UNLIMITED

RSRE
MEMORANDUM No. 4409

ROYAL SIGNALS & RADAR ESTABLISHMENT

THE QR-DECOMPOSITION BASED LEAST-SQUARES LATTICE ALGORITHM FOR ADAPTIVE FILTERING

Authors: I K Proudler, J G McWhirter

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

RSRE MEMORANDUM No. 4409

UNLIMITED

CONDITIONS OF RELEASE

0083098 BR-115234

************************** DRIC U

************************** DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 4409

TITLE: THE QR DECOMPOSITION BASED LEAST-SQUARES LATTICE ALGORITHM FOR ADAPTIVE FILTERING

AUTHORS: I K Proudler and J G McWhirter.

DATE: July 1990

SUMMARY

We derive, from first principles, the least squares lattice algorithm for adaptive filtering based on the QR decomposition (QRD). In common with other lattice algorithms for adaptive filtering, this algorithm only requires $O(p)$ operations for the solution of a p-th order problem. The algorithm has as its root the QRD-based recursive least squares minimisation algorithm and hence is expected to have superior numerical properties when compared with other fast algorithms.

This algorithm contains within it the QRD-based algorithm for solving the least squares linear prediction problem. These algorithms are presented in two forms: one that involves taking square-roots and one that does not. Some computer simulations of a channel equaliser, using finite-precision arithmetic, are presented in which the lattice algorithms are compared to the more established triangular systolic array ones.

The relationship between the QRD-based lattice algorithm and other least squares lattice algorithms is briefly discussed. Various extensions to this work are discussed including the multi-channel QRD-based adaptive filtering algorithm that can be used for wide-band beamforming.

THIS PAGE IS LEFT BLANK INTENTIONALLY

CONTENTS

1. INTRODUCTION 1
2. NOTATION 4
3. ADAPTIVE FILTERING 5
4. LINEAR PREDICTION 10
4.1. Motivation 10
4.2. Forward Linear Prediction 11
4.3. Backward Linear Prediction 14
5. IMPLEMENTATION 19
6. MULTI-CHANNEL CASE 22
6.1. Problem Definition 22
6.2. Forward Linear Prediction 23
6.3. Backward Linear Prediction 25
7. SIMULATIONS 29
8. CONCLUSION 34
9. REFERENCES 35
10. APPENDIX 37
10.1. Joint Process Estimation 37
10.2 Givens Rotations 40
10.3. Algorithms 43
10.3.1. "Normal" algorithm 43
10.3.2. "Square-root-free with feedback" algorithm 44
10.4 Post-processor Architectures 45
10.4.1 Parallel Weight Extraction 45
10.4.2 Linear Constraints 47
11. ANNEX 50

Tables

Table 1. Eigenvalue Spread 30

Figures

Figure 1 Relationship to Covariance Domain Lattice Algorithm and RMGS 2
Figure 2 QRD-Based Lattice Algorithm 17
Figure 3 QRD-Based Fast Kalman Algorithm 17
Figure 4 "Delayed" Adaptive Filtering Lattice 21
Figure 5 "Normal" Rotation PE's 21
Figure 6 "Square-root-free with Feedback" Rotation PE's 21

Figure 7 Multi-channel Adaptive Filter 22
Figure 8. Lattice of Triangular Arrays. 27
Figure 9. Triangular Systolic Array. 27
Figure 10 Channel Equaliser Experiment. 29
Figure 11 Normal QRD Lattice: Effect of Eigenvalue Spread. 31
Figure 12 Normal QRD Lattice: Effect of Wordlength. 31
Figure 13 Comparison of Lattice and Array. 32
Figure 14 Comparison of Givens Algorithms. 32
Figure 15 "True" Adaptive Filtering Lattice 39
Figure 16 Parallel Weight Extraction 45
Figure 17 Parallel Weight Extraction from a Lattice 46
Figure 18 Linearly Constrained Least Squares Minimisation 47
Figure 19 MVDR Beamforming 48
Figure 20 Constraint Post-processor 49

1. INTRODUCTION

Least squares minimisation can be applied to a wide range of digital signal processing problems including that of adaptive filtering. The adaptive filtering problem is, however, special in the sense that the "data matrix" ($X_p(n)$ of equation (5)) has a Toeplitz structure so that each row of the data matrix contains only one new datum when compared to the previous row. Various algorithms [8] have been devised that take advantage of this redundancy to reduce the computational load, for a p-th order filter, from $O(p^2)$ to $O(p)$. Unfortunately these fast algorithms are not *explicitly* well-conditioned and some tend to be numerically unstable.

It is possible, however, to solve a least squares minimisation problem using the technique of QR decomposition[8]. This technique has the advantage that it operates on the data matrix directly, rather than the corresponding covariance matrix, and involves only orthogonal rotations, which are numerically well-conditioned. The recursive QRD algorithm can be implemented on the triangular systolic array as devised by Gentleman and Kung[5] and subsequently modified by McWhirter[14]. This algorithm solves a general recursive least squares minimisation problem and will require $O(p^2)$ operations to generate the solution to a p-th order adaptive filter problem. Extensive computer simulations of this more general algorithm[28] have shown the QRD-based least squares minimisation algorithm to have excellent numerical properties. The possibility of producing an $O(p)$ QRD-based algorithm for the special case of adaptive filtering has thus long been of interest.

Here we present a full derivation of the recently derived QRD-based least squares lattice algorithm[16] starting from the $O(p^2)$ QRD-based algorithm and incorporating directly the time-shift redundancy found in an adaptive filtering problem. The derivation presented here owes much to the work of Cioffi[3]. Recently he showed how to take advantage of the time-shift redundancy that is present in the problem of the linear prediction of time series data and produced a QRD-based "fast Kalman" algorithm (see [1] or [16] for alternative, simpler derivations). This algorithm can recursively update, from one time instant to the next, the solution to a p-th order linear prediction problem in $O(p)$ operations. However, unlike other fast Kalman algorithms, the QRD-based fast Kalman algorithm also produces the solution to all lower order problems as well (see [19]).

The QRD-based lattice algorithm, like all lattice algorithms, is designed to solve the linear prediction problem recursively in time *and* order, again in $O(p)$ operations: thus it, too, solves all lower order problems. The two classes of fast QRD-based algorithms *are* different however. The fast Kalman algorithms are completely different in structure from the QRD-based lattice algorithms (see Slock [24] for more discussion). In particular the fast Kalman algorithms have a smaller operation count than the lattice algorithms (although both are linear in the problem order). The level at which the two different algorithms can be pipelined is also different - the lattice algorithms having a higher degree of concurrency. It is also worth noting that the fast Kalman algorithms implicitly have a downdating step which gives cause for concern, from a numerical point of view, although no problems have been observed in any simulations done to date.

In recent months, the $O(p)$ QRD-based least squares lattice algorithm has also been derived, independently, by two other groups: Ling[13] and Yang and Böhme[29]. Both of these derivations begin from a previously known (non-orthogonal[1]) algorithm and by a series of transformations arrive at one

with only orthogonal operations. As such these derivations provide an interesting insight into the problem and are complementary to that presented here.

Yang and Böhme bring together the work of Lewis[9] and McWhirter[14]. Lewis began with the standard (covariance domain) multi-channel least squares lattice equations and, driven firstly by the desire not to explicitly invert the covariance matrices and secondly to perform all matrix computations in a "numerically sound" way, reformulated part of the algorithm (the calculation of the reflection coefficients) by the use of the matrix inversion lemma and QR decomposition. As the bulk of the calculation is exactly the computation of the reflection coefficients, Lewis proceeded no further with this re-formulation and apparently failed to notice that the "non-orthogonal" part of his algorithm is in fact redundant. Following the work of McWhirter, Yang and Böhme noticed that the adaptive filtering residuals can be found directly: this observation results in the construction of a purely orthogonal algorithm (see Figure 1).

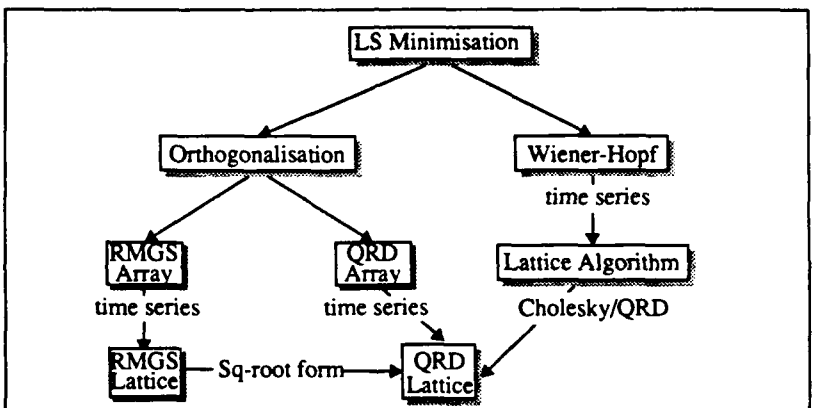


Figure 1 Relationship to Covariance Domain Lattice Algorithm and RMGS

The other derivation, by Ling, relies on a well known equivalence[10] between two general least squares minimisation algorithms: the $O(p^2)$ QRD-based algorithm and the recursive modified Gram-Schmidt (RMGS) algorithm[11]. Strictly speaking, the equivalence is between a modification of the RMGS algorithm and a form of the QRD-based algorithm that does not require the square-root operation[2]. Ling and Proakis[12] have shown how to create an $O(p)$ lattice algorithm for solving the adaptive filtering problem by taking advantage of the time-shift redundancy within the RMGS approach: the so called "error feedback" lattice algorithm. This is arguably the most well-behaved of the least squares lattice algorithms presented to date. Using the above equivalence, Ling [13] has recently shown that the error feedback lattice algorithm can be reinterpreted as being a square-root-free form of an associated algorithm that consists only of orthogonal rotations. Comparison with the algorithm derived here shows that Ling's associated orthogonal algorithm and that of Yang and Böhme, when specialised to the single

1. We use the phrase "orthogonal algorithm" to mean one that generates the required solution exclusively by the application of orthogonal transformations, such as Givens rotations, to the input data.
2. See section 5 for details of the square-root-free version of the QRD-based algorithm.

channel case, are indeed identical and are equivalent to the QRD-based least squares lattice algorithm.

Another approach to fast orthogonal adaptive filtering algorithms has been presented by Regalia and Bellanger [20]. Based on the work of Cioffi[3], Regalia and Bellanger realised that certain quantities in the QRD-based fast Kalman algorithm were the same as those in a conventional (covariance domain) lattice algorithm. In particular, the identification of the reflection coefficients as the sines of certain rotation angles led them to develop an alternative, Kalman-type algorithm for solving the linear prediction problem. Regalia has shown theoretically [21] that his structure is stable. However it is not as yet clear whether the same analysis will work for the other fast QRD-based adaptive filter algorithms.

As well as presenting the QRD-based least-squares lattice algorithm for linear prediction, the extension of this technique to the solution of the adaptive filtering problem is also given in this memorandum. The resulting algorithm has a lattice-ladder structure. In common with Cioffi's original formulation of the QRD-based fast Kalman algorithm, the lattice-ladder algorithm presented here operates on pre-windowed data (i.e. all data before the first time instant is assumed to be zero). The extension of this work to the multi-channel case (wide-band beamforming) is relatively straightforward and is briefly discussed in section 6.

We begin, in section 3, by reviewing the mathematics of the solution to an adaptive filtering problem using the method of QR decomposition. The key to the fast adaptive filtering algorithm is the development of a fast (lattice) algorithm for the solution of an associated linear prediction problem. In section 4 the connection between these two, related, problems is outlined and the $O(p)$ solution to the linear prediction problem, and hence the adaptive filtering problem, is developed. Section 5 discusses various aspects involved in the implementation of the lattice algorithm. The derivation of the multi-channel lattice algorithm is sketched out in section 6 and the results of some computer simulations are presented in section 7. The algorithm is given explicitly, in the appendix, in two forms: one that involves the square-root operation and one that does not.

2. NOTATION

$\|\bullet\|$	L_2 norm.
$\underline{0}$	a vector of zeros[3].
O	a matrix of zeros.
$\underline{\pi}_n$	an n-dimensional vector full of zeros except for the n-th component which is unity (pinning vector).
Q	an orthogonal rotation matrix.
$\hat{Q}$	time recursive increment of Q. E.g. $Q(n) = \hat{Q}(n) \begin{bmatrix} Q(n-1) & \underline{0} \\ \underline{0}^t & 1 \end{bmatrix}$.
R	an upper triangular matrix.
X	a data matrix.
$\underline{y}$	a reference vector.
$\underline{u}, \underline{v}$	the two components of the rotated reference vector: $Q\underline{y}$.
$\underline{a}_p, \underline{g}_p$	the two components of the right-hand column of the matrix Q_p. Alternatively, the two components of the rotated pinning vector: $Q\underline{\pi}$
e	a least squares residual.
α	a rotated, or angle-normalised residual.
β	the exponential weighting factor.
γ_p	*the lower right-hand* element of the matrix Q_p.
ϵ	the sum of the squared prediction errors.
Subscript "f"	indicates a quantity connected with a forward linear prediction problem or its solution.
Subscript "b"	indicates a quantity connected with a backward linear prediction problem or its solution.
Subscript "p"	*indicates a quantity connected with a p-th order problem or its solution* (similarly subscripts "p-1", "p+1" etc.).

Hence, for example:

$u_{b,p-1}(n-1)$	is the top component of the rotated reference vector $\underline{v}_{b,p-1}(n-1)$ associated with the (p-1)st order backward linear prediction problem at time (n-1).

Several rotation matrices are introduced in this memorandum that, clearly, have to be distinguished from each other. There is, however, no obvious choice as to how they should be labelled. Following the convention used for labelling of the reflection coefficients in the covariance-domain least squares lattice algorithm, we choose to label these rotation matrices according to the problem to which they relate. For example the matrix $Q_{f,p}(n)$ is a rotation matrix used in the calculation of the solution to the p-th order forward linear prediction problem at time n: despite the fact that it operates on the vector $\underline{v}_{b,p-1}(n-1)$!

3. The dimensionality of the all zero vectors and matrices used in this memorandum should be obvious from the context in which they are used.

3. ADAPTIVE FILTERING

In order to simplify the analysis we consider only real signals. The extension to the complex case is straight forward and indeed the algorithms presented in the appendix are for complex signals.

In a least squares adaptive filtering problem, a set of p (say) weights, $\underline{\omega}_p(n) = [\omega_p^{(0)}(n), \omega_p^{(1)}(n), \ldots, \omega_p^{(p-1)}(n)]^t$, is to be found, at time n, that minimises that the sum of the differences, squared, between a reference signal y(n) and a linear combination of p samples from a data time series x(n-i) ($0 \le i \le p-1$). This decision is to be based on all the data accumulated so far. Specifically, the measure $E_p(n)$ is to be minimised, where:

$$E_p(n) = \|B(n)\,\underline{e}_p(n)\| \tag{1}$$

$$B(n) = \begin{bmatrix} \beta^{n-1} & 0 & \ldots & 0 \\ 0 & \beta^{n-2} & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \tag{2}$$

$$0<\beta\le 1 \tag{3}$$

$$\underline{e}_p(n) = X_p(n)\,\underline{\omega}_p(n) + \underline{y}(n) \tag{4}$$

$$X_p(n) = \begin{bmatrix} x(1) & x(0) & \ldots & x(2-p) \\ x(2) & x(1) & \ldots & x(3-p) \\ \vdots & \vdots & & \vdots \\ x(n) & x(n-1) & \ldots & x(n-p+1) \end{bmatrix} \tag{5}$$

$$\underline{y}(n) = [y(1), \ldots, y(n)]^t \tag{6}$$

The diagonal matrix B(n) represents an exponential "forgetting" factor that allows the algorithm to work with quasi-stationary signals. Note that there are effectively two time indices present in equation (4) because it is necessary to distinguish between the components of the error vector $\underline{e}_p(n)$ and the amount of data used to calculate the coefficients (and hence the time index for $\underline{\omega}_p(n)$). Specifically we have

$$\underline{e}_p(n) = [e_p(1, n), e_p(2, n), \ldots, e_p(n, n)]^t \tag{7}$$

where

$$e_p(m, n) = y(m) + \sum_{i=0}^{p-1} \omega_p^{(i)}(n)\,x(m-i) \tag{8}$$

is the error in estimating the datum y(m) as a linear combination of x(m-i) $0 \le i \le p-1$ when using the optimum coefficients calculated at time n. There are two quantities of particular interest: the a-posteriori error (equation (9)) which uses the most up to date estimate of y(n) and the a-priori error (equation (10)) which is the estimation error for y(n) based on the coefficients calculated at the previous time.

$$e_p(n, n) = y(n) + \sum_{i=0}^{p-1} \omega_p^{(i)}(n) x(n-i) \tag{9}$$

$$e_p(n, n-1) = y(n) + \sum_{i=0}^{p-1} \omega_p^{(i)}(n-1) x(n-i) \tag{10}$$

The solution of least squares minimisation problems via QR decomposition is now well established [8] and requires the determination of an orthogonal matrix $Q_p(n)$ that transforms the matrix $B(n)\,X_p(n)$ into an upper triangular form (as indicated by the shading). Let

$$Q_p(n)\, B(n)\, X_p(n) = \begin{bmatrix} R_p(n) \\ O \end{bmatrix} \tag{11}$$

so that, from equations (4) and (11),

$$Q_p(n)\, B(n)\, \underline{e}_p(n) = \begin{bmatrix} R_p(n) \\ O \end{bmatrix} \underline{\omega}(n) + \begin{bmatrix} \underline{u}_p(n) \\ \underline{v}_p(n) \end{bmatrix} \tag{12}$$

where

$$\begin{bmatrix} \underline{u}_p(n) \\ \underline{v}_p(n) \end{bmatrix} \equiv Q_p(n)\, B(n)\, \underline{y}(n) \tag{13}$$

Note that $\underline{u}_p(n)$ is defined to be a p-dimensional vector so that the partitioning in equation (12) is consistent. Due to the orthogonal nature of $Q_p(n)$ it is clear that

$$E_p(n) = \|B(n)\, \underline{e}_p(n)\| = \|Q_p(n)\, B(n)\, \underline{e}_p(n)\| = \left\| \begin{bmatrix} R_p(n) \\ O \end{bmatrix} \underline{\omega}(n) + \begin{bmatrix} \underline{u}_p(n) \\ \underline{v}_p(n) \end{bmatrix} \right\| \tag{14}$$

and by inspection it can be seen that the least squares solution is obtained when

$$R_p(n)\, \underline{\omega}(n) + \underline{u}_p(n) = \underline{0} \tag{15}$$

and that

$$\min_{\underline{\omega}(n)} \{ E_p(n) \} = \|\underline{v}_p(n)\| \tag{16}$$

In principle equation (15) can be solved by back substitution, since $R_p(n)$ is a triangular matrix, and the optimum coefficient vector $\underline{\omega}(n)$ found. It is often the case, and adaptive filtering is a prime example, that the a-posteriori residual (equation (9)) is explicitly required[4]. The reason for this is the fact that the a-posteriori residual represents that part of the reference signal uncorrelated with the data signals x(n-i) ($0 \le i \le p-1$) and as such is the "filtered" signal. Clearly the a-posteriori residual can be calculated, once $\underline{\omega}_p(n)$ is known, as the last component of $\underline{e}_p(n)$ (equation (7)). It has been shown [14], however, that the adaptive filtering residual can be obtained directly from the product of two quantities naturally present in the QRD-based algorithm:

$$e_p(n,n) = \gamma_p(n)\,\alpha_p(n) \tag{17}$$

where

$$\gamma_p(n) = \underline{\pi}_n^t\, Q_p(n)\, \underline{\pi}_n \tag{18}$$

$$\alpha_p(n) = \underline{\pi}_{n-p}^t\, \underline{v}_p(n) \tag{19}$$

and

$$\underline{\pi}_n = [0, \ldots, 0, 1]^t \tag{20}$$

is an n-dimensional vector. Vectors of the form shown on equation (20) consisting of zeros except for the final element, which is unity, play an important rôle in the following mathematics. These vectors are often referred to as "pinning" vectors. In equations (18) and (19), the pinning vectors merely serve to select the lower right-hand element of $Q_p(n)$ and the last element of $\underline{v}_p(n)$ respectively.

The ability to calculate the adaptive filtering residual directly, using equation (17), is an important result since although inverting the matrix $R_p(n)$ is relatively easy (it is triangular and therefore can be inverted by back-substitution) this process is potentially numerically unstable whereas the two quantities $\gamma_p(n)$ and $\alpha_p(n)$ are always well defined. It can be shown [13] that $\gamma_p(n)$ is the square-root of the maximum likelihood factor of more conventional algorithms. We follow Ling [13] in referring to $\alpha_p(n)$ as the "angle normalised" residual.

It is worth noting, at this point, that there are two methods for calculating the orthogonal matrix Q: via Givens (planar) rotations and via the Householder transformation (a reflection). The Householder technique leads to a lower operation count; however, unlike the Givens approach it does not admit a time recursive implementation, which is often preferred[5].

It can not be stressed too much that once $Q_p(n)$ has been found the problem has effectively been

4. Note that it is also possible to extract the coefficient vector from the QRD-based algorithm in a systolic fashion: see section 10.4.1.
5. Recently, Cioffi [4] and Slock [23] have published recursive QRD-based "fast Kalman" algorithms that involve Householder transformations, but the steps that involve time updating are operations on 2-dimensional vectors and as such the required Householder transformations are equivalent to Givens rotations. There does, however, appear to be some controversy as to the validity of these algorithms: see [23].

solved. Knowledge of $Q_p(n)$ means that $\gamma_p(n)$ is known and also allows $\alpha_p(n)$ to be calculated and thus the least squares residual may then be found. The development of the fast QRD algorithm for adaptive filtering is based, almost entirely, on the principle of constructing partially triangularised matrices from known quantities and then finding a set of rotations to complete the process. This principle is also a key element in the derivation of the well known [5] time-recursive version of the algorithm outlined above. Here the solution at time n, along with the new input data for time (n+1), is used to simplify the calculation of the solution at time (n+1). The approach may be summarised as follows: since

$$X_p(n+1) = \begin{bmatrix} X_p(n) \\ x(n+1)\ldots x(n-p+2) \end{bmatrix} \tag{21}$$

it follows, from equation (11), that

$$\begin{bmatrix} Q_p(n) & \varrho \\ \varrho^t & 1 \end{bmatrix} B(n+1)\, X_p(n+1) = \begin{bmatrix} \beta R_p(n) \\ O \\ x(n+1)\ldots x(n-p+2) \end{bmatrix} \tag{22}$$

Note that the matrix on the right-hand side in equation (22) is very nearly triangular and composed entirely of known quantities. Thus the *actual* application of the rotations specified in equation (22) can be circumvented by the direct *construction* of the partially triangularised matrix shown in equation (22). To complete the triangularisation, define $\hat{Q}_p(n+1)$ to be that set of rotations that annihilates the new data samples by rotating them into the matrix $\beta R_p(n)$. In which case

$$\hat{Q}_p(n+1) \begin{bmatrix} Q_p(n) & \varrho \\ \varrho^t & 1 \end{bmatrix} B(n+1)\, X_p(n+1) = \begin{bmatrix} R_p(n+1) \\ O \end{bmatrix} \tag{23}$$

As only the rotations in $\hat{Q}_p(n+1)$ have to be constructed (as a series of Givens rotations[5]), the computational load is reduced from $O(np)$ to $O(p^2)$. Note that, from equations (22) and (23),

$$Q_p(n+1) = \hat{Q}_p(n+1) \begin{bmatrix} Q_p(n) & \varrho \\ \varrho^t & 1 \end{bmatrix} \tag{24}$$

so that

$$\begin{aligned} \gamma_p(n+1) &= \pi_{n+1}^t\, Q_p(n+1)\, \pi_{n+1} \\ &= \pi_{n+1}^t\, \hat{Q}_p(n+1)\, \pi_{n+1} \end{aligned} \tag{25}$$

and

$$\begin{bmatrix} \underline{u}_p(n+1) \\ \underline{v}_p(n+1) \end{bmatrix} = \hat{Q}_p(n+1) \begin{bmatrix} \beta \underline{u}_p(n) \\ \beta \underline{v}_p(n) \\ y(n+1) \end{bmatrix} = \begin{bmatrix} \underline{u}_p(n+1) \\ \beta \underline{v}_p(n) \\ \alpha_p(n+1) \end{bmatrix} \tag{26}$$

Equation (25) is significant because it shows that "direct residual extraction" is still possible knowing only $\hat{Q}_p(n+1)$ - see equations (18) and (19) and associated discussion. The "time update" technique, presented above, forms an important part of the derivation of the fast adaptive filtering algorithm presented in this memorandum. In particular, we will explicitly use the decomposition for $\underline{v}_p(n)$ shown in equation (26).

4. LINEAR PREDICTION

4.1. Motivation

The $O(p^2)$ QRD-based algorithm for the solution of a p-th order adaptive filtering problem developed above has many desirable features including being a "data domain" algorithm and having a time-recursive formulation, a time-independent computational requirement and a systolic architecture [28]. The time shift redundancy in the adaptive filtering problem can, however, be used to reduce the computational load still further: from $O(p^2)$ to $O(p)$. Note that the set of rotations, either $Q_p(n)$ or $\hat{Q}_p(n)$, that are necessary for the solution of the problem are entirely dependent on the matrix $X_p(n)$. The matrix $X_p(n)$ can, however, be built up in an order recursive manner by adding extra columns which, because of the Toeplitz nature of $X_p(n)$, consist of one new element and a time-shifted version of the previous column. Consider the following decompositions:

$$X_p(n) = \begin{bmatrix} x(1) & \dots & x(2-p) \\ \vdots & & \vdots \\ x(n) & \dots & x(n-p+1) \end{bmatrix} \tag{27}$$

$$= \left[X_{p-1}(n) \; \underline{y}_{b,\,p-1}(n) \right] \tag{28}$$

$$= \begin{bmatrix} x(1) & \underline{z}^t \\ \underline{y}_f(n) & X_{p-1}(n-1) \end{bmatrix} \tag{29}$$

where

$$\underline{y}_f(n) = [x(2), \dots, x(n)]^t \tag{30}$$

$$\underline{y}_{b,p-1}(n) = [x(2-p), \dots, x(n-p+1)]^t \tag{31}$$

and

$$\underline{z} = [x(0), \dots, x(2-p)]^t \tag{32}$$

Note that, from equation (28), if we had already determined the rotation matrix $Q_{p-1}(n)$ that triangularises the matrix[6] $X_{p-1}(n)$ then we could use it to operate on $X_p(n)$ to produce a partially triangularised matrix. In doing so we also have to rotate the vector $\underline{y}_{b,p-1}(n)$ however these are exactly the steps that are required in the QRD-based solution of the (p-1)st order backward linear prediction problem. In the (p-1)st order backward problem, an estimate, at time n, of x(n-p+1) is formed from a linear combination of the data {x(n), ..., x(n-p+2)}. The solution to this problem depends on the triangularisation of the matrix $X_{p-1}(n)$ and the transformation of the reference vector $\underline{y}_{b,p-1}(n)$ - see equation (31). Hence, knowing the solution to the (p-1)st order backward problem at time n would allow us to construct a par-

6. We really mean the matrix $B(n)\,X_{p-1}(n)$ but for the sake of readability $B(n)$ will often be omitted.

tially triangularised version of $X_p(n)$ and so save a certain amount of computation.

Equation (29) allows another partially triangularised version of $X_p(n)$ to be constructed, this time using quantities from the (p-1)st order forward linear prediction problem. The (p-1)st order (forward) linear prediction problem, at time n, is defined as the estimation of x(n) based upon the data {x(n-1), ..., x(n-p+1)}. This involves the triangularisation of the matrix $X_{p-1}(n-1)$ and the transformation of the relevant reference vector $\underline{y}_f(n)$ - see equation (30). As before we can use the decomposition given in equation (29) to produce a partially triangularised version of $X_p(n)$ from known quantities. It should now be clear that the two linear prediction problems of order (p-1) are intimately connected to the problem of determining a set of rotations that triangularise the data matrix $X_p(n)$. The triangularisation of $X_p(n)$ is however central not only to the adaptive filtering problem but also to the p-th order linear prediction problems. We therefore have the beginnings of an order recursive algorithm for linear prediction and for adaptive filtering.

4.2. Forward Linear Prediction

The p-th order forward linear prediction problem, at time n, requires the determination of the vector of filter coefficients $\underline{\omega}_{f,p}(n) = \left[\omega_{f,p}^{(0)}(n), \ldots, \omega_{f,p}^{(p-1)}(n)\right]^t$ that minimises the total prediction error $E_{f,p}(n) = \|B(n)\, \underline{e}_{f,p}(n)\|$ where

$$\underline{e}_{f,p}(n) = X_p(n-1)\, \underline{\omega}_{f,p}(n) + \underline{y}_f(n) \tag{33}$$

As in section 3, the least squares solution to this problem can be found by the method of QR decomposition. It is necessary to determine the rotation matrix $Q_p(n-1)$ that triangularises the weighted data matrix $B(n-1)X_p(n-1)$ and then to apply it to the weighted vector $B(n-1)\, \underline{y}_f(n)$ in order to calculate the angle normalised residual $\alpha_{f,p}(n)$ (cf. equation(19)). We also need to be able to calculate $\gamma_p(n-1)$ (see equation(18)) in order to generate the a-posteriori prediction residual. Note also that the triangularisation of $X_p(n-1)$ is exactly what is required in the solution of the p-th order adaptive filtering problem at time (n-1) - see equation (4). Consider, therefore, the following composite matrix:

$$\left[\underline{y}_f(n)\ X_p(n-1)\ \underline{y}(n-1)\ \underline{\pi}_{n-1}\right] \tag{34}$$

From equations (22) and (23), it is clear that

$$Q_p(n-1)\, \underline{\pi}_{n-1} = \hat{Q}_p(n-1)\pi_{n-1} = [\underline{a}^t_p(n-1), \underline{0}^t, \gamma_p(n-1)]^t \tag{35}$$

where $\underline{a}_p(n-1)$ is a p-dimensional vector. It should be clear, therefore, that we include the vector $\underline{\pi}_{n-1}$ in the above matrix (equation (34)) in order to be able to calculate $\gamma_p(n-1)$ just as the vector $\underline{y}_f(n)$ is present to allow $\alpha_{f,p}(n)$ to be calculated. Similarly the presence of the vector y(n-1) will allow us to calculate $\alpha_p(n-1)$.

From equation (28) we have that

$$\left[\underline{y}_f(n)\ X_p(n-1)\ \underline{y}(n-1)\ \pi_{n-1}\right] = \left[\underline{y}_f(n)\ X_{p-1}(n-1)\ \underline{y}_{b,p-1}(n-1)\ \underline{y}(n-1)\ \pi_{n-1}\right] \quad (36)$$

Hence
$$Q_{p-1}(n-1)\,B(n-1)\left[\underline{y}_f(n)\ X_{p-1}(n-1)\ \underline{y}_{b,p-1}(n-1)\ \underline{y}(n-1)\ \pi_{n-1}\right]$$

$$= \begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{\nu}_{f,p-1}(n) & O & \underline{\nu}_{b,p-1}(n-1) & \underline{\nu}_{p-1}(n-1) & \underline{g}_{p-1}(n-1) \end{bmatrix} \quad (37)$$

where
$$\underline{g}_{p-1}(n-1) = [\underline{0}^t, \gamma_{p-1}(n-1)]^t \quad (38)$$

It is clear that $\underline{\nu}_{f,p-1}(n)$ and $\underline{\nu}_{b,p-1}(n-1)$ must have a time recursive decomposition similar to that given in equation (26) for $\underline{\nu}_{p-1}(n-1)$. Hence

$$\begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{\nu}_{f,p-1}(n) & O & \underline{\nu}_{b,p-1}(n-1) & \underline{\nu}_{p-1}(n-1) & \underline{g}_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \beta\underline{\nu}_{f,p-1}(n-1) & O & \beta\underline{\nu}_{b,p-1}(n-2) & \beta\underline{\nu}_{p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \alpha_{p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix} \quad (39)$$

Now suppose that we had already calculated a rotation matrix, $Q_{f,p}(n-1)$ say, that rotates the vector $\underline{\nu}_{b,p-1}(n-2)$ into a form where only the top element is non-zero. Then

$$\begin{bmatrix} Q_{f,p}(n-1) & \underline{0} \\ \underline{0}^t & 1 \end{bmatrix} \begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \beta\underline{\nu}_{f,p-1}(n-1) & O & \beta\underline{\nu}_{b,p-1}(n-2) & \beta\underline{\nu}_{p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \alpha_{p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \beta\mu_{f,p-1}(n-1) & \underline{0}^t & \beta\varepsilon_{b,p-1}(n-2) & \beta\mu_{p-1}(n-2) & 0 \\ \beta\underline{\lambda}_{f,p-1}(n-1) & O & \underline{0} & \beta\underline{\lambda}_{p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \alpha_{p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix} \quad (40)$$

where the new quantities $\mu_{f,p-1}(n-1)$, $\underline{\lambda}_{f,p-1}(n-1)$, $\mu_{p-1}(n-2)$, $\underline{\lambda}_{p-1}(n-2)$ and $\varepsilon_{b,p-1}(n-1)$ are defined by this operation. Note by analogy with equation (16) that $\varepsilon_{b,p-1}(n-2)$ is the (p-1)st order backward prediction energy at time (n-2). Now in order to complete the triangularisation of the matrix $X_p(n-1)$ (see equation (37)) all that is required is the annihilation of the single element $\alpha_{b,p-1}(n-1)$. This can be carried out using a single Givens rotation:

$$\hat{Q}_{f,p}(n)\begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \mu_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \beta\mu_{f,p-1}(n-1) & \underline{0}^t & \beta\varepsilon_{b,p-1}(n-2) & \beta\mu_{p-1}(n-2) & 0 \\ \beta\underline{\lambda}_{f,p-1}(n-1) & O & \underline{0} & \beta\underline{\lambda}_{p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \alpha_{p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \mu_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \mu_{p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \mu_{f,p-1}(n) & \underline{0}^t & \varepsilon_{b,p-1}(n-1) & \mu_{p-1}(n-1) & \kappa_p(n-1) \\ \beta\underline{\lambda}_{f,p-1}(n-1) & O & \underline{0} & \beta\underline{\lambda}_{p-1}(n-2) & \underline{0} \\ \alpha_{f,p}(n) & \underline{0}^t & 0 & \alpha_p(n-1) & \gamma_p(n-1) \end{bmatrix} \tag{41}$$

$$= \begin{bmatrix} \mu_{f,p}(n) & R_p(n-1) & \underline{\mu}_p(n-1) & \underline{a}_p(n-1) \\ \underline{\nu}_{f,p}(n) & O & \underline{\nu}_p(n-1) & \underline{g}_p(n-1) \end{bmatrix} \tag{42}$$

where $\kappa_p(n\text{-}1)$ is defined by this operation and the identity in equation (42), and hence the labelling of some of the elements in the second matrix above follows by definition (see equation (37)). Thus the sequence of orthogonal transformations shown in equations (37), (40) and (41) solve the p-th order forward linear prediction problem. Note, however, that the intermediate matrix shown on the right-hand side of equation (40) consists entirely of quantities that would be available if the (p-1)st order forward and backward problems had already been solved. If this assumption were true then we could have constructed this intermediate matrix directly, thereby circumventing the need for the operations as outlined in equations (37) and (40). Only the single Givens rotation of equation (41) need actually be performed. This requires a fixed amount of computation because only eight elements, one of which is zero, of the left hand matrix in equation (41) are affected by the required rotations. In order to complete the lattice algorithm for the linear prediction problem, the fast update for the auxiliary (backward) problem must be derived. This can be done along similar line to the above (see section 4.3).

Before considering the backward linear prediction problem, note from above that the "new" quantities $\kappa_p(n\text{-}1)$ and $\underline{\lambda}_{f,p\text{-}1}(n\text{-}1)$ have the following interpretation:

$$\underline{\lambda}_{f,p\text{-}1}(n\text{-}1) = \underline{\nu}_{f,p}(n\text{-}1) \tag{43}$$

$$\begin{bmatrix} \underline{a}_{p-1}(n-1) \\ \kappa_p(n-1) \end{bmatrix} = \underline{a}_p(n\text{-}1) \tag{44}$$

Note also that equation (41) provides a recursive decomposition of the matrix $R_p(n\text{-}1)$. This shows that the diagonal elements of this matrix are in fact the backward prediction residual energy terms for each of the sub-order problems. It also explains why the Givens rotations used in QRD-based linear prediction algorithms should ensure that the diagonal elements of the R matrix are positive (see Bellanger

and Regalia[20] for further insight).

4.3. Backward Linear Prediction

The p-th order backward linear prediction problem, at time n, requires the determination of the vector of filter coefficients $\underline{\omega}_{b,p}(n) = \left[\omega_{b,p}^{(0)}(n), \ \ldots, \ \omega_{b,p}^{(p-1)}(n)\right]^t$ that minimises the total prediction error $E_{b,p}(n) = \|B(n)\, \underline{\varepsilon}_{b,p}(n)\|$ where

$$\underline{\varepsilon}_{b,p}(n) = X_p(n)\, \underline{\omega}_{b,p}(n) + \underline{y}_{b,p}(n) \tag{45}$$

Again the least squares solution to this problem can be found by the method of QR decomposition. It is necessary to determine the rotation matrix $Q_p(n)$ that triangularises the data matrix $X_p(n)$ and then to apply it to the vector $\underline{y}_{b,p}(n)$ in order to calculate $\alpha_{b,p}(n)$ (*cf.* equation (19)). We also need to be able to calculate $\gamma_p(n)$ (see equation (18)) in order to generate the a-posteriori prediction residual. Consider, therefore, the following composite matrix and the illustrated decomposition:

$$\left[X_p(n)\ \underline{y}_{b,p}(n)\ \pi_n\right] = \begin{bmatrix} x(1) & \underline{o}^t & 0 & 0 \\ \underline{y}_f(n) & X_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \pi_{n-1} \end{bmatrix} \tag{46}$$

In equation (46), it has been assumed that the data sequence x(n) is pre-windowed (i.e. $x(n) = 0$ for $n \le 0$). Note that this is the only place in the analysis that requires this assumption. Consider the effect of the rotation matrix $Q_{p-1}(n-1)$ on the lower n-1 rows of the matrix in equation (46) - after weighting by B(n) of course:

$$\begin{bmatrix} 1 & \underline{o}^t \\ \underline{o} & Q_{p-1}(n-1) \end{bmatrix} B(n) \begin{bmatrix} x(1) & \underline{o}^t & 0 & 0 \\ \underline{y}_f(n) & X_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \pi_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \beta^{n-1}x(1) & \underline{o}^t & 0 & 0 \\ \underline{\mu}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{y}_{f,p-1}(n) & O & \underline{y}_{b,p-1}(n-1) & \underline{g}_{p-1}(n-1) \end{bmatrix} \tag{47}$$

As before, all the vectors on the bottom row of the above matrix have a decomposition in terms of their value at the previous time instant and a new element:

$$\begin{bmatrix} \beta^{n-1}x(1) & \underline{0}^t & 0 & 0 \\ \underline{y}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{y}_{f,p-1}(n) & O & \underline{y}_{b,p-1}(n-1) & \underline{g}_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \beta^{n-1}x(1) & \underline{0}^t & 0 & 0 \\ \underline{y}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \beta\underline{y}_{f,p-1}(n-1) & O & \beta\underline{y}_{b,p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix} \tag{48}$$

Now suppose that we have already constructed a rotation matrix $Q_{b,p}(n-1)$ that annihilated the vector $\underline{y}_{f,p-1}(n-1)$ by rotation against the element $\beta^{n-2}x(1)$. Then

$$\begin{bmatrix} Q_{b,p}(n-1) & \underline{0} \\ \underline{0}^t & 1 \end{bmatrix} \begin{bmatrix} \beta^{n-1}x(1) & \underline{0}^t & 0 & 0 \\ \underline{y}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \beta\underline{y}_{f,p-1}(n-1) & O & \beta\underline{y}_{b,p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \beta\varepsilon_{f,p-1}(n-1) & \underline{0}^t & \beta\mu_{b,p-1}(n-2) & 0 \\ \underline{y}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{0} & O & \beta\underline{\lambda}_{b,p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix} \tag{49}$$

Let $\hat{Q}_{b,p}(n)$ be the rotation matrix that annihilates the element $\alpha_{f,p-1}(n)$ by rotation against the element $\beta\varepsilon_{f,p-1}(n-1)$. Application of the transformation $\hat{Q}_{b,p}(n)$ to the above matrix thus yields the result:

$$\hat{Q}_{b,p}(n) \begin{bmatrix} \beta\varepsilon_{f,p-1}(n-1) & \underline{0}^t & \beta\mu_{b,p-1}(n-2) & 0 \\ \underline{y}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{0} & O & \beta\underline{\lambda}_{b,p-1}(n-2) & \underline{0} \\ \alpha_{f,p-1}(n) & \underline{0}^t & \alpha_{b,p-1}(n-1) & \gamma_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \varepsilon_{f,p-1}(n) & \underline{0}^t & \mu_{b,p-1}(n-1) & \varphi_p(n) \\ \underline{y}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{y}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ \underline{0} & O & \beta\underline{\lambda}_{b,p-1}(n-2) & \underline{0} \\ 0 & \underline{0}^t & \alpha_{b,p}(n) & \gamma_p(n) \end{bmatrix} \tag{50}$$

where the new quantity $\varphi_p(n)$ is defined by this equation. The identity of the elements in the bottom row of the above matrix follows because of the following reasoning: bearing in mind the underlying data matrix (see equation (46)), we are attempting to create an upper-triangular pxp matrix in the upper left-hand corner on the matrix in equation (50). At present this sub-matrix is rather sparse and a little thought shows that it is easy to construct a matrix ($\tilde{Q}_{b,p}(n)$ say) that will perform this triangularisation[7]. Specifically,

$$\tilde{Q}_{b,p}(n)\begin{bmatrix} \varepsilon_{f,p-1}(n) & o^t & \mu_{b,p-1}(n-1) & \varphi_p(n) \\ \underline{u}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{u}_{b,p-1}(n-1) & \underline{a}_{p-1}(n-1) \\ o & O & \beta\underline{\lambda}_{b,p-1}(n-2) & o \\ 0 & o^t & \alpha_{b,p}(n) & \gamma_p(n) \end{bmatrix}$$

$$= \begin{bmatrix} R_p(n) & \underline{u}_{b,p}(n) & \underline{a}_p(n) \\ O & \underline{v}_{b,p}(n) & g_p(n) \end{bmatrix} \qquad (51)$$

The important thing to note is that the action of the matrix $\tilde{Q}_{b,p}(n)$ on the above matrix only affects the upper p rows of elements. Thus we conclude that the lower n-p rows of elements of the matrix in equation (50) must be equal to the lower n-p rows of elements of the matrix in equation (51) and the result holds.

Hence the sequence of orthogonal rotations given in equations (47), (49), (50) and (51) solve the p-th order backward linear prediction problem. Following the development of the solution to the forward problem in section 4.2, note that the data matrix on the left-hand side of equation (50) can be constructed directly given the solutions to the (p-1)st order forward and backward problems. Thus the transformations shown in equations (47) and (49) can be by-passed. Furthermore, as both $\alpha_{b,p}(n)$ and $\gamma_p(n)$ are available in the matrix on the right-hand side of equation (50), the transformation shown in equation (51) is not required either, provided we are only interested in the prediction residuals. Thus only the rotations summarised in equation (50) need actually be performed explicitly. This requires a fixed amount of computation because only six elements, one of which is zero, of the left hand matrix in equation (50) are affected by the required rotations.

---

7. In actual fact, it is not necessary that we specify how this triangularisation is achieved since the QR decomposition theorem guarantees us that it is possible. Note, however, that it is crucial to QRD-based fast Kalman algorithm that this transformation is done in a specific way. Interestingly, it has recently been pointed out [20] that the sines of the angles involved in the $\tilde{Q}_{b,p}(n)$ rotation are in fact the reflection coefficients of a "conventional" least squares lattice algorithm.

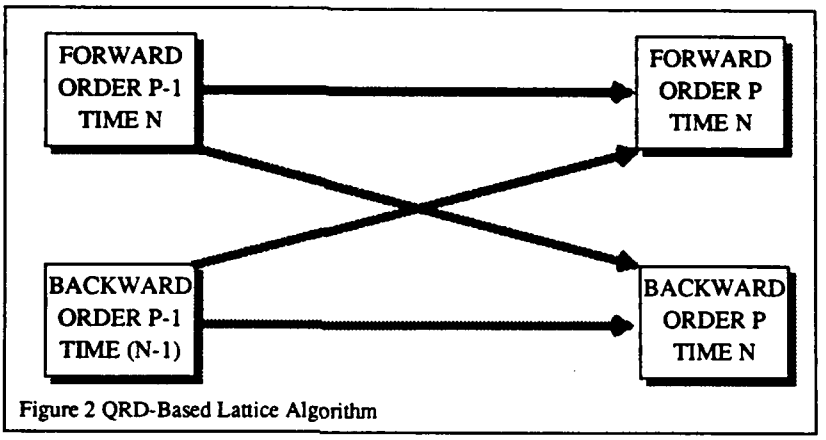


Figure 2 QRD-Based Lattice Algorithm

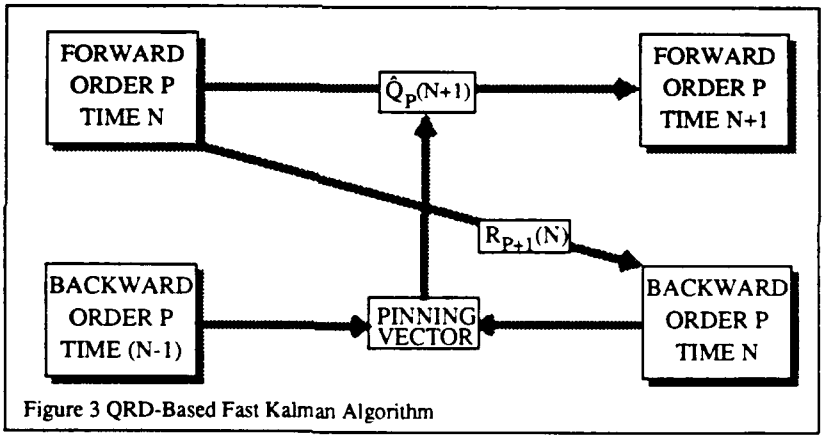


Figure 3 QRD-Based Fast Kalman Algorithm

Note, in passing, that the "new" quantities $\varphi_p(n)$ and $\underline{\lambda}_{b,p-1}(n-2)$ have the following interpretation:

$$\underline{\lambda}_{b,p-1}(n-2) = \underline{v}_{b,p}(n-1) \tag{52}$$

$$\begin{bmatrix} \varphi_p(n) \\ \underline{a}_{p-1}(n-1) \end{bmatrix} = \underline{a}_p(n) \tag{53}$$

Gathering together the results of sections 4.2 and 4.3 we see that it is possible to transform various quantities from the solution to the (p-1)st order forward and backward linear prediction problems, at time n and (n-1) respectively, into the same quantities from the solution to the p-th order problems at time n (see Figure 2). Thus, given that 0-th order linear prediction is trivial, we can generate the solution

to the p-th order problem by iteration in order. The resultant architecture has a lattice structure and, since the number of operations per stage is fixed, requires $O(p)$ operations. Note that by including the adaptive filtering reference vector $\underline{y}(n-1)$ in the calculation of the p-th order forward linear prediction problem (section 4.2) we automatically solve the p-th order adaptive filtering problem for time (n-1).

In most cases, the fact that the adaptive filtering residual is delayed by one time-step will be of no consequence, especially given the regularity of the algorithm/architecture and the benefits this brings with it. It is possible, however, to alter the algorithm presented above so as to solve the adaptive filtering problem at time n but at the expense of more complicated mathematics and a slightly less regular architecture: see section 10.1 for more details.

Note from equation (41) and equations (50) and (51) that it is possible to transform a matrix of quantities from the solution to the (p-1)st order forward problem at time n and a similar matrix from the solution to the (p-1)st order backward problem at time n into the same quantity (viz $R_p(n)$). Thus it is possible to transform quantities from the forward problem at time n into quantities from the backward problem at time n, although this involves a numerically unsound "inverse rotation". This is one of the crucial steps in the derivation of the QRD fast Kalman algorithm[3] (see Figure 3). The remaining step is to deduce the rotation $\hat{Q}_p(n+1)$ from knowledge of quantities from the backward problems at time n and (n-1). This latter step involves using the pinning vector as in the classical Fast Kalman algorithm (see Proudler, McWhirter and Shepherd[16] for more details).

5. IMPLEMENTATION

The QRD-based lattice algorithm developed above can be implemented using the processor architecture shown in Figure 4. The functionality of the processing elements (PE's) is shown in Figure 5 and a "pseudo-code" listing of this algorithm is given in section 10.3.

In the derivation of the algorithm (section 4), no mention was made of how the Givens rotations were to be implemented. There are several ways in which this can be done. The obvious, or "normal", approach involves the calculation of a square-root (see Figure 5), which is a computationally intensive step when using floating point numbers. It is possible to implement the square root operation efficiently by use of CORDIC algorithms[27]. However this requires the use of fixed point arithmetic and it has been found [28] that for most practical applications, a floating point implementation is required. On the other hand, Gentleman[6] and Hammarling[7] have derived a modified Givens rotation that requires no square-root operation and this is clearly advantageous for floating point implementations.

The essence of this "square-root-free" technique is to factorise the triangularised matrix (e.g. $R_p(n)$) into two parts, one of which contains all the quantities which involve a square-root. This latter factor can be calculated indirectly as its square thus avoiding the square-root operation (see section 10.2). However this means that the rotation (an orthogonal transformation) is now implemented by two separate non-orthogonal transformations acting upon the factorised quantities. As such, one would be quite justified in questioning whether or not this square-root-free Givens rotation leads to an orthogonal algorithm. Nevertheless, simulations have shown that, in floating point arithmetic at least, the square-root-free version is more stable numerically (see section 7). This is despite the reservations about the "square-root-free" Givens rotations algorithm *that* have been raised by the numerical analysis community [6][7].

Closer analysis of the normal Givens rotation shows that square-root operation is required only in situations where the square-root of the sum of the squares of two quantities is required. The numerical stability of the "square-root-free" method may well be due[8] to the fact that the act of squaring these two numbers only to take the square-root of their sum is numerically inferior to propagating the squared quantities directly.

The "square-root-free" algorithm mentioned above is an algorithm for *calculating* the required planar rotation. In the QRD-based least squares minimisation problem these rotations also have to be *applied* to various vectors. It is possible to implement a rotation in two ways: in a feedforward or a feedback mode. When implementing a planar rotation (a two-input, two-output transformation), the normal choice would be to calculate each output in terms of the two inputs separately. This leads to a "feedforward" system. However, it is possible to reformulate the transformation so that one output is now dependent on one input and the other output. Initially the motivation behind this reformulation was to reduce the number of multiplications required [25]. It has been shown, however, that this alternative implementation corresponded to the "error-feedback" technique proposed by Ling and Proakis [12].

It is believed that this feedback has a stabilising effect when errors are made in the calculation due to finite-precision effects in the arithmetic. Indeed, computer simulations (see section 7) have shown

8. P A Regalia, Private Communication.

that this error-feedback has a significant effect on the numerical stability of the QRD-based lattice algorithm: the feedback version is quite capable of working with 4 bit mantissas (at least for sequence lengths of up to 10,000 - the longest simulation run so far). It is interesting to note that the feedback structure can be thought of as one half planar rotation and one half hyperbolic rotation. It is well known that a planar rotation is numerically superior to a hyperbolic rotation; however, it would appear that this half-and-half mixture is better than both.

Combined with the two methods for calculating the rotation parameters, the feedforward/feedback choice results in four different variations. The computer simulations of section 7 show that, of these four possible variations, the one that is equivalent to the RMGS error-feedback algorithm (square-root-free with feedback) performs the best in terms of numerical stability. A "pseudo-code" listing of this version of the algorithm is given in the appendix. It is worth emphasising that the basic architecture (Figure 4) is not affected by the choice of rotation technique so that the only difference between the different options is a change of PE's. The function of the "square-root-free with feedback" PE's is shown in Figure 6. Comparison with the $O(p^2)$ QRD-based algorithm [28] shows that the PE's presented here are exactly the same as used in the triangular systolic array. Indeed, when the QRD-based lattice algorithm is generalised to the multi-channel case (section 6), the processing required in each of the lattice stages necessitates the use of triangular arrays of PE's. In fact the lattice stages shown in Figure 4 may be considered to consist of 1×1 triangular arrays!

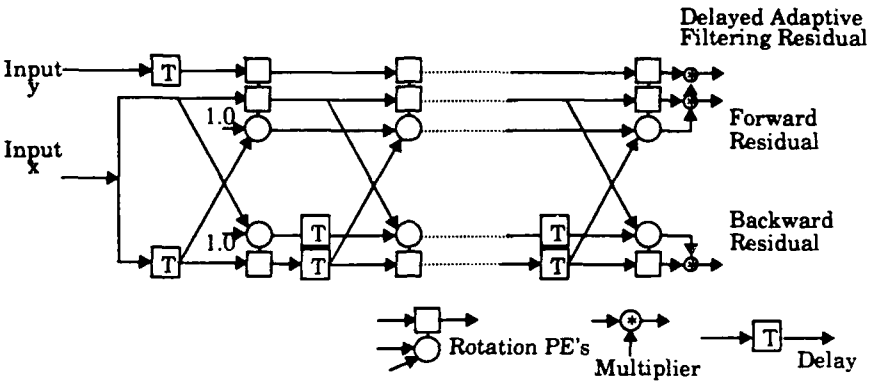
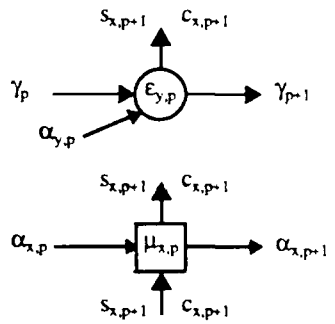


Figure 4 "Delayed" Adaptive Filtering Lattice



$$\varepsilon' := \sqrt{(\beta\varepsilon_{y,p})^2 + |\alpha_{y,p}|^2};$$

IF $\varepsilon' = 0$ THEN $c_{x,p+1} := 1$; $s_{x,p+1} := 0$

ELSE $c_{x,p+1} := \beta\varepsilon_{y,p} / \varepsilon'$; $s_{x,p+1} := \alpha_{y,p} / \varepsilon'$

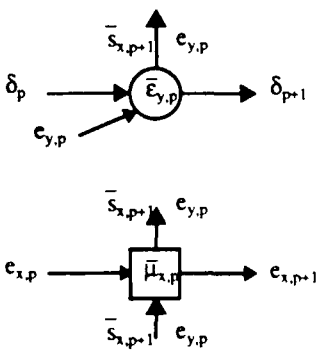
END_IF;

$$\gamma_{p+1} := c_{x,p+1}\,\gamma_p;$$

$$\mu' := c_{x,p+1}\,\beta\mu_{x,p} + s^*_{x,p+1}\,\alpha_{x,p};$$

$$\alpha_{x,p+1} := c_{x,p+1}\,\alpha_{x,p} - s_{x,p+1}\,\beta\mu_{x,p};$$

Figure 5 "Normal" Rotation PE's



$$\bar{\varepsilon}' := \beta^2\bar{\varepsilon}_{y,p} + \delta_p\,|e_{y,p}|^2;$$

IF $\bar{\varepsilon}' = 0$ THEN $\delta_{p+1} := \delta_p$; $\bar{s}_{x,p+1} := 0$

ELSE $\delta_{p+1} := (\beta^2\bar{\varepsilon}_{y,p}/\bar{\varepsilon}')\,\delta_p$; $\bar{s}_{x,p+1} := \delta_p\,e_{y,p} / \bar{\varepsilon}'$

END_IF;

$$\varepsilon_{y,p} := \bar{\varepsilon}';$$

$$e_{x,p+1} := e_{x,p} - e_{y,p}\,\bar{\mu}_{x,p};$$

$$\bar{\mu}_{x,p} := \bar{\mu}_{x,p} + \bar{s}^*_{x,p+1}\,e_{x,p+1};$$

Figure 6 "Square-root-free with Feedback" Rotation PE's

6. MULTI-CHANNEL CASE

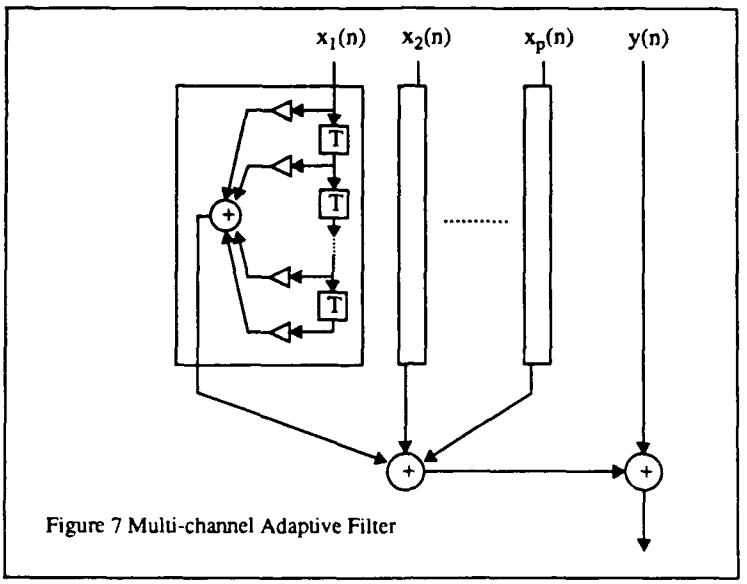


Figure 7 Multi-channel Adaptive Filter

6.1. Problem Definition

The extension of the adaptive filtering algorithm presented above to the wide-band beamforming problem is relatively straight forward: all that is require is that certain scalar quantities be replaced by vectors and some vectors be replace by matrices. The essential features of the derivation presented in section 4 carry over exactly. Rather than reproduce the mathematics of section 4, in the following we merely outline the salient points of the derivation of the multi-channel algorithm. As before, we consider only real signals: the extension to the complex case is straight forward.

In a multi-channel least squares adaptive filtering problem, a set of N p-dimensional weight vectors, $\underline{\omega}_p^{(i)}(n)$ $\{0 \le i \le N-1\}$, is to be found, at time n, that minimises that the sum of the differences, squared, between a reference signal y(n) and a linear combination of N samples from each of p data time series $x_i(n-j)$ $(1 \le i \le p, 0 \le j \le N-1)$. This is equivalent to adaptively filtering p separate time series in order to form the best estimate of the reference signal (see Figure 7). If the p data sequences come from spatially separate antennae then we have a spatial as well as a temporal filtering problem. Specifically, the measure $\|B(n)\, \underline{\varepsilon}_N(n)\|$ is to be minimised, where:

$$\underline{\varepsilon}_N(n) = X_N(n)\, \underline{\omega}_N(n) + \underline{y}(n) \tag{54}$$

$$X_N(n) =$$

$$\begin{bmatrix} x_1(1) & x_2(1) & \dots & x_p(1) & x_1(0) & x_2(0) & \dots & x_p(0) & \dots\dots & x_1(2-N) & x_2(2-N) & \dots & x_p(2-N) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ x_1(n) & x_2(n) & \dots & x_p(n) & x_1(n-1) & x_2(n-1) & \dots & x_p(n-1) & \dots\dots & x_1(n-N+1) & x_2(n-N+1) & \dots & x_p(n-N+1) \end{bmatrix} \tag{55}$$

$$\equiv \begin{bmatrix} \underline{x}^t(1) & \dots & \underline{x}^t(2-N) \\ \vdots & & \vdots \\ \underline{x}^t(n) & \dots & \underline{x}^t(n-N+1) \end{bmatrix} \tag{56}$$

$$\underline{\omega}_N(n) = \begin{bmatrix} \underline{\omega}_p^{(0)}(n) \\ \underline{\omega}_p^{(1)}(n) \\ \vdots \\ \underline{\omega}_p^{(N-1)}(n) \end{bmatrix}, \tag{57}$$

and

$$\underline{y}(n) = [y(1), \dots, y(n)]^t \tag{58}$$

Equation (56) serves to define the new vector quantity $\underline{x}(n)$. Note that, apart from the change from scalar to vector quantities, equation (56) is identical to equation (5). The solution of this least squares minimisation problem via QR decomposition is no different from any other: it requires the determination of an orthogonal matrix $Q_N(n)$ that transforms the matrix $B(n)\,X_N(n)$ into an upper triangular form. The fact that the matrix $X_N(n)$ is block-Toeplitz allows us to use the ideas developed in section 4 to construct an order recursive algorithm. Again this relies on the iterative structure present in the multi-channel linear prediction problems.

6.2. Forward Linear Prediction

In the N-th order multi-channel forward linear prediction problem, an estimate of the vector $\underline{x}(n)$ is formed, at time n, from a linear combination of the data $\{\underline{x}(n-1), \dots, \underline{x}(n-N)\}$. Thus it is necessary to determine the rotation matrix $Q_N(n-1)$ that triangularises the data matrix $X_N(n-1)$. Consider, therefore, the following composite matrix:

$$\left[Y_f(n)\ X_N(n-1)\ \underline{y}(n-1)\ \underline{\pi}_{n-1}\right] \tag{59}$$

where the "reference matrix" $Y_f(n)$ is defined as

$$Y_f(n) \equiv \begin{bmatrix} \underline{x}^t(2) \\ \vdots \\ \underline{x}^t(n) \end{bmatrix} = \begin{bmatrix} x_1(2) & x_2(2) & \dots & x_p(2) \\ \vdots & \vdots & & \vdots \\ x_1(n) & x_2(n) & \dots & x_p(n) \end{bmatrix} \tag{60}$$

and it the multi-channel equivalent to $\underline{y}_f(n)$ - see equation section (30). From equation (56) we have that

$$\left[Y_f(n)\ X_N(n-1)\ \underline{y}(n-1)\ \underline{\pi}_{n-1} \right] = \left[Y_f(n)\ X_{N-1}(n-1)\ Y_{b,N-1}(n-1)\ \underline{y}(n-1)\ \underline{\pi}_{n-1} \right] \tag{61}$$

where $Y_{b,N-1}(n-1)$ is the reference matrix for the (N-1)st order backward linear prediction problem at time (n-1) (see section 4.3):

$$Y_{b,N-1}(n-1) \equiv \begin{bmatrix} \underline{x}^t(2-N) \\ \vdots \\ \underline{x}^t(n-N) \end{bmatrix} = \begin{bmatrix} x_1(2-N) & x_2(2-N) & \dots & x_p(2-N) \\ \vdots & \vdots & & \vdots \\ x_1(n-N) & x_2(n-N) & \dots & x_p(n-N) \end{bmatrix} \tag{62}$$

Hence
$$Q_{N-1}(n-1)\, B(n-1) \left[Y_f(n)\ X_{N-1}(n-1)\ Y_{b,N-1}(n-1)\ \underline{y}(n-1)\ \underline{\pi}_{n-1} \right]$$

$$= \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{\mu}_{N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ V_{f,N-1}(n) & O & V_{b,N-1}(n-1) & \underline{\nu}_{N-1}(n-1) & \underline{g}_{N-1}(n-1) \end{bmatrix} \tag{63}$$

where the matrices $U_{f,N-1}(n)$, $V_{f,N-1}(n)$, $U_{b,N-1}(n-1)$ and $V_{b,N-1}(n-1)$ are the multi-channel equivalents of $\underline{u}_{f,p-1}(n)$, $\underline{v}_{f,N-1}(n)$, $\underline{u}_{b,p-1}(p-1)$ and $\underline{v}_{b,p-1}(p-1)$ respectively. Note that $V_{f,N-1}(n)$ and $V_{b,N-1}(n-1)$ have a time recursive decomposition similar to that given in equation (26) for $\underline{v}_{p-1}(n-1)$ and that $R_{N-1}(n-1)$ is a (N-1)p×(N-1)p upper triangular matrix.

Now suppose that we had already calculated a rotation matrix, $Q_{f,N}(n-1)$ say, that transforms the matrix $V_{b,N-1}(n-2)$ into an upper-triangular form[9]. Then

$$\begin{bmatrix} Q_{f,N}(n-1) & \underline{o} \\ \underline{o}^t & 1 \end{bmatrix} \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{\mu}_{N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta V_{f,N-1}(n-1) & O & \beta V_{b,N-1}(n-2) & \beta \underline{\nu}_{N-1}(n-2) & \underline{o} \\ \underline{\alpha}^t_{f,N-1}(n) & \underline{o}^t & \underline{\alpha}^t_{b,N-1}(n-1) & \alpha_{N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{\mu}_{N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta M_{f,N-1}(n-1) & O & \beta E_{b,N-1}(n-2) & \beta \underline{\mu}_{N-1}(n-2) & \underline{o} \\ \beta \Lambda_{f,N-1}(n-1) & O & O & \beta \underline{\lambda}_{N-1}(n-2) & \underline{o} \\ \underline{\alpha}^t_{f,N-1}(n) & \underline{o}^t & \underline{\alpha}^t_{b,N-1}(n-1) & \alpha_{N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix} \tag{64}$$

9. Remember that we are restricted to using only orthogonal operations!

Note that there are now p residuals for both the forward and backward linear prediction problems hence the vectors $\underline{\alpha}_{f,N-1}(n)$ and $\underline{\alpha}_{b,N-1}(n-1)$. Now in order to complete the triangularisation of the matrix $X_N(n-1)$ all that is required is the annihilation of the vector $\underline{\alpha}_{b,N-1}(n-1)$. This can be carried out using a sequence of Givens rotations rather like that used in equation (23):

$$\hat{Q}_{f,N}(n)\begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{\mu}_{N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta M_{f,N-1}(n-1) & O & \beta E_{b,N-1}(n-2) & \beta\underline{\mu}_{N-1}(n-2) & \varrho \\ \beta\Lambda_{f,N-1}(n-1) & O & O & \beta\underline{\lambda}_{N-1}(n-2) & \varrho \\ \underline{\alpha}^t_{f,N-1}(n) & \varrho^t & \underline{\alpha}^t_{b,N-1}(n-1) & \alpha_{N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{\mu}_{N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ M_{f,N-1}(n) & O & E_{b,N-1}(n-1) & \underline{\mu}_{N-1}(n-1) & \kappa_N(n-1) \\ \beta\Lambda_{f,N-1}(n-1) & O & O & \beta\underline{\lambda}_{N-1}(n-2) & \varrho \\ \underline{\alpha}^t_{f,N}(n) & \varrho^t & \varrho^t & \alpha_N(n-1) & \gamma_N(n-1) \end{bmatrix} \tag{65}$$

Note that the intermediate matrix shown on the left-hand side of equation (41) consists entirely of quantities that would be available if the (N-1)st order forward and backward problems had already been solved. Thus only the Givens rotations defined by equation (41) need actually be performed. This can be implemented on the standard triangular systolic array [28] in $O(p^2)$ operations per time instant.

6.3. Backward Linear Prediction

The N-th order backward linear prediction problem is defined as the estimation, at time n, of $\underline{x}(n-N)$ from a linear combination of the data $\{\underline{x}(n), \ldots, \underline{x}(n-N+1)\}$. Consider, therefore, the following composite matrix and the illustrated decomposition:

$$\begin{bmatrix} X_p(n) & Y_{b,N}(n) & \underline{\pi}_n \end{bmatrix} = \begin{bmatrix} \underline{x}^t(1) & \varrho^t & \varrho^t & 0 \\ Y_f(n) & X_{N-1}(n-1) & Y_{b,N-1}(n-1) & \underline{\pi}_{n-1} \end{bmatrix} \tag{66}$$

The rotation matrix $Q_{N-1}(n-1)$ will triangularise the data matrix $X_{N-1}(n-1)$ hence:

$$\begin{bmatrix} 1 & \varrho^t \\ \varrho & Q_{N-1}(n-1) \end{bmatrix} B(n) \begin{bmatrix} \underline{x}^t(1) & \varrho^t & \varrho^t & 0 \\ Y_f(n) & X_{N-1}(n-1) & Y_{b,N-1}(n-1) & \underline{\pi}_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \beta^{n-1}\underline{x}^t(1) & \varrho^t & \varrho^t & 0 \\ U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta V_{f,N-1}(n-1) & O & \beta V_{b,N-1}(n-2) & \varrho \\ \underline{\alpha}^t_{f,N-1}(n) & \varrho^t & \underline{\alpha}^t_{b,N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix} \tag{67}$$

where we have explicitly used the familiar decomposition for the matrices $V_{f,N-1}(n)$ and $V_{b,N-1}(n-1)$.

The calculation of the quantities necessary for direct residual extraction can now be achieved in three steps. First imagine that we had operated on the above matrix with an orthogonal transformation $(Q^{(e)}_{b,N}(n-1))$ that moves the top row of elements down to the row just above the top of the matrix $V_{f,N-1}(n-1)$. Then suppose that we have already constructed a rotation matrix $Q_{b,N}(n-1)$ that performed a QR decomposition on the composite matrix

$$\begin{bmatrix} \beta^{n-2}\underline{x}^t(1) \\ V_{f,N-1}(n-1) \end{bmatrix} \tag{68}$$

Then

$$\begin{bmatrix} Q_{b,N}(n-1) & \underline{0} \\ \underline{0}^t & 1 \end{bmatrix} \begin{bmatrix} Q^{(e)}_{b,N}(n-1) & \underline{0} \\ \underline{0}^t & 1 \end{bmatrix} \begin{bmatrix} \beta^{n-1}\underline{x}^t(1) & \underline{0}^t & \underline{0}^t & 0 \\ U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta V_{f,N-1}(n-1) & O & \beta V_{b,N-1}(n-2) & \underline{0} \\ \underline{\alpha}^t_{f,N-1}(n) & \underline{0}^t & \underline{\alpha}^t_{b,N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta E_{f,N-1}(n-1) & O & \beta M_{b,N-1}(n-2) & \underline{0} \\ O & O & \beta \Lambda_{b,N-1}(n-2) & \underline{0} \\ \underline{\alpha}^t_{f,N-1}(n) & \underline{0}^t & \underline{\alpha}^t_{b,N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix} \tag{69}$$

Finally, following section 4.3, let $\hat{Q}_{b,N}(n)$ be the rotation matrix that annihilates the row vector $\underline{\alpha}^t_{f,N-1}(n)$ by rotation against the triangular matrix $\beta E_{f,N-1}(n-1)$. Then

$$\hat{Q}_{b,N}(n) \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ \beta E_{f,N-1}(n-1) & O & \beta M_{b,N-1}(n-2) & \underline{0} \\ O & O & \beta \Lambda_{b,N-1}(n-2) & \underline{0} \\ \underline{\alpha}^t_{f,N-1}(n) & \underline{0}^t & \underline{\alpha}^t_{b,N-1}(n-1) & \gamma_{N-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} U_{f,N-1}(n) & R_{N-1}(n-1) & U_{b,N-1}(n-1) & \underline{a}_{N-1}(n-1) \\ E_{f,N-1}(n) & O & M_{b,N-1}(n-1) & \underline{\varphi}_N(n) \\ O & O & \beta \Lambda_{b,N-1}(n-2) & \underline{0} \\ \underline{0}^t & \underline{0}^t & \underline{\alpha}^t_{b,N}(n) & \gamma_N(n) \end{bmatrix} \tag{70}$$

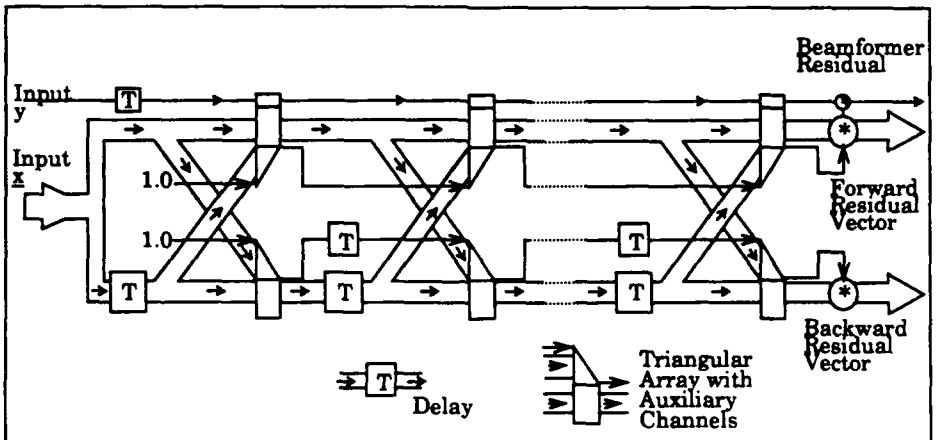


Figure 8. Lattice of Triangular Arrays.

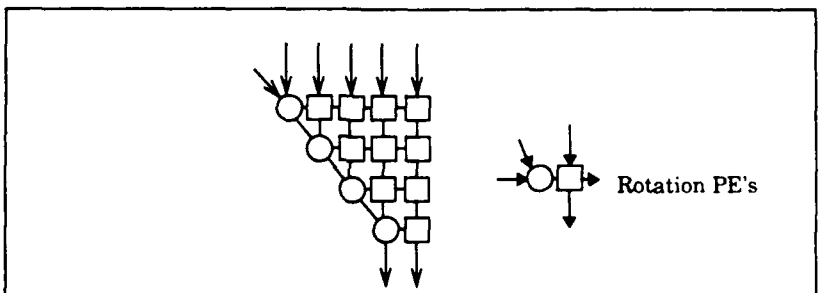


Figure 9. Triangular Systolic Array.

and we have achieved our objective. As in section 4.3, we do not have to complete the formation of the triangular matrix (an Np×Np matrix in this case) since any operation necessary to achieve this would not affect the lower two rows of the composite matrix shown in equation (50). Again the rotation shown in equation (50) can be implemented using a triangular systolic array in $O(p^2)$ operations.

Thus by the use of equations (41) and (50), we can calculate the solution to the N-th order multi-channel linear prediction problems in $O(p^2)$ operations given the solutions to the (N-1)st order problems. As before, the rotation that are necessary to solve the forward linear prediction problem automatically solve the p-th order adaptive filtering problem for time (n-1) as well. The resultant architecture (see Figure 8) has a lattice structure where each stage of the lattice contains two triangular systolic arrays (Figure 9). Each of these arrays solve a p-th order recursive least squares minimisation. The total number of operations necessary to solve an N-th order multi-channel adaptive filtering problem, with p channels, is $O(Np^2)$.

Once again, it is possible to alter the algorithm presented above so as to solve the adaptive filtering problem at time n but at the expense of more complicated mathematics and a slightly less regular archi-

tecture: see section 10.1 for details of the single channel case: the extension to the multi-channel case should be obvious.

7. SIMULATIONS

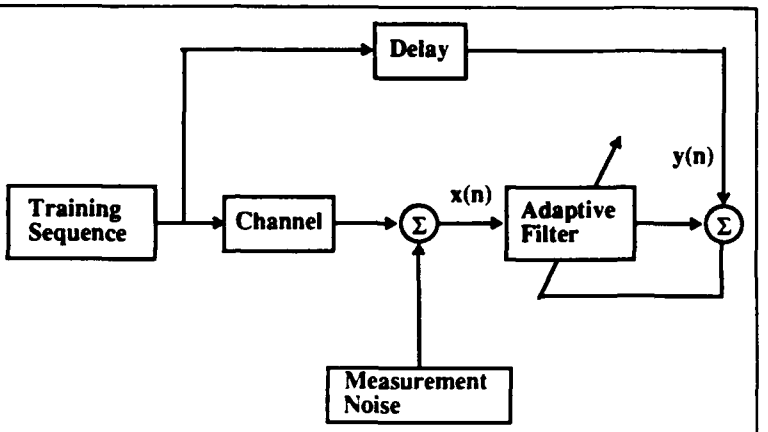


Figure 10 Channel Equaliser Experiment.

The single-channel QRD-based lattice filter algorithm has been simulated on a computer and compared with the full $O(p^2)$ QRD triangular array algorithm. The results show that the two algorithms produce virtually identical results for "sensible" wordlengths. As the wordlength is reduced the lattice algorithm begins to suffer before the full triangular array does. This is to be expected as the lattice algorithm relies on an exact mathematical relationship between the forward and backward linear prediction problems which is progressively made void as numerical errors are made.

The situation considered[10] is that of a channel equaliser (Figure 10) for a data channel that has a "raised cosine" impulse response (equation (71)).

$$h(n) = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{W}(n-2)\right)\right] & n = 1, 2, 3. \\ 0 & \text{otherwise} \end{cases} \qquad (71)$$

By varying the parameter W, the amount of intersymbol-interference between a given symbol and the two either side of it can be changed. This in effect controls the eigenvalue spread of the data covariance matrix (see Table 1.).

An 11th order adaptive QRD-based least-squares lattice filter is used to equalise the channel response. In order to "train" the equaliser, the transmission channel is fed with a polar (± 1) pseudorandom training sequence. This sequence, delayed by seven time instants, is used as the reference signal for the adaptive filtering algorithm. The delay is inserted to ensure that the adaptive filter has an impulse

10. Suggested by S Haykin.

W	$\lvert\lambda_{max}/\lambda_{min}\rvert$
2.9	6.1
3.1	11.2
3.3	21.9
3.5	47.5

Table 1. Eigenvalue Spread

response that is symmetrical about the centre tap. A small quantity of "measurement" noise, in the form of a pseudorandom sequence with an approximately gaussian probability distribution function, is added to the channel output. The noise sequence used has zero mean and a variance of 0.001. The "forget factor" β (see equation (2)) was fixed at a value of 0.996, which implies an effective data window of 250 time samples.

All calculations within the algorithm were performed using limited-precision floating point arithmetic. Only the number of bits in the mantissa are varied in the experiments: the number of bits in the exponent is fixed at eight. No quantities internal to the adaptive filtering algorithm are held to a greater precision than for the outputs: the results of *all* arithmetic operations are immediately reduced to the required precision.

The performance of the equaliser is monitored by recording the ensemble-averaged, squared a-priori equalisation error (see equation (10)). This has the advantage that it shows how close to convergence the algorithm is whilst still showing, asymptotically, the least square equalisation error. The ensemble average is taken over 100 realisations of the experiment. Several experiments were performed using various combinations of parameters and algorithms The main results are discussed below, however, for completeness a complete set of results are reproduced in the attached annex.

Figure 11 and Figure 12 show the basic performance of the QRD-based lattice equaliser system for different values of wordlength and eigenvalue spread. Figure 11 shows that, with double-precision arithmetic, the rate of convergence is more or less insensitive to the different eigenvalue spread settings: as would be expected from a recursive least squares minimisation process. Figure 12 illustrates how the wordlength affects the performance for a fixed eigenvalue spread (W=2.9). Note that there is very little discernable difference between the systems using 12, 16 and 56 (IEEE double-precision) bit mantissas.

Figure 13 shows a comparison of the QRD-based lattice algorithm with the full QRD-based triangular systolic array version. Two other systems are also shown in this figure: they are the "square-root-free with feedback" forms of the lattice algorithm and the array algorithm. This figure shows the case of 4 bit mantissas and a fixed eigenvalue spread setting (W=2.9). This may be considered to be an excessively short wordlength. The reason for this choice is that often finite precision effect are often only manifested after the round-off errors have had time to accumulate [2]. By using a small wordlength, the appearance of such effects occur sooner thus reducing the time necessary to perform the simulation.

In most cases, a p-th order adaptive filter will converge within 2p time instants. At this point the a-priori residual will have reached a value primarily determined by the eigenvalue spread and not the wordlength. As round-off errors accumulate, the a-priori error will increase indicating a loss of accuracy

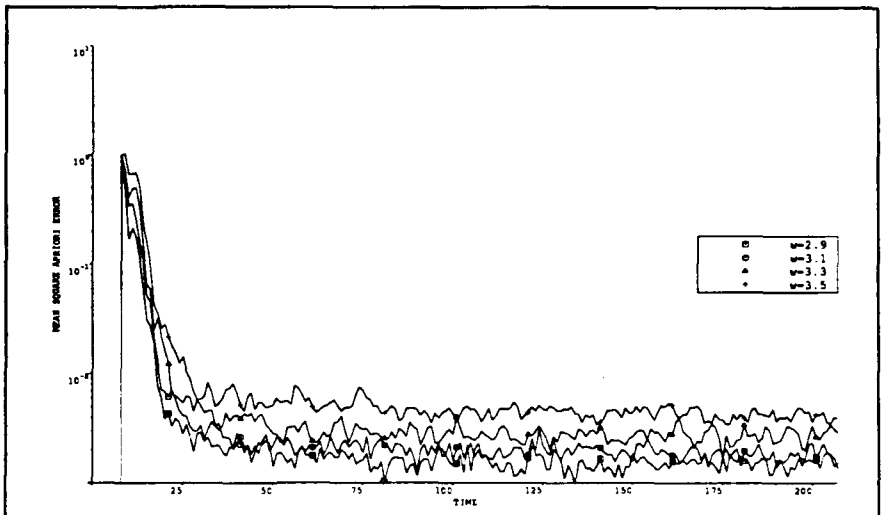


Figure 11 Normal QRD Lattice: Effect of Eigenvalue Spread.

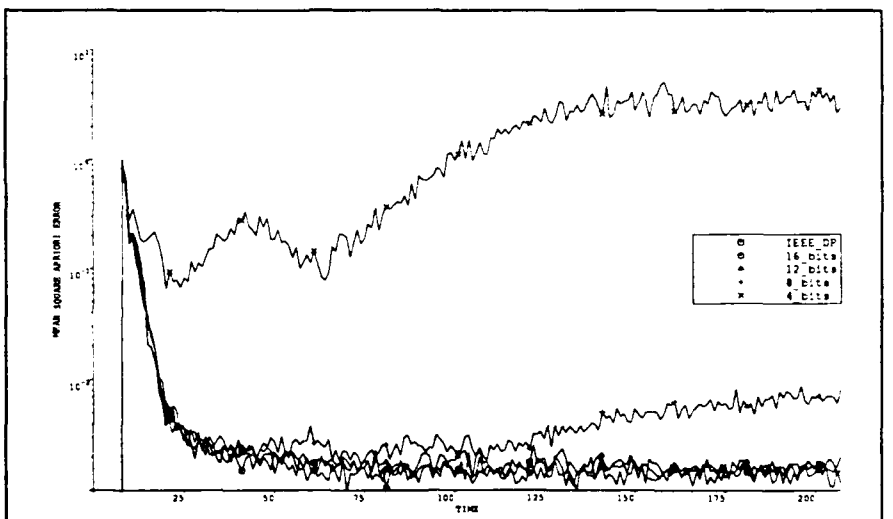


Figure 12 Normal QRD Lattice: Effect of Wordlength.

in the algorithm. Up to a run length of 10000, the longest simulation run to date, the normal lattice algorithm retains its post-convergence accuracy for 12 bit mantissas. In the case of the square-root-free with feedback lattice, the same behaviour is seen using only 4 bit mantissas.

It can be seen, in Figure 13, that the faster, lattice algorithm is only marginally worse than the full triangular array version thus demonstrating that little penalty has been paid in reducing the computa-

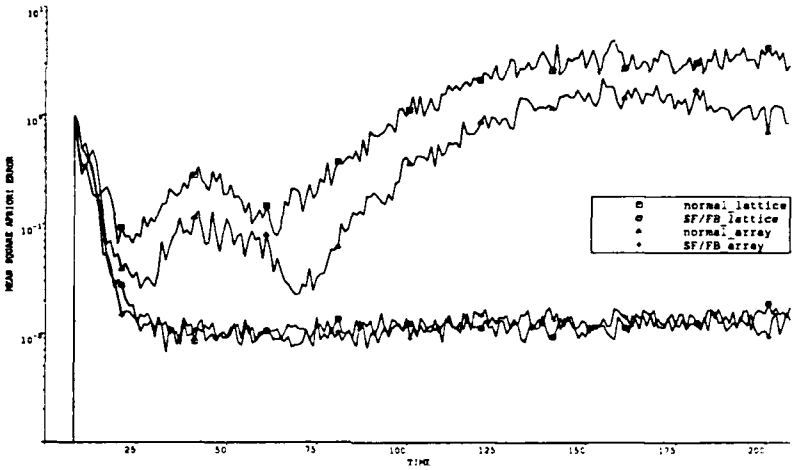


Figure 13 Comparison of Lattice and Array.

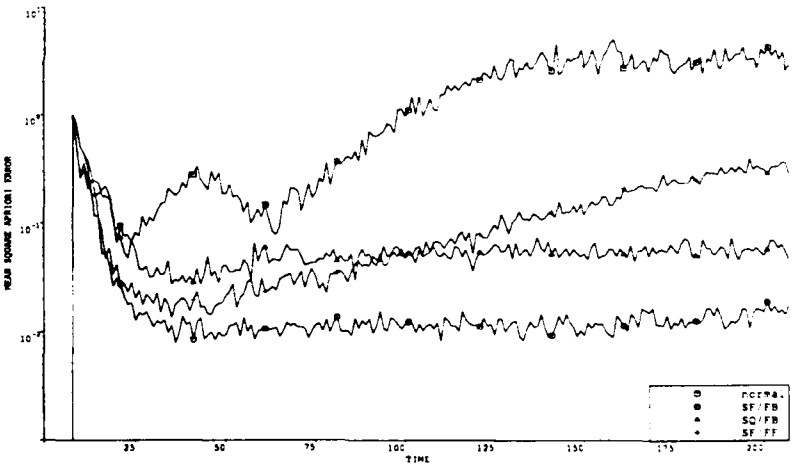


Figure 14 Comparison of Givens Algorithms.

tional load. As expected, the square-root-free with feedback versions of the algorithm perform better than the basic version. There was no discernable difference between the lattice version and the array version in any of the simulations run so far. This would seem to demonstrate the power of the feedback technique in improving the numerical accuracy of these algorithms.

The relative effect of the square-root-free and the feedback techniques can be seen in Figure 14. This shows the performance of the lattice algorithms with 4 bit mantissas and fixed eigenvalue spread

setting (W=2.9) for the four possible Givens rotation algorithms. From this it can be seen that there is indeed a numerical advantage to avoiding the square-root operation but that the most significant improvement comes about by introducing the "error feedback".

All of the above observations appear to hold essentially independently of the eigenvalue spread.

8. CONCLUSION

We have presented a derivation, from first principles, of a fast QRD-based lattice-ladder algorithm for solving the adaptive filter problem (with pre-windowed data). Contained within this algorithm is a new lattice filter algorithm for least squares linear prediction. The extension to the multi-channel case highlights the similarity of the adaptive filtering algorithm with the more general $O(p^2)$ QRD triangular systolic array. The derivation resented here shows that other, recent, orthogonal least squares lattice algorithms are true QRD-based algorithms. The relationship between these different derivations has been highlighted.

The algorithm has as its root the QRD recursive least squares minimisation algorithm and therefore is expected to be numerically stable. Computer simulations would seem to confirm this expectation: the results of the simulations of the new algorithm, using limited-precision floating-point arithmetic, show that very little penalty has been paid in reducing the computational load. The QRD-based lattice algorithm works essentially as well as the QRD-based triangular systolic array algorithm but requires only $O(p)$ operations per time instant as compared with $O(p^2)$ for the array.

Of the four possible algorithms for implementing the Givens rotations, the simulation results show that the square-root-free with feedback form of the algorithm is empirically better than the standard form. The former implementation coincides with the previously known RMGS lattice algorithm of Ling, Manolakis and Proakis [11]. The algorithm is explicitly presented in both the normal (square-root/ feedforward) and square-root-free with feedback forms.

9. REFERENCES

[1] M G Bellanger, "The FLS-QR Algorithm for Adaptive Filtering", *Proc. NATO Advanced Study Institute on Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, Leuven, Belgium, 1-12th August 1988.

[2] J M Cioffi, "Finite Precision Effects in Adaptive Filtering", *IEEE trans. CAS*, vol.34, no.7, pp.821-833, July 1987.

[3] J M Cioffi, "The Fast Adaptive Rotors RLS Algorithm", *IEEE trans. ASSP* - to be published (see J M Cioffi, "High Speed Systolic Implementation of Fast QR Adaptive Filters", *Proc. IEEE International Conference on ASSP*, vol. DIII, pp. 1584-1587, New York, 11-14th April 1988).

[4] J. M. Cioffi, "The Fast Householder Filters RLS Adaptive Filter", *Proc. IEEE Int. Conf. on ASSP*, vol. D, pp. 1619-1622, Albuquerque, NM, USA, 3-6th April 1990.

[5] W M Gentleman and H T Kung, "Matrix Triangularisation by Systolic Array", *Proc. SPIE Real Time Signal Processing IV*, vol. 298, 1981.

[6] W M Gentleman, "Least-squares Computations by Givens Transformations without Square-Roots", *J. Inst. Maths. Applic.* vol. 12, pp. 329-336, 1973.

[7] S Hammarling, "A Note on Modifications to the Givens Plane Rotation", *J. Inst. Maths. Applics.*, vol. 13, pp. 215-218, 1974.

[8] S Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1981.

[9] P S Lewis, QR-based Algorithms for Multichannel Adaptive Least Squares Lattice Filters, *submitted to IEEE Trans. ASSP*, (but see: "QR Algorithm and Array Architectures for Multichannel Adaptive Least Squares Lattice Filters", *Proc. 1988 Int. Conf. ASSP*, pp.2041-2044, New York, USA, April 1988).

[10] F Ling, D Manolakis and J G Proakis, "A Flexible, Numerically Robust Array Processing Algorithm and its Relationship to the Givens Transformation", *Proc. IEEE Int. Conf. on ASSP*, April 1986, Tokyo, Japan.

[11] F Ling, D Manolakis and J G Proakis, "A Recursive Modified Gram-Schmidt Algorithm for Least-Squares Estimation", *IEEE Trans. ASSP*, vol.34, no.4, pp. 829-836, Aug. 1986.

[12] F Ling and JG Proakis, "A Generalised Multichannel Least Squares Lattice Algorithm Based on Sequential Processing Stages", *IEEE trans. ASSP*, vol. 32, no. 2, pp. 381-389, Apr. 1987.

[13] F Ling, "Givens Rotation Based Least-squares Lattice and Related Algorithms", *submitted to IEEE Trans. ASSP*, (see "Efficient Least-Squares Lattice Algorithms based on Givens Rotation with Systolic Array Implementations", *Proc. IEEE Int. Conf. on ASSP*, paper no. 40.D9.4, Glasgow, UK, May 1989.

[14] J G McWhirter, "Recursive Least Squares Minimisation using a Systolic Array", *Proc. SPIE Real Time Signal Processing IV*, vol. 431, pp. 105-112, 1983.

[15] J G McWhirter and T J Shepherd, "Systolic Array Processor for MVDR Beamforming", *IEE Proceedings*, vol. 136, Pt. F, no. 2, pp. 75-80, April 1989.

[16] I K Proudler, J G McWhirter and T J Shepherd, "Fast QRD-based Algorithms for Least Squares Linear Prediction", *Proc. IMA Conference on Mathematics in Signal Processing*, Warwick, England, 13-15th December 1988.

[17] I K Proudler, J G McWhirter and T J Shepherd, "Computationally Efficient QRD-based Wide-band Beamforming", *Proc. IEEE Int. Conf. on ASSP*, paper no. 348 A13.12, Albuquerque, NM, USA, April 1990.

[18] I K Proudler, J G McWhirter and T J Shepherd, "The QRD-based Least Squares Lattice Algorithm: Some Computer Simulations Using Finite Wordlengths", *Proc. IEEE Int. Symp. on Circuits and Systems*, New Orleans, Lo, USA, 1-3rd May 1990, pp.258-261.

[19] I K Proudler, J G McWhirter and T J Shepherd, "The QRD-Based Least Squares Lattice Algorithm for Wide-band Beamforming and Adaptive Filtering", *Proc. ISSPA-90*, Brisbane, Australia, 27-31st August 1990

[20] P A Regalia and M G Bellanger, "On the Duality Between Fast QR Methods and Lattice

Methods in Least Squares Adaptive Filtering", *submitted to IEEE Trans. ASSP*, 1989.

[21] P A Regalia, "System Theoretical Properties in the Stability Analysis of QR-based Fast Least-squares Algorithms", *Internal Report*, Dépt. Electronique et Communications, Institut National des Télécommunications Paris, France, 15 March 1990.

[22] M J Shensa, "Recursive Least Squares Lattice Algorithms - A Geometric Approach", *IEEE trans. AC*, vol. 26, no. 3, pp. 696-702, June 1981.

[23] D. T. M. Slock, "A Fast Householder Transversal Filter (FHTF) Algorithm for RLS Adaptive Filtering?", *Proc. 1990 Conf. on Information Systems and Sciences*, Princeton, 21-23rd March 1990.

[24] D. T. M. Slock, "Reconciling Fast RLS Lattice Algorithms and QR Algorithms", *Proc. IEEE. International Conference on ASSP.*, Albuquerque, NM, USA, 3-6th April 1990.

[25] T J Shepherd and J G McWhirter, "A Pipelined Array for Linearly Constrained Least Squares Optimisation", *Proc. IMA Conference on Mathematics in Signal Processing*, pp. 457-483, September 1985, Bath, England, Oxford University Press, T S Durrani et al (Ed).

[26] T J Shepherd and J Hudson, "Parallel Weight Extraction from a Systolic Adaptive Beamformer", *Proc. IMA Conference on Mathematics in Signal Processing*, Warwick, England, 13-15th December 1988.

[27] J E Volder, "The CORDIC Trigonometric Computing Technique", IRE trans. Electronic Computers, vol. EC-8, pp.330-334, September 1959.

[28] C R Ward, P J Hargrave, J G McWhirter, A Novel Algorithm and Architecture for Adaptive Digital Beamforming, *IEEE trans. Antennas and Propagation*, vol. AP-34, no.3, 1986, pp.338-346.

[29] B Yang and J F Böhme, "On a Parallel Implementation of the Adaptive Multichannel Least-squares Lattice Filter", *Proc. Int. Symp. on Signals Systems and Electronics*, Erlangen, Fed. Rep. of Germany, Sept. 1989.

[30] B Yang and J F Böhme, "A Linear Systolic Array for Constrained Least-squares Problems", *Proc. IMA Conference on Mathematics in Signal Processing*, Warwick, England, 13-15th December 1988.

10. APPENDIX

10.1. Joint Process Estimation

It was shown in sections 4.2 and 4.3 that a lattice filter algorithm could be developed to solve the p-th order adaptive filtering problem for the previous time instant. Producing the adaptive filtering residual for time (n-1) at time n may well be sufficient for some purposes, however, it is possible to obtain the most up-to-date solution possible by including the effect of the very latest datum x(n). Consider the following data matrix:

$$\left[X_p(n)\ \underline{y}(n)\ \underline{\pi}_n\right] = \begin{bmatrix} x(1) & \underline{o}^t & y(1) & 0 \\ \underline{y}_f(n) & X_{p-1}(n-1) & \underline{\Psi}(n) & \underline{\pi}_{n-1} \end{bmatrix} \tag{72}$$

where

$$\underline{\Psi}(n) = [y(2), \ldots, y(n)]^t \tag{73}$$

is the reference vector of what may be considered to be an auxiliary joint process estimation problem. Note that this definition is more closely akin to that for forward linear prediction (equation (30)) than the original definition of the adaptive filtering problem (equation (6)). In equation (72), it has again been assumed that the data sequence x(n) is pre-windowed (i.e. $x(n) = 0$ for $n \leq 0$).

By comparison with the development of the order update for backward linear prediction (section 4.3) it should be clear that the matrix $X_p(n)$, in equation (72), can be triangularised by the series of rotations outlined in equations (47), (49), (50) and (51). In fact

$$\tilde{Q}_{b,p}(n)\ \hat{Q}_{b,p}(n) \begin{bmatrix} Q_{b,p}(n-1) & \underline{o} \\ \underline{o}^t & 1 \end{bmatrix} \begin{bmatrix} 1 & \underline{o}^t \\ \underline{o} & Q_{p-1}(n-1) \end{bmatrix} B(n) \begin{bmatrix} x(1) & \underline{o}^t & y(1) & 0 \\ \underline{y}_f(n) & X_{p-1}(n-1) & \underline{\Psi}(n) & \underline{\pi}_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} R_p(n) & \underline{\mu}_p(n) & a_p(n) \\ O & \underline{y}_p(n) & \underline{g}_p(n) \end{bmatrix} \tag{74}$$

As before, because certain matrices can be constructed directly, given the solutions to the (p-1)st order problems, only the rotations summarised in equation (50) need actually be performed i.e.

$$\hat{Q}_{b,p}(n)\begin{bmatrix} \beta\varepsilon_{f,p-1}(n-1) & \varrho^t & \beta\mu_{\psi,p-1}(n-1) & 0 \\ \underline{\mu}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{\psi,p-1}(n) & \underline{a}_{p-1}(n-1) \\ \varrho & O & \beta\underline{\lambda}_{\psi,p-1}(n-1) & \varrho \\ \alpha_{f,p-1}(n) & \varrho^t & \alpha_{\psi,p-1}(n) & \gamma_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \varepsilon_{f,p-1}(n) & \varrho^t & \mu_{\psi,p-1}(n) & \varphi_p(n) \\ \underline{\mu}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{\psi,p-1}(n) & \underline{a}_{p-1}(n-1) \\ \varrho & O & \beta\underline{\lambda}_{\psi,p-1}(n-1) & \varrho \\ 0 & \varrho^t & \alpha_p(n) & \gamma_p(n) \end{bmatrix} \tag{75}$$

Note that we have had to introduce some extra quantities similar to those introduced, in equation (49), for the backward problem. It should now be clear that if we have available the auxiliary angle-normalised residual $\alpha_{\psi,p-1}(n)$, then we can use the matrix $\hat{Q}_{b,p}(n)$ to calculate the adaptive filtering residual $\alpha_p(n)$.

All that remains is to show how the auxiliary adaptive filtering residual $\alpha_{\psi,p-1}(n)$ can be calculated in an order recursive manner. Due to the fact that $\alpha_{\psi,p-1}(n)$ is just an angle-normalised adaptive filtering residual, albeit based on the sequence $\underline{\psi}(n)$, it can be calculated along with the forward linear prediction residual just as $\alpha_p(n-1)$ was (see section 4.2 but with $\underline{y}(n-1)$ replaced by $\underline{\psi}(n)$). We then find the following order update (cf. equation (41)):

$$\hat{Q}_{f,p}(n)\begin{bmatrix} \underline{\mu}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{\psi,p-1}(n) & \underline{a}_{p-1}(n-1) \\ \beta\mu_{f,p-1}(n-1) & \varrho^t & \beta\varepsilon_{b,p-1}(n-2) & \beta\mu_{\psi,p-1}(n-1) & 0 \\ \beta\underline{\lambda}_{f,p-1}(n-1) & O & \varrho & \beta\underline{\lambda}_{\psi,p-1}(n-1) & \varrho \\ \alpha_{f,p-1}(n) & \varrho^t & \alpha_{b,p-1}(n-1) & \alpha_{\psi,p-1}(n) & \gamma_{p-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} \underline{\mu}_{f,p-1}(n) & R_{p-1}(n-1) & \underline{\mu}_{b,p-1}(n-1) & \underline{\mu}_{\psi,p-1}(n) & \underline{a}_{p-1}(n-1) \\ \mu_{f,p-1}(n) & \varrho^t & \varepsilon_{b,p-1}(n-1) & \mu_{\psi,p-1}(n) & \kappa_p(n-1) \\ \beta\underline{\lambda}_{f,p-1}(n-1) & O & \varrho & \beta\underline{\lambda}_{\psi,p-1}(n-1) & \varrho \\ \alpha_{f,p}(n) & \varrho^t & 0 & \alpha_{\psi,p}(n) & \gamma_p(n-1) \end{bmatrix} \tag{76}$$

The resulting architecture is shown in Figure 15: the PE's can be either those shown in Figure 5 or Figure 6 depending on the type of Givens rotation method preferred.

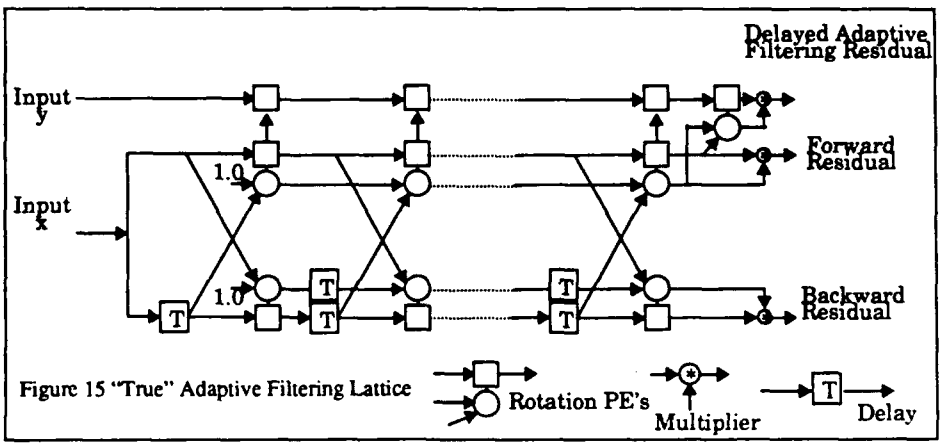


Figure 15 "True" Adaptive Filtering Lattice

10.2 Givens Rotations

As mentioned in section 5, there are two ways of calculating the parameters of a Givens rotation: with and without a square-root operation. In this section the basic mathematics is sketched out.

A Givens rotation is a planar rotation that annihilates one component of a 2-D vector. Assuming complex quantities, let

$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} \beta\varepsilon(n-1) \\ \alpha(n-1) \end{bmatrix} = \begin{bmatrix} \varepsilon(n) \\ 0 \end{bmatrix} \tag{77}$$

where, without loss of generality, c is real and

$$c\,\beta\varepsilon(n\text{-}1) + s^*\,\alpha(n\text{-}1) = \varepsilon(n) \tag{78}$$

$$c\,\alpha(n\text{-}1) - s\,\beta\varepsilon(n\text{-}1) = 0 \tag{79}$$

$$c^2 + |s|^2 = 1 \tag{80}$$

now from equation (79) $$s = c\,\alpha(n\text{-}1) / \beta\varepsilon(n\text{-}1) \tag{81}$$

then from equation (81), assuming ε to be real:

$$c^2\,(1 + |\alpha(n\text{-}1)|^2 / (\beta\,\varepsilon(n\text{-}1))^2) = 1 \tag{82}$$

hence $$c = \beta\,\varepsilon(n\text{-}1) / \sqrt{(\beta\varepsilon(n-1))^2 + |\alpha(n-1)|^2} \tag{83}$$

$$s = \alpha(n\text{-}1) / \sqrt{(\beta\varepsilon(n-1))^2 + |\alpha(n-1)|^2} \tag{84}$$

Note that $$\varepsilon(n) = \sqrt{(\beta\varepsilon(n-1))^2 + |\alpha(n-1)|^2} \tag{85}$$

In order to avoid calculating a square-root, we factorise the upper triangular matrix (R say) as follows:

$$R = D^{1/2}\bar{R} \tag{86}$$

where
$$D = \begin{bmatrix} r_{11}^2 & 0 & \dots & 0 \\ 0 & r_{22}^2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & r_{pp}^2 \end{bmatrix} \tag{87}$$

It is also necessary to remove the factor of $D^{1/2}$ from any other quantities that are operated upon by the Q matrix. Hence let

$$\begin{bmatrix} \beta\varepsilon(n-1) & \beta\mu(n-1) & 0 \\ \alpha_f(n) & \alpha_{p-1}(n) & \gamma_{p-1}(n) \end{bmatrix} = \begin{bmatrix} \beta\sqrt{\bar{\varepsilon}(n-1)} & 0 \\ 0 & \sqrt{\delta_{p-1}(n)} \end{bmatrix} \begin{bmatrix} 1 & \bar{\mu}(n-1) & 0 \\ \bar{\alpha}_f(n) & \bar{\alpha}_{p-1}(n) & 1 \end{bmatrix} \tag{88}$$

so that
$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} \beta\sqrt{\bar{\varepsilon}(n-1)} & 0 \\ 0 & \sqrt{\delta_{p-1}(n)} \end{bmatrix} \begin{bmatrix} 1 & \bar{\mu}(n-1) & 0 \\ \bar{\alpha}_f(n) & \bar{\alpha}_{p-1}(n) & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{\bar{\varepsilon}(n)} & 0 \\ 0 & \sqrt{\delta_p(n)} \end{bmatrix} \begin{bmatrix} 1 & \bar{\mu}(n) & \bullet \\ 0 & \bar{\alpha}_p(n) & 1 \end{bmatrix} \tag{89}$$

(where • represents a quantity of no interest). Thus we find that

$$\bar{\varepsilon}(n) = \beta^2\bar{\varepsilon}(n-1) + \delta_{p-1}(n)\,|\bar{\alpha}_f(n)|^2 \tag{90}$$

$$\bar{\mu}(n) = (\beta^2\bar{\varepsilon}(n-1)\bar{\mu}(n-1) + \delta_{p-1}(n)\bar{\alpha}_f^*(n)\bar{\alpha}_{p-1}(n))/\bar{\varepsilon}(n) \tag{91}$$

$$= \bar{c}\bar{\mu}(n-1) + \bar{s}^*\bar{\alpha}_{p-1}(n) \tag{92}$$

where
$$\bar{c} = (\beta^2\bar{\varepsilon}(n-1))/\bar{\varepsilon}(n) \text{ and } \bar{s} = (\delta_{p-1}(n)\bar{\alpha}_f(n))/\bar{\varepsilon}(n) \tag{93}$$

Similarly
$$\bar{\alpha}_p(n) = \bar{\alpha}_{p-1}(n) - \bar{\alpha}_f(n)\bar{\mu}(n-1) \tag{94}$$

using the fact that
$$\delta_p(n) = \frac{\beta^2\bar{\varepsilon}(n-1)\delta_{p-1}(n)}{\bar{\varepsilon}(n)} = \bar{c}\delta_{p-1}(n). \tag{95}$$

From equations (90) and (93) note that

$$\bar{c} + \bar{\alpha}_f(n)\bar{s}^* = 1 \tag{96}$$

and hence, from equations (92) and (94), that

$$\bar{\mu}(n) = \bar{\mu}(n-1) + \bar{s}^* \bar{\alpha}_p(n) \tag{97}$$

This latter form (equation (97)) of the update for the quantity $\bar{\mu}(n)$ is the most stable method for implementing the square-root-free algorithm and is intimately connected with the "error-feedback" algorithm of Ling and Proakis[11].

10.3. Algorithms

These algorithms, written in pseudo-ALGOL, calculate the adaptive filter residual for a p-th order system fed with a pre-windowed data sequence x(n) and a reference sequence y(n). The first algorithm uses the obvious implementation of Givens rotations using square-roots. The second one avoids taking square-roots by calculating transformed quantities and implements the rotations via the feedback algorithm.

10.3.1. "Normal" algorithm

START
  INITIALISE {all variables} := 0;
  FOR n FROM 1 DO
    LET $\alpha_{f,0}(n) := x(n)$; $\alpha_{b,0}(n-1) := x(n-1)$; $\alpha_0(n-1) := y(n-1)$; $\gamma_0(n-1) := 1$;
    FOR q FROM 1 TO p DO
      LET $\varepsilon_{b,q-1}(n-1) := \sqrt{(\beta\varepsilon_{b,q-1}(n-2))^2 + |\alpha_{b,q-1}(n-1)|^2}$;
      IF $\varepsilon_{b,q-1}(n-1) = 0$ THEN LET $c_{f,q} := 1$; $s_{f,q} := 0$
      ELSE LET $c_{f,q} := \beta\varepsilon_{b,q-1}(n-2) / \varepsilon_{b,q-1}(n-1)$; $s_{f,q} := \alpha_{b,q-1}(n-1) / \varepsilon_{b,q-1}(n-1)$
      END_IF;
      LET $\mu_{f,q-1}(n) := c_{f,q}\,\beta\mu_{f,q-1}(n-1) + s^{*}_{f,q}\alpha_{f,q-1}(n)$;
        $\alpha_{f,q}(n) := c_{f,q}\,\alpha_{f,q-1}(n) - s_{f,q}\,\beta\mu_{f,q-1}(n-1)$;
        $\mu_{q-1}(n-1) := c_{f,q}\,\beta\mu_{q-1}(n-2) + s^{*}_{f,q}\alpha_{q-1}(n-1)$;
        $\alpha_q(n-1) := c_{f,q}\,\alpha_{q-1}(n-1) - s_{f,q}\,\beta\mu_{q-1}(n-2)$;
        $\gamma_q(n-1) := c_{f,q}\,\gamma_{q-1}(n-1)$;
COMMENT prediction residual $e_{f,p}(n,n) = \gamma_q(n-1)\,\alpha_{f,q}(n)$ COMMENT
        $e_p(n-1,n-1) = \gamma_q(n-1)\,\alpha_q(n-1)$ COMMENT q-th order filtered residual COMMENT
      LET $\varepsilon_{f,q-1}(n) := \sqrt{(\beta\varepsilon_{f,q-1}(n-1))^2 + |\alpha_{f,q-1}(n)|^2}$;
      IF $\varepsilon_{f,q-1}(n) = 0$ THEN LET $c_{b,q} := 1$; $s_{b,q} := 0$
      ELSE LET $c_{b,q} := \beta\varepsilon_{f,q-1}(n-1) / \varepsilon_{f,q-1}(n)$; $s_{b,q} := \alpha_{f,q-1}(n) / \varepsilon_{f,q-1}(n)$
      END_IF;
      LET $\mu_{b,q-1}(n-1) := c_{b,q}\,\beta\mu_{b,q-1}(n-2) + s^{*}_{b,q}\alpha_{b,q-1}(n-1)$;
        $\alpha_{b,q}(n) := c_{b,q}\,\alpha_{b,q-1}(n-1) - s_{b,q}\,\beta\mu_{b,q-1}(n-2)$;
COMMENT $\gamma_q(n) := c_{b,q}\,\gamma_{q-1}(n-1)$; backward prediction residual $e_{b,p}(n,n) := \gamma_q(n)\,\alpha_{b,q}(n)$
COMMENT
    END_DO
  END_DO
FINISH

10.3.2. "Square-root-free with feedback" algorithm

START
  INITIALISE {all variables} := 0;
  FOR n FROM 1 DO
    LET $e_{f,0}(n,n-1) := x(n)$; $e_{b,0}(n-1,n-2) := x(n-1)$; $e_0(n-1,n-2) := y(n-1)$; $\delta_0(n-1) := 1$;
    FOR q FROM 1 TO p DO

      LET $\bar{\varepsilon}_{b,q-1}(n-1) := \beta^2 \bar{\varepsilon}_{b,q-1}(n-2) + \delta_{q-1}(n-1)\, |e_{b,q-1}(n-1,n-2)|^2$;

      IF $\bar{\varepsilon}_{b,q-1}(n-1) = 0$ THEN LET $\bar{c}_{f,q} := 1$; $\bar{s}_{f,q} := 0$

      ELSE LET $\bar{c}_{f,q} := \beta^2 \bar{\varepsilon}_{b,q-1}(n-2) / \bar{\varepsilon}_{b,q-1}(n-1)$; $\bar{s}_{f,q} := \delta_{q-1}(n-1)\, e_{b,q-1}(n-1,n-2) / \bar{\varepsilon}_{b,q-1}(n-1)$
      END_IF;
      LET $e_{f,q}(n,n-1) := e_{f,q-1}(n,n-1) - e_{b,q-1}(n-1,n-2)\, \bar{\mu}_{f,q-1}(n-1)$;

        $\bar{\mu}_{f,q-1}(n) := \bar{\mu}_{f,q-1}(n-1) + \bar{s}^*_{f,q} e_{f,q}(n,n-1)$;
        $e_q(n-1,n-2) := e_{q-1}(n-1,n-2) - e_{b,q-1}(n-1,n-2)\, \bar{\mu}_{q-1}(n-2)$;
        $\bar{\mu}_{q-1}(n-1) := \bar{\mu}_{q-1}(n-1) + \bar{s}^*_{f,q} e_q(n-1,n-2)$;
        $\delta_q(n-1) := \bar{c}_{f,q}\, \delta_{q-1}(n-1)$;
COMMENT prediction residual $e_{f,p}(n,n) = \delta_q(n-1)\, e_{f,q}(n,n-1)$ COMMENT
      $e_q(n-1,n-1) := \delta_q(n-1)\, e_q(n-1,n-2)$ COMMENT q-th order filtered residual COMMENT

      LET $\bar{\varepsilon}_{f,q-1}(n) := \beta^2 \bar{\varepsilon}_{f,q-1}(n-1) + \delta_{q-1}(n-1)\, |e_{f,q-1}(n,n-1)|^2$;
      IF $\bar{\varepsilon}_{f,q-1}(n) = 0$ THEN LET $\bar{c}_{b,q} := 1$; $\bar{s}_{b,q} := 0$

      ELSE LET $\bar{c}_{b,q} := \beta^2 \bar{\varepsilon}_{f,q-1}(n-1) / \bar{\varepsilon}_{f,q-1}(n)$; $\bar{s}_{b,q} := \delta_{q-1}(n-1)\, e_{f,q-1}(n,n-1) / \bar{\varepsilon}_{f,q-1}(n)$
      END_IF;
      LET $e_{b,q}(n,n-1) := e_{b,q-1}(n-1,n-2) - e_{f,q-1}(n,n-1)\, \bar{\mu}_{b,q-1}(n-2)$;

      $\bar{\mu}_{b,q-1}(n-1) := \bar{\mu}_{b,q-1}(n-2) + \bar{s}^*_{b,q} e_{b,q-1}(n,n-1)$;
COMMENT $\delta_q(n) := \bar{c}_{b,q}\, \delta_{q-1}(n-1)$; backward prediction residual $e_{b,p}(n,n) := \delta_q(n)\, e_{b,q}(n,n-1)$
COMMENT
    END_DO
  END_DO
FINISH

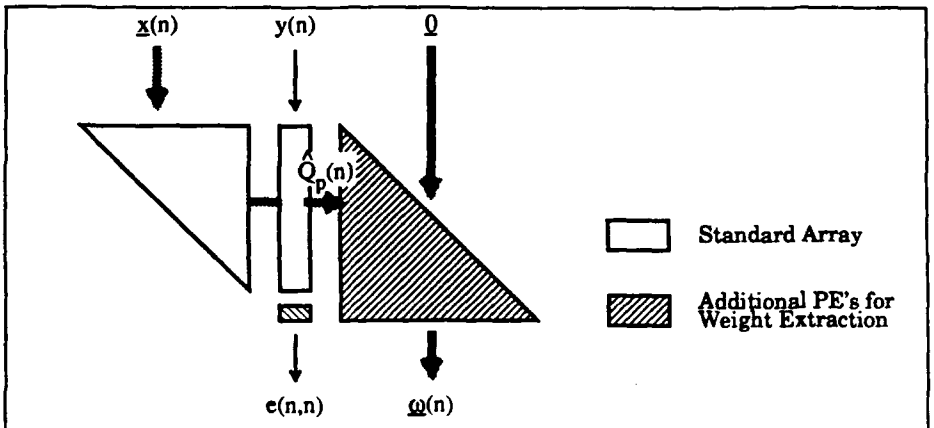


Figure 16 Parallel Weight Extraction

10.4 Post-processor Architectures

The algorithm presented in this memorandum solves a canonical adaptive filtering problem where the reference signal (y(n)) is known. There are situations, however, when no such reference signal is available and a *constrained* least squares minimisation is required (e.g. wide-band MVDR beamforming). The algorithm also calculates the adaptive filtering residual directly without finding the optimum coefficients. It is useful in system identification problems to know what the optimum coefficients are and as such the algorithm derived so far cannot be used. All of the above draw-backs also apply to the triangular systolic array [28] and a series of pre- and post-processor architectures have been developed to over come them [15][25][26][30]. It is of great interest, therefore, to consider the application of these techniques to enhance the lattice algorithm presented here.

10.4.1 Parallel Weight Extraction

It has been shown [26] that it is possible to extract the optimum weight vector (e.g. $\underline{\omega}$ in equation (4)) in parallel with the computation of the standard triangular systolic array (Figure 16). The rotation parameters ($\hat{Q}_p(n)$) are passed across into an "inverted" triangular array of processors that apply these rotations. The data that is fed into this array is a vector of zeros. As this vector passes down the array it is transformed into the optimum weight vector.

Now if the data $\underline{x}(n)$ is in fact delayed samples from a time series, then the system is an adaptive filter and the left-hand triangle in Figure 16 can be replaced by a lattice filter. Note however that the rotation parameters ($\hat{Q}_p(n)$) are still produced by the lattice. This can be seen as follows: consider the series of Givens rotations that constitute the matrix $\hat{Q}_p(n)$. Recall that these operations are intended to annihilate the new data vector by rotation against the previous version of the triangular matrix $R_p(n-1)$ (see equations (22) and (23)):

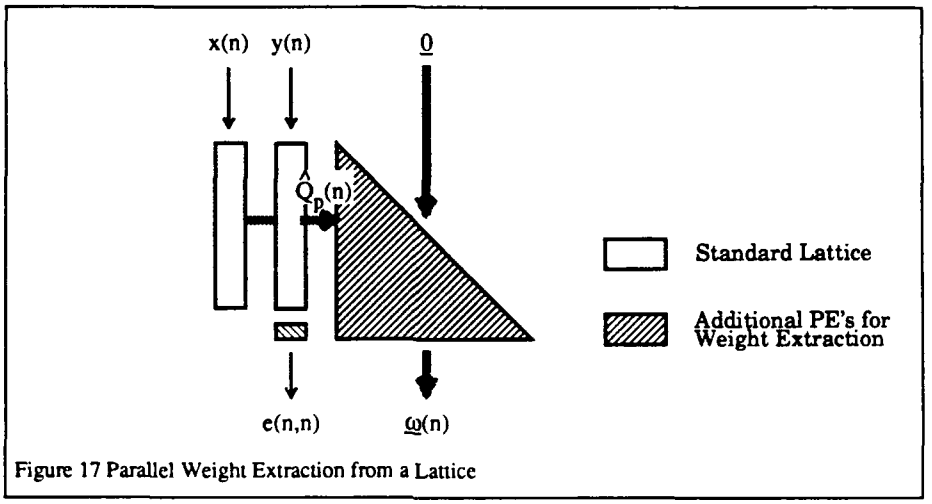


Figure 17 Parallel Weight Extraction from a Lattice

$$\hat{Q}_p(n)\begin{bmatrix} \beta R_p(n-1) \\ O \\ x(n) \ldots x(n-p+1) \end{bmatrix} = \begin{bmatrix} R_p(n) \\ O \end{bmatrix} \tag{98}$$

The sequence of Givens rotations are constructed [5] such that the element $x(n)$ is annihilated first, by rotation against the top row of the matrix $\beta R_p(n-1)$, next the second element of the bottom row ($x(n-1)$) is annihilated by rotation against the second row of $\beta R_p(n-1)$ and so on. Consider the situation after the first $q < p$ (say) elements of the bottom row have been annihilated:

$$Q_q Q_{q-1} \cdots Q_1 \begin{bmatrix} Q_p(n-1) & \varrho \\ \varrho^t & 1 \end{bmatrix} B(n) X_p(n) = \begin{bmatrix} R_q(n) & \underline{y}_{b,q}(n) & S \\ \varrho^t & \beta e_{b,q}(n-1) & \underline{r} \\ O & \varrho & T \\ O & \varrho & O \\ \varrho^t & \alpha_{b,q}(n) & \underline{z}^t \end{bmatrix} \tag{99}$$

where Q_n is a Givens rotation and we have used the fact that the $(q+1)$st diagonal element of $R_p(n-1)$ is $e_{b,q}(n-1)$ (see last paragraph of section 4.2). The identity of the elements $\underline{y}_{b,q}(n)$ and $\alpha_{b,q}(n)$ follows from the underlying data matrix and the fact that upper-triangular matrix $R_q(n)$ has been formed. It is clear that the Givens rotation Q_{q+1} is that which annihilates the element $\alpha_{b,q}(n)$ by rotation against $\beta e_{b,q}(n-1)$ but this is exactly the definition of $Q_{f,q+1}(n+1)$ (see equation (41)). Thus the lattice algorithm *does* calculate the rotations that constitute $\hat{Q}_p(n)$ and it is possible to "bolt-on" the weight-extraction processor to the lattice (Figure 17).

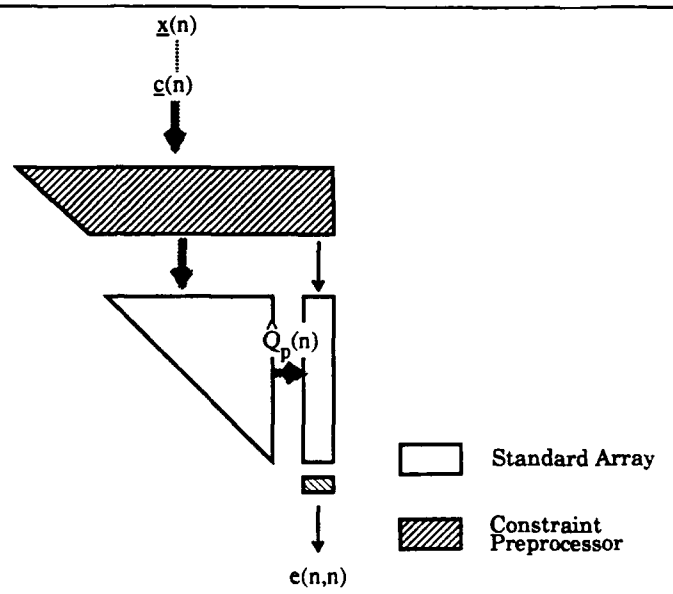


Figure 18 Linearly Constrained Least Squares Minimisation

10.4.2 Linear Constraints

It has been shown [25] how the triangular systolic array can be adapted to solve a linearly constrained least squares problem. The technique is to use a trapezoidal preprocessor in front of the triangular array (Figure 18). Briefly, system works as follows: the constraint vectors are first fed into the combined array followed by the data in the usual way (note that the combined array is triangular). After the constraint vectors have been entered into the array, the preprocessor section is "frozen", that is to say the stored parameters that are normally updated from time instant to time instant are kept constant. As the data subsequently passes through the preprocessor, the constraints are incorporated into the data *before the least squares minimisation is performed in the standard triangular array.*

It would be very useful to be able to solve linearly constrained adaptive filtering problems efficiently. However, when a constrained least squares minimisation problem is re-formulated as a canonical one (see [25]) the data matrix does not have Toeplitz symmetry (even if the input data does). Thus it is not possible to "collapse" the triangular array into a lattice. It would be interesting to know what function the lattice equivalent to the architecture shown in Figure 18 performs. By equivalent architecture we mean a lattice where the first q (say) lattice stages were operated frozen mode once the constraint data had been fed into them.

The pre-processor architecture for implementing the linear constraints has many desirable features however it is not the only method for achieving this aim [30]. It is possible to apply constraints to the least squares minimisation problem using a post-processor: this architecture is a modification of the systolic QRD-based MVDR beamforming method [15].

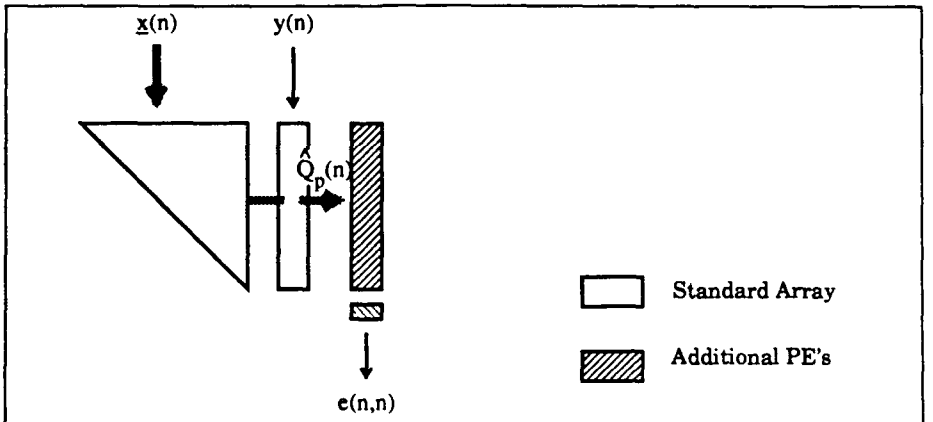


Figure 19 MVDR Beamforming

The MVDR beamforming problem is a least squares minimisation problem with a single linear constraint: namely that the antenna gain in the "look" direction is fixed. The QRD-based architecture shown in Figure 19 can be used to produce a MVDR residual. The triangular array is fed with data to initialise it, then the constraint vector is fed into the array which, operating in the frozen mode, outputs a vector that is loaded in to the new array on the right hand side. The triangular array is then return to its normal adaptive mode and, as more data arrives, it passes the rotation information $\hat{Q}_p(n)$ to the post-processor. The output of the combined structure is the MVDR beamformer residual.

Clearly because the QRD-based lattice is "black box" equivalent to a triangular array fed by a tapped delay line, we may use the same technique to implement a MVDR spectrum analyser. The main advantage to this post-processor MVDR technique is that several single constraints, or look directions, can be implemented simultaneously since the rotation parameters in $\hat{Q}_p(n)$ can be fed to any number of post-processors. The main draw-back is that the magnitude of the numbers stored in the post-processor continually diminishes as time progresses and the processor requires periodic re-initialisation (as above) if serious numerical problems are to be avoided.

Yang and Böhme [30] have shown that by combining the outputs of several different MVDR problems, the solution to a multiply constrained problem can be generated (Figure 20). Apart from the problem with the MVDR post-processor mentioned above, this technique has the draw-back that the final trapezoidal array has to implement a hyperbolic rotation which is numerically unstable.

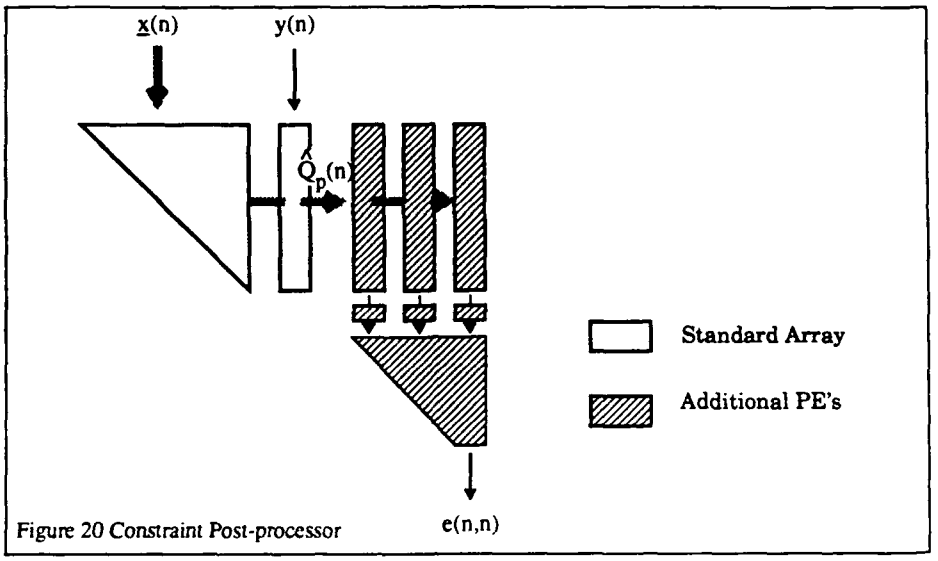


Figure 20 Constraint Post-processor

11. ANNEX

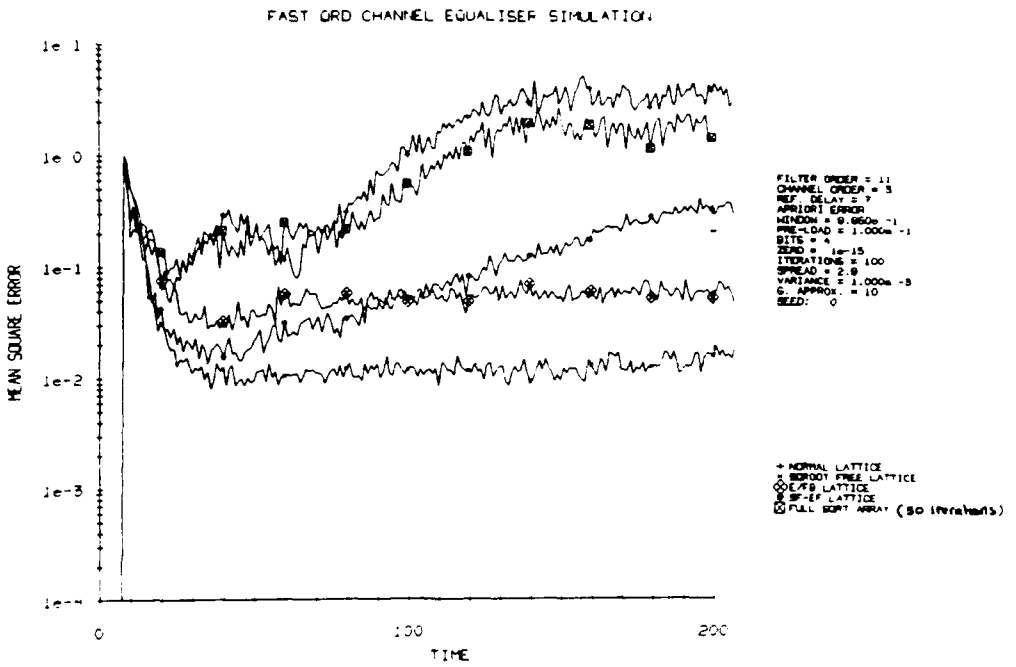
As was mentioned in section 7, many computer simulation experiments were undertaken and only the salient points have been discussed so far. For the sake of completeness, all of the simulation results run to date are included in this annex.

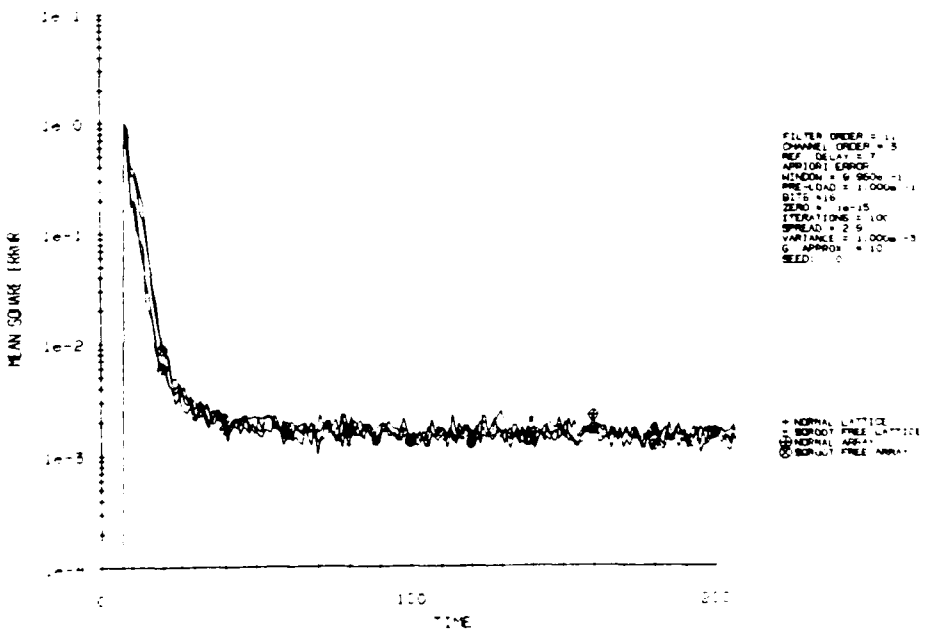
Each of the following graphs has a key which lists all the relevant parameters. The following is a short explanation of each item:

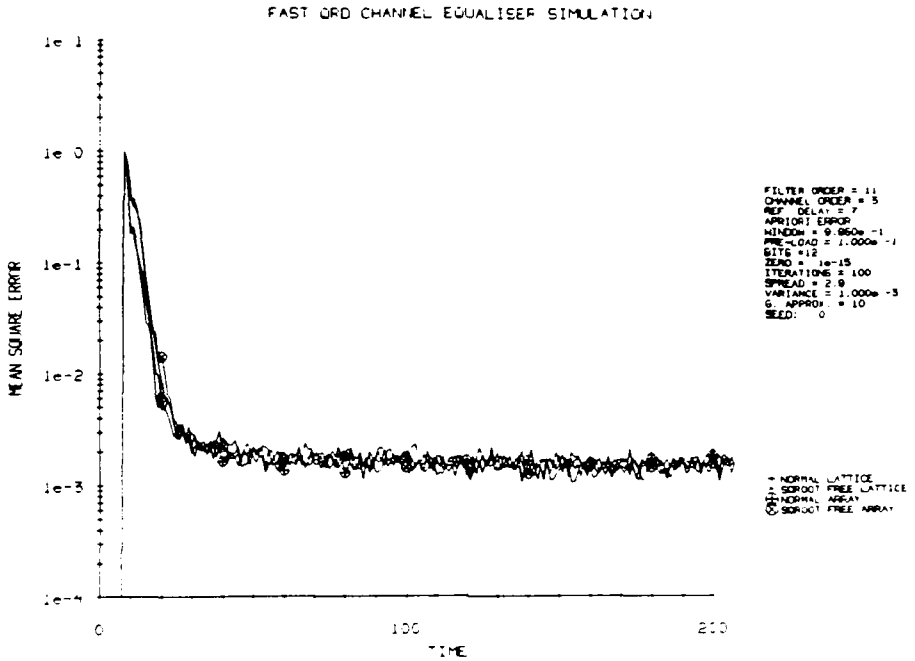
FILTER ORDER..... the order of the adaptive filter (always = 11).
CHANNEL ORDER..... the order of the "raised cosine" channel (always = 3).
REF. DELAY ..... the amount of delay in the reference signal path (always = 7).
APRIORI ERROR..... confirmation of the type of error being plotted.
WINDOW ..... value of β (see equation (2)).
PRE-LOAD..... value used to initialise energy terms.
BITS ..... number of bits in the mantissa (0 = double precision).
ZERO..... value below which quantities are considered to be effectively zero.
ITERATIONS ..... number of realisations of the simulation used to calculate the ensemble-average error.
SPREAD ..... value of W (see equation (71)).
VARIANCE ..... variance of gaussian measurement noise (always = 10^{-3}).
G. APPROX ..... number of iid random variables added together to form a gaussian random variable (central limit theorem!).
SEED..... seed value for random number generator.

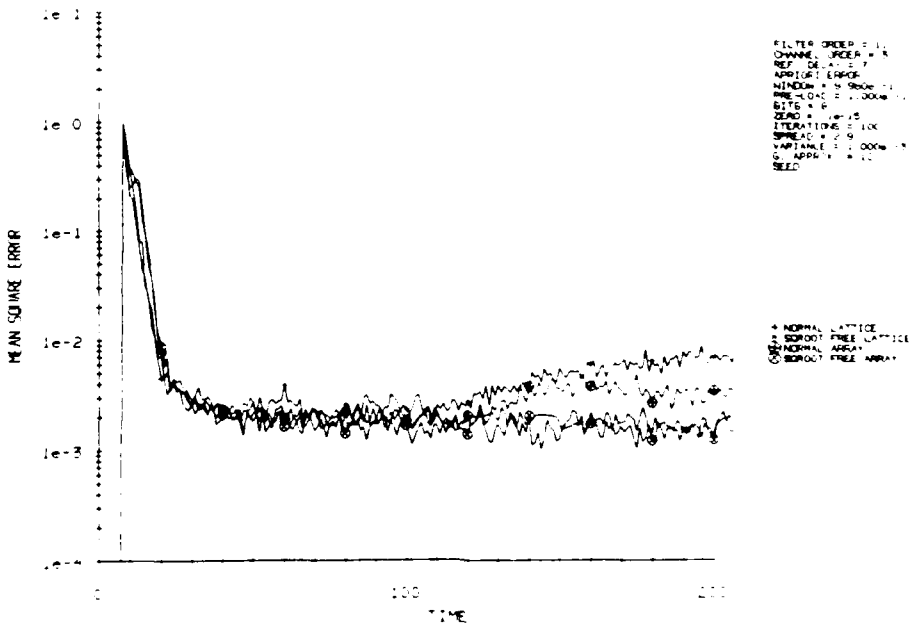
Several different algorithms were used in the simulations and each one was allocated a different tag:

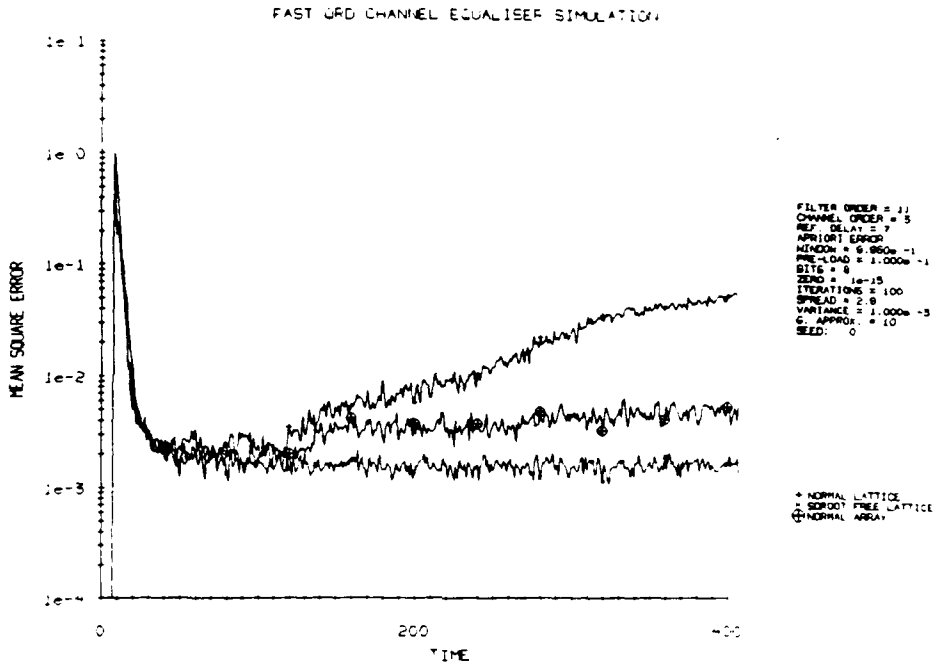
NORMAL LATTICE ..... Lattice algorithm, normal Givens rotations.
SQROOT LATTICE..... Lattice algorithm, square-root-free with feedback Givens rotations.
E/FB LATTICE..... Lattice algorithm, square-root with feedback Givens rotations.
SF-EF LATTICE..... Lattice algorithm, square-root-free without feedback Givens rotations.
FULL SQRT ARRAY..... Lattice algorithm (sic), normal Givens rotations but with double precision arithmetic inside the square-root operator.
NORMAL ARRAY..... Triangular systolic array algorithm, normal Givens rotations.
SQROOT ARRAY ..... Triangular systolic array algorithm, square-root-free with feedback Givens rotations.

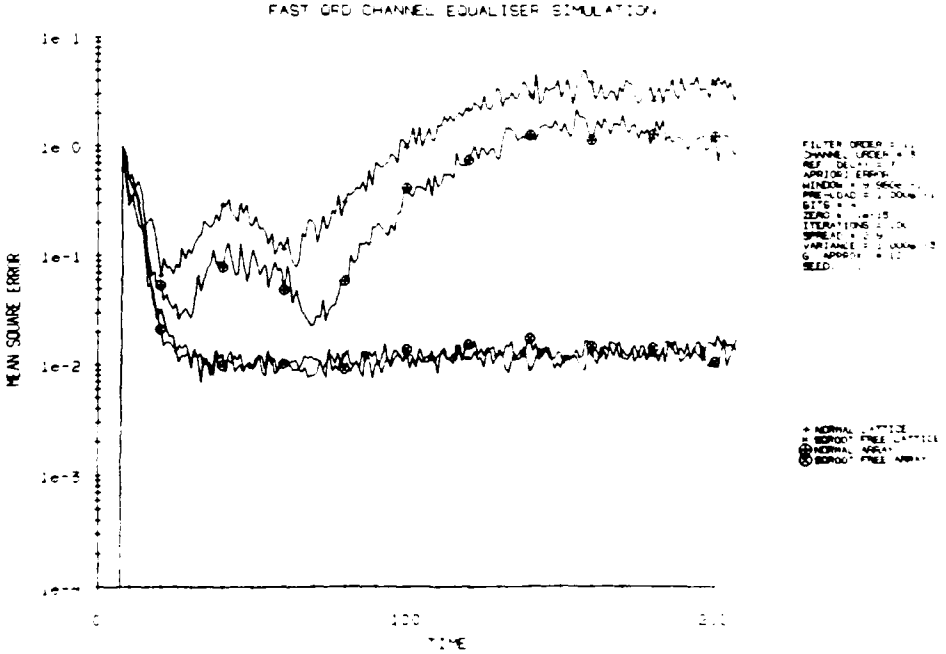

FAST QRD CHANNEL EQUALISER SIMULATION
MEAN SQUARE ERROR
1e 1
1e 0
1e-1
1e-2
1e-3
0
100
200
TIME
FILTER ORDER = 11
CHANNEL ORDER = 3
REF. DELAY = 7
APRIORI ERROR
WINDOW = 9.950e-1
PRE-LOAD = 1.000e-1
BITS = 4
ZERO = 1e-15
ITERATIONS = 100
SPREAD = 2.0
VARIANCE = 1.000e-3
G. APPROX. = 10
SEED: 0
+ NORMAL LATTICE
BORDOT FREE LATTICE
E/F/B LATTICE
SF-EF LATTICE
FULL SQRT ARRAY (SO Inrahams)

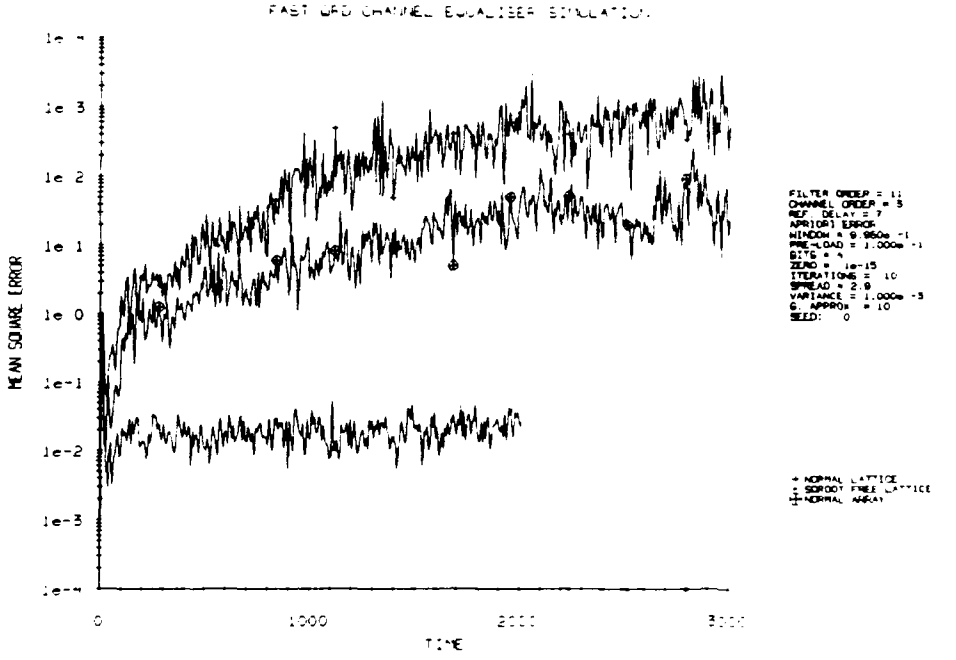

FAST QRD CHANNEL EQUALISER SIMULATION
MEAN SQUARE ERROR
1e 0
1e-1
1e-2
1e-3
TIME
FILTER ORDER = 11
CHANNEL ORDER = 3
REF. DELAY = 7
APRIORI ERROR
BITS = 16
ZERO = 1e-15
ITERATIONS = 100
SPREAD = 2.0
G. APPROX = 10
SEED: 0
+ NORMAL LATTICE
BORDOT FREE LATTICE

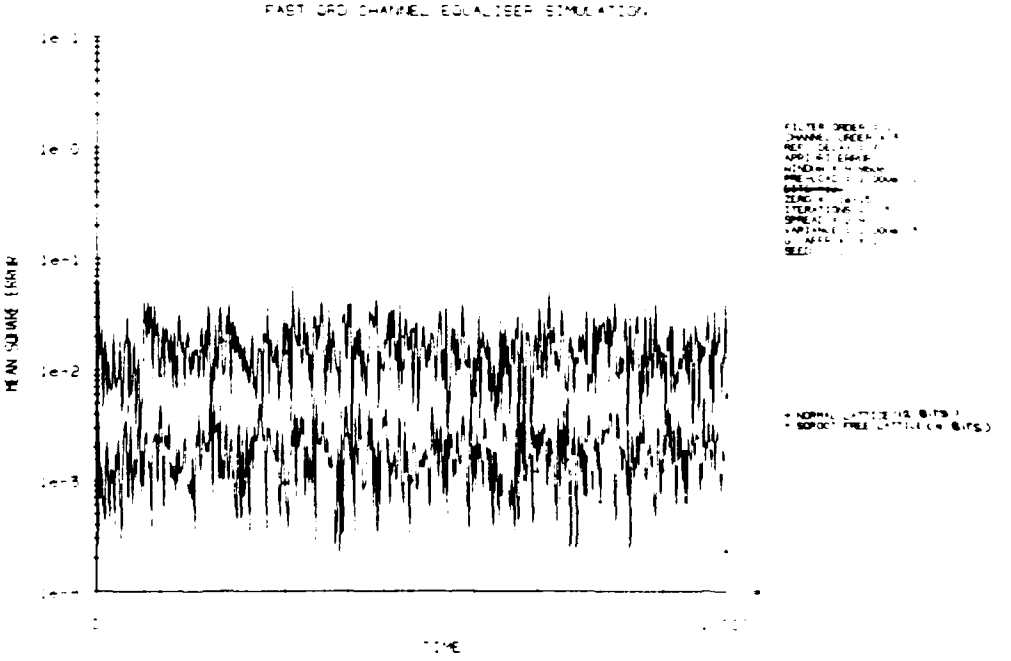

FAST QRD CHANNEL EQUALISER SIMULATION
MEAN SQUARE ERROR
1e 1
1e 0
1e-1
1e-2
1e-3
1e-4
0
100
200
TIME
FILTER ORDER = 11
CHANNEL ORDER = 5
REF DELAY = 7
APRIORI ERROR
WINDOW = 9.950e -1
PRE-LOAD = 1.000e +1
BITS =12
ZERO = 1e-15
ITERATIONS = 100
SPREAD = 2.9
VARIANCE = 1.000e -5
G. APPROX = 10
SEED: 0
+ NORMAL LATTICE
SQROOT FREE LATTICE
NORMAL ARRAY
SQROOT FREE ARRAY

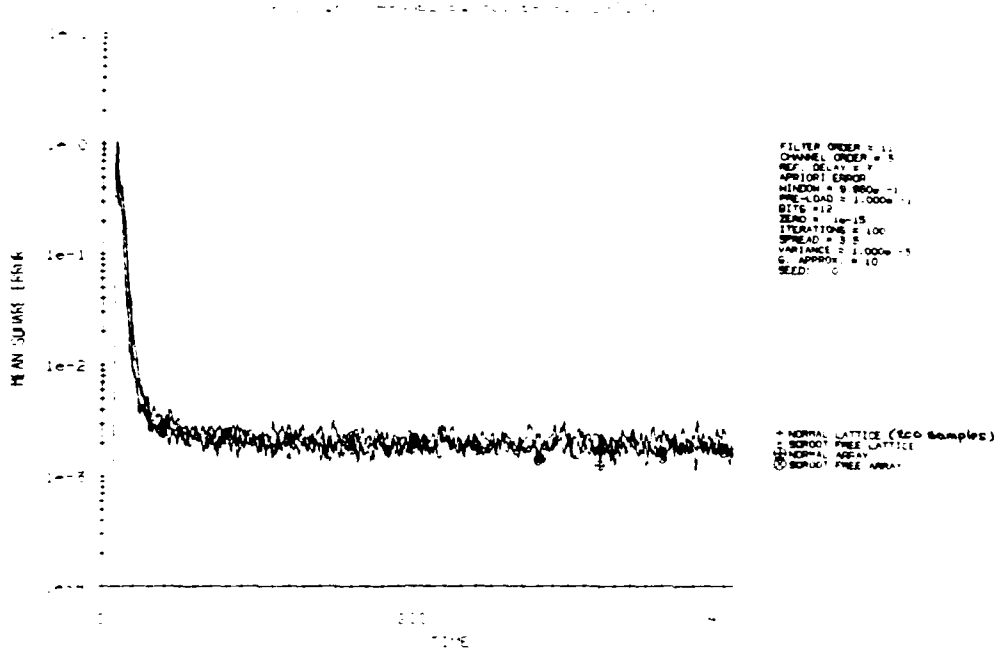

FAST QRD CHANNEL EQUALISER SIMULATION
MEAN SQUARE ERROR
1e 1
1e 0
1e-1
1e-2
1e-3
100
TIME
+ NORMAL LATTICE
SQROOT FREE LATTICE
NORMAL ARRAY
SQROOT FREE ARRAY

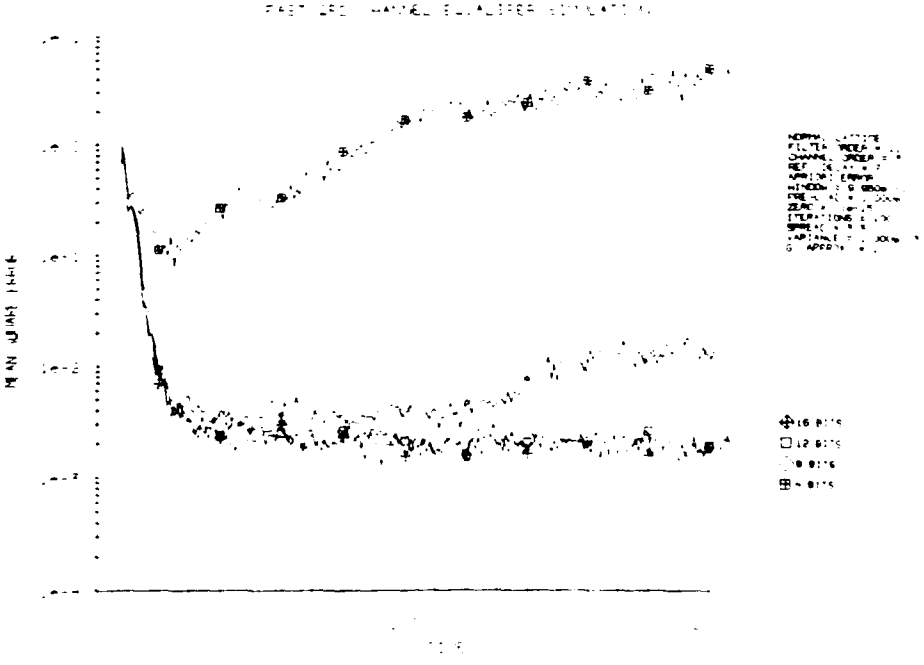

FAST QRD CHANNEL EQUALISER SIMULATION
1e 1
1e 0
1e-1
1e-2
1e-3
0
200
TIME
MEAN SQUARE ERROR
FILTER ORDER = 11
CHANNEL ORDER = 5
REF. DELAY = 7
APRIORI ERROR
WINDOW = 9.950e-1
PRE-LOAD = 1.000e-1
BITS = 8
ZERO = 1e-15
ITERATIONS = 100
SPREAD = 2.0
VARIANCE = 1.000e-3
G: APPROX. = 10
SEED: 0
+ NORMAL LATTICE
SQROOT FREE LATTICE
NORMAL ARRAY

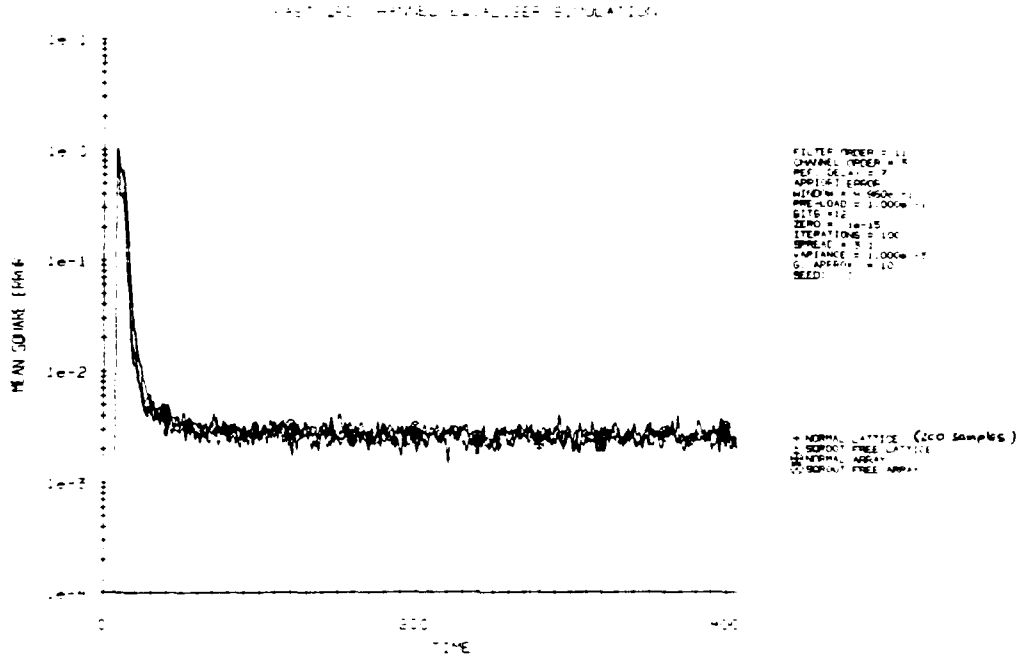

FAST QRD CHANNEL EQUALISER SIMULATION
1e 1
1e 0
1e-1
1e-2
1e-3
0
100
TIME
MEAN SQUARE ERROR

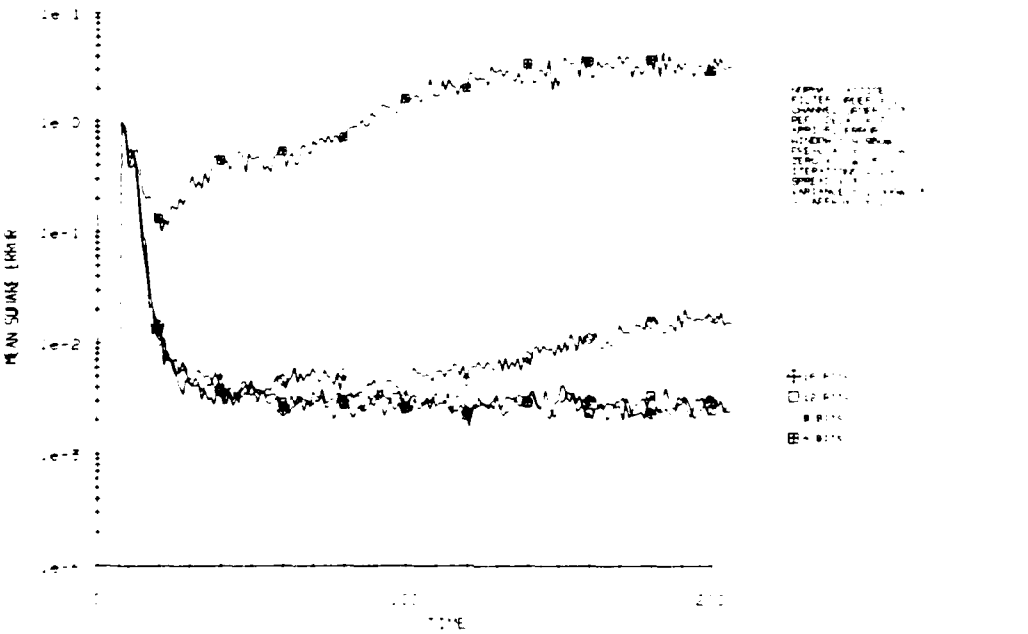

FAST GRD CHANNEL EQUALISER SIMULATION
1e 3
1e 2
1e 1
1e 0
1e-1
1e-2
1e-3
1e-4
MEAN SQUARE ERROR
0
1000
TIME
FILTER ORDER = 11
CHANNEL ORDER = 5
REF. DELAY = 7
APRIORI ERROR
WINDOW = 9.950e -1
PRE-LOAD = 1.000e +1
BITS = 4
ZERO = 1e-15
ITERATIONS = 10
SPREAD = 2.0
VARIANCE = 1.000e -5
G. APPROX = 10
SEED: 0
+ NORMAL LATTICE
* SOFDCT FREE LATTICE
NORMAL ARRAY

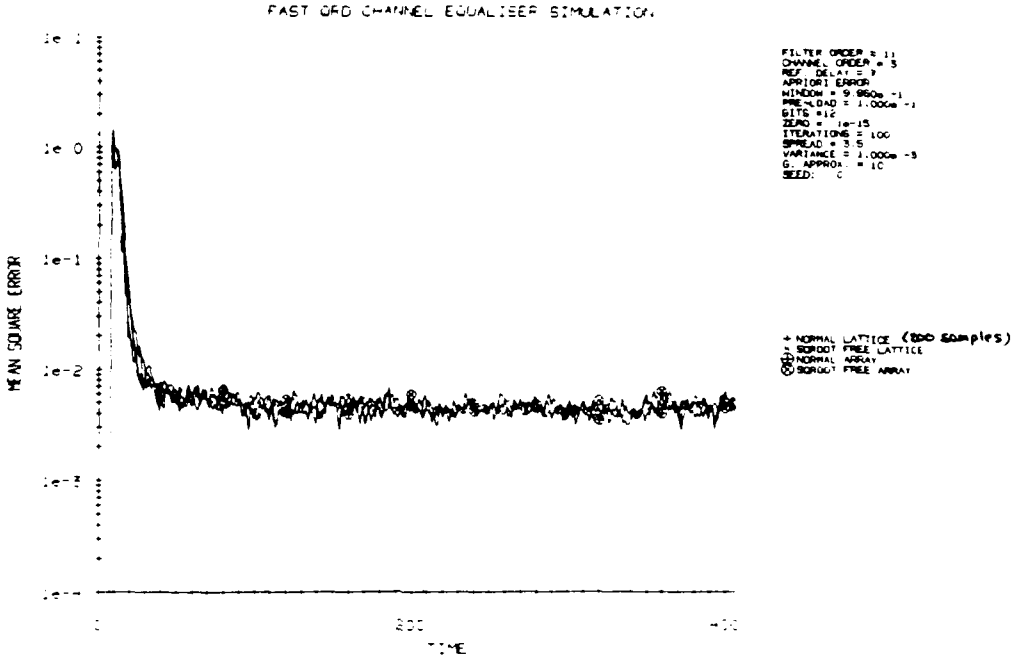

FAST GRD CHANNEL EQUALISER SIMULATION
1e 0
1e-1
1e-2
1e-3
MEAN SQUARE ERROR
TIME

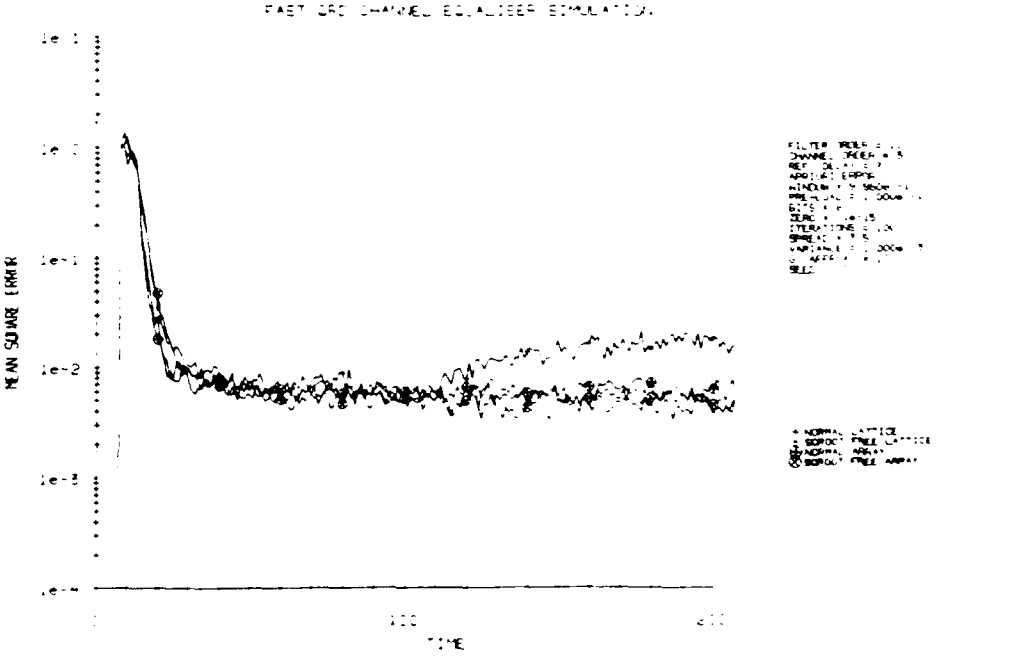

1e-1
1e-2
+ NORMAL LATTICE (500 Samples)
× SQROOT FREE LATTICE
NORMAL ARRAY
SQROOT FREE ARRAY

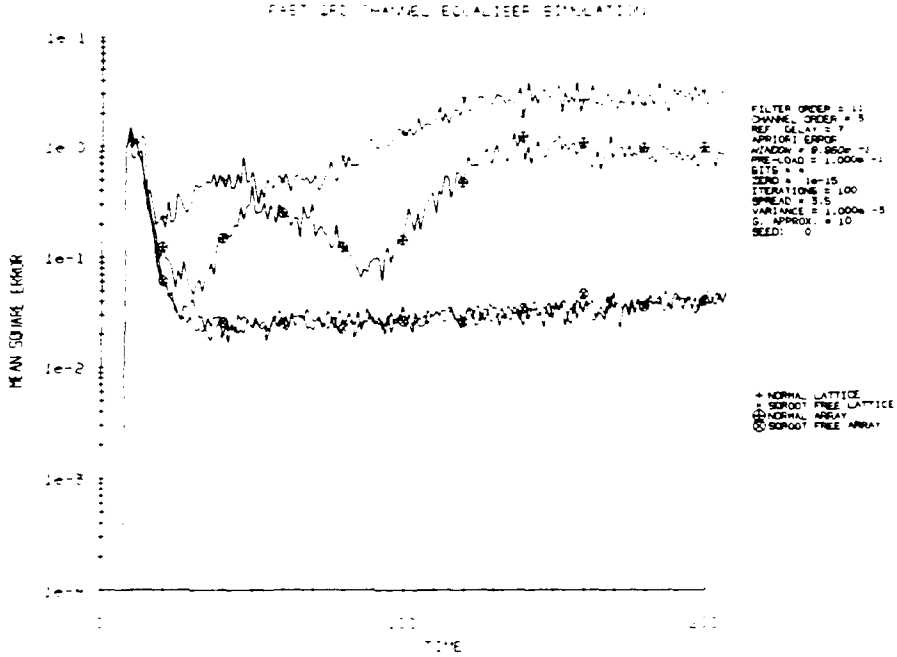

16 BITS
12 BITS


1e-1
1e-2
1e-3
TIME


1e 1
1e 0
1e-1
1e-2
1e-3
TIME


FAST QRD CHANNEL EQUALIZER SIMULATION
MEAN SQUARE ERROR
TIME
FILTER ORDER = 11
CHANNEL ORDER = 5
+ NORMAL LATTICE (800 samples)
× SQRDOT FREE LATTICE
NORMAL ARRAY
SQRDOT FREE ARRAY


FAST QRD CHANNEL EQUALIZER SIMULATION
MEAN SQUARE ERROR
TIME


MEAN SQUARE ERROR
TIME
+ NORMAL LATTICE
+ SQROOT FREE LATTICE
NORMAL ARRAY
SQROOT FREE ARRAY

REPORT DOCUMENTATION PAGE

DRIC Reference Number (if known) .....

Overall security classification of sheet .....Unclassified.....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).

Originators Reference/Report No	Month	Year
MEMO 4409	JULY	1990

Originators Name and Location
RSRE, St Andrews Road
Malvern, Worcs WR14 3PS

Monitoring Agency Name and Location

Title
THE QR DECOMPOSITION BASED LEAST-SQUARES LATTICE ALGORITHM FOR ADAPTIVE FILTERING

Report Security Classification	Title Classification (U, R, C or S)
UNCLASSIFIED	U

Foreign Language Title (in the case of translations)

Conference Details

Agency Reference	Contract Number and Period

Project Number	Other References

Authors	Pagination and Ref
PROUDLER, I K; McWHIRTER, J G	58

Abstract

We derive, from first principles, the least squares lattice algorithm for adaptive filtering based on the QR decomposition (QRD). In common with other lattice algorithms for adaptive filtering, this algorithm only requires O(p) operations for the solution of a p-th order problem. The algorithm has as its root the QRD-based recursive least squares minimisation algorithm and hence is expected to have superior numerical properties when compared with other fast algorithms.

This algorithm contains within it the QRD-based algorithm for solving the least squares linear prediction problem. These algorithms are presented in two forms: one that involves taking square-roots and one that does not. Some computer simulations of a channel equaliser, using finite-precision arithmetic, are presented in which the lattice algorithms are compared to the more established triangular systolic array ones.

The relationship between the QRD-based lattice algorithm and other least squares lattice algorithms is briefly discussed. Various extensions to this work are discussed including the multi-channel QRD-based adaptive filtering algorithm that can be used for wide-band beamforming.

Abstract Classification (U R C or S)
U

Descriptors

Distribution Statement (Enter any limitations on the distribution of the document)
UNLIMITED

THIS PAGE IS LEFT BLANK INTENTIONALLY