Roundoff Error Problems and Solutions for Conventional Recursive Least Squares Filters

Gregory E. Bottomley

Center for Communication and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, North Carolina
November 1989

CCSP TR-89/26

ABSTRACT

BOTTOMLEY, GREGORY E., "Roundoff Error Problems and Solutions for Conventional Recursive Least Squares Filters," under the direction of S. T. Alexander.

Recursive least squares filters are not widely used in adaptive signal processing applications because of instability problems resulting from finite precision implementation. One problem, explosive divergence, is characterized by a sudden deviation in filter performance. A second problem, weight lock-up, results in a loss of adaptivity.

This dissertation provides a novel fixed point analysis of the conventional recursive least squares (CRLS) algorithm. The analysis completely models all roundoff errors associated with multiplication and division. The analytical results serve two important functions. First, they are used to develop an understanding for how roundoff error can accumulate to cause explosive divergence. Second, the results identify which specific roundoff errors are contributing to the divergence problem. This information is then used to develop techniques which bias the algorithm towards stable performance.

Explosive divergence is also shown to result from overflow of certain algorithmic quantities. The weight lock-up problem is shown to be an underflow problem. A fixed point implementation of the CRLS algorithm is designed so that underflow and overflow are minimized.

The stabilization techniques are integrated into the fixed point implementation to produce a robust finite precision implementation of the CRLS algorithm. While stability is not guaranteed, stable performance is demonstrated under a variety of signal environments.

Acknowledgements

I would like to personally thank those that contributed to this dissertation. First, I would like to thank my advisor, Dr. S. T. Alexander, for introducing me to adaptive signal processing, for allowing me to explore the area of finite precision effects, and for providing guidance and encouragement. I would like to thank the Department of Electrical and Computer Engineering and the National Science Foundation for providing support for this research. I am also indebted to my fellow graduate students and Center for Communications and Signal Processing personnel for their help and assistance.

Finally, I would also like to thank my family for their encouragement and support. I would especially like to thank my wife, Laura Bottomley, for agreeing to start this venture and for providing much needed encouragement, help, and support.

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION

Within the last three decades, adaptive filtering has emerged as a powerful tool for many signal processing applications. Adaptive filters provide greater flexibility because of their ability to adapt to unknown or time-varying situations [ALEX 86]. As a result, performance can be optimized even when operating conditions are not known in advance.

Applications for adaptive filters can be found in a many different disciplines. In telecommunications, adaptive filters are used to provide echo cancellation for long distance calling. Also, data transmission systems using the telephone network rely on adaptive channel equalization to minimize distortion. In sonar signal processing, adaptive beamforming is used to minimize side-lobe interference. In control engineering, adaptive filtering is used to identify unknown system parameters. Many other applications exist [WIDR 85, GIOR 85].

1.1 Background

The basic adaptive filtering structure is shown in Figure 1-1. The filter operates on input data, x(n), to produce an output y(n). This output is then compared to a desired signal, d(n). The difference, e(n), is the error signal. The filter parameters, denoted by the vector w(n), are modified or updated each time iteration n so as to minimize the error signal in some way.

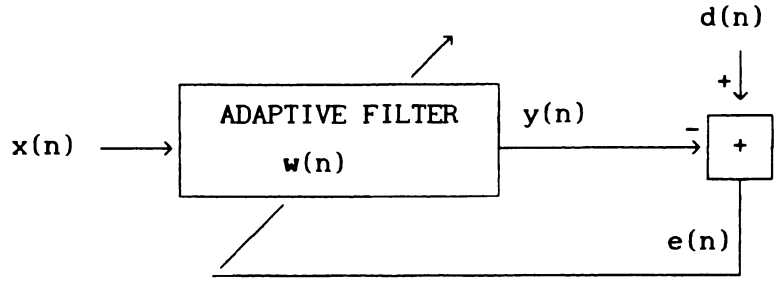

Figure 1-1. Adaptive FIR filter

The most common adaptive filtering structure is the finite impulse response (FIR) filter. The output of the adaptive filter is simply a weighted sum of the input signal as indicated by the following equations:

$$\mathbf{x}(n) = [x(n) \quad x(n-1) \quad x(n-2) \ldots x(n-N+1)]^T \tag{1-1}$$

$$\mathbf{w}(n) = [w_0(n) \quad w_1(n) \quad w_2(n) \ldots w_{N-1}(n)]^T \tag{1-2}$$

$$y(n) = \mathbf{x}^T(n)\,\mathbf{w}(n) \tag{1-3}$$

$$e(n) = d(n) - y(n) \tag{1-4}$$

where $\mathbf{x}(n)$ is a vector of most recent input data values.

The most commonly used method for updating the weight vector is known as the Least Mean Square (LMS) algorithm. This algorithm is based on an iterative technique for minimizing the expected value of the error squared. Because of its relative simplicity and robustness, the LMS algorithm is widely used in adaptive signal processing applications.

Another approach for updating the weight vector is to use a Recursive Least Squares (RLS) algorithm. This family of algorithms is based on minimizing the following error criterion [ALEX 86]:

$$\varepsilon(n) = \sum_{i=1}^{n} \alpha^{n-i} \left[d(i) - \mathbf{x}^T(i)\, \mathbf{w}(n) \right]^2 \qquad (1\text{-}5)$$

The criterion is a weighted sum of squared errors, giving rise to the name "least squares." Each error consists of the difference between the desired signal and the filter output using the most recent weight vector. The factor α, known as the forgetting factor, allows for the filter to place more importance on errors resulting from recent data.

The conventional recursive least squares (CRLS) algorithm is one member of the RLS family of algorithms [GOOD 77, PICI 78, JOHN 82, LJUN 83, PROA 83, GOOD 84, COWA 85, ALEX 86, HAYK 86, BELL 87b, LJUN 87]. The algorithm is specified as follows [ALEX 86]:

Initial conditions:

$$\mathbf{w}(0) = \mathbf{x}(0) = \mathbf{0} \qquad (1\text{-}6a)$$

$$\mathbf{C}(0) = \delta\, \mathbf{I} \qquad (\delta \gg 1) \qquad (1\text{-}6b)$$

For n = 1, 2, ... **do:**

$$e(n|n-1) = d(n) - \mathbf{x}^T(n)\, \mathbf{w}(n-1) \qquad (1\text{-}7a)$$

$$\mu(n) = \mathbf{x}^T(n)\, \mathbf{C}(n-1)\, \mathbf{x}(n) \qquad (1\text{-}7b)$$

$$\mathbf{g}(n) = \frac{\mathbf{C}(n-1)\, \mathbf{x}(n)}{\alpha + \mu(n)} \qquad (1\text{-}7c)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)\, e(n|n-1) \qquad (1\text{-}7d)$$

$$\mathbf{C}(n) = \frac{1}{\alpha} \left[\mathbf{C}(n-1) - \mathbf{g}(n)\, \mathbf{x}^T(n)\, \mathbf{C}(n-1) \right] \qquad (1\text{-}7e)$$

(optional)

$$y(n) = \mathbf{x}^T(n)\, \mathbf{w}(n) \qquad (1\text{-}7f)$$

$$e(n) = d(n) - y(n) \qquad (1\text{-}7g)$$

The notation "e(n:i)" is used to indicate that the error is computed at iteration n using the weights from iteration i. The vector **g**(n) is sometimes referred to as the Kalman gain or simply gain. The scalar α, the same forgetting factor as given in (1-5), is a design parameter chosen between 0 and 1, usually close to 1.

The matrix **C**(n) can be viewed as a scaled estimate of the inverse autocorrelation matrix of the data [COWA 85, HAYK 86]. As a result, it has the properties of symmetry and positive definiteness [COWA 85].

Compared with the LMS algorithm, RLS algorithms require more computations per iteration. However, RLS algorithms provide faster convergence of the weight vector to some optimal set of weights [ALEX 86, HAYK 86]. Unfortunately, these filters have not gained widespread use due to serious instability problems which occur when these filters are implemented in finite precision [CIOF 87].

1.2 Finite precision effects

When implemented in finite precision, several RLS filters, including the CRLS filter, deviate significantly from optimal performance [CIOF 87]. Two major problems arise as a result of finite precision effects: explosive divergence and weight lock-up.

When explosive divergence occurs, the weights diverge away from the optimal least squares solution. This divergence occurs when accumulation of roundoff error causes the **C**(n) matrix in (1-7e) to lose the property of being positive definite [HSU 82, FABR 85, COWA 85, CIOF 87, BELL 87b].

The second problem, weight lock-up, occurs when the weights cease to adapt. This can result from the property that the **C**(n) matrix converges to a zero matrix when the forgetting factor is 1 [GODA 74]. Consequently, the Kalman gain in (1-7c) approaches zero, and the weight vector in (1-7d) remains unchanged. Thus, the filter loses the ability to adapt to changing conditions. This problem can occur even when the forgetting factor is less than 1.

1.3 Purpose and overview

The purpose of this dissertation is to study the fixed point arithmetic implementation of the CRLS algorithm. The first goal is to uncover the specific underlying mechanisms which cause these finite precision problems to occur. The second goal is to develop techniques which overcome these problems without significantly degrading RLS performance.

To meet these goals, the dissertation is organized as follows. First, Chapter 2 provides a review of previous research on the finite precision problems in RLS filtering. The problems of explosive divergence and weight lock-up are identified and characterized. It is shown that most analytical work does not deal directly with these problems.

A detailed fixed point analysis of the conventional RLS (CRLS) algorithm is presented in Chapter 3. Results indicate that explosive divergence is caused by a roundoff error bias in the finite precision **C**(n) matrix. The analysis also predicts a scenario for divergence which agrees with previously published experimental results.

In Chapter 4, a symmetry preserving form of the CRLS algorithm is selected for implementation. It is shown that this form is also susceptible to explosive divergence. Specific roundoff errors contributing to the problem of explosive divergence are identified. Techniques are developed which bias these roundoff errors so that explosive divergence can be prevented.

Chapter 5 addresses the problems of underflow and overflow. Underflow causes weight lock-up, and overflow can lead to explosive divergence. A detailed 16-bit fixed point design which prevents underflow and overflow is presented.

In Chapter 6, the results of Chapters 4 and 5 are integrated together to form a robust fixed point implementation of the CRLS filter. Simulation results are used to show that the problems of explosive divergence and weight lock-up have been prevented while still preserving the least squares performance of the CRLS algorithm. While stability is not guaranteed, stable performance under diverse signal environments is demonstrated.

Chapter 7 presents conclusions and areas of future research. The main conclusion is that analysis can be used to identify and largely overcome the roundoff error mechanisms which lead to explosive divergence and weight lock-up. Future work is needed to apply this novel approach to the fast RLS filters.

CHAPTER 2 - LITERATURE REVIEW

Finite precision effects have long been of interest to those implementing adaptive filters. For the conventional recursive least squares (CRLS) filter, there has been disagreement on how the filter performs when implemented in finite precision. The purpose of this chapter is to summarize previous work related to the numerical stability of the CRLS algorithm and related algorithms.

The earliest finite precision work in this area can be found in the Kalman filtering literature. This body of work is summarized in section 1. Section 2 discusses work specifically addressing the numerical stability of the CRLS algorithm. Finally, since the fast RLS algorithms also suffer from similar numerical instability problems, section 3 provides a summary of finite precision problems with fast RLS algorithms.

2.1 Kalman filtering finite precision problems

The CRLS filter can be viewed as a special case of the conventional Kalman filter (CKF) [GENI 68, ASTR 71]. In fact, the CRLS algorithm was derived directly from the Kalman filter for use in adaptive equalization [GODA 74]. As a result, both filters exhibit similar finite precision problems. Since these problems were documented in the Kalman filtering literature well before they appeared in the RLS literature, a discussion of the Kalman filtering literature is provided first. The discussion consists of three sub-sections: experimental results, analytical results, and solutions.

2.1.1 Experimental results

An early reference on finite precision effects [SORE 66] discusses two problems that occur in Kalman filtering: 1) the autocorrelation matrix **P**(n) (corresponding to **C**(n) in the CRLS filter) can lose the property of nonnegative definiteness, and 2) the Kalman gain can go to zero, causing the estimates (weights) to freeze or lock up. These are the problems of explosive divergence and weight lock-up discussed in Chapter 1.

Later work concentrates on explosive divergence. Several examples of this type of divergence were collected by Leondes [LEON 70]. Bierman and Thornton [BIER 77a] emphasize the numerical instability of the conventional Kalman filter, providing simulation results demonstrating explosive divergence. They also list other simulation work performed in the late 1960s. Bierman's text [BIER 77b] provides additional simulation references.

2.1.2 Analytical results

Analysis of finite precision problems occurs much later [VERH 86]. Analysis of the finite precision problem is broken down into three steps: 1) bounding the error that occurs in each iteration, 2) determining how that error propagates in subsequent iterations, and 3) determining how the errors interact and accumulate. In all three steps, the authors make several assumptions, including a) small error bounds (based on several approximations) and b) no interaction of

errors (total error is simply the sum of each propagated error). The results of the analysis are bounds on the errors of the various filter quantities. Simulation results are used to confirm these bounds. The authors only analyze a form of the CKF which preserves symmetry, stating that loss of symmetry can lead to divergence.

2.1.3 Solutions

Several solutions to the problem of numerical instability in the conventional Kalman filter have been proposed:

1. Use a square root factorization algorithm [SORE 66, BIER 77a, BIER 77b, VERH 86] or U-D factorization [BIER 77a].
2. Include input noise in the state update model [SORE 66, BIER 77b].
3. Preserve symmetry in the **P**(n) matrix [BIER 77b, VERH 86] by:
   a. averaging the off-diagonal elements of the updated matrix [BIER 77b, VERH 86];
   b. only computing half of the entries in the **P**(n) matrix [BIER 77b, VERH 86]; or
   c. using Joseph's stabilized Kalman filter, which preserves symmetry [VERH 86]. This technique is questionable since simulation results show this algorithm to be numerically unstable [BIER 77a].
4. Replace the expression for updating **P**(n) with a longer, but numerically more stable, expression [BIER 77b].
5. Use more precision [BIER 77b].

6. Bound the diagonal elements of **P**(n) from below, and bound the magnitude of the off-diagonal elements of **P**(n) from above [BIER 77b].

Of these techniques, square root factorization appears to be most often recommended. Several of these techniques are explored again later in works concentrating on the numerical instability of the CRLS and fast RLS algorithms.

2.2 CRLS filtering finite precision problems

This section deals specifically with finite precision work on the CRLS filter. The discussion consists of four sub-sections: experimental results, analytical results, general statements, and solutions.

2.2.1 Experimental results

Many researchers have published experimental results using the CRLS algorithm. Early publications demonstrate stable performance using various levels of precision [SARI 74, ISER 74, GODA 74, GITL 77, GRAU 80, MUEL 81]. All of these results, except for one [MUEL 81], use the forgetting factor α equal to one. Based on these results, many researchers recommended the CRLS algorithm for implementation. However, later experimental works demonstrating numerical stability state that extreme precision is necessary for stable performance [GIOR 85, LING 86].

More recent experimental results show that the CRLS filter suffers from the same finite precision effects found in the conventional Kalman

filter. The first examples of finite precision problems with the CRLS filter are explosive divergence examples. Hsu [HSU 82] states that, in simulating the exponentially weighted CRLS filter (α less than 1) using fixed point precision, performance degrades with word size. Hsu observes that **C**(n) loses positive definiteness at the time of divergence. Loss of positive definiteness is blamed primarily on the fact that the **C**(n) matrix is updated as the difference of two positive semi-definite matrices, reducing numerical accuracy at each iteration. Cioffi [CIOF 87] points out that this is more of a problem when **C**(n) is near singular and the number of iterations is large.

Explosive divergence was also observed by Cioffi and Kailath [CIOF 84], using 32-bit floating point precision and a forgetting factor of 0.98. While the CRLS algorithm was stable for the first 2000 iterations, divergence eventually occurred.

Actual graphs showing explosive divergence of the weights were presented by Fabre and Gueguen [FABR 85]. They observe that the quantity $1/(\alpha + \mu(n))$ becomes negative just before divergence. From equation (1-7b), this implies a loss of positive definiteness. The authors blame the problem on loss of symmetry in the **C**(n) matrix. They suggest two ways of preserving symmetry, and they claim that the problem is solved when symmetry is preserved.

Experimental evidence of weight lock-up is also available. When the forgetting factor is equal to one or very close to one, weight lock-up has been observed [ARDA 86, ARDA 87].

Thus, both explosive divergence and weight lock-up have been observed in experimental results using the CRLS filter. The results indicate that divergence is more likely with less precision and more iterations. This would explain why early results using a high degree of precision and relatively few iterations did not have divergence problems.

2.2.2 Analytical results

Within recent years, much work has been done to analyze how the CRLS filter performs in finite precision. A brief summary of each work is provided, along with a discussion of the results.

1. Ljung and Ljung [LJUN 85]: Ljung and Ljung provide analyses of several RLS algorithms, including CRLS. Their approach is to model the algorithm as a nonlinear dynamical system and to determine how perturbations in the state variables (the weight vector $\mathbf{w}(n)$ and the $\mathbf{C}(n)$ matrix) propagate in an *infinite precision* implementation. Since infinite precision is used, the authors claim that the CRLS and U-D factorization methods will have the same propagation characteristics as the mathematically equivalent RLS algorithm. This algorithm updates an estimate of the data autocorrelation matrix $\mathbf{R}(n)$ (the inverse of $\mathbf{C}(n)$) by:

$$\mathbf{R}(n) = \alpha\, \mathbf{R}(n-1) + \mathbf{x}(n)\, \mathbf{x}^T(n) \qquad (2\text{-}1)$$

where α is the forgetting factor and $\mathbf{x}(n)$ is the vector of current data values. The authors then show that the RLS filter damps out perturbations in both $\mathbf{R}(n)$ and the weights. From this, they conclude

that all three algorithms, the RLS, CRLS, and U-D algorithms, are exponentially stable, with the propagated error in one iteration being α times the original error.

By showing that small perturbations are damped out by an infinite precision algorithm, this analysis helps explain the stable performance observed in early experimental works. However, this analytical approach does not predict the explosive divergence and weight lock-up phenomena. Explosive divergence is not predicted because:

1. interaction and accumulation of multiple errors is not considered.
2. loss of positive definiteness from a finite amount of accumulated error is not considered.

Weight lock-up is not predicted because conditions leading to a zero **C**(n) matrix are not considered.

2. Ling et al. [LING 84, LING 86]: Ling et al. specifically analyze the mechanism for updating the **C**(n) matrix, as opposed to the mechanism in (2-1). By making various approximations and assumptions, they show that the propagation of the error in **C**(n), denoted δ**C**(n), follows:

$$\delta \mathbf{C}(n) = \alpha \, \delta \mathbf{C}(n-1) \qquad (2\text{-}2)$$

which agrees with [LJUN 85]. From this, they conclude that the algorithm is numerically stable as long as the forgetting factor α is less than one. Simulation results using 22-bit floating point mantissas, running 10^6 iterations with α between 0.85 and 0.999, support the stability claim. The filter order used is not indicated.

However, they claim their results only hold if roundoff error is small. Indeed, they were unable to provide stable simulation results for either fixed point or floating point precision using word lengths less than 16 bits. Clearly, this weakens the claim that the CRLS filter is numerically stable. But, it does indicate that stability is a function of the precision used.

3. Ljung [LJUN 86]: Ljung extends previous work [LJUN 85] to the case of nonpersistent excitation. Persistent excitation is a requirement for convergence of the infinite precision CRLS algorithm [JOHN 82, ISER 82]. Persistent excitation means that the input data must have a certain spectral richness [KUBI 88], so that the sample autocorrelation matrix has full rank [SLOC 88]. Explicit mathematical definitions of persistent excitation can be found in the literature [JOHN 82, GOOD 84]. To provide filter convergence when persistent excitation is not available, regularization techniques are used [LJUN 86, KUBI 88].

Ljung provides a novel form of regularization and shows that the resulting algorithm is exponentially stable. Since the analytical approach is the same as that found in the previously discussed work [LJUN 85], the same comments are applicable. Neither explosive divergence nor weight lock-up are addressed. However, unlike the previous work, the author recognizes that only the second of the following three problems is being addressed: the size of instantaneous roundoff error, the propagation of errors, and the accumulation of errors.

4. Ardalan [ARDA 86]: Ardalan examines how floating point roundoff error affects RLS filtering in general. This analysis models the errors made in some of the computations and determines how the weight error variance is affected by error accumulation. Results show that there is a tradeoff between two sources of error depending on the choice of the forgetting factor α. If α is one, exponential divergence of the weights can occur. However, this may be prevented by weight lock-up, which prevents the weight error variance from growing without bound. If α is less than one, the weight error variance is bounded, allowing for stable performance.

This modeling of errors is more sophisticated than previous works, and the results help to quantify how the choice of α affects the amount of weight error due to precision effects. Also, recognition is given to the problem of weight lock-up. However, as with the previously discussed analyses, no consideration is given to loss of positive definiteness in the $\mathbf{C}(n)$ matrix (explosive divergence).

5. Ardalan and Alexander [ARDA 87]: This work is similar to the one above, providing similar results for the fixed point implementation of CRLS. Divergence of the weights is described as a random walk phenomenon, occurring when α is one, or the quantization noise in the Kalman gain and the input data variance are high. The authors conclude that, in general, the algorithm is stable if α is less than one.

As with the previous work, this work helps quantify the amount of weight error due to finite precision implementation. Again, weight lock-up is recognized when α is one or very close to one. As with

previous works, loss of positive definiteness is not considered.

6. Slock and Kailath [SLOC 88]: This paper analyzes the CRLS algorithm by assuming that roundoff error is so small, that a linearization of the error propagation mechanism can be used. Using this approach, the algorithm is shown to be exponentially stable under persistent excitation. When persistent excitation is not present, the weights exhibit the random walk phenomenon.

The comments of Ljung and Ljung's work [LJUN 85] are also applicable here. Linearization of the error propagation does not allow for error interaction. Also, loss of positive definiteness of **C**(n) due to roundoff error accumulation is not considered. Finally, weight lock-up is not considered.

7. Verhaegen [VERH 89]: Verhaegen studies the problem of explosive divergence, attributing it to the loss of symmetry in the **C**(n) matrix. First, two forms of the CRLS algorithm are considered. While neither guarantees a symmetric **C**(n) matrix under finite precision, one form is shown to be better at preserving symmetry than the other.

The paper then presents a theorem which states that preserving symmetry also guarantees positive definiteness, preventing explosive divergence. The theorem is based on the assumptions that 1) the total roundoff error at time iteration n is the *sum* of the propagated error from previous iterations, and 2) the instantaneous errors at each iteration satisfy certain bounds.

The paper's analysis is one of the first to consider the problem of maintaining positive definiteness in the **C**(n) matrix. However, the

conclusion that preserving symmetry preserves positive definiteness only holds under the assumptions listed. Interaction of roundoff errors is not modeled, and weight lock-up is not addressed. In Chapter 4, it will be shown both analytically and experimentally that preserving symmetry does not guarantee positive definiteness.

2.2.3 General statements

Much of the literature contains statements concerning numerical stability, with or without the support of simulation or analytical results. Statements can be found claiming numerical stability, warning about weight lock-up, and warning about explosive divergence.

First, there are a few claims that the CRLS algorithm is numerically stable. Based on analysis, several authors have concluded that the CRLS algorithm is stable [LJUN 85, LING 84, LING 86]. Graupe's text [GRAU 84] recommends the CRLS algorithm as being numerically robust.

However, the majority of statements point out that the CRLS algorithm has problems when implemented in finite precision. Some researchers simply state that the algorithm is sensitive to roundoff, resulting in instability [PROA 83, CIOF 84, CIOF 85, ELEF 84, ELEF 86, GIOR 85, HAYK 86, MANO 87]. Others specifically discuss the loss of positive definiteness [COWA 85, CIOF 87, BELL 87b]. Cioffi [CIOF 87] actually lists three effects caused by finite precision: 1) error accumulation leading to an indefinite solution, 2) additional noise in the weights, and 3) amplification of numerical effects when $\alpha < 1$.

Reasons given for loss of positive definiteness or explosive divergence include: 1) **C**(n) is updated as the difference of two matrices [HSU 82, CIOF 87] leading to loss of precision, and 2) loss of symmetry in the **C**(n) matrix [FABR 85, VERH 89]. The possibility of weight lock-up when α is one is discussed by Godard [GODA 74]. Lack of persistent excitation (PE) has also been blamed for numerical instability [CIOF 87, SLOC 88]. However, Verhaegen [VERH 89] points out that lack of PE also causes the infinite precision algorithm to diverge. He comments that the roundoff error does grow larger without PE, but that the size of the error relative to the diverging theoretical values remains the same. Recent work [KUBI 88] suggests that a signal with PE in infinite precision may appear to lack PE when quantized and manipulated using finite precision.

2.2.4 Solutions

Several solution techniques, some similar to those suggested for the Kalman filter, have been recommended to prevent explosive divergence:

1. Use a factorization algorithm which factors the **C**(n) matrix, such as square root factorization [PROA 83, COWA 85, GIOR 85], U-D factorization [HSU 82, HAYK 86, CIOF 87], or QR factorization [CIOF 87]. The Modified Gram-Schmidt algorithm has also been proposed [MANO 87].
2. Force symmetry in the **C**(n) matrix [FABR 85, VERH 89].

3. Add a small value periodically to the diagonal of the equivalent **R**(n) matrix [CIOF 87], or use some other form of regularization [LJUN 86, KUBI 88].

To avoid weight lock-up, the following have been suggested:

1. Freeze the Kalman gain before it goes to zero, then use a steepest descent algorithm [GODA 74].
2. Require a large input noise variance [ARDA 87].

The use of a factorization algorithm is highly recommended. Since this technique was successful when used with the more general Kalman filter, it should also be effective in RLS filtering. In Chapter 4 of this dissertation, preserving symmetry in the **C**(n) matrix is shown both analytically and experimentally to be ineffective in preventing explosive divergence. More information on regularization can be found in the references cited. Finally, the schemes recommended to prevent weight lock-up appear impractical for many applications.

2.3 Fast RLS filtering finite precision problems

Several other RLS algorithms, known collectively as fast RLS algorithms, also exhibit numerical instability. Explosive divergence has been observed in the fast Kalman filters [MUEL 81, HSU 82, CIOF 84, LIN 84, FABR 85, FABR 86], the FAEST filter [CIOF 84], the FTF filter [CIOF 84, CIOF 87, KIM 88, BOTO 89], and other related forms [CIOF 84, CIOF 85, FABR 86, CIOF 87].

At the time of divergence, the angle parameter $\gamma(n)$, an algorithmic quantity that normally takes on values between 0 and 1 [ALEX 86], becomes negative [CIOF 84, LIN 84, FABR 86, HAYK 86]. The corresponding parameter in the CRLS algorithm is [HAYK 86, BELL 87b, KUBI 88]:

$$\gamma(n) = \frac{\alpha}{\alpha + \mu(n)} \tag{2-3}$$

though earlier references appear to omit the α in the numerator [FABR 85, FABR 86]. The expression in (2-3) can be easily derived using standard filter relationships. In Chapter 3, it is shown that $\gamma(n)$ in the CRLS algorithm behaves in the same manner as $\gamma(n)$ is the fast RLS algorithms when explosive divergence occurs.

Analytical work with fast RLS algorithms has concentrated on showing that the algorithms are unstable [LJUN 85, CIOF 87, KIM 88, SLOC 88]. As for preventing divergence, the most recommended solutions are periodic re-initialization [CIOF 84, LIN 84, ELEF 84, FABR 86, ELEF 86, HAYK 86, CIOF 87, KIM 88] and the use of error feedback techniques [BOTO 87, BOTO 89, BELL 87a, BELL 87b, CIOF 87, SLOC 88, BENA 88]. None of these techniques guarantees stability [BOTO 89, CIOF 87]. Techniques for prolonging the period between re-initialization have also been suggested [CIOF 84, BELL 87b, KIM 88]. Finally, weight lock-up is also a problem for the fast RLS algorithms [BOTO 87, BOTO 89].

Since this dissertation concentrates on the CRLS algorithm, the interested reader is referred to the references cited above for more information on the fast RLS algorithms.

2.4. Conclusion

The literature contains sufficient evidence to show that the CRLS filter suffers from the finite precision problems of explosive divergence and weight lock-up. Explosive divergence is linked to loss of positive definiteness in the **C**(n) matrix, and weight lock-up is likely to occur when the forgetting factor is 1.

However, most analytical works do not address these problems. Analysis has concentrated on error propagation properties, basing stability claims on exponential error decay. Other work has quantified finite precision effects under stable operation. The only work that has addressed explosive divergence appears to be incomplete.

Several solutions to the divergence problem have been presented. The most effective solution appears to be to use a factorization algorithm instead of the CRLS algorithm. Finally, fast RLS algorithms suffer from the same finite precision problems.

CHAPTER 3 - COMPLETE ERROR MODELING ANALYSIS OF CRLS

In this chapter, a fixed point analysis of the CRLS algorithm is performed. The goal is to provide an analytical explanation for why and how explosive divergence occurs. Also, qualitative relations between instability and numerical precision, filter parameters, and signal statistics are derived. These relations allow environments which are prone to instability to be identified.

The chapter is organized as follows. First, recursion equations for accumulated roundoff error are derived. Second, these equations are manipulated to determine how errors propagate and to identify biases. These results are then used to explain how explosive divergence occurs, predicting behavior that agrees with previously published results. Sensitivity of instability to the forgetting factor, signal statistics, and numerical precision is discussed. Next, simulation results demonstrating the divergence behavior of the CRLS algorithm are presented. Finally, it is concluded that analytical results can be used to explain explosive divergence.

3.1 Derivation of error recursions

In this section, recursive equations describing how roundoff error accumulates in the fixed point CRLS algorithm are derived. A few relatively minor assumptions are made: 1) the operations of addition and subtraction do not introduce any roundoff error, and 2) the desired signal d(n) and data signal x(n) are fixed point signals. The first

assumption, commonly used in fixed point analyses [SAMS 83, ARDA 87], holds as long as overflow does not occur. As will be shown in Chapter 5, overflow can be prevented by various scaling techniques. The second assumption is made to separate the problem of quantization introduced in the A/D converter from the performance of the fixed point adaptive filter.

The analysis approach taken is to account for all errors introduced in a fixed point implementation of the CRLS algorithm. There are basically two types of error: instantaneous roundoff error and accumulated roundoff error. Instantaneous roundoff error occurs whenever a multiplication or division is performed [BARN 85]. The following notation is used to express the roundoff error, $r_m[a,b]$, introduced by rounding the product of "a" and "b":

$$r_m[a,b] = ab - Q[ab] \tag{3-1}$$

where ab denotes the infinite precision product and Q[ab] denotes the rounded product. Similarly, the roundoff error, $r_d[a,b]$, introduced by rounding the quotient a/b, is given by:

$$r_d[a,b] = a/b - Q[a/b] \tag{3-2}$$

The second type of error, accumulated roundoff error, represents the total error introduced by not using an infinite precision implementation. Each quantity in the CRLS algorithm (equation 1-7) has an accumulated error associated with it. For notation purposes, the subscript q is used to denote fixed point quantities, and the letter d is used to represent accumulated error. For example, the fixed point filter's weight vector at iteration n, denoted $\mathbf{w}_q(n)$, differs from the

infinite precision filter's weight vector, $\mathbf{w}(n)$, by an error $\mathbf{d}_w(n)$ such that:

$$\mathbf{d}_w(n) = \mathbf{w}(n) - \mathbf{w}_q(n) \tag{3-3}$$

This error includes instantaneous errors as well as interaction and accumulation of all past errors resulting from the fixed point implementation. Unlike past analyses [SLOC 88, VERH 89], the assumption that roundoff error accumulation is additive is not made. Similar definitions can be made for the other accumulated error quantities: $d_e(n|n-1)$, $d_\mu(n)$, $\mathbf{d}_g(n)$, and $\mathbf{D}_C(n)$.

With this notation, a fixed point implementation of the CRLS algorithm specified in equation (1-7) can be expressed as [ALEX 86]:

Initial conditions:

$$\mathbf{w}_q(0) = \mathbf{x}(0) = \mathbf{0} \tag{3-4a}$$

$$\mathbf{C}_q(0) = \delta\, \mathbf{I} \qquad (\delta >> 1) \tag{3-4b}$$

For n = 1, 2, ... **do:**

$$e_q(n|n-1) = d(n) - Q[\, \mathbf{x}^T(n)\, \mathbf{w}_q(n-1)\,] \tag{3-5a}$$

$$\mathbf{p}_q(n) = Q[\, \mathbf{C}_q(n-1)\, \mathbf{x}(n)\,] \tag{3-5b}$$

$$\mu_q(n) = Q[\, \mathbf{x}^T(n)\, \mathbf{p}_q(n)\,] \tag{3-5c}$$

$$\mathbf{g}_q(n) = Q\left[\frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)} \right] \tag{3-5d}$$

$$\mathbf{w}_q(n) = \mathbf{w}_q(n-1) + Q[\, \mathbf{g}_q(n)\, e_q(n|n-1)\,] \tag{3-5e}$$

$$\mathbf{b}_q^T(n) = Q[\, \mathbf{x}^T(n)\, \mathbf{C}_q(n-1)\,] \tag{3-5f}$$

$$\mathbf{C}_q(n) = Q\left[\frac{\mathbf{C}_q(n-1) - Q[\, \mathbf{g}_q(n)\, \mathbf{b}_q^T(n)\,]}{\alpha} \right] \qquad (3\text{-}5g)$$

Some forms of the CRLS algorithm do not implement (3-5f), replacing $\mathbf{b}_q^T(n)$ with $\mathbf{p}_q^T(n)$ in (3-5g) [HAYK 86]. This is valid when $\mathbf{C}_q(n)$ is symmetric, which holds in infinite precision. Since symmetry is not guaranteed in the above fixed point implementation (see Chapter 4), (3-5f) has been included. Forms of the CRLS algorithm which preserve symmetry even in finite precision are considered in Chapter 4. In the above form, it is assumed that α and δ have been chosen so that no error occurs in converting them to fixed point quantities.

Using the infinite precision CRLS definition (1-7), the fixed point precision CRLS definition (3-5), and the notation given in equations (3-1) through (3-3), error recursions for the accumulated error in each algorithmic quantity can be derived.

As a complete example of the procedure, the recursion for $d_e(n|n-1)$ is presented in detail. The procedure starts with the infinite precision expression for e(n|n-1) given by equation (1-7a):

$$e(n|n-1) = d(n) - \mathbf{x}^T(n)\, \mathbf{w}(n-1) \qquad (3\text{-}6a)$$

Substituting (3-3) for $\mathbf{w}(n)$ in (3-6a) gives:

$$e(n|n-1) = d(n) - \mathbf{x}^T(n)\, \mathbf{w}_q(n-1) - \mathbf{x}^T(n)\, \mathbf{d}_w(n-1) \qquad (3\text{-}6b)$$

Expanding $\mathbf{w}_q(n-1)$ using (3-3) and applying (3-1) gives:

$$e(n|n-1) = d(n) - Q[\, \mathbf{x}^T(n)\, \mathbf{w}_q(n-1)\,] - r_{m1}(n) - \mathbf{x}^T(n)\, \mathbf{d}_w\,(n-1) \qquad (3\text{-}6c)$$

where, for simplicity of notation,

$$r_{m1}(n) = r_m[\, \mathbf{x}^T(n),\, \mathbf{w}_q(n-1)\,] \qquad (3\text{-}6d)$$

Using (3-5a) in (3-6d) then gives:

$$e(n|n-1) = e_q(n|n-1) - r_{m1}(n) - \mathbf{x}^T(n)\,\mathbf{d}_w(n-1) \qquad (3\text{-}6e)$$

Therefore, using the definition for accumulated error in (3-3) in (3-6e) gives:

$$d_e(n|n-1) = -r_{m1}(n) - \mathbf{x}^T(n)\,\mathbf{d}_w(n-1) \qquad (3\text{-}6f)$$

Using this approach, error recursions for all algorithmic quantities can be derived. Detailed derivations of the other error recursions are provided in Appendix A. The resulting error recursions are summarized below.

Initial conditions:

$$\mathbf{d}_w(0) = \mathbf{0}, \quad \mathbf{D}_C(0) = \mathbf{0} \qquad (3\text{-}7)$$

For n = 1, 2, ... **do:**

$$d_e(n|n-1) = -r_{m1}(n) - \mathbf{x}^T(n)\,\mathbf{d}_w(n-1) \qquad (3\text{-}8a)$$

$$d_\mu(n) = \mathbf{x}^T(n)\,\mathbf{D}_C(n-1)\,\mathbf{x}(n) + \mathbf{x}^T(n)\,\mathbf{r}_{m2}(n) + r_{m3}(n) \qquad (3\text{-}8b)$$

$$\mathbf{d}_g(n) = \frac{\mathbf{C}(n-1)\,\mathbf{x}(n)}{\alpha + \mu(n)} - \frac{\mathbf{C}_q(n-1)\,\mathbf{x}(n)}{\alpha + \mu_q(n)} + \frac{\mathbf{r}_{m2}(n)}{\alpha + \mu_q(n)} + \mathbf{r}_{d1}(n) \qquad (3\text{-}8c)$$

$$\mathbf{d}_w(n) = \mathbf{d}_w(n-1) + \mathbf{r}_{m4}(n) + \mathbf{g}(n)\,e(n|n-1) - \mathbf{g}_q(n)\,e_q(n|n-1) \qquad (3\text{-}8d)$$

$$\mathbf{D}_C(n) = \mathbf{R}_{d2}(n) + 1/\alpha\,[\,\mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) - \mathbf{g}_q(n)\,\mathbf{r}_{m5}^T(n)\,] + 1/\alpha\,[\,\mathbf{g}_q(n)\,\mathbf{x}^T(n)\,\mathbf{C}_q(n-1) - \mathbf{g}(n)\,\mathbf{x}^T(n)\,\mathbf{C}(n-1)\,] \qquad (3\text{-}8e)$$

where

$$r_{m1}(n) = r_m[\, \mathbf{x}^T(n),\ \mathbf{w}_q(n-1)\,] \tag{3-9a}$$

$$\mathbf{r}_{m2}(n) = r_m[\, \mathbf{C}_q(n-1),\ \mathbf{x}(n)\,] \tag{3-9b}$$

$$r_{m3}(n) = r_m[\, \mathbf{x}^T(n),\ \mathbf{p}_q(n)\,] \tag{3-9c}$$

$$\mathbf{r}_{d1}(n) = r_d[\, \mathbf{p}_q(n),\ \alpha + \mu_q(n)\,] \tag{3-9d}$$

$$\mathbf{r}_{m4}(n) = r_m[\, \mathbf{g}_q(n),\ e_q(n|n-1)\,] \tag{3-9e}$$

$$\mathbf{r}_{m5}^T(n) = r_m[\, \mathbf{x}^T(n),\ \mathbf{C}_q(n-1)\,] \tag{3-9f}$$

$$\mathbf{R}_{m6}(n) = r_m[\, \mathbf{g}_q(n),\ \mathbf{b}_q^T(n)\,] \tag{3-9g}$$

$$\mathbf{R}_{d2}(n) = r_d[\, \mathbf{C}_q(n-1) - Q[\, \mathbf{g}_q(n)\, \mathbf{b}_q^T(n)\,],\ \alpha\,] \tag{3-9h}$$

General closed form solutions of (3-7) and (3-8) are probably impossible, due to the nonlinear and stochastic nature of (3-8). However, it is clear that accumulated error is passed from one iteration to the next through the terms $\mathbf{d}_w(n-1)$ and $\mathbf{D}_C(n-1)$. Consequently, the error recursions for $\mathbf{d}_w(n)$ and $\mathbf{D}_C(n)$ determine the propagation characteristics of roundoff error in the CRLS algorithm.

3.2 Propagation and bias analysis

The goal of this section is to use the results in section 3.1 to determine the underlying mechanism by which errors propagate from one iteration to the next in the CRLS algorithm. A conditional expected value will be used to determine any error biases. As discussed in section 3.1, errors are propagated from iteration n-1 to iteration n through the $\mathbf{d}_w(n-1)$ and $\mathbf{D}_C(n-1)$ error terms. Consequently, the

recursions for these two error terms are studied in this section.

The analysis consists of three steps: 1) expanding expressions (3-8d) and (3-8e) in terms of instantaneous and accumulated error terms, 2) taking a conditional expected value with respect to instantaneous roundoff error, and 3) assuming steady-state performance of the infinite precision CRLS filter to simplify the expressions. Expression (3-8d) for the weight error is analyzed first, followed by analysis of (3-8e).

3.2.1 Analysis of the weight error recursion

Starting with expression (3-8d) for the weight error, the following substitutions are made:

$$\mathbf{g}_q(n) = \mathbf{g}(n) - \mathbf{d}_g(n) \tag{3-10a}$$

$$e_q(n|n-1) = e(n|n-1) - d_e(n|n-1) \tag{3-10b}$$

Then, the expressions for $d_e(n|n-1)$ and $\mathbf{d}_g(n)$ in (3-8a) and (3-8c) are used. After some algebraic manipulation, the following result is obtained:

$$\mathbf{d}_w(n) = \left[\mathbf{I} - \frac{1}{\alpha + \mu_q(n)} \left(\mathbf{C}(n-1) - \mathbf{D}_C(n-1) \right) \mathbf{x}(n)\, \mathbf{x}^T(n) \right] \mathbf{d}_w(n-1)$$

$$+ \left(1 - \frac{\alpha + \mu(n)}{\alpha + \mu_q(n)} \right) \mathbf{g}(n)\, e(n|n-1) + \frac{\mathbf{D}_C(n-1)\, \mathbf{x}(n)\, e(n|n-1)}{\alpha + \mu_q(n)}$$

$$+ \left(r_{d1}(n) + \frac{r_{m2}(n)}{\alpha + \mu_q(n)} \right) \left(\mathbf{x}^T(n)\, \mathbf{d}_w(n-1) + e(n|n-1) + r_{m1}(n) \right)$$

$$- \frac{1}{\alpha + \mu_q(n)} \left[\mathbf{C}(n-1) - \mathbf{D}_C(n-1) \right] \mathbf{x}(n)\, r_{m1}(n) + \mathbf{r}_{m4}(n) \qquad (3\text{-}11)$$

where

$$\mu_q(n) = \mu(n) - d_\mu(n)$$

$$= \mu(n) - \mathbf{x}^T(n)\, \mathbf{D}_C(n-1)\, \mathbf{x}(n) + \mathbf{x}^T(n)\, \mathbf{r}_{m2}(n) + r_{m3}(n) \qquad (3\text{-}12)$$

At this point, it is assumed that the instantaneous roundoff error terms are zero mean and uncorrelated with each other as well as the other terms in (3-11) and (3-12). With these assumptions, the last three terms in (3-11) can be eliminated by taking the following conditional expected value:

$$\bar{\mathbf{d}}_w(n) = E \{ \mathbf{d}_w(n) \mid \mathbf{C}(n-1), \mathbf{w}(n-1), \mathbf{x}(n), d(n) \} \qquad (3\text{-}13)$$

where the overbar is used to denote a conditional expected value. The conditional expectation is well defined [PAPO 65] and has been employed in other analyses [VITE 66]. The expectation is approximate for (3-11) because instantaneous roundoff terms appear in $\mu_q(n)$, which appears in the denominator of some of the terms. The result is:

$$\bar{\mathbf{d}}_w(n) \cong \left[\mathbf{I} - \frac{1}{\alpha + \bar{\mu}_q(n)} \left[\mathbf{C}(n-1) - \bar{\mathbf{D}}_C(n-1) \right] \mathbf{x}(n)\, \mathbf{x}^T(n) \right] \bar{\mathbf{d}}_w(n-1)$$

$$+ \left[1 - \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \right] \mathbf{g}(n)\, e(n|n-1) + \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)}\, \bar{\mathbf{D}}_C(n-1)\, \frac{\mathbf{x}(n)\, e(n|n-1)}{\alpha + \mu(n)} \qquad (3\text{-}14)$$

where

$$\bar{\mu}_q(n) = \mu(n) - \mathbf{x}^T(n)\, \bar{\mathbf{D}}_C(n-1)\, \mathbf{x}(n) \qquad (3\text{-}15)$$

The next step is to assume that the data are stationary and that the iteration number n is sufficiently large so that the infinite precision quantities in (3-14) and (3-15) have reached steady-state. This approach has been used in past analyses to simplify results [SAMS 83, LING 84, LING 86]. The following steady-state assumptions are made:

$$\mathbf{C}(n) \cong (1-\alpha)\, \mathbf{R}_{xx}^{-1} \qquad (3\text{-}16)$$

$$\mathbf{w}(n) \cong \mathbf{w}^* \qquad (3\text{-}17)$$

where $\mathbf{R}_{xx}$ is the true data autocorrelation matrix and $\mathbf{w}^*$ is the true weight vector. Assumption (3-16) was used in [LING 84] and [LING 86]. Equations (3-16) and (3-17) imply:

$$\mathbf{C}(n) \cong \mathbf{C}(n-1) \qquad (3\text{-}18)$$

$$\mathbf{w}(n) \cong \mathbf{w}(n-1) \qquad (3\text{-}19)$$

Applying equations (3-16) through (3-19) to the infinite precision CRLS algorithm (1-7), the following results can be derived (see Appendix B):

$$\mathbf{g}(n)\, \mathbf{x}^T(n)\, \mathbf{C}(n-1) \cong (1-\alpha)^2\, \mathbf{R}_{xx}^{-1} \qquad (3\text{-}20a)$$

$$\frac{\mathbf{x}(n)\, \mathbf{x}^T(n)}{\alpha + \mu(n)} \cong \mathbf{R}_{xx} \qquad (3\text{-}20b)$$

$$\mathbf{g}(n)\, e(n|n-1) \cong \mathbf{0} \qquad (3\text{-}20c)$$

$$\frac{\mathbf{x}(n)\, e(n|n-1)}{\alpha + \mu(n)} \cong \mathbf{0} \qquad (3\text{-}20d)$$

Applying equations (3-20c) and (3-20d) to (3-14) eliminates the last two terms in (3-14). Using (3-16) and (3-20b) gives:

$$\bar{d}_w(n) \cong \left[1 - (1 - \alpha) \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \right] \bar{d}_w(n-1) \quad \text{(LINEAR TERM)}$$

$$+ \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \bar{D}_C(n-1) R_{xx} \bar{d}_w(n-1) \quad \text{(INTERACTION TERM)} \tag{3-21}$$

which consists of two terms. The first term, denoted the linear term, is a scalar times the previous error vector. The second term, denoted the interaction term, involves interaction with the $\bar{D}_C(n-1)$ error matrix.

First, consider the linear term. For stable propagation of error, the scalar multiplier should be less than one in magnitude. This is clearly true for the case of $\bar{\mu}_q(n) \cong \mu(n)$. In this case, the scalar multiplier is simply α, the forgetting factor. Thus, considering only the linear term, errors are exponentially damped according to α. This result agrees with previous analytical work [LJUN 85].

Now consider the interaction term. This term does not appear in [LJUN 85] because, in analyzing the weight error, it was assumed that the $C_q(n)$ matrix is computed with infinite precision. Using a Schur decomposition for symmetric matrices [GOLU 83], R_{xx} can be expressed as:

$$R_{xx} = M^T \Lambda M \tag{3-22}$$

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues of R_{xx}. Assuming the data satisfy persistent excitation conditions, then R_{xx} is positive definite and has real, positive eigenvalues. Factoring out the largest eigenvalue of R_{xx}, denoted

λ_{max}, gives:

$$\mathbf{R}_{xx} = \lambda_{max} \mathbf{M}^T \mathbf{K} \mathbf{M} \tag{3-23}$$

where $\mathbf{K}$ is a diagonal matrix whose elements are contained in the interval (0,1]. Substituting (3-23) into (3-21) gives:

$$\bar{\mathbf{d}}_w(n) \cong \left[1 - (1 - \alpha) \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \right] \bar{\mathbf{d}}_w(n-1) \quad \text{(LINEAR TERM)}$$

$$+ \lambda_{max} \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \bar{\mathbf{D}}_C(n-1) \mathbf{M}^T \mathbf{K} \mathbf{M} \bar{\mathbf{d}}_w(n-1) \quad \text{(INTERACTION TERM)} \tag{3-24}$$

which reveals that the elements in the interaction term vector are proportional to λ_{max}. Thus, if the elements in the $\bar{\mathbf{D}}_C(n-1)$ matrix are large and λ_{max} is large, then this term can become significant.

3.2.2 Analysis of the C matrix error recursion

Starting with expression (3-8e) for the $\mathbf{C}$ matrix error, the following substitutions are made:

$$\mathbf{g}_q(n) = \mathbf{g}(n) - \mathbf{d}_g(n) \tag{3-25a}$$

$$\mathbf{C}_q(n) = \mathbf{C}(n) - \mathbf{D}_C(n) \tag{3-25b}$$

Then, the expression for $\mathbf{d}_g(n)$ in (3-8c) is used. After some algebraic manipulation, the following result is obtained:

$$\mathbf{D}_C(n) = 1/\alpha \, \mathbf{D}_C(n-1) + 1/\alpha \left(\frac{\mathbf{D}_C(n-1) \, \mathbf{x}(n) \, \mathbf{x}^T(n) \, \mathbf{D}_C(n-1)}{\alpha + \mu_q(n)} \right)$$

$$- 1/\alpha \left(\frac{\mathbf{C}(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{D}_C(n-1) + \mathbf{D}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1)}{\alpha + \mu_q(n)} \right)$$

$$+ 1/\alpha \left(\frac{\alpha + \mu(n)}{\alpha + \mu_q(n)} - 1 \right) \mathbf{g}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) + 1/\alpha \left(\frac{\mathbf{r}_{m2}(n)\ \mathbf{r}_{m5}^T(n)}{\alpha + \mu_q(n)} \right)$$

$$+ 1/\alpha \left(\mathbf{r}_{d1}(n) + \frac{\mathbf{r}_{m2}(n)}{\alpha + \mu_q(n)} \right) \mathbf{x}^T(n) \left(\mathbf{D}_C(n-1) - \mathbf{C}(n-1) \right)$$

$$+ 1/\alpha \left(\frac{\mathbf{D}_C(n-1) - \mathbf{C}(n-1)}{\alpha + \mu_q(n)} \right) \mathbf{x}(n)\ \mathbf{r}_{m5}^T(n) + 1/\alpha\ \mathbf{r}_{d1}(n)\ \mathbf{r}_{m5}^T(n)$$

$$- 1/\alpha\ \mathbf{R}_{m6}(n) + \mathbf{R}_{d2}(n) \qquad (3\text{-}26)$$

where $\mu_q(n)$ is given in (3-12).

Again, it is assumed that instantaneous roundoff error terms are zero mean and uncorrelated with each other as well as the other quantities in the above expression. The same type of conditional expectation given in (3-13) is taken. The result is:

$$\bar{\mathbf{D}}_C(n) \cong 1/\alpha\ \bar{\mathbf{D}}_C(n-1) + 1/\alpha \left(\frac{\bar{\mathbf{D}}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \bar{\mathbf{D}}_C(n-1)}{\alpha + \bar{\mu}_q(n)} \right)$$

$$- 1/\alpha \left(\frac{\mathbf{C}(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \bar{\mathbf{D}}_C(n-1) + \bar{\mathbf{D}}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1)}{\alpha + \bar{\mu}_q(n)} \right)$$

$$+ 1/\alpha \left(\frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} - 1 \right) \mathbf{g}(n)\, \mathbf{x}^T(n)\, \mathbf{C}(n-1) + 1/\alpha \left(\frac{\sigma_r^2\, \mathbf{I}}{\alpha + \bar{\mu}_q(n)} \right) \tag{3-27}$$

where $\bar{\mu}_q(n)$ is given in (3-15). Notice that $E\{\mathbf{r}_{m2}(n)\, \mathbf{r}_{m5}^T(n)\}$ has been replaced with $\sigma_r^2\, \mathbf{I}$, where σ_r^2 is the variance of a zero mean roundoff error. This is because $\mathbf{r}_{m2}(n)$ and $\mathbf{r}_{m5}(n)$ are identical if $\mathbf{C}_q(n-1)$ is symmetric. While this form of CRLS does not guarantee symmetry, it does appear to be quite good at keeping $\mathbf{C}_q(n)$ symmetric [VERH 89]. If $\mathbf{b}_q^T(n)$ had been replaced by $\mathbf{p}_q^T(n)$ in (3-5g), then $\mathbf{r}_{m5}(n)$ would exactly equal $\mathbf{r}_{m2}(n)$.

The third step in this process is to assume that the infinite precision quantities in (3-27) have reached steady-state. Using expressions (3-16), (3-20a) and (3-20b) gives:

$$\bar{\mathbf{D}}_C(n) \cong 1/\alpha \left[1 - 2(1-\alpha) \left(\frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \right) \right] \bar{\mathbf{D}}_C(n-1) \qquad \text{(LINEAR)}$$

$$+ 1/\alpha \left(\frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \right) \bar{\mathbf{D}}_C(n-1)\, \mathbf{R}_{xx}\, \bar{\mathbf{D}}_C(n-1) \qquad \text{(QUADRATIC)}$$

$$+ 1/\alpha \left(\frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} - 1 \right) (1-\alpha)^2\, \mathbf{R}_{xx}^{-1} \qquad \text{(SHIFT)}$$

$$+ 1/\alpha \left(\frac{1}{\alpha + \bar{\mu}_q(n)} \right) \sigma_r^2\, \mathbf{I} \qquad \text{(ROUNDOFF)}$$

(3-28)

The expression in (3-28) consists of four terms. The first term, denoted the linear term, is a scalar times the previous error matrix. The second term, denoted the quadratic term, involves a quadratic interaction of the error matrix with the true data autocorrelation matrix. The third term, denoted the shift term, is a bias proportional to $(1 - \alpha)\mathbf{R}_{xx}^{-1}$, the steady-state value for $\mathbf{C}(n)$. The fourth term, denoted the roundoff term, is a bias along the diagonal, causing $\mathbf{C}_q(n)$, which equals $\mathbf{C}(n) - \mathbf{D}_C(n)$, to have smaller diagonal elements.

First, consider the linear term. For stable propagation of error, the scalar multiplier should be less than one in magnitude. This is clearly true for the case of $\bar{\mu}_q(n) \cong \mu(n)$. In this case, the scalar multiplier is $2 - 1/\alpha$ which equals $\alpha - (1-\alpha)^2/\alpha$. This is approximately equal to α when α is very close to one. Thus, considering only the linear term, errors are exponentially damped according to α. This result agrees with previous analytical work [LJUN 85, LING 84, LING 86]. Also, if $\bar{\mu}_q(n) \cong \mu(n)$, the scalar multiplier is positive, preserving the sign or definiteness of the error matrix.

Next, consider the quadratic term. This term is always positive semi-definite, unless $\alpha + \bar{\mu}_q(n)$ is negative. Substituting (3-23) for $\mathbf{R}_{xx}$ would reveal that this term is proportional to λ_{max}, the largest eigenvalue of $\mathbf{R}_{xx}$. Thus, if the accumulated error is significant ($\bar{\mathbf{D}}_C(n-1)$ contains large elements) and λ_{max} is large, then this term can be large.

The third term, the shift term, causes a scaling shift or bias in the $\mathbf{C}_q(n)$ matrix. In infinite precision at steady-state, $\mathbf{C}(n) \cong$

$(1-\alpha)\mathbf{R}_{xx}^{-1}$. If the shift term were the only error term present, then $\mathbf{C}_q(n)$ would also be proportional to $\mathbf{R}_{xx}^{-1}$, but with a different proportionality factor. As long as this new factor is positive, then $\mathbf{C}_q(n)$ will be positive definite. The shift error term is positive definite if $\bar{\mu}_q(n) < \mu(n)$, which according to (3-15) occurs when $\bar{\mathbf{D}}_C(n-1)$ is positive definite.

Finally the fourth term, the roundoff term, is a bias resulting from roundoff errors being squared when forming the diagonal of the $\mathbf{C}_q(n)$ matrix. This bias causes the diagonal elements of $\mathbf{C}_q(n)$ to be smaller than those of $\mathbf{C}(n)$. This roundoff term is always positive definite, unless $\alpha + \bar{\mu}_q(n)$ is negative.

3.3 Explosive divergence

In this section, the analytical results in section 3.2 are used to develop a scenario for explosive divergence. As discussed in Chapter 2, explosive divergence is linked with $\mathbf{C}_q(n)$ losing the property of positive definiteness. Specifically, the angle parameter $\gamma_q(n)$, given as $\alpha/(\alpha + \mu_q(n))$ according to (2-3), becomes negative (implying $\mu_q(n)$ becomes negative). Consequently, this section begins by examining under what conditions loss of positive definiteness occurs.

3.3.1 Loss of positive definiteness

This sub-section explores how loss of positive definiteness occurs in the fixed point CRLS implementation. First, conditions under which loss of positive definiteness and the start of explosive divergence

occur are identified. Then, the results of section 3.2 are used to show that such conditions can be met by the fixed point implementation.

First, the error matrix $\mathbf{D}_C(n-1)$ is defined such that:

$$\mathbf{C}_q(n-1) = \mathbf{C}(n-1) - \mathbf{D}_C(n-1) \tag{3-29}$$

If $\mathbf{D}_C(n-1)$ is positive definite, then it is possible for $\mathbf{C}_q(n-1)$ to not be positive definite, even though $\mathbf{C}(n-1)$ is positive definite. This relationship is best understood by considering the scalar $\bar{\mu}_q(n)$. From (3-15) and (1-7b):

$$\bar{\mu}_q(n) = \mathbf{x}^T(n)\, \mathbf{C}(n-1)\, \mathbf{x}(n) - \mathbf{x}^T(n)\, \bar{\mathbf{D}}_C(n-1)\, \mathbf{x}(n) \tag{3-30}$$

Also,

$$\bar{\mu}_q(n) = \mathbf{x}^T(n)\, \bar{\mathbf{C}}_q(n-1)\, \mathbf{x}(n) \tag{3-31}$$

where $\bar{\mathbf{C}}_q(n-1)$ is the conditional expected value of $\mathbf{C}_q(n-1)$. The first term in (3-30) is always positive and equals $\mu(n)$. The second term in (3-30) can be positive when $\bar{\mathbf{D}}_C(n-1)$ is not negative definite. It is always positive when $\bar{\mathbf{D}}_C(n-1)$ is positive definite. If the second term is larger than the first, then $\bar{\mu}_q(n)$ is negative. From (3-31), this implies $\bar{\mathbf{C}}_q(n-1)$ is no longer positive definite. Since $\mu(n)$ is theoretically always non-negative, the event that $\bar{\mu}_q(n)$ becomes negative can be considered the start of explosive divergence.

Thus, the conditions under which $\bar{\mu}_q(n)$ becomes negative are the following:

a) the second term in (3-30) is positive, indicating that the error matrix $\bar{\mathbf{D}}_C(n-1)$ contains at least one positive eigenvalue,

b) the second term in (3-30) is large, indicating that the error accumulated in $\bar{\mathbf{D}}_C(n-1)$ is significant, and

c) the first term in (3-30) is small. This term is smallest when the data vector $\mathbf{x}(n)$ aligns with the eigenvector of the smallest eigenvalue of $\mathbf{C}(n-1)$. Using (3-16), this minimum eigenvalue is $(1-\alpha)/\lambda_{max}$, where λ_{max} is the maximum eigenvalue of $\mathbf{R}_{xx}$ as discussed in section 3.2. Thus, the larger λ_{max}, the smaller the first term in (3-30) can become.

So, $\bar{\mathbf{C}}_q(n-1)$ can lose the property of positive definiteness if a) $\bar{\mathbf{D}}_C(n-1)$ is positive definite, b) the elements of $\bar{\mathbf{D}}_C(n-1)$ are significant in magnitude, and c) λ_{max} is large. The last condition is a function of the signal statistics, although scaling of the data can be used to alter it. The first two conditions depend on how errors propagate in the fixed point CRLS algorithm. Equation (3-28) is now used to test whether conditions a) and b) are possible.

First, examination of (3-28) replacing n with n-1 reveals that $\bar{\mathbf{D}}_C(n-1)$ is biased towards being positive definite. Two of the terms in (3-28), the quadratic and roundoff terms, are positive semi-definite and positive definite respectively under stable operation ($\bar{\mu}_q(n)$ is positive). Once these error terms are introduced, the linear term preserves the definiteness of $\bar{\mathbf{D}}_C(n-1)$. From (3-15), this causes $\bar{\mu}_q(n)$ to become less than $\mu(n)$, causing the shift term to also be positive definite. Thus, all four terms in (3-28) become either positive definite or positive semi-definite, causing $\bar{\mathbf{D}}_C(n-1)$ to be positive definite.

Second, the terms in (3-28) become significant when small word lengths are used. Assuming roundoff errors are uniformly distributed

between $-\Delta/2$ and $\Delta/2$, then σ_r^2 equals $\Delta^2/12$. If b bits are used to represent fractions in the fixed point arithmetic, then $\Delta = 2^{-b}$ and:

$$\sigma_r^2 = \frac{2^{-2b}}{12} \tag{3-32}$$

Thus, if b is small, then σ_r^2 in the roundoff driving term in (3-28) is large. This causes the linear and quadratic terms on subsequent iterations to be large as well.

Finally, the quadratic term in (3-28) can become significant when λ_{max} is large, as discussed in section 3.2. Thus, the three conditions under which loss of positive definiteness occurs are satisfied when the fixed point precision is low and the maximum eigenvalue of $\mathbf{R}_{xx}$ is large.

3.3.2 Prior to explosive divergence

When the conditions above are met, eventual loss of positive definiteness is possible. Examination of equation (3-28) provides insight into how, over successive iterations, error accumulates to produce loss of positive definiteness.

First, consider the fact that $\bar{\mathbf{D}}_C(n-1)$ is biased to becoming positive definite. From (3-30), this causes $\bar{\mu}_q(n)$ to be less than $\mu(n)$. As a result, the coefficients of the four terms in (3-28) change. The coefficient of the linear term actually becomes smaller, providing more damping of previous error. However, the coefficients of the remaining three terms in (3-28) become larger, magnifying their effect on the total error. Under the conditions for loss of positive

definiteness, these three terms can dominate the first term, causing the entries in $\bar{D}_C(n-1)$ to grow from iteration to iteration. This results in $\bar{\mu}_q(n)$ becoming smaller and smaller with each iteration, eventually becoming negative.

3.3.3 Explosive divergence

When $\bar{\mu}_q(n)$ becomes negative, it is clear from (3-31) that $\bar{C}_q(n-1)$ is no longer positive definite. At this time, the coefficients of the three terms in (3-28) driving $\bar{D}_C(n)$ positive definite continue to grow larger. When $\bar{\mu}_q(n)$ becomes less than $-\alpha$, then the nature of the error terms in (3-28) changes significantly.

When $\bar{\mu}_q(n)$ becomes less than $-\alpha$, then the term $\alpha + \bar{\mu}_q(n)$ becomes negative. First, this causes the linear term's multiplier to become suddenly greater than one. However, if the other three error terms have been dominating this term, then this should not be significant. Second, the large quadratic term becomes suddenly negative semi-definite. Thirdly, both the shift and roundoff terms become suddenly negative definite. Thus, the three terms which were driving $\bar{D}_C(n)$ positive definite suddenly change to driving the error negative definite. If these three terms dominate the linear term, then the resulting error matrix is suddenly negative definite. Thus, if $\alpha + \bar{\mu}_q(n)$ becomes negative at iteration n_1, then $\bar{D}_C(n_1)$ would be suddenly negative definite.

Now consider the next iteration, $n_1 + 1$. With $\bar{D}_C(n_1)$ negative definite, $\bar{\mu}_q(n_1 + 1)$ would suddenly be positive and greater than $\mu(n)$,

possibly much greater than $\mu(n)$. In this case, the linear term would dampen $\bar{D}_C(n_1)$ term slightly and remain negative definite. The other three terms would return to being positive definite or semi-definite, canceling out some of the negative definiteness of the linear term. In subsequent iterations, the linear term would dampen the negative definite error, and the other three terms would continue to drive $\bar{D}_C(n)$ positive definite. As a result, the whole explosive divergence scenario just described would eventually re-occur.

This completes the explosive divergence scenario with regards to the C matrix. Now consider the behavior of the angle parameter $\gamma_q(n)$ during this scenario. Since $\mu_q(n) \cong \bar{\mu}_q(n)$, the conditional expected value notation has been dropped. As $\mu_q(n)$ becomes small, $\gamma_q(n)$ grows in value. When $\mu_q(n)$ becomes negative, considered the start of explosive divergence, then $\gamma_q(n)$ exceeds one. This is exactly what happens when the fast RLS algorithms diverge. While this behavior has not been pointed out until very recently [BELL 87b], it is apparent from published plots of $\gamma_q(n)$ showing explosive divergence [FABR 86]. The scenario continues with $\mu_q(n)$ decreasing in value until $\alpha + \mu_q(n)$ is negative. This results in $\gamma_q(n)$ becoming suddenly negative. Again, this agrees exactly with previously published observations regarding the fast RLS algorithms [CIOF 84, LIN 84, FABR 86]. Finally, the algorithm recovers on the next iteration with $\mu_q(n)$ returning to a positive value. This results in a positive $\gamma_q(n)$ less than one.

Next, consider the behavior of the weight error described by (3-21). Again, the conditional expected value notation has been

dropped. Initially, when $D_C(n)$ is small, the linear term dominates (3-21). As a result, the weight error is damped out. As $D_C(n-1)$ grows positive definite, the interaction term becomes significant. This significance grows as $\mu_q(n)$ grows smaller, eventually dominating the linear term and causing weight divergence. When $\alpha + \mu_q(n)$ becomes negative, the interaction term suddenly switches sign. At the same time, the linear term suddenly becomes unstable. On the next iteration, with $\mu_q(n)$ large and positive, the interaction term becomes smaller and the linear term becomes stable.

3.3.4 Sensitivity considerations

The likelihood of explosive divergence is affected by signal statistics, the precision, and the forgetting factor. First, one of the conditions contributing to the start of explosive divergence is when the maximum eigenvalue of $\mathbf{R}_{xx}$, denoted λ_{max}, is large. The larger λ_{max} is, the larger the entries in the quadratic term of (3-28). Also, the larger λ_{max}, the smaller the error matrix elements need to be for $\mu_q(n)$ to become negative. Thus, the likelihood of explosive divergence is related to the maximum eigenvalue of the data autocorrelation matrix. If the minimum eigenvalue is small, then this condition is equivalent to ill-conditioning or near singularity, which has been linked to finite precision problems [CIOF 87].

Second, the amount of precision used determines the relative size of all errors, especially the roundoff term in (3-28). The less precision used, the larger the roundoff variance σ_r^2 in (3-28) and the

more likely divergence will occur.

Finally, the forgetting factor α can affect the likelihood of divergence. First, if α is very small, then the effective window of data used to form $\mathbf{C}(n)$ is smaller. This can cause $\mathbf{C}(n)$ to deviate significantly from the statistical value of $(1-\alpha)\ \mathbf{R}_{xx}^{-1}$, leading to the possibility of an even lower minimum eigenvalue (effectively a larger λ_{max}). On the other hand, as α becomes very close to one, the elements in $\mathbf{C}(n)$, and hence $\mathbf{C}_q(n)$, become small. This allows roundoff errors to have more impact on the positive definiteness of the matrix. If $\mathbf{C}_q(n)$ is normalized in some way (as is the case in floating point), then making α close to one should make explosive divergence less likely. This agrees with observations made concerning the fast RLS algorithms [CIOF 84, FABR 85, FABR 86]. If α is chosen to be unity, then $\mathbf{C}(n)$ will tend towards zero and either weight lock-up or weight error growth will be a problem [ARDA 87].

3.4 Simulation results

To test the accuracy of the divergence scenario developed in section 3.3, a fixed point simulation capability was developed. The simulation software was written in "C" language on an AT&T PC 6300. All fixed point operations are performed with integer arithmetic using 16-bit storage and 32-bit accumulation. The rounding and quantization routines are based on those in [ARDA 87]. However, unlike the simulation results in [ARDA 87], *all* filter quantities were computed and propagated in fixed point arithmetic.

For this example, the one-step linear prediction application was simulated. The data were generated by passing a white zero-mean Gaussian sequence v(n) through a deterministic one-pole filter to generate a first-order auto-regressive (AR) process. Mathematically:

$$z(n) = 0 \qquad n < 0 \tag{3-33}$$

$$z(n) = a_0\, z(n-1) + v(n) \qquad n \geq 0 \tag{3-34}$$

where a_0 is the first order AR coefficient. The resulting signal was then quantized to fixed point values for processing by a fixed point CRLS filter:

$$d(n) = Q\,[\, z(n)\,] \tag{3-35}$$

$$x(n) = Q\,[\, z(n-1)\,] \tag{3-36}$$

The first-order AR coefficient was set to 0.93, and the white Gaussian driving process standard deviation was set at unity, yielding a theoretical data variance of 7.4019.

The CRLS filter was implemented entirely in fixed point arithmetic. The CRLS parameters were set at δ = 13.51 (based on standard guidelines [COWA 85, BELL 87b]) and α = 0.98 (represented as 0.98046875). In both data quantization and the CRLS filter, 8 of the 16 data bits were used to represent fractions. Two's complement representation and arithmetic were employed. While saturation logic was available in case of overflow, overflow never occurred.

Two quantities were monitored: the angle parameter $\gamma_q(n)$ and the normalized weight error [LIN 84] $e_w(n)$. The angle parameter for the fixed point algorithm, $\gamma_q(n)$, is defined similarly to $\gamma(n)$ in (2-3) with $\mu(n)$ replaced by $\mu_q(n)$. The normalized weight error is given by:

$$e_w(n) = 10 \log_{10} \left(\frac{\sum_{i=0}^{N-1} \left[w_i(n) - a_i \right]^2 + \sum_{i=N}^{M-1} a_i^2}{\sum_{i=0}^{M-1} a_i^2} \right) \text{ (dB)} \qquad (3\text{-}37)$$

where $w_i(n)$ are the elements of the CRLS weight vector, N is the order of the CRLS filter, M is the order of the true process, and a_i are the true parameters.

Figures 3-1 and 3-2 show plots of $\gamma_q(n)$ and $e_w(n)$ for the fixed point simulation. Explosive divergence occurred twice in the first 300 iterations. The scenario predicted in the previous section is clearly evident. In Figure 3-1, at the time of divergence, $\gamma_q(n)$ exceeds one, becomes negative, then returns to a value between 0 and 1. At the same time, the weight error grows significantly.

3.5 Conclusion

From the results of this chapter, several conclusions can be made. First, under reasonable assumptions, analytical results explaining the explosive divergence phenomenon of the CRLS filter can be obtained. These results accurately predict the behavior of the filter during divergence. They also indicate that divergence is most likely when the maximum eigenvalue of the data autocorrelation matrix is large, the forgetting factor is small, and the fixed point word length used is small. Finally, experimental results support the analytical results obtained.

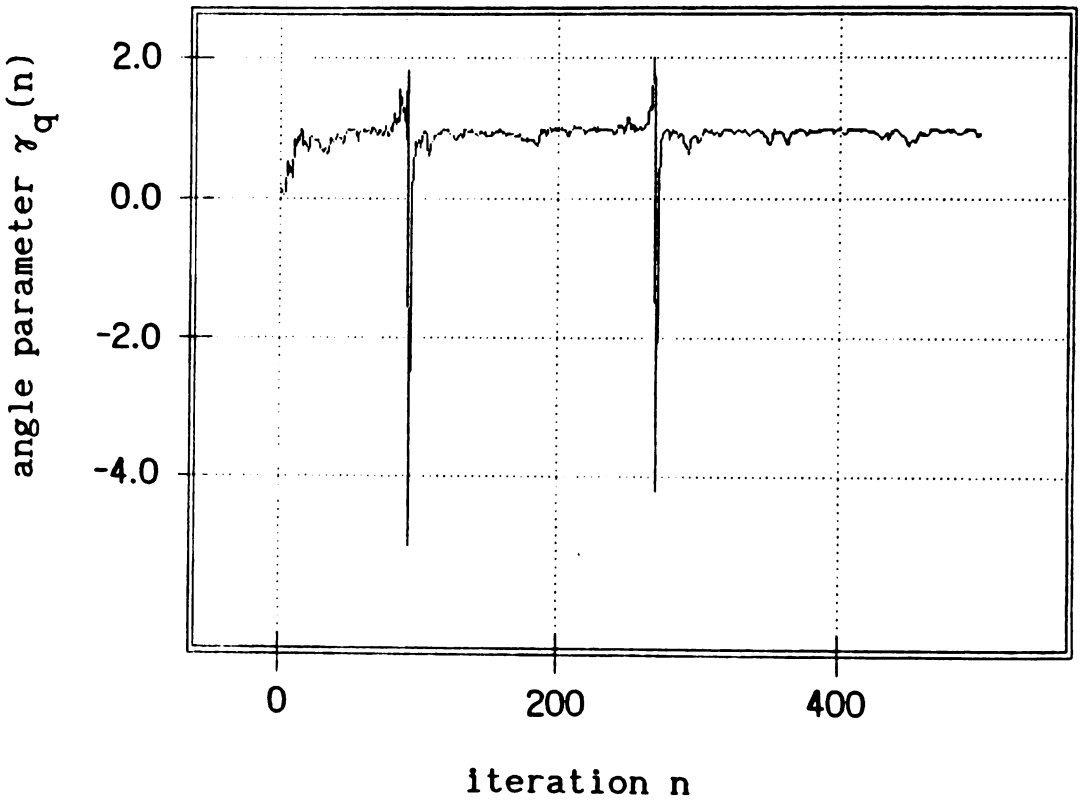

Figure 3-1. Fixed point simulation example of explosive divergence, plotting the angle parameter $\gamma_q(n)$ as a function of iteration n.

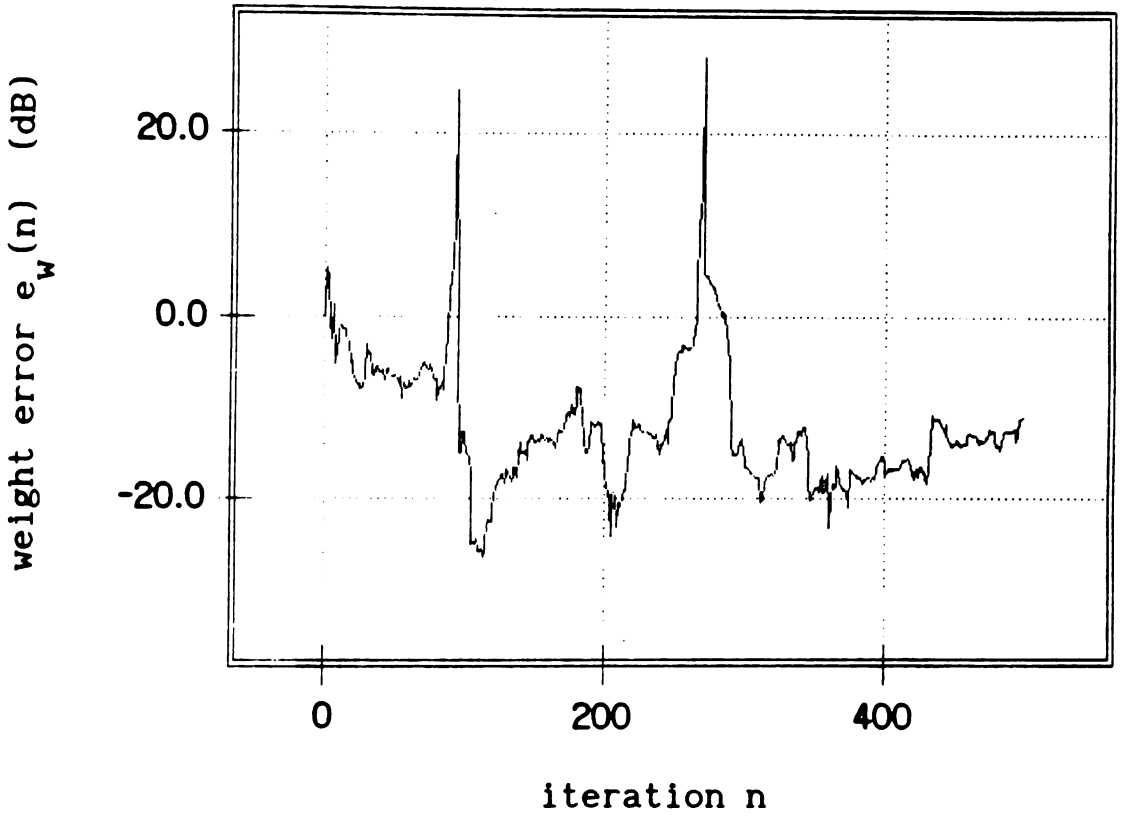

Figure 3-2. Fixed point simulation example of explosive divergence, plotting the weight error $e_w(n)$ as a function of iteration n.

CHAPTER 4 - PREVENTING EXPLOSIVE DIVERGENCE

The goal of this chapter is to prevent explosive divergence in a practical fixed point implementation of the CRLS algorithm. The analytical approach and results of Chapter 3 are used to develop techniques which prevent explosive divergence by preserving the positive definite nature of the $\mathbf{C}_q(n)$ matrix.

The first step is to select a practical fixed point implementation of the CRLS algorithm, considering those implementations which guarantee symmetry in the $\mathbf{C}_q(n)$ matrix. Second, the particular implementation selected is analyzed and shown to possess the same propagation characteristics as the original implementation studied in Chapter 3. Simulation results are used to support the analytical conclusion that symmetry preserving algorithms also suffer from explosive divergence. Next, the analytical results are used to develop techniques for preventing explosive divergence in the particular implementation selected. Finally, simulation results are used to demonstrate the effectiveness of the techniques developed.

4.1 Algorithm selection

In this section, a practical implementation of the CRLS algorithm is selected and specified for implementation. First, a new symmetry preserving form of the CRLS algorithm is chosen for implementation. Second, an intermediate quantity is introduced into the algorithm to replace costly divisions by multiplications. Finally, a complete specification of the implementation is given.

4.1.1 Selection of CRLS implementation

The form of the CRLS algorithm analyzed in Chapter 3 is incapable of preserving symmetry in the $\mathbf{C}_q(n)$ matrix. Since a) symmetry is preserved in the infinite precision algorithm and b) loss of symmetry has been blamed for loss of positive definiteness [FABR 85, VERH 89], two forms of the CRLS algorithm which preserve symmetry in $\mathbf{C}_q(n)$ are considered. One form is selected because it introduces fewer errors in the $\mathbf{C}_q(n)$ update equation than the other form.

First, in infinite precision, the $\mathbf{C}(n)$ matrix in the CRLS algorithm has the property of being symmetric for all iterations [HAYK 86]. However, when implemented in finite precision, the form specified in (1-7) of Chapter 1 does not guarantee this property. This is because the term $\mathbf{g}(n)\mathbf{x}^T(n)\mathbf{C}(n-1)$ in (1-7e) is not necessarily symmetric in finite precision. In the analysis of Chapter 3, asymmetry in $\mathbf{C}_q(n)$ appears as asymmetric error terms in (3-26).

By simply re-arranging the terms in (1-7e), the following symmetry preserving form is obtained [BOTT 89]:

$$\mathbf{C}(n) = 1/\alpha \left[\mathbf{C}(n-1) - \frac{\left(\mathbf{p}(n)\ \mathbf{p}^T(n) \right)}{\alpha + \mu(n)} \right] \tag{4-1}$$

$$\mathbf{p}(n) = \mathbf{C}(n-1)\ \mathbf{x}(n) \tag{4-2}$$

Another symmetry preserving form was developed by Fabre and Gueguen [FABR 85]:

$$\mathbf{C}(n) = 1/\alpha \left[\mathbf{C}(n-1) - (\alpha + \mu(n)) \left(\mathbf{g}(n)\ \mathbf{g}^T(n) \right) \right] \tag{4-3}$$

with $\mathbf{g}(n)$ given in (1-7c). Both forms preserve symmetry by first performing an outer product of two vectors, which is symmetric even in finite precision. The outer product is then scaled by a scalar, which preserves symmetry even in finite precision.

The form in (4-1) is selected over the form in (4-3) because the form in (4-1) introduces fewer roundoff error terms into the $\mathbf{C}(n)$ update equation. The outer product in (4-1) is created using $\mathbf{p}(n)$, which is formed using only one multiplication. In contrast, the outer product in (4-3) is created using $\mathbf{g}(n)$, which is formed using one multiplication, one division, and several operations to form $\mu(n)$. Thus, using the analytical approach in Chapter 3, the form in (4-1) introduces fewer error terms when updating $\mathbf{C}(n)$ than the form in (4-3). Therefore, the form given in (4-1) was selected for implementation.

4.1.2 Reduction of the number of divisions performed

In current practical hardware implementations, performing a division requires significantly more clock cycles than a multiplication. In this sub-section, the number of divisions in the CRLS form selected in section 4.1.1 is reduced by introducing an intermediate quantity into the algorithm.

If (4-1) is implemented exactly as specified, the outer product matrix is divided by the scalar term $\alpha + \mu(n)$. This requires N^2 divisions, where N is the order of the CRLS filter. To avoid such a large number of divisions, the following modified form is proposed:

$$\beta(n) = 1/(\alpha + \mu(n)) \tag{4-4}$$

$$\mathbf{C}(n) = 1/\alpha \left[\mathbf{C}(n-1) - \beta(n) \left(\mathbf{p}(n)\, \mathbf{p}^T(n) \right) \right] \tag{4-5}$$

where $\beta(n)$ is an intermediate scalar quantity. By using (4-4) and (4-5), the N^2 divisions are replaced by 1 division and N^2 multiplications. Unfortunately, additional roundoff error is introduced when forming $\beta(n)$. The intermediate $\beta(n)$ term can also be used when computing the Kalman gain $\mathbf{g}(n)$ for updating the weight vector $\mathbf{w}(n)$.

Finally, the division by α in (4-5) can be replaced by a multiplication by $1/\alpha$. The term $1/\alpha$ can be pre-computed and stored before filtering begins. This converts N^2 divisions into N^2 multiplications.

4.1.3 Implementation specification

In this sub-section, the form of the CRLS algorithm selected in section 4.1.1, with the modifications suggested in section 4.1.2, is completely specified. First, using the notation of Chapter 3, a fixed point implementation of the symmetry preserving CRLS algorithm in (4-2), (4-4) and (4-5) is given by:

Initial conditions:

$$\mathbf{w}_q(0) = \mathbf{x}(0) = \mathbf{0} \tag{4-6a}$$

$$\mathbf{C}_q(0) = \delta\, \mathbf{I} \qquad (\delta >> 1) \tag{4-6b}$$

For n = 1, 2, ... **do:**

$$e_q(n|n-1) = d(n) - Q[\, \mathbf{x}^T(n)\, \mathbf{w}_q(n-1)\,] \tag{4-7a}$$

$$\mathbf{p}_q(n) = Q[\, \mathbf{C}_q(n-1)\, \mathbf{x}(n)\,] \tag{4-7b}$$

$$\mu_q(n) = Q[\, \mathbf{x}^T(n)\, \mathbf{p}_q(n)\,] \tag{4-7c}$$

$$\beta_q(n) = Q\left[\frac{1}{\alpha + \mu_q(n)} \right] \tag{4-7d}$$

$$\mathbf{g}_q(n) = Q[\, \beta_q(n)\, \mathbf{p}_q(n)\,] \tag{4-7e}$$

$$\mathbf{w}_q(n) = \mathbf{w}_q(n-1) + Q[\, \mathbf{g}_q(n)\, e_q(n|n-1)\,] \tag{4-7f}$$

$$\mathbf{C}_q(n) = Q\Big[(1/\alpha) \Big(\mathbf{C}_q(n-1) - Q\Big[\beta_q(n)\, Q\Big[\mathbf{p}_q(n)\, \mathbf{p}_q^T(n) \Big] \Big] \Big) \Big] \tag{4-7g}$$

An equivalent specification, defining all intermediate terms, is given below. This form will be useful when computing operations counts as well as when specifying stabilization techniques.

Initial conditions:

$$\mathbf{w}_q(0) = \mathbf{x}_q(0) = \mathbf{0} \tag{4-8a}$$

$$\mathbf{C}_q(0) = \delta\, \mathbf{I} \qquad (\delta \gg 1) \tag{4-8b}$$

For n = 1, 2, ... **do:**

$$\hat{d}_q(n) = Q\Big[\mathbf{x}_q^T(n)\, \mathbf{w}_q(n-1) \Big] \tag{4-9a}$$

$$e_q(n|n-1) = d_q(n) - \hat{d}_q(n) \tag{4-9b}$$

$$\mathbf{p}_q(n) = Q\Big[\mathbf{C}_q(n-1)\, \mathbf{x}(n) \Big] \tag{4-9c}$$

$$\mu_q(n) = Q\Big[\mathbf{x}^T(n)\, \mathbf{p}_q(n) \Big] \tag{4-9d}$$

$$\xi_q(n) = \alpha + \mu_q(n) \tag{4-9e}$$

$$\beta_q(n) = Q\left[1/\xi_q(n) \right] \tag{4-9f}$$

$$\mathbf{g}_q(n) = Q\left[\beta_q(n)\ \mathbf{p}_q(n) \right] \tag{4-9g}$$

$$\mathbf{f}_q(n) = Q\left[\mathbf{g}_q(n)\ e_q(n|n-1) \right] \tag{4-9h}$$

$$\mathbf{w}_q(n) = \mathbf{w}_q(n-1) + \mathbf{f}_q(n) \tag{4-9i}$$

$$\mathbf{H}_q(n) = Q\left[\mathbf{p}_q(n)\ \mathbf{p}_q^T(n) \right] \tag{4-9j}$$

$$\mathbf{J}_q(n) = Q\left[\beta_q(n)\ \mathbf{H}_q(n) \right] \tag{4-9k}$$

$$\mathbf{M}_q(n) = \mathbf{C}_q(n-1) - \mathbf{J}_q(n) \tag{4-9l}$$

$$\mathbf{C}_q(n) = Q\left[(1/\alpha)\ \mathbf{M}_q(n) \right] \tag{4-9m}$$

Note that $1/\alpha$ is pre-computed only once, then used in (4-9m) at each iteration.

The equations above can be used to compute an operations count per iteration. Since symmetry is preserved, only the lower triangular elements in (4-9j) through (4-9m) need be calculated. This reduces the number of elements computed from N^2 to $N + 0.5\ (N^2 - N)$ or simply $0.5\ (N^2 + N)$. The total operations count is $2.5\ N^2 + 5.5\ N$ multiplications, 1 division, and $1.5\ N^2 + 2.5\ N$ additions/subtractions. For large N, the total number of operations is less than those quoted in [ALEX 86] because of the computational savings described in section 4.1.2.

Equations (4-6) and (4-7), or equivalently equations (4-8) and (4-9), completely specify the efficient symmetry preserving algorithm selected for implementation. This form will be referred to as SCRLS.

4.2 Analysis of algorithm selected

In this section, the SCRLS algorithm in section 4.1 is analyzed using the approach detailed in Chapter 3. The propagation of error results are shown to be identical to those for the implementation studied in Chapter 3. A similar analysis of the other symmetry preserving algorithm considered in section 4.1, though not presented, also yields identical error propagation characteristics.

The analysis proceeds as follows. First, error recursions are derived for each algorithmic quantity. Second, propagation analyses of the error recursions for the weight vector and **C** matrix are presented. It is shown that the SCRLS form has the same underlying error propagation mechanism as the form analyzed in Chapter 3.

4.2.1 SCRLS error recursions

Error recursion equations for the SCRLS algorithm are derived using the same approach detailed in Chapter 3 and Appendix A. The detailed derivations are presented in Appendix C. The resulting error recursions for the SCRLS algorithm are summarized below.

Initial conditions:

$$\mathbf{d}_w(0) = \mathbf{0}, \quad \mathbf{D}_C(0) = \mathbf{0} \tag{4-10}$$

For n = 1, 2, ... **do:**

$$d_e(n|n-1) = -r_{m1}(n) - \mathbf{x}^T(n)\,\mathbf{d}_w(n-1) \tag{4-11a}$$

$$\mathbf{d}_p(n) = \mathbf{r}_{m2}(n) + \mathbf{D}_C(n-1)\,\mathbf{x}(n) \tag{4-11b}$$

$$d_\mu(n) = \mathbf{x}^T(n)\,\mathbf{D}_C(n-1)\,\mathbf{x}(n) + \mathbf{x}^T(n)\,\mathbf{r}_{m2}(n) + r_{m3}(n) \tag{4-11c}$$

$$d_{\beta}(n) = r_{d1}(n) + \beta(n) - \frac{1}{\alpha + \mu_q(n)} \tag{4-11d}$$

$$\mathbf{d}_g(n) = \mathbf{r}_{m4}(n) + \mathbf{g}(n) - \beta_q(n)\, \mathbf{p}_q(n) \tag{4-11e}$$

$$\mathbf{d}_w(n) = \mathbf{d}_w(n-1) + \mathbf{r}_{m5}(n) + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1) \tag{4-11f}$$

$$\mathbf{D}_C(n) = \mathbf{R}_{m8}(n) + 1/\alpha\, [\, \mathbf{D}_C(n-1) - \mathbf{R}_{m7}(n) - \beta_q(n)\, \mathbf{R}_{m6}(n)\,] + 1/\alpha\, [\, \beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n)\,] \tag{4-11g}$$

where

$$r_{m1}(n) = r_m[\, \mathbf{x}^T(n),\, \mathbf{w}_q(n-1)\,] \tag{4-12a}$$

$$\mathbf{r}_{m2}(n) = r_m[\, \mathbf{C}_q(n-1),\, \mathbf{x}(n)\,] \tag{4-12b}$$

$$r_{m3}(n) = r_m[\, \mathbf{x}^T(n),\, \mathbf{p}_q(n)\,] \tag{4-12c}$$

$$r_{d1}(n) = r_d[\, 1,\, \alpha + \mu_q(n)\,] \tag{4-12d}$$

$$\mathbf{r}_{m4}(n) = r_m[\, \beta_q(n),\, \mathbf{p}_q(n)\,] \tag{4-12e}$$

$$\mathbf{r}_{m5}(n) = r_m[\, \mathbf{g}_q(n),\, e(n|n-1)\,] \tag{4-12f}$$

$$\mathbf{R}_{m6}(n) = r_m[\, \mathbf{p}_q(n),\, \mathbf{p}_q^T(n)\,] \tag{4-12g}$$

$$\mathbf{R}_{m7}(n) = r_m[\, \beta_q(n),\, Q[\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n)\,] \tag{4-12h}$$

$$\mathbf{R}_{m8}(n) = r_m[\, (1/\alpha), \{\, \mathbf{C}_q(n-1) - Q[\, \beta_q(n)\, Q[\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n)\,]\,]\}\,] \tag{4-12i}$$

As in the analysis in Chapter 3, accumulated error is passed from one iteration to the next through $\mathbf{d}_w(n-1)$ and $\mathbf{D}_C(n-1)$. Consequently, the

error recursions for these two error terms are studied in the next sub-section.

4.2.2 Propagation analysis

In this sub-section, the error recursions for the weight vector and **C** matrix errors are analyzed to determine how errors propagate from one iteration to the next. The analysis consists of the same three steps used in Chapter 3. The analysis of the weight error recursion is performed first.

The recursion for the weight error is given in (4-11f). First, the substitutions given in (3-10a) and (3-10b) are employed along with the following:

$$\beta_q(n) = \beta(n) - d_\beta(n) \tag{4-13}$$

$$\mathbf{p}_q(n) = \mathbf{p}(n) - \mathbf{d}_p(n) \tag{4-14}$$

Second, the expressions for $d_e(n|n-1)$, $\mathbf{d}_p(n)$, $d_\beta(n)$, and $\mathbf{d}_g(n)$ given in equations (4-11a), (4-11b), (4-11d), and (4-11e) respectively are used. After much algebraic manipulation, the following result is obtained:

$$\mathbf{d}_w(n) = \left[\mathbf{I} - \frac{1}{\alpha + \mu_q(n)} \left(\mathbf{C}(n-1) - \mathbf{D}_C(n-1) \right) \mathbf{x}(n)\, \mathbf{x}^T(n) \right] \mathbf{d}_w(n-1)$$

$$+ \left(1 - \frac{\alpha + \mu(n)}{\alpha + \mu_q(n)} \right) \mathbf{g}(n)\, e(n|n-1) + \frac{\mathbf{D}_C(n-1)\, \mathbf{x}(n)\, e(n|n-1)}{\alpha + \mu_q(n)}$$

$$+ \left(r_{m4}(n) + \frac{r_{m2}(n)}{\alpha + \mu_q(n)} \right) \left(e(n|n-1) + r_{m1}(n) + \mathbf{x}^T(n)\, \mathbf{d}_w(n-1) \right)$$

$$- \frac{1}{\alpha + \mu_q(n)} \left[\mathbf{C}(n-1) - \mathbf{D}_C(n-1) \right] \mathbf{x}(n)\, r_{m1}(n) + \mathbf{r}_{m5}(n)$$

$$+ r_{d1}(n) \left[\mathbf{C}(n-1) - \mathbf{D}_C(n-1) \right] \mathbf{x}(n) \left[e(n|n-1) + r_{m1}(n) + \mathbf{x}^T(n)\, \mathbf{d}_w(n-1) \right]$$

$$- r_{d1}(n)\, \mathbf{r}_{m2}(n) \left[e(n|n-1) + r_{m1}(n) + \mathbf{x}^T(n)\, \mathbf{d}_w(n-1) \right] \qquad (4\text{-}15)$$

where $\mu_q(n)$ is given in (3-12). Equation (4-15) contains two more terms than (3-11) because the SCRLS form introduces an intermediate quantity $\beta(n)$ into the weight update.

Next, using the same analytical approach to derive (3-14), the conditional expected value given in (3-13) is taken. The result is:

$$\bar{\mathbf{d}}_w(n) \cong \left[\mathbf{I} - \frac{1}{\alpha + \bar{\mu}_q(n)} \left[\mathbf{C}(n-1) - \bar{\mathbf{D}}_C(n-1) \right] \mathbf{x}(n)\, \mathbf{x}^T(n) \right] \bar{\mathbf{d}}_w(n-1)$$

$$+ \left[1 - \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \right] \mathbf{g}(n)\, e(n|n-1) + \frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} \bar{\mathbf{D}}_C(n-1) \frac{\mathbf{x}(n)\, e(n|n-1)}{\alpha + \mu(n)} \qquad (4\text{-}16)$$

which is *identical* to (3-14). Consequently, employing the assumption of steady-state would give (3-21). Thus, the average weight error propagation characteristic at steady-state for SCRLS is identical to the one for the original CRLS algorithm analyzed in Chapter 3.

Next, the recursion for the error in the **C** matrix given by (4-11g) is analyzed. First, (3-25a) and (3-25b) are used along with (4-2) and

(4-14). Second, the expressions derived for $\mathbf{d}_p(n)$ and $d_\beta(n)$ given by (4-11b) and (4-11d) respectively are employed. After some algebraic manipulation the following result is obtained:

$$\begin{aligned}
\mathbf{D}_C(n) &= 1/\alpha\ \mathbf{D}_C(n-1) + 1/\alpha\ \beta_q(n)\ \mathbf{D}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{D}_C(n-1) \\
&- 1/\alpha\ \beta_q(n) \left[\mathbf{C}(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{D}_C(n-1) + \mathbf{D}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) \right] \\
&+ 1/\alpha \left[\frac{\alpha + \mu(n)}{\alpha + \mu_q(n)} - 1 - (\alpha + \mu(n))\ r_{d1}(n) \right] \mathbf{g}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) \\
&+ 1/\alpha\ \beta_q(n)\ \mathbf{r}_{m2}(n)\ \mathbf{r}_{m2}^T(n) \\
&- 1/\alpha\ \beta_q(n) \left[\mathbf{r}_{m2}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) + \mathbf{C}(n-1)\ \mathbf{x}(n)\ \mathbf{r}_{m2}^T(n) \right] \\
&+ 1/\alpha\ \beta_q(n) \left[\mathbf{r}_{m2}(n)\ \mathbf{x}^T(n)\ \mathbf{D}_C(n-1) + \mathbf{D}_C(n-1)\ \mathbf{x}(n)\ \mathbf{r}_{m2}^T(n) \right] \\
&- 1/\alpha\ \beta_q(n)\ \mathbf{R}_{m6}(n) - 1/\alpha\ \mathbf{R}_{m7}(n) + \mathbf{R}_{m8}(n)
\end{aligned} \tag{4-17}$$

where

$$\beta_q(n) = \beta(n) - d_\beta(n) = \frac{1}{\alpha + \mu_q(n)} - r_{d1}(n) \tag{4-18}$$

and $\mu_q(n)$ is given in (3-12).

Taking the same type of conditional expected value given in (3-13) of (4-17) and (4-18) and combining the results gives:

$$\bar{\mathbf{D}}_C(n) \cong 1/\alpha\ \bar{\mathbf{D}}_C(n-1) + 1/\alpha \left[\frac{\bar{\mathbf{D}}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \bar{\mathbf{D}}_C(n-1)}{\alpha + \bar{\mu}_q(n)} \right]$$

$$- 1/\alpha \left(\frac{\mathbf{C}(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \bar{\mathbf{D}}_C(n-1) + \bar{\mathbf{D}}_C(n-1)\ \mathbf{x}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1)}{\alpha + \bar{\mu}_q(n)} \right)$$

$$+ 1/\alpha \left(\frac{\alpha + \mu(n)}{\alpha + \bar{\mu}_q(n)} - 1 \right) \mathbf{g}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) + 1/\alpha \left(\frac{\sigma_r^2\ \mathbf{I}}{\alpha + \bar{\mu}_q(n)} \right)$$

(4-19)

which is *identical* to (3-27). Consequently, employing the assumption of steady-state would give (3-28). Thus, the average **C** matrix error propagation characteristic at steady-state for SCRLS is also identical to the one for the original CRLS form analyzed in Chapter 3.

Thus, the results of analysis indicate that the SCRLS algorithm has the same underlying error propagation mechanism as the CRLS form analyzed in Chapter 3. Since the error propagation mechanisms are the same, the SCRLS form also suffers from explosive divergence, following the scenario described in Chapter 3. A similar analysis of the symmetry preserving form suggested by Fabre and Gueguen [FABR 85], though not presented here, indicates that it also has the error propagation characteristic described in Chapter 3.

Thus, preserving symmetry in the **C** matrix does not guarantee that positive definiteness is preserved. This conclusion appears to contradict earlier results [FABR 85, VERH 89]. However, these earlier results appear to be based on large word lengths (in excess of 16 bits) and well-conditioned data. In the next section, simulation results are presented which demonstrate explosive divergence for the symmetry preserving algorithms considered when the wordlength used is 16 bits.

4.3 Simulation results

The SCRLS form was simulated under the same conditions described in section 3.4. Figure 4-1 shows a plot of $\gamma_q(n)$ for the first 500 iterations. Clearly $\gamma_q(n)$ exceeds 1 and becomes negative several times. The original symmetry preserving algorithm, without the introduction of $\beta(n)$, was also simulated under the same conditions. Explosive divergence was still a problem. Finally, the symmetry preserving form in [FABR 85] was simulated under the same conditions. Again, the results, which are plotted in Figure 4-2, indicate that explosive divergence is a problem.

Thus, the simulation results support the analytical conclusion of the previous section that the symmetry preserving forms of CRLS still suffer from the explosive divergence problem.

4.4 Preventing explosive divergence

In this section, the analytical results of section 4.2 are used to develop techniques which prevent explosive divergence. By biasing the errors introduced into the $\mathbf{C}_q(n)$ matrix, positive definiteness can be preserved. Also, by making the biases as small as possible, least squares performance is preserved.

4.4.1 Stabilization concept

First, the analysis in Chapter 3 indicates that divergence occurs because $\mathbf{D}_C(n)$ becomes positive definite, which in turn causes the error terms in (3-28) to become larger. Thus, to prevent explosive

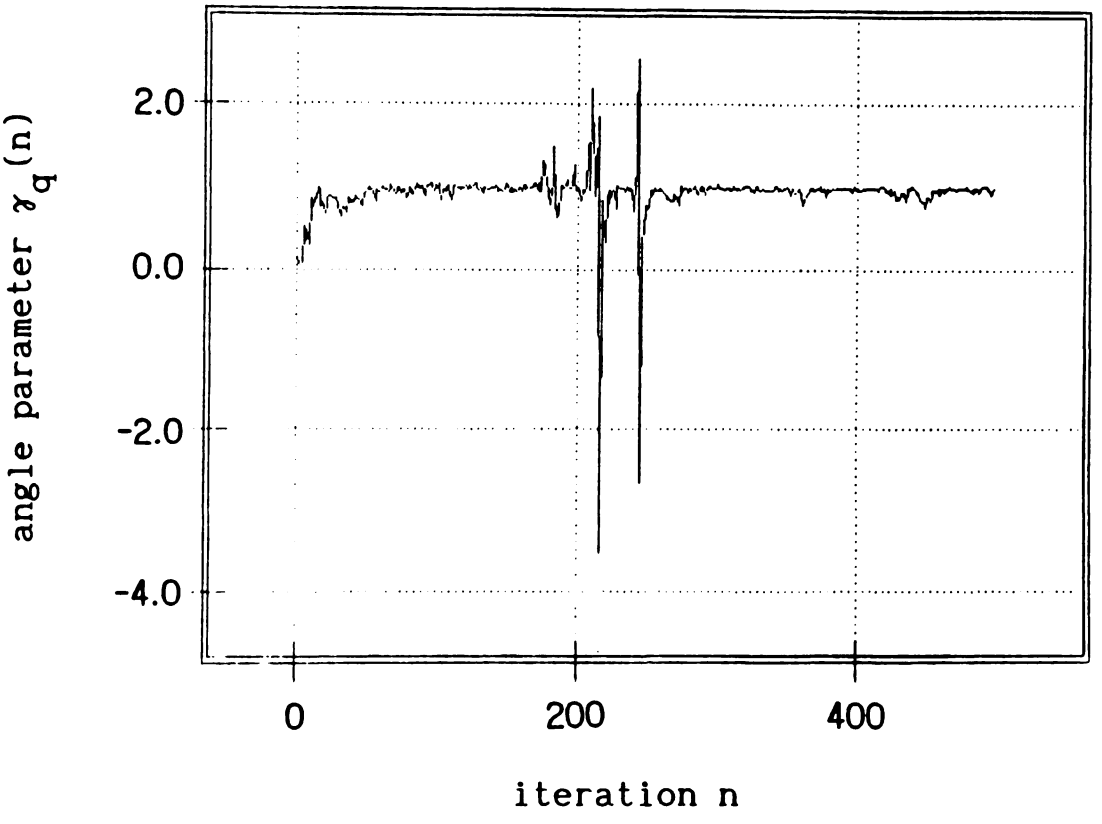

Figure 4-1. Fixed point simulation example of explosive divergence in the SCRLS algorithm. Plotted is the angle parameter $\gamma_q(n)$ as a function of iteration n.

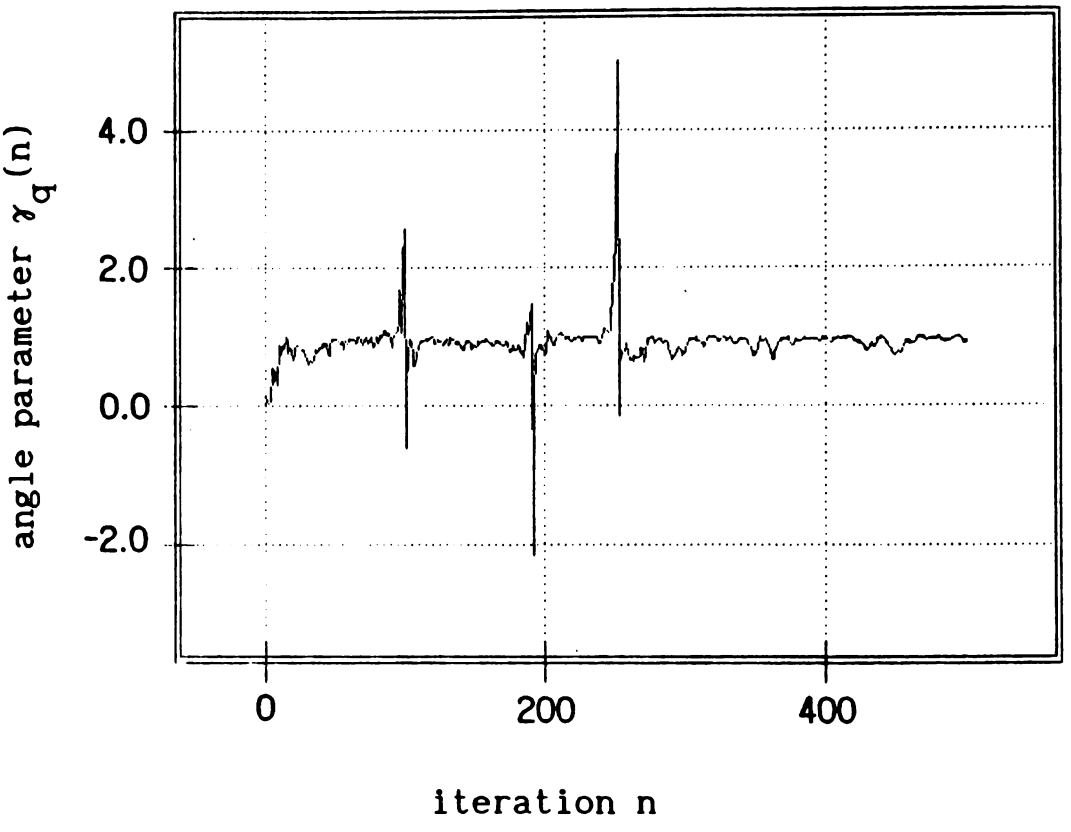

Figure 4-2. Fixed point simulation example of explosive divergence in the symmetry preserving algorithm of Fabre and Gueguen [FABR 85]. Plotted is the angle parameter $\gamma_q(n)$ as a function of iteration n.

divergence, $\mathbf{D}_C(n)$ must be biased negative definite. This implies that $\mathbf{C}_q(n)$ must be biased positive definite.

To determine how to bias $\mathbf{D}_C(n)$ to be negative definite, the driving terms in (4-17) are examined. Note that (4-17) is an exact result for the SCRLS form, derived from simple algebraic manipulation of the original error recursion in (4-11g). To concentrate on the driving error terms only, $\mathbf{D}_C(n-1)$ is set to zero. Using (4-2), this gives:

$$\tilde{\mathbf{D}}_C(n) = 1/\alpha \left(\frac{\alpha + \mu(n)}{\alpha + \tilde{\mu}_q(n)} - 1 - (\alpha + \mu(n))\, r_{d1}(n) \right) \frac{\mathbf{p}(n)\, \mathbf{p}^T(n)}{\alpha + \mu(n)} \qquad \text{(term 1)}$$

$$- 1/\alpha\, \tilde{\beta}_q(n) \left(\mathbf{r}_{m2}(n)\, \mathbf{p}^T(n) + \mathbf{p}(n)\, \mathbf{r}_{m2}^T(n) \right) \qquad \text{(term 2)}$$

$$+ 1/\alpha\, \tilde{\beta}_q(n)\, \mathbf{r}_{m2}(n)\, \mathbf{r}_{m2}^T(n) \qquad \text{(term 3)}$$

$$- 1/\alpha\, \tilde{\beta}_q(n)\, \mathbf{R}_{m6}(n) \qquad \text{(term 4)}$$

$$- 1/\alpha\, \mathbf{R}_{m7}(n) \qquad \text{(term 5)}$$

$$+ \mathbf{R}_{m8}(n) \qquad \text{(term 6)}$$

(4-20)

where the ~ is used to indicate that previously accumulated error has been set to zero and

$$\tilde{\mu}_q(n) = \mu(n) - \mathbf{x}^T(n)\, \mathbf{r}_{m2}(n) - r_{m3}(n) \qquad (4\text{-}21)$$

$$\tilde{\beta}_q(n) = \frac{1}{\alpha + \tilde{\mu}_q(n)} - r_{d1}(n) \qquad (4\text{-}22)$$

The expression in (4-20) is the driving error assuming no past accumulated error or negligible accumulated error. The premise of this sub-section is that by biasing the terms in (4-20) to be negative definite, the SCRLS algorithm can be biased towards stable performance. The six terms in (4-20) will be referred to as terms 1-6 as labeled. For each term, techniques are developed which bias the term to be negative definite. Each technique is made as simple as possible and the bias is made as small as possible so that implementation is straightforward and least squares performance is preserved.

4.4.2 Term 1

Since $1/\alpha\ \mathbf{p}(n)\mathbf{p}^T(n)/(\alpha + \mu(n))$ is always positive semi-definite, term 1 in (4-20) is biased negative definite by making the scalar in parentheses in term 1 of (4-20) negative. This can be done by a) making $r_{d1}(n)$ positive, and b) making $\tilde{\mu}_q(n) > \mu(n)$.

First, consider making $r_{d1}(n)$ positive. To make $r_{d1}(n)$ positive implies rounding down so that the quantized value is less than the unquantized value (refer to equation (3-2)). Thus, $r_{d1}(n)$ can be made positive (or zero if no rounding takes place) when $\beta_q(n)$ is computed using a rounding down operation. Since $\beta_q(n)$ is always positive, this can be done using truncation.

Second, consider making $\tilde{\mu}_q(n) > \mu(n)$. From (4-21) this implies a) making $r_{m3}(n)$ negative, and b) making $\mathbf{x}^T(n)\ \mathbf{r}_{m2}(n)$ negative. The first can be done by rounding up the computation of $\mu_q(n)$. The second can be done by rounding when forming $\mathbf{p}_q(n)$ in such a way that the

roundoff error has the opposite sign of the corresponding term in the $\mathbf{x}(n)$ vector. Denoting $x_i(n)$ and $p_{qi}(n)$ as the ith components of $\mathbf{x}(n)$ and $\mathbf{p}_q(n)$ respectively, then the rounding rule for biasing $\mathbf{x}^T(n)\,\mathbf{r}_{m2}(n)$ negative is as follows:

if $x_i(n) > 0$, round up when forming $p_{qi}(n)$

if $x_i(n) < 0$, round down when forming $p_{qi}(n)$

Note that if $x_i(n) = 0$, then no rounding occurs. One disadvantage of this approach is that the bias introduced is a function of N, the order of the filter. The higher the order, the more bias introduced. Making $\tilde{\mu}_q(n) > \mu(n)$ also reduces the magnitude of roundoff error terms 2 through 4 in (4-20).

4.4.3 Term 2

Since $\tilde{\beta}_q(n)$ is always positive, the second term in (4-20) can be biased to be negative definite by biasing the matrices $\mathbf{r}_{m2}(n)\,\mathbf{p}^T(n)$ and $\mathbf{p}(n)\,\mathbf{r}_{m2}^T(n)$ to be positive definite. Since the trace of a matrix is the sum of its eigenvalues, a matrix can be biased to have positive eigenvalues (positive definite) by biasing the diagonal elements to be positive. Denoting $r_{m2i}(n)$ and $p_i(n)$ as the ith components of $\mathbf{r}_{m2}(n)$ and $\mathbf{p}(n)$, a positive bias on the diagonals of the two matrices can be achieved by forcing $r_{m2i}(n)$ to have the same sign of $p_i(n)$. Assuming that the finite precision $p_{qi}(n)$ has the same sign as the infinite precision $p_i(n)$, the bias can be created using the following rule:

if $p_{qi}(n) > 0$, then round down when forming $p_{qi}(n)$

if $p_{qi}(n) < 0$, then round up when forming $p_{qi}(n)$

Note that if $p_{qi}(n) = 0$, no rounding occurs. This technique can be termed "rounding in" since the rounded value is smaller in magnitude than the original value.

Observe that this technique may conflict with the technique given in the previous sub-section for forming $p_{qi}(n)$. This conflict can be avoided by forming two $\mathbf{p}_q(n)$ vectors. The first vector, denoted $\mathbf{p}_{\mu q}(n)$, is formed according to the rule of the previous sub-section and is used only when computing $\mu_q(n)$. The second vector, denoted $\mathbf{p}_{Hq}(n)$, is formed according to the rule given in this sub-section and is used when forming $\mathbf{H}_q(n)$ given in (4-9j).

While this technique produces a desired diagonal bias, larger errors are introduced on the off-diagonal when forming the off-diagonal entries of $\mathbf{H}_q(n)$. To prevent this problem, $\mathbf{p}_{Hq}(n)$ is only used when forming the diagonal entries of $\mathbf{H}_q(n)$. The off-diagonal entries are forming using $\mathbf{p}_q(n)$, which is formed using conventional rounding.

4.4.4 Term 3

Unfortunately, since α and $\tilde{\beta}_q(n)$ are always positive and $\mathbf{r}_{m2}(n)\mathbf{r}_{m2}{}^T(n)$ is always positive semi-definite, term 3 in (4-20) cannot be biased negative definite. It is possible to form $\mathbf{H}_q(n)$ in (4-9j) using two *different* $\mathbf{p}_q(n)$ vectors, denoted $\mathbf{p}_{qa}(n)$ and $\mathbf{p}_{qb}(n)$. The vectors would be computed such that the roundoff error vectors, denoted $\mathbf{r}_{m2a}(n)$ and $\mathbf{r}_{m2b}(n)$, had opposite signs. This would change term 3 to $1/\alpha\ \tilde{\beta}_q(n)\mathbf{r}_{m2a}(n)\mathbf{r}_{m2b}{}^T(n)$, which would be biased negative definite.

However, the property of symmetry in $\mathbf{C}_q(n)$ would be lost.

A better approach is to introduce an new additive term which will always compensate for term 3. Consider the new term, denoted term a, given by:

$$\text{term a} = -1/\alpha\ \tilde{\beta}_q(n)\ \Delta^2\ \mathbf{I} \tag{4-23}$$

where Δ is the fixed point step size for representing $\mathbf{p}_q(n)$. Combining this term with term 3 gives:

$$\text{term 3 + term a} = 1/\alpha\ \tilde{\beta}_q(n) \left(\mathbf{r}_{m2}(n)\ \mathbf{r}_{m2}^{T}(n) - \Delta^2\ \mathbf{I} \right)$$

$$= 1/\alpha\ \tilde{\beta}_q(n)\ \mathbf{A}(n) \tag{4-24}$$

where

$$\mathbf{A}(n) = \mathbf{r}_{m2}(n)\ \mathbf{r}_{m2}^{T}(n) - \Delta^2\ \mathbf{I} \tag{4-25}$$

Thus, to bias the combined term negative definite, the diagonal elements of $\mathbf{A}(n)$ should be biased to be negative. If $\mathbf{r}_{m2}(n)$ is formed by rounding in, as described in section 4.4.3, then the elements of $\mathbf{r}_{m2}(n)$ satisfy:

$$-\Delta < r_{m2i}(n) < \Delta \tag{4-26}$$

Thus, the diagonal elements of $\mathbf{A}(n)$ are biased to be negative as desired.

This technique is implemented by subtracting "1" from the least significant bit of the accumulator for the diagonal elements of $\mathbf{H}_q(n)$. However, if the accumulator is zero, then this subtraction is not necessary, since theoretically the diagonal elements of $\mathbf{H}_q(n)$ should always be non-negative.

4.4.5 Term 4

This term can be biased negative definite by biasing the diagonal elements of $\mathbf{R}_{m6}(n)$ to be positive. These values can be biased positive by rounding down the diagonal elements of $\mathbf{H}_q(n)$. Since these elements are always non-negative, this can be accomplished by simple truncation.

4.4.6 Term 5

This term is biased negative definite by biasing the diagonal elements of $\mathbf{R}_{m7}(n)$ to be positive. This can be done by rounding down or truncating the diagonal elements of $\mathbf{J}_q(n)$ in (4-9k).

4.4.7 Term 6

Finally, this term can be biased negative definite by biasing the diagonal elements of $\mathbf{R}_{m8}(n)$ to be negative. This implies rounding up when forming the diagonal elements of $\mathbf{C}_q(n)$.

4.4.8 Summary

Based on the analysis of section 4.2, techniques were derived to stabilize the SCRLS algorithm. The techniques are identified using the label "a.b", where "a" indicates which term in equation (4-20) is being addressed and "b" is used to enumerate multiple techniques. The techniques are summarized below.

1.1 Round down or truncate when forming $\beta_q(n)$.

1.2 Round up when forming $\mu_q(n)$.

1.3 When forming $\mathbf{p}_q(n)$ to compute $\mu_q(n)$, denoted $\mathbf{p}_{\mu q}(n)$, use:

if $x_i(n) > 0$, round up

if $x_i(n) < 0$, round down

2.1 When forming $\mathbf{p}_q(n)$ to compute the diagonal elements of $\mathbf{H}_q(n)$, denoted $\mathbf{p}_{Hq}(n)$, round in (round down in magnitude).

3.1 When forming the diagonal elements of $\mathbf{H}_q(n)$, subtract "1" from the accumulator if the accumulator is nonzero.

4.1 When forming $\mathbf{H}_q(n)$, round down on the diagonal.

5.1 When forming $\mathbf{J}_q(n)$ in (4-9k), round down on the diagonal.

6.1 When forming $\mathbf{C}_q(n)$, round up on the diagonal.

Preserving positive definiteness in $\mathbf{C}_q(n)$ should prevent the problem of loss of nonsingularity during initialization [MEND 87]. Also, technique 6.1 prevents the problem of loss of adaptivity on the diagonal of $\mathbf{C}_q(n)$ [BOTT 89].

4.5 Simulation of stabilization techniques

The effectiveness of the stabilization techniques developed in section 4.4 is demonstrated in this section through simulation. The stabilization techniques were incorporated into the SCRLS algorithm previously described. Under the same conditions described in section 3.4, the modified SCRLS algorithm was simulated. The result, shown in Figure 4-3, indicates that explosive divergence is not a problem. Figure 4-3 shows a plot of the angle parameter $\gamma_q(n)$, which remains in the range (0,1] as desired.

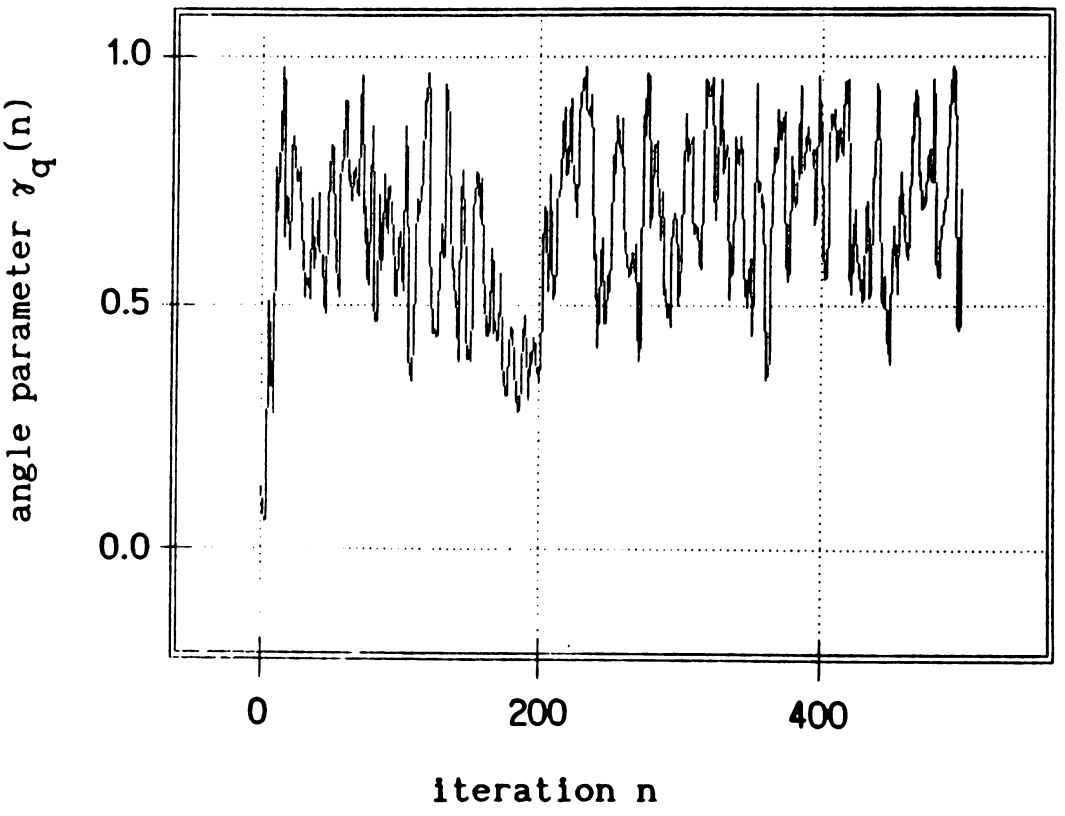

Figure 4-3. Fixed point simulation example of stable performance in the modified SCRLS algorithm. Plotted is the angle parameter $\gamma_q(n)$ as a function of iteration n.

Unfortunately, the problem of explosive divergence is not yet completely solved. Implicit in the development of the stabilization techniques is the assumption that overflow does not occur. If overflow occurs in forming $\mu_q(n)$, then techniques 1.1 - 1.3 are ineffective in biasing $\mu_q(n) > \mu(n)$. Also, if overflow occurs in the final scaling by $1/\alpha$ of the diagonal elements of $\mathbf{C}_q(n)$, then 6.1 is ineffective. Under similar simulation conditions, the modified SCRLS filter diverged shortly after overflow in $\mathbf{C}_q(n)$.

Underflow can also be a problem. If all the elements in $\mathbf{C}_q(n)$ underflow to zero, then the filter "locks up," and the weights cease to adapt (weight lock-up). In fact, if $e_q(n|n-1)$, $\mathbf{p}_q(n)$, $\beta_q(n)$, $\mathbf{g}_q(n)$, or $\mathbf{f}_q(n)$, given in equations (4-9b), (4-9c), (4-9f), (4-9g), and (4-9h) respectively, underflow, then weight lock-up will occur.

Thus, underflow and overflow need to also be examined in order to provide a stable adaptive fixed point implementation. These two problems are addressed in the next chapter.

4.6 Conclusion

In this chapter, a symmetry preserving form of CRLS is specified for fixed point implementation. The form is analyzed and shown to suffer from explosive divergence. Using the analytical results, techniques are developed to prevent explosive divergence. Simulation results are used to demonstrate the effectiveness of the stabilization techniques.

CHAPTER 5 - PREVENTING UNDERFLOW AND OVERFLOW

The goal of this chapter is to design a fixed point implementation of the SCRLS algorithm which minimizes the problems of underflow and overflow. Underflow and overflow can lead to weight lock-up and explosive divergence. By normalizing certain algorithmic quantities and carefully choosing how each algorithmic quantity is represented, underflow and overflow problems can be minimized.

As discussed in Chapter 4, underflow and overflow can cause weight lock-up and explosive divergence. If certain quantities related to updating $\mathbf{w}_q(n)$ underflow, weight lock-up occurs. If certain quantities related to updating $\mathbf{C}_q(n)$ overflow (i.e. saturate), then explosive divergence can occur. Thus, a careful design of the fixed point implementation of SCRLS is needed to prevent weight lock-up and to ensure that the techniques derived in Chapter 4 are effective in preventing explosive divergence.

This chapter is organized as follows. First, fixed point arithmetic and implementation practices used in this dissertation are presented. Second, normalizing the $\mathbf{C}$ matrix of the CRLS algorithm is discussed. A design for implementing a normalized version of the SCRLS algorithm is developed, detailing how each algorithmic quantity is to be computed and stored. Finally, the impact of normalization and other design choices on explosive divergence and weight lock-up is discussed.

5.1 Fixed point implementation practices

In this section, fixed point arithmetic and implementation conventions used in this dissertation are given. Background information can be found in a recent tutorial article on programmable DSP chips [LEE 88], as well as a standard text on computer architecture [MANO 76]. Two's complement arithmetic is assumed throughout. Application to 16-bit data storage is discussed, since most current DSP chips have at least 16 bits for data storage [LEE 88].

5.1.1 Storage

In fixed point arithmetic, values are stored in a register or memory location consisting of b_d bits. The most significant bit is used to represent the sign of the value. The next b_i significant bits are used to represent the integer part of the value, and the remaining b_f bits are used to represent the fractional part of the value. This yields the following relationship [ARDA 87]:

$$b_d = 1 + b_i + b_f \qquad (5\text{-}1)$$

To denote how a value is stored (i.e. the format used), the notation $b_i.b_f$ will be used. For example, storing a value using 7.8 format means that 7 bits are used for the integer part and 8 bits are used for the fractional part. It is possible for b_i or b_f to be negative. For example, -2.17 format implies that the most significant data bit (to the right of the sign bit) is used to represent the 1/4's coefficient.

Underflow occurs when the value being represented is so small that the number of bits used to represent the fractional part (b_f) is

inadequate. As a result, the number is represented as zero. Overflow occurs when the number of bits used to represent the integer part (b_i) is inadequate. Saturation logic [LEE 88] will be used when overflow occurs.

When specifying a fixed point quantity, several conventions may be used. The 16-bit value 0000000011110000 may specified as:

a. $0^8 1^4 0^4$, where the power indicates the number of consecutive 1's or 0's [BERT 87].

b. integer 240, where the bit pattern is interpreted as a 16-bit integer.

c. binary 0000000011.110000, assuming 9.6 format.

d. decimal 3.75, assuming the value is in 9.6 format.

These forms are used throughout this chapter.

5.1.2 Addition and subtraction

In fixed point arithmetic, addition is performed as if the two binary representations were integers. Thus, the two quantities being added must have the binary point in the same place. In two's complement arithmetic, subtraction is similar to addition.

5.1.3 Multiplication

Multiplication of two N-bit numbers is usually performed using an accumulator with 2N or more bits [LEE 88]. Thus, a 32-bit accumulator would be used when computing the product of two 16-bit data values. To store the product as a 16-bit quantity, the accumulator is shifted so

that the 16 bits to be stored appear in the lower 16 bits of the accumulator.

Which 16 bits should be kept depends upon a) the number of bits used to represent the fractional part of the multiplicand (b_{fa}), b) the number of bits used to represent the fractional part of the multiplier (b_{fb}), and c) the number of bits used to represent the fractional part of the product (b_{fc}). It can be shown that for given values of b_{fa}, b_{fb}, and b_{fc}, the accumulator should be shifted right by the following amount (s_r) to obtain the 16-bit quantized product in the lower 16 bits of the accumulator:

$$s_r = b_{fa} + b_{fb} - b_{fc} \qquad (5\text{-}2)$$

If the value in the accumulator is non-negative, then 0's should be shifted in from the left. Otherwise, to preserve the sign, 1's should be shifted in from the left.

Simply shifting the accumulator by s_r results in the product being truncated. To perform rounding, the accumulator is right shifted by s_r-1. Then, the 32-bit value $0^{31}1$ is added to the accumulator. The accumulator is then shifted once to the right. This is the technique used by Ardalan and Alexander [ARDA 87].

It is possible for s_r to be zero or negative. In this case, there is no roundoff error. When s_r is negative, the right shift by s_r is performed as a left shift by $-s_r$. Care must be taken when left shifting to avoid accumulator overflow.

5.1.4 Division

Division is usually performed by dividing a 2N bit dividend by an N bit divisor to produce an N bit quotient [MOTO 85]. The dividend and divisor are treated as integers, and simple integer division (using truncation) is performed. To mimic standard division of two numbers, the dividend must be converted from its N-bit value to a 2N-bit value such that the N bit quotient is that desired. To do this, the dividend is left shifted by the following amount:

$$s_l = b_{fr} + b_{fq} - b_{fd} \tag{5-3}$$

where b_{fr}, b_{fq} and b_{fd} are the number of bits used to represent the fractional parts of the divisor, quotient, and dividend respectively.

In the case of rounding, an extra shift left of the dividend must be performed. The resulting quotient is then treated as an accumulated product and $0^{31}1$ is added before right shifting by one.

This completes the discussion of fixed point practices. Equations (5-2) and (5-3) will be used to calculate the accumulator shifts for each computation in the fixed point implementation of SCRLS.

5.2 Normalization

In this section, normalization of the CRLS algorithm is discussed. The C(n) matrix is normalized by a scalar factor so that underflow (which causes weight lock-up) and overflow (which causes explosive divergence) can be prevented.

First, an argument must be made for the necessity of normalization. Allocation of bits to represent the fractional and

integer parts of $\mathbf{C}(n)$ is determined by the range of values $\mathbf{C}(n)$ can take. Consider just the possible steady-state values given by (3-16):

$$\mathbf{C}(n) \cong (1 - \alpha)\, \mathbf{R}_{xx}^{-1} \tag{5-4}$$

Depending on the elements of $\mathbf{R}_{xx}$, the elements of $\mathbf{C}(n)$ could be very large or very small. In many applications, $\mathbf{R}_{xx}$ is either not known in advance or changes with time. Thus, it may be impossible to know in advance how best to choose b_i and b_f for storing the elements of the $\mathbf{C}(n)$ matrix. However, by normalizing the $\mathbf{C}(n)$ matrix in some way, the elements of the normalized $\mathbf{C}(n)$ matrix may be constrained to take on values that neither underflow nor overflow a certain $b_i.b_f$ combination. This is the motivation for normalizing the CRLS algorithm.

Equation (5-4) can also be used to explain when weight lock-up will occur. Consider the case where the elements of $\mathbf{C}(n)$ are so small that they are represented as zeroes in fixed point precision. The algorithmic description of CRLS given in (1-7) reveals that if $\mathbf{C}(n)$ is replaced by a zero matrix, the weight vector $\mathbf{w}(n)$ will cease to adapt (weight lock-up). Equation (5-4) indicates when this underflow problem will occur. This equation can be rewritten as:

$$\mathbf{C}(n) \cong \frac{1 - \alpha}{\sigma_x^2} R_{xx}^{-1} \tag{5-5}$$

where σ_x^2 is the data variance (assuming the data has zero mean) and R_{xx}^{-1} is the inverse of the normalized data autocorrelation matrix (R is normalized to have unity on the diagonal).

Equation (5-5) indicates that the elements of $\mathbf{C}(n)$ become small when either a) the forgetting factor α is close to unity, or b) the

data variance σ_x^2 is large. Thus, weight lock-up can be a problem when either of these conditions is met. Weight lock-up has been linked with the first condition, a forgetting factor close to unity, in past research [GODA 74, ARDA 86, ARDA 87].

An example of weight lock-up is given in Figure 5-1. The simulation conditions were almost the same as those in section 3.4. The filter order is 1 instead of 3, the standard deviation of the white Gaussian driving process is 2.0 instead of 1.0, and the initialization constant δ is 3.3775 in accordance with [COWA 85, BELL 87b]. Plotted is the single weight value as a function of iteration. Observe that the weight stops adapting.

Normalization can be used to combat the problems of underflow and overflow in the **C** matrix. Normalization is achieved by factoring the **C**(n) matrix as:

$$\mathbf{C}(n) = s\,\tilde{\mathbf{C}}(n) \qquad (5\text{-}6)$$

where s is a scalar, and $\tilde{\mathbf{C}}(n)$ is the normalized matrix. The scaling factor s can be made adaptive, as long as for each change in s, $\tilde{\mathbf{C}}(n)$ is re-scaled so that (5-6) holds.

The "parent" algorithm of the CRLS algorithm, the order N^3 general RLS algorithm, is used in Appendix D to derive four normalized CRLS forms. Two of these forms, denoted form 1b and form 3, are used in the next section as part of the fixed point CRLS implementation design.

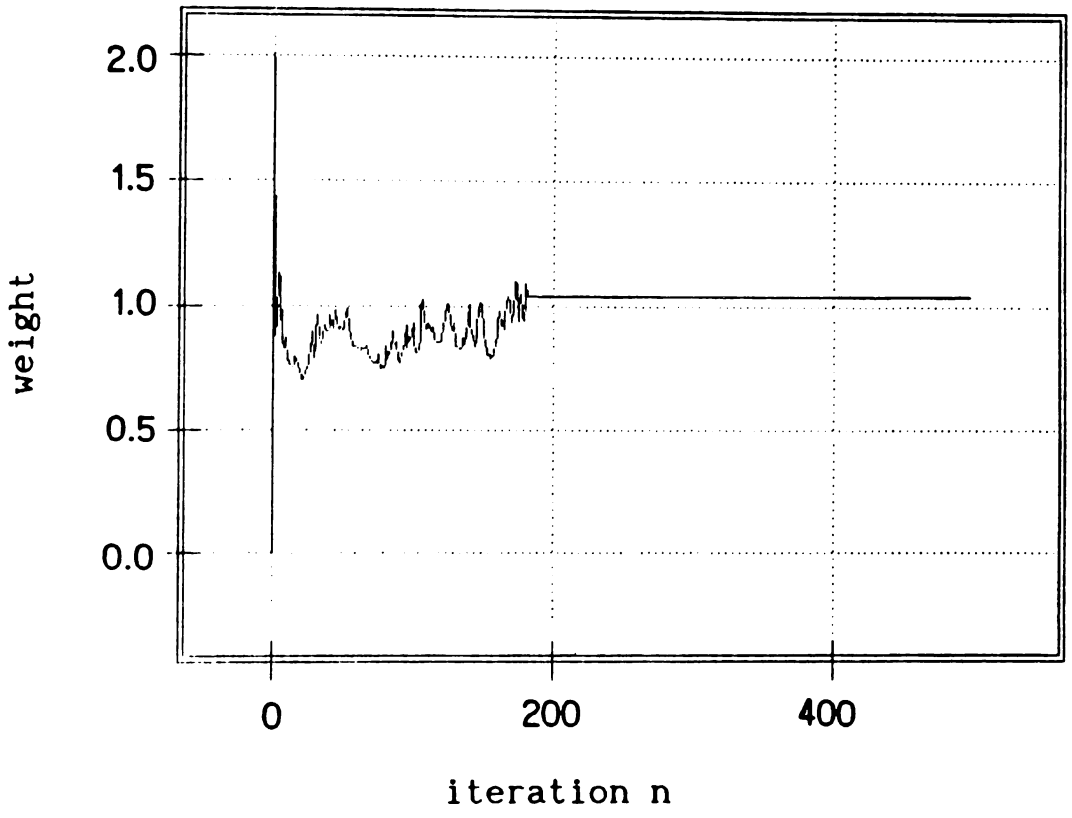

Figure 5-1. Fixed point simulation example of weight lock-up, plotting the weight w(n) as a function of iteration n.

5.3 Fixed point implementation design

This section summarizes the fixed point implementation design of the SCRLS algorithm specified in Chapter 4. Normalization is used to prevent underflow and overflow in the **C** matrix. Also, the storage format for each algorithmic quantity is chosen so that the problems of underflow and overflow are minimized. The design is for a fixed point implementation with 16-bit data and 32-bit accumulation. The main design issues are discussed below.

5.3.1 Normalization approach

The first major design issue is the choice of normalization scheme. Normalization form 1b was chosen for dynamic range purposes, since $s^{1/2}$ is used each iteration instead of s.. The scaling factor s was made adaptive (s(n)), so that the largest element in $\tilde{\mathbf{C}}_q(n)$ is a fraction (0.15 format) that makes full use of the 16 bits available (preventing both underflow and overflow). To make scaling and re-scaling operations simple, s(n) was constrained to the form:

$$s(n) = 2^{2m(n)} \tag{5-7}$$

where m(n) is an integer. Using this form of normalization, (5-6) can be interpreted as using a simple block floating point representation for **C**(n). However, the form 1b equations separate s(n) (the exponent term) from $\tilde{\mathbf{C}}(n)$ (the mantissa term).

5.3.2 Computation of m(n)

It is necessary to compute m(n) at the end of each iteration. The

updated m(n) should satisfy the following criteria:

1. the largest element in $\tilde{C}_q(n)$ is a fraction close to 1 in magnitude, allowing full use of the 15 storage bits.
2. at iteration n+1, the updated $\tilde{C}_q(n+1)$ matrix does not overflow. This implies that the largest element in $\tilde{C}_q(n)$ should not be made too close to 1.

To satisfy the second criterion, the maximum possible change in the largest element of $\tilde{C}(n)$ due to updating must be known. Simulation experience suggests that the element with the largest magnitude always lies on the diagonal of C(n), whether normalized or not. There is an analytical result to support this observation. Under certain conditions, the C(n) matrix is the weight error covariance matrix [MEND 87]. Second, since C(n) is positive definite, all the diagonal values are positive [GRAH 87]. Thus, it is reasonable to assume that the largest element of $\tilde{C}(n)$ is one of the positive diagonal elements.

Next, consider how the diagonal elements of C(n) are updated. From (4-5), the largest $C_{ii}(n)$ can be is $1/\alpha\ C_{ii}(n-1)$. Thus, to satisfy the two criteria above, m(n) should be chosen so that the largest diagonal element of $\tilde{C}_q(n)$ is less than α. The stabilization techniques derived in Chapter 4 will eventually be incorporated into the algorithm. These techniques could cause $C_{ii}(n)$ to exceed $1/\alpha\ C_{ii}(n-1)$. For this reason the largest value for the re-scaled $\tilde{C}_q(n)$ is chosen to be 0.75, which is much less than typical values of α.

Because of the discrete nature of s(n) given in (5-7), it is not possible to make the largest element in $\tilde{C}_q(n)$ exactly equal to 0.75.

However, if the largest element in the unscaled $\tilde{C}_q(n)$ falls below one fourth of 0.75 (0.1875), then clearly, from (5-7), m(n) can be set to m(n-1) - 1 to make elements in the re-scaled $\tilde{C}_q(n)$ larger. With (5-7), re-scaling requires multiplication by either 4 (two left shifts) or 1/4 (one right shift, addition of $0^{15}1$, then another right shift).

5.3.3 Storage of $\mu_q(n)$ and $\xi_q(n)$

Another major design issue is the storage of $\mu_q(n)$ and $\xi_q(n)$ given in (4-9d) and (4-9e) respectively. If these quantities overflow, then explosive divergence can occur. To prevent overflow, the number of bits integer used to store these quantities, denoted $\psi(n)$, is computed each iteration such that overflow is prevented. While overflow of the final 16-bit quantity is prevented, overflow of the 32-bit accumulator prior to storage can still be a problem.

Computation of $\xi_q(n)$ is also an issue. This term is computed using:

$$\xi_q(n) = \alpha + \mu_q(n) \qquad (5\text{-}8)$$

where α is the forgetting factor. To add two fixed point quantities, they must be stored using the same format. Since the format for storing $\mu_q(n)$ is adaptive, the representation for α in (5-8) must also be adaptive. Since α is usually a fraction close to one, α will have different representations depending on how many bits fraction are available. This problem is overcome by incorporating normalization form 3, with the scaling factor s set equal to α. As a result, $\xi_q(n)$, now normalized to $\tilde{\xi}_q(n)$, is given by:

$$\tilde{\xi}_q(n) = 1 + \tilde{\mu}_q(n) \qquad (5\text{-}9)$$

The value "1" be can exactly represented under a variety of formats.

5.3.4 Computation of $\mathbf{w}_q(n)$

The next major design issue is the computation of $\mathbf{w}_q(n)$ such that weight lock-up does not occur. The weight vector $\mathbf{w}_q(n)$ is stored using wfrac bits fraction and is updated as:

$$\mathbf{w}_q(n) = \mathbf{w}_q(n-1) + \mathbf{f}_q(n) \qquad (5\text{-}10)$$

where

$$\mathbf{f}_q(n) = Q[\, \mathbf{g}_q(n)\, e_q(n|n-1)\,] \qquad (5\text{-}11)$$

First, $e_q(n|n-1)$ is computed with an adaptive number of bits fraction efrac(n), preventing underflow. Second, storage of $\mathbf{g}_q(n)$ is carefully chosen so that underflow is not a problem. This leaves the computation and storage of $\mathbf{f}_q(n)$. Since the weight vector $\mathbf{w}_q(n)$ is stored using wfrac bits fraction, $\mathbf{f}_q(n)$ is also stored using wfrac bits fraction. It is possible then for $\mathbf{f}_q(n)$ to underflow when wfrac is not large enough to represent the product $\mathbf{g}_q(n)\, e_q(n|n-1)$. As a result, weight lock-up occurs.

This problem can be alleviated by forcing the entries in $\mathbf{f}_q(n)$ to be nonzero. When the rounded value of an entry in $\mathbf{f}_q(n)$ is computed to be zero, the value is re-computed so that the magnitude is effectively rounded up. This rule is referred to as rounding out. Note that $\mathbf{f}_q(n)$ is computed using the regular rounding rule when the final result is nonzero.

This approach is similar to one presented by Sakurai [SAKU 84]. Sakurai quantized $\mathbf{f}_q(n)$ using an off-set truncation rule, which does not allow the quantized value to take on the value of 0. This is effectively rounding out when computing all values of $\mathbf{f}_q(n)$. The technique used in this dissertation uses rounding out only when needed to prevent underflow.

5.3.5 Overflow prevention

Finally, storage formats were chosen so that overflow resulting from the product of two scalars would not be a problem. However, the quantities $\mathbf{p}_q(n)$ and $\mu_q(n)$ are formed by accumulating multiple scalar products. Overflow of $\mu_q(n)$ is prevented by increasing the number of bits integer, $\psi(n)$, until overflow is not a problem. It is assumed that the order of the filter is small enough so that overflow when forming $\mathbf{p}_q(n)$ is not a problem. Should the filter order be large or the data values be extremely large, then the storage of $\mathbf{p}_q(n)$ should be modified to accommodate larger values. Also, overflow of the accumulators for $\mathbf{p}_q(n)$ and $\mu_q(n)$ can be a problem. Using DSP chips which have extra accumulator bits can alleviate this problem [LEE 88].

This completes the discussion of the major design issues. The resulting fixed point design is summarized in the next sub-section.

5.3.6 Design summary

The normalized fixed point implementation of the SCRLS algorithm is given by:

Initial conditions:

$$\mathbf{w}_q(0) = \mathbf{x}(0) = \mathbf{0} \tag{5-12a}$$

$$\tilde{\mathbf{C}}_q(0) = 1/\alpha \; 2^{-2m(0)} \; \delta \; \mathbf{I} \qquad (\delta >> 1) \tag{5-12b}$$

where m(0) is chosen such that $0.1875 < 1/\alpha \; 2^{-2m(0)} \; \delta \leq 0.75$.

For n = 1, 2, ... **do:**

$$\tilde{\mathbf{x}}(n) = 2^{m(n-1)} \; \mathbf{x}(n) \tag{5-13a}$$

compute efrac(n) so $\hat{d}_q(n)$ and $e_q(n|n-1)$ do not overflow (5-13b)

$$d_q(n) = 2^{efrac(n)-15} \; d(n) \tag{5-13c}$$

$$\hat{d}_q(n) = Q\left[\mathbf{x}^T(n) \; \mathbf{w}_q(n-1) \right] \tag{5-13d}$$

$$e_q(n|n-1) = d_q(n) - \hat{d}_q(n) \tag{5-13e}$$

$$\tilde{\mathbf{p}}_q(n) = Q\left[\tilde{\mathbf{C}}_q(n-1) \; \tilde{\mathbf{x}}(n) \right] \tag{5-13f}$$

compute $\psi(n)$ so that $\tilde{\mu}_q(n)$ and $\tilde{\xi}_q(n)$ do not overflow (5-13g)

$$\tilde{\mu}_q(n) = Q\left[\tilde{\mathbf{x}}^T(n) \; \tilde{\mathbf{p}}_q(n) \right] \tag{5-13h}$$

$$\tilde{\xi}_q(n) = 1_1 + \tilde{\mu}_q(n) \tag{5-13i}$$

$$\tilde{\beta}_q(n) = Q\left[1_2 \; / \; \tilde{\xi}_q(n) \right] \tag{5-13j}$$

$$\mathbf{g}_q(n) = Q\left[s(n-1)^{1/2} \; \tilde{\beta}_q(n) \; \tilde{\mathbf{p}}_q(n) \right] \tag{5-13k}$$

$$\mathbf{f}_q(n) = Q\left[\mathbf{g}_q(n) \; e_q(n|n-1) \right]$$

$$\text{if } f_{qi}(n) = 0, \; \mathbf{f}_q(n) = Q_o\left[\mathbf{g}_q(n) \; e_q(n|n-1) \right] \tag{5-13l}$$

$$\mathbf{w}_q(n) = \mathbf{w}_q(n-1) + \mathbf{f}_q(n) \tag{5-13m}$$

$$\tilde{\mathbf{H}}_q(n) = Q\left[\tilde{\mathbf{p}}_q(n)\, \tilde{\mathbf{p}}_q^{T}(n) \right] \tag{5-13n}$$

$$\tilde{\mathbf{J}}_q(n) = Q\left[\tilde{\beta}_q(n)\, \tilde{\mathbf{H}}_q(n) \right] \tag{5-13o}$$

$$\tilde{\mathbf{M}}_q(n) = \tilde{\mathbf{C}}_q(n-1) - \tilde{\mathbf{J}}_q(n) \tag{5-13p}$$

$$\tilde{\mathbf{C}}_q(n) = Q\left[(1/\alpha)\, \tilde{\mathbf{M}}_q(n) \right] \tag{5-13q}$$

update m(n) and re-scale $\tilde{\mathbf{C}}_q(n)$ (5-13r)

(optional)

$$y_q(n) = \mathbf{x}^{T}(n)\, \mathbf{w}_q(n) \tag{5-13s}$$

$$e_q(n) = d(n) - y_q(n) \tag{5-13t}$$

where $Q_o[\cdot]$ denotes rounding out. Note that (5-13a) requires no computation because of the way $\tilde{\mathbf{x}}(n)$ is stored. The subscripts on 1 in (5-13i) and (5-13j) indicate different storage formats for unity.

As a result of normalizing using forms 1a and 3, the overall scaling factor is equal to:

$$s(n) = \alpha\, 2^{2m(n-1)} \tag{5-14}$$

Ignoring roundoff errors, the normalized quantities are related to the original CRLS quantities as follows:

$$\tilde{\mathbf{p}}(n) = 1/\alpha\, 2^{-m(n-1)}\, \mathbf{p}(n) \tag{5-15a}$$

$$\tilde{\mu}(n) = 1/\alpha\, \mu(n) \tag{5-15b}$$

$$\tilde{\xi}(n) = 1/\alpha\, \xi(n) \tag{5-15c}$$

$$\tilde{\beta}(n) = \alpha\, \beta(n) = \gamma(n) \tag{5-15d}$$

$$\tilde{\mathbf{H}}(n) = 1/\alpha^2\, 2^{-2m(n-1)}\, \mathbf{H}(n) \tag{5-15e}$$

$$\tilde{J}(n) = 1/\alpha \; 2^{-2m(n-1)} \; J(n) \qquad (5\text{-}15f)$$

$$\tilde{\mathbf{M}}(n) = 1/\alpha \; 2^{-2m(n-1)} \; \mathbf{M}(n) \qquad (5\text{-}15g)$$

$$\tilde{\mathbf{C}}(n) = 1/\alpha \; 2^{-2m(n-1)} \; \mathbf{C}(n) \qquad (5\text{-}15h)$$

The computation and storage requirements are summarized in Tables 5-1 and 5-2. First, Table 5-1 details the number of bits fraction for each quantity in the normalized fixed point SCRLS algorithm. Second, Table 2 indicates the right or left shift required for each equation in (5-13). These shifts were computed using (5-2) and (5-3). This completes the fixed point design specification for the normalized SCRLS algorithm.

5.4 Impact on divergence and lock-up

The normalized fixed point implementation of the SCRLS algorithm specified in the previous section impacts the likelihood of both explosive divergence and weight lock-up. In this section, the impact of the design on these two problems is discussed.

First, consider the results of Chapter 3 with regards to explosive divergence. Recall that the likelihood of divergence increases as the number of bits allocated to fractions decreases. Normalization allows all 15 data bits available for representing $\mathbf{C}(n)$ to be allocated for fractions. Second, despite a wordlength of 16 bits, the effective wordlength, in terms of number of bits actually used, may be much less without normalization. If the effective wordlength is small, the relative size of roundoff errors can be large. Normalization allows

QUANTITY	BITS	QUANTITY	BITS
$\mathbf{x}(n)$	15	$\tilde{\beta}_q(n)$	13 + $\psi(n)$
d(n)	15	$\mathbf{g}_q(n)$	14 + $\psi(n)$ - 2m(n-1)
$\tilde{\mathbf{x}}(n)$	15 - m(n-1)	$\mathbf{f}_q(n)$	wfrac
efrac(n)	0	$\mathbf{w}_q(n)$	wfrac
$d_q(n)$	efrac(n)	$\tilde{\mathbf{H}}_q(n)$	15 - $\psi(n)$
$\hat{d}_q(n)$	efrac(n)	$\tilde{J}_q(n)$	15
$e_q(n\|n-1)$	efrac(n)	$\tilde{\mathbf{M}}_q(n)$	15
$\tilde{\mathbf{p}}_q(n)$	15 - m(n-1)	$\tilde{\mathbf{C}}_q(n)$	15
$\tilde{\mu}_q(n)$	15 - $\psi(n)$	$1/\alpha$	14
$\tilde{\xi}_q(n)$	15 - $\psi(n)$	$y_q(n)$	15
1_1	15 - $\psi(n)$	$e_q(n)$	15
1_2	0	m(n)	0
$\psi(n)$	0		

Table 5-1. Number of bits fraction allocated to each quantity in the normalized SCRLS algorithm.

EQUATION	SHIFT	EQUATION	SHIFT
(5-13a)	---	(5-13k)	14
(5-13b)	---	(5-13l)	14 + ψ(n) - 2m(n-1) + efrac(n) - wfrac
(5-13c)	15 - efrac(n)	(5-13m)	0
(5-13d)	15+wfrac-efrac(n)	(5-13n)	15 + ψ(n) - 2m(n-1)
(5-13e)	0	(5-13o)	13
(5-13f)	15	(5-13p)	0
(5-13g)	---	(5-13q)	14
(5-13h)	15+ψ(n)-2m(n-1)	(5-13r)	---
(5-13i)	0	(5-13s)	wfrac
(5-13j)	28*	(5-13t)	0

Table 5-2. Amount to right shift after accumulation for each equation.

* - amount to left shift dividend before division.

the effective wordlength to be large, making roundoff errors relatively small.

Finally, overflow of certain quantities introduces large errors which bias the algorithm towards explosive divergence. Specifically, if $\tilde{\mathbf{p}}_q(n)$, $\tilde{\mu}_q(n)$, or $\tilde{\mathbf{C}}_q(n)$ overflow, then explosive divergence can occur. Normalization prevents overflow of $\tilde{\mathbf{p}}_q(n)$ and $\tilde{\mathbf{C}}_q(n)$. Using an adaptive representation for $\tilde{\mu}_q(n)$ prevents this quantity from overflowing. Thus, the fixed point design presented in section 5.3 should be less susceptible to explosive divergence.

Now, consider the problem of weight lock-up. Weight lock-up occurs when $\mathbf{f}_q(n)$ remains zero over successive iterations. By inspection of (5-13), weight lock-up occurs if any of the following underflow: $\mathbf{x}(n)$, $e_q(n|n-1)$, $\tilde{\mathbf{C}}_q(n)$, $\tilde{\mathbf{p}}_q(n)$, $\tilde{\beta}_q(n)$, $\mathbf{g}_q(n)$, or $\mathbf{f}_q(n)$. As for $\mathbf{x}(n)$, underflow is prevented by setting the A/D voltage range such that the 16 bit dynamic range of the A/D output is fully utilized. Assuming $\mathbf{x}(n)$ does not underflow, the normalized SCRLS implementation helps prevent underflow of the remaining quantities in the following ways:

1. By setting efrac(n), the number of bits fraction for $e_q(n|n-1)$, to 15 and only decreasing it in the case of overflow, the quantity $e_q(n|n-1)$ should not underflow.
2. By normalizing $\tilde{\mathbf{C}}_q(n)$, underflow is prevented. Also, $\tilde{\mathbf{p}}_q(n)$ should not underflow.
3. By normalizing $\tilde{\mu}_q(n)$, $\tilde{\xi}_q(n)$, and $\tilde{\beta}_q(n)$ using $\psi(n)$, underflow in $\tilde{\beta}_q(n)$ is prevented.

4. Normalization using m(n) and $\psi(n)$ prevents $\mathbf{g}_q(n)$ from underflowing.

5. By choosing wfrac (number of bits fraction for $\mathbf{w}_q(n)$) large enough, $\mathbf{f}_q(n)$ should not underflow. However, even if wfrac is small, the technique of rounding out an element of $\mathbf{f}_q(n)$ when it is equal to zero will keep $\mathbf{f}_q(n)$ from underflowing as long as $\mathbf{g}_q(n)$ and $e_q(n|n-1)$ are nonzero.

Thus, the fixed point design presented in section 5.3 prevents underflow, which is the cause of weight lock-up in the CRLS algorithm.

5.5 Conclusion

In conclusion, this chapter presents a detailed design for implementing the SCRLS algorithm in fixed point hardware. Normalization of the $\mathbf{C}(n)$ matrix is developed and used to prevent underflow and overflow problems. Simple floating point schemes are used where necessary to prevent other underflow/overflow problems. The resulting design makes efficient use of the bits available while minimizing underflow and overflow problems. This in turn prevents the problems of explosive divergence and weight lock-up.

CHAPTER 6 - INTEGRATION AND TESTING

In this chapter, the stabilization techniques of Chapter 4 are integrated into the fixed point implementation designed in Chapter 5. The integrated design is tested under a variety of conditions to show that explosive divergence and weight lock-up can be effectively prevented. Finally, the chapter concludes that increased stability can be achieved with small cost in performance.

6.1 Integration

The previous chapter detailed the design for a 16-bit fixed point implementation of the SCRLS algorithm. In this section, the stabilization techniques of Chapter 4 are integrated into the fixed point design.

First, the techniques listed in section 4.4.8 must be applied to the normalized quantities given in (5-13). This results in the following stabilization techniques:

1.1 Round down or truncate when forming $\tilde{\beta}_q(n)$.

1.2 Round up when forming $\tilde{\mu}_q(n)$.

1.3 When forming $\tilde{p}_q(n)$ to compute $\tilde{\mu}_q(n)$, denoted $\tilde{p}_{\mu q}(n)$, use:

if $\tilde{x}_i(n) > 0$, round up

if $\tilde{x}_i(n) < 0$, round down

2.1 When forming $\tilde{p}_q(n)$ to compute the diagonal elements of $\tilde{H}_q(n)$, denoted $\tilde{p}_{Hq}(n)$, round in (round down in magnitude).

3.1 When forming the diagonal elements of $\tilde{H}_q(n)$, subtract "1" from the least significant bit of the accumulator if the accumulator is nonzero.

4.1 When forming $\tilde{H}_q(n)$, round down on the diagonal.

5.1 When forming $\tilde{J}_q(n)$, round down on the diagonal.

6.1 When forming $\tilde{C}_q(n)$, round up on the diagonal.

Most of the stabilization techniques simply change a rounding operation, previously denoted $Q[\cdot]$, to either rounding up or rounding down. These operations will be denoted $Q_u[\cdot]$ and $Q_d[\cdot]$ respectively. The operations of rounding in (rounding down in magnitude) and rounding out (rounding up in magnitude) will be denoted $Q_i[\cdot]$ and $Q_o[\cdot]$ respectively. Technique 6.1 also applies to re-scaling $\tilde{C}_q(n)$.

Adding the above techniques to the fixed point design specified in (5-12) and (5-13) gives a stabilized fixed point implementation of the SCRLS algorithm, which will be referred to as the SSCRLS filter. The filter is specified as follows.

Initial conditions:

$$w_q(0) = x(0) = 0 \tag{6-1a}$$

$$\tilde{C}_q(0) = 1/\alpha \; 2^{-2m(0)} \; \delta \; I \qquad (\delta >> 1) \tag{6-1b}$$

where $m(0)$ is chosen such that $0.1875 < 1/\alpha \; 2^{-2m(0)} \; \delta \leq 0.75$.

For n = 1, 2, ... **do:**

$$\tilde{\mathbf{x}}(n) = 2^{m(n-1)}\, \mathbf{x}(n) \text{ (not implemented)} \tag{6-2a}$$

compute efrac(n) so $\hat{d}_q(n)$ and $e_q(n|n-1)$ do not overflow (6-2b)

$$d_q(n) = 2^{efrac(n)-15}\, d(n) \text{ (right shift with rounding)} \tag{6-2c}$$

$$\hat{d}_q(n) = Q\left[\, \mathbf{x}^T(n)\, \mathbf{w}_q(n-1) \,\right] \tag{6-2d}$$

$$e_q(n|n-1) = d_q(n) - \hat{d}_q(n) \tag{6-2e}$$

$$\tilde{\mathbf{p}}_{\mu q}(n):\ p_{\mu qi}(n) = Q_u\left[\, \tilde{\mathbf{C}}_q(n-1)\, \tilde{\mathbf{x}}(n) \,\right] \quad \text{if } \tilde{x}_i(n) > 0$$

$$p_{\mu qi}(n) = Q_d\left[\, \tilde{\mathbf{C}}_q(n-1)\, \tilde{\mathbf{x}}(n) \,\right] \quad \text{if } \tilde{x}_i(n) < 0 \tag{6-2f}$$

compute $\psi(n)$ so that $\tilde{\mu}_q(n)$ and $\tilde{\xi}_q(n)$ do not overflow (6-2g)

$$\tilde{\mu}_q(n) = Q_u\left[\, \tilde{\mathbf{x}}^T(n)\, \tilde{\mathbf{p}}_q(n) \,\right] \tag{6-2h}$$

$$\tilde{\xi}_q(n) = 1_1 + \tilde{\mu}_q(n) \tag{6-2i}$$

$$\tilde{\beta}_q(n) = Q_d\left[\, 1_2 \,/\, \tilde{\xi}_q(n) \,\right] \tag{6-2j}$$

$$\mathbf{g}_q(n) = Q\left[\, s(n-1)^{1/2}\, \tilde{\beta}_q(n)\, \tilde{\mathbf{p}}_q(n) \,\right] \tag{6-2k}$$

$$\mathbf{f}_q(n) = Q\left[\, \mathbf{g}_q(n)\, e_q(n|n-1) \,\right]$$

$$\text{if } f_{qi}(n) = 0,\ f_{qi}(n) = Q_o\left[\, \mathbf{g}_q(n)\, e_q(n|n-1) \,\right] \tag{6-2l}$$

$$\mathbf{w}_q(n) = \mathbf{w}_q(n-1) + \mathbf{f}_q(n) \tag{6-2m}$$

$$\tilde{\mathbf{p}}_{Hq}(n) = Q_i\left[\, \tilde{\mathbf{C}}_q(n-1)\, \tilde{\mathbf{x}}(n) \,\right] \tag{6-2n}$$

$$\tilde{H}_q(n):\ H_{ii} = Q_d\left[\ \tilde{p}_{Hq}(n)\ \tilde{p}_{Hq}^{T}(n)\ \right] \quad \text{if } \tilde{p}_{Hqi}(n) = 0$$

$$= Q_d\left[\ \tilde{p}_{Hq}(n)\ \tilde{p}_{Hq}^{T}(n) - 0^{31}1\ \right] \quad \text{if } \tilde{p}_{Hqi}(n) \neq 0$$

$$H_{ij} = Q\left[\ \tilde{p}_q(n)\ \tilde{p}_q^{T}(n)\ \right] \qquad i \neq j \qquad (6\text{-}2o)$$

$$\tilde{J}_q(n):\ J_{ii} = Q_d\left[\ \tilde{\beta}_q(n)\ \tilde{H}_q(n)\ \right]$$

$$J_{ij} = Q\left[\ \tilde{\beta}_q(n)\ \tilde{H}_q(n)\ \right] \qquad i \neq j \qquad (6\text{-}2p)$$

$$\tilde{M}_q(n) = \tilde{C}_q(n\text{-}1) - \tilde{J}_q(n) \qquad (6\text{-}2q)$$

$$\tilde{C}_q(n):\ C_{ii} = Q_u\left[\ (1/\alpha)\ \tilde{M}_q(n)\ \right]$$

$$C_{ij} = Q\left[\ (1/\alpha)\ \tilde{M}_q(n)\ \right] \qquad i \neq j \qquad (6\text{-}2r)$$

update m(n) and re-scale $\tilde{C}_q(n)$ (6-2s)

(optional)

$$y_q(n) = \mathbf{x}^T(n)\ \mathbf{w}_q(n) \qquad (6\text{-}2t)$$

$$e_q(n) = d(n) - y_q(n) \qquad (6\text{-}2u)$$

where 1_1 is unity stored in $\psi(n).15\text{-}\psi(n)$ format, and 1_2 is unity stored in 15.0 format.

Equation (6-2b) consists of computing efrac(n), the number of bits fraction for $e_q(n|n\text{-}1)$, such that $e_q(n|n\text{-}1)$ does not overflow. To do this, efrac(n) is first set to 15. If $\hat{d}_q(n)$ or $e_q(n|n\text{-}1)$ overflow, then efrac(n) is decremented and the terms $\hat{d}_q(n)$ and $e_q(n|n\text{-}1)$ are re-computed. This process continues until overflow is not a problem.

Equation (6-2g) requires computing $\psi(n)$, the number of bits

fraction for $\tilde{\mu}_q(n)$ and $\tilde{\xi}_q(n)$, so that the terms $\tilde{\mu}_q(n)$ and $\tilde{\xi}_q(n)$ do not overflow. To do this, $\psi(n)$ is initially set to one, and the two terms are computed. If overflow occurs, $\psi(n)$ is incremented, and the two terms are re-computed. This process continues until overflow is not a problem.

Finally, equation (6-2s) requires updating m(n) and re-scaling $\tilde{C}_q(n)$. First, the largest diagonal entry of $\tilde{C}_q(n)$ is extracted (denoted cmax), and m(n) is initially set to m(n-1). If cmax falls below the representation for 0.1875, then m(n) is decremented and each entry of $\tilde{C}_q(n)$ is left shifted by two places (i.e. multiplied by 4). The process repeats until the maximum diagonal entry of $\tilde{C}_q(n)$ is at least 0.1875. Then, a similar procedure is used to ensure that the maximum value of $\tilde{C}_q(n)$ is not greater than the representation for 0.75.

Simulation results using the SSCRLS filter for the examples given in Chapters 3, 4 and 5 are presented. Shown in Figure 6-1 is a plot of the angle parameter $\gamma_q(n)$ under the conditions described in section 3.4. Clearly, $\gamma_q(n)$ stays within its theoretical range of (0,1]. Thus, explosive divergence, as shown in Figure 3-1, has been prevented. Shown in Figure 6-2 are the normalized weight error trajectories for the SSCRLS filter and a double precision implementation of the CRLS filter. The trajectories show close agreement, indicating that least squares performance is still preserved in the SSCRLS filter.

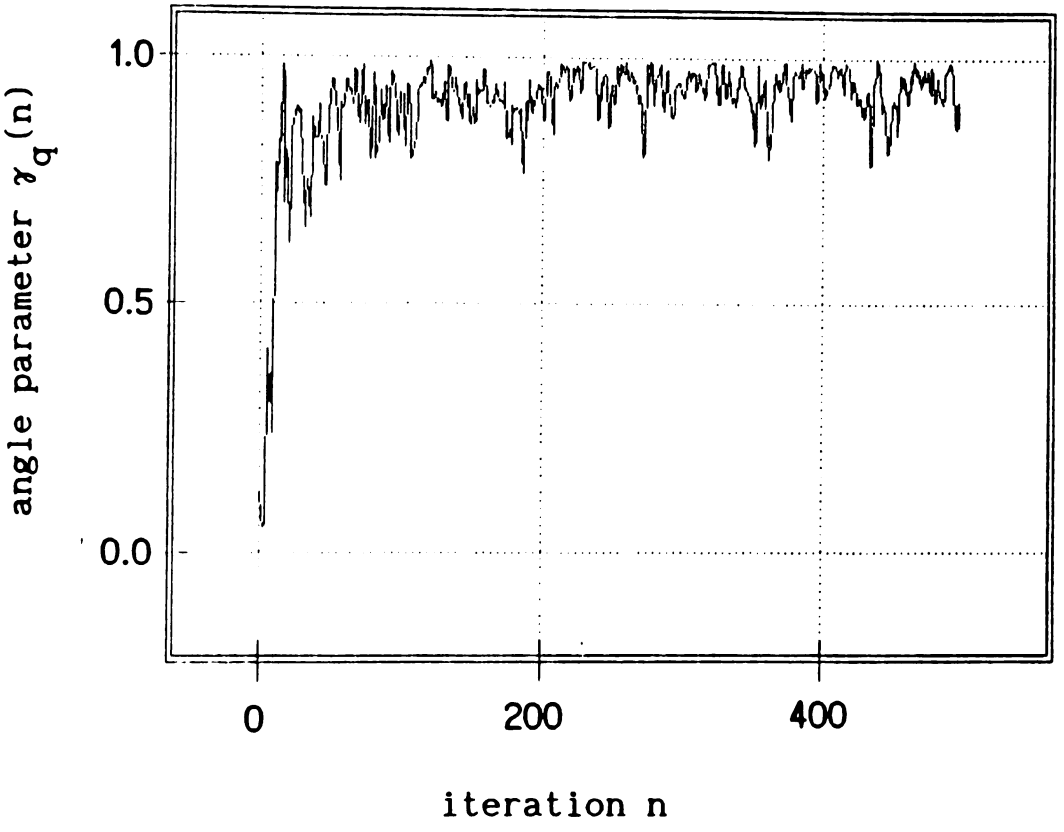

Figure 6-1. SSCRLS simulation example demonstrating that explosive divergence is prevented. Plotted is the angle parameter $\gamma_q(n)$ as a function of iteration n.

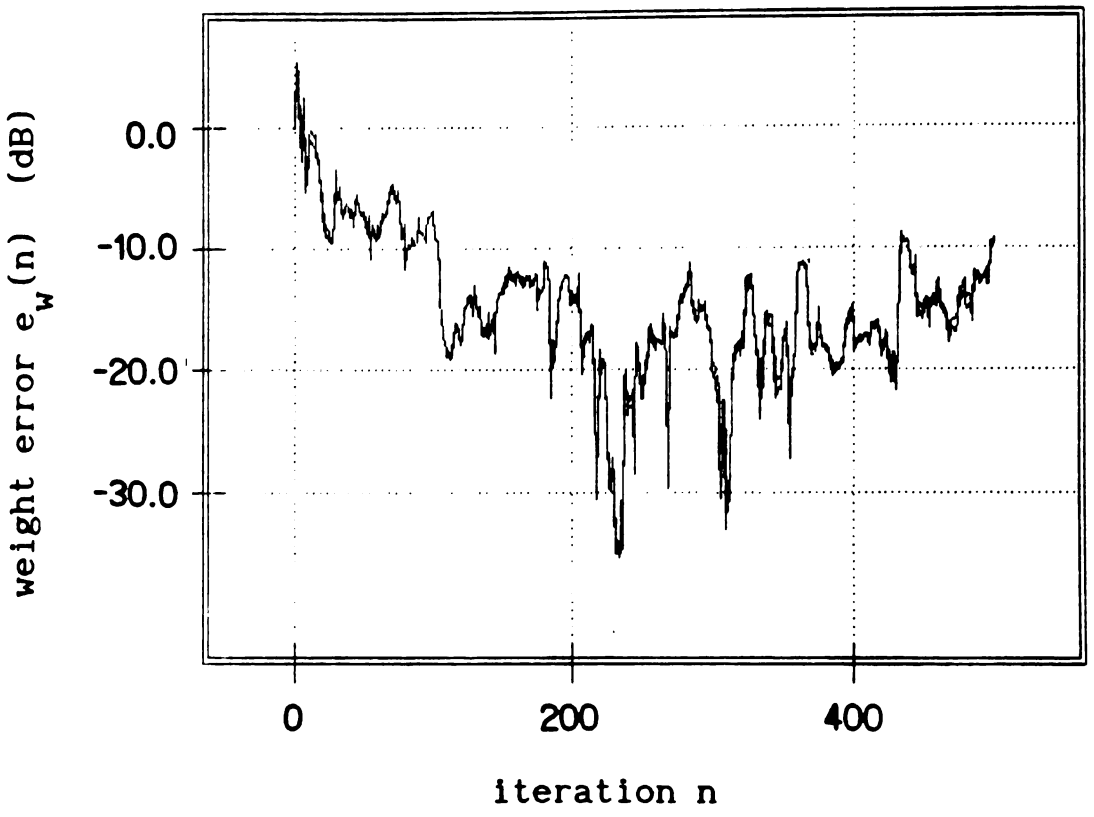

Figure 6-2. Comparison of normalized weight error trajectories for SSCRLS filter and double precision floating point CRLS filter. Results for explosive divergence example in Chapter 3.

Next, the weight lock-up example of Figure 5-1 was simulated using the SSCRLS filter. Shown in Figure 6-3 are the weight trajectories for the SSCRLS filter and a double precision floating point CRLS implementation. Weight lock-up is not a problem, and the weight trajectories show close agreement, indicating preservation of least squares performance.

More extensive tests were performed to determine the effectiveness of the SSCRLS design. Test cases prone to explosive divergence and weight lock-up are developed in the next section.

6.2 Simulation test cases

In this section, two test scenarios are developed to test the effectiveness of the SSCRLS design. The scenarios incorporate conditions that are known to cause explosive divergence and weight lock-up.

Conditions leading to explosive divergence in the CRLS algorithm are:

1. Near singular **C**(n) [CIOF 87]. This occurs when the data autocorrelation matrix, $\mathbf{R}_{xx}$, is ill-conditioned. It can also occur when the maximum eigenvalue of $\mathbf{R}_{xx}$ is large, as discussed in Chapter 3.
2. A large number of iterations [CIOF 87]. Explosive divergence in CRLS occurs after 1000's of iterations.

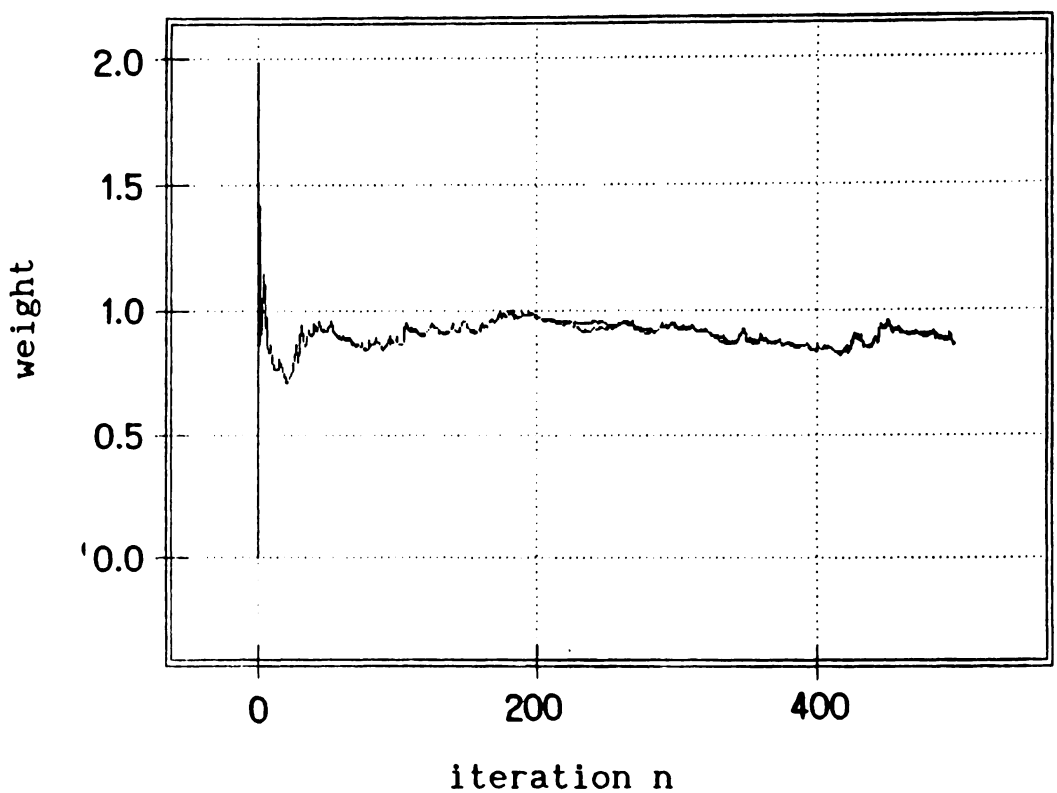

Figure 6-3. Comparison of a SSCRLS simulation example demonstrating weight lock-up prevention with a double precision floating point simulation. Plotted are the weight value $w_q(n)$ trajectories as a function of iteration n.

3. Small wordlengths [HSU 82]. In this dissertation, the wordlength has been fixed at 16 bits, which is relatively small compared to recent Motorola DSP chips [LEE 88].

An additional factor known to contribute to explosive divergence in fast RLS filters is:

4. Small forgetting factor (α) [CIOF 84, FABR 85, FABR 86], around 0.95 [FABR 85].

Conditions leading to weight lock-up are the following:

1. The forgetting factor equal to 1 [GODA 74, ARDA 86, ARDA 87], or very close to 1 [ARDA 87, BOTT 89]. A value of 0.999 was shown to lead to weight lock-up [ARDA 87].
2. Low additive noise in the system identification application [ARDA 87].
3. Large data variance (actually data mean square value) [BOTT 89] (see section 5.1).

Based on these conditions, two test scenarios were developed.

6.2.1 Test scenario 1

The first scenario is the one-step linear prediction application described in section 3.4. The auto-regressive (AR) generating process is first-order, with the AR coefficient set to 0.98 and the driving noise variance set to 1. This gives a data variance of 25.25.

For a three tap adaptive filter, this process has a data autocorrelation matrix $\mathbf{R}_{xx}$ with a condition number of 219.3 and a maximum eigenvalue of 74.42. Thus, condition 1 leading to explosive

divergence is satisfied by this scenario.

6.2.2 Test scenario 2

The second test scenario simulates the systems identification application [ARDA 87]. The system is a moving average (MA) process, generated as follows:

$$z(n) = 0 \qquad n \le 0$$

$$z(n) = w^*_0\, v(n) + \ldots + w^*_{M-1}\, v(n-M+1) + n(n) \qquad n > 0 \qquad (6\text{-}3)$$

where w^*_i are the system parameters, denoted collectively as $\mathbf{w}^*$, and the processes v(n) and n(n) are independent zero-mean, white Gaussian sequences such that:

$$v(n) = 0 \qquad n \le 0 \qquad (6\text{-}4a)$$

$$n(n) = 0 \qquad n \le 0 \qquad (6\text{-}4b)$$

The quantized signals used by the adaptive filter are formed as follows:

$$d(n) = Q[\, z(n) \,] \qquad (6\text{-}5a)$$

$$x(n) = Q[\, v(n) \,] \qquad (6\text{-}5b)$$

For this scenario, the variance of v(n), denoted σ_v^2, is set to 1, and a third order system is used with parameter vector $\mathbf{w}^*$ set to:

$$\mathbf{w}^* = [\, 0.8 \quad 0.0 \quad 0.6 \,]^T \qquad (6\text{-}6)$$

As a result, both z(n) and v(n) have a variance of 1. The variance of the additive noise, n(n), is set to 0.1, satisfying condition 2 for weight lock-up.

6.2.3 Data quantization

As indicated in the previous two sub-sections, the data are quantized to 16-bit values to be processed by the 16-bit fixed point SSCRLS filter. This models the effect of an analog-to-digital or A/D converter. How the data are quantized depends upon two selectable parameters: a) the number of bits to be used to quantize the fractional part of the data (vfrac), and b) the total number of bits to be used to quantize the data.

With a 16-bit SSCRLS filter, the maximum number of total data bits is 16. However, the A/D converter may not produce that many bits. As a result, the data bits are treated and stored as the least significant bits of a 16-bit word.

For simulation purposes, there is a need to use less than 16 bits to represent the data. In many standard DSP chips, the accumulator used to accumulate multiple scalar products consists of more than the number of bits necessary to store a single scalar product [LEE 88]. For example, the AT&T DSP16 has a 36-bit accumulator for accumulating the 32-bit products of two 16-bit quantities. The four extra bits prevent overflow of the accumulator before quantization to 16 bits takes place. These four headroom bits provide for $2^4 - 1$ or 15 scalar products to be accumulated without overflow [LEE 88].

Unfortunately, the largest integer quantity available to the simulation software is 32 bits long. Consequently, accumulator overflow can occur when simulating a SSCRLS filter order greater than 1. To overcome this problem, the data are quantized to 12 bits. As a

result, each scalar product is 12 + 16 = 28 bits long, giving the 32 bit accumulator 4 headroom bits. Thus, filters up to 15 taps long can be simulated without risk of accumulator overflow.

For test case 1, the data are quantized to 12-bit values using 6 bits for fractions. For test case 2, the data are quantized to 12-bit values using 9 bits for fractions.

6.2.4 Filter parameters

For all simulations, the number of iterations generated is 1,000,000. The large number of iterations satisfies condition 2 for explosive divergence. The adaptive filter is run with several combinations of forgetting factors and filter orders. To satisfy condition 4 for explosive divergence and condition 1 for weight lock-up, forgetting factor values of 0.9, 0.95, 0.98, and 0.999 are used. Filter orders of 1, 3, and 15 are used to simulate both small and large order filters. Fifteen is the largest order filter that guarantees prevention of accumulator overflow with 12-bit data.

The adaptive filter is initialized with δ chosen according to the following criterion [COWA 85, BELL 87b]:

$$\delta = \frac{100}{\sigma_x^2} \quad (1/\text{volts}^2) \tag{6-7}$$

The SSCRLS treats d(n) and $\mathbf{x}(n)$ as if 15 bits are being used to represent fractions. If the data are not quantized using 15 bits for fractions of volts, then effectively the units for the data have been changed. Consequently, the value for δ must be scaled to be in the

corresponding units. This scaling is given by:

$$\delta' = \delta \, 2^{2(15 - \text{vfrac})} \tag{6-8}$$

where vfrac is the number of bits used to represent the fractional part of the original voltage waveform.

6.2.5 Test case effectiveness

Detailed above are two test scenarios. For each scenario, combinations of 3 possible filter orders and 4 possible forgetting factors are simulated. This gives a total of 24 simulation runs, each for 1,000,000 iterations. To efficiently run such simulations, the simulation program code was transferred from the AT&T PC to a VAX 3100.

To test whether explosive divergence and weight lock-up are problems under these test conditions, the test scenarios were first simulated using a simple fixed point CRLS filter. Seven of the 15 data bits were used to represent fractions for all quantities. Ten thousand iterations were generated for each case. For test scenario 1, explosive divergence was a problem for filter orders of 3 and 15, regardless of forgetting factor. Weight lock-up occurred when the filter order was 15 and the forgetting was 0.999, 0.95, or 0.90. For test scenario 2, weight lock-up was a problem for all filter order and forgetting factor combinations. Explosive divergence only occurred when the filter order was 15. Thus, the two test scenarios clearly provide examples of both explosive divergence and weight lock-up.

6.3 Simulation test results

This section reports the results of simulation tests performed to demonstrate the performance of the SSCRLS design. The main result is that explosive divergence and weight lock-up were effectively prevented. Comparisons to double precision floating point simulations of the original CRLS algorithm show that the SSCRLS filter performance agrees closely with the ideal least squares recursive filter.

To make the conditions for the SSCRLS filter the same as for the simple fixed point filter of the previous section, the number of bits used to represent the fractional part of $w_q(n)$, wfrac, was set to 7. Then, the 24 simulation tests were performed. For all 24 runs, neither explosive divergence nor weight lock-up occurred. Thus, the techniques developed in Chapters 4 and 5 effectively prevented these problems from occurring.

To check whether least squares performance was also being achieved, comparisons of the normalized weight error trajectories of the SSCRLS filter and a double precision floating point CRLS filter were made. To distinguish multiple curves on the same plot, only the first 100,000 iterations are shown. For plotting purposes, only every 1000th value is plotted.

Shown in Figure 6-4 are results for test scenario 1 with a filter order of 3 and a forgetting factor of 0.98. Clearly, there is good agreement in the two normalized weight error trajectories. Thus, the stabilization techniques had little impact on least squares performance for this example.

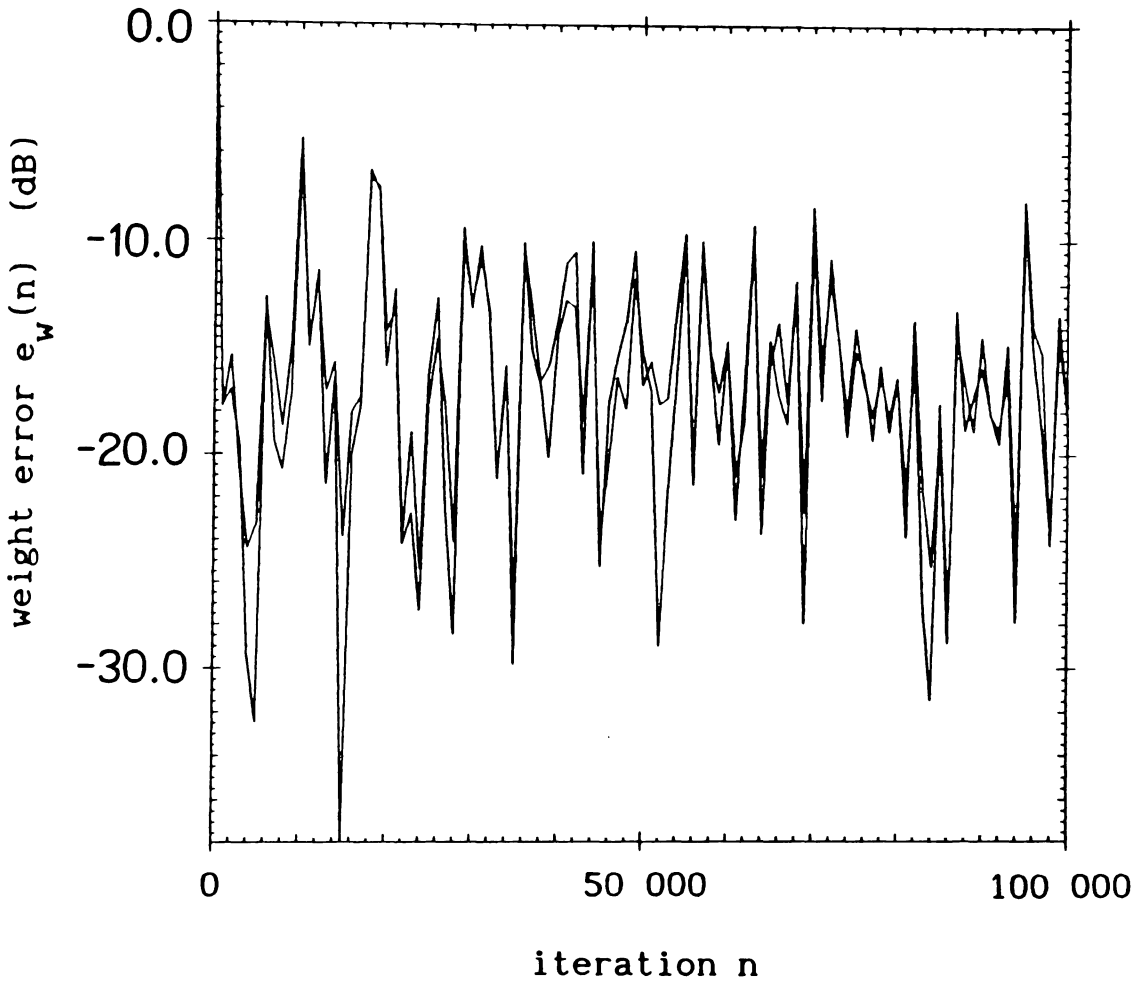

Figure 6-4. Comparison of normalized weight error trajectories for SSCRLS filter and double precision floating point CRLS filter. Results for test scenario 1, filter order 3, and forgetting factor 0.98.

Shown in Figure 6-5 are results for test scenario 2, modified to use a 12-tap MA filter with filter taps given by:

$$\mathbf{a} = [\ 0.1\ 0.2\ 0.3\ 0.4\ 0.5\ 0.4\ 0.3\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2\ ]^T \qquad (6\text{-}9)$$

The adaptive filter also had 12 taps, a forgetting factor of 0.999, and a weight vector stored with 12 bits for fractions. Since both filters converged fairly rapidly, only the first 500 iterations are plotted. The agreement in the two trajectories indicates that the SSCRLS can provide least squares convergence and performance properties.

The differences between the double precision floating point results and the 16-bit fixed point SSCRLS results are not entirely due to the stabilization techniques introduced in Chapter 4. Part of the difference is simply due to the fact that the fixed point implementation has much fewer bits of precision to work with. Thus, part of the difference is simply due to the lack of precision available to the SSCRLS filter.

6.4 Limits to stable performance

The simulation examples presented in the previous section all demonstrate the ability of the SSCRLS filter to provide stable least squares performance under a variety of conditions. In general, the stabilization techniques in Chapter 4 do not *guarantee* stability. Also, the fixed point design in Chapter 5 is not entirely immune to overflow.

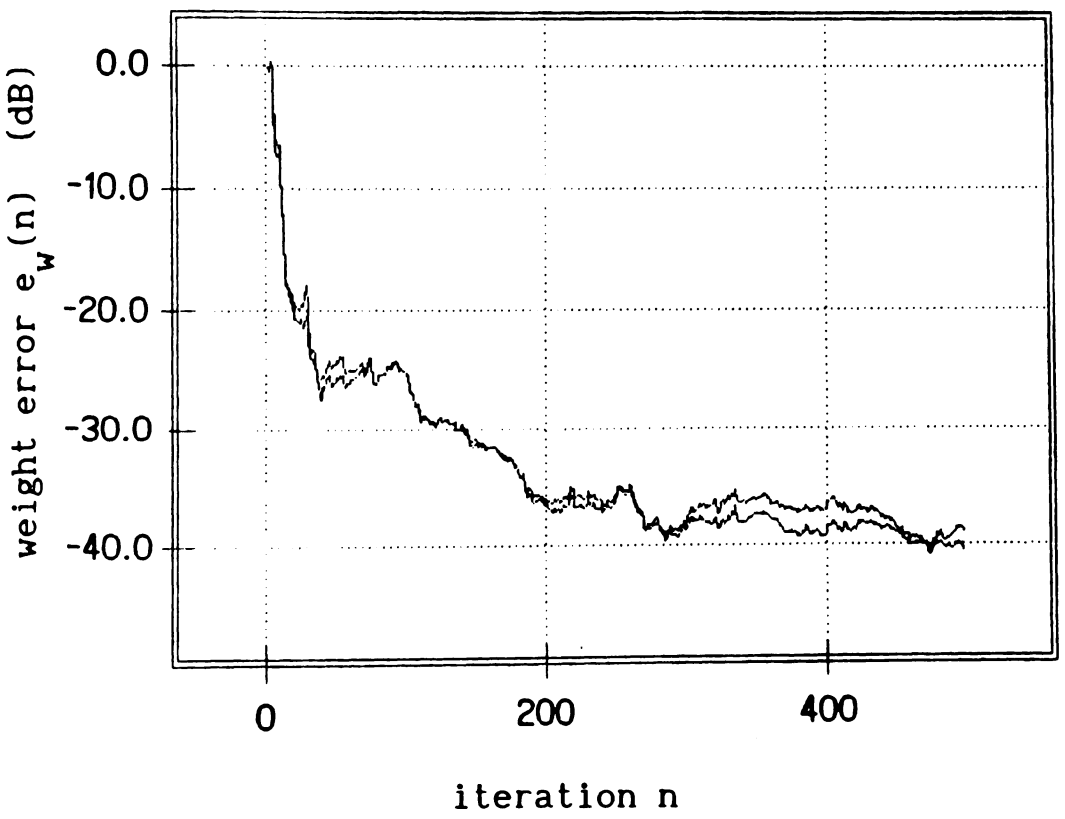

Figure 6-5. Comparison of normalized weight error trajectories for SSCRLS filter and double precision floating point CRLS filter. Results for 12-tap modification of test scenario 2, forgetting factor 0.999.

To demonstrate the latter, a simulation for test case 2 (N = 16, α = 0.9) was run using 16-bit data instead of 12-bit data (i.e. no headroom bits available in the accumulator). To keep the same non-saturation range, the A/D converter parameter vfrac was set to 13. Without any headroom bits for the accumulator, accumulator overflow occurred when computing $\mu_q(n)$ at iteration 18,238, resulting in explosive divergence.

Demonstrating explosive divergence in the absence of overflow proved to be difficult. Using test case 1 with a forgetting factor of 0.9, a 31-tap filter processing 11-bit data remained stable for all 1,000,000 iterations. Explosive divergence in the absence of overflow was eventually achieved using a constant signal quantized at the maximum 12-bit level with N = 15 and α = 0.9. Such data has an infinite condition number (since its autocorrelation matrix is singular) and lacks persistent excitation, a condition necessary for the convergence of the infinite precision CRLS algorithm [JOHN 82]. Despite this extreme situation, the 15-tap SSCRLS filter with forgetting factor 0.9 converged to one of the infinite ideal weight solutions, and $\tilde{\mu}_q(n)$ remained positive until iteration 158,597.

Thus, stability of the SSCRLS filter is not *guaranteed*. However, the likelihood of stable performance is greatly increased over the original CRLS filter.

6.5 Conclusion

In this chapter, the results of Chapters 4 and 5 are integrated together to form a stabilized fixed point implementation of the CRLS algorithm, referred to as the SSCRLS filter. Simulation tests are used to demonstrate the effectiveness of the techniques developed in Chapters 4 and 5 for preventing explosive divergence and weight lock-up. Comparisons to a double precision floating point implementation of the original CRLS filter indicate that least squares performance is only slightly degraded by the stabilization techniques introduced. While stability cannot be guaranteed, the SSCRLS filter greatly increases the set of conditions under which stable fixed point least squares filtering is possible.

CHAPTER 7 - CONCLUSION

Recursive least squares (RLS) filters lack widespread application because of the finite precision problems of explosive divergence and weight lock-up. Previous research has documented the behavior of the RLS filter when these problems occur, but little work has been done to determine the underlying mechanisms causing these problems.

This dissertation provides a detailed fixed point roundoff error analysis of the conventional RLS (CRLS) algorithm. The analysis shows exactly how roundoff error, including overflow errors, can accumulate to cause explosive divergence. The analytical results provide a basis for a set of stabilization techniques which bias the roundoff errors so that stable performance can be achieved. Thus, the solution for the problem of explosive divergence is based directly on an analytical understanding of the problem.

The problem of weight lock-up is shown to result from underflow in certain algorithmic quantities. Also, explosive divergence can result from overflow of other algorithmic quantities. This dissertation provides a detailed fixed point implementation design of the CRLS algorithm which prevents both underflow and overflow. As a result, weight lock-up and explosive divergence resulting from overflow are prevented.

The stabilization techniques and fixed point design are integrated together to provide a robust fixed point CRLS filter. While stability is not guaranteed, stable least squares performance has been demonstrated under extremely ill-conditioned signal environments.

Future work should concentrate on applying the analysis and synthesis techniques developed in this dissertation to stabilizing the fast RLS filters. So far, these filters have been stabilized using indirect means of controlling roundoff error accumulation. Also, the stabilization techniques developed in Chapter 4 should be extended to floating point implementation. Floating point has the advantage of a much simpler implementation design.

REFERENCES

[ALEX 86] Alexander, S. T., *Adaptive Signal Processing*. New York: Springer-Verlag, 1986.

[ARDA 86] Ardalan, S. H., "Floating-point roundoff error analysis of the RLS and LMS adaptive algorithms," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 1192-1208, Dec. 1986.

[ARDA 87] Ardalan, S. H. and S. T. Alexander, "Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time- varying systems," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-35, pp. 770-783, June 1987.

[ASTR 71] Astrom, K. J. and P. Eykhoff, "System identification - a survey," *Automatica*, vol. 7, pp. 123-162, 1971.

[BARN 85] Barnes, C. W., B. N. Tran, and S. H. Leung, "On the statistics of fixed point roundoff error," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-33, pp. 595-606, June 1985.

[BELL 87a] Bellanger, M. G., "Engineering aspects of recursive least squares filtering," *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Dallas, TX, pp. 2149-2152, April 1987.

[BELL 87b] Bellanger, M. G., *Adaptive Digital Filters and Signal Analysis*. New York: Marcel Dekker, 1987.

[BENA 88] Benallal, A. and A. Gilloire, "A new method to stabilize fast RLS algorithms based on a first-order model of the propagation of numerical errors," *Proc. 1988 IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, pp. 1373-1376, April 1988.

[BERT 87] Bertsekas, D. and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[BIER 77a] Bierman, G. J. and C. L. Thornton, "Numerical comparison of Kalman filter algorithms: orbit determination case study," *Automatica*, vol. 13, pp. 23-35, 1977.

[BIER 77b] Bierman, G. J., *Factorization Methods for Discrete Sequential Estimation*. New York: Academic Press, 1977.

[BOTO 87] Botto, J. -L., "Stabilization of fast recursive least squares transversal filters for adaptive filtering," *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Dallas, TX, pp. 403-406, April 1987.

[BOTO 89] Botto, J. -L. and G. V. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-37, pp. 1342-1348, Sept. 1989.

[BOTT 89] Bottomley, G. E. and S. T. Alexander, "A theoretical basis for the divergence of conventional recursive least squares filters, *Proc. 1989 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, Scotland, pp. 908-911, May 1989.

[CIOF 84] Cioffi, J. M. and T. Kailath, "Fast, RLS, transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-32, pp. 304-337, April 1984.

[CIOF 85] Cioffi, J. M. and T. Kailath, "Windowed fast transversal filters adaptive algorithms with normalization," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-33, pp. 607-625, June 1985.

[CIOF 87] Cioffi, J. M., "Limited-precision effects in adaptive filtering," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 821-833, July 1987.

[COWA 85] Cowan, C. F. N. and P. M. Grant (eds.), *Adaptive Filters*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[ELEF 84] Eleftheriou, E. and D. D. Falconer, "Restart methods for stabilizing FRLS adaptive equalizers in digital HF transmission," *Proc. GLOBECOM '84*, Atlanta, GA, paper 48-4, Nov. 1984.

[ELEF 86] Eleftheriou, E. and D. D. Falconer, "Tracking properties and steady state performance of RLS adaptive filter algorithms," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 1097-1110, October 1986.

[FABR 85] Fabre, P. and C. Gueguen, "Fast recursive least-squares algorithms: preventing divergence," *Proc. 1985 IEEE Int. Conf. Acoust., Speech, Signal Processing*, paper 30.3.

[FABR 86] Fabre, P. and C. Gueguen, "Improvement of the fast recursive least-squares algorithms via normalization: A comparative study," *IEEE Trans. Acoust., Speech, and Signal Processing*,, vol. ASSP-34, pp. 296-308, April 1986.

[GENI 68] Genin, Y, "A note on linear minimum variance estimation problems," *IEEE Trans. Aut. Control*, AC-13, p. 103, 1968.

[GIOR 85] Giordano, A. A. and F. M. Hsu, *Least Square Estimation with Applications to Digital Signal Processing*. New York: John Wiley & Sons, 1985.

[GITL 77] Gitlin, R. D. and F. R. Magee Jr., "Self-orthogonalizing adaptive equalization algorithms," *IEEE Trans. Comm.*, pp. 666-672, July 1977.

[GODA 74] Godard, D., "Channel equalization using a Kalman filter for fast data transmission," *IBM J. Res. Dev.*, May 1974, pp. 267-273.

[GOLU 83] Golub, G. H. and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 1983.

[GOOD 77] Goodwin, G. C. and R. L. Payne, *Dynamic System Identification: Experiment Design and Data Analysis*. New York: Academic Press, 1977.

[GOOD 84] Goodwin, G. C. and K. S. Sin, *Adaptive Filtering Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[GRAH 87] Graham, A., *Nonnegative Matrices and Applicable Topics in Linear Algebra*. New York: Halsted Press, 1987.

[GRAU 80] Graupe, D., V. K. Jain and J. Salahi, "A comparative analysis of various least-squares identification algorithms," *Automatica*, vol. 16, pp. 663-681, Nov. 1980.

[GRAU 84] Graupe, D., *Time Series Analysis, Identification and Adaptive Filters*. Malabar, FL: R. E. Krieger Publishing, 1984.

[HAYK 86] Haykin, S., *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[HSU 82] Hsu, F. S., "Square root Kalman filtering for high-speed data received over fading dispersive HF channels, *IEEE Trans. Info. Theory*, vol. IT-28, pp. 753-763, Sept., 1982.

[ISER 74] Isermann, R., U. Baur, W. Bamberger, P. Kneppo, and H. Siebert, "Comparison of six on-line identification and parameter estimation methods," *Automatica*, vol. 10, pp. 81-103, 1974.

[ISER 82] Isermann, R., "Parameter adaptive control algorithms - a tutorial," *Automatica*, vol. 18, no. 5, pp. 513-528, 1982.

[JOHN 82] Johnstone, R. M., C. R. Johnson, R. R. Bitmead, and B. D. O. Anderson, "Exponential convergence of recursive least squares with exponential forgetting factor," *Systems & Control Letters*, vol. 2, August 1982.

[KIM 88] Kim, D. and W. E. Alexander, "Stability analysis of the fast RLS adaptation algorithm," *Proc. 1988 IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, pp. 1361-1364, April 1988.

[KUBI 88] Kubin, G., "Stabilization of the RLS algorithm in the absence of persistent excitation, *Proc. 1988 IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, pp. 1369-1372, April 1988.

[LEE 88] Lee, E. A., "Programmable DSP Architectures: Part 1," *IEEE ASSP Magazine*, vol. 5, pp. 4-19, October 1988.

[LEON 70] Leondes, C. T. (ed.), *Theory and applications of Kalman Filtering*, NATO Advisory Group for Aerospace Research and Development, AGARDograph 139, 1970.

[LIN 84] Lin, D., "On digital implementation of the fast Kalman algorithms," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-32, pp. 998-1005, October 1984.

[LING 84] Ling, F. and J. G. Proakis, "Numerical accuracy and stability: Two problems of adaptive estimation algorithms caused by round-off error," *Proc. 1984 IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, pp. 30.3.1 - 30.3.4, March 1984.

[LING 86] Ling, F., D. Manolakis, and J. Proakis, "Finite word-length effects in RLS algorithms with application to adaptive equalization," *Annales des Telecommunications.*, vol. 41, nos. 5-6, pp. 328-336, May - June 1986.

[LJUN 83] Ljung, L. and T. Soderstrom, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

[LJUN 85] Ljung, S. and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica*, vol. 21, no. 2, pp. 157-167, March 1985.

[LJUN 86] Ljung, L., "Error propagation in adaptive algorithms with poorly exciting signals," *Annales des Telecommunications*, vol. 41, nos. 5-6, pp. 322-327, 1986.

[LJUN 87] Ljung, L., *System Identification: Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[MANO 76] Mano, M. M., *Computer System Architecture*. Englewood CLiffs, NJ: Prentice-Hall, 1976.

[MANO 87] Manolakis, D., F. Ling, and J. G. Proakis, "Efficient time-recursive least-squares algorithms for finite-memory adaptive filtering," *IEEE Trans. Circ. Syst.*, vol. CAS-34, pp. 400-407, Apr. 1987.

[MEND 87] Mendel, J. M., *Lessons in Digital Estimation Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[MOTO 85] Motorola Inc., *MC68020 32-bit Microprocessor User's Manual*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[MUEL 81] Mueller, M. S., "Least-squares algorithms for adaptive equalizers," *Bell Systems Technical Journal*, vol. 60, pp. 1905-1925, Oct. 1981.

[PAPO 65] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1965.

[PICI 78] Picinbono, B., "Adaptive signal processing for detection and communication," in *Communication Systems and Random Process Theory*, J. K. Skwirzynski (ed.), Alphen aan den Rijn, The Netherlands: Sijthoff and Noordhuff, 1978.

[PROA 83] Proakis, J. G., *Digital Communications*. New York: McGraw-Hill, 1983.

[SAKU 84] Sakurai, M. and J. Murakami, "Reduction of quantization effects in adaptive filters," *Proc. 1984 IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, pp. 30.4.1-30.4.4, March 1984.

[SAMS 83] Samson, C. G. and V. U. Reddy, "Fixed point error analysis of the normalized ladder algorithm," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-31, pp. 1177-1191, Oct. 1983.

[SARI 74] Saridis, G. N., "Comparison of six on-line identification algorithms," *Automatica*, vol. 10, pp. 69-79, 1974.

[SLOC 88] Slock, D. T. M. and T. Kailath, "Numerically stable fast recursive least-squares transversal filters," *Proc. 1988 IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, pp. 1365-1368, April 1988.

[SORE 66] Sorenson, H. W., "Kalman Filter Techniques," in C. T. Leondes (ed.), *Advances in Control Systems Theory and Applications*, vol. 3, New York: Academic Press, 1966.

[VERH 86] Verhaegen, M. and P. Van Dooren, "Numerical aspects of different Kalman filter implementations," *IEEE Trans. Automatic Control*, vol. AC-31, pp. 907-917, Oct. 1986.

[VERH 89] Verhaegen, M. H. "Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes," *Automatica*, vol. 25, pp. 437-444, 1989.

[VITE 66] Viterbi, H. A., *Principles of Coherent Communication*. New York: McGraw-Hill, 1966.

[WIDR 85] Widrow, B. and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

APPENDIX A - DERIVATION OF CRLS ERROR RECURSIONS

In section 3.1, the complete derivation of the error recursion for the error in $e_q(n|n-1)$, denoted $d_e(n|n-1)$, is presented. The complete derivations of the remaining error terms are given in this appendix.

A.1 Error recursion for $d_\mu(n)$

The derivation of the error term $d_\mu(n)$ given in (3-8b) starts with the infinite precision expression for $\mu(n)$ in equation (1-7b):

$$\mu(n) = \mathbf{x}^T(n)\, \mathbf{C}(n-1)\, \mathbf{x}(n) \qquad \text{(A-1a)}$$

Based on the order of operation specified by (3-5c), an intermediate term $\mathbf{p}(n)$ is computed first. This term is defined as:

$$\mathbf{p}(n) = \mathbf{C}(n-1)\, \mathbf{x}(n) \qquad \text{(A-1b)}$$

Applying the form of (3-3) to (A-1b) yields:

$$\mathbf{p}(n) = \mathbf{C}_q(n-1)\, \mathbf{x}(n) + \mathbf{D}_C(n-1)\, \mathbf{x}(n) \qquad \text{(A-1c)}$$

Expanding $\mathbf{C}_q(n-1)\, \mathbf{x}(n)$ using (3-1):

$$\mathbf{p}(n) = Q[\, \mathbf{C}_q(n-1)\, \mathbf{x}(n)\,] + \mathbf{r}_{m2}(n) + \mathbf{D}_C(n-1)\, \mathbf{x}(n) \qquad \text{(A-1d)}$$

where

$$\mathbf{r}_{m2}(n) = r_m[\, \mathbf{C}_q(n-1),\, \mathbf{x}(n)\,] \qquad \text{(A-1e)}$$

Using (3-5b) in (A-1d) gives:

$$\mathbf{p}(n) = \mathbf{p}_q(n) + \mathbf{r}_{m2}(n) + \mathbf{D}_C(n-1)\, \mathbf{x}(n) \qquad \text{(A-1f)}$$

From the form of (3-3), equation (A-1f) implies:

$$\mathbf{d}_p(n) = \mathbf{p}(n) - \mathbf{p}_q(n) = \mathbf{r}_{m2}(n) + \mathbf{D}_C(n-1)\, \mathbf{x}(n) \qquad \text{(A-1g)}$$

Substituting (A-1f) for $\mathbf{p}(n)$ into (A-1a) (based on (A-1b)):

$$\mu(n) = \mathbf{x}^T(n)\, [\, \mathbf{p}_q(n) + \mathbf{r}_{m2}(n) + \mathbf{D}_C(n-1)\, \mathbf{x}(n)\,] \qquad \text{(A-1h)}$$

Expanding $\mathbf{x}^T(n)\ \mathbf{p}_q(n)$ using (3-1) in (A-1h):

$$\mu(n) = Q[\ \mathbf{x}^T(n)\ \mathbf{p}_q(n)\ ] + r_{m3}(n) + \mathbf{x}^T(n)\ \mathbf{r}_{m2}(n) + \mathbf{x}^T(n)\ \mathbf{D}_C(n-1)\ \mathbf{x}(n) \qquad \text{(A-1i)}$$

where

$$r_{m3}(n) = r_m[\ \mathbf{x}^T(n)\ \mathbf{p}_q(n)\ ] \qquad \text{(A-1j)}$$

Using (3-5c) and the form of (3-3), (A-1i) yields:

$$d_\mu(n) = \mathbf{x}^T(n)\ \mathbf{D}_C(n-1)\ \mathbf{x}(n) + \mathbf{x}^T(n)\ \mathbf{r}_{m2}(n) + r_{m3}(n) \qquad \text{(A-1k)}$$

which is (3-8b) in the text.

A.2 Error recursion for $d_g(n)$

The derivation of the error term $\mathbf{d}_g(n)$ given in (3-8c) starts with the infinite precision expression for $\mathbf{g}(n)$ in equation (1-7c) with (A-1b) substituted for $\mathbf{C}(n-1)\ \mathbf{x}(n)$:

$$\mathbf{g}(n) = \frac{\mathbf{p}(n)}{\alpha + \mu(n)} \qquad \text{(A-2a)}$$

Adding and subtracting the same term in (A-2a) yields:

$$\mathbf{g}(n) = \frac{\mathbf{p}(n)}{\alpha + \mu(n)} + \frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)} - \frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)} \qquad \text{(A-2b)}$$

Expanding (A-2b) using (3-2) and (3-5d):

$$\mathbf{g}(n) = Q\left[\frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)}\right] + \mathbf{r}_{d1}(n) + \frac{\mathbf{p}(n)}{\alpha + \mu(n)} - \frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)}$$

$$= \mathbf{g}_q(n) + \mathbf{r}_{d1}(n) + \frac{\mathbf{p}(n)}{\alpha + \mu(n)} - \frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)} \qquad \text{(A-2c)}$$

where

$$\mathbf{r}_{d1}(n) = r_d[\ \mathbf{p}_q(n),\ \alpha + \mu_q(n)\ ] \qquad \text{(A-2d)}$$

Thus, based on the form in (3-3), (A-2c) implies:

$$\mathbf{d}_g(n) = \mathbf{r}_{d1}(n) + \frac{\mathbf{p}(n)}{\alpha + \mu(n)} - \frac{\mathbf{p}_q(n)}{\alpha + \mu_q(n)} \qquad \text{(A-2e)}$$

Substituting (A-1b) and (3-5b) into (A-2e):

$$\mathbf{d}_g(n) = \mathbf{r}_{d1}(n) + \frac{\mathbf{C}(n-1)\ \mathbf{x}(n)}{\alpha + \mu(n)} - \frac{Q[\ \mathbf{C}_q(n-1)\ \mathbf{x}(n)\ ]}{\alpha + \mu_q(n)} \qquad \text{(A-2f)}$$

Expanding (A-2f) using (3-1) and (A-1e):

$$\mathbf{d}_g(n) = \mathbf{r}_{d1}(n) + \frac{\mathbf{C}(n-1)\ \mathbf{x}(n)}{\alpha + \mu(n)} - \frac{\mathbf{C}_q(n-1)\ \mathbf{x}(n) - \mathbf{r}_{m2}(n)}{\alpha + \mu_q(n)}$$

$$= \mathbf{r}_{d1}(n) + \frac{\mathbf{r}_{m2}(n)}{\alpha + \mu_q(n)} + \frac{\mathbf{C}(n-1)\ \mathbf{x}(n)}{\alpha + \mu(n)} - \frac{\mathbf{C}_q(n-1)\ \mathbf{x}(n)}{\alpha + \mu_q(n)}$$

$$= \frac{\mathbf{C}(n-1)\ \mathbf{x}(n)}{\alpha + \mu(n)} - \frac{\mathbf{C}_q(n-1)\ \mathbf{x}(n)}{\alpha + \mu_q(n)} + \frac{\mathbf{r}_{m2}(n)}{\alpha + \mu_q(n)} + \mathbf{r}_{d1}(n)$$

(A-2g)

which is (3-8c) in the text.

A.3 Error recursion for $\mathbf{d}_w(n)$

The derivation of the error term $\mathbf{d}_w(n)$ given in (3-8d) starts with the infinite precision expression for $\mathbf{w}(n)$ in equation (1-7d):

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)\ e(n|n-1) \qquad \text{(A-3a)}$$

Adding and subtracting the same term in (A-3a) gives:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}_q(n)\, e_q(n|n-1) + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1) \qquad \text{(A-3b)}$$

Expanding (A-3b) using (3-1), (3-3), and (3-5e):

$$\begin{aligned}\mathbf{w}(n) &= \mathbf{w}_q(n-1) + \mathbf{d}_w(n-1) + Q[\, \mathbf{g}_q(n)\, e_q(n|n-1)\,] + \mathbf{r}_{m4}(n) \\ &\quad + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1) \\ &= \mathbf{w}_q(n) + \mathbf{d}_w(n-1) + \mathbf{r}_{m4}(n) + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1)\end{aligned} \qquad \text{(A-3c)}$$

where

$$\mathbf{r}_{m4}(n) = r_m[\, \mathbf{g}_q(n),\, e_q(n|n-1)\,] \qquad \text{(A-3d)}$$

Thus, equation (A-3c) gives:

$$\mathbf{d}_w(n) = \mathbf{d}_w(n-1) + \mathbf{r}_{m4}(n) + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1) \qquad \text{(A-3e)}$$

which is (3-8d) in the text.

A.4 Error recursion for $D_C(n)$

The derivation of the error term $D_C(n)$ given in (3-8e) starts with the definition of $\mathbf{b}^T(n)$:

$$\mathbf{b}^T(n) = \mathbf{x}^T(n)\, \mathbf{C}(n-1) \qquad \text{(A-4a)}$$

Substituting this expression into (1-7e) gives:

$$\mathbf{C}(n) = 1/\alpha\, [\, \mathbf{C}(n-1) - \mathbf{g}(n)\, \mathbf{b}^T(n)\,] \qquad \text{(A-4b)}$$

Adding and subtracting the same term in (A-4b) yields:

$$\mathbf{C}(n) = 1/\alpha\, [\, \mathbf{C}(n-1) - \mathbf{g}_q(n)\, \mathbf{b}_q^T(n) + \mathbf{g}_q(n)\, \mathbf{b}_q^T(n) - \mathbf{g}(n)\, \mathbf{b}^T(n)\,] \qquad \text{(A-4c)}$$

As far as $\mathbf{b}_q(n)$ is concerned, applying form (3-3) to the definition of $\mathbf{b}^T(n)$ in (A-4a) gives:

$$\mathbf{b}^T(n) = \mathbf{x}^T(n) \left[\mathbf{C}_q(n-1) + \mathbf{D}_C(n-1) \right] \quad \text{(A-4d)}$$

Expanding (A-4d) using (3-1) yields:

$$\mathbf{b}^T(n) = Q\left[\mathbf{x}^T(n)\, \mathbf{C}_q(n-1) \right] + \mathbf{r}_{m5}{}^T(n) + \mathbf{x}^T(n)\, \mathbf{D}_C(n-1) \quad \text{(A-4e)}$$

where

$$\mathbf{r}_{m5}{}^T(n) = r_m\left[\mathbf{x}^T(n),\, \mathbf{C}_q(n-1) \right] \quad \text{(A-4f)}$$

Substituting (3-5f) in (A-4e) gives:

$$\mathbf{b}^T(n) = \mathbf{b}_q{}^T(n) + \mathbf{r}_{m5}{}^T(n) + \mathbf{x}^T(n)\, \mathbf{D}_C(n-1) \quad \text{(A-4g)}$$

From the form of (3-3), (A-4g) gives:

$$\mathbf{d}_b{}^T(n) = \mathbf{r}_{m5}{}^T(n) + \mathbf{x}^T(n)\, \mathbf{D}_C(n-1) \quad \text{(A-4h)}$$

Next, applying the form of (3-3) and (3-1) to (A-4c):

$$\mathbf{C}(n) = 1/\alpha \left[\mathbf{C}_q(n-1) + \mathbf{D}_C(n-1) - Q\left[\mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) \right] - \mathbf{R}_{m6}(n) \right]$$
$$+ \; 1/\alpha \left[\mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) - \mathbf{g}(n)\, \mathbf{b}^T(n) \right] \quad \text{(A-4i)}$$

where

$$\mathbf{R}_{m6}(n) = r_m\left[\mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) \right] \quad \text{(A-4j)}$$

Expanding (A-4i) using (3-2) yields:

$$\mathbf{C}(n) = Q\left[\frac{\mathbf{C}_q(n-1) - Q\left[\mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) \right]}{\alpha} \right] + \mathbf{R}_{d2}(n)$$
$$+ \; 1/\alpha \left[\mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) + \mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) - \mathbf{g}(n)\, \mathbf{b}^T(n) \right] \quad \text{(A-4k)}$$

where

$$\mathbf{R}_{d2}(n) = r_d\left[\mathbf{C}_q(n-1) - Q\left[\mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) \right],\, \alpha \right] \quad \text{(A-4l)}$$

Using (3-5g) and the form in (3-3), (A-4k) implies:

$$\mathbf{D}_C(n) = 1/\alpha \left[\mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) + \mathbf{g}_q(n)\, \mathbf{b}_q{}^T(n) - \mathbf{g}(n)\, \mathbf{b}^T(n) \right]$$
$$+ \; \mathbf{R}_{d2}(n) \quad \text{(A-4m)}$$

Substituting (A-4a) and (3-5f) in (A-4m) gives:

$$\mathbf{D}_C(n) = \mathbf{R}_{d2}(n) + 1/\alpha \, [\, \mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) \,]$$

$$+ 1/\alpha \, [\, \mathbf{g}_q(n) \, \{ \, Q[\, \mathbf{x}^T(n) \, \mathbf{C}_q(n-1) \,] \} - \mathbf{g}(n) \, \mathbf{x}^T(n) \, \mathbf{C}(n-1)]$$

(A-4n)

Expanding (A-4n) using (3-1) and (A-4f) gives:

$$\mathbf{D}_C(n) = \mathbf{R}_{d2}(n) + 1/\alpha \, [\, \mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) \,]$$

$$+ 1/\alpha \, [\, \mathbf{g}_q(n) \, \{ \, \mathbf{x}^T(n) \, \mathbf{C}_q(n-1) - \mathbf{r}^T_{m5}(n) \, \} - \mathbf{g}(n) \, \mathbf{x}^T(n) \, \mathbf{C}(n-1)]$$

$$= \mathbf{R}_{d2}(n) + 1/\alpha \, [\, \mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) \,]$$

$$+ 1/\alpha \, [\, \mathbf{g}_q(n) \, \mathbf{x}^T(n) \, \mathbf{C}_q(n-1) - \mathbf{g}(n) \, \mathbf{x}^T(n) \, \mathbf{C}(n-1) \,]$$

$$- 1/\alpha \, \mathbf{g}_q(n) \, \mathbf{r}^T_{m5}(n) \qquad \text{(A-4o)}$$

Re-arranging terms in (A-4o) gives:

$$\mathbf{D}_C(n) = \mathbf{R}_{d2}(n) + 1/\alpha \, [\, \mathbf{D}_C(n-1) - \mathbf{R}_{m6}(n) - \mathbf{g}_q(n) \, \mathbf{r}^T_{m5}(n) \,]$$

$$+ 1/\alpha \, [\, \mathbf{g}_q(n) \, \mathbf{x}^T(n) \, \mathbf{C}_q(n-1) - \mathbf{g}(n) \, \mathbf{x}^T(n) \, \mathbf{C}(n-1) \,] \qquad \text{(A-4p)}$$

which is equation (3-8e) in the text.

This completes the derivations of (3-8b) through (3-8e).

APPENDIX B - DERIVATION OF STEADY-STATE APPROXIMATIONS

In this appendix, the steady-state approximations given in (3-20) are derived, using (3-16) through (3-19). First, equation (1-7e) is manipulated to obtain:

$$\mathbf{g}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) \;=\; \mathbf{C}(n-1) - \alpha\ \mathbf{C}(n) \tag{B-1}$$

Using (3-16) and (3-18) on the right side of (B-1) gives:

$$\mathbf{g}(n)\ \mathbf{x}^T(n)\ \mathbf{C}(n-1) \;\cong\; (1-\alpha)^2\ \mathbf{R}_{xx}^{-1} \tag{B-2}$$

which is (3-20a). Using (3-16) and (3-18) in (B-2) gives:

$$\mathbf{g}(n)\ \mathbf{x}^T(n) \;\cong\; (1-\alpha)\ \mathbf{I} \tag{B-3}$$

Finally, substituting (1-7c) for $\mathbf{g}(n)$ into (B-3) and using (3-16) and (3-18) gives:

$$\mathbf{x}(n)\ \mathbf{x}^T(n)/\ (\alpha + \mu(n)) \;\cong\; \mathbf{R}_{xx} \tag{B-4}$$

which is (3-20b). Using (3-17) and (3-19) with (1-7d) gives:

$$\mathbf{g}(n)\ e(n|n-1) \;\cong\; \mathbf{0} \tag{B-5}$$

which is (3-20c). Substituting (1-7c) for $\mathbf{g}(n)$ into (B-5) gives:

$$\frac{\mathbf{C}(n-1)\ \mathbf{x}(n)}{\alpha\ +\ \mu(n)}\ e(n|n-1) \;\cong\; \mathbf{0} \tag{B-6}$$

Using (3-16) and (3-18) in (B-6) gives:

$$\frac{(1-\alpha)\ \mathbf{R}_{xx}^{-1}\ \mathbf{x}(n)\ e(n|n-1)}{\alpha\ +\ \mu(n)} \;\cong\; \mathbf{0} \tag{B-7}$$

Left multiplying both sides by $(1-\alpha)^{-1}\ \mathbf{R}_{xx}$ gives:

$$\frac{\mathbf{x}(n)\ e(n|n-1)}{\alpha\ +\ \mu(n)} \;\cong\; \mathbf{0} \tag{B-8}$$

which is (3-20d).

APPENDIX C - DERIVATION OF SCRLS ERROR RECURSIONS

In this appendix, the error recursions for the SCRLS algorithm are derived. First, the derivation of $d_e(n|n-1)$ is the same as that presented in Chapter 3, section 3.1. Also, the derivations for $\mathbf{d}_p(n)$ and $d_\mu(n)$ are the same as those given in Appendix A. Thus, the first new error recursion to be derived is for $d_\beta(n)$.

C.1 Error recursion for $d_\beta(n)$

An expression for $d_\beta(n)$ is derived by starting with (4-4):

$$\beta(n) = 1/(\alpha + \mu(n)) \tag{C-1a}$$

Adding and subtracting the same term on the right side of (C-1a) gives:

$$\beta(n) = 1/(\alpha + \mu_q(n)) + 1/(\alpha + \mu(n)) - 1/(\alpha + \mu_q(n)) \tag{C-1b}$$

Expanding (C-1b) using (3-2) gives:

$$\beta(n) = Q\left[\frac{1}{\alpha + \mu_q(n)} \right] + r_{d1}(n) + \frac{1}{\alpha + \mu(n)} - \frac{1}{\alpha + \mu_q(n)} \tag{C-1c}$$

where

$$r_{d1}(n) = r_d[\, 1,\ \alpha + \mu_q(n) \,] \tag{C-1d}$$

From (4-7d), the first term on the right side of (C-1c) is $\beta_q(n)$. Substituting (C-1a) in (C-1c) and using the form in (3-3):

$$d_\beta(n) = r_{d1}(n) + \beta(n) - 1/(\alpha + \mu_q(n)) \tag{C-1e}$$

C.2 Error recursion for $d_g(n)$

The derivation of $d_g(n)$ starts with the expression for $g(n)$ based on (4-7e):

$$g(n) = \beta(n)\, p(n) \tag{C-2a}$$

Adding and subtracting the same term on the right side of (C-2a) gives:

$$g(n) = \beta_q(n)\, p_q(n) + \beta(n)\, p(n) - \beta_q(n)\, p_q(n) \tag{C-2b}$$

Expanding (C-2b) using (3-1) and (C-2a),

$$g(n) = Q\Big[\, \beta_q(n)\, p_q(n) \,\Big] + r_{m4}(n) + g(n) - \beta_q(n)\, p_q(n) \tag{C-2c}$$

where

$$r_{m4}(n) = r[\, \beta_q(n)\, p_q(n) \,] \tag{C-2d}$$

Substituting (4-7e) in (C-2c) and using the form in (3-3),

$$d_g(n) = r_{m4}(n) + g(n) - \beta_q(n)\, p_q(n) \tag{C-2e}$$

C.3 Error recursion for $d_w(n)$

The derivation of $d_w(n)$ starts with the infinite precision expression for $w(n)$ based on (4-7f):

$$w(n) = w(n-1) + g(n)\, e(n|n-1) \tag{C-3a}$$

Adding and subtracting the same term on the right side of (C-3a) gives:

$$w(n) = w(n-1) + g_q(n)\, e_q(n|n-1) + g(n)\, e(n|n-1) - g_q(n)\, e_q(n|n-1) \tag{C-3b}$$

Expanding (C-3b) using (3-1) and the form in (3-3) gives:

$$\mathbf{w}(n) = \mathbf{w}_q(n-1) + Q\left[\mathbf{g}_q(n)\, e_q(n|n-1) \right] + \mathbf{r}_{m5}(n) + \mathbf{d}_w(n-1) + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1) \qquad \text{(C-3c)}$$

where

$$\mathbf{r}_{m5}(n) = r_m[\, \mathbf{g}_q(n)\, e_q(n|n-1) \,] \qquad \text{(C-3d)}$$

Substituting (4-7f) in (C-3c) and using the form in (3-3) gives:

$$\mathbf{d}_w(n) = \mathbf{d}_w(n-1) + \mathbf{r}_{m5}(n) + \mathbf{g}(n)\, e(n|n-1) - \mathbf{g}_q(n)\, e_q(n|n-1) \qquad \text{(C-3e)}$$

C.4 Error recursion for $D_C(n)$

The derivation of the error recursion for $D_C(n)$ starts with the infinite precision equivalent of (4-7g):

$$\mathbf{C}(n) = 1/\alpha \left(\mathbf{C}(n-1) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n) \right) \qquad \text{(C-4a)}$$

Adding and subtracting the same term on the right side of (C-4a) gives:

$$\mathbf{C}(n) = 1/\alpha \left(\mathbf{C}(n-1) - \beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q{}^T(n) \right) + 1/\alpha \left(\beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q{}^T(n) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n) \right) \qquad \text{(C-4b)}$$

Expanding the first $\mathbf{p}_q(n)\, \mathbf{p}_q{}^T(n)$ term in (C-4b) using (3-1) gives:

$$\mathbf{C}(n) = 1/\alpha \left(\mathbf{C}(n-1) - \beta_q(n)\, Q\left[\mathbf{p}_q(n)\, \mathbf{p}_q{}^T(n) \right] - \beta_q(n)\, \mathbf{R}_{m6}(n) \right) + 1/\alpha \left(\beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q{}^T(n) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n) \right) \qquad \text{(C-4c)}$$

where

$$\mathbf{R}_{m6}(n) = r_m[\, \mathbf{p}_q(n)\, \mathbf{p}_q{}^T(n) \,] \qquad \text{(C-4d)}$$

Expanding (C-4c) using (3-1) again and the form in (3-3) yields:

$$\mathbf{C}(n) = 1/\alpha \left(\mathbf{C}_q(n-1) - Q\left[\beta_q(n)\, Q\left[\mathbf{p}_q(n)\, \mathbf{p}_q^T(n) \right] \right] \right) + 1/\alpha \left(\mathbf{D}_C(n-1) - \mathbf{R}_{m7}(n) - \beta_q(n)\, \mathbf{R}_{m6}(n) \right) + 1/\alpha \left(\beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n) \right) \qquad \text{(C-4e)}$$

where

$$\mathbf{R}_{m7}(n) = r_m[\, \beta_q(n)\, Q[\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n)\,]\,] \qquad \text{(C-4f)}$$

Applying (3-1) one last time to (C-4e) gives:

$$\mathbf{C}(n) = Q\left[1/\alpha \left(\mathbf{C}_q(n-1) - Q\left[\beta_q(n)\, Q\left[\mathbf{p}_q(n)\, \mathbf{p}_q^T(n) \right] \right] \right) \right] + \mathbf{R}_{m8}(n) + 1/\alpha \left(\mathbf{D}_C(n-1) - \mathbf{R}_{m7}(n) - \beta_q(n)\, \mathbf{R}_{m6}(n) \right) + 1/\alpha \left(\beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n) \right) \qquad \text{(C-4g)}$$

where

$$\mathbf{R}_{m8}(n) = r_m[\, (1/\alpha),\, \{\, \mathbf{C}_q(n-1) - Q[\, \beta_q(n)\, Q[\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n)\,]\,]\, \}\,] \qquad \text{(C-4h)}$$

From (4-7g) and (3-3), equation (C-4g) implies:

$$\mathbf{D}_C(n) = \mathbf{R}_{m8}(n) + 1/\alpha \left(\mathbf{D}_C(n-1) - \mathbf{R}_{m7}(n) - \beta_q(n)\, \mathbf{R}_{m6}(n) \right) + 1/\alpha \left(\beta_q(n)\, \mathbf{p}_q(n)\, \mathbf{p}_q^T(n) - \beta(n)\, \mathbf{p}(n)\, \mathbf{p}^T(n) \right) \qquad \text{(C-4i)}$$

APPENDIX D - NORMALIZATION THEORY

In this appendix, several normalizations of the CRLS algorithm with respect to the **C** matrix are derived. Normalizing the **C** matrix is used to prevent underflow (causing weight lock-up) and overflow (causing explosive divergence). Starting with the general RLS algorithm (order N^3), four equivalent normalized forms of CRLS are derived. The relationship between the normalized forms and the original form is discussed.

The CRLS algorithm can be normalized by considering its "parent" algorithm, the order N^3 general RLS algorithm. This algorithm is given as follows [ALEX 86]:

Initial conditions:

$$\mathbf{R}(0) = 1/\delta\ \mathbf{I} \tag{D-1a}$$

$$\mathbf{r}(0) = \mathbf{0} \tag{D-1b}$$

For n = 1, 2, ... **do:**

$$\mathbf{R}(n) = \alpha\ \mathbf{R}(n-1) + \mathbf{x}(n)\ \mathbf{x}^T(n) \tag{D-2a}$$

$$\mathbf{r}(n) = \alpha\ \mathbf{r}(n-1) + d(n)\ \mathbf{x}(n) \tag{D-2b}$$

$$\mathbf{w}(n) = \mathbf{R}^{-1}(n)\ \mathbf{r}(n) \tag{D-2c}$$

To derive the CRLS algorithm, the matrix inversion lemma is applied to (D-2) [ALEX 86].

To derive normalized forms of CRLS, the equations in (D-2) are manipulated slightly before applying the matrix inversion lemma. The manipulation is such that the solution for **w**(n) at each iteration is the same as that given above. First, the right side of (D-2c) is

multiplied by 1, represented as s/s, where s is a nonzero, positive scaling value. This gives:

$$\begin{aligned} \mathbf{w}(n) &= s/s\ \mathbf{R}^{-1}(n)\ \mathbf{r}(n) \\ &= \left(s\ \mathbf{R}(n) \right)^{-1} s\ \mathbf{r}(n) \\ &= \tilde{\mathbf{R}}^{-1}(n)\ \tilde{\mathbf{r}}(n) \end{aligned} \qquad \text{(D-3)}$$

where

$$\tilde{\mathbf{R}}(n) = s\ \mathbf{R}(n) \qquad \text{(D-4a)}$$

$$\tilde{\mathbf{r}}(n) = s\ \mathbf{r}(n) \qquad \text{(D-4b)}$$

Using (D-4a) and (D-4b) with (D-2a) and (D-2b) gives:

$$\begin{aligned} \tilde{\mathbf{R}}(n) &= \alpha\ \tilde{\mathbf{R}}(n-1) + s\ \mathbf{x}(n)\ \mathbf{x}^{T}(n) \\ &= \alpha\ \tilde{\mathbf{R}}(n-1) + \left(s^{1/2}\ \mathbf{x}(n) \right) \left(s^{1/2}\ \mathbf{x}(n) \right)^{T} \end{aligned} \qquad \text{(D-5a)}$$

$$\begin{aligned} \tilde{\mathbf{r}}(n) &= \alpha\ \tilde{\mathbf{r}}(n-1) + s\ d(n)\ \mathbf{x}(n) \\ &= \alpha\ \tilde{\mathbf{r}}(n-1) + \left(s^{1/2}\ d(n) \right) \left(s^{1/2}\ \mathbf{x}(n) \right) \end{aligned} \qquad \text{(D-5b)}$$

From (D-1a) and (D-4a), $\tilde{\mathbf{R}}(n)$ is initialized to $s/\delta\ \mathbf{I}$.

To derive a normalized CRLS algorithm, the matrix inversion lemma and steps detailed in [ALEX 86] are applied to (D-5a), (D-5b) and (D-3). An equivalent derivation results from recognizing that these equations are almost identical to (D-2a), (D-2b), and (D-2c), except that the quantities d(n) and $\mathbf{x}(n)$ have been replaced by $s^{1/2}\ d(n)$ and $s^{1/2}\ \mathbf{x}(n)$ respectively. Making the same substitution into (1-7), with slight manipulation, yields:

Initial conditions:

$$\mathbf{w}(0) = \mathbf{x}(n) = \mathbf{0} \qquad \text{(D-6a)}$$

$$\tilde{\mathbf{C}}(0) = 1/s\ \delta\ \mathbf{I} \qquad (\delta >> 1) \qquad \text{(D-6b)}$$

For n = 1, 2, ... **do:**

$$\tilde{\mathbf{x}}(n) = s^{1/2}\,\mathbf{x}(n) \tag{D-7a}$$

$$\tilde{d}(n) = s^{1/2}\,d(n) \tag{D-7b}$$

$$\tilde{e}(n|n-1) = \tilde{d}(n) - \tilde{\mathbf{x}}^T(n)\,\mathbf{w}(n-1) \tag{D-7c}$$

$$\mu(n) = \tilde{\mathbf{x}}^T(n)\,\tilde{\mathbf{C}}(n-1)\,\tilde{\mathbf{x}}(n) \tag{D-7d}$$

$$\tilde{\mathbf{g}}(n) = \frac{\tilde{\mathbf{C}}(n-1)\,\tilde{\mathbf{x}}(n)}{\alpha + \mu(n)} \tag{D-7e}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \tilde{\mathbf{g}}(n)\,\tilde{e}(n|n-1) \tag{D-7f}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\,\tilde{\mathbf{C}}(n-1) - \tilde{\mathbf{g}}(n)\,\tilde{\mathbf{x}}^T(n)\,\tilde{\mathbf{C}}(n-1)\,\right] \tag{D-7g}$$

where the normalized values are related to the original values in (1-7) as follows:

$$\tilde{e}(n|n-1) = s^{1/2}\,e(n|n-1) \tag{D-8a}$$

$$\tilde{\mathbf{g}}(n) = s^{-1/2}\,\mathbf{g}(n) \tag{D-8b}$$

$$\tilde{\mathbf{C}}(n) = 1/s\;\;\mathbf{C}(n) \tag{D-8c}$$

This form, referred to as form 1a, indicates that normalization can be performed by simply scaling the incoming data and desired signals. The scaling by $s^{1/2}$ of the data causes the computed $\tilde{\mathbf{C}}(n)$ matrix to be the original $\mathbf{C}(n)$ matrix scaled by 1/s.

By using (D-8a) and (D-8b) to manipulate (D-7c) and (D-7e), the following equivalent form, denoted form 1b, can be obtained:

Initial conditions:

$$\mathbf{w}(0) = \mathbf{x}(n) = \mathbf{0} \tag{D-9a}$$

$$\tilde{\mathbf{C}}(0) = 1/s\;\delta\;\mathbf{I} \qquad (\delta \gg 1) \tag{D-9b}$$

For n = 1, 2, ... **do:**

$$\tilde{\mathbf{x}}(n) = s^{1/2}\,\mathbf{x}(n) \tag{D-10a}$$

$$e(n|n-1) = d(n) - \mathbf{x}^T(n)\,\mathbf{w}(n-1) \tag{D-10b}$$

$$\mu(n) = \tilde{\mathbf{x}}^T(n)\,\tilde{\mathbf{C}}(n-1)\,\tilde{\mathbf{x}}(n) \tag{D-10c}$$

$$\mathbf{g}(n) = s^{1/2}\,\frac{\tilde{\mathbf{C}}(n-1)\,\tilde{\mathbf{x}}(n)}{\alpha + \mu(n)} \tag{D-10d}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)\,e(n|n-1) \tag{D-10e}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\,\tilde{\mathbf{C}}(n-1) - s^{-1/2}\,\mathbf{g}(n)\,\tilde{\mathbf{x}}^T(n)\,\tilde{\mathbf{C}}(n-1)\,\right] \tag{D-10f}$$

Equation (D-8c) still indicates how the normalized $\tilde{\mathbf{C}}$ matrix relates to the original **C** matrix. Since $\tilde{d}(n)$ is not used, equation (D-7b) has been omitted.

The same normalization approach in forms 1a and 1b can be applied to the SCRLS algorithm given by (4-1) and (4-2). Equations (D-7e) and (D-7g) can be manipulated to obtain:

$$\tilde{\mathbf{p}}(n) = \tilde{\mathbf{C}}(n-1)\,\tilde{\mathbf{x}}(n) \tag{D-11a}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\,\tilde{\mathbf{C}}(n-1) - \frac{\tilde{\mathbf{p}}(n)\,\tilde{\mathbf{p}}^T(n)}{\alpha + \mu(n)}\,\right] \tag{D-11b}$$

where (D-8c) holds and

$$\tilde{\mathbf{p}}(n) = s^{-1/2}\,\mathbf{p}(n) \tag{D-12}$$

The same equations apply to form 1b.

Another equivalent form can be derived by manipulating (D-10) so that the powers of s appear next to the $\tilde{\mathbf{C}}(n-1)$ terms. This form, referred to as form 2, is given by:

Initial conditions:

$$\mathbf{w}(0) = \mathbf{x}(n) = \mathbf{0} \tag{D-13a}$$

$$\tilde{\mathbf{C}}(0) = 1/s\ \delta\ \mathbf{I} \qquad (\delta >> 1) \tag{D-13b}$$

For n = 1, 2, ... **do:**

$$e(n|n-1) = d(n) - \mathbf{x}^T(n)\ \mathbf{w}(n-1) \tag{D-14a}$$

$$\mu(n) = \mathbf{x}^T(n)\ s\ \tilde{\mathbf{C}}(n-1)\ \mathbf{x}(n) \tag{D-14b}$$

$$\mathbf{g}(n) = \frac{s\ \tilde{\mathbf{C}}(n-1)\ \mathbf{x}(n)}{\alpha + \mu(n)} \tag{D-14c}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)\ e(n|n-1) \tag{D-14d}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\ \tilde{\mathbf{C}}(n-1) - \mathbf{g}(n)\ \mathbf{x}^T(n)\ \tilde{\mathbf{C}}(n-1)\ \right] \tag{D-14e}$$

where (D-8c) still holds. But (D-14) is simply (1-7) with (D-8c) substituted for $\mathbf{C}(n)$. Thus, normalization can be viewed as factoring out s from $\mathbf{C}(n)$ to give $\tilde{\mathbf{C}}(n)$. This factor need not be the same at each iteration. However, if s is made adaptive (s(n)), then $\tilde{\mathbf{C}}(n)$ must be re-scaled at each iteration such that (D-8c) holds.

The same normalization approach in form 2 can also be applied to the SCRLS algorithm. Equation (D-14e) can be manipulated to obtain:

$$\mathbf{p}(n) = s\ \tilde{\mathbf{C}}(n-1)\ \mathbf{x}(n) \tag{D-15a}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\ \tilde{\mathbf{C}}(n-1) - s^{-1}\ \frac{\mathbf{p}(n)\ \mathbf{p}^T(n)}{\alpha + \mu(n)}\ \right] \tag{D-15b}$$

where (D-8c) holds and $\mathbf{p}(n)$ is not normalized.

Finally, (D-14b) and (D-14c) can be manipulated to obtain form 3:

Initial conditions:

$$\mathbf{w}(0) = \mathbf{x}(n) = \mathbf{0} \tag{D-16a}$$

$$\tilde{\mathbf{C}}(0) = 1/s\ \delta\ \mathbf{I} \qquad (\delta >> 1) \tag{D-16b}$$

For n = 1, 2, ... **do:**

$$e(n|n-1) = d(n) - \mathbf{x}^T(n)\ \mathbf{w}(n-1) \tag{D-17a}$$

$$\tilde{\mu}(n) = \mathbf{x}^T(n)\ \tilde{\mathbf{C}}(n-1)\ \mathbf{x}(n) \tag{D-17b}$$

$$\mathbf{g}(n) = \frac{\tilde{\mathbf{C}}(n-1)\ \mathbf{x}(n)}{\alpha/s + \tilde{\mu}(n)} \tag{D-17c}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)\ e(n|n-1) \tag{D-17d}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\ \tilde{\mathbf{C}}(n-1) - \mathbf{g}(n)\ \mathbf{x}^T(n)\ \tilde{\mathbf{C}}(n-1)\ \right] \tag{D-17e}$$

where (D-8c) holds and

$$\tilde{\mu}(n) = 1/s\ \mu(n) \tag{D-18}$$

Notice that this form is the same as (1-7), except the initialization of $\tilde{\mathbf{C}}(n)$ and the denominator of $\mathbf{g}(n)$. This indicates that the term added to $\mu(n)$ in the denominator of $\mathbf{g}(n)$ does not have to be the forgetting factor α. Changing this value simply scales the $\mathbf{C}(n)$ matrix according to (D-8c). The exponential weighting of past errors given in (1-5) is still preserved. Thus, the "true" forgetting factor appears only in (D-17e) in the term $1/\alpha$.

Applying the same normalization approach to the SCRLS algorithm yields:

$$\tilde{\mathbf{p}}(n) = \tilde{\mathbf{C}}(n-1)\ \mathbf{x}(n) \tag{D-19a}$$

$$\tilde{\mathbf{C}}(n) = \frac{1}{\alpha}\left[\ \tilde{\mathbf{C}}(n-1) - \frac{\tilde{\mathbf{p}}(n)\ \tilde{\mathbf{p}}^T(n)}{\alpha/s + \tilde{\mu}(n)}\ \right] \tag{D-19b}$$

where

$$\tilde{\mathbf{p}}(n) = s^{-1}\ \mathbf{p}(n) \tag{D-20}$$

and (D-8c) holds.

In summary, four equivalent ways of normalizing the **C** matrix in the CRLS algorithm have been derived. Several conclusions have been drawn:

1. Scaling the data by a factor $s^{1/2}$ effectively scales the **C**(n) matrix by a factor 1/s.
2. The scaling factor can be changed at each iteration as long as $\tilde{\mathbf{C}}$(n) is re-scaled so that (D-8c) holds.
3. In the original CRLS algorithm given in (1-7), the α that appears in the denominator of **g**(n) can be replaced by an arbitrary nonzero positive constant. The effect is simply to scale **C**(n). The exponential weighting and least squares solution are both preserved.

It should be noted that s could be a negative number. However, this would cause the normalized $\tilde{\mathbf{C}}$ matrix to be always negative definite instead of positive definite. To keep the property of positive definiteness, only positive values for s are considered.