IMPROVED LMS ALGORITHM FOR ADAPTIVE BEAMFORMING

Lal C. Godara

Department of Electrical Engineering, University College, University of New South Wales, Australian Defence Force Academy, Northcott Drive, Campbell, ACT 2600, Australia.

1. INTRODUCTION

The application of the constrained least mean square (LMS) algorithm to adaptive beamforming and its analysis has been studied by many authors [1]-[4]. The algorithm updates the array weights employing a noisy estimate of the required gradient. In its usual form an estimate of the required gradient is made by multiplying the array output with the receiver outputs [1], [2]. This is referred to as the standard algorithm throughout this paper.

The structured gradient algorithm studied in [4], exploits the structure of the array correlation matrix (ACM) for estimating the gradient. For a linear array of equispaced receivers, immersed in a homogeneous noise field, the ACM has a Toeplitz structure, i.e. the entries along each diagonal are equal. The noisy estimate of the ACM, used in the standard algorithm to calculate the gradient, is not constrained to have this structure. The structured gradient algorithm uses an estimate of the ACM constrained to have the same structure as the exact ACM. The standard LMS algorithm and the structured gradient algorithm are useful when all the receiver outputs are accessible. In the absence of that, perturbation algorithms [3], [5] are used to estimate the required gradient by perturbing the array weights and measuring the output power of the system.

Though, all these algorithms employ different methods of gradient estimation, they all have one feature in common. At the (n+1)st iteration of the weight iteration they all use samples available after nth iteration to estimate the gradient. None of these algorithms utilise the previous samples which are available. In this paper we study two LMS algorithms which make use of the previous available samples to estimate the required gradient to update the array weights. The first algorithm, referred to as the recursive LMS algorithm, is applicable for a general array whereas the second algorithm, referred to as the improved LMS algorithm, exploits the Toeplitz structure of the ACM and can only be used for an equispaced linear array.

2. PRELIMINARY BEAMFORMING CONSIDERATION

Consider a linear array of L equispaced, omnidirectional elements immersed in the far field of sinusoidal point sources. Let an L dimensional vector $\underline{X}(n)$ represent the L outputs of the array, and an L dimensional vector $\underline{W}$ represent the weights of an element space beamformer.

Let $\hat{\underline{W}}$ be the solution of the following beamforming problem:

$$\underset{\underline{W}}{\text{minimize}}\ \underline{W}^H R \underline{W} \tag{2.1}$$

$$\text{subject to } \underline{W}^H \underline{S}_0 = 1 \tag{2.2}$$

where $\underline{S}_0$ is the steering vector associated with the look direction and R is the ACM.

The complex vector $\underline{\tilde{W}}$, thus represents the L optimal weights of the beamformer which minimizes the mean output power and has unity response in the look direction.

2.1 Standard LMS Algorithm

A real time constrained algorithm for determining the optimal weight vector $\underline{\tilde{W}}$ is

$$\underline{W}(n+1) = P(\underline{W}(n) - \mu g(\underline{W}(n)) + \underline{S}_0/\underline{S}_0^H\underline{S}_0 \tag{2.3}$$

where P is a projection operator and is given by

$$P = I - \underline{S}_0\underline{S}_0^H/L \tag{2.4}$$

$\underline{W}(n+1)$ denotes the new weight vector computed at (n+1)st iteration, μ is a positive scalar which controls the convergence characteristics of the adaptive algorithm and $g(\underline{W}(n))$ is an unbiased estimate of the gradient of $\underline{W}^H(n)R\underline{W}(n)$ with respect to $\underline{W}(n)$.

The gradient of $\underline{W}^H(n)R\underline{W}(n)$ with respect to $\underline{W}(n)$ is given by

$$\nabla w\underline{W}^H R\underline{W}|w = w(n) = 2R\underline{W}(n) \tag{2.5}$$

When all receiver outputs are accessible, the usual estimate of the gradient is made by multiplying the array output with the receiver outputs, that is,

$$\underline{g}(\underline{W}(n)) = 2\underline{X}(n+1)y^*(\underline{W}(n)) \tag{2.6}$$

For a given $\underline{W}(n)$ the estimate given by (2.6) is unbiased.

3. RECURSIVE LMS ALGORITHM

Let $\underline{g}_R(\underline{W})(n))$ denote the estimated gradient by recursive method for a given $\underline{W}(n)$ and be defined as

$$\underline{g}_R(\underline{W}(n)) = 2\,R\,(n+1)\,\underline{W}(n) \tag{3.1}$$

where

$$R(n+1) = [n\,R\,(n) + \underline{X}(n+1)\,\underline{X}^H(n+1)]\,/\,(n+1) \tag{3.2}$$

It follows from (3.2) that $\lim_{n\to\infty} R(n) = R$.

Thus

$$\lim_{n\to\infty} \underline{g}_R(\underline{W}(n)) = 2\ R\ \underline{W}(n) \tag{3.3}$$

that is, the gradient estimate approaches the true gradient as $n\to\infty$.

The following result for the covariance of the gradient is established in [6]. The result is valid for large n, such that $R(n) \cong R$.

Result 3.1: Let $V\underline{g}_R(\underline{W}(n))$ denote the covariance of the gradient estimate defined by (3.1) and (3.2) for a given $\underline{W}(n)$. If $\{\underline{X}(k)\}$ is an independent and identically distributed (iid) complex Gaussian sequence, then

$$V\underline{g}_R(\underline{W}(n)) = \frac{4}{(n+1)^2}\ \underline{W}^H(n)\ R\ \underline{W}(n)\ R \tag{3.4}$$

It follows from (3.4) that the covariance of the estimated gradient by recursive method decreases as iteration number increases and $(n+1)^2$ times less than the covariance of the gradient estimated by (2.6), that is, by the standard method. The covariance of the gradient estimated by the standard method, $V\underline{g}_S(\underline{W}(n))$, is given by [5]

$$V\underline{g}_S(\underline{W}(n)) = 4\ \underline{W}^H(n)\ R\ \underline{W}(n)\ R \tag{3.5}$$

The covariance of the estimated gradient plays an important role in determining the amount of the weight covariance which in turn affects the misadjustment. It is shown in [7] that the component of the covariance of the gradient which affects the weight covariance is $P\ V\underline{g}(\underline{W}(n))\ P$. Let this be referred to as the projected covariance. Taking the projection on both sides of (3.4) and (3.5), and noting that PRP is independent of the look direction signal, one observes that the projected covariance in both the cases is proportional to the mean output power. This implies that for both the cases the projected covariance is a function of the look direction signal. This in turn makes the weight covariance at each iteration sensitive to the look direction signal. However, the signal sensitivity of the weight covariance for the recursive LMS case is $(n+1)^2$ times less than that for the standard LMS case. The signal sensitivity of the standard LMS algorithm has been studied in detail in [7].

4. IMPROVED LMS ALGORITHM

For a linear array of equispaced receivers the array correlation matrix R has Toeplitz structure. The standard LMS algorithm and the recursive LMS algorithm considered in the previous sections do not utilize this property of the array correlation matrix. An algorithm exploiting this structure on per sample basis is reported in [4]. However, this algorithm does not make use of the previous samples when estimating the gradient at nth iteration. A method is presented in this section which exploits the structure of the array correlation matrix and uses the past samples. The method is referred to as the improved method and is presented in a form such that it has a finite memory.

An estimate of the gradient using the improved method is given by

$$\underline{g}_I(\underline{W}(n)) = 2\tilde{R}(n+1)\underline{W}(n) \tag{4.1}$$

where

$$\tilde{R}(n+1) = (1-\alpha)\,\tilde{R}(n) + \alpha\,\hat{R}(n+1) \tag{4.2}$$

with $0<\alpha<1$,

$$\hat{R}(n) = \begin{bmatrix} \hat{r}_0(n) & \hat{r}_1(n) & \cdots & \hat{r}_{L-1}(n) \\ \hat{r}^*_1(n) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \hat{r}_1(n) \\ \hat{r}^*_{L-1}(n) & \cdots & \hat{r}^*_1(n) & \hat{r}_0(n) \end{bmatrix} \tag{4.3}$$

and

$$\hat{r}_l(n) = \frac{1}{L-l}\sum_i x_i(n)x^*_{i+l}(n) \qquad l = 0,\ldots,L-1 \tag{4.4}$$

It can easily be shown that the gradient estimate is unbiased.

The performance and the signal sensitivity of the above algorithm is now compared with a recursive least square (RLS) algorithm which makes use of the past samples and requires the same order of computation for computing the optimal weights.

The following form of the RLS algorithm is used for the comparison

$$\underline{W}(n) = R^{-1}(n)\underline{S}_0 / \underline{S}_0^H R^{-1}(n)\underline{S}_0 \tag{4.5}$$

where $R^{-1}(n)$ is updated using the Matrix Inversion Lemma as follows:

$$R^{-1}(n) = R^{-1}(n-1) - \frac{R^{-1}(n-1)\underline{X}(n)\underline{X}^{H}(n)R^{-1}(n-1)}{1+\underline{X}^{H}(n)R^{-1}(n-1)\underline{X}(n)} \tag{4.6}$$

with

$$R^{-1}(0) = \frac{1}{\varepsilon}\, I, \qquad \varepsilon > 0 \tag{4.7}$$

Note that in the absence of errors as $n \to \infty$, $R^{-1}(n) \to R^{-1}$ and $\underline{W}(n) \to \tilde{\underline{W}}$.

Figures 1 and 2 compare the mean output noise power $P_N(\underline{W}(n))$ versus the iteration number for two look direction signal powers when the weights $\underline{W}(n)$ are adjusted using the two algorithms. The mean output noise power is calculated using

$$P_N(\underline{W}(n)) = \underline{W}^H(n)\, R_N\, \underline{W}(n) \tag{4.8}$$

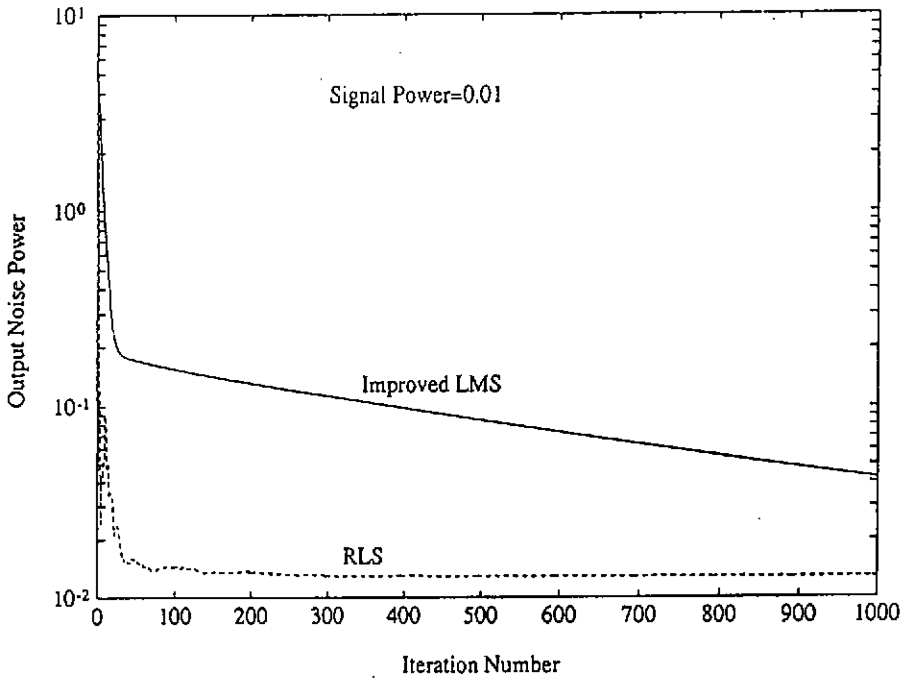


Figure 1: $\underline{W}^H(n)\, R_N\, \underline{W}(n)$ versus the iteration number for a 10 element linear array with one half wavelength spacing. Two interferences: $\theta_1 = 72°$, $p_1 = 100$, $\theta_2 = 98°$, $p_2 = 1$, $\sigma_n^2 = 0.1$, look direction angle $= 90°$.

where R_N is the noise only array correlation matrix, that is,

$$R_N = \sum_{i=1} p_i \underline{S}_i \underline{S}_i^H + \sigma_n^2 I \tag{4.9}$$

with p_i and $\underline{S}_i$ respectively denoting the power and the steering vector corresponding to the ith interference source, σ_n^2 is the variance of uncorrelated noise measured on each element and I is the identity matrix.

A linear array of ten elements with half wavelength spacing is assumed for these examples. The variance of uncorrelated noise present on each element is assumed to be equal to 0.1. Two interference sources are assumed to be present. The first interference falls in the main lobe of the conventional array pattern and makes an angle of 98 degrees with the line of the array. The power of this interference is taken to be 10dB more than the uncorrelated noise power. The second interference makes an angle of 72 degrees with the line of the array and falls in the first side lobe of the conventional pattern. The power of this interference is 30dB more than the uncorrelated noise power. The look direction is broadside to the array. The signal power for Figure 1 is –10dB below the uncorrelated noise power whereas for Figure 2 it is 30dB above the uncorrelated noise

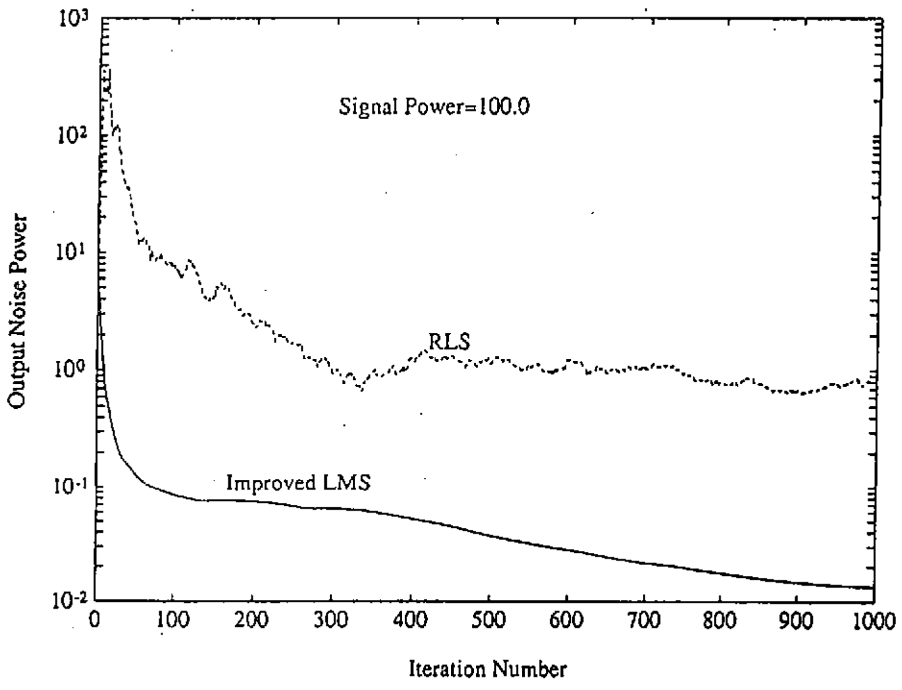


Figure 2: $\underline{W}^H(n)\, R_N\, \underline{W}(n)$ versus the iteration number for a 10 element linear array with one half wavelength spacing. Two interferences: $\theta_1 = 72°$, $p_1 = 100$, $\theta_2 = 98°$, $p_2 = 1$, $\sigma_n^2 = 0.1$, look direction angle = 90°.

power. The gradient algorithm is initialized with the conventional weights. For the improved LMS algorithm the gradient step size is taken to be equal to 0.00005 and for the RLS algorithm ε is taken to be equal to 0.0001.

One observes from these figures that for a very weak signal the RLS algorithm performs better than the improved algorithm. However, as the input signal power increases the output noise power of the processor using the RLS algorithm increases. Thus the RLS algorithm used in the present form is sensitive to the look direction signal. On the other hand this is not the case for the improved LMS algorithm. The performance of the improved LMS algorithm improves as the signal power is increased and in the present of a strong signal it performs much better than the RLS algorithm, both in terms of the convergence and the output SNR.

Figure 3 compares the performance of the standard LMS algorithm presented in Section 2, the recursive LMS algorithm presented in Section 3 and the improved LMS algorithm presented in this section. The noise field and the array geometry used for this example is the same as used for the previous examples. The input signal power is 30dB more than the uncorrelated noise power and the gradient step size used is 0.00005. It is clear from the figure that the output noise power of the processor at each iteration is less when the two algorithms proposed in this paper are used in comparison to the output noise power using the standard algorithm. A comparison of the recursive LMS and the improved LMS shows that the latter performs better, both in terms of the amount of the noise and its variation as a function of iteration number.

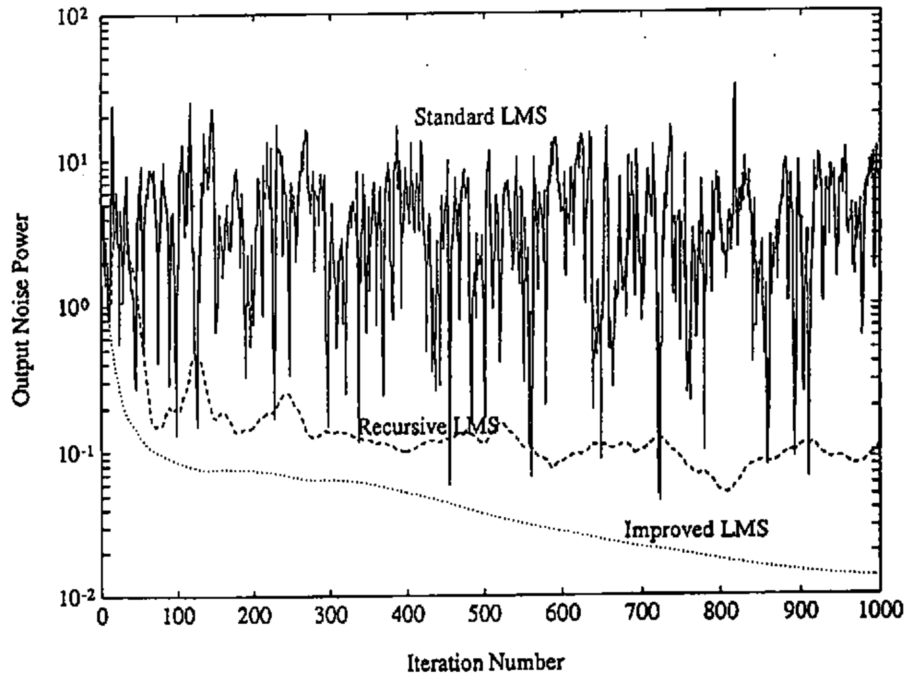


Figure 3: $\underline{W}^H(n)\, R_N\, \underline{W}(n)$ versus the iteration number for a 10 element linear array with one half wavelength spacing. Two interferences: $\theta_1 = 72°$, $p_1 = 100$, $\theta_2 = 98°$, $p_2 = 1$, $\sigma_n^2 = 0.1$, look direction angle = 90°.

5. CONCLUSION

The paper has proposed two algorithms for adaptive beamforming. These algorithms use all the previously available sample to update the array weights in comparison to only one latest sample used in the standard LMS algorithms. One algorithm is applicable to an array of arbitrary geometry. Analysis presented in the paper shows that weights estimated by this algorithm have less variance and are less sensitive to the look direction signal than those estimated by the standard LMS algorithms.

The second algorithm is applicable for a line array. This algorithm not only uses all the previous sample to estimate the weights but also exploit the structure of the array correlation matrix for a line array. The result presented in the paper shows that the performance of the algorithm is not sensitive to look direction signal. This algorithm performs better than the recursive least square algorithm in the presence of a strong look direction signal.

6. ACKNOWLEDGEMENT

Part of the work was carried out when the author was with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford Ca. 94306 and was supported in part by the NASA headquarters, Center for Aeronautics and Space Information Sciences under Grant NAGW-419-S1.3, the SDI/IST Program managed by the Office of Naval Research, under Contract N00014-85-K-0550, and the Department of the Navy, Office of Naval Research under Contract N00014-85-K-0612.

7. REFERENCES

[1] O.L. Frost, "An algorithm for linearly constrained adaptive array", Proc. IEEE, Vol. 68, No. 8, pp926-935, August 1972.

[2] J.E. Hudson, "Adaptive array principles", New York, London: Peter Peregrins, 1981.

[3] L.C. Godara and A. Cantoni, "Analysis of constrained LMS algorithm with application to adaptive beamforming using perturbation sequences", IEEE Trans. Antennas Propagat., Vol. AP-34, No. 3, pp368-379, March 1986.

[4] L.C. Godara and D.A. Gray, "A structured gradient algorithm for adaptive beamforming", ICASSP-88.

[5] A. Cantoni, "Application of orthogonal perturbation sequences to adaptive beamforming", IEEE Trans. Antennas Propagat., Vol. AP-28, No. 3, pp191-202, March 1980.

[6] L.C. Godara, "Improved LMS algorithm for adaptive beamforming", to be published.

[7] L.C.Godara, "Performance analysis of structured gradient algorithm", 1989 International Symposium on Circuits and Systems, 9-11 May 1989, Portland, Oregon, USA.

APPLICATIONS OF SYSTOLIC ARRAYS TO BROADBAND ADAPTIVE BEAMFORMING

P. Gosling (1), J.E. Hudson (1) & J.G. McWhirter (2)

(1) STC Technology Ltd, London Road, Harlow, Essex
(2) RSRE Malvern, Worcestershire.

INTRODUCTION

In this paper we present a class of exact least-squares algorithms based on a modular structure, known as the multistage lattice predictor. These algorithms are known collectively as Least-Squares Lattice (LSL) algorithms, involving both order update and time update recursions. Lattice algorithms are robust, being insensitive to variations in the eigenvalue spread of the correlation matrix of the input data, and have a computational cost that increases linearly with the number of adjustable tap weights.

The history of LSL algorithms can be traced back to the pioneering work Morf [1] on efficient solutions for least-squares predictors. The formal derivation of the LSL algorithm was presented by Morf, Vieira and Lee [2] and by Morf and Lee [3]. Since this time, many refinements, generalisations and applications have been described in the literature. Much of the early work on single channel lattice algorithms is now summarised in the books by Haykin [4] and by Cowan and Grant [5].

An important modification to the LSL formulation came with the generalisation of the algorithm to multi-channel (MLSL) form. For many years, most of the published work dealt with applications of single channel lattice structures. The main objection to the multi-channel formulation being the increase in computational complexity that arises, in particular, from the matrix inversions that the MLSL algorithm requires. Efficient processing architectures for the multi-channel problem started to appear in 1982 when Lev-Ari [6] identified wave-front array processing techniques for the multi-channel lattice algorithms.

In many of the early papers on multi-channel lattices the algorithms were treated as a simple generalisation of the LSL equations [7,8]. This lead to the unnecessary restriction that an equal number of taps is required on each channel. In 1984 Ling and Proakis [9] presented a generalised multi-channel lattice algorithm along with a processing architecture that utilised a Gram-Schmidt orthogonalization process. A QR based algorithm and more general derivation of the MLSL equations was presented by Lewis in the 1988 ICASSP proceedings [10,11].

The papers by Lewis [10,11] and the earlier paper by Ling et al. [9] provide the basis for work described in this paper. Using this previous work we have been able to derive and simulate a processing structure, capable of exact implementation of the MLSL algorithm.

LEAST-SQUARES LATTICE THEORY

The lattice predictor consists of a number of stages connected in cascade, the name lattice being derived from the shape of this cascaded structure. The number of lattice stages is equal to the equivalent prediction-error filter order. Thus for a prediction-error filter of order p, there are p stages in the lattice realisation. The basic algorithm that characterises a single channel lattice predictor may be derived in a number of ways. The starting point for any derivation is the Levinson-Durbin relation so named in recognition of its use first by Levinson [12] and then its independent reformulation by Durbin some thirteen years later [13].

A derivation of the lattice filter algorithm will not be reproduced here however the particular form of algorithm used will be described. For a complete derivation the reader is referred to the book by Haykin [4] and the references contained therein. The Levinson-Durbin relation may be used to derive the following order update recursions that characterise any particular stage m of the lattice predictor.

$$f_m(t) = f_{m-1}(t) - K^f_{m-1}(t)b_{m-1}(t-1) \tag{1}$$

$$b_m(t) = b_{m-1}(t-1) - K^b_m(t)f_{m-1}(t) \tag{2}$$

In this notation f refers to forward prediction errors and b to backward. K^f and K^b are the forward and backward reflection coefficients which in turn are related to the filter coefficients of the corresponding forward and backward prediction error filters. Subscripts are used to denote the filter order and the brackets used to identify the time. Note that in some instances expressions such as X^{f-1} appear in the text. This refers to the inverse of the forward matrix X^f and is not intended to imply X to the power of f-1. A similar notation is used for backward quantities.

The derivation of the MLSL algorithm can, as in the single channel case, follow many different routes. Perhaps the simplest derivation can be obtained by extending the LSL equations to matrix-vector form. As noted by Ling et al. [9] and by Lewis [10], this can impose a considerable loss of generality in the resulting algorithm, however this less general form is sufficient for all practical purposes. To extend the LSL equations to matrix-vector form we make the following assumptions. For an n channel lattice we replace the forward and backward prediction errors (f and b) with the n vectors $\underline{f}$ and $\underline{b}$, and the parameters K^f, K^b, σ^f, σ^b and Δ become nxn matrices. The resulting algorithm is given by,

Cross Error Covariance: $$\Delta_{m-1}(t) = \lambda\Delta_{m-1}(t-1) + \underline{f}_{m-1}(t)\underline{b}^T_{m-1}(t-1)/\gamma_{m-1}(t-1) \tag{3}$$

Forward Prediction Error: $$\underline{f}_m(t) = \underline{f}_{m-1}(t) - \Delta_{m-1}(t)\sigma^{b-1}_{m-1}(t-1)\underline{b}_{m-1}(t-1)) \tag{4}$$

Backward Prediction Error: $$\underline{b}_m(t) = \underline{b}_{m-1}(t-1) - \Delta^T_{m-1}(t)\sigma^{f-1}_{m-1}(t)\underline{f}_{m-1}(t) \tag{5}$$

Forward Error Covariance: $$\sigma^f_m(t) = \lambda\sigma^f_m(t-1) - \underline{f}_m(t)\underline{f}^T_m(t)/\gamma_m(t-1) \tag{6}$$

Backward Error Covariance: $$\sigma^b_m(t) = \lambda\sigma^b_m(t-1) - \underline{b}_m(t)\underline{b}^T_m(t)/\gamma_m(t) \tag{7}$$

Likelihood Variable: $$\gamma_m(t) = \gamma_{m-1}(t-1) - \underline{f}^T_{m-1}\sigma^{f-1}_{m-1}(t)\underline{f}_{m-1}(t) \tag{8}$$

The reflection coefficients are not computed directly, but the necessary computations are included in eqns. 4 & 5. Lewis showed that there is no need to compute the inverse of σ^f and σ^b explicitly each sample since, from the matrix inversion lemma, the inverses may be updated directly. Thus eqns. 6 & 7 become,

$$\sigma^{f-1}_m(t) = \frac{[\sigma^{f-1}_m(t-1) - \sigma^{f-1}_m(t-1)\underline{f}_m(t)\underline{f}^T_m(t)\sigma^{f-1}_m(t-1)]}{\lambda[\lambda\gamma_m(t-1) + \underline{f}^T_m(t)\sigma^{f-1}_m(t-1)\underline{f}(t)]} \tag{9}$$

$$\sigma^{b-1}_m(t) = \frac{[\sigma^{b-1}_m(t-1) - \sigma^{b-1}_m(t-1)\underline{b}_m(t)\underline{b}^T_m(t)\sigma^{b-1}_m(t-1)]}{\lambda[\lambda\gamma_m(t) + \underline{b}^T_m(t)\sigma^{b-1}_m(t-1)\underline{b}(t))]} \tag{10}$$

Lewis has shown that this leads to an algorithm of $O(pn^2)$ computations per sample update. Ling and Proakis [9] generalised the multichannel analysis further to allow for different numbers of taps on each channel. In this case we will confine ourselves to the case where each channel has the same number of taps as most applications will have this format.

JOINT PROCESS ESTIMATION

A lattice predictor consisting of p stages will produce a sequence of m backward prediction errors that are uncorrelated with each other in a deterministic sense for all instants of time. In other words the time averaged or deterministic correlation matrix of the backward prediction errors is a diagonal matrix. This follows directly from the decoupling properties of the LSL algorithm. Hence this structure is often referred to as a "whitening" filter. It follows, therefore, that by using these backward prediction errors as tap inputs applied to a corresponding set of regression coefficients κ, it is possible to determine the least-squares estimate of some desired response $e^x(t)$ exactly and in an efficient manner. The resulting two-channel structure is often referred to as a "joint-process estimator" because it solves the problem of estimating one process from observations of a related process [14]. Mathematically joint process estimation for a single related process channel can be described by the following

$$\Lambda_m(t) = \lambda\Lambda_m(t-1) + b_m(t)e_m^x(t)/\gamma_m(t) \tag{11}$$

$$\kappa_m(t) = \kappa_m(t)/\sigma^b{}_m(t) \tag{12}$$

$$e_{m+1}^x(t) = e_m^x(t) - \kappa_m(t)b_m(t) \tag{13}$$

The corresponding set of equations for multiple related processes (i.e. MLSL) can be obtained using the matrix-vector generalisation described in the previous section.

FORMULATION OF THE LATTICE FILTER USING THE QR DECOMPOSITION ALGORITHM.

The MLSL equations recently described have been reformulated by Lewis [11] using the process of QR decomposition. This formulation gives the necessary insight required to propose a systolic structure for the lattice predictor. We will start by identifying a square-root term for the error covariance matrices. Therefore let us define the n x t matrices Z^f and Z^b by the following,

$$Z_m^f(t) = \begin{bmatrix} \underline{f}_{m-1}^T(t)/\sqrt{\gamma_{m-1}(t-1)} \\ \sqrt{\lambda}\underline{f}_{m-1}^T(t-1)/\sqrt{\gamma_{m-1}(t-2)} \\ \vdots \\ \lambda^{(t-1)/2}\,\underline{f}_{m-1}^T(1)/\sqrt{\gamma_{m-1}(0)} \end{bmatrix} \tag{14}$$

$$Z_m^b(t) = \begin{bmatrix} \underline{b}_{m-1}^T(t)/\sqrt{\gamma_{m-1}(t)} \\ \sqrt{\lambda}\underline{b}_{m-1}^T(t-1)/\sqrt{\gamma_{m-1}(t-1)} \\ \vdots \\ \lambda^{(t-1)/2T}\underline{f}_{m-1}^T(1)/\sqrt{\gamma_{m-1}(1)} \\ \underline{0}^T \end{bmatrix} \tag{15}$$

The error covariance matrices σ^f, σ^b and Δ may then be expressed as,

$$\sigma_{m-1}^f(t) = Z_m^{fT}(t)Z_m^f(t) \qquad \sigma_{m-1}^b(t) = Z_m^{bT}(t)Z_m^b(t) \qquad \Delta_{m-1}(t) = Z_m^{fT}(t)Z_m^b(t-1)$$

Defining a unit vector $\underline{\pi}$ by, $\underline{\pi} = [1\ 0\ ....0\ ]^T$. Then $\underline{f}$ and $\underline{b}$ can be expressed as,

$$\underline{f}_{m-1}(t) = \surd\gamma_{m-1}(t-1)\, Z_m^{fT}(t)\, \underline{\pi} \qquad (16)$$

$$\underline{b}_{m-1}(t) = \surd\gamma_{m-1}(t)\, Z_m^{bT}(t)\, \underline{\pi} \qquad (17)$$

Therefore the recursive variables σ^f, σ^b and Δ can be replaced by recursive computations of Z^f and Z^b. Hence eqns. (4), (5) and (8) may be written as,

$$\underline{f}_m(t) = \underline{f}_{m-1}(t) - \surd\gamma_{m-1}(t-1) Z_m^{fT}(t) Z_m^b(t-1)[Z_m^{bT}(t-1)Z_m^b(t-1)]^{-1} Z_m^{bT}(t-1)\underline{\pi} \qquad (18)$$

$$\underline{b}_m(t) = \underline{b}_{m-1}(t-1) - \surd\gamma_{m-1}(t-1) Z_m^{bT}(t-1) Z_m^f(t)[Z_m^{fT}(t)Z_m^f(t)]^{-1} Z_m^{fT}(t)\underline{\pi} \qquad (19)$$

$$\gamma_m(t) = \gamma_{m-1}(t-1) - \gamma_{m-1}(t-1)\, \underline{\pi}^T Z_m^f(t)[Z_m^{fT}(t)Z_m^f(t)]^{-1} Z_m^{fT}(t)\underline{\pi} \qquad (20)$$

We may now express this problem in terms of the QR factorization of Z^f and Z^b. Applying the QR decomposition we define the following,

$$Z^f(t) = Q^{fT}(t)S^f(t) = [M^{fT}(t)\ N^{fT}(t)] \begin{bmatrix} R^f(t) \\ 0 \end{bmatrix} = M^{fT}(t)R^f(t) \qquad (21)$$

$$Z^b(t-1) = Q^{bT}(t)S^b(t) = [M^{bT}(t)\ N^{bT}(t)] \begin{bmatrix} R^b(t) \\ 0 \end{bmatrix} = M^{bT}(t)R^b(t) \qquad (22)$$

where the subscript m has been omitted for clarity. The matrices Q represent orthogonal transformations and the matrices R are upper triangular. Eqns (18), (19), and (20) contain projections of the form,

$$Z(Z^TZ)^{-1}Z^T = M^TM$$

Therefore by substitution and defining the variables,

$$\underline{\beta}_m^f(t) = \surd\gamma_{m-1}(t-1)\, M_m^f(t)\, \underline{\pi} \qquad \underline{\beta}_m^b(t) = \surd\gamma_{m-1}(t-1)\, M_m^b(t)\, \underline{\pi}$$

$$X_m^f(t) = M_m^f(t) Z_m^b(t-1) \qquad X_m^b(t) = M_m^b(t) Z_m^f(t)$$

The final form of the lattice algorithm is given by the equations,

$$\underline{f}_m(t) = \underline{f}_{m-1}(t) - X_m^{bT}(t)\underline{\beta}_m^b(t) \qquad (23)$$

$$\underline{b}_m(t) = \underline{b}_{m-1}(t-1) - X_m^{fT}(t)\underline{\beta}_m^f(t) \qquad (24)$$

$$\gamma_m(t) = \gamma_{m-1}(t-1) - \underline{\beta}_m^{fT}(t)\underline{\beta}_m^f(t) \qquad (25)$$

BROAD BAND SYSTOLIC ARCHITECTURES

The use of a systolic array for beamforming problems is without doubt a desirable option. The basic triangular array structure has been shown to have good numerical properties and the inherent parallelism within the algorithm makes it simple to distribute the arithmetic tasks and exploit multiple processor capacity to the full [15]. Furthermore the systolic array implements an open loop or direct solution algorithm, which rapidly provides the optimum adaptive solution.

The basic broad band processing scheme consists of a number of sensor inputs. Each sensor input is followed by a tap delay line to which weights are applied and then summed to form the beamformed output. This obviously represents a considerable increase in the magnitude of the processing problem as now for every input channel there are p tap delay inputs increasing the number of inputs from n to np. A simple generalisation of the triangular systolic array used to solve the narrow band problem results in a structure requiring some np inputs and therefore requires $O(n^2p^2)$ operations per sample. Such a structure is basically inefficient. In the broad band case multiple delayed versions of the same signal are passed through the array but, the basic triangular structure makes no allowance for this fact.

In order to realise an efficient systolic structure we must refer again to the earlier analysis where the lattice algorithm is formulated using QR decomposition. To compute the X(t) and $\underline{\beta}(t)$ variables described earlier it is sufficient to update the R(t-1) matrix to the corresponding R(t) matrix and then apply the same rotations to the other appropriate quantities. The matrices R(t) and R(t-1) are simply Cholesky square root factors of the forward and backward data covariance matrices and they may be stored in a triangular systolic array structure [15]. These operations may be summarised by the following augmented matrices,

$$\begin{bmatrix} \sqrt{\lambda} R_m^f(t-1) & \underline{0} & \sqrt{\lambda} X_m^f(t-1) \\ \underline{f}^T(t)/\sqrt{\gamma_{m-1}(t-1)} & \sqrt{\gamma_{m-1}(t-1)} & \underline{b}_{m-1}^T(t-1)/\sqrt{\gamma_{m-1}(t-1)} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} R_m^f(t) & \underline{\beta}_m^f(t) & X_m^f(t) \\ \underline{0}^T & * & * \end{bmatrix} \qquad (26)$$

and

$$\begin{bmatrix} \sqrt{\lambda} R_m^b(t-1) & \underline{0} & \sqrt{\lambda} X_m^b(t-1) \\ \underline{b}_{m-1}^T(t-1)/\sqrt{\gamma_{m-1}(t-1)} & \sqrt{\gamma_{m-1}(t-1)} & \underline{f}_{m-1}^T(t)/\sqrt{\gamma_{m-1}(t-1)} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} R_m^b(t) & \underline{\beta}_m^b(t) & X_m^b(t) \\ \underline{0}^T & * & * \end{bmatrix} \qquad (27)$$

The basic processing structure for a single lattice stage consists of a pair of triangular systolic arrays with additional columns added to the right hand side (Fig. 1(a)). The node algorithms themselves are presented in fig. 1(b). From fig. 1 we see that the basic operation of each stage is to decorrelate the forward residuals from each other in the triangular array and from the delayed versions of the backward residuals through a post-processor structure added to the side of this structure. The corresponding operation is performed to the backward residuals to decorrelate them from one another and from the forward residuals. Each of these stages may be cascaded together to form the structure shown in fig 2.

This systolic lattice structure consists of $O(n^2)$ nodes per lattice stage, the total number of nodes being $O(n^2p)$. Similarly the processing requirement is $O(n^2p)$ representing a considerable saving over an np by np triangular systolic architecture. Using a detailed computer simulation of the systolic architectures we have verified that there is exact agreement between these two systolic methods and the lattice equations 3 to 8.

OPERATION COUNTS

We will now make a more complete analysis of the arithmetic operations required for each of these algorithms. In a recent article [11] Lewis presented the operation counts for the multi-channel lattice equations in normal (i.e. non QR) form. In other words a direct computation of eqns.(3)→(8). The multi-channel equations require a matrix inversion as part of the operation count which is an $O(n^3)$ operations per stage.

Using the matrix inversion lemma, Lewis showed how this can be reduced to O(n^2) operations per stage, the precise number of operations, for real data, being given by $p(17n^2+12n+8)$.

Fig. 1(a).

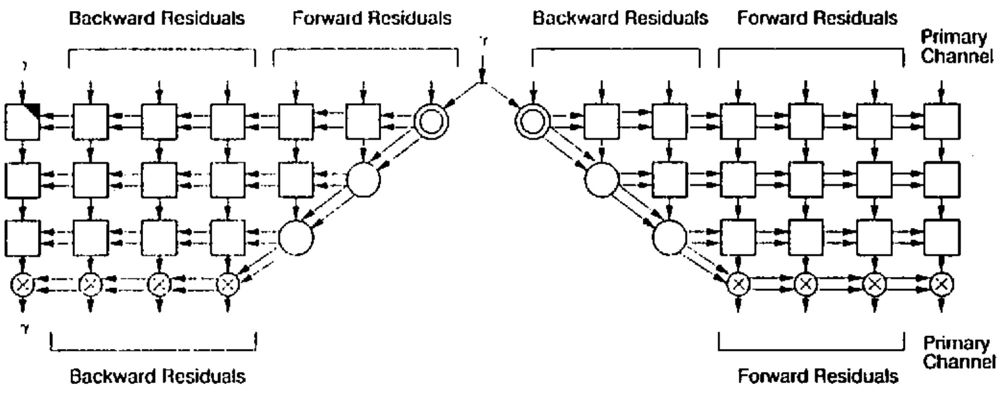


Fig 1(b).

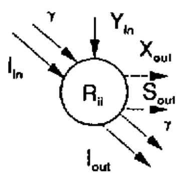
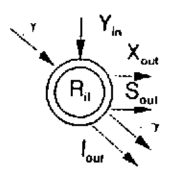
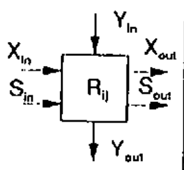
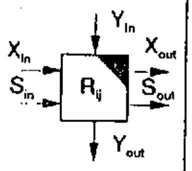
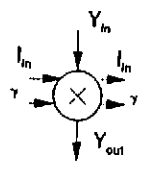
Boundary	Modified Boundary	Internal	Modified Internal	Final
				
	$l_{in}=1/\gamma$			
$d=\beta.R_{ii}$	$d=\beta.R_{ii}$	$Y_{out}=Y_{in}-R_{ij}.X_{in}$	$Y_{out}=Y_{in}^2-R_{ij}.X_{in}$	$Y_{out}=\gamma.l_{in}.Y_{in}$
$R_{ii}=d+Y_{in}^2.l_{in}$	$R_{ii}=d+Y_{in}^2.l_{in}$	$R_{ij}=R_{ij}+S_{in}.Y_{out}$	$R_{ij}=R_{ij}+S_{in}.Y_{out}$	
$t=l_{in}/R_{ii}$	$t=l_{in}/R_{ii}$	$X_{out}=X_{in}$	$X_{out}=X_{in}$	
$l_{out}=t.d$	$l_{out}=t.d$	$S_{out}=S_{in}$	$S_{out}=S_{in}$	
$S_{out}=t.Y_{in}$	$S_{out}=t.Y_{in}$			
$X_{out}=Y_{in}$	$X_{out}=Y_{in}$			

Figure 1: Node arrangement for the systolic multi-channel lattice filter

It is possible to obtain a similar operation count for the MLSL systolic architecture. The basic building blocks of the systolic structure are the node cells so they form an appropriate starting point for any operation count. The node algorithms we will consider are those which avoid the need for computation of square root terms in the boundary cells. In this analysis we will assume there is a requirement for deweighting of past data to maintain numerical stability, this adds one more multiply operation to the boundary cell.

Table 1 illustrates the operation counts for the boundary, internal and final node cells. The operation counts will be different if the data is real or complex. Complex data operation counts are added in brackets.

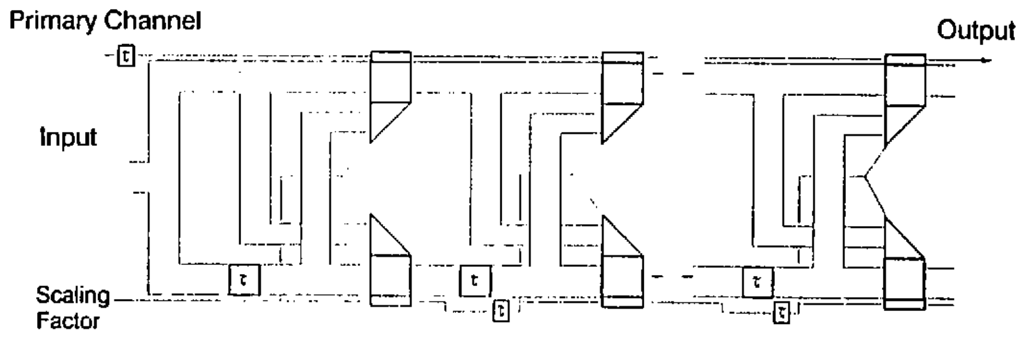


Figure 2: Systolic Multi-channel Least-Squares Lattice filter

Cell type	+/-	X	1/x	TOTAL
Boundary	1 [2]	5 [7]	1 [1]	7 [10]
Internal	2 [8]	2 [8]	0 [0]	4 [16]
Final	0 [0]	2 [3]	0 [0]	2 [3]

Table 1: Operation counts for node algorithms

In the lattice structure there are p stages with $O(n^2)$ nodes in each stage. The total numbers of nodes and operations for all stages are given in table 2.

Cell type	Number	No. of Ops. (real data)
Boundary	$2p(n-1)+(n-1)$	$14p(n-1)+7(n-1)$
Internal	$p(3n^2-5n+2)+(n^2-n)/2$	$4p(3n^2-5n+2)+2(n^2-n)$
Final	$2p(n-1)+1$	$4p(n-1)+2$
Total	$p(3n^2-n-2)+(n^2+n)/2$	$2p(6n^2-n-5)+2n^2+5n-5$

Table 2: Operation count for the systolic lattice structure

For the lattice structure the dominant term in the operation count is $12pn^2$ operations, compared to $2(np)^2$ for a solution involving a full triangular array of np by np nodes. If we equate these two terms we find that, for large n, the lattice structure will be more efficient than the triangular array if there are more than 6 taps. This figure is reduced if terms in n as well as n^2 are included in the computation.

CONCLUSIONS

We have identified the multi-channel least-squares lattice filter as a particularly efficient architecture for broad-band adaptive beamforming. We have described how a primary channel may be incorporated into the systolic structure using the joint process estimation procedure. We have verified that the lattice structure converges within a few samples to simulated broad-band sources and therefore provides much more rapid convergence compared to conventional steepest-descent algorithms, which are sensitive to the eigenvalue spread of the data covariance matrix.

In this work we have presented the systolic architecture required for MLSL and LSL operations. We have simulated both the full equations and the QR form and found them to be in exact agreement, thus verifying the method proposed by Lewis [11] and the extensions proposed here.

For an n channel lattice with p taps per channel the total number of processors required is $O(pn^2)$ for a structure giving an estimate in constant time. This compares favourably to a system in which a triangle of np by np nodes ($O(n^2p^2)$ processors) is used to perform full QR decomposition on the input data. For a system with a less stringent bandwidth constraint a modified architecture using one lattice stage a number of times would result in $O(n^2)$ processors giving an estimate every p samples.

ACKNOWLEDGEMENTS

The authors wish to thank the Directors of STC Technology Ltd. for permission to publish this paper. This work was supported by the Procurement Executive, Ministry of Defence.

REFERENCES

[1] M Morf, 'Fast algorithms for multivariable systems', Ph.D dissertation, Stanford University, Calif. (1974)
[2] M Morf, A Vierra & D T Lee, 'Ladder forms for identification and speech processing', Proc. IEEE Conf. Decision and control, Clearwater beach, Fla., pp. 916-921 (1977)
[3] M Morf & D T Lee, 'Recursive least-squares ladder forms for fast parameter tracking', Proc. IEEE Conf. Decision and control, San Diego, Calif. pp. 1362-1367 (1978)
[4] S Haykin 'Adaptive filter theory', Prentice-Hall information and system sciences series, (1986)
[5] C F N Cowan & P M Grant, 'Adaptive filters', Prentice-Hall signal processing series (1985).
[6] H Lev-Ari, 'Modular architectures for adaptive Multi-channel lattice algorithms', ICASSP 83, Boston (1983)
[7] D T Lee, et. al., 'Recursive least-squares ladder estimation algorithms', IEEE Trans. ASSP, Vol. 29, pp. 627-641, June (1981)
[8] M Morf et. al., 'A classification of algorithms for ARMA models and ladder realisations', ICASSP 77, Hartford, CT, pp 13-19, April (1977)
[9] F Ling, & J G Proakis, 'A generalised multi-channel least-squares lattice algorithm based on sequential processing stages', IEEE trans. ASSP 32(2), April (1984)
[10] P S Lewis, 'Multi-channel adaptive least-squares--relating the "Kalman" recursive least squares (RLS) and least-squares lattice (LSL) adaptive algorithms', ICASSP 88, New York, April (1988)
[11] P S Lewis, "QR algorithm and array architectures for multi-channel adaptive least-squares lattice filters', ICASSP 88, New York, April (1988)
[12] N Levinson, 'The Weiner RMS(root mean square)error criterion in filter design and prediction', J. Math. Phys. Vol. 25, pp 261-278 (1947)
[13] J Durbin, 'The fitting of time series models', Rev. Intern. Statist. Inst. Vol. 28, pp 233-244 (1960)
[14] J Makhoul, 'A class of all-zero lattice digital filters: Properties and applications', IEEE Trans. ASSP Vol. 26, pp 304-314 (1978).
[15] C R Ward, P J Hargrave & J G McWhirter, 'A novel algorithm and architecture for adaptive digital beamforming', IEEE Trans. Antennas and Propagation, Vol. 34(3), pp 338-346, (1986)