FINITE STATE MACHINE REPRESENTATION

of

DIGITAL SIGNAL PROCESSING SYSTEMS

by

Lawrence John Anthony Albinson

A dissertation submitted to the University of Cambridge for the degree of Doctor of Philosophy

September 1981

Acknowledgements

I would like to thank my supervisor, Dr. Peter Rayner, for his guidance and encouragement during the course of this research. I would also like to thank the Science and Engineering Research Council for providing me with financial support. In addition, my thanks are due to my colleagues and friends both in the engineering department and at Leckhampton for many hours of useful discussion.

Finally, I am endebted to Miss Karen Cripps and Miss Jane Wallpole for their excellent typing, to Mr. Michael Brown for his assistance with hardware and to Mr. Les Peters for his photography.

Declaration

I certify that this dissertation contains an account of my own work and not of work undertaken by or in collaboration with others. I further certify that it has not been submitted to any other university or for any other degree. This dissertation is 173 pages long.

Lawrence John Anthony Albinson
Corpus Christi College, Cambridge

L. Albinson.

25th. September, 1981

Summary

A new method for implementing digital filters is discussed. The method maximises the output signal to noise ratio of a filter by assigning at each of the filter variables an optimal quantization law. A filter optimised for a gaussian process is considered in detail. An error model is developed and applied to first and second order canonic form filter sections. Comparisons are drawn between the gaussian optimised filter and the equivalent fixed point arithmetic filter. The performance of gaussian optimised filters under sinusoidal input signal conditions is considered; it is found that the gaussian optimised filter exhibits a lower approximation error than the equivalent fixed point arithmetic filter. It is shown that when high order filters are implemented as a cascade of second order sections - with if necessary one first order section - the section ordering has a very small effect on the overall signal to noise ratio performance. A similar result for the pairing of poles and zeroes is found. Bounds on the maximum limit cycle amplitude for first and second order all-pole sections are presented. It is shown that for a first order all-pole the maximum limit cycle amplitude is lower than would be expected in the equivalent fixed point arithmetic filter, whereas, for the second order all-pole the bound is twice as large. Examples of a low-pass, band-pass and wideband differentiating filter,designed using free quantization law techniques,are presented.

This new design method leads to a filter whose arithmetic operations can not be performed using fixed point arithmetic hardware. Instead, the filter must be represented as a finite state machine and then implemented using sequential logic circuit synthesis techniques. The logic complexity is found to depend - amongst other considerations - on the so called state (code) assignment. Some preliminary results on this problem are presented for the case of a next state function computed using the AND/EXCLUSIVE-OR (ring-sum) logic expansion. A review of the state assignment techniques in the literature is

included. A part of the state assignment problem - for the case of AND/EX-OR logic - requires the numerous and consequently rapid computation of the Reed-Muller Transformation. A hardware processor - designed as an add-on to a minicomputer - is described; speed comparisons are drawn with the equivalent software algorithm.

Contents

	Title page	i
	Acknowledgements	ii
	Declaration	ii
	Summary	iii
1	Introduction	
1.1	General Introduction	1
1.2	Signal Quantization	2
1.3	Coefficient Quantization	4
1.4	Overflow Oscillations	5
1.5	The Deadband Effect - Limit Cycle Oscillations	8
1.6	Data Sampling Rate	9
1.7	A New Method for Implementing Digital Filters	11
1.8	Layout of Dissertation	12
2	First Order Free Quantization Law Digital Filters	
2.1	Introduction	16
2.2	Optimal Quantizers	17
2.3	Optimum Step Size for a Uniform Law Quantizer used to Quantize a Gaussian Process	22
2.4	Signal and System Characteristics	25
2.5	First Order All-Pole - Design	27
2.6	First Order All-Pole - Error Model	28
2.7	Uniform Quantization Law First Order All-Pole Driven by a Gaussian Process - Error Model	31
2.8	Section Ordering - Two First Order All-Poles	39
2.9	First Order All-Zero - Design	40
2.10	First Order All-Zero - Error Model	41
2.11	First Order Pole-Zero Design	43
2.12	Stability in First Order Systems	45
2.13	First Order All-Pole - Frequency Response	47
3	Second Order Free Quantization Law Digital Filters	
3.1	Introduction	52
3.2	Second Order All-Pole - Design	52
3.3	Second Order All-Pole - Error Model	54
3.4	Second Order All-Zero - Design	59
3.5	Second Order All-Zero - Error Model	60
3.6	Second Order Pole-Zero (section)	62

3 continued

3.7 Section Ordering and Pole-Zero Pairing 64
3.8 Stability Considerations 67

4 The Implementation of Free Quantization Law Digital Filters

4.1 Introduction 72
4.2 The Representation of a Digital Filter as a Finite State Machine 72
4.3 The Implementation of a Finite State Machine as a Sequential Circuit 76
4.5 The Code Assignment Problem 81
4.6 Next State Polynomials Minimisation 95
4.7 Concluding Remarks 96

5 A Reed-Muller Transform Processor

5.1 Introduction 98
5.2 The Reed-Muller Transform 98
5.3 A Reed-Muller Transform Processor 101
5.4 A Software Reed-Muller Transform Algorithm 110

6 Selected Examples of Free Quantization Law Filter Designs

6.1 Introduction 112
6.2 Example 1 - Wide-Band Digital Differentiator 113
6.3 Example 2 - Sixth Order All-Pole Low-Pass Filter 120
6.4 Example 3 - Sixth Order Elliptic Band-Pass Filter 128

7 Conclusions and Suggestions for Further Work

7.1 Conclusions 136
7.2 Suggestions for Further Work 139

A Appendix A - Experimental Techniques

A.1 Signal to Noise Ratio Measurement 140
A.2 Complex Amplitude Measurement 143
A.3 Noise Process Generation 148
A.4 General Purpose Computer Simulation of a Free Quantization Law Digital Filter 149

B Appendix B - Cumulative Density Function of a Gaussian Process - Numerical Approximation

B 153

C Appendix C - Implementation of Free Quantization Law Digital Filters - A Special Case

C 154

References 158

Glossary of Terms 165

1. Introduction

1.1 General Introduction

Digital Signal Processing [1-3] is the term used to describe the processing of sampled versions of signals by computer program or by specially designed digital hardware. The signals may be intrinsically sampled - discrete in time - or they may be observations of a process that continually varies with time. Digital systems may be used to replace analogue systems giving improved parameter stability, eliminating production spreads and offering higher reliability. More importantly, digital systems make it possible to solve problems which were previously considered intractable; an example is the rapid estimation of power spectra using the Fast Fourier Transform. The catalogue of digital signal processing applications continues to widen as the speed and complexity of the assoicated microcircuits increases. An account of some of these applications areas is to be found in [4]. This dissertation is concerned with the implementation of one of the building blocks of signal processors, the infinite duration impulse response digital filter.

Analogue linear time invariant (LTI) systems are described in the time and frequency domains by linear constant coefficient (LCC) differential equations and Laplace Transforms (transfer functions) respectively; similarly, digital LTI systems are described in the time and frequency domains by LCC difference equations and z-transforms [9] respectively. A general N-pole, M-zero LTI digital filter is described by the LCC difference equation:

$$\sum_{i=0}^{M} a_i x_{n-i} = y_n + \sum_{i=1}^{N} b_i y_{n-i} \qquad (1.1.1)$$

and by the z-domain transfer function:

$$H(z^{-1}) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{\sum_{i=0}^{M-1} a_i z^{-i}}{1 + \sum_{i=1}^{N-1} b_i z^{-i}} \qquad (1.1.2)$$

where the a_i and b_i are constants, z is a complex variable, and x_{n-i}, y_{n-i} are the input and output respectively at time t=(n-i)T. When each of the coefficients b_i is zero the filter is said to have a finite duration impulse response, and is called an FIR filter. The FIR filter is so named because its output, in response to a digital impulse at its input, cannot be non-zero for more than M samples. By contrast, if at least one of the b_i in equation 1.1.1 is non-zero the filter, again in response to a digital impulse, may be capable of producing a non-zero output indefinitely. In this case the filter is said to have an infinite duration impulse response, and is described as an IIR filter.

When a filters' input signal x_n is expressable in closed form its response y_n can be found using the z-transform method; however, when x_n is a stochastic process [7] y_n must be computed by iterating the difference equation. In the second case either a program has to be written for a general purpose digital computer or a special piece of digital hardware has to be designed. Whether a software or a hardware implementation of a filter is chosen, its parameters, that is its variable x_n and y_n and its constants a_i and b_i, must be stored using finite word length binary numbers. The process of choosing a finite word length binary number to represent a signal sample value is called quantization; the error in representation is called quantization error. Parameter quantization has four effects on the performance of an IIR filter, these are discussed in sections 1.2 through 1.5.

1.2 Signal Quantization

Quantizing signal samples has the effect of introducing noise into the output of a filter, and hence imposing a lower bound on the signal

amplitude [29]. A quantized signal sample is usually modelled as an unquantized signal sample corrupted by the addition of a noise sample. The spectral properties of quantization noise have been studied by Bennett [27]. To simplify noise analysis it is usually assumed that quantization noise sources have white power spectra [30]; necessary and sufficient conditions for this to be true have been derived by Stripad and Snyder [32].

In a fixed point arithmetic implementation of a digital filter there are two sources of quantization noise, the input signal quantizer (analogue to digital converter) and the multipliers. Digital multipliers produce quantization noise for the following reason: suppose each register in a signal processor contains k bits, then the product of any pair of registers contains 2k bits; since this product is to be restored to a k bit register its least significant k bits must be discarded. Floating point filter implementations [35-40] produce quantization noise following an addition also. The noise performance of a filter implementation is analysed using the ideal (unlimited arithmetic precision) filter as a model; following each arithmetic operation which the implementation performs imprecisely a noise source is added in. It is usual to make the reasonable assumption that all such noise sources are uncorrelated with each other and with the input signal.

Although in a practical filter implementation quantization noise cannot be eliminated completely, its effects can be reduced to an acceptable level by the choice of a large enough word length. Another method of reducing quantization noise is to perform the summation in equation 1.1.1 to 2k bits precision and then to quantize the result to k bits. The desired output signal to noise ratio will ultimately determine the required word length.

1.3 Coefficient Quantization

Coefficient quantization causes perturbations in a filters' pole-zero pattern; in turn this causes an error in the filters' complex magnitude response relative to the design specification. The maximum permissible error will determine the coefficient word length to be used. The coefficient quantization effect has received attention in the literature (13-26).

The selection of a large enough coefficient word length might proceed as follows: let $M(\omega)$ and $M_Q(\omega)$ be the complex magnitude response of a filter with unquantized and quantized coefficients respectively, and define the response error as:

$$M_e = || \; | M(\omega) - M_Q(\omega) | \; ||_p \qquad (1.3.1)$$

where $|| \; ||_p$ denotes the L_p-norm. Increase the coefficient word length until M_e is smaller than the design tolerance. (Useful norms are p=2 and p=∞ corresponding respectively to the mean square and maximum value.) The disadvantages of this method are:

(1) it requires a lot of computation, and

(2) it does not account for the response error being more sensitive to errors in some coefficients than in others.

A statistical method for selecting coefficient word lengths has been proposed by Avenhaus (21) and modified by Crochiere (24). With this method the magnitude response error is expressed as a weighted sum of the coefficient errors the weights being the coefficient sensitivities:

$$S_{c_i} = \frac{\delta M_e}{\delta c_i} \qquad (1.3.2)$$

where c_i is one of the coefficients a_i or b_i.

As well as increasing word length two further methods have been proposed for reducing coefficient quantization effects. The first method

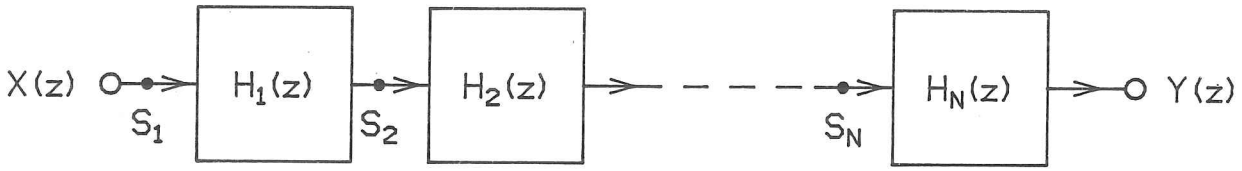
is called discrete optimisation; with this method an optimum set is selected from among the discrete space of coefficient values. The benefit of this method derives from the fact that rounding coefficient values does not necessarily yield the best magnitude approximation. The second method, cycled coefficient after McLeod [26] or dithered coefficient after Bolton [25], realises a desired coefficient value using a time varying quantized coefficient. With this method the time average of the magnitude response error is zero provided the time averages of the coefficient errors are all zero.

1.4 Overflow Oscillations

When the result of an arithmetic operation is outside the range of representable values overflow occurs. As an example consider the operation $c=a+b$ where $a > r$, $b > 2^k - r$ and hence $c > 2^k$. Variables a and b are within the range of k bit two's complement arithmetic but c is not. The effect of arithmetic overflow on an IIR filter is to cause its output to produce large amplitude overflow oscillations. This phenomenon has received considerable attention in the literature [41,49,53,56,58].

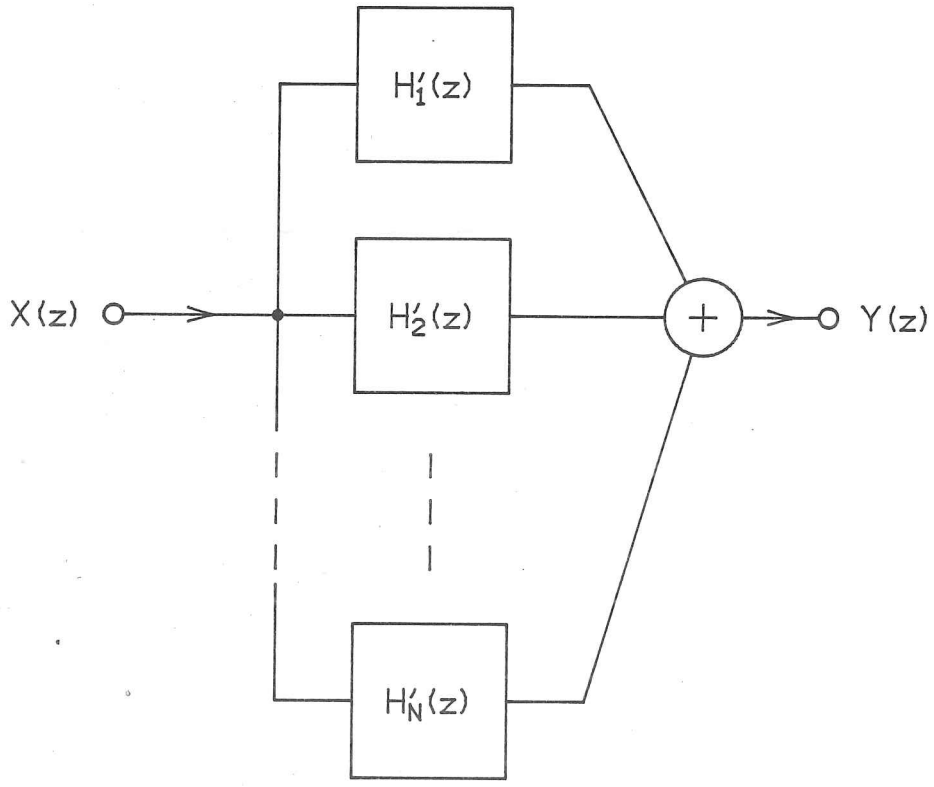
Arithmetic overflow can be detected and dealt with using some additional hardware; usually out of range results are forced to the largest magnitude representable value with the same sign. The effect of this is to replace overflow oscillations by signal clipping which is a less disturbing form of signal distortion. Overflow effects can be avoided by restricting the filters' input signal amplitude to a low enough value; as the number of bits of arithmetic precision is increased so too is the maximum allowable signal amplitude. It follows that the maximum achievable signal to noise ratio is determined by the word length used.

To ensure stability high order filters (order > 2) are implemented as a cascade or parallel connection of first or second order sub-filters as in figure 1.4.1 rather than in the direct form as in figure 1.4.2. The



$$Y(z)/X(z) = H(z) = S_1 \times H_1(z) \times S_2 \times H_2(z) \times \ldots \times S_N \times H_N(z)$$

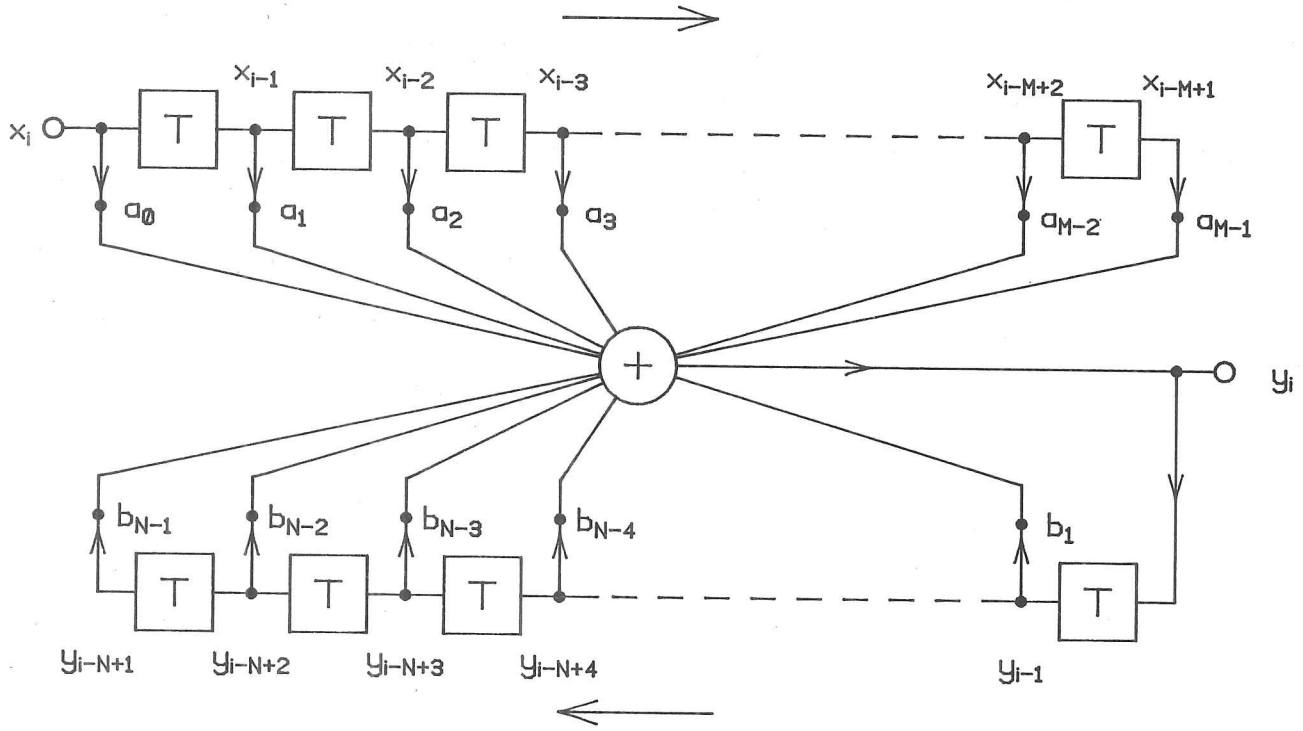
Cascade Implementation of Filter Transfer Function H(z)



$$Y(z)/X(z) = H(z) = H_1'(z) + H_2'(z) + \ldots + H_N'(z)$$

Parallel Implementation of Filter Transfer Function H(z)

Figure 1.4.1



Direct Form Implementation of an N-Pole, M-Zero Recursive Digital Filter

Figure 1.4.2

signal amplitude at the input to the first sub-filter is chosen to be consistent with no overflow effects in any of the sub-filters; particularly when some of the sub-filters have a high gain, this leads to certain filter variables occupying few of the quantization levels available to them; in turn this leads to the filter operating with a low signal to noise ratio. To circumvent this problem Jackson [29] proposes that scaling multipliers be introduced at the input to each of the sub-filters. The value of each scaling multiplier is chosen using a normed measure of its associated sub-filters' frequency response. Typically the L_2 or the L_∞ norms would be used.

1.5 The Deadband Effect - Limit Cycle Oscillations

When studying the effects of quantization noise it is assumed that a noise source is not correlated with the signal to which it adds; this is made possible by assuming that the signal, from sample to sample, makes an excursion of many quantization levels. If a filters' input signal is simple - for example a constant value - then this assumption is unreasonable and the analysis breaks down. In the steady state a filter would be expected to produce a constant output in response to a constant input, however, due to the deadband effect limit cycles, superimposed on the expected signal, are possible.

In a fixed point implementation limit cycles result when one or more of a filters' poles is temporarily placed on the z-plane unit circle; in turn this is caused by rounding arithmetic giving a coefficient an effective value of 1. For example, 0.95 x 4 rounded = 4.

The magnitude of limit cycles increases as a filters' poles approach the z-plane unit circle. The acceptable level of limit cycles may be thought of as placing an upper bound on the achievable 'Q' of a filter. Increasing word length reduces limit cycle amplitude relative to peak signal amplitude. As already mentioned equation 1.1.1 is never implemented

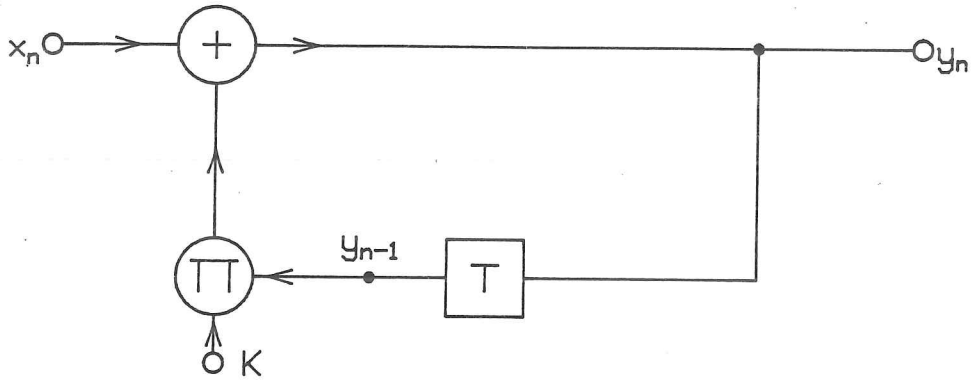
in the direct form for order greater than 2. This is because as the order of a filter section increases so too does the maximum limit cycle amplitude.

A method for analysing limit cycles was proposed by Jackson [12]. Further analysis of limit cycles is to be found in [43-52,51-57]. Methods for suppressing limit cycles are proposed in [49-54]. Bounds on the maximum amplitude of limit cycles have been derived by Sandberg and Kaiser [44], Long and Trick [46], Chang [50] and Rahman, Maria and Fahmy [51].

1.6 Data Sampling Rate

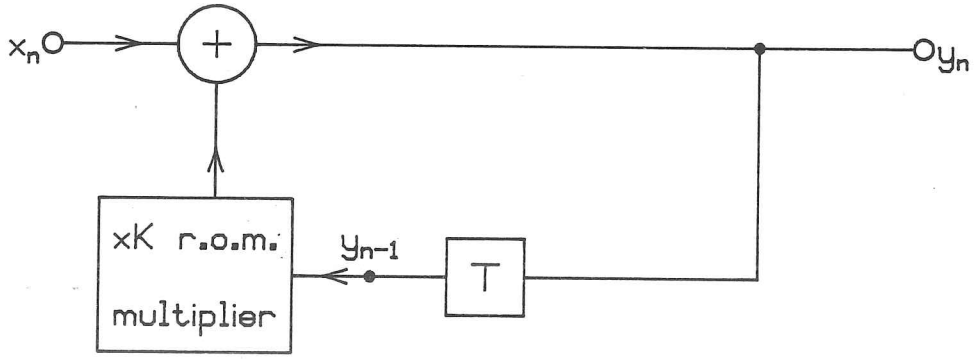
The parameter quantization effects discussed in sections 1.2 through 1.5 all have a detrimental influence on the performance of a filter. Although other cures have been proposed it is universally true that an increase in word length leads to an improved performance. For a given implementation increasing the word length reduces the maximum data sampling rate. It follows that if a filter is to be used at high data rates it is desirable to use just the right word length consistent with meeting the specification.

For a first order all-pole filter there are three basically different implementation structures, these are shown in figures 1.6.1a-c. When a constant coefficient filter is required the structure in figure 1.6.1a is redundant; this structure should, where possible, be avoided as its multipliers will usually have to be sequential devices; instead the structure in figure 1.6.1b, where multiplication is performed using a read only memory, should be used. The fastest implementation of the three is shown in figure 1.6.1c; here the code for y_n is a two level logical function of the codes for x_n and y_{n-1}. This method will be used as a basic model throughout this dissertation. With this method the shortest possible word length should be used so as to minimise



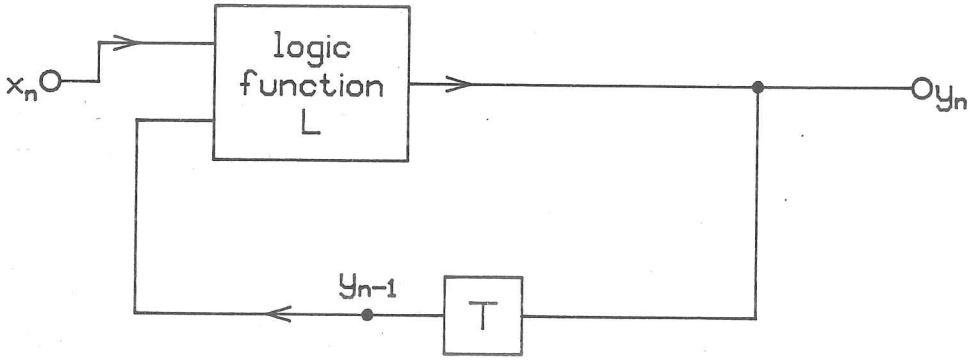
Variable Coefficient First Order All-Pole Filter Section

Figure 1.6.1a



Constant Coefficient First Order All-Pole Filter Section

Figure 1.6.1b



Finite State Machine Representation of a First Order Filter Section

Figure 1.6.1c

1.7 A New Method for Implementing Digital Filters

A fixed point arithmetic digital filter can be thought of as an ideal filter with a uniform law quantizer introduced at the input to each of the delay stores and at the input of the filter. The purpose of each of the quantizers is to reduce a continuous infinite valued variable to a discrete finite valued variable which is capable of being stored in a finite word length register. As this is the only fundamental purpose for each quantizer the associated quantization law need not necessarily be uniform; indeed, not all quantizers need employ the same quantization law. The new implementation method proposed obtains performance improvements, over the fixed point implementation, by judicious choice of the various quantization laws. In turn this leads to a reduced word length requirement for a given signal to noise ratio.

Each quantization law is chosen to best match the signal which it will quantize. The measure of best match can be defined in a variety of ways [66]; among these are minimum mean square error (m.m.s.e.), maximum entropy and constant fractional error. Much work is to be found in the literature on the subject of quantizer optimisation [59-67], the earliest results being presented by Max [59]. As signal to noise is defined as the ratio of powers the m.m.s.e. optimised quantizer will be used in this work. The maximum entropy quantizer is designed to have equiprobable quantization levels; the constant fractional error quantizer is designed to produce an error which in magnitude is no greater than a certain percentage of the absolute signal value.

In this work filters will be optimised for operation on gaussian input processes. This is because the shape of a gaussian process probability density function is invariant under a linear transformation and hence each filter variable employs the same quantizer but at a different variance level. Also, for high order filters the variables

towards the latter sub-filters will exhibit gaussian statistics regardless of the input statistics (central limit theorem). Without loss of generality the input process will always be assumed to be white. A coloured input process can be modelled using a white process and a spectrum colouring filter.

Filters which use optimal quantization laws can only be implemented using the hardware structure shown in figure 1.6.1c. This is the basic model of a finite state machine. An efficient implementation of a finite state machine depends on the following:

(1) the choice of a logic form for function L (eg. minterm, ring-sum or other),

(2) the choice of a low cost state (code) assignment, and

(3) the minimisation of the next state function (eg. Quine McCluskey reduction for minterm).

These issues have received attention in the literature; Relative complexity of different logic forms is considered in [68-72], State code assignment is considered in [73-89]. Logic minimisation for logic forms other than minterm is a relatively new subject; in particular, in some work to be published Rayner considers the minimisation of ring-sum forms. The implementation of a recursive digital filter as a finite state machine was first considered by Rayner [97]; he proposed the method as a means of increasing the data sampling rate of a fixed point arithmetic filter.

1.8 Layout of Dissertation

In chapter 2 the design of optimal memoryless quantizers is reviewed; the optimal quantizer for a gaussian process is obtained. The optimum step size for a uniform quantizer applied to a gaussian process is evaluated for different numbers of quantization levels; these values are needed in order to make a fair comparison between free and uniform quantization law filters. The remainder of the chapter is concerned with

the design and error analysis of a first order pole, zero and pole-zero (section); experimental results are presented in support of the theoretical model. An analysis of the effects of section ordering is presented. The chapter concludes with some results on the limit cycle behaviour and magnitude approximation error response of free quantization law filters.

Chapter 3 is concerned with the design and error analysis of a free quantization law second order pole-zero section. Different structures for a second order section are considered, the canonic form, figure 1.8.1, is shown to give the highest signal to noise ratio. Although the section ordering problem defies exact analysis, experimental results on its effects are presented. The pole-zero pairing problem is also treated. Finally, an analysis of the limit cycle behaviour of second order poles is given.

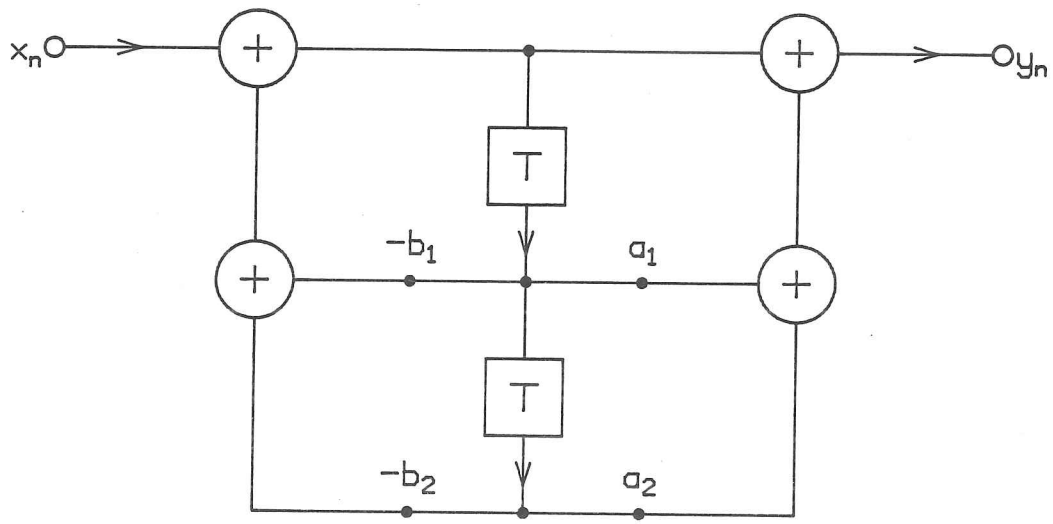
As already mentioned, the efficient implementation of free quantization law filters is by no means trivial. In chapter 4 the various stages in the sequential circuit (finite state machine) implementation of a digital filter are considered. The chapter begins with a review of finite state machines and their application to digital filter modelling. The following section deals with the choice of logic form. Two of the state (code) assignment algorithms in the literature are reviewed and a new algorithm is presented. The chapter concludes with some work on logic minimisation.

In chapter 5 a hardware Reed-Muller Transform (98) Processor is presented. The processor is intended to speed up the new state (code) assignment algorithm presented in chapter 4.

Three examples of higher order free quantization law filters are presented in chapter 6. For each a comparison is drawn with the equivalent fixed point arithmetic filter. The examples are:

(1) a Wideband Differentiator

(2) a Low Pass All-Pole filter designed to be optimally linear phase in the passband, and



Canonic Form of Second Order Filter Section

Figure 1.8.1

(3) a 6th order Band Pass elliptic filter.

Chapter 7 concludes the dissertation and contains some suggestions for further work.

To enable all experimental results to be repeated, experimental techniques are outlined in Appendix A.

2. First Order Free Quantization Law Digital Filters

2.1 Introduction

In this chapter a new approach to the design of digital filters is considered. In this approach each of the variables of an ideal-unlimited arithmetic precision-digital filter is assigned its own optimal quantization law. The object of the method is to achieve the maximum possible signal to noise ratio, for a given wordlength, at the output of the filter. Use of the method relies on having a-priori knowledge of the statistics of the signal to be filtered. The entire class of filters designed by this method will be called free quantization law filters. Specific examples, for instance a filter optimised for a gaussian input process, will be given the name of the input process - in this case gaussian optimised filter. The essential difference between a fixed point arithmetic filter and a free quantization law filter is that the latter is not only designed around a frequency or time domain specification but also around the signal to be processed. It is reasonable to suppose that this might give an improved output signal to noise ratio, and indeed this will be shown to be the case.

The choice of an optimal quantizer for a particular variable will in general depend on the following considerations:

(1) the amplitude probability density function (pdf) of the signal at that variable,

(2) the number of amplitude levels to be used to quantize that variable,

(3) the optimality criterion,

and (4) the quantization law at each of the other variables.

In section 2.4 the signal pdf will be considered. The number of quantization levels is arbitrary, although for economy it is desirable to use as few as possible. For the purposes of this research 255 levels will be used. The optimality criterion used will be minimum mean square (mms) error as this is

equivalent to minimising the time averaged distortion power. Optimisation of a quantizer on the basis of considerations (1) - (3) is relatively straightforward. It is usually referred to as the optimisation of a memoryless quantizer. The last consideration (4) complicates the optimisation so much so that it will be ignored. This is justified by arguing that provided sufficient levels are used at each variable the system can be modelled by a linear systems driven at various points by independent noise sources.

This chapter is concerned with the design of first order recursive filters. Where appropriate comparisons will be drawn between free and uniform - fixed point arithmetic - quantization law filters. The chapter continues with a review of optimal memoryless quantizers.

2.2 Optimal Quantizers

A bandlimited stationary stochastic process is sampled every T seconds. The sample values $x(nT)$ which are distributed in the interval $|-\infty,+\infty|$ with a probability density $\rho(x)$ are then quantized by an N-level quantizer. This quantizer is to be designed to produce minimum signal distortion.

The function $\rho(x)$ is assumed to be even. It follows that the corresponding signal is zero mean. Obviously this does not limit the generality of the analysis to follow, for if a signal has a probability density $\rho(x-\mu)$ then subtracting μ from each sample will change its density to $\rho(x)$. That $\rho(x)$ is even is a reasonable assumption for many naturally occurring signals.

Following some definitions the conditions under which a quantizer is optimal and hence produces least signal distortion will be derived. This derivation is based on the work of Max[59].

Let x be the value of an unquantized signal sample and define a set of decision levels, d_i, and a set of quantization levels, q_i, constrained by the inequality:

$$-\infty = d_1 < q_1 < d_2 < q_2 < \ldots < q_N < d_{N+1} = +\infty \qquad (2.2.1)$$

such that if x lies in the interval $|d_i, d_{i+1}|$ for $1 \leq i \leq N$ then it is quantized to (replaced by) q_i. The resulting instantaneous error, e, is given by:

$$e = x - q_i \tag{2.2.2}$$

Further define an error measuring function f(.) having the following properties:

$$f(0) = 0 \tag{2.2.3}$$

$$f(a) = f(-a)$$

$$f(a) < f(b) \iff 0 < a < b$$

The distortion introduced by the quantizer, denoted by D, is defined as:

$$D = E[f(e)] \tag{2.2.4}$$

where E is the expectation operator.

Now, as mentioned in the introduction to this chapter, the error measure is minimum mean square (m.m.s.) and so function f(.) becomes:

$$f(x) = x^2 \tag{2.2.5}$$

Equation 2.2.4 can now be rewritten as:

$$D = \sum_{i=1}^{N} \int_{d_i}^{d_{i+1}} (x - q_i)^2 \rho(x)\, dx \tag{2.2.6}$$

Equating to zero each of the partial derivatives of D with respect to each of the q_i and d_i the following necessary conditions for optimality are obtained:

$$q_i = \frac{\int_{d_i}^{d_{i+1}} x\, \rho(x)\, dx}{\int_{d_i}^{d_{i+1}} \rho(x)\, dx} \qquad \text{for } 1 \leq i \leq N \tag{2.2.7a}$$

and

$$d_{i+1} = \frac{q_i + q_{i+1}}{2} \qquad \text{for } 1 \leq i \leq N-1 \qquad (2.2.7b)$$

Fleischer[60] has shown that if the inequality:

$$\frac{d^2}{dx^2}\left[\ln(\rho(x))\right] < 0 \qquad (2.2.8)$$

holds and relation 2.2.1 is satisfied then the set of equations 2.2.7 have a unique solution.

Analytical solution of equations 2.2.7 is only possible if $\rho(x)$ = constant (in which case the result is a uniform quantizer). Numerical solution is straightforward; the details will now be presented for the case of a zero mean gaussian white process described by the probability density function:

$$\rho(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \qquad (2.2.9)$$

where σ is the standard deviation. The reason for this choice of $\rho(x)$ will be explained later in this chapter.

Since $\rho(x)$ is an even function the quantization and decision levels obey the following relations:

N - odd

$$q_{\frac{N+1}{2}} = 0 \qquad (2.2.10a)$$

$$q_{N-K+1} = -q_K \quad \text{for } 1 \leq K \leq N \qquad (2.2.10b)$$

$$d_{N-K+2} = -d_K \quad \text{for } 1 \leq K \leq N+1 \qquad (2.2.10c)$$

These equations enable the amount of computing required in the solution of 2.2.7 to be halved.

Following Wood[64], define two sets of dependent variables Δ_i and δ_i by:

$$\Delta_i = d_{i+1} - d_i \qquad (2.2.11)$$

$$\delta_i = \frac{d_{i+1} + d_i}{2} \qquad \text{for } 2 \leq i \leq N-1$$

substitute them for d_i and d_{i+1} in equations 2.2.7a and expand the integrals as Taylor Series. Provided the expansion of $\rho(x)$ about δ_i exists equations 2.2.7a become:

$$q_i = \delta_i + \frac{1}{12} \Delta_i^2 \frac{\rho'(\delta_i)}{\rho(\delta_i)} \quad \text{for } 2 \leq i \leq N-1 \qquad (2.2.12)$$

Now from equation 2.2.9:

$$\frac{\rho'(\delta_i)}{\rho(\delta_i)} = \delta_i$$

and so:

$$q_i = \delta_i \left(1 - \frac{1}{12} \Delta_i^2\right) = \frac{d_{i+1} + d_i}{2} \left(1 - \frac{1}{12} (d_{i+1} - d_i)^2\right)$$

which when rearranged leads to the following recurrence formula:

$$d_{i+1}^3 - d_i d_{i+1}^2 - (d_i^2 + 12)\, d_{i+1} + d_i (d_i^2 - 12) + 24 q_i = 0 \qquad (2.2.13)$$

The Taylor Series expansion may not be used to solve equation 2.2.7a when i = N since the upper integral limit is infinite. Instead, the following relation is used:

$$q_N = \frac{p(d_N)}{1 - P(d_N)} \qquad (2.2.14)$$

where

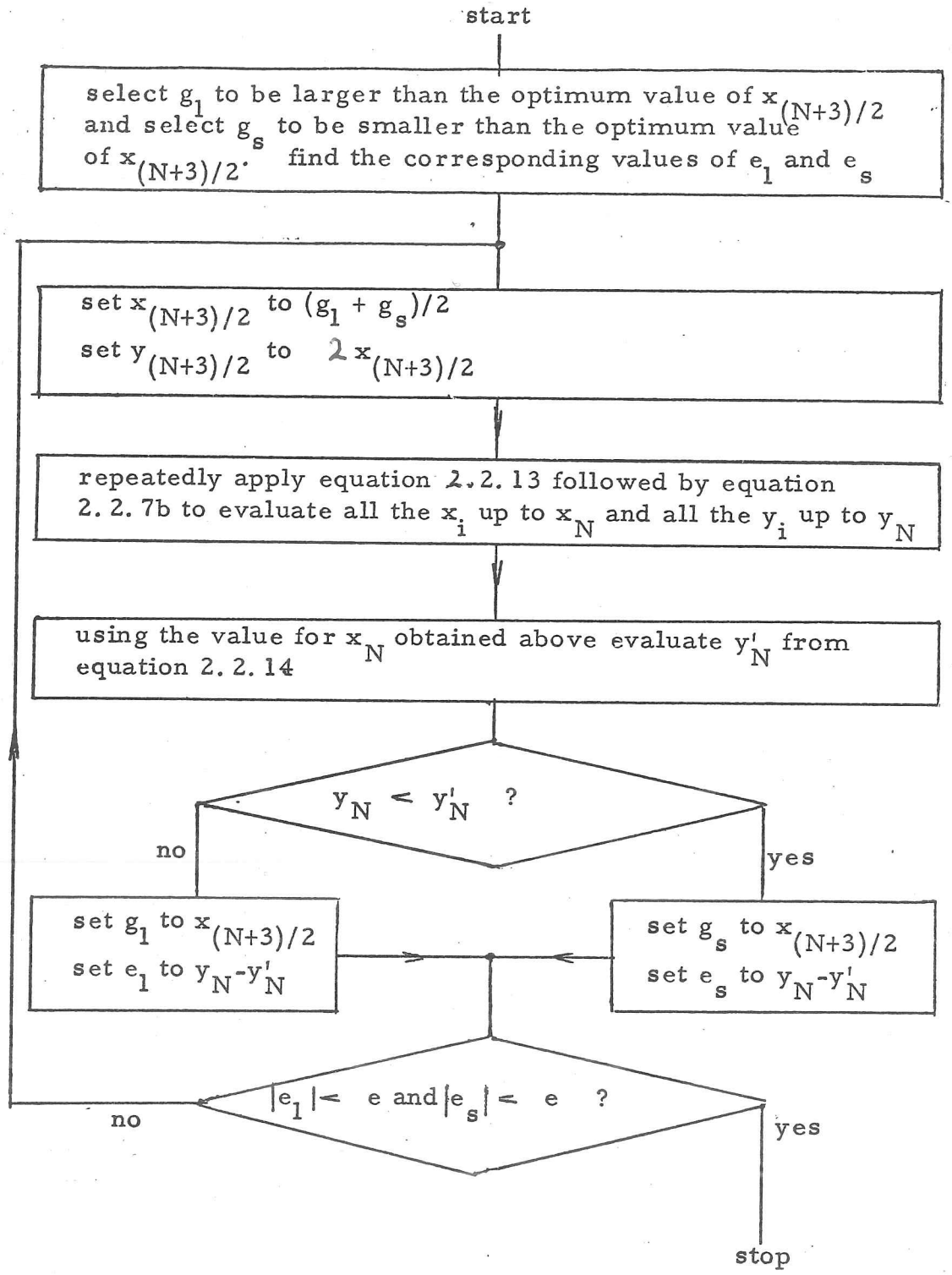
$$P(d_N) = \int_{-\infty}^{d_N} \rho(x)\, dx$$

is obtained using a rational approximation. Appendix B.

From equations 2.2.7b and 2.2.10a it can be seen that the quantization and decision levels immediately to the right of the origin are related by:

$$q_{\frac{N+3}{2}} = 2\, d_{\frac{N+3}{2}} \qquad (2.2.15)$$

The strategy for obtaining an optimal quantizer for a gaussian process which is based on the bisection method of iterative equation solving, and employs equations 2.2.7a, 13, 14,15 is shown in figure 2.2.1. The graph



Algorithm to find the decision and quantization levels of the m.m.s.e. optimal quantizer for a gaussian process.

Figure 2.2.1

in figure 2.2.2 shows q_i versus i. The optimum 255-level quantizer for a gaussian process gives a signal to noise ratio of 44 dB.

An important property of the m.m.s. optimised quantizer is that it produces an instantaneous distortion which is orthogonal to its output. This can be seen by combining equations 2.2.7a and equation 2.2.6 to give:

$$D = \int_{-\infty}^{+\infty} x^2 \rho(x)\, dx - \sum_{i=1}^{N} y_i^2 \int_{x_i}^{x_{i+1}} \rho(x)\, dx.$$

which is recognised as:

$$\sigma_e^2 = \sigma_y^2 - \sigma_{y'}^2, \qquad (2.2.16)$$

where σ_e^2 is the distortion power induced by the quantizer,

σ_y^2 is the power of the input signal,

and $\sigma_{y'}^2$, is the power of the output signal.

In addition, the quantizer input, y, and output, y', are related to the instantaneous quantization distortion, e, by:

$$y = y' + e$$

and so:

$$\sigma_y^2 = \sigma_{y'}^2 + 2\sigma_{y'e} + \sigma_e^2 \qquad (2.2.17)$$

From equation 2.2.16 and 2.2.17 it follows that:

$$\sigma_{ye} = 0$$

This is important because it is always assumed to be true when analysing the signal to noise performance of a digital filter implementation [1].

2.3 Optimum Step Size for a Uniform Law Quantizer used to Quantize a Gaussian Process

In order to make a valid comparison between a free quantization law filter and a uniform quantization law filter - fixed point arithmetic filter -

in figure 2.2.2 shows q_i versus i. The optimum 255-level quantizer for a gaussian process gives a signal to noise ratio of 44 dB.

An important property of the m.m.s. optimised quantizer is that it produces an instantaneous distortion which is orthogonal to its output. This can be seen by combining equations 2.2.7a and equation 2.2.6 to give:

$$D = \int_{-\infty}^{+\infty} x^2 \rho(x)\, dx - \sum_{i=1}^{N} y_i^2 \int_{x_i}^{x_{i+1}} \rho(x)\, dx.$$

which is recognised as:

$$\sigma_e^2 = \sigma_y^2 - \sigma_{y'}^2, \qquad (2.2.16)$$

where σ_e^2 is the distortion power induced by the quantizer,

σ_y^2 is the power of the input signal,

and $\sigma_{y'}^2$ is the power of the output signal.

In addition, the quantizer input, y, and output, y', are related to the instantaneous quantization distortion, e, by:

$$y = y' + e$$

and so:

$$\sigma_y^2 = \sigma_{y'}^2 + 2\sigma_{y'e} + \sigma_e^2 \qquad (2.2.17)$$

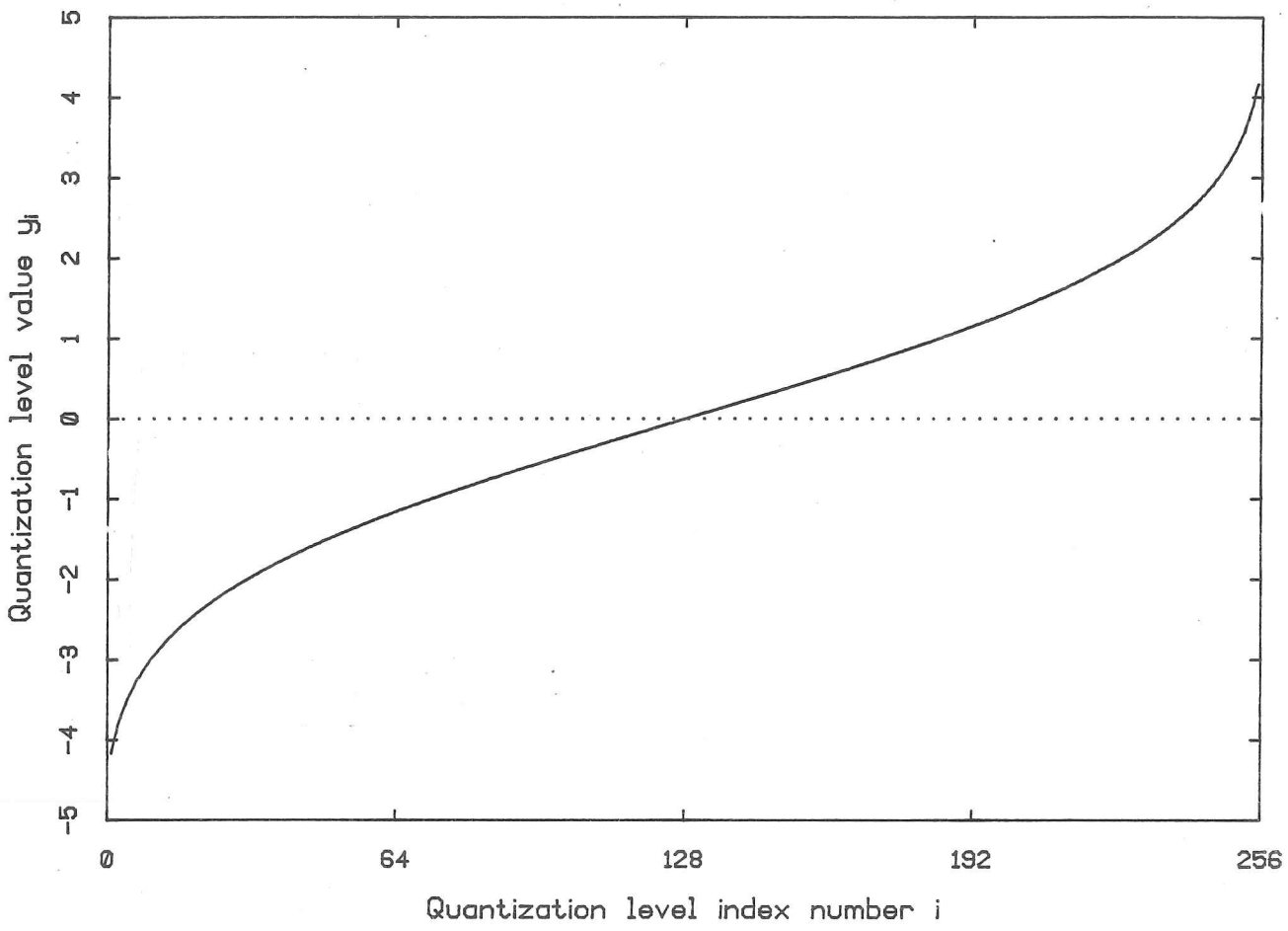
From equation 2.2.16 and 2.2.17 it follows that:

$$\sigma_{ye} = 0$$

This is important because it is always assumed to be true when analysing the signal to noise performance of a digital filter implementation [1].

2.3 Optimum Step Size for a Uniform Law Quantizer used to Quantize a Gaussian Process

In order to make a valid comparison between a free quantization law filter and a uniform quantization law filter - fixed point arithmetic filter -



Quantization level v. index number for a
255-level gaussian process optimal quantizer

Figure 2.2.2

it is of course necessary to use the same wordlength for corresponding filter variables. It is also necessary to drive the uniform quantization law filter with a signal the power level of which is chosen to give maximum output signal to noise ratio. In order to find the appropriate power level the following problems must be solved:

(1) To find the optimum step size for a uniform law quantizer used to quantize a gaussian process of known variance,

and (2) To find the signal to noise ratio derating on deviating from the optimum step size found in (1).

In this section a formula for the optimum step size as a function of the probability and cumulative density functions of the quantizer input signal is derived. The formula is evaluated for various numbers of quantization levels for the case of a gaussian process.

The formula is derived as follows. Define x, $\phi(x)$ and $\Phi(x)$ as the value, probability density function and cumulative density function respectively of a gaussian process. Define, 2Δ, the quantization step size as:

$$2\Delta = q_{i+1} - q_i \qquad \text{for } 1 \leqslant i \leqslant N-1 \tag{2.3.1}$$

where the quantization levels, q_i, are as defined in section 2.2.

Substituting the constraint equation 2.3.1 into equations 2.2.7 it follows that the optimum value of 2Δ is the solution of:

$$2\Delta = \frac{\sum_{n=1}^{\frac{N-1}{2}} \phi((2n-1)\Delta)}{\frac{(N-1)^2}{4} - \sum_{n=1}^{\frac{N-1}{2}} (2n-1)\,\Phi((2n-1)\Delta)} \tag{2.3.2}$$

where, N, the number of quantization levels is odd.

The solution to equation 2.3.2 is most readily obtained using the recurrence:

$$\Delta_k = \tfrac{1}{2} f(\Delta_{k-1})$$

where $f(\Delta_k)$ is the value of the right hand side of 2.3.2 at $\Delta = \Delta_k$. For some useful values of N the optimum step size is given in figure 2.3.1.

No. of quantization levels, N	Optimum step size 2Δ
63	0.105485
127	0.057254
255	0.030856
511	0.016504
1023	0.008752

Figure 2.3.1

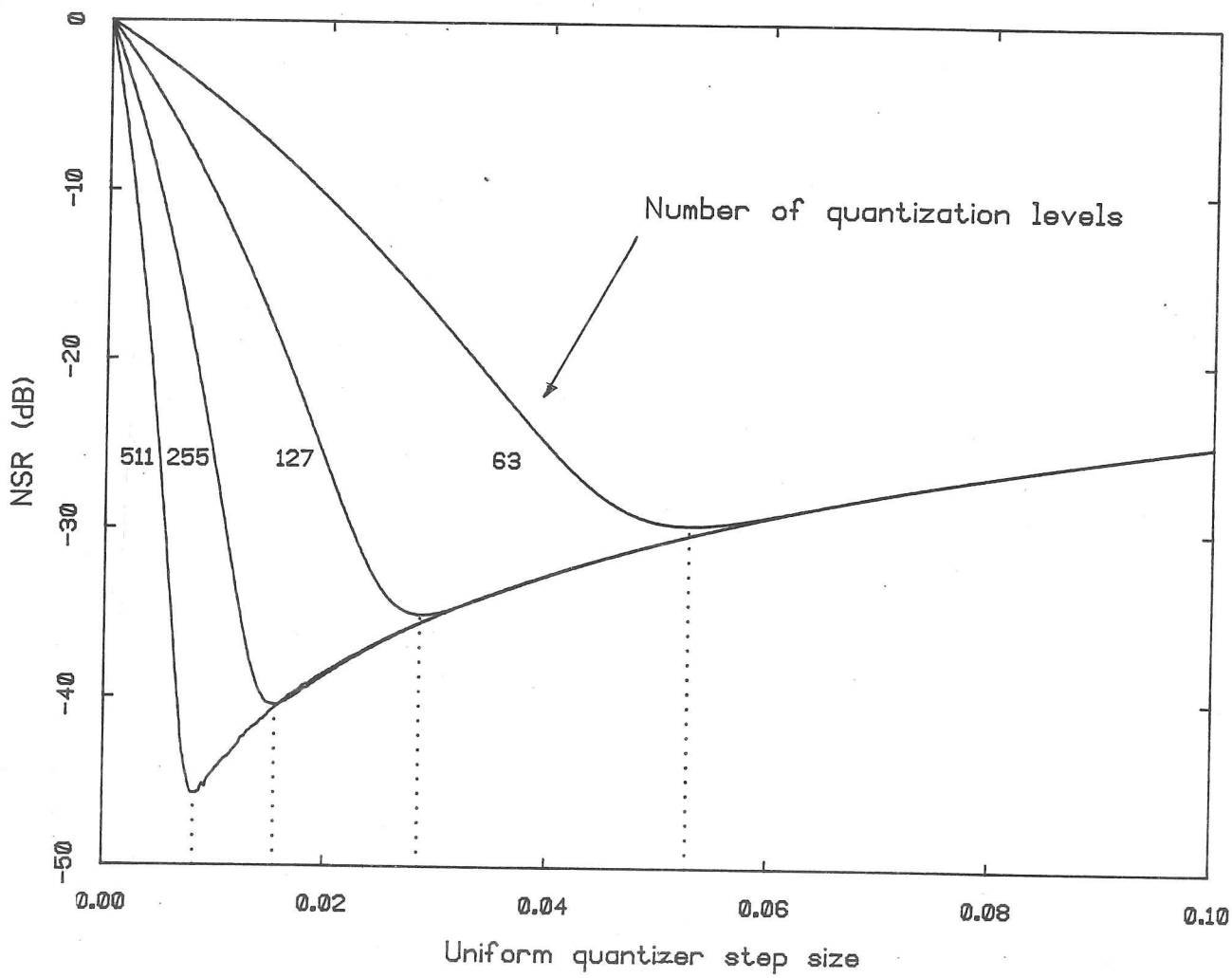
Attention now turns to how the signal to noise ratio performance of a uniform law quantizer varies with step size. Using the definition of distortion, D, given in equation 2.2.6 the signal to noise ratio performance of a uniform law quantizer was computed for the same set of values of N as in figure 2.3.1. The graph in figure 2.3.2 shows noise to signal ratio as a function of step size.

2.4 Signal and System Characteristics

Except where otherwise stated all filters will be driven by a zero mean, gaussian and white stochastic process. All signals will be assumed to be ergodic and hence the words variance and power will be used interchangeably according to the context.

Filters will be modelled using an ideal linear system driven at various points by white noise processes. Filter structures will be presented using a variant on the signal flow graph. For example a box labelled T will be used in place of z^{-1}. The following properties of linear systems will be required.

(1) If a stable time invariant linear system is driven by a gaussian process then each of the nodes of that system exhibits a gauss-markov process [8],



Noise to Signal Ratio as a function of step size
for a uniform quantizer operating on a
unit variance gaussian process,
and for various numbers of quantization levels

Figure 2.3.2

(2) If a stable time invariant linear system described by a pulse transfer function H(z) is driven by a white process of variance σ_x^2 then the variance at the output σ_y^2 is given by:

$$\sigma_y^2 = \frac{\sigma_x^2}{2\pi j} \oint_c H(z)\, H(z^{-1})\, z^{-1}\, dz \qquad (2.4.1)$$

where c is the z-plane unit circle.

Equation 2.4.1 is a special case of the complex convolution integral[1,3]

If a linear system is driven by a zero mean process then its output also is a zero mean process. This follows from the linearity equation.

$$L\,(ax + by) = aL(x) + bL(y) \qquad (2.4.2)$$

where a and b are constants.

2.5 First Order All-Pole - Design

The signal flow graph for a first order all-pole is shown in figure 2.5.1.

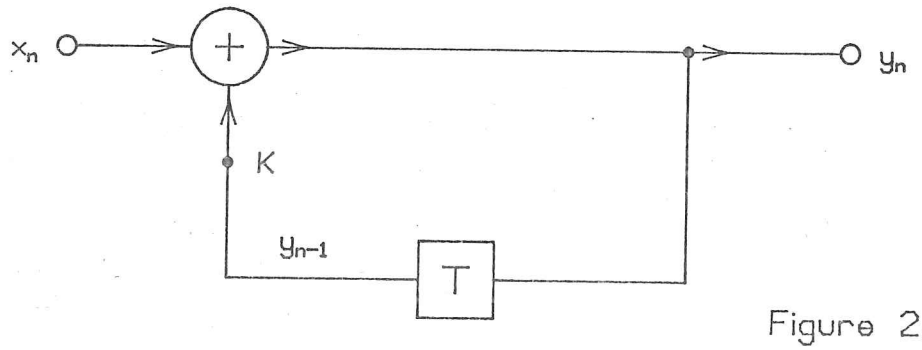


Figure 2.5.1

This system is stable provided K lies in the closed interval $|-1, +1|$, and is said to be marginally stable if K = +1 or K = -1. Assuming stability and given that x_n, the input sample at time nT, is distributed with gaussian statistics the output sample y_n is distributed with gauss-markov statistics.

The pulse transfer function corresponding to figure 2.5.1, denoted by H(z), is given by:

$$H(z) = \frac{z}{z - K} \qquad (2.5.1)$$

Substituting 2.5.1 into 2.4.1 and performing the contour integration it is seen that the power of process y_n, denoted by σ_y^2, is related to σ_x^2, the power of process x_n, by:

$$\sigma_y^2 = \sigma_x^2 \frac{1}{1 - K^2} \qquad (2.5.2)$$

It follows that for a stable first order all-pole :

$$\sigma_y^2 > \sigma_x^2 \qquad (2.5.3)$$

Let $\sigma_x^2 = 1$ then the optimum quantizer for x_n is the one shown in figure 2.2.2. The optimum quantizer for y_n is also the one shown in figure 2.2.2 but with all the quantization levels multiplied by $\sqrt{G}$. The parameter G will be called the integrated power gain. It is given by:

$$G = \frac{1}{1 - K^2} \qquad (2.5.4)$$

This design procedure makes one of the assumptions mentioned in the introduction. It assumes that the optimal quantizers for x_n and y_n do not depend on each other.

2.6 First Order All-Pole - Error Model

An error model for the gaussian optimised free quantization law first order all-pole will now be presented.

A block diagram showing all the arithmetic operations of the filter is shown in figure 2.6.1.

The boxes Q_x and Q_y are the input and output quantizers respectively. Primed variables are quantized versions of unprimed variables.

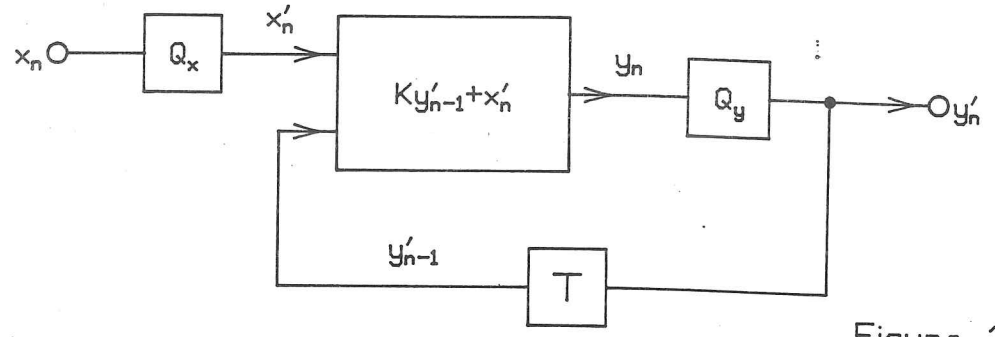


Figure 2.6.1

In section 2.2 it was shown that if quantizers Q_x and Q_y are optimised using an m.m.s. error criterion then the following statistical model is exact.

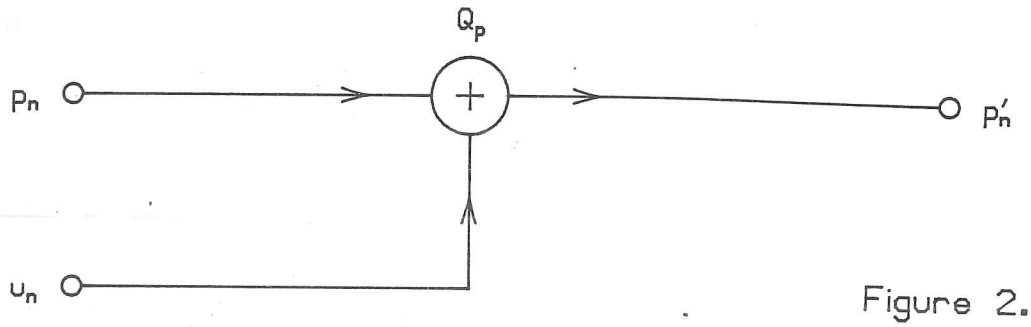


Figure 2.6.2

where p_n is the unquantized input sample,

p'_n is the quantized output sample,

u_n is the instantaneous error and is orthogonal to p_n',

and Q_p is the optimum quantizer for process p_n.

Replacing quantizers Q_x and Q_y in figure 2.6.1 by the model in 2.6.2 the following signal flow graph is obtained:

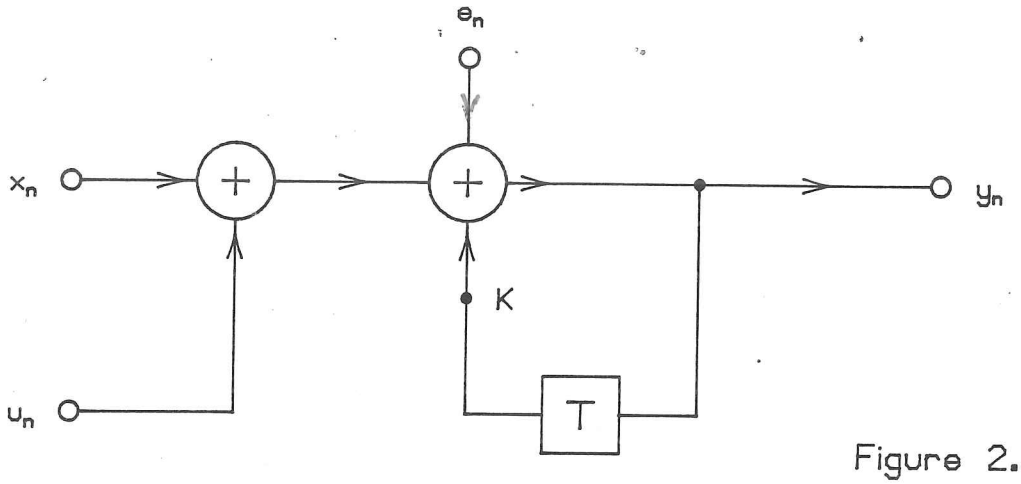


Figure 2.6.3

where u_n is the noise which results from quantizing x_n and e_n is the noise due to quantizing y_n.

Denoting by σ_u^2 the power of noise process u_n and by σ_e^2 the power of noise process e_n it is seen that:

$$\frac{\sigma_y^2}{\sigma_x^2} = \frac{\sigma_e^2}{\sigma_u^2} \tag{2.6.1}$$

It follows from equation 2.5.2 and 2.6.1 that:

$$\sigma_e^2 = \sigma_u^2 \frac{1}{1-K^2} \tag{2.6.2}$$

Assume that processes u_n and e_n are uncorrelated with each other and further assume that both have white power spectra; then the output noise power, denoted by σ_{yn}^2, is given by:

$$\sigma_{yn}^2 = \frac{\sigma_u^2 + \sigma_e^2}{1 - K^2} \tag{2.6.3}$$

which using equation 2.6.2 becomes:

$$\sigma_{yn}^2 = \frac{\sigma_u^2}{1 - K^2} \left[1 + \frac{1}{1 - K^2}\right] \tag{2.6.4a}$$

$$= \sigma_u^2 \frac{2 - K^2}{(1 - K^2)^2} \tag{2.6.4b}$$

Now the output signal power, denoted by σ_{ys}^2, is given by:

$$\sigma_{ys}^2 = \sigma_x^2 \cdot \frac{1}{1 - K^2} \qquad (2.6.5)$$

From equations 2.6.4b and 2.6.5 it follows that the output signal to noise ratio, SNR_o, in dB is:

$$SNR_o = 10 \log_{10} \frac{\sigma_x^2}{\sigma_u^2} \left[\frac{1 - K^2}{2 - K^2}\right] \text{ dB} \qquad (2.6.6a)$$

$$= 10 \log_{10} \left[\frac{\sigma_x^2}{\sigma_u^2}\right] - 10 \log_{10} \left[\frac{2 - K^2}{1 - K^2}\right] \text{ dB} \qquad (2.6.6b)$$

The left term is simply the signal to noise ratio performance of the optimal memoryless quantizer for process x_n. It may be thought of as the input signal to noise ratio SNR_I. The signal to noise equation becomes:

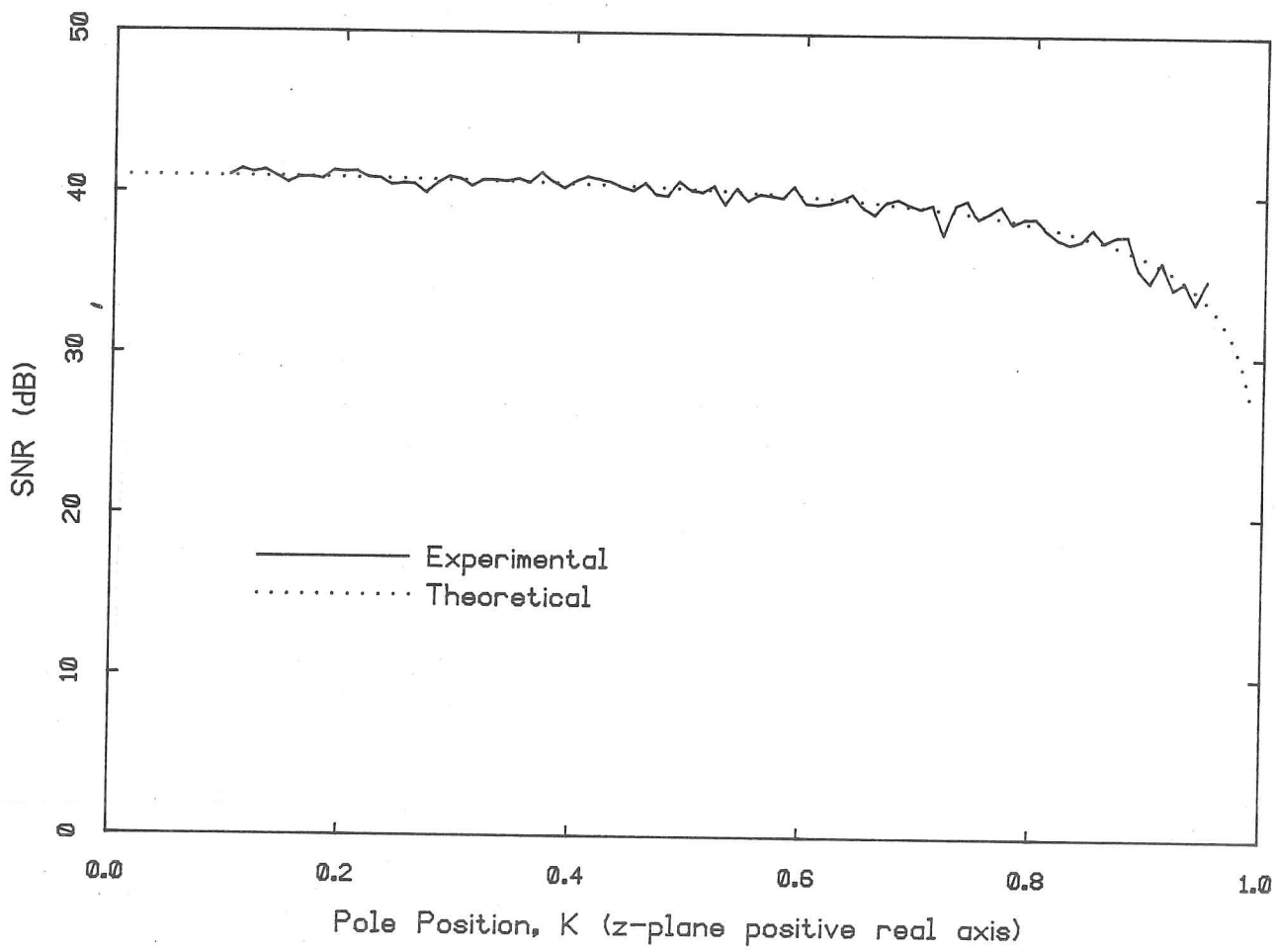
$$SNR_o = SNR_I - 10 \log_{10} \left[\frac{2 - K^2}{1 - K^2}\right] \text{ dB} \qquad (2.6.6c)$$

The graph in figure 2.6.4 shows theoretical and measured signal to noise ratio performance of a first order all-pole as a function of pole position K. The error model is seen to be accurate.

It is not obvious how the SNR performance of an optimal gaussian quantizer changes with variations in the input signal power. This is because the distortion power depends on the signal level. By experiment though 2.6.6c was found to fall by 6 dB each time σ_x^2 was halved.

2.7 Uniform Quantization Law First Order All-Pole Driven by a Gaussian Process - Error Model

In this section the results presented in section 2.3 are used to estimate the step size to be used in a uniform quantization law digital filter, as a function of the variance of the input gaussian process, such that the output signal to noise ratio performance is maximised.



Experimentally observed and theoretically predicted
SNR performance of a gaussian optimised
first order recursive all-pole .

Figure 2.6.4

The error model to be used is shown in figure 2.7.1 Rabiner and Gold[1].

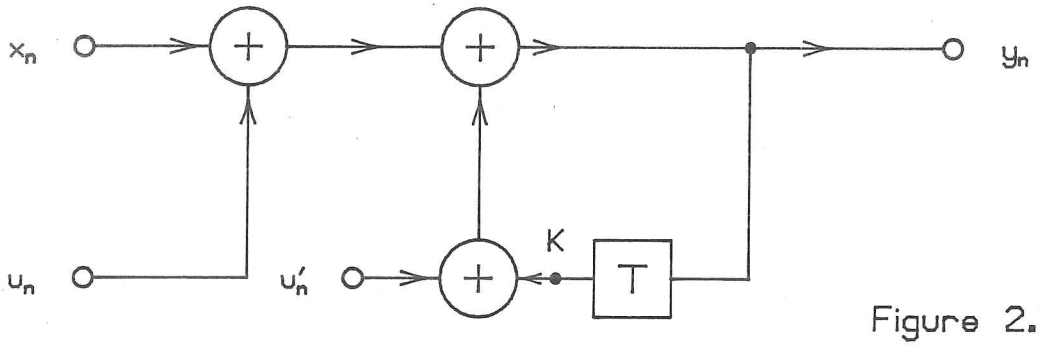


Figure 2.7.1

where x_n and y_n are input and output samples,

u_n is a noise process introduced by the input quantizer,

and u_n' is a noise process introduced by quantization after the xK multiplier.

The following usual assumptions about noise sources u_n and u_n' will be made:

(1) Both have white power spectra,

(2) They are uncorrelated with each other,

and (3) Both are uncorrelated with the input signal.

From the graph in figure 2.3.2. it can be seen that the signal to noise ratio performance of a uniform quantizer rapidly reduces for only small amounts of overdrive above the optimum level. However, the reduction in SNR due to underdrive is much less, being approximately 6 dB each time the signal power is halved. It is therefore reasonable to assume that a filter system, implemented using a uniform quantization law, will give best performance when each of its variable is operating at the highest possible power level such that none of them are overdriven. Clearly, it is necessary to find the variable with the highest power level and to arrange that it operates at the optimum point. The input level is then given by the reciprocal of the integrated power gain from the input to that variable.

For the case of the filter in figure 2.7.1 the input and output power levels obey the inequality:

$$\sigma_y^2 > \sigma_x^2 \qquad (2.7.1)$$

Since σ_y^2 is the larger it will be treated as a reference. Making $\sigma_y^2 = 1$, the optimum step size for a 255-level quantizer and the input variance are given by:

step size, $2\Delta = 0.030856$ from figure 2.3.1,

and, input variance, $\sigma_x^2 = 1 - K^2$ from equation 2.5.2.

It follows that the noise power at the output of the filter, denoted by σ_{yn}^2, is given by:

$$\sigma_{yn}^2 = \frac{(\sigma_u^2 + \sigma_{u'}^2)}{(1 - K^2)} \qquad (2.7.2)$$

$$= 2 \times \frac{(2\Delta)^2}{12} \times \frac{1}{1 - K^2}$$

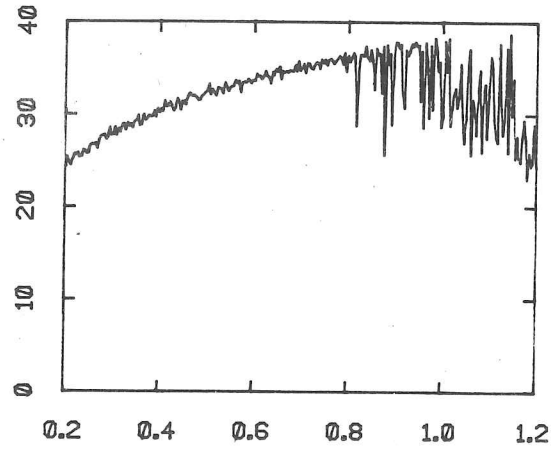
and hence the output signal to noise ratio is given by:

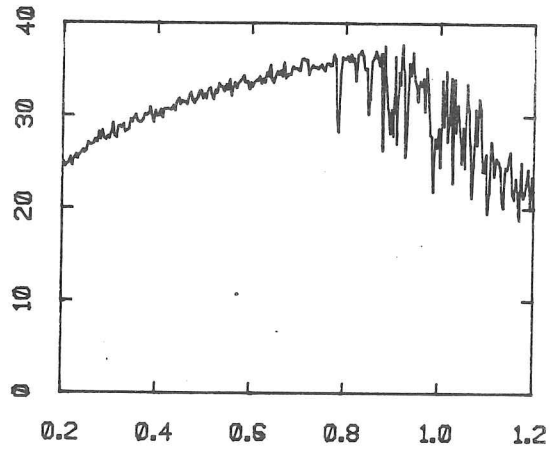
$$SNR_o = 10 \log_{10} \left[1/ \left(2 \times \frac{(2\Delta)^2}{12} \times \frac{1}{1 - K^2} \right) \right] \text{ dB} \qquad (2.7.3)$$

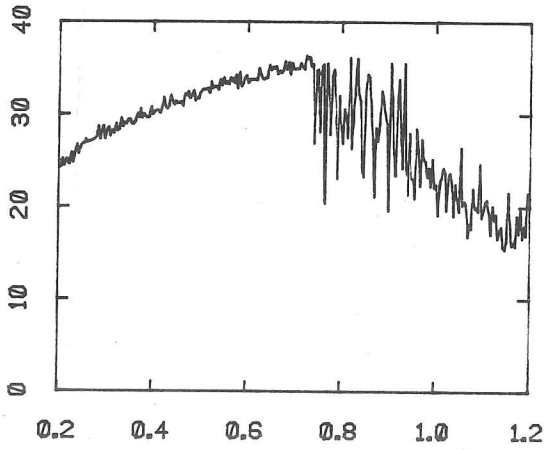
$$= 37.995 + 10 \log_{10} (1 - K^2) \text{ dB}$$

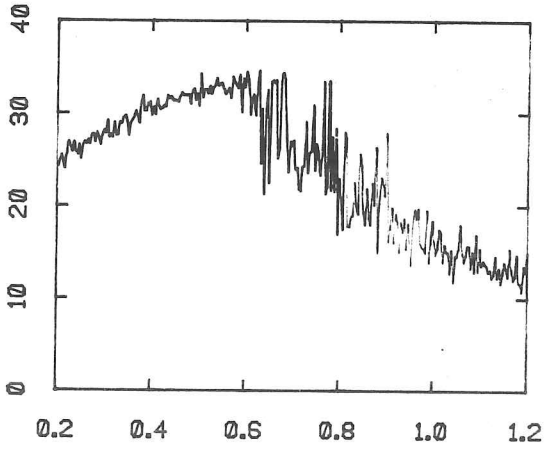
To assess the accuracy of this analysis the filter in figure 2.7.1 was simulated by computer. It was driven by a unit variance gaussian process fed through a scaling multiplier. The filter step size was set as above. The scaling multiplier was varied by small amounts from 0.2 to 1.2 and at each point the signal to noise ratio was measured. The experiment was repeated for five different values of the filter coefficient K. The graphs in figure 2.7.2 shows the results. The predicted optimum value for the scaling multiplier is $\sqrt{1-K^2}$. It is found in each case that the SNR is no more than 1 dB lower than the optimum.

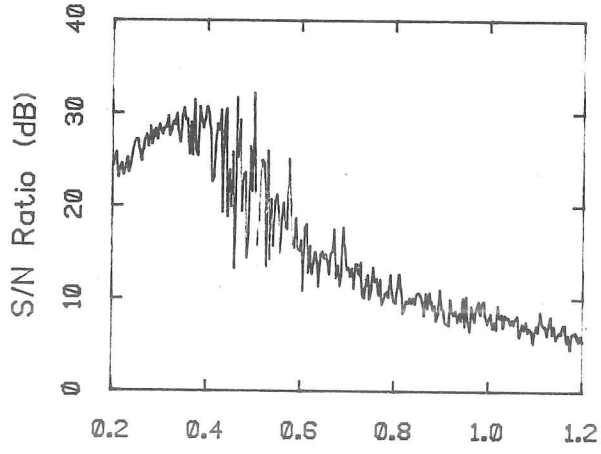
It is now possible to present a comparison between the gaussian optimised and the uniform quantization law filters. The graph in figure 2.7.3 shows the





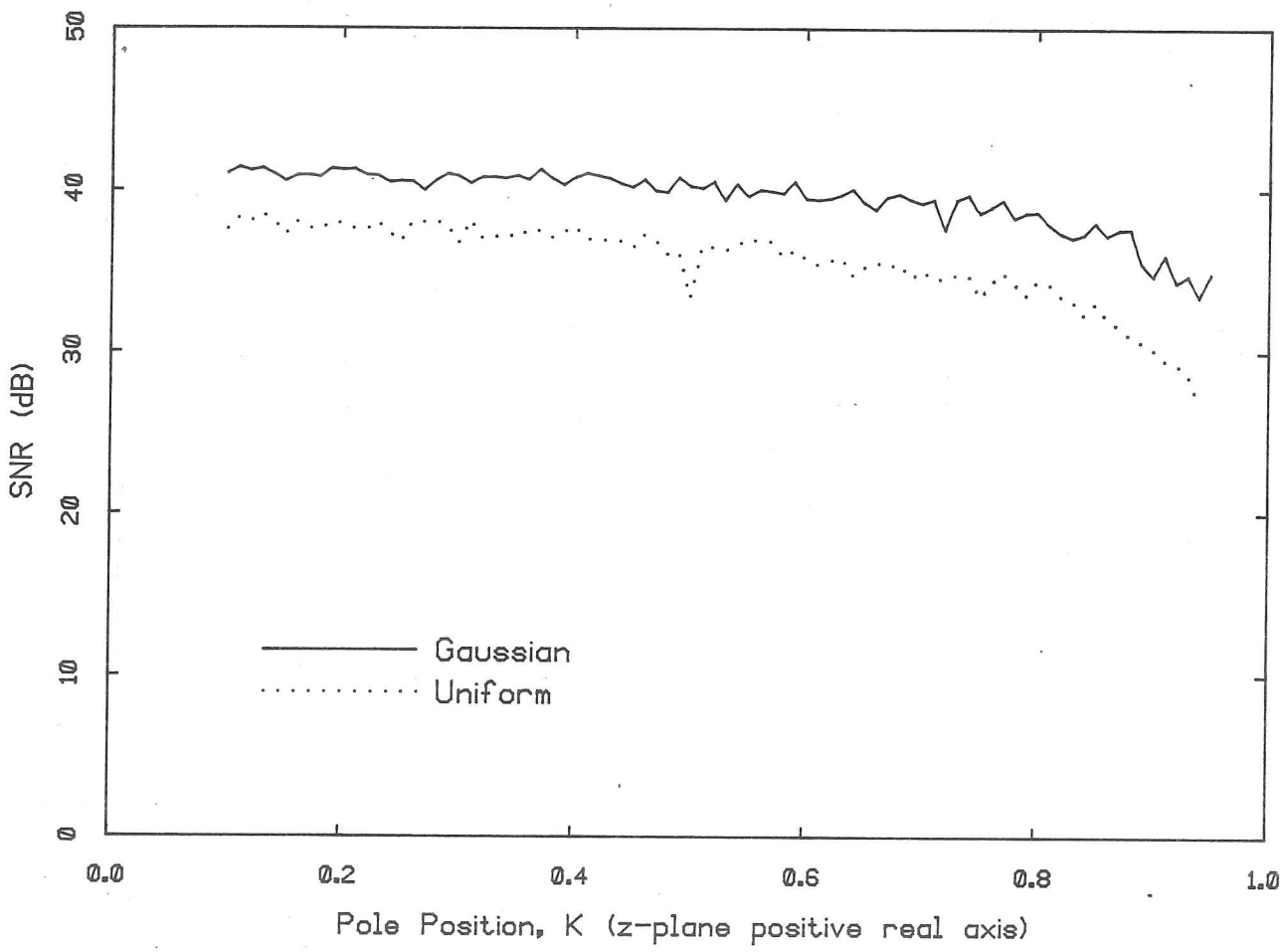






Signal to Noise Ratio as a function of input signal level and of K, the pole position, for a uniform quantization law first order all-pole.

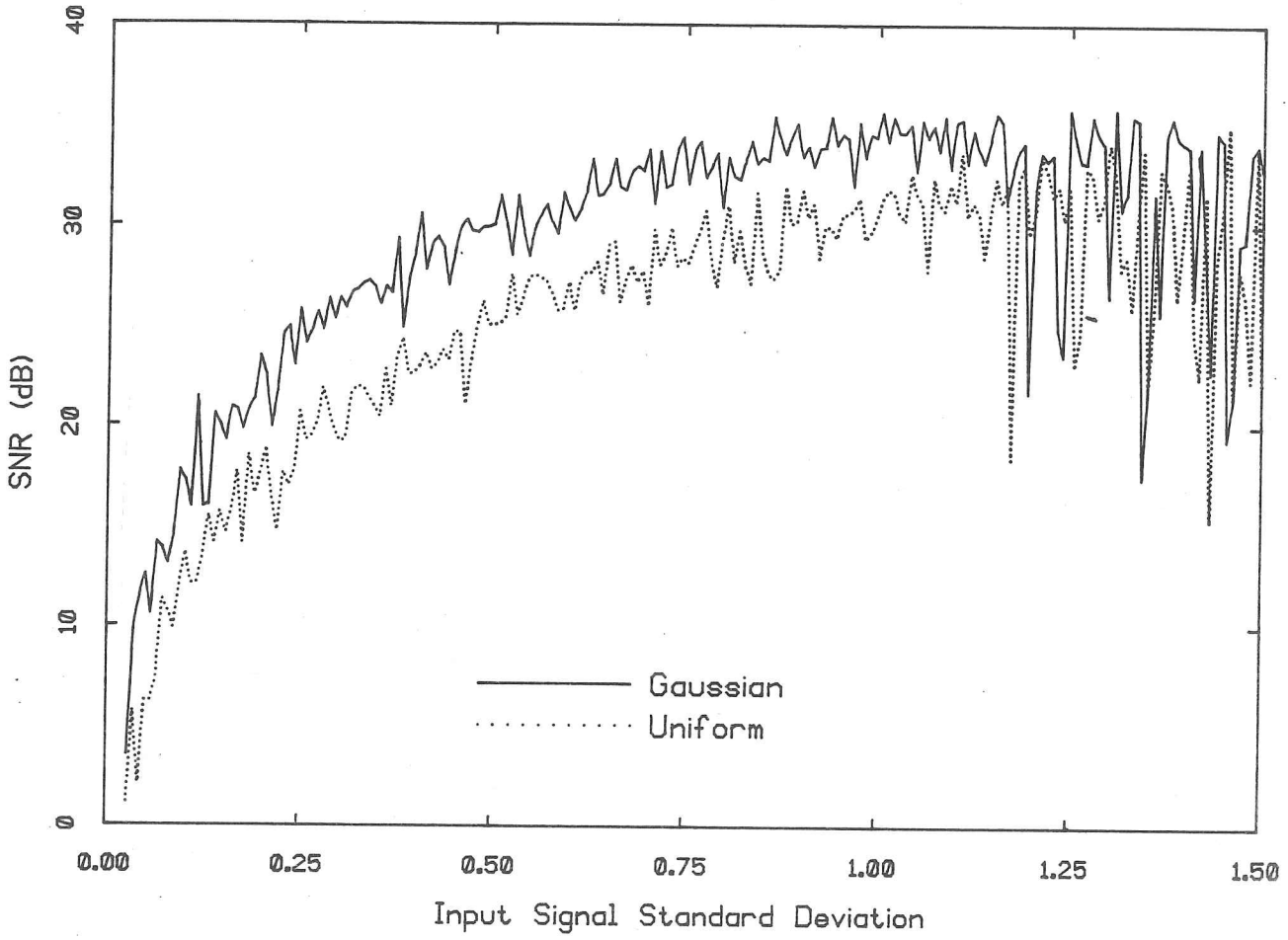
Figure 2.7.2



Signal to Noise ratio performance as a function of pole position
for a first order all-pole section .
Comparison between uniform and gaussian optimal quantization .

Figure 2.7.3

signal to noise ratio performance for both implementations of a first order all-pole. The independent variable is the pole position, K. Only positive values of K are shown, however, the SNR was found to be an even function of K. The graph in figure 2.7.4 shows the variation in SNR for both filters as a function of input signal standard deviation.



Signal to Noise Ratio as a function of input signal standard deviation for a first order all-pole implemented using both uniform and gaussian optimal quantization laws.

Figure 2.7.4

2.8 Section Ordering - Two First Order All-Poles

The system to be analysed is shown below in Figure 2.8.1.

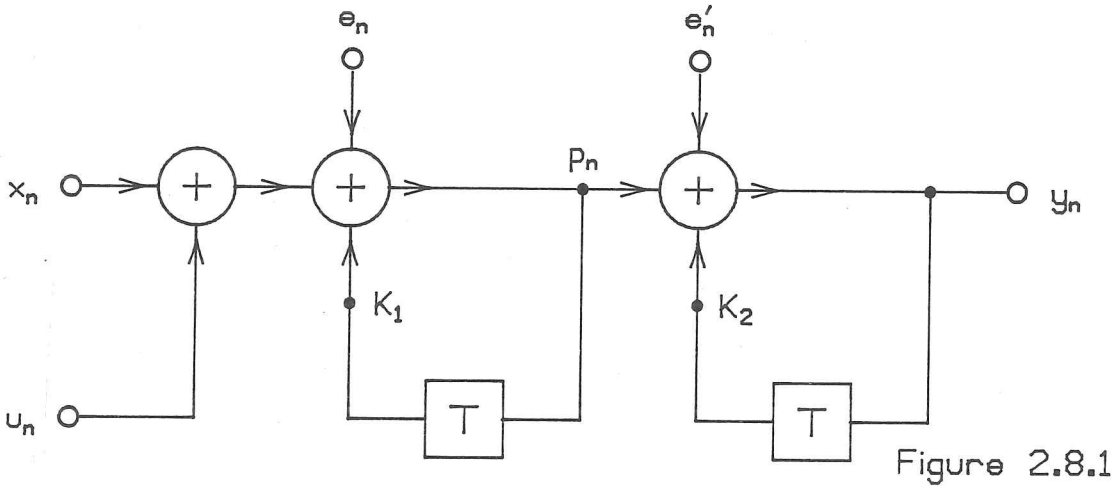


Figure 2.8.1

Sources u_n, e_n and e'_n are white noise processes. All other parameters are defined in the usual way.

The pulse transfer functions from input to p_n and from input to y_n are given by:

$$H_p(z) = \frac{P(z)}{X(z)} = \frac{z}{z - K_1} \qquad (2.8.1)$$

and

$$H_y(z) = \frac{Y(z)}{X(z)} = \frac{z^2}{(z-K_1)(z-K_2)} \qquad (2.8.2)$$

respectively.

Using equations 2.8.1, 2.8.2 and the complex convolution theorem in 2.4.1 it follows that, if x_n is a white process of variance σ_x^2, the variances at p_n and y_n, denoted by σ_p^2 and σ_y^2, respectively are given by:

$$\sigma_p^2 = \sigma_x^2 \frac{1}{1 - K_1^2} \qquad (2.8.3)$$

and

$$\sigma_y^2 = \sigma_x^2 \frac{1 + K_1K_2}{1 - K_1K_2} \quad \frac{1}{(1 - K_1^2)(1 - K_2^2)} \tag{2.8.4}$$

Now if σ_u^2 is the noise power produced by the input quantizer then from equations 2.8.3 and 2.8.4 it follows that noise sources e_n and e'_n have powers σ_e^2 and $\sigma_{e'}^2$ given by:

$$\sigma_e^2 = \sigma_u^2 \frac{1}{1 - K_1^2} \tag{2.8.5}$$

$$\sigma_{e'}^2 = \sigma_u^2 \frac{1 + K_1K_2}{1 - K_1K_2} \quad \frac{1}{(1 - K_1^2)(1 - K_2^2)} \tag{2.8.6}$$

The noise appearing at the filters output results from u_n and e_n filtered by both sections and from e_n' filtered by the second section only. It can be shown that the output signal to noise ratio, SNR_O, is given in terms of the input signal to noise ratio, SNR_I, by:

$$SNR_O = 10 \log_{10} \left[\frac{\sigma_x^2}{\sigma_u^2} \Big/ \left(1 + \frac{1}{1 - K_1^2} + \frac{1}{1 - K_2^2} \right) \right] \tag{2.8.7a}$$

$$\Rightarrow \quad SNR_O = SNR_I - 10 \log_{10} \left[1 + \frac{1}{1 - K_1^2} + \frac{1}{1 - K_2^2} \right] \tag{2.8.7b}$$

Since equation 2.8.7b is symmetrical in parameters K_1 and K_2 it follows that the overall signal to noise ratio SNR_o is insensitive to the section ordering.

2.9 First Order All-Zero - Design

So far only the design of a first order all-pole has been considered. Before considering the design of a pole-zero section the design procedure for a single zero will be presented.

Let x_n and y_n be the input and output respectively of a first order zero described by the constant coefficient difference equation:

$$y_n = x_n + Lx_{n-1} \tag{2.9.1}$$

The corresponding pulse transfer function:

$$H(z) = 1 + Lz^{-1} \tag{2.9.2}$$

has a single zero at $z = -L^{-1}$.

By the complex convolution theorem the integrated power gain, G, from the input to output is given by:

$$G = 1 + L^2 \tag{2.9.3}$$

The optimum quantizer for x_n, assuming $\sigma_x^2 = 1$, is the one shown in Figure 2.2.2. The optimum quantizer for y_n is similar except that its levels are scaled by a factor $\sqrt{1+L^2}$.

2.10 First Order All-Zero - Error Model

The first order zero described by the difference equation:

$$y_n = x_n + Lx_{n-1} \tag{2.10.1}$$

is shown together with noise sources appropriate to a gaussian optimised realisation in figure 2.10.1.

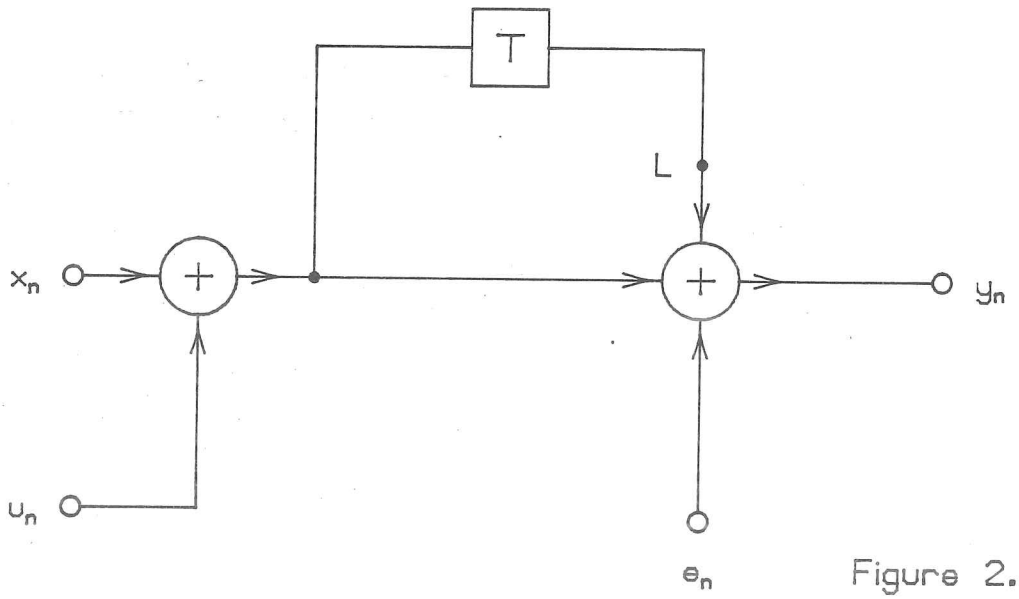


Figure 2.10.1

The integrated power gain from x_n to y_n was found in the last section to be:

$$G = 1 + L^2 \tag{2.10.2}$$

From 2.10.2 it follows that the noise sources u_n and e_n have power levels related by:

$$\sigma_e^2 = (1 + L^2)\,\sigma_u^2 \tag{2.10.3}$$

The total power at the output of the zero denoted by σ_{yn}^2 is given by:

$$\sigma_{yn}^2 = \underbrace{\sigma_u^2(1+L^2)}_{\text{input}} + \underbrace{\sigma_u^2(1+L^2)}_{\text{filter}} \tag{2.10.4}$$

The total signal power at the output σ_{ys}^2 is given by:

$$\sigma_{ys}^2 = \sigma_x^2(1+L^2) \tag{2.10.5}$$

From equations 2.10.(4,5) it follows that the output signal to noise ratio is given by:

$$SNR_O = 10 \log_{10} \frac{\sigma_x^2}{2\sigma_u^2}$$

$$= SNR_I - 3 \text{ dB} \tag{2.10.6}$$

This result is in agreement with experiment.

By contrast the uniform quantization realisation of a first order zero prescaled to prevent output overflow, see Figure 2.10.2, is found to have an output signal to noise ratio given by:

$$SNR_O = SNR_I - 10 \log_{10} (3 + L^2) \tag{2.10.7}$$

The parameter σ_v^2, in Figure 2.10.2, is given by:

$$\sigma_v^2 = \frac{(2q)^2}{12}$$

where 2q is the quantizer step size.

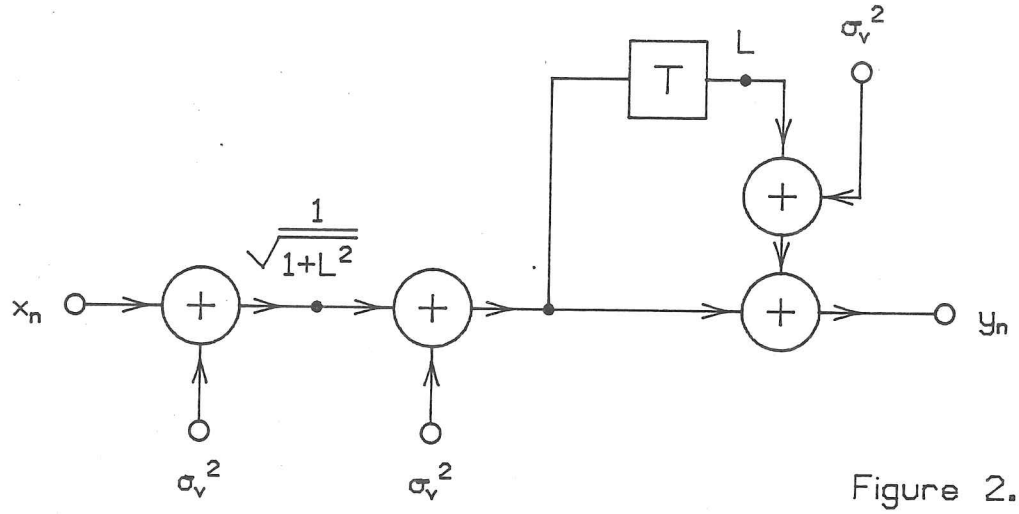


Figure 2.10.2

2.11 First Order Pole-Zero Design

In this section the signal to noise ratio performance for two different realisations of a first order pole-zero will be studied. It will be shown that of those two realisations, the so-called direct and canonic forms, the canonic form unconditionally yields a higher signal to noise ratio.

The direct form (zero first) realisation is considered first; its signal flow graph is shown in figure 2.11.1.

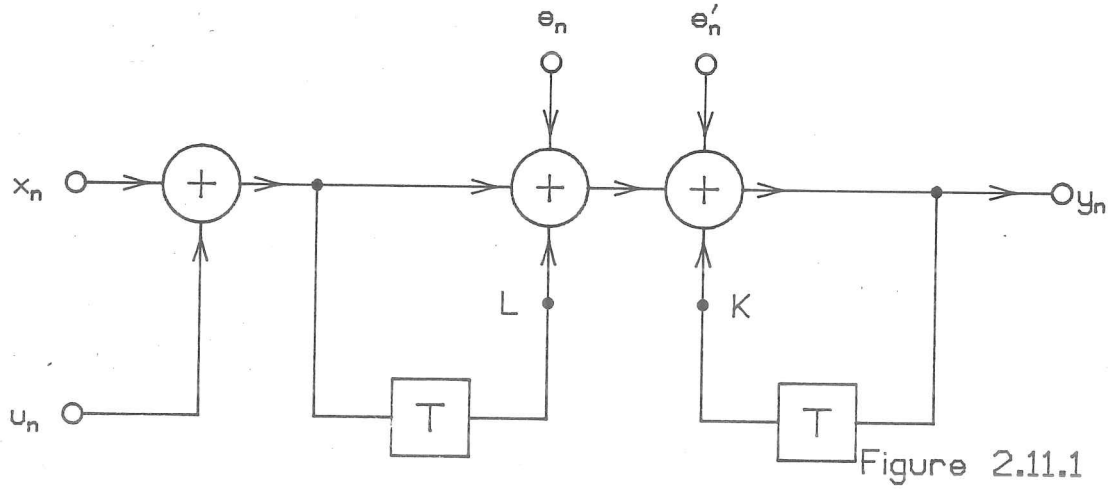


Figure 2.11.1

Each of the noise sources u_n, e_n and e'_n is orthogonal to the signal to which it adds. Additionally, all three are assumed uncorrelated with each other and with the input x_n. In section 2.9 it was shown that the power of noise source e_n is related to the power of u_n by:

$$\sigma_e^2 = \sigma_u^2 (1 + L^2) \tag{2.11.1}$$

The power of noise source e'_n is also related to the power of u_n by:

$$\sigma_{e'}^2 = G\sigma_u^2 \tag{2.11.2}$$

where G is the integrated power gain from x_n to y_n. Now the filters transfer function is:

$$H(z) = \frac{z + L}{z - K} \tag{2.11.3}$$

and so the filters integrated power gain:

$$G = \frac{1}{2\pi j} \oint_c H(z)\, H(z^{-1})\, \frac{dz}{z}$$

where c is the unit circle, is:

$$G = \frac{1}{K} \left[\frac{(K + L)(1 + KL)}{(1 - K^2)} - L \right] \tag{2.11.4}$$

The noise power appearing at the filters output, which will be denoted by σ_{yn}^2, is given by:

$$\sigma_{yn}^2 = G\sigma_u^2 + G\sigma_e^2 + \frac{\sigma_{e'}^2}{1-K^2} \tag{2.11.5}$$

and the signal power at the filters output, denoted by σ_{ys}^2, is given by:

$$\sigma_{ys}^2 = G\sigma_x^2 \tag{2.11.6}$$

and hence the signal to noise ratio is:

$$SNR_o = 10 \log_{10} \left[\frac{G\sigma_x^2}{G(\sigma_u^2 + \sigma_e^2) + \frac{\sigma_{e'}^2}{1-K^2}} \right] \tag{2.11.7}$$

which after substitution and rearrangement gives:

$$SNR_O = SNR_I - 10 \log_{10} \left[\frac{(3 - 2K^2)}{(1 - K^2)} - \frac{2LK}{(1 + 2LK + L^2)} \right] \qquad (2.11.8)$$

Making a similar analysis of the canonic form, Figure 2.11.2, the signal to noise ratio at the output is found to be:

$$SNR_O = SNR_I - 10 \log_{10} \left[\frac{(2 - K^2)}{(1 - K^2)} \right] \qquad (2.11.9)$$

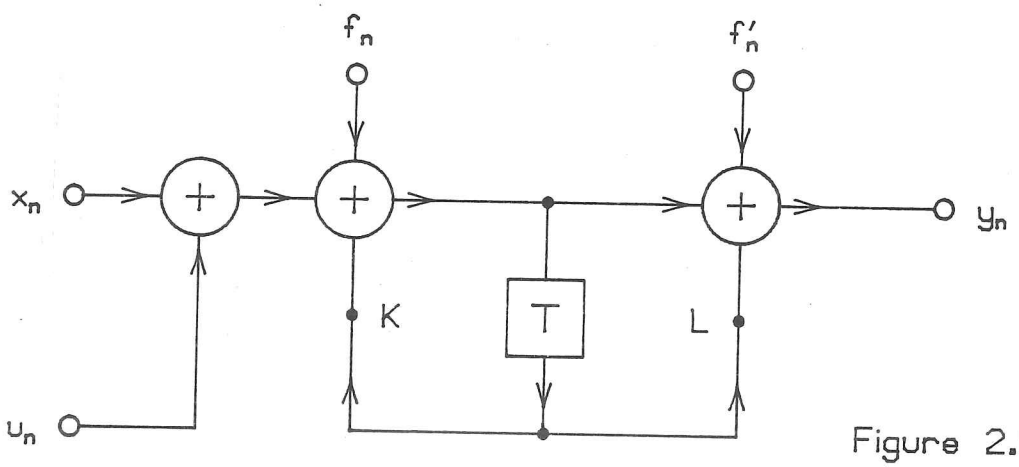


Figure 2.11.2

Comparing equations 2.11.8 and 2.11.9 it can be seen that for all values of K and irrespective of the value of L the canonic form gives a higher output signal to noise ratio. This result has been checked by experiment for a wide range of pole-zero configurations.

2.12 Stability In First-Order Systems

Only recursive systems are capable of becoming unstable and so only a first-order pole will be considered here.

Consider the first order pole whose input x_n and output y_n are related by the difference equation:

$$y_n = Ky_{n-1} + x_n \qquad (2.12.1)$$

Such a system is asymptotically stable provided that its only pole z = K lies inside the unit circle. It follows that the ideal system with all arithmetic performed to unlimited precision is stable if K lies in the closed interval

$$K = [-1, +1] \qquad (2.12.2)$$

In a digital realisation of 2.12.1 K and y_{n-1} are represented by p-bit values and hence their product requires 2p-bits. In order to restore this product to y_n the least significant p-bits must be discarded. There are two ways to do this. The first, Truncation, makes equation 2.12.2 both necessary and sufficient for stability. The second, rounding, makes 2.12.2 insufficient. It is, however, desirable to employ rounding as this leads to a less noisy realisation. The instability which results from rounding manifests itself as a, hopefully, low level oscillation when K is negative and a constant value when K is positive. This instability which is referred to as a limit cycle is said to result from the deadband effect.

The limit cycle amplitude $|N|q$ which can be supported by the filter in 2.12.1 is given by the standard result:

$$|N|q \leq \frac{0.5}{1 - |K|} q \qquad (2.13.3)$$

where q is the quantization step size.

No general analytical result has been found for the free quantization law first-order filter. There are, however, two ways in which progress can be made with the gaussian optimised filter.

In section 2.2 the decision and quantization levels for a gaussian optimal quantizer were found. From these it is possible to ascertain for a given value of K the maximum limit cycle amplitude. This is done by noting that for limit cycling the following inequality holds:

$$di \leq K q_i \qquad (2.12.4)$$

where q_i is the quantization level representing the previous filter output and d_i is the nearest smaller decision level. When inequality 2.12.4 holds K is said to have an effective absolute value of one.

From 2.12.4 it follows that for a limit cycle of amplitude, q_i, K is bounded by:

$$1 > K \geq \frac{d_i}{q_i} \qquad (2.12.5)$$

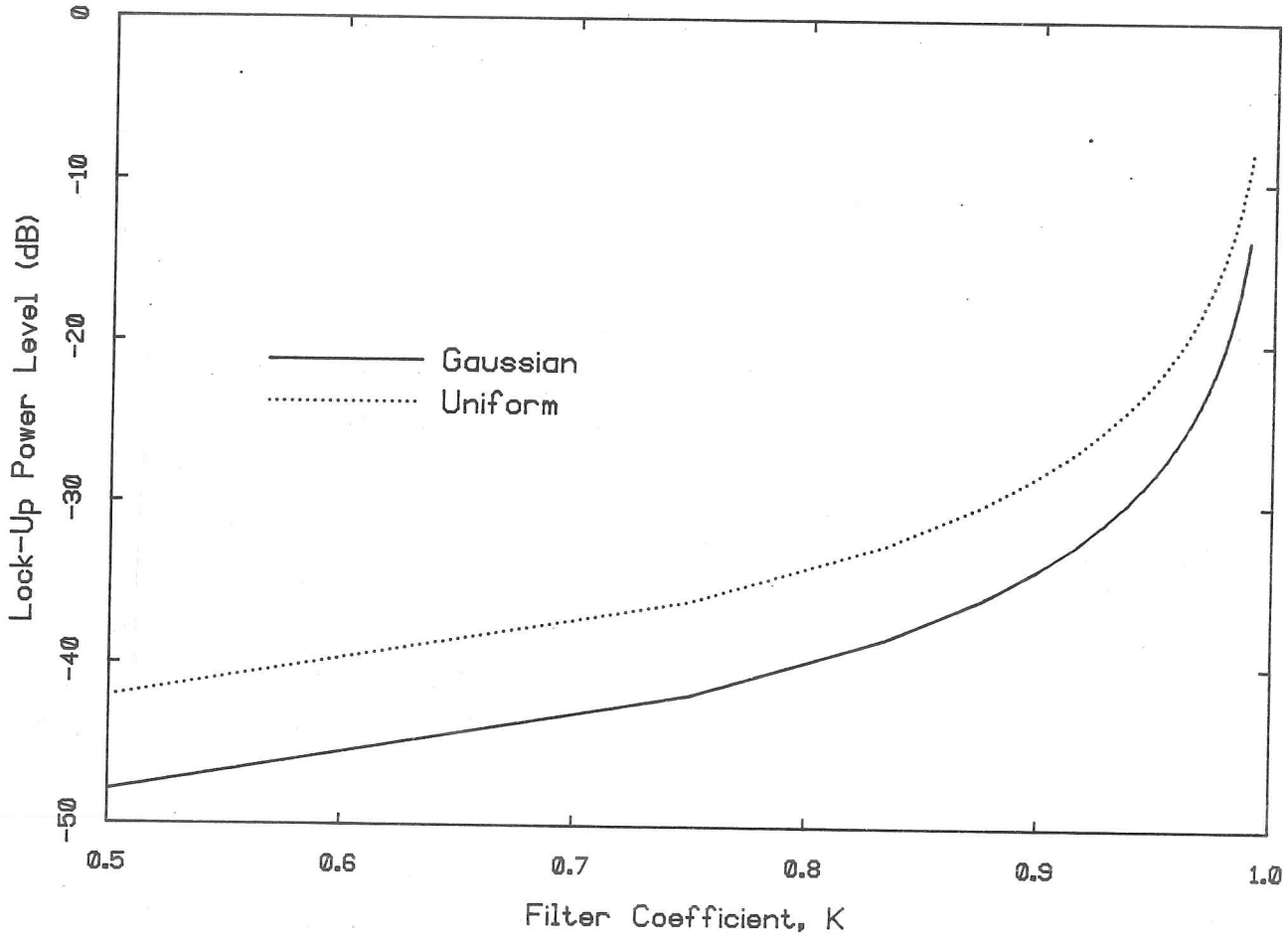
The graph in Figure 2.12.1 shows limit-cycle power level in dB, relative to the quantizer saturation level, as a function of coefficient K. The gaussian optimised filter is seen to be some 5 dB quieter for a given value of K.

A second approach is to consider the gaussian optimal quantizer to be piecewise linear in the region where limit cycles will occur. Referring to Figure 2.2.2 this is seen to be true for approximately 100 levels centred around q_{128}. The effective step size is found to be 0.01682. The optimum step size for a 255 level uniform quantizer applied to a gaussian process is found to be 0.03086. The limit cycle power level is therefore 5.3 dB lower in the case of the gaussian optimised filter.

2.14 First Order All-Pole - Frequency Response

It is well known that the effect of parameter quantization in a digital filter is to cause that filter to have a frequency-phase response different from the design specification. A further complicating factor is that a filters response to a sinusoid differs with the peak to peak amplitude of that sinusoid. The exact interaction between parameter quantization, signal amplitude and filter frequency response defies exact analysis.

While studying gaussian optimised high order filters it has been observed that, compared with similar uniform quantization law filters, their frequency response is closer to that of the ideal system. Despite the difficulties of analysis sighted above it is essential to find a justification for this observation.



First order filter lock-up power level
as a function of the filter coefficient K
for gaussian and uniform quantization laws.

Figure 2.12.1

In this section a statistical treatment of the problem is presented for the case of a first order all-pole.

Consider the filter described by the difference equation:

$$y_n = [Ky_{n-1}]' + x_n \tag{2.13.1}$$

where x_n and y_n are members of a discrete finite set, the set of quantization levels, and $[\ ]'$ is the operation of re-quantization.

As it is the frequency response which is being investigated only the natural response of the filter need be considered. Let the value of y_{n-1} be q_i the i th. quantization level. This maps after multiplication and re-quantization to $[Kq_i]'$. In the equivalent ideal system the apparent value of K corresponding to quantization level q_i and denoted by K_i is given by:

$$K_i = \frac{[Kq_i]}{q_i} \tag{2.13.2}$$

Now, if q_i is bracketed by decision levels d_i and d_{i+1} and assuming gaussian signal statistics the probability of occurence of quantization level q_i, denoted by p_i, is:

$$p_i = \int_{d_i}^{d_{i+1}} \phi(x)\, dx \tag{2.13.3}$$

where $\phi(x)$ is the gaussian probability density function.

It follows that in the long term the expected value of K_i denoted by K' is given by:

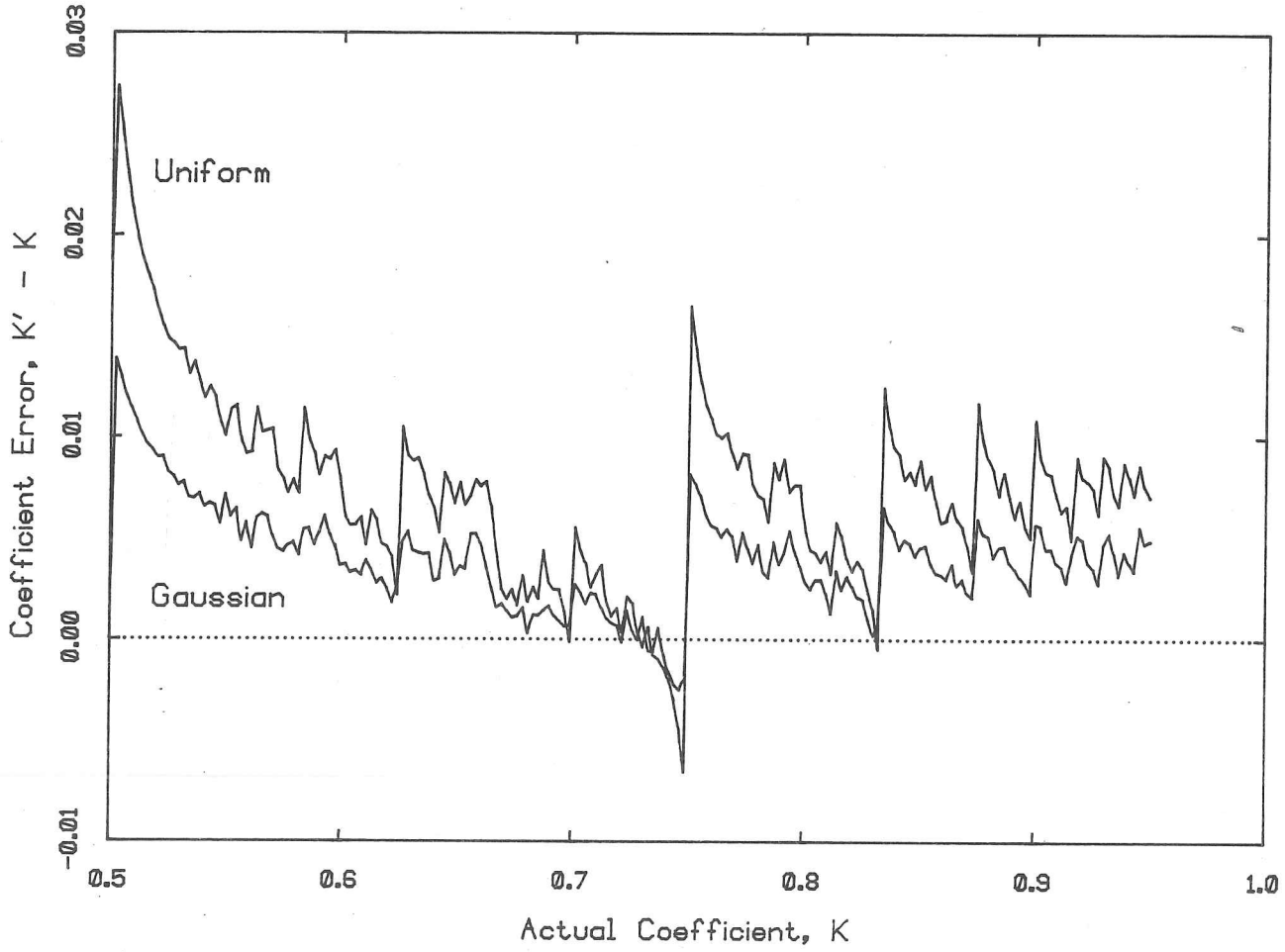
$$K' = \sum_{i=1}^{N} p_i K_i$$

$$= \sum_{i=1}^{N} \left[\frac{[Kq_i]'}{q_i} \int_{d_i}^{d_{i+1}} \phi(x)\, dx \right] \tag{2.13.4}$$

This analysis has assumed that a system with a pole a z = K is still representable by a single pole at z = K'. If K' is different from K, then, of course, a change in the frequency response has occured. The name 'expected'

rather than 'effective' coefficient will be given to K'. This is to avoid confusion with the analysis of limit cycles presented by Jackson [42].

A graph of K'-K versus K is shown in figure 2.13.1. Both the uniform and the gaussian optimal quantizer are considered. It is clear that for all but a small set of values of K the gaussian filter compared with the uniform filter has an expected pole nearer to that of the ideal system.



Coefficient error v. actual coefficient
for a first order all-pole

Figure 2.13.1

3. Second Order Free Quantization Law Digital Filters

3.1 Introduction

A digital filter implemented using rounding type fixed point arithmetic is subject to the deadband effect. The number of limit cycle modes becomes too large to analyse in filters of order 3 or higher. In addition the maximum possible limit cycle amplitude becomes comparable to the signal amplitude as the filter order is increased. For these reasons the direct form implementation of filters of order 3 or above is never used; instead either a cascade or parallel composition of sub-filters is used, each sub-filter being of order 1 or 2. In this chapter free quantization law design techniques are applied to a second order pole-zero section. Comparisons are drawn between free and uniform quantization law implementations of second order structures.

3.2 Second Order All-Pole - Design

The filter to be considered is a second order all-pole with input x_n and output y_n related by the difference equation:

$$y_n = K_1 y_{n-1} + K_2 y_{n-2} + x_n \qquad (3.2.1)$$

The corresponding signal flow graph is shown in figure 3.2.1. This configuration is not the only one possible. In principle the operations $x_n + K_1 y_{n-1} + K_2 y_{n-2}$ could be combined into one mapping $f(x_n, y_{n-1}, y_{n-2}) \rightarrow y_n$; however, this will be seen in the next chapter to lead to an intractable implementation problem. It is for this reason that the variable v_n has been introduced.

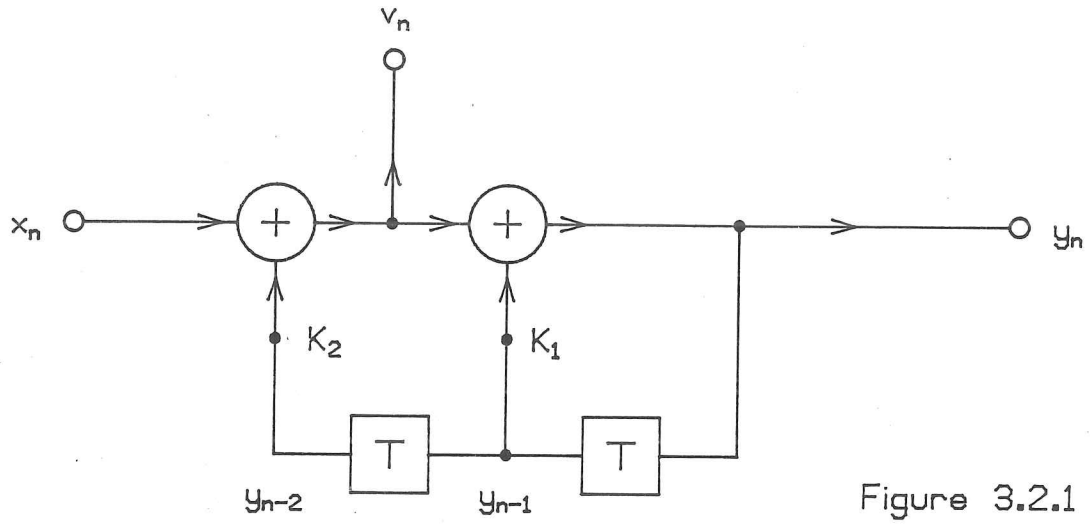


Figure 3.2.1

The filter described by equation 3.2.1 will be asymptotically stable provided that p and p*, the conjugate roots of:

$$z^2 - K_1 z - K_2 = 0 \tag{3.2.2}$$

lie inside the z-plane unit circle. For convenience p will be expressed in polar form as:

$$p = re^{j\theta} \tag{3.2.3a}$$

where $r = \sqrt{-K_2}$ (3.2.3b)

and $\theta = \cos^{-1}\sqrt{\frac{-K_1^2}{4K_2}}$ (3.2.3c)

The design of the filter in figure 3.2.1 proceeds as follows. Assume x_n is a zero mean, guassian and white process of variance σ_x^2. Since x_n is zero mean it follows that v_n and y_n are zero mean. The variance of processes v_n and y_n, denoted by σ_v^2 and σ_y^2 respectively, can be found from the complex convolution theorem; their values are:

$$\sigma_y^2 = \sigma_x^2 \frac{1 + r^2}{1 - r^2} \frac{1}{1 - 2r^2\cos 2\theta + r^4} = G_y \sigma_x^2 \tag{3.2.4}$$

and

$$\sigma_v^2 = \sigma_y^2 - \sigma_x^2 \frac{4r^2\cos 2\theta}{1-2r^2\cos 2\theta+r^4} \qquad (3.2.5)$$

$$= G_v \sigma_x^2$$

The optimum quantizers for x_n, v_n and y_n are all similar in shape to the one shown in figure 2.2.2; the scale factors by which each of the quantization and decision levels are multiplied are given, for each variable, in figure 3.2.2.

Signal variable	Variance	Scale factor
x_n	$\sigma_x^2 = 1$	1
v_n	$\sigma_v^2 = G_v\sigma_x^2$	$\sqrt{G_v}$
y_n	$\sigma_y^2 = G_y\sigma_x^2$	$\sqrt{G_y}$

figure 3.2.2

3.3 Second Order All-Pole - Error Model

A second order all-pole filter, which is described by the difference equation

$$y_n = 2r\cos\theta y_{n-1} - r^2 y_{n-2} + x_n \qquad (3.3.1)$$

where x_n and y_n are the input and output respectively,

r is the pole radius

and θ is the pole angle,

is shown, together with noise sources appropriate to a gaussian optimised implementation, in the signal flow graph in figure 3.3.1.

Using the same notation as in the previous section, if noise process u_n has variance σ_u^2 then noise processes e_n and $\grave{e}_n$ have variances

$$\sigma_e^2 = G_v\sigma_u^2 \qquad (3.3.2a)$$

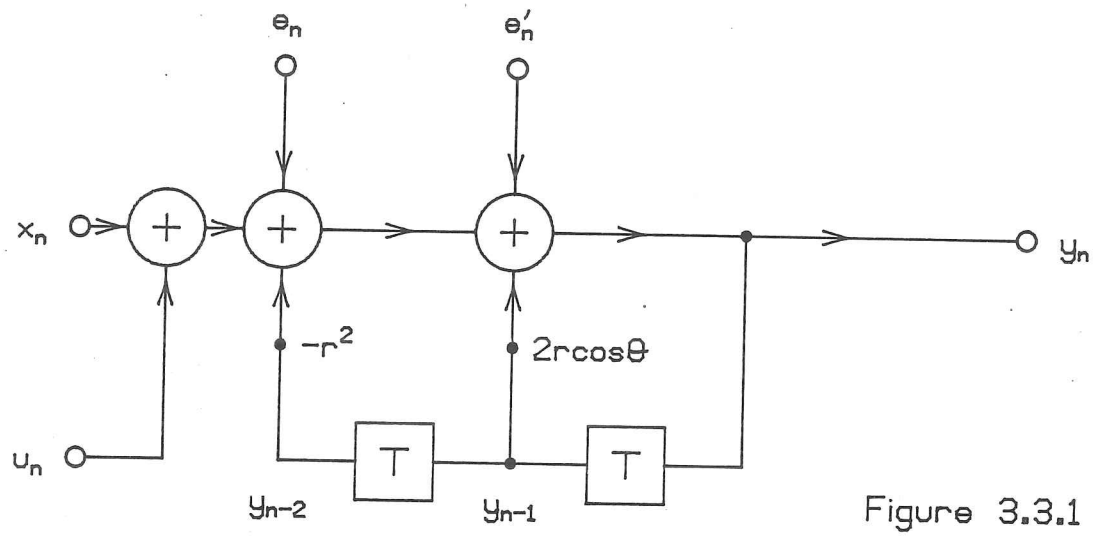


Figure 3.3.1

and

$$\sigma_{\dot{e}}^2 = G_y \sigma_u^2 \qquad (3.3.2b)$$

The noise which appears at the output of the filter is given by:

$$\sigma_{yn}^2 = G_y \left[\sigma_u^2 + \sigma_e^2 + \sigma_{\dot{e}}^2 \right] \qquad (3.3.3)$$

$$= G_y \sigma_u^2 \left[1 + G_v + G_y \right]$$

The signal power at the output of the filter, denoted by σ_{ys}^2, is given by:

$$\sigma_{ys}^2 = G_y \sigma_x^2 \qquad (3.3.4)$$

and hence the signal to noise ratio at the output of the filter is:

$$SNR_O = 10 \log_{10}\left[\frac{G_y \sigma_x^2}{G_y \sigma_u^2 \left[1 + G_v + G_y \right]} \right] \text{ dB} \qquad (3.3.5)$$

$$= 10 \log_{10}\left[\frac{\sigma_x^2}{\sigma_u^2} \frac{1}{1 + G_v + G_y} \right] \text{ dB}$$

$$= SNR_I - 10 \log_{10} (1 + G_v + G_y) \text{ dB}$$

where SNR_I is recognised as the input signal to noise ratio.

The validity of this model has been experimentally confirmed. The graph in figure 3.3.3. shows both experimental and theoretical SNR measurements for a wide range of values of r and θ. The signal to noise ratio performance of the uniform quantization law filter, whose error model signal flow graph appears in figure 3.3.2, is shown in figure 3.3.4; also included on this graph is the theoretical SNR performance of the gaussian optimised filter illustrating its improved performance.

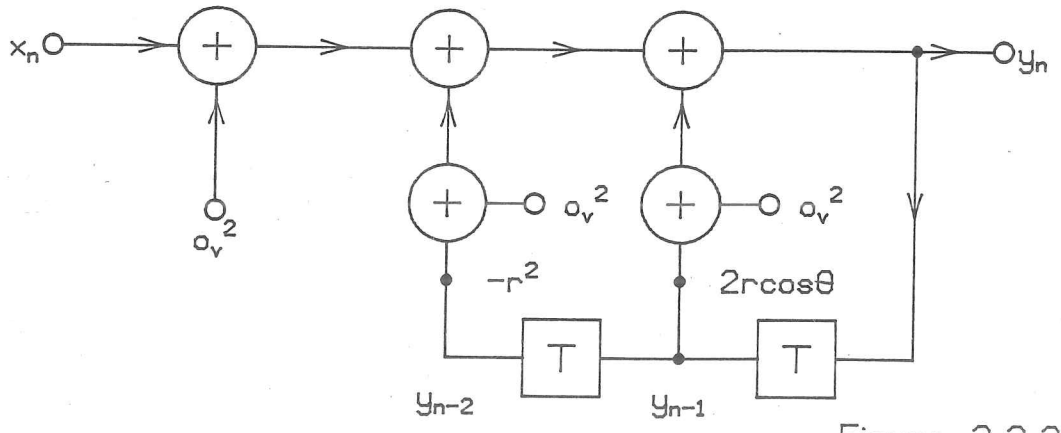
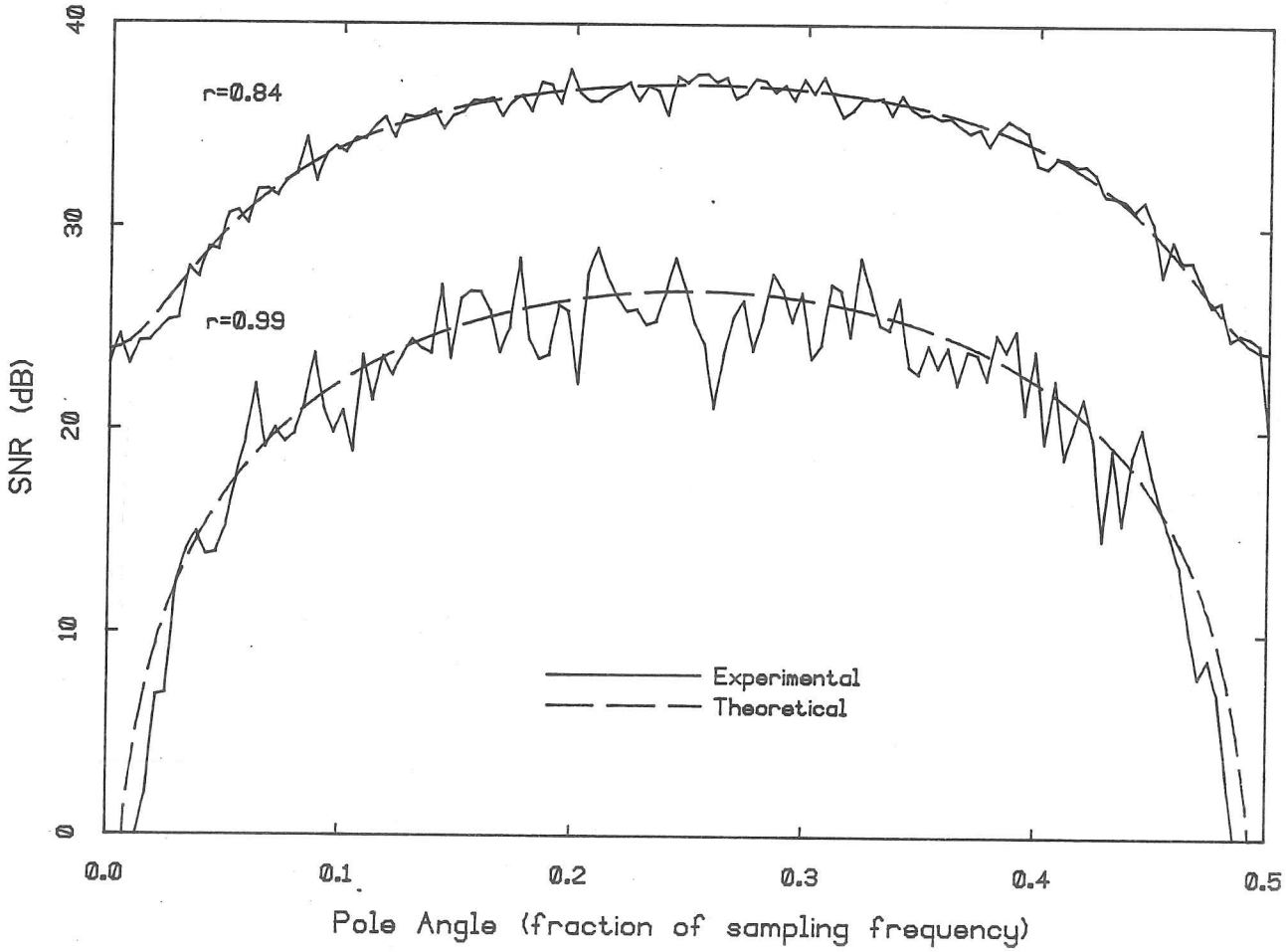
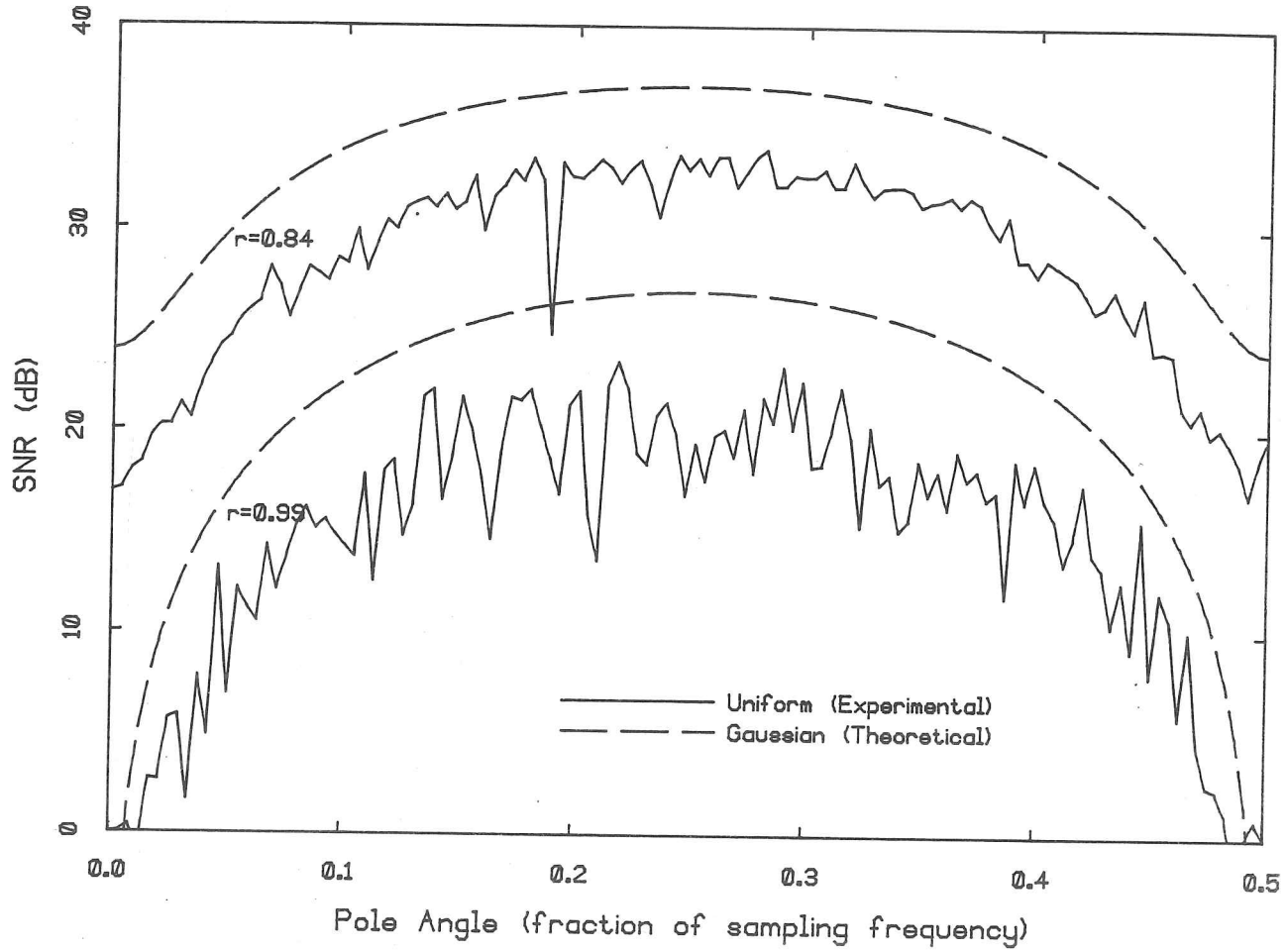


Figure 3.3.2



Experimentally observed and theoretically predicted SNR performance of a gaussian optimised second order all-pole

Figure 3.3.3



Experimentally observed signal to noise ratio performance of a uniform quantization law second order all-pole compared with the theoretical performance of a gaussian optimised second order all-pole.

Figure 3.3.4

3.4 Second Order All-Zero - Design

The signal flow graph for the second order all-zero described by the difference equation:

$$y_n = x_n - 2s\cos\phi x_{n-1} + s^2 x_{n-2} \tag{3.4.1}$$

is shown in figure 3.4.1.

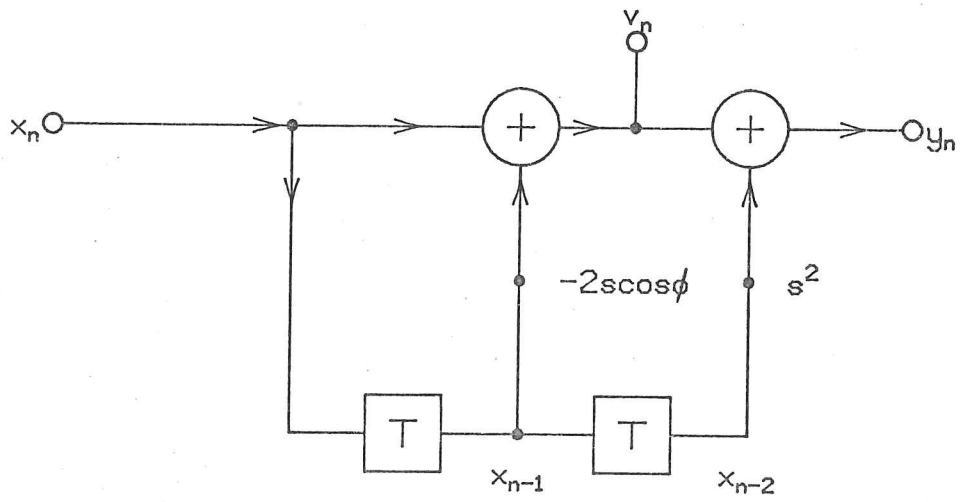


Figure 3.4.1

The extra signal variable v_n is introduced to seperate the adders thereby making a three to one mapping into a pair of two to one mappings.

The purpose of the following analysis is to establish the signal variance at variables v_n and y_n given that x_n is gaussian white of variance σ_x^2 and mean 0.

In the time domain v_n and x_n are related by:

$$v_n = x_n - 2s\cos\phi x_{n-1} \tag{3.4.2}$$

and so the variance of v_n, denoted by σ_v^2, is given by:

$$\sigma_v^2 = E\left[v_n^2 \right] = E\left[x_n^2 - 4s\cos\phi x_n x_{n-1} + 4s^2\cos^2\phi x_{n-1}^2 \right] \tag{3.4.3a}$$

$$= \sigma_x^2 \left[1 + 4s^2\cos^2\phi \right] \tag{3.4.3b}$$

where E [ ] is the expectation operator. For examples of this kind, FIR filters, the use of the complex convolution theorem is unnecessary.

By a similar derivation the value of σ_y^2, the variance of y_n, is obtained:

$$\sigma_y^2 = \sigma_x^2 \left[1 + 4s^2\cos^2\phi + s^4 \right] \qquad (3.4.4)$$

The quantizers employed at nodes v_n and y_n will be similar in shape to the one at x_n but their levels will be larger in magnitude by the multiplying factors in figure 3.4.2.

Variable	Scale factor
v_n	$\sqrt{(1 + 4s^2\cos^2\phi)}$
y_n	$\sqrt{(1 + 4s^2\cos^2\phi + s^4)}$

figure 3.4.2

3.5 Second Order All-Zero - Error Model

The signal flow graph to be used in the following analysis is the same as the one in figure 3.4.1 with three noise sources introduced; it is shown in figure 3.5.1.

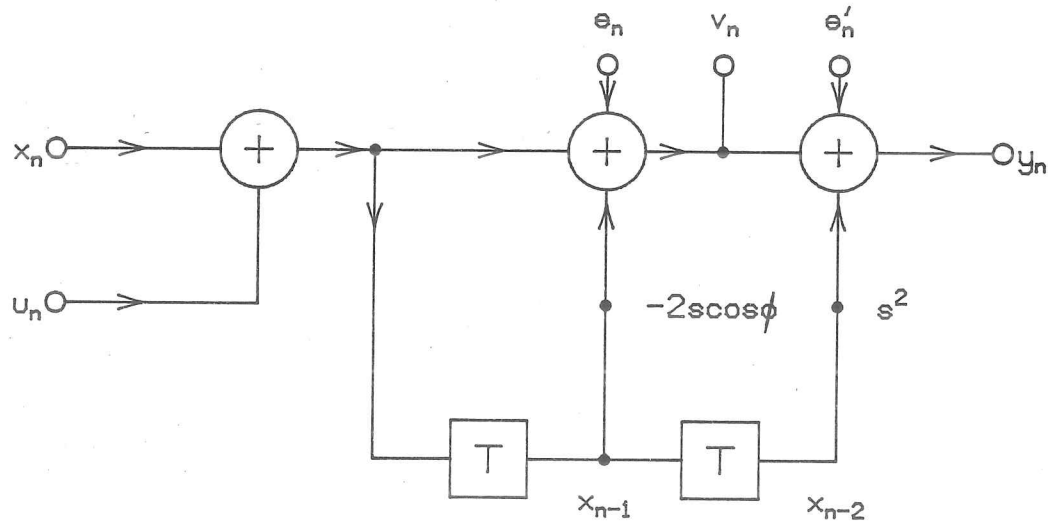


Figure 3.5.1

The power level of noise processes e_n and e_n' as a function of the input power and zero positions are given by:

$$\sigma_e^2 = \sigma_u^2 \left[1 + 4s^2\cos^2\phi \right] \tag{3.5.1}$$

$$\sigma_{e'}^2 = \sigma_u^2 \left[1 + 4s^2\cos^2\phi + s^4 \right] \tag{3.5.2}$$

It follows, from equations 3.5.1 and 3.5.2 that the output noise power σ_{yn}^2 is given by:

$$\sigma_{yn}^2 = \sigma_u^2 \left[1 + 4s^2\cos^2\phi + s^4 \right] + \sigma_u^2 \left[1 + 4s^2\cos^2\phi \right]$$

↑ due to noise at variable x_n ↑ due to noise at variable v_n

$$+ \sigma_u^2 \left[1 + 4s^2\cos^2\phi + s^4 \right] \tag{3.5.3}$$

↑ due to noise at variable y_n

Now, the signal power at the output, denoted by σ_{ys}^2, is given by:

$$\sigma_{ys}^2 = \sigma_x^2 \left[1 + 4s^2\cos^2\phi + s^4 \right] \tag{3.5.4}$$

and hence the output signal to noise ratio is given by:

$$SNR_O = SNR_I - 10 \log_{10} \left[\frac{2s^4 + 3(1 + 4s^2\cos^2\phi)}{1 + 4s^2\cos^2\phi + s^4} \right] \text{ dB} \tag{3.5.5}$$

where SNR_I is the input signal to noise ratio.

From equation 3.5.5 it can be seen that $SNR_O - SNR_I$, the loss in signal to noise ratio, is bounded by the inequality:

$$0 > SNR_O - SNR_I > -4.77 \text{ dB} \tag{3.3.6}$$

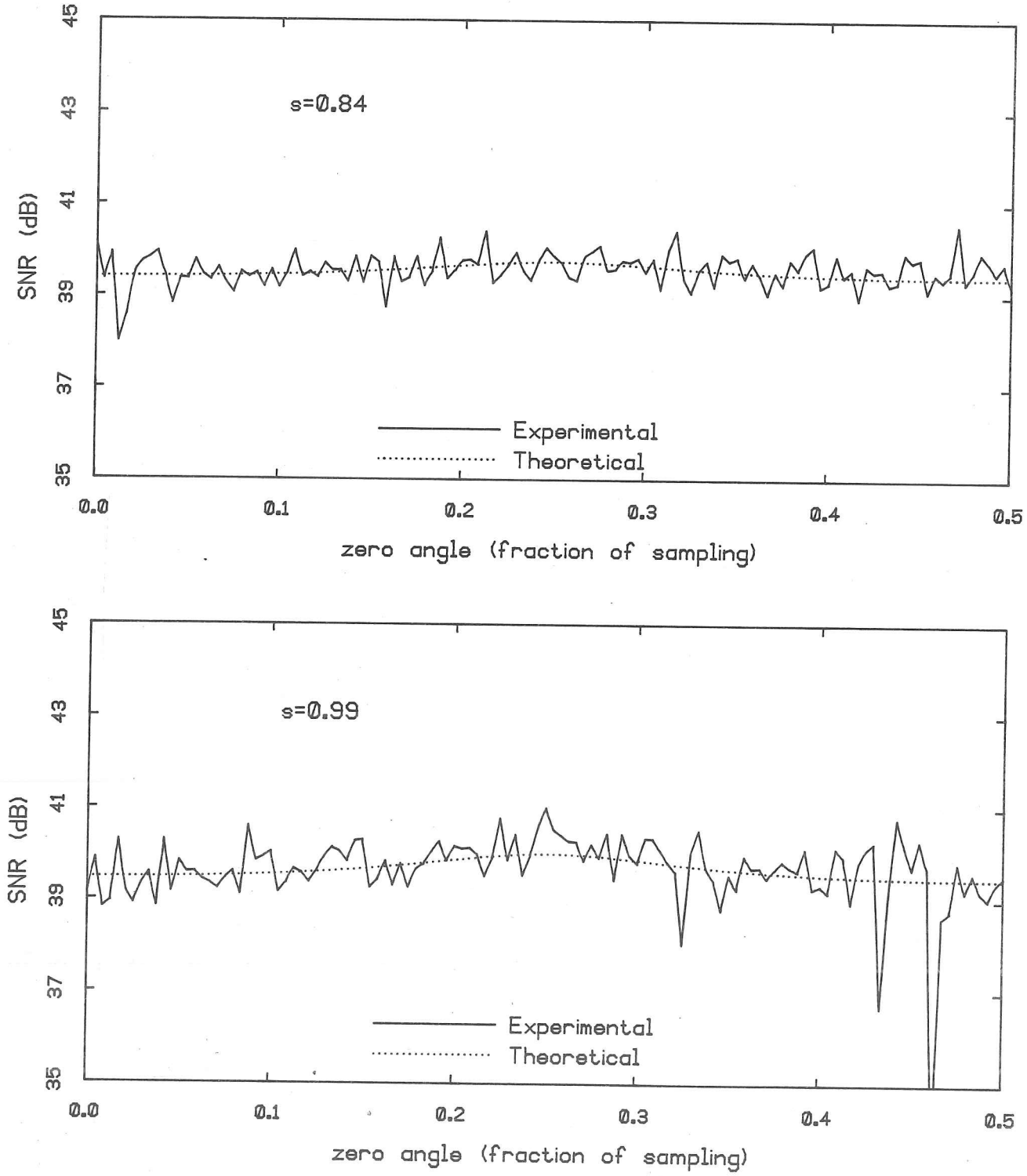
An experiment was performed to check the validity of equation 3.5.5; the graph in figure 3.5.2 shows measured signal to noise ratio for two values of zero radius, s, and in each case for 121 values of ϕ. The dotted lines are the theoretical SNR estimates.

3.6 Second Order Pole-Zero (Section)

As with the first order pole-zero (section), there are a number of different signal flow graph topologies which will lead to the desired second order transfer function. It has been observed by other workers that different topologies, which realise the same transfer function, do not necessarily give the same signal to noise ratio performance or the same frequency-phase response accuracy. In section 2.11 an analytical result was obtained; this result showed that a canonic form implemention of a first order free quantization law p-z section gives a higher SNR performance than the zero first implementation.

A similar analysis to that in section 2.11 is in principle possible for the case of a second order p-z section; however, it is lengthy and does not lead to an obvious rule. To overcome this difficulty a random test was performed; the details of this test are as follows. A pair of conjugate poles and a pair of conjugate zeroes were randomly placed inside the z-plane unit circle. The section was then implemented using both the zero first and canonic form implementation, in both cases the signal to noise ratio was calculated. In all tests the canonic form gave an SNR performance equal to or better than the zero first form.

$$H(z) = \frac{z^2 + L_1 z + L_2}{z^2 + K_1 z + K_2}$$



Experimentally observed and theoretically predicted SNR performance for a gaussian optimised second order all-zero. Measurements taken for two values of zero radius and for 121 values of zero angle

Figure 3.5.2

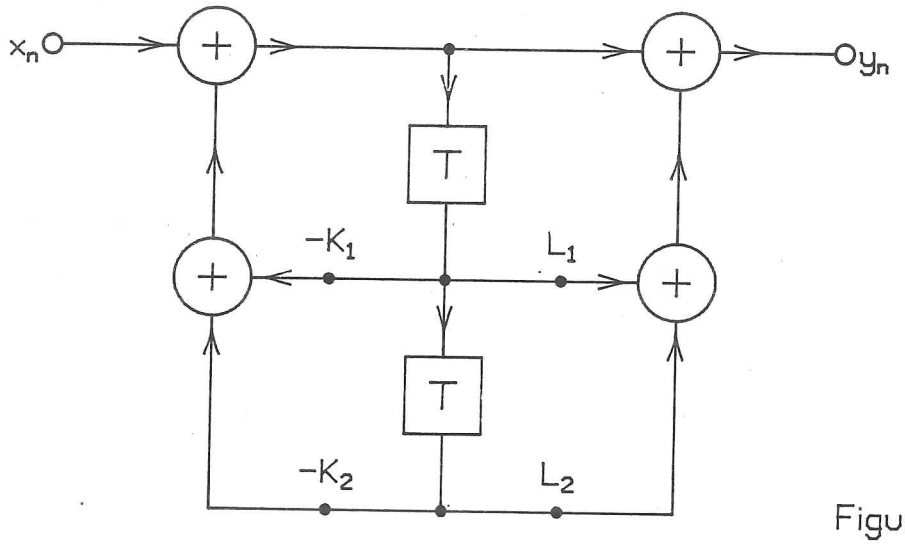


Figure 3.6.1

3.7 Section Ordering and Pole-Zero Pairing

In fixed point implementations of high order (order > 2) recursive digital filters it is usual to decompose the filter into either a cascade or a parallel connection of first and second order sections (sub-filters). With both decompositions a pairing of poles and zeroes has to be chosen, and with the cascade decomposition a section ordering has to be chosen. It is known that both ordering and pole-zero pairing can have an effect on the output signal to noise ratio performance of a filter; in principle at least the filter configuration which gives the maximum output signal to noise ratio can be found by exhaustive search of all possible configurations. For a filter constructed from N cascaded sections there are N ! section orderings and N ! pole-zero pairings and hence $N!^2$ configurations. For filters of order eight or less ($N \leqslant 4$) an exhaustive search is practical since $N!^2 \leqslant 576$; however, for N = 5 the number of possible configurations being 14400 is becoming too large to admit an exhaustive search.

There is no known analytical solution to the problems of section ordering and pole-zero pairing, however, various heuristic and some optimisation solutions have been proposed [30,91,93].

In this section it will be shown that the output signal to noise ratio of a cascade form free quantization law filter is essentially

independent of sub-filter ordering and pole-zero pairing. These results were obtained experimentally; the details are as follows:

Section Ordering

Two pairs of poles were chosen at random; these will be denoted by A and B. The radii and angles were obtained using uniform random number generators in the range [0,1] and [0,π] respectively. Only those trials with A and B further than a certain minimum distance apart were considered. This was done to avoid numerical instability in the computation of the

$$\frac{1}{A-B} \tag{3.7.1}$$

factors appearing in the contour integration. Two filters were then implemented; the first was a cascade connection of two second order all-poles with pole pair A followed by pole pair B, and the second was the converse. The respective signal to noise ratios SNR_{AB} and SNR_{BA} were then calculated using the theoretical noise model. The statistics of

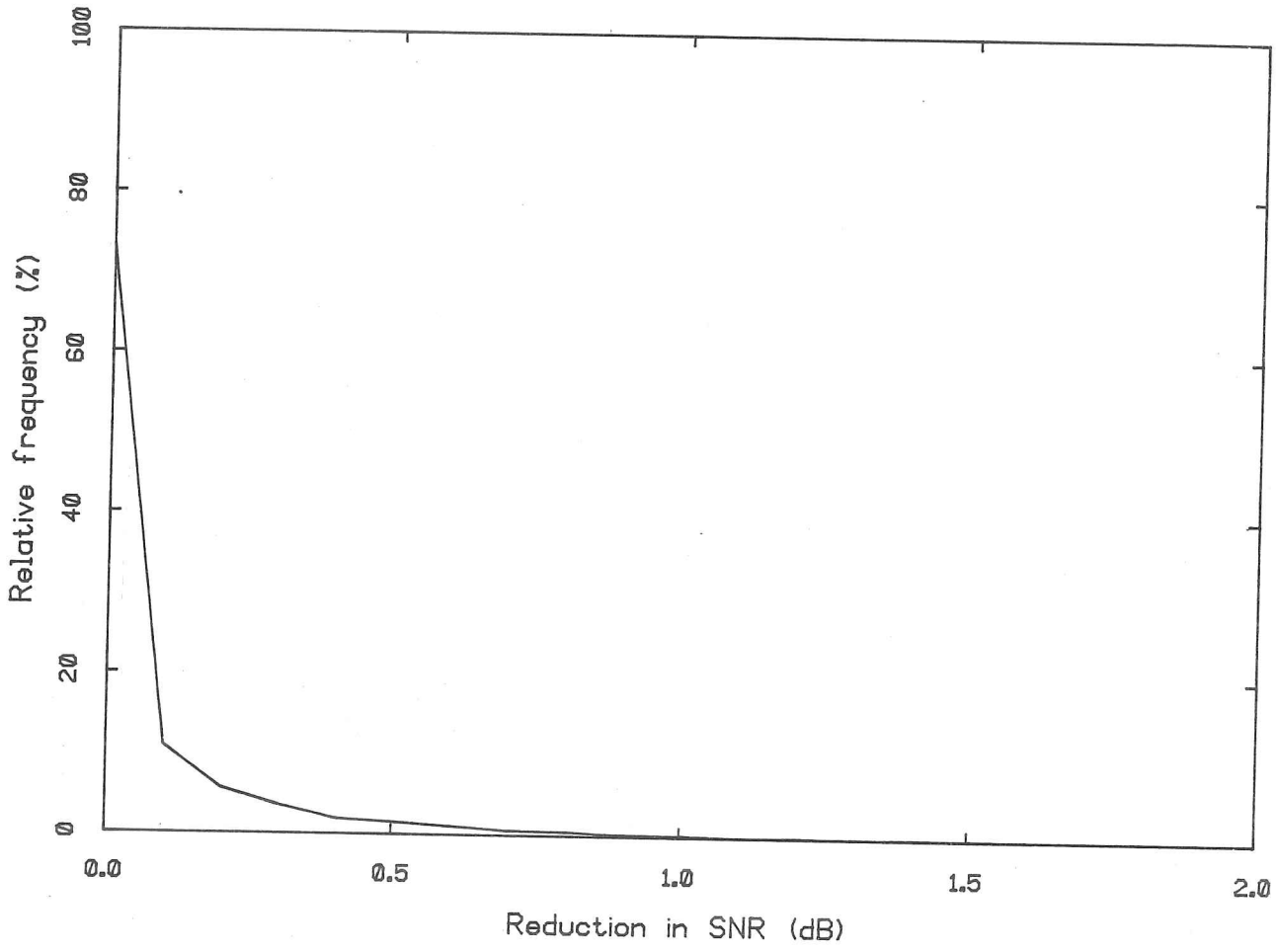
$$SNR_E = SNR_{AB} - SNR_{BA} \tag{3.7.2}$$

were found to be

mean value	=	0.000 dB
standard deviation	=	0.219 dB
maximum value	=	1.600 dB

The histogram in figure 3.7.1 gives a coarse indication of the frequency of various bands of error values. The maximum error in choosing the wrong ordering is seen to be insignificant.

For each trial performed above, the SNR for the equivalent fixed point arithmetic filter was calculated; in all cases the wrongly ordered gaussian optimised filter gave a higher signal to noise ratio than the fixed point arithmetic filter.



Histogram showing relative frequency of various reductions in SNR caused by the selection of a wrong section ordering. Tests performed on a cascade of two second order all-poles; each conjugate pole pair chosen at random. Total of 10000 trials performed.

Figure 3.7.1

Pole-Zero Pairing

Two pole pairs A and B and two zero pairs C and D were selected at random. Four filters were implemented, each using a cascade of two second order sections. The filters had the following pole-zero pairings:

Filter	1st section		2nd section	
	pole	zero	pole	zero
1	A	C	B	D
2	A	D	B	C
3	B	C	A	D
4	B	D	A	C

figure 3.7.2

For each of the four filters the SNR, denoted by SNR_i $(1 \leqslant i \leqslant 4)$, was calculated. The statistics of parameter :

$$SNR_E = \left| \max_i SNR_i - \min_i SNR_i \right| \qquad (3.7.3)$$

were found to be as follows:

mean value = 0.000 dB

standard deviation = 0.321 dB

maximum value = 2.100 dB

The histogram was found to have the same shape as the one in figure 3.7.1. It can be seen that the arbitrary choice of a pole-zero pairing is unlikely to yield an SNR which is significantly lower than the optimum.

3.8 Stability Considerations

The filter described by the constant coefficient difference equation:

$$y_n = x_n + K_1 y_{n-1} + K_2 y_{n-2} \qquad (3.8.1)$$

has the z-domain transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - K_1 z^{-1} - K_2 z^{-2}} \qquad (3.8.2)$$

The poles of H(z) are both real if $K_1^2 \geq 4K_2$, and are conjugate complex if $K_1^2 < 4K_2$ in which case they are of the form:

$$z = re^{\pm j\theta} \qquad (3.8.3)$$

where $r = \sqrt{-K_2}$

and $\theta = \cos^{-1} \sqrt{\frac{-K_1^2}{4K_2}}$

When the filter described by equation 3.8.1 is implemented using finite precision arithmetic it is possible for y_n to exhibit sustained oscillations in the absence of an input signal. These oscillations are referred to as limit cycles. In the analysis presented by Jackson [42] two kinds of limit cycle are shown to exist. The first results from a real effective pole at $z = +1$ or $z = -1$; the second results from conjugate complex effective poles at $z = e^{\pm j\theta}$.

In the case of a real effective pole Jackson shows that for a fixed point arithmetic digital filter an upper bound on the limit cycle amplitude is given by:

$$|y_n| \leq \frac{0.5}{1 - |K_1| - K_2} \qquad (3.8.4)$$

A similar analysis will now be applied to a free quantization law all-pole.

Consider the signal flow graph in figure 3.8.1.

It is assumed that the limit cycles are constrained to a region of the (y_n, y_{n-1}, y_{n-2}) space where the quantization laws at p_n and y_n are piecewise linear. The parameters q_x, q_p and q_y are the effective step sizes at x_n, p_n and y_n respectively. The values of q_x, q_p are related by equations 3.2.4 and 3.2.5 as follows:

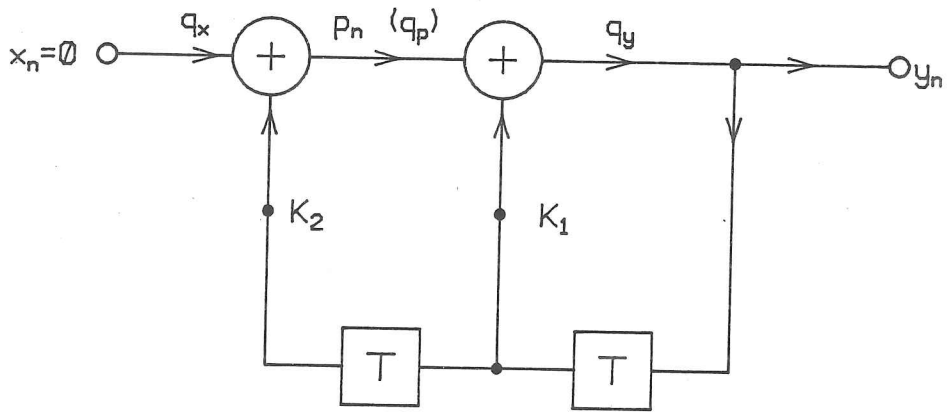


Figure 3.8.1

$$q_p = q_x \sqrt{\frac{\sigma_p^2}{\sigma_x^2}} \tag{3.8.5}$$

$$q_y = q_x \sqrt{\frac{\sigma_y^2}{\sigma_x^2}} \tag{3.8.6}$$

The difference equation which describes the autonomous behaviour of the filter in figure 3.8.1 is

$$y_n = \left[K_1 y_{n-1} + \left[K_2 y_{n-2} \right]^p \right]^y \tag{3.8.7}$$

where $\left[\quad \right]^p$ and $\left[\quad \right]^y$ denote quantization to an allowed value of p_n and y_n respectively.

Given that:

$$\left[A \right]^p = A + D_p \tag{3.8.8a}$$

and

$$\left[A \right]^y = A + D_y \tag{3.8.8b}$$

where $D_p \leq 0.5q_p$ and $D_y \leq 0.5q_y$ equation 3.8.7 becomes:

$$y_n = K_1 y_{n-1} + K_2 y_{n-2} + D_p + D_y \qquad (3.8.9)$$

Now, if equation 3.8.9 has a real effective pole at either z = + 1 or z = -1 then the response y_n is either:

$$y_n = Y \qquad (3.3.10a)$$

or

$$y_n = Y\,(-1)^n \qquad (3.8.10b)$$

where Y is a constant.

Substituting equations 3.8.10 for y_n in equation 3.8.9 and using the inequalities for D_p and D_y, the upper bound on the limit cycle amplitude is obtained; it is given by:

$$|Y| \leq \frac{0.5}{1 - |K_1| - K_2}\,(q_p + q_y) \qquad (3.8.11)$$

In order to compare this result with a fixed point arithmetic filter it is necessary to divide the bound by a factor $1/q_y$ giving:

$$|Y'| \leq \frac{0.5}{1 - |K_1| - K_2}\left[1 + \frac{q_p}{q_y}\right] \qquad (3.8.12)$$

From equations 3.8.5, 3.8.6 and 3.2.5 it can be seen that:

$$\frac{q_p}{q_y} \leq 1$$

and hence the gaussian optimised second order all-pole is capable of supporting limit cycles which are twice as large as in the fixed point arithmetic counterpart.

The second case to be examined is when equation 3.8.9 has complex conjugate effective poles at $z = e^{\pm j\theta}$. In this case equation 3.8.9 may

be written as:

$$K_1 y_{n-1} + K_2 y_{n-2} + D_p + D_y = \hat{K_1} y_{n-1} + \hat{K_2} y_{n-2} \qquad (3.8.13)$$

where $\hat{K_1}$ and $\hat{K_2}$ are effective coefficient values. Since the filter has complex conjugate effective poles on the z-plane unit circle it follows that $K_2 = 1$; it further follows that:

$$y_{n-2} (1 - K_2) = (K_1 - \hat{K_1}) y_{n-1} + D_p + D_y \qquad (3.8.14)$$

Assuming $K_1 - \hat{K_1}$ to be small, equation 3.8.14 reduces to:

$$\left| y_{n-2} \right| \leq \frac{0.5}{1 - K_2} (q_p + q_y) \qquad (3.8.15)$$

After normalisation the upper bound on the limit cycle amplitude is found to be:

$$\left| \hat{Y} \right| \leq \frac{0.5}{1 - K_2} \left[1 + \frac{q_p}{q_y} \right] \qquad (3.8.16)$$

Again, this is up to twice as large as in the fixed point arithmetic case.

4. The Implementation of Free Quantization Law Digital Filters

4.1 Introduction

Digital filters are usually implemented using binary adders, multipliers and storage registers selected from an appropriate logic family. With this implementation method the binary codes held in the storage registers bear some simple relationship to the value of the signal samples which they represent. The two's complement coding scheme is an example; here the code held in a register gives the sample value as an integer multiple of the quantization step size. It follows that the quantization law used in the filter is either uniform or piecewise uniform; the latter accounts for the cases of floating and block floating point. As free quantization law filters can, by definition, use non-uniform quantization laws they cannot be implemented, except in special cases, using binary arithmetic hardware.

In this chapter an implementation method is proposed which places no restriction on the filters quantization laws. The method is based on the use of boolean polynomials as a means of computing the next state function of a finite state machine. Some results on the minimisation of the complexity of these polynomials are included. The chapter continues with a description of how digital filters may be represented as finite state machines.

4.2 The Representation of a Digital Filter as a Finite State Machine

The terminology to be used to describe a finite state machine is the same as that used by Hartmanis(73).

A finite state machine, M, is a machine which comprises a finite set of inputs $I = < I_1, I_2, I_3, \ldots, I_{N_I} >$, a finite set of states $S = < S_1, S_2, S_3, \ldots, S_{N_S} >$ and a finite set of outputs $O = < O_1, O_2, O_3, \ldots, O_{N_O} >$. The integers N_I, N_S and N_O indicate the number of members in sets I, S and O

respectively; they are referred to as the set cardinalities. Two rules D and L are defined on sets I, S and O. The rule D, called the next state rule, associates with the current state and the current input a new state. In set notation it is written:

$$D : S \times I \rightarrow S \qquad (4.2.1)$$

where $\times$ denotes cartesian product [10]. Rule L, the next output rule, depending on whether the machine is a Mealy or a Moore model, is defined as:

$$L : S \times I \rightarrow O \qquad \text{Mealy}$$

$$L : S \rightarrow O \qquad \text{Moore} \qquad (4.2.2)$$

The product set $S \times I$ is called the 'total state' set.

A Moore model finite state machine is shown in figure 4.2.1.

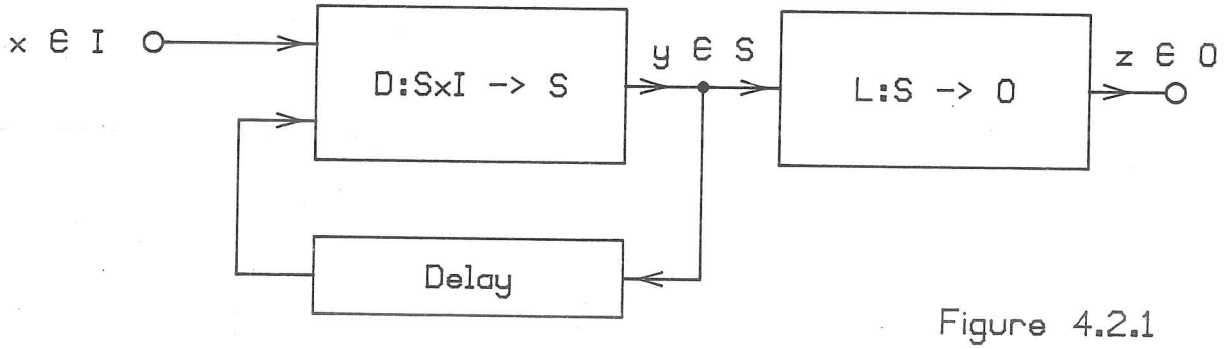


Figure 4.2.1

Only the Moore model will be required in this dissertation.

The two commonly used methods for describing rules D and L, for the case of a Moore model, are the flow table [10] and the state transition graph [10].

The flow table is a rectangular table comprising N_S rows and $N_I + 1$ columns. Each of the rows corresponds to a member of the state set S, and each of the columns corresponds to a member of the input set I. The table entry at the intersection of row S_j with column I_k ($1 \leq j \leq N_S$ and $1 \leq k \leq N_I$) specifies

respectively; they are referred to as the set cardinalities. Two rules D and L are defined on sets I, S and O. The rule D, called the next state rule, associates with the current state and the current input a new state. In set notation it is written:

$$D : S \times I \rightarrow S \qquad (4.2.1)$$

where × denotes cartesian product [10]. Rule L, the next output rule, depending on whether the machine is a Mealy or a Moore model, is defined as:

$$L : S \times I \rightarrow O \qquad \text{Mealy}$$

$$L : S \rightarrow O \qquad \text{Moore} \qquad (4.2.2)$$

The product set S × I is called the 'total state' set.

A Moore model finite state machine is shown in figure 4.2.1.

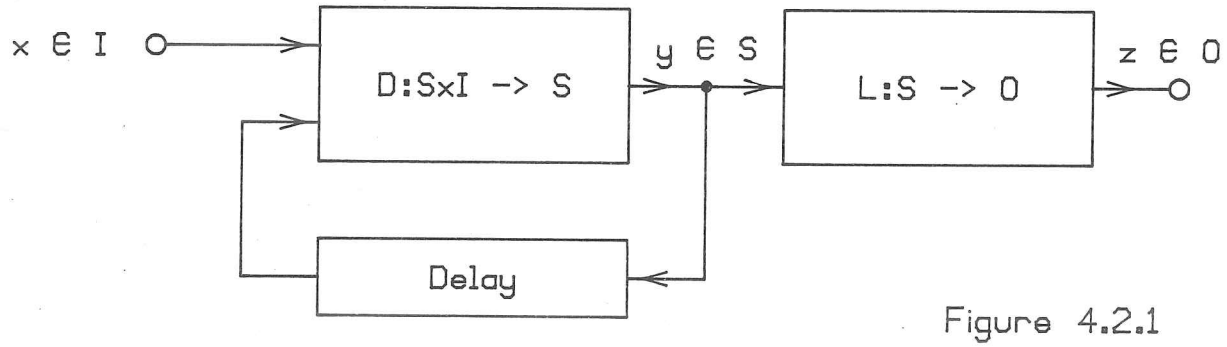


Figure 4.2.1

Only the Moore model will be required in this dissertation.

The two commonly used methods for describing rules D and L, for the case of a Moore model, are the flow table [10] and the state transition graph [10].

The flow table is a rectangular table comprising N_S rows and $N_I + 1$ columns. Each of the rows corresponds to a member of the state set S, and each of the columns corresponds to a member of the input set I. The table entry at the intersection of row S_j with column I_k ($1 \leq j \leq N_S$ and $1 \leq k \leq N_I$) specifies

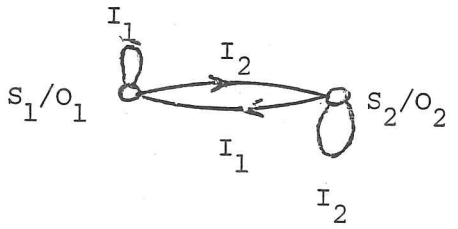
the next state. The last column specifies the next output.

The state transition graph is a directed graph with N_S nodes and N_I arcs. Each node is marked with a member of the state set and with a member of the output set. Each of the arcs is marked with a member of the input set. As the name suggests each arc specifies a state transition.

An example of a machine is shown using both method in figure 4.2.2.

	I_1	I_2	Z
S_1	S_1	S_2	O_1
S_2	S_1	S_2	O_2

Flow Table



State Transition Graph

Figure 4.2.2

The machine in figure 4.2.1 is sometimes referred to as a first order finite state machine. A cascade of k such machines would be called a k th. order finite state machine. This is an unfortunate use of the term order since a filter of any order may be represented by a so called 'first order' finite state machine.

A finite state machine may be used to represent a first order all-pole digital filter by assigning to each input quantization level a member of the set I and to each output quantization level a member of the set S. In this case sets S and O are the same. An example follows:

Consider the filter, with input x_n and output y_n, described by the difference equation

$$y_n = Q\left[\frac{x_n + y_{n-1}}{2}\right] \qquad (4.2.3)$$

where Q denotes truncation, and x_n and y_n are members of the set < -2, -1, 0, 1, 2 >. The body of the table in figure 4.2.3 shows y_n for all possible

combinations of x_n and y_{n-1}.

y_{n-1} \ x_n	-2	-1	0	1	2
-2	-2	-1	-1	0	0
-1	-1	-1	0	0	0
0	-1	0	0	0	1
1	0	0	0	1	1
2	0	0	1	1	2

Figure 4.2.3

Making an arbitrary assignment to each of the values of x_n and y_n members of the sets I and S respectively the flow table for the filters finite state machine equivalent is obtained.

D: SxI	I_0	I_1	I_2	I_3	I_4	L:S
S_0	S_o	S_1	S_1	S_2	S_2	O_0
S_1	S_1	S_1	S_2	S_2	S_2	O_1
S_2	S_1	S_2	S_2	S_2	S_3	O_2
S_3	S_2	S_2	S_2	S_3	S_3	O_3
S_4	S_2	S_2	S_3	S_3	S_4	O_4

Figure 4.2.4

By expressing a free quantization law filter as a finite state machine, all trace of the underlying arithmetic structure - the difference equation - disappears. This is useful because, as already mentioned, conventional filter implementation methods which relate strongly to the arithmetic structure are unsuitable. In addition much effort has been devoted, by a number of workers, to finding efficient methods for synthesising sequential circuits. The results of this work can be used since a sequential circuit is just an implementation of a finite state machine.

A difficulty which will repeatedly appear in this work is problem dimensionality. The first example is that of representing the second order filter, described by the difference equation:

$$y_n = K_1 y_{n-1} + K_2 y_{n-2} + x_n \qquad (4.2.4)$$

as a finite state machine. Let S_1 be the set of quantization levels to which the values of y_n, y_{n-1} and y_{n-2} belong. Denote by S_2 the cartesian product of S_1 with itself:

$$S_2 = S_1 \times S_1 \qquad (4.2.5)$$

The cardinality of set S_2 is N_S^2 where N_S is the cardinality of set S_1. Now, if the filter in equation 4.2.4 is to be modelled by a first order finite state machine (figure 4.2.1) then the machine will have as its state set S_2. The next state function, D, will be the mapping

$$D : S_2 \rightarrow S_2 \qquad (4.2.6)$$

which is considerably more complex than in the case of the first order filter. In addition, the output function will be non-trivial being the mapping:

$$D : S_2 \rightarrow O \qquad (4.2.7)$$

In this section the association of a finite state machine with a recursive digital filter has been considered. In particular, the way in which quantization levels can be associated with machine states and inputs has been described. This association will in future be called the 'value assignment'.

4.3 The Implementation of a Finite State Machine as a Sequential Circuit

In order to implement a finite state machine as a sequential circuit using binary logic it is necessary to assign to each member of the machine state set a uniquely decodable binary code. A similar assignment must be made to the input and output sets. The code used may be fixed or variable in length;

here it will be assumed to be fixed length. Denoting by P_I, P_S and P_O the number of bits required to uniquely code sets I, S and O it can be seen that:

$$\begin{aligned} P_I &\geq |\log_2 N_I| \\ P_S &\geq |\log_2 N_S| \\ P_O &\geq |\log_2 N_O| \end{aligned} \qquad (4.3.1)$$

where $|\ |$ denotes next largest integer.

Among the names used to describe this assignment are code assignment [79], state assignment [73], and internal code assignment [76]. Here the term 'code assignment' will be preferred. The term 'state assignment' will be used to refer collectively to the value assignment and to the code assignment.

Having selected a code assignment the next step is to implement the next state function, D, and the output function, L, using combinatorial logic networks. The logic implementation of the abstract finite state machine of figure 4.2.1 is shown in figure 4.3.1.

The combinatorial logic networks which implement functions D and L may be constructed using read-only memories, logic arrays or random logic. Indeed, they may be sequential circuits which appear from the viewpoint of the main machines clock to be combinatorial.

The speed at which a machine is able to clock will be determined mainly by the time taken for the outputs of networks D and L to settle. This in turn will depend on the number of gates in the longest gate chain from any input to any output. The length of the longest chain is usually referred to as the level. The example in figure 4.3.2 is a three level logic network.

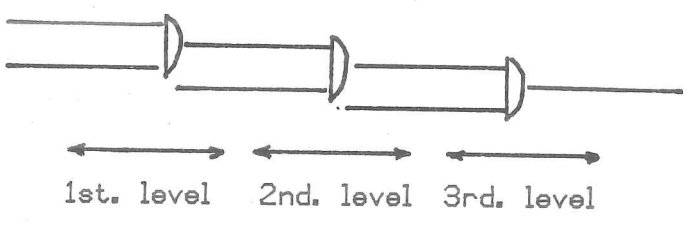
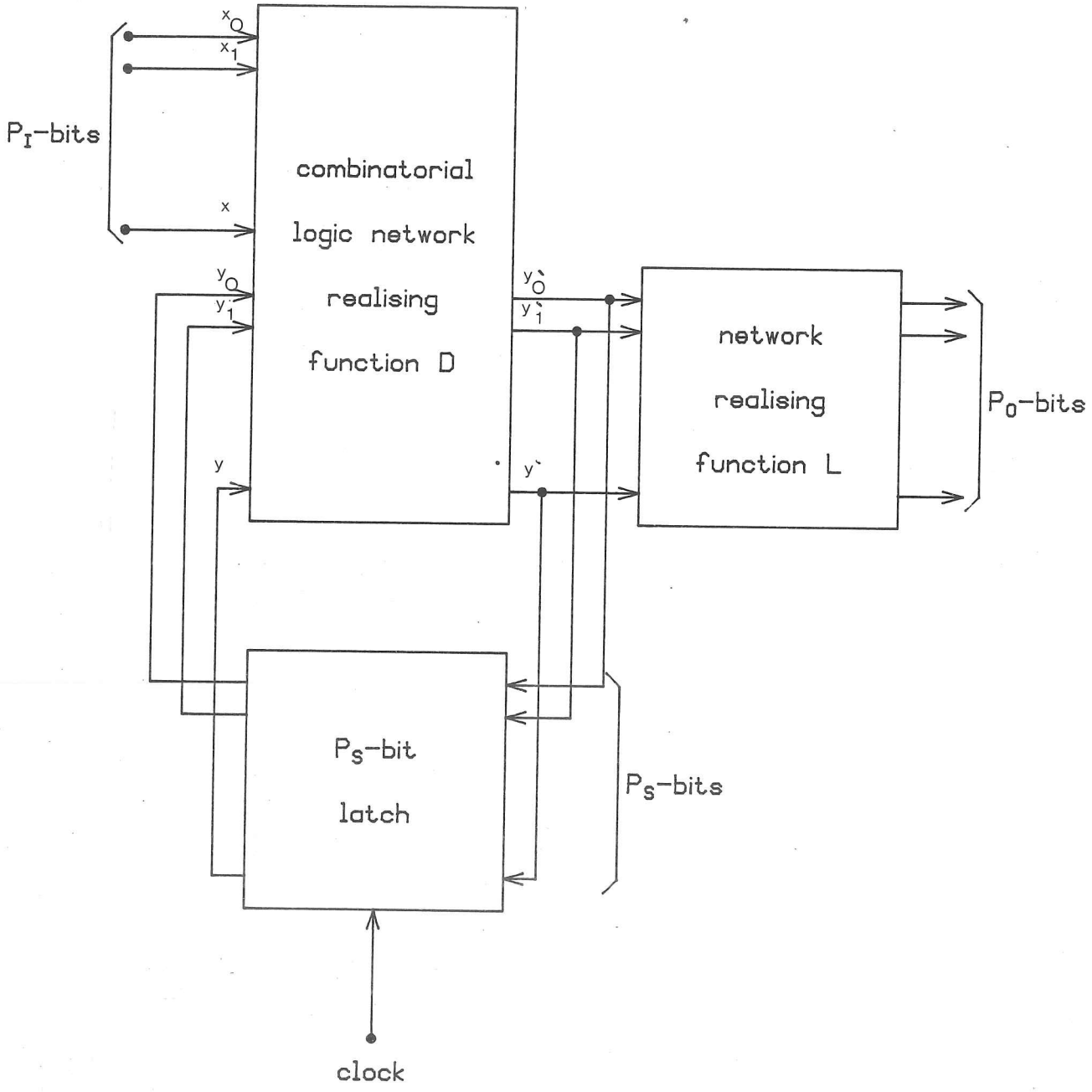


Figure 4.3.2



Logic Implementation of Abstract Finite State Machine

Figure 4.3.1

For a given logic function it is usually found that the lower the level is the more complex the logic network becomes. The logic complexity is also found to depend on the code assignment and on the type of storage latch used. In the following sections logic complexity is considered in more detail.

4.4 Logic Network Complexity

Consider the binary scalar valued function f defined by:

$$y = f(\underset{\sim}{x}) \tag{4.4.1}$$

where the vector

$$\underset{\sim}{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

is constructed of the binary scalars $x_1, x_2, \ldots, x_N$.

The complexity of function f will depend on the type of logic used, and also on the number of logic levels. For example, consider the function f defined by the two level minterm [12] expansion:

$$f(\underset{\sim}{x}) = x_o \vee (x_1 \wedge x_2) \tag{4.4.2}$$

where $\vee$ denotes logical OR, and $\wedge$ denotes logical AND. This function can be rewritten, using a two level ring-sum [12] expansion, as:

$$f(\underset{\sim}{x}) = x_0 \oplus (x_1 x_2) \oplus (x_0 x_1 x_2) \tag{4.4.3}$$

where $\oplus$ denotes logic NOT-EQUIVALENCE (EXCLUSIVE-OR). Cost is defined as the number of gates plus the number of gate inputs, Phister [12]; using this definition, the cost of implementing equations 4.4.2 and 4.4.3 is 6 units

and 11 units respectively. Although it is a useful complexity indicator, this cost measure is insensitive to the relative complexity of different gates; for example, a transistor-transistor logic (TTL) EXCLUSIVE-OR gate contains more transistors than a TTL OR gate. What is indicated is that there is a difference in complexity between the various logic forms.

In order to assess the relative complexity of different logic forms used to implement the same logical function Kellerman[99] proposes the following experiment. Let function f be defined on the N binary scalar variables $x_1, \ldots, x_N$. There are 2^N possible values of the N-tuple $x_1, x_2, \ldots, x_N$; in response to r of these f has the value '0', and in response to the remaining 2^N-r it has the value '1'. For each of the $^{2^N}C_r$ possibilities function f is minimised; the average cost over all these possibilities is called the average minimum cost. Performing this experiment for minimised minterm and ring-sum expansions of f, it is found that the ring-sum expansion by the previous definition of cost is the less costly in the average minimum sense. Hellerman[100] justifies this observation by the use of an entropy definition for the OR and EXCLUSIVE-OR gates.

The details are as follows:

Define a function f of two binary scalar variable x_0 and x_1 where x_0 and x_1 each have an equal probability of being '0' or '1'. If function f is the logical operation OR then $f(x_0, x_1)$ has probability 0.25 of being '0' and 0.75 of being '1'. The entropy of function f is therefore:

$$\begin{aligned} H(f) &= -0.25 \log_2 0.25 - 0.75 \log_2 0.75 \\ &= 0.8113 \end{aligned} \qquad (4.4.4)$$

By a similar reasoning if function f is the logical operation EXCLUSIVE-OR then the entropy of f is:

$$\begin{aligned} H(f) &= -0.5 \log_2 0.5 - 0.5 \log_2 0.5 \\ &= 1 \end{aligned} \qquad (4.4.5)$$

With the development of high density ring-sum form programmable logic arrays and in view of its reduced complexity this particular expansion has become of practical interest.

4.5 The Code Assignment Problem

As mentioned in section 4.3 the complexity of the logic networks which implement functions D and L not only depends on the logic form used, but also on the code assignment. The code assignment problem has received considerable attention in the literature since the late 1950's. To date no general solution to the problem has been found. Various methods of solution have been proposed (73-89), most of which depend for their success on certain structural properties of the machines flow table. Without exception these methods assume an AND-OR logic form for the next state and output functions. Furthermore for any appreciable number of machine states, typically fifty or more, their use becomes computationally impractical.

To illustrate the importance of selecting an economical code assignment an example will be presented. Consider the machine described by the flow table in figure 4.5.1.

Next state

	I_0	I_1	I_2	I_3
S_0	S_3	S_2	S_1	S_0
S_1	S_3	S_2	S_1	S_0
S_2	S_3	S_2	S_1	S_0
S_3	S_1	S_0	S_2	S_3

Figure 4.5.1

Two possible codings of this machine are shown in figure 4.5.2. The associated next state ring-sum polynomials are shown; the variables are those given in figure 4.3.1. The associated cost is quoted; the cost difference is seen to be significant.

1st coding

	y_1	y_0
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

	x_1	x_0
I_0	0	0
I_1	0	1
I_2	1	0
I_3	1	1

$$y_0' = 1 \oplus x_0 \oplus x_1 y_0 y_1$$

$$y_1' = 1 \oplus x_1 \oplus y_0 y_1$$

cost = 15

2nd coding

	y_1	y_0
S_0	1	1
S_1	1	0
S_2	0	1
S_3	0	0

	x_1	x_0
I_0	0	0
I_1	0	1
I_2	1	0
I_3	1	1

$$y_0' = x_0 \oplus x_1 \oplus x_1 y_0 \oplus x_1 y_1 \oplus x_1 y_0 y_1$$

$$y_1' = 1 \oplus x_1 \oplus y_0 \oplus y_1 \oplus y_0 y_1$$

cost = 25

Figure 4.5.2

A solution to the problem of finding a minimum cost code assignment must exist; in principle an exhaustive search of all possible code assignments could be performed. If the machine has N_s states then there are $N_s!$ different ways of assigning N_s codes. If $N_s = 4$, giving $N_s! = 24$, then an exhaustive search is a good method, however, if $N_s = 8$, giving $N_s! = 40320$, such a method

becomes impractical. The number of code assignments which need to be tested can be reduced. Consider the ordered set of binary digits $P_{N-1}, P_{N-2}, \cdots, P_2, P_1, P_0$ to be an N bit code used to code each of the N_s states of a machine ($N_s = 2^N$). Now reverse a pair of bits: for example $P_{N-1}, P_{N-2}, \cdots, P_1, P_2, P_0$. This has the effect of changing the code assignment but not the next state polynomials. As there are N! ways of ordering N bits the number of possible code assignments, C, reduces to:

$$C = \frac{N_s!}{(\log_2 N_s)!} \qquad (4.5.1)$$

$N_s = 8$ gives $C = 6720$

Various authors [68-72] consider ways by which to reduce C still further. These methods depend on the type of delay store. For example, if each of the state bits is available in true and complement form then code assignments which differ only by a complement will have the same cost. Bianchini [69] presents a method for systematically generating all distinct code assignments. The method does not require that all the previously generated assignments be stored.

Even for a modest number of states the computational savings offered by the above methods are inadequate. Typically 32 states would require that some 10^{33} code assignments be tested. Examining the expression given in (4.5.1) for the number of code assignments it is seen that C cannot be expressed as a finite degree polynomial in N_s except for limited values of N_s. This being the case, the code assignment problem belongs to the class of problems which are termed non-polynomial solvable. A number of workers have proposed methods for rapidly finding optimal or near optimal code assigments. Of these methods two of the more well known ones are those of Hartmanis and Stearns [74] and Story, Harrison and Rainhard - SHR - [86].

The former method relies on the isolation of so called sub-computations

within the machines next state function. To explain this further the following definition is required. Let S be a set which is divided up into subsets $B_0, B_1, \ldots, B_N$. If the following properties exist:

$$S = \bigcup_{i=0}^{N} B_i \tag{4.5.1a}$$

$$B_i \cap B_j = \begin{cases} \emptyset & i \neq j \\ B_i & i = j \end{cases} \tag{4.5.1b}$$

where U denotes set union, $\cap$ denotes set intersection and $\emptyset$ is the empty set, then set S is said to have been partitioned into blocks $B_0, B_1, \ldots, B_N$. Returning to the method of Hartmanis and Stearns. They propose an algorithm for finding a partition, on a machines state set S, which has the following property; the block containing the next state only depends on the input and on the block containing the current state. In general the exact next state will depend on both the previous state and the input. Effectively this method selects a code assignment such that some of the next state polynomials only depend on the input bits and on a subset of the state bits. The rest of the next state polynomials will depend on all the total state bits. Arbitrarily selecting a code assignment would usually cause each of the next state polynomials to depend on all the input and state bits. It follows that when a machine is partitionable some reduction in the complexity of the next state polynomials is to be expected. There are two drawbacks to this method. The machine must possess a partitionable state structure. Also, for a large number of states the partitioning algorithm would require excessive computing resources. For a digital filter realised as a finite state machine there is one sub-computation which can be isolated by inspection; the details follow. Consider the constant coefficient filter, with input x_n and output y_n, which is described by the difference equation:

$$y_n = Ky_{n-1} + x_n \tag{4.5.2}$$

The values which the variables x_n and y_n may assume are constrained to their respective finite sets. Both these sets have the property that for each positive member - quantization level - there is a negative member which is equal in magnitude. If x_n and y_{n-1} are both positive then:

$$y_n = |Ky_{n-1}| + |x_n| \qquad (4.5.3a)$$

where | | denotes 'magnitude of', whereas, if both are negative then

$$y_n = -|Ky_{n-1}| - |x_n| \qquad (4.5.3b)$$

If, however, y_{n-1} is positive and x_n is negative or y_{n-1} is negative and x_n is positive then:

$$y_n = |Ky_{n-1}| - |x_n| \qquad (4.5.3c)$$

and:

$$y_n = -|Ky_{n-1}| + |x_n| \qquad (4.5.3d)$$

respectively. Equation 4.5.3b can be evaluated by evaluating equation 4.5.3a and changing the sign of the result. Similarly, equation 4.5.3d can be evaluated using equation 4.5.3c. This result suggests that there may be some advantage in choosing codings for x_n and y_n which use one bit to specify the sign. Suppose that a read only memory (r.o.m.) is used to evaluate equation 4.5.2. If x_n and y_n both require N-bit codes then the r.o.m. would contain $N \times 2^{2N}$ cells. If, however, a r.o.m. is used to evaluate equations 4.5.3a and c then, since $|Ky_{n-1}|$ and $|x_n|$ are codable in N-1 bits, $2N2^{2N-2}$ cells are required. The number of cells in the r.o.m. has been halved. This type of code is referred to as a sign-magnitude code. Unfortunately here this name is inappropriate as the 'magnitude' part of the code need not bear a simple relationship with the signal sample value. Some additional logic is required to decide the sign of the result and to decide whether to apply

equation 4.5.3a or 4.5.3c. The logic specification will be given in terms of the following variables:

S_x - sign bit of x_n

S_x = 0 => x_n positive

S_x = 1 => x_n negative

S_y - sign bit of y_n

definition as for S_x

S_c - sign complement bit

S_c = 0 => leave result unchanged

S_c = 1 => change sign of result

f - equation selector

f = 0 => apply equation 4.5.3a

f = 1 => apply equation 4.5.3c

S_r - sign of result given by r.o.m.

definition as for S_x

S_R - sign of final result

Figure 4.5.4

The truth table relating f, S_c and S_R to S_x, S_y and S_r is shown in figure 4.5.3.

The following logical expressions for f, S_c and S_R follow from Table 4.5.3:

$$f = S_x \oplus S_y$$

$$S_c = S_y$$

$$S_R = S_r \oplus S_c \qquad (4.5.4)$$

S_x	S_y	S_r	f	S_c	S_R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	1	0

Figure 4.5.3

Additional logic amounting to two EXCLUSIVE-OR gates is all that is required to calculate the sign.

A description of the SHR code assignment algorithm will now be given. It is observed that, although the number of distinct code assignments is:

$$C = \frac{2^N!}{N!} \tag{4.5.5a}$$

where $N = \log_2 N_S$, there are only:

$$H = \frac{2^N!}{(2^{N-1}!)^2} \tag{4.5.5b}$$

different columns which go to make up the C row assignments. For example, if the machine has 4 states then the C = 12 distinct code assignments are:

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Figure 4.5.5

Among these 12 assignments are the following distinct columns:

```
0 0 1 0 1 1
0 1 0 1 0 1
1 0 0 1 1 0
1 1 1 0 0 0
```

Figure 4.5.6

The SHR algorithm associates with each distinct column a so called minimum number (MN). This number is a measure of the cost of using the associated column as a part of a complete code assignments. It is called a miminum number as it is a lower cost bound; in other words, if the column is accompanied by the appropriate N-1 other columns it will contribure MN units to the total cost of the machine. The algorithm, having computed the MN for each of the H columns, proceeds by combining the N lowest cost columns subject to the following condition; the combined use of N columns must lead to a proper code assignments, that is, an assignment in which each state has a unique code. For example, the first and last columns in figure 4.5.6 cannot be used together. The SHR algorithm finds the MN for each column without having to consider any of the other columns. It is designed for use with minterm (AND-OR) next state logic and J-K flip flop delay elements. It is likely that this method could be re-developed for use with ring-sum logic.

A third method for finding an economical code assignment will now be described. Consider the machine shown in figure 4.3.1 with its next state function, D, implemented using ring-sum polynomials. These polynomials are found by applying the Reed-Muller Transformation to the coded next state table (see Chapter 5 for a description of the R.M. Transform). The cost of the machine will be measured by counting the number of terms in all the polynomials. A term is a product of binary scalars, for example $x_0x_2x_5$. The method is based on partitioning the space of all possible code assignments into blocks (sub-spaces): Each block has the property that all its members are related to

each other by a linear transformation. To illustrate this consider the 2-input 8-state machine in figure 4.5.7.

$y_2 y_1 y_0$	$x_0 = 0$	$x_0 = 1$
0 0 0	001	010
0 0 1	010	011
0 1 0	011	100
0 1 1	100	101
1 0 0	101	110
1 0 1	110	111
1 1 0	111	000
1 1 1	000	001

Figure 4.5.7a

The corresponding next state polynomials are:

$$y_0' = 1 \oplus y_0 \oplus x_0$$

$$y_1' = y_0 \oplus y_1 \oplus x_0 \oplus x_0 y_0$$

$$y_2' = y_0 y_1 \oplus y_2 \oplus x_0 y_1 \oplus x_0 y_0 y_1 \qquad (4.5.6a)$$

Now, applying the following linear transformation:

$$\begin{bmatrix} 1 \\ z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

the next state table becomes:

z_2	z_1	z_0	x_0 = 0	x_0 = 1
0	0	0	001	111
0	0	1	111	110
1	1	1	110	100
1	1	0	100	101
1	0	0	101	011
1	0	1	011	010
0	1	1	010	000
0	1	0	000	001

Figure 4.5.7b

and the next state polynomials become:

$$z_0' = 1 \oplus z_1 \oplus x_0z_0 \oplus x_0z_1$$

$$z_1' = z_0 \oplus x_0 \oplus x_0z_0 \oplus x_0z_1$$

$$z_2' = z_0 \oplus z_0z_1 \oplus z_2 \oplus x_0 \oplus x_0z_0 \oplus x_0z_1 \oplus x_0z_0z_1 \qquad (4.5.6b)$$

The objective is to randomly select a code assignment and then find a linear transformation which maps this to a low cost code assignment. If, within the current block, the lowest cost code assignment is too costly then a non-linear transformation is applied to change to a different block. If the machine has N_S states which are coded by $N = \lceil \log_2 N_S \rceil$ bits then there are

$$2^N \prod_{i=0}^{N-1} (2^N - 2^i)$$

different linear transformations. This is readily shown. Consider the general linear transformation of N binary variables onto N binary variables described by the matrix equation:

$$\begin{bmatrix} 1 \\ z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ c_0 & a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,N-1} \\ c_1 & a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,N-1} \\ c_2 & a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,N-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ c_{N-1} & a_{N-1,0} & a_{N-1,1} & a_{N-1,2} & & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} 1 \\ y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix}$$

$$\underset{\sim}{z} = \underline{T}\,\underset{\sim}{y} \qquad (4.5.7)$$

Matrix $\underline{T}$ must be non-singular and so

Row 2 may be chosen in 2^N-1 ways, as it may not be zero;

Row 3 may be chosen in 2^N-2 ways, as it may not be zero or equal to Row 2

Row 4 may be chosen in 2^N-4 ways, as it may not be zero, equal to Rows 2 or 3 or equal to Row 2 $\oplus$ Row 3.

Repeating this up to Row N the following product is obtained:

$$\prod_{i=0}^{N-1} (2^N - 2^i)$$

Now each of the c_j may be 0 or 1 and hence the total number of linear transformations is:

$$N_T = 2^N \prod_{i=0}^{N-1} (2^N - 2^i) \qquad (4.5.8)$$

When a transformation can be described by a matrix equation that transformation will be called linear. All other mappings which are one-to-one and onto will be called non-linear transformations.

An algorithm has been developed for removing unnecessary complementation. Complementation is a special case of the general linear transformation. The details are as follows:

(1) Arrange the next state polynomials in a table. Each row of the table corresponds to a different polynomial and each column to a different term. The entry at row i, column j is the coefficient of term j, in polynomial i. For example, the polynomials:

$$y_0' = 1 \oplus x_0 \oplus y_0$$
$$y_1' = 1 \oplus x_1 \oplus x_0y_1 \oplus x_0x_1y_0y_1 \qquad (4.5.9)$$

would be displayed in tabular form as follows:

	1	x_0	x_1	x_0x_1	y_0	y_0x_0	y_0x_1	$y_0x_0x_1$
y_0'	1	1	0	0	1	0	0	0
y_1'	1	0	1	0	0	0	0	0
	y_1	y_1x_0	y_1x_1	$y_1x_0x_1$	y_0y_1	$y_0y_1x_1$	$y_0y_1x_1$	$y_0y_1x_0x_1$
y_0' cont'd	0	0	0	0	0	0	0	0
y_1' cont'd	0	1	0	0	0	0	0	1

Figure 4.5.8

(2) Replace each of the state code bits (y_0, y_1, ..., y_0', y_1', ...) by its conditional complement ($c_0 + y_0$, $c_1 + y_1$, ..., $c_0 + y_0'$, $c_1 + y_1'$, ...). The coefficients c_j are as described in matrix equation 4.5.7.

(3) Re-arrange the polynomials into canonic form.

Combining steps (2) and (3) the polynomial:

$$y_0' = 1 \oplus x_0 \oplus x_1y_1$$

becomes

$$y_0' = (1 + c_0) \oplus x_0 \oplus c_1x_1 \oplus x_1y_1$$

(4) Form the list of coefficient products $P_k(i,j)$. The value $P_k(i,j)$ is the product of the coefficients in figure 4.5.8 at column k, row i and column k, row j.

(5) Find the values of c_i which make as many of the $P_k(i,j)$ as possible zero.

In effect, the algorithm aims to find a transformation made up of just complements, which causes the next state polynomials to have as few common terms as possible. An example of the algorithm will now be given.

Consider the machine described by the next state polynomials

$$y_0' = 1 \oplus x_0 \oplus x_1 \oplus x_0x_1 \oplus x_0y_1 \oplus x_1y_0 \oplus x_0x_1y_0 \oplus x_0x_1y_1 \oplus x_0x_1y_0y_1$$

$$y_1' = 1 \oplus x_0 \oplus y_0$$

Applying step (1)

	1	x_0	x_1	x_0x_1	y_0	y_0x_0	y_0x_1	$y_0x_0x_1$
y_0'	1	1	0	0	1	0	0	0
y_1'	1	1	1	1	0	0	1	1
	y_1	y_1x_0	y_1x_1	$y_1x_0x_1$	y_0y_1	$y_0y_1x_0$	$y_0y_1x_1$	$y_0y_1x_0x_1$
y_0'	0	0	0	0	0	0	0	0
y_1'	0	1	0	1	0	0	0	1

Figure 4.5.9a

Applying steps (2) and (3)

$1+c$	1	0	0	1	0	0	0
1	$1+c_1$	$1+c_0$	$(1+c_0)x$ $(1+c_1)$	0	0	1	$1+c_1$
0	0	0	0	0	0	0	0
0	1	0	$1+c_0$	0	0	0	1

Figure 4.5.9b

The corresponding values of $P_k(i,j)$ are

k	$P_k(0,1)$	k	$P_k(0,1)$
0	$1 + c_0$	8	0
1	$1 + c_1$	9	0
2	0	10	0
3	0	11	0
4	0	12	0
5	0	13	0
6	0	14	0
7	0	15	0

Figure 4.5.10

By inspection $P_0(0,1) = P_1(0,1) = 0$ when $c_0 = c_1 = 1$. Substituting these values for c_0 and c_1 in Figure 4.5.9b the reduced complexity polynomials are

$$y_0' = x_0 \oplus y_0$$

$$y_1' = 1 \oplus x_1y_0 \oplus x_0y_1 \oplus x_0x_1y_0y_1$$

For larger numbers of polynomials solution by inspection for the values of the coefficients c_i is not possible. Instead, the Reed Muller transformation has to be used. Transforming the polynomials for $P_0(0,1)$ and $P_1(0,1)$ in the example above;

term	polynomial $P_0(0,1)$	polynomial $P_1(0,1)$		value of $P_0(0,1)$,	$P_1(0,1)$ at	c_1	c_0
1	1	1	RMT	1	1	0	0
c_0	1	0	⟶	0	1	0	1
c_1	0	1		1	0	1	0
c_0c_1	0	0		0	0	1	1

Figure 4.5.11

it is seen that when $c_0 = c_1 = 1$, $P_0(0,1) = P_1(0,1) = 0$, as expected.

The disadvantage of this algorithm, as with the first two, is that of scale. Consider how much storage is required for table 4.5.9b in the case of k polynomials. Each polynomial has 2^{2k} terms; each term has a coefficient which requires 2^k bits of storage. In total $k\ 2^{3k}$ bits of storage are needed for k polynomials. In addition, storage for the values of $P_k(i,j)$ is required. For each column there are ${}^kC_2$ products; each product requires 2^k bits of storage; there are 2^{2k} columns and hence ${}^kC_2\ 2^{3k}$ bits of storage are required. The total storage requirement is

$$(k + {}^kC_2)\ 2^{3k} = k(k+1)2^{3k-1}$$

This algorithm was used successfully for up to k = 5 polynomials.

4.6 Next State Polynomials Minimisation

A logic function may be expressed in a number of different ways. Even within a particular logic form alternatives exist. The complexity of the associated hardware varies so much from one expression to another that it is important to have means of finding a minimum complexity expression. When several logic functions are to be evaluated simultaneously additional savings are possible by identifying sub-evaluations which occur in more than one function. Perhaps the best known example of a minimisation method is the multiple output method of Quine and McCluskey. The QM technique has become the standard method of minimising minterm expansions by digital computer. No standard technique has yet been developed for minimising ring-sum expanded logic functions. As the following example shows there is much to be saved by finding such a technique.

$$\begin{aligned} y &= 1 \oplus x_0 \oplus x_1 \oplus x_0x_1 \\ &= (1 \oplus x_0)(1 \oplus x_1) \\ &= x_0'x_1' \end{aligned}$$

where ' denotes complement

If x_0 and x_1 are available in complement form then just one AND gate is required. The first expression requires one AND gate and one four input EXOR gate.

No new method is to be presented here. The section is included because logic minimisation is as important in the design of finite state machines as the choice of a good code assignment and the choice of the best memory elements. In the literature methods for minimising ring-sum polynomials are proposed [101]. More recently, Rayner has developed such an algorithm.

4.7 Concluding Remarks

In this chapter attention has been drawn to the problem of realising free quantization law digital filters. It has been shown that conventional digital arithmetic hardware is unsuitable for implementing these filters. As an alternative a sequential circuit implementation of a finite state machine has been proposed. The steps in the design of a sequential circuit which have been discussed are:

(1) Using the filters difference equation and its quantization laws to evaluate the next state table of the equivalent finite state machine.

(2) Select a logic form for implementing the machines next state function.

(3) Code the machine states using a suitable length binary word, so that the next state polynomials are cost minimal.

(4) Apply a logic reduction technique to further reduce the complexity of the next state polynomials.

It has been implicitly assumed that the machines next state table is irredundant. Between steps (1) and (2) a state reduction should be performed. The reason for this is as follows: Consider a machine with a state set which has 2^N-1 members each coded using N bits; with this scheme there is one redundant state. The transitions from this state may be selected in a way which reduces the

next state polynomial complexity. This is analogous to 'don't-care' conditions in a combinational logic circuit. If, in fact, the machine has further redundant states then further reductions in complexity are possible. Such redundant states are the so called equivalent states [10]. Some techniques for identifying equivalent states are to be found in the literature. Their use in this application is a matter for further work.

Finally, a special case of the free quantization law filter which leads to a simple hardware implementation has been found; it is discussed in Appendix C.

5. A Reed-Muller Transform Processor

5.1 Introduction

In chapter 4 a technique for minimising the next state function of a finite state machine, when it is implemented using AND/EXCLUSIVE-OR logic, was presented. A method for transforming from the operational domain of a logic network to the associated ring-sum multinomials (polynominals) domain was assumed to exist. In this chapter the required transformation, the so called Reed-Muller (RM) Transformation [98], will be described; a special purpose processor for the rapid computation of the RM transform will also be described.

5.2 The Reed-Muller Transform

Consider the logic network L shown in figure 5.2.1. Each of the binary valued output variables (literals) $y_1 \ldots y_N$ is related by a corresponding multinomial $f_1 \ldots f_N$ to each of the input literals $x_1 \ldots x_N$.

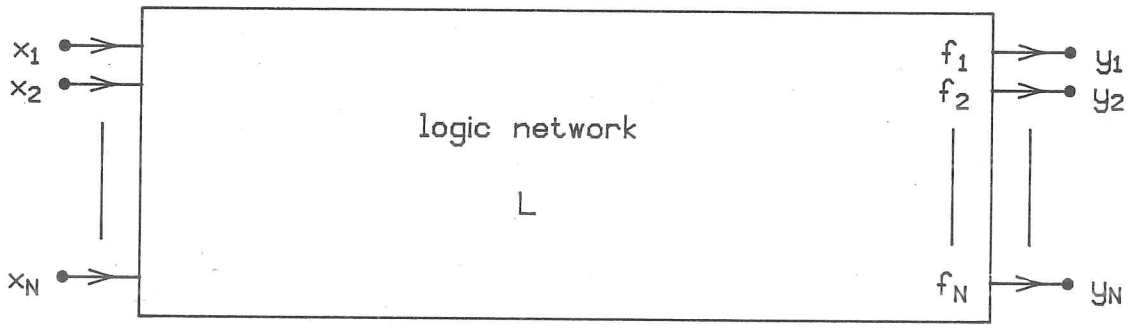


Figure 5.2.1

It will be assumed that for each of the possible values of the input binary N-vector $(x_1\ x_2 \ldots x_N)$ the value of the output binary N-vector $(y_1\ y_2 \ldots y_N)$ is known. The problem is to find the logic network L which will realise this input to output relationship. In principle functions

5. A Reed-Muller Transform Processor

5.1 Introduction

In chapter 4 a technique for minimising the next state function of a finite state machine, when it is implemented using AND/EXCLUSIVE-OR logic, was presented. A method for transforming from the operational domain of a logic network to the associated ring-sum multinomials (polynominals) domain was assumed to exist. In this chapter the required transformation, the so called Reed-Muller (RM) Transformation [98], will be described; a special purpose processor for the rapid computation of the RM transform will also be described.

5.2 The Reed-Muller Transform

Consider the logic network L shown in figure 5.2.1. Each of the binary valued output variables (literals) $y_1 \ldots . y_N$ is related by a corresponding multinomial $f_1 \ldots . f_N$ to each of the input literals $x_1 \ldots . x_N$.

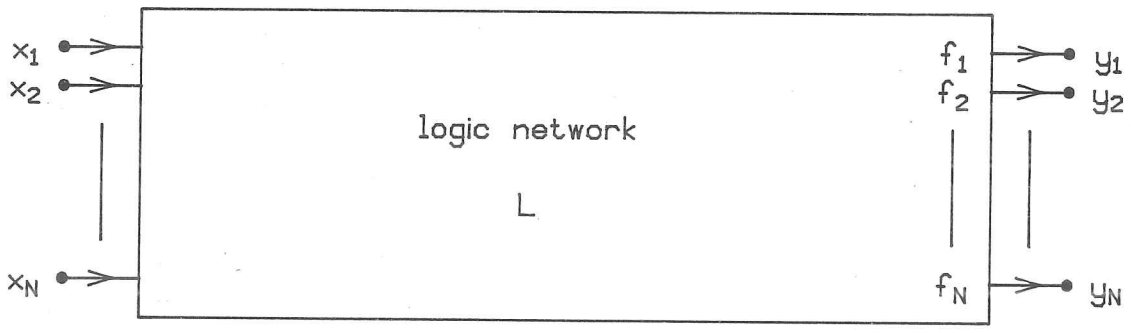


Figure 5.2.1

It will be assumed that for each of the possible values of the input binary N-vector $(x_1\ x_2 \ldots . x_N)$ the value of the output binary N-vector $(y_1\ y_2 \ldots . y_N)$ is known. The problem is to find the logic network L which will realise this input to output relationship. In principle functions

$f_1 \ldots . f_N$ could be realised using a random connection of AND, OR and EXCLUSIVE-OR gates, however, to allow automation a standard form such as minterm (AND/OR) or ring-sum (AND/EXCLUSIVE-OR) is normally used. Here the ring-sum expansion will be used.

To simplify the following discussion L will be assumed to be a two input (x_1,x_2) and one output (y) network. The general ring-sum expansion of y in terms of x_1 and x_2 is given by:

$$y = a_{11} \oplus a_{12}x_1 \oplus a_{21}x_2 \oplus a_{22}x_1x_2 \qquad (5.2.1)$$

where $\oplus$ denotes modulo-2 addition,

p.q (abbreviated to pq) denotes binary multiplication,

and the a_{ij} are constant binary valued coefficients

Denoting by y(p,q) the value of y when $x_1 = p$ and $x_2 = q$ the following table is obtained:

x_2	x_1	y
0	0	$y(0,0) = a_{11}$
0	1	$y(0,1) = a_{11} \oplus a_{12}$
1	0	$y(1,0) = a_{11} \oplus a_{21}$
1	1	$y(1,1) = a_{11} \oplus a_{12} \oplus a_{21} \oplus a_{22}$

figure 5.2.2

By operating on the y column of the table in figure 5.2.2 the coefficients a_{ij} are obtained in terms of the operational domain values y(p,q) as follows:

$$\begin{aligned} a_{11} &= y(0,0) \\ a_{12} &= y(0,0) \oplus y(0,1) \\ a_{21} &= y(0,0) \oplus y(1,0) \\ a_{22} &= y(0,0) \oplus y(0,1) \oplus y(1,0) \oplus y(1,1) \end{aligned} \qquad (5.2.2)$$

which in binary matrix notation becomes:

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y(0,0) \\ y(0,1) \\ y(1,0) \\ y(1,1) \end{bmatrix} \qquad (5.2.3)$$

or $\underset{\sim}{A}_2 = \underline{R}_2 \underset{\sim}{Y}_2$

Matrix $\underline{R}_2$ is called the two variable Reed-Muller transformation matrix.

For the case of three input variables (x_1, x_2, x_3) the desired transformation matrix is:

$$\underline{R}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad (5.2.4)$$

It can be seen that matrix $\underline{R}_3$ partitions as follows:

$$\underline{R}_3 = \left[\begin{array}{c|c} \underline{R}_2 & \underline{\emptyset} \\ \hline \underline{R}_2 & \underline{R}_2 \end{array}\right] \qquad (5.2.5)$$

where $\underline{\emptyset}$ denotes the null matrix.

This in turn is recognised as the Kronecker matrix product:

$$\underline{R}_3 = \underline{R}_1 \otimes \underline{R}_2 \qquad (5.2.6)$$

It can be shown that in general:

$$\underline{R}_N = \otimes \underline{R}_1^N \qquad (5.2.7)$$

The binary values Ø and 1 together with the operations AND(.) and EXCLUSIVE-OR ($\oplus$) form a finite field called Galois Field.2 and abbreviated to GF(2). The RM transformation discussed above is a special case of the more general transformation defined for multiple valued logic over GF(p) (p-prime) and discussed by Green[98]. Multiple valued logic is currently an active research area, however, it will receive no further consideration here as it is currently not conveniently realised.

5.3 A Reed-Muller Transform Processor

When designing a sequential circuit to realise a finite state machine the first step is to select a code assignment. The next step is to find the multinomials which relate the input and current state to the next state. If the multinomials are ring-sum expansions then what is required is to rearrange the machines next state table as a linear array and then to compute the Reed-Muller transform. An example of such a rearrangement is shown in figure 5.3.1.

next state $y_2'y_1'$ \ current state $y_2\ y_1$	00	01	10	11
00	P_0	P_1	P_2	P_3
01	P_4	P_5	P_6	P_7
10	P_8	P_9	P_{10}	P_{11}
11	P_{12}	P_{13}	P_{14}	P_{15}

($x_2\ x_1$ input)

linearise ↔

input x_2	input x_1	current state y_2	current state y_1	next state $y_2'\ y_1'$
0	0	0	0	P_0
0	0	0	1	P_1
0	0	1	0	P_2
0	0	1	1	P_3
0	1	0	0	P_4
0	1	0	1	P_5
0	1	1	0	P_6
0	1	1	1	P_7
1	0	0	0	P_8
1	0	0	1	P_9
1	0	1	0	P_{10}
1	0	1	1	P_{11}
1	1	0	0	P_{12}
1	1	0	1	P_{13}
1	1	1	0	P_{14}
1	1	1	1	P_{15}

figure 5.3.1

By choosing a suitable initial arrangement in a computers main memory this rearrangement is implied.

A difficulty which arises when designing a high speed hardware RM transform processor is that for large numbers of variables the amount of high speed memory required is prohibitively expensive. This difficulty can be overcome by trading processor speed for processor cost. The processor to be described is designed to compute the RM transform across one row or down one column of a machines next state table; the entire transform is constructed by applying the processor to each of the rows (columns) and then to each of the columns (rows). The decomposition of a two dimensional 2^N-point by 2^M-point RM transform into 2^N, 2^M-point and 2^M, 2^N-point one dimensional transforms will now be illustrated by example:

x_2 x_1 \ y_2 y_1	00	01	10	11
00	00	01	10	11
01	01	10	11	11
10	10	11	11	11
11	11	11	11	11

operational domain

figure 5.3.2

00	01	10	00
01	11	10	11
10	01	01	01
11	00	00	00

after 4 row transforms

figure 5.3.3

00	01	10	00
01	10	00	11
10	00	11	01
00	11	01	10

after 4 column transforms

figure 5.3.4

Figure 5.3.2 is shown rearranged as a linear array in figure 5.3.5. Its RM transform is also shown; the transform is seen to be a linearised version of figure 5.3.4.

x_2	x_1	y_2	y_1	Y_2	Y_1	Y_2	Y_1
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1
0	0	1	0	1	0	1	0
0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	1	0	1	0
0	1	1	0	1	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	0
1	0	1	0	1	1	1	1
1	0	1	1	1	1	0	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	1	1
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

figure 5.3.5

The processor is designed to communicate directly with the main memory of a general purpose minicomputer; its specifications are as follows:

1. Up to 8 multinomials simultaneously computed
2. Software selectable multinomial sizes of 16, 32, 64, 128 and 256 coefficients
3. Capability to deal with operational domain data which is either consecutively organised in the host computers main memory or is seperated by 2, 4, 8, 16, 32, 64, 128 or 256 locations.

A brief description of the processor is given in this dissertation; complete details are contained in a technical report [90].

The processor divides into three distinct units; they are as follows:

1. The processor - host computer communications unit
2. The processor arithmetic unit

and 3. The processor control unit.

All three units are indicated on the diagram in figure 5.3.6.

The sequence of events through which the host computer and the processor pass in order to compute one transform are as follows:

1. The host computer provides the processor with:
   (i) The address in main memory of the first data point
   (ii) The offset from the Nth to the N + 1th data point
   and (iii) The number of data points
2. The host computer instructs the processor to commence transforming
3. The transform processor reads the operational domain data through the host computers direct memory access (DMA) channel into its own high speed store
4. The transform is computed
5. The transform processor returns the multinomials in place to the computers main memory; this is again done via the DMA channel

and 6. The processor sets the transform completed flag.

After step 2 the host computer is free to perform other computations.

The transfers to and from the host computers main memory via the DMA channel are sequenced by the processors control logic (unit 3 in figure 5.3.6). Assoicated with each transfer are two addresses, one to the hosts main memory and one to the processors local high speed memory; these are generated by the DMA and processor address generators respectively (units 1 and 2). The DMA address generator is non-standard in that it is capable of generating non-consecutive addresses.

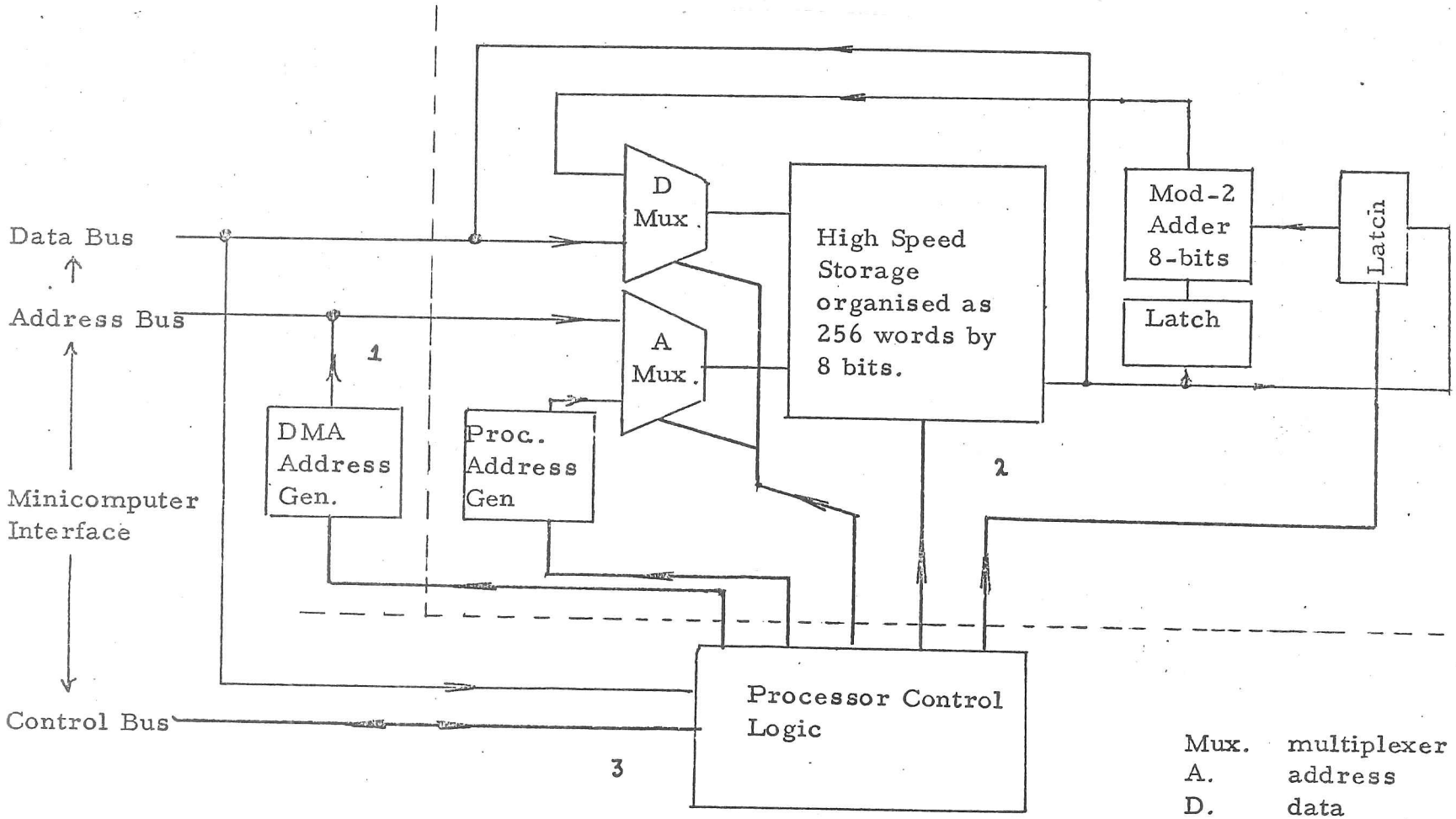


Figure 5.3.6 Reed-Muller Transform Processor. Block Diagram

Next attention is focussed on the actual transform computation. It will be assumed that the operational domain data is in consecutive locations in the processors high speed store; the sequence of events required to produce in place the multinomial domain data will be considered.

To illustrate the actions of the processor an 8-point transform will be considered. First the following convention is required; the left most column of matrix $\underline{R}_3$ is column 0.

The sequence of addresses which must be generated in order to multiply in place by $\underline{R}_3$ is as follows:

0, 1, 2, 3, 4, 5, 6, 7

0, 2, 4, 6

0, 1, 4, 5

0, 4

0, 1, 2, 3

0, 2

0, 1

0

figure 5.3.7

This address sequence does not lead to a neat hardware realisation of the processor address generator, however, by factorizing $\underline{R}_3$ appropriately this difficulty can be overcome.

Consider the following factorization of $\underline{R}_3$:

$$\underline{R}_3 = \begin{bmatrix} 1 & & & & & & & \\ 1 & 1 & & & & & & \\ 1 & & 1 & & & & & \\ 1 & 1 & 1 & 1 & & & & \\ 1 & & & & 1 & & & \\ 1 & 1 & & & 1 & 1 & & \\ 1 & & 1 & & 1 & & 1 & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \underset{\text{stage 3}}{\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ 1 & & & & 1 & & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & 1 \end{bmatrix}} \times \underset{\text{stage 2}}{\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ 1 & & 1 & & & & & \\ & 1 & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & 1 & & 1 & \\ & & & & & 1 & & 1 \end{bmatrix}} \times$$

$$\underset{\text{stage 1}}{\begin{bmatrix} 1 & & & & & & & \\ 1 & 1 & & & & & & \\ & & 1 & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & 1 & 1 & \\ & & & & & & & 1 \\ & & & & & & 1 & 1 \end{bmatrix}}$$

all blank entries have the value 0;

(5.3.1)

$$= \underline{C}_3 \times \underline{B}_3 \times \underline{A}_3$$

This is called a Radix - 2 factorization.

The address sequences involved in non-trivial operations are as follows:

0, 1, 2, 3, 4, 5, 6, 7 - stage 1

0, 2, 1, 3, 4, 6, 5, 7 - stage 2

0, 4, 1, 5, 2, 6, 3, 7 - stage 3

figure 5.3.8

Written out in binary these become:

Stage 1	Stage 2	Stage 3
000	000	000
001	010	100
010	001	001
011	011	101
100	100	010
101	110	110
110	101	011
111	111	111

figure 5.3.9

It can be seen that stage 1 involves an ordinary binary count from 0 to 7. Stage 2 also involves an ordinary binary count but with the least significant two bits interchanged. Finally, stage 3 is an ordinary binary count with the following bits exchanged:

bit 0 to bit 2

bit 1 to bit 0

bit 2 to bit 1

figure 5.3.10

As the processor is to be designed to compute transforms which are up to 256 - points it is necessary to establish the addressing mechanism for up to 8 address lines. The radix-2 factorization of $\underline{R}_N$ contains N matrices and thus the transform is constructed from N stages. Denoting by c_0 and c_7

the least and most signficant bits respectively of an 8-bit binary counter, it can be shown that the address bit exchanges as a function of the transform stage are as shown in figure 5.3.11.

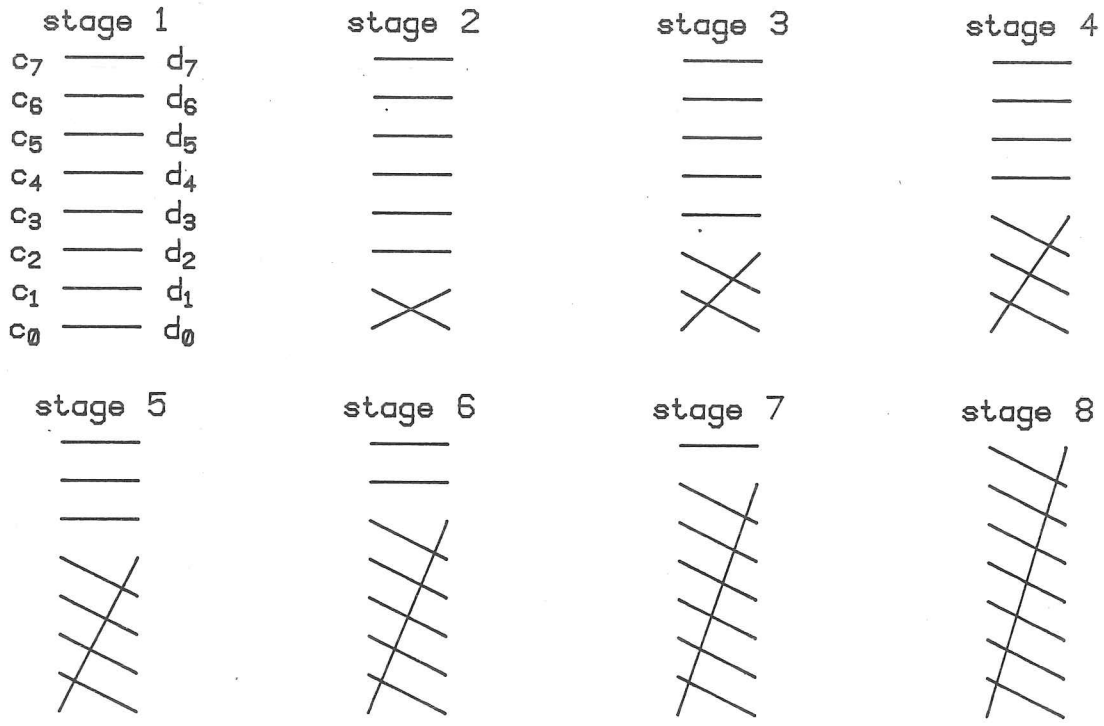
A number of ways of realising this addressing mechanism were considered. Among them were the use of an array of tri-state couplers as shown in figure 5.3.12 and the use of steered clock signals in an 8 flip-flop counter.

Neither of these solutions was considered satisfactory; the solution which was adopted is as follows. Notice that $d_0 \ldots d_3$ depend on $c_0 \ldots c_4$ and on the stage bits $s_0 \ldots s_2$; also notice that $d_4 \ldots d_7$ depend on c_0, $c_4 \ldots c_7$ and $s_0 \ldots s_2$. It follows that d_0 through d_7 can be generated using two 256 word by 4 bit read only memories. From the viewpoint of printed circuit board area this is the least costly solution requiring just two integrated circuits; by comparison the tri-state coupler array in figure 5.3.12 requires 6 integrated circuits.

A further advantage of the radix-2 factorization is that each factor matrix has either 1 or 2 one's in each row and column. Where just 1 one appears on a row no operation need be performed; where 2 one's appear the operations required are as follows:

1. Read memory at address N1 into latch 1
2. Read memory at address N2 into latch 2
3. Modulo-2 add contents of latch 1 to contents of latch 2
4. Write result to memory at address N2.

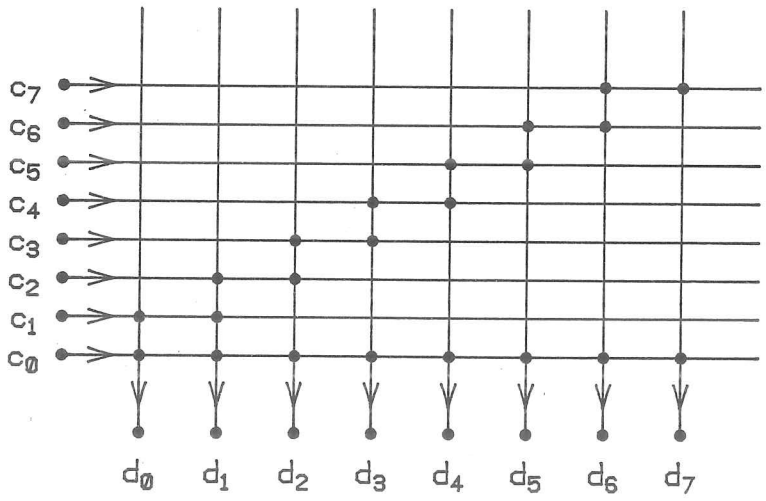
It follows that each transform stage involves an equal number of non-trivial operations; this is in complete contrast to the direct multiplication by $\underline{R}_N$. The effect is to simplify the processor control logic.



Address Bit Exchanges

Figure 5.3.11

22 gates



Tri-State Coupler Addressing Mechanism

Figure 5.3.12

The radix-2 factorization of $\underline{R}_N$ requires $N2^N$ memory reads and $N2^{N-1}$ memory writes; its associated arithmetic unit is just one EXCLUSIVE-OR gate and two latches. The radix-4 factorization of $\underline{R}_N$ (N-even) also requires $N2^N$ memory reads, however, it only requires $\frac{3N}{4} 2^{N-1}$ memory writes; its assoicated arithmetic unit requires five EXCLUSIVE-OR gates and four latches. Although radix-2 requires more memory operations and as a result is slower it was chosen because of its reduced hardware complexity and because it is not restricted to the case of N even.

5.4 A Software Reed-Muller Transform Algorithm

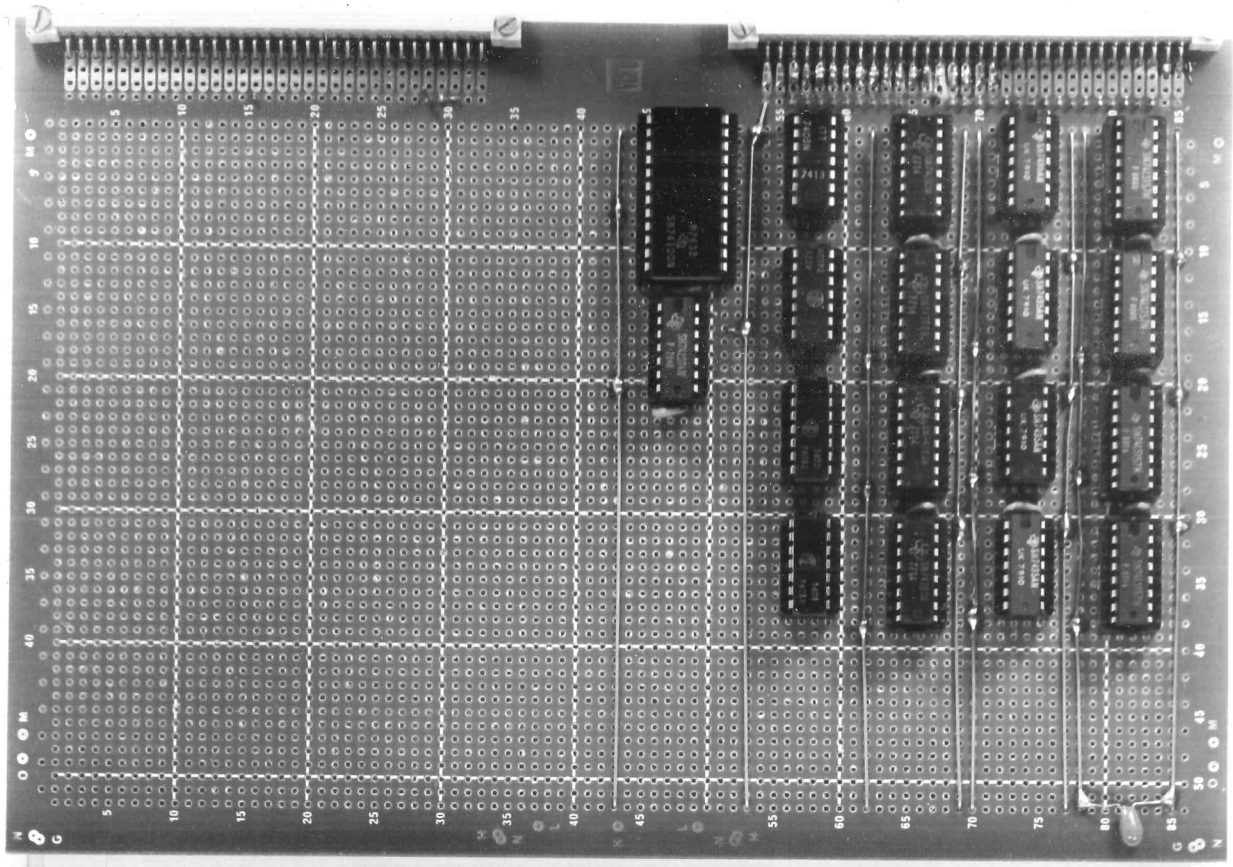
In figure 5.4.1 an algorithm to compute in place the Reed-Muller transform of a vector V is shown. The second argument to the procedure (BITS) specifies the number of bits on which the function is defined; it follows that vector V contains 2^{BITS} elements. The algorithm like the hardware processor is based on the radix-2 factorization of the RM transform.

The hardware processor was constructed using Schottky transistor-transistor logic; the software algorithm was coded in the BCPL systems programming language and a check was made on the quality of the generated code. The hardware processor was found to be 160 times faster than a Nova 820.

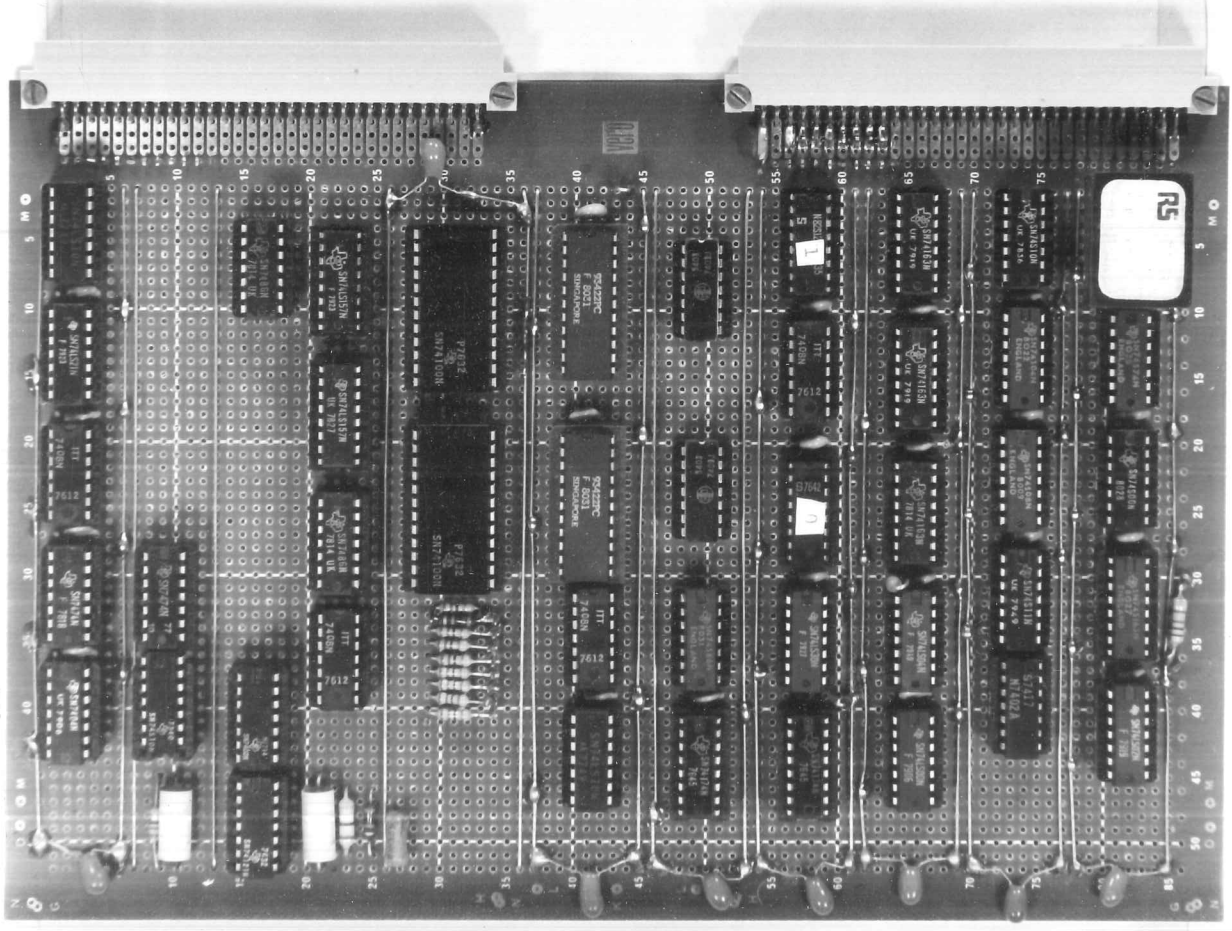
```
procedure RMT(V, BITS)

   declare V vector
   declare GROUPS, STEP, GPNO, GPSIZE, OFFSET, BITS integer
   declare ADDR, ADDR1, ADDR2, K integer

   for STEP=0 to BITS-1 do
     begin
       GROUPS := 2**(BITS-1-STEP)
       for GPNO=0 to GROUPS-1 do
         begin
           OFFSET := 2**STEP
           GPSIZE := OFFSET*2
           for K=0 to OFFSET-1 do
             begin
               ADDR := GPNO*GPSIZE
               ADDR1 := ADDR+K
               ADDR2 := ADDR1+OFFSET
               V(ADDR2) := V(ADDR2) EXOR V(ADDR1)
             end
         end
     end
end
```

Algorithm to compute Reed-Muller Transform of vector V.
Transform is computed over 2**BITS points.

Figure 5.4.1



D.M.A. Address Generator Board



Arithmetic and Control Board

REED-MULLER TRANSFORM PROCESSOR

6. Selected Examples of Free Quantization Law Filter Designs

6.1 Introduction

In this chapter three examples of the use of free quantization law techniques in the design of digital filters are presented. The first example is a digital differentiator synthesised from a second order recursive section. The second example is a sixth order low-pass filter; this filter is designed to be optimally linear phase in the passband. The final example is a sixth order band-pass filter of the elliptic type. In each case the SNR performance is compared with a similar fixed point arithmetic filter. The approximation error under sinusoidal input signal conditions is also considered.

6.2 Example 1

Wide-Band Digital Differentiator

This first example, a wide band digital differentiator, is realised using a single canonic form second order section. The approximation method used to obtain the pole and zero positions is described by Steiglitz [94]. The pole and zero positions are given in figure 6.2.1.

Poles	Zeroes
-0.14240300 + j 0.0	1.00000000 + j 0.0
-0.71698670 + j 0.0	-0.67082621 + j 0.0

figure 6.2.1

The coefficient values which realise this pole-zero pattern, using the canonic form in figure 6.2.3, are given in figure 6.2.2

Pole Coefficients	Zero Coefficients
K_1 = 0.859389200	L_1 = -0.329173790
K_2 = 0.102101057	L_2 = -0.67082621

figure 6.2.2

Using the special case of the complex convolution integral, equation 2.4.1, the integrated power gain from the filter input to each of the filters' variables can be found. Figure 6.2.4 shows the power level (variance) at each of the filters' variables assuming a unit power (variance) input signal; these values are required in order to assign to each filter variable an optimal quantization law. The integrated power gain from input to output is also required when calculating the SNR performance of the equivalent fixed point arithmetic (uniform quantization law) filter.

First a theoretical estimate of the SNR performance of the equivalent fixed point arithmetic filter will be obtained; a gaussian input process will be assumed. Referring to figure 6.2.3, following each

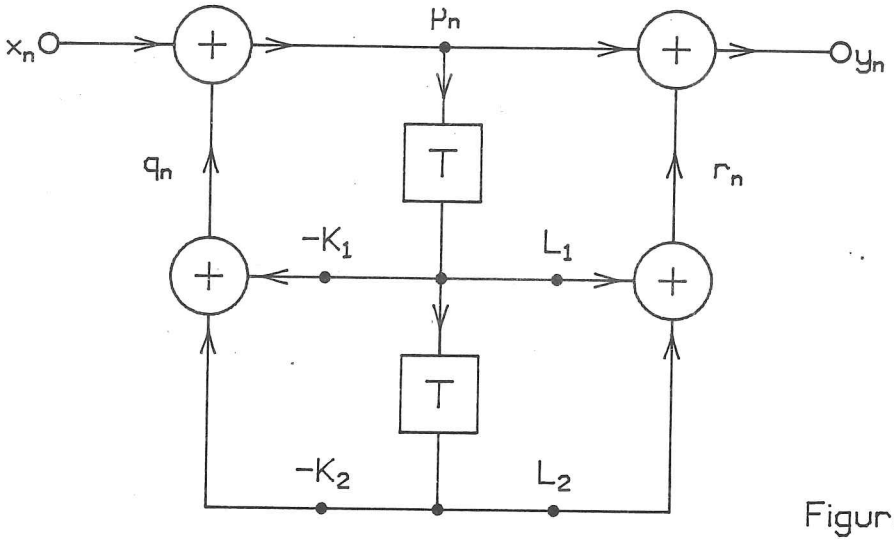


Figure 6.2.3

Variable	Variance
x_n	$\sigma_x^2 = 1.000000$
p_n	$\sigma_p^2 = 2.578280$
y_n	$\sigma_y^2 = 2.488641$
q_n	$\sigma_q^2 = 1.578228$
r_n	$\sigma_r^2 = 0.551703$

figure 6.2.4

of the four coefficient multiplications a noise source is introduced. Additionally, a noise source appears at the input due to the input quantizer. Each of the five noise sources has a variance of:

$$\frac{q^2}{12} \tag{6.2.1}$$

where q, the optimum step size for a uniform quantizer applied to a gaussian process, was found in section 2.3. Following the reasoning in section 2.7 q should be chosen to match the filter variable with the largest variance (i.e. p_n). It follows from the signal flow graph (figure 6.2.3) and from equation 6.2.1 that the noise power at the filters' output is given by:

$$\sigma_{yn}^2 = 3\,\frac{q^2}{12}\,\frac{\sigma_y^2}{\sigma_x^2} + 2\,\frac{q^2}{12} \tag{6.2.2}$$

Now for a 255-level unit variance quantizer q = 0.030856 and so for a variance value of 2.578230 (σ_p^2) q = 0.049545. Substituting for all

the variables in equation 6.2.2 the output noise power is found to be:

$$\sigma_{yn}^2 = 1.936 \times 10^{-3} \qquad (6.2.3)$$

Now the output signal power is given by:

$$\sigma_{ys}^2 = \sigma_y^2 = 2.488641 \qquad (6.2.4)$$

and hence the output signal to noise ratio is given by:

$$SNR_O = 10 \log_{10} \frac{\sigma_{ys}^2}{\sigma_{yn}^2} \text{ dB} = 31.0 \text{ dB} \qquad (6.2.5)$$

The corresponding result for the gaussian optimised filter will now be obtained. Again there are five noise sources; they add to the variables x_n, p_n, q_n, r_n and y_n. The variance of each is proportional to the variance of the signal to which it adds. Denoting by σ_u^2 the variance of the noise source which adds to x_n and using the aforementioned proportionality it follows that the output noise power is given by the equation:

$$\sigma_{yn}^2 = \frac{\sigma_y^2}{\sigma_x^2}\left[\sigma_u^2 + \frac{\sigma_p^2}{\sigma_x^2}\sigma_u^2 + \frac{\sigma_q^2}{\sigma_x^2}\sigma_u^2\right] + \left[\frac{\sigma_y^2}{\sigma_x^2}\sigma_u^2 + \frac{\sigma_r^2}{\sigma_x^2}\sigma_u^2\right] \qquad (6.2.6)$$

the signal power at the output is given by:

$$\sigma_{ys}^2 = \frac{\sigma_y^2}{\sigma_x^2}\sigma_x^2 \qquad (6.2.7)$$

and the output signal to noise ratio is:

$$SNR_O = 10 \log_{10} \frac{\sigma_x^2}{\sigma_u^2} \frac{1}{1 + \frac{\sigma_p^2 + \sigma_q^2}{\sigma_x^2} + \frac{\sigma_r^2}{\sigma_y^2}} \text{ dB}$$

$$= SNR_I - 10 \log_{10}\left[1 + \frac{\sigma_p^2 + \sigma_q^2}{\sigma_x^2} + \frac{\sigma_r^2}{\sigma_y^2}\right] \text{ dB} = 35.93 \text{ dB} \qquad (6.2.8)$$

Both results were validated by experiment. An improvement in signal to noise ratio of approximately 5dB has been obtained by using a gaussian optimised filter. If the fixed point arithmetic filter were to employ one extra bit throughout its arithmetic operations it would give an equal signal to noise ratio. The implication is that the gaussian optimised filter has gained one bit of arithmetic precision.

Now attention turns to how the gaussian optimised wide band differentiator behaves under sinusoidal input.

The differentiator, which is optimised for a unit variance gaussian input process, was driven by a unit amplitude sinusoid. The output signal to noise ratio as a function of frequency was measured; the results are shown in figure 6.2.5 by the solid curve. The dotted curve shows how the SNR performance of the equivalent fixed point arithmetic filter varies with frequency when it is driven such that the signal at p_n occupies the full dynamic range.

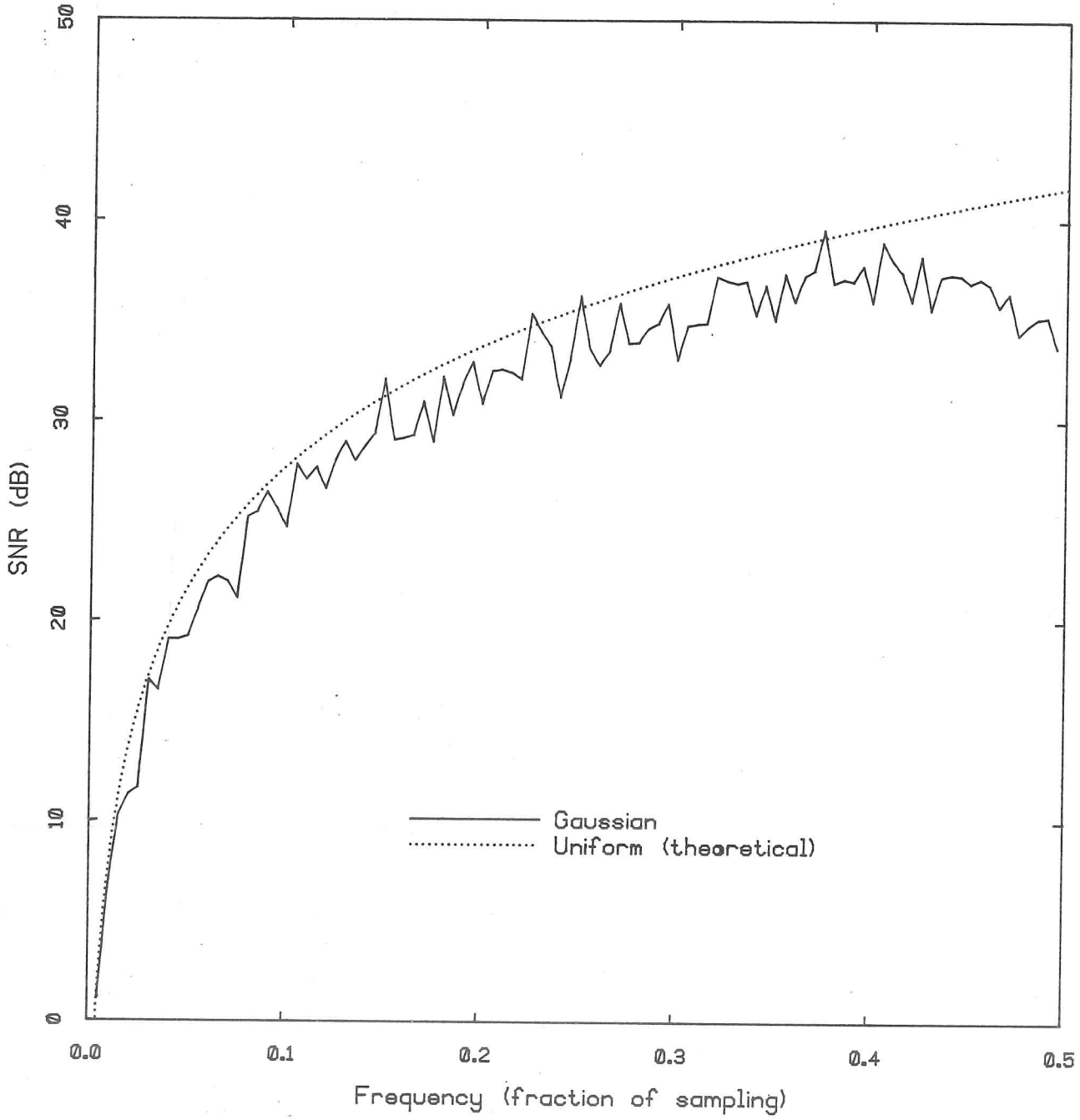
Ideally the differentiator would have a magnitude response, $A(\omega)$, and a phase response, $\Phi(\omega)$, given by:

$$A(\omega) = \frac{\omega}{\pi} \qquad (6.2.9)$$

and

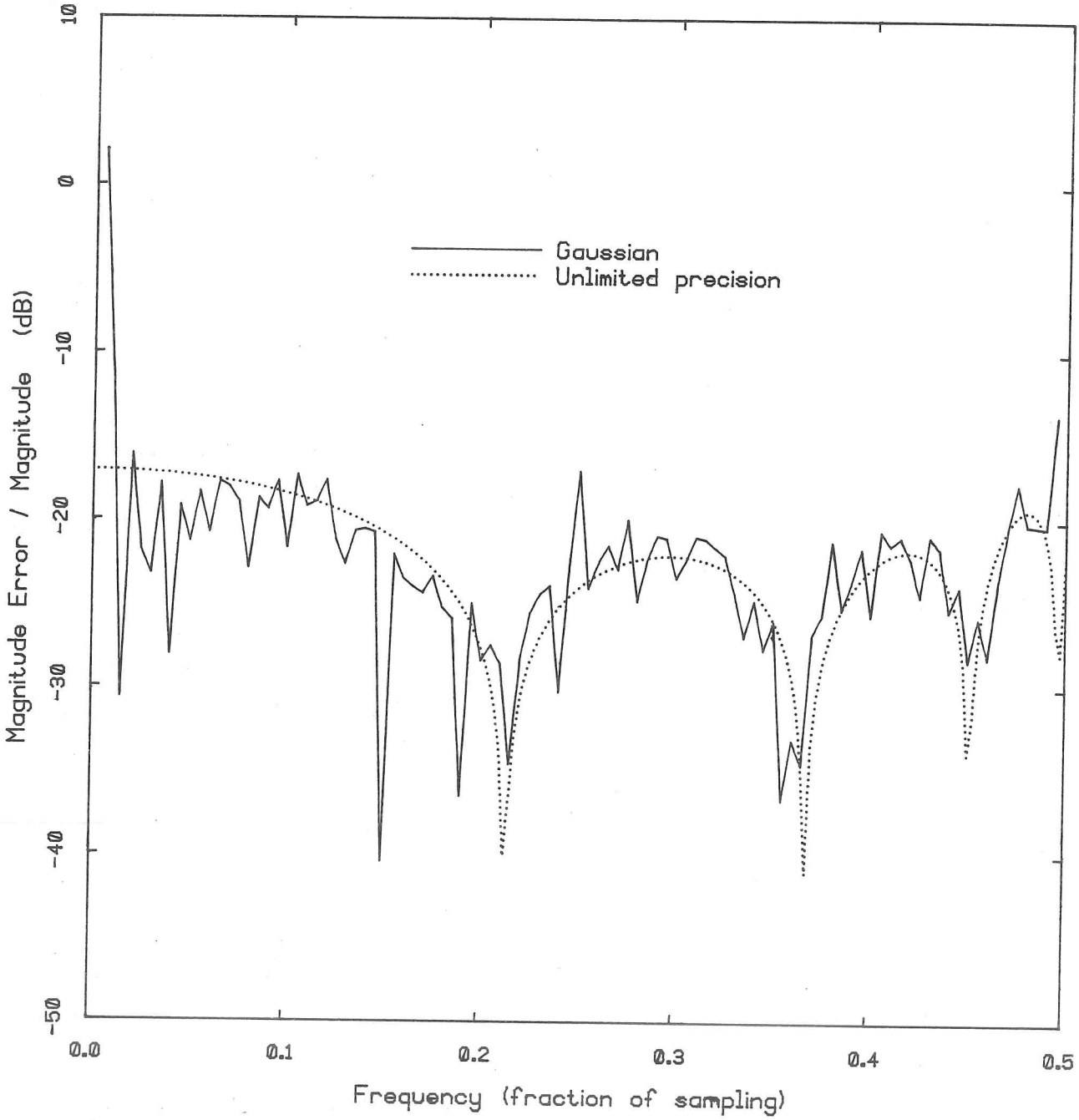
$$\Phi(\omega) = \frac{\pi-\omega}{2} \qquad (6.2.10)$$

however, the pole-zero pattern in figure 6.2.1 can only approximate this. When a p-z pattern is implemented in digital hardware, be it using gaussian optimised or fixed point arithmetic techniques, still further approximation error must be expected. The graphs in figures 6.2.6 and 6.2.7 show respectively the fractional error in the magnitude and phase response as a function of frequency. The dotted curve is obtained by assuming an unlimited arithmetic precision.



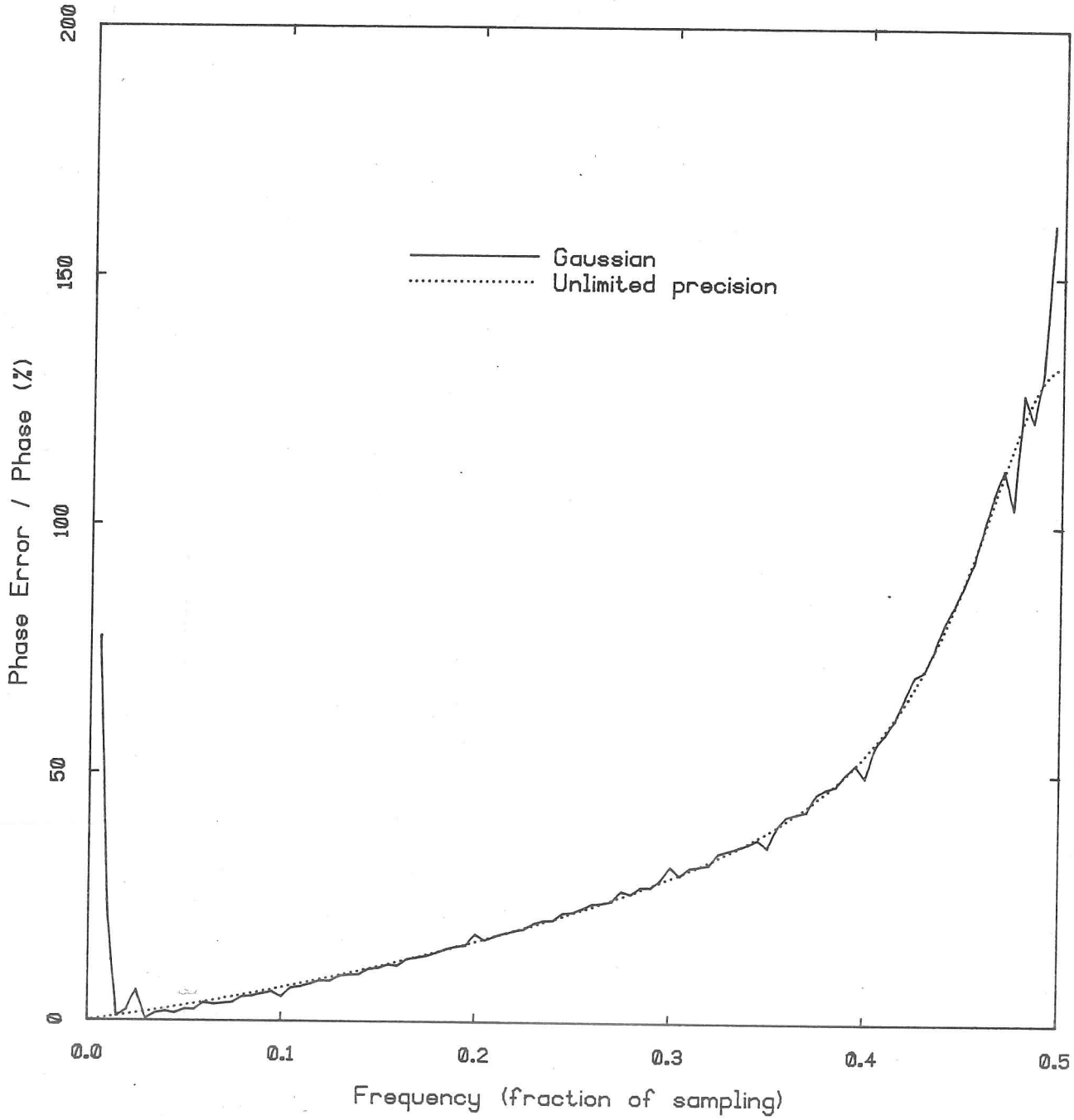
Signal to Noise Ratio as a function of frequency for a second order wide band differentiating filter implemented using both uniform and gaussian optimal quantization laws.

Figure 6.2.5



Magnitude approximation error as a function of frequency for a gaussian optimised second order wide band differentiating filter.

Figure 6.2.6



Phase approximation error as a function of frequency for a gaussian optimised second order wide band differentiating filter.

Figure 6.2.7

6.3 Example 2

Sixth Order All-Pole Low-Pass Filter

This example is a sixth order all-pole low-pass filter which has the frequency response shown in figure 6.3.7a. The phase response, which is shown in figure 6.3.8a, is designed to be optimally linear in the minimum mean sqaure sense from d.c. to ω = 0.6772 radians/s. The implementation chosen is a cascade of three second order all-poles. A list of the pole positions is given in figure 6.3.1. This pole pattern was obtained by Thajchayapong (95)

Section	Pole Position
1	0.68796454 $\pm$ j 0.16338426
2	0.64608842 $\pm$ j 0.62217429
3	0.57429475 $\pm$ j 0.36691520

figure 6.3.1

For each all-pole the coefficients are as defined in section 3.2; they have the following values:

Section i	K_{i1}	K_{i2}
1	1.37592	-0.49999
2	1.29217	-0.80453
3	1.14858	-0.45444

figure 6.3.2

The sections have been placed in order of decreasing integrated power gain; section 1 receives the input signal. In figure 6.3.3 the filter is shown with each of its variables marked; the variances at each of its variables are given in the table in figure 6.3.4.

Variable	Variance
x_n	$\sigma_x^2 = 1.0000$
a_n	$\sigma_a^2 = 8.4081$
b_n	$\sigma_b^2 = 61.3192$
y_n	$\sigma_y^2 = 598.4839$
p_n	$\sigma_p^2 = 3.1019$
q_n	$\sigma_q^2 = 25.5246$
r_n	$\sigma_r^2 = 158.8111$

figure 6.3.4

In addition, the integrated power gain values in figure 6.3.5 will be required.

Parameter	Value
G_{13}	598.453900
G_{23}	43.030470
G_{33}	3.313118

figure 6.3.5

The parameter G_{ij} is the integrated power gain from the input of section i to the output of section j.

For the case of the gaussian optimised filter the quantization laws at the different filter variables are chosen according to the variance values in figure 6.3.4; each quantizer has a shape similar to the one shown in figure 2.2.2. Denoting by σ_u^2 the noise produced by the input quantizer and using a method of noise analysis similar to the one proposed in chapter 2 the output noise power is found to be:

$$\sigma_{yn}^2 = G_{12}\,\sigma_u^2 \left[1 + \frac{\sigma_p^2}{\sigma_x^2} + \frac{\sigma_a^2}{\sigma_x^2}\right] \qquad \text{1st section and input}$$

$$+ G_{23}\,\sigma_u^2 \left[\frac{\sigma_q^2}{\sigma_x^2} + \frac{b^2}{\sigma_x^2}\right] \qquad \text{2nd section}$$

$$+ G_{33}\,\sigma_u^2 \left[\frac{\sigma_r^2}{\sigma_x^2} + \frac{\sigma_y^2}{\sigma_x^2} \right] \qquad \text{3rd section} \tag{6.3.1a}$$

Substituting values the output noise power is found to be:

$$\sigma_{yn}^2 = 13732.5955\ \sigma_u^2 \tag{6.3.1b}$$

The signal power at the filters' output is:

$$\sigma_{ys}^2 = 598.4539\ \sigma_x^2 \tag{6.3.2}$$

and hence the output signal to noise ratio is given by:

$$SNR_O = SNR_I - 13.6072\ \text{dB} = 30.4\ \text{dB} \tag{6.3.3}$$

The same filter, when implemented using fixed point arithmetic, requires three scaling multipliers; one at the input to each section. The multiplier values are chosen to give unit variance at the output of each section; a unit variance input process is assumed. The values are as follows:

Section	Scale Factor
1	$S_1 = 0.34487$
2	$S_2 = 0.37030$
3	$S_3 = 0.32008$

figure 6.3.6

Each section in a fixed point arithemtic implementation is subjected to three sources of noise; these are due to the two coefficient multipliers and the input scaling multiplier. In addition, the first section sees a noise source due to the input quantizer. The noise which appears at the output of the last section is given by:

$$\sigma_{yn}^2 = G_{13}\, S_2^2\, S_3^2\, \frac{3q^2}{12} + G_{23}\, S_3^2\, \frac{3q^2}{12}$$

$$+ G_{33}\, \frac{3q^2}{12} + \frac{q^2}{12} \qquad (6.3.4)$$

$$= \frac{q^2}{12}\left[3\, G_{13}\, S_2^2\, S_3^2 + 3\, G_{23}\, S_3^2 + 3\, G_{33} + 1\right]$$

Now, the output signal power is given by:

$$\sigma_{ys}^2 = \sigma_x^2 \qquad (6.3.5)$$

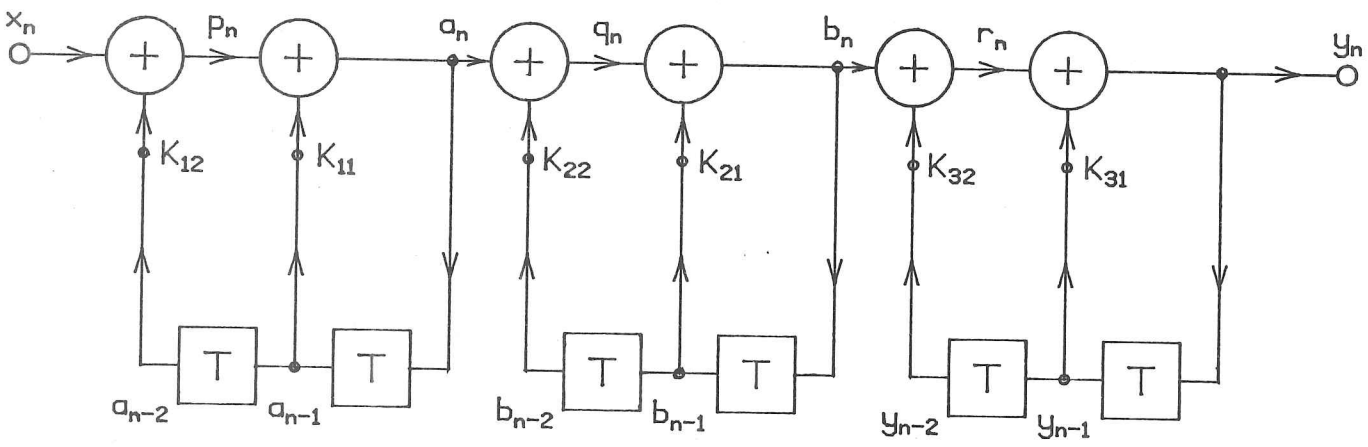
From equations 6.3.4 and 6.3.5 it follows that the output signal to noise ratio is:

$$SNR_o = 23.5 \text{ dB} \qquad (6.3.6)$$

The SNR figures given in 6.3.3 and 6.3.6 were verified by experiment.

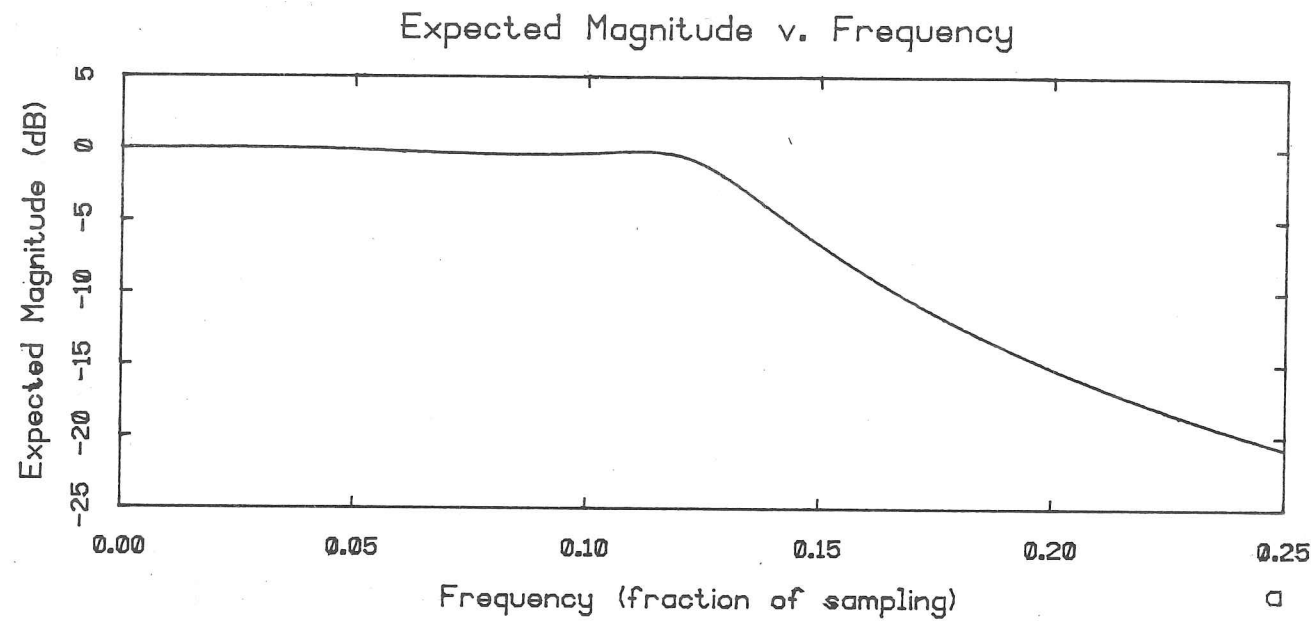
The improvement in signal to noise ratio of 7dB using free quantization law techniques shows that a 127 level gaussian optimal filter would perform as well as a 255-level fixed point arithmetic filter. With 127 levels this filter could be made to operate at very high data sampling rates using a r.o.m. based or similar implementation.

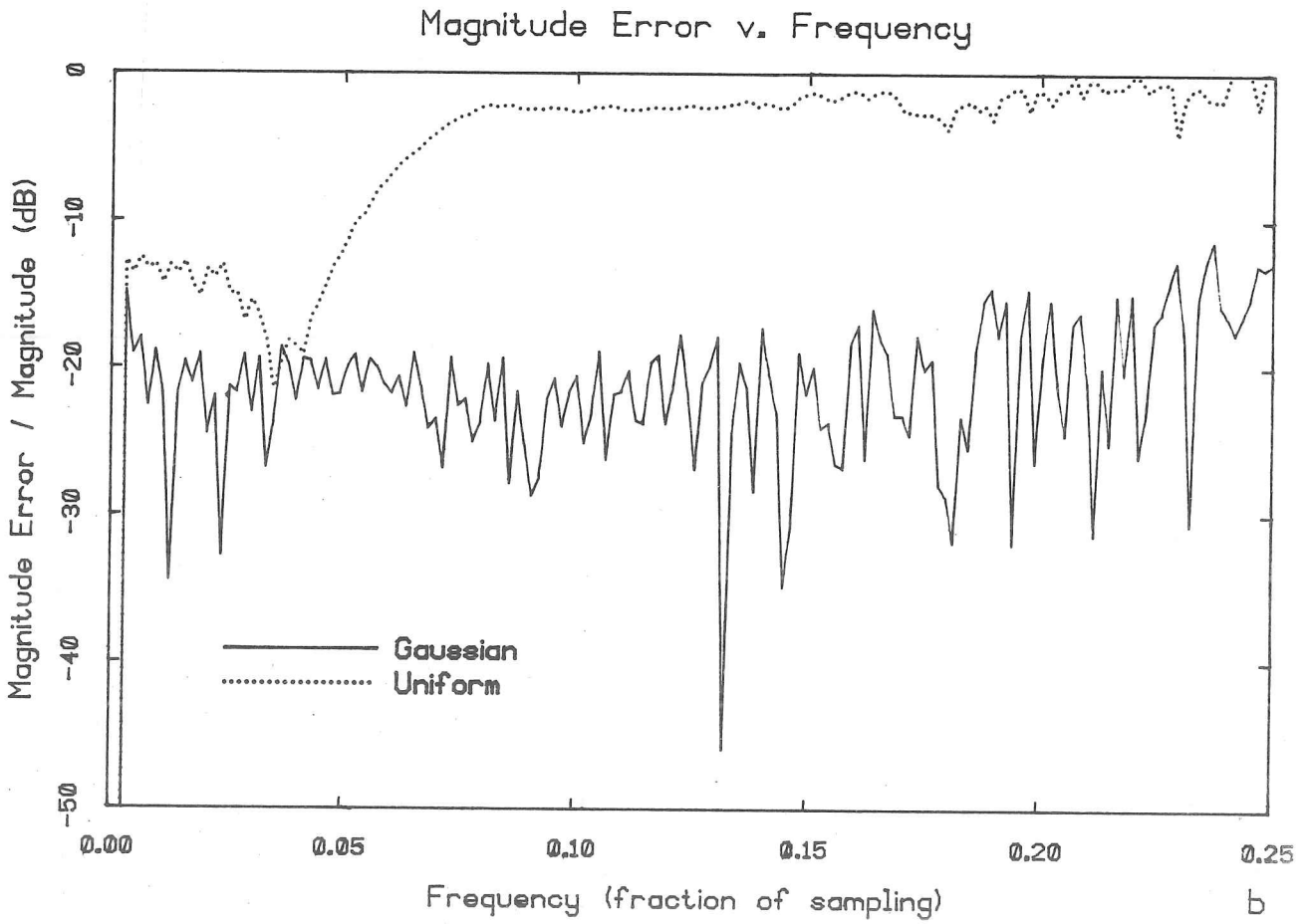
This filter was also tested under sinusoidal input signal conditions. The graphs in figure 6.3.7b and 6.3.8b show as a function of frequency the fractional magnitude and phase errors respectively for both the fixed point arithmetic and the gaussian optimised filter. At all frequencies the gaussian optimised filter is seen to be the more accurate. Finally, the graph in figure 6.3.9 shows the variation of SNR with frequency. It can be seen that the gaussian optimised filter behaves well in the pass band.



Cascade Realisation of Sixth Order All-Pole Filter

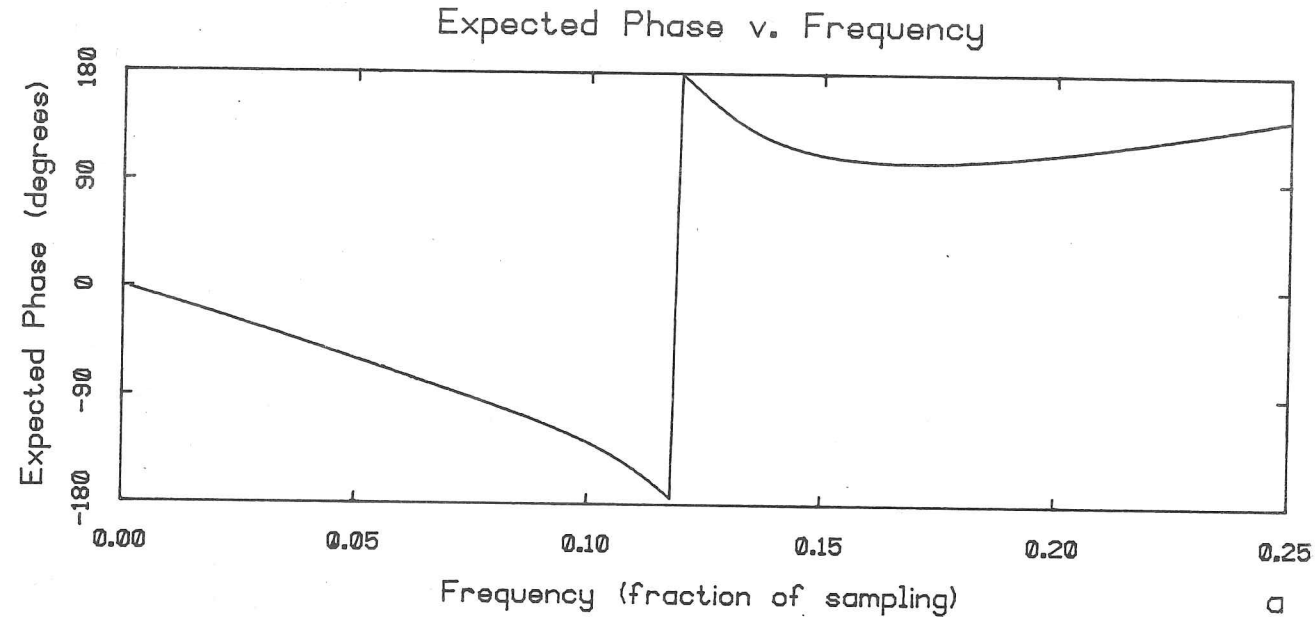
Figure 6.3.3

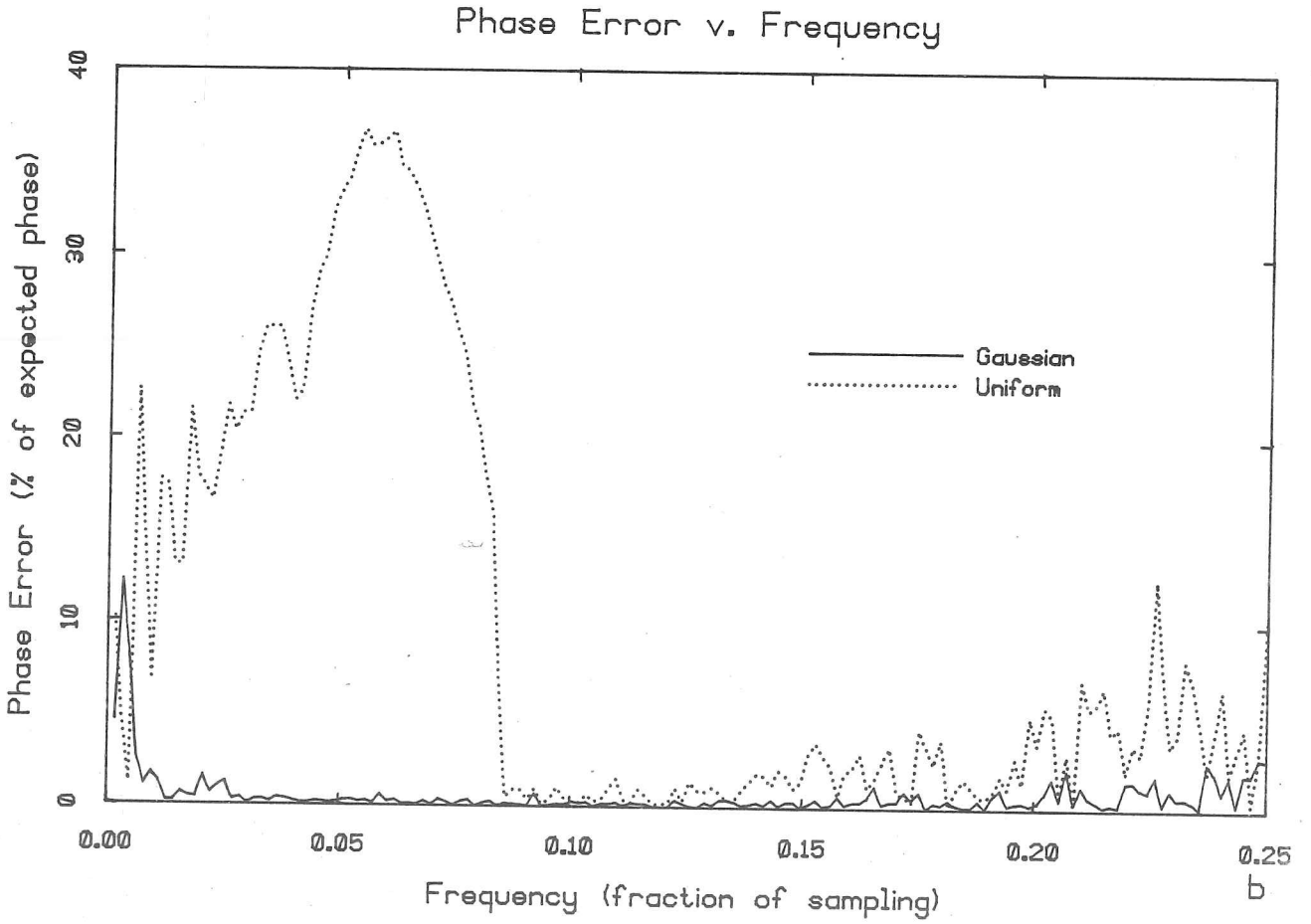




Expected magnitude and magnitude approximation error as a function of frequency for a sixth order all-pole low-pass filter implemented using both gaussian optimised and uniform quantization laws.

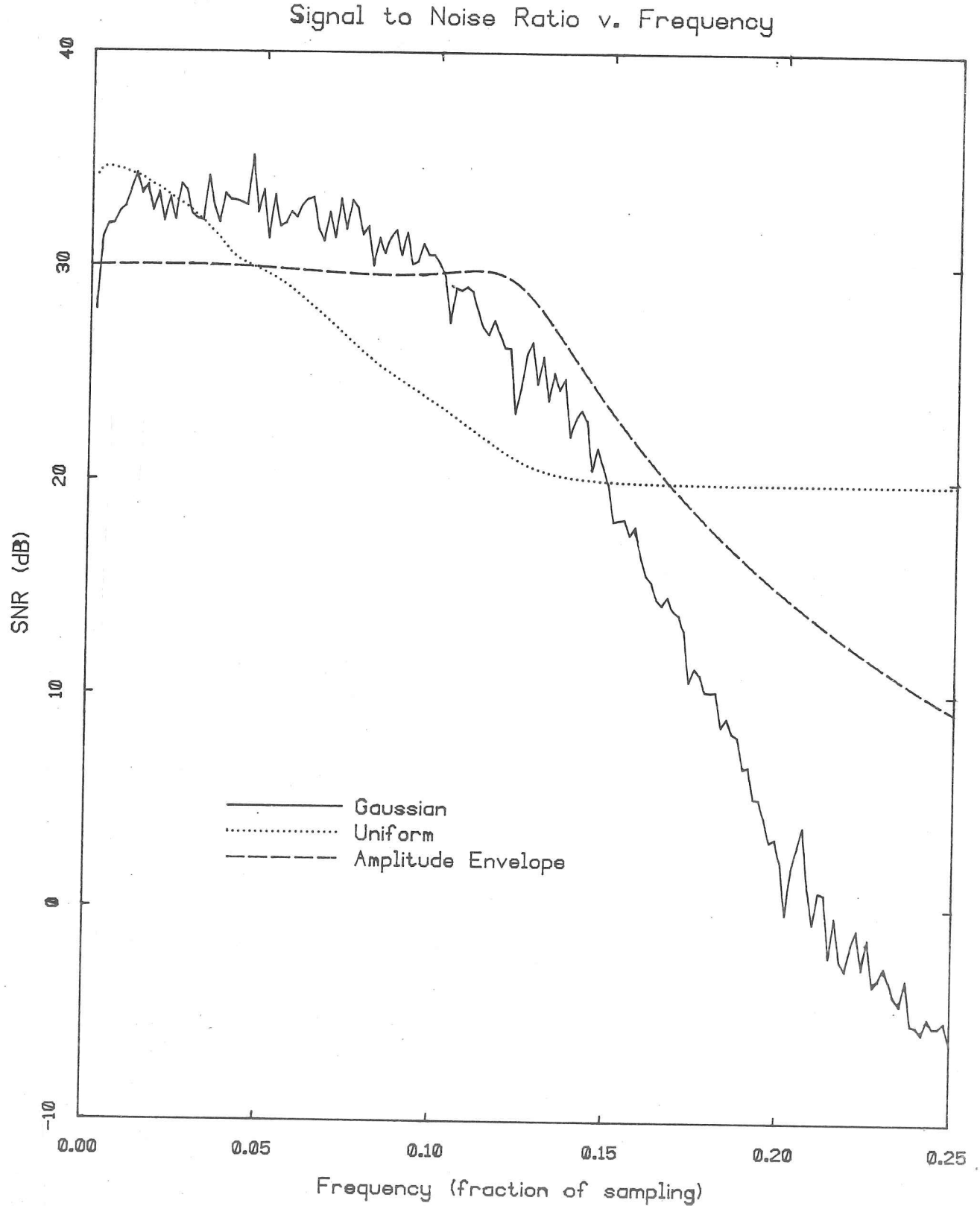
Figure 6.3.7





Expected phase and phase approximation error as a function of frequency for a sixth order all-pole low-pass filter implemented using both gaussian optimised and uniform quantization laws.

Figure 6.3.8



Signal to Noise Ratio as a function of frequency for a sixth order all-pole low-pass filter implemented using both gaussian optimal and uniform quantization laws.

Figure 6.3.9

6.4 Example 3

Sixth Order Elliptic Band-Pass Filter

This last example, a sixth order band pass filter, was designed using the programs described in [96]. Its specification is as follows:

Lower stopband edge, f_{al} = 0.974 radians/s

Lower passband edge, f_{cl} = 1.100 radians/s

Upper passband edge, f_{cu} = 2.042 radians/s

Upper stopband edge, f_{au} = 2.168 radians/s

Stopband attenuation $\geqslant$ 20.750 dB

Passband ripple $\leqslant$ 1.000 dB

Sampling frequency, f_s = 2π radians/s

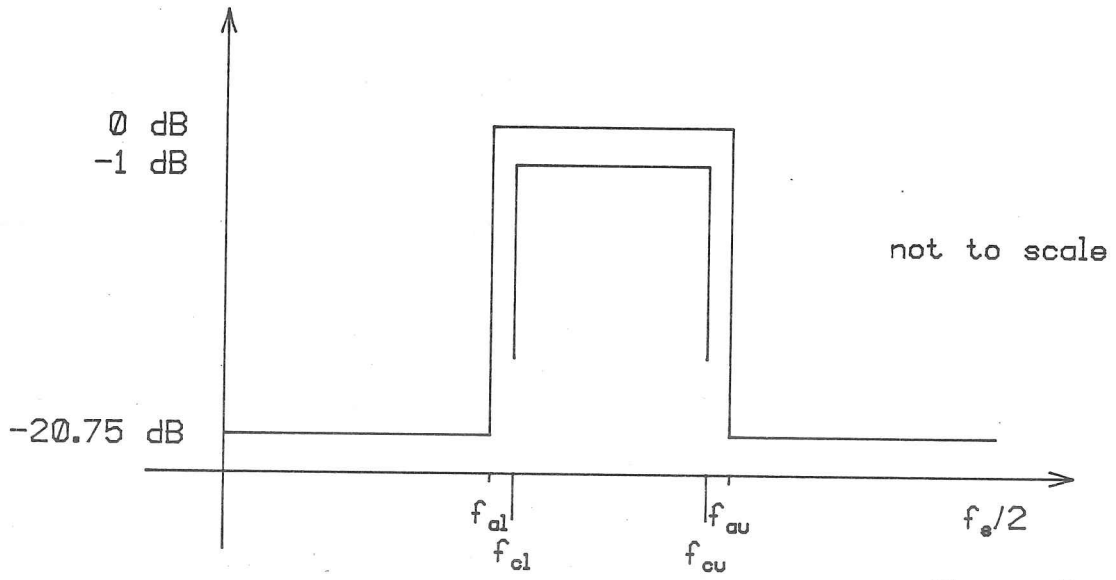


Figure 6.4.1

This specification is met with a sixth order elliptic filter. The implementation, which is shown in figure 6.4.2, is a cascade of three second order section. The pole-zero positions and the corresponding coefficients are given in the tables in figure 6.4.3.

Section	Pole Positions
1	0.000000 ± j 0.714691
2	+0.427169 ± j 0.831655
3	-0.427169 ± j 0.831655

Section	Zero Positions
1	±1.000000 + j 0.000000
2	+0.600000 ± j 0.800000
3	-0.600000 ± j 0.800000

	Pole Coefficients	
Section i	K_{i1}	K_{i2}
1	0.000000	+0.510785
2	+0.854337	+0.874124
3	-0.854337	+0.874124

	Zero Coefficients	
Section i	L_{i1}	L_{i2}
1	0.000000	-1.000000
2	1.200000	1.000000
3	-1.200000	1.000000

figure 6.4.3

Also shown in figure 6.4.2 are all the filters' variables; assuming a unit variance process at x_n, the variance to be expected at each of these variables is given in figure 6.4.4.

Section	Pole Positions
1	0.000000 ± j 0.714691
2	+0.427169 ± j 0.831655
3	-0.427169 ± j 0.831655

Section	Zero Positions
1	±1.000000 + j 0.000000
2	+0.600000 ± j 0.800000
3	-0.600000 ± j 0.800000

	Pole Coefficients	
Section i	K_{i1}	K_{i2}
1	0.000000	+0.510785
2	+0.854337	+0.874124
3	-0.854337	+0.874124

	Zero Coefficients	
Section i	L_{i1}	L_{i2}
1	0.000000	-1.000000
2	1.200000	1.000000
3	-1.200000	1.000000

figure 6.4.3

Also shown in figure 6.4.2 are all the filters' variables; assuming a unit variance process at x_n, the variance to be expected at each of these variables is given in figure 6.4.4.

Variable	Variance
x_n	1.000000
a_n	0.352998
p_n	1.35299
d_n	1.35299
y_n	4.08817
b_n	24.6172
q_n	26.4679
e_n	40.6032
z_n	8.85478
c_n	41.1611
r_n	42.5274
f_n	67.8222
$t_n = G_{13}$	18.0662

figure 6.4.4

In addition the following integrated power gain values are required in the calculation of signal to noise ratio.

$G_{23} = 2.40868$

$G_{33} = 1.51218$

figure 6.4.5

The variance values given in figure 6.4.4 are required in order to select optimal quantization laws for each of the filter variables in a gaussian optimised implementation.

For a gaussian optimised implementation the output noise power, σ_{tn}^2, is given by:

$$\sigma_{tn}^2 = G_{13}\,\sigma_u^2 \qquad \text{- input}$$

$$+ G_{13}\left[\sigma_a^2 + \sigma_p^2\right] + G_{23}\left[\sigma_d^2 + \sigma_y^2\right]$$

$$+ G_{23}\left[\sigma_b^2 + \sigma_q^2\right] + G_{23}\left[\sigma_e^2 + \sigma_z^2\right]$$

$$+ G_{33}\left[\sigma_c^2 + \sigma_r^2\right] + \left[\sigma_f^2 + \sigma_t^2\right]$$

$$= 472.2704\ \sigma_u^2 \qquad (6.4.1)$$

and the output signal power, σ_{ts}^2, by:

$$\sigma_{ts}^2 = 18.0662\ \sigma_x^2 \qquad (6.4.2)$$

and hence the output signal to noise ratio is given by:

$$SNR_O = SNR_I = 14.1732\ dB = 29.83\ dB \qquad (6.4.3)$$

The equivalent fixed point arithmetic filter requires the intersection scaling multipliers given in figure 6.4.6; these values were chosen to ensure that in each of the three sections the variance at the largest variable was one.

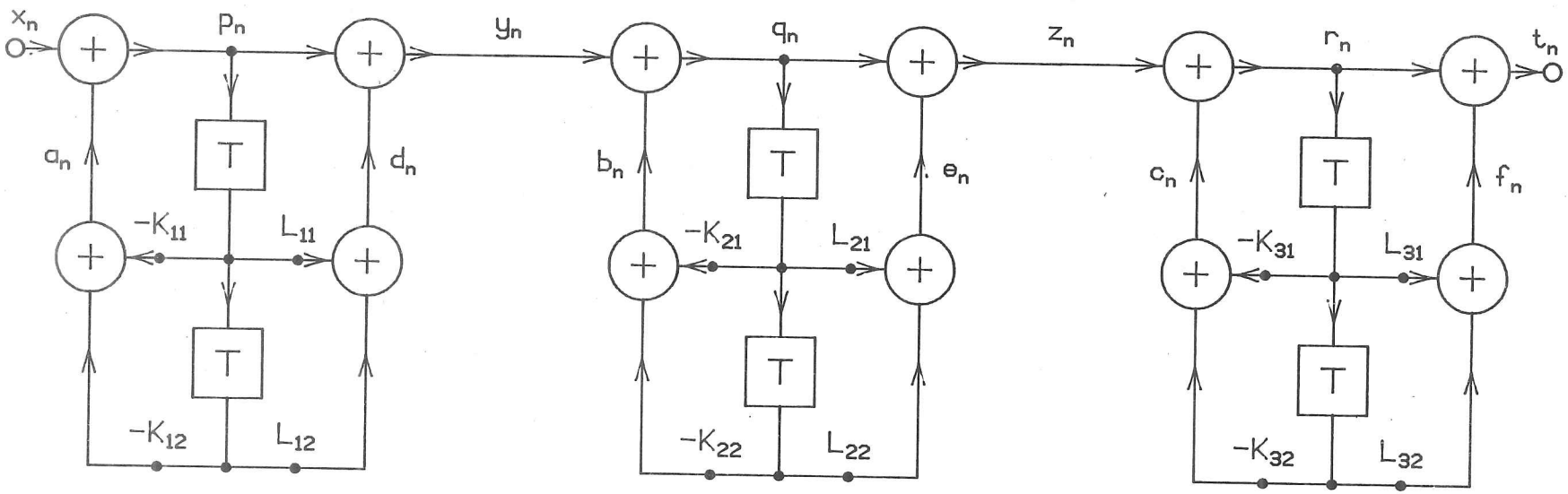
Section	Scale Factor
1	0.49458
2	0.31731
3	0.77374

figure 6.4.6

The output signal to noise ratio was calculated to be:

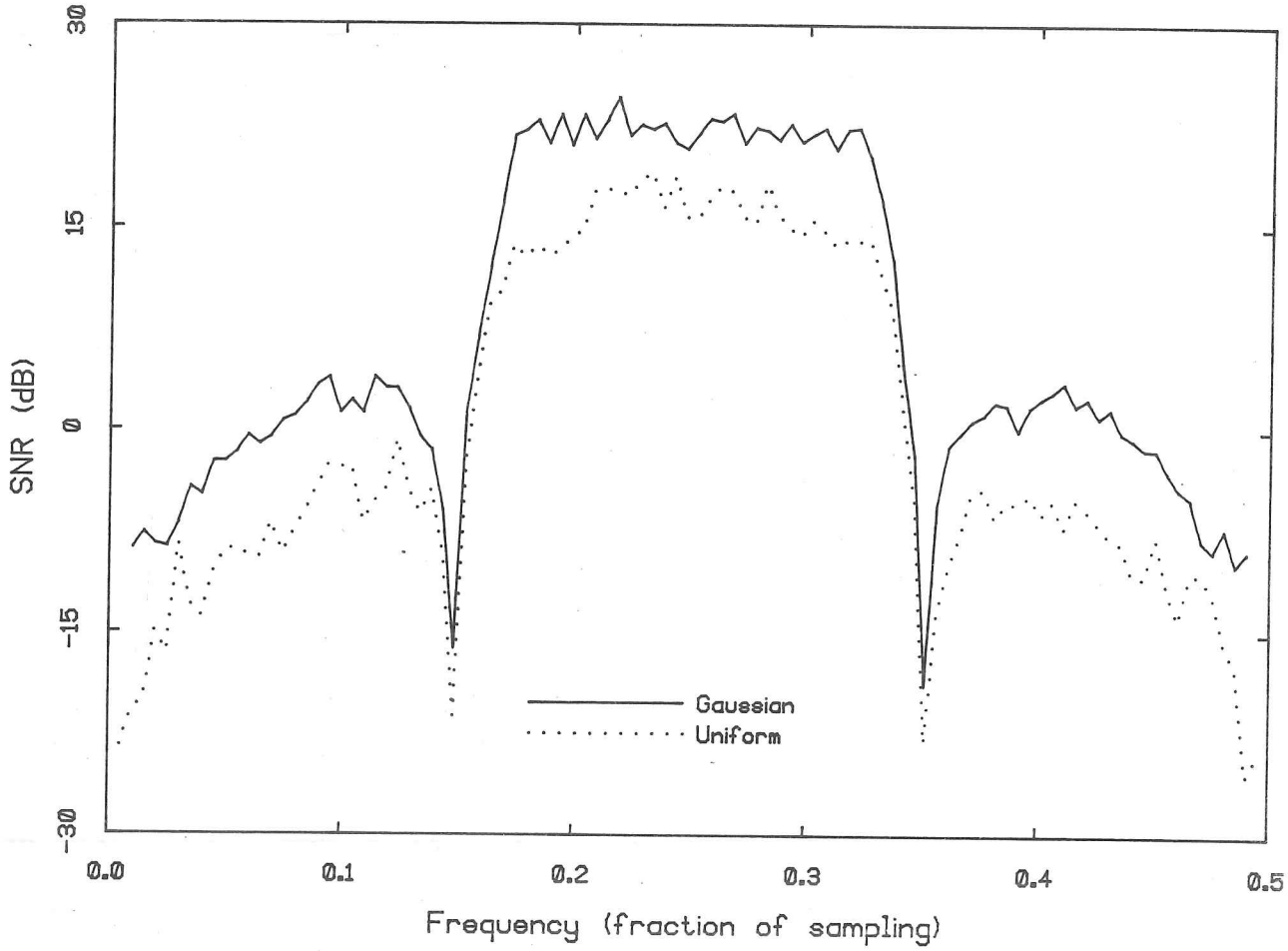
$$SNR_O = 28.64\ dB \qquad (6.4.4)$$

Under sinusoidal input the signal to noise ratio was found to vary with frequency as shown by the graph in figure 6.4.7. The variation of magnitude and phase error with frequency is shown respectively in figures 6.4.8 and 6.4.9. For comparison the same results are presented, as dotted curves, for the fixed point arithmetic filter.



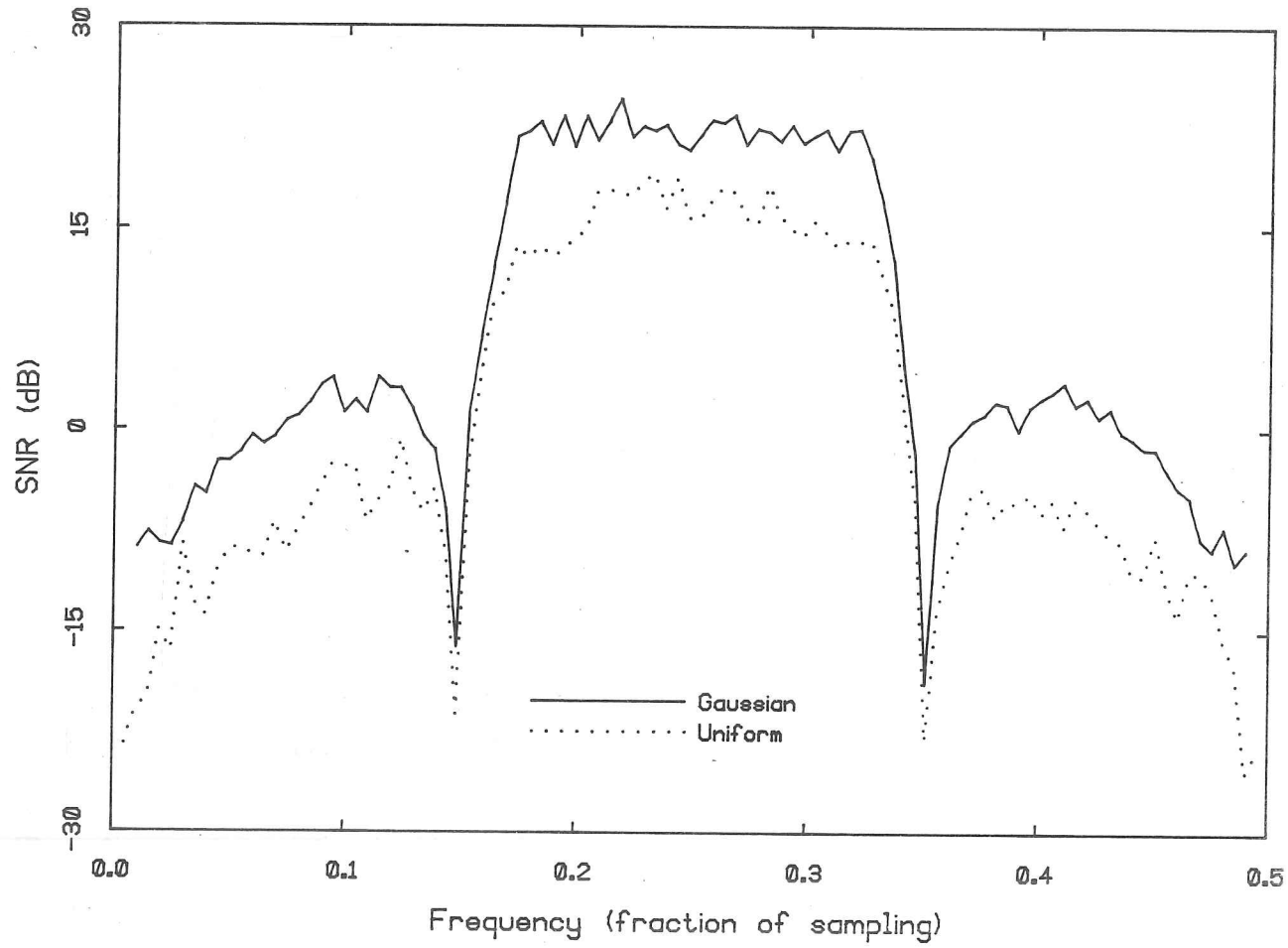
Cascade Realisation of a Sixth Order Filter

Figure 6.4.2



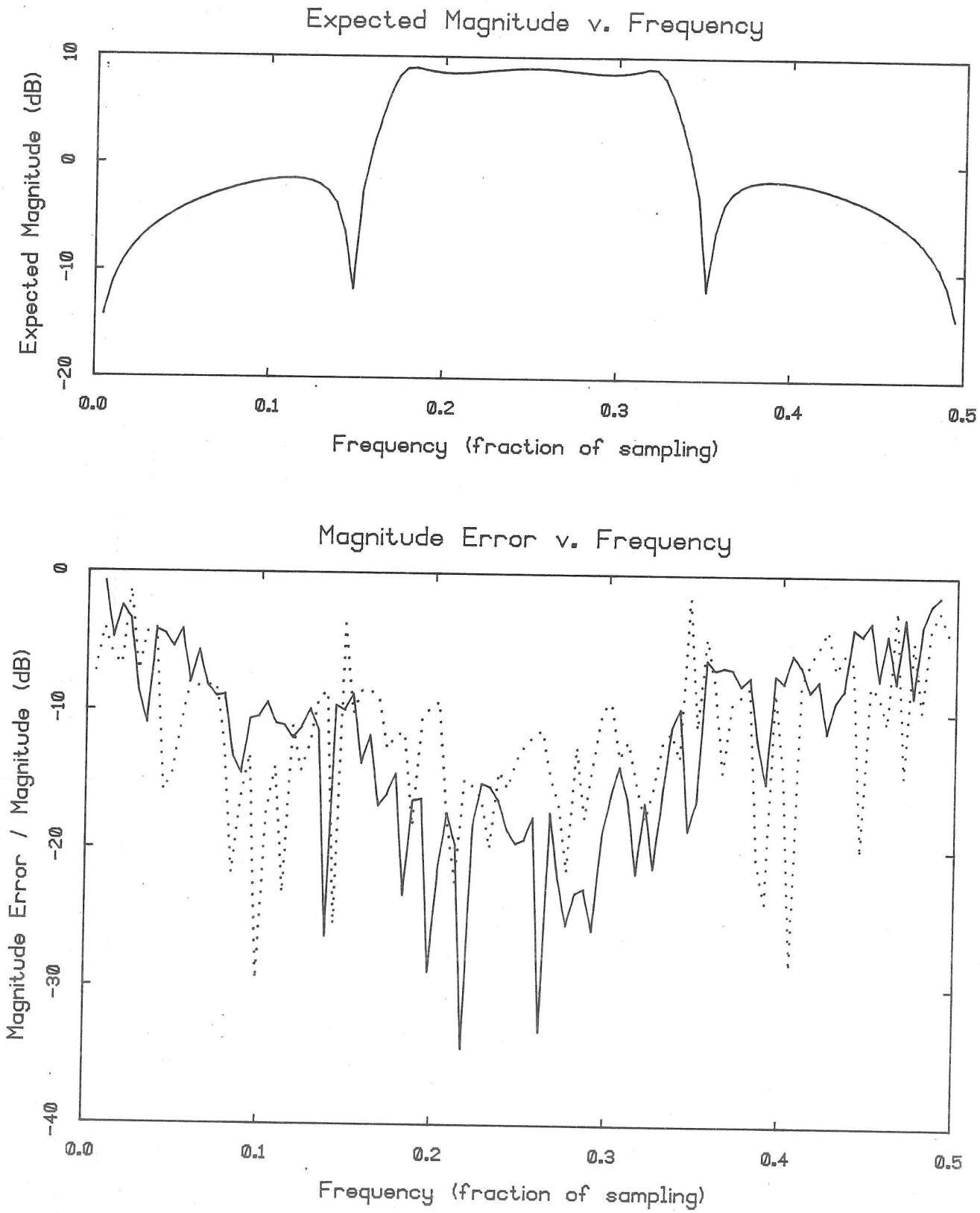
Signal to Noise Ratio as a function of frequency for a sixth order elliptic band-pass filter implemented using both gaussian optimised and uniform quantization laws.

Figure 6.4.7



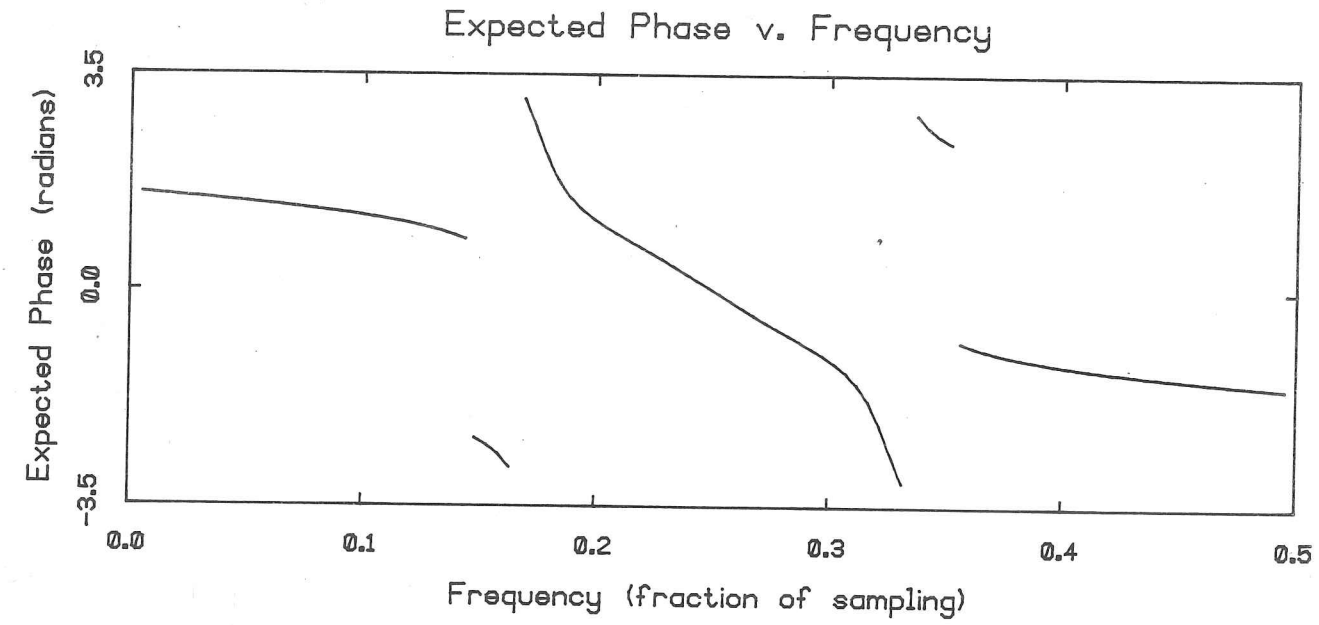
Signal to Noise Ratio as a function of frequency for a sixth order elliptic band-pass filter implemented using both gaussian optimised and uniform quantization laws.

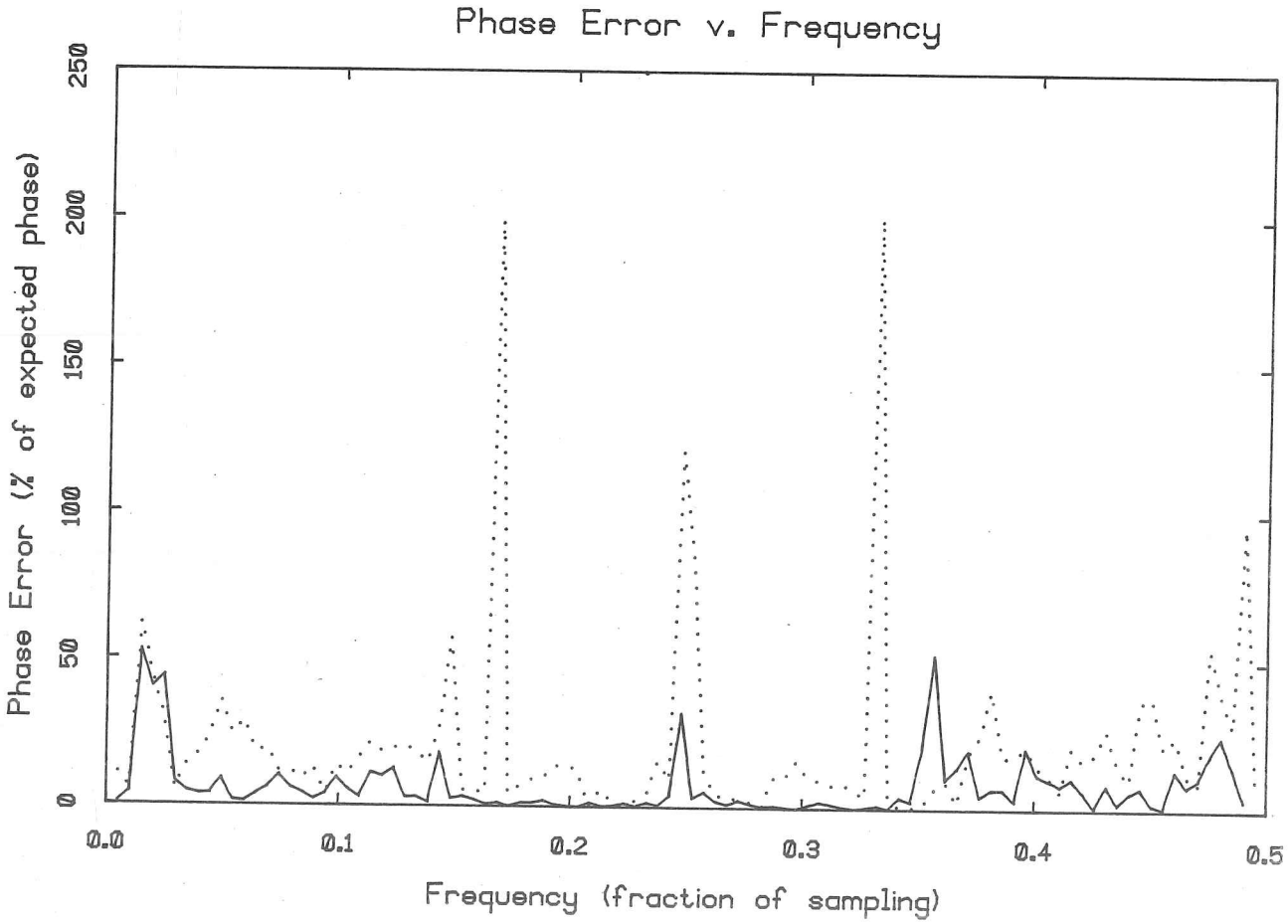
Figure 6.4.7



Expected magnitude and magnitude approximation error as a function of frequency for a sixth order elliptic band-pass filter implemented using both gaussian optimised and uniform quantization laws.

Figure 6.4.8





Expected phase and phase approximation error as a function of frequency for a sixth order elliptic band-pass filter implemented using both gaussian optimised and uniform quantization laws.

Figure 6.4.9

7. Conclusions and Suggestions for Further Work

7.1. Conclusions

This dissertation has been concerned with the design of high data sampling rate low distortion recursive digital filters. Different hardware configurations have been considered and it has been concluded that, for a given device technology, the fastest configuration is the finite state machine (f.s.m.). Further advantages of the f.s.m. configuration are that it is easily implemented using either read only memories (r.o.m.'s) or programmable logic arrays (p.l.a.'s), and that its data sampling rate does not depend on the size of its state set. The second advantage should be contrasted with the case of the fixed point arithmetic filter where an increase in wordlength - the analogue of state set size - results in a reduction in sampling rate. With an f.s.m. implementation of a filter it is important to minimise circuit size, and hence cost, by using as few states as possible consistent with meeting the signal to noise ratio and magnitude approximation error specifications. In turn this means that each state should be used to its best possible advantage.

A new filter design technique has been proposed; with this technique each variable in a filter is assigned its own individual quantization law. Each of these laws is chosen, using a minimum mean square error (m.m.s.e.) criterion, to be the optimal quantizer for the respective signal. Furthermore, each is chosen independently of all the others; this is justified by arguing that from the viewpoint of a given variable the remainder of the system is linear, but perturbed by noise.

An error model for this new filter, the so called free quantization law filter, has been developed and experimentally verified. It has been shown to be a more accurate model than the one for the fixed point arithmetic filter; it is reasoned that this is due to the distortion at the output of

an m.m.s.e.optimal quantizer being orthogonal to the signal. The model has been tested on first and second order all-poles, all-zeroes and sections. In particular, the model was found to maintain its accuracy with high-Q poles.

Simulations, on a general purpose computer, were performed on filters which were optimised for a gaussian input process. There are two reasons for this choice of process. Firstly, a linearly transformed gaussian process is itself a gauss-markov process, that is to say its sample statistics are gaussian. This means that only the decision and quantization levels for a unit variance gaussian optimal quantizer need be stored; the levels for a quantizer of arbitrary variance can be obtained by scaling. Secondly, in the latter stages of a filter, regardless of the statistics of the stochastic input process, the variables will exhibit gaussian sample statistics.

The free quantization law filter has been shown to give a better SNR performance than the equivalent fixed point arithmetic filter with the same number of states. Care was taken when making this comparison to ensure that the fixed point arithmetic filter was driven at a variance level consistent with producing the maximum possible output signal to noise ratio.

Sinusoidal input signal tests have been performed on gaussian optimised filters. These tests have indicated an improved approximation error performance compared with equivalent fixed point arithmetic filters. A justification for this, based on an 'expected coefficient' argument, has been presented for the case of a first order all-pole.

Limit cycles in free quantization law filters have been considered. For the first order all-pole the limit cycle amplitude was found to be below that of the equivalent fixed point filter. For the second order all-pole the limit cycles were found to be as much as twice the amplitude. This is, however, not a serious problem since with the f.s.m. implementation method limit cycles can be eliminated altogether by altering the next state table.

Three examples of actual filter designs have been presented, they were

selected from a wide range of tested examples. A certain SNR improvement was always obtained using free quantization law techniques. The size of the improvement was found to depend on whether the filter had any high-Q poles. This is an intuitively satisfying result because the lower the Q of the poles, the more similar each of the quantization laws becomes. As the laws become more similar so the filter tends to a fixed point arithmetic filter. A marked feature of the gaussian optimised filter is its low magnitude and phase approximation error compared with the fixed point arithmetic equivalent. In each example this was in evidence. In particular, very little error was observed where there were large changes in the magnitude response and discontinuities in the phase response; this is in complete contrast to the fixed point arithmetic case.

The f.s.m. implementation of a digital filter is particularly suited to very large scale integration (v.l.s.i.). This is because of the high degree of regularity in the logical expansion of a next state function. The complexity of the next state logic has been shown to depend on the following factors:-

1). The form of the next state logic function (minterm, ring-sum etc.),

2). The code assignment,

and 3). The type of delay store.

In this work the ring-sum form was used; this is because it offers some economies compared with the minterm form and because a generation of p.l.a.'s is becoming available which synthesises this form. The various code assignment algorithms to be found in the literature were reviewed and a new algorithm, which is specific to the ring-sum form, has been proposed. In the work so far a true output only delay store has been assumed.

The new code assignment algorithm requires the numerous computation of the Reed-Muller transform. A general purpose computer is not ideally suited to evaluating this transform and so it was decided to construct a special

purpose processor. A description of this processor has been given. The efficiency of the processor has been compared with an equivalent software algorithm.

7.2 Suggestions for Further Work

In this work a method for improving the SNR performance of a filter has been described. It is by no means clear that the method leads to the best possible SNR. A method is required for directly mapping from a filter specification to a finite state machine; this is the only way to gaurantee maximum possible SNR.

The last example in chapter 6, a sixth order elliptic band-pass filter, illustrated how a gaussian optimised filter is not only robust to sinusoidal input signals, but how it can in fact yield a better signal to noise ratio performance than its fixed point arithmetic equivalent. It is reasoned that this robustness derives from the m.m.s.e. optimal quantizer for a gaussian process being similar in shape to the constant fractional error quantizer for a sinusoid. This raises the question of whether the m.m.s.e. optimisation strategy is the correct choice.

Although multiple valued logic is currently of no practical engineering importance it should be considered as means for fabricating it are currently being researched. An advantage of multiple valued logic from the viewpoint of the code assignment problem is that it possesses a metric; this metric can be used to define a distance between different state codes. An algorithm for performing code assignment optimisation assuming a multiple valued logic next state function remains a matter for further research.

A problem still to be resolved is that of representing a second order recursive digital filter as a finite state machine. It is undesirable to represent it directly, as in chapter 4, because of the size of the state set.

Appendix A Experimental Techniques

A.1 Signal to Noise Ratio Measurement

Where experimental signal to noise ratio figures have been quoted in this dissertation, be they for stochastic or deterministic input signals, they have been obtained as described below.

Let F_I be an actual implementation of a digital filter and let F_R be the same filter implemented without limited arithmetic precision. The suffices I and R indicate implementation and reference respectively. In practice filter F_R would be iterated using the full floating point precision of a general purpose computer. Both filters begin an experimental run with the same initial conditions, and both are driven from the same signal source. The difference in the outputs from F_I and F_R will be called the roar error sequence; this is not the true error sequence since it may be in part correlated with the output of the reference filter. The experiment is shown in the system diagram in figure A.1.1.

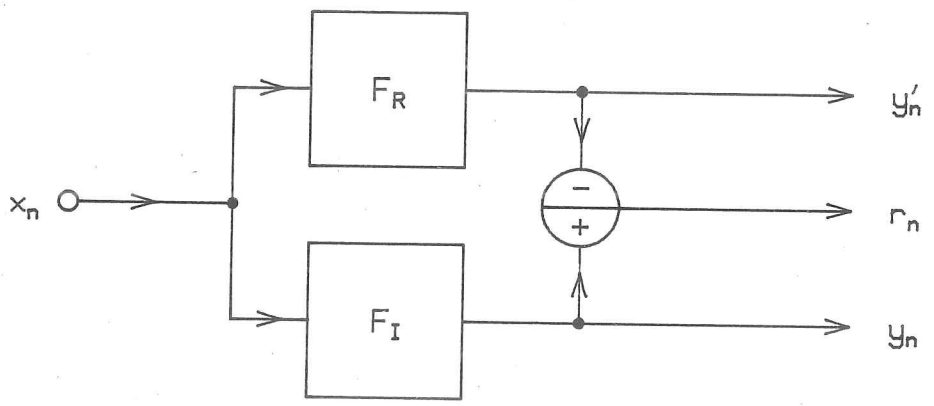


Figure A.1.1

Sequence y_n is the response of F_R to sequence x_n,

sequence $y_{\grave{n}}$ is the response of F_I to sequence x_n,

and sequence r_n is the roar error signal $y_n - y_{\grave{n}}$.

Since in part r_n is correlated with y_n, a new error sequence e_n (the true error sequence) may be defined as:

$$e_n = y_{\grave{n}} - Ay_n \tag{A.1.1}$$

where the constant A depends on the correlation of r_n and y_n. In effect A accounts for the frequency independent difference in the gains of F_R and F_I.

From equation A.1.1 it follows that:

$$E\left[y_{\grave{n}}^2\right] = E\left[(Ay_n + e_n)^2\right] = A^2 E\left[y_n^2\right] + E\left[e_n^2\right] \tag{A.1.2}$$

where E is the expectation operator, since by definition $E\left[y_n e_n\right] = 0$. Multiplying through equation A.1.1 by y_n and taking expected values:

$$E\left[y_n y_{\grave{n}}\right] = E\left[Ay_n^2 + y_n e_n\right] = A\,E\left[y_n^2\right] \tag{A.1.3}$$

from which it immediately follows that:

$$A = \frac{E\left[y_n y_{\grave{n}}\right]}{E\left[y_n^2\right]} \tag{A.1.4}$$

Finally, the value of $E\left[e_n^2\right]$ is obtained as:

$$E\left[e_n^2\right] = E\left[y_{\grave{n}}^2\right] - E\left[y_n^2\right]\left[\frac{E\left[y_n y_{\grave{n}}\right]}{E\left[y_n^2\right]}\right]^2 \tag{A.1.5}$$

which when rearranged and rewritten as variances becomes:

$$\sigma_e^2 = \frac{\sigma_{\grave{y}}^2 \sigma_y^2 - \sigma_{y\grave{y}}^2}{\sigma_y^2} \tag{A.1.6}$$

Since the signal power is given by σ_y^2 it follows that the signal to noise ratio at the output of filter F_I is given by:

$$SNR = 10 \log_{10} \left[\frac{\left[\sigma_y^2\right]^2}{\sigma_{\hat{y}}^2 \sigma_y^2 - \sigma_{\hat{y}y}^2} \right] \qquad (A.1.7)$$

When the signal to noise ratio is high the subtraction in the denominator of equation A.1.7 becomes numerically unstable; this is because $\sigma_{\hat{y}}^2 \sigma_y^2$ and $\sigma_{\hat{y}y}^2$ have almost exactly the same value and each is much larger than their difference. For SNR figures up to 40 dB and a signal power of $\sigma_y^2 = 1$ the denominator of A.1.7 is greater than 10^{-4}; this is two orders of magnitude larger than the precision limit of 32-bit floating point arithmetic. For SNR's above 40 dB double precision arithmetic is recommended.

This method of SNR measurement, as defined by equation A.1.6, is suitable for computing a running estimate each time a new sample of y_n and $\hat{y}_n$ arrives. This is particularly attractive since a filter simulation need only be run for sufficient samples to obtain the required degree of convergence. Furthermore, it is possible to prevent the saturation characteristics of F_I from giving a low estimate of SNR; this can be done by rejecting samples which give an error which is larger in absolute value than a half of the largest quantization interval. Saturation was not tested for as it was found to happen so infrequently (typically 1 in every 1000 experiments).

A.2 Complex Amplitude Measurement

In chapter 6 various filter design examples are presented. The behaviour of these filters under sinusoidal input is considered. Where experimental gain and phase measurements are shown they were obtained as follows.

Let x_n and y_n denote two discrete time sinusoidal signals corresponding respectively to the input and output of a linear system; it immediately follows that both sinusoids have the same frequency but may differ in phase and magnitude. In addition y_n is known to be corrupted by noise. It is required to estimate the complex amplitude of the linear system for a specific frequency.

Without loss of generality cosinusoid x_n will be assumed to have zero starting phase, that is:

$$x_n = A_x \left[\frac{e^{jnw} + e^{-jnw}}{2} \right] \qquad (A.2.1)$$

where n is the sample index number,

w is the angular frequency,

and A_x is the magnitude.

The output of the linear system, y_n, is phase shifted from the input by $\emptyset$ radians, is of magnitude A_y and is corrupted by a noise process u_n:

$$y_n = A_y \left[\frac{e^{j(nw + \emptyset)} + e^{-j(nw + \emptyset)}}{2} \right] + u_n \qquad (A.2.2)$$

The parameter to be estimated is:

$$C = \frac{A_y}{A_x} e^{j\emptyset} \qquad (A.2.3)$$

First, analytic signals x_n' and y_n' are obtained from the N samples of x_n and y_n (N > 1) using the following relations:

$$x_n' = e^{jw} x_n - x_{n-1} \qquad (A.2.4a)$$

$$y_n' = e^{jw} y_n - y_{n-1} \qquad \text{(A.2.4b)}$$

where $1 < n \leq N$

Substituting for x_n in equation A.2.4a using equation A.2.1 and for y_n in equation A.2.4b using equation A.2.2 the following expressions for x_n' and y_n' are obtained:

$$x_n' = jA_x \sin w \, e^{jnw} \qquad \text{(A.2.5a)}$$

$$y_n' = jA_y \sin w \, e^{j\phi} e^{jnw} + u_n' \qquad \text{(A.2.5b)}$$

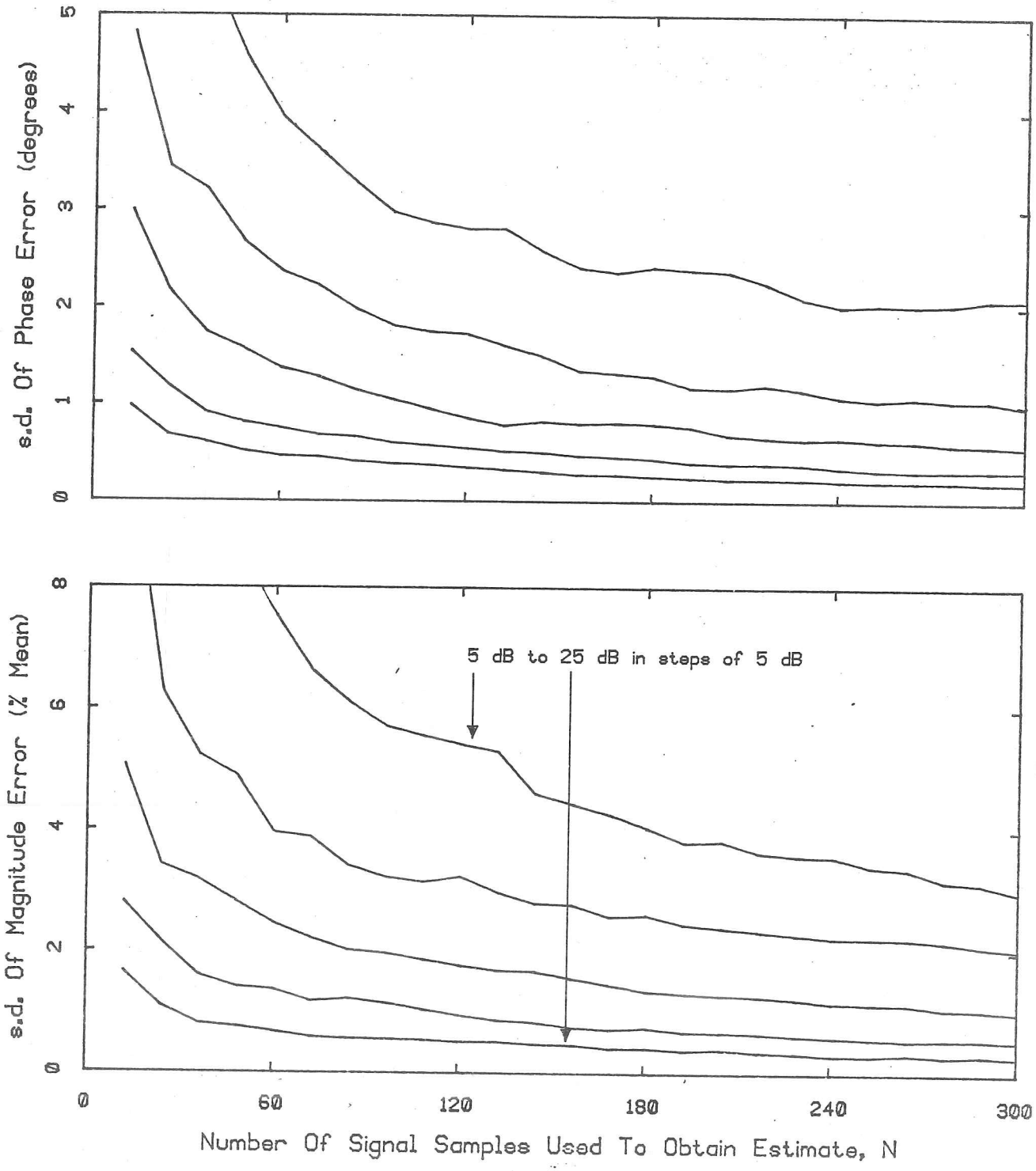
where u_n' is the analytic noise signal.

The complex amplitude C is obtained from the following expression:

$$C = \frac{\sum_{n=2}^{N} x_n' \, y_n'^*}{\sum_{n=2}^{N} x_n' \, x_n'^*} \qquad \text{(A.2.6)}$$

This estimate of C is unbiassed provided that noise process u_n is zero mean; it is reasonable to expect that this is true for a digital filter. The variance of the estimate of C depends on fourth order moments of u_n; for this reason no simple analytical expression can be found for it. Instead some experiments are included to give insight into the required sequence length N.

The first experiment is with a single sinusoid corrupted by gaussian white noise; various SNR levels are considered. The graph in figure A.2.1 shows that for an SNR of 20 dB (a typical value at the output of a filter) as few as 100 samples gives standard deviations in magnitude and phase estimate errors of 1.5% of mean and 0.75° respectively. If the estimate error is assumed to have gaussian statistics then with probability 0.99 the estimate will be in error by no more than 3.87% of mean (magnitude) and 1.94° (phase).

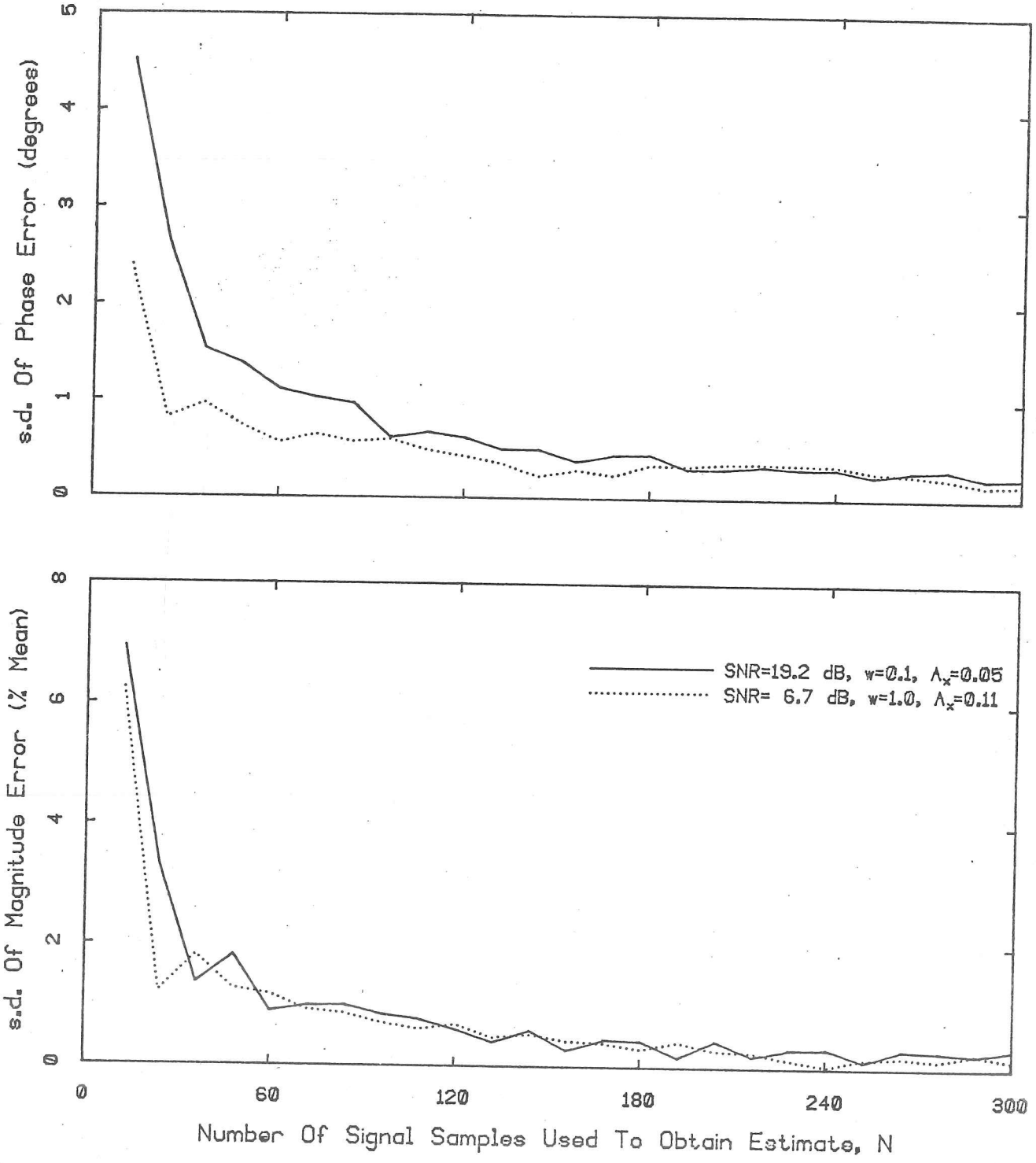


Performance of complex amplitude estimator as a function of the number of signal samples used and of the signal to noise ratio. White noise test.

Figure A.2.1

The second experiement is with a sinusoid applied at the input of a 1st order all-pole digital filter. The results are shown in the graph in figure A.2.2. This experiment illustrates that the estimator is still reliable in the presence of coloured noise.

The two experiements described are not intended to suggest that this method of complex amplitude estimation is robust to a broad class of noise conditions, more to suggest that it might be a reasonable method. Confidence in the use of this method rests on experience.



Performance of complex amplitude estimator as a function of the number of signal samples used and of the signal to noise ratio. Coloured noise test.

Figure A.2.2

A.3 Noise Process Generation

Where guassian white noise has been applied at the input of a filter this has been generated by summing groups of twelve consecutive uniform white random numbers; these, in turn, were generated using a linear shift register sequence. If the uniform random numbers lie in the interval $|-\frac{1}{2}, \frac{1}{2}|$ then the gaussian noise has unit variance. It follows that the maximum noise sample value is six standard deviations. Clearly, the tails of the gaussian noise process are poorly approximated, however, since the 255-level gaussian optimal quantizer saturates at three standard deviations this is not a problem.

A.4 General Purpose Computer Simulation of a Free Quantization Law Digital Filter

To illustrate the simulation of a free quantization law filter a specific example will be considered. The example is a first order all-pole optimised for a unit variance gaussian input process. From the filter difference equation:

$$y_n = Ky_{n-1} + x_n \tag{A.4.1}$$

where x_n and y_n are the input and output at time $t = nT$, and from the complex convolution theorem it follows that the output variance is given by:

$$\sigma_y^2 = \frac{1}{1-K^2} \tag{A.4.2}$$

Furthermore, from linear system theory y_n will be a gauss-markov process.

As x_n is a gaussian process, its associated optimal quantizer is the one shown in figure 2.2.2; the optimal quantizer for y_n has a similar shape but all its decision and quantization levels are larger in magnitude by a multiplying factor:

$$\frac{1}{\sqrt{1-K^2}} \tag{A.4.3}$$

As the shapes of both quantizers are the same only the decision and quantization levels for a unit variance quantizer need be stored; the levels for a quantizer of arbitrary variance are easy to find.

Following some data structure definitions, the quantization algorithm will be described.

Let AD and AQ be the names of two N element real valued arrays. Locations 2 through to N in array AD contain the N-1 values of the non-infinite decision levels; the first location of AD is unused. The N locations of array AQ contain the N values of the quantization levels. The indices of arrays AD and AQ are as defined in section 2.2; location i in AD and AQ contain the values of d_i and q_i respectively.

The first step in the quantization algorithm is to check if the value to be quantized lies in either of the intervals $|d_1, d_2| = |-\infty, d_2|$ or $|d_N, d_{N+1}| = |d_N, +\infty|$; denoting by S the value of the unquantized sample, this step becomes:

```
IF  S  ⩾  AD(N)  THEN
  BEGIN
    S: =  AQ(N)
    RETURN
  END
IF  S <  AD(2)  THEN
  BEGIN
    S: =  AQ(1)
    RETURN
  END
```

figure A.4.1

Assuming that S lies in the interval $|d_2, d_N|$ it is necessary to find the precise interval $|d_i, d_{i+1}|$ $(2 \leqslant i \leqslant N-1)$ and to set S to the value q_i. In principle a search of all the intervals from $|d_2, d_3|$ to $|d_{N-1}, d_N|$ could be performed; in practice this is too inefficient. Instead an algorithm based on the dichotomous search is used. With this method the interval $|d_2, d_N|$ is divided in two and a test for the interval containing S is performed. This sub-interval is divided again and the half containing S is noted. This process repeats until a single decision interval $|d_i, d_{i+1}|$ has been found. The algorithm is coded as follows:

```
    PL, PU and PT are pointer variables
    PL : =  2            ;  left extreme of interval
    PU : =  N            ;  right extreme of interval
L :  PT : = (PL + PU)/2   ;  centre of interval
```

```
IF  S ≥  AD(PT) THEN PL : = PT
IF  S <  AD(PT) THEN PU : = PT
UNLESS   PU = (PL + 1) GOTO L
S : =    AQ(PL)
```

figure A.4.2

For N quantization levels this algorithm takes $\log_2 N$ steps to complete; by comparison, the exhaustive search would require N/2 steps. A sample run of this algorithm is shown in figure A.4.3.

Arrays AD and AQ

i	1	2	3	4	5	6	7
AD(i)	n/u	-2.5	-1.5	-0.5	0.5	1.5	2.5
AQ(i)	-3	-2	-1	0	+1	+2	+3

PL	PU	PT	AD(PT)	S	S≥AD(PT)?	PU=PL+1?
2	7	4	-0.5	0.6	Yes	No
4	7	5	0.5	0.6	Yes	No
5	7	6	1.5	0.6	No	No
5	6	5	0.5	0.6	Yes	Yes
				1.0		

n/u not used

figure A.4.3

This algorithm is used frequently in a filter simulation and so it is coded as a sub-program. Calling this sub-program QUANT, the main program statement:

```
S : = QUANT (S)
```

(A.4.4)

would be read as 'the unquantized value S is passed to and quantized by sub-program QUANT, the result is stored in S'. If S, instead of being a sample of a unit variance process, is a sample of a process of variance V^2 then the appropriate call to QUANT is:

```
S : = V * QUANT (S/V)
```

(A.4.5)

Finally denoting by X, Y and Y1 the values of x_n, y_n and y_{n-1} in equation A.4.1, the algorithm for simulating a first order gaussian optimised all-pole is:

```
V : = 1.0/ (1.0 - K^2)^1/2

M :  X : =  GAUSS ( )
     X : =  QUANT (X)
     Y : =  K * Y1 + X
     Y : =  V * QUANT (Y/V)
    Y1 : =  Y

;    perform analysis on filter output
     GOTO  M
```

Sub-program GUASS ( ) produces samples of a zero mean, unit variance gaussian process.

Appendix B Cumulative Density Function of a Gaussian Process - Numerical Approximation

The formula used to obtain the value of the following integral:

$$\Phi(A) = \int_{-\infty}^{A} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx \qquad (B.1)$$

is that described by Abramowitz and Steagan (11). pp 932); it is as follows:

$$\Phi(A) = 1 - Z(A) \left[b_1 T + b_2 T^2 + b_3 T^3 + b_4 T^4 + b_5 T^5 \right] + E(A) \qquad (B.2)$$

where $Z(A) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}A^2}$

and $T = \frac{1}{1+pA}$

The coefficients b_i $(1 \leq i \leq 5)$ and p have the following values:

$b_1 = 0.31938\ 1530$

$b_2 = -0.35656\ 3782$

$b_3 = 1.78147\ 7937$

$b_4 = -1.82125\ 5978$

$b_5 = 1.33027\ 4429$

$p = 0.23164\ 1900$

figure B.1

The approximation error is bounded by:

$$|E(A)| \leq 7.5 \times 10^{-8}$$

Appendix C Implementation of Free Quantization Law Digital Filters - A Special Case

Whilst the use of free quantization law techniques in the design of digital filters gives an improved signal to noise ratio performance, the associated hardware realisation is more complex than in the fixed point arithmetic case. In chapter 4 a general approach to the realization of free quantization law filters was presented. In this appendix a special case of this new filter is considered; with this special case a very simple hardware structure is possible.

The filter to be discussed employs a uniform quantization law at each of its variables; what distinguishes it from the fixed point arithmetic filter is that the step size at each variable need not be the same. If the different step sizes are constrained to be related to each other by multiplying factors which are powers of 2 then a simple hardware realisation results. To illustrate this consider the problem of adding two sampled signals which are quantized using step sizes of q and $2^p q$ (p-integer) respectively. If a_n and b_n are both represented using M-bits then to add the two together requires an M + p bit binary full adder. As shown in figure C.1, variable a_n is connected at the A-port to bits 0 through M-1 and variable b_n is connected at the B-port to bits p through M + p-1.

A first order all-pole implemented in this way will now be analysed. The signal flow graph which represents the filter is shown in figure C.2.

The variables e_n^1, e_n^2 and e_n^3 represent noise processes. Variable e_n^1 models the distortion produced by the input quantizer, variable e_n^2 models the distortion produced by rounding the product Ky_{n-1} and variable e_n^3 models the noise produced by dropping some of the least significant bits at the output of the full adder.

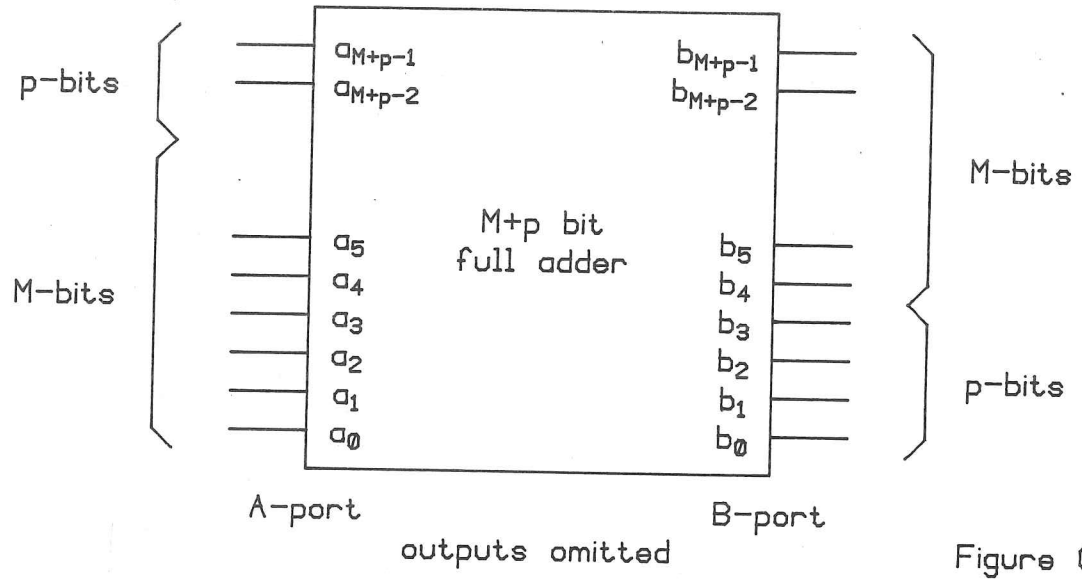


Figure C.1

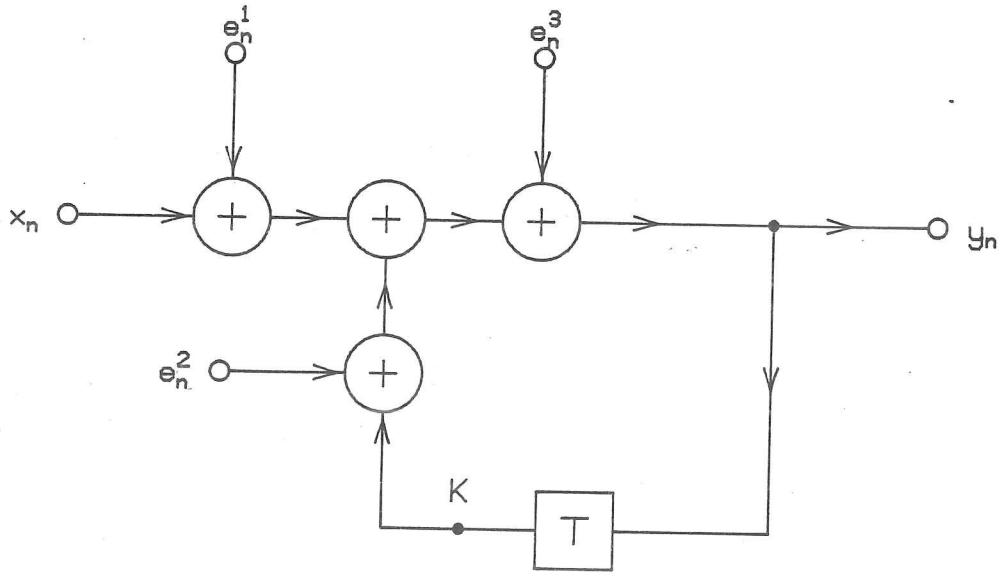


Figure C.2

Taking $\sigma_y^2 = 1$, where σ_y^2 is the variance of the output process y_n, as reference the noise source e_n^3 has a power of:

$$\sigma_3^2 = \frac{q^2}{12} \qquad (C.1)$$

where q is the optimum step size for a uniform quantizer applied to a gaussian process. The input variance has a value:

$$\sigma_x^2 = 1 - K^2 \qquad (C.2)$$

and hence the noise process e_n^1 has a power of:

$$\sigma_1^2 = (1 - K^2)\ \frac{q^2}{12} \qquad (C.3)$$

Because σ_y^2 is unconditionally larger than σ_x^2, Ky_{n-1} adds into adder A at p-bits higher significance than x_n. As Ky_{n-1} is of M + p bits precision the power of e_n^2 is:

$$\sigma_2^2 = (1 - K^2)\ \frac{q^2}{12} \qquad (C.4)$$

The total noise appearing at the output of the filter:

$$\sigma_{yn}^2 = \left[\sigma_1^2 + \sigma_2^2 + \sigma_3^2\right] \frac{1}{1 - K^2} = \frac{q^2}{12}\left[\frac{3 - 2K^2}{1 - K^2}\right] \qquad (C.5)$$

and hence the signal to noise ratio at the output is given by:

$$SNR = 10 \log_{10} \frac{12}{q^2}\left[\frac{1 - K^2}{3 - 2K^2}\right] \quad dB \qquad (C.6)$$

By a similar analysis it can be shown that for the equivalent filter with equal step sizes throughout the output signal to noise ratio is given by:

$$SNR = 10 \log_{10} \frac{12}{q^2}\left[\frac{1 - K^2}{2}\right] \quad dB \qquad (C.7)$$

Comparing equations C.6 and C.7 it is seen that the 'free quantization law' filter yieldsthe higher signal to noise ratio. The improvement in SNR is given by:

$$\Delta_{SNR} = 10 \log_{10} \left[\frac{2}{3 - 2K^2} \right] dB \qquad (C.8)$$

As K approaches unity Δ_{SNR} tends to 3dB.

By analysing the signal flow graph in figure C.3 it can be shown that for a second order all-pole this technique gives an SNR improvement of:

$$\Delta_{SNR} = 10 \log_{10} \left[\frac{3G}{G + 3} \right] dB \qquad (C.9)$$

The parameter G is the integrated power gain of the filter. For large values of G, Δ_{SNR} tends to 4.8dB.

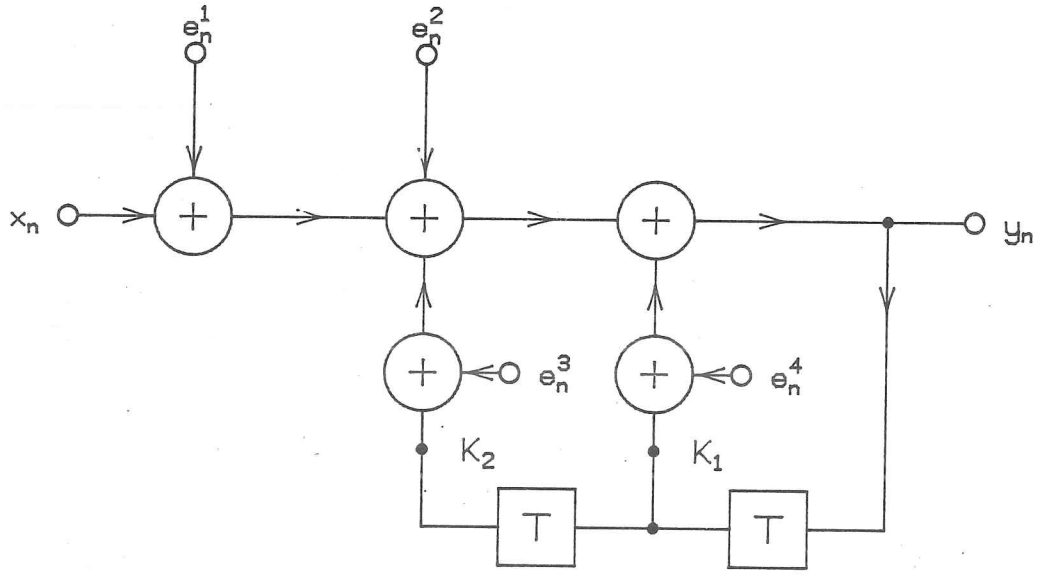


Figure C.3

References

BOOKS

General Digital Signal Processing

1. Rabiner L.R., Gold B., "Theory and Applications of Digital Signal Processing", Prentice-Hall, 1975.

2. Oppenheim A.V., Schafer R.W., "Digital Signal Processing". Prentice-Hall, 1975.

3. Gold B., Rader C., "Digital Processing of Signals", McGraw-Hill, 1969.

4. Oppenheim A.V. (editor), "Applications of Digital Signal Processing", Prentice Hall, 1978.

5. Schwarz M., Shaw L., "Signal Processing, Discrete Spectral Analysis, Detection and Estimation", McGraw-Hill, 1975.

6. Antoniou A., "Digital Filters: Analysis and Design", McGraw-Hill, 1979.

7. Papoulis A., "Probability Random Variables and Stochastic Processes", McGraw-Hill, 1965.

8. Meditch J.S., "Stochastic Optimal Linear Estimation and Control", McGraw-Hill, 1969.

9. Jury E.I., "Theory and Applications of the z-Transform Method", John Wiley & Sons, 1964.

Other Subjects

10. Stone H.S., "Discrete Mathematical Structures and Their Applications", Science Research Associates, 1975.

11. Abramowitz M., Stegun I.A., "Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables", Dover, 1965.

12. Phister M., "Logical Design of Digital Computers", John Wiley & Sons, 1959.

PAPERS

Coefficient Quantization

13. Knowles J.B., Edwards R., "Finite Wordlength Effects in a Multirate Direct Digital Control System", Proceedings IEE, Vol. 112, pp2376-2384, December 1965.

14. Rader C.M., Gold B., "Effects of Parameter Quantization on the Poles of a Digital Filter", Proceedings IEEE, Vol.55, pp688-689, May 1967.

15. Knowles J.B., Olcayto E.M., "Coefficient Accuracy and Digital Filter Response", IEEE Transactions on Circuit Theory, Vol. CT-15, pp31-41, March 1968.

16. Avenhaus E., Schuessler H.W., "On the Approximation Problem in the Design of Digital Filters with Limited Wordlength", Arch. Elek. Ubertragung, Vol. 24, pp571-572, 1970.

17. Crochiere R., "Digital Ladder Filter Structures and Coefficient Sensitivity", Research Laboratory of Electronics, M.I.T., Quarterly Progress Report, 103, October 1971.

18. Steiglitz K., "Designing Short-word Recursive Digital Filters", Proceedings of the 9th Allerton Conference on Circuit and System Theory, pp778-788, October 1971.

19. Liu B., "Effects of Finite Word Length on the Accuracy of Digital Filters - A Review". IEEE Transactions on Circuit Theory, Vol. CT-18, pp670-677, November 1971.

20. Oppenheim A.V., Weinstein C.J., "Effects of Finite Register Length on Digital Filtering and the Fast Fourier Transform", Proceedings IEEE, pp957-976, August 1972.

21. Avenhaus E., "On the Design of Digital Filters with Coefficients of Limited Word Length", IEEE Transactions on Audio and Electroacoustics, Vol. AU-20, pp206-212, August 1972.

22. Mitra S., Sherwood R., "Estimation of Pole-Zero Displacements of a Digital Filter due to Coefficient Quantization", IEEE Transactions on Circuits and Systems, Vol. CAS-21, pp116-124, January 1974.

23. Charamlambous C., Best M.J., "Optimisation of Recursive Digital Filters with Finite Word Lengths", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-22, pp424-431, December 1974.

24. Crochiere R.E., "A New Statistical Approach to the Coefficient Word Length Problem for Digital Filters", IEEE Transactions on Circuits and Systems, Vol. CAS-22, pp190-196, March 1975.

25. Bolton A.G., "Analysis of Digital Filters with Randomly Dithered Coefficients", IEE Proceedings Part G, Vol. 128, pp61-66, April 1981.

26. McLeod M., "Digital Signal Processing Systems with Discrete Parameters", Ph.D. Dissertation, University of Cambridge, 1979.

Signal Quantization- Roundoff Noise

27. Bennett W.R., "Spectra of Quantized Signals", Bell Systems Technical Journal, Vol. 27, pp446-472, July 1948.

28. Knowles J.B., Edwards R., "Effects of a Finite Word Length Computer in a Sampled-Data Feedback System", Proceedings IEE, Vol. 112, pp1197-1207, June 1965.

29. Jackson L.B., "On the Interaction of Roundoff Noise and Dynamic Range", Bell Systems Technical Journal, Vol. 49, pp159-184, 1970.

30. Jackson L.B., "Roundoff Noise Analysis for Fixed Point Digital Filters Realized in Cascade or Parallel Form", IEEE Transactions on Audio and Electroacoustics, Vol. AU-18, pp107-122, June 1970.

31. Hwang S.Y., "Roundoff Noise in State-Space Digital Filtering: A General Analysis", IEEE Transactions on Acoustics Speech and Signal Processing", Vol. ASSP-24, pp256-261, June 1976.

32. Stripad A.B., Snyder D.L., "A Necessary and Sufficient Condition for Quantization Errors to be Uniform and White", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-25, pp442-448, October 1977.

33. Butterwick H.J., "On the Quantization Noise Contribution in Digital Filters which are Uncorrelated with the Output Signal", IEEE Transactions on Circuits and Systems, Vol. CAS-26, pp901-910, November 1979.

34. Albinson L.J.A., Rayner P.J.W., "Finite Field Representation of Digital Filters", Proceedings of the International Conference on Digital Signal Processing, Florence, Italy, September 1981.

35. Sandberg I.W., "Floating Point Roundoff Accumulation in Digital Filter Realization", Bell Systems Technical Journal, Vol. 46, pp1775-1791, October 1967.

36. Weinstein C., Oppenheim A.V., "A Comparison of Roundoff Noise in Floating Point and Fixed Point Digital Filter Realizations", Proceedings IEEE, Vol. 57, pp1181-1183, June 1969.

37. Liu B., Kaneko T., "Error Analysis of Digital Filters Realized with Floating Point Arithmetic", Proceedings IEEE, Vol. 57, pp1735-1747, October 1969.

38. Oppenheim A.V., "Realization of Digital Filters using Block Floating Point Arithmetic", IEEE Transactions on Audio and Electroacoustics, Vol. AU-18, pp130-136, 1970.

39. Kan E.P.F., Aggarwal J.K., "Error Analysis of Digital Filters Employing Floating Point Arithmetic", IEEE Transactions on Circuit Theory, Vol. CT-18, pp678-686, November 1971.

40. Heath J.R., Nagle Jr. H.T., Stiva S.G., "Realization of Digital Filters using Input Scaled Floating Point", IEEE Transactions Vol. ASSP-27, pp469-477, October 1979.

Limit Cycles and Overflow Oscillations

41. Ebert P.M., Mazo J.E., Taylor M.C., "Overflow Oscillations in Digital Filters", Bell Systems Technical Journal, Vol. 48, pp2999-3020, 1969.

42. Jackson L.B., "An analysis of Limit Cycles due to Multiplication Rounding in Recursive Digital Filters", Proceedings of 4th Allerton Conference on Circuit and System Theory, pp69-78, 1969.

43. Parker A.T., Hess S.F., "Limit Cycle Oscillations in Digital Filters", IEEE Transactions on Circuit Theory, Vol. CT-8, pp687-697, November 1971.

44. Sandberg I.W., Kaiser J.F., "A Bound on Limit Cycles in Fixed Point Implementations of Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. AU-20, ppl10-112, June 1972.

45. Brubaker T.A., Gowdy J.N., "Limit Cycles in Digital Filters", IEE Transactions on Automatic Control, Vol. AC-17, pp675-677, October 1972.

46. Long J.L., Trick T.N., "A Note on Absolute Bounds on Limit Cycles due to Roundoff Errors in Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. AU-21, pp27-30, February 1973.

47. Kaneko T., "Limit Cycle Oscillations in Floating Point Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. AU-21, pp100-106, April 1973.

48. Classen T.A.C.M., Mecklenbrauker W.F.G., Peek J.B.H., "Some Remarks on the Classification of Limit Cycles in Digital Filters", Philips Research Report, Vol. 20, pp297-303, August 1973.

49. Classen T.A.C.M., Kristiansson L.O.G., "Necessary and Sufficient Conditions for the Absence of Overflow Phenomena in a Second Order Recursive Digital Filter", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. ASSP-23, pp509-515, December 1975.

50. Chang T.L., "A Note on Upper Bounds on Limit Cycles in Digital Filters", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. ASSP-24, pp99-100, February 1976.

51. Rahman M.H., Maria G.A., Fahmy M.M., "Bounds on Zero Input Cycles in All-Pole Digital Filters", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. ASSP-24, ppl89-192, April 1976.

52. Thong T., Liu B., "Limit Cycles in the Combinatorial Implementation of Digital Filters", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. ASSP-24, pp248-256, June 1976.

53. Classen T.A.C.M., Mecklenbrauker W.F.G., Peek J.B.H., "Effects of Quantization and Overflow in Recursive Digital Filters", IEEE Transactions on Acoustic Speech and Signal Processing. Vol. ASSP-24, pp517-529, December 1976.

54. Buttner M., "Elimination of Limit Cycles in Digital Filters with very low Increase in Quantization Noise", IEEE Transactions on Circuits and Systems, Vol. CAS-24, pp300-304, June 1977.

55. Lawrence V.B., Mina K.V., "Control of Limit Cycle Oscillations in Second Order Recursive Digital Filters using constrained Random Quantization", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. ASSP-26, ppl27-134, April 1978.

56. Mills W.L., Mullis C.T., Roberts R.A., "Digital Filter Realization without Overflow Oscillations", IEEE Transactions on Acoustics Speech and Signal Processing, Vol. ASSP-26, pp334-338, August 1978.

57. Jackson L.B., "Limit Cycles in State-Space Structures for Digital Filters", IEEE Transactions on Circuits and Systems, Vol. CAS-26, pp67-68, January 1979.

58. Barnes C.W., "Roundoff Noise and Overflow in Normal Digital Filters", IEEE Transactions on Circuits and Systems, Vol. CAS-26, pp154-159, March 1979.

Optimum Quantization

59. Max J., "Quantizing for Minimum Distortion", IRE Transactions on Information Theory, Vol. IT-6, pp7-12, March 1960.

60. Fleischer P.E.,"Sufficient Conditions for achieving Minimum Distortion in a Quantizer", IEEE Internation Convention Record, pp104-111, 1964.

61. Bluestein L.I., Asymptotically Optimal Quantizers and Optimum Analgue to Digital Conversion for Continuous Signals", IEEE Transactions on Information Theory, Vol. IT-10, July 1964, pp242-246.

62. Roe G.M., "Quantizing for Minimum Distortion", IEEE Transactions on Information Theory, Vol. IT-10, pp384-385, October 1964.

63. Goblick T.J., "Analogue Source Digitisation: A Comparison of Theory and Practice", IEEE Transactions on Information Theory, Vol. IT-13, pp323-326, April 1967.

64. Wood R.C., "On Optimum Quantization", IEEE Transactions on Infomation Theory, Vol. IT-15, pp248-252, March 1969.

65. Elias P.,"Bounds on Performance of Optimum Quantizers", IEEE Transactions on Information Theory, Vol. IT-16, pp172-184, March 1970.

66. Sharman D.K., "Design of Absolutely Optimal Quantizers for a Wide Class of Distortion Measures", IEEE Transactions on Information Theory, Vol. IT-24, pp693-702, November 1978.

67. Mauersberger W., "Experimental Results on the Performance of Mismatched Quantizer", IEEE Transactions on Information Theory, Vol. IT-25, pp381-386, July 1979.

Number of Code Assignments

68. McCluskey E.J., Unger S.J., "A Note on the Number of Internal Variable Assignments for Sequential Switching Circuits", IRE Transaction on Electronic Computers, Vol. EC-8, pp489-490, December 1959.

69. Bianchini R., "On Internal Variable Assignments for Sequential Switching Circuits", IRE Transactions on Electronic Computers, Vol. EC-100, pp95-96, March 1961.

70. Weiner P., Smith E.J., "On the Number of Distinct State Assignments for Synchronous Sequential Machines", IEEE Transactions on Electronic Computers, Vol. EC-16, pp220-221, April 1967.

71. Harrison M.A., "On Equivalence of State Assignments", IEEE Transactions on Computers, Vol. C-17, pp55-57, January 1968.

72. Parchmann R., "The Number of State Assignments for Sequential Machines, IEEE Transactions on Computers, C-21, pp613-614, June 1972.

Code Assignment

73. Hartmanis J., "On the State Assignment Problem for Sequential Machines 1", IRE Transactions on Electronic Computers, Vol. EC-10, pp157-165, June 1961.

74. Stearns R.E., Hartmanis J., "On the State Assignment Problem for Sequential Machines II", IRE Transactions on Electronic Computers, Vol. EC-10, pp593-603, December 1961.

75. Armstrong D.B., "A Programmed Algorithm for Assigning Internal Codes to Sequential Machines", IRE Transactions on Electronic Computers, Vol. EC-11, pp466-472, August 1962.

76. Armstrong D.B., "On the Efficient Assignment of Internal Codes to Sequential Machines", IRE Transactions on Electronic Computers, Vol. EC-11, pp611-622, October 1962.

77. Nichols A.J., "Comments on Armstrong's State Assignment Technique", IRE Transactions on Electronic Computers, Vol. EC-12, pp407-408, August 1963, Author's reply pp408-409.

78. Kohavi Z., "Secondary State Assignment for Sequential Machines", IEEE Transactions on Electronic Computers, Vol. EC-13, pp193-203, June 1964.

79. Dolotta T.A., McCluskey E.J., "The Coding of Internal States of Sequential Circuits", IEEE Transactions on Electronic Computers, Vol. EC-13, pp549-562, October 1964.

80. Zahel T.U., "On Coding the States of Sequential Machines with the Use of Partition Pairs", IEEE Transactions on Electronic Computers, Vol. EC-15, pp249-253, April 1966.

81. Davis W.A., "An Approach to the Assignment of Input Codes", IEEE Transactions on Electronic Computers, Vol. EC-16, pp435-442, August 1967.

82. Torng H.C., "An Algorithm for finding Secondary Assignments of Synchronous Sequential Machines", IEEE Transactions on Computers, Vol. C-17, pp461-469, May 1968.

83. Karp R.M., "Some Techniques of State Assignment for Synchronous Sequential Machines", IEEE Transactions on Electronic Computers, Vol. EC-13, pp507-518, October 1964.

84. Johnson D.L., O'Keefe K.H., "The Application of Shift Registers to Secondary State Assignment: Part 1", IEEE Transactions on Computers, Vol. C-17, pp954-965, October 1968.

85. Johnson D.L., O'Keefe K.H., "The Application of Shift Registers to Secondary State Assignment: Part 11", IEEE Transactions on Computers, Vol. C-17, pp966-977, October 1968.

86. Story J.R., Harrison H.J., Reinhard E.A., "Optimum State Assignment for Synchronous Sequential Circuits", IEEE Transactions on Computers, Vol. C-21, ppl365-1373, December 1972.

87. Noe P.S., Rhyne V.T., "A Modification to the SHR Optimal State Assignment Procedure", IEEE Transactions on Computers, Vol. C-23, pp327-329, March 1974.

88. Lala P.K., "An Algorithm for the State Assignment of Synchronous Sequential Machines", Electronics Letters, Vol. 14, No. 6, March 1978.

89. Edwards C.R., Eris E., "State Assignment and Entropy", Electronics Letters, Vol. 14, No. 13, June 1978.

Other Subjects

90. Albinson L.J., Rayner P.J.W., "Reed-Muller Transform Processor", Internal Report (to be published), Engineering Department, University of Cambridge.

91. Liu B., Peled A., "Heuristic Optimization of Cascade Realization of Fixed-Point Digital Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-23, pp464-473, October 1975.

92. Peled A., Liu B., "Combinatorial Realization of Digital Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-22, pp456-462, December 1974.

93. Steiglitz K., Liu B., "An Improved Algorithm for Ordering Poles and Zeroes of Fixed-Point Recursive Digital Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-24, pp341-343, August 1976.

94. Steiglitz K., "Computer-Aided Design of Recursive Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. AU-18, ppl23-129, June 1970.

95. Thajchayapong P., "The Optimal Design of Recursive Digital Filters", Ph.D. Dissertation, University of Cambridge, 1974.

96. Wells H.J., "Digital Filtering Facility for Computer Processing", Proceedings IEE, Vol. 125, No. 10, October 1978.

97. Rayner P.J.W., "The Application of Finite Arithmetic Structures to the Design of Digital Processing Systems", International Conference on Digital Signal Processing, Florence, Italy, 1978.

98. Green D.H., "Modular Representation of Multiple Valued Logic Systems", IEE Proceedings, Vol. 121, No. 6, June 1974.

99. Kellerman E., "A Formula for Logical Network Cost", IEEE Transactions on Computers, Vol. C-17, No. 9, September 1968.

100. Hellerman L., "A Measure of Computational Work", IEEE Transactions on Computers, Vol. C-21, No. 5, May 1972.

101. Mukhopadhyay A., Schmitz G., "Minimisation of Exclusive-Or and Logical Equivalence Switching Circuits", IEEE Transactions on Computers, Vol. C-19, No. 2, February 1970.

GLOSSARY OF TERMS

N_{C_i}	the number of ways in which it is possible to choose i items from N
d_i	ith decision level (of a quantizer)
D	distortion produced by a quantizer
e	instantaneous error produced by a quantizer
E [ ]	statistical expectation operator
G_y	integrated power gain from the input of a filter to variable y
G_{ij}	integrated power gain from the input of the ith section to the ouput of the jth section of an N section cascade filter. $1 \leqslant i \leqslant j \leqslant N$
K	coefficient in recursive part of a filter
L	coefficient in non-recursive part of a filter
N	(i) total number of quantization levels (ii) length of a sequence of signal samples
q	step size of a uniform law quantizer
q_i	ith quantization level
Q_x	optimum (minimum mean square error) quantizer for process x
SNR_I	signal to noise (power) ratio at the input of a filter
SNR_O	signal to noise (power) ratio at the output of a filter
T	one signal sample (T second) delay
y_n	nth sample of process y
σ_y^2	power of process y
$\sigma_{y_n}^2$	noise power at variable y_n
$\sigma_{y_s}^2$	signal power at variable y_n
Δ	half optimum step size for a uniform quantizer applied to a gaussian process
$\oplus$	EXCLUSIVE-OR logical operation
$\boxtimes$	Kronecker matrix product operation