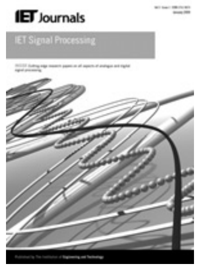
Published in IET Signal Processing
Received on 17th November 2007
Revised on 23rd August 2008
doi: 10.1049/iet-spr:20070203



ISSN 1751-9675

Identification of nonlinear systems by the genetic programming-based Volterra filter

L. Yao   C.-C. Lin

Deptartment of Electrical Engineering, National Taipei University of Technology, Taipei 10642, Taiwan, Republic of China
E-mail: ltyao@ntut.edu.tw

Abstract: The genetic programming (GP) algorithm is utilised to search for the optimal Volterra filter structure. A Volterra filter with high order and large memories contains a large number of cross-product terms. Instead of applying the GP algorithm to search for all cross-products of input signals, it is utilised to search for a smaller set of primary signals that evolve into the whole set of cross-products. With GP's optimisation, the important primary signals and the associated cross-products of input signals contributing most to the outputs are chosen whereas the primary signals and the associated cross-products of input signals that are trivial to the outputs are excluded from the possible candidate primary signals. To improve GP's learning capability, an effective directed initialisation scheme, a tree pruning and reorganisation approach, and a new operator called tree extinction and regeneration are proposed. Several experiments are made to justify the effectiveness and efficiency of the proposed modified by the GP algorithm.

1 Introduction

It is common to have signals generated from a nonlinear source or processed by nonlinear systems in real world signal-processing applications. Filtering and identification of nonlinear signals and systems are widely encountered in practical applications. Linear models may not adequately approximate or describe the behaviours of nonlinear systems. One of the popular approaches to modelling nonlinear systems is the Volterra filter [1]. Finite support Volterra filters feature universal approximation and linear regression capability [2]. The applications of Volterra filters include echo cancellation [3, 4], channel equalisation [5, 6], system identification [7, 8], detection and estimation [9, 10] and physiological system modelling [11, 12]. Various Volterra estimation approaches have been proposed, based mainly on the mean squared error (MSE) or least squares error (LSE) formulations. With MSE or LSE formulations, linear adaptive filtering approaches [13, 14] are applied to the estimation of Volterra kernels without too many modifications. The least mean square (LMS) and recursive least squares (RLS) approaches were, respectively, proposed in [15–17] and [18–20] to learn the Volterra kernels based on input and output signals. One of the schemes for reducing computational efforts for the RLS approach is to design the linear regression using a multi-channel set up such as in [19, 20] and [21–23], in which the Volterra filter is realised using a linear filter bank. Each filter in the filter bank processes different cross-products of input signals and performs linear regression in parallel so that the computational efficiency can be greatly improved. The inputs for the multi-channel linear model are delays of a basic set of signals called primary signals [21]. Each channel contains different memory lengths of delayed primary signals. In [24], a novel algebraic structure called $\boldsymbol{v}$-vector based on non-rectangular matrices was proposed for the development of Volterra-adaptive filter algorithms. With $\boldsymbol{v}$-vector, the multi-channel linear adaptive approaches to estimating Volterra kernels are treated in the same way as classic multi-channel linear filter.

Identifying nonlinear systems using Volterra filters, signifies the important features of the Volterra filter, which employs combinations of polynomial basis functions for system modelling. With predetermined order and memory lengths, all possible combinations of cross-products of delayed input signals are utilised to model the nonlinear system. It results in a large number of Volterra kernel coefficients associated with the cross-products of input signals. Although the estimation of Volterra kernel coefficients can be solved using multi-channel linear adaptive approaches, the number of channels still remains a large quantity, especially for the Volterra filter with large order and/or long memory lengths. For some practical applications, the large Volterra filter structure leads to hardware or firmware complications. Moreover, most of the multi-channel Volterra kernel estimation methods consider each kernel coefficient equally important and estimate the complete set of kernel coefficient of the Volterra filter. All of the kernel coefficients are identified despite the fact that only a small number may contribute significantly to the output signal. Apparently, a great deal of computation effort is wasted as only a small number of the terms in the Volterra filter actually contribute to the output. The estimated Volterra kernel coefficients will also lose accuracy because estimation distortion is inevitably introduced when estimating a large number of insignificant kernel coefficients. In this paper, the genetic programming (GP) algorithm [25, 26] is proposed to search among all candidate primary signals for the most suitable set of primary signal combinations. With GP's optimisation capability, the most significant primary signals and their associated cross-products of input signals that contribute most to the outputs will be chosen. As the optimised Volterra filter structure obtained using the GP algorithm results in fewer primary signals than the regular Volterra filter, the number of channels in the multi-channel linear filtering approach for kernel coefficient estimation can be greatly reduced. With this much smaller filter structure, the estimations of Volterra kernel coefficients are more accurate. The optimised Volterra filter is more efficient in terms of implementation and performance.

Evolutionary algorithms have also been applied to find the minimal Volterra filter structure. In [27, 28], the genetic algorithm (GA) with binary coded chromosomes was applied to find the minimal structure of the sparse Volterra filter. The approximate size of the optimised Volterra filter structure needs to be predetermined in the approach proposed in [27, 28] as the predetermined approximate filter size is taken as the chromosomes length in the GA. This constraint sometimes limits the prevalence of the approach proposed in [27, 28] as for some applications, the approximate size of the Volterra filter's minimal structure is unknown *a priori*. The GP algorithm proposed in this paper can overcome this difficulty as the GP algorithm with modifications proposed in this paper tends to find the minimal size of Volterra filter automatically without assuming approximate size of the final optimised structure. Similar scheme to the one proposed in this paper was adopted in [29] where the GP algorithm was also utilised to search for the best combinations of input signal cross-products. However, in [29], the GP algorithm was directly utilised to search all possible candidates of cross-products of input signals rather than the primary signals in this paper. The computational efficiency of searching for the primary signals instead of cross-products of input signals is numerically analysed in this paper. It will be shown that the number of primary signals to be searched is significantly less than the number of cross-products of input signals. One of the contributions of this paper is the idea of searching for primary signals rather than all possible candidates of cross-products of input signals. To improve GP's learning capability, an effective tree pruning approach, a directed initialisation scheme, and a new operator called tree extinction and regeneration are proposed in this paper. In this paper, GP's search results are the best combinations of primary signals. Each primary signal and its associated delay terms are inputs of the Volterra filter's multi-channel linear model. The multi-channel learning scheme in [19] for the second-order Volterra filter was slightly modified to learn the Volterra filter of all possible orders in this paper. With primary signals taken as the searching candidates, the computationally efficient multi-channel learning schemes can be directly utilised to learn the Volterra kernel coefficients.

2 Primary signals of the Volterra model

The single input and single output nonlinear system to be discussed is expressed as

$$y(n) = f(x(n), x(n-1), \dots, x(n-d_{\max})) \tag{1}$$

where $y(\cdot)$ and $x(\cdot)$ denote the outputs and inputs of the system, respectively. Assume that the maximum 'memory' of the system in (1) is $d_{\max}$. The relationship between the delayed inputs and system output is represented by the nonlinear function $f(\cdot)$. The nonlinear system in (1) is assumed to be modelled by a pth order finite support Volterra filter

$$y_v(n) = h_0 + \sum_{k=1}^{p}\sum_{i_1=0}^{d_k-1}\sum_{i_2=0}^{d_k-1}\cdots\sum_{i_k=0}^{d_k-1} h_k(i_1, i_2, \dots, i_k)\prod_{j=1}^{k} x(n-i_j) \tag{2}$$

where $y_v(\cdot)$ is the output of Volterra filter, $h_k(i_1, i_2, \dots, i_k)$ is the kth order Volterra kernel, $k = 1, \dots, p$; $d_i, i = 1, \dots, p$, are the memories of the corresponding nonlinearity and $d_i \le d_{\max}$. The Volterra model in (2) is assumed to be symmetric, that is $h_k(i_1, i_2, \dots, i_k)$ is unchanged for any of $k!$ permutations of the indices $i_1, i_2, \dots, i_k$. Some parameterisation redundancies exist in (2) as $k!$ permutations of the sequence of multiplications $x(n-i_1)x(n-i_2), \dots, x(n-i_k)$ are considered the same. The Volterra model in (2) can be simplified to the following triangular form

$$y_v(n) = z_0 + \sum_{k=1}^{p}\sum_{i_1=0}^{d_k-1}\sum_{i_2=i_1}^{d_k-1}\cdots \times \sum_{i_k=i_{k-1}}^{d_k-1} z_k(i_1, i_2, \dots, i_k)\prod_{j=1}^{k} x(n-i_j) \tag{3}$$

where $z_k(i_1, i_2, \ldots, i_k)$ is the kth order triangular Volterra kernel. With the inherent redundancy in (2) being removed, the number of Volterra kernel coefficients in (3) is much less than the one in (2). As in [29], the triangular form in (3) can also be reorganised as a new form called regular form, which is more suitable for the computation and the following analysis. The regular form is intrinsically the same as the triangular form except that the ordering of each item in the Volterra filter is rearranged as follows

$$y_v(n) = r_0 + \sum_{k=1}^{p}\sum_{i_1=0}^{d_k-1}\sum_{i_2=0}^{d_k-1-i_1}\cdots\sum_{i_k=0}^{d_k-1-\sum_{j=1}^{k-1} i_j} \times r_k(i_1, i_2, \ldots, i_k)\prod_{j=1}^{k} x(n - \sum_{q=j}^{k} i_q) \quad (4)$$

where $r_k(i_1, i_2, \ldots, i_k)$ is the kth order regular kernel. The triangular and regular forms of the Volterra filter are permutation equivalent. The number of coefficients in the regular form of Volterra filter in (4) is

$$N_r = 1 + \sum_{i=1}^{p}\binom{d_i + i - 1}{i} \quad (5)$$

If the memories of every order are all the same, that is, $d_i = d$, $\forall i = 2, \ldots, p$, N_r is simplified as

$$N_r = \binom{d+p}{p} \quad (6)$$

The regular form in (4) can be rewritten as a vector form. Define the regressor vector $\boldsymbol{\Phi}_r \in R^{N_r \times 1}$ as

$$\boldsymbol{\Phi}_r(\boldsymbol{n}) = \left[1, \boldsymbol{\theta}_1^T(\boldsymbol{n}), \boldsymbol{\theta}_2^T(\boldsymbol{n}), \ldots, \boldsymbol{\theta}_p^T(\boldsymbol{n})\right]^T \quad (7)$$

where the kth vector component $\boldsymbol{\theta}_k(n) \in R^{\binom{d_k+k-1}{k}\times 1}$ is defined as

$$\begin{aligned}\boldsymbol{\theta}_k(n) = [&x^k(n), x^{k-1}(n)x(n-1), \ldots, x(n - i_1 - i_2 \cdots - i_k) \\ &\times x(n - i_2 - i_3 \cdots - i_k), \ldots, x(n - i_k), \ldots, \\ &\times x(n - d_k + 1)x^{k-1}(n)]^T\end{aligned} \quad (8)$$

$i_1 = 0, \ldots (d_1 - 1)$, $i_2 = 0, \ldots (d_2 - 1 - i_1), \ldots, i_k = 0, \ldots (d_k - 1 - \sum_{q=1}^{k-1} i_q)$.

If $\boldsymbol{W}$ denotes the vector containing the regular kernels, the vector form of (4) is written as

$$y_v(n) = \boldsymbol{W}^T\boldsymbol{\Phi}_r(n) \quad (9)$$

The regular form of the Volterra filter in (4) can also be reorganised as a form that is based on multi-channel linear regression. The multi-channel linear regression form is based on a set of primary signals that carry all the information needed for the estimation of convolution in (4). The kth order primary signal associated with the indices i_1, $i_2, \ldots, i_{k-1}$ is defined as

$$\begin{aligned}\varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n) = x(n)x(n - i_{k-1})x(n - i_{k-1} - i_{k-2}), \ldots, \\ x(n - i_{k-1} - i_{k-2} \cdots - i_1)\end{aligned} \quad (10)$$

$i_1 = 0, \ldots, (d_k - 1), i_2 = 0, \ldots, (d_k - 1 - i_1), \ldots, i_{k-1} = 0, \ldots, (d_k - 1 - \sum_{q=1}^{k-2} i_q), k = 2, \ldots, p$. Note that $i_0 = 0$. The number of primary signals in the form of (10) is calculated as

$$N_p = 2 + \sum_{i=2}^{p}\binom{d_i + i - 2}{i - 1} \quad (11)$$

The regressor of multi-channel linear regression form is given as the matrix containing all the signals evolved from the primary signals with time delays as defined in (10). As the regression signals evolve from the primary signals with time delays, the delays need to be constrained within the range between 0 and $(d_k - 1)$. For a primary signal $\varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n)$, the maximum delay range is defined as

$$\lambda_{i_1,i_2,\ldots,i_{k-1}} = d_k - 1 - \max(i_1, i_2, \ldots, i_{k-1}) \quad (12)$$

The subvector that carries the signals evolved from the primary signal $\varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n)$ with delays within the maximum range is defined as

$$\begin{aligned}\boldsymbol{\rho}^k_{i_1,i_2,\ldots,i_{k-1}}(n) = [&\varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n), \varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n-1), \ldots, \\ &\varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n - \lambda_{i_1,i_2,\ldots,i_{k-1}})]^T \quad \forall k = 1, \ldots, p\end{aligned} \quad (13)$$

Denote $\boldsymbol{\sigma}_k(n) \in R^{\binom{d_k+k-2}{k-1}\times 1}$ as the matrix containing all the subvectors evolved from all the kth order primary signals with different combinations of indices, that is

$$\begin{aligned}\boldsymbol{\sigma}_k(n) = [&\boldsymbol{\rho}^{kT}_{0,0,\ldots 0}(\boldsymbol{n}), \boldsymbol{\rho}^{kT}_{0,0,\ldots 1}(\boldsymbol{n}), \ldots, \boldsymbol{\rho}^{kT}_{i_1,i_2,\ldots i_{k-1}}(\boldsymbol{n}), \ldots, \\ &\boldsymbol{\rho}^{kT}_{d_k-1,0,\ldots 0}(\boldsymbol{n})]^T \quad \forall k = 1 \ldots p\end{aligned} \quad (14)$$

where $i_1 = 0, \ldots, (d_k - 1), i_2 = 0, \ldots, (d_k - 1 - i_1), \ldots, i_{k-1} = 0, \ldots, (d_k - 1 - \sum_{q=1}^{k-2} i_q)$. The regressor for multi-channel linear regression form is expressed in terms of $\boldsymbol{\sigma}_k(n)$ as

$$\boldsymbol{\Phi}(n) = [1, \boldsymbol{\sigma}_1^T(\boldsymbol{n}), \boldsymbol{\sigma}_2^T(\boldsymbol{n}), \ldots, \boldsymbol{\sigma}_p^T(\boldsymbol{n})]^T \quad (15)$$

The regressors $\boldsymbol{\Phi}$ in (15) and $\boldsymbol{\Phi}_r$ in (7) are permutation equivalent. If $\boldsymbol{C}$ denotes the vector containing the kernels of the multi-channel linear regression model, the vector form of the multi-channel linear regression model can be written as:

$$y_v(n) = \boldsymbol{C}^T\boldsymbol{\Phi}(n) \quad (16)$$

The number of signals in $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}_r$ both equal to N_r are as calculated in (5). It is straight forward if all the kernel coefficients in either regular form in (9) or multi-channel linear regression form in (16) are estimated based on the input and output signals.

3 Learning of primary signals

3.1 GP and fitness measure

The identification of nonlinear systems via Volterra filter aims to search for the best set of cross-products of input signals and to estimate the corresponding kernel coefficients. In this paper, a modified GP algorithm is utilised as the search approach. The GP algorithm is a global optimisation scheme based on the mechanism of natural selection and offspring generation. It starts with a population of randomly generated individual trees. Every tree corresponds to the linear combination of primary signals. Every generated tree is evaluated for fitness. The fitness value of every tree is utilised as the measure for selection to generate offspring trees. As the number of primary signals needed to construct the Volterra filter is unknown *a priori*, the GP algorithm is especially suitable to evolve the structure of primary signals. To improve GP's searching efficiency, the candidates for the GP algorithm to search are N_p primary signals rather than N_r terms of cross-products of input signals. Referring to (5) and (11), the difference between N_r and N_p is calculated as

$$\begin{aligned} N_r - N_p &= d_1 - 1 + \sum_{i=2}^{p}\left(\binom{d_i+i-1}{i} - \binom{d_i+i-2}{i-1}\right) \\ &= d_1 - 1 + \sum_{i=2}^{p}\binom{d_i+i-2}{i} \\ &= \sum_{i=1}^{p}\binom{d_i+i-2}{i} \end{aligned} \tag{17}$$

If $d_i = d$, $\forall i = 1, \ldots, p$, then

$$N_r - N_p = \binom{d-1+p}{p} - 1 \tag{18}$$

For the Volterra filter with high order and large time delays, the proposed approach utilising the GP algorithm to search for primary signals is computationally efficient. For instance, a seventh-order Volterra filter with $d_i = 25$, $\forall i = 1, \ldots, 7$, $(N_r - N_p) = 2.6296 \times 10^6$ according to (18). To further investigate the values of N_r and N_p, $N_r = 3.3659 \times 10^6$ and $N_p = 7.3628 \times 10^5$. Apparently, it saves a lot of computational efforts by searching N_p primary signals instead of N_r terms of cross-products.

The Volterra filter can be modelled using GP's tree structure that consists of operators and operands. Let V be the set of primary signals that the GP algorithm could possibly search at the time instant n. Let T be the set of operators associated with V. Referring to (10) and (11), the primary signals are collections of $\varphi^k_{i_1,i_2,\ldots,i_{k-1}}(n)$, $i_1 = 0, \ldots, (d_k - 1)$, $i_2 = 0, \ldots, (d_k - 1 - i_1), \ldots, i_{k-1} = 0, \ldots, \left(d_k - 1 - \sum_{q=1}^{k-2} i_q\right)$, $k = 2, \ldots, p$. For the convenience of representation, the primary signal indices are renumbered from 1 to N_p so that all of the terminal nodes in the GP's tree structure are numbered from 1 to N_p. Let $\varphi_j(n)$ be the re-indexed primary signal at time instant $n, j = 1, \ldots, N_p$, then

$$V = \{\varphi_1(n), \varphi_2(n), \ldots, \varphi_{N_p}(n)\} \tag{19}$$

As for the set T, because the Volterra filter is a linear combination of input signal cross-products generated from the primary signals, T is a set of operators '+', that is

$$\begin{aligned} T = \{&(a_1, a_2, \ldots, a_{N_p-1}) | a_i = \text{'}+\text{'}, \\ &i = 1, \ldots, (N_p - 1)\} \end{aligned} \tag{20}$$

The set of all possible Volterra filters decoded from GP's tree structures is defined as

$$\Gamma = \{(v, t) | v \subset V, t \subset T\} \tag{21}$$

For the qth tree in the pth generation of GP, assume that it contains L_{pq} primary signals. Let $\varphi^{pq}_m(n)$ be the mth primary signal in this tree. According to (13), the vector evolved from $\varphi^{pq}_m(n)$ is defined as

$$\boldsymbol{\rho}^{pq}_m(n) = [\varphi^{pq}_m(n), \varphi^{pq}_m(n-1), \ldots, \varphi^{pq}_m(n - \lambda^{pq}_m)] \tag{22}$$

where λ^{pq}_m is the delay defined in (12). The regressor associated with the searched primary signals is thus defined as

$$\hat{\boldsymbol{\Phi}}_{pq}(n) = [\boldsymbol{\rho}^{pq}_1(n), \boldsymbol{\rho}^{pq}_2(n), \ldots, \boldsymbol{\rho}^{pq}_{L_{pq}}(n)] \tag{23}$$

Note that the regressor $\hat{\boldsymbol{\Phi}}_{pq}(\cdot)$ is based on L_{pq} primary signals contained in GP's tree structure. The size of $\hat{\boldsymbol{\Phi}}_{pq}(\cdot)$ is calculated as

$$N_{pq} = \sum_{m=1}^{L_{pq}} (\lambda^{pq}_m + 1) \tag{24}$$

Recall that the size of regressor $\boldsymbol{\Phi}(\cdot)$ in (15) for the complete set of input signals is N_r in (5). It is obvious that N_{pq} is much less than N_r. The estimation of Volterra kernels based on input and output signals can thus be greatly simplified.

Let $\hat{\boldsymbol{C}}_{pq}(\cdot)$ be the vector of kernel coefficients associated with $\hat{\boldsymbol{\Phi}}_{pq}(\cdot)$, a fast RLS algorithm is employed along with the GP algorithm to estimate $\hat{\boldsymbol{C}}_{pq}(\cdot)$. Let $\gamma \in (0, 1]$ be an exponential window forgetting factor, the fast RLS algorithm is applied to recursively minimise the following dynamic estimation error based on L data points.

$$e_{pq}(n) = \sum_{k=0}^{n} \gamma^{n-k}(y(n) - \hat{\boldsymbol{C}}_{pq}^{T}(n)\hat{\boldsymbol{\Phi}}_{pq}(n))^2, \quad n = 1, \ldots, L \tag{25}$$

Based on the estimation error in (27), a fast transversal adaptive algorithm such as in [19] for the coefficient estimation of the Kalman gain vector in the RLS algorithm is applied. Let $\hat{\boldsymbol{C}}_{pq}$ be the convergent kernel coefficient estimated using the fast transversal adaptive algorithm in [19], the fitness value for the qth tree in the pth generation is defined as the (MSE) as follows

$$E_{pq} = \sqrt{\frac{\sum_{n=1}^{L} (y(n) - \hat{\boldsymbol{C}}_{pq}^{T}(n)\hat{\boldsymbol{\Phi}}_{pq}(n))^2}{L}} \tag{26}$$

where L is the number of data points used for evaluations. If G trees are generated in every generation, the Volterra filter learned in the pth generation is contained in the terminals of the best tree associated with the least fitness value E_p^* defined as

$$E_p^* = \min_{q=1,\ldots,G}(E_{pq}) \tag{27}$$

3.2 Initialisation and crossover

If the GP algorithm is applied to the combinatorial optimisation, as the one in this paper, the GP algorithm has important characteristics that prevent any of the terminal nodes from vanishing in the gene pool. A new node will be generated only through mutation in the evolved gene pool. Therefore if the primary signals are not generated in the gene pool by initialisation, they cannot be regenerated in the evolved gene pool except when they have the chance to replace some terminals through mutation and the tree with the mutated terminals happens to survive in the following generations. To assure full varieties in the gene pool brought by initialisation, a directed initialisation scheme that allows every primary signal to appear in the gene pool is proposed. As opposed to the conventional GP algorithm, which initialises every tree in the gene pool by generating trees with randomly chosen structures and randomly assigned nodes, every initialised tree in the proposed directed initialisation scheme is assumed to contain g terminal nodes in r layers, where $g = 2^r$. The initialised tree with $r = 3$ is illustrated in Fig. 1. Let G denote the number of trees contained in the gene pool. Then, if G is designed as

$$G \geq \mathrm{Int}(N_p/g) + 1 \tag{28}$$

where the Int(·) denotes the operator taking the integer part of the operand, the gene pool contains the trees with enough number of terminals indexing all possible primary signals after initialisation. Let the terminals in the trees obtained by the directed initialisation scheme be indexed in sequence, that is, from 1 to $(g \times G)$. Then, for the jth terminal, $j \in [1, N_p]$, it is randomly assigned a number from 1 to N_p but it is different from the previous $(j-1)$ terminals to avoid duplicated primary signals. For the kth terminal that exceeds the range of N_p, that is, $k \in [N_p + 1, g \times G]$, a number from the interval $[1, N_p]$ is assigned randomly. With this directed initialisation scheme, every possible index of the primary signals are contained in the terminals of the trees in the gene pool.

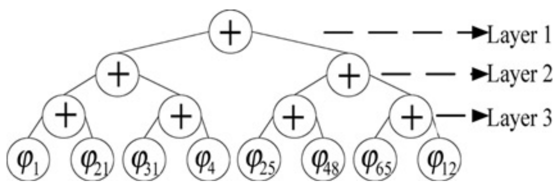


Figure 1 *Illustration of initialised tree with $r = 3$*

The elitist scheme is utilised in this paper for parents' selection, that is, only the parental trees with fitness values ranked at the top among the population are allowed to survive into the next generation and generate offspring. Given any two parental trees S_{p1}, $S_{p2} \in \Gamma$, where $S_{p1} = (v_{p1}, t_{p1})$ and $S_{p2} = (v_{p2}, t_{p2})$, the offspring trees are generated from S_{p1} and S_{p2} through crossover operation. Referring to Fig. 2, the crossover operation is a process that randomly chooses a slice point from both S_{p1} and S_{p2} so that each parental tree is divided into two parts, that is, $S_{p1} = S_{p1}^c + \bar{S}_{p1}^c$ and $S_{p2} = S_{p2}^c + \bar{S}_{p2}^c$. The two sliced parts of each parental tree are then mixed so that the children trees are generated as

$$S_{d1} = S_{p1}^c \cup \bar{S}_{p2}^c \tag{29}$$

$$S_{d2} = \bar{S}_{p1}^c \cup S_{p2}^c \tag{30}$$

Note that the union $\cup$ is utilised in the crossover operation as it might be possible that one sliced part of a parental tree contains some primary signals identical to the ones in the compliment part of the other parental tree. The union of two parental trees consists of two operations, pruning the duplicated primary signals and node merging. Referring to Fig. 3, if any of the branches generated by crossover consists of duplicated primary signals, the duplicated signals in the compliment part of the tree, such as $\bar{S}_{p1}^c$ and $\bar{S}_{p2}^c$, are pruned. After pruning the duplicated nodes, the entire tree is then reorganised based on the following two tree reorganisation rules:

Rule 1: For any node $\wp$, if it is a terminal node and $\wp \in T$, $\wp$ is pruned.

Rule 2: For any node $\wp$, if it is not a terminal node but $\wp \in T$ and $\wp$ has only one branch, $\wp$ is pruned and $\wp$'s associated upper and lower branches are merged.

By both reorganisation rules, the tree with duplicated nodes being pruned can be further simplified.

3.3 Tree extinction and regeneration

Applying the GP algorithm to select a suitable set of primary signals and consequently a suitable set of cross-products in Volterra filter can be considered as combinatorial optimisation. As opposed to the conventional combinatorial optimisation problem, the Volterra filter features the fact that the candidate primary signals for the GP algorithm to search are not independent from one another. As the selected primary signals are close to the optimal solution, the set of selected primary signals might contain most of the solutions and yet still lacks a few target primary signals. As the primary signals are not independent, as long as the target primary signals are not selected, there might be some selected primary signals dependent upon the unfound target primary signals and yet not as significant as the target primary signals. It increases the size of the trees in the GP algorithm and stagnates the learning process. The convergence stagnation also reduces the gene pool variety although the number of tree's terminal nodes increases. In this paper, a scheme called 'tree extinction and regeneration' is proposed to rejuvenate the gene pool and increase the variety of the gene pool, consequently, to help the GP algorithm get around the trap at local optima. If GP's learning process remains no improvement for a certain number of generations, the proposed 'tree extinction and regeneration' operator is triggered. This operator first makes all but the best tree 'extinct', that is, removes all but the best tree from the gene pool. The extinct trees are replaced by the trees generated by the directed initialisation scheme. A tree regeneration scheme that resembles the crossover operator is proposed to 'regenerate' the trees based on the initialised trees and the best tree. One of the parental trees for tree regeneration is designed as the one with the insignificant nodes in the best tree being pruned.

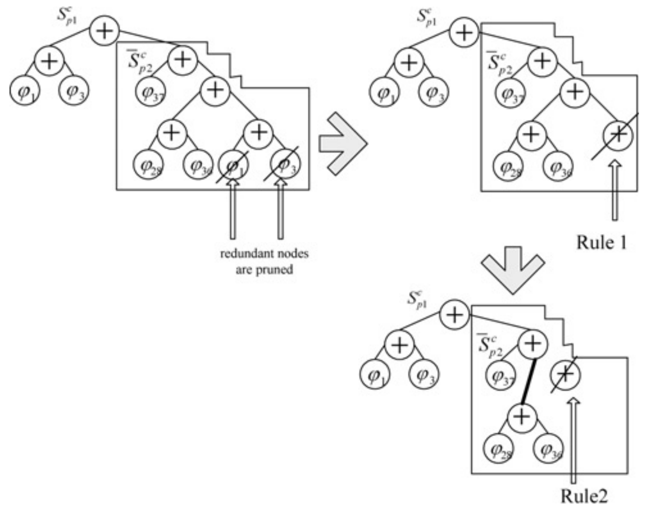


Figure 3 *Illustration of tree reorganisation rules*

The other parental tree for tree regeneration is designed as the initialised tree. The tree generated by the tree regeneration scheme then replaces the original initialised tree. However, it is unknown *a priori* that to what extent the insignificant nodes need to be pruned from the best tree. Let Γ_b be the set containing the trees that originate from the best tree but being pruned according to different thresholds. Assume that the tree extinction and regeneration operator get activated in the βth generation. As the trees in the gene pool are sorted based on the fitness values, let $\hat{\boldsymbol{C}}_\beta$ be the convergence of kernel coefficient associated with the best tree in this generation. Denote $\hat{c}^i_\beta$ as the ith element in the vector $\hat{\boldsymbol{C}}_\beta$ and χ_β as the maximum absolute value of all the elements in $\hat{\boldsymbol{C}}_\beta$, that is

$$\chi_\beta = \max_i (|\hat{c}^i_\beta|) \equiv |\hat{\boldsymbol{C}}_\beta|_\infty \tag{31}$$

Assume that the parental trees in Γ_b are obtained by pruning the best tree according to the v different thresholds $\eta_1, \eta_2, \ldots, \eta_v$. The Volterral kernels satisfying the following conditions are

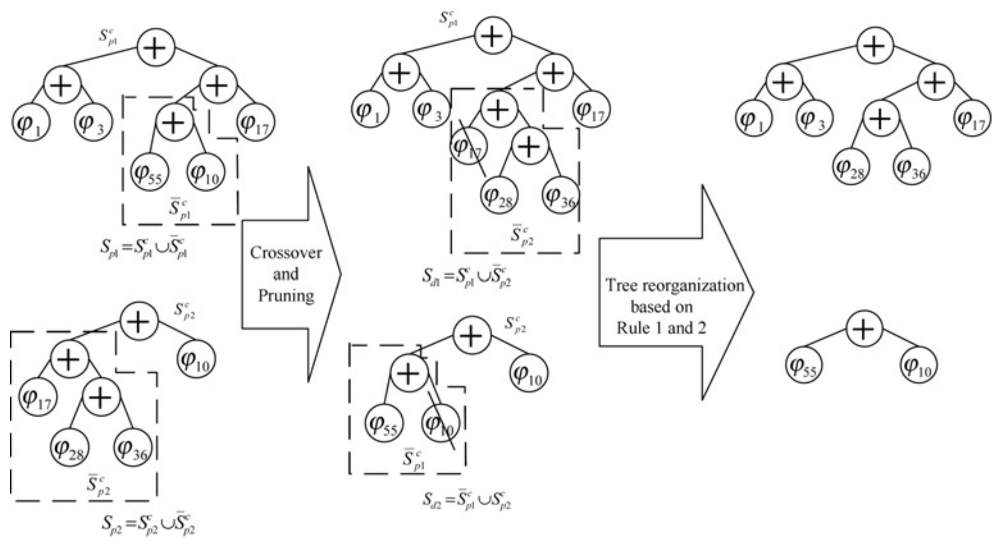

S^c_{p1}
$\bar{S}^c_{p1}$
$S_{p1} = S^c_{p1} \cup \bar{S}^c_{p1}$
S^c_{p2}
$\bar{S}^c_{p2}$
$S_{p2} = S^c_{p2} \cup \bar{S}^c_{p2}$
Crossover and Pruning
$S_{d1} = S^c_{p1} \cup \bar{S}^c_{p2}$
$S_{d2} = \bar{S}^c_{p1} \cup S^c_{p2}$
Tree reorganization based on Rule 1 and 2
φ_1
φ_3
φ_{17}
φ_{55}
φ_{10}
φ_{28}
φ_{36}

Figure 2 *Tree crossover and reorganisation*

considered as the insignificant ones

$$|\hat{c}_{\beta}^{i}| < \eta_j \chi_{\beta}, j = 1, \ldots, \nu \tag{32}$$

It is obvious that different pruning thresholds result in different number of kernel coefficients being considered as insignificant ones. The original best tree is the one with threshold $\eta_j = 0$. If all of the cross-products of input signals evolved from one primary signal are associated with insignificant kernel coefficients, the primary signal is considered as the insignificant primary signal. With the insignificant primary signals being pruned and the tree being reorganised based on tree reorganisation Rules 1 and 2, the parental trees with different sizes as a result of ν pruning thresholds are obtained in Γ_b. After the operation of 'extinction', all but the best tree are removed from the gene pool. As shown in Fig. 4, every tree in the rest of $(G-1)$ spaces is, respectively, filled with the regenerated tree obtained by directly merging one randomly selected tree $\hat{s}_{p1}$ from Γ_b with the tree $\hat{s}_{p2}$ generated through directed initialisation scheme.

The criterion triggering the 'tree extinction and regeneration' operation can be designed as $|E_p^* - E_{p-1}^*| \le \varepsilon$ for continuous M_t generations. The proposed tree extinction and regeneration scheme helps the GP algorithm get around local minima and extend the searching space in the neighbourhood of estimated locally optimal solutions.

4 Computer simulations

The identification of nonlinear systems using the proposed GP algorithm-based Volterra filter is tested through computer simulations. The tuning parameters of the GP algorithm are set as follows.

$M_s \equiv$ the upper limit of GP's running generations = 150.

$M_t \equiv$ the generation threshold triggering the tree extinction and regeneration operator = 5.

$\varepsilon \equiv$ the error deviation threshold triggering the tree extinction and regeneration operator = 0.1.

$\eta_1, \ldots, \eta_6 \equiv$ pruning thresholds = 0, 0.001, 0.005, 0.01, 0.05, 0.1.

G $\equiv$ the number trees in the gene pool = 60.

$L \equiv$ the number of data points used to evaluate the fitness value in (26) = 2000.

The number of chromosomes chosen as the parental chromosomes for crossover, denoted by N_c, is set as $N_c \approx G/3$. The fitness tolerance τ_s for stopping the algorithm is defined based on the magnitude of the noise $\omega(\cdot)$. In the following examples, τ_s is expressed as

$$\tau_s = 1.1\sqrt{\frac{\sum_{n=1}^{L} \omega^2(n)}{L}} \tag{33}$$

where the constant 1.1 is an adjusting constant.

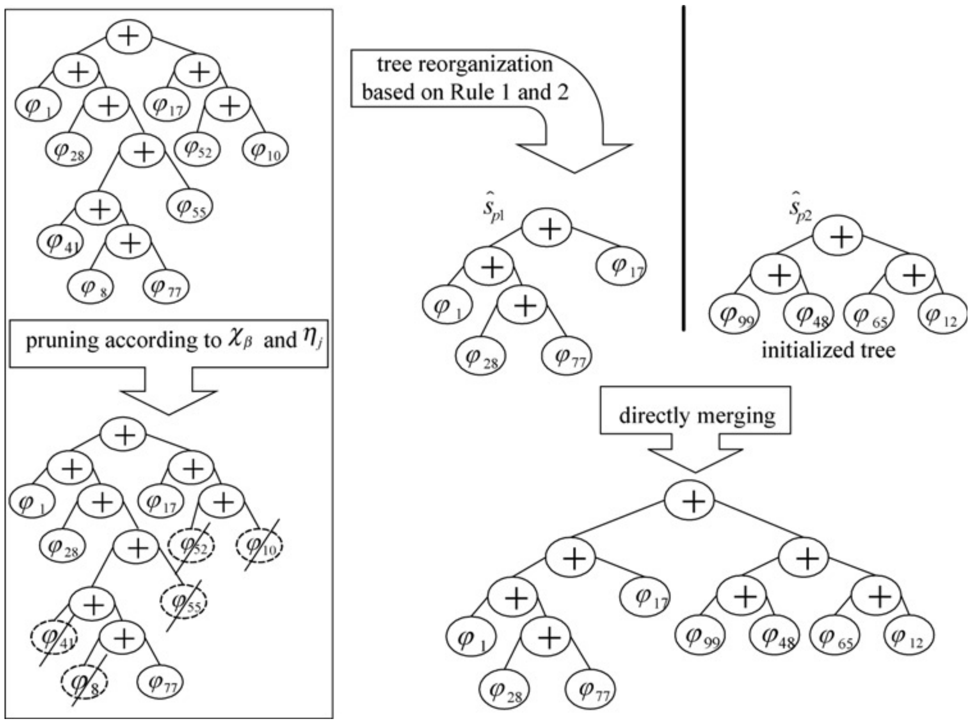


Figure 4 *Illustration of tree regeneration*

Example 1: The system to be identified is a nonlinear system

$$y(n) = 3x^3(n-1)x(n-6) - 5.6x(n)x^2(n-1)x(n-4) + 5.2x(n-2)x^2(n-3)x(n-6) - 4x(n-1)x(n-3)x(n-5) + 3.2x(n-2)x(n-4)x(n-6) + 3.6x(n-2)x(n-3) - 2.4x(n-5)x(n-6) - 2x(n-1)x(n-4) + 2.8x(n-2)x(n-5) + 4.6x(n-3)x(n-6) - 4.3x(n-2) + \omega(n) \tag{34}$$

where $\omega(\cdot)$ denotes the measurement noises. The input signals $x(\cdot)$ are random signals uniformly distributed between -2 and 2. The measurement noises $\omega(\cdot)$ are uniformly distributed signals adjusted so that the signal-to-noise ratio (SNR) of the system in (34) is 10, 20 and 30 dB, respectively. Associated with different SNRs 10 dB, 20 dB and 30 dB, the fitness tolerances τ_s in (33) for stopping the algorithm are set as 8.05, 2.58, 1.03, respectively. A fourth-order Volterra filter with maximum memories 7, 7, 9 and 9 for the first-, second-, third- and fourth-order nonlinearities is utilised to identify the nonlinear system in (34). There are 696 cross-products of input signals and 219 primary signals in the proposed Volterra filter according to (5) and (11). As shown in Table 1, 11 input signal cross-product terms in (35) are located in six sets of cross-products evolved from the primary signals. The GP algorithm is employed to search for these six target primary signals within 219 candidate primary signals. With every initialised tree being set at two layers and four terminal nodes, the reasonable number of chromosomes, G, in the gene pool can be approximately set as 60 according to (28). By selecting the top one-third of the chromosomes as the parental chromosomes for crossover, the number of parental chromosomes $N_c = 20$.

The GP algorithm with and without directed initialisation scheme was run five times for different SNRs to test the effectiveness of the proposed directed initialisation scheme. The computer simulations were conducted on a PC with 1.5 GHz CPU and 512 MB RAM. The fast transversal adaptive algorithm proposed in [19] has been extended to estimate the kernel coefficients in the fourth-order Volterra filter in (34). The running time of the GP algorithm with the least and most running times for different SNRs are compared in Table 2. It is obvious in Table 2 that the directed initialisation scheme improves the convergence of the GP algorithm and requires less running time on an average. Table 2 also shows that the fast transversal adaptive algorithm takes less time than the GP algorithm to produce convergent estimation results. However, the fast transversal adaptive algorithm takes each of 696 kernel coefficients equally important and estimates each of them. To evaluate the performance of different Volterra learning approaches, the squared norm of the coefficient error vector is usually utilised. Let $\boldsymbol{C}$ and $\hat{\boldsymbol{C}}$ be the vector of the original and estimated kernel coefficients. The squared norm of the coefficient error vector is defined as

$$\varsigma = 10\log\frac{\|\boldsymbol{C}-\hat{\boldsymbol{C}}\|_2^2}{\|\boldsymbol{C}\|_2^2} \tag{35}$$

In Table 3, the squared norms of coefficient error vector because of the GP algorithm and the fast transversal adaptive algorithm are compared. It shows that the GP algorithm outperforms the fast transversal adaptive algorithm as the GP algorithm estimates only the kernel coefficients associated with the terms that are significantly dependent on the output.

The effect and efficiency of the proposed tree extinction and regeneration operator are further analysed. Let the GP algorithm with and without tree extinction and regeneration operators be GP1 and GP2, respectively. The same simulation setups were designed to precisely compare the performance of GP1 and GP2. GP1 and GP2 were both applied with the same initial conditions, input, output and noise signals. GP1 was first applied to learn the Volterra filter followed by GP2. GP2 was applied under the same sequence of input, output and noise data points utilised by GP1 until the first tree extinction and regeneration was triggered. The convergence of MSE (in dB) with and without 'tree extinction and regeneration' for SNR equal to 20 dB are illustrated in Fig. 5. It is shown in Fig. 5 that the tree extinction and regeneration operator significantly improves the learning of Volterra filter.

Example 2: The nonlinear system to be identified by a Volterra filter is as following

$$y(n) = \sum_{i=4,5,6}[(x^2(n-i) - 1.2(n-i)x(n-11-i) + x^2(n-11-i)) \times (2 + 1.2x(n-11-i) + 2x(n-16-i))] + \omega(n) \tag{36}$$

Table 1 Primary signals and the cross-products in the Volterra filter (Example 1)

Primary signals	Corresponding cross-products in the Volterra filter
$x(n)$	$x(n-i),\ i = 2$
$x(n)x(n-1)$	$x(n-i)x(n-1-i),\ i = 2, 5$
$x(n)x(n-3)$	$x(n-i)x(n-3-i),\ i = 1, 2, 3$
$x(n)x(n-2)x(n-4)$	$x(n-i)x(n-2-i)x(n-4-i),\ i = 1, 2$
$x(n)x^2(n-1)x(n-4)$	$x(n-i)x^2(n-1-i)x(n-4-i),\ i = 0, 2$
$x^3(n)x(n-5)$	$x^3(n-i)x(n-5-i),\ i = 1$

Table 2 Comparison of running time (Example 1)

Approaches	SNR = 10 dB (s)			SNR = 20 dB (s)			SNR = 30 dB (s)		
	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
GP with directed initialisation	152	654	295	139	669	426	227	807	323
GP without directed initialisation	169	725	406	215	697	465	299	948	371
Fast transversal adaptive algorithm	250			228			228		

Table 3 Comparison of squared norm of coefficient error vector (Example 1)

	SNR = 10 dB	SNR = 20 dB	SNR = 30 dB
Fast transversal adaptive algorithm	−9.51 dB	−19.32 dB	−23.01 dB
GP	−15.56 dB	−25.60 dB	−35.54 dB

where $\omega(\cdot)$ denotes the measurement noises. The input signals $x(\cdot)$ are random signals uniformly distributed between -2 and 2. The measurement noises $\omega(\cdot)$ are uniformly distributed signals adjusted so that the SNR of the system in (36) is 10, 20 and 30dB, respectively. Associated with different SNRs 10 dB, 20 dB and 30 dB, the fitness tolerances τ_s defined according to (33) for stopping the algorithm are set as 9.35, 2.98 and 0.85, respectively. A fourth-order Volterra filter with maximum memories, 22, from the first- to the fourth-order nonlinearities is utilised to identify the nonlinear system in (36). There are 2599 cross-products of input signals and 300 primary signals in the proposed Volterra filter. As shown in Table 4, 27 input signal cross-product terms are located in eight sets of cross-products evolved from the primary signals. With every initialised tree being set at two layers and four terminal nodes, the reasonable number of chromosomes, G, in the gene pool can be approximately set as 75 according to (28). The number of parental chromosomes $N_c = 25$.

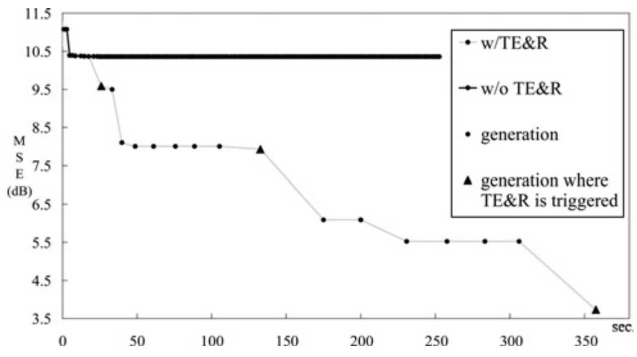


Figure 5 *Convergence of the GP with and without tree extinction and regeneration operator (SNR = 20 dB)*

The running times of the GP algorithm with and without directed initialisation scheme and the running time of the fast transversal adaptive algorithm are compared in Table 4. From the comparison in Table 5, it is obvious that the directed initialisation scheme improves the convergence of the GP algorithm. Note that the fast transversal adaptive algorithm fails to converge to a given MSE tolerance. As the fast transversal adaptive algorithm fails to converge to a given MSE tolerance, the running is calculated until the time after feeding the algorithm 6000 data points. Therefore the running time for the fast transversal adaptive algorithm in Table 5 is given as the time it is stopped rather than the time it converges to the MSE tolerance.

The squared norms of coefficient error vector as shown in (35) because of the GP algorithm and the fast transversal adaptive algorithm are compared in Table 6. It is shown in Table 6 that the GP algorithm outperforms the fast transversal adaptive algorithm for the situations with different SNRs.

Example 3: The nonlinear system to be identified by a Volterra filter is as following

$$y(n) = \sum_{i=0,1,5} 50e^{-(x(n-i))/10} \sin(x(n-2-i)) + \omega(n) \quad (37)$$

where $\omega(\cdot)$ denotes the measurement noise. The input signals $x(\cdot)$ are random signals uniformly distributed between 0 and

Table 4 Primary signals and the cross-products in the Volterra filter (Example 2)

Primary signals	Cross-products in the Volterra filter
$x^2(n)$	$x^2(n-i)$, $i = 4, 5, 6, 15, 16, 17$
$x(n)\,x(n-11)$	$x(n-i)\,x(n-11-i)$, $i = 4, 5, 6$
$x^3(n)$	$x^3(n-i)$, $i = 15, 16, 17$
$x^2(n)\,x(n-5)$	$x^2(n-i)\,x(n-5-i)$, $i = 15, 16, 17$
$x^2(n)\,x(n-11)$	$x^2(n-i)\,x(n-11-i)$, $i = 4, 5, 6$
$x^2(n)\,x(n-16)$	$x^2(n-i)\,x(n-16-i)$, $i = 4, 5, 6$
$x(n)\,x^2$(n-11)	$x(n-i)\,x^2(n-11-i)$, $i = 4, 5, 6$
$x(n)\,x(n-11)$ $x(n-16)$	$x(n-i)\,x(n-11-i)$ $x(n-16-i)$, $i = 4, 5, 6$

Table 5 Comparison of running time (Example 2)

Approaches	SNR = 10 dB (s)			SNR = 20 dB (s)			SNR = 30 dB (s)		
	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
GP with directed initialisation	964	2385	1699	1343	2121	1779	1159	2229	1683
GP without directed initialisation	2144	4077	3216	2471	5198	3534	1071	4358	2612
Fast transversal adaptive algorithm	Fails to converge to the MSE tolerance								

Table 6 Comparison of squared norm of coefficient error vector (Example 2)

	SNR = 10 dB	SNR = 20 dB	SNR = 30 dB
Fast transversal adaptive algorithm	3.8 dB	−5.54 dB	−17 dB
GP with directed initialisation	−7.27 dB	−16.43 dB	−28.28 dB

4. The measurement noises $\omega(\cdot)$ are uniformly distributed signals adjusted so that the SNRof the system in (37) is 20 dB. With the SNR set at 20 dB, the fitness tolerances τ_s in (33) for stopping the algorithm is set as 4.79. A fourth-order Volterra filter with maximum memories, 7, for each order is utilised to identify the system in (37). Referring to (5) and (11), there are 494 cross-products of input signals in the proposed Volterra filter whereas there are only 165 primary signals. As in Example 1, every initialised tree is assumed to contain four terminal nodes in two layers. The reasonable number of chromosomes, G, in the gene pool can approximately be set at 42 according to (24). The number of parental chromosomes $N_c = 14$. The Volterra filter learned by the proposed GP-based learning scheme is shown in Table 7. Out of the 165 candidate primary signals, 10 primary signals are finally learned. Table 7 shows all the cross-products and associated kernel coefficients corresponding to those 10 primary signals.

The MSE convergence (in dB) of the GP algorithm and the fast transversal adaptive algorithm for SNR = 20 dB are shown in Fig. 6. Fig. 6 clearly shows that the GP

Table 7 Volterra filter learned to approximate the nonlinear system in Example 3

Primary signals	Cross-products and kernel coefficients corresponding to every primary signal
$x(n)$	$6.04x(n) - 3.36x(n-1) + 53.6x(n-2) + 51x(n-3) - 5.04x(n-4) - 1.03x(n-5) + 7.92x(n-6) + 54.46x(n-7)$
$x^2(n)$	$-3.75x^2(n) + 2.01x^2(n-1) - 6.44x^2(n-2) - 2.43x^2(n-3) + 7.23x^2(n-4) + 1.83x^2(n-5) - 6.56x^2(n-6) - 6.75x^2(n-7)$
$x(n)x(n-2)$	$-7.87x(n)x(n-2) - 5.45x(n-1)x(n-3) - 0.95x(n-2)x(n-4) - 0.35x(n-3)x(n-5) - 1.67x(n-4)x(n-6) - 5.57x(n-5)x(n-7)$
$x(n)x(n-4)$	$-0.16x(n)x(n-4) + 0.11x(n-1)x(n-5) + 0.12x(n-2)x(n-6) + 0.07x(n-3)x(n-7)$
$x^3(n)$	$0.97x^3(n) - 0.51x^3(n-1) - 7.53x^3(n-2) - 8.7x^3(n-3) - 3.04x^3(n-4) - 0.97x^3(n-5) + 1.75x^3(n-6) - 7.02x^3(n-7)$
$x^2(n)x(n-2)$	$0.25x^2(n)x(n-2) - 0.11x^2(n-1)x(n-3) + 1.04x^2(n-2)x(n-4) - 0.91x^2(n-3)x(n-5) + 0.42x^2(n-4)x(n-6) - 0.42x^2(n-5)x(n-7)$
$x(n)x^2(n-2)$	$3.49x(n)x^2(n-2) + 2.27x(n-1)x^2(n-3) - 0.04x(n-2)x^2(n-4) + 0.98x(n-3)x^2(n-5) + 0.67x(n-4)x^2(n-6) + 2.39x(n-5)x^2(n-7)$
$x^4(n)$	$-0.08x^4(n) + 0.05x^4(n-1) + 1.3x^4(n-2) + 1.38x^4(n-3) + 0.39x^4(n-4) + 0.14x^4(n-5) - 0.14x^4(n-6) + 1.17x^4(n-7)$
$x^3(n)x(n-2)$	$-0.04x^3(n)x(n-2) + 0.02x^3(n-1)x(n-3) - 0.19x^3(n-2)x(n-4) + 0.16x^3(n-3)x(n-5) - 0.07x^3(n-4)x(n-6) + 0.1x^3(n-5)x(n-7)$
$x(n)x^3(n-2)$	$-0.38x(n)x^3(n-2) - 0.15x(n-1)x^3(n-3) - 0.003x(n-2)x^3(n-4) - 0.15x(n-3)x^3(n-5) - 0.1x(n-4)x^3(n-6) - 0.18x(n-5)x^3(n-7)$

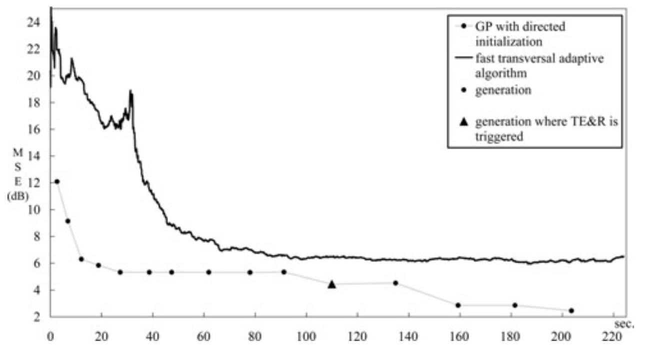


Figure 6 *Convergence of the GP algorithm and the fast transversal adaptive algorithm for SNR = 20 dB*
(Example 3)

algorithm outperforms the fast transversal adaptive algorithm as it converges faster and better. Different from the previous two examples, the nonlinear system to be identified in this example is not a polynomial of cross-products of input signals. It is meaningless to calculate the squared norm of coefficient error in this example.

5 Conclusion

In this paper, the GP algorithm with several modifications is applied to identify a nonlinear system using the Volterra filter. In order to increase GP's search efficiency, the GP algorithm is utilised to search for primary signals rather than cross-products in the Volterra filter. The conventional GP algorithm cannot be directly applied to the learning of Volterra filter as it is easy to get trapped at local minima. In order to solve this problem, an effective directed initialisation scheme, a tree pruning and reorganisation approach, and a new operator called tree extinction and regeneration are proposed. From the simulation results, the proposed modifications significantly improve the learning results. Although this paper focuses on applying the GP algorithm to the identification of nonlinear system using the Volterra filter, the proposed modifications of the GP algorithm can also be applied to other applications and still receive effective improvements because the proposed modifications overcome GP's inherent limitations.

6 References

[1] MATHEWS V.J., SICURANZA G.L.: 'Polynomial signal processing' (Wiley, 2000)

[2] KALOUPTSIDIS N.: 'Signal processing systems: theory and design' (Wiley, 1997)

[3] KUECH F., KELLERMANN W.: 'Partitioned block frequency-domain adaptive second-order volterra filter', *IEEE Trans. Signal Process.*, 2005, **53**, (2), pp. 564–575

[4] GUÉRIN A., FAUCON G., REGINE LE BOUQUIN-JEANNÈS : 'Nonlinear acoustic echo cancellation based on Volterra filters', *IEEE Trans. Speech Audio Process.*, 2003, **11**, (6), pp. 672–683

[5] PARK D.-C., JUNG JEONG T.-K.: 'Complex-bilinear recurrent neural network for equalization of a digital satellite channel', *IEEE Trans. Neural Netw.*, 2002, **13**, (3), pp. 711–725

[6] RAZ G.M., VAN VEEN B.D.: 'Blind equalization and identification of nonlinear and IIR systems – a least squares approach', *IEEE Trans. Signal Process.*, 2000, **48**, (1), pp. 192–200

[7] WENG B., BARNER K.E.: 'Nonlinear system identification in impulsive environment', *IEEE Trans. Signal Process.*, 2005, **53**, (7), pp. 2588–2594

[8] LEVANONY D., BERMAN N.: 'Recursive nonlinear system identification by a stochastic gradient algorithm: stability, performance, and model nonlinearity considerations', *IEEE Trans. Signal Process.*, 2004, **52**, (9), pp. 2540–2550

[9] MONIN A., SALUT G., TEULIERE V.: 'Multisource discrimination using IIR Volterra filtering', *IEEE Trans. Commun.*, 2001, **49**, (5), pp. 922–931

[10] WITRISAL K., LEUS G., PAUSINI M., KRALL C.: 'Equivalent system model and equalization of differential impulse radio UWB systems', *IEEE J. Selected Areas Commun.*, 2005, **23**, (9), pp. 1851–1862

[11] ASYALI M.H., JUUSOLA M.: 'Use of Meixner functions in estimation of Volterra kernels of nonlinear systems with delay', *IEEE Trans. Biomed. Eng.*, 2005, **52**, (2), pp. 229–237

[12] CHON K.H., CHEN Y.-M., HOLSTEIN-RATHLOU N.-H., MARMARELIS V.Z.: 'Nonlinear system analysis of renal autoregulation in normotensive and hypertensive rats', *IEEE Trans. Biomed. Eng.*, 2005, **45**, (3), pp. 342–353

[13] WIDROW B., STEARNS S.D.: 'Adaptive signal processing' (Prentice-Hall, Englewood Cliffs, NJ, 1985)

[14] SCHARF L.L.: 'Statistical signal processing' (Addison-Wesley, 1991)

[15] KOH T., POWERS J.E.: 'Second-order Volterra filtering and its application to nonlinear system identification', *IEEE Trans. Acoust. Speech Signal Process.*, 1985, **ASSP-33**, pp. 1445–1455

[16] SAYADI M., FNAIECH F., NAJIM M.: 'An LMS adaptive second-order Volterra filter with a zeroth-order term: steady-state performance analysis in a time-varying environment', *IEEE Trans. Signal Process.*, 1999, **47**, (3), pp. 872–876

[17] GRIFFITH D.W. JR., ARCE G.R.: 'Partially decoupled Volterra filters: formulation and LMS adaptation', *IEEE Trans. Signal Process.*, 1997, **45**, (6), pp. 1485–1494

[18] EUN C., POWERS E.J.: 'A new Volterra predistorter based on the indirect learning architecture', *IEEE Trans. Signal Process.*, 1997, **45**, (1), pp. 223–227

[19] LEE J., MATHEWS V.J.: 'A fast recursive least squares adaptive second-order Volterra filter and its performance analysis', *IEEE Trans. Signal Process.*, 1993, **41**, (3), pp. 1087–1102

[20] SYED M.A., MATHEWS V.J.: 'Lattice algorithms for recursive least squares adaptive second-order Volterra filtering', *IEEE Trans. Signal Process.*, 1994, **42**, (3), pp. 202–214

[21] GLENTIS G.-O.A., KOUKOULAS P., KALOUPTSIDIS N.: 'Efficient algorithms for Volterra system identification', *IEEE Trans. Signal Process.*, 1999, **47**, (11), pp. 3042–3057

[22] SICURANZA G.L., CARINI A.: 'A multichannel hierarchical approach to adaptive Volterra filters employing filtered-X affine projection algorithms', *IEEE Trans. Signal Process.*, 2005, **53**, (4), pp. 1463–1473

[23] SAYADI M., FNAIECH F., NAJIM M.: 'Multichannel linear and quadratic adaptive filtering based on the Chandrasekhar fast algorithm', *IEEE Trans. Signal Process.*, 1999, **47**, (3), pp. 860–864

[24] CARINI A., MUMOLO E., SICURANZA G.L.: 'V-vector algebra and its application to Volterra-adaptive filtering', *IEEE Trans. Signal Process.*, 1999, **47**, (5), pp. 1585–1598

[25] KOZA J.: 'Genetic programming II: automatic discovery of reusable program' (The MIT Press, 1994)

[26] BANZHAF W., NORDIN P., KELLER R., FRANCONE F.: 'Genetic programming – an introduction on the automatic evolution of computer programs and its application' (Morgan Kaufmann, 1998)

[27] YAO L.: 'Genetic algorithm based identification of nonlinear systems by sparse Volterra filters', *IEEE Trans. Signal Process.*, 1999, **47**, (12), pp. 3433–3435

[28] LIU G.P., KADIRKAMANATHAN V.: 'Multiobjective criteria for neural network structure selection and identification of nonlinear systems using genetic algorithms', *IEE Proc. – Control Theory Appl.*, 1999, **146**, (5), pp. 373–382

[29] NIKOLAEV N.Y., IBA H.: 'Learning polynomial feedforward neural networks by genetic programming and backpropagation', *IEEE Trans. Neural Netw.*, 2003, **14**, (2), pp. 337–350

7 Appendix

Theorem:

$$\binom{d+p}{p} = 1 + \sum_{i=1}^{p} \binom{d+i-1}{i}$$

Proof:

$$\begin{aligned}
&\binom{d+p}{p} - \left[\binom{d+p-1}{p} + \binom{d+p-2}{p-1} \right.\\
&\quad \left. + \binom{d+p-3}{p-2} + \cdots + \binom{d+1}{2} + \binom{d}{1} + 1\right]\\
&= \frac{(d+p)!}{d!p!} - \left[\frac{(d+p-1)!}{p!(d-1)!} + \frac{(d+p-2)!}{(p-1)!(d-1)!}\right.\\
&\quad + \frac{(d+p-3)!}{(p-2)!(d-1)!} + \cdots + \frac{(d+1)!}{2!(d-1)!}\\
&\quad \left. + \frac{d!}{(d-1)!} + 1\right]\\
&= \frac{\begin{array}{c}(d+p)! - d(d+p-1)! - pd\\ (d+p-2)! - p(p-1)d(d+p-3)!\\ - \cdots - dd!p! - d!p!\end{array}}{d!p!} \qquad (38)
\end{aligned}$$

The calculation results of the first two terms in the numerator

$$\begin{aligned}
&(d+p)! - d(d+p-1)!\\
&\quad = (d+p-d)(d+p-1)!\\
&\quad = p(d+p-1)! \qquad (39)
\end{aligned}$$

Based on (39), the calculation results of the first three terms in the numerator

$$\begin{aligned}
&p(d+p-1)! - pd(d+p-2)!\\
&\quad = (p(d+p-1) - pd)(d+p-2)!\\
&\quad = p(p-1)(d+p-2)! \qquad (40)
\end{aligned}$$

Based on (40), the calculation results of the first four terms in the numerator

$$\begin{aligned}
&p(p-1)(d+p-2)! - p(p-1)d(d+p-3)!\\
&\quad = (p(p-1)(d+p-2) - p(p-1)d)(d+p-3)!\\
&\quad = p(p-1)(p-2)(d+p-3)! \qquad (41)
\end{aligned}$$

Therefore it can be inferred from (38)–(41) that the calculation results of the first $(p+1)$ terms in the numerator are

$$p(p-1)(p-2)\dots 2(d+1)! - dd!p! \\ = (p!(d+1) - p!d)d! = p!d! \quad (42)$$

Finally, based on (42), the calculation results of all the $(p+2)$ terms in the numerator

$$p!d! - d!p! = 0 \quad (43)$$

As the numerator in (38) is zero, the theorem can thus be justified. □