<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Transform domain adaptive Volterra filter algorithm based onconstrained optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Zou, Y; Chan, SC; Ng, TS</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>1999</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/46116">http://hdl.handle.net/10722/46116</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>©1999 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
TRANFORM DOMAIN ADPTIVE VOLterra FILTER ALGORITHM BASED ON CONSTRAINED OPTIMIZATION

Yuexian Zou, Shing-Chow Chan and Tung-Sang Ng

Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong
{yzou, schan, tsng}@eee.hku.hk

ABSTRACT

In this paper, a transform domain normalised partially decoupled LMS (TP-LMS) algorithm is proposed based on the partially decoupled transform domain Volterra filter. It is formulated by applying an orthogonal transform to the first order of the partially decoupled Volterra filter, in which the filter weights of a given order are optimised independently of those in the higher order. This approach results in solving the minimum mean square error (MSE) filtering problem as a series of constrained optimisation problem and the modular structure order by order. Simulation of a system identification application indicates TP-LMS algorithm provides a trade-off between convergence speed and computational complexity.

1. INTRODUCTION

Nonlinear adaptive filtering techniques employing truncated Volterra series model have become a powerful tool in some important applications. Adaptive Volterra filter has been developed for echo cancellation [2], system identification [4], interference and noise reduction [3]. With the Volterra expansion, the nonlinear filter weights can be treated in a fashion analogous to linear filtering problem since there is a linear relationship between its filter output and filter weights. The widely used linear adaptive algorithm such as stochastic gradient descent (SGD) algorithms and recursive least square (RLS) algorithms can be directly applied to Volterra filter. However, the convergence speed of Volterra filter is exacerbated not only by the input signal statistics but also by the nonlinearity itself [5, 12]. Although RLS algorithm is insensitive to the condition number and has fast convergence, it has complexity of $O(N^3)$ in direct implementation and $O(N^5)$ in fast form implementation [6] for the quadratic Volterra filter realization with $N$-sample memory. The general lattice implementation of the Volterra filter suffered from the over-parameterised problem [4]. Another fast RLS algorithm for adaptive Volterra filter has been developed with $O(N^3)$ [6] by exploring the forward predictor error and backward predictor error. All of these fast algorithms suffer from the numerical unstable problem [10]. Recently, a time domain partially decoupled Volterra filter has been proposed in [7], in which its filter weights of a given order are optimised independently of those in the higher order. The Volterra filtering problem is formulated as a series of constrained optimization problem. The advantage of this approach is its possibility to increase the convergence speed by using different adaptive algorithm for different order. In [7], a new normal equation is derived and the partially decoupled LMS (P-LMS) algorithm is also developed which is with the similar form of the LMS algorithm but the stepsize bound can be computed straightforward.

In this paper, we propose a modification to the partially decoupled Volterra filter. We apply an orthogonal transform to the first order Volterra filter input signal as shown in Fig. 1.

2. BASIC THEORY

2.1 The Volterra Filter Theory

A nonlinear system can be modelled as a $N$-memory and the $Z$-order Volterra filter with the input-output relationship,

$$y(n) = H^T_z[Z]X_z(n, Z) .$$

where, $X_z(n, Z) = [x_1 (n), ..., x_j (n), ..., x_{j+1} (n)]^T$ denotes the $Z$-order Volterra filter observation vector and $H^T_z[Z] = [H_1[^{T}], ..., H_j[^{T}], ..., H_N[^{T}]]$ denotes the corresponding filter weight vector; $n$, $j$ are time and order index respectively; $X_z(n) = [x_1, x_2, ..., x_N]^T$ is the input vector of the system. $X_z(n)$, $H_z$ denote the $j$-th order input vector containing all the $j$th-order products of the elements in $X_z(n)$ and the corresponding filter weight vector, respectively; $Y_z(n)$ is the output of the system. For example, the second-order filter input vector can be described as,

$$X_z(n) = [x_1, x_2, ..., x_N]^T = [x_1, x_2, ..., x_1, x_2, ..., x_1, x_2]^T,$$

where, $N_z = N(N+3)/2$ with corresponding weight vector $H_z (n)$. Based on the Volterra expansion, $X_z (n)$ consists of $N_z = \binom{N+j-1}{j}$ elements, so does $H_z (n)$. Therefore, there are $M_z = \sum_{j=1}^{N_z} N_j$ elements in $X_z(n, Z)$ and $H_z(Z)$. The input correlation matrix and cross-correlation vector of the $Z$-order Volterra filter can be defined as,

$$R_z[Z] = E[X_z(n, Z)X_z^T(n, Z)] = \begin{bmatrix} R_{1,1} & \cdots & R_{1,z} \\ \vdots & \ddots & \vdots \\ R_{z,1} & \cdots & R_{z,z} \end{bmatrix} ,$$

$$P_z[Z] = E[d(n)X_z(n, Z)] = [P_{1}[^{T}], ..., P_{z}[^{T}]]^T .$$

where, $R_{i,j} = E[X_z(n)X_z^T(n)]$ is a correlation matrix of $X_z(n)$ and $X_z(n)$. $P_z[Z] = E[d(n)X_z(n, Z)]$ is cross-correlation vector of $d(n)$ and $X_z(n)$. Here, the Volterra filter problem can be solved as a linear filtering problem since there is a linear relationship between its output and the filter weights as shown in (1). The minimisation of the MSE results in the Wiener normal equation,

$$R_z[Z]H_z[Z] = P_z[Z].$$

This is a basic theory of the Volterra filter and its applications.
and the minimum mean square error (MMSE) is given by [10]:

\[ J_{\text{MMSE}} = \sigma_v^2 - P_1^T[Z]H_{\text{opt}}[Z] = \sigma_v^2 - (\mathbf{R}_1^2[Z]H_{\text{opt}}[Z])^T\mathbf{H}_{\text{opt}}[Z] \]

\[ = \sigma_v^2 - P_1^T[Z](\mathbf{R}_1^2[Z])^{-1}P_1[Z] \quad (6) \]

where, \( H_{\text{opt}}[Z] \) is the optimal solution of the Volterra filter in MSE sense; \( \sigma_v^2 = \mathbb{E}[d^2(n)] \) is the desired signal variance.

2.2 Partially Decoupled Volterra Filter [7]

The partially decoupled Volterra filter was proposed in [7], which optimises the given order filter weights independently of the higher order filter. For \( Z = 1 \), the first order filter weights \( H_1 \) is optimised independently of the higher order filter without any constrained condition. The MSE is defined as \( J[H_1] = \mathbb{E}[(d(n) - H_1^T X_1(n))^2] \) and the optimal filter weights can be obtained by \( H_{\text{opt}}[Z] = \mathbf{R}_0^2[Z] \). The quadratic Volterra filter is constructed by adding the second order filter on top of the first one. So, the Volterra filter weights \( H_2[Z] \) can be optimised by minimising the constrained MSE cost function,

\[ J[H_1,H_2,C_1] = \mathbb{E}[e^2(n)] + C_1^T (\mathbf{R}_1^2[H_1,H_2] - P_1) \quad (7) \]

Here, \( e(n) = d(n) - H_1^T X_1(n) \). \( C_1 \) is \( N \times 1 \) the Lagrange constant vector. The constrained minimum is obtained by taking partial gradients with respect to \( H_1,H_2 \) and \( C_1 \) in (7) and let them to zeros which results in a new normal equation

\[ \begin{bmatrix} \mathbf{R}_{1,1} & 0 \\ \mathbf{R}_{1,2} & \mathbf{R}_{2,2} \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \quad (8) \]

From (8), we can see that \( H_1 \) is decoupled from \( X_1(n) \) and \( H_2 \). Similarly for the \( Z \)-order Volterra filter, the constrained normal equation can be obtained [7]:

\[ \mathbf{R}_Z^2[Z] H_{\text{opt},Z}[Z] = P_1[Z] \quad (9a) \]

where,

\[ \mathbf{R}_Z^2[Z] = \begin{bmatrix} \mathbf{R}_{1,1} & 0 & \cdots & 0 \\ \mathbf{R}_{1,2} & \mathbf{R}_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1,Z} & \mathbf{R}_{2,Z} & \cdots & \mathbf{R}_{Z,Z} \end{bmatrix} \quad (9b) \]

Note that, the upper triangular entries of \( \mathbf{R}_Z^2[Z] \) vanish due to the \( Z-1 \) sets of constrained conditions and compared with (3),

\[ \mathbf{R}_Z[Z] = \mathbf{R}_Z^2[Z] + (\mathbf{R}_Z^2[Z])^T - Q \quad (10) \]

where \( Q = \text{diag} \{ \text{diag}(\mathbf{R}_Z[Z]) \} \) is a block-diagonal matrix.

2.3 Partially Decoupled SGD and LMS algorithm

In general, the direct solution of (9a) for higher order Volterra filter in many applications is formidable [6,7,10]. Obviously, many adaptive algorithms can be applied to get the solution of (9a). Without loss of generality, we choose \( Z = 2 \) as an example. The stochastic gradient descent (SGD) algorithm for the quadratic Volterra filter is [10],

\[ H_z[n+1] = H_z[n] + \frac{1}{2} \mu_z \left( -\frac{\partial J_{\text{MMSE}}}{\partial H_z} \right) = H_z[n] - \mu_z \tilde{V}_z(n) \quad (11) \]

\[ \mathbf{H}_z[n+1] = \mathbf{H}_z[n] + \frac{1}{2} \mu_z \left( -\frac{\partial J_{\text{MMSE}}}{\partial \mathbf{H}_z} \right) = \mathbf{H}_z[n] - \mu_z \mathbf{V}_z(n) \quad (12) \]

where, \( \mathbf{V}_z(n) = \mathbf{R}_{1,1} \mathbf{H}_z(n) - P_1, \quad \tilde{V}_z(n) = \mathbf{R}_{2,1} \mathbf{H}_z(n) - P_2 + \mathbf{R}_{2,2} \mathbf{H}_z(n) \) are the gradient vector. \( \mu_1, \mu_2 \) are two stepsizes. The range of the stepsizes \( \mu_z \) to guarantee convergence for the \( j \)-th order Volterra filter has been proved to be [7]:

\[ 0 < \mu_z < 2/\lambda_{\text{max}} \quad (13) \]

where, \( \lambda_{\text{max}} \) is the maximum eigenvalues of the submatrix \( \mathbf{R}_{ij} \) in \( \mathbf{R}_Z[Z] \). From (11), (12), we can see that, for quadratic Volterra filter, the SGD algorithm needs \( O(N^4) \) operations in each iteration to update \( \mathbf{H}_1, \mathbf{H}_2 \). In order to reduce the computational complexity, the LMS algorithm has been suggested[7], and gradient vectors \( \tilde{V}_z(n) = e(n) X_z(n) \), \( \tilde{V}_z(n) = e(n) X_z(n) \) replace gradient vectors \( V_z(n) \) and \( V_z(n) \) in (11), (12) respectively. The filter weights can be updated by

\[ \mathbf{H}_z[n+1] = \mathbf{H}_z[n] + \mu_z \tilde{V}_z(n) = \mathbf{H}_z[n] + \mu_z e(n) X_z(n) \quad (14) \]

\[ \mathbf{H}_z[n+1] = \mathbf{H}_z[n] + \mu_z \tilde{V}_z(n) = \mathbf{H}_z[n] + \mu_z e(n) X_z(n) \quad (15) \]

Equation (14), (15) are called the partially decoupled LMS (P-LMS) algorithm to distinguish the conventional LMS algorithm. Analysis also shown that the stepsizes satisfy the same conditions as showed in (13) for convergence of the algorithm [7].

3. TRANSFORM DOMAIN PARTIALLY ECOPULDED LMS (TP-LMS) ALGORITHM

The main problem of LMS and P-LMS algorithm is their slow convergence rate. In the present work, we focus on improving the convergence rate of P-LMS algorithm by a transform approach with power normalisation. The block diagram of transform domain adaptive \( Z \)-order Volterra filter is shown in Fig.1.

\[ \tilde{X}(n) = \mathbf{FX}(n) = [\tilde{x}_1, \ldots, \tilde{x}_N]^T \quad (16) \]

where, \( \tilde{x}_k = f_k^T X(n) = \sum_{s=1}^{L} f_{k,s} X_s(n) \) \( k = 2, \ldots, N \) and \( f_k = [f_{k,1}, \ldots, f_{k,s,1}] \) is the \( k \)-th row vector of the orthogonal transform matrix \( \mathbf{F} \) of rank \( N \). In particular, variables with the over bar represent the transform domain variables. Obviously, the transform domain Volterra filter weights can also be adapted by the P-LMS algorithm,

\[ \tilde{\mathbf{H}}_z[n+1] = \tilde{\mathbf{H}}_z[n] + \tilde{\mu}_z \tilde{V}_z(n) \tilde{X}_z(n) \quad (17) \]

\[ \tilde{\mathbf{H}}_z[n+1] = \tilde{\mathbf{H}}_z[n] + \tilde{\mu}_z \tilde{V}_z(n) \tilde{X}_z(n) \quad (18) \]

where, \( \tilde{\mathbf{H}}_z[n+1], \tilde{\mathbf{H}}_z[n+1] \) are the first and the second order weight vector at time \( n+1 \). \( \tilde{e}(n) = d(n) - \mathbf{R}_Z[Z]\tilde{X}_z(n) \) is the error signal. \( \tilde{X}_z(n) = \tilde{X}(n) \), \( \tilde{X}_z(n) \) contains all the second-order products of the elements in \( \tilde{X}(n) \). \( \tilde{\mu}_1, \tilde{\mu}_2 \) are two different stepsizes which satisfy the condition

\[ 0 < \tilde{\mu}_j < 2/\tilde{\lambda}_{\text{max}} \quad (j = 1, 2) \]

where, \( \tilde{\lambda}_{\text{max}} \) is the maximum eigenvalues of the \( \tilde{\mathbf{R}}_{ij} \) is \( \mathbb{E}[\tilde{X}_z(n) \tilde{X}_z(n)^T] \). The transform approach in Fig.1 will result in an equivalent non-orthogonal transform for the Volterra filter.
Based on the definition, the constrained Volterra filter can be obtained when.

\[ x(n) = b(n) + \sum_{i=1}^N \sum_{k=1}^p \alpha_{ik} x(n-i) \]

From (16), (20), we can get the relationship between \( \mathbf{X}_s(n, Z) \) and \( \mathbf{X}_r(n, Z) \).

\[ \mathbf{X}_s(n, Z) = \mathbf{B} \mathbf{X}_r(n, Z) = \begin{bmatrix} \mathbf{F} & 0 \\ \mathbf{F} \mathbf{R}_1 \mathbf{F}^T & \mathbf{R}_2 \end{bmatrix} \mathbf{X}_r(n, Z) \]

where, \( \mathbf{B} \) is a new equivalent linear transform matrix. Each element of \( \mathbf{F}_i \) can be computed from (20) according to the corresponding sequence order of \( \mathbf{X}_r(n, Z) \). It is easy to derive the following equation.

\[ \mathbf{R}_1(n, Z) = \mathbf{B} \mathbf{R}_1(n, Z) \mathbf{B}^T = \begin{bmatrix} \mathbf{F} & \mathbf{F} \mathbf{R}_1 \mathbf{F}^T & \mathbf{F} \mathbf{R}_2 \mathbf{F}^T \end{bmatrix} \]

In (22), \( \mathbf{R}_1(n, Z) \) is approximately diagonalized by orthogonal transform \( \mathbf{F} \). \( \mathbf{R}_1(n, Z) \) is a band matrix resulting from non-orthogonal transform \( \mathbf{F} \). With the power normalised approach, the transform domain partially decoupled normalised LMS (TP-LMS) algorithm can be directly derived from (17), (18)

\[ \mathbf{H}_r(n+1) = \mathbf{H}_r(n) + \sum_{i=1}^N \mathbf{R}_1(n, Z) \mathbf{X}_r(n, Z) \]

where, \( \mathbf{R}_1(n, Z) = \text{diag}(\mathbf{F}_1, \ldots, \mathbf{F}_N) \). We can estimate the signal power by a single-pole low pass filter as following in a recursive way [11],

\[ \mathbf{R}_1(n, Z) = \mathbf{F}_1 \mathbf{R}_1(n, Z) \mathbf{F}_1^T = \mathbf{F}_1 \mathbf{F}_1^T \mathbf{R}_1(n, Z) \mathbf{F}_1^T \mathbf{F}_1 \]

where, \( \alpha \in [0, 1] \) is a forgetting factor.

4. PERFORMANCE ANALYSIS

4.1 The Convergence Performance

The convergence conditions for P-LMS algorithm in time domain have been explored in [7]. For P-LMS algorithm, the stepsize need to satisfy the sufficient condition in (13). For TP-LMSA, it is well-known that the convergence speed can be improved over time domain approach by using the data dependent stepsize \( \mu_j \) in (23a), (23b) and \( \mu_j \) satisfies (19).

4.2 The Minimum Mean Square Error (MMSE)

Firstly, in this section, we will investigate the MMSE of the constrained approach comparing to the unconstrained one. We assume that the tap-input vectors of the different order of the Volterra filter and the desire signal are jointly stationary. Based on the definition, the constrained MMSE of \( Z \) order Volterra filter can be obtained when \( \mathbf{H}_r(n, Z) = (\mathbf{R}_1 Z)^{-1} \mathbf{P}_r(Z) \).

\[ J_{\text{MMSE}} = \sigma^2_r - \mathbf{P}_r(Z) \mathbf{H}_r(n, Z) + C_r(Z) \mathbf{P}_r(Z - 1) - \mathbf{P}_r(Z - 1) \]

Comparing (6) to (26), we can see that \( J_{\text{MMSE}} \) is determined by the \( \mathbf{R}_1(Z) \) which contains all information of \( \mathbf{X}_r(n, Z) \), but \( J_{\text{MMSE}} \) is by \( \mathbf{R}_2(Z) \) which loses some information between the cross-correlation between the different order. Only when the input vector is fully decoupled that means the \( \mathbf{R}_1(Z) \) is a diagonal matrix, we have \( J_{\text{MMSE}} = J_n \). Otherwise, there exist some performance degrade for partially decoupled approach.

Secondly, we consider the transform domain TP-LMS algorithm. It is easy to get the relationship between time domain and transform domain without the power normalisation.

\[ \mathbf{R}_r(Z) = \mathbf{E}[\mathbf{X}_r(n, Z) \mathbf{X}_r(n, Z)^T] = \mathbf{B} \mathbf{R}_1(Z) \mathbf{B}_r(Z) \]

Based on the definition, we can compute the MMSE as,

\[ \mathbf{J}_{\text{MMSE}} = \mathbf{J}_n - \mathbf{P}_r(Z)[\mathbf{R}_r(Z) - \mathbf{B} \mathbf{R}_1(Z) \mathbf{B}_r(Z)] \mathbf{P}_r(Z) \]

Comparing (28) to (26), \( \mathbf{J}_{\text{MMSE}} = J_n \), it means the transform \( \mathbf{B} \) does not change the MMSE just like the orthogonal transform does. The power normalisation process in orthogonal transform always increase the steady error [9, 12]. So, it can be predictable that the normalised non-orthogonal transform will also increase the steady error.

4.3 Computational Complexity

Finally, for TP-LMS algorithm, we discuss the computational complexity of quadratic Volterra filter. Fast recursive transform algorithm in general would require complexity \( O(N^2) \) [11]. The estimation of \( \lambda_j, j = 1, 2 \) in (23a), (23b) requires \( N(N + 5) \) multiplications, \( N(N + 3)/2 \) additions and \( N(N + 3)/2 \) divisions. The update of the filter weights in (23a), (23b) needs \( N(N + 3) + 2 \) multiplications and \( N(N + 3)/2 \) additions. The total computation requirement is therefore of order \( O(N^2) \) which is the same as the number of Volterra filter weights.

5. SIMULATION RESULTS

In this section, we present simulation studies using a system identification application to demonstrate the speed of convergence and MSE of TP-LMS algorithm. The adaptive filter has the same nonlinear structure as that of the unknown nonlinear system which can be described as

\[ y(n) = \text{[\ldots]} + 0.54x^2(n) - 1.6x^3(n) + 1.4x(n - 2) - 0.13x^3(n - 1) + 3.72x(n)(n - 1) + 1.86x(n)(n - 2) - 0.76x(n)(n - 3) - 0.76x(n - 1)(n - 2) - 0.12x(n - 1)(n - 3) - 1.52x(n - 2)(n - 3) \]

The coloured input signal \( x(n) \) satisfies the input-output relationship:

\[ x(n) = b(n) + \sqrt{1 - b^2} \cdot v(n) \]

where \( v(n) \) is Guassian process with zero-mean, variance unit, \( b = 9 \) is a constant parameter which decides the level of correlation between adjacent samples of \( x(n) \). The output SNR is defined as...
**SNR = 20 \log_{10}(\sigma_1^2/\sigma_2^2)** and is chosen equal to 20 dB; \( \mu, \mu_2, \bar{\mu}, \bar{\mu}_2 \) are chosen equal to 0.04, 0.02, 0.06, 0.02 respectively and \( F \) is chosen as DCT. The performance is evaluated by MSE and the normalised squared norm of the weight error (NSWE) [6],

\[
\text{NSWE} = 10 \log \frac{1}{n} \sum_{i=1}^{n} |h_i(n) - \hat{h}_i(n)|^2 \]

where, \( M_i = 14 \), \( k_i \) is the \( i^{\text{th}} \) unknown system coefficients in (29) and \( h_i(n) \) is the corresponding adaptive filter weight at time \( n \). The ensemble average MSE and the NSWE obtained by averaging over 20 independent runs are plotted in Fig.2 and Fig.3 respectively. Simulation results in Fig.2 and Fig.3 demonstrate that the proposed TP-LMS algorithm has a better convergence rate than that of P-LMS algorithm, but slower than that of RLS algorithm. In particular, from curves (2), (3) in Fig.2 and Fig.3, we can see that the steady error of TP-LMS algorithm is a slightly greater than that of P-LMS algorithm.

**6. CONCLUSION**

A new transform domain partially decoupled normalised LMS (TP-LMS) algorithm is derived in this paper. By applying an orthogonal transform on the first order filter, we constructed the so-called partially decoupled transform domain Volterra filter, where the optimal weights satisfy the constrained normal equation. Based on this structure, the equivalent transform matrix is formulated from the elements of the orthogonal transform following the Volterra expansion. Power normalisation is applied to improve the convergence speed. The advantage of this constrained approach lies on the modular filter structure and the possibility of applying different adaptation algorithm to different Volterra filter order. Simulation results indicated that TP-LMS algorithm has better convergence speed than that of P-LMS algorithm but with small increase in complexity. It can be viewed as a trade-off between the convergence speed and complexity.

**7. REFERENCE**