both figures show the frequency spectrum of $\mathbf{V}_m$, $\mathbf{V}_a$, $\mathbf{V}_d$, and $\mathbf{V}_0$, respectively. These spectrums are plotted for the signals long after the algorithms converge. Parts (e) and (f) of both figures show the learning curves of α and β, respectively. As expected, the tracking of α using the LMS is relatively slow [see Fig. 4(e)]. Before α in Fig. 4(e) converges to the optimal value, β in Fig. 4(e) tries to respond to the incorrect α, which leads β to drift away from the correct initial value. After α converges, β also converges, but slowly. We can clearly see the gradient noise of both parameters α and β in Fig. 4(e) and (f), respectively, after both parameters converge. On the other hand, Fig. 5(e) shows that α in the proposed algorithm converged much faster, and furthermore, the tracking noise smooths out as time passes. Since α in Fig. 5(e) converges quite fast, β in Fig. 5(f) does not drift far from the initial value.

REFERENCES

[1] L. Sundstrom, M. Faulkner, and M. Johansson, "Quantization analysis and design of a digital predistortion linearizer for RF power amplifiers," *IEEE Trans. Veh. Technol.,* vol. 45, pp. 707–719, Nov. 1996.

[2] M. A. Briffa and M. Faulkner, "Stability analysis of Cartesian feedback linearization for amplifiers with weak nonlinearities," in *Proc. Inst. Elect. Eng. Commun.,* vol. 143, pp. 212–218, Aug. 1996.

[3] E. E. Eid and F. M. Ghannouchi, "Adaptive nulling loop control for 1.7-GHz feedforward linearization systems," *IEEE Trans. Microwave Theory Tech.,* vol. 45, pp. 83–86, Jan. 1997.

[4] S. J. Grant, J. K. Cavers, and P. A. Goud, "A DSP controlled adaptive feedforward amplifier linearizer," in *5th Int. Conf. Universal Personal Communications,* Cambridge, MA, 1996, vol. 2, pp. 788–792.

[5] G. Zhao, F. M. Ghannouchi, F. Beauregard, and A. B. Kouki, "Digital implementations of adaptive feed-forward amplifier linearization techniques," in *IEEE MTT-S Int. Microwave Symp. Dig.,* New York, 1996, vol. 2 of 3, pp. 543–546.

[6] S. Haykin, *Adaptive Filter Theory,* 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[7] J. K. Cavers, "Adaptation behavior of a feedforward amplifier linearizer," *IEEE Trans. Veh. Technol.,* vol. 44, pp. 31–40, Feb. 1995.

Efficient Block-Adaptive Parallel-Cascade Quadratic Filters

Linshan Li and V. John Mathews

***Abstract*—This brief presents computationally efficient block-adaptive algorithms for quadratic filters employing parallel-cascade realizations of the system model. Parallel-cascade realizations implement higher order Volterra systems using a parallel connection of multiplicative combinations of lower order systems. Such realizations are modular and therefore well-suited for very large scale integrate circuit implementation. They also permit efficient approximations of truncated Volterra systems. Mixed frequency- and time-domain realizations of the least-mean-square (LMS) adaptive filter, as well as that of a normalized LMS adaptive filter, are presented in this brief. The adaptive normalized LMS parallel-cascade quadratic filter has the advantages of computational simplicity and superior performance over its direct form, and unnormalized adaptive parallel-cascade counterparts.**

***Index Terms*—Adaptive filters, frequency-domain adaptation, nonlinear filters.**

I. INTRODUCTION

This brief introduces computationally efficient block-adaptive parallel-cascade truncated Volterra filters implemented in mixed frequency and time domain. Parallel-cascade realizations implement higher order Volterra systems as parallel combinations of multiplicative connections of lower order systems [1]. Stochastic gradient adaptive filters employing such structures have been presented in [1]–[3]. Frequency-domain implementations of direct-form adaptive Volterra filters were presented in [4]. Experimental results demonstrating that the mixed-domain realizations of parallel-cascade adaptive volterra filters exhibit faster convergence speeds and lower computational complexity than their direct-form counterpart are also included in this brief.

II. BLOCK-ADAPTIVE PARALLEL-CASCADE QUADRATIC FILTERS

Let $d(n)$ and $x(n)$ represent the desired response and input signals, respectively, of the adaptive filter. For simplicity of presentation we assume that the adaptive filter employs a homogeneous quadratic system model with finite memory. Extensions to the higher order and inhomogeneous systems are straightforward. The adaptive filter computes an estimate of $d(n)$ using a parallel-cascade realization as

$$\hat{d}(lN+n) = \sum_{i=1}^{r} \sigma_i(l)\mathbf{x}^T(lN+n)\mathbf{u}_i(l)\mathbf{u}_i^T(l)\mathbf{x}(lN+n). \quad (1)$$

That is, the output of the homogeneous quadratic filter is obtained as a weighted sum of the squared outputs of r linear filters as shown in Fig. 1. In (1), the variable l represents the block number, $(\cdot)^T$ represents matrix transpose and N represents the offset in number of samples between two adjacent blocks of data. The vector $\mathbf{u}_i(l)$ denotes the coefficients of the ith linear filter during the lth block,

Manuscript received October 28, 1997; revised October 13, 1998. This paper was recommended by Associate Editor M. Simaan. This paper was presented in part at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Seattle, WA, May 1998.

L. Li is with the College of Marine Engineering, Northwestern Polytechnical University, Xi'an 710072, China.

V. J. Mathews is with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112 USA.

Publisher Item Identifier S 1057-7130(99)02361-7.

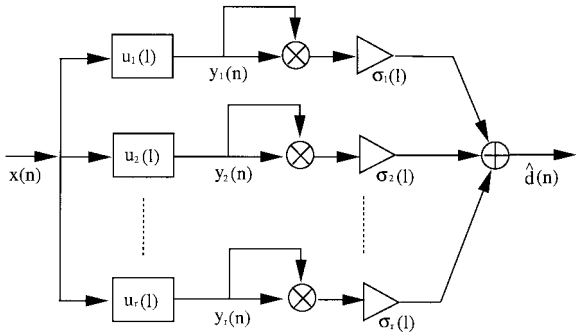

Fig. 1. Parallel-cascade realization of a homogenous quadratic filter.

and the parameter $\sigma_i(l)$ is the weighting factor for the squared values of this filter. For simplicity, we assume that the number of coefficients in each filter, as well as the number of output samples evaluated during each block, is identical to N. The input vector $\mathbf{x}(n)$ is defined as $\mathbf{x}(n) = [x(n)\ x(n-1)\ \cdots\ x(n-N+1)]^T$. In what follows, we denote the output of the linear filters at time $lN+n$ as $y_i(lN+n) = \mathbf{x}^T(lN+n)\mathbf{u}_i(l)$. The coefficients of a block-adaptive LMS parallel-cascade quadratic filter that attempts to reduce

$$J_l = \sum_{n=0}^{N-1} e^2(lN+n) = \sum_{n=0}^{N-1}\left(d(lN+n) - \sum_{i=1}^{r}\sigma_i(l)y_i^2(lN+n)\right)^2 \tag{2}$$

during each iteration can be shown to be updated as

$$\sigma_i(l+1) = \sigma_i(l) + \mu\sum_{n=0}^{N-1} e(lN+n)y_i^2(lN+n) \tag{3}$$

and

$$\mathbf{u}_i(l+1) = \mathbf{u}_i(l) + 2\mu\sum_{n=0}^{N-1}\sigma_i(l)e(lN+n)y_i(lN+n)\mathbf{x}(lN+n) \tag{4}$$

where μ is the step-size of the adaptive filter.

A. Block-Adaptive Normalized LMS (NLMS) Parallel-Cascade Filters

A time-domain NLMS adaptive filter for parallel-cascade realizations was recently derived in [3], and was shown to have significantly improved convergence properties over its direct-form LMS and parallel-cascade LMS counterparts. For block adaptation, we can derive a similar algorithm as follows. Let

$$\sigma_i(l+1) = \sigma_i(l) + \Delta\sigma_i(l)$$

and

$$\mathbf{u}_i(l+1) = \mathbf{u}_i(l) + \Delta\mathbf{u}_i(l) \tag{5}$$

be the coefficients updated during the iterations on the ith branch for the lth block of data. We wish to choose $\Delta\sigma_i(l)$ and $\Delta\mathbf{u}_i(l)$, such that

$$d(lN+n) = \sum_{i=1}^{r}(\sigma_i(l)+\Delta\sigma_i(l))\overline{y}_i^2(lN+n), \qquad n = 0, 1, \cdots, N-1 \tag{6}$$

where $\overline{y}_i(lN+n) = (\mathbf{u}_i(l)+\Delta\mathbf{u}_i(l))^T\mathbf{x}(lN+n)$. Our goal is to solve for $\Delta\sigma_i(l)$ and $\Delta\mathbf{u}_i(l)$ by minimizing the magnitude of the increments defined as

$$\sum_{i=1}^{r}(\|\Delta\sigma_i(l)\|^2 + \|\Delta\mathbf{u}_i(l)\|^2) \tag{7}$$

subject to the equality of (6). For this purpose, we define a cost function

$$C(l) = \sum_{i=1}^{r}(\|\Delta\sigma_i(l)\|^2 + \|\Delta\mathbf{u}_i(l)\|^2) + \lambda(l) \cdot \sum_{m=0}^{N-1}\left(d(lN+m) - \sum_{i=1}^{r}(\sigma(l)+\Delta\sigma_i(l))\overline{y}_i^2(lN+m)\right) \tag{8}$$

where $\lambda(l)$ is a Lagrange multiplier. The above formulation employs a single constraint involving the sum of the estimation errors in a block in the formulation of the optimization problem. Even though this constraint can be met by choices of coefficients that do not force the error values at each instant to be zero, this formulation results in a relatively simple adaptive filter structure that works well in practice.

Taking the partial derivative of (8) with respect to $\Delta\sigma_i(l)$ and $\Delta\mathbf{u}_i(l)$, and equating the results to zero gives

$$\frac{\partial C(l)}{\partial\Delta\sigma_i(l)} = 2\Delta\sigma_i(l) - \lambda(l)\sum_{m=0}^{N-1}\overline{y}_i^2(lN+m) = 0 \tag{9}$$

and

$$\frac{\partial C(l)}{\partial\Delta\mathbf{u}_i(l)} = 2\Delta\mathbf{u}_i(l) - 2\lambda(l)\sum_{m=0}^{N-1}(\sigma_i(l)+\Delta\sigma_i(l)) \cdot \overline{y}_i(lN+m)\mathbf{x}(lN+m) = 0. \tag{10}$$

Solving for $\Delta\sigma_i(l)$ and $\Delta\mathbf{u}_i(l)$ from (9) and (10) requires the simultaneous solution of a set of $2r$ coupled nonlinear equations. To derive a simpler solution, we assume that the coefficients experience only small changes during the updates, so that we can employ the approximate relationships $\sigma_i(l) \approx \sigma_i(l) + \Delta\sigma_i(l)$ and $\mathbf{u}_i(l) \approx \mathbf{u}_i(l) + \Delta\mathbf{u}_i(l)$ in appropriate parts of the derivation. Even though this approximation can be rigorously justified only during the final stages of adaptation, and for low noise levels in the desired response signal, experimental results have shown that the resulting algorithms provide fast convergence behavior at the initial stages of the operation of the adaptive filter as well as in cases when the noise levels are moderately large.

Multiplying (9) with $\overline{y}_i^2(lN+n)$ and adding $2\sigma_i(l)\overline{y}_i^2(lN+n)$ to the resulting equation gives

$$2(\sigma_i(l)+\Delta\sigma_i(l))\overline{y}_i^2(lN+n) = \lambda(l)\sum_{m=0}^{N-1}\overline{y}_i^2(lN+m)\overline{y}_i^2(lN+n) + 2\sigma_i(l)\overline{y}_i^2(lN+n). \tag{11}$$

We now use $\sigma_i(l)$ and $\mathbf{u}_i(l)$ to approximate $(\sigma_i(l)+\Delta\sigma_i(l))$ and $(\mathbf{u}_i(l)+\Delta\mathbf{u}_i(l))$, respectively, on the right-hand side of the above equation, and add the results over all i to get

$$2e(lN+n) = \lambda(l)\sum_{i=1}^{r}\sum_{m=0}^{N-1}y_i^2(lN+m)y_i^2(lN+n). \tag{12}$$

The derivation of this step assumes that

$$\sum_{i=1}^{r}2(\sigma_i(l)+\Delta\sigma_i(l))\overline{y}_i^2(lN+n) = 2d(lN+n). \tag{13}$$

According to our approximation

$$\sum_{i=1}^{r}2\sigma_i(l)\overline{y}_i^2(lN+n) \approx \sum_{i=1}^{r}2\sigma_i(l)y_i^2(lN+n) = 2\hat{d}(lN+n). \tag{14}$$

Substituting (13) and (14) in (11) results in (12). Adding (12) over all n in the block gives

$$2\sum_{n=0}^{N-1} e(lN+n) = \lambda(l)\sum_{i=1}^{r}\left(\sum_{m=0}^{N-1} y_i^2(lN+m)\right)^2. \tag{15}$$

Let

$$\mathbf{y}_i(l) = [y_i(lN),\, y_i(lN+1),\, \cdots,\, y_i(lN+N-1)]^T. \tag{16}$$

Then, (15) can be expressed as

$$2\sum_{n=0}^{N-1} e(lN+n) = \lambda(l)\sum_{i=1}^{r}\|\mathbf{y}_i(l)\|^4. \tag{17}$$

A similar derivation also gives the approximate relationship

$$\sum_{n=0}^{N-1} e(lN+n) = \lambda(l)\sum_{i=1}^{r}\sigma_i^2(l)\left\|\sum_{m=0}^{N-1} y_i(lN+m)\mathbf{x}(lN+m)\right\|^2. \tag{18}$$

Let $\mathbf{X}(l) = [\mathbf{x}(lN), \mathbf{x}(lN+1), \cdots, \mathbf{x}(lN+N-1)]$. Then, (18) can be expressed as

$$\sum_{n=0}^{N-1} e(lN+n) = \lambda(l)\sum_{i=1}^{r}\sigma_i^2(l)\|\mathbf{X}(l)\mathbf{y}_i(l)\|^2. \tag{19}$$

Combining (17) with (19) and solving for $\lambda(l)$ gives

$$\lambda(l) = \frac{3\sum_{n=0}^{N-1} e(lN+n)}{\sum_{i=1}^{r}\|\mathbf{y}_i(l)\|^4 + \sum_{i=1}^{r}\sigma_i^2(l)\|\mathbf{X}(l)\mathbf{y}_i(l)\|^2}. \tag{20}$$

We can solve for $\Delta\sigma_i(l)$ and $\Delta\mathbf{u}_i(l)$ from (9) and (10), respectively. We once again invoke the assumption of slow coefficient variation, and get a pair of realizable update equations by replacing $\overline{y}_i^2(lN+n)$ and $(\sigma_i(l)+\Delta\sigma_i(l))$ in the solutions so obtained by $y_i^2(lN+n)$ and $\sigma_i(l)$, respectively.

Employing the resulting expressions in the coefficient adaption equations may create large fluctuations in the coefficients. Consequently, the normalized block LMS adaptive parallel-cascade filter changes the coefficients a fraction of the distance suggested by the solution to the constrained optimization problem. Thus, the coefficient updates are given by

$$\sigma_i(l+1) = \sigma_i(l) + \tfrac{1}{2}\,\mu\lambda(l)\sum_{m=0}^{N-1} y_i^2(lN+m) \tag{21}$$

and

$$\mathbf{u}_i(l+1) = \mathbf{u}_i(l) + \mu\lambda(l)\sum_{m=0}^{N-1}\sigma_i(l)y_i(lN+m)\mathbf{x}(lN+m). \tag{22}$$

Finding a bound on the step-size for stable operation of the adaptive filter is a difficult problem. However, we can argue heuristically that choosing the step-size in the range $0 < \mu < 2$ will cause the average value of the squared estimation error in each block to be smaller than the corresponding value prior to the coefficient update, indicating that the choice of the step-size in the above range should ensure stable operation of the adaptive filter.

B. Mixed Frequency- and Time-Domain Realizations

In order to derive the most efficient realizations, we have used a mixture of time-domain and frequency-domain implementation of the components of the adaptive filters in what follows.

1) Adaptive LMS Parallel-Cascade Filter: Let us define the lth block of the input and desired response signals as

$$x_l(n) = x((l-1)N+n), \qquad 0 \le n \le 2N-1 \tag{23}$$

and

$$d_l(n) = \begin{cases} 0, & 0 \le n \le N-1 \\ d((l-1)N+n), & N \le n \le 2N-1 \end{cases} \tag{24}$$

respectively. Also, let $X_l(k)$ and $D_l(k)$ represent the $2N$-point discrete Fourier transform (DFT) of $x_l(n)$ and $d_l(n)$, respectively. Finally, let $U_{i,\,l}(k)$ denote the $2N$-point DFT of the coefficients of the linear filter in the ith branch of the parallel-cascade realization. We note here that the memory span of the filters is N samples each, and therefore, the $2N$-point DFT is computed after appropriate zero padding. With these definitions, we can compute the estimation error during $lN \le n \le (l+1)N-1$ in the following manner. Define $\tilde{y}_{i,\,l}(n)$ as the inverse DFT of $X_l(k)U_{i,\,l}(k)$. Using standard procedures, it can be shown straighforwardly that $y_i(lN+n) = \tilde{y}_{i,\,l}(n+N)\ 0 \le n \le N-1$. We can find an estimate of $d_l(n)$ as

$$\hat{d}_l(n) = \begin{cases} 0, & 0 \le n \le N-1 \\ \displaystyle\sum_{i=1}^{r}\sigma_i(l)\tilde{y}_{i,\,l}^2(n), & N \le n \le 2N-1. \end{cases} \tag{25}$$

The corresponding estimation error is

$$\tilde{e}_l(n) = d_l(n) - \hat{d}_l(n). \tag{26}$$

One can show that $e(lN+n) = \tilde{e}_l(n+N)\ 0 \le n \le N-1$.

Consider the update equations for the coefficients given by (3) and (4). We note that $\sigma_i(l)$ is a scalar element. There is no significant computational simplification possible by implementing (3) in the frequency domain. Let

$$g_{i,\,l}(m) = \begin{cases} -4\displaystyle\sum_{n=0}^{N-1}\sigma_i(l)e(lN+n)y_i(lN+n)x(lN+n-m), \\ \qquad 0 \le n \le N-1 \\ 0, \quad N \le 2N-1 \end{cases} \tag{27}$$

denote the mth element of the gradient vector computed during the lth block. Using arguments similar to the derivation of frequency-domain direct-form adaptive Volterra filters in [4], we can show that (4) can be implemented in the frequency domain using the overlap-save technique in the following manner.

Let us define a windowed version of the product signal

$$\tilde{e}_{y_{i,\,l}}(n) = \begin{cases} 0, & 0 \le n \le N-1 \\ \tilde{e}_l(n)\tilde{y}_{i,\,l}(n), & N \le n \le 2N-1. \end{cases} \tag{28}$$

Let $\tilde{E}_{y_{i,\,l}}(k)$ denote the $2N$-point DFT of $\tilde{e}_{y_{i,\,l}}(n)$. Then

$$g_{i,\,l}(m) = \begin{cases} \dfrac{-4\sigma_i(l)}{2N}\displaystyle\sum_{k=0}^{2N-1}\tilde{E}_{y_{i,\,l}}(k)X_l^*(k)e^{j(2\pi/2N)km}, \\ \qquad 0 \le m \le N-1 \\ 0, \quad N \le n \le 2N-1 \end{cases} \tag{29}$$

where $X_l^*(k)$ denotes the complex conjugate of $X_l(k)$. Let $\hat{\nabla}_{g_{i,\,l}}(k)$ represent the $2N$-point DFT of $g_{i,\,l}(m)$. Now, the coefficients of the linear filters can be updated in the frequency domain as

$$U_{i,\,l+1}(k) = U_{i,\,l}(k) - \frac{\mu_u}{2}\,\hat{\nabla}_{g_{i,\,l}}(k). \tag{30}$$

2) Adaptive NLMS Parallel-Cascade Filter: The only significant difference between the normalized and unnormalized algorithms is that the normalized algorithm requires calculation of the normalization factor $\lambda(l)$. As discussed earlier, we can implement the update equations for the scalar elements $\sigma_i(l)$ and $\lambda(l)$ efficiently in the time-domain. Let $f_{i,l}(m)$ denotes the mth element of the increment vector in (22) for the lth segment, i.e.,

$$f_{i,l}(m) = \sum_{n=0}^{N-1} \sigma_i(l) y_i(lN+n) x(lN+n-m). \tag{31}$$

Using the same arguments as before, we can calculate $f_{i,l}(m)$ in the frequency domain as follows. Let $Y_{i,l}(k)$ denote the $2N$-point DFT of the second N samples of $\tilde{y}_{i,l}(n)$, i.e.,

$$Y_{i,l}(k) = \sum_{n=N}^{2N-1} \tilde{y}_{i,l}(n)\, e^{-j(2\pi/2N)nk}. \tag{32}$$

Then

$$f_{i,l}(m) = \begin{cases} \dfrac{\sigma_i(l)}{2N} \displaystyle\sum_{k=0}^{2N-1} Y_{i,l}(k) X_l^*(k) e^{j(2\pi/2N)km}, \\ \qquad 0 \le m \le N-1 \\ 0, \quad N \le 2N-1. \end{cases} \tag{33}$$

We can now calculate the normalization factor $\lambda(l)$ using the previous results as

$$\lambda(l) = \frac{3 \displaystyle\sum_{n=0}^{N-1} \tilde{e}_l(n+N)}{\displaystyle\sum_{i=1}^{r} \left(\sum_{n=0}^{N-1} \tilde{y}_{i,l}^2(n+N) \right)^2 + \sum_{i=1}^{r} \sum_{m=0}^{N-1} f_{i,l}^2(m)}. \tag{34}$$

Now, the coefficient can be updated as

$$U_{i,l+1}(k) = U_{i,l}(k) + \mu \lambda(l) \hat{\nabla}_{f_{i,l}}(k) \tag{35}$$

where $\hat{\nabla}_{f_{i,l}}(k)$ represents the $2N$-point DFT of $f_{i,l}(m)$.

3) Computational Complexity: A careful count of the number of arithmetical operations required to implement the mixed-domain algorithms will show that both the algorithms require $O(rN \log_2(2N))$ complex operations per block that computes N output samples. The maximum complexity of the filters occurs when the number of branches r equals N, and in this case, the complexity of our filters is about half of the complexity of the frequency-domain realization of the direct-form filter in [4]. The complexity of the filters is significantly smaller than that of time-domain realizations of quadratic filters. Additional complexity reduction is possible by using approximate realizations that employ fewer number of branches than the maximum required number N.

III. Experimental Results

In the experiments presented here, the adaptive filters were employed to identify an unknown, homogeneous quadratic system with kernel given by

$$h_2(i,j) = \frac{1.5}{2\pi \left[1.5^2 + \left(i - \frac{63}{2} \right)^2 + \left(j - \frac{63}{2} \right)^2 \right]^{1.5}}, \quad 0 \le i,j \le 63 \tag{36}$$

using measurements of the input and output signals. The system model employed by the adaptive filter matched the unknown system exactly, and the data block size M was chosen to be 128, twice the system memory length. For the parallel-cascade realizations, the coefficients σ_i and the ith element of the linear filter coefficient vector $\mathbf{u}_i$ were initialized to one. The rest of the elements in $\mathbf{u}_i$ were initialized to zero. In order to get the same initial values of the mean-square estimation error in the comparisons, the coefficients of the direct-form LMS adaptive filter, implemented in the frequency domain and employing the same block size as the parallel-cascade filters, were initialized to the values corresponding to the initial coefficient values of the adaptive parallel-cascade filter. The input sequence was obtained as the output of a one-pole filter with transfer function $A(z) = 0.8/(1 - 0.6z^{-1})$ when the input to the filter was an i.i.d. Gaussian signal with zero mean value and variance equal to 0.1. The desired response signal $d(n)$ was obtained by corrupting the output of the unknown system with zero mean, white, and pseudo-Gaussian additive measurement noise with variance 0.001, which corresponds to an output signal-to-noise ratio of approximately 6.2 dB. All of the simulation results presented in this section are averages over 50 independent runs.

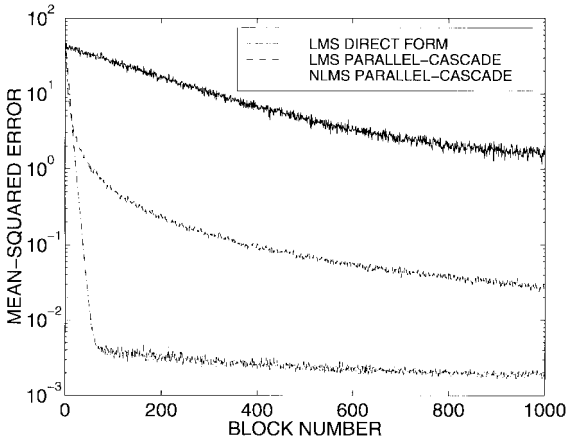

Fig. 2. Comparison of the MSE for three competing systems.

Fig. 2 displays the mean-square estimation error associated with the LMS and NLMS parallel-cascade systems, and the LMS direct-form filter for this problem when the step-size was chosen to be 1.408×10^{-5}, 5.791×10^{-4}, and 0.015 for the direct-form LMS, parallel-cascade LMS, and the NLMS parallel-cascade filters, respectively. These step-size values were chosen experimentally so as to get approximately the same steady-state excess mean-square error (MSE) for all three filters. The convergence behavior of the normalized direct-form LMS quadratic filter was much slower than that of its unnormalized counterpart, and therefore, the results are not included in this brief. We can see from the figure that the NLMS parallel-cascade realization converges much faster than the direct-form realization, as well as the unnormalized LMS parallel-cascade adaptive filter. Even though many of the approximations in the derivation of the NLMS parallel-cascade system can be rigorously justified only for low noise levels in the desired response signal, we see from these results that the NLMS adaptive filter works well for a variety of noise levels.

One of the advantages of the parallel-cascade realizations of Volterra systems is their ability to approximate systems with fewer number of branches. Fig. 3 shows the evolution of the MSE for three cases resulting from choosing the number of branches in the system to be 4, 16, and 64, when a step-size value of $\mu = 0.015$ was employed. From this figure, we can see that all three curves exhibit similar convergence behaviors, but reach slightly different steady-state values. We observe from these results that much reduced complexity without a significant loss of performance can be achieved with parallel-cascade approximations, at least for the system model

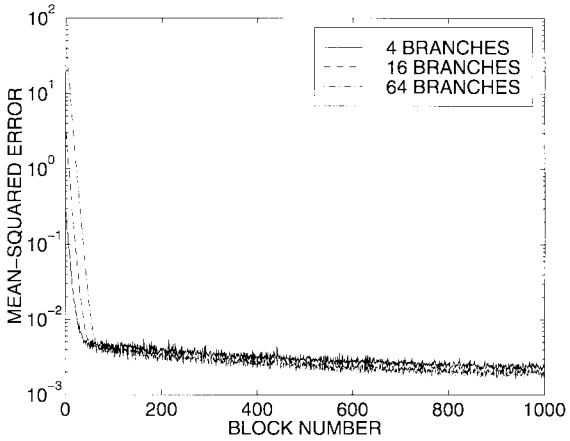

Fig. 3. MSE as a function of the number of branches retained in the parallel-cascade system.

employed in this experiment. Obviously, the performance of the reduced-complexity adaptive filter depends on the underlying system. However, many real world systems satisfy the property of rapidly decaying coefficient values with increasing lag values, as was the case in this example, and preliminary experiments we have conducted on approximating measured Volterra kernels from actual data have produced promising results. Such results are not included here because of space limitations.

IV. Conclusion

This brief presented mixed frequency- and time-domain realizations of adaptive parallel-cascade quadratic filters. The advantages of these algorithms are their reduced computational complexity and superior convergence performance. Derivations of computationally efficient block-adaptive realizations of truncated Volterra filters with higher orders of nonlinearity are straightforward extensions of the results presented in this brief.

References

[1] Y. Lou, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient VLSI array processing structures for adaptive quadratic digital filters," *Circuits, Syst., Signal Processing,* vol. 7, no. 2, pp. 253–273, Feb. 1988.

[2] S. Marsi and G. L. Sicuranza, "On reduced-complexity approximations of quadratic filters," in *Proc. 27th Asilomar Conf. Signals, Syst. Comput.,* vol. 2, pp. 1026–1030, Nov. 1993.

[3] T. M. Panicker, V. J. Mathews, and G. L. Sicuranza, "Adaptive parallel-cascade truncated Volterra filters," *IEEE Trans. Signal Processing,* vol. 46, pp. 2664–2673, Oct. 1998.

[4] S. Im and E. J. Powers, "A third-order frequency-domain adaptive Volterra filter," *IEEE Signal Processing Lett.,* vol. 4, pp. 75–78, Mar. 1997.

Discrete-Time Signals Arising from Simple Polynomial Sets

Dean J. Schmidlin

***Abstract*—The simple polynomial set is introduced as a mathematical tool for representing and generating discrete-time signals. Two benefits result. A signal derived from a simple polynomial set comes with an expression for its z-transform and a recurrence relation for its rapid computation. Two examples are presented. The purpose of these examples is to show that simple sets of polynomials can give rise to signals having more diverse properties than signals with rational z-transforms.**

***Index Terms*—Branch points, discrete-time signals, essential singularities, polynomial sets, time-varying difference equations, z-transforms.**

I. Introduction

The z-transform and the difference equation have been two of the important tools in the representation, analysis, and generation of discrete-time signals. In the special case of signals having rational z-transforms, the difference equation is a linear difference equation with constant coefficients. Exponential signals and damped and undamped sinusoids are the most common signals of this type. The z-transform of these signals has only poles as singularities. Associating linear constant-coefficient difference equations in the time domain with pole singularities in the z-domain leads us to anticipate that signals having more complex singularities, such as branch points and essential singularities, require linear difference equations with time-varying coefficients. For example, Kayhan [1] recently presented a time-varying coefficient difference equation representation of single- and multiple-chirp signals and used them to estimate instantaneous frequency and amplitude. He showed that a single sinusoid with a time-varying frequency and amplitude can be represented by a second-order difference equation with two time-varying coefficients. These coefficients are dependent on values of the instantaneous frequency and amplitude functions.

In studying the properties of discrete-time signals having non-rational z-transforms, it would be useful to have a mathematical tool that would not only provide a linear time-varying coefficient difference equation for the signal, but also furnish an expression for its z-transform. Such a mathematical tool is the simple polynomial set. There are a wealth of simple polynomial sets in the mathematical literature. Each polynomial set comes with a generating function and a pure recurrence relation. Section II defines a simple polynomial set and, by means of the Laguerre polynomials, demonstrates how a simple set of polynomials gives rise to a discrete-time signal together with its z-transform and a time-varying coefficient difference equation. Presented in Section III is a second example which, like the first example, shows that signals derived from simple polynomial sets can have more diverse properties than signals whose only singularities in the z-domain are poles. Finally, Section IV gives concluding remarks.

II. The Set of Laguerre Polynomials

A set of polynomials $\{p_n(w)\}; n = 0, 1, 2, \cdots$, is called a *simple set* if $p_n(w)$ is of degree precisely n in w so that the set contains

Manuscript received January 16, 1998; revised August 12, 1998. This paper was recommended by Associate Editor M. Simaan.

The author is with the University of Massachusetts, North Dartmouth, MA 02747-2300 USA.

Publisher Item Identifier S 1057-7130(99)02363-0.