NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

SUPPRESSION OF LIMIT CYCLES IN DIGITAL FILTERS

by

Shedman TAM, B.A.Sc.

A thesis submitted to the

School of Graduate Studies and Research

in partial fulfillment of the requirements
for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering

Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

August, 1987

ISBN 0-315-46863-7

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ACKNOWLEDGEMENT

The author wishes to express her sincere gratitude to her thesis supervisor Dr. Willem Steenaart for his patient guidance and moral support throughout the course of this work.

Special thanks goes to Dr. Daniel Dubois for his invaluable advice and assistance.

ABSTRACT

Signal quantization is necessitated by the implementation of digital filters as finite state machines. In a digital recursive filter, among the damaging effects of signal quantization is the potential presence of limit cycles when the filter input has become zero, and a contribution to the overall quantization noise output during normal filter operation. Controlled quantization is an effective technique for the suppression of limit cycles.

In this thesis a new controlled quantization scheme for a second order recursive filter, shown to be effective in suppressing zero input limit cycles, is proposed. The derivation of this method is based on the Lyapunov stability theory. Specifically, a Lyapunov function is derived as a limit cycle free criterion. A proof that this criterion can be satisfied by the appropriate signal quantization choice is provided. An inplementation scheme using the Stored Product Digital Filter (SPDF) is described and specific computer simulation results showing the effectiveness of this method are given. The quantization noise generated by this quantization method is simulated and compared with that generated by the existing controlled quantization method (Lawrence and Mitra). The comparison shows that the new scheme generates less

quantization noise when the filter is driven by a White Gaussian input. This superior noise performance is true for the entire filter coefficient plane which guarantees stability.

Abstract

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 DIGITAL FILTER IMPLEMENTATION AND SIGNAL QUANTIZATION

A digital filter is a computational process or an algorithm whereby a sequence of binary numbers, the filter input, is transformed to a second sequence of binary numbers, the filter output. It can be described by the following difference equation:

$$y(n) = \sum_{k=0}^{N} c(k)x(n-k) - \sum_{k=1}^{N} b(k)y(n-k) \qquad (1.1)$$

where x() and y() are the input and the output sequence respectively; c() and b() are the filter coefficients. The second summation in the equation represents the feedback part of the filter.

This computational process can be realized by either hardware using shift registers, binary adders and multipliers; or by software on a general or special purpose computer. With either form of realization there is the limitation of representing numbers using finite wordlength. One immediate

consequence is that signals in the filter must be quantized to conform to the number of bits available for their representation. In itself, this finite wordlength only causes quantization error. However, arithmetical operations such as multiplication and addition, generally lead to an increased number of bits required to store the resultant product signals. The requantization of these resultant signals will introduce nonlinearities into the otherwise linear and stable filter [1].

Two methods of wordlength reduction are in common use, both involve the treatment of a digitized signal's least significant bits. In **Roundoff Quantization**, a signal is substituted by the nearest possible word that can be represented by the available number of bits; whereas in **Truncation Quantization** the excessive least significant bits are simply discarded. In the case where signals are represented in the sign-magnitude and fixed-point format, the truncation approach leads to **Magnitude Truncation**. This form of quantization introduces larger mean square noise error than Roundoff, however, it has the advantage of simpler realization. Besides noise performance, the different methods of signal quantization will affect the filter's nonlinear behaviour differently; in particular, the limit cycle behaviour in the filter. A more detailed explanation of this will be given in Chapter 2.

1.2 LIMIT CYCLE OSCILLATION

Limit Cycles can be defined as periodic oscillations or a constant level signal which may be observed in the output of a digital filter when the input has become zero.

The wordlength limitation in a digital filter may give rise to two types of limit cycles. The first type is caused by the overflow nonlinearities resulting from the modification of those signals which have a value greater than the maximum level representable by the internal wordlength. Limit cycles of this origin are usually very large in amplitude, but in general, they can be completely suppressed by using an appropriate saturation arithmetic [2].

The second and by far more common type of limit cycles is caused by quantization nonlinearities. As mentioned earlier, the resultant signals from multiplication and addition often have longer wordlength than the original signal. Where there is a closed loop, as is the case in a recursive filter, some form of wordlength reduction must be performed to the output of these arithmetic elements. The propagation of the quantization nonlinearity in the feedback path in the filter may cause it to oscillate after the input has become zero. In the case of a non-recursive filter, i.e. the feedback part in (1.1) is zero, the quantization effect is observed as an additive noise in the

output and no limit cycle will occur.

If the filter is designed to be used as a digital oscillator, the occurrence of limit cycles may be exploited to its advantage [4]. However, if stable linear filtering operation is desired, the effect of limit cycles can be very damaging. For example, it has been observed that limit cycles are very disturbing in speech processing because the human ears are very sensitive to harmonic signals [5]. Thus from the perception point of view, the presence of limit cycles during pauses in the speech, that is, input to the filter has become zero, is more harmful than the presence of a similar level of quantization noise in the speech.

Studies have also shown that limit cycles of different frequencies and amplitudes may occur in the same filter depending only on the state of the filter when the input becomes zero [6]. Some limit cycles have a higher probability of occurrence and are more likely to be observed than others due to the existence of 'branch points' in these cycles [7]. They are termed the accessible limit cycles because they can be reached from any random initial conditions. Moreover, most limit cycles have frequencies in the passband of the filter and will propagate through the system [8]. This consideration precludes the use of selective filtering as a means to eliminate limit cycles.

Methods of limit cycle suppression have been the focus of

recent research efforts in filter nonlinearities. These include the derivation of limit cycle free filter structures and new approaches in signal quantization which will eliminate or diminish the occurrence of limit cycles. It should be noted that limit cycles can be made arbitrarily small if the number of bits used for signal representation is large enough. However, this approach will lead to increased filter complexity and hence increased cost. Another important consideration is the trade-off between limit cycles and quantization noise.

1.3 PROBLEM STATEMENT

The criteria for designing linearly stable digital filters are well established. However, linear stability is guaranteed only for idealized operating conditions. In the actual implementation of the filter as a finite state system, certain types of nonlinearity often arise due to the wordlength limitation. One consequence is that the otherwise stable filter may exhibit an unstable behaviour often termed limit cycle oscillation. A technique to suppress the limit cycles caused by quantization nonlinearity is the subject matter of this thesis.

1.4 THESIS OUTLINE

Chapter 1 INTRODUCTION.

Chapter 2 LIMIT CYCLE IN DIGITAL FILTERS: Relevant results in limit cycle research are given. These include the causes of limit cycles, amplitude bounds, and factors which influence the limit cycle behaviour. As well, the available methods of limit cycle suppression are outlined and evaluated.

Chapter 3 CRITERION FOR A LIMIT CYCLE FREE FILTER: An introduction to the Lyapunov stability analysis is given. It is followed by the development of the limit cycle suppression criterion of this thesis which utilizes the Lyapunov stability concept.

Chapter 4 APPLICATION AND IMPLEMENTATION OF THE LIMIT CYCLE FREE CRITERION: The applicability of the limit cycle free criterion in the quantization environment of a second order digital filter is proven. An implementation scheme using the Stored Product Digital Filter is described. Computer simulation results are presented.

Chapter 5 QUANTIZATION NOISE EVALUATION: The noise

performance of the proposed quantization scheme is investigated and compared with the existing controlled quantization method.

Chapter 6 CONCLUSION.

The original contribution of this thesis is contained in Chapters 3, 4 and 5.

CHAPTER 2 LIMIT CYCLE IN DIGITAL FILTERS

2.1 GENERAL

A recursive digital filter can be realized by various types of configurations utilizing one or more feedback loops. For linear stability in a second order digital filter, it is required that the poles of the filter be within the unit circle on the Z-plane. The filter coefficients must be computed with due consideration to this stability constraint. However, even those coefficients which initially fall within the stability region can lead to limit cycle oscillations in the output due to the effects of finite arithmetic of the digital implementation.

The effects of quantization nonlinearities are determined by the following factors:

1. the type of number representation chosen - e.g. floating point or fixed point with ones complement or twos complement representation for negative numbers.

2. the type of quantization - e.g. roundoff or truncation.

3. the filter structure - e.g. the direct form, cascade parallel, wave digital, or state space.

4. the wordlength used for the quantized signal.

5. the location of the quantizers in the filter.

When describing limit cycle behaviour it is essential that the filter conditions be specified in order to avoid misunderstanding. In this thesis, unless otherwise stated, it is assumed that signals are represented by fixed-point sign-magnitude numbers, thus the fixed-point sign-magnitude arithmetic is used. Also, in order to simplify notation, it is assumed that the filter input signal has been quantized and scaled as integers. This permits the description of the filter output, which is dependent on the filter coefficients, to be also represented in integer form. Product quantization then involves the treatment of the fractional part of the resultant signal from a multiplication operation. This filter output can be converted into the actual output by the application of an appropriate scale factor.

The most commonly employed quantization methods are roundoff (RO) and magnitude truncation (MT). The transfer function for these two types of quantizer are shown in Figures 2.1 and 2.2.

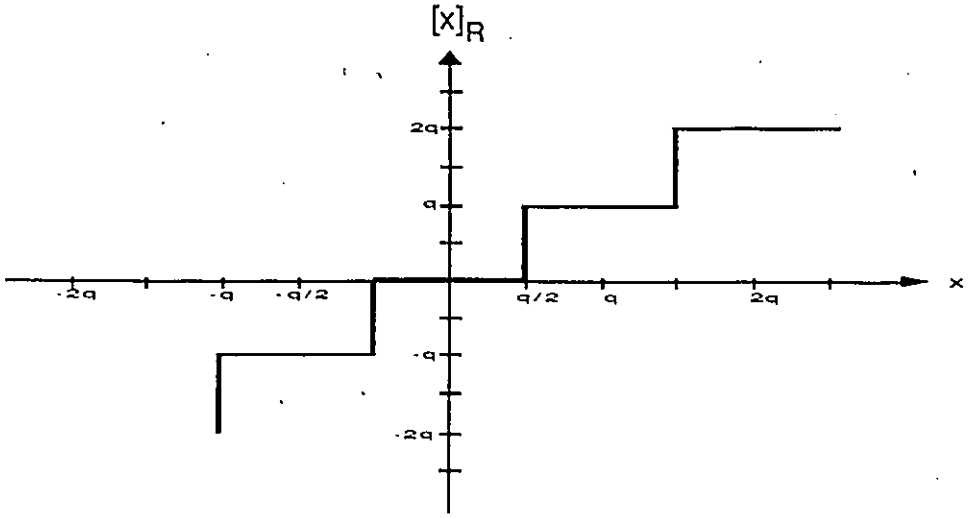


Figure 2.1 Transfer Characteristic of the RO quantizer: with quantization step size q

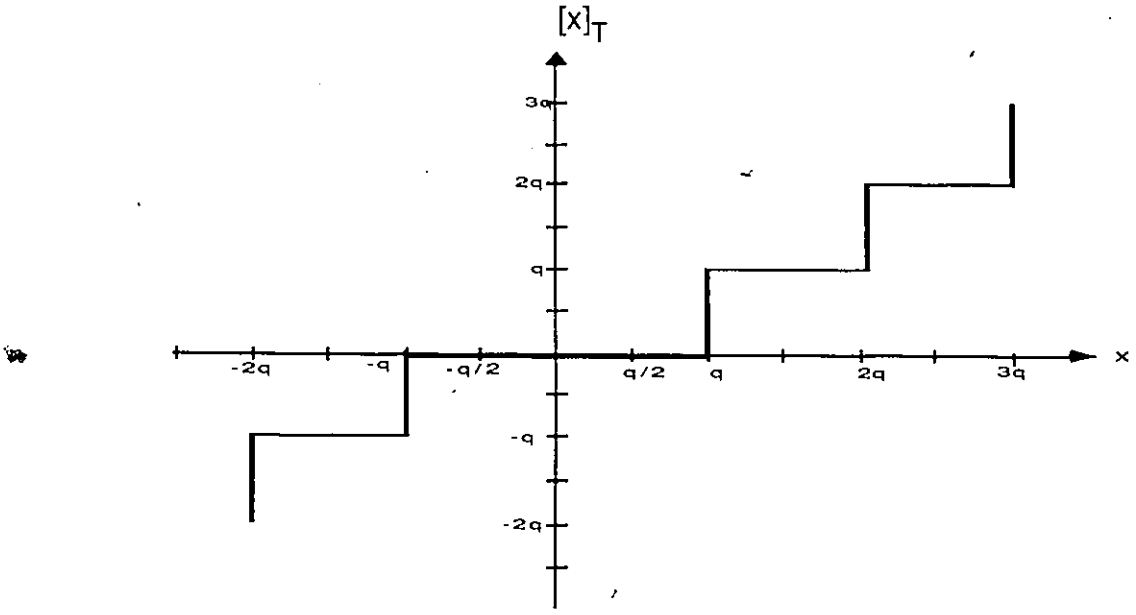


Figure 2.2 Transfer Characteristic of the MT quantizer: with quantization step size q

2.2 LIMIT CYCLE IN FIRST ORDER DIGITAL FILTER

Under zero input condition the output of a first order digital filter is given by:

$$y(n) = -[by(n-1)]_Q \qquad (2.1)$$

where b is the filter coefficient and $[\ ]_Q$ indicates possible product quantization. In order to satisfy the linear stability criterion, the coefficient |b| must be less than 1. This ensures that the magnitude of the signal in the feedback path is reduced each time it circulates through the loop and eventually becomes arbitrarily small. Figure 2.3 illustrates the filter structure and its impulse responses for a given coefficient b.

It can be seen that for |b| = 1 the filter is marginally stable. In the zero input case, such a system will have a d.c. output if b is negative, and a constant magnitude with alternating sign output if b is positive.

In a digital system the necessity to quantize the product after multiplication may cause |b| to have an effective value of 1. If this happens then limit cycles will be observed in the filter output. This means that instead of approaching the zero state asymptotically when the input becomes zero, the system will

oscillate about the origin. The amplitude interval within which these oscillations are confined is known as the deadband [9]. For a first order filter the frequency of oscillation is either d.c. or $f_s/2$, where $f_s = 1/T$ is the system sampling frequency. This type of oscillation is called the first order limit cycle [10].

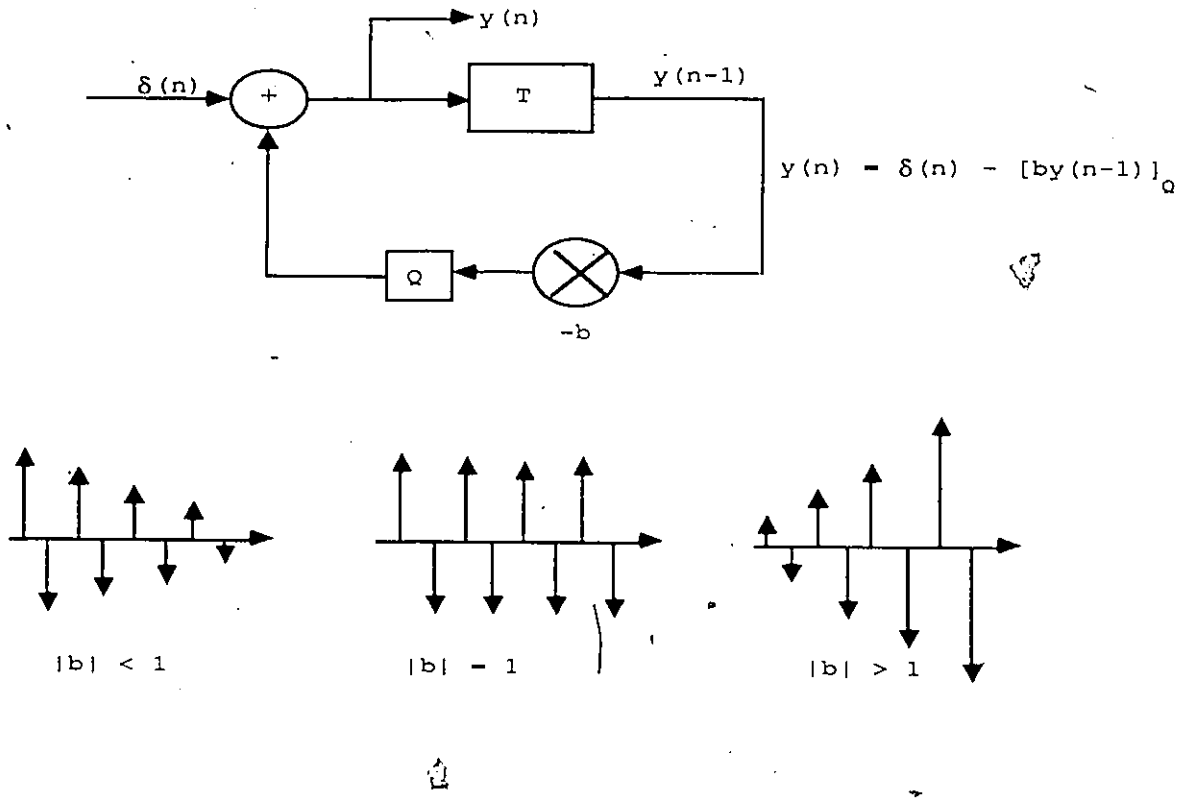


Figure 2.3 First Order Digital Filter: structure and impulse responses

It has been shown that if roundoff quantization is used then limit cycles always occur for $|b| \geq 0.5$ [9], [10]. The bound for the first order deadband is $D = [-k, k]$, where k is the smallest integer satisfying

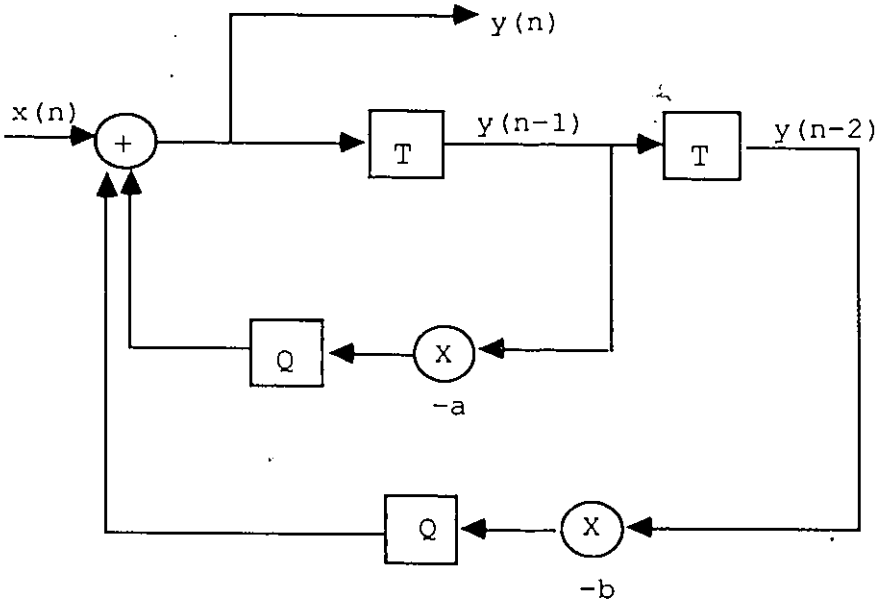
$$k \geq 0.5 \; / \; 1-|b| \qquad (2.2)$$

2.3 LIMIT CYCLE IN SECOND ORDER DIGITAL FILTER

A large portion of the research work on the limit cycle problem has been concentrated on the second order digital filter. This is so because higher order filters can be constructed using the lower order filters as the basic building blocks. For example, Kaiser has shown that a cascade of first- and second-order subfilters is preferable over any direct realization of a higher order digital filter [11]. For the purpose of studying the limit cycle behaviour in a filter's natural response (i.e. zero input, initial conditions only), a filter section with two poles and no zero is desirable in order to avoid the distracting influence of the zeros on the response [6]. With this in mind, the equation describing the second order filter section with zero input is given by:

$$y(n) = - [ay(n-1)]_Q - [by(n-2)]_Q \qquad (2.3)$$

where a and b are the filter coefficients and $[\ ]_Q$ indicates possible product quantization. This second order filter section is shown in Figure 2.4. The coefficient pairs (a,b) corresponding to every stable pole pair position can be conveniently represented in the a-b coefficient plane as the area within the triangle shown in Figure 2.5. When transformed to the Z-plane this triangular area maps onto the stability region of $|z| < 1$. Figure 2.6 shows the impulse response of the filter.



$$y(n) = x(n) - [ay(n-1)]_Q - [by(n-2)]_Q$$

Figure 2.4 Second Order Filter Section

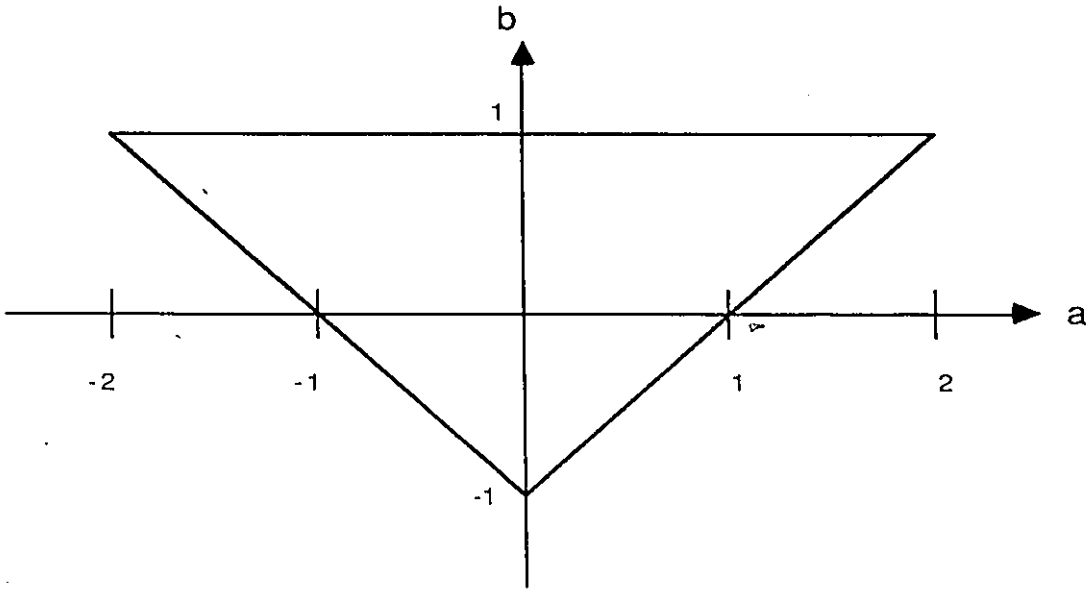


Figure 2.5 Coefficient Plane for Second Order Digital Filter

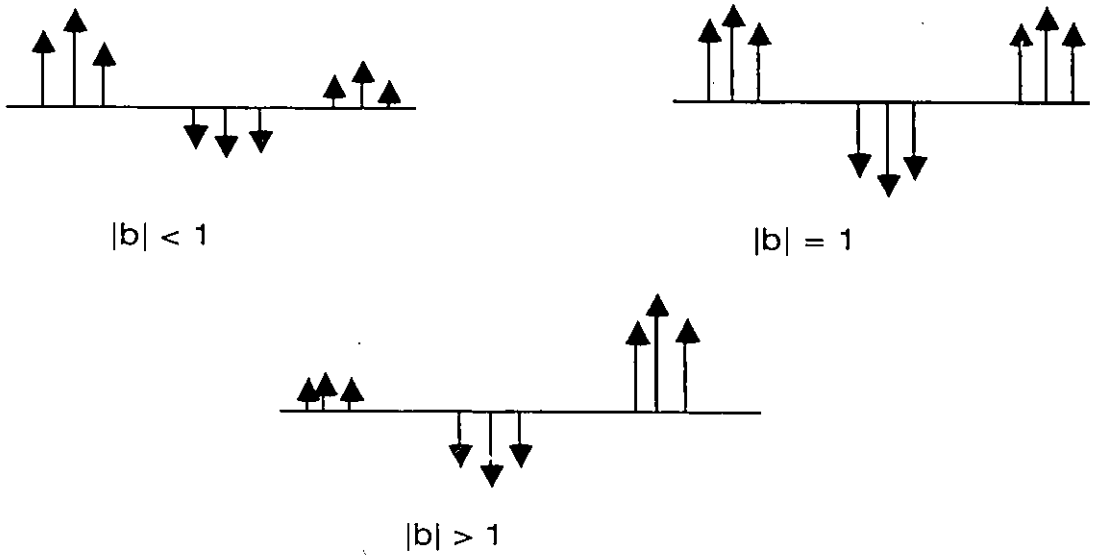


Figure 2.6 Second Order Filter Impluse Response

Linear stability is guaranteed for the coefficient pairs (a,b) within the triangular region bounded by the equations $-b \pm a = 1$ and $b = 1$. However, product quantization may cause the coefficients to move outside of this stability region. Jackson's Effective Value model [10] postulates that limit cycles arise as a result of the effect of quantization which has led to either an effective complex pole pair on the unit circle, or an effective real pole pair located on the real axis, on the Z-plane. He further observes that an effective real pole pair always causes the first order limit cycles, similar to those found in a first order digital filter; whereas an effective complex pole pair may give rise to either first or second order limit cycles which are oscillations with periods greater than two. Jackson has also partitioned the area inside the stability triangle into regions corresponding to the limit cycle amplitudes. This is shown in Figure 2.7. The amplitude bound for the second order deadband is the same as (2.2). However, exceptions to this bound have been found [6].

Several authors have shown that the condition for the existence of limit cycles in a second order digital filter using roundoff quantization is [6], [7], [10]:

$$|b| \geq 0.5 \qquad (2.4)$$

Claasen et al [7] have shown that if the filter coefficient

b satisfies (2.4) then a limit cycle will result for every non-zero initial condition because the zero state cannot be reached from any other state in the filter.

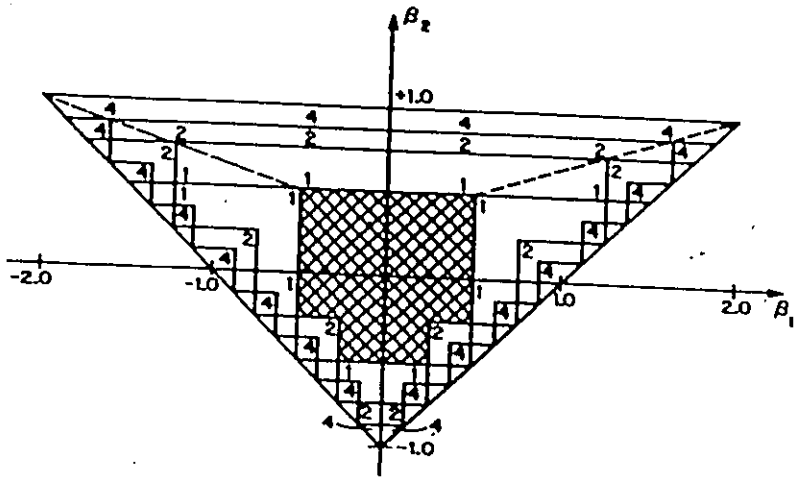


Figure 2.7 Limit Cycle Amplitude due to Roundoff: Jackson

2.4 LIMIT CYCLE BOUNDS

Given that limit cycles exist, bounds on the limit cycle amplitude can be found as a function of the filter coefficients. Three different types of amplitude bounds for roundoff quantization have been given.

- Absolute bounds [6], [14] : these are bounds on the maximum value of the quantization error and tend to be rather pessimistic.

- RMS bounds [15] : this is a bound on the rms value of the quantization error and does not give any information on the maximum limit cycle amplitude.

- Approximate bound [10] : this is the bound derived by Jackson based on the effective value model. Limit cycles exceeding this bound have been found [6], [7].

2.5 EXISTING METHODS OF LIMIT CYCLE SUPPRESSION

A number of quantization methods and filter structures have been proposed as means of eliminating or reducing the occurrence of limit cycles. Some of these are considered in the remainder of this chapter.

1. Longer wordlengths can be used for the internal filter signal than for the output signal, the excessive least significants bits can then be discarded (truncated) just prior to the output. In order to use this method

effectively, the number of additional bits must be long enough to contain the largest possible limit cycles. The limit cycle amplitude bounds can be used to determine the number of additional bits required. However, the approximate and the rms bounds do not always account for the maximum limit cycles and the absolute bounds tend to be rather pessimistic and may lead to a considerable number of extra bits.

2. It has been shown that magnitude truncation quantization is effective in reducing the occurrence of limit cycles in digital filters [12], [13]. Kao has shown that a first order digital filter using magnitude truncation quantization will have no zero input limit cycles [12]. In the case of the second order filter with two MT quantizers placed as shown in Figure 2.4 it was found that only limit cycles of period 1 and 2 will occur for two triangular sub-regions in the stable coefficient plane. The remaining cycles are bounded by:

$$|y(n)| < 1 \;/\; (1-|a|+b) \qquad (2.5)$$

These results are shown in Figure 2.8.

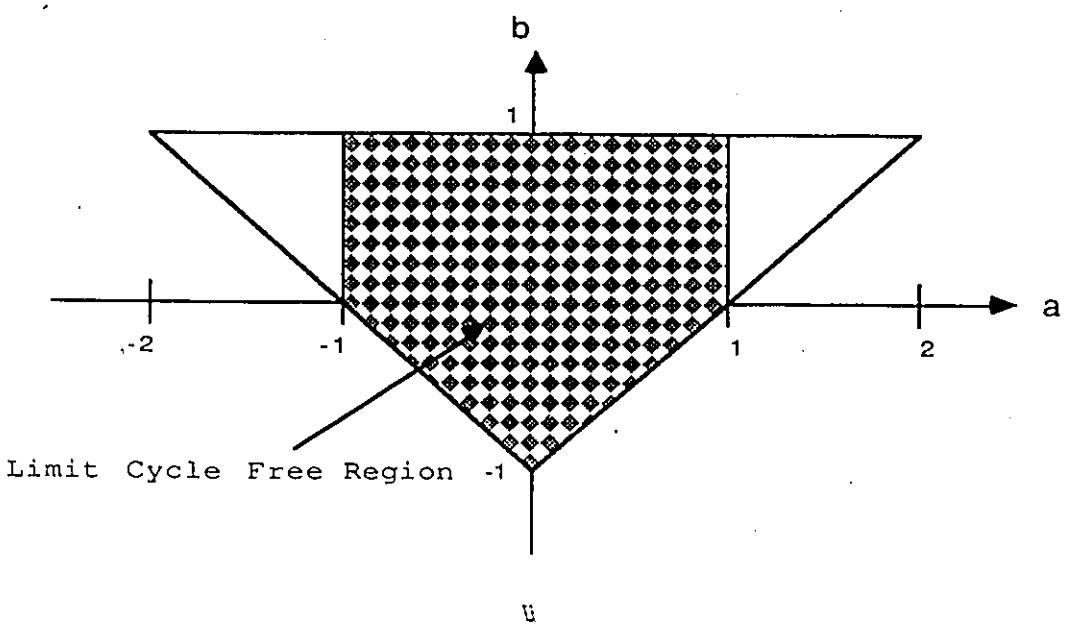


Figure 2.8 Limit Cycle due to 2 MT Quantizers: Kao

It is also possible to implement the second order digital filter with one quantizer. In this case the results of the two multipliers are added with full precision and then quantized. Claasen et al have shown that if one MT quantizer is used in this manner then limit cycles are only found in the two trapezoid areas in the stable coefficient plane [13]. This is shown in Figure 2.9. It should be noted that longer internal wordlength is required in this case than when two quantizers are used. The disadvantage of magnitude truncation is that it has a 6 db lower signal-to-noise ratio than the roundoff case. The amplitude of the remaining limit cycles can get very large as the poles of the transfer function get close to the unit circle on the Z-plane.

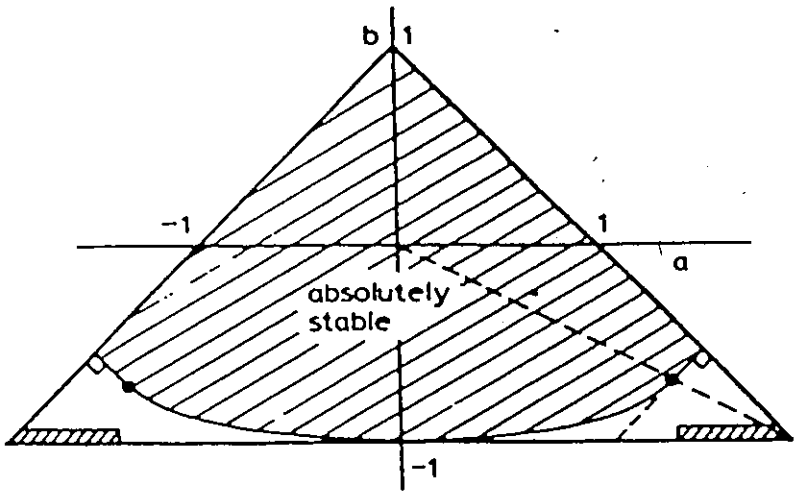


Figure 2.9 Limit Cycle Due to 1 MT Quantizer: Claasen et al

3. Fettweis and Meerkotter have shown that for a certain class of wave digital filter it is possible to suppress all limit cycles by using fixed point two's complement number representation and magnitude truncation quantization [16]. With this structure the entire filter must be built as a complete block and may not be appropriate in certain applications where the modular approach is more desirable. Meerkotter and Wegener have derived a related second order section which suppresses all limit cycles [17]. However, the quantization noise of this filter structure is considerably higher than the one using a single magnitude

truncation quantizer [18].

4. Randomized quantization has been proposed by Kieburtz, Lawerence and Mina [19]. This quantization approach combines the advantage of magnitude truncation as regards to limit cycles and that of roundoff as regards to quantization noise. It consists of switching randomly between MT and RO quantization, with RO being used for a larger portion of the time. In this manner the correlated nature of the quantization error is broken and so are the limit cycles. The disadvantage of this approach is that there is no guarantee that limit cycles are completely avoided. There are examples of d.c. deadbands which cannot be broken up by this quantization method. These are the coefficient-signal products which roundoff and truncate to the same integers. In a situation where second order filter sections are cascaded to form a higher order filter, these d.c. limit cycles, acting as inputs, will cause other types of limit cycles in the subsequent sections.

5. Controlled quantization was first proposed by Butterweck [20]. Based on the consideration of an "energy" function, an algorithm is given which guarantees the absence of limit cycles of periods longer than two. The implementation of this algorithm requires that quantization takes place after the addition of the two full-precision product signals.

The sum is then quantized either up (i.e. the next higher integer) or down (i.e. the next lower integer) depending on the state variables in the filter. Lawrence and Mitra further improve this scheme so that all limit cycles are eliminated [21]. More details of this controlled quantization method will be given in chapter 5.

CHAPTER 3 CRITERION FOR A LIMIT CYCLE FREE FILTER

3.1 INTRODUCTION

In order to employ controlled quantization as a means to suppress limit cycles, a criterion whereby stability is guaranteed, is required to determine in what way the signal-coefficient product is to be quantized; that is, whether to quantize up to the next higher integer or to quantize down to the next lower integer. It is important to note the difference between the conventional roundoff quantization and the controlled quantization. In roundoff, the signal is quantized to the next higher or the next lower integer depending on the magnitude of the signal itself. In controlled quantization, the direction of quantization is dictated by a control signal. To emphasis this characteristic, the terminology 'quantize up' is used to mean 'quantize to the next higher integer' and 'quantize down' means 'quantize to the next lower integer'. The criterion for controlled quantization developed here is presented in this chapter. The approach employed is based on the Lyapunov Stability Analysis.

Lyapunov's Second Method (also known as the Direct Method) is a general method for determining system stability. It is

applicable to linear or nonlinear systems, as well as to stationary or time-varying systems. One attractive feature of the Second Method is that it answers questions of system stability, utilizing the form of the system equations without requiring explicit knowledge of their solutions.

This chapter is organized as follows: section 3.2 contains the preliminary material. The various stability concepts and other required terminologies are defined. In section 3.3 the main theorem is presented. It gives the conditions which guarantee the system's stability. This main theorem is applied to the stability analysis of a linear second order digital filter. Results from this exercise are used in section 3.4 in which the stability of a practical second order digital filter with quantization nonlinearities is considered.

3.2 DEFINITIONS

3.2.1 System

In order to employ Lyapunov's Second Method it is necessary to describe a system from the point of view of 'states'. This section gives a brief description of a digital system in terms of its mathematical equations.

The behaviour of a dynamic digital system can be represented by a vector difference equation in the form of

$$\underline{X}(t_{k+1}) = F(\underline{X}(t_k);\ \underline{U}(t_k);\ t_k) \qquad (3.1)$$

where t_k indicates an independent, discrete time variable, $t_{k+1} > t_k$ for all integers k, $-\infty < k > \infty$, and $t_k \rightarrow \infty$ as $k \rightarrow \infty$. The vector $\underline{X}$ is the state of the system and its components x_i are the state variables. The vector $\underline{U}$ is the forcing function or the input of the system. (3.1) is the equivalent of a set of n scalar difference equations

$$x_i(t_{k+1}) = f_i(x_1(t_k) \ldots . x_n(t_k);\ u_1(t_k) \ldots . u_m(t_k);\ t_k) \qquad (3.2)$$

$$i = 1 \ldots \ldots \ldots n$$

The integer n is the order of the system and in general $m < n$. It is assumed that f is a 1-to-1 function for any fixed $\underline{U}$ and t_k, continous in all of its arguments. If $\underline{U}(t_k) = 0$ for all t_k, the system is described as being free or unforced

$$\underline{X}(t_{k+1}) = F(\underline{X}(t_k);\ t_k) \qquad (3.3)$$

For a given initial condition X_0 and initial time t_0, the solution of (3.3) is denoted by

$$\Phi(t_k;\ X_0,\ t_0) = X(t_k) \tag{3.4}$$

(3.4) also satisfies the following relations:

$$\Phi(t_0;\ X_0,\ t_0) = X(t_0) \tag{3.5}$$

for all X_0 and t_0

$$\Phi(t_a;\ X_c,\ t_c) = \Phi(t_a;\ \Phi(t_b;\ X_c,\ t_c),\ t_b), \tag{3.6}$$

for all $t_a \geq t_b \geq t_c$

If a state X_e satisfies the following

$$F(X_e,\ t_k) = X_e \quad \text{for all } t_k \tag{3.7}$$

then it is called the equilibrium state of the system. It is assumed that $F(0,t_k) = 0$, i.e. $X = 0$ is an equilibrium state of the system.

3.2.2 Stability

In the discussion of system stability there are two possible approaches. In one, stability is considered in the absence of a system input. An equilibrium state is presumed to exist and the focus is on the system's ability to maintain a state in the vicinity of the equilibrium as $t \rightarrow \infty$. In the other, the system is subjected to a bounded input and the concern then is with the resulting behaviour of the state variables. Lyapunov's Second Method deals with the stability of an equilibrium. For this reason it is applicable to the analysis of limit cycles since these oscillations are instabilities with respect to the equilibrium at the origin.

Up until now the term 'stability' has been used rather loosely. The following definitions (see References 23, 24) state precisely the three types of stability which are relevant to the discussion of Lyapunov's Second Method.

L-stability: the equilibrium state X_e is said to be L-stable (stable in the sense of Lyapunov) if, for any t_0 and any number $\varepsilon > 0$, there corresponds a number $\delta(\varepsilon, t_0) > 0$ such that if $\| X_0 - X_e \| \leq \delta(\varepsilon, t_0)$ then $\| \Phi(t_k; X_0, t_0) \| \leq \varepsilon$ for

all $t_k \geq t_0$. ($\| \|$ denotes the Euclidean norm defined by

$\| X \| = [(x_1)^2 + \ldots + (x_n)^2]^{1/2}$).

If δ does not depend on t_0, then the equilibrium state is said to be uniformly stable.

Asymptotical stability: the equilibrium state X_e is asymptotically stable if it is L-stable and if there exists a number $r(t_0) > 0$ such that

$$\lim \| \Phi(t_k; X_0, t_0) \| = X_e \qquad (3.8)$$

for all $\| X_0 - X_e \| < r(t_0)$; and if r is independent of t_0 then the equilibrium state is uniformly asymptotically stable.

The difference between an L-stable equilibrum and an asymptotically stable equilibrium is that in the former case the system state stays close to the equilibrium and in the latter the system state converges to the equilibrium as t increases indefinitely. Note that both types of stability are local concepts and it is usually not known how small ε or r has to be. Therefore it is not guaranteed that a system

possessing either one of these two types of stability will also operate properly.

Asymptotical stability in the large: the equilibrium state X_e is said to be asymptotically stable in the large if it is asymptotically stable for any initial state X_0.

Two other definitions are required in the discussion of Lyapunov's Second Method.

Positive definite: a scalar function $E(X; t_k)$ is said to be positive definite in a neighbourhood Π of the point $X = 0$ if $E(0; t_k) = 0$ and if there exists a continuous, nondecreasing scalar function W such that $W(0) = 0$ and

$$E(X; t_k) \geq W(\| X \|) \tag{3.9}$$

for all t_k and all X in Π.

Quadratic form: a scalar function $E(X) = X^t P X$ is called a quadratic form if X is a n-element vector, (X^t denotes the transpose of X), and P is a n*n symmetric matrix. The positive definiteness of the quadratic form $E(X)$ can be

determined by the Sylvester's criterion which states that the necessary and sufficient condition for $E(\underline{X})$ to be positive definite are that all the successive principal minors of $\underline{P}$ be positive, i.e.

$$p_{11} > 0, \quad \begin{vmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{vmatrix} > 0, \ \ldots\ , \quad \begin{vmatrix} p_{11} \cdots\cdots p_{1n} \\ \\ p_{1n} \cdots\cdots p_{nn} \end{vmatrix} > 0$$

3.3 LYAPUNOV'S SECOND METHOD

3.3.1 The Main Stability Theorem

It is commonly known in the theory of mechanics that a vibratory system is stable if its total energy (a positive definite function) decreases continually (the time derivative of the energy function is negative) until a minimum is reached at the equilibrium state. In other words, a dissipative system is a stable system since it will always return to its equilibrium after a perturbation. Lyapunov's Second Method is a generalization of this physical phenomenon. In the Second Method a pseudo energy function, namely the 'Lyapunov

function' is used. Any scalar function satisfying the stipulations in Lyapunov's stability theorem (see below) can serve as a Lyapunov function for a given system. Besides being widely applicable, the Second Method is also more stringent than the conventional energy consideration. In most cases the energy function can serve as the Lyapunov function. However, a system with energy (E) decreasing on the average, but not at every time instant, is considered stable; but E is not a Lyapunov function since it does not satisfy condition (ii) listed below.

Lyapunov's Stability Theorem states that if for a given system a positive definite function $E(X)$ can be found such that its time derivative along a trajectory is always negative, except at the equilibrium where it is zero, then the system is asymptotically stable.

For the discrete-time, unforced system we have:

$$X(t_{k+1}) = F(X(t_k), t_k) \qquad (3.9)$$

where $F(0,t_k) = 0$ for all t_k. Suppose there exists a scalar function $E(X,t_k)$ such that $E(0,t_k) = 0$ for all t_k and

(i) $E(X,t_k)$ is positive definite

$$\text{(ii)}\quad \Delta E(\underline{X},t_k) = \frac{E(\Phi(t_{k+1};\ \underline{X},\ t_k);\ t_{k+1}) - E(\underline{X},t_k)}{t_{k+1} - t_k}$$

is negative definite and

$$\text{(iii)}\quad E(\underline{X},t_k) \rightarrow \infty \text{ as } \| \underline{X} \| \rightarrow \infty$$

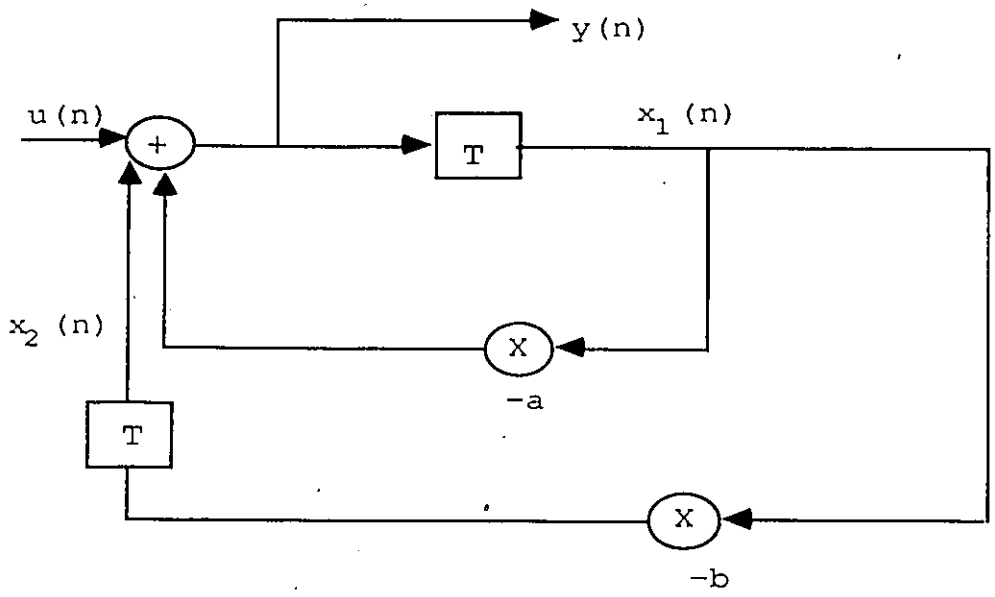
then the equilibrium state $\underline{X}_e = 0$ is asymptotically stable in the large and $E(\underline{X},t_k)$ is a Lyapunov function of the system (3.9).

For asymptotical stability, only condition (i) and (ii) are required; and for L-stability condition (ii) can be relaxed to $\Delta E(\underline{X},t_k) \leq 0$.

3.3.2 Stability of the Second Order Recursive Digital Filter

The application of the Second Method consists of defining a Lyapunov function possessing the appropriate conditions for the desired type of stability (e.g. L-stability) for the system under consideration. The analysis is shown for the second order recursive digital filter section in Figure 3.1. This second order section, with one of the multipliers moved and placed ahead of the delay element, is functionally identical to the one shown

in Figure 2.4 [27].



$$y(n) = u(n) - [ax_1(n)]_Q + x_2(n)$$

Figure 3.1 Second Order Filter Section Variation

For $\underline{U} \equiv 0$ the state equation of the filter is:

$$\begin{bmatrix} x_1(kT+T) \\ x_2(kT+T) \end{bmatrix} = \begin{bmatrix} -a & 1 \\ -b & 0 \end{bmatrix} \times \begin{bmatrix} x_1(kT) \\ x_2(kT) \end{bmatrix} \qquad (3.10)$$

or

$$\underline{X}(kT+T) = \underline{A}\underline{X}(kT) \qquad (3.11)$$

Define as a Lyapunov function the following positive definite

quadratic form

$$E(\underline{X}) = \underline{X}^t \underline{Q} \underline{X} \qquad (3.12)$$

In accordance with the requirements of the main theorem

$$E(\underline{X}) > 0 \quad \text{for} \quad \underline{X} \neq 0 \quad \text{and}$$

$$E\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

In (3.12), $\underline{Q}$ is a symmetric, constant and positive definite matrix to be determined. Let $\underline{R}$ be another symmetric, positive definite matrix such that

$$\Delta E(\underline{X}) = \underline{X}^t \underline{R} \underline{X} \qquad (3.13)$$

where ΔE is defined as

$$\Delta E(\underline{X}) \triangleq E(\underline{X}(kT)) - E(\underline{X}(kT + T)) \qquad (3.14)$$

Note that ΔE is defined as the backward difference and therefore is required to be positive definite instead of negative definite as stated in section 3.3.1 where ΔE is defined as the forward difference.

Substituting (3.11) and (3.12) into (3.14) and making use of

the fact that $(\underline{A}\underline{X})^t = \underline{X}^t\underline{A}^t$ we have

$$\begin{aligned}\Delta E(\underline{X}) &= \underline{X}^t(kT)\underline{Q}\underline{X}(kT) - \underline{X}^t(kT+T)\underline{Q}\underline{X}(kT+T) \\ &= \underline{X}^t(kT)\underline{Q}\underline{X}(kT) - (\underline{A}\underline{X}(kT))^t\underline{Q}\underline{A}\underline{X}(kT) \\ &= \underline{X}^t(kT)\underline{Q}\underline{X}(kT) - \underline{X}^t(kT)\underline{A}^t\underline{Q}\underline{A}\underline{X} \\ &= \underline{X}^t(kT)[\underline{Q} - \underline{A}^t\underline{Q}\underline{A}]\underline{X}(kT) \qquad (3.15)\end{aligned}$$

Equating (3.14) and (3.15) we get

$$\underline{R} = \underline{Q} - \underline{A}^t\underline{Q}\underline{A} \qquad (3.16)$$

$\underline{Q}$ can be found by solving equation (3.16) for a given symmetric positive matrix $\underline{R}$. If it can be shown that the matrix $\underline{Q}$ so obtained is also symmetric positive definite, then, in accordance with Lyapunov's stability theorem, the filter is asymptotically stable.

A simple choice for $\underline{R}$ is:

$$\underline{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (3.17)$$

then (3.16) can be written as

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} - \begin{bmatrix} -a & -b \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} -a & 1 \\ -b & 0 \end{bmatrix} \qquad (3.18)$$

which yields the following three equations:

$$q_{11} - a^2 q_{11} - 2abq_{12} - b^2 q_{22} = 1$$

$$q_{12} + aq_{11} + bq_{12} = 0$$

$$q_{22} - q_{11} = 1$$

Solving these three equations simultaneously we have

$$q_{11} = \frac{(1+a)\ (1+b)}{\Delta} \tag{3.19}$$

$$q_{12} = \frac{-a(1+b^2)}{\Delta} \tag{3.20}$$

$$q_{22} = \frac{(1+b)\ (2-a^2) + 2a^2 b}{\Delta} \tag{3.21}$$

where $\Delta = (1-b)\ (1+b+a)\ (1+b-a)$

It is easy to show that if a, b lie inside the linear stability triangle then $\underline{Q}$ is positive definite; that is

$$q_{11} > 0$$

$$q_{22} > 0$$

$$q_{11}q_{22} - q_{12}^2 > 0 .$$

3.4 Development of the Controlled Quantization Criterion

The same structure as shown in Figure 3.1, but with error signals introduced by product quantization after each multiplier, is now considered.

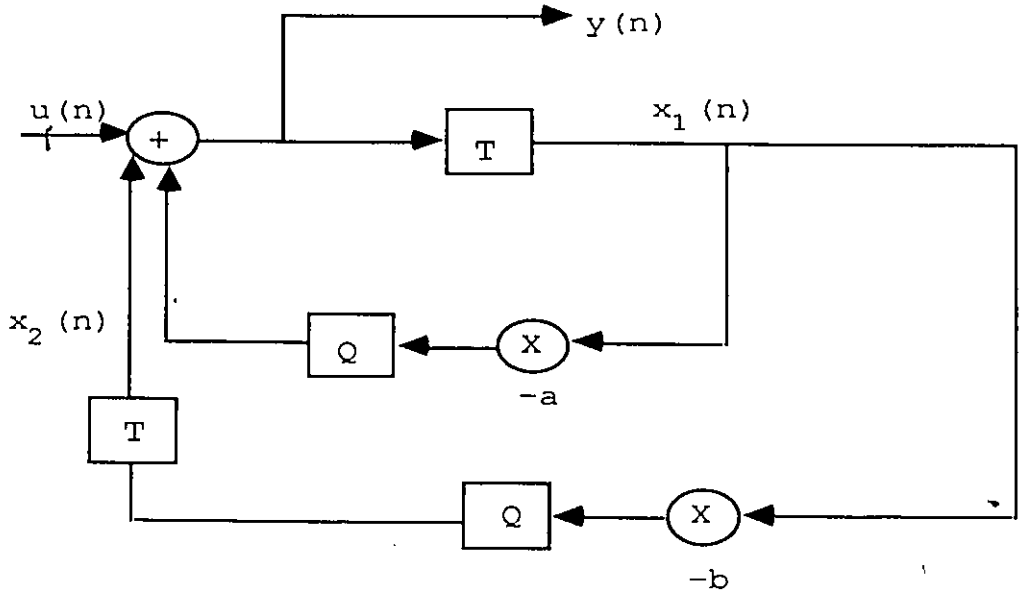


Figure 3.2 Second Order Filter Section with Quantization Error Signals

For zero input, the state equation can be written as

$$\begin{bmatrix} x_1(kT+T) \\ x_2(kT+T) \end{bmatrix} = \begin{bmatrix} -a & 1 \\ b & 0 \end{bmatrix} \begin{bmatrix} x_1(kT) \\ x_2(kT) \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \qquad (3.22)$$

or

$$\underline{X}(kT+T) = \underline{A}\underline{X}(kT) + \underline{\alpha} \qquad (3.23)$$

where α_i are the quantization errors, and $0 \leq |\alpha_i| < 1$.

Using (3.11) and (3.12) we can write

$$\Delta E(X) = \underline{X}^t(kT)\underline{Q}\underline{X}(kT) - [\underline{A}\underline{X}(kT) + \underline{\alpha}]^t\underline{Q}[\underline{A}\underline{X}(kT) + \underline{\alpha}] \qquad (3.24)$$

Rearranging terms (3.24) becomes

$$\Delta E(\underline{X}) = \underline{X}(kT)[\underline{Q} - \underline{A}^t\underline{Q}\underline{A}]\underline{X}(kT) - [\underline{\alpha}^t\underline{Q}\underline{A}\underline{X}(kT)]^t - \underline{\alpha}^t\underline{Q}\underline{A}\underline{X}(kT) - \underline{\alpha}^t\underline{Q}\underline{\alpha} \qquad (3.25)$$

Making use of the fact that $\underline{R} = \underline{Q} - \underline{A}^t\underline{Q}\underline{A} = \underline{I}$ and $\underline{Q} = \underline{Q}^t$, equation (3.25) can be simplified to

$$\Delta E(\underline{X}) = \underline{X}^t(kT)\underline{X}(kT) - 2\underline{\alpha}^t\underline{Q}\underline{A}\underline{X}(kT) - \underline{\alpha}^t\underline{Q}\underline{\alpha} \qquad (3.26)$$

Writing (3.26) in scalar form we get

$$\begin{aligned}\Delta E = {} & [x_1 + (q_{12}a + q_{22}b)\alpha_2 - q_{12}\alpha_1]^2 + [x_2 - (q_{11}\alpha_1 + q_{12})]^2 \\ & - [(q_{12}a + q_{22}b)\alpha_2 - q_{12}\alpha_1]^2 - (q_{11}\alpha_1 + q_{12}\alpha_2)^2 \\ & - \alpha_1^2 q_{11} - \alpha_2^2 q_{22} - 2\alpha_1\alpha_2 q_{12}\end{aligned} \qquad (3.27)$$

where the time argument for the state variables has been omitted and the q_{ij}'s are given by (3.19) to (3.21). To insure asymptotic stability we must have $\Delta E > 0$, i.e.

$$[x_1 + (q_{12}a + q_{22}b)\alpha_2 - q_{12}\alpha_1]^2 + [x_2 - (q_{11}\alpha_1 + q_{12}\alpha_2)]^2 > \alpha_1^2 q_{11} + \alpha_2^2 q_{22} + 2\alpha_1\alpha_2 q_{12} + [(q_{12}a + q_{22}b)\alpha_2 - q_{12}\alpha_1]^2 + (q_{11}\alpha_1 + q_{12}\alpha_2)^2 \quad (3.28)$$

In (3.28) if the 'greater than' sign is replaced by an 'equal' sign the resulting equation defines a circle in the state-space (x_1 vs x_2) plane centered at

$$x_{1c} = q_{12}\alpha_1 - (q_{12}a + q_{22}b)\alpha_2 \quad (3.29)$$

$$x_{2c} = q_{11}\alpha_1 + q_{12}\alpha_2 \quad (3.30)$$

The radius of this circle is given by

$$R_o = \{\alpha_1^2 q_{11} + \alpha_2^2 q_{22} + 2\alpha_1\alpha_2 q_{12} + [(q_{12}a + q_{22}b)\alpha_2 - q_{12}\alpha_1]^2 + (q_{11}\alpha_1 + q_{12}\alpha_2)^2\}^{1/2} \quad (3.31)$$

With the above notation (3.28) becomes

$$(x_1 - x_{1c})^2 + (x_2 - x_{2c})^2 > R_o^2 \quad (3.32)$$

(3.32) can be interpreted as follows: for every state $[x_1(kT), x_2(kT)]$ of the filter, the quantization errors (α_1, α_2) (associated with one particular quantization combination of $[x_1, x_2]$), define a circle in the x_1-x_2 plane. If the pair $[x_1(kT), x_2(kT)]$ lies outside of this circle then ΔE is greater than 0 and the function $E(\underline{X})$ is a Lyapunov function of the filter. If this is the case then the second order filter with quantization nonlinearities is asymptotically stable. This interpretation will be made use of in the implementation of the limit cycle free filter.

CHAPTER 4 APPLICATION AND IMPLEMENTATION OF THE LIMIT CYCLE FREE CRITERION

4.1 INTRODUCTION

In order to employ the stability criterion derived in the previous chapter to suppress limit cycles, it is necessary to show that the criterion can be satisfied in the quantization environment of the second order digital filter. Such a proof is presented in the next section. An implementation algorithm of the criterion is described in section 4.3. This is followed by numerical examples from computer simulation showing the limit-cycle-free output of the filter. The incorporation of the limit cycle suppression algorithm in the Stored Product Digital Filter (SPDF) is described in section 4.5.

4.2 THE QUANTIZATION ENVIRONMENT

The first backward difference of the Lyapunov function $E(\underline{X})$ derived for the second order recursive filter in Chapter 3 is:

$$\Delta E(\underline{X}) = \underline{X}^t\underline{X} - 2\underline{\alpha}^t\underline{Q}\underline{A}\underline{X} - \underline{\alpha}^t\underline{Q}\underline{\alpha} \qquad (4.1)$$

where

$\underline{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ is the state vector of the filter

$\underline{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$ is the quantization error vector associated with the products $-ax_1$ and $-bx_1$ respectively

$Q = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix}$ is positive definite with q_{11}, $q_{22} > 0$ and $q_{12} > 0$ for $a > 0$, $q_{12} < 0$ for $a < 0$

$\underline{A} = \begin{bmatrix} -a & 1 \\ -b & 0 \end{bmatrix}$ is the state matrix of the filter

According to the Lyapunov stability theory, the filter is asymptotically stable if ΔE is positive for all t(nT). It is therefore required to prove that for at least one possible combination of quantization (i.e. for one particular $\underline{\alpha}$), ΔE is greater than zero.

The method of quantization used is either magnitude truncation (MT) or magnitude enhancement (ME); i.e., a product containing a fractional part is quantized to either the next lower (MT or quantize down) or the next higher (ME or quantize up) integer. The quantization error is defined as

α_i = quantized product i - exact product i, i = 1, 2

Of particular importance is the fact that α_i is either positive or negative depending on the sign of the product being quantized and whether ME (quantized up) or MT (quantized down) has been applied to the product.

For the second order digital filter there are four possible combinations of quantization, they are:

(1) UU: $-ax_1$ and $-bx_1$ are both quantized up

(2) UD: $-ax_1$ is quantized up and $-bx_1$ is quantized down

(3) DU: $-ax_1$ is quantized down and $-bx_1$ is quantized up

(4) DD: $-ax_1$ and $-bx_1$ are both quantized down

The possible sign combinations of $(\alpha_1\ \alpha_2)$ are:

(+ +), (+ -), (- +), (- -).

4.2.1 Proof of Applicability

The following proof will demonstrate that, among the four possible combinations of quantization for the two signal-coefficient products, at least one of them will satisfy

the requirement that equation (4.1) is positive.

Proof: The first term in (4.1), $\underline{X}^t\underline{X}$, is a positive definite quadratic due to the states of the filter and is positive for $\underline{X} \neq 0$. The second and third term are due to product quantization errors. Note that the third term, $\underline{\alpha}^t\underline{Q}\underline{\alpha}$, is also a positive definite quadratic since $\underline{Q}$ is positive definite. In order to prove that ΔE is positive for a given $\underline{X}$ ($\underline{X} \neq 0$) and a possible $\underline{\alpha}$, we will show that with the available quantization choices it is always possible to make the second term in (4.1) negative and have magnitude greater than or equal to that of the third term. If this is the case then the two terms due to product quantization errors will at least cancel each other and ΔE is quaranteed to be positive.

Rewrite the second term in equation (4.1) as:

$$2\underline{\alpha}^t\underline{Q}\underline{A}\underline{X} = 2\underline{\alpha}^t\underline{Q}\underline{V} \tag{4.2}$$

where $\underline{V} = \underline{A}\underline{X}$, i.e. $v_1 = -ax_1 + x_2$
$v_2 = -bx_1$

then expanding the matrix expression on the right we get:

$$\underline{\alpha}^t\underline{Q}\underline{V} = \begin{bmatrix}\alpha_1 & \alpha_2\end{bmatrix} \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$= \alpha_1 v_1 q_{11} + \alpha_2 v_2 q_{22} + \alpha_1 v_2 q_{12} + \alpha_2 v_1 q_{12} \tag{4.3}$$

This is similar in form to the third term in (4.1):

$$\underline{\alpha}^T Q \underline{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$$

$$= \alpha_1^2 q_{11} + \alpha_2^2 q_{22} + 2\alpha_1 \alpha_2 q_{12} \qquad (4.4)$$

Two possibilities exist regarding the sign of the individual terms in (4.3) which can make $\Delta E > 0$:

(1) all 4 terms can be made negative.

(2) 2 out of 4 terms can be made negative.

Case 1: all 4 terms are negative

In order that all 4 terms in (4.3) can be made negative, one of the following conditions must be true:

a) v_1, v_2 are of the same sign and q_{12} is positive.

b) v_1, v_2 are of different signs and q_{12} is negative.

If one of these conditions exists then to make all 4 terms negative one only needs to quantize each product so that v_i and α_i have different signs. It can be easily checked (see Tables

I and II below) that the quantization combination DD (or UD as explained in the footnote of Table I) is the appropriate choice. Since all 4 terms are negative, the value of equation (4.3) must be negative. Also since both products are magnitude truncated we have $|v_1| \geq |\alpha_1|$ and $|v_2| \geq |\alpha_2|$*. Comparing terms in (4.3) and (4.4) it follows that the magnitude of (4.3) must be greater than that of (4.4).

* The two inequalities are still true if UD has been used instead of DD. In this case we can express v_1 as:

$$|v_1| \geq 1 - |\text{fractional part of } -ax_1|$$

and the quantization error due to magnitude enhancement is:

$$|\alpha_{1u}^{\oplus}| = 1 - |\alpha_{1d}^{\oplus}|$$
$$= 1 - |\text{fractional part of } -ax_1|$$

Hence $|v_1| \geq |\alpha_1|$

$^{\oplus}$ The subscripts 'u' and 'd' in α_i are used to emphsize the fact that the error is associated with magnitude enhancement (quantize up) and magnitude truncation (quantize down) respectively.

Sgn	Sgn	Quant.*	Sign of Terms in (4.3)				Sign of Terms in (4.4)			
$(v_1\ v_2)$	$(\alpha_1\alpha_2)$		$\alpha_1 v_1 q_{11}$	$\alpha_2 v_2 q_{22}$	$\alpha_1 v_2 q_{12}$	$\alpha_2\ v_1 q_{12}$	$\alpha_1^2 q_{11}$	$\alpha_2^2 q_{22}$	$\alpha_1\alpha_2 q_{12}$	$\alpha_1\alpha_2 q_{12}$
(+ +)	(- -)	DD	-	-	-	-	+	+	+	+
	(- +)	DU	-	+	-	+	+	+	-	-
	(+ -)	UD	+	-	+	-	+	+	-	-
	(+ +)	UU	+	+	+	+	+	+	+	+
(- -)	(+ +)	DD								
	(+ -)	DU		SAME	AS	ABOVE				
	(- +)	UD								
	(- -)	UU								

Table I v_1 v_2 are of the same sign and q_{12} is positive

* The corresponding quantizations are as shown for the case where v_1 has the same sign as the product $-ax_1$. However, if the sign of v_1 is reversed as a result of x_2 being greater than and has opposite sign to $-ax_1$ (recall that $v_1 = -ax_1 + x_2$) then the quantization of the first product should also be reversed (i.e. D to U and U to D) so that $\text{sgn}(\alpha_1\ \alpha_2)$ is consistent with those shown in the Table.

Sgn	Sgn	Quant.*	Sign of Terms in (4.3)				Sign of Terms in (4.4)			
$(v_1 \ v_2)$	$(\alpha_1 \alpha_2)$		$\alpha_1 v_1 q_{11}$	$\alpha_2 v_2 q_{22}$	$\alpha_1 v_2 q_{12}$	$\alpha_2 v_1 q_{12}$	$\alpha_1^2 q_{11}$	$\alpha_2^2 q_{22}$	$\alpha_1 \alpha_2 q_{12}$	$\alpha_1 \alpha_2 q_{12}$
(+ -)	(- +)	DD	-	-	-	-	+	+	+	+
	(- -)	DU	-	+	-	+	+	+	-	-
	(+ +)	UD	+	-	+	-	+	+	-	-
	(+ -)	UU	+	+	+	+	+	+	+	+
(- +)	(+ -)	DD								
	(+ +)	DU		SAME	AS	ABOVE				
	(- -)	UD								
	(- +)	UU								

Table II v_1 v_2 are of different signs and q_{12} is negative

Case 2: 2 out of 4 terms are negative

Two other possible conditions exist regarding the signs of v_1, v_2 and q_{12}. They are:

a) v_1, v_2 are of the same sign and q_{12} is negative

b) v_1, v_2 are of different signs and q_{12} is positive

Tables III and IV show that for both of these conditions any quantization combination will make 2 out of 4 terms in (4.3) negative. With 2 out of 4 terms negative it is always possible to make (4.3) negative by negating the the two terms with the larger amplitudes. It then remains to be shown that when equation (4.3) is made negative (by the appropriate quantization choice) it is also greater than or equal to the magnitude of the corresponding equation (4.4).

Sgn	Sgn	Quant.*	Sign of Terms in (4.3)				Sign of Terms in (4.4)			
$(v_1\ v_2)$	$(\alpha_1\alpha_2)$		$\alpha_1 v_1 q_{11}$	$\alpha_2 v_2 q_{22}$	$\alpha_1 v_2 q_{12}$	$\alpha_2 v_1 q_{12}$	$\alpha_1^2 q_{11}$	$\alpha_2^2 q_{22}$	$\alpha_1\alpha_2 q_{12}$	$\alpha_1\alpha_2 q_{12}$
(+ +)	(- -)	DD	-	-	+	+	+	+	-	-
	(- +)	DU	-	+	+	-	+	+	+	+
	(+ -)	UD	+	-	-	+	+	+	+	+
	(+ +)	UU	+	+	-	-	+	+	-	-
(- -)	(+ +)	DD								
	(+ -)	DU		SAME	AS	ABOVE				
	(- +)	UD								
	(- -)	UU								

Table III v_1 v_2 are of the same sign and q_{12} is negative

Sgn	Sgn	Quant.*	Sign of Terms in (4.3)				Sign of Terms in (4.4)			
$(v_1\ v_2)$	$(\alpha_1\alpha_2)$		$\alpha_1 v_1 q_{11}$	$\alpha_2 v_2 q_{22}$	$\alpha_1 v_2 q_{12}$	$\alpha_2 v_1 q_{12}$	$\alpha_1^2 q_{11}$	$\alpha_2^2 q_{22}$	$\alpha_1\alpha_2 q_{12}$	$\alpha_1\alpha_2 q_{12}$
(+ -)	(- +)	DD	-	-	+	+	+	+	-	-
	(- -)	DU	-	+	+	-	+	+	+	+
	(+ +)	UD	+	-	-	+	+	+	+	+
	(+ -)	UU	+	+	-	-	+	+	-	-
(- +)	(+ -)	DD								
	(+ +)	DU		SAME	AS	ABOVE				
	(- -)	UD								
	(- +)	UU								

Table IV $v_1\ v_2$ are of different signs and q_{12} is positive

(I) First consider the case when $|v_1| > |v_2|$ then $|\alpha_1 v_1 q_{11}| > |\alpha_1 v_2 q_{12}|$ (refer to Tables III and IV). Depending on the magnitudes of the remaining 2 terms either DU or DD or both will make equation (4.3) negative; that is, the negative terms will be greater in magnitude than the positive terms.

Assume that only DU makes (4.3) negative then the following must be true:

i) $|\alpha_2 v_1 q_{12}| > |\alpha_2 v_2 q_{22}|$

ii) $(1 - |\alpha_{2u}|)(|v_1 q_{12}| - |v_2 q_{22}|) > |\alpha_1|(|v_1 q_{11}| - |v_2 q_{12}|)$

(ii) is derived from the assumption that the quantization DD will make (4.3) positive. It implies that $(1 - |\alpha_{2u}|) > |\alpha_1|$ since the second expression on the left side is smaller than the second expression on the right side.

In order that the magnitude of (4.3) be greater than that of (4.4) we must have:

$$2|\alpha_1|(|v_1 q_{11}| - |v_2 q_{12}|) + 2|\alpha_2|(|v_1 q_{12}|) - |v_2 q_{22}| > \alpha_1^2 q_{11} + \alpha_2^2 q_{22} + 2\alpha_1 \alpha_2 q_{12} \qquad (4.5)$$

It can easily be checked that inequality (4.5) is true if $|v_1| \geq |v_2| + |\alpha_1| + |\alpha_2|$. Considering the quantization used (i.e. DU), v_1 and v_2 can be expressed as:

$|v_1| = |I| + |\alpha_1|$ where I is the integer part of v_1

$|v_2| = |J| + (1-|\alpha_{2u}|)$ where J is the integer part of v_2

Now if $|I| = |J|$ then $|\alpha_1|$ must be greater than $(1 - |\alpha_{2u}|)$. However, this cannot be true because of (ii) above. Then $|I|$ must be greater than $|J|$; and since I, J are integers we must have $|I| \geq |J| + 1$. If this is the case then

$$|v_1| - |v_2| \geq |J| + 1 + |\alpha_1| - |J| - (1 - |\alpha_2|)$$
$$\geq |\alpha_1| + |\alpha_2|$$

or

$$|v_1| \geq |v_2| + |\alpha_1| + |\alpha_2|^{\dagger\dagger}$$

If $|v_1| < |v_2| + |\alpha_1| + |\alpha_2|$ then as shown above the quantization DU will not guarantee that the magnitude of (4.3) is greater than that of (4.4). However in this case condition (ii) above must also be false. We then have the situation where both DU and DD will make (4.3) negative. If DD is used we require that:

$$2|\alpha_1|(|v_1q_{11}| - |v_2q_{12}|) - 2|\alpha_2|(|v_1q_{12}| - |v_2q_{22}|) > \alpha_{12}q_{11} + \alpha_{22}q_{22} - 2|\alpha_1\alpha_2q_{12}| \quad (4.6)$$

and v_1, v_2 can be expressed as:

$$|v_1| = |I| + |\alpha_1|$$

$$|v_2| = |J| + |\alpha_2|$$

If $|I| = |J|$ then $|\alpha_1| > |\alpha_2|$; and if $|I| > |J|$ then we have either $|v_1| > |v_2| + |\alpha_1|$ or $|v_1| > |v_2| + |\alpha_2|$. It can easily be checked that under these conditions inequality (4.6) is true.

†† In the case where the sign of v_1 has been reversed and the first product $-ax_1$ is quantized up instead of down, we have

$$|v_1| = |I| + (1 - |\text{fractional part of } -ax_1|)$$
$$= |I| + (1 - |\alpha_{1d}|)$$
$$= |I| + |\alpha_{1u}|$$

Since $|v_1|$ can be expressed in the same way in terms of the sum of an integer part and the quantization error as in the case where MT is used, the same proof applies.

(II) Next consider the case where $|v_2| > |v_1|$. If $|v_2| > |v_1|$ then $|\alpha_1 v_2 q_{22}| > |\alpha_2 v_1 q_{12}|$. Depending on the magnitudes of the remaining 2 terms either ŌD or DD or both will make equation (4.2) negative. The proof procedure is the same as when $|v_1| > |v_2|$ and will not be repeated here.

4.3 LIMIT CYCLE SUPPRESSION ALGORITHM

It has now been shown that the limit cycle free criterion developed in Chapter 3 can be satisfied in the quantization environment of a second order digital filter, where the coefficient-signal products are quantized individually. A limit cycle suppression algorithm based on equation (4.1), which can be employed in a practical implementation scheme, will be presented in this section.

It was mentioned briefly in Chapter 3 that equation (3.32) can be interpreted as a function of $(\alpha_1\ \alpha_2)$ and that if the inequality sign is replaced by an equal sign it defines a circle on the x_1-x_2 plane. In order to eliminate limit cycles we must

ensure that the filter states $[x_1(kT)\ x_2(kT)]$ lie outside of this circle. In other words, we must choose the quantization combination resulting in a $(\alpha_1\ \alpha_2)$ pair such that the circle defined by (3.32) on the x_1-x_2 plane excludes the current filter states $x_1(kT)$ and $x_2(kT)$.

Equation (3.32) can be rewritten as:

$$x_2 \gtrless x_{2c} \pm \sqrt{R_o^2 - (x_1 - x_{1c})^2} \qquad (4.7)$$

where x_{1c}, x_{2c} and R_o are as defined in equations (3.29), (3.30) and (3.31) respectively.

For a given pair of state variables $[x_1(kT)\ x_2(kT)]$ and a quantization combination, the error pair $(\alpha_1\ \alpha_2)$ is known and hence are x_{1c}, x_{2c} and R_o. Equation (4.7) then allows us to calculate, for the given $x_1(kT)$ and $x_2(kT)$, where the state variable $x_2(kT)$ must not be. The geometric interpretation of (4.7) is given in Figure 4.1.

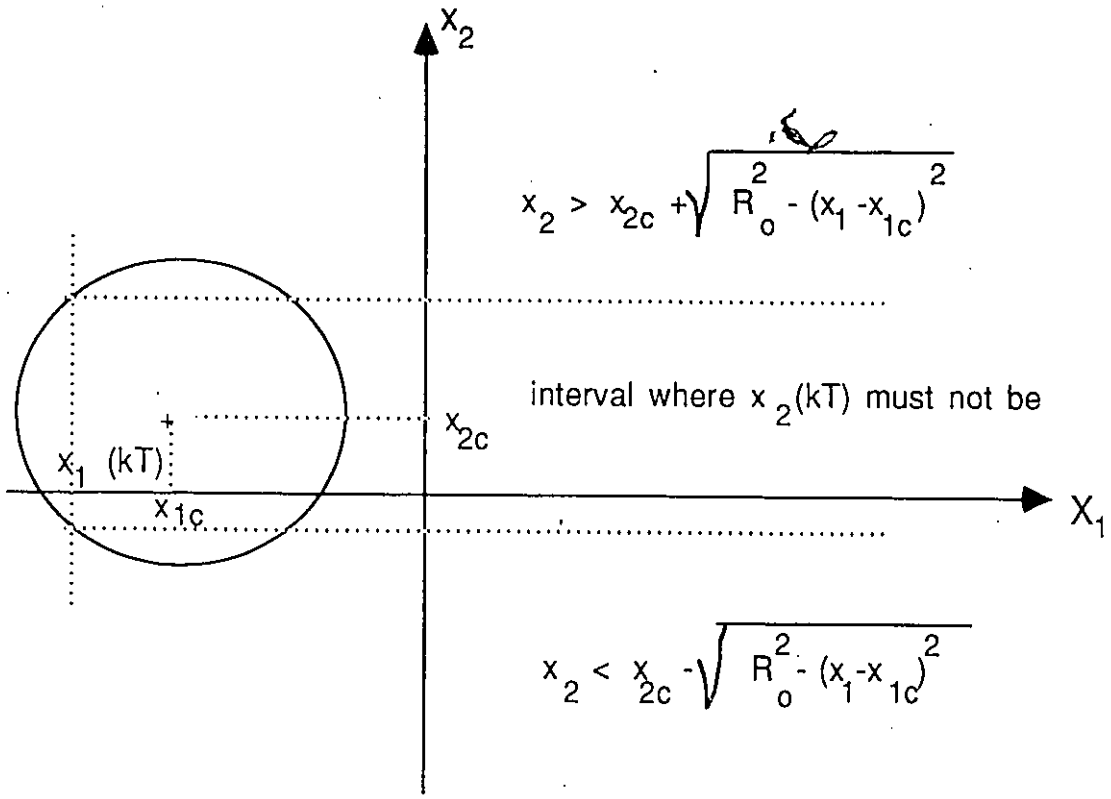


Figure 4.1 X_2 Intervals Described by Equation (4.7)

Since there are four possible combinations of quantization (q_i, i = 1, 2, 3,4), there are four possible circles defined by the corresponding (α_1 α_2) on the x_1-x_2 plane. Each of these circles will project an interval on the x_2-axis defining where $x_2(kT)$ must not be. Extensive experimentation shows that these x_2 intervals may overlap each other but at least two of them will be distinct. Two such disjoint intervals are represented

qualitatively in Figure 4.2.

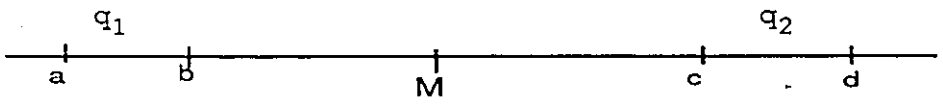


Figure 4.2 Intervals on the X_2-axis

A middle point, M, between these two intervals can easily be determined, that is, M = (b + c) / 2. A simple test involving the comparison of $x_2(kT)$ and M will decide which quantization procedure, that is either q_1 or q_2, should be used. If $x_2(kT)$ is greater than M then q_1 should be used, otherwise q_2 is used. It may happen that for a particular value of $x_1(kT)$ and a given pair of $(\alpha_1\ \alpha_2)$, the term under the square root sign in equation (4.7) is negative. This means that whatever value $x_2(kT)$ takes, the pair $[x_1(kT)\ x_2(kT)]$ is always outside of the circle and ΔE is positive.

The limit cycle suppression algorithm is summarized in the flow chart in Figure 4.3.

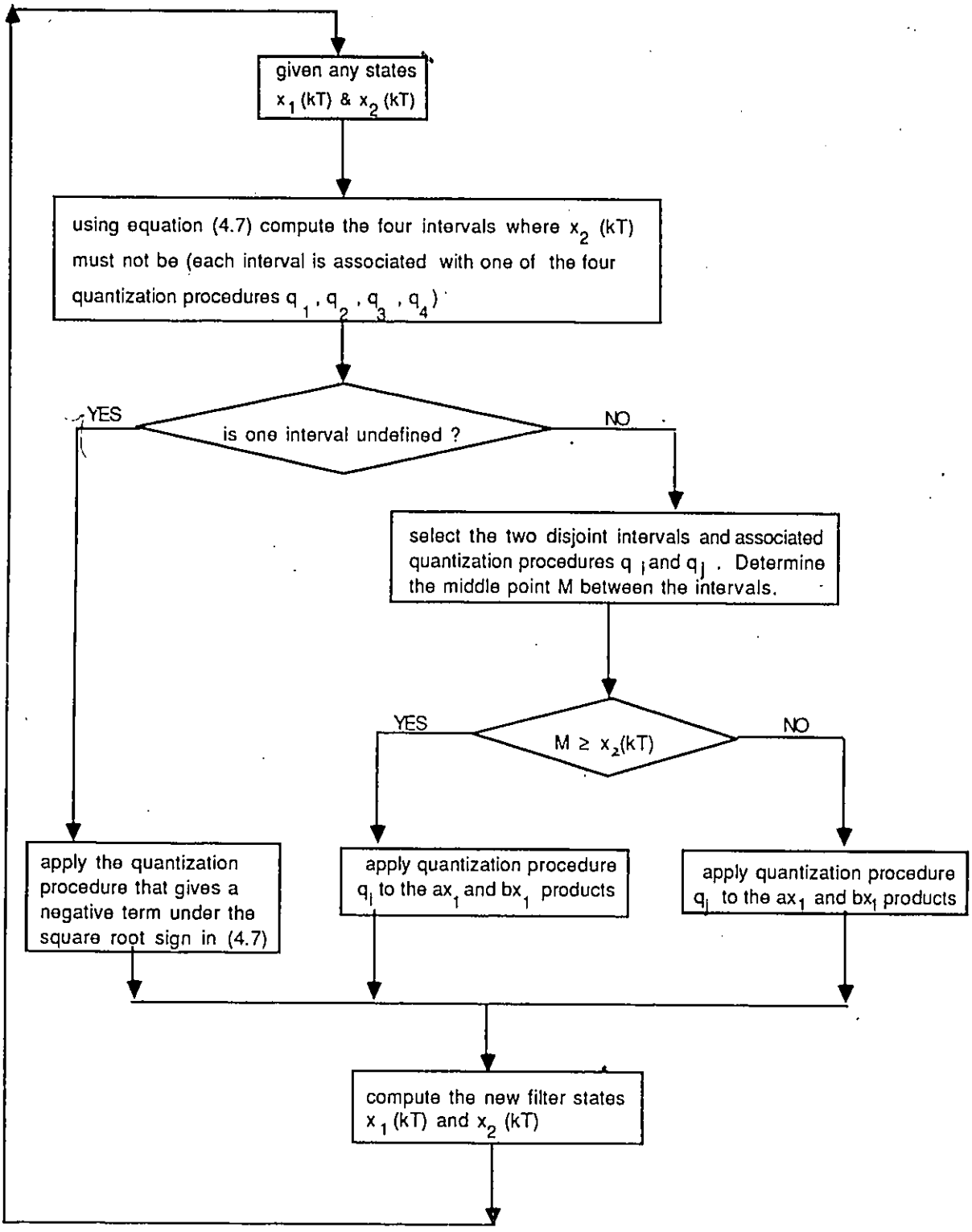


Figure 4.3 Flow Chart for Limit Cycle Suppression

4.4 SIMULATION RESULTS

Extensive computer simulation had been carried out to test the effectiveness of the limit cycle suppression algorithm. The computer source program used for this simulation is listed in the appendix. It was assumed in the program that fixed binary point sign magnitude arithmetic is used for the filter multipliers and that the signal-coefficient products are quantized to integers. A wide variety of values for the filter coefficients and initial conditions were used with emphasis on those coefficients corresponding to high-Q filters since they often have the worst limit cycle oscillations in the sense of large amplitudes and that other suppression methods have failed to eliminate them. In every case the filter states converge to zero within a reasonable number of iterations. The number of iterations required is dependent on the value of the coefficients as well as the initial filter states. In general, the larger the initial condition, the longer it takes for the filter to converge to zero, which is to be expected. As an example, for the filter with coefficients $a = -1.84375$ and $b = 0.9375$, with initial conditions $x_1 = 8$ and $x_2 = 8$ it takes 36 iterations for the filter to reach the zero state; but if the initial conditions are changed to $x_1 = 16$ and $x_2 = 16$ then it takes 55 iterations. Figure 4.4 is a sample of the simulation outputs. For an analysis of the limit cycle

behaviour of some of the coefficient pairs see Reference [6] and [19].

a=1.84375 b=0.9375 x1(0)=9 x2(0)=9	-8 7 -6 5 -4 3 -2 1 0 0
a=-1.84375 b=0.9375 x1(0)=8 x2(0)=8	23 34 40 42 40 34 25 15 4 -7 -16 -22 -25 -26 -24 -21 -16 -10 -3 4 9 13 15 15 14 12 9 6 3 -2 -3 -3 -2 -1 0 0
a=1.92 b=0.98876 x1(0)=30 x2(0)=30	-28 25 -21 16 -10 4 2 -6 9 -11 12 -12 11 -10 9 -8 7 -6 5 -4 3 -2 1 0 0
a=1.85 b=0.95 x1(0)=20 x2(0)=20	-17 13 -8 3 2 -5 7 -8 8 -7 6 -5 4 -3 2 -1 0 0
a=1.6 b=0.97 x1(0)=34 x2(0)=34	-20 -1 20 -31 29 -16 -3 19 -28 27 -16 -1 16 -24 23 -13 -2 15 -22 20 -11 -2 13 -18 16 -8 -3 11 -15 14 -8 -1 8 -11 10 -6 0 5 -8 7 -5 2 1 -2 2 -1 0 0
a=-0.8 b=0.94 x1(0)=17 x2(0)=17	31 9 -23 -27 0 26 21 -8 -26 -14 14 24 7 -17 -20 0 19 15 -6 -19 -10 10 17 5 -12 -14 0 13 10 -4 -12 -7 6 11 4 -7 -8 0 7 5 -2 -5 -3 2 3 1 -1 -1 0 0
a=1.76 b=0.95 x1(0)=20 x2(0)=20	-15 7 2 -9 14 -16 14 -10 4 2 -6 8 -9 8 -6 3 0 -2 3 -3 2 -1 0 0
a=-1.8 b=0.937 x1(0)=16 x2(0)=16	44 66 77 77 66 46 21 -6 -31 -50 -60 -61 -53 -37 -17 4 23 37 46 48 42 31 16 -1 -15 -26 -33 -34 -30 -22 -12 -1 10 18 22 23 20 15 8 0 -7 -12 -16 -16 -14 -11 -7 -2 3 6 8 8 7 5 3 1 -1 -1 0 0
a=-1.92 b=0.98876 x1(0)=30 x2(0)=30	-28 25 -21 16 -10 4 2 -6 9 -11 12 -12 11 -10 9 -8 7 -6 5 -4 3 -2 1 0 0

Figure 4.4 Samples of Simulation Output

4.5 LIMIT CYCLE SUPPRESSION IN SPDF

The Stored-Product Digital Filter (SPDF) architecture has been proposed in [25] and [26]. It is a digital filter implementation technique whereby Read-Only-Memories (ROMs) are used in place of binary multipliers. The basic idea is to store the full signal-coefficient products of the filter into ROMs, with each product addressed by the corresponding signal which is its multiplicand. The ROMs are organized into independently addressable blocks, one for each distinct coefficient. The products can thus be accessed in parallel and then added to produce the output sample value. It has been shown that the SPDF architecture can achieve very high speed in parallel processing [27]. Another advantage of this technique is the high accuracy of the filter transfer function as there is virtually no coefficient quantization effect in the ROM implementation. Figure 4.5 illustrates the SPDF for a second order recursive filter section with the signal wordlength equals to L_s bits.

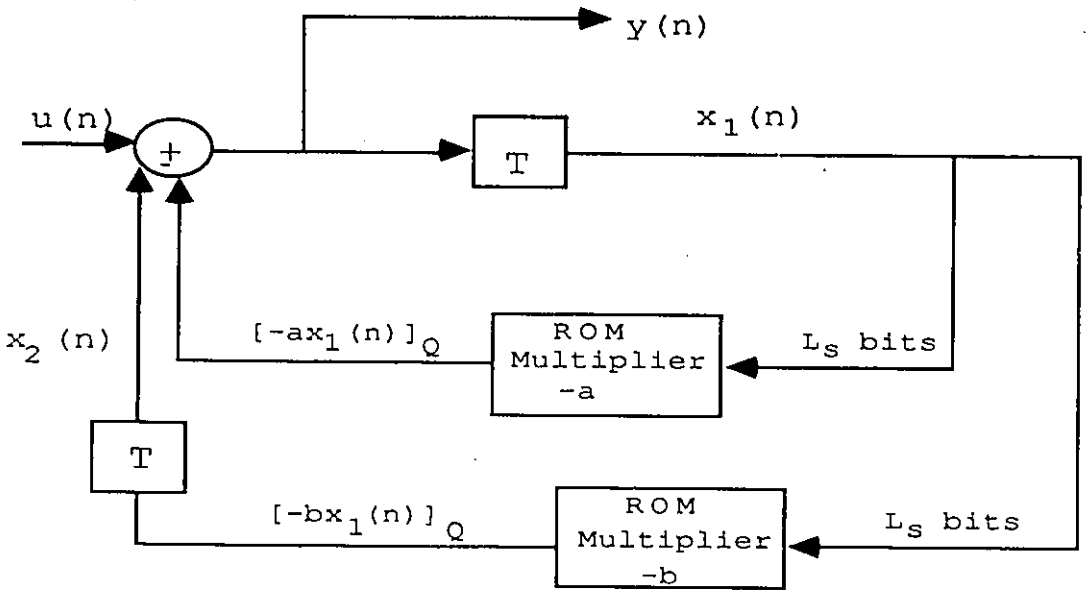


Figure 4.5 SPDF Implementation of the Second Order Recursive Section

To implement the limit cycle suppression algorithm in the SPDF some additional hardware is required. In the normal realization of a SPDF, all possible products of an N-bit signal with a given coefficient, quantized in an arbitrary manner (for example roundoff), are stored in ROM, replacing the multiplier. In order to incorporate the limit cycle suppression algorithm two sets of products, one using MT quantization, the other using ME quantization, must be stored for each coefficient. The values of M (the mid-point between disjoint x_2 intervals) for the filter section are computed and stored in a separate ROM (size $2^{Ls} \times L_s+1$ bits). Memory locations in this ROM is addressed by the signal

$x_1(kT)$ to generate the corresponding M value before the ROM multipliers are accessed. The value of M and the signal $x_2(kT)$ are inputs to a comparator. The output of the comparator, together with the state signals will access the products with the desired quantization. The modifications to the basic SPDF structure are shown in Figure 4.6.

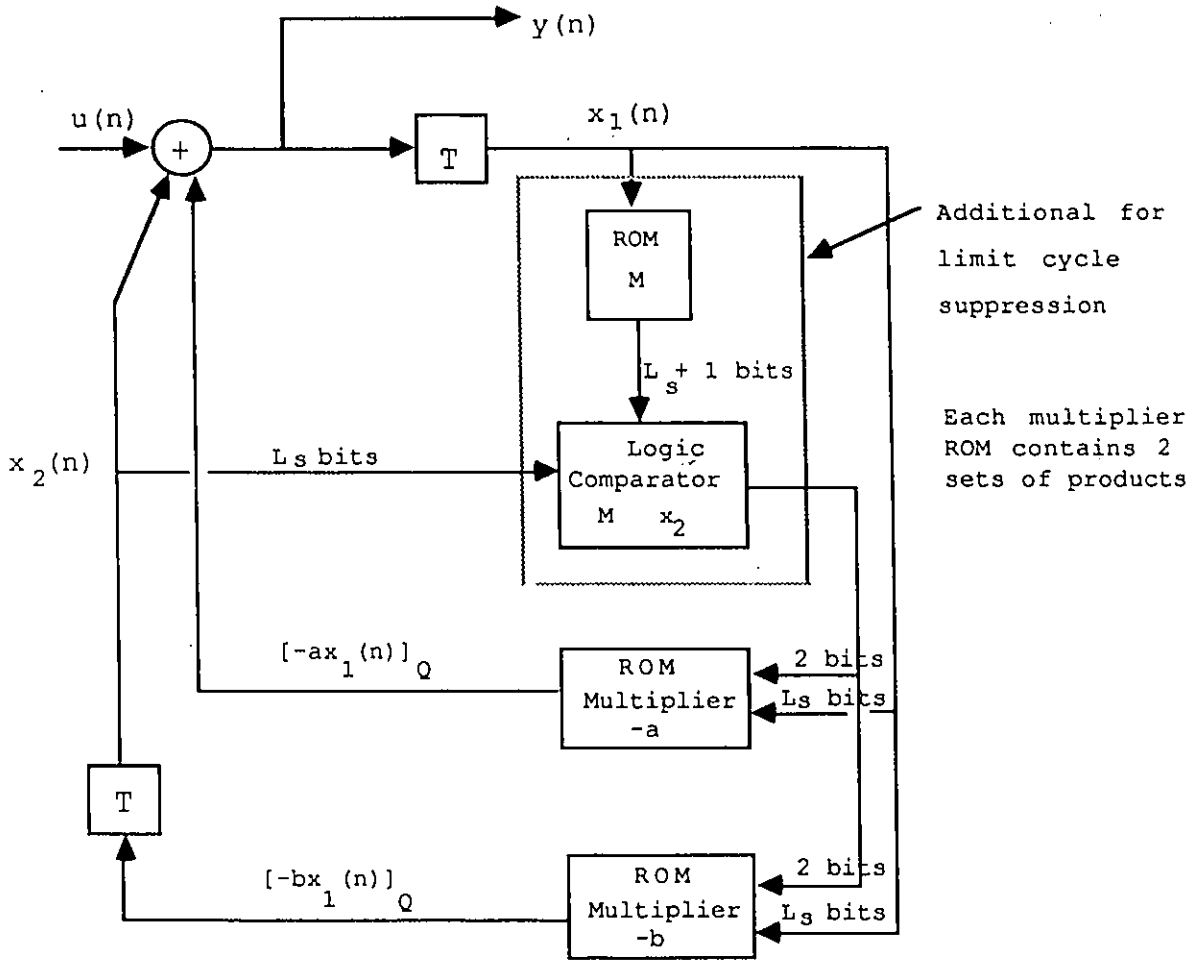


Figure 4.6 SPDF Structure with Limit Cycle Suppression

CHAPTER 5 QUANTIZATION NOISE EVALUATION

5.1 INTRODUCTION

The effectiveness of the proposed control quantization (CQ) scheme in suppressing zero input limit cycles has been demonstrated in the preceeding chapter. Another important consideration in evaluating the performance of this new scheme is its quantization noise behaviour during normal filter operation. In general, filter implementations with superior stability with respect to limit cycles also tend to have higher quantization noise [5]. For example, a filter using magnitude truncation, which eliminates limit cycles of period longer than two has a 6 db higher quantization noise power in the output than one using roundoff [12].

This chapter will investigate the noise behaviour of the proposed CQ scheme when driven by a white Gaussian input, by comparing it to that of the CQ method developed by Lawrence and Mitra [21].

5.2 EXISTING CONTROLLED QUANTIZATION METHOD

The schematic of a second order digital filter employing the existing controlled quantization (CQ) method is shown in Figure 5.1. The concept of controlled rounding was first introduced by H. J. Butterweck [20]. The development of his quantization criterion is based on the analysis of an "energy function" for the filter given by:

$$W = [x_{n-1},\ x_n] \begin{bmatrix} 1-b & -a \\ -a & 1-b \end{bmatrix} \begin{bmatrix} x_{n-1} \\ x_n \end{bmatrix}$$

$$= (1-b)\ [x^2_{n-1} + x^2_n] - 2ax_{n-1}x_n \qquad (5.1)$$

From (5.1) the following relation is obtained for an unforced filter:

$$W_n - W_{n+1} = (1+b)\,[x_{n+1}-x_{n-1}]^2 - 2e_{n+1}[x_{n+1}-x_{n-1}] \qquad (5.2)$$

In Butterweck's terminology, equation (5.2) represents the filter energy decrease per time step. In order to eliminate limit cycles $[W_n - W_{n+1}]$ must be positive for every time instant t_n. Since the first term in (5.2) is positive it is only necessary to ensure that the second term is either negative or

zero so that when combined with the first term the equation remains positive. This is achieved by choosing the appropriate quantization for the signal x_{n+1} such that the sign of the error signal e_{n+1} is opposite to that of $(x_{n+1} - x_{n-1})$. The manner of quantization is then controlled by the internal states of the filter since $x_{n+1} = -ax_n - bx_{n-1}$. Difficulties arise when $x_{n+1} = x_{n-1}$, since in this case (5.2) will have a value of zero and the absence of limit cycles cannot be guaranteed. In fact, limit cycles of periods 1 and 2 may be sustained under this quantization scheme.

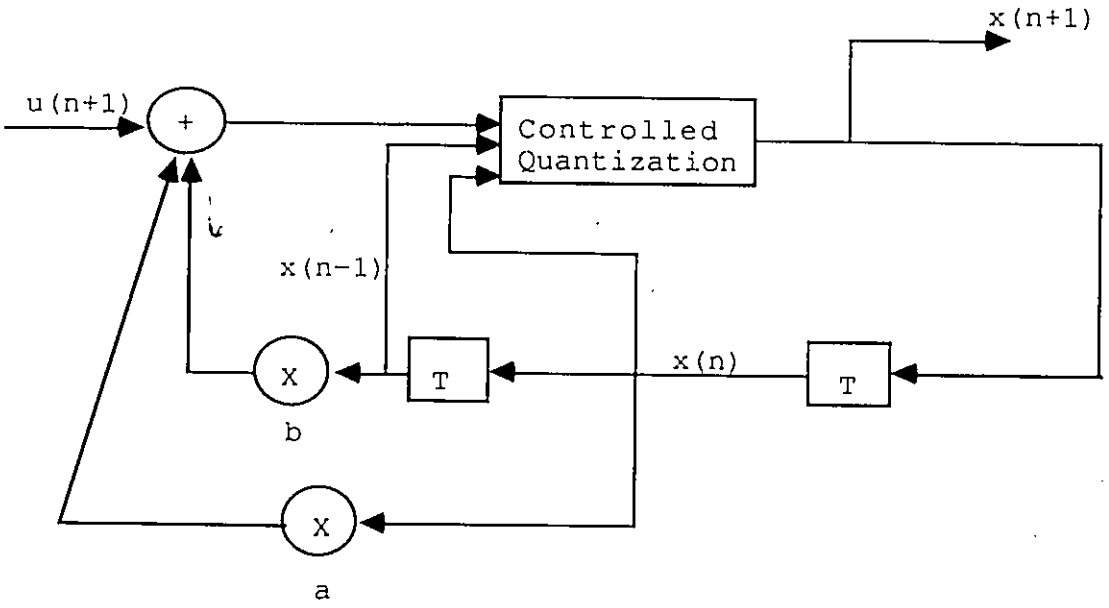


Figure 5.1 Schematic of Second Order Filter Section with Existing CQ

Lawrence and Mitra [21] have adopted Butterweck's "energy function", noting that W(n) is a positive definite quadratic function of the filter state vector but does not possess any unique characteristic which qualifies it as the energy function. In Lawrence and Mitra's scheme, the stable coefficient plane is divided into three sub-regions as shown in Figure 5.2.

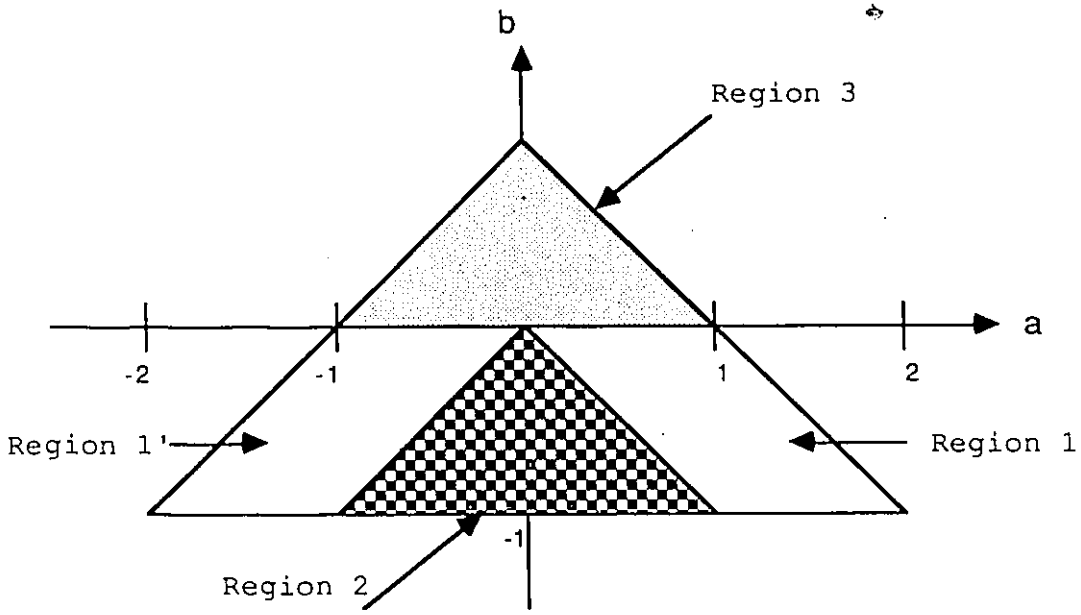


Figure 5.2 Coefficient Plane Sub-regions in Existing CQ

The quantization rules for each region is as follows:

Region 1 $e_{n+1}(x_{n+1} - x_{n-1}) \leq 0$ if $x_n \neq x_{n-1}$
$x_n \neq -x_{n-1}$ for Region 1'

> 0 if $x_n = x_{n-1}$
$x_n = -x_{n-1}$ for Region 1'

Region 2 $e_{n+1}(x_{n+1} - x_{n-1}) \leq 0$

Region 3 use sign-magnitude truncation

The quantization rule used in Region 2 is the same as that used by Butterweck. The Region 3 arithmetic is the same as that used in Reference [13]. The major innovation in Lawrence and Mitra's scheme is in the arithmetic of Region 1 and 1'. These are the regions where Butterweck's quantization rule may generate large amplitude limit cycles of period 1 and 2. Results in [21] show that all zero-input limit cycles are suppressed. It is also shown that under certain restrictions some limit cycles assoicated with D.C. or symmetric alternating inputs are suppressed.

There are two notable differences between the existing controlled quantization methods and the one proposed in this thesis. The first is the nature of the control signal. In the existing methods the control signal is a linear combination of the filter states and can be obtained directly from the filter. In the proposed scheme the control signals are precomputed and accessed from ROM during filter operation. The second difference

is the placement of the quantizer in the filter. In the existing method one quantizer is used and quantization takes place after the addition of the product signals. This placement of the quantizer requires the use of one double precision adder. In the proposed scheme quantization takes place after each multiplication and does not require any double precision adder.

5.3 QUANTIZATION NOISE COMPARISON

The noise effect of a quantizer can be studied by theoretical analysis and modelling. For example, the error of a RO quantizer can generally be modelled with sufficient accuracy as an additive white noise in the filter output with an uniform probability density distribution. Such a quantization model provides a relatively easy means to analyse and predict the noise behaviour of the filter.

Two quantization noise models have been developed for the type of controlled quantization which utilizes a linear combination of the filter states as the control signal [28]. It has been shown that CQ of this type is very similar to MT with regard to quantization errors. Specifically, it is found that the total quantization noise generated by a filter using the existing CQ method is almost equal to that generated by one using

a single MT quantizer. Both noise samples contain parts which are correlated to the processed signal. The correlation is higher in the case of the CQ quantizer and especially for high-Q filters.

Because of the complex nature of the control signal in the proposed scheme, a theoretical analysis of the quantization noise has not been attempted. As an alternative, computer simulation of a second order digital filter utilizing the proposed quantizer and of the associated ideal filter is carried out. To determine the amount of noise generated by the quantizer the output of the non-linear filter is subtracted from that of the ideal filter. The schematic for this process is shown in Figure 5.3. The input to both filters is a white Gaussian process. This simulation is repeated for an extensive set of filter coefficient pairs. The quantization noise generated by Lawrence and Mitra's CQ scheme is determined in a similar manner. It is found that in the latter case the simulation results are very close to those predicted by the quantization noise models given in [28].

The comparison of the simulation results are shown in the following graphs. In the graphs, the power of the quantization error in the filter output is plotted as a function of the filter coefficient b for various values of the coefficient a. It is found that the quantization noise levels are very similar for the same value of coefficient a regardless of its sign. This is true

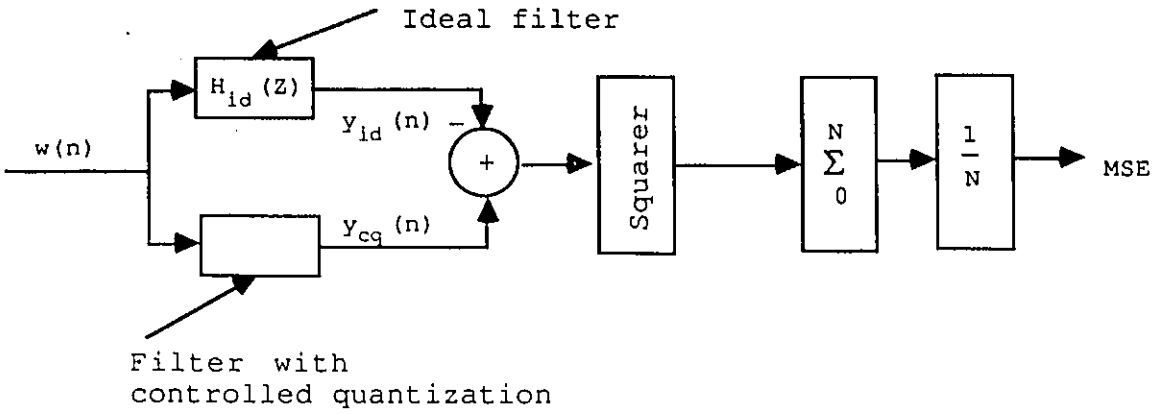
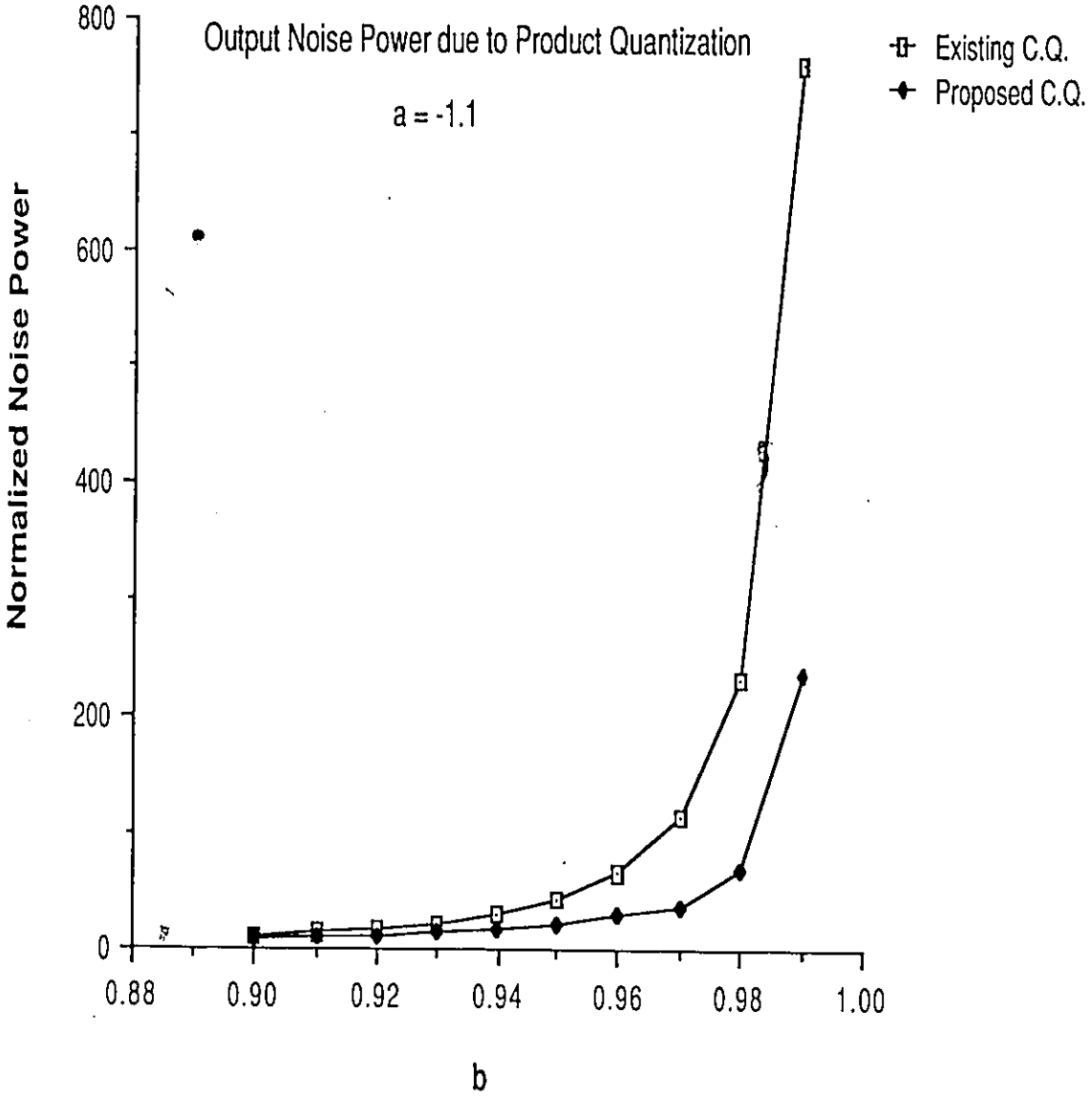
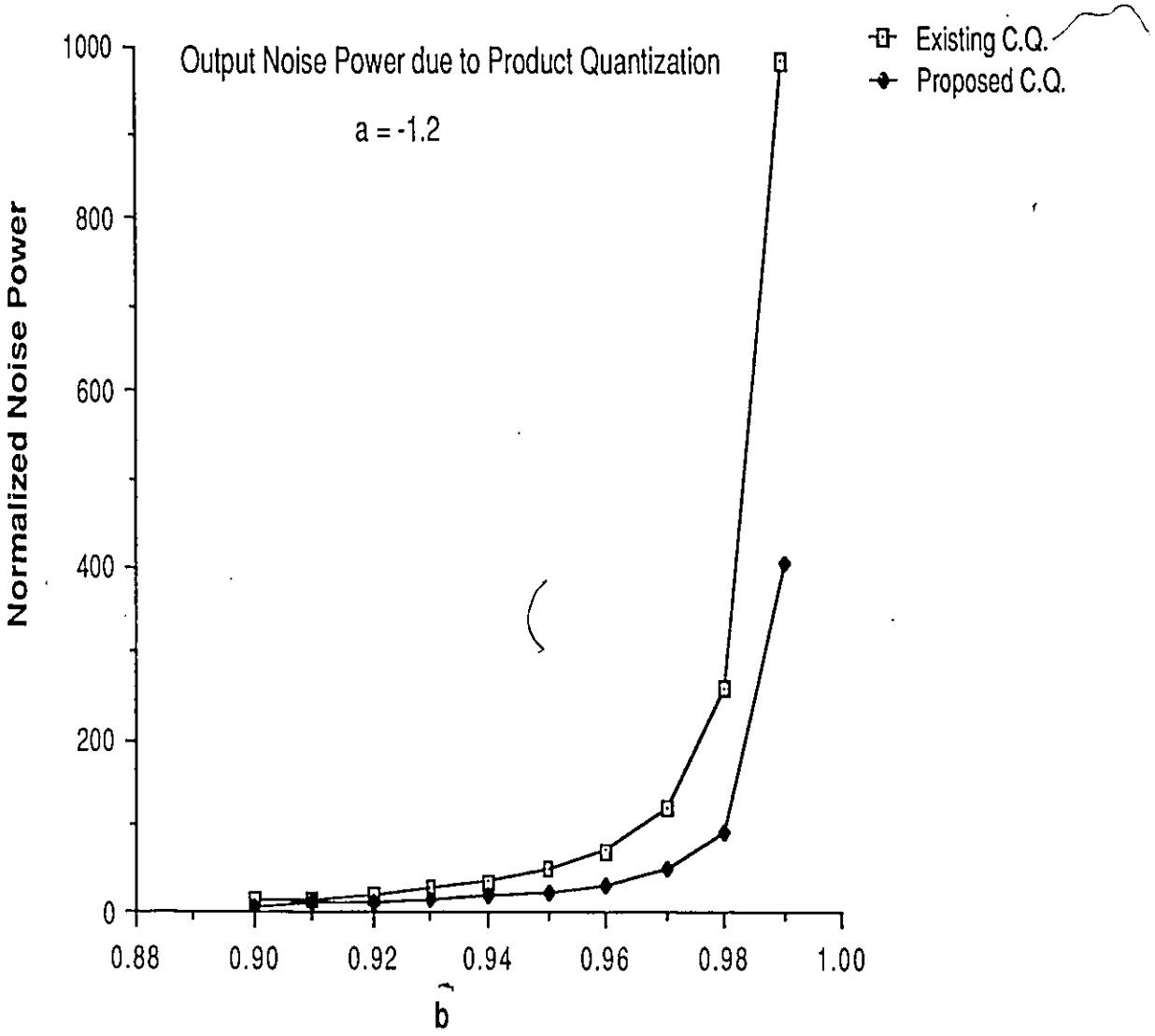
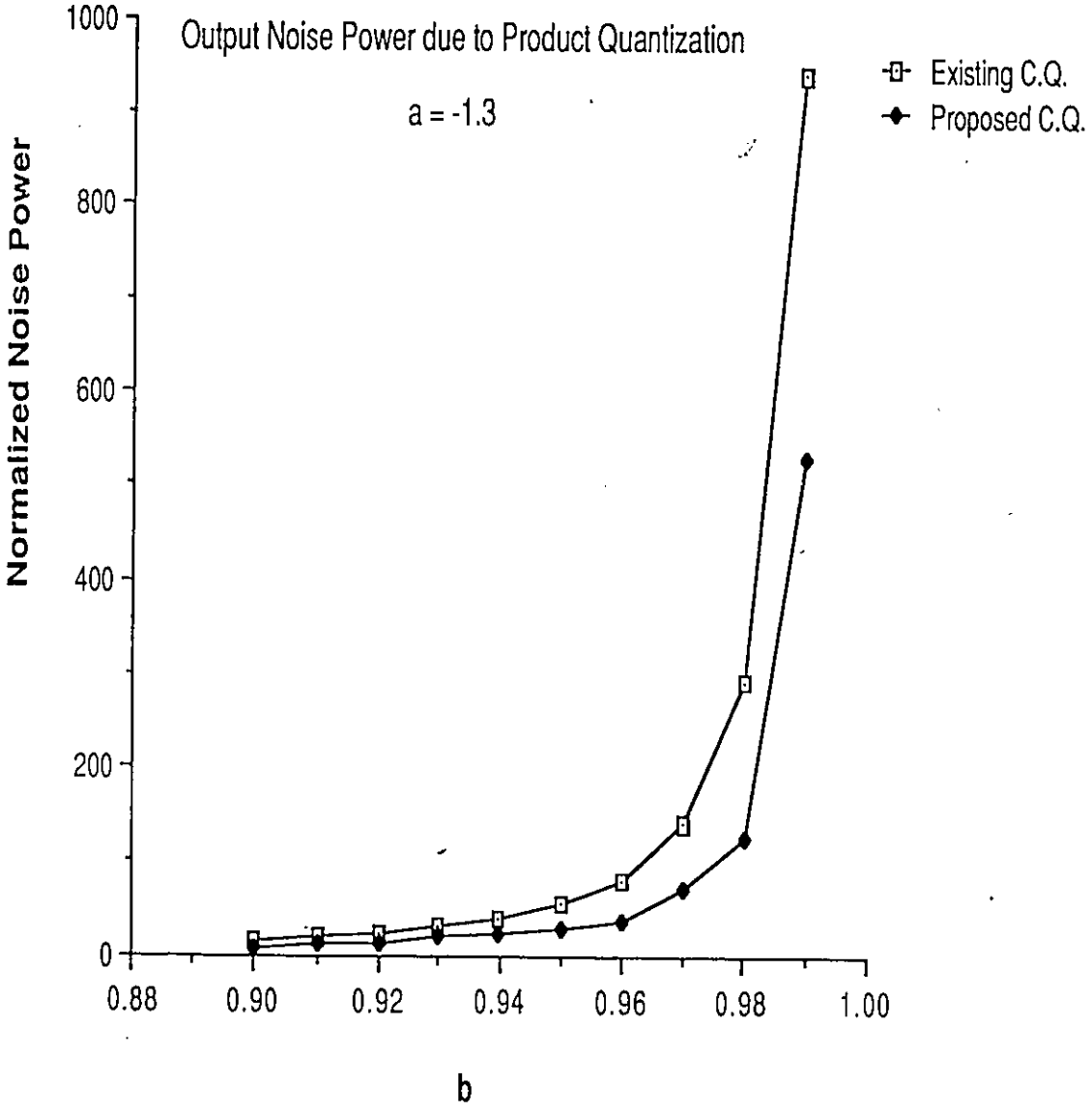


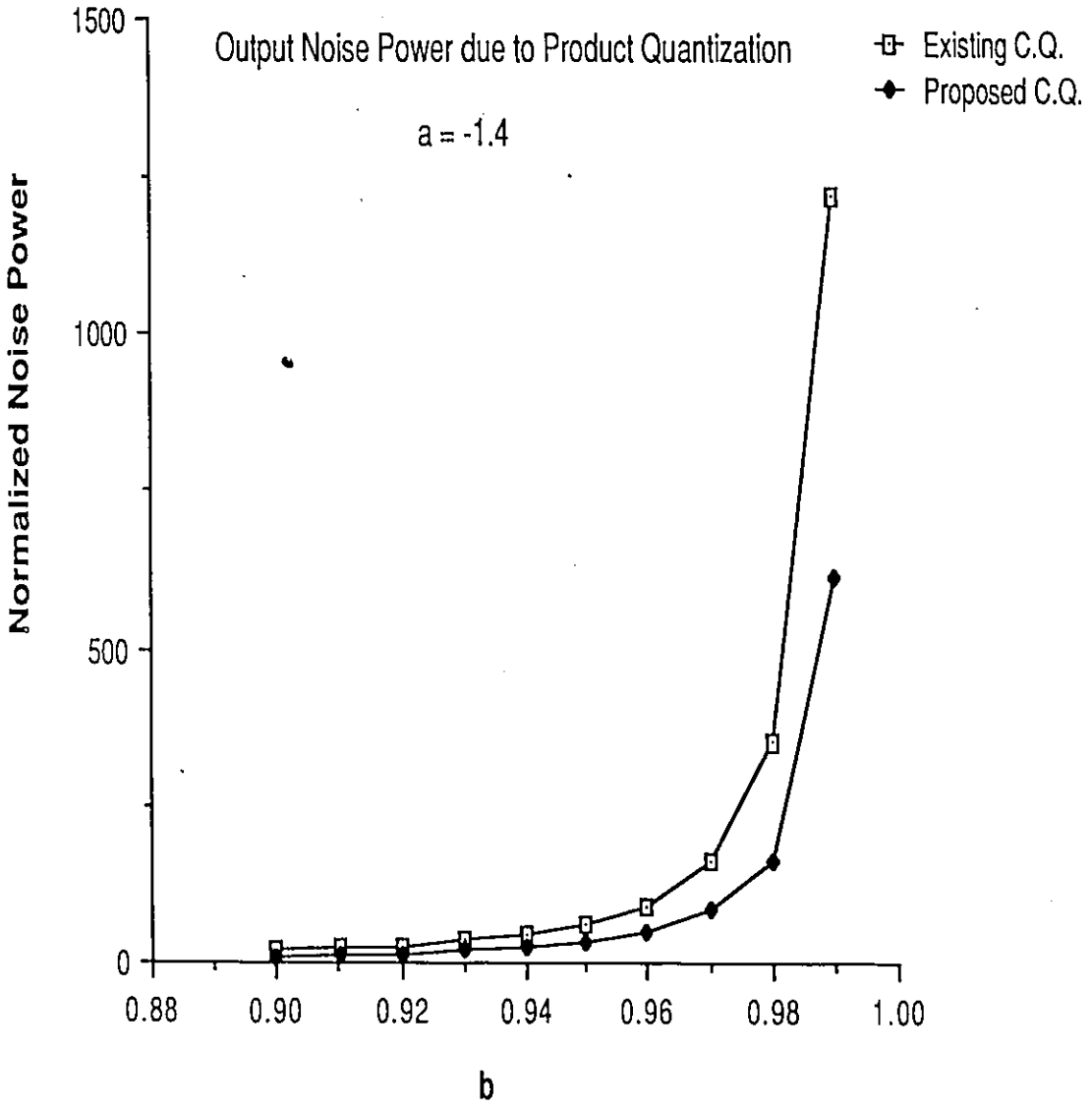
Figure 5.3 Schematic for Measuring Quantization Error Power

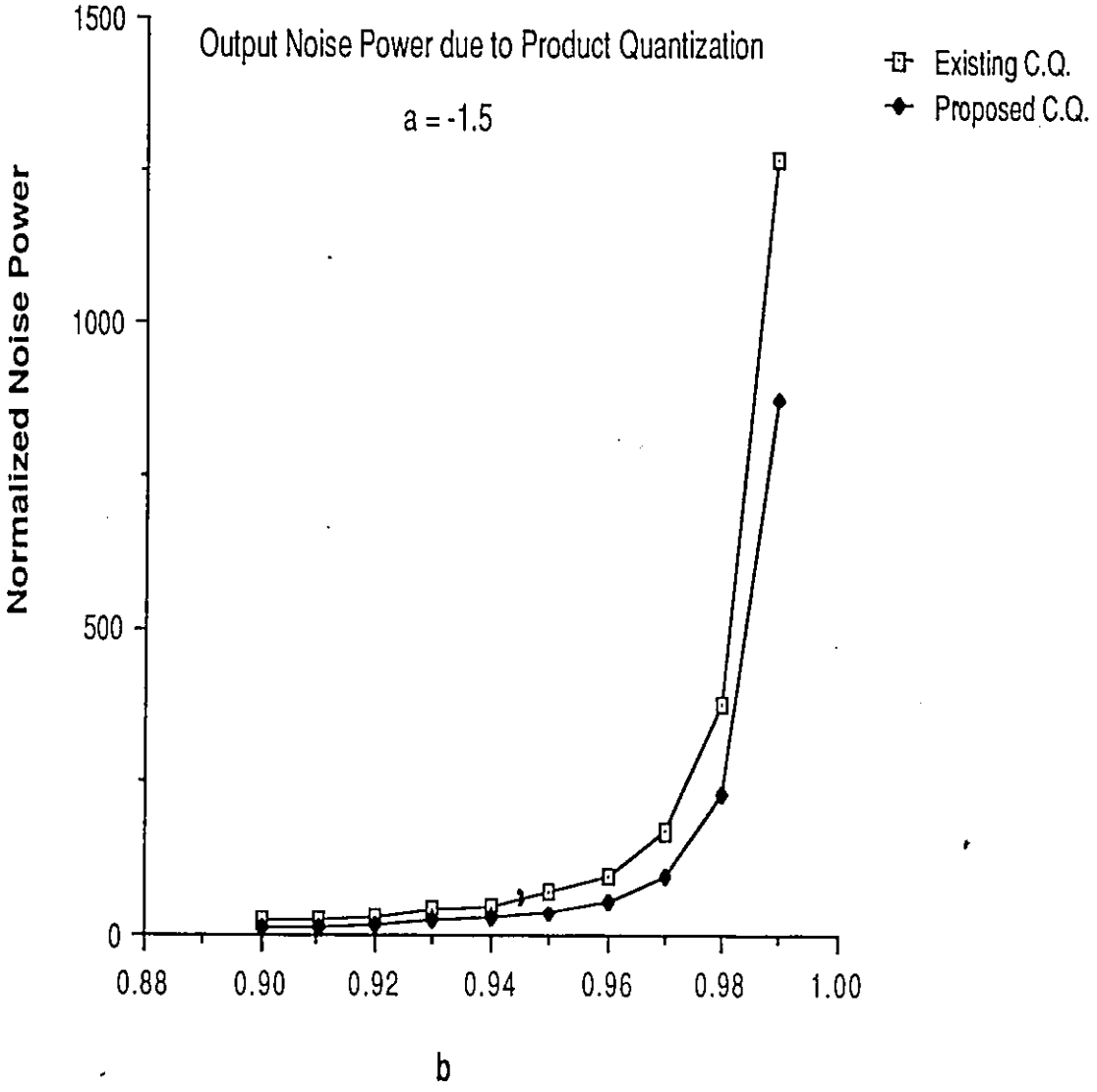
for both of the CQ methods. Negative 'a' values have been used to plot the graphs shown; the results for positive 'a' values are similar. As well, the values of coefficient b chosen for the graphs are in the range of (0.90 0.99) because these are the values which generate the largest quantization noise and the noise reduction by the proposed method is more pronounced. The actual simulation covers the following coefficient values: 'a' was varied from ±1.1 to ±1.9 in increments of 0.1; 'b' was varied from ±0.5 (since there is no limit cycles for $|b| < 0.5$) to ±0.9 in increments of 0.1, and from ±0.91 to ±0.99 in increments of 0.01. In every instance the quantization noise generated by the existing controlled quantization is sigificantly higher than the proposed method.

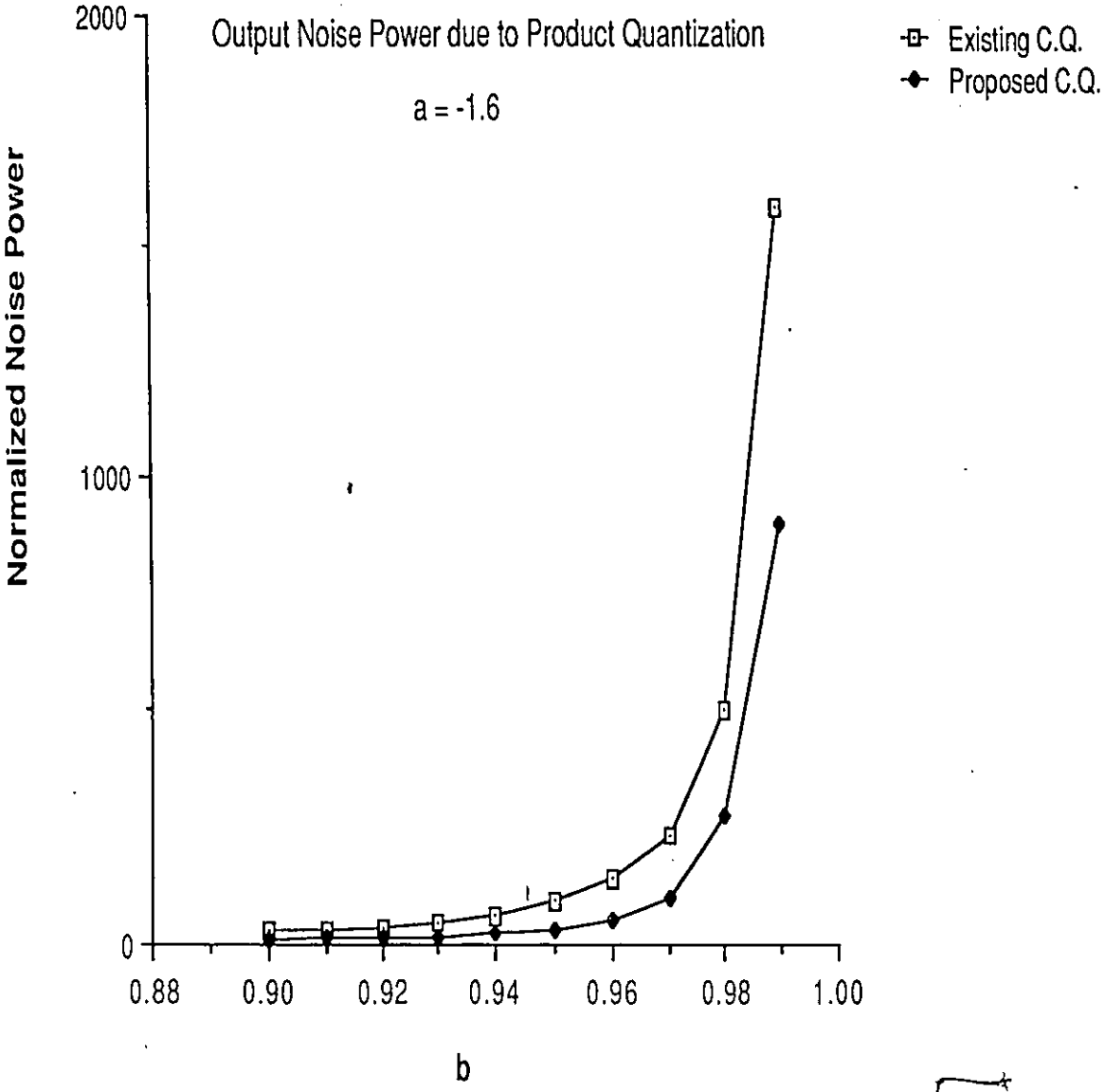

Output Noise Power due to Product Quantization
a = -1.1
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
800
600
400
200
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b

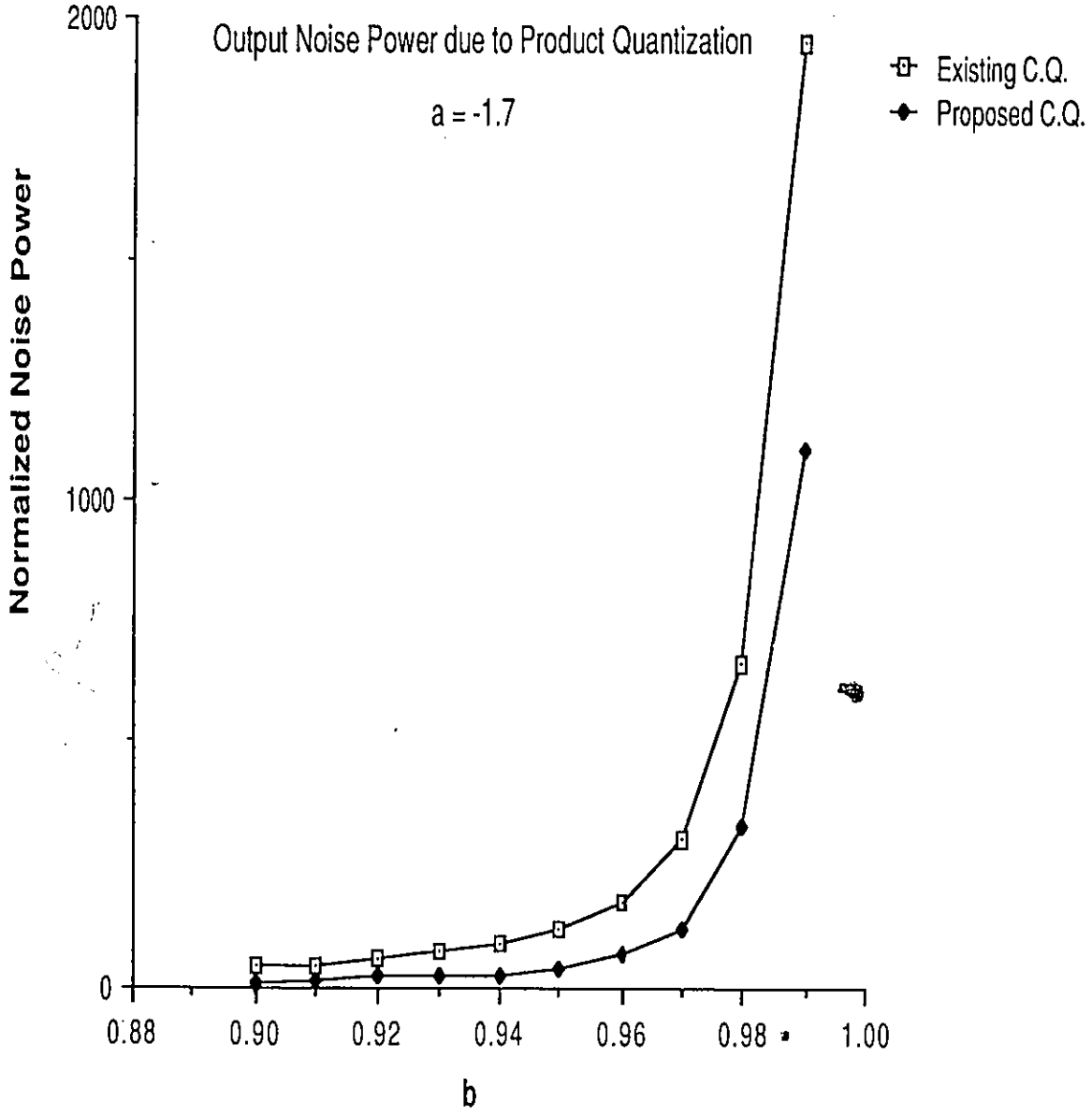

Output Noise Power due to Product Quantization
a = -1.2
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
1000
800
600
400
200
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b̂

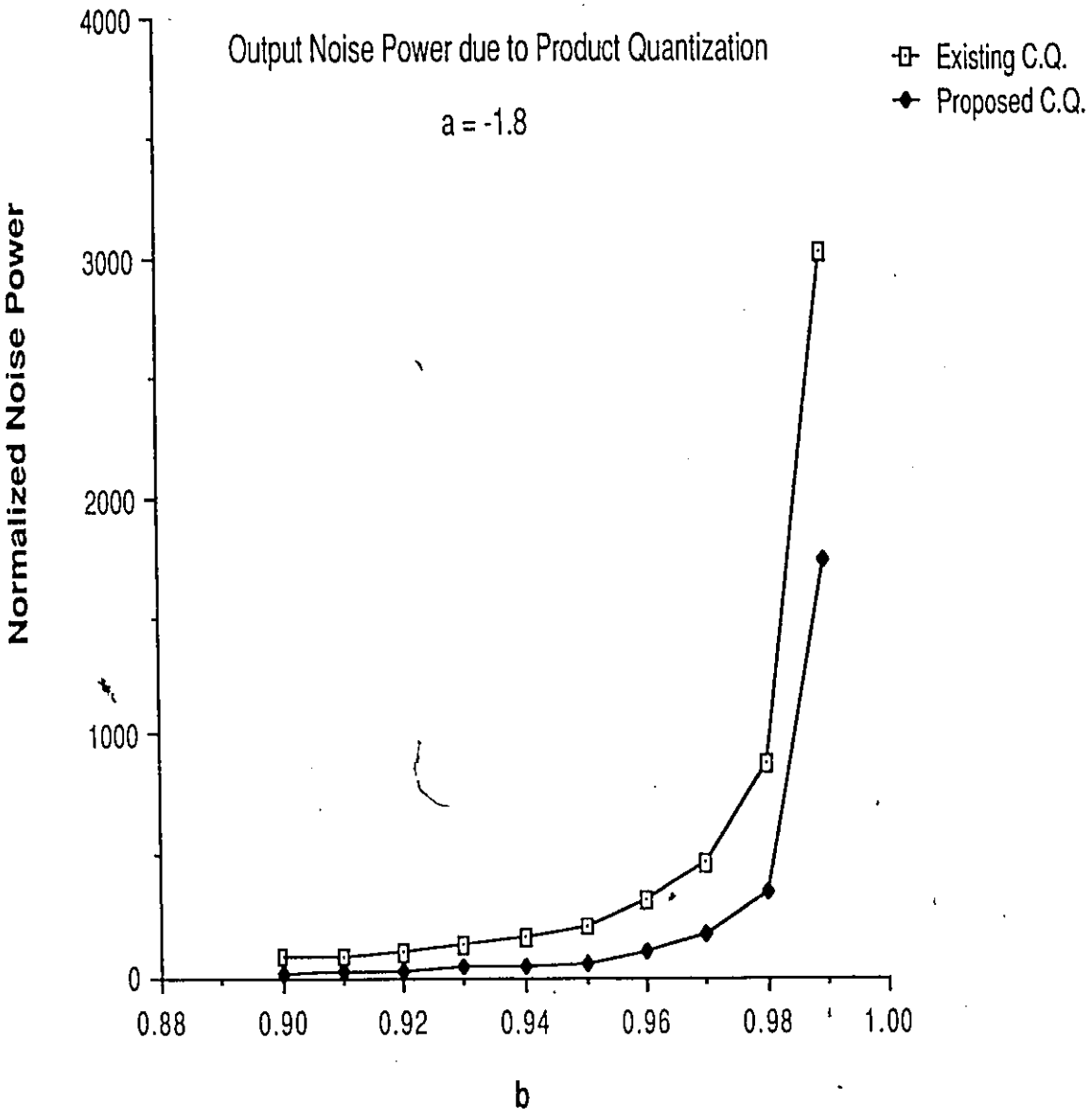

Output Noise Power due to Product Quantization
a = -1.3
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
1000
800
600
400
200
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b

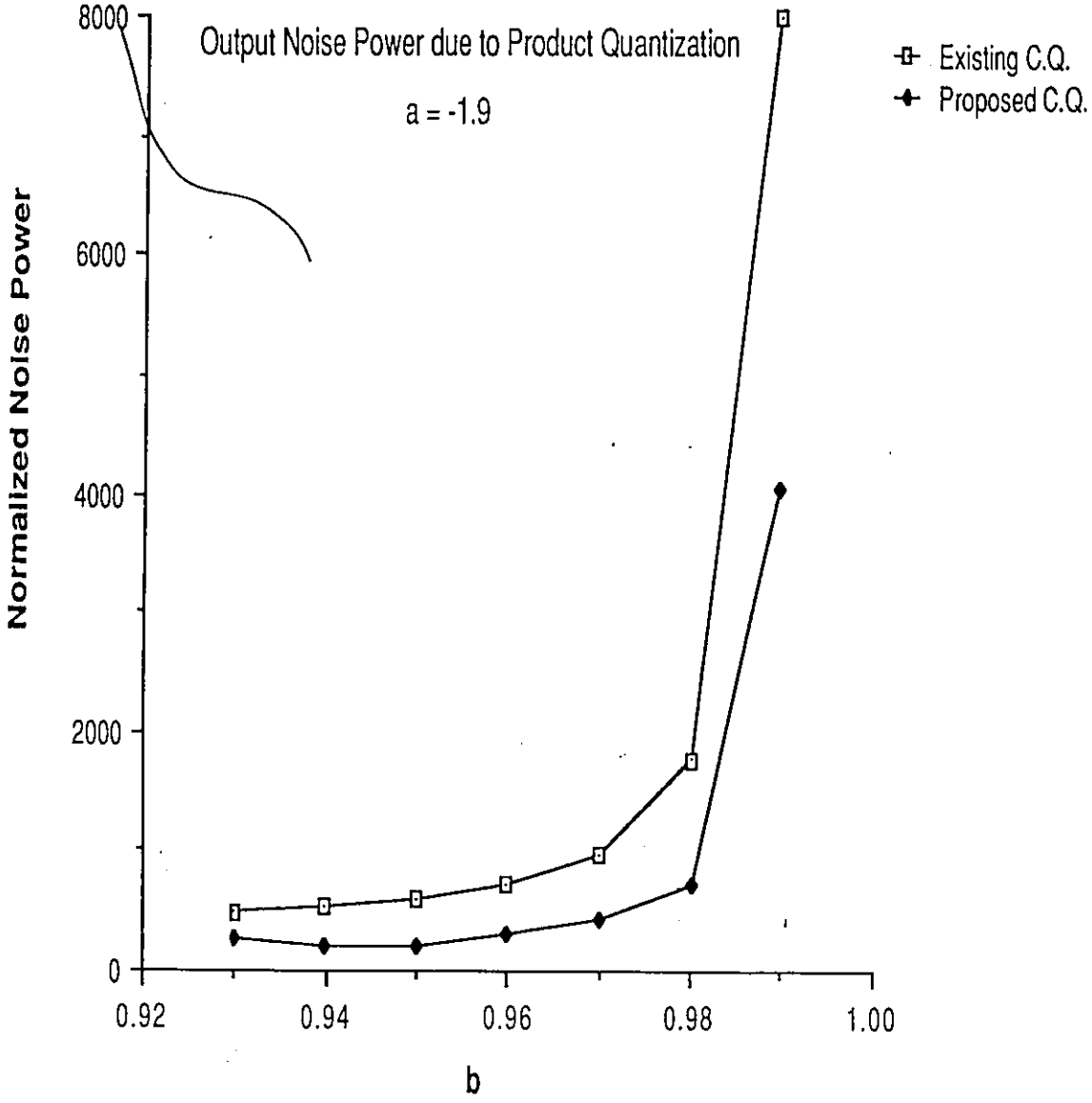

Output Noise Power due to Product Quantization
a = -1.4
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
1500
1000
500
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b


Output Noise Power due to Product Quantization
a = -1.5
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
1500
1000
500
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b


Output Noise Power due to Product Quantization
a = -1.6
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
2000
1000
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b


Output Noise Power due to Product Quantization
a = -1.7
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
2000
1000
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b


Output Noise Power due to Product Quantization
a = -1.8
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
4000
3000
2000
1000
0
0.88
0.90
0.92
0.94
0.96
0.98
1.00
b


Output Noise Power due to Product Quantization
a = -1.9
Existing C.Q.
Proposed C.Q.
Normalized Noise Power
8000
6000
4000
2000
0
0.92
0.94
0.96
0.98
1.00
b

CHAPTER 6 CONCLUSION

The Lyapunov Stability Theorem has been applied to the analysis of a second order digital filter's stability requirement. From this analysis, a limit cycle free criterion for the filter was derived. The applicability of this criterion was confirmed by a proof which showed that it can be satisfied within the quantization environment of the filter. The specific quantization requirement is that either magnitude truncation or magnitude enhancement must be used so that the absolute value of the resultant quantization error is in the range of [0,1).

The limit cycle free criterion was translated into a controlled quantization algorithm whereby all zero input limit cycles in the filter output can be suppressed. Computer simulation of this algorithm, utilizing a large variety of filter coefficients and initial conditions has confirmed its effectiveness.

The implementation of the limit cycle suppression algorithm in the Stored Product Digital Filter was proposed. It was shown that the additional hardware required was very simple. Further study can be in the area of finding alternative implementations.

For example, instead of storing two sets of products in the ROMs, only the magnitude enhanced products are stored, and depending on the outcome of the comparator the signal-coefficient product may be truncated outside of the ROM before the addition operation which produces the filter output.

A survey of the existing limit cycle suppression methods shows that some are not effective for the entire filter coefficient plane, and the ones which are applicable to all coefficients tend to generate large quantization noise. The quantization noise power generated by the proposed controlled quantization scheme while driven by a white Gaussian input was shown to be considerably lower than that generated by the existing controlled quantization method. The complete avoidance of limit cycles and the superior quantization noise behaviour should make the proposed method a very attractive alternative.

REFERENCES:

1. Classen, T.A.C.M.; Mecklenbrauker, W.F.G.; Peek, J.B.H.; "Effects of Quantization and Overflow in Recursive Digital Filters". *Trans. Acoustics, Speech and Signal Processing* vol. ASSP-24, December 1976, pp.517-527.

2. Ebert, P.M.; Mazo, J.E.; Taylor, M.G.; "Overflow Oscillations in Digital Filters". *BSJT* vol. 48, November 1969, pp.2994-3020.

3. Kaiser, J.F.; "On the Limit Cycle Problem", *IEEE Proc. ISCAS 1976*, pp.642-645.

4. Willson, Jr. A.N.; "A Stability Criterion for Nonautonomous Difference Equations with Application to the Design of a Digital FSK Oscillator". *IEEE Trans. Circuits and System* vol. CAS-21, January 1974, pp.124-130.

5. Claasen, T.A.C.M.; Mecklenbrauker, W.F.G.; Peek, T.B.H.; "A Survey of Quantization and Overflow Effects in Recursive Digital Filters". *IEEE Proc. ISCAS 1976*, pp.621-624.

6. Parker, S.R.; Hess, S.F.; "Limit Cycle Oscillations in Digital Filters". IEEE Trans. Circuit Theory vol. CT-18, November 1971, pp.687-697.

7. Claasen, T.A.C.M.; Mecklenbrauker, W.F.G.; Peek, T.B.H.; "Some Remarks on the Classification of Limit Cycles in Digital Filters". *Philips Res. Rep.* vol. 28, August 1973, pp.297-305.

8. Kieburtz, R.B.; "An Experimental Study of Roundoff Effects in a Tenth-Order Recursive Digital Filter". *IEEE Trans. Communication* vol. COM-21, June 1973, pp.757-763.

9. Blackman, R.B.; Linear Data-Smoothing and Prediction in Theory and Practice. Addition-Wesley Company, 1965.

10. Jackson, L.B.; "An Analysis of Limit Cycles Due to Multiplication Rounding in Recursive Digital (Sub)filters". *Proc. of Seventh Annual Allerton Conference on Circuit and System Theory*, Monticello Illinois, October 1969, pp.69-78.

11. Kaiser, J.F.; "Some Practical Considerations in the Realization of Linear Digital Filters'. *Proc. of Third Allerton Annual Conference on Circuit and System Theory*, 1965.

12. Kao, C.; "An Analysis of Limit Cycles Due to Sign-Magnitude Truncation in Multiplication in Recursive Digital Filters". *Proc. of Fifth Asimolar Conference on Circuit and System Theory*, Pacific Grove, 1971, pp.349-353.

13. Claasen, T.A.C.M.; Mecklenbrauker, W.F.G.; Peek, J.B.H.; "Second-Order Digital Filter with Only One Magnitude-Truncation Quantizer and Having Practically No Limit Cycle'. *Electron Letter* vol. 9, November 1973, pp.531-532.

14. Long, J.L.; Trick, T.N.; "A Note on Absolute Bounds on Limit Cycles Due to Roundoff Errors in Digital Filters". *IEEE Trans. Audio Electroacoustic* vol. AU-21, Feb. 1973, pp.27-30.

15. Sandberg, I.W.; Kaiser, J.F. "A Bound on Limit Cycles in Fixed-Point Implementations of Digital Filters". *IEEE Trans. Audio Electroacoustic* vol. AU-20, June 1972, pp.110-112.

16. Fettweis, A.; Meerkotter, K.; "Suppression of Parasitic Oscillations in Wave Digital Filters". *IEEE Trans. on Circuits and System* vol. CAS-22, March 1975, pp.239-246.

17. Meerkotter, K.; Wegener, W.; "A New Second-Order Digital Filter Without Parasitic Oscillations:. *AEU* 29, July/Aug. 1975, pp.312-314.

18. Buttner, M.; "Elimination of Limit Cycles in Digital Filters with Very Low Increase in the Quantization Noise". *IEEE Trans. on Circuits and System* vol. CAS-24, June 1977, pp.300-304.

19. Kieburtz, R.B; Lawrence, V.B.; Mina, K.V.; "Control of Limit Cycles in Recursive Digital Filters By Randomized Quantization". *IEEE Trans. on Circuits and Systems* vol. CAS-24, June 1977, pp.300-304.

20. Butterweck, H.J.; "Suppression of Parasitic Oscillation in Second-Order Digital Filters By Means of A Controlled-Rounding Arithmetic'. *AEU*, DEC 1975, pp.371-374.

21. Mitra, D.; Lawrence, V.B.; "Controlled Rounding Arithmetics for Second-Order Direct-Form Digital Filters that Eliminate All Self-Sustained Oscillations". *IEEE Trans. on Circuits and Systems* vol. CAS-28, September 1981, pp.894-902.

22. Kalman, R.E.; Bertram, J.E.; "Control System Analysis and Design Via the Second Method of Lyapunov -- Continuous-Time System". *Journal of Basic Engineering*, June 1960, pp.371-393.

23. Kalman, R.E.; Bertram, J.E.; "Control System Analysis and Design Via the Second Method of Lyapunov -- Discrete-Time Systems"; *Trans. of the ASME*, June 1960, pp.394-400.

24. H. Freeman; Discrete-Time System, Prentice Hall, Englewood Cliffs, New Jersey, 1973. Chapter 7, pp.157-175

25. Monkewich, O.; Steenaart, W.; "Companding for Digital Filters". *Proc. IEEE ISCAS 1975*, pp.68-71.

26. Monkewich, O.; Steenaart, W.; "Stored Product Digital Filtering with Non-Linear Quantization". *Proc. IEEE ISCAS 1976*, pp.157-160.

27. Dubois, D.; Steenaart, W.; "High Speed Stored Product Digital Filtering". *IEEE Trans. on Circuits and Systems* vol. CAS-29, June 1982, pp.390-393.

28. Claasen, T.A.C.M.; "Quantization Noise Analysis of Digital Filters With Controlled Quantization". Electronics Letters vol. 12, Jan. 1976, pp.46-47.

Appendix Computer Simulation Program Listing

```
C LCYCLE FORTRAN                                                        LCY00010
C PROGRAM TO SIMULATE THE PROPOSED C.Q. METHOD                          LCY00020
C THE QUANTIZATION STEP SIZE IS 1.  I.E. PRODUCTS ARE QUANTIZED TO      LCY00030
C INTEGERS.                                                             LCY00040
C THE PROGRAM STARTS WITH THE INITIAL CONDITIONS SUPPLIED BY THE USER   LCY00050
C AND CONTINUES UNTIL THE ZERO STATES, OR A MAXIMUM NUMBER OF           LCY00060
C ITERATIONS HAS ELAPSED                                                LCY00070
C                                                                       LCY00080
C                                                                       LCY00090
C VARIABLE DEFINITION:                                                  LCY00100
C                                                                       LCY00110
C   YCQ                                                                 LCY00120
C   B1, B2 - FILTER COEFFICIENTS.                                       LCY00130
C   M - THE MID-POINT VALUE IN THE X2-INTERVAL IN THE PROPOSED C.Q.     LCY00140
C   X1, X2 - THE FILTER STATES, THE USER ENTERS THE INITIAL CONDITIONS. LCY00150
C   MAXITN - THE MAXIMUM NUMBER OF ITERATIONS THE PROGRAM IS TO RUN.    LCY00160
C                                                                       LCY00170
C                                                                       LCY00180
C                                                                       LCY00190
      DIMENSION  EP(4,2)                                                LCY00200
      REAL M                                                            LCY00210
C                                                                       LCY00220
C                                                                       LCY00230
C INPUT SECTION: THE PROGRAM READS INPUT FROM THE FILE 'FILE FT08O01'   LCY00240
C                                                                       LCY00250
      READ(8,1) B1, B2                                                  LCY00260
      READ(8,1) X1, X2                                                  LCY00270
    1 FORMAT(2F10.5)                                                    LCY00280
C                                                                       LCY00290
C                                                                       LCY00300
      WRITE(9,2) B1, B2, X1, X2                                         LCY00310
    2 FORMAT(/,5X,'B1=',F8.5,5X,'B2=',F8.5,5X,/,5X,                     LCY00320
     # 'INITIAL CONDITIONS:',4X,'X1=',F8.5,5X,'X2=',F8.5)               LCY00330
C                                                                       LCY00340
C                                                                       LCY00350
C COMPUTE Q-MATRIX                                                      LCY00360
C                                                                       LCY00370
      DET=(1.-B2)*(1.+B2+B1)*(1.+B2-B1)                                 LCY00380
      Q11=(1.+B2)*(1.+B2**2)/DET                                        LCY00390
      Q12=(-B1*(1.+B2**2))/DET                                          LCY00400
      Q22=Q11+1.                                                        LCY00410
      WRITE(9,4) Q11, Q12, Q22                                          LCY00420
    4 FORMAT(/,5X,'Q11=',F8.3,5X,'Q12=',F8.3,5X,'Q22=',F8.3)            LCY00430
C                                                                       LCY00440
C OUTPUT HEADING                                                        LCY00450
C                                                                       LCY00460
      WRITE(9,3)                                                        LCY00470
    3 FORMAT(//,2X,'N',7X,'Y',9X,'X1',7X,'X2',8X,'P1',6X,'P1Q',         LCY00480
     # 6X,'P2',6X,'P2Q',5X,'COD')                                       LCY00490
C                                                                       LCY00500
C                                                                       LCY00510
      MAXITN=300                                                        LCY00520
C                                                                       LCY00530
C                                                                       LCY00540
      N=1                                                               LCY00550
```

```
   80  IF((X1 .EQ. 0 .AND. X2 .EQ. 0) .OR. (N .GT. MAXITN)) GO TO 100   LCY00560
       IF(X1 .NE. 0) GO TO 20                                           LCY00570
           PROD1=0.                                                     LCY00580
           PROD1Q=0.                                                    LCY00590
           PROD2=0.                                                     LCY00600
           PROD2Q=0.                                                    LCY00610
           ICODE=0                                                      LCY00620
           GO TO 90                                                     LCY00630
C                                                                       LCY00640
C CALCULATE THE FOUR POSSIBLE PRODUCTS AND QUANTIZATION ERRORS          LCY00650
C                                                                       LCY00660
   20  CALL CALPNE(B1, X1, PROD1, PROD1U, EROR1U, PROD1D, EROR1D)       LCY00670
       CALL CALPNE(B2, X1, PROD2, PROD2U, EROR2U, PROD2D, EROR2D)       LCY00680
C                                                                       LCY00690
C CALCULATE THE ASSOCIATED X2-INTERVALS                                 LCY00700
C SUBROUTINE X2INTV RETURNS THE END POINTS OF THE INTERVALS IN          LCY00710
C THE ARRAY EP()                                                        LCY00720
C                                                                       LCY00730
       CALL X2INTV(Q11, Q12, Q22, B1, B2, X1, EROR1U, EROR1D,           LCY00740
     #             EROR2U, EROR2D, EP, ICODE)                           LCY00750
C                                                                       LCY00760
       IF(ICODE .NE. 0) GO TO 70                                        LCY00770
C                                                                       LCY00780
C FIND ANY 2 NON-OVERLAPPING X2-INTERVALS AND COMPUTE THE MID-POINT     LCY00790
C M BETWEEN THEM                                                        LCY00800
C                                                                       LCY00810
       CALL GENM(EP, M, I1, I2)                                         LCY00820
       IF(I1 .NE. 0) GO TO 21                                           LCY00830
C                                                                       LCY00840
C SUBROUTINE GENM CANNOT FIND ANY NON-OVERLAPPING X2-INTERVALS, ABORT   LCY00850
C PROGRAM EXECUTION. THIS SHOULD NEVER HAPPEN !                         LCY00860
C                                                                       LCY00870
       WRITE(9,5)                                                       LCY00880
    5  FORMAT(/,'  NO NON-OVERLAPPING X2-INTERVALS')                    LCY00890
       GO TO 100                                                        LCY00900
C                                                                       LCY00910
   21  ICODE=I2                                                         LCY00920
       IF(X2 .LE. M) ICODE=I1                                           LCY00930
C                                                                       LCY00940
C ASSIGN THE CORRECT QUANTIZATION TO THE FILTER PRODUCTS                LCY00950
C                                                                       LCY00960
   70  CALL ASSIGN(ICODE, PROD1Q, PROD2Q, PROD1U, PROD1D,               LCY00970
     #             PROD2U, PROD2D)                                      LCY00980
C                                                                       LCY00990
   90  YCQ=PROD1Q+X2                                                    LCY01000
C                                                                       LCY01010
C OUTPUT RESULTS                                                        LCY01020
C                                                                       LCY01030
       WRITE(9,6) N, YCQ, X1, X2, PROD1, PROD1Q, PROD2, PROD2Q, ICODE   LCY01040
    6  FORMAT(/,I4,4X,7(F7.2,2X),I3)                                    LCY01050
C                                                                       LCY01060
       X2=PROD2Q                                                        LCY01070
       X1=YCQ                                                           LCY01080
       N=N+1                                                            LCY01090
       GO TO 80                                                         LCY01100
```

```
  100  STOP                                                              LCYO1110
       END                                                               LCYO1120
C                                                                        LCYO1130
C                                                                        LCYO1140
C END OF MAIN PROGRAM                                                    LCYO1150
C                                                                        LCYO1160
C                                                                        LCYO1170
C                                                                        LCYO1180
      SUBROUTINE CALPNE(COEF,SIGNAL,PROD,PRODUP,ERORUP,PRODDN,ERORDN)    LCYO1190
C                                                                        LCYO1200
C THIS SUBROUTINE CALCULATES THE PRODUCT AND QUANTIZATION ERROR OF       LCYO1210
C (-COEF * SIGNAL) USING "UP" AND "DOWN" QUANTIZATION                    LCYO1220
C THE QUANTIZED PRODUCT IS AN INTEGER                                    LCYO1230
C                                                                        LCYO1240
      PROD=-COEF*SIGNAL                                                  LCYO1250
      PTEMP=AINT(PROD)                                                   LCYO1260
      IF((PROD-PTEMP) .NE. 0) GO TO 20                                   LCYO1270
        PRODUP=PROD                                                      LCYO1280
        PRODDN=PROD                                                      LCYO1290
        ERORUP=0.                                                        LCYO1300
        ERORDN=0.                                                        LCYO1310
        RETURN                                                           LCYO1320
   20 PRODUP=PTEMP+1.                                                    LCYO1330
      IF(PTEMP .LT. 0) PRODUP=PTEMP-1.                                   LCYO1340
      IF(PTEMP .EQ. 0) PRODUP=SIGN(1.,PROD)                              LCYO1350
C                                                                        LCYO1360
      PRODDN=PTEMP                                                       LCYO1370
C                                                                        LCYO1380
      ERORUP=PRODUP-PROD                                                 LCYO1390
      ERORDN=PRODDN-PROD                                                 LCYO1400
      RETURN                                                             LCYO1410
      END                                                                LCYO1420
C                                                                        LCYO1430
C                                                                        LCYO1440
C                                                                        LCYO1450
C                                                                        LCYO1460
      SUBROUTINE X2INTV(Q11, Q12, Q22, B1, B2, X1, EROR1U, EROR1D,       LCYO1470
     # EROR2U, EROR2D, EP, ID)                                           LCYO1480
C                                                                        LCYO1490
C THIS SUBROUTINE COMPUTES THE END POINTS OF THE X2-INTERVALS            LCYO1500
C RESULTS ARE PUT IN THE ARRAY EP                                        LCYO1510
C                                                                        LCYO1520
      DIMENSION    EP(4,2)                                               LCYO1530
C                                                                        LCYO1540
      ID=0                                                               LCYO1550
      K=1                                                                LCYO1560
C                                                                        LCYO1570
   50 CALL ASSIGN(K, EROR1, EROR2, EROR1U, EROR1D, EROR2U, EROR2D)       LCYO1580
C                                                                        LCYO1590
      CALL CIRCLE(Q11, Q12, Q22, B1, B2, EROR1, EROR2, X1C, X2C, RSQ)    LCYO1600
C                                                                        LCYO1610
      RTEST=RSQ-(X1-X1C)**2                                              LCYO1620
      IF(RTEST .GT. 0) GO TO 22                                          LCYO1630
        ID=K                                                             LCYO1640
        GO TO 100                                                        LCYO1650
```

```
C                                                                       LCYO1660
   22  EP(K,1)=X2C+SQRT(RTEST)                                          LCYO1670
       EP(K,2)=X2C-SQRT(RTEST)                                          LCYO1680
       K=K+1                                                            LCYO1690
       IF(K .LE. 4) GO TO 50                                            LCYO1700
  100  RETURN                                                           LCYO1710
       END                                                              LCYO1720
C                                                                       LCYO1730
C                                                                       LCYO1740
C                                                                       LCYO1750
C                                                                       LCYO1760
       SUBROUTINE CIRCLE(Q11, Q12, Q22, B1, B2, E1, E2, X1C, X2C, RSQ)  LCYO1770
C                                                                       LCYO1780
       X1C=Q12*E1-(Q12*B1+Q22*B2)*E2                                    LCYO1790
       X2C=Q11*E1+Q12*E2                                                LCYO1800
       RSQ=E1**2*Q11+E2**2*Q22+2.*E1*E2*Q12+                            LCYO1810
     # ((Q12*B1+Q22*B2)*E2-Q12*E1)**2+(Q11*E1+Q12*E2)**2                LCYO1820
C                                                                       LCYO1830
       RETURN                                                           LCYO1840
       END                                                              LCYO1850
C                                                                       LCYO1860
C                                                                       LCYO1870
C                                                                       LCYO1880
C                                                                       LCYO1890
       SUBROUTINE GENM(EP, M, I1, I2)                                   LCYO1900
C                                                                       LCYO1910
       DIMENSION    EP(4,2)                                             LCYO1920
       REAL M                                                           LCYO1930
C                                                                       LCYO1940
       I1=0                                                             LCYO1950
       I2=0                                                             LCYO1960
C                                                                       LCYO1970
       DO 5 I=1, 3                                                      LCYO1980
         N=I+1                                                          LCYO1990
         DO 6 J=N, 4                                                    LCYO2000
           IF((EP(J,1) .GT. EP(I,2) .AND. EP(J,1) .LT. EP(I,1)  .OR.    LCYO2010
     #         EP(J,2) .GT. EP(I,2) .AND. EP(J,2) .LT. EP(I,1)  .OR.    LCYO2020
     #         EP(J,1) .GE. EP(I,1) .AND. EP(J,2) .LE. EP(I,2)))        LCYO2030
     #         GO TO 6                                                  LCYO2040
C                                                                       LCYO2050
           IF(EP(I,1) .LT. EP(J,1)) GO TO 20                            LCYO2060
             I1=I                                                       LCYO2070
             I2=J                                                       LCYO2080
             M=(EP(I,2)+EP(J,1))/2.                                     LCYO2090
             GO TO 100                                                  LCYO2100
C                                                                       LCYO2110
   20      I1=J                                                         LCYO2120
           I2=I                                                         LCYO2130
           M=(EP(I,1)+EP(J,2))/2.                                       LCYO2140
           GO TO 100                                                    LCYO2150
    6    CONTINUE                                                       LCYO2160
    5  CONTINUE                                                         LCYO2170
  100  RETURN                                                           LCYO2180
       END                                                              LCYO2190
C                                                                       LCYO2200
```

```
C                                                                       LCY02210
C                                                                       LCY02220
C                                                                       LCY02230
      SUBROUTINE ASSIGN(K, E1, E2, E1U, E1D, E2U, E2D)                  LCY02240
C                                                                       LCY02250
      IF(K .NE. 1) GO TO 20                                             LCY02260
        E1=E1U                                                          LCY02270
        E2=E2U                                                          LCY02280
        RETURN                                                          LCY02290
C                                                                       LCY02300
   20 IF(K .NE. 2) GO TO 21                                             LCY02310
        E1=E1U                                                          LCY02320
        E2=E2D                                                          LCY02330
        RETURN                                                          LCY02340
C                                                                       LCY02350
   21 IF(K .NE. 3) GO TO 22                                             LCY02360
        E1=E1D                                                          LCY02370
        E2=E2U                                                          LCY02380
        RETURN                                                          LCY02390
C                                                                       LCY02400
   22 E1=E1D                                                            LCY02410
      E2=E2D                                                            LCY02420
      RETURN                                                            LCY02430
      END                                                               LCY02440
```

```
C NOISE1 FORTRAN                                                        NOI00010
C PROGRAM TO COMPUTE THE QUANTIZATION NOISE OUTPUT OF A SECOND ORDER    NOI00020
C DIGITAL FILTER SECTION (RECURSIVE PART) WHEN EXCITED BY AN (O,V)      NOI00030
C GAUSSIAN INPUT.                                                       NOI00040
C THE CASES CONSIDERED ARE:                                             NOI00050
C   1. THE IDEAL FILTER - NO QUANTIZATION NOISE, USED AS THE STANDARD   NOI00060
C      TO COMPUTE THE QUANTIZATION NOISE IN THE OTHER FILTER            NOI00070
C      IMPLEMENTATIONS.                                                 NOI00080
C   2. FILTER PRODUCTS ARE ROUNDED.                                     NOI00090
C   3. FILTER PRODUCTS ARE TRUNCATED.                                   NOI00100
C   4. ONE MAGNITUDE TRUNCATION QUANTIZER.                              NOI00110
C   5. LAWRENCE AND MITRA'S CONTROLLED QUANTIZATION.                    NOI00120
C   6. THE PROPOSED CONTROLLED QUANTIZATION.                            NOI00130
C                                                                       NOI00140
C                                                                       NOI00150
C VARIABLE DEFINITION:                                                  NOI00160
C   W() - FILTER INPUT ARRAY CONTAINING GAUSSIAN SAMPLES GENERATED BY   NOI00170
C         SUBROUTINE GENN                                               NOI00180
C                                                                       NOI00190
C   YIDEAL - OUTPUT FROM THE IDEAL FILTER.                              NOI00200
C   YROUND - OUTPUT FROM FILTER WITH ROUNDED PRODUCTS.                  NOI00210
C   YTRUNC - OUTPUT FROM FILTER WITH TRUNCATED PRODUCTS.                NOI00220
C   YTRONE - OUTPUT FROM FILTER WITH 1 M.T. QUANTIZER.                  NOI00230
C   YCQ1,  - OUTPUT FROM FILTER USING LAWRENCE AND MITRA'S              NOI00240
C            CONTROLLED QUANTIZATION.                                   NOI00250
C   YCQ2   - OUTPUT FROM FILTER USING THE PROPOSED C.Q.                 NOI00260
C   B1, B2 - FILTER COEFFICIENTS, THESE MUST BE WITHIN THE STABILITY    NOI00270
C            REGION DEFINED BY  |B2| < 1                                NOI00280
C                               |B1| < 1 + B2                           NOI00290
C   M - THE MID-POINT VALUE IN THE X2-INTERVAL IN THE PROPOSED C.Q.     NOI00300
C   X1, X2 - THE INITIAL CONDITIONS OF THE FILTER STATES                NOI00310
C   ITN - THE NUMBER OF ITERATIONS THE PROGRAM IS TO RUN.  THIS SHOULD  NOI00320
C   INCLUDE 4000 SAMPLES FOR THE FILTER TRANSIENTS                      NOI00330
C                                                                       NOI00340
C                                                                       NOI00350
C                                                                       NOI00360
      DIMENSION  W(15000)                                               NOI00370
      DIMENSION  EP(4,2)                                                NOI00380
      DOUBLE PRECISION  DSEED                                           NOI00390
      REAL M                                                            NOI00400
C                                                                       NOI00410
C                                                                       NOI00420
C INPUT SECTION: THE PROGRAM READS INPUT FROM THE FILE 'FILE FT08O01'   NOI00430
      READ(8,1) B1, B2                                                  NOI00440
      READ(8,1) X1, X2                                                  NOI00450
    1 FORMAT(2F10.5)                                                    NOI00460
C                                                                       NOI00470
      READ(8,2) ITN                                                     NOI00480
    2 FORMAT(I5)                                                        NOI00490
C                                                                       NOI00500
      WRITE(9,44) B1, B2, X1, X2                                        NOI00510
   44 FORMAT(/,5X,'B1=',F8.5,5X,'B2=',F8.5,5X,/,5X,                     NOI00520
     # 'INITIAL CONDITIONS:',4X,'X1=',F8.5,5X,'X2=',F8.5,/,5X,          NOI00530
     # 'INPUT IS N(O,V) GAUSSIAN NOISE')                                NOI00540
C                                                                       NOI00550
```

```
C GENERATE GAUSSIAN INPUT                                              NOI00560
C                                                                      NOI00570
      DSEED=17356.DO                                                   NOI00580
      CALL GENN(DSEED, ITN, W)                                         NOI00590
C                                                                      NOI00600
C QUANTIZE NOISE SAMPLES TO INTEGERS                                   NOI00610
C                                                                      NOI00620
      CALL QUANT(W, ITN)                                               NOI00630
C                                                                      NOI00640
C                                                                      NOI00650
C COMPUTE Q-MATRIX                                                     NOI00660
C                                                                      NOI00670
      DET=(1.-B2)*(1.+B2+B1)*(1.+B2-B1)                                NOI00680
      Q11=(1.+B2)*(1.+B2**2)/DET                                       NOI00690
      Q12=(-B1*(1.+B2**2))/DET                                         NOI00700
      Q22=Q11+1.                                                       NOI00710
      WRITE(9,45) Q11, Q12, Q22                                        NOI00720
   45 FORMAT(/,5X,'Q11=',F8.3,5X,'Q12=',F8.3,5X,'Q22=',F6.3)           NOI00730
C                                                                      NOI00740
C                                                                      NOI00750
C INITIALIZATION                                                       NOI00760
C                                                                      NOI00770
      NTRANS=4000                                                      NOI00780
      NSAMPL=ITN-NTRANS                                                NOI00790
      EROUND=0.                                                        NOI00800
      ETRUNC=0.                                                        NOI00810
      ETRUN1=0.                                                        NOI00820
      ECQ1=0.                                                          NOI00830
      ECQ2=0.                                                          NOI00840
      X1ID = X1                                                        NOI00850
      X1RD = X1                                                        NOI00860
      X1TR = X1                                                        NOI00870
      X1TR1 = X1                                                       NOI00880
      X1CQ1 = X1                                                       NOI00890
      X1CQ2 = X1                                                       NOI00900
      X2ID = X2                                                        NOI00910
      X2RD = X2                                                        NOI00920
      X2TR = X2                                                        NOI00930
      X2TR1 = X2                                                       NOI00940
      X2CQ1 = X2                                                       NOI00950
      X2CQ2 = X2                                                       NOI00960
      B1L=-B1                                                          NOI00970
      B2L=-B2                                                          NOI00980
C                                                                      NOI00990
C                                                                      NOI01000
C START CONTROL LOOP FOR ITN ITERATIONS                                NOI01010
C                                                                      NOI01020
      DO 300 J=1, ITN                                                  NOI01030
C                                                                      NOI01040
C       IDEAL FILTER                                                   NOI01050
C                                                                      NOI01060
        YIDEAL=-B1*X1ID+X2ID+W(J)                                      NOI01070
        X2ID=-B2*X1ID                                                  NOI01080
        X1ID=YIDEAL                                                    NOI01090
C                                                                      NOI01100
```

```
C                                                                       NOIO1110
C       FILTER WITH ROUNDOFF QUANTIZATION                               NOIO1120
C                                                                       NOIO1130
        CALL ROUND(B1, X1RD, PROUND)                                    NOIO1140
        YROUND=PROUND+X2RD+W(J)                                         NOIO1150
        CALL ROUND(B2, X1RD, X2RD)                                      NOIO1160
        X1RD=YROUND                                                     NOIO1170
C                                                                       NOIO1180
C                                                                       NOIO1190
C       FILTER WITH MAGNITUDE TRUNCATION QUANTIZATION                   NOIO1200
C                                                                       NOIO1210
        CALL TRUNC(B1, X1TR, PTRUNC)                                    NOIO1220
        YTRUNC=PTRUNC+X2TR+W(J)                                         NOIO1230
        CALL TRUNC(B2, X1TR, X2TR)                                      NOIO1240
        X1TR=YTRUNC                                                     NOIO1250
C                                                                       NOIO1260
C                                                                       NOIO1270
C       FILTER USING LAWRENCE & MITRA'S C.Q.                            NOIO1280
C                                                                       NOIO1290
        YTEMP=B1L*X1CQ1+B2L*X2CQ1+W(J)                                  NOIO1300
        IF((1.-ABS(B1L)-B2L) .GT. 1.) GO TO 110                         NOIO1310
          CALL REGON1(B1L, X1CQ1, X2CQ1, YCQ1, YTEMP, W(J))             NOIO1320
          GO TO 120                                                     NOIO1330
  110   CALL REGON2(X1CQ1, X2CQ1, YCQ1, YTEMP, W(J))                    NOIO1340
  120   X2CQ1=X1CQ1                                                     NOIO1350
        X1CQ1=YCQ1                                                      NOIO1360
C                                                                       NOIO1370
C                                                                       NOIO1380
C       FILTER WITH ONE MAGNITUDE TRUNCATION                            NOIO1390
C                                                                       NOIO1400
        YTRUN1=-B1*X1TR1-B2*X2TR1+W(J)                                  NOIO1410
        YTRUN1=AINT(YTRUN1)                                             NOIO1420
        X2TR1=X1TR1                                                     NOIO1430
        X1TR1=YTRUN1                                                    NOIO1440
C                                                                       NOIO1450
C                                                                       NOIO1460
C       FILTER WITH THE PROPOSED C.Q.                                   NOIO1470
C                                                                       NOIO1480
        IF(X1 .NE. 0) GO TO 20                                          NOIO1490
          PROD1Q=0.                                                     NOIO1500
          PROD2Q=0.                                                     NOIO1510
          ICODE=0                                                       NOIO1520
          GO TO 90                                                      NOIO1530
                                                                        NOIO1540
C CALCULATE THE FOUR POSSIBLE PRODUCTS AND QUANTIZATION ERRORS          NOIO1550
   20   CALL CALPNE(B1, X1, PROD1U, EROR1U, PROD1D, EROR1D)             NOIO1560
        CALL CALPNE(B2, X1, PROD2U, EROR2U, PROD2D, EROR2D)             NOIO1570
C                                                                       NOIO1580
C CALCULATE THE ASSOCIATED X2-INTERVALS                                 NOIO1590
C SUBROUTINE X2INTV RETURNS THE END POINTS OF THE INTERVALS IN          NOIO1600
C THE ARRAY EP()                                                        NOIO1610
C                                                                       NOIO1620
        CALL X2INTV(Q11, Q12, Q22, B1, B2, X1, EROR1U, EROR1D,          NOIO1630
     #              EROR2U, EROR2D, EP, ICODE)                          NOIO1640
C                                                                       NOIO1650
```

```
        IF(ICODE .NE. 0) GO TO 70                                       NOIO1660
C                                                                       NOIO1670
C FIND ANY 2 NON-OVERLAPPING X2-INTERVALS AND COMPUTE THE MID-POINT     NOIO1680
C M BETWEEN THEM                                                        NOIO1690
C                                                                       NOIO1700
        CALL GENM(EP, M, I1, I2)                                        NOIO1710
        IF(I1 .NE. 0) GO TO 21                                          NOIO1720
C SUBROUTINE GENM CANNOT FIND ANY NON-OVERLAPPING X2-INTERVALS, ABORT   NOIO1730
C PROGRAM EXECUTION. THIS SHOULD NEVER HAPPEN !                         NOIO1740
C                                                                       NOIO1750
          WRITE(9,9)                                                    NOIO1760
    9     FORMAT(/,'  NO NON-OVERLAPPING X2-INTERVALS')                 NOIO1770
          GO TO 100                                                     NOIO1780
C                                                                       NOIO1790
   21   ICODE=I2                                                        NOIO1800
        IF(X2 .LE. M) ICODE=I1                                          NOIO1810
C                                                                       NOIO1820
C ASSIGN THE CORRECT QUANTIZATION TO THE FILTER PRODUCTS                NOIO1830
C                                                                       NOIO1840
   70   CALL ASSIGN(ICODE, PROD1Q, PROD2Q, PROD1U, PROD1D,              NOIO1850
     #              PROD2U, PROD2D)                                     NOIO1860
C                                                                       NOIO1870
   90   YCQ2=PROD1Q+X2+W(J)                                             NOIO1880
        X2=PROD2Q                                                       NOIO1890
        X1=YCQ2                                                         NOIO1900
C                                                                       NOIO1910
C                                                                       NOIO1920
C ACCUMULATE QUANTIZATION ERROR POWER AFTER TRANSIENT SAMPLES           NOIO1930
C                                                                       NOIO1940
        IF(J .LE. NTRANS) GO TO 300                                     NOIO1950
        EROUND=EROUND+(YROUND-YIDEAL)*(YROUND-YIDEAL)                   NOIO1960
        ETRUNC=ETRUNC+(YTRUNC-YIDEAL)*(YTRUNC-YIDEAL)                   NOIO1970
        ETRUN1=ETRUN1+(YTRUN1-YIDEAL)*(YTRUN1-YIDEAL)                   NOIO1980
        ECQ1=ECQ1+(YCQ1-YIDEAL)*(YCQ1-YIDEAL)                           NOIO1990
        ECQ2=ECQ2+(YCQ2-YIDEAL)*(YCQ2-YIDEAL)                           NOIO2000
C                                                                       NOIO2010
  300  CONTINUE                                                         NOIO2020
C                                                                       NOIO2030
C COMPUTE THE MEAN ERROR POWER.  THE RESULT REPRESENTS THE NORMALIZED   NOIO2040
C MEAN ERROR POWER BECAUSE THE QUANTIZATION STEP SIZE IS 1              NOIO2050
C                                                                       NOIO2060
      EROUND=EROUND/FLOAT(NSAMPL)                                       NOIO2070
      ETRUNC=ETRUNC/FLOAT(NSAMPL)                                       NOIO2080
      ETRUN1=ETRUN1/FLOAT(NSAMPL)                                       NOIO2090
      ECQ1=ECQ1/FLOAT(NSAMPL)                                           NOIO2100
      ECQ2=ECQ2/FLOAT(NSAMPL)                                           NOIO2110
C                                                                       NOIO2120
C OUTPUT SECTION                                                        NOIO2130
C                                                                       NOIO2140
      WRITE(9,47) NSAMPL                                                NOIO2150
   47 FORMAT(/,4X,'NO. OF SAMPLES USED IN MSE CALCULATION = ',          NOIO2160
     # I5)                                                              NOIO2170
      WRITE(9,48) EROUND, ETRUNC, ETRUN1, ECQ1, ECQ2                    NOIO2180
   48 FORMAT(/,5X,                                                      NOIO2190
     # 'NORMALIZED QUANTIZATION ERROR POWER DUE TO ROUNDING = ',        NOIO2200
```

```
     # F12.4,/,5X,                                                          NOIO2210
     # 'NORMALIZED QUANTIZATION ERROR POWER DUE TO MT = ',                  NOIO2220
     # F12.4,/,5X,                                                          NOIO2230
     # 'NORMALIZED QUANTIZATION ERROR POWER DUE TO 1 MT = ',                NOIO2240
     # F12.4,/,5X,                                                          NOIO2250
     # 'NORMALIZED QUANTIZATION ERROR POWER DUE TO LAWRENCE CQ = ',         NOIO2260
     # F12.4,/,5X,                                                          NOIO2270
     # 'NORMALIZED QUANTIZATION ERROR POWER DUE TO PROPOSED CQ = ',         NOIO2280
     # F12.4)                                                               NOIO2290
 100 STOP                                                                   NOIO2300
     END                                                                    NOIO2310
C                                                                           NOIO2320
C END OF MAIN PROGRAM                                                       NOIO2330
C                                                                           NOIO2340
C                                                                           NOIO2350
C                                                                           NOIO2360
C                                                                           NOIO2370
      SUBROUTINE GENN(DSEED, NSAMP, W)                                      NOIO2380
C                                                                           NOIO2390
C SUBROUTINE TO GENERATE NSAMP GAUSSIAN SAMPLES                             NOIO2400
C OUTPUT IS RETURNED IN THE ARRAY W()                                       NOIO2410
C                                                                           NOIO2420
      DOUBLE PRECISION    DSEED                                             NOIO2430
      DIMENSION           W(15000)                                          NOIO2440
C                                                                           NOIO2450
      DO 10 K=1, NSAMP                                                      NOIO2460
        TEMP=0.                                                             NOIO2470
        DO 20 I=1, 12                                                       NOIO2480
          TEMP=TEMP+GGUBFS(DSEED)                                           NOIO2490
 20     CONTINUE                                                            NOIO2500
        W(K)=10.*(TEMP-6.0)                                                 NOIO2510
 10   CONTINUE                                                             NOIO2520
      RETURN                                                                NOIO2530
      END                                                                   NOIO2540
C                                                                           NOIO2550
C                                                                           NOIO2560
C                                                                           NOIO2570
C                                                                           NOIO2580
      SUBROUTINE QUANT(X, NSAMP)                                            NOIO2590
C                                                                           NOIO2600
C THIS SUBROUTINE QUANTIZES A VECTOR WITH NSAMP ELEMENTS TO INTEGERS        NOIO2610
C USING ROUNDING                                                            NOIO2620
C                                                                           NOIO2630
      DIMENSION    X(15000)                                                 NOIO2640
C                                                                           NOIO2650
      DO 10 K=1, NSAMP                                                      NOIO2660
        XTEMP=AINT(X(K))                                                    NOIO2670
        IF((ABS(X(K))-ABS(XTEMP)) .GE. 0.5) GO TO 20                        NOIO2680
          X(K)=XTEMP                                                        NOIO2690
          GO TO 10                                                          NOIO2700
C                                                                           NOIO2710
 20     IF(X(K) .GE. 0) X(K)=XTEMP+1.                                       NOIO2720
        IF(X(K) .LT. 0) X(K)=XTEMP-1.                                       NOIO2730
 10   CONTINUE                                                             NOIO2740
      RETURN                                                                NOIO2750
```

```
      END                                                                NOI02760
C                                                                        NOI02770
C                                                                        NOI02780
C                                                                        NOI02790
C                                                                        NOI02800
      SUBROUTINE ROUND(COEF, SIGNAL, PROUND)                             NOI02810
C                                                                        NOI02820
C THIS SUBROUTINE COMPUTES THE PRODUCT OF (-COEF * SIGNAL) AND ROUND IT  NOI02830
C TO AN INTEGER                                                          NOI02840
C                                                                        NOI02850
      IF(SIGNAL .NE. 0) GO TO 10                                         NOI02860
        PROUND=0.                                                        NOI02870
        RETURN                                                           NOI02880
   10 P=-COEF*SIGNAL                                                     NOI02890
      PTEMP=AINT(P)                                                      NOI02900
      IF((ABS(P)-ABS(PTEMP)) .GE. 0.5) GO TO 20                          NOI02910
        PROUND=PTEMP                                                     NOI02920
        RETURN                                                           NOI02930
   20 PROUND=PTEMP+1.                                                    NOI02940
      IF(P .LT. 0) PROUND=PTEMP-1.                                       NOI02950
      RETURN                                                             NOI02960
      END                                                                NOI02970
C                                                                        NOI02980
C                                                                        NOI02990
C                                                                        NOI03000
C                                                                        NOI03010
      SUBROUTINE TRUNC(COEF, SIGNAL, PTRUNC)                             NOI03020
C                                                                        NOI03030
C THIS SUBROUTINE COMPUTES THE PRODUCT OF (-COEF * SIGNAL) AND TRUNCATES NOI03040
C IT TO AN INTEGER                                                       NOI03050
C                                                                        NOI03060
      IF(SIGNAL .NE. 0) GO TO 10                                         NOI03070
        PTRUNC=0.                                                        NOI03080
        RETURN                                                           NOI03090
   10 P=-COEF*SIGNAL                                                     NOI03100
      PTRUNC=AINT(P)                                                     NOI03110
      RETURN                                                             NOI03120
      END                                                                NOI03130
C                                                                        NOI03140
C                                                                        NOI03150
C                                                                        NOI03160
C                                                                        NOI03170
      SUBROUTINE CALPNE(COEF,SIGNAL,PRODUP,ERORUP,PRODDN,ERORDN)         NOI03180
C                                                                        NOI03190
C THIS SUBROUTINE CALCULATES THE PRODUCT AND QUANTIZATION ERROR OF       NOI03200
C (-COEF * SIGNAL) USING "UP" AND "DOWN" QUANTIZATION                    NOI03210
C THE QUANTIZED PRODUCT IS AN INTEGER                                    NOI03220
C                                                                        NOI03230
      P=-COEF*SIGNAL                                                     NOI03240
      PTEMP=AINT(P)                                                      NOI03250
      IF((P-PTEMP) .NE. 0) GO TO 20                                      NOI03260
        PRODUP=P                                                         NOI03270
        PRODDN=P                                                         NOI03280
        ERORUP=0.                                                        NOI03290
        ERORDN=0.                                                        NOI03300
```

```
        RETURN                                                          NOI03310
   20   PRODUP=PTEMP+1.                                                 NOI03320
        IF(PTEMP .LT. 0) PRODUP=PTEMP-1.                                NOI03330
        IF(PTEMP .EQ. 0) PRODUP=SIGN(1.,P)                              NOI03340
C                                                                       NOI03350
        PRODDN=PTEMP                                                    NOI03360
C                                                                       NOI03370
        ERORUP=PRODUP-P                                                 NOI03380
        ERORDN=PRODDN-P                                                 NOI03390
        RETURN                                                          NOI03400
        END                                                             NOI03410
C                                                                       NOI03420
C                                                                       NOI03430
C                                                                       NOI03440
C                                                                       NOI03450
      SUBROUTINE X2INTV(Q11, Q12, Q22, B1, B2, X1, EROR1U, EROR1D,      NOI03460
     # EROR2U, EROR2D, EP, ID)                                          NOI03470
C                                                                       NOI03480
C THIS SUBROUTINE COMPUTES THE END POINTS OF THE X2-INTERVALS           NOI03490
C RESULTS ARE PUT IN THE ARRAY EP                                       NOI03500
C                                                                       NOI03510
      DIMENSION    EP(4,2)                                            NOI03520
C                                                                       NOI03530
      ID=0                                                              NOI03540
      K=1                                                               NOI03550
C                                                                       NOI03560
   50 CALL ASSIGN(K, EROR1, EROR2, EROR1U, EROR1D, EROR2U, EROR2D)      NOI03570
C                                                                       NOI03580
      CALL CIRCLE(Q11, Q12, Q22, B1, B2, EROR1, EROR2, X1C, X2C, RSQ)   NOI03590
C                                                                       NOI03600
      RTEST=RSQ-(X1-X1C)**2                                             NOI03610
      IF(RTEST .GT. 0) GO TO 22                                         NOI03620
        ID=K                                                            NOI03630
        GO TO 100                                                       NOI03640
C                                                                       NOI03650
   22 EP(K,1)=X2C+SQRT(RTEST)                                           NOI03660
      EP(K,2)=X2C-SQRT(RTEST)                                           NOI03670
      K=K+1                                                             NOI03680
      IF(K .LE. 4) GO TO 50                                             NOI03690
  100 RETURN                                                            NOI03700
      END                                                               NOI03710
C                                                                       NOI03720
C                                                                       NOI03730
C                                                                       NOI03740
C                                                                       NOI03750
      SUBROUTINE CIRCLE(Q11, Q12, Q22, B1, B2, E1, E2, X1C, X2C, RSQ)   NOI03760
C                                                                       NOI03770
      X1C=Q12*E1-(Q12*B1+Q22*B2)*E2                                     NOI03780
      X2C=Q11*E1+Q12*E2                                                 NOI03790
      RSQ=E1**2*Q11+E2**2*Q22+2.*E1*E2*Q12+                             NOI03800
     # ((Q12*B1+Q22*B2)*E2-Q12*E1)**2+(Q11*E1+Q12*E2)**2                NOI03810
C                                                                       NOI03820
      RETURN                                                            NOI03830
      END                                                               NOI03840
C                                                                       NOI03850
```

```
C                                                                       NOI03860
C                                                                       NOI03870
C                                                                       NOI03880
      SUBROUTINE GENM(EP, M, I1, I2)                                    NOI03890
C                                                                       NOI03900
      DIMENSION    EP(4,2)                                            NOI03910
      REAL M                                                            NOI03920
C                                                                       NOI03930
      I1=0                                                              NOI03940
      I2=0                                                              NOI03950
C                                                                       NOI03960
      DO 5 I=1, 3                                                       NOI03970
        N=I+1                                                           NOI03980
        DO 6 J=N, 4                                                     NOI03990
          IF((EP(J,1) .GT. EP(I,2) .AND. EP(J,1) .LT. EP(I,1)  .OR.     NOI04000
     #        EP(J,2) .GT. EP(I,2) .AND. EP(J,2) .LT. EP(I,1)  .OR.     NOI04010
     #        EP(J,1) .GE. EP(I,1) .AND. EP(J,2) .LE. EP(I,2)))         NOI04020
     #        GO TO 6                                                   NOI04030
C                                                                       NOI04040
          IF(EP(I,1) .LT. EP(J,1)) GO TO 20                             NOI04050
            I1=I                                                        NOI04060
            I2=J                                                        NOI04070
            M=(EP(I,2)+EP(J,1))/2.                                      NOI04080
            GO TO 100                                                   NOI04090
C                                                                       NOI04100
   20     I1=J                                                         NOI04110
          I2=I                                                          NOI04120
          M=(EP(I,1)+EP(J,2))/2.                                        NOI04130
          GO TO 100                                                     NOI04140
    6   CONTINUE                                                        NOI04150
    5 CONTINUE                                                          NOI04160
  100 RETURN                                                            NOI04170
      END                                                               NOI04180
C                                                                       NOI04190
C                                                                       NOI04200
C                                                                       NOI04210
C                                                                       NOI04220
      SUBROUTINE ASSIGN(K, E1, E2, E1U, E1D, E2U, E2D)                  NOI04230
C                                                                       NOI04240
      IF(K .NE. 1) GO TO 20                                             NOI04250
        E1=E1U                                                          NOI04260
        E2=E2U                                                          NOI04270
        RETURN                                                          NOI04280
C                                                                       NOI04290
   20 IF(K .NE. 2) GO TO 21                                             NOI04300
        E1=E1U                                                          NOI04310
        E2=E2D                                                          NOI04320
        RETURN                                                          NOI04330
C                                                                       NOI04340
   21 IF(K .NE. 3) GO TO 22                                             NOI04350
        E1=E1D                                                          NOI04360
        E2=E2U                                                          NOI04370
        RETURN                                                          NOI04380
C                                                                       NOI04390
   22 E1=E1D                                                            NOI04400
```

```
      E2=E2D                                                            NOI04410
      RETURN                                                            NOI04420
      END                                                               NOI04430
C                                                                       NOI04440
C                                                                       NOI04450
C                                                                       NOI04460
C                                                                       NOI04470
      SUBROUTINE REGON1(B1, X1, X2, YCQ1, Y, U)                         NOI04480
C                                                                       NOI04490
C THIS SUBROUTINE IMPLEMENTS THE REGION 1 QUANTIZATION ARITHMATIC IN     NOI04500
C LAWRENCE AND MITRA'S C.Q. METHOD                                      NOI04510
C                                                                       NOI04520
      IF(X1 .NE. 0  .OR.  X2 .NE. 0) GO TO 10                           NOI04530
        YCQ1=U                                                          NOI04540
        RETURN                                                          NOI04550
C                                                                       NOI04560
   10 IF(B1 .LT. 0) GO TO 20                                            NOI04570
        CALL REG1(X1, X2, YCQ1, Y)                                      NOI04580
        RETURN                                                          NOI04590
   20 X2TEMP=-X2                                                        NOI04600
      CALL REG1(X1, X2TEMP, YCQ1, Y)                                    NOI04610
      RETURN                                                            NOI04620
      END                                                               NOI04630
C                                                                       NOI04640
C                                                                       NOI04650
C                                                                       NOI04660
C                                                                       NOI04670
      SUBROUTINE REG1(X1, X2, YCQ1, Y)                                  NOI04680
C                                                                       NOI04690
      IF(X1 .EQ. X2) GO TO 10                                           NOI04700
      IF(X2 .GT. Y) GO TO 20                                            NOI04710
C                                                                       NOI04720
      CALL RNDDN(Y, YCQ1)                                               NOI04730
      RETURN                                                            NOI04740
C                                                                       NOI04750
   20 CALL RNDUP(Y, YCQ1)                                               NOI04760
      RETURN                                                            NOI04770
C                                                                       NOI04780
   10 IF(X2 .LT. Y) GO TO 30                                            NOI04790
      CALL RNDDN(Y, YCQ1)                                               NOI04800
      RETURN                                                            NOI04810
C                                                                       NOI04820
   30 CALL RNDUP(Y, YCQ1)                                               NOI04830
      RETURN                                                            NOI04840
      END                                                               NOI04850
C                                                                       NOI04860
C                                                                       NOI04870
C                                                                       NOI04880
C                                                                       NOI04890
      SUBROUTINE RNDDN(Y, YQUANT)                                       NOI04900
C                                                                       NOI04910
      YTEM=AINT(Y)                                                      NOI04920
      IF(Y .LT. 0) GO TO 10                                             NOI04930
        YQUANT=YTEM                                                     NOI04940
        RETURN                                                          NOI04950
```

```
C                                                                       NOI04960
   10  YQUANT=YTEM-1.                                                   NOI04970
       RETURN                                                           NOI04980
       END                                                              NOI04990
C                                                                       NOI05000
C                                                                       NOI05010
C                                                                       NOI05020
C                                                                       NOI05030
       SUBROUTINE RNDUP(Y, YQUANT)                                      NOI05040
C                                                                       NOI05050
       YTEM=AINT(Y)                                                     NOI05060
       IF(Y .LT. 0) GO TO 10                                            NOI05070
         YQUANT=YTEM+1.                                                 NOI05080
         RETURN                                                         NOI05090
C                                                                       NOI05100
   10  YQUANT=YTEM                                                      NOI05110
       RETURN                                                           NOI05120
       END                                                              NOI05130
C                                                                       NOI05140
C                                                                       NOI05150
C                                                                       NOI05160
C                                                                       NOI05170
       SUBROUTINE REGON2(X1, X2, YQUANT, Y, U)                          NOI05180
C                                                                       NOI05190
C THIS SUBROUTINE IMPLEMENTS THE REGION 2 QUANTIZATION ARITHMETIC IN     NOI05200
C LAWRENCE AND MITRA'S C.Q. METHOD                                      NOI05210
C                                                                       NOI05220
       IF(X1 .NE. 0  .OR.  X2 .NE. 0) GO TO 10                          NOI05230
         YQUANT=U                                                       NOI05240
         RETURN                                                         NOI05250
C                                                                       NOI05260
   10  IF(X2 .GT. Y) GO TO 20                                           NOI05270
         CALL RNDDN(Y, YQUANT)                                          NOI05280
         RETURN                                                         NOI05290
C                                                                       NOI05300
   20  CALL RNDUP(Y, YQUANT)                                            NOI05310
       RETURN                                                           NOI05320
       END                                                              NOI05330
```