Conjugate Gradient Projection Subspace Tracking

Zuqiang Fu and Eric M. Dowling

The University of Texas at Dallas, EC33
Erik Jonsson School of Engineering and Computer Science
Richardson, TX 75083–0688
(214) 690–2784 emd@utdallas.edu Fax: (214) 690–2710

In this paper we present a conjugate gradient subspace projection algorithm for subspace tracking. We base the algorithm on the subspace projection cost function recently introduced by B. Yang, and use a conjugate gradient iteration to provide rapid convergence. The update complexity is $O(r^2m)$ where r is the dimension of the tracked subspace which is assumed to be much smaller than m. The algorithm has immediate application in radar and sonar array signal processing systems.

I. Introduction

There has been a surge of interest recently in algorithms that track the dominant subspace associated with time-varying data and data correlation matrices. These time-varying matrices arise in frequency and direction of arrival tracking problems, as discussed in [1–4] and the references therein. The time-varying subspaces are used with eigenstructure based high resolution angle and frequency estimation algorithms such as MUSIC and Minimum Norm.

As discussed in [5], conjugate gradient methods have been employed to quickly converge to the dominant or subdominant *eigenvector* of a time-varying data or data correlation matrix. More recently, Fu and Dowling have developed a conjugate gradient algorithm that rapidly converges to the dominant or subdominant *subspace* [6]. In this paper, we develop a new conjugate gradient algorithm based on the projection cost function recently introduced by B. Yang [2, 3]. More details concerning this conjugate gradient subspace tracking algorithm as well as a systolic array implementation appear in [7, 8]. The algorithm we present uses the deflation concept presented by B. Yang and others, as well as a subspace averaging based deflation technique [9]. The algorithm has complexity $O(r^2m)$ where r is the dimension of the tracked subspace which is assumed to be much smaller than m. The algorithm is a bit more expensive than the algorithm of B. Yang, but is on the same order of complexity if reorthogonalization is needed as is the case in practice. The deflated conjugate gradient projection algorithm demonstrates faster convergence than the PASTd algorithm although at a slightly higher cost.

This research was supported in part by the NSF Grant MIP-9203296 and the Texas Adv. Tech. Prog. Grant 009741-022.

II. The Cost Function

As proposed by Yang [2, 3], we will start with the cost function,

$$J(\mathbf{U}_s(n)) = \sum_{i=1}^{n} \beta^{n-i} \left\| \mathbf{x}(i) - \mathbf{U}_s(n)\mathbf{U}_s^H(n)\mathbf{x}(i) \right\|^2. \tag{1}$$

Note that this RLS style cost function has as its argument $\mathbf{U}_s(n) \in \mathbf{C}^{m\times r}$ whose columns are assumed to be orthonormal. The cost function will be minimized when the columns of $\mathbf{U}_s(n)$ span the dominant subspace of the data correlation matrix, $\mathbf{R} = \sum_{i=1}^{n} \beta^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)$. Our aim is to provide a recursive algorithm that computes $\mathbf{U}_s(n)$ given $\mathbf{U}_s(n-1)$ and the new data vector, $\mathbf{x}(n)$. To do this, we will break the subspace cost function into vector cost functions that can be collectively minimized using conjugate gradient iterations. First we expand the cost function according to

$$J(\mathbf{U}_s(n)) = \sum_{i=1}^{n} \beta^{n-i} \| \mathbf{x}(i) - \mathbf{u}_1(n)\mathbf{u}_1^H(n)\mathbf{x}(i) - \cdots - \mathbf{u}_r(n)\mathbf{u}_r^H(n)\mathbf{x}(i) \|^2. \tag{2}$$

If we define

$$\begin{aligned} \mathbf{y}_j(i,n) &= \mathbf{x}(i) - \sum_{l=1}^{j-1} \mathbf{u}_l(n)\mathbf{u}_l^H(n)\mathbf{x}(i) \\ &= \mathbf{y}_{j-1}(i,n) - \mathbf{u}_{j-1}(n)\mathbf{u}_{j-1}^H(n)\mathbf{x}(i), \end{aligned} \tag{3}$$

then $\mathbf{y}_j(i,n)$ represents the projection of $\mathbf{x}(i)$ onto the orthogonal complement of the span of the first $j-1$ columns of $\mathbf{U}_s(n)$. We can then define the j^{th} cost function as

$$J(\mathbf{u}_j(n)) = \sum_{i=1}^{n} \beta^{n-i} \left\| \mathbf{y}_j(i,n) - \mathbf{u}_j(n)\mathbf{u}_j^H(n)\mathbf{x}(i) \right\|^2. \tag{4}$$

Using these definitions, the j^{th} eigenvector in the signal subspace can be obtained by minimizing $J(\mathbf{u}_j(n))$.

III. Conjugate gradient projection signal subspace tracking

To set up the problem, we can manipulate the j^{th} cost function to read,

$$J(\mathbf{u}_j(n)) = \sum_{i=1}^{n} \beta^{n-i} \left\| \mathbf{y}_j(i,n) - \mathbf{u}_j(n)\mathbf{u}_j^H(n)\mathbf{x}(i) \right\|^2 \quad (5)$$
$$= \sum_{i=1}^{n} \beta^{n-i} \mathbf{y}_j^H(i,n)\mathbf{y}_j(i,n) -$$
$$\mathbf{u}_j^H(n)(\mathbf{R}_{\mathbf{x}\mathbf{y}_j}(n) + \mathbf{R}_{\mathbf{y}_j\mathbf{x}}(n) - \mathbf{R}_{\mathbf{x}\mathbf{x}}(n))\mathbf{u}_j(n),$$

where

$$\mathbf{R}_{\mathbf{x}\mathbf{x}}(n) = \sum_{i=1}^{n} \beta^{n-i}\mathbf{x}(i)\mathbf{x}^H(i) \quad (6)$$

$$\mathbf{R}_{\mathbf{x}\mathbf{y}_j}(n) = \sum_{i=1}^{n} \beta^{n-i}\mathbf{x}(i)\mathbf{y}_j^H(i,n)$$
$$= \mathbf{R}_{\mathbf{x}\mathbf{x}}(n) - \mathbf{R}_{\mathbf{x}\mathbf{x}}(n)\sum_{l=1}^{j}\mathbf{u}_l\mathbf{u}_l^H$$

$$\mathbf{R}_{\mathbf{y}_j\mathbf{x}}(n) = \sum_{i=1}^{n} \beta^{n-i}\mathbf{y}_j(i,n)\mathbf{x}^H(i)$$
$$= \mathbf{R}_{\mathbf{x}\mathbf{y}_j}^H(n).$$

If we define $\mathbf{R}_j(n) = \mathbf{R}_{\mathbf{x}\mathbf{y}_j}(n) + \mathbf{R}_{\mathbf{y}_j\mathbf{x}}(n) - \mathbf{R}_{\mathbf{x}\mathbf{x}}(n)$ and $Y_j(n) = \sum_{i=1}^{n} \beta^{n-i}\mathbf{y}_j^H(i,n)\mathbf{y}_j(i,n)$, then the j^{th} cost function becomes

$$J(\mathbf{u}_j(n)) = Y_j(n) - \mathbf{u}_j^H(n)\mathbf{R}_j(n)\mathbf{u}_j(n). \quad (7)$$

These functions can be minimized by maximizing

$$\tilde{J}(\mathbf{u}_j(n)) = \mathbf{u}_j^H(n)\mathbf{R}_j(n)\mathbf{u}_j(n). \quad (8)$$

using Sarkar's conjugate gradient iterations, [10, 7]. To update all r basis vectors of the subspace, the complexity becomes $O(rm^2)$. In the updating process, we initialize the iterations with the previous estimates, that is, we start with $\mathbf{u}_j(n-1)$ and take a conjugate gradient step to obtain $\mathbf{u}_j(n)$. We also orthogonalize $\mathbf{u}_j(n)$ against the previous vectors, $\mathbf{u}_1(n), \mathbf{u}_2(n), \cdots, \mathbf{u}_{j-1}(n)$. We find in practice that one conjugate gradient iteration, k, per vector per time update, n, is needed. We call this algorithm the Conjugate Gradient Projection Subspace Tracker (CG-PST), but proceed to develop a deflated algorithm with lower complexity.

IV. Deflated conjugate gradient projection subspace tracking

In practice, we often find $r << m$ and hence we would much rather have an $O(r^2m)$ algorithm than an $O(rm^2)$ algorithm [1]. We will accomplish this reduction using the concept of subspace averaging [9]. We first compute the normalized projection of the new data vector $\mathbf{x}(n)$ onto the old noise subspace according to

$$\mathbf{u}_{r+1}'(n-1) = \frac{\mathbf{x}(n) - \mathbf{U}_s(n-1)\mathbf{U}_s^H(n-1)\mathbf{x}(n)}{\|\mathbf{x}(n) - \mathbf{U}_s(n-1)\mathbf{U}_s^H(n-1)\mathbf{x}(n)\|_2}, \quad (9)$$

where $\mathbf{U}_s(n-1) = [\mathbf{u}_1(n-1), \mathbf{u}_2(n-1), \cdots \mathbf{u}_r(n-1)]$. We then define $\mathbf{U}'(n-1) = \left[\mathbf{u}_1(n-1), \cdots, \mathbf{u}_r(n-1), \mathbf{u}_{r+1}'(n-1)\right] \in \mathbb{C}^{m\times(r+1)}$ so that the columns of $\mathbf{U}'(n-1)$ span the $(r+1)$–dimensional subspace that contains $\mathbf{x}(n)$. The cost functions of (4) can be dimension-reduced using $\mathbf{U}'^H(n-1)$,

$$J(\mathbf{u}_j(n)) = \sum_{i=1}^{n} \beta^{n-i} \left\| \mathbf{y}_j(i,n) - \mathbf{u}_j(n)\mathbf{u}_j^H(n)\mathbf{x}(i) \right\|^2 \quad (10)$$
$$\approx \sum_{i=1}^{n} \beta^{n-i} \|\mathbf{U}'^H(n-1)(\mathbf{y}_j(i,n)$$
$$- \mathbf{u}_j(n)\mathbf{u}_j^H(n)\mathbf{x}(i))\|^2$$
$$= \sum_{i=1}^{n} \beta^{n-i} \|\mathbf{U}'^H(n-1)\mathbf{y}_j(i,n) -$$
$$\mathbf{U}'^H(n-1)\mathbf{u}_j(n)\mathbf{u}_j^H(n)\mathbf{x}(i)\|^2.$$

This approximation comes from the way we defined $\mathbf{U}'(n-1)$, *i.e.*, we have $\mathbf{x}(i) = \mathbf{U}'(i-1)\mathbf{U}'^H(i-1)\mathbf{x}(i)$. If we assume that the subspaces are slowly time-varying, we can approximate

$$\mathbf{x}(i,n) = \mathbf{U}'(n-1)\mathbf{U}'^H(n-1)\mathbf{x}(i) \approx \mathbf{x}(i). \quad (11)$$

This approximation will greatly simplify the mathematics to reduce computations, but will not affect performance significantly as our simulations demonstrate. The basic idea is that we will update $\mathbf{U}'(n-1)$ according to

$$\mathbf{U}(n) = \mathbf{U}'(n-1)\mathbf{Q}(n), \quad (12)$$

where $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_r, \mathbf{q}_{r+1}] \in \mathbb{C}^{(r+1)\times(r+1)}$ is a orthonormal matrix whose columns solve lower dimensional constrained optimization problems. During the $(n+1)^{st}$ update $\mathbf{U}(n)$ will be altered to $\mathbf{U}'(n)$ by replacing the $(r+1)^{st}$ column of $\mathbf{U}(n)$ with $\mathbf{u}_{r+1}'(n)$ computed according to (9). For now, we will focus on the n^{th} update.

In order to reduce the m–dimensional problem to an $(r+1)$–dimensional problem, we can now write the second term in the last line of (10) as

$$\mathbf{U}'^{H}(n-1)\mathbf{u}_j(n)\mathbf{u}_j^H(n)\mathbf{U}'(n-1)\mathbf{U}'^{H}(n-1)\mathbf{x}(i) \tag{13}$$

or,

$$\mathbf{q}_j(n)\mathbf{q}_j^H(n)\tilde{\mathbf{x}}(i,n) \tag{14}$$

where

$$\begin{aligned}\tilde{\mathbf{x}}(i,n) &= \mathbf{U}'^{H}(n-1)\mathbf{x}(i) \\ \mathbf{q}_j(n) &= \mathbf{U}'^{H}(n-1)\mathbf{u}_j(n).\end{aligned} \tag{15}$$

Note that $\tilde{\mathbf{x}}(i,n)$ is a reduced dimensioned input, and $\mathbf{q}_j(n)$ is a reduced dimensioned version of $\mathbf{u}_j(n)$. Along these lines, next define

$$\tilde{\mathbf{y}}_j(i,n) = \mathbf{U}'^{H}(n-1)\mathbf{y}_j(i,n), \tag{16}$$

so that we can plug (14), (15) and (16) in the cost function expression (10), to obtain the reduced dimensional cost function,

$$J(\mathbf{u}_j(n)) \approx J(\mathbf{q}_j(n)) = \sum_{i=1}^{n} \beta^{n-i} \left\| \tilde{\mathbf{y}}_j(i,n) - \mathbf{q}_j(n)\mathbf{q}_j^H(n)\tilde{\mathbf{x}}(i,n) \right\|^2. \tag{17}$$

To set up for the recursive solution of (17), expand this by taking the innerproduct of the normed vector to write

$$J(\mathbf{q}_j(n)) = \tilde{Y}_j(n) - \mathbf{q}_j^H(n)\tilde{\mathbf{R}}_j(n)\mathbf{q}_j(n), \tag{18}$$

where

$$\tilde{\mathbf{R}}_j(n) = \tilde{\mathbf{R}}_{\mathbf{x}\mathbf{y}_j}(n) + \tilde{\mathbf{R}}_{\mathbf{y}_j\mathbf{x}}(n) - \tilde{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(n). \tag{19}$$

and

$$\begin{aligned}\tilde{Y}_j(n) &\triangleq \sum_{i=1}^{n} \beta^{n-i}\tilde{\mathbf{y}}_j^H(i,n)\tilde{\mathbf{y}}_j(i,n) \\ \tilde{\mathbf{R}}_{\mathbf{x}\mathbf{y}_j}(n) &\triangleq \sum_{i=1}^{n} \beta^{n-i}\tilde{\mathbf{x}}(i,n)\tilde{\mathbf{y}}_j^H(i,n) \\ &= \beta\mathbf{Q}^H(n-1)\tilde{\mathbf{R}}_{\mathbf{x}\mathbf{y}_j}(n-1)\mathbf{Q}(n-1) \\ &\quad + \tilde{\mathbf{x}}(n,n)\tilde{\mathbf{y}}_j^H(n,n) \\ \tilde{\mathbf{R}}_{\mathbf{y}_j\mathbf{x}}(n) &\triangleq \tilde{\mathbf{R}}_{\mathbf{x}\mathbf{y}_j}^H(n) \\ \tilde{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(n) &\triangleq \sum_{i=1}^{n} \beta^{n-i}\tilde{\mathbf{x}}(i,n)\tilde{\mathbf{x}}^H(i,n) \\ &= \beta\mathbf{Q}^H(n-1)\tilde{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(n-1)\mathbf{Q}(n-1) \\ &\quad + \tilde{\mathbf{x}}(n,n)\tilde{\mathbf{x}}^H(n,n),\end{aligned} \tag{20}$$

and note from (12), (15) and (16) that for $i < n$

$$\begin{aligned}\tilde{\mathbf{x}}(i,n) &= \mathbf{Q}^H(n-1)\tilde{\mathbf{x}}(i,n-1) \\ \tilde{\mathbf{y}}_j(i,n) &= \mathbf{Q}^H(n-1)\tilde{\mathbf{y}}_j(i,n-1),\end{aligned} \tag{21}$$

and for $i = n$,

$$\begin{aligned}\tilde{\mathbf{x}}(n,n) &= \mathbf{U}'^{H}(n-1)\mathbf{x}(n) \\ \tilde{\mathbf{y}}_j(n,n) &= \mathbf{U}'^{H}(n-1)\mathbf{y}_j(n,n).\end{aligned} \tag{22}$$

This deflated cost function (18) can be minimized by maximizing

$$\tilde{J}(\mathbf{q}_j(n)) = \mathbf{q}_j^H(n)\tilde{\mathbf{R}}_j(n)\mathbf{q}_j(n) \tag{23}$$

using Sarkar et.al.'s conjugate gradient iterations [10], to estimate $\mathbf{Q}(n) = [\mathbf{q}_1(n), \mathbf{q}_2(n), \cdots, \mathbf{q}_r(n), \mathbf{q}_{r+1}(n)]$. Note in the undeflated version of the algorithm we initialize the eigenvector estimates with $\mathbf{u}_j(n-1)$ and take a conjugate gradient step to obtain $\mathbf{u}_j(n)$. Now in the deflated algorithm, the $\mathbf{Q}$–matrix represents the *change* in the subspace. Hence under no noise and steady state conditions we will find $\mathbf{Q}(n) = \mathbf{I}$ identically, and in practice, $\mathbf{Q}(n) \approx \mathbf{I}$. So in the deflated version of the algorithm, at each step we initialize the conjugate gradient iterations with $\mathbf{q}_j^{(0)}(n) = \mathbf{e}_j$, where $\mathbf{e}_j \in \mathbf{R}^{r+1}$ is a standard unit basis vector. Once formed, $\mathbf{Q}(n)$ is orthonormalized via a QR decomposition and the subspace is updated using (12).

This algorithm also estimates dominant eigenvalues. From (6), (15) and (20), $\tilde{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(n)$ can be expressed as

$$\begin{aligned}\tilde{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(n) &= \sum_{i=1}^{n} \alpha^{n-i}\tilde{\mathbf{x}}(i,n)\tilde{\mathbf{x}}^H(i,n) \\ &= \sum_{i=1}^{n} \alpha^{n-i}\mathbf{U}'^{H}(n-1)\mathbf{x}(i)\mathbf{x}^H(i)\mathbf{U}'(n-1) \\ &= \mathbf{U}'^{H}(n-1)\mathbf{R}_{\mathbf{x}\mathbf{x}}(n)\mathbf{U}'(n-1) \\ &= \alpha\mathbf{U}'^{H}(n-1)\mathbf{R}_{\mathbf{x}\mathbf{x}}(n-1)\mathbf{U}'(n-1) \\ &\quad + \tilde{\mathbf{x}}(n,n)\tilde{\mathbf{x}}^H(n,n).\end{aligned} \tag{24}$$

But for n large, $\mathbf{U}'^{H}(n-1)\mathbf{R}_{\mathbf{x}\mathbf{x}}(n-1)\mathbf{U}'(n-1)$ is nearly diagonal. So by comparing the last line of (24) with the third line of (20), we note that $\mathbf{Q}^H(n-1)\tilde{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(n-1)\mathbf{Q}(n-1)$ is also nearly diagonal with the diagonal elements being approximately equal to $\lambda_1 \cdots \lambda_r$, the eigenvalues of the signal subspace. To understand the meaning of the λ_{r+1} term, we note that at each update, $\mathbf{u}_{r+1}(n)$ is the projection of the data vector, $\mathbf{x}(n)$, into the $(m-r)$ dimensional noise subspace. Hence we see that if we desire the λ_{r+1} term to represent

the power in a given dimension of the spherical noise subspace, we must update it according to

$$\lambda_{r+1}(n) = \frac{\alpha(m-r-1)\lambda_{r+1}(n-1) + \beta_{r+1}^2}{(m-r)} \quad (25)$$

where $\beta_{r+1} \triangleq |\mathbf{u}_{r+1}^H \mathbf{x}(n)|$. With this averaging, the customized MDL method of B.Yang [11] can be used with the CGPSTd tracker. For more information on related rank tracking schemes, see [12, 13].

Now we will summarize the $O(mr^2)$ conjugate gradient projection subspace tracking algorithm. We refer to this algorithm as the conjugate gradient projection subspace tracker with deflation, or CGPSTd. To initialize the algorithm:
1) Start with an initial guess $\mathbf{R}_{xx}(0)$ and $\mathbf{U}(0) = [\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_r \ \mathbf{u}_{r+1}]$, where $\mathbf{U}^H(0)\mathbf{U}(0) = \mathbf{I}$.
2) Compute $\mathbf{y}_j(0,0)$ by (3) and use this in (22) with $\mathbf{U}'(-1) = \mathbf{U}'(0)$ to compute $\tilde{\mathbf{x}}(0,0)$ and $\tilde{\mathbf{y}}_j(0,0)$.
3) Calculate $\tilde{\mathbf{R}}_{xx}(0)$, $\tilde{\mathbf{R}}_{xy_j}(0)$ and $\tilde{\mathbf{R}}_{y_jx}(0)$ using

$$\begin{aligned} \tilde{\mathbf{R}}_{xx}(0) &= \mathbf{U}^H(0)\mathbf{R}_{xx}(0)\mathbf{U}(0) \quad (26)\\ \tilde{\mathbf{R}}_{xy_j}(0) &= \mathbf{U}^H(0)(\mathbf{R}_{xx}(0) - \\ &\qquad \mathbf{R}_{xx}(0)\sum_{l=1}^{j-1}\mathbf{u}_l(0)\mathbf{u}_l^H(0))\mathbf{U}(0) \\ \tilde{\mathbf{R}}_{y_jx} &= \tilde{\mathbf{R}}_{xy_j}^H. \end{aligned}$$

After the initialization, for each update n, we first replace the last column of $\mathbf{U}(n-1)$ with $\mathbf{u}'_{r+1}(n-1)$ using (9) to form $\mathbf{U}'(n-1)$. Next we compute $\tilde{\mathbf{x}}(n,n)$, $\tilde{\mathbf{y}}_j(n,n)$ by (22), and update $\tilde{\mathbf{R}}_{xx}(n)$ by (20) with a combined complexity $O(mr + r^3)$. Then the conjugate gradient projection iterations are carried out as follows:

1. Initialize $\mathbf{Q}(n) = \mathbf{I}$, and
   For $j = 1 : r$

   Update $\tilde{\mathbf{R}}_{xy_j}(n)$, and $\tilde{\mathbf{R}}_{y_jx}(n)$ by (20). Complexity: $O(r^3)$.
   Form $\tilde{\mathbf{R}}_j(n)$ by (19). Complexity: $O(r^2)$.
   Estimate $\mathbf{q}_j(n)$ using a single conjugate gradient iteration, with (?). Complexity: $O(r^2)$.
2. Orthonormalize $\mathbf{Q}(n) = [\mathbf{q}_1(n), \cdots, \mathbf{q}_{r+1}(n)]$ using a QR decomposition, and update $\mathbf{U}$ according to (12). Complexity: $O(r^3) + O(mr^2)$.
3. Update λ_{r+1} according to (25).
4. Update the subspace $\mathbf{U}(n) = \mathbf{U}'(n-1)\mathbf{Q}(n)$.

V. Performance Evaluation

The algorithm was tested using simulated frequency and direction of arrival tracking data. The first simulation demonstrates the convergence rate of the CGPSTd algorithm using the J. Yang and Kaveh's test. The idea behind the test is to start with an initial estimate $\mathbf{R} = .01\mathbf{I}$ so that the initial signal subspace is $\mathbf{U}_s(0) = [\mathbf{e}_1\mathbf{e}_2]$. This is the usual "cold start" method we use to initialize our algorithm. Two signals, one at 9^o and the other at 12^o are then applied and we are interested in seeing how fast the tracker can lock onto the signals, where ROOTMUSIC [14] is used to extract the DOA estimates from the subspace assuming a $\lambda/2$ linear equally spaced 8–sensor array. By comparing figure 1 to the twelve curves of figure 6 of [15], we see that the conjugate gradient projection algorithm converges considerably faster than its stochastic gradient cousins.

We next benchmark the CGPSTd algorithm with the Comon and Golub test [1]. In this test, we define a vector MA-process according to

$$\begin{aligned} \mathbf{r}(k) &= s_1(k)\mathbf{e}_1 + s_2(k)\mathbf{e}_2 + \mathbf{n}(k), \ k \le t_0 = 10 \quad (27)\\ \mathbf{r}(k) &= s_1(k)\mathbf{e}_3 + s_2(k)\mathbf{e}_4 + \mathbf{n}(k), \ k > t_0 = 10 \end{aligned}$$

where the $s_1(k)$ and $s_2(k)$ are random scalar process with variance $\sigma_1 = 4$ and $\sigma_2 = 1$. Hence we initialize things so the subspace should lie in the $\mathbf{e}_1 - \mathbf{e}_2$ plane, and then abruptly switch into the $\mathbf{e}_3 - \mathbf{e}_4$ plane. Subspace convergence is measured in terms of the amount of time it takes for the angles between the algorithm under consideration and the true SVD to cross the one degree line which is indicated on Figures 2 and 3. In this test, the fading factor is taken to be $\alpha(k) = 1 - \frac{1}{k-t_0}$, $k > t_0$ with $t_0 = 10$. In Figure 2 we see the convergence rate of the CGPSTd compares quite well with the algorithms in [1]. To see how CGPSTd compares with PASTd subspace tracker with reorthonormalization, we plot the response on the PASTd to the Comon and Golub test. We note that the conjugate gradient based tracker demonstrates a slightly superior convergence performance, but at a slightly higher cost.

VI. Conclusion

A new subspace tracking algorithm was developed using the projection approximation cost function of B. Yang and conjugate gradient iterations originally proposed by Sarkar [10]. The algorithm is similar in ways to the PASTd algorithm, but converges faster with a slightly higher cost in computation. The algorithm is applicable to frequency and angle tracking problems as well as any other problem where it is desirable to track the dominant subspace associated with a time-varying data or data correlation matrix.

References

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

[2] B. Yang, "Projection approximation subspace tracking," *submitted to IEEE Trans. SP*, 1993.

[3] B. Yang, "Subspace tracking based on the projection approach and the recursive least squares method," in *IEEE ICASSP*, pp. IV145–IV148, 1993.

[4] E. M. Dowling, L. P. Ammann, and R. D. DeGroat, "A TQR-iteration based SVD for real time angle and frequency tracking," *IEEE Transactions on Signal Processing*, pp. 914–925, April 1994.

[5] T. S. X.Yang and E.Arvas, "A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix," *IEEE Transaction on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1550–1556, October 1989.

[6] Z. Fu and E. M. Dowling, "Conjugate gradient eigenstructure tracking for adaptive spectral estimation," *to appear in the IEEE Trans. Signal Processing*, 1995.

[7] Z. Fu and E. M. Dowling, "Conjugate gradient projection subspace tracking and systolic implementation," *Submitted to the IEEE Trans. Signal Processing*, May 1994.

[8] Z. Fu, *Conjugate gradient subspace tracking and real time subspace tracking array architectures*. PhD thesis, University of Texas at Dallas, 1994.

[9] I. Karasalo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Trans. ASSP*, vol. ASSP-34, pp. 8–12, Feb. 1986.

[10] T. Sarkar and X.Yang, "Application of the Conjugate Gradient and Steepest Descent for Computing the Eig envalues of an Operator," *Signal Processing*, vol. 17, pp. 31–38, 1989.

[11] B. Yang and F. Gersemsky, "An adaptive algorithm of linear computational complexity for both rank and subspace tracking," in *Proceedings of the ICASSP*, pp. IV33–36, April 1994.

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

[13] R. DeGroat, E. M. Dowling, H. Ye, and D. A. Linebarger, "Spherical subspace tracking for efficient, high performance adaptive signal processing applications," *IEEE Trans. Signal Processing*, Submitted August 1994.

[14] A. J. Barabell, "Improving the resolution of eigenstructure-based direction-finding algorithms," in *Proceedings ICASSP*, pp. 336–339, 1983.

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

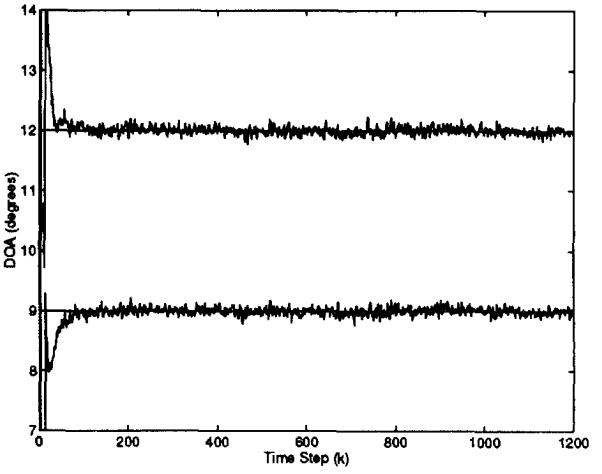

Figure 1 J.Yang and Kaveh's test results for CGPSTd

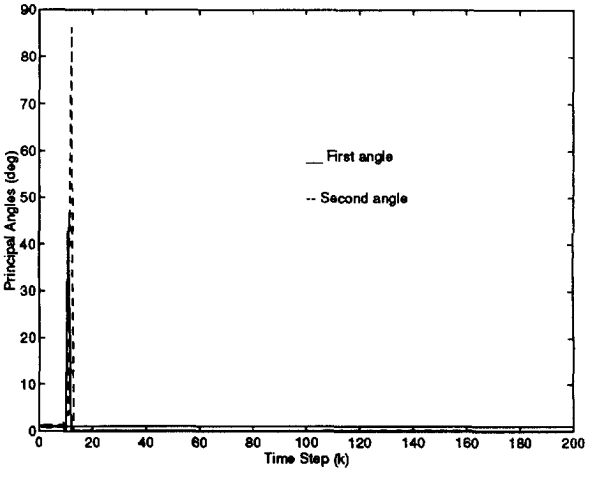

Figure 2 Comon and Golub's test results for CGPSTd

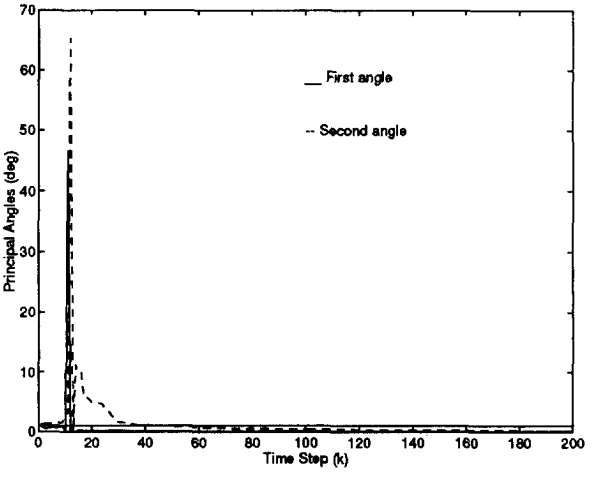

Figure 3 Comon and Golub's test results for PASTd