1

ULV AND GENERALIZED ULV SUBSPACE TRACKING ADAPTIVE ALGORITHMS

Srinath Hosur *Ahmed H. Tewfik* *Daniel Boley*

Dept. of Electrical Engineering and Computer Science,

University of Minnesota, Minneapolis, MN 55455

EDICS 2.6

November, 1995

Abstract

Traditional adaptive filters assume that the effective rank of the input signal is the same as the input covariance matrix or the filter length N. Therefore, if the input signal lives in a subspace of dimension less than N, these filters fail to perform satisfactorily. In this paper we present two new algorithms for adapting only in the dominant signal subspace. The first of these is a low-rank recursive-least-squares (RLS) algorithm which uses a ULV decomposition to track and adapt in the signal subspace. The second adaptive algorithm is a subspace tracking least-mean-squares (LMS) algorithm which uses a generalized ULV (GULV) decomposition, developed in this paper, to track and adapt in subspaces corresponding to several well conditioned singular value clusters. The algorithm also has an improved convergence speed compared to that of the LMS algorithm. Bounds on the quality of subspaces isolated using the GULV decomposition are derived and the performance of the adaptive algorithms are analyzed.

1 Introduction

Conventional adaptive algorithms assume that the desired signal lives in a space whose dimension is the same as the input covariance matrix or the length of the filter. However, in many signal processing applications, such as interference suppression using the adaptive-line-enhancer (ALE), the input signals exist in a subspace whose dimension is much lower than the filter length. In such cases, adaptive filtering only in those subspaces which contain dominant signal components results in a performance improvement due to the exclusion of the noise only modes. In this paper, we develop two new algorithms for adaptive filtering in the signal subspaces. The first of these algorithms is a low rank recursive-least-squares (RLS) algorithm which uses the ULV algorithm to track and adapt only in the signal subspaces. The second

This work was supported in part by ONR under grant N00014-92-J-1678, AFOSR under grant AF/F49620-93-1-0151DEF, DARPA under grant USDOC6NANB2D1272 and NSF under grant CCR-9405380.

algorithm is a subspace tracking least-mean-squares (LMS) algorithm which uses a generalization of the ULV decomposition developed in this paper.

Traditionally the singular value decomposition (SVD) is used to compute the low-rank least squares solution [1]. However, in real time applications, it is expensive to update the SVD. Recently, a low rank, eigensubspace RLS algorithm has been proposed in [2] using a Schur-type decomposition of the input correlation matrix. This algorithm requires $O(rN)$ flops, where r is the effective rank of the input correlation matrix. However, the algorithm requires the knowledge of this rank r. Hence, it is not suitable to applications where r varies with time. Rank-revealing QR (RRQR) decompositions may also be used to solve the least-squares (LS) problem [3]. These algorithms can track the rank and therefore do not suffer from the disadvantage of [2]. The computational complexity of this approach is $O(N^2)$. However, it has been shown [4] that the quality of approximation of the singular value subspaces using RRQR (and hence the closeness of the truncated RRQR decomposition to the truncated SVD) depends on the gap between the singular values. The LS problem was also solved by using a truncated ULV decomposition to approximate the data matrix [4]. Although the computational expense, $O(N^2)$, of this algorithm is greater than that of [2], this method offers the advantage that it is able to track rank changes. Furthermore, the quality of the approximations to the singular value subspaces that it produces does not depend on the magnitude of the gap in the singular values.

In many applications in signal processing and in particular for low rank subspace domain adaptive filtering, one is not interested in the exact singular vectors but in the subspaces corresponding to clusters of singular values of the same order of magnitude. Recently, some subspace updating techniques have been suggested [5]-[9]. A Kalman filter was used to update the eigenvector corresponding to the smallest eigenvalue in [5]. However it was not suggested how to modify the algorithm in case of multiple eigenvalues corresponding to noise. In [6], a fast eigen-decomposition algorithm which replaced the noise and signal eigenvalues by their corresponding average values was proposed. This technique could work well if the exact eigenvalues could be grouped together in two tight clusters. In [7] and [8], the averaging technique of [6] is used. However, the SVD is updated instead of the eigenvalue decomposition. This reduces the condition numbers to their square roots and increases numerical accuracy. Again the assumption that the singular values could be grouped into two tight clusters is made. In normal signal scenarios and in particular for the application targeted in this paper, this assumption is generally not valid.

The ULV decomposition was first introduced by Stewart [10], to break the eigenspace of the input correlation matrix $\mathbf{R}^N$, where N is the length of the impulse response of the adaptive filter, into two subspaces, one corresponding to the cluster of largest singular values and the other corresponding to the smaller singular values or noise subspace. This method is easily updated when new data arrives without making any a priori assumptions about the overall distribution of the singular values. Each ULV update requires only $O(N^2)$ operations. An analysis of the ULV algorithm was also performed [4, 11]. It was shown in [4] that the "noise" subspace (the subspace corresponding to the cluster of small singular values) is close to the corresponding SVD subspace. The analysis of [11] also shows that the ULV subspaces are only slightly more sensitive to perturbations. These analyses show that the ULV algorithm can be used in many situations where SVD was the only available alternative to date.

We use the ULV decomposition to develop a low rank recursive-least-squares (RLS) algorithm. The proposed algorithm tracks the subspace that contains the signal of interest using the ULV decomposition and adapts only in that subspace. Though the ULV decomposition requires $O(N^2)$ flops, the increase in computational complexity can be justified by the fact that the ULV decomposition is able to track changes in the numerical rank.

We also develop a new subspace tracking least-mean-squares (LMS) algorithm. The ULV decomposition tracks only two subspaces, the dominant signal subspace and the smaller singular value subspace. Even though, the dominant subspace contains strong signal components, its condition number might still be large. Now, recall that the convergence speed of the LMS algorithm depends inversely on the condition number of the input autocorrelation matrix (the ratio of its maximum to minimum eigenvalue) [12, 13]. Thus, a low rank LMS algorithm which uses the ULV decomposition would still have a poor convergence performance. We therefore develop a generalization of the ULV algorithm to track several well conditioned subspaces of the input correlation matrix. The input is then projected onto these subspaces and LMS adaptive filtering is performed in these well conditioned subspaces. This improves the convergence speed of the subspace tracking LMS algorithm.

This paper is organized as follows. In Section 2 we introduce the rank revealing ULV decomposition and the idea of tracking subspaces corresponding to clusters of singular values. Readers already familiar with the ULV decomposition can skip Section 2.1 and the first part of Section 2.2. The concluding part of this section contains a discussion on various heuristics used to decide if a gap exists between the singular values. This discussion motivates the use of a new heuristic introduced in this paper with the ULV and especially the GULV algorithms. Some of the bounds on the quality of the subspaces are reviewed in Section 2.3. We also derive a new bound on the angle between the subspaces generated using the ULV algorithm on a perturbed data matrix and the corresponding subspaces obtained using a SVD on the true data matrix. Next, we develop the ULV-RLS algorithm. The GULV algorithm is presented in Section 5. In this section, we also show that the bounds on the subspace quality derived for the plain ULV decomposition can be recursively applied to estimate the quality of the subspaces obtained using the GULV procedure. Section 6 introduces the idea of subspace domain LMS adaptive filtering and Section 7 analyzes its performance. Numerical examples are discussed in Section 8.

2 The ULV decomposition

Many signal processing problems require that we isolate the smallest singular values of a matrix. The matrix is typically a covariance matrix or a data matrix that is used to estimate a covariance matrix. The decision as to how many singular values to isolate is usually based on a threshold value (find all the singular values below the threshold) or on a count (find the last r singular values). While extracting the singular values one often wants to keep clusters of the singular values together as a unit. In the SVD, this extraction is easy since all the singular values are "displayed". One can therefore easily traverse the entire sequence of singular values to isolate the desired set. Therefore, in order to isolate the smallest singular values, one needs to choose a proper threshold and identify all the singular values which lie below this threshold. As mentioned earlier, the drawback of the SVD is that it requires $O(N^3)$ flops. Here, we review the ULV decomposition of Stewart [10]. The ULV decomposition can be used to divide the singular values into two groups and compute a basis for the space spanned by the corresponding groups of singular vectors.

2.1 Data Structure

The ULV decomposition of a real $k \times N$ matrix $\mathbf{A}$ (where $k \geq N$) is a triple of 3 matrices $\mathbf{U}$, $\mathbf{L}$, $\mathbf{V}$ plus a rank index r, where

$$\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T, \tag{2.1}$$

$\mathbf{V}$ is a $N \times N$ orthogonal matrix, $\mathbf{L}$ is a $N \times N$ lower triangular matrix and $\mathbf{U}$ has the same shape as $\mathbf{A}$ with orthonormal columns. The lower triangular matrix $\mathbf{L}$ can be partitioned as

$$\mathbf{L} \doteq \begin{pmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}, \tag{2.2}$$

where, $\mathbf{C}$, the leading $r \times r$ part of $\mathbf{L}$ has a Frobenius norm approximately equal to the norm of a vector of the r leading singular values of $\mathbf{A}$. That is, if the singular values of $\mathbf{A}$ satisfy

$$\sigma_1(\mathbf{A}) \geq \cdots \sigma_r(\mathbf{A}) > \sigma_{r+1}(\mathbf{A}) \geq \cdots \sigma_N(\mathbf{A}) \tag{2.3}$$

then $\|\mathbf{C}\|_F^2 \approx \sigma_1^2(\mathbf{A}) + \cdots + \sigma_r^2(\mathbf{A})$. This implies that $\mathbf{C}$ encapsulates the "large" singular values of $\mathbf{L}$ and $(\mathbf{E}, \mathbf{F})$ (the trailing $N-r$ rows of $\mathbf{L}$) approximately encapsulate the $N-r$ smallest singular values. The last $N-r$ columns of $\mathbf{V}$ encapsulate the corresponding trailing right singular vectors.

In the data structure actually used for computation, $\mathbf{L}$ is needed to determine the rank index at each stage as new rows are appended. However, $\mathbf{U}$ is not needed to obtain the right singular vectors. Therefore, a given ULV decomposition can be represented just by the triple $[\mathbf{L}, \mathbf{V}, r]$. The ULV decomposition is *rank revealing*[1] in the sense that the norm of the matrix $[\mathbf{E}\ \mathbf{F}]$ is smaller than some specified tolerance.

Thus, this decomposition immediately provides us with the sub-spaces corresponding to a group of largest singular values and another corresponding to the group of smallest singular values.

The ULV updating procedure updates the ULV decomposition of the data matrix corresponding to the input process, as additional data vectors become available. In essence, it updates the subspaces corresponding to the group of large eigenvalues and that of small eigenvalues of the correlation matrix of the input to the adaptive filter.

[1]term coined by T. F. Chan

$$\begin{array}{ccccc} \mathtt{C}&.&.&.&. \\ \mathtt{C}&\mathtt{C}&.&.&. \\ \mathtt{C}&\mathtt{C}&\mathtt{C}&.&. \\ \mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{f}&. \\ \mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{f}&\mathtt{f} \\ \mathtt{R}&\mathtt{R}&\mathtt{R}&\mathtt{R}&\mathtt{R} \end{array} \quad \begin{array}{l}\text{rotate}\\ \text{from} \Longrightarrow\\ \text{right}\end{array} \quad \begin{array}{ccccc} \mathtt{C}&+&.&.&. \\ \mathtt{C}&\mathtt{C}&+&.&. \\ \mathtt{C}&\mathtt{C}&\mathtt{C}&+&. \\ \mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{f}&+ \\ \mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{f}&\mathtt{f} \\ \mathtt{R}&.&.&.&. \end{array} \quad \begin{array}{l}\text{rotate}\\ \text{from} \Longrightarrow\\ \text{left}\end{array} \quad \begin{array}{ccccc} \mathtt{C}&.&.&.&. \\ \mathtt{C}&\mathtt{C}&.&.&. \\ \mathtt{C}&\mathtt{C}&\mathtt{C}&.&. \\ \mathtt{E}&\mathtt{E}&\mathtt{E}&\mathtt{F}&. \\ \mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{f}&\mathtt{f} \\ .&.&.&.&. \end{array} \quad \begin{array}{l}\text{chop}\\ \text{away}\\ \text{zeros} \Longrightarrow\\ \text{increment}\\ \text{rank index}\end{array} \quad \begin{array}{ccccc} \mathtt{C}&.&.&.&. \\ \mathtt{C}&\mathtt{C}&.&.&. \\ \mathtt{C}&\mathtt{C}&\mathtt{C}&.&. \\ \mathtt{C}&\mathtt{C}&\mathtt{C}&\mathtt{C}&. \\ \mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{e}&\mathtt{f} \end{array}$$

Figure 1: Sketch of `Absorb_One` procedure. Upper case letters denote large entries, lower case letters small entries in the ULV partitioning, `R` denotes an entry of the new row, `+` a temporary fill, and `.` a zero entry.

2.2 Primitive Procedures

The ULV updating process consists of five primitive procedures. The first three procedures are designed to allow easy updating of the ULV decomposition as new rows are appended. Each basic procedure costs $O(N^2)$ operations and consists of a sequence of plane (Givens) rotations [14]. Pre-multiplication by a plane rotation operates on the rows of the matrix while post-multiplication operates on its columns. By using a sequence of such rotations in a very special order, we can annihilate desired entries while filling in as few zero entries as possible. We then restore the few zeroes that are filled in. We show the operations on $\mathbf{L}$, partitioned as in (2.2). Each rotation applied from the right is also accumulated in $\mathbf{V}$, to maintain the identity $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T$, where $\mathbf{U}$ is not saved. The last two procedures use the first three to complete a ULV update.

- `Absorb_One`: Absorb a new row. The matrix $\mathbf{A}$ is augmented by one row, yielding

$$\begin{pmatrix} \mathbf{A} \\ \mathbf{a}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{L} \\ \mathbf{a}^T\mathbf{V} \end{pmatrix} \mathbf{V}^T.$$

  The matrices $\mathbf{L}$, $\mathbf{V}$ are then updated to restore the ULV structure, and the rank index r is incremented by 1. The process is sketched in Fig. (1).

- `Extract_Info`: The following information is extracted from the ULV decomposition: (a) an approximation of the last singular value of $\mathbf{C}$ (i.e., the leading $r \times r$ part of $\mathbf{L}$), and (b) a left singular vector of $\mathbf{C}$ corresponding to this singular value. These are computed using a *condition number estimator* [15].

- `Deflate_One`: Deflate the ULV decomposition by one (i.e., apply transformation and decrement the rank index by one so that the smallest singular value in the leading $r \times r$ part of $\mathbf{L}$ is "moved" to the trailing rows). Specifically, transformations are applied to isolate the smallest singular value in the leading $r \times r$ part of $\mathbf{L}$ into the last row of this leading part. The transformations are constructed using item (c) from `Extract_Info` and applied in a manner similar to `Absorb_One`. Then the rank index is decremented by 1, effectively moving the smallest singular value from the leading part to the trailing part of $\mathbf{L}$. This operation just moves the singular value without checking whether the singular value moved is close to zero or any other singular value.

- `Deflate_To_Gap`: This procedure tries to move the rank boundary, represented by the rank index r, toward a gap among the singular values. Let s be the smallest singular value of $\mathbf{C}$ obtained using `Extract_Info`. The `Deflate_To_Gap` procedure essentially decides if the magnitude of this singular value is of the same order as that of the singular values in the trailing part of $\mathbf{L}$. This decision is made using a heuristic. After applying the heuristic, if the procedure decides that s is of the same order of magnitude as the trailing singular values, a deflation is performed using `Deflate_One`. The procedure then calls `Extract_Info` with the new rank index. This process is repeated till a gap in the singular values is found. Various heuristics can be used to try and determine if a gap between the singular values exists. However some of them are not suitable for use with generalization of the ULV. We examine some of these techniques in Section 2.2.1 and discuss their relative merits and demerits.
- `Update`: This procedure encompasses the entire process. It takes an old ULV decomposition and a new row to append, and incorporates the row into the ULV decomposition. The new row is absorbed, and the rank is deflated if necessary to find a gap among the singular values.

2.2.1 Choice of heuristics for deflation

Various heuristics can be used to decide if a gap exists in the singular values. The choice of the heuristic is extremely important as it is used to cluster the singular values and hence obtain the correct singular subspaces. To our knowledge three heuristics (including the one used in this paper) have been proposed in literature.

The heuristic proposed in [16] estimates the smallest singular value, s of $\mathbf{C}$ and compares it with a user specified tolerance. This tolerance, provided by the user, is usually based on some knowledge of the eigenvalue distribution. The choice of the tolerance is important. If it is too large, the rank may be underestimated and if it is too small, it may be overestimated. Also, as we shall see later, the user has to provide several tolerances to track more than two clusters using the generalized ULV decomposition. In practice all these tolerances may not be available. Therefore, this heuristic cannot be used in the GULV algorithm that we describe next.

A second heuristic has been proposed in [9]. This heuristic decides that a gap in the singular values exists if $s^2 > d^2(f^2 + b^2)$ where f is the Frobenius norm of the trailing part $[\mathbf{E}, \mathbf{F}]$. The parameter b, called `Zero_Tolerance`, is selected by the user. It is included to allow for round-off or other errors. This heuristic has the nice feature that only the `Spread` d and the `Zero_Tolerance` b need to be specified. The algorithm then uses this heuristic to cluster all singular values of similar order of magnitude. However, when the rank increase is greater than one or the algorithm underestimates the numerical rank, this heuristic leads to some problems. In particular if one of the larger singular values lies within the trailing part, the heuristic might decide that no gap exists. Deflation is then repeatedly applied till the rank index becomes zero. As the algorithm is limited to growing the rank boundary by no more than one for each iteration, the algorithm has to be reinitialized by artificially moving the rank boundary all the way down to the smallest eigenvalue of $\mathbf{L}$

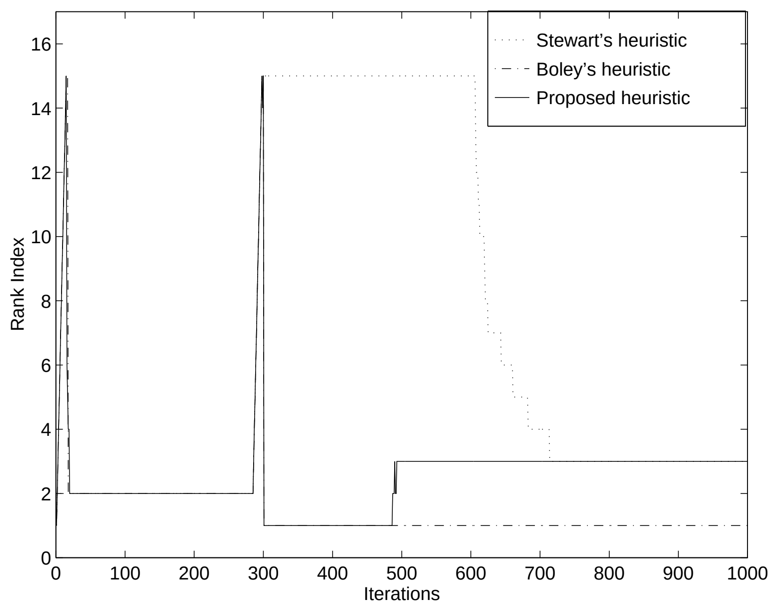


Figure 2: Tracking performance of ULV using different heuristics for finding a gap in singular values

and searching for the rank boundary (using deflations). Though it is a reasonable assumption that the rank usually does not change by more than one, the problem still exists if the rank is underestimated. Figure 2 shows the rank tracking behavior of the ULV algorithm using the heuristics of [10], [9] and that proposed in this paper. The input initially consisted of two complex sinusoids each having an amplitude of 0.1. The background noise was white with a variance of 10^{-12}. Therefore, there are initially two large singular values with magnitudes on the order of 0.01 each. The ULV algorithms using all the heuristics converge to a rank estimate of two. Next a complex exponential of unit amplitude is added to the input. Now the larger group of singular values is $\{1, 0.01, 0.01\}$. The figure shows that after some time, the ULV algorithm using the proposed heuristic and that of [10] converged to a rank of three. However, the ULV algorithm using the heuristic of [9] converges to one. This is because, the heuristic initially under estimated the rank as two. Thus, a singular value of magnitude 0.01 is isolated into the trailing part of $\mathbf{L}$, making the Frobenius norm of this part of the same order as the smallest singular value of the leading part and forcing a deflation.

We will see later that even though the outermost rank boundary in a generalized ULV decomposition cannot change by more than one, the inner rank boundaries can change by more than one. As the rank increase of each boundary is limited to one per iteration, a large singular value would be isolated into the next group of small singular values. Thus, the situation described above might occur frequently and the inner rank boundary might be erroneously estimated. Therefore, this heuristic also cannot be used with the GULV decomposition.

The heuristic proposed in this paper tries to combine the advantages of the two heuristics that we discussed above. By using the heuristic proposed in this paper, we can automatically isolate clusters of singular values of similar order of magnitude i.e., the condition number of each cluster lies within the user defined `Spread`. Also, it does not suffer from the disadvantage of the second heuristic. If the estimate of the rank boundary is too low, the heuristic allows the rank to grow until it attains the correct value. This heuristic estimates the smallest singular value, f, of $\mathbf{L}$ in addition to that of $\mathbf{C}$ (s). The heuristic then decides that a gap between the singular values exists if $s > df$, where d is the `Spread` chosen by the user. Thus, this heuristic does not require a user specified tolerance. In case of the GULV decomposition, f is simply the smallest value of the small singular value group adjacent to the group on which `Deflate_To_Gap` is being applied. We shall see later that this heuristic can be used with the GULV decomposition with minimum additional computations. The tracking performance of the ULV decomposition using this heuristic is shown in Fig. 2. Note that if we replace df by the user specified tolerance in our heuristic, we obtain the heuristic of [10].

2.3 Quality of Subspaces

Consider the orthogonal projector onto a subspace $\mathcal{S}$, $\mathbf{P}_{\mathcal{S}}$. For two equi-dimensional subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, the distance between subspaces is characterized by

$$\sin\theta(\mathcal{S}_1, \mathcal{S}_2) \doteq \|(\mathbf{I} - \mathbf{P}_{\mathcal{S}_1})\mathbf{P}_{\mathcal{S}_2}\| = \|(\mathbf{I} - \mathbf{P}_{\mathcal{S}_2})\mathbf{P}_{\mathcal{S}_1}\|. \tag{2.4}$$

Bounds have been derived for the ULV algorithm to assess the distance between the ULV subspaces and the corresponding singular subspaces and to measure sensitivity of the subspaces to perturbations.

Let the ULV decomposition of $\mathbf{A}$ be represented as

$$\mathbf{A} = \begin{bmatrix}\mathbf{U}_r\mathbf{U}_0\mathbf{U}_-\end{bmatrix} \begin{bmatrix} \mathbf{L}_r & \mathbf{0} \\ \mathbf{H} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix}\mathbf{V}_r\mathbf{V}_0\end{bmatrix}^T \tag{2.5}$$

and the SVD of $\mathbf{A}$ be given by

$$\mathbf{A} = \begin{bmatrix}\mathbf{U}_1\mathbf{U}_2\mathbf{U}^-\end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{n-r} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix}\mathbf{V}_1\mathbf{V}_2\end{bmatrix}^T. \tag{2.6}$$

The following theorem due to Fierro and Bunch [4] shows that as the off-diagonal block $\mathbf{H}$ decreases, the ULV subspaces converge to their SVD counterparts.

Theorem 1 (Fierro & Bunch) *Let $\mathbf{A}$ have the ULV in (2.5) and the SVD in (2.6). Assume $\|\cdot\| = \|\cdot\|_2$. If $\|\mathbf{E}\| < \sigma_{\min}(\mathbf{L}_r)$ then*

$$\begin{aligned} \sin\theta(\mathcal{R}(\mathbf{V}_r), \mathcal{R}(\mathbf{V}_1)) &\leq \frac{\|\mathbf{H}\|\|\mathbf{E}\|}{\sigma^2_{\min}(\mathbf{L}_r) - \|\mathbf{E}\|^2} \\ \sin\phi(\mathcal{R}(\mathbf{U}_r), \mathcal{R}(\mathbf{U}_1)) &\leq \frac{\|\mathbf{H}\|}{\sigma_{\min}(\mathbf{L}_r) - \|\mathbf{E}\|} \\ \frac{\|\mathbf{H}\|}{\|\mathbf{L}_r\| + \|\mathbf{E}\|} &\leq \sin\phi(\mathcal{R}(\mathbf{U}_r), \mathcal{R}(\mathbf{U}_1)). \end{aligned}$$

These bounds also reveal that there is a limit on how close some subspaces can be.

In most applications, we will have access to the perturbed matrix $\tilde{\mathbf{A}} = \mathbf{A} + \delta\mathbf{A}$ rather than $\mathbf{A}$ itself. Let $\tilde{\mathbf{A}}$ have the ULV decomposition

$$\tilde{\mathbf{A}} = \left[\tilde{\mathbf{U}}_r \tilde{\mathbf{U}}_0 \tilde{\mathbf{U}}_-\right] \begin{bmatrix} \tilde{\mathbf{L}}_r & \mathbf{0} \\ \tilde{\mathbf{H}} & \tilde{\mathbf{E}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \left[\tilde{\mathbf{V}}_r \tilde{\mathbf{V}}_0\right]^T. \tag{2.7}$$

Further, let $\tilde{\mathbf{X}}_r$ and $\tilde{\mathbf{Y}}_r$ form an orthogonal basis for $\mathcal{R}(\tilde{\mathbf{V}}_r)$ and $\mathcal{R}(\tilde{\mathbf{U}}_r)$ respectively. Define

$$\epsilon \doteq \max\left(\|\delta\mathbf{A}\tilde{\mathbf{X}}_r\|, \|\tilde{\mathbf{Y}}_r^T \delta\mathbf{A}\|\right). \tag{2.8}$$

Then, the following theorem bounds the sensitivity of the ULV subspaces.

Theorem 2 (Fierro) *Let* $\mathbf{A}$ *and* $\tilde{\mathbf{A}}$ *have the ULV decompositions (2.5) and (2.7) respectively. If* $\sigma_{\max}(\mathbf{E}) < \sigma_{\min}(\tilde{\mathbf{L}}_r)$ *then for* ϵ *as defined in (2.8) we have*

$$\begin{aligned} \sin\theta(\mathcal{R}(\tilde{\mathbf{V}}_r), \mathcal{R}(\mathbf{V}_r)) &\leq \frac{(\|\mathbf{H}\| + \|\tilde{\mathbf{H}}\|)\sigma_{\max}(\mathbf{E})}{\sigma_{\min}^2(\tilde{\mathbf{L}}_r) - \|\mathbf{E}\|_2^2} + \frac{\epsilon}{\sigma_{\min}(\tilde{\mathbf{L}}_r) - \|\mathbf{E}\|_2} \\ \sin\phi(\mathcal{R}(\tilde{\mathbf{U}}_r), \mathcal{R}(\mathbf{U}_r)) &\leq \frac{\|\mathbf{H}\| + \|\tilde{\mathbf{H}}\|}{\sigma_{\min}(\tilde{\mathbf{L}}_r) - \|\mathbf{E}\|_2} + \frac{\epsilon}{\sigma_{\min}(\tilde{\mathbf{L}}_r) - \|\mathbf{E}\|_2}. \end{aligned}$$

These results indicate that the ULV subspaces are only slightly more sensitive to perturbations than the singular subspaces [11].

The above theorems provide us with bounds on the distance between ULV subspaces and the SVD subspaces. They also provide us with bounds on the angle between the subspaces obtained by performing a ULV decomposition on the actual data matrix and a perturbed data matrix. However, they do not provide any bounds on the distance between the subspaces obtained using the ULV decomposition on a perturbed matrix and those obtained using the SVD on the actual matrix. In many signal processing applications, the ULV decomposition is preferred to the SVD due to its lower computational complexity. One such application, adaptive filtering in subspaces, is discussed in this paper. Here, the input data is projected onto several well conditioned subspaces containing the input signal energy. The projected data in these signal subspaces is then adaptively combined to generate an estimate of the desired response. In applications such as these, the data matrix is usually corrupted by noise. We therefore need to provide bounds on the angle between the subspaces obtained using the *ULV decomposition on a perturbed matrix* and those obtained using the *SVD* on the *actual matrix* as a measure of the quality of the subspaces isolated using the ULV decomposition. Such bounds may be obtained directly from Theorem 2 by noting that the SVD of $\mathbf{A}$ may be viewed as a ULV decomposition with $\mathbf{H} = 0$, $\mathbf{L}_r = \mathbf{\Sigma}_r$, $\mathbf{E} = \mathbf{\Sigma}_{n-r}$ and $\mathbf{V}_r = \mathbf{V}_1$. Hence, we have the following new theorem.

Theorem 3 *Let* $\mathbf{A}$ *and* $\tilde{\mathbf{A}}$ *have the SVD and ULV decompositions (2.6) and (2.7) respectively. If* $\sigma_{\max}(\mathbf{E}) < \sigma_{\min}(\tilde{\mathbf{L}}_r)$ *then for* ϵ *as defined in (2.8) we have*

$$\sin\theta(\mathcal{R}(\tilde{\mathbf{V}}_r), \mathcal{R}(\mathbf{V}_1)) \leq \frac{\|\tilde{\mathbf{H}}\|\sigma_{r+1}}{\sigma_{\min}^2(\tilde{\mathbf{L}}_r) - \sigma_{r+1}^2} + \frac{\epsilon}{\sigma_{\min}(\tilde{\mathbf{L}}_r) - \sigma_{r+1}}.$$

The above theorem indicates that as the norm of the off-diagonal block $\tilde{\mathbf{H}}$ decreases, the error between the ULV subspace and the corresponding true SVD subspace is dominated by the magnitude of the perturbation in the data matrix. Note that by setting the matrix $\tilde{\mathbf{H}} = \mathbf{0}$ in the ULV decomposition, we decouple the first singular value group from the second singular value cluster. In particular, we effectively have obtained the singular subspaces for the matrix. If, furthermore, there exist α and δ such that

$$\sigma_{\min}(\tilde{\mathbf{L}}) \geq \alpha + \delta \quad \text{and} \quad \sigma_{r+1} \leq \alpha, \tag{2.9}$$

the above theorem reduces to the perturbation bounds for singular subspaces obtained by Wedin [17]. We therefore, obtain the perturbation bound for singular subspaces as a special case of the ULV bound. Note also that this discussion implies that by using refinement strategies (at extra computational cost), we can increase the accuracy of the ULV estimates of the SVD subspaces by reducing the norm of $\tilde{\mathbf{H}}$. Therefore, the perturbation $\delta\mathbf{A}$ yields the ultimate limit on the accuracy of the subspaces obtained using the ULV decomposition.

3 Low Rank RLS Algorithm

In this section we use the ULV decomposition to develop a subspace tracking RLS algorithm. Let the input signal vector at time n be given by

$$\mathbf{x}(n) = [x_0(n), x_1(n), \ldots, x_{N-1}(n)]^T. \tag{3.1}$$

Note that in case of filtering $x_k(n) = x(n-k)$. Also, let the adaptive filter weight vector at this time be $\mathbf{h}(n)$. The corresponding filter output is the obtained as

$$\hat{d}(n) = \mathbf{x}^T(n)\mathbf{h}(n). \tag{3.2}$$

The error between the desired response $d(n)$ and that estimated by the adaptive filter $\hat{d}(n)$ can be written as

$$e(n) = d(n) - \hat{d}(n) \; = \; d(n) - \mathbf{x}^T(n)\mathbf{h}(n). \tag{3.3}$$

The RLS algorithm tries to recursively solve the weighted LS problem

$$\min_{\mathbf{h}(n)} \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2. \tag{3.4}$$

By rewriting (3.4), we find that the RLS algorithm solves the following problem [12].

$$\min_{\mathbf{h}} \|\mathbf{\Lambda}^{1/2}(n)(\mathbf{X}(n)\mathbf{h}(n) - \mathbf{d}(n))\|^2. \tag{3.5}$$

In (3.5), $\mathbf{d}(n) = [d(1), d(2), \cdots, d(n)]^T$ is the desired response vector, $\mathbf{X}(n)$ is the input data matrix given by

$$\mathbf{X}(n) = \Big[\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)\Big]^T, \tag{3.6}$$

and $\mathbf{\Lambda}(n)$ is the $n \times n$ diagonal exponential weighting matrix given by

$$\mathbf{\Lambda}(n) = \mathrm{diag}\Big(\lambda^{n-1}, \lambda^{n-2}, \ldots, 1\Big). \tag{3.7}$$

When $\mathbf{\Lambda} = \mathbf{I}$, the LS solution can be expressed in terms of the SVD as

$$\mathbf{h}_{\mathrm{LS}}(n) = \mathbf{X}^{\dagger}(n)\mathbf{d}(n) = \mathbf{V}(n)\mathbf{\Sigma}^{\dagger}(n)\mathbf{U}^{T}(n)\mathbf{d}(n). \tag{3.8}$$

Here, $(\cdot)^{\dagger}$ denotes the pseudoinverse [1].

When $\mathbf{X}(n)$ is close to rank deficient, the least squares solution is ill conditioned due to the inversion of the small singular values in $\mathbf{\Sigma}^{\dagger}(n)$. In such cases, a rank r approximant $\mathbf{X}_r(n)$ of the matrix $\mathbf{X}(n)$ is constructed by setting the $N - r$ small singular values of $\mathbf{X}(n)$ to zero in its singular value decomposition. Thus, if the singular value decomposition of $\mathbf{X}(n)$ is given by (2.6), its low rank approximate is given by

$$\mathbf{X}_r(n) = \Big[\mathbf{U}_1(n)\mathbf{U}_2(n)\mathbf{U}^{-}(n)\Big] \begin{bmatrix} \mathbf{\Sigma}_r(n) & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{n-k} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \Big[\mathbf{V}_1(n)\mathbf{V}_2(n)\Big]^{T} = \mathbf{U}_1(n)\mathbf{\Sigma}_r(n)\mathbf{V}_1^{T}(n). \tag{3.9}$$

The solution to the modified LS problem is then obtained as

$$\mathbf{h}_{\mathrm{MLS}}(n) = \mathbf{X}_r^{\dagger}(n)\mathbf{d}(n) = \mathbf{V}_1(n)\mathbf{\Sigma}_r^{-1}(n)\mathbf{U}_1^{T}(n)\mathbf{d}(n). \tag{3.10}$$

It has been recently suggested, [18], that the rank r approximate of $\mathbf{X}(n)$, discussed above, be replaced by a matrix obtained using a truncated ULV decomposition. The main motivations behind the use of the ULV decomposition is its lower computational expense, $O(N^2)$ as compared to the SVD $O(N^3)$. Thus, the modified minimum norm solution can be computed as [18]

$$\mathbf{h}_{\mathrm{ULV-LS}}(n) = \mathbf{V}_r(n)\mathbf{L}_r^{-1}(n)\mathbf{U}_r^{T}(n)\mathbf{d}(n). \tag{3.11}$$

It was however not suggested how the algorithm is to be modified in case the data matrix $\mathbf{X}(n)$ grows as new data becomes available. Clearly, in such a case storing $\mathbf{U}(n)$ is not a viable option. In this section, we suggest a method to obtain a recursive solution to the modified LS problem (3.10).

The LS minimization problem discussed above is invariant under any unitary transformation. For some $n \geq N$, let the ULV decomposition of the weighted data matrix be given as

$$\mathbf{\Lambda}^{1/2}(n)\mathbf{X}(n) = \mathbf{U}(n) \begin{bmatrix} \mathbf{L}_r(n) & \mathbf{0} \\ \mathbf{H}(n) & \mathbf{E}(n) \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{V}^{T}(n), \tag{3.12}$$

where the columns of $\mathbf{U}(n)$ and $\mathbf{V}(n)$ can again be clustered as in (2.5). Then,

$$\mathbf{U}^{T}(n)\mathbf{\Lambda}^{1/2}(n)(\mathbf{d}(n) - \mathbf{X}(n)\mathbf{h}(n)) = \begin{bmatrix} \mathbf{p}(n) - \begin{bmatrix} \mathbf{L}_r(n) & \mathbf{0} \\ \mathbf{H}(n) & \mathbf{E}(n) \end{bmatrix} \mathbf{V}^{T}(n)\mathbf{h}(n) \\ \mathbf{v}(n) \end{bmatrix}, \tag{3.13}$$

where

$$\mathbf{U}^T(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{d}(n) = \begin{bmatrix} \mathbf{p}(n) \\ \mathbf{v}(n) \end{bmatrix}. \tag{3.14}$$

Thus the RLS problem is equivalent to solving for $\mathbf{h}(n)$ the system of linear equations

$$\begin{bmatrix} \mathbf{L}_r(n) & \mathbf{0} \\ \mathbf{H}(n) & \mathbf{E}(n) \end{bmatrix} \mathbf{V}^T(n)\mathbf{h}(n) = \mathbf{p}(n). \tag{3.15}$$

Note that $\mathbf{V}(n) = [\mathbf{V}_r(n)\mathbf{V}_0(n)]$. Therefore, the above equation can be rewritten as

$$\begin{bmatrix} \mathbf{L}_r(n) & \mathbf{0} \\ \mathbf{H}(n) & \mathbf{E}(n) \end{bmatrix} \begin{bmatrix} \mathbf{V}_r^T(n)\mathbf{h}(n) \\ \mathbf{V}_0^T(n)\mathbf{h}(n) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_r(n) \\ \mathbf{p}_{N-r}(n) \end{bmatrix}, \tag{3.16}$$

where $\mathbf{p}_r(n)$ are the first r elements of $\mathbf{p}(n)$. Thus, the low rank RLS solution can be obtained in two steps as

$$\begin{aligned} \mathbf{g}(n) &= \mathbf{L}_r^{-1}(n)\mathbf{p}_r(n) & (3.17) \\ \mathbf{h}_{\mathrm{ULV-LS}}(n) &= \mathbf{V}_r(n)\mathbf{g}(n). & (3.18) \end{aligned}$$

The computation of the ULV-LS solution using Eqs. (3.17)-(3.18) requires only $O(Nr)$ flops.

Note that the ULV algorithm updates $\mathbf{U}(n)$, $\mathbf{L}(n)$ and $\mathbf{V}(n)$ from previously computed values. To do this, it appends the new input vector $\mathbf{x}(n)$ as a row to $\lambda^{1/2}\mathbf{L}(n-1)$ and applies Givens rotations from the right and the left in a special order to compute $\mathbf{L}(n)$. This operation can be written as

$$\begin{bmatrix} \mathbf{L}(n) \\ \mathbf{0} \end{bmatrix} = \mathbf{T}^T(n) \begin{bmatrix} \lambda^{1/2}\mathbf{L}(n-1) \\ \mathbf{x}^T(n) \end{bmatrix} \mathbf{G}(n). \tag{3.19}$$

The ULV algorithm discards all the rotations applied from the left. The right rotations are absorbed into $\mathbf{V}(n)$. However, the vector $\mathbf{p}(n)$ can be easily updated from $\mathbf{p}(n-1)$ as

$$\begin{bmatrix} \mathbf{p}(n) \\ v(n) \end{bmatrix} = \mathbf{T}^T(n) \begin{bmatrix} \lambda^{1/2}\mathbf{p}(n-1) \\ d(n) \end{bmatrix}. \tag{3.20}$$

Since each Givens rotation affects only two elements of the vector, we must perform $O(N)$ flops to update $\mathbf{p}(n)$. Thus, the low rank RLS requires $O(N^2)$ flops per iteration.

4 ULV-RLS Performance Analysis

In this section we perform a simple convergence analysis of the ULV-RLS algorithm. The ULV subspaces are in general perturbed from their SVD counterparts. Therefore, the main purpose of this section is to analyze these effects on the convergence behavior of the ULV-RLS algorithm. For the rest of this section, we assume that the algorithm operates in a stationary environment. Therefore, we set the exponential weighting factor $\lambda = 1$ to get the optimal steady-state result. This assumption implies that the environment is stationary and allows us to draw some general conclusions about the ULV-RLS algorithm. The expressions can be easily modified for a non-unit λ.

In order to analyze the behavior of the ULV-RLS algorithm, we consider the following multiple linear regression model. Assume that the input is in a low rank subspace of dimension r where $r < N$. The $N \times 1$ input vector $\mathbf{x}(n)$, at time n, is projected into the r dimensional subspace using a $r \times N$ transformation matrix, $\mathbf{S}_r$. The columns of $\mathbf{S}_r^T$ are the r eigenvectors of the covariance matrix $\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^T(n)]$ corresponding to its r non-zero eigenvalues. The resulting $r \times 1$ vector $\mathbf{y}(n) = \mathbf{S}_r\mathbf{x}(n)$ is weighted by $\mathbf{g}_o$, a $r \times 1$ *regression parameter vector* in the transform domain. The desired output $d(n)$ is then generated, according to this model, as

$$d(n) = \mathbf{y}^T(n)\mathbf{g}_o + e_o(n) \ = \ \mathbf{x}^T(n)\mathbf{h}_o + e_o(n), \tag{4.21}$$

where $e_o(n)$ is called the *measurement error* and $\mathbf{h}_o$ is the corresponding time domain $N \times 1$ regression parameter vector,

$$\mathbf{h}_o = \mathbf{S}_r^T\mathbf{g}_o. \tag{4.22}$$

The process $\{e_o(n)\}$ is assumed to be white with zero mean and variance σ^2. Since algorithm operates in a stationary environment, the vector $\mathbf{h}_o$ is constant.

4.1 Weight vector behavior

The rank-r ULV-RLS weight vector at time n, $\mathbf{h}_{\mathrm{ULV-LS}}(n)$, satisfies (see (3.11)),

$$\mathbf{U}_r(n)\mathbf{L}_r(n)\mathbf{V}_r^T(n)\mathbf{h}_{\mathrm{ULV-LS}}(n) = \mathbf{d}(n). \tag{4.23}$$

From (4.21), we see that the regression vector satisfies

$$\mathbf{X}(n)\mathbf{h}_o = \mathbf{d}(n) - \mathbf{e}_o(n), \tag{4.24}$$

where $\mathbf{e}_o(n) = [e_o(1), e_o(2), \ldots, e_o(n)]^T$ is the vector of the measurement errors up to time n. As the rank of $\mathbf{X}(n)$ is r, we can rewrite (4.24), using the SVD of $\mathbf{X}(n)$ defined in (2.6), as

$$\mathbf{U}_1(n)\mathbf{\Sigma}_r(n)\mathbf{V}_1^T(n)\mathbf{h}_o = \mathbf{d}(n) - \mathbf{e}_o(n). \tag{4.25}$$

Subtracting (4.25) from (4.23), multiplying both sides of the result by $\mathbf{X}^T(n)$ and performing some simple mathematical manipulations, we obtain

$$\mathbf{V}_r(n)\mathbf{L}_r^T(n)\mathbf{L}_r(n)\mathbf{V}_r^T(n)\mathbf{h}_{\mathrm{ULV-LS}}(n) - \mathbf{V}_1(n)\mathbf{\Sigma}_r^2(n)\mathbf{V}_1^T(n)\mathbf{h}_o = \mathbf{X}^T(n)\mathbf{e}_o(n). \tag{4.26}$$

Notice that $\hat{\mathbf{\Phi}}(n) = \mathbf{V}_r(n)\mathbf{L}_r^T(n)\mathbf{L}_r(n)\mathbf{V}_r^T(n)$ is an approximation to the matrix, $\mathbf{\Phi}(n) = \mathbf{V}_1(n)\mathbf{\Sigma}_r^2(n)\mathbf{V}_1^T(n)$. Therefore,

$$\begin{aligned}\mathbf{V}_r(n)\mathbf{L}_r^T(n)\mathbf{L}_r(n)\mathbf{V}_r^T(n) &= \mathbf{V}_1(n)\mathbf{\Sigma}_r^2(n)\mathbf{V}_1^T(n) + \mathbf{F}(n), \quad \text{or} && (4.27)\\ \hat{\mathbf{\Phi}}(n) &= \mathbf{\Phi}(n) + \mathbf{F}(n), && (4.28)\end{aligned}$$

where $\mathbf{F}(n)$ is the error in the approximation. Thus, from (4.26) and (4.28), we obtain

$$\hat{\mathbf{\Phi}}(n)\left(\mathbf{h}_{\mathrm{ULV-LS}}(n)-\mathbf{h}_o\right)=\mathbf{F}(n)\mathbf{h}_o+\mathbf{X}^T(n)\mathbf{e}_o(n). \tag{4.29}$$

Taking the expectation of both sides of (4.29) for a given realization $\mathbf{x}(k), 1\le k\le n$, and noting that the measurement error $\mathbf{e}_o(n)$ has zero mean, we obtain

$$E[\mathbf{h}_{\mathrm{ULV-LS}}|\mathbf{x}(k), 1\le k\le n]=\mathbf{h}_o+\hat{\mathbf{\Phi}}^{\dagger}(n)\mathbf{F}(n)\mathbf{h}_o. \tag{4.30}$$

Assuming the stochastic process represented by $\mathbf{x}(n)$ is ergodic, we can approximate the ensemble averaged covariance matrix $\mathbf{R}$ of $\mathbf{x}(n)$ as,

$$\mathbf{R}\approx\frac{1}{n}\mathbf{\Phi}(n)\quad\text{large } n. \tag{4.31}$$

Thus, (4.30) can be rewritten as,

$$E[\mathbf{h}_{\mathrm{ULV-LS}}(n)|\mathbf{x}(k), 1\le k\le n]\approx\mathbf{h}_o+\frac{1}{n}(\mathbf{R}+\frac{1}{n}\mathbf{F}(n))^{\dagger}\mathbf{F}(n)\mathbf{h}_o. \tag{4.32}$$

In the above expression, as n tends to infinity, the perturbation $\frac{1}{n}\mathbf{F}(n)$ tends to a finite value due to the inaccuracy in the subspaces estimated by the ULV decomposition. In fact, it has been shown that [18]

$$\frac{\|\mathbf{U}_1\mathbf{\Sigma}_r\mathbf{V}_1^T-\mathbf{U}_r\mathbf{L}_r\mathbf{V}_r^T\|}{\|\mathbf{U}_1\mathbf{\Sigma}_r\mathbf{V}_1^T\|}\le\sin\theta(\mathcal{R}(\mathbf{V}_r),\mathcal{R}(\mathbf{V}_1)). \tag{4.33}$$

Thus, unlike traditional RLS algorithm, which is asymptotically unbiased [12], the ULV-RLS algorithm has a small bias. However, the magnitude of this bias depends on the closeness of the ULV subspace, $\mathbf{V}_r$, to its corresponding singular subspace. This closeness, in turn, depends on the magnitude of the off diagonal matrix $\mathbf{H}$ (See Theorem 1). Therefore, using extra refinements, it is possible to reduce the norm of $\mathbf{H}$ to close to zero and make the bias negligible.

4.2 Mean squared error

Let us now perform a convergence analysis of the RLS algorithm based on the mean squared value of the a priori estimation error of the RLS algorithm.

The a priori estimation error $\alpha(n)$ is given by

$$\alpha(n)=d(n)-\mathbf{h}_{\mathrm{ULV-LS}}^T(n-1)\mathbf{x}(n). \tag{4.34}$$

Eliminating $d(n)$ between (4.34) and (4.21), we obtain

$$\alpha(n)=e_o(n)-(\mathbf{h}_{\mathrm{ULV-LS}}(n-1)-\mathbf{h}_o)^T\mathbf{x}(n)\;=\;e_o(n)-\boldsymbol{\epsilon}^T(n-1)\mathbf{x}(n). \tag{4.35}$$

Now note that it follows from (4.29) that

$$\mathbf{K}(n)=E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^T(n)|\mathbf{x}(k), 1\le k\le n]\;=\;\hat{\mathbf{\Phi}}^{\dagger}(n)\mathbf{F}(n)\mathbf{h}_o\mathbf{h}_o^T\mathbf{F}(n)\hat{\mathbf{\Phi}}^{\dagger}(n)+\sigma^2\hat{\mathbf{\Phi}}^{\dagger}(n)\mathbf{\Phi}(n)\hat{\mathbf{\Phi}}^{\dagger}(n). \tag{4.36}$$

Define, $J'(n) = \alpha^2(n)$. We then have

$$\begin{aligned}
E[J'(n)|\mathbf{x}(k), 1 \le k \le n] &= E[\alpha^2(n)|\mathbf{x}(k), 1 \le k \le n] \\
&= E[e_o^2(n)|\mathbf{x}(k), 1 \le k \le n] \\
&\quad + E[\mathbf{x}^T(n)\boldsymbol{\epsilon}(n-1)\boldsymbol{\epsilon}^T(n-1)\mathbf{x}(n)|\mathbf{x}(k), 1 \le k \le n] \\
&\quad - E[\boldsymbol{\epsilon}^T(n-1)\mathbf{x}(n)e_o(n)|\mathbf{x}(k), 1 \le k \le n] \\
&\quad - E[e_o(n)\mathbf{x}^T(n)\boldsymbol{\epsilon}(n-1)|\mathbf{x}(k), 1 \le k \le n] \qquad (4.37) \\
&= \sigma^2 + \mathrm{Tr}[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{K}(n-1)], \qquad (4.38)
\end{aligned}$$

where we used the fact that $\epsilon(n-1)$ is independent of $e_o(n)$ given $\mathbf{x}(k), 1 \le k \le n$.

We therefore have

$$E[J'(n)|\mathbf{x}(k), 1 \le k \le n] = \sigma^2 + \mathrm{Tr}[\mathbf{x}(n)\mathbf{x}^T(n)\hat{\boldsymbol{\Phi}}^\dagger(n)\mathbf{F}(n)\mathbf{h}_o\mathbf{h}_o^T\mathbf{F}(n)\hat{\boldsymbol{\Phi}}^\dagger(n)] + \sigma^2\mathrm{Tr}[\mathbf{x}(n)\mathbf{x}^T(n)\hat{\boldsymbol{\Phi}}^\dagger(n)\boldsymbol{\Phi}(n)\hat{\boldsymbol{\Phi}}^\dagger(n)]. \qquad (4.39)$$

Notice that the second term in the above equation depends on the distance between the ULV and the SVD subspaces. Now, the magnitude of $\frac{1}{n}\mathbf{F}(n)$ depends on the norm of the off diagonal matrix $\mathbf{H}$ (see [18], (4.33) and Theorem 1). This norm can be made arbitrarily small using extra refinements. Furthermore,the magnitude of the third term in the RHS of (4.39) is $O(\frac{1}{n})$. Therefore, for small perturbations the a priori MSE is

$$E[J'(n)|\mathbf{x}(k), 1 \le k \le n] \approx \sigma^2 + \sigma^2\mathrm{Tr}[\mathbf{x}(n)\mathbf{x}^T(n)\boldsymbol{\Phi}^\dagger(n)\boldsymbol{\Phi}(n)\boldsymbol{\Phi}^\dagger(n)] \;=\; \sigma^2 + \sigma^2\mathrm{Tr}[\mathbf{x}(n)\mathbf{x}^T(n)\boldsymbol{\Phi}^\dagger(n)]. \qquad (4.40)$$

If we now take the expectation of both sides of the above equation with respect to $\mathbf{x}(\cdot)$ and use (4.31) and the definition of $\boldsymbol{\Phi}(n)$, we obtain for large n

$$E[J'(n)] \approx \sigma^2 + \frac{r\sigma^2}{n}. \qquad (4.41)$$

Based on (4.41), we can make the following observations about the ULV-RLS algorithm: 1) the ULV-RLS algorithm converges in the mean square in about $r + N$ iterations, i.e., its rate of convergence is of the same order as that of the traditional RLS algorithm, and 2) if the quality of the subspaces approximated are high, e.g., in a stationary environment, the a priori MSE of the ULV-RLS approaches the variance of the measurement error. Therefore, in theory, in such an environment it has a zero excess MSE. Thus, its MSE performance is similar to that of the traditional RLS algorithm.

5 Generalized ULV Update

As mentioned in the introduction, the ULV decomposition tracks only two subspaces, the dominant signal subspace and the smaller singular value subspace. Even though, the dominant subspace contains strong signal components, its condition number might still be large. Thus, a low rank LMS algorithm which uses the ULV decomposition would still have a poor convergence performance. We therefore generalize the

ULV decomposition in this section to track subspaces corresponding to more than two clusters of singular values. Each iteration of the GULV decomposition can be thought of as a recursive application of the ULV decomposition on the larger singular value cluster, i.e., the ULV decomposition is applied to obtain two singular value clusters. The ULV decomposition is applied to the larger singular value cluster to decompose it into two clusters. The ULV decomposition is again applied to the larger of these clusters and so on.

The following GULV decomposition primitive procedures are implemented by calling the ordinary ULV decomposition procedures.

- `Generalized_Absorb_One`. Add a new row and update all the rank boundaries. This procedure just calls `Absorb_One` using the outermost boundary, i.e., with the data structure $[\mathbf{L}, \mathbf{V}, r_k]$ (assuming that there are $k+1$ clusters). This has the effect of incrementing r_k. The resulting rotations have the effect of expanding the top group of singular values by one extra row, hence all the inner boundaries, $r_1, \cdots r_{k-1}$ are incremented by one.
- `Generalized_Deflate_One`. This procedure deflates the lowest (outermost) singular value boundary provided to it. Thus if r_l is the boundary provided to this procedure, `Deflate_One` is applied to $[\mathbf{L}, \mathbf{V}, r_l]$. As in `Generalized_Absorb_One`, the upper boundaries must be incremented by one. In order to restore the separation that existed between all the singular value clusters before application of these update procedures, the upper boundaries must be deflated. Therefore, deflation of the r_{l-1} boundary necessitates that all boundaries inner to r_{l-1} be incremented by one. In particular, r_{l-2} has to be deflated twice. This further implies that all boundaries inner to r_{l-2} have to be once again incremented by two and so on. However, while incrementing the inner boundaries, care should be taken so that any inner boundary value is never greater than its next outer boundary i.e., the i^{th} boundary $r_i \leq r_{i+1}$. Thus the number of deflations at any boundary, r_i, turns out to be the separation between r_i and r_{i+1} that existed before `Generalized_Absorb_One`. As the deflations on any boundary r_i are performed using `Deflate_One` on $[\mathbf{L}, \mathbf{V}, r_i]$, all rank boundaries outer to r_i, i.e., $r_{i+1}, \cdots r_k$ are not modified. In other words, the `Generalized_Deflate_One` procedure just deflates the outer most boundary by one and restores all the existing separations between inner boundaries.
- `Generalized_Deflate_To_Gap`. This procedure is similar to the `Deflate_To_Gap` procedure of the ULV. When applied on the i^{th} rank boundary, represented by the rank index r_i, it uses the heuristic used by `Deflate_To_Gap` to try to move the boundary toward a gap among the singular values. The smallest singular value of the current cluster, s_{r_i}, is compared with the smallest singular value of the next cluster i.e., $s_{r_{i+1}}$. Note that for the outermost cluster $s_{r_{i+1}}$ is the same as σ_N. The procedure then uses the heuristic that a gap exists if $s_{r_i} > d s_{r_{i+1}}$, where $d > 1$ is the user chosen `Spread`. If this condition fails, `Generalized_Deflate_One` is called repeatedly until this condition is satisfied. Note that we need to compute $s_{r_{i+1}}$ for only the outermost rank boundary (for this boundary, $s_{r_{i+1}}$ is the smallest singular value of $\mathbf{L}$). The update algorithm follows a "bottom-up" approach, i.e., outer rank boundaries are updated before the inner ones are. Thus, when updating an inner rank boundary, the value $s_{r_{i+1}}$ of

the minimum singular value of the next cluster is available at no extra cost. Therefore this approach would require only $O(N^2)$ flops more than the heuristic given in [16] in order to obtain the smallest singular value of $\mathbf{L}$. This extra complexity can be avoided if an estimate of this singular value (e.g., an estimate of the noise power) is provided by the user. The main advantage of this heuristic over that proposed in [16] is that it avoids the need for the user to provide the different tolerances needed to check if a gap exists at each rank boundary.

- `Generalized_Update`: This procedure encompasses the entire process. It takes an old GULV decomposition and a new row to append, and incorporates the row into the GULV decomposition. The new row is absorbed, and the new rank boundaries are found using the procedures described above.

5.1 Performance Bounds

The bounds derived for the quality of the subspaces obtained using the ULV algorithm can be directly applied to determine the bounds on the quality of subspaces using the GULV algorithm. Consider the case where there are four clusters of singular values. The GULV decomposition for this case is given by

$$\mathbf{A} = (\, \mathbf{U}_{k_1} \quad \mathbf{U}_{k_2} \quad \mathbf{U}_{k_3} \quad \mathbf{U}_{-} \,) \begin{pmatrix} \mathbf{L}_{k_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{H}_1 & \mathbf{L}_{k_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{L}_{k_3} & \mathbf{0} \\ \mathbf{H}_4 & \mathbf{H}_5 & \mathbf{H}_6 & \mathbf{E} \end{pmatrix} (\, \mathbf{V}_{k_1} \quad \mathbf{V}_{k_2} \quad \mathbf{V}_{k_3} \quad \mathbf{V}_0 \,)^T . \tag{5.1}$$

In the above decomposition, any lower triangular portion of the $\mathbf{L}$ matrix together with the corresponding trailing part is a valid ULV decomposition. For example $\begin{pmatrix} \mathbf{L}_{k_1} & 0 \\ \mathbf{H}_1 & \mathbf{L}_{k_2} \end{pmatrix}$ corresponds to the leading lower triangular portion while the rest of the matrix, $\begin{pmatrix} \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{L}_{k_3} & \mathbf{0} \\ \mathbf{H}_4 & \mathbf{H}_5 & \mathbf{H}_6 & \mathbf{E} \end{pmatrix}$ corresponds to the trailing part of a valid ULV decomposition. In such a case, the subspace spanned by $(\, \mathbf{V}_{k_1} \quad \mathbf{V}_{k_2} \,)$ would correspond to the large singular value subspace and the remaining columns of $\mathbf{V}$ would span the subspace corresponding to the smaller singular values. Thus, for these subspaces, the bounds discussed in the previous section are valid.

Consider the orthonormal matrix $\mathbf{Z} = (\, \mathbf{Z}_1 \quad \mathbf{Z}_2 \quad \mathbf{Z}_3 \quad \mathbf{Z}_4 \,)$ where the sub-matrices $\mathbf{Z}_k$ are mutually orthogonal. Let its perturbed counterpart be $\tilde{\mathbf{Z}} = (\, \tilde{\mathbf{Z}}_1 \quad \tilde{\mathbf{Z}}_2 \quad \tilde{\mathbf{Z}}_3 \quad \tilde{\mathbf{Z}}_4 \,)$ where the sub-matrices $\tilde{\mathbf{Z}}_k$ are again mutually orthogonal. Consider the product $\mathbf{Z}^T\tilde{\mathbf{Z}}$

$$\mathbf{Z}^T\tilde{\mathbf{Z}} = \begin{pmatrix} \mathbf{Z}_1^T\tilde{\mathbf{Z}}_1 & \mathbf{Z}_1^T\tilde{\mathbf{Z}}_2 & \mathbf{Z}_1^T\tilde{\mathbf{Z}}_3 & \mathbf{Z}_1^T\tilde{\mathbf{Z}}_4 \\ \mathbf{Z}_2^T\tilde{\mathbf{Z}}_1 & \mathbf{Z}_2^T\tilde{\mathbf{Z}}_2 & \mathbf{Z}_2^T\tilde{\mathbf{Z}}_3 & \mathbf{Z}_2^T\tilde{\mathbf{Z}}_4 \\ \mathbf{Z}_3^T\tilde{\mathbf{Z}}_1 & \mathbf{Z}_3^T\tilde{\mathbf{Z}}_2 & \mathbf{Z}_3^T\tilde{\mathbf{Z}}_3 & \mathbf{Z}_3^T\tilde{\mathbf{Z}}_4 \\ \mathbf{Z}_4^T\tilde{\mathbf{Z}}_1 & \mathbf{Z}_4^T\tilde{\mathbf{Z}}_2 & \mathbf{Z}_4^T\tilde{\mathbf{Z}}_3 & \mathbf{Z}_4^T\tilde{\mathbf{Z}}_4 \end{pmatrix} . \tag{5.2}$$

The distance between the subspaces $\mathcal{R}(\mathbf{Z}_1)$ and $\mathcal{R}(\tilde{\mathbf{Z}}_1)$ is bounded by the norm of the matrix $(\, \mathbf{Z}_1^T\tilde{\mathbf{Z}}_2 \quad \mathbf{Z}_1^T\tilde{\mathbf{Z}}_3 \quad \mathbf{Z}_1^T\tilde{\mathbf{Z}}_4 \,)$. Similarly, the distance between the subspaces $\mathcal{R}((\, \mathbf{Z}_1 \quad \mathbf{Z}_2 \,))$ and $\mathcal{R}((\, \tilde{\mathbf{Z}}_1 \quad \tilde{\mathbf{Z}}_2 \,))$ is bounded by the norm of the matrix $\begin{pmatrix} \mathbf{Z}_1^T\tilde{\mathbf{Z}}_3 & \mathbf{Z}_1^T\tilde{\mathbf{Z}}_4 \\ \mathbf{Z}_2^T\tilde{\mathbf{Z}}_3 & \mathbf{Z}_2^T\tilde{\mathbf{Z}}_4 \end{pmatrix}$. A bound on the distance between the subspaces $\mathcal{R}(\mathbf{Z}_2)$ and $\mathcal{R}(\tilde{\mathbf{Z}}_2)$ can be obtained by noting that elements of the matrix $(\, \mathbf{Z}_2^T\tilde{\mathbf{Z}}_1 \quad \mathbf{Z}_2^T\tilde{\mathbf{Z}}_3 \quad \mathbf{Z}_2^T\tilde{\mathbf{Z}}_4 \,)$ are elements of the matrices required to characterize the distances between the subspaces $\mathcal{R}(\mathbf{Z}_1)$ and $\mathcal{R}(\tilde{\mathbf{Z}}_1)$ and $\mathcal{R}((\,\mathbf{Z}_1 \quad \mathbf{Z}_2\,))$ and $\mathcal{R}((\,\tilde{\mathbf{Z}}_1 \quad \tilde{\mathbf{Z}}_2\,))$. Thus, a bound on the quality of the subspaces, $\tilde{\mathbf{Z}}_2$, i.e., the distance between $\mathcal{R}(\mathbf{Z}_2)$ and $\mathcal{R}(\tilde{\mathbf{Z}}_2)$, is the square root of the sum of squares of the bounds on the distance between $\mathcal{R}(\mathbf{Z}_1)$ and $\mathcal{R}(\tilde{\mathbf{Z}}_1)$, and the distance between $\mathcal{R}((\,\mathbf{Z}_1 \quad \mathbf{Z}_2\,))$ and $\mathcal{R}((\,\tilde{\mathbf{Z}}_1 \quad \tilde{\mathbf{Z}}_2\,))$.

In general if the matrix $\mathbf{Z}$ (correspondingly $\tilde{\mathbf{Z}}$) is partitioned into L groups, for the subspace spanned by the l^{th} group $\mathcal{R}(\mathbf{Z}_l)$ we have

$$\begin{aligned}
\sin\theta(\mathcal{R}(\mathbf{Z}_l), \mathcal{R}(\tilde{\mathbf{Z}}_l)) \leq\; & \big((\text{upper bound on } \sin\theta(\mathcal{R}((\,\mathbf{Z}_1 \quad \cdots \quad \mathbf{Z}_{l-1}\,)), \mathcal{R}((\,\tilde{\mathbf{Z}}_1 \quad \cdots \quad \tilde{\mathbf{Z}}_{l-1}\,))))^2 \\
& + \quad (\text{upper bound on } \sin\theta(\mathcal{R}((\,\mathbf{Z}_{l+1} \quad \cdots \quad \mathbf{Z}_L\,)), \mathcal{R}((\,\tilde{\mathbf{Z}}_{l+1} \quad \cdots \quad \tilde{\mathbf{Z}}_L\,))))^2\big)^{\frac{1}{2}} \\
\leq\; & \text{upper bound on} \sin\theta(\mathcal{R}((\,\mathbf{Z}_1 \quad \cdots \quad \mathbf{Z}_{l-1}\,)), \mathcal{R}((\,\tilde{\mathbf{Z}}_1 \quad \cdots \quad \tilde{\mathbf{Z}}_{l-1}\,))) \\
& + \quad \text{upper bound on} \sin\theta(\mathcal{R}((\,\mathbf{Z}_{l+1} \quad \cdots \quad \mathbf{Z}_L\,)), \mathcal{R}((\,\tilde{\mathbf{Z}}_{l+1} \quad \cdots \quad \tilde{\mathbf{Z}}_L\,))) \qquad (5.3)
\end{aligned}$$

In order to generalize the bounds of Theorem 1 for the k^{th} GULV subspace, we first replace $\tilde{\mathbf{Z}}$ by the matrices $\mathbf{V}$ or $\mathbf{U}$, obtained using the GULV decomposition, and $\mathbf{Z}$ by the corresponding matrix obtained using the SVD in (5.3). We now note that the upper bounds in the RHS of (5.3) are bounds on valid ULV subspaces and can be obtained from Theorem 1. Substituting these bounds into (5.3) generalizes Theorem 1.

Again, to generalize Theorem 2, we replace $\tilde{\mathbf{Z}}$ by $\tilde{\mathbf{V}}$ or $\tilde{\mathbf{U}}$, obtained using the GULV decomposition on $\tilde{\mathbf{A}}$ and the corresponding matrix obtained using the GULV decomposition on $\mathbf{A}$, in (5.3). The upper bounds in the RHS of (5.3) are obtained by noting that these are bounds on valid ULV subspaces and applying Theorem 2. Theorem 3 can also be generalized in a similar fashion.

6 Generalized ULV-LMS Algorithm

The LMS algorithm tries to minimize the mean squared value of the error $e(n)$, given by (3.3) by updating the weight vector $\mathbf{h}(n)$ with each new data sample received as

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu\mathbf{x}(n)e(n) \qquad (6.1)$$

where the step size μ is a positive constant.

As noted in the introduction, the convergence of the LMS algorithm depends on the condition number of the input autocorrelation matrix, $\chi(\mathbf{R}_x)$ [19, 12], where $E\big[\mathbf{R}_x\big] \doteq E\big[\mathbf{x}(n)\mathbf{x}^T(n)\big]$. When all the eigenvalues of the input correlation matrix are equal, i.e., the condition number $\chi(\mathbf{R}_x) = 1$, the algorithm converges fastest. As the condition number increases (i.e., as the eigenvalue spread increases or the input correlation matrix becomes more ill-conditioned), the algorithm converges more slowly.

Instead of using a Newton-LMS or a transform domain LMS algorithm to improve the convergence speed of the LMS algorithm, we will develop here a GULV based LMS procedure. The GULV decomposition groups the singular values of any matrix into an arbitrary number of groups. The number of groups or clusters is determined automatically by the largest condition number that can be tolerated in each cluster. This condition number in turn is determined by each cluster has singular values of nearly the same order of magnitude, i.e., the condition number in each cluster is improved. If we now apply an LMS algorithm to a projection of the filter weights in each subspace, the projected weights will have faster convergence. The convergence of the overall adaptive procedure will depend on the most ill-conditioned subspace, i.e., the maximum of the ratio of the largest singular value in each cluster to its smallest singular value.

Let us transform the input using the unitary matrix $\mathbf{V}(n)$ obtained by the GULV decomposition. As the GULV decomposition is updated at relatively low cost, this would imply a savings in the computational expense. We note that $\mathbf{V}$ almost block diagonalizes $\mathbf{R}_x$ in the sense that it exactly block diagonalizes a small perturbation of it. In particular, let the input data matrix, $\mathbf{X} = [\mathbf{x}(1), \ldots, \mathbf{x}(n)]^T = \mathbf{ULV}^T$ with $\mathbf{L}$ defined by (2.2). Since $\mathbf{R}_x = \mathbf{VL}^T\mathbf{LV}^T$, $\mathbf{V}$ exactly block diagonalizes $\mathbf{R}_x - \mathbf{\Delta}$ as follows:

$$\mathbf{V}(\mathbf{R}_x - \mathbf{\Delta})\mathbf{V}^T = \begin{pmatrix} \mathbf{C}^T\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}^T\mathbf{F} \end{pmatrix}$$

where

$$\mathbf{\Delta} = \mathbf{V}^T \begin{pmatrix} \mathbf{E}^T\mathbf{E} & \mathbf{E}^T\mathbf{F} \\ \mathbf{F}^T\mathbf{E} & \mathbf{0} \end{pmatrix} \mathbf{V}.$$

Here, $\|\mathbf{\Delta}\|_F \leq f^2$ is small, with $f = \|[\mathbf{E}, \mathbf{F}]\|_F$.

Let the input data vector $\mathbf{x}(n)$ be transformed using $\mathbf{V}(n)$ as

$$\mathbf{y}(n) = \mathbf{V}^T(n)\mathbf{x}(n). \tag{6.2}$$

The first r_1 coefficients of $\mathbf{y}(n)$ belong to the subspace corresponding to the first singular value cluster, the next r_2 coefficients to the second singular value cluster and so on. The variance of coefficients of $\mathbf{y}(n)$ in each such cluster is nearly the same. This is due to the fact that each subspace is selected to cluster the singular values to minimize the condition number in that subspace. This implies that the adaptive filter coefficients in the transform domain can also be similarly clustered.

The GULV-LMS update equations for updating the transform domain adaptive filter vector $\mathbf{g}(n)$ are given as

$$\tilde{\mathbf{g}}(n+1) = \mathbf{g}(n) + \mathbf{M}e(n)\mathbf{y}(n) \tag{6.3}$$

$$\mathbf{g}(n+1) = \mathbf{Q}^T(n+1)\tilde{\mathbf{g}}(n+1) \tag{6.4}$$

where $\mathbf{M}$ is a diagonal matrix of step sizes used and $\mathbf{Q}(n+1)$ is an orthogonal matrix indicating the cumulative effect of Givens rotations performed to update $\mathbf{V}(n+1)$ from $\mathbf{V}(n)$, i.e.,

$$\mathbf{V}(n+1) = \mathbf{V}(n)\mathbf{Q}(n+1). \tag{6.5}$$

It is easy to deduce, from the fact that the output of the transform domain adaptive filter should be the same as that of the tapped delay line adaptive filter, that

$$\mathbf{g}(n) = \mathbf{V}^T(n)\mathbf{h}(n), \tag{6.6}$$

and

$$\tilde{\mathbf{g}}(n+1) = \mathbf{V}^T(n)\mathbf{h}(n+1). \tag{6.7}$$

As the transformed coefficients belonging to a single cluster have nearly the same variance, the corresponding coefficients of $\mathbf{g}(n)$ can be adapted using the same step size. In other words, the diagonal elements of $\mathbf{M}$ are clustered into values of equal step sizes. The size of each cluster depends on the size of the corresponding subspace. The adaptation within each subspace therefore has nearly optimal convergence speed. Thus, for the subspace tracking LMS filter to have a fast convergence, it should converge with the same speed in each subspace. This implies that the slow converging subspace projections (usually the ones with lower signal energy) should be assigned large step sizes. Note that the average time constant τ_{av} of the learning curve [12] is $\tau_{\text{av}} \approx \frac{1}{2\mu\lambda_{\text{av}}}$, where λ_{av} is the average eigenvalue of the input correlation matrix or the average input power. Therefore, the step size for coefficients in a subspace should be made inversely proportional to the average energy in that subspace. Now note that the diagonal values of the lower triangular matrix $\mathbf{L}$ generated by the GULV decomposition reflect the average magnitude of the singular values in each cluster. This information can therefore be directly used to select the step sizes.

As the slowly converging subspace projections are usually those subspaces with lower signal energy, a large step size for these subspaces implies that the noise in these subspaces is boosted. By not adapting in these subspaces, we can reduce the excess MSE. This can be done by setting the corresponding diagonal entries of $\mathbf{M}$ to zero. Also, in case the autocorrelation matrix is close to singular, the projections of the tap weights onto the subspaces corresponding to zero singular values need not be updated. This results in stable adaptation.

7 GULV-LMS Performance Analysis

Several analyses of the LMS and Newton-LMS algorithm have appeared in literature. The GULV-LMS algorithm differs from traditional LMS and Newton-LMS type algorithms in that the subspaces estimated by the GULV algorithm are perturbed from the true subspaces by a small amount. The goal of this section is to analyze the effect of this perturbation on the performance of the algorithm. Specifically, we will consider its effect on the mean and mean-squared behavior of the weight vectors in our algorithm. We also study its effect on the steady-state mean square error of the proposed algorithm. Our analyses are approximate in that they rely on the standard simplifying assumptions that have been used in the literature to analyze the various variants of the LMS algorithm. They nevertheless provide guide lines for selecting $\mathbf{M}$ and an understanding of the performance of the algorithms that is confirmed by simulations (cf. Section 8). Specifically, we make the following standard assumptions

1. Each sample vector $\mathbf{x}(n)$ is statistically independent of all previous vectors $\mathbf{x}(k)$, $k = 0, \ldots, n-1$,
$$E[\mathbf{x}(n)\mathbf{x}^T(k)] = \mathbf{0}, \quad k = 0, \ldots, n-1. \tag{7.1}$$
2. Each sample vector $\mathbf{x}(n)$ is statistically independent of all previous samples of the desired response, $d(k)$, $k = 0, \ldots, n-1$
$$E[\mathbf{x}(n)d(k)] = \mathbf{0}, \quad k = 0, \ldots, n-1. \tag{7.2}$$

3. The desired response at the n^{th} instance, $d(n)$ depends on the corresponding input vector $\mathbf{x}(n)$.
4. The desired response $d(n)$ and the input vector $\mathbf{x}(n)$ are jointly Gaussian.

7.1 Weight vector behavior

The LMS algorithm tries to minimize the output MSE, $E[J(n)]$, given as

$$E[J(n)] = E[e^2(n)] = E[d^2(n) - 2d(n)\mathbf{g}^T(n)\mathbf{y}(n) + \mathbf{g}^T(n)\mathbf{y}(n)\mathbf{y}^T(n)\mathbf{g}(n)]. \tag{7.3}$$

As the transformation is a unitary transformation, the above equation is equivalent to

$$E[J(n)] = E[d^2(n) - 2d(n)\mathbf{h}^T(n)\mathbf{x}(n) + \mathbf{h}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{h}(n)] = \sigma_d^2 + \mathbf{h}^T(n)\mathbf{R}_x\mathbf{h}(n) - 2\mathbf{h}^T(n)\mathbf{r}_{dx}, \tag{7.4}$$

where σ_d^2 is the variance of the desired signal and $\mathbf{r}_{dx}$ is the cross-correlation of the input vector $\mathbf{x}(n)$ and the desired output $d(n)$

$$\mathbf{r}_{dx} = E[\mathbf{x}(n)d(n)]. \tag{7.5}$$

The MSE is a quadratic function of the weight vector $\mathbf{h}(n)$, and the optimum solution corresponding to its minimum, $\mathbf{h}^*$ is the solution of he Wiener equation

$$\mathbf{R}_x\mathbf{h}^* = \mathbf{r}_{dx}. \tag{7.6}$$

In particular, the optimum weight vector, is given by

$$\mathbf{h}^* = \mathbf{R}_x^{-1}\mathbf{r}_{dx}. \tag{7.7}$$

Now consider a low rank solution to the Wiener equation (7.6). Assume that the eigenvalues of $\mathbf{R}_x$ can be clustered into p groups. Let the rank l low rank approximate for $\mathbf{R}_x$ be the matrix constructed by replacing the $p-l$ smallest eigenvalue clusters of $\mathbf{R}_x$ in its eigendecomposition by zeros, i.e.,

$$\mathbf{R}_{x,l} = \sum_{k=1}^{l}\sum_{i=1}^{p_k} \lambda_{i_k}\mathbf{s}_{i_k}\mathbf{s}_{i_k}^T, \tag{7.8}$$

where p_k is the number of eigenvalues in the k^{th} cluster. The l-order solution, $\mathbf{h}^l$, is then obtained by solving the modified Wiener equation

$$\mathbf{R}_{x,l}\mathbf{h}^l = \mathbf{r}_{dx}. \tag{7.9}$$

Note that when $l = p$, $\mathbf{h}^l = \mathbf{h}^*$.

Suppose now that we use a GULV-LMS algorithm that adapts only in the dominant l subspaces produced by the GULV decomposition. Let $\mathbf{h}(n)$ be the time domain weight vector that it produces. We now proceed to show that under the simplifying assumptions listed above, $\mathbf{h}(n)$ converges to $\mathbf{h}^l$ with a very small bias. This bias depends on the quality of the estimated subspaces i.e., how close they are to the true eigenvector subspaces. We also show that when $l = p$, $\mathbf{h}(n)$ converges to $\mathbf{h}^*$ with a zero bias as n tends to infinity.

Let the unitary transformation matrix $\mathbf{V}(n)$ be partitioned into p blocks, each block corresponding to the subspace of a cluster eigenvalues having the same order of magnitude. i.e.,

$$\mathbf{V}(n) = [\mathbf{V}_1(n)|\mathbf{V}_2(n)|\cdots|\mathbf{V}_p(n)]. \tag{7.10}$$

The orthogonal projection matrix, $\mathbf{P}_k(n)$ in the subspace corresponding to the k^{th} cluster of eigenvalues is obtained by the GULV algorithm at the n^{th} as

$$\mathbf{P}_k(n) = \mathbf{V}_k(n)\mathbf{V}_k^T(n). \tag{7.11}$$

The GULV-LMS update equation (6.3) can therefore be rewritten as

$$\mathbf{V}^T(n)\mathbf{h}(n+1) = \mathbf{V}^T(n)\mathbf{h}(n) + \mathbf{M}e(n)\mathbf{V}^T(n)\mathbf{x}(n). \tag{7.12}$$

Note that step size matrix $\mathbf{M}$ is chosen to be a block diagonal matrix, with each block being a scalar multiple of the identity of appropriate dimension. This is due to the fact that we have a single step size for the modes belonging to a single cluster of eigenvalues. Pre-multiplying the above equation by $\mathbf{V}(n)$, we obtain

$$\begin{aligned} \mathbf{h}(n+1) &= \mathbf{h}(n) + \sum_{k=1}^{l} m_k \mathbf{P}_k(n) e(n)\mathbf{x}(n) \quad (7.13) \\ &= \mathbf{h}(n) + \sum_{k=1}^{l} m_k e(n)\mathbf{x}_k(n), \quad (7.14) \end{aligned}$$

where $\mathbf{x}_k(n) = \mathbf{P}_k(n)\mathbf{x}(n)$ denotes the projection of the input vector $\mathbf{x}(n)$ onto the subspace estimate corresponding to the k^{th} cluster of eigenvalues.

We define the weight error vector as

$$\boldsymbol{\epsilon}(n) = \mathbf{h}(n) - \mathbf{h}_l, \tag{7.15}$$

where $\mathbf{h}_l$ is the modified Wiener solution as discussed above.

Subtracting $\mathbf{h}_l$ from both sides of Eq. (7.14), and using (7.15) and the definitions of $e(n)$ and $d(n)$, we obtain

$$\boldsymbol{\epsilon}(n+1) = (\mathbf{I} - \sum_{k=1}^{l} m_k \mathbf{x}_k(n)\mathbf{x}^T(n))\boldsymbol{\epsilon}(n) + \sum_{k=1}^{l} m_k (d(n)\mathbf{x}_k(n) - \mathbf{x}_k(n)\mathbf{x}^T(n)\mathbf{h}_l). \tag{7.16}$$

From Theorems 2 and 3, it can be seen that the distance between the perturbed ULV subspaces and the corresponding true subspaces depends on the amount of perturbation, ϵ. The input correlation matrix is estimated as a sample average, $\hat{\mathbf{R}}_x = 1/n\mathbf{X}^T(n)\mathbf{X}(n)$. Therefore, the perturbation ϵ tends to 0 as n tends to infinity. This implies that for large n, we can assume that $\mathbf{P}_k(n)$ converges to its steady state value $\mathbf{P}_k$, which is independent of $\mathbf{x}(n)$. Taking the expectation of both sides of Eq. (7.16), for large n, we obtain

$$\begin{aligned} E[\boldsymbol{\epsilon}(n+1)] &= E[(\mathbf{I} - \sum_{k=1}^{l} m_k \mathbf{x}_k(n)\mathbf{x}^T(n))\boldsymbol{\epsilon}(n)] + \sum_{k=1}^{l} m_k E[\mathbf{x}_k(n)d(n) - \mathbf{x}_k(n)\mathbf{x}^T(n)\mathbf{h}_l)] \\ &= (\mathbf{I} - \sum_{k=1}^{l} m_k \mathbf{P}_k \mathbf{R}_x)E[\boldsymbol{\epsilon}(n)] + \sum_{k=1}^{l} m_k \mathbf{P}_k(\mathbf{r}_{dx} - \mathbf{R}_x\mathbf{h}_l), \quad (7.17) \end{aligned}$$

where we have made use of the independence assumptions. Noting that

$$\mathbf{h}_l = \mathbf{h}^* - \sum_{j=l+1}^{p} \mathbf{P}_j \mathbf{h}^*, \tag{7.18}$$

and using Eq (7.6) we obtain,

$$E[\boldsymbol{\epsilon}(n+1)] = (\mathbf{I} - \sum_{k=1}^{l} m_k \mathbf{P}_k \mathbf{R}_x) E[\boldsymbol{\epsilon}(n)] + \sum_{k=1}^{l} m_k \sum_{j=l+1}^{p} \mathbf{P}_k \mathbf{R}_x \mathbf{P}_j \mathbf{h}^*. \tag{7.19}$$

Rewrite $\mathbf{R}_x$, in terms of its eigenvalues and eigenvectors,

$$\mathbf{R}_x = \sum_{k=1}^{p} \sum_{i=1}^{p_k} \lambda_{i_k} \mathbf{s}_{i_k} \mathbf{s}_{i_k}^T. \tag{7.20}$$

If the subspaces estimated by the GULV algorithm have converged exactly to the subspaces spanned by the corresponding clusters of eigenvectors, $\mathbf{P}_k \mathbf{R}_x$ is given by

$$\mathbf{P}_k \mathbf{R}_x = \sum_{i=1}^{p_k} \lambda_{i_k} \mathbf{s}_{i_k} \mathbf{s}_{i_k}^T. \tag{7.21}$$

This is due to the fact that $\mathbf{s}_{i_k}$'s lie in the subspace spanned by the columns of $\mathbf{P}_k$ and therefore

$$\mathbf{P}_k \mathbf{s}_{i_j} \quad = \quad \delta_{kj} \mathbf{s}_{i_j},$$

where δ_{kj} is the Kronecker-delta, $\delta_{kj} = 1$, $k = j$ and $\delta_{kj} = 0$ otherwise. Thus whenever $l = p$ or the estimated subspaces converge to the exact subspaces, the second term in Eq. (7.19) is zero.

However, the subspaces estimated by the GULV algorithm will in general be perturbed from the true subspaces by a small amount. Therefore we have

$$\mathbf{P}_k \mathbf{R}_x = \sum_{i=1}^{p_k} \lambda_{i_k} \mathbf{s}_{i_k} \mathbf{s}_{i_k}^T + \mathbf{E}_k, \tag{7.22}$$

and

$$\mathbf{P}_k \mathbf{R}_x \mathbf{P}_j = \mathbf{E}_{k,j}. \tag{7.23}$$

Using (7.23) in (7.19), the weight error update equation can be rewritten as,

$$\begin{aligned} E[\boldsymbol{\epsilon}(n+1)] &= (\mathbf{I} - \sum_{k=1}^{l} m_k (\sum_{i=1}^{p_k} \lambda_{i_k} \mathbf{s}_{i_k} \mathbf{s}_{i_k}^T + \mathbf{E}_k)) E[\boldsymbol{\epsilon}(n)] + \sum_{k=1}^{l} m_k \sum_{j=l+1}^{p} \mathbf{E}_{k,j} \mathbf{h}^* && (7.24) \\ &= \mathbf{S}(\mathbf{I} - \mathbf{M}(\boldsymbol{\Lambda} + \mathbf{E}))\mathbf{S}^T E[\boldsymbol{\epsilon}(n)] + \sum_{k=1}^{l} m_k \sum_{j=l+1}^{p} \mathbf{E}_{k,j} \mathbf{h}^*, && (7.25) \end{aligned}$$

where $\mathbf{R}_x = \mathbf{S}\boldsymbol{\Lambda}\mathbf{S}^T$ is the eigendecomposition of $\mathbf{R}_x$. By making the transformation $\tilde{\boldsymbol{\epsilon}} = \mathbf{S}^T \boldsymbol{\epsilon}$ and $\tilde{\mathbf{h}}^* = \mathbf{S}^T \mathbf{h}^*$, we can rewrite Eq. (7.25) as,

$$E[\tilde{\boldsymbol{\epsilon}}(n+1)] = (\mathbf{I} - \mathbf{M}(\boldsymbol{\Lambda} + \mathbf{E})) E[\tilde{\boldsymbol{\epsilon}}(n)] + \mathbf{F}_l \mathbf{h}^*, \tag{7.26}$$

where $\mathbf{F}_l = \sum_{k=1}^{l} m_k \sum_{j=l+1}^{p} \mathbf{E}_{k,j}$. This error equation is of the form derived for the LMS algorithm [12] and can be rewritten as

$$E[\tilde{\boldsymbol{\epsilon}}(n+1)] = (\mathbf{I} - \mathbf{M}(\boldsymbol{\Lambda} + \mathbf{E}))^{n+1} E[\tilde{\boldsymbol{\epsilon}}_0] + (\mathbf{I} - (\mathbf{M}(\boldsymbol{\Lambda} + \mathbf{E}))^n)(\mathbf{I} - \mathbf{M}(\boldsymbol{\Lambda} + \mathbf{E}))^{-1} \mathbf{F}_l \mathbf{h}^*. \tag{7.27}$$

Thus, if the step-size for each cluster m_k satisfies the condition [19], [20]

$$0 < m_k < \frac{1}{\max(\lambda_{i_k} + \eta_{i_k})}, \tag{7.28}$$

where η_{i_k} is the perturbation in the corresponding eigenvalue due to the perturbation matrix $\mathbf{E}$, we get

$$E[\tilde{\boldsymbol{\epsilon}}_\infty] = (\mathbf{I} - \mathbf{M}(\boldsymbol{\Lambda} + \mathbf{E}))^{-1}\mathbf{F}_l\mathbf{h}^*. \tag{7.29}$$

In particular, note that unlike the LMS algorithm, $E[\tilde{\boldsymbol{\epsilon}}_\infty] \neq \mathbf{0}$. It follows from Theorem 3, that if the norm of the off diagonal matrix $\tilde{\mathbf{H}}$ is reduced close to zero using refinement strategies, then for large n the ULV subspaces converge to the true SVD subspaces. Therefore, the norms $\|\mathbf{E}_{k,j}\|_F$ and $\|\mathbf{F}_l\|_F$ are usually small. Also as pointed out earlier, in case $l = p$, $\mathbf{F}_l = \mathbf{0}$. Thus, $E[\tilde{\boldsymbol{\epsilon}}_\infty] = \mathbf{0}$. Therefore, for this case, the algorithm converges in mean to the optimum weight vector $\mathbf{h}^*$.

Using the inequality (Weyl's Thm., p. 203 of [21]),

$$\lambda_i(\mathbf{A} + \mathbf{B}) \leq \lambda_i(\mathbf{A}) + \|\mathbf{B}\|_2, \tag{7.30}$$

we get,

$$\max(\lambda_{i_k} + \eta_{i_k}) \leq \max(\lambda_{i_k}) + \|\mathbf{E}\|_2. \tag{7.31}$$

Thus for convergence it is sufficient to choose m_k as,

$$0 < m_k \leq \frac{1}{\max(\lambda_{i_k}) + \|\mathbf{E}\|_2}. \tag{7.32}$$

7.2 Mean squared Error

The MSE, $E[J(n)]$ of the LMS algorithm is given by

$$E[J(n)] = E[e^2(n)] = E[(d(n) - \mathbf{g}^T(n)\mathbf{y}(n))^2]. \tag{7.33}$$

However, as the transformation is unitary, it can be written as,

$$E[J(n)] = E[(d(n) - \mathbf{h}^T(n)\mathbf{x}(n))^2] = \sigma_d^2 + \mathbf{h}^T(n)\mathbf{R}_x\mathbf{h}(n) - 2\mathbf{h}^T(n)\mathbf{r}_{dx}. \tag{7.34}$$

The above equation can be re-written in terms of the weight error vector $\boldsymbol{\epsilon}(n)$ as [12]

$$E[J(n)] = J_{\min} + E[\boldsymbol{\epsilon}^T(n)\mathbf{R}\boldsymbol{\epsilon}(n)]. \tag{7.35}$$

In the above equation, $J_{\min}$ denotes the minimum MSE achieved by the optimum weight vector. The excess MSE is then given by

$$E[J_{\text{ex}}(n)] = E[\boldsymbol{\epsilon}^T(n)\mathbf{R}\boldsymbol{\epsilon}(n)] = \text{Tr}[\mathbf{R}\mathbf{K}(n)], \tag{7.36}$$

where $\text{Tr}[\cdot]$ denotes the trace operator and $\mathbf{K}(n) = E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^T(n)]$ is the weight error correlation matrix. The misadjustment error is the excess MSE after the adaptive filter has converged, $E[J_{\text{ex}}(\infty)]$.

It is show in [22] that $E[J_{\text{ex}}(\infty)]$ is given by

$$E[J_{\text{ex}}(\infty)] \approx \frac{\text{Tr}(\mathbf{M}\boldsymbol{\Lambda})J_{\min}}{2 - \text{Tr}(\mathbf{M}\boldsymbol{\Lambda})} \tag{7.37}$$

where $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix of the input correlation matrix $\mathbf{R}_x$. The proof of (7.37) is straightforward and we omit it for lack of space.

7.3 Discussion

The condition on m_k for convergence, (7.32), is similar to that obtained in [19], [20]. Specifically, when the matrix $\mathbf{M}$ is replaced by a multiple of the identity matrix, $\mu\mathbf{I}$ and $\mathbf{E} = \mathbf{0}$, the convergence condition becomes

$$0 < \mu \leq \frac{1}{\lambda_{\max}}. \tag{7.38}$$

Note also that if the subspaces estimated using the GULV algorithm were the true subspaces and the condition number of each cluster is 1, then $\mathbf{E} = \mathbf{0}$, and by choosing $\mathbf{M}$ such that

$$\mathbf{M\Lambda} = \mu\mathbf{I}, \tag{7.39}$$

the convergence condition becomes

$$0 < \mu < 1. \tag{7.40}$$

This is equivalent to whitening the input process by preconditioning with the appropriate $\mathbf{M}$. In practice, the elements of $\mathbf{E}$ are negligible and $\sigma_{\max}(\mathbf{E})$ is very small. Further, the step size for each cluster m_k is chosen to be the inverse of the estimate of the variance of the component of the input signal that lies in the cluster (i.e., the inverse of the average of the eigenvalues in the cluster). This implies that the condition for convergence (7.32) is almost always satisfied in practice. Choosing the step sizes as the inverse of the estimated power in each cluster also matches the speeds of adaptation across the clusters.

The step sizes for modes/subspaces which contain essentially noise and little signal components can be chosen to be very small. As the subspaces have been decoupled on the basis of signal strengths, adapting very slowly or not at all in the noise only subspaces leads to little loss of information which is confirmed by simulations (cf. Sec. 8). As discussed above, adapting only in the l dominant subspaces corresponds to adaptively estimating the solution to the modified Wiener equation (7.9). This implies that there is an inherent "noise cleaning" associated with such an approach. It is noted [23] that as we slowly increase the number of nonzero m_k's corresponding to the subspaces containing significant signal strengths, the MSE decreases until the desired signals are contained in these spaces. A further increase would only increase the MSE due to the inclusion of the noise eigenvalues. Also note that the solution to the unmodified normal equations (7.6) involve inverting the input correlation matrix. Therefore, the contribution of the noise eigenvalues to the MSE using this solution is inversely proportional to their magnitudes. Hence, if the noise eigenvalues are small (high SNR), this amounts to noise boosting, resulting in a larger MSE. As conventional adaptive filters recursively estimate this solution their MSE at convergence is also high.

Eqs. (7.25) and (7.26) give a recursive update equation for the weight error vector. The speed with which this weight error vector tends to zero determines the speed of convergence of the algorithm. Assume the GULV projections have converged at step K. Also assume that we can cluster the eigenvalues of $\mathbf{R}_x$ into p

clusters, i.e., the diagonal eigenvalue matrix, $\mathbf{\Lambda}$, of $\mathbf{R}_x$ can be written as,

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \cdots & 0 \\ & \mathbf{\Lambda}_2 & \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{\Lambda}_p \end{bmatrix}, \tag{7.41}$$

where $\mathbf{\Lambda}_k$ is the diagonal matrix of eigenvalues corresponding to the k^{th} cluster. Rewriting (7.26) in terms of some weight error vector $\tilde{\boldsymbol{\epsilon}}_K$, we obtain for $n > K$,

$$\begin{aligned} E[\tilde{\boldsymbol{\epsilon}}(n+1)] &= (\mathbf{I} - \mathbf{M}(\mathbf{\Lambda} + \mathbf{E}))^{n+1-K} E[\tilde{\boldsymbol{\epsilon}}_K] \\ &= (\mathbf{I} - \text{diag}(m_1\mathbf{I}, \cdots, m_p\mathbf{I})[\text{diag}(\mathbf{\Lambda}_1, \cdots, \mathbf{\Lambda}_p) + \mathbf{E}]^{n+1-K} E[\tilde{\boldsymbol{\epsilon}}_K]. \end{aligned} \tag{7.42}$$

For sufficiently small $\mathbf{E}$, Eq. (7.42) indicates that the convergence speed in each subspace depends on the step size matrix $m_k\mathbf{I}$ and the condition number of $\mathbf{\Lambda}_k$. For the same step size, the modes corresponding to smaller eigenvalues of $\mathbf{\Lambda}_k$ converge more slowly than those corresponding to its larger eigenvalues. Also note that to achieve minimum MSE, one needs to adapt only in the signal subspaces. Therefore, the convergence of the GULV-LMS adaptive filter to the required solution depends only on the condition number of the cluster identified by the GULV decomposition corresponding to each of the $\mathbf{\Lambda}_k$.

For the k'th subspace identified by the GULV algorithm, the condition number is given as

$$\chi(k^{\text{th}} \text{ subspace}) = \frac{\max{(\lambda_{i_k} + \eta_{i_k})}}{\min{(\lambda_{i_k} + \eta_{i_k})}} \leq \frac{\max{(\lambda_{i_k})} + \sigma\text{max}(\mathbf{E})}{\min{(\lambda_{i_k})} - \sigma\text{max}(\mathbf{E})}. \tag{7.43}$$

However, as noted above, in steady state the perturbation is very small and the condition number of the subspace identified by the GULV decomposition is close to the condition number of the corresponding cluster of eigenvalues. The speed of the adaptive algorithm therefore depends on the speed of convergence in that subspace which has the maximum condition number. By proper application of the GULV algorithm and choice of the subspaces, this condition number can be made to be close to unity for fast convergence.

The misadjustment error expression given by Eq. (7.37) is approximate. In deriving this result, we have made use of the fact that $\mathbf{E}$ is a very small perturbation, and can be neglected. Now if the step size m_k for the k^{th} dominant cluster is chosen such that

$$m_k = \mu/\text{Tr}(\mathbf{\Lambda}_k),$$

where $\text{Tr}(\mathbf{\Lambda}_k)$ is the total input power in that cluster, then it can be easily verified that $\text{Tr}(\mathbf{M\Lambda}) = \mu \times$ (size of the input signal subspace). Thus, for small μ, the misadjustment error depends linearly on the step size and the effective rank of the input signal.

We therefore conclude that for a step size leading to the same misadjustment error, the GULV-LMS algorithm would converge at least as fast as the LMS algorithm. It would converge faster than the LMS algorithm, when the condition number of the input correlation matrix $\chi(\mathbf{R}_x)$ is large.

8 Simulation Results

An Adaptive Line Enhancer (ALE) experiment was conducted to illustrate the performance of the algorithm when the adaptation is done only in the signal subspaces. The input to the ALE was chosen to be $0.01\cos\left(\frac{\pi}{15}n\right) + \cos\left(\frac{5\pi}{16}n\right)$. White Gaussian noise with a variance of -60 dB and -160 dB was added to obtain the learning curves of Fig. 3 and Fig. 4 respectively. The figures show the performance of the GULV-LMS algorithm, the plain ULV-LMS, plain LMS, traditional RLS, QRD-RLS, GULV-RLS and the Schur pseudoinverse based least squares method discussed in [2]. The learning curves show the improved performance of the GULV-LMS algorithm. It can be seen from these figures that breaking up the subspaces corresponding to the larger singular values into sub clusters using the GULV algorithm further reduces the condition numbers of these clusters. This in turn yields an improvement in the convergence performance of the GULV-LMS algorithm.

Fig. 4 also demonstrates the stability of the GULV based RLS and LMS algorithms when the input matrix becomes numerically ill-conditioned. As the GULV based algorithms are based on Givens rotations, they enjoy the same stability properties of the QRD-RLS algorithm.

Fig. 5 shows the tracking behavior of the GULV based procedures. The input to the ALE in this figure is initially $\cos\left(\frac{\pi}{15}n\right)$. A new signal, $0.01\cos\left(\frac{5\pi}{16}n\right)$ was added to the input at the 500^{th} iteration. Thus the input correlation matrix initially has a rank of 2 which suddenly changes to 4. The GULV algorithm tracks this change of rank. Also, it divides the larger singular value cluster (which now has 4 singular values) into two clusters with two singular values of the same order of magnitude in each cluster. This results in the improved performance of the GULV based RLS and LMS algorithms. On the other hand, the algorithm of [2] has a starting value of the rank set to 2. Therefore its behavior is similar to that of the GULV based algorithms initially. However, as it is not able to track the rank change, its performance worsens.

9 Conclusions

In this paper we developed the GULV algorithm for isolating subspaces corresponding to clusters of singular values of the same order of magnitude. A new heuristic was also suggested for deflation in the ULV decomposition. We derived a new bound on the distance between the subspaces obtained using a ULV algorithm on a perturbed data matrix and those obtained using the SVD on an exact data matrix. For the special case where the off-diagonal block is zero, this bound reduces to the perturbation bounds on the SVD subspaces. The bounds on the accuracy of the ULV subspaces were also extended to those isolated using the GULV algorithm. The GULV algorithm was then used to obtain a subspace domain LMS algorithm to improve the convergence rate whenever the input autocorrelation matrix is ill-conditioned. The rate of convergence of this algorithm depends on the largest condition number among the clusters of singular values isolated using the GULV decomposition. This in turn depends on the user defined `Spread`. However, there is a tradeoff between the convergence speed and the computational expense of the GULV algorithm. We also demonstrated the power of this algorithm, using simulations, by adapting only in subspaces containing strong signal components. The GULV algorithm was also used to obtain a low-rank GULV-RLS algorithm.

Present work involves investigating the tradeoffs between error due to non-adaptation in noisy subspaces and the reduction in the excess MSE. Future work targets reducing the computational complexity of the GULV algorithm.

References

[1] G. H. Golub and C. F. Van Loan, *Matrix Computations.* The John Hopkins University Press, 1983.

[2] P. Strobach, "Fast recursive eigensubspace adaptive filters," in *Proc. ICASSP*, (Detroit), May 1995.

[3] T. F. Chan and P. C. Hansen, "Some applications of the rank revealing QR factorization," *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 727–741, 1992.

[4] R. D. Fierro and J. R. Bunch, "Bounding the subspaces from rank revealing two-sided orthogonal decompositions." Preprint (to appear in SIAM Matrix Anal. Appl.).

[5] C. E. Davila, "Recursive total least squares algorithm for adaptive filtering," in *Proc. ICASSP*, pp. 1853–1856, May 1991.

[6] K. B. Yu, "Recursive updating the eigenvalue decomposition of a covariance matrix," *IEEE Trans. Signal Processing*, vol. 39, pp. 1136–1145, 1991.

[7] E. M. Dowling and R. D. DeGroat, "Recursive total least squares adaptive filtering," in *Proceedings of the SPIE Conf. on Adaptive Signal Proc.*, vol. 1565, pp. 35–46, SPIE, July 1991.

[8] R. D. DeGroat, "Noniterative subspace tracking," *IEEE Trans. Signal Processing*, vol. 40, pp. 571–577, 1992.

[9] D. L. Boley and K. T. Sutherland, "Recursive total least squares : An alternative to the discrete kalman filter," Tech. Rep. TR 93-92, Dept. of Comp. Sci., Univ. of Minn., 1993.

[10] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Trans. Signal Processing*, vol. 40, pp. 1535–1541, June 1992.

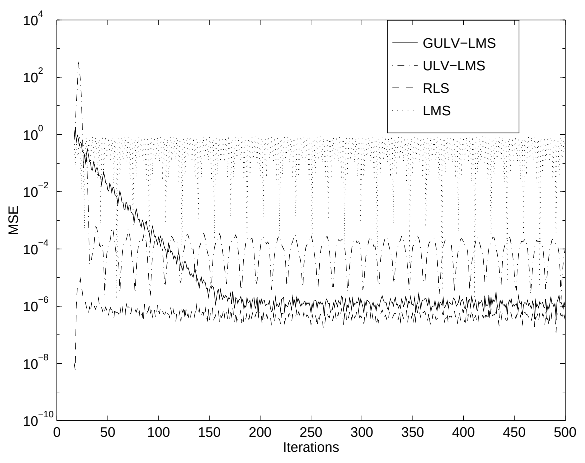
[11] R. D. Fierro, "Perturbation analysis for two-sided (or complete) orthogonal decompositions." Preprint.

[12] S. Haykin, *Adaptive Filter Theory.* Engelwood Cliffs, N.J.: Prentice Hall, 1991.

[13] D. F. Marshall, W. K. Jenkins, and J. J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 474–483, April 1989.

[14] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, MD: Johns Hopkins University Press, 2nd ed., 1989.

[15] N. J. Higham, "A survey of condition number estimators for triangular matrices," *SIAM Rev*, pp. 575–596, 1987.



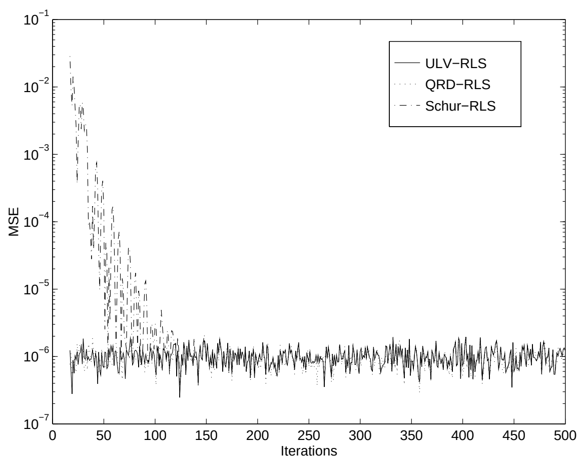


Figure 3: Learning curves illustrating performance. Noise at -60 dB. Curves are averages of 20 runs.

[16] G. W. Stewart, "Updating a rank revealing ULV decomposition," Tech. Rep. UMIACS-TR-91-39 and CS-TR 2627, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, March 1991.

[17] P. Å. Wedin, "Perturbation bounds in connection with singular value decomposition," *BIT*, vol. 12, pp. 99–111, 1973.

[18] R. D. Fierro and P. C. Hansen, "Accuracy of TSVD solutions computed from rank revealing decompositions." Preprint.

[19] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Engelwood Cliffs, N.J.: Prentice Hall, 1985.

[20] G. Ungerboeck, "Theory on the speed of convergence in adaptive equalizers for digital communication," *IBM Journal of Research and Development*, vol. 16, pp. 546–555, 1972.

[21] G. W. Stewart and J. Sun, *Matrix Perturbation Theory.* San Diego, USA: Academic Press Inc., 1990.

[22] S. Hosur, *Recursive Matrix Factorization Algorithms in Adaptive Filtering and Mobile Communications.* PhD thesis, University of Minnesota, October 1995.

[23] J.S.Goldstein, M.A.Ingram, E.J.Holder, and R. Smith, "Adaptive subspace selection using subband decompositions for sensor array processing," in *Proc. ICASSP*, vol. IV, pp. 281–284, April 1994.

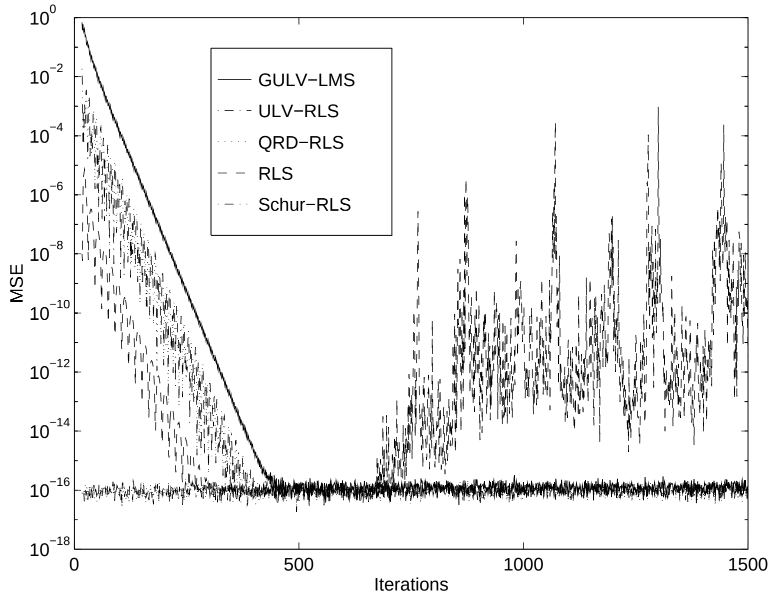


Figure 4: Learning curves illustrating behavior when input correlation matrix is extremely ill conditioned. Noise at -160 dB. Curves are averages of 20 runs.

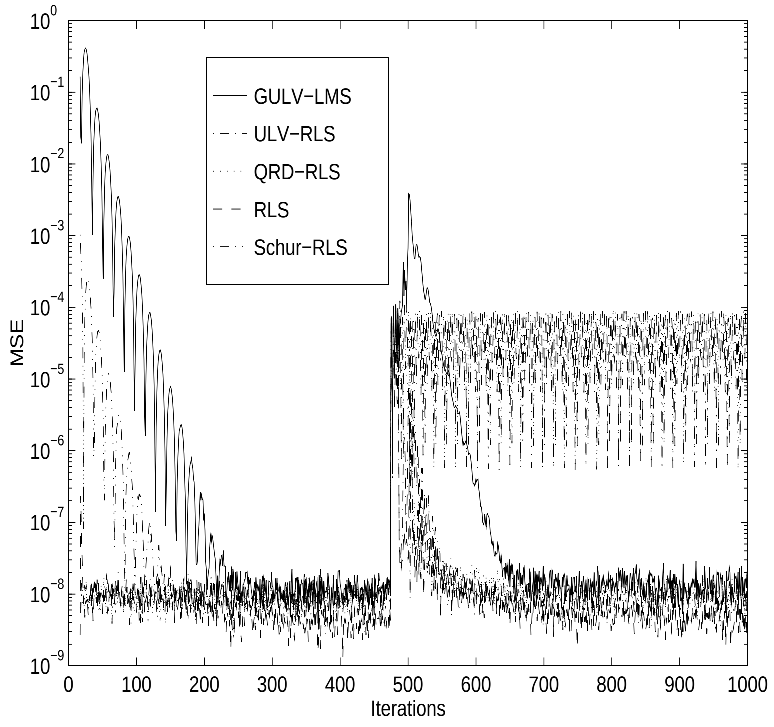


Figure 5: Learning curves illustrating rank tracking performance. Curves are averages of 20 runs.