An Adaptive Moving Window Multiple Sidelobe Canceller for Seismic Data

Gregory Robertson, Dr. R. Lynn Kirlin and Dr.W.-S. Lu

Department of Electrical and Computer Engineering
University of Victoria, P.O. Box 3055,
Victoria, B.C., Canada, V8W 3P6
FAX: (604) 721-6052

Abstract

The Multiple Sidelobe Canceller (MSC) [3] has a higher resolution than the more common Semblance algorithm allowing it to reveal more details in the data (i.e. seismic data). In a conventional MSC the calculation of a pseudo-inverse of a matrix makes the algorithm very slow for repeated applications. A method is presented to reduce this computational complexity of the MSC when applied to a moving window. Two updating procedures using the change in the data from one window position to the next are shown to reduce the computational complexity of the MSC by more than 75% for a downward movement (with time) and more than 55% for a sideways movement (across traces) of the window. In addition these methods for updating the pseudo-inverse appear to produce more accurate results.

1 Introduction

The major problem with the MSC algorithm, as already mentioned, is the large number of calculations necessary for determining the pseudo-inverse of the auxiliary correlation matrix. The solution to this problem lies in finding a method that updates the actual pseudo-inverse matrix from one iteration to the next.

Any method that is chosen must be versatile enough to handle both the "down-trace" analysis, as well as the much more complicated "across-trace" analysis. The Matrix Inversion Lemma, also called the *Sherman- Morrison- Woodbury formula*, reveals a solution to the problem of updating the pseudo-inverse.

2 The Matrix Inversion Lemma

This inversion lemma [1] can be found in use in such areas as adaptive filtering and optimization. It is extremely useful in this situation with the sliding of an analysis window with only a slight amount of data changing from one window position to the next.

The lemma states that when a matrix can be represented in the form of

$$M = A + B \cdot D \cdot C, \tag{1}$$

then the inverse of the matrix M, can be calculated using the following expression;

$$M^{-1} = A^{-1} - A^{-1}B\left(D + CA^{-1}B\right)^{-1}CA^{-1}. \tag{2}$$

3 The Moving Window

The key for using the matrix inversion lemma in updating the pseudo-inverse is to observe how (1) relates to the change of the correlation matrix as the processing window slides.

Consider the sliding window to be a frame that encloses M traces and N time samples. The data within this frame forms the data matrix X.

$$X = \left[\bar{x}_1\ \bar{x}_2\ \cdots\ \bar{x}_N\right] = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \cdot & \cdot & \cdot & \cdot \\ x_{M,1} & x_{M,2} & \cdots & x_{M,N} \end{bmatrix} \tag{3}$$

The sample correlation matrix can then be calculated as

$$R_x = X \cdot X^T / N. \tag{4}$$

This is a matrix algebra notation equivalent to the average of the outer product of the data vectors $\bar{x}_i$, the columns of X.

It is possible to relate the sample correlation matrix at position k-1 to the sample correlation matrix at k with the use of a difference matrix; i.e.

$$R_x^{(k)} = R_x^{(k-1)} + R_\Delta^{(k)}. \tag{5}$$

By noticing that one can relate the "A" from (1) with the sample correlation matrix $R_x^{(k-1)}$ and the "BDC" with the difference matrix $R_\Delta^{(k)}$, then this lemma can be used to update the pseudo-inverse of the auxiliary correlation matrix when the difference matrix is factored into smaller and simpler matrices.

4 Down-trace Analysis

The easier of the two analysis methods is the analysis "down-trace" (or with time). This is simply due to the fact that one entire vector is dropped and one entire vector is added during each move of the window.

1058-6393/96 $5.00 © 1996 IEEE
Proceedings of ASILOMAR-29

The whole analysis process for analyzing down the traces starts with the initialization of the window matrix $X^{(k)}$ and the pseudo-inverse of the first (k=0) auxiliary correlation matrix $R_a^{\dagger(k)}$. During the first iteration the pseudo-inverse must be calculated using the computationally heavy SVD procedure. Once these initial matrices are determined, the more efficient update algorithm can then take over for the rest of the iterations down the trace. The initial matrices must be re-calculated for each steering direction.

When one defines the dropped vector x_{old} as the first column vector from the previous window matrix $X^{(k-1)}$ and the added vector x_{new} as the last column vector from the current window matrix $X^{(k)}$, then R_Δ in (5) becomes

$$R_\Delta^{(k)} = \left(x_{new}x_{new}^T - x_{old}x_{old}^T \right). \quad (6)$$

In order to use the matrix inversion lemma the difference matrix should be able to be split into a "BDC" configuration. This is easily accomplished by noticing that the difference matrix is a rank 2 matrix. The difference matrix can be simply written as

$$R_\Delta^{(k)} = U^{(k)} \cdot I_2 \cdot V^{(k)}, \quad (7)$$

where I_2 is a 2 by 2 identity matrix and $U^{(k)}$ and $V^{(k)}$ (below) have the dimensions of M by 2 and 2 by M, respectively.

$$U^{(k)} = \left[x_{new} \quad x_{old} \right] \quad (8)$$

$$V^{(k)} = \left[x_{new} \quad -x_{old} \right]^T \quad (9)$$

To make the factors of the difference matrix applicable to the auxiliary correlation matrix, both $U^{(k)}$ and $V^{(k)}$ must be projected onto the noise domain by multiplying these two matrices by the blocking matrix.

$$U_a^{(k)} = B \cdot U^{(k)}, \quad (10)$$

$$V_a^{(k)} = V^{(k)} \cdot B, \quad (11)$$

where B is a blocking matrix defined as

$$B = I_M - \left(u \cdot u^T \right)/M, \quad (12)$$

where I_M is an MxM identity matrix and u is vector of ones. With these definitions, the current auxiliary correlation matrix can be written as

$$R_a^{(k)} = R_a^{(k-1)} + U_a^{(k)} \cdot I_2 \cdot V_a^{(k)}. \quad (13)$$

From this equation the update of the pseudo-inverse of the auxiliary correlation matrix, $R_a^\dagger$, for the "Down the Trace Analysis" becomes

$$R_a^{\dagger(k)} = R_a^{\dagger(k-1)} \cdot (I_M - C), \quad (14)$$

where $C = U_a^{(k)} \cdot A^{-1} \cdot V_a^{(k)} \cdot R_a^{\dagger(k-1)}$, (15)

and $A = I_2 + V_a^{(k)} \cdot R_a^{\dagger(k-1)} \cdot U_a^{(k)}$, (16)

where I_M is an "M x M" identity matrix and A is 2 by 2.

5 Across-trace Analysis

In comparison the across-trace analysis is far more complicated than the down-trace analysis. In the across-trace analysis the problem has changed so that upon each new iteration each vector in the window has one added element and one dropped element. In the previous analysis the effect of the downward slide of the window changed the correlation matrix R_x by a simple delta correlation matrix R_Δ (composed of the difference between the outer product of the new vector and the outer product of the old vector). However, in this new situation there is no simple delta correlation matrix. As before the relationship between the correlation matrices for the data in the previous and current window positions remains the same as (5). The result of the sliding window is a "diagonally upward shift" in the values of the correlation matrix. Therefore in order to obtain an efficient updating procedure for the pseudo-inverse this shifting effect must be taken into consideration.

The correct approach uses a "shifting matrix" in combination with some elementary vector operations. After some careful calculations and observations, the following steps for the algorithm have been determined.

In the first step, as in the previous case with the analysis going down the traces, certain initial calculations had to be made. As before the final stage to this step (k=0) is the storage of the initial X as $X^{(k)}$ and the pseudo-inverse of R_a as $R_a^{\dagger(k)}$, where "†" indicates the pseudo-inverse of a matrix.

After the window position is changed, effecting $X^{(k+1)}$, the next step is to determine an interim "shifted version" of the pseudo-inverse of R_a.

$$\hat{R}_a^{\dagger(k-1)} = S \cdot R_a^{\dagger(k-1)} \cdot S^T, \quad (17)$$

The M by M shifting matrix (shown below) translates the lower right corner M-1 square matrix of the initial matrix to the upper left corner for a new matrix.

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}. \quad (18)$$

As previously discussed, when a matrix can be put into terms of a previous matrix and a difference matrix then the update of the inverse of the matrix can be found assuming one has the proper inverse value for the k-1 cor-

relation matrix, and one can appropriately separate the difference matrix.

In this situation, the proper inverse value of the previous matrix must be the shifted version of the $R_a^{\dagger(k-1)}$. One must now find the values of the $U^{(k)}$ and $V^{(k)}$ matrices, whose product is the $R_\Delta^{(k)}$ matrix, in order to determine the update formula.

If one applies the shifting matrix to the k-1 correlation matrix in (5) and then subtracts the k-1 and k correlation matrices in order to find the difference matrix, one would notice that the difference matrix would have the following form;

$$R_\Delta^{(k)} = \begin{bmatrix} 0 & 0 & \dots & 0 & a_1 \\ 0 & 0 & \dots & 0 & a_2 \\ . & . & . & . & . \\ 0 & 0 & \dots & 0 & a_{M-1} \\ a_1 & a_2 & \dots & a_{M-1} & a_M \end{bmatrix}, \quad (19)$$

where each element, $[a_1, a_2, \dots, a_M]$, is composed of a difference of another two scalars from within the two correlation matrices. For instance, using the notation in equation 1, the value of a_1 would be "$r_{2,M-1} - r_{2,1}$". By finding the relationship between the new and old data of the k and k-1 correlation matrices, the factors of $R_\Delta^{(k)}$ can be determined and updated easily.

When the window moves from one position to the next, the window matrix X drops and adds a "row" of data, or trace. If the dropped row of data y_{old} is defined as the first row in the previous window matrix $X^{(k-1)}$ and if the added row y_{new} is defined as the M^{th} row in the new window matrix $X^{(k)}$, then these two vectors can be used to determine the components contributing to the difference matrix. These two vectors are defined as

$$y_{old} = \left[X^{(k-1)}_{(1,\ \{1,2,\dots,N\})} \right]^T, \quad (20)$$

$$y_{new} = \left[X^{(k)}_{(M,\ \{1,2,\dots,N\})} \right]^T. \quad (21)$$

From these two "N x 1" dimensional vectors, the following vector and three scalars can be defined. The difference vector δ is the mean difference between the two vectors; c_1 is the square root of the mean of the inner product of y_{new}; c_2 is the square root of the mean of the inner product of y_{old}; and c_3 is the sum of the other two scalars:

$$\delta = (y_{new} - y_{old})/N, \quad (22)$$

$$c_1 = \sqrt{\left(y_{new}^T \cdot y_{new} \right)/N}, \quad (23)$$

$$c_2 = \sqrt{\left(y_{old}^T \cdot y_{old} \right)/N}, \quad (24)$$

$$c_3 = c_1 + c_2. \quad (25)$$

With these definitions we have:

$$U^{(k)} = \begin{bmatrix} u & v \end{bmatrix}, \quad (26)$$

$$V^{(k)} = \begin{bmatrix} u & -v \end{bmatrix}^T, \quad (27)$$

$$\text{where } u_{(\{1,2,\dots,M-1\},\,1)} = \left(X^{(k)} \cdot \delta \right)/c_3, \quad (28)$$

$$v_{(\{1,2,\dots,M-1\},\,1)} = u_{(\{1,2,\dots,M-1\},\,1)}, \quad (29)$$

$$u_{(M,\,1)} = c_1, \quad (30)$$

$$v_{(M,\,1)} = -c_2. \quad (31)$$

The "M x 1" dimensional vectors u and v have the first "M-1" elements in common. The values of the first "M-1" elements come from the effect of the sliding of the window on the data within the window, itself. The M^{th} elements of both vectors are different in order to account for the cross term in the multiplication of the $U^{(k)}$ and $V^{(k)}$ matrices. With these values and using equations 10 and 11, the completed equation for the update of the auxiliary correlation matrix can be written

$$R_a^{(k)} = S \cdot R_a^{(k-1)} \cdot S^T + U_a^{(k)} \cdot I_2 \cdot V_a^{(k)}. \quad (32)$$

From this equation and the previous update method, the update of the pseudo-inverse of the auxiliary correlation matrix, $R_a^\dagger$, for the across-trace analysis becomes

$$R_a^{\dagger(k)} = \hat{R}_a^{\dagger(k-1)} \cdot (I_M - C), \quad (33)$$

$$\text{where } C = U_a^{(k)} \cdot A^{-1} \cdot V_a^{(k)} \cdot \hat{R}_a^{\dagger(k-1)}, \quad (34)$$

$$\text{and } A = I_2 + V_a^{(k)} \cdot \hat{R}_a^{\dagger(k-1)} \cdot U_a^{(k)}. \quad (35)$$

6 Comments on Results

In figures 1 and 2, a comparison of some simulated flop count results are shown. The results are presented as a percentage of the ratios of the Semblance (the dashed line) and the Updating MSC (the solid line) to the conventional MSC algorithm. As one can observe in both figures, as the value of M drops with a fixed number of time samples (N = 49), the efficiency of the Updating MSC reduces as compared to the MSC algorithm. In figure 1, with a value of N set to 25 or more, the average number of flops for the Updating MSC is only about 22.56% of that of the MSC algorithm. While for the across-trace analysis results for the same N value, figure 2, show the average number of flops to be slightly higher at about 42.60% of that of the MSC algorithm. Although the flop count comparison for the Semblance method is much smaller than the Updating MSC values, the Updating MSC provides a large calculation efficiency improvement over the MSC algorithm.

7 Conclusion

The Semblance, the MSC, and the Updating MSC algorithms have been tested on real and simulated data. Although the results indicate that the Semblance method is computationally less complex than the two MSC algorithms, it has also been shown that the Semblance algorithm has far less resolution than either of the MSC methods. The updating procedure of the Updating MSC algorithm has shown an improved ability to track changes in the data as compared to the conventional MSC. This updating procedure has also greatly reduced the computational complexity of the MSC algorithm by removing the need to calculate the pseudo-inverse of the auxiliary correlation matrix on each iteration. The Updating MSC has reduced the flop counts to 22.56% and 42.60% of that of the MSC algorithm for the down and across-trace analyses, respectively.

With the better tracking and reduced flop count, the Updating MSC algorithm has been shown to be a very useful implementation for the MSC algorithm when applied to a sliding window configuration.

REFERENCES

[1] Simon Haykin. *Adaptive Filter Theory*, New Jersey: Prentice-Hall Inc., 1991.

[2] R. Lynn Kirlin. *Preliminary Considerations on High Resolution Moving Window Multiple Coherency Estimators*, (Report for Amoco Inc.), March 1993.

[3] Barry D. Van Veen & Kevin M. Buckley. *Beamforming: A Versatile Approach to Spatial Filtering*, IEEE ASSP Magazine, April 1988

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations: Second Edition*, Baltimore, Maryland: Johns Hopkins University Press, 1989

[5] Bart Kosko. *Neural Networks and Fuzzy Systems*, New Jersey: Prentice-Hall Inc., 1992.

[6] M. Mahon, L. Sibul, and H. Valenzuela. *A Sliding Window Update for the Basis Matrix of the QR Decomposition*, IEEE Transaction on Signal Processing, Vol. 41, No. 5, May 1993, pages 1951-1953.

[7] *MATLAB*™, Natick, MA: The MathWorks, Inc., Version 4.1, January 15th, 1993. (A mathematical software package used on SUN™ workstations.)

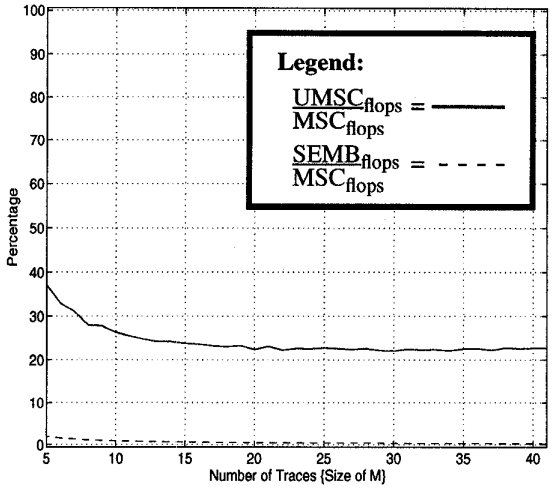

Figure 1: Flop Count Comparison in Percentage for Down-trace Analysis

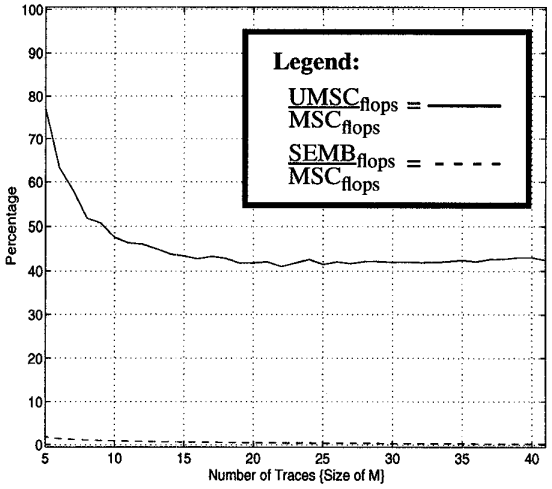

Figure 2: Flop Count Comparison in Percentage for Across-trace Analysis