Fast null-steering algorithm for broadband power inversion array

C.C. Ko, BSc, PhD

Indexing terms: Algorithms, Antennas (adaptive arrays), Signal processing

Abstract: A fast algorithm for broadband power inversion adaptive arrays is investigated. Essentially, in this algorithm, each broadband jammer is individually tracked by steering a broadband null formed from the use of a set of tapped delay line filters in a Davies beamformer. To steer a particular broadband null to its optimal position, the algorithm first swaps the associated set of null-controlling filters into the last stage of the beam former, and then employs the LMS algorithm with a simple time-invariant decorrelation preprocessor to adjust the filter parameters to minimise the output power for a number of iterations. Performing this procedure for each null in turn results in continuous tracking of all the broadband jammers in a cyclical manner. Although the algorithm requires implementation of the broadband tapped delay-line Davies beamformer, only one tapped delay-line filter is being adaptively updated at any instant. Thus, when the number of elements is small, the overall implementation complexity of the proposed algorithm is comparable to that of using the LMS algorithm directly. However, since each broadband null is independently steered, the proposed algorithm has a much faster convergence behaviour, particularly under situations of severe jamming.

1 Introduction

Because of its ability to steer independent nulls, the Davies beamformer [1] has been attracting considerable interest recently for use in narrowband adaptive arrays. Khanna and Madan [2] have proposed a linearly constrained adaptive array based on the beamformer in which the residue power from each stage of the beamformer is locally minimised by controlling the weights in that stage. However, this local minimisation procedure will not result in optimal SNR improvement, particularly if some jammers have roughly the same powers [3]. The use of the stochastic steepest descent [4] and various simple perturbation algorithms [5] on the beamformer has also been investigated in two recent papers.

For broadband applications, tapped delay-line transversal filters [6], instead of quadrature weights or amplitude/phase controls, are commonly employed for processing behind the array elements. This is because, as discussed in Reference 7 for the broadband power inversion array, the number of weights that need to be controlled is considerably less than the number required in an equivalent array system using a number of narrowband arrays in parallel. However, the use of tapped delay-line processing increases the spread of eigenvalues of the covariance matrix, deteriorating the convergence behaviour of the array substantially [8].

In this paper, a fast null-steering algorithm for broadband power inversion adaptive arrays employing tapped delay-line processing is presented and investigated. To achieve broadband null steering, the narrowband quadrature weights in the Davies beam former [1] are replaced by tapped delay-line filters. By swapping the set of filters controlling a specific null into the last stage of the beamformer, and using the LMS algorithm together with a simple time-invariant decorrelating preprocessor [9] to update the filter parameters for a number of iterations, the null can be adaptively steered to track a broadband jammer. Repeating this adjustment procedure for each null in a cyclical manner, the algorithm leads to the independent steering of all the nulls to track the external environment.

Though requiring implementation of the broadband tapped delay-line Davies beam former, the null-steering algorithm updates only one tapped delay-line filter adaptively at any instant. Thus, for a small number of elements, the overall implementation complexity of the proposed algorithm is comparable to that of using the LMS algorithm directly. However, as will be shown later, since all the nulls are independently steered, the proposed algorithm has much faster convergence behaviour, particularly under situations of severe jamming.

2 Broadband null-steering beamformer

Fig. 1 shows the structure of the narrowband Davies beamformer [1]. For a regular linear array with $M + 1$ elements, it consists of M sets of quadrature weights and summers and has a directional response given by

$$D(z) = \prod_{m=1}^{M} (z - z_m) \tag{1}$$

where

$$z = e^{-j\omega_0 d \sin\theta/c} \tag{2}$$

ω_0 is the centre frequency, d is the element spacing, c is the velocity of wave propagation and θ, measured from the normal of the array as shown, is the direction of interest. From eqn. 1, the mth zero of $D(z)$ is simply given by z_m and thus, by varying the sets of quadrature weights in the beamformer, independent nulls can be steered.

To steer broadband nulls, the sets of quadrature weights can be replaced by sets of adjustable delay lines.

Paper 7442F (E5, E15), first received 13th October 1989 and in revised form 3rd May 1990

The author is with the Department of Electrical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 0511

Using a delay line with gain a_m and delay τ_m to replace z_m, the Davies beamformer will become the broadband null-steering structure of Fig. 2 with a response of

$$D(\theta, \omega) = \prod_{m=1}^{M} (e^{-j\omega d \sin\theta/c} - a_m e^{-j\omega\tau_m}) \quad (3)$$

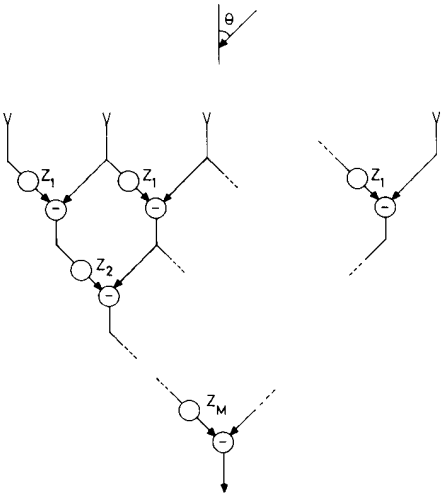

Fig. 1 *Narrowband Davies beamformer for a linear array with M + 1 elements*

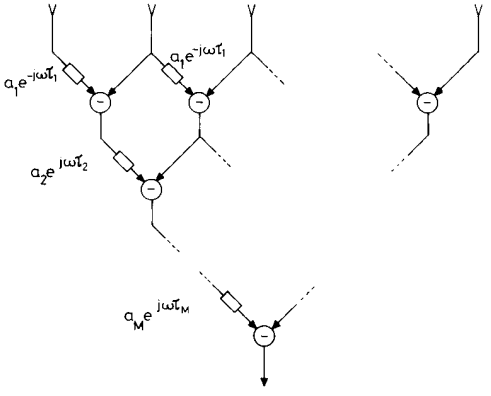

Fig. 2 *Broadband null-steering beamformer*

Obviously, at any particular frequency ω, the broadband structure of Fig. 2 is equivalent to the narrowband beam former of Fig. 1 with

$$z_m = a_m e^{-j\omega\tau_m} \quad (4)$$

Furthermore, if $a_m \simeq 1$, the mth set of delay lines will give rise to a broadband null at

$$\frac{d \sin\theta}{c} = \tau_m \quad (5)$$

Thus, independent broadband null steering can be achieved by simply adjusting the sets of delay lines in Fig. 2.

The main problem in the broadband null-steering structure of Fig. 2 is that variable-length delay lines are not readily available. Therefore, to achieve broadband null steering, a tapped delay-line filter will be used to approximate a variable-length delay line in this paper.

3 Tapped delay-line filter structure

Fig. 3 shows the structure of the tapped delay-line filter that can be used to approximate the mth delay line

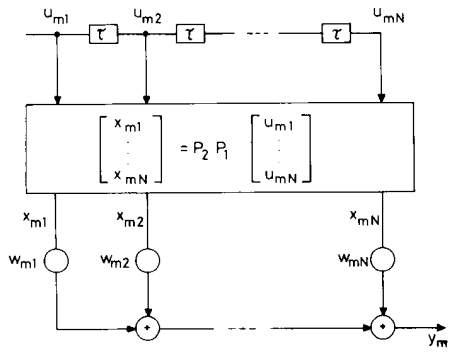

Fig. 3 *Tapped delay-line filter used to approximate the variable length delay line* $a_m \exp(-j\omega\tau_m)$

$a_m e^{-j\omega\tau_m}$. The tap spacing is τ and the number of taps is N. Note that, unlike in the conventional transversal filter, the tap signals $u_{m1}, \ldots, u_{mN}$ are passed through two fixed preprocessing matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ before being multiplied by the N real weights $w_{m1}, \ldots, w_{mN}$ to form the filter output y_m. Mathematically,

$$y_m = \boldsymbol{W}_m^T \boldsymbol{X}_m \quad (6)$$

where

$$\boldsymbol{W}_m = [w_{m1} \cdots w_{mN}]^T \quad (7)$$

$$\boldsymbol{X}_m = [x_{m1} \cdots x_{mN}]^T = \boldsymbol{P}_2 \boldsymbol{P}_1 \boldsymbol{U}_m \quad (8)$$

$$\boldsymbol{U}_m = [u_{m1} \cdots u_{mN}]^T \quad (9)$$

and the superscript T denotes matrix transposition.

As shown in Reference 9, the main purpose of the preprocessing matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ is to transform and decorrelate the tap signals $u_{m1}, \ldots, u_{mN}$ so that fast and predictable convergence behaviour results when simple adaptive algorithms are employed to minimise the output power. Basically, for an array with a fractional bandwidth of β, the input to the filter of Fig. 3 will be bandpass in nature with a spectrum that is nonzero only within the band $(\beta/2) > |(\omega - \omega_0)/\omega_0|$. The covariance matrix for the original tap signals $u_{m1}, \ldots, u_{mN}$ can then be expressed in a power series of β as

$$E[\boldsymbol{U}_m \boldsymbol{U}_m^T] = \boldsymbol{R}_0 + \beta \boldsymbol{R}_1 + \beta^2 \boldsymbol{R}_2 + \cdots \quad (10)$$

where $\boldsymbol{R}_0, \boldsymbol{R}_1, \ldots$ are independent of β. By examining the form of these matrices, the lowest order approximation for a nonsingular $E[\boldsymbol{U}_m \boldsymbol{U}_m^T]$ is

$$E[\boldsymbol{U}_m \boldsymbol{U}_m^T] \simeq \boldsymbol{R}_0 + \beta \boldsymbol{R}_1 + \cdots + \beta^{2N-2} \boldsymbol{R}_{2N-2} \quad (11)$$

assuming N to be even. A similar approximation can be obtained when N is odd. However, because an odd-tap filter is inefficient for approximating a variable-length delay line [10], this paper will be concerned only with the case when N is even.

From the approximation given by eqn. 11, it can be shown that if the preprocessor $\boldsymbol{P}_1$ is chosen to satisfy

$$P_1^T[C_0 \cdots C_{N/2-1}] = \begin{bmatrix} 1 & 0 & \cdots \\ j & 0 & \\ 0 & 1 & \\ 0 & j & \\ \vdots & & \end{bmatrix} \quad (12)$$

with

$$C_n = \frac{(j\beta\omega_0\tau/2)^n}{n!}[0^n e^{j0\omega_0\tau} \cdots (N-1)^n e^{j(N-1)\omega_0\tau}]^T \quad (13)$$

then the covariance matrix will be transformed to become

$$P_1 E[U_m U_m]P_1^T \simeq A \quad (14)$$

where A is dependent only on the shape of the input spectrum. Note that, although C_n is a complex vector, P_1 and in fact all the processing in the algorithm are real in nature. Basically, eqns. 12 and 13 define the $N \times N$ real preprocessor P_1 in terms of $N \times (N/2)$ complex equations with which the signal correlation due to the effects of finite bandwidth and tap delays is eliminated.

Further reduction in signal correlation can be obtained from the fact that the input spectrum is most likely to be flat across the entire frequency band. This is because the jammers will normally not have any knowledge of the frequency band of operation of the array, and thus the most effective jamming signal will be a broadband signal covering the entire suspected frequency band. The input spectrum after bandpass filtering will therefore be roughly flat across the whole band of interest, particularly when the bandwidth is small. Under these situations, if the second preprocessor P_2 is such that

$$P_2^T \begin{bmatrix} 1 & 0 & 1/3 & 0 & 1/5 & \cdots \\ 0 & 1/3 & 0 & 1/5 & & \\ 1/3 & 0 & 1/5 & & & \\ 0 & 1/5 & & & & \\ 1/5 & & & & & \\ \vdots & & & & & \end{bmatrix} P_2 = I \quad (15)$$

the matrix A in eqn. 14, which depends only on the shape of the input spectrum, can be diagonalised and the covariance matrix for the transformed signals becomes

$$E[X_m X_m^T] = P_2 P_1 E[U_m U_m^T]P_1^T P_2^T \simeq E[x_{m1}^2]I \quad (16)$$

In other words, all the transformed signals are now decorrelated and have unity power.

Although preprocessor P_2 is derived from knowledge of the shape of the input spectrum and the entire derivation is based on a first approximation of the covariance matrix, the two preprocessors P_1 and P_2 are found to be effective in eliminating the signal correlation and reduce the eigenvalue spread for bandwidths up to 10% and even when the input spectrum is skewed or has other shapes [9]. Also, as defined by eqns. 12, 13 and 15, they are time-invariant and depend only on the array parameters. The forms of these matrices for the situation of $\tau = (1/4f_0)$ and $N = 2, 4, 6$ are shown in Table 1. Clearly, the preprocessors can be easily implemented.

4 Algorithm

As described in the previous Sections, the proposed fast algorithm is based on a broadband null-steering structure obtained by replacing each variable-length delay line in the beamformer of Fig. 2 by the corresponding tapped delay-line filter of Fig. 3. The resulting structure is shown in Fig. 4, where the mth broadband null is controlled by the mth set of filters $H_m(\omega)$.

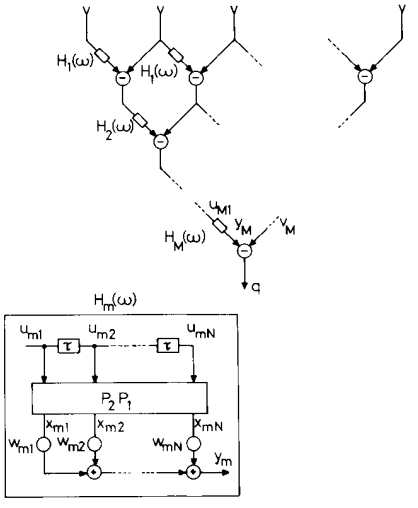

Fig. 4 *Broadband null-steering structure for proposed algorithm*

Without loss of generality, suppose the broadband beamformer is operating in a severe jamming environment where there are M strong broadband jammers with

Table 1: Preprocessors P_1 and P_2 for $\tau = 1/(f_0)$

N	P_1	P_2
2	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
4	$\begin{bmatrix} 0 & 3/2 & 2/(\pi\beta) & 0 \\ -2 & 0 & 0 & 2/(\pi\beta) \\ 0 & 1/2 & 2/(\pi\beta) & 0 \\ -1 & 0 & 0 & 2/(\pi\beta) \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{3} & 0 \\ 0 & 0 & 0 & \sqrt{3} \end{bmatrix}$
6	$\begin{bmatrix} 0 & 15/8 & 4/(\pi\beta) & 0 & 0 & -4/(\pi\beta)^2 \\ -3 & 0 & 0 & 5/(\pi\beta) & 4/(\pi\beta)^2 & 0 \\ 0 & 5/4 & 6/(\pi\beta) & 0 & 0 & -8/(\pi\beta)^2 \\ -3 & 0 & 0 & 8/(\pi\beta) & 8/(\pi\beta)^2 & 0 \\ 0 & 3/8 & 2/(\pi\beta) & 0 & 0 & -4/(\pi\beta)^2 \\ -1 & 0 & 0 & 3/(\pi\beta) & 4/(\pi\beta)^2 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 & -\sqrt{(5/2)} & 0 \\ 0 & 1 & 0 & 0 & 0 & -\sqrt{(5/2)} \\ 0 & 0 & \sqrt{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{(45/2)} & 0 \\ 1 & 0 & & 0 & 0 & \sqrt{(45/2)} \end{bmatrix}$

powers $s_1, \ldots, s_M$ arriving from directions $\theta_1, \ldots, \theta_M$, respectively. In this environment, it can be envisaged that each set of filters will ultimately track a distinct broadband jammer when an adaptive algorithm is used to adjust these to minimise the array output power. Thus, if the first $M-1$ sets of filters $H_1(\omega), \ldots, H_{M-1}(\omega)$ are fairly close to tracking the first $M-1$ jammers from $\theta_1, \ldots, \theta_{M-1}$, respectively, the signals u_{M1} and v_M before the last stage of the beamformer in Fig. 4 will be due mainly to the last jammer coming from θ_M. The use of an adaptive algorithm to process u_{M1} and v_M to minimise the array output power will then give rise to the independent steering of a broadband null, as controlled by $H_M(\omega)$, towards this jammer.

With the above considerations, the proposed algorithm adjusts the sets of tapped delay-line filters one after another cyclically to track broadband jammers in a sequence of adjustment cycles. Each adjustment cycle is concerned with the updating of a specific set of filters, say $H_m(\omega)$, to steer the associated null towards the jammer from θ_m, while the other sets of filters and nulls are kept unchanged.

The adjustment cycle for updating $H_m(\omega)$ starts with the swapping of $H_m(\omega)$ with the last filter $H_M(\omega)$ so that the order of the sets of filters in the beam former of Fig. 4 becomes $H_1(\omega), \ldots, H_{m-1}(\omega), H_M(\omega), H_{m+1}(\omega), \ldots, H_{M-1}(\omega), H_m(\omega)$. The purpose of this swapping is to ensure that, when the other sets of filters are fairly close to tracking the associated jammers, the signals, u_{m1} and v_m to be adaptively processed in the last stage of the beam former will be due predominantly to the jammer being tracked by $H_m(\omega)$. A broadband null will therefore be steered towards this jammer by updating the weights $w_{m1}, \ldots, w_{mN}$ in $H_m(\omega)$ to minimise the output power with an adaptive algorithm for L iterations. Using the LMS algorithm [6] for this purpose, the adjustment being performed in each iteration can be written as

$$\boldsymbol{W}_m \rightarrow \boldsymbol{W}_m + \mu_m q \boldsymbol{X}_m \tag{17}$$

where

$$q = v_m - \boldsymbol{W}_m^T \boldsymbol{X}_m \tag{18}$$

is the array output, μ_m is the feedback factor, and $\boldsymbol{W}_m$ and $\boldsymbol{X}_m$ are the weight and signal vectors as defined in Fig. 4 and eqns. 7 and 8. The choice of L and μ_m and their effects on the performance of the algorithm will be discussed in later Sections.

5 Average convergence behaviour

From eqn. 18, the array output power is

$$E[q^2] = E[v_m^2] - 2\boldsymbol{W}_m^T E[v_m \boldsymbol{X}_m] + \boldsymbol{W}_m^T E[\boldsymbol{X}_m \boldsymbol{X}_m^T] \boldsymbol{W}_m \tag{19}$$

The gradient of this with respect to $\boldsymbol{W}_m$ is

$$\nabla_m = -2E[v_m \boldsymbol{X}_m] + 2E[\boldsymbol{X}_m \boldsymbol{X}_m^T] \boldsymbol{W}_m \tag{20}$$

which gives the optimal weight vector for $H_m(\omega)$ as

$$E[\boldsymbol{X}_m \boldsymbol{X}_m^T] \boldsymbol{W}_{mopt} = E[v_m \boldsymbol{X}_m] \tag{21}$$

Taking the expectation of eqn. 17 and using eqn. 18, the average convergence behaviour of $\boldsymbol{W}_m$ is described by

$$E[\boldsymbol{W}_m] \rightarrow E[\boldsymbol{W}_m] + \mu_m E[v_m \boldsymbol{X}_m - \boldsymbol{X}_m \boldsymbol{X}_m^T \boldsymbol{W}_m] \tag{22}$$

Assuming that the input and weight vectors are independent so that

$$E[\boldsymbol{X}_m \boldsymbol{X}_m^T \boldsymbol{W}_m] = E[\boldsymbol{X}_m \boldsymbol{X}_m^T] E[\boldsymbol{W}_m] \tag{23}$$

and using eqn. 21 then leads to

$$E[\boldsymbol{W}_m - \boldsymbol{W}_{mopt}] \rightarrow \{\boldsymbol{I} - \mu_m E[\boldsymbol{X}_m \boldsymbol{X}_m^T]\} E[\boldsymbol{W}_m - \boldsymbol{W}_{mopt}] \tag{24}$$

Using eqn. 16, this becomes

$$E[\boldsymbol{W}_m - \boldsymbol{W}_{mopt}] \rightarrow \{1 - \mu_m E[x_{m1}^2]\} E[\boldsymbol{W}_m - \boldsymbol{W}_{mopt}] \tag{25}$$

From eqn. 25, it can be seen that $E[\boldsymbol{W}_m]$ converges towards its optimal value with a single time constant of

$$\tau_m = \frac{1}{\mu_m E[x_{m1}^2]} \text{ iterations} \tag{26}$$

Also, after L iterations, the difference of $E[\boldsymbol{W}_m]$ from its optimal value would have decreased by a factor of

$$\rho = \{1 - \mu_m E[x_{m1}^2]\}^L \tag{27}$$

when compared to that before the first iteration. Thus, given the desired ρ, the value of L should be

$$L = \frac{-\ln \rho}{\mu_m E[x_{m1}^2]} = -\tau_m \ln \rho \tag{28}$$

6 Misadjustment

As implied by eqn. 26, the algorithm will have faster convergence behaviour when the feedback factor μ_m is increased. The value of μ_m, however, is restricted by the tolerable amount of extra noise that will be generated at the array output due to the jittering of $\boldsymbol{W}_m$ about its mean value after convergence. This jittering is inevitable because the input signals are stochastic in nature and a finite-memory adaptive algorithm is used in updating $\boldsymbol{W}_m$. The extra output noise that results from this process is often measured in terms of the misadjustment of the algorithm, defined as

$$\text{misadjustment} = \frac{\text{extra output noise power due to jittering of } \boldsymbol{W}_m \text{ after convergence}}{\text{optimum output power}} \tag{29}$$

For the LMS algorithm given by eqns. 17 and 18, the misadjustment for the updating of $\boldsymbol{W}_m$ can be shown [6] to be given by

$$v_m \simeq \frac{\mu_m N E[x_{m1}^2]}{2} \tag{30}$$

Clearly, if the misadjustments from $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_M$ are to be equal, it will be necessary to scale μ_m with respect to $E[x_{m1}^2]$. Specifically, if the total misadjustment is to be v, μ_m has to be given by

$$\mu_m = \frac{2v}{NME[x_{m1}^2]} \tag{31}$$

and it is necessary to measure and estimate the power of x_{m1} before μ_m can be properly normalised.

With the feedback factor μ_m normalised, the convergence time constant for $H_m(\omega)$, as given by eqn. 26, becomes

$$\tau_m = \frac{2v}{NM} \tag{32}$$

Clearly, all the broadband nulls will converge with the same time constant, which is dependent only on the misadjustment desired and the total number of adjustable parameters. As a result, the proposed algorithm can be

envisaged to have fast convergence and tracking behaviour.

In spite of its speed, the implementation complexities of the proposed algorithm can be seen to be comparable to those of using the LMS algorithm directly, particularly when the number of elements is small. On the one hand, the number of parameters that is being adaptively updated at any instant is only N, as compared with NM when the LMS algorithm is used directly on a broadband power inversion array with $M + 1$ elements and N taps. On the other hand, it is computationally more intensive to implement the broadband beam-forming structure of Fig. 4 than to implement a conventional tapped delay-line array.

7 Simulation results

Some simulation results will now be presented in this Section to demonstrate the improvement in convergence behaviour that can result by using the proposed algorithm as well as to verify some of the analytical derivations.

Fig. 5 compares the average output power convergence curve of the new algorithm with curves of using

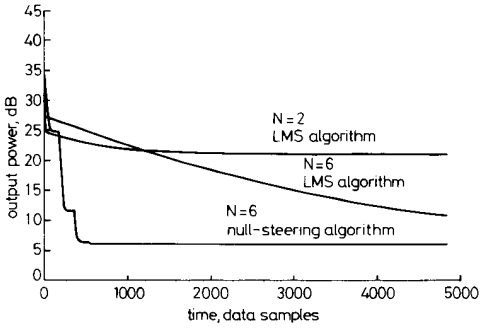

Fig. 5 *Average output power convergence behaviour using the new and LMS algorithms on a three-element array is an external environment consisting of two jammers with powers of 40 dB and 20 dB arriving from 30° and −10°, respectively*

Array: $1/(4f_0)$ tap spacing, 10% bandwidth, 2 or 6 taps, 10% misadjustment. Receiver noise = 0 dB. Each adjustment cycle of the null-steering algorithm consists of 179 iterations

the LMS algorithm directly on a three-element array with $1/(4f_0)$ tap spacing and 10% bandwidth. The external noise environment consists of two jammers with powers of 40 dB and 20 dB coming from 30° and −10°, respectively. The receiver noise power is 0 dB and all the power density spectra are flat across the entire array passband. Also, the misadjustment for the various algorithms is set to 10%, and the value of ρ is fixed at 5% so that each adjustment cycle of the proposed algorithm consists of $L = 179$ LMS iterations.

As can be seen from Fig. 5, the output powers for the two- and six-tap arrays converge towards their optimal values of 21 and 6 dB, respectively. Since the receiver noise power is 0 dB, the use of narrowband processing in this broadband environment is clearly inadequate. To reject the two broadband jammers well, broadband tapped delay-line processing is essential.

Ignoring the difference in asymptotic values, the two LMS curves in Fig. 5 can be seen to converge with at least two main time constants. This demonstrates the well known fact for the LMS algorithm that the weight vector component in the direction of the mth eigenvector converges with a time constant that is inversely proportional to the mth eigenvalue [6]. A large eigenvalue spread will thus lead to a large spread in convergence time constants, and subsequently poor convergence behaviour. For a narrowband array or an equivalent broadband array with only two taps, the spread in eigenvalues is a result of the power and spatial distributions of the jammers. However, for a broadband array with more than two taps, additional spread in eigenvalues may be incurred due to the increase in the number of parameters, the finite bandwidth effects of the signal and the use of tapped delay-line processing. Hence, as can be seen from Fig. 5, the LMS curve for the two-tap array converges significantly faster than that for the broadband six-tap array, although the latter array actually has a much better steady-state performance. (Though it may not be obvious from Fig. 5, the LMS algorithm, when given long enough time, will drive the six-tap array to converge towards a solution which is approximately the same as that shown for the proposed algorithm.)

Intuitively, the slow convergence behaviour using the LMS algorithm directly on the conventional tapped delay-line array is due to the fact that the array weights are not related to the nulls of the array response in a straightforward manner. To overcome this problem, the proposed algorithm makes use of a broadband null-steering beamformer, and attempts to steer the broadband nulls of the array to track the jammers directly. The resulting improvement in convergence behaviour can be seen by comparing the average output power convergence curves depicted in Fig. 5.

To demonstrate the behaviour of the proposed algorithm in more detail, Fig. 6 shows the directional responses of the array for the situation of Fig. 5 at the centre frequency f_0 as well as at the band edges $0.95f_0$ and $1.05f_0$. Fig. 6*a* shows the directional response at the end of the first adjustment cycle when the proposed algorithm is employed. Clearly, as expected, a deep null across the whole passband is steered towards the strong jammer at 30° as a result of adjusting the filter in the last stage of the beam former of Fig. 4. If this filter is now swapped with the other set of filters in the three-element beamformer and the newly swapped filter in the last stage of the beamformer adjusted as previously, the strong jammer at 30° will continue to be well rejected while another null will be steered towards the weaker jammer at −10° in the next adjustment cycle. This is illustrated in Fig. 6*b*, which shows that, after the second adjustment cycle, the array has already steered another null towards the weak jammer. Fig. 7 shows the same sets of curves as Fig. 6, except that the LMS algorithm is now employed directly on a conventional broadband tapped delay-line array. Comparing Fig. 6*a* with Fig. 7*a* and Fig. 6*b* with Fig. 7*b*, it can be seen that using the LMS algorithm directly initially leads to the rapid steering of relatively shallow nulls towards the two jammers. However, further increase in the depth and frequency extent of these nulls, particularly that for the weak jammer, occurs extremely slowly. Again, this illustrates the problem of large eigenvalue spread when the LMS algorithm is employed directly.

Fig. 8 shows the stochastic convergence behaviour of using the proposed algorithm and the LMS algorithm directly on the six-tap array of Fig. 5. Note that the output power curves shown are obtained after averaging 100 runs. The misadjustment measured after convergence is 10% and 9% for the proposed and the LMS algorithms, respectively. Since the designed misadjustment is

10%, this demonstrates the validity of the analytical results derived earlier.

Fig. 9 shows another set of stochastic output power convergence curves obtained from using the proposed

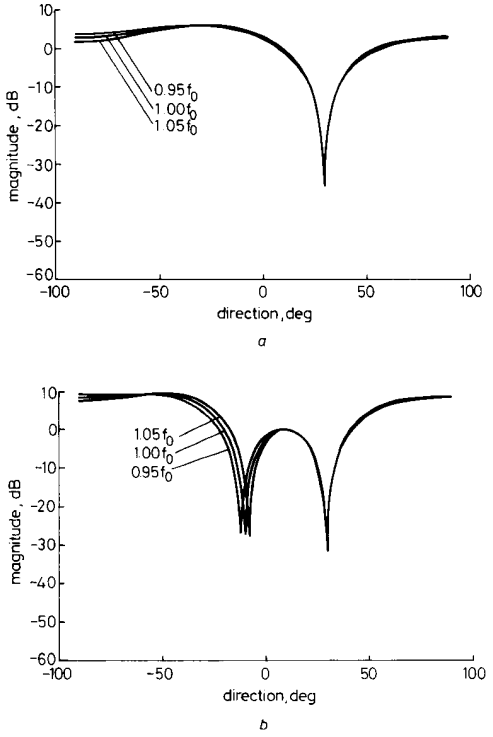

Fig. 6 *Array directional responses at the centre frequency and the band edges for the six-tap array of Fig. 5 when the new algorithm is employed*

a Responses at the end of the first adjustment cycle after 179 data samples
b Responses at the end of the second adjustment cycle after 358 data samples

and the LMS algorithms on the array of Fig. 8 in another severe jamming situation. All the parameters are the same as those for Fig. 8, except that the two jammers have equal power of 40 dB and come from 30° and 40°. Comparing the two convergence curves illustrates once more the superior convergence speed of the proposed algorithm. Furthermore, note that the misadjustments measured after convergence for both algorithms are roughly 9%. Compared with a designed value of 10%, this demonstrates again the validity of the analytical results presented in previous Sections.

Apart from its speed of convergence, another performance measure for any adaptive algorithm is its implementation complexity. To implement the broadband three-element six-tap array of Fig. 5 when the LMS algorithm is used directly, a total of roughly $12 + 12 = 24$ multiplications per iteration are required. This is because 12 multiplications are needed to calculate the output and another 12 multiplications are needed to update the 12 weights for each set of inputs. Now consider the number of multiplications needed if the proposed algorithm is employed. From Fig. 4 and Table 1, this requires a total of roughly $2 \times 6 + 8 + 6 + 6 = 32$ multiplications per iteration, assuming that the preprocessors are efficiently implemented as outlined below.

Firstly, the determination of the two outputs from the first stage of the beam former requires two calculations of the form $\boldsymbol{W}_1^T \boldsymbol{P}_2 \boldsymbol{P}_1 \boldsymbol{U}_1$. Since $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are fixed matrices and $\boldsymbol{W}_1$ is also not changed during the current adjustment cycle, the 6×1 row vector $\boldsymbol{W}_1^T \boldsymbol{P}_2 \boldsymbol{P}_1$ can be predetermined for the entire adjustment cycle, and about 2×6 multiplications per iteration are needed to implement the first stage of the beam former. Next, to implement the LMS algorithm to adjust the last stage of the beam former, the signal vector after preprocessing, $\boldsymbol{X}_2 = \boldsymbol{P}_2 \boldsymbol{P}_1 \boldsymbol{U}_2$, has to be calculated. Because of the favourable structure of the two matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ as shown in Table 1, this calculation can be accomplished with as

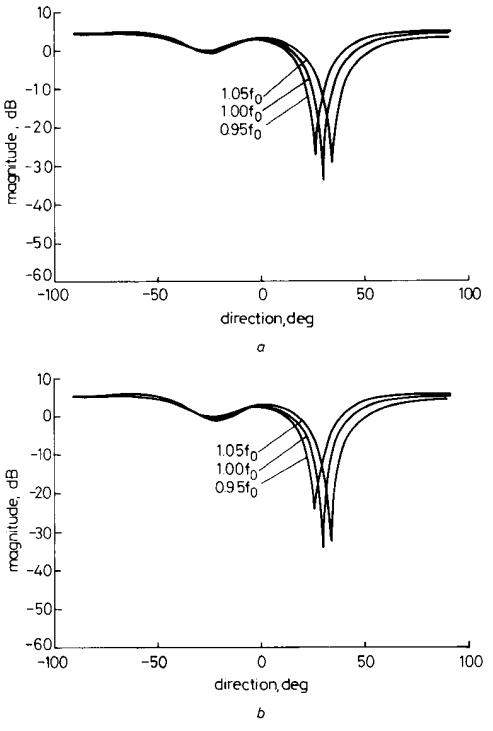

Fig. 7 *Array directional responses at the centre frequency and the band edges for the six-tap array of Fig. 5 when the LMS algorithm is employed*

a Responses after 179 data samples
b Responses after 358 data samples

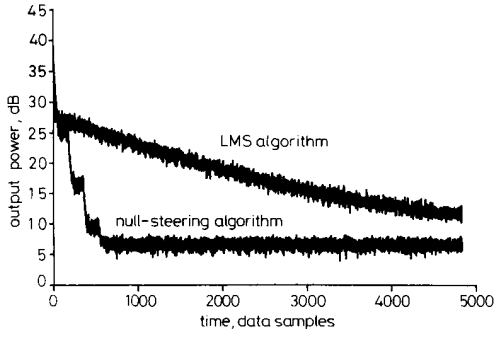

Fig. 8 *Stochastic output power convergence curves, averaged over 100 runs, using the new and LMS algorithms on the six-tap array of Fig. 5*

little as eight multiplications, after using as many additions and shiftings as possible. Finally, with only six adaptive weights, only a further 6 + 6 multiplications are needed to calculate the array output and implement and the LMS algorithm for null-steering purposes.

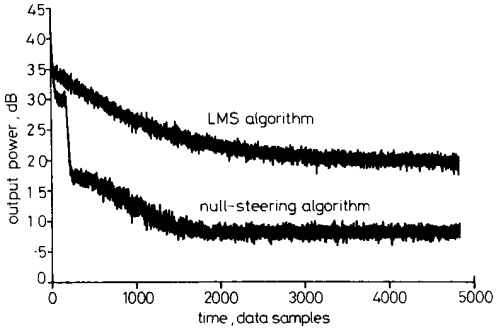

Fig. 9 *Stochastic output power convergence curves, averaged over 100 runs, using the new and LMS algorithms on the six-tap array of Fig. 5 in another severe jamming environment when two 40 dB jammers arrive from 30° and 40°*

10% designed misadjustment; 9% measured misadjustment for both algorithms

Table 2 shows the results of applying the above and gives the number of multiplications that are required to implement the new and the LMS algorithms at a tap spacing of $1/(4f_0)$ for an array with $M + 1$ elements and N taps per delay line. Clearly, provided the number of elements is not large, the proposed fast null-steering algorithm has an implementation complexity that is still comparable to that of using the LMS algorithm directly. As demonstrated in the three-element six-tap array studied, one reason for this is of course that the number of weights that has to be adaptively updated at any time is reduced in the new algorithm. As a last example, to implement a six-element six-tap array, the new and the LMS algorithms require a total of about 104 and 60 multiplications per iteration, respectively, although the former algorithm will have more predictable convergence behaviour and can be orders of magnitude faster.

Table 2: Number of multiplications per iteration required to implement the new and LMS algorithms at $\tau = 1/(4f_0)$ for an array with $M + 1$ elements and N taps per delay line

Algorithm	Multiplications
LMS	$NM^* + NM^{**}$
new algorithm, $N = 2$	$2[M(M+1)-1]\dagger + 4\ddagger$
new algorithm, $N = 4$	$4[M(M+1)-1]\dagger + 10\ddagger$
new algorithm, $N = 6$	$6[M(M+1)-1]\dagger + 20\ddagger$

* Calculating output
** Updating weights
† Implementing all nonadaptive stages in beamformer
‡ Implementing last adaptive stage of beamformer

8 Conclusion

This paper has proposed and investigated a new fast null-steering algorithm for broadband power inversion arrays employing tapped delay-line processing. With this algorithm, each broadband null is individually steered to track a different jammer by employing the LMS algorithm to cyclically update the associated sets of controlling tapped delay-line filters in a broadband null-steering beamformer. By swapping the set of controlling filters for the last stage of the beamformer before adaptively updating its parameters and using a simple time-invariant decorrelation preprocessor to transform the input signals, each broadband null can be made to converge to its optimal position with the same convergence time constant, resulting in rapid convergence and tracking behaviour. In spite of its speed, the overall implementation complexity of the proposed algorithm is still found to be comparable to that of using the LMS algorithm directly on a conventional broadband tapped delay-line array, particularly when the number of elements is small.

9 References

1 DAVIES, D.E.N.: 'Independent angular steering of each zero of the directional pattern of a linear array', *IEEE Trans.*, 1967, **AP-15,** pp. 296–298
2 KHANNA, R., and MADAN, B.B.: 'Adaptive beamforming using a cascade configuration', *IEEE Trans.*, 1983, **AP-31,** pp. 940–945
3 COMPTON, R.T.: 'On the Davies tree for adaptive array processing', *IEEE Trans.*, **1985, ASSP-33,** pp. 1019–1021
4 ORFANIDIS, S.J., and VAIL, L.M.: 'Zero-tracking adaptive filters', *IEEE Trans.*, 1986, **ASSP-34,** pp. 1566–1572
5 KO, C.C.: 'Adaptive array processing using the Davies beamformer', *IEE Proc. H, Microwaves, Antennas & Propag.*, 1986, **133,** pp. 467–473
6 FROST, O.L.: 'An algorithm for linearly constrained adaptive array processing', *Proc. IEEE*, August 1972, **60,** pp. 926–935
7 KO, C.C.: 'The jamming rejection capability of the broadband Frost power inversion array', *IEE Proc. F, Commun., Radar & Signal Process.*, June 1981, **128,** pp. 140–151
8 KO, C.C.: 'Structure of covariance matrix and eigenvalues of broadband tapped delay line adaptive array'. Proc. 1984 IEEE Int. Conf. on Acoustics, speech and signal processing, pp. 33.8.1–33.8.4, March 1984
9 KO, C.C.: 'Simple eigenvalue-equalizing preprocessor for broadband power inversion array', *IEEE Trans.*, January 1985, **AES-21,** pp. 117–127
10 KO, C.C., and LIM, Y.C.: 'Approximation of a variable-length delay line by using tapped delay line processing', *Signal Process.*, June 1988, **14,** pp. 363–369