William S. Hodgkiss, Jr. (S'69-M'75) was born in Bellefonte, PA, on August 20, 1950. He received the B.S.E.E. degree from Bucknell University, Lewisburg, PA, in 1972, and the M.S. and Ph.D. degrees from Duke University, Durham, NC, in 1973 and 1975, respectively.

From 1975 to 1977 he worked with the Naval Ocean Systems Center, San Diego, CA. From 1977 to 1978 he was a faculty member in the Electrical Engineering Department, Bucknell University, Lewisburg, PA. Since 1978 he has been a member of

the faculty of the Scripps Institution of Oceanography and on the staff of the Marine Physical Laboratory, University of California, San Diego. His present research interests are in the areas of adaptive digital signal processing, adaptive array processing, application of these to underwater acoustics, and the statistical properties of ambient ocean noise.

Dimitri Alexandrou (M'85) was born in Salonica, Greece, on August 23, 1954. He received the B.S. and M.S. degrees in electrical engineering from the University of Illinois, Champaign-Urbana, IL, in 1976 and 1978, respectively, and the Ph.D. degree in electrical engineering/ocean science from the University of California, San Diego, in 1985.

From 1978 to 1980 he was a member of the Technical Staff at Bell Telephone Laboratories, Naperville, IL. His research interests include detection and estimation theory, digital signal processing, adaptive filtering, underwater acoustics and the statistical properties of oceanic reverberation. He is a member of the Acoustical Society of America and the American Geophysical Union.

Fast Orthogonalization Networks

KARL GERLACH, MEMBER, IEEE

Abstract—A numerically efficient and stable algorithm is developed to adaptively filter multiple input channels into desired multiple output channels. The algorithm is a type of adaptive lattice filter that employs a fast orthogonalization network (FON) algorithm for numerical efficiency. Past researchers have concentrated on developing efficient lattice algorithms to process stationary input channels; i.e., the input covariance matrix is Toeplitz in form. The algorithm developed herein is designed to adaptively filter nonstationary input channels. Various implementations of the FON algorithm are given.

I. INTRODUCTION

THE DIRECT ADAPTIVE filtering of multiple input channels by Gram-Schmidt (GS) orthogonalization has been the subject of intense research during the past decade [1]–[5]. The Gram-Schmidt technique (sometimes called the adaptive lattice filter) has been shown to yield superior performance simultaneously in arithmetic efficiency, stability, and convergence times over other adaptive algorithms. Arithmetic efficiency has been especially demonstrated in the filtering of stationary covariance sequences [4], [5]. The stability of the algorithm is enhanced because it does not require the calculation of an inverse covariance matrix as does the sample matrix inversion (SMI) algorithm of Reed, Mallet, and Brennan [6]. An overview of adaptive lattice filters and a large bibliography on this subject are contained in [5].

Most of the research on adaptive lattice filters has concentrated on the processing of stationary covariance sequences especially in the area of discrete-time linear prediction systems

[4], [5]. This implies that if r_{ij} is the i, j element to of the channel-to-channel covariance matrix, then $r_{ij} = r_{i-j}$. Hence, this covariance matrix has the Toeplitz form. In this paper, we consider the efficient processing of channels that are not necessarily stationary with respect to one another; however, the input data on a given channel is assumed stationary with respect to other data in that channel. The algorithm developed below is a multichannel adaptive lattice filter that is structured for arithmetic efficiency in addition to retaining the good stability and fast convergence properties of orthogonalization networks. The structuring for the fast orthogonalization network (FON) presented below is given in [3] and [4] for a small number of input channels. In this paper, we have generalized the structuring technique for an arbitrary number of input channels.

Further applications of the FON technique are discussed in [7]. In this paper it is shown how to transform a multiple steering vector adaptive filtering problem where the adaptive weights are solved for using the SMI algorithm into a multiple channel adaptive lattice filter which used a FON. This technique is further extended to handle the multiple channel adaptive radar Doppler processing problem.

II. GRAM-SCHMIDT DECOMPOSITION

The Gram-Schmidt decomposition technique is well documented in the literature [1]–[5], so we give only a brief review. GS decomposition is illustrated in Fig. 1. The DP functional block stands for a two-input decorrelation processor building block [3]. The output from a given block is statistically decorrelated from the right side input to this block by properly weighting this input and subtracting the weighted input from the left side input. The inputs $X_1, X_2, \dots, X_N$

Fig. 1. Gram-Schmidt decomposition or N -channel decorrelator.

represent N input channels of statistical data that may be correlated from input channel to channel. GS decomposition decorrelates the inputs one at a time from the other inputs by using the basic two-input DP. For example as seen in Fig. 1, in the first level of decomposition, X_N is decorrelated with $X_1, X_2, \dots, X_{N-1}$. Next, the output channel that results from decorrelating X_N with X_{N-1} is decorrelated from the other outputs of the first level of DP's. The decomposition proceeds as seen in Fig. 1 until a final output channel is generated. This output channel is totally decorrelated with the input: $X_2, X_3, \dots, X_N$. Note that the GS decomposition is not unique; i.e., the order in which $X_2, X_3, \dots, X_N$ are decorrelated from X_1 is arbitrary.

For N channels, the total number of DP's needed for GS decomposition is $0.5N(N-1)$; hence, this number of decorrelation weights must be computed. For a digital implementation, these weights are determined sequentially: the first level weights are estimated; then their output data are calculated. These output data are used as inputs to the second level from which the second level weights can be calculated. The output data of the second level are generated by using these weights and the second level input data. The process continues until the $(N-1)$ th level weight and outputs are calculated.

Let us consider the problem of decorrelating each input channel with all the other input channels; i.e. there will be N GS decompositions. For a complete N -channel GS decomposition (or multiple channel lattice filter), if there were no logic behind the structuring of the decomposition, the number of weights would be equal to $0.5N^2(N-1)$. In the following section, we develop an algorithm that requires approximately $1.5N(N-1)$ weights for the same decorrelation process.

For notational purposes, we define the channel input appearing on the right side of the basic two-input DP (Fig. 1) as the input whose output of the building block circuit is decorrelated with the channel appearing on the left side. For the multiple channel case, all inputs appearing to the right of the far left input are decorrelated from the output associated with this input.

III. EFFICIENT ORTHOGONALIZATION STRUCTURES

In this section, we present a methodology of configuring the two input DP's to synthesize the N -channel orthogonalization

network so that numerical efficiency is achieved. We therefore introduce the following notation. A single output channel decorrelator is represented as

$$\text{Ch}_1 = [X_1, X_2, X_3, \dots, X_N]$$

where $X_2, X_3, \dots, X_N$ are decorrelated from X_1 and X_N is decorrelated first, X_{N-1} is decorrelated second, and so on. Fig. 1 showed the structure of this decorrelator. The channel variable Ch_1 references this structure to channel 1 or the X_1 channel. The $X_n, n = 1, 2, \dots, N$ are called the elements of the structure.

Numerical efficiency of the algorithm to be presented is achieved by taking advantage of redundancies that can occur for two different decorrelator structures. For example, let there be eight channels. Channels 1 and 4 can be generated as follows:

$$\text{Ch}_1 = [X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8],$$

$$\text{Ch}_4 = [X_4, X_3, X_2, X_1, X_5, X_6, X_7, X_8].$$

Note that Ch_1 and Ch_4 have the same four input channels at the far right. In the actual implementation, the substructure associated with these four rightmost channels can be shared by Ch_1 and Ch_4 as illustrated in Fig. 2. In fact, anytime two channels have exactly the same far-right channels as indicated by the decorrelator structure, the substructure associated with these far-right elements can be shared in the implementation process. Let $N = 2^m$. In general we can configure 2^{m-1} output channels to have the common substructure of 2^{m-1} input channel elements, 2^{m-2} output channels to have the common substructure of 2^{m-2} input channel elements, and so on.

The following algorithm sequentially generates structures that can be implemented in a numerically efficient manner.

Step 1	Generate root structure: $[X_1, X_2, \dots, X_{2^m}]$.
Step 2	Generate a structure that is the inverted order of root structure: $[X_{2^m}, \dots, X_1]$.
Step 3	Generate two structures from the preceding two structures that have the first 2^{m-1} elements of the preceding structures in inverted order. All other elements remain the same.
Step 4	Generate four structures from the preceding four structures that have the first 2^{m-2} elements of the preceding structures in inverted order. All other elements remain the same.
	• • •
Step k	Generate 2^{k-2} structures from the preceding 2^{k-2} structures that have the first 2^{m-k+2} elements of the preceding structures in inverted order. All other element remain the same.
	• • •
Step $m+1$	Generate 2^{m-1} structures from the preceding 2^{m-1} structures that have the first two

Fig. 2. Example of substructure sharing.

elements of the preceding structures in inverted order. All other elements remain the same.

For example, if $N = 2^3$ (where $m = 3$), the following structures would be generated sequentially by using the above procedure:

- Step 1— $[X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8] = Ch_1$
 Step 2— $[X_8, X_7, X_6, X_5, X_4, X_3, X_2, X_1] = Ch_8$
 Step 3— $[X_4, X_3, X_2, X_1, X_5, X_6, X_7, X_8] = Ch_4$
 $[X_5, X_6, X_7, X_8, X_4, X_3, X_2, X_1] = Ch_5$
 Step 4— $[X_2, X_1, X_3, X_4, X_5, X_6, X_7, X_8] = Ch_2$
 $[X_7, X_8, X_6, X_5, X_4, X_3, X_2, X_1] = Ch_7$
 $[X_3, X_4, X_2, X_1, X_5, X_6, X_7, X_8] = Ch_3$
 $[X_6, X_5, X_7, X_8, X_4, X_3, X_2, X_1] = Ch_6$.

Note that channels 1, 2, 3, 4 have the substructure associated with X_5, X_6, X_7, X_8 and that channels 5, 6, 7, 8 have the substructure associated with X_4, X_3, X_2, X_1 . Also note that Ch_1 and Ch_2 have the same six-element substructure as do the channel pairs: (Ch_4, Ch_3) , (Ch_8, Ch_7) , and (Ch_5, Ch_6) . Fig. 3 illustrates complete realization of the eight output channels.

Each two input DP's of the FON as depicted in Fig. 3 has a weight associated with it. The number of DP's or weights associated with a FON can be found by considering the number of DP's at each level of the network. From Fig. 3, we see that the number of levels equals $N - 1$. The total number of DP's N_{DP} needed for a FON is derived by adding the number of DP's at each level. It can be shown [7] that

$$N_{DP} = 1.5N(N-1) - 0.5N \log_2 N, \quad N = 2^m. \quad (1)$$

The total number of operations associated with a FON depends on whether the algorithm is implemented 1) to update

recursively the weights as new data samples arrive or 2) to calculate the weights as a function of a block of N_s data samples in each of the N channels. A comparison of two implementations is made in [7] by using the FON technique versus the sample matrix inversion. This comparison showed that the two techniques are approximately equal in numerical efficiency. Again, however, the FON technique has numerical stability advantages over the SMI technique.

If $2^{m-1} < N < 2^m$, then the structuring for multiple channel orthogonalization can be derived by first generating all the structures for 2^m inputs and then deleting the X_n , $n > 2^m - N$, elements from these structures. Hence, a FON can be constructed for an arbitrary number of inputs.

IV. SOFTWARE ALGORITHM

A software algorithm called the FON algorithm has been devised that generates the 2^m decorrelator output channels from the 2^m input channels by using the common substructures of the various channels as described in Section III (note that $N = 2^m$). Let each of the 2^m input channels have N_s sample points. Thus in the n th channel, $X(1, n)$, $X(2, n)$, $\dots$, $X(N_s, n)$ are observed. Note, the algorithm to be presented uses block processing of the input data.

The algorithm requires at various points to reduce 2^k input channels to 2^{k-1} output channel through a partial orthogonalization, i.e., the 2^{k-1} rightmost input channels are decorrelated with the 2^{k-1} leftmost input channels. If $U(I, J)$ are the input samples where $J = 1, 2, \dots, 2^k$ indicates the channel number, and $I = 1, 2, \dots, N_s$ is the sample index, the following software algorithm called the k th order partial orthogonalization algorithm generates the desired 2^{k-1} output channels:

- 1) Set $V^{(0)}(I, J) = U(I, J)$; $J = 1, 2, \dots, 2^k$; $I = 1, 2, \dots, N_s$
- 2) Set $h = 1, J_0 = 2^k$
- 3) Calculate recursively

$$V^{(h)}(I, J) = V^{(h-1)}(I, J) - V^{(h-1)}(I, J_0) W^{(h)}(J)$$

where

$$W^{(h)}(J) = \frac{\sum_{M=1}^{N_s} V^{(h-1)}(M, J) V^{(h-1)*}(M, J_0)}{\sum_{M=1}^{N_s} |V^{(h-1)}(M, J_0)|^2}$$

$$J = 1, 2, \dots, 2^k - h; \quad I = 1, 2, \dots, N_s$$

- 4) Set $h = h + 1, J_0 = J_0 - 1$
- 5) If $h \leq 2^{k-1}$ GO TO 3
- 6) End.

The final outputs are contained in $V^{(2^{k-1})}(I, J)$. Note that the asterisk indicates the complex conjugate.

The FON algorithm also requires that the input channels, $X_1, X_2, \dots, X_N$, be commutated so that the final output channels of the FON are properly aligned. (This problem also occurs with the fast Fourier transform (FFT) algorithm, but a commutation algorithm matches the proper output channel

Fig. 3. Complete realization of an eight-channel FON.

with the input channel.) By properly aligned, we mean that the k th output channel of the FON algorithm is decorrelated with input channels: $1, 2, \dots, k-1, k+1, \dots, N$. The following algorithm called the commutated indexing algorithm computes the commutated indices and stores them in an N element array called INDEX.

- 1) Set INDEX (1) = 1, INDEX (2) = 2
- 2) Set $J = 1$
- 3) Set INDEX ($2^J + k$) = INDEX (k) + $3 \cdot 2^{J-1}$; $k = 1, \dots, 2^{J-1}$
- 4) Set INDEX ($2^J + k$) = INDEX (k) + 2^{J-1} ; $k = 2^{J-1} + 1, \dots, 2^J$
- 5) $J = J + 1$
- 6) If $J < m$ GO TO 3
- 7) End.

After defining these preliminary algorithms, we now give the complete FON algorithm.

- 1) Input $X(I, J)$ array; $J = 1, 2, \dots, 2^m, I = 1, 2, \dots, N_s$
- 2) Calculate the commutated indices by using the Commutated Indexing algorithm.
- 3) Transfer $X(I, \text{INDEX}(J)) \rightarrow Y(I, J)$
 $J = 1, 2, \dots, 2^m, I = 1, 2, \dots, N_s$
- 4) Set $k = m$
- 5) Set $L = 0$
- 6) Transfer $Y(I, J + 2^k L) \rightarrow U(I, J)$
 $J = 1, 2, \dots, 2^k; I = 1, 2, \dots, N_s$
- 7) Calculate $V^{(2^{k-1})}(I, J)$; $J = 1, 2, \dots, 2^{k-1}, I = 1, 2, \dots, N_s$ by using the k th order partial orthogonalization algorithm.
- 8) Transfer $V^{(2^{k-1})}(I, J) \rightarrow T(I, J)$
 $J = 1, 2, \dots, 2^{k-1}, I = 1, 2, \dots, N_s$
- 9) Transfer $Y(I, 2^k L + J) \rightarrow U(I, 2^k + 1 - J)$
 $J = 1, 2, \dots, 2^k; I = 1, 2, \dots, N_s$
- 10) Calculate $V^{(2^{k-1})}(I, J)$; $J = 1, 2, \dots, 2^{k-1}, I = 1, 2, \dots, N_s$ by using the k th order partial orthogonalization algorithm.
- 11) Transfer $T(I, J) \rightarrow Y(I, 2^k L + J)$

- $J = 1, 2, \dots, 2^{k-1}, I = 1, 2, \dots, N_s$
- 12) Transfer $V^{(2^{k-1})}(I, J) \rightarrow Y(I, 2^k L + 2^{k-1} + I)$
 $J = 1, 2, \dots, 2^k; I = 1, 2, \dots, N_s$
- 13) $L = L + 1$
- 14) If $L \leq 2^{m-k}$ GO TO 6
- 15) $k = k - 1$
- 16) If $k > 0$ GO TO 5
- 17) End.

V. PARALLEL PROCESSING

Fig. 3 shows that there are exactly $N - 1$ levels of two-input DP's associated with a FON, and [7] showed that the maximum number of two-input DP's per level is $2N - 2$ and the minimum number is N . Let L_k equal the number of DP's on the k th level. For the block processing algorithm, we see that as the data (all $N_s L_{k-1}$ data points) are processed through the k th level, the input data may be discarded and the output data ($N_s L_k$ data points) become the new input data set. Hence $2N - 2$ parallel two-input DP's could be configured as seen in Fig. 4.

We allow these $2N - 2$ DP's to simultaneously perform all of the two-input DP's at a given level as it was shown in Fig. 3. However, this requires a routing algorithm whose function it is to see that the L_{k-1} input data channels are inputted into the proper DP's. We do not describe the routing algorithm in this paper. From Fig. 4, we see that the output data are stored back into the input data memory bank. After sequencing $N - 1$ times through the DP's, the algorithm is finished.

It is shown in [7] that the processing time through a DP is proportional to the number of data points per input channel N_s . Since the bank of $2N - 2$ DP's is used $N - 1$ times, the total processing time T , by using a parallel arithmetic architecture (Fig. 4), is approximately proportional to the number of input channels and the number of data points per channel; that is, $T \sim NN_s$. Hence, parallel processing can significantly decrease the processing time. This reduction occurs because of the inherent structure of the FON. In addition, the number of parallel two-input DP's required is $2N - 2$ which reduces the hardware requirements of a fully implemented FON.

Fig. 4. Parallel processing architecture of the FON.

REFERENCES

- [1] B. L. Lewis and F. F. Kretschmer Jr., unpublished work of limited distribution, dating back to Feb. 1974.

- [2] R. A. Monzingo and T. W. Miller, *Introduction of Adaptive Arrays*. New York: Wiley, 1980, ch. 8.
 [3] W. F. Gabriel, "Building block for an orthonormal-lattice-filter adaptive network," NRL Rep. 8409, July 10, 1980.
 [4] M. A. Alam, "Orthonormal lattice filter—A multistage, multichannel estimation technique," *Geophys.*, vol. 43, pp. 1368–1383, Dec. 1978.
 [5] B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, vol. 70, no. 8, pp. 829–867, Aug. 1982.
 [6] I. S. Reed, J. D. Mallet, and L. E. Brennan, "Rapid convergence rate in adaptive arrays," *IEEE Trans. Aerospace Electron Syst.*, vol. AES-10, pp. 853–863, Nov. 1974.
 [7] K. Gerlach, "Multiple channel adaptive filtering using a fast orthogonalization network: An application of efficient pulsed Doppler radar processing," NRL Rep. 8840, Sept. 1984.

Karl Gerlach (M'81), for a photograph and biography please see page 403 of this issue.