A PIPELINED FAST NEWTON TRANSVERSAL FILTER

George-Othon Glentis
Technologincal Education Institute of Crete, Branch at Chania
Department of Electronics, 3, Romanou Str, Halepa, Chania 73133, GREECE
Email: gglentis@chania.teiher.gr

ABSTRACT

In this paper a Pipelined Fast Newton Transversal Filter, (PFNTF), is presented for adaptive filtering and system identification. First, adaptation delay is introduced that allows for pipelining of the adaptive filter. Proper correction terms are subsequently utilized that compensate for the adaptation delay and provide results identical to the original FNTF algorithm, subject to an output delay. The performance of the proposed pipelined scheme is illustrated by computer simulation.

1 INTRODUCTION

The design of adaptive filters and system identification algorithms with optimum learning, in the sense of minimizing the accumulated squared error between the output signal and a desired response signal, has been the subject of major research for a long time, [1], [4]. Typical examples include the design of decision feedback equalizers in digital communications, the design of acoustical echo cancelers in hands-free telephony and in teleconferencing, etc.

Efficient Quasi-Newton (QN) algorithms have been recently proposed for adaptive filtering and system identification, (see [4] for an extended review). The Fast Newton Transversal Filter, (FNTF), [2], is one of the most widely used adaptive QN algorithm, that has been designed for efficient filtering and system identification of FIR models with long impulse response. Fast QN algorithms have low computational complexity, comparable to that of the LMS algorithm, yet the convergence performance of these methods, is comparable to that of the higher complexity fast RLS schemes.

Parallelism and pipelining in the computational flow of the FNTF algorithm is an issue related to performance, when high speed implementation of the algorithm on VLSI ASIC is required. Direct parallel or pipelined implementation of the FNTF algorithm is however prohibited, due to the inner product computations that are involved. Following [5]-[10], a pipelined architecture for the FNTF adaptive filter is proposed in this paper. Pipelining of the original FNTF adaptive filtering algorithm is obtained by introducing a proper amount of adaptation delay in the filtering error feedback loop. Proper correction terms are subsequently utilized that compensate for the adaptation delay and give results identical to the original FNTF algorithm, subject to an output delay (latency).

2 PIPELINING THE FNTF ALGORITHM

Let us consider the FNTF adaptive algorithm, for the case when the input signal $x(n)$, and the desired response signal $z(n)$, are hold up (delayed) by an amount of D time units. The resulting scheme is described by the following set of equations

$$\mathbf{C}(n) = \mathbf{C}(n-1) + \mathbf{W}(n-D)\epsilon(n) \tag{1}$$

$$\epsilon(n) = z(n-D) + \mathbf{X}^T(n-D)\mathbf{C}(n) \tag{2}$$

$\mathbf{X}(n) = [x(n)\ x(n-1)\dots x(n-m+1)]^T$ is the regressor vector. $\mathbf{C}(n)$ is the vector that carries the filter coefficients. m is the order of the FIR model under consideration. $\mathbf{W}(n)$ is the FNTF gain vector that determines the search direction. It is defined as, [2],

$$\lambda\mathbf{R}(n-1)\mathbf{W}(n) = -\mathbf{X}(n) \tag{3}$$

Matrix $\mathbf{R}(n-1)$ is computed in the FNTF sense, [2]. The output of the modified FNTF algorithm described above, is identical to the output of the original FNTF algorithm of [2], except from a output delay of size equal to D. From eqs.(1) and (2) it is readily seen that the a posteriori filtering error $\epsilon(n)$ can be estimated in terms of the a priori filtering error $e(n)$, according to the scheme

$$\epsilon(n) = e(n)/\alpha(n-D) \tag{4}$$

$$e(n) = z(n-D) + \mathbf{X}^T(n-D)\mathbf{C}(n-1) \tag{5}$$

where $\alpha(n) = 1 - \mathbf{C}^T(n)\mathbf{W}(n)$.

Pipelining of the FNFT algorithm is achieved by considering the following three steps: a) Pipelining of the filtering operation, which can accomplished by introducing an appropriate amount of time delay in order to overcome the associated inner product bottleneck. b) Pipelining of the gain computation part, and c) Efficient estimation of the correction terms introduced, so

that (a) provides output data identical (subject to a certain amount of output delay), to the original FNTF algorithm.

2.1 Pipelining the filtering part

Eq.(1) is backward expanded so that $\mathbf{C}(n)$ is expressed in terms of $\mathbf{C}(n-D)$, i.e.,

$$\mathbf{C}(n) = \mathbf{C}(n-D) + \sum_{i=0}^{D-1} \mathbf{W}(n-D-i)\epsilon(n-i) \quad (6)$$

Using eq.(6) in eq. (5) we obtain

$$e(n) = z(n-D) + \mathbf{X}^T(n-D)\mathbf{C}(n-D-1) + \Lambda_D(n) \quad (7)$$

where $\Lambda_D(n)$ is a correction factor defined as

$$\Lambda_D(n) = \sum_{i=1}^{D} r_i(n-D)\epsilon(n-i) \quad (8)$$

Notice that the inner product computation that appears in eq.(7) depends on $\mathbf{C}(n-D-1)$ instead of $\mathbf{C}(n-1)$. The presence of the delay in the computation of the inner product allows for pipelining of the computational flow. $\Lambda_D(n)$ is a correction factor, that is estimated in terms of the a posteriori errors $\epsilon(n-i)$ and the cross correlation terms $r_i(n)$, defined as,

$$r_i(n) = \mathbf{X}^T(n)\mathbf{W}(n-i) \quad i = 1, 2 \ldots D \quad (9)$$

2.2 Pipelining the gain computation part

The gain $\mathbf{W}(n)$ that appears in the filter updating equation (1) is computed in terms of the forward and backward predictors of low order $p << m$, according to the scheme, [2],

$$\begin{bmatrix} \mathbf{W}(n) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{W}(n-1) \end{bmatrix} - \begin{bmatrix} \mathbf{s}(n) \\ \mathbf{0}_{m-p} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{m-p} \\ \mathbf{u}(n^*) \end{bmatrix} \quad (10)$$

$$\mathbf{s}(n) = \begin{bmatrix} 1 \\ \mathbf{a}(n-1) \end{bmatrix} k^f(n), \quad \mathbf{u}(n) = \begin{bmatrix} \mathbf{b}(n-1) \\ 1 \end{bmatrix} k^b(n)$$

$$k^f(n) = \frac{e^f(n)}{\lambda\alpha^f(n-1)}, \quad k^b(n) = \frac{e^b(n)}{\lambda\alpha^b(n-1)}$$

Variables $\mathbf{a}(n)$ and $\mathbf{b}(n)$ denote the one-step ahead and the one step backwards linear predictors of order $p << m$, of the input signal $x(n)$. $e^f(n)$ and $e^b(n)$ are the corresponding a priori prediction errors. $\alpha^f(n)$ and $\alpha^b(n)$ are the forward and backward prediction error powers, respectively. n^* is set equal $n^* = n - m + p$. Finally, the gain power is recursively computed as

$$\alpha(n) = \alpha(n-1) + s_1(n)e^f(n) - u_{p+1}(n^*)e^b(n^*) \quad (11)$$

where $s_1(n)$ and $u_{p+1}(n)$ are the first and the last element of the corresponding vector variables, respectively.

Inspection of eq.(10) reveals that the computation of $\mathbf{W}(n)$ possesses an inherently pipeline structure. Thus, it is computed elementwise in a pipelineable way, as

$$W_i(n) = W_{i-1}(n-1) - s_i(n) \quad i = 1, 2 \ldots p+1$$
$$W_i(n) = W_{i-1}(n-1) \quad i = p+2, \ldots m-p-1$$
$$W_i(n) = W_{i-1}(n-1) + u_i(n^*) \quad i = m-p+1, \ldots m$$

where the initial value $W_0(n-1)$ is set equal to zero.

2.3 Computation of the cross-correlation terms

Using eqs. (9) and (10) it is easily shown that the following relationship holds, [3],

$$r_i(n) = r_i(n-1) - k^f(n-i)e_i^f(n) + k^b(n^*-i)e_i^b(n^*) \quad (12)$$

The forward and backward prediction errors appear above, are defined in terms of delayed predictors, as

$$e_i^f(n) = x(n) + \mathbf{x}^T(n-1)\mathbf{a}(n-i-1) \quad (13)$$

$$e_i^b(n) = x(n-p) + \mathbf{x}^T(n)\mathbf{b}(n-i-1) \quad (14)$$

$\mathbf{x}(n)$ is the regressor vector of order $p << m$ associated to the forward and backward prediction setup, i.e., $\mathbf{x}(n) = [x(n)\ x(n-1)\ \ldots x(n-p+1)]^T$. Clearly, $e^f(n) = e_0^f(n)$ and $e^b(n) = e_0^b(n)$. The LS forward and backward predictors are adapted in terms terms of the Kalman gain vector. Consider the update formulae

$$\mathbf{a}(n) = \mathbf{a}(n-1) + \mathbf{w}(n-1)\epsilon^f(n) \quad (15)$$

Replacing n by $n-i$ and solving for $\mathbf{a}(n-i-1)$ we get

$$\mathbf{a}(n-i-1) = \mathbf{a}(n-i) - \mathbf{w}(n-i-1)\epsilon^f(n-i) \quad (16)$$

Using eq.(16) in eq.(13), it follows that

$$e_i^f(n) = e_{i-1}^f(n) - \epsilon^f(n-i)q_i(n-1) \quad (17)$$

In a similar way, we get

$$e_i^b(n) = e_{i-1}^b(n) - \epsilon^b(n-i)q_i(n) \quad (18)$$

Cross-correlation terms $q_i(n)$ that appear above are defined as

$$q_i(n) = \mathbf{x}_p^T(n)\mathbf{w}_p(n-1) \quad (19)$$

They are recursively updated as, [3],

$$q_i(n) = q_i(n-1) - k^f(n-i)e_i^f(n) + k^b(n-i)e_i^b(n) \quad (20)$$

In order to have a recursive scheme, $q_i(n)$ should be estimated in terms of $e_{i-1}^b(n)$. Using eq.(18) in (20) we get

$$q_i(n) = \frac{q_i(n-1) - k^f(n-i)e_i^f(n) + k^b(n-i)e_{i-1}^b(n)}{1 + k^b(n)\epsilon^b(n)}$$

or,

$$q_i(n) = \frac{\alpha^b(n-1)}{\alpha^b(n)} \times \left(q_i(n-1) - k^f(n-i)e_i^f(n) + k^b(n-i)e_{i-1}^b(n)\right) \quad (21)$$

GIVEN

$\mathbf{a}(n)$, $\mathbf{b}(n)$, $e^f(n)$,$e^b(n)$, $\epsilon^f(n)$, $\epsilon^b(n)$ $\alpha^f(n)$, $\alpha^b(n)$

ESTIMATE

A. Gain Updating Part

$$k^f(n) = \frac{e^f(n)}{\lambda\alpha^f(n-1)}, \quad k^b(n) = \frac{e^b(n)}{\lambda\alpha^b(n-1)}$$

$$\mathbf{s}(n) = \begin{bmatrix} 1 \\ \mathbf{a}(n-1) \end{bmatrix} k^f(n), \quad \mathbf{u}(n) = \begin{bmatrix} \mathbf{b}(n-1) \\ 1 \end{bmatrix} k^b(n)$$

$$W_i(n) = W_{i-1}(n-1) - s_i(n) \quad \underline{i = 1, \ldots p+1}$$

$$W_i(n) = W_{i-1}(n-1) \quad \underline{i = p+2, \ldots m-p-1}$$

$$W_i(n) = W_{i-1}(n-1) + u_i(n^*) \quad \underline{i = m-p+1, \ldots m}$$

$$\alpha(n) = \alpha(n-1) + s_1(n)e^f(n) - u_{p+1}(n^*)e^b(n^*)$$

C. Correction Term Updating Part

$$\underline{\text{For} \quad i = 1 \quad \text{to} \quad D}$$

$$e_i^f(n) = e_{i-1}^f(n) - \epsilon^f(n-i)q_i(n-1)$$

$$q_i(n) = \frac{\alpha^b(n-1)}{\alpha^b(n)} \times$$

$$\left(q_i(n-1) - k^f(n-i)e_i^f(n) + k^b(n-i)e_{i-1}^b(n)\right)$$

$$e_i^b(n) = e_{i-1}^b(n) - \epsilon^b(n-i)q_i(n)$$

$$r_i(n) = r_i(n-1) - k^f(n-i)e_i^f(n) + k^b(n^*-i)e_i^b(n^*)$$

$$\underline{\text{End} \quad i}$$

$$\Lambda_D(n) = \sum_{i=1}^{D} r_i(n-D)\epsilon(n-i)$$

C. Filtering Part

$$y_D(n) = -\mathbf{X}^T(n)\mathbf{C}(n-1)$$

$$e(n) = z(n-D) + y_D(n-D) + \Lambda_D(n)$$

$$\epsilon(n) = e(n)/\alpha(n-D)$$

$$\mathbf{C}(n) = \mathbf{C}(n-1) + \mathbf{W}(n-D)\epsilon(n)$$

Table 1: The Pipelined Fast Newton Transversal Filter (PFNTF) adaptive algorithm.

2.4 Overall organization

The basic recursions developed so far, can be utilized for the pipelined implementation of the FNTF algorithm. A suitable pipelined algorithm for the computation of the information required at each step by the FNTF scheme (i.e., the forward and backward predictors $\mathbf{a}(n)$, $\mathbf{b}(n)$, the prediction error power variables, $\alpha^f(n)$, $\alpha^b(n)$, the a priori and the a posteriori prediction errors, $e^f(n)$, $e^b(n)$, $\epsilon^f(n)$, $\epsilon^b(n)$), is also required. These variables can be computed by an adaptive pipelined algorithm, such as the adaptive Schur-Levinson algorithm, or the adaptive lattice algorithm, followed by an adaptive Levinson part, [1]. Both schemes have low $O(p^2)$ complexity.

The pipelined FNTF algorithm is summarized in Table 1. The complexity of the algorithm is

$$C_{PFNTF} = 2m + 6D + O(p^2) \quad \text{MADS}$$

where, the second term is due to the computations involved into the correction terms and the last term corresponds to the complexity of the utilized pipilined low order LS predictors estimation scheme.

2.5 Pipelined architecture

The signal flow graph of the PFNTF algorithm is depicted in Figure 1, for the special case of $m = 10$ and $p = 3$. The pipelined architecture consists of the following parts:

a) The pipelined RLS preprocessing part. It can be implemented by a pipelined adaptive algorithm, such as the adaptive Schur-Levinson algorithm, or the adaptive lattice algorithm, followed by an adaptive Levinson part. The number of pipelined latches required for the pipelining of this part, is $D_f = O(p)$.

b) The gain computation part.

c) The filtering update part.

d) The filtering error estimation part. This can be pipelined by retiming the existing D delay elements. The exact number of D depends on the pipelining strategy adopted, and varies from the minimum of $D = [\log_2(m)]+1$, when a binary tree adder is utilized, to the maximum of $D = m-1$ when full pipelining is required.

e) The correlation-correction part. The pipelining of the correlation estimation part requires an amount of $D_r = 4$ additional delays. The correction part is implemented by a transposed form FIR filter of order D.

The critical path of the PFNTF algorithm is $T_C = \max\{t_M + t_A, t_D\}$ time units, where t_M, t_A and t_D is the time required for a single multiplication, addition and division respectively. The latency of the pipelined architecture is $D_{out} = D_f + D + D_r$.

3 SIMULATION RESULTS

The performance of the pipelined FNTF is illustrated by a typical system identification experiment. Consider an FIR filter of order $M = 256$. The impulse response was a typical impulse response of the acoustic echo path of a car enclosure. The input signal was a synthetic speech signal simulated by a stationary AR process of order 14, resulting to an eigenvalue spread of the sampled autocorrelation matrix of the input signal of the order $O(10^3)$, [4]. At the output of the FIR system white gaussian noise was added, resulting to an SNR equal to about 30dB. The system changes at time instant $n = 25000$ from $\mathbf{C}_o$ to $-\mathbf{C}_o$. The unknown FIR system was estimated using: a) the FNTF algorithm, and b) the proposed pipelined PFNTF algorithm, where the adaptation delay was set equal to $D = 20$. In both cases, we set $m = 256$, $p = 14$, $\lambda = .998$. The simulation results are shown in Figure 2. The output of the pipelined PFNTF algorithm equals to that of the original FNTF algorithm, except from a time delay, equal to D. The difference signal, $e_{FNTF}(n-D) - e_{PFNTF}(n)$, is depicted in Figure 2.

4 CONCLUSIONS

A pipelined implementation of the Fast Newton Transversal Filter has been presented in this paper. Pipelining of the original adaptive scheme has been

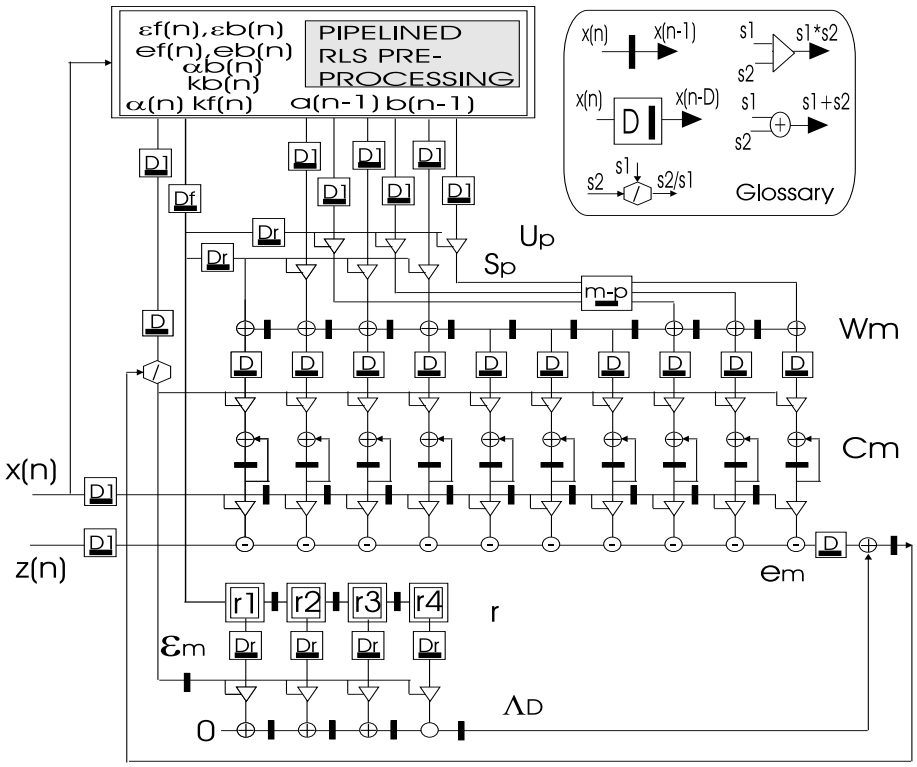

Figure 1. The PFNTF algorithm, (m=10,p=3). Thick lines represent multichannel data busses, while thin lines represent scalar data busses. ($D1 = D_f + D$)

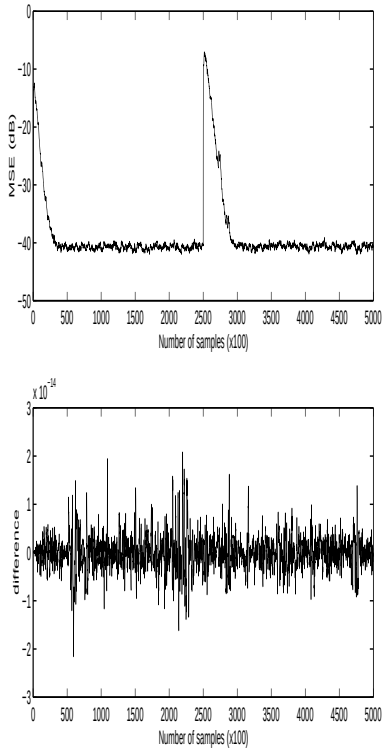

Figure 2. Simulation results

managed, by holding up the input and output signals and by proper algebraic manipulation of the delayed algorithm. Thus, adaptation delay has been introduced that allows for pipelining of the adaptive filter. Proper correction terms have been utilized, compensating for the adaptation delay and providing results identical to the original FNTF algorithm, subject to an output delay. The critical path of the pipelined scheme is reduced to $\max\{t_M + t_A, t_D\}$ time units. The performance of the proposed pipelined scheme has been illustrated by computer simulation.

References

[1] N. Kalouptsidis, and S. Theodoridis, eds, Adaptive system identification and signal processing algorithms, Prentice Hall 1993.

[2] G. Moustakides, and S. Theodoridis, 'Fast Newton transversal algorithms- A new class of adaprive estimation algorithms,' IEEE Trans. on Signal Processing, vol. 39, no. 10, pp. 2184-2193, Oct. 1991.

[3] K. Berberidis, S. Theodoridis, "A New Fast Block Adaptive Algorithm", IEEE Trans. on Signal Processing, pp. 75-87, Jan. 1999.

[4] G. Glentis, K. Berberidis, and S. Theodoridis, 'Efficient least squares adaptive algorithms for FIR transversal filtering: a unified view,' IEEE Signal Processing Magazine, pp. 13-42, July 1999.

[5] G. Long, F. Ling, and J. Proakis, 'The LMS algorithm with delayed coefficients adaptation, 'IEEE Trans. Acoust. Speech, Signal Processing, pp. 1397-1405, Sept. 1989; (pp. 230-232, Jan. 1992 corrections).

[6] S. Douglas, Q. Zhu, and K. Smith, 'A pipelined LMS adaptive FIR filter architecture without adaptation delay,' IEEE Trans. Signal Processing, pp. 775-779, March 1998.

[7] K. Matsubara, K. Nishikawa, and H. Kiya, 'Pipelined LMS adaptive filter using a new look-ahead transformation,' IEEE Trans. Circuits Sys. II, pp. 51-55, Jan. 1999.

[8] R.D. Poltmann, 'Conversion of the delayed LMS algorithm into the LMS algorithm, IEEE Signal Proc. Letter, vol. 2, pp. 223, Dec. 1995.

[9] G.O. Glentis, 'A pipelined TDLMS adaptive filter,' ICASSP-01, Salt Lake City, USA, May 7-11, 2001

[10] G.O. Glentis, 'An efficient pipelined LMS algorithm for Volterra system identification,' NISP-01, Baltimore, USA, June 3-6, 2001