Distributed Adaptive Algorithms for Large Dimensional MIMO Systems

Barry D. Van Veen, *Senior Member, IEEE*, Olivier Leblond, Vijay P. Mani, and Daniel J. Sebald, *Member, IEEE*

***Abstract*—An algorithm for multi-input multi-output (MIMO) adaptive filtering is introduced that distributes the adaptive computation over a set of linearly connected computational modules. Each module has an input and an output and transmits data to and receives data from its nearest neighbor.**

A gradient-based algorithm for adapting the parameters in each module to minimize the global mean-squared error is derived using principles of back propagation. The performance surface is explored to understand the characteristics of the adaptive algorithm. The minimum mean-squared error is a many to one function of the parameters; therefore, upper bounds on each parameter are used to prevent excessive parameter drift and insure stability with fixed step sizes. Guidelines for choosing the LMS algorithm step sizes and initial conditions are developed. Several examples illustrate the performance of the algorithm.

***Index Terms*—Active noise control, distributed computation, gradient based adaptation, MIMO adaptive systems.**

I. Introduction

ADAPTIVE systems with large numbers of inputs and outputs present significant challenges. The computational burden can be large because the number of adaptive filters is given by the product of the number of inputs and outputs. As a practical matter, the wiring for a system with a single central processor can also be expensive when the inputs and outputs are spatially separated. Furthermore, adaptive systems based on a single central processor are not very flexible since the entire system must be reprogrammed if an input or output is added or removed. An example of an application with these challenges is active noise control since many tens of inputs and outputs are often used [1], [2].

In this paper, we present a distributed algorithm for implementing adaptive multiple-input, multiple-output (MIMO) systems that is based on splitting the adaptive algorithm over a set of linearly connected computational modules [4]. Each module is assumed to have at least one input signal or one output signal and exchanges data with its two neighboring modules. The output of each module is based on a linear combination of its input signal and the data from adjacent modules. The data passed to the module on the right (left) is a linear combination of the input and data received from the module on the left (right). This approach distributes the computational burden over many local processors. The primary advantages of this approach over conventional schemes are the local communication properties and flexibility in adding or removing inputs and outputs. In particular, the number of inputs and outputs may be increased or decreased by simply adding or removing modules without reprogramming the functionality of existing modules. The local communication properties can significantly reduce the wiring requirements for applications with spatially distributed inputs and outputs.

Although the computation associated with a conventional MIMO adaptive system can be distributed over a set of computational modules, such approaches do not possess the local communication attributes and flexibility of the proposed distributed algorithm. For example, one approach to distributing the conventional MIMO computation is to implement the filters associated with each output transducer on separate processors. In this case, global communication of inputs is required since each input must be available to each processor. Furthermore, addition or removal of an input requires changing the functionality of each processor. Other schemes for distributing the conventional MIMO computation over distinct processors suffer from similar disadvantages. In contrast, the proposed approach requires only local communication of inputs and outputs. Hence, the proposed algorithm offers the greatest potential benefit when the number of inputs and outputs are approximately equal and relatively large. In addition, the functionality of each module is independent of the number of inputs and outputs; therefore, changing the number of inputs or outputs does not require changing the algorithm implemented by any of the remaining modules.

The algorithm presented here has the added feature of robustness to failures in the computational units. We show that if one of the computational modules fail, the remaining modules simply act as if the input and output of the failed module are not present and adapt to the corresponding optimum solution. In contrast, if the central processor fails in a conventional MIMO architecture, then the entire system shuts down.

Although not derived from a neural network perspective, the proposed algorithm may be viewed as a very special case of a multilayer *linear* neural network [see, for example, [5] for analysis of a single-input, single-output (SISO) FIR filter neural network]. The similarity to a neural network occurs solely as a consequence of the local connections between modules. The data passed between modules can be viewed as the data at "hidden layers" in a neural network. In contrast to conventional neural networks, the number of hidden layers varies through the MIMO filter since it depends on the number of modules between the

Manuscript received March 31, 1998; revised August 26, 1999. This work was supported by Digisonix, LLC, Madison, WI and Nelson Industries, Inc., Stoughton, WI. The associate editor coordinating the review of this paper and approving it for publication was Prof. Chi Chung Ko.

The authors are with the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706 USA (e-mail: vanveen@engr.wisc.edu).

Publisher Item Identifier S 1053-587X(00)02373-4.

input and output transducers. For example, there is no hidden layer between an input and output associated with any given module since the input and output are related through a conventional FIR filter. If, however, a given input and output are separated by three modules, then the data passed between modules can be associated with hidden layers; therefore, there are three hidden layers between this input and output. Although this special structure could, in principle, be obtained by appropriately pruning a general neural network, it is more intuitive to view it as an implementation of a MIMO filter with local communication constraints. We are not aware of any neural networks with this special structure.

A backpropagation approach is used to derive an LMS adaptive algorithm for adjusting the parameters of each module. Although global distribution of error signals is required, all backpropagated error variables required by the adaptive algorithm share the local communication properties of the filtering process. An analysis of the mean-squared error performance surface is given to develop guidelines for selecting initial conditions and the step sizes associated with each parameter. This analysis differs from that in [5] since [5] considers a simplified network with no interconnections between the hidden layers associated with different samples of the input time series. The distributed algorithm presented here cannot be implemented in this manner.

The paper is organized as follows. Section II defines the function of each module and expresses the MIMO impulse response in terms of the parameters associated with each module. An LMS-based adaptive algorithm for adjusting the parameters in each module is then presented in Section III. The characteristics of the mean-squared error performance surface are explored in Section IV, where guidelines for selecting the initial conditions and step sizes associated with each parameter are given. In Section V, we present simulations that illustrate the performance characteristics of the algorithm. The paper concludes with a summary. Although the results presented in this paper were motivated from the context of an adaptive noise control application, they are broadly applicable to general adaptive control or filtering problems.

II. Modular Implementation for MIMO Systems

Consider a J input, L output MIMO system consisting of M tap FIR filters. Let $u_j(n)$ denote the input from the jth channel and $y_l(n)$ represent the output of the lth channel. Define $\mathbf{u}_j(n) = [u_j(n)\ u_j(n-1)\cdots u_j(n-M+1)]^T$ as a column vector of present and past inputs so that the lth output is expressed as

$$y_l(n) = \sum_{j=1}^{J} \mathbf{h}_{j,l}^T \mathbf{u}_j(n), \qquad l = 1, 2, \cdots, L \tag{1}$$

where $\mathbf{h}_{j,l}$ is a vector representing the impulse response from input j to output l. Equation (1) is rewritten in matrix form as

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ \vdots \\ y_L(n) \end{bmatrix} = \mathbf{H}^T \begin{bmatrix} \mathbf{u}_1(n) \\ \mathbf{u}_2(n) \\ \vdots \\ \mathbf{u}_J(n) \end{bmatrix} \tag{2}$$

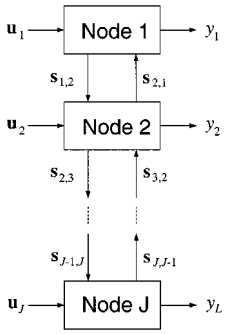


Fig. 1. Block diagram of the distributed architecture depicting one input signal per node. If $L > J$, then at least one node has more than one output. If $L < J$, then at least one node has no output, and if $L = J$, then every node has one output.

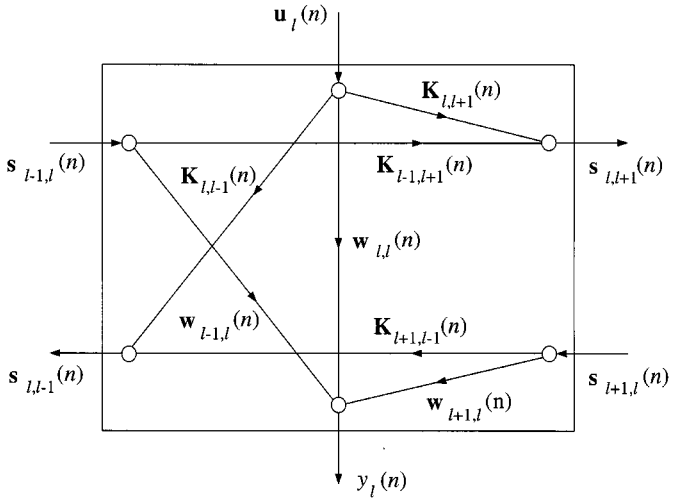


Fig. 2. Signal flow graph for the lth node.

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_{1,1} & \mathbf{h}_{1,2} & \cdots & \mathbf{h}_{1,L} \\ \mathbf{h}_{2,1} & \mathbf{h}_{2,2} & \cdots & \mathbf{h}_{2,L} \\ \vdots & & & \vdots \\ \mathbf{h}_{J,1} & \mathbf{h}_{J,2} & \cdots & \mathbf{h}_{J,L} \end{bmatrix}. \tag{3}$$

Lowercase and uppercase boldface symbols represent vector and matrix quantities, respectively.

The modular implementation depicted in Fig. 1 assumes that each module or node has one input and that the modules are connected in a linear fashion. It is straightforward to extend the results to modules with no inputs and to topologies other than linear connections of modules. As illustrated in Fig. 2, the output for the lth node is defined by

$$y_l(n) = \mathbf{w}_{l,l}^T \mathbf{u}_l(n) + \mathbf{w}_{l-1,l}^T \mathbf{s}_{l-1,l}(n) + \mathbf{w}_{l+1,l}^T \mathbf{s}_{l+1,l}(n) \tag{4}$$

where $\mathbf{w}_{l,l}$, $\mathbf{w}_{l-1,l}$, and $\mathbf{w}_{l+1,l}$ are weight vectors, and $\mathbf{s}_{l-1,l}(n)$ and $\mathbf{s}_{l+1,l}(n)$ denote the P-dimensional data vector communicated to node l from nodes $l-1$ and $l+1$, respectively. That is, the output of each node is a linear combination of the input and the data received from adjacent nodes. Note that we have suppressed the time dependence of the weight vectors

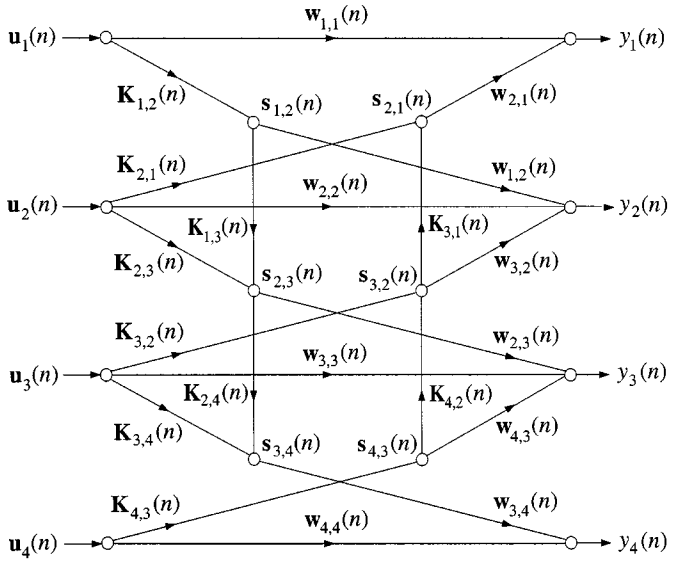


Fig. 3. Signal flow graph for the distributed architecture when $J = 4$ and $L = 4$.

and matrices for notational convenience. The lth node also generates data vectors $\mathbf{s}_{l,\,l-1}(n)$ and $\mathbf{s}_{l,\,l+1}(n)$ as

$$\mathbf{s}_{l,\,l-1}(n) = \mathbf{K}_{l,\,l-1}^T \mathbf{u}_l(n) + \mathbf{K}_{l+1,\,l-1}^T \mathbf{s}_{l+1,\,l}(n) \tag{5}$$

$$\mathbf{s}_{l,\,l+1}(n) = \mathbf{K}_{l,\,l+1}^T \mathbf{u}_l(n) + \mathbf{K}_{l-1,\,l+1}^T \mathbf{s}_{l-1,\,l}(n). \tag{6}$$

Here, the matrices $\mathbf{K}_{l,\,l-1}$ and $\mathbf{K}_{l,\,l+1}$ are M by P, whereas $\mathbf{K}_{l-1,\,l+1}$ and $\mathbf{K}_{l+1,\,l-1}$ are P by P. That is, the data shared to the left (right) is a linear combination of the input and the data received from the right (left). Our convention is to use the first subscript in doubly subscripted quantities to represent *from* and the second subscript to denote *to*. For example, $\mathbf{K}_{l-1,\,l+1}$ describes how data is communicated *from* node $l-1$ *to* node $l+1$. A vector signal flow graph depicting the modular structure for $J = 4$ inputs and $L = 4$ outputs is shown in Fig. 3.

In general, each node can have multiple inputs and/or outputs. Multiple outputs within a node are implemented by changing the $\mathbf{w}$'s in (4) to matrices, whereas multiple inputs are incorporated by concatenating delayed versions of these inputs into the vector $\mathbf{u}_l(n)$ in (4)–(6). For example, if the lth node has two inputs and three outputs, then

- $\mathbf{u}_l(n)$ is a column vector of dimension $2M$;
- $\mathbf{w}_{l,\,l}$ is $2M$ by 3;
- $\mathbf{w}_{l-1,\,l}$ and $\mathbf{w}_{l+1,\,l}$ are P by 3;
- $\mathbf{K}_{l,\,l+1}$ and $\mathbf{K}_{l,\,l-1}$ are $2M$ by P.

Note that each node is amenable to manufacture as an integrated unit containing input electronics, computational and communication resources, and an output transducer. If such hardware devices are employed to implement the MIMO filter, then unequal numbers of inputs and outputs imply some nodes have no input or no output. If there is no input to the lth node, then $\mathbf{w}_{l,\,l}$, $\mathbf{K}_{l,\,l-1}$, and $\mathbf{K}_{l,\,l+1}$ do not exist or, equivalently, are set to zero, whereas both $\mathbf{K}_{l+1,\,l-1}$ and $\mathbf{K}_{l-1,\,l+1}$ are set to identity matrices. If there is no output to the lth node, then the $\mathbf{w}$'s do not exist.

The impulse response from any input to any output is generally a function of a $\mathbf{w}$ vector and multiple $\mathbf{K}$ matrices. Using (4)–(6), it is a straightforward exercise to show that

$$\mathbf{h}_{j,\,l} = \begin{cases} \mathbf{w}_{j,\,j}, & l = j \\ \mathbf{K}_{j,\,j+1}\mathbf{w}_{l-1,\,l}, & l = j+1 \\ \mathbf{K}_{j,\,j-1}\mathbf{w}_{l+1,\,l}, & l = j-1 \\ \mathbf{K}_{j,\,j+1}\displaystyle\prod_{m=j+1}^{l-1}\mathbf{K}_{m-1,\,m+1}\mathbf{w}_{l-1,\,l}, & l > j+1 \\ \mathbf{K}_{j,\,j-1}\displaystyle\prod_{m=j-1}^{l+1}\mathbf{K}_{m+1,\,m-1}\mathbf{w}_{l+1,\,l}, & l < j-1. \end{cases} \tag{7}$$

If $J = L = 4$,, then the MIMO impulse response $\mathbf{H}$ in (3) is given by (8), shown at the bottom of the page.

There is full coupling between all inputs and outputs since all the entries in $\mathbf{H}$ are nonzero. The capability of the distributed algorithm to implement an arbitrary set of impulse responses is strongly dependent on P because different $\mathbf{h}_{j,\,l}$ share the same $\mathbf{K}$'s and $\mathbf{w}$'s. For example, $\mathbf{h}_{1,\,2}$, $\mathbf{h}_{1,\,3}$, and $\mathbf{h}_{1,\,4}$ lie in the P-dimensional subspace spanned by the columns of $\mathbf{K}_{1,\,2}$. In this case, we must at least have $P \geq 3$ before $\mathbf{h}_{1,\,2}$, $\mathbf{h}_{1,\,3}$, and $\mathbf{h}_{1,\,4}$ can be linearly independent.

Conventional approaches for distributing implementation of $\mathbf{H}$ over multiple processors involve partitioning $\mathbf{H}$ in (3) into subsets of filters. For example, implementing the filters associated with each output transducer on a separate processor corresponds to partitioning $\mathbf{H}$ by rows. Note that with this partitioning, all processors (rows of $\mathbf{H}$) must have access to every input; hence, global communication of inputs is required. Furthermore, the functionality of all processors must change if an input is added or removed since this changes the number of filters in each row of $\mathbf{H}$. Similarly, partitioning $\mathbf{H}$ by columns corresponds to implementing the filters associated with each input on separate processors. In this case, global communication of outputs is required, and the functionality of all processors must change if an output is added or removed. Any other partitioning of $\mathbf{H}$ will be characterized by similar global communication requirements and/or lack of flexibility.

The distributed algorithm presented in this section overcomes the limitations of partitioning techniques by factoring $\mathbf{H}$ into a product of matrices [see (7)]. The algorithm does not require

$$\mathbf{H} = \begin{bmatrix} \mathbf{w}_{1,\,1} & \mathbf{K}_{1,\,2}\mathbf{w}_{1,\,2} & \mathbf{K}_{1,\,2}\mathbf{K}_{1,\,3}\mathbf{w}_{2,\,3} & \mathbf{K}_{1,\,2}\mathbf{K}_{1,\,3}\mathbf{K}_{2,\,4}\mathbf{w}_{3,\,4} \\ \mathbf{K}_{2,\,1}\mathbf{w}_{2,\,1} & \mathbf{w}_{2,\,2} & \mathbf{K}_{2,\,3}\mathbf{w}_{2,\,3} & \mathbf{K}_{2,\,3}\mathbf{K}_{2,\,4}\mathbf{w}_{3,\,4} \\ \mathbf{K}_{3,\,2}\mathbf{K}_{3,\,1}\mathbf{w}_{2,\,1} & \mathbf{K}_{3,\,2}\mathbf{w}_{3,\,2} & \mathbf{w}_{3,\,3} & \mathbf{K}_{3,\,4}\mathbf{w}_{3,\,4} \\ \mathbf{K}_{4,\,3}\mathbf{K}_{4,\,2}\mathbf{K}_{3,\,1}\mathbf{w}_{2,\,1} & \mathbf{K}_{4,\,3}\mathbf{K}_{4,\,2}\mathbf{w}_{3,\,2} & \mathbf{K}_{4,\,3}\mathbf{w}_{4,\,3} & \mathbf{w}_{4,\,4} \end{bmatrix} \tag{8}$$

global communication of inputs or outputs. It also provides flexibility since inputs and outputs can be added or removed by simply adding or removing processors (modules) without changing the functionality of the existing processors. Local communication and flexibility are the primary advantages of this distributed algorithm. They become even more significant when adaptive implemenations are considered.

Note that this algorithm is relatively robust to failures of the computational capabilities in any node. Suppose the processor in the lth node fails, and this node has one input and one output. Assuming the lth node retains its local communication capabilities, that is, it sets $\mathbf{s}_{l,\,l+1}(n) = \mathbf{s}_{l-1,\,l}(n)$ and $\mathbf{s}_{l,\,l-1}(n) = \mathbf{s}_{l+1,\,l}(n)$, then the system with the failed node is equivalent to a fully operational system having one less input and output. This corresponds to simply removing the lth row and column of impulse responses in the system matrix of (3) and results in a $J-1$ input, $L-1$ output system. The exact performance degradation associated with loss of one input and output dimension system is strongly scenario dependent. A well-designed system would have enough inputs and outputs so that it is not highly sensitive to the loss of any pair. In contrast, computational failure in a central processor-based system results in complete system failure.

This algorithm is not robust to local communication failures. If the communication between the lth and $l+1$st nodes is disrupted, then the algorithm will independently adapt two systems: one associated with the first l nodes and the other with nodes $l+1$ through J. Independent adaptation of coupled systems is generally undesirable as the systems may compete with one another and experience convergence difficulties [11].

III. LMS-Based Adaptive Algorithm

In adaptive control applications, the errors that are to be minimized by the adaptive algorithm are often measured at the output of a second MIMO system. This system represents the transfer functions of the output transducers and the physical path between the output transducers and error sensors. For example, in active noise control applications, this second system represents the combined effects of the speakers and acoustic paths from each speaker to each error microphone. The presence of systems between the output and error measurement motivated the development of the filtered-X [6] and filtered-U [7] adaptive algorithms. We assume the impulse responses representing the systems between the outputs and errors are either known or identified prior to adaptation of the algorithm.

For simplicity of presentation, we derive the adaptive algorithm, assuming each node has a single input and output. This implies $L = J$. If a node has either no input or no output, then the adaptive algorithm is easily obtained by setting the appropriate $\mathbf{w}$'s and $\mathbf{K}$'s in that node to the values noted in the previous section. It also is straightforward, but cumbersome, to extend the derivations presented here to nodes having multiple inputs and/or outputs.

Let the vector $\mathbf{c}_{l,\,k}$ denote the length-N impulse response between the lth output $y_l(n)$ and the kth error measurement $e_k(n)$, $k = 1, 2, \cdots, K$. Let $d_k(n)$ denote the signal to be approximated at the kth error measurement, and define $\mathbf{y}_l(n) = [y_l(n)\ y_l(n-1) \cdots y_l(n-N+1)]^T$ in order to write the kth error as

$$e_k(n) = d_k(n) - \sum_{l=1}^{L} \mathbf{c}_{l,\,k}^T \mathbf{y}_l(n). \tag{9}$$

The cost function to be minimized is the mean-squared error (MSE)

$$C = \frac{1}{2K} \sum_{k=1}^{K} E\{e_k^2(n)\}. \tag{10}$$

The MSE depends only indirectly on the parameters that we seek to adapt: the $\mathbf{w}$'s and $\mathbf{K}$'s. Furthermore, the MSE depends on products of $\mathbf{w}$'s and $\mathbf{K}$'s because each $\mathbf{h}_{j,\,l}$ in (1) is a product of one $\mathbf{w}$ and several $\mathbf{K}$'s. Hence, we have taken a backpropagation approach [8], [9] to develop an LMS-based adaptive algorithm. Note that the LMS update for a generic parameter a is of the form

$$a(n+1) = a(n) - \mu \frac{\partial \hat{C}(n)}{\partial a(n)} \tag{11}$$

where $\hat{C}(n)$ is the sample MSE.

The traditional LMS algorithm sets the sample MSE equal to the instantaneous squared error. This approach does not work here due to the presence of the transfer functions between the outputs and errors. Note that modifying any parameter at time n introduces a change in the output at time n. Any change in the output at time n effects the errors at times n through $n+N-1$ because the transfer functions between the outputs and errors have length N impulse responses. Hence, to fully capture the impact on the error of changing a parameter at time n, the sample MSE must include the errors from times n through $n+N-1$ as

$$\hat{C}(n) = \frac{1}{2K} \sum_{k=1}^{K} \sum_{q=n}^{n+N-1} e_k^2(q). \tag{12}$$

We now evaluate the gradient of the sample MSE with respect to each parameter with the chain rule.

For example, we obtain the gradient for the $\mathbf{w}$'s by writing

$$\frac{\partial \hat{C}(n)}{\partial \mathbf{w}_{i,\,i}(n)} = \frac{\partial \hat{C}(n)}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial \mathbf{w}_{i,\,i}(n)} \tag{13}$$

$$\frac{\partial \hat{C}(n)}{\partial \mathbf{w}_{i+1,\,i}(n)} = \frac{\partial \hat{C}(n)}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial \mathbf{w}_{i+1,\,i}(n)}. \tag{14}$$

The gradient for $\mathbf{w}_{i-1,\,i}(n)$ is expressed similarly. Let $\delta_i^y(n) = (\partial \hat{C}(n)/\partial y_i(n))$. We show in the Appendix that

$$\delta_i^y(n) = -\frac{1}{K} \sum_{k=1}^{K} [e_k(n)\ e_k(n+1)\ \cdots\ e_k(n+N-1)] \mathbf{c}_{i,\,k}. \tag{15}$$

The noncausal dependence of $\delta_i^y(n)$ on the error is a consequence of any change in the output influencing the present and $N-1$ future values of the error. Now, (4) implies

$$\frac{\partial y_i(n)}{\partial \mathbf{w}_{i,\,i}(n)} = \mathbf{u}_i(n) \tag{16}$$

$$\frac{\partial y_i(n)}{\partial \mathbf{w}_{i+1,\,i}(n)} = \mathbf{s}_{i+1,\,i}(n). \tag{17}$$

Therefore, (13) and (14) become

$$\frac{\partial \hat{C}(n)}{\partial \mathbf{w}_{i,\,i}(n)} = \delta_i^y(n)\mathbf{u}_i(n) \tag{18}$$

$$\frac{\partial \hat{C}(n)}{\partial \mathbf{w}_{i+1,\,i}(n)} = \delta_i^y(n)\mathbf{s}_{i+1,\,i}(n). \tag{19}$$

We delay the gradient by $N-1$ samples so that it is a causal function of the error and obtain the followingLMS updates for the $\mathbf{w}$'s:

$$\begin{aligned}\mathbf{w}_{i,\,i}(n+1) &= \mathbf{w}_{i,\,i}(n) - \mu_w^i \delta_i^y(n-N+1) \\ &\quad \cdot \mathbf{u}_i(n-N+1) \end{aligned} \tag{20}$$

$$\begin{aligned}\mathbf{w}_{i+1,\,i}(n+1) &= \mathbf{w}_{i+1,\,i}(n) - \mu_{w+}^i \delta_i^y(n-N+1) \\ &\quad \cdot \mathbf{s}_{i+1,\,i}(n-N+1) \end{aligned} \tag{21}$$

$$\begin{aligned}\mathbf{w}_{i-1,\,i}(n+1) &= \mathbf{w}_{i-1,\,i}(n) - \mu_{w-}^i \delta_i^y(n-N+1) \\ &\quad \cdot \mathbf{s}_{i-1,\,i}(n-N+1). \end{aligned} \tag{22}$$

The step sizes are assumed to be a function of both the node index and parameter type. The update for each term is the product of a filtered error signal and the input to the weights. Although derived differently, this is similar to the filtered error algorithm given in [3] for adapting a conventional MIMO filter.

Similar applications of the chain rule are used to derive the gradient of the cost function with respect to each of the $\mathbf{K}$'s. For example, we write

$$\frac{\partial \hat{C}(n)}{\partial \mathbf{K}_{i,\,i-1}(n)} = \frac{\partial \hat{C}(n)}{\partial \mathbf{s}_{i,\,i-1}(n)} \frac{\partial \mathbf{s}_{i,\,i-1}(n)}{\partial \mathbf{K}_{i,\,i-1}(n)}. \tag{23}$$

Note that $(\partial \mathbf{s}_{i,\,i-1}(n)/\partial \mathbf{K}_{i,\,i-1}(n)) = \mathbf{u}_i(n)$. We define $\boldsymbol{\delta}_{i,\,i-1}^s(n) = (\partial \hat{C}(n)/\partial \mathbf{s}_{i,\,i-1}(n))$ and write

$$\begin{aligned}\boldsymbol{\delta}_{i,\,i-1}^s(n) &= \frac{\partial \hat{C}(n)}{\partial y_{i-1}(n)} \frac{\partial y_{i-1}(n)}{\partial \mathbf{s}_{i,\,i-1}(n)} \\ &\quad + \frac{\partial \mathbf{s}_{i-1,\,i-2}^T(n)}{\partial \mathbf{s}_{i,\,i-1}(n)} \frac{\partial \hat{C}(n)}{\partial \mathbf{s}_{i-1,\,i-2}(n)} \end{aligned} \tag{24}$$

from which we obtain

$$\begin{aligned}\boldsymbol{\delta}_{i,\,i-1}^s(n) &= \delta_{i-1}^y(n-N+1)\mathbf{w}_{i,\,i-1}(n) \\ &\quad + \mathbf{K}_{i,\,i-2}^T(n)\boldsymbol{\delta}_{i-1,\,i-2}^s(n). \end{aligned} \tag{25}$$

Repeating this process for each case yields the following LMS updates for the $\mathbf{K}$'s:

$$\begin{aligned}\mathbf{K}_{i,\,i+1}(n+1) &= \mathbf{K}_{i,\,i+1}(n) - \mu_{K1+}^i \boldsymbol{\delta}_{i,\,i+1}^s(n) \\ &\quad \cdot \mathbf{u}_i^T(n-N+1) \end{aligned} \tag{26}$$

$$\begin{aligned}\mathbf{K}_{i,\,i-1}(n+1) &= \mathbf{K}_{i,\,i-1}(n) - \mu_{K1-}^i \boldsymbol{\delta}_{i,\,i-1}^s(n) \\ &\quad \cdot \mathbf{u}_i^T(n-N+1) \end{aligned} \tag{27}$$

$$\begin{aligned}\mathbf{K}_{i-1,\,i+1}(n+1) &= \mathbf{K}_{i-1,i+1}(n) - \mu_{K2-}^i \boldsymbol{\delta}_{i,i+1}^s(n) \\ &\quad \cdot \mathbf{s}_{i-1,\,i}^T(n-N+1) \end{aligned} \tag{28}$$

$$\begin{aligned}\mathbf{K}_{i+1,\,i-1}(n+1) &= \mathbf{K}_{i+1,\,i-1}(n) - \mu_{K2+}^i \boldsymbol{\delta}_{i,\,i-1}^s(n) \\ &\quad \cdot \mathbf{s}_{i+1,\,i}^T(n-N+1) \end{aligned} \tag{29}$$

where the local error vectors are given by

$$\boldsymbol{\delta}_{L-1,\,L}^s(n) = \delta_L^y(n-N+1)\mathbf{w}_{L-1,\,L}(n) \tag{30}$$

$$\begin{aligned}\boldsymbol{\delta}_{i,\,i+1}^s(n) &= \delta_{i+1}^y(n-N+1)\mathbf{w}_{i,\,i+1}(n) \\ &\quad + \mathbf{K}_{i,\,i+2}^T(n)\boldsymbol{\delta}_{i+1,\,i+2}^s(n) \end{aligned} \tag{31}$$

for $i = L-2$ down to 1, and

$$\boldsymbol{\delta}_{2,\,1}^s(n) = \delta_1^y(n-N+1)\mathbf{w}_{2,\,1}(n) \tag{32}$$

$$\begin{aligned}\boldsymbol{\delta}_{i,\,i-1}^s(n) &= \delta_{i-1}^y(n-N+1)\mathbf{w}_{i,\,i-1}(n) \\ &\quad + \mathbf{K}_{i,\,i-2}^T(n)\boldsymbol{\delta}_{i-1,\,i-2}^s(n) \end{aligned} \tag{33}$$

for $i = 3$ to L.

Note that the update for each parameter has a familiar form. The new value is equal to the old value minus a step size times the product of a local error and the data associated with the parameter. The local errors are obtained by propagating the output errors backward through the distributed algorithm. First, the local error associated with $y_i(n)$, $\delta_i^y(n)$ is obtained by filtering the output errors $e_k(n)$, $k = 1, \cdots K$ backward through the filters represented by $\mathbf{c}_{i,\,k}$, $k = 1, \cdots K$. Next, the local errors associated with the $\mathbf{s}_{i-1,\,i}(n)$ and $\mathbf{s}_{i+1,\,i}(n)$ are obtained by passing the $\delta_i^y(n)$ backward through the flow graph in Fig. 3. For example, Fig. 3 indicates that $\mathbf{s}_{2,\,3}(n)$ influences both $y_3(n)$ [through $\mathbf{w}_{2,\,3}(n)$] and $\mathbf{s}_{3,\,4}(n)$ [through $\mathbf{K}_{2,\,4}(n)$]. Hence, the local error associated with $\mathbf{s}_{2,\,3}(n)$, $\boldsymbol{\delta}_{2,\,3}^s(n)$, is evaluated by passing $\delta_i^y(n)$ through $\mathbf{w}_{2,\,3}(n)$ and $\boldsymbol{\delta}_{3,\,4}^s(n)$ through $\mathbf{K}_{2,\,4}(n)$. This local error computation is represented by (31) with $i = 2$.

Last, we note that the LMS algorithm for updating parameters retains the local communication properties of the modular architecture. The output errors $e_k(n)$, $k = 1, 2, \cdots, K$, are required by all modules. Knowledge of the ith node output $y_i(n)$ and the output errors enables the ith node to use a system identification algorithm to estimate the filter impulse responses $\mathbf{c}_{i,\,k}$, $k = 1, \cdots, K$ required to evaluate $\delta_i^y(n)$. The local errors $\boldsymbol{\delta}_{i,\,i+1}^s(n)$ and $\boldsymbol{\delta}_{i,\,i-1}^s(n)$ required in the ith module are computed in the $i+1$st and $i-1$st modules. Hence, $\mathbf{s}_{i-1,\,i}(n)$, $\mathbf{s}_{i,\,i-1}(n)$, $\boldsymbol{\delta}_{i,\,i-1}^s(n)$, and $\boldsymbol{\delta}_{i-1,\,i}^s(n)$ are the only quantities communicated between nodes, and the output errors are the only quantities that require global distribution.

IV. Convergence Analysis

The shape of the MSE surface as a function of the $\mathbf{w}$'s and $\mathbf{K}$'s offers information about the convergence behavior of the algorithm. We explore this surface by first expressing it as a function of the $\mathbf{h}_{j,\,l}$ and then substituting the relationships between the $\mathbf{h}_{j,\,l}$ and the $\mathbf{K}$'s and $\mathbf{w}$'s. Express the errors as a function of the $\mathbf{h}_{j,\,l}$ by substituting (1) into (9) to obtain

$$\begin{aligned} e_k(n) &= d_k(n) - \sum_{j,\,l=1}^{J} \mathbf{c}_{l,\,k}^T \begin{bmatrix} \mathbf{u}_j^T(n) \\ \mathbf{u}_j^T(n-1) \\ \vdots \\ \mathbf{u}_j^T(n-N) \end{bmatrix} \mathbf{h}_{j,\,l} \\ &= d_k(n) - \sum_{j,\,l=1}^{J} \mathbf{q}_{j,\,l}^{k\,T}(n)\mathbf{h}_{j,\,l} \end{aligned} \tag{34}$$

where $\mathbf{q}_{j,\,l}^k(n) = [\mathbf{u}_j(n) \;\; \mathbf{u}_j(n-1) \cdots \mathbf{u}_j(n-N)]\mathbf{c}_{l,\,k}$ represents a column vector containing values of the jth input

filtered by the impulse response between the lth output and kth error $\mathbf{c}_{l,k}$. Now, define vectors $\tilde{\mathbf{q}}_k(n) = [\mathbf{q}_{1,1}^{k\,T}(n)\mathbf{q}_{2,1}^{k\,T}(n) \cdots \mathbf{q}_{J,J}^{k\,T}(n)]$ and $\tilde{\mathbf{h}} = \text{vec}\{\mathbf{H}\}$, where vec$\{\mathbf{H}\}$ stacks the columns of $\mathbf{H}$ into a vector. With these definitions, (9) can be rewritten as

$$e_k(n) = d_k(n) - \tilde{\mathbf{q}}_k(n)\tilde{\mathbf{h}}. \tag{35}$$

Hence, the MSE (10) can be expressed in the form

$$C = \sigma^2 - 2\mathbf{p}^T\tilde{\mathbf{h}} + \tilde{\mathbf{h}}^T\mathbf{R}\tilde{\mathbf{h}} \tag{36}$$

where $\sigma^2 = (1/K)\sum_{k=1}^{K} E\{d_k^2(n)\}$, $\mathbf{p} = (1/K)\sum_{k=1}^{K} E\{\tilde{\mathbf{q}}_k^T(n)\ d_k(n)\}$, and $\mathbf{R} = (1/K)\sum_{k=1}^{K} E\{\tilde{\mathbf{q}}_k^T(n)\tilde{\mathbf{q}}_k(n)\}$. Here, $\mathbf{p}$ represents the *average* cross-correlation between the filtered inputs and desired signals, whereas $\mathbf{R}$ is the *average* correlation matrix associated with the filtered inputs. We assume that $\mathbf{R}$ is positive definite, which implies the filtered inputs represented by $\tilde{\mathbf{q}}_k(n)$ are persistently exciting.

A. The Performance Surface

First, note that the MSE is a quadratic form in $\tilde{\mathbf{h}}$ but cannot be expressed as a quadratic form in the $\mathbf{w}$'s and $\mathbf{K}$'s since (7) indicates that $\tilde{\mathbf{h}}$ is a function of their products. Second, the global minimum is unique in $\tilde{\mathbf{h}}$ since $\mathbf{R}$ is nonsingular. However, the global minimum is not unique in the $\mathbf{w}$'s and $\mathbf{K}$'s. This nonuniqueness is also a consequence of $\tilde{\mathbf{h}}$ being a function of products of $\mathbf{w}$'s and $\mathbf{K}$'s. Many different choices for the $\mathbf{w}$'s and $\mathbf{K}$'s result in the same $\tilde{\mathbf{h}}$.

Considerable information about the shape of the performance surface is obtained by examining the Hessian of the cost function. In particular, a positive definite Hessian matrix implies a single global minimum, a semi-definite Hessian indicates the existence of a plateau, and a negative definite Hessian indicates the presence of a global maximum. If α_p denotes the pth parameter (an element of a $\mathbf{w}$ or $\mathbf{K}$), then the p, q element of the Hessian matrix ∇ is given by

$$\begin{aligned}\nabla_{p,q} &= \frac{\partial^2 C}{\partial\alpha_p\partial\alpha_q} \\ &= 2\frac{\partial\tilde{\mathbf{h}}^T}{\partial\alpha_p}\mathbf{R}\frac{\partial\tilde{\mathbf{h}}}{\partial\alpha_q} + 2(\tilde{\mathbf{h}}^T\mathbf{R} - \mathbf{p}^T)\frac{\partial^2\tilde{\mathbf{h}}}{\partial\alpha_p\partial\alpha_q}.\end{aligned} \tag{37}$$

The first term in (37) contributes a term to the Hessian matrix ∇ that can be expressed in the form $\mathbf{A}^T\mathbf{R}\mathbf{A}$, where the pth column of $\mathbf{A}$ is $\partial\tilde{\mathbf{h}}/\partial\alpha_p$. The matrix $\mathbf{A}$ is full rank since each $\mathbf{h}_{j,l}$ depends on a unique set of $\mathbf{w}$'s and $\mathbf{K}$'s. Hence, the first term in the Hessian is positive definite since we have assumed $\mathbf{R}$ is positive definite. However, the second term in (37) may make the overall Hessian indefinite.

Note that $(\partial^2\tilde{\mathbf{h}}/\partial\alpha_p^2) = 0$ because each $\mathbf{h}_{j,l}$ is a linear function of the $\mathbf{w}$'s and $\mathbf{K}$'s. Hence, (37) implies that the diagonal elements of ∇ are given by the quadratic form $(\partial\tilde{\mathbf{h}}^T/\partial\alpha_p)\mathbf{R}(\partial\tilde{\mathbf{h}}/\partial\alpha_p)$. This quantity is always positive; therefore, the performance surface in any single parameter is quadratic with a single global minimum. Second, the trace of the Hessian is always positive since the diagonal elements of the Hessian are positive. Thus, the Hessian matrix cannot be negative definite, and there are no local maxima. Saddle points exist when one or more eigenvalues are negative.

The second term in the Hessian is also zero when α_p and α_q are either elements in the same $\mathbf{w}$, the same $\mathbf{K}$, or elements in any $\mathbf{w}$ and $\mathbf{K}$ that do not appear in the same $\mathbf{h}_{j,l}$ [see (7)]. This is because $\mathbf{h}_{j,l}$ is a linear function of each $\mathbf{K}$ and $\mathbf{w}$; therefore, $(\partial^2\tilde{\mathbf{h}}/\partial\alpha_p\partial\alpha_q) = 0$ under these conditions. Hence, the performance surface is quadratic with a single global minimum in any subspace satisfying these conditions. For example, $\mathbf{K}_{2,1}$ and $\mathbf{K}_{1,2}$ only appear in disjoint $\mathbf{h}_{j,l}$, and the performance surface is quadratic in the subspace defined by the elements of $\mathbf{K}_{2,1}$ and $\mathbf{K}_{1,2}$. We conclude that the performance surface may be nonquadratic only in subspaces defined by parameters α_p and α_q that interact in a single $\mathbf{h}_{j,l}$, such as elements of $\mathbf{K}_{4,3}$ and $\mathbf{K}_{4,2}$, which both appear in $\mathbf{h}_{4,1}$.

B. Uniqueness of Adaptive Trajectories

The $\mathbf{h}_{j,l}$ are not a unique function of the $\mathbf{w}$'s and $\mathbf{K}$'s when $j \neq l$. Considering $j < l$, the set of $\mathbf{w}$'s and $\mathbf{K}$'s defined by

$$\mathbf{w}_{j+1,j}' = \mathbf{Q}_j\mathbf{w}_{j+1,j} \tag{38}$$

$$\mathbf{K}_{j+1,j}' = \mathbf{K}_{j+1,j}\mathbf{Q}_j^{-1} \tag{39}$$

$$\mathbf{K}_{j+1,j-1}' = \mathbf{Q}_j\mathbf{K}_{j+1,j-1}\mathbf{Q}_{j-1}^{-1} \tag{40}$$

result in the same $\mathbf{h}_{j,l}$ and output $y_l(n)$ if the $\mathbf{Q}_j$ represent arbitrary nonsingular matrices. For example

$$\begin{aligned}\mathbf{h}_{4,1}' &= \mathbf{K}_{4,3}'\mathbf{K}_{4,2}'\mathbf{K}_{3,1}'\mathbf{w}_{2,1}' \\ &= \mathbf{K}_{4,3}\mathbf{Q}_3^{-1}\mathbf{Q}_3\mathbf{K}_{4,2}\mathbf{Q}_2^{-1}\mathbf{Q}_2\mathbf{K}_{3,1}\mathbf{Q}_1^{-1}\mathbf{Q}_1\mathbf{w}_{2,1} \\ &= \mathbf{K}_{4,3}\mathbf{K}_{4,2}\mathbf{K}_{3,1}\mathbf{w}_{2,1} \\ &= \mathbf{h}_{4,1}.\end{aligned} \tag{41}$$

In the special case where $\mathbf{Q}_j^{-1} = \mathbf{Q}_j^T$, it is a straightforward exercise to show using induction that the LMS adaptive algorithm results in identical outputs for the set of $\mathbf{w}$'s and $\mathbf{K}$'s related through (39). That is, if we consider the set of all $\mathbf{w}$'s and $\mathbf{K}$'s having identical norms

$$\mathbf{w}_{j+1,j}'(n) = \mathbf{Q}_j\mathbf{w}_{j+1,j}(n) \tag{42}$$

$$\mathbf{K}_{j+1,j}'(n) = \mathbf{K}_{j+1,j}(n)\mathbf{Q}_j^T \tag{43}$$

$$\mathbf{K}_{j+1,j-1}'(n) = \mathbf{Q}_j\mathbf{K}_{j+1,j-1}(n)\mathbf{Q}_{j-1}^T \tag{44}$$

then it can be shown that

$$\mathbf{w}_{j+1,j}'(n+1) = \mathbf{Q}_j\mathbf{w}_{j+1,j}(n+1) \tag{45}$$

$$\mathbf{K}_{j+1,j}'(n+1) = \mathbf{K}_{j+1,j}(n+1)\mathbf{Q}_j^T \tag{46}$$

$$\mathbf{K}_{j+1,j-1}'(n+1) = \mathbf{Q}_j\mathbf{K}_{j+1,j-1}(n+1)\mathbf{Q}_{j-1}^T \tag{47}$$

and thus, $\mathbf{h}_{j,l}'(n) = \mathbf{h}_{j,l}(n)$. Hence, all initial conditions satisfying (39) with $\mathbf{Q}_j^{-1} = \mathbf{Q}_j^T$ result in identical output and error trajectories, even though the adaptive trajectories of the $\mathbf{w}$'s and $\mathbf{K}$'s differ.

C. Step-Size Selection

We now turn our attention to selection of the step-size parameters. Consider adapting a single parameter α while holding all other parameters fixed. The cost function is then quadratic in α and may be expressed as $\alpha^2 r - 2p_o\alpha + \sigma_o^2$, where $r = \frac{1}{2}(\partial^2 C/\partial\alpha^2)$. Analysis of the steepest descent algorithm for adapting α indicates that the step size μ must satisfy $\mu < (2/r)$

to ensure convergence. The maximum step size decreases as r increases. An upper bound on r is obtained from (37) by noting that

$$\begin{aligned}\frac{\partial^2 C}{\partial \alpha^2} &= 2\frac{\partial \tilde{\mathbf{h}}^T}{\partial \alpha}\mathbf{R}\frac{\partial \tilde{\mathbf{h}}}{\partial \alpha} \\ &\le 2\left\|\frac{\partial \tilde{\mathbf{h}}}{\partial \alpha}\right\|_2^2 \|\mathbf{R}\|_2 \\ &\le 2\left\|\frac{\partial \tilde{\mathbf{h}}}{\partial \alpha}\right\|_2^2 \operatorname{tr}\mathbf{R}. \end{aligned} \tag{48}$$

Note, however, that $\|\partial\tilde{\mathbf{h}}/\partial\alpha\|_2^2$ depends on the values of the parameters that are not being adapted. As their magnitudes increase, $\|\partial\tilde{\mathbf{h}}/\partial\alpha\|_2^2$ generally increases without bound. This suggests that we must place upper bounds on all parameters in order to ensure stability with a fixed step size. In addition, the minimum MSE solution for the $\mathbf{w}$'s and $\mathbf{K}$'s is not unique; therefore, upper bounding all parameters prevents excessive drift along any trajectory associated with minimum MSE.

An upper bound on the parameters must be chosen so that the trajectory associated with minimum MSE intersects the region in which adaptation is permitted. One way to ensure this condition is satisfied is by assuming that the maximum gain from input to output at any frequency is upper bounded by a constant c. That is, we assume the optimum filters satisfy $\max_\Omega |H_{j,l}(\Omega)| \le c$.

Consider deriving the bound resulting from the constraint $c \ge H_{j,j+1}(\Omega) = \mathbf{w}_{j,j+1}^T\mathbf{K}_{j,j+1}^T\mathbf{d}(\Omega)$, where $\mathbf{d}(\Omega) = [1\ e^{-j\Omega}\ e^{-j2\Omega}\cdots e^{-j(M-1)\Omega}]^T$. Using various matrix norm inequalities, we obtain

$$\begin{aligned}|H_{j,j+1}(\Omega)| &= |\mathbf{w}_{j,j+1}^T\mathbf{K}_{j,j+1}^T\mathbf{d}(\Omega)| \\ &\le \|\mathbf{w}_{j,j+1}\|_2\|\mathbf{K}_{j,j+1}\|_2\|\mathbf{d}(\Omega)\|_2 \\ &\le \left\{P^{1/2}\max_k\left|[\mathbf{w}_{j,j+1}]_k\right|\right\} \\ &\quad\cdot\left\{(PM)^{1/2}\max_{k,l}\left|[\mathbf{K}_{j,j+1}]_{k,l}\right|\right\}M^{1/2} \\ &= PMW_1K_1 \end{aligned} \tag{49}$$

where W_1 and K_1 are the upper bounds on the absolute values of any element of a $\mathbf{w}$ and a $\mathbf{K}$, respectively, with indices offset by one. Similarly, letting K_2 denote the upper bound on the absolute value of any element of $\mathbf{K}$ with indices offset by 2 and W_0 be the upper bound on the absolute value of any element of $\mathbf{w}_{j,j}$, we see that the condition $\max_\Omega |H_{j,l}(\Omega)| \le c$ is satisfied when

$$W_0 \le \frac{c}{M} \tag{50}$$

$$W_1K_1 \le \frac{c}{PM}, \quad \text{for } |j-l| = 1 \tag{51}$$

$$W_1K_1(PK_2)^{|j-l|-1} \le \frac{c}{PM}, \quad \text{for } |j-l| \ge 2. \tag{52}$$

These conditions are met with equality independent of j and l by choosing $W_0 = (c/M)$, $W_1 = K_1 = \sqrt{c/PM}$, and $K_2 = (1/P)$. Hence, we limit adaptation to the interior of a hypercube defined by the maximum gain c, number of filter taps M, and dimension of the data vector communicated between nodes P. The upper bounds on each parameter are summarized in Table I.

TABLE I
UPPER BOUNDS AND STEP-SIZE BOUNDS FOR EACH PARAMETER IN THE DISTRIBUTED ALGORITHM

Parameters	Step Size Bound	Upper Bound
$\mathbf{w}_{i,i}$	$\mu_w \le \frac{2}{\operatorname{tr}\mathbf{R}}$	$W_0 = \frac{c}{M}$
$\mathbf{w}_{i+1,i}$	$\mu_{w+}^i \le \frac{2}{c(L-i)\operatorname{tr}\mathbf{R}}$	$W_1 = \sqrt{\frac{c}{PM}}$
$\mathbf{w}_{i-1,i}$	$\mu_{w-}^i \le \frac{2}{c(i-1)\operatorname{tr}\mathbf{R}}$	
$\mathbf{K}_{i,i+1}$	$\mu_{K1+}^i \le \frac{2M}{c(L-i)\operatorname{tr}\mathbf{R}}$	$K_1 = \sqrt{\frac{c}{PM}}$
$\mathbf{K}_{i,i-1}$	$\mu_{K1-}^i \le \frac{2M}{c(i-1)\operatorname{tr}\mathbf{R}}$	
$\mathbf{K}_{i+1,i-1}$	$\mu_{K2+}^i \le \frac{2M}{c^2(L-i)(i-1)\operatorname{tr}\mathbf{R}}$	$K_2 = \frac{1}{P}$
$\mathbf{K}_{i-1,i+1}$	$\mu_{K2-}^i \le \frac{2M}{c^2(L-i)(i-1)\operatorname{tr}\mathbf{R}}$	

Bounds on the step size may now be determined using the upper bounds on the parameter values. Again, applying matrix norm inequalities, we obtain upper bounds on $\|\partial\tilde{\mathbf{h}}/\partial\alpha\|_2^2$ in (48) and the corresponding step size bounds. For example, consider the bound on the step size for $\mathbf{w}_{j-1,j}$, μ_{w-}^j. Let α be the kth element of $\mathbf{w}_{j-1,j}$. Using (7), we obtain

$$\frac{\partial \mathbf{h}_{l,p}}{\partial \alpha} = \begin{cases} \mathbf{K}_{l,l+1}\mathbf{e}_k, & l = j-1, p = j \\ \mathbf{K}_{l,l+1}\displaystyle\prod_{m=l+1}^{p-1}\mathbf{K}_{m-1,m+1}\mathbf{e}_k & \\ & l < j-1, p = i \\ 0, & \text{otherwise} \end{cases} \tag{53}$$

where $\mathbf{e}_k$ is a column vector of zeros with a one in the kth position. Now, use matrix norm inequalities to obtain

$$\begin{aligned}&\left\|\frac{\partial \mathbf{h}_{l,p}}{\partial \alpha}\right\|_2^2 \\ &\quad\le \begin{cases} PMK_1^2, & l = j-1, p = j \\ PMK_1^2\displaystyle\prod_{m=l+1}^{p-1}P^2K_2^2, & l < j-1, p = i \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{54}$$

Using $P^2K_2^2 = 1$ and $PMK_1^2 = c$ gives $\|\partial\mathbf{h}_{l,p}/\partial\alpha\|_2^2 \le c$ for $p = j$ and $l \le j-1$. This implies

$$\left\|\frac{\partial \tilde{\mathbf{h}}}{\partial \alpha}\right\|_2^2 = \sum_{i=1}^{j-1}\left\|\frac{\partial \mathbf{h}_{i,j}}{\partial \alpha}\right\|_2^2 \le \sum_{i=1}^{j-1} c = c(j-1). \tag{55}$$

Thus, the step-size bound is $\mu_{w-}^j \le (2/c(j-1)\ \operatorname{tr}\mathbf{R})$. A similar approach gives the remaining step-size bounds summarized in Table I. Note that the maximum step size for each parameter is a function of the node index.

The term $\operatorname{tr}\mathbf{R}$ may be upper bounded in terms of easily measured quantities by applying standard matrix norm inequalities. We obtain

$$\operatorname{tr}\mathbf{R} \le NM\left(\sum_{l,k=1}^{L,K}\mathbf{c}_{l,k}^T\mathbf{c}_{l,k}\right)\sum_{j=1}^{J}\sigma_j^2 \tag{56}$$

where σ_j^2 is the power in the jth input channel.

Several qualifications are in order regarding the step-size bounds. First, the step-size analysis did not consider the effect of the delay by N samples in the update term associated with backpropagation through the $\mathbf{c}_{l,k}$. Delays in the update have a destabilizing influence [10], and thus, the upper bounds will need to be reduced to insure stability when $N \geq 1$. Second, the bounds were obtained by considering the worst-case stability conditions for adaptation of a single parameter. Although this does not guarantee stability when all parameters are simultaneously adapted, we have not yet encountered an example in which instability occured using these bounds.

D. Initial Condition Selection

Last, note that we cannot use initial conditions of all zeros since then, $\mathbf{s}_{j-1,j}(n)$ and $\mathbf{s}_{j+1,j}(n)$ are zero, and the $\mathbf{K}$'s and $\mathbf{w}$'s (except for $\mathbf{w}_{j,j}$) never adapt to a nonzero value. All zeros represents a saddle point in the performance surface. Thus, the gradient is very small in the vicinity of the origin, and adaptation is slow. The step-size analysis indicates that the gradient is largest at the upper bound of each parameter. This suggests that fastest initial convergence is obtained by choosing initial conditions away from the origin.

Initialization of the $\mathbf{K}$'s and $\mathbf{w}$'s at nonzero values does not imply that the initial $\mathbf{h}_{j,l}$ must be nonzero. In fact, zero initial values for the $\mathbf{h}_{j,l}$ are desirable since they ensure that the adaptive system does not increase the error at start up. Zero initial values for the $\mathbf{h}_{j,l}$ are obtained, provided $\mathbf{w}_{j+1,j}(0)$ is orthogonal to the rows of both $\mathbf{K}_{j+1,j}(0)$ and $\mathbf{K}_{j+2,j}(0)$, and $\mathbf{w}_{j-1,j}(0)$ is orthogonal to the rows of both $\mathbf{K}_{j-1,j}(0)$ and $\mathbf{K}_{j-2,j}(0)$. There is a very large class of initial conditions that satisfy these requirements. Furthermore, it follows that any set of initial conditions defined by (39) with $\mathbf{Q}_j^{-1} = \mathbf{Q}_j^T$ result in identical trajectories for the outputs and errors.

A very useful set of initial conditions for P even sets all values in the $\mathbf{w}$'s and $\mathbf{K}$'s at plus or minus the upper bounds such that the initial values of the $\mathbf{h}_{j,l}$ are zero. One way to accomplish this is to set all the elements of the $\mathbf{K}$'s at their upper bound and then set half the elements of the $\mathbf{w}$'s at their upper bound and the remainder at the negative of the upper bound. Any other initial conditions of plus or minus the upper bounds with $\mathbf{h}_{j,l}$ zero may be obtained from this set using (39) with $\mathbf{Q}_j$, which is the product of a permutation matrix and diagonal matrix having ± 1 on the diagonal. Such $\mathbf{Q}_j$ satisfy $\mathbf{Q}_j^{-1} = \mathbf{Q}_j^T$; therefore, all initial conditions of this form result in identical output and error trajectories. If P is odd, then a similar result is obtained by setting all elements of the $\mathbf{K}$'s at the positive upper bound, one of the elements of each $\mathbf{w}$ to zero, and splitting the remainder of the elements of each $\mathbf{w}$ between the positive and negative of the upper bound.

V. Simulations

The adaptive algorithm is demonstrated using a $J = 5$ node system based on $M = 10$ tap filters and $P = 10$ data values communicated between nodes. A block diagram illustrating the simulation configuration is depicted in Fig. 4. The desired signal is given by the output of a MIMO system $\mathbf{G}$ with additive independent Gaussian noise $\mathbf{n}$ of variance 5×10^{-4} in each output channel. The impulse responses in $\mathbf{G}$ were chosen so that zero error resulted in the absence of the additive noise $\mathbf{n}$. The $N = 10$ coefficients in the impulse responses $c_{l,k}(n)$ were obtained as the product of an exponentially decaying window 0.9^n and independent random numbers uniformly distributed on $[-1, 1]$. The maximum gain of $\mathbf{H}$ to any frequency was set at $c = 1$, and the step sizes were chosen as the upper bounds given in Table I. The initial conditions for the $\mathbf{w}$'s and $\mathbf{K}$'s were chosen at plus or minus the upper bounds such that the initial values of the $\mathbf{h}_{j,l}$ are zero.

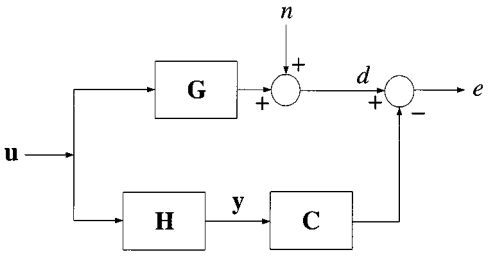


Fig. 4. Block diagram of simulation configuration.

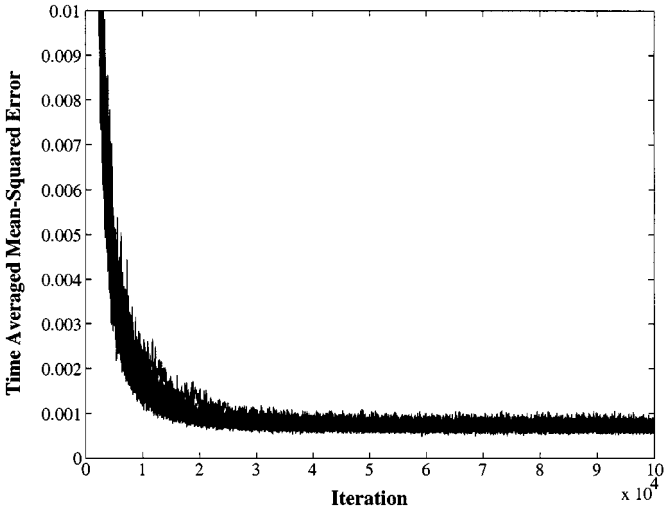


Fig. 5. Time averaged sample MSE of the distributed algorithm for 20 different $\mathbf{G}$ systems and white noise input.

Fig. 5 depicts a 100-point moving average of the sample MSE for 20 different randomly chosen $\mathbf{G}$ systems with the inputs $u_j(n)$ independent, identically distributed Gaussian white noise. The 100-point moving average is used to approximate the true MSE. For comparison, the identical 20 simulations were run using the filtered-X algorithm to adapt the impulse responses $\mathbf{h}_{j,l}(n)$, and the resulting sample MSE's are shown in Fig. 6. The filtered-X algorithm step sizes were also chosen as one half the theoretically maximum step size. The convergence rates of the distributed and filtered-X algorithms are approximately the same. Both converge to the minimum possible value of the MSE, that is, the additive noise variance. The distributed algorithm has larger misadjustment, most likely due to the much larger number of parameters being adapted. Fig. 7 depicts 20 simulations with the sinusoidal input

$$u_j(n) = a_j \cos\left(\frac{\pi}{5}n + \phi_j\right) + b_j \cos\left(\frac{\pi}{10}n + \eta_j\right) \qquad (57)$$

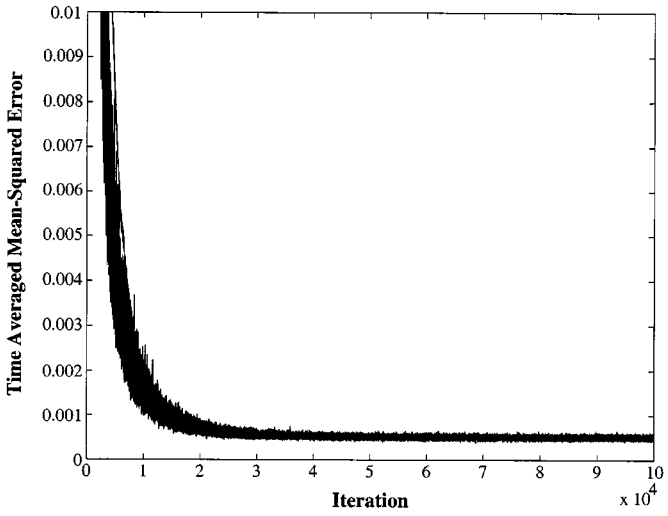


Fig. 6. Time averaged sample MSE of the filtered-X algorithm for 20 different $\mathbf{G}$ systems and white noise input.

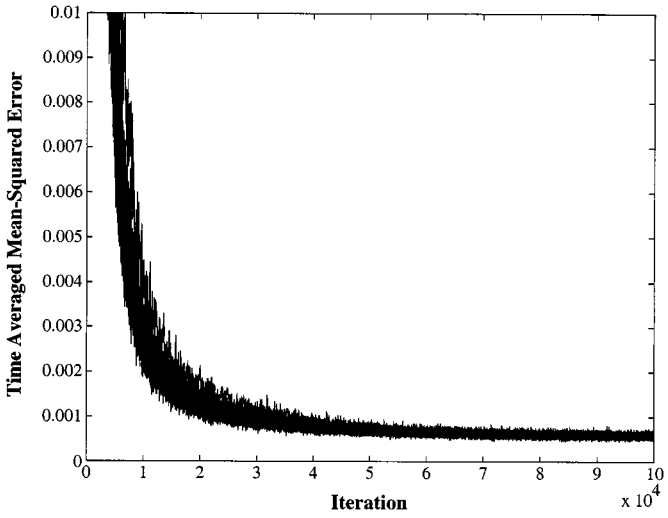


Fig. 7. Time averaged sample MSE of the distributed algorithm for 20 different $\mathbf{G}$ systems and input consisting of two sinusoids.

where the amplitudes a_j and b_j are uniformly distributed random variables on the interval $[0, 1]$, and the phases ϕ_j and η_j are uniformly distributed on $[-\pi, \pi]$. The convergence with tonal inputs is slightly slower, and the misadjustment is somewhat less than with white noise inputs, which is most likely due to the much larger eigenvalue spread in the tonal input covariance matrix.

Last, we illustrate the effect of FIR filters in the $\mathbf{C}$ system. Fig. 8 depicts the MSE for 20 simulations with unit variance white noise inputs and $N = 1$ taps in the $\mathbf{C}$ system. That is, in this case $\mathbf{C}$ is a matrix with entries uniformly distributed on $[-1, 1]$. Note that the distributed algorithm converges about ten times as fast as the comparable $N = 10$ case depicted in Fig. 5.

VI. Summary

An adaptive algorithm for MIMO systems is introduced that distributes the filtering and adaptive algorithm computation over a set of locally connected computational nodes. In general, each node has an input, output, receives data from adjacent nodes,

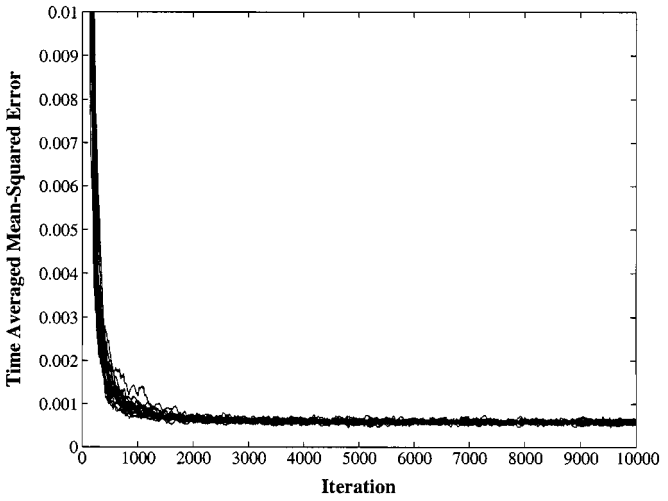


Fig. 8. Time averaged sample MSE of the distributed algorithm for 20 different $\mathbf{G}$ systems with white noise input and the impulse responses $\mathbf{c}_{i,j}$ consisting of scalars.

and transmits data to adjacent nodes. The transmitted data consists of a linear combination of the input to the node and the data received from the adjacent nodes. This configuration implements a fully coupled MIMO system using only local communication between nodes. This approach increases the flexibility of the overall system since nodes (inputs and outputs) can be added or removed without reprogramming the other nodes.

An LMS-based adaptive algorithm for updating the parameters in each node is derived using principles of backpropagation. The performance surface is explored to identify upper bounds on each parameter, develop bounds on the LMS algorithm step-size parameters, and identify appropriate initial conditions. Several simulations illustrate the performance of the algorithm and demonstrate that its convergence characteristics are similar to those of the conventional filtered-X algorithm for MIMO systems.

Appendix
Derivation of (15)

Similar to a derivation in [9], the partial derivative of the error signals—at an arbitrary index q—with respect to output $y_i(n)$ is required for the derivation of (15). Using the error signal definition of (9), this partial derivative becomes

$$\begin{aligned}\frac{\partial e_k(q)}{\partial y_i(n)} &= \frac{\partial}{\partial y_i(n)}\left\{d_k(q) - \sum_{l=1}^{L}\sum_{r=0}^{N-1} c_{l,k}(r)y_l(q-r)\right\} \\ &= -\sum_{l=1}^{L}\sum_{r=0}^{N-1} c_{l,k}(r)\,\frac{\partial y_l(q-r)}{\partial y_i(n)} \\ &= -\sum_{r=0}^{N-1} c_{i,k}(r)\,\frac{\partial y_i(q-r)}{\partial y_i(n)} \end{aligned} \tag{58}$$

where $\partial y_l(q-r)/\partial y_i(n) = 0$ if $l \neq i$. Similarly, the right-hand term of (58) $\partial y_i(q-r)/\partial y_i(n)$ equals zero unless $q-r = n$, in which case, it equals one. Let $r = q-n$ in (58) to conclude that

$$\frac{\partial e_k(q)}{\partial y_i(n)} = \begin{cases} -c_{i,k}(q-n), & 0 \le q-n \le N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{59}$$

Return to the expression of interest [the gradient $\delta_i^y(n) = \partial\hat{C}(n)/\partial y_i(n)$] using the definition of the instantaneous cost function of (12) to find that

$$\delta_i^y(n) = \frac{\partial}{\partial y_i(n)} \left\{ \frac{1}{2K} \sum_{k=1}^{K} \sum_{q=n}^{n+N-1} e_k^2(q) \right\} = \frac{1}{K} \sum_{k=1}^{K} \sum_{q=n}^{n+N-1} e_k(q) \frac{\partial e_k(q)}{\partial y_i(n)} \tag{60}$$

Continue by substituting result (59) into (60) and then letting $s = q - n$ to conclude that

$$\begin{aligned} \delta_i^y(n) &= -\frac{1}{K} \sum_{k=1}^{K} \sum_{q=n}^{n+N-1} e_k(q) c_{i,k}(q-n) \\ &= -\frac{1}{K} \sum_{k=1}^{K} \sum_{s=0}^{N-1} e_k(n+s) c_{i,k}(s) \\ &= -\frac{1}{K} \sum_{k=1}^{K} [e_k(n)\ e_k(n+1)\ \cdots\ e_k(n+N-1)]\, \mathbf{c}_{i,k}. \end{aligned}$$

REFERENCES

[1] S. J. Elliot and P. A. Nelson, "Active noise control," *IEEE Signal Processing Mag.*, vol. 10, pp. 12–35, Oct. 1993.

[2] "Multi-channel active attenuation system with error signal inputs," U.S. Patent 5 216 722.

[3] ——, "A simplified parameter update for identification of multiple input, multiple output systems," in *Proc. Inter Noise*, 1994, pp. 1229–1232.

[4] "Adaptive acoustic attenuation system having distributed processing and shared state nodal architecture," U.S. Patent 5 963 651.

[5] M. I. Kahla, Z. Faraj, F. Castanie, and J. C. Hoffmann, "Multi-layer adaptive filters trained with back propagation: A statistical approach," *Signal Process.*, vol. 40, pp. 65–85, 1994.

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

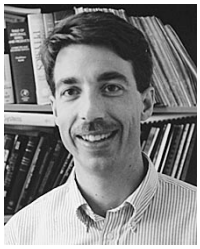
[7] L. J. Eriksson, "Development of the filtered-U algorithm for active noise control," *J. Acoust. Soc.*, vol. 89, pp. 257–265, 1991.

[8] S. Haykin, *Neural Networks*. New York: Macmillan, 1994.

[9] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend and N. A. Gershenfeld, Eds. Reading, MA: Addison Wesley, 1993.

[10] D. R. Morgan and C. Sanford, "A control theory approach to the stability and transient analysis of the filtered-X LMS adaptive notch filter," *IEEE Trans. Signal Processing*, vol. 40, pp. 2341–2346, Sept. 1992.

[11] L. J. Eriksson, M. C. Allie, R. H. Hoops, and J. V. Warner, "Higher order mode cancellation in ducts using active noise control," in *Proc. Internoise*, 1989, pp. 495–500.



Barry D. Van Veen (S'81–M'86–SM'98) was born in Green Bay, WI. He received the B.S. degree from Michigan Technological University, Houghton, in 1983 and the Ph.D. degree from the University of Colorado, Boulder, in 1986, both in electrical engineering. He was an ONR Fellow while pursuing the Ph.D. degree.

In the spring of 1987, he was with the Department of Electrical and Computer Engineering at the University of Colorado. Since August of 1987, he has been with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, and currently holds the rank of Professor. His research interests include signal processing for sensor arrays, nonlinear systems, adaptive filtering, wireless communications, and biomedical applications of signal processing. He coauthored *Signals and Systems* (New York: Wiley, 1999) with S. Haykin.

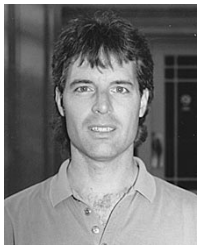
Dr. Van Veen was a recipient of a 1989 Presidential Young Investigator Award from the National Science Foundation and a 1990 IEEE Signal Processing Society Paper Award. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and on the IEEE Signal Processing Society's Technical Committee on Statistical Signal and Array Processing from 1991 through 1997. He received the Holdridge Teaching Excellence Award from the Department of Electrical and Computer Engineering, University of Wisconsin, in 1997.

Olivier Leblond was born in Compiegne, France. He received the E.E. degree in 1996 from Supélec, Paris, France, and the M.Sc. degree from the University of Wisconsin, Madison, in December of 1996.

As a Research Assistant at the University of Wisconsin, he studied digital signal processing and implemented an active noise control algorithm. Following his graduation, he performed research for the Ultrasonic Group at the University of Strathclyde, Glasgow, U.K., as part of his French military service. While in Glasgow, he implemented a visualization algorithm for nondestructive testing. Currently, he is with MicroStrategy, Vienna, VA, as a consultant in decision support systems.

Vijay P. Mani received the B.S. degree in electrical engineering in 1996 from BITS, Pilani, India, and the M.S. degree in electrical engineering in 1998 from the University of Wisconsin, Madison.

He was a Research Assistant with the University of Wisconsin from 1996 to 1998. His research interests include neural networks, signal processing, and image processing.



Daniel J. Sebald (S'89–M'99) received the B.S. degree from the Milwaukee School of Engineering, Milwaukee, WI, in 1987 and the M.S. degree from Marquette University, Milwaukee, in 1992, both in electrical engineering. He is currently pursuing the Ph.D. in electrical engineering at the University of Wisconsin, Madison.

He is a registered P.E. in the state of Wisconsin and has worked for Camtronics Medical Systems, Hartland, WI, and Nicolet Instrument Technologies, Madison. His research interests include signal processing, image processing, and wireless communications.