

Identification Of Volterra Systems With a Polynomial Neural Network

Robert E. Parker, Jr. and Murali Tummala
Department of Electrical and Computer Engineering
Naval Postgraduate School, Monterey, California 93943

Abstract

The Volterra series finds wide application in the general representation of nonlinear systems. Here, a method of identifying linear and second order time invariant nonlinear systems is proposed using a variation of Group Method of Data Handling (GMDH) algorithm, a polynomial network, employing a combination of quadratic polynomial and linear layers. The principal advantage of this method is that the degree of nonlinearity as well as the memory of the system does not have to be known *a priori* and are determined recursively. The GMDH method allows a Volterra series to be modeled solely from a set of input-output data. System identification using GMDH consists of applying a set of input-output data to train the network by computing the necessary coefficient sets and to select the optimum combination of these coefficient sets to obtain the model parameters.

1 Introduction

Identification of nonlinear systems is of considerable interest in many areas of engineering and science. Recently, with the emergence of neural networks, there is a revived interest in identification of nonlinear systems. Neural network algorithms based on multilayer perceptron type of architecture have been used for this purpose [1].

The Volterra series provides a general representation of nonlinear systems. Confining our discussion for simplicity to third order finite time invariant systems here, the output equation is given by [2]

$$y(n) = \sum_{i=0}^{p-1} a_i x(n-i) + \sum_{i=0}^{q-1} \sum_{j=0}^{q-1} a_{ij} x(n-i)x(n-j) \quad (1)$$

$$+ \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} \sum_{k=0}^{r-1} a_{ijk} x(n-i)x(n-j)x(n-k)$$

where $x(n)$ are the input samples and a_i are the linear coefficients, and a_{ij} and a_{ijk} correspond to the input terms of second and third order, respectively. Typically, modeling with Volterra series based nonlinear systems requires the assumption of a specific system order and then estimating the parameters in some manner, perhaps with an iterative technique. Here we propose a hybrid linear/polynomial neural network to identify the parameters of a Volterra system without the *a priori* knowledge of its order of nonlinearity or memory.

2 The Network

The group method of data handling (GMDH) is a neural network that models a set of data to a user selected polynomial function [3]-[4]. The network consists of a set of input nodes which are filtered through a series of hidden processing layers to produce a single output. The number of inputs to the network is chosen to realize a certain degree of system memory and nonlinearity. The processing elements within each layer each take two inputs from the preceding layer and form combinations of the inputs. Figure 1 shows a schematic diagram of this network with three inputs and three layers.

The network consists of a number of interconnected layers of processing elements. The flow of data is unidirectional or of feed forward nature. Two types of processing elements, linear and quadratic types, are considered here. Figure 2 shows the schematic of a processing element. The output of a linear element is given by

$$y(i, j) = a_j x_{j1} + b_j x_{j2} + c_j \quad (2)$$

IV-561

U.S. Government work not protected by U.S. copyright.

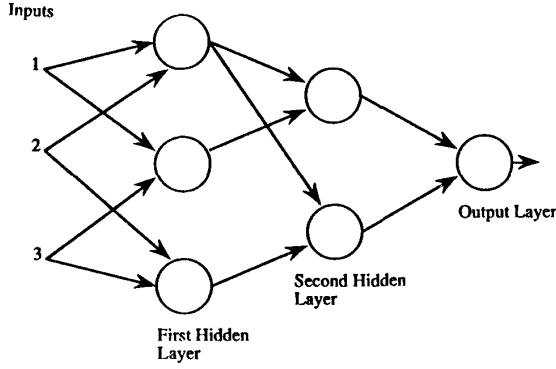


Figure 1: The schematic of the network.

and that of the quadratic element is

$$y(i, j) = a_j x_{j1}^2 + b_j x_{j2}^2 + c_j x_{j1} x_{j2} + d_j x_{j1} + e_j x_{j2} + f_j \quad (3)$$

where a, b, c, \dots are the connection weights of the processing element, and i and j represent the index of layer and node, respectively. In the network considered in this study, the first layer of processing elements are of quadratic type while the following layers are formed by interconnecting a number of linear elements. Thus the network can represent a polynomial of at most order 2. By increasing the number of non-linear element layers to M , however, a polynomial of up to order $2M$ may be realized.

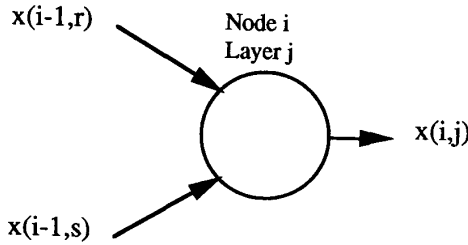


Figure 2: The schematic of a processing element.

The principle behind GMDH is that all possible combinations of the inputs are realized and tested for suitability. Those which perform successfully are retained and unsuccessful combinations are deleted. In order to determine the unsuccessful nodes in a given layer, first we compute the mean squared errors between desired final output and the actual node output. Based on the relative magnitude of the mean

squared error, those nodes with large errors are then eliminated while the ones with smaller error magnitudes are retained for determining the model parameters. Sometimes there is no clear dichotomy between nodes with large and/or small error magnitudes. A threshold value has to be determined for each layer. Ideally the method ultimately eliminates any redundant nodes that are erroneously selected during the node deletion process. However, care should be exercised not to delete one of the essential nodes.

The number of layers and the number of elements in each layer of the network are not predetermined, unlike in the case of the popular multilayer perceptron structure. The number of elements in the $i+1^{\text{th}}$ layer of the network is given by $N_{i+1} = (N_i - 1)!$, where N_i is the number of successful nodes in the i^{th} layer. The network would continue to generate additional layers until a layer with a single successful node is realized. However, frequently it would be necessary to place an upper limit on these numbers for ease of simulation.

Training the network proceeds as follows. From the system identification experiment, the training data is arranged into sets of input data vectors and the corresponding output sample combinations. For layer i , nodes are created by forming all combinations of the outputs from the preceding layer, two at a time; input data nodes serve as the zeroth layer. Each input vector of the training set filters through the preceding layers and produces an output from the j^{th} node of i^{th} layer, $y(i, j)$ expressed as in (2) or (3). At the output of each node, the objective is to produce the true output of the system being modeled, $y(k)$, the output for the k^{th} training data set. With N training data vectors, the set of N output equations can be compactly represented as

$$\mathbf{y}(i, j) = X_j \theta_j \quad (4)$$

where

$$\mathbf{y}(i, j) = [y_1(i, j) \ y_2(i, j) \ \dots \ y_N(i, j)]^T$$

$$X_j = [\mathbf{x}_{1j}^T \ \mathbf{x}_{2j}^T \ \dots \ \mathbf{x}_{Nj}^T]^T$$

$$\mathbf{x}_{kj} = [x_{kj1}^2 \ x_{kj2}^2 \ x_{kj1}x_{kj2} \ x_{kj1} \ x_{kj2} \ 1]$$

$$\theta_j = [a_j \ b_j \ c_j \ d_j \ d_j \ e_j \ f_j]^T$$

for quadratic element and

$$\mathbf{x}_{kj} = [x_{kj1} \ x_{kj2} \ 1]$$

$$\theta_j = [a_j \ b_j \ c_j]^T$$

for linear element,

$\mathbf{y}(i, j)$ is an $N \times 1$ j^{th} node output vector corresponding to N input training data vectors \mathbf{x}_{kj} , X_j is the data matrix of size $N \times 6$ for a quadratic or $N \times 3$

for a linear element, and $k = 1, 2, \dots, N$ represents the training data set index. The coefficient vector θ_j is then obtained by solving (4) to minimize the mean squared error between $y(k)$ and $y_k(i, j)$:

$$\theta_j = (X_j^T X_j)^{-1} X_j^T \mathbf{y}(i, j) \quad (5)$$

This procedure is repeated for each element of the layer and for all layers of the network as the training progresses from the input layer to an unspecified output layer.

After the coefficients have been determined for each node in the given layer, they are subjected to a "threshold test" to select only those nodes with the best performance. Removal of poorly performing nodes allows resources to be dedicated to nodes with desirable characteristics and checks the otherwise explosive growth of nodes in successive layers to be kept manageable. Performance of nodes is determined by computing the mean squared error of each element after processing the data vectors of the test set; the test data set is produced in much the same way as the training data set from the system identification experiment. There is no definitive manner of setting the threshold, but one approach involves rejecting those elements whose mean square error exceeds that of the best performing element by some ratio selected by the user.

The connection weights of processing elements in each layer are estimated with independent training data sets. For estimating the connection weights of nodes in the j^{th} layer, the training and/or test data is first propagated through the previous $j - 1$ layers of the network. The outputs from the successful nodes in the $j - 1^{\text{th}}$ layer are then used as inputs to the nodes in the j^{th} layer. In this manner, nodes retained after the deletion process influence the selection of the later ones. While independent training data sets are used for estimating the connection weights, all testing is performed with a single test data set in this study.

The network grows layer by layer until the best performance by the network has been realized. As each successive layer is added, typically the mean squared error of the best performing node decreases. Eventually, however, a point of minimum mean squared error will be reached and further layers degrade performance. Figure 3 illustrates this trend for a third order Volterra system identification example.

Once the final layer has been chosen, the best performing node is selected as the output element and all other nodes in that layer are deleted. Further, the nodes from previous layers that do not feed the output node are also deleted by tracing the data flow path backwards.

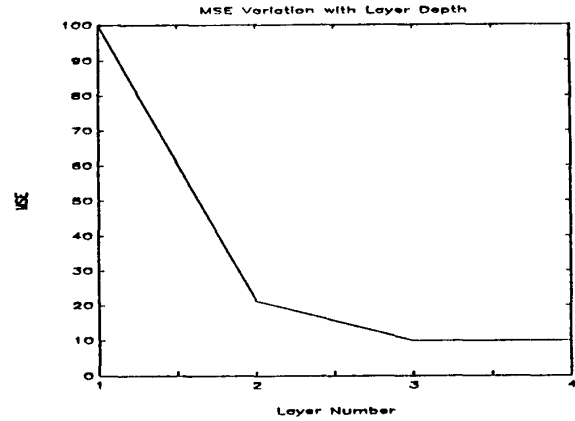


Figure 3: Plot of the mean squared error versus the layer growth.

In order to identify the parameters of a Volterra system of arbitrary order and memory we arrange the input-output data into N number of training and testing data sets as per (4). We now pass these data sets through each layer of the network to determine the coefficients of each processing element in each layer while eliminating the poorly performing nodes. Once we choose the output layer as outlined in the previous discussion and trace the flow path backwards, the coefficients of all the processing elements in the flow path are appropriately combined to obtain the Volterra system parameters.

3 Results

To characterize the performance of the network in modeling nonlinear systems, we have chosen both nonlinear and linear difference equations to represent the reference system to be modeled in the system identification experiment. In this paper, the systems chosen are limited to those of quadratic nonlinearity or linear.

A sequence of uniformly distributed random numbers was used as input to the reference system as well as the model being realized by the network. The length of the input sequence was chosen to satisfy the size of anticipated training and testing data requirements for the network. The input data is passed through the reference system to obtain corresponding output samples.

In this simulation, the network was arbitrarily set to employ four input nodes. Typically, a large number of input layer nodes would be chosen where the degree of system memory was in doubt. Then the network

could indicate the appropriate number of significant inputs as the node deletion process is carried out at each layer; thus giving an estimation of the system memory. If too few inputs are given, the network would not converge to a satisfactory solution.

For the quadratic nonlinear systems tested here, two layers should have been sufficient for accurate modeling. In practice, at least three layers were necessary for accurate modeling. The network was typically terminated at five layers if a satisfactory solution had not been obtained to that point.

To estimate and test the node connection weights, the input/output data was arranged into vectors containing an output and a predetermined number of input data samples corresponding to the chosen reference system, given by

$$z_k = [y(n) \quad x(n) \quad x(n-1) \quad \dots \quad x(n-P)].$$

The minimum number of such vectors required is six for the quadratic processing element in any layer and three for the linear element. In order to produce a better mean squared error estimate of the at each node, more data than the minimum required must be used. Typically $N = 24$ was used for each layer in our simulations which gave satisfactory results with a reasonable layer convergence. A larger N gave no appreciable increase in accuracy and often slowed down the simulation of the network due to the greatly increased number of calculations to be performed. The testing data set also employed $N = 24$.

After the network had grown to the extent that further layers increased the mean squared error of the best performing nodes in these additional layers, the network was considered to have converged. Figure 3 shows a minimum at layer number three indicating that the output node must be the best performing node of layer three. Once the best performing node of the output layer was identified, nodes in the preceding layers that are not contributing to it are removed from the network. By combining the coefficients at each node and working backward through the layers, a polynomial representation is obtained. Inconsequential elements were removed from the expression to give a final system representation.

The three different reference systems used in our simulation are

$$\begin{aligned} y_1(n) &= (3x(n) + 2x(n-2))^2 \\ y_2(n) &= x(n) + 3x(n-1) + x(n-3) \\ y_3(n) &= 3x(n) + x^2(n-1) + 1.5x(n-2) \end{aligned}$$

and their counterparts estimated by the network are

$$y_1(n) = 9x^2(n) + 12x(n)x(n-2) + 4x^2(n-2)$$

$$\begin{aligned} y_2(n) &= x(n) + 3x(n-1) + x(n-3) \\ y_3(n) &= 2.94x(n) + 1.03x^2(n-1) + 1.58x(n-2). \end{aligned} \quad (6)$$

Note that reference systems 1 and 3 have quadratic nonlinear terms while 2 is a linear system. In modeling systems 1 and 3, the first layer of the network is realized by quadratic elements while the following layers are formed by linear elements. However, system 2 is modeled by employing only linear elements. The parameter estimation accuracy is almost perfect for systems 1 and 2, whereas noticeable estimation errors are encountered for system 3.

4 Conclusion

Although GMDH has an expensive price due to the number of large matrix calculations involved, the following advantages are conferred. The precise degree of nonlinearity of the system being studied may not be known in most instances. The proposed method, because of its dynamic layer appending and elimination of poorly performing processing elements as specified by a threshold test, can effectively determine the degree of nonlinearity. The same applies to the length of the system memory, which can be handled by starting with more nodes in the input layer than are necessary. Additionally, the method can be made data adaptive for slowly time varying cases.

References

- [1] S. Chen, S. A. Billings and P. M. Grant, "Non-linear System Identification Using Neural Networks," *Int. J. Control*, Vol. 51, No. 6, pp. 1191-1214, June 1990.
- [2] M. Tummala, "Iterative Algorithm for Identification of Third Order Volterra Systems," In *Proc. 1991 Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 3489-3492, Toronto, Canada, May 1991.
- [3] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley: Reading, MA, 1990.
- [4] A. G. Ivakhnenko, "Polynomial Theory of Complex Systems," *IEEE Trans. Systems, Man & Cyber.*, SMC-12, 364-378, October 1971.