Identification of a Nonlinear System Modeled by a Sparse Volterra Series

Leehter Yao, William A. Sethares and Yu-Hen Hu

Dept. of Electrical and Computer Engr.
University of Wisconsin - Madison
Madison, WI 53706

Abstract: An algorithm based on recursive approximation and estimation is proposed for the identification of nonlinear systems which can be modeled by a sparse Volterra series. The algorithm detects the terms of the Volterra series on which the output depends and estimates the associated Volterra kernels using a least squares criterion. The performance of the algorithm is primarily dependent on the number of nonzero Volterra kernels and not on their distribution in the whole series. The input sequence can be either i.i.d. or correlated. The algorithm can also be directly applied to the delay estimation of a sparse FIR filter.

1. INTRODUCTION

In some applications such as chemical process control, underwater acoustics or geophysical exploration, it is known a priori that the nonlinear system to be modeled involves large time delays. A Volterra series with appropriately chosen order and time delays can be used to model such nonlinear system. Due to the large time delays, the nonzero Volterra kernels are often sparsely distributed. Equivalently, most of the terms in the expansion are zero. We call such a Volterra series a *sparse* series.

The number of Volterra kernels increases geometrically as the delays (or memory) increases. Therefore, the price of using a Volterra series with larger time delays is to immensely increase the number of Volterra kernels to be identified. Several researchers have investigated ways of estimating Volterra kernels [1]-[7]. However, these methods consider each kernel equally important and estimate the complete set of kernels of the Volterra series. Apparently, a great deal of computation effort will be wasted if these methods are applied to identify the kernels of sparse Volterra series since only a mild number of terms in the Volterra series actually contribute to the output. Moreover, the estimated Volterra kernels will lose accuracy since estimation distortion is introduced when estimating a large number of unwanted kernels.

Instead of estimating all of the Volterra kernels, the Recursive Approximation and Estimation (RAE) algorithm is proposed in this paper, which estimates only the kernels associated with the terms contributing most significantly to the output. At each iteration, the RAE algorithm constructs candidate vectors of input signals for each term in the Volterra series. In conjunction with the candidate vectors selected in previous iterations, the candidate vector that best approximates the output is selected. This vector corresponds to the term in Volterra series that the output is most linearly dependent on. Since the approximation is done in the sense of least square error, a matrix inverse is required for each candidate vector at every iteration. Fortunately, the RAE algorithm utilizes an efficient recursive method to relieve this computational burden. It is shown in [11] that the least square of residual error generated in each iteration is a monotonically decreasing sequence.

In [8] and [9], an adaptive delay filter is used to solve the delay estimation of a sparse FIR filter. Since the linear Finite Impulse Response (FIR) filter is a special case of the Volterra series, the RAE algorithm can be applied to the time delay estimation of a sparse FIR filter as well. It will be shown that the RAE algorithm solves the delay estimation for sparse FIR filters efficiently for either independent identically distributed (i.i.d.) or correlated input sequences.

2. VOLTERRA SERIES AND PROBLEM STATEMENT

A truncated p-th order Volterra series expansion is given as:

$$y(k) = h_0 + \sum_{m_1=0}^{N-1} h_1(m_1)u(k-m_1) + \ldots + \sum_{m_p=0}^{N-1} \ldots \sum_{m_1=0}^{N-1} h_p(m_1,..,m_p)u(k-m_1)\ldots u(k-m_p) + n(k), \quad (2.1)$$

where $h_p(m_1,...,m_p)$ is known as the p-th order Volterra kernel of the series; $y(\cdot)$ is the output signal; $u(\cdot)$ is the input signal and $n(\cdot)$ is the measurement noise. The measurement noise is assumed to be an i.i.d. random sequence with zero mean and uncorrelated with the input sequence. Without loss of generality, we assume $h_0 = 0$ in our future analysis. The Volterra kernels in (2.1) are assumed to be symmetric, i.e., $h_p(m_1,...,m_p)$ is unchanged for any of the possible p! permutations of the indices $m_1,...,m_p$. So, the number of different kernels in the 1st, 2nd, ...,p-th order partition of p-th order Volterra are calculated as follows:

$M_1 = N$; $M_2 = N(N+1)/2$;

$$M_p = \sum_{m_1=1}^{N} \sum_{m_2=1}^{m_1} \ldots \sum_{m_{p-2}=1}^{m_{p-2+1}} \frac{m_{p-2}(m_{p-2}+1)}{2}; \text{ for } p \geq 3. \quad (2.2)$$

The total number of different kernels M in (2.1) is

$$M = \sum_{i=1}^{p} M_i . \quad (2.3)$$

The nonlinear discrete time system considered here is assumed to be a time-invariant, finite-memory system which can be modeled by a sparse Volterra series. That is, only a small number (say q) of the Volterra kernels in (2.1) are nonzero. If the identification schemes such as in [1]-[7] are used to estimate these kernels, then all M kernels need to be identified despite the fact that only q of these M kernels are nonzero.

Throughout this paper, a vector is denoted by a bold, lower case letter, and a matrix is denoted by a bold, capitalized letter. Suppose the length of input and output data to be taken for identification is L; where L can be either less or larger than M. Nevertheless, L is assumed to be always larger than q. Then, if the output vector is defined as

92CH3179-9/92/$3.00

$$\mathbf{y} = [y(k), y(k+1),...,y(k+L-1)]^T, \quad (2.4)$$

the truncated Volterra series in (2.1) can be rewritten as a linear vector equation

$$\mathbf{y} = \sum_{i=0}^{M-1} w_i \mathbf{x}_i + \mathbf{n}, \quad (2.5)$$

where w_i denotes the distinctive Volterra kernel; $\mathbf{x}_i$ denotes the vector of associated input signals and $\mathbf{n}$ denotes the vector of measurement noise. The input sequence $u(\cdot)$ can be either an i.i.d. or correlated random sequence such that the vectors $\mathbf{x}_i$, $i = 0..M-1$, are linearly independent on one another and on the noise vector $\mathbf{n}$. Among those M kernels, there are only q nonzero terms. Thus, (2.5) can be rewritten as

$$\mathbf{y} = \sum_{i=1}^{q} w_{d_i} \mathbf{x}_{d_i} + \mathbf{n}, \quad (2.6)$$

where $w_{d_i} \neq 0$ and $d_i \in (0..M-1)$, $\forall\, i = 1..q$.

Based on the input $u(\cdot)$ and output $y(\cdot)$, the identification task is to estimate q, $d_1,...,d_q$ and $w_{d_1},...,w_{d_q}$ instead of the complete set of Volterra kernels $w_0,..,w_{M-1}$.

3. RECURSIVE APPROXIMATION AND ESTIMATION ALGORITHM

Referring to (2.6), we want to search for the vectors $\mathbf{x}_{d_1},...,\mathbf{x}_{d_q}$ among the candidate vectors $\mathbf{x}_0,..,\mathbf{x}_{M-1}$ that $\mathbf{y}$ most linearly depends on. The main idea to achieve this goal is to recursively search for a set of vectors that best approximates $\mathbf{y}$ in the sense of least square error. The algorithm starts by projecting $\mathbf{y}$ onto each of the candidate vectors and choosing the vector associated with the least residual error. This residual error of approximating $\mathbf{y}$ by each of the projected vectors is

$$\mathbf{e}_{1i}^T\mathbf{e}_{1i} = (\mathbf{y} - \mathbf{x}_i(\mathbf{x}_i^T\mathbf{x}_i)^{-1}\mathbf{x}_i^T\mathbf{y})^T(\mathbf{y} - \mathbf{x}_i(\mathbf{x}_i^T\mathbf{x}_i)^{-1}\mathbf{x}_i^T\mathbf{y})$$
$$= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{x}_i(\mathbf{x}_i^T\mathbf{x}_i)^{-1}\mathbf{x}_i^T\mathbf{y}, \ \forall\, i = 0..(M-1). \quad (3.1)$$

Then, the first selected vector is

$$\mathbf{s}_1 = \arg(\min_{\mathbf{x}_i} (\mathbf{e}_{1i}^T\mathbf{e}_{1i})) \quad (3.2)$$

In the subsequent iterations, one vector per iteration is selected from among the remaining candidate vectors so that the least square residual error is obtained in each iteration. When calculating the residual error, the vectors selected in the previous iterations must also be considered. Among the candidate vectors $\mathbf{x}_0,...,\mathbf{x}_{M-1}$, suppose $\mathbf{s}_1,..,\mathbf{s}_{p-1}$ are the vectors selected from iteration 1 to p-1. By constructing a matrix

$$\mathbf{Q}_{pi} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{p-1}, \mathbf{x}_i], \quad (3.3)$$

$\forall\, \mathbf{x}_i \in ((\mathbf{x}_0,...,\mathbf{x}_{M-1}) - (\mathbf{s}_1,...,\mathbf{s}_{p-1}))$; the residual error to be compared is

$$\mathbf{e}_{pi}^T\mathbf{e}_{pi} = (\mathbf{y}-\mathbf{Q}_{pi}(\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1}\mathbf{Q}_{pi}^T\mathbf{y})^T(\mathbf{y}-\mathbf{Q}_{pi}(\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1}\mathbf{Q}_{pi}^T\mathbf{y})$$
$$= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{Q}_{pi}(\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1}\mathbf{Q}_{pi}^T\mathbf{y}, \quad (3.4)$$

$\forall\, \mathbf{x}_i \in ((\mathbf{x}_0,...,\mathbf{x}_{M-1}) - (\mathbf{s}_1,...,\mathbf{s}_{p-1}))$.

The vector to be selected at iteration p will be

$$\mathbf{s}_p = \arg(\min_{\mathbf{x}_i} (\mathbf{e}_{pi}^T\mathbf{e}_{pi})). \quad (3.5)$$

The least square of residual error at iteration p is

$$\mathbf{e}_p^T\mathbf{e}_p = \min_i (\mathbf{e}_{pi}^T\mathbf{e}_{pi})$$
$$= \mathbf{y}^T\mathbf{y} - \mathbf{w}_p^T\mathbf{Q}_p^T\mathbf{y}, \quad (3.6)$$

where $\mathbf{Q}_p = [\mathbf{s}_1,...,\mathbf{s}_p] = [\mathbf{Q}_{p-1}\ \mathbf{s}_p]$; (3.7)

and the associated estimates of kernels

$$\mathbf{w}_p = (\mathbf{Q}_p^T\mathbf{Q}_p)^{-1}\mathbf{Q}_p^T\mathbf{y}. \quad (3.8)$$

Referring to (3.4), the matrix inverse $(\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1}$ is calculated for every candidate vector at each iteration. As the number of detected vectors increases, this calculation will dominate the computation time of the RAE algorithm. To improve the efficiency of the algorithm, $(\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1}\mathbf{Q}_{pi}^T$ is iteratively calculated as follows.

Referring to (3.3), $\mathbf{Q}_{pi} = [\mathbf{Q}_{p-1}, \mathbf{x}_i]$, the matrix inversion lemma [10] shows that

$$(\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1} = \begin{bmatrix} \mathbf{Q}_{p-1}^T\mathbf{Q}_{p-1} & \mathbf{Q}_{p-1}^T\mathbf{x}_i \\ \mathbf{x}_i^T\mathbf{Q}_{p-1} & \mathbf{x}_i^T\mathbf{x}_i \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} (\mathbf{Q}_{p-1}^T\mathbf{Q}_{p-1})^{-1} + \delta_{pi}\mathbf{z}_{pi}\mathbf{z}_{pi}^T & -\delta_{pi}\mathbf{z}_{pi} \\ -\delta_{pi}\mathbf{z}_{pi}^T & \delta_{pi} \end{bmatrix}, \quad (3.9)$$

where $\mathbf{z}_{pi} = (\mathbf{Q}_{p-1}^T\mathbf{Q}_{p-1})^{-1}\mathbf{Q}_{p-1}^T\mathbf{x}_i$; (3.10)

and $\delta_{pi} = (\mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{Q}_{p-1}(\mathbf{Q}_{p-1}^T\mathbf{Q}_{p-1})^{-1}\mathbf{Q}_{p-1}^T\mathbf{x}_i)^{-1}$

$$= (\mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{Q}_{p-1}\mathbf{z}_{pi})^{-1}. \quad (3.11)$$

Let $\mathbf{R}_{pi} = (\mathbf{Q}_{pi}^T\mathbf{Q}_{pi})^{-1}\mathbf{Q}_{pi}^T$. Then, referring to (3.9),

$$\mathbf{R}_{pi} = \begin{bmatrix} (\mathbf{Q}_{p-1}^T\mathbf{Q}_{p-1})^{-1}\mathbf{Q}_{p-1} + \delta_{pi}\mathbf{z}_{pi}(\mathbf{z}_{pi}^T\mathbf{Q}_{p-1}^T - \mathbf{x}_i^T) \\ -\delta_{pi}(\mathbf{z}_{pi}^T\mathbf{Q}_{p-1}^T - \mathbf{x}_i^T) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{R}_{p-1} + \mathbf{z}_{pi}\mathbf{g}_{pi} \\ -\mathbf{g}_{pi} \end{bmatrix}, \quad (3.12)$$

where $\mathbf{g}_{pi} = \delta_{pi}(\mathbf{z}_{pi}^T\mathbf{Q}_{p-1}^T - \mathbf{x}_i^T)$. (3.13)

So, $\mathbf{R}_{pi}$ can be iteratively calculated by using $\mathbf{R}_{p-1}$ which is calculated at the previous iteration. Similarly, $\mathbf{z}_{pi}$ is iteratively calculated by

$$\mathbf{z}_{pi} = \mathbf{R}_{p-1}\mathbf{x}_i. \quad (3.14)$$

Note that $\mathbf{R}_p$ is the matrix $\mathbf{R}_{pi}$ in which $\mathbf{z}_{pi}$ and $\mathbf{g}_{pi}$ are respectively calculated as in (3.10) and (3.13) with $\mathbf{x}_i$ being substituted by $\mathbf{s}_p$ in (3.5).

The associated estimates of the Volterra kernels can also be updated by

$$\mathbf{w}_{pi} = \mathbf{R}_{pi}\mathbf{y}$$

$$= \begin{bmatrix} \mathbf{w}_{p-1} + \mathbf{z}_{pi}\mathbf{g}_{pi}\mathbf{y} \\ -\mathbf{g}_{pi}\mathbf{y} \end{bmatrix}. \quad (3.15)$$

The associated residual error of each candidate vector in (3.4) is thus rewritten as

$$\mathbf{e}_{pi}^T\mathbf{e}_{pi} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{Q}_{pi}\mathbf{w}_{pi}. \quad (3.16)$$

The algorithm is summarized in Figure (1). Note that in steps (b3) - (b5) and (f7) - (f9) of the RAE algorithm, as long as $\mathbf{s}_p$ is selected, $\mathbf{w}_p$ and $\mathbf{e}_p^T\mathbf{e}_p$ are the associated weights and square of residual error which need not be searched again as stated in these steps.

If the measurement noise is zero and $\mathbf{Q}_{p-1}$ contains the vectors $\mathbf{x}_{d_1},...,\mathbf{x}_{d_q}$,

$$\mathbf{g}_{pi}\mathbf{y} = \delta_{pi}(\mathbf{x}_i^T\mathbf{Q}_{p-1}(\mathbf{Q}_{p-1}^T\mathbf{Q}_{p-1})^{-1}\mathbf{Q}_{p-1}^T\mathbf{y} - \mathbf{x}_i^T\mathbf{y})$$
$$= \delta_{pi}(\mathbf{x}_i^T\mathbf{y} - \mathbf{x}_i^T\mathbf{y}) = 0. \quad (3.17)$$

Therefore, from (3.15),

$$w_{pl} = \begin{bmatrix} w_p \\ 0 \end{bmatrix}. \qquad (3.18)$$

This shows that in the noise free case, once the true vectors are detected, the updates of the estimated Volterra kernels also stops.

In [11], the RAE algorithm is shown to be stable in the sense of Lyapunov in the noise free case. In fact, in the noise free case, a convergence analysis is straightforward. Suppose the measurement noise $n(\cdot) = 0$. If $L < M$, then when $p = L$, Q_p is a square matrix, and

$$e_p^T e_p = y^T y - y^T Q_p (Q_p^T Q_p)^{-1} Q_p^T y$$
$$= y^T y - y^T y = 0.$$

If $L > M$, then when $p = M$, all of the candidate vectors have been searched. It follows that $x_{d_1},\ldots,x_{d_q} \in (s_1,\ldots,s_p)$, and $e_p^T e_p = 0$.

For the RAE algorithm, an i.i.d. random input sequence gives the best results since the candidate vectors are well separated. In particular, for the modeling of sparse FIR filters, the candidate vectors are nearly orthogonal, making it easier for the algorithm to detect the true vectors that the output vector y linearly depends on. However, correlated input sequences may be used as long as the candidate vectors are linearly independent of one another. The tradeoff is that the RAE algorithm will take more iterations to detect the true vectors, since more false vectors that are closely correlated with the true vectors tend to be detected during the estimation process.

It is not necessarily true that

$$s_j \in (x_{d_1},\ldots,x_{d_q}), \quad \forall\ j = 1..p.$$

Before the whole set of vectors $x_{d_1},\ldots,x_{d_q}$ are all detected, the algorithm might select some vectors which better approximate y in association with previously selected vectors. The estimation might be misled by these falsely selected vectors. However, since each of the selected vectors is appended into the data matrix Q_p as in (3.7), the effects of false vectors decreases with time through the least square estimation of the associated Volterra kernels. Acceptance of false vectors is more likely to occur in the early iterations, but since the effects of these false vectors decrease in every iteration, the true vectors are ultimately detected.

Since the RAE algorithm is able to come back to the right track of estimation process in a small number of iterations once it is misled by detecting the false vectors, the performance of this algorithm is not affected greatly by increasing the number of candidate vectors.

4. SIMULATION RESULTS

In this section, a simulation example is made with the input sequences being both i.i.d. and correlated random sequences. It can be verified from the simulation that the RAE algorithm converges within a small number of iterations for both types of input sequences. If the number of candidate vectors is greatly increased, it will be shown that the performance of the algorithm is barely affected.

The i.i.d. input sequence is assumed to be the random sequence uniformly distributed between +3 and -3; while the correlated input sequence is assumed to be the output of

$$u(k) = 1.4u(k-1) - 0.48u(k-2) + \omega(k), \qquad (4.1)$$

where $\omega(k)$ is a random sequence uniformly distributed between +3 and -3. The measurement noise $n(\cdot)$ is assumed to be the random sequence uniformly distributed between +0.6 and -0.6 and uncorrelated with the input signal. The data length L is set to be 300 and the error bound ϵ is set to be 0.2.

Example 4.1: Assume that the nonlinear system to be modeled is

$$y(k) = 2u(k-4) + 5u^2(k) - 4u(k-1)u(k-2) - 3u(k-2)u(k-3) + u^2(k-5) - 2.5u(k)u(k-2)u(k-8) + 3.5u(k)u(k-2) u(k-9) + 1.5u(k)u^2(k-3) + 3.25u(k)u^2(k-6) + n(k), \qquad (4.2)$$

where $n(k)$ is the measurement noise as described above. For the simulation, a 3rd order Volterra series with the length of memory $N = 10$ and 15 and a 4th order Volterra series with $N = 10$ are used to model this nonlinear system. Rewriting (4.2) as a linear vector equation

$$y = 2x_4 + 5x_{10} - 4x_{21} - 3x_{30} + x_{50} - 2.5x_{90} + 3.5x_{91} + 1.5x_{92} + 3.25x_{110} + n; \qquad (4.3)$$

for the 3rd and 4th order Volterra series with $N = 10$; and

$$y = 2x_4 + 5x_{15} - 4x_{31} - 3x_{45} + x_{80} - 2.5x_{170} + 3.5x_{171} + 1.5x_{177} + 3.25x_{210} + n; \qquad (4.4)$$

for the 3rd order Volterra series with $N = 15$. The simulation results are summarized as in Table 1.

Take the simulation of the 3rd order Volterra series with $N = 15$ as an example. For the correlated input, the vectors detected (in the order of occurrence) are x_{180}, x_{213}, x_{165}, x_{228}, x_{143}, x_{171}, x_{210}, x_{177}, x_{170}, x_{45}, x_{15}, x_{31}, x_{80} and x_4; the estimates of the associated Volterra kernels are:

$w_{180} = 0.00128$, $w_{213} = 0.00099$, $w_{165} = 0.00053$, $w_{228} = 0.00005$, $w_{143} = -0.00044$, $w_{171} = 3.49959$, $w_{210} = 3.24859$, $w_{177} = 1.49900$, $w_{170} = -2.49970$, $w_{45} = -3.00021$, $w_{15} = 5.00151$, $w_{31} = -4.00147$, $w_{80} = 0.99931$, and $w_4 = 2.01070$.

The false vectors are thus easily identified from the magnitude of these kernels. This is also true for the other two cases in Table 1. The error convergence and the estimation of Volterra kernels of the other two cases are not reiterated here. ΔΔΔ

5. CONCLUDING REMARKS

In this paper, the RAE algorithm is proposed and applied to the identification of a nonlinear system which can be modeled by a sparse Volterra series. It efficiently detects the terms on which the output linearly depends in the sparse Volterra series. In fact, the nonlinear system that the RAE algorithm can be applied are not restricted to the ones modeled by sparse Volterra series. It can be applied to any nonlinear system which can be modeled by a Volterra series (not necessarily being sparse). With the order and number of delays chosen, the RAE algorithm is able to select the terms that contribute most significantly to the output. As a result, the number of terms required in the Volterra series can be minimized. Unlike some of the currently developed methods, the input sequence for the RAE algorithm does not have to be an i.i.d. sequence.

References

[1] W. J. Rugh, *Nonlinear System Theory, The Volterra Wiener Approach*, Johns Hopkins University Press, Baltimore, Maryland, 1981.

[2] M. Schetzen, The Volterra and Wiener Theory of the Nonlinear Systems, Wiley and Sons, New York, 1980.

[3] S. Boyd, Y. S. Tang and L. O. Chua, "Measuring Volterra kernels," *IEEE Trans. Circuits and Systems*, Vol. CAS-30, No. 8, pp. 571-577, Aug. 1983.

[4] M. J. Korenberg, "Identifying nonlinear difference equation and functional expansion representations: the fast orthogonal algorithm," *Annals of Biomedical Engineering*, Vol. 16, pp. 123-142, 1988.

[5] M. J. Korenberg, S. B. Bruder and P. J. McIlroy, "Exact orthogonal kernel estimation from finite data records: extending Wiener's identification of nonlinear systems,"

Annals of Biomedical Engineering, Vol. 16, pp. 201-214, 1988.

[6] T. Koh, and E. J. Powers, "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Trans. Acoust., Speech and Signal Processing*, Vol. ASSP-33, No.6, pp. 1445-1455, Dec. 1985.

[7] V. J. Mathews and J. Lee, "A fast recursive least-square second-order Volterra filter," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, New York, pp. 1383-1386, April 1988.

[8] D. M. Etter and Y. F. Cheng, "System modeling using an adaptive delay filter," *IEEE Trans. Circuits Syst.*, Vol. CAS-34, pp. 770-774, July 1987.

[9] Y. F. Cheng and D. M. Etter, "Analysis of an adaptive technique for modeling sparse systems," *IEEE Trans. Acoust., Speech and Signal Processing*, Vol. ASSP-37, pp. 254-264, Feb. 1989.

[10] T. Kailath, *Linear Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1980.

[11] L. Yao, W. A. Sethares and Y. H. Hu, "Identification of a Nonlinear System Modeled by a Sparse Volterra Series," Submitted to *IEEE. Trans. Signal Processing*.

Recursive Approximation and Estimation (RAE) algorithm

(a) Set $p = 1$, ε = tolerance of residual error.

(b) If $p = 1$, then $\forall\ \mathbf{x}_l \in (\mathbf{x}_0,\ldots,\mathbf{x}_{M-1})$, calculate

(b1) $w_{11} = \mathbf{x}_l^T\mathbf{y}/\mathbf{x}_l^T\mathbf{x}_l$.

(b2) $\mathbf{e}_{1l}^T\mathbf{e}_{1l} = \mathbf{y}^T\mathbf{y} - w_{11}\mathbf{x}_l^T\mathbf{y}$.

(b3) $\mathbf{s}_1 = \arg(\min_{\mathbf{x}_l}(\mathbf{e}_{1l}^T\mathbf{e}_{1l}))$.

(b4) $w_1 = \arg(\min_{w_{1l}}(\mathbf{e}_{1l}^T\mathbf{e}_{1l}))$.

(b5) $\mathbf{e}_1^T\mathbf{e}_1 = \min_l(\mathbf{e}_{1l}^T\mathbf{e}_{1l})$.

(c) $\mathbf{Q}_1 = [\mathbf{s}_1]$.

(d) $\mathbf{R}_1 = \mathbf{Q}_1^T/\mathbf{Q}_1^T\mathbf{Q}_1$.

(e) $p = p+1$.

(f) $\forall\ \mathbf{x}_l \in ((\mathbf{x}_0,\ldots,\mathbf{x}_{M-1}) - (\mathbf{s}_1,\ldots,\mathbf{s}_{p-1}))$, calculate

(f1) $\mathbf{z}_{pl} = \mathbf{R}_{p-1}\mathbf{x}_l$.

(f2) $\delta_{pl} = (\mathbf{x}_l^T\mathbf{x}_l - \mathbf{x}_l^T\mathbf{Q}_{p-1}\mathbf{z}_{pl})^{-1}$.

(f3) $\mathbf{g}_{pl} = \delta_{pl}(\mathbf{z}_{pl}^T\mathbf{Q}_{p-1}^T - \mathbf{x}_l^T)$.

(f4) $\mathbf{w}_{pl} = \begin{bmatrix} \mathbf{w}_{p-1} + \mathbf{z}_{pl}\mathbf{g}_{pl}\mathbf{y} \\ -\mathbf{g}_{pl}\mathbf{y} \end{bmatrix}$.

(f5) $\mathbf{Q}_{pl} = [\mathbf{Q}_{p-1}, \mathbf{x}_l]$.

(f6) $\mathbf{e}_{pl}^T\mathbf{e}_{pl} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{Q}_{pl}\mathbf{w}_{pl}$.

(f7) $\mathbf{s}_p = \arg(\min_{\mathbf{x}_l}(\mathbf{e}_{pl}^T\mathbf{e}_{pl}))$.

(f8) $\mathbf{w}_p = \arg(\min_{\mathbf{w}_{pl}}(\mathbf{e}_{pl}^T\mathbf{e}_{pl}))$.

(f9) $\mathbf{e}_p^T\mathbf{e}_p = \min_l(\mathbf{e}_{pl}^T\mathbf{e}_{pl})$.

(g) $\mathbf{Q}_p = [\mathbf{Q}_{p-1}\ \mathbf{s}_p]$.

(h) Substituting $\mathbf{x}_l$ with $\mathbf{s}_p$ in (f1) - (f3) to calculate $\mathbf{z}_p$, δ_p and $\mathbf{g}_p$, then update $\mathbf{R}_p$ by $\mathbf{R}_p = \begin{bmatrix} \mathbf{R}_{p-1} + \mathbf{z}_p\mathbf{g}_p \\ -\mathbf{g}_p \end{bmatrix}$.

(i) If $\mathbf{e}_p^T\mathbf{e}_p/L < \varepsilon$ or $p = L$, stop; otherwise go back to step (e) and continue.

Figure 1. The RAE Algorithm

	order = 3, N = 10	order = 3, N = 15	order = 4, N = 10
# of candidate vectors M	285	815	1000
# of iterations required to converge for i.i.d. input	9	9	9
# of iterations required to converge for correlated input	13	14	12

Table 1. Comparison of simulation results for the 3rd order Volterra series with N = 10 and 15 respectively and the 4th order Volterra series with N = 10.