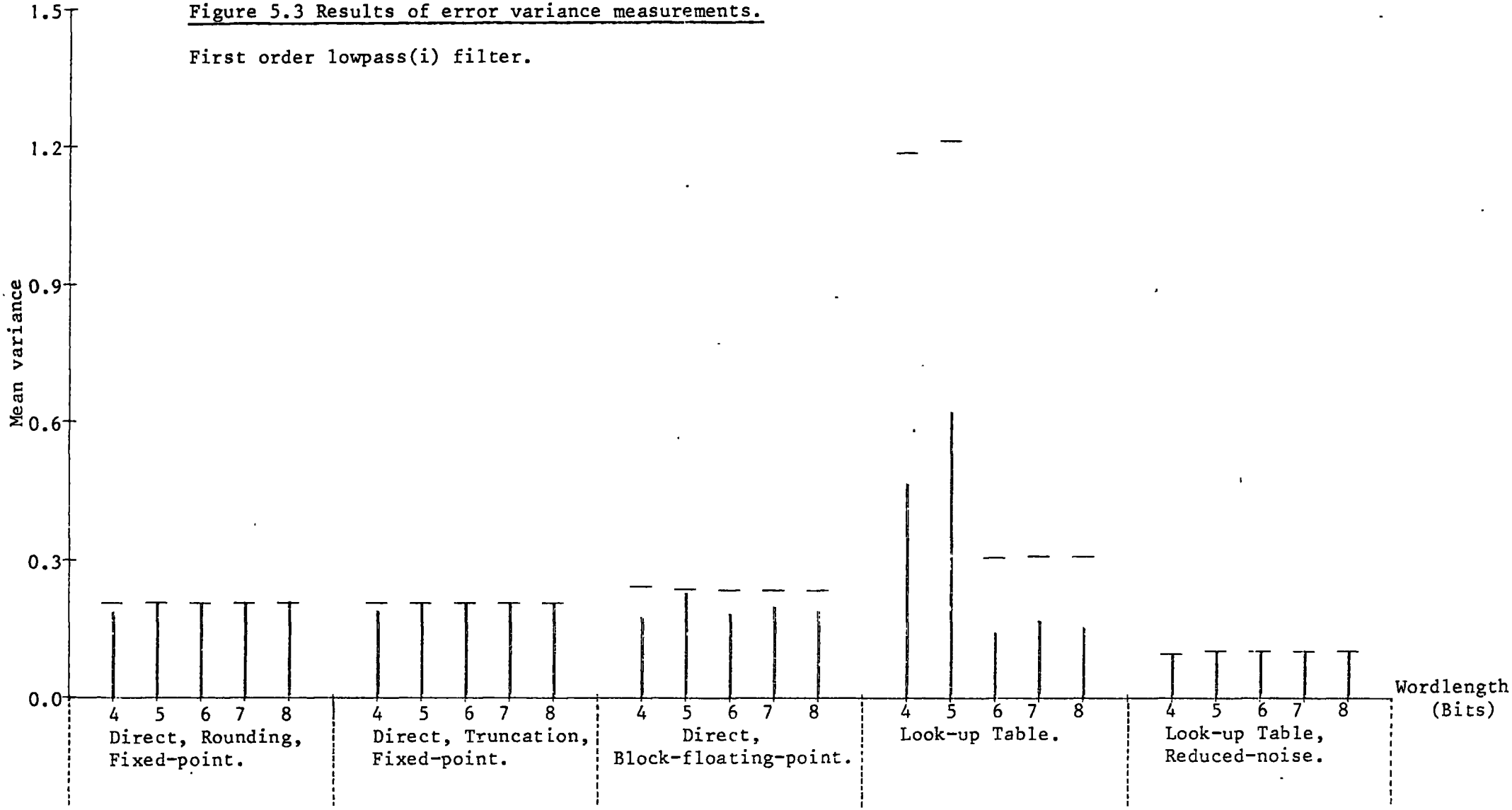


Table 5.3 Error variance measurements.

First order lowpass(i) filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.	0.1861 ± 0.0009	0.2073 ± 0.0008	0.206 ± 0.001	0.207 ± 0.001	0.209 ± 0.002
Direct, Truncation, Fixed-point.	0.1894 ± 0.0009	0.204 ± 0.002	0.206 ± 0.001	0.208 ± 0.002	0.206 ± 0.001
Direct, Block-floating-point, Experimental.	0.1750 ± 0.0009	0.229 ± 0.002	0.1831 ± 0.0009	0.199 ± 0.001	0.1896 ± 0.0009
Theory.	0.24346	0.23751	0.23543	0.23409	0.23360
Look-up Table, Experimental.	0.464 ± 0.002	0.621 ± 0.003	0.1420 ± 0.0008	0.1692 ± 0.0008	0.152 ± 0.001
Theory.	1.1890	1.2148	0.30693	0.30855	0.30936
Look-up Table, Reduced-noise, Experimental.	0.0956 ± 0.0005	0.1023 ± 0.0004	0.1026 ± 0.0005	0.1032 ± 0.0005	0.1034 ± 0.0005
Theory.	0.09697	0.10306	0.10306	0.10306	0.10306

Theory: Direct, Fixed-point 0.20678

Figure 5.4 Results of error variance measurements.

First order highpass(ii) filter.

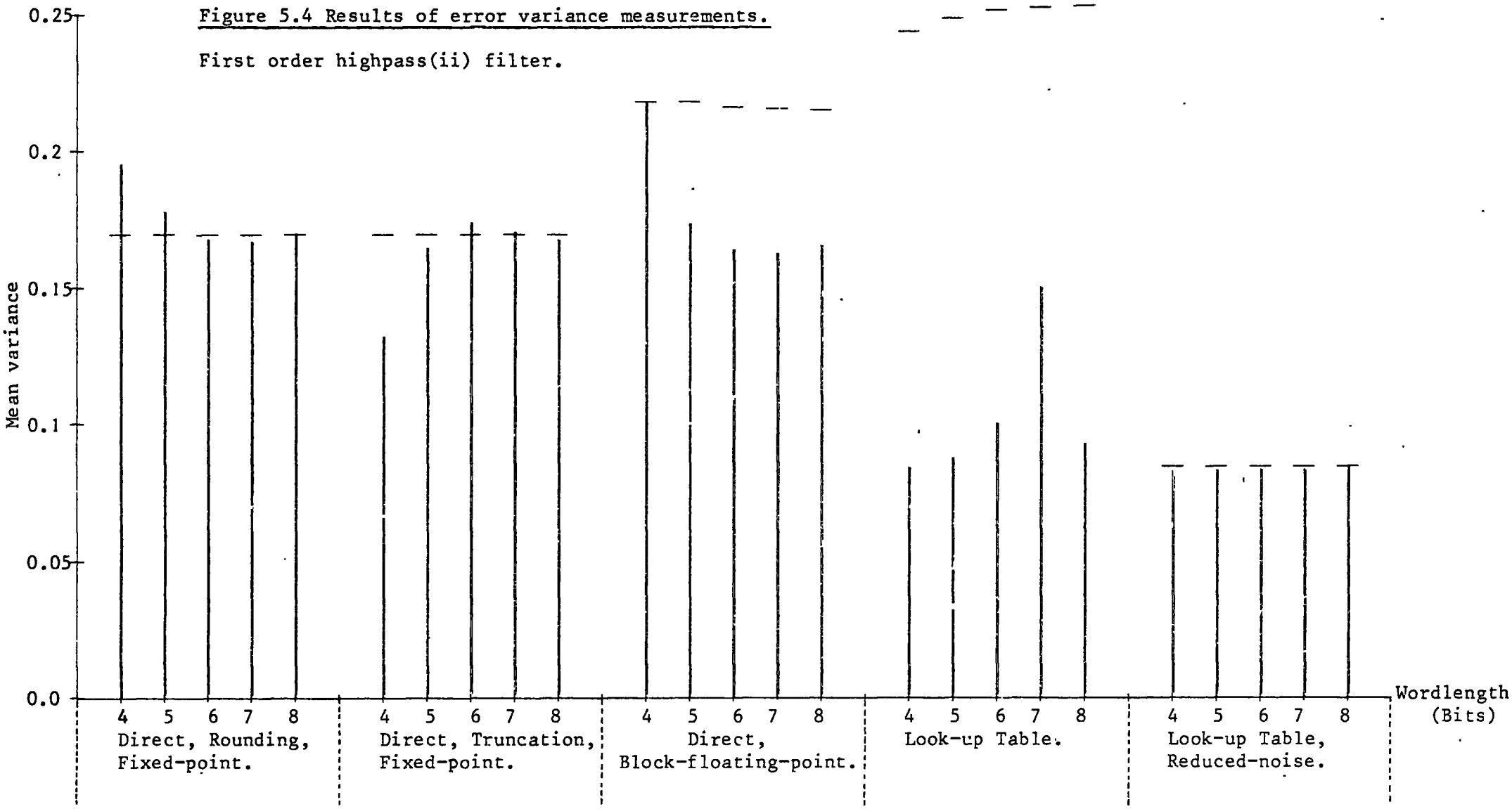


Table 5.4 Error variance measurements.

First order highpass(ii) filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.	0.1953 ± 0.0009	0.1778 ± 0.0007	0.1677 ± 0.0007	0.1669 ± 0.0008	0.1698 ± 0.0006
Direct, Truncation, Fixed-point.	0.1318 ± 0.0006	0.1643 ± 0.0008	0.1739 ± 0.0007	0.1704 ± 0.0007	0.1675 ± 0.0007
Direct, Block-floating-point, Experimental.	0.2173 ± 0.0008	0.1735 ± 0.0007	0.1639 ± 0.0007	0.1627 ± 0.0007	0.1656 ± 0.0006
Theory.	0.21804	0.21821	0.21610	0.21549	0.21515
Look-up Table, Experimental.	0.0841 ± 0.0003	0.0875 ± 0.0003	0.0998 ± 0.0004	0.1500 ± 0.0007	0.0924 ± 0.0003
Theory.	0.24333	0.24862	0.25126	0.25258	0.25324
Look-up Table, Reduced-noise, Experimental.	0.0824 ± 0.0003	0.0832 ± 0.0003	0.0833 ± 0.0003	0.0831 ± 0.0003	0.0847 ± 0.0003
Theory.	0.08466	0.08466	0.08466	0.08466	0.08466

Theory: Direct, Fixed-point 0.16927

circle in the z-plane. In consequence the value of the coefficient a_0 must be made very low. This in turn gives rise to the large number of omissions from Figure 5.5 and Table 5.5 at the shorter wordlengths. The most significant feature of these results is the deviation from theory of the error variances obtained for direct fixed-point arithmetic realisations. The error variances measured for the 6- and 7-bit direct fixed-point arithmetic forms employing rounding exceed the theoretical prediction by more than an order of magnitude. Use of block-floating-point arithmetic, however, brings greatly reduced noise, and an increased conformity to theory. The reduced-noise look-up table form is again the only one both to yield a low error variance and to act in a predictable manner.

5.2.2.5 Second order bandpass filter.

As this filter has its poles very close to the unit circle in the z-plane its pole section has a high gain. This causes problems of excessive signal levels, especially at the intermediate stage of canonic realisations. The fact that $|b_1| + |b_2| > 1$ means that only relatively low amplitude input sequences can be successfully processed by the canonic forms, indeed when block-floating-point arithmetic is used the input amplitude has to be reduced to zero to avoid signal overflow. The fixed-point arithmetic direct and canonic realisations yield results which differ more markedly from the theory than is generally the case for first order filters. The error variances measured for the canonic forms are in general somewhat lower than those obtained for the direct fixed-point arithmetic realisations, and comparable to the results for the direct block-floating-point arithmetic filters, which once again perform better than theory would suggest. The worth of the reduced-noise look-up table form is demonstrated once more, while the normal look-up table realisations yield relatively low noise levels significantly below

Figure 5.5 Results of error variance measurements.

First order lowpass(ii) filter.

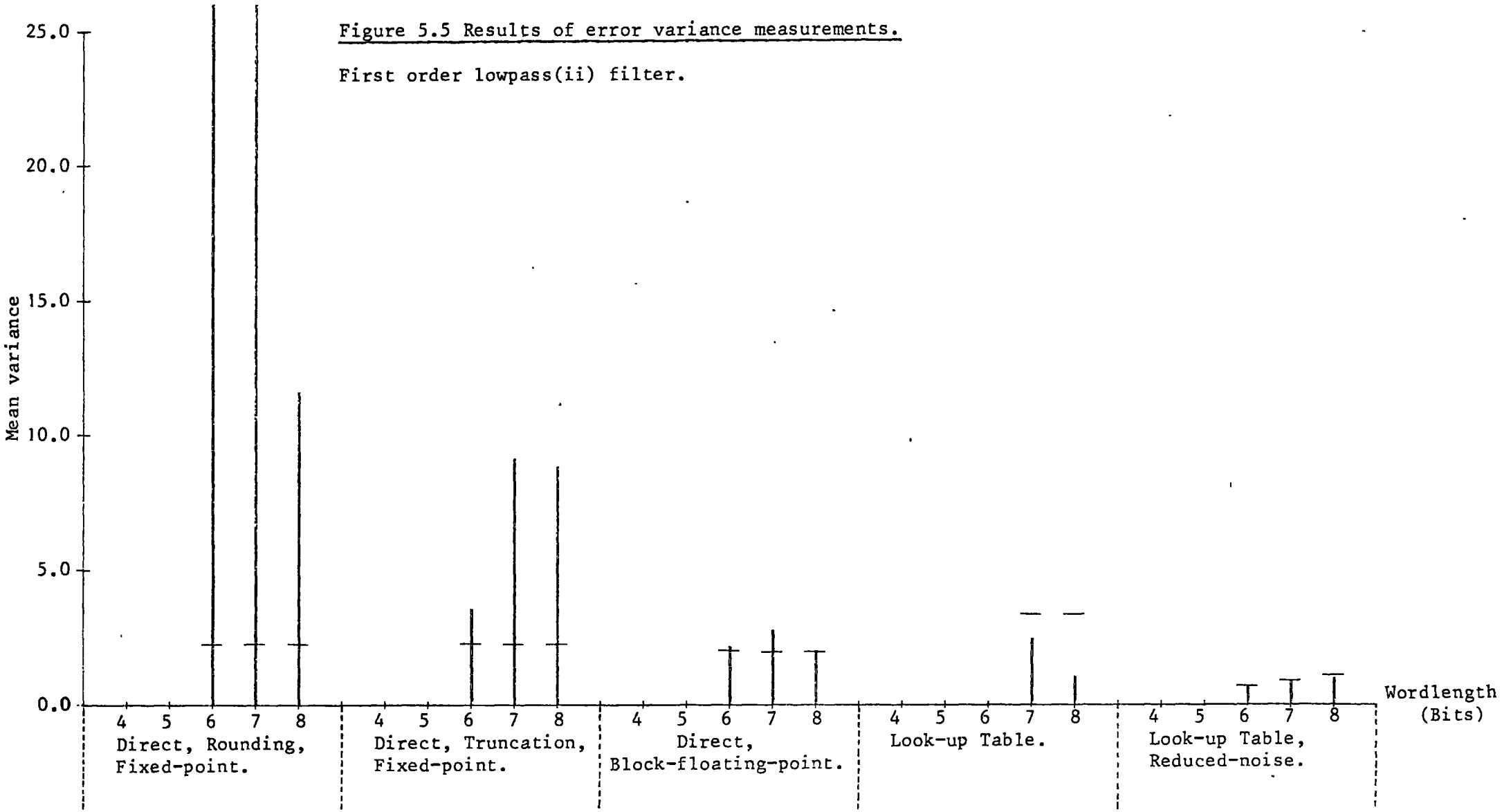


Table 5.5 Error variance measurements.

First order lowpass(ii) filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.			26 ± 2	26 ± 2	11.6 ± 0.3
Direct, Truncation, Fixed-point.			3.57 ± 0.08	9.1 ± 0.2	8.8 ± 0.2
Direct, Block-floating-point, Experimental.			2.13 ± 0.07	2.73 ± 0.08	2.01 ± 0.06
Theory.			1.9874	1.9662	1.9567
Look-up Table, Experimental.				2.46 ± 0.07	1.04 ± 0.03
Theory.				3.3283	3.3371
Look-up Table, Reduced-noise, Experimental.			0.67 ± 0.02	0.88 ± 0.02	1.01 ± 0.03
Theory.			0.68817	0.91022	1.0879

Theory: Direct, Fixed-point 2.2305

Figure 5.6(a) Results of error variance measurements.

Second order bandpass filter.

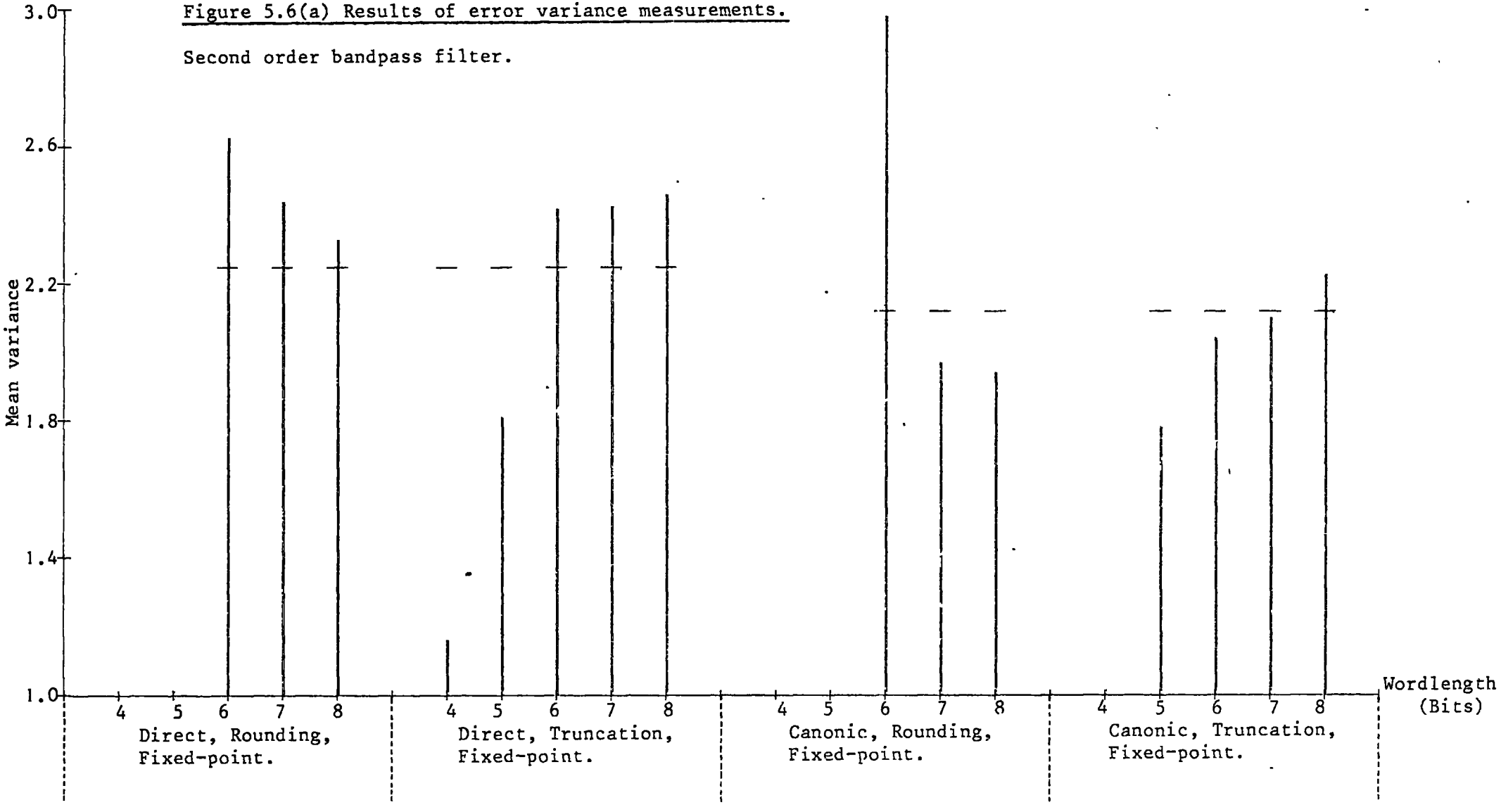


Figure 5.6(b) Results of error variance measurements.

Second order bandpass filter.

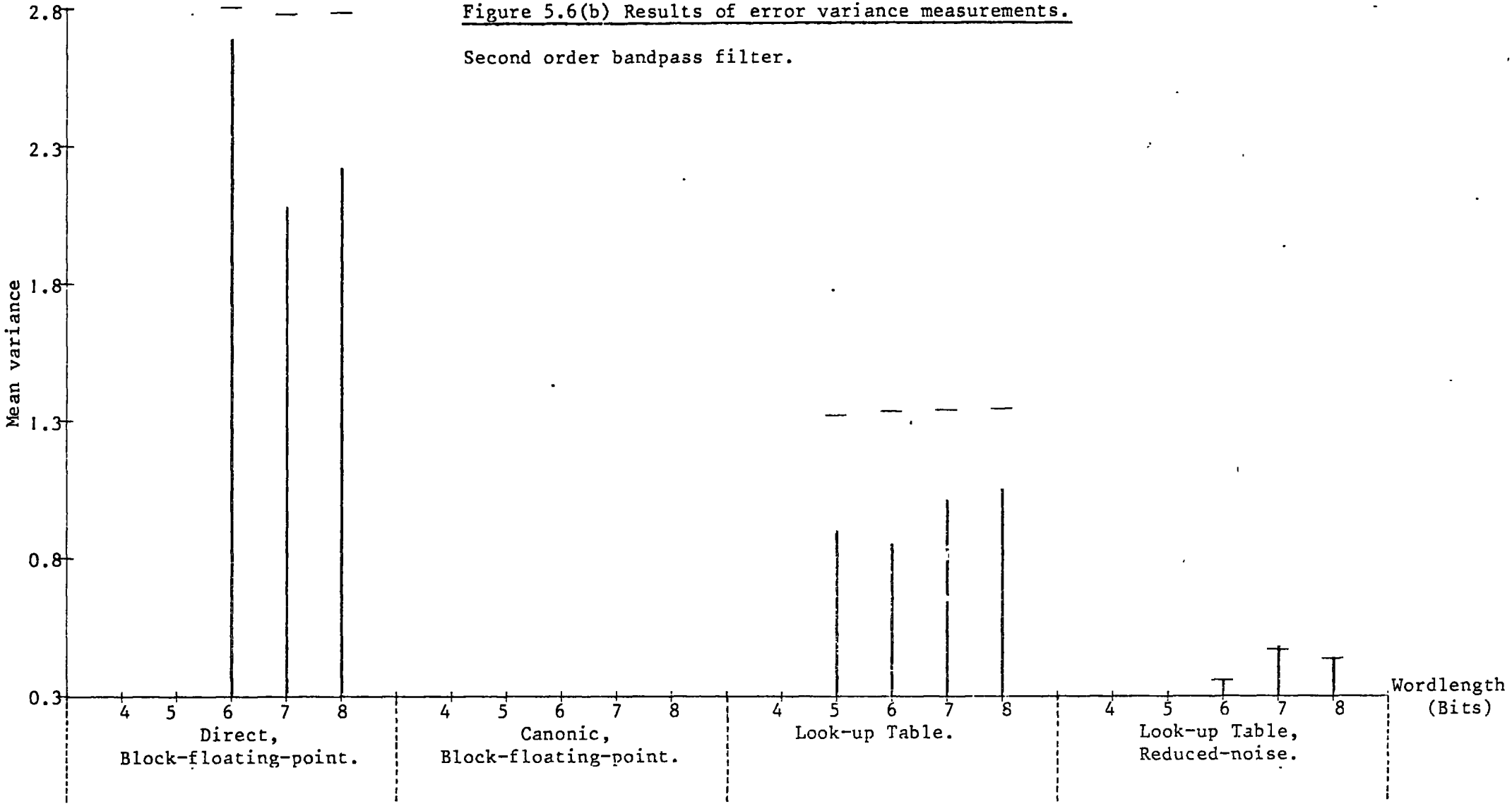


Table 5.6 Error variance measurements.

Second order bandpass filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.			2.63 ± 0.04	2.44 ± 0.04	2.33 ± 0.04
Direct, Truncation, Fixed-point.	1.16 ± 0.02	1.81 ± 0.03	2.42 ± 0.04	2.43 ± 0.04	2.46 ± 0.04
Canonic, Rounding, Fixed-point.			2.98 ± 0.04	1.97 ± 0.02	1.94 ± 0.03
Canonic, Truncation, Fixed-point.		1.78 ± 0.02	2.04 ± 0.03	2.10 ± 0.03	2.23 ± 0.03
Direct, Block-floating-point, Experimental.			2.69 ± 0.04	2.08 ± 0.04	2.22 ± 0.04
Theory.			2.8073	2.7794	2.7845
Canonic, Block-floating-point, Experimental.	Unsuitable for implementation as $\|b_1\| + \|b_2\| > 1$.				
Theory.					
Look-up Table, Experimental.		0.90 ± 0.02	0.85 ± 0.02	1.01 ± 0.02	1.05 ± 0.02
Theory.		1.3199	1.3340	1.3410	1.3445
Look-up Table, Reduced-noise, Experimental.			0.357 ± 0.005	0.481 ± 0.006	0.442 ± 0.006
Theory.			0.35915	0.47166	0.43721

Theory: Direct, Fixed-point 2.2467
Canonic, Fixed-point 2.1216

the theoretical predictions.

5.2.2.6 Second order bandstop filter.

The results obtained for this filter, as presented in Figures 5.7(a) and (b) and Table 5.7, demonstrate well the main characteristics of the results of the entire experiment. The canonic forms produce rather less noise than the direct forms. The use of block-floating-point arithmetic brings a small, unpredicted reduction in the degree of error variance. The normal look-up table form yields a further decrease, which is larger than anticipated. Finally, the reduced-noise modification to the look-up table realisation produces the lowest noise level and excellent agreement between theory and practice. The direct and canonic forms show a general trend towards a diminishing discrepancy between experiment and theory as the signal wordlength is increased.

5.2.2.7 Second order lowpass filter.

The error variance measurements recorded for this filter are presented in Figures 5.8(a) and (b) and Table 5.8. As predicted by the theory, the canonic forms yield lower noise than their direct equivalents. The use of block-floating-point rather than fixed-point arithmetic in the direct form makes little difference to the resulting error variance, despite the theoretical prediction of degraded performance. The canonic form does, however, benefit from the employment of block-floating-point arithmetic, and the results show much better agreement between theory and experiment than is generally the case for this arithmetic mode. For once the look-up table forms do not compare favourably with other realisations; this is especially true of the unmodified forms. This arises because of the need to rescale the filter outputs by a factor of two, which degrades the error variance by a factor of four. Despite this, the canonic realisation employing block-floating-

Figure 5.7(a) Results of error variance measurements.

Second order bandstop filter.

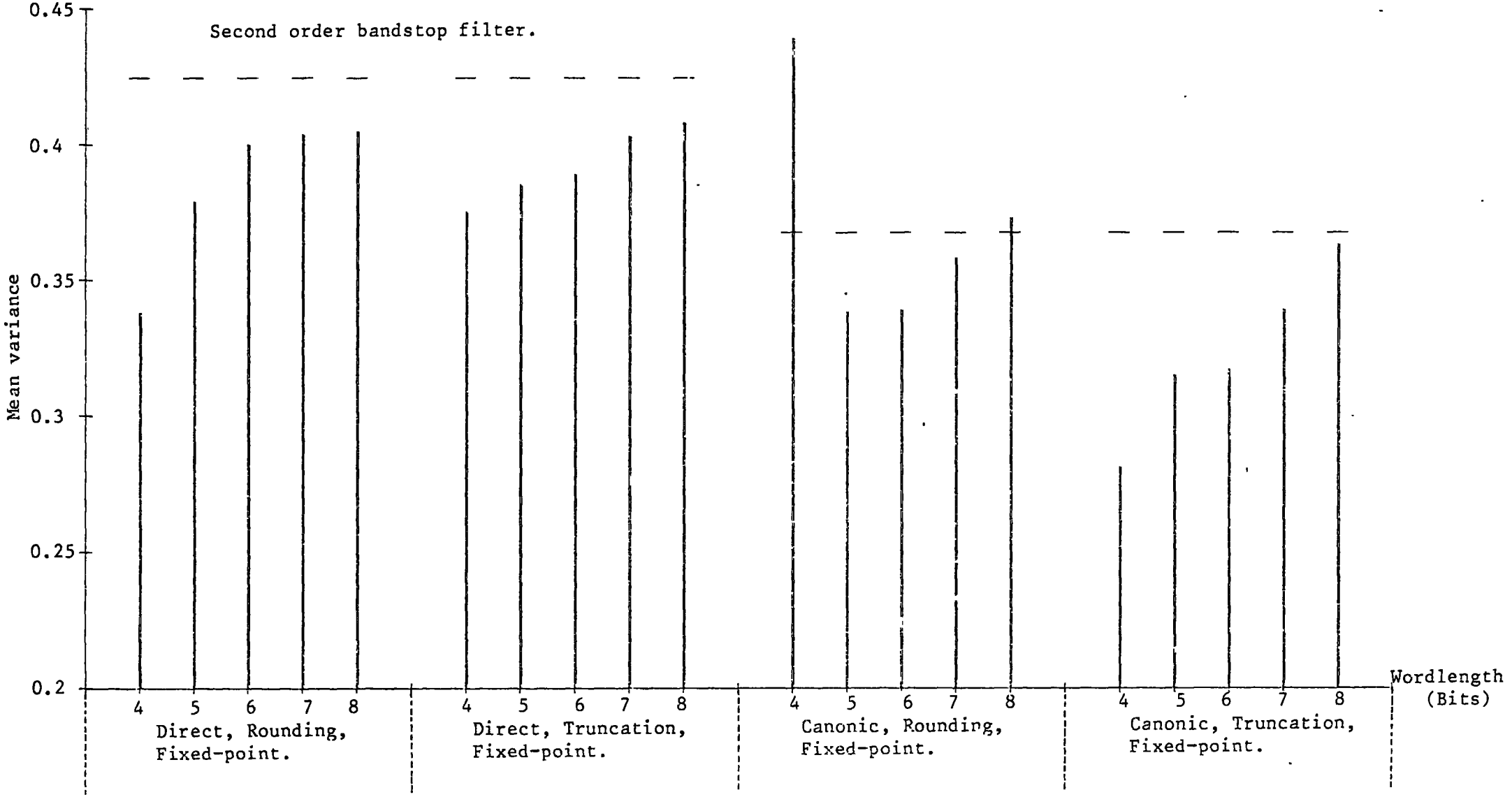


Figure 5.7(b) Results of error variance measurements.

Second order bandstop filter.

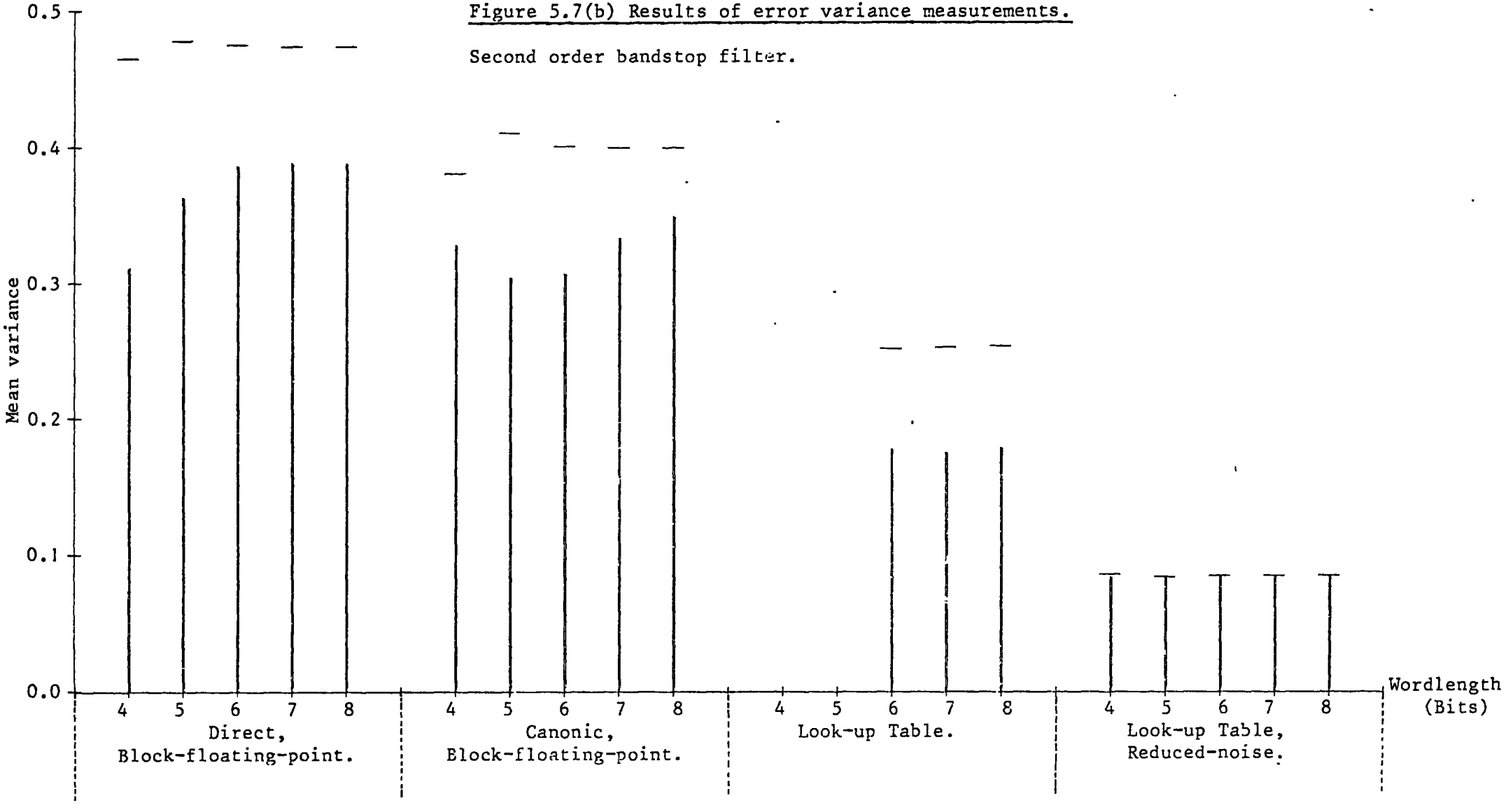


Table 5.7 Error variance measurements.

Second order bandstop filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.	0.338 ± 0.002	0.379 ± 0.002	0.400 ± 0.002	0.404 ± 0.002	0.405 ± 0.002
Direct, Truncation, Fixed-point.	0.375 ± 0.002	0.385 ± 0.002	0.389 ± 0.002	0.403 ± 0.002	0.408 ± 0.002
Canonic, Rounding, Fixed-point.	0.439 ± 0.002	0.338 ± 0.002	0.339 ± 0.002	0.358 ± 0.002	0.373 ± 0.002
Canonic, Truncation, Fixed-point.	0.281 ± 0.002	0.315 ± 0.002	0.317 ± 0.002	0.339 ± 0.002	0.363 ± 0.002
Direct, Block-floating-point, Experimental.	0.311 ± 0.002	0.363 ± 0.002	0.386 ± 0.002	0.388 ± 0.002	0.388 ± 0.002
Theory.	0.46464	0.47777	0.47531	0.47431	0.47382
Canonic, Block-floating-point, Experimental.	0.328 ± 0.002	0.304 ± 0.002	0.307 ± 0.002	0.333 ± 0.002	0.349 ± 0.002
Theory.	0.38115	0.41100	0.40115	0.40036	0.39959
Look-up Table, Experimental.			0.1778 ± 0.0008	0.1755 ± 0.0008	0.1793 ± 0.0008
Theory.			0.25203	0.25336	0.25402
Look-up Table, Reduced-noise, Experimental.	0.0841 ± 0.0003	0.0839 ± 0.0003	0.0847 ± 0.0003	0.0848 ± 0.0003	0.0851 ± 0.0003
Theory.	0.08571	0.08395	0.08519	0.08492	0.08492

Theory: Direct, Fixed-point 0.42448
Canonic, Fixed-point 0.36775

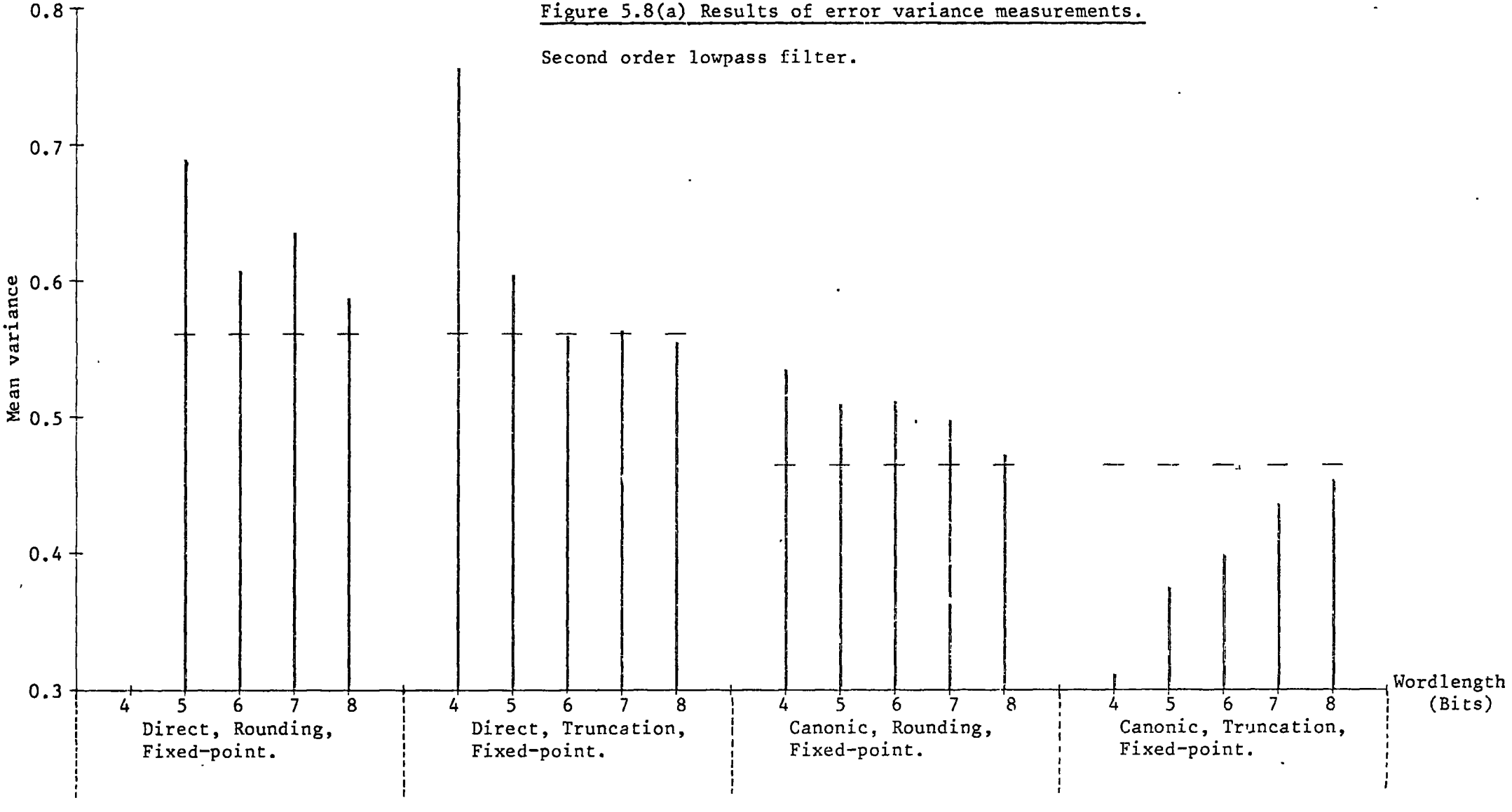


Figure 5.8(a) Results of error variance measurements.

Second order lowpass filter.

Figure 5.8(b) Results of error variance measurements.

Second order lowpass filter.

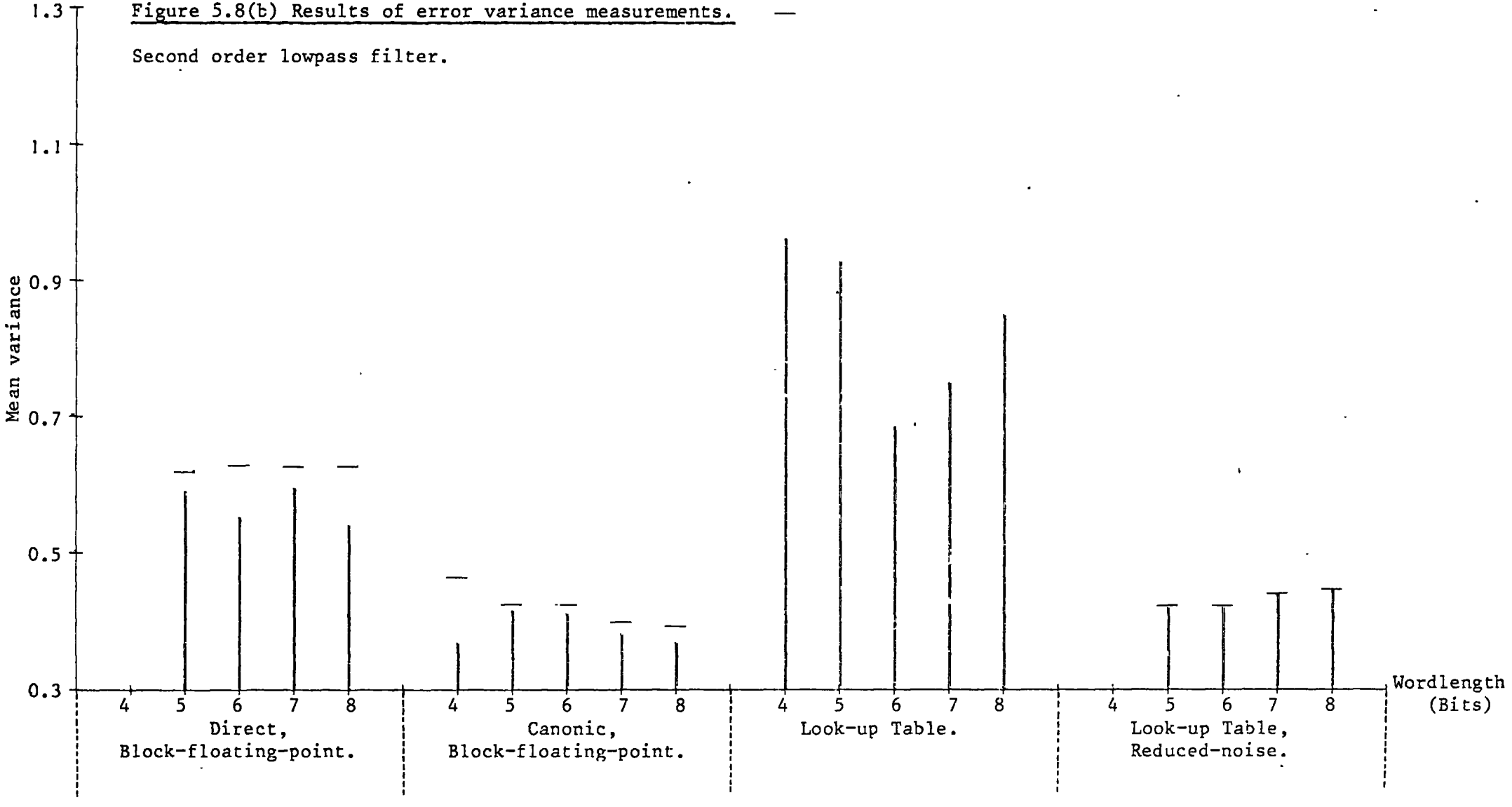


Table 5.8 Error variance measurements.

Second order lowpass filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.		0.689 ± 0.005	0.607 ± 0.004	0.635 ± 0.004	0.587 ± 0.003
Direct, Truncation, Fixed-point.	0.756 ± 0.005	0.604 ± 0.004	0.559 ± 0.004	0.563 ± 0.004	0.554 ± 0.003
Canonic, Rounding, Fixed-point.	0.534 ± 0.004	0.509 ± 0.004	0.511 ± 0.003	0.497 ± 0.003	0.471 ± 0.003
Canonic, Truncation, Fixed-point.	0.311 ± 0.002	0.374 ± 0.002	0.398 ± 0.002	0.435 ± 0.002	0.453 ± 0.003
Direct, Block-floating-point, Experimental.		0.590 ± 0.004	0.553 ± 0.004	0.594 ± 0.004	0.540 ± 0.004
Theory.		0.61892	0.62786	0.62652	0.62593
Canonic, Block-floating-point, Experimental.	0.369 ± 0.002	0.415 ± 0.003	0.411 ± 0.003	0.381 ± 0.003	0.369 ± 0.003
Theory.	0.46449	0.42397	0.42344	0.39899	0.39228
Look-up Table, Experimental.	0.960 ± 0.006	0.927 ± 0.006	0.684 ± 0.004	0.749 ± 0.004	0.849 ± 0.005
Theory.	1.2901	1.3181	1.3321	1.3392	1.3427
Look-up Table, Reduced-noise, Experimental.		0.418 ± 0.002	0.418 ± 0.002	0.440 ± 0.002	0.447 ± 0.003
Theory.		0.42198	0.42198	0.44012	0.44567

Theory: Direct, Fixed-point 0.56090
Canonic, Fixed-point 0.46446

point arithmetic is the only one to give less noise than the modified look-up table form, and the latter maintains its excellent correlation with theoretical expectations.

5.2.2.8 Second order highpass filter.

Figures 5.9(a) and (b) and Table 5.9 record the results obtained for this filter, which theoretically has a response which is the mirror-image of that for the lowpass filter considered in the previous paragraph. In practice too the performances of the various realisations bear a very high degree of similarity to those for the lowpass filter. This is not true, however, for the look-up table forms; rescaling of the filter output is no longer required, enabling these realisations to produce low noise levels, especially those which have been modified for the purpose.

5.2.3 Discussion of the experimental results.

Having noted the experimental results in some detail it is now necessary to discuss their general implications and hence decide what conclusions may reasonably be drawn from them. The most significant experimental finding is the generally large discrepancy between the experimental results obtained and the theoretically predicted error variances; this must be given careful consideration and explanation. First, however, a few other points arising from the experimental results should be discussed.

5.2.3.1 Block-floating-point arithmetic.

The results obtained when using block-floating-point arithmetic appear to contradict Oppenheim's own findings[16]. The results just reported do not contain a single instance where the use of this arithmetic mode has caused a really significant noise reduction. Furthermore in

Figure 5.9(a) Results of error variance measurements.

Second order highpass filter.

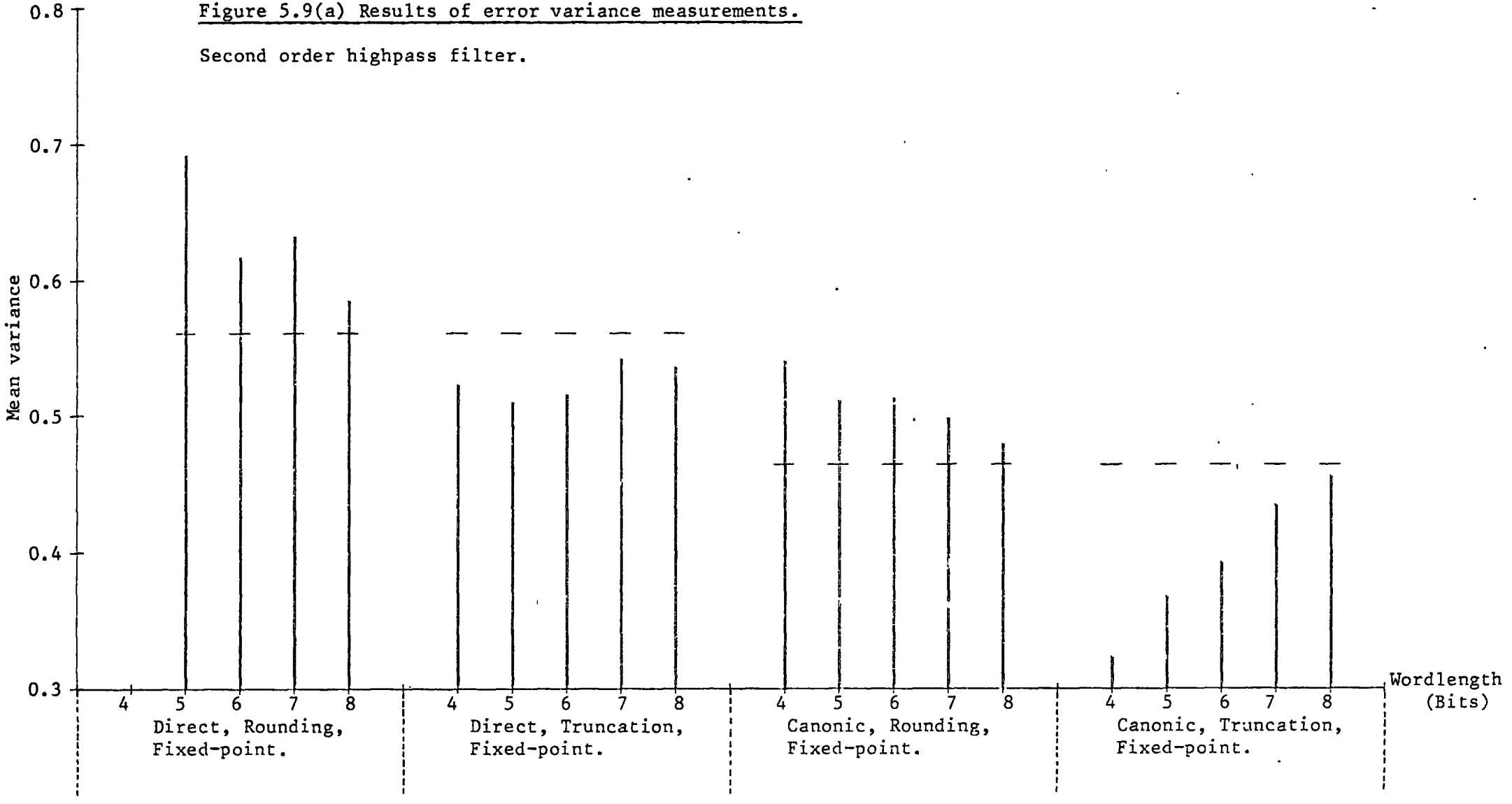


Figure 5.9(b) Results of error variance measurements.

Second order highpass filter.

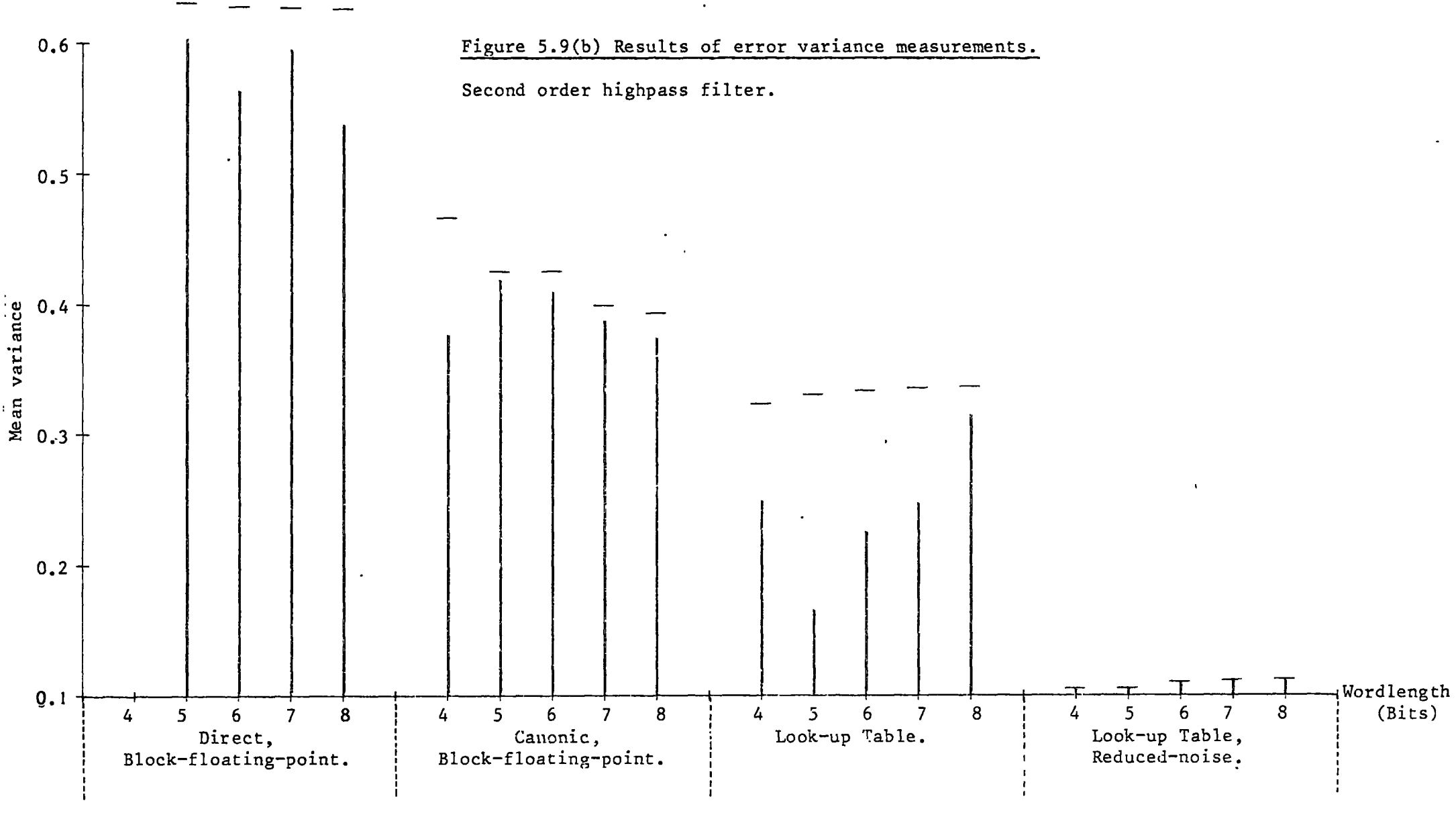


Table 5.9 Error variance measurements.

Second order highpass filter.

Realisation	Wordlength (Bits).				
	4	5	6	7	8
Direct, Rounding, Fixed-point.		0.692 ± 0.004	0.617 ± 0.004	0.632 ± 0.005	0.585 ± 0.003
Direct, Truncation, Fixed-point.	0.523 ± 0.003	0.510 ± 0.003	0.516 ± 0.003	0.542 ± 0.003	0.536 ± 0.004
Canonic, Rounding, Fixed-point.	0.540 ± 0.004	0.511 ± 0.003	0.513 ± 0.003	0.498 ± 0.003	0.479 ± 0.003
Canonic, Truncation, Fixed-point.	0.323 ± 0.002	0.367 ± 0.002	0.392 ± 0.003	0.434 ± 0.002	0.455 ± 0.003
Direct, Block-floating-point, Experimental.		0.603 ± 0.004	0.563 ± 0.004	0.594 ± 0.004	0.537 ± 0.004
Theory.		0.63072	0.62794	0.62660	0.62601
Canonic, Block-floating-point, Experimental.	0.376 ± 0.003	0.418 ± 0.003	0.409 ± 0.003	0.387 ± 0.003	0.373 ± 0.002
Theory.	0.46551	0.42506	0.42484	0.39946	0.39289
Look-up Table, Experimental.	0.249 ± 0.002	0.1647 ± 0.0008	0.225 ± 0.002	0.247 ± 0.002	0.314 ± 0.002
Theory.	0.32252	0.32953	0.33304	0.33479	0.33566
Look-up Table, Reduced-noise, Experimental.	0.1039 ± 0.0005	0.1056 ± 0.0005	0.1099 ± 0.0005	0.1116 ± 0.0005	0.1113 ± 0.0006
Theory.	0.10549	0.10549	0.11003	0.11142	0.11174

Theory: Direct, Fixed-point 0.56090
Canonic, Fixed-point 0.46446

many of the cases tested the theory predicts a degradation in the performance, although this never really materialises because of the large discrepancies between theory and practice already noted. It is important to consider firstly, why this arithmetic mode is so ineffective, and secondly, why the theoretical predictions deviate so markedly from the actual findings.

The ability of this arithmetic mode to yield reduced noise levels depends on its scaling up of the signals within a filter to make fuller use of the available signal wordlength. Equations (3.6.18), (3.6.21) and (3.6.22) indicate that the predicted noise variance is approximately proportional to k^2, the mean squared rescaling factor at the filter output. Oppenheim records values of k^2 which range from 0.0137 to 0.1162 for his first order filters, and from 0.00151 to 0.15527 for the second order realisations. In contrast the k^2 values for the results presented above vary from 0.5742 to 0.7897 for the first order forms, and from 0.6337 to 0.9239 for the second order implementations. That is, much higher levels of signal scaling were achieved within Oppenheim's filters than was the case for those tested above, which explains the predicted ineffectiveness of the arithmetic mode on these latter filters.

This marked discrepancy in the degree of signal scaling obtained can be traced to two important differences between the sets of experiments. The first is that Oppenheim tested filters which possessed only multipliers to define pole positions; and the second that, to avoid output signal overflow, he restricted the amplitude of the input white noise sequences to 1/16 and 1/128 of that allowed by the available wordlength for first and second order filters respectively. Presumably this high degree of prescaling was found to be necessary in order to accomodate high instantaneous signal levels in the output. Calculation shows, however, that the r.m.s. output signal levels obtained from Oppenheim's filters are generally very low and make exceedingly poor use

of the available signal wordlength, so that the inherent signal to noise ratio is relatively low. Hence plenty of scope is allowed for significantly increasing the signal levels within the filter, and thus achieving a high degree of noise reduction and a corresponding improvement in the signal to noise ratio. In comparison the filter designs presented in Appendix 5 and the input amplitudes used are intended to result in relatively high general signal levels which, for a given pole-zero map, make the fullest use possible of the available register space, so that the inherent signal to noise ratio is relatively high. There is, therefore, little opportunity for increasing the general signal levels within the filters, and thus reducing the absolute noise level and improving the signal to noise ratio.

The degree of signal scaling possible also depends on the number of signals which have to be considered when determining the instantaneous scaling factors, A_n; in general the larger the number of signals taken into account, the smaller will be the mean scaling level obtained. Oppenheim's first order filters consider two signals, the current input and the delayed output, but the input amplitude is so restricted that scaling is effectively only controlled by the delayed output sequence. His second order filters consider three signals, the current input and the two delayed outputs, but similarly this can be reduced by one. The first order filters presented in Appendix 5 were designed to process input sequences of the full amplitude allowed by the signal wordlength, so scaling is controlled both by the input and by the delayed output sequence. The addition of a zero-section to the second order filters requires that scaling be dependent on 5 signals for a direct realisation and 3 for the canonic form. This is the second reason for the low levels of signal scaling obtained.

Having discussed the theoretical performance of block-floating-point arithmetic, the obvious lack of agreement between theory and

practice needs to be considered. Again the low level of signal scaling can be shown to be at the source of the problem, this time by causing some parts of the equivalent noise models for block-floating-point arithmetic filters to become invalid. (Figures 3.12 and 3.13 depict the equivalent noise models.) The values of k^2 obtained indicate that on the majority of occasions the instantaneous rescaling factor, $1/A_n$, is unity. However, for a significant proportion of the time, some 20% for a typical k^2 of 0.8, the factor is $\frac{1}{2}$. Factors of less than $\frac{1}{2}$ can be reasonably assumed to occur with negligible frequency. The model states that the final scaling by $1/A_n$ yields a roundoff error sequence with the usual variance of σ_ε^2, that is, 1/12 for Δ equal to unity. If, however, $1/A_n$ is unity, no roundoff occurs, and if it is $\frac{1}{2}$ then an error is only incurred if the sample being rescaled is an odd integer. Hence roundoff probably only occurs about 10% of the time, giving rise to a roundoff error sequence with an approximate variance of only 1/40. A low level of signal scaling also has implications for the roundoff error sequences created at the δ_n multipliers in the filter pole-sections. Continuing the assumption that the instantaneous scaling factor A_n only takes the values 1 and 2, the δ_n multipliers can only have the magnitudes $\frac{1}{2}$, 1 and 2. Roundoff can only occur when δ_n is $\frac{1}{2}$ and then only if the sample being scaled is an odd integer. Again a value of 1/40 seems a reasonable approximation for the variance of error sequences created at the δ_n multipliers. Such rough calculation is sufficient to demonstrate the inadequacy of the model in situations where only a low level of signal scaling can be obtained. It also indicates that there is good reason for the noise levels practically achieved to be significantly less than those predicted.

In conclusion, it seems reasonable to suggest that block-floating-point arithmetic is not of great significance. Its effectiveness and predictability depend too greatly on the level of signal scaling achieved

within a filter, and this can only be roughly estimated before the filter is actually implemented. Moreover, it is thought that filters and conditions, such as Oppenheim's, which allow block-floating-point arithmetic to yield a significant, predictable noise reduction, would seldom be found in practice. For these reasons the consideration of this arithmetic mode is carried no further in this work.

5.2.3.2 Look-up table forms.

One of the interesting features of the results obtained for the look-up table filter realisations is that the normal form as suggested by Peled and Liu[60] gives generally poor agreement between experiment and theory, whereas the results of the form modified to give a reduction in noise correlate extremely well with the theoretical predictions. As previously mentioned, this modification involves storing ψ values in the table which do not contain any roundoff error. This suggests that the theoretical noise model for the unmodified look-up table realisation does not adequately deal with the roundoff errors in the stored ψ values. Reference to §3.7.1 indicates that these roundoff errors are taken to yield an error sequence $\{e_n''\}$ with the customary $\Delta^2/12$ variance. This is not necessarily true, however, as demonstrated by the so-called reduced-noise modification which is nothing more than a special case where $\{e_n''\}$ has zero variance. Some thought shows that the variance of this sequence is dependent both on the set of ψ values stored and on the relative frequency with which each member of the set is accessed. Given the condition of a random input sequence, it is reasonable to assume that each table location has an equal probability of being addressed. This allows the calculation of an error sequence variance for a particular set of ψ values, which should enhance the accuracy of the model.

No such problems arise, however, when using the reduced-noise modification of the look-up table form. Such realisations yield

consistently excellent results both in terms of low noise and of the degree of theoretical agreement. The coefficient quantisation involved in achieving this modified form should not be a problem in the majority of applications; it is no more severe than the coefficient quantisation which would normally be imposed in the practical realisation of a conventional filter form. The ease of hardware implementation demonstrated by Peled and Liu[60] is obviously unchanged for the modified form. Indeed the look-up table form is particularly well suited to realisation on a small programmable device such as a microprocessor. For all these reasons the reduced-noise modification of the look-up table filter form is to be strongly recommended.

5.2.3.3 Uniform error distribution.

One of the main assumptions of the theoretical model is that all roundoff error magnitudes within the allowed limits have an equal probability of occurrence[29]. Hence when averaged over a long time the error distribution obtained at any multiplier should be as shown in Figure 3.4 for rounding and Figure 3.5 for truncation. If this is the case, then the variance of the roundoff error about the mean is $\Delta^2/12$. However, the previous two sections both consider situations where this is not true so that the assumption of uniformly distributed roundoff error sequences produces inaccurate theoretical predictions. It seems reasonable, therefore, to try to discover whether this assumption is likely to be at the root of the discrepancies obtained between experiment and theory for the conventional fixed-point arithmetic direct and canonic form filters.

5.3 Error variance at single multipliers.

It is necessary to return to the examination of roundoff error sequences at single multipliers begun in the previous chapter, in order

to discover whether or not uniform error distributions are achieved in a practical situation. This may be done by measuring the variance of a roundoff error sequence and conparing it with the $\Delta^2/12$ which would result from a uniform distribution of errors.

5.3.1 The measurement of error variance.

The fundamental method is the two channel processing technique as employed in the experiment represented in Figure 5.1. However, on this occasion the two processing channels contain single multipliers as in Chapter 4. It is desired to measure the error variance at a given multiplier as a function of the input sequence amplitude, as this will show any effect of signal wordlength on the error distribution. As before the input samples are to be uniformly distributed within the allowed amplitude limits. Because a single multiplier has no memory of previous signals, unlike a filter, the particular ordering of the input sequence is meaningless. Hence there is no need to use a random signal generator. The input sequence used is in fact a positive integer ramp from 0 to 127 in steps of one, thus examining wordlengths up to 8 bits inclusive. As rounding is the only error process to be tested, there is no need for the input ramp to go negative as this would duplicate information, because of the symmetrical roundoff process. The zero sample is given half the weight of the other ramp levels in order to simulate a uniformly distributed symmetrical input sequence. At each step of the input ramp the accumulating roundoff is calculated and its variance computed. Because the use of a ramp yields an absolutely uniformly distributed input sequence, and because the ordering of the input sequence has no effect on the measured error variance, there is nothing to be gained from multiple runs of this experiment. The main program, VARAMP, used to conduct this investigation is listed in Appendix 6.

5.3.2 Results of error variance measurements.

The results of testing three 10-bit coefficients are presented graphically in Figures 5.10 to 5.12. The three multipliers considered are 11/1024, 111/1024, and 411/1024, that is, those also examined in the previous chapter. The error variance values are again absolute and so should be compared to the theoretical 1/12 (0.0833') expected of uniformly distributed roundoff error sequences.

The conclusions to be drawn from the results are threefold. First, the error variance is seen to be dependent on the input amplitude in all three cases. Second, the deviation of the results from 1/12 is progressively reduced with increasing input amplitude. Finally, the results for the smallest coefficient, shown in Figure 5.10, show the highest degree of variation from the ideal; this deviation decreases with the successive increases of coefficient value. Because of the symmetrical nature of the rounding process a coefficient (1-x) produces the same error variance as a coefficient x, for a given input sequence. It may therefore be concluded that the possible deviation of the error variance from the ideal value increases as the coefficient approaches zero or unity. This inference, however, takes no account of the possible effects of variations in coefficient wordlength.

5.3.3 Explanation of results.

Inspection of Figures 5.10 to 5.12 indicates that the points do not deviate haphazardly from the ideal, but that for a given coefficient there is a definite mechanism connecting the input sequence amplitude and the error variance. The nature of this relationship must briefly be considered.

If a two's complement signal sample is multiplied by a 10-bit unsigned fraction, and the result rounded to the signal wordlength, then a 10-bit error remains. Hence there are 2^{10} (1024) discrete error levels

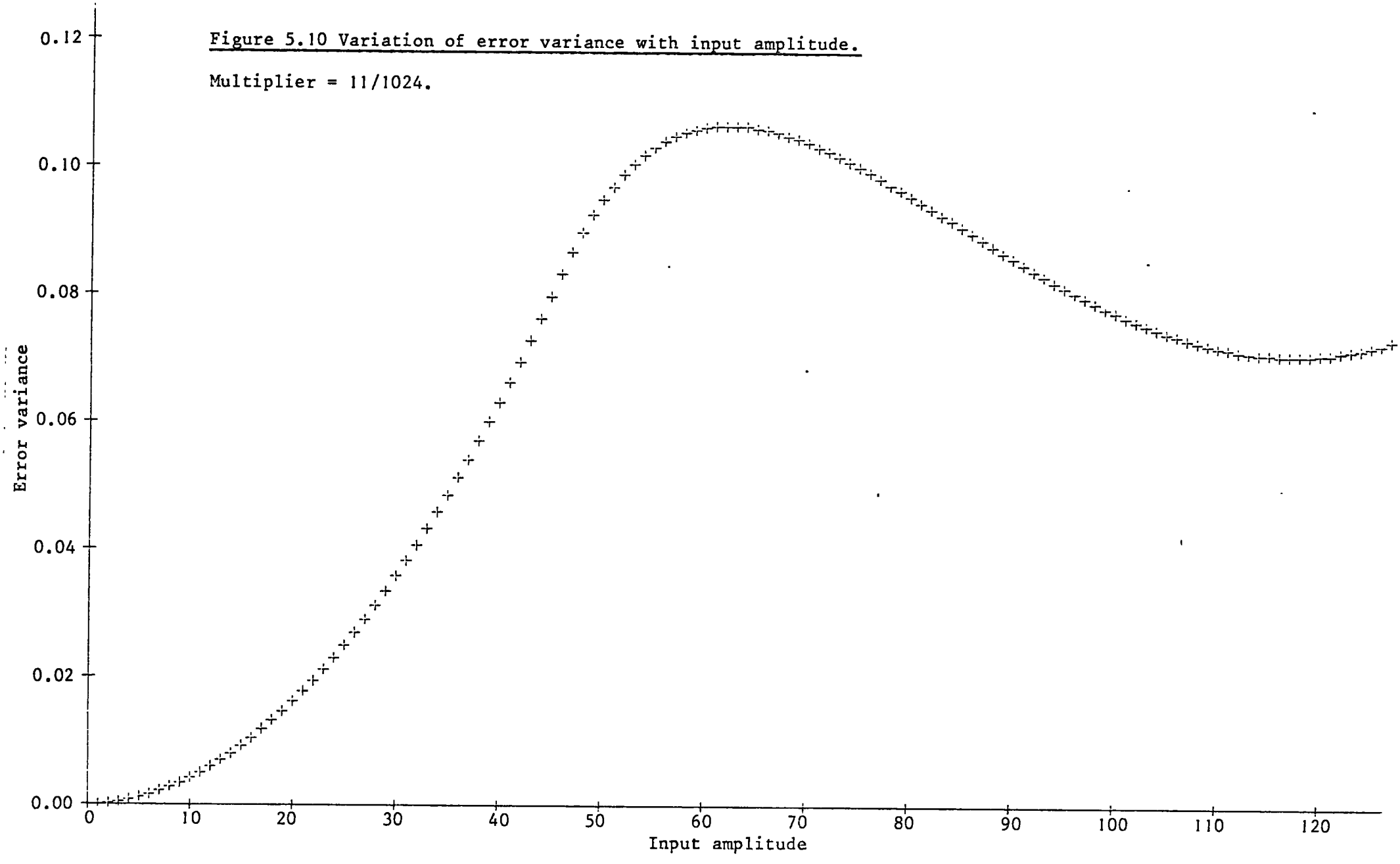


Figure 5.10 Variation of error variance with input amplitude.

Multiplier = 11/1024.

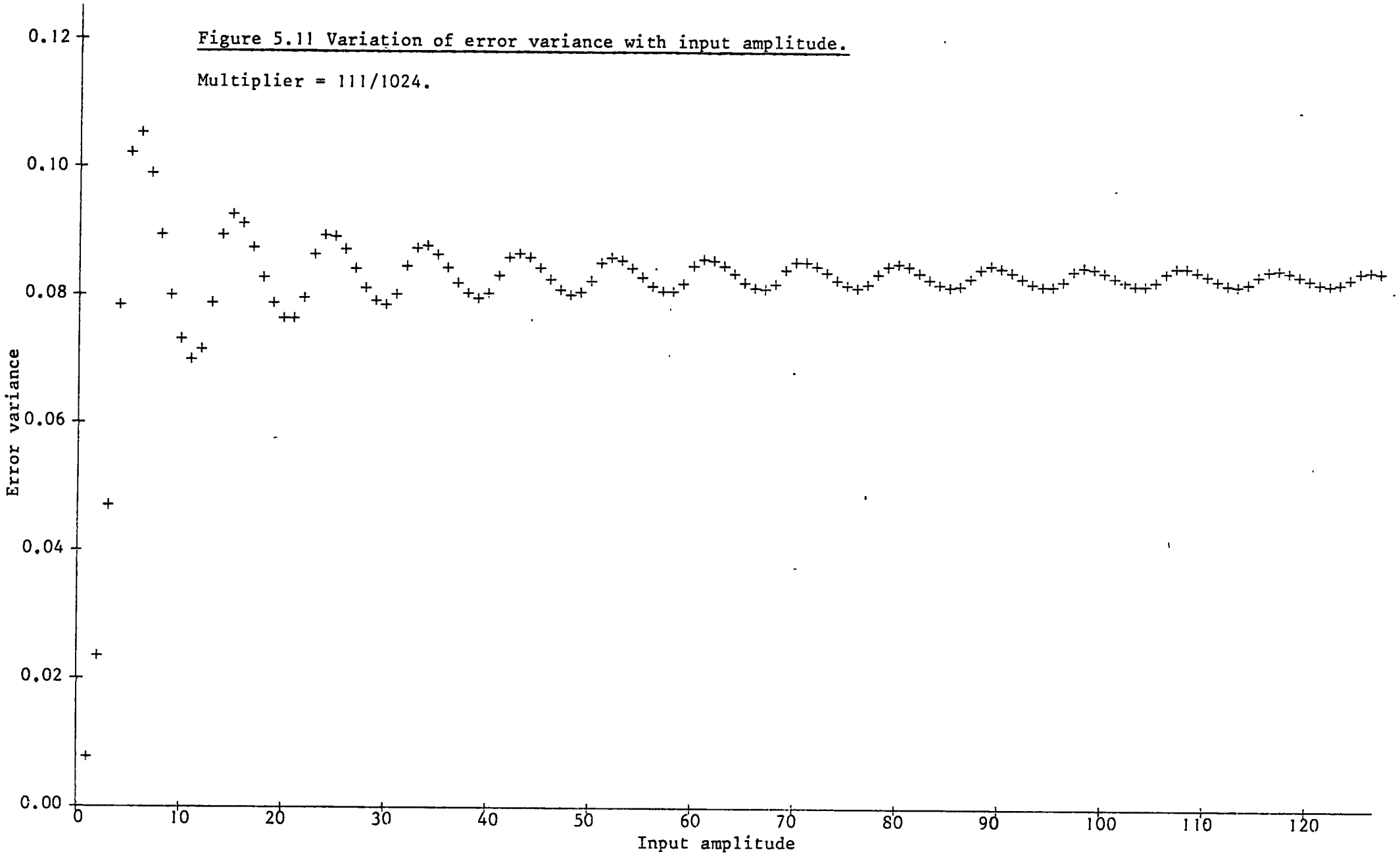


Figure 5.11 Variation of error variance with input amplitude.

Multiplier = 111/1024.

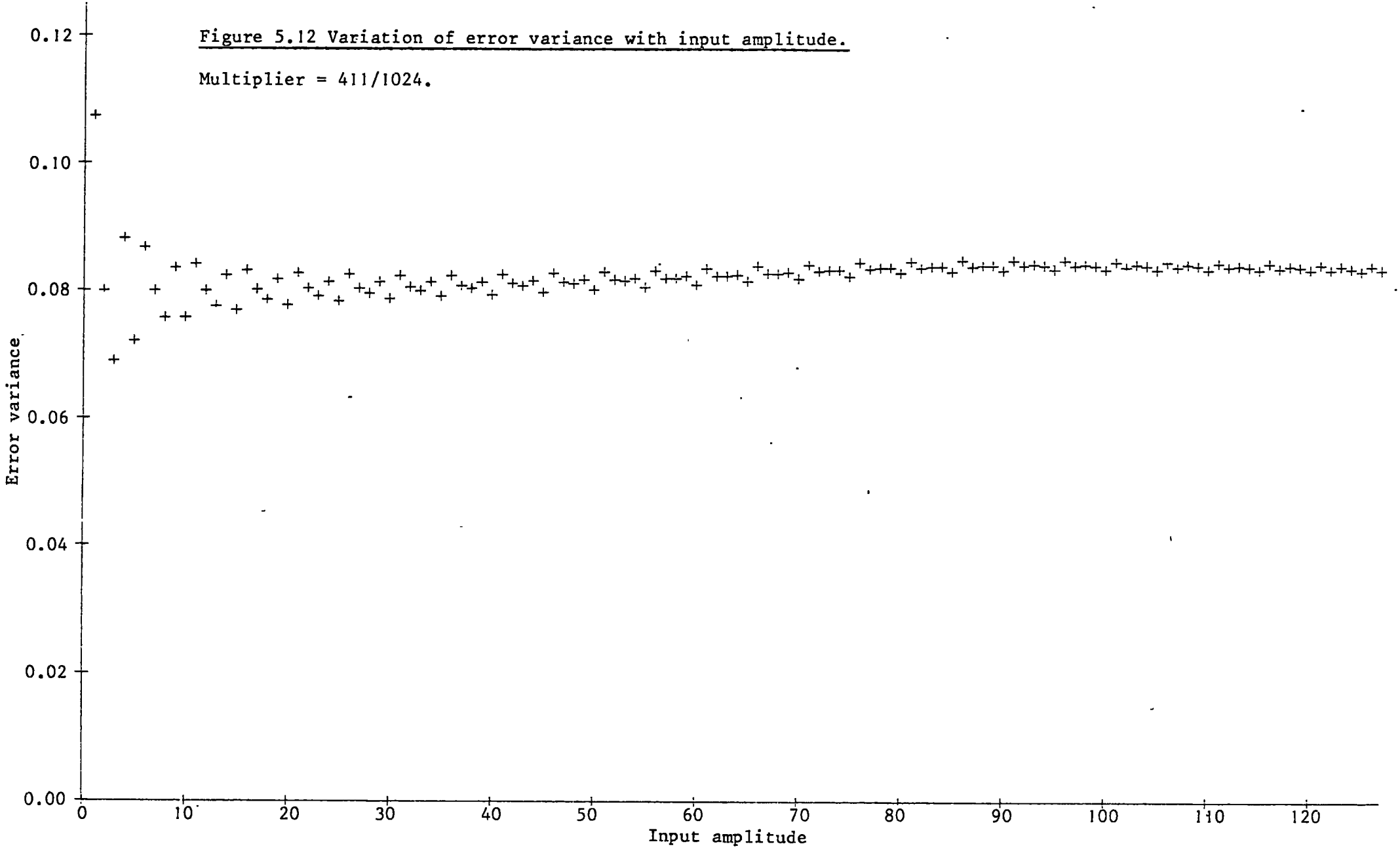


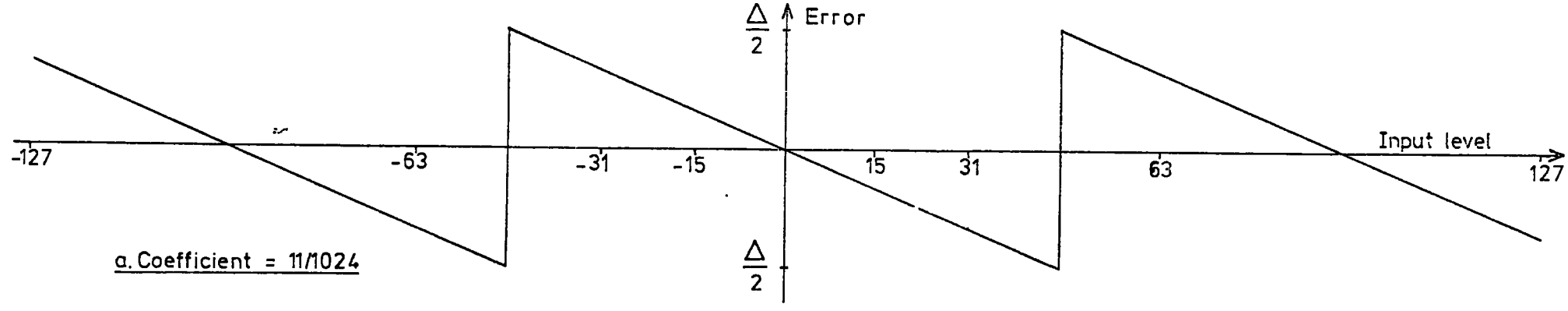
Figure 5.12 Variation of error variance with input amplitude.

Multiplier = 411/1024.

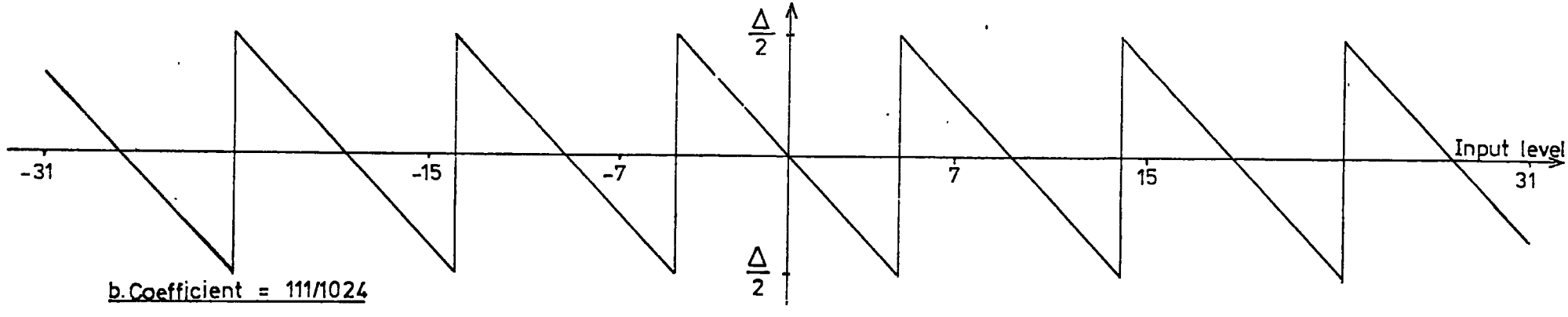
within the permitted range of $-\Delta/2$ to $+\Delta/2$. If all these error levels were to be used once then a uniformly distributed input sequence with an amplitude of 511 would be required, that is, the maximum amplitude permissible for 10-bit signals. Hence for the wordlengths of 8 bits and less being considered only a relatively small proportion of the error levels are ever occupied. This does not inherently cause deviation of the error variance from $\Delta^2/12$; it is the distribution of the magnitudes of the occupied error levels which is all important. If all the occupied error levels are of relatively low magnitude then the roundoff error variance will also be low, and vice versa.

Figure 5.13 presents the error functions for two of the coefficient values tested, 11/1024, and 111/1024. The magnitudes of the discrete error levels are found by sampling the curves at integral values of input level. Figure 5.13(a), for the coefficient 11/1024, shows the error level magnitude rising from zero as the input level is increased in the region 0 to 46. Hence the error variance, as shown in Figure 5.10, also rises progressively above zero with the input amplitude over this range. After the discontinuity in the error function, at a theoretical input of 512/11 (just over 46), error magnitudes start to decrease, but the error variance does not begin to drop until the squared magnitude of the roundoff error falls below the current variance, which occurs at an input amplitude of 65. For input samples in excess of 93 the error magnitude once more begins to increase, which eventually causes the error variance to start rising again at an input amplitude of 120. With each increment in the input sequence amplitude comes the occupation of a new error level. This has the effect of damping the oscillation of the error variance so that it converges towards the ideal with increasing input amplitude. Figure 5.13(b) is the error function for the coefficient 111/1024. (Note that the input level scale is different from Figure 5.13(a)). This has the same basic features, but differs in what may be

Figure 5.13 Error functions



a. Coefficient = 11/1024



b. Coefficient = 111/1024

termed its 'periodicity', which for a coefficient value $|x| \leq \frac{1}{2}$ may be defined to be 1/x. Figure 5.11 shows the increased rate of oscillation of the error variance which arises from the decreased periodicity of the error function of Figure 5.13(b). Not only is the rate of oscillation affected, but its initial amplitude is also reduced. Hence the error variance comes within a given tolerance about the ideal at a lower input amplitude than with the smaller multiplier value. Figure 5.12, for the coefficient value 411/1024, demonstrates the progression of these trends.

The particular shapes, however, of Figure 5.10 to 5.12 are not of great significance. Their purpose is to demonstrate that the error variance can deviate widely from the ideal, and to indicate the effects of coefficient value and signal wordlength. The important conclusion is that the error variance at a single multiplier cannot safely be considered to be a constant. Moreover it should be clear that in choosing to test with uniformly distributed input sequences, one particular case has been selected from the infinite number of possibilities. In general, the error variance at a given multiplier is a function of the amplitude distribution of the samples in its input sequence. If this is known, however, because of the one to one relationship between an input sample and an error level, the error variance can be determined.

5.4 Conclusions.

Throughout this chapter conclusions drawn from the experimental results have been noted. However, it is useful to collect these together here in summary.

First, with regard to the effectiveness of block-floating-point arithmetic[16], the results indicate that only under certain conditions, for example those considered by Oppenheim, is there a significant and predictable noise reduction. In general, however, this arithmetic mode

does not significantly diminish the absolute noise level, neither does it behave in the manner that the theoretical model assumes. This arithmetic mode is seldom likely to prove advantageous and because of the difficulty of accurately predicting its effectiveness it is not recommended for use.

Second, the usefulness of the look-up table filter form as suggested by Peled and Liu[60] is demonstrated. The value of modifying this form as previously proposed is also clearly indicated. This modification yields both a useful increase in the signal to noise ratio and a very high degree of agreement between experiment and theory. It is therefore worthy of some attention.

Finally, considering the direct and canonic fixed-point arithmetic filter forms, relatively large discrepancies between the experimental results and the theoretical predictions are seen to be general. The measurements of error variance at a single multiplier indicate that the assumption of uniform error distributions[29], fundamental to the theoretical model, is not justifiable at the signal wordlengths under examination. As the error variance at a multiplier is seen to be dependent on the amplitude distribution of the multiplier input sequence, a more accurate theoretical prediction should be permitted if these distributions can be determined and taken into account. The next chapter begins by examining this possibility.

CHAPTER 6 - PREDICTIVE NOISE MODELS.

6.1 Introduction.

The results reported in the previous chapter demonstrated the deficiencies of the predictive noise model as originally proposed by Knowles and Edwards[29], even under random signal conditions. In particular the weakness of the assumption of a uniform distribution of roundoff errors at each multiplier was shown. This chapter begins by considering and applying methods of predicting actual error distributions, again taking the random input sequence case, thus permitting the development of a more sophisticated noise model. Attention is restricted to what may be called distinct multiplier filter forms, as opposed to the look-up table realisation; fixed-point arithmetic is the only mode considered.

6.2 Error distribution prediction.

The unique correspondence between an input sample and an error level at a given multiplier allows the distribution of errors to be calculated easily once a prediction is made of the likely amplitude distribution of the input sequence. It should be clear that any attempt to predict the amplitude distributions of the input sequences to all the multipliers of a filter must involve some kind of assumption about the amplitude distribution of the input sequence to the filter. The nature of such an assumption must properly depend on the signal characteristics expected in a given application. Likely input signals may, in general, be classified into two categories, those with a uniform amplitude distribution, and those with a Gaussian distribution. The models tested in this chapter assume uniformly distributed input sequences, but the modification required to accomodate any given distribution is simple.

6.2.1 A model for first order filters.

The general difference equation for an ideal digital filter is

$$y_n = \sum_{i=0}^{k} a_i . x_{n-i} - \sum_{i=1}^{m} b_i . y_{n-i} \qquad (6.2.1)$$

This indicates that the current output y_n is dependent on the current and delayed input samples and on the delayed output samples. For a first order filter, however, the current output y_n is formed only from the current input x_n and the delayed output y_{n-1}. This delayed output is itself a function of x_{n-1} (and of course y_{n-2}), but as $\{x_n\}$ is taken to be a random sequence, x_n and y_{n-1} should be statistically independent. Such independence of the samples from which the current filter output is formed is unique to the first order case and leads to a method of predicting signal amplitude distributions which is not applicable to higher order filters.

If x_n and y_{n-1} are independent, and if given values of x_n and y_{n-1} have probabilities of occurrence ρ_x and ρ_y respectively, then the probability of the particular value of y_n being formed from these signal samples is $\rho_x . \rho_y$. In a practical filter a given value of y_n may be formed from a finite number of combinations of x_n and y_{n-1} values. Hence for a given output level y_n an equation can be set up which states that the probability of the given y_n is the sum of a number of joint probability terms each referring to a particular combination of x_n and y_{n-1} from which the given y_n may be formed. Such an equation can be written for each of the allowed output levels, that is, in general,

$$\rho_{y_i} = \sum_{m} \sum_{k} (\rho_{x_m} . \rho_{y_k}) \qquad (6.2.2)$$

where ρ_{y_i} is the probability of occurrence of output level i, and ρ_{x_m} is the probability of input level m. The summation is not performed over

all allowed values of m and k, but only those which give rise to the output level i.

At this stage the need to make an assumption about the distribution of the input sequence $\{x_n\}$ becomes clear. Once $\{x_n\}$ has been accredited with a particular distribution, the terms ρ_{x_m} can be given actual values. Hence the only unknowns in equation (6.2.2) are the ρ_y's. If there are N allowed output levels then N equations of the form of (6.2.2) can be written down for i=1 to N; these constitute N simultaneous equations relating the N variables $\rho_{y_1} \ldots \rho_{y_N}$. This set of simultaneous equations is unsuitable, however, because no constant term appears. That is, in the general matrix formulation of simultaneous equations

$$A\underline{x} = \underline{b} \tag{6.2.3}$$

where A is the coefficient matrix, $\underline{x}$ is the column vector of the variables, and $\underline{b}$ is the column vector of the constants, $\underline{b}$ consists only of zeros. This can be remedied by replacing one of the N equations of the form of (6.2.2) by the equation which states that the probabilities of all the permitted output levels must sum to unity:

$$\sum_{i=1}^{N} \rho_{y_i} = 1 \tag{6.2.4}$$

Hence the column vector $\underline{b}$ now contains the single constant unity in addition to (N-1) zeros, and the simultaneous equations (6.2.3) can be solved.

The probability values $\rho_{y_1} \ldots \rho_{y_N}$ which result from this solution constitute a definition of the amplitude distribution of the filter output sequence $\{y_n\}$ and therefore also of the input sequence $\{y_{n-1}\}$ to the multiplier b_1. Once the amplitude distributions of the input sequences at both filter multipliers have been predicted, the calculation

of the error distributions and resulting noise variances at each multiplier can easily proceed. The normal assumption that the error sequences at separate multipliers are statistically independent is retained, so the two error variances are summed to give the variance of the equivalent input error sequence. Whereas the simple model states that this sequence will have a variance of $2.(\Delta^2/12)$, the solution of the simultaneous equations permits a more accurate prediction of this variance for a given filter and input sequence amplitude distribution. The results of testing this model development are presented later in this chapter.

6.2.2 Models for higher order filters.

The prediction of signal amplitude distributions in higher order filters is much more complicated. Taking the example of a second order direct filter realisation, y_n is a function of x_n, x_{n-1}, x_{n-2}, y_{n-1} and y_{n-2}; y_{n-1} is a function of x_{n-1}, x_{n-2} and y_{n-2}; and y_{n-2} is a function of x_{n-2}. Clearly with such a degree of statistical interrelationship a method using simultaneous probability equations is out of the question.

It is not at all obvious whether, given a random input sequence $\{x_n\}$ with a particular amplitude distribution, there is a unique amplitude distribution for the corresponding output sequence $\{y_n\}$. Indeed the statistical interdependence just mentioned gives theoretical reason to expect that the amplitude distribution of $\{y_n\}$ should be dependent on the ordering of $\{x_n\}$ right from n=0, and also on the initial values given to x_{n-1}, x_{n-2}, y_{n-1} and y_{n-2} at n=0. How strong such dependence might prove to be in practice is not certain. The current filter output sample is much less dependent on the input and output samples at some considerably earlier stage in the sequences than on the most recent samples. This suggests that in practice, given sequences of sufficient length, the amplitude distribution of the output sequence should be

essentially independent of the initial conditions. By the same kind of argument, the dependence of the output sequence amplitude distribution on the ordering of the input sequence might be expected to diminish with increasing sequence length, so that it would converge to a shape dictated only by the amplitude distribution of the input sequence.

If it can be demonstrated in the future that the output sequence $\{y_n\}$ of a given second or higher order filter has a unique amplitude distribution for a random input sequence with a particular distribution, then the search for a method by which this may be predicted will be both justified and feasible. In the meantime it is possible to suggest some developments to the standard model which should result in more accurate noise predictions.

As stated in the previous chapter, given a 10-bit coefficient, a uniformly distributed input sequence with an amplitude of 511 is required for all the error levels to be used with equal frequency, and this cannot be true for signal wordlengths of 9 bits and less. It would seem reasonable to suggest, therefore, as a first stage of improvement, that the allowed signal amplitude be taken into consideration. In this case the roundoff errors may be assumed to be uniformly distributed among those error levels which can be occupied, so that each allowed output level must be assumed to have an equal probability of occurrence. As mentioned previously, the filter designs presented in Appendix 5 have theoretical gains just less than unity at their respective frequencies of maximum response. In consequence the output sequences from these filters have smaller variances than their corresponding random input sequences. During the investigation reported in Chapter 5, however, many of the output sequences obtained were found to have higher amplitudes than their uniformly distributed input sequences. This combination of a decrease in variance and an increase in amplitude can only be explained if the output sequences do not have uniform amplitude

distributions. The indication is that low magnitude output samples have a greater probability of occurrence than high magnitude samples. This suggests that it could be helpful to assume that a filter output sequence has a Gaussian amplitude distribution with a variance equal to that theoretically expected of the output sequence. In general, for a random input sequence with variance σ_x^2, the output variance σ_y^2 is given by

$$\sigma_y^2 = \sigma_x^2 \cdot \frac{1}{2\pi j} \cdot \oint_{|z|=1} H(z).H(z^{-1}).\frac{dz}{z} \qquad (6.2.5)$$

Obviously the distribution cannot be purely Gaussian as this theoretically has finite probabilities for all deviations up to infinity. However, the inaccuracy involved in neglecting the area under the tail of the Gaussian is almost certainly less than that incurred in making the assumption of a uniformly distributed output sequence. Whether or not the existence of a unique output sequence amplitude distribution can be demonstrated, it is suggested that the assumption of an appropriate Gaussian distribution can only improve the model accuracy.

6.2.3 Testing the model for first order filters.

The model development outlined in §6.2.1 needs to be tested experimentally to see whether or not its employment produces the agreement between theory and practice which the simple model lacks. Experimental error variance measurements are obtained by the method described in the previous chapter. Once more the signal wordlength range of 4 to 8 bits is under consideration. In the experiments recorded previously in this work effects have been found which depend in some way on the multiplier value involved. In order to detect any such effect in this investigation, first order highpass filters have been designed to use the whole range of possible coefficients. That is, the radius of the filter pole location in the z-plane, equal to the coefficient b_1, is

varied systematically within the limits 0 to 1. The theoretical maximum gain of a first order highpass filter, at the Nyquist frequency, is given by

$$G_{max} = \frac{a_0}{(1 - b_1)} \qquad (6.2.6)$$

This is made unity by setting a_0 equal to $(1-b_1)$ for all the filter realisations tested in this chapter. As before the coefficient values selected all require 10 bits for representation.

If both the filter input sequence and the roundoff process are symmetrical, then the number of simultaneous equations which have to be solved, to give the predicted amplitude distribution of the output sequence, can be halved. In other words, there is no need to set up equations for both positive and negative output levels. Because the computer time and space required to formulate and solve the simultaneous equations is basically proportional to the square of their number, this represents a considerable saving. For example, when calculating the predicted noise variance for an 8-bit signal wordlength filter, the computer memory allocation required for the coefficient matrix of the simultaneous equations is reduced from 64k to 16k 32-bit words. For this reason only filters using the symmetrical rounding process are tested. This should not, however, be taken to imply that there is any difficulty in implementing the model when the asymmetrical truncation process is employed, apart from the quantity of computer memory required; time is not a problem as the IBM 370/168 takes about 1 second of CPU time to return the predicted noise variance.

The subroutine SIMQ used for solving the simultaneous equations does not differ significantly from the IBM supplied subroutine of the same name. The algorithm employed is a standard successive elimination method; all calculation is performed in 32-bit floating-point arithmetic.

SETA is the subroutine which sets up the initial coefficient matrix A of equation (6.2.3). The algorithm involves considering all the possible combinations of x_n and y_{n-1} values which yield a given value of y_n, determining the probability of occurrence of the particular combination, and appropriately modifying the coefficient in the correct location of the matrix A. These two are the most important of several subroutines called by the main program AMPMOD during the execution of a noise variance prediction; all are listed in Appendix 6.

6.2.4 Experimental results.

The results of the investigation are presented in Figures 6.1 to 6.5, that is, one graph for each wordlength under consideration. The figures are plots of the deviation of the measured error variance from the predicted value for different filter responses. The asterisks refer to the predictions of the simple uniform error distribution model, whereas the triangles correspond to the more sophisticated model. Note that the scale of the x-axes changes at a pole radius of 0.8. The deviation of the experimental error variance from the model predictions as plotted in decibels is defined as

$$\delta N = \left| 10.\log_{10}\left(\frac{\sigma_E^2}{\sigma_P^2}\right) \right| \text{ dB} \qquad (6.2.7)$$

where σ_E^2 is the experimental error variance and σ_P^2 is the value predicted by the particular model. Standard errors were measured for all the mean experimental values of roundoff error variance but these are not presented; error bars would have an insignificant separation compared with the size of the symbols used on the graphs.

Figures 6.1 to 6.5 indicate that the use of the more sophisticated noise model, rather surprisingly, has little or no effect on the predicted values of noise variance. The graphs for 6-, 7-, and 8-bit

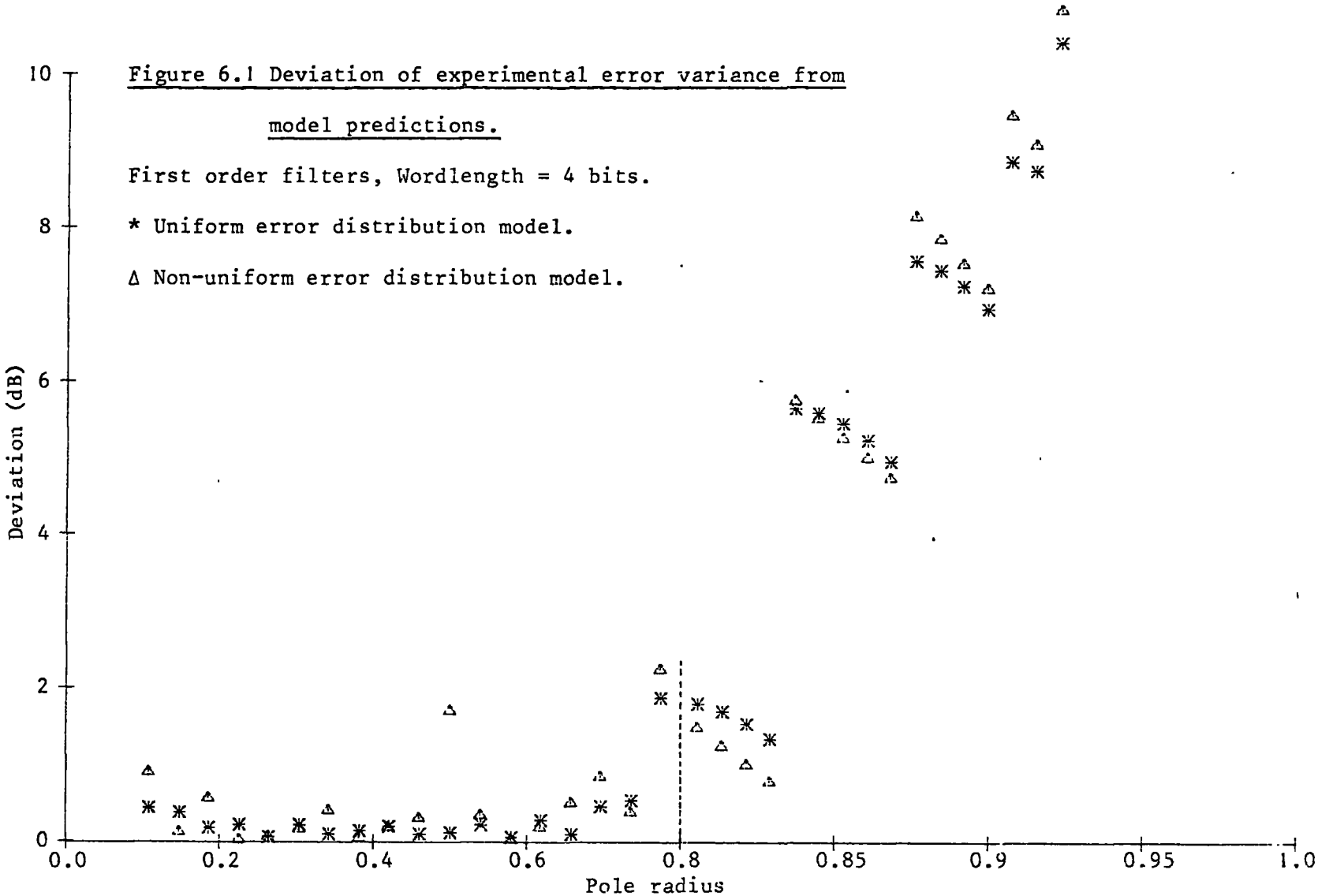


Figure 6.1 Deviation of experimental error variance from model predictions.

First order filters, Wordlength = 4 bits.

* Uniform error distribution model.

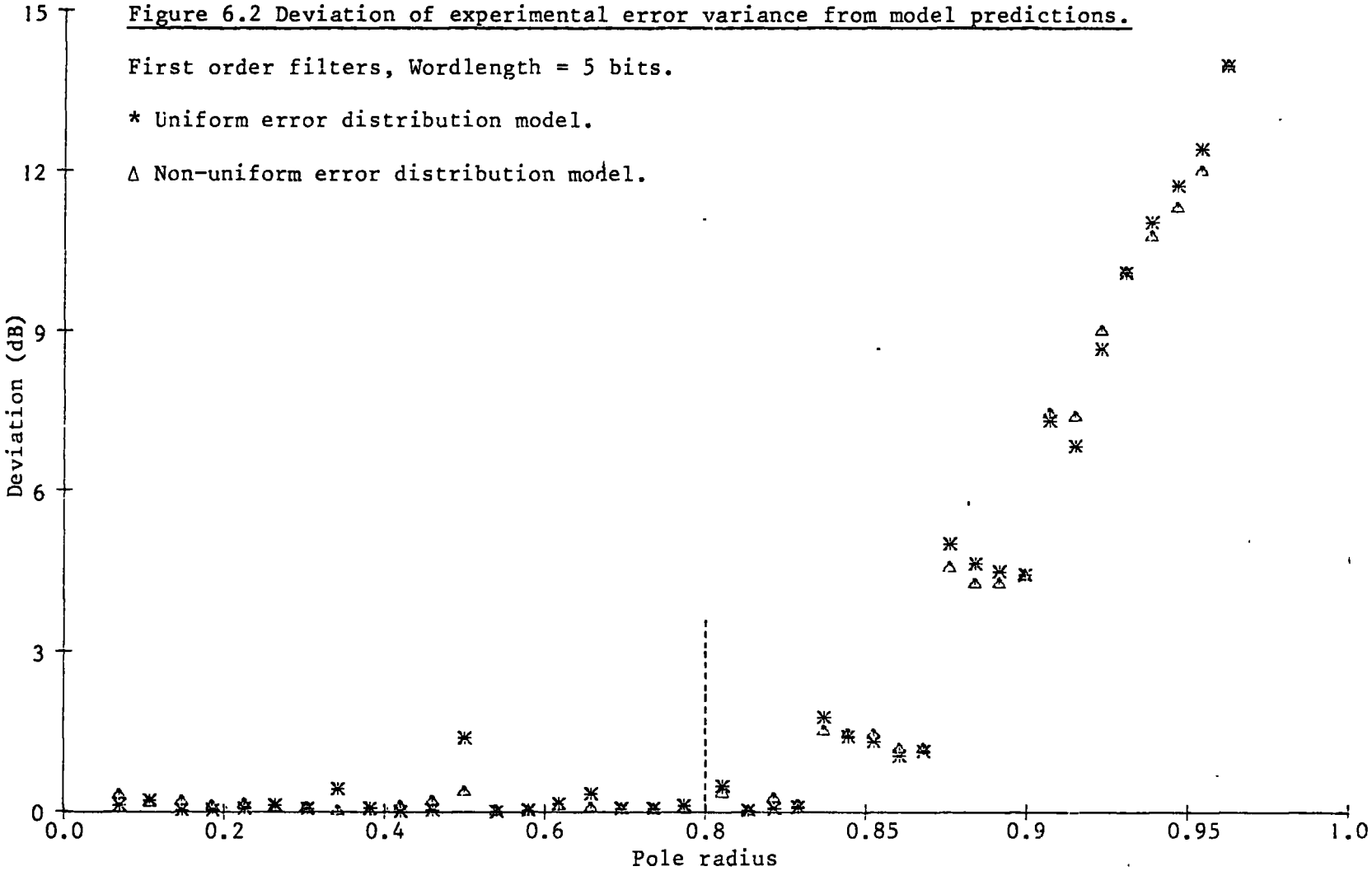
Δ Non-uniform error distribution model.

Figure 6.2 Deviation of experimental error variance from model predictions.

First order filters, Wordlength = 5 bits.

* Uniform error distribution model.

Δ Non-uniform error distribution model.



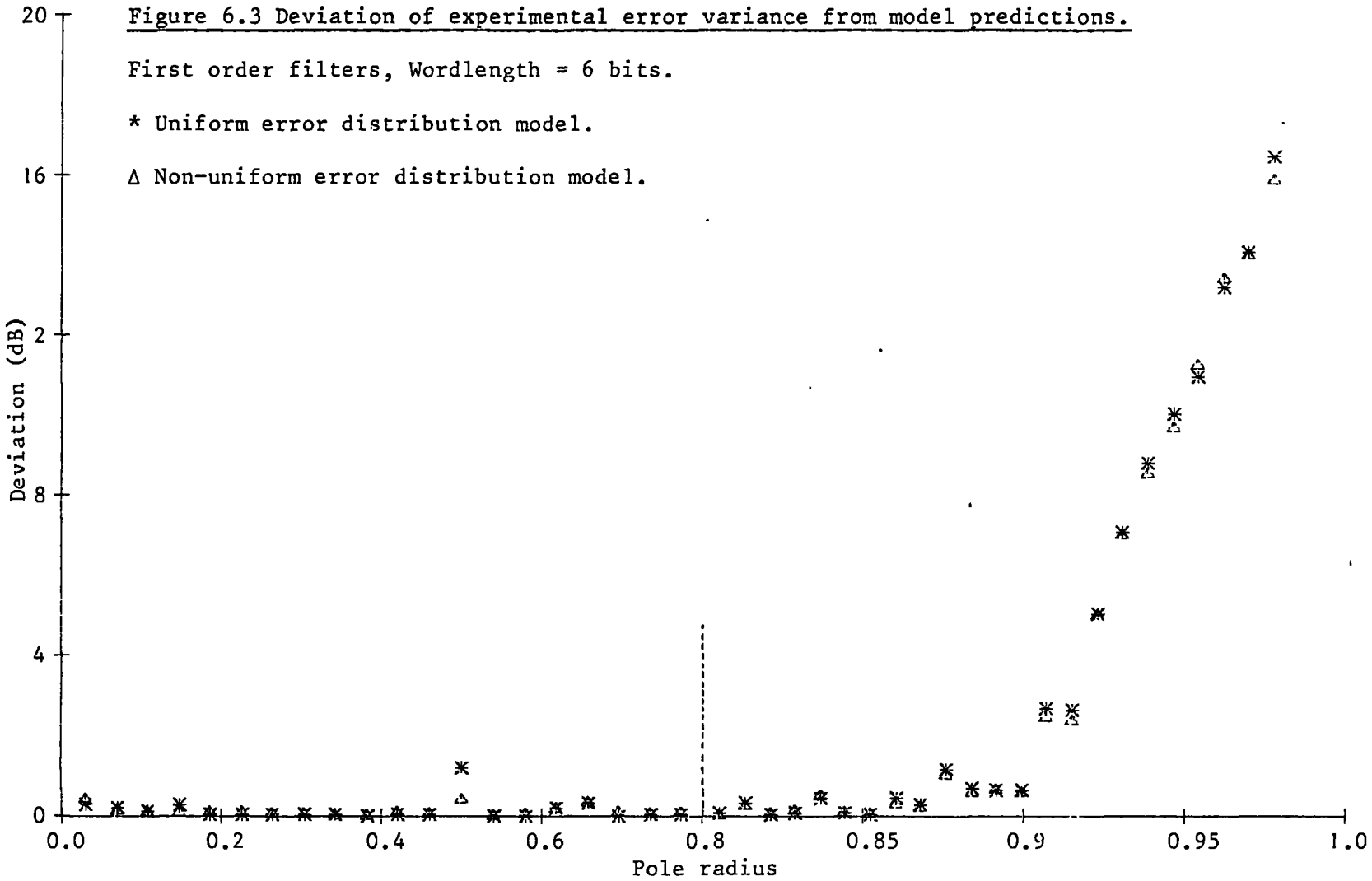


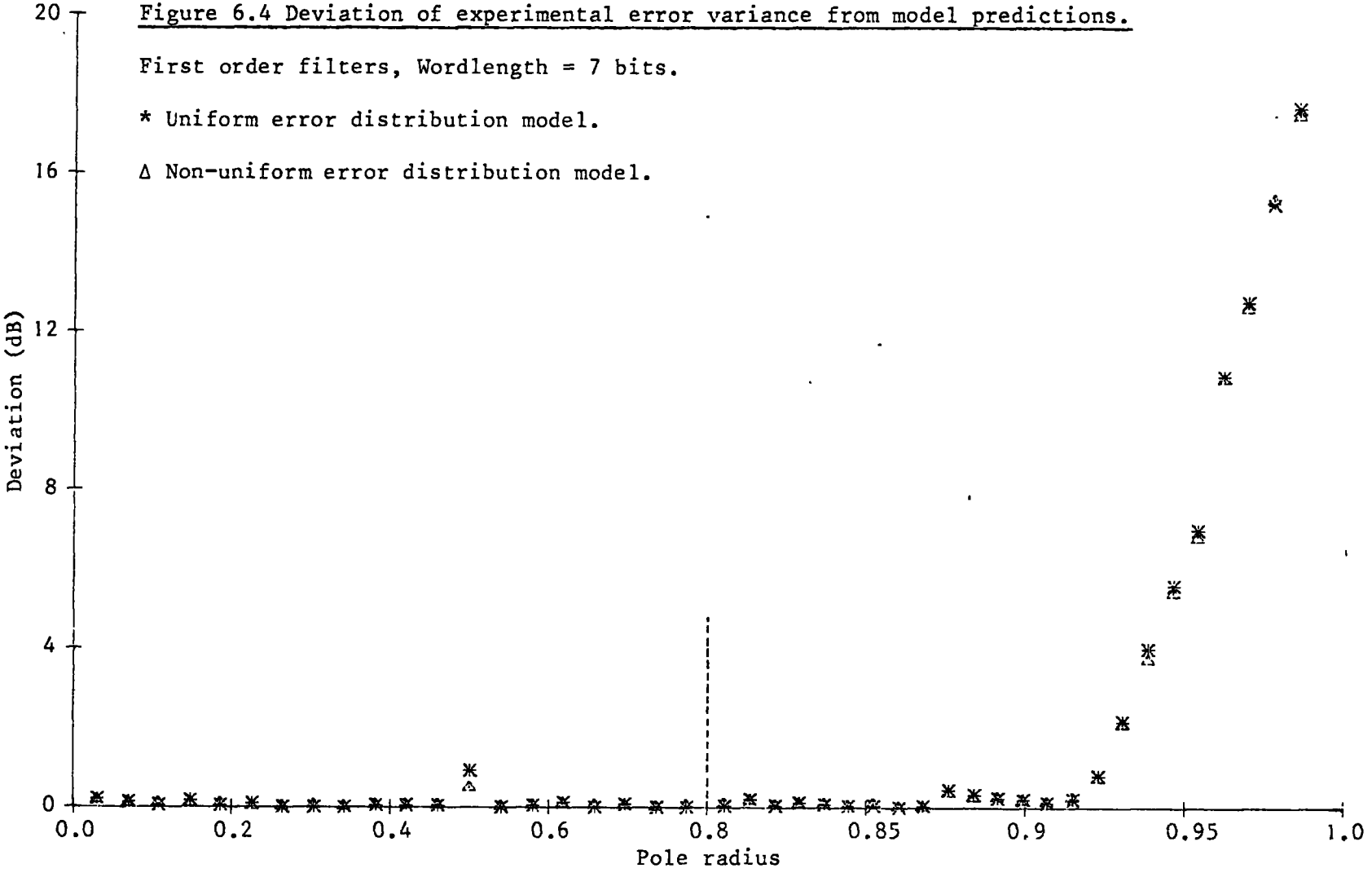
Figure 6.3 Deviation of experimental error variance from model predictions.
First order filters, Wordlength = 6 bits.
* Uniform error distribution model.
Δ Non-uniform error distribution model.
Deviation (dB)
20
16
2
8
4
0
0.0
0.2
0.4
0.6
0.8
0.85
0.9
0.95
1.0
Pole radius

Figure 6.4 Deviation of experimental error variance from model predictions.

First order filters, Wordlength = 7 bits.

* Uniform error distribution model.

Δ Non-uniform error distribution model.



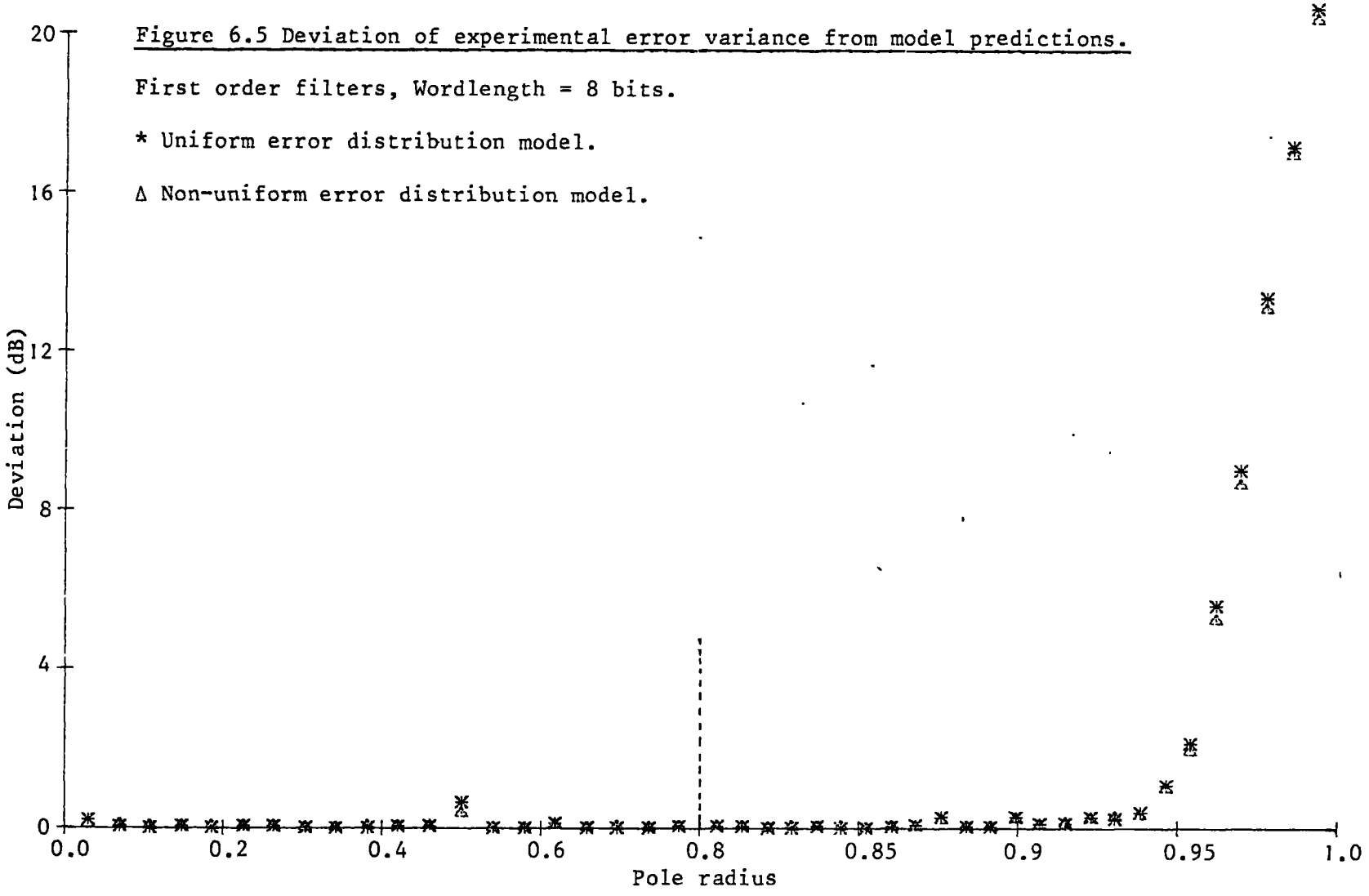


Figure 6.5 Deviation of experimental error variance from model predictions.
First order filters, Wordlength = 8 bits.
* Uniform error distribution model.
Δ Non-uniform error distribution model.
Deviation (dB)
20
16
12
8
4
0
0.0
0.2
0.4
0.6
0.8
0.85
0.9
0.95
1.0
Pole radius

signal wordlengths show more difference between the two models, but neither is consistently better than the other. Moreover, the experimental results are seen to deviate by up to 20dB from the model predictions. The results shown in Figure 6.5 for the 8-bit signal wordlength filters most clearly demonstrate the manner in which the deviation varies with the filter pole radius. For pole radii below 0.95 there is good agreement between the predictions of either model and the experimental results. However, above this pole radius the deviation rises sharply to about 20dB. With successive decreases of wordlength the pole radius at which the deviation begins to rise significantly above 0dB becomes diminished, and for a given pole radius the deviation decreases with increasing signal wordlength. Filters with a pole radius close to unity require a coefficient a_0 correspondingly near to zero, but as the signal wordlength, and therefore the filter input amplitude, is decreased, there is a growing tendency for the output sequence from the multiplier a_0 to consist only of zeros, rendering the results meaningless. Hence the maximum pole radius for which results are presented falls from 0.99 for an 8-bit signal wordlength to 0.92 in the 4-bit case.

It is unfortunate that the large deviations from theory should occur for filters with a pole very close to the unit circle in the z-plane as these are often the most useful in practice because of the high degree of rejection of unwanted frequencies which they yield. It is important, therefore, to seek to understand what is causing the observed deviation from the theoretically predicted performance. As a first stage in this investigation, the practical variances of the equivalent input error sequences were checked against the values predicted by the modified noise model. The agreement here was extremely good for all wordlengths and pole radii, indicating, firstly, the statistical independence of the roundoff error samples created at multipliers a_0 and b_1 which are summed in the theoretical model to form the equivalent input error sequence,

and secondly, that the simultaneous equations accurately yield the amplitude distribution of the filter output sequence. Consequently such input error sequences must experience, on filtering, a variance gain which deviates from that predicted by the theoretical models.

6.3 Correlation effects.

The autocorrelation function of a sequence $\{e_n\}$ may be defined as

$$R_{ee}(i) = E(e_n . e_{n-i}) / \sigma_e^2 \qquad (6.3.1)$$

where E( ) indicates the mean, and σ_e^2 is the variance of $\{e_n\}$. If the sequence $\{e_n\}$ is truly random then its autocorrelation function is unity for i=0 and zero for all other i. If, however, it is not truly random then the autocorrelation function takes on finite values when the lag i is not zero. The equation relating the variance of the equivalent input error sequence σ_ξ^2 to the output error variance σ_η^2 is, for a first order direct filter, usually written

$$\sigma_\eta^2 = \sigma_\xi^2 . \frac{1}{2\pi j} \cdot \oint_{|z|=1} \frac{dz}{B(z).B(z^{-1}).z} \qquad (6.3.2)$$

where

$$B(z^{-1}) = (1 + b_1 z^{-1}) \qquad (6.3.3)$$

This can be evaluated to give

$$\sigma_\eta^2 = \sigma_\xi^2 \left(\frac{1}{1 - b_1^2} \right) \qquad (6.3.4)$$

However equation (6.3.2) is a simplification of a more general equation

$$\sigma_\eta^2 = \frac{1}{2\pi j} \cdot \oint_{|z|=1} \frac{S_\xi(z).dz}{B(z).B(z^{-1}).z} \qquad (6.3.5)$$

where $S_\xi(z)$ is termed the power spectral density of the equivalent input error sequence. It is related to the autocorrelation function of the sequence by the equation

$$S_\xi(z) = \sigma_\xi^2 \sum_{i=-\infty}^{\infty} R_{ee}(i).z^{-i} \qquad (6.3.6)$$

that is, by a transformation from the time-domain to the z-domain. If the autocorrelation function is zero for $i \neq 0$, then the power spectral density is simply a constant equal to the variance σ_ξ^2. Hence in such a case of a truly random equivalent input error sequence equation (6.3.5) simplifies to equation (6.3.2). However, when a non-random sequence is being treated equation (6.3.5) must be used.

The results recorded in the previous section indicate that in some circumstances equivalent input error sequences do not experience a variance gain of $1/(1-b_1^2)$ on filtering. This demonstrates that in such cases the use of the simplified equation (6.3.2) is not justified, which in turn means that the error sequences do not have autocorrelation functions which go to zero for all finite lags i. That is, they are not truly random. The remainder of this chapter is devoted to an investigation of how a non-random input error sequence can arise and what may be done to take account of such an effect in a predictive model.

6.3.1 Signal to error correlation at a single multiplier.

If the input signal to a filter is taken to be a random sequence with variance σ_x^2 then the ideal autocorrelation function of the filter output sequence is given by

$$R_{yy}(i) = \left(\frac{\sigma_x^2}{\sigma_y^2}\right) \cdot \frac{1}{2\pi j} \cdot \oint_{|z|=1} H(z).H(z^{-1}).z^{i-1}.dz \qquad (6.3.7)$$

where the variance of the output sequence σ_y^2 is given by

$$\sigma_y^2 = \sigma_x^2 . \frac{1}{2\pi j} \cdot \oint_{|z|=1} H(z).H(z^{-1}).z^{-1}.dz \qquad (6.3.8)$$

The important point of equation (6.3.7) is that although the filter input sequence is random, the output sequence is not. Hence if there were any mechanism by which the roundoff error at the coefficient b_1 could be correlated to the filter output sequence, this error sequence would also be non-random. This would result in the non-randomness of the equivalent input error sequence and hence explain the experimental results. In the absence of any other obvious explanation it is convenient to proceed on the assumption that such correlation exists and to examine what can be done about it.

6.3.1.1 Achieving zero correlation.

At first sight the only way of taking such signal to error correlation into account in a predictive model might seem to be by incorporating the evaluation of equations (6.3.5) and (6.3.6). In addition to the mathematical complexity of such a calculation, it is not at all clear how the autocorrelation function of the equivalent input error sequence could be predicted. In the light of these difficulties it seems wise to seek an alternative solution to the problem.

Consider the sample x_i of an input sequence $\{x_n\}$ to the multiplier α; the ideal output is y_i but when this is rounded y_i' is produced and a roundoff error e_i is created. That is,

$$y_i' = \alpha.x_i + e_i \qquad (6.3.9)$$

The correlation coefficient r between $\{x_n\}$ and $\{e_n\}$ is defined as

$$r = \frac{E(x_i.e_i)}{\sigma_x.\sigma_e} \qquad (6.3.10)$$

where the mean is taken over the entire length of the sequences. Making the definition

$$r' = \sum_i (x_i . e_i) \qquad (6.3.11)$$

it can be seen that zero correlation is only achieved when r' is zero. Using equation (6.3.9) it is possible to rewrite equation (6.3.11)

$$r' = \sum_i x_i (y_i' - \alpha . x_i) \qquad (6.3.12)$$

Regarding r' as a function of α alone, and differentiating with respect to α, yields

$$\frac{dr'}{d\alpha} = - \sum_i x_i^2 \qquad (6.3.13)$$

By postulating a change in the theoretical coefficient value the correlation coefficient may be reduced to zero. The coefficient value must be modified to $\alpha+\delta\alpha$, where

$$\delta\alpha = \frac{-r'}{\left(\frac{dr'}{d\alpha}\right)} = \frac{\sum_i (x_i . e_i)}{\sum_i (x_i^2)} \qquad (6.3.14)$$

Hence, for a given practical multiplier α, calculation of $\delta\alpha$ requires only the knowledge of the amplitude distribution of the input sequence $\{x_n\}$. Such information can be made available in a predictive model for a first order filter by the solution of simultaneous equations as described previously. Therefore by using equation (6.3.14) it is possible to predict the modified theoretical values of the coefficients a_0 and b_1 required to give zero signal to error correlation at the two multipliers.

It must be understood that no change in the practical filter coefficients or actual signal sequences is involved in the proposed

modification for zero correlation. By altering the theoretical coefficients the ideal filter response is redefined. Because the practical signal sequences remain unaltered the roundoff error sequences become modified; it is this modification which allows zero correlation to be achieved. The redefined ideal filter response is dictated by the amplitude distribution of the filter input sequence; the ordering of the sequence is insignificant.

Once the required theoretical coefficient modifications have been predicted it is a simple matter to calculate the variances of the redefined roundoff error sequences at a_0 and b_1, sum them to give the equivalent input error variance, and finally to compute the output error variance using the simplified equation (6.3.4) employing the modified theoretical value of b_1.

6.3.1.2 Sign-magnitude truncation.

The sign-magnitude truncation roundoff process was described in Chapter 3, and the fact that it causes a significant degree of signal to error correlation was noted. It has not been used in any of the experimental work so far, because the simple noise model is clearly not applicable. Liu and Van Valkenburg[52] have presented a noise model which takes the particular form of signal to error correlation into account, but the mathematical difficulty in so doing is very great. It would appear that the accurate prediction of noise by their model not only requires the knowledge of sequence amplitude distributions but also of their precise ordering; it is not clear how this information may be obtained. It is possible to simplify the mathematics slightly and calculate a bound on the error variance. This calculation has the advantage of not requiring data on sequence orders. However, Liu and Van Valkenburg admit that this bound is rather too loose to be of any great predictive value.

It should be clear that the method described in the previous section for achieving zero signal to error correlation is also applicable to filters employing sign-magnitude truncation. Not only does this approach lead to a much simpler prediction and description of the practical behaviour of a filter, it can also be argued that the description is more precise because the error sequence at the filter output is statistically independent of the output signal, whereas this is not true when signal to error correlation is present at the multipliers.

6.3.1.3 Testing the zero correlation model.

The desirability of applying the zero correlation modification to the filters tested earlier in this chapter is obvious. It is clearly necessary to see whether the predictions of a noise model incorporating the postulated alterations to the theoretical filter coefficients do indeed come into agreement with the measured output error variances. Moreover, at this stage the existence of signal to error correlation in filters employing rounding has not been directly proved, but merely assumed as the only obvious explanation for the results reported previously. If, however, it can be shown that the theoretical filter coefficients do require modification according to equation (6.3.14), the original existence of signal to error correlation will be proved. In order to demonstrate the applicability of the technique to sign-magnitude truncation, filters using this roundoff method are tested in addition to those employing rounding.

The application of the zero correlation modifications makes the practical measurement of error variance a little more complicated than usual. Basically each 1024 sample input sequence has to be processed twice. During the first time the roundoff errors at each multiplier are calculated relative to the ideal filter response defined by the

unmodified coefficients; the terms required by equation (6.3.14) for calculating the required coefficient modifications are also determined. These coefficient modifications are then evaluated and the ideal filter response thus redefined. The second time, the variance of the output error sequence with respect to the redefined ideal response is computed. The mean of 100 variance results is recorded as usual along with the standard error. In addition, the means of the 100 values of the modified theoretical coefficients a_0 and b_1 are recorded with their standard errors. This permits a comparison with the modified theoretical coefficient values predicted by the model and also gives some idea of the variation of the degree of signal to error correlation from run to run. ZCDIR1 is the subroutine, called by the main program EXEC, which both performs the filtering and measures the above mentioned parameters. ZCMOD is the main program which yields the model predictions. This algorithm is virtually identical to AMPMOD, described in the first part of this chapter, but it has the additional capability of calculating and using the modified theoretical coefficients.

6.3.1.4 Presentation of results for first order filters.

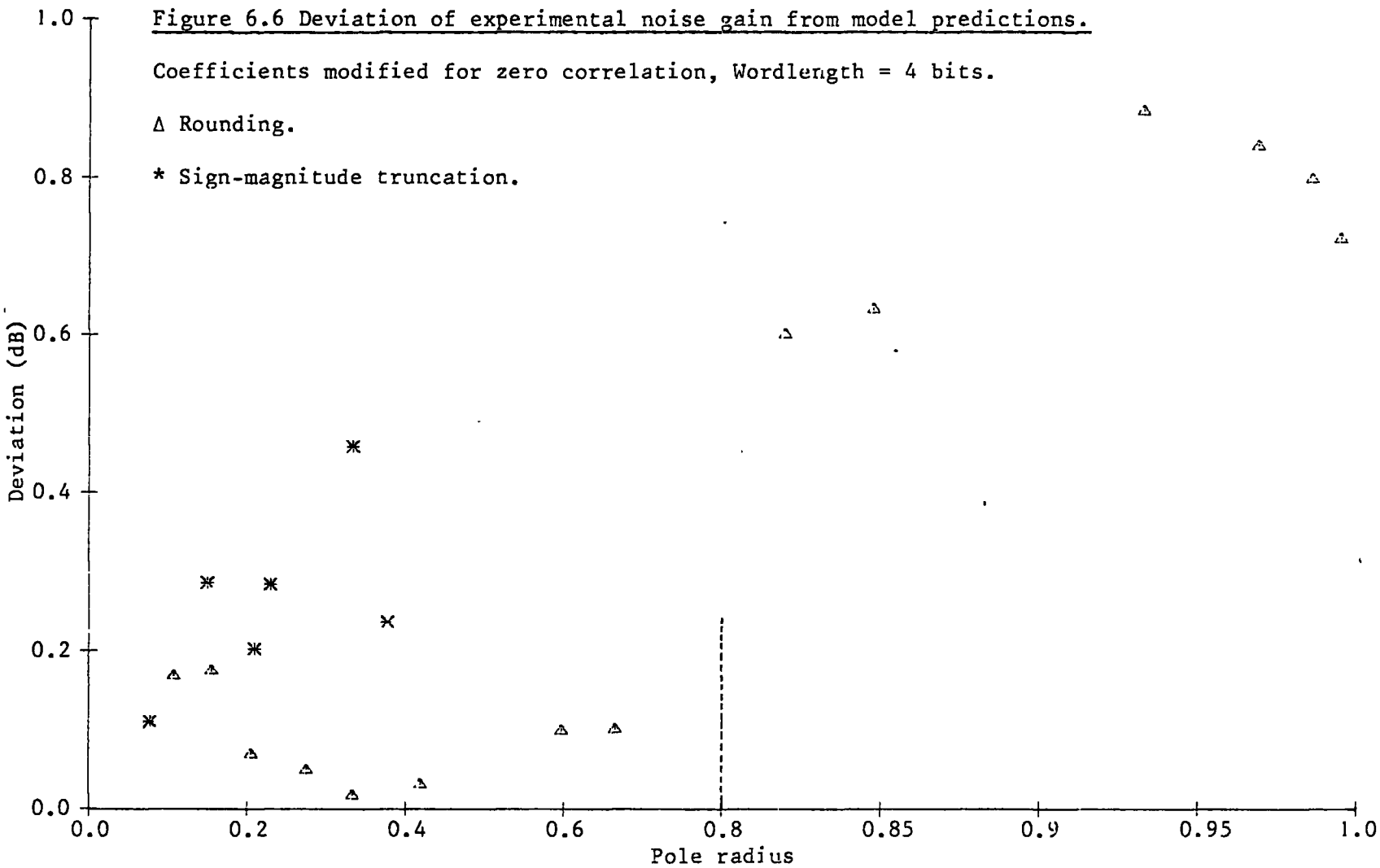
The results of this experiment are shown in Figures 6.6 to 6.10; again each graph refers to one of the signal wordlengths under consideration. Instead of plotting the deviation of the measured output error variance from the predicted value, as before, the deviation of the observed noise variance gain from that theoretically anticipated is recorded. This has the advantage of taking account of any inaccuracy in the prediction of the variance of the equivalent input error sequence. In fact, the observed values are in very close agreement with those predicted. The pole radius attributed to a particular practical filter is that defined by the theoretical value of b_1 as modified for zero signal to error correlation. There is no problem in doing this as

Figure 6.6 Deviation of experimental noise gain from model predictions.

Coefficients modified for zero correlation, Wordlength = 4 bits.

Δ Rounding.

* Sign-magnitude truncation.



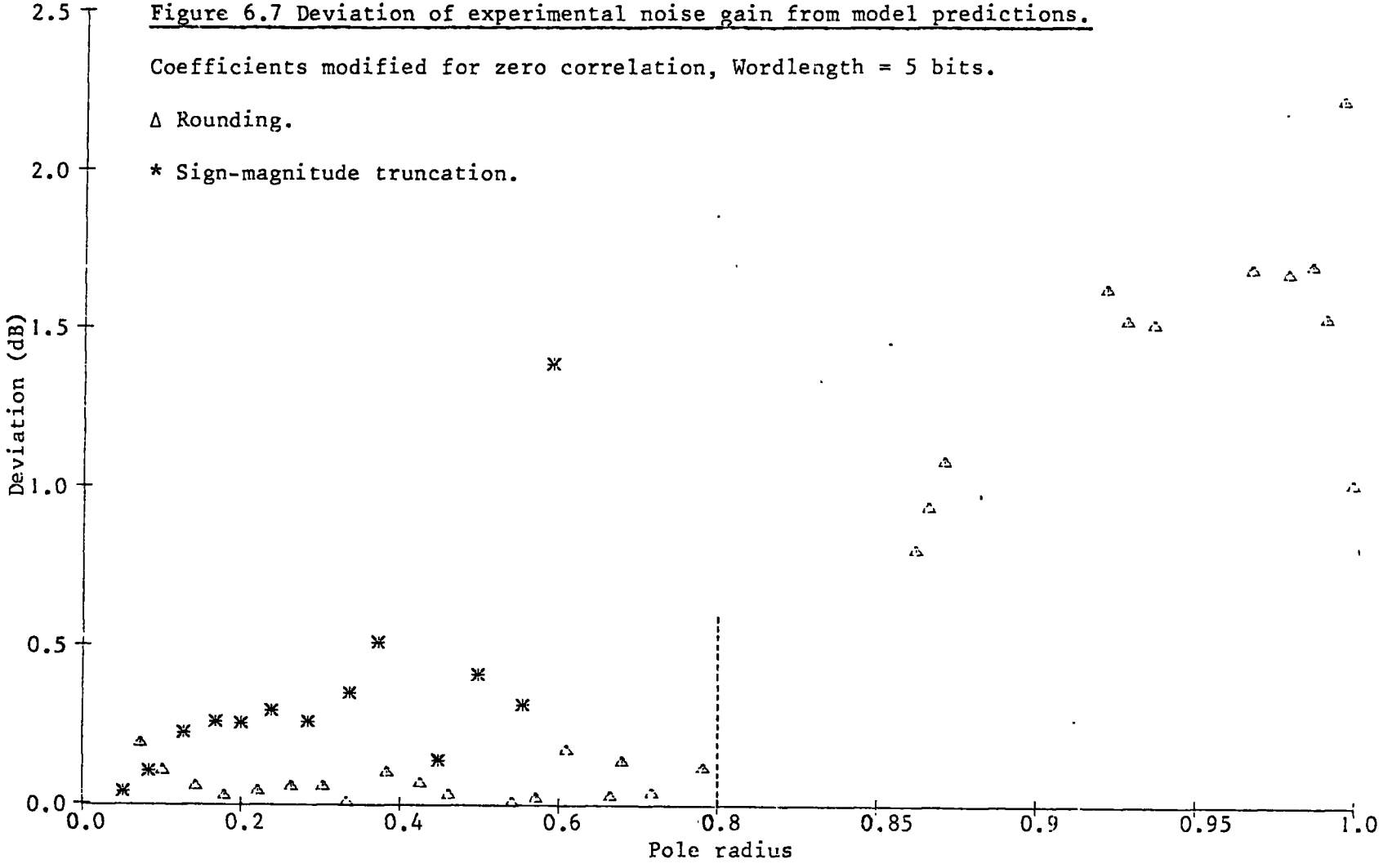


Figure 6.7 Deviation of experimental noise gain from model predictions.

Coefficients modified for zero correlation, Wordlength = 5 bits.

Δ Rounding.

* Sign-magnitude truncation.

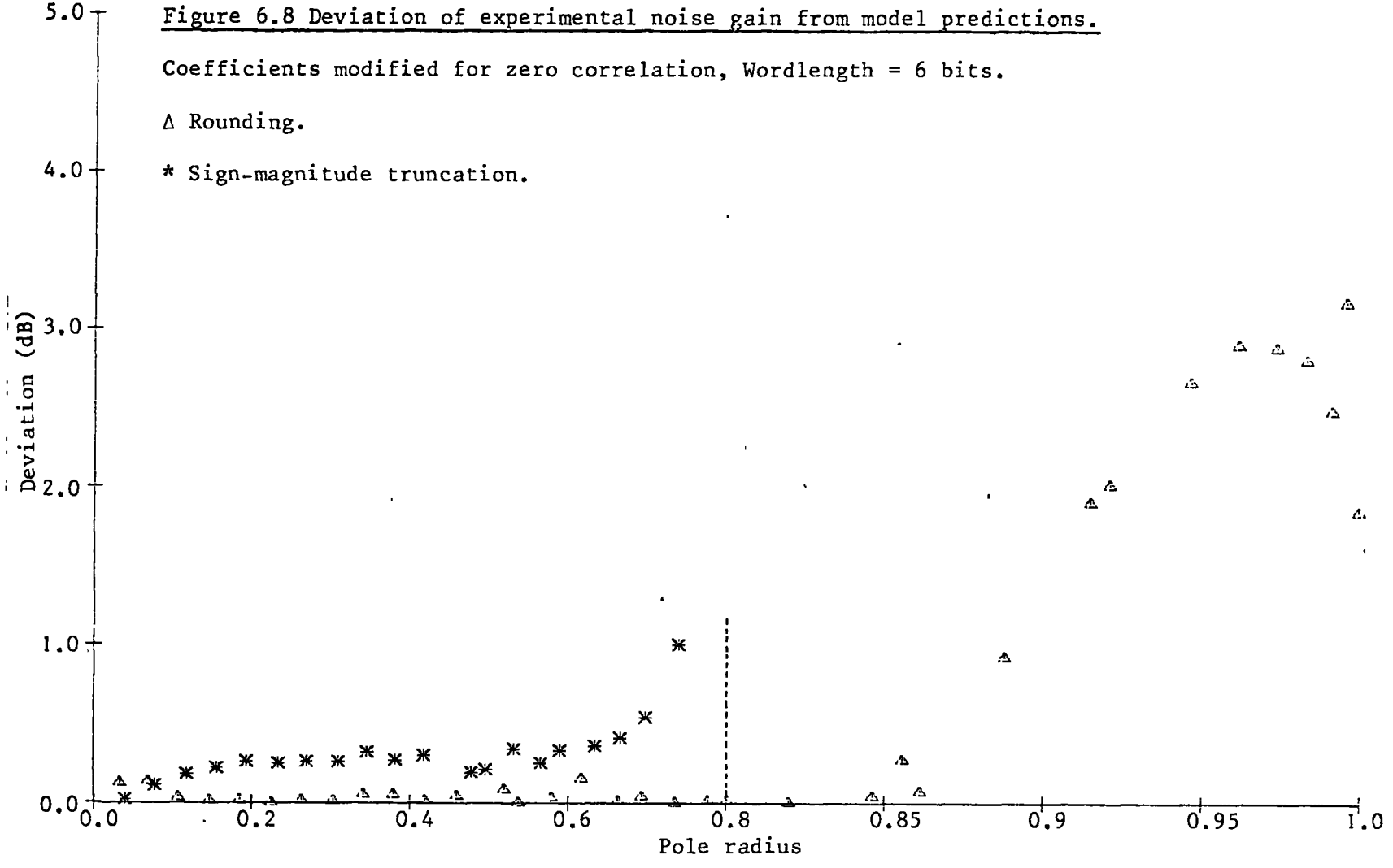


Figure 6.8 Deviation of experimental noise gain from model predictions.

Coefficients modified for zero correlation, Wordlength = 6 bits.

Δ Rounding.

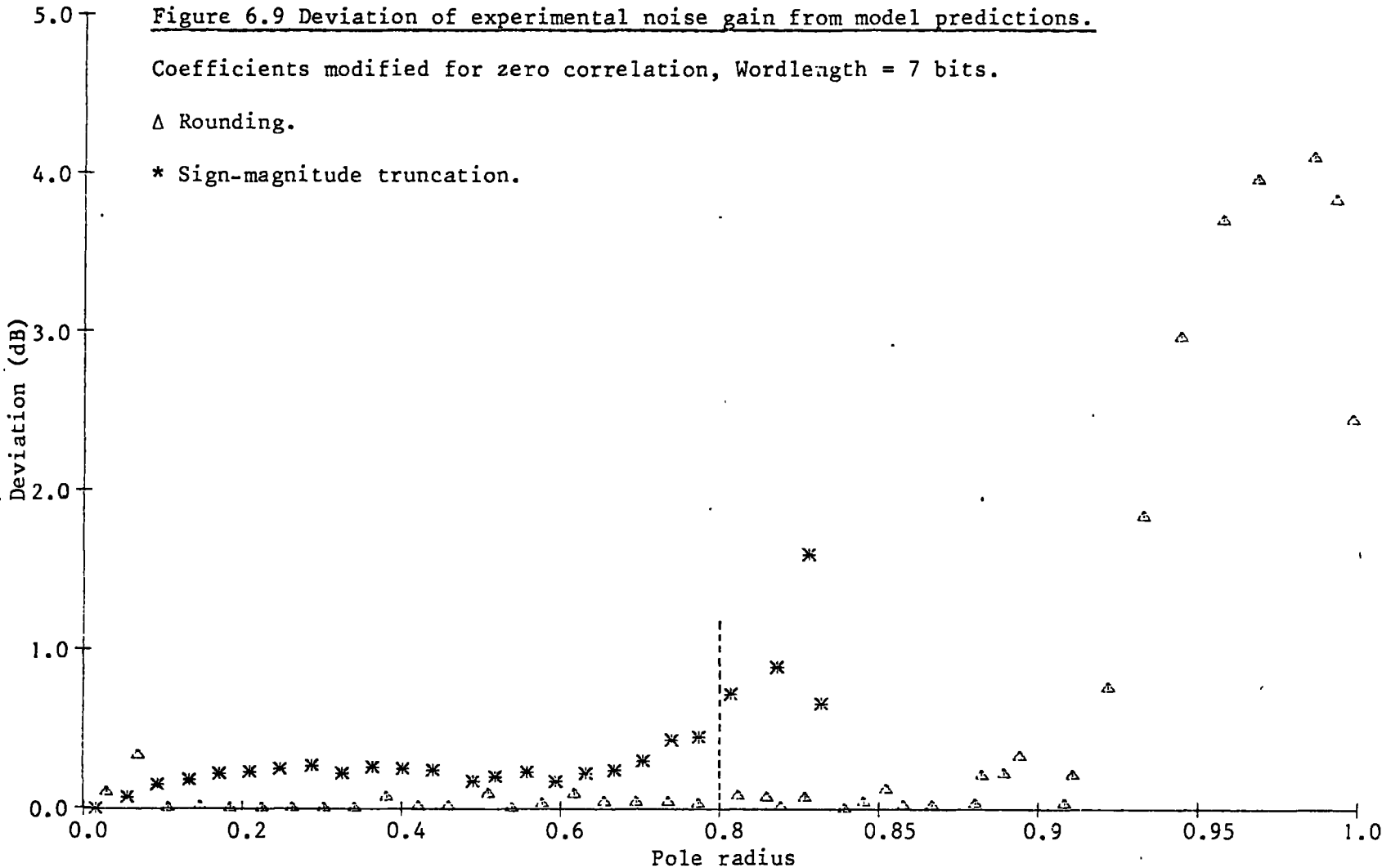
* Sign-magnitude truncation.

Figure 6.9 Deviation of experimental noise gain from model predictions.

Coefficients modified for zero correlation, Wordlength = 7 bits.

Δ Rounding.

* Sign-magnitude truncation.



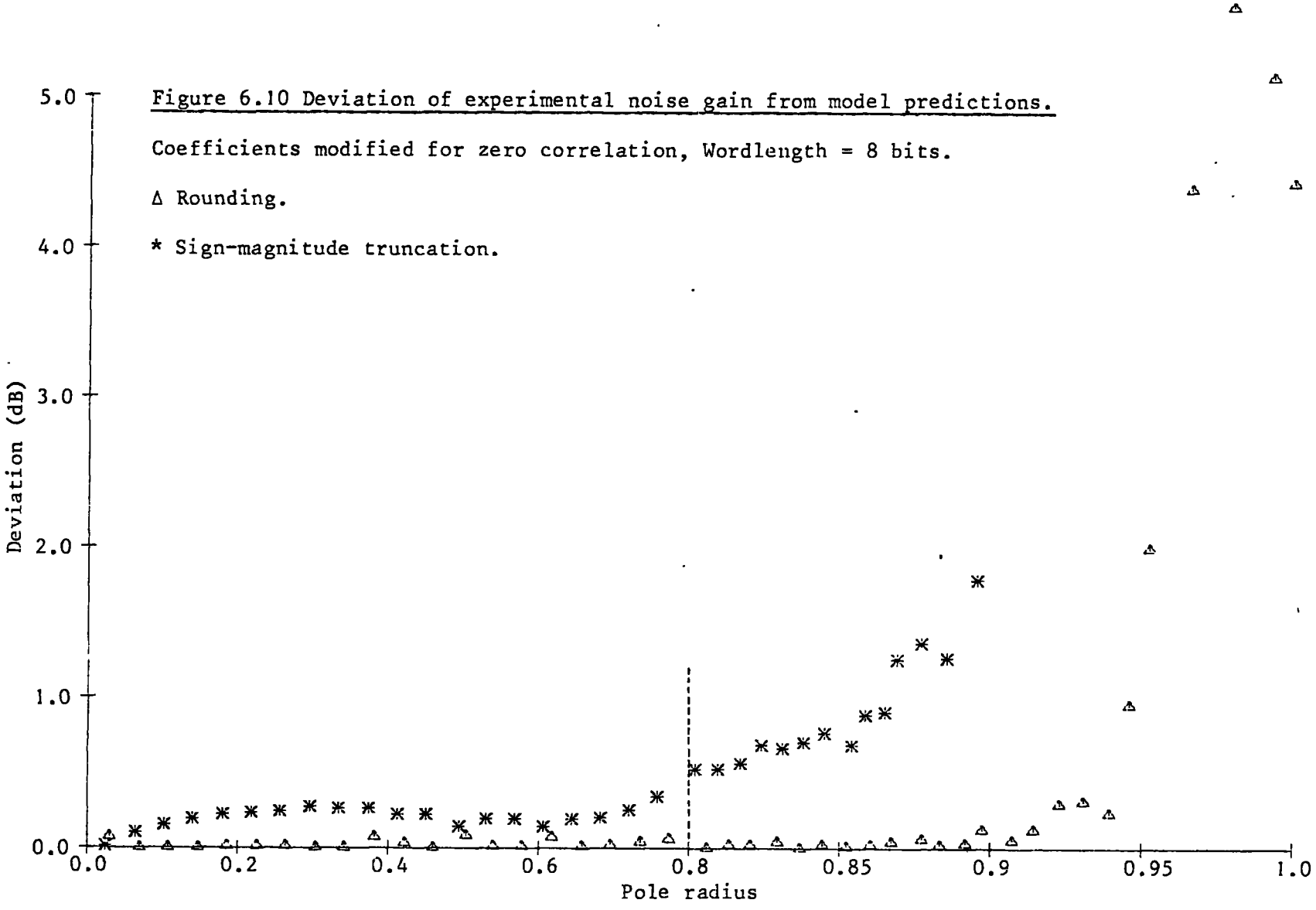


Figure 6.10 Deviation of experimental noise gain from model predictions.

Coefficients modified for zero correlation, Wordlength = 8 bits.

Δ Rounding.

* Sign-magnitude truncation.

extremely good agreement is found between the values predicted by the model for the modified coefficients and those obtained in practice. The variation in these experimentally determined values from run to run, as indicated by the standard errors in the means, is found to be insignificantly small. The symbols on these graphs do not have the same meaning as in Figures 6.1 to 6.5. This time the triangles refer to filter realisations employing rounding, while the asterisks indicate the use of sign-magnitude truncation. As usual error bars would be very small and are omitted for clarity.

Although it is not made absolutely clear in the presentation of the results, careful comparison of these graphs with those for the previous experiment shows that the coefficient values b_1 are indeed modified to achieve zero signal to error correlation; in fact, this is also true for the a_0 coefficients. As the signal wordlength is decreased so the theoretical coefficient modifications become greater. This proves that signal to error correlation is normally present in these filters and that such correlation grows stronger with reduced wordlength. An absence of results is to be noted for filters with a high pole radius which use sign-magnitude truncation; this increasingly becomes the case as the wordlength is shortened. This is partly caused, as mentioned before, by the tendency for small values of a_0 to become effectively zero in practice, and partly by the fact that for sign-magnitude truncation the application of the zero correlation modification must always result in theoretical coefficients of reduced magnitude.

Figures 6.6 to 6.10 show that the application of the theoretical coefficient modification to achieve zero signal to error correlation permits a significant improvement in the accuracy of noise predictions. For example, the deviation recorded for an 8-bit wordlength filter employing rounding and with a pole radius of 0.98 is reduced from 18dB to 5dB. However, the equivalent input error sequences still do not experience

the variance gain expected of pure noise, as defined by equation (6.3.2).

6.3.1.5 Discussion of results.

By the same argument as was used before, in a case where an equivalent input error sequence does not experience a variance gain as predicted by equation (6.3.2), it must be concluded that the sequence has an autocorrelation function which does not go to zero for all finite values of the lag i.

Once again it is suspected that the problem may have its source at the multiplication of the delayed filter output y_{n-1} by the coefficient b_1. For ease of notation, let the sequence $\{y_{n-1}\}$ be renamed $\{v_n\}$. The autocorrelation function of $\{v_n\}$ is identical to that of the filter output $\{y_n\}$ which, as previously shown, has finite terms for non-zero lags i. If $\{e_n\}$ is the roundoff error sequence created at b_1 then, when the theoretical modification to b_1 is applied, there is zero correlation between $\{e_n\}$ and $\{v_n\}$. However, it is quite possible for $\{e_n\}$ to be correlated with $\{v_{n-1}\}$ or $\{v_{n-2}\}$ and so on. For example, $\{v_{n-1}\}$ may be analysed into two component sequences $\{u_{n-1}\}$ and $\{w_{n-1}\}$. $\{u_{n-1}\}$ is taken to be perfectly correlated with $\{v_n\}$, while $\{w_{n-1}\}$ is completely independent of $\{v_n\}$. $\{e_n\}$ could be perfectly correlated with $\{w_{n-1}\}$, and so be partially correlated with $\{v_{n-1}\}$ and still have zero correlation with $\{v_n\}$. The sequence $\{w_{n-1}\}$ may reasonably be supposed to have an autocorrelation function which does not go to zero for all non-zero lags. The cross-correlation function between $\{e_n\}$ and $\{w_{n-1}\}$, $E(e_{n-i} \cdot w_{n-1})$, therefore has finite values even when i is not equal to zero. In consequence the roundoff error sequence $\{e_n\}$ has an autocorrelation function which is not zero at all finite lags, that is, $\{e_n\}$ is not a random sequence. This in turn has the effect of making the equivalent input error sequence non-random, and the use of equation (6.3.2) strictly invalid.

The above paragraph demonstrates the theoretical possibility of obtaining a non-random roundoff error sequence even when the main signal to error correlation has been reduced to zero. Having used the coefficient modification technique to reduce the correlation of $\{e_n\}$ with $\{v_n\}$ to zero, it is not possible to apply the technique a second time to remove subsidiary correlations such as perhaps $\{e_n\}$ with $\{v_{n-1}\}$. This possibly indicates that the reduction of $E(e_n.v_n)$ to zero represents the most that can be done to bring the predicted behaviour of a filter into agreement with that obtained in practice. On the other hand, a method may be found of modifying the theoretical filter coefficients according to some different relationship from equation (6.3.14), in such a way that the randomness of $\{e_n\}$ is optimised. As stated, this is all speculation and there is obviously great scope for more work to be done before the roundoff processes can be more fully understood and better theoretical predictive models produced.

6.3.2 Correlation between error sequences in higher order filters.

A very recent paper by Parker and Girard[77] demonstrates that correlations can arise between the roundoff error sequences at various points in the structure of a filter.

Consider the second order canonic filter structure as depicted in Figure 3.9. Parker and Girard show that when the same sequence is multiplied by two coefficients, the roundoff error sequences at the two multipliers may be partially correlated. So in Figure 3.9 $\{\varepsilon_{2,n}\}$ and $\{\varepsilon_{4,n}\}$ may be correlated, as may $\{\varepsilon_{3,n}\}$ and $\{\varepsilon_{5,n}\}$. Furthermore $\{\varepsilon_{2,n}\}$ and $\{\varepsilon_{4,n}\}$ may both be correlated with $\{\varepsilon_{1,n-1}\}$, and by the same argument the following pairs of roundoff error sequences may also have non-zero correlation coefficients: $\{\varepsilon_{3,n}\}$ and $\{\varepsilon_{1,n-2}\}$, $\{\varepsilon_{5,n}\}$ and $\{\varepsilon_{1,n-2}\}$, $\{\varepsilon_{3,n}\}$ and $\{\varepsilon_{2,n-1}\}$, $\{\varepsilon_{3,n}\}$ and $\{\varepsilon_{4,n-1}\}$, $\{\varepsilon_{5,n}\}$ and $\{\varepsilon_{2,n-1}\}$, and finally $\{\varepsilon_{5,n}\}$ and $\{\varepsilon_{4,n-1}\}$. Note that these are not the signal to error

correlations at a single multiplier treated previously in this chapter. It is also important to realise that such correlations cannot occur in a first order filter as the two multipliers a_0 and b_1 can never process the same signal sequence.

Parker and Girard show that the error functions for two given multipliers coincide periodically along the input level axis at a zero error level. The greater the period of coincidence the less will be the correlation between the roundoff error sequences at the two multipliers. One of the consequences of this is that correlation is likely to be most significant when short coefficient wordlengths are in use. (This may explain the statement made in a recent paper by Hadjifotiou and Appleby[45] that for a short coefficient wordlength roundoff errors tend to become deterministic). Parker and Girard consider an example which uses a 2-bit and a 3-bit coefficient and they show that in such a case significant correlation is obtained even when the input signal to the multipliers is supposedly unquantised. Moreover, if a quantised input sequence is used then the degree of correlation is found to increase. On the basis of these findings it seems reasonable to suggest that significant levels of correlation will also be obtained in a situation where the coefficient wordlength is relatively long while that of the signal is short. This suggests, incidentally, that significant correlations existed between the various error sequences in the second order direct and canonic filters tested in Chapter 5.

Parker and Girard indicate how the output error variance may be predicted when correlations between roundoff error sequences are considered. Signal to error correlations will also be present but these are not taken into account. The mathematics involved in the noise prediction is extremely complicated. The final expression for the noise variance includes, as might be expected, correlation coefficients for all the correlations between error sequences present. The evaluation then

continues by assuming integral values for some of the filter coefficients. No roundoff occurs at such multipliers, and the expression for the noise variance thus becomes simplified so that only one of the correlation coefficients is still required. It is still not possible, however, to attribute a precise value to this correlation coefficient, but only to state bounds within which it must lie. Although the result of this is a prediction that the noise variance should be somewhere in a fairly narrow band of values, it does not appear that the method could be applied with great ease or certainty to a more realistic example.

It is possible to start thinking of trying to solve this problem by reducing the correlations to zero by making appropriate modifications to the theoretical filter coefficient values in a similar way to that used before. More careful consideration, however, indicates several difficulties. First, while it may be possible to predict correlations of the type $\{\varepsilon_{2,n}\}$ with $\{\varepsilon_{4,n}\}$, those of the type $\{\varepsilon_{3,n}\}$ with $\{\varepsilon_{4,n-1}\}$ require knowledge of the ordering of the filter signal sequences. Second, although an equation can be formulated to calculate the required modification to a given theoretical filter coefficient, this modification is not the same as that indicated by equation (6.3.14) for zero signal to error correlation; it is, therefore, impossible to reduce both forms of correlation simultaneously to zero. The last problem is that already discussed of predicting signal amplitude distributions in higher than first order filters. It would seem, therefore, that the accurate prediction of error variance in practical discrete multiplier filters of higher than first order is likely to remain a difficult problem for some time.

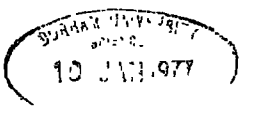
6.4. Summary.

The experiment reported in the first part of the chapter indicates that, while it is possible, by the solution of simultaneous equations, to predict roundoff error distributions very accurately for first order

filters, the incorporation of this information into a noise model yields no improvement on the simple model which assumes uniformly distributed errors. Furthermore, it is shown that the observed discrepancies between the measured error variances and the theoretically predicted values must be explained in terms of the non-randomness of the equivalent input error sequence.

This non-randomness can be caused by correlation between the signal and error sequences at a filter multiplier. A method is presented which allows zero signal to error correlation to be achieved at a multiplier by a redefinition of the theoretical value of the coefficient. The prediction of such theoretical coefficient modifications requires the precise assessment of signal and error amplitude distributions as provided by the solution of simultaneous equations. The experimental investigation reported in the second part of the chapter demonstrates that the employment of the above technique yields a significant improvement in the agreement between the theoretical and practical results. It also shows that the prediction of error variance in filters employing sign-magnitude truncation is facilitated by this technique. Reduction of the main signal to error correlation to zero, however, still does not yield true agreement between experiment and theory. The possibility of this being caused by subsidiary signal to error correlations is discussed, but no solution to the problem is presented.

Brief consideration has been given to the possible correlation between the roundoff error sequences at various points in a filter structure demonstrated very recently by Parker and Girard[77]. This kind of correlation is likely to exist in any discrete multiplier filter of higher than first order; this is in addition to the signal to error correlations already mentioned. Unfortunately a method of modelling such behaviour, without the need for highly complicated mathematics and the estimation of correlation coefficients, remains to be found.



CHAPTER 7 - CONCLUSION.

The first section of this final chapter summarises the detailed theoretical conclusions which have been presented in previous chapters alongside the experimental results from which they are inferred.

The first experimental results are presented in Chapter 4. The experiment reported uses a fast Fourier transform to analyse the frequency spectra of roundoff error sequences created when a quantised sinusoid is processed by a single multiplier. Each error sequence is found not to possess a flat frequency spectrum, but to consist of odd harmonics of the input signal frequency. It is shown that harmonic content can arise whenever the input sequence contains a periodic component. The presence of the first harmonic in the error sequence indicates a degree of correlation between the signal and error. Hence, under these experimental conditions, the assumption that roundoff error sequences can be treated as white noise, and the assumption that error sequences are uncorrelated with any other sequence are both seen to be invalid.

Chapter 5 presents the results of measuring the output error variance for several first and second order filter realisations when tested with random input sequences. The main conclusion to be drawn from these results is that there is generally poor agreement between the experimental measurements and the values predicted using the normal noise model[29]. The assumption that roundoff errors are uniformly distributed within the range allowed is considered and it is demonstrated that, for the signal wordlengths under examination, this assumption is not valid, and its use can cause very misleading predictions of error variance. Several of the filter realisations tested employed block-floating-point arithmetic[16]. These results show that only under rather rare experimental conditions can this arithmetic mode produce a

significant noise reduction; in general, its use is not recommended. One category of filters tested showed consistently close agreement between experiment and theory and also a marked noise reduction when compared with other realisations of the same filter response; this was the look-up table form[60] with the modification for reduced noise applied as suggested in Chapter 3.

Chapter 6 suggests an algorithm which can be used to predict actual roundoff error distributions in first order filters processing a random input sequence. The effect of incorporating this algorithm into a predictive noise model is tested experimentally for a range of first order responses. The results indicate that, in the great majority of cases considered, the addition of this feature to the noise model has little effect on the error variance predicted; so the discrepancies between experiment and theory remain. These can only be explained by the presence of signal to error correlations causing non-random error sequences. The algorithm for predicting error distributions can be further developed to predict the correlation coefficients and a method is proposed for their reduction to zero by a modification to the theoretical filter coefficient values. This suggestion is tested experimentally and is found to cause a significant improvement in the agreement between the theoretical and experimental results. Lagged signal to error correlations are presumed to account for the remaining level of discrepancy.

Of necessity this investigation has only given close consideration to a few selected signal processing methods. It is important, therefore, to attempt to use the above results and conclusions as a basis for recommending which techniques should be given wider application, and for suggesting the future investigation of some topics which might prove very fruitful.

The digital filter implementation proposed by Peled and Liu[60] was

aimed at increasing the processing speed by reducing the quantity of arithmetic performed during each filter cycle; this is achieved by an increase in the amount of data permanently stored in the filter to define its response. Such a reduction in the arithmetic required also gives the look-up table form a potential for low noise. This potential is only fully realised when filter coefficients are chosen so as to allow the data stored in the look-up table to be free from error. This is the essential feature of the so-called reduced-noise modification to this form. Correlations exist between error and signal sequences when a discrete multiplier filter is implemented using a short signal wordlength. For filters of higher than first order this problem is compounded by the presence of correlations between the various error sequences. Any such correlations effectively prevent a simple but accurate theoretical prediction of error variance. Consideration of the look-up table realisation in its reduced-noise form suggests that there is no possible mechanism whereby correlation can occur. This is borne out by the experimental results which show that this is the only filter form whose performance can be accurately predicted using an uncomplicated model. For this reason and for its low noise this filter form is to be highly recommended for further use.

The work carried out during this project has all been concerned with the realisation of digital filters by the implementation of recursive linear difference equations. For low order filters this is probably the best method, but it becomes ever more difficult to use as the filter order is increased. If a particular application allows the filter input sequence to be processed in blocks rather than sample by sample, the cyclic convolution property possessed by the discrete Fourier transform can be used to perform filtering. The property of cyclic convolution means that the convolution of two sequences can be found by transforming them, multiplying the two function transforms

together, and taking the inverse transform of the result. Hence the convoltuion process of filtering can be achieved by two discrete Fourier transforms, one on the input sequence block and one on the desired filter impulse response, and one inverse DFT on the product of the two transforms. Because of the computational efficiency of the FFT algorithm used for these three transformations, high order filtering can often be achieved more rapidly in this way than by the implementation of recursive linear difference equations. A useful investigation could be carried out into the effects of a short wordlength on filters realised using the fast Fourier transform. In the light of the present project it would be particularly interesting to study Peled and Liu's proposal for the implementation of the fast Fourier transform[61], using a similar design philosophy to their filter realisation and employing a short wordlength.

Any transform possessing the vital property of cyclic convolution must involve multiplying the N numbers of the sequence being transformed by integer powers of the Nth root of unity. In the conventional number system the Nth root of unity is a complex number, as are its integer powers. This explains the need for sequence samples to be multiplied by complex coefficients during the discrete Fourier transformation. There is, therefore, an inherent difficulty in the implementation of the FFT on a processor having only an unsophisticated arithmetic unit and a limited capability for data storage. For these reasons there is great current interest in a class of transformations called number theoretic transforms (NTT)[17] which possess the property of cyclic convolution and can be performed using the same algorithm structure as the FFT. Because arithmetic is performed modulo-M, where M is an integer, the sequence length N can be chosen so that the Nth root of unity and all its integer powers are integers. Use of these transforms only requires the storage of positive integers. The arithmetic operations needed are restricted to additions and word shifts so that even in a short wordlength processor

no roundoff errors are generated; the only error source is the quantisation of the input sequence prior to the transformation. These transforms have an obvious potential both in spectral analysis and filtering applications. An investigation of the implementation of these transforms using a short wordlength device such as a microprocessor should prove to be particularly fruitful.

These recent developments demonstrate the increasing interest in the implementation of digital signal processors using relatively unsophisticated systems or arrays of simple elements. It is hoped that, in a small way, this current project will augment the existing expertise and encourage further activity in this field. The birth and early development of digital signal processing were prompted by the existence of the large computer. Time may prove, however, that the continued growth in the importance of the subject will be fostered by a trend towards the use of comparatively simple and inexpensive processors.

APPENDIX 1 - MATHEMATICAL PROOF OF ALIASING FOR REAL SIGNALS.

The discrete Fourier transform is defined as

$$F(k.\Delta\omega) = \sum_{n=0}^{N-1} f(nT).e^{-jkn.\Delta\omega.T} \qquad (A1.1)$$

Let $W=e^{j2\pi/N}$ so that equation (A1.1) may be rewritten as

$$F(k) = \sum_{n=0}^{N-1} f(n).W^{-kn} \qquad (A1.2)$$

Therefore

$$F(N-k) = \sum_{n=0}^{N-1} f(n).W^{-n(N-k)} \qquad (A1.3)$$

However, $W^{-nN} = 1$ so therefore

$$F(N-k) = \sum_{n=0}^{N-1} f(n).W^{kn} \qquad (A1.4)$$

Comparing equations (A1.2) and (A1.4) it can be seen that if f(n) is a real function

$$F(N-k) = F(k)^{*} \qquad (A1.5)$$

If, as is normally the case, only the magnitude of the frequency-domain function is of interest then

$$|F(N-k)| = |F(k)| \qquad (A1.6)$$

and aliasing is observed.

However, if f(n) is an even function so that f(n) = f(-n), this proof may be taken a stage further. Expanding equation (A1.2) into its real and imaginary parts gives

$$\sum_{n=0}^{N-1} f(n).W^{-kn} = \sum_{n=0}^{N-1} f(n).\cos\frac{2\pi nk}{N} - j\sum_{n=0}^{N-1} f(n).\sin\frac{2\pi nk}{N} \qquad (A1.7)$$

and since F(k) is periodic with period = $2\pi/T$ and f(n) is an even function

$$\sum_{n=0}^{N-1} f(n).W^{-kn} = \sum_{n=-N/2}^{N/2-1} f(n).W^{-kn}$$

$$= \sum_{n=-N/2}^{N/2-1} f(n).\cos\frac{2\pi nk}{N} - j\sum_{n=-N/2}^{N/2-1} f(n).\sin\frac{2\pi nk}{N} \qquad (A1.8)$$

But since sin x is an odd function and f(n) is even

$$\sum_{n=-N/2}^{N/2-1} f(n).\sin\frac{2\pi nk}{N} = f(-N/2).\sin\frac{2\pi k}{N}.\left(\frac{-N}{2}\right) \qquad (A1.9)$$

for all integral k. Hence equation (A1.8) reduces to

$$\sum_{n=0}^{N-1} f(n).W^{-kn} = \sum_{n=-N/2}^{N/2-1} f(n).\cos\frac{2\pi nk}{N} = F(k) \qquad (A1.10)$$

indicating that F(k) is a real, even function so that F(N-k) = F(k) which is aliasing in the fullest sense.

APPENDIX 2 - WINDOW FUNCTIONS.

The time-domain functions, g(t), for all the window functions considered are defined below. The corresponding frequency-domain functions, G(ω), are also given in the majority of cases. Unless otherwise stated, the time-domain functions, g(t), are defined for $|t| < NT/2$ and are zero outside these limits.

(i) The Rectangular window[10].

$$g(t) = 1 \tag{A2.1}$$

$$G(\omega) = \frac{\sin(\omega NT/2)}{\omega NT/2} \tag{A2.2}$$

(ii) The Bartlett window[63].

$$g(t) = 1 - \left|\frac{2t}{NT}\right| \tag{A2.3}$$

$$G(\omega) = \left\{\frac{\sin(\omega NT/4)}{\omega NT/4}\right\}^2 \tag{A2.4}$$

(iii) The Hann or Cosine Bell window[64].

$$g(t) = \frac{1}{2}\left\{1 + \cos\frac{2\pi t}{NT}\right\} \tag{A2.5}$$

$$G(\omega) = \frac{\sin(\omega NT/2)}{\omega NT/2} + \frac{1}{2}\left\{\frac{\sin(\omega NT/2+\pi)}{\omega NT/2+\pi} + \frac{\sin(\omega NT/2-\pi)}{\omega NT/2-\pi}\right\} \tag{A2.6}$$

(iv) The variation of the Hann window proposed by Welch[65].

$$g(t) = 1 - \left\{\frac{t - (N-1)T/2}{(N+1)T/2}\right\}^2 \quad \text{for } 0 < t < NT \tag{A2.7}$$

$$= 0 \text{ for other } t.$$

Welch gives an approximate corresponding frequency-domain function. Normalising factors have been ignored.

$$G(\omega) = \left\{\frac{1}{\omega}.2\left[\frac{\sin(N+1)\omega T/2}{(N+1)\omega T/2} - \cos(N+1)\omega T/2\right]\right\}^2 \qquad \text{(A2.8)}$$

(v) The extended Cosine Bell window proposed by Bingham, Godfrey & Tukey[66].

$$g(t) = 1 \quad \text{for } |t| < 0.4NT \qquad \text{(A2.9)}$$
$$= \frac{1}{2}\left[1 - \cos\frac{\pi(|t| - 0.4NT)}{0.1NT}\right] \quad \text{for } 0.4NT \leq |t| < 0.5NT$$

The corresponding frequency-domain function has not been given.

(vi) The Parzen window[67].

$$g(t) = 1 - 6\left|\frac{2t}{NT}\right|^2 + 6\left|\frac{2t}{NT}\right|^3 \quad \text{for } |t| \leq NT/4 \qquad \text{(A2.10)}$$

$$= 2\left[1 - \left|\frac{2t}{NT}\right|\right]^3 \quad \text{for } NT/4 < |t| < NT/2$$

$$G(\omega) = \left[\frac{\sin(\omega NT/8)}{\omega NT/8}\right]^4 \qquad \text{(A2.11)}$$

neglecting normalisation factors.

(vii) The variation of the Parzen window proposed by Welch[65].

$$g(t) = 1 - \left|\frac{t - (N-1)T/2}{(N+1)T/2}\right|^2 \quad \text{for } 0 \leq t < NT \qquad \text{(A2.12)}$$
$$= 0 \quad \text{for other } t.$$

Welch gives an approximate corresponding frequency-domain function. Normalising factors have been ignored.

$$G(\omega) = \left(\frac{\sin(N+1)\omega T/4}{(N+1)\omega T/4}\right)^4 \qquad \text{(A2.13)}$$

(viii) The Blackman window[64].

$$g(t) = 0.42 + 0.5\cos\left(\frac{2\pi t}{NT}\right) + 0.08\cos\left(\frac{4\pi t}{NT}\right) \qquad \text{(A2.14)}$$

$$G(\omega) = 0.42\frac{\sin(\omega NT/2)}{\omega NT/2} + 0.25\left(\frac{\sin(\omega NT/2+\pi)}{\omega NT/2+\pi} + \frac{\sin(\omega NT/2-\pi)}{\omega NT/2-\pi}\right)$$

$$+ 0.04\left(\frac{\sin(\omega NT/2+2\pi)}{\omega NT/2+2\pi} + \frac{\sin(\omega NT/2-2\pi)}{\omega NT/2-2\pi}\right) \qquad \text{(A2.15)}$$

(ix) The Hamming window[64].

$$g(t) = 0.54 + 0.46\cos\left(\frac{2\pi t}{NT}\right) \qquad \text{(A2.16)}$$

$$G(\omega) = 0.54\frac{\sin(\omega NT/2)}{\omega NT/2} + 0.23\left(\frac{\sin(\omega NT/2+\pi)}{\omega NT/2+\pi} + \frac{\sin(\omega NT/2-\pi)}{\omega NT/2-\pi}\right) \qquad \text{(A2.17)}$$

(x) The window formed by convoluting the Hamming window with itself, proposed by Richards[68].

$$g(t) = 0.74\left|\frac{2t}{NT}\right|\left\{1 + 0.35\cos\left(2\pi\left|\frac{2t}{NT}\right|\right)\right\} + 0.157\sin\left(2\pi\left|\frac{2t}{NT}\right|\right) \qquad \text{(A2.18)}$$

The frequency-domain function is the square of the G(ω) given in (ix) above.

(xi) Windows formed by using the Dolph-Chebyshev functions[12,69]. Here the Dolph-Chebyshev function has the required form for the frequency-domain function which is given by

$$G(\omega) = \frac{\left|\cos\left(N\cos^{-1}\left|z_0\cos\left(\omega T/2\right)\right|\right)\right|}{\cosh\left(N\cosh^{-1}z_0\right)} \qquad \text{(A2.19)}$$

The parameter z_0 controls the half-width of the main-lobe, $\delta\omega$, which is defined by the equation

$$z_0 = 1/\cos(\delta\omega T/2) \qquad \text{(A2.20)}$$

that is,

$$\delta\omega = \frac{2}{T}.\cos^{-1}(1/z_0) \qquad \text{(A2.21)}$$

The parameter z_0 also determines the main-lobe : side-lobe height, which is $\cosh(N.\cosh^{-1}z_0)$. It should be noted that all the side-lobes of a Dolph-Chebyshev function are the same height. Increasing z_0 has the effect of increasing both the width of the main-lobe and the ratio of main-lobe height : side-lobe height.

The window function in the time-domain has not been formulated and therefore has to be derived from the frequency-domain function by means of the inverse Fourier transform

$$g(t) = \mathcal{F}^{-1}\{G(\omega)\} = \frac{1}{2\pi}.\int_{-\infty}^{\infty} G(\omega).e^{jt\omega}.d\omega \qquad \text{(A2.22)}$$

For use with discrete-time signals the frequency-domain function must be sampled to give G(k), then the inverse discrete Fourier transform must be used to determine the discrete window function g(n).

$$g(n) = \sum_{k=-N/2}^{N/2-1} G(k).e^{j2\pi nk/N} \qquad \text{(A2.23)}$$

APPENDIX 3 - EXPERIMENTAL SELECTION OF A WINDOW FUNCTION.

This experiment attempts to determine the amplitude of the side-lobes of the resulting frequency-domain function at various deviations, $|\omega-\omega_0|$, from the centre frequency ω_0. 1024 consecutive samples of a sinusoid of frequency ω_0 were modified by transmission through a particular window. The DFT of the resulting sequence was then computed. Many of the window functions give rise to frequency-domain functions which go to zero at integral multiples of $\Delta\omega$ from the centre frequency ω_0. The discrete frequency function produced by the DFT samples at integral multiples of $\Delta\omega$, so if the centre frequency ω_0 is also chosen to be an integral multiple of $\Delta\omega$, the frequency function will for the most part be sampled at its zero points, hence yielding no information on the height of its side-lobes. For this reason the centre frequency, ω_0, was chosen to be half way between two integral multiples of $\Delta\omega$. In consequence the discrete frequency function consists of samples which are $(2n-1).\Delta\omega/2$ from the centre frequency ω_0, where n is an integer. As a result, the side-lobes of the frequency function are sampled very close to their maximum value. Finally, the frequency of the sine wave, ω_0, is chosen to be very close to $2\pi/4T$. This gives a maximum spacing between the main-lobe of the frequency function and its alias, which in turn gives minimum interference between the two.

Table A3.1 summarises the results of this experiment. On the basis of these results the convoluted Hamming window suggested by Richards[68] has been used for all succeeding power spectrum measurements. It can be seen that for $|\omega-\omega_0|\geq 11.\Delta\omega/2$ the side-lobe height is less than 10^{-4} of the main-lobe.

The window formed by use of the Dolph-Chebyshev function[12,69] with $z_0=1/\cos(3{\cdot}5\pi/N)$ has a rather better performance. In theory the maximum side-lobe height for $|\omega-\omega_0|\geq 7.\Delta\omega/2$ is $3{\cdot}35.10^{-5}$, but in practice, because

Table A3.1 Magnitudes of frequency functions from some window functions.

Window Function	Frequency separation from centre frequency ω_o ($\Delta\omega/2 = \pi/NT$)					
	$\Delta\omega/2$	$3.\Delta\omega/2$	$5.\Delta\omega/2$	$7.\Delta\omega/2$	$9.\Delta\omega/2$	$11.\Delta\omega/2$
Rectangular[10]	6.37×10^{-1}	2.12×10^{-1}	1.27×10^{-1}	9.10×10^{-2}	7.07×10^{-2}	5.79×10^{-2}
Bartlett[63]	8.11×10^{-1}	9.01×10^{-2}	3.24×10^{-2}	1.65×10^{-2}	1.00×10^{-2}	6.70×10^{-3}
Hann Cosine Bell 64	8.49×10^{-1}	1.70×10^{-1}	2.43×10^{-2}	8.08×10^{-3}	3.67×10^{-3}	1.98×10^{-3}
Welch-Hann[65]	7.74×10^{-1}	2.92×10^{-2}	6.54×10^{-3}	2.51×10^{-3}	1.26×10^{-3}	7.48×10^{-4}
Extended 66 Cosine Bell	6.97×10^{-1}	2.06×10^{-1}	9.43×10^{-2}	4.08×10^{-2}	1.01×10^{-2}	7.49×10^{-3}
Parzen[67]	9.02×10^{-1}	3.78×10^{-1}	4.90×10^{-2}	3.76×10^{-4}	1.37×10^{-4}	2.09×10^{-3}
Welch-Parzen[65]	8.10×10^{-1}	8.95×10^{-2}	3.26×10^{-2}	1.63×10^{-2}	1.01×10^{-2}	6.57×10^{-3}
Blackman[64]	8.81×10^{-1}	2.94×10^{-1}	1.42×10^{-2}	1.22×10^{-3}	1.06×10^{-3}	6.76×10^{-4}
Hamming[64]	8.17×10^{-1}	1.13×10^{-1}	1.82×10^{-3}	6.59×10^{-3}	7.35×10^{-3}	6.89×10^{-3}
Convoluted Hamming 68	9.04×10^{-1}	3.89×10^{-1}	5.60×10^{-2}	7.32×10^{-4}	1.03×10^{-4}	6.66×10^{-5}
Dolph Cheby-shev $z_o=1/\cos(2\pi/N)$ [12,69]	8.23×10^{-1}	1.22×10^{-1}	4.62×10^{-3}	8.08×10^{-4}	1.10×10^{-2}	1.25×10^{-3}
Dolph Cheby-shev $z_o=1/\cos(3\pi/N)$	8.77×10^{-1}	2.83×10^{-1}	1.49×10^{-2}	3.80×10^{-4}	1.67×10^{-4}	1.98×10^{-4}
Dolph Cheby-shev $z_o=1/\cos3.5\pi/N$	8.93×10^{-1}	3.46×10^{-1}	3.69×10^{-2}	7.21×10^{-5}	6.06×10^{-6}	9.63×10^{-5}
Dolph Cheby-shev $z_o=1/\cos(4\pi/N)$	9.06×10^{-1}	4.00×10^{-1}	6.35×10^{-2}	1.53×10^{-3}	2.15×10^{-5}	2.29×10^{-5}
Dolph Cheby-shev $z_o=1/\cos4.5\pi/N$	9.16×10^{-1}	4.46×10^{-1}	9.23×10^{-2}	5.24×10^{-3}	2.87×10^{-6}	1.94×10^{-7}
Dolph Cheby-shev $z_o=1/\cos(6\pi/N)$	9.37×10^{-1}	5.50×10^{-1}	1.80×10^{-1}	2.90×10^{-2}	1.69×10^{-3}	1.22×10^{-5}

of the limited accuracy of the computer, 10^{-4} appears to be the maximum. However; this function was not used as it requires an inverse Fourier transformation to produce the time-domain window from the Dolph-Chebyshev frequency-domain function. The extra computing time involved in this was not thought to be justified by the small improvement made possible over the convoluted Hamming window[68].

APPENDIX 4 - FAST FOURIER TRANSFORM ALGORITHM.

The mathematical derivation of the actual algorithm used is given below[1]. The example of N=8 is used for clarity. It will be seen, however, that this can be extended for N equal to any power of two. The discrete Fourier transform may be written in the form

$$F_k = \sum_{n=0}^{N-1} f_n . W^{nk} \qquad \text{(A4.1)}$$

where

$$W = e^{-j(2\pi/N)} \qquad \text{(A4.2)}$$

The indices n and k can be represented as binary numbers,

that is,
$$n = 2^2 n_2 + 2^1 n_1 + 2^0 n_0 \qquad \text{(A4.3)}$$

where n_2, n_1, n_0 equal 0 or 1. So f_n can be represented as $f_n = f(n_2,n_1,n_0)$, where the leftmost parameter n_2 represents the most significant bit of the index and the rightmost parameter n_0 represents the least significant bit. Using this notation equation (A4.1) can be rewritten as

$$F(k_2,k_1,k_0) = \sum_{n_0=0}^{1} \sum_{n_1=0}^{1} \sum_{n_2=0}^{1} \left(f(n_2,n_1,n_0) . W^{(2^2 n_2+2^1 n_1+2^0 n_0)(2^2 k_2+2^1 k_1+2^0 k_0)} \right) \qquad \text{(A4.4)}$$

By noting that $W^{m+n} = W^m . W^n$ and that W is the Nth root of unity, equation (A4.4) reduces to

$$F(k_2,k_1,k_0) = \sum_{n_0=0}^{1} \sum_{n_1=0}^{1} \sum_{n_2=0}^{1} \left(f(n_2,n_1,n_0) . W^{(k_0).2^2 n_2} . W^{(2^1 k_1+2^0 k_0).2^1 n_1} . W^{(2^2 k_2+2^1 k_1+2^0 k_0).2^0 n_0} \right) \qquad \text{(A4.5)}$$

This may be analysed into the three ($\log_2 8$) separate summations defined by

$$\text{(i)} \quad F'(n_0,n_1,k_0) = \sum_{n_2=0}^{1} f'(n_0,n_1,n_2).W^{(k_0).2^2 n_2} \qquad \text{(A4.6)}$$

where $f'(n_0,n_1,n_2) = f(n_2,n_1,n_0)$

$$\text{(ii)} \quad F''(n_0,k_1,k_0) = \sum_{n_1=0}^{1} F'(n_0,n_1,k_0).W^{(2^1 k_1+2^0 k_0).2^1 n_1} \qquad \text{(A4.7)}$$

$$\text{(iii)} \quad F(k_2,k_1,k_0) = \sum_{n_0=0}^{1} F''(n_0,k_1,k_0).W^{(2^2 k_2+2^1 k_1+2^0 k_0).2^0 n_0} \qquad \text{(A4.8)}$$

In general there are $\log_2 N$ of these calculation stages. Prior to these stages, the order of the original sequence f_n is rearranged to give the sequence denoted by f'_n. This rearrangement is defined by the equation

$$f'(n_0,n_1,n_2) = f(n_2,n_1,n_0) \qquad \text{(A4.9)}$$

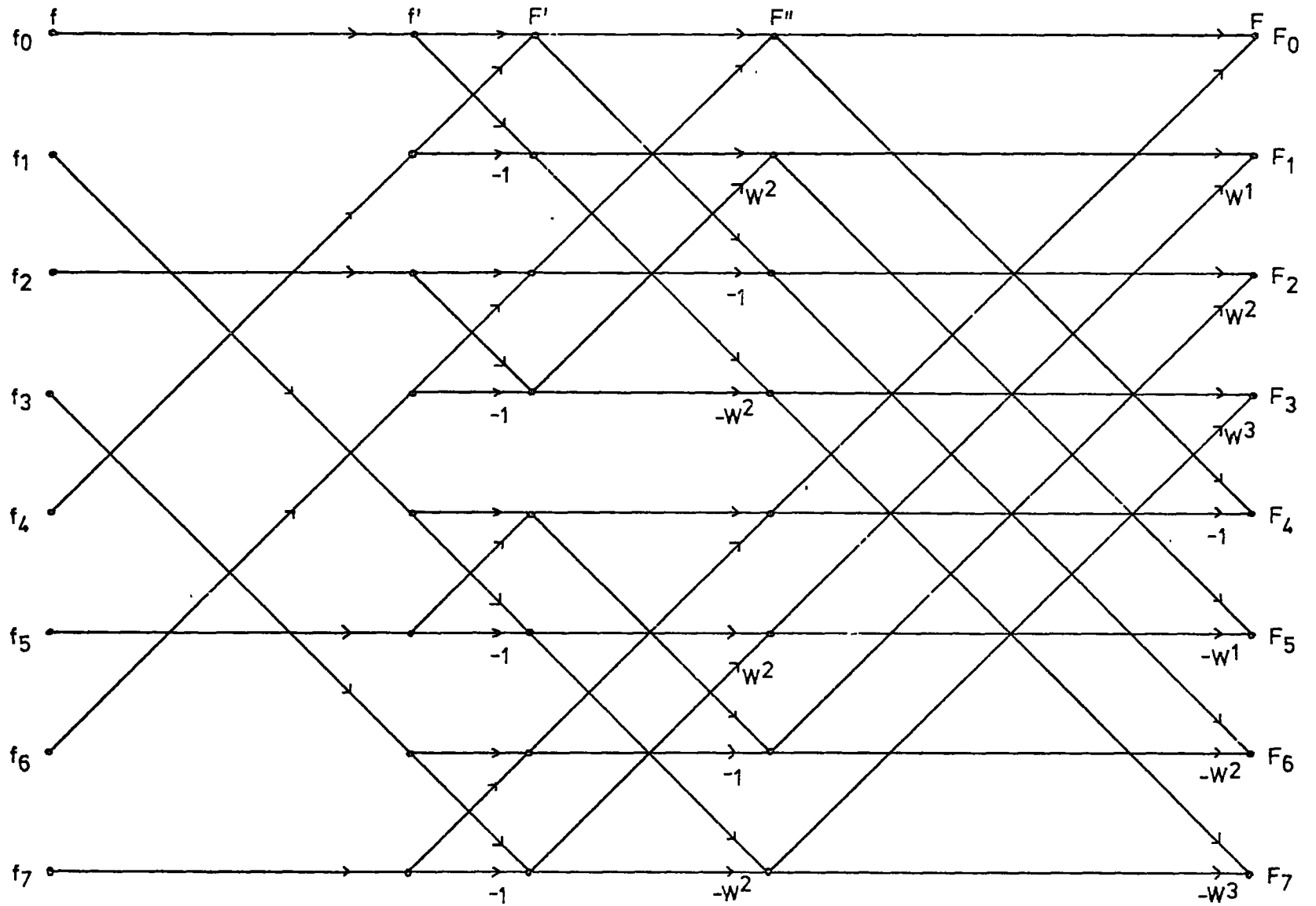
This reordering is called bit reversal and it can be seen that the MSB parameter of the sequence f_n becomes the LSB parameter of the sequence f'_n and vice versa. If N were 16 then the equation defining the reordering would become

$$f'(n_0,n_1,n_2,n_3) = f(n_3,n_2,n_1,n_0) \qquad \text{(A4.10)}$$

so that, for example, f_{10} with n_3, $n_1 = 1$ and n_2, $n_0 = 0$ would become f'_6.

Figure A4.1 is a flow diagram summarising the FFT algorithm for N = 8. The dots represent terms in an array and the arrows indicate the transfer of an array term. If an expression, such as W^2, accompanies an arrow this indicates that the array term is multiplied by the expression

Figure A4.1 A flow diagram of a fast Fourier transform algorithm.



before transfer. The diagram includes the further simplification of equations (A4.6) to (A4.8) which can be made by noting that for $(n.k) \geq N/2$

$$W^{nk} = W^{(\ell - N/2)} = W^{\ell}.W^{-N/2} = W^{\ell}.e^{j\pi} = -W^{\ell}$$

and, of course, $W^0 = 1$.

It can be seen from Figure A4.1 that, once the initial shuffling of the array has been done, the basic operation of the algorithm is to take a pair of terms from the array, f_j and f_k, and to produce a new pair, g_j and g_k. For $j<k$ this operation can be defined by

$$g_j = f_j + (W^x.f_k) \qquad (A4.11)$$

and

$$g_k = f_j - (W^x.f_k) \qquad (A4.12)$$

where x is an integer in the range from 0 to $(N/2 - 1)$. The new terms g_j and g_k overwrite f_j and f_k respectively in the array, so that there is no need to use a second array for any of the calculation stages. However, a second array must be used for the initial reordering of the array. This array would normally be redundant during the ensuing calculation stages, but in the Fortran subroutine FFT listed in Appendix 6 this array has been used to store the N/2 required values of W^n. This permits a further saving in computing time as it is only necessary to work out each required value of W^n once. The appropriate power of W can then be looked up in the array. Hence for every pair of new terms formed the only calculation required is one complex multiplication, one complex addition and one complex subtraction. So, excluding the initial N/2 computations of W^n, this algorithm requires $(N/2).\log_2 N$ of each of the three arithmetic operations mentioned above for the complete Fourier transformation, which demonstrates its efficiency.

APPENDIX 5 - FILTER CHARACTERISTICS.

Table A5.1 First order highpass(i) filter.

Realisation	a_0	b_1	Pole position
Direct Form	0.3115234375	0.6826171875	-0.6826171875
Look-up Table. Reduced-noise. 4 bits.	0.25	0.625	-0.625
Look-up Table. Reduced-noise. 5 bits.	0.3125	0.6875	-0.6875
Look-up Table. Reduced-noise. 6 bits.	0.3125	0.6875	-0.6875
Look-up Table. Reduced-noise. 7 bits.	0.3125	0.6875	-0.6875
Look-up Table. Reduced-noise. 8 bits.	0.3125	0.6796875	-0.6796875

Table A5.2 First order lowpass(i) filter.

Realisation	a_0	b_1	Pole position
Direct Form	0.5400390625	-0.4404296875	0.4404296875
Look-up Table. Reduced-noise. 4 bits.	0.5	-0.375	0.375
Look-up Table. Reduced-noise. 5 bits.	0.5	-0.4375	0.4375
Look-up Table. Reduced-noise. 6 bits.	0.53125	-0.4375	0.4375
Look-up Table. Reduced-noise. 7 bits.	0.546875	-0.4375	0.4375
Look-up Table. Reduced-noise. 8 bits.	0.5390625	-0.4375	0.4375

Table A5.3 First order highpass(ii) filter.

Realisation	a_0	b_1	Pole position
Direct Form	0.8681640625	0.1240234375	-0.1240234375
Look-up Table. Reduced-noise. 4 bits.	0.875	0.125	-0.125
Look-up Table. Reduced-noise. 5 bits.	0.875	0.125	-0.125
Look-up Table. Reduced-noise. 6 bits.	0.875	0.125	-0.125
Look-up Table. Reduced-noise. 7 bits.	0.875	0.125	-0.125
Look-up Table. Reduced-noise. 8 bits.	0.8671875	0.125	-0.125

Table A5.4 First order lowpass(ii) filter.

Realisation	a_0	b_1	Pole position
Direct Form	0.0302734375	-0.9619140625	0.9619140625
Look-up Table. Reduced-noise. 6 bits.	0.03125	-0.9375	0.9375
Look-up Table. Reduced-noise. 7 bits.	0.03125	-0.953125	0.953125
Look-up Table. Reduced-noise. 8 bits.	0.03125	-0.9609375	0.9609375

Table A5.5 Second order bandpass filter.

	Direct Form	Canonic Form	Look-up Table. Reduced-noise. 6 bits.	Look-up Table. Reduced-noise. 7 bits.	Look-up Table. Reduced-noise. 8 bits.
a_0	0.1923828125	0.192308236	0.1875	0.1875	0.1953125
a_1	-0.0302734375	-0.0303068161	-0.03125	-0.03125	-0.03125
a_2	0.0947265625	0.09469699859	0.09375	0.09375	0.09375
b_1	-0.2001953125	-0.2001953125	-0.1875	-0.203125	-0.203125
b_2	0.9013671875	0.9013671875	0.875	0.90625	0.8984375
zeros	0.078680203 ± j0.6972769984	0.07879234165 ± j0.6972679094	0.08333333333 ± j0.7021791477	0.08333333333 ± j0.7021791477	0.08 ± j0.6881860213
poles	0.1000976562 ± j0.9441120943	0.1000976562 ± j0.9441120943	0.09375 ± j0.9307045382	0.1015625 ± j0.9465384612	0.1015625 ± j0.9424025459

Table A5.6 Second order bandstop filter.

	Direct Form	Canonic Form	Look-up Table. Reduced-noise. 4 bits.	Look-up Table. Reduced-noise. 5 bits.	Look-up Table. Reduced-noise. 6 bits.	Look-up Table. Reduced-noise. 7 bits.	Look-up Table. Reduced-noise. 8 bits.
a_0	0.4404296875	0.4400167464	0.375	0.4375	0.4375	0.4375	0.4375
a_1	0.087890625	0.0881729126	0.125	0.0625	0.09375	0.09375	0.0859375
a_2	0.4404296875	0.4400167464	0.375	0.4375	0.4375	0.4375	0.4375
b_1	0.1103515625	0.1103515625	0.125	0.0625	0.125	0.109375	0.109375
b_2	0.0908203125	0.0908203125	0.125	0.0625	0.09375	0.09375	0.09375
zeros	-0.0997782705 ± j0.9950096969	-0.1001926782 ± j0.9949680532	-0.1666666666 ± j0.986013297	-0.0714285714 ± j0.9974457173	-0.1071428571 ± j0.9942436361	-0.1071428571 ± j0.9942436361	-0.0982142857 ± j0.9951652897
poles	-0.0551757813 ± j0.2962700552	-0.0551757813 ± j0.2962700552	-0.0625 ± j0.3479852726	-0.03125 ± j0.2480391854	-0.0625 ± j0.2997394702	-0.0546875 ± j0.3012628044	-0.0546875 ± j0.3012628044

Table A5.7 Second order lowpass filter.

	Direct Form	Canonic Form	Look-up Table. Reduced-noise. 5 bits.	Look-up Table. Reduced-noise. 6 bits.	Look-up Table. Reduced-noise. 7 bits.	Look-up Table. Reduced-noise. 8 bits.
a_0	0.3115234375	0.3111553192	0.25	0.3125	0.3125	0.3125
a_1	0.1240234375	0.1239347458	0.125	0.125	0.125	0.125
a_2	0.1044921875	0.1044216156	0.125	0.125	0.09375	0.109375
b_1	-0.5498046875	-0.5498046875	-0.5	-0.5	-0.53125	-0.546875
b_2	0.1005859375	0.1005859375	0.125	0.125	0.09375	0.109375
zeros	-0.1990595611 ± j0.543873596	-0.1991525423 ± j0.5439958503	-0.25 ± j0.6614378277	-0.2 ± j0.6	-0.2 ± j0.5099019513	-0.2 ± j0.5567764362
poles	0.2749023437 ± j0.1581601685	0.2749023437 ± j0.1581601685	0.25 ± j0.25	0.25 ± j0.25	0.265625 ± j0.1522936616	0.2734375 ± j0.186029389

Table A5.8 Second order highpass filter.

	Direct Form	Canonic Form	Look-up Table. Reduced-noise. 4 bits.	Look-up Table. Reduced-noise. 5 bits.	Look-up Table. Reduced-noise. 6 bits.	Look-up Table. Reduced-noise. 7 bits.	Look-up Table. Reduced-noise. 8 bits.
a_0	0.3115234375	0.3111553192	0.25	0.3125	0.3125	0.3125	0.3125
a_1	-0.1240234375	-0.1239347458	-0.125	-0.125	-0.125	-0.125	-0.125
a_2	0.1044921875	0.1044216156	0.125	0.125	0.09375	0.109375	0.1015625
b_1	0.5498046875	0.5498046875	0.5	0.5	0.53125	0.546875	0.546875
b_2	0.1005859375	0.1005859375	0.125	0.125	0.09375	0.109375	0.1015625
zeros	0.1990595611 ± j0.543873596	0.1991525423 ± j0.5439958503	0.25 ± j0.6614378277	0.2 ± j0.6	0.2 ± j0.5099019513	0.2 ± j0.5567764362	0.2 ± j0.5338539126
poles	-0.2749023437 ± j0.1581601685	-0.2749023437 ± j0.1581601685	-0.25 ± j0.25	-0.25 ± j0.25	-0.265625 ± j0.1522936616	-0.2734375 ± j0.186029389	-0.2734375 ± j0.1636900534

APPENDIX 6 - FORTRAN IV PROGRAM LISTINGS.

Index of Programs.

SPEC

```
C SPEC - MAIN PROGRAM.
C SPECTRAL ANALYSIS OF ROUNDOFF ERROR SEQUENCE.
C QUANTISED SINE WAVE INPUT.
C SINGLE MULTIPLIER.
C ROUNDING ONLY.
C
      DIMENSION IH(1024),SPEC(1024),R(100,512)
      REAL*8 AMP(100),AMUL,SCAL,ANM,PI,A,FR,TOTR,TOTG,AI,AM,X,Y,Z,
     * G,ER,ANORM,ERR,MEAN
      COMPLEX*16 AA(1024),C(1024)
      WRITE (6,400)
  400 FORMAT (1H1,10X,36H SPECTRAL ANALYSIS OF ERROR SEQUENCE)
      MTYPE=0
C
C READ AND WRITE EXPERIMENTAL PARAMETERS.
C
      READ (5,100) MUL,NORM
  100 FORMAT (2I10)
      READ (5,101) LENGTH
      READ (5,101) KR
  101 FORMAT (I10)
      NM=1024
      WRITE (6,401) MUL,NORM
  401 FORMAT (1HO,20X,8H AMUL = ,I10,3H / ,I10)
      WRITE (6,402) LENGTH
  402 FORMAT (1HO,20X,26H SIGNAL DATA WORDLENGTH = ,I2)
      WRITE (6,403) KR,NM
  403 FORMAT (1HO,20X,28H RELATIVE INPUT FREQUENCY = ,I10,3H / ,I10)
      AMUL=DFLOAT(MUL)/DFLOAT(NORM)
      SCAL=DFLOAT(2**(LENGTH-1)-1)
      ANM=DFLOAT(NM)
      PI=3.14159265400
      A=4.000*PI/ANM
      FR=DFLOAT(KR)/ANM
      TOTR=0.000
C
C DO 100 RUNS.
C
      DO 3 I=1,100
      TOTG=0.000
      AI=DFLOAT(I)
      AI=(AI-1.000)/0.203
C
C "AI" DETERMINES THE INITIAL PHASE OFFSET OF THE INPUT SIGNAL.
C PROCESS 1024 SAMPLES.
C
      DO 2 N=1,NM
C
C FORM QUANTISED INPUT SAMPLE FROM THE IDEAL SINE WAVE.
C
      AM=DFLOAT(N-1)
      X=2.000*PI*((FR*AM)+AI)
      X=DSIN(X)*SCAL
      IX=X
      AM=DFLOAT(IX)
      IF (AM-X-0.500) 4,5,6
    6 IX=IX-1
      GO TO 5
    4 IF (AM-X+0.500) 7,5,5
    7 IX=IX+1
    5 X=DFLOAT(IX)
```

SPEC (cont.)

```
C
C DETERMINE THE TRANSMISSION "G" OF THE WINDOW FUNCTION.
C
      AM=DFLOAT(N-1)
      Y=DABS(AM-ANM/2.0D0)
      Z=A*Y
      G=0.74D0*(ANM-2.0D0*Y)/ANM*(1.0D0+0.35D0*DCOS(Z))
     * +0.15700*DSIN(Z)
      TOTG=TOTG+G
C
C PERFORM PRACTICAL THEN IDEAL MULTIPLICATION.
C "ER" IS THE ROUNDOFF ERROR, WHICH IS MULTIPLIED BY THE WINDOW
C TRANSMISSION "G", THEN STORED READY FOR FOURIER ANALYSIS.
C
      CALL PMUL (MUL,IX,IX,NORM,MTYPE)
      X=AMUL*X
      ER=DFLOAT(IX)-X
      AA(N)=DCMPLX(ER*G,0.0D0)
    2 CONTINUE
C
C SUBROUTINE "FFT" PERFORMS A FAST FOURIER TRANSFORM ON THE CONTENTS OF "AA".
C
      Z=-1.0D0
      CALL FFT (AA,D,IH,NM,Z)
      ANORM=ANM/TOTG
C
C FILL ARRAY "R" WITH THE NORMALISED, ABSOLUTE VALUES OF THE TRANSFORMED FN..
C
      DO 1 N=1,512
      N1=N+1
      R(I,N)=ANORM*CDABS(AA(N1))
      TOTR=TOTR+R(I,N)
    1 CONTINUE
    3 CONTINUE
C
C "TOTR" IS THE MEAN RESPONSE OVER ALL FREQUENCIES.
C
      TOTR=TOTR/(0.502*ANM)
      DO 50 N=1,512
      MEAN=0.0D0
      DO 60 I=1,100
      AMP(I)=R(I,N)/TOTR
      MEAN=MEAN+AMP(I)
   60 CONTINUE
C
C "MEAN" IS THE MEAN RELATIVE RESPONSE AT THIS FREQUENCY.
C SUBROUTINE "ERROR" MUST BE USED IF RESULTS ARE TABULATED.
C
      MEAN=MEAN/0.103
      CALL ERROR (AMP,MEAN,ERR)
C
C CONVERT TO DECIBELS.
C
      MEAN=2.0D1*DLOG10(MEAN)
      SPEC(N)=SNGL(MEAN)
   50 CONTINUE
C
C PLOT RESULTS.
C
      CALL PLOT (SPEC,KR)
      STOP
      END
```

```
      SUBROUTINE FFT (AA,D,IH,N,Z)
C
C  FAST FOURIER TRANSFORM ON ARRAY AA.
C  AA HOLDS ORIGINAL SEQUENCE AND RETURNS TRANSFORMED SEQUENCE.
C  N HOLDS LENGTH OF SEQUENCE AND IS A POWER OF 2.
C  Z=-1.0 INDICATES DISCRETE FOURIER TRANSFORMATION.
C  Z=1.0  INDICATES INVERSE DISCRETE FOURIER TRANSFORMATION.
C
      DIMENSION IH(N)
      REAL*8 PI,Z,ANORM,AN
      COMPLEX*16 AA(N),D(N),B,CI,W,Y
      AN=DFLOAT(N)
      PI=3.14159265DO
      CI=(0.0D0,1.0D0)
      W=CDEXP(Z*CI*2.0DO*PI/AN)
C
C  W TAKES THE APPROPRIATE VALUE FOR NORMAL OR INVERSE TRANSFORMATION.
C
      IF (Z) 20,20,21
C
C  NORM IS USED LATER AS A NORMALISING FACTOR.
C  NORM TAKES THE APPROPRIATE VALUE FOR NORMAL OR INVERSE TRANSFORMATION.
C
   20 NORM=N
      GO TO 22
   21 NORM=1
   22 M=N
      DO 1 I=1,N
C
C  THIS LOOP EVALUATES LN=LOG2(N)
C  N LOOPS ENSURE JUMP OUT OF DO LOOP.
C
      LN=I
      M=M/2
      IF (M.EQ.1) GO TO 23
    1 CONTINUE
C
C  LN=LOG2(N)
C
   23 J=LN/2
      DO 2 I=1,N
C
C  L=INDEX OF TERM IN ORIGINAL ARRAY.
C
      L=I-1
      DO 3 K=1,LN
      IH(K)=MOD(L,2)
      IF (L-IH(K)) 25,24,25
   24 L=0
      GO TO 3
   25 L=(L-IH(K))/2
    3 CONTINUE
C
C  IH(K) HOLDS THE KTH. BIT OF THE INDEX L.
C  IH(K)=0 OR 1 FOR K=1 TO LOG2(N).
C  BIT REVERSAL IN IH(K).
C
      DO 4 K=1,J
      M=K-1
      II=IH(K)
      IH(K)=IH(LN-M)
      IH(LN-M)=II
    4 CONTINUE
      M=0
      DO 5 K=1,LN
      IH(K)=2**(K-1)*IH(K)
      M=M+IH(K)
    5 CONTINUE
      M=M+1
```

FFT (cont.)

```
C
C  M IS BIT REVERSED INDEX CORRESPONDING TO I.
C  ORIGINAL ARRAY IS NOW SHUFFLED.
C
      IF (I-M) 26,2,27
   26 D(I)=AA(I)
      AA(I)=AA(M)
      GO TO 2
   27 AA(I)=D(M)
    2 CONTINUE
C
C  AA HAS NOW BEEN SHUFFLED.
C  THIS LOOP EVALUATES ALL THE INTEGER POWERS OF W UP TO N/2.
C  THESE VALUES ARE STORED IN THE ARRAY D.
C  THE ARRAY D IS USED AS A LOOK UP TABLE.
C
      M=N/2
      DO 8 I=1,M
      J=I-1
      D(I)=W**J
    8 CONTINUE
      DO 6 I=1,LN
      J=1
      L=0
      M=N/(2**I)
      II=2**(I-1)
   28 K=J+II
      IND=(M*L)+1
C
C  HERE IS THE COMPLEX MULTIPLICATION.
C
      Y=D(IND)*AA(K)
      B=AA(J)
      AA(J)=B+Y
      AA(K)=B-Y
C
C  NEW TERMS FORMED. COMPLEX ADDITION & SUBTRACTION.
C  IF K=N START NEW STAGE OF TRANSFORMATION.
C
      IF (K.EQ.N) GO TO 6
      L=L+1
      IF (L.EQ.II) L=0
      IF (MOD(J,II).EQ.0) J=K
      J=J+1
C
C  EVALUATE NEXT PAIR OF TERMS.
C
      GO TO 28
    6 CONTINUE
C
C  AA HAS BEEN COMPLETELY TRANSFORMED.
C  THIS LOOP NORMALISES AA.
C
      ANORM=DFLOAT(NORM)
      DO 7 I=1,N
      AA(I)=2.0D0*AA(I)/ANORM
    7 CONTINUE
      RETURN
      END
```

EXEC

```
C EXEC - MAIN PROGRAM.
C EXECUTIVE PROGRAM FOR ALL NOISE VARIANCE EXPERIMENT SIMULATIONS.
C
      INTEGER*4 PSI(32)
      REAL*8 AA,A0,A1,A2,B1,B2,ANORM,TOTEE(100),MEANEE,ERR,AMSQK,
     * TOTSQK(100),ER(1024)
      WRITE (6,400)
  400 FORMAT (1H1,20X,37H NOISE VARIANCE RESULTS OF SIMULATION)
C
C READ DEFINITION OF FILTER.
C "NORDER" ENTERS ORDER OF FILTER.
C "IMPNT" NEGATIVE INDICATES CANONIC FORM, ZERO INDICATES DIRECT FORM,
C POSITIVE INDICATES LOOK UP TABLE FORM.
C "MODE" NEGATIVE INDICATES BLOCK FLOATING POINT ARITHMETIC,
C OTHERWISE FIXED POINT ARITHMETIC IS USED.
C "MTYPE" NEGATIVE INDICATES TRUNCATION, ZERO INDICATES ROUNDING,
C POSITIVE INDICATES SIGN MAGNITUDE TRUNCATION.
C "NCORR" EQUALS ZERO INDICATES THAT THE FILTER COEFFICIENTS ARE TO BE
C MODIFIED TO GIVE AN UNCORRELATED ERROR SEQUENCE.
C
      READ (5,200) NORDER,IMPNT,MODE,MTYPE,NCORR
  200 FORMAT (5I10)
C
C PRINT THIS INFORMATION.
C
      WRITE (6,401) NORDER
  401 FORMAT (1HO,20X,19H ORDER OF FILTER = ,I2)
      IF (IMPNT) 1,2,3
    1 WRITE (6,402)
  402 FORMAT (1HO,20X,13H CANONIC FORM)
      GO TO 4
    2 WRITE (6,403)
  403 FORMAT (1HO,20X,12H DIRECT FORM)
      GO TO 4
    3 WRITE (6,404)
  404 FORMAT (1HO,20X,19H LOOK UP TABLE FORM)
    4 IF (MODE) 6,5,5
    5 WRITE (6,405)
  405 FORMAT (1HO,20X,23H FIXED POINT ARITHMETIC)
      IF (NCORR) 8,7,8
    7 WRITE (6,406)
  406 FORMAT (1HO,20X,17H ZERO CORRELATION)
      GO TO 8
    6 WRITE (6,407)
  407 FORMAT (1HO,20X,32H BLOCK FLOATING POINT ARITHMETIC)
    8 IF (MTYPE) 9,10,11
    9 WRITE (6,408)
  408 FORMAT (1HO,20X,11H TRUNCATION)
      GO TO 12
   10 WRITE (6,409)
  409 FORMAT (1HO,20X,9H ROUNDING)
      GO TO 12
   11 WRITE (6,410)
  410 FORMAT (1HO,20X,26H SIGN MAGNITUDE TRUNCATION)
C
C READ AND WRITE SIGNAL WORDLENGTH.
C SET MAXIMUM ALLOWED SIGNAL AMPLITUDE "MAX".
C
   12 READ (5,201) LENGTH
  201 FORMAT (I10)
      WRITE (6,411) LENGTH
  411 FORMAT (1HO,20X,26H SIGNAL DATA WORDLENGTH = ,I2)
      MAX=2**(LENGTH-1)-1
```

EXEC (cont.)

```
C
C BRANCH ACCORDING TO FILTER ORDER.
C
      IF (NORDER-1) 13,13,14
C
C FIRST ORDER.
C READ, WRITE AND CONVERT FILTER COEFFICIENTS TO FLOATING POINT.
C
   13 READ (5,202) IA0,IB1,NORM
  202 FORMAT (3I10)
      WRITE (6,413) IA0,NORM
      WRITE (6,416) IB1,NORM
      ANORM=DFLOAT(NORM)
      A0=DFLOAT(IA0)/ANORM
      B1=DFLOAT(IB1)/ANORM
C
C BRANCH ACCORDING TO FILTER FORM.
C
      IF (IMPNT) 15,15,16
C
C DIRECT FORM.
C SUBROUTINE "LIMS1" DETERMINES SIGNAL AMPLITUDES.
C STOP IF EITHER COEFFICIENT IS EFFECTIVELY ZERO.
C
   15 CALL LIMS1 (MAX,IA0,IB1,NORM,INAMP,LIMOUT,MTYPE,LIM1,LIM2)
      IF (LIM1) 60,50,60
   50 WRITE (6,500)
  500 FORMAT (1H0,25X,23H A0 IS EFFECTIVELY ZERO)
      STOP
   60 IF (LIM2) 70,80,70
   80 WRITE (6,501)
  501 FORMAT (1H0,25X,23H B1 IS EFFECTIVELY ZERO)
      STOP
C
C BRANCH ACCORDING TO ARITHMETIC MODE.
C
   70 IF (MODE) 17,18,18
C
C BLOCK FLOATING POINT ARITHMETIC.
C SUBROUTINE "BOUND" DETERMINES "IM", THE UPPER BOUND ON THE SCALED OUTPUT.
C CALL FILTERING SUBROUTINE "BLKD1".
C
   17 CALL BOUND (MAX,IA0,IB1,NORM,INAMP,LIMOUT,IM,MTYPE)
      CALL BLKD1
     * (IA0,IB1,NORM,MTYPE,INAMP,A0,B1,ER,TOTEE,MEANEE,IM,TOTSQK,AMSQK)
      GO TO 300
C
C FIXED POINT. BRANCH IF ZERO CORRELATION REQUIRED.
C
   18 IF (NCORR) 19,20,19
C
C CALL FILTERING SUBROUTINE "DIR1".
C
   19 CALL DIR1
     * (IA0,IB1,NORM,MTYPE,INAMP,A0,B1,ER,TOTEE,MEANEE)
      GO TO 300
C
C CALL FILTERING SUBROUTINE "ZCDIR1".
C
   20 CALL ZCDIR1
     * (IA0,IB1,NORM,MTYPE,INAMP,A0,B1,ER,TOTEE,MEANEE)
      GO TO 300
C
C LOOK UP TABLE FORM.
C SUBROUTINE "SETPSI" FILLS TABLE "PSI".
C SUBROUTINE "LIMPSI" DETERMINES LIMITS OF SIGNAL AMPLITUDES.
C
   16 NADDR=2
      CALL SETPSI
      (PSI,IA0,IA1,IA2,IB1,IB2,NORM,ICONST,LENGTH,NULL,NERR,NADDR)
```

```
C
C TEST FOR NULL SET IN TABLE. IF SO, RETURN.
C
      IF (NULL) 30,40,30
   40 WRITE (6,305)
  305 FORMAT (1HO,25X,18H NULL SET IN TABLE)
      STOP
   30 IF (IB1) 35,35,45
   35 MB1=1
      GO TO 46
   45 MB1=-1
   46 CALL LIMPSI (MAX,PSI,LENGTH,ICONST,INAMP,LIMOUT,MB1,NORM)
      WRITE (6,306) ICONST
  306 FORMAT (1HO,25X,35H RESCALING FACTOR FOR PSI VALUES = ,I2)
      IF (INAMP) 61,51,61
   61 IF (LIMOUT) 71,51,71
   51 STOP
C
C CALL FILTERING SUBROUTINE "TAB1".
C
   71 CALL TAB1 (PSI,ICONST,NORM,INAMP,A0,B1,ER,TOTEE,MEANEE)
      GO TO 300
C
C SECOND ORDER.
C INITIALISE SIGNAL BOUNDS.
C
   14 INAMP=MAX
      IM=MAX
C
C BRANCH ACCORDING TO FILTER FORM.
C READ, WRITE AND CONVERT FILTER COEFFICIENTS TO FLOATING POINT.
C
      IF (IMPNT) 89,90,90
   89 READ (5,203) IAA,IA0,IA1,IA2,IB1,IB2,NORM
  203 FORMAT (7I10)
      ANORM=DFLOAT(NORM)
      AA=DFLOAT(IAA)/ANORM
      WRITE (6,412) IAA,NORM
  412 FORMAT (1HO,20X,6H AA = ,I10,3H / ,I10)
      GO TO 91
   90 READ (5,204) IA0,IA1,IA2,IB1,IB2,NORM
  204 FORMAT (6I10)
      ANORM=DFLOAT(NORM)
   91 WRITE (6,413) IA0,NORM
      WRITE (6,414) IA1,NORM
      WRITE (6,415) IA2,NORM
      WRITE (6,416) IB1,NORM
      WRITE (6,417) IB2,NORM
  413 FORMAT (1HO,20X,6H A0 = ,I10,3H / ,I10)
  414 FORMAT (1HO,20X,6H A1 = ,I10,3H / ,I10)
  415 FORMAT (1HO,20X,6H A2 = ,I10,3H / ,I10)
  416 FORMAT (1HO,20X,6H B1 = ,I10,3H / ,I10)
  417 FORMAT (1HO,20X,6H B2 = ,I10,3H / ,I10)
      A0=DFLOAT(IA0)/ANORM
      A1=DFLOAT(IA1)/ANORM
      A2=DFLOAT(IA2)/ANORM
      B1=DFLOAT(IB1)/ANORM
      B2=DFLOAT(IB2)/ANORM
C
C BRANCH ACCORDING TO FILTER FORM.
C THEN BRANCH ON ARITHMETIC MODE AND CALL APPROPRIATE FILTERING SUBROUTINE.
C
      IF (IMPNT) 25,26,27
C
C CANONIC FORM.
C
   25 IF (MODE) 28,29,29
   28 CALL BLKC2 (MAX,IAA,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     * A0,A1,A2,B1,B2,ER,TOTEE,MEANEE,IM,TOTSQK,AMSQK)
      GO TO 300
   29 CALL CAN2 (MAX,IAA,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     * AA,A0,A1,A2,B1,B2,ER,TOTEE,MEANEE)
      GO TO 300
```

EXEC (cont.)

```
C
C DIRECT FORM.
C
   26   IF (MODE) 38,39,39
   38   CALL BLKD2 (MAX,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     *  A0,A1,A2,B1,B2,ER,TOTEE,MEANEE,IM,TOTSQK,AMSQK)
        GO TO 300
   39   CALL DIR2 (MAX,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     *  A0,A1,A2,B1,B2,ER,TOTEE,MEANEE)
        GO TO 300
C
C LOOK UP TABLE FORM. FIXED POINT ARITHMETIC ONLY.
C SUBROUTINE "SETPSI" FILLS THE LOOK UP TABLE "PSI".
C
   27   NADDR=5
        CALL SETPSI
     *  (PSI,IA0,IA1,IA2,IB1,IB2,NORM,ICONST,LENGTH,NULL,NERR,NADDR)
C
C TEST FOR NULL SET IN TABLE. IF SO, RETURN.
C
        IF (NULL) 37,40,37
   37   WRITE (6,306) ICONST
        CALL TAB2
     *  (MAX,PSI,ICONST,NORM,INAMP,A0,A1,A2,B1,B2,ER,TOTEE,MEANEE)
C
C RE-ENTRY POINT FOR ALL FILTERING SUBROUTINES.
C WRITE INPUT AMPLITUDE.
C
  300   WRITE (6,301) INAMP
  301   FORMAT (1HO,25X,20H INPUT AMPLITUDE   = ,I3)
C
C BRANCH ON ARITHMETIC MODE.
C
        IF (MODE) 96,95,95
C
C FIXED POINT ARITHMETIC. BRANCH ON FILTER ORDER.
C
   95   IF (NORDER-1) 98,97,98
C
C FIRST ORDER. WRITE OUTPUT AMPLITUDE.
C
   97   WRITE (6,302) LIMCUT
  302   FORMAT (1HO,25X,20H OUTPUT AMPLITUDE = ,I3)
        GO TO 98
C
C BLOCK FLOATING POINT ARITHMETIC.
C PRINT UPPER BOUND OF SCALED OUTPUT.
C PRINT MEAN SQUARED RESCALING FACTOR "AMSQK".
C
   96   AMSQK=AMSQK/0.1D3
        WRITE (6,303) IM
  303   FORMAT (1HO,25X,26H BOUND OF SCALED OUTPUT = ,I3)
        WRITE (6,304) AMSQK
  304   FORMAT (1HO,10X,33H MEAN SQUARED RESCALING FACTOR = ,D15.8)
C
C "MEANEE" IS THE MEAN VARIANCE OF THE ROUNDOFF ERROR SEQUENCES.
C SUBROUTINE "ERROR" CALCULATES THE ERROR IN "MEANEE".
C PRINT RESULTS.
C
   98   MEANEE=MEANEE/0.1D3
        CALL ERROR (TOTEE,MEANEE,ERR)
        WRITE (6,310) MEANEE,ERR
  310   FORMAT (1HO,10X,23H MEAN NOISE VARIANCE = ,D15.8,5H +/- ,D9.2)
C
C SUBROUTINE "THVAR" CALCULATES THE ERROR VARIANCE ACCORDING TO THE SIMPLE MODEL
C THIS MODEL DOES NOT APPLY TO THE ZERO CORRELATION FORM.
C
        IF (NCORR) 77,78,77
   77   CALL THVAR
     *  (A0,A1,A2,B1,B2,AMSQK,ICONST,LENGTH,NERR,NORDER,IMPNT,MODE)
   78   STOP
        END
```

DIR1

```
      SUBROUTINE DIR1
     * (IA0,IB1,NORM,MTYPE,INAMP,A0,B1,ER,TOTEE,MEANEE)
C
C FILTERING SUBROUTINE.
C FIRST ORDER FILTER.
C DIRECT FORM.
C FIXED POINT ARITHMETIC.
C
      DIMENSION NUM(128)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,TOTR,ER(1024)
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
      AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.0D0
      Y1=0.0D0
      IY1=0
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
      CALL PMUL (IA0,IX,IW,NORM,MTYPE)
      CALL PMUL (IB1,IY1,IV1,NORM,MTYPE)
      IY=IW-IV1
      Y=A0*X-B1*Y1
      IY1=IY
      Y1=Y
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
      ER(N)=DFLOAT(IY)-Y
      TOTR=TOTR+ER(N)
    2 CONTINUE
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
      II=I-1
      IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4 TOTR=TOTR/ANM
      TOTEE(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      MEANEE=MEANEE+TOTEE(II)
    3 CONTINUE
      RETURN
      END
```

ZCDIR1

```
      SUBROUTINE ZCDIR1
     * (IA0,IB1,NORM,MTYPE,INAMP,A0,B1,ER,TOTEE,MEANEE)
C
C FILTERING SUBROUTINE.
C FIRST ORDER FILTER.
C DIRECT FORM.
C FIXED POINT ARITHMETIC.
C MODIFIES THEORETICAL MULTIPLIERS TO GIVE UNCORRELATED ERROR SEQUENCE.
C
      DIMENSION JX(1024),JY(1025),NUM(128),JW(1024),JV(1024)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,
     * MEANA0,MEANB1,RX,RY,CX,CY,E,DELTA,EA0(100),EB1(100),W,V1,GAIN,
     * ERRA0,ERRB1,TOTR,ER(1024),ERA0(1024),ERB1(1024),ERI(1024),
     * TOTEA0(100),TOTEB1(100),TOTEI(100),MERA0,MERB1,MERIN,
     * A0ERR,B1ERR,INERR
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
      AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANA0=0.0D0
      MEANB1=0.0D0
      MEANEE=0.0D0
      MERA0=0.0D0
      MERB1=0.0D0
      MERIN=0.0D0
      Y1=0.0D0
      IY1=0
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
      TOTRA0=0.0D0
      TOTRB1=0.0D0
      TOTRI=0.0D0
      RX=0.0D0
      RY=0.0D0
      CX=0.0D0
      CY=0.0D0
      JY(1)=IY1
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C COMPUTE CORRELATION BETWEEN SIGNALS AND ERRORS AT MULTIPLIERS.
C STORE INPUT SEQUENCE IN ARRAY "JX".
C STORE OUTPUT SEQUENCE IN ARRAY "JY".
C STORE SEQUENCE AT OUTPUT OF A0 IN ARRAY "JW".
C STORE SEQUENCE AT OUTPUT OF B1 IN ARRAY "JV".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
      CX=CX+DFLOAT(IX*IX)
      CY=CY+DFLOAT(IY1*IY1)
      CALL PMUL (IA0,IX,IW,NORM,MTYPE)
      CALL PMUL (IB1,IY1,IV1,NORM,MTYPE)
      IY=IW-IV1
      JX(N)=IX
      JY(N+1)=IY
      JW(N)=IW
      JV(N)=IV1
      E=DFLOAT(IW)-A0*X
      RX=RX+X*E
      E=DFLOAT(IV1)-B1*DFLOAT(IY1)
      RY=RY+DFLOAT(IY1)*E
      IY1=IY
    2 CONTINUE
```

```
      II=I-1
C
C IF THIS IS THE FIRST RUN, RESET "II" TO 1.
C THE ARRAY LOCATIONS FILLED WILL BE OVERWRITTEN ON THE NEXT RUN.
C
      IF (II) 7,7,4
    7 II=1
C
C DELTA IS THE REQUIRED MODIFICATION TO THE COEFFICIENT VALUE.
C THIS MODIFICATION ENSURES ZERO CORRELATION BETWEEN SIGNAL AND ERROR.
C THE MODIFIED VALUE OF "A0" IS STORED IN ARRAY "EA0".
C THE MODIFIED VALUE OF "B1" IS STORED IN ARRAY "EB1".
C
    4 DELTA=RX/CX
      EA0(II)=A0+DELTA
      DELTA=RY/CY
      EB1(II)=B1+DELTA
C
C THE 'INFINITE' CHANNEL FILTERING IS NOW PERFORMED USING THE
C MODIFIED COEFFICIENT VALUES.
C
      DO 6 N=1,NM
      W=EA0(II)*DFLOAT(JX(N))
      V1=EB1(II)*Y1
      Y=W-V1
      Y1=Y
C
C ARRAY "ER" STORES THE ROUNDOFF ERROR AT THE FILTER OUTPUT.
C ARRAY "ERA0" STORES THE ROUNDOFF ERROR COMMITTED AT A0.
C ARRAY "ERB1" STORES THE ROUNDOFF ERROR COMMITTED AT B1.
C ARRAY "ERI" STORES THE ROUNDOFF ERROR AT THE INPUT TO THE POLE SECTION.
C
      ER(N)=DFLOAT(JY(N+1))-Y
      TOTR=TOTR+ER(N)
      ERA0(N)=DFLOAT(JW(N))-W
      TOTRA0=TOTRA0+ERA0(N)
      ERB1(N)=DFLOAT(JV(N))-EB1(II)*DFLOAT(JY(N))
      TOTRB1=TOTRB1+ERB1(N)
      ERI(N)=ERA0(N)-ERB1(N)
      TOTRI=TOTRI+ERI(N)
    6 CONTINUE
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND AFTER RESETTING ACCUMULATORS.
C
      II=I-1
      IF (II) 15,15,16
   15 TOTR=0.0D0
      TOTRA0=0.0D0
      TOTRB1=0.0D0
      TOTRI=0.0D0
      GO TO 3
C
C THE MEANS OF THE ERROR SEQUENCES ARE COMPUTED.
C THE VARIANCES OF THE ERROR SEQUENCES ABOUT THEIR MEANS ARE DETERMINED.
C THESE VARIANCES ARE STORED IN THEIR RESPECTIVE ARRAYS.
C
   16 TOTR=TOTR/ANM
      TOTRA0=TOTRA0/ANM
      TOTRB1=TOTRB1/ANM
      TOTRI=TOTRI/ANM
      TOTEE(II)=0.0D0
      TOTEA0(II)=0.0D0
      TOTEB1(II)=0.0D0
      TOTEI(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
      Y=ERA0(N)-TOTRA0
      TOTEA0(II)=TOTEA0(II)+Y*Y
      Y=ERB1(N)-TOTRB1
      TOTEB1(II)=TOTEB1(II)+Y*Y
      Y=ERI(N)-TOTRI
      TOTEI(II)=TOTEI(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      TOTEA0(II)=TOTEA0(II)/ANM
      TOTEB1(II)=TOTEB1(II)/ANM
      TOTEI(II)=TOTEI(II)/ANM
      MEANA0=MEANA0+EA0(II)
      MEANB1=MEANB1+EB1(II)
      MEANEE=MEANEE+TOTEE(II)
      MERA0=MERA0+TOTEA0(II)
      MERB1=MERB1+TOTEB1(II)
      MERIN=MERIN+TOTEI(II)
    3 CONTINUE
```

ZCDIR1 (cont.)

```
C
C "MERA0" IS THE MEAN VARIANCE OF THE ERROR SEQUENCE CREATED AT A0.
C "MERB1" IS THE MEAN VARIANCE OF THE ERROR SEQUENCE CREATED AT B1.
C "MERIN" IS THE MEAN VARIANCE OF THE EQUIVALENT INPUT ERROR SEQUENCE.
C "MEANA0" IS THE MEAN MODIFIED VALUE OF A0.
C "MEANB1" IS THE MEAN MODIFIED VALUE OF B1.
C "GAIN" IS THE MEAN SQUARED NOISE GAIN OF THE POLE SECTION.
C SUBROUTINE "ERROR" CALCULATES THE ERRORS IN THESE MEANS.
C PRINT RESULTS.
C
      MEANA0=MEANA0/0.1D3
      MEANB1=MEANB1/0.1D3
      MERA0=MERA0/0.1D3
      MERB1=MERB1/0.1D3
      MERIN=MERIN/0.1D3
      CALL ERROR (EA0,MEANA0,ERRA0)
      CALL ERROR (EB1,MEANB1,ERRB1)
      CALL ERROR (TOTEA0,MERA0,A0ERR)
      CALL ERROR (TOTEB1,MERB1,B1ERR)
      CALL ERROR (TOTEI,MERIN,INERR)
      GAIN=MEANEE/MERIN
      WRITE (6,200) MEANA0,ERRA0
  200 FORMAT (1H0,10X,30H MEAN EFFECTIVE VALUE OF A0 = ,D15.8,
     * 5H +/- ,D9.2)
      WRITE (6,205) MEANB1,ERRB1
  205 FORMAT (1H0,10X,30H MEAN EFFECTIVE VALUE OF B1 = ,D15.8,
     * 5H +/- ,D9.2)
      WRITE (6,210) MERA0,A0ERR
  210 FORMAT (1H0,10X,29H MEAN ERROR VARIANCE AT A0 = ,
     * D15.8,5H +/- ,D9.2)
      WRITE (6,220) MERB1,B1ERR
  220 FORMAT (1H0,10X,29H MEAN ERROR VARIANCE AT B1 = ,
     * D15.8,5H +/- ,D9.2)
      WRITE (6,230) MERIN,INERR
  230 FORMAT (1H0,10X,40H MEAN EQUIVALENT INPUT ERROR VARIANCE = ,
     * D15.8,5H +/- ,D9.2)
      WRITE (6,240) GAIN
  240 FORMAT (1H0,10X,13H MEAN GAIN = ,D15.8)
      RETURN
      END
```

ERROR

```
      SUBROUTINE ERROR (AMEAN,AM,ERR)
C
C   EVALUATES THE ERROR (ERR) IN THE MEAN (AM) FROM 100 RUNS (AMEAN)
C
      REAL*8 TOTD,AMEAN(100),AM,DEV,ERR
C
C SUM SQUARED DEVIATION OF INDIVIDUAL VALUES FROM THE MEAN.
C
      TOTD=0.0D0
      DO 1 I=1,100
      DEV=(AMEAN(I)-AM)**2
      TOTD=TOTD+DEV
    1 CONTINUE
      TOTD=DSQRT(TOTD)
      ERR=TOTD/0.1D3
      RETURN
      END
```

SETPSI

```
      SUBROUTINE SETPSI
     * (PSI,IA0,IA1,IA2,IB1,IB2,NORM,ICONST,LENGTH,IM,JM,NADDR)
C
C FILLS THE ARRAY "PSI" WITH THE APPROPRIATE INTEGER VALUES.
C THE FILTER COEFFICIENT DATA "IA0", "IA1", "IA2", "IB1", "IB2" AND "NORM"
C ARE ENTERED.
C THE SIGNAL WORDLENGTH "LENGTH" IS ALSO REQUIRED.
C "ICONST" RETURNS THE SCALING FACTOR OF THE STORED PSI VALUES.
C IF "IM" = 0 IS RETURNED, THE TABLE ONLY CONTAINS ZEROS.
C IF "JM" = 0 IS RETURNED, NO ROUNDOFF HAS OCCURRED IN FORMING
C THE STORED PSI VALUES.
C "NADDR" IS THE NUMBER OF ADDRESS LINES TO THE LOOK UP TABLE.
C "NADDR" EQUALS 2 FOR A FIRST ORDER FILTER.
C "NADDR" EQUAL 5 FOR A SECOND ORDER FILTER.
C
      INTEGER*4 PSI(32),ICOEFF(32),IP(5)
      PSI(1)=0
C
C TEST FOR FIRST OR SECOND ORDER.
C
      I=2
      IF (NADDR-I) 30,30,50
C
C FIRST ORDER. SET IDEAL TABLE "ICOEFF".
C
   30 ICOEFF(2)=-IB1
      ICOEFF(3)=IA0
      ICOEFF(4)=IA0-IB1
      GO TO 60
C
C SECOND ORDER. SET IDEAL TABLE "ICOEFF".
C
   50 DO 11 J1=1,2
      IP(1)=J1-1
      DO 12 J2=1,2
      IP(2)=J2-1
      DO 13 J3=1,2
      IP(3)=J3-1
      DO 14 J4=1,2
      IP(4)=J4-1
      DO 15 J5=1,2
      IP(5)=J5-1
      K=16*IP(1)+8*IP(2)+4*IP(3)+2*IP(4)+IP(5)+1
      ICOEFF(K)=IP(1)*IA0+IP(2)*IA1+IP(3)*IA2-IP(4)*IB1-IP(5)*IB2
   15 CONTINUE
   14 CONTINUE
   13 CONTINUE
   12 CONTINUE
   11 CONTINUE
   60 MTYPE=0
      MUL=2**(LENGTH-1)
      MAX=MUL-1
      ICONST=1
      IM=0
      LM=2**NADDR
C
C FILL ARRAY "PSI".
C
    3 DO 1 I=2,LM
      M=ICOEFF(I)
      CALL PMUL (MUL,M,M,NORM,MTYPE)
      PSI(I)=M
      M=IABS(M)
      IM=IM+M
C
C IS M TOO GREAT?
C
      IF (M-MAX) 1,1,2
    1 CONTINUE
      GO TO 4
C
C PSI VALUES MUST BE SCALED DOWN BY A FACTOR OF 2.
C RESET ARRAY "PSI".
C
    2 MUL=MUL/2
      ICONST=ICONST*2
      IM=0
      GO TO 3
```

SETPSI (cont.)

```
C
C TEST FOR NULL SET IN TABLE.
C
    4   IF (IM) 5,10,5
C
C THIS LOOP DETECTS IF ROUNDOFF HAS OCCURRED IN FORMING THE PSI VALUES.
C
    5   JM=0
        DO 6 I=2,LM
        M=PSI(I)*NORM/MUL
        JM=JM+IABS(M-ICOEFF(I))
    6   CONTINUE
   10   RETURN
        END
```

LIMPSI

```
        SUBROUTINE LIMPSI (MAX,PSI,LENGTH,ICONST,I,J,K,NORM)
C
C DETERMINES "I" THE MAXIMUM ALLOWED INPUT AMPLITUDE.
C DETERMINES "J" THE RESULTING OUTPUT AMPLITUDE.
C "PSI" IS THE LOCK UP TABLE.
C "LENGTH" IS THE SIGNAL WORDLENGTH.
C "MAX" IS THE MAXIMUM PERMITTED AMPLITUDE.
C "ICONST" IS THE RESCALING FACTOR FOR STORED PSI VALUES.
C "K" ENTERS INFORMATION ON THE SIGN OF THE FILTER POLE COEFFICIENT B1.
C K = 1 INDICATES B1 NEGATIVE.
C K = -1 INDICATES B1 POSITIVE.
C
        INTEGER*4 PSI(32)
C
C SET UP INITIAL CONDITIONS.
C START WITH "I" AND "J" AT THE MAXIMUM.
C
        MINUS=-1
        LM1=LENGTH-1
        IMUL=NORM/2
        I=MAX+1
        J=(MAX+1-ICONST)*K
C
C REDUCE INPUT AMPLITUDE.
C
    6   I=I-1
        IF (I) 3,5,3
C
C INPUT AMPLITUDE IS ZERO, RETURN.
C
    5   J=0
        RETURN
C
C DETERMINE FILTER OUTPUT "M" BY CALLING SUBROUTINE "RES2".
C
    3   CALL RES2 (I,J,M,PSI,LM1,ICONST,IMUL,NORM,MINUS)
C
C IS OUTPUT "M" TOO GREAT?
C
        IF (M-MAX) 4,4,6
C
C DETERMINE FILTER OUTPUT "M".
C
   10   CALL RES2 (I,J,M,PSI,LM1,ICONST,IMUL,NORM,MINUS)
    4   IF (IABS(J)-M) 11,8,9
C
C THIS VALUE OF "J" COULD NOT BE ATTAINED.
C REDUCE "J". IF "J" = 0, RETURN.
C
    9   J=K*M
        IF (J) 10,20,10
C
C REDUCE "J" AND TRY AGAIN. IF "J" = 0, RETURN.
C
    8   J=J-K*ICONST
        IF (J) 10,20,10
C
C "M" IS NOW THE MAXIMUM FILTER OUTPUT.
C SET "J" = "M".
C
   11   J=M
   20   RETURN
        END
```

TAB1

```
      SUBROUTINE TAB1 (PSI,ICONST,NORM,INAMP,A0,B1,ER,TOTEF,MEANEE)
C
C FILTERING SUBROUTINE.
C FIRST ORDER FILTER.
C LOOK UP TABLE REALISATION.
C FIXED POINT ARITHMETIC.
C ROUNDING ONLY.
C
      INTEGER*4 NUM(128),PSI(32)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,TOTR,ER(1024)
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
      IMUL=NORM/2
      LM1=LENGTH-1
      MINUS=-1
      AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.0D0
      Y1=0.0D0
      IY1=0
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C SUBROUTINE "RES2" DETERMINES THE FILTER OUTPUT FROM THE LOOK UP TABLE.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
      CALL RES2 (IX,IY1,IY,PSI,LM1,ICONST,IMUL,NORM,MINUS)
      Y=A0*X-B1*Y1
      IY1=IY
      Y1=Y
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
      ER(N)=DFLOAT(IY)-Y
      TOTR=TOTR+ER(N)
    2 CONTINUE
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
      II=I-1
      IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4 TOTR=TOTR/ANM
      TOTEE(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      MEANEE=MEANEE+TOTEE(II)
    3 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RES2 (IX,IY1,IY,PSI,LM1,ICONST,IMUL,NORM,MINUS)
C
C FIRST ORDER FILTER.
C LOOK UP TABLE FORM.
C FIXED POINT ARITHMETIC.
C DETERMINES FILTER OUTPUT "IY" GIVEN INPUTS "IX" AND "IY1".
C "PSI" IS THE LOOK UP TABLE.
C "LM1" IS THE SIGNAL WORDLENGTH MINUS ONE.
C "ICONST" IS THE RESCALING FACTOR.
C "IMUL" EQUALS NORM/2.
C "MINUS" IS A NEGATIVE INTEGER.
C
      INTEGER*4 PSI(32),IP(5)
C
C INITIALISE.
C
      IW=0
      IRO=IX
      IS1=IY1
C
C LOOK UP ALL BUT THE LAST PSI VALUE.
C
      DO 10 L=1,LM1
C
C "IP(1)" PRESENTS SERIAL BIT STREAM OF "IX".
C "IP(2)" PRESENTS SERIAL BIT STREAM OF "IY1".
C
      IP(1)=IABS(MOD(IRO,2))
      IP(2)=IABS(MOD(IS1,2))
      IPP=IP(2)+2*IP(1)
      IRO=IRO/2
      IS1=IS1/2
      IF (IX) 1,2,2
    1 IF (IP(1)) 2,2,3
    3 IRO=IRO-1
    2 IF (IY1) 4,5,5
    4 IF (IP(2)) 5,5,6
    6 IS1=IS1-1
C
C SET ADDRESS OF REQUIRED PSI VALUE.
C ADD PSI VALUE TO ACCUMULATOR.
C
    5 K=IPP+1
      IW=IW+PSI(K)
      IF (L-LM1) 8,7,8
C
C DIVIDE ACCUMULATOR BY 2 TRUNCATING.
C
    8 CALL PMUL (IMUL,IW,IW,NORM,MINUS)
      GO TO 10
C
C LAST BUT ONE PSI VALUE.
C WHEN THE ACCUMULATOR IS DIVIDED BY 2 A DIFFERENT FORM OF ROUNDOFF IS USED.
C THE OVERALL EFFECT IS THAT THE ACCUMULATOR VALUE IS ROUNDED.
C
    7 IF (IW) 30,30,31
   30 IW=IW/2
      GO TO 10
   31 IR=IABS(MOD(IW,2))
      IW=IW/2+IR
   10 CONTINUE
C
C ADDRESS FINAL PSI VALUE.
C
      IPP=0
      IF (IX) 11,12,12
   11 IPP=IPP+2
   12 IF (IY1) 13,14,14
   13 IPP=IPP+1
   14 K=IPP+1
C
C SUBTRACT FROM THE ACCUMULATOR.
C RESCALE AS REQUIRED TO FORM FILTER OUTPUT "IY".
C
      IW=IW-PSI(K)
      IY=IW*ICONST
      RETURN
      END
```

BLKFT2

```
      SUBROUTINE BLKFT2 (KO,IX,IY1,ISCAL,N,MTYPE)
C
C DETERMINES SCALING OF SIGNAL SAMPLES FOR BLOCK FLOATING POINT ARITHMETIC.
C FIRST ORDER FILTER.
C "ISCAL" IS THE INPUT AMPLITUDE OF THE SIGNAL.
C "N" IS THE AMPLITUDE TO WHICH OUTPUT SAMPLES MAY BE SCALED UP.
C "KO" ENTERS THE POWER OF 2 OF THE SCALING FACTOR FOR THE PREVIOUS
C FILTER CYCLE, AND RETURNS THE VALUE FOR THE CURRENT CYCLE.
C
      K1=KO
      K=0
C
C SCALE NEW INPUT BY PREVIOUS SCALING FACTOR.
C TEST TO DETERMINE WHETHER THIS IS TOO GREAT.
C
      M=IABS(IX)*(2**K1)
      IF (M-ISCAL) 7,7,1
C
C FIND MAXIMUM MAGNITUDES AT INPUT AND OUTPUT.
C IF BOTH ZERO, RETURN.
C
    7 IX=IX*(2**K1)
      MAX2=IABS(IX)
      MAX3=IABS(IY1)
      IF (MAX2) 3,8,3
    8 IF (MAX3) 3,2,3
C
C DETERMINE REQUIRED CHANGE IN SCALING FACTOR.
C
    3 L2=(2**K)*MAX2
      L3=(2**K)*MAX3
      IF (L2-ISCAL) 9,9,4
    9 IF (L3-N) 10,10,4
   10 K=K+1
      GO TO 3
    4 K=K-1
C
C DETERMINE NEW SCALING FACTOR.
C SCALE SIGNAL SAMPLES APPROPRIATELY.
C
      KO=K1+K
      L=2**K
      IX=IX*L
      IY1=IY1*L
    2 RETURN
C
C SIGNAL SAMPLES MUST BE SCALED DOWN.
C DETERMINE NEW SCALING FACTOR.
C
    1 MIX=IABS(IX)
    5 L=(2**K)*MIX
      IF (L-ISCAL) 11,11,6
   11 K=K+1
      GO TO 5
    6 K=K-1
      KO=K
C
C SCALE SIGNAL SAMPLES APPROPRIATELY.
C
      MUL=2**KO
      IX=IX*MUL
      IDNOM=(2**K1)
      CALL PMUL (MUL,IY1,IY1,IDNOM,MTYPE)
      RETURN
      END
```

```
      SUBROUTINE BLKD1
     * (IA0,IB1,NORM,MTYPE,INAMP,A0,B1,ER,TOTEE,MEANEE,IM,TOTSQK,AMSQK)
C
C FILTERING SUBROUTINE.
C FIRST ORDER FILTER.
C DIRECT FORM.
C BLOCK FLOATING POINT ARITHMETIC.
C
      DIMENSION NUM(128)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,TOTR,ER(1024),
     * AMSQK,AK,TOTSQK(100)
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
      AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.0D0
      AMSQK=0.0D0
      K0=0
      Y1=0.0D0
      IY1=0
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
      AK=0.0D0
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
C
C SCALE SIGNAL SAMPLES.
C
      CALL BLKFT2 (K0,IX,IY1,INAMP,IM,MTYPE)
      CALL PMUL (IA0,IX,IW,NORM,MTYPE)
      CALL PMUL (IB1,IY1,IV1,NORM,MTYPE)
      IV=IW-IV1
C
C RESCALE FILTER OUTPUT.
C
      MUL=NORM/(2**KC)
      CALL PMUL (MUL,IV,IY,NORM,MTYPE)
      Y=A0*X-B1*Y1
      IY1=IV
      Y1=Y
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
      ER(N)=DFLOAT(IY)-Y
      TOTR=TOTR+ER(N)
      AK=AK+(1.0D0/(2.0D0**(2*K0)))
    2 CONTINUE
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
      II=I-1
      IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4 TOTR=TOTR/ANM
      TOTSQK(II)=AK/ANM
      AMSQK=AMSQK+TOTSQK(II)
      TOTEE(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      MEANEE=MEANEE+TOTEE(II)
    3 CONTINUE
      RETURN
      END
```

DIR2

```
      SUBROUTINE DIR2 (MAX,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     * A0,A1,A2,B1,B2,ER,TOTEE,MEANEE)
C
C FILTERING SUBROUTINE.
C DIRECT FORM.
C SECOND ORDER FILTER.
C FIXED POINT ARITHMETIC.
C
      DIMENSION NUM(128)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,TOTR,ER(1024),
     * A1,A2,B2,X1,X2,Y2
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      ANM=1024
      ANM=DFLOAT(NM)
   70 AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.000
      Y1=0.000
      Y2=Y1
      X1=Y1
      X2=Y1
      IY1=0
      IY2=IY1
      IX1=IY1
      IX2=IY1
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.000
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
      CALL PMUL (IA0,IX,IW,NORM,MTYPE)
      CALL PMUL (IA1,IX1,IW1,NORM,MTYPE)
      CALL PMUL (IA2,IX2,IW2,NORM,MTYPE)
      CALL PMUL (IB1,IY1,IV1,NORM,MTYPE)
      CALL PMUL (IB2,IY2,IV2,NORM,MTYPE)
      IY=IW+IW1+IW2-IV1-IV2
      MAG=MAXO(IABS(IY),IABS(IV1))
      IF (MAG-MAX) 71,71,72
   72 INAMP=INAMP-1
      GO TO 70
   71 Y=A0*X+A1*X1+A2*X2-B1*Y1-B2*Y2
      IX2=IX1
      IX1=IX
      IY2=IY1
      IY1=IY
      X2=X1
      X1=X
      Y2=Y1
      Y1=Y
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
      ER(N)=DFLOAT(IY)-Y
      TOTR=TOTR+ER(N)
    2 CONTINUE
```

DIR2 (cont.)

```
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
      II=I-1
      IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4 TOTR=TOTR/ANM
      TOTEE(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      MEANEE=MEANEE+TOTEE(II)
    3 CONTINUE
      RETURN
      END
```

LIMS1

```
      SUBROUTINE LIMS1(MAX,IA0,IB1,NORM,I,J,MTYPE,M,N)
C
C DETERMINES "I", THE MAXIMUM FILTER INPUT AMPLITUDE.
C DETERMINES "M", THE MAXIMUM OUTPUT AT MULTIPLIER A0.
C DETERMINES "J", THE MAXIMUM FILTER OUTPUT AMPLITUDE.
C DETERMINES "N", THE MAXIMUM OUTPUT AT MULTIPLIER B1.
C "MAX" IS THE MAXIMUM ALLOWABLE SIGNAL AMPLITUDE.
C
      J=MAX+2
      I=J-1
      IB=IABS(IB1)
      M=1
C
C REDUCE INPUT AMPLITUDE.
C
    7 MM=M
      I=I-1
      II=-I
      CALL PMUL(IA0,II,M,NORM,MTYPE)
      IF (M) 11,10,11
   11 IF (MM-M) 9,7,9
C
C FIND CORRESPONDING OUTPUT AMPLITUDE.
C
    9  N=1
C
C REDUCE OUTPUT AMPLITUDE.
C
    3 NN=N
      J=J-1
      JJ=-J
      CALL PMUL(IB,JJ,N,NORM,MTYPE)
      IF (N) 13,10,13
   13 IF (NN-N) 3,2,3
    2 K=J+N+1
      MM=-M
      IF (K-MM) 4,5,3
C
C INPUT AMPLITUDE TOO HIGH.
C
    4 J=J+2
      GO TO 7
    5 IF (J-MAX) 8,4,4
    8 J=J+1
      M=-M
      N=-N
   10 RETURN
      END
```

CAN2

```
      SUBROUTINE CAN2 (MAX,IAA,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     * A0,A1,A2,B1,B2,ER,TOTEE,MEANEE)
C
C FILTERING SUBROUTINE.
C SECOND ORDER FILTER.
C CANONIC FORM.
C FIXED POINT ARITHMETIC.
C
      DIMENSION NUM(128)
      REAL*8 ANM,AMP,MEANEE,TOTEE(100),TOTR,ER(1024),X,Y,W,W1,W2,
     * AA,A0,A1,A2,B1,B2
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
   70 AMP=DFLOAT(INAMP)+0.500
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.000
      W1=0.000
      W2=W1
      IW1=0
      IW2=IW1
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.000
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
      CALL PMUL (IAA,IX,IX,NORM,MTYPE)
      CALL PMUL (IB1,IW1,IV1,NORM,MTYPE)
      CALL PMUL (IB2,IW2,IV2,NORM,MTYPE)
      IW=IX-IV1-IV2
      CALL PMUL (IA0,IW,IU,NORM,MTYPE)
      CALL PMUL (IA1,IW1,IU1,NORM,MTYPE)
      CALL PMUL (IA2,IW2,IU2,NORM,MTYPE)
      IY=IU+IU1+IU2
C
C CHECK THAT SAMPLES ARE WITHIN LIMITS.
C
      MAG=MAXO(IABS(IW),IABS(IV1),IABS(IY),IABS(IU),IABS(IU1),
     * IABS(IU2))
      IF (MAG-MAX) 71,71,72
   72 INAMP=INAMP-1
      GO TO 70
   71 W=AA*X-B1*W1-B2*W2
      Y=A0*W+A1*W1+A2*W2
      IW2=IW1
      IW1=IW
      W2=W1
      W1=W
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
      ER(N)=DFLOAT(IY)-Y
      TOTR=TOTR+ER(N)
    2 CONTINUE
```

CAN2 (cont.)

```
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
      II=I-1
      IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4 TOTR=TOTR/ANM
      TOTEE(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      MEANEE=MEANEE+TOTEE(II)
    3 CONTINUE
      RETURN
      END
```

BOUND

```
      SUBROUTINE BOUND (MAX,IA0,IB1,NORM,I,J,K,MTYPE)
C
C DETERMINES "K" THE MAXIMUM OUTPUT BEFORE RESCALING.
C IF THE OUTPUT AMPLITUDE WITHOUT SCALING "J" IS EQUAL TO "MAX",
C THEN "K" EQUALS "MAX".
C
      IB=IABS(IB1)
      IF (J-MAX) 1,2,1
    2 K=MAX
      RETURN
C
C FIND THE MINIMUM VALUE OF THE SCALING FACTOR "L" SUCH THAT THE
C POSSIBLE SCALED OUTPUT EXCEEDS "MAX".
C
    1 L=0
    3 L=L+1
      JJ=J*(2**L)
      IF (JJ-MAX) 3,4,4
    4 M=2**L
C
C FIND THE MAXIMUM INPUT SIGNAL WHICH CAN BE ACHIEVED BY THIS SCALING FACTOR.
C
      II=MAX+1
    5 II=II-M
      IF (II-I) 6,6,5
C
C DETERMINE CORRESPONDING OUTPUT AT MULTIPLIER A0.
C
    6 II=-II
      CALL PMUL (IA0,II,II,NORM,MTYPE)
      II=-II
C
C DETERMINE MAXIMUM ALLOWABLE OUTPUT AT MULTIPLIER B1.
C
      K=MAX+1
    7 K=K-1
      IF (K-J) 9,9,8
    8 KK=-K
      CALL PMUL (IB,KK,KK,NORM,MTYPE)
      M=II-KK
      IF (M-K) 9,9,7
    9 RETURN
      END
```

TAB2

```
      SUBROUTINE TAB2
     * (MAX,PSI,ICONST,NORM,INAMP,A0,A1,A2,B1,B2,ER,TOTEE,MEANEE)
C
C FILTERING SUBROUTINE.
C SECOND ORDER FILTER.
C LOOK UP TABLE REALISATION.
C FIXED POINT ARITHMETIC.
C ROUNDING ONLY.
C
      INTEGER*4 NUM(128),PSI(32)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,TOTR,ER(1024),
     * A1,A2,B2,X1,X2,Y2
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
      IMUL=NORM/2
      LM1=LENGTH-1
      MINUS=-1
   70 AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.0D0
      Y1=0.0D0
      Y2=Y1
      X1=Y1
      X2=Y1
      IY1=0
      IY2=IY1
      IX1=IY1
      IX2=IY1
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C SUBROUTINE "RES5" DETERMINES THE FILTER OUTPUT FROM THE LOOK UP TABLE.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
      CALL RES5(IX,IX1,IX2,IY1,IY2,IY,PSI,LM1,ICONST,IMUL,NORM,MINUS)
C
C CHECK THAT SAMPLES ARE WITHIN LIMITS.
C
      IF (IABS(IY)-MAX) 71,71,72
   72 INAMP=INAMP-1
      GO TO 70
   71 Y=A0*X+A1*X1+A2*X2-B1*Y1-B2*Y2
      IX2=IX1
      IX1=IX
      IY2=IY1
      IY1=IY
      X2=X1
      X1=X
      Y2=Y1
      Y1=Y
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
      ER(N)=DFLOAT(IY)-Y
      TOTR=TOTR+ER(N)
    2 CONTINUE
```

TAB2 (cont.)

```
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
      II=I-1
      IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4 TOTR=TOTR/ANM
      TOTEE(II)=0.0D0
      DO 5 N=1,NM
      Y=ER(N)-TOTR
      TOTEE(II)=TOTEE(II)+Y*Y
    5 CONTINUE
      TOTEE(II)=TOTEE(II)/ANM
      MEANEE=MEANEE+TOTEE(II)
    3 CONTINUE
      RETURN
      END
```

INTGRL

```
      SUBROUTINE INTGRL (A,B,N,IN,V)
C
C     CALCULATES THE CONTOUR INTEGRAL AROUND THE UNIT CIRCLE OF
C     (1/2*PI*J)*B(Z)*B(Z**-1)/(A(Z)*A(Z**-1)*Z)
C     A(Z)=A(1)*Z**N+A(2)*Z**(N-1)+.....+A(N+1)
C     B(Z)=B(1)*Z**N+B(2)*Z**(N-1)+.....+B(N+1)
C     N=ORDER OF POLYNOMIALS A AND B.
C     IN=DIMENSION OF A AND B IN MAIN PROGRAM.
C     INTEGRAL IS RETURNED IN V.
C     SEE "ASTROM, JURY & AGNIEL".
C
      REAL*8 A(IN),B(IN),AS(11),V,A0,ALFA,BETA
      A0=A(1)
      V=0.0D0
      DO 10 K=1,N
      L=N+1-K
      L1=L+1
      ALFA=A(L1)/A(1)
      BETA=B(L1)/A(1)
      V=V+BETA*B(L1)
      DO 20 I=1,L
      M=L+2-I
      AS(I)=A(I)-ALFA*A(M)
      B(I)=B(I)-BETA*A(M)
   20 CONTINUE
      DO 40 I=1,L
      A(I)=AS(I)
   40 CONTINUE
   10 CONTINUE
      V=V+B(1)**2/A(1)
      V=V/A0
      RETURN
      END
```

RES5

```
      SUBROUTINE RES5
     * (IX,IX1,IX2,IY1,IY2,IY,PSI,LM1,ICONST,IMUL,NORM,MINUS)
C
C SECOND ORDER FILTER.
C LOOK UP TABLE FORM.
C FIXED POINT ARITHMETIC.
C DETERMINES FILTER OUTPUT "IY" GIVEN INPUTS "IX", "IX1", "IX2", "IY1", "IY2".
C "PSI" IS THE LOOK UP TABLE.
C "LM1" IS THE SIGNAL WORDLENGTH MINUS ONE.
C "ICONST" IS THE RESCALING FACTOR.
C "IMUL" EQUALS NORM/2.
C "MINUS" IS A NEGATIVE INTEGER.
C
      INTEGER*4 PSI(32),IP(5)
C
C INITIALISE.
C
      IW=0
      IR0=IX
      IR1=IX1
      IR2=IX2
      IS1=IY1
      IS2=IY2
C
C LOOK UP ALL BUT THE LAST PSI VALUE.
C
      DO 10 L=1,LM1
C
C "IP(1)" PRESENTS SERIAL BIT STREAM OF "IX".
C "IP(2)" PRESENTS SERIAL BIT STREAM OF "IX1".
C "IP(3)" PRESENTS SERIAL BIT STREAM OF "IX2".
C "IP(4)" PRESENTS SERIAL BIT STREAM OF "IY1".
C "IP(5)" PRESENTS SERIAL BIT STREAM OF "IY2".
C
      IP(1)=IABS(MOD(IR0,2))
      IP(2)=IABS(MOD(IR1,2))
      IP(3)=IABS(MOD(IR2,2))
      IP(4)=IABS(MOD(IS1,2))
      IP(5)=IABS(MOD(IS2,2))
      IPP=16*IP(1)+8*IP(2)+4*IP(3)+2*IP(4)+IP(5)
      IR0=IR0/2
      IR1=IR1/2
      IR2=IR2/2
      IS1=IS1/2
      IS2=IS2/2
      IF (IX) 1,2,2
    1 IF (IP(1)) 2,2,3
    3 IR0=IR0-1
    2 IF (IX1) 21,22,22
   21 IF (IP(2)) 22,22,23
   23 IR1=IR1-1
   22 IF (IX2) 34,32,32
   34 IF (IP(3)) 32,32,33
   33 IR2=IR2-1
   32 IF (IY1) 41,42,42
   41 IF (IP(4)) 42,42,43
   43 IS1=IS1-1
   42 IF (IY2) 51,5,5
   51 IF (IP(5)) 5,5,53
   53 IS2=IS2-1
C
C SET ADDRESS OF REQUIRED PSI VALUE.
C ADD PSI VALUE TO ACCUMULATOR.
C
    5 K=IPP+1
      IW=IW+PSI(K)
      IF (L-LM1) 8,7,8
C
C DIVIDE ACCUMULATOR BY 2 TRUNCATING.
C
    8 CALL PMUL (IMUL,IW,IW,NORM,MINUS)
      GO TO 10
```

```
C
C LAST BUT ONE PSI VALUE.
C WHEN THE ACCUMULATOR IS DIVIDED BY 2 A DIFFERENT FORM OF ROUNDOFF IS USED.
C THE OVERALL EFFECT IS THAT THE ACCUMULATOR VALUE IS ROUNDED.
C
    7   IF (IW) 30,30,31
   30   IW=IW/2
        GO TO 10
   31   IR=IABS(MOD(IW,2))
        IW=IW/2+IR
   10   CONTINUE
C
C ADDRESS FINAL PSI VALUE.
C
        IPP=0
        IF (IX) 11,12,12
   11   IPP=IPP+16
   12   IF (IX1) 13,14,14
   13   IPP=IPP+8
   14   IF (IX2) 15,16,16
   15   IPP=IPP+4
   16   IF (IY1) 17,18,18
   17   IPP=IPP+2
   18   IF (IY2) 19,20,20
   19   IPP=IPP+1
   20   K=IPP+1
C
C SUBTRACT FROM THE ACCUMULATOR.
C RESCALE AS REQUIRED TO FORM FILTER OUTPUT "IY".
C
        IW=IW-PSI(K)
        IY=IW*ICONST
        RETURN
        END
```

BLKFT5

```
      SUBROUTINE BLKFT5 (KO,IX,IX1,IX2,IY1,IY2,ISCAL,N,MTYPE)
C
C DETERMINES SCALING OF SIGNAL SAMPLES FOR BLOCK FLOATING POINT ARITHMETIC.
C SECOND ORDER DIRECT FILTER.
C "ISCAL" IS THE INPUT AMPLITUDE OF THE SIGNAL.
C "N" IS THE AMPLITUDE TO WHICH OUTPUT SAMPLES MAY BE SCALED UP.
C "KO" ENTERS THE POWER OF 2 OF THE SCALING FACTOR FOR THE PREVIOUS
C FILTER CYCLE, AND RETURNS THE VALUE FOR THE CURRENT CYCLE.
C
      K1=KO
      K=0
C
C SCALE NEW INPUT BY PREVIOUS SCALING FACTOR.
C TEST TO DETERMINE WHETHER THIS IS TOO GREAT.
C
      M=IABS(IX)*(2**K1)
      IF (M-ISCAL) 7,7,1
C
C FIND MAXIMUM MAGNITUDES AT INPUT AND OUTPUT.
C IF BOTH ZERO, RETURN.
C
    7 IX=IX*(2**K1)
      MAX2=MAXO(IABS(IX),IABS(IX1),IABS(IX2))
      MAX3=MAXO(IABS(IY1),IABS(IY2))
      IF (MAX2) 3,8,3
    8 IF (MAX3) 3,2,3
C
C DETERMINE REQUIRED CHANGE IN SCALING FACTOR.
C
    3 L2=(2**K)*MAX2
      L3=(2**K)*MAX3
      IF (L2-ISCAL) 9,9,4
    9 IF (L3-N) 10,10,4
   10 K=K+1
      GO TO 3
    4 K=K-1
C
C DETERMINE NEW SCALING FACTOR.
C SCALE SIGNAL SAMPLES APPROPRIATELY.
C
      KO=K1+K
      L=2**K
      IX=IX*L
      IX1=IX1*L
      IX2=IX2*L
      IY1=IY1*L
      IY2=IY2*L
    2 RETURN
C
C SIGNAL SAMPLES MUST BE SCALED DOWN.
C DETERMINE NEW SCALING FACTOR.
C
    1 MIX=IABS(IX)
    5 L=(2**K)*MIX
      IF (L-ISCAL) 11,11,6
   11 K=K+1
      GO TO 5
    6 K=K-1
      KO=K
C
C SCALE SIGNAL SAMPLES APPROPRIATELY.
C
      MUL=2**KO
      IX=IX*MUL
      IDNOM=(2**K1)
      CALL PMUL (MUL,IX1,IX1,IDNOM,MTYPE)
      CALL PMUL (MUL,IX2,IX2,IDNOM,MTYPE)
      CALL PMUL (MUL,IY1,IY1,IDNOM,MTYPE)
      CALL PMUL (MUL,IY2,IY2,IDNOM,MTYPE)
      RETURN
      END
```

```
      SUBROUTINE BLKD2 (MAX,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     * A0,A1,A2,B1,B2,ER,TOTE,MEANEE,IM,TOTSQK,AMSQK)
C
C FILTERING SUBROUTINE.
C DIRECT FORM.
C SECOND ORDER FILTER.
C BLOCK FLOATING POINT ARITHMETIC.
C
      DIMENSION NUM(128)
      REAL*8 ANM,AMP,MEANEE,Y1,TOTEE(100),X,Y,A0,B1,TOTR,ER(1024),
     * A1,A2,B2,X1,X2,Y2,AK,TOTSQK(100),AMSQK
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
   70 AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.0D0
      AMSQK=0.0D0
      K0=0
      Y1=0.0D0
      Y2=Y1
      X1=Y1
      X2=Y1
      IY1=0
      IY2=IY1
      IX1=IY1
      IX2=IY1
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
      AK=0.0D0
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,INAMP,X)
C
C SCALE SIGNAL SAMPLES.
C
      CALL BLKFTS (K0,IX,IX1,IX2,IY1,IY2,INAMP,IM,MTYPE)
      CALL PMUL (IA0,IX,IW,NORM,MTYPE)
      CALL PMUL (IA1,IX1,IW1,NORM,MTYPE)
      CALL PMUL (IA2,IX2,IW2,NORM,MTYPE)
      CALL PMUL (IB1,IY1,IV1,NORM,MTYPE)
      CALL PMUL (IB2,IY2,IV2,NORM,MTYPE)
      IV=IW+IW1+IW2-IV1-IV2
C
C CHECK THAT SAMPLES ARE WITHIN LIMITS.
C
      IF (IABS(IV)-IM) 71,71,72
   72 INAMP=INAMP-1
      GO TO 70
   71 IF (IABS(IV1)-MAX) 74,74,73
   73 IM=IM-1
      INAMP=MAX
      GO TO 70
```

BLKD2 (cont.)

```
C
C RESCALE FILTER OUTPUT.
C
   74   MUL=NORM/(2**K0)
        CALL PMUL (MUL,IV,IY,NORM,MTYPE)
        Y=A0*X+A1*X1+A2*X2-B1*Y1-B2*Y2
        IX2=IX1
        IX1=IX
        IY2=IY1
        IY1=IV
        X2=X1
        X1=X
        Y2=Y1
        Y1=Y
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
        ER(N)=DFLOAT(IY)-Y
        TOTR=TOTR+ER(N)
        AK=AK+(1.0D0/(2.0D0**(2*K0)))
    2   CONTINUE
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
        II=I-1
        IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4   TOTR=TOTR/ANM
        TOTSQK(II)=AK/ANM
        AMSQK=AMSQK+TOTSQK(II)
        TOTEE(II)=0.0D0
        DO 5 N=1,NM
        Y=ER(N)-TOTR
        TOTEE(II)=TOTEE(II)+Y*Y
    5   CONTINUE
        TOTEE(II)=TOTEE(II)/ANM
        MEANEE=MEANEE+TOTEE(II)
    3   CONTINUE
        RETURN
        END
```

VAR

```
        SUBROUTINE VAR (MUL,NORM,AMUL,P,LIM,V,MTYPE)
C
C CALCULATES THE NOISE VARIANCE CONTRIBUTED BY THE MULTIPLIER
C (MUL/NORM): "VAR".
C ARRAY "P" HOLDS THE AMPLITUDE DISTRIBUTION FUNCTION FOR THE
C SIGNAL AT THE MULTIPLIER INPUT.
C "LIM" IS THE NUMBER OF SIGNIFICANT ELEMENTS IN P.
C
        DIMENSION P(128)
        REAL*8 AMUL,V,AI
        V=0.0D0
        DO 1 I=2,LIM
        II=I-1
        AI=DFLOAT(II)
        CALL PMUL (MUL,II,IY,NORM,MTYPE)
        AI=AI*AMUL
        AI=AI-DFLOAT(IY)
        V=V+AI*AI*P(I)
    1   CONTINUE
        V=V*2.0D0
        RETURN
        END
```

```
      SUBROUTINE BLKFT3 (KO,IX,IY1,IY2,ISCAL,N,MTYPE)
C
C DETERMINES SCALING OF SIGNAL SAMPLES FOR BLOCK FLOATING POINT ARITHMETIC.
C SECOND ORDER CANONIC FILTER.
C "ISCAL" IS THE INPUT AMPLITUDE OF THE SIGNAL.
C "N" IS THE AMPLITUDE TO WHICH OUTPUT SAMPLES MAY BE SCALED UP.
C "KO" ENTERS THE POWER OF 2 OF THE SCALING FACTOR FOR THE PREVIOUS
C FILTER CYCLE, AND RETURNS THE VALUE FOR THE CURRENT CYCLE.
C
      K1=KO
      K=0
C
C SCALE NEW INPUT BY PREVIOUS SCALING FACTOR.
C TEST TO DETERMINE WHETHER THIS IS TOO GREAT.
C
      M=IABS(IX)*(2**K1)
      IF (M-ISCAL) 7,7,1
C
C FIND MAXIMUM MAGNITUDES AT INPUT AND OUTPUT.
C IF BOTH ZERO, RETURN.
C
    7 IX=IX*(2**K1)
      MAX2=IABS(IX)
      MAX3=MAXO(IABS(IY1),IABS(IY2))
      IF (MAX2) 3,8,3
    8 IF (MAX3) 3,2,3
C
C DETERMINE REQUIRED CHANGE IN SCALING FACTOR.
C
    3 L2=(2**K)*MAX2
      L3=(2**K)*MAX3
      IF (L2-ISCAL) 9,9,4
    9 IF (L3-N) 10,10,4
   10 K=K+1
      GO TO 3
    4 K=K-1
C
C DETERMINE NEW SCALING FACTOR.
C SCALE SIGNAL SAMPLES APPROPRIATELY.
C
      KO=K1+K
      L=2**K
      IX=IX*L
      IY1=IY1*L
      IY2=IY2*L
    2 RETURN
C
C SIGNAL SAMPLES MUST BE SCALED DOWN.
C DETERMINE NEW SCALING FACTOR.
C
    1 MIX=IABS(IX)
    5 L=(2**K)*MIX
      IF (L-ISCAL) 11,11,6
   11 K=K+1
      GO TO 5
    6 K=K-1
      KO=K
C
C SCALE SIGNAL SAMPLES APPROPRIATELY.
C
      MUL=2**KO
      IX=IX*MUL
      IDNOM=(2**K1)
      CALL PMUL (MUL,IY1,IY1,IDNOM,MTYPE)
      CALL PMUL (MUL,IY2,IY2,IDNOM,MTYPE)
      RETURN
      END
```

```
      SUBROUTINE BLKC2 (MAX,IAA,IA0,IA1,IA2,IB1,IB2,NORM,MTYPE,INAMP,
     * A0,A1,A2,B1,B2,ER,TOTEE,MEANEE,IM,TOTSQK,AMSQK)
C
C FILTERING SUBROUTINE.
C SECOND ORDER FILTER.
C CANONIC FORM.
C BLOCK FLOATING POINT ARITHMETIC.
C
      DIMENSION NUM(128)
      REAL*8 ANM,AMP,MEANEE,TOTEE(100),TOTR,ER(1024),X,Y,W,W1,W2,
     * AA,A0,A1,A2,B1,B2,AMSQK,TOTSQK(100),AK
C
C "NM" IS THE LENGTH OF EACH RANDOM SEQUENCE.
C "AMP" IS THE AMPLITUDE OF THE FLOATING POINT
C RANDOM SEQUENCE, BEFORE QUANTISATION.
C SUBROUTINE "PRAND" INITIALISES THE RANDOM NUMBER GENERATOR.
C
      NM=1024
      ANM=DFLOAT(NM)
   70 AMP=DFLOAT(INAMP)+0.5D0
      CALL PRAND (IUK,IVK,NUM)
      MEANEE=0.0D0
      AMSQK=0.0D0
      K0=0
      W1=0.0D0
      W2=W1
      IW1=0
      IW2=IW1
C
C DO 101 RUNS.
C THE FIRST RUN IS TO ALLOW THE MEAN ERROR TO REACH EQUILIBRIUM.
C THE VARIANCE RECORDED FOR THE FIRST RUN IS REJECTED.
C
      DO 3 I=1,101
      TOTR=0.0D0
      AK=0.0D0
C
C FILTER A RANDOM SEQUENCE OF LENGTH "NM".
C
      DO 2 N=1,NM
C
C SUBROUTINE "RAND" RETURNS A RANDOM INTEGER "IX" AND THE
C FLOATING POINT EQUIVALENT "X".
C THE PARALLEL FILTERING IN THE INTEGER AND FLOATING POINT
C CHANNELS FOLLOWS.
C
      CALL RAND (IUK,IVK,NUM,IX,AMP,IINAMP,X)
C
C SCALE SIGNAL SAMPLES.
C
      CALL BLKFT3 (K0,IX,IW1,IW2,INAMP,IM,MTYPE)
      CALL PMUL (IAA,IX,IX,NORM,MTYPE)
      CALL PMUL (IB1,IW1,IV1,NORM,MTYPE)
      CALL PMUL (IB2,IW2,IV2,NORM,MTYPE)
      IW=IX-IV1-IV2
C
C CHECK THAT INTERMEDIATE OUTPUT IS WITHIN LIMIT.
C
      IF (IABS(IW)-IM) 71,71,72
   72 INAMP=INAMP-1
      GO TO 70
   71 CALL PMUL (IA0,IW,IU,NORM,MTYPE)
      CALL PMUL (IA1,IW1,IU1,NORM,MTYPE)
      CALL PMUL (IA2,IW2,IU2,NORM,MTYPE)
      IV=IU+IU1+IU2
C
C CHECK THAT SAMPLES ARE WITHIN LIMITS.
C
      MAG=MAX0(IABS(IV1),IABS(IV),IABS(IU),IABS(IU1),IABS(IU2))
      IF (MAG-MAX) 74,74,73
   73 IM=IM-1
      INAMP=MAX
      GO TO 70
```

BLKC2 (cont.)

```
C
C RESCALE FILTER OUTPUT.
C
   74  MUL=NORM/(2**KO)
       CALL PMUL (MUL,IV,IY,NORM,MTYPE)
       W=AA*X-B1*W1-B2*W2
       Y=A0*W+A1*W1+A2*W2
       IW2=IW1
       IW1=IW
       W2=W1
       W1=W
C
C THE ROUNDOFF ERROR IS STORED IN ARRAY "ER".
C
       ER(N)=DFLOAT(IY)-Y
       TOTR=TOTR+ER(N)
       AK=AK+(1.0D0/(2.0D0**(2*KO)))
    2  CONTINUE
C
C IF THIS IS THE FIRST RUN, COMMENCE SECOND IMMEDIATELY.
C
       II=I-1
       IF (II) 3,3,4
C
C CALCULATE MEAN ERROR "TOTR".
C CALCULATE VARIANCE WITH RESPECT TO MEAN.
C STORE VARIANCE IN ARRAY "TOTEE".
C
    4  TOTR=TOTR/ANM
       TOTSQK(II)=AK/ANM
       AMSQK=AMSQK+TOTSQK(II)
       TOTEE(II)=0.0D0
       DO 5 N=1,NM
       Y=ER(N)-TOTR
       TOTEE(II)=TOTEE(II)+Y*Y
    5  CONTINUE
       TOTEE(II)=TOTEE(II)/ANM
       MEANEE=MEANEE+TOTEE(II)
    3  CONTINUE
       RETURN
       END
```

PRAND

```
      SUBROUTINE PRAND (II,JJ,NUM)
C
C SETS 128 RANDOM INTEGERS IN THE TABLE "NUM".
C INITIALISES "II" AND "JJ" FOR USE IN SUBROUTINE "RAND".
C
      DIMENSION NUM(128)
      II=1
      JJ=0
      DO 1 M=1,128
      II=II*65539
      IF (II) 2,3,3
    2 II=II+2147483647+1
    3 JJ=JJ*129+1
      IF (JJ) 4,5,5
    4 JJ=JJ+2147483647+1
    5 NUM(M)=II
    1 CONTINUE
      RETURN
      END
```

RAND

```
      SUBROUTINE RAND (K,J,NUM,I,AMP,INAMP,X)
C
C RETURNS RANDOM INTEGERS "I" UNIFORMLY DISTRIBUTED.
C "INAMP" LIMITS THE AMPLITUDE OF "I".
C "X" IS THE FLOATING POINT EQUIVALENT OF "I".
C "NUM" IS THE TABLE OF RANDOM INTEGERS SET UP INITIALLY BY "PRAND".
C
      DIMENSION NUM(128)
      REAL*8 X,EX,AMP
C
C DETERMINE NEXT RANDOM INTEGER "J".
C
      J=J*129+1
      IF (J) 4,5,5
    4 J=J+2147483647+1
C
C USE LOW ORDER 7 BITS OF "J" TO ADDRESS TABLE "NUM".
C RANDOM INTEGER "I" IS THEN READ FROM "NUM".
C
    5 M=MOD(J,128)+1
      I=NUM(M)
C
C DETERMINE NEW RANDOM INTEGER "K" TO BE PLACED IN LOCATION OF "NUM".
C
      K=K*65539
      IF (K) 2,3,3
    2 K=K+2147483647+1
    3 NUM(M)=K
C
C FORM FLOATING POINT RANDOM NUMBER "X" FROM "I".
C SCALE "X" APPROPRIATELY.
C
      X=DFLOAT(I)
      X=X*0.46566128750-9
      X=(X-0.5D0)*2.0D0*AMP
C
C SET "I" TO THE BEST INTEGER APPROXIMATION TO "X".
C
      I=X
      EX=DFLOAT(I)
      IF (EX-X-0.5D0) 10,12,11
   11 I=I-1
      GO TO 13
   10 IF (EX-X+0.5D0) 14,12,13
   14 I=I+1
      GO TO 13
   12 IF (I) 14,13,11
   13 IF (INAMP-IABS(I)) 15,16,16
   15 IF (I) 17,16,18
   17 I=I+1
      GO TO 16
   18 I=I-1
C
C SET "X" EQUAL TO "I".
C
   16 X=DFLOAT(I)
      RETURN
      END
```

```
      SUBROUTINE THVAR
     * (A0,A1,A2,B1,B2,SQK,ICONST,LENGTH,JM,NORDER,IMPNT,MODE)
C
C CALCULATES THE ERROR VARIANCE ACCORDING TO THE SIMPLE MODEL.
C
      REAL*8 A0,A1,A2,B1,B2,SQK,A(3),B(3),CONST,SIGMA,AJ,V,ZETA,ETA
C
C SET UP ENTRY PARAMETERS FOR "INTGRL".
C
      A(1)=1.0D0
      A(2)=B1
      A(3)=B2
C
C BRANCH ON FILTER FORM.
C
      IF (IMPNT) 1,2,2
C
C CANONIC FORM.
C
    1 B(1)=A0
      B(2)=A1
      B(3)=A2
      GO TO 10
C
C DIRECT OR LOOK UP FORM.
C
    2 B(1)=0.0D0
      B(2)=B(1)
      B(3)=A(1)
C
C SUBROUTINE "INTGRL" EVALUATES THE SQUARED NOISE GAIN "V" OF THE FILTER.
C
   10 IN=NORDER+1
      CALL INTGRL (A,B,NORDER,IN,V)
C
C BRANCH ON FILTER FORM.
C
      IF (IMPNT) 5,4,3
C
C LOOK UP TABLE FORM.
C
    3 CONST=DFLOAT(ICONST*ICONST)
C
C TEST FOR "JM"=0
C
      IF (JM) 6,7,6
C
C NO ROUNDOFF IN PSI VALUES.
C
    7 ETA=V*CONST/1.201
      WRITE (6,210)
  210 FORMAT (1H0,10X,26H NO ROUNDOFF IN PSI VALUES)
      GO TO 20
C
C ROUNDOFF IN PSI VALUES.
C
    6 SIGMA=0.0D0
      DO 12 J=1,LENGTH
      AJ=DFLOAT(1-J)
      SIGMA=SIGMA+2.0D0**AJ
   12 CONTINUE
      ETA=(SIGMA+1.0D0)*CONST*V/1.201
      GO TO 20
```

THVAR (cont.)

```
C
C DIRECT FORM. BRANCH ACCORDING TO ARITHMETIC MODE.
C
    4  IF (MODE) 30,40,40
C
C FIXED POINT. BRANCH ACCORDING TO ORDER OF FILTER.
C
   40  IF (NORDER-1) 35,35,45
C
C FIRST ORDER.
C
   35  ETA=2.0D0*V/1.2D1
       GO TO 20
C
C SECOND ORDER.
C
   45  ETA=5.0D0*V/1.2D1
       GO TO 20
C
C BLOCK FLOATING POINT ARITHMETIC. BRANCH ACCORDING TO FILTER ORDER.
C
   30  IF (MODE) 31,31,41
C
C FIRST ORDER.
C
   31  ZETA=SQK*(2.0D0+B1*B1)
       GO TO 19
C
C SECOND ORDER.
C
   41  ZETA=SQK*(5.0D0+B1*B1+2.0D0*B2*B2)
       GO TO 19
C
C CANONIC FORM. BRANCH ACCORDING TO ARITHMETIC MODE.
C
    5  IF (MODE) 32,42,42
C
C FIXED POINT ARITHMETIC.
C
   42  ETA=(1.0D0+V)/4.0D0
       GO TO 20
C
C BLOCK FLOATING POINT ARITHMETIC.
C
   32  ZETA=SQK*(3.0D0+B1*B1+2.0D0*B2*B2)
       ETA=(1.0D0+3.0D0*SQK+ZETA*V)/1.2D1
       GO TO 20
   19  ETA=(V*ZETA+1.0D0)/1.2D1
   20  WRITE (6,200) ETA
  200  FORMAT (1H0,10X,30H THEORETICAL NOISE VARIANCE = ,D15.8)
       RETURN
       END
```

VARAMP

```
C VARAMP - MAIN PROGRAM.
C PLOTS THE VARIANCE OF THE ERROR SEQUENCE AS A FUNCTION OF INPUT AMPLITUDE.
C SINGLE MULTIPLIER.
C A UNIFORM AMPLITUDE DISTRIBUTION POSITIVE INTEGER RAMP IS THE INPUT SEQUENCE.
C ROUNDING ONLY.
C
      DIMENSION A(127),B(127)
      REAL*8 TOTEE,AI,AMUL,Y,ER
      WRITE (6,400)
  400 FORMAT(1H1,5X,46H PLOT OF ERROR VARIANCE AT A SINGLE MULTIPLIER)
      MTYPE=0
C
C READ, WRITE AND CONVERT MULTIPLIER VALUE TO FLOATING POINT.
C
      READ (5,100) MUL,NORM
  100 FORMAT (2I10)
      AMUL=DFLOAT(MUL)/DFLOAT(NORM)
      TOTEE=0.0D0
C
C "I" IS THE MAXIMUM AMPLITUDE OF THE INTEGER RAMP.
C
      DO 1 I=1,127
      AI=DFLOAT(I)
      A(I)=SNGL(AI)
      CALL PMUL (MUL,1,IY,NORM,MTYPE)
      Y=AI*AMUL
C
C "ER" IS THE ROUNDOFF ERROR. THIS IS THEN SQUARED AND ADDED TO THE
C ACCUMULATING SQUARED DEVIATION "TOTEE".
C
      ER=DFLOAT(IY)-Y
      ER=ER*ER
      TOTEE=TOTEE+ER
C
C "B(I)" IS THE VARIANCE AT THIS AMPLITUDE.
C
      B(I)=SNGL(TOTEE/(AI+0.5D0))
    1 CONTINUE
C
C PLOT "B(I)" AS A FUNCTION OF "A(I)".
C
      CALL PLOT2 (A,B)
      STOP
      END
```

PMUL

```
      SUBROUTINE PMUL(ICOEFF,IDATA,IANS,NORM,MTYPE)
C
C "IDATA" IS MULTIPLIED BY (ICOEFF/NORM) AND THE ANSWER IS
C APPROXIMATED TO AN INTEGER.
C "MTYPE" NEGATIVE INDICATES TRUNCATION.
C "MTYPE" ZERO INDICATES ROUNDING.
C "MTYPE" POSITIVE INDICATES SIGN MAGNITUDE TRUNCATION.
C FOR ROUNDING IF EPROR = 1/2 MAGNITUDE IS DECREASED.
C "IANS" RETURNS ANSWER.
C
      IANS=IDATA*ICOEFF
      IR=MOD(IANS,NORM)
C
C IR<0 INDICATES IANS<0
C IR>0 INDICATES IANS>0
C MAGNITUDE OF "IR" INCREASES WITH MAGNITUDE OF ERROR.
C MAGNITUDE OF "IR" IS IN RANGE 0 TO (NORM-1).
C
      IANS=IANS/NORM
C
C  IANS IS SIGN MAGNITUDE TRUNCATED RESULT.
C
      IF (MTYPE) 10,20,30
   10 IF (IR) 13,30,30
   13 IANS=IANS-1
   30 RETURN
   20 IF (IR) 21,30,23
   21 IF (IR+NORM/2) 13,30,30
   23 IF (IR-NORM/2) 30,30,25
   25 IANS=IANS+1
      RETURN
      END
```

AMPMOD

```
C AMPMOD - MAIN PROGRAM.
C NOISE VARIANCE PREDICTION BY SOLUTION OF SIMULTANEOUS EQUATIONS.
C FIRST ORDER FILTER.
C DIRECT FORM.
C FIXED POINT ARITHMETIC.
C ASSUMES UNIFORM INPUT AMPLITUDE DISTRIBUTION.
C ASSUMES SYMMETRICAL INPUT SIGNAL.
C APPROPRIATE FOR SYMMETRICAL ERROR PROCESSES.
C SOLVES LINEAR SIMULTANEOUS EQUATIONS TO FIND AMPLITUDE DISTRIBUTION OF OUTPUT
C FILTER COEFFICIENTS AND SIGNAL WORDLENGTH REQUIRED AS DATA.
C ALSO REQUIRES DATA ON TYPE OF ROUNDOFF TO BE USED.
C "MTYPE" EQUAL TO ZERO INDICATES ROUNDING.
C "MTYPE" POSITIVE  INDICATES SIGN MAGNITUDE TRUNCATION.
C
      DIMENSION ICP(128),PO(128),A(16384)
      REAL*8 ANORM,A0,B1,VARA0,VARB1,VAREQ
      WRITE (6,400)
  400 FORMAT (1H1,20X,26H NOISE VARIANCE PREDICTION)
      WRITE (6,404)
  404 FORMAT (1H0,20X,38H BY SOLUTION OF SIMULTANEOUS EQUATIONS)
      WRITE (6,401)
  401 FORMAT (1H0,20X,40H FIRST ORDER, DIRECT, FIXED POINT FILTER)
C
C READ INTEGER FILTER COEFFICIENTS AND NORMALISING FACTOR (A POWER OF 2).
C "IA0" MUST BE POSITIVE.
C SUM OF MAGNITUDES OF "IA0" AND "IB1" MUST BE LESS THAN "NORM".
C
      READ (5,100) IA0,IB1,NORM
  100 FORMAT (3I10)
C
C READ SIGNAL DATA WORDLENGTH,
C A POSITIVE INTEGER LESS THAN OR EQUAL TO 8.
C
      READ (5,101) LENGTH
  101 FORMAT (I10)
C
C READ VALUE OF "MTYPE" TO DEFINE ROUNDOFF PROCESS.
C
      READ (5,101) MTYPE
      IF (MTYPE) 9,10,9
    9 WRITE (6,402)
  402 FORMAT (1H0,20X,26H SIGN MAGNITUDE TRUNCATION)
      GO TO 11
   10 WRITE (6,403)
  403 FORMAT (1H0,20X,9H ROUNDING)
   11 WRITE (6,300) IA0,NORM
  300 FORMAT (1H0,20X,6H A0 = ,I10,3H / ,I10)
      WRITE (6,301) IB1,NORM
  301 FORMAT (1H0,20X,6H B1 = ,I10,3H / ,I10)
      WRITE (6,302) LENGTH
  302 FORMAT (1H0,20X,26H SIGNAL DATA WORDLENGTH = ,I2)
C
C CONVERT COEFFICIENTS TO FLOATING POINT.
C
      ANORM=DFLOAT(NORM)
      A0=DFLOAT(IA0)/ANORM
      B1=DFLOAT(IB1)/ANORM
C
C FIND MAXIMUM SIGNAL AMPLITUDE, "MAX", ALLOWED BY WORDLENGTH.
C CALL "LIMS1" WHICH DETERMINES THE MAXIMUM ALLOWED INPUT AMPLITUDE
C TO THE FILTER, AND THE CORRESPONDING OUTPUT AMPLITUDE.
C
      MAX=2**(LENGTH-1)-1
      CALL LIMS1 (MAX,IA0,IB1,NORM,INAMP,LIMOUT,MTYPE,LIM1,LIM2)
      WRITE (6,303) INAMP
  303 FORMAT (1H0,25X,20H INPUT AMPLITUDE  = ,I4)
      WRITE (6,304) LIMOUT
  304 FORMAT (1H0,25X,20H OUTPUT AMPLITUDE = ,I4)
      IF (LIM1) 60,70,60
   70 WRITE (6,410)
  410 FORMAT (1H0,25X,23H A0 IS EFFECTIVELY ZERO)
      STOP
   60 IF (LIM2) 65,75,65
   75 WRITE (6,420)
  420 FORMAT (1H0,25X,23H B1 IS EFFECTIVELY ZERO)
      STOP
```

```
C
C CALCULATE "CONST", THE PROBABILITY OF A GIVEN SIGNAL AT THE
C FILTER INPUT. FILL ARRAY "PO" WITH THIS VALUE.
C CALL "VAR" WHICH CALCULATES THE NOISE VARIANCE AT MULTIPLIER A0.
C
   65  CONST=1.0E0/FLOAT(2*INAMP+1)
       INAMP=INAMP+1
       DO 1 I=1,INAMP
       PO(I)=CONST
    1  CONTINUE
       CALL VAR (IA0,NORM,A0,PO,INAMP,VARA0,MTYPE)
C
C CALL "MULAR" WHICH FILLS ARRAY "IOP" WITH THE PRACTICAL VALUES
C OF MULTIPLYING THE INPUT SIGNAL LEVELS BY (IA0/NORM).
C
       CALL MULAR (IA0,IOP,INAMP,NORM,MTYPE)
C
C ARRAY "PO" IS NOW SET TO THE AMPLITUDE PROBABILITY FUNCTION
C OF THE SIGNAL AT THE OUTPUT OF THE MULTIPLIER A0.
C IT IS INITIALISED TO ZERO.
C
       LIM1=LIM1+1
       DO 2 I=1,LIM1
       PO(I)=0.0E0
    2  CONTINUE
       PO(1)=CONST
       DO 3 I=2,INAMP
       JK=IOP(I)
       IF (JK) 50,51,50
C
C NON-ZERO SIGNAL AT OUTPUT OF A0.
C
   50  JK=JK+1
       PO(JK)=PO(JK)+CONST
       GO TO 3
C
C ZERO SIGNAL AT OUTPUT OF A0.
C
   51  PO(1)=PO(1)+2.0E0*CONST
    3  CONTINUE
C
C CALL "MULAR" WHICH FILLS ARRAY "IOP" WITH THE PRACTICAL RESULTS
C OF MULTIPLYING THE OUTPUT SIGNAL LEVELS BY (IB1/NORM).
C
       LIMOUT=LIMOUT+1
       CALL MULAR (IB1,IOP,LIMOUT,NORM,MTYPE)
C
C SUBROUTINE "SETA" SETS UP THE MATRIX "A" WITH THE COEFFICIENTS
C WHICH DEFINE THE SIMULTANEOUS EQUATIONS WHICH GOVERN THE
C AMPLITUDE DISTRIBUTION OF THE OUTPUT SIGNAL.
C
       CALL SETA (A,PO,IOP,LIM1,LIMOUT)
C
C SUBROUTINE "SIMQ" SOLVES THE SIMULTANEOUS EQUATIONS.
C ARRAY "A" ENTERS THE SQUARE MATRIX OF VARIABLE COEFFICIENTS.
C ARRAY "PO" ENTERS THE COLUMN VECTOR OF CONSTANTS
C AND RETURNS THE SOLUTIONS OF THE EQUATIONS.
C
       CALL SIMQ  (A,PO,LIMOUT,KS)
       IF (KS) 5,4,5
C
C EQUATIONS HAVE BEEN SOLVED.
C SUBROUTINE "VAR" NOW CALCULATES THE VARIANCE OF THE NOISE
C GENERATED AT B1.
C
    4  CALL VAR (IB1,NORM,B1,PO,LIMOUT,VARB1,MTYPE)
C
C "VAREQ" IS THE EQUIVALENT INPUT NOISE VARIANCE, WHICH IS THEN
C AMPLIFIED BY THE POLE-SECTION GAIN TO GIVE THE
C PREDICTED OUTPUT NOISE VARIANCE.
C
       VAREQ=VARA0+VARB1
       VAREQ=VAREQ/(1.000-B1*B1)
       WRITE (6,202) VAREQ
  202  FORMAT (1H0,25X,35H PREDICTED OUTPUT NOISE VARIANCE = ,D15.8)
       GO TO 6
C
C SIMULTANEOUS EQUATIONS CANNOT BE SOLVED.
C
    5  WRITE (6,203)
  203  FORMAT (1H0,25X,40H SIMULTANEOUS EQUATIONS CANNOT BE SOLVED)
    6  STOP
       END
```

SETA

```
      SUBROUTINE SETA (A,B,IB1,LIM1,N)
C
C THIS SUBROUTINE SETS UP THE SIMULTANEOUS EQUATIONS.
C "N" IS THE NUMBER OF VARIABLES.
C "A" IS CONSIDERED TO BE AN N*N SQUARE MATRIX IN WHICH
C ROWS VARY MOST RAPIDLY AND COLUMNS MOST SLOWLY.
C "B" IS A COLUMN VECTOR OF LENGTH N.
C "B" INITIALLY HOLDS THE SIGNAL AMPLITUDE DISTRIBUTION FUNCTION
C AT THE FILTER LOOP INPUT.
C "LIM1" IS THE NUMBER OF DIFFERENT LEVELS AT THE LOOP INPUT.
C START WITH EQUATION SUMMING PROBABILITIES TO UNITY.
C
      DIMENSION IB1(128),A(16384),B(128)
      ND=N*N
      DO 2 I=1,ND
      A(I)=0.0E0
    2 CONTINUE
      J=N+1
      DO 1 I=2,N
      A(J)=2.0E0
      J=J+N
    1 CONTINUE
C
C NOW CONSIDER OTHER EQUATIONS.
C CONSIDER ALL POSSIBLE INPUTS TO MULTIPLIER IB1 IN TURN.
C SET UP MATRIX COLUMNS IN TURN.
C
      IJ=1
      DO 3 K=1,N
      M=IB1(K)
      IT=(K-1)*N
      IF (M) 4,5,4
C
C M IS NOT ZERO.
C FIRSTLY CONSIDER ZERO INPUT.
C
    4 J=IT+M+1
      A(J)=A(J)-B(1)
C
C NOW CONSIDER ALL OTHER INPUTS.
C
      DO 6 L=2,LIM1
C
C CONSIDER DIFFERENCE SIGNAL.
C
      L1=L-1
      ND=IABS(L1-M)
      IF (ND) 9,8,9
    9 J=IT+ND+1
      A(J)=A(J)-B(L)
C
C CONSIDER SUM SIGNAL.
C
    8 J=IT+M+L
      A(J)=A(J)-B(L)
    6 CONTINUE
      GO TO 10
C
C M IS ZERO.
C
    5 DO 7 L=2,LIM1
      J=IT+L
      A(J)=A(J)-B(L)
    7 CONTINUE
C
C ADD 1 TO EACH ELEMENT OF MATRIX DIAGONAL.
C
   10 A(IJ)=A(IJ)+1.0E0
      IJ=IJ+N+1
    3 CONTINUE
C
C FINALLY SET B COLUMN VECTOR.
C B(1) = 1, ALL OTHERS = 0.
C
      B(1)=1.0E0
      DO 11 I=2,N
      B(I)=0.0E0
   11 CONTINUE
      RETURN
      END
```

SIMQ

```
      SUBROUTINE SIMQ (A,B,N,KS)
C
C SOLVES N SIMULTANEOUS EQUATIONS.
C "A" IS CONSIDERED TO BE AN N*N SQUARE MATRIX HOLDING
C VARIABLE COEFFICIENTS.
C "B" IS A COLUMN VECTOR OF LENGTH N HOLDING THE EQUATION CONSTANTS.
C "B" RETURNS THE SOLUTIONS.
C THE MAXIMUM VALUE OF N IS 128.
C
      DIMENSION A(16384),B(128)
C
C FORWARD SOLUTION.
C CONSIDER ALL VARIABLES (MATRIX COLUMNS) IN TURN.
C
      KS=0
      JJ=-N
      DO 65 J=1,N
      JY=J+1
C
C JJ POINTS TO MATRIX DIAGONAL.
C
      JJ=JJ+N+1
      BIGA=0.0E0
      IT=JJ-J
C
C SEARCH ROWS FOR MAXIMUM COEFFICIENT IN COLUMN.
C
      DO 30 I=J,N
      IJ=IT+I
      IF (ABS(BIGA)-ABS(A(IJ))) 20,30,30
   20 BIGA=A(IJ)
      IMAX=I
   30 CONTINUE
C
C "BIGA" IS THE PIVOT. TEST FOR BIGA = 0.
C IF SO SET KS = 1 AND RETURN TO CALLING PROGRAM AS
C MATRIX HAS A ZERO COLUMN.
C
      IF (BIGA) 40,35,40
   35 KS=1
      RETURN
C
C SWAP ROW WITH PIVOT TO LEADING POSITION.
C
   40 I1=J+N*(J-2)
      IT=IMAX-J
C
C SWAP ROW PAIR COLUMN BY COLUMN.
C
      DO 50 K=J,N
      I1=I1+N
      I2=I1+IT
      SAVE=A(I2)
      A(I2)=A(I1)
C
C DIVIDE VARIABLE OF LEADING ROW BY PIVOT.
C RESULTING PIVOT IS UNITY.
C
      A(I1)=SAVE/BIGA
   50 CONTINUE
C
C ROWS IN COLUMN VECTOR B ARE ALSO SWAPPED.
C CONSTANT IN LEADING ROW DIVIDED BY BIGA.
C
      SAVE=B(IMAX)
      B(IMAX)=B(J)
      B(J)=SAVE/BIGA
C
C ELIMINATE VARIABLE J.
C IF J = N GO TO BACK SOLUTION.
C
      IF (J-N) 55,70,55
   55 IQS=N*(J-1)
```

SIMQ (cont.)

```
C
C CALCULATE NEW MATRIX.
C CONSIDER APPROPRIATE ROWS OF PARTICULAR COLUMN.
C
      DO 64 IX=JY,N
      IXJ=IQS+IX
      IT=J-IX
C
C GO THROUGH APPROPRIATE COLUMNS.
C
      DO 60 JX=JY,N
      IXJX=N*(JX-1)+IX
      JJX=IXJX+IT
C
C NEW MATRIX ELEMENT FORMED.
C
      A(IXJX)=A(IXJX)-A(IXJ)*A(JJX)
   60 CONTINUE
C
C NEW ELEMENT IN COLUMN VECTOR FORMED.
C
      B(IX)=B(IX)-B(J)*A(IXJ)
   64 CONTINUE
   65 CONTINUE
C
C BACK SOLUTION.
C STARTS WITH VARIABLE (N-1) AND WORKS BACK TO VARIABLE 1.
C
   70 NY=N-1
      IT=N*N
C
C WORK UP THE ROWS FROM (N-1) TO 1.
C
      DO 80 J=1,NY
      IA=IT-J
      IB=N-J
      IC=N
C
C WORK FROM NTH. COLUMN TO MATRIX DIAGONAL.
C MATRIX HAS UNIT DIAGONAL.
C B(IB) WILL RETURN SOLUTION TO VARIABLE IB.
C
      DO 90 K=1,J
      B(IB)=B(IB)-A(IA)*B(IC)
      IA=IA-N
      IC=IC-1
   90 CONTINUE
   80 CONTINUE
      RETURN
      END
```

MULAR

```
      SUBROUTINE MULAR (MUL,MAR,LIM,NORM,MTYPE)
C
C FILLS ARRAY "MAR" WITH PRACTICAL RESULTS OF
C MULTIPLYING BY (MUL/NORM).
C "LIM" IS THE LENGTH OF ARRAY MAR.
C "MTYPE" DEFINES THE ROUNDOFF PROCESS.
C
      DIMENSION MAR(128)
      DO 1 I=1,LIM
      J=I-1
      CALL PMUL (MUL,J,K,NORM,MTYPE)
      MAR(I)=K
    1 CONTINUE
      RETURN
      END
```

ZCMOD

```
C ZCMOD - MAIN PROGRAM.
C NOISE VARIANCE PREDICTION BY SOLUTION OF SIMULTANEOUS EQUATIONS.
C MODIFIES THEORETICAL MULTIPLIERS TO GIVE UNCORRELATED ERROR SEQUENCES.
C FIRST ORDER FILTER.
C DIRECT FORM.
C FIXED POINT ARITHMETIC.
C ASSUMES UNIFORM INPUT AMPLITUDE DISTRIBUTION.
C ASSUMES SYMMETRICAL INPUT SIGNAL.
C APPROPRIATE FOR SYMMETRICAL ERROR PROCESSES.
C SOLVES LINEAR SIMULTANEOUS EQUATIONS TO FIND AMPLITUDE DISTRIBUTION OF OUTPUT
C FILTER COEFFICIENTS AND SIGNAL WORDLENGTH REQUIRED AS DATA.
C ALSO REQUIRES DATA ON TYPE OF ROUNDOFF TO BE USED.
C "MTYPE" EQUAL TO ZERO INDICATES ROUNDING.
C "MTYPE" POSITIVE INDICATES SIGN MAGNITUDE TRUNCATION.
C
      DIMENSION IOP(128),PO(128),A(16384)
      REAL*8 ANORM,A0,B1,VARA0,VARB1,VAREQ,EA0,EB1
      WRITE (6,400)
  400 FORMAT (1H1,20X,26H NOISE VARIANCE PREDICTION)
      WRITE (6,404)
  404 FORMAT (1H0,20X,38H BY SOLUTION OF SIMULTANEOUS EQUATIONS)
      WRITE (6,401)
  401 FORMAT (1H0,20X,40H FIRST ORDER, DIRECT, FIXED POINT FILTER)
C
C READ INTEGER FILTER COEFFICIENTS AND NORMALISING FACTOR (A POWER OF 2).
C "IA0" MUST BE POSITIVE.
C SUM OF MAGNITUDES OF "IA0" AND "IB1" MUST BE LESS THAN "NORM".
C
      READ (5,100) IA0,IB1,NORM
  100 FORMAT (3I10)
C
C READ SIGNAL DATA WORDLENGTH,
C A POSITIVE INTEGER LESS THAN OR EQUAL TO 8.
C
      READ (5,101) LENGTH
  101 FORMAT (I10)
C
C READ VALUE OF "MTYPE" TO DEFINE ROUNDOFF PROCESS.
C
      READ (5,101) MTYPE
      IF (MTYPE) 9,10,9
    9 WRITE (6,402)
  402 FORMAT (1H0,20X,26H SIGN MAGNITUDE TRUNCATION)
      GO TO 11
   10 WRITE (6,403)
  403 FORMAT (1H0,20X,9H ROUNDING)
   11 WRITE (6,300) IA0,NORM
  300 FORMAT (1H0,20X,6H A0 = ,I10,3H / ,I10)
      WRITE (6,301) IB1,NORM
  301 FORMAT (1H0,20X,6H B1 = ,I10,3H / ,I10)
      WRITE (6,302) LENGTH
  302 FORMAT (1H0,20X,26H SIGNAL DATA WORDLENGTH = ,I2)
C
C CONVERT COEFFICIENTS TO FLOATING POINT.
C
      ANORM=DFLOAT(NORM)
      A0=DFLOAT(IA0)/ANORM
      B1=DFLOAT(IB1)/ANORM
C
C FIND MAXIMUM SIGNAL AMPLITUDE, "MAX", ALLOWED BY WORDLENGTH.
C CALL "LIMS1" WHICH DETERMINES THE MAXIMUM ALLOWED INPUT AMPLITUDE
C TO THE FILTER, AND THE CORRESPONDING OUTPUT AMPLITUDE.
C
      MAX=2**(LENGTH-1)-1
      CALL LIMS1 (MAX,IA0,IB1,NORM,INAMP,LIMOUT,MTYPE,LIM1,LIM2)
      WRITE (6,303) INAMP
  303 FORMAT (1H0,25X,20H INPUT AMPLITUDE  = ,I4)
      WRITE (6,304) LIMOUT
  304 FORMAT (1H0,25X,20H OUTPUT AMPLITUDE = ,I4)
      IF (LIM1) 60,70,60
   70 WRITE (6,410)
  410 FORMAT (1H0,25X,23H A0 IS EFFECTIVELY ZERO)
      STOP
   60 IF (LIM2) 65,75,65
   75 WRITE (6,420)
  420 FORMAT (1H0,25X,23H B1 IS EFFECTIVELY ZERO)
      STOP
```

```
C
C CALCULATE "CONST", THE PROBABILITY OF A GIVEN SIGNAL AT THE
C FILTER INPUT. FILL ARRAY "PO" WITH THIS VALUE.
C CALL "ZCVAR" WHICH CALCULATES THE REQUIRED EFFECTIVE VALUE OF THE
C MULTIPLIER TO GIVE AN ERROR SEQUENCE WHICH HAS A ZERO
C CORRELATION FACTOR WITH THE MULTIPLIER INPUT SIGNAL.
C "ZCVAR" THEN CALCULATES THE VARIANCE OF THE ERROR SEQUENCE.
C
   65  CONST=1.0E0/FLOAT(2*INAMP+1)
       INAMP=INAMP+1
       DO 1 I=1,INAMP
       PO(I)=CONST
    1  CONTINUE
       CALL ZCVAR(IAO,NORM,AO,PO,INAMP,EAO,VARAO,MTYPE)
       WRITE (6,600) EAO
  600  FORMAT (1HO,25X,26H EFFECTIVE VALUE OF AO IS ,D15.8)
C
C CALL "MULAR" WHICH FILLS ARRAY "IOP" WITH THE PRACTICAL VALUES
C OF MULTIPLYING THE INPUT SIGNAL LEVELS BY (IAO/NORM).
C
       CALL MULAR (IAO,IOP,INAMP,NORM,MTYPE)
C
C ARRAY "PO" IS NOW SET TO THE AMPLITUDE PROBABILITY FUNCTION
C OF THE SIGNAL AT THE OUTPUT OF THE MULTIPLIER AO.
C IT IS INITIALISED TO ZERO.
C
       LIM1=LIM1+1
       DO 2 I=1,LIM1
       PO(I)=0.0E0
    2  CONTINUE
       PO(1)=CONST
       DO 3 I=2,INAMP
       JK=IOP(I)
       IF (JK) 50,51,50
C
C NON-ZERO SIGNAL AT OUTPUT OF AO.
C
   50  JK=JK+1
       PO(JK)=PO(JK)+CONST
       GO TO 3
C
C ZERO SIGNAL AT OUTPUT OF AO.
C
   51  PO(1)=PO(1)+2.0E0*CONST
    3  CONTINUE
C
C CALL "MULAR" WHICH FILLS ARRAY "IOP" WITH THE PRACTICAL RESULTS
C OF MULTIPLYING THE OUTPUT SIGNAL LEVELS BY (IB1/NORM).
C
       LIMOUT=LIMOUT+1
       CALL MULAR (IB1,IOP,LIMOUT,NORM,MTYPE)
C
C SUBROUTINE "SETA" SETS UP THE MATRIX "A" WITH THE COEFFICIENTS
C WHICH DEFINE THE SIMULTANEOUS EQUATIONS WHICH GOVERN THE
C AMPLITUDE DISTRIBUTION OF THE OUTPUT SIGNAL.
C
       CALL SETA (A,PO,IOP,LIM1,LIMOUT)
C
C SUBROUTINE "SIMQ" SOLVES THE SIMULTANEOUS EQUATIONS.
C ARRAY "A" ENTERS THE SQUARE MATRIX OF VARIABLE COEFFICIENTS.
C ARRAY "PO" ENTERS THE COLUMN VECTOR OF CONSTANTS
C AND RETURNS THE SOLUTIONS OF THE EQUATIONS.
C
       CALL SIMQ  (A,PO,LIMOUT,KS)
       IF (KS) 5,4,5
C
C EQUATIONS HAVE BEEN SOLVED.
C SUBROUTINE "ZCVAR" NOW CALCULATES THE EFFECTIVE VALUE OF B1, "EB1",
C AND THE VARIANCE OF THE ZERO CORRELATED ERROR SEQUENCE.
C
    4  CALL ZCVAR(IB1,NORM,B1,PO,LIMOUT,EB1,VARB1,MTYPE)
       WRITE (6,601) EB1
  601  FORMAT (1HO,25X,26H EFFECTIVE VALUE OF B1 IS ,D15.8)
```

ZCMOD (cont.)

```
C
C "VAREQ" IS THE EQUIVALENT INPUT NOISE VARIANCE, WHICH IS THEN
C AMPLIFIED BY THE POLE-SECTION GAIN TO GIVE THE
C PREDICTED OUTPUT NOISE VARIANCE.
C THE EFFECTIVE VALUE OF B1, "EB1", IS USED.
C
      VAREQ=VARAO+VARB1
      VAREQ=VAREQ/(1.000-CB1*EB1)
      WRITE (6,202) VAREQ
  202 FORMAT (1H0,25X,35H PREDICTED OUTPUT NOISE VARIANCE = ,D15.8)
      GO TO 6
C
C SIMULTANEOUS EQUATIONS CANNOT BE SOLVED.
C
    5 WRITE (6,203)
  203 FORMAT (1H0,25X,40H SIMULTANEOUS EQUATIONS CANNOT BE SOLVED)
    6 STOP
      END
```

ZCVAR

```
      SUBROUTINE ZCVAR(MUL,NORM,AMUL,P,LIM,EFMUL,VAR,MTYPE)
C
C CALCULATES THE REQUIRED THEORETICAL MULTIPLIER VALUE TO GIVE AN ERROR
C SEQUENCE WHICH HAS A ZERO CORRELATION FACTOR WITH THE INPUT
C SEQUENCE TO THE MULTIPLIER.
C THEN CALCULATES THE VARIANCE OF THE MODIFIED ERROR SEQUENCE.
C ARRAY "P" HOLDS THE AMPLITUDE DISTRIBUTION FUNCTION FOR THE
C SIGNAL AT THE MULTIPLIER INPUT.
C "LIM" IS THE NUMBER OF SIGNIFICANT ELEMENTS IN P.
C
      DIMENSION P(128)
      REAL*8 AMUL,VAR,EFMUL,AI,A,C,R,DELTA,ER(128)
C
C FIRSTLY COMPUTE "EFMUL" THE EFFECTIVE VALUE OF "AMUL" TO GIVE
C AN UNCORRELATED ERROR SEQUENCE.
C
      C=0.000
      R=0.000
      DO 1 I=2,LIM
      I1=I-1
      AI=DFLOAT(I1)
      C=C+DFLOAT(I1*I1)*P(I)
      CALL PMUL (MUL,I1,IY,NORM,MTYPE)
      A=AI*AMUL
      ER(I)=DFLOAT(IY)-A
      R=R+AI*ER(I)*P(I)
    1 CONTINUE
      DELTA=R/C
      EFMUL=AMUL+DELTA
C
C NOW COMPUTE THE VARIANCE OF THE MODIFIED ERROR SEQUENCE.
C
      VAR=0.000
      DO 2 I=2,LIM
      I1=I-1
      AI=DFLOAT(I1)
C
C MODIFY THE ERROR VALUE.
C
      R=ER(I)
      R=R-AI*DELTA
      VAR=VAR+R*R*P(I)
    2 CONTINUE
      VAR=VAR*2.000
      RETURN
      END
```

REFERENCES.

1. Bergland,G.D., 'A guided tour of the fast Fourier transform', IEEE Spectrum 6 pp 41-52, July 1969.
2. Rader,C.M., et al., 'On digital filtering', IEEE Trans. Audio & Electroac. AU-16 No.3 pp 303-14, September 1968.
3. Steiglitz,K., 'The equivalence of digital and analog signal processing', Information & Control 8 pp 455-67, October 1965.
4. Ruston,H. & Bordogna,J., 'Electric networks: functions, filters, analysis', McGraw-Hill 1966.
5. James,H.M.,Nichols,N.B. & Philips,R.S., 'Theory of servomechanisms', McGraw-Hill 1947.
6. Rader,C.M. & Gold,B., 'Digital filter design techniques in the frequency domain', Proc. IEEE 55 pp 149-71, February 1967.
7. Tustin,A., 'A method of analyzing the behavior of linear systems in terms of time series', Jour. IEE 94 pp 130-42, May 1947.
8. Kaiser,J.F., 'Digital filters', in 'Systems analysis by digital computer', eds., F.F.Kuo & J.F.Kaiser, New York, Wiley 1966.
9. Golden,R.M. & Kaiser,J.F., 'Design of wideband sampled-data filters', Bell Systems Tech. J. 43 pp 1533-46, July 1964.
10. Gold,B. & Rader,C.M., 'Digital processing of signals', McGraw-Hill 1969.
11. Kaiser,J.F., 'Design methods for sampled-data filters', Proc. 1st Allerton conf. on Circuit & System Theory pp 221-36, 1963.
12. Helms,H.D., 'Nonrecursive digital filters: design methods for achieving specifications on frequency response', IEEE Trans. Audio & Electroac. AU-16 pp 336-42, September 1968.
13. Jackson,L.B., 'Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form', IEEE Trans. Audio & Electroac. AU-18 pp 107-22, June 1970.

14. Jackson,L.B.,Kaiser,J.F. & McDonald,H.S., 'An approach to the implementation of digital filters', IEEE Trans. Audio & Electroac. AU-16 pp 413-21, September 1968.

15. Rabiner,L.R. et al., 'Terminology in digital signal processing', IEEE Trans. Audio & Electroac. AU-20 pp 322-37, December 1972.

16. Oppenheim,A.V., 'Realization of digital filters using block-floating-point arithmetic', IEEE Trans. Audio & Electroac. AU-18 pp 130-6, June 1970.

17. Agarwal,R.C. & Burrus,C.S., 'Number theoretic transforms to implement fast digital convolution', Proc. IEEE 63 pp 550-60, April 1975.

18. Burrus,C.S., 'Block realization of digital filters', IEEE Trans. Audio & Electroac. AU-20 pp 230-5, October 1972.

19. Agarwal,R.C. & Burrus,C.S., 'Fast convolution using Fermat number transforms with applications to digital filtering', IEEE Trans. Acoustics, Speech & Signal Processing ASSP-22 pp 87-97, April 1974.

20. Rader,C.M., 'On the application of the number theoretic transforms of high speed convolution to two-dimensional filtering', IEEE Trans. Circuits & Systems CAS-22 p 575, June 1975.

21. McClellan,J.H., 'Hardware realization of a Fermat number transform', IEEE Trans. Acoustics, Speech & Signal Processing ASSP-24 pp 216-25, June 1976.

22. Bennett,W.R., 'Spectra of quantized signals', Bell Systems Tech. J. 27 pp 446-72, July 1948.

23. Widrow,B., 'A study of rough amplitude quantization by means of Nyquist sampling theory', IRE Trans. Circuit Theory CT-3 No.4 pp 266-76, 1956.

24. Widrow,B., 'Statistical analysis of amplitude-quantized sampled-data systems', AIEE Trans. Applications & Industry 79 pp 555-68, January 1961.

25. Kosyakin,A.A., 'The statistical theory of amplitude quantization', Avtomat. i Telemekh. 22 No.6 pp 722-9, 1961.

26. Katzenelson,J., 'On errors introduced by combined sampling and quantization', IRE Trans. Automat. Contr. AC-7 pp 58-68, April 1962.

27. Watts,D.G. & Katzenelson,J., 'Discussion of "On errors introduced by combined sampling and quantization"', IEEE Trans. Automat. Contr. AC-8 pp 187-8, April 1963.

28. Watts,D.G., 'A general theory of amplitude quantisation with applications to correlation determination', IEE Proc. 109C pp 209-18, 1962.

29. Knowles,J.B. & Edwards,R., 'Effect of a finite-word-length computer in a sampled-data feedback system', IEE Proc. 112 pp 1197-207, June 1965.

30. Gold,B. & Rader,C.M., 'Effects of quantization noise in digital filters', AFIPS Conf. Proc. 28 pp 213-9, 1966.

31. Kaiser,J.F., 'Some practical considerations in the realization of linear digital filters', Proc. 3rd Allerton conf. on Circuit & System Theory pp 621-33, October 1965.

32. Rader,C.M. & Gold,B., 'Effects of parameter quantization on the poles of a digital filter', IEEE Proc. 55 pp 688-9, May 1967.

33. Knowles,J.B. & Olcayto,E.M., 'Coefficient accuracy and digital filter response', IEEE Trans. Circuit Theory CT-15 pp 31-41, March 1968.

34. Mantey,P.E., 'Eigenvalue sensitivity and state-variable selection', IEEE Trans. Automat Contr. AC-13 pp 263-9, June 1968.

35. Weaver,C.S. et al., 'Digital filtering with applications to electrocardiogram processing', IEEE Trans. Audio & Electroac. AU-16 pp 350-91, September 1968.

36. Golden,R.M. & White,S.A., 'A holding technique to reduce number of bits in digital transfer functions', IEEE Trans. Audio & Electroac. AU-16 pp 433-6, September 1968.

37. Rabiner,L.R.,Gold,B. & McGonegal,C.A., 'An approach to the approximation problem for non-recursive digital filters', IEEE Trans. Audio & Electroac. AU-18 pp 83-106, June 1970.

38. Gardener,W.A., 'Reduction of sensitivity in sampled-data filters', IEEE Trans. Circuit Theory CT-17 pp 660-3, November 1970.

39. Herrman,O. & Schüssler,W., 'On the accuracy problem in the design of non-recursive digital filters', Arch. Elek. Übertragung 24 pp 525-6, November 1970.

40. Avenhaus,E. & Schüssler,W., 'On the approximation problem in the design of digital filters with limited wordlength', Arch. Elek. Übertragung 24 pp 571-2, December 1970.

41. Otnes,R.K. & McNamee,L.P., 'Instability thresholds in digital filters due to coefficient rounding', IEEE Trans. Audio & Electroac. AU-18 pp 456-63, December 1970.

42. Herrman,O., 'On the approximation problem in non-recursive digital filter design', IEEE Trans. Circuit Theory CT-18 pp 411-3, May 1971.

43. Avenhaus,E., 'On the design of digital filters with coefficients of limited wordlength', IEEE Trans. Audio & Electroac. AU-20 pp 206-12, August 1972.

44. Avenhaus,E., 'A proposal to find suitable canonical structures for the implementation of digital filters with small coefficient wordlength', Nachrichtentech. Z. (NTZ) (Germany) 25 pp 377-82, August 1972.

45. Hadjifotiou,A. & Appleby,D.G., 'Design of digital filters with severely quantised coefficients', Radio & Electronic Engineer 46 pp 23-8, January 1976.

46. Agarwal,R.C. & Burrus,C.S., 'New recursive digital filter structures having very low sensitivity and roundoff noise', IEEE Trans. Circuits & Systems CAS-22 pp 921-7, December 1975.

47. Edwards,R.,Bradley,J. & Knowles,J.B., 'Comparison of noise performance of programming methods in the realization of digital filters', Proc. Polytechnic Inst. Brooklyn Symp. Computer Processing in Communications, pp 295-311, April 1969.

48. Weinstein,C. & Oppenheim,A.V., 'A comparison of roundoff noise in floating point and fixed point digital filter realizations', IEEE Proc. 57 pp 1181-3, June 1969.

49. Jackson,L.B., 'On the interaction of roundoff noise and dynamic range in digital filters', Bell Systems Tech. J. 49 pp 159-84, February 1970.

50. Liu,B., 'Effect of finite word length on the accuracy of digital filters - a review', IEEE Trans. Circuit Theory CT-18 pp 670-7, November 1971.

51. Oppenheim,A.V. & Weinstein,C.J., 'Effects of finite register length in digital filtering and the fast Fourier transform', IEEE Proc. 60 pp 957-76, August 1972.

52. Liu,B. & Van Valkenburg,M.E., 'On roundoff error of fixed-point digital filters using sign-magnitude truncation', IEEE Trans. Circuit Theory CT-19 pp 536-7, September 1972.

53. Fettweis,A., 'On the connection between multiplier word length limitation and roundoff noise in digital filters', IEEE Trans. Circuit Theory CT-19 pp 486-91, September 1972.

54. Fettweis,A., 'Roundoff noise and attenuation sensitivity in digital filters with fixed point arithmetic', IEEE Trans. Circuit Theory CT-20 pp 174-5, March 1973.

55. Chan,D.S.K. & Rabiner,L.R., 'An algorithm for minimizing roundoff noise in cascade realizations of finite impulse response digital filters', Bell Systems Tech. J. 52 pp 347-85, March 1973.

56. Chan.D.S.K. & Rabiner,L.R., 'Theory of roundoff noise in cascade realizations of finite impulse response digital filters', Bell Systems Tech. J. 52 pp 329-45, March 1973.

57. Lee,W.S., 'Optimization of digital filters for low roundoff noise', IEEE Trans. Circuits & Systems CAS-21 pp 424-31, May 1974.

58. Claasen,T.A.C.M.,Mecklenbräukler,W.F.G. & Peek,J.B.H., 'Quantisation noise analysis for fixed point digital filters using magnitude truncation for quantisation', IEEE Trans. Circuits & Systems CAS-22 pp 887-95, November 1975.

59. Aström,K.J.,Jury,E.I. & Agniel,R.G., 'A numerical method for the evaluation of complex integrals', IEEE Trans. Automat. Contr. AC-13 pp 468-71, August 1970.

60. Peled,A. & Liu,B., 'A new hardware realization of digital filters', IEEE Trans. Acoustics, Speech & Signal Processing ASSP-22 pp 456-63, December 1974.

61. Liu,B. & Peled,A., 'A new hardware realization of high-speed fast Fourier transformers', IEEE Trans. Acoustics, Speech & Signal Processing ASSP-23 pp 543-7, December 1975.

62. Peled,A.,Liu,B. & Steiglitz,K., 'A note on implementation of digital filters', IEEE Trans. Acoustics, Speech & Signal Processing ASSP-23 pp 387-9, August 1975.

63. Bartlett,M.S., 'Periodogram analysis and continuous spectra', Biometrika 37 pp 1-16, 1950.

64. Blackman,R.B. & Tukey,J.W., 'The measurement of power spectra', New York, Dover, 1958.

65. Welch,P.D., 'The use of fast Fourier transform for the estimation of power spectra', IEEE Trans. Audio & Electroac. AU-15 pp 70-3, June 1967.

66. Bingham,C.,Godfrey,M.D. & Tukey,J.W., 'Modern techniques of power spectrum estimation', IEEE Trans. Audio & Electroac. AU-15 pp 56-66, June 1967.

67. Parzen,E., 'Mathematical considerations in the estimation of spectra', Technometrics 3 pp 167-90, May 1961.

68. Richards,P.I., 'Computing reliable power spectra', IEEE Spectrum 4 pp 83-90, January 1967.

69. Dolph,C.L., 'A current distribution for broadside arrays which optimizes the relacionship between beam width and side-lobe level', Proc. IRE Waves & Electrons 35 pp 335-48, June 1946.

70. Cooley,J.W. & Tukey,J.W., 'An algorithm for the machine calculation of complex Fourier series', Mathematics of Computation 19 pp 297-301, 1965.

71. Stockham,T.G., 'High speed convolution and correlation', Spring Joint Computer conf., AFIPS Conf. Proc. 28 pp 229-33, 1966.

72. Rader,C.M., 'An improved algorithm for high speed autocorrelation with applications to spectral estimation', IEEE Trans. Audio & Electroac. AU-18 pp 439-41, December 1970.

73. Cochran,W.T. et al., 'What is the fast Fourier transform?', IEEE Trans. Audio & Electroac. AU-15 pp 45-55, June 1967.

74. Cooley,J.W.,Lewis,P.A.W. & Welch,P.D., 'The finite Fourier transform', IEEE Trans. Audio & Electroac. AU-17 pp 77-85, June 1969.

75. Rader,C.M., 'Discrete transforms via Mersenne transforms', IEEE Trans. Computing C-21 pp 1269-73, December 1972.

76. MacLaren,M.D. & Marsaglia,G., 'Uniform random number generators', J. Assoc. Computing Machinery (JACM) 12 pp 83-9, January 1965.

77. Parker,S.R. & Girard,P.E., 'Correlated noise due to roundoff in fixed point digital filters', IEEE Trans. Circuits & Systems CAS-23 pp 204-11, April 1976.