A Subspace Tracking Algorithm Using the Fast Fourier Transform

James W. Cooley, *Fellow, IEEE*, Timothy M. Toolan, and Donald W. Tufts, *Life Fellow, IEEE*

***Abstract*—At *ICASSP '97* and in the July 1999 IEEE TRANSACTIONS ON SIGNAL PROCESSING, Real *et al.* presented an algorithm for fast tracking of a signal subspace or interference subspace for application in adaptive detection or estimation. For cases in which the signal matrix is formed from a single-channel discrete-time signal, we show how one can further reduce computation in the fast approximate subspace tracking (FAST) algorithm by using the fast Fourier transform.**

***Index Terms*—Fast approximate subspace tracking (FAST) algorithm, Fourier transform, subspace tracking, singular value decomposition (SVD).**

I. SUBSPACE TRACKING ALGORITHM

AT EACH time step, the *fast approximate subspace tracking* (FAST) algorithm [1] computes an approximate *singular value decomposition* (SVD) of an $r \times c$ signal matrix

$$\boldsymbol{M}_{\text{new}} = [\,\boldsymbol{m}_2 \quad \boldsymbol{m}_3 \quad \cdots \quad \boldsymbol{m}_{c+1}\,] \tag{1}$$

where each $\boldsymbol{m}_j$ is an r-element column vector. The approximation is based on the use of a smaller dimensional subspace that contains most of the signal. This subspace comes from the previous iteration, which computed the approximate SVD of the matrix

$$\boldsymbol{M}_{\text{old}} = [\,\boldsymbol{m}_1 \quad \boldsymbol{m}_2 \quad \cdots \quad \boldsymbol{m}_c\,] \tag{2}$$

which shares $c-1$ of its c columns with $\boldsymbol{M}_{\text{new}}$.

This note shows how the fast Fourier transform (FFT) algorithm may be used to reduce computation in the FAST algorithm for the situation in which $\boldsymbol{M}_{\text{old}}$ and $\boldsymbol{M}_{\text{new}}$ are formed from a single-channel discrete-time signal [2]–[4]. In this case, $\boldsymbol{M}_{\text{old}}$ and $\boldsymbol{M}_{\text{new}}$ are $r \times c$ Hankel-like (Hankel if $r = c$) matrices of the form

$$\boldsymbol{M}_{\text{new}} = \begin{bmatrix} d_2 & d_3 & \cdots & d_{c+1} \\ d_3 & d_4 & \cdots & d_{c+2} \\ \vdots & \vdots & & \vdots \\ d_{r+1} & d_{r+2} & \cdots & d_{r+c} \end{bmatrix}. \tag{3}$$

The $N = r + c - 1$ unique elements of $\boldsymbol{M}_{\text{new}}$ can be written as the $N \times 1$ column vector

$$\boldsymbol{d}_{\text{new}} = [\,d_2 \quad d_3 \quad \cdots \quad d_{N+1}\,]^T. \tag{4}$$

Manuscript received September 17, 2002; revised January 17, 2003. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Darren B. Ward.

The authors are with the Department of Electrical Engineering, University of Rhode Island, Kingston, RI 02881 USA (e-mail: toolan@ele.uri.edu).

Digital Object Identifier 10.1109/LSP.2003.819352

At the beginning of the current iteration, one has k principal left singular vectors of $\boldsymbol{M}_{\text{old}}$ in the columns of the matrix

$$\boldsymbol{U}_{\text{old}} = [\,\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \cdots \quad \boldsymbol{u}_k\,] \tag{5}$$

as well as the matrix $\boldsymbol{M}_{\text{new}}$.

Summarizing the FAST algorithm presented in [1, eqs. (14)–(32)], we calculate a $k+1 \times c$ matrix, $\boldsymbol{E}$, which when premultiplied by $[\boldsymbol{U}_{\text{old}}\ \boldsymbol{q}]$, is the projection of $\boldsymbol{M}_{\text{new}}$ onto the columns of $\boldsymbol{U}_{\text{old}}$ plus the component of $\boldsymbol{m}_{c+1}$ orthogonal to $\boldsymbol{U}_{\text{old}}$. We form the matrix $\boldsymbol{E}$ as follows:

$$\boldsymbol{E} = \left[\begin{array}{cccc|c} \boldsymbol{a}_2 & \boldsymbol{a}_3 & \cdots & \boldsymbol{a}_c & \boldsymbol{a}_{c+1} \\ 0 & 0 & \cdots & 0 & b \end{array}\right] \tag{6}$$

where the $\boldsymbol{a}_j$'s are k element vectors

$$\boldsymbol{a}_j = \boldsymbol{U}_{\text{old}}^H \boldsymbol{m}_j, \qquad j = 2, 3, \ldots, c+1 \tag{7}$$

b is the norm of the component of $\boldsymbol{m}_{c+1}$ orthogonal to the columns of $\boldsymbol{U}_{\text{old}}$, and $\boldsymbol{q}$ is that component normalized. They are computed as follows:

$$\boldsymbol{z} = \left(\boldsymbol{I} - \boldsymbol{U}_{\text{old}}\boldsymbol{U}_{\text{old}}^H\right)\boldsymbol{m}_{c+1} \quad b = \|\boldsymbol{z}\| \quad \boldsymbol{q} = \frac{\boldsymbol{z}}{b}. \tag{8}$$

Instead of computing the $O\left(rc^2\right)$ SVD [5] of the $r \times c$ matrix $\boldsymbol{A} = [\boldsymbol{U}_{\text{old}}\ \boldsymbol{q}]\boldsymbol{E}$, or the $O\left((k+1)c^2\right)$ SVD of the matrix $\boldsymbol{E}$, we compute the $O\left((k+1)^3\right)$ SVD of the $k+1 \times k+1$ matrix

$$\boldsymbol{F} = \boldsymbol{E}\boldsymbol{E}^H = \boldsymbol{U}_F \boldsymbol{\Sigma}_F \boldsymbol{U}_F^H \tag{9}$$

where our estimates of the $k+1$ principal left singular vectors and values of $\boldsymbol{M}_{\text{new}}$ are

$$\boldsymbol{U}_{\text{new}} = [\boldsymbol{U}_{\text{old}}\ \boldsymbol{q}]\,\boldsymbol{U}_F \tag{10}$$

and

$$\boldsymbol{\Sigma}_{\text{new}} = \sqrt{\boldsymbol{\Sigma}_F}. \tag{11}$$

The original contribution of this letter is the demonstration that the amount of computation in the FAST algorithm can be reduced for the case in which the computation of (7) can be written as a product of a matrix and a Hankel-like matrix as in (13). This case arises whenever the matrices $\boldsymbol{M}_{\text{old}}$ and $\boldsymbol{M}_{\text{new}}$ are formed from a single-channel discrete-time signal [2]–[4]. In the last section, the amount of computational reduction is quantified, and a formula (24) based on the dimensions of the signal matrix (r and c) is given to determine if this method actually reduces

computation or not for a given specific interference subspace dimension (k).

II. Use of the FFT

The suggestion made here is that one make use of the fact that the $\boldsymbol{a}_j$'s are a set of convolutions that can be computed using the FFT [5], [6]. To display this as a convolution, we write the ith component of $\boldsymbol{a}_j$ as

$$a_{ij} = \sum_{l=1}^{r} u_{li}^{*}\, d_{l+j-1}, \qquad j = 2, \ldots c+1 \tag{12}$$

or in matrix form

$$\boldsymbol{A} = \boldsymbol{U}_{\text{old}}^{H} \boldsymbol{M}_{\text{new}}. \tag{13}$$

Note that the elements of $\boldsymbol{u}_i$ in (12) are reversed and conjugated in the convolution sum, like a cross correlation. This is equivalent to conjugating the DFT of $\boldsymbol{u}_i$.

To efficiently calculate the a_{ij}'s, we pad the columns of $\boldsymbol{U}_{\text{old}}$ with zeros to make them of length N and compute their discrete Fourier transforms (DFTs)

$$\bar{\boldsymbol{U}} \overset{\text{FFT}}{\longleftrightarrow} \begin{bmatrix} \boldsymbol{U}_{\text{old}} \\ \boldsymbol{0} \end{bmatrix}. \tag{14}$$

We then take the DFT of $\boldsymbol{d}_{\text{new}}$

$$\bar{\boldsymbol{d}} \overset{\text{FFT}}{\longleftrightarrow} \boldsymbol{d}_{\text{new}}. \tag{15}$$

Now, the convolution in (12) can be performed by the Hadamard product of the conjugate transpose of each column of $\bar{\boldsymbol{U}}$, with the transpose of $\bar{\boldsymbol{d}}$.

$$\bar{\boldsymbol{A}} = \bar{\boldsymbol{U}}^{H} \odot [\bar{\boldsymbol{d}} \cdots \bar{\boldsymbol{d}}]^{T} \tag{16}$$

Finally, we take the inverse DFT of the *rows* of $\bar{\boldsymbol{A}}$, whose first c columns are $\boldsymbol{a}_2$ through $\boldsymbol{a}_{c+1}$.

$$[\,\boldsymbol{a}_2 \quad \boldsymbol{a}_3 \quad \cdots \quad \boldsymbol{a}_{c+1} \quad \cdots\,] \overset{\text{IFFT}}{\longleftrightarrow} \bar{A}. \tag{17}$$

III. Operations Count

The two dominant operations in the algorithm are the calculations of the $\boldsymbol{a}_j$'s (7) for small signal subspace dimensions, and the SVD of F (9) for large dimensions. This method addresses the calculation of the $\boldsymbol{a}_j$'s.

For the comparisons, we will estimate the number of floating-point operations (flops) for each method. We will assume that both real addition and real multiplication require one flop, while complex addition requires two flops, and complex multiplication requires six flops.

The calculation of the $\boldsymbol{a}_j$'s using (7) requires

$$N_{\text{Adir}} = kN_{\text{DIR}} \text{ flops} \tag{18}$$

where

$$N_{\text{DIR}} = 2rc \text{ flops} \tag{19}$$

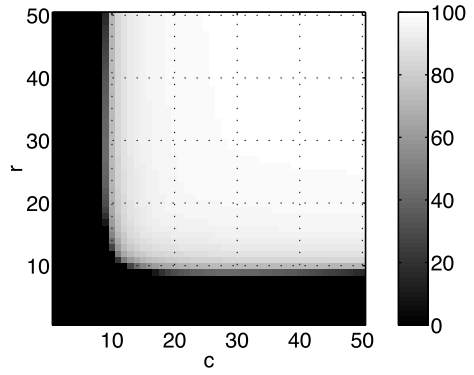

Fig. 1. Percentage of all k's calculated more efficiently using the FFT method for r and c from 1 to 50.

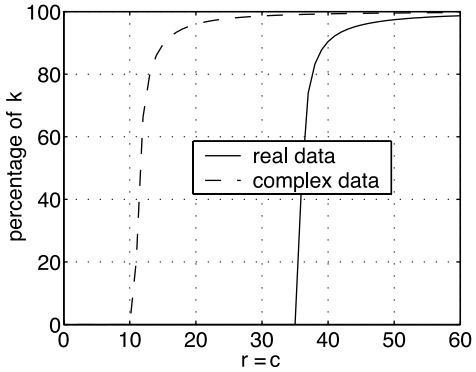

Fig. 2. Percentage of all k's calculated more efficiently using the FFT method for $r = c$ from 1 to 60 and both real and complex data.

when the data are real and

$$N_{\text{DIR}} = 8rc \text{ flops} \tag{20}$$

when the data are complex. Remember that r and c are the dimensions of $\boldsymbol{M}_{\text{new}}$, and k is the signal subspace dimension.

When calculating the $\boldsymbol{a}_j$'s using (14)–(17), we must first come up with an approximation of the flop count of an N-point FFT. We take the following as an approximation of the number of flops for a simple radix 2 FFT:

$$N_{\text{FFT}} = \frac{5}{2} N \log_2(N) - \frac{N}{2} \text{ flops} \tag{21}$$

when the data is real, and

$$N_{\text{FFT}} = 5N \log_2(N) - 3N \text{ flops} \tag{22}$$

when the data are complex [7]. A radix 4 algorithm, with a radix 2 step for N equal to an odd power of 2, takes 25% fewer flops. When N is not a power of 2 and N has large prime factors, the FFT can take more flops. There are $2k+1$ FFTs, so, adding to the above the $6Nk$ flops for the complex multiplication of (16), we get

$$N_{\text{Afft}} = (2k+1)N_{\text{FFT}} + 6Nk \text{ flops}. \tag{23}$$

For any r and c, k can range from zero to $k_{\max}$, where $k_{\max} = \min(r, c)$. The transition signal subspace dimension k_{trans}, where all values of k above that value take less flops

using the FFT method and all values of k below that value take less flops using the direct multiplication method, is

$$k_{\text{trans}} = \frac{N_{\text{FFT}}}{N_{\text{DIR}} - 2N_{\text{FFT}} - 6N}. \tag{24}$$

When $k_{\text{trans}} < 0$, the direct multiplication method is more efficient for all k.

To give an idea of how k_{trans} evaluates for different r and c, we use the radix 2 FFT flop count and assume complex data. Fig. 1 shows that there is a very steep transition from where all k are more efficiently calculated using the direct multiplication method to where all k are more efficiently calculated using the FFT method. Given values for r and c, it is easy to determine which method to use.

Fig. 2 shows the percentage of signal subspace dimensions k, where the FFT method is more efficient than the direct multiplication method. In this figure, $r = c$, and both real and complex data are shown. It can be seen that for complex matrices with dimensions greater than 15 (which is often the case) and real matrices with dimensions greater than 40, one would probably want to use the FFT method.

IV. Summary

We have established that, for the above signal tracking method, where the signal matrix is a Hankel-like matrix, the FFT method may be superior for large r and c. The above formulas should enable one to evaluate which method is more efficient for any given parameters of the problem.

References

[1] E. C. Real, D. W. Tufts, and J. W. Cooley, "Two algorithms for fast approximate subspace tracking," *IEEE Trans. Signal Processing*, vol. 47, pp. 1036–1045, July 1999.

[2] D. W. Tufts, R. Kumaresan, and I. Kirsteins, "Data adaptive signal estimation by singular value decomposition of data matrix," *Proc. IEEE*, vol. 70, pp. 684–685, June 1982.

[3] D. W. Tufts and R. Kumaresan, "Singular value decomposition and improved frequency estimation using linear prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 671–675, Aug. 1982.

[4] L. L. Scharf and D. W. Tufts, "Rank reduction for modeling stationary signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 350–355, Mar. 1987.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[6] T. Kailath and A. H. Sayed, *Fast Reliable Algorithms for Matrices With Structure*. Philadelphia, PA: SIAM, 1999.

[7] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The Fast Fourier Transform Algorithm and its Applications," IBM Watson Research Center, Yorktown Heights, NY, Res. Paper RC-1743, 1967.