EDICS category SP 2-ADPT

Efficient, High Performance, Subspace Tracking for Time-Domain Data

Carlos E. Davila

Electrical Engineering Department, Southern Methodist University
Dallas, Texas 75275
Phone: (214) 768-3197
Fax: (214) 768-3573
Email: cd@seas.smu.edu

Abstract

This paper describes two new algorithms for tracking the subspace spanned by the principal eigenvectors of the correlation matrix associated with time-domain (i.e. time series) data. The algorithms track the d principal N-dimensional eigenvectors of the data covariance matrix with a complexity of $O(Nd^2)$, yet have performance comparable to algorithms having $O(N^2d)$ complexity. The computation reduction is achieved by exploiting the shift-invariance property of temporal data covariance matrices. Experiments are used to compare our algorithms with other well-known approaches of similar and/or lower complexity. Our new algorithms are shown to outperform the subset of the general approaches having the same complexity. The new algorithms are useful for applications such as subspace-based speech enhancement, and low-rank adaptive filtering.

May 12, 2000

I. Background

The problem of subspace tracking can be formulated as follows, let

$$x_n = \left[\begin{array}{cccc} x(n) & x(n-1) & \cdots & x(n-N+1) \end{array} \right]^T$$

be the vector whose covariance matrix is given by $R_x = E\left[x_n x_n^T\right]$. In some applications, x_n can be modeled as $x_n = Au_n + v_n$ where the $N \times d$ matrix A is constant, the $d \times 1$ vector u_n is a random vector having the property $E\left[u_n u_n^T\right] = P$, and the $N \times 1$ vector v_n corresponds to additive white noise having variance σ_v. It follows that $R_x = APA^T + \sigma_v I_N$, where I_N is the $N \times N$ identity matrix. When zero noise is present, it can be seen that the rank of R_x is d, and that knowledge of the column space of A, sometimes called the signal subspace, can be used to derive useful information about the signal. Moreover, the signal subspace is spanned by the eigenvectors of R_x corresponding to the d principal (largest) eigenvalues, even for $\sigma_v > 0$. Most of the applications that have appeared in the literature are in the context of array processing of spatial data, in which case knowledge of the signal subspace can be used to determine the angles of arrival of multiple plane waves [1, 2, 3]. For temporal (time series) data, the signal subspace can be used to determine the frequencies of a sum of sinusoids [4]. Other applications for the temporal data case include signal enhancement. Knowledge of the signal subspace can be used to increase the SNR of x_n by projecting it onto the signal subspace. This has been used for speech enhancement [5, 6]. In some adaptive filtering applications, the signal to be filtered lies in a low-dimensional subspace and hence conventional adaptive filtering algorithms like LMS can have very slow convergence as a result of the large eigenvalue disparity present in the data covariance matrix [7]; in these cases signal subspace adaptive filtering can be useful [8, 9, 10, 11, 12, 13, 14]. If the matrix A, is changing with time, then one must reestimate the signal subspace repeatedly, to do this, the covariance matrix is typically estimated using an exponential time window

$$R_n = \sum_{k=N}^{n} \lambda^{n-k} x_k x_k^T \tag{1}$$

where n corresponds to the current sampling instant. If the d principal eigenvectors of R_n are computed at every sampling instant, the computational burden becomes excessive. As a result, algorithms with reduced complexity have been developed which are capable of computing less than exact eigenvector estimates (or an equivalent basis). Algorithms having $O(N^2d)$ complexity have been described [15, 16, 17, 18, 19, 20] however these are generally considered to have excessive computational requirements for online tracking applications. Another class of $O(Nd^2)$ algorithms based on the idea of subspace averaging have been described in [21, 22, 23, 24]. Karasolo's algorithm appears to provide a good trade-off between tracking performance and computational requirements and has been used as the benchmark for comparing fast subspace tracking algorithms [17]. Subspace averaging involves modeling the covariance matrix as

$$R_n \approx Q_n \Lambda_n Q_n^T + \hat{\sigma}_v I_N \tag{2}$$

where $Q_n \Lambda_n Q_n^T$ is an estimate of the eigendecomposition of APA^T at time n and $\hat{\sigma}_v$ is an estimate of the additive noise power. The complexity can be reduced to $O(Nd)$ by several other methods. DeGroat assumes that all of the signal subspace eigenvalues are equal leading to a spherical signal subspace as well as a spherical noise subspace, this leads to only having to solve a 2-dimensional eigenvalue problem, unlike the Karasolo algorithm which by assuming a spherical noise subspace, requires the solution of an $r+1$ dimensional eigenvalue problem. The spherical subspace assumption allows one to rotate the basis vectors to ultimately reduce the dimensionality of the problem [25]. Other $O(Nd)$ methods are described in [26, 27, 8, 9, 24, 28, 29]. Several stochastic gradient algorithms having $O(Nd^2)$ complexity have also been proposed [30, 31, 32]. This paper describes two new subspace tracking algorithms which have $O(Nd^2)$ complexity. The algorithms are based on the idea of projecting the matrix A onto the subspace spanned by the current eigenvector estimates and the current data vector. This approach would normally require $O(N^2d)$ complexity; however, we exploit the shift invariance property of R_n to reduce the computational complexity of $O(Nd^2)$ [33]. The algorithms are derived in Sections II and III. Section IV gives a proof of convergence, while Section V describes several experiments which compare the new algorithms with a number

of other algorithms having $O(Nd^2)$ and $O(Nd)$ complexity. The new algorithms are seen to perform better than the algorithms having similar complexity as well as those having less complexity (which is not unexpected given the difference in complexity). Finally, Section VI gives a summary of the results of this paper.

II. Subspace Projection

The proposed algorithms seek to obtain estimates of the signal subspace eigenvectors at time n, based a linear combination of the estimates at time $n-1$ and the data vector x_n. Define

$$\tilde{Q}_n = \begin{bmatrix} Q_{n-1} & x_n \end{bmatrix} \tag{3}$$

and assume that

$$Q_n = \begin{bmatrix} q_n^1 & q_n^2 & \cdots q_n^d \end{bmatrix} \tag{4}$$

where q_n^i is the $N \times 1$ eigenvector estimate associated with ith largest eigenvalue of R_n. Then q_n^1 can be obtained by maximizing the Rayleigh quotient

$$\mu\left(R_n, w_n^1\right) \equiv \frac{w_n^{1T} \tilde{Q}_n^T R_n \tilde{Q}_n w_n^1}{w_n^{1T} \tilde{Q}_n^T \tilde{Q}_n w_n^1} \tag{5}$$

over w_n^1, giving $q_n^1 = \tilde{Q}_n w_n^1$. The vector w_n^1 can be found by solving the symmetric generalized eigenvalue problem for the maximum eigenvalue and corresponding eigenvector,

$$\tilde{Q}_n^T R_n \tilde{Q}_n w_n^1 = \lambda_1(n) \tilde{Q}_n^T \tilde{Q}_n w_n^1 \tag{6}$$

The other eigenvector estimates $q_n^i, i = 2, \ldots, d$ can be obtained by solving

$$\tilde{Q}_n^T R_n \tilde{Q}_n W_n = \Lambda_n \tilde{Q}_n^T \tilde{Q}_n W_n \tag{7}$$

$$
\begin{array}{l}
\hline
\\
Initial Values \\
\quad Q_N = \left[\begin{array}{c} I_d \\ 0_{N-d\times d} \end{array} \right] \\
\quad R_N = x_N x_N^T \\
\\
for \quad n = N+1, N+2, \ldots \\
\quad R_n = \lambda R_{n-1} + x_n x_n^T \\
\quad \tilde{Q}_n(:, 1:d) = Q_{n-1} \\
\quad \tilde{Q}_n(:, d+1) = x_n \\
\quad U_n = R_n \tilde{Q}_n \\
\quad A = \tilde{Q}_n^T U_n \\
\quad B = \tilde{Q}_n^T \tilde{Q}_n \\
\quad solve\,(A - \Lambda_n B)\, W_n = 0_{N\times(d+1)} \\
\quad Q_n = \tilde{Q}_n W_n(:, 1:d) \\
\hline
\end{array}
$$

Table I: $O(N^2 d)$ complexity subspace projection algorithm (SP-1).

where $W_n = \left[\begin{array}{cccc} w_n^1 & w_n^2 & \cdots & w_n^{d+1} \end{array} \right]$ and $\Lambda_n = diag\,(\lambda_1(n), \lambda_2(n), \ldots, \lambda_{d+1}(n))$. Since typically $d \ll N$, solving the $(d+1)$-dimensional generalized eigenvalue problem (7) is relatively efficient (the well-known Karasolo algorithm also requires the solution of a $d+1$ dimensional eigenvalue problem [21]). It is interesting to note that the eigenvector estimates $Q_n = \tilde{Q}_n W_n(:, 1:d)$ will be mutually orthogonal. This can be seen by observing that for the symmetric eigenvalue problem, W_n satisfies [34]

$$
W_n(:, 1:d)^T \tilde{Q}_n^T \tilde{Q}_n W_n(:, 1:d) = I_d \tag{8}
$$

So if Q_1 has orthogonal columns, then all subsequent Q_n will have orthogonal columns. Table I lists the subspace projection (SP-1) algorithm.

The algorithm seeks the subspace of the column space of $\tilde{Q}_n$ which is closest to the signal subspace, the column space of A, hence the name "subspace projection". A second algorithm results by setting

$$\overline{Q}_n = \left[\begin{array}{ccc} Q_{n-1} & x_n & R_{n-1}x_n \end{array} \right] \tag{9}$$

and using $\overline{Q}_n$ in place of $\tilde{Q}_n$. This algorithm, referred to as SP-2, provides faster convergence than SP-1, since the columns of $\overline{Q}_n$ span a higher dimensional subspace than those of $\tilde{Q}_n$. For SP-2, the resulting symmetric generalized eigenvalue problem is $d+2$ dimensional compared to $d+1$ dimensional for SP-1.

III. Fast Subspace Projection

The SP-1 algorithm shown in Table I requires $O(N^2 d)$ operations per update. To reduce the complexity to $O(Nd^2)$ the matrix product $U_n = R_n \tilde{Q}_n$ is not explicitly computed but rather, U_n is recursively updated in time. First we note that

$$\begin{aligned} U_n &= \left(\lambda R_{n-1} + x_n x_n^T\right) \tilde{Q}_n \\ &= \lambda R_{n-1}\tilde{Q}_n + x_n x_n^T \tilde{Q}_n \end{aligned} \tag{10}$$

Using (3) leads to

$$U_n = \lambda \left[\begin{array}{cc} R_{n-1}Q_{n-1} & R_{n-1}x_n \end{array} \right] + x_n x_n^T \tilde{Q}_n \tag{11}$$

The quantity $\tilde{U}_n \equiv R_n Q_n$ can be updated in $O(Nd^2)$ operations by using the fact that $U_n = R_n \tilde{Q}_n$, and $Q_n = \tilde{Q}_n W_n(:, 1:d)$, hence

$$\tilde{U}_n = U_n W_n(:, 1:d) \tag{12}$$

Also by using the shift-invariance property of R_n, the quantity $g_{n-1} \equiv R_{n-1}x_n$ can be computed in $O(N)$ operations as seen in Appendix A. Therefore

$$U_n = \lambda \left[\begin{array}{cc} \tilde{U}_{n-1} & g_{n-1} \end{array} \right] + x_n x_n^T \tilde{Q}_n \tag{13}$$

Table II lists the fast version of algorithm SP-1 along with an operation count. The operations are counted on the basis of multiplies and accumulates (MAC's). Algorithms for computing the symmetric $(d+1)$ and $(d+2)$-dimensional generalized eigenvalue decomposition for algorithms SP-1 and SP-2, respectively in $O(d^3)$ operations can be found in Section 8.7.2 of [35].

As mentioned above, algorithm SP-2 utilizes an additional search direction, $R_{n-1}x_n$ to increase convergence speed. Define the matrix product $V_n = R_n\overline{Q}_n$. Once again, we have

$$\begin{aligned} V_n &= \left(\lambda R_{n-1} + x_n x_n^T\right)\overline{Q}_n \\ &= \lambda R_{n-1}\overline{Q}_n + x_n x_n^T \overline{Q}_n \end{aligned} \tag{14}$$

Using (9) leads to

$$U_n = \lambda \left[\begin{array}{ccc} R_{n-1}Q_{n-1} & R_{n-1}x_n & R_{n-1}^2 x_n \end{array} \right] + x_n x_n^T \overline{Q}_n \tag{15}$$

Let $\tilde{V}_n \equiv R_n Q_n$, as was the case with $\tilde{U}_n$, $\tilde{V}_n$ can be updated in $O(Nd^2)$ operations as

$$\tilde{V}_n = V_n W_n(:, 1:d) \tag{16}$$

Using the shift-invariance property of R_n, the quantity $h_{n-1} \equiv R_{n-1}^2 x_n$ can be computed in $O(N)$ operations as seen in Appendix B. Therefore

$$V_n = \lambda \left[\begin{array}{ccc} \tilde{U}_{n-1} & g_{n-1} & h_{n-1} \end{array} \right] + x_n x_n^T \overline{Q}_n \tag{17}$$

Table III lists the fast version of algorithm SP-2 along with an operation count. The fast updates of g_n and h_n use the shift invariance property of R_n. This property is also used in the derivation of fast Kalman filter adaptive filter algorithms [36]. However, unlike the fast Kalman filter algorithms, the g_n and h_n updates do not involve any matrix inversions and hence are very stable.

	MAC $count$
$Initial \quad Values:$	
$Q_N = \begin{bmatrix} I_d \\ 0_{(N-d)\times d} \end{bmatrix}$	
$\tilde{U}_N = x_N x_N^T Q_N$	
$g_N = x_N x_N^T x_{N+1}$	
$For \quad n = N+1, N+2, \ldots$	
$\tilde{Q}_n(:, 1:d) = Q_{n-1}$	
$\tilde{Q}_n(:, d+1) = x_n$	
$U_n = \lambda \begin{bmatrix} \tilde{U}_{n-1} & g_{n-1} \end{bmatrix} + x_n x_n^T \tilde{Q}_n$	$3Nd + 3N$
$A = \tilde{Q}_n^T U_n$	Nd^2
$B = \tilde{Q}_n^T \tilde{Q}_n$	Nd^2
$solve \quad (A - \Lambda_n B) W_n = 0_{N\times(d+1)}$	$O(d^3)$
$Q_n = \tilde{Q}_n W_n(:, 1:d)$	Nd^2
$\tilde{U}_n = U_n W_n(:, 1:d)$	Nd^2
$update \quad g_n \quad using \quad y_n = x_{n+1}$	$\underline{9N+4}$
	$4Nd^2 + 3Nd + 12N + 4 + O(d^3)$

Table II: Fast implementation of algorithm SP-1 showing number of operations.

	MAC $count$
$Initial \quad Values:$	
$Q_N = \begin{bmatrix} I_d \\ 0_{(N-d)\times d} \end{bmatrix}$	
$\tilde{U}_N = x_N x_N^T Q_N$	
$\tilde{V}_n = x_N x_N^T \tilde{U}_N$	
$g_N = x_N x_N^T x_{N+1}$	
$h_N = x_N x_N^T g_N$	
$For \quad n = N+1, N+2, \ldots$	
$\overline{Q}_n(:, 1:d) = Q_{n-1}$	
$\overline{Q}_n(:, d+1) = x_n$	
$\overline{Q}_n(:, d+2) = g_{n-1}$	
$V_n = \lambda \begin{bmatrix} \tilde{V}_{n-1} & g_{n-1} & h_{n-1} \end{bmatrix} + x_n x_n^T \overline{Q}_n$	$3Nd + 6N$
$A = \overline{Q}_n^T V_n$	Nd^2
$B = \overline{Q}_n^T \overline{Q}_n$	Nd^2
$solve \quad (A - \Lambda_n B) W_n = 0_{N\times(d+2)}$	$O(d^3)$
$Q_n = \tilde{Q}_n W_n(:, 1:d)$	Nd^2
$\tilde{V}_n = V_n W_n(:, 1:d)$	Nd^2
$update \quad g_n \quad using \quad y_n = x_{n+1}$	$9N + 4$
$update \quad h_n \quad using \quad y_n = x_{n+1}$	$\underline{32N + 30}$
	$4Nd^2 + 3Nd + 47N + 34 + O(d^3)$

Table III: Fast implementation of algorithm SP-2 showing number of operations.

IV. Algorithm Convergence

Here, we will prove the convergence of algorithm SP-1. We can simplify the convergence analysis by replacing the sample autocorrelation matrix R_n in algorithm SP-1 with $R_x = E\left[x_n x_n^T\right]$. Let[1]

$$q_n^i = \tilde{Q}_n W_n(:,i), \qquad i = 1, \ldots, d \tag{18}$$

be the estimate of eigenvector q^i generated by algorithm SP-1 under the assumption that $R_n = R_x$.

Lemma 1: The sequence $\mu\left(R_x, q_n^i\right), \quad i = 1, \ldots, d$ is monotonically increasing with n.

Proof: Assume that $\mu\left(R_x, q_n^i\right) < \mu\left(R_x, q_{n-1}^i\right)$, then $W_n(:,i)$ was not chosen to maximize the Rayleigh quotient since[2] $W_n(:,i) = e_i$ would have given $\mu\left(R_x, q_n^i\right) = \mu\left(R_x, q_{n-1}^i\right)$. □

Lemma 2: The Rayleigh quotient gradient evaluated at the updated eigenvector estimate

$$\nabla\mu\left(R_x, q_n^i\right) = R_x q_n^i - \mu\left(R_x, q_n^i\right) q_n^i \tag{19}$$

is orthogonal to the columns of $\tilde{Q}_n$.

Proof: Substituting (18) into (19), using (3) and premultiplying by $\tilde{Q}_n^T$ gives

$$\tilde{Q}_n^T \nabla\mu\left(R_x, q_n^i\right) = \tilde{Q}_n^T R_x \tilde{Q}_n W(:,i) - \mu\left(R_x, q_n^i\right) \tilde{Q}_n^T \tilde{Q}_n W(:,i) \tag{20}$$

Clearly, the right hand side of (20) is zero since $\mu\left(R_x, q_n^i\right)$ is the generalized eigenvalue solving (7). □

Theorem: Assume that R_x, has eigenvectors $q^1, q^2, \ldots, q^N$, and associated eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_N$. Also assume that $\|\tilde{Q}_n\|_2 > 0$. Then $q_n^i \to q^i, \quad i = 1, \ldots, d$.

Proof: Consider the convergence of q_n^1. Since by *Lemma 1*, $\mu\left(R_x, q_n^1\right)$ is monotonically increasing and since the Rayleigh quotient associated with q_n^1 is bounded by λ_1, q_n^1 must

[1] $W_n(:,i)$ corresponds to the ith column of W_n

[2] $e_i = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T$ with the 1 in the ith position

converge to an accumulation point q^*. Hence we have,

$$\begin{aligned} \|\tilde{Q}_n^T \nabla\mu\left(R_x, q^*\right)\|_2 &= \|\left(\nabla\mu\left(R_x, q^*\right) - \nabla\mu\left(R_x, q_n^1\right)\right)^T \tilde{Q}_n\|_2 \\ &\leq \|\left(\nabla\mu\left(R_x, q^*\right) - \nabla\mu\left(R_x, q_n^1\right)\right)\|\|\tilde{Q}_n\|_2 \end{aligned} \tag{21}$$

The second term in the right-hand side of the first line of (21) is zero by *Lemma 2* while the inequality is due to a well-known property of matrix norms [37]. There is no loss of generality in taking $\|\tilde{Q}_n\| = 1$ since the norm of $\tilde{Q}_n$ can always be absorbed into $W(:,1)$. Since the Rayleigh quotient is continuous, there exists a constant $\epsilon > 0$ and an integer n_o such that for all $n > n_o$,

$$\|\tilde{Q}_n^T \nabla\mu\left(R_x, q_n^*\right)\|_2 \leq \|\mu\left(R_x, q_n^*\right) - \mu\left(R_x, q_n^1\right)\|_2 < \epsilon \tag{22}$$

Since ϵ can be made arbitrarily small, and $\|\tilde{Q}_n\|_2 > 0$ by assumption, q^* must be a critical point of the Rayleigh quotient. To show that $q^* = q^1$, we note that if $q^* \neq q^1$, then from the projection interpretation of SP-1, it follows that for all n, $q^1 \perp x_n$, otherwise, x_n would be able to drive q_n^1 in a direction closer to q^1 compared to q_{n-1}^1. But this would then imply that q^1 is not an eigenvector of R_x which is a contradiction. Finally, since $q_n^1 \rightarrow q^1$, it follows that $q_n^i \rightarrow q^i, \quad i = 2, \ldots d$ in lieu of the orthogonality of the estimated eigenvectors (see the comments associated with equation (8)). □

The above theorem proves that for stationary data, using a forgetting factor of $\lambda = 1$, SP-1 will converge asymptotically since one could take as the starting point of SP-1, a large value of n for which $R_n \approx R_x$, in which case, the eigenvector estimates q_n^i would converge to quantities arbitrarily close to q^i.

V. Tracking Experiments and Discussion

The following algorithms were implemented in Matlab

Algorithm	Complexity	Domain
SP-1	$O(Nd^2)$	time
SP-2	$O(Nd^2)$	time
KARA [21]	$O(Nd^2)$	time and/or space
BISVD [24]	$O(Nd^2)$	time and/or space
KFST [8]	$O(Nd)$	time and/or space
PAST [27]	$O(Nd)$	time and/or space
ROSA [25]	$O(Nd)$	time and/or space
PC [26]	$O(Nd)$	time and/or space
PROT [28]	$O(Nd)$	time and/or space

Algorithm KARA is the subspace averaging algorithm of Karasolo [21]. In most of the previously cited literature, this algorithm appears to be the standard by which fast subspace algorithms are measured. KFST is a subspace tracker due to Rabideau that has performance identical to the Karasolo algorithm [8]. ROSA is the subspace averaging algorithm of DeGroat which uses both a noise and a signal spherical subspace leading to an efficient $O(Nd)$ algorithm [25]. BISVD is an algorithm due to Strobach which is based on bi-iteration [24]. PAST is an algorithm by Yang which recursively minimizes a quadratic performance measure [27]. And PC and PROT (Proteus-2) are algorithms developed by Champagne based on perturbation theory [26, 28]. The stochastic gradient algorithms mentioned earlier were not tested since these have been found to have inferior performance relative to the other algorithms [17]. The first experiment involved tracking a step change in the frequencies of two sinusoids in additive noise. The following signal was generated

$$x(n) = \begin{cases} cos(0.3\pi n) + cos(0.7\pi n + 0.35\pi) + v(n), & n = 1, \ldots, 999 \\ cos(0.6\pi n) + cos(0.8\pi n + 0.35\pi) + v(n), & n = 1000, \ldots, 2000 \end{cases} \tag{23}$$

where $v(n)$ was Gaussian white noise giving an SNR of 10 dB ($\sigma_v = 0.1$). For each algorithm tested, the error between the estimated eigenvectors and the actual eigenvectors of R_n was measured as

$$\epsilon(n) = \|Q_n \left(Q_n^T Q_n\right)^{-1} Q_n^T - Q_n^o \left(Q_n^{oT} Q_n^o\right)^{-1} Q_n^{oT}\|_F \tag{24}$$

where $\| \quad \|_F$ is the Forbeneus norm, Q_n is the matrix whose columns are the eigenvector estimates and Q_n^o is the matrix containing the actual eigenvectors of R_n as computed with Matlab routine "eig". The error measure $\epsilon(n)$ represents the distance between two subspaces [35]. The algorithm parameters were set to $N = 50, d = 4$, and $\lambda = 0.99$. Figures 1 - 3 show the resulting error measures comparing the new SP-1 and SP-2 algorithms with the other algorithms. In each case algorithms SP-1 and SP-2 are seen to out-perform the other algorithms. Algorithm SP-2, has a significantly lower noise floor and tracks the step change in frequencies at $n = 1000$ much faster than the other algorithms. We note that SP-2 converges considerably faster than SP-1, though SP-1 still tracks the step change faster than any of the other algorithms. As mentioned previously the improvement shown by SP-2 over SP-1 results from the additional search direction used. Moreover, this added search direction $R_{n-1}x_n$ is a "good" choice for a search direction since it will tend to point in the direction of the eigenvector corresponding to the maximum eigenvalue of R_n. This can be seen by viewing $R_{n-1}x_n$ as one iteration of the power method used to find the eigenvector associated with the maximum eigenvalue [35]. Since there does not appear to be a consensus on which error measure to use when comparing subspace tracking algorithms, the experiment was repeated using the following error measure:

$$\delta(n) = \|Q_n \left(Q_n^T Q_n\right)^{-1} Q_n^T - Q_n^\dagger \left(Q_n^{\dagger T} Q_n^\dagger\right)^{-1} Q_n^{\dagger T}\|_F \tag{25}$$

where $Q_n^\dagger$ is the theoretical signal subspace for the sinusoidal signal used in the experiment [4]. Under this error measure, algorithms SP-1 and SP-2 behaved very similarly, while the three algorithms KARA, KFST, and BISVD each had virtually identical performance, and the algorithm pair PROT and PC had similar performance. This made it possible to reduce

the total number of error curves as seen in Figures 4-5. Algorithms SP-1 and SP-2 are again seen to outperform the other algorithms.

We have been interested in applying subspace tracking to speech enhancement. The idea behind this is that by projecting a noisy low-rank signal onto the signal subspace, the SNR of the noisy signal can be improved [5, 6]. A drawback to subspace speech enhancement is the extensive amount of computation required to compute the signal subspace basis. We have observed that better quality speech results by computing the signal subspace at every sampling instant rather than on a block by block basis. Hence, subspace tracking algorithms which update the signal subspace at every sample number are useful. The signal subspace was tracked for all nine algorithms for an actual speech segment. The speech utterance "nine two one two" was sampled at 8 kHz, white Gaussian noise was added to give an SNR of 10 dB. The algorithm parameters used were $N = 50, d = 6$, and $\lambda = 0.999$. The two error criteria $\epsilon(n)$ and $\delta(n)$ are plotted in Figure 6. Since the theoretical signal subspace of speech is unknown, it was estimated by computing the eigendecomposition of the noise-free sample covariance matrix. Again algorithms SP-2, and SP-1 have the best performance, while the other algorithms appear to have relatively poor performance.

The reduced performance of KARA, KFST, ROSA, and BISVD can be explained by noting that these algorithms are based on the idea of subspace averaging where the sample covariance matrix is estimated as in (2), repeated here

$$R_n \approx Q_n \Lambda_n Q_n^T + \hat{\sigma}_v I_n \tag{26}$$

Speech tends to have relatively high dimensional signal subspace, compared to say, sinusoids, Fig. 7 shows a plot of the mean eigenvalues obtained from the speech signal used in the above experiment. There appear to be at least 12 eigenvalues which clearly lie above the noise floor. However in subspace-based speech enhancement, it is desirable to keep the signal subspace dimension low while still attempting to span as much of the signal subspace as possible, otherwise excess noise will project onto the signal subspace. We have found that a relatively

low number of eigenvectors (i.e. 6) tends to produce good quality enhanced speech. However $d = 6$ will produce poor results with subspace averaging algorithms because the model in (26) is violated. Algorithms SP-1 and SP-2 do not use subspace averaging and hence are not subject to these modeling errors. Increasing the signal subspace dimension in the subspace averaging algorithms did not improve their performance. Some of the algorithms against which SP-1 and SP-2 were compared can be run in a rank-adaptive mode and this may improve their performance whan tracking speech signal subspaces.

VI. Summary

Two new algorithms for tracking the signal subspace of low-rank temporal signals were described. The algorithms are based on estimating the signal subspace based on linear combinations of the current eigenvector estimates and the current data vector, hence, the algorithms are called subspace projection algorithms. The computational complexity of the algorithms is reduced to $O(Nd^2)$ by exploiting the shift-invariance property of temporal data covariance matrices. A proof of convergence was given, and in experiments, the algorithms were shown to converge more rapidly and provide more accurate tracking then 7 other previously published algorithms. The new algorithms were also found to outperform the other algorithms in tracking the signal subspace associated with a speech signal. Of these 7 other algorithms, 2 had similar ($O(Nd^2)$) complexity while the remainder had $O(Nd)$ complexity, hence one would expect the new algorithms to have faster convergence than the lower complexity algorithms. Also, the other algorithms which were tested have more general applicability since they do not require that the data have the shift invariance property. The new algorithms have applications in low-rank adaptive filtering, which require temporal signal subspace tracking [9, 12, 11, 13, 14] and speech enhancement [5, 6].

A Appendix: $O(N)$ computation of $g_n = R_n y_n$

For prewindowed data, an algorithm for carrying out this matrix vector product in $O(N)$ operations was described in [38]. Here, we present a version for covariance-type windowing, which has been found to be more appropriate for subspace tracking involving large N. We note that the data begins at $n = 1$; also we assume that

$$\begin{aligned} \overline{x}_n &= \left[\begin{array}{cc} x(n) & x_{n-1}^T \end{array} \right]^T \\ &= \left[\begin{array}{cc} x_n^T & x(n-N) \end{array} \right]^T \end{aligned} \tag{27}$$

$$\begin{aligned} \overline{y}_n &= \left[\begin{array}{cc} y(n) & y_{n-1}^T \end{array} \right]^T \\ &= \left[\begin{array}{cc} y_n^T & y(n-N) \end{array} \right]^T \end{aligned} \tag{28}$$

with

$$x_n = \left[\begin{array}{cccc} x(n) & x(n-1) & \cdots & x(n-N+1) \end{array} \right]^T$$

and

$$y_n = \left[\begin{array}{cccc} y(n) & y(n-1) & \cdots & y(n-N+1) \end{array} \right]^T$$

We begin by defining the $(N+1) \times 1$ vector

$$\overline{g}_n \equiv \overline{R}_n \overline{y}_n$$

where

$$\overline{R}_n \equiv \sum_{k=N+1}^{n} \lambda^{n-k} \overline{x}_k \overline{x}_k^T, \qquad n \geq N+1 \tag{29}$$

It is straight-forward to verify that

$$\overline{R}_n = \left[\begin{array}{cc} R_n^1 & r_n \\ r_n^T & r(n-N) \end{array} \right] = \left[\begin{array}{cc} r(n) & \tilde{r}_n \\ \tilde{r}_n^T & R_{n-1} \end{array} \right] \tag{30}$$

where

$$R_n = \sum_{k=N}^{n} \lambda^{n-k} x_k x_k^T, \qquad n \geq N \tag{31}$$

$$R_n^1 = \sum_{k=N+1}^{n} \lambda^{n-k} x_k x_k^T, \qquad n \geq N+1 \tag{32}$$

$$r_n = \sum_{k=N+1}^{n} \lambda^{n-k} x_k x(k-N) \tag{33}$$

$$\tilde{r}_n = \sum_{k=N+1}^{n} \lambda^{n-k} x_{k-1} x(k) \tag{34}$$

$$r(n) = \sum_{k=N+1}^{n} \lambda^{n-k} x(k)^2 \tag{35}$$

Using (29), (27), (28), and (30) we have

$$\overline{g}_n = \begin{bmatrix} g_n^1 + r_n y(n-N) \\ r_n^T y_n + r(n-N) y(n-N) \end{bmatrix} \tag{36}$$

$$= \begin{bmatrix} r(n) y(n) + \tilde{r}_n^T y_{n-1} \\ \tilde{r}_n y(n) + g_{n-1} \end{bmatrix} \tag{37}$$

where $g_n^1 \equiv R_n^1 y_n$ and $g_n \equiv R_n y_n$. Observe that (37) can be used to compute $\overline{g}_n$. Using (36) gives

$$g_n^1 = [\overline{g}_n]_{1,N} - r_n y(n-N) \tag{38}$$

and

$$g_n = g_n^1 + \lambda^{n-N} x_N x_N^T y_n, \qquad n \geq N+1 \tag{39}$$

The only other quantities that need updating are the vectors in (30) which can be efficiently updated as

$$r_n = \lambda r_{n-1} + x(n-N) x_n, \qquad n \geq N+1 \tag{40}$$

$$\tilde{r}_n = \lambda \tilde{r}_{n-1} + x(n) x_{n-1}, \qquad n \geq N+1 \tag{41}$$

	MAC count
Initial Values :	
$g_N = x_N(x_N^T y_n)$	
$r_N = 0_{N\times 1}$	
$\tilde{r}_N = 0_{N\times 1}$	
$r(N) = 0$	
For $n = N+1, N+2, \ldots$	
$r_n = \lambda r_{n-1} + x(n-N)x_n$	$2N$
$\tilde{r}_n = \lambda \tilde{r}_{n-1} + x(n)x_{n-1}$	$2N$
$r(n) = \lambda r(n-1) + x(n)^2$	2
$[\overline{g}_b]_1 = r(n)y(n) + \tilde{r}_n^T y_{n-1}$	$N+1$
$[\overline{g}_b]_{2,N+1} = \tilde{r}_n y(n) + g_{n-1}$	N
$g_n^1 = [\overline{g}_b]_{1,N} - r_n y(n-N)$	N
$g_n = g_n^1 + \lambda^{n-N} x_N x_N^T y_n$	$\underline{2N+1}$
	$9N+4$

Table IV: Algorithm for updating $g_n = R_n y_n$ showing number of operations.

$$r(n) = \lambda r(n-1) + x(n)^2, \qquad n \geq N+1 \tag{42}$$

The complete algorithm listing is found in Table IV.

B Appendix: $O(N)$ computation of $h_n \equiv (R_n)^2 y_n$

Derivation of the algorithm for computing h_n follows along the same lines as that of g_n, but is a bit more tedious due to the extra terms resulting from the squaring of R_n. The notation $(C)^2 \equiv CC$ will be used to denote the square of a matrix. We note that

$$\begin{bmatrix} R_n^1 & r_n \\ r_n^T & r(n-N) \end{bmatrix} \begin{bmatrix} R_n^1 & r_n \\ r_n^T & r(n-N) \end{bmatrix} = \begin{bmatrix} (R_n^1)^2 + r_n r_n^T & R_n^1 r_n + r_n r(n-N) \\ r_n^T R_n^1 + r(n-N) r_n^T & r_n^T r_n + r(n-N)^2 \end{bmatrix} \tag{43}$$

Hence, letting $\overline{h}_n \equiv \left(\overline{R}_n\right)^2 \overline{y}_n$ and using (43) and (28) gives

$$\overline{h}_n = \begin{bmatrix} h_n^1 + r_n r_n^T y_n + e_n^1 y(n-N) + r_n r(n-N) y(n-N) \\ r_n^T g_n^1 + r(n-N) r_n^T y_n + r_n^T r_n y(n-N) + r(n-N)^2 y(n-N) \end{bmatrix} \tag{44}$$

where

$$h_n^1 = \left(R_n^1\right)^2 y_n \tag{45}$$

$$e_n^1 = R_n^1 r_n \tag{46}$$

We can also write

$$\begin{bmatrix} r(n) & \tilde{r}_n^T \\ \tilde{r}_n & R_{n-1} \end{bmatrix} \begin{bmatrix} r(n) & \tilde{r}_n^T \\ \tilde{r}_n & R_{n-1} \end{bmatrix} = \begin{bmatrix} r(n)^2 + \tilde{r}_n^T \tilde{r}_n & r(n)\tilde{r}_n^T + \tilde{r}_n^T R_{n-1} \\ \tilde{r}_n r(n) + R_{n-1}\tilde{r}_n & \tilde{r}_n \tilde{r}_n^T + \left(R_{n-1}\right)^2 \end{bmatrix} \tag{47}$$

Using (28) we get

$$\overline{h}_n = \begin{bmatrix} r(n)^2 y(n) + \tilde{r}_n^T \tilde{r}_n y(n) + r(n)\tilde{r}_n^T y_{n-1} + \tilde{r}_n^T g_{n-1} \\ \tilde{r}_n r(n) y(n) + \tilde{e}_n y(n) + \tilde{r}_n \tilde{r}_n^T y_{n-1} + h_{n-1} \end{bmatrix} \tag{48}$$

where

$$\tilde{e}_n = R_{n-1}\tilde{r}_n \tag{49}$$

and

$$e_n \equiv R_n \tilde{r}_n = \lambda \tilde{e}_n + x_n x_n^T \tilde{r}_n \tag{50}$$

Now the quantities e_n and e_n^1 can be updated as

$$\begin{aligned} e_n &= \left(\lambda R_{n-1} + x_n x_n^T\right)\left(\lambda \tilde{r}_{n-1} + x(n) x_{n-1}\right) \\ &= \lambda^2 e_{n-1} + \lambda x_n x_n^T \tilde{r}_{n-1} + \lambda \tilde{g}_{n-1} x(n) + x_n \left(x_n^T x_{n-1}\right) x(n) \end{aligned} \tag{51}$$

and

$$e_n^1 = \left(\lambda R_{n-1}^1 + x_n x_n^T\right)\left(\lambda r_{n-1} + x(n-N)x_n\right) \quad (52)$$

$$= \lambda^2 e_{n-1}^1 + \lambda x_n x_n^T r_{n-1} + \lambda g_{n-1}^1 x(n-N) + x_n\left(x_n^T x_n\right)x(n-N) \quad (53)$$

where $\tilde{g}_n \equiv R_n x_n$. Equation (48) can be used to compute $\overline{h}_n$, then from (44) we have

$$h_n^1 = \left[\overline{h}_n\right]_{1,N} - r_n r_n^T y_n - e_n^1 y(n-N) - r_n r(n-N) y(n-N) \quad (54)$$

Since $R_n^2 = \left(R_n^1 + \lambda^{n-N} x_N x_N^T\right)^2$, it follows that

$$h_n = h_n^1 + \lambda^{n-N} x_N x_N^T g_n^1 + \lambda^{n-N} f_n x_N^T y_n + \lambda^{2(n-N)} x_N\left(x_N^T x_N\right) x_N^T y_n \quad (55)$$

The quantity $f_n \equiv R_n^1 x_N$ can be efficiently updated as

$$f_n = \lambda f_{n-1} + x_n\left(x_n^T x_N\right), \qquad n \geq N+1 \quad (56)$$

Table V lists the complete algorithm for updating h_n.

	MAC $count$
$Initial \quad Values:$	
$e_N = 0_{N\times 1}$	
$e_N^1 = 0_{N\times 1}$	
$f_N = 0_{N\times 1}$	
$For \quad n = N+1, N+2, \ldots$	
$e_n = \lambda^2 e_{n-1} + \lambda x_n x_n^T \tilde{r}_{n-1} + \lambda \tilde{g}_{n-1} x(n) + x_n \left(x_n^T x_{n-1}\right) x(n)$	$5N+8$
$e_n^1 = \lambda^2 e_{n-1}^1 + \lambda x_n x_n^T r_{n-1} + \lambda g_{n-1}^1 x(n-N) + x_n \left(x_n^T x_n\right) x(n-N)$	$5N+8$
$\tilde{e}_n = \left(e_n - x_n x_n^T \tilde{r}_n\right)/\lambda$	$3N$
$\left[\overline{h}_n\right]_1 = r(n)^2 y(n) + \tilde{r}_n^T \tilde{r}_n y(n) + r(n)\tilde{r}_n^T y_{n-1} + \tilde{r}_n^T g_{n-1}$	$2N+4$
$\left[\overline{h}_n\right]_{2,N} = \tilde{r}_n r(n) y(n) + \tilde{e}_n y(n) + \tilde{r}_n \tilde{r}_n^T y_{n-1} + h_{n-1}$	$3N+2$
$h_n^1 = \left[\overline{h}_n\right]_{1,N} - r_n r_n^T y_n - e_n^1 y(n-N) - r_n r(n-N) y(n-N)$	$4N+1$
$f_n = \lambda f_{n-1} + x_n \left(x_n^T x_N\right)$	$3N$
$h_n = h_n^1 + \lambda^{n-N} x_N x_N^T g_n^1 + \lambda^{n-N} f_n x_N^T y_n + \lambda^{2(n-N)} x_N \left(x_N^T x_N\right) x_N^T y_n$	$4N+7$
$\tilde{g}_n = \lambda g_{n-1} + x_n x_n^T x_n$	$\underline{3N}$
	$32N+30$

Table V: Algorithm for updating $h_n = (R_n)^2 y_n$ showing number of operations. The operation count does not reflect the computation of quantities already computed in Table I.

References

[1] D. H. Johnson and D. E. Dudgeon, *Array Signal Processing*. Prentice Hall, 1993.

[2] N. Owsley, "Sonar array processing," in *Array Signal Processing* (S. Haykin, ed.), Prentice Hall, 1985.

[3] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*. Wiley-Interscience, 1980.

[4] S. M. Kay, *Modern Spectral Estimation*. Prentice Hall, 1988.

[5] Y. Ephraim and H. L. Van Trees, "A signal subspace approach for speech enhancement," *IEEE Trans. on Speech and Audio Processing*, vol. 3, pp. 251–266, July 1995.

[6] J. Huang and Y. Zhao, "An energy-constrained signal subspace method for speech enhancement and recognition in white and colored noises," *Speech Communication*, vol. 26, pp. 165–181, 1998.

[7] S. Haykin, *Adaptive Filter Theory*. Prentice Hall, 1991.

[8] D. J. Rabideau, "Fast, rank adaptive subspace tracking and applications," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. 44, pp. 2229–2244, September 1996.

[9] P. Strobach, "Low-rank adaptive filters," *IEEE Transactions on Signal Processing*, vol. 44, pp. 2932–2947, Dec 1996.

[10] P. Strobach, "Square Hankel SVD subspace tracking algorithms," *Signal Processing*, vol. 57, pp. 1–18, Feb. 1997.

[11] J. S. Goldstein and I. S. Reed, "Reduced-rank adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 45, pp. 492–496, Feb 1997.

[12] D. Linebarger, B. Raghothaman, D. Begusic, E. Dowling, R. DeGroat, and S. Oh, "Low rank transform domain adaptive filtering," in *Proceedings of the 1997 31st Asilomar Conf. on Signals, Systems and Computers*, (Monterey, CA), pp. 123–127, 1997.

[13] S. Hosur, A. H. Tewfik, and D. Boley, "ULV and generalized ULV subspace tracking adaptive algorithms," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1282–1297, May 1998.

[14] M. L. Honig, "Adaptive linear interference suppression for packet DS-CDMA," *European Transactions on Telecommunications*, vol. 9, pp. 173–181, Mar-Apr 1998.

[15] C. Yeh, "Simple computation of projection matrix for bearing estimations," *IEE Proceedings, Part F*, vol. 134, pp. 146–150, April 1987.

[16] D. Fuhrmann, "An algorithm for subspace computation with applications in signal processing," *SIAM Jour. Matrix Anal. Appl.*, vol. 9, pp. 213–220, April 1988.

[17] P. Comon and G. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proc. IEEE*, vol. 78, pp. 1327–1343, Aug. 1990.

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

[19] G. Xu and T. Kailath, "Fast subspace decomposition," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. SP-42, pp. 539–551, March 1994.

[20] A. Eriksson and P. Stoica and T. Söderström, "On-line subspace algorithms for tracking moving sources," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. SP-42, pp. 2319–2330, Sept. 1994.

[21] I. Karasolo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. 34, pp. 8–12, Feb. 1986.

[22] E. M. Dowling, L. P. Ammann, and R. D. DeGroat, "A TQR-iteration based adaptive SVD for real time angle and frequency tracking," *IEEE Transactions on Signal Processing*, vol. 42, pp. 914–926, April 1994.

[23] Z. Fu and E. M. Dowling, "Conjugate gradient eigenstructure tracking for adaptive spectral estimation," *IEEE Transactions on Signal Processing*, vol. 43, pp. 1151–1160, May 1995.

[24] P. Strobach, "Bi-iteration SVD subspace tracking algorithms," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. 45, pp. 1222–1240, May 1997.

[25] R. DeGroat, "Noniterative subspace tracking," *IEEE Trans. Signal Processing*, vol. SP-40, pp. 571–577, March 1992.

[26] B. Champagne, "Adaptive eigendecomposition of data covariance matrices based on first-order perturbations," *IEEE Transactions on Signal Processing*, vol. 1994, pp. 2758–2771, October 1994.

[27] B.Yang, "Projection approximation subspace tracking," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. 43, pp. 95–107, January 1995.

[28] B. Champagne and Q. Liu, "Plane rotation-based evd updating schemes for efficient subspace tracking," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. 46, pp. 1886–1900, July 1998.

[29] E. C. Real, D. W. Tufts, and J. W. Cooley, "Two algorithms for fast approximate subspace tracking," *IEEE Transactions on Signal Processing*, vol. 47, pp. 1936–1945, July 1999.

[30] N. Owsley, "Adaptive data orthogonalization," in *Proc. ICASSP*, pp. 109–112, 1978.

[31] P. Thompson, "An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise," in *Proc. 13th Asilomar Conf. Circuits, Sys. , Comput.*, (Monterrey), pp. 529–533, Nov. 1979.

[32] J. Yang and M. Kaveh, "Adaptive eigensubspace algorithms for direction or frequency estimation and tracking," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. ASSP-36, pp. 241–251, Feb. 1988.

[33] C. Davila and M. Mobin, "Efficient tracking of time-varying signal subspaces," in *Proc. 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, (San Francisco), pp. 133–136, March 1992.

[34] G. Strang, *Linear Algebra and Its Applications*. Academic Press, 1980.

[35] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1989.

[36] L. Ljung, M. Morf, and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Int. J. Contr.*, vol. 27, pp. 1–19, 1978.

[37] G. W. Stewart, *Introduction to Matrix Computations*. Academic Press, 1973.

[38] C. E. Davila, "Line search algorithms for adaptive filtering," *IEEE Trans. Signal Proc.*, vol. SP-41, pp. 2490–2494, July 1993.

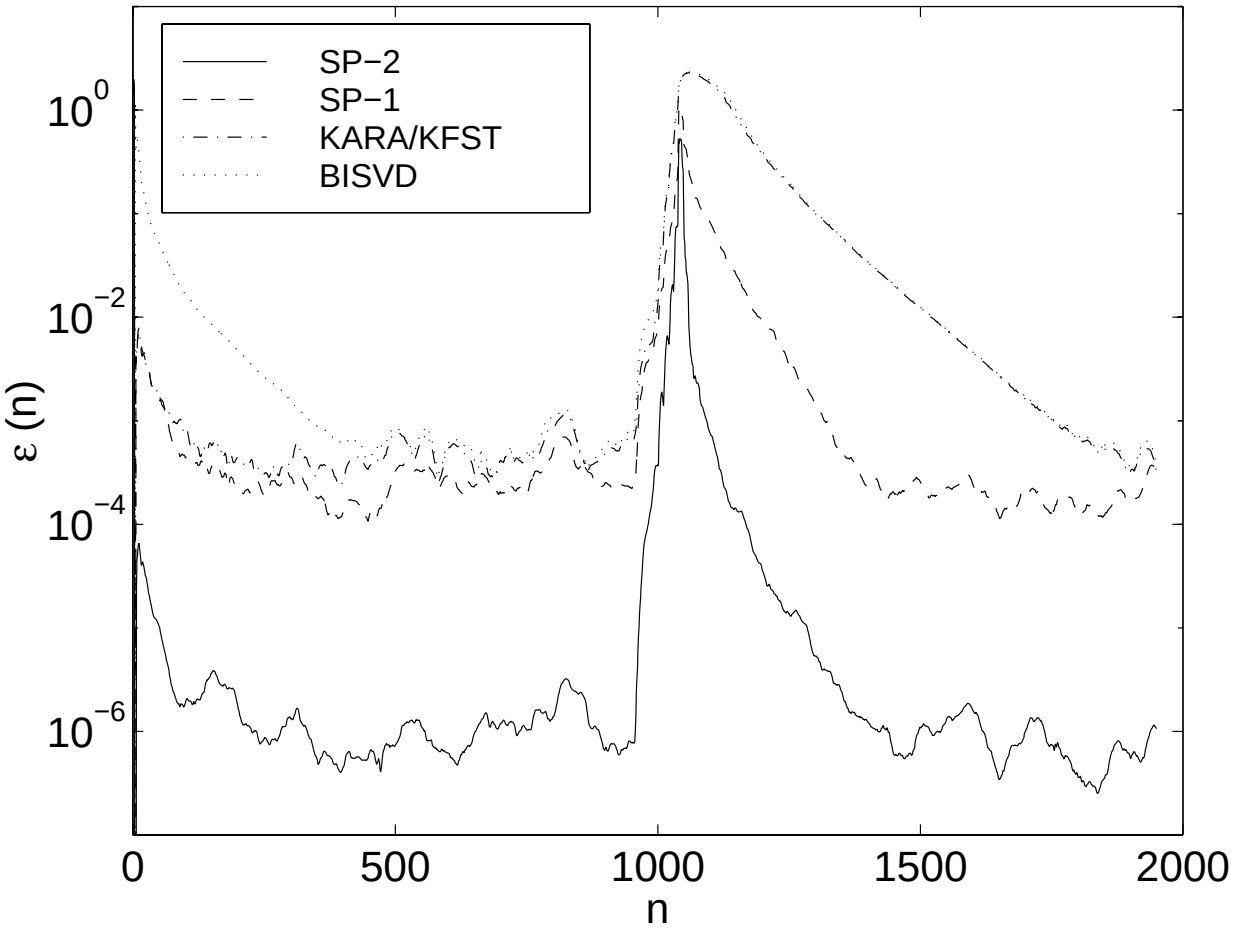

Figure 1: Subspace tracking experiments for noisy sinusoids. This plot compares SP-1 and SP-2 against algorithms KAR, KFST, and BISVD. The increase in the error at $n = 1000$ corresponds to a step change in the sinusoidal frequencies.

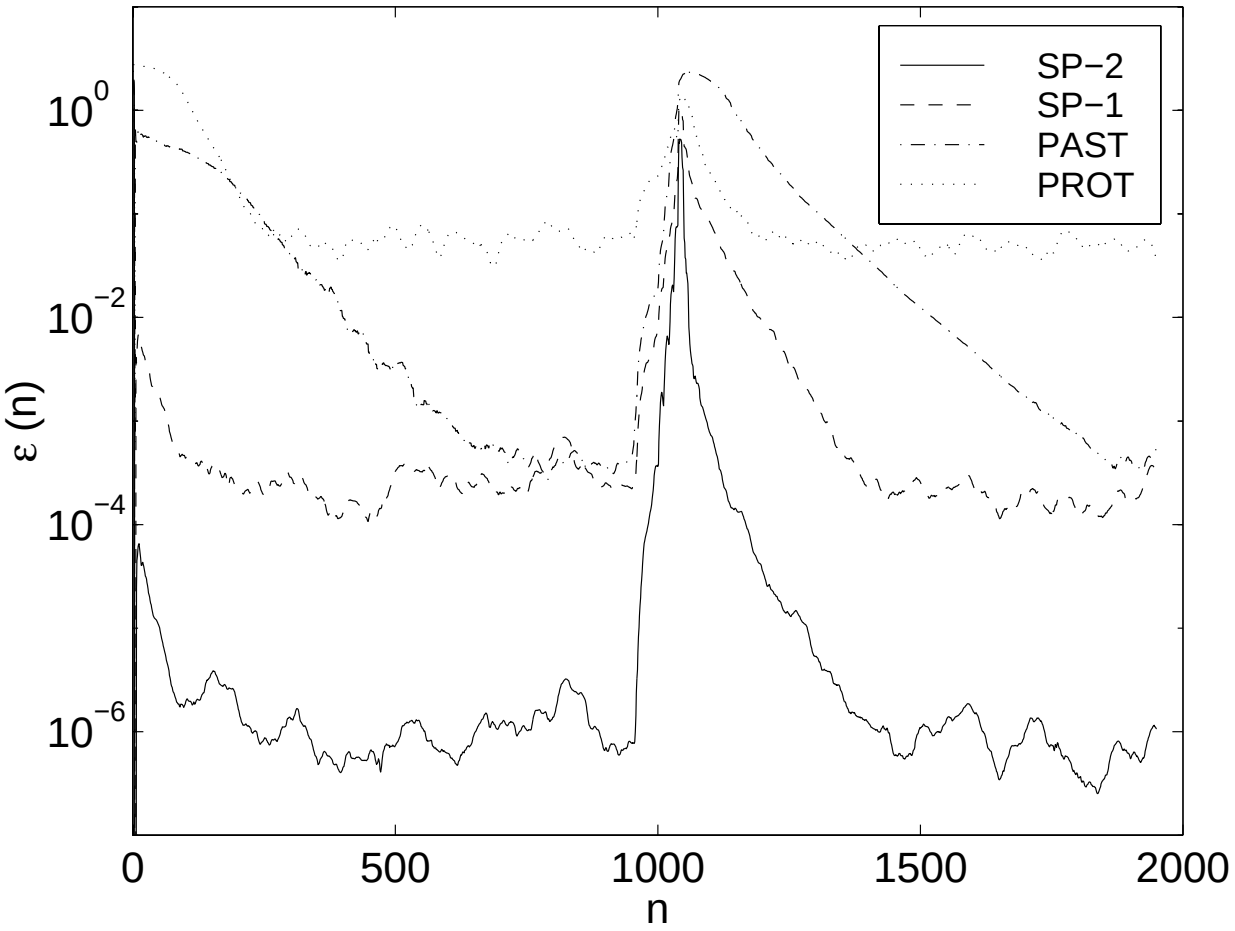

Figure 2: Subspace tracking experiments for noisy sinusoids. This plot compares SP-1 and SP-2 against algorithms PAST, and PROT.

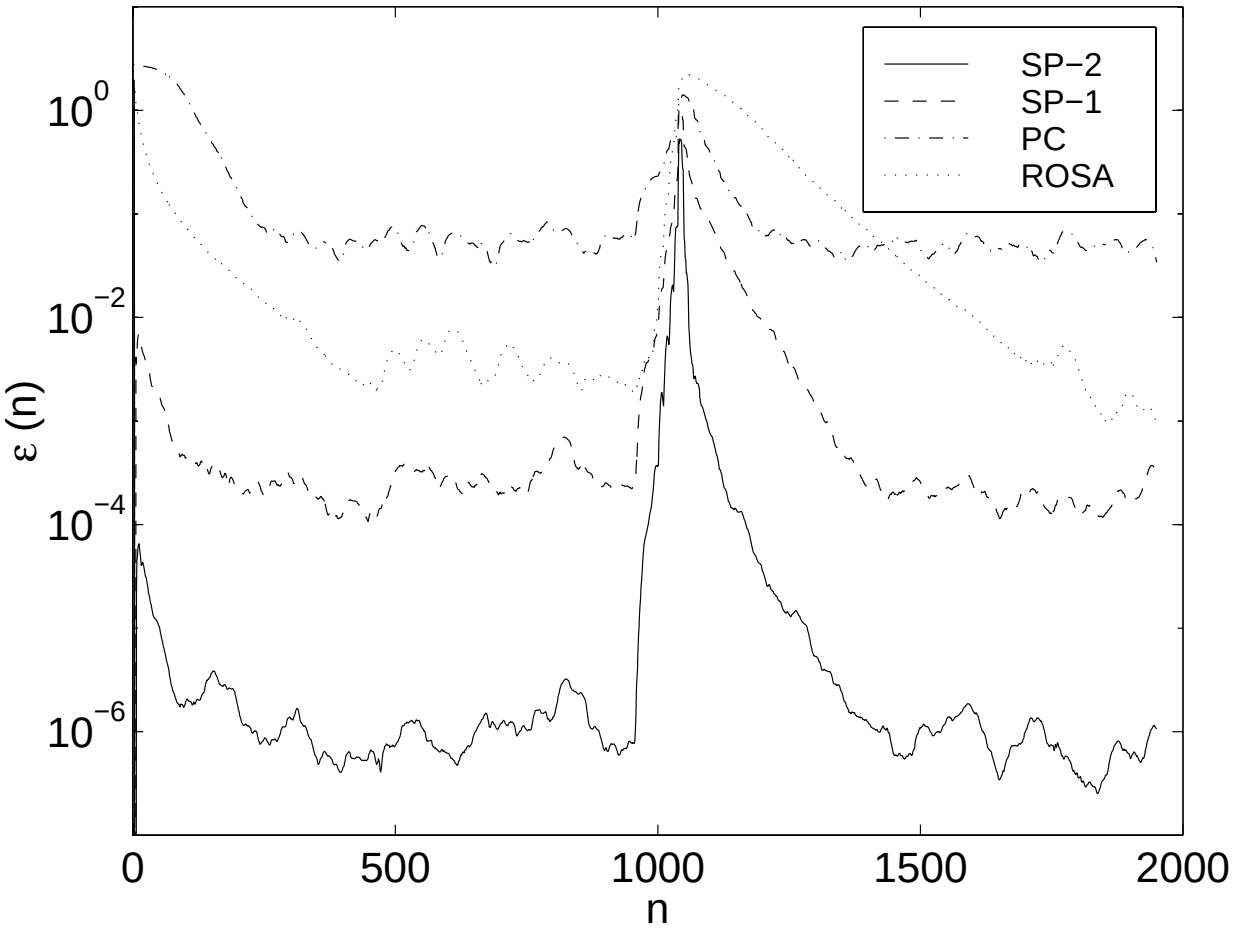

Figure 3: Subspace tracking experiments for noisy sinusoids. This plot compares SP-1 and SP-2 against algorithms PC, and ROSA.

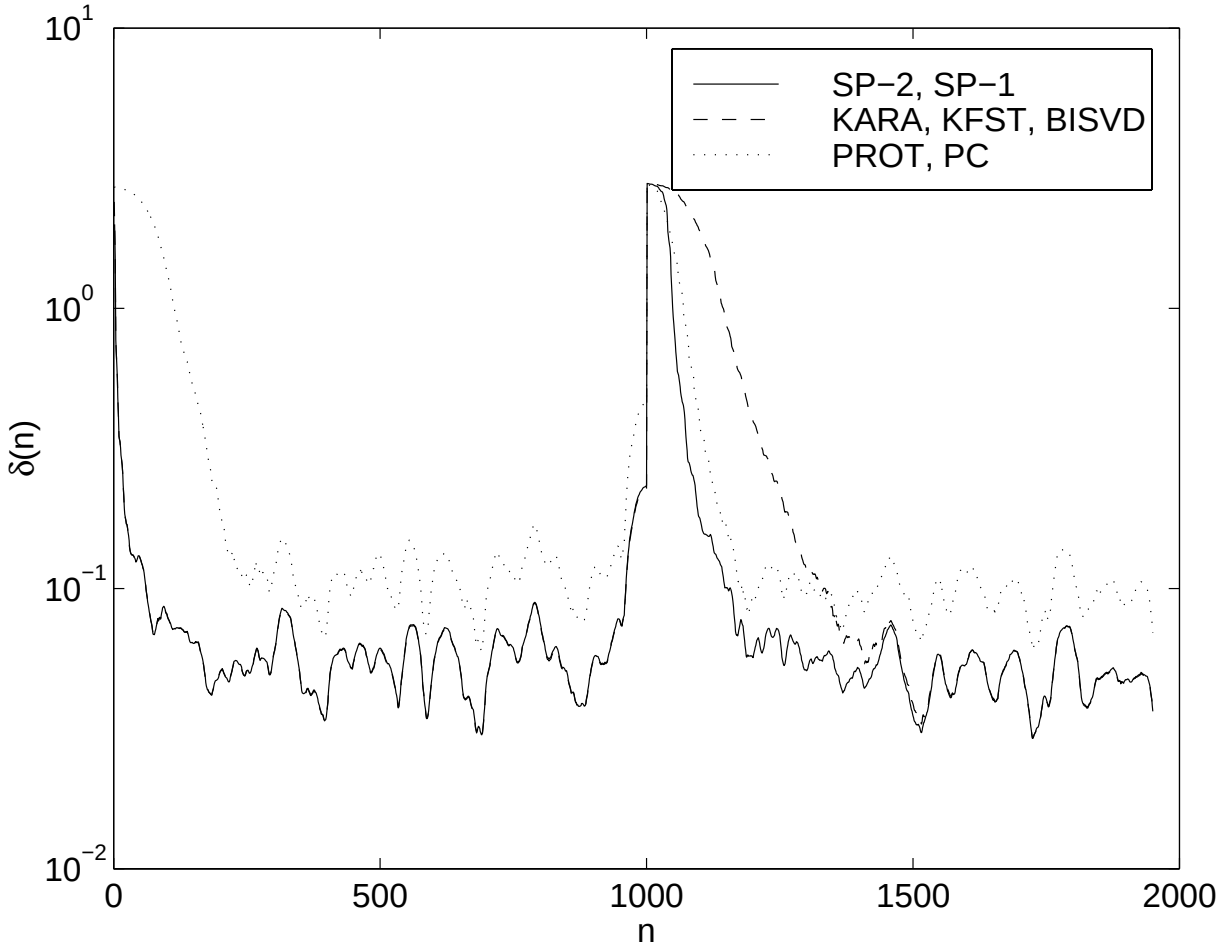

Figure 4: Comparison of SP-1 and SP-2 with subspace averaging algorithms (KARA, KFST, BISVD), and perturbation algorithms (PROT, PC) for error measure $\delta(n)$.

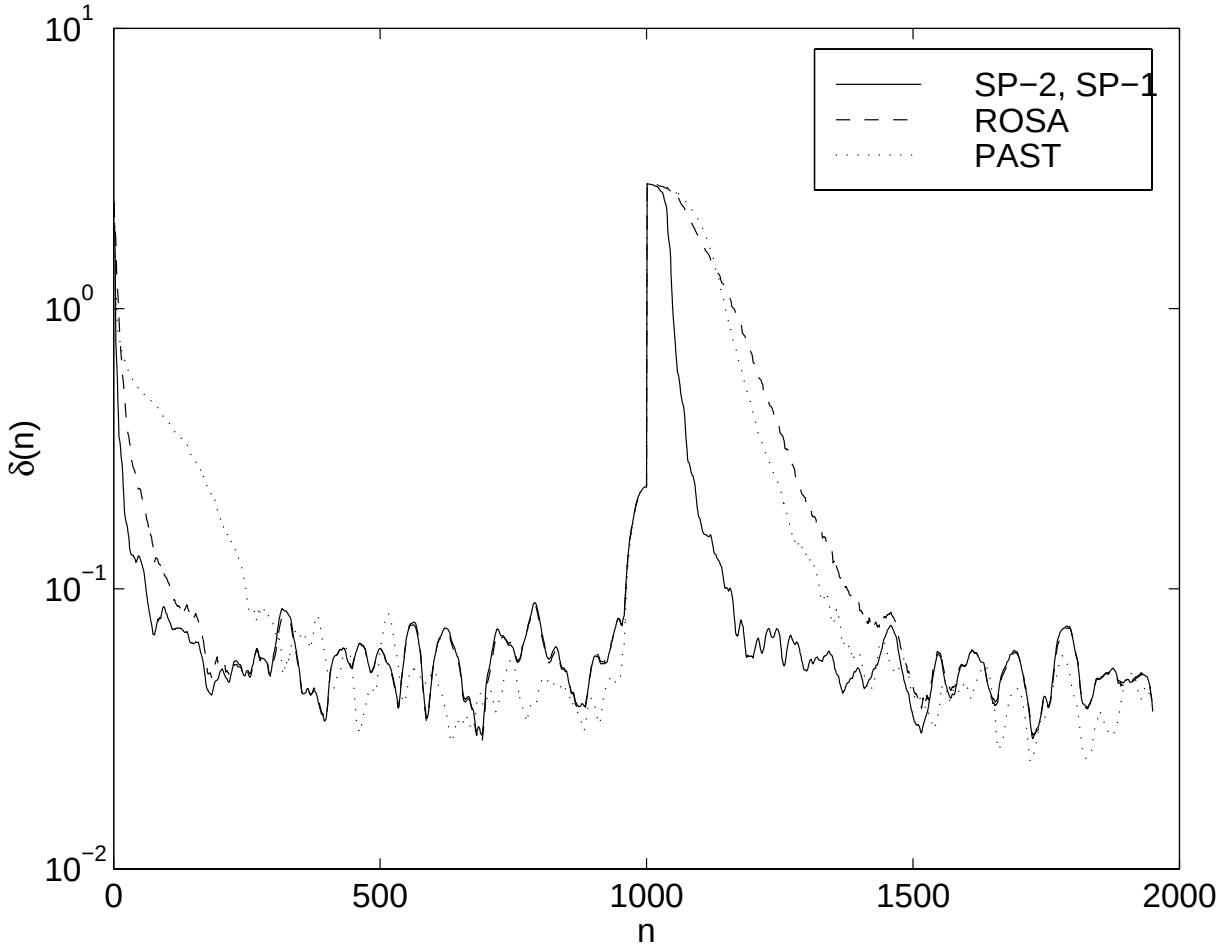

Figure 5: Comparison of SP-1 and SP-2 with ROSA and PAST for error measure $\delta(n)$.

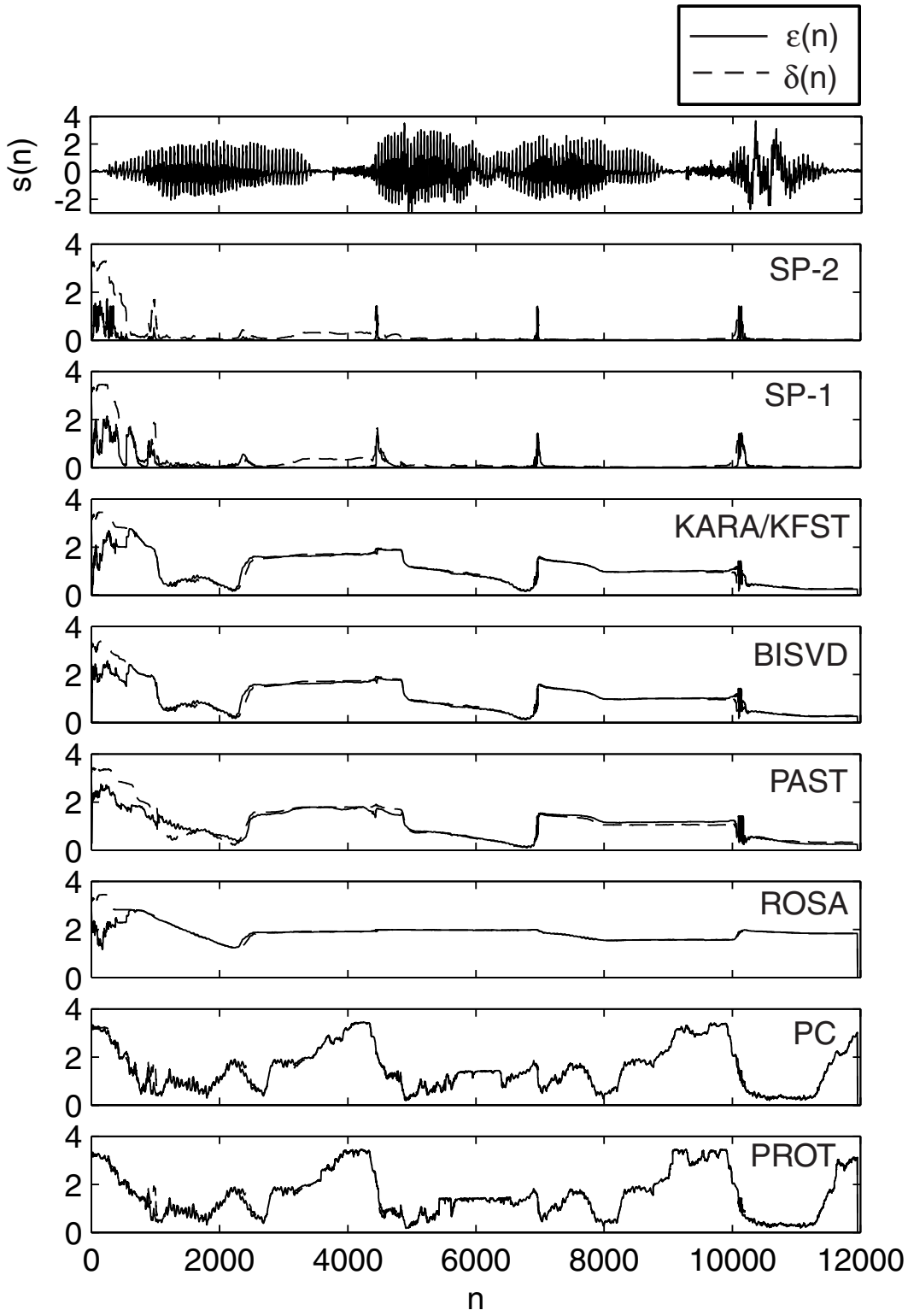

Figure 6: Subspace tracking experiments for a noisy speech signal. The top plot is the noise-free speech segment, the remaining plots show the error measures $\epsilon(n)$ and $\delta(n)$ vs. sample number.

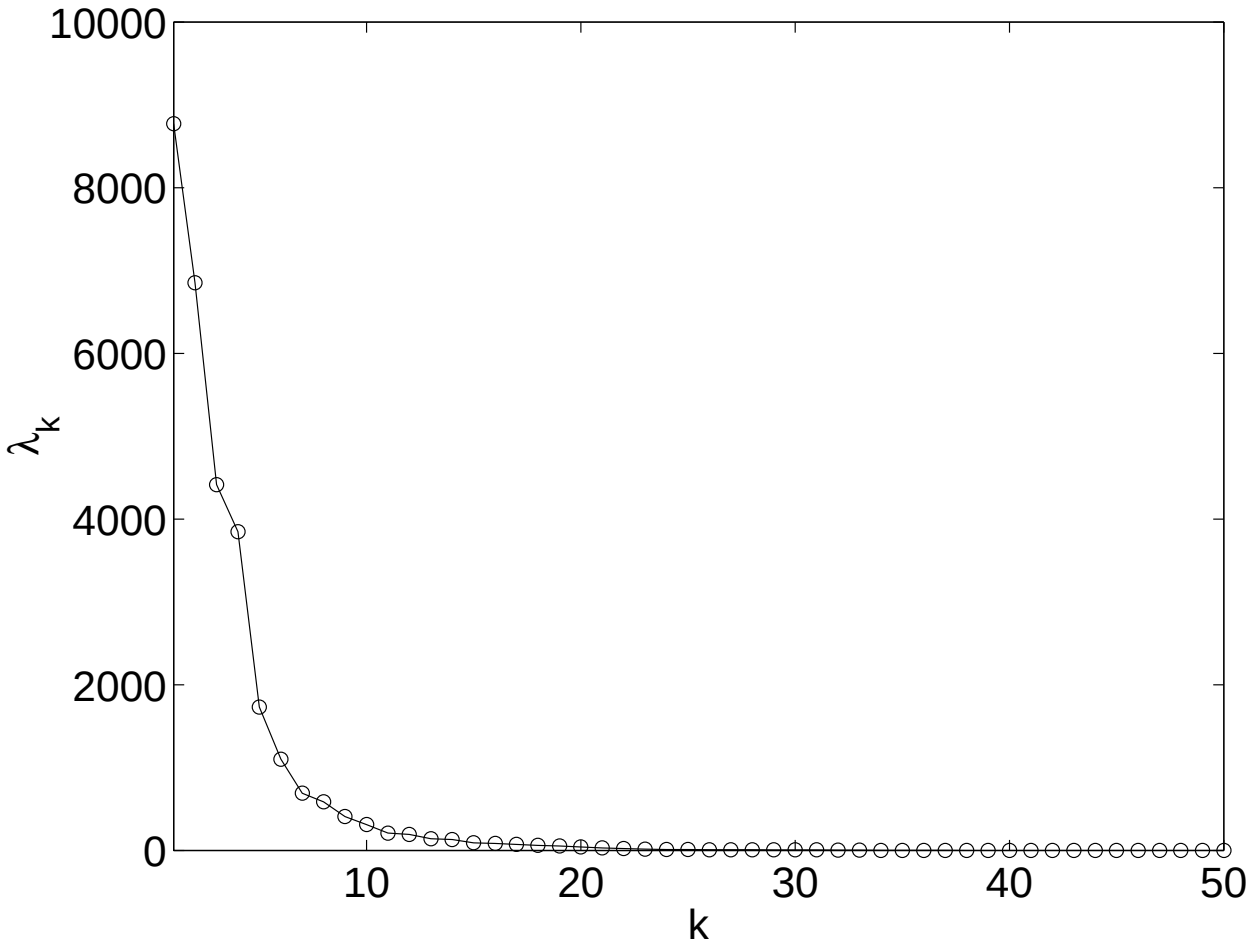

Figure 7: Mean eigenvalue distribution for the speech signal used in the experiments.