

A Direction Set Based Algorithm for Adaptive Least Squares Problems in Signal Processing

Mei-Qin Chen*

Abstract

A fast algorithm, called the direction set based algorithm, has been developed recently for solving a class of adaptive least squares problems arising in signal processing. The algorithm is based on the direction set method developed by Powell and Zangwill for solving unconstrained minimization problems without using derivatives. It is designed so as to fully take advantage of the special structure of the adaptive least squares problems. It is a fast algorithm because it requires only $O(N)$ multiplications for each system update where N is the number of parameters used in the system design. The algorithm has been implemented and applied to applications in signal processing. Computer simulation results have shown that the algorithm is stable and converges fast often with a rate which is comparable to that of the known Conjugate Gradient algorithm and Recursive Least Squares algorithm. The algorithm has been proved to converge linearly for adaptive least squares problems in general and its rate of convergence can be faster than linear for some applications. The algorithm can be simplified and its rate of convergence can be improved with some direction sets. The direction set based algorithm has also been modified to solve constrained adaptive least squares problems arising in spectral estimation. Computer simulations illustrate that the algorithm is effective for spectral estimation.

Keywords: adaptive least squares problems, direction set methods, adaptive filtering, spectral estimation

1 Introduction

The adaptive least squares (ALS) problems of the form given below are arising in many applications in signal processing [20]. At the n th state, the

*Department of Mathematics and Computer Science, The Citadel, Charleston, SC 29409. E-mail: chenm@citadel.edu. This research is supported by the Citadel Development Foundation Research Grant.

minimization problem

$$\min_{x \in R^N} J_n(x) = \sum_{i=0}^n \lambda_i^{(n)} (a_i^T x - s_i)^2, \quad (1)$$

is to be solved where $a_i \in R^N$, $s_i \in R$, and $0 \leq \lambda_i^{(n)} \leq 1$. Vectors a_i 's are formed by input signals u_k and desired output signals s_k at the k th state for all $k \leq n$. For example, a_i 's are formed only by input signals for FIR filters; by input and output signals for IIR filters; and by input and output signals and cross terms $u_i s_j$ for bilinear filters. Vector x , called a system update, is an estimate of the solution of the problem in (1) and is updated numerically by minimizing the objective function $J_n(x)$. Choices of $\lambda_i^{(n)}$ are problem dependent and some commonly used choices will be discussed in Section 2. Note that the objective function in (1) changes from one state to the next by adding on the weighted current error square which can be relatively small if x is close to the solution.

Among iterative methods for solving problems in (1), the Least Mean Square (LMS) algorithm is widely used because of its simplicity in implementation and its $O(N)$ computational complexity. However, it is known that the convergence of the LMS algorithm is very slow. The Recursive Least Squares (RLS) algorithm, on the other hand, converges significantly faster over the LMS algorithm, but is computationally intensive and requires $O(N^2)$ multiplications to compute each system update. Fast RLS algorithms are designed to reduce the computational complexity of the RLS algorithm to $O(N)$ with different ways to approximate the gain vector. Detailed derivations and analysis of some of these algorithms can be found in [6, 22, 23, 27, 28]. As pointed out in [20], some of the fast RLS algorithms have a tendency to become numerically unstable.

The Conjugate Gradient (CG) algorithm with a block of M pairs of input and desired output signals for linear and nonlinear filters is studied in [1, 2]. It requires $O(MN)$ multiplications for each system update and is more efficient than the RLS algorithm even when $M = N$ because of its fast convergence rate which leads to the fact that the overall computational complexity is much less. The algorithm with $M = 1$ has been studied, modified and tested for adaptive filtering and spectral estimation [14, 15, 16]. The algorithm takes the structure of the ALS problems into account to perform sample-by-sample updating of the system. Its performance can be comparable to the RLS and LMS-Newton algorithms with a much lower cost though the computational complexity is still $O(N^2)$ in general.

The preconditioned Conjugate Gradient (PCG) algorithms with FFT-based preconditioners and Strang-type preconditioners for Toeplitz matrices are introduced in [10, 11, 12, 13, 24, 25]. These algorithms are very efficient and require only $O(N \log N)$ multiplications for each system update for ALS problems with a Toeplitz or a near Toeplitz structure.

The CG based algorithms and PCG algorithms have fast rates of convergence, however, they require $O(N^2)$ and $O(N \log(N))$ multiplications respectively, for each system update for LAS problems in general because of the matrix-vector multiplication required by the algorithms. Clearly, an algorithm to be sought is the one which can maintain the fast rate of convergence that the CG based algorithms have, is stable and requires $O(N)$ multiplications for each system update.

In this paper, we present a newly developed fast algorithm, called the direction set (DS) based algorithm, for solving ALS problems in (1). The algorithm is based on the direction set method introduced first by Powell [26] and then modified by Zangwill [33]. The DS based algorithm computes each system update in $O(N)$ multiplications, is stable and converges as fast as the CG based algorithms or the RLS based algorithm for many applications. The rest of the paper is organized as follows. The ALS problems with choices of $\lambda_i^{(n)}$ and their vector-matrix notations will be discussed in Section 2. The DS methods, the DS method developed by Powell and Zangwill and the DS based algorithm for solving ALS problems will be stated and discussed in Section 3. The computational complexity and convergence results of the algorithm will also be given in Section 3. The computational complexities and performances of the DS based algorithm with different choices of direction sets will be discussed in Section 4. Computer simulations of the DS based algorithm for applications in adaptive filtering will be given in Section 5. The algorithm has been modified to solve constrained ALS problems arising in spectral estimation. The modified algorithm and the computer simulations for applications in spectral estimation will be presented in Section 6.

2 Structures and Properties of ALS Problems

In this section, we first discuss the choices of $\lambda_i^{(n)}$ in the ALS problems that are commonly used and are considered in this paper and we then reformulate the ALS problems in vector-matrix notations.

2.1 Choices of $\lambda_i^{(n)}$

Choices of $\lambda_i^{(n)}$ in ALS problems in (1) are problem dependent and some of commonly used ones are listed below.

Type I: $\lambda_i^{(n)} = \lambda^{n-i}$, where $0 \leq \lambda \leq 1$. The objective function $J_n(x)$ is defined by the sum of exponentially weighted least squares:

$$J_n(x) = \sum_{i=0}^n \lambda^{n-i} (a_i^T x - s_i)^2. \quad (2)$$

The constant λ is commonly called the exponential weighting factor or the forgetting factor. Two extreme choices for λ are 0 and 1. When $\lambda = 1$, the ALS problems are equivalent to total least squares problems:

$$\min_{x \in R^N} J_n(x) = \sum_{i=0}^n (a_i^T x - s_i)^2. \quad (3)$$

When $\lambda = 0$ (letting $\lambda^0 = 1$), the objective function becomes very simple and only depends on the current error square:

$$J_n(x) = (a_n^T x - s_n)^2. \quad (4)$$

Note that this objective function is also used by the LMS algorithm.

Type II: $\lambda_i^{(n)} = \lambda^{(n)} = 1/n$. The objective function is defined by the average of the total sum of the error squares:

$$J_n(x) = \frac{1}{n+1} \sum_{i=0}^n (a_i^T x - s_i)^2. \quad (5)$$

2.2 Vector-Matrix Notations

Let

$$\begin{aligned} A_n &= [\sqrt{\lambda_0^{(n)}} a_0, \sqrt{\lambda_1^{(n)}} a_1, \dots, \sqrt{\lambda_n^{(n)}} a_n], \\ b_n &= [\sqrt{\lambda_0^{(n)}} b_0^T, \sqrt{\lambda_1^{(n)}} s_1, \dots, \sqrt{\lambda_n^{(n)}} s_n]^T. \end{aligned}$$

Then the objective function J_n in (1) can be formulated in vector-matrix notations as

$$J_n(x) = x^T A_n A_n^T x - 2x^T A_n b_n + b_n^T b_n.$$

If we further let $Q_n = A_n A_n^T$, $B_n = A_n b_n$ and $c_n = b_n^T b_n$, then the minimization problem in (1) becomes

$$\min_{x \in R^N} J_n(x) = x^T Q_n x - 2x^T B_n + c_n. \quad (6)$$

Note that the following properties (P1)-(P4) hold for J_n in (6) and $\lambda_i^{(n)}$ in Type I and Type II.

(P1) $J_{n+1}(x) = \phi_n J_n(x) + \psi_n (a_{n+1}^T x - s_{n+1})^2$

(P2) $Q_{n+1} = \phi_n Q_n + \psi_n a_{n+1} a_{n+1}^T$

(P3) $B_{n+1} = \phi_n B_n + \psi_n s_{n+1} a_{n+1}$

(P4) $c_{n+1} = \phi_n c_n + \psi_n s_{n+1}^2$

where $\phi_n = \lambda$ and $\psi_n = 1$ for Type I; and $\phi_n = \frac{n}{n+1}$ and $\psi_n = \frac{1}{n+1}$ for Type II.

Remark 2.1 Properties (P1)-(P4) describe how objective function $J_n(x)$ changes from one state to the next. Property (P2) states that the change of the Hessian matrix Q_n between two consecutive states is a rank-one matrix. If the vector $w = Q_n v$ is known, then the calculation of

$$Q_{n+1}v = \phi_n w + \psi_n a_{n+1}^T v a_{n+1}$$

can be done in $O(N)$ multiplications. This is the key to the development of the direction set based algorithm for ALS problems presented in the next section.

3 The DS Based Algorithm for ALS Problems

In this section, we first introduce the algorithm based on the direction set method developed by Powell and Zangwill for the ALS problems. We then discuss the computational complexity and convergence properties of the algorithm.

3.1 Direction Set Methods

Direction Set (DS) methods are originally designed for solving unconstrained minimization problems of the form:

$$\min_{x \in R^N} f(x),$$

without using the first partial derivatives of f for the cases where the derivatives either do not have the analytic forms or are very expensive to evaluate. Having a set of N linearly independent directions $\{d^{(1)}, \dots, d^{(N)}\}$ in R^N and an starting estimate $x^{(0)} \in R^N$ to the solution, a DS method searches along each direction with an exact line search in a cyclical manner:

$$x^{(i)} = x^{(i-1)} + \alpha^{(i)} d^{(i)}, \text{ for } i = 1, \dots, N$$

where $\alpha^{(i)}$ is chosen so that the objective function is minimized along the direction $d^{(i)}$, i.e.,

$$f(x^{(i-1)} + \alpha^{(i)} d^{(i)}) = \min_{\alpha \in R} f(x^{(i-1)} + \alpha d^{(i)}).$$

N estimates $x^{(1)}, \dots, x^{(N)}$ are generated after searching through all N directions, this is called one searching cycle. Another searching cycle starts if $x^{(N)}$ is not close enough to the solution. Before the next searching cycle starts, directions may be modified towards conjugate directions [26, 33] with respect to the Hessian of f or towards eigenvectors [4] of the Hessian of f ; or directions remain the same [29], and a new starting estimate may be chosen.

3.2 The Powell and Zangwill DS Algorithm

Powell's DS method [26] iteratively modifies directions towards a set of conjugate directions with respect to the Hessian of f . At the end of each searching cycle, Powell's DS method updates the directions by letting

$$d^{(i)} = d^{(i+1)} \text{ for } i = 1, \dots, N - 1$$

and

$$d^{(N)} = (x^{(N)} - x^{(0)}) / \|x^{(N)} - x^{(0)}\|,$$

and computing the new starting estimate as

$$x^{(0)} = x^{(N)} + \alpha^{(N+1)} d^{(N)}$$

where $\alpha^{(N+1)}$ is chosen such that

$$f(x^{(N)} + \alpha^{(N+1)} d^{(N)}) = \min_{\alpha \in \mathbb{R}} f(x^{(N)} + \alpha d^{(N)}).$$

The newly obtained direction $d^{(N)}$ is conjugate to the directions $d^{(1)}, \dots, d^{(N-1)}$ with respect to the Hessian if the starting estimate $x^{(0)}$ of the previous searching cycle is the minimum over the subspace spanned by directions $d^{(1)}, \dots, d^{(N-1)}$. For a quadratic convex problem, if an initial direction is replaced by the conjugate direction at every searching cycle, then the solution x^* is found in at most $N - 1$ searching cycles. So, this algorithm has the finite termination property. A detailed analysis of this method is given in [18, 26].

The finite termination property holds with the assumption that N directions at each searching cycle are linearly independent. However, this assumption may not hold for some minimization problems [33]. A checking procedure is added by Zangwill to the algorithm to determine at each searching cycle which direction should be replaced by the new direction or no direction should be replaced in order to maintain a set of linearly independent directions. With this checking procedure, the algorithm guarantees that

$$\det \left(\left[d^{(1)}, \dots, d^{(N)} \right] \right) = \epsilon$$

for any given $\epsilon > 0$. The Powell and Zangwill DS algorithm for solving

$$\min_{x \in \mathbb{R}^N} f(x)$$

is stated in Table 1.

3.3 The DS Based Algorithm for ALS Problems

When the Powell and Zangwill DS algorithm is applied to solve an ALS problem given in (1), it can be simplified by the properties (P1)-(P4). Observe the following.

Table 1: The Powell and Zangwill DS Algorithm

<p>Given N linearly independent and normalized directions $d^{(1)}, \dots, d^{(N)}$, an initial estimate $x^{(0)}$, and scalars $0 < \epsilon \leq 1$ and $\tau > 0$, repeat Step I and Step II.</p>	
Step I	<p>For $i = 1, \dots, N$,</p> <p>(1) compute $\alpha^{(i)}$ such that</p> $f(x^{(i-1)} + \alpha^{(i)} d^{(i)}) = \min_{\alpha \in R} f(x^{(i-1)} + \alpha d^{(i)});$ <p>(2) set $x^{(i)} = x^{(i-1)} + \alpha^{(i)} d^{(i)}$.</p>
Step II	<p>Let $\beta = \ x^{(N)} - x^{(0)}\$.</p> <p>If $\beta \leq \tau$, then the algorithm terminates and $x^* \approx x^{(N)}$.</p> <p>Otherwise, let $d^{(N+1)} = (x^{(N)} - x^{(0)})/\beta$ and</p> <p>(1) calculate $\alpha^{(N+1)}$ such that</p> $f(x^{(N)} + \alpha^{(N+1)} d^{(N+1)}) = \min_{\alpha \in R} f(x^{(N)} + \alpha d^{(N+1)});$ <p>(2) set $x^{(0)} = x^{(N)} + \alpha^{(N+1)} d^{(N+1)}$.</p> <p>Let $\alpha^{(s)} = \max_{1 \leq i \leq N} \{\alpha^{(i)}\}$.</p> <p>If $\frac{\alpha^{(s)} \delta}{\beta} \geq \epsilon$, then $d^{(s)} = d^{(N+1)}$, and $\delta = \frac{\alpha^{(s)} \delta}{\beta}$.</p>

Observation 3.1 Since the objective function in (1) at each searching cycle is quadratic, the minimization problem:

$$\min_{\alpha \in R} J_n(x^{(i-1)} + \alpha d^{(i)})$$

can be solved exactly and the stepsizes $\alpha^{(i)}$ are

$$\alpha^{(i)} = -\frac{(d^{(i)})^T (Q_n x^{(i-1)} - B_n)}{(d^{(i)})^T Q_n d^{(i)}}, \quad \text{for } i = 1, \dots, N+1. \quad (7)$$

Observation 3.2 With saved vectors $v^{(i)} = Q_n x^{(i)}$ and $w^{(i)} = Q_n d^{(i)}$,

$$v^{(i+1)} = Q_n x^{(i+1)} = v^{(i)} + \alpha^{(i)} w^{(i)}, \quad \text{for } i = 0, \dots, N-1.$$

Observation 3.3 With saved vectors $v^{(N)}$ and $v^{(0)}$,

$$w^{(N+1)} = Q_n d^{(N+1)} = Q_n (x^{(N)} - x^{(0)})/\beta = (v^{(N)} - v^{(0)})/\beta,$$

where $\beta = \|x^{(N)} - x^{(0)}\|$; and the vector $v^{(0)}$ for the next searching cycle can be computed as

$$v^{(0)} = Q_{n+1} x^{(N+1)} = \phi_n(v^{(N)} + \alpha^{(N+1)} w^{(N+1)}) + \psi_n a_{n+1}^T x^{(N+1)} a_{n+1}.$$

Observation 3.4 With saved vectors $w^{(i)}$,

$$w^{(i)} = Q_{n+1} d^{(i)} = \phi_n w^{(i)} + \psi_n a_{n+1}^T d^{(i)} a_{n+1}, \quad \text{for } i = 1, \dots, N+1.$$

The DS based algorithm for ALS problems is stated in Table 2.

Table 2: The DS Based Algorithm (1)

<p>Given N linearly independent and normalized directions $d_0^{(1)}, \dots, d_0^{(N)}$, an initial estimate $x_0^{(0)}$, and scalars $0 < \epsilon \leq 1$ and $\tau > 0$, compute $v_0^{(0)} = Q_0 x_0^{(0)}$ and $w_0^{(i)} = Q_0 d_0^{(i)}$ for $i = 1, \dots, N$, and repeat Step I and Step II for $k = 1, 2, \dots$.</p>	
Step I	<p>For $1 \leq i \leq N$, compute</p> <ol style="list-style-type: none"> (1) if $k \geq 1$, then <ol style="list-style-type: none"> if $i \neq s$ then $w_k^{(i)} = \phi_{k-1} w_{k-1}^{(i)} + \psi_{k-1} a_k^T d_{k-1}^{(i)} a_k$; if $i = s$ then $w_k^{(i)} = \phi_{k-1} w_{k-1}^{(N+1)} + \psi_{k-1} a_k^T d_{k-1}^{(N+1)} a_k$; (2) $\alpha_k^{(i)} = -\frac{(d_k^{(i)})^T (v_k^{(i-1)} - B_k)}{(d_k^{(i)})^T w_k^{(i)}}$; (3) $x_k^{(i)} = x_k^{(i-1)} + \alpha_k^{(i)} d_k^{(i)}$; (4) $v_k^{(i)} = v_k^{(i-1)} + \alpha_k^{(i)} w_k^{(i)}$.
Step II	<p>For $i = N + 1$, let $\beta_k = \ x_k^{(N)} - x_k^{(0)}\$. If $\beta_k \leq \tau$, then the algorithm terminates, and $x^* \approx x_k^{(N)}$. Otherwise, set implicitly $d_k^{(N+1)} = (x_k^{(N)} - x_k^{(0)})/\beta_k$ and $w_k^{(N+1)} = (v_k^{(N)} - v_k^{(0)})/\beta_k$.</p> <ol style="list-style-type: none"> (1) If $\ Q_k x_k^{(N)} - B_k\ \leq \tau$, then the algorithm terminates, and $x^* \approx x_k^{(N)}$; otherwise $\alpha_k^{(N+1)} = -\frac{(d_k^{(N+1)})^T (v_k^{(N)} - B_k)}{(d_k^{(N+1)})^T w_k^{(N+1)}}$; (2) $x_{k+1}^{(0)} = x_k^{(N)} + \alpha_k^{(N+1)} d_k^{(N+1)}$; (3) $v_{k+1}^{(0)} = \phi_k (v_k^{(N)} + \alpha_k^{(N+1)} w_k^{(N+1)}) + \psi_k a_{k+1}^T x_k^{(N+1)} a_{k+1}$; (4) $B_{k+1} = B_k + \psi_k s_{k+1} a_{k+1}$. (5) Let $\alpha_k^{(s)} = \max_{1 \leq i \leq N} \{\alpha_k^{(i)}\}$. If $\frac{\alpha_k^{(s)} \delta_k}{\beta_k} \geq \epsilon$, then <ol style="list-style-type: none"> (i) set $\delta_{k+1} = \frac{\alpha_k^{(s)} \delta_k}{\beta_k}$; (ii) let $d_{k+1}^{(i)} = d_k^{(i)}$ for $i \neq s$ and $d_{k+1}^{(s)} = d_k^{(N+1)}$. Otherwise, <ol style="list-style-type: none"> (i) set $\delta_{k+1} = \delta_k$; (ii) let $d_{k+1}^{(i)} = d_k^{(i)}$ for $i = 1, \dots, N$; (iii) set $s = 0$.

Table 3: # of Multiplications for the DS Based Algorithm (1)

$x_k^{(i)}, 1 \leq i \leq N$		$x_k^{(N+1)}$	
Item	# of multiplications	Item	# of multiplications
$w_k^{(i)}$	$3N$	β_k	N
$\alpha_k^{(i)}$	$2N + 1$	$\alpha_k^{(N+1)}$	$2N + 2$
$x_k^{(i)}$	N	$x_k^{(N+1)}$	$N + 1$
$v_k^{(i)}$	N	$v_k^{(N+1)}$	$4N + 1$
		B_{k+1}	$N + 1$
Total	$7N + 1$	Total	$9N + 5$

Table 4: Comparisons of Computational Cost

algorithm	# of multiplications
RLS	$4N^2 + 4N$
CG (block size M)	$4MN + 8N + 2$
Stabilized FTF	$9N$
Fast RLS (Qiao)	$6N$
DS Based Algorithm (1)	$9N + 5$
DS Based Algorithm (2) and (3)	$7N + 5$

3.4 Computational Complexity of the DS Based Algorithm (1)

The number of multiplications required by each step in Step I and Step II is given in Table 3. The DS based algorithm computes N system updates by the steps given in Step I with $7N + 1$ multiplications for each system update, and one system update by the steps given in Step II with $9N + 5$ multiplications. The computational complexity of the algorithm is therefore $9N + 5$. For comparison purposes, the numbers of multiplications for the RLS algorithm, the block CG algorithm [2] with a block size of M , the stabilized FTF algorithm given in [28], the fast RLS algorithm given in [27] and the DS Based Algorithm (2) and (3) which are discussed in Section 4.1 are listed in Table 4. Note that the DS based algorithm computes one system update for each oncoming pair of input and desired output signals but updates the objective function only once for every searching cycle. If the input data indicate the system changes rapidly, then updating the objective function with each oncoming pair of input and desired output signals may be important. In this case, the computational complexity of the DS Based Algorithm (1) is no longer $O(N)$. However, with choosing a special set of directions and weights $\lambda_i^{(n)}$, the DS Based Algorithm (1) can still maintain the $O(N)$ computational complexity as the objective function is updated for each oncoming pair of input and desired output signals. Details

are given in Section 4.1.

3.5 Convergence Analysis

Assume that $\text{rank}(A_l) = N$ for some positive integer $l \geq 0$. Then $\text{rank}(A_k) = N$ and Q_k are symmetric and positive definite $\forall k \geq l$. Since

$$(d_k^{(i)})^T Q_k d_k^{(i)} > 0, \text{ for all } k \geq l \text{ and } i = 1, \dots, N + 1,$$

stepsizes $\alpha_k^{(i)}$ in (7) are well-defined for all $k \geq l$ and $i = 1, \dots, N + 1$. For $k < l$ and for some i , it is possible that $(d_k^{(i)})^T Q_k d_k^{(i)} = 0$. In this case, we let $\alpha_k^{(i)} = 0$. So the algorithm is well defined for all $k \geq 0$. Since

$$J_k(x_k^{(i-1)}) - J_k(x_k^{(i)}) = \frac{[(d_k^{(i)})^T (Q_k x_k^{(i-1)} - B_k)]^2}{(d_k^{(i)})^T Q_k d_k^{(i)}} \geq 0$$

for all $k \geq l$ and $i = 1, \dots, N + 1$, directions $d_k^{(1)}, \dots, d_k^{(N+1)}$ are non-ascent. Furthermore, at least one of the directions is descent at each searching cycle, that is explained in Lemma 3.1.

Assume that the solution x^* of a given ALS problem in (1) exists, i.e., $A_k A_k^T x^* - A_k b_k = 0$, for all $k \geq 0$. Lemma 3.1 shows that the algorithm terminates as the sequence of estimates converges to the solution.

Lemma 3.1 *Let $x_k^{(1)}, \dots, x_k^{(N+1)} = x_{k+1}^{(0)}$ be generated by the DS based algorithm at the k th searching cycle from $x_k^{(0)}$. Let x^* be the solution of the ALS problem in (1). Then (a) $\beta_k = \|x_k^{(N)} - x_k^{(0)}\| = 0$ if and only if $x_k^{(0)} = x^*$; and (b) $\|Q_k x_k^{(N)} - B_k\| = 0$ if and only if $x_k^{(N)} = x^*$.*

The following convergence results show that the DS Based Algorithm (1) converges asymptotically and linearly for ALS problems. Proofs are given in [8]. Let $\lambda_i(Z)$ be denoted as the i th eigenvalue of matrix Z where

$$\lambda_1(Z) \leq \dots \leq \lambda_N(Z).$$

Let $m_k = \lambda_1(Q_k)$ and $M_k = \lambda_N(Q_k)$ for $k \geq 0$. Let $E(z)$ be denoted as the expectation value of z where z can be a scalar, a vector or a matrix. Let $R = E(a_i a_i^T)$ for all a_i given in (1).

Lemma 3.2 *Let $x_k^{(1)}, \dots, x_k^{(N+1)} = x_{k+1}^{(0)}$ be generated by the DS based algorithm at the k th searching cycle from $x_k^{(0)}$. Let $\theta_k^{(i)}$ be denoted as the angle between the direction $d_k^{(i)}$ and the gradient vector $\nabla J_k(x_k^{(i-1)})$ for $i = 1, \dots, N + 1$. Then*

$$\|x_k^{(N+1)} - x^*\|_{Q_k}^2 \leq \prod_{j=N}^0 \left(1 - \frac{m_k}{M_k} \cos^2 \theta_k^{(j+1)}\right) \|x_k^{(0)} - x^*\|_{Q_k}^2, \quad \forall k. \quad (8)$$

Theorem 3.1 Let $x_k^{(1)}, \dots, x_k^{(N+1)} = x_{k+1}^{(0)}$ be generated by the DS based algorithm at the k th searching cycle from $x_k^{(0)}$. Let $\theta_k^{(i)}$ be denoted as the angle between the direction $d_k^{(i)}$ and the gradient vector $\nabla J_k(x_k^{(i-1)})$ for $i = 1, \dots, N+1$. For a given ALS problem in (1), there exists a large positive integer K such that for $k \geq K$,

$$\|x_{k+L}^{(0)} - x^*\|_{Q_{k+L}}^2 \leq \prod_{l=L-1}^0 \prod_{j=N}^0 \left(1 - \frac{m_k}{M_k} \cos^2 \theta_{k+l}^{(j+1)}\right) \|x_k^{(0)} - x^*\|_{Q_k}^2, \quad \forall L \geq 1. \quad (9)$$

Theorem 3.2 Let $x_k^{(1)}, \dots, x_k^{(N+1)} = x_{k+1}^{(0)}$ be generated by the DS based algorithm at the k th searching cycle from $x_k^{(0)}$. Let x^* be the solution of the ALS problem in (1). Then $\lim_{k \rightarrow \infty} x_k^{(0)} = x^*$ linearly.

Remark 3.1 In the proof of Theorem 3.2, it is shown that

$$\|x_{k+1}^{(0)} - x^*\|_R^2 < \|x_k^{(0)} - x^*\|_R^2$$

for k sufficiently large. So, the DS based algorithm is stable for the ALS problems.

Remark 3.2 The inequality in (9) indicates that the rate of convergence of the DS based algorithm for the ALS problems depends on the ratio $\frac{m_k}{M_k}$ and $\cos \theta_k^{(i)}$. The algorithm converges faster if $\frac{m_k}{M_k} \approx 1$. So the rate of convergence of the algorithm can be improved by reducing the condition number of Q_k . Based on this idea, a computational procedure is designed to accelerate the rate of convergence. Details are given in Section 4.2. The value of $\cos \theta_k^{(i)}$ is determined by vectors $d_k^{(i)}$ and $\nabla J_k(x_k^{(i-1)})$, and $\cos^2 \theta_k^{(i)} = 1$ if $d_k^{(i)} = -\nabla J_k(x_k^{(i-1)})$. This choice for each $d_k^{(i)}$ increases the computational complexity of the algorithm to $O(N^2)$ and therefore is not in general an acceptable option for accelerating the rate of convergence.

Remark 3.3 Originally, the purpose of modifying the direction set by replacing $d_n^{(s)}$ by $d_n^{(N+1)}$ is to iteratively generate a set of conjugate directions with respect to the Hessian Q_n of the objective function $J_n(x)$. In the case where Q_n changes significantly from one state to the next it may not be important to generate a conjugate direction at each searching cycle. Also, the proofs of convergence results do not depend on the direction $d_n^{(N+1)}$ nor the new starting estimate $x_n^{(N+1)}$. To further simplify the algorithm, we can let $x_{n+1}^{(0)} = x_n^{(N)}$.

4 Choices of Direction Sets

As given in the previous section, the DS based algorithm converges asymptotically and linearly for ALS problems with a set of linearly independent directions. The algorithm can be simplified and its rate of convergence can be

improved by some direction sets. In this section, we present the DS based algorithm with several special direction sets.

4.1 N Euclidean Coordinate Directions in R^N

Let $e_i = [0, \dots, 0, 1, 0, \dots, 0]^T$ be the i th standard basis vector in R^N . Let $\{d_k^{(1)}, \dots, d_k^{(N)}\} = \{e_1, \dots, e_N\}$ for $k \geq 0$. Vector x_{k+1} is updated from x_k by N Euclidean coordinate directions with searching along one direction at a time in a cyclical manner. Let $(z)_i$ be the i th element of the vector z . Observe that

- (i) $(d_k^{(i)})^T z = e_i^T z = (z)_i$; and
- (ii) $x_k^{(N)} - x_k^{(0)} = \alpha_k$ where $\alpha_k = [\alpha_k^{(1)}, \dots, \alpha_k^{(N)}]^T$
and $\beta_k = \|x_k^{(N)} - x_k^{(0)}\| = \sqrt{(\alpha_k^{(1)})^2 + \dots + (\alpha_k^{(N)})^2}$.

The DS Based Algorithm (2), the DS Based Algorithm (1) with a set of N Euclidean coordinate directions, is stated in Table 5 and the number of multiplications at each step is given in Table 6. This algorithm computes N system updates by the steps in Step I with $3N + 4$ multiplications for each system update and one system update by the steps in Step II with $7N + 5$ multiplications. Hence, without and with a new starting estimate at each searching cycle, the total numbers of multiplications required are $3N + 4$ and $7N + 5$, respectively.

Now consider updating x_{k+1} from x_k using N Euclidean coordinate directions with searching along two directions at a time in a cyclical manner assuming N is an even number. Then

$$x_k^{(i)} = x_k^{(i-1)} + \alpha_k^{(2i-1)} e_{2i-1} + \alpha_k^{(2i)} e_{2i}, \text{ for } i = 1, \dots, \frac{N}{2}$$

where $\alpha_k^{(2i-1)}$ and $\alpha_k^{(2i)}$ are chosen so that

$$J_k(x_k^{(i-1)} + \alpha_k^{(2i-1)} e_{2i-1} + \alpha_k^{(2i)} e_{2i}) = \min_{\alpha_1, \alpha_2 \in R} J_k(x_k^{(i-1)} + \alpha_1 e_{2i-1} + \alpha_2 e_{2i}).$$

Stepsizes $\alpha_k^{(2i-1)}$ and $\alpha_k^{(2i)}$ are solutions of the linear system:

$$\begin{bmatrix} \alpha_k^{(2i-1)} \\ \alpha_k^{(2i)} \end{bmatrix} = -\frac{1}{\nu_k^{(i)}} \begin{bmatrix} (w_k^{(2i)})_{2i} & -(w_k^{(2i)})_{2i-1} \\ -(w_k^{(2i)})_{2i-1} & (w_k^{(2i-1)})_{2i-1} \end{bmatrix} \begin{bmatrix} (v_k^{(i-1)} - B_k)_{2i-1} \\ (v_k^{(i-1)} - B_k)_{2i} \end{bmatrix} \quad (10)$$

where $\nu_k^{(i)} = (w_k^{(2i-1)})_{2i-1} (w_k^{(2i)})_{2i} - ((w_k^{(2i)})_{2i-1})^2$ for $i = 1, \dots, \frac{N}{2}$. The DS Based Algorithm (3), the DS Based Algorithm (1) with a set of $N/2$ double Euclidean coordinate directions, is stated in Table 7 and the number of multiplications at each step is given in Table 8. This algorithm computes $N/2$ system

Table 5: The DS Based Algorithm (2) (N single Euclidean coordinate directions)

<p>Given an initial estimate $x_0^{(0)}$, scalars $0 < \epsilon \leq 1$ and $\tau > 0$, and $Q_0 = [q_0^{(1)}, \dots, q_0^{(N)}]$, compute $v_0^{(0)} = Q_0 x_0^{(0)}$ and $w_0^{(i)} = q_0^{(i)}$ for $i = 1, \dots, N$, and repeat Step I and Step II for $k = 1, 2, \dots$.</p>	
Step I	<p>For $1 \leq i \leq N$, compute</p> <ol style="list-style-type: none"> (1) $w_k^{(i)} = \phi_{k-1} w_{k-1}^{(i)} + (\psi_{k-1} (a_k)_i) a_k$; (2) $\alpha_k^{(i)} = -\frac{(v_k^{(i-1)} - B_k)_i}{(w_k^{(i)})_i}$; (3) $(x_k)_i = (x_k^{(0)})_i + \alpha_k^{(i)}$; (4) $v_k^{(i)} = v_k^{(i-1)} + \alpha_k^{(i)} w_k^{(i)}$. (5) $\alpha_{ki} = (\alpha_k^{(i)})^2$ and $\mu_{ki} = \alpha_k^{(i)} (v_k^{(0)} - B_k)_i$.
Step II	<p>For $i = N + 1$, let $\beta_k = \sqrt{\alpha_{k1} + \dots + \alpha_{kN}}$. If $\beta_k \leq \tau$, then the algorithm terminates and $x^* \approx x_k$. Otherwise,</p> <ol style="list-style-type: none"> (1) $\alpha_k^{(N+1)} = -(1 + \frac{\mu_{k1} + \dots + \mu_{kN}}{\alpha_k^T (v_k^{(N)} - v_k^{(0)})}) \beta_k$ (2) $x_{k+1}^{(0)} = x_k + (\frac{\alpha_k^{(N+1)}}{\beta_k}) \alpha_k$; (3) $v_{k+1}^{(0)} = \phi_k (v_k^{(N)} + (\frac{\alpha_k^{(N+1)}}{\beta_k}) (v_k^{(N)} - v_k^{(0)})) + \psi_k a_{k+1}^T x_{k+1}^{(0)} a_{k+1}$; (4) $B_{k+1} = B_k + \psi_k s_{k+1} a_{k+1}$.

Table 6: # of Multiplications for the DS Based Algorithm (2)

$(x_k)_i, 1 \leq i \leq N$		$x_k^{(N+1)}$	
Item	# of multiplications	Item	# of multiplications
$w_k^{(i)}$	$2N + 1$	β_k	0
$\alpha_k^{(i)}$	1	$\alpha_k^{(N+1)}$	$N + 2$
$(x_k)_i$	0	$x_k^{(N+1)}$	$N + 1$
$v_k^{(i)}$	N	$v_k^{(N+1)}$	$4N + 1$
α_{ki}, μ_{ki}	2	B_{k+1}	$N + 1$
Total	$3N + 4$	Total	$7N + 5$

updates by the steps in Step I with $6N + 14$ multiplications for each system update and one system update by the steps in Step II with $7N + 5$ multiplications. Hence, without and with a new starting estimate at each searching cycle, the total numbers of multiplications required are $6N + 14$ and $7N + 5$, respectively.

Similarly, x_{k+1} can be updated from x_k using N Euclidean coordinate directions with searching along M directions at a time assuming N/M is an integer. However, the total number of multiplications required in Step I for each system update increases because of the computational complexity of solving the minimization problem:

$$\begin{aligned} & J_k(x_k^{(i-1)} + \alpha_k^{(Mi-1)} e_{Mi-1} + \cdots + \alpha_k^{(Mi)} e_{Mi}) \\ &= \min_{\alpha_1, \dots, \alpha_M \in R} J_k(x_k^{(i-1)} + \alpha_1 e_{Mi-1} + \cdots + \alpha_M e_{Mi}). \end{aligned}$$

The Choice $M = 2$ seems a practical choice.

Remark 4.1 Without a new starting estimate at each searching cycle ($x_{k+1}^{(0)} = x_k^{(N)}$), the DS Based Algorithm (2) is equivalent to the Gauss-Seidel iteration in the sense that $x_k^{(N)}$ is identical to one Gauss-Seidel iteration for solving $Q_k x = B_k$ with starting estimate $x_k^{(0)}$. The DS Based Algorithm (3) is equivalent to the block Gauss-Seidel iteration with a block size of 2 in the same sense. Though the Hessian matrix Q_k for the ALS problems is changing from one state to the next, $Q_k \approx \eta R$ for sufficiently large k where $R = E(a_i a_i^T)$ and $\eta > 0$ as shown in the proof of Theorem 3.2 in [8]. Since Q_k are symmetric for all $k \geq 0$ and positive definite for sufficiently large k , matrix R is symmetric and positive definite. It is known [19] that the Gauss-Seidel iteration converges to the solution of $Ax = b$ for any starting estimate if A is symmetric and positive definite. This also proves that the DS Algorithm (1) with N Euclidean coordinate directions converges asymptotically for the ALS problems.

Remark 4.2 The simplicity of the DS based algorithm with a set of Euclidean coordinate directions and without a new starting estimate at each searching cycle allows us to update the objective function for each oncoming input and desired output signals without increasing the computational complexity to $O(N^2)$ for some applications. All DS based algorithms given so far do not require to form Hessian matrices Q_k explicitly. When $d_k^{(i)} = e_i$ for all $k \geq 0$, $Q_k e_i = q_k^{(i)}$ the i th column of Q_k . So, if Q_k can be formed from Q_{k-1} by $O(N)$ multiplications, then the DS based algorithm can be further simplified. For some applications, the number of multiplications needed for forming Q_k from Q_{k-1} is $O(N)$. For example, the vector a_k for FIR filters is formed by shifting down a_{k-1} with one element and adding one new sample. For IIR filters, the vector a_k is composed of two subvectors, one each for the inputs and past outputs, respectively. Each of the subvectors is individually shifted down one element

Table 7: The DS Based Algorithm (3) ($N/2$ double Euclidean coordinate directions)

<p>Given an initial estimate $x_0^{(0)}$, scalars $0 < \epsilon \leq 1$ and $\tau > 0$, and $Q_0 = [q_0^{(1)}, \dots, q_0^{(N)}]$, compute $v_0^{(0)} = Q_0 x_0^{(0)}$ and $w_0^{(i)} = q_0^{(i)}$ for $i = 1, \dots, N$, and repeat Step I and Step II for $k = 1, 2, \dots$.</p>	
Step I	<p>For $1 \leq i \leq N$, compute</p> <ol style="list-style-type: none"> (1) $w_k^{(2i-1)} = \phi_{k-1} w_{k-1}^{(2i-1)} + (\psi_{k-1} (a_k)_{2i-1}) a_k$; $w_k^{(2i)} = \phi_{k-1} w_{k-1}^{(2i)} + (\psi_{k-1} (a_k)_{2i}) a_k$; (2) $\nu_{ki} = (w_k^{(2i-1)})_{2i-1} (w_k^{(2i)})_{2i} - ((w_k^{(2i)})_{2i-1})^2$ $wv11 = (w_k^{(2i)})_{2i} (v_k^{(i-1)} - B_k)_{2i-1}$; $wv12 = (w_k^{(2i)})_{2i-1} (v_k^{(i-1)} - B_k)_{2i}$; $wv21 = (w_k^{(2i)})_{2i-1} (v_k^{(i-1)} - B_k)_{2i-1}$; $wv22 = (w_k^{(2i-1)})_{2i-1} (v_k^{(i-1)} - B_k)_{2i}$; $\alpha_k^{(2i-1)} = (-wv11 + wv12) / \nu_{ki}$ $\alpha_k^{(2i)} = (wv21 - wv22) / \nu_{ki}$; (3) $(x_k)_{2i-1} = (x_k^{(0)})_{2i-1} + \alpha_k^{(2i-1)}$, and $(x_k)_{2i} = (x_k^{(0)})_{2i} + \alpha_k^{(2i)}$; (4) $v_k^{(i)} = v_k^{(i-1)} + \alpha_k^{(2i-1)} w_k^{(2i-1)} + \alpha_k^{(2i)} w_k^{(2i)}$. (5) $\alpha_{ki} = (\alpha_k^{(2i-1)})^2 + (\alpha_k^{(2i)})^2$ and $\mu_{ki} = \alpha_k^{(2i-1)} (v_k^{(0)} - B_k)_{2i-1} + \alpha_k^{(2i)} (v_k^{(0)} - B_k)_{2i}$.
Step II	<p>For $i = N/2 + 1$, let $\beta_k = \sqrt{\alpha_{k1} + \dots + \alpha_{k(N/2)}}$. If $\beta_k \leq \tau$, then the algorithm terminates and $x^* \approx x_k$. Otherwise,</p> <ol style="list-style-type: none"> (1) $\alpha_k^{(N/2+1)} = -(1 + \frac{\mu_{k1} + \dots + \mu_{k(N/2)}}{\alpha_k^T (v_k^{(N)} - v_k^{(0)})}) \beta_k$ (2) $x_{k+1}^{(0)} = x_k + (\frac{\alpha_k^{(N+1)}}{\beta_k}) \alpha_k$; (3) $v_{k+1}^{(0)} = \phi_k (v_k^{(N)} + (\frac{\alpha_k^{(N+1)}}{\beta_k}) (v_k^{(N)} - v_k^{(0)})) + \psi_k a_{k+1}^T x_{k+1}^{(0)} a_{k+1}$; (4) $B_{k+1} = B_k + \psi_k s_{k+1} a_{k+1}$.

Table 8: # of Multiplications for the DS Based Algorithm (3)

$(x_k)_{2i-1}, (x_k)_{2i}, 1 \leq i \leq N/2$		$x_k^{(N/2+1)}$	
Item	# of multiplications	Item	# of multiplications
$w_k^{(2i-1)}, w_k^{(2i)}$	$4N + 2$	β_k	0
$\alpha_k^{(2i-1)}, \alpha_k^{(2i)}$	8	$\alpha_k^{(N+1)}$	$N + 2$
$(x_k)_{2i-1}, (x_k)_{2i}$	0	$x_k^{(N+1)}$	$N + 1$
$v_k^{(i)}$	$2N$	$v_k^{(N+1)}$	$4N + 1$
α_{ki}, μ_{ki}	4	B_{k+1}	$N + 1$
Total	$6N + 14$	Total	$7N + 5$

and added a new data sample. The shifted subvectors are then concatenated to form a_k . With these features, the calculation of $a_k a_k^T$ can be done in $O(N)$ multiplications when the matrix $a_{k-1} a_{k-1}^T$ is saved, and therefore the update of Q_k can be done explicitly in $O(N)$ multiplications. In the following, we discuss two DS based algorithms, both search along N Euclidean coordinate directions without a new starting estimate at each searching cycle and update the objective function for each oncoming input and desired output signals.

For the first algorithm, we let $\lambda = 1$, $Q_0 = a_0 a_0^T$ and $Q_k = Q_{k-1} + a_k a_k^T = [q_k^{(1)}, \dots, q_k^{(N)}]$ where $q_k^{(i)} = [q_k^{(i1)}, \dots, q_k^{(iN)}]^T$. Observe that

- (i) $Q_k e_i = q_k^{(i)}$; and
- (ii) $e_i^T Q_k e_j = q_k^{(ij)}$.

The DS Based Algorithm (4) is stated in Table 9. The computational complexity depends on the number of multiplications required to update Q_k . In Table 10, we list the number of multiplications needed at each step for a FIR filter. The computational complexity is $3N + 2$.

The second one is suggested in [3]. Let a block of N error squares be given the same weight $\lambda_i^{(kN+j)} = \lambda^j$, for $j = 1, \dots, N$. Then the ALS problems become:

$$\min_{x \in R^N} J_n(x) = \sum_{i=0}^{k-1} \lambda^{k-i} \left[\sum_{l=1}^N (a_{iN+l}^T x - s_{iN+l})^2 \right] + \sum_{l=1}^j (a_{kN+l}^T x - s_{kN+l})^2 \quad (11)$$

where $n = kN + j$, k is a nonnegative integer, $1 \leq j \leq N$ and $0 \leq \lambda \leq 1$. When $\lambda = 1$, the minimization problems in (11) are equivalent to the total least

Table 9: The DS Based Algorithm (4)

<p>Given an initial estimate $x_0^{(0)}$, scalars $0 < \epsilon \leq 1$ and $\tau > 0$, matrix Q_0, and vector B_0, set $Q = Q_0 = [q^{(1)}, \dots, q^{(N)}]$, $B = B_0 = [b^{(1)}, \dots, b^{(N)}]^T$, and $x = x_0^{(0)}$, and repeat Step I and Step II for $k = 1, \dots$.</p>	
Step I	<p>For $1 \leq i \leq N$, compute</p> <ol style="list-style-type: none"> (1) $\alpha = -\frac{(q^{(i)})^T x - b^{(i)}}{q^{(ii)}}$; (2) $(x)_i = (x)_i + \alpha$; (3) $\alpha_i = \alpha^2$; (4) $Q = Q + a_k a_k^T$; (5) $B = B + s_k a_k$;
Step II	<p>If $\beta = \sqrt{\alpha_1 + \dots + \alpha_N} < \tau$, then terminate the algorithm and $x^* \approx x$.</p>

Table 10: # of Multiplications for the DS Based Algorithm (4)

$(x)_i, 1 \leq i \leq N$	
Step	# of multiplications
α	$N + 1$
$(x)_i$	0
α_i	1
Q	N (FIR filter)
B	N
Total	$3N + 2$

Table 11: The DS Based Algorithm (5)

<p>Given an initial estimate $x_0^{(0)}$, scalars $0 < \epsilon \leq 1$ and $\tau > 0$, matrix Q_0, and vector B_0, set $\bar{Q} = [\bar{q}^{(1)}, \dots, \bar{q}^{(N)}]$ to a zero matrix, $\bar{B} = [\bar{b}^{(1)}, \dots, \bar{b}^{(N)}]^T$ to a zero vector, $Q = Q_0 = [q^{(1)}, \dots, q^{(N)}]$, $B = B_0 = [b^{(1)}, \dots, b^{(N)}]^T$, and $x = x_0^{(0)}$, and repeat Step I and Step II for $k = 0, 1, \dots$.</p>	
Step I	<p>For $1 \leq i \leq N$, compute</p> <ol style="list-style-type: none"> (1) $\alpha = -\frac{(q^{(i)} + \bar{q}^{(i)})^T x - (b^{(i)} + \bar{b}^{(i)})}{q^{(ii)} + \bar{q}^{(ii)}}$; (2) $(x)_i = (x)_i + \alpha$; (3) $\alpha_i = \alpha^2$; (4) $q^{(i)} = \lambda q^{(i)}$, $\bar{Q} = \bar{Q} + a_{kN+i} a_{kN+i}^T$; (5) $b^{(i)} = \lambda b^{(i)}$, $\bar{B} = \bar{B} + s_{kN+i} a_{kN+i}$;
Step II	<p>If $\beta = \sqrt{\alpha_1 + \dots + \alpha_N} < \tau$, then terminate the algorithm and $x^* \approx x$. Otherwise, $Q = Q + \bar{Q}$, $B = B + \bar{B}$, set \bar{Q} to a zero matrix and \bar{B} to a zero vector.</p>

squares problems in (3). When $\lambda = 0$, the objective function is the same one used by the block CG algorithm.

The DS Based Algorithm (5) is stated in Table 11. The computational complexity also depends on the number of multiplications required to update Q_k . In Table 12, we list the number of multiplications needed at each step for a FIR filter. The computational complexity is $4N + 3$.

Table 12: # of Multiplications for the DS Based Algorithm (5)

$(x)_i, 1 \leq i \leq N$	
Item	# of multiplications
α	$N + 1$
$(x)_i$	0
α_i	1
$q^{(i)}, Q$	$2N$ (FIR filter)
$b^{(i)}, B$	$N + 1$
Total	$4N + 3$

4.2 Near Conjugate Direction Sets

Let $D_l = [d_l^{(1)}, \dots, d_l^{(N)}]$ for some $l \geq 0$ such that $\text{rank}(A_l) = N$. Then $Q_l = A_l A_l^T$ is positive definite. If $d_l^{(i)}$'s are conjugate with respect to Q_l , i.e., $(d_l^{(i)})^T Q_l d_l^{(j)} = 0$, for $i \neq j$ and $1 \leq i, j \leq N$, then $x_l^{(N)} = x^*$ when input and output signals are without noise. This can be seen as follows. Let $D_l^T Q_l D_l = \bar{D}$ where $\bar{D} = \text{diag}\{(d_l^{(1)})^T Q_l d_l^{(1)}, \dots, (d_l^{(N)})^T Q_l d_l^{(N)}\}$. Since $d_l^{(i)}$'s are also linearly independent, D_l is nonsingular and $Q_l D_l = D_l^{-T} \bar{D}$. Since $(d_l^{(i)})^T Q_l d_l^{(j)} = 0$, for $i \neq j$,

$$\begin{aligned} (d_l^{(i+1)})^T (Q_l x_l^{(i)} - B_l) &= (d_l^{(i+1)})^T (Q_l (x_l^{(0)} + \sum_{j=1}^i \alpha_l^{(j)} d_l^{(j)}) - B_l) \\ &= (d_l^{(i+1)})^T (Q_l x_l^{(0)} - B_l), \quad \text{for } i = 0, \dots, N-1. \end{aligned}$$

Let $\alpha = [\alpha_l^{(1)}, \dots, \alpha_l^{(N)}]^T$. Then $\alpha = -(D_l \bar{D}^{-1})^T (Q_l x_l^{(0)} - B_l)$ and $x_l^{(N)} = x_l^{(0)} + D_l \alpha$. Since

$$\begin{aligned} Q_l x_l^{(N)} &= Q_l x_l^{(0)} - Q_l D_l (D_l \bar{D}^{-1})^T (Q_l x_l^{(0)} - B_l) \\ &= Q_l x_l^{(0)} - (D_l)^{-T} \bar{D} \bar{D}^{-1} D_l^T (Q_l x_l^{(0)} - B_l) \\ &= Q_l x_l^{(0)} - Q_l x_l^{(0)} + B_l = B_l, \end{aligned}$$

$x_l^{(N)} = x^*$.

One of the ways to choose a set of conjugate directions with respect to Q_l is letting $D_l^T = A_l^{-1}$ because $D_l^T Q_l D_l = I$. The computation of A_l^{-1} can be very expensive when N is large. However, this promotes the idea to choose a set of near conjugate directions such that $D_l^T \approx A_l^{-1}$. When A_l is a Toeplitz matrix, the optimal circulant preconditioning techniques given in [7, 31] can be adopted to approximate A_l^{-1} . Let $D_l^{(Chan)}$ be Chan's optimal circulant matrix obtained from A_l and $D_l^{(Tyrtyshnikov)}$ be Tyrtyshnikov's optimal circulant matrix obtained from A_l , i.e., $\|D_l^{(Chan)} - A_l\|_F$ and $\|I - D_l^{(Tyrtyshnikov)} A_l\|_F$ are minimized over all circulant matrices.

Another way to generate a set of N conjugate directions $d_l^{(1)}, \dots, d_l^{(N)}$ is to carry out N CG steps with a block of N pairs of input and desired output signals. Then use these N conjugate directions as the initial directions and the last estimate $x_l^{(N)}$ as the initial estimate for the DS based algorithm. The computational complexity is $O(N^2)$ for each CG iteration. However, since this is done only once at the initialization step, the overall performance of the algorithm can be computationally more efficient.

A computational procedure is designed to generate a set of near conjugate directions with respect to Q_l by using Chan's optimal circulant matrix, Tyrtyshnikov's optimal circulant matrix, or the CG algorithm. The DS based algorithm with this procedure is applied to applications in system identification.

Table 13: Coefficients for the Bilinear System of Order 5

c_i	a_i	b_{0i}	b_{1i}	b_{2i}	b_{3i}
0.4940	-0.0263	0.0734	-0.0152	-0.0788	0.1235
0.1016	-0.0121	-0.0077	-0.0555	0.1091	0.0479
0.0177	0.0932	-0.0539	0.0198	0.0061	-0.0528
-0.085	0.2355	0.0589	0.0449	-0.0980	-0.0700
	0.1785	0.1123	-0.0263	-0.0646	0.1103

Computer simulations given in [8] have shown that the set of near conjugate directions generated by the CG algorithm is the best choice and the set of near conjugate directions obtained by Tyrtyshnikov’s optimal circulant matrix is the second best choice.

5 Implementation and Applications

The DS based algorithm has been implemented in MatLab and applied to solve ALS problems arising in system identification and adaptive equalization. The DS Based Algorithm (3) with a new starting estimate at each searching cycle is used. One system update is computed for each oncoming pair of input and desired output signals but the objective function is updated once for each searching cycle. We also test the algorithm for all applications with updating the objective function for each oncoming input and desired output signals and results have shown that the rate of convergence is not affected by less frequently updating the objective function. If the system changes rapidly, we suggest to use the DS Based Algorithm (4) or (5). In all computer simulations, the zero vector is used as the initial estimate $x_0^{(0)}$.

5.1 System Identification

Consider the bilinear system of order 5:

$$y(n) = \sum_{i=1}^4 c_i y(n-i) + \sum_{i=0}^4 \sum_{j=1}^4 b_{i,j} y(n-j) u(n-i) + \sum_{i=0}^4 a_i u(4-i)$$

where $y(n)$ and $u(n)$ are the output and the input of the system respectively. Coefficients c_i , $b_{i,j}$ and a_i are given in Table 13. A zero-mean white normally distributed sequence with variance 0.05 is generated for input and a white Gaussian sequence is then added to generate the desired output signals. The DS based algorithm (3) with a new starting estimate at each searching cycle and $\lambda = 1$ are used to identify this bilinear system and it reaches a steady state error of 10^{-8} in about 1,500 iterations. As reported in [2], to reach the same steady state error,

Table 14: Comparisons of Computational Cost - System Identification

algorithm	# of iterations	# of multiplications
RLS	300	1,044,000
Block CG (block size N)	100	359,000
DS Based Algorithm (3)	1500	282,000

the RLS required about 300 iterations and the CG with a block of N data pairs required about 100 iterations. By comparing the numbers of iterations, it seems that the DS based algorithm is far slower than the CG based algorithm and the RLS algorithm. However, the DS based algorithm uses 282,000 multiplications to well identify the system while the CG based algorithm requires 359,000 multiplications and the RLS algorithm requires 1,044,000 multiplications. Comparisons are listed in Table 14. So, by comparing overall performances of these algorithms, the DS based algorithm is still computationally more efficient.

5.2 Adaptive Equalizer

In this section, a transversal adaptive equalizer is considered for equalizing the distortion introduced in a band-limited channel. This is the same example as used in [1] and originally published in [20]. The input is a white gaussian sequence which is distorted by a channel with impulse response $h(i) = \frac{1}{2}(1 + \cos(2\pi(i-2)/W))$, $i = 1, 2, 3$ and $h(i) = 0$ otherwise, where W controls the amplitude distortion in the channel. An increase in W increases channel distortion. An additive white gaussian sequence $v(i)$ is introduced in the channel. A 11-tap FIR filter is used as the equalizer model as in [1]. The channel input is delayed appropriately to form the desired output of the equalizer. The experiment is conducted with $W = 2.9$ and $W = 3.5$ and a SNR of 30dB. The parameter W controls the eigenvalue spread $X(R)$ of the correlation matrix of the inputs to the equalizer. The above values of W correspond to eigenspread $X(R)$ of 6.07 and 46.8 respectively. The DS Based Algorithm (3) with a new starting estimate at each searching cycle is used with $\lambda = 0.99$ to adapt the filter coefficients and an ensemble average of 200 runs are taken. A steady state error (the mean squared error (MSE)) of 10^{-2} is reached within 150 iterations for $X(R) = 46.8$ and a MES of 10^{-3} is reached within 100 iterations for $X(R) = 6.07$. Numbers of iterations and corresponding steady state MSEs needed by the RLS algorithm [20] and the block CG algorithm with a block size 3 [1] to solve the same problem are listed in Table 15. The block CG algorithm with a small block size reaches a higher MSE. For both examples, the DS based algorithm converges faster than other two algorithms.

Table 15: Comparisons of Convergence - Adaptive Equalizer

	$X(R) = 6.07 (W = 2.9)$		$X(R) = 46.8 (W = 3.5)$	
algorithm	# of iterations	MSE	# of iterations	MSE
RLS	50	10^{-3}	50	10^{-2}
Block CG (size=3)	60	10^{-2}	60	10^{-1}
DS Based Alg. (3)	100	10^{-3}	150	10^{-2}

6 The DS Based Algorithm for Spectral Estimation

Spectral estimation using an adaptive version of Pisarenko's harmonic retrieval method was suggested in [30]. Pisarenko's harmonic retrieval method involves finding the minimum eigenvalue λ_{\min} and its associated eigenvector x_{\min} of the correlation matrix R , and then computing the roots of the polynomial whose coefficients are the components of x_{\min} . In [21], a least-squares type of such adaptive algorithm was proposed. Recent work [5, 32, 15] on adaptive spectral estimation has shown that the CG based algorithm appears to be one of the most suitable descent method to iteratively seek the minimum eigenvalue and associated eigenvector of a symmetric matrix. In [15], a CG based algorithm for spectral estimation was presented by solving a constrained minimization problem:

$$\begin{aligned} \min_{x \in R^N} \quad & x^T Q_n x \\ \text{s.t.} \quad & x^T x = 1 \end{aligned} \quad (12)$$

where Q_n is the correlation matrix of the samples and x is the unit norm eigenvector of the matrix Q_n associated with its minimum eigenvalue. Authors also recast CG into an unconstrained minimization problem:

$$\min_{x \in R^N} f_n(x) = \frac{x^T Q_n x}{2} + \frac{\mu(x^T x - 1)^2}{4} \quad (13)$$

where μ is a constant such that $\mu \geq \text{trace}(Q_n)/2$ as given in [17]. It was shown in [15] that the rate of convergence of the CG based algorithm is comparable to that of the least-square type algorithm in [21], while being computationally more efficient.

The DS based algorithm is applied to solve the adaptive unconstrained minimization problem in (13). Note that Q_n has the same structure as the Hessian matrix for the ALS problems and its structure is preserved by the DS based algorithm. Note also that since f_n is no longer a quadratic function, the minimization problem:

$$\min_{\alpha \in R} f_n(x_n^{(i-1)} + \alpha d_n^{(i)}) \quad (14)$$

cannot be solved exactly in general. So, the stepsize $\alpha_n^{(i)}$ along the direction $d_n^{(i)}$ cannot be computed exactly in general. To well approximate the solution of the minimization problem in (14), we first approximate $f_n(x_n^{(i-1)} + \alpha d_n^{(i)})$ by its quadratic approximation

$$F(x_n^{(i-1)} + \alpha d_n^{(i)}) = \frac{1}{2}(x_n^{(i-1)} + \alpha d_n^{(i)})^T \nabla^2 f_n(x_n^{(i-1)})(x_n^{(i-1)} + \alpha d_n^{(i)}) + (x_n^{(i-1)} + \alpha d_n^{(i)})^T \nabla f_n(x_n^{(i-1)}) + f_n(x_n^{(i-1)}),$$

and then solve the quadratic minimization problem exactly:

$$\min_{\alpha \in \mathbb{R}} F(x_n^{(i-1)} + \alpha d_n^{(i)}). \quad (15)$$

Stepsize $\alpha_n^{(i)}$ is approximated by the solution of the minimization problem in (15) and

$$\alpha_n^{(i)} = -\frac{(Q_n x_n^{(i-1)} + \mu((x_n^{(i-1)})^T x_n^{(i-1)} - 1)x_n^{(i-1)})^T d_n^{(i)}}{(d_n^{(i)})^T (Q_n + \mu((x_n^{(i-1)})^T x_n^{(i-1)} - 1)I + 2\mu(x_n^{(i-1)})^T x_n^{(i-1)})d_n^{(i)}}.$$

When $d_k^{(i)} = e_i$ for all k , $\alpha_n^{(i)}$ can be simplified as

$$\alpha^{(i)} = -\frac{(Q_n x^{(i-1)} + \mu((x^{(i-1)})^T x^{(i-1)} - 1)x^{(i-1)})^T e_i}{e_i^T (Q_n + \mu((x^{(i-1)})^T x^{(i-1)} - 1)I + 2\mu(x^{(i-1)})^T x^{(i-1)})e_i}.$$

When $d_k^{(i)} = e_i$ for all k and $Q_n = [q_n^{(1)}, \dots, q_n^{(N)}]$ is updated for each oncoming input and desired output signals, $\alpha_n^{(i)}$ can be computed as

$$\alpha^{(i)} = -\frac{(q_n^{(i)} + \bar{q}_n^{(i)})^T x + \mu(x)_i(x^T x - 1)}{q_n^{(ii)} + \bar{q}_n^{(ii)} + \mu(x^T x - 1) + 2\mu((x)_i)^2}.$$

The modified DS Based Algorithm (5) is applied to solve spectral estimation problem given in [15].

Consider a signal composed of two sinusoids with normalized frequencies of 0.3 and 0.4 corrupted with 10dB of white Gaussian noise. A 6th-order FIR filter is chosen as the system model. Let $\lambda = 0.99$, $\sigma = 0.001$. The ensemble average of 100 simulations are performed. Simulation results are plotted in Figure 1. Fig. 1(a) and 1(b) give the frequency estimates obtained using the DS based algorithm with $\mu = \mu_k = 10000(1 - \lambda^k)$. Fig. 1(c) and 1(d) give the frequency estimates obtained using the CG based algorithm with $\mu = 10000$. The rates of convergence are comparable for these algorithms. However, the estimates are smoother in the case of the DS based algorithm than those in the case of the CG based algorithm. The mean and standard deviation of the frequency estimate errors are listed in Table 16. Both the mean and the variance of the errors of

Figure 1: Comparisons of Performance - Spectral Estimation

Table 16: Summary on the power spectral density estimates

algorithm	DS Based Algorithm		CG based Algorithm	
Frequency	f_1	f_2	f_1	f_2
Mean	0.0428×10^{-4}	0.0486×10^{-4}	0.4191×10^{-4}	0.4382×10^{-4}
STD	0.0067×10^{-3}	0.0062×10^{-3}	0.2127×10^{-3}	0.1660×10^{-3}

frequency estimates obtained by the DS based algorithm are significantly lower than those obtained by the CG based algorithm.

Acknowledgments: Author would like to thank colleagues Tamal Bose and Guo Fang Xu at the Department of Electrical Engineering of University of Colorado at Denver for their collaboration in the development of the algorithms presented in this paper.

References

- [1] G. Boray and M. Srinath, "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. Circuits Syst.*, vol. CAS-39, Jan. (1992), pp. 1-10.
- [2] T. Bose and M.-Q. Chen, "Conjugate Gradient Method in Adaptive Bilinear Filtering", *IEEE Transactions on Signal Processing*, Vol. 43, No. 6 (1995), pp. 1503-1508.
- [3] T. Bose, M.-Q. Chen and G.-F. Xu, "A Fast Adaptive Algorithm for Spectral Estimation", *Proceeding of IEEE PACRIM'97*, Vol.1, pp. 146-149.
- [4] K. W. Brodlie, "A new direction set methods for unconstrained minimization without evaluating derivatives," *J. Inst. Maths. Applns.*, 15 (1975), pp.385-396.
- [5] H. Chen, T. K. Sarkar, S. A. Dianat, and J. D. Brule, "Adaptive spectral estimation by the conjugate gradient method", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP_34, pp. 272-284, April, 1986.
- [6] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. Assp-31, No. 6, December (1983), pp. 1394-1402.
- [7] T. Chan, "An optimal circulant preconditioner for Toeplitz systems," *SIAM J. Sci. Stat. Computing*, Vol. 9, No. 4, July (1988), pp. 766-771.
- [8] M.-Q. Chen, "A Direction Set Based Algorithm for Least Squares Problems in Adaptive Signal Processing", *Linear Algebra and Its Applications*, Vol. 284, No. 1-3, November 1998, pp. 73-94.

- [9] M.-Q. Chen, T. Bose and G. F. Xu, "A Direction Set Based Method for Adaptive Filtering," *IEEE Transactions on Signal Processing*, Vol. 47, No. 2, February 1999.
- [10] R. Chan, J. Nagy and R. Plemmons, "FFT-based preconditioners for Toeplitz-block least squares problems", *SIAM J. Numerical Analysis*, 2, 1994, pp. 1740-1768.
- [11] R. Chan, M. Ng and R. Plemmons, "Generalization of Strang's preconditioner with applications to Toeplitz least squares problem," *Numerical Linear Algebra and Applications*, Vol. 3 (1996), pp. 45-64.
- [12] R. Chan, J. Nagy and R. Plemmons, "Displacement Preconditioner for Toeplitz Least Squares Iterations," preprint.
- [13] R. Chan, J. Nagy and R. Plemmons, "Circulant Preconditioned Toeplitz Least Squares Iterations," preprint.
- [14] P. S. Chang and A. N. Willson, Jr., "Adaptive Filtering Using Modified Conjugate Gradient," *Proc. 38th Midwest Symposium on Circuits and Systems*, Rio de Janeiro, pp. 243-246, Aug. 1995.
- [15] P. S. Chang and A. N. Willson, Jr., "Adaptive Spectral Estimation Using the Conjugate Gradient Algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, pp. 2979-2982, May. 1996.
- [16] P. S. Chang and A. N. Willson, Jr., "Analysis of Conjugate Gradient Algorithms for Adaptive Filtering," *Proc. IEEE Int. Symposium on Circuits and Systems*, Hong Kong, pp. 2292-2295, Jun. 1997.
- [17] G. Mathew, V. U. Reddy and S. Dasgupta, "Adaptive estimation of eigensubspace," *IEEE Trans. on Signal Processing*, vol. 43, pp. 401-411, Feb. 1995.
- [18] R. Fletcher, *Practical Methods of Optimization*, Second Edition, John Wiley and Sons Ltd., 1987.
- [19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins, 1985.
- [20] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, NJ, 3rd Edition, 1996.
- [21] V. U. Reddy, B. Edgardt, and T. Kailath, "Least-squares type algorithm for adaptive implementation of Pisarenko's harmonic retrieval method," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 399-405, Jun. 1982.

- [22] L. Ljung, M. Morf and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Int. J. Control*, Jan. (1978), pp. 1-19.
- [23] G.V. Moustakides and S. Theodoridis, "Fast Newton transversal filters - A new class of adaptive estimation algorithms," *IEEE Trans. on Signal Processing*, vol. SP-39, Oct. 1991, pp. 2184-2193.
- [24] Michael K. Ng and Robert Plemmons, "Fast RLS Adaptive Filtering by FFT-based Conjugate Gradient Iteration," *SIAM J. on Scientific Computing*, Vol. 17, July, 1996, pp. 920-941.
- [25] Michael K. Ng and Robert Plemmons, "LMS-Newton Adaptive Filtering using FFT-based Conjugate Gradient Iterations," *Electronic J. Numer. Anal.*, Vol. 4, 1996, pp. 14-36.
- [26] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, 7 (1964), pp. 155-162.
- [27] S. Qiao, "Fast adaptive RLS algorithms: a generalized inverse approach and analysis," *IEEE Transactions on Signal Processing*, Vol. 39, June 1991, pp. 1455-1459.
- [28] D. T. M. Slock and T. Kailath, "Numerically stable fast recursive least-squares transversal filters," *Proc. ICASSP*, 1988, pp. 1365-1368.
- [29] H. A. Spang, "A review of minimization techniques for nonlinear functions," *SIAM Review*, Vol. 4 (1962), pp. 343-365.
- [30] P. A. Thompsom, "An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise", *Proc. 13th Asilomar Conf. on Circuits, Syst. and Comp.*, Pacific Grove, Calif., pp. 529-533, Nov. 1979.
- [31] E. E. Tyrtshnikov, "Optimal and superoptimal circulant preconditioners," *SIAM J. Matrix Analysis and Applications*, Vol. 13, No. 2, pp. 459-473, April 1992.
- [32] X. Yang, T.K. Sarkar and E. Arvas, "A survey of Conjugate Gradient algorithms for solution of extreme eigen-problems of a symmetric matrix," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 1550-1556, Oct. 1989.
- [33] W. Zangwill, "Minimizing a function without calculating derivatives," *The Computer Journal*, 10 (1967), pp. 293-296.