3

Neural network Hammerstein models

3.1 Introduction

This chapter deals with gradient-based learning algorithms for training neural network Hammerstein models. As in the case of neural network Wiener models discussed in Chapter 2, four different gradient calculation algorithms, i.e., backpropagation for series-parallel models (BPS), backpropagation (BPP), the sensitivity method (SM), and backpropagation through time (BPTT) for parallel models are derived. Having the rules for gradient calculation derived, steepest descent or other gradient-based learning algorithms can be implemented easily. Besides steepest descent algorithms, four other learning algorithms, which combine steepest descent algorithms with the recursive least squares (RLS) algorithm or the recursive pseudolinear regression (RPLR) algorithm, are proposed. For the truncated BPTT algorithm, gradient calculation accuracy is analyzed. It is shown that gradient calculation accuracy depends on impulse responses of sensitivity models and the linear dynamic model. Knowing these impulse responses, the errors of the calculation of partial derivatives of the model output w.r.t. model parameters can be evaluated. Computational complexity of the algorithms is analyzed and expressed in terms of the polynomial orders na and nb, the number of nonlinear nodes, and the number of unfolded time steps for the BPTT algorithm.
This chapter is organized as follows: In Section 3.2, the identification problem is formulated. Section 3.3 introduces both the series-parallel and parallel neural network Hammerstein models. Gradient calculation in the SISO and MIMO Hammerstein models is considered in Section 3.4. Based on sensitivity models, an analysis of gradient calculation accuracy with the truncated BPTT algorithm is performed. The results of this analysis are shown to be useful in the determination of the number of unfolded time steps, which is necessary to calculate the gradient at an assumed level of accuracy. Section 3.4 gives also a comparison of computational complexity of the algorithms. Section 3.5 contains a comparative simulation study based on the identification of a second order Hammerstein system. The combined steepest descent and the RLS or

RPLR algorithms, which use different approximations of the gradient obtained with BPS, BPP, the SM, and truncated BPTT, are described in Section 3.6. Finally, a few concluding remarks are presented in Section 3.7.

3.2 Problem formulation

The Hammerstein system, shown in Fig. 3.1, is an example of a block-oriented nonlinear system composed of a zero-memory nonlinear element followed by a linear dynamic system.
Let $f(\cdot)$ denote the characteristic of the nonlinear element; $a_1, \ldots, a_{na}$, $b_1, \ldots, b_{nb}$, are the parameters of the linear dynamic system, q^{-1} is the backward shift operator, and $\varepsilon(n)$ is the additive output disturbance. The output $y(n)$ of the Hammerstein system to the input $u(n)$ at the time n is

$$y(n) = \frac{B(q^{-1})}{A(q^{-1})} f\big(u(n)\big) + \varepsilon(n), \tag{3.1}$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}, \tag{3.2}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{nb} q^{-nb}. \tag{3.3}$$

The following assumptions are made about the system:
Assumption 3.1. The function $f(\cdot)$ is continuous.
Assumption 3.2. The linear dynamic system is casual and asymptotically stable.
Assumption 3.3. The polynomial orders na and nb are known.
The identification problem can be formulated as follows: Given a set of the system input and output data $\{u(n), y(n)\}$, $n = 1, 2, ..., N$, the goal is to estimate the parameters of the linear dynamic system and the characteristic of the nonlinear element minimizing the following global cost function:

$$J = \frac{1}{2} \sum_{n=1}^{N} \big(y(n) - \hat{y}(n)\big)^2, \tag{3.4}$$

where $\hat{y}(n)$ is the output of the neural network Hammerstein model. This can be done by an off-line iterative procedure that uses the whole training set and is called batch learning or the batch mode. For control or filtering applications, it may be more convenient to use another iterative learning procedure that employs the local cost function

$$J(n) = \frac{1}{2} \big(y(n) - \hat{y}(n)\big)^2. \tag{3.5}$$

This results in an on-line learning method that is also called pattern learning or the sequential mode as it process the training data sequentially. In both

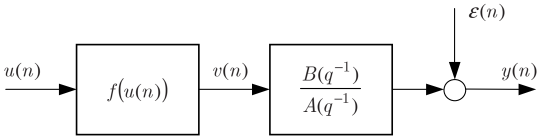

Fig. 3.1. SISO Hammerstein system

learning procedures, the weights are updated along the negative gradient of the cost function J or $J(n)$, respectively. Applying a pattern learning version of the steepest descent method, the actual values of the linear dynamic model parameters $\hat{a}_k(n)$, $\hat{b}_k(n)$ and any parameter $w_c(n)$ of the nonlinear element model are updated as follows:

$$\hat{a}_k(n) = \hat{a}_k(n-1) - \eta \frac{\partial J(n)}{\partial \hat{a}_k(n-1)}, \ k = 1, \ldots, na, \tag{3.6}$$

$$\hat{b}_k(n) = \hat{b}_k(n-1) - \eta \frac{\partial J(n)}{\partial \hat{b}_k(n-1)}, \ k = 1, \ldots, nb, \tag{3.7}$$

$$w_c(n) = w_c(n-1) - \eta \frac{\partial J(n)}{\partial w_c(n-1)}, \ c = 1, \ldots, 3M+1, \tag{3.8}$$

where η is the learning rate, and $3M+1$ is the number of adjustable weights of the nonlinear element model.

3.3 Series-parallel and parallel neural network Hammerstein models

Neural network Hammerstein models, considered in this section, consist of a multilayer perceptron model of the nonlinear element followed by a model of the linear dynamic system. Both the SISO and MISO structures of neural network Hammerstein models are introduced. In the SISO case, a single linear node with two tapped delay lines forms the model of the linear dynamic system. The number of linear nodes in the MISO case is equal to the number of model outputs while the number of tapped delay lines equals the sum of the number of model inputs and outputs.

3.3.1 SISO Hammerstein models

Consider a neural network model of the nonlinear element of a multilayer perceptron architecture, composed of one hidden layer with M nonlinear nodes and one linear output node (Fig. 3.2). The output $\hat{f}\big(u(n),\mathbf{w}\big)$ of this model to the input $u(n)$ is

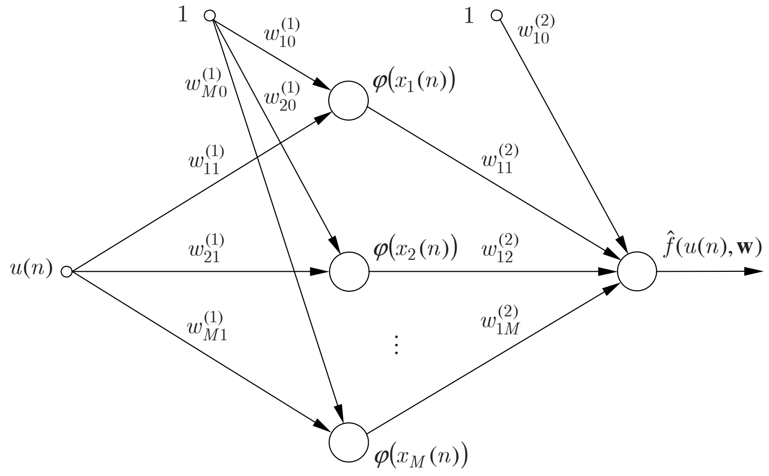

Fig. 3.2. Neural network model of the SISO nonlinear element

$$\hat{f}\big(u(n),\mathbf{w}\big) = \sum_{j=1}^{M} w_{1j}^{(2)} \varphi\big(x_j(n)\big) + w_{10}^{(2)}, \tag{3.9}$$

$$x_j(n) = w_{j1}^{(1)} u(n) + w_{j0}^{(1)}, \tag{3.10}$$

where $\mathbf{w} = [w_{10}^{(1)} \dots w_{M1}^{(1)}\ w_{10}^{(2)} \dots w_{1M}^{(2)}]^T$ is the parameter vector, $x_j(n)$ is the activation of the jth node of the hidden layer at the time n, $\varphi(\cdot)$ is the activation function of hidden layer nodes, and $w_{10}^{(1)}, \dots, w_{M1}^{(1)}$, and $w_{10}^{(2)}, \dots, w_{1M}^{(2)}$ are the weights and the thresholds of hidden layer nodes and the output node, respectively. Both the feedforward and recurrent neural networks can be employed as models of the Hammerstein system. In the feedforward model (Fig. 3.3), past values of the system input $u(n)$ and the system output $y(n)$ are used as model inputs:

$$\hat{y}(n) = \big[1 - \hat{A}(q^{-1})\big] y(n) + \hat{B}(q^{-1}) \hat{f}\big(u(n),\mathbf{w}\big), \tag{3.11}$$

where

$$\hat{A}(q^{-1}) = 1 + \hat{a}_1 q^{-1} + \cdots + \hat{a}_{na} q^{-na}, \tag{3.12}$$

$$\hat{B}(q^{-1}) = \hat{b}_1 q^{-1} + \cdots + \hat{b}_{nb} q^{-nb}. \tag{3.13}$$

Unlike the feedforward model, the recurrent one (Fig. 3.4) does not use past values of the system output $y(n)$ but past values of its own output $\hat{y}(n)$ instead:

$$\hat{y}(n) = \big[1 - \hat{A}(q^{-1})\big] \hat{y}(n) + \hat{B}(q^{-1}) \hat{f}\big(u(n),\mathbf{w}\big). \tag{3.14}$$

In the system identification literature, the models of these two types are traditionally known as the series-parallel model and the parallel model [112, 121].

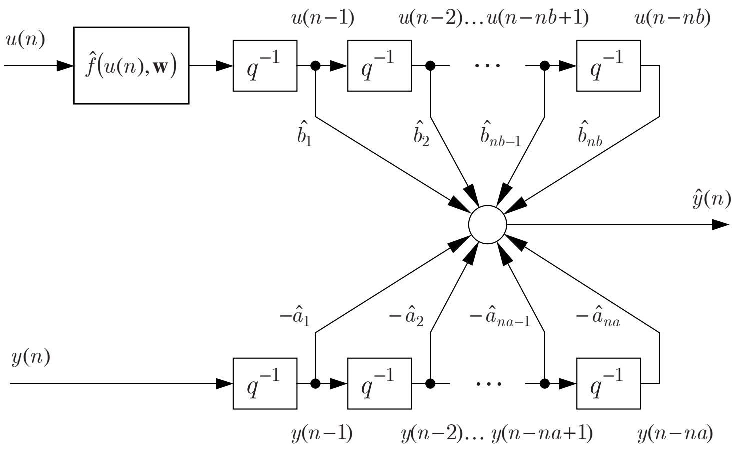

Fig. 3.3. Series-parallel SISO neural network Hammerstein model

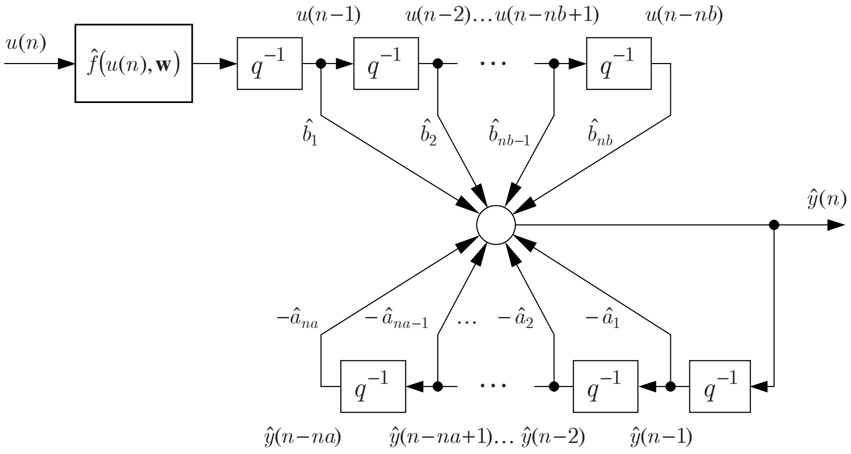

Fig. 3.4. Parallel SISO neural network Hammerstein model

While the series-parallel model is based on the equation error definition, the parallel model corresponds to the output error definition. The identification error for the series-parallel model (3.11) is

$$\begin{aligned} e(n) = y(n) - \hat{y}(n) = \hat{A}(q^{-1})\frac{B(q^{-1})}{A(q^{-1})}f\big(u(n)\big) \\ - \hat{B}(q^{-1})\hat{f}\big(u(n),\mathbf{w}\big) + \hat{A}(q^{-1})\varepsilon(n). \end{aligned} \tag{3.15}$$

Therefore, it is clear that if the system is disturbance free ($\varepsilon(n) = 0$) and model parameters converge to true parameters of the system, then the identification error $e(n)$ converges to zero. If we assume that the system output is corrupted by the additive white noise disturbance ($\varepsilon(n) \neq 0$) and model parameters are equal to their true values, the identification error is correlated, i.e., $e(n) = \hat{A}(q^{-1})\varepsilon(n)$.
For the parallel model of the Hammerstein system (3.14), the identification error is

$$e(n) = y(n) - \hat{y}(n) = \frac{B(q^{-1})}{A(q^{-1})} f\big(u(n)\big) - \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} \hat{f}\big(u(n),\mathbf{w}\big) + \varepsilon(n). \tag{3.16}$$

As in the previous case, the identification error converges also to zero if model parameters converge to their true values and the system is disturbance free. However, contrary to the series-parallel model, if model parameters are equal to true parameters of the system, for a system corrupted by the additive white noise, the identification error is not correlated, i.e., $e(n) = \varepsilon(n)$.
The characterization of the Hammerstein system is not unique as the nonlinear element and the linear dynamic system are connected in series. In other words, Hammerstein systems described by $\big(B(q^{-1})/A(q^{-1})\big)/\alpha$ and $\alpha f\big(u(n)\big)$ reveal the same input-output behavior for any $\alpha \neq 0$. Therefore, to obtain a unique characterization of the neural network model, either the nonlinear element model or the linear dynamic model should be normalized. To normalize the gain of the linear dynamic model to 1, the model weights are scaled as follows: $\hat{b}_k = \tilde{b}_k/\alpha$, $k = 1, \ldots, nb$, $w_{1j}^{(2)} = \alpha \tilde{w}_{1j}^{(2)}, j = 1, \ldots, M$, where $\tilde{b}_k$, $\tilde{w}_{1j}^{(2)}$ denote the parameters of the unnormalized model, and $\alpha = \tilde{B}(1)/\tilde{A}(1)$ is the linear dynamic model gain.

3.3.2 MIMO Hammerstein models

Consider a parallel MIMO neural network Hammerstein model with nu inputs, ny outputs, nf outputs of the nonlinear element model, and M nonlinear nodes in the hidden layer. The lth output of the nonlinear element model (Fig. 3.5) at the time n is

$$\hat{f}_l\big(\mathbf{u}(n),\mathbf{w}_l\big) = \sum_{j=1}^{M} w_{lj}^{(2)} \varphi\big(x_j(n)\big) + w_{l0}^{(2)},\ l = 1, \ldots, nf, \tag{3.17}$$

$$x_j(n) = \sum_{i=1}^{nu} w_{ji}^{(1)} u_i(n) + w_{j0}^{(1)}, \tag{3.18}$$

and the lth vector of weights is defined as

$$\mathbf{w}_l = \big[w_{10}^{(1)}, \ldots, w_{Mnu}^{(1)},\ w_{l0}^{(2)} \ldots, w_{lM}^{(2)}\big]^T. \tag{3.19}$$

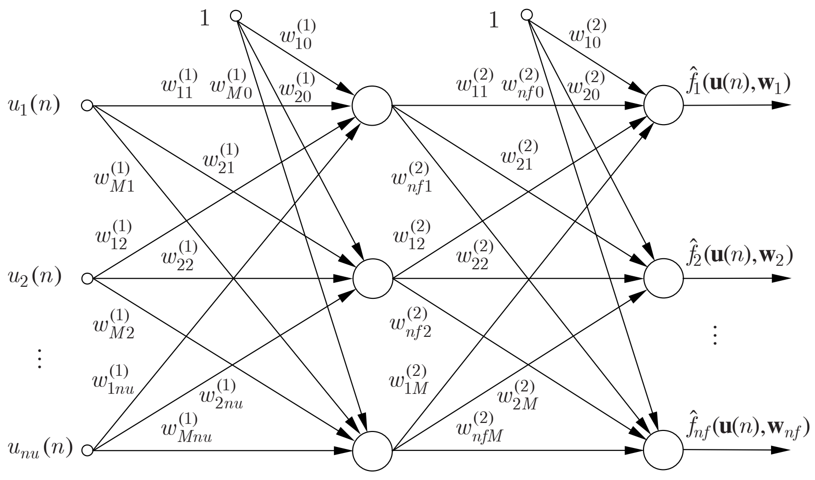

Fig. 3.5. Neural network model of the MIMO nonlinear element

The output $\hat{\mathbf{y}}(n)$ of the parallel MIMO Hammerstein model can be expressed as

$$\hat{\mathbf{y}}(n) = -\sum_{m=1}^{na} \hat{\mathbf{A}}^{(m)}\hat{\mathbf{y}}(n-m) + \sum_{m=1}^{nb} \hat{\mathbf{B}}^{(m)}\hat{\mathbf{f}}\big(\mathbf{u}(n-m),\mathbf{W}\big), \tag{3.20}$$

where

$$\hat{\mathbf{y}}(n) = \big[\hat{y}_1(n)\dots\hat{y}_{ny}(n)\big]^T, \tag{3.21}$$

$$\mathbf{u}(n) = \big[u_1(n)\dots u_{nu}(n)\big]^T, \tag{3.22}$$

$$\hat{\mathbf{f}}\big(\mathbf{u}(n),\mathbf{W}\big) = \big[\hat{f}_1\big(\mathbf{u}(n),\mathbf{w}_1\big)\dots\hat{f}_{nf}\big(\mathbf{u}(n),\mathbf{w}_{nf}\big)\big]^T, \tag{3.23}$$

$$\hat{\mathbf{A}}^{(m)} \in \mathbb{R}^{ny\times ny}, \tag{3.24}$$

$$\hat{\mathbf{B}}^{(m)} \in \mathbb{R}^{ny\times nf}, \tag{3.25}$$

$$\mathbf{W} = \big[w_{10}^{(1)},\dots,w_{Mnu}^{(1)},\ w_{10}^{(2)}\dots,w_{nfM}^{(2)}\big]^T. \tag{3.26}$$

From (3.20), it follows that the tth output of the parallel model is

$$\hat{y}_t(n) = -\sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)}\hat{y}_{m_1}(n-m) + \sum_{m=1}^{nb}\sum_{m_1=1}^{nf} \hat{b}_{tm_1}^{(m)}\hat{f}_{m_1}\big(\mathbf{u}(n-m),\mathbf{w}_{m_1}\big), \tag{3.27}$$

where the superscript m associated with the elements $\hat{a}$ and $\hat{b}$ denotes the matrix number, and the subscripts t and m_1 denote the row and the column number, respectively. Replacing $\hat{y}_{m_1}(n-m)$ with $y_{m_1}(n-m)$ on the r.h.s. of (3.27), a series-parallel model can be obtained:

$$\hat{y}_t(n) = -\sum_{m=1}^{na}\sum_{m_1=1}^{ny}\hat{a}_{tm_1}^{(m)} y_{m_1}(n-m) + \sum_{m=1}^{nb}\sum_{m_1=1}^{nf}\hat{b}_{tm_1}^{(m)}\hat{f}_{m_1}\big(\mathbf{u}(n-m),\mathbf{w}_{m_1}\big). \quad (3.28)$$

Note that $\hat{\mathbf{A}}^{(m)}$, $m = 1,\ldots,na$, are diagonal matrixes if the outputs $\hat{y}_t(n)$, $t = 1,\ldots,ny$, are mutually independent, i.e., they do not depend on $\hat{y}_{m_1}(n-m)$, where $m_1 \neq t$. Moreover, if, additionally, $\hat{\mathbf{B}}^{(m)}$, $m = 1,\ldots,nb$, are diagonal matrices, a MIMO Hammerstein model with uncoupled dynamics is obtained.

3.4 Gradient calculation

Due to its static nature, the gradient in the series-parallel SISO Hammerstein model, shown in Fig. 3.3, can be calculated with the well-known backpropagation algorithm. Although the backpropagation algorithm can be also employed to calculate the gradient in recurrent models, it fails to evaluate the gradient properly. This often results in an extremely slow convergence rate. One method of gradient calculation in recurrent models is the sensitivity method, known also as dynamic backpropagation or on-line recurrent backpropagation [23, 121, 139, 150]. In this method, to compute partial derivatives of the cost function, a set of linear difference equations is solved on-line by simulation. An alternative to the SM is the backpropagation through time method [163], which uses a model unfolded back in time. In this case, the gradient can be evaluated with a method similar to backpropagation. With both of these methods, a higher convergence rate may be expected. Nevertheless, their computational complexity is higher than the computational complexity of the backpropagation method.

3.4.1 Series-parallel SISO model. Backpropagation method

The output of the series-parallel Hammerstein model depends on past inputs and outputs of the system. There are no feedback connections and the computationally effective backpropagation learning algorithm (BPS) can be used to calculate the gradient. Differentiating (3.11), partial derivatives of $\hat{y}(n)$ w.r.t. model parameters can be obtained as follows:

$$\frac{\partial \hat{y}(n)}{\partial \hat{a}_k} = -y(n-k), \; k = 1,\ldots,na, \quad (3.29)$$

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \hat{f}\big(u(n-k),\mathbf{w}\big), \; k = 1,\ldots,nb, \quad (3.30)$$

$$\frac{\partial \hat{y}(n)}{\partial w_c} = \sum_{m=1}^{nb}\hat{b}_m \frac{\partial \hat{f}\big(u(n-m),\mathbf{w}\big)}{\partial w_c}, \; c = 1,\ldots,3M+1, \quad (3.31)$$

where w_c denotes any parameter of the nonlinear element model. Partial derivatives of $\hat{f}\big(u(n),\mathbf{w}\big)$ w.r.t. the elements of the vector $\mathbf{w}$ can be calculated differentiating (3.9):

$$\frac{\partial \hat{f}\big(u(n),\mathbf{w}\big)}{\partial w_{j1}^{(1)}} = \frac{\partial \hat{f}\big(u(n),\mathbf{w}\big)}{\partial \varphi\big(x_j(n)\big)} \frac{\partial \varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial w_{j1}^{(1)}} = w_{1j}^{(2)} \varphi'\big(x_j(n)\big) u(n), \quad (3.32)$$

$$\frac{\partial \hat{f}\big(u(n),\mathbf{w}\big)}{\partial w_{j0}^{(1)}} = \frac{\partial \hat{f}\big(u(n),\mathbf{w}\big)}{\partial \varphi\big(x_j(n)\big)} \frac{\partial \varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial w_{j0}^{(1)}} = w_{1j}^{(2)} \varphi'\big(x_j(n)\big), \quad (3.33)$$

$$\frac{\partial \hat{f}\big(u(n),\mathbf{w}\big)}{\partial w_{1j}^{(2)}} = \varphi\big(x_j(n)\big), \quad (3.34)$$

$$\frac{\partial \hat{f}\big(u(n),\mathbf{w}\big)}{\partial w_{10}^{(2)}} = 1, \quad (3.35)$$

where $j = 1, \ldots, M$.

3.4.2 Parallel SISO model. Backpropagation method

Owing to its low computational complexity, the backpropagation method is also used for training recurrent models in some cases. Neglecting the dependence of past outputs of the parallel model (3.14) on its parameters, the following expressions can be obtained:

$$\frac{\partial \hat{y}(n)}{\partial \hat{a}_k} = -\hat{y}(n-k), \ k = 1, \ldots, na, \quad (3.36)$$

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \hat{f}\big(u(n-k),\mathbf{w}\big), \ k = 1, \ldots, nb, \quad (3.37)$$

$$\frac{\partial \hat{y}(n)}{\partial w_c} = \sum_{m=1}^{nb} \hat{b}_m \frac{\partial \hat{f}\big(u(n-m),\mathbf{w}\big)}{\partial w_c}, \ c = 1, \ldots, 3M+1. \quad (3.38)$$

3.4.3 Parallel SISO model. Sensitivity method

Parallel models are dynamic systems themselves as their outputs depend not only on current and past system inputs but also on past model outputs. This makes the calculation of the gradient more complex. The differentiation of (3.14) w.r.t. model parameters yields

$$\frac{\partial \hat{y}(n)}{\partial \hat{a}_k} = -\hat{y}(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{a}_k}, \ k = 1, \ldots, na, \quad (3.39)$$

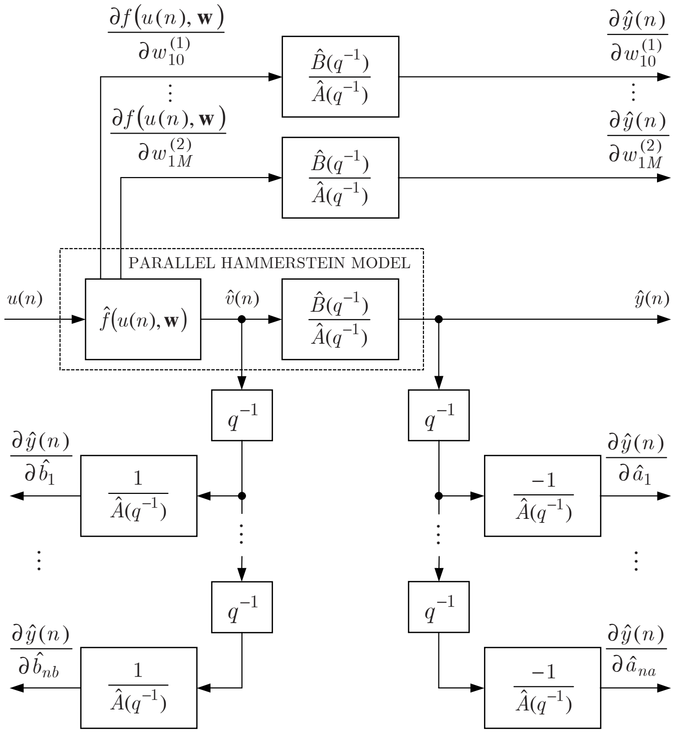
$\frac{\partial f(u(n),\mathbf{w})}{\partial w_{10}^{(1)}}$
$\frac{\partial f(u(n),\mathbf{w})}{\partial w_{1M}^{(2)}}$
$\frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}$
$\frac{\partial \hat{y}(n)}{\partial w_{10}^{(1)}}$
$\frac{\partial \hat{y}(n)}{\partial w_{1M}^{(2)}}$
PARALLEL HAMMERSTEIN MODEL
$u(n)$
$\hat{f}(u(n),\mathbf{w})$
$\hat{v}(n)$
$\hat{y}(n)$
q^{-1}
$\frac{\partial \hat{y}(n)}{\partial \hat{b}_1}$
$\frac{1}{\hat{A}(q^{-1})}$
$\frac{-1}{\hat{A}(q^{-1})}$
$\frac{\partial \hat{y}(n)}{\partial \hat{a}_1}$
$\frac{\partial \hat{y}(n)}{\partial \hat{b}_{nb}}$
$\frac{\partial \hat{y}(n)}{\partial \hat{a}_{na}}$

Fig. 3.6. Parallel model of the Hammerstein system and its sensitivity models

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \hat{f}\big(u(n-k),\mathbf{w}\big) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{b}_k}, \; k = 1,\ldots,nb, \tag{3.40}$$

$$\frac{\partial \hat{y}(n)}{\partial w_c} = \sum_{m=1}^{nb} \hat{b}_m \frac{\partial \hat{f}\big(u(n-m),\mathbf{w}\big)}{\partial w_c} - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial w_c}, \; c=1,\ldots,3M+1. \tag{3.41}$$

Equations (3.39) – (3.41) define the sensitivity models shown in Fig. 3.6. To obtain partial derivatives, these linear difference equations of the order na are solved on-line. Note that this requires the simulation of $na + nb + 3M + 1$ sensitivity models.

In the SM, we assume that the parameters of the linear dynamic model are time invariant. This is true in the batch mode but not in the sequential mode, in which these parameters are updated after the presentation of each learning pattern. Thus, only an approximate gradient can be obtained if we apply a recursive version of the SM. The calculated gradient approaches the true one if parameter changes are small enough. This can be achieved at small learning rates but it may result in a slow convergence rate.

3.4.4 Parallel SISO model. Backpropagation through time method

In the BPTT method, it is also assumed that model parameters do not change in successive time steps. The Hammerstein model (3.14), unfolded for one step back in time, can be written as

$$\begin{aligned}\hat{y}(n) =& -\hat{a}_1\hat{y}(n-1) - \sum_{m=2}^{na}\hat{a}_m\hat{y}(n-m) + \sum_{m=1}^{nb}\hat{b}_m\hat{f}\big(u(n-m),\mathbf{w}\big)\\ =& -\hat{a}_1\Big[-\sum_{m=1}^{na}\hat{a}_m\hat{y}(n-m-1) + \sum_{m=1}^{nb}\hat{b}_m\hat{f}\big(u(n-m-1),\mathbf{w}\big)\Big]\\ &-\sum_{m=2}^{na}\hat{a}_m\hat{y}(n-m) + \sum_{m=1}^{nb}\hat{b}_m\hat{f}\big(u(n-m),\mathbf{w}\big).\end{aligned} \tag{3.42}$$

The unfolding procedure can be continued until initial conditions of the model are achieved. As we proceed with the unfolding procedure, an unfolded model equation is obtained, which becomes more and more complex and makes gradient calculation more and more difficult. On the other hand, the unfolded model can be represented by a multilayer feedforward neural network, see Fig. 3.7 for an example of an unfolded-in-time Hammerstein model. The completely unfolded neural network corresponding to (3.14) is no longer of the recurrent type and can be trained with an algorithm similar to backpropagation and called backpropagation through time (BPTT) [84]. Nevertheless, the complexity of the unfolded network depends on the number of time steps and both the computational and memory requirements of the BPTT method are time dependent. In practice, to overcome this inconvenience, unfolding in time can be restricted to a fixed number of time steps in a method called truncated backpropagation through time. In this way, it is often possible to obtain quite a good approximation of the gradient, even for models unfolded back in time only for a few time steps. A detailed description of gradient calculation with the truncated BPTT method is given in Appendix 3.1.

3.4.5 Series-parallel MIMO model. Backpropagation method

For the MIMO Hammerstein model, the following cost function is minimized in the sequential mode:

$$J(n) = \frac{1}{2}\sum_{t=1}^{ny}\big(y_t(n) - \hat{y}_t(n)\big)^2. \tag{3.43}$$

The gradient in series-parallel MIMO Hammerstein models can be calculated with a version of the BPS algorithm extended to the multidimensional case. The differentiation of (3.28) w.r.t the weights of the MIMO nonlinear element model gives

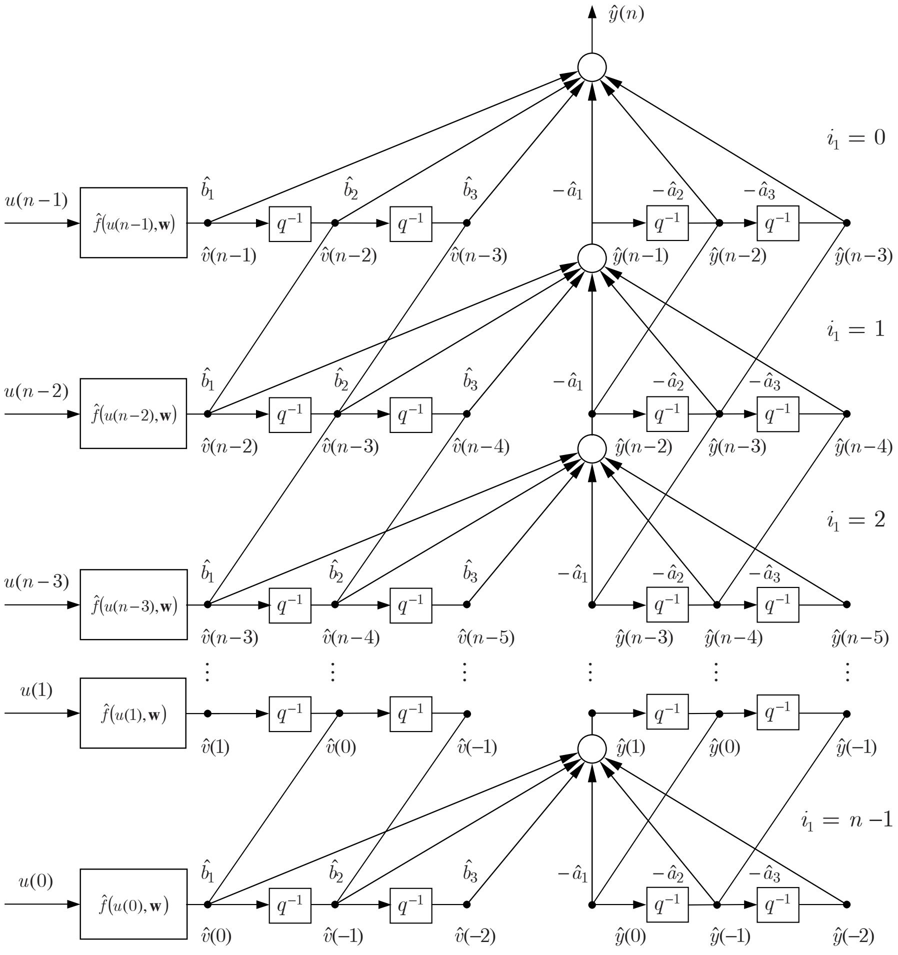
$\hat{y}(n)$
$i_1 = 0$
$i_1 = 1$
$i_1 = 2$
$i_1 = n-1$
$u(n-1)$
$\hat{f}(u(n-1),\mathbf{w})$
$u(n-2)$
$\hat{f}(u(n-2),\mathbf{w})$
$u(n-3)$
$\hat{f}(u(n-3),\mathbf{w})$
$u(1)$
$\hat{f}(u(1),\mathbf{w})$
$u(0)$
$\hat{f}(u(0),\mathbf{w})$
$\hat{b}_1$
$\hat{b}_2$
$\hat{b}_3$
$-\hat{a}_1$
$-\hat{a}_2$
$-\hat{a}_3$
q^{-1}
$\hat{v}(n-1)$ $\hat{v}(n-2)$ $\hat{v}(n-3)$ $\hat{y}(n-1)$ $\hat{y}(n-2)$ $\hat{y}(n-3)$
$\hat{v}(n-2)$ $\hat{v}(n-3)$ $\hat{v}(n-4)$ $\hat{y}(n-2)$ $\hat{y}(n-3)$ $\hat{y}(n-4)$
$\hat{v}(n-3)$ $\hat{v}(n-4)$ $\hat{v}(n-5)$ $\hat{y}(n-3)$ $\hat{y}(n-4)$ $\hat{y}(n-5)$
$\hat{v}(1)$ $\hat{v}(0)$ $\hat{v}(-1)$ $\hat{y}(1)$ $\hat{y}(0)$ $\hat{y}(-1)$
$\hat{v}(0)$ $\hat{v}(-1)$ $\hat{v}(-2)$ $\hat{y}(0)$ $\hat{y}(-1)$ $\hat{y}(-2)$

Fig. 3.7. Parallel Hammerstein model of the third order unfolded back in time

$$\frac{\partial J(n)}{\partial w_{ji}^{(1)}} = -\sum_{t=1}^{ny}\Big(y_t(n) - \hat{y}_t(n)\Big)\frac{\partial \hat{y}_t(n)}{\partial w_{ji}^{(1)}}, \tag{3.44}$$

$$\frac{\partial J(n)}{\partial w_{j0}^{(1)}} = -\sum_{t=1}^{ny}\Big(y_t(n) - \hat{y}_t(n)\Big)\frac{\partial \hat{y}_t(n)}{\partial w_{j0}^{(1)}}, \tag{3.45}$$

$$\frac{\partial J(n)}{\partial w_{lj}^{(2)}} = -\sum_{t=1}^{ny}\Big(y_t(n) - \hat{y}_t(n)\Big)\frac{\partial \hat{y}_t(n)}{\partial w_{lj}^{(2)}}, \tag{3.46}$$

$$\frac{\partial J(n)}{\partial w_{l0}^{(2)}} = -\sum_{t=1}^{ny}\Big(y_t(n) - \hat{y}_t(n)\Big)\frac{\partial \hat{y}_t(n)}{\partial w_{l0}^{(2)}}, \tag{3.47}$$

where $i = 1, \ldots, nu$, $j = 1, \ldots, M$, $l = 1, \ldots, nf$.

The calculation of partial derivatives of the model outputs w.r.t. the weights of the MIMO nonlinear element requires the calculation of partial derivatives of the outputs of the MIMO nonlinear element $\hat{f}_l\big(\mathbf{u}(n),\mathbf{w}_l\big)$ w.r.t. its weights. The differentiation of (3.17) gives

$$\begin{aligned}\frac{\partial f_l\big(\mathbf{u}(n),\mathbf{w}_l\big)}{\partial w_{ji}^{(1)}} &= \frac{\partial f_l\big(\mathbf{u}(n),\mathbf{w}_l\big)}{\partial \varphi\big(x_j(n)\big)}\frac{\partial \varphi\big(x_j(n)\big)}{\partial x_j(n)}\frac{\partial x_j(n)}{\partial w_{ji}^{(1)}} \\ &= w_{lj}^{(2)}\varphi'\big(x_j(n)\big)u_i(n),\end{aligned} \tag{3.48}$$

$$\begin{aligned}\frac{\partial f_l\big(\mathbf{u}(n),\mathbf{w}_l\big)}{\partial w_{j0}^{(1)}} &= \frac{\partial f_l\big(\mathbf{u}(n),\mathbf{w}_l\big)}{\partial \varphi\big(x_j(n)\big)}\frac{\partial \varphi\big(x_j(n)\big)}{\partial x_j(n)}\frac{\partial x_j(n)}{\partial w_{j0}^{(1)}} \\ &= w_{lj}^{(2)}\varphi'\big(x_j(n)\big),\end{aligned} \tag{3.49}$$

$$\frac{\partial f_l\big(\mathbf{u}(n),\mathbf{w}_l\big)}{\partial w_{lj}^{(2)}} = \varphi\big(x_j(n)\big), \tag{3.50}$$

$$\frac{\partial f_l\big(\mathbf{u}(n),\mathbf{w}_l\big)}{\partial w_{l0}^{(2)}} = 1. \tag{3.51}$$

Taking into account (3.28) and (3.48) – (3.51), we have

$$\begin{aligned}\frac{\partial \hat{y}_t(n)}{\partial w_{ji}^{(1)}} &= \sum_{m=1}^{nb}\sum_{m_1=1}^{nf}\hat{b}_{tm_1}^{(m)}\frac{\partial \hat{f}_{m_1}\big(\mathbf{u}(n-m),\mathbf{w}_{m_1}\big)}{\partial w_{ji}^{(1)}} \\ &= \sum_{m=1}^{nb}\sum_{m_1=1}^{nf}\hat{b}_{tm_1}^{(m)}w_{m_1j}^{(2)}\varphi'\big(x_j(n-m)\big)u_i(n-m),\end{aligned} \tag{3.52}$$

$$\begin{aligned}\frac{\partial \hat{y}_t(n)}{\partial w_{j0}^{(1)}} &= \sum_{m=1}^{nb}\sum_{m_1=1}^{nf}\hat{b}_{tm_1}^{(m)}\frac{\partial \hat{f}_{m_1}\big(\mathbf{u}(n-m),\mathbf{w}_{m_1}\big)}{\partial w_{j0}^{(1)}} \\ &= \sum_{m=1}^{nb}\sum_{m_1=1}^{nf}\hat{b}_{tm_1}^{(m)}w_{m_1j}^{(2)}\varphi'\big(x_j(n-m)\big),\end{aligned} \tag{3.53}$$

$$\frac{\partial \hat{y}_t(n)}{\partial w_{lj}^{(2)}} = \sum_{m=1}^{nb}\hat{b}_{tl}^{(m)}\frac{\partial \hat{f}_l\big(\mathbf{u}(n-m),\mathbf{w}_l\big)}{\partial w_{lj}^{(2)}} = \sum_{m=1}^{nb}\hat{b}_{tl}^{(m)}\varphi\big(x_j(n-m)\big), \tag{3.54}$$

$$\frac{\partial \hat{y}_t(n)}{\partial w_{l0}^{(2)}} = \sum_{m=1}^{nb}\hat{b}_{tl}^{(m)}\frac{\partial \hat{f}_l\big(\mathbf{u}(n-m),\mathbf{w}_l\big)}{\partial w_{l0}^{(2)}} = \sum_{m=1}^{nb}\hat{b}_{tl}^{(m)}. \tag{3.55}$$

As only the output $\hat{y}_t(n)$ depends upon the parameters $\hat{a}_{tm_1}^{(k)}$ and $\hat{b}_{tm_1}^{(k)}$, the differentiation of (3.43) yields

$$\frac{\partial J(n)}{\partial \hat{a}_{tm_1}^{(k)}} = -\big(y_t(n) - \hat{y}_t(n)\big)\frac{\partial \hat{y}_t(n)}{\partial \hat{a}_{tm_1}^{(k)}},\ k = 1,\ldots,na, m_1 = 1,\ldots,ny, \tag{3.56}$$

$$\frac{\partial J(n)}{\partial \hat{b}_{tm_1}^{(k)}} = -\big(y_t(n) - \hat{y}_t(n)\big)\frac{\partial \hat{y}_t(n)}{\partial \hat{b}_{tm_1}^{(k)}},\ k = 1,\ldots,nb, m_1 = 1,\ldots,nf. \tag{3.57}$$

Differentiating (3.28), partial derivatives of the model output w.r.t. the parameters of the MIMO linear dynamic model can be calculated as

$$\frac{\partial \hat{y}_t(n)}{\partial \hat{a}_{tm_1}^{(k)}} = -y_{m_1}(n-k), \tag{3.58}$$

$$\frac{\partial \hat{y}_t(n)}{\partial \hat{b}_{tm_1}^{(k)}} = \hat{f}_{m_1}\big(\mathbf{u}(n-k),\mathbf{w}_{m_1}\big). \tag{3.59}$$

3.4.6 Parallel MIMO model. Backpropagation method

Neglecting the dependence of past values of output signals upon the parameters of the parallel model and differentiating (3.27) results in the multidimensional version of the BPP method.
In the BPP method, partial derivatives of the model output w.r.t. the parameters of the nonlinear element model are calculated in the same way as in the BPS method. The only difference is the calculation of partial derivatives of the model output w.r.t. the parameters $\hat{a}_{tm_1}^{(k)}$ of the linear dynamic model, which requires replacing $y_{m_1}(n-k)$ in (3.58) with $\hat{y}_{m_1}(n-k)$. The BPP algorithm has the same computational complexity as the BPS algorithm but it suffers from a low convergence rate as it uses a crude approximation of the gradient.

3.4.7 Parallel MIMO model. Sensitivity method

A much better evaluation of the gradient, at the price of computational complexity, can be obtained with the multidimensional version of the SM. The SM differs from the BPS method in the calculation of partial derivatives of the model output w.r.t. model parameters. To calculate partial derivatives of the model output w.r.t. the parameters of the nonlinear element model, a set of linear difference equations is solved by simulation. From (3.27) and (3.48) – (3.51), it follows that

$$\begin{aligned}\frac{\partial \hat{y}_t(n)}{\partial w_{ji}^{(1)}} = &\sum_{m=1}^{nb}\sum_{m_1=1}^{nf} \hat{b}_{tm_1}^{(m)} w_{m_1 j}^{(2)} \varphi'\big(x_j(n-m)\big) u_i(n-m) \\ &- \sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial w_{ji}^{(1)}},\end{aligned} \tag{3.60}$$

$$\begin{aligned}\frac{\partial \hat{y}_t(n)}{\partial w_{j0}^{(1)}} = &\sum_{m=1}^{nb}\sum_{m_1=1}^{nf} \hat{b}_{tm_1}^{(m)} w_{m_1 j}^{(2)} \varphi'\big(x_j(n-m)\big) \\ &- \sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial w_{j0}^{(1)}},\end{aligned} \tag{3.61}$$

$$\frac{\partial \hat{y}_t(n)}{\partial w_{lj}^{(2)}} = \sum_{m=1}^{nb} \hat{b}_{tl}^{(m)} \varphi\big(x_j(n-m)\big) - \sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial w_{lj}^{(2)}}, \tag{3.62}$$

$$\frac{\partial \hat{y}_t(n)}{\partial w_{l0}^{(2)}} = \sum_{m=1}^{nb} \hat{b}_{tl}^{(m)} - \sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial w_{l0}^{(2)}}, \tag{3.63}$$

where $i = 1, \ldots, nu$, $j = 1, \ldots, M$, $l = 1, \ldots, nf$. Similarly, partial derivatives of the model output w.r.t. the parameters of the linear dynamic model are obtained from another set of linear difference equations:

$$\frac{\partial \hat{y}_t(n)}{\partial \hat{a}_{rp}^{(k)}} = -\,\delta_{tr} y_p(n-k) - \sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial \hat{a}_{rp}^{(k)}}, \tag{3.64}$$

$$\delta_{tr} = \begin{cases} 1 & \text{for } t = r \\ 0 & \text{for } t \neq r \end{cases}, \tag{3.65}$$

where $k = 1, \ldots, na$, $r = 1, \ldots, ny$, $p = 1, \ldots, ny$,

$$\frac{\partial \hat{y}_t(n)}{\partial \hat{b}_{rp}^{(k)}} = \delta_{tr} \hat{f}_p\big(\mathbf{u}(i-k), \mathbf{w}_p\big) - \sum_{m=1}^{na}\sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial \hat{b}_{rp}^{(k)}}, \tag{3.66}$$

where $k = 1, \ldots, nb$, $r = 1, \ldots, ny$, $p = 1, \ldots, nf$.

3.4.8 Parallel MIMO model. Backpropagation through time method

The details of gradient calculation with truncated backpropagation through time method in MIMO Hammerstein models are given in Appendix 3.2.

3.4.9 Accuracy of gradient calculation with truncated BPTT

As in the case of neural network Wiener models, we will assume that the input $u(n)$ is a sequence of zero-mean i.i.d. random variables.

Theorem 3.1. Define the computation error

$$\Delta \hat{y}_{\hat{a}_k}(n) = \frac{\partial \hat{y}(n)}{\partial \hat{a}_k} - \frac{\partial^+ \hat{y}(n)}{\partial \hat{a}_k},$$

where $\partial \hat{y}(n)/\partial \hat{a}_k$, $k = 1, \ldots, na$, denote partial derivatives calculated with the BPTT method, i.e. unfolding the model (3.14) $n-1$ times back in time, and $\partial^+ \hat{y}(n)/\partial \hat{a}_k$ denote partial derivatives calculated with the truncated BPTT method unfolding the model (3.14) K times back in time. Assume that

(A1) The linear dynamic model $\hat{B}(q^{-1})/\hat{A}(q^{-1})$ is asymptotically stable;
(A2) $\hat{y}(n) = 0$, $n = 0, \ldots, -na+1$; $\hat{f}\big(u(n),\mathbf{w}\big) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial \hat{y}(n)/\partial \hat{a}_k = 0$, $n = 0, \ldots, -na+1$;
(A4) The input $u(n)$, $n = 0, 1, \ldots$, is a sequence of zero-mean i.i.d. random variables of finite moments,

$$\mathrm{E}[u(n)] = 0,$$
$$\mathrm{E}[u(n)u(m)] = \begin{cases} \sigma^2 & \text{for } n = m \\ 0 & \text{for } n \neq m \end{cases}.$$

Then

$$\operatorname{var}\big(\Delta \hat{y}_{\hat{a}_k}(n)\big) = \sigma_f^2 \sum_{i_1=K+1}^{n-k} \left(\sum_{i_2=K+1}^{i_1} h_1(i_2) h_2(i_1 - i_2) \right)^2, \; k = 1, \ldots, na, \tag{3.67}$$

for $n > K + k$ and 0 otherwise, where $\sigma_f^2 = \operatorname{var}\big[\hat{f}\big(u(n),\mathbf{w}\big)\big]$, and $h_1(n)$ is the impulse response function of the system

$$H_1(q^{-1}) = \frac{1}{\hat{A}(q^{-1})}, \tag{3.68}$$

and $h_2(n)$ is the impulse response function of the system

$$H_2(q^{-1}) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}. \tag{3.69}$$

Proof: The proof is shown in Appendix 3.3.

Theorem 3.2. Define the computation error

$$\Delta \hat{y}_{\hat{b}_k}(n) = \frac{\partial \hat{y}(n)}{\partial \hat{b}_k} - \frac{\partial^+ \hat{y}(n)}{\partial \hat{b}_k},$$

where $\partial\hat{y}(n)/\partial\hat{b}_k$, $k = 1, \ldots, nb$, denote partial derivatives calculated with the BPTT method, i.e. unfolding the model (2.14) $n-1$ times back in time, and $\partial^+\hat{y}(n)/\partial\hat{b}_k$ denote partial derivatives calculated with the truncated BPTT method unfolding the model (2.14) K times back in time. Assume that

(A1) The linear dynamic model $\hat{B}(q^{-1})/\hat{A}(q^{-1})$ is asymptotically stable;
(A2) $\hat{y}(n) = 0$, $n = 0, \ldots, -na+1$; $\hat{f}\big(u(n),\mathbf{w}\big) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial\hat{y}(n)/\partial\hat{b}_k = 0$, $n = 0, \ldots, -na+1$;
(A4) The input $u(n)$, $n = 0, 1, \ldots$, is a sequence of zero-mean i.i.d. random variables of finite moments,

$$\mathrm{E}[u(n)] = 0,$$

$$\mathrm{E}[u(n)u(m)] = \begin{cases} \sigma^2 & \text{for } n = m \\ 0 & \text{for } n \neq m \end{cases}.$$

Then

$$\operatorname{var}\big(\varDelta\hat{y}_{\hat{b}_k}(n)\big) = \sigma_f^2 \sum_{i_1=K+1}^{n-k} h_1^2(i_1), \;\; k = 1, \ldots, nb, \tag{3.70}$$

for $n > K + k$ and 0 otherwise, where $\sigma_f^2 = \operatorname{var}\big[\hat{f}\big(u(n),\mathbf{w}\big)\big]$, and $h_1(n)$ is the impulse response function of the system

$$H_1(q^{-1}) = \frac{1}{\hat{A}(q^{-1})}. \tag{3.71}$$

Proof: The proof is shown in Appendix 3.4.

Theorem 3.3. Define the computation error

$$\varDelta\hat{y}_{w_c}(n) = \frac{\partial\hat{y}(n)}{\partial w_c} - \frac{\partial^+\hat{y}(n)}{\partial w_c},$$

where $\partial\hat{y}(n)/\partial w_c$ denote partial derivatives calculated with the BPTT method, i.e. unfolding the model (3.14) $n-1$ times back in time, and $\partial^+\hat{y}(n)/\partial w_c$ denote partial derivatives calculated with the truncated BPTT method unfolding the model (3.14) K times back in time and w_c is the cth element of $\big[w_{10}^{(1)} \ldots w_{M1}^{(1)} \ldots w_{11}^{(2)} \ldots w_{1M}^{(2)}\big]$, $c = 1, \ldots, 3M$.

Assume that

(A1) The linear dynamic model $\hat{B}(q^{-1})/\hat{A}(q^{-1})$ is asymptotically stable;
(A2) $\hat{y}(n) = 0$, $n = 0, \ldots, -na+1$; $\hat{f}\big(u(n),\mathbf{w}\big) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial\hat{y}(n)/\partial w_c = 0$, $n = 0, \ldots, -na+1$; $\partial\hat{f}\big(u(n),\mathbf{w}\big)/\partial w_c = 0$, $n = -1, \ldots, -nb$;
(A4) The input $u(n)$, $n = 0, 1, \ldots$, is a sequence of zero-mean i.i.d. random variables of finite moments,

$$\mathrm{E}[u(n)] = 0,$$
$$\mathrm{E}[u(n)u(m)] = \begin{cases} \sigma^2 & \text{for } n = m \\ 0 & \text{for } n \neq m \end{cases}.$$

Then

$$\operatorname{var}\bigl(\Delta\hat{y}_{w_c}(n)\bigr) = \sigma^2_{w_c} \sum_{i_1=K+2}^{n} h_2^2(i_1), \ c = 1, \ldots, 3M, \tag{3.72}$$

for $n > K + k$ and 0 otherwise, where $\sigma^2_{w_c} = \operatorname{var}\bigl[\partial\hat{f}\bigl(u(n),\mathbf{w}\bigr)/\partial w_c\bigr]$ and $h_2(n)$ is the impulse response function of the system

$$H_2(q^{-1}) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}. \tag{3.73}$$

Proof: The proof is shown in Appendix 3.5.

Theorem 3.4. Define the computation error

$$\Delta\hat{y}_{w_{10}^{(2)}}(n) = \frac{\partial\hat{y}(n)}{\partial w_{10}^{(2)}} - \frac{\partial^+\hat{y}(n)}{\partial w_{10}^{(2)}},$$

where $\partial\hat{y}(n)/\partial w_{10}^{(2)}$ is a partial derivative calculated with the BPTT method, i.e. unfolding the model (3.14) $n - 1$ times back in time, and $\partial^+\hat{y}(n)/\partial w_{10}^{(2)}$ is a partial derivative calculated with the truncated BPTT method unfolding the model (3.14) K times back in time.

Assume that

(A1) The linear dynamic model $\hat{B}(q^{-1})/\hat{A}(q^{-1})$ is asymptotically stable;
(A2) $\hat{y}(n) = 0$, $n = 0, \ldots, -na + 1$; $\hat{f}\bigl(u(n),\mathbf{w}\bigr) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial\hat{y}(n)/\partial w_{10}^{(2)} = 0$, $n = 0, \ldots, -na + 1$; $\partial\hat{f}\bigl(u(n),\mathbf{w}\bigr)/\partial w_{10}^{(2)} = 0$, $n = -1, \ldots, -nb$;

Then

$$\Delta\hat{y}_{w_{10}^{(2)}}(n) = \sum_{i_1=K+2}^{n} h_2(i_1), \tag{3.74}$$

for $n > K + k$ and 0 otherwise, where $h_2(n)$ is the impulse response function of the system

$$H_2(q^{-1}) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}. \tag{3.75}$$

Proof: The proof is shown in Appendix 3.6.

Remark 3.1. The lowest gradient calculation accuracy and the highest values of the variances (3.67), (3.70), (3.72) and the error (3.74) are achieved with the BPP method for which $K = 0$.

Remark 3.2. From (3.67) and (3.70), it follows that the variances of computation errors do not depend on k for $n \to \infty$, i.e,

$$\lim_{n\to\infty} \operatorname{var}\big(\Delta\hat{y}_{\hat{a}_k}(n)\big) = \sigma_f^2 \sum_{i_1=K+1}^{\infty} \left(\sum_{i_2=K+1}^{i_1} h_1(i_2) h_2(i_1-i_2) \right)^2, \tag{3.76}$$

$$\lim_{n\to\infty} \operatorname{var}\big(\Delta\hat{y}_{\hat{b}_k}(n)\big) = \sigma_f^2 \sum_{i_1=K+1}^{\infty} h_1^2(i_1). \tag{3.77}$$

Theorems 3.1 – 3.4 provide a useful tool for the determination of the number of unfolded steps K. From Theorems 3.1 and 3.2 it follows that, for a fixed value of K, calculation accuracy of partial derivatives of the parallel neural network Hammerstein model output w.r.t. the parameters $\hat{a}_k$ and $\hat{b}_k$ with the truncated BPTT method depends on the number of discrete time steps necessary for the impulse response $h_1(n)$ to decrease to negligible small values. Analogously, calculation accuracy of partial derivatives of the parallel neural network Hammerstein model output w.r.t. the parameters w_c, $c = 1, \ldots, 3M$, and $w_{10}^{(2)}$ depends on the number of discrete time steps necessary for the impulse response $h_2(n)$ to decrease to negligible small values. Note that these impulse responses can differ significantly from the impulse responses of the system $1/A(q^{-1})$ or $B(q^{-1})/A(q^{-1})$, particularly at the beginning of the learning process. Therefore, both $h_1(n)$ and $h_2(n)$ can be traced during the training of the neural network model and K can be changed adaptively to meet the assumed degrees of gradient calculation accuracy $\xi_{a_k}(n)$, $\xi_{b_k}(n)$ and $\xi_{w_c}(n)$ defined as the ratio of the variances (3.67), (3.70) and (3.72) to the variances of the corresponding partial derivatives:

$$\xi_{a_k}(n) = \frac{\operatorname{var}\big(\Delta\hat{y}_{\hat{a}_k}(n)\big)}{\operatorname{var}\left(\dfrac{\partial \hat{y}(n)}{\partial \hat{a}_k}\right)} = \frac{\displaystyle\sum_{i_1=K+1}^{n-k} \left(\sum_{i_2=K+1}^{i_1} h_1(i_2) h_2(i_1-i_2) \right)^2}{\displaystyle\sum_{i_1=0}^{n-k} \left(\sum_{i_2=0}^{i_1} h_1(i_2) h_2(i_1-i_2) \right)^2}, \tag{3.78}$$

$$\xi_{b_k}(n) = \frac{\operatorname{var}\big(\Delta\hat{y}_{\hat{b}_k}(n)\big)}{\operatorname{var}\left(\dfrac{\partial \hat{y}(n)}{\partial \hat{b}_k}\right)} = \frac{\displaystyle\sum_{i_1=K+1}^{n-k} h_1^2(i_1)}{\displaystyle\sum_{i_1=0}^{n-k} h_1^2(i_1)}, \tag{3.79}$$

$$\xi_{w_c}(n) = \frac{\operatorname{var}\big(\Delta\hat{y}_{w_c}(n)\big)}{\operatorname{var}\left(\dfrac{\partial\hat{y}(n)}{\partial w_c}\right)} = \frac{\sum\limits_{i_1=K+2}^{n} h_2^2(i_1)}{\sum\limits_{i_1=0}^{n} h_2^2(i_1)}, \tag{3.80}$$

where $c = 1, \ldots, 3M$. An initial value of K can be determined based on initial conditions or a priori knowledge of system dynamics.
As in the case of Wiener systems, the required value of K for Hammerstein systems depends strongly on the system sampling rate. If the sampling rate increases for a given system, the numbers of discrete time steps, necessary for the impulse responses $h_1(n)$ and $h_2(n)$ to decrease to negligible small values, increase and a higher value of K is necessary to achieve the same accuracy of gradient calculation.

3.4.10 Gradient calculation in the sequential mode

The derivation of gradient calculation algorithms has been made under the assumption that the Hammerstein model is time invariant. In the sequential mode, this assumption is not valid and only approximated values of the gradient are obtained for parallel models. Hence, to obtain a good approximation of the gradient, the learning rate should be small enough to achieve negligible small changes of model parameters. Applying the BPTT method, the actual values of model parameters can be used in the unfolding procedure. Although the gradient calculated in this way is still approximate, such an approach may increase the convergence rate of the learning process even if model unfolding is restricted to only a few time steps. As in the case of neural network Wiener models, the BPPT method is able to provide exact values of the gradient if linear dynamic model outputs are unfolded back in time according to the following rule:

$$\begin{aligned}\hat{y}(n-na-i_1) = \frac{1}{\hat{a}_{na}}\bigg(&-\hat{y}(n-i_1) - \sum_{m=1}^{na-1}\hat{a}_m\hat{y}(n-m-i_1)\\ &+\sum_{m=1}^{nb}\hat{b}_m\hat{f}\big(u(n-m-i_1),\mathbf{w}\big)\bigg),\end{aligned} \tag{3.81}$$

where $i_1 = 1, \ldots, n-1$. The formula (3.81) can be employed provided that $\hat{a}_{na} \neq 0$. In practice, if $\hat{a}_{na} = 0$ or $\hat{a}_{na} \simeq 0$, then $\hat{y}(n-na-i_1)$ does not influence or does not influence significantly $\hat{y}(n-i_1)$, and this unfolding step can be omitted.
Alternatively, it is possible to unfold the outputs of the nonlinear element model instead the outputs of the linear dynamic model:

$$\hat{f}\big(u(n-nb-i_1),\mathbf{w}\big) = \frac{1}{\hat{b}_{nb}}\Bigg(\hat{y}(n-i_1) + \sum_{m=1}^{na} \hat{a}_m \hat{y}(n-m-i_1) - \sum_{m=1}^{nb-1} \hat{b}_m \hat{f}\big(u(n-m-i_1),\mathbf{w}\big)\Bigg). \tag{3.82}$$

Nevertheless, applying (3.82) instead of (3.81) is more complex as it requires not only that $\hat{b}_{nb} \neq 0$ but also needs calculating the model inputs $u(u(n–nb–i_1)$ that correspond to the nonlinear element outputs $\hat{f}\big(u(n-m-i_1),\mathbf{w}\big)$.

3.4.11 Computational complexity

Computational complexity of algorithms, given in Table 3.1, is evaluated based on the same assumptions as those given in Section 2.4.11.

Table 3.1. Computational complexity of learning algorithms

	Computational complexity
BPS or BPP	$3na + 5nb + 15nbM + 3M + 1$
SM	$5na + 5nb + 15nbM + 6naM + 3M + 1$
BPTT	$(13nbM + 5nb + 3na)K + 3na + 5nb + 15nbM + 3M + 1$

3.5 Simulation example

A Hammerstein system composed of a linear dynamic system

$$G(s) = \frac{1}{6s^2 + 5s + 1} \tag{3.83}$$

and the nonlinear element (Fig. 3.8):

$$f\big(u(n)\big) = \sin\big(0.6\pi u(n)\big) + 0.1\cos\big(1.5\pi u(n)\big) \tag{3.84}$$

was used in the simulation study. The system (3.83) was converted to discrete time, assuming the zero order hold on the input and the sampling interval 1s, resulting in the following difference equation:

$$s(n) = 1.3231s(n–1) - 0.4346s(n-2) + 0.0635u(n–1) + 0.0481u(n-2). \tag{3.85}$$

The system (3.83) was driven by a pseudo-random sequence uniformly distributed in $(-1, 1)$. The nonlinear element model contained 25 nonlinear nodes of the hyperbolic tangent activation function. Both the series-parallel and parallel Hammerstein models were trained recursively using the steepest descent method to update the weights, and BPS, BPP, SM, and truncated BPTT algorithms to calculate the gradient. To compare the different algorithms, training was carried out based on the same sequence of 20000 input-output patterns with the learning rate of 0.2 for nonlinear nodes and 0.02 for linear ones. The overall number of training steps was 60000 as the training sequence was used three times. All calculations were performed at the same initial values of neural network parameters. The mean square error of moving averages defined as

$$J_I(n) = \begin{cases} \dfrac{1}{n}\sum\limits_{j=1}^{n}\big(\hat{y}(j) - y(j)\big)^2 & \text{for } n \leqslant I \\ \dfrac{1}{I}\sum\limits_{j=n-I+1}^{n}\big(\hat{y}(j) - y(j)\big)^2 & \text{for } n > I \end{cases} \tag{3.86}$$

is used to compare the convergence rate of the algorithms. The indices $P(n)$ and $F(n)$ are used to compare the identification accuracy of the linear dynamic system and the nonlinear function $f(\cdot)$ and, respectively,

$$P(n) = \frac{1}{4}\sum_{j=1}^{2}\left[(\hat{a}_j - a_j)^2 + (\hat{b}_j - b_j)^2\right], \tag{3.87}$$

$$F(n) = \frac{1}{100}\sum_{j=1}^{100}\left[\hat{f}\big(u(j)\big) - f\big(u(j), \mathbf{w}\big)\right]^2, \tag{3.88}$$

where $u(j)$ is a testing sequence consisting of 100 linearly equally spaced values between -1 and 1. The results of model training for both a noise-free Hammerstein system and a system disturbed by the additive output Gaussian noise $\mathcal{N}(0, \sigma_e)$ are given in Tables 3.2–3.4.

Noise-free case. The identification results are given in Table 3.2 and illustrated in Figs 3.8–3.14. The BPS example uses the true gradient. This results in a high convergence rate and smooth decreasing of $J_{2000}(n)$ (Fig. 3.9). As can be expected, the lowest convergence rate can be observed in the BPP example. This can be explained by the low accuracy of gradient approximation with the backpropagation method.
A better gradient approximation results in a higher convergence rate that can be seen in both the SM and BPTT examples. Maximum accuracy of both the linear dynamic model and the nonlinear element model can be observed in the BPTT example at $K = 4$. The results of nonlinear element identification obtained in the BPTT examples at $K = 3, \ldots, 11$ are more accurate in comparison with those obtained using the SM algorithm. Further unfolding of the model does not improve identification accuracy and even some deterioration can be observed. Comparing the results of linear dynamic system identification, it can be noticed that BPTT outperforms the SM when $K = 2, \ldots, 7$.

Noise case I. The identification results for a high signal to noise ratio $SNR = 18.46$, $SNR = \sqrt{\text{var}(y(n) - \varepsilon(n))/\text{var}(\varepsilon(n))}$ are given in Table 3.3. The highest accuracy of the nonlinear element model was achieved in the BPTT example at $K = 15$, and the highest accuracy of the linear dynamic model was achieved in the BPTT example at $K = 2$. The overall identification accuracy measured by the indices $J_{60000}(60000)$ increases with an increase in the number of unfolded time steps K. The index $J_{2000}(60000)$ has its minimum at $K = 4$.

Noise case II. The identification results for a low signal to noise ratio $SNR = 3.83$ are given in Table 3.4. The model was trained using the time-dependent learning rate

$$\eta(n) = \frac{0.1}{\sqrt[4]{n}}. \tag{3.89}$$

The highest accuracy of both the nonlinear element model and the linear dynamic model was achieved using the BPTT method, at $K = 4$ and $K = 3$, respectively.

In all three cases, the BPTT algorithm provides the most accurate results. This can be explained by the time-varying nature of the model used in the SM algorithm. In other words, to obtain the exact value of the gradient, model parameters should be time invariant. That is the case in the batch mode but

Table 3.2. Comparison of estimation accuracy ($\sigma_e = 0$)

Algorithm	$J_{60000}(60000)$	$J_{2000}(60000)$	$F(60000)$	$P(60000)$
BPS	3.76×10^{-4}	2.86×10^{-4}	8.22×10^{-4}	2.00×10^{-3}
BPP	6.81×10^{-3}	4.89×10^{-2}	1.02×10^{-1}	4.18×10^{-3}
SM	1.15×10^{-3}	5.85×10^{-4}	3.29×10^{-5}	3.30×10^{-6}
BPTT, $K = 1$	1.51×10^{-3}	1.09×10^{-3}	1.19×10^{-4}	1.85×10^{-5}
BPTT, $K = 2$	1.09×10^{-3}	5.41×10^{-4}	4.18×10^{-5}	2.24×10^{-6}
BPTT, $K = 3$	9.61×10^{-4}	4.24×10^{-4}	2.48×10^{-5}	9.55×10^{-7}
BPTT, $K = 4$	9.31×10^{-4}	4.04×10^{-4}	2.14×10^{-5}	8.27×10^{-7}
BPTT, $K = 5$	9.31×10^{-4}	4.12×10^{-4}	2.20×10^{-5}	9.96×10^{-7}
BPTT, $K = 6$	9.44×10^{-4}	4.37×10^{-4}	2.18×10^{-5}	1.71×10^{-6}
BPTT, $K = 7$	9.63×10^{-4}	4.64×10^{-4}	2.31×10^{-5}	2.64×10^{-6}
BPTT, $K = 8$	9.82×10^{-4}	4.91×10^{-4}	2.52×10^{-5}	3.59×10^{-6}
BPTT, $K = 9$	9.99×10^{-4}	5.14×10^{-4}	2.73×10^{-5}	4.11×10^{-6}
BPTT, $K = 10$	1.01×10^{-3}	5.32×10^{-4}	2.99×10^{-5}	4.29×10^{-6}
BPTT, $K = 11$	1.03×10^{-3}	5.42×10^{-4}	3.25×10^{-5}	4.17×10^{-6}
BPTT, $K = 12$	1.03×10^{-3}	5.52×10^{-4}	3.44×10^{-5}	4.10×10^{-6}
BPTT, $K = 13$	1.04×10^{-3}	5.61×10^{-4}	3.56×10^{-5}	4.05×10^{-6}
BPTT, $K = 14$	1.05×10^{-3}	5.65×10^{-4}	3.63×10^{-5}	3.99×10^{-6}
BPTT, $K = 15$	1.05×10^{-3}	5.69×10^{-4}	3.68×10^{-5}	3.99×10^{-6}

Table 3.3. Comparison of estimation accuracy ($\sigma_e = 0.012$, $SNR = 18.46$)

Algorithm	$J_{60000}(60000)$	$J_{2000}(60000)$	$F(60000)$	$P(60000)$
BPS	7.28×10^{-4}	5.99×10^{-4}	9.17×10^{-4}	6.70×10^{-3}
BPP	7.44×10^{-3}	4.90×10^{-3}	1.09×10^{-1}	3.82×10^{-3}
SM	1.40×10^{-3}	7.83×10^{-4}	1.20×10^{-4}	6.43×10^{-5}
BPTT, $K = 1$	1.83×10^{-3}	1.20×10^{-3}	1.68×10^{-4}	5.39×10^{-5}
BPTT, $K = 2$	1.27×10^{-3}	6.95×10^{-4}	1.95×10^{-4}	1.33×10^{-5}
BPTT, $K = 3$	1.14×10^{-3}	5.87×10^{-4}	1.98×10^{-4}	1.58×10^{-5}
BPTT, $K = 4$	1.12×10^{-3}	5.68×10^{-4}	1.35×10^{-4}	1.66×10^{-5}
BPTT, $K = 5$	1.14×10^{-3}	5.84×10^{-4}	1.33×10^{-4}	2.75×10^{-5}
BPTT, $K = 6$	1.16×10^{-3}	6.14×10^{-4}	2.04×10^{-4}	4.24×10^{-5}
BPTT, $K = 7$	1.18×10^{-3}	6.46×10^{-4}	1.59×10^{-4}	6.75×10^{-5}
BPTT, $K = 8$	1.22×10^{-3}	6.76×10^{-4}	1.65×10^{-4}	8.56×10^{-5}
BPTT, $K = 9$	1.24×10^{-3}	7.01×10^{-4}	1.59×10^{-4}	8.81×10^{-5}
BPTT, $K = 10$	1.26×10^{-3}	7.23×10^{-4}	1.27×10^{-4}	9.82×10^{-5}
BPTT, $K = 11$	1.27×10^{-3}	7.33×10^{-4}	1.19×10^{-4}	9.68×10^{-5}
BPTT, $K = 12$	1.29×10^{-3}	7.45×10^{-4}	1.08×10^{-4}	9.85×10^{-5}
BPTT, $K = 13$	1.32×10^{-3}	7.55×10^{-4}	1.06×10^{-4}	9.76×10^{-5}
BPTT, $K = 14$	1.35×10^{-3}	7.64×10^{-4}	1.05×10^{-4}	9.82×10^{-5}
BPTT, $K = 15$	1.37×10^{-3}	7.71×10^{-4}	1.04×10^{-4}	9.90×10^{-5}

Table 3.4. Comparison of the estimation accuracy ($\sigma_e = 0.06$, $SNR = 3.83$)

Algorithm	$J_{60000}(60000)$	$J_{2000}(60000)$	$F(60000)$	$P(60000)$
BPS	6.96×10^{-3}	6.72×10^{-3}	2.55×10^{-3}	1.76×10^{-1}
BPP	1.60×10^{-2}	1.32×10^{-2}	2.16×10^{-2}	5.54×10^{-2}
SM	7.97×10^{-3}	5.72×10^{-3}	4.46×10^{-4}	1.75×10^{-2}
BPTT, $K = 1$	6.42×10^{-3}	6.18×10^{-3}	9.35×10^{-3}	1.15×10^{-2}
BPTT, $K = 2$	5.22×10^{-3}	5.21×10^{-3}	1.32×10^{-3}	6.07×10^{-3}
BPTT, $K = 3$	5.04×10^{-3}	5.04×10^{-3}	4.97×10^{-4}	5.76×10^{-3}
BPTT, $K = 4$	5.02×10^{-3}	4.99×10^{-3}	4.40×10^{-4}	5.80×10^{-3}
BPTT, $K = 5$	5.06×10^{-3}	5.02×10^{-3}	4.84×10^{-4}	7.06×10^{-3}
BPTT, $K = 6$	5.09×10^{-3}	5.03×10^{-3}	5.62×10^{-4}	7.47×10^{-3}
BPTT, $K = 7$	5.12×10^{-3}	5.05×10^{-3}	5.25×10^{-4}	7.80×10^{-3}
BPTT, $K = 8$	5.15×10^{-3}	5.08×10^{-3}	4.48×10^{-4}	8.35×10^{-3}
BPTT, $K = 9$	5.17×10^{-3}	5.11×10^{-3}	4.43×10^{-4}	8.56×10^{-3}
BPTT, $K = 10$	5.19×10^{-3}	5.13×10^{-3}	3.89×10^{-4}	8.91×10^{-3}
BPTT, $K = 11$	5.20×10^{-3}	5.15×10^{-3}	3.80×10^{-4}	9.01×10^{-3}
BPTT, $K = 12$	5.22×10^{-3}	5.17×10^{-3}	3.66×10^{-4}	9.18×10^{-3}
BPTT, $K = 13$	5.23×10^{-3}	5.18×10^{-3}	3.62×10^{-4}	9.24×10^{-3}
BPTT, $K = 14$	5.23×10^{-3}	5.18×10^{-3}	3.55×10^{-4}	9.19×10^{-3}
BPTT, $K = 15$	5.24×10^{-3}	5.19×10^{-3}	3.52×10^{-4}	9.26×10^{-3}

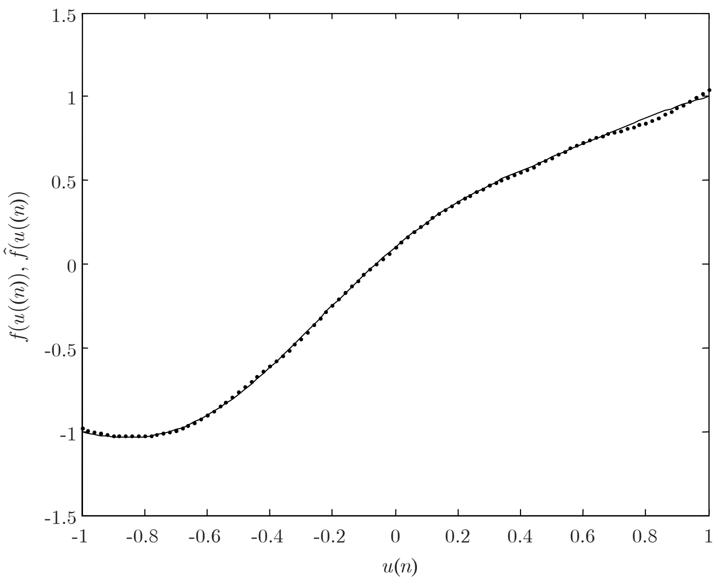

Fig. 3.8. BPTT results, $K = 1$. True (solid line) and estimated (dotted line) nonlinear functions

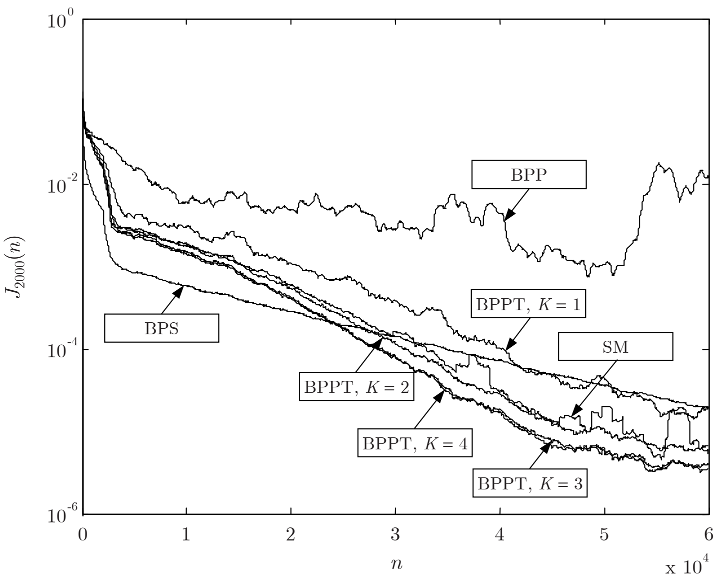

Fig. 3.9. Comparison of convergence rates of different learning algorithms

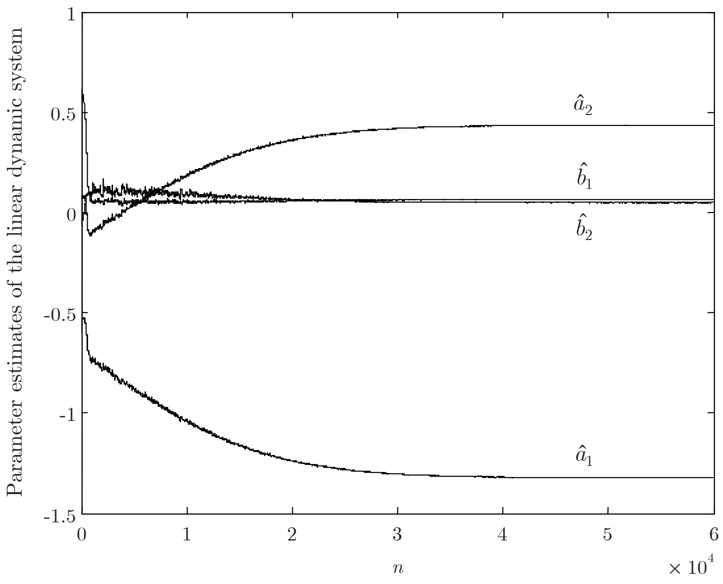

Fig. 3.10. SM results. Evolution of the parameters of the linear dynamic model

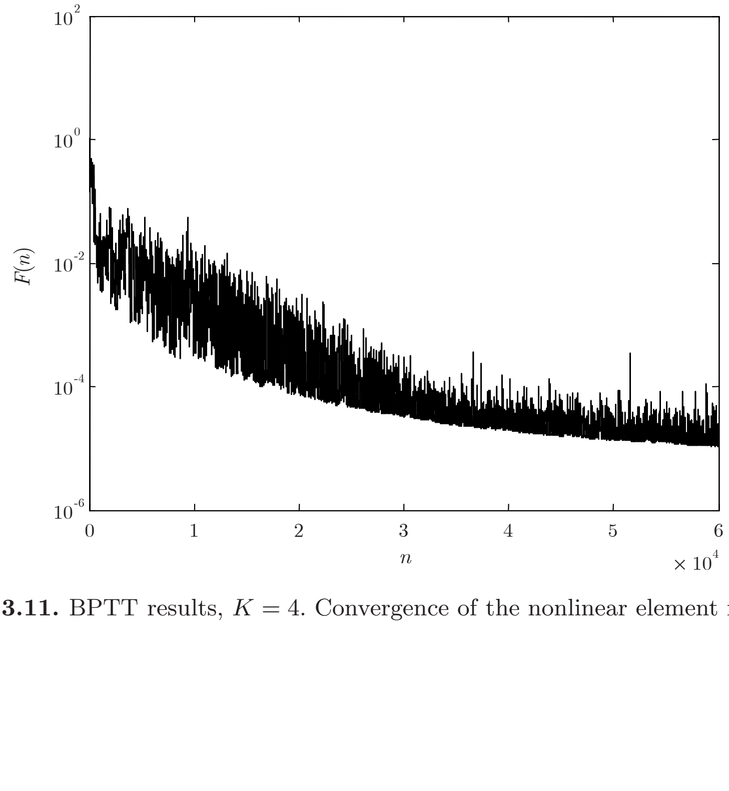

Fig. 3.11. BPTT results, $K = 4$. Convergence of the nonlinear element model

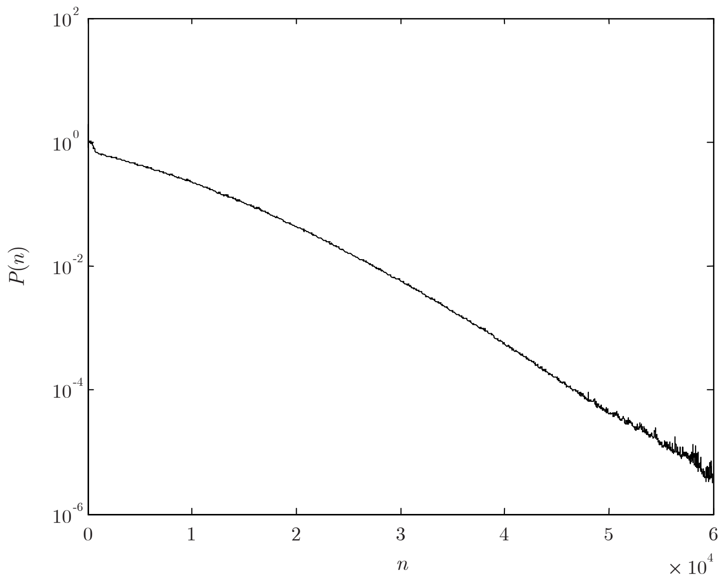

Fig. 3.12. BPTT results, $K = 4$. Convergence of the linear dynamic model

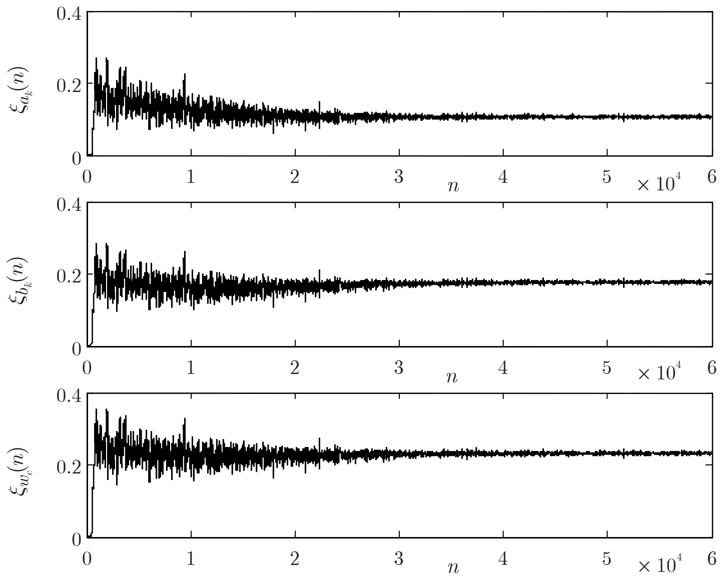

Fig. 3.13. Evolution of gradient calculation accuracy degrees, (BPTT, $K = 4$)

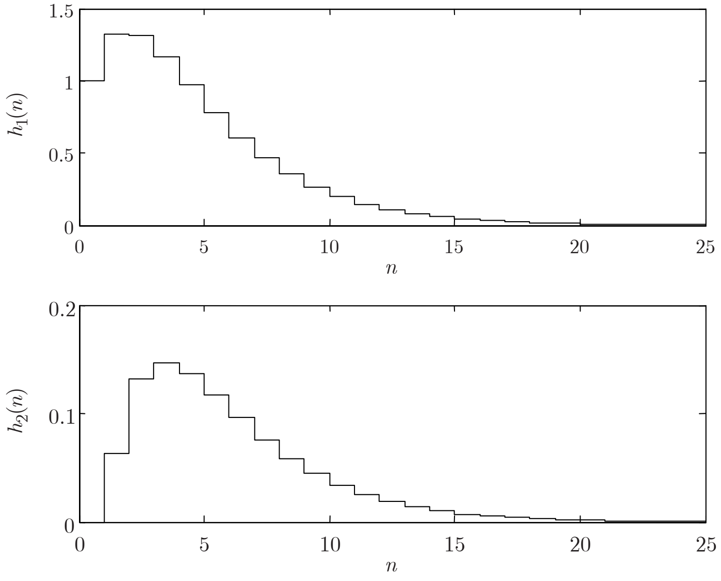

Fig. 3.14. Impulse responses of sensitivity models and the linear dynamic model (BPTT, $K = 4$)

not in the sequential mode. Clearly, the calculated gradient approaches its true value if the learning rate η is small enough and changes of model parameters in subsequent time steps are negligible. Note that, in the sequential mode, the exact value of the gradient can be obtained only with the BPTT method. This, however, requires not only using the actual values of linear dynamic model parameters but also the calculation of corrected values of the past model outputs $\hat{y}(n - i_1)$, $i_1 = 1, \ldots, n - 1$, for a time-invariant Hammerstein model according to the formula (3.81).

3.6 Combined steepest descent and least squares learning algorithms

The parameters of the linear part of the series-parallel Hammerstein model can be computed with the RLS algorithm. In this case, the overall learning algorithm combines the BPS learning algorithm for the parameters of the nonlinear part $w_{j0}^{(1)}$, $w_{j1}^{(1)}$, $w_{1j}^{(2)}$ and $w_{10}^{(2)}$, $j = 1, \ldots, M$, with the RLS algorithm for $\hat{a}_1, \ldots, \hat{a}_{na}$, $\hat{b}_1, \ldots, \hat{b}_{nb}$.
Let $\hat{\boldsymbol{\theta}}(n) = [\hat{a}_1, \ldots, \hat{a}_{na}, \hat{b}_1, \ldots, \hat{b}_{nb}]^T$ denote the parameter vector of the linear dynamic model at the time n. The vector $\hat{\boldsymbol{\theta}}(n)$ can be computed on-line using the RLS algorithm as follows:

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)e(n), \tag{3.90}$$

$$\mathbf{K}(n) = \frac{\mathbf{P}(n-1)\mathbf{x}(n)}{1 + \mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{x}(n)}, \tag{3.91}$$

$$\mathbf{P}(n) = \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)}{1 + \mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{x}(n)}, \tag{3.92}$$

$$e(n) = y(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}(n-1), \tag{3.93}$$

where $\mathbf{P}(n) \in \mathbb{R}^{(na+nb)\times(na+nb)}$ is a symmetrical matrix, $e(n)$ is the one-step ahead prediction error of the system output, and $\mathbf{x}(n)$ is the regression vector:

$$\mathbf{x}(n) = \big[-y(n-1)\ldots - y(n-na)\ \hat{f}\big(u(n-1),\mathbf{w}\big)\ldots \hat{f}\big(u(n-nb),\mathbf{w}\big)\big]^T. \tag{3.94}$$

The output $\hat{y}(n)$ of the parallel Hammerstein model is a nonlinear function of the parameters $\hat{a}_1, \ldots, \hat{a}_{na}, \hat{b}_1, \ldots, \hat{b}_{nb}$ as not only the actual output $\hat{y}(n)$ but also the past outputs $\hat{y}(n-1), \ldots, \hat{y}(n-m)$ depend on these parameters. In spite of this, the same recursive scheme (3.90) – (3.93) can be also applied to the parallel model with the vector $\mathbf{x}(n)$ defined as

$$\mathbf{x}(n) = \big[-\hat{y}(n-1)\ldots - \hat{y}(n-na)\ \hat{f}\big(u(n-1),\mathbf{w}\big)\ldots \hat{f}\big(u(n-nb),\mathbf{w}\big)\big]^T. \tag{3.95}$$

Such an approach is known as recursive pseudolinear regression (RPLR) for the output error model [112]. The combined RPLR and BPP algorithm was used by Al-Duwaish [3]. It is also possible to combine RPLR and the SM or BPTT algorithms. This may result in a higher convergence rate as both the SM and BPTT algorithms evaluate the gradient more accurately.
Note that both (3.94) and (3.95) define the gradient of the model output w.r.t. $\hat{\boldsymbol{\theta}}(n)$ computed with the BPS and BPP methods, respectively. For the parallel model, the gradient calculation can be performed more accurately with the SM or the BPTT algorithm. Taking into account (3.39) and (3.40), for the SM, we have

$$\begin{aligned}\mathbf{x}(n) = \Bigg[&-\hat{y}(n-1) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{a}_1} \ldots - \hat{y}(n-na) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{a}_{na}} \\ &\hat{f}\big(u(n-1),\mathbf{w}\big) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{b}_1} \ldots \hat{f}\big(u(n-nb),\mathbf{w}\big) \\ &- \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{b}_{nb}}\Bigg]^T.\end{aligned} \tag{3.96}$$

For BPTT, see (3.100) and (3.101) in Appendix 3.1, it follows that

$$\begin{aligned}\mathbf{x}(n) = \Bigg[& -\hat{y}(n-1) - \sum_{i_1=1}^{K} \hat{y}(n-i_1-1) \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \dots - \hat{y}(n-na) \\ & - \sum_{i_1=1}^{K} \hat{y}(n-i_1-na) \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \quad \hat{f}\big(u(n-1), \mathbf{w}\big) \\ & + \sum_{i_1=1}^{K} \hat{f}\big(u(n-i_1-1), \mathbf{w}\big) \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \dots \hat{f}\big(u(n-nb), \mathbf{w}\big) \\ & + \sum_{i_1=1}^{K} \hat{f}\big(u(n-i_1-nb), \mathbf{w}\big) \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \Bigg]^T . \end{aligned} \tag{3.97}$$

Therefore, the vector $\mathbf{x}(n)$ defined by (3.95) can be replaced in (3.91) and (3.92) by (3.96) or (3.97).

3.7 Summary

This chapter deals with different gradient calculation methods for neural network Hammerstein models. An important advantage of neural network models is that they do not require static nonlinearity to be expressed by a polynomial of a given order. The gradient in series-parallel models, which are suitable for the identification of disturbance-free systems, can be effectively calculated using the BPS method. On the other hand, parallel models due to their recurrent nature are more useful in the case of additive output disturbances but their training procedures are more complex. In general, the calculation of the gradient in parallel models requires much more computational effort than in series-parallel ones. Fortunately, the parallel Hammerstein model contains only one recurrent node which makes the SM algorithm only a little more computationally intensive. Computational complexity of the truncated BPTT algorithm is moderate as well. It is approximately equal to the complexity of BPS multiplied by the number of unfolded time steps K, which is usually not greater than a few. The combined steepest descent and RLS or RPLR algorithms require only slightly more computation than homogeneous ones. This stems from the fact that the number of operations necessary to calculate $\hat{\boldsymbol{\theta}}(n)$ depends on the square of the number of linear dynamic model parameters, which is commonly assumed to be low.
A higher accuracy of gradient approximation in the SM and BPTT learning algorithms may result in their higher convergence rate. The application of the combined steepest descent and RLS or RPLR algorithms may increase the convergence rate further.
In spite of the fact that only sequential versions of gradient-based learning algorithms are derived and discussed in this chapter, their batch mode counterparts are easy to be derived having the rules of gradient calculation

determined. Sequential algorithms have low computational and memory requirements. Moreover, they are less likely to be trapped in a shallow local minimum due to the use of pattern by pattern updating of weights, which makes the search more stochastic in nature. In the batch mode, in contrast to the sequential mode, the parameters of the linear dynamic model are kept constant during each sweeping of the input-output data (iteration). As a result, the exact value of the gradient is obtained applying the SM or BPTT algorithms. In the sequential mode, the parameters of the linear dynamic model are updated after processing every input-output pattern and thus only an approximate gradient can be obtained using the SM. An advantage of the sequential version of the BPTT algorithm is that model unfolding can be made on the basis of actual values of linear dynamic model parameters. Therefore, the BPTT algorithm can provide a more accurate approximation of the gradient than the SM. Theoretically, exact values of the gradient can be obtained if not only actual values of the linear dynamic model parameters are used but also the linear dynamic model is completely unfolded back in time, according to (3.81).

3.8 Appendix 3.1. Gradient derivation of truncated BPTT. SISO Hammerstein models

Assume that the parallel SISO Hammerstein model is K times unfolded back in time. Our primary goal is to find expressions for partial derivatives of the actual model output $\hat{y}(n)$ w.r.t. the past model outputs $\hat{y}(n–i_1)$, $i_1 = 1, \ldots, K$. Differentiating (3.14), we have

$$\frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} = -\sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n-m)}{\partial \hat{y}(n-i_1)}. \tag{3.98}$$

The terms $\partial \hat{y}(n-m)/\partial \hat{y}(n-i_1)$ are equal to zero for $m > i_1$. The partial derivatives (3.98) can be expressed by the parameters $\hat{a}_1, \ldots, \hat{a}_{na}$. Therefore, (3.98) can be written in a more convenient form as

$$\frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} = -\sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1+m)}. \tag{3.99}$$

Using (3.99), we can start the calculation with $i_1 = 1$ and then proceed back in time employing the partial derivatives calculated previously to calculate the successive ones. Having calculated (3.99), the next step is the calculation of partial derivatives of $\hat{y}(n)$ w.r.t. the model parameters $\hat{a}_1, \ldots, \hat{a}_{na}, \hat{b}_1, \ldots, \hat{b}_{nb}$, and $w_{10}^{(1)}, \ldots, w_{M1}^{(1)}, w_{10}^{(2)}, \ldots, w_{1M}^{(2)}$.
Let $\partial^+ \hat{y}(n)/\partial \hat{a}_k$, $k = 1, \ldots, na$, and $\partial^+ \hat{y}(n)/\partial \hat{b}_k$, $k = 1, \ldots, nb$, denote partial derivatives of the model output unfolded back in time, and $\partial \hat{y}(n)/\partial \hat{a}_k$ and $\partial \hat{y}(n)/\partial \hat{b}_k$ – partial derivatives calculated from (3.14) without taking into account the dependence of past model outputs on $\hat{a}_k$ and $\hat{b}_k$, respectively. The differentiation of (3.14) gives

$$\begin{aligned}\frac{\partial^+ \hat{y}(n)}{\partial \hat{a}_k} &= \frac{\partial \hat{y}(n)}{\partial \hat{a}_k} + \sum_{i_1=1}^{K} \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \frac{\partial \hat{y}(n-i_1)}{\partial \hat{a}_k} \\ &= -\hat{y}(n-k) - \sum_{i_1=1}^{K} \hat{y}(n-k-i_1) \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)},\end{aligned} \tag{3.100}$$

$$\begin{aligned}\frac{\partial^+ \hat{y}(n)}{\partial \hat{b}_k} &= \frac{\partial \hat{y}(n)}{\partial \hat{b}_k} + \sum_{i_1=1}^{K} \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \frac{\partial \hat{y}(n-i_1)}{\partial \hat{b}_k} \\ &= \hat{f}\big(u(n-k),\mathbf{w}\big) + \sum_{i_1=1}^{K} \hat{f}\big(u(n-k-i_1),\mathbf{w}\big) \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)}.\end{aligned} \tag{3.101}$$

Partial derivatives of the output of the model unfolded back in time w.r.t. w_c, $c = 1, \ldots, 3M + 1$, can be calculated as follows:

$$
\begin{aligned}
\frac{\partial^+ \hat{y}(n)}{\partial w_c} &= \sum_{i_1=0}^{K} \sum_{m=1}^{nb} \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \frac{\partial \hat{y}(n-i_1)}{\partial \hat{f}\big(u(n-m-i_1),\mathbf{w}\big)} \frac{\partial \hat{f}\big(u(n-m-i_1),\mathbf{w}\big)}{\partial w_c} \\
&= \sum_{i_1=0}^{K} \sum_{m=1}^{nb} \hat{b}_m \frac{\partial \hat{y}(n)}{\partial \hat{y}(n-i_1)} \frac{\partial \hat{f}\big(u(n-m-i_1),\mathbf{w}\big)}{\partial w_c}.
\end{aligned}
\tag{3.102}
$$

To calculate (3.102), we also need $\partial \hat{f}\big(u(n-m-i_1),\mathbf{w}\big)/\partial w_c$, which can be calculated in the same way as in the backpropagation method using (3.32) – (3.35).

3.9 Appendix 3.2. Gradient derivation of truncated BPTT. MIMO Hammerstein models

Assume that the parallel MIMO Hammerstein model is K times unfolded back in time. Differentiating (3.27), we have partial derivatives of the actual model outputs $\hat{y}_t(n)$, $t = 1, \ldots, ny$, w.r.t. the past model outputs $\hat{y}_r(n-i_1)$, $r = 1, \ldots, ny$, $i_1 = 1, \ldots, K$,

$$
\begin{aligned}
\frac{\partial \hat{y}_t(n)}{\partial \hat{y}_r(n-i_1)} &= -\sum_{m=1}^{na} \sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n-m)}{\partial \hat{y}_r(n-i_1)} \\
&= -\sum_{m=1}^{na} \sum_{m_1=1}^{ny} \hat{a}_{tm_1}^{(m)} \frac{\partial \hat{y}_{m_1}(n)}{\partial \hat{y}_r(n-i_1+m)},
\end{aligned}
\tag{3.103}
$$

where

$$
\frac{\partial \hat{y}_{m_1}(n-m)}{\partial \hat{y}_r(n-i_1)} = 0, \text{if } m > i_1 \text{ or } m = i_1 \text{ and } m_1 \neq r. \tag{3.104}
$$

Denote with $\partial^+ \hat{y}_t(n)/\partial \hat{a}_{rp}^{(k)}$, $p = 1, \ldots, ny$, and $\partial^+ \hat{y}_t(n)/\partial \hat{b}_{rp}^{(k)}$, $p = 1, \ldots, nu$, partial derivatives of the model output unfolded back in time, and $\partial \hat{y}_t(n)/\partial \hat{a}_{rp}^{(k)}$ and $\partial \hat{y}_t(n)/\partial \hat{b}_{rp}^{(k)}$ – partial derivatives calculated from (3.27) without taking into account the dependence of past model outputs on $\hat{a}_{rp}^{(k)}$ and $\hat{b}_{rp}^{(k)}$, respectively. The differentiation of (3.27) gives

$$
\begin{aligned}
\frac{\partial^+ \hat{y}_t(n)}{\partial \hat{a}_{rp}^{(k)}} &= \frac{\partial \hat{y}_t(n)}{\partial \hat{a}_{rp}^{(k)}} + \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)} \frac{\partial \hat{y}_{i_2}(n-i_1)}{\partial \hat{a}_{rp}^{(k)}} \\
&= -\delta_{tr} \hat{y}_p(n-k) - \sum_{i_1=1}^{K} \hat{y}_p(n-k-i_1) \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_r(n-i_1)},
\end{aligned}
\tag{3.105}
$$

$$
\delta_{tr} = \begin{cases} 1 & \text{for } t = r \\ 0 & \text{for } t \neq r \end{cases}, \tag{3.106}
$$

where $k = 1, \ldots, na$,

$$\begin{aligned}\frac{\partial^+ \hat{y}_t(n)}{\partial \hat{b}_{rp}^{(k)}} &= \frac{\partial \hat{y}_t(n)}{\partial \hat{b}_{rp}^{(k)}} + \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)} \frac{\partial \hat{y}_{i_2}(n-i_1)}{\partial \hat{b}_{rp}^{(k)}} \\ &= \delta_{tr} \hat{f}_p\big(\mathbf{u}(n-k),\mathbf{w}_p\big) + \sum_{i_1=1}^{K} \hat{f}_p\big(\mathbf{u}(n-k-i_1),\mathbf{w}_p\big) \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_r(n-i_1)},\end{aligned} \tag{3.107}$$

where $k = 1, \ldots, nb$. Taking into account (3.52) – (3.55), partial derivatives of the output of the model unfolded back in time w.r.t. the parameters of the nonlinear element model are calculated as

$$\begin{aligned}\frac{\partial^+ \hat{y}_t(n)}{\partial w_{ji}^{(1)}} &= \frac{\partial \hat{y}_t(n)}{\partial w_{ji}^{(1)}} + \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)} \frac{\partial \hat{y}_{i_2}(n-i_1)}{\partial w_{ji}^{(1)}} \\ &= \sum_{m=1}^{nb} \sum_{m_1=1}^{nf} \hat{b}_{tm_1}^{(m)} w_{m_1 j}^{(2)} \varphi'\big(x_j(n-m)\big) u_i(n-m) \\ &+ \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \sum_{m=1}^{nb} \sum_{m_1=1}^{nf} \hat{b}_{i_2 m_1}^{(m)} w_{m_1 j}^{(2)} \varphi'\big(x_j(n-m-i_1)\big) \\ &\quad u_i(n-m-i_1) \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)},\end{aligned} \tag{3.108}$$

$$\begin{aligned}\frac{\partial^+ \hat{y}_t(n)}{\partial w_{j0}^{(1)}} &= \frac{\partial \hat{y}_t(n)}{\partial w_{j0}^{(1)}} + \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)} \frac{\partial \hat{y}_{i_2}(n-i_1)}{\partial w_{j0}^{(1)}} \\ &= \sum_{m=1}^{nb} \sum_{m_1=1}^{nf} \hat{b}_{tm_1}^{(m)} w_{m_1 j}^{(2)} \varphi'\big(x_j(n-m)\big) \\ &+ \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \sum_{m=1}^{nb} \sum_{m_1=1}^{nf} \hat{b}_{i_2 m_1}^{(m)} w_{m_1 j}^{(2)} \varphi'\big(x_j(n-m-i_1)\big) \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)},\end{aligned} \tag{3.109}$$

$$\begin{aligned}\frac{\partial^+ \hat{y}_t(n)}{\partial w_{lj}^{(2)}} &= \frac{\partial \hat{y}_t(n)}{\partial w_{lj}^{(2)}} + \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)} \frac{\partial \hat{y}_{i_2}(n-i_1)}{\partial w_{lj}^{(2)}} \\ &= \sum_{m=1}^{nb} \hat{b}_{tl}^{(m)} \varphi\big(x_j(n-m)\big) \\ &+ \sum_{i_1=1}^{K} \sum_{i_2=1}^{ny} \sum_{m=1}^{nb} \hat{b}_{i_2 l}^{(m)} \varphi\big(x_j(n-m-i_1)\big) \frac{\partial \hat{y}_t(n)}{\partial \hat{y}_{i_2}(n-i_1)},\end{aligned} \tag{3.110}$$

$$\begin{aligned}\frac{\partial^+\hat{y}_t(n)}{\partial w_{l0}^{(2)}} &= \frac{\partial\hat{y}_t(n)}{\partial w_{l0}^{(2)}} + \sum_{i_1=1}^{K}\sum_{i_2=1}^{ny}\frac{\partial\hat{y}_t(n)}{\partial\hat{y}_{i_2}(n-i_1)}\frac{\partial\hat{y}_{i_2}(n-i_1)}{\partial w_{l0}^{(2)}}\\ &= \sum_{m=1}^{nb}\hat{b}_{tl}^{(m)} + \sum_{i_1=1}^{K}\sum_{i_2=1}^{ny}\sum_{m=1}^{nb}\hat{b}_{i_2l}^{(m)}\frac{\partial\hat{y}_t(n)}{\partial\hat{y}_{i_2}(n-i_1)},\end{aligned} \tag{3.111}$$

where $i = 1,\dots,nu$, $j = 1,\dots,M$, $l = 1,\dots,nf$.

3.10 Appendix 3.3. Proof of Theorem 3.1

In the BPTT method, the Hammerstein model is unfolded back in time, then the unfolded model is differentiated w.r.t. model parameters. This is equivalent to the differentiation of the model and unfolding the differentiated model, i.e., sensitivity models back in time. Taking into account (3.68) and (3.69), it follows from (3.39) that the outputs of sensitivity models for the parameters $\hat{a}_k$, $k = 1,\dots,na$, can be expressed as functions of the input $u(n)$:

$$\begin{aligned}\frac{\partial\hat{y}(n)}{\partial\hat{a}_k} &= -\,H_1(q^{-1})\hat{y}(n-k) = -H_1(q^{-1})H_2(q^{-1})\hat{f}\big(u(n-k),\mathbf{w}\big)\\ &= -\,H(q^{-1})\hat{f}\big(u(n-k),\mathbf{w}\big),\end{aligned} \tag{3.112}$$

where

$$H(q^{-1}) = H_1(q^{-1})H_2(q^{-1}). \tag{3.113}$$

The impulse response of the system (3.113) is given by the convolution relationship

$$h(n) = \sum_{i_2=0}^{n} h_1(i_2)h_2(n-i_2). \tag{3.114}$$

The partial derivatives $\partial\hat{y}(n)/\partial\hat{a}_k$ are related to the input $u(n)$ with the impulse response functions $h_1(n)$ and $h_2(n)$ and the function $\hat{f}(\cdot)$:

$$\begin{aligned}\frac{\partial\hat{y}(n)}{\partial\hat{a}_k} &= -\sum_{i_1=0}^{n-k} h(i_1)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big)\\ &= -\sum_{i_1=0}^{n-k}\sum_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big).\end{aligned} \tag{3.115}$$

The calculation of (3.115) requires unfolding sensitivity models $n-1$ times back in time. Thus, the partial derivatives $\partial^+\hat{y}(n)/\partial\hat{a}_k$ calculated for the sensitivity models unfolded K times back in time are

$$\frac{\partial^+ \hat{y}(n)}{\partial \hat{a}_k} = \begin{cases} -\sum\limits_{i_1=0}^{n-k}\sum\limits_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big), & \text{for } n \leqslant K+k \\ -\sum\limits_{i_1=0}^{K}\sum\limits_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big) & \\ -\sum\limits_{i_1=K+1}^{n-k}\sum\limits_{i_2=0}^{K} h_1(i_2)h_2(i_1-i_2)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big), & \text{for } n > K+k. \end{cases} \tag{3.116}$$

The errors of computing partial derivatives with the truncated BPTT method are

$$\varDelta\hat{y}_{\hat{a}_k}(n) = \begin{cases} 0, & \text{for } n \leqslant K+k \\ -\sum\limits_{i_1=K+1}^{n-k}\sum\limits_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big), & \text{for } n > K+k. \end{cases} \tag{3.117}$$

As the input $\{u(n\})$ is a sequence of zero-mean i.i.d. random variables, the output of the nonlinear element model $\{\hat{f}\big(u(n),\mathbf{w}\big)\}$ is a sequence of i.i.d. random variables as well. Their expected values m_f and variances σ_f^2 depend on both the neural network model architecture and the parameter vector $\mathbf{w}$.

$$\mathrm{E}[\hat{f}\big(u(n),\mathbf{w}\big)] = m_f, \tag{3.118}$$

$$\mathrm{E}\big\{\big[\hat{f}\big(u(n),\mathbf{w}\big) - m_f\big]^2\big\} = \sigma_f^2. \tag{3.119}$$

Therefore, the expected values of (3.117) are

$$\mathrm{E}\big(\varDelta\hat{y}_{\hat{a}_k}(n)\big) = -m_f \sum_{i_1=K+1}^{n-k}\sum_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2), \tag{3.120}$$

and the variances of the errors (3.117) are

$$\begin{aligned} \mathrm{var}\big(\varDelta\hat{y}_{\hat{a}_k}(n)\big) = {} & \mathrm{E}\Bigg[\Bigg(-\sum_{i_1=K+1}^{n-k}\sum_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big) \\ & + m_f \sum_{i_1=K+1}^{n-k}\sum_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2)\Bigg)^2\Bigg] \\ = \mathrm{E}\Bigg[\Bigg(\sum_{i_1=K+1}^{n-k}\sum_{i_2=K+1}^{i_1} & h_1(i_2)h_2(i_1-i_2)\big[\hat{f}\big(u(n-k-i_1),\mathbf{w}\big)-m_f\big]\Bigg)^2\Bigg]. \end{aligned} \tag{3.121}$$

Since $\hat{f}\big(u(n-k-i_1),\mathbf{w}\big) - m_f$, $i_1 = K+1, \ldots, n-k$, are zero-mean i.i.d. random variables, the variances of the errors (3.117) are given by (3.67) for $n > K+k$, and are equal to zero otherwise.

3.11 Appendix 3.4. Proof of Theorem 3.2

In the BPTT method, the Hammerstein model is unfolded back in time, then the unfolded model is differentiated w.r.t. model parameters. This is equivalent to the differentiation of the model and unfolding the differentiated model, i.e., sensitivity models back in time. Taking into account (3.68), it follows from (3.40) that the outputs of sensitivity models for the parameters $\hat{b}_k$, $k = 1, \ldots, nb$, can be expressed as functions of the input $u(n)$:

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = H_1(q^{-1}) \hat{f}\big(u(n-k),\mathbf{w}\big). \tag{3.122}$$

The partial derivatives $\partial \hat{y}(n)/\partial \hat{b}_k$ are related to the input $u(n)$ with the impulse response functions $h_1(n)$ and the function $\hat{f}(\cdot)$:

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \sum_{i_1=0}^{n-k} h_1(i_1) \hat{f}\big(u(n-k-i_1),\mathbf{w}\big). \tag{3.123}$$

The calculation of (3.123) requires unfolding sensitivity models $n-1$ times back in time. Thus, the partial derivatives $\partial^+ \hat{y}(n)/\partial \hat{b}_k$ calculated for sensitivity models unfolded K times back in time are

$$\frac{\partial^+ \hat{y}(n)}{\partial \hat{b}_k} = \begin{cases} \displaystyle\sum_{i_1=0}^{n-k} h_1(i_1) \hat{f}\big(u(n-k-i_1),\mathbf{w}\big), & \text{for } n \leqslant K+k \\ \displaystyle\sum_{i_1=0}^{K} h_1(i_1) \hat{f}\big(u(n-k-i_1),\mathbf{w}\big), & \text{for } n > K+k. \end{cases} \tag{3.124}$$

The errors of computing partial derivatives with the truncated BPTT method are

$$\Delta \hat{y}_{\hat{b}_k}(n) = \begin{cases} 0, & \text{for } n \leqslant K+k \\ \displaystyle\sum_{i_1=K+1}^{n-k} h_1(i_1) \hat{f}\big(u(n-k-i_1),\mathbf{w}\big), & \text{for } n > K+k. \end{cases} \tag{3.125}$$

As the input $u(n)$ is a sequence of zero-mean i.i.d. random variables, the output of the nonlinear element model $\hat{f}\big(u(n),\mathbf{w}\big)$ is a sequence of i.i.d. random variables as well. Their expected values m_f and variances σ_f^2 depend on both the neural network model architecture and the parameter vector $\mathbf{w}$:

$$\mathrm{E}[\hat{f}\big(u(n),\mathbf{w}\big)] = m_f, \tag{3.126}$$

$$\mathrm{E}\big\{\big[\hat{f}\big(u(n),\mathbf{w}\big) - m_f\big]^2\big\} = \sigma_f^2. \tag{3.127}$$

Therefore, the expected values of (3.125) are

$$\mathrm{E}\big(\varDelta\hat{y}_{\hat{b}_k}(n)\big) = m_f \sum_{i_1=K+1}^{n-k} h_1(i_1), \tag{3.128}$$

and the variances of the errors (3.125) are

$$\begin{aligned}\operatorname{var}\big(\varDelta\hat{y}_{\hat{b}_k}(n)\big) &= \mathrm{E}\Bigg[\bigg(\sum_{i_1=K+1}^{n-k} h_1(i_1)\hat{f}\big(u(n-k-i_1),\mathbf{w}\big) - m_f\sum_{i_1=K+1}^{n-k} h_1(i_1)\bigg)^2\Bigg]\\ &= \mathrm{E}\Bigg[\bigg(\sum_{i_1=K+1}^{n-k} h_1(i_1)\big[\hat{f}\big(u(n-k-i_1),\mathbf{w}\big) - m_f\big]\bigg)^2\Bigg].\end{aligned} \tag{3.129}$$

Since $\hat{f}\big(u(n-k-i_1),\mathbf{w}\big) - m_f$, $i_1 = K+1,\ldots,n-k$, are zero-mean i.i.d. random variables, the variances of the errors (3.125) are given by (3.70) for $n > K+k$, and are equal to zero otherwise.

3.12 Appendix 3.5. Proof of Theorem 3.3

To prove Theorem 3.3, we will consider gradient calculation with the sensitivity model (3.41) completely unfolded back in time, and the sensitivity model unfolded back in time up to K steps. Using the backward shift operator notation, (3.41) can be written as

$$\frac{\partial\hat{y}(n)}{\partial w_c} = qH_2(q^{-1})\frac{\partial\hat{f}\big(u(n-1),\mathbf{w}\big)}{\partial w_c}, \tag{3.130}$$

where $c = 1,\ldots,3M$. The above partial derivatives, corresponding to the model completely unfolded back in time, can be expressed by the following convolution summations:

$$\begin{aligned}\frac{\partial\hat{y}(n)}{\partial w_c} &= \sum_{i_1=0}^{n-1} h_2(i_1+1)\frac{\partial\hat{f}\big(u(n-i_1-1),\mathbf{w}\big)}{\partial w_c}\\ &= \sum_{i_1=1}^{n} h_2(i_1)\frac{\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)}{\partial w_c}.\end{aligned} \tag{3.131}$$

In the truncated BPTT method, these summations are performed only up to K times back in time and partial derivatives of the model output w.r.t. w_c are

$$\frac{\partial^+\hat{y}(n)}{\partial w_c} = \begin{cases}\displaystyle\sum_{i_1=1}^{n} h_2(i_1)\frac{\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)}{\partial w_c}, & \text{for } n \leqslant K+1\\ \displaystyle\sum_{i_1=1}^{K+1} h_2(i_1)\frac{\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)}{\partial w_c}, & \text{for } n > K+1.\end{cases} \tag{3.132}$$

From (3.131) and (3.132), it follows that the errors of computing partial derivatives with the truncated BPTT method are

$$\Delta\hat{y}_{w_c}(n) = \begin{cases} 0, & \text{for } n \leqslant K+1 \\ \displaystyle\sum_{i_1=K+2}^{n} h_2(i_1)\frac{\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)}{\partial w_c}, & \text{for } n > K+1. \end{cases} \tag{3.133}$$

As the input $u(n)$ is a sequence of zero-mean i.i.d. random variables, the partial derivatives $\partial\hat{f}\big(u(n),\mathbf{w}\big)/\partial w_c$, are sequences of i.i.d. random variables as well. Their expected values m_{w_c} and variances $\sigma^2_{w_c}$ depend on both the neural network model architecture and the parameter vector $\mathbf{w}$:

$$\mathrm{E}\left[\frac{\partial\hat{f}\big(u(n),\mathbf{w}\big)}{\partial w_c}\right] = m_{w_c}, \tag{3.134}$$

$$\mathrm{E}\left\{\left[\frac{\partial\hat{f}\big(u(n),\mathbf{w}\big)}{\partial w_c} - m_{w_c}\right]^2\right\} = \sigma^2_{w_c}. \tag{3.135}$$

Therefore, the expected values of (3.133) are

$$\mathrm{E}\big(\Delta\hat{y}_{w_c}(n)\big) = m_{w_c}\sum_{i_1=K+2}^{n} h_2(i_1), \tag{3.136}$$

and the variances of (3.133) are

$$\begin{aligned} \mathrm{var}\big(\Delta\hat{y}_{w_c}(n)\big) &= \mathrm{E}\left[\left(\sum_{i_1=K+2}^{n} h_2(i_1)\frac{\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)}{\partial w_c} - m_{w_c}\sum_{i_1=K+2}^{n} h_2(i_1)\right)^2\right] \\ &= \mathrm{E}\left[\left(\sum_{i_1=K+2}^{n} h_2(i_1)\left[\frac{\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)}{\partial w_c} - m_{w_c}\right]\right)^2\right]. \end{aligned} \tag{3.137}$$

Since $\partial\hat{f}\big(u(n-i_1),\mathbf{w}\big)/\partial w_c - m_{w_c}$, $i_1 = K+2,\ldots,n$, are zero-mean i.i.d. random variables, the variances of the errors (3.133) are given by (3.72) for $n > K+1$, and are equal to zero otherwise.

3.13 Appendix 3.6. Proof of Theorem 3.4

From (3.41), it follows that

$$\frac{\partial\hat{y}(n)}{\partial w^{(2)}_{10}} = qH_2(q^{-1})\frac{\partial\hat{f}\big(u(n-1),\mathbf{w}\big)}{\partial w^{(2)}_{10}}. \tag{3.138}$$

Taking into account (3.35), we have

$$\frac{\partial \hat{y}(n)}{\partial w_{10}^{(2)}} = \sum_{i_1=0}^{n-1} h_2(i_1+1) = \sum_{i_1=1}^{n} h_2(i_1). \tag{3.139}$$

Unfolding (3.138) back in time for K steps gives

$$\frac{\partial^{+} \hat{y}(n)}{\partial w_{10}^{(2)}} = \begin{cases} \displaystyle\sum_{i_1=1}^{n} h_2(i_1), & \text{for } n \leqslant K+1 \\ \displaystyle\sum_{i_1=1}^{K+1} h_2(i_1), & \text{for } n > K+1. \end{cases} \tag{3.140}$$

Therefore,

$$\Delta\hat{y}_{w_{10}^{(2)}}(n) = \begin{cases} 0, & \text{for } n \leqslant K+1 \\ \displaystyle\sum_{i_1=K+2}^{n} h_2(i_1), & \text{for } n > K+1. \end{cases} \tag{3.141}$$