Report Number: WUCS-87-19

1987-01-01

A Novel MVDR Beamforming Algorithm

Authors: Adam W. Bojanczyk and Franklin T. Luk

We propose a novel algorithm and architecture for minimum variance distortions less response (MVDR) beamforming. Our approach extracts the least squares residual element directly, and requires only one systolic array for the many look directions.

A NOVEL MVDR BEAMFORMING ALGORITHM

Adam W. Bojanczyk and Franklin T. Luk

WUCS-87-19

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This paper was presented at the 1987 SPIE Conference in San Diego, August 17-21, 1987.

Work of the second author was supported in part by the SDIO/IST and managed by the Army Research Office under contract DAAL03-86-K-0109.

A Novel MVDR Beamforming Algorithm

Adam W. Bojanczyk

School of Electrical Engineering
Cornell University
Ithaca, New York 14853

Franklin T. Luk

School of Electrical Engineering
Cornell University
Ithaca, New York 14853

ABSTRACT

We propose a novel algorithm and architecture for minimum variance distortionless response (MVDR) beamforming. Our approach extracts the least squares residual element directly, and requires only one systolic array for the many look directions.

Keywords and Phrases: MVDR beamforming, constrained least squares, systolic arrays.

Work of the second author was supported in part by the SDIO/IST and managed by the Army Research Office under contract DAAL03-86-K-0109

1. Introduction

We consider the minimum variance distortionless response (MVDR) procedure for adaptive beamforming [1], [5]-[9]. The method can be formulated as a sequence of closely related constrained least squares problem. Given a sequence of p-dimensional data vectors $\{x(i)\}$, define recursively a sequence of data matrices $\{X(i)\}$ by

$$X(1) = x^T(1)$$

and

$$X(i) = \begin{pmatrix} X(i-1) \\ x^T(i) \end{pmatrix}.$$

For each $n \times p$ data matrix $X(n)$, we are interested in a set of optimization problems :

$$|| X(n)\ w^{(k)}(n)\ ||_2 = \min \qquad s.t. \quad d^{(k)T} w^{(k)}(n) = \mu^{(k)}\ , \tag{1.1}$$

where $d^{(k)}$ is a given direction vector and $\mu^{(k)}$ a given scalar. Corresponding to every direction vector $d^{(k)}$, our principal object of interest is the current least squares residual element $e_n^{(k)}(n)$, i.e., the last element of the residual vector

$$e^{(k)}(n) \equiv X(n) w^{(k)}(n).$$

Recently, Schreiber [7] proposed an efficient adaptive procedure for directly extracting the residual element. His procedure is based on the process for updating the Cholesky factorization of the covariance matrix $X^T(n)X(n)$, and it requires order of $p^2 + pK$ operations per iteration where K is the number of direction vectors. However, it is not clear how one can efficiently realize this procedure on parallel processor arrays.

We present here another adaptive algorithm for direct extraction of the residual element, by building on previous work of McWhirter [4], [5] for the unconstrained case. Although the algorithm requires more operations per iteration than the algorithm proposed by Schreiber, the principal advantage of our approach is that only *one* systolic array is required for the *many* different look directions with a throughput of one residual element per array cycle. Our results are new.

This paper is organized as follows. In §2 we present Schreiber's and our algorithms for the canonical problem. In §3 we outline our algorithm for the general problem, and in §4 we show how our algorithm can be implemented in a pipelined fashion on a systolic array for various look directions and for increasing values of n.

2. Canonical Problem

In this section we describe two different ways for finding the least squares residual element of the canonical problem:

$$|| Xw ||_2 = \min \qquad s.t. \quad d^T w = 1\,, \tag{2.1}$$

where X is an $n \times p$ matrix ($n > p$) that has full column rank. >From here on we shall drop the subscript 2 for the euclidean vector norm. First, we give an outline of the algorithm proposed by Schreiber [7]. Next, we develop a new algorithm based on the ideas introduced in McWhirter [5].

2.1. Schreiber's Algorithm

An explicit formula for the solution of (2.1) can be obtained by using the Lagrange multiplier technique. It can be shown that the unknown weight vector w is given by

$$w = \rho(d)(X^T X)^{-1} d\,, \tag{2.1.1}$$

where $\rho(d)$ is a scalar satisfying

$$\rho(d) = \frac{1}{d^T (X^T X)^{-1} d}\,.$$

The covariance matrix $X^T X$ has a Cholesky factorization:

$$X^T X = L L^T\,, \tag{2.1.2}$$

where L is lower triangular. Hence we get

$$\rho(d) = \frac{1}{v^T v}\,,$$

where v solves the triangular system of linear equations

$$Lv = d\,. \tag{2.1.3}$$

Let x^T denote the nth row of X. Then, for the least squares residual element e_n, we have

$$e_n = x^T w = (y^T v)/(v^T v), \tag{2.1.4}$$

where y solves the triangular system of linear equations

$$Ly = x\,. \tag{2.1.5}$$

Thus, in order to compute e_n, it suffices to know the Cholesky factor L of X^TX. We show how the factor can be computed recursively. Let

$$X = \begin{pmatrix} \widetilde{X} \\ x^T \end{pmatrix}. \tag{2.1.6}$$

Suppose that we know the Cholesky decomposition of $\widetilde{X}^T\widetilde{X}$:

$$\widetilde{X}^T\widetilde{X} = \widetilde{L}\widetilde{L}^T,$$

and the solution $\tilde{v}$ to the triangular system

$$\widetilde{L}\tilde{v} = d.$$

The Cholesky factor L of X^TX can be obtained from $\widetilde{L}$ and x by applying the updating technique of Gill et al. [3]. Specifically, a sequence of $(n-1)$ plane rotations is applied to

$$\begin{pmatrix} \widetilde{L}^T \\ x^T \end{pmatrix}$$

to transform it to the upper triangular form

$$\begin{pmatrix} L^T \\ 0^T \end{pmatrix}.$$

Let Q denote the product of such $(n-1)$ rotations. Then

$$Q\begin{pmatrix} \widetilde{L}^T \\ x^T \end{pmatrix} = \begin{pmatrix} L^T \\ 0^T \end{pmatrix}. \tag{2.1.7}$$

The matrix L is the desired Cholesky factor of X^TX. It turns out that the vector v can also be computed recursively. Note that

$$d = \begin{pmatrix} \widetilde{L}, x \end{pmatrix}\begin{pmatrix} \tilde{v} \\ 0 \end{pmatrix} = \begin{pmatrix} \widetilde{L}, x \end{pmatrix}Q^TQ\begin{pmatrix} \tilde{v} \\ 0 \end{pmatrix} = \begin{pmatrix} L, 0 \end{pmatrix}\begin{pmatrix} v \\ \xi \end{pmatrix},$$

where ξ is a certain scalar. Thus,

$$\begin{pmatrix} v \\ \xi \end{pmatrix} = Q\begin{pmatrix} \tilde{v} \\ 0 \end{pmatrix}. \tag{2.1.8}$$

This suggests the following algorithm.

Algorithm 1 (Canonical Problem)

Initialization

1. Compute the Cholesky factor of the $p \times p$ covariance matrix $X(p)^T X(p)$.
2. Compute $v(p)$ by solving the triangular system $L(p)v(p) = d$.

For every new sample $x(i)$, $i = p+1, p+2, \cdots$, do

3. Update $L(i-1)$, i.e., compute $L(i)$ and $Q(i)$.
4. Solve the triangular system $L(i)y(i) = x(i)$.
5. Update $v(i-1)$ according to (2.1.8).
6. Form $e_i(i) = (y^T(i)v(i))/(v^T(i)v(i))$.

On inspection, it is easy to see that Algorithm 1 requires order of $p^2 + p$ operations per iterative step.

2.2. Our Algorithm

Now we describe an alternative way for computing e_n. Construct a sequence of plane rotations P_{12}, P_{23}, $\cdots$, $P_{p-1,p}$ that annihilate the leading nonzero elements of the direction vector:

$$d^T(\, P_{12}\, P_{23} \cdots\ P_{p-1,p}) = \|\, d\,\|\, e_p^T, \tag{2.2.1}$$

where e_p denotes the pth unit coordinate vector. Let

$$P = P_{12}\, P_{23} \cdots\ P_{p-1,p}\ .$$

Problem (2.1) thus simplifies to

$$\|\, XPv\ \| = \min \qquad s.t. \quad v_p = 1\ /\|\, d\,\|, \tag{2.2.2}$$

where

$$v \equiv P^T w.$$

Let

$$X = \begin{pmatrix} \widetilde{X} \\ x^T \end{pmatrix},$$

and suppose that the QR decomposition of the $(n-1)\times p$ matrix $\widetilde{X}$ is known:

$$\widetilde{X} = \widetilde{Q}\begin{pmatrix} \widetilde{U} \\ 0 \end{pmatrix}.$$

We get

$$XP = \begin{pmatrix} \widetilde{Q} & 0 \\ 0^T & 1 \end{pmatrix}\begin{pmatrix} \widetilde{U} \\ 0 \\ x^T \end{pmatrix}P = \hat{Q}\begin{pmatrix} H \\ 0 \\ \hat{x}^T \end{pmatrix}, \tag{2.2.3}$$

where

$$H = \widetilde{U}P \qquad \text{and} \qquad \hat{x}^T = x^T P\ .$$

We now apply an orthogonal transformation Z to triangularize the upper Hessenberg matrix H and to annihilate the leading elements of the vector $\hat{x}$. It is *essential* that we do not annihilate the vector completely. The matrix Z is composed of two sequences of rotations $\{J_{l,l+1}\}$ and $\{K_{l,n}\}$, operating in the $(l,l+1)$ and (l,n) planes, respectively. For example, the transformation $J_{l,l+1}$ equals the identity matrix except for four strategic elements in the pivoting positions :

$$\begin{pmatrix} c_l^J & s_l^J \\ -s_l^J & c_l^J \end{pmatrix}.$$

The order of annihilations is very important :

$$K_{p-1,n}^T\ J_{p-1,p}^T \cdots\ K_{2n}^T\ J_{23}^T\ K_{1n}^T\ J_{12}^T \begin{pmatrix} H \\ 0 \\ \hat{x}^T \end{pmatrix} = \begin{pmatrix} R & y \\ 0^T & \sigma \\ 0^T & 0 \\ 0^T & \tau \end{pmatrix}. \tag{2.2.4}$$

Problem (2.2.2) thus simplifies to

$$|| R\hat{v} + y || = \min \ , \tag{2.2.5}$$

where

$$v = \begin{pmatrix} \hat{v} \\ v_p \end{pmatrix},$$

and the "transformed" residual vector is

$$\sigma \ e_p + \tau \ e_n \ ,$$

where e_n denotes the nth unit coordinate vector. The desired residual element is therefore given by

$$\begin{aligned} e_n^T \hat{Q} (\ J_{12} \ K_{1n} \ J_{23} \ K_{2n} \cdots J_{p-1,p} \ K_{p-1,n}) \ (\sigma \ e_p + \tau \ e_n) \\ = (\ c_1^K \ c_2^K \cdots c_{p-1}^K) \ \tau \ , \end{aligned} \tag{2.2.6}$$

a result very similar to that for the unconstrained case [5], [9]. Finally, as a preparation for the next step of the adaptive procedure, the QR factorization of X is computed from the QR factorization of $\widetilde{X}$ and the sample vector x. Here the standard updating technique due to Gill et al. [3] is used. A sequence of rotations $\{S_{i,n}\}$, $i = 1, \cdots, p$, is applied to transform the matrix $[\widetilde{U}^T, 0, x]^T$ into the upper triangular form $[U^T, 0, 0]^T$,

$$S_{1,n} S_{2,n} \cdots S_{p,n} \begin{pmatrix} \widetilde{X} \\ 0 \\ x^T \end{pmatrix} = \begin{pmatrix} U \\ 0 \\ 0 \end{pmatrix}.$$

With the knowledge of $\{P_{i,i+1}\}$ and the upper triangular matrix U the recursive procedure can be repeated for a new sample vector $x(n+1)$. This suggests the following algorithm.

Algorithm 2 (Canonical Problem)

Initialization

1. Compute rotations $\{P_{j,j+1}\}$ as defined by (2.2.1).

2. Using updating technique determine the upper trapezoidal factor $U(p-1)$ of the QR factorization of $X(p-1)$.

For each new sample $x(i)$, i=p,p+1, $\cdots$ do

3. Determine $H(i)$ and $\hat{x}(i)$.

4. Triangularize the upper Hessenberg matrix $H(i)$ and annihilate the leading elements of the vector $\hat{x}(i)$.

5. Compute the residual element $e_i(i)$ using (2.2.6).

6. Compute $U(i)$ from $U(i-1)$ and $x(i)$.

It is easy to see that, for the canonical problem, Algorithm 2 also requires order of $p^2 + p$ operations.

3. General Case

In practice there are many direction vectors $\{d^{(k)}\}$, $k = 1, \cdots, K$, for each the procedures described in the previous section have to be repeated.

3.1. Schriber's Algorithm

There are two stages in the general procedure, the initialization and the recursive stage. In the initialization stage, the Cholesky factor $L(p)$ of the initial covariance matrix $X^T(p)X(p)$ and the solutions $v(p)$ to the triangular systems $L(p)v(p) = d^{(k)}$, k=1, $\cdots$ K are computed. In the recursive stages, the Cholesky factor and the solutions to the triangular systems are updated. In addition, a single triangular system is solved. Residual elements are obtained as quotients of appropriate scalar products. An outline of the algorithm is given below.

Algorithm 1 (General case)

Initialization {

1. Compute the Cholesky factor of pxp covariance matrix $X^t(p)X(p)$

2. Compute $v(p)$ by solving the triangular system $L(p)v(p) = d$

}

For every new sample $x(i)$, i=p+1,p+2, $\cdots$ do {

3. Update $L(i-1)$, i.e., compute $L(i)$ and $Q(i)$

4. Solve the triangular system $L(i)y(i) = x(i)$

For $k = 1, \cdots K$ do {

5. Update $v^{(k)}(i-1)$ according to (2.1.10)

6. Form $e_i^{(k)}(i) = (y^T(i)v^{(k)}(i))/(v^{(k)T}(i)v^{(k)}(i))$

}

}

Because only one updating of Cholesky factor is performed and only one triangular system is solved for each new sample, the total cost of a single recursive step is of order $p^2 + Kp$ multiplications and additions.

This procedure is very efficient as far as the number of arithmetic operations is concerned. Moreover, fast systolic algorithms exist for updating, solution of triangular systems and scalar products. However, it is not clear how one could combine these systolic arrays into a system without incurring delays between successive stages. Part of the problem here is that while systolic updating requires top-to-bottom processing, solving an upper triangular system requires bottom-to-top processing which means that these stages cannot be pipelined. In Section 4 we will show that Algorithm 2 can be pipelined.

3.2. New Algorithm

Again, there are two stages in the general procedure, the initialization and the recursive stage. In the initialization stage direction vectors are rotated onto e_p and $X(p-1)$ is transformed into upper trapezoidal form. In the recursive stage residual elements corresponding to the current sample matrix and individual directions are computed followed by updating of the triangular factor of the sample matrix. An outline of the algorithm is given below.

Algorithm 2 (General Case)

Initialization {

1. For $k = 1, \cdots, K$ compute rotations $\{P_{i,i+1}^{(k)}\}$ as

defined by (2.2)

2. Using updating technique determine the upper trapezoidal factor $U(p-1)$ of the QR factorization of $X(p-1)$

}

For each new sample $x(i)$, $i = p, p+1, \cdots$, repeat {

For $k = 1, \cdots, K$ repeat {

3. Determine $H^{(k)}(i)$ and $\hat{x}^{(k)}$

4. Triangularize the upper Hessenberg matrix $H^{(k)}$ and annihilate the leading elements of the vector $\hat{x}^{(k)}$

5. Compute the residual element $e_i^{(k)}(i)$ using (2.6)

}

6. Compute $U(i)$ from $U(i-1)$ and $x(i)$

}

The algorithm described here is a combination of the standard approach to the linearly constrained linear least squares problem and the method of McWhirter for directly extracting the residual element. Each recursive step requires order of $Kp^2 + Kp$ arithmetic operations. This is K times as many as in Schreiber's algorithm. However, as will be shown in the next section, our algorithm can be efficiently realized on a single square array of processors with the throughput of one residual element per array cycle. It is not know to these authors whether Schreiber's algorithm or its modification allows similar realization.

4. Systolic Implementation

In this section we outline a possible systolic realization of Algorithm 2. We shall not specify details of the implementation but rather give general ideas from which it should be clear that an efficient implementation is possible.

For a systolic implementation of our algorithm, we propose a mesh-connected trapezoidal array of processors. As the algorithm is heterogeneous, it is not surprising that the array is, too; different regions of the array execute different subtasks of the algorithm. In order to attain such flexibility we postulate that the cells be microprogrammable, i.e., a program executed by a cell can be changed if required. By allowing such generality we can guarantee a very smooth flow of data.

The first subtask to be realized is Step 1. This step can be viewed as a pre-processing step which is to be executed only once. The results of the pre-processing, namely rotation coefficients, have to be stored in auxiliary registers in a manner that facilitate an efficient access when needed. We shall assume that rotation parameters are available and can be accessed in the space-time order specified later in this section.

The skeleton of the array is the pxp triangular array of Gentleman, Kung and McWhirter [2], [5]. However, the individual cells are more flexible in the sense that, in the course of

computation, they can assume several different states corresponding to different steps of the algorithm. The triangular array is augmented by an additional subdiagonal of $p-1$ cells to accommodate subdiagonal elements of upper Hessenberg matrices, and by an additional row of p cells to accommodate new sample vectors. In Figure 1, r's represent the additional row, h's represent the additional subdiagonal and t's represent the triangular array.

$$
\begin{array}{ccccccc}
r & r & r & r & r & r \\
t & t & t & t & t & t \\
h & t & t & t & t & t \\
\cdot & h & t & t & t & t \\
\cdot & \cdot & h & t & t & t \\
\cdot & \cdot & \cdot & h & t & t \\
\cdot & \cdot & \cdot & \cdot & h & t
\end{array}
$$

Figure 1. Trapezoidal array of processors

Step 2 can be treated either as a pre-processing step or merged with the iterative steps. For ease of exposition we assume that, in addition to rotation parameters, the upper trapezoidal factor $U(p-1)$ of the QR factorization of $X(p-1)$ is available and stored row by row in the first $p-1$ rows of the t subarray.

When a new sample vector arrives it is stored in the r processors. The array is ready to execute Steps 3, 4 and 5 which are repeated for each direction vector.

First, the rotation parameters corresponding to the first direction vector are brought from the auxiliary registers. The rotations are applied to the new sample vector and the current triangular factor. This operation is executed column by column in left-right and top-down directions. New subdiagonal elements are created. The results, the transformed sample and the Hessenberg matrix, are stored *in place*.

Remark: It is important that the original sample vector and the triangular matrix be not

destroyed but stored *in place* for later transformations.

It is easy to see that a single *wavefront* of activities propagates through the array in the south-east direction, starting from the north-west corner of the array. This wavefront corresponds to the execution of Step 3.

When the first wavefront has traveled far enough the execution of Step 4 may begin. Annihilation of subdiagonal elements interlaced with annihilation of the transformed sample vector form the second wavefront of activities. The operations are executed row by row with the rows of the Hessenberg matrix staying in place while the transformed sample vector travels in top-down direction. Again, the wavefront of activities propagates in the south-east direction and follows after the first wavefront. The appropriate product of cosines is formed by the diagonal processors while the multiplier τ is computed by the rightmost column of cells. Finally, the residual element is determined in the south-east corner cell.

Note that in the course of computation only the elements laying below and including the second wavefront are required. The elements above the second wavefront are not needed. Moreover, the cells above the wavefront are idle. Thus, the computation corresponding to the second direction vector can start right after the appropriate rotation coefficients are fetched from the auxiliary registers - now it becomes clear why the original sample vector and the current triangular factor must not be destroyed.

The sequence of operations, postmultiplications followed by the recovery of the triangular form, is repeated until all direction vectors are processed.

When all direction are processed the array is set to update the triangular factor, i.e., to realize Step 6. At this moment a single wavefront is formed which again propagates in the usual south-east direction. A new sample vector can now be received and the whole cycle repeated.

Once the array has been initialized it creates a number of parallel wavefronts. Note that two wavefronts are created for each direction vector plus one wavefront for the updating

operation, for the total number of $2K + 1$ wavefronts per cycle. Wavefronts travel with a constant speed through the array in the south-east direction. Wavefronts are pipelined but are separated by a certain constant distance which is necessary to guarantee that different wavefronts do not interfere with each other. Thus it can be concluded that one residual element is computed in the south-east cell, approximately, every second wavefront, or, the throughput is one residual element per array's unit of time where array's unit of time corresponds to the time separation of two consecutive wavefronts.

References

[1] A.W. Bojanczyk and F.T. Luk, "A unified systolic array for adaptive beamforming," Technical Report WUCS-87-8, Department of Computer Science, Washington University, St Louis, MO, 1987.

[2] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays," *Real Time Signal Processing IV*, T.F. Tao, Ed., Proc. SPIE, vol. 298 (1981), pp. 19-26.

[3] P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders, "Methods for Modifying Matrix Factorizations", *Math. Comp.*, vol. 28 (1974), pp. 505-535.

[4] F.T. Luk and S. Qiao, "Analysis of a recursive least-squares signal processing algorithm," *Advanced Algorithms and Architectures for Signal Processing I*, J.M. Speiser, Ed., Proc. SPIE, vol. 696 (1986), pp. 88-93.

[5] J.G. McWhirter, "Recursive least-squares minimization using a systolic array," *Real Time Signal Processing VI*, K. Bromley, Ed., Proc. SPIE, vol. 431 (1983), pp. 105-112.

[6] J.G. McWhirter and T.J. Shepherd, "A systolic array for linearly constrained least-squares problems," *Advanced Algorithms and Architectures for Signal Processing I*, J.M. Speiser, Ed., Proc. SPIE, vol. 696 (1986), pp. 80-87.

[7] R. Schreiber, "Implementation of Adaptive Array Algorithm", *IEEE Trans. Acoustics, Speech, and Signal Processing*, ASSP-34 (1986), pp. 1038-1045.

[8] R. Schreiber and P.J. Kuekes, "Systolic linear algebra machines in digital signal processing," in *VLSI and Modern Signal Processing*, S.Y. Kung, H.J. Whitehouse and T. Kailath, Eds., Prentice-Hall, Englewood Cliffs, NJ (1985), pp. 389-405.

[9] C.R. Ward, P.J. Hargrave and J.G. McWhirter, "A novel algorithm and architecture for adaptive digital beamforming," *IEEE Trans. Antennas and Propagation*, AP-34 (1986), pp. 338-346.