Australian Telecommunications Research Institute

Numerical Properties of Adaptive Recursive Least-Squares (RLS) Algorithms with Linear Constraints

Jia Quan Huo

This thesis is presented as part of the requirements for the award of the
Degree of Master of Engineering
of
Curtin University of Technology

March 9, 1999

Acknowledgments

I would like to express my sincere thanks to Dr. Yee Hong Leung for his guidance and enormous patience to correct the erros in my works.

Also thanks to all the staffs at ATRI who have been helping me all through these two years of my study.

I am also grateful to my family for their support and understanding.

Abstract

Adaptive filters have found applications in many signal processing problems. In some situations, linear constraints are imposed on the filter weights such that the filter is forced to exhibit a certain desired response. Several algorithms for linearly constrained least-squares adaptive filtering have been developed in the literature. When implemented with finite precision arithmetic, these algorithms are inevitably subjected to rounding errors. It is essential to understand how these algorithms react to rounding errors.

In this thesis, the numerical properties of three linearly constrained least-squares adaptive filtering algorithms, namely, the linearly constrained fast least algorithm, the linear systolic array for MVDR beamforming and the linearly constrained QRD-RLS algorithm, are studied. It is shown that all these algorithms can be separated into a constrained part and an unconstrained part. The numerical properties of unconstrained least-squares algorithms (i.e., the unconstrained part of the linearly constrained algorithms under study) are reviewed from the perspectives of error propagation, error accumulation and numerical persistency. It is shown that persistent excitation and sufficient numerical resolution are needed to ensure the stability of the CRLS algorithm, while the QRD-RLS algorithm is unconditionally stable. The numerical properties of the constrained algorithms are then examined. Based on the technique of how the constraints are applied, these algorithms can be grouped into two categories. The first two algorithms admit a similar structure in that the unconstrained parts preceed the constrained parts. Error propagation analysis shows that this structure gives rise to unstable error propagation in the constrained part. In contrast, the constrained part of the third algorithm preceeds the unconstrained part. It is shown that this algorithm gives an exact solution to a linearly constrained least-squares adaptive filtering problem with perturbed constraints and perturbed input data. A minor modification to the constrained part of the linearly constrained QRD-RLS algorithm is proposed to avoid a potential numerical difficulty due to the Gaussian elimination operation employed in the algorithm.

1 Introduction

1.1 Linearly Constrained Adaptive Filters

1.1.1 Adaptive Filters

Adaptive filters have found applications in many signal processing problems due to their ability to provide satisfactory performance in unknown stationary environments and to track variations in the input signal statistics in non-stationary environments [8, 22-25, 42, 54]. A generic adaptive filter configuration is shown in Figure 1.1.

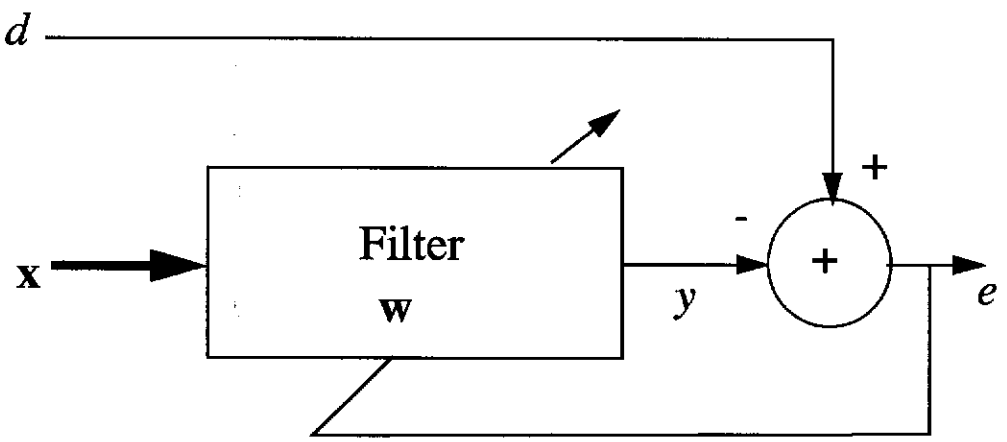


Figure 1.1: Adaptive Filter

The filter can be of different forms, but in this thesis, it is assumed to be a linear combiner. The input signal x, which can be drawn from either an input sequence or the output of an array of sensors, is applied to the filter.

The filter weights $\mathbf{w}$ are adjusted such that the filter output $y = \mathbf{w}^H\mathbf{x}$ gives a good approximation, in some sense, to the reference, or desired, signal d. The algorithm for adjusting, or adapting, $\mathbf{w}$ is usually based on an optimization criterion. It adapts $\mathbf{w}$ on the basis of the error signal e and the filter input signal $\mathbf{x}$.

Two commonly used optimization criteria are the minimum mean square error (MMSE) criterion and the exponentially weighted least squares (LS) criterion [22-25, 42, 54]. The MMSE criterion is defined by the following minimization problem:

$$\min_{\mathbf{w}}[\mathrm{E}(|d - \mathbf{w}^H\mathbf{x}|^2)] \qquad \text{P.1}$$

where E(.) denotes the expectation operation, and the superscript ^{H} stands for Hermitian transposition. The optimal solution of P.1, in a stationary environment, is given by the Wiener filter [23]:

$$\mathbf{w}_{MMSE} = \mathbf{R}^{-1}\mathbf{q} \qquad (1.1)$$

where the input signal covariance matrix is defined as $\mathbf{R} = \mathrm{E}(\mathbf{x}\mathbf{x}^H)$ and the cross-covariance vector as $\mathbf{q} = \mathrm{E}(\mathbf{x}d^*)$ with * denoting complex conjugate.

In contrast, the exponentially weighted least squares (LS) filter solves the following minimization problem [23]

$$\min_{\mathbf{w}(n)} \left[\sum_{i=0}^{n} \beta^{n-i} \left| d(i) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(i) \right|^2 \right], \qquad \text{P.2}$$

where $0 < \beta < 1$ is the exponential weighting factor. By differentiating the cost function with respect to $\mathbf{w}(n)$, it is easy to show that the optimum solution for P.2 is given by

$$\mathbf{w}_{LS}(n) = \Phi^{-1}(n)\theta(n) \qquad (1.2)$$

where

$$\Phi(n) = \sum_{i=0}^{n} \beta^{n-i}\mathbf{x}(i)\mathbf{x}^{\mathrm{H}}(i) \qquad (1.3)$$

$$\theta(n) = \sum_{i=0}^{n} \beta^{n-i}\mathbf{x}(i)d^{*}(i) \qquad (1.4)$$

Assuming the signals are ergodic, P.2 is in fact equivalent to P.1 in the sense that for n large, the solution given in (1.2) converges to (1.1). The matrix $\Phi(n)$ is termed the covariance matrix and the vector $\theta(n)$ is termed the cross-covariance vector. In this thesis, we focus only on the LS criterion.

The proper operation of an adaptive filter relies heavily on the algorithm controlling the update of its weights. For a low computational complexity, recursive algorithms where the previous values of the weights are used in

conjunction with the new input to calculate the new weights are preferred.

A whole family of least mean square (LMS) algorithms has been developed to solve the minimization problem P.1. These algorithms do not solve P.1 exactly but have been shown to converge in the mean to the Wiener filter solution (1.1) [23].

Least squares algorithms, which solve the least squares minimization problem P.2 exactly, offer an alternative to the LMS algorithms. These algorithms provide faster convergence rates, and unlike the LMS algorithms, the LS algorithms are insensitive to the eigenvalue spread of the underlying covariance matrix [24]. Furthermore, the LS algorithms are shown to converge to the Wiener filter in both mean and mean square [23]. There are two types of LS algorithms: those that update the inverse data covariance matrix (explicitly or implicitly), such as the conventional recursive least squares (CRLS) algorithm and the fast transversal filter (FTF) algorithm, and those that update the square root of the covariance matrix (or its inverse), such as the QR decomposition based recursive least squares (QRD-RLS) algorithm and the inverse QRD-RLS algorithm.

There are several considerations in the choice of a particular adaptive algorithm [23]. These considerations include the rate by which the algorithm converges; the deviation of the final (steady state) mean square error from that given by the Wiener filter; the capability to track statistical variation in the environment; the computational cost; the filter structure

(transversal, systolic array or lattice); and the numerical properties of the algorithm. This thesis is concerned mainly with the last issue.

1.1.2 Linear Constraints

In some applications such as beamforming, the filter should be prevented from responding undesirably to signals with specific characteristics [55]. There are several techniques to achieve this goal. Some of these techniques may require access to a reference signal, and some may need *a priori* knowledge of the statistics of the signal. One general approach that allows extensive control over the response of the adaptive filter is through the application of linear constraints on the filter weights [55].

By imposing linear constraints on an adaptive LS filter, the filter weights are given by the solution to the following linearly constrained optimization problem

$$\min_{\mathbf{w}(n)} \left[\sum_{i=0}^{n} \beta^{n-i} \left| d(i) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(i) \right|^2 \right] \text{ subject to } \mathbf{C}^{\mathrm{H}}\mathbf{w}(n) = \mathbf{f} \qquad \text{P.3}$$

where the constraint matrix C and constraint vector **f** together specify the constrained response of the adaptive filter.

Conventionally, P.3 is solved by the method of Lagrange multipliers resulting in

$$\mathbf{w}(n) = \Phi^{-1}(n)\theta(n) - \Phi^{-1}(n)\mathbf{C}[\mathbf{C}^{\mathbf{H}}\Phi^{-1}(n)\mathbf{C}]^{-1} \cdot [\mathbf{f} - \mathbf{C}^{\mathbf{H}}\Phi^{-1}(n)\theta(n)] \quad (1.5)$$

There have been two approaches presented in the literature to solve the linearly constrained optimization problem P.3 recursively. One is the direct method which is based on the method of Lagrange multipliers and updates the optimal solution directly [17, 43, 44, 64-66]. The other is called generalised sidelobe canceller (GSC) by which the linear constrained optimization problem is refomulated as a unconstrained problem where any adaptive filtering algorithm can then be applied [20, 21, 40, 41, 50]. In this thesis, we shall study three algorithms. Two are derived by the direct method, namely, the linearly constrained fast least squares (LC-FLS) algorithm of Resende et. al. [43, 44] and the linear systolic array for MVDR (Minimum Variance Distortionless Response) beamforming of Yang and Böhme [65]. The third algorithm is the linearly constrained QRD-RLS (LC-QRD-RLS) algorithm of McWhirter and Shepherd [40, 41, 50] which is an example of the GSC.

1.1.3 Applications of Linearly Constrained Adaptive Filters

Several applications are presented below to demonstrate the usefulness of the linearly constrained adaptive filter.

Example 1: Adaptive beamforming [3, 8, 25, 42, 55]

Linearly constrained adaptive filters arise naturally in beamforming

where an array of sensors are steered to a certain direction. Consider a uniformly distributed linear array of N elements with spacing d between adjacent elements.

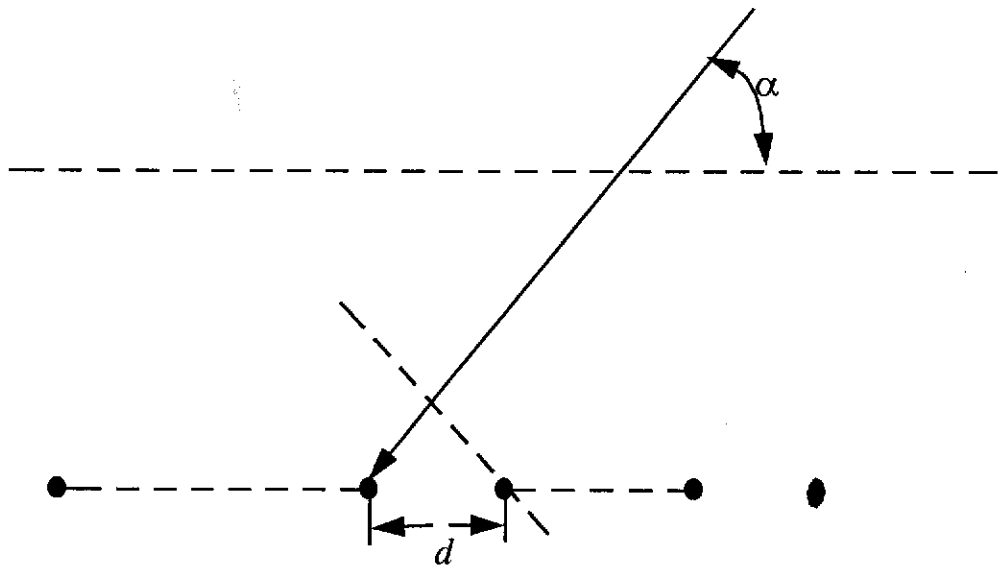


Figure 1.2: Uniformly Distributed Linear Array

Assume the signal is from a narrowband point source in the farfield. An adaptive filter is designed to have unit response to the signal from this source while suppressing those from elsewhere. This can be accomplished by imposing the linear constraint

$$\begin{bmatrix} 1 & e^{j\frac{2\pi d}{\lambda}\sin\alpha} & \cdots & e^{j\frac{2\pi d}{\lambda}(N-1)\sin\alpha} \end{bmatrix} \mathbf{w} = 1, \tag{1.6}$$

where λ is the wavelength of the desired signal. With linear constraints, one not only can define the response of the filter to a certain direction in

space, but also specify the shape of the mainbeam [25]. Multiple beams can also be easily formed by imposing multiple constraints on the filter. Theoretically, at most (N - 1) beams can be formed with N array elements.

Example 2: MVDR spectrum estimation [1, 22, 54]

In this application, the power of an input signal at a certain frequency ω_0 is to be estimated. To do this, we pass the signal through a filter whose output power is minimized without suppressing the signal from the prescribed frequency. An adaptive filter with the following constraint can be used in this situation

$$\left[1 \; e^{j\omega_0} \; \ldots \; e^{j\omega_0(N-1)}\right]\mathbf{w} = 1 \tag{1.7}$$

The spectrum of the input signal could be obtained by plotting the filter output power against ω_0.

1.2 Numerical Issues

When an adaptive filter is implemented digitally, its performance will most certainly deviate from that expected theoretically due to rounding errors. Thus it is essential to understand the effects of quantization errors [14] so that (i) unacceptable performance, especially overflow can be avoided; (ii) a proper choice of wordlength can be made; and (iii) a better algorithm structure can be selected.

1.2.1 Sources of Quantization Errors

There are two sources of quantization errors [14, 23, 24]:

- Analog-to-digital conversion (ADC);
- Finite wordlength arithmetic.

ADC quantization errors arise when the analog input is converted to its numerical representation. Assume B bits are used in the ADC. The quantization errors can be modelled as a zero mean noise with a variance [29]

$$\sigma^2 = \frac{2^{-2B}}{12}. \tag{1.8}$$

In a digital processor, arithmetic quantities are stored with a finite wordlength. All computational results are rounded or truncated to suit the given wordlength. In fixed point arithmetic, additions and substractions do not introduce errors assuming no overflow occurs, while multiplications and divisions introduce errors after the product or quotient is rounded or truncated. If floating point arithmetic is used, then all arithmetic operations will introduce errors [15]. Moreover, these finite wordlength arithmetic errors may have a nonzero mean.

1.2.2 Accumulation of Quantization Errors

Adaptive filters are by nature recursive. Therefore, one can expect that in a finite precision implementation the quantization errors can accumulate

and grow. There are two important issues to be considered as the result of the accumulation of quantization errors:

- numerical stability;
- numerical accuracy.

A numerically stable algorithm will limit the maximum deviation from its infinite precision performance such that all the internal arithmetic quantities maintain their theoretical properties [14, 23, 24, 59]. Numerical stability is especially important when the algorithm is to be used in a continuous manner, and it is an inherent property of the algorithm, that is, independent of the number of bits in an implementation [14, 23, 24].

In contrast, numerical accuracy is concerned with the size of the deviation from the infinite precision performance. Accuracy is a strong function of the numerical resolution for a numerically stable algorithm [14]. One may always improve accuracy by increasing the number of bits in the internal computation (if allowed). We will not consider this issue further in this thesis.

1.2.3 Numerical Issues in Linearly Constrained Adaptive Least Squares Filtering Algorithms

Several numerical divergence phenomena have been observed with the LS algorithms [14, 23, 24]:

- stalling phenomenon;

- exponentially increased effect of numerical error;
- explosive divergence.

When linear constraints are imposed on an adaptive filter, potentially there exists another instability phenomenon [17]:

- constraint drift.

Stalling occurs when the computed inverse covariance matrix is so small that with the given wordlength, multiplying by it is equivalent to multiplying by a null matrix. When this happens, the filter loses its capability to adapt to the environment. This can be solved by either using a smaller forgetting factor or increasing the number of bits.

The exponentially increased effect of numerical error is unique to the fast RLS algorithms. This problem has been studied in several works [14, 9, 37, 38, 51]. The computational redundancy in the $7N$ FTF algorithm was exploited to derive a complete stabilized $8N$ algorithm. The stabilization is complete in the sense that the resulting $8N$ algorithm is exponentially stable [51].

The most famous and commonly experienced numerical problem is explosive divergence [10, 23, 37]. This phenomenon has evoked an active research [9-11, 13, 14, 16, 19, 27, 34-38, 46, 47, 49, 51, 52, 58, 59, 67]. Figure 1.3 demonstrates the effect of explosive divergence in the conventional RLS (CRLS) algorithm. It is observed that the instantaneous output power increases dramatically at some iterations. This happens

when a certain algorithmic variable termed the *angle parameter* (also called the conversion factor or the likelihood parameter) falls out of its theoretical range, signalling the loss of positive definiteness of the computed inverse covariance matrix. This issue will be discussed in more details in Chapter 3.

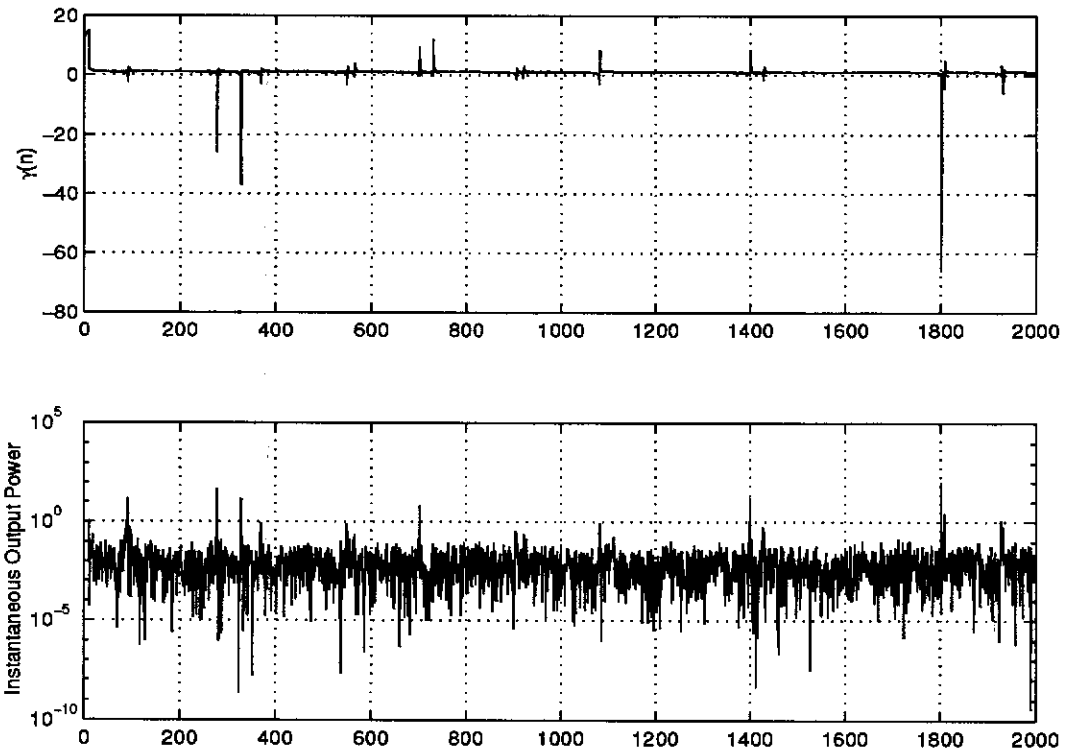


Figure 1.3: Explosive Divergence

The inverse of the angle parameter falls below unity. This is associated with a significant increase in the instant output power.

When linear constraints are applied to an adaptive filter, in addition to the numerical problems in the unconstrained algorithms, the computed filter

weights may deviate from the constraints due to quantization errors. In fact, this deviation could possibly grow unacceptably large in size. An example is the gradient projection algorithm [17]. If the adaptive filter weights fail to satisfy the imposing constraint, the desired signal could be suppressed as interference.

In this thesis, the explosive divergence and the constraint drift characteristics of the LC-FLS algorithm, the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithm will be examined.

1.3 Contribution of This Thesis

In this thesis, we

- analysed the constraint drift characteristics of the LC-FLS algorithm, the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithm;
- examined the error propagation properties of the LC-FLS algorithm and shows that the error propagation of this algorithm follows a random walk model;
- proved the stability of the LC-QRD-RLS algorithm.

This thesis is supported by the following published paper:
Huo, J. and Leung, Y.H., "Numerical Properties of the Linearly Constrained QRD-RLS Adaptive Filter," *Proc. IEEE ICASSP'98*, May 12-15, 1998, Seattle, WA, pp. 1685-1688.

1.4 Thesis Organisation

This thesis is organized as below. In Chapter 2, brief derivations of the three algorithms under study are presented. Their explosive divergence characteristics are then discussed in Chapter 3. The techniques commonly used to study the numerical stability properties of adaptive algorithms are also discussed in Chapter 3. The constraint drift issue is studied in Chapter 4. Chapter 5 concludes the thesis and proposes some possible directions for future research.

2 Linearly Constrained Least Squares Adaptive Filtering Algorithms

In this chapter, we present the LC-FLS algorithm of Resende et. al. [43, 44], the linear Systolic array for MVDR beamforming of Yang and Böhme [64-66], and the LC-QRD-RLS algorithm of McWhirter and Shepherd [40, 41, 50]. Some general comments about these algorithms are also made.

2.1 The Linearly Constrained LS Adaptive Filtering Problem

For a linearly constrained LS adaptive filter, the filter weights are chosen to minimize the exponentially weighted sum of the filter output power subject to a set of linear constraints

$$\min_{\mathbf{w}(n)}\left[\sum_{i=0}^{n}\beta^{n-i}\left|d(i)-\mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(i)\right|^{2}\right] \text{ subject to } \mathbf{C}^{\mathrm{H}}\mathbf{w}(n) = \mathbf{f} \qquad \text{P.4}$$

where $0<\beta<1$ is the exponentially weighting factor, the $N\times 1$ vector $\mathbf{w}(n)$ denotes the filter weights at time n, $d(i)$ is the reference signal, $\mathbf{x}(i)$ is the $N\times 1$ input data vector, $\mathbf{C}$ is the constraint matrix with dimension $N\times K$, and $\mathbf{f}$ is a $K\times 1$ constraint vector.

As mentioned in Chapter 1, the optimal solution of P.4 is given by

$$\mathbf{w}(n)= \Phi^{-1}(n)\theta(n) - \Phi^{-1}(n)\mathbf{C}[\mathbf{C}^{\mathbf{H}}\Phi^{-1}(n)\mathbf{C}]^{-1} \cdot [\mathbf{f} - \mathbf{C}^{\mathbf{H}}\Phi^{-1}(n)\theta(n)] \tag{2.1}$$

where

$$\Phi(n) = \sum_{i=0}^{n} \beta^{n-i}\mathbf{x}(i)\mathbf{x}^{\mathrm{H}}(i) \tag{2.2}$$

$$\theta(n) = \sum_{i=0}^{n} \beta^{n-i}\mathbf{x}(i)d^{*}(i) \tag{2.3}$$

2.2 The Linearly Constrained FLS Algorithm [43, 44]

With the close form expression of the optimal solution (2.1), the filter weights can be calculated at every snapshot with new input signal. One may update $\Phi^{-1}(n)$ and then calculate the filter weights as (2.1). However this would be computationaly very expensive, especially with the matrix inversion involved in the computation of the matrix $[\mathbf{C}^{\mathbf{H}}\Phi^{-1}(n)\mathbf{C}]^{-1}$. In [43, 44], an efficient recursion for updating the matrix $\Phi^{-1}(n)\mathbf{C}[\mathbf{C}^{\mathbf{H}}\Phi^{-1}(n)\mathbf{C}]^{-1}$ and the filter weights $\mathbf{w}(n)$ has been derived.

2.2.1 Derivation

Define matrices

$$\mathbf{P}(n) = \Phi^{-1}(n) \tag{2.4}$$

$$\mathbf{T}(n) = \mathbf{P}(n)\mathbf{C}. \tag{2.5}$$

Assume the Kalman gain is available at iteration n. The matrix P(n) can be updated as

$$\mathbf{P}(n) = \beta^{-1}[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{H}(n)]\mathbf{P}(n-1) \tag{2.6}$$

where $\mathbf{g}(n)$ is the Kalman gain and $\mathbf{I}$ is an identity matrix with compatible dimension. As we shall see in the next chapter, this recursion is unstable. Post-multiply both sides of (2.6) by the constraint matrix, it follows that

$$\mathbf{T}(n) = \beta^{-1}[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{H}(n)]\mathbf{T}(n-1). \tag{2.7}$$

Now pre-multiply (2.7) by $\mathbf{C}^{H}$, and define

$$\Psi(n) = \mathbf{C}^{H}\mathbf{T}(n), \tag{2.8}$$

one then obtains

$$\Psi(n) = \beta^{-1}[\Psi(n-1) - \mathbf{C}^{H}\mathbf{g}(n)\mathbf{x}^{H}(n)\mathbf{T}(n-1)] \tag{2.9}$$

Applying the famous matrix inversion lemma[1], let

---

1. Suppose A is a positive definite matrix, u and v are vectors. The inverse of $(\mathbf{A} + \mathbf{u}\mathbf{v}^{H})^{-1}$ is given by [32]

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^{H})^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^{H}\mathbf{A}^{-1}}{1 + \mathbf{v}^{H}\mathbf{A}^{-1}\mathbf{u}} \tag{1.5}$$

$$\mathbf{A} = \Psi(n-1) \tag{2.10}$$

$$\mathbf{u} = -\mathbf{C}^{\mathrm{H}}\mathbf{g}(n) \tag{2.11}$$

and

$$\mathbf{v}^{\mathrm{H}} = \mathbf{x}^{\mathrm{H}}(n)\mathbf{T}(n-1). \tag{2.12}$$

it follows that

$$\Psi^{-1}(n) = \beta\Psi^{-1}(n-1) \cdot \left[\mathbf{I} + \frac{\mathbf{C}^{\mathrm{H}}\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)\mathbf{T}(n-1)}{1 - \mathbf{x}^{\mathrm{H}}(n)\mathbf{T}(n-1)\Psi^{-1}(n-1)\mathbf{C}^{\mathrm{H}}\mathbf{g}(n)}\right]. \tag{2.13}$$

Combining the recursions (2.7) and (2.13) together, and defining

$$\mathbf{Q}(n) = \mathbf{T}(n)\Psi^{-1}(n), \tag{2.14}$$

a more compact update equation is now reached

$$\mathbf{Q}(n) = [\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{Q}(n-1) \cdot \left[\mathbf{I} + \frac{\mathbf{C}^{\mathrm{H}}\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)\mathbf{T}(n-1)}{1 - \mathbf{x}^{\mathrm{H}}(n)\mathbf{Q}(n-1)\mathbf{C}^{\mathrm{H}}\mathbf{g}(n)}\right]. \tag{2.15}$$

Define

$$\mathbf{w}_{unc}(n) = \Phi^{-1}(n)\theta(n), \tag{2.16}$$

The update equation for $\mathbf{w}_{unc}(n)$ is given by

$$\mathbf{w}_{unc}(n) = \mathbf{w}_{unc}(n-1) + \mathbf{g}(n)e_{unc}(n-1) \tag{2.17}$$

where

$$e_{unc}(n-1) = d(n) - \mathbf{w}_{unc}^{\mathrm{H}}(n)\mathbf{x}(n). \tag{2.18}$$

The solution (2.1) becomes

$$\mathbf{w}(n) = \mathbf{w}_{unc}(n) - \mathbf{Q}(n)[\mathbf{f} - \mathbf{C}^{\mathbf{H}}\mathbf{w}_{unc}(n)]. \tag{2.19}$$

Applying (2.15) and (2.18) to (2.19), we obtain

$$\begin{aligned}\mathbf{w}(n) = \mathbf{w}(n-1) &+ \mathbf{g}(n)e(n-1) \\ &+ \mathbf{Q}(n)\{\mathbf{f} - \mathbf{C}^{\mathbf{H}}[\mathbf{w}(n-1) + \mathbf{g}(n)e(n-1)]\}\end{aligned} \tag{2.20}$$

where

$$e(n-1) = d(n) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(n) \tag{2.21}$$

However, the presence of rounding errors can cause the algorithm to become unstable numerically. According to the definition of the matrix $\mathbf{Q}(n)$, it should satisfy the constraint

$$\mathbf{C}^{\mathrm{H}}\mathbf{Q}(n) = \mathbf{I}. \tag{2.22}$$

However, rounding errors will surely violate this constraint. Without careful control, the filter weights will become unconstrained due to the

violation of the constraint (2.22). A projection is introduced to compensate for the effect of rounding errors

$$\mathbf{Q}(n) = \hat{\mathbf{Q}}(n) + \mathbf{C}[\mathbf{C}^{\mathrm{H}}\mathbf{C}]^{-1}[\mathbf{I} - \mathbf{C}^{\mathrm{H}}\hat{\mathbf{Q}}(n)] \tag{2.23}$$

1 Update the Kalman gain

2 Update matrix $\mathbf{Q}(n)$

2.1 $\mathbf{l} = \mathbf{C}^{\mathrm{H}}\mathbf{g}(n)$

2.2 $\mathbf{m}^{\mathrm{H}} = \mathbf{x}^{\mathrm{H}}(n)\mathbf{Q}(n-1)$

2.3 $$\hat{\mathbf{Q}}(n) = [\mathbf{Q}(n-1) - \mathbf{g}(n)\mathbf{m}^{\mathrm{H}}]\left[\mathbf{I} + \frac{\mathbf{l}\mathbf{m}^{\mathrm{H}}}{1 - \mathbf{m}^{\mathrm{H}}\mathbf{l}}\right]$$

2.4 $\mathbf{Q}(n) = \hat{\mathbf{Q}}(n) + \mathbf{C}[\mathbf{C}^{\mathrm{H}}\mathbf{C}]^{-1}[\mathbf{I} - \mathbf{C}^{\mathrm{H}}\hat{\mathbf{Q}}(n)]$

3 Calculate filter weights $\mathbf{w}(n)$.

3.1 $e(n-1) = d(n) - \mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(n)$

3.2 $$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)e(n-1) + \mathbf{Q}(n)\{\mathbf{f} - \mathbf{C}^{\mathrm{H}}[\mathbf{w}(n-1) + \mathbf{g}(n)e(n-1)]\}$$

Table 2.1 Summary of the LC-FLS Algorithm

In [43, 44] it was claimed that by applying this projection, the recursion for updating $\mathbf{Q}(n)$ is numerically robust. However, as we shall see in the next chapter, this is not the whole story.

The algorithm is summarised in Table 2.1.

2.2.2 General Comments

1 The LC-FLS algorithm is of reasonable computational complexity. The number of multiplications, excluding the error correction (2.23), is proportional to $NK^2 + 5NK + K^2 + K + 2N$ [44]. Moreover, K divisions are needed. The projection operation (2.23), with the matrix $\mathbf{C}[\mathbf{C}^H\mathbf{C}]^{-1}$ pre-computed, requires $2K^2N$ multiplications. Notice that it is not necessary to implement (2.23) at every iteration. It can be done at a regular time interval, depending on the precision and the parameters of the algorithm.

2 The algorithm can be separated into two parts: an unconstrained part where an unconstrained LS algorithm is used to update the Kalman gain, and a constrained part where the matrix $\mathbf{Q}(n)$ is updated. The unconstrained part updates independently of the constrained part, i.e., the update of the matrix $\mathbf{Q}(n)$ has no effect on the calculation of the Kalman gain.

3 Simulation results show that the LC-FLS algorithm has a much faster convergence and a lower residual error than the Frost algorithm [43, 44].

2.3 Linear Systolic Array for MVDR Beamforming

In [64-66], Yang and Böhme presented an algorithm for linearly

constrained LS filtering in the context of beamforming and direction finding. This algorithm is also based on the method of Lagarange multipliers. It computes the filter output without explicitly updating the filter weights. In this section, we present this algorithm for MVDR beamforming [66].

2.3.1 Derivation

For a single constraint, the term $[\mathbf{C}^{\mathrm{H}}\Phi^{-1}(n)\mathbf{C}]$ becomes a scalar. Let it be denoted by $D(n)$, and the vector $\Phi^{-1/2}(n)\mathbf{C}$ be denoted as $\mathbf{t}(n)$. The filter output $y(n)$ is given by (assuming the reference signal $d(n)$ is not presented)

$$y(n) = \frac{\mathbf{C}^{\mathrm{H}}(n)\Phi^{-\mathrm{H}}(n)\mathbf{x}(n)}{D(n)}. \tag{2.24}$$

Consider the update of the square-root of the data covariance matrix

$$\mathbf{Q}(n)\begin{bmatrix}\mu\Phi^{1/2}(n-1)\\ \mathbf{x}^{\mathrm{H}}(n)\end{bmatrix} = \begin{bmatrix}\Phi^{1/2}(n)\\ \mathbf{0}^{\mathrm{H}}\end{bmatrix} \tag{2.25}$$

where $\mu = \sqrt{\beta}$, $\mathbf{Q}(n)$ is a unitary matrix and $\left[\Phi^{1/2}(n)\right]$ is an upper triangular matrix. In [49], Schreiber showed that the vector $\mathbf{t}(n)$ and $D(n)$ can be updated as follows

$$\mathbf{Q}(n)\begin{bmatrix}\dfrac{\mathbf{t}(n-1)}{\mu}\\ 0\end{bmatrix} = \begin{bmatrix}\mathbf{t}(n)\\ z(n)\end{bmatrix}, \tag{2.26}$$

Initialize: $\Phi^{-1/2}(0)$, $\mathbf{t}(0)$

At iteration n

1 Unitary transformation

$$\mathbf{Q}(n)\begin{bmatrix}\mu\Phi^{1/2}(n-1) & \mathbf{0} & \dfrac{\mathbf{t}(n-1)}{\mu}\\ \mathbf{x}^{\mathrm{H}}(n) & 1 & 0\end{bmatrix} = \begin{bmatrix}\Phi^{1/2}(n) & \mathbf{h} & \mathbf{t}(n)\\ \mathbf{0}^{\mathrm{H}} & \alpha(n) & z(n)\end{bmatrix}$$

2 Beamforming output computation

$$D(n) = D(n-1)/\mu - |z(n)|^2$$

$$y(n) = \frac{\alpha(n)z^*(n)}{D(n)}$$

Table 2.2 Linear Systolic Array for MVDR Beamforming

$$D(n) = \frac{D(n-1)}{\mu} - |z(n)|^2, \tag{2.27}$$

where, from the direct residual extraction of [50], it can be seen that

$$z(n) = \beta^{-1}\mathbf{x}^{\mathrm{H}}(n)\Phi^{-1}(n-1)\mathbf{C}. \tag{2.28}$$

$z(n)$ differs from $\mathbf{x}^{\mathrm{H}}(n)\Phi^{-1}(n)\mathbf{C}$ by the scaling factor [23, 50]

$$\alpha(n) = \frac{1}{1+\beta^{-1}\mathbf{x}^{H}(n)\Phi^{-1}(n-1)\mathbf{x}(n)} = \mathbf{Q}(n)\begin{bmatrix}\mathbf{0}\\1\end{bmatrix}. \qquad (2.29)$$

Therefore, the filter output $y(n)$ can be computed as

$$y(n) = \frac{\alpha(n)z^{*}(n)}{D(n)}. \qquad (2.30)$$

This algorithm is summarised in Table 2.2.

2.3.2 General Comments

1 It is interesting to observe that this algorithm only involves a unitary transformation, which is very suitable for VLSI implementation [64-66]. It can be extended to deal with multiple constraints where an additional Hyperbolic transformation and a Gaussian elimination is needed [64, 65].

2 The algorithm has a structure very similar to the LC-FLS algorithm. It can be separated into an unconstrained part and a constrained part where the constrained part is uniquely defined by the unconstrained part.

2.4 The Linearly Constrained QRD-RLS Algorithm [40,41, 50]

Another algorithm to solve P.4 is the linearly constrained QRD-RLS algorithm [40, 41, 50]. This algorithm offers features such as parallelism and regularity which are highly desirable for VLSI implementation.

2.4.1 Derivation

Premultiplying both side of the constraint by a unitary matrix $\mathbf{Q}_0$ such that

$$\mathbf{Q}_0\mathbf{C}^{\mathrm{H}} = \begin{bmatrix}\mathbf{U} & \mathbf{V}\end{bmatrix} \tag{2.31}$$

where U is a upper triangular matrix, the constraint becomes

$$\begin{bmatrix}\mathbf{U} & \mathbf{V}\end{bmatrix}\begin{bmatrix}\mathbf{w}_a(n) \\ \mathbf{w}_b(n)\end{bmatrix} = \mathbf{m} \tag{2.32}$$

where

$$\mathbf{w}^{\mathrm{H}}(n) = \begin{bmatrix}\mathbf{w}_a^{\mathrm{H}}(n) & \mathbf{w}_b^{\mathrm{H}}(n)\end{bmatrix} \tag{2.33}$$

and

$$\mathbf{m} = \mathbf{Q}_0\mathbf{f}. \tag{2.34}$$

Solving for $\mathbf{w}_a(n)$ in term of $\mathbf{w}_b(n)$, it follows that

$$\mathbf{w}_a(n) = \mathbf{U}^{-1}\mathbf{m} - \mathbf{U}^{-1}\mathbf{V}\mathbf{w}_b(n). \tag{2.35}$$

Substituting (2.35) into (2.33), on obtains

$$\mathbf{w}(n) = \begin{bmatrix} \mathbf{U}^{-1}\mathbf{m} - \mathbf{U}^{-1}\mathbf{V}\mathbf{w}_b(n) \\ \mathbf{w}_b(n) \end{bmatrix} = \begin{bmatrix} \mathbf{U}^{-1}\mathbf{m} \\ 0 \end{bmatrix} + \begin{bmatrix} -\mathbf{U}^{-1}\mathbf{V} \\ \mathbf{I} \end{bmatrix} \mathbf{w}_b(n). \quad (2.36)$$

Let

$$\mathbf{B} = \begin{bmatrix} -\mathbf{U}^{-1}\mathbf{V} \\ \mathbf{I} \end{bmatrix} \quad (2.37)$$

$$\mathbf{w}_c = \begin{bmatrix} \mathbf{U}^{-1}\mathbf{m} \\ 0 \end{bmatrix}. \quad (2.38)$$

Applying (2.36) to P.4 to remove the constraint on the filter weights, the linearly constrained LS adaptive filtering problem is changed into an unconstrained problem as follows

$$\min_{\mathbf{w}_b(n)} \sum_{i=1}^{n} \beta^{n-i} [\underline{d}(i) - \mathbf{w}_b^H(n)\underline{\mathbf{x}}(i)]^2 \quad (2.39)$$

where $\underline{d}(i)$ and $\underline{\mathbf{x}}(i)$ is defined as

$$\underline{d}(i) = \mathbf{w}_c^H \mathbf{x}(i) \quad (2.40)$$

$$\underline{\mathbf{x}}(i) = \mathbf{B}^H \mathbf{x}(i). \quad (2.41)$$

The remaining unconstrained minimization is done with the standard QRD-RLS algorithm [24].

The algorithm is summarised in Table 2.3.

1	Pre-calculate matrix **B** and vector $\mathbf{w}_c$;
2	Signal pre-processing $\underline{d}(n) = d(n) - \mathbf{w}_c^H \mathbf{x}(n)$ $\underline{\mathbf{x}}(n) = \mathbf{B}^H \mathbf{x}(n)$
3	LS filtering with $\underline{d}(n)$ as reference signal and $\underline{\mathbf{x}}(n)$ as filter input using the QRD-RLS algorithm.

Table 2.3 Summary of LC-QRD-RLS Algorithm

2.4.2 Systolic Array Implementation

In [40, 41, 50], the authors showed how the LC-QRD-RLS algorithm can be implemented very efficiently with a single systolic array. Figure 2.1 depicts the systolic array implementation of this algorithm. Detailed description can be found in [50]. The system includes two sections: A frozen network which processes the input signal to generate $\underline{d}(i)$ and $\underline{\mathbf{x}}(i)$, and a canonical systolic array for the least squares minimization.

At the beginning of the algorithm, all the cells in the systolic array are initialised to zero. Before any input signal arrives, the constraint matrix is triangularised and stored in the first K rows of the array. This is done by feeding the constraint matrix and vector into the systolic array as input

data and reference signal respectively with an exponentially weighting factor of 1. After that, the rows that store the triangularised constraint matrix are frozen. When the input signal arrives, the exponentially weighting factor β is set to its appropriate value, and the residual output is observed at the output of the final cell of the array. If the weights are of interest, one can freeze the whole systolic array, and feed it with an identity matrix as input and a null vector as the reference signal and observe the output of the array. This is called *serial weight flushing*.

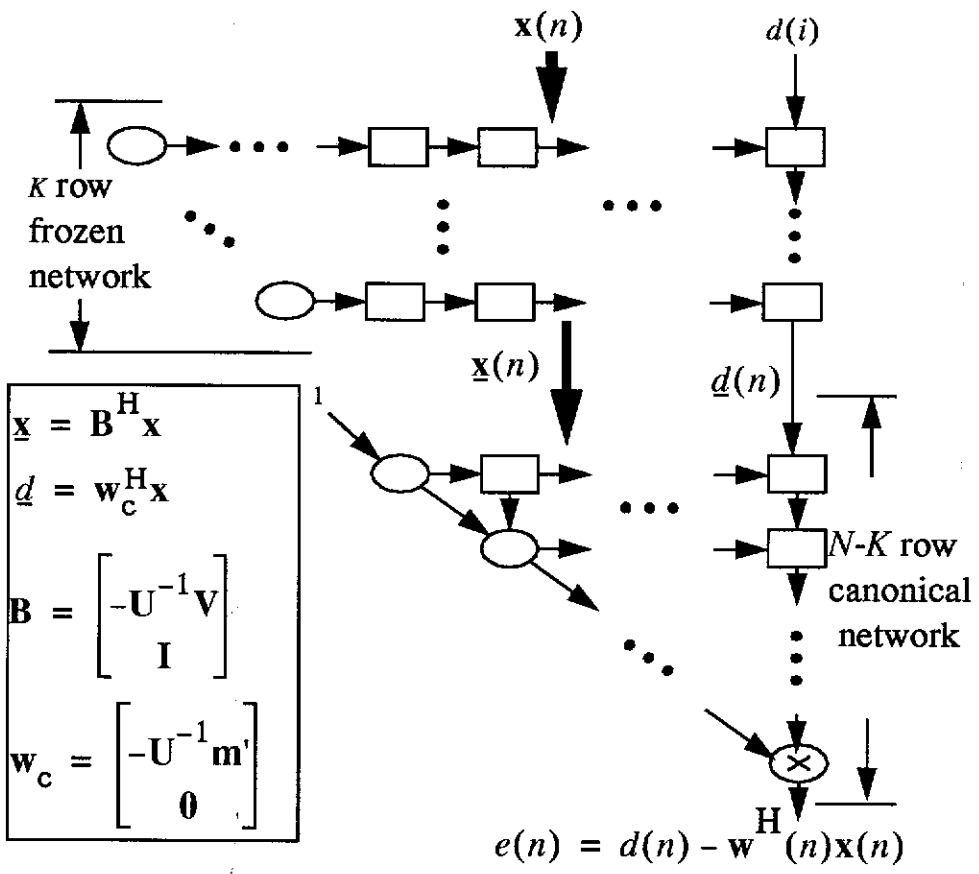


Figure 2.1: Systolic Array Implementation of LCQRD-RLS Algorithm

A close investigation of the operation of the frozen network shows that passing the signal through the frozen network is equivalent to applying Gaussian elimination to the matrix

$$\begin{bmatrix} \mathbf{U}\ \mathbf{V}\ \mathbf{m} \\ \mathbf{x}(i)\ d(i) \end{bmatrix}. \tag{2.42}$$

2.4.3 General Comments

1 The LC-QRD-RLS algorithm is indeed a special case of the generalised sidelobe canceller (GSC) [20] as depicted in Figure 2.2.

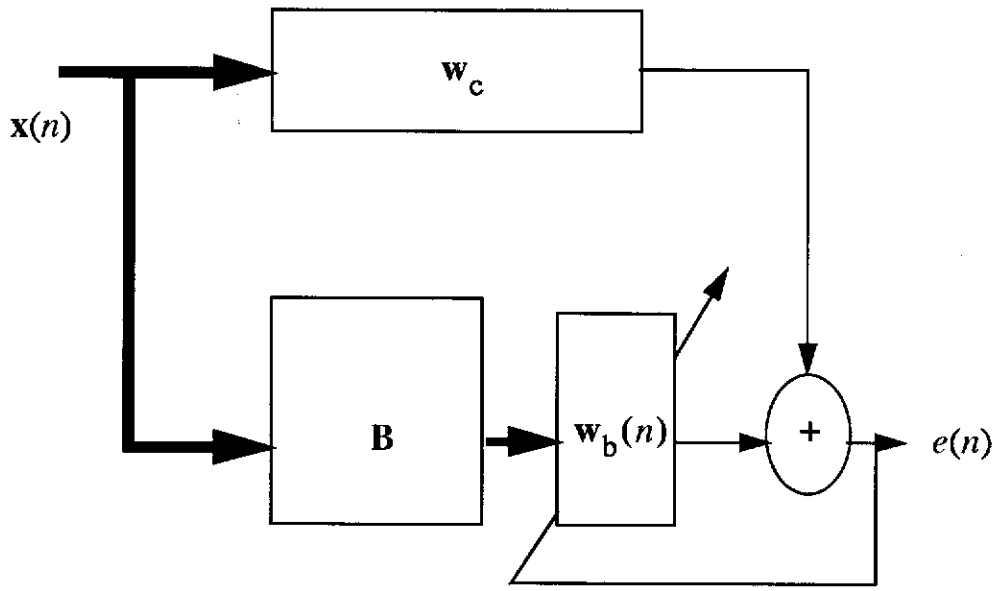


Figure 2.2: Generalised Sidelobe Canceller

In Figure 2.2, the matrix filter $\mathbf{B}$ and $\mathbf{w}_c$ are given by

$$\mathbf{C}^H\mathbf{B} = \mathbf{0}, \tag{2.43}$$

$$\mathbf{C}^H\mathbf{w}_c = \mathbf{f}. \tag{2.44}$$

Compare with the LC-QRD-RLS algorithm, it can be seen that the algorithm is a GSC with $\mathbf{B}$ and $\mathbf{w}_c$ as defined in (2.38) and (2.39). It is interesting to observe that Frost's LC-LMS algorithm is also a GSC with

$$\mathbf{B} = \mathbf{I} - \mathbf{C}[\mathbf{C}^H\mathbf{C}]^{-1}\mathbf{C}^H \tag{2.45}$$

$$\mathbf{w}_c = \mathbf{C}[\mathbf{C}^H\mathbf{C}]^{-1}\mathbf{f} \tag{2.46}$$

and the adaptive weights updated with a LMS algorithm.

As one may expect, there are infinitely many choices of $\mathbf{B}$ and $\mathbf{w}_c$. Different choice will result in different performance in terms of the performance improvement due to adaptation (PIA) [28]. The most important advantage of the GSC lies in that the adaptive part of the filter is unconstrained and this allows the application of much simpler algorithms.

2 The computational load to enforce the constraint is only $NK - \frac{3}{2}K - \frac{K^2}{2}$ multiplications plus K divisions (implemented with a systolic array). Moreover, the K divisions can be avoided by storing the inverse of the diagonal elements of the matrix $\mathbf{U}$ in the frozen network. When the algorithm is implemented with a systolic array, the

computations are distributed among cells and for each cell the computational complexity is of an order of unity.

3 The LC-QRD-RLS algorithm have the same features of fast convergence rate and low residual error as the LC-FLS algorithm.

2.5 Summary

In this chapter, the LC-FLS, the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithms are presented. The first two algorithms are derived from the classical method of Lagarange multipliers. The LC-FLS algorithm offers an alternative of the Frost algorithm that enjoys a much faster convergence rate and lower residual error with a price of a increase in computational complicity. The linear systolic array for MVDR beamforming computes the filter output directly without explicitly calculating the filter weights. The LC-QRD-RLS algorithm is a special case of the generalised sidelobe canceller. Both the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithm demonstrate desirable features such as parallelism and regularity and can be implemented with a single systolic array efficiently.

3 Explosive Divergence

As discussed in Chapter 2, the three linearly constrained algorithms can be separated into an unconstrained part and a constrained part. In this chapter, we would focus on the implementation of the unconstrained part of these algorithms which would be a normal LS filtering algorithm.

When implemented digitally, LS adaptive filtering algorithms are likely to experience numerical difficulties, of these explosive divergence is the most well-documented one. This unexpected wild behaviour has evoked an active field of research. It has been studied from different perspectives and some interesting results have been achieved.

In the first section of this chapter, a brief review of the most commonly used analysis methods, namely error propagation, error accumulation and numerical consistency, is given. Then explosive divergence of the CRLS algorithm and the QRD-RLS algorithm is examined from these perspectives.

3.1 Review of Studies on the Numerical Stability of Adaptive LS Algorithms

There exists a rich literature on the numerical stability of adaptive filtering algorithms. The various approaches to study the numerical stability of adaptive filtering algorithms fall generally into two categories: the first category studies the generation, propagation and accumulation of

rounding errors with the aim of deriving bounds on the computed quantities; while the second category exploits the concept of backward stability to examine the physical nature of the numerical instability phenomenon. In this section, these techniques are reviewed.

3.1.1 Error System

The first category of analysis methods is based on the following mathematical model. Consider the following representation of an adaptive filtering algorithm

$$S(n) = f(S(n-1), d(n), \mathbf{x}(n)) \tag{3.1}$$

where S denotes the algorithm state (i.e., the internal variables that are updated), and f is the update function of the algorithm which is generally nonlinear. The error system of this algorithm is defined as the difference between its finite precision and infinite precision implementation

$$e(n) \equiv \tilde{S}(n) - S(n) \tag{3.2}$$

where $\tilde{x}$ denotes a quantity computed with finite precision arithmetic.

Substituting (3.1) into (3.2), it can be shown that an the error update equation of the form shown below can be obtained [51]

$$e(n) = g(e(n-1), d(n), \mathbf{x}(n)) + \zeta(n) \tag{3.3}$$

where $g(.)$ denotes the propagation of the error at iteration n - 1 to

iteration n, and $\zeta(n)$ denotes the local error made at iteration n. Indeed, this can be viewed as a nonlinear dynamic system as shown in Figure 3.1, thus allowing the rich literature of control systems to be exploited [30, 1, 60].

Figure 3.1 and (3.3) suggest that a complete picture of the numerical behaviour of an adaptive filtering algorithm should include the study of [51]:

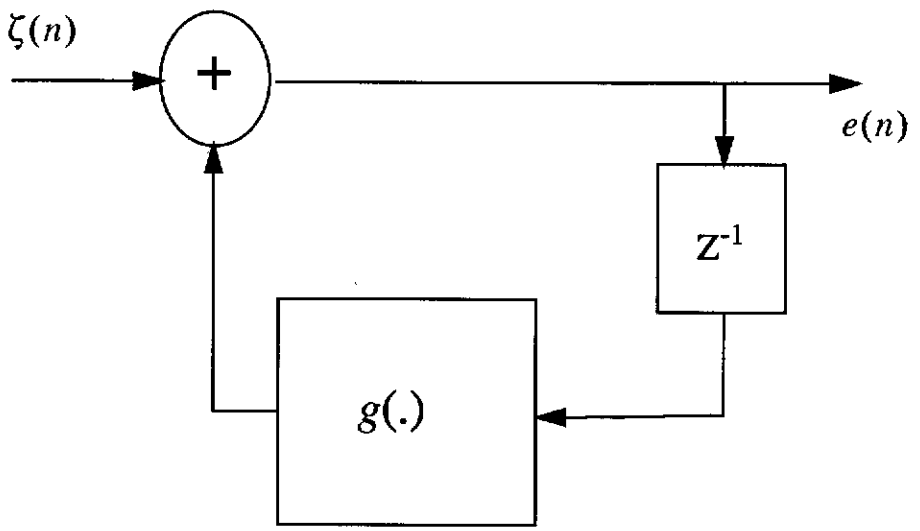


Figure 3.1: Modelling Error System as a Nonlinear and Dynamic System

- Error generation: How is $\zeta(n)$ generated?

  The description of error generation can follow either a deterministic approach [56, 57, 59] or a stochastic one [4, 5, 12, 48, 61]. The characteristics of $\zeta(n)$ depends heavily on the implementation (fixed-point, floating-point, wordlength and so on). This issue is not as crucial as the next two, and thus will not be discussed further in

this thesis.

- Error propagation: How do errors at iteration n propagate in subsequent recursions, assuming no further errors are introduced?
- Error accumulation: How do errors generated at different iteration interact and accumulate?

(1) Error Propagation

Error propagation is studied by analysing the behaviour of the following homogeneous system

$$e(n) = g(e(n-1), d(n), \mathbf{x}(n)) \tag{3.4}$$

with the initial condition

$$e(0) = e_0. \tag{3.5}$$

A quantitative description of the error propagation of an algorithm turns out to be extremely difficult due to the nonlinear and dynamic nature of the error system. For mathematical tractability, it is common practice to make the following assumptions:

(ii) The precision available for internal computation is good enough such that $e(n)$ is small and the linear approximation of a system (3.4) is accurate;

(iii) The input $\mathbf{x}(n)$ is an i.i.d. Gaussian process;

(iv) The algorithm has reached its steady state.

Assumption (ii) enables us to write the nonlinear system (3.4) as follows

$$e(n) = G(n)e(n-1); \tag{3.6}$$

where

$$G(n) = \frac{\partial}{\partial e(n)}[g(e(n-1), d(n), \mathbf{x}(n))]\Big|_{e(n)=0}. \tag{3.7}$$

Assumption (iii) says that the covariance matrix $\Phi(n)$ is independent of $\Phi(n-1)$; and assumption (iii) and (iv) together ensure that $\Phi(n)$ is of full rank for all n.

Moreover, it is shown in [2, 30, 31] that under certain conditions, the time varying system (3.6) can be further simplified to a time-invariant system

$$e(n) = Ge(n-1) \tag{3.8}$$

where

$$G = \mathrm{E}[G(n)]. \tag{3.9}$$

The system (3.8) is termed the *average system* of (3.6) and it is highly desirable that the error system (3.6) or (3.8) is exponentially stable because this then implies that the error accumulation system is BIBO stable. Note, however, that the above approach assume that the precision is good enough such that the accumulated errors remain in the region

where the linear approximation of the nonlinear error system is accurate [10, 38, 51, 58, 59, 67]. Note also that exponential stability is only a necessary but not sufficient condition for numerical stability.

(2) Error Accumulation

Several authors have studied the nonlinear accumulation of rounding errors [11, 13, 35, 36, 53] and established the error bounds of some of the algorithms. The most interesting results seem to be those presented in the recent papers [35, 36]. In these papers, Liavas and Regalia established a condition which guarantees the CRLS algorithm to be free of explosive divergence. However, the requirements of the derived condition are generally very conservative.

3.1.5 Numerical Consistency

Another way of examining numerical stability is through the study of numerical consistency [45-47, 52]. This approach is based on the notion of backward stability due to Wilkinson [62].

Let S_i denotes the collection of all possible states which the algorithm can reach using infinite precision arithmetic. Similarly, let S_f denotes the collection of all possible states which the algorithm can reach using finite precision arithmetic. The algorithm is said to be backward consistent if

$$S_f \subset S_i. \tag{3.10}$$

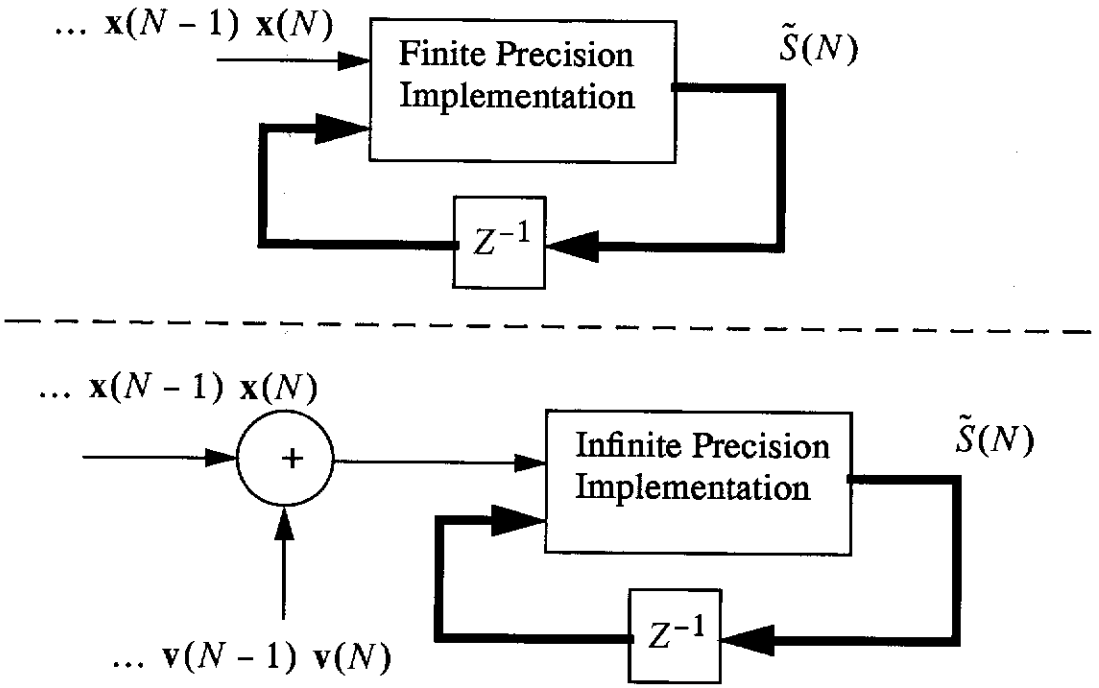


Figure 3.2: Conceptual Model to Establish Stability for an Backward Consistent Adaptive Algorithm

Now, since

$$\tilde{S}(n) \in S_f, \tag{3.11}$$

requirement (3.10) can be written in its equivalent form as

$$\tilde{S}(n) \in S_i. \tag{3.12}$$

Furthermore, S_i can be defined by the structural constraints imposed on the algorithm state by the physical nature of the filtering problem. For example, with persistent excitation, for a LMS algorithm, S_i is defined as

$$S_i := \{\mathbf{w}(n) : \mathbf{w}(n) \in C^{N\times 1}\}\ , \tag{3.13}$$

and for a CRLS algorithm,

$$S_i := \{[\mathbf{P}(n), \mathbf{w}(n)] : \mathbf{P}(n) = \mathbf{P}^{\mathrm{H}}(n) > 0, \mathbf{w}(n) \in C^{N\times 1}\}. \tag{3.14}$$

If the computed algorithm state $\tilde{S}(n)$ satisfies the structural constraints at all times, i.e., (3.12) holds, the algorithm is said to be backward consistent.

A backward consistent implementation of an algorithm is stable provided the algorithm converges in exact arithmetic. The effect of rounding errors in a backward consistent implementation can be viewed as an additional additive noise in the input signal (see Figure 3.2) [46, 47].

3.2 Normal Equation Based Algorithms

Normal equation based RLS algorithms update the inverse covariance matrix explicitly or implicitly. It is well-known that these algorithms can suffer from explosive divergence. In this section, we review various results that have been reported in the literature concerning explosive divergence, particularly with regarding to the CRLS algorithm.

3.2.1 CRLS Algorithm

The CRLS algorithm is summarised in Table 3.1. Notice there are 3

versions of this algorithm. They differ only in the way the inverse covariance matrix $\mathbf{P}(n)$ is updated.

Version I

$\mathbf{P}(0) = \partial\mathbf{I}$, $\mathbf{w}(0) = \mathbf{0}$;

At iteration n,

1. $\mathbf{h} = \mathbf{P}(n-1)\mathbf{x}(n)$
2. $\alpha = 1/[\beta + \mathbf{x}^{H}(n)\mathbf{h}]$
3. $\mathbf{g}(n) = \mathbf{h}\alpha$
4. $\mathbf{P}(n) = \beta^{-1}[\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{h}^{H}]$
5. $\xi = d(n) - \mathbf{w}^{H}(n-1)\mathbf{x}(n)$
6. $\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)e$

Version II

Identical as Version I except for

4. $\mathbf{P}(n) = \beta^{-1}\{\mathbf{P}(n-1) - \mathbf{g}(n)[\mathbf{x}^{H}(n)\mathbf{P}(n-1)]\}$

Version III

Identical as Version I except for

4. $\mathbf{P}(n) = \mathrm{Tri}\{\beta^{-1}[\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{h}^{H}]\}$

Table 3.1 CRLS Algorithm

where in Table 3.1, Tri(**X**) stands for the upper triangular part of the matrix **X**.

In Section 1.2.3, we plot the instantaneous output power of Version II of

the CRLS algorithm in a typical simulation scenario. It was noted that the instantaneous output power could increase suddenly after the algorithm had converged. This phenomenon, called explosive divergence, has evoked an active area of research [9-11, 13, 14, 16, 19, 27, 34-38, 46, 47, 49, 51, 52, 58, 59, 67]. The general conclusion is that explosive divergence can be attributed to the loss of positive definiteness in the computed covariance matrix as a result of the accumulation of rounding errors. This loss of positive definiteness of the covariance matrix is signified by the negative value of the arithmetic variable α in Table 3.1.

3.2.2 Error Propagation

Error propagation of the CRLS algorithm includes two aspects [38, 59, 67]:

- Error propagation through the update of the filter weights;
- Error propagation through the update of the inverse covariance matrix.

Although stability in the update of the filter weights is necessary for a satisfactory performance, it does not give much insight into the phenomenon of explosive divergence. The analysis here focuses on the behaviour of the computed inverse covariance matrix.

Let

$$\tilde{\mathbf{P}}(n-1) = \mathbf{P}(n-1) + \mathbf{E}(n-1) \tag{3.15}$$

denotes the computed inverse covariance matrix with rounding error $\mathbf{E}(n-1)$. Substituting (3.15) into the update recursion (equations 1-4 in Table 3.1), one obtains the error propagation models for the three versions of the CRLS algorithm listed in Table 3.1.

(A) Version I

For Version I, the error propagation is given by

$$\begin{aligned}\mathbf{E}(n) &= \frac{1}{\beta}[\mathbf{I}-\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{E}(n-1)[\mathbf{I}-\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]^{\mathrm{H}} \\ &\quad + \frac{1}{\beta}[\mathbf{E}(n-1)-\mathbf{E}^{\mathrm{H}}(n-1)]\mathbf{x}(n)\mathbf{g}^{\mathrm{H}}(n) + O(\mathbf{E}^2)\end{aligned} \tag{3.16}$$

where $O(\mathbf{E}^2)$ indicates the order of magnitude of $\|\mathbf{E}(n-1)\|^2$.

Separating the symmetrical and anti-symmetrical part of the error

$$\begin{aligned}\mathbf{E}(n) &= \frac{1}{\beta}[\mathbf{I}-\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{E}_s(n-1)[\mathbf{I}-\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]^{\mathrm{H}} \\ &\quad + \frac{1}{\beta}[\mathbf{I}-\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{E}_a(n-1)[\mathbf{I}-\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]^{\mathrm{H}} \\ &\quad + \frac{2}{\beta}\mathbf{E}_a(n-1)\mathbf{x}(n)\mathbf{g}^{\mathrm{H}}(n) + O(\mathbf{E}^2)\end{aligned} \tag{3.17}$$

where the symmetrical part is given by

$$\mathbf{E}_s(n-1) = \frac{1}{2}[\mathbf{E}(n-1)+\mathbf{E}^{\mathrm{H}}(n-1)], \tag{3.18}$$

and the anti-symmetrical part is given by

$$\mathbf{E}_a(n-1) = \frac{1}{2}[\mathbf{E}(n-1) - \mathbf{E}^{\mathrm{H}}(n-1)]. \tag{3.19}$$

In steady state

$$\mathbf{x}(n)\mathbf{g}^{\mathrm{H}}(n) \approx (1-\beta)\mathbf{I}. \tag{3.20}$$

Hence, the matrix $\mathbf{E}_a(n-1)\mathbf{x}(n)\mathbf{g}^{\mathrm{H}}(n)$ is mainly anti-symmetrical. By analysing the average system of (3.19) or using the steady state approximation, it can be shown that the anti-symmetrical part of the error is propagated according to

$$\mathbf{E}_a(n) = \left[\beta + \frac{2}{\beta}(1-\beta)\right]\mathbf{E}_a(n-1). \tag{3.21}$$

Note that $\beta + \frac{2}{\beta}(1-\beta) > 1$. Therefore, the anti-symmetrical error is amplified leading to a build-up of rounding errors and eventually to explosive divergence as $\mathbf{P}(n)$ loses its positive definiteness. The simulation in [59] confirms this analysis.

(B) Versions II and III

For Versions II and III

$$\mathbf{E}(n) = \frac{1}{\beta}[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{E}(n-1)[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]^{\mathrm{H}} + O(\mathbf{E}^2). \tag{3.22}$$

Assume that the computation is accurate enough such that the nonlinear term $O(\mathbf{E}^2)$ can be ignored, we obtain

$$\mathbf{E}(n) = \frac{1}{\beta}[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{E}(n-1)[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]^{\mathrm{H}}. \tag{3.23}$$

But

$$\mathbf{P}(n) = \beta^{-1}[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)]\mathbf{P}(n-1), \tag{3.24}$$

Therefore

$$[\mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)] = \beta\mathbf{P}(n)\Phi(n-1). \tag{3.25}$$

Let us now vectorise $\mathbf{E}(n)$. Using (3.25), it follows that (see Appendix A)

$$\begin{aligned} cs(\mathbf{E}(n)) &= \beta\{[\mathbf{P}(n)\Phi(n-1)]^* \otimes [\mathbf{P}(n)\Phi(n-1)]\}cs(\mathbf{E}(n-1)) \\ &= \beta[\mathbf{P}^*(n) \otimes \mathbf{P}(n)][\Phi^*(n-1) \otimes \Phi(n-1)]cs(\mathbf{E}(n-1)) \end{aligned}, \tag{3.26}$$

where

$$cs(\mathbf{E}(n)) = \begin{bmatrix} e_1(n) \\ \cdots \\ e_N(n) \end{bmatrix} \tag{3.27}$$

with $e_i(n)$ denoting the ith column of the matrix $\mathbf{E}(n)$.

Using the results in Appendix B, it can be seen that the average system of (3.26) is given by (see Section 3.1.1)

$$cs(\mathbf{E}(n)) = \beta cs(\mathbf{E}(n-1)). \tag{3.28}$$

We thus conclude that Versions II and III of the CRLS algorithm have an exponentially stable error propagation mechanism for $\beta < 1$, and the total error is bounded **PROVIDED** (i) the precision is good enough such that the accumulated errors remain in the region where the linear approximation of the nonlinear error system is accurate; and (ii) the input signal is persistent exciting. This conclusion has also been presented in [59, 67] by arguing that the matrix $\beta\mathbf{P}(n)\Phi(n-1)$ is a contraction (i.e. its norm is smaller that unity) for all n.

The technique used above has also been applied to analyse and stabilise the Fast Transversal Filter (FTF) algorithm [51].

Although the error propagation analysis gives useful insight into the numerical behaviour of the CRLS algorithm, it leaves several questions unanswered

Why is there a difference between the normal equation based algorithms and the square-root algorithms? In [59], the author shows that Version II of the CRLS and two square-root implementations of the RLS algorithms are exponentially stable. However, it is well-known that in

practice the CRLS algorithms suffer from explosive divergence while its square-root counterparts do not.

How precise should the computation be in order that the effects of the nonlinear term can be ignored?

How precise should the computation be in order that the computed inverse covariance matrix remains positive definite?

3.2.3 Error Accumulation

So far, very little has been published about the error accumulation mechanism of the CRLS algorithm. In [11], the authors attempted to examine the nonlinear accumulation of rounding errors of the CRLS algorithm. However, the approach there is mostly qualitative and fails to guarantee numerical stability.

The recent work of Liavas and Regalia [35, 36] shows that explosive divergence can be avoided provided

- The input signal is persistently exciting;
- enough bits are used in the internal computation.

It is shown that for stationary input signal, an upper bound on the relative precision ε that guarantees no explosive divergence is given by

$$\varepsilon \leq \frac{1-\beta}{\Sigma} F_1(\rho_1^1), \tag{3.29}$$

where Σ is a constant such that the rounding errors made in one single iteration are bounded by

$$\|\varepsilon \mathbf{P}(n)\|_1 \leq \Sigma\varepsilon, \ \forall n, \tag{3.30}$$

the function $F_1(r)$ is defined by

$$F_1(r) = r - \frac{a_1 r^2}{\beta(1-\beta)(\beta - \phi^2 r)}, \tag{3.31}$$

ρ_1^1 is a constant given by

$$\rho_1^1 = \frac{\beta}{\phi^2}\left(1 - \frac{\sqrt{a_1^2 + a_1 A_2}}{a_1 + A_2}\right), \tag{3.32}$$

and where ϕ is a constant such that the input data vector is bounded by

$$\|\mathbf{x}(n)\| \leq \phi. \tag{3.33}$$

The constants a_1 and A_2 are defined by

$$a_1 = \phi^2[(1-\beta)P\phi^2 + 3 - 2\beta] \tag{3.34}$$

and

$$A_2 = \beta(1-\beta)\phi^2; \tag{3.35}$$

and P denotes the upper bound of the 1-norm of the inverse covariance matrix.

The corresponding upper bound of the accumulated error is given by

$$\|\Delta \mathbf{P}(n)\|_1 \le \rho_1^1 \le \frac{\beta^2(1-\beta)}{a_1 + A_2}, \quad \forall n \tag{3.36}$$

In [35, 36], the authors also showed that with the relative precision given by (3.29), the finite precision inverse covariance matrix remains positive definite for all n. This is because the largest eigenvalue of the error matrix is always smaller than the smallest eigenvalue of the inverse covariance matrix.

The above result establishes the relation between the numerical stability of the CRLS algorithm and the conditioning of the problem. However, as is expected, the bounds obtained is very conservative, i.e., the condition (3.29) is a sufficient but not necessary condition for numerical stability. Moreover, the requirement (3.29) could be too restrictive for many applications.

3.2.4 Numerical Consistency

The numerical stability of RLS algorithms has also been studied from the point of view of backward consistency [52,45, 46, 47]. In [52], the author

shows that the state of the conventional RLS algorithm is subject to the following structural constraint

$$\mathbf{P}(n) = \mathbf{P}^{\mathrm{H}}(n) > 0 \tag{3.37}$$

where $\mathbf{P}(n) > 0$ denotes that the matrix $\mathbf{P}(n)$ is positive definite. The difference between Version I and Versions II and III of the CRLS algorithm is highlighted by (3.37). Version I violates the equality part of the structural constraint while Versions II and III maintain it. However, there is no guarantee that the inequality part of the constraint is satisfied.

Suppose the inequality part of the constraint is violated at time n_0. This can be viewed as initialising the algorithm with an indefinite matrix at time instant n_0. The effect of such a initial value is demonstrated in [52] for the case where $\mathbf{P}(n)$ is a scalar. In particular, it was shown that whenever $p(n)$ became negative, a bursting phenomenon followed before the algorithm returned to a stable behaviour.

The numerical consistency approach also gives a clear explanation of the importance of persistent excitation. If the input to the filter is not persistently exciting, the structural constraint becomes

$$\mathbf{P}(n) = \mathbf{P}^{\mathrm{H}}(n) \geq 0, \tag{3.38}$$

which means the infinite precision inverse covariance matrix has zero eigenvalue(s). Thus even the slightest perturbation can make these zero eigenvalue negative, which then results in explosive divergence.

In conclusion, numerical instability is possible only when the numerical consistency is violated. However, this approach is merely qualitative. It does not provide much quantitative insight into how an algorithm reacts to rounding errors.

3.3 Square-Root Algorithms

By updating the square root of the (inverse) covariance matrix, the square-root algorithms can provide a much better numerical performance than their normal equation-based counterparts. In this section, we discuss the error propagation, the error accumulation, and the numerical consistency of the QRD-RLS algorithm.

3.3.1 QRD-RLS Algorithm

The QRD-RLS algorithm is listed below for convenience.

$\Phi^{1/2}(0) = \mathbf{0}$, $\mathbf{p}(0) = \mathbf{0}$

At iteration n,

$$\mathbf{Q}(n)\begin{bmatrix} \beta\Phi^{1/2}(n-1) & \beta\mathbf{p}(n-1) & \boldsymbol{0} \\ \mathbf{x}^{\mathrm{H}}(n) & d(n) & 1 \end{bmatrix} = \begin{bmatrix} \Phi^{1/2}(n) & \mathbf{p}(n) & \mathbf{g}(n) \\ \boldsymbol{0}^{\mathrm{H}} & e(n) & \alpha^{1/2}(n) \end{bmatrix}$$

$$\mathbf{w}(n) = \Phi^{-1/2}(n)\mathbf{p}(n)$$

Table 3.2 QRD-RLS Algorithm

It is well-known that this algorithm exhibits excellent numerical behaviour. Moreover, it possesses features such as parallelism and regularity which are highly desirable for VLSI implementation [23, 39, 19]. The state of this algorithm is given by $\left[\Phi^{1/2}(n)\ \mathbf{p}(n)\right]$.

3.3.2 Error Propagation

Assume that at time instant n, the algorithm state is subject to rounding errors, i.e.

$$\begin{aligned}\left[\tilde{\Phi}^{1/2}(n)\ \tilde{\mathbf{p}}(n)\right] &= \left[\Phi^{1/2}(n)\ \mathbf{p}(n)\right] + \mathbf{E}(n) \\ &= \left[\Phi^{1/2}(n) + \mathbf{E}_{\Phi}(n) \quad \mathbf{p}(n) + \mathbf{E}_{\mathbf{p}}(n)\right]\end{aligned} \tag{3.39}$$

This implies a perturbation in the covariance matrix and the cross covariance vector

$$\tilde{\Phi}(n) = \Phi(n) + \Delta\Phi(n) \tag{3.40}$$

and

$$\tilde{\theta}(n) = \theta(n) + \Delta\theta(n). \tag{3.41}$$

Suppose now that at all subsequent iterations the computations are exact. The algorithm update is then equivalent to [59]

$$\tilde{\Phi}(n+1) = \beta\tilde{\Phi}(n) + \mathbf{x}(n+1)\mathbf{x}^{H}(n+1) \tag{3.42}$$

and

$$\tilde{\theta}(n+1) = \beta\tilde{\theta}(n) + \mathbf{x}(n+1)d^*(n+1). \tag{3.43}$$

From (3.42), (3.43), it is immediately clear that the effect of a single error decays exponentially. A more rigorous proof is presented in [53].

In the discussion above, no assumption is made about the size of the error matrix $\mathbf{E}(n)$. In fact, the conclusion above holds for arbitrary large $\mathbf{E}(n)$ as long as no overflow occurs. Furthermore, the matrix $\tilde{\Phi}^{1/2}(n)$ is not even assumed to be nonsingular. This implies that the algorithm is exponentially stable even with poorly exciting input signals.

3.3.3 Error Accumulation

The error bounds for the QR decomposition of a matrix have been well documented in the numerical analysis literature. In [33, 18], the bounds for the QR decomposition of a matrix are presented. As expected, without an exponential weighting factor, the error bounds increase with n. With exponential weighting, the QR algorithm is unconditionally stable and the backward error bound (assuming floating point arithmetic) is given by [53]

$$\|\Delta\mathbf{X}(n)\|_F \leq \frac{C\varepsilon\rho}{(1-\beta)(1-C\varepsilon)} \tag{3.44}$$

where $\mathbf{X}$ is the data matrix defined by

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(0) & \ldots & \mathbf{x}(n) \\ d(0) & \ldots & d(n) \end{bmatrix}, \tag{3.45}$$

$\|\mathbf{M}\|_F$ denotes the Frobenius norm of $\mathbf{M}$, C is a constant depending on the order of the filter, ε is the machine precision, and ρ is the upper bound on the F-norm of the matrix $\mathbf{X}$. As seen in (3.44), the error bound is independent of the conditioning of the problem. This confirms the conclusion reached in Section 3.3.2.

3.3.4 Numerical Consistency

From a numerical consistency perspective, it can be easily shown that the QRD-RLS algorithm is free from explosive divergence. As mentioned in Section 3.2.4, the physical nature of the LS filtering problem places the structural constraint on the algorithm state such that the implied inverse covariance matrix satisfies (3.37). But the matrix $\Phi^{1/2}(n)$is the square root of the data covariance matrix. Since the square of an arbitrary matrix must be Hermitian and positive definite, therefore requirement (3.37) is always satisfied. From this observation, one can conclude that the QRD-RLS algorithm is backward consistent, and the effect of rounding errors can be modelled as additive noise. The same conclusion is also reached in [52].

When the input signal is not persistently exciting, the algorithm remains backward consistent in the sense that the constraint (3.38) is now satisfied. Hence the algorithm remains stable. However, caution has to be

taken when the filter weights are of interest since a zero eigenvalue in the covariance matrix can result in infinitely large filter weights.

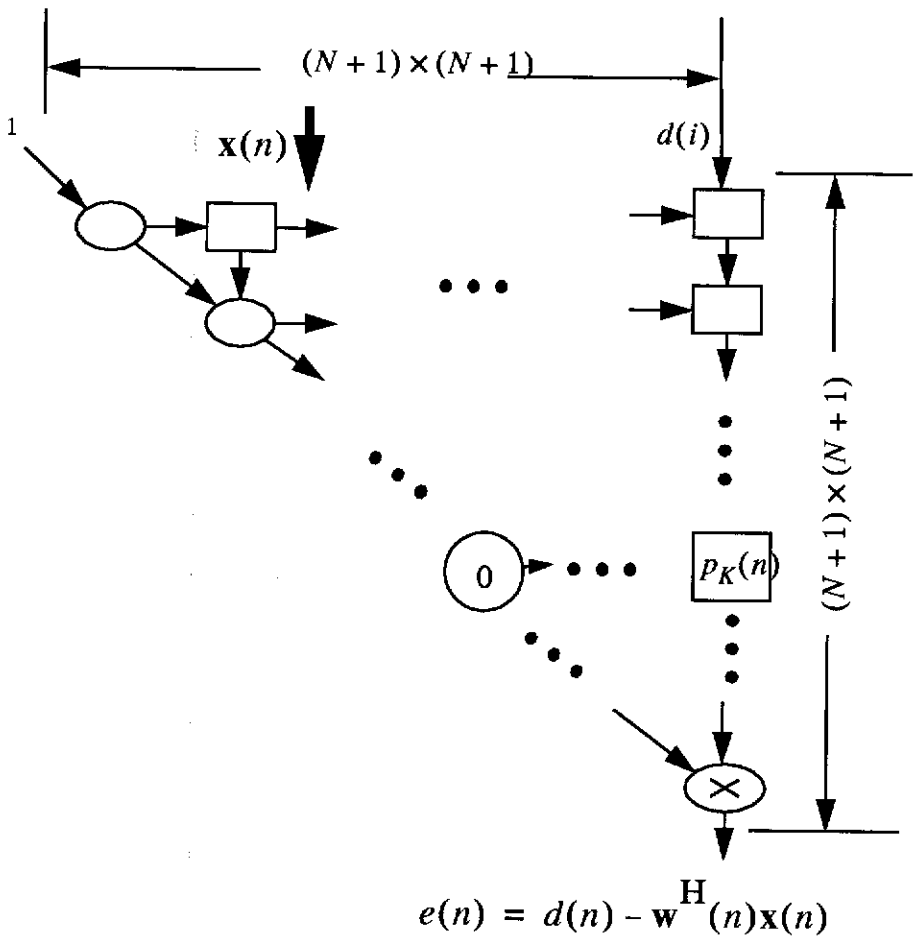


Figure 3.3: Systolic Array with Poorly Exciting Input

Consider the $(N+1)\times(N+1)$ systolic array given in [39] with poorly exciting input signal. Suppose the Kth boundary cell is zero as depicted in Figure 3.3.

We consider two situations:

- The array operating in its adaptive mode

  Since the signal is not persistently exciting, the input to the Nth boundary cell would be zero, and the Kth row is bypassed. This is nothing more than setting the Kth filter weight to zero to obtain the basic solution [15]. Therefore, the systolic array works perfectly well even without persistent excitation in adaptive mode;

- The array operating in frozen mode

  This happens when we flush the filter weights [50] (see Section 2.4.2). If the output of the boundary cell in a frozen network is

$$c = \frac{x}{r} \tag{3.46}$$

where x is its input and r is the value stored in the cell, as is defined in [50], a zero boundary cell will lead to infinitely large filter weights. However, a small modification can be made to overcome this problem:

$$c = \begin{cases} 0 \ , \forall r = 0 \\ \frac{x}{r}, \ otherwise \end{cases} . \tag{3.47}$$

This bypasses the rows with zero boundary cell and sets the corresponding filter weights to zero. The filter weights so obtained is

the basic solution to an LS problem for rank-deficient case [15].

As shown above, the QRD-RLS algorithm is stable even in the absence of persistent excitation.

3.4 Summary

In this chapter, a review of the existing results on the numerical properties of RLS algorithms are reviewed. It is shown that with persistent excitation and enough numerical precision, the CRLS algorithm can be designed to be free from explosive divergence. On the other hand, the QRD-RLS algorithm is unconditionally stable.

4 Constraint Drift

As mentioned in Chapter 1, when linear constraints are imposed on an adaptive filter, the computed filter weights may diverge from the constraints due to the accumulation of rounding errors. When this happens, the filter may treat the useful signals as interferences and suppress them. We call this phenomenon *constraint drift*.

In this chapter, the constraint drift characteristics of the LC-FLS, the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithms are studied. The LC-FLS algorithm is examined in the first section, followed by a analysis of the linear systolic array for MVDR beamforming. The LC-QRD-RLS algorithm is studied in Section 4.3.

4.1 Linearly Constrained Fast Least Squares Algorithm

The LC-FLS algorithm is listed in Table 4.1 for convenience.

As mentioned in Chapter 2, the algorithm can be divided into a constrained part and an unconstrained part, and the constrained part does not alter the properties of the unconstrained part. Therefore, in the following analysis, we assume that the Kalman gain is exact.

1 Update the Kalman gain $\mathbf{g}(n)$

2 Update matrix $\mathbf{Q}(n)$

2.1 $\mathbf{l} = \mathbf{C}^{H}\mathbf{g}(n)$

2.2 $\mathbf{m}^{H} = \mathbf{x}^{H}(n)\mathbf{Q}(n-1)$

2.3 $$\hat{\mathbf{Q}}(n) = [\mathbf{Q}(n-1) - \mathbf{g}(n)\mathbf{m}^{H}]\left[\mathbf{I} + \frac{\mathbf{l}\mathbf{m}^{H}}{1 - \mathbf{m}^{H}\mathbf{l}}\right]$$

2.4 $$\mathbf{Q}(n) = \hat{\mathbf{Q}}(n) + \mathbf{C}[\mathbf{C}^{H}\mathbf{C}]^{-1}[\mathbf{I} - \mathbf{C}^{H}\hat{\mathbf{Q}}(n)]$$

3 Calculate filter weights $\mathbf{w}(n)$.

3.1 $e(n-1) = d(n) - \mathbf{w}^{H}(n)\mathbf{x}(n)$

3.2 $$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n)e(n-1) + \mathbf{Q}(n)\{\mathbf{f} - \mathbf{C}^{H}[\mathbf{w}(n-1) + \mathbf{g}(n)e(n-1)]\}$$

Table 4.1 Summary of the LC-FLS Algorithm

We first note the following:

- The update of the matrix $\mathbf{Q}(n)$ (2.3 in Table 4.1) involves a multiplication of a matrix $\mathbf{I} + \frac{\mathbf{l}\mathbf{m}^{H}}{1 - \mathbf{m}^{H}\mathbf{l}}$ whose norm is larger than unity. Although this does not necessarily mean that the algorithm is unstable, it does give strong indication that numerical instability is possible;
- From the numerical consistency point of view, the algorithm state $\{\mathbf{P}(n), \mathbf{Q}(n), \mathbf{w}(n)\}$ must satisfy the following conditions to be a

valid solution to the linearly constrained LS filtering problem. (Note that $\mathbf{P}(n)$ may not necessarily appear in the algorithm state if algorithms such as the FTF are used to compute the Kalman gain. It is used here just for the convenience of presentation.)

$$\mathbf{P}(n) = \mathbf{P}^{\mathrm{H}}(n) > 0$$

$$\mathbf{C}^{\mathrm{H}}\mathbf{w}(n) = \mathbf{f}$$

$$\mathbf{Q}(n) = \mathbf{P}(n)\mathbf{C}[\mathbf{C}^{\mathrm{H}}\mathbf{P}(n)\mathbf{C}]^{-1} \tag{4.1}$$

Since we assume that the Kalman gain is computed exactly, the first condition of (4.1) is satisfied. However, conditions 2 and 3 will surely not hold in a finite precision implementation. From (4.1), we can see that the matrix $\mathbf{Q}(n)$ is redundant in the sense that it is uniquely defined by the matrix $\mathbf{P}(n)$. This redundancy may give rise to an unstable error propagation mechanism [46].

4.1.1 Error Propagation

An analysis of the error propagation mechanism of the LC-FLS algorithm is presented in this section. This analysis is validated by simulation results.

The following assumptions, as well as those stated in Section 3.1.1, are made in the analysis.

- The update of the Kalman gain is exact.
- At iteration n - 1,

$$\tilde{\mathbf{Q}}(n-1) = \mathbf{Q}(n-1) + \mathbf{E}(n-1), \tag{4.2}$$

with

$$\mathbf{C}^{H}\mathbf{E}(n-1) = \mathbf{0}. \tag{4.3}$$

(4.3) holds in view of the projection operation in (2.23) and the assumption that the computations after iteration n are exact.

Now substitute (4.2) into the update equation of the matrix $\mathbf{Q}(n)$ (2.3 in Table 4.1) and ignore all nonlinear terms, it follows that

$$\begin{aligned}\tilde{\mathbf{Q}}(n) &= \mathbf{M}(n)\mathbf{Q}(n-1)\mathbf{L}(n) \\ &\quad + \mathbf{M}(n)\mathbf{Q}(n-1)\mathbf{C}^{\mathrm{H}}\mathbf{E}(n-1)\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n) \\ &\quad + \mathbf{M}(n)\mathbf{E}(n-1)\mathbf{L}(n) \\ &= \mathbf{Q}(n) + \mathbf{E}(n)\end{aligned} \tag{4.4}$$

where

$$\mathbf{M}(n) = \mathbf{I} - \mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n) \tag{4.5}$$

and

$$\mathbf{L}(n) = \left[\mathbf{I} + \frac{\mathbf{C}^{\mathrm{H}}\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n)\mathbf{Q}(n-1)}{1 + \mathbf{Q}(n-1)\mathbf{x}^{\mathrm{H}}(n)\mathbf{C}^{\mathrm{H}}\mathbf{g}(n)}\right]. \tag{4.6}$$

Hence the effect of the rounding error $\mathbf{E}(n)$ at iteration $n+1$ is given by

$$\begin{aligned}\mathbf{E}(n) &= \mathbf{M}(n)\mathbf{Q}(n-1)\mathbf{C}^{\mathrm{H}}\mathbf{E}(n-1)\mathbf{g}(n)\mathbf{x}^{\mathrm{H}}(n) \\ &\quad + \mathbf{M}(n)\mathbf{E}(n-1)\mathbf{L}(n)\end{aligned} \tag{4.7}$$

Using (4.3) and recognizing that

$$\mathbf{M}(n) = \beta\mathbf{P}(n)\Phi(n-1) \tag{4.8}$$

$$\mathbf{L}(n) = \beta^{-1}\psi(n-1)\Theta(n) \tag{4.9}$$

where $\mathbf{P}(n)$ is the inverse data covariance matrix at time instant n, $\Phi(n-1)$ is the data covariance matrix at iteration $n-1$, and

$$\psi(n-1) = [\mathbf{C}^{\mathrm{H}}\mathbf{P}(n-1)\mathbf{C}] \tag{4.10}$$

and

$$\Theta(n) = \psi^{-1}(n), \tag{4.11}$$

(4.7) becomes

$$\delta(n) = \mathbf{P}(n)\Phi(n-1)\mathbf{E}(n-1)\psi(n-1)\Theta(n) \tag{4.12}$$

We next column stack the error matrix. It follows that

$$\begin{aligned}cs(\mathbf{E}(n)) &= \{[\Theta(n)\psi(n-1)]^* \otimes [\mathbf{P}(n)\Phi(n-1)]\}cs(\mathbf{E}(n-1)) \\ &= [\Theta^*(n)\mathbf{P}(n)] \otimes [\psi^*(n-1)\Phi(n-1)]cs(\mathbf{E}(n-1))\end{aligned}, \tag{4.13}$$

and by using the results in Appendix B, it follows that

$$\mathbf{E}(n) = \mathbf{E}(n-1). \tag{4.14}$$

Now, since by assumption, no further rounding errors are introduced, the projection operation in the update of the matrix $\mathbf{Q}(n)$ has no effect on the error propagation.

To this point, we have shown that the rounding errors made in a certain iteration stay as they are in subsequent iterations. This results in a random walk model [24, 52]. Thus, there is no guarantee that the accumulated error is bounded.

In [49], the author suggested that in the case of a single constraint, the quantity $\Theta(n)$, and thus $\mathbf{Q}(n)$, should be calculated directly rather than updated recursively. This leaves only the unconstrained part of the algorithm (i.e. step 2 in Table 4.1) running adaptively, and numerical stability can be achieved provided the unconstrained part is stable. However, when the filter is subject to multiple constraints, direct calculation of Θ and its inverse is computationaly expensive and is impractical.

In conclusion, the LC-FLS algorithm is unstable with respect to finite word-length effects.

4.1.2 Simulation Results

In this section, we validate the above analysis with simulation results.

The first simulation concerns the propagation of a single error of the LC-FLS algorithm. In this simulation, a 16th order filter was constrained to have unit response at normalised frequencies 0.1π and 0.25π. This yields the following constraint

$$\begin{bmatrix} 1 & \cos(0.2\pi) & \cos(0.4\pi) & \dots & \cos(3.2\pi) \\ 0 & \sin(0.2\pi) & \sin(0.4\pi) & \dots & \sin(3.2\pi) \\ 1 & \cos(0.5\pi) & \cos(\pi) & \dots & \cos(8\pi) \\ 0 & \sin(0.5\pi) & \sin(\pi) & \dots & \sin(8\pi) \end{bmatrix} \mathbf{w} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \tag{4.15}$$

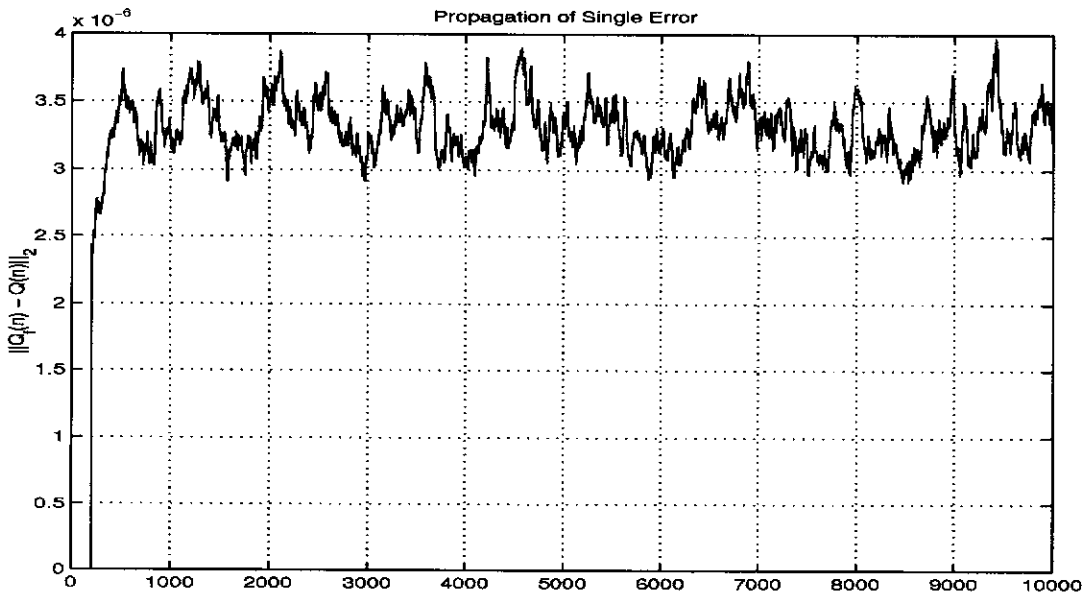


Figure 4.1: Single Error Propagation

The filter was fed with a Gaussian white noise with unit variance. The Kalman gain was calculated with the inverse QR algorithm [24] in full precision (1-bit sign, 11-bit exponent, 52-bit mantissa) so that the

unconstrained part of the algorithm is stable. At iteration 201, an error was introduced by updating the matrix $\mathbf{Q}(n)$ with the mantissa rounded to 16 bits. The deviation of the computed matrix $\mathbf{Q}_f(n)$ from its full precision value is depicted in Figure 4.1. It was observed that the error introduced at iteration 201 grew slightly then stayed at a certain level. This confirms the theoretical analysis.

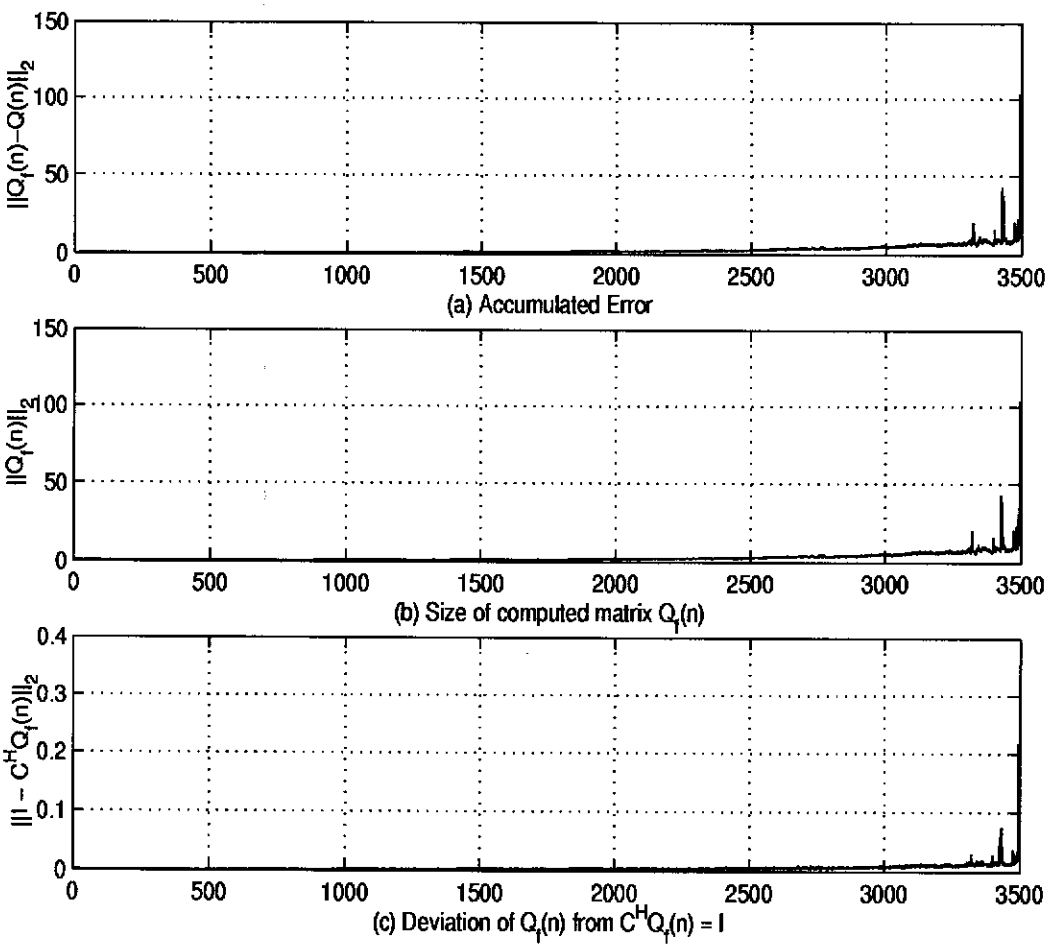


Figure 4.2: Accumulated Error

The second simulation focused on the effect of the accumulated errors. In this simulation, the constraint (4.15) was imposed on two 16th order filters, one operating with full precision and the other with a precision of 8 bits in mantissa for updating the matrix $\mathbf{Q}(n)$. Input signal consisted of three sinusoids of normalised frequencies 0.15π, 0.175π and 0.25π, and an additive Gaussian white noise of unit variance. The forgetting factor is set to 0.99.

Figure 4.2 illustrates the error build up. Comparing Figure 4.2 (a) with Figure 4.2 (b), it can be seen that the computed quantity $\mathbf{Q}_f(n)$ is completely erroneous after 3000 iterations. With 8 bits in the mantissa and 11 bits in the exponent, it is highly possible for the errors to grow to such a level that the size of $\mathbf{Q}_f(n)$ exceeds the dynamic range provided by the given wordlength and the filter overflows at around the 5000th iteration. It is also observed that when the error grows to a certain size, the deviation of the matrix $\mathbf{Q}_f(n)$ from the constraint specified in (2.22) became so significant that some form of explosive phenomenon occurred. Other measurements such as frequency response, and instantaneous output power are presented in the Section 4.3 together with the simulation results of the LC-QRD-RLS algorithm for comparison.

4.2 Linear Systolic Array for MVDR Beamforming

The linear systolic array for MVDR beamforming of Yang and Böhme [64-66] is summarized in Table 4.2.

It is observed in Section 2.3.2 that this algorithm has a structure very similar to the LC-FLS algorithm in that its constrained part is adaptive and is uniquely defined by the unconstrained part. This structure, as seen in the previous section, may cause the algorithm to diverge.

4.2.1 Error Propagation

Consider at time instant $n-1$, the computed vector $\mathbf{t}(n)$ is erroneous

$$\tilde{\mathbf{t}}(n-1) = \mathbf{t}(n-1) + \mathbf{e}(n-1) \tag{4.16}$$

Initialize: $\Phi^{-1/2}(0)$, $\mathbf{t}(0)$

At iteration n

1 Unitary transformation

$$\mathbf{Q}(n)\begin{bmatrix} \mu\Phi^{1/2}(n-1) & \mathbf{0} & \dfrac{\mathbf{t}(n-1)}{\mu} \\ \mathbf{x}^{H}(n) & 1 & 0 \end{bmatrix} = \begin{bmatrix} \Phi^{1/2}(n) & \mathbf{b} & \mathbf{t}(n) \\ \mathbf{0}^{H} & \alpha(n) & z(n) \end{bmatrix}$$

2 Beamforming output computation

$$D(n) = D(n-1)/\mu - |z(n)|^2$$

$$y(n) = \frac{\alpha(n)z^*(n)}{D(n)}$$

Table 4.2 Linear Systolic Array for MVDR Beamforming

Let [50]

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{A}(n) & \mathbf{b}(n) \\ \mathbf{r}^{\mathrm{H}}(n) & \alpha(n) \end{bmatrix}, \tag{4.17}$$

where $\mathbf{A}(n)$ is unitary. Substituting (4.16), (4.17) into the update equation of $\mathbf{t}(n)$ (2.26), and noting that because $\mathbf{Q}(n)$ is independent of $\tilde{\mathbf{t}}(n-1)$, it follows that

$$\mathbf{e}(n) = \mathbf{A}(n)\frac{\mathbf{e}(n-1)}{\mu}. \tag{4.18}$$

(4.18) clearly indicates that the error propagation mechanism of the linear systolic array for MVDR beamforming is exponentially unstable for $\mu<1$.

It is also shown in [64] that if

$$\tilde{\Phi}^{\mathrm{H}/2}(n-1)\tilde{\mathbf{t}}(n-1) = \mathbf{C} + \delta(n-1), \tag{4.19}$$

where the deviation of $\tilde{\Phi}^{1/2}(n-1)\tilde{\mathbf{t}}(n-1)$ from C signifies the deviation of the implied filter weights from the constraint, and if the subsequent computations are to be carried out accurately, then at iteration n

$$\begin{aligned} \tilde{\Phi}^{\mathrm{H}/2}(n)\tilde{\mathbf{t}}(n) &= \tilde{\Phi}^{\mathrm{H}/2}(n-1)\mathbf{A}^{\mathrm{H}}(n)\mathbf{A}(n)\tilde{\mathbf{t}}(n-1) \\ &= \mathbf{C} + \delta(n-1) \end{aligned}. \tag{4.20}$$

Hence, the algorithm would fail to enforce the constraint in finite precision arithmetic.

4.2.2 Simulation Results

In this section, two computer simulations of the linear systolic array for MVDR beamforming are presented. In the simulations, a 5-tap filter is constrained to have unit response at normalised frequency 0.1π, resulting in the following complex constraint

$$\begin{bmatrix} 1 & e^{-j0.2\pi} & e^{-j0.4\pi} & e^{-j0.6\pi} & e^{-j0.8\pi} \end{bmatrix} \mathbf{w} = 1 . \tag{4.21}$$

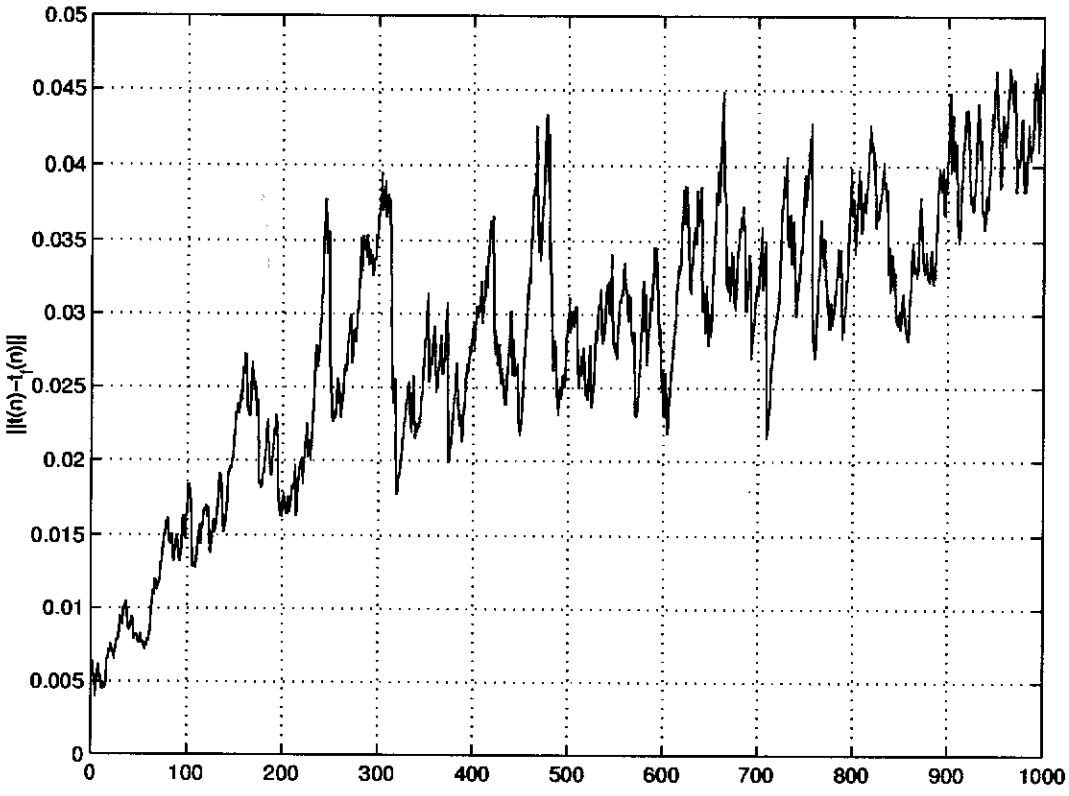


Figure 4.3: Accumulated Error-Linear Systolic Array

Forgetting factor is set to 0.9. 8-bit mantissa precision is used to simulate the finite precision effects

The first simulation examined the accumulation of rounding errors. The filter is fed with Gaussian white noise with unit variance. Full precision performance is used as reference. The result is depicted in Figure 4.3. It is seen that the accumulated errors grew with time.

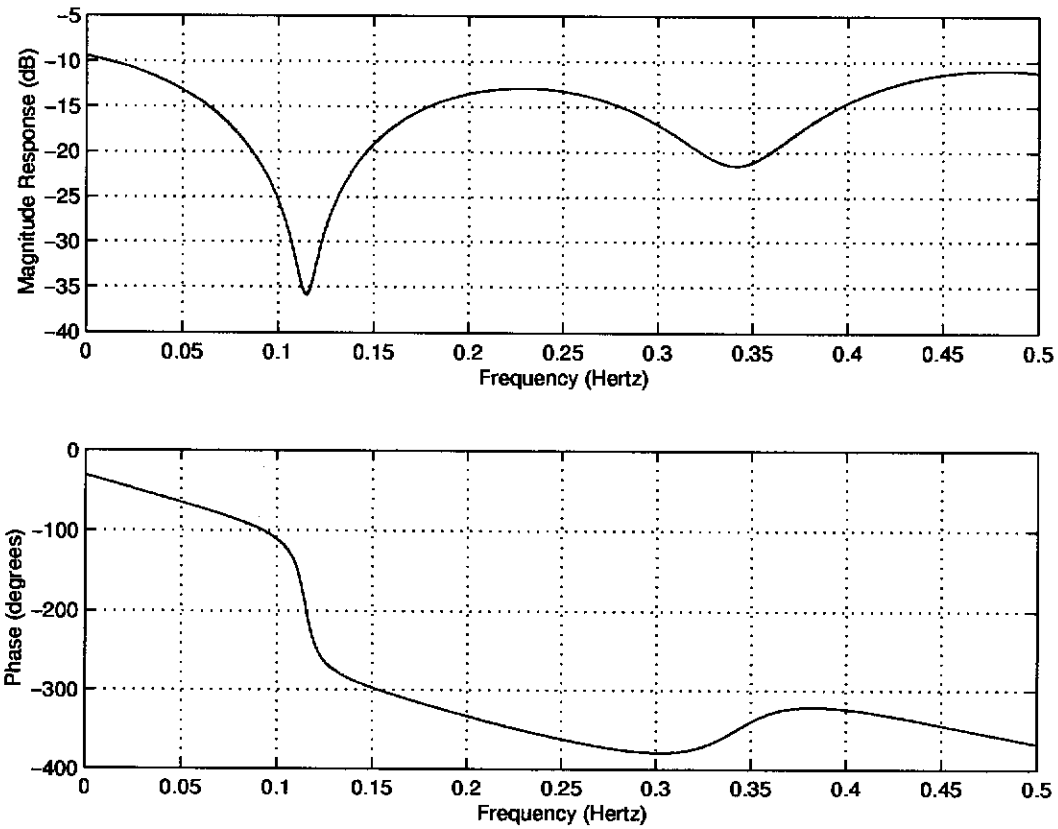


Figure 4.4: Frequency Response

In the second simulation, the filter is fed with a sinusoid of normalised

frequency 0.1π in Gaussian white noise of unit variance. The frequency response at iteration 1000 is plotted in Figure 4.4. It is clearly seen in Figure 4.4 that rather than protecting the signal, the filter actually suppressed it. This demonstrates the ill-effect of constraint drift as suffered by the algorithm.

4.3 Linearly Constrained QRD-RLS Algorithm

The LC-QRD-RLS algorithm is derived in Section 2.4 and is listed in Table 4.3 for convenience.

1	Pre-calculate matrix **B** and vector $\mathbf{w}_c$;
2	Signal pre-processing $\underline{d}(n) = d(n) - \mathbf{w}_c^H \mathbf{x}(n)$ $\underline{\mathbf{x}}(n) = \mathbf{B}^H \mathbf{x}(n)$
3	LS filtering with $\underline{d}(n)$ as reference signal and $\underline{\mathbf{x}}(n)$ as filter input using QRD-RLS algorithm.

Table 4.3 Summary of LC-QRD-RLS Algorithm

As mentioned in Section 2.4.3, the LC-QRD-RLS algorithm is a special case of the GSC. The adaptive part of the algorithm is nothing more than an unconstrained QRD-RLS algorithm with reduced dimensions which has been proved to be backward consistent in Section 3.3.2 and

Section 3.3.4. Hence, we can expect the LC-QRD-RLS algorithm to give the exact solution to a linearly constrained LS filtering problem which is perturbed from the original one.

4.3.1 Backward Error Analysis

The three steps of the LC-QRD-RLS algorithm can be viewed as three sub-routines. The rounding errors introduced in every sub-routine can be interpreted as a perturbation of the input to it. We consider the systolic array implementation of this algorithm.

I) Precomputation of the frozen network

The pre-computation of the block matrix **B** and the fixed FIR filter $\mathbf{w}_c$ is nothing more than a QR decomposition of the $K \times (N+1)$ matrix $\left[\mathbf{C}^H \; -\mathbf{f}\right]$. In [18], the computed result is shown to be the exact QR decomposition of the matrix

$$\left[\tilde{\mathbf{C}}^H \; -\tilde{\mathbf{f}}\right] = \left[\mathbf{C}^H \; -\mathbf{f}\right] + \mathbf{E}_c \tag{4.22}$$

with

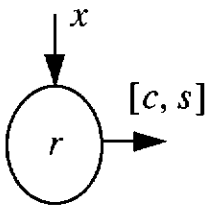
$$\left\|\mathbf{E}_c\right\|_F \le \eta(2K-3)(1+\eta)^{2K-4}\left\|\left[\mathbf{C}^H \; -\mathbf{f}\right]\right\|_F \tag{4.23}$$

where η is the bound on the backward error when applying Givens rotation to a two-element vector and whose value depends on the

implementation of the Givens rotation [18].

II) Pre-filtering of input signal

Boundary Cell:

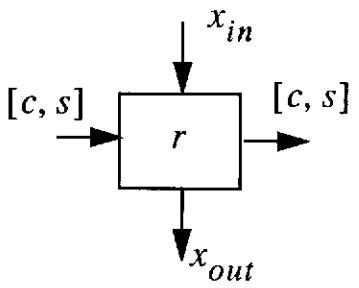


$$r' = \sqrt{r^2 + x^2}$$

$$s = \begin{cases} x/r', & x \neq 0 \\ 0, & x = 0 \end{cases}$$

$$c = \begin{cases} r/r', & x \neq 0 \\ 1, & x = 0 \end{cases}$$

Internal Cells



$$x_{out} = cx_{in} - sr$$

Table 4.4 Modification to the Frozen Network

As discussed before, the pre-filtering of the input signal by passing the signal through the frozen network of a systolic array [50] corresponds to a Gaussian elimination. But as presented in [15], Gaussian elimination without pivoting can lead to an arbitrary large error. A modification to the operation of the frozen network can be made to avoid this difficulty. This modification is presented in Table 4.4 (see Appendix C for the derivation). It can be viewed as applying a unitary transformation to the matrix

$$\begin{bmatrix} \mathbf{U}\ \mathbf{V}\ \mathbf{m} \\ \mathbf{x}(n)\ 0 \end{bmatrix}, \tag{4.24}$$

then restoring the frozen network with the matrices **U**, **V** and **m**. This pre-filtering operation introduces an error

$$\|\mathbf{E}_{\mathrm{p}}\|_F \leq \eta K(1+\eta)^{K-1}\left(\left\|\begin{bmatrix} \mathbf{U}\ \mathbf{V}\ \mathbf{m} \end{bmatrix}\right\|_F + X\right) \tag{4.25}$$

where X is the upper bound of the 2-norm of the input signal vector $\mathbf{x}(n)$. This error is independent of the conditioning of the input signal and the constraint matrix.

III) Unconstrained LS filtering

The backward error bound for the unconstrained LS filtering with the QRD-RLS algorithm has been discussed in Section 3.3.3, and it is shown that the error is bounded by

$$\|\mathbf{E}_{\mathrm{X}}(n)\|_F \leq \frac{C\varepsilon\rho}{(1-\beta)(1-C\varepsilon)}. \tag{4.26}$$

This error may just as well be viewed as an error in the input signal to the filter as follows

$$\begin{bmatrix} \mathbf{0}\ \mathbf{E}_{\mathrm{X}}(n) \end{bmatrix}. \tag{4.27}$$

IV) Total Error

Together with the error introduced in signal pre-filtering stage, the total error in the input signal is given by

$$\mathbf{E}(n) = \mathbf{E}_{\mathrm{p}}(n)\Lambda + \mathbf{E}_{\mathrm{X}}(n) \tag{4.28}$$

where

$$\Lambda = diag\left[\beta^{\frac{n}{2}} \quad \beta^{\frac{n-1}{2}} \quad \ldots \quad 1\right] \tag{4.29}$$

So far, we have shown that the LC-QRD-RLS algorithm provides the exact solution to the following perturbed linearly constrained LS filtering problem

$$\min_{\mathbf{w}(n)}\left[\sum_{i=0}^{n} \beta^{n-i}\left|\tilde{d}(i) - \mathbf{w}^{\mathrm{H}}(n)\tilde{\mathbf{x}}(i)\right|^2\right] \text{ subject to } \tilde{\mathbf{C}}^{\mathrm{H}}\mathbf{w}(n) = \tilde{\mathbf{f}} \tag{4.30}$$

where the perturbation on the input signal is given by (4.25), (4.26) and (4.28), and that on the imposing constraint is given by (4.23). Hence the algorithm is perfectly stable with respect to arithmetic errors.

A geometrical interpretation which gives some interesting insights into the numerical behaviour of the algorithm is included in Appendix D.

4.3.2 Simulation Results

In this section, simulation results focusing on the effect of accumulated error in the LC-QRD-RLS are presented. These results are presented together with those of the LC-FLS algorithm for comparison.

Here, the same simulation scenario as that in the second simulation in Section 4.1.2 is used. All computations in the LC-QRD-RLS algorithm except for weight flushing were carried out with 8 bits in the mantissa.

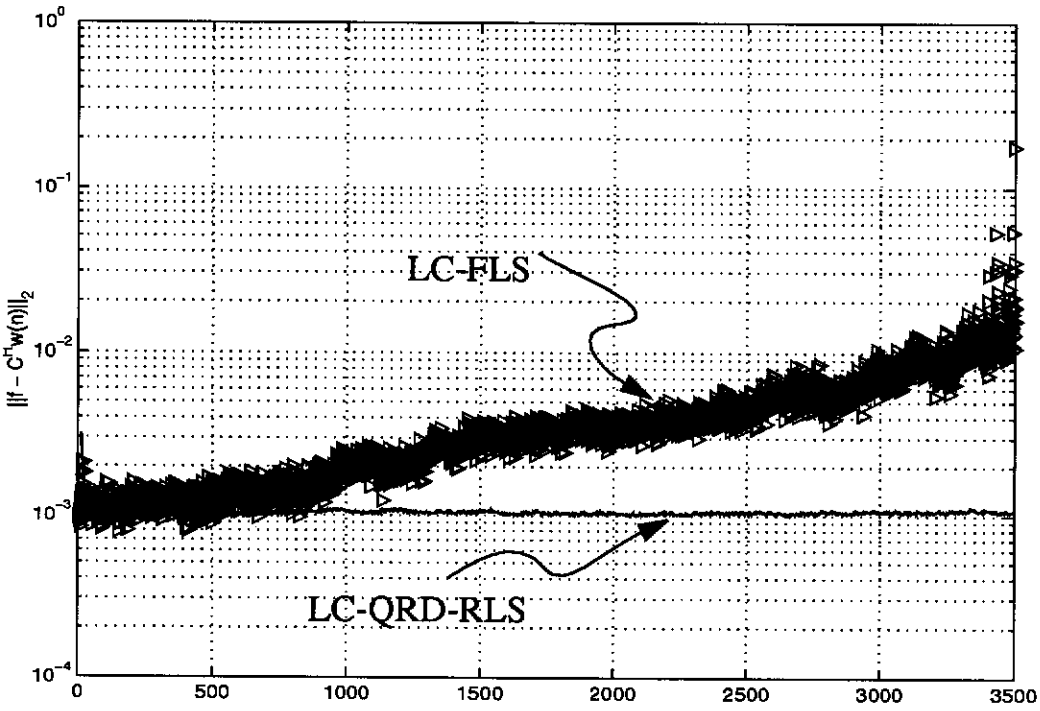


Figure 4.5: Constrain Drift

Figure 4.5 depicts the deviation of the computed weights from the

imposed constraint. As observed in the figure, the weights computed by the LC-FLS algorithm deviated further and further away from the imposed constraint due to the accumulation of rounding errors. In contrast, there is no sign that the weights given by the LC-QRD-RLS algorithm diverge from the constraint.

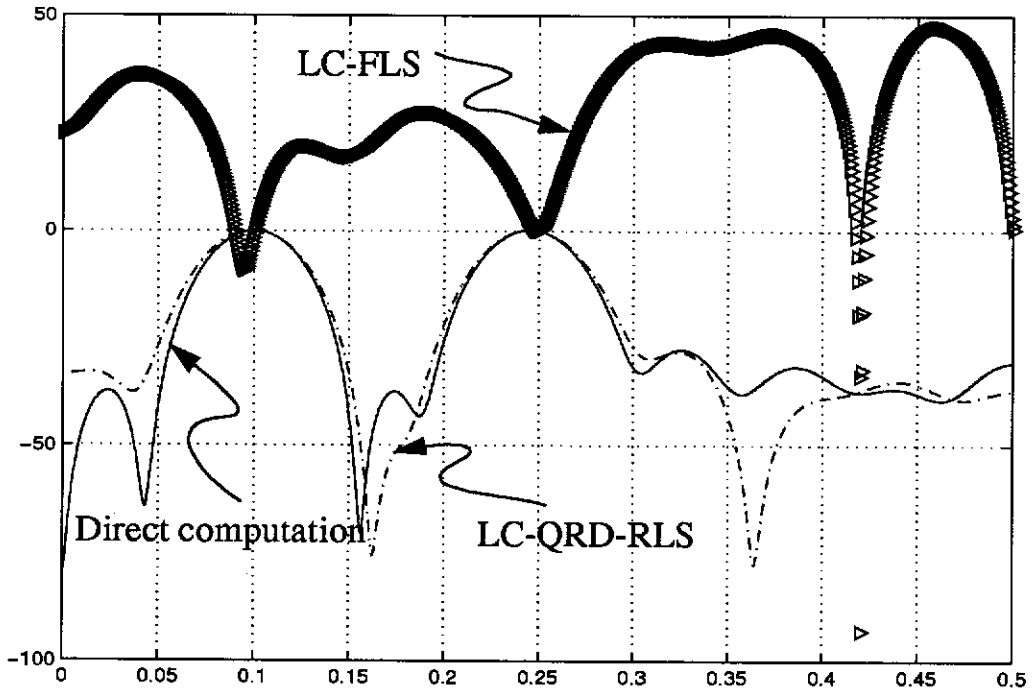


Figure 4.6: Frequency Response

The frequency response of the finite precision filters were observed at iteration number 3500. For reference, we computed the full precision filter weights by computing it directly from the closed form expression of the optimum solution (2.1). In Figure 4.6, it is seen that the filter updated by the LC-FLS algorithm completely failed to minimize the filter output

and amplified the interferences and noise. On the other hand, the LC-QRD-RLS algorithm yielded a performance which is close to that obtained by direct computation.

Figure 4.7 illustrates the performance of the algorithms in terms of the instantaneous filter output power. As expected, the output power of the LC-FLS filter grew larger and larger while that of the LC-QRD-RLS algorithm remained roughly the same as that given by direct computation.

The simulation results confirm that the LC-QRD-RLS algorithm is stable in finite precision arithmetic and highlight its superior performance compared to the LC-FLS algorithm.

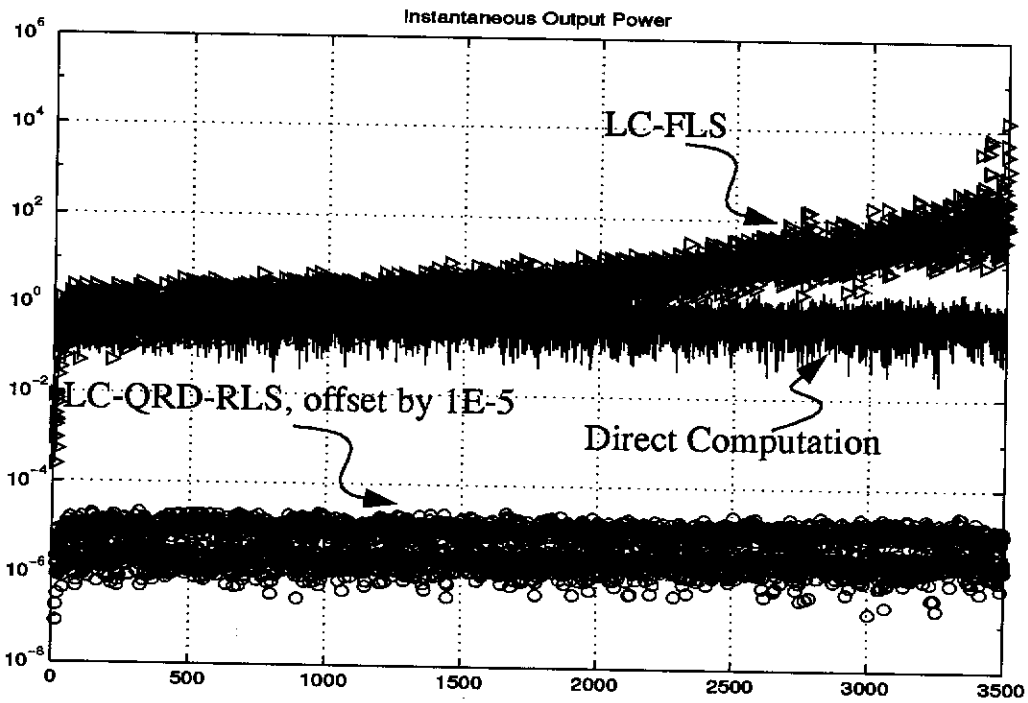


Figure 4.7: Instantaneous Output Power

4.4 Summary

In this chapter, the numerical properties of the LC-FLS, the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithms are studied. It is shown that the constraint drift error of the LC-FLS algorithm follows a random walk type error propagation mechanism and thereby is unstable with respect to rounding errors. Similarly, the linear systolic array for MVDR beamforming propagates a single error in the unconstrained part in an exponentially unstable fashion. In contrast, the LC-QRD-RLS algorithm is perfectly stable with finite precision arithmetic and provides the exact solution to a linear constrained LS filtering problem with perturbed constraints and input data. The theoretical analysis is confirmed by simulation.

5 Conclusions

5.1 Summary

The numerical stability properties of three linearly constrained adaptive LS filtering algorithms, namely the LC-FLS, the linear systolic array for MVDR beamforming and the LC-QRD-RLS algorithm, are studied in this thesis. We showed that all three algorithms can be separated into a constrained part and an unconstrained part where the unconstrained part is independent of the constrained one.

Regarding the unconstrained part, the well-known phenomenon of explosive divergence is discussed from different points of view. It is shown that both the CRLS algorithm (provided the symmetrical property of the computed inverse covariance matrix is preserved) and the QRD-RLS algorithm is exponentially stable. However, persistent excitation and sufficiently good numerical resolution is required to ensure the CRLS algorithm to be free of explosive divergence. In contrast, the QRD-RLS algorithm is unconditionally stable even in the absence of persistent excitation. The QRD-RLS algorithm provides an exact solution of a LS filtering problem with perturbed input data. Error bounds for both algorithms are presented. By using the numerical consistency argument, it is shown that the property of minimality of the algorithm state (i.e., the property that the algorithm state is not subject to any structural constraint) is the key difference that gives rise to the different numerical behaviour of

these two algorithms. The algorithm state of the CRLS algorithm is not minimal, therefore, in finite precision arithmetic, it is possible for the structural constraint on the algorithm state to be violated and thus explosive divergence is resulted. On the other hand, the algorithm state of the QRD-RLS algorithm is not subject to any structural constraint even with poorly exciting input. This ensures that the QRD-RLS algorithm is free from explosive divergence under any circumstance.

The constrained parts of the LC-FLS algorithm and the linear systolic array for MVDR beamforming are adaptive whose values are uniquely defined by the unconstrained parts. An error propagation analysis shows that this structure leads to an unstable error propagation mechanism. Specifically, the LC-FLS algorithm demonstrates a random walk type error propagation model, and the linear systolic array for MVDR beamforming amplifies the errors in the constrained part. In contrast, the constrained part of the LC-QRD-RLS algorithm consists of two fixed filters and the algorithm state of the algorithm remains minimal. A backward error analysis shows that the LC-QRD-RLS algorithm gives the exact solution to a linearly constrained LS adaptive filtering problem with perturbed constraint and perturbed input data. Some modification has been proposed to avoid some potential numerical difficulties that may be experienced by the algorithm.

The theoretical findings are confirmed by simulation results.

5.2 Future Research Areas

- Stabilization of the LC-FLS algorithm and the linear systolic array for MVDR beamforming

  In Section 4.1, we showed that the redundancy in these algorithms contributes to the unstable propagation of errors. However, it is shown [51] that the redundancy in the FTF algorithm can be exploited to stabilize the algorithm. It could be possible that some feedback mechanisms can be introduced so that the errors in the constrained parts of these two algorithm can be propagated in a stable manner;

- Development of parallel algorithms and architecture

  Adaptive filters are computationally intensive and sometimes, the computation load limits the application of a certain algorithm in real time. Despite the rapidly increased computational power of modern computer systems, it is still highly desirable to employ parallel structures such as systolic arrays so that the computational load is distributed through a set of simple processors operating concurrently.

References

[1] Anderson, B.D.O. and Johnson, C.R. Jr, "Exponential Convergence of Adaptive Identification and Control Algorithms," *Automatica*, Vol. 18, No. 1, pp. 1-13, 1982

[2] Anderson, B.D.O., et al., *Stability of Adaptive Systems: Passivity and Averaging Analysis*, Cambridge, MA: MIT Press, 1986

[3] Applebaum, S.P. and Chapman, D.J., "Adaptive Arrays with Main Beam Constraints", *IEEE Tran. AP*, VOL AP-24, NO. 5, pp. 650-662, 1976

[4] Ardalan, S.H., "Floating-Point Error Analysis of Recursive Least Squares and Least Mean-Squares Adaptive Filters," IEEE Trans. Circuits Syst., vol CAS-33, no. 12, pp. 1192-1208, Dec. 1986

[5] Ardalan, S.H. and Alexander, S.T., "Fixed-Point Roundoff Error Analysis of the Exponentially Windowed RLS Algorithm for Time-Varying Systems," *IEEE Trans. ASSP*, vol. ASSP-35, no. 6, pp. 770-783, June 1987

[6] Bierman, G.J. and Thornton, C.L., "Numerical Comparison of Kalman Filter Algorithms: Orbit Determination Case Study", *Automatica*, Vol. 13, pp. 23-35, 1977

[7] Brewer, J.W., "Kronecker Products and Matrix Caculus in System Theory" in *IEEE Trans. Curcuits and Systems*, Vol CAS-25, No. 9, Sept 1978, pp. 772-781

[8] Böhme, J.F., "Array Processing" in *Advances in Spectrum Analysis and Array Processing*, Vol. 2, S. Haykin, Ed., Englewood Cliffs, NJ:Pretice-Hall, 1991, ch 1, pp 1-63

[9] Botto, J. and Moustakides, G.V., "Stabilizing the Fast Kalman Algorithms", *IEEE Trans. ASSP*, Vol. 37, No. 9, pp. 1342 - 1348, Sept. 1989

[10] Bottomley, G.E. and Alexander S.T., "A Theoretical Basis for the Divergence of Conventional Recursive Least Squares Filters," *IEE International Conference on Acoustics, Speech, Signal Processing* (ICASSP'89), pp. 908-911, Glasgow, Scotland, 1989

[11] Bottomley, G.E. and Alexander S.T., "A Novel Approach for Stabilizing Recursive Least Squares Filters," *IEEE Trans. SP*, Vol. 39, No. 8, pp. 1770-1779, Aug. 1991

[12] Caraiscos, C. and Liu, B., "A Roundoff Analysis of the LMS Adaptivee Algorithm," *IEEE Trans. ASSP*, vol. ASSP-32, no. 1, pp. 34 - 41, Feb. 1984

[13] Chansarkar, M.M. and Desai, U. B., "A Robust Recursive Least Squares Algorithm", *IEEE Trans SP,* VOL. 45, NO. 7, July 1997

[14] Cioffi, J.M., "Limited-Precision Effects in Adaptive Filtering," *IEEE Trans. Circuits and Systems,* Vol. CAS-34, No. 7, pp. 821-833, July 1987

[15] Datta, B.N., *Numerical Linear Algebra And Applications,* Pacific Grove: Brooks/Cole Pub., 1995, ch3, pp 67-88.

[16] Fabre, P. and Gueguen, C., "Improvement of the Fast Recursive Least-Squares Algorithms Via Normalization: A Comparative Study", *IEEE Trans. ASSP,* VOL. ASSP-34, No. 2, pp 296-308, 1986

[17] Frost III, O.L., "An Algorithm for linearly Constrained Adaptive Array Processing," *Proc. IEEE,* Vol. 60, No. 8, pp. 926-935, Aug. 1972

[18] Gentleman, W.M., "Error Analysis of QR Decomposition by Givens Transformations," in *Linear Algebra and Its Applications,* 10, 1975, pp 189-197

[19] Gentleman, W.M. and Kung, H.T., "Matrix Triangularization by systolic arrays," in *Proc. SPIE,* vol. 298, Real Time Signal Processing IV, 1981, pp 298-303.

[20] Griffiths, L.J. and Jim, C.W., "An alternative aproach to linearly constrained adaptive beamforming", *IEEE Trans. AP*-30, 1982, pp 27-34

[21] Griffiths, L.J., "Linearly Constrained Adaptive Signal Processing methods," in SPIE Vol 826, *Adaptive Algorithms and Architectures for Signal Processing II,* Aug. 1987, pp 96-101

[22] Haykin, S., *Modern Filters,* New York: Macmillan Publishing, 1989

[23] Haykin, S., *Adaptive Filter Theory,* 2nd ed., Englewood Cliffs, NJ:Prentice Hall, 1991

[24] Haykin, S., *Adaptive Filter Theory,* 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1996, ch 17, pp 738-769

[25] Hudson, J.E., *Adaptive Array Principles,* London: Peter Peregrinus Ltd., 1989

[26] Huo, J. and Leung, Y.H., "Numerical Properties of the Linearly Constrained QRD-RLS Adaptive Filter," *Proc. ICASSP'98,* May 12-15, 1998, Seattle, WA, pp. 1685-1688

[27] Hsu, F.M., "Square Root Kalman Filtering for High-Speed Data Received over Fading Dispersive HF Channels", *IEEE Trans. IT*, Vol. IT-28, No. 5, pp. 753 - 763, Sept. 1982

[28] Jablon, N.K., "Steady State Analysis of the Generalized Sidelobe Canceller by Adaptive Noise Cancelling Techniques", *IEEE Trans. AP*, VOL. AP-34, NO. 3, MARCH 1986, pp. 330-337

[29] Jackson, L.B., *Digital Filters and Signal Processing*, Boston: Kluwer Academic Publishers, 1996

[30] Kosut, R.L., Anderson, B.O. and Mareels, I.M.Y., "Stability Theory for Adaptive Systems: Method of Averaging and Persistency of Excitation", *IEEE Trans. Automatic Control*, Vol. AC-32, No. 1, pp. 26-34, Jan., 1987

[31] Kushner, R.L., *Approximation and Weak Convergence Methods for Ramdom Processes, with Applications to Stochastic Systems Theory*, Cambridge, MA: MIT Press, 1984

[32] Lam, J. and Yan, W., "Summary on Matrix Properties", Dept Systems Engineering, Research School of Physical Sciences and Engineering, Australian National University

[33] Lawson, C.L. and Hanson, R.J., Solving Least Squares Problems, Englewood Cliffs, NJ: Prentice-Hall, 1974

[34] Leung, H. and Haykin, S., "Stability of Recursive QRD-LS Algorithms Using Finite-Precision Systolic Array Implementation," *IEEE Trans ASSP*, Vol. 37, No. 5, pp.760-763, May 1989

[35] Liavas, A.P. and Regalia, P.A., "Numerical Stability Issues of the Conventional Recursive Least Squares Algorithm", *Proc. ICASSP'98*, pp. 1409-1412, May 12-15, 1998, Seattle, Washington, USA

[36] Liavas, A.P. and Regalia, P.A., "On the Numerical Stability and Accuracy of the Conventional Recursive Least Algorithm," accepted for publication, *IEEE trans. ASSP*

[37] Lin, D.W., On Digital Implementation of the Fast Kalman Algorithms, *IEEE Trans. ASSP*, Vol. ASSP-32, No.5, pp. 998 - 1005, Oct. 1984

[38] Ljung, S. and Ljung, L., "Error Propagation Properties of Recursive Least-Squares Adaptation Algorithms," *Automatica*, Vol. 21, No. 2, pp. 157-167, 1985

[39] McWhirter, J.G., "Recursive Least-Squares Minimization Using a Systolic Array," *Proc. SPIE, Real-time Signal Processing VI*, vol 431, San Diego, Calif, 1983

[40] McWhirter, J.G. and Shepherd, T.J., "A Systolic Array for Linearly Constrained Least-Squares Problems," SPIE Vol. 696, *Advanced Algorithms and Architectures for Signal Processing*, pp. 80-87, 1986

[41] McWhirter, J.G., "Systolic Array Processor for MVDR Beamforming," *IEE Proc.*, Vol. 136, Pt. F, No. 2, Apr. 1989

[42] Nitzberg, R., *Adaptive Signal Processing for Radar*, Norwood, MA: Artech House, 1992, ch 6, pp 87-106

[43] Resende, L.S., Romano, J.M.T. and Bellanger, M.G., "A Robust FLS Algorithm for Linearly-Constrained Adaptive Filtering," *International Conference on Acoustics, Speech, Signal Processing (ICASSP'94)*, pp. III 381 - III 384, Adelaide, Australia, April 1994

[44] Resende, L.S., Romano, J.M.T. and Bellanger, M.G., "Fast Least-Squares Algorithm for Linearly Constrained Adaptive Filtering," *IEEE Trans. SP*, Vol. 44, No. 5, pp. 1168-1174, May 1996

[45] Regalia, P.A., "System Theoretic Properties in the Stability Analysis of QR-Based Fast Least-Squares Algorithms," Institut National Des Telecommunications internal report, DEC-0390-003, Mar 1990

[46] Regalia, P.A., "Numerical Stability Issues in Fast Least-Squares Adaptation Algorithms," *Opt. Eng.* Vol. 31, No. 6, pp. 1144-1152, June 1992

[47] Regalia, P.A., "Numerical Stability Properties of a QR-Based Fast Least Squares Algorithm," *IEEE Trans. SP*, Vol. 40, No. 6, June 1993

[48] Samson, C.G. and Reddy, V.U., "Fixed Point Error Analysis of the Nomalized Ladder Algorithm," *IEEE Trans. ASSP*, vol. ASSP-35, no. 5, pp . 1177-1191, Oct. 1983

[49] Schreiber, R., "Implementation for Adaptive Array Algorithms", *IEEE Trans. ASSP*-34, pp 1038-1045, 1986

[50] Shepherd, T.J. and McWhirter J.G., "Systolic adaptive Beamforming," in *Radar Array Processing*, S. Haykin, Ed., Springer-Verlag Berlin, Heidelberg, 1993, ch. 5, pp. 153-243

[51] Slock, D.T.M. and Kailath, T., "Numerically Stable Fast Transversal Filters for Recursive Least Squares Adaptive Filtering," *IEEE Trans. SP*, Vol. 39, No. 1, Jan. 1991

[52] Slock, D.T.M., "Backward Consistency Concept and Round-off Error Propagation Dynamics in Recursive Least-Squares Algorithms," *Opt. Eng.* Vol. 31, No. 6, pp. 1153-1169, June 1992

[53] Stewart, G.W., "Error Analysis of QR Updating with Exponential Windowing," in *Math. Comput.* vol. 59, no. 199, Jul. 1992, pp 135-140

[54] Theodoridis, S. and Kalouptsidis, N., "Spectral Analysis," in *Adaptive System Identification And Signal Processing Algorithms*, N. Kalouptsidis and S. Theodoridis, Ed., Prentice Hall International (UK) Limited, 1993, ch. 8, pp. 322-387

[55] Van Veen, B.D. and Buckley, K.M., "Beamforming: A Versatile Approach to Spatial Filtering," *IEEE ASSP Mag.*, pp. 4-24, April 1986

[56] Verhaegen, M.H., "A New Class of Algorithms in Linear System Theory, with Application to Real-Time Aircraft Model Identification," Ph.D. dissertation, Catholic Univ. Leuven, leuven, Belgium, Nov. 1985

[57] Verhaegen, M.H. and Van Dooren, P., "Numerical Aspects of Different Kalman Filter Implementations," *IEEE Trans Automatic Control*, Vol. AC-31, No. 10, pp. 907-917, Oct. 1986

[58] Verhaegen, M.H., "Improved Understanding of the Loss-of-Symmetry Phenomenon in the Conventional Kalman Filter," *IEEE Trans. Automatic Control*, Vol. 34, No. 3, pp. 331-333, Mar. 1989

[59] Verhaegen, M.H., "Round-off Error Propagation in Four Generally-applicable, Recursive, Least-squares Estimation Schemes," *Automatica*, Vol. 25, No. 3, pp. 437-444, 1989

[60] Vidyasagar, M., *Nonlinear System Analysis,* 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1993, ch 5, pp 135-269

[61] Weiss, A. and Mitra, D., "Digital Adaptive Filters: Conditions for Convergence, Rates of Convergence, Effects of Noise and Errors Arising from the Implementation," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 6, pp. 637 - 652

[62] Wilkinson, J.H., *The Algebraic Eigenvalue Problem*, Oxford: Clarendon Press, 1965

[63] Wilkinson, J.H., Rounding Errors in Algebra Processes, Englewood Cliffs, NJ: Prentice-Hall, 1963

[64] Yang, B. and Böhme, J.F., "A New Orthogonal Adaptive Algorithm and Its Systolic Implementation for the RLS Problem without a Desired Signal," in *SIGNAL PROCESSING V: Theories and Applications*, I. Torres, E. Masgrau and M. A. Lagunas Eds., Elsevier Science, B. V., 1990, pp 677-680

[65] Yang, B. and Böhme, J.F., "Linear Systolic Arrays for Constrained Least Squares Problems," in *Mathematics in Signal Processing*, Clarendon Press, Oxford, 1990, pp 689-712

[66] Yang, B. and Böhme, J. F., "A Linear Systolic Array for the Adaptive MVDR Beamformer," in *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, G. H. Golub and P. Van Dooren, Eds. Springer-Verlag, Berlin, 1991, pp 705-712

[67] Yang, B., "A Note on the Error Propagation Analysis of Recursive Least Squares Algorithms", *IEEE Trans. SP*, Vol. 42, No. 12, pp. 3523 - 3525, Dec. 1994

Appendix A: Kronecker Products

The error propagation analysis of adaptive algorithms relies on the study of the transition matrix of the error system. The Kronecker product plays an important role in the derivation of the transition matrix. In this appendix, we present the definition and some basic theorems regarding the Kronecker products.

Definition

Let $\mathbf{A}$ be a $p \times q$ matrix and $\mathbf{B}$ be an $m \times n$ matrix. The Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ denoted by $\mathbf{A} \otimes \mathbf{B}$ is a $pm \times qn$ matrix defined as [7]

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1q}\mathbf{B} \\ \cdots & \cdots & \cdots \\ a_{p1}\mathbf{B} & \cdots & a_{pq}\mathbf{B} \end{bmatrix}. \tag{A.1}$$

Some Useful Results

Some useful theorems of Kronecker products are listed below [32].

1. $(\mathbf{A} \otimes \mathbf{B})^{\mathrm{H}} = \mathbf{A}^{\mathrm{H}} \otimes \mathbf{B}^{\mathrm{H}}$;

2. $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$;

3. $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$;

4. $tr(\mathbf{A} \otimes \mathbf{B}) = tr(\mathbf{A})tr(\mathbf{B})$;

5. $rank(\mathbf{A} \otimes \mathbf{B}) = rank(\mathbf{A})\text{rank}(\mathbf{B})$;

6. $det(\mathbf{A} \otimes \mathbf{B}) = rank(\mathbf{A})^{m}\text{rank}(\mathbf{B})^{n}$

where $\mathbf{A}$ is an $n \times n$ matrix and $\mathbf{B}$ is an $m \times m$;

7. There exists a matrix $\mathbf{F}$ such that

$$\mathbf{A} \otimes \mathbf{B} = \mathbf{F}^{-1}(\mathbf{B} \otimes \mathbf{A})\mathbf{F};$$

8. $cs(\mathbf{AXB}) = (\mathbf{B}^{T} \otimes \mathbf{A})cs(\mathbf{X})$;

9. Any eigenvalue of $\mathbf{A} \otimes \mathbf{B}$ is a product of an eigenvalue of $\mathbf{A}$ and an eigenvalue of $\mathbf{B}$; while any eigenvalue of $\mathbf{A} \otimes \mathbf{I}_{m} + \mathbf{I}_{n} \otimes \mathbf{B}$ is the sum of an eigenvalue of $\mathbf{A}$ and an eigenvalue of $\mathbf{B}$.

Appendix B: Average Analysis of (3.26) and (4.13)

In the analysis of the CRLS algorithm, it is shown that the error propagation model for versions II and II is given as

$$\begin{aligned} cs(\delta(n)) &= \beta\{[\mathbf{P}(n)\Phi(n-1)]^* \otimes [\mathbf{P}(n)\Phi(n-1)]\}cs(\delta(n-1)) \\ &= \beta[\mathbf{P}^*(n) \otimes \mathbf{P}(n)][\Phi^*(n-1) \otimes \Phi(n-1)]cs(\delta(n-1)) \end{aligned} \quad .(B.1)$$

And in Section 4.1.1, we derived the error propagation model for the LC-FLS algorithm as

$$\begin{aligned} cs(\delta(n)) &= \{[\Theta(n)\psi(n-1)]^* \otimes [\mathbf{P}(n)\Phi(n-1)]\}cs(\delta(n-1)) \\ &= [\Theta^*(n)\mathbf{P}(n)] \otimes [\psi^*(n-1)\Phi(n-1)]cs(\delta(n-1)) \end{aligned} \quad . \quad (B.2)$$

As mentioned in Section 3.1.1, we study the average of the above two systems to obtain insight into the performance of the algorithms. To do this, we just evaluate expectation of the transition matrices

$$\mathrm{E}(\beta[\mathbf{P}^*(n) \otimes \mathbf{P}(n)][\Phi^*(n-1) \otimes \Phi(n-1)]) \quad (B.3)$$

and

$$\mathrm{E}([\Theta^*(n)\mathbf{P}(n)] \otimes [\psi^*(n-1)\Phi(n-1)]). \quad (B.4)$$

Both (B.3) and (B.4) involve the evaluation of an expectation of the form

$$\mathrm{E}([\mathbf{A}(n) \otimes \mathbf{B}(n)][\mathbf{C}(n-1) \otimes \mathbf{D}(n-1)]) \quad (B.5)$$

where the matrices have the following properties:

- All matrices are Wishart distributed [24] (from the assumption of i.i.d. input);
- The matrices at iteration $n-1$ is independent of those at n (from the independent assumption);
- The matrix $\mathbf{A}(n)$ is uniquely defined by $\mathbf{B}(n)$, and $\mathbf{C}(n-1)$ is uniquely defined by $\mathbf{D}(n-1)$;
- $\mathbf{A}(n) = \mathbf{C}^{-1}(n), \mathbf{B}(n) = \mathbf{D}^{-1}(n)$.

Employing the second property, (B.5) can be written as

$$\mathrm{E}([\mathbf{A}(n) \otimes \mathbf{B}(n)])\mathrm{E}([\mathbf{C}(n-1) \otimes \mathbf{D}(n-1)]). \tag{B.6}$$

Using the fourth property, (B.6) becomes

$$\mathrm{E}([\mathbf{A}(n) \otimes \mathbf{B}(n)])\mathrm{E}([\mathbf{A}(n-1) \otimes \mathbf{B}(n-1)]^{-1}). \tag{B.7}$$

By the first and the third properties, it is easy to show that the matrix $\mathbf{A}(n) \otimes \mathbf{B}(n)$ is Wishart distributed. Therefore, for large n, (B.5) converges to an identity matrix

$$\mathrm{E}([\mathbf{A}(n) \otimes \mathbf{B}(n)][\mathbf{C}(n-1) \otimes \mathbf{D}(n-1)]) = \mathbf{I}. \tag{B.8}$$

Substituting (B.8) into (B.1) and (B.2) gives (3.28) and (4.14).

Appendix C: Derivation of Table 4.4

Consider the trapezoid systolic array below in its adaptive mode with a forgetting factor $\beta = 1$.

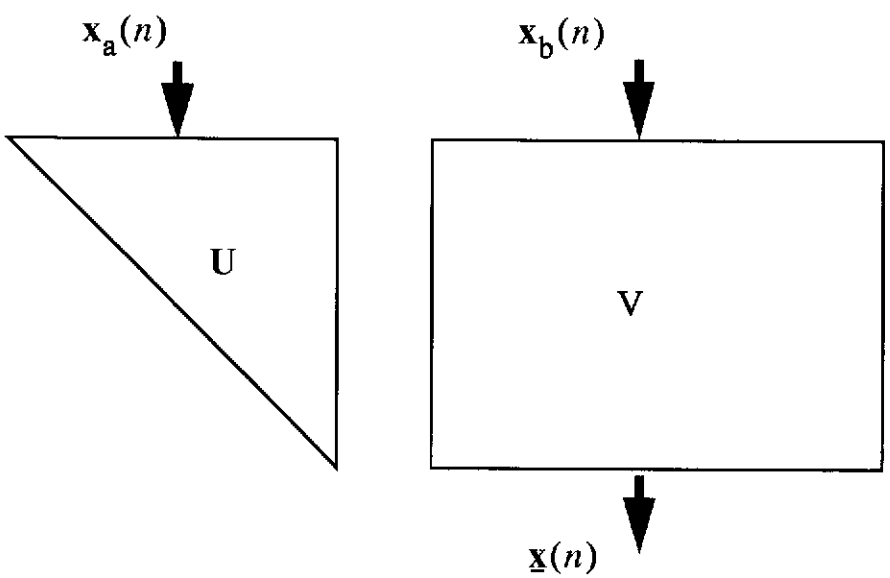


Figure C.1: Trapezoid Adaptive Systolic Array

At the bottom of the ith column of the retangular part, the output is given by (see a priori residual extraction in [50])

$$\underline{x}_i(n) = x_{b,\,i}(n) - \mathbf{v}_i^H \mathbf{U}^{-H} \mathbf{x}_a(n). \tag{A.1}$$

Therefore, the output of the whole array is

$$\underline{\mathbf{x}}(n) = \mathbf{x}_b(n) - \mathbf{V}^H \mathbf{U}^{-H} \mathbf{x}_a(n). \tag{C.1}$$

Compare (A.1), (C.1) with (2.40) and (2.41), it is seen that at every iteration, the pre-filtering operation can be implemented by passing the input signal through an adaptive systolic array which stores the trapezoid matrix $\begin{bmatrix} \mathbf{U} & \mathbf{V} & \mathbf{m} \end{bmatrix}$ where $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{m}$ are defined in Section 2.4.1. Because the stored values of the systolic array are destroyed by the unitary rotations, they have to be restored at the next iteration. A more efficient way is to suppress the update of the content of the systolic array during the rotation, which is exactly the one given in Table 4.4. Note that this method can also be applied to the series weight flushing. Moreover, the rotation parameters can be computed in a square-root free manner , which greatly reduces the computational load.

Appendix D: Geometrical Interpretation of the LC-QRD-RLS Algorithm

Using the definitions in Figure 2.1, we can write the weight vector at iteration n as the sum of two orthogonal components as follows

$$\begin{aligned} \mathbf{w}(n) &= \mathbf{B}\mathbf{w}_b(n) + \mathbf{w}_c \\ &= \begin{bmatrix} -\mathbf{U}^{-1}\mathbf{V} \\ \mathbf{I} \end{bmatrix} \mathbf{w}_b(n) + \begin{bmatrix} \mathbf{U}^{-1}\mathbf{m}' \\ \mathbf{0} \end{bmatrix} \\ &= \left(\begin{bmatrix} -\mathbf{U}^{-1}\mathbf{V} \\ \mathbf{0} \end{bmatrix} \mathbf{w}_b(n) + \begin{bmatrix} \mathbf{U}^{-1}\mathbf{m}' \\ \mathbf{0} \end{bmatrix} \right) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{w}_b(n) \\ &= (\mathbf{A}\mathbf{w}_b(n) + \mathbf{w}_c) + \mathbf{w}'_b(n) \end{aligned} \tag{D.1}$$

where $\mathbf{w}_b(n)$ is the unconstrained weight vector implied by the canonical section.

When finite precision arithmetic is used, rounding errors are inevitably introduced. Firstly, there are rounding errors in the stored values of the pre-computed frozen network. These introduce a fixed perturbation in the matrix A and the vector $\mathbf{w}_c$, and can be interpreted as a perturbation of the imposed constraints. Secondly, the preprocessing of the input data in the frozen network to compute $\hat{\mathbf{u}}(n)$ and $\hat{y}(n)$ will introduce additional rounding errors which can be viewed as the result of applying a perturbation on the input data. Thus the weight vector derived from the systolic array implementation of the LCQRD-RLS algorithm corresponds to the solution of problem P.3 with perturbed constraint matrix, perturbed

constraint vector and perturbed input data. The important point to note here is that the weight vector always satisfies the perturbed constraints exactly.

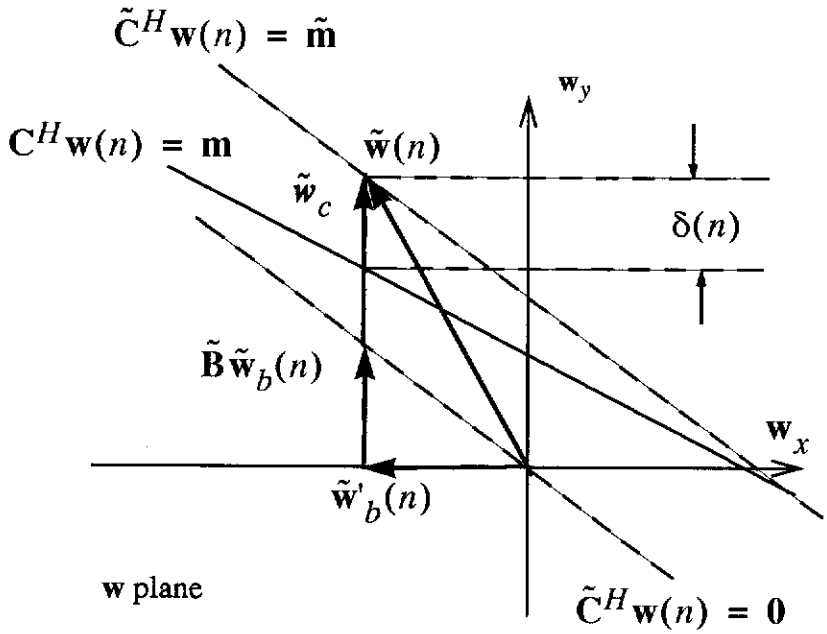


Figure D.1: Geometrical Interpretation

Equation D.1 and the above discussions can be visualized in Figure D.1. $\tilde{x}$ denotes the computed quantity of x. The axes $\mathbf{w}_x$ represents the second term in equation (2) while $\mathbf{w}_y$ represents the first term. $\mathbf{C}^H\mathbf{w}(n) = \mathbf{m}$ are the original constraints. However, due to rounding errors, the constraints that the computed weight vector actually satisfies are given by $\tilde{\mathbf{C}}^H\mathbf{w}(n) = \tilde{\mathbf{m}}$. This results in the computed weight vector $\tilde{\mathbf{w}}(n)$ deviating from the original constraints by $\delta(n)$.

From Figure D.1 we see that the deviation of the computed weight vector from the imposed constraints depends on how close the perturbed constraints are to the original ones, and how large is the vector $\tilde{\mathbf{w}}_b(n)$. To keep the perturbed constraints close to the imposed ones, we can simply use more bits in the frozen network. For $d(n)$ to be bounded, we require $\tilde{\mathbf{w}}_b(n)$ to be bounded which is the case if the input data $\mathbf{u}(n)$ do not approach zero asymptotically [34].