LOCAL STABILITY ANALYSIS AND SYSTOLIC IMPLEMENTATION OF A SUBSPACE TRACKING ALGORITHM

Fan Xu and Alan N. Willson, Jr.

Electrical Engineering Department
University of California
Los Angeles, CA 90095 USA

ABSTRACT

We discuss the *DPASTd* algorithm for signal subspace tracking. Our analysis shows that, under a sufficient condition on the step size, the *DPASTd* algorithm is locally stable even though delayed updating is applied. A pipelined realization of the algorithm and the corresponding systolic architecture are also proposed and a method for reciprocal computation is discussed. We also present simulation results to validate the algorithm.

1. INTRODUCTION

Estimating and tracking the signal subspace is required in many signal processing applications such as direction of arrival (DOA) estimation and blind equalization. Traditionally, this operation has been performed in software because of its high computational complexity. In recent years, however, the appearance of low-cost subspace tracking algorithms, along with the development of VLSI technology, are tending to make it possible to construct a pure hardware solution.

The *PASTd* algorithm described in [1] is an example of such an algorithm. This algorithm has a very low computational complexity of $O(rN)$ and is recursive least-squares (RLS) based, thus potentially converging fast.

One unfortunate aspect of the original *PASTd* algorithm is that it suffers from long computation delay. In order to facilitate a high-speed implementation, we have proposed in [2] the *DPAST* algorithm, based on the idea of delayed updating. In this paper, we discuss the local stability and a systolic implementation of a deflated version of the *DPAST* algorithm (*DPASTd*).

In the second section of this paper, we briefly review the *PASTd* algorithm. Section 3 discusses the *DPASTd* algorithm and especially its local stability under an independence assumption. We propose in Section 4 a pipelined realization of the algorithm and the corresponding systolic architecture. The design of critical cells is also studied. Section 5 summarizes our conclusions.

2. THE ALGORITHM UNDER DISCUSSION

A novel cost function is proposed in [1] to solve the subspace tracking problem adaptively. Let $x(n)$ be a real random sequence, let $\mathbf{x}(n)$ be an N-vector with $\mathbf{x}(n) = [x(n),\ x(n-1),\ \ldots,\ x(n-N+1)]^T$ and let $\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}(n)^T]$ be its correlation matrix. The subspace constructed by the $r < N$ largest eigenvectors[1] can be identified by minimizing

$$J(\mathbf{W}) = E\|\mathbf{x} - \mathbf{W}\mathbf{W}^T\mathbf{x}\|^2. \tag{1}$$

It is proved in [1] that $\mathbf{W}$ is the global minimum of $J(\mathbf{W})$ if and only if $\mathbf{W} = \mathbf{U_r}\mathbf{Q}$, where $\mathbf{U_r}$ contains the r largest eigenvectors and $\mathbf{Q}$ is an arbitrary unitary matrix.

Modifying the cost function to be

$$J'(\mathbf{W}(n)) = \sum_{i=1}^{n} \beta^{n-i}\|\mathbf{x}(i) - \mathbf{W}(n)\mathbf{y}(i)\|^2$$

where $\mathbf{y}(i) = \mathbf{W}^T(i-1)\mathbf{x}(i)$, and following the same procedure as is used in deriving the RLS algorithm, we can obtain the so-called *PAST* algorithm. The *PASTd* algorithm is derived from the *PAST* using the deflation technique. The computational complexity is thus reduced to $O(rN)$. Descriptions of both algorithms can be found in [1].

3. THE *DPASTD* ALGORITHM

3.1. The algorithm

We have proposed an alternative cost function, which could lead to a high-speed solution. That is, let

$$J_D(\mathbf{W}(n)) = E[\|\mathbf{x}(n-D) - \mathbf{W}\mathbf{W}^T\mathbf{x}(n-D)\|^2]. \tag{2}$$

Under the condition that $\mathbf{x}(n)$ *is a stationary signal, or a time-varying signal but one whose statistical properties do not change during the length-D time period,* the solution to (2) is the same as that to (1). Both an LMS-like algorithm and an RLS-like algorithm can be derived to minimize (2). In particular, the RLS-like algorithm is called *DPAST* and it is summarized in [2].

Similar to [1], we invoke the deflation technique to reduce the computational complexity of *DPAST* to $O(rN)$. The deflated version of the *DPAST* algorithm (to be called *DPASTd*) is shown in Table I.

Notice that we use $\mathbf{w}_i(n-2)$, instead of $\mathbf{w}_i(n-1)$ as in *PASTd*, in the updating equation for the error term $\mathbf{e}_i(n)$ in order to further reduce the computation delay in a pipelined realization. Furthermore, $\mathbf{x}_i^D(n)$ in the table is not a delayed version of signal $\mathbf{x}_i(n)$, but rather, it is deflated from $\mathbf{x}_{i-1}^D(n)$. Therefore, unlike in the *PASTd* algorithm, two deflations need to be carried out in the *DPASTd* algorithm.

This work was supported in part by UC MICRO GRANT 00-104.

[1] We use the term "largest eigenvector" to indicate the eigenvector that corresponds to the largest eigenvalue of $\mathbf{R}$. That is, we order the eigenvectors according to the natural ordering of the corresponding eigenvalues.

TABLE I
DPASTd ALGORITHM FOR SUBSPACE TRACKING

Choose $d_i(0)$ and $\mathbf{w}_i(0)$ properly;
for $n = 1,\ 2,\ \ldots$ do
  $x_1(n) = x(n)$;
  $x_1^D(n) = x(n-D)$;
  for $i = 1,\ , 2, \ldots,\ r$
    $y_i(n) = \mathbf{w}_i(n-1)^T \mathbf{x}_i(n)$;
    $d_i(n) = \beta d_i(n-1) + y_i^2(n-D)$;
    $\mathbf{e}_i(n) = \mathbf{x}_i^D(n) - \mathbf{w}_i(n-2) y_i(n-D)$;
    $\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + \mathbf{e}_i(n) y_i(n-D)/d_i(n)$;
    $\mathbf{x}_{i+1}(n) = \mathbf{x}_i(n) - \mathbf{w}_i(n) y_i(n)$;
    $\mathbf{x}_{i+1}^D(n) = \mathbf{x}_i^D(n) - \mathbf{w}_i(n) y_i(n-D)$;
  end
end

3.2. Analysis of local stability

Since extra delays have been inserted, the difference equation for the *DPASTd* algorithm has a different order than that of the *PASTd* algorithm. Certain mathematical analyses must be carried out to guarantee the algorithm's convergence and stability.

In this paper, we only analyze the local stability of the algorithm and our analysis is carried out in three steps. We start with a special case of the algorithm where $r = 1$ and $\mu(n) = \frac{1}{d(n)} = \mu$ is a constant and we show that *DPASTd* is locally stable under these two assumptions. Then, the assumption on $\mu(n)$ is released and we discuss how a time-varying step-size affects the proof. r is still assumed to be one at this point. Finally, the results in step 1 and step 2 are extended to the stability proof of the more general *DPASTd* algorithm, where r can be an arbitrary positive integer.

3.2.1. Step 1: a special case where $r = 1$ and $\mu(n) = \mu$

In this special case, the *DPASTd* algorithm coincides with an LMS-like algorithm similar to the one in [1], which reads

$$\begin{aligned}\mathbf{w}(n) = \mathbf{w}(n-1) + \mu(\mathbf{x}(n-D) \\ - \mathbf{w}(n-2)y(n-D))y(n-D).\end{aligned} \tag{3}$$

In order to prove the local stability of (3), let us construct an expanded nonlinear state-space model for it. Defining $\mathcal{W}(n) = [\mathbf{w}(n),\ \mathbf{w}(n-1),\ \ldots,\ \ldots,\ \mathbf{w}(n-D)]^T$ as an $N(D+1) \times 1$ state vector, we obtain the following recursion from (3)[2]:

$$\mathcal{W}(n) = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ & \mathbf{I} & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \mathbf{I} & \mathbf{0} \end{bmatrix} \mathcal{W}(n-1) + \mu \begin{bmatrix} \mathbf{x}\mathbf{x}^T\mathbf{w}(n-D-1) \\ -\mathbf{w}(n-2)\mathbf{w}^T(n-D-1)\mathbf{x}\mathbf{x}^T\mathbf{w}(n-D-1) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{4}$$

[2] In this paper, $\mathbf{x}$ represents $\mathbf{x}(n-D)$, if not mentioned explicitly.

An assumption that is usually made in the literature is that the eigenvalues of $\mathbf{R}$, $\lambda_1, \lambda_2, \ldots, \lambda_N$, satisfy

$$\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_N.$$

That is, λ_1 is strictly larger than the other eigenvalues. The "independence theory" of [4] is also assumed in this paper.

Let $\mathbf{W}_1 = [\mathbf{w}_1, \mathbf{w}_1, \ldots, \mathbf{w}_1]^T$, where $\mathbf{w}_1$ is the eigenvector corresponding to λ_1. We define $\tilde{\mathcal{W}}(n) = \mathcal{W}(n) - \mathbf{W}_1$. Since only the local behavior is of interest, we linearize (4) around $\mathbf{W}_1$ and obtain a linear state-space model for $\tilde{\mathcal{W}}(n)$, which is not shown here. However, it can be predicted from (4) that the resulting equation, although linear, is still time-varying and stochastic, hence too complicated to analyze.

If μ is small, on the other hand, the "direct-averaging method" can be invoked to simplify this equation [4]. It has been proved that the behavior of such a stochastic difference equation, under the assumption of small μ, can be approximated by another stochastic difference equation with a constant system-matrix. More precisely, the linear updating equation for $\tilde{\mathcal{W}}(n)$, under the quasi-stationary assumption, can be approximated by

$$\tilde{\mathcal{W}}(n) = \mathbf{A}\tilde{\mathcal{W}}(n-1) + \mu\mathbf{u}(n) \tag{5}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & -\mu\lambda_1\mathbf{I} & \ldots & \ldots & \mu\bar{\mathbf{R}} \\ \mathbf{I} & \mathbf{0} & \ldots & \ldots & \mathbf{0} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \mathbf{I} & \mathbf{0} \end{bmatrix}$$

$\bar{\mathbf{R}} = \sum_{i=2}^{N} \lambda_i \mathbf{w}_i \mathbf{w}_i^T$ and

$$\mathbf{u}(n) = [\mathbf{x}\mathbf{x}^T\mathbf{w}_1 - \mathbf{w}_1\mathbf{w}_1^T\mathbf{x}\mathbf{x}^T\mathbf{w}_1, \mathbf{0}, \ldots, \mathbf{0}]^T.$$

Notice that $\mathbf{u}(n)$ is zero-mean.

It follows [5] that the stability of (5) depends on the eigen-distribution of the matrix $\mathbf{A}$.

Proposition 1 *Any eigenvalue λ of the matrix $\mathbf{A}$ must satisfy*

$$\lambda^{D-1}(\lambda^2 - \lambda + \mu\lambda_1) = \mu\lambda_i \tag{6}$$

for $i = 2,\ \ldots,\ N$.

Proof: Suppose λ is an eigenvalue of $\mathbf{A}$ and $\mathbf{y}$ is the corresponding eigenvector. Write $\mathbf{y}$ as $[\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_D]^T$ where $\mathbf{y}_i$, $i = 0,\ 1, \ldots,\ D$, is a length-N vector. We can expand $\mathbf{A}\mathbf{y} = \lambda\mathbf{y}$ as

$$\begin{bmatrix} \mathbf{I} & -\mu\lambda_1\mathbf{I} & \ldots & \ldots & \mu\bar{\mathbf{R}} \\ \mathbf{I} & \mathbf{0} & \ldots & \mathbf{0} & \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \vdots \\ \mathbf{y}_D \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \vdots \\ \mathbf{y}_D \end{bmatrix}.$$

It is not difficult to verify that

$$\begin{aligned} \mathbf{y}_0 - \mu\lambda_1\mathbf{y}_1 + \mu\bar{\mathbf{R}}\mathbf{y}_D &= \lambda\mathbf{y}_0 \\ \mathbf{y}_0 &= \lambda\mathbf{y}_1 \\ \mathbf{y}_1 &= \lambda\mathbf{y}_2 \\ &\vdots \\ \mathbf{y}_{D-1} &= \lambda\mathbf{y}_D \end{aligned}$$

which implies

$$\lambda^{D-1}(\lambda^2 - \lambda + \mu\lambda_1)\mathbf{y}_D = \mu\bar{\mathbf{R}}\mathbf{y}_D. \quad (7)$$

Eqn. (7) shows that $\lambda^{D-1}(\lambda^2 - \lambda + \mu\lambda_1)$ is an eigenvalue of $\mu\bar{\mathbf{R}}$. From the definition of $\bar{\mathbf{R}}$ we conclude that (6) must be satisfied. ■

Proposition 2 *A sufficient condition for* $\mathbf{A}$ *to be a stable matrix is that*

$$|\lambda^{D-1}(\lambda^2 - \lambda + \mu\lambda_1)| > \mu\lambda_2$$

for any $|\lambda| \geq 1$.

The sufficiency of this condition is obvious because it excludes the possibility that an eigenvalue of $\mathbf{A}$ falls in the region $|\lambda| \geq 1$.

Proposition 3 *Let* $f(\lambda) = \lambda^{D-1}(\lambda^2 - \lambda + \mu\lambda_1)$. *The minimum of* $|f(\lambda)|$ *in the region* $|\lambda| \geq 1$ *is* $\mu\lambda_1$ *if* $\mu < \frac{1}{3\lambda_1}$.

Proof: It is evident [6] that the minima of $|f(\lambda)|$ are on the unit circle. Thus, let $\lambda = e^{j\theta}$. We derive $|f(\lambda)|^2 = 4\mu\lambda_1\cos\theta^2 - 2(1+\mu\lambda_1)\cos\theta + 2 - 2\mu\lambda_1 + \mu^2\lambda_1^2$. It is not difficult to verify that, if $\mu < \frac{1}{3\lambda_1}$, the minima of $|f(\lambda)|^2$ occur at $\theta = 2k\pi$, $k = 0,\ 1,\ \ldots$, and, therefore, $|f(\lambda)|_{min} = \mu\lambda_1$. ■

We easily conclude, from Proposition 2, Proposition 3 and our assumption on the eigen-distribution of the matrix $\mathbf{R}$, that Eqn. (3), if its step size is less than $\frac{1}{3\lambda_1}$, is locally stable, regardless of the actual value D. Notice that the condition on μ is only sufficient.

3.2.2. Step 2: proof without assuming $\mu(n)$ being a constant

The only difference between (3) and the *DPASTd* algorithm, when $r = 1$, is that the step size μ is replaced by $\mu_n = \frac{1}{d(n)}$ in *DPASTd*, where $d(n) = \beta d(n-1) + |y(n-D)|^2$. The linearized state-space model for *DPASTd* hence becomes

$$\tilde{\mathcal{W}}(n) = \mathbf{A}_{\mu_n}\tilde{\mathcal{W}}(n-1) + \mu_n\mathbf{u}(n) \quad (8)$$

where

$$\mathbf{A}_{\mu_n} = \begin{bmatrix} \mathbf{I} & -\mu_n\mathbf{w}_1^T\mathbf{x}\mathbf{x}^T\mathbf{w}_1 & \ldots & \mathbf{0} & \mu_n(\mathbf{x}\mathbf{x}^T - \mathbf{w}_1\mathbf{w}_1^T\mathbf{x}\mathbf{x}^T) \\ \mathbf{I} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ & \ddots & \ddots & \vdots & \vdots \\ & & \ddots & \mathbf{0} & \vdots \\ & & & \mathbf{I} & \mathbf{0} \end{bmatrix}.$$

To analyze this system's local stability, let us assume that μ_n is uniformly bounded from below, i.e., $\mu_n \geq \delta > 0$ for any $n \geq 0$. We can show intuitively that μ_n, for a proper initial value μ_0 and sufficiently large n, will vary slowly around a non-zero mean value. In this case, we make an assumption that is similar to Assumption 1 in [7]. That is, we assume that μ_n and $\mathbf{x}$ are independent. Under this assumption, the system matrix for the "averaged" system reads

$$A = \begin{bmatrix} \mathbf{I} & -E[\mu_n]\lambda_1\mathbf{I} & \ldots & \ldots & E[\mu_n]\bar{\mathbf{R}} \\ \mathbf{I} & \mathbf{0} & \ldots & \ldots & \mathbf{0} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \mathbf{I} & \mathbf{0} \end{bmatrix}. \quad (9)$$

The stability is then proved in a similar manner as before. We just use $E[\mu_n]$ instead of a constant μ. A sufficient condition for local stability is: $E[\mu_n] < \frac{1}{3\lambda_1}$ which can easily be satisfied by choosing a proper initial value $d(0)$.

3.2.3. Step 3: extension to the case $r > 1$

In the case where $r > 1$, we further assume that $\lambda_1 > \lambda_2 > \ldots > \lambda_r > \lambda_{r+1} \geq \ldots \geq \lambda_N$. It can be seen from Table I that the updating equation for $\mathbf{w}_2(n)$, for example, is the same as the one for $\mathbf{w}_1(n)$ except its inputs are $\mathbf{x}_2(n)$ and $\mathbf{x}_2^D(n)$. In other words, this equation estimates the largest eigenvector of $\mathbf{x}_2(n)$. We learn from the deflation equations that, for sufficiently large n, the largest eigenvalue of the correlation matrices of both $\mathbf{x}_2(n)$ and $\mathbf{x}_2^D(n)$ is λ_2, with corresponding eigenvector $\mathbf{w}_2$, since both $\mathbf{w}_1(n)$ and $\mathbf{w}_1(n-D)$ have already converged to $\mathbf{w}_1$. The stability of the algorithm can be proved in the same way as before. Similar arguments were made in the proof of the original *PASTd* algorithm [3].

4. THE PIPELINED REALIZATION

4.1. A hardware-realizable form of the algorithm

In order to make the algorithm more favorable to a pure hardware realization, we propose the following simplifications.

First of all, we release the dependency between $y_i(n)$ and $\mathbf{w}_{i-1}(n)$ by substituting the updating equation for $\mathbf{x}_i(n)$ into the equation for $y_i(n)$

$$\begin{aligned} y_i(n) &= \mathbf{w}_i(n-1)^T\mathbf{x}_i(n) \\ &= \mathbf{w}_i(n-1)^T\mathbf{x}_{i-1}(n) \\ &\quad - \mathbf{w}_i(n-1)^T\mathbf{w}_{i-1}(n)y_{i-1}(n-D),\ \text{for } i \geq 2. \end{aligned} \quad (10)$$

Since $\mathbf{W}(n)$ converges to an orthogonal matrix, the second term in (10) has little effect on the computation of $y_i(n)$ in the steady state. Therefore, we can assume that $\mathbf{w}_i(n-1)^T\mathbf{w}_{i-1}(n) \approx 0$ in the steady state and (10) is hence simplified to

$$\begin{aligned} y_i(n) &\approx \mathbf{w}_i(n-1)^T\mathbf{x}_{i-1}(n) \\ &\approx \mathbf{w}_i(n-1)^T\mathbf{x}_{i-2}(n) \\ &\approx \ldots \\ &\approx \mathbf{w}_i(n-1)^T\mathbf{x}_1(n),\ \text{ for } i \geq 2. \end{aligned} \quad (11)$$

In other words, $y_i(n) \approx \mathbf{w}_i(n-1)^T\mathbf{x}(n)$ depends only on the input data at time n.

Second, we derive the following updating equation for $\mathbf{e}_i(n)$:

$$\mathbf{e}_i(n) = \begin{cases} \mathbf{x}(n-D) - \mathbf{w}_i(n-2)y_i(n-D),\ i = 1 \\ \beta\alpha_{i-1}(n)\mathbf{e}_{i-1}(n) - \mathbf{w}_i(n-2)y_i(n-D),\ i \geq 2. \end{cases} \quad (12)$$

where

$$\alpha_{i-1}(n) = \frac{1}{\beta + \mu_i(n-1)y_i^2(n-D)}$$

and $\mu_i(n-1) = \frac{1}{d_i(n-1)}$.

Finally, $\mu_i(n) = \frac{1}{d_i(n)}$ can be updated directly as

$$\mu_i(n) = \alpha_i(n)\mu_i(n-1). \quad (13)$$

The modified algorithm (to be called pipelined *DPASTd*) can then be summarized as follows:

$$\begin{aligned} &y_i(n) = \mathbf{w}_i(n-1)^T\mathbf{x}(n) \\ &z_i(n) = \mu_i(n-1)y_i(n-D)^2 \\ &\alpha_i(n) = \tfrac{1}{\beta + z_i(n)} \\ &\mathbf{e}_i(n) = \begin{cases} \mathbf{x}(n-D) - \mathbf{w}_i(n-2)y_i(n-D),\ i = 1 \\ \beta\alpha_{i-1}(n)\mathbf{e}_{i-1}(n) - \mathbf{w}_i(n-2)y_i(n-D),\ i \geq 2 \end{cases} \\ &\mu_i(n) = \alpha_i(n)\mu_i(n-1) \\ &\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + \mu_i(n)\mathbf{e}_i(n)y_i(n-D). \end{aligned}$$

Fig. 1 illustrates the simulation results of both *PASTd* and the pipelined *DPASTd* algorithms with system parameters $N = 10$, $D = 13$ and $r = 3$. The learning curves of all three eigenvector estimations are shown. It is seen from the figure that the pipelined *DPASTd* reaches almost the same MSEs with slight degradation in convergence speed.

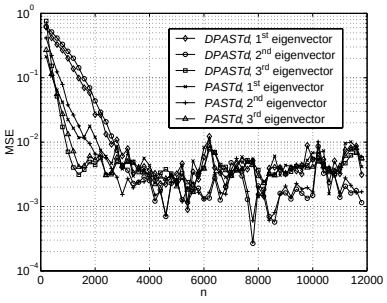

Fig. 1. Simulation results of *PASTd* and *DPASTd* algorithms.

4.2. The systolic implementation

A broader-sense 2-D systolic array implementation of the pipelined *DPASTd* algorithm can be designed by using retiming. Two different time scales are present in the pipelined *DPASTd* algorithm: the data sampling time "n" and the eigenvector index "i." Both are available for retiming. Although pipelining on the "i" scale does not reduce the overall time required for updating one eigenvector, it does improve the concurrency of the computations of multiple eigenvectors and, hence, it increases the system throughput. Therefore, we have pipelined on both time scales in our development. Fig. 2 shows an example of the resulting arrays.

In this example, we choose $D = N + 3$ and assume $r = 3$. Two types of registers are found in the array, corresponding to the two time scales in the algorithm. However, if the "n" registers (labeled with "T" in the figure) can be seen as external inputs, the array itself is driven exclusively by an "i" scale clock.

The function blocks in this implementation can be encapsulated into either internal cells or boundary cells. All internal cells have a similar structure, containing a multiplier, an adder, a register file and possibly a multiplexer. The only difference is the interconnection. Hence, it is possible to design a universal cell to realize them all, as shown in Fig. 3.

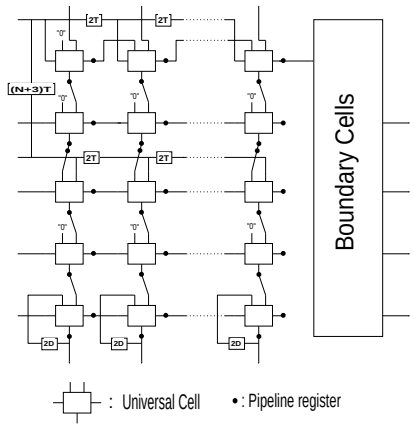

Fig. 2. A systolic realization of the pipelined *DPASTd* algorithm.

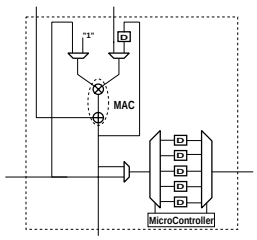

Fig. 3. The universal cell.

Most of the processing packaged in the boundary cells consists of only multiplications and additions. They can be realized by using either dedicated hardware or even the above universal cells. The only exception is the reciprocal operation needed in computing $\alpha_i(n)$. In the steady state, $\alpha_i(n)$ can be approximated by $\frac{1}{\beta} - \frac{1}{\beta^2} y_i^2(n)\mu_i(n)$, thus requiring no division. During the convergence period, however, this approximation is invalid, and during this period we propose to use a table look-up followed by one iteration of the Newton-Raphson method. That is, we look up in a ROM-table, indexed by the value of $y_i^2(n)\mu_i(n)$, an $\alpha_i^0(n)$ value, which is an initial approximation of $\alpha_i(n)$, and we then refine the estimation by using $\alpha_i(n) = \alpha_i^0(n)\left(2 - \alpha_i^0(n) z_i(n)\right)$ where $z_i(n) = y_i^2(n)\mu_i(n)$. This operation can be integrated into other boundary cells and can be pipelined.

5. CONCLUSION

In this paper, we have studied the *DPASTd* algorithm for estimating the complete signal subspace. We have shown that delayed updating does not affect the algorithm's local stability under a mild condition on the step size.

A systolic realization of the algorithm has also been discussed. The processing time of the structure is as low as, roughly, the delay of one MAC. It is suitable to be employed in a high-speed communication environment. More details of the systolic implementations can also be found in [8].

6. REFERENCES

[1] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Processing*, vol. 43, pp. 95–107, Jan. 1995.

[2] F. Xu and A. N. Willson, Jr., "A high-performance architecture for an RLS-like eigenvector algorithm," in *Proc. of ICSP2000*, pp. 559–562, 2000.

[3] B. Yang, "Asymptotic convergence analysis of the projection approximation subspace tracking algorithms," *Signal Processing*, vol. 50, pp. 123–136, April 1996.

[4] S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NJ: Prentice Hall, 3rd ed., 1996.

[5] A. Balakrishnan, *Introduction to Random Processes in Engineering*. New York: Wiley, 1995.

[6] A. I. Markushevich, *Theory of Functions of a Complex Variable*. New York: Chelsea, 1977.

[7] R. H. Kwong and E. Johnston, "A variable step size LMS algorithm," *IEEE Trans. Signal Processing*, vol. 40, pp. 1633–1642, July 1992.

[8] F. Xu and A. N. Willson, Jr., "Novel systolic architectures for signal subspace tracking," in *Proc. of MWSCAS2000*, 2000.