Solving the SVD updating problem for subspace tracking on a fixed sized linear array of processors

C. Sengupta, J. R. Cavallaro, B. Aazhang

Summary

1 Introduction

Many signal processing algorithms are based on the computation of the eigenstructure (eigenvalues and eigenvectors) of the covariance matrix of a data matrix. Applications include: direction of arrival estimation in array processing, spectral estimation and CDMA synchronization [1]. The advantages of using the eigenvector-based methods (also called subspace based methods) are well-known.

In the usual way of using such algorithms, the covariance matrix is first estimated from the received data. Then, by using some numerical method, the Eigen Value Decomposition (EVD) is computed before applying a subspace based algorithm, such as, MUSIC [6]. Before applying the subspace based algorithm, the EVD is used to separate the signal subspace from the noise subspace. The covariance matrix is estimated as :

$$\hat{R}_n = Y_n' Y_n, \tag{1}$$

where Y_n is the data matrix at time n and $'$ is the conjugate transpose. The data matrix is constructed from the received data vectors :

$$Y_n = \begin{pmatrix} \beta^{n-1} y_0' \\ \vdots \\ \beta y_{n-1}' \\ y_n' \end{pmatrix} = \begin{pmatrix} \beta Y_{n-1} \\ y_n' \end{pmatrix}, \tag{2}$$

where y_n is the data vector at time n and $0 < \beta \leq 1$ is a forgetting factor. In a non-stationary environment the received data vectors are expected to vary with time. This time varying nature of the data is taken into account by using the forgetting factor, β - thus, previously received data vectors are weighted less than the current data vector.

A practical approach to computing the EVD of the covariance matrix is to use the Singular Value Decomposition (SVD) of the data matrix Y_n. The SVD of Y_n is defined as

$$Y_n = U \Sigma V', \tag{3}$$

where, U and V are unitary matrices and Σ is a diagonal matrix of singular values. From this it follows that

$$R_n = Y_n' Y_n = V \Sigma U' U \Sigma V' = V \Sigma^2 V', \tag{4}$$

which is the eigenvalue decomposition of the covariance matrix. Hence, the eigenvalues and eigenvectors can be computed directly from the SVD of the data matrix instead of computing them from an intermediary covariance matrix. For a given wordlength, this alternate formulation would result in superior numerical performance.

In a non-stationary envronment, where the received data varies with time, the SVD of the data matrix needs to be updated to incorporate the change. The updating of the SVD is used to continuously track the time-varying signal or noise subspace (subspace tracking). Thus, our problem is to calculate the SVD of the data matrix Y_n at each time step n, using the SVD of the data matrix Y_{n-1} calculated at time step $n-1$.

As the problem needs to be solved in real time, it is natural to look for a parallel algorithm so that computation time can be reduced by distributing the work among a number of processing units. This paper proposes a parallel scheme for SVD updating that can be implemented on a fixed size array of off-the-shelf processors.

2 Algorithm description

Our algorithm is based on Hestenes method [3] for computing the SVD. In the Hestenes method, starting with any given matrix A, an orthogonal matrix V is built such that AV has orthogonal columns. Thus, $AV = US$, where U has orthonormal columns and S is non-negative and diagonal. The SVD of A is $A = USV'$.

Let us denote US by the single matrix W. All the transformations applied to A to get W, are simultaneously applied to an identity matrix to get the V matrix. So given matrix A, this method yields matrices W and V such that

$$A = WV'. \tag{5}$$

The singular values are equal to the columns norms of matrix W and U can be obtained by normalizing the columns of W. However, since most subspace-based applications require V only, this step is not necessary.

As, new rows are appended to matrix Y_n, its size as well as that of W_n increases with time. It may be noted, in equation 2, less weightage is given to "older" data vectors as compared to recently received data vectors by multiplying with higher powers of β. We take this a step further and consider only the last L data vectors. This will keep the size of Y_n and W_n fixed. The size of the parameters L and β will depend on the application. So, we define Y_n as:

$$Y_n = \begin{pmatrix} \beta^{n-L} y_{n-L+1}' \\ \vdots \\ \beta y_{n-1}' \\ y_n' \end{pmatrix}. \tag{6}$$

Let us assume that, at time step $n-1$, we have calculated the SVD of Y_{n-1} using Hestenes method :

$$Y_{n-1} = W_{n-1} V_{n-1}', \tag{7}$$

where $W_{n-1} = U_{n-1} S_{n-1}$. The computation in time step n is based on the following equations :

$$Y_n = \begin{pmatrix} \beta Y_{n-1} \\ y_n' \end{pmatrix}. \tag{8}$$

Using equation 7 with 8,

$$Y_n = \begin{pmatrix} \beta W_{n-1} \\ y_n' V_{n-1} \end{pmatrix} V_{n-1}'. \tag{9}$$

Let us denote the first matrix on the right hand side of equation 9 by X_n. The columns X_n is orthogonalized using r sweeps of Hestenes method [3]. So we get :

$$Y_n = W_n \tilde{V}_n' V_{n-1}' = W_n V_n', \tag{10}$$

where $V_n = V_{n-1}\tilde{V}_n$.

Simulations of application of this method to the frequency estimation problem (refer section 4) show that a “good” enough approximation is obtained after just one *sweep* of Hestenes method, for each update. Here a *sweep* is defined as the process of orthogonalizing each pair of columns, once, in any order. If only one *sweep* is performed, this algorithm requires $O(n^2)$ computation.

In the next section, we will first discuss a previously proposed alternative scheme for parallel SVD updating. In section 4 we will compare the performance of both schemes for a frequency estimation problem and show that they are practically equivalent. Finally, in section 5, we will discuss the parallelization scheme for our algorithm and the advantages to be gained from it.

3 Related work

To date, the most powerful parallel scheme for SVD updating is the systolic array proposed by Moonen, Van Dooren and Vandewalle [5]. This scheme is based on a SVD updating algorithm proposed in [4] and [2]. This SVD algorithm is constructed by combining QR updating with a Jacobi type SVD diagonalization procedure (Kogbetliantz's algorithm, modified for triangular matrices). As the Hestenes method for SVD is a modified version of Jacobi's method for SVD, we can expect that both algorithms will perform very similarly, when tracking a subspace. This conjecture is substantiated by our simulation results.

However, as the data dependence structure of this algorithm is such that, at each update, the computation using each column depends on the results of the computation using all the columns to its left. Also, computation using each row depends on the results of the computation using all rows above it. Thus, the data cannot be easily partitioned onto a fixed number of processors for processing. The parallelization scheme must depend on pipelining of the problem on a large number of custom-built processing units to get sufficient speed-ups.

However, in the case of our algorithm, the computation using each column is independent of the computation using all other columns. Hence, it can be easily parallelized on a linear array of prcessors to get high speed-ups. This issue is discussed further in section 5.

4 Simulation results

In order to show the tracking capabilities of our algorithm, we will use the examples used by Ferzali and Proakis in [2]. In this example, a covariance eigenstructure based algorithm is used in a spectral estimation application, where the problem is to track the variations in the instantaneous frequencies of a signal $s(n)$ composed of multiple, superimposed, time varying sinusoidal components. The received signal $r(n)$ is the sum of the k sinusoids contaminated with an additive zero mean Gaussian white noise $W(n)$.

$$r(n) = \sum_{i=1}^{k} A_i sin(\omega_i n) + W(n). \tag{11}$$

The data matrix Y_n are constructed from $r(n)$. The eigenvector based estimation algorithm used was MUSIC [6].

In the first experiment, we track an abruptly changing subspace. A sudden jump in frequency is introduced. This will change the subspace suddenly. The first 250 data vectors were generated from

$$r_1(n) = 2cos(2\pi \times .35t) + W(n). \tag{12}$$

Another 250 data vectors were formed from

$$r_2(n) = 2cos(2\pi \times .45t) + W(n). \tag{13}$$

where the frequency of the only sinusoid has jumped from .35 to .45. Data vectors of size denoted by m were constructed. Figure 1.a shows that our proposed scheme and Moonen et al's method perform very similarly with only 1 sweep of their respective SVD algorithms for same values of β.

In the next example, we track a slowly but continuously changing frequency. The data vectors were generated from the received signal

$$\begin{aligned} r_3(n) = 2cos(2\pi \times .10n) + 2cos(2\pi \times (.25 + (inc \times n))n) \\ +2cos(2\pi \times .45n) + W(n). \end{aligned} \tag{14}$$

The gradual change in the second frequency is defined by inc and the value of inc used in our experiments was 0.1/200, that is, inc changes by 0.1 over 200 time steps. Again, both schemes perform very similarly and successfully track all 3 frequencies (Figure 1.b).

5 Parallelization scheme

Our parallelization scheme for the SVD updating problem is based on the solution of singular value problems on an undersized linear array proposed by Schreiber [7], using the Hestenes method [3] for the SVD.

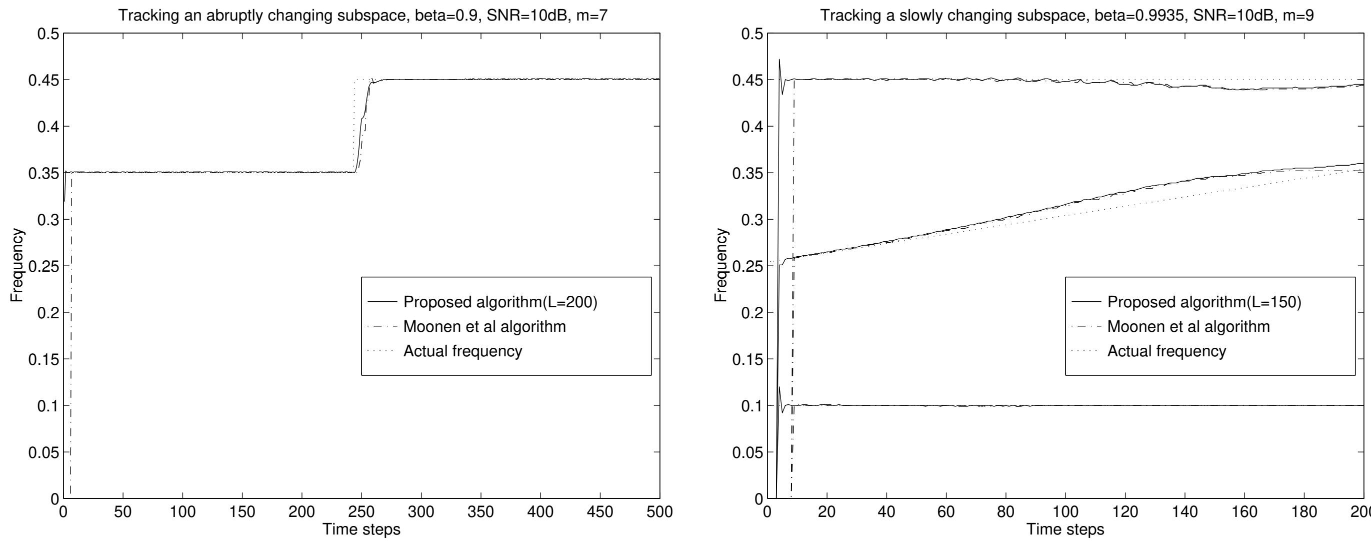
The linear array of P processors used for our scheme is shown in Figure 2. Let us recall, from section 2, that at the beginning of each time step, n, we have the matrices W_{n-1} and V_{n-1} from the SVD update of the previous step. The number of columns of each of these matrices is equal to the size of the data vectors, m. Figure 2 shows the initial distribution, (at the beginning of each time step) of the columns of both the matrices on the P processors. The data memory of each processor is conceptually divided into 2 parts - LEFT and RIGHT memory. Initially, processor i holds columns $(i-1) \times \frac{m}{p} + 1$ to $i \times \frac{m}{p}$ of W_{n-1} and V_{n-1} in its data memory. The columns are distributed equally between its LEFT and RIGHT memory as shown in Figure 2.

At each time step, the data vector y_n is broadcast to all the processors by a host processor. The broadcasting of y_n can be done elegantly by pipelining the broadcasting of y_{n+1} with the computation of time step n, if y_{n+1} is already available. This issue is not elaborated upon in this summary. Once, y_n is available to all the processors, each processor multiplies its columns of W_{n-1} with β and the columns of V_{n-1} by y_n' (refer equation 9). Thus we form the first matrix in the right hand side of equation 9, that is matrix X_n. We now need to orthogonalize the columns of X_n.

Next, the array performs one *sweep* of Hestenes SVD algorithm. In each *sweep*, first, each processor orthogonalizes all possible pairs of columns in its LEFT and RIGHT memory. These pairs are formed by taking both columns either from LEFT or RIGHT memory but not one from each. The rest of the *sweep* is accomplished in $n-1$ *cycles*. In each *cycle*, each processor orthogonalizes all possible pairs of columns formed by taking one column from its LEFT memory and one from its RIGHT memory. Then, the processors exchange the contents of their LEFT and RIGHT memories along the arrows shown in Figure 2. The connections between the processors are such that in each *sweep* (and also each update) all possible pairs of columns of X_n is orthogonalized exactly once [7]. The transformations applied to X_n are also applied to V_{n-1} to get V_n.

6 Advantages of using the proposed algorithm and its parallelization scheme

- The proposed algorithm and parallelization scheme can be implemented on a fixed sized array of off-the-shelf DSPs (Digital Signal Processors) thus avoiding the high design time as well as inflexibility due to custom processors. The array proposed by Moonen, Van Dooren and Vandewalle



(a) (b)

Figure 1: Frequency estimation

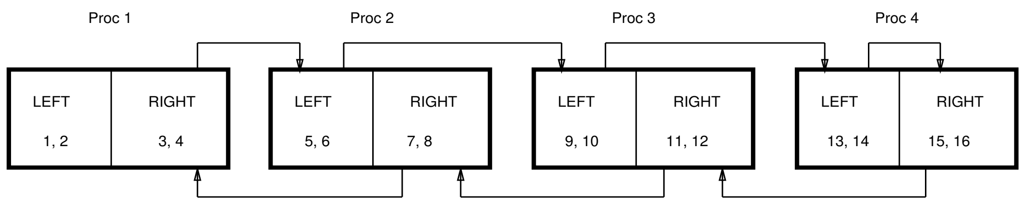


Figure 2: The linear array of processors used for the SVD updating, $P = 4, m = 16$

[5] is based on a custom VLSI processor array of size proportional to the size of the problem. Though the design of the array and its processing units have been reported in literature, actual execution times on this array is not available. So it is not possible to make a quantitative comparison of the two schemes.

- A software solution on off-the-shelf DSPs would be more flexible with regards to changing parameters like β, m and the size of the signal subspace according to the time-variance of the data vectors. Also, the subspace-based algorithm which uses the results of the SVD can be implemented on the same DSPs. This would reduce the physical size of the hardware and would be of great advantage in applications like subspace-based CDMA synchronization [8], where this hardware would be a part of a base station.

- This scheme uses *coarse grained parallelism*, that is, communication between processors is interleaved with computation involving $\frac{m}{P}$ columns of matrix W, instead of only a few elements. Hence, it is very suitable for computation on an array of typical DSPs like the Texas Instruments TMS320C40, which has single-cycle functional units, but, several cycles are required to transfer data across the communication ports.

- The parallelization scheme can be implemented on a scalable number of processors. Even if P is much less than the problem size (defined here by m), this scheme will yield speed-ups close to P and processor utilization close to 100%. Hence, with a processor array of given fixed size, we can update the SVD of arbitrarily large matrices.

References

[1] S. E. Bensley and B. Aazhang. Subspace-based channel estimation for code division multiple access communication systems. *Accepted for publication in IEEE Trans. Commun*, 1996.

[2] W. Ferzali and J. G. Proakis. Adaptive SVD algorithm with application to narrowband signal tracking. *SVD and Signal Processing*, II:149–159, 1991.

[3] M. R. Hestenes. Inversion of matrices by biorthogonalization and related results. *J. SIAM*, 6:51–90, 1958.

[4] M. Moonen, P. Van Dooren, and J. Vandewalle. An SVD updating algorithm for subspace tracking. *SIAM J. Matrix Anal. Appl.*, 13:1015–1038, 1992.

[5] M. Moonen, P. Van Dooren, and J. Vandewalle. A systolic array for SVD updating. *SIAM J. Matrix Anal. Appl.*, 14:353–371, 1993.

[6] R. O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Trans. on Antennas and Propagation*, AP-34(3):276–279, March 1986.

[7] R. Schreiber. Solving eigenvalue and singular value problems on an undersized systolic array. *SIAM Journal Sci. Stat. Comput.*, 7(2):441–451, April 1986.

[8] C. Sengupta, K. Kota, and J. R. Cavallaro. Parallel algorithms and architectures for subspace based channel estimation for CDMA communication systems. *SPIE*, 2846, 1996.