Lecture Notes
in Control and Information Sciences 310

Editors: M. Thoma · M. Morari

Lecture Notes in Control and Information Sciences

Edited by M. Thoma and M. Morari

Further volumes of this series are listed at the end of the book or found on our homepage: springeronline.com

Vol. 309: Kumar, V.; Leonard, N.; Morse, A.S. (Eds.)
Cooperative Control
301 p. 2005 [3-540-22861-6]

Vol. 308: Tarbouriech, S.; Abdallah, C.T.; Chiasson, J. (Eds.)
Advances in Communication Control Networks
358 p. 2005 [3-540-22819-5]

Vol. 307: Kwon, S.J.; Chung, W.K.
Perturbation Compensator based Robust Tracking Control and State Estimation of Mechanical Systems
158 p. 2004 [3-540-22077-1]

Vol. 306: Bien, Z.Z.; Stefanov, D. (Eds.)
Advances in Rehabilitation
472 p. 2004 [3-540-21986-2]

Vol. 305: Nebylov, A.
Ensuring Control Accuracy
256 p. 2004 [3-540-21876-9]

Vol. 304: Margaris, N.I.
Theory of the Non-linear Analog Phase Locked Loop
303 p. 2004 [3-540-21339-2]

Vol. 303: Mahmoud, M.S.
Resilient Control of Uncertain Dynamical Systems
278 p. 2004 [3-540-21351-1]

Vol. 302: Filatov, N.M.; Unbehauen, H.
Adaptive Dual Control: Theory and Applications
237 p. 2004 [3-540-21373-2]

Vol. 301: de Queiroz, M.; Malisoff, M.; Wolenski, P. (Eds.)
Optimal Control, Stabilization and Nonsmooth Analysis
373 p. 2004 [3-540-21330-9]

Vol. 300: Nakamura, M.; Goto, S.; Kyura, N.; Zhang, T.
Mechatronic Servo System Control
Problems in Industries and their Theoretical Solutions
212 p. 2004 [3-540-21096-2]

Vol. 299: Tarn, T.-J.; Chen, S.-B.; Zhou, C. (Eds.)
Robotic Welding, Intelligence and Automation
214 p. 2004 [3-540-20804-6]

Vol. 298: Choi, Y.; Chung, W.K.
PID Trajectory Tracking Control for Mechanical Systems
127 p. 2004 [3-540-20567-5]

Vol. 297: Damm, T.
Rational Matrix Equations in Stochastic Control
219 p. 2004 [3-540-20516-0]

Vol. 296: Matsuo, T.; Hasegawa, Y.
Realization Theory of Discrete-Time Dynamical Systems
235 p. 2003 [3-540-40675-1]

Vol. 295: Kang, W.; Xiao, M.; Borges, C. (Eds)
New Trends in Nonlinear Dynamics and Control, and their Applications
365 p. 2003 [3-540-10474-0]

Vol. 294: Benvenuti, L.; De Santis, A.; Farina, L. (Eds)
Positive Systems: Theory and Applications (POSTA 2003)
414 p. 2003 [3-540-40342-6]

Vol. 293: Chen, G. and Hill, D.J.
Bifurcation Control
320 p. 2003 [3-540-40341-8]

Vol. 292: Chen, G. and Yu, X.
Chaos Control
380 p. 2003 [3-540-40405-8]

Vol. 291: Xu, J.-X. and Tan, Y.
Linear and Nonlinear Iterative Learning Control
189 p. 2003 [3-540-40173-3]

Vol. 290: Borrelli, F.
Constrained Optimal Control of Linear and Hybrid Systems
237 p. 2003 [3-540-00257-X]

Vol. 289: Giarré, L. and Bamieh, B.
Multidisciplinary Research in Control
237 p. 2003 [3-540-00917-5]

Vol. 288: Taware, A. and Tao, G.
Control of Sandwich Nonlinear Systems
393 p. 2003 [3-540-44115-8]

Vol. 287: Mahmoud, M.M.; Jiang, J. and Zhang, Y.
Active Fault Tolerant Control Systems
239 p. 2003 [3-540-00318-5]

Vol. 286: Rantzer, A. and Byrnes C.I. (Eds)
Directions in Mathematical Systems Theory and Optimization
399 p. 2003 [3-540-00065-8]

Vol. 285: Wang, Q.-G.
Decoupling Control
373 p. 2003 [3-540-44128-X]

Vol. 284: Johansson, M.
Piecewise Linear Control Systems
216 p. 2003 [3-540-44124-7]

Vol. 283: Fielding, Ch. et al. (Eds)
Advanced Techniques for Clearance of Flight Control Laws
480 p. 2003 [3-540-44054-2]

Vol. 282: Schröder, J.
Modelling, State Observation and Diagnosis of Quantised Systems
368 p. 2003 [3-540-44075-5]

A. Janczak

Identification of Nonlinear Systems Using Neural Networks and Polynomial Models

A Block-Oriented Approach

With 79 Figures and 22 Tables

Springer

Author

Prof. Andrzej Janczak
University of Zielona Góra
Institute of Control and Computation Engineering
ul. Podgorna 50
65-246 Zielona Góra
Poland

ISSN 0170-8643

ISBN 3-540-23185-4 **Springer Berlin Heidelberg New York**

Library of Congress Control Number: 2004097177

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Data conversion by the authors.
Final processing by PTP-Berlin Protago-TEX-Production GmbH, Germany
Cover-Design: design & production GmbH, Heidelberg
Printed on acid-free paper 62/3020Yu - 5 4 3 2 1 0

Preface

The identification of nonlinear systems using the block-oriented approach has been developed since the half of 1960s. A large amount of knowledge on this subject has been accumulated through literature. However, publications are scattered over many papers and there is no book which presents the subject in a unified framework. This has created an increasing need to systemize the existing identification methods and along with a presentation of some original results have been the main incentive to write this book. In writing the book, an attempt has been made at the presentation of some new ideas concerning the model parameter adjusting with gradient-based techniques.

Two types of models, considered in this book, use neural networks and polynomials as representations of Wiener and Hammerstein systems. The focus is placed on Wiener and Hammerstein models in which the nonlinear element is represented by a polynomial or a two-layer perceptron neural network with hyperbolic tangent hidden layer nodes and linear output nodes. Pulse transfer function models are common representations of system dynamics in both neural network and polynomial Wiener and Hammerstein models.

Neural network and polynomial models reveal different properties such as the approximation accuracy, computational complexity, available parameter and structure optimization methods, etc. All these differences make them complementary in solving many practical problems. For example, it is well known that the approximation of some nonlinear functions requires polynomials of a high order and this, in turn, results in a high parameter variance error. The approximation with neural network models is an interesting alternative in such cases.

The book results mainly from my research in the area of nonlinear system identification that have been performed since 1995. Two exceptions from this rule are Chapter 1, containing the introductory notes, and Chapter 5, which reviews the well-known Hammerstein system identification methods based on polynomial models of the nonlinearity. In writing the book, an emphasis has been put on presenting various identification methods, which are applicable to both neural network and polynomial models of Wiener and Hammerstein systems, in a unified framework.

The book starts with a survey of discrete-time models of time-invariant dynamic systems. Then the multilayer perceptron neural network is introduced and a brief review of the existing methods for the identification of Wiener and Hammerstein systems is presented. Two subsequent Chapters (2 and 3) introduce neural network models of Wiener and Hammerstein systems and present different algorithms for the calculation of the gradient or the approximate gradient of the model output w.r.t. model parameters. For both Wiener and Hammerstein models, the accuracy of gradient evaluation with the truncated backpropagation through time algorithm is analyzed. The discussion also includes advantages and disadvantages of the algorithms in terms of their approximation accuracy, computational requirements, and weight updating methods. Next, in Chapter 4, we present identification methods, which use polynomial models of Wiener systems. The parameters of the linear dynamic system and the inverse nonlinearity are estimated with the least squares method, and a combined least squares and instrumental variables approach. To estimate parameters of the noninverted nonlinearity, the recursive prediction error and the pseudolinear regression methods are proposed. Then the existing identification methods based on polynomial Hammerstein models are reviewed and presented in Chapter 5. Wiener and Hammerstein models are two examples of block-oriented models which have found numerous industrial applications. The most important of them, including nonlinear system modelling, control, and fault detection and isolation, are reviewed in Chapter 6. This chapters presents also two applications of Wiener and Hammerstein models – estimation of system parameter changes, and modelling vapor pressure dynamics in a five stage sugar evaporation station.

The book contains the results od research conducted by the author with the kind support of the State Committee for Scientific Research in Poland under the grant No 4T11A01425 and with the additional support of the European Union within the 5th Framework Programme under *DAMADICS* project No HPRN-CT-2000-00110.

A few people have contributed directly or indirectly to this book. First of all, I would like to express my sincere gratitude to Professor Józef Korbicz for his constant help, advice, encouragement, and support.
I am very grateful to Ms Agnieszka Rożewska for proofreading and linguistic advice on the text. I highly appreciate the help of my colleagues from the Institute of Control and Computational Engineering, University of Zielona Góra.

Zielona Góra, *Andrzej Janczak*
July 2004

Contents

Symbols and notation

a_k	kth parameter of the polynomial $A(q^{-1})$
$\hat{a}_k$	kth parameter of the polynomial $\hat{A}(q^{-1})$
$A(q^{-1})$	denominator polynomial of the system pulse transfer function
$\hat{A}(q^{-1})$	denominator polynomial of the model pulse transfer function
$\mathcal{A}(q^{-1})$	denominator polynomial of the faulty system pulse transfer function
$\mathbf{A}$	state space matrix, $(nx \times nx)$
$\hat{\mathbf{A}}^{(m)}$	mth parameter matrix of the MIMO dynamic model. The MIMO Wiener model, $(ns \times ns)$; the MIMO Hammerstein model, $(ny \times ny)$
b_k	kth parameter of the polynomial $B(q^{-1})$
$\hat{b}_k$	kth parameter of the polynomial $\hat{B}(q^{-1})$
$B(q^{-1})$	nominator polynomial of the system pulse transfer function
$\hat{B}(q^{-1})$	nominator polynomial of the model pulse transfer function
$\mathcal{B}(q^{-1})$	nominator polynomial of the faulty system pulse transfer function
$\mathbf{B}$	control matrix, $(nx \times nu)$
$\hat{\mathbf{B}}^{(m)}$	mth parameter matrix of the MIMO dynamic model. The MIMO Wiener model, $(ns \times nu)$; the MIMO Hammerstein model, $(ny \times nf)$
$\mathbf{C}$	observation matrix, $(ny \times nx)$
$\mathbf{D}$	matrix describing the effect of inputs on outputs, $(ny \times nu)$
$e(n)$	identification error (one-step-ahead prediction error)
E	mathematical expectation
$f(\cdot)$	nonlinear function of the system
$\hat{f}(\cdot)$	nonlinear function of the model
$\hat{f}_k(\cdot)$	kth nonlinear function of the MIMO nonlinear element model
$\hat{\mathbf{f}}(\cdot)$	nonlinear function of the MIMO nonlinear element model
$f^{-1}(\cdot)$	inverse nonlinear function of the system
$\hat{f}^{-1}(\cdot)$	inverse nonlinear function of the model
$\hat{g}_k(\cdot)$	kth nonlinear function of the MIMO inverse nonlinear element model
$\hat{\mathbf{g}}(\cdot)$	nonlinear function of the MIMO inverse nonlinear element model
$h(n)$	system impulse response
$h_1(n)$	impulse response of the sensitivity model
$h_2(n)$	impulse response of the linear dynamic model
$H_1(q^{-1})$	pulse transfer function of sensitivity models
$H_2(q^{-1})$	pulse transfer function of the linear dynamic model
J	global error function
$J(n)$	local error function
K	number of unfolded time steps
m_f	expected value of $\hat{f}(u(n), \mathbf{w})$

m_{w_c}	expected value of $\partial \hat{f}(u(n), \mathbf{w})/\partial w_c$
M	number of nonlinear nodes
n	discrete time
na	order of the polynomials $A(q^{-1})$ and $\hat{A}(q^{-1})$
nb	order of the polynomials $B(q^{-1})$ and $\hat{B}(q^{-1})$
nf	number of outputs of MIMO Hammerstein nonlinear element model
ns	number of outputs of MIMO Wiener linear dynamic model;
nu	number of system (model) inputs
ny	number of system (model) outputs
N	number of input-output measurements
q^{-1}	backward shift operator
$\mathbb{R}^d$	Euclidean d-dimensional space
$s(n)$	output of the linear dynamic part of Wiener system
$\hat{s}_k(n)$	kth output of the linear dynamic part of MIMO Wiener model
$\hat{s}(n)$	output of the linear dynamic part of Wiener model
$\hat{\mathbf{s}}(n)$	output of the linear dynamic part of MIMO Wiener model
$u(n)$	system input
$\mathbf{u}(n)$	MIMO system input
var	variance
$v_{ji}^{(1)}$	ith weight of the jth hidden layer node of the inverse nonlinear element model
$v_{kj}^{(2)}$	jth weight of the kth output node of the inverse nonlinear element model
$\hat{v}(n)$	output of the nonlinear element part of Hammerstein model
$\mathbf{v}$	weight vector of the inverse nonlinear element model
$\mathbf{v}_k$	kth path weight vector of the MIMO inverse nonlinear element model
$\mathbf{V}$	weight vector of the MIMO inverse nonlinear element model
$w_{ji}^{(1)}$	ith weight of the jth hidden layer node of the nonlinear element model
$w_{kj}^{(2)}$	jth weight of the kth output node of the nonlinear element model
$\mathbf{w}$	weight vector of the nonlinear element model
$\mathbf{w}_k$	kth path weight vector of the MIMO nonlinear element model,
$\mathbf{W}$	weight vector of the MIMO nonlinear element model
$x_j(n)$	activation of the jth nonlinear node
$\mathbf{x}(n)$	regression vector; system state
$\hat{\mathbf{x}}(n)$	model state
$y(n)$	system output
$\hat{y}(n)$	model output
$y_k(n)$	kth output of the MIMO Wiener (Hammerstein) system
$\hat{y}_k(n)$	kth output of the MIMO Wiener (Hammerstein) model
$\hat{y}(n\|n-1)$	one-step-ahead predictor of $y(n)$
$\mathbf{y}(n)$	MIMO system output
$\hat{\mathbf{y}}(n)$	MIMO model output

$z_j(n)$	activation of the jth nonlinear node of the inverse nonlinear element model
$\mathbf{z}(n)$	instrumental variables vector
$\hat{\gamma}_k$	kth parameter of the polynomial of the inverse nonlinear element model
γ_k	kth parameter of the polynomial of the inverse nonlinear element
$\Delta A(q^{-1})$	change in the pulse transfer function denominator of the linear dynamic system
$\Delta f(\cdot)$	change in the nonlinear function of the nonlinear element
$\Delta f^{-1}(\cdot)$	change in the nonlinear function of the inverse nonlinear element
$\Delta\hat{s}_{\hat{a}_k}(n)$	computation error of $\partial\hat{s}(n)/\partial\hat{a}_k$
$\Delta\hat{s}_{\hat{b}_k}(n)$	computation error of $\partial\hat{s}(n)/\partial\hat{b}_k$
$\Delta\hat{y}_{\hat{a}_k}(n)$	computation error of $\partial\hat{y}(n)/\partial\hat{a}_k$
$\Delta\hat{y}_{\hat{b}_k}(n)$	computation error of $\partial\hat{y}(n)/\partial\hat{b}_k$
$\Delta\hat{y}_{w_c}(n)$	computation error of $\partial\hat{y}(n)/\partial w_c$
$\epsilon(n)$	discrete white noise disturbance
$\varepsilon(n)$	additive system output disturbance
η	learning rate
$\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}$	parameter vector
λ	exponential forgetting factor
μ_k	kth parameter of the polynomial of the nonlinear element,
$\hat{\mu}_k$	kth parameter of the polynomial of the nonlinear element model
$\xi_{a_k}(n)$	calculation accuracy degree of $\partial\hat{s}(n)/\partial\hat{a}_k$ – the Wiener model; $\partial\hat{y}(n)/\partial\hat{a}_k$ – the Hammerstein model
$\xi_{b_k}(n)$	calculation accuracy degree of $\partial\hat{s}(n)/\partial\hat{b}_k$ – the Wiener model; $\partial\hat{y}(n)/\partial\hat{b}_k$ – the Hammerstein model
$\xi_{w_c}(n)$	calculation accuracy degree of $\partial\hat{y}(n)/\partial w_c$
σ^2	variance of $u(n)$
σ_f^2	variance of $\hat{f}(u(n), \mathbf{w})$
$\sigma_{w_c}^2$	variance of $\partial\hat{f}(u(n), \mathbf{w})/\partial w_c$
$\varphi(\cdot)$	nonlinear activation function
$\boldsymbol{\psi}(n)$	gradient of the Wiener model output

List of abbreviations

i.i.d.	independent and identically distributed
r.h.s	right hand side
w.r.t	with respect to
AR	autoregressive
ARMA	autoregressive moving average
ARMAX	autoregressive moving average with exogenous input

ARX	autoregressive with exogenous input
BJ	Box-Jenkins
BP	back propagation
BPP	back propagation for parallel models
BPPT	back propagation through time
BPS	back propagation for series-parallel models
CSTR	continuous stirred tank reactor
DLOP	discrete Legendre orthogonal polynomial
FDI	fault detection and isolation
ELS	extended least squares
FIR	finite impulse response model
IMC	internal model control
IV	instrumental variables
MA	moving average model
MIMO	multiple-input multiple-output
MISO	multiple-input single-output
MLP	multilayer perceptron
MPC	model-based predictive control
MSE	mean square error
NAR	nonlinear autoregressive
NARMA	nonlinear autoregressive moving average
NARMAX	nonlinear autoregressive moving average with exogenous input
NARX	nonlinear autoregressive with exogenous input
NBJ	nonlinear Box-Jenkins
NFIR	nonlinear finite impulse response
NOBF	nonlinear orthonormal basis function
NOE	nonlinear output error
NMA	nonlinear moving average
OBFP	orthogonal basis with fixed poles
OE	output error
PE	prediction error
PI	proportional plus integral
PID	proportional plus integral plus derivative
PRBS	pseudorandom binary sequence
RIV	recursive instrumental variables
RLS	recursive least squares
RELS	recursive extended least squares
RMS	root mean square
RPE	recursive prediction error
RPLR	recursive pseudolinear regression
SIMO	single-input multiple-output
SISO	single-input single-output
SM	sensitivity method
WMPC	Wiener model-based predictive control

1

Introduction

The class of block-oriented nonlinear models includes complex models which are composed of linear dynamic systems and nonlinear static elements. Wiener and Hammerstein models are the most known and the most widely implemented members of this class. A model is called the Wiener model if the linear dynamic block (element) precedes the nonlinear static one. In the Hammerstein model, the connection order is reversed.

Models of nonlinear static elements can be realized in different forms such as polynomials, splines, basis functions, wavelets, neural networks, look-up tables, and fuzzy models. Impulse response models, pulse transfer models, and state space models are common representations of linear dynamic systems. Depending on realization forms of both of these elements, various structures of Wiener and Hammerstein models can be obtained. To evaluate and compare them, the following properties are commonly taken into account: approximation accuracy, extrapolation behavior, interpolation behavior, smoothness, sensitivity to noise, available parameter optimization methods, and available structure optimization methods.

From the approximation theorem of Weierstrass, it follows that any continuous function defined on the interval $[a, b]$ can be approximated arbitrarily closely by a polynomial. Polynomial models are widely used as models of nonlinear elements. A great advantage of polynomial models is effective parameter optimization, which can be performed off-line with the least squares method or on-line with its recursive version. Moreover, structure selection can also be performed effectively with the orthogonal least squares algorithm, in which the set of regressors is transformed into a set of orthogonal basis vectors. With this algorithm, it is possible to calculate individual contribution from each basis vector to output variance.

Polynomials have some fundamental disadvantages as well. First of all, although any continuous function can be approximated arbitrary closely by a polynomial, some nonlinear functions require a very high polynomial order. In multi-input multi-output models, the number of parameters grows strongly when the number of inputs increases. This, increases the model uncer-

A. Janczak: Identification of Nonlinear Systems, LNCIS 310, pp. 1–30, 2005.

tainty and may cause the optimization problem numerically ill-conditioned. The other disadvantages of polynomials are their oscillatory interpolation and extrapolation properties. Therefore, in practice, the application of polynomial models is recommended only in some specific cases where the system structure can be assumed to be approximately of the polynomial type.

An alternative to polynomial models are neural network models of the multilayer perceptron architecture. Multilayer perceptrons are feedforward neural networks containing one or more hidden layers of nonlinear elements, but one hidden layer is the most common choice in practice. The application of multilayer perceptrons in approximation problems is justified by their universal approximation property which states that a single hidden layer is sufficient to uniformly approximate any continuous function with support in a unit hypercube [31]. Multilayer perceptrons, owing to their advantages such as high approximation accuracy, lower numbers of nodes and weights in comparison with other model architectures, capability to generate a wide variety of functions, are the most frequently applied neural networks. Unlike polynomials, multilayer perceptron models do not suffer oscillatory interpolation and extrapolation behavior. They reveal a tendency to monotonic interpolation. The extrapolation behavior is also smooth but in a long range the network response tends to a constant value owing to the saturation of commonly applied sigmoidal functions. Multilayer perceptron models are very useful for high dimensional problems as well. This comes from the fact that the numer of weights in a multilayer perceptron model is proportional to the number of inputs. In contrast to polynomial models, multilayer perceptron models can be successfully used not only to represent systems described by polynomials but also by infinite power series expansions. Summarizing, as they offer the user some different interesting features, multilayer perceptron models are complementary to polynomial ones.

Obviously, multilayer perceptron models are not free of drawbacks. The most significant of them are the use of local optimization methods to update the weights, and a risk of getting trapped in a shallow local minimum. This leads often to the necessity of repeated training with different weight initializations. Moreover, trial and error techniques have to be used for some parameters such as initial weights, learning rates, etc. Also, the available model structure optimization methods are rather computationally intensive.

Now, with the development of the identification of Wiener and Hammerstein systems it is possible to systemize different techniques and to present them in a unified framework. We attempt at such a presentation in this monograph by reviewing the existing approaches along with a presentation of original research papers and some results that have not been published yet.

Neural network models of Wiener and Hammerstein systems considered in Chapters 2 and 3 are composed of a multilayer perceptron model of the nonlinear element and one or more linear nodes with tapped delay lines constituting a model of the linear dynamic system. Series-parallel Wiener models contain also another multilayer perceptron model of the inverse nonlinear element.

Two basic configurations of models, i.e., series-parallel and parallel models are discussed. In series-parallel models, the gradient can be calculated with the well-known backpropagation method. In parallel models, only a crude approximation of the gradient can be obtained with the backpropagation method. Therefore, two other methods, referred to as the sensitivity method and the backpropagation through time method, which provide the exact value of the gradient or its more accurate approximation, should be taken into account. All these gradient calculation methods are derived in a unified manner, both for the SISO and MIMO cases. Computational complexity of the methods is analyzed and expressed in terms of polynomial orders and the number of unfolded time steps in the case of the truncated backpropagation through time method. The accuracy of gradient calculation with the truncated backpropagation through time method is analyzed as well. It is shown that the accuracy of gradient calculation depends on the numbers of discrete time steps necessary for the impulse responses of sensitivity models to decrease to negligible small values. Based on this result, adaptive procedures for adjusting the number of unfolded time steps are proposed to meet the specified degrees of accuracy. The original contribution of the book comprises new approaches to the identification of Wiener and Hammerstein systems and some concerned theoretical results. For both SISO and MIMO neural network models, the following gradient calculation methods are derived and analyzed:

- Backpropagation for series-parallel Wiener models.
- Backpropagation for parallel Wiener models.
- Sensitivity method for parallel Wiener models.
- Truncated backpropagation through time for parallel Wiener models.
- Backpropagation for series-parallel Hammerstein models.
- Backpropagation for parallel Hammerstein models.
- Sensitivity method for parallel Hammerstein models.
- Truncated backpropagation through time for parallel Hammerstein models.

Having the rules for gradient calculation derived, various gradient-based learning algorithms can be implemented easily. Sequential versions of the steepest descent, prediction error, and combined steepest descent and recursive least squares algorithm are discussed in detail in Chapters 2 and 3.
Another group of identification methods, derived and discussed in the book, uses polynomial representation of Wiener systems:

- Least squares method for polynomial Wiener systems with the linear term.
- Least squares method for polynomial Wiener systems without the linear term.
- Combined least squares and instrumental variables method for polynomial Wiener systems with the linear term.
- Combined least squares and instrumental variables method for polynomial Wiener systems without the linear term.

– Recursive prediction error method.
– Recursive pseudolinear regression method.

All these polynomial-based identification methods employ a pulse transfer function representation of the system dynamics and polynomial models of the nonlinear element or its inverse. In spite of the fact that polynomial models of both Wiener and Hammerstein systems can be expressed in linear-in-parameters forms, such transformations lead to parameter redundancy as transformed models have a higher number of parameters than the original ones. The number of parameters of transformed models grows strongly with increasing the model order. As a result, the variance error increases and some numerical problems may occur as well. Moreover, as shown in Chapter 4, to transform a Wiener system into the linear-in-parameters form, the nonlinearity have to be invertible, i.e., the nonlinear mapping must be strictly monotonic. It is also shown that the least squares parameter estimates are inconsistent. To obtain consistent parameter estimates, a combined least-squares and instrumental-variables method is proposed. The restrictive assumption of invertibility of the nonlinear element is no more necessary in both the prediction error method and the pseudolinear regression method. In this case, however, the Wiener model is nonlinear in the parameters and the parameter estimation becomes a nonlinear optimization task.
Wiener and Hammerstein models have found numerous industrial applications for system modelling, control, fault detection and isolation. Chapter 6 gives a brief review of applications which includes the following systems and processes:

– pH neutralization process,
– heat exchangers,
– distillation columns,
– chromatographic separation process,
– polymerization reactor,
– quartz microbalance polymer-coated sensors,
– hydraulic plants,
– electro-hydraulic servo-systems,
– pneumatic valves,
– pump-valve systems,
– electrooptical dynamic systems,
– high power amplifiers in satellite communication channels,
– loudspeakers,
– active noise cancellation systems,
– charging process in diesel engines,
– iron oxide pellet cooling process,
– sugar evaporator.

Wiener and Hammerstein models reveal the capability of describing a wide class of different systems and apart from industrial examples, there are many other applications in biology and medicine.

1.1 Models of dynamic systems

This section gives a brief review of discrete-time models of time-invariant dynamic systems. By way of introduction, the section starts with linear model structures and shows nonlinear models as generalizations of linear ones.

1.1.1 Linear models

According to Ljung [112], a time-invariant model can be specified by the impulse response $h(n)$, the spectrum $\Phi(\omega)$ of the additive disturbance $H(q^{-1})\varepsilon(n)$, and the probability density function $f_\varepsilon(\cdot)$ of the disturbance $\varepsilon(n)$. The output $y(n)$ of a discrete-time linear model excited by an input $u(n)$ and disturbed additively by $\varepsilon(n)$ is

$$y(n) = G(q^{-1})u(n) + H(q^{-1})\varepsilon(n), \tag{1.1}$$

where $G(q^{-1})$ is called the input pulse transfer function, $H(q^{-1})$ is the noise pulse transfer function, and q^{-1} denotes the backward shift operator. Unless stated more precisely, we assume $\varepsilon(n)$ to be a zero-mean stationary independent stochastic process. Representing $G(q^{-1})$ and $H(q^{-1})$ as rational functions leads to the general model structure of the following form [112, 149]:

$$A(q^{-1})y(n) = \frac{B(q^{-1})}{F(q^{-1})}u(n) + \frac{C(q^{-1})}{D(q^{-1})}\varepsilon(n), \tag{1.2}$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}, \tag{1.3}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{nb} q^{-nb}, \tag{1.4}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}, \tag{1.5}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + \cdots + d_{nd} q^{-nd}, \tag{1.6}$$

$$F(q^{-1}) = 1 + f_1 q^{-1} + \cdots + f_{nf} q^{-nf}. \tag{1.7}$$

In practice, the structure (1.2) is usually too general. Depending on which of the polynomials (1.3) – (1.7) are used, 32 different model sets can be distinguished. A few commonly used structures, which belong to the general family of structures, are listed below.

Finite impulse response (FIR) structure. The choice of $A(q^{-1}) = C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$ results in the simplest model structure known as the finite impulse response model

$$y(n) = B(q^{-1})u(n) + \varepsilon(n). \tag{1.8}$$

The output of the model (1.8) is a weighted sum of nb past inputs $u(n-1)$, …, $u(n-nb)$:

$$y(n) = b_1 u(n-1) + \cdots + b_{nb} u(n-nb) + \varepsilon(n). \tag{1.9}$$

The optimal one-step-ahead predictor, i.e., the predictor that minimizes the prediction error variance is

$$\hat{y}(n|n-1) = B(q^{-1})u(n) = b_1 u_(n-1) + \cdots + b_{nb} u(n-nb). \tag{1.10}$$

Introducing the parameter vector $\boldsymbol{\theta}$,

$$\boldsymbol{\theta} = \begin{bmatrix} b_1 \dots b_{nb} \end{bmatrix}^T, \tag{1.11}$$

and the regression vector $\mathbf{x}(n)$,

$$\mathbf{x}(n) = \begin{bmatrix} u(n-1) \dots u(n-nb) \end{bmatrix}^T, \tag{1.12}$$

(1.10) can equivalently be expressed in a regression form:

$$\hat{y}(n|n-1) = \mathbf{x}^T(n)\boldsymbol{\theta}. \tag{1.13}$$

The FIR model is able to approximate asymptotically stable dynamic systems quite well if their impulse responses decay reasonably fast.

Autoregressive (AR) structure. The AR model structure is defined with the choice of $B(q^{-1}) = 0$, and $C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$:

$$y(n) = \frac{1}{A(q^{-1})}\varepsilon(n). \tag{1.14}$$

In this case, the parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$ become

$$\boldsymbol{\theta} = \begin{bmatrix} a_1 \dots a_{na} \end{bmatrix}^T, \tag{1.15}$$

$$\mathbf{x}(n) = \begin{bmatrix} -y(n-1) \dots -y(n-na) \end{bmatrix}^T. \tag{1.16}$$

Moving average (MA) structure. The MA model structure corresponds to the choice of $B(q^{-1}) = 0$, and $A(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$:

$$y(n) = C(q^{-1})\varepsilon(n). \tag{1.17}$$

The parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$ are

$$\boldsymbol{\theta} = \begin{bmatrix} c_1 \dots c_{nc} \end{bmatrix}^T, \tag{1.18}$$

$$\mathbf{x}(n) = \begin{bmatrix} \varepsilon(n-1) \dots \varepsilon(n-nc) \end{bmatrix}^T. \tag{1.19}$$

Autoregressive with exogenous input (ARX) structure. The ARX model structure can be obtained with the choice of $C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$:

$$y(n) = \frac{B(q^{-1})}{A(q^{-1})}u(n) + \frac{1}{A(q^{-1})}\varepsilon(n). \tag{1.20}$$

For the ARX model, the parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$ have the forms

$$\boldsymbol{\theta} = \left[a_1 \dots a_{na}\ b_1 \dots b_{nb}\right]^T, \tag{1.21}$$

$$\mathbf{x}(n) = \left[-y(n-1) \dots -y(n-na)\ u(n-1) \dots u(n-nb)\right]^T. \tag{1.22}$$

Autoregressive moving average (ARMA) structure. A combination of the autoregressive model and the moving average model results in the ARMA model. It can be obtained with the choice of $B(q^{-1}) = 0$ and $D(q^{-1}) = F(q^{-1}) = 1$:

$$y(n) = \frac{C(q^{-1})}{A(q^{-1})}\varepsilon(n). \tag{1.23}$$

In this case, the parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$ become

$$\boldsymbol{\theta} = \left[a_1 \dots a_{na}\ c_1 \dots c_{nc}\right]^T, \tag{1.24}$$

$$\mathbf{x}(n) = \left[-y(n-1) \dots -y(n-na)\ \varepsilon(n-1) \dots \varepsilon(n-nc)\right]^T. \tag{1.25}$$

Autoregressive moving average with exogenous input (ARMAX) structure. The ARMAX model is the most general structure of all those considered up to now as it contains all of them as special cases. To obtain the ARMAX model, we choose $D(q^{-1}) = F(q^{-1}) = 1$:

$$y(n) = \frac{B(q^{-1})}{A(q^{-1})}u(n) + \frac{C(q^{-1})}{A(q^{-1})}\varepsilon(n). \tag{1.26}$$

For the ARMAX model, the parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$ are defined as follows:

$$\boldsymbol{\theta} = \left[a_1 \dots a_{na}\ b_1 \dots b_{nb}\ c_1 \dots c_{nc}\right]^T, \tag{1.27}$$

$$\begin{aligned}\mathbf{x}(n) = \big[&-y(n-1) \dots -y(n-na)\ u(n-1) \dots u(n-nb)\\ &\varepsilon(n-1) \dots \varepsilon(n-nc)\big]^T.\end{aligned} \tag{1.28}$$

Output error (OE) structure. The OE model can be obtained if we choose $A(q^{-1}) = C(q^{-1}) = D(q^{-1}) = 1$:

$$y(n) = \frac{B(q^{-1})}{F(q^{-1})}u(n) + \varepsilon(n). \tag{1.29}$$

In this case, the parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$ are defined as

$$\boldsymbol{\theta} = \left[f_1 \dots f_{nf}\ b_1 \dots b_{nb}\right]^T, \tag{1.30}$$

$$\mathbf{x}(n) = \left[-\hat{y}(n-1) \dots -\hat{y}(n-nf)\ u(n-1) \dots u(n-nb)\right]^T, \tag{1.31}$$

where

$$\hat{y}(n) = \frac{B(q^{-1})}{F(q^{-1})}u(n). \tag{1.32}$$

Box-Jenkins (BJ) structure. A structure which is a more general development of the OE model, is called the Box-Jenkins model. To obtain the BJ model, the choice of $A(q^{-1}) = 1$ should be made:

$$y(n) = \frac{B(q^{-1})}{F(q^{-1})}u(n) + \frac{C(q^{-1})}{D(q^{-1})}\varepsilon(n). \tag{1.33}$$

The one-step-ahead predictor for the BJ model has the form [112]:

$$\hat{y}(n|n-1) = \frac{D(q^{-1})}{C(q^{-1})}\frac{B(q^{-1})}{F(q^{-1})}u(n) + \frac{C(q^{-1}) - D(q^{-1})}{C(q^{-1})}y(n). \tag{1.34}$$

1.1.2 Nonlinear models

Nonlinear counterparts of linear model structures can be defined assuming that there is a nonlinear relationship between the actual system output and past system inputs, the past system or model outputs, and the actual and past additive disturbances. For nonlinear input-output models we have

$$y(n) = g\big(\mathbf{x}(n), \boldsymbol{\theta}\big) + \varepsilon(n) \tag{1.35}$$

or, in the predictor form,

$$\hat{y}(n|n-1) = g\big(\mathbf{x}(n), \boldsymbol{\theta}\big). \tag{1.36}$$

Depending on the form of $\mathbf{x}(n)$, nonlinear model structures known as NFIR, NAR, NMA, NARMA, NARX, NARMAX, NOE, NBJ can be defined. The function $g(\cdot)$ is a nonlinear mapping, which for any given $\boldsymbol{\theta}$ maps $\mathbb{R}^d$ to $\mathbb{R}^p$, where d is the number of regressors (elements of the vector $\mathbf{x}(n)$) and p is the number of model outputs. In a parametric approach, $g(\cdot)$ is expressed by a function expansion:

$$g\big(\mathbf{x}(n), \boldsymbol{\theta}\big) = \sum_{m=1}^{ng} \beta_m g_m\big(\mathbf{x}(n)\big), \tag{1.37}$$

where $g_m(\cdot)$ is called the basis function, and $\boldsymbol{\theta} = [\beta_1 \dots \beta_{ng}]^T$. There are different forms of function expansions, used for nonlinear systems representation, which are based on polynomials, Volterra kernels, Fourier series, piecewise constant functions, radial basis functions, wavelets, kernel estimators, neural networks, and fuzzy models [112].

Discrete-time Volterra models. If the function $g(\cdot)$ is analytic, then the system response can be represented by the Volterra series [109]:

$$y(n) = \sum_{i_1=1}^{n} h_1(i_1)u(n-i_1) + \sum_{i_1=1}^{n}\sum_{i_2=1}^{n} h_2(i_1,i_1)u(n-i_1)u(n-i_2) + \cdots + \varepsilon(n), \tag{1.38}$$

where the kernel functions $h_j(i_1,\ldots,i_j)$, $j = 1,2,\ldots$, describe system dynamics. Two basic problems associated with practical application of Volterra series are difficulty concerning the measurement of Volterra kernel functions and the convergence of Volterra series [145]. The Volterra series representation is really a Taylor series with memory. Therefore, the problem of convergence is the same as that of Taylor series representation of a function, i.e., the Volterra series representation of a system may converge for only a limited range of the system input amplitude. To circumvent this problem, Wiener formed a new set of functionals from the Volterra functionals – G-funtionals which have an orthogonal property when their input is a Gaussian process.
Extensive studies of Volterra models show their successful application to low order systems. The reasons for this system order limitation are practical difficulties in extending kernel estimation to orders higher than the third [117].

Kolmogorov-Gabor models. The application of generalized polynomial models to the representation of nonlinear dynamic systems

$$y(n) = f\big(u(n-1),\ldots,u(n-nb),y(n-1),\ldots,y(n-na)\big) + \varepsilon(n) \tag{1.39}$$

results in Kolmogorov-Gabor models:

$$\begin{aligned} y(n) = a_0 &+ \sum_{i_1=1}^{nab} a_{i_1}x_{i_1}(n) + \sum_{i_1=1}^{nab}\sum_{i_2=1}^{i_1} a_{i_1i_2}x_{i_1}(n)x_{i_2}(n) \\ &+ \cdots + \sum_{i_1=1}^{nab}\sum_{i_2=1}^{i_1}\cdots\sum_{i_l=1}^{i_{l-1}} a_{i_1i_2\ldots i_l}x_{i_1}(n)x_{i_2}(n)\ldots x_{i_l}(n) + \varepsilon(n), \end{aligned} \tag{1.40}$$

where $nab = na + nb$,

$$x_j(n) = \begin{cases} u(n-j) & \text{if } \ 1 \leqslant j \leqslant nb \\ y(n-j+nb) & \text{if } \ nb < j \leqslant nab. \end{cases} \tag{1.41}$$

The number of parameters M in (1.40) increases strongly as nab or l grow [123]:

$$M = \frac{(l+nab)!}{l!nab!}. \tag{1.42}$$

The large model complexity of the Kolmogorov-Gabor model restricts its practical applicability and leads to reduced polynomial models containing only selected terms of (1.40). Note that (1.40) has the NARX structure. The other nonlinear structures such as NFIR, NAR, NMA, NARMA, NARX, NARMAX, NBJ can be obtained via a relevant redefinition of $x_j(n)$.

Nonlinear orthonormal basis function models (NOBF). The main disadvantage of both the FIR and NFIR models is that many parameters may be needed to describe a system adequately if its impulse response decays slowly. This disadvantage can be reduced by introducing linear filters which incorporate prior knowledge about process dynamics. Orthonormal Laguerre and Kautz filters, which have orthonormal impulse responses, are commonly applied. The Laguerre filter can be described with only one parameter α, a real pole

$$L_k(q) = \frac{1}{q-\alpha}\left(\frac{1-\alpha q}{q-\alpha}\right)^{k-1}. \tag{1.43}$$

Therefore, this kind of filters is suitable for modelling systems with well-damped behavior. For systems with resonant behavior, Kautz filters are suitable as they have a complex pole pair. In practice, estimates of a system dominant pole or dominant conjugate complex poles are used. The regression vector for the NOBF model has the form

$$\mathbf{x}(n) = \left[L_1(q)\ L_2(q)\ \ldots\ L_r(q)\right]^T. \tag{1.44}$$

1.1.3 Series-parallel and parallel models

Models of dynamic systems can be used in two basic configurations: a prediction configuration or a simulation configuration. The prediction configuration permits the prediction of future system outputs based on past system inputs and outputs. Examples of the prediction configuration are the FIR, AR, ARX models in the linear case, and the NFIR, NOBF, NAR, NARX models in the nonlinear case. In the simulation configuration, future system outputs are also predicted but only on the basis of past system inputs without employing past system outputs. The OE, BJ, and NOE, NBJ models are examples of the simulation configuration. In the system identification literature, the one-step prediction configuration is called a series-parallel model and the simulation configuration is called a parallel model [121, 123]. Parallel models, being dynamic systems themselves, are also called recursive models as their mathematical description has the form of difference equations. In contrast to parallel models, series-parallel models are described by algebraical equations. In the context of neural networks, these models are called feedforward ones. In the identification process, model parameters are calculated in such a way so as to minimize a chosen cost function dependent on the identification error $e(n)$. The two model configurations above entail two different definitions of the identification error. For the series-parallel model, the identification error is called the equation error, for the parallel model – the output error.

1.1.4 State space models

The extension of a linear state space model to the nonlinear case results in the following nonlinear state space model:

$$\mathbf{x}(n+1) = \mathbf{h}\big(\mathbf{x}(n), \mathbf{u}(n)\big) + \boldsymbol{\upsilon}(n), \tag{1.45}$$

$$\mathbf{y}(n) = \mathbf{g}\big(\mathbf{x}(n)\big) + \boldsymbol{\varepsilon}(n), \tag{1.46}$$

where $\mathbf{x}(n) \in \mathbb{R}^{nx}$, $\mathbf{u}(n) \in \mathbb{R}^{nu}$, $\boldsymbol{\upsilon}(n) \in \mathbb{R}^{nx}$, $\mathbf{y}(n) \in \mathbb{R}^{ny}$, $\boldsymbol{\varepsilon}(n) \in \mathbb{R}^{ny}$. If the system state $\mathbf{x}(n)$ is available for measurement, the identification is equivalent to the determination of the vector functions $\mathbf{h}(\cdot)$ and $\mathbf{g}(\cdot)$:

$$\hat{\mathbf{x}}(n+1) = \mathbf{h}\big(\mathbf{x}(n), \mathbf{u}(n)\big), \tag{1.47}$$

$$\hat{\mathbf{y}}(n) = \mathbf{g}\big(\mathbf{x}(n)\big). \tag{1.48}$$

The model (1.47), (1.48) is of the series-paralel type. In practice, at least some of state variables are unknown and they have to be estimated. If no state variables are measured, simultaneous estimation of system states and the determination of the functions $\mathbf{h}(\cdot)$ and $\mathbf{g}(\cdot)$ is required. The high complexity of such a task is the main reason for the dominance of the much simpler input-output approaches [123].

1.1.5 Nonlinear models composed of sub-models

Except for artificial "phenomena" created by mathematical equations, models are always different from modelled phenomena [131]. Mathematical models are simply equations which are derived on the basis of the first principles and/or the experiment data and they are different from the underlaying phenomena of a nonmathematical nature. In general, models can be characterized by their time resolution and granularity where granularity specifies the level of details included into the model.

Up to now, we have considered models which are classified as black box models [148]. These models are based on the measurement data only, i.e., their parameters and structure are determined through experiments. Typically, the parameters of black box models have no interpretation in terms of physical, chemical, biological, economical, and other laws.

Another class of models are white box models as they are completely derived from the underlying principal laws. Even if some of the parameters are estimated from data, a model is included into this class as well. In contrast to black box models, the parameters of white box models have a clear interpretation [94].

Gray box models combine features of the white and black box models. They are derived both on the basis of the underlying laws and the measurement data. For example, the system structure can be determined utilizing a priori knowledge about the system nature, and system parameters can be estimated from data.

Nonlinear models of a given internal structure composed of sub-models, referred to as also block-oriented models, are members of the class of gray box models. Wiener and Hammerstein models, shown in Figs 1.1 – 1.2, are two

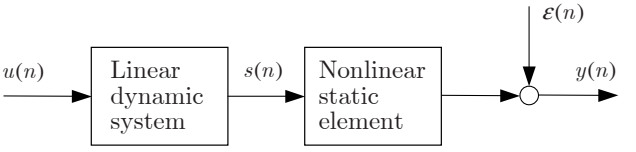

Fig. 1.1. SISO Wiener system

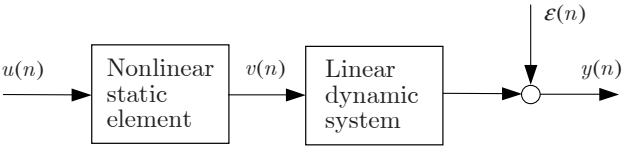

Fig. 1.2. SISO Hammerstein system

well-known examples of such models, composed of sub-models [61, 108, 109]. Both of them contain a linear dynamic system and a nonlinear static element in a cascade. While in the Wiener system the nonlinear element follows the linear dynamic system, in the Hammerstein model, both of these sub-models are connected in reverse order. The Wiener model is given by

$$y(n) = f\left(\frac{B(q^{-1})}{A(q^{-1})}u(n)\right) + \varepsilon(n), \tag{1.49}$$

where $f(\cdot)$ denotes the nonlinear function describing the nonlinear element, and $B(q^{-1})/A(q^{-1})$ is the pulse transfer function of the linear dynamic system. With the same notation, the Hammerstein model can be expressed as

$$y(n) = \frac{B(q^{-1})}{A(q^{-1})}f\big(u(n)\big) + \varepsilon(n). \tag{1.50}$$

The multi-input multi-output (MIMO) Wiener model can be described by the following equation:

$$\mathbf{y}(n) = \mathbf{f}\big(\mathbf{s}(n)\big) + \boldsymbol{\varepsilon}(n), \tag{1.51}$$

where $\mathbf{f}(\cdot) : \mathbb{R}^{ns} \to \mathbb{R}^{ny}$ is a nonzero vector function, $\mathbf{y}(n) \in \mathbb{R}^{ny}$, $\boldsymbol{\varepsilon}(n) \in \mathbb{R}^{ny}$. The output $\mathbf{s}(n)$, $\mathbf{s}(n) \in \mathbb{R}^{ns}$, of the MIMO linear dynamic system is

$$\mathbf{s}(n) = -\sum_{m=1}^{na} \mathbf{A}^{(m)}\mathbf{s}(n-m) + \sum_{m=1}^{nb} \mathbf{B}^{(m)}\mathbf{u}(n-m), \tag{1.52}$$

where $\mathbf{u}(n) \in \mathbb{R}^{nu}$, $\mathbf{A}^{(m)} \in \mathbb{R}^{ns\times ns}$, $\mathbf{B}^{(m)} \in \mathbb{R}^{ns\times nu}$. In a similar way, the MIMO Hammerstein model can be obtained by connecting a MIMO linear dynamic system with a MIMO static nonlinear element – Fig. 1.4. The output of the MIMO Hammerstein model is

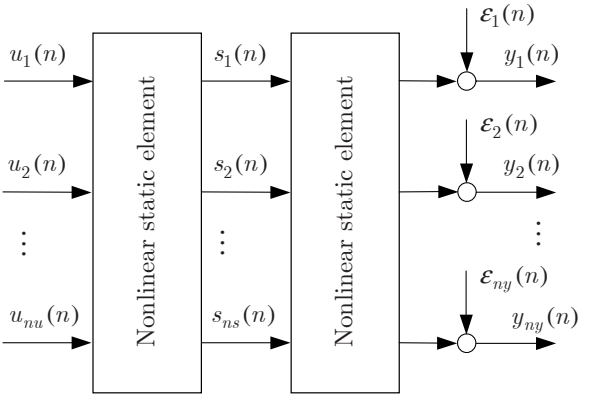

Fig. 1.3. MIMO Wiener system

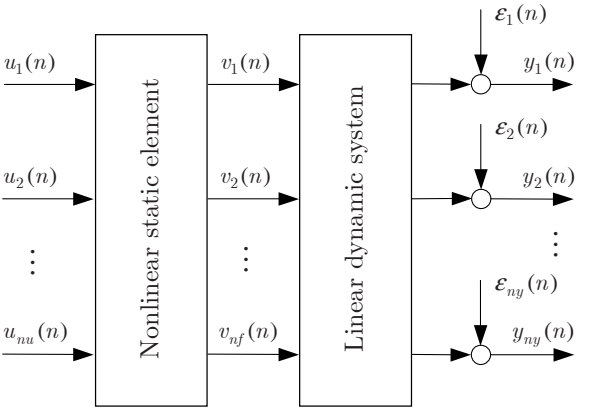

Fig. 1.4. MIMO Hammerstein system

$$\mathbf{y}(n) = -\sum_{m=1}^{na} \mathbf{A}^{(m)}\mathbf{y}(n-m) + \sum_{m=1}^{nb} \mathbf{B}^{(m)}\mathbf{f}\big(\mathbf{u}(n-m)\big) + \boldsymbol{\varepsilon}(n), \quad (1.53)$$

where $\mathbf{A}^{(m)} \in \mathbb{R}^{ny\times ny}$, $\mathbf{B}^{(m)} \in \mathbb{R}^{ny\times nf}$, $\mathbf{f}(\cdot) : \mathbb{R}^{nu} \to \mathbb{R}^{nf}$ is a nonzero vector function, $\mathbf{u}(n) \in \mathbb{R}^{nu}$.

Note that the definitions (1.51), (1.52), and (1.53) describe models with a coupled static part and coupled dynamics. We can also consider models with an uncoupled static part and coupled dynamics or a coupled static part and uncoupled dynamics as special cases of these general forms.

The general Wiener model considered by Sieben [147] is a SISO (single-input single-output) model in which a nonlinear static MISO (multi-input single-output) element follows a linear SIMO (single-input multiple-output) dynamic

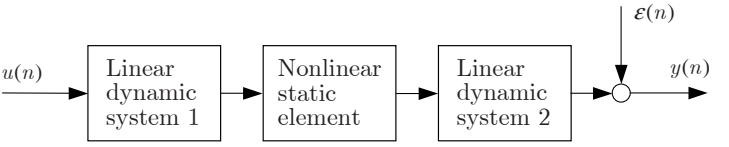

Fig. 1.5. SISO Wiener-Hammerstein system

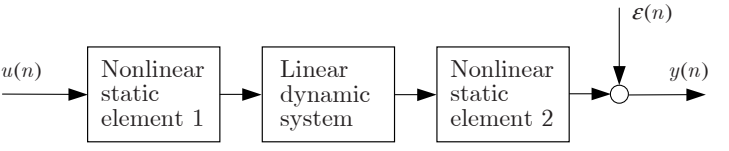

Fig. 1.6. SISO Hammerstein-Wiener system

system. Another structure, known as the Uryson model [40], consists of several Hammerstein models in parallel, each path having the same input and with several outputs summed.

More complicated structures arise through the interconnection of three submodels in a cascade. In this way, structures called Wiener-Hammerstein (Fig. 1.5) and Hammerstein-Wiener models (Fig. 1.6) can be obtained. The Wiener-Hammerstein structure is given by

$$y(n) = \frac{B_2(q^{-1})}{A_2(q^{-1})} f\left[\frac{B_1(q^{-1})}{A_1(q^{-1})} u(n)\right] + \varepsilon(n), \tag{1.54}$$

where $B_1(q^{-1})/A_1(q^{-1})$ and $B_2(q^{-1})/A_2(q^{-1})$ are pulse transfer functions of the first and the second linear dynamic system, respectively. The identification of Wiener-Hammerstein systems with correlation methods was studied by Billings and Fakhouri [17] and Hunter and Korenberg [71]. A recursive identification method for the MISO Wiener-Hammerstein model was proposed by Boutayeb and Darouach [21]. Bloemen *et al.* [19] considered the application of Hammerstein-Wiener models to the predictive control problem.

Bai [7] developed a two-stage identification algorithm for Hammerstein-Wiener systems in which optimal parameter estimates of both the nonlinear elements and the linear dynamic system are obtained using the RLS algorithm followed by singular value decomposition of two matrices. The algorithm is convergent in the absence of noise and convergent with the probability 1 in the presence of white noise.

Recently, the identification of Hammerstein-Wiener systems was also studied by Bai [9]. In the Hammerstein-Wiener model, two nonlinear blocks, described by the functions $f_1(\cdot)$ and $f_1(\cdot)$, are separated by a linear dynamic system:

$$y(n) = f_2\left[\frac{B(q^{-1})}{A(q^{-1})} f_1\big(u(n)\big)\right] + \varepsilon(n). \tag{1.55}$$

1.1.6 State-space Wiener models

A state-space description of MIMO Wiener systems with nu inputs, ny outputs, nx states variables, and ns internal variables has the form [113, 158, 164]

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{u}(n), \tag{1.56}$$

$$\mathbf{s}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\mathbf{u}(n), \tag{1.57}$$

$$\mathbf{y}(n) = \mathbf{f}\big(\mathbf{s}(n)\big) + \boldsymbol{\varepsilon}(n), \tag{1.58}$$

where $\mathbf{x}(n) \in \mathbb{R}^{nx}$, $\mathbf{u}(n) \in \mathbb{R}^{nu}$, $\mathbf{s}(n) \in \mathbb{R}^{ns}$, $\mathbf{y}(n) \in \mathbb{R}^{ny}$, and $\mathbf{f}(\cdot) : \mathbb{R}^{nf} \to \mathbb{R}^{ny}$ is a nonzero vector function, $\boldsymbol{\varepsilon}(n) \in \mathbb{R}^{ny}$ is a zero-mean stochastic process. The parallel representation of the state-space Wiener model has the form

$$\hat{\mathbf{x}}(n+1) = \mathbf{A}\hat{\mathbf{x}}(n) + \mathbf{B}\mathbf{u}(n), \tag{1.59}$$

$$\hat{\mathbf{s}}(n) = \mathbf{C}\hat{\mathbf{x}}(n) + \mathbf{D}\mathbf{u}(n), \tag{1.60}$$

$$\hat{\mathbf{y}}(n) = \mathbf{f}\big(\hat{\mathbf{s}}(n)\big). \tag{1.61}$$

Replacing the model state $\hat{\mathbf{x}}(n)$ in (1.59) and (1.60) with the system state $\mathbf{x}(n)$ results in the series-parallel representation of the state-space Wiener model:

$$\hat{\mathbf{x}}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{u}(n), \tag{1.62}$$

$$\hat{\mathbf{s}}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\mathbf{u}(n), \tag{1.63}$$

$$\hat{\mathbf{y}}(n) = \mathbf{f}\big(\hat{\mathbf{s}}(n)\big). \tag{1.64}$$

1.1.7 State-space Hammerstein models

A state space MIMO Hammerstein model can be described by the following equations [157]:

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{v}(n), \tag{1.65}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\mathbf{v}(n) + \boldsymbol{\varepsilon}(n), \tag{1.66}$$

$$\mathbf{v}(n) = \mathbf{f}\big(\mathbf{u}(n)\big), \tag{1.67}$$

where $\mathbf{x}(n) \in \mathbb{R}^{nx}$, $\mathbf{u}(n) \in \mathbb{R}^{nu}$, $\mathbf{y}(n) \in \mathbb{R}^{ny}$, $\boldsymbol{\varepsilon}(n) \in \mathbb{R}^{ny}$ is a zero-mean stochastic process, $\mathbf{v}(n) \in \mathbb{R}^{nf}$, and $\mathbf{f}(\cdot)$: $\mathbf{f}(\cdot) : \mathbb{R}^{nu} \to \mathbb{R}^{nf}$ is a nonzero vector function. The MIMO Hammerstein model in a parallel state-space representation has the form

$$\hat{\mathbf{x}}(n+1) = \mathbf{A}\hat{\mathbf{x}}(n) + \mathbf{B}\hat{\mathbf{v}}(n), \tag{1.68}$$

$$\hat{\mathbf{y}}(n) = \mathbf{C}\hat{\mathbf{x}}(n) + \mathbf{D}\hat{\mathbf{v}}(n), \tag{1.69}$$

$$\hat{\mathbf{v}}(n) = \mathbf{f}\big(\mathbf{u}(n)\big). \tag{1.70}$$

Replacing the model state $\hat{\mathbf{x}}(n)$ in (1.68) and (1.69) with the system state $\mathbf{x}(n)$, we obtain the series-parallel representation of the state-space Hammerstein model:

$$\hat{\mathbf{x}}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\hat{\mathbf{v}}(n), \tag{1.71}$$

$$\hat{\mathbf{y}}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\hat{\mathbf{v}}(n), \tag{1.72}$$

$$\hat{\mathbf{v}}(n) = \mathbf{f}\big(\mathbf{u}(n)\big). \tag{1.73}$$

1.2 Multilayer perceptron

Multilayer feedforward neural networks, also referred to as multilayer perceptrons, are the most widely known and used neural networks [60, 68, 69, 127, 144, 169]. In the multilayer perceptron (MLP), the neurons are ordered into one or more hidden layers and connected to an output layer. This type of neural network is used in Chapters 2 and 3 intensively for modelling both the nonlinear element and its inverse.

1.2.1 MLP architecture

The ith output $x_i(n)$ of the first hidden layer (Fig. 1.7) is

$$x_i(n) = \varphi\left(\sum_{j=1}^{nu} w_{ij}^{(1)} u_j(n) + w_{i0}^{(1)}\right), \tag{1.74}$$

where $w_{ij}^{(1)}$ is the jth weight of the ith neuron, $w_{i0}^{(1)}$ is the bias of the ith neuron, $\varphi(\cdot)$ is the activation function of hidden layer neurons, nu is the number of inputs, and $u_j(n)$ is the jth input. Common choices of the activation function are sigmoidal functions such as the logistic function

$$\varphi(x) = \frac{1}{1+\exp(-x)} \tag{1.75}$$

and the hyperbolic tangent function

$$\varphi(x) = \tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}. \tag{1.76}$$

The outputs of the first hidden layer can be connected to the successive hidden layers and finally to the output layer. In the commonly used MLP neural network with one hidden layer, the outputs $x_i(n)$, $i = 1, \ldots, M$, where M is the number of hidden layer neurons, are transformed by the output layer into the outputs $y_k(n)$:

$$y_k(n) = \phi\left(\sum_{i=1}^{M} w_{ki}^{(2)} x_i(n) + w_{k0}^{(2)}\right), \tag{1.77}$$

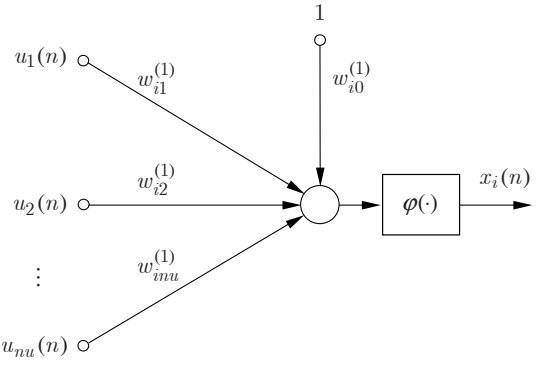

Fig. 1.7. The ith hidden layer neuron

where $w_{ki}^{(2)}$ is the ith weight of the kth output neuron, $w_{k0}^{(2)}$ is the bias of the kth output neuron, and $\phi(\cdot)$ is the activation function of output layer neurons. Although $\phi(\cdot)$ can be a nonlinear function, the linear activation function $\phi(x) = x$ is a typical choice.

1.2.2 Learning algorithms

Nonlinear optimization of neural network weights is the most common technique used for training the MLP. Using gradient-based learning methods, the cost function J, typically the sum of squared errors between the system output and the neural network model output, is minimized. As a result, neural network weights are adjusted along the negative gradient of the cost function. The backpropagation (BP) learning algorithm is an implementation of the gradient descent optimization method for weight updating. The backpropagation learning algorithm uses the backpropagation algorithm as a technique for the computation of the gradient of the MLP w.r.t. its weights [68, 123, 132]. In spite of its computational simplicity, training the MLP with the BP learning algorithm may cause several problems such as very slow convergence, oscillations, divergence, and the "zigzagging" effect. A large number of improvements, extensions, and modifications of the basic BP learning algorithm have been developed to circumvent this problem [60, 68, 123].
The reason for the slow convergence of the BP learning algorithm is that it operates on the basis of a linear approximation of the cost function. To achieve a significantly higher convergence rate, higher order approximations of the cost function should be used. Examples of such learning techniques are the Levenberg-Marquard method, quasi-Newton methods, conjugate gradient methods [68, 123], and the RLS learning algorithms [144].
To extract information from the training data and increase the the effectiveness of learning, some data preprocessing such as filtering, removing redundancy, and removing outliers is usually necessary.

Also, scaling the data is essential to make learning algorithms robust and decrease the learning time. The recommended scaling technique is based on removing the mean and scaling signals to the same variance [127]. Alternatively, the mean can be removed from signals and zero-mean signals scaled with respect to their maximum absolute values, to obtain values in a specified interval, e.g. $[-1, 1]$.
In general, to minimize the learning time, applying nonzero-mean input signals should be avoided. This comes from the fact that the learning time for the steepest descent algorithm is sensitive to variations in the condition number $\lambda_{max}/\lambda_{min}$, where λ_{max} is the largest eigenvalue of the Hessjan of the cost function and λ_{min} is its smallest nonzero eigenvalue. Experimental results, show that for nonzero-mean input signals the condition number $\lambda_{max}/\lambda_{min}$ is larger than for zero-mean input signals [68]. Note that the choice of asymmetric activation function, e.g. the logistic function, introduces systematic bias for hidden layer neurons. This has a similar effect on the condition number $\lambda_{max}/\lambda_{min}$ as nonzero-mean inputs. Therefore, to increase the convergence speed of gradient-based learning algorithms, the choice of antisymmetric activation functions, such as the hyperbolic tangent, is recommended.

1.2.3 Optimizing the model architecture

In practice, MLP models of real processes are of a rather large size. It is well known that models should not be too complex because they would learn noise and thus generalize badly to new data. On the other hand, they should not be too simple because they would not be capable to capture the process behavior. In the MLP with one hidden layer, the problem of architecture optimizing boils to choosing the number of hidden layer nodes and eliminating insignificant weights.
The overall model error is composed of two components – a bias error which express the systematic error caused by the restricted model flexibility and a variance error, being the stochastic error due to the restricted accuracy of parameter estimates. Both these components of the model error are in conflict, known as a bias/variance dilemma, because the bias error decreases and the variance error increases for growing model complexity. Therefore it is necessary to find a compromise – the so called bias/error tradeoff. To accomplish this, the optimization of the model architecture is necessary.
There are two groups of methods used to optimize the neural network architecture, known as network growing and network pruning. In network growing methods, new nodes or layers are added starting from a small size network until the enlarged structure meets the assumed requirements. A well-known example of a growing method is the cascade-correlation algorithm [36]. In pruning methods, the initial structure is large, then we prune it by weakening or eliminating some selected weights.
The idea of pruning is based on the assumption that there is a large amount of redundant information stored in a fully connected MLP. Network pruning

is commonly accomplished by two approaches – one based on complexity regularization and the other based on removing some weights using information on second-order derivatives of the cost function.
In complexity regularization methods, the complexity penalty term is added to the cost function. The standard cost function in back-propagation learning is the mean-square error. Depending on the form of the complexity penalty term different regularization techniques can be defined such as the weight decay, the weight elimination, the approximate smoother, the Chauvin's penalty approach [60, 123]. In fact, regularization methods do not change the model structure but reduce the model flexibility by keeping some weights at their initial values or constraining their values. In this way the reduction of the variance error can be achieved at the price of the bias error.
The optimal brain damage (OBD) [107] and the optimal brain surgeon (OBS) [67] are the most widely known an used methods based on the use of information on second-order derivatives of the cost function. Both of them are used for reducing the size of the network by selectively deleting the weights. Both of them employ the second-order Taylor expansion of the cost function about the operating point – the nw-dimensional weight vector $\mathbf{w}^* = [w_1^* \dots w_{nw}^*]^T$ for which the cost function has a local minimum. Their objective is to find a set of weights whose removing cause the least change of the cost function. To achieve reasonable low computational complexity, only diagonal terms of the second-order Taylor expansion are included into the definition of the saliency of parameters in the OBD method. This corresponds to the assumption that the Hessjan matrix is diagonal matrix. The OBD method is an iterative procedure of the following form:

1. Train the MLP to minimum mean-square error of the cost function.
2. Compute the diagonal second-order derivatives h_{ii}, $i = 1, \dots, nw$, of the cost function.
3. Compute the saliencies for weights: $S_i = h_{ii}(w_i^*)^2/2$.
4. Delete some weights that have small saliencies.
5. Return to Step 1.

No such assumption about the Hessjan matrix is made in the OBS method and the OBD can be considered as a special case of the OBS.

1.3 Identification of Wiener systems

Many different approaches to Wiener system identification have been proposed based on correlation analysis, linear optimization, nonparametric regression, nonlinear optimization with different nonlinear models such as polynomials, neural networks, wavelets, orthogonal functions, and fuzzy sets models.

Correlation methods. Billings and Fakhouri used a correlation analysis approach to the identification of block-oriented systems based on the theory

of separable processes [14]. When the input is a white Gaussian signal, it is possible to separate the identification of the linear dynamic system from the identification of the nonlinear element [15, 17]. For Wiener systems, the first order correlation function $R_{uy}(k)$ of the system input $u(n)$ and the system output $y(n)$ is directly proportional to the impulse response $h(k)$ of the linear system, and the second order correlation function $R_{u^2y}(k)$ is directly proportional to the square of $h(k)$. Therefore, if $R^2_{uy}(k)$ and $R_{u^2y}(k)$ are equal except for a constant of proportionality, the system has a Wiener-type structure.
A correlation approach to the identification of direction-dependent dynamic systems using Wiener models was used by Barker *et al.* [12]. They considered Wiener systems containing the linear dynamic part with different transfer functions for increasing and decreasing system output. The Wiener system was excited with maximum-length pseudo-random or inverse maximum-length pseudo-random binary signals. The determination of Wiener model parameters was performed by matching the system and model correlation functions, outputs, and discrete Fourier transforms of the outputs.

Linear optimization methods. In linear optimization methods, it is assumed that a model can be parameterized by a finite set of parameters. The nonlinear element is commonly modelled by a polynomial along with a pulse transfer function model of the linear dynamic system. A Wiener model parameterized in this way is nonlinear in the parameters and parameter estimation becomes a nonlinear optimization problem. Note that the inversion of the Wiener model is a Hammerstein model whose linear part is described by the inverse transfer function and nonlinear element is described by the inverse nonlinear function. A necessary condition for such a transformation is the invertibility of the function $f(\cdot)$ describing the nonlinear part of the Wiener model. To obtain an asymptotically stable inverse model, the Wiener model should be minimum phase. The inverse Wiener model is still nonlinear in the parameters but it is much more convenient for parameter estimation as it can be transformed into the linear-in-parameters MISO form with the method proposed by Chang and Luus [27] for Hammerstein systems. The parameters of the transformed model can be calculated with the least squares method. Assuming the knowledge of $f(\cdot)$, the identification of the inverse Wiener system was considered by Pearson and Pottman [138]. To overcome the practical difficulty related to the fact that the inverse system identification approach penalizes prediction errors of the input signal $u(n)$ instead of the output signal $y(n)$, they used the weighted least squares method with weighting parameters that weight the relative importance of the error.
A polynomial inverse model of the nonlinear element and a frequency sampling filter model of the linear dynamic system

$$\hat{s}(n) = \sum_{m=0}^{n_f-1} G(jw_m)F_m(n), \tag{1.78}$$

where $\hat{s}(n)$ is the output of the linear dynamic model, $G(jw_m)$, $m = 0, \ldots, n_f - 1$, is the discrete frequency response of the linear system at $w_m = 2\pi m/n_f$, $F_m(n)$ is the output of the mth frequency sampling filter defined as

$$F_m(n) = \frac{1}{n_f} \frac{1 - q^{-n_f}}{1 - e^{j(2\pi m/n_f)} q^{-1}} u(n), \tag{1.79}$$

where $u(n)$ is the system input and q^{-1} is the backward shift operator, was used by Kalafatis *et al.* [95]. The model output $\hat{y}(n)$ can then be expressed in the linear-in-parameters form:

$$\hat{y}(n) = \sum_{m=0}^{n_f - 1} G(jw_m) F_m(n) - \sum_{k=2}^{r} \gamma_k y^k(n), \tag{1.80}$$

where $y(n)$ is the system output and $\gamma_k, k = 2, \ldots, r$, are the parameters of the polynomial inverse model of the nonlinear element. The parameters $G(jw_m)$ and γ_k can be calculated with the least squares method. Unfortunately, such an approach leads to inconsistent parameter estimates for Wiener systems with additive output disturbances as the regression vector is correlated with the disturbance. To overcome this problem, an iterative algorithm was proposed [96], which consists of the following three steps: First, parameter estimates are calculated using the least squares method. Then using the obtained estimates, predicted system outputs are calculated. Finally, to calculate corrected parameter estimates, the predicted system outputs are employed in another estimation step using the least squares method. The above estimation procedure is repeated until the parameter estimates converge to constant values.

The recursive least squares scheme is used in the orthonormal basis function-based identification method proposed by Marciak *et al.* [116]. They use a noninverted model of the linear system composed of Laguerre filters and an inverse polynomial model of the nonlinear element.

In the frequency approach to the identification of Wiener systems of Bai [10], the phase estimate of the linear dynamic system output is determined based on the discrete Fourier transform (DFT) of the filtered system output. Having the phase estimate, the structure of nonlinearity can be determined from the graph of the system output versus an estimated linear dynamic system output and approximated by a polynomial.

The identification of Wiener systems based on a modified series-parallel Wiener model, defined by a non-inverted pulse transfer model of the linear element and an inverse polynomial model of the nonlinear element, can be performed with the method proposed by Janczak [80, 83]. The modified series-parallel model is linear in the parameters and its parameters are calculated with the least squares method. The method requires the nonlinear function $f(\cdot)$ to be invertible and the linear term of the polynomial model to be nonzero. However, in the case of additive output noise, direct application of this approach results in inconsistent parameter estimates. As a remedy against such a situation, a combined least squares and instrumental variables estimation procedure is

proposed. A detailed description of this method, along with its extension to the identification of Wiener systems without the linear term of the nonlinear characteristic, is given in Chapter 4.
Parameter estimation of Wiener systems composed of a finite impulse response (FIR) model of the linear system and an inverse polynomial model of the nonlinear element was considered by Mzyk [119]. To obtain consistent parameter estimates, the instrumental variables method is employed with instrumental variables defined as a sum of powered input signals and some tuning constants selected by the user.
The identification of MIMO Wiener systems with the use of basis functions for the representation of both the linear dynamic system and the nonlinear element was proposed by Gómez and Baeyens [44]. It is assumed that the nonlinear element is invertible and can be described by nonlinear basis functions. The MIMO linear dynamic system is represented by rational orthonormal bases with fixed poles (OBFP). Special cases of OBFP are the FIR, Laguerre, and Kautz bases. Under the above assumptions, the MIMO Wiener model can be transformed into the linear-in-parameters form and parameter estimates of the transformed model that minimize the quadratic cost function on prediction errors are calculated using the least squares method. To calculate matrix parameters of the nonlinear element from the parameter estimates of the transformed model, the singular value decomposition technique is applied.

Nonparametric regression methods. Parametric regression methods are based on a restrictive assumption concerning the class of nonlinear functions. The kernel nonparametric regression approach, which considerably enlarges the class of nonlinearities identified in Wiener systems, was introduced by Greblicki [46] and studied further in [48]. Next, the idea of nonparametric regression was advanced by employing orthogonal series for recovering the inverse of the nonlinear characteristic [47]. For nonparametric regression algorithms that use trigonometric, Legendre and Hermite orthogonal functions, pointwise consistency was shown and the rates of convergence were given. Greblicki also proposed and analyzed recursive identification algorithms based on the kernel regression for both discrete-time [51] and continuous-time Wiener systems [50, 52].

Nonlinear optimization methods. The identification of Wiener systems using prediction error methods was discussed in [85, 88, 128, 165, 166]. Wigren [165] analyzed recursive Gauss-Newton and stochastic gradient identification algorithms assuming that system nonlinearity is known a priori. He established conditions for local and global convergence of parameter estimates to the true system parameters at correlated measurement disturbances. Another recursive prediction error method, proposed by Wigren [166], estimates the parameters of a pulse transfer function model of the linear system and a piecewise linear model of the nonlinear element. With the technique of a linearized differential equation, the local convergence of estimates to the system para-

meters is proved. It is also shown that the input signal should be such that there is energy in the whole range of piecewise linear approximation.

Vörös [162] used a parametric model to describe Wiener systems with a special kind of discontinuous nonlinear element – a piecewise-linear function with a preload and a dead zone. The pure preload, dead zone, and a two-segment piecewise-linear asymmetric nonlinearities are special cases of the general discontinuous nonlinearity. He proposed an identification method that uses a description of the nonlinear element based on the key separation principle. Such a formulation of the problem makes it possible to transform the nonlinear element model into the pseudolinear-in-parameters form. Parameter estimation is conducted iteratively as the model contains three internal variables that are unmeasurable. The iterative procedure is based on the use of parameter estimates from the preceding step to estimate these unmeasurable internal variables.

Another approach to the identification of Wiener systems with the assumed forms of hard-type nonlinearity parameterized by a single unknown parameter a was proposed by Bai [8]. Examples of nonlinearities of such type are the saturation, preload, relay, dead zone, hysteresis-relay, and the hysteresis. To find the unknown parameter a, the separable least squares method is used, in which the identification problem is transformed into a one-dimensional minimization problem. As the cost function is one-dimensional, global search methods can be applied to find its minimum. Alternatively, it is also possible to find the minimum directly from the plot of the cost function versus a. Having found the optimal estimate of a, the parameters of the linear dynamic system can be estimated using the least squares method. For this approach, the conditions under which the strong consistency of parameter estimates can be achieved are given. Although the separable least squares method can be extended to the two-dimensional case easily, its extension to the case of nonlinearities parameterized by a larger number of parameters is more complicated.

An iterative scheme for the identification of Wiener systems with the prediction error method was proposed by Norquay *et al.* [128]. In this approach, the nonlinear element is modelled by a polynomial. To calculate the approximated Hessjan, the Levenberg-Marquardt method is used along with the calculation of the gradient via the simulation of sensitivity models. A recursive version of this method was used in [85, 88].

The pseudolinear regression algorithm, described by Janczak [86, 88], can be obtained from the prediction error scheme if the dependence of both the delayed and powered output signals of the linear model on model parameters is ignored. In this way, the model is treated as a linear-in-parameters one. Such a simplification reduces computational complexity of the algorithm at the price of gradient approximation accuracy.

A neural network-based method for the identification of Wiener systems was developed by Al-Duwaish [3]. In this method, the linear dynamic system and the nonlinear element are identified separately. First, the parameters of the linear system, described by a linear difference equation, are estimated with

the recursive least squares (RLS) algorithm based on a response to a small input signal which ensures a linear perturbation of the nonlinear element. Then having the linear system identified, the backpropagation learning algorithm is applied to train a multilayer neural network model of the nonlinear element. This step is performed with another response to an increased input signal which perturbs the nonlinear element nonlinearly.

The identification of a MISO Wiener system was studied by Ikonen and Najim [72]. Based on the input-output data from a pump-valve pilot system, a neural network Wiener model was trained with the Levenberg-Marquardt method.

Visala *et al.* [159] used a MIMO Wiener model for the modelling of the chromatographic separation process. The linear dynamic part of their model is composed of Laguerre filters while the MIMO feedforward neural network is employed as a static nonlinear mapping. It is assumed that the MIMO linear dynamic model is decoupled. Parameter estimation is reduced to training the nonlinear part of the model with the Levenberg-Marquardt method as the linear dynamics is assumed to be known. This means that suitable values of Laguerre parameters are selected on the basis of a priori information.

The identification of a nonlinear dynamic system using a model composed of a SIMO dynamic system followed by a MISO nonlinear element was studied by Alataris *et al.* [1]. Their methodology employs a multilayer perceptron with a single hidden layer and polynomial activation functions as a model of static nonlinearity. The inputs to this models are outputs of a Laguerre filter bank used as a model of the linear dynamic system. The parameters of the neural network model are adjusted by means of the gradient descent method. The choice of a real pole, being the only degree of freedom of Laguerre filters, is based on a trial and error rule. Applying the polynomial activation function permits an easy transition between the neural network and Volterra series models.

A neural network model of both the linear dynamic system and the nonlinear element is used to describe SISO Wiener systems in the single step sequential procedure proposed by Janczak [89]. For more details about this approach and its extension to the MIMO case refer to Chapter 2.

The identification of Wiener systems using a genetic method, revealing a global optimization property, was studied by Al-Duwaish [5]. In this approach, the parameters of a Wiener model composed of a pulse transfer function model of the linear part and a polynomial model of the nonlinear element are calculated. The pulse transfer function model is defined in the pole-zero form. With the fitness function defined as a sum of squared errors between the system and model outputs and by performing genetic operations of cross-over, mutation, and selection, new generations of solutions are generated and evaluated until predetermined accuracy of approximation is achieved. A parallel neural network Wiener model was also trained using recursive evolutionary programming with a time-dependent learning rate by Janczak and Mrugalski [93].

1.4 Identification of Hammerstein systems

As in the case of Wiener systems, discussed in Section 1.3, we will review briefly various identification methods for Hammerstein systems.

Correlation methods. For a white Gaussian input, the computation of the cross-correlation function makes it possible to decouple the identification of Hammerstein systems and identify the linear dynamic system and the nonlinear element separately [14, 15, 16, 17]. First, the impulse response $h(k)$ of the linear dynamic system is estimated using the correlation technique. The first order correlation function $R_{uy}(k)$ is directly proportional to the impulse response, $R_{uy}(k) = \alpha h(k)$. Then if necessary, the parameters of the pulse transfer function $\mathcal{Z}[\alpha h(k)] = B(q^{-1})/A(q^{-1})$ can be calculated from the impulse response easily. Having an estimate of $\alpha h(k)$ available, the parameters of a polynomial model of the nonlinear element can be calculated with the least squares algorithm [16].
For Hammerstein systems, the second order correlation function $R_{u^2y}(k)$ is directly proportional to the impulse response of the linear element and provides a convenient test of the system structure. If the first and second order correlation functions are equal except for a constant of proportionality, the system must have the structure of a Hammerstein model.

Linear optimization methods. In contrast to correlation methods, which use a nonparametric model of the linear dynamic system and a parametric model of the nonlinear element, linear optimization methods use parametric representations for both parts of the Hammerstein system.
The parameters of a pulse transfer function representation of the linear element and a polynomial model of the nonlinear element can be estimated using the iterative least squares method proposed by Narendra and Gallman [120]. The method is based on an alternate adjustment of the parameters of the linear and nonlinear parts of the model.
Another iterative approach was proposed by Haist *et al.* [63] for Hammerstein systems with correlated additive output disturbances. This method, being an extension of the method of Narendra and Gallman, overcomes its obvious drawback of biased estimates via estimation of both parameters of the Hammerstein model and a linear noise model.
The transformation of the SISO Hammerstein model into the MISO form makes it possible to estimate the parameters of the transformed model utilizing the least squares method noniteratively [27]. In spite of its simplicity and the elegant form of a one step solution, this method has an inconvenience in the form of redundancy in the calculation of the parameters of the nonlinear element. More precisely, nb different sets of these parameters can be calculated from the parameters of the transformed MISO model, where nb is the nominator order of the pulse transfer function.
With instrumental variables defined as linear filter outputs and powers of

system inputs, instrumental variables methods were studied by Stoica and Söderström [152]. They proved that the instrumental variables approach gives consistent parameter estimates under mild conditions.
An algorithm that uses an OBFP model of the MIMO linear dynamic system and a nonlinear basis function model of the MIMO nonlinear element was proposed by Gómez ana Baeyens [44]. In this approach, the MIMO Hammerstein model is transformed into the linear-in-parameters form and its parameters are calculated using the least squares method. To calculate matrix parameters of the nonlinear element from parameter estimates of the transformed model, the singular value decomposition technique is applied. The algorithm provides consistent parameter estimates under weak assumptions about the persistency of the excitation of system inputs.
Bai [11] proposed a two-step algorithm that decouples the identification of the linear dynamic system from the identification of the nonlinear element. In the first step, the algorithm uses a pseudo-random binary sequence (PRBS) input to identify the linear dynamic system. With the PRBS signal, the effect of nonlinearity can be eliminated as any static nonlinear function can be completely characterized by a linear function under the PRBS input. Therefore, any identification method of linear systems can be applied to obtain a linear dynamic model. The identification of the nonlinear element is made in the other step. As the PRBS signal assumes only two values, a new input signal that is rich enough, e.g., a pseudo-random sequence of a uniform distribution is used to identify the nonlinear element. An important advantage of this decoupling technique is that the asymptotic variance of the estimate of the Hammerstein system transfer function is equal to the asymptotic variance of the estimate of the system transfer function in the linear case [126].
A piecewise linear model of the nonlinear element and a pulse transfer function model of the linear dynamic system are used in the identification scheme proposed by Giri et al. [43]. The Hammerstein model, defined in this way, is transformed into the linear-in-parameters form and parameter estimation is performed using a recursive gradient algorithm augmented by a parameter projection. To ensure the convergence of the model to the true system, a persistently exciting input sequence is generated.

Nonparametric regression methods. In parametric regression methods, it is assumed that the nonlinear characteristic belongs to a class that can be parameterized by a finite set of parameters. Clearly, such an assumption is a very restrictive one. For example, a commonly used polynomial representation of the nonlinear element excludes typical discontinuous characteristics such as dead-zone limiters, hard-limiters, and quantizers. A very large class of nonlinear functions, including all measurable L_2 functions, can be identified using nonparametric regression. A nonparametric regression approach to the identification of Hammerstein systems was originally proposed by Greblicki and Pawlak [55] and studied further in [56, 57, 103]. Kernel regression identification procedures for different block-oriented systems, including Ham-

merstein ones, were also studied by Krzyżak and Partyka [105]. Nonparametric regression methods comprise two separate steps. First, the impulse response function of the dynamic system is estimated with a standard correlation method. Then the characteristic of the nonlinear element $f(\cdot)$ is estimated as a kernel regression estimate

$$\hat{f}\big(u(n)\big) = \frac{\sum\limits_{k=0}^{N-1} y(k+1)K\left(\dfrac{u(n)-u(k)}{l(N)}\right)}{\sum\limits_{k=0}^{N-1} K\left(\dfrac{u(n)-u(k)}{l(N)}\right)}, \tag{1.81}$$

where N is the number of measurements, $K(\cdot)$ is the kernel function, $l(\cdot)$ is a sequence of positive numbers. In this definition, $0/0$ is understood as zero.
It can be shown that nonparametric regression estimates converge to the true characteristic of the nonlinear element as the number of measurements N tends to infinity. Greblicki and Pawlak also showed that for sufficiently smooth characteristics, the rate of convergence is $O(N^{-2/5})$. All of the above identification algorithms recover the nonlinear characteristic in a nonrecursive manner. Two other algorithms based on kernel regression estimates, proposed by Greblicki and Pawlak [58], allow one to identify the nonlinear characteristic recursively.
Another class of nonparametric methods uses orthogonal expansions of the nonlinear function. Greblicki [45] considered a Hammerstein system driven by a random white input, with the system output disturbed by random a white noise, and proposed two nonparametric procedures based on the trigonometric and Hermite orthogonal expansions. Both of these algorithms converge to the nonlinear characteristic of the system in a pointwise manner, and the integrated error converges to zero. The pointwise convergence rate is $O(N^{-(2q-1)/4q})$ in probability, where q is the number of derivatives of the nonlinear characteristic. The identification of Hammerstein systems by algorithms based on the Hermite series expansion with the number of terms depending nonlinearly on input-output measurements was considered by Krzyżak *et al.* [106]. In this approach, the system is driven by the stationary white noise, and the linear and nonlinear components are estimated simultaneously. The application of the Fourier series estimate to identify nonlinearities in block-oriented systems, including Hammerstein ones, was studied by Krzyżak [102, 104]. For nonlinear functions and input signal densities having finite Fourier series expansions, such an approach has two advantages over kernel regression ones – higher computational efficiency and higher rates of convergence. Recovering the nonlinear function using a Legendre polynomial-based method with an adaptively selected number of terms was studied by Pawlak [136]. It was shown that the estimate of $f(\cdot)$ is globally consistent and the rates of convergence were established. Greblicki and Pawlak [59] considered also the identification of Hammerstein systems with Laguerre polynomials.
An alternative to nonparametric methods based on orthogonal expansions

are algorithms that employ multiresolution approximation. In the context of Hammerstein systems identification, the Haar multiresolution analysis was first used by Pawlak and Hasiewicz [137]. This idea was studied further by Hasiewicz [64, 65, 66]. Haar multiresolution approximation algorithms converge pointwise. Their forms and convergence conditions are the same for white and correlated additive output noise. Their another advantage is the faster convergence rate in comparison with other nonparametric identification algorithms that use orthogonal series expansions.
The idea of the identification of nonlinearities in block-oriented systems with Daubechies wavelets was studied by Śliwiński and Hasiewicz [155]. As the lack of a closed analytical form makes Daubechies wavelets practically unapplicable, they used an estimation procedure that employs approximations that are easy to compute.

Nonlinear optimization methods. A prediction error approach to the identification of Hammerstein systems was discussed by Eskinat *et al.* [34]. Contrary to the least squares approach used by Chang and Luus [27], prediction error methods make it possible to estimate the parameters of a pulse transfer function of the linear system and a polynomial nonlinear characteristic element directly, without any transformation of the parameters, and there is no problem of parameter redundancy. The prediction error method uses the Levenberg-Marquardt method to approximate the Hessjan of the sum-squared cost function. Neglecting the fact that a model is nonlinear in the parameters and treating it as a linear one leads to pseudolinear regression methods. An example of such an approach is the method proposed by Boutayeb and Darouach [21], in which the parameters of a MISO Hammerstein system with the output disturbed additively by correlated noise are estimated recursively.
An identification method for Hammerstein systems which uses a two-segment model of the nonlinear element, composed of separate polynomial maps for positive and negative inputs, was proposed by Vörös [161]. This method also employs models in the pseudolinear form to calculate parameters iteratively. The idea of the method is based on splitting the nonlinear characteristic, which can be described accurately by a polynomial of a high order, into two segments that can be approximated with polynomials of a much lower order. A similar technique was also applied by Vörös [160] for the identification of Hammerstein systems with the nonlinear element described by a discontinuous function.
A neural network approach can be applied to the identification of Hammerstein systems with nonlinear elements described by a continuous function. The identification of Hammerstein systems with neural network models was considered by Su and McAvoy [153]. They used the steady-state and the transient data to train a neural network Hammerstein model. In this approach, a neural network model of the nonlinear element and a model of the linear dynamic system are trained separately. First, the neural network is trained with the backpropagation (BP) learning algorithm on the steady-state data. After the

training, the neural network serves as a nonlinear operator. The linear dynamic model is then trained on the set of transient data employing system input variables transformed by the nonlinear operator as inputs. Considering two basic configurations of the linear dynamic model, i.e., the series-parallel and the parallel one it is shown that the gradients of the cost function can be obtained via the backpropagation method for the series-parallel model, and the backpropagation through time method for the parallel one. Having the gradient calculated, a steepest descent or a conjugate gradient algorithm is suggested to adjust the model parameters.

The identification of SISO Hammerstein systems by multilayered feedforward neural networks was also studied by Al-Duwaish *et al.* [3]. They considered a parallel neural network Hammerstein model trained recursively using the BP learning algorithm for training a model of the nonlinear element and the recursive least squares algorithm (RLS) for training the linear dynamic model. In this approach, partial derivatives of the squared cost function are calculated in an approximate way without taking into account the dependence of past model outputs on model parameters. This corresponds to Equations (3.36) – (3.38). Moreover, the fact that partial derivatives of the model output w.r.t. the weights of the nonlinear element model depend not only on the actual but also on past outputs of the nonlinear element model is not taken ino account. Both of these simplifications of gradient calculation reduce computational complexity of training at the price of reduced accuracy of gradient calculation and may result in a decreased convergence rate. The extension of the RLS/BP algorithm to a MIMO case is discussed in [4].

A genetic approach to the identification of Hammerstein systems is considered in [5]. In this approach, the pole-zero form of the linear dynamic system and the nonlinearity of a known structure but unknown parameters are identified.

In general, SISO neural network Hammerstein models can be represented by a multilayer perceptron model of the nonlinear element and a linear node with two tapped delay lines used as a model of the linear system [73]. Both the series-parallel and parallel models can be considered. They have a similar architecture, the only difference is the feedback connection in the parallel model [100]. For the series-parallel model, the gradient of its output with respect to model parameters can be obtained using the computationally effective backpropagation algorithm. The calculation of the gradient in parallel models can be made with the sensitivity method or the backpropagation through time method [75, 77, 90]. As the Hammerstein model contains a linear part, it is also possible to use non-homogeneous algorithms that combine the recursive least squares or recursive pseudolinear regression algorithms with all of the above-mentioned methods [75]. The neural network Hammerstein models have simple architectures, commonly with a few tens of processing nodes and adjustable weights, and the trained models are easy to be applied in practice. Due to the simple architecture of neural network Hammerstein models, their training algorithms have low computational complexity. More details on the

neural network approach to the identification of Hammerstein system can be found in Chapter 3.

1.5 Summary

The material presented in this chapter starts with a concise introductory note explaining an underlaying incentive to write the book. Next, Section 1.1 contains a review of discrete time models of dynamic systems. It starts with the well-known linear model structures and shows nonlinear models as generalizations of linear ones. Section 1.2 introduces the multilayer perceptron, a neural network architecture that is used in both neural network Wiener and Hammerstein models. Next, in Sections 1.3 and 1.4, various available identification methods for Wiener and Hammerstein systems are briefly reviewed and classified into the following four groups: correlation methods, linear optimization methods, nonparametric regression methods, and nonlinear optimization methods.

2

Neural network Wiener models

2.1 Introduction

This chapter introduces different structures of neural network Wiener models and shows how their weights can be adjusted, based on a set of system input-output data, with gradient learning algorithms. The term 'neural network Wiener models' refers to models composed of a linear dynamic model followed by a nonlinear multilayer perceptron model. Both the SISO nad MISO Wiener models in their two basic configurations known as a series-parallel and a parallel model are considered. In series-parallel Wiener models, another multilayer perceptron is used to model the inverse nonlinear element. For neural network Wiener models, four different rules for the calculation of the gradient or the approximate gradient are derived and presented in a unified framework. In series-parallel models, represented by feedforward neural networks, the calculation of the gradient can be carried out with the backpropagation method (BPS). Three other methods, i.e., backpropagation for parallel models (BPP), the sensitivity method (SM), and truncated backpropagation through time (BPTT) are used to calculate the gradient or the approximate gradient in parallel models. For the BPTT method, it is shown that the accuracy of gradient approximation depends on both the number of unfolded time steps and impulse response functions of the linear dynamic model and its sensitivity models. Computational complexity of the algorithms is also analyzed and expressed in terms of the orders of polynomials describing the linear dynamic model, the number of nonlinear nodes, and the number of unfolded time steps. Having the gradient calculated, different gradient-based algorithms such as the steepest descent, quasi-Newton (or variable metric), and conjugate gradient can be applied easily.

All the learning algorithms, discussed in this chapter, are one-step identification procedures, i.e., they allow one to identify both the nonlinear element and the linear dynamic system simultaneously. The sequential (on-line) mode of the algorithms makes them also suitable for the identification of systems with slowly varying parameters. An important advantage of the parallel Wie-

A. Janczak: Identification of Nonlinear Systems, LNCIS 310, pp. 31–75, 2005.

ner model is that it does not require the assumption of invertibility of the nonlinear element. Due to output error formulation of the prediction error, parallel Wiener models are also useful for the identification of systems disturbed additively by the white output noise.
The chapter is organized as follows: The identification problem is formulated in Section 2.2. Then the SISO and MISO series-parallel and parallel neural network Wiener models are introduced in Section 2.3. Section 2.4 contains details of different gradient calculation algorithms and an analysis of gradient computation accuracy with the truncated BPTT method. Simulation results and a real data example of a laboratory two-tank system are shown in Sections 2.5 and 2.6 to compare the convergence rates of the algorithms. The recursive prediction error learning algorithm is derived in Section 2.7. Finally, Section 2.8 summarizes the essential results.

2.2 Problem formulation

Consider a two-block structure composed of a linear dynamic system and a static nonlinear element in a cascade, referred to as the SISO Wiener system – Fig. 2.1. The output $y(n)$ to the input $u(n)$ at the time n is

$$y(n) = f\left(\frac{B(q^{-1})}{A(q^{-1})}u(n)\right) + \varepsilon(n), \tag{2.1}$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}, \tag{2.2}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{nb} q^{-nb}, \tag{2.3}$$

and q^{-1} is the backward shift operator, with the properties that $q^{-m}y(n) = y(n-m)$ and $q^{-m}f\big(s(n)\big) = f\big(s(n-m)\big)$, $f(\cdot)$ is the characteristic of the nonlinear element, $a_1, \ldots, a_{na}$, $b_1, \ldots, b_{nb}$ are the unknown parameters of the linear dynamic system, and $\varepsilon(n)$ is the system output disturbance. Let us assume that:
Assumption 2.1. The function $f(\cdot)$ is continuous.
Assumption 2.2. The linear dynamic system is casual and asymptotically stable.
Assumption 2.3. The polynomial orders na and nb are known.
The identification problem can be formulated as follows: Given the sequence of the system input and output measurements $\{u(n), y(n)\}, n = 1, \ldots, N$, estimate the parameters of the linear dynamic system and the characteristic of the nonlinear element minimizing the following global cost function:

$$J = \frac{1}{2}\sum_{n=1}^{N}\big(y(n) - \hat{y}(n)\big)^2, \tag{2.4}$$

where $\hat{y}(n)$ is the output of the neural network Wiener model.

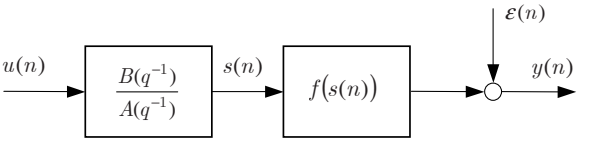

Fig. 2.1. Wiener system

There are two basic approaches to find the minimum of the global cost function (2.4). The first method is called the sequential mode or pattern learning as it uses pattern-by-pattern updating of model parameters, changing their values by an amount proportional to the negative gradient of the local cost function

$$J(n) = \frac{1}{2}\big(y(n) - \hat{y}(n)\big)^2. \tag{2.5}$$

This results in the following rule for the adaptation of model parameters:

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \eta \frac{\partial J(n)}{\partial \mathbf{w}(n-1)}, \tag{2.6}$$

$$\frac{\partial J(n)}{\partial \mathbf{w}(n-1)} = -\big(y(n) - \hat{y}(n)\big) \frac{\partial \hat{y}(n)}{\partial \mathbf{w}(n-1)}, \tag{2.7}$$

where $\mathbf{w}(n)$ is the weight vector containing all model parameters at the time n, and $\eta > 0$ is the learning rate. It has been shown that such a learning procedure minimizes the global cost function J provided that the learning rate η is sufficiently small [167].

The other approach is called batch learning as it uses the whole set of input-output data $\{u(n), y(n)\}, n = 1, \ldots, N$, to update model parameters. In this technique, the global cost function J is minimized iteratively in such a way that, at each iteration, the parameter changes over all training patterns are accumulated before the parameters are actually changed.

The gradient learning algorithm in its basic version has some serious drawbacks. The fixed learning rate η may be chosen too large leading to unnecessary oscillations or even the divergence of the learning process. On the other hand, at too small η, the learning process may become extremely slow. Many different modifications of the gradient algorithm exist which improve and speed up the learning process considerably. Adding a momentum term is one simple way to improve the learning. Including the momentum term smoothes weight changes making it possible to increase the learning rate η. This may not only speed up the learning process but may also prevent it from getting trapped in a shallow local minimum. Also, applying diverse learning rates for different weights, neurons, or layers may be a useful tool to improve the learning process. Other effective modifications of the gradient algorithm are based on the adaptation of the learning rate η according to some heuristic rules [68].

It is well known that parallel models can loose their stability during the learning process even if the identified system is stable [121]. A common way to avoid this is to keep the learning rate η small. To preserve the stability of Wiener models, some other heuristic rules can be applied easily. For example, asymptotic stability of the model can be tested after each parameter update and if the model becomes unstable, the unstable poles can be moved back into the unit circle by scaling. This requires recalculation and scaling the parameters of the linear dynamic model to keep the steady-state gain of the model unchanged. Another simple approach, which can be applied in the sequential mode, is based on the idea of a trial update of parameters, testing the stability of the model, and performing an actual update only if the obtained model is asymptotically stable; otherwise the parameters are not updated and the next training pattern is processed.

2.3 Series-parallel and parallel neural network Wiener models

In analogy with linear dynamic models, neural network Wiener models can be used in two basic configurations known as series parallel and parallel models. Neural network architectures of both of these models are introduced and discussed below.

2.3.1 SISO Wiener models

To introduce series-parallel Wiener models, it is necessary to assume that the nonlinear function $f(\cdot)$ is invertible. A series-parallel neural network Wiener model, shown in Fig. 2.2, is composed of a multilayer perceptron model of the inverse nonlinear element, a linear node with two tapped delay lines, used as a model of the linear dynamic system, and another multilayer perceptron used as a model of the nonlinear element. The input to the model are the system input $u(n)$ and system output $y(n)$, and the model is of the feedforward type. Multilayer perceptrons are universal approximators. This means that the multilayer perceptron with at least one hidden layer can approximate any smooth function to an arbitrary degree of accuracy as the number of hidden layer neurons increases [37]. Multilayer perceptrons with one hidden layer are most common. Therefore, we assume that both the model of the nonlinear element and the inverse model of the nonlinear element have the same architecture, and contain one hidden layer of M nonlinear processing elements, see Fig. 2.3 for the nonlinear element model and Fig 2.4 for the inverse nonlinear element model. The output $\hat{y}(n)$ of the series-parallel neural network Wiener model is

$$\hat{y}(n) = \hat{f}\big(\hat{s}(n),\mathbf{w}\big), \tag{2.8}$$

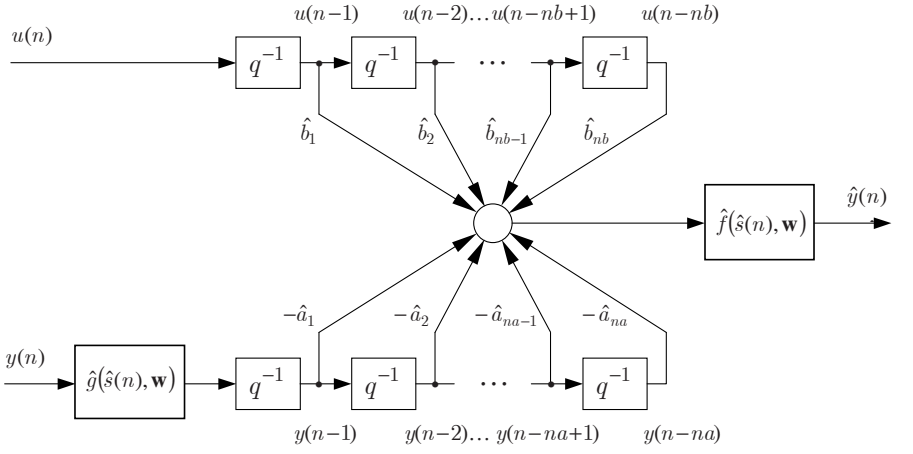

Fig. 2.2. Series-parallel SISO neural network Wiener model

with

$$\hat{s}(n) = -\sum_{m=1}^{na} \hat{a}_m \hat{g}\big(y(n-m), \mathbf{v}\big) + \sum_{m=1}^{nb} \hat{b}_m u(n-m), \tag{2.9}$$

$$\hat{f}\big(\hat{s}(n), \mathbf{w}\big) = \sum_{j=1}^{M} w_{1j}^{(2)} \varphi\big(x_j(n)\big) + w_{10}^{(2)}, \tag{2.10}$$

$$x_j(n) = w_{j1}^{(1)} \hat{s}(n) + w_{j0}^{(1)}, \tag{2.11}$$

$$\hat{g}\big(y(n), \mathbf{v}\big) = \sum_{j=1}^{M} v_{1j}^{(2)} \varphi\big(z_j(n)\big) + v_{10}^{(2)}, \tag{2.12}$$

$$z_j(n) = v_{j1}^{(1)} y(n) + v_{j0}^{(1)}, \tag{2.13}$$

where the function $\hat{f}(\cdot)$ describes the nonlinear element model, the function $\hat{g}(\cdot)$ describes the inverse nonlinear element model, $\varphi(\cdot)$ is the activation function, $\hat{a}_1, \ldots, \hat{a}_{na}$, $\hat{b}_1, \ldots, \hat{b}_{nb}$ are the parameters of the linear dynamic model, $\mathbf{w} = [w_{10}^{(1)} \ldots \; w_{M1}^{(1)} \; w_{10}^{(2)} \; \ldots \; w_{1M}^{(2)}]^T$ is the parameter (weight) vector of the nonlinear element model, and $\mathbf{v} = [v_{10}^{(1)} \; \ldots \; v_{M1}^{(1)} \; v_{10}^{(2)} \; \ldots \; v_{1M}^{(2)}]^T$ is the parameter vector of the inverse nonlinear element model. Note that $\hat{g}(\cdot)$ does not denote the inverse of $\hat{f}(\cdot)$ but describes the inverse nonlinear element. The architecture of the parallel model (Fig. 2.5), which does not contain any inverse model, is even simpler in comparison with that of the series-parallel one. The parallel model is of the recurrent type as it contains a single feedback

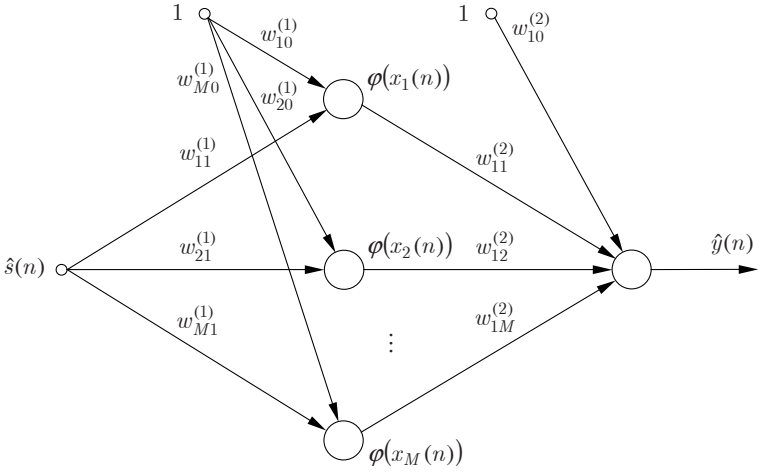

Fig. 2.3. Neural network model of the SISO nonlinear element

connection and $u(n)$ is the only input to the model. The output $\hat{y}(n)$ of the parallel Wiener model is given by

$$\hat{y}(n) = \hat{f}\big(\hat{s}(n),\mathbf{w}\big), \tag{2.14}$$

with $\hat{f}(\cdot)$ defined by (2.10) and (2.11), and where the output of the linear dynamic model is given by the following difference equation:

$$\hat{s}(n) = -\sum_{m=1}^{na} \hat{a}_m \hat{s}(n-m) + \sum_{m=1}^{nb} \hat{b}_m u(n-m). \tag{2.15}$$

Using the backward shift operator notation, (2.15) can be written as

$$\hat{s}(n) = \big[1 - \hat{A}(q^{-1})\big]\hat{s}(n) + \hat{B}(q^{-1})u(n), \tag{2.16}$$

where $\hat{A}(q^{-1}) = 1+\hat{a}_1 q^{-1}+\cdots+\hat{a}_{na}q^{-na}$ and $\hat{B}(q^{-1}) = \hat{b}_1 q^{-1}+\cdots+\hat{b}_{nb}q^{-nb}$.

Note that the characterization of the Wiener system is not unique as the linear dynamic system and the nonlinear element are connected in series. In other words, Wiener systems described by $\big(B(q^{-1})/A(q^{-1})\big)/\alpha$ and $f\big(\alpha s(n)\big)$ reveal the same input-output behavior for any $\alpha \neq 0$. Therefore, to obtain a unique characterization of the neural network model, either the linear dynamic model or the nonlinear element model should be normalized. For example, to normalize the gain of the linear dynamic model to 1, the model weights are scaled as follows: $\hat{b}_k = \tilde{b}_k/\alpha$, $k = 1,\ldots,nb$, $w_{j1}^{(1)} = \alpha\tilde{w}_{j1}^{(1)}$, $j = 1,\ldots,M$, where $\tilde{b}_k$, $\tilde{w}_{j1}^{(1)}$ denote the parameters of the unnormalized model, and $\alpha = \tilde{B}(1)/\tilde{A}(1)$ is the linear dynamic model gain.

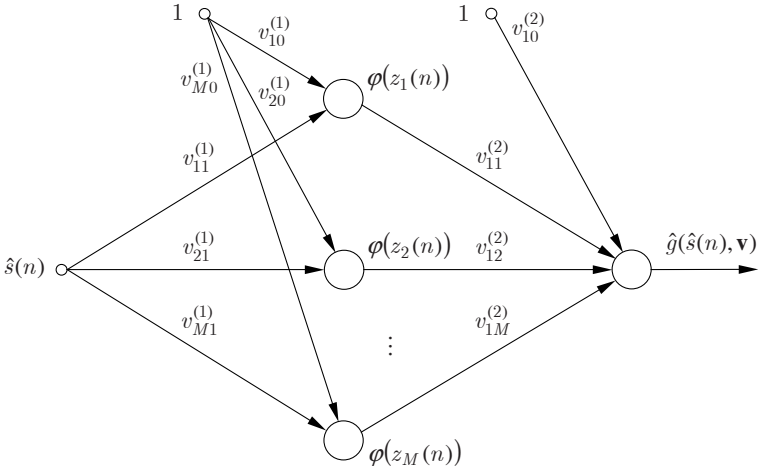

Fig. 2.4. Neural network model of the SISO inverse nonlinear element

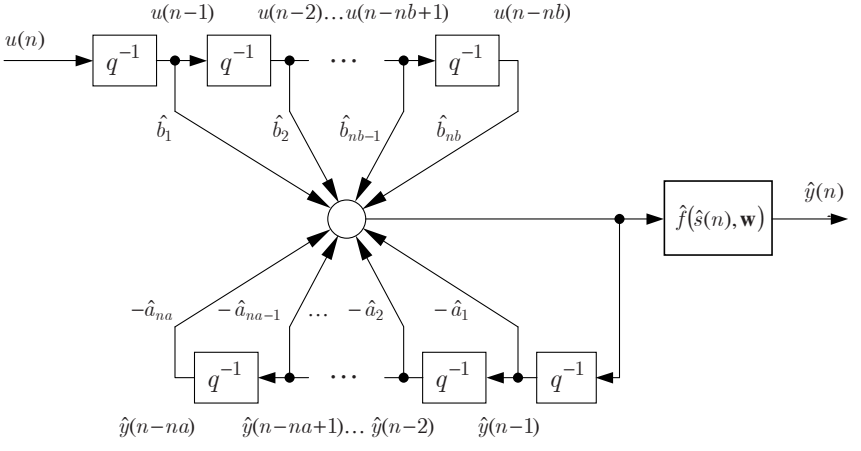

Fig. 2.5. Parallel SISO neural network Wiener model

2.3.2 MIMO Wiener models

Consider a parallel MIMO neural network Wiener model with nu inputs, ny outputs, ns outputs of the MIMO linear dynamic model, and M nonlinear nodes in the hidden layer. Assume that a single hidden layer multilayer perceptron with ns inputs and ny outputs, containing M nonlinear nodes in its hidden layer, is used as a model of the nonlinear element – Fig 2.6. The output $\hat{\mathbf{y}}(n)$ of the model at the time n is

$$\hat{\mathbf{y}}(n) = \hat{\mathbf{f}}\big(\hat{\mathbf{s}}(n),\mathbf{W}\big), \tag{2.17}$$

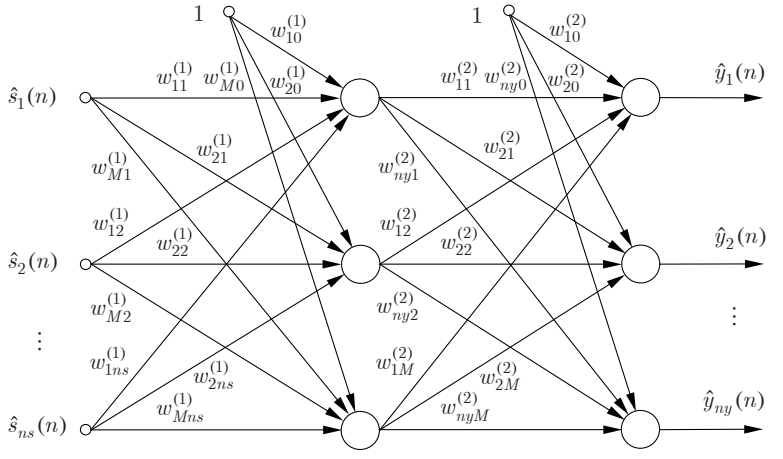

Fig. 2.6. Neural network model of the MIMO nonlinear element

where

$$\hat{\mathbf{y}}(n) = \big[\hat{y}_1(n) \dots \hat{y}_{ny}(n)\big]^T, \tag{2.18}$$

$$\hat{\mathbf{f}}(\hat{\mathbf{s}}(n), \mathbf{W}) = \big[\hat{f}_1\big(\hat{\mathbf{s}}(n), \mathbf{w}_1\big) \dots \hat{f}_{ny}\big(\hat{\mathbf{s}}(n), \mathbf{w}_{ny}\big)\big]^T, \tag{2.19}$$

$$\hat{\mathbf{s}}(n) = \big[\hat{s}_1(n) \dots \hat{s}_{ns}(n)\big]^T, \tag{2.20}$$

$$\mathbf{W} = \big[w_{10}^{(1)} \dots w_{Mns}^{(1)}\ w_{10}^{(2)} \dots w_{nyM}^{(2)}\big]^T, \tag{2.21}$$

$$\mathbf{w}_t = \big[w_{10}^{(1)} \dots w_{Mns}^{(1)}\ w_{t0}^{(2)} \dots w_{tM}^{(2)}\big]^T, \tag{2.22}$$

where $t = 1, \dots, ny$. Then the tth output of the nonlinear element model is

$$\hat{y}_t(n) = \hat{f}_t\big(\hat{\mathbf{s}}(n), \mathbf{w}_t\big) = \sum_{j=1}^{M} w_{tj}^{(2)} \varphi\big(x_j(n)\big) + w_{t0}^{(2)}, \tag{2.23}$$

$$x_j(n) = \sum_{i=1}^{ns} w_{ji}^{(1)} \hat{s}_i(n) + w_{j0}^{(1)}. \tag{2.24}$$

The output $\hat{\mathbf{s}}(n)$ of the MIMO dynamic model can be expressed as

$$\hat{\mathbf{s}}(n) = -\sum_{m=1}^{na} \hat{\mathbf{A}}^{(m)} \hat{\mathbf{s}}(n-m) + \sum_{m=1}^{nb} \hat{\mathbf{B}}^{(m)} \mathbf{u}(n-m), \tag{2.25}$$

where

$$\mathbf{u}(n) = \big[u_1(n) \dots u_{nu}(n)\big]^T, \tag{2.26}$$

$$\hat{\mathbf{A}}^{(m)} \in \mathbb{R}^{ns \times ns}, \tag{2.27}$$

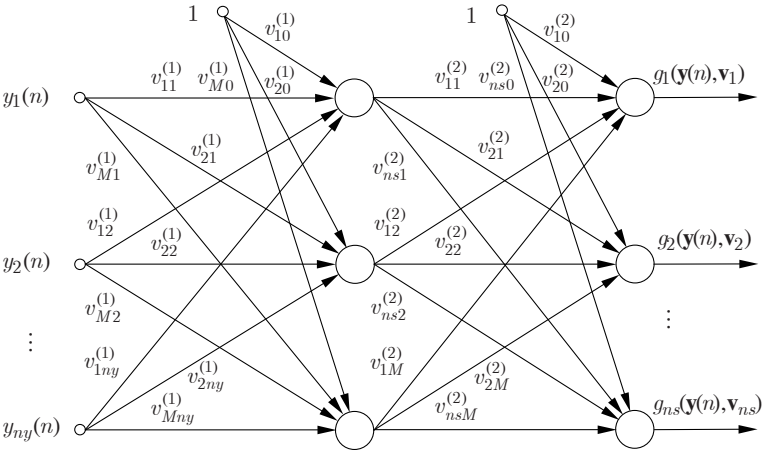

Fig. 2.7. Inverse neural network model of the MIMO nonlinear element

$$\hat{\mathbf{B}}^{(m)} \in \mathbb{R}^{ns \times nu}. \tag{2.28}$$

The series-parallel MIMO Wiener model is more complex than the parallel one as it uses an inverse model of the nonlinear element – Fig. 2.7. To derive the series-parallel MIMO Wiener model, assume that the nonlinear function $\mathbf{f}(\cdot)$ describing the nonlinear element is invertible. Then the output of the MIMO linear dynamic model can be expressed as

$$\hat{\mathbf{s}}(n) = -\sum_{m=1}^{na} \hat{\mathbf{A}}^{(m)} \hat{\mathbf{g}}\big(\mathbf{y}(n-m), \mathbf{V}\big) + \sum_{m=1}^{nb} \hat{\mathbf{B}}^{(m)} \mathbf{u}(n-m), \tag{2.29}$$

where

$$\hat{\mathbf{g}}\big(\mathbf{y}(n), \mathbf{V}\big) = \big[\hat{g}_1\big(\mathbf{y}(n), \mathbf{v}_1\big) \dots \hat{g}_{ns}\big(\mathbf{y}(n), \mathbf{v}_{ns}\big)\big]^T, \tag{2.30}$$

$$\mathbf{V} = \big[v_{10}^{(1)} \dots v_{Mny}^{(1)} \; v_{10}^{(2)} \dots v_{nsM}^{(2)}\big]^T, \tag{2.31}$$

$$\mathbf{v}_t = \big[v_{10}^{(1)} \dots v_{Mny}^{(1)} \; v_{t0}^{(2)} \dots v_{tM}^{(2)}\big]^T, \tag{2.32}$$

where $t = 1, \dots, ns$.

Note that $\hat{\mathbf{g}}(\cdot)$ in (2.29) describes the inverse nonlinear element model and does not denote the inverse of $\hat{\mathbf{f}}(\cdot)$. Assuming that the inverse nonlinear element is modelled by a single hidden layer perceptron with ny inputs, ns outputs, and M nonlinear nodes (Fig. 2.7), its tth output is

$$\hat{g}_t\big(\mathbf{y}(n), \mathbf{v}_t\big) = \sum_{j=1}^{M} v_{tj}^{(2)} \varphi\big(z_j(n)\big) + v_{t0}^{(2)}, \tag{2.33}$$

$$z_j(n) = \sum_{i=1}^{ny} v_{ji}^{(1)} y_i(n) + v_{j0}^{(1)}. \tag{2.34}$$

2.4 Gradient calculation

The calculation of the gradient in series-parallel Wiener models can be carried out with the well-known backpropagation algorithm. Series-parallel Wiener models are specialized multilayer perceptron networks with a multilayer perceptron model of the inverse nonlinear element followed by a single linear node model of the linear dynamic system followed by another multilayer perceptron structure of the nonlinear element model. In contrast, parallel Wiener models are dynamic systems and the corresponding neural networks are recurrent ones with one linear recurrent node that models the linear dynamic system followed by the multilayer perceptron model of the nonlinear element. This complicates the calculation of the gradient considerably. A crude approximation of the gradient can be obtained with the backpropagation method, which does not take into account the dynamic nature of the model. To evaluate the gradient more precisely, two other methods, known as the sensitivity method and the backpropagation through time method are used [88].
All pattern algorithms considered here have their batch counterparts. Given the rules of gradient computation, the batch versions of the algorithms can be derived easily, taking into account the fact that parameter updating should be performed iteratively, in a cumulative manner, after the presentation of all training patterns.

2.4.1 Series-parallel SISO model. Backpropagation method

Taking into account (2.9)–(2.13), the differentiation of (2.8) w.r.t. the model parameters $\hat{a}_k$, $k=1,\ldots,na$, $\hat{b}_k$, $k=1,\ldots,nb$, and v_c, w_c, $c=1,\ldots,3M+1$, where v_c and w_c denote the cth elements of $\mathbf{v}$ and $\mathbf{w}$, yields

$$\frac{\partial\hat{y}(n)}{\partial\hat{a}_k}=\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}\frac{\partial\hat{s}(n)}{\partial\hat{a}_k}=-\hat{g}\big(y(n-k),\mathbf{v}\big)\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}, \tag{2.35}$$

$$\frac{\partial\hat{y}(n)}{\partial\hat{b}_k}=\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}\frac{\partial\hat{s}(n)}{\partial\hat{b}_k}=u(n-k)\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}, \tag{2.36}$$

$$\frac{\partial\hat{y}(n)}{\partial v_c}=\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}\frac{\partial\hat{s}(n)}{\partial v_c}=-\sum_{m=1}^{na}\hat{a}_m\frac{\partial\hat{g}\big(y(n-m),\mathbf{v}\big)}{\partial v_c}\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}, \tag{2.37}$$

$$\frac{\partial\hat{y}(n)}{\partial w_c}=\frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial w_c}. \tag{2.38}$$

From (2.10) and (2.11), it follows that the partial derivative of the Wiener model output w.r.t. the output of the linear dynamic model is

$$\begin{aligned}\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)} &= \frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial\hat{s}(n)} = \sum_{j=1}^{M} w_{1j}^{(2)} \frac{\partial\varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial\hat{s}(n)} \\ &= \sum_{j=1}^{M} w_{j1}^{(1)} w_{1j}^{(2)} \varphi'\big(x_j(n)\big).\end{aligned} \tag{2.39}$$

Differentiating (2.12), partial derivatives of the inverse nonlinear element model output w.r.t. its parameters $v_{j1}^{(1)}$, $v_{j0}^{(1)}$, $v_{1j}^{(2)}$, $j = 1, \ldots, M$, and $v_{10}^{(2)}$ can be calculated as

$$\frac{\partial\hat{g}\big(y(n),\mathbf{v}\big)}{\partial v_{j1}^{(1)}} = \frac{\partial\hat{g}\big(y(n),\mathbf{v}\big)}{\partial\varphi\big(z_j(n)\big)} \frac{\partial\varphi\big(z_j(n)\big)}{\partial z_j(n)} \frac{\partial z_j(n)}{\partial v_{j1}^{(1)}} = v_{1j}^{(2)} \varphi'\big(z_j(n)\big) y(n), \tag{2.40}$$

$$\frac{\partial\hat{g}\big(y(n),\mathbf{v}\big)}{\partial v_{j0}^{(1)}} = \frac{\partial\hat{g}\big(y(n),\mathbf{v}\big)}{\partial\varphi\big(z_j(n)\big)} \frac{\partial\varphi\big(z_j(n)\big)}{\partial z_j(n)} \frac{\partial z_j(n)}{\partial v_{j0}^{(1)}} = v_{1j}^{(2)} \varphi'\big(z_j(n)\big), \tag{2.41}$$

$$\frac{\partial\hat{g}\big(y(n),\mathbf{v}\big)}{\partial v_{1j}^{(2)}} = \varphi\big(z_j(n)\big), \tag{2.42}$$

$$\frac{\partial\hat{g}\big(y(n),\mathbf{v}\big)}{\partial v_{10}^{(2)}} = 1. \tag{2.43}$$

From (2.10) and (2.11), it follows that partial derivatives of the output of the nonlinear element model w.r.t. the weights $w_{j1}^{(1)}$, $w_{j0}^{(1)}$, $w_{1j}^{(2)}$, $j = 1, \ldots, M$, and $w_{10}^{(2)}$ are

$$\frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial w_{j1}^{(1)}} = \frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial\varphi\big(x_j(n)\big)} \frac{\partial\varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial w_{j1}^{(1)}} = w_{1j}^{(2)} \varphi'\big(x_j(n)\big)\hat{s}(n), \tag{2.44}$$

$$\frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial w_{j0}^{(1)}} = \frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial\varphi\big(x_j(n)\big)} \frac{\partial\varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial w_{j0}^{(1)}} = w_{1j}^{(2)} \varphi'\big(x_j(n)\big), \tag{2.45}$$

$$\frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial w_{1j}^{(2)}} = \varphi\big(x_j(n)\big), \tag{2.46}$$

$$\frac{\partial\hat{f}\big(\hat{s}(n),\mathbf{w}\big)}{\partial w_{10}^{(2)}} = 1. \tag{2.47}$$

In spite of the fact that the series-parallel model is of the feedforward type, its training with the backpropagation (BPS) method is quite complex. This comes from the fact that the total number of hidden layers in the series-parallel model equals 4. Moreover, although both the nonlinear element and its inverse are identified, only an approximate inverse relationship between them is obtained in practice.

2.4.2 Parallel SISO model. Backpropagation method

The parallel Wiener model does not contain any inverse model of the nonlinear element (Fig. 2.5). This simplifies the training of the model considerably. On the other hand, as only an approximate gradient is calculated with the backpropagation (BPP) method, a very slow convergence rate can be observed. In the BPP method, the dependence of the past linear dynamic model outputs $\hat{s}(n-m), m = 1, \ldots, na$, on the parameters $\hat{a}_k$ and $\hat{b}_k$ is neglected. Hence, from (2.14) and (2.15), it follows that

$$\frac{\partial \hat{y}(n)}{\partial \hat{a}_k} = \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)} \frac{\partial \hat{s}(n)}{\partial \hat{a}_k} = -\hat{s}(n-k) \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)}, \ k = 1, \ldots, na, \tag{2.48}$$

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)} \frac{\partial \hat{s}(n)}{\partial \hat{b}_k} = u(n-k) \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)}, \ k = 1, \ldots, nb, \tag{2.49}$$

$$\frac{\partial \hat{y}(n)}{\partial w_c} = \frac{\partial \hat{f}\big(\hat{s}(n), \mathbf{w}\big)}{\partial w_c}. \tag{2.50}$$

The partial derivative of the parallel Wiener model output w.r.t. the output of the linear dynamic model is

$$\begin{aligned} \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)} &= \frac{\partial \hat{f}\big(\hat{s}(n), \mathbf{w}\big)}{\partial \hat{s}(n)} = \sum_{j=1}^{M} w_{1j}^{(2)} \frac{\partial \varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial \hat{s}(n)} \\ &= \sum_{j=1}^{M} w_{j1}^{(1)} w_{1j}^{(2)} \varphi'\big(x_j(n)\big). \end{aligned} \tag{2.51}$$

Partial derivatives of the parallel Wiener model output w.r.t. the weights $w_{j1}^{(1)}$, $w_{j0}^{(1)}$, $w_{1j}^{(2)}$, $j = 1, \ldots, M$, and $w_{10}^{(2)}$ are calculated in the same way as for the series-parallel model using (2.44) – (2.47).

2.4.3 Parallel SISO model. Sensitivity method

The sensitivity method (SM) differs from the BPP method markedly in the calculation of partial derivatives of the output of the linear dynamic model

w.r.t. its parameters. In general, as the SM uses a more accurate evaluation of the gradient than the BPP method, a higher convergence rate can be expected. Assuming that the parameters $\hat{a}_k$ and $\hat{b}_k$ do not change and differentiating (2.15), we have

$$\frac{\partial \hat{s}(n)}{\partial \hat{a}_k} = -\hat{s}(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{s}(n-m)}{\partial \hat{a}_k}, \ k = 1, \ldots, na, \tag{2.52}$$

$$\frac{\partial \hat{s}(n)}{\partial \hat{b}_k} = u(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{s}(n-m)}{\partial \hat{b}_k}, \ k = 1, \ldots, nb. \tag{2.53}$$

The partial derivatives (2.52) and (2.53) can be computed on-line by simulation, usually with zero initial conditions. The substitution of (2.52) and (2.53) into (2.48) and (2.49) gives

$$\frac{\partial \hat{y}(n)}{\partial \hat{a}_k} = \left(-\hat{s}(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{s}(n-m)}{\partial \hat{a}_k} \right) \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)}, \tag{2.54}$$

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \left(u(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{s}(n-m)}{\partial \hat{b}_k} \right) \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)}. \tag{2.55}$$

The calculation of partial derivatives for the parallel Wiener model is illustrated in Fig. 2.8. Other partial derivatives are calculated in the SM in the same way as in the BPP method. In comparison with the BPP method, the SM is only a little more computationally intensive. The increase in computational burden comes from the simulation of $na + nb$ sensitivity models, whereas dynamic models of a low order are used commonly.

Note that to obtain the exact value of the gradient, the parameters $\hat{a}_k$ and $\hat{b}_k$ should be kept constant. That is the case only in the batch mode, in which the parameters are updated after the presentation of all learning patterns. In the sequential mode (pattern learning), the parameters $\hat{a}_k$ and $\hat{b}_k$ are updated after each learning pattern and, as a result, an approximate value of the gradient is obtained. Therefore, to achieve a good approximation accuracy, the learning rate η should be sufficiently small to keep changes in the parameters negligible.

2.4.4 Parallel SISO model. Backpropagation through time method

In the backpropagation through time (BPTT) method, partial derivatives of the model output w.r.t. the weights of the nonlinear element model are calculated in the same way as in BPS and BPP methods, or the SM from (2.44) – (2.47). Also, partial derivatives of the model output w.r.t. the output of the linear dynamic model are calculated in the same way from (2.51).

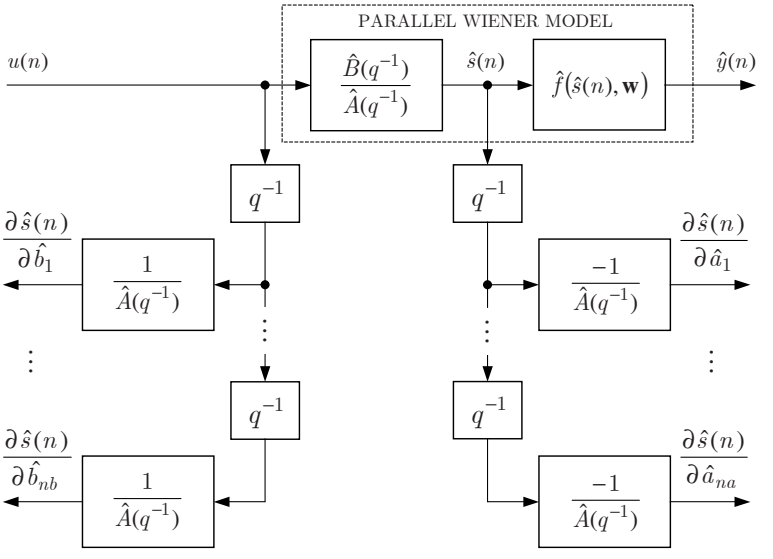

Fig. 2.8. Parallel Wiener model and its sensitivity models

The only difference is in the computation of partial derivatives of the linear dynamic model output w.r.t. its parameters $\hat{a}_k$ and $\hat{b}_k$. To perform this, the linear dynamic model is unfolded back in time. Unfolding (2.15) back in time for one step gives

$$\begin{aligned}\hat{s}(n) = &- \hat{a}_1\hat{s}(n-1) - \sum_{m=2}^{na}\hat{a}_m\hat{s}(n-m) + \sum_{m=1}^{nb}\hat{b}_m u(n-m) = \\ &- \hat{a}_1\Big[-\sum_{m=1}^{na}\hat{a}_m\hat{s}(n-m-1) + \sum_{m=1}^{nb}\hat{b}_m u(n-m-1)\Big] \\ &- \sum_{m=2}^{na}\hat{a}_m\hat{s}(n-m) + \sum_{m=1}^{nb}\hat{b}_m u(n-m).\end{aligned} \tag{2.56}$$

Such an unfolding procedure can be continued until the initial time step is obtained. The unfolded Wiener model is no more of the recurrent type and can be represented by a feedforward neural network with the model of the nonlinear element on its top and copies of the linear dynamic model below, see Fig. 2.9 for an example of the 3rd order model. When discrete time elapses, the number of copies of the linear dynamic model increases. At the time n, a fully unfolded model contains n copies of the linear dynamic model. To keep computational complexity constant, unfolding in time is commonly restricted to only a given number of time steps K in a method called truncated

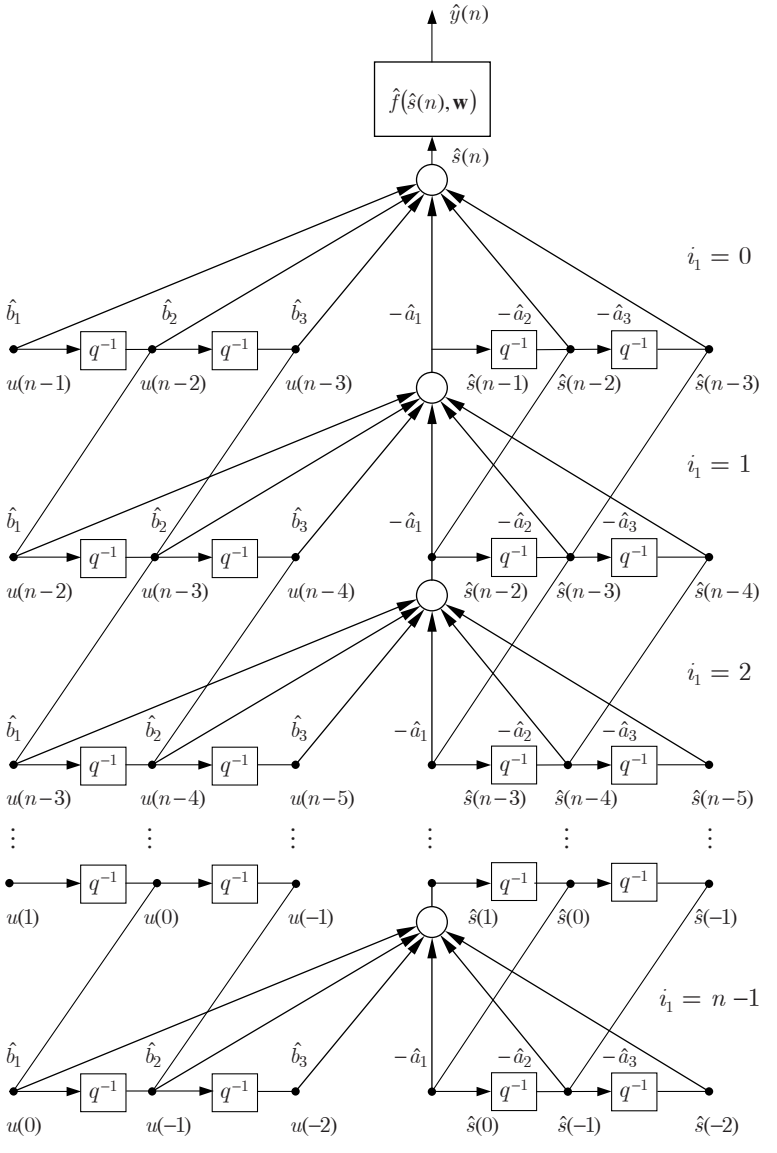

Fig. 2.9. Unfolded-in-time Wiener model of the third order

BPTT. In this way, only an approximate gradient is computed. The unfolded network is of the feedforward type and differs from a common multilayer perceptron distinctly. First, apart from adjustable weights, marked in Fig. 2.9 with thick lines, it contains nonadjustable short-circuit connections between the neighboring layers, marked with thin lines. Then for models of the order

$na > 2$, some of these short-circuit weights are connected in series and form lines connecting more distant layers. Finally, the unfolded model contains as many copies of the linear dynamic model as unfolded time steps. All these differences should be taken into account in the training algorithm. A detailed description of the truncated BPTT algorithm is given in Appendix 2.1.

2.4.5 Series-parallel MIMO model. Backpropagation method

In a sequential version of the gradient-based learning algorithm extended to the MIMO case, the following cost function is minimized w.r.t. the model parameters:

$$J(n) = \frac{1}{2}\sum_{t=1}^{ny}\big(y_t(n) - \hat{y}_t(n)\big)^2, \tag{2.57}$$

where $\hat{y}_t(n) = \hat{f}_t\big(\hat{\mathbf{s}}(n),\mathbf{w}_t\big)$ is the tth output of the Wiener model, and $\hat{\mathbf{s}}(n)$ denotes the vector of the outputs of the linear dynamic model. From (2.29), it follows that the lth output of the linear dynamic model can be written as

$$\hat{s}_l(n) = -\sum_{m=1}^{na}\sum_{m_1=1}^{ns}\hat{a}_{lm_1}^{(m)}\hat{g}_{m_1}\big(\mathbf{y}(n-m),\mathbf{v}_{m_1}\big) + \sum_{m=1}^{nb}\sum_{m_1=1}^{nu}\hat{b}_{lm_1}^{(m)}u_{m_1}(n-m), \tag{2.58}$$

where $l = 1,\ldots,ns$, and $\hat{a}_{lm_1}^{(m)}$ and $\hat{b}_{lm_1}^{(m)}$ denote the elements of $\hat{\mathbf{A}}^{(m)}$ and $\hat{\mathbf{B}}^{(m)}$. Partial derivatives of the series-parallel model (2.17) and (2.29) can be calculated with the backpropagation method (BPS). Differentiating (2.23), partial derivatives of the model output $\hat{y}_t(n)$, $t = 1,\ldots,ny$, w.r.t. the parameters of the nonlinear element model can be obtained as

$$\frac{\partial\hat{y}_t(n)}{\partial w_{ji}^{(1)}} = \frac{\partial\hat{f}_t\big(\hat{\mathbf{s}}(n),\mathbf{w}_t\big)}{\partial\varphi\big(x_j(n)\big)}\frac{\partial\varphi\big(x_j(n)\big)}{\partial x_j(n)}\frac{\partial x_j(n)}{\partial w_{ji}^{(1)}} = w_{tj}^{(2)}\varphi'\big(x_j(n)\big)\hat{s}_i(n), \tag{2.59}$$

$$\frac{\partial\hat{y}_t(n)}{\partial w_{j0}^{(1)}} = \frac{\partial\hat{f}_t\big(\hat{\mathbf{s}}(n),\mathbf{w}_t\big)}{\partial\varphi\big(x_j(n)\big)}\frac{\partial\varphi\big(x_j(n)\big)}{\partial x_j(n)}\frac{\partial x_j(n)}{\partial w_{j0}^{(1)}} = w_{tj}^{(2)}\varphi'\big(x_j(n)\big), \tag{2.60}$$

$$\frac{\partial\hat{y}_t(n)}{\partial w_{tj}^{(2)}} = \varphi\big(x_j(n)\big), \tag{2.61}$$

$$\frac{\partial\hat{y}_t(n)}{\partial w_{t0}^{(2)}} = 1, \tag{2.62}$$

where $j = 1,\ldots,M$, and $i = 1,\ldots,ns$. From (2.23) and (2.58), it follows that

$$\frac{\partial\hat{y}_t(n)}{\partial\hat{a}_{lm_1}^{(k)}} = \frac{\partial\hat{y}_t(n)}{\partial\hat{s}_l(n)}\frac{\partial\hat{s}_l(n)}{\partial\hat{a}_{lm_1}^{(k)}} = -\hat{g}_{m_1}\big(\mathbf{y}(n-k),\mathbf{v}_{m_1}\big)\frac{\partial\hat{y}_t(n)}{\partial\hat{s}_l(n)}, \quad m_1 = 1,\ldots,ns, \quad k = 1,\ldots,na, \tag{2.63}$$

$$\frac{\partial \hat{y}_t(n)}{\partial \hat{b}^{(k)}_{lm_1}} = \frac{\partial \hat{y}_t(n)}{\partial \hat{s}_l(n)} \frac{\partial \hat{s}_l(n)}{\partial \hat{b}^{(k)}_{lm_1}} = u_{m_1}(n-k) \frac{\partial \hat{y}_t(n)}{\partial \hat{s}_l(n)}, \quad m_1 = 1, \ldots, nu, \quad k = 1, \ldots, nb. \tag{2.64}$$

From (2.23) and (2.24), it follows that

$$\begin{aligned} \frac{\partial \hat{y}_t(n)}{\hat{s}_l(n)} &= \frac{\partial \hat{f}_t(\hat{\mathbf{s}}(n), \mathbf{w}_t)}{\partial \hat{s}_l(n)} = \sum_{j=1}^{M} w^{(2)}_{tj} \frac{\partial \varphi\big(x_j(n)\big)}{\partial x_j(n)} \frac{\partial x_j(n)}{\partial \hat{s}_l(n)} \\ &= \sum_{j=1}^{M} w^{(1)}_{jl} w^{(2)}_{tj} \varphi'\big(x_j(n)\big). \end{aligned} \tag{2.65}$$

Taking into account (2.58), partial derivatives of the model output w.r.t. the parameters of the inverse nonlinear element model can be calculated according to the following rule:

$$\begin{aligned} \frac{\partial \hat{y}_t(n)}{\partial v} &= \frac{\partial \hat{f}_t\big(\mathbf{s}(n), \mathbf{w}_t\big)}{\partial v} = \sum_{l=1}^{ns} \frac{\partial \hat{y}_t(n)}{\hat{s}_l(n)} \frac{\partial \hat{s}_l(n)}{\partial v} \\ &= \sum_{l=1}^{ns} \frac{\partial \hat{y}_t(n)}{\hat{s}_l(n)} \sum_{m=1}^{na} \sum_{m_1=1}^{ns} \hat{a}^{(m)}_{lm_1} \frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n-m), \mathbf{v}_{m_1}\big)}{\partial v}, \end{aligned} \tag{2.66}$$

where v is any parameter of the inverse nonlinear element model. From (2.33) and (2.34), it follows that partial derivatives of the inverse nonlinear element output w.r.t. its parameters are

$$\begin{aligned} \frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n), \mathbf{v}_{m_1}\big)}{\partial v^{(1)}_{ji}} &= \frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n), \mathbf{v}_{m_1}\big)}{\partial \varphi\big(z_j(n)\big)} \frac{\partial \varphi\big(z_j(n)\big)}{\partial z_j(n)} \frac{\partial z_j(n)}{\partial v^{(1)}_{ji}} \\ &= v^{(2)}_{m_1 j} \varphi'\big(z_j(n)\big) y_i(n), \end{aligned} \tag{2.67}$$

$$\begin{aligned} \frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n), \mathbf{v}_{m_1}\big)}{\partial v^{(1)}_{j0}} &= \frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n), \mathbf{v}_{m_1}\big)}{\partial \varphi\big(z_j(n)\big)} \frac{\partial \varphi\big(z_j(n)\big)}{\partial z_j(n)} \frac{\partial z_j(n)}{\partial v^{(1)}_{j0}} \\ &= v^{(2)}_{m_1 j} \varphi'\big(z_j(n)\big), \end{aligned} \tag{2.68}$$

$$\frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n), \mathbf{v}_{m_1}\big)}{\partial v^{(2)}_{m_1 j}} = \varphi\big(z_j(n)\big), \tag{2.69}$$

$$\frac{\partial \hat{g}_{m_1}\big(\mathbf{y}(n), \mathbf{v}_{m_1}\big)}{\partial v^{(2)}_{m_1 0}} = 1, \tag{2.70}$$

where $m_1 = 1, \ldots, ns$, $j = 1, \ldots, M$, $i = 1, \ldots, ny$.

2.4.6 Parallel MIMO model. Backpropagation method

From (2.25), it follows that the lth output of the linear dynamic model can be written as

$$\hat{s}_l(n) = -\sum_{m=1}^{na}\sum_{m_1=1}^{ns}\hat{a}_{lm_1}^{(m)}\hat{s}_{m1}(n-m) + \sum_{m=1}^{nb}\sum_{m_1=1}^{nu}\hat{b}_{lm_1}^{(m)}u_{m1}(n-m), \qquad (2.71)$$

where $l = 1,\ldots,ns$. Applying backpropagation rules to the parallel MIMO Wiener model, the dependence of the delayed outputs $\hat{s}_{m_1}(n-m)$, $m = 1,\ldots,na$, on the parameters $\hat{a}_{lm_1}^{(m)}$ and $\hat{b}_{lm1}^{(m)}$ is neglected. This reduces computational complexity of the backpropagation method for the parallel model (BPP) in comparison with the BPS method as no inverse nonlinear model is utilized. Using the BPP method, the calculation of partial derivatives of the model output w.r.t. the parameters of the nonlinear element model is performed in the same way as in the BPS method according to the formulae (2.59) – (2.62). Also, the calculation of $\partial\hat{y}_t(n)/\partial\hat{b}_{lm_1}^{(k)}$ is performed in the same way as in the BPS method according to (2.64). As the parallel Wiener model uses delayed outputs of the linear dynamic model instead of the outputs of the inverse nonlinear model, BPP differs from BPS in the calculation of $\partial\hat{y}_t(n)/\partial\hat{a}_{lm_1}^{(k)}$:

$$\frac{\partial\hat{y}_t(n)}{\partial\hat{a}_{lm_1}^{(k)}} = \frac{\partial\hat{y}_t(n)}{\partial\hat{s}_l(n)}\frac{\partial\hat{s}_l(n)}{\partial\hat{a}_{lm_1}^{(k)}} = -\hat{s}_{m_1}(n-k)\frac{\partial\hat{y}_t(n)}{\partial\hat{s}_l(n)},\ k = 1,\ldots,na. \qquad (2.72)$$

2.4.7 Parallel MIMO model. Sensitivity method

Assume that the linear dynamic model is time invariant. As in the case of the BPS and BPP methods for MIMO Wiener models, the calculation of partial derivatives of model output w.r.t. the parameters of the nonlinear element model in the SM is performed in the same way. From (2.71), it follows that to calculate $\partial\hat{y}_t(n)/\partial\hat{a}_{rp}^{(k)}$, $t = 1,\ldots,ny$, $k = 1,\ldots,na$, $r = 1,\ldots,ns$, $p = 1,\ldots,ns$, and $\partial\hat{y}_t(n)/\partial\hat{b}_{rp}^{(k)}$, $t = 1,\ldots,ny$, $k = 1,\ldots,nb$, $r = 1,\ldots,ns$, $p = 1,\ldots,nu$, the following set of linear difference equations is solved on-line by simulation:

$$\frac{\partial\hat{s}_l(n)}{\partial\hat{a}_{rp}^{(k)}} = -\delta_{lr}\hat{s}_p(n-k) - \sum_{m=1}^{na}\sum_{m_1=1}^{ns}\hat{a}_{lm_1}^{(m)}\frac{\partial\hat{s}_{m_1}(n-m)}{\partial\hat{a}_{rp}^{(k)}}, \qquad (2.73)$$

$$\frac{\partial\hat{s}_l(n)}{\partial\hat{b}_{rp}^{(k)}} = \delta_{lr}u_p(n-k) - \sum_{m=1}^{na}\sum_{m_1=1}^{ns}\hat{a}_{lm_1}^{(m)}\frac{\partial\hat{s}_{m_1}(n-m)}{\partial\hat{b}_{rp}^{(k)}}, \qquad (2.74)$$

where

$$\delta_{lr} = \begin{cases} 1 & \text{for } l = r \\ 0 & \text{for } l \neq r \end{cases}. \tag{2.75}$$

To solve (2.73) and (2.74), zero initial conditions are assumed.

2.4.8 Parallel MIMO model. Backpropagation through time method

Detailed derivation of the truncated backpropagation through time (BPTT) method for the parallel MIMO Wiener model is given in Appendix 2.2.

2.4.9 Accuracy of gradient calculation with truncated BPTT

An important issue that arises in the practical application of truncated BPTT is a proper choice of the number of unfolded time steps K. The number K should be large enough to ensure a good approximation of the gradient. On the other hand, too large K does not improve the convergence rate significantly but it increases computational complexity of the algorithm. Theorems 2.1 and 2.2 below give some insight into gradient calculation accuracy on the basis of the linear dynamic model and its sensitivity models and allows one to draw a conclusion on how the choice of K is to be made [89].

Theorem 2.1. Define the computation error

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

where $\partial\hat{s}(n)/\partial\hat{a}_k$, $k = 1, \ldots, na$, denote partial derivatives calculated with the BPTT method, i.e. unfolding the model (2.15) $n-1$ times back in time, and $\partial^+\hat{s}(n)/\partial\hat{a}_k$ denote partial derivatives calculated with the truncated BPTT method unfolding the model (2.15) K times back in time.

Assume that

(A1) The linear dynamic model $\hat{B}(q^{-1})/\hat{A}(q^{-1})$ is asymptotically stable;
(A2) $\hat{s}(n) = 0$, $n = 0, \ldots, -na+1$; $u(n) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial\hat{s}(n)/\partial\hat{a}_k = 0$, $n = 0, \ldots, -na+1$;
(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

$$\mathrm{var}\big(\Delta\hat{s}_{\hat{a}_k}(n)\big) = \sigma^2 \sum_{i_1=K+1}^{n-k} \left(\sum_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2) \right)^2, \; k = 1, \ldots, na, \tag{2.76}$$

for $n > K + k$ and 0 otherwise, where $h_1(n)$ is the impulse response function of the system

$$H_1(q^{-1}) = \frac{1}{\hat{A}(q^{-1})}, \tag{2.77}$$

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

$$H_2(q^{-1}) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}. \tag{2.78}$$

Proof: The proof is shown in Appendix 2.3.

Theorem 2.2. Define the computation error

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

where $\partial\hat{s}(n)/\partial\hat{b}_k$, $k = 1, \ldots, nb$, denote partial derivatives calculated with the BPTT method, i.e. unfolding the model (2.15) $n-1$ times back in time, and $\partial^+\hat{s}(n)/\partial\hat{b}_k$ denote partial derivatives calculated with the truncated BPTT method unfolding the model (2.15) K times back in time.

Assume that

(A1) The linear dynamic model $\hat{B}(q^{-1})/\hat{A}(q^{-1})$ is asymptotically stable;
(A2) $\hat{s}(n) = 0$, $n = 0, \ldots, -na + 1$; $u(n) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial\hat{s}(n)/\partial\hat{b}_k = 0$, $n = 0, \ldots, -na + 1$;
(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

$$\mathrm{var}\big(\Delta\hat{s}_{\hat{b}_k}(n)\big) = \sigma^2 \sum_{i_1=K+1}^{n-k} h_1^2(i_1),\ k = 1, \ldots, nb, \tag{2.79}$$

for $n > K + k$ and 0 otherwise, where $h_1(n)$ is the impulse response function of the system

$$H_1(q^{-1}) = \frac{1}{\hat{A}(q^{-1})}. \tag{2.80}$$

Remark 2.1. From (2.76) and (2.79), it follows that the variances of computation errors do not depend on k for $n \to \infty$,

$$\lim_{n\to\infty} \operatorname{var}\big(\Delta\hat{s}_{\hat{a}_k}(n)\big) = \sigma^2 \sum_{i_1=K+1}^{\infty} \left(\sum_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2) \right)^2, \tag{2.81}$$

$$\lim_{n\to\infty} \operatorname{var}\big(\Delta\hat{s}_{\hat{b}_k}(n)\big) = \sigma^2 \sum_{i_1=K+1}^{\infty} h_1^2(i_1). \tag{2.82}$$

Theorems 2.1 and 2.2 provide a useful tool for the determination of the number of unfolded time steps K. From Theorems 2.1 and 2.2, it follows that, for a fixed value of K, the accuracy of gradient calculation 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. Note that this impulse response can differ significantly from an impulse response of the system $1/A(q^{-1})$, 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 the gradient calculation accuracy $\xi_{a_k}(n)$ and $\xi_{b_k}(n)$ defined as the ratio of the variances (2.76) or (2.79) to the variances of the corresponding partial derivatives:

$$\xi_{a_k}(n) = \frac{\operatorname{var}\big(\Delta\hat{s}_{\hat{a}_k}(n)\big)}{\operatorname{var}\left(\dfrac{\partial\hat{s}(n)}{\partial\hat{a}_k}\right)} = \frac{\displaystyle\sum_{i_1=K+1}^{n-k} \left(\sum_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2) \right)^2}{\displaystyle\sum_{i_1=0}^{n-k} \left(\sum_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2) \right)^2}, \tag{2.83}$$

$$\xi_{b_k}(n) = \frac{\operatorname{var}\big(\Delta\hat{s}_{\hat{b}_k}(n)\big)}{\operatorname{var}\left(\dfrac{\partial\hat{s}(n)}{\partial\hat{b}_k}\right)} = \frac{\displaystyle\sum_{i_1=K+1}^{n-k} h_1^2(i_1)}{\displaystyle\sum_{i_1=0}^{n-k} h_1^2(i_1)}. \tag{2.84}$$

The initial value of K can be determined based on the assumed initial values of $\hat{a}_k$ and $\hat{b}_k$ or a priori knowledge of system dynamics.
The required value of K depends strongly on the system sampling rate. If the sampling rate increases for a given system, the number of discrete time steps, necessary for the impulse response $h_1(n)$ to decrease to negligible small values, increases and a higher value of K is necessary to achieve the same accuracy of gradient calculation.

2.4.10 Gradient calculation in the sequential mode

The derivation of gradient calculation algorithms has been made under the assumption that a model is time invariant. Obviously, pattern by pattern

updating of model parameters in the sequential mode makes this assumption no longer true. As a result, approximated values of the gradient are obtained for parallel models. To obtain a good approximation of the gradient, the learning rate should be small enough to achieve negligible small changes of model parameters. In comparison with the SM, an important advantage of the BPTT method is that 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, as illustrated in the simulation example below, even if model unfolding is restricted to only a few time steps. Another interesting feature of the BPTT method is that it can provide exact values of the gradient when not only the actual values of parameters are employed but also linear dynamic model outputs are unfolded back in time according to the following rule:

$$\hat{s}(n-na-i_1) = \frac{1}{\hat{a}_{na}}\Bigg(- \hat{s}(n-i_1) - \sum_{m=1}^{na-1} \hat{a}_m \hat{s}(n-m-i_1) + \sum_{m=1}^{nb} \hat{b}_m u(n-m-i_1)\Bigg), \tag{2.85}$$

where $i_1 = 1, \ldots, n-1$. The formula (2.85) can be used provided that $\hat{a}_{na} \neq 0$. In practice, if $\hat{a}_{na} = 0$ or $\hat{a}_{na} \simeq 0$, then $\hat{s}(n-na-i_1)$ does not influence or does not influence significantly $\hat{s}(n-i_1)$, and the unfolding step can be omitted. Note that, provided that $\hat{b}_{nb} \neq 0$, unfolding linear dynamic model outputs can be replaced with unfolding model inputs:

$$u(n-nb-i_1) = \frac{1}{\hat{b}_{nb}}\Bigg(\hat{s}(n-i_1) + \sum_{m=1}^{na} \hat{a}_m \hat{s}(n-m-i_1) - \sum_{m=1}^{nb-1} \hat{b}_m u(n-m-i_1)\Bigg). \tag{2.86}$$

In this case, if $\hat{b}_{nb} = 0$ or $\hat{b}_{nb} \simeq 0$, then $u(n-na-i_1)$ does not influence or does not influence significantly $\hat{s}(n-i_1)$, and the unfolding step can be omitted.

2.4.11 Computational complexity

Computational complexity of on-line learning algorithms for recurrent neural network models is commonly much higher than computational complexity of the on-line backpropagation algorithm and depends on the number of the recurrent nodes L. For example, for the fully connected recurrent network, computational complexity per one step of BPTT, truncated up to K time steps backwards, is $O(L^2K)$. Computational complexity of the real time recurrent learning algorithm of Williams and Zipser [167] is even much higher $O(L^4)$ [115]. Fortunately, the parallel neural network Wiener model contains

only one recurrent node, i.e., $L = 1$. This considerably reduces computational requirements of the examined learning algorithms. The evaluation of computational complexity per one step given below is expressed in terms of the polynomial orders na and nb, and the number of the nonlinear nodes M. For the truncated BBTT algorithm, computational complexity depends also on the number of unfolded time steps K. This evaluation of computational complexity is made under the following assumptions:

1. The number of bits of precision for all quantities is fixed.
2. The costs of storing and reading from computer memory can be neglected.
3. No distinction is made between the costs of different operations such as addition, multiplication, calculation of input-output transfer functions such as $\varphi(\cdot)$ or its first derivative.

The computation of the model output requires the function $\varphi(\cdot)$ to be calculated M times, and to compute the gradient of the model output, its first derivative has to be calculated M times. Both of these operations are included into computational costs of the weight update given in Table 2.1. Computational complexity of BPS is approximately two times higher than computational complexity of BPP. The application of the SM requires only $2na(na + nb)$ operations more than BPP. Computational complexity of the BPTT algorithm depends also on the number of unfolded time steps K and is higher than the complexity of the BPP algorithm by $2K(2na + nb)$ operations.

Table 2.1. Computational complexity of on-line learning algorithms

Algorithm	Computational complexity
BPS	$(31 + 2na)M + 5(na + nb) + 5$
BPP	$16M + 6(na + nb) + 3$
SM	$16M + 2na(na + nb) + 6(na + nb) + 3$
BPTT	$16M + 2K(2na + nb) + 6(na + nb) + 3$

2.5 Simulation example

In the simulation example, the second order Wiener system composed of a continuous linear dynamic system given by the pulse transfer function

$$G(s) = \frac{1}{6s^2 + 5s + 1} \tag{2.87}$$

was converted to discrete time, assuming a zero order hold on the input and the sampling interval 1s, leading to the following difference equation:

$$s(n) = 1.3231s(n–1) – 0.4346s(n – 2) + 0.0635u(n–1) + 0.0481u(n – 2). \tag{2.88}$$

The system contained also a nonlinear element (Fig. 2.10) given by

$$f\big(s(n)\big) = \begin{cases} -1 & \text{for } s(n) < -1.5 \\ s(n) + 0.5 & \text{for } -1.5 \leqslant s(n) \leqslant -0.5 \\ 0 & \text{for } \mid s(n) \mid < 0.5 \\ s(n) - 0.5 & \text{for } 0.5 \leqslant s(n) \leqslant 1.5 \\ 1 & \text{for } s(n) > 1.5 \end{cases} . \tag{2.89}$$

The system was driven by a sequence of 60000 random numbers uniformly distributed in $(-2, 2)$. The parallel neural network Wiener model was trained recursively using the steepest descent method, with the learning rate of 0.005, and calculating the gradient with the BPP, truncated BPTT, and SM algorithms. The nonlinear element model contained 25 nodes of the hyperbolic tangent activation. To compare convergence rates of the algorithms, a mean square error of moving averages defined as

$$J_I(n) = \begin{cases} \dfrac{1}{n} \displaystyle\sum_{j=1}^{n} \big(\hat{y}(j) - y(j)\big)^2 & \text{for } n \leqslant I \\ \dfrac{1}{I} \displaystyle\sum_{j=n-I+1}^{n} \big(\hat{y}(j) - y(j)\big)^2 & \text{for } n > I \end{cases} \tag{2.90}$$

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

$$P(n) = \frac{1}{4} \sum_{j=1}^{2} \big[(\hat{a}_j - a_j)^2 + (\hat{b}_j - b_j)^2\big], \tag{2.91}$$

$$F(n) = \frac{1}{100} \sum_{j=1}^{100} \big[\hat{f}(s(j)) - f(s(j), \mathbf{w})\big]^2, \tag{2.92}$$

where $\{s(j)\}$ is a testing sequence consisting of 100 linearly equally spaced values between -2 and 2. The simulation results for a noise-free Wiener system and a Wiener system disturbed by the additive output Gaussian noise $\mathcal{N}(0, \sigma_e)$ are summarized in Tables 2.2 – 2.4.

Noise-free case. The identification results are given in Table 2.2 and illustrated in Figs. 2.10 – 2.16. As shown in Table 2.2, the highest accuracy of nonlinear element identification, measured by the $F(60000)$ index, was achieved for the BPTT algorithm at $K = 6$. The best fitting of the linear dynamic model, expressed by the $P(60000)$ index, can be observed for the the BPTT algorithm, at $K = 4$. The moving averages index $J_{2000}(60000)$ that describes the accuracy of overall model identification also has its minimum for the

BPTT algorithm, at $K = 4$. The lowest accuracy of identification, measured by all four indices, was obtained for the BPP algorithm. The results obtained for the SM algorithm are more accurate in comparison with the results of the BPTT algorithm at $K = 1$ and less accurate than these at $K = 2, \ldots, 8$.

Noise case I. The results obtained at a high signal to noise ratio $SNR = 17.8$, defined as $SNR = \sqrt{\text{var}(y(n) - \varepsilon(n))/\text{var}(\varepsilon(n))}$, are a little less accurate than these for the noise-free case (Table 2.3). The minimum values of all four indices were obtained for the BPTT algorithm, i.e., $F(60000)$ at $K = 6$, $P(60000)$ at $K = 5$, $J_{2000}(60000)$ at $K = 4$, and $J_{60000}(60000)$ at $K = 3$ and 4.

Noise case II. The results are given in Table 2.4. In this case, training the neural network Wiener model at a low signal to noise ratio $SNR = 3.56$ was made using the following time-dependent learning rate:

$$\eta(n) = \frac{0.01}{\sqrt[4]{n}}. \tag{2.93}$$

In general, the BPP algorithm has the lowest convergence rate. This result can be explained by employing in BPP the approximate gradient instead of the true one. In the batch mode, the truncated BPTT algorithm is an approximation of the SM algorithm and the gradient approximation error decreases with an increase in K and approaches 0 for the fully unfolded model. In the sequential mode, the actual values of linear dynamic model parameters are used to unfold the model back in time. Therefore, as the truncated BPTT algorithm is not a simple approximation of the SM algorithm, the BPTT estimation results do not converge to the results obtained for the SM algorithm. Note that, unlike the SM algorithm, BPTT makes it possible to calculate the true value of the gradient if the linear dynamic model output $s(n)$ is unfolded back in time.

Pattern learning is usually more convenient and effective when the number of training patterns N is large, which is the case in the simulation example.

Another advantage of pattern learning is that it explores the parameter space in a stochastic way by its nature, thus preventing the learning process from getting trapped in a shallow local minimum [68]. In practice, the number of the available training patterns may be too small to achieve the minimum of the global cost function J after a single presentation of all N training patterns. In such a situation, the learning process can be continued with the same set of N training patterns presented cyclically. Each cycle of repeated pattern learning corresponds to one iteration (epoch) in the batch mode.

Table 2.2. Comparison of estimation accuracy, ($\sigma_e = 0$)

Algorithm	$J_{60000}(60000)$	$J_{2000}(60000)$	$F(60000)$	$P(60000)$
BPP	5.51×10^{-3}	1.29×10^{-3}	8.01×10^{-3}	2.07×10^{-3}
SM	1.40×10^{-3}	7.94×10^{-5}	3.53×10^{-4}	2.66×10^{-6}
BPTT, $K = 1$	2.17×10^{-3}	3.80×10^{-4}	8.05×10^{-4}	4.16×10^{-5}
BPTT, $K = 2$	1.34×10^{-3}	7.52×10^{-5}	2.37×10^{-4}	4.12×10^{-7}
BPTT, $K = 3$	1.19×10^{-3}	3.81×10^{-5}	1.15×10^{-4}	6.81×10^{-7}
BPTT, $K = 4$	1.19×10^{-3}	2.71×10^{-5}	4.68×10^{-5}	3.82×10^{-7}
BPTT, $K = 5$	1.22×10^{-3}	2.78×10^{-5}	4.66×10^{-5}	9.66×10^{-7}
BPTT, $K = 6$	1.25×10^{-3}	2.86×10^{-5}	4.37×10^{-5}	1.74×10^{-6}
BPTT, $K = 7$	1.27×10^{-3}	2.97×10^{-5}	4.54×10^{-5}	1.35×10^{-6}
BPTT, $K = 8$	1.31×10^{-3}	3.99×10^{-5}	9.23×10^{-5}	2.00×10^{-6}
BPTT, $K = 9$	1.35×10^{-3}	5.25×10^{-5}	9.12×10^{-5}	4.10×10^{-6}
BPTT, $K = 10$	1.38×10^{-3}	9.62×10^{-5}	3.50×10^{-4}	3.05×10^{-6}
BPTT, $K = 11$	1.38×10^{-3}	9.03×10^{-5}	4.25×10^{-4}	2.98×10^{-6}
BPTT, $K = 12$	1.39×10^{-3}	8.46×10^{-5}	4.15×10^{-4}	3.02×10^{-6}
BPTT, $K = 13$	1.39×10^{-3}	8.20×10^{-5}	3.92×10^{-4}	2.89×10^{-6}
BPTT, $K = 14$	1.39×10^{-3}	8.08×10^{-5}	3.80×10^{-4}	2.86×10^{-6}
BPTT, $K = 15$	1.39×10^{-3}	8.03×10^{-5}	3.70×10^{-4}	2.85×10^{-6}

Table 2.3. Comparison of estimation accuracy, ($\sigma_e = 0.02$, $SNR = 17.8$)

Algorithm	$J_{60000}(60000)$	$J_{2000}(60000)$	$F(60000)$	$P(60000)$
BPP	5.99×10^{-3}	1.72×10^{-3}	6.01×10^{-3}	2.15×10^{-3}
SM	1.93×10^{-3}	8.75×10^{-5}	2.30×10^{-4}	1.32×10^{-5}
BPTT, $K = 1$	2.66×10^{-3}	8.97×10^{-4}	1.04×10^{-3}	7.77×10^{-5}
BPTT, $K = 2$	1.82×10^{-3}	5.71×10^{-4}	2.22×10^{-4}	5.84×10^{-6}
BPTT, $K = 3$	1.67×10^{-3}	5.32×10^{-4}	1.87×10^{-4}	2.80×10^{-6}
BPTT, $K = 4$	1.67×10^{-3}	5.31×10^{-4}	9.49×10^{-5}	1.97×10^{-6}
BPTT, $K = 5$	1.72×10^{-3}	5.45×10^{-4}	6.32×10^{-5}	1.45×10^{-6}
BPTT, $K = 6$	1.75×10^{-3}	5.59×10^{-4}	5.68×10^{-5}	5.93×10^{-6}
BPTT, $K = 7$	1.77×10^{-3}	5.82×10^{-4}	6.23×10^{-5}	4.49×10^{-6}
BPTT, $K = 8$	1.81×10^{-3}	6.04×10^{-4}	9.88×10^{-5}	5.23×10^{-6}
BPTT, $K = 9$	1.86×10^{-3}	6.15×10^{-4}	1.04×10^{-4}	1.04×10^{-5}
BPTT, $K = 10$	1.90×10^{-3}	7.68×10^{-4}	2.82×10^{-4}	1.17×10^{-5}
BPTT, $K = 11$	1.91×10^{-3}	8.61×10^{-4}	3.23×10^{-4}	1.26×10^{-5}
BPTT, $K = 12$	1.91×10^{-3}	9.24×10^{-4}	2.92×10^{-4}	1.36×10^{-5}
BPTT, $K = 13$	1.92×10^{-3}	9.94×10^{-4}	2.62×10^{-4}	1.52×10^{-5}
BPTT, $K = 14$	1.93×10^{-3}	1.07×10^{-3}	2.37×10^{-4}	1.70×10^{-5}
BPTT, $K = 15$	1.93×10^{-3}	1.15×10^{-3}	2.19×10^{-4}	1.85×10^{-5}

Table 2.4. Comparison of estimation accuracy, ($\sigma_e = 0.1$, $SNR = 3.56$)

Algorithm	$J_{60000}(60000)$	$J_{2000}(60000)$	$F(60000)$	$P(60000)$
BPP	1.94×10^{-2}	1.55×10^{-2}	2.10×10^{-2}	2.16×10^{-2}
SM	1.30×10^{-2}	1.09×10^{-2}	1.36×10^{-3}	1.07×10^{-3}
BPTT, $K = 1$	1.41×10^{-2}	1.12×10^{-2}	5.26×10^{-3}	2.79×10^{-3}
BPTT, $K = 2$	1.28×10^{-2}	1.07×10^{-2}	2.94×10^{-3}	1.18×10^{-3}
BPTT, $K = 3$	1.25×10^{-2}	1.06×10^{-2}	1.39×10^{-3}	7.23×10^{-4}
BPTT, $K = 4$	1.25×10^{-2}	1.06×10^{-2}	7.65×10^{-4}	7.93×10^{-4}
BPTT, $K = 5$	1.26×10^{-2}	1.07×10^{-2}	5.73×10^{-4}	7.55×10^{-4}
BPTT, $K = 6$	1.27×10^{-2}	1.08×10^{-2}	6.56×10^{-4}	8.89×10^{-4}
BPTT, $K = 7$	1.27×10^{-2}	1.08×10^{-2}	6.45×10^{-4}	8.69×10^{-4}
BPTT, $K = 8$	1.27×10^{-2}	1.08×10^{-2}	7.34×10^{-4}	8.83×10^{-4}
BPTT, $K = 9$	1.28×10^{-2}	1.08×10^{-2}	1.12×10^{-3}	9.23×10^{-4}
BPTT, $K = 10$	1.29×10^{-2}	1.08×10^{-2}	1.55×10^{-3}	9.69×10^{-4}
BPTT, $K = 11$	1.29×10^{-2}	1.08×10^{-2}	1.46×10^{-3}	1.01×10^{-3}
BPTT, $K = 12$	1.29×10^{-2}	1.08×10^{-2}	1.42×10^{-3}	1.05×10^{-3}
BPTT, $K = 13$	1.29×10^{-2}	1.08×10^{-2}	1.42×10^{-3}	1.04×10^{-3}
BPTT, $K = 14$	1.29×10^{-2}	1.09×10^{-2}	1.41×10^{-3}	1.05×10^{-3}
BPTT, $K = 15$	1.29×10^{-2}	1.09×10^{-2}	1.41×10^{-3}	1.06×10^{-3}

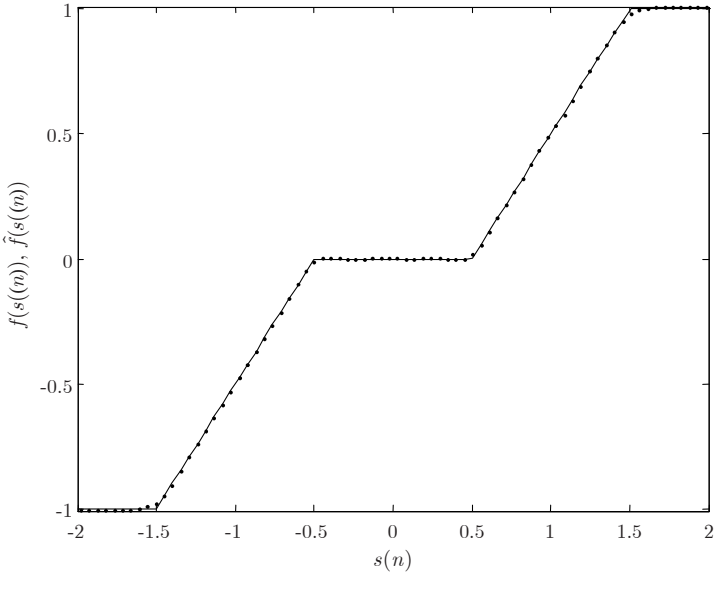

Fig. 2.10. True (solid line) and estimated (dotted line) nonlinear functions, (BPTT, $K = 4$)

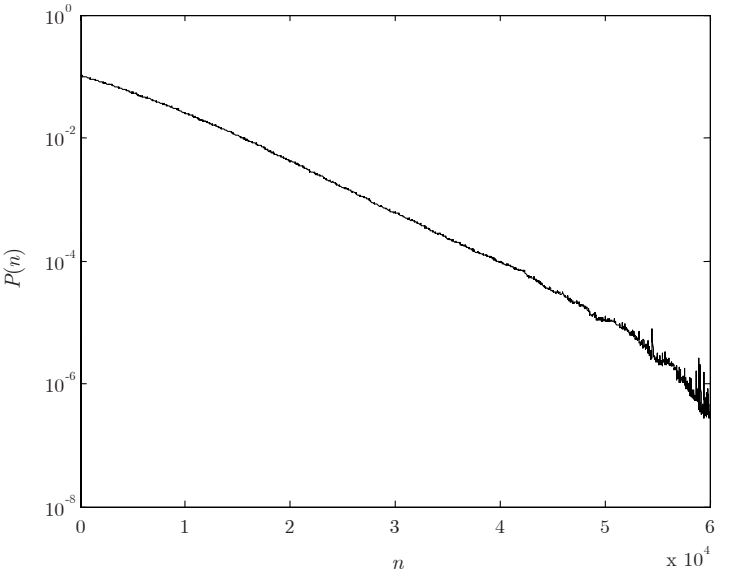

Fig. 2.11. Convergence rate of linear model parameters, (BPTT, $K = 4$)

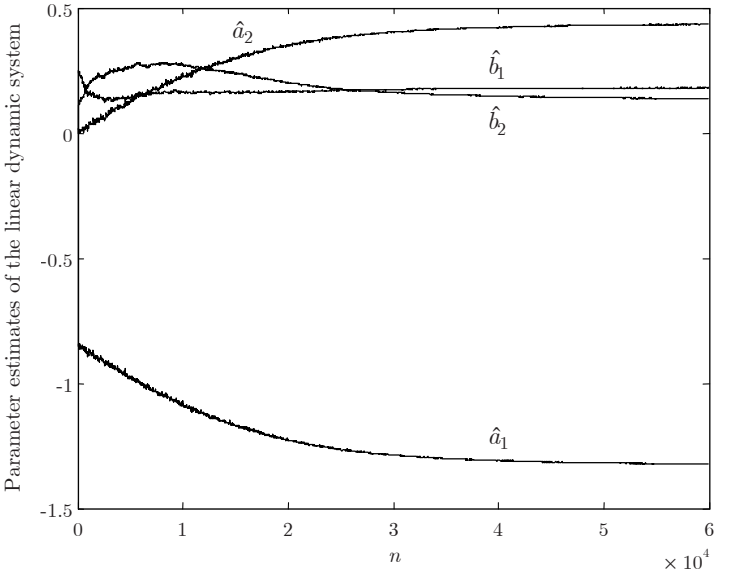

Fig. 2.12. Evolution of linear model parameters, (BPTT, $K = 4$)

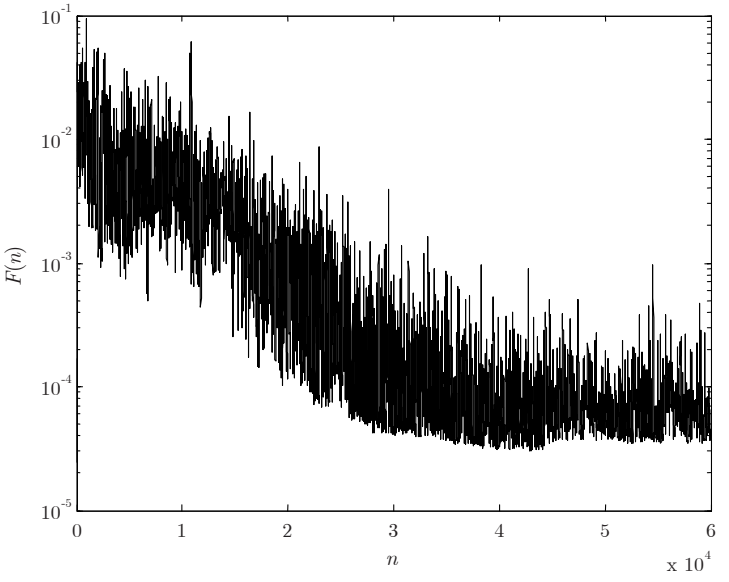

Fig. 2.13. Convergence rate of the nonlinear element model, (BPTT, $K = 4$)

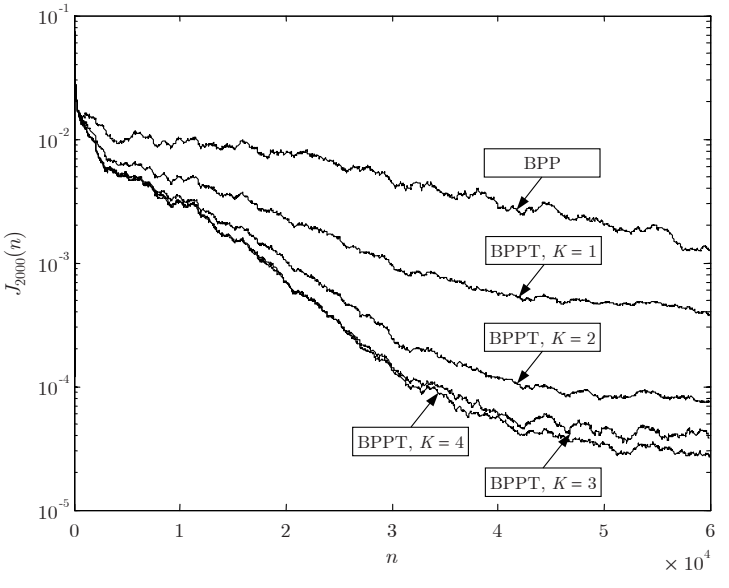

Fig. 2.14. Convergence rate of the BPP and BPPT algorithms

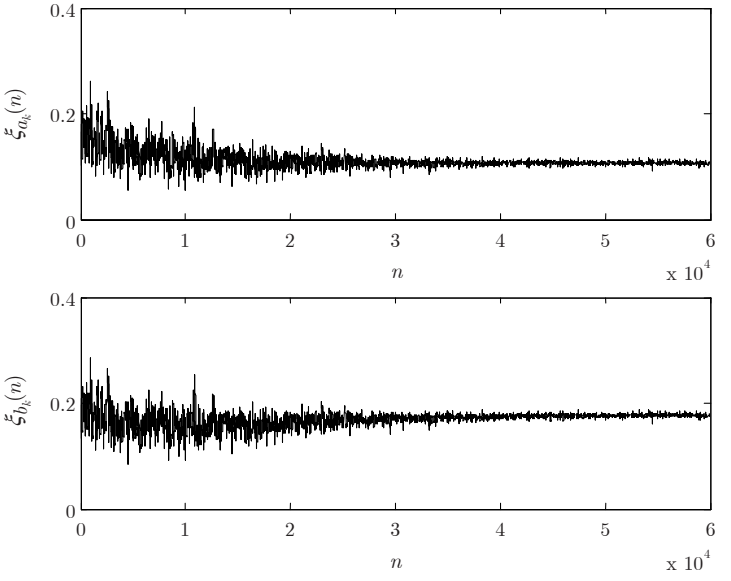

Fig. 2.15. Evolution of gradient calculation accuracy degrees, (BPTT, $K = 4$)

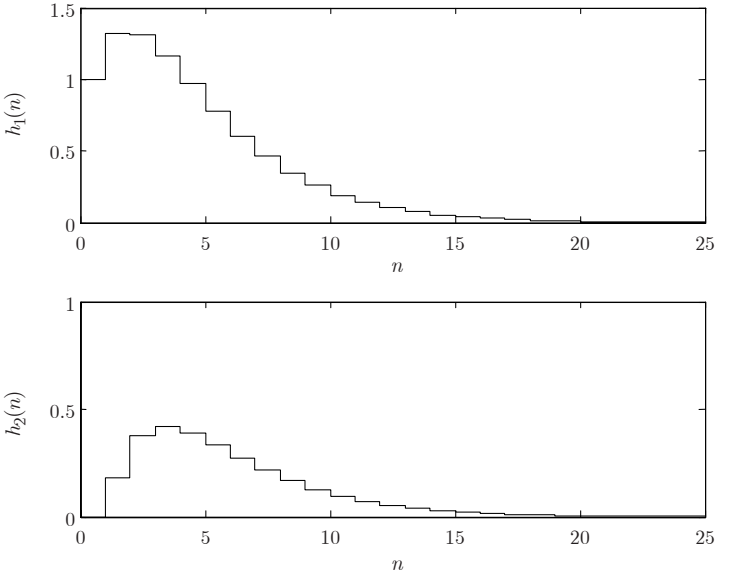

Fig. 2.16. Impulse responses of sensitivity models and the linear dynamic model, (BPTT, $K = 4$)

2.6 Two-tank system example

A laboratory two-tank system with transport delay, shown in Fig. 2.17, was used as a source of data for a real process example [89]. A system actuator, consisting of the dc motor M, the diaphragm water pump P, and the bypass valve V, was identified. The bypass valve V was left in an open position, introducing a dead zone into the system steady-state characteristic, see Fig. 2.21 for the characteristic obtained via an experiment with recording a system response to a stairs-wise input. The parameters of both a parallel neural network Wiener model and a linear autoregressive with an exogenous input (ARX) model were determined based on a sequence of 6000 input and output data recorded in another open loop experiment controlled by a computer at the sampling interval of 1s. The motor M was driven by a control signal obtained from a pseudorandom generator of uniform distribution and transformed in such a way that each pseudorandom value was kept constant for ten sampling intervals constituting ten successive control values. The flow rate of water, measured with the Venturi flow meter F, was chosen as a system output and the squared control signal was used as the model input. Before identification, the input and output data were scaled to zero mean values and variances 1.

First, the parameters of the ARX model were estimated with the least squares method. Based on the results of ARX model estimation and applying the Akaike final prediction error criterion [112], the second order model with delay 1 was chosen, i.e., $na = 2$, $nb = 2$. Then the parallel neural network Wiener model containing 30 nonlinear nodes of the hyperbolic tangent activation function was trained recursively with BPP, the SM, and truncated BPTT at $K = 1, \ldots, 15$ with the input-output data set processed cyclically five times. Finally, the trained neural network Wiener models were tested with another set of 3000 input-output data. The obtained results, i.e., the values of $J_{3000}(30000)$ for the training set and $J_{3000}(3000)$ for the testing set, are given in Table 2.5 and illustrated in Figs 2.18 – 2.21. The inspection of the impulse responses $h_1(n)$ and $h_2(n)$, presented in Figs 2.18 and 2.19, shows that both of them decrease to practically negligible small values at $n = 12$. Therefore, according to Theorems 2.1 and 2.2, no significant improvement can be expected with unfolding the model for more than 12 time steps – see the results in Table 2.5 for a confirmation of this conclusion. As can be expected, the results obtained for the Wiener model with the SM and BPTT at $K = 2, \ldots, 15$ are better than the result for the ARX model. Moreover, contrary to the simulation example, a gradual improvement in model fitting can be seen (Fig. 2.20) with an increase in K which can be explained by applying a sufficiently low learning rate of 0.001. The operation of the diaphragm water pump at constant excitation causes some periodic changes of the water flow rate. These changes can be considered as additive output disturbances and, as a result, the cost function achieves much higher values than in the case of the noise-free simulation example, see Tables 2.2 and 2.5.

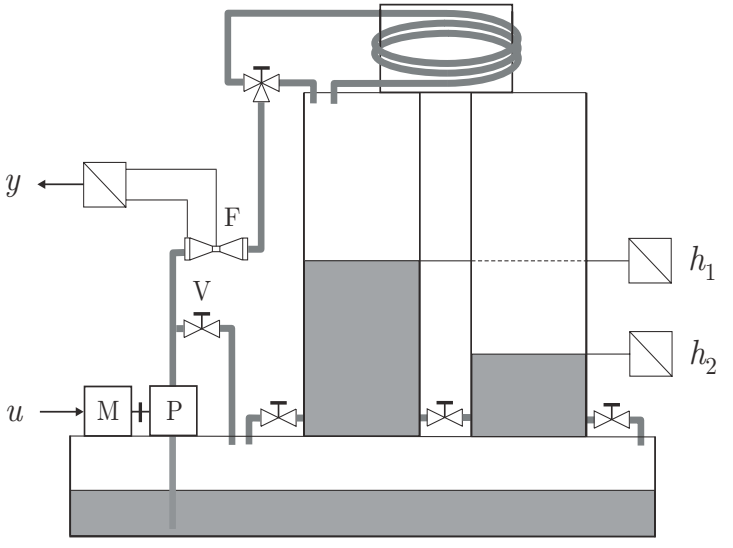

Fig. 2.17. Two-tank system with transport delay

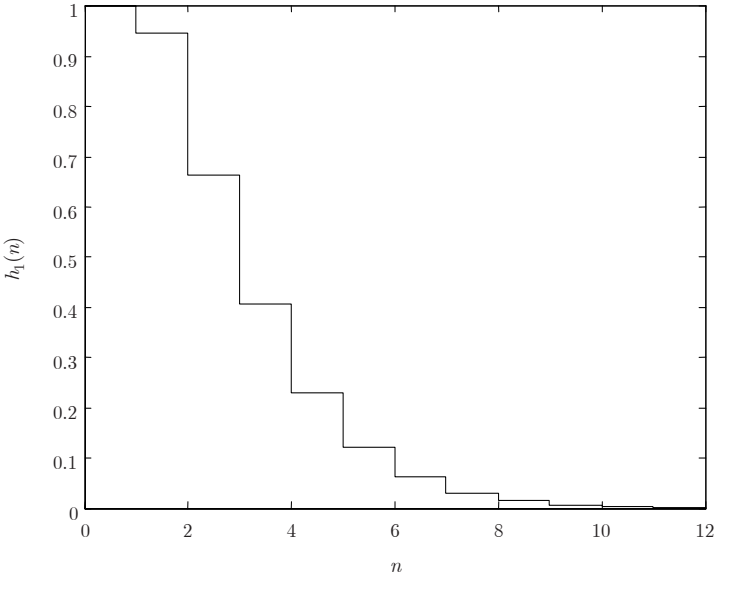

Fig. 2.18. Two-tank system. Impulse response of the system $\hat{H}_1(q^{-1})$, $K=15$

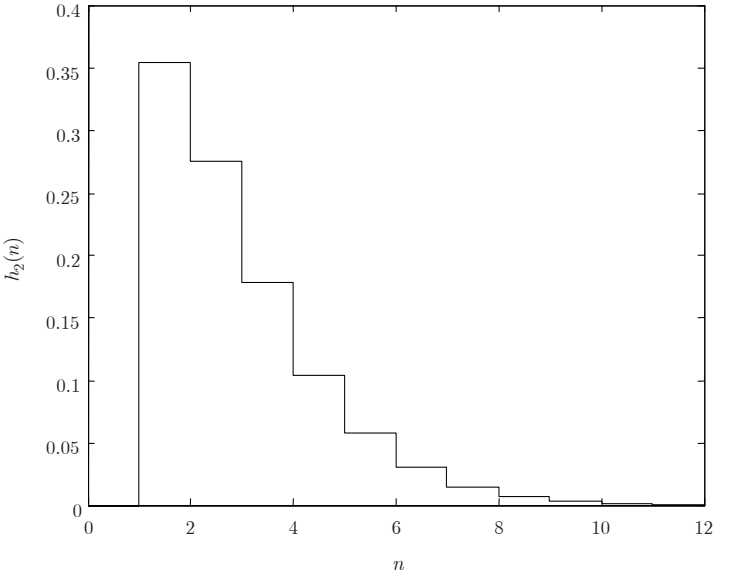

Fig. 2.19. Two-tank system. Impulse response of the system $\hat{H}_2(q^{-1})$, $K=15$

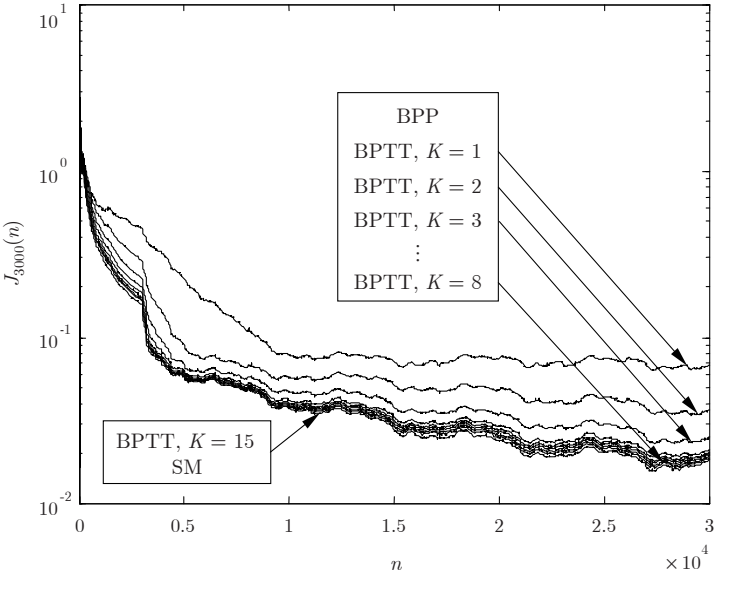

Fig. 2.20. Two-tank system. Training the Wiener model with BPP, BPTT, and the SM

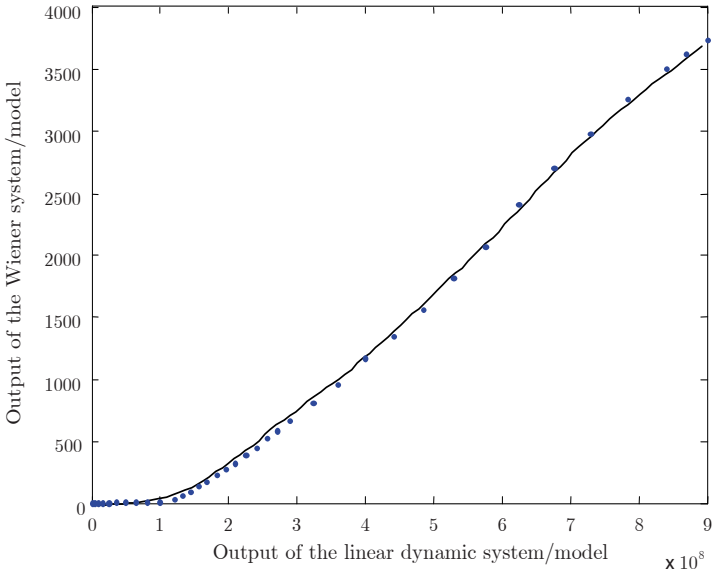

Fig. 2.21. Two-tank system. The true (dotted line) and estimated (solid line) nonlinear characteristics, BPTT, $K = 15$

Table 2.5. Two-tank system. Comparison of estimation accuracy

Algorithm	$J_{3000}(30000)$	$J_{3000}(3000)$
ARX		0.0406
BPP	0.0690	0.0860
SM	0.0176	0.0196
BPTT, $K = 1$	0.0374	0.0461
BPTT, $K = 2$	0.0253	0.0306
BPTT, $K = 3$	0.0210	0.0246
BPTT, $K = 4$	0.0200	0.0229
BPTT, $K = 5$	0.0194	0.0221
BPTT, $K = 6$	0.0188	0.0213
BPTT, $K = 7$	0.0183	0.0206
BPTT, $K = 8$	0.0181	0.0202
BPTT, $K = 9$	0.0179	0.0200
BPTT, $K = 10$	0.0178	0.0199
BPTT, $K = 11$	0.0178	0.0198
BPTT, $K = 12$	0.0178	0.0198
BPTT, $K = 13$	0.0178	0.0197
BPTT, $K = 14$	0.0177	0.0197
BPTT, $K = 15$	0.0177	0.0197

It is well known that parallel models are more suitable for the identification of noise-corrupted systems as they do not use past system outputs, which are noise-corrupted, to calculate the model output. Clearly, the parallel neural network Wiener model is also well suited for the identification of Wiener systems with additive white output noise. On the other hand, from the fact that series-parallel models use past system outputs, it follows that these models are not able to describe systems with additive white output noise appropriately, and the series-parallel neural network Wiener model is not suitable for the identification of Wiener systems additive disturbances either. The noise interfering with the system makes the identification a stochastic optimization problem.
Therefore, a large number of input-output data should be used to achieve a required level of model accuracy for systems with a high level of output disturbances. Moreover, small or slowly decreasing values of the learning rate η are necessary to achieve the convergence of steepest descent algorithms.

2.7 Prediction error method

It is well known that steepest descent algorithms have a linear convergence rate and can be quite slow in many practical cases. A faster convergence rate can be achieved with methods which are based on a second-order approximation of the optimization criterion to determine the search direction. An alternative to steepest descent methods is the prediction error (PE) method or its pattern version – the recursive prediction error (RPE) method. The PE and RPE algorithms have superior convergence properties in comparison with steepest descent algorithms as they use the approximate Hessjan to compute the search direction. The properties of the PE and RPE algorithms for different neural network models are discussed in [29, 127]. RPE algorithms for Wiener models with known static nonlinearity or the nonlinear model approximated with a piecewise linear function were analyzed by Wigren [165, 166].
A batch PE algorithm for Wiener models with a polynomial model of the nonlinear element was used by Norquay *et al.* [128]. A sequential version of the PE algorithm for the training of a SISO neural network Wiener model was proposed in [85]. In both these algorithms, the gradient of the model output w.r.t. the parameters of its linear part is computed with the SM.

2.7.1 Recursive prediction error learning algorithm

The identification problem can be formulated as follows: Given a set of input and output data $\{Z^N = \{u(n), y(n)\},\ n = 1, \ldots, N,\}$ and a candidate neural network Wiener model, estimate the parameters $\boldsymbol{\theta}$ of the model so that the predictions $\hat{y}(n|n-1)$ of the system output are close to the system output $y(n)$ in the sense of the following mean square error criterion:

$$J(\boldsymbol{\theta}, Z^N) = \frac{1}{2N}\sum_{n=1}^{N}\big(y(n) - \hat{y}(n|n-1)\big)^2. \tag{2.94}$$

A solution to this problem, for nonlinearly parameterized models, is a gradient technique based on the approximation of the Hessjan known as the prediction error method. The PE method is a batch optimization method of the Gauss-Newton type. The RPE algorithm is a recursive counterpart of the PE method. Both the PE and RPE algorithms are guaranteed to converge to a local minimum of $J(\boldsymbol{\theta}, Z^N)$ with the probability 1 as $N \to \infty$ [29]. Given the gradient

$$\begin{aligned}\boldsymbol{\psi}(n) = \Bigg[&\frac{\partial\hat{y}(n|n-1)}{\partial\hat{a}_1}\cdots\frac{\partial\hat{y}(n|n-1)}{\partial\hat{a}_{na}}\ \frac{\partial\hat{y}(n|n-1)}{\partial\hat{b}_1}\cdots\frac{\partial\hat{y}(n|n-1)}{\partial\hat{b}_{nb}}\\ &\frac{\partial\hat{y}(n|n-1)}{\partial w_{10}^{(1)}}\cdots\frac{\partial\hat{y}(n|n-1)}{\partial w_{M1}^{(1)}}\ \frac{\partial\hat{y}(n|n-1)}{\partial w_{10}^{(2)}}\cdots\frac{\partial\hat{y}(n|n-1)}{\partial w_{1M}^{(2)}}\Bigg]^T\end{aligned} \tag{2.95}$$

of the model output w.r.t. the parameter vector

$$\boldsymbol{\theta} = \big[\hat{a}_1\ldots\hat{a}_{na}\ \hat{b}_1\ldots\hat{b}_{nb}\ w_{10}^{(1)}\ldots w_{M1}^{(1)} w_{10}^{(2)}\ldots w_{1M}^{(2)}\big]^T, \tag{2.96}$$

the RPE algorithm can be expressed as [29, 127]:

$$\mathbf{K}(n) = \frac{\mathbf{P}(n-1)\boldsymbol{\psi}(n)}{1+\boldsymbol{\psi}^T(n)\mathbf{P}(n-1)\boldsymbol{\psi}(n)}, \tag{2.97}$$

$$\boldsymbol{\theta}(n) = \boldsymbol{\theta}(n-1) + \mathbf{K}(n)\big(y(n) - \hat{y}(n|n-1)\big), \tag{2.98}$$

$$\mathbf{P}(n) = \mathbf{P}(n-1) - \mathbf{K}(n)\boldsymbol{\psi}^T(n)\mathbf{P}(n-1). \tag{2.99}$$

The RPE algorithm is an alternative to the well-known steepest descent methods and has much better convergence properties and lower memory requirements. Computational complexity of the RPE algorithm is much higher and it is not recommended for problems with a large number of estimated parameters. Note that replacing $\mathbf{K}(n)$ with $\eta\boldsymbol{\psi}(n)$ in (2.98), where η denotes the learning rate, results in the steepest descent algorithm.
In some cases, it may be useful to apply algorithms that combine quasi-Newton methods and the steepest descent approach. In the RPE and SM algorithm, used in the example below for comparison, a search direction is calculated as a sum of the search directions of the RPE method and the SM.

2.7.2 Pneumatic valve simulation example

As a source of data for testing the RPE algorithm and comparing it with the SM, and the RPE and SM algotithm, a model of a pneumatic valve was used [165]. The dynamic balance between the pneumatic control signal $u(n)$ applied

to the valve stem, a counteractive spring force and friction is described by the linear dynamics

$$\begin{aligned} s(n) = {} & 1.4138s(n-1) - 0.6065s(n-2) + 0.1044u(n-1) \\ & + 0.0833u(n-2). \end{aligned} \tag{2.100}$$

The flow through the valve is a nonlinear function of the valve stem position $s(n)$:

$$f\big(s(n)\big) = \frac{0.3163s(n)}{\sqrt{0.1 + 0.9s^2(n)}}. \tag{2.101}$$

It is assumed that the model output is additively disturbed by a zero-mean discrete white Gaussian noise $\varepsilon(n)$ with the standard deviation $\sigma_\varepsilon = 0.01$:

$$y(n) = f\big(s(n)\big) + \varepsilon(n). \tag{2.102}$$

A sequence of 10000 pseudorandom numbers, uniformly distributed in $(-1, 1)$, was used as the model input. Based on the simulated input-output data, the parallel neural network Wiener model containing 10 nonlinear nodes of the hyperbolic tangent activation function was trained with the RPE, SM, and RPE and SM algorithms.
The identification results are illustrated in Figs 2.22 – 2.25 [87]. To compare estimation accuracy of linear system parameters and the nonlinear function $f(\cdot)$, the indices (2.91) and (2.92) are used.

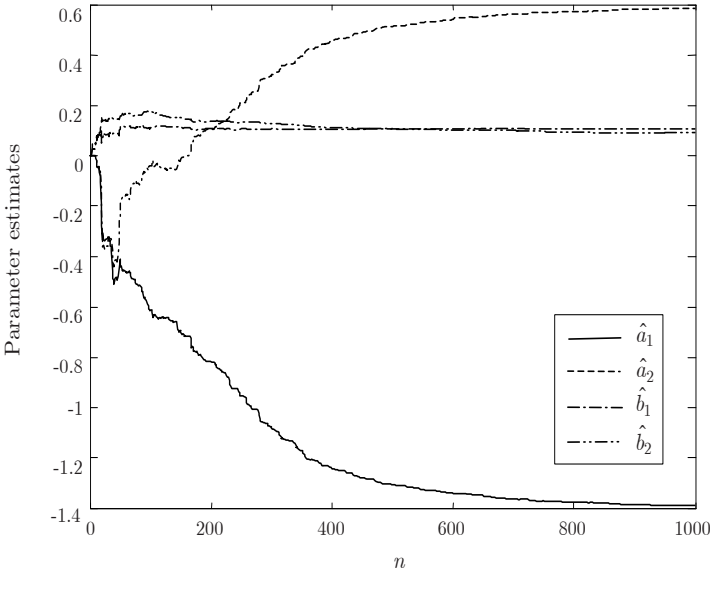

Fig. 2.22. RPE algorithm. Evolution of parameter estimates of the linear element

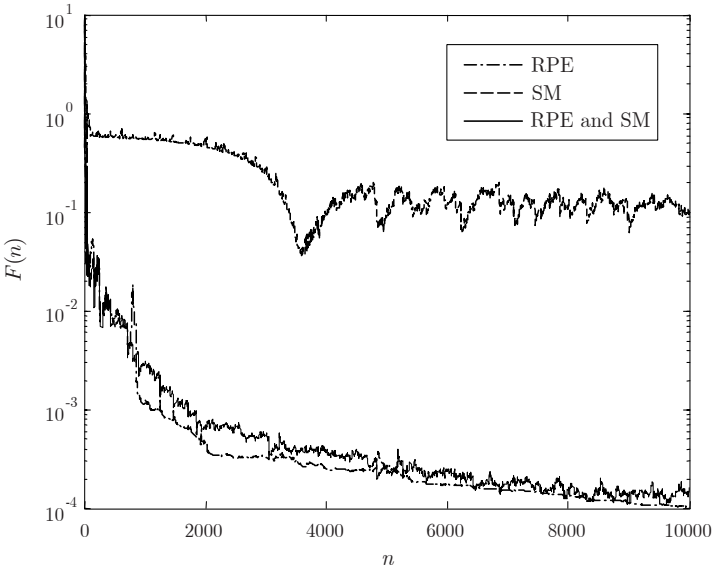

Fig. 2.23. Convergence of the linear element model

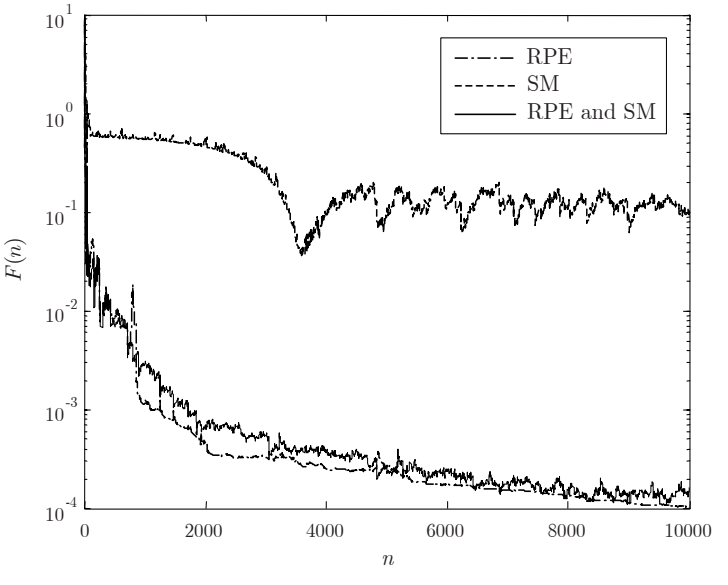

Fig. 2.24. Convergence of the nonlinear element model

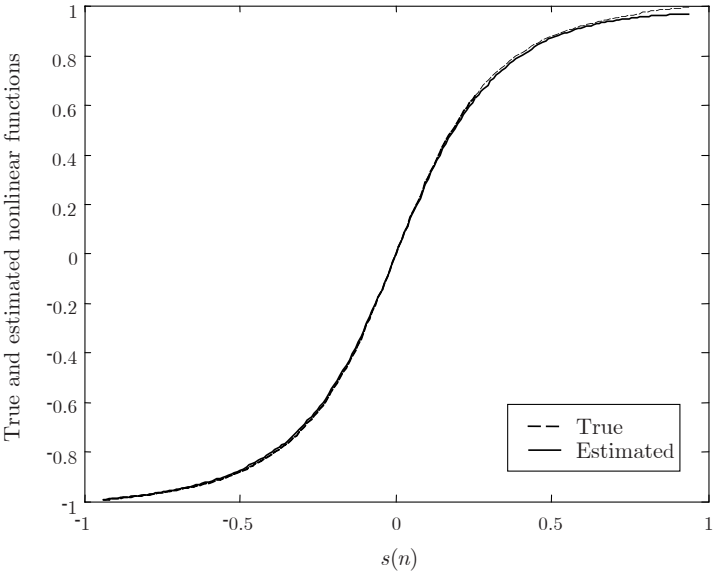

Fig. 2.25. RPE algorithm. True and estimated nonlinear characteristics

2.8 Summary

In this chapter, various steepest descent learning algorithms for neural network Wiener models have been derived, analyzed and compared in a unified framework. For the truncated BPTT method, it has been shown that the accuracy of gradient calculation depends on impulse responses of the linear dynamic model and its sensitivity models. The number of unfolded time steps can be determined on the basis of the number of time steps necessary for the impulse response of sensitivity models to decrease to negligible small values. The application of these algorithms to both simulated and real data systems has been demonstrated as well.

The parallel neural network Wiener model is simpler than its series-parallel counterpart as it does not contain the inverse model of the static nonlinear element. Another important advantage of the parallel neural network model is that its training with the BPP algorithm requires almost half the computational burden for the training of the series-parallel model with the BPS algorithm. Applying the SM or truncated BPTT algorithms, one can expect a higher convergence rate, in comparison with the BPP algorithm, as a more accurate gradient approximation is used. In comparison with gradient computation in feedforward neural networks, gradient computation in recurrent networks is commonly much more computationally intensive. That is not the case for the parallel neural network Wiener model, as both the SM and truncated BPTT algorithms require only a little more computational effort than

the BPP algorithm. This comes from the fact that, in all these algorithms, $3M + 1$ parameters of the nonlinear element model are adjusted in the same way. The algorithms update the parameters of the linear dynamic model in different manners but the number of the parameters $na + nb$ is commonly not greater than a few.

Both the simulation example and the real process example show the lowest convergence rate of the BPP algorithm. In the simulation examples, the highest accuracy of both the linear dynamic model and the nonlinear element has been obtained using the truncated BPTT algorithm and unfolding the linear dynamic model only a few steps back in time.

The RPE identification algorithm has been also extended to training recurrent neural network Wiener models. The calculation of the gradient is performed with the SM and it does not require much more computation than the BPP algorithm. In comparison with steepest descent methods, the RPE algorithm has much better convergence properties at the expense of higher computational complexity. The RPE algorithm can be very useful when the system output is disturbed additively by the white noise. Note that in such cases, the steepest descent algorithms require a small step size to achieve convergence to a local minima of the performance function.

Although only sequential learning versions of the basic steepest descent algorithm for neural network Wiener models are described and discussed in this chapter, their batch counterparts can be derived easily given the rules for gradient computation. Also, the implementation of other gradient methods such as variable metric methods, conjugate gradient methods, or the RLS learning algorithms (a review and discussion on their hardware implementation using the systolic technology is given by Rutkowski [144]) is straightforward.

2.9 Appendix 2.1. Gradient derivation of truncated BPTT. SISO Wiener models

Assume that the parallel SISO Wiener model is K times unfolded back in time. Let $\partial^+\hat{s}(n)/\partial\hat{a}_k$ and $\partial^+\hat{s}(n)/\partial\hat{b}_k$ denote partial derivatives calculated for the model unfolded back in time for K time steps, and $\partial\hat{s}(n)/\partial\hat{a}_k$ and $\partial\hat{s}(n)/\partial\hat{b}_k$ – partial derivatives calculated without taking into account the dependence of past model outputs on $\hat{a}_k$ and $\hat{b}_k$, respectively. The differentiation of (2.15) gives

$$\begin{aligned}\frac{\partial^+\hat{s}(n)}{\partial\hat{a}_k} &= \frac{\partial\hat{s}(n)}{\partial\hat{a}_k} + \sum_{i_1=1}^{K}\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)}\frac{\partial\hat{s}(n-i_1)}{\partial\hat{a}_k} \\ &= -\hat{s}(n-k) - \sum_{i_1=1}^{K}\hat{s}(n-k-i_1)\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)},\end{aligned} \tag{2.103}$$

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

$$\begin{aligned}\frac{\partial^+\hat{s}(n)}{\partial\hat{b}_k} &= \frac{\partial\hat{s}(n)}{\partial\hat{b}_k} + \sum_{i_1=1}^{K}\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)}\frac{\partial\hat{s}(n-i_1)}{\partial\hat{b}_k} \\ &= u(n-k) + \sum_{i_1=1}^{K}u(n-k-i_1)\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)},\end{aligned} \tag{2.104}$$

where $k = 1, \ldots, nb$. From (2.15), it follows that partial derivatives of the actual output of the linear dynamic model $\hat{s}(n)$ w.r.t. the delayed outputs $\hat{s}(n-1), \ldots, \hat{s}(n-K)$ are

$$\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)} = -\sum_{m=1}^{na}\hat{a}_m\frac{\partial\hat{s}(n-m)}{\partial\hat{s}(n-i_1)} = -\sum_{m=1}^{na}\hat{a}_m\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1+m)}, \tag{2.105}$$

where $i_1 = 1, \ldots, K$, and $\partial\hat{s}(n-m)/\partial\hat{s}(n-i_1) = 0$ for $m > i_1$. Finally, to obtain the partial derivatives $\partial\hat{y}(n)/\partial\hat{a}_k$ and $\partial\hat{y}(n)/\partial\hat{b}_k$, the partial derivatives $\partial\hat{s}(n)/\partial\hat{a}_k$ and $\partial\hat{s}(n)/\partial\hat{b}_k$ in (2.48) and (2.49) are replaced with $\partial^+\hat{s}(n)/\partial\hat{a}_k$ and $\partial^+\hat{s}(n)/\partial\hat{b}_k$:

$$\frac{\partial\hat{y}(n)}{\partial\hat{a}_k} = \left(-\hat{s}(n-k) - \sum_{i_1=1}^{K}\hat{s}(n-k-i_1)\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)}\right)\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}, \tag{2.106}$$

$$\frac{\partial\hat{y}(n)}{\partial\hat{b}_k} = \left(u(n-k) + \sum_{i_1=1}^{K}u(n-k-i_1)\frac{\partial\hat{s}(n)}{\partial\hat{s}(n-i_1)}\right)\frac{\partial\hat{y}(n)}{\partial\hat{s}(n)}. \tag{2.107}$$

2.10 Appendix 2.2. Gradient derivation of truncated BPTT. MIMO Wiener models

Assume that the parallel MIMO Wiener model is K times unfolded back in time. Let $\partial^+\hat{s}_l(n)/\partial\hat{a}_{rp}^{(k)}$ and $\partial^+\hat{s}_l(n)/\partial\hat{b}_{rp}^{(k)}$ denote partial derivatives calculated for the model unfolded back in time for K time steps, and $\partial\hat{s}_l(n)/\partial\hat{a}_{rp}^{(k)}$ and $\partial\hat{s}_l(n)/\partial\hat{b}_{rp}^{(k)}$ – partial derivatives calculated without taking into account the dependence of past model outputs on $\hat{a}_{rp}^{(k)}$ and $\hat{b}_{rp}^{(k)}$, respectively. The differentiation of (2.71) gives

$$\begin{aligned}\frac{\partial^+\hat{s}_l(n)}{\partial\hat{a}_{rp}^{(k)}} &= \frac{\partial\hat{s}_l(n)}{\partial\hat{a}_{rp}^{(k)}} + \sum_{i_1=1}^{K}\sum_{i_2=1}^{ns}\frac{\partial\hat{s}_l(n)}{\partial\hat{s}_{i_2}(n-i_1)}\frac{\partial\hat{s}_{i_2}(n-i_1)}{\partial\hat{a}_{rp}^{(k)}} \\ &= -\delta_{lr}\hat{s}_p(n-k) - \sum_{i_1=1}^{K}\hat{s}_p(n-k-i_1)\frac{\partial\hat{s}_l(n)}{\partial\hat{s}_r(n-i_1)},\end{aligned} \tag{2.108}$$

where

$$\delta_{lr} = \begin{cases} 1 & \text{for } l = r \\ 0 & \text{for } l \neq r \end{cases}, \tag{2.109}$$

where $l = 1,\ldots,ns$, $r = 1,\ldots,ns$, $p = 1,\ldots,ns$, $k = 1,\ldots,na$,

$$\begin{aligned}\frac{\partial^+\hat{s}_l(n)}{\partial\hat{b}_{rp}^{(k)}} &= \frac{\partial\hat{s}_l(n)}{\partial\hat{b}_{rp}^{(k)}} + \sum_{i_1=1}^{K}\sum_{i_2=1}^{ns}\frac{\partial\hat{s}_l(n)}{\partial\hat{s}_{i_2}(n-i_1)}\frac{\partial\hat{s}_{i_2}(n-i_1)}{\partial\hat{b}_{rp}^{(k)}} \\ &= \delta_{lr}u_p(n-k) + \sum_{i_1=1}^{K}u_p(n-k-i_1)\frac{\partial\hat{s}_l(n)}{\partial\hat{s}_r(n-i_1)},\end{aligned} \tag{2.110}$$

where $l = 1,\ldots,ns$, $r = 1,\ldots,ns$, $p = 1,\ldots,nu$, $k = 1,\ldots,nb$. From (2.71), it follows that partial derivatives of the actual lth output of the linear dynamic model w.r.t. its delayed rth output are

$$\frac{\partial\hat{s}_l(n)}{\partial\hat{s}_r(n-i_1)} = -\sum_{m=1}^{na}\sum_{m_1=1}^{ns}\hat{a}_{lm_1}^{(m)}\frac{\partial\hat{s}_{m_1}(n-m)}{\partial\hat{s}_r(n-i_1)} = -\sum_{m=1}^{na}\hat{a}_{lm_1}^{(m)}\frac{\partial\hat{s}_{m_1}(n)}{\partial\hat{s}_r(n-i_1+m)}, \tag{2.111}$$

where

$$\frac{\partial\hat{s}_{m_1}(n-m)}{\partial\hat{s}_r(n-i_1)} = 0 \text{ for } m > i_1 \text{ or } m = i_1 \text{ and } m_1 \neq r. \tag{2.112}$$

Finally, partial derivatives of the tth model output w.r.t. the parameters $\hat{a}_{rp}^{(k)}$ and $\hat{b}_{rp}^{(k)}$ are

$$\frac{\partial\hat{y}_t(n)}{\partial\hat{a}_{rp}^{(k)}} = \left(-\delta_{lr}\hat{s}_p(n-k) - \sum_{i_1=1}^{K}\hat{s}_p(n-k-i_1)\frac{\partial\hat{s}_l(n)}{\partial\hat{s}_r(n-i_1)}\right)\frac{\partial\hat{y}_t(n)}{\hat{s}_l(n)}, \tag{2.113}$$

$$\frac{\partial \hat{y}_t(n)}{\partial \hat{b}_{rp}^{(k)}} = \left(\delta_{lr} u_p(n-k) + \sum_{i_1=1}^{K} u_p(n-k-i_1) \frac{\partial \hat{s}_l(n)}{\partial \hat{s}_r(n-i_1)}\right) \frac{\partial \hat{y}_t(n)}{\hat{s}_l(n)}, \qquad (2.114)$$

where $\partial \hat{y}_t(n)/\hat{s}_l(n)$ is given by (2.65).

2.11 Appendix 2.3. Proof of Theorem 2.1

In the BPTT method, the Wiener model is unfolded back in time and 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 (2.77), and (2.78), from (2.52) it follows that the outputs of sensitivity models for the parameters $\hat{a}_k$, $k = 1, \ldots, na$, can be expressed as functions of the input $u(n)$:

$$\begin{aligned} \frac{\partial \hat{s}(n)}{\partial \hat{a}_k} &= -H_1(q^{-1})\hat{s}(n-k) = -H_1(q^{-1})H_2(q^{-1})u(n-k) \\ &= -H(q^{-1})u(n-k), \end{aligned} \qquad (2.115)$$

where

$$H(q^{-1}) = H_1(q^{-1})H_2(q^{-1}). \qquad (2.116)$$

The impulse response of the system (2.116) is given by the convolution relationship

$$h(n) = \sum_{i_2=0}^{n} h_1(i_2)h_2(n-i_2). \qquad (2.117)$$

Therefore, the partial derivative $\partial \hat{s}(n)/\partial \hat{a}_k$ is related to the input $u(n)$ with the impulse response functions $h_1(n)$ and $h_2(n)$:

$$\frac{\partial \hat{s}(n)}{\partial \hat{a}_k} = -\sum_{i_1=0}^{n-k} h(i_1)u(n-k-i_1) = -\sum_{i_1=0}^{n-k} \sum_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2)u(n-k-i_1). \qquad (2.118)$$

The calculation of (2.118) requires unfolding sensitivity models $n-1$ times back in time. Thus, the partial derivatives $\partial^+ \hat{s}(n)/\partial \hat{a}_k$ calculated for sensitivity models unfolded K times back in time are

$$\frac{\partial^+ \hat{s}(n)}{\partial \hat{a}_k} = \begin{cases} -\displaystyle\sum_{i_1=0}^{n-k} \sum_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2)u(n-k-i_1), & \text{for } n \leqslant K+k \\ -\displaystyle\sum_{i_1=0}^{K} \sum_{i_2=0}^{i_1} h_1(i_2)h_2(i_1-i_2)u(n-k-i_1) & \\ \quad -\displaystyle\sum_{i_1=K+1}^{n-k} \sum_{i_2=0}^{K} h_1(i_2)h_2(i_1-i_2)u(n-k-i_1), & \text{for } n > K+k. \end{cases} \qquad (2.119)$$

From (2.119) it follows that, the errors of computing partial derivatives with the truncated BPTT method are

$$\Delta\hat{s}_{\hat{a}_k}(n) = \begin{cases} 0, & \text{for } n \leqslant K+k \\ -\sum\limits_{i_1=K+1}^{n-k} \sum\limits_{i_2=K+1}^{i_1} h_1(i_2)h_2(i_1-i_2)u(n-k-i_1), & \text{for } n > K+k. \end{cases} \tag{2.120}$$

Since $\{u(n)\}$ is a sequence of zero-mean i.i.d. random variables, $\Delta\hat{s}_{\hat{a}_k}(n)$ is a weighted sum of $n-k-K$ zero-mean i.i.d. random variables. Therefore, the variances of the errors (2.120) and are given by (2.76) for $n > K+k$, and are equal to zero otherwise.

2.12 Appendix 2.4. Proof of Theorem 2.2

In the BPTT method, the Wiener model is unfolded back in time and 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 (2.80), from (2.53) it follows that the outputs of sensitivity models for the parameters $\hat{b}_k$, $k = 1, \ldots, nb$, can be expressed as functions of the input $u(n)$:

$$\frac{\partial\hat{s}(n)}{\partial\hat{b}_k} = H_1(q^{-1})u(n-k). \tag{2.121}$$

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

$$\frac{\partial\hat{s}(n)}{\partial\hat{b}_k} = \sum_{i_1=0}^{n-k} h_1(i_1)u(n-k-i_1). \tag{2.122}$$

The calculation of (2.122) requires unfolding sensitivity models $n-1$ times back in time. Thus, the partial derivatives $\partial^+\hat{s}(n)/\partial\hat{b}_k$ calculated for sensitivity models unfolded K times back in time are

$$\frac{\partial^+\hat{s}(n)}{\partial\hat{b}_k} = \begin{cases} \sum\limits_{i_1=0}^{n-k} h_1(i_1)u(n-k-i_1), & \text{for } n \leqslant K+k \\ \sum\limits_{i_1=0}^{K} h_1(i_1)u(n-k-i_1), & \text{for } n > K+k. \end{cases} \tag{2.123}$$

Therefore, the errors of computing partial derivatives with the truncated BPTT method are

$$\Delta\hat{s}_{\hat{b}_k}(n) = \begin{cases} 0, & \text{for } n \leqslant K+k \\ \displaystyle\sum_{i_1=K+1}^{n-k} h_1(i_1)u(n-k-i_1), & \text{for } n > K+k. \end{cases} \tag{2.124}$$

Since $\{u(n)\}$ is a sequence of zero-mean i.i.d. random variables, $\Delta\hat{s}_{\hat{b}_k}(n)$ is a weighted sum of $n-k-K$ zero-mean i.i.d. random variables. Therefore, the variances of the errors (2.124) are given by (2.79) for $n > K+k$, and are equal to zero otherwise.

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

A. Janczak: Identification of Nonlinear Systems, LNCIS 310, pp. 77–116, 2005.

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

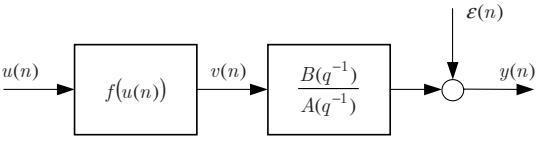

Fig. 3.1. SISO Hammerstein system

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

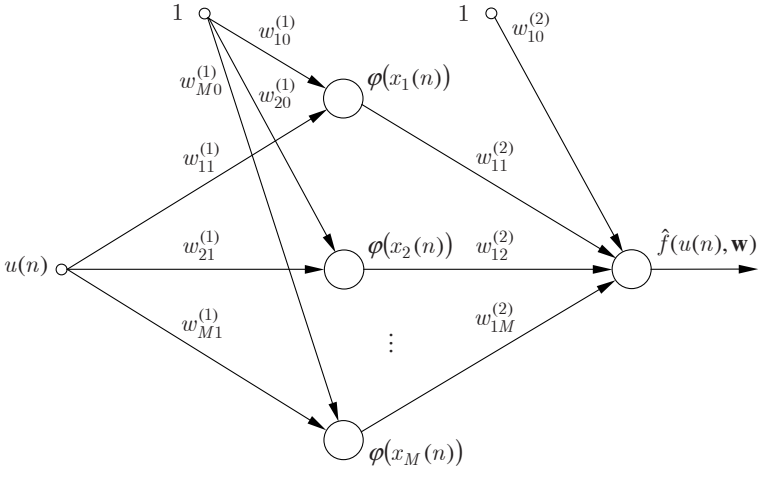

Fig. 3.2. Neural network model of the SISO nonlinear element

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

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

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

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

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

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].

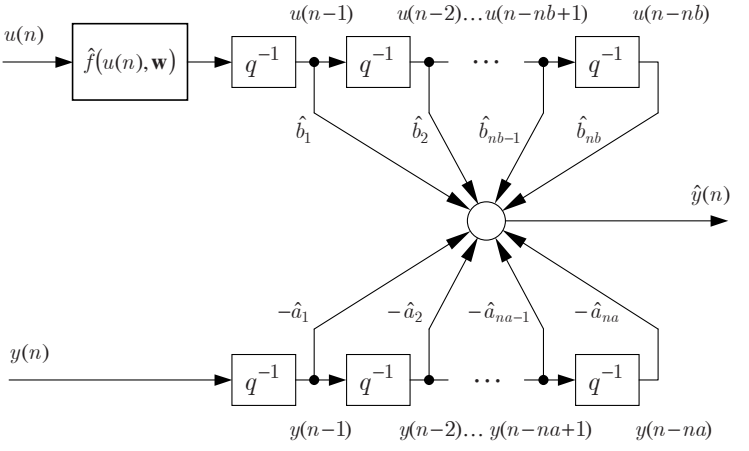

Fig. 3.3. Series-parallel SISO neural network Hammerstein model

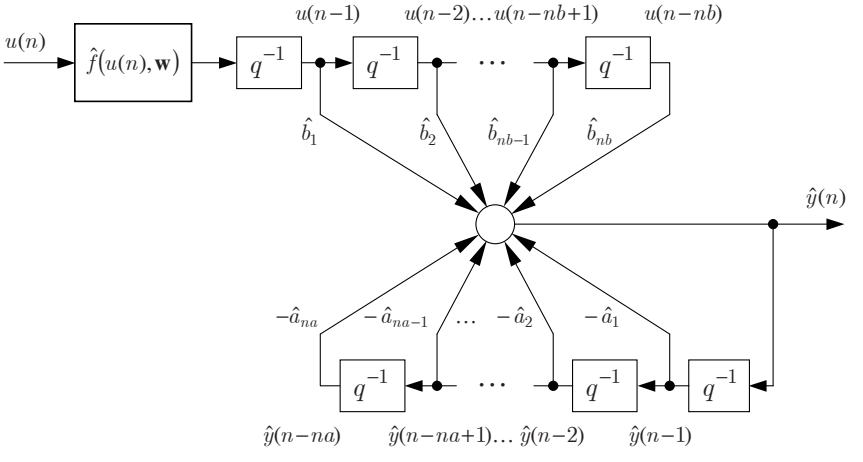

Fig. 3.4. Parallel SISO neural network Hammerstein model

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

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

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

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

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

and the lth vector of weights is defined as

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

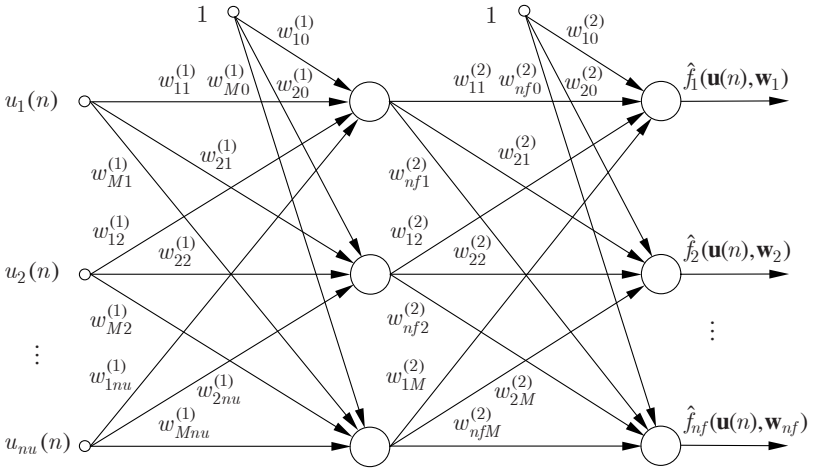

Fig. 3.5. Neural network model of the MIMO nonlinear element

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

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

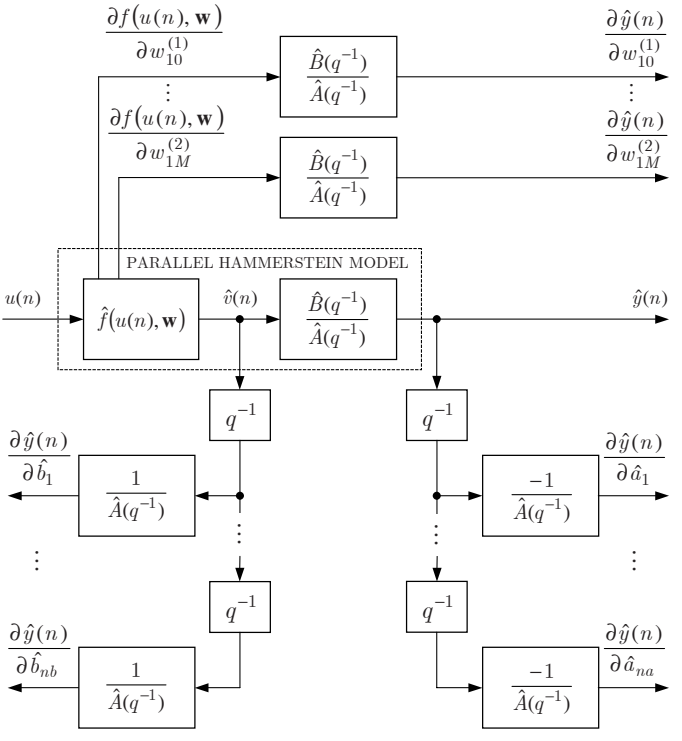

Fig. 3.6. Parallel model of the Hammerstein system and its sensitivity models

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

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

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

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

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

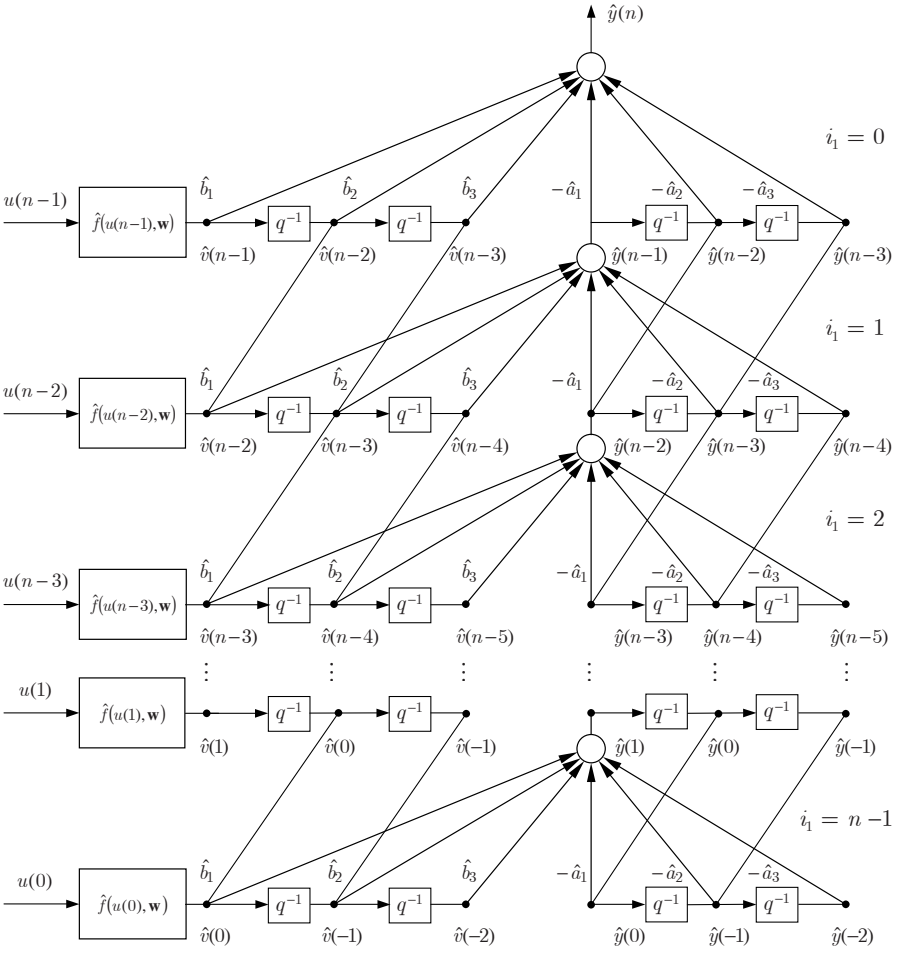

Fig. 3.7. Parallel Hammerstein model of the third order unfolded back in time

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

for $n > K + k$ and 0 otherwise, where $\sigma_f^2 = \mathrm{var}\big[\hat{f}\big(u(n),\mathbf{w}\big)\big]$, 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

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

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

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

for $n > K + k$ and 0 otherwise, where $\sigma^2_{w_c} = \mathrm{var}\big[\partial\hat{f}\big(u(n),\mathbf{w}\big)/\partial w_c\big]$ 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) $\hat{y}(n) = 0$, $n = 0, \ldots, -na + 1$; $\hat{f}\big(u(n),\mathbf{w}\big) = 0$, $n = -1, \ldots, -nb$;
(A3) $\partial\hat{y}(n)/\partial w_{10}^{(2)} = 0$, $n = 0, \ldots, -na + 1$; $\partial\hat{f}\big(u(n),\mathbf{w}\big)/\partial w_{10}^{(2)} = 0$, $n = -1, \ldots, -nb$;

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:

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

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

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

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

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

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,

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

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

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}

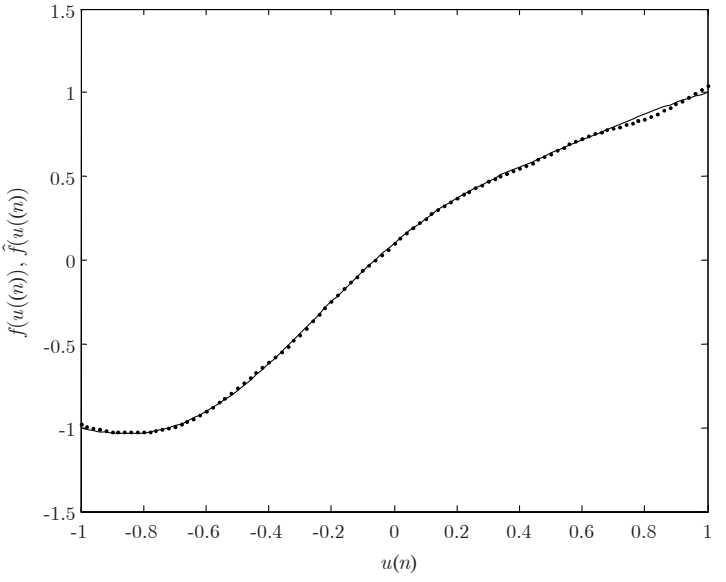

Fig. 3.8. BPTT results, $K = 1$. True (solid line) and estimated (dotted line) nonlinear functions

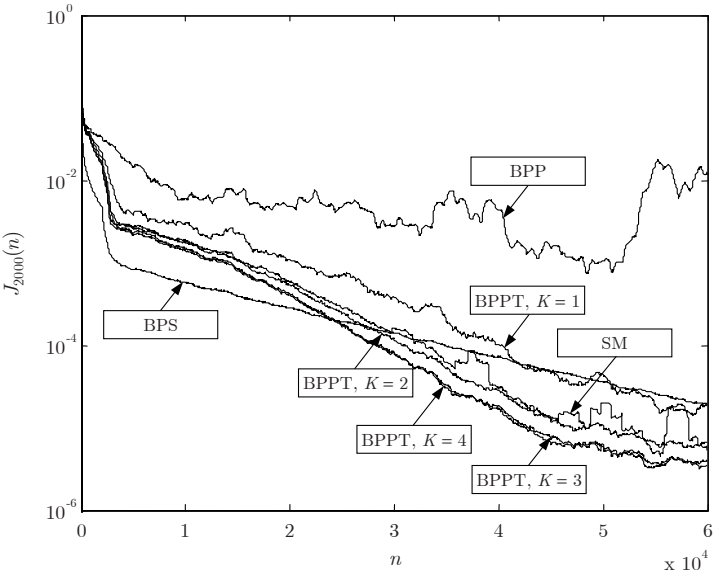

Fig. 3.9. Comparison of convergence rates of different learning algorithms

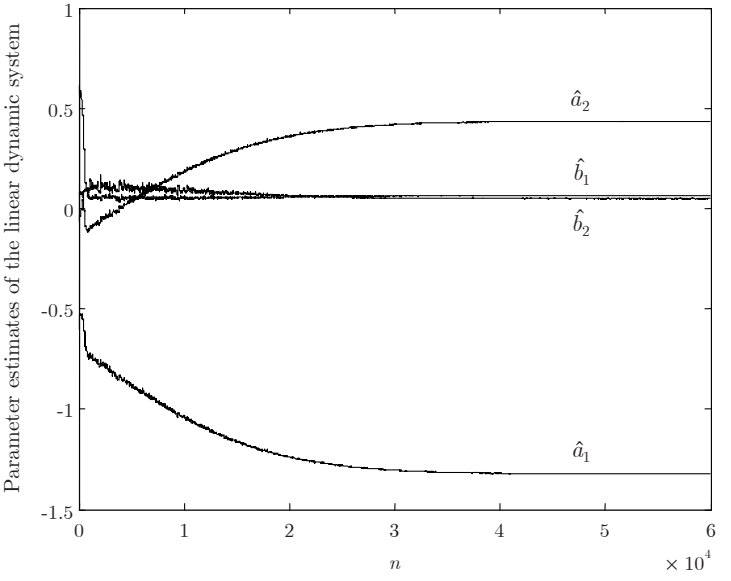

Fig. 3.10. SM results. Evolution of the parameters of the linear dynamic model

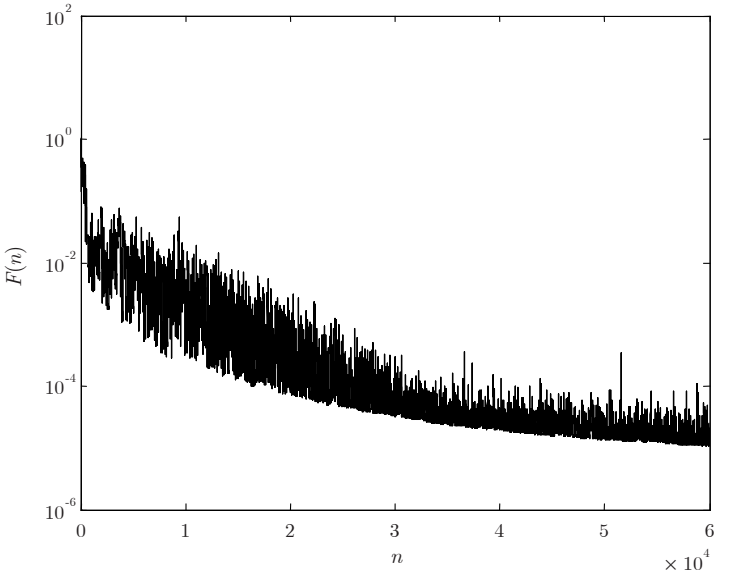

Fig. 3.11. BPTT results, $K = 4$. Convergence of the nonlinear element model

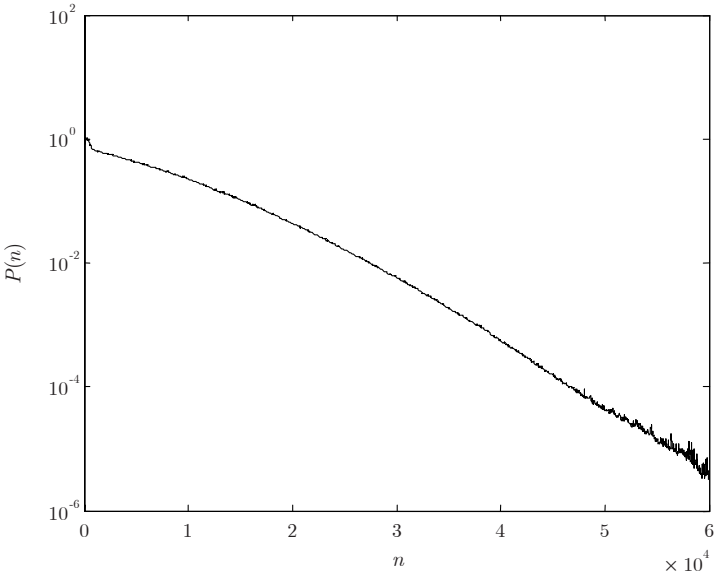

Fig. 3.12. BPTT results, $K = 4$. Convergence of the linear dynamic model

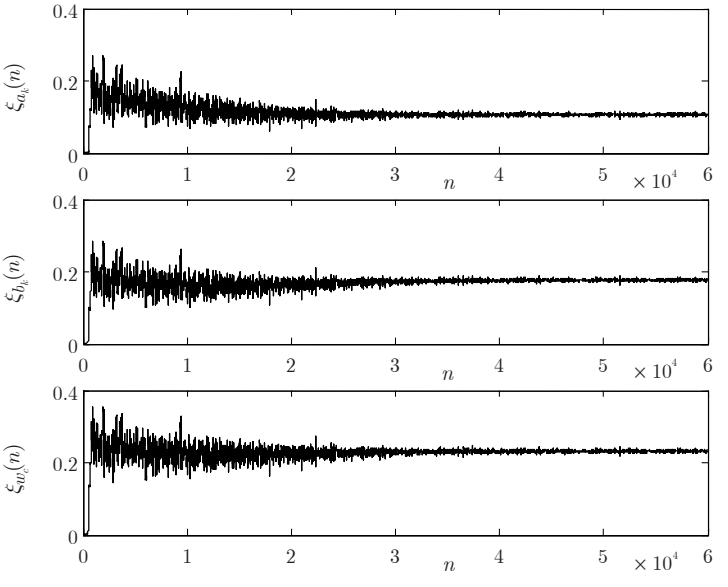

Fig. 3.13. Evolution of gradient calculation accuracy degrees, (BPTT, $K = 4$)

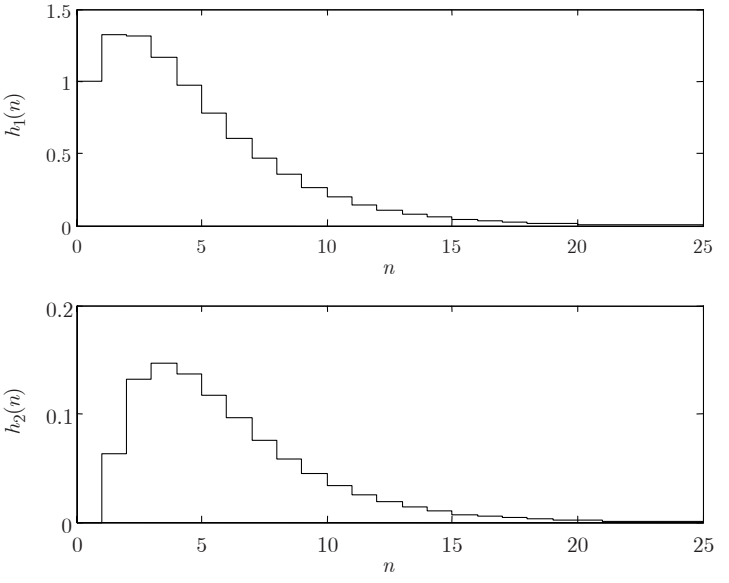

Fig. 3.14. Impulse responses of sensitivity models and the linear dynamic model (BPTT, $K = 4$)

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

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

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 $\hat{y}_t(n)$, $t=1,\dots,ny$, w.r.t. the past model outputs $\hat{y}_r(n-i_1)$, $r=1,\dots,ny$, $i_1=1,\dots,K$,

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

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 $\partial^+\hat{y}_t(n)/\partial\hat{a}_{rp}^{(k)}$, $p=1,\dots,ny$, and $\partial^+\hat{y}_t(n)/\partial\hat{b}_{rp}^{(k)}$, $p=1,\dots,nu$, 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 $i = 1,\ldots,nu$, $j = 1,\ldots,M$, $l = 1,\ldots,nf$.

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 $\hat{a}_k$, $k = 1,\ldots,na$, 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

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

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

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

and the variances of the errors (3.117) are

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

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

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

and the variances of the errors (3.125) are

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

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

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

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

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

Therefore,

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

4

Polynomial Wiener models

This chapter deals with polynomial Wiener models, i.e., models composed of a pulse transfer model of the linear dynamic system and a polynomial model of the nonlinear element or the inverse nonlinear element. A modified definition of the equation error and a modified series-parallel Wiener model are introduced. Assuming that the nonlinear element is invertible and the inverse nonlinear element can be described by a polynomial, the modified series-parallel Wiener model can be transformed into the linear-in-parameters form and its parameters can be calculated with the least squares method. Such an approach, however, results in inconsistent parameter estimates. As a remedy against this problem, an instrumental variables method is proposed with instrumental variables chosen as delayed system inputs and delayed and powered delayed outputs of the model obtained using the least squares method.

An alternative to this combined least squares-instrumental variables approach is the prediction method, in which the parameters of the noninverted nonlinear element are estimated, see [128] for the batch version and [85] for the sequential one. The pseudolinear regression method [86], being a simplified version of the prediction error method of lower computational requirements, is another effective technique for parameter estimation in Wiener systems disturbed additively by a discrete-time white noise.

This chapter is organized as follows: First, the least squares approach to the identification of Wiener systems based on the modified series-parallel model is introduced in Section 4.1. Two different cases of a Wiener system with and without the linear term are considered. Section 4.2 contains details of the recursive prediction error approach to the identification of polynomial Wiener systems. The pseudolinear regression method is discussed in Section 4.3. Finally, a brief summary is given in Section 4.4.

A. Janczak: Identification of Nonlinear Systems, LNCIS 310, pp. 117–141, 2005.

4.1 Least squares approach to the identification of Wiener systems

This section presents a least squares approach to the identification of polynomial Wiener systems. To transform the Wiener model into the linear-in-parameters form, the noninverted model of the linear dynamic system and the inverse model of the nonlinear element are used. The following assumptions are made about the identified Wiener system:

Assumption 4.1. The SISO Wiener system is

$$y(n) = f\left(\frac{B(q^{-1})}{A(q^{-1})}u(n) + \varepsilon(n)\right), \tag{4.1}$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}, \tag{4.2}$$

$$B(q^{-1}) = b_1 q^{-1} + \ldots + b_{nb} q^{-nb}, \tag{4.3}$$

and $\varepsilon(n)$ is the additive disturbance.

Assumption 4.2. The polynomials $A(q^{-1})$ and $B(q^{-1})$ are coprime.

Assumption 4.3. The orders na and nb of the polynomials $A(q^{-1})$ and $B(q^{-1})$ are known.

Assumption 4.4. The linear dynamic system is casual and asymptotically stable.

Assumption 4.5. The input $u(n)$ has finite moments and is independent of $\varepsilon(k)$ for all n and k.

Assumption 4.6. The nonlinear function $f(\cdot)$ is defined on the interval $[a, b]$.

Assumption 4.7. The nonlinear function $f(\cdot)$ is invertible.

Assumption 4.8. The inverse nonlinear function $f^{-1}\big(y(n)\big)$ can be expressed by the polynomial

$$f^{-1}\big(y(n)\big) = \gamma_0 + \gamma_1 y(n) + \gamma_2 y(n)^2 + \cdots + \gamma_r y(n)^r \tag{4.4}$$

of a known order r.

The identification problem can be formulated as follows: Given the sequence of the Wiener system input and output measurements $\{u(n), y(n)\}$, $i = 1, \ldots, N$, estimate the parameters of the linear dynamic system and the inverse nonlinear element minimizing the following criterion:

$$J(n) = \frac{1}{2}\sum_{j=1}^{N}\big(y(n) - \hat{y}(n)\big)^2, \tag{4.5}$$

where $\hat{y}(n)$ is the output of the Wiener model.

4.1.1 Identification error

For a polynomial Wiener model, both its parallel and series-parallel forms are nonlinear functions of model parameters. Moreover, the series-parallel model contains not only a model of the nonlinear element but also its inverse [73] – Fig. 4.2. Consider the parallel model of the Wiener model given by

$$\hat{y}(n) = \hat{f}\left(\frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}u(n)\right), \tag{4.6}$$

with

$$\hat{A}(q^{-1}) = 1 + \hat{a}_1 q^{-1} + \cdots + \hat{a}_{na} q^{-na}, \tag{4.7}$$

$$\hat{B}(q^{-1}) = \hat{b}_1 q^{-1} + \cdots + \hat{b}_{nb} q^{-nb}. \tag{4.8}$$

If $\hat{f}(\cdot)$ is invertible, (4.6) can be written as

$$\hat{f}^{-1}\big(\hat{y}(n)\big) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}u(n). \tag{4.9}$$

Assumption 4.9. The inverse nonlinear function $\hat{f}^{-1}(\cdot)$ has the form of a polynomial of the order r:

$$\hat{f}^{-1}\big(\hat{y}(n)\big) = \hat{\gamma}_0 + \hat{\gamma}_1\hat{y}(n) + \hat{\gamma}_2\hat{y}^2(n) + \cdots + \hat{\gamma}_r\hat{y}^r(n). \tag{4.10}$$

Assume also that the polynomial (4.10) contains the linear term, i.e., $\hat{\gamma}_1 \neq 0$. Then combining (4.10) and (4.9), the output of the model can be expressed as [80, 83]:

$$\hat{y}(n) = \frac{1}{\hat{\gamma}_1}\left(\frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}u(n) - \Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right), \tag{4.11}$$

where

$$\Delta\hat{f}^{-1}\big(\hat{y}(n)\big) = \hat{\gamma}_0 + \hat{\gamma}_2\hat{y}^2(n) + \hat{\gamma}_3\hat{y}^3(n) + \cdots + \hat{\gamma}_r\hat{y}^r(n). \tag{4.12}$$

The model (4.11) can be written as

$$\hat{y}(n) = \big(1-\hat{A}(q^{-1})\big)\hat{y}(n) + \frac{1}{\hat{\gamma}_1}\left(\hat{B}(q^{-1})u(n) - \hat{A}(q^{-1})\Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right). \tag{4.13}$$

Replacing $\hat{y}(n)$ by $y(n)$ on the r.h.s. of (4.13), the following modified series-parallel model can be obtained:

$$\hat{y}(n) = \big(1-\hat{A}(q^{-1})\big)y(n) + \frac{1}{\hat{\gamma}_1}\left[\hat{B}(q^{-1})u(n) - \hat{A}(q^{-1})\Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right]. \tag{4.14}$$

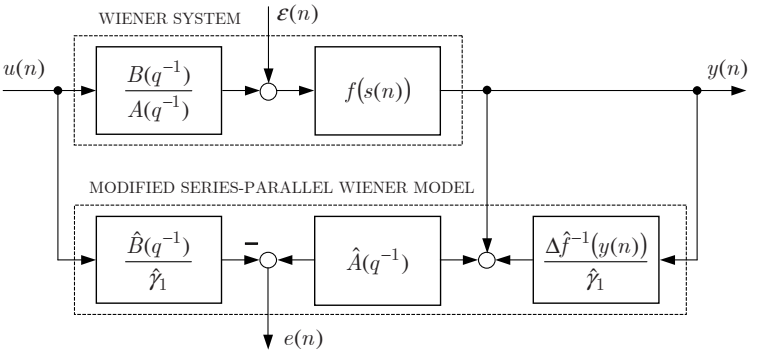

Fig. 4.1. Modified series-parallel Wiener model. The identification error definition for systems with the linear term

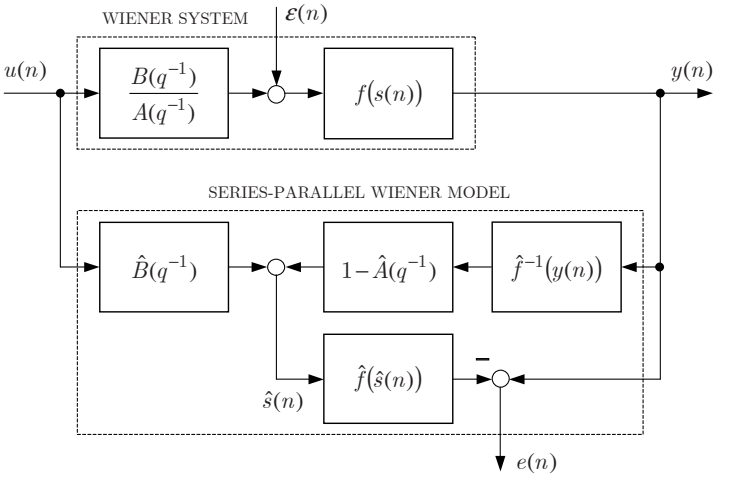

Fig. 4.2. Series-parallel Wiener model. The definition of the identification error

The modified series-parallel model, shown in Fig. 4.1, is different from the series-parallel model, which contains both the model of the nonlinear element and its inverse,

$$\hat{y}(n) = \hat{f}\Big[\big(1 - \hat{A}(q^{-1})\big)\hat{f}^{-1}\big(y(n)\big) + \hat{B}(q^{-1})u(n)\Big], \tag{4.15}$$

and the inverse series-parallel model

$$\hat{u}(n-1) = \frac{1}{\hat{b}_1}\left[\big(\hat{b}_1 - \hat{B}(q^{-1})\big)u(n) + \hat{A}(q^{-1})\hat{f}^{-1}\big(\hat{y}(n)\big)\right], \tag{4.16}$$

see Figs 4.2 – 4.3 for comparison. Applying (4.14), the following modified definition of the identification error can be introduced:

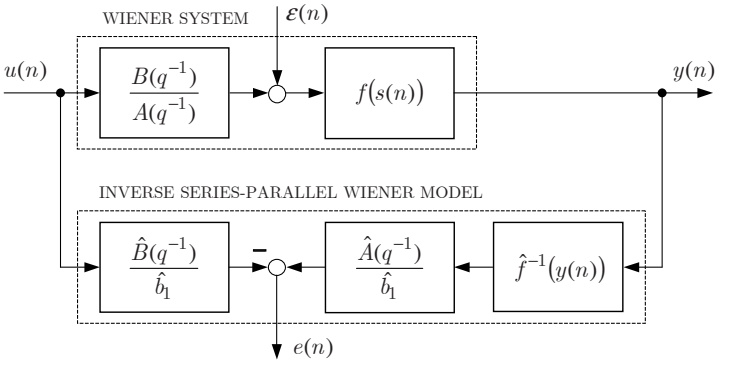

Fig. 4.3. Inverse series-parallel Wiener model. The definition of the identification error

$$e(n) = y(n) - \hat{y}(n) \tag{4.17}$$

$$e(n) = \hat{A}(q^{-1})y(n) - \frac{1}{\hat{\gamma}_1}\left[\hat{B}(q^{-1})u(n) - \hat{A}(q^{-1})\Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right]. \tag{4.18}$$

4.1.2 Nonlinear characteristic with the linear term

Assuming that the identified Wiener system has an invertible nonlinear characteristic with $\gamma_1 \neq 0$, we will express the modified series-parallel Wiener in the linear-in-parameters form.
Introduce the parameter vector $\hat{\boldsymbol{\theta}}$

$$\hat{\boldsymbol{\theta}} = \left[\hat{a}_1 \dots \hat{a}_{na}\ \hat{\beta}_1 \dots \hat{\beta}_{nb}\ \hat{\alpha}_{00}\ \hat{\alpha}_{20} \dots \hat{\alpha}_{rna}\right]^T, \tag{4.19}$$

and the regression vector $\mathbf{x}(n)$

$$\begin{aligned}\mathbf{x}(n) = \big[&- y(n-1) \dots - y(n-na)\ u(n-1) \dots u(n-nb) \\ &1 \quad - y^2(n) \dots - y^r(n-na)\big]^T,\end{aligned} \tag{4.20}$$

where

$$\hat{\beta}_k = \frac{\hat{b}_k}{\hat{\gamma}_1},\ k = 1, \dots, nb, \tag{4.21}$$

$$\hat{\alpha}_{jk} = \begin{cases} \left(1 + \sum_{m=1}^{na} \hat{a}_m\right)\dfrac{\hat{\gamma}_j}{\hat{\gamma}_1}, & k = 0,\ j = 0, \\ \dfrac{\hat{\gamma}_j}{\hat{\gamma}_1}, & k = 0,\ j = 2, 3, \dots, r, \\ \hat{a}_k\dfrac{\hat{\gamma}_j}{\hat{\gamma}_1}, & k = 1, \dots, na,\ j = 2, 3, \dots, r. \end{cases} \tag{4.22}$$

Then the model (4.14) can be written as

$$\hat{y}(n) = \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}. \tag{4.23}$$

Minimizing (4.5), the parameter vector $\hat{\boldsymbol{\theta}}$ can be calculated with the least squares (LS) or recursive least squares (RLS) method. Note that the number of parameters in (4.14) is $na+nb+r(na+1)$, while the number of parameters of $\hat{A}(q^{-1})$, $\hat{B}(q^{-1})$, and $\hat{f}(\cdot)$ is $na+nb+r+1$. Therefore, to obtain a unique solution, methods similar to these proposed in [34] for the identification of Hammerstein models can be employed.

4.1.3 Nonlinear characteristic without the linear term

Consider a Wiener system that fulfills the following assumptions:

- The linear term $\gamma_1 = 0$.
- The second order term $\gamma_2 \neq 0$.

In this case, the following modified series-parallel model can be defined (Fig. 4.4):

$$\hat{y}^2(n) = \big(1-\hat{A}(q^{-1})\big)y^2(n) + \frac{1}{\hat{\gamma}_2}\left[\hat{B}(q^{-1})u(n) - \hat{A}(q^{-1})\Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right]. \tag{4.24}$$

Now, the identification error can be defined as

$$\begin{aligned} e(n) &= y^2(n) - \hat{y}^2(n) \\ &= \hat{A}(q^{-1})y^2(n) - \frac{1}{\hat{\gamma}_2}\left[\hat{B}(q^{-1})u(n) - \hat{A}(q^{-1})\Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right]. \end{aligned} \tag{4.25}$$

Hence, (4.24) can be written in the following linear-in-parameters form:

$$\hat{y}^2(n) = \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}, \tag{4.26}$$

with the parameter vector $\hat{\boldsymbol{\theta}}$,

$$\hat{\boldsymbol{\theta}} = \left[\hat{a}_1 \ldots \hat{a}_{na}\ \hat{\beta}_1 \ldots \hat{\beta}_{nb}\ \hat{\alpha}_{00}\ \hat{\alpha}_{30} \ldots \hat{\alpha}_{rna}\right]^T, \tag{4.27}$$

and the regression vector $\mathbf{x}(n)$,

$$\begin{aligned} \mathbf{x}(n) = \big[&-y^2(n-1) \ldots -y^2(n-na)\ u(n-1) \ldots u(n-nb) \\ &1\ -y^3(n) \ldots -y^r(n-na)\big]^T, \end{aligned} \tag{4.28}$$

where

$$\hat{\beta}_k = \frac{\hat{b}_k}{\hat{\gamma}_2},\ k = 1, \ldots, nb, \tag{4.29}$$

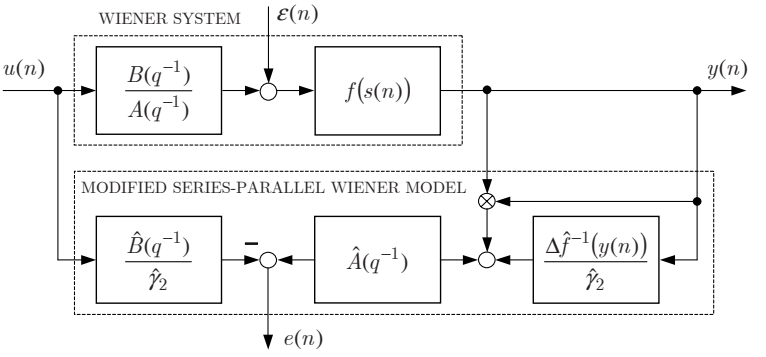

Fig. 4.4. Modified series-parallel model. The identification error definition for systems without the linear term

$$\hat{\alpha}_{jk} = \begin{cases} \left(1 + \sum_{m=1}^{na} \hat{a}_m\right)\frac{\hat{\gamma}_j}{\hat{\gamma}_2}, & k = 0,\ j = 0, \\ \frac{\hat{\gamma}_j}{\hat{\gamma}_2}, & k = 0,\ j = 3, 4, \ldots, r, \\ \hat{a}_k\frac{\hat{\gamma}_j}{\hat{\gamma}_2}, & k = 1, \ldots, na, j = 3, 4, \ldots, r. \end{cases} \tag{4.30}$$

As in the previous case, the parameter vector $\hat{\boldsymbol{\theta}}$ can be calculated with the least squares (LS) or recursive least squares (RLS) method minimizing the following criterion:

$$J = \frac{1}{2}\sum_{j=1}^{N}\left(y^2(n) - \hat{y}^2(n)\right)^2. \tag{4.31}$$

4.1.4 Asymptotic bias error of the LS estimator

Consider a polynomial Wiener system (4.1) – (4.4) that contains the linear term, i.e., $\gamma_1 \neq 0$, and the modified series-parallel Wiener model (4.23).We will show now that parameter estimates of the Wiener system obtained with the LS method are nonconsistent, i.e. asymptotically biased, even if the additive disturbance $\varepsilon(n)$ is

$$\varepsilon(n) = \frac{\epsilon(n)}{A(q^{-1})}, \tag{4.32}$$

where $\epsilon(n)$ is the discrete time white noise.

Theorem 4.1. Let $\hat{\boldsymbol{\theta}}$ denote the vector of parameter estimates, defined by (4.19), and $\boldsymbol{\theta}$ – the corresponding true parameter vector of the Wiener system, defined by (4.1) – (4.4).

Then the LS estimate of $\boldsymbol{\theta}$ is asymptotically biased, i.e., $\hat{\boldsymbol{\theta}}$ does not converge (with the probability 1) to the true parameter vector $\boldsymbol{\theta}$.

Proof: The output $y(n)$ of the Wiener system, defined by (4.1) and (4.32), is

$$y(n)=\big(1-A(q^{-1})\big)y(n)+\frac{1}{\gamma_1}\Big[B(q^{-1})u(n)-A(q^{-1})\Delta f^{-1}\big(\hat{y}(n)\big)+\epsilon(n)\Big]. \quad (4.33)$$

Introducing the true parameter vector $\boldsymbol{\theta}$,

$$\boldsymbol{\theta} = \big[a_1 \dots a_{na}\ \beta_1 \dots \beta_{nb}\ \alpha_{00}\ \alpha_{20} \dots \alpha_{rna}\big]^T, \quad (4.34)$$

where

$$\beta_k = \frac{b_k}{\gamma_1},\ k = 1, \dots, nb, \quad (4.35)$$

$$\alpha_{jk} = \begin{cases} \left(1 + \sum_{m=1}^{na} a_m\right)\frac{\gamma_j}{\gamma_1}, & k = 0,\ j = 0, \\ \frac{\gamma_j}{\gamma_1}, & k = 0,\ j = 2, 3, \dots, r, \\ a_k\frac{\gamma_j}{\gamma_1}, & k = 1, \dots, na,\ j = 2, 3, \dots, r, \end{cases} \quad (4.36)$$

the system output can be expressed as

$$y(n) = \mathbf{x}^T(n)\boldsymbol{\theta} + \frac{1}{\gamma_1}\,\epsilon(n). \quad (4.37)$$

The solution to the LS estimation problem is given by

$$\hat{\boldsymbol{\theta}} = \left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)\mathbf{x}^T(n)\right]^{-1}\left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)y(n)\right]. \quad (4.38)$$

From (4.37) and (4.38), it follows that the parameter estimation error $\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}$ is

$$\begin{aligned}\hat{\boldsymbol{\theta}}-\boldsymbol{\theta} &= \left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)\mathbf{x}^T(n)\right]^{-1}\left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)y(n)-\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)\mathbf{x}^T(n)\right)\boldsymbol{\theta}\right] \\ &= \frac{1}{\gamma_1}\left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)\mathbf{x}^T(n)\right]^{-1}\left[\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)\epsilon(n)\right].\end{aligned} \quad (4.39)$$

Therefore, if $N \to \infty$,

$$\hat{\boldsymbol{\theta}}-\boldsymbol{\theta} \to \frac{1}{\gamma_1}\Big[\mathrm{E}\big(\mathbf{x}(n)\mathbf{x}^T(n)\big)\Big]^{-1}\Big[\mathrm{E}\big(\mathbf{x}(n)\epsilon(n)\big)\Big] \neq \mathbf{0}, \quad (4.40)$$

as $\mathrm{E}\big[y^2(n)\epsilon(n)\big] \neq 0, \dots, \mathrm{E}\big[y^r(n)\epsilon(n)\big] \neq 0$, and thus $\mathrm{E}\big[\mathbf{x}(n)\epsilon(n)\big] \neq \mathbf{0}$.

Remark 4.1. In a similar way, it can also be shown that the parameter vector $\hat{\boldsymbol{\theta}}$ of the modified series-parallel Wiener model (4.24), calculated with the LS method, is asymptotically biased.

Remark 4.2. It can also be proved that asymptotically biased LS parameter estimates are obtained using other linear-in-parameter models which contain the inverse polynomial model of the nonlinear element. Examples of such models are the frequency sampling model [95, 96], the inverse Wiener model, and the model based on Laguerre filters [116].

4.1.5 Instrumental variables method

To obtain consistent parameter estimates, the regression vector $\mathbf{x}(n)$ should be uncorrelated with system disturbances. That is not the case if we use the modified series-parallel model, as the powered system outputs $y^2(n), \ldots, y^r(n)$ depend on $\epsilon(n)$. The instrumental variables method is a well-known remedy against such a situation. Applying the instrumental variables method, parameter estimation can be performed according to the following scheme:

1. Estimate the parameters of the system using the LS or the RLS method.
2. Simulate the model to determine the instrumental variables $\mathbf{z}(n)$.
3. Estimate the parameters of the system using the IV or the RIV method with the instrumental variables $\mathbf{z}(n)$.

The choice of instrumental variables is a vital design problem in any instrumental variables approach, see [149] for more details. Clearly, the best choice would be undisturbed powered system outputs, but these are not available for measurement. Instead, we can employ powered outputs of the model, or powered outputs of the linear dynamic model, calculated with the LS method, and define the instrumental variables as

$$\begin{aligned}\mathbf{z}(n) = \big[&- \hat{y}(n-1) \ldots - \hat{y}(n-na)\ u(n-1) \ldots u(n-nb) \\ &1\ - \hat{y}^2(n) \ldots - \hat{y}^r(n-na)\big]^T\end{aligned} \tag{4.41}$$

in the case of Wiener systems with the linear term, or

$$\begin{aligned}\mathbf{z}(n) = \big[&- \hat{y}^2(n-1) \ldots - \hat{y}^2(n-na)\ u(n-1) \ldots u(n-nb) \\ &1\ - \hat{y}^3(n) \ldots - \hat{y}^r(n-na)\big]^T\end{aligned} \tag{4.42}$$

in the case of Wiener systems without the linear term. The instrumental variables $\mathbf{z}(n)$ are uncorrelated with system disturbances:

$$E[\mathbf{z}(n)\varepsilon(n)] = \mathbf{0}. \tag{4.43}$$

4.1.6 Simulation example. Nonlinear characteristic with the linear term

A linear dynamic system given by the continuous-time polynomial pulse transfer function

$$G(s) = \frac{1}{6s^2 + 5s + 1} \tag{4.44}$$

was converted to discrete time, assuming a zero order hold on the input and the sampling interval 1s, leading to the following difference equation:

$$s(n) = 1.3231s(n–1) - 0.4346s(n-2) + 0.0635u(n–1) + 0.0481u(n-2). \tag{4.45}$$

The linear dynamic system was followed by a nonlinear element described by the function

$$f\big(s(n)\big) = 4\big(\sqrt[3]{0.75s(n)} - 1\big). \tag{4.46}$$

Therefore, the inverse nonlinear function $f^{-1}\big(y(n)\big)$ is a polynomial:

$$f^{-1}\big(y(n)\big) = \frac{4}{3} + y(n) + 0.25y^2(n) + \frac{1}{48}y^3(n). \tag{4.47}$$

The input sequence $\{u(n)\}$ consisted of 40000 pseudo-random numbers uniformly distributed in $(-5, 5)$. Parameter estimation was performed with both the LS method and the IV method, assuming that $r = 3$ and $\hat{\gamma}_1 = 1$. Additive system disturbances were given by $\varepsilon(n) = [1/A(q^{-1})]\,\epsilon(n)$ with $\{\epsilon(n)\}$ – a normally distributed pseudo-random sequence $\mathcal{N}(0, 0.1)$. This corresponds with the signal to noise ratio $SNR = \sqrt{\mathrm{var}(y(n) - \varepsilon(n))/\mathrm{var}(\varepsilon(n))} = 3.14$. The identification results, given in Tables 4.1 and 4.2 and illustrated in Figs 4.5 and 4.6, show a considerable improvement in IV parameter estimates in comparison with LS ones.

Table 4.1. Parameter estimates, $SNR = 3.14$

Parameter	True	LS	IV
	$\sigma_\varepsilon = 0$	$\sigma_\varepsilon = 0.1$	$\sigma_\varepsilon = 0.1$
a_1	−1.3231	−1.2803	−1.3292
a_2	0.4346	0.4158	0.4370
b_1	0.0635	0.0558	0.0636
b_2	0.0481	0.0423	0.0482
γ_0	1.3333	1.0764	1.3813
γ_1	1.0000	1.0000	1.0000
γ_2	0.2500	0.2473	0.2503
γ_3	0.0208	0.0199	0.0209

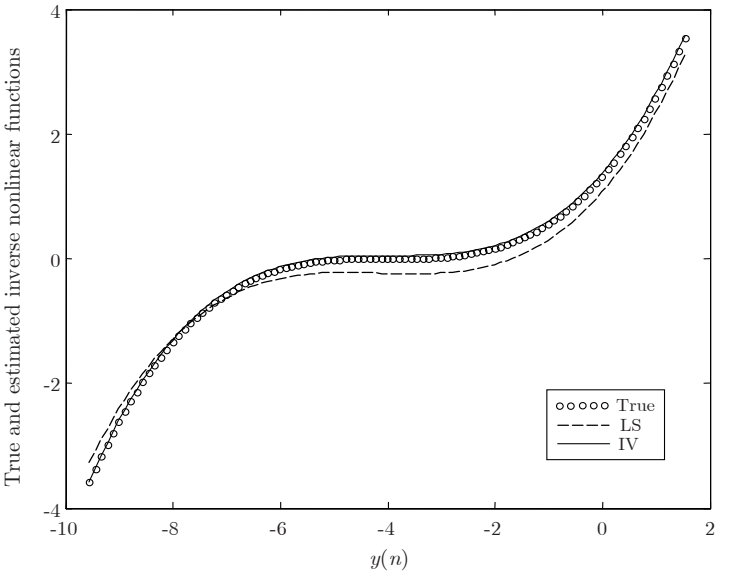

Fig. 4.5. Wiener system with the linear term. True $f^{-1}\big(y(n)\big)$ and estimated $\hat{f}^{-1}\big(y(n)\big)$ inverse nonlinear functions

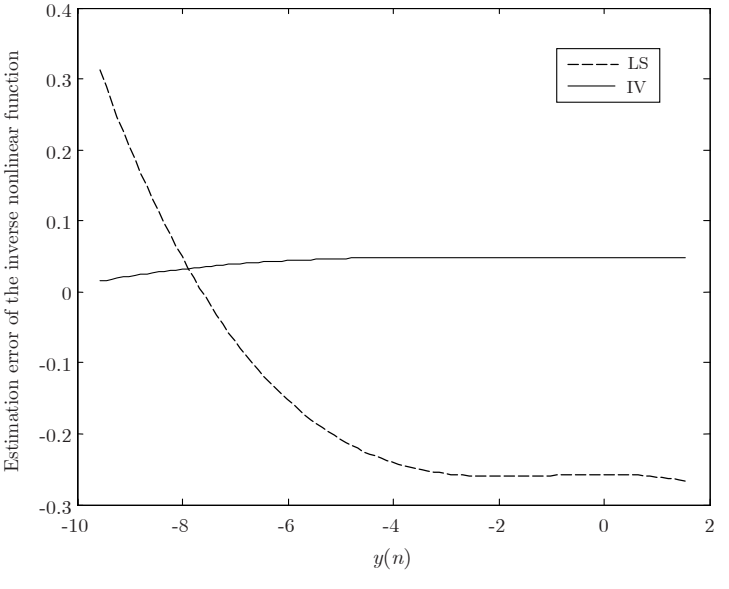

Fig. 4.6. Wiener system with the linear term. Estimation error $\hat{f}^{-1}\big(y(n)\big) - f^{-1}\big(y(n)\big)$.

Table 4.2. Comparison of estimation accuracy

Performance index	LS ($\sigma_\varepsilon = 0$)	LS	IV
$\frac{1}{4}\sum_{j=1}^{2}\left[(\hat{a}_j - a_j)^2 + (\hat{b}_j - b_j)^2\right]$	4.62×10^{-23}	5.70×10^{-4}	1.08×10^{-5}
$\frac{1}{3}\left[(\hat{\gamma}_0 - \gamma_0)^2 + \sum_{j=2}^{3}(\hat{\gamma}_j - \gamma_j)^2\right]$	1.13×10^{-21}	2.20×10^{-2}	7.66×10^{-4}
$\frac{1}{50}\sum_{i=1}^{50}\left[\hat{f}^{-1}\big(y(n)\big) - f^{-1}\big(y(n)\big)\right]^2$	3.78×10^{-21}	4.74×10^{-2}	1.91×10^{-3}

4.1.7 Simulation example. Nonlinear characteristic without the linear term

The linear dynamic system (4.45) and a nonlinear element defined by the function

$$f\big(s(n)\big) = \sqrt{\sqrt{s(n)} + 0.5}, \quad s(n) \geqslant 0 \tag{4.48}$$

were used in the example of a Wiener system without the linear term and a nonzero second order term. The inverse nonlinear function is a polynomial:

$$f^{-1}\big(y(n)\big) = 0.25 - y^2(n) + y^4(n), \quad y(n) \geqslant \sqrt{0.5}. \tag{4.49}$$

The input sequence $\{u(n)\}$ contained 50000 pseudo-random numbers uniformly distributed in $(1.5, 6)$. Additive system disturbances were $\varepsilon(n) = [1/A(q^{-1})]\,\epsilon(n)$ with $\{\epsilon(n)\}$ – a normally distributed pseudo-random sequence $\mathcal{N}(0, 0.025)$. As in the previous example, parameter estimation was performed using the LS method and the IV method and assuming: $r = 4$, $\hat{\gamma}_1 = \hat{\gamma}_3 = 0$, $\hat{\gamma}_2 = 1$. The identification results, given in Tables 4.3 and 4.4 and illustrated in Figs 4.7 and 4.8, confirm practical feasibility of the proposed approach.

Table 4.3. Parameter estimates, $SNR = 19.37$

Parameter	True	LS	IV
	$\sigma_\varepsilon = 0$	$\sigma_\varepsilon = 0.025$	$\sigma_\varepsilon = 0.025$
a_1	-1.3231	-1.4448	-1.2898
a_2	0.4346	0.6233	0.4107
b_1	0.0635	0.0014	0.0635
b_2	0.0481	0.0011	0.0481
γ_0	0.2500	1.2119	0.2187
γ_2	1.0000	1.0000	1.0000
γ_4	1.0000	0.2212	1.0005

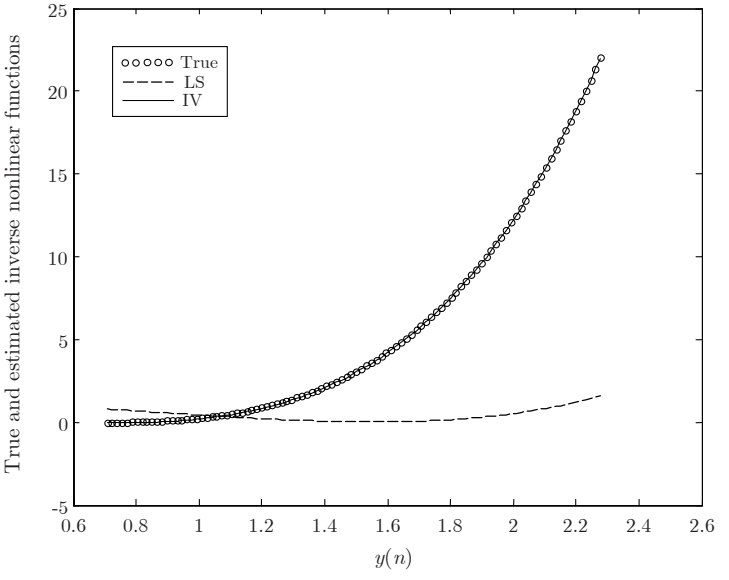

Fig. 4.7. Wiener system without the linear term. True $f^{-1}\big(y(n)\big)$ and estimated $\hat{f}^{-1}\big(y(n)\big)$ inverse nonlinear functions

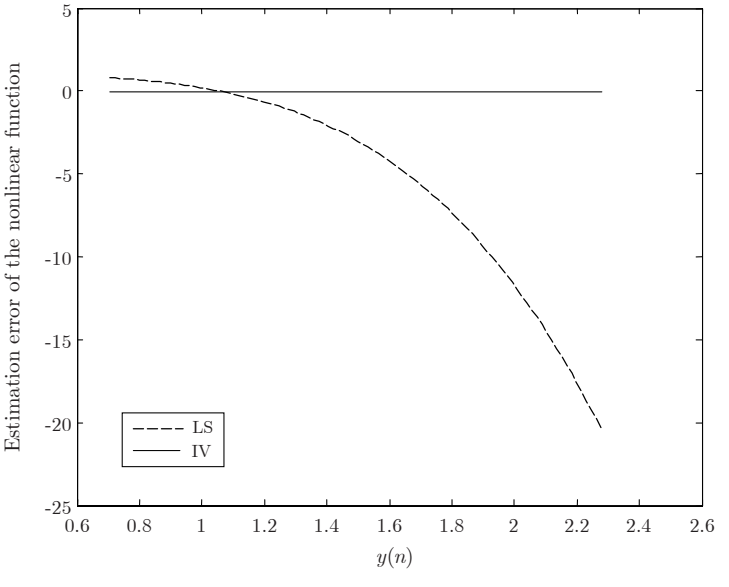

Fig. 4.8. Wiener system without the linear term. Estimation error $\hat{f}^{-1}\big(y(n)\big) - f^{-1}\big(y(n)\big)$

Table 4.4. Comparison of estimation accuracy

Performance index	LS ($\sigma_\varepsilon = 0$)	LS	IV
$\frac{1}{4}\sum_{j=1}^{2}\left[(\hat{a}_j - a_j)^2 + (\hat{b}_j - b_j)^2\right]$	3.85×10^{-16}	1.41×10^{-2}	4.20×10^{-4}
$\frac{1}{2}\left[(\hat{\gamma}_0 - \gamma_0)^2 + (\hat{\gamma}_4 - \gamma_4)^2\right]$	1.24×10^{-13}	7.66×10^{-1}	4.92×10^{-4}
$\frac{1}{50}\sum_{i=1}^{50}\left[\hat{f}^{-1}\big(y(n)\big) - f^{-1}\big(y(n)\big)\right]^2$	5.80×10^{-12}	6.16×10^{1}	7.58×10^{-4}

Although only one technique for instrumental variables generation is discussed and illustrated here, other known techniques can be considered as well. Contrary to the identification of the inverse Wiener model, an attractive feature of this approach is that the linear sub-system is not required to be minimum phase.

4.2 Identification of Wiener systems with the prediction error method

In the linear regression approach, described in Section 4.1, the class of identified systems is restricted by the assumption of invertibility of the nonlinear characteristic. This assumption is not necessary in the recursive prediction error method of Wigren [165], in which the nonlinear characteristic is approximated with a piecewise linear function.
This section presents an identification algorithm for Wiener systems which uses the recursive prediction error (RPE) approach with a polynomial model of the nonlinear element and a pulse transfer function model of the linear dynamic system – Fig. 4.9.

4.2.1 Polynomial Wiener model

Consider a discrete-time Wiener system (Fig. 4.9) composed of a SISO linear dynamic system in a cascade with a SISO nonlinear element. The output $y(n)$ of the Wiener system at the time n is

$$y(n) = f\big(s(n)\big) + \varepsilon(n), \tag{4.50}$$

where $f(\cdot)$ is the steady state characteristic, $\varepsilon(n)$ is the additive output disturbance, and $s(n)$ is the output of the linear dynamic system:

$$s(n) = \frac{B(q^{-1})}{A(q^{-1})}u(n) \tag{4.51}$$

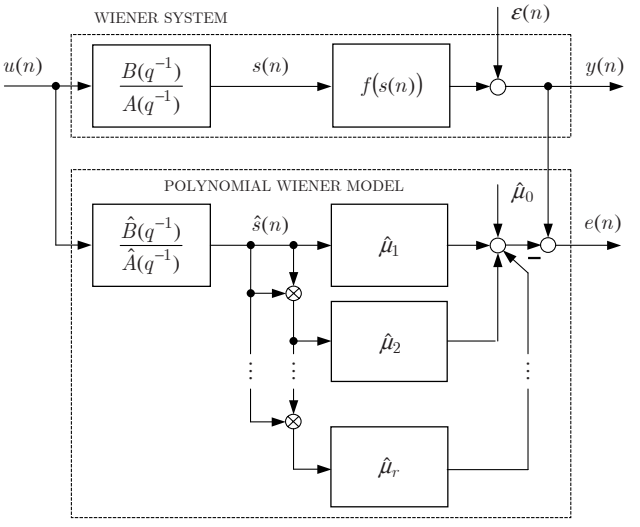

Fig. 4.9. Wiener system and its polynomial model

with

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}, \tag{4.52}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{nb} q^{-nb}, \tag{4.53}$$

where $a_1, \ldots, a_{na}, b_1, \ldots, b_{nb}$ are the parameters of the linear dynamic system. Assume that the linear dynamic system is casual and asymptotically stable, and $f(\cdot)$ is a continuous function. Assume also that the polynomials $A(q^{-1})$ and $A(q^{-1})$ are coprime and $u(n)$ has finite moments and is independent of $\varepsilon(k)$ for all n and k. The steady state characteristic of the system can be approximated by a polynomial $\hat{f}(\cdot)$ of the order r:

$$\hat{f}\big(\hat{s}(n)\big) = \hat{\mu}_0 + \hat{\mu}_1 \hat{s}(n) + \hat{\mu}_2 \hat{s}^2(n) + \cdots + \hat{\mu}_r \hat{s}^r(n), \tag{4.54}$$

where $\hat{s}(n)$ is the output of the linear dynamic system model

$$\hat{s}(n) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(n) \tag{4.55}$$

with

$$\hat{A}(q^{-1}) = 1 + \hat{a}_1 q^{-1} + \cdots + \hat{a}_{na} q^{-na}, \tag{4.56}$$

$$\hat{B}(q^{-1}) = \hat{b}_1 q^{-1} + \cdots + \hat{b}_{nb} q^{-nb}, \tag{4.57}$$

where $\hat{a}_1, \ldots, \hat{a}_{na}, \hat{b}_1, \ldots, \hat{b}_{nb}$ are the parameters of the linear dynamic model. Therefore, the parameter vector $\hat{\boldsymbol{\theta}}$ of the model defined by (4.54) and (4.55) is

$$\hat{\boldsymbol{\theta}} = \left[\hat{a}_1 \ldots \hat{a}_{na}\ \hat{b}_1 \ldots \hat{b}_{nb}\ \hat{\mu}_0\ \hat{\mu}_1 \ldots \hat{\mu}_r\right]^T. \tag{4.58}$$

4.2.2 Recursive prediction error method

The identification problem considered here can be formulated as follows: Given a set of input and output data $Z^N = \{(u(n), y(n)),\ k = 1, \ldots, N\}$ estimate the parameters of the Wiener system so that the predictions $\hat{y}(n|n-1)$ of the system output are close to the system output $y(n)$ in the sense of the following mean square error criterion:

$$J_N(\hat{\boldsymbol{\theta}}, Z^N) = \frac{1}{2N} \sum_{k=1}^{N} \big(y(n) - \hat{y}(n|n-1)\big)^2. \tag{4.59}$$

Given the gradient of the model output w.r.t. the parameter vector

$$\boldsymbol{\psi}(n) = \left[\frac{d\hat{y}(n|n-1)}{d\hat{\boldsymbol{\theta}}}\right]^T, \tag{4.60}$$

the RPE algorithm can be expressed by (2.97) – (2.99). In practice, it is useful to modify the criterion (4.59) with an exponential forgetting factor λ. The forgetting factor $\lambda \in [0, 1]$ and values close to 1 are commonly selected. To protect the algorithm from the so-called covariance blow-up phenomenon, other modifications of the algorithm may be useful that impose an upper bound on the eigenvalues of the matrix $\boldsymbol{P}(n)$ and are known as the constant trace and exponential forgetting and resetting algorithms [127].

4.2.3 Gradient calculation

The gradient $\boldsymbol{\psi}(n)$ of the model output w.r.t. to the model parameters is defined as

$$\boldsymbol{\psi}(n) = \left[\frac{\partial\hat{y}(n)}{\partial a_1} \ldots \frac{\partial\hat{y}(n)}{\partial a_{na}}\ \frac{\partial\hat{y}(n)}{\partial b_1} \ldots \frac{\partial\hat{y}(n)}{\partial b_{nb}}\ \frac{\partial\hat{y}(n)}{\partial\hat{\mu}_0}\ \frac{\partial\hat{y}(n)}{\partial\hat{\mu}_1} \ldots \frac{\partial\hat{y}(n)}{\partial\hat{\mu}_r}\right]^T. \tag{4.61}$$

Although the model given by (4.54) and (4.55) is a recurrent one due to the difference equation (4.55), the calculation of the gradient does not require much more computation than in the case of the pure static model. The only difference is in the calculation of partial derivatives of the model output w.r.t. the parameters of the linear dynamic model. This can be done with the sensitivity method solving by simulation the following set of linear difference equations [73, 74]:

$$\frac{\partial \hat{s}(n)}{\partial \hat{a}_k} = -\hat{s}(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{s}(n-m)}{\partial \hat{a}_k},\ k = 1, \ldots, na, \tag{4.62}$$

$$\frac{\partial \hat{s}(n)}{\partial \hat{b}_k} = u(n-k) - \sum_{m=1}^{na} \hat{a}_m \frac{\partial \hat{s}(n-m)}{\partial \hat{b}_k},\ k = 1, \ldots, nb. \tag{4.63}$$

Hence, partial derivatives of the model output w.r.t. the parameters $\hat{a}_k$ and $\hat{b}_k$ can be calculated as

$$\frac{\partial \hat{y}(n)}{\partial \hat{a}_k} = \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)} \frac{\partial \hat{s}(n)}{\partial \hat{a}_k}, \tag{4.64}$$

$$\frac{\partial \hat{y}(n)}{\partial \hat{b}_k} = \frac{\partial \hat{y}(n)}{\partial \hat{s}(n)} \frac{\partial \hat{s}(n)}{\partial \hat{b}_k}, \tag{4.65}$$

where

$$\frac{\partial \hat{y}(n)}{\partial \hat{s}(n)} = \hat{\mu}_1 + 2\hat{\mu}_2 \hat{s}(n) + \cdots + r\hat{\mu}_r \hat{s}^{r-1}(n). \tag{4.66}$$

Partial derivatives of the model output w.r.t. the parameters of the nonlinear element model are

$$\frac{\partial \hat{y}(n)}{\partial \hat{\mu}_j} = \hat{s}^j(n),\ j = 0, 1, \ldots, r. \tag{4.67}$$

Note that the derivation of the partial derivatives (4.62) and (4.63) is made under the assumption that the parameters of the linear dynamic model are time invariant. As this assumption is not true because of the sequential nature of the RPE algorithm, an approximate gradient is obtained. A more accurate evaluation of the gradient can be calculated using the truncated BPTT algorithm.

4.2.4 Pneumatic valve simulation example

The model of a pneumatic valve (2.100) – (2.101) was used in the simulation example. It was assumed that the system output $y(n)$ was additively disturbed by the zero-mean discrete white Gaussian noise $\varepsilon(n)$ with the standard deviation of $\sigma_\varepsilon = 0.005$ and 0.05:

$$y(n) = f\big(s(n)\big) + \varepsilon(n). \tag{4.68}$$

A sequence of 20000 pseudorandom numbers, uniformly distributed in $(0, 1)$, was used as the system input. Based on the simulated input-output data, the Wiener system was identified using the RPE algorithm. To compare estimation accuracy of the linear system parameters and the nonlinear function $f(\cdot)$, the indices (2.91) and (2.92) were used with $\{s(n)\}$ defined as a sequence of 100 linearly equally spaced values between $-\min\big(s(n)\big)$ and $\max\big(s(n)\big)$. The results shown in Tables 4.5 and 4.6 are illustrated in Figs 4.10 – 4.13.

In the example, the nonlinear characteristic is of an infinite order while a finite order model is estimated. In spite of the fact that the estimated parameters are different from the parameters of the polynomial approximating (2.101), the nonlinear characteristic is approximated quite well showing practical applicability of this approach.

Table 4.5. Parameter estimates

Parameter	Approximating polynomial	Estimated $\sigma_\varepsilon = 0.005$	Estimated $\sigma_\varepsilon = 0.05$
a_1	-1.4138	-1.4167	-1.4159
a_2	0.6065	0.6088	0.6092
b_1	0.1044	0.1059	0.0984
b_2	0.0833	0.0812	0.0899
μ_0	0.0010	0.0054	0.0595
μ_1	0.9530	1.0581	0.6871
μ_2	0.8149	-1.1304	-0.5716
μ_3	-11.651	-0.2255	0.1347
μ_4	34.749	0.8751	-0.0036
μ_5	-57.593	0.2007	-0.0121
μ_6	59.242	-0.3143	-0.0045
μ_7	-37.639	-0.3218	-0.0011
μ_8	13.5602	-0.0583	-0.0002
μ_9	-2.1210	0.2263	-0.0000

Table 4.6. Comparison of estimation accuracy, $s(j)$ – a pseudorandom sequence, uniformly distributed in $(\min\big(s(n)\big), \max\big(s(n)\big)$

Performance index	$\sigma_\varepsilon = 0.005$	$\sigma_\varepsilon = 0.05$
$\frac{1}{4}\sum_{j=1}^{2}\left[(\hat{a}_j - a_j)^2 + (\hat{b}_j - b_j)^2\right]$	2.95×10^{-6}	2.44×10^{-4}
$\frac{1}{100}\sum_{j=1}^{100}\left[\hat{f}\big(s(j)\big) - f\big(s(j)\big)\right]^2$	5.38×10^{-5}	1.87×10^{-3}

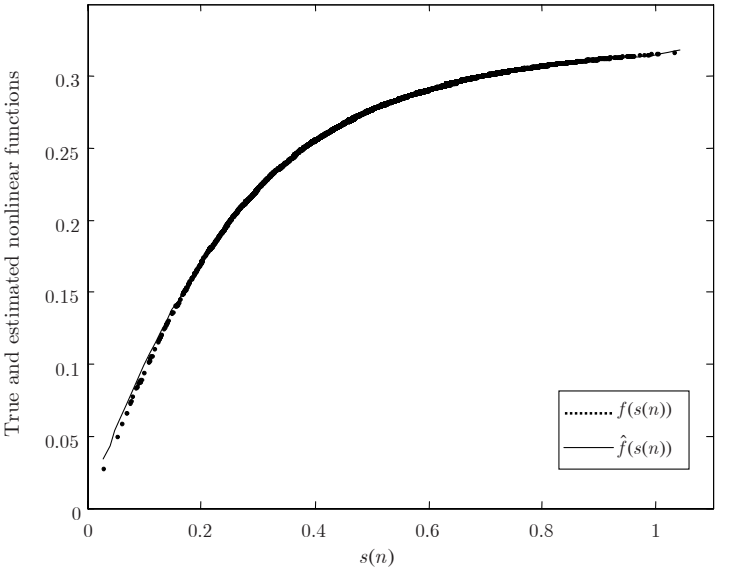

Fig. 4.10. True and estimated nonlinear functions ($\sigma_\varepsilon = 0.005$)

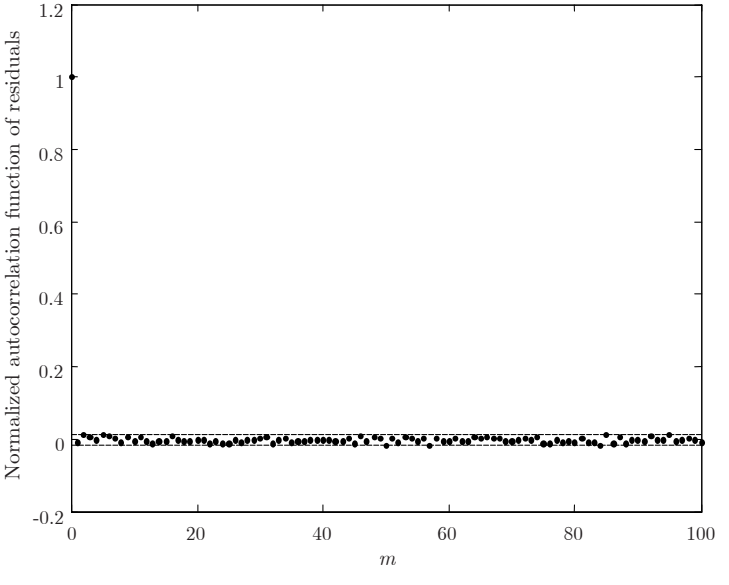

Fig. 4.11. Autocorrelation function of residuals and the 95% confidence interval ($\sigma_\varepsilon = 0.005$)

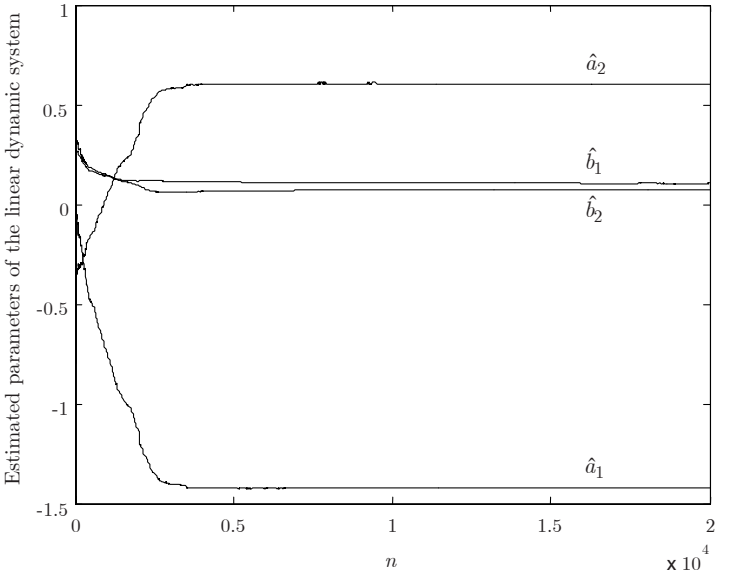

Fig. 4.12. Evolution of linear dynamic system parameter estimates ($\sigma_\varepsilon = 0.005$)

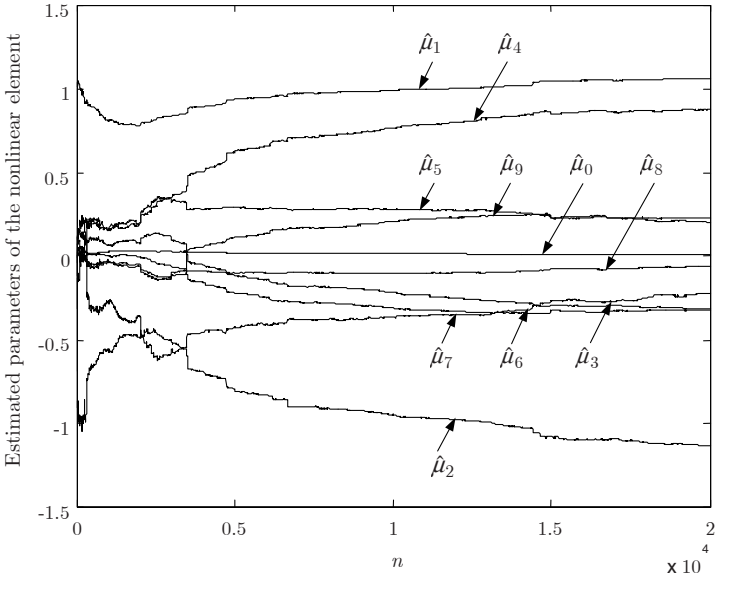

Fig. 4.13. Evolution of nonlinear element parameter estimates ($\sigma_\varepsilon = 0.005$)

4.3 Pseudolinear regression method

The pseudolinear regression approach to parameter estimation is based on the assumption that nonlinear components of a model can be neglected and the model can be treated as a linear-in-parameters one. For polynomial Wiener systems, we estimate the parameters of a pulse transform function and the parameters of nonlinear characteristic. It is assumed that the nonlinear characteristic contains the linear term. An obvious advantage of the pseudolinear regression approach, in comparison with other regression methods, is its low computational complexity and applicability for the identification of Wiener systems with noninvertible nonlinear characteristics. The pseudolinear regression method can be considered as a simplified prediction error approach, in which the exact gradient is replaced with an approximate one. Such a simplification reduces computational complexity of the method but may deteriorate its convergence rate.

4.3.1 Pseudolinear-in-parameters polynomial Wiener model

Consider the Wiener system (4.50) – (4.53) and its polynomial model (4.54) – (4.57). To derive the pseudolinear regression method, it is necessary to assume that the polynomial model $\hat{f}\big(\hat{s}(n)\big)$ has a nonzero linear term, i.e., $\hat{\mu}_1 \neq 0$. For convenience, we can assume that $\hat{\mu}_1 = 1$. Note that there is no loss of generality if we assume that $\hat{\mu}_1 = 1$, as the steady state gain of the linear dynamical model can be multiplied by $1/\hat{\mu}_1$. The polynomial model output can be written as

$$\begin{aligned}\hat{y}(n) = &- \hat{a}_1\hat{s}(n-1) - \cdots - \hat{a}_{na}\hat{s}(n-na) + \hat{b}_1 u(n-1) + \cdots \\ &+ \hat{b}_{nb}u(n-nb) + \hat{\mu}_0 + \hat{\mu}_2\hat{s}^2(n) + \cdots + \hat{\mu}_r\hat{s}^r(n) = \hat{\boldsymbol{\theta}}^T\boldsymbol{\psi},\end{aligned} \tag{4.69}$$

where

$$\hat{\boldsymbol{\theta}} = \big[\hat{\boldsymbol{\theta}}_l^T \;\; \hat{\mu}_0 \;\; \hat{\mu}_2 \ldots \hat{\mu}_r\big]^T, \tag{4.70}$$

$$\hat{\boldsymbol{\theta}}_l = \big[\hat{a}_1 \ldots \hat{a}_{na} \;\; \hat{b}_1 \ldots \hat{b}_{nb}\big]^T, \tag{4.71}$$

$$\boldsymbol{\psi}(n) = \big[\boldsymbol{\varphi}^T(n) \;\; 1 \;\; \hat{s}^2(n) \ldots - \hat{s}^r(n)\big]^T, \tag{4.72}$$

$$\boldsymbol{\varphi}(n) = \big[- \hat{s}(n-1) \ldots - \hat{s}(n-na) \;\; u(n-1) \ldots u(n-nb)\big]^T. \tag{4.73}$$

The model (4.69) is a linear function of the parameters $\hat{\mu}_0, \hat{\mu}_2, \ldots, \hat{\mu}_r$, but it is a nonlinear function of $\hat{a}_1,\ldots,\hat{a}_{na}$, $\hat{b}_1,\ldots,\hat{b}_{nb}$. This comes from the fact that both $\hat{s}^2(n),\ldots,\hat{s}^r(n)$ and $\hat{s}(n-1),\ldots,\hat{s}(n-na)$ depend on $\hat{\boldsymbol{\theta}}_l$.

4.3.2 Pseudolinear regression identification method

Minimization with respect to model parameters of the weighted cost function

$$J = \frac{1}{2}\sum_{j=1}^{N} \lambda^{N-j}\big(y(n) - \hat{y}(n)\big)^2 \tag{4.74}$$

results in the following identification algorithm:

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \boldsymbol{K}(n)\, e(n), \tag{4.75}$$

$$e(n) = y(n) - \boldsymbol{\psi}^T(n)\hat{\boldsymbol{\theta}}(n-1), \tag{4.76}$$

$$\boldsymbol{K}(n) = \boldsymbol{P}(n)\boldsymbol{\psi}(n) = \frac{\boldsymbol{P}(n-1)\boldsymbol{\psi}(n)}{\lambda + \boldsymbol{\psi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\psi}(n)}, \tag{4.77}$$

$$\boldsymbol{P}(n) = \frac{1}{\lambda}\Big(\boldsymbol{P}(n-1) - \boldsymbol{K}(n)\boldsymbol{\psi}^T(n)\boldsymbol{P}(n-1)\Big), \tag{4.78}$$

where λ denotes the exponential forgetting factor.

4.3.3 Simulation example

The Wiener system described by the second order pulse transfer function

$$\frac{B(q^{-1})}{A(q^{-1})} = \frac{0.125q^{-1} - 0.025q^{-2}}{1 - 1.75q^{-1} + 0.85q^{-2}} \tag{4.79}$$

and the nonlinear characteristic (Fig. 4.16)

$$f\big(s(n)\big) = -0.25 + s(n) + s^2(n) - 0.5s^3(n) - 0.2s^4(n) + 0.2s^5(n) \tag{4.80}$$

was used in the numerical example. The system was excited with a sequence of 10000 pseudo-random numbers of uniform distribution in $(-1, 1)$. The system output was disturbed additively with another pseudo-random sequence of uniform distribution in $(-\sqrt{3\alpha}, \sqrt{3\alpha})$.
The identification results, obtained at $\alpha = 3.34\times10^{-5},\ 3.34\times10^{-3},\ 3.34\times10^{-1}$ and the forgetting factor $\lambda = 0.9995$, are summarized in Tables 4.7 and 4.8 and illustrated in Figs. 4.15 – 4.17.

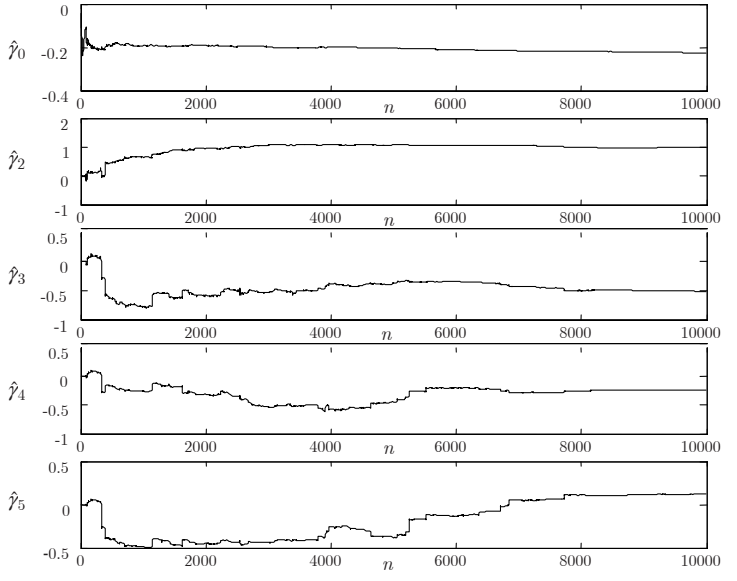

Fig. 4.14. Evolution of nonlinear element parameter estimates ($\alpha = 3.34 \times 10^{-3}$)

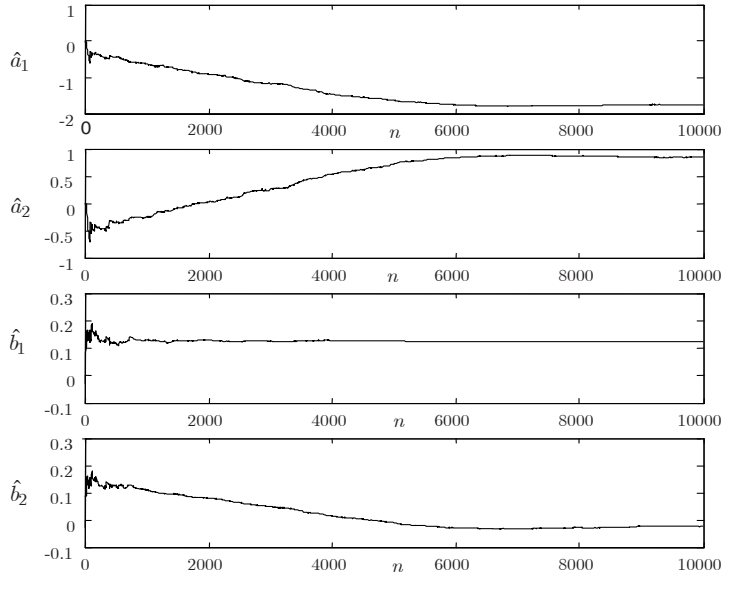

Fig. 4.15. Evolution of linear dynamic system parameter estimates ($\alpha = 3.34 \times 10^{-3}$)

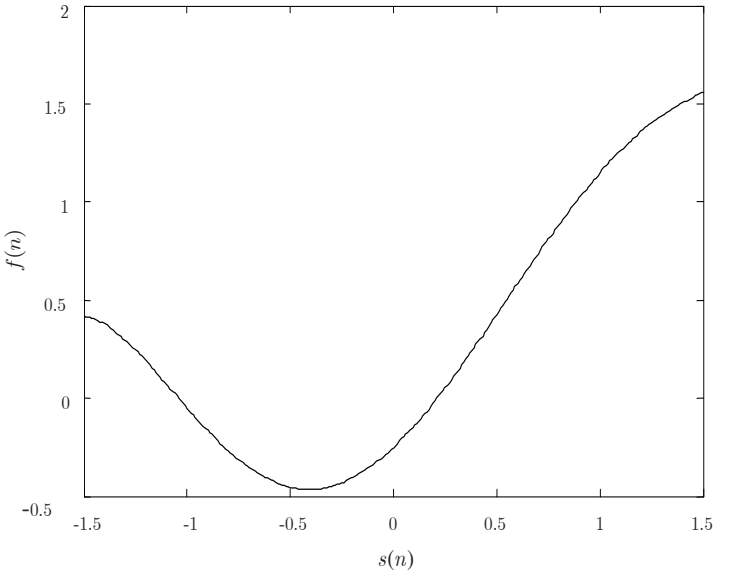

Fig. 4.16. Nonlinear element characteristic

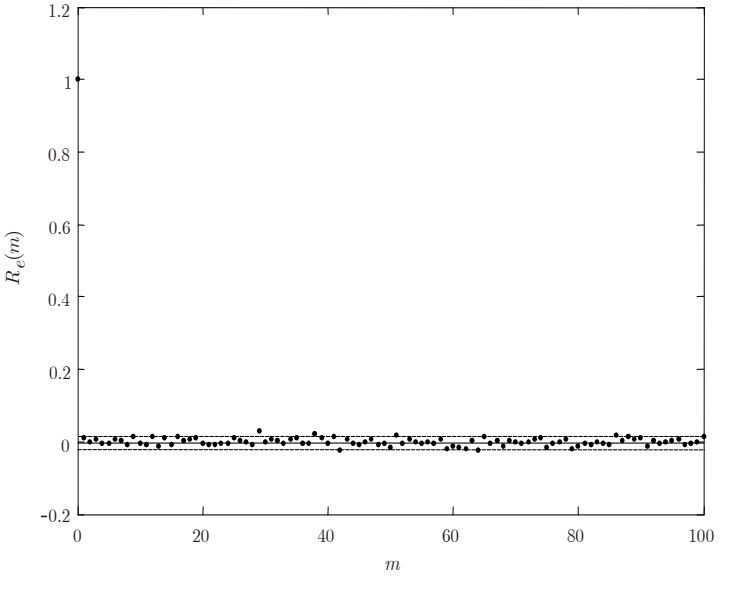

Fig. 4.17. Autocorrelation function of residuals and the 95% confidence interval ($\alpha = 3.3403 \times 10^{-3}$)

Table 4.7. Parameter estimates

Parameter	True value	Estimated	Estimated	Estimated
α		3.34×10^{-5}	3.34×10^{-3}	3.34×10^{-1}
a_1	-1.7500	-1.7504	-1.7500	-1.7327
a_2	0.8500	0.8504	0.8501	0.8374
b_1	0.1250	0.1249	0.1242	0.1199
b_2	-0.0250	-0.0251	-0.0245	-0.0124
μ_0	-0.2500	-0.2499	-0.2500	-0.2454
μ_2	1.0000	1.0003	1.0053	1.0035
μ_3	-0.5000	-0.4992	-0.4913	-0.5493
μ_4	-0.2000	-0.2007	-0.2040	-0.2834
μ_5	0.1000	0.0991	0.0912	0.1493

Table 4.8. Comparison of estimation accuracy

Index	$\alpha = 3.34 \times 10^{-5}$	$\alpha = 3.34 \times 10^{-3}$	$\alpha = 3.34 \times 10^{-1}$
$\frac{1}{N}\sum_{i=1}^{N} e(n)^2$	3.3656×10^{-6}	3.3408×10^{-3}	3.3441×10^{-1}
$\sum_{j=0}^{5}(\hat{\mu}_j - \mu_j)^2$	2.3762×10^{-6}	1.9759×10^{-4}	1.2501×10^{-2}

4.4 Summary

In this chapter, it has been shown that the linear-in-parameters definition of the modified equation error makes it possible to use the linear regression approach to estimate the parameters of Wiener systems with an invertible nonlinear element. As such an approach results in inconsistent parameter estimates, a combined least squares-instrumental variables method has been proposed to overcome this problem. Contrary to linear regression approaches, prediction error methods make it possible to identify Wiener systems with both invertible and noninvertible nonlinear characteristics. Moreover, there is no problem of parameter redundancy, as the number of the estimated parameters is equal to the total number of the model parameters $na + nb + r + 1$. In comparison with the prediction error method, the pseudo-linear regression approach has lower computational requirements as it uses an approximate gradient instead of the true one.

5

Polynomial Hammerstein models

In this chapter, we will review seven methods of the identification of Hammerstein systems which use Hammerstein models with a polynomial model of the nonlinear element. The reviewed methods use models of the linear dynamic system in the form of pulse transfer models [21, 27, 34, 63, 120, 161], or a Laguerre expansion of the impulse response [156]. Wiener system parameters are estimated with different procedures such as the ordinary least squares [27], iterative least squares [63, 120, 161], iterative correlation and steepest descent [156], prediction error [34], and pseudolinear regression [21] methods.

5.1 Noniterative least squares identification of Hammerstein systems

One of the best known identification methods for Hammerstein systems is based on the transformation of a nonlinear-in-parameters SISO identification problem into a linear-in-parameters MISO one and the application of least squares parameter algorithm. This method was originally proposed by Chang and Luus [27]. A short description of the method is given below.
First, we introduce a parallel polynomial Hammerstein model, i.e., a model with the linear part defined by the transfer function $\hat{B}(q^{-1})/\hat{A}(q^{-1})$:

$$\hat{y}(n) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}\hat{v}(n), \tag{5.1}$$

$$\hat{B}(q^{-1}) = \hat{b}_1 q^{-1} + \cdots + \hat{b}_{nb} q^{-nb}, \tag{5.2}$$

$$\hat{A}(q^{-1}) = 1 + \hat{a}_1 q^{-1} + \cdots + \hat{a}_{na} q^{-na}, \tag{5.3}$$

and the nonlinear part defined by a polynomial model:

$$\hat{v}(n) = \hat{f}\big(u(n)\big) = \hat{\mu}_1 u(n) + \hat{\mu}_2 u^2(n) + \ldots + \hat{\mu}_r u^r(n). \tag{5.4}$$

A. Janczak: Identification of Nonlinear Systems, LNCIS 310, pp. 143–157, 2005.

Without loss of generality, $\hat{\mu}_1$ can be assumed to be a unity, and $\hat{\mu}_2, \ldots, \hat{\mu}_r$ can be normalized accordingly. Then (5.1) can be expressed as

$$\hat{y}(n) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(n) + \sum_{k=2}^{r} \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} \hat{\mu}_k u^k(n). \tag{5.5}$$

Introducing $\hat{B}_1(q^{-1}) = \hat{B}(q^{-1}), \ldots, \hat{B}_k(q^{-1}) = \hat{\mu}_k \hat{B}(q^{-1})$ and $u_1(n) = u(n), \ldots, u_k(n) = u^k(n)$, (5.5) can be rewritten as

$$\hat{y}(n) = \sum_{k=1}^{r} \frac{\hat{B}_k(q^{-1})}{\hat{A}(q^{-1})} u_k(n). \tag{5.6}$$

The MISO Wiener model (Fig. 5.1) consists of r linear dynamic sub-models with a common denominator in each branch. Equivalently, (5.6) can be written as

$$\hat{y}(n) = \left[1 - \hat{A}(q^{-1})\right] \hat{y}(n) + \sum_{k=1}^{r} \hat{B}_k(q^{-1}) u_k(n). \tag{5.7}$$

By using the definitions of $\hat{A}(q^{-1})$ and $\hat{B}_k(q^{-1})$, we have

$$\hat{y}(n) = -\sum_{m=1}^{na} \hat{a}_m \hat{y}(n-m) + \sum_{k=1}^{r} \sum_{m=1}^{nb} \hat{b}_{km} u_k(n-m), \tag{5.8}$$

where $\hat{b}_{km}$ is the mth parameter of the polynomial $B_k(q^{-1})$. The parallel model (5.8) can be transformed into a linear-in-parameters series-parallel model using the available delayed system outputs $y(n-m)$ instead of the model outputs $\hat{y}(n-m)$:

$$\hat{y}(n) = -\sum_{m=1}^{na} \hat{a}_m y(n-m) + \sum_{k=1}^{r} \sum_{m=1}^{nb} \hat{b}_{km} u_k(n-m). \tag{5.9}$$

Introducing the parameter vector $\hat{\boldsymbol{\theta}}$ and the regression vector $\mathbf{x}(n)$:

$$\hat{\boldsymbol{\theta}} = \left[\hat{a}_1 \ldots \hat{a}_{na}\ \hat{b}_{11} \ldots \hat{b}_{1nb} \ldots \hat{b}_{r1} \ldots \hat{b}_{rnb}\right]^T, \tag{5.10}$$

$$\begin{aligned} \mathbf{x}(n) = \big[&- y(n-1) \ldots - y(n-na)\ u_1(n-1) \ldots u_1(n-nb) \ldots \\ &u_r(n-1) \ldots u_r(n-nb)\big]^T, \end{aligned} \tag{5.11}$$

we have

$$\hat{y}(n) = \mathbf{x}^T(n) \hat{\boldsymbol{\theta}}. \tag{5.12}$$

The minimization of the sum of N squared errors between the system output and the model output w.r.t. the regression vector $\hat{\boldsymbol{\theta}}$ gives

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \tag{5.13}$$

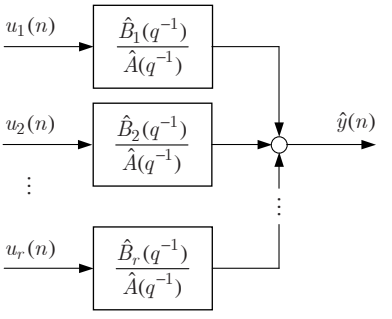

Fig. 5.1. Hammerstein model transformed into the MISO form

where

$$\mathbf{X} = \big[\mathbf{x}(1) \dots \mathbf{x}(N)\big]^T, \tag{5.14}$$

$$\mathbf{y} = \big[y(1) \dots y(N)\big]^T. \tag{5.15}$$

In this way, the parameters of the transformed MISO model (5.9) can be calculated, but our primary goal is to determine the parameters of the model defined by (5.1) and (5.4). The parameters $\hat{a}_m$ are elements of the vector $\hat{\boldsymbol{\theta}}$. Also, the parameters $\hat{b}_m$ are available directly as $\hat{b}_m = \hat{b}_{1m}$. The parameters $\hat{\mu}_k$ can be calculated from the remaining elements of $\hat{\boldsymbol{\theta}}$ but there is a certain amount of redundancy as

$$\hat{\mu}_k = \frac{\hat{b}_{mk}}{\hat{b}_{1k}},\ k = 2, \dots, r,\ m = 1, \dots, nb, \tag{5.16}$$

and nb different sets of parameters $\hat{\mu}$ can be calculated. Chang and Luus [27] suggested computing the root mean-square error (RMS) of the model output for all nb sets of $\hat{\mu}$ and accept the set that yields the least value for the RMS as a more reliable approach in comparison with computing the mean of the nb values.

5.2 Iterative least squares identification of Hammerstein systems

In the iterative least squares method of Narendra and Gallman [120], the parameters of the linear dynamic system and the nonlinear static element are updated separately and sequentially. This method utilizes an alternate adjustment of the parameters of the linear dynamic system and the nonlinear static element to minimize the sum of squared errors. The error $e(n)$ at the time n is

$$e(n) = y(n) - \left(- \sum_{m=1}^{na} \hat{a}_m y(n-m) + \sum_{m=1}^{nb} \hat{b}_m \hat{v}(n-m) \right), \tag{5.17}$$

where

$$\hat{v}(n) = \sum_{k=1}^{r} \hat{\mu}_k u^k(n). \tag{5.18}$$

Introducing $\hat{\boldsymbol{\theta}} = \left[\hat{\boldsymbol{\theta}}_a^T \ \hat{\boldsymbol{\theta}}_b^T\right]^T$ and $\mathbf{x}(n) = \left[\mathbf{x}_a^T(n) \ \mathbf{x}_b^T(n)\right]^T$, where

$$\hat{\boldsymbol{\theta}}_a = \left[\hat{a}_1 \dots \hat{a}_{na}\right]^T, \tag{5.19}$$

$$\hat{\boldsymbol{\theta}}_b = \left[\hat{b}_1 \dots \hat{b}_{nb}\right]^T, \tag{5.20}$$

$$\mathbf{x}_a(n) = \left[- y(n-1) \dots - y(n-na)\right]^T, \tag{5.21}$$

$$\mathbf{x}_b(n) = \left[\hat{v}(n-1) \dots \hat{v}(n-nb)\right]^T, \tag{5.22}$$

(5.17) can be written as

$$e(n) = y(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}} = y(n) - \mathbf{x}_a^T(n)\hat{\boldsymbol{\theta}}_a - \mathbf{x}_b^T(n)\hat{\boldsymbol{\theta}}_b. \tag{5.23}$$

Now, introducing the vectors $\hat{\boldsymbol{\mu}} = \left[\hat{\mu}_1 \dots \hat{\mu}_r\right]^T$ and $\mathbf{u}(n-1) = \left[u(n-1) \ u^2(n-1) \dots u^r(n-1)\right]^T$, the matrix $\mathbf{U}(n)$ can be defined:

$$\mathbf{U}(n) = \begin{bmatrix} \mathbf{u}^T(n-1) \\ \vdots \\ \mathbf{u}^T(n-nb) \end{bmatrix}. \tag{5.24}$$

From (5.18), it follows that the vector $\mathbf{x}_b(n)$ can be written as

$$\mathbf{x}_b(n) = \mathbf{U}(n)\hat{\boldsymbol{\mu}}, \tag{5.25}$$

and (5.23) becomes

$$e(n) = y(n) - \mathbf{x}_a^T(n)\hat{\boldsymbol{\theta}}_a - \left(\mathbf{U}(n)\hat{\boldsymbol{\mu}}\right)^T\hat{\boldsymbol{\theta}}_b. \tag{5.26}$$

In this iterative approach, parameter estimation is performed according to the following iterative procedure:

1. Based on N measurements of the system input and output signals and the assumed initial vector $\hat{\boldsymbol{\mu}}^{(1)}$, the parameter vector $\hat{\boldsymbol{\theta}}^{(1)} = [\hat{\boldsymbol{\theta}}_a^{(1)T} \ \hat{\boldsymbol{\theta}}_b^{(1)T}]^T$ of the model (5.23) is estimated:

$$\hat{\boldsymbol{\theta}}^{(1)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{5.27}$$

where

$$\mathbf{y} = \left[y(1) \dots y(N)\right]^T, \tag{5.28}$$

$$\mathbf{X} = \left[\mathbf{x}(1) \dots \mathbf{x}(N)\right]^T. \tag{5.29}$$

2. With $[\hat{\boldsymbol{\theta}}_a^{(1)T}\ \hat{\boldsymbol{\theta}}_b^{(1)T}]^T$, $\hat{\boldsymbol{\mu}}^{(2)}$ is calculated by minimizing the sum of squared errors for the model (5.26)

$$\hat{\boldsymbol{\mu}}^{(2)} = \big(\mathbf{U}^T\mathbf{U}\big)^{-1}\mathbf{U}^T(\mathbf{y} - \mathbf{X}_a\hat{\boldsymbol{\theta}}_a^{(1)}), \tag{5.30}$$

where

$$\mathbf{X}_a = \big[\mathbf{x}_a(1)\ldots\mathbf{x}_a(N)\big]^T, \tag{5.31}$$

$$\mathbf{U} = \big[\mathbf{U}^T(1)\hat{\boldsymbol{\theta}}_b^{(1)}\ldots\mathbf{U}^T(N)\hat{\boldsymbol{\theta}}_b^{(1)}\big]^T. \tag{5.32}$$

3. Using $\hat{\boldsymbol{\mu}}^{(2)}$, $\hat{\boldsymbol{\theta}}^{(2)} = [\hat{\boldsymbol{\theta}}_a^{(2)T}\ \hat{\boldsymbol{\theta}}_b^{(2)T}]^T$ is calculated according to the scheme used in Step 1, and the process is continued.

An obvious advantage of the iterative method is nonredundant model parameterization. The simulation study performed by Gallman [41] shows that such an iterative procedure gives significantly lower parameter variance and a slightly lower RMS error than the noniterative method of Chang and Luus. The iterative method applied to a variety of problems such as polynomial nonlinearity, a half-wave linear detector or saturating nonlinearity has proven to be very successful [120]. In spite of the reported successful experience, the method may not converge in some cases.

5.3 Identification of Hammerstein systems in the presence of correlated noise

A direct application of the method of Chang and Luus [27] for Hammerstein systems with correlated additive output disturbance results in asymptotically biased parameter estimates. To overcome this problem, an iterative least squares procedure, which estimates the parameters of the linear dynamic system, the parameters of the nonlinear static element and the parameters of a noise model, was proposed by Haist *et al.* [63]. Their approach is based on the assumption that the Hammerstein system with an output disturbance has the form

$$y(n) = \frac{B(q^{-1})}{A(q^{-1})}f\big(u(n)\big) + \frac{\varepsilon(n)}{D(q^{-1})}, \tag{5.33}$$

where

$$D(q^{-1}) = 1 + d_1q^{-1} + \cdots + d_{nd}q^{-nd}, \tag{5.34}$$

and $\varepsilon(n)$ is a zero-mean discrete-time white noise. Assume that the nonlinear element is described by the polynomial

$$f\big(u(n)\big) = \mu_1u(n) + \mu_2u^2(n) + \ldots + \mu_ru^r(n). \tag{5.35}$$

Denoting the undisturbed system output by $\bar{y}(n)$ and the additive disturbance at the system output by $\eta(n)$, we have

$$y(n) = \bar{y}(n) + \eta(n) \tag{5.36}$$

with

$$\bar{y}(n) = \sum_{k=1}^{r} \frac{B_k(q^{-1})}{A(q^{-1})} u_k(n), \tag{5.37}$$

$$\eta(n) = \frac{\varepsilon(n)}{D(q^{-1})}, \tag{5.38}$$

where $B_k(q^{-1}) = \mu_k B(q^{-1})$ and $u_k(n) = u^k(n)$ or, equivalently,

$$\begin{aligned} y(n) = &- \sum_{m=1}^{na} a_m \bar{y}(n-m) + \sum_{k=1}^{r} \sum_{m=1}^{nb} b_{km} u_k(n-m) \\ &- \sum_{m=1}^{nd} d_m \eta(n-m) + \varepsilon(n). \end{aligned} \tag{5.39}$$

Introducing the parameter vector $\boldsymbol{\theta}$ and the regression vector $\mathbf{x}(n)$:

$$\boldsymbol{\theta} = \left[a_1 \dots a_{na}\ b_{11} \dots b_{1nb} \dots b_{r1} \dots b_{rnb}\ d_1 \dots d_{nd}\right]^T, \tag{5.40}$$

$$\begin{aligned} \mathbf{x}(n) = \big[&- \bar{y}(n-1) \dots - \bar{y}(n-na)\ u_1(n-1) \dots u_1(n-nb) \dots \\ &u_r(n-1) \dots u_r(n-nb)\ - \eta(n-1) \dots - \eta(n-nd)\big]^T, \end{aligned} \tag{5.41}$$

we have

$$y(n) = \mathbf{x}^T(n)\boldsymbol{\theta} + \varepsilon(n). \tag{5.42}$$

Now, parameter updating can be made according to the following iterative procedure:

1. Since the sequences $\{\bar{y}(n)\}$ and $\{\eta(n)\}$ are not known initially, calculate the parameters of a deterministic part of the model using the noniterative procedure of Chang and Luus, described in Section 5.1. Generate an estimate of the sequence $\{\bar{y}(n)\}$ from (5.37) and calculate an estimate of the sequence $\{\eta(n)\}$ from (5.36) using the estimate of the sequence $\{\bar{y}(n)\}$.
2. Calculate improved parameter estimates from

$$\hat{\boldsymbol{\theta}} = \left[\sum_{i=1}^{N} \mathbf{x}(n)\mathbf{x}^T(n)\right]^{-1} \left[\sum_{i=1}^{N} \mathbf{x}(n) y(n)\right], \tag{5.43}$$

   where the estimated values of $\bar{y}(n)$ and $\eta(n)$ are used in $\mathbf{x}(n)$ instead of their unknown true values.
3. The improved parameter estimates yield another estimate of the sequence $\{\bar{y}(n)\}$. Also, the estimated sequence $\{\eta(n)\}$ is calculated again, from (5.36).
4. Continue the procedure from Step 2 until the change in the normalized RMS is less than some specified minimum.

To ensure numerical stability of the procedure, employing a stepping factor ϵ is suggested. With the stepping factor, the adopted value of the model parameter vector $\hat{\boldsymbol{\theta}}^{*(j+1)}$ at the iteration $j+1$ is calculated according to the formula

$$\hat{\boldsymbol{\theta}}^{*(j+1)} = \epsilon\, \hat{\boldsymbol{\theta}}^{(j+1)} + (1-\epsilon)\hat{\boldsymbol{\theta}}^{*(j)}, \tag{5.44}$$

where $\hat{\boldsymbol{\theta}}^{(j+1)}$ is the estimate of the parameter vector obtained from (5.43). As in the case of the method of Chang and Luus, it follows from (5.41) that there is some redundancy in parameter determination. In spite of its practically confirmed successful applications, no proof of the convergence of this method is available.

5.4 Identification of Hammerstein systems with the Laguerre function expansion

An example of identification methods which use orthonormal basis functions is the method proposed by Thathachar and Ramaswamy [156]. In this method, the nonlinear part of the Hammerstein model is represented by the polynomial (5.4), and the linear part – by a Laguerre expansion of its impulse response $h(n)$:

$$h(n) = \sum_{i=1}^{nl} A_i l_i(n), \tag{5.45}$$

where $l_i(n)$ is the ith Laguerre function, and A_i, $i = 1, \ldots, nl$, are the parameters. Discrete Laguerre functions are defined as

$$L_i(q^{-1}) = \sqrt{1-\exp(-2T)}\,\frac{\left(q^{-1}-\exp(-T)\right)^{i-1}}{\left(1-\exp(-T)q^{-1}\right)^{i}}, \text{ for } i = 1, 2, \ldots, \tag{5.46}$$

where T is the sampling period, and $L_i(q)$ is the positive half $\mathcal{Z}$ transform of $l_i(n)$:

$$L_i(q^{-1}) = \sum_{n=0}^{\infty} l_i(n) q^{-n}. \tag{5.47}$$

The functions $l_i(n)$ fulfill the condition of orthonormality:

$$\sum_{n=0}^{\infty} l_i(n) l_k(n) = \begin{cases} 0 \text{ for } i \neq k \\ 1 \text{ for } i = k \end{cases} \quad i, k \leqslant n, \tag{5.48}$$

and

$$l_i(n) = 0 \text{ for } n < 0 \text{ and all } i. \tag{5.49}$$

The identification problem is to determine the parameters $\hat{\mu}_k$, $k = 1, \ldots, r$, of the nonlinear part, and A_i, $i = 1, \ldots, nl$, of the linear part. The problem

is solved in an iterative manner, where each iteration comprises two steps. In the first step, the parameters $\hat{\mu}_k$ are kept constant at their values from the previous iteration and A_i are calculated by solving a set of linear equations. To start the identification procedure, some arbitrary initial values of $\hat{\mu}_k$ are assumed permitting the calculation of the output of the nonlinear part of the model.
The expressions for the parameters A_i can be derived from the input-output cross-correlation function of the linear part:

$$\left.\begin{aligned} A_1 &= \frac{C_{nl}}{B_{nl}} \\ A_s &= \frac{C_{nl-s+1} - \sum_{i=1}^{s-1} A_i B_{nl-s+i}}{B_{nl}} \\ A_{nl} &= \frac{C_1 - \sum_{i=1}^{nl-1} A_i B_i}{B_{nl}} \end{aligned}\right\}. \tag{5.50}$$

The coefficients $B_j,\ j = 1, \ldots, nl$, are calculated as the following averages:

$$B_j = \lim_{N\to\infty} \frac{1}{N+1} \sum_{n=0}^{N} u(n) f_j(n), \tag{5.51}$$

where $f_j(n)$ is the output of $L_j(q^{-1})$ excited by $\hat{f}\big(u(n)\big)$. In a similar way, the coefficients $C_j,\ j = 1, \ldots, nl$ are calculated as

$$C_j = \lim_{N\to\infty} \frac{1}{N+1} \sum_{n=0}^{N} u(n) y_{j-1}(n), \tag{5.52}$$

where $y_{j-1}(n)$ is the output of $L_{j-1}(q^{-1})$ excited by $y(n)$.
In the other step, the parameters A_i are kept constant while $\hat{\mu}_k$ are determined using an algorithm that adjusts $\hat{\mu}_k$ towards their optimal values minimizing the mean square error (MSE) between the model output and the system output. The minimization is carried out with gradient methods, i.e., steepest descent in the disturbance-free case, and stochastic approximation in the presence of an additive output disturbance. The MSE is defined as

$$J = \frac{1}{N} \sum_{n=1}^{N} \Big[\hat{\boldsymbol{\mu}}^T \sum_{m_1=0}^{\infty} h(m_1) \mathbf{u}(n-m_1) - y(n) \Big]^2, \tag{5.53}$$

where $\mathbf{u}(n) = \big[u(n)\ u^2(n) \ldots u^r(n)\big]^T$ and $\hat{\boldsymbol{\mu}}_i = \big[\hat{\mu}_1\ \hat{\mu}_2 \ldots \hat{\mu}_r\big]^T$. In the disturbance-free case, the parameters of the nonlinear element model are adjusted according to following rule:

$$\hat{\boldsymbol{\mu}}^{(new)} = \hat{\boldsymbol{\mu}}^{(old)} - \eta \frac{\partial J}{\hat{\boldsymbol{\mu}}}, \tag{5.54}$$

where η is the step size. In the presence of additive output disturbances, the iteration-dependent step size $\eta(n)$ is used instead of a constant one, which fulfills the following conditions:

$$\eta(n) \geqslant 0, \quad \sum_{k=1}^{\infty} \eta^2(n) < \infty, \quad \sum_{k=1}^{\infty} \eta(n) = \infty. \tag{5.55}$$

5.5 Prediction error method

In the prediction error method, the Hammerstein system is assumed to be given by

$$y(n) = \frac{B(q^{-1})}{A(q^{-1})} \sum_{k=1}^{r} \mu_k u^k(n) + \varepsilon(n), \tag{5.56}$$

where $\varepsilon(n)$ is a zero-mean white noise used to model disturbances. For the nonwhite case, the parameters of a noise filter should also be identified but for the sake of complexity this problem is not considered here. The one-step-ahead predictor for the system output has the form [34]:

$$\hat{y}(n|n-1) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} \sum_{k=1}^{r} \hat{\mu}_k u^k(n). \tag{5.57}$$

The parameter vector $\hat{\boldsymbol{\theta}} = \left[\hat{a}_1 \ldots \hat{a}_{na}\ \hat{b}_1 \ldots \hat{b}_{nb}\ \hat{\mu}_1 \ldots \hat{\mu}_r\right]^T$ should be estimated based on a set of N input-output measurements to minimize the sum of squared prediction errors

$$J = \frac{1}{N} \sum_{i=1}^{N} e^2(n, \hat{\boldsymbol{\theta}}), \tag{5.58}$$

where

$$e(n, \hat{\boldsymbol{\theta}}) = y(n) - \hat{y}(n|n-1). \tag{5.59}$$

The prediction error method requires the calculation of the gradient of (5.58) w.r.t. its parameters. For the Hammerstein model, partial derivatives of (5.58) are nonlinear in the parameters. Therefore, since a direct solution to the optimization problem cannot be found, iterative methods have to be used. The prediction error method is a second order gradient-based technique, in which the parameter vector $\hat{\boldsymbol{\theta}}$ is adjusted along the negative gradient of (5.58):

$$\hat{\boldsymbol{\theta}}^{(j)} = \hat{\boldsymbol{\theta}}^{(j-1)} - \eta_j H^{-1}(\hat{\boldsymbol{\theta}}^{(j-1)}) G(\hat{\boldsymbol{\theta}}^{(j-1)}), \tag{5.60}$$

where η_j is the step size, $H(\cdot)$ is the Hessjan of (5.58) or its approximation, and $G(\cdot)$ is the gradient of (5.58). The gradient of (5.58) can be computed as

$$G(\hat{\boldsymbol{\theta}}^{(j)}) = \frac{dJ}{d\hat{\boldsymbol{\theta}}} = \frac{2}{N}\sum_{i=1}^{N} e(n,\hat{\boldsymbol{\theta}})\frac{\partial e(n,\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}} = -\frac{2}{N}\sum_{i=1}^{N} e(n,\hat{\boldsymbol{\theta}})\boldsymbol{\psi}(n), \tag{5.61}$$

where

$$\begin{aligned}\boldsymbol{\psi}(n) = \Bigg[&\frac{\partial \hat{y}(n|n-1)}{\partial \hat{a}_1}\cdots\frac{\partial \hat{y}(n|n-1)}{\partial \hat{a}_{na}}\ \frac{\partial \hat{y}(n|n-1)}{\partial \hat{b}_1}\cdots\frac{\partial \hat{y}(n|n-1)}{\partial \hat{b}_{nb}}\\ &\frac{\partial \hat{y}(n|n-1)}{\partial \hat{\mu}_1}\cdots\frac{\partial \hat{y}(n|n-1)}{\partial \hat{\mu}_r}\Bigg]\end{aligned} \tag{5.62}$$

with

$$\frac{\partial \hat{y}(n|n-1)}{\partial \hat{a}_k} = -\frac{1}{\hat{A}(q^{-1})}\sum_{m=1}^{r}\frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}\hat{\mu}_m u^m(n-k), \tag{5.63}$$

$$\frac{\partial \hat{y}(n|n-1)}{\partial \hat{b}_k} = \frac{1}{\hat{A}(q^{-1})}\sum_{m=1}^{r}\hat{\mu}_m u^m(n-k), \tag{5.64}$$

$$\frac{\partial \hat{y}(n|n-1)}{\partial \hat{\mu}_m} = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})}u^m(n). \tag{5.65}$$

The main drawback of using the Hessjan in (5.60) is that it requires second order derivatives. To avoid the calculation of second order derivatives, an approximate Hessjan can be used. In the Levenberg-Marquardt method, the approximate Hessjan is calculated as

$$H(\hat{\boldsymbol{\theta}}^{(j)}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial e(n,\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}}\frac{\partial e^T(n,\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}} + \mu I = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\psi}(n)\boldsymbol{\psi}^T(n) + \mu I, \tag{5.66}$$

where μ is a nonnegative small scalar and I is the identity matrix with an appropriate dimension. The iterative prediction algorithm comprises the following steps:

1. Start iterations with an initial estimate of the parameters $\hat{\boldsymbol{\theta}}^{(0)}$ and set $\hat{\mu}_1 = 1$.
2. Pick a small value for μ (a typical choice is 0.0001).
3. Compute $\hat{y}(n|n-1)$ and $J(\hat{\boldsymbol{\theta}}^{(j-1)})$.
4. Compute the gradient and the Hessjan through (5.61) – (5.66).
5. Update parameter estimates through (5.60), and calculate $J(\hat{\boldsymbol{\theta}}^{(j)})$.
6. If $J(\hat{\boldsymbol{\theta}}^{(j)}) > J(\hat{\boldsymbol{\theta}}^{(j-1)})$, decrease μ by a factor (say 10) and go to Step 4. $J(\hat{\boldsymbol{\theta}}^{(j)}) < J(\hat{\boldsymbol{\theta}}^{(j-1)})$, update solution and increase μ by a factor (say 10) and go to Step 2.

Having the rules for the calculation of the gradient derived, a recursive version of the algorithm can be implemented easily, see Equations (2.97) – (2.99).

5.6 Identification of MISO systems with the pseudolinear regression method

Consider a MISO Hammerstein model with nu inputs and the output disturbed additively by correlated measurement noise [22]:

$$y(n) = \sum_{j=1}^{nu} \frac{B_j(q^{-1})}{A_j(q^{-1})} f_j\big(u_j(n)\big) + \frac{C(q^{-1})}{D(q^{-1})}\varepsilon(n), \tag{5.67}$$

and the polynomial models of static nonlinearities

$$f_j\big(u_j(n)\big) = \mu_{j1}u_j(n) + \ldots + \mu_{jr_j}u_j^{r_j}(n), \tag{5.68}$$

where

$$A_j(q^{-1}) = 1 + a_{j1}q^{-1} + \ldots + a_{jna_j}q^{-na_j}, \tag{5.69}$$

$$B_j(q^{-1}) = 1 + b_{j1}q^{-1} + \ldots + b_{jnb_j}q^{-nb_j}, \tag{5.70}$$

$$C(q^{-1}) = 1 + c_1q^{-1} + \ldots + c_{nc}\,q^{-nc}, \tag{5.71}$$

$$D(q^{-1}) = 1 + d_1q^{-1} + \ldots + d_{nd}\,q^{-nd}. \tag{5.72}$$

Assume that $\varepsilon(n)$ is a zero-mean white noise, the model (5.67) is asymptotically stable, and the polynomial orders na_j, nb_j, r_j, nc, and nd are known. The model (5.67) can be transformed into the following equivalent form:

$$\bar{A}(q^{-1})y(n) = \sum_{j=1}^{nu} \bar{B}_j(q^{-1}) f_j\big(u_j(n)\big) + \frac{\bar{C}(q^{-1})}{D(q^{-1})}\varepsilon(n), \tag{5.73}$$

where

$$\bar{A}(q^{-1}) = \prod_{k=1}^{nu} A_j(q^{-1}) = 1 + \bar{a}_1q^{-1} + \ldots + \bar{a}_{na}q^{-na}, \tag{5.74}$$

$$\bar{B}_j(q^{-1}) = B_j(q^{-1}) \prod_{\substack{k=1 \\ k\neq j}}^{nu} A_k(q^{-1}) = \bar{b}_{j1}q^{-1} + \ldots + \bar{b}_{jnab_j}q^{-nab_j}, \tag{5.75}$$

$$\bar{C}(q^{-1}) = C(q^{-1})\bar{A}(q^{-1}) = 1 + \bar{c}_1q^{-1} + \ldots + \bar{c}_{nac}q^{-nac}, \tag{5.76}$$

with $na = na_1 + \ldots + na_{nu}$, $nab_j = na_1 + \ldots + na_{j-1} + nb_j + na_{j+1} + \ldots + na_{nu}$, $nac = na + nc$. Performing another transformation results in a model in the pseudolinear-in-parameters form:

$$\mathcal{A}(q^{-1})y(n) = \sum_{j=1}^{nu} \mathcal{B}_j(q^{-1}) f_j\big(u_j(n)\big) + \bar{C}(q^{-1})\varepsilon(n), \tag{5.77}$$

where

$$\mathcal{A}(q^{-1}) = \bar{A}(q^{-1})D(q^{-1}) = 1 + \alpha_1 q^{-1} + \ldots + \alpha_{n\alpha} q^{-n\alpha}, \tag{5.78}$$

$$\mathcal{B}_j(q^{-1}) = \bar{B}_j(q^{-1})D(q^{-1}) = \beta_{j1} q^{-1} + \ldots + \beta_{jn\beta_j} q^{-n\beta_j}, \tag{5.79}$$

with $n\alpha = na + nd$ and $n\beta_j = nab_j + nd$. Introducing the parameter vector $\boldsymbol{\theta}$,

$$\begin{aligned}\boldsymbol{\theta} = \big[&\alpha_1 \ldots \alpha_{n\alpha}\ \beta_{11}\mu_{11} \ldots \beta_{11}\mu_{1r_1} \ldots \beta_{1n\beta_1}\mu_{11} \ldots \beta_{1n\beta_1}\mu_{1r_1} \ldots \\ &\beta_{nu1}\mu_{nu1} \ldots \beta_{nu1}\mu_{nur_{nu}} \ldots \beta_{nun\beta_{nu}}\mu_{nu1} \ldots \beta_{nun\beta_{nu}}\mu_{nur_{nu}} \\ &\bar{c}_1 \ldots \bar{c}_{nac}\big]^T,\end{aligned} \tag{5.80}$$

and the vector $\mathbf{x}_0(n)$,

$$\begin{aligned}\mathbf{x}_0(n) = \big[&y(n-1) \ldots y(n-n\alpha)\ u_1(n) \ldots u_1^{r_1}(n) \ldots u_1(i-n\beta_1) \ldots \\ &u_1^{r_1}(i-n\beta_1) \ldots u_{nu}(n) \ldots u_{nu}^{r_{nu}}(n) \ldots u_{nu}(n-n\beta_1) \ldots \\ &u_{nu}^{r_{nu}}(n-n\beta_1)\ \ \varepsilon(n-1) \ldots \varepsilon(n-nac)\big]^T,\end{aligned} \tag{5.81}$$

we have

$$y(n) = \mathbf{x}_0^T(n)\boldsymbol{\theta} + \varepsilon(n). \tag{5.82}$$

Introduce the prediction error $e(n)$

$$e(n) = y(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}(n-1), \tag{5.83}$$

where $\hat{\boldsymbol{\theta}}$ is the vector of adjustable parameters

$$\begin{aligned}\hat{\boldsymbol{\theta}} = \big[&\hat{\alpha}_1 \ldots \hat{\alpha}_{n\alpha}\ \hat{\beta}_{11}\hat{\mu}_{11} \ldots \hat{\beta}_{11}\hat{\mu}_{1r_1} \ldots \hat{\beta}_{1n\beta_1}\hat{\mu}_{11} \ldots \hat{\beta}_{1n\beta_1}\hat{\mu}_{1r_1} \ldots \\ &\hat{\beta}_{nu1}\hat{\mu}_{nu1} \ldots \hat{\beta}_{nu1}\hat{\mu}_{nur_{nu}} \ldots \hat{\beta}_{nun\beta_{nu}}\hat{\mu}_{nu1} \ldots \hat{\beta}_{nun\beta_{nu}}\hat{\mu}_{nur_{nu}} \\ &\hat{\bar{c}}_1 \ldots \hat{\bar{c}}_{nac}\big]^T,\end{aligned} \tag{5.84}$$

and $\mathbf{x}(n)$ is defined as

$$\begin{aligned}\mathbf{x}(n) = \big[&y(n-1) \ldots y(n-n\alpha)\ u_1(n) \ldots u_1^{r_1}(n) \ldots u_1(i-n\beta_1) \ldots \\ &u_1^{r_1}(i-n\beta_1) \ldots u_{nu}(n) \ldots u_{nu}^{r_{nu}}(n) \ldots u_{nu}(n-n\beta_1) \ldots \\ &u_{nu}^{r_{nu}}(n-n\beta_1)\ \ e(n-1) \ldots e(n-nac)\big]^T.\end{aligned} \tag{5.85}$$

The estimate $\hat{\boldsymbol{\theta}}(n)$ of $\boldsymbol{\theta}_0$, which minimizes the sum of squared errors between the system and model outputs, can be obtained with the recursive pseudolinear regression algorithm as follows:

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)e(n), \tag{5.86}$$

$$\mathbf{K}(n) = \boldsymbol{P}(n)\mathbf{x}(n) = \frac{\boldsymbol{P}(n-1)\mathbf{x}(n)}{1 + \mathbf{x}^T(n)\boldsymbol{P}(n-1)\mathbf{x}(n)}, \tag{5.87}$$

$$\boldsymbol{P}(n) = \boldsymbol{P}(n-1) - \mathbf{K}(n)\mathbf{x}^T(n)\boldsymbol{P}(n-1). \tag{5.88}$$

5.7 Identification of systems with two-segment nonlinearities

In some cases, only polynomials of a higher order can approximate nonlinear characteristics adequately. An increase in the polynomial order r causes a multiple increase in the overall number of parameters in the linear-in-parameters model (5.12). Moreover, a single polynomial model can be inaccurate in a whole range of the system input signal. An alternative approach to single polynomial model solutions, discussed earlier, was proposed by Vörös [161]. In this method, a two-segment description of the nonlinear characteristic, composed of separate polynomial maps for positive and negative inputs, is used. The main motivation for such an approach is a better fit without increasing polynomial orders for some types of nonlinearities.
Assume that the nonlinear characteristic $\hat{f}(\cdot)$ is

$$\hat{v}(n) = \hat{f}\big(u(n)\big) = \begin{cases} \hat{f}_1\big(u(n)\big), \text{ if } u(n) > 0 \\ \hat{f}_2\big(u(n)\big), \text{ if } u(n) < 0 \end{cases}, \tag{5.89}$$

where

$$\hat{f}_1\big(u(n)\big) = \sum_{k=1}^{r} \hat{\mu}_{1k} u^k(n), \tag{5.90}$$

$$\hat{f}_2\big(u(n)\big) = \sum_{k=1}^{r} \hat{\mu}_{2k} u^k(n). \tag{5.91}$$

Introducing a switching sequence $\{g(n)\}$,

$$g(n) = g\big(u(n)\big) = \begin{cases} 0, \text{ if } u(n) > 0 \\ 1, \text{ if } u(n) < 0 \end{cases}, \tag{5.92}$$

(5.89) can be written as

$$\hat{v}(n) = \hat{f}_1\big(u(n)\big) + \big[\hat{f}_2\big(u(n)\big) - \hat{f}_1\big(u(n)\big)\big] g(n). \tag{5.93}$$

Then taking into account (5.90) and (5.91),

$$\hat{v}(n) = \sum_{k=1}^{r} \hat{\mu}_{1k} u^k(n) + \sum_{k=1}^{r} p_k u^k(n) g(n), \tag{5.94}$$

where

$$p_k = \hat{\mu}_{2k} - \hat{\mu}_{1k}. \tag{5.95}$$

The substitution of (5.94) into (5.1) results in a model which is nonlinear in the parameters. However, assuming $\hat{b}_1 = 1$, the model (5.1) can be expressed as

$$\hat{y}(n) = \hat{v}(n-1) + \big(\hat{B}(q^{-1}) - 1\big)\hat{v}(n) + \big(1 - \hat{A}(q^{-1})\big)\hat{y}(n). \tag{5.96}$$

Then substituting (5.94) only for $\hat{v}(n-1)$, the following pseudolinear-in-parameters model of the parallel type can be obtained:

$$\begin{aligned}\hat{y}(n) = &\sum_{k=1}^{r} \hat{\mu}_{1k} u^k(n-1) + \sum_{k=1}^{r} p_k u^k(n-1) g(n-1) \\ &+ \big(\hat{B}(q^{-1}) - 1\big)\hat{v}(n) + \big(1 - \hat{A}(q^{-1})\big)\hat{y}(n).\end{aligned} \tag{5.97}$$

Replacing $\hat{y}(n)$ with $y(n)$ transforms the model (5.97) into the series-parallel form:

$$\begin{aligned}\hat{y}(n) = &\sum_{k=1}^{r} \hat{\mu}_{1k} u^k(n-1) + \sum_{k=1}^{r} p_k u^k(n-1) g(n-1) \\ &+ \big(\hat{B}(q^{-1}) - 1\big)\hat{v}(n) + \big(1 - \hat{A}(q^{-1})\big)y(n)\end{aligned} \tag{5.98}$$

or

$$\hat{y}(n) = \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}, \tag{5.99}$$

where the parameter vector is defined as

$$\hat{\boldsymbol{\theta}} = \big[\hat{\mu}_{11} \dots \hat{\mu}_{1r}\ p_1 \dots p_r\ \hat{b}_2 \dots \hat{b}_{nb}\ \hat{a}_1 \dots \hat{a}_{na}\big]^T, \tag{5.100}$$

and the regression vector is

$$\begin{aligned}\mathbf{x}(n) = \big[&u(n-1) \dots u^r(n-1)\ u(n-1)g(n-1) \dots u^r(n-1)g(n-1) \\ &\hat{v}(n-2) \dots \hat{v}(n-nb)\ -y(n-1) \dots -y(n-na)\big]^T.\end{aligned} \tag{5.101}$$

The model (5.98) is also of the pseudolinear form as $\hat{v}(n)$ is an unmeasurable variable which depends on the parameters of the nonlinear function $\hat{f}(.)$. Therefore, no noniterative algorithm can be applied to estimate $\hat{\boldsymbol{\theta}}$. Vörös proposed an iterative algorithm which uses the preceding parameter estimates of $\hat{\mu}_{1k}$, and p_k to estimate $\hat{v}(n)$. Denote with $\hat{v}^{(j)}(n)$, $\hat{\mu}_{1k}^{(j)}$, and $p_k^{(j)}$ the estimates of $\hat{v}(n)$, $\hat{\mu}_{1k}$ and p_k obtained at the step j:

$$\hat{v}^{(j)}(n) = \sum_{k=1}^{r} \hat{\mu}_{1k}^{(j)} u^k(n) + \sum_{k=1}^{r} p_k^{(j)} u^k(n) g(n). \tag{5.102}$$

Then the error to be minimized can be expressed as

$$e(n) = y(n) - \mathbf{x}^{(j)T}(n)\hat{\boldsymbol{\theta}}(j+1), \tag{5.103}$$

where $\mathbf{x}^{(j)}(n)$ is the regression vector with the estimates of $\hat{v}(n)$ calculated according to (5.102), and $\hat{\boldsymbol{\theta}}(j+1)$ is the $(j+1)$th estimate of $\hat{\boldsymbol{\theta}}$.
The iterative identification procedure can be divided into the following steps:

1. Using the regression vector $\mathbf{x}^{(j)}(n)$, minimize a proper criterion based on (5.103) to estimate $\hat{\boldsymbol{\theta}}(j+1)$.

2. Using (5.102), calculate $\hat{v}^{(j+1)}(n)$.
3. Repeat Steps 1 and 2 until the parameter estimates converge to constant values.

Testing the above identification procedure with differently shaped two-segment polynomial and exponential nonlinearities has revealed its good convergence properties, although the formal proof of the convergence is not available. Although the iterative procedure which uses the whole input-output data set was originally proposed by Vörös, the derivation of the sequential version of the method is straightforward.
A similar approach can also be used for the identification of discontinuous Hammerstein systems, i.e., systems with the nonlinear element described by a discontinuous function, or Wiener systems with this type of nonlinearity [160].

5.8 Summary

We have presented seven different methods of the identification of Hammerstein systems, which use Hammerstein models with a polynomial model of the nonlinear element. The iterative least squares method of Narendra and Gallman and the noniterative least squares method of Chang and Luus are the oldest and belong to the best known methods. Both of them have some drawbacks. While the convergence of the first one is not guaranteed, the drawbacks of the other one are parameter redundancy and a huge number of parameters in the case of high order models. The iterative method of Haist *et al.* [63] also uses a non-unique parameterization but, in contrast to a one-step method of Chang and Luus, its advantage are consistent parameter estimates in the case of correlated output noise. In the method of Thathachar and Ramaswamy, a Laguerre expansion is used to represent the linear part of the model. The parameters of the model are adjusted iteratively with a gradient based method. The parameters of a pulse transfer function model of the linear dynamic system and a polynomial model of the nonlinear element are estimated iteratively with the Levenberg-Marquardt method in the prediction error approach discussed by Eskinat *et al.* [34]. An alternative to the prediction error method is the pseudolinear regression method proposed by Boutayeb *et al.* Finally, the iterative method of Vörös allows one to identify Hammerstein systems using a two-segment polynomial model of the nonlinear element. The method uses a linear regression approach but a formal proof of convergence is not available.

6

Applications

This chapter starts with a brief survey of the reported applications of Wiener and Hammerstein models in both system modelling and control. Next, the estimation of parameter changes in the context of fault detection and isolation is considered in Section 6.2. Modelling vapor pressure dynamics in a five stage sugar evaporation station is studied in Section 6.3. Two nominal models of the process, i.e., a linear model and a neural network Wiener model are developed based on the real process data recorded at the Lublin Sugar Factory in Poland. Finally, Section 6.4 summarizes the results.

6.1 General review of applications

A few authors [39, 42, 95, 96, 128, 130, 133, 134, 154] have studied the control of the pH neutralization process with control methods that use Wiener and Hammerstein models.

Based on the Hammerstein model, Fruzzetti *et al.* [39] proposed a model predictive control (MPC) strategy for a pH neutralization process and a binary distillation column. MPC is a control strategy in which a model of the process is used to predict future process outputs [62, 127]. Using the Hammerstein model and an inverse model of the nonlinear element, it is possible to apply a linear MPC controller in the analyzed control scheme. The controlled system is identified using the Hammerstein model, defined as a special form or the NARMAX model, and applying a forward regression orthogonal estimator [18].

A model of the pH neutralization process was employed by Gerkšič *et al.* [42] for testing the MPC scheme. The actual and future control values are calculated minimizing a performance criterion over a certain future horizon. The method of Gerkšič *et al.* uses a two-step procedure for Wiener system identification. In the first step, with a testing input signal slowly varied from the lower to the upper edge of the working range, a polynomial model of the

A. Janczak: Identification of Nonlinear Systems, LNCIS 310, pp. 159–185, 2005.

nonlinear element is determined. Another testing signal composed of a sequence of random steps and a small noise sequence is used in the other step to identify the linear system. The performance of three MPC schemes with different disturbance rejection techniques was compared. Also, comparisons with other control schemes were made showing that MPC is superior to the proportional plus integral (PI) scheme.

The modelling of the pH neutralization process with a polynomial model of the inverse nonlinear element and a frequency response model of the linear element Wiener was studied by Kalafatis *et al.* [95, 96]. In a pilot scale pH plant [95], a continuous stirred tank reactor (CSTR), a strong base (NaOH) reacts with feed streams of a strong acid (HCL) and a buffering solution ($NaHCO_3$). The flow rate of the strong base is the process input and the pH of the exiting stream is the process output.

An experimental pH neutralization process, in which a 0.82 mM HCL solution is added to a 2.0 mM NaOH solution, was examined by Norquay *et al.* [128]. The Wiener model of the process, consisting of a step-response model followed by cubic spline nonlinearity, is determined based on a response to an input signal with a random amplitude and frequency and applied to the MPC scheme. The obtained Wiener model gives an excellent fit to data in comparison with a linear step response model, which captures system dynamics but not the gain.

In [130], Norquay *et al.* discuss two different Wiener system identification procedures and apply Wiener models in MPC of a pH neutralization process. In a two stage procedure, a nonlinear element model is fitted to the steady-state data followed by the estimation of linear model parameters. A single stage procedure calculates the parameters of both models simultaneously with prediction error techniques.

MPC used by Norquay *et al.*, see Fig. 6.1 for the block diagram, is based on the internal model control scheme (IMC) [127]. In Fig. 6.1, G_p represents the controlled process, G_c and G_m represent the controller and the process model, respectively. The inverse model of the nonlinear element that follows the Wiener model cancels its nonlinear gain. Applying another inverse model of the nonlinear element, nonlinear properties of the neutralization process are also cancelled. In this way, the nonlinear predictive control problem reduces to a linear one.

It has also been shown by experimental evaluation and testing that MPC developed by Norquay *et al.* outperforms the proportional plus integral plus derivative (PID) and linear MPC control strategies.

The model reference adaptive control scheme for the control of the pH neutralization process in a continuous flow reactor was studied by Pajunen [133]. In this approach, the nonlinear element is modelled by a piecewise linear function and a pulse transfer model is used as a model of the linear system. The parameters of the controller are calculated recursively with the least squares type adaptation algorithm. The method is useful for systems with unknown and time-varying parameters that can be described by the Wiener model. Global

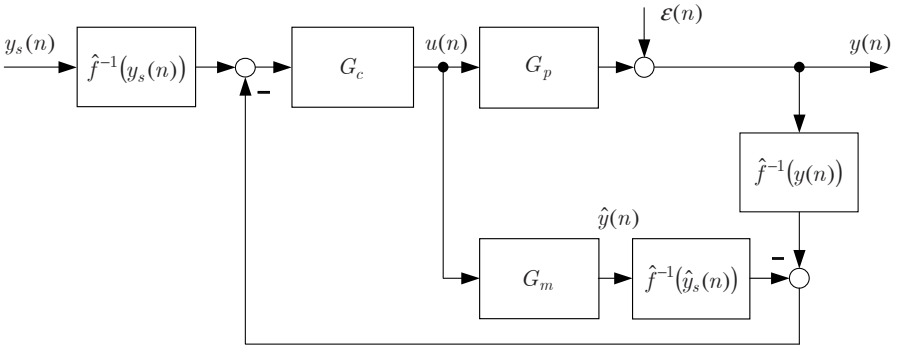

Fig. 6.1. Wiener-model-based prediction control based on the inversion of nonlinearity

stability of the control algorithm is established assuming that the nonlinear element can be exactly represented by a linear spline function for a given set of breakpoints.

Patwardhan *et al.* [134] used a Wiener model-based MPC method to control the pH level in an acid-base neutralization process. The method proposed by Patwardhan *et al.* uses a MIMO Wiener model with two inputs, the acid flow rate and the base flow rate, and two outputs, i.e., the level and pH. The Wiener system was identified with the partial least squares method. The performance of three model predictive controllers, i.e., linear, Hammerstein model-based, and Wiener model-based ones, was compared. Both the Hammerstein and Wiener model-based MPC schemes outperform the linear one. The MPC scheme is also able to meet several set-point changes into nonlinear regions that the other controllers cannot handle.

A control strategy that linearizes nonlinearity using the inverse nonlinear element model was applied by Sung [154] to control a simulated pH neutralization process in the CSTR. The identification of the pH neutralization process is performed by a simple relay-based feedback method that separates the identification of the nonlinear element from the identification of the linear dynamic system. As a result, a nonparametric nonlinear element model, in the form of a look-up table, is obtained and the ultimate gain, i.e., the gain at which the system controlled by the proportional controller oscillates, and the oscillation frequency are determined.

A continuous stirred-tank fermentation process that exhibits both nonlinear and nonstationary features was studied by Roux *et al.* [143]. In this application, the parameters of the Hammerstein model are estimated using the RLS algorithm. Based on three different models, i.e., linear, nonlinear, and Hammerstein ones, adaptive prediction control algorithms are derived and tested. Both one-step-ahead and multi-step-ahead cost functions are applied.

The application of Wiener models in the control of distillation columns was studied in [19, 20, 110, 114, 129, 138, 146, 159]. Bloemen *et al.* [19, 20] used the Wiener model in MPC algorithms based on polytopic descriptions and an inverse of nonlinearity and compared the obtained results with two other algorithms based on a linear state space model and an FIR model. The identified system is a computer simulator of a moderate-high purity distillation column. The Wiener system is identified via an indirect closed loop algorithm based on the subspace identification methods [113, 157, 164].
A pilot binary distillation column with 26 trays was identified by Ling and Rivera [110]. They used a pulse transfer model of the linear system, a polynomial model of the nonlinear element, and a white noise input signal uniformly distributed in $[-0.5, 0.5]$. Four different models, i.e., second order Volterra series, Wiener, Hammerstein, and NARX models were identified. Their performance was evaluated based on their steady-state responses and step responses at different amplitudes. The obtained results can be summarized as follows:

- The NARX model provides the most accurate approximation of column dynamics.
- The second order Volterra series model can capture correctly column dynamics only in a quite limited operation range.
- The Wiener and Hammerstein models can approximate well column dynamics if the input design is selected properly.

Luyben and Eskinat [114] used a continuous-time Hammerstein model for modelling a 20-tray binary distillation column. The identification of the column is performed via consecutive use of two or more relay-feedback tests with different relay heights and different known dynamic elements inserted into the loop. The Hammerstein model was also applied by Luyben and Eskinat to the identification of a steam-water heat exchanger. Exciting the system input with the PRBS signal, the parameters of the Hammerstein model are estimated using the Narendra-Gallman algorithm [120], see Section 5.2.
Considering the identification of nonlinear systems with block-oriented models, Pearson and Pottman [138] illustrated it with the application to a simulated distillation column example. Under the assumption that the nonlinear steady-state characteristics are known a priori, they determined both the Wiener and Hammerstein models and compared their performance with linear and feedback block-oriented models.
A model predictive control of a C2-splitter based on a MIMO Wiener model was considered by Norquay *et al.* [129]. The C2-splitter is a high purity 114 tray distillation column, which separates a feed stream consisting primarily of ethylene and ethane with a small amount of methane into an ethylene product and ethane waste stream. The Wiener model employed in this application is constructed using cubic splines for nonlinear elements and the first order plus dead time models for linear elements. While the reboiler duty (in terms of the flow of the heating stream) and the reflux flow rate are model inputs, the top composition and the bottom temperature of the column are model outputs.

To identify the static nonlinear element, steady-state responses to changes of the process input about a nominal operating point were recorded. Based on the steady state data, a piecewise polynomial model was constructed. To gain some understanding of system dynamics, step tests were performed, and the first order plus dead time models fitted.
Before industrial implementation, the control strategy was tested with a simulated C2-splitter. The applied predictive control algorithm of the IMC type appears successful in the rejection of major disturbances. Comparisons with linear IMC have shown the Wiener model-based approach to be superior.
Sentoni *et al.* [146] used a Wiener model consisting of a set of discrete-time Laguerre systems and a multilayer perceptron model of the nonlinear element to model a simulated binary distillation column. The application of the model was illustrated with a nonlinear model predictive control example.
The chromatographic separation process was modelled with a MIMO neural network Wiener model by Visala *et al.* [159]. The separation unit consists of two interconnected columns and is used for the separation of different compounds from the incoming solution. Separate input-output models are identified for both columns. The model is used for process monitoring. In this way, changes in dynamics and drifts into undesired disturbed states can be observed by on-line simulation. To include changes in process dynamics, the model can be retrained with recent production data.
A two-step identification procedure was used by Cervantes *et al.* to identify the CSTR and the polymerization reactor. First, the linear dynamic system is identified using the correlation method, then the nonlinear element is approximated with a piecewise linear function. On the basis of the obtained Wiener models, an MPC control strategy is developed and tested.
Another two-step identification procedure was used by Rollins and Bhandari [142] for the identification of MIMO Wiener and Hammerstein systems. In the first step, the nonlinear element is identified from the ultimate response data of sequential step tests. The parameters of the linear dynamic system are estimated in the second step under the constraint of the fitted nonlinear element model. A simulation study, based on simulated CSTR, showed higher prediction accuracy of the proposed method in comparison with that of the unconstrained identification procedure.
Nonlinear structure identification of chemical processes based on a nonlinear input-output operator was studied by Menold *et al.* [118]. They introduced a deterministic suitability measure that qualifies the capability of a model class to capture the input-output behavior of a nonlinear system. The suitability measure can be used to select the model structure prior to the actual parameter identification. A strategy for computing the suitability measures with respect to the Hammerstein, the linear, the Wiener, and the diagonal Volterra model, was evaluated with the example of simulated CSTR.
Davide *et al.* [32] applied the Wiener model to the modelling of quartz microbalance polymer-coated sensors used for the detection of mixtures of n-octane and toulene. The Wiener model is composed of two linear dynamic models

followed by a single nonlinear element. Exciting system inputs with two independent white Gaussian signals, linear dynamic systems are identified using the correlation method. Having the linear dynamic systems identified, a polynomial model of the nonlinear element can be fitted.

A radial basis function neural network was applied as a model of a hydraulic pilot plant by Knohl *et al.* [99]. Due to the linear-in-parameters form of the model, the RLS algorithm with a constant trace can be used for parameter estimation. Based on the developed neural network model, an indirect adaptive control strategy was derived and tested.

An inverse nonlinear element model was applied by Knohl and Unbehauen [98] for the compensation of input nonlinearity in an electro-hydraulic servo system. The system is controlled by the standard indirect adaptive control method. In this application, both the nonlinear element and its inverse are modelled with the RBF neural network. As the Hammerstein model of the system which contains a dead zone nonlinearity can be written in the linear-in-parameters form, the RLS algorithm is used for parameter estimation.

A Wiener model describing a pneumatic valve for the control of fluid flow was used by Wigren [165]. In this example, a linear dynamic model is used to describe the dynamic balance between a control signal, a counteractive spring force and friction. A flow through the valve, being the model output, is a nonlinear function of the valve stem position.

Ikonen and Najim [72] modelled the valve pressure difference in a pump-valve pilot system with a MISO Wiener model. The inputs to the model are the pump command signal, the true valve position, and the consistency of the pulp, whereas the pressure difference over the valve is the model output.

Celka *et al.* [25] considered modelling an electrooptical dynamic system running in a chaotic mode. They showed that, under certain assumptions, such a system can be represented by a Wiener model.

An adaptive precompensation of nonlinear distortion in Wiener systems via a precompensator of a Hammerstein model structure was studied by Kang *et al.* [97]. The examined system consists of an adaptive Wiener model-type estimator and an adaptive precompensator that linearizes the overall system. It is assumed that the nonlinear element is modelled by a polynomial. The parameters of the compensator, being ideally the inverse of the Wiener system, are estimated minimizing the mean square error defined as a difference between the delayed input signal and the system output. An adaptive algorithm for updating the parameters of the precompensator uses the stochastic gradient method. The proposed technique can be applied to the compensation of nonlinear distortions in electronic devices or electromechanical components, e.g., high power amplifiers in satellite communication channels, distortion reduction of loudspeakers, active noise cancellation.

The charging process in diesel engines, considered by Aoubi [6], also reveals nonlinear input-output behavior and a strong dependence of its dynamics on the operating point. The SISO generalized Hammerstein model is defined as

$$\hat{A}(q^{-1})\hat{y}(n) = b_0 + \sum_{k=1}^{r} \hat{B}_k(q^{-1})u^k(n). \tag{6.1}$$

Based on real experimental data, two-input one-output generalized Hammerstein model composed of a third order polynomial model of the nonlinear element and a second order linear system is derived. The inputs to the model are the engine speed and the fuel mass, the output is the loading pressure. The model contains 48 adjustable parameters, which are calculated with the least squares technique.

Modelling continuous-time and discrete-time chaotic systems, including the typical Duffing, Henon, and Lozi systems, is another interesting application of Wiener models proposed by Chen *et al.* [28]. In this application, a single hidden layer neural network model of the nonlinear element, trained with the steepest descent algorithm, acts as a neural network controller of the linear part of the Wiener model. The identification of discrete-time chaotic systems using both Wiener and Hammerstein neural network models was also studied by Xu *et al.* [168].

Different control strategies for Hammerstein systems have been considered by several authors, see [13] for adaptive predictive control based on the polynomial Hammerstein model, and [124, 125] for minimum-time dead-beat controllers. The application of the MIMO Hammerstein model to an iron oxide pellet cooling process used in pyrometallurgical installations was studied by Pomerleau *et al.* [140]. For the cooling zone of an induration furnace, they used a two-input two-output Hammerstein model with decoupled models of nonlinear elements and a coupled model of the linear dynamic system. Based on the empirical Hammerstein model, a linear MPC strategy was developed and compared with a phenomenological nonlinear MPC.

The estimation of parameter changes in Wiener and Hammerstein systems in the context of fault detection and isolation is considered in [78, 79, 91, 92], see Section 6.2 for more details. The identification of steam pressure dynamics in a five stage sugar evaporator is studied in [82, 91], more details are given in Section 6.3.

A theoretical model of the muscle relaxation process that expresses the interaction between the drug dose and the relaxation effect can be described as a continuous-time Wiener model [33]. This Wiener model consists of a third order linear dynamic system plus dead time followed by static nonlinearity. The dead time is introduced to model the transport delay of the drug.

Quaglini *et al.* [141] used a Wiener model, composed of the ARX model and a polynomial model of the nonlinear element, for the relaxation function of soft biological tissues. The identification is made by iterative minimization of two different cost functions that are linear in the parameters of the linear dynamic model and the nonlinear element model, respectively.

Wiener and Hammerstein models have also been applied to the modelling of some other biological systems. A brief survey of such applications is given by Hunter and Korenberg [71].

6.2 Fault detection and isolation with Wiener and Hammerstein models

Model-based fault detection and isolation (FDI) methods require both a nominal model of a system, i.e., a model of the system in its normal operation conditions, and models of the faulty system to be available [38, 135, 30]. A nominal model of the system is used in a fault detection step to generate residuals, defined as a difference between output signals of the system and its model. An analysis of these residuals gives an answer to the question whether any fault occurs or not. If a fault occurs, a fault isolation step is performed in a similar way analyzing residual sequences generated with the models of the faulty system – see Fig. 6.2

In the case of complex industrial systems, e.g., a five-stage sugar evaporation station, the above procedure can be even more useful if it is applied not only to the overall system but also to its chosen sub-modules. Designing an FDI system with such an approach may result in both high fault detection sensitivity and high fault isolation reliability.

The problem considered here can be stated as follows: Given system input and output sequences and knowing the nominal models of Wiener or Hammerstein systems, generate a sequence of residuals and process this sequence to detect and isolate all changes of system parameters caused by any system fault. Both abrupt (step-like) and incipient (slowly developing) faults are to be considered as well.

Assume that the nominal models of a Wiener or Hammerstein system defined by the nonlinear function $f(\cdot)$ and the polynomials $A(q^{-1})$ and $B(q^{-1})$ are known. These models describe the systems in their normal operating conditions with no malfunctions (faults). Moreover, assume that at the time k a step-like fault occurred, and caused a change in the mathematical model of the system. This change can be expressed in terms of additive components of pulse transfer function polynomials. The polynomials $\mathcal{A}(q^{-1})$ and $\mathcal{B}(q^{-1})$ of the pulse transfer function $\mathcal{B}(q^{-1})/\mathcal{A}(q^{-1})$ of the faulty system have the form:

$$\mathcal{A}(q^{-1}) = A(q^{-1}) + \Delta A(q^{-1}), \tag{6.2}$$

$$\mathcal{B}(q^{-1}) = B(q^{-1}) + \Delta B(q^{-1}), \tag{6.3}$$

where

$$A(q^{-1}) = a_1 q^{-1} + \cdots + a_{na} q^{-na}, \tag{6.4}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{nb} q^{-nb}, \tag{6.5}$$

$$\Delta A(q^{-1}) = \alpha_1 q^{-1} + \cdots + \alpha_{na} q^{-na}, \tag{6.6}$$

$$\Delta B(q^{-1}) = \beta_1 q^{-1} + \cdots + \beta_{nb} q^{-nb}. \tag{6.7}$$

The characteristic of the static nonlinear element $g(\cdot)$ in the faulty state can be expressed as a sum of $f(\cdot)$ and its change $\Delta f(\cdot)$:

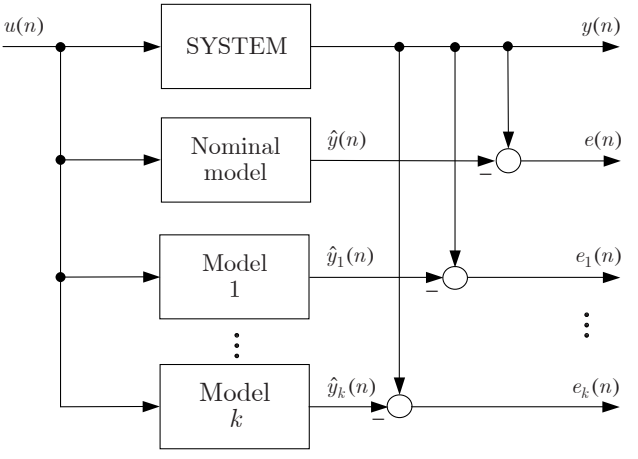

Fig. 6.2. FDI with the nominal model and a bank of models of the faulty system

$$g\big(u(n)\big) = f\big(u(n)\big) + \Delta f\big(u(n)\big), \tag{6.8}$$

where

$$\Delta f\big(u(n)\big) = \mu_0 + \mu_2 u^2(n) + \mu_3 u^3(n) + \cdots . \tag{6.9}$$

For Wiener systems, it is assumed that both $f(\cdot)$ and $g(\cdot)$ are invertible. The inverse nonlinear function of the faulty system can be written as a sum of the inverse function $f^{-1}(\cdot)$ and its change $\Delta f^{-1}(\cdot)$:

$$g^{-1}\big(y(n)\big) = f^{-1}\big(y(n)\big) + \Delta f^{-1}\big(y(n)\big), \tag{6.10}$$

where

$$\Delta f^{-1}\big(y(n)\big) = \gamma_0 + \gamma_2 y^2(n) + \gamma_3 y^3(n) + \cdots . \tag{6.11}$$

Note that $\Delta f^{-1}(\cdot)$ here does not denote the inverse of $\Delta f(\cdot)$ but only a change in the inverse nonlinear characteristic.

6.2.1 Definitions of residuals

The residual $e(n)$ is defined as a difference between the output of the system $y(n)$ and the output of its nominal model $\hat{y}(n)$:

$$e(n) = y(n) - \hat{y}(n). \tag{6.12}$$

Although both series-parallel and parallel Wiener and Hammerstein models can be employed as nominal models for residual generation, we will show that the major advantage of serial-parallel ones is their property of transformation into the linear-in-parameters form. With the nominal series-parallel Hammerstein model (Fig. 6.3) defined by the following expression:

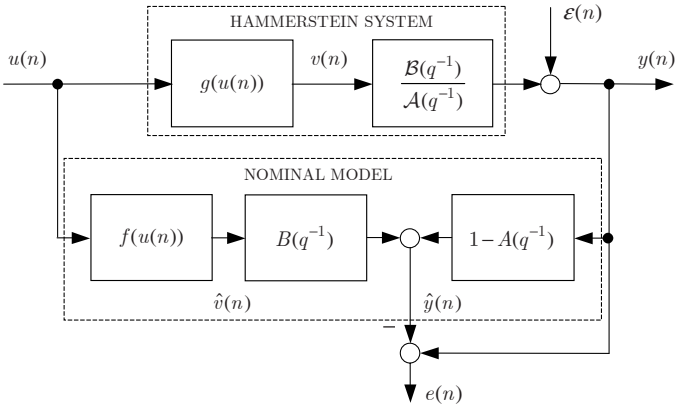

Fig. 6.3. Generation of residuals using the series-parallel Hammerstein model

$$\hat{y}(n) = \big[1 - A(q^{-1})\big]y(n) + B(q^{-1})f\big(u(n)\big), \tag{6.13}$$

we have

$$e(n) = A(q^{-1})y(n) - B(q^{-1})f\big(u(n)\big). \tag{6.14}$$

The output of the faulty Hammerstein system is

$$y(n) = \frac{\mathcal{B}(q^{-1})}{\mathcal{A}(q^{-1})}g\big(u(n)\big) + \varepsilon(n). \tag{6.15}$$

Thus, taking into account (6.2), (6.3) and (6.8), (6.15) can be written in the following form:

$$\begin{aligned} y(n) = {} & \big[1 - A(q^{-1}) - \Delta A(q^{-1})\big]y(n) + \big[B(q^{-1}) + \Delta B(q^{-1})\big] \\ & \big[f\big(u(n)\big) + \Delta f\big(u(n)\big)\big] + \mathcal{A}(q^{-1})\varepsilon(n). \end{aligned} \tag{6.16}$$

Next, assume that the disturbance $\varepsilon(n)$ is

$$\varepsilon(n) = \frac{\epsilon(n)}{\mathcal{A}(q^{-1})}, \tag{6.17}$$

where $\epsilon(n)$ is a zero-mean white noise. Now, the substitution of (6.17) into (6.16), and (6.13) and (6.16) into (6.12) results in a residual equation expressed in terms of changes of the polynomials $A(q^{-1})$, $B(q^{-1})$ and the function $f\big(u(n)\big)$:

$$\begin{aligned} e(n) = {} & -\Delta A(q^{-1})y(n) + \Delta B(q^{-1})f\big(u(n)\big) + \big[B(q^{-1}) \\ & + \Delta B(q^{-1})\big]\Delta f\big(u(n)\big) + \epsilon(n). \end{aligned} \tag{6.18}$$

Similar deliberations for the parallel Hammerstein model (Fig. 6.4) give

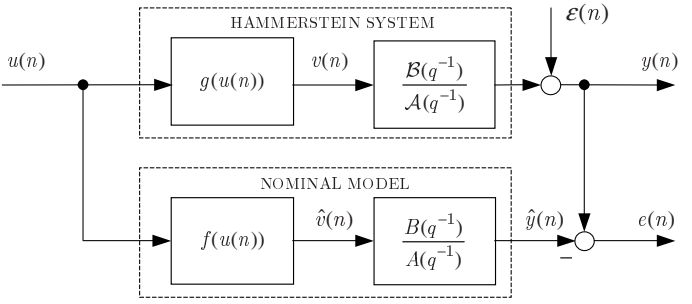

Fig. 6.4. Generation of residuals using the parallel Hammerstein model

$$\begin{aligned} e(n) = {} & \frac{A(q^{-1})\Delta B(q^{-1}) - B(q^{-1})\Delta A(q^{-1})}{A(q^{-1})\big[A(q^{-1}) + \Delta A(q^{-1})\big]} f\big(u(n)\big) \\ & + \frac{B(q^{-1}) + \Delta B(q^{-1})}{A(q^{-1}) + \Delta A(q^{-1})} \Delta f\big(u(n)\big) + \varepsilon(n). \end{aligned} \tag{6.19}$$

In the case of Wiener systems, residual generation with the series-parallel Wiener model is more complicated as both the model of the nonlinear element and the model of the inverse nonlinear element are used (Fig. 6.5). A simpler form of the series-parallel model can be obtained if we assume that the function $f(\cdot)$ is invertible and introduce the following modified definition of the residual [78]:

$$e(n) = f^{-1}\big(y(n)\big) - f^{-1}\big(\hat{y}(n)\big). \tag{6.20}$$

The nominal parallel Wiener model can be expressed as

$$\hat{y}(n) = f\left(\frac{B(q^{-1})}{A(q^{-1})} u(n)\right). \tag{6.21}$$

The output of the faulty Wiener system is

$$y(n) = g\left[\frac{\mathcal{B}(q^{-1})}{\mathcal{A}(q^{-1})} g\big(u(n)\big) + \varepsilon(n)\right]. \tag{6.22}$$

From (6.21), it follows that the output of the linear dynamic model is

$$f^{-1}\big(\hat{y}(n)\big) = \frac{B(q^{-1})}{A(q^{-1})} u(n). \tag{6.23}$$

Equation (6.23) can be written in the form

$$f^{-1}\big(\hat{y}(n)\big) = \big[1 - A(q^{-1})\big] f^{-1}\big(\hat{y}(n)\big) + B(q^{-1})u(n). \tag{6.24}$$

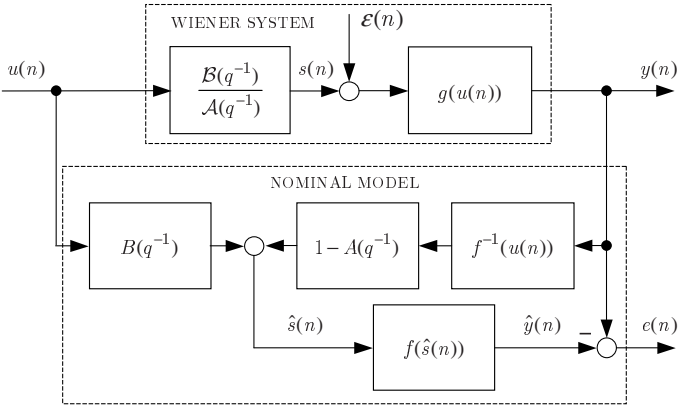

Fig. 6.5. Generation of residuals using the series-parallel Wiener model

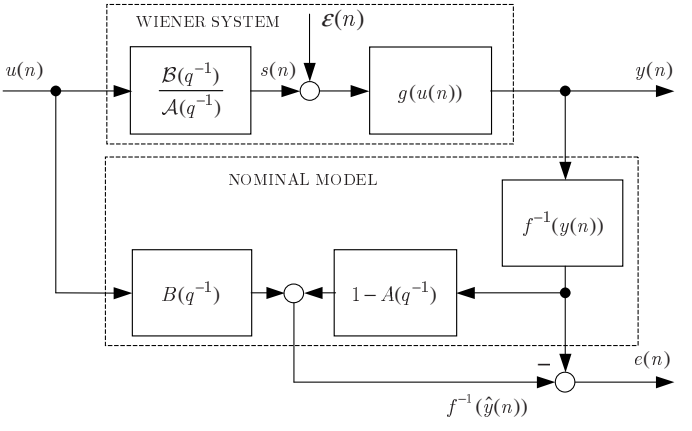

Fig. 6.6. Generation of residuals using the modified series-parallel Wiener model.

Therefore, the modified series-parallel model can be defined as follows:

$$f^{-1}\big(\hat{y}(n)\big) = \big[1 - A(q^{-1})\big] f^{-1}\big(y(n)\big) + B(q^{-1})u(n). \tag{6.25}$$

The generation of residuals with the modified series-parallel Wiener model is illustrated in Fig. 6.6. Taking into account (6.2), (6.3) and (6.10), it follows from (6.22) that the signal $f^{-1}\big(y(n)\big)$ for the faulty Wiener system can be expressed as

$$f^{-1}\big(y(n)\big) = \big[1 - A(q^{-1}) - \Delta A(q^{-1})\big]\big[f^{-1}\big(y(n)\big) + \Delta f^{-1}\big(y(n)\big)\big] + \big[B(q^{-1}) + \Delta B(q^{-1})\big]u(n) - \Delta f^{-1}\big(y(n)\big) + \mathcal{A}(q^{-1})\varepsilon(n). \tag{6.26}$$

Hence, assuming that the disturbance $\varepsilon(n)$ is given by (6.17), the residual (6.20) is

$$e(n) = -\Delta A(q^{-1})f^{-1}\big(y(n)\big) + \Delta B(q^{-1})u(n) - \big[A(q^{-1}) + \Delta A(q^{-1})\big]\Delta f^{-1}\big(y(n)\big) + \epsilon(n). \tag{6.27}$$

A major advantage of both the series-parallel Hammerstein model and the modified series-parallel Wiener model is that their residual equations (6.18) and (6.27), after a proper change of parameterization, can be written in the linear-in-parameters form. Then these redefined parameters can be estimated with linear regression methods.

6.2.2 Hammerstein system. Parameter estimation of the residual equation

Assume that the function $\Delta f(\cdot)$ describing a change of the steady-state characteristic $f(\cdot)$ caused by a fault has the form of a polynomial of the order r:

$$\Delta f\big(u(n)\big) = \mu_0 + \mu_2 u^2(n) + \cdots + \mu_r u^r(n). \tag{6.28}$$

Thus, the residual equation (6.18) can be written in the following linear-in-parameters form:

$$e(n) = -\sum_{j=1}^{na} \alpha_j y(n-j) + \sum_{j=1}^{nb} \beta_j f\big(u(n-j)\big) + d_0 + \sum_{k=2}^{r}\sum_{j=1}^{nb} d_{kj} u^k(n-j) + \epsilon(n), \tag{6.29}$$

where

$$d_0 = \mu_0 \sum_{j=1}^{nb} (b_j + \beta_j), \tag{6.30}$$

$$d_{kj} = \mu_k (b_j + \beta_j). \tag{6.31}$$

Note that (6.29) has $M = na + nb + nb(r-1) + 1$ unknown parameters α_j, β_j, d_{kj} and d_0. In a disturbance-free case ($\epsilon(n) = 0$), these parameters can be calculated performing $N = M$ measurements of the input and output signals and solving the following set of linear equations:

$$e(n) = -\sum_{j=1}^{na} \alpha_j y(n-j) + \sum_{j=1}^{nb} \beta_j f\big(u(n-j)\big) + d_0 + \sum_{k=2}^{r}\sum_{j=1}^{nb} d_{kj} u^k(n-j), \quad n = 1, 2, \ldots, M. \tag{6.32}$$

In practice, it is more realistic to assume that the system output is disturbed by the disturbance (6.17). Now, performing $N \gg M$ measurements of the input and output signals, the parameters of the residual equation (6.29) can be estimated using the least squares method with the parameter vector $\hat{\boldsymbol{\theta}}(n)$ and the regression vector $\mathbf{x}(n)$ defined as follows:

$$\begin{aligned}\hat{\boldsymbol{\theta}}(n) &= \left[\hat{\alpha}_1 \ \ldots \ \hat{\alpha}_{na} \ \hat{\beta}_1 \ \ldots \ \hat{\beta}_{nb} \ \hat{d}_0 \ \hat{d}_{21} \ldots \hat{d}_{2nb} \ldots \hat{d}_{r1} \ldots \hat{d}_{rnb}\right]^T, \\ \mathbf{x}(n) &= \big[-y(n-1) \ldots -y(n-na) \ f\big(u(n-1)\big) \ldots f\big(u(n-nb)\big) \\ &\qquad 1 \ u^2(n-1) \ldots u^2(n-nb) \ldots u^r(n-1) \ldots u^r(n-nb)\big]^T.\end{aligned} \tag{6.33}$$

The vector $\hat{\boldsymbol{\theta}}(n)$ can be calculated on-line using the recursive least squares algorithm:

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \boldsymbol{P}(n)\mathbf{x}(n)\big(e(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}(n-1)\big), \tag{6.34}$$

$$\boldsymbol{P}(n) = \boldsymbol{P}(n-1) - \frac{\boldsymbol{P}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\boldsymbol{P}(n-1)}{1 + \mathbf{x}^T(n)\boldsymbol{P}(n-1)\mathbf{x}(n)}, \tag{6.35}$$

with $\hat{\boldsymbol{\theta}}(0) = \mathbf{0}$ i $\mathbf{P}(0) = \alpha\mathbf{I}$, where $\alpha \gg 1$ and $\mathbf{I}$ is an identity matrix. Parameter estimation of the residual equation (6.29) often results in asymptotically biased estimates. That is the case if system additive output disturbances are not given by (6.17) but have the property of zero-mean white noise, i.e., $\varepsilon(n) = \epsilon(n)$. In this case, the residual equation (6.18) has the form

$$\begin{aligned}e(n) = &-\Delta A(q^{-1})y(n) + \Delta B(q^{-1})f\big(u(n)\big) + \big[B(q^{-1}) \\ &+ \Delta B(q^{-1})\big]\Delta f\big(u(n)\big) + \big[A(q^{-1}) + \Delta A(q^{-1})\big]\epsilon(n).\end{aligned} \tag{6.36}$$

The term $[A(q^{-1}) + \Delta A(q^{-1})]\,\epsilon(n)$ in (6.36) makes the parameter estimates asymptotically biased and the residuals $e(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}(n-1)$ correlated. To obtain unbiased parameter estimates, other known parameter estimation methods, such as the instrumental variables method, the generalized least squares method, or the extended least squares method, can be employed [35, 149]. Using the extended least squares method, the vectors $\hat{\boldsymbol{\theta}}(n)$ i $\mathbf{x}(n)$ are defined as follows:

$$\hat{\boldsymbol{\theta}}(n) = \left[\hat{\alpha}_1 \ \ldots \ \hat{\alpha}_{na} \ \hat{\beta}_1 \ldots \hat{\beta}_{nb} \ \hat{d}_0 \ \hat{d}_{21} \ldots \hat{d}_{2nb} \ldots \hat{d}_{r1} \ldots \hat{d}_{rnb} \ \hat{c}_1 \ldots \hat{c}_{na}\right]^T, \tag{6.37}$$

$$\begin{aligned}\mathbf{x}(n) = &\big[-y(n-1)\ldots-y(n-na) \ f\big(u(n-1)\big)\ldots f\big(u(n-nb)\big) \ 1 \ u^2(n-1) \\ &\ldots u^2(n-nb)\ldots u^r(n-1)\ldots u^r(n-nb) \ \hat{e}(n-1)\ldots\hat{e}(n-na)\big]^T,\end{aligned} \tag{6.38}$$

where $\hat{e}(n)$ denotes the one step ahead prediction error of $e(n)$:

$$\hat{e}(n) = e(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}(n-1). \tag{6.39}$$

Example 6.1 The nominal Hammerstein model composed of the linear dynamic system

$$\frac{B(q^{-1})}{A(q^{-1})} = \frac{0.63467q^{-1} + 0.48069q^{-2}}{1 - 1.3231q^{-1} + 0.4346q^{-2}} \tag{6.40}$$

and the nonlinear element

$$f\big(u(n)\big) = \tanh(u(n)) \tag{6.41}$$

was used in the simulation example. The faulty Hammerstein system is described as

$$\frac{\mathcal{B}(q^{-1})}{\mathcal{A}(q^{-1})} = \frac{0.1752q^{-1} + 0.1534q^{-2}}{1 - 1.6375q^{-1} + 0.6703q^{-2}}, \tag{6.42}$$

$$\begin{aligned} g\big(u(n)\big) = {} & \tanh(u(n)) - 0.25u^2(n) - 0.2u^3(n) + 0.15u^4(n) \\ & - 0.1u^5(n) + 0.05u^6(n) - 0.025u^7(n) + 0.0125u^8(n). \end{aligned} \tag{6.43}$$

The steady-state characteristics of both the nominal model and the faulty Hammerstein system are shown in Fig. 6.7. The system input was excited with a sequence of $N = 25000$ pseudorandom numbers of uniform distribution in the interval $(-1.5,\ 1.5)$. The system output was disturbed by the disturbance $\varepsilon(n)$ defined as

$$\varepsilon(n) = \frac{\epsilon(n)}{\mathcal{A}(q^{-1})} \tag{6.44}$$

or

$$\varepsilon(n) = \epsilon(n), \tag{6.45}$$

where $\epsilon(n)$ are the values of a Gaussian pseudorandom sequence $\mathcal{N}(0, 0.05)$. For the system disturbed by (6.44), the parameters of (6.29) were estimated with the RLS method – the RLS_{I} example. In the case of the disturbance (6.45), the parameter estimation was performed using both the RLS and RELS methods – the RLS_{II} and RELS_{II} examples. The obtained parameter estimates are given in Table 6.1. The identification error of $\Delta f\big(u(n)\big)$ is shown in Fig. 6.8. A comparison of estimation accuracy measured by four different indices defining estimation accuracy of the residuals $e(n)$, the function $\Delta f\big(u(n)\big)$, the parameters $\mu_j, j = 2, \ldots, 8$, and α_j i β_j, $j = 1, 2$, is given in Table 6.2. The highest accuracy is obtained in the RLS_{I} example for the system disturbed by (6.44). For the system disturbed with (6.45), a comparison of parameter estimates and indices confirms asymptotical biasing of parameter estimates obtained in the RLS_{II} example.

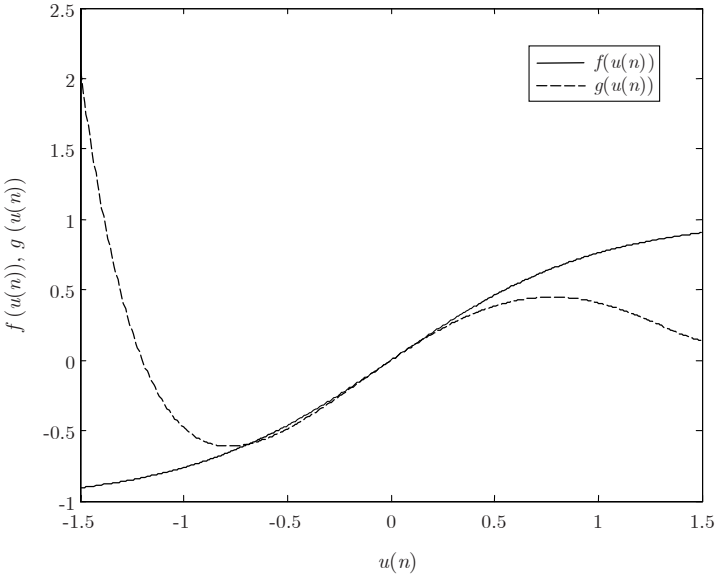

Fig. 6.7. Hammerstein system. Nonlinear functions $f\big(u(n)\big)$ and $g\big(u(n)\big)$

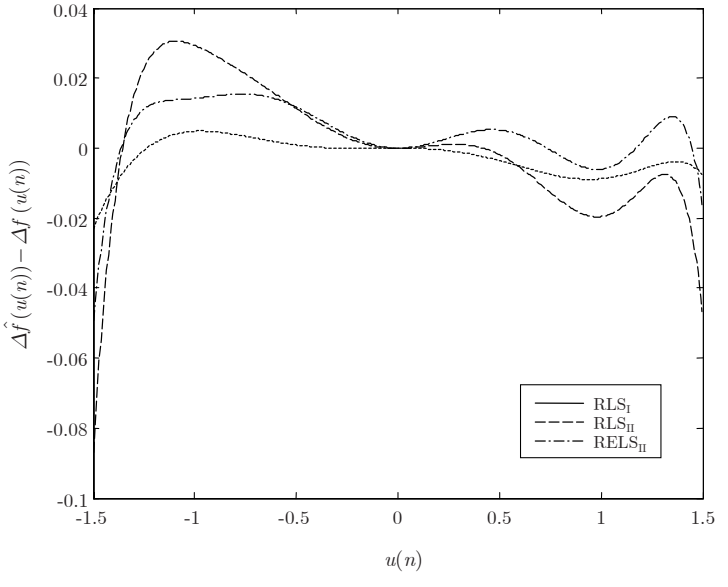

Fig. 6.8. Identification error of the function $\Delta f\big(u(n)\big)$

Table 6.1. Parameter estimates

Parametr	True value	RLS_I $\varepsilon(n) = \frac{\epsilon(n)}{\mathcal{A}(q^{-1})}$	RLS_II $\varepsilon(n) = \epsilon(n)$	RELS_II $\varepsilon(n) = \epsilon(n)$
α_1	-0.3144	-0.3151	-0.1478	-0.3062
α_2	0.2357	0.2369	0.0708	0.2269
β_1	-0.4594	-0.4616	-0.4664	-0.4625
β_2	-0.3273	-0.3305	-0.2972	-0.3278
μ_2	-0.2500	-0.2431	-0.2871	-0.3124
μ_3	-0.2000	-0.1770	-0.1362	-0.1671
μ_4	0.1500	0.1440	0.2431	0.2908
μ_5	-0.1000	-0.1202	-0.1460	-0.1275
μ_6	0.0500	0.0498	-0.0372	-0.0604
μ_7	-0.0250	-0.0210	-0.0183	-0.0202
μ_8	0.0125	0.0138	0.0387	0.0405
c_1	-1.6375			-1.1253
c_2	0.6703			0.1341

Table 6.2. Comparison of estimation accuracy

Index	RLS_I $\varepsilon(n) = \frac{\epsilon(n)}{\mathcal{A}(q^{-1})}$	RLS_II $\varepsilon(n) = \epsilon(n)$	RELS_II $\varepsilon(n) = \epsilon(n)$
$\frac{1}{N}\sum_{i=1}^{N}\left[e(n)-\hat{e}(n)\right]^2$	2.82×10^{-3}	9.62×10^{-3}	4.49×10^{-3}
$\frac{1}{400}\sum_{i=1}^{400}\left[f\big(u(n)\big)-\hat{f}\big(u(n)\big)\right]^2$	3.69×10^{-5}	8.77×10^{-4}	1.37×10^{-4}
$\frac{1}{7}\sum_{j=2}^{8}(\mu_j-\hat{\mu}_j)^2$	3.08×10^{-5}	3.45×10^{-4}	9.87×10^{-5}
$\frac{1}{4}\sum_{j=1}^{2}\left[(a_j-\hat{a}_j)^2+(b_j-\hat{b}_j)^2\right]$	4.07×10^{-6}	1.40×10^{-2}	3.88×10^{-5}

6.2.3 Wiener system. Parameter estimation of the residual equation

Assume that the function $\Delta f^{-1}(\cdot)$ describing a change in the inverse steady-state characteristic caused by a fault has the form of a polynomial of the order r:

$$\Delta f^{-1}\big(y(n)\big) = \gamma_0 + \gamma_2 y^2(n) + \cdots + \gamma_r y^r(n). \tag{6.46}$$

As in the case of the Hammerstein system, the residual equation (6.27) can be expressed in the linear-in-parameters form:

$$\begin{aligned} e(n) = &- \sum_{j=1}^{na} \alpha_j f^{-1}\big(y(n-j)\big) + \sum_{j=1}^{nb} \beta_j u(n-j) + d_0 \\ &+ \sum_{k=2}^{r} \sum_{j=0}^{na} d_{kj} y^k(n-j) + \epsilon(n), \end{aligned} \tag{6.47}$$

where

$$d_0 = -\gamma_0 \Big[1 + \sum_{j=1}^{na} (a_j + \alpha_j)\Big], \tag{6.48}$$

$$d_{kj} = \begin{cases} -\gamma_k, & j = 0 \\ -\gamma_k (a_j + \alpha_j), & j = 1, \ldots, na. \end{cases} \tag{6.49}$$

For disturbance-free Wiener systems, the parameters α_j, β_j, d_{kj} and d_0 can be calculated solving a set of $M = na + nb + (na+1)(r-1) + 1$ linear equations. In the stochastic case, parameter estimation with the parameter vector $\hat{\boldsymbol{\theta}}(n)$ and the regression vector $\mathbf{x}(n)$ defined as

$$\hat{\boldsymbol{\theta}}(n) = \big[\hat{\alpha}_1 \ \ldots \ \hat{\alpha}_{na} \ \hat{\beta}_1 \ \ldots \ \hat{\beta}_{nb} \ \hat{d}_0 \ \hat{d}_{20} \ldots \hat{d}_{2na} \ldots \hat{d}_{r0} \ldots \hat{d}_{rna}\big]^T, \tag{6.50}$$

$$\begin{aligned} \mathbf{x}(n) = \big[&-f^{-1}\big(y(n-1)\big) \ldots - f^{-1}\big(y(n-na)\big) \ u(n-1) \ldots u(n-nb) \\ &1 \ y^2(n) \ldots y(n-na)^2 \ldots y^r(n) \ldots y(n-na)^r\big]^T \end{aligned} \tag{6.51}$$

can be performed with the LS or the RLS method.
An alternative to the above procedure can be parameter estimation with the extended least squares (ELS) algorithm. In this case, the parameter vector $\boldsymbol{\theta}(n)$ and the regression vector are defined as

$$\hat{\boldsymbol{\theta}}(n) = \big[\hat{\alpha}_1 \ \ldots \hat{\alpha}_{na} \ \hat{\beta}_1 \ldots \hat{\beta}_{nb} \ \hat{d}_0 \ \hat{d}_{20} \ldots \hat{d}_{2na} \ldots \hat{d}_{r0} \ldots \hat{d}_{rna} \ \hat{c}_1 \ldots \hat{c}_{na}\big]^T, \tag{6.52}$$

$$\begin{aligned} \mathbf{x}(n) = \big[&- f^{-1}\big(y(n-1)\big) \ldots - f^{-1}(y(n-na)\big) \ u(n-1) \ldots u(n-nb) \\ &1 \ y^2(n) \ldots y(n-na)^2 \ldots y^r(n) \ldots y(n-na)^r \ \hat{e}(n-1) \ldots \\ &\hat{e}(n-na)\big]^T, \end{aligned} \tag{6.53}$$

where $\hat{e}(n)$ denotes the one step ahead prediction error of $e(n)$:

$$\hat{e}(n) = e(n) - \mathbf{x}^T(n)\hat{\boldsymbol{\theta}}(n-1). \tag{6.54}$$

The parameters obtained with the LS method or the ELS method are asymptotically biased. This comes from the well-known property of the least squares

method stating that to obtain consistent parameter estimates, the regression vector $\mathbf{x}(n)$ should be uncorrelated with the system disturbance $\epsilon(n)$, i.e., $E[\,\mathbf{x}(n)\,\epsilon(n)\,] = \mathbf{0}$. Obviously, this condition is not fulfilled for the model (6.47) as the powered system outputs $y^2(n), \ldots, y^r(n)$ depend on $\epsilon(n)$. A common way to obtain consistent parameter estimates in such cases is the application of the instrumental variable method (IV) or its recursive version (RIV). Instrumental variables should be chosen to be correlated with regressors and uncorrelated with the system disturbance $\epsilon(n)$. Although different choices of instrumental variables can be made, replacing $y^2(n), \ldots, y^r(n-na)$ with their approximated values obtained by filtering $u(n)$ through the nominal model is a good choice. This leads to the following estimation procedure:

1. Simulate the nominal model (6.21) to obtain $\hat{y}(n)$.
2. Estimate the parameters of (6.47) using the IV or the RIV method with the instrumental variables

$$\begin{aligned}\mathbf{z}(n) = \big[&- f^{-1}\big(y(n-1)\big) \ldots - f^{-1}\big(y(n-na)\big)\ u(n-1) \ldots u(n-nb) \\ &1\ \hat{y}^2(n) \ldots \hat{y}^2(n-na) \ldots \hat{y}^r(n) \ldots \hat{y}^r(n-na)\big]^T.\end{aligned}$$

Example 6.2. The nominal Wiener model used in the simulation example consists of the linear dynamic system

$$\frac{B(q^{-1})}{A(q^{-1})} = \frac{0.5q^{-1} - 0.3q^{-2}}{1 - 1.5q^{-1} + 0.7q^{-2}} \tag{6.55}$$

and the nonlinear element of the following inverse nonlinear characteristic:

$$f^{-1}\big(y(n)\big) = y(n) - \frac{1}{6}y^3(n). \tag{6.56}$$

The faulty Wiener system is defined by the pulse transfer function

$$\frac{\mathcal{B}(q^{-1})}{\mathcal{A}(q^{-1})} = \frac{0.3q^{-1} - 0.2q^{-2}}{1 - 1.75q^{-1} + 0.85q^{-2}} \tag{6.57}$$

and the inverse nonlinear characteristic (Fig. 6.9) of the form

$$g^{-1}\big(y(n)\big) = \tan\big(y(n)\big). \tag{6.58}$$

The system was excited with a sequence of $N = 50000$ pseudorandom values of uniform distribution in $(-1, 1)$. The system output was disturbed additively with the disturbances (6.45), with $\{\epsilon(n)\}$ – a pseudorandom sequence, uniformly distributed in $(-0.01,\ 0.01)$. Parameter estimation was performed with the RLS and RELS methods for both a disturbance-free case – $\mathrm{RLS_I}$ example, and for the system disturbed by (6.45) – the $\mathrm{RLS_{II}}$ $\mathrm{RELS_{II}}$ examples. The parameter estimates are given in Table 6.4, and Table 6.3 shows a comparison of estimation accuracy. The identification error of the function

$\Delta f^{-1}(\cdot)$ is shown in Fig. 6.10. An assumed finite order of the polynomial (6.46), i.e., $r = 11$ is another source of the identification error as the function $\Delta f^{-1}(\cdot)$ can be represented accurately with the power series. A comparison of the results obtained in the RLS_{II} example and the RELS_{II} example shows the inconsistency of the RLS estimator.

Table 6.3. Comparison of estimation accuracy

Example Index	RLS_{I} $\varepsilon(n) = 0$	RLS_{II} $\varepsilon(n) = \epsilon(n)$	RELS_{II} $\varepsilon(n) = \epsilon(n)$
$\frac{1}{N}\sum_{i=1}^{N}\left[e(n) - \hat{e}(n)\right]^2$	8.07×10^{-6}	1.64×10^{-4}	8.60×10^{-5}
$\frac{1}{400}\sum_{i=1}^{400}\left[\Delta f^{-1}\big(\hat{y}(n)\big) - \Delta\hat{f}^{-1}\big(\hat{y}(n)\big)\right]^2$	1.39×10^{-6}	1.29×10^{-4}	3.68×10^{-5}
$\frac{1}{10}\sum_{j=2}^{11}\left(\gamma_j - \hat{\gamma}_j\right)^2$	1.73×10^{-4}	1.44×10^{-2}	4.31×10^{-3}
$\frac{1}{4}\sum_{j=1}^{2}\left[(a_j - \hat{a}_j)^2 + (b_j - \hat{b}_j)^2\right]$	1.72×10^{-6}	4.09×10^{-4}	5.75×10^{-6}

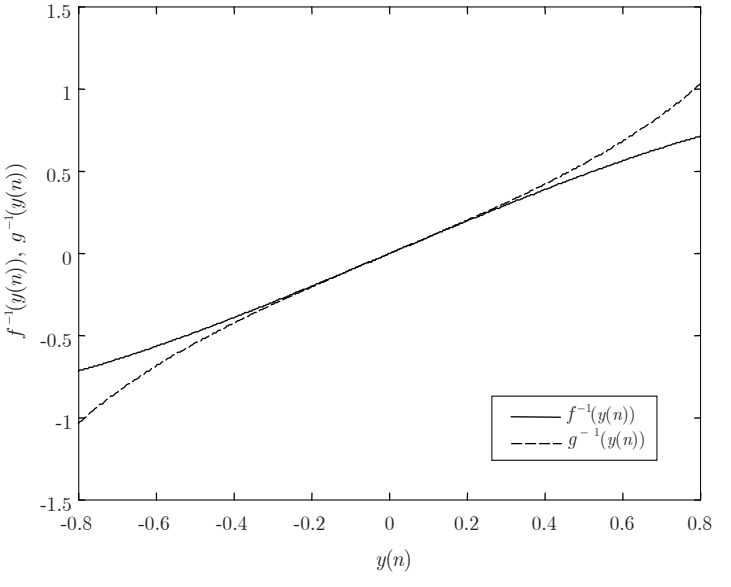

Fig. 6.9. Wiener system. Inverse nonlinear functions $f^{-1}\big(\hat{y}(n)\big)$ and $g^{-1}\big(\hat{y}(n)\big)$

Table 6.4. Parameter estimates

Parameter	True value	RLS_{I} $\varepsilon(n) = 0$	RLS_{II} $\varepsilon(n) = \epsilon(n)$	RELS_{II} $\varepsilon(n) = \epsilon(n)$
α_1	-0.2500	-0.2482	-0.2164	-0.2487
α_2	0.1500	0.1491	0.1293	0.1526
β_1	-0.2000	-0.2006	-0.2060	-0.2032
β_2	0.1000	0.1006	0.1065	0.1021
γ_2	0	-0.0000	-0.0019	-0.0002
γ_3	0.5000	0.4833	0.3445	0.4177
γ_4	0	0.0002	0.0054	-0.0002
γ_5	0.1333	0.1646	0.4240	0.2895
γ_6	0	-0.0004	-0.0057	-0.0011
γ_7	0.0540	0.0372	-0.0835	-0.0277
γ_8	0	-0.0004	-0.0053	-0.0029
γ_9	0.0219	0.0136	-0.0846	-0.0346
γ_{10}	0	0.0010	0.0084	0.0068
γ_{11}	0.0089	0.0199	0.0753	0.0539
c_1	-1.7500			-0.9514
c_2	0.8500			-0.0061

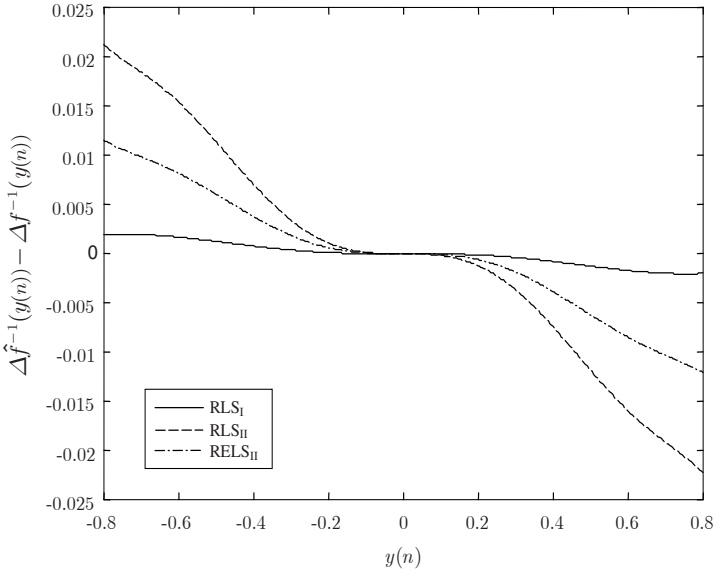

Fig. 6.10. Identification error of the inverse nonlinear function $\Delta f^{-1}\big(u(n)\big)$

6.3 Sugar evaporator. Identification of the nominal model of steam pressure dynamics

The main task of a multiple-effect sugar evaporator is thickening the thin sugar juice from sugar density of approximately 14 to 65–70 Brix units (Bx). Other important tasks are steam generation, waste steam condensation, and supplying waste-heat boilers with water condensate. In a multiple-effect evaporation process, the sugar juice and the saturated steam are fed to the successive stages of the evaporator at a gradually decreasing pressure. Many complex physical phenomena and chemical reactions occur during the thickening of the sugar juice, including sucrose decomposition, the precipitation of calcium compounds, the decomposition of acid amides, etc. A flow of the steam and sugar juice through the successive stages of the evaporator results in a close relationship between temperatures and pressures of the juice steam in these stages. Moreover, juice steam temperature and juice steam pressure depend also on other physical quantities such as the rate of the juice flow or juice temperature.

6.3.1 Theoretical model

A theoretical approach to process modelling relies upon deriving a mathematical model of the process by applying the laws of physics. Theoretical models obtained in this way are often too complicated to be used, for example, in control applications. Despite this, theoretical models are a source of valuable information on the nature of the process. This information can be very useful for the identification of experimental mathematical models based on process the input-output data. Moreover, theoretical models can serve as benchmarks for the evaluation and verification of experimental models.
Modelling the multiple-effect sugar evaporator is complicated by the fact that it consists of a number of evaporators connected both in series and in parallel. All this makes the formulation of a theoretical model a difficult task. Theoretical models are derived based on mass and energy balances for all stages. To perform this, the following assumptions are made [111]:

1. The juice and the steam are in a saturated equilibrium.
2. Both the mass of the steam in the juice chamber and the mass of the steam in the steam chamber are constant.
3. The operation of juice level controllers makes it possible to neglect variations of juice levels.
4. Heat losses to the surrounding environment are negligible.
5. Mixing is perfect.

The model of the dependence of steam pressure in the steam chamber of the stage k on steam pressure in the steam chamber of the stage $k-1$, given by Carlos and Corripio [24], has the form

$$P_k = P_{k-1} - \frac{\gamma_k (O_{k-1})^2}{\rho_k^2 (T_{k-1}, P_{k-1})}, \tag{6.59}$$

$$P_1 = P_0 - \frac{\gamma_1 S^2}{\rho_1^2 (T_0, P_0)}, \tag{6.60}$$

where P_k denotes juice steam pressure in the steam chamber of the stage k, γ_k is the conversion factor, O_k is the steam flow rate from the stage k, ρ_k is the steam density at the stage k, and T_k is steam temperature at the stage k. The models (6.59) and (6.60) describe steady-state behavior of the process. To include dynamic properties of the process, a modified model described with differential equations of the first order was proposed:

$$P_k = \tau_k \frac{dP_{k-1}}{dt} + P_{k-1} - \frac{\gamma_k (O_{k-1})^2}{\rho_k^2 (T_{k-1}, P_{k-1})}, \tag{6.61}$$

$$P_1 = \tau_1 \frac{dP_0}{dt} + P_0 - \frac{\gamma_1 S^2}{\rho_1^2 (T_0, P_0)}, \tag{6.62}$$

where τ_k is the time constant.
Equations (6.59) – (6.62) are a part of the overall model of the sucrose juice concentration process. This model comprises also mass and energy balances for all stages [111].

6.3.2 Experimental models of steam pressure dynamics

In a neural network model (Fig. 6.11), it is assumed that the model output $\hat{P}_2(n)$ is a sum of two components defined by the nonlinear function of the pressure $P_3(n)$ and the linear function of the pressure $P_1(n)$. The model output at the time n is

$$\hat{P}_2(n) = \hat{f}\left[\frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} P_3(n)\right] + \hat{C}(q^{-1}) P_1(n), \tag{6.63}$$

with

$$\hat{A}(q^{-1}) = 1 + \hat{a}_1 q^{-1} + \hat{a}_2 q^{-2}, \tag{6.64}$$

$$\hat{B}(q^{-1}) = \hat{b}_1 q^{-1} + \hat{b}_2 q^{-2}, \tag{6.65}$$

$$\hat{C}(q^{-1}) = \hat{c}_1 q^{-1} + \hat{c}_2 q^{-2}, \tag{6.66}$$

where $\hat{a}_1$, $\hat{a}_2$, $\hat{b}_1$, $\hat{b}_2$, $\hat{c}_1$ i $\hat{c}_2$ denote model parameters. The function $\hat{f}(\cdot)$ is modelled with a multilayer perceptron containing one hidden layer consisting of M nodes of the hyperbolic tangent activation function:

$$\hat{f}(\hat{s}(n)) = \sum_{j=1}^{M} w_{1j}^{(2)} \tanh\left(x_j(n)\right) + w_{10}^{(2)}, \tag{6.67}$$

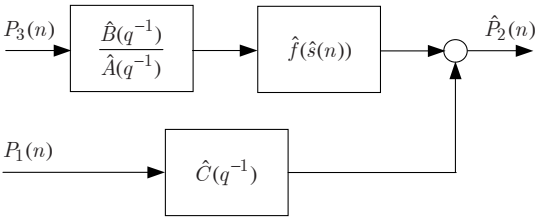

Fig. 6.11. Neural network model of steam pressure dynamics

$$x_j(n) = w_{j1}^{(1)} \hat{s}(n) + w_{j0}^{(1)}, \tag{6.68}$$

where $\hat{s}(n) = \left[\hat{B}(q^{-1})/\hat{A}(q^{-1})\right] P_3(n)$ is the output of a linear dynamic model, and $w_{10}^{(1)}, \ldots, w_{M1}^{(1)}$, $w_{10}^{(2)}, \ldots, w_{1M}^{(2)}$ are the weights of the neural network. The neural network structure contains a Wiener model [76, 75] in the path of the pressure $P_3(n)$ and a linear finite impulse response filter of the second order in the path of the pressure $P_1(n)$. In a linear model of steam pressure dynamics, the Wiener model is replaced with a linear dynamic model:

$$\hat{P}_2(n) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} P_3(n) + \hat{C}(q^{-1}) P_1(n). \tag{6.69}$$

6.3.3 Estimation results

Parameter estimation of the models (6.63) i (6.69) was performed based on a set of 10000 input-output measurements recorded at the sampling rate of 10s. The RLS algorithm was employed for the estimation of the ARX model. The neural network Wiener model of the structure $\mathcal{N}(1-24-1)$ was trained recursively with the backpropagation learning algorithm [81]. For both models, 50000 sequential steps were made, processing the overall data set five times. Then the models were tested with another data set of 8000 input-output measurements.

The results of the estimation and the testing are shown in Figs 6.13 – 6.15, and a comparison of the test results for both models is given in Table 6.5. Lower values of the mean square prediction error of the pressure P_2 obtained for the neural network model for both the training and testing sets confirm the nonlinear nature of the process.

As the analyzed model of steam pressure dynamics is characterized by a low time constant of approximately 20s, fast fault detection is possible. Moreover, as steam pressure is not controlled automatically, there is no problem of closed-loop system identification.

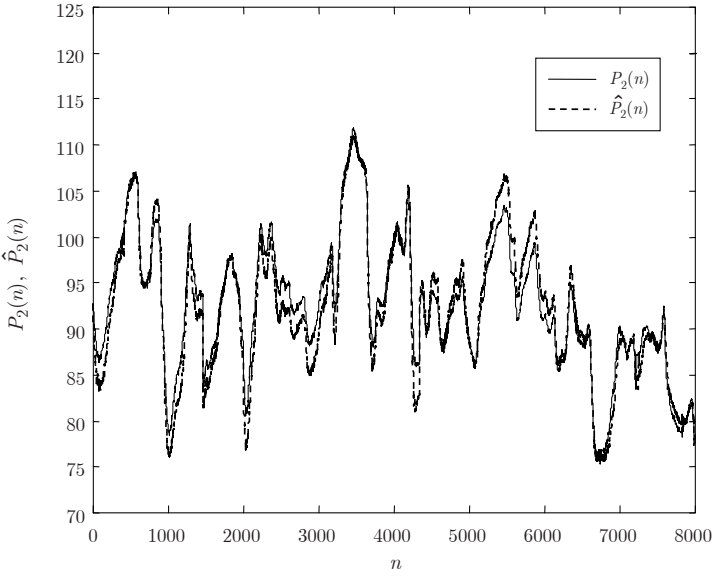

Fig. 6.12. Steam pressure in the steam chamber 2 and the output of the neural network model – the testing set

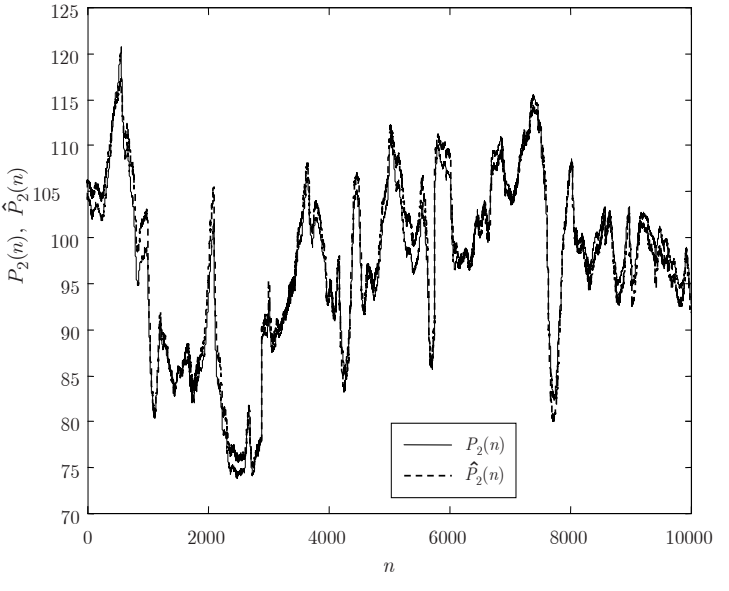

Fig. 6.13. Steam pressure in the steam chamber 2 and the output of the neural network model – the training set

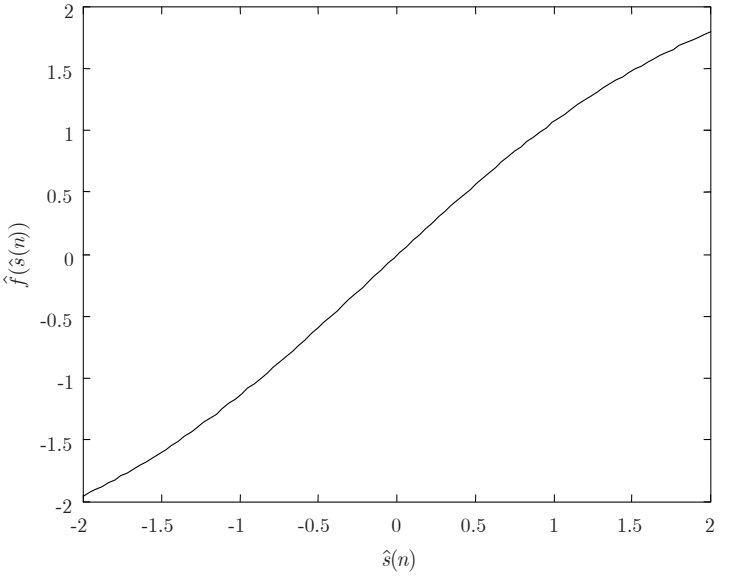

Fig. 6.14. Estimated nonlinear function

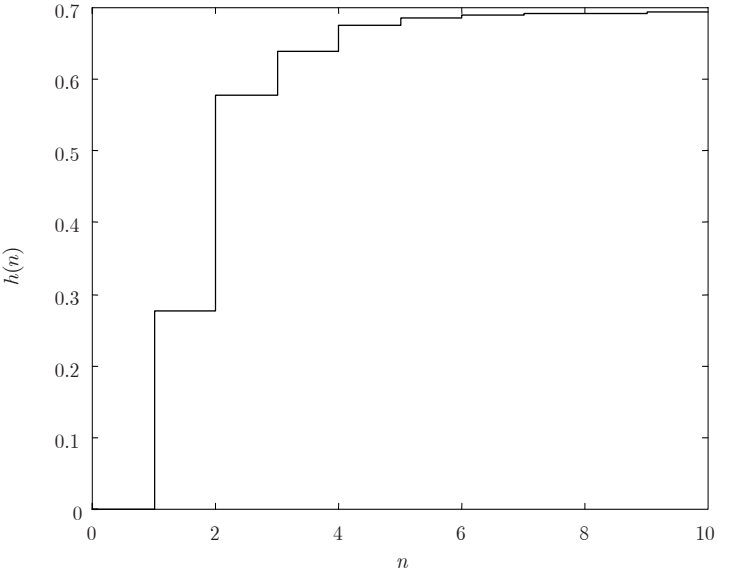

Fig. 6.15. Step response of the linear dynamic model

Table 6.5. Mean square prediction error of P_2 [kPa]

	ARX model	Neural network model
Training set	2.103	1.804
Testing set	2.322	2.084

6.4 Summary

Identification methods of system parameter changes, discussed in this chapter, require a nominal model of the system, the sequences of the system input and output signals and the sequence of residuals to be available. If the system is disturbance free, it is possible to calculate system parameter changes in both Wiener and Hammerstein systems by solving a set of linear equations. In practice, however, it is more realistic to assume that the system output is disturbed by additive disturbances and to estimate system parameter changes using parameter estimation techniques.

In Hammerstein systems with polynomial nonlinearities, system parameter changes can be estimated with the LS method. In this case, consistent parameter estimates are obtained provided that the system output is disturbed additively with (6.17). For other types of output disturbances, the use of the LS method results in inconsistent parameter estimates. Therefore, to obtain consistent parameter estimates, other parameter estimation methods, e.g., the ELS method can be used.

The estimation of system parameter changes in Wiener systems with inverse nonlinear characteristics described by polynomials with the LS method also results in inconsistent parameter estimates. To obtain consistent parameter estimates, an IV estimation procedure has been proposed.

For systems with characteristics described by the power series, the estimation of the parameters of linear dynamic systems and changes of the nonlinear characteristic for Hammerstein systems or the inverse nonlinear characteristic for Wiener systems can be made employing neural network models of the residual equation. Note that neural network models can be also useful in the case when the order of the system polynomial nonlinearity r is high. This comes from the well-known fact that the high order r introduces errors due to noise and model uncertainty, and slows down the convergence rate [9].

Nominal models of systems are identified based on input-output measurements recorded in the normal operation conditions. In spite of the fact that such measurements are often available for many real processes, the determination of nominal models at a high level of accuracy is not an easy task, as has been shown in the sugar evaporator example. The complex nonlinear nature of the sugar evaporation process, disturbances of a high intensity and correlated inputs that do not fulfill the persistent excitation condition are among the reasons for the difficulties in achieving high accuracy of system modelling.

References

1. Alataris K., Berger T. W., Marmarelis V. Z. (2000) A novel network for nonlinear modeling of neural systems with arbitrary point-process inputs. *Neural Networks*, 13:255–266.
2. Al-Duwaish H., Karim M. N., Chandrasekar V. (1996) Use of multilayer feedforward neural networks in identification and control of Wiener model. *Proc. IEE – Contr. Theory Appl.*, 143:225–258.
3. Al-Duwaish H., Nazmul Karim M., Chandrasekar V. (1997) Hammerstein model identification by multilayer feedforward neural networks. *Int. J. Syst. Sci.*, 18:49–54.
4. Al-Duwaish H., Karim M. N. (1997) A new method on the identification of Hammerstein model. *Automatica*, 33:1871–1875.
5. Al-Duwaish H. (2000) A genetic approach to the identification of linear dynamical systems with static nonlinearities. *Int. J. Contr.*, 31:307–313.
6. Aoubi M. (1998) Comparison between the dynamic multi-layered perceptron and generalised Hammerstein model for experimental identification of the loading process in diesel engines. *Contr. Engng. Practice*, 6:271–279.
7. Bai E.-W. (1998) An optimal two-stage identitcation algorithm for Hammerstein-Wiener nonlinear systems. *Automatica*, 34:333–338.
8. Bai E.-W. (2002) Identifcation of linear systems with hard input nonlinearities of known structure. *Automatica*, 38:853–860.
9. Bai E.-W. (2002) A blind approach to the Hammerstein-Wiener model identification. *Automatica*, 38:967–979.
10. Bai E.-W. (2003) Frequency domain identifcation of Wiener models. *Automatica*, 39:1521–1530.
11. Bai E.-W. (2004) Decoupling the linear and nonlinear parts in Hammerstein model identication. *Automatica*, 40:671–676.
12. Barker H. A., Tan A. H., Godfrey K. R. (2003) *Automatica*, 39:127–133.
13. Bars R., Haber R., Lengyel O. (1997) Extended horizon nonlinear adaptive predictive control applied for the parametric Hammerstein model. In: Domek S., Emirsajow Z., Kaszyński R. (eds), *Proc. 4th Int. Symp. Methods and Models in Automation and Robotics, MMAR'97*, Technical University of Szczecin Press, Szczecin, 447–451.
14. Billings S. A., Fakhouri S. Y. (1978a) Theory of separable processes with applications to the identification of non-linear systems. *Proc. IEE*, 125:1051–1058.
15. Billings S. A., Fakhouri S. Y. (1978b) Identification of a class of nonlinear systems using correlation analysis. *Proc. IEE*, 125:691–697.
16. Billings S. A., Fakhouri S. Y. (1979) Non-linear system identification using the Hammerstein model. *Int. J. Syst. Sci.*, 10: 567–578.
17. Billings S. A., Fakhouri S. Y. (1982) Identification of systems containing linear dynamic and static non-linear elements. *Automatica*, 18:15–26.

18. Billings S. A., Chen S., Korenberg M. J. (1989) Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator. *Int. J. Contr.*, 49:2157–2189.
19. Bloemen H. H. J., Boom T. J. J., Verbruggen H. B. (2001) Model-based predictive control for Hammerstein-Wiener systems. *Int. J. Contr.*, 74:482–495.
20. Bloemen H. H. J., Chou C. T., Boom T. J. J., Verdult V., Verhaegen M., Backx T. C. (2001) Wiener model identification and predictive control for dual composition control of a distillation column. *J. Process Contr.*, 11:601–620.
21. Boutayeb M., Darouach M. (1995) Recursive identification method for MISO Wiener-Hammerstein model. *IEEE Trans. Automat. Contr.*, 40:287–291.
22. Boutayeb M., Aubry D., Darouach M. (1996) A robust and recursive identification method for MIMO Hammerstein model. In: *Proc. Int. Conf. Contr, UKACC'96*, Exeter, UK, 482–495.
23. Campolucci P., Uncini A., Piazza F., Rao B. D. (1999) On-line learning algorithms for locally recurrent neural networks. *IEEE Trans. Neural Networks*, 10:253–271.
24. Carlos A., Corripio A. B. (1985) Principles and pracitice of automatic control. John Wiley and Sons, New York.
25. Celka P., Bershad N. J., Vesin J. (2001) Stochastic gradient identification of polynomial Wiener systems: analysis and application. *IEEE Trans. Signal Processing*, 49:301–313.
26. Cervantes A. L., Agamennoni O. E., Figueroa J. L. (2003) A nonlinear model predictive control based on Wiener piecewise linear models. *J. Process Contr.*, 13:655–666.
27. Chang F. H. I., Luus R. (1971) A noniterative method for identification using Hammerstein model. *IEEE Trans. Automat. Contr.*, AC-16:464–468.
28. Chen G., Chen Y., Ogmen H. (1997) Identifying chaotic systems via a Wiener-type cascade model. *IEEE Contr. Syst. Mag.*, 17:29–36.
29. Chen S., Billings S. A., Grant P. M. (1990) Non-linear system identification using neural networks. *Int. J. Contr.*, 51:1191–1214.
30. Chen J., Patton R. J. (1999) Robust model-based fault diagnosis for dynamic systems. Kluwer Academic Publishers, London.
31. Cybenko G. (1989) Approximation by superposition of sigmoidal functions. *Math. Contr. Signals and Syst.*, 2:359–366.
32. Davide F. A. M., Di Natale C., D'Amico A., Hierlemann A., Mitrovics J., Schweizer M., Weimar U., Göpel W. (1995) Structure identification of nonlinear models for QMB polymer-coated sensors. *Sensors and Actuators*, B 24–25:830–842.
33. Drewelow W., Simanski O., Hofmockel R., Pohl B. (1997) Identification of neuromuscular blockade in anaesthesia. In: Domek S., Emirsajow Z., Kaszyński R. (eds) *Proc. 4th Int. Symp. Methods and Models in Automation and Robotics, MMAR'97*, Technical University of Szczecin Press, Szczecin, 781–784.
34. Eskinat E., Johnson S. H., Luyben W. L. (1991) Use of Hammerstein models in identification of nonlinear systems. *AIChE J.*, 37:255–268.
35. Eykhoff P. (1980) System identification. Parameter and state estimation. John Wiley and Sons, London.
36. Fahlman S., Lebiere C. (1990) The cascase-correlation learning architecture. In: Touretzky D. S. (ed.) *Advances in Neural Information Processing Syst. 2*, Morgan Kaufmann, San Mateo.
37. Fine T. L. (1999) Feedforward neural network methodology. Springer, New York, Berlin, Heidelberg.
38. Frank P. M. (1990) Fault diagnosis in dynamical systems using analytical and knowledge-based redundancy – A survey of some new results. *Automatica*, 26:459–474.
39. Fruzzetti K. P., Palazogğlu A., McDonald K. A. (1997) Nonlinear model predictive control using Hammerstein models. *J. Process Contr.*, 7:31–41.
40. Gallman P. (1975) An iterative method for the identification of nonlinear systems using a Uryson model. *IEEE Trans. Automat. Contr.*, AC-20:771–775.

41. Gallman P. (1976) A comparison of two Hammerstein model identification algorithms. *IEEE Trans. Automat. Contr.*, AC-21:124–77.
42. Gerkšič S., Juričič D., Strmčnik S., Matko D. (2000) Wiener model based nonlinear predictive control. *Int. J. Syst. Sci.*, 31:189–202.
43. Giri F., Chaoui F. Z., Rochdi Y. (2001) Parameter identification of a class of Hammerstein plants. *Automatica*, 37:749–756.
44. Gómez J. C., Baeyens E. (2004) Identification of block-oriented nonlinear systems using orthonormal bases *J. Process Contr.*, 14:685–697.
45. Greblicki W. (1989) Non-parametric orthogonal series identification of Hammerstein systems. *Int. J. Syst. Sci.*, 20: 2355–2367.
46. Greblicki W. (1992) Nonparametric identification of Wiener systems. *IEEE. Trans. Inf. Theory*, 38:1487–1492.
47. Greblicki W. (1994) Nonparametric identification of Wiener systems by orthogonal series. *IEEE Trans. Automat. Contr.*, 39:2077–2086.
48. Greblicki W. (1997) Nonparametric approach to Wiener system identification. *IEEE Trans. Circ. Syst. I*, 44:538–545.
49. Greblicki W. (1998) Continuous-time Wiener system identification. *IEEE Trans. Automat. Contr.*, 43:1488–1493.
50. Greblicki W. Recursive identification of continuous-time Wiener systems. (1999) *Int. J. Contr.*, 72:981–989.
51. Greblicki W. (2001) Recursive identification of Wiener systems. *Int. J. Appl. Math. and Comp. Sci.*, 11:977–991.
52. Greblicki W. (2002) Recursive identification of continuous-time Hammerstein systems. *Int. J. Syst. Sci.*, 33: 969–977.
53. Greblicki W., Krzyżak A. (1979) Non-parametric identification of a memoryless system with a cascade structure. *Int. J. Syst. Sci.*, 10:1311–1321.
54. Greblicki W., Pawlak M. (1985) Fourier and Hermite series estimates of regression functions. *Ann. Inst. Statist. Math.*, 37:443–454.
55. Greblicki W., Pawlak M. (1986) Identification of discrete Hammerstein systems using kernel regression estimates. *IEEE Trans. Automat. Contr.*, AC-31:74–77.
56. Greblicki W., Pawlak M. (1987) Hammerstein system identification by nonparametric regression estimation. *Int. J. Contr.*, 45:343–354.
57. Greblicki W., Pawlak M. (1989) Nonparametric identification of Hammerstein systems. *IEEE Trans. Inf. Theory*, 35:409–417.
58. Greblicki W., Pawlak M. (1989) Recursive nonparametric identification of Hammerstein systems. *J. Franklin Inst.*, 326:461–481.
59. Greblicki W., Pawlak M. (1994) Nonparametric recovering nonlinearities in block oriented systems with the help of Laguerre polynomials. *Contr. Theory Advanced Techn.*, 10:771–791.
60. Gupta M. M., Jin L., Homma N. (2003) Neural networks. From fundamentals to advanced theory. John Wiley and Sons, Hoboken.
61. Haber R., Unbehauen H. (1990) Structure identification of non-linear dynamic systems – a survey on input/output approaches. *Automatica*, 26:651–677.
62. Haber R. (1995) Predictive control of nonlinear dynamic processes. *Appl. Math. and Comp.*, 70: 169–184.
63. Haist N. D., Chang F. H. I., Luus R. (1973) Nonlinear identification in the presence of correlated noise using Hammerstein model. *IEEE Trans. Automat. Contr.*, AC-18:552–555.
64. Hasiewicz Z. (1999) Hammerstein system identification by the Haar multiresolution approximation. *Int. J. Adaptive Contr. Signal Processing*, 13:697–717.
65. Hasiewicz Z. (2000) Modular neural networks for nonlinearity recovering by the Haar approximation. *Neural Networks*, 13:1107–1133.
66. Hasiewicz Z. (2001) Non-parametric estimation of nonlinearity in a cascade time series system by multiscale approximation. *Signal Processing*, 81:791–807.
67. Hassibi B., Stork D. G. (1993) Second derivatives for network pruning: optimal brain surgeon. In: Hanson S. J., Cowan J. D., Giles C. L. (eds) *Advances in Neural Information Processing Syst. 5*, Morgan Kaufmann, San Mateo.

68. Haykin S. (1999) Neural networks. A comprehensive foundation. Prentice Hall, Upper Saddle River.
69. Hertz J., Krogh A., Palmer R. G. (1991) Introduction to the theory of neural computation. Addison-Wesley, Redwood City.
70. Hunt K. J., Sbarbaro D., Żbikowski R., Gawthrop P. J. (1992) Neural networks for control systems – a survey. *Automatica*, 28:1083–1112.
71. Hunter I. W., Korenberg M. J. (1986) The identification of nonlinear biological systems: Wiener and Hammersein cascade models. *Biol. Cybern.*, 55:135–144.
72. Ikonen E., Najim K. (2001) Identification of Wiener systems with steady-state non-linearities. In: *Proc. Europ. Contr. Conf., ECC'01*, Porto, Portugal, CD-ROM.
73. Janczak A. (1995) Identification of a class of non-linear systems using neural networks. In: Bańka S., Domek S., Emirsajow Z. (eds) *Proc. 2nd Int. Symp. Methods and Models in Automation and Robotics, MMAR'95*, Technical University of Szczecin Press, Szczecin, 697–702.
74. Janczak A. (1997) Identification of Wiener models using recurrent neural networks. In: Domek S., Emirsajow Z., Kaszyński R. (eds) *Proc. 4th Int. Symp. Methods and Models in Automation and Robotics, MMAR'97*, Technical University of Szczecin Press, Szczecin, 727–732.
75. Janczak A. (1997) Recursive identification of Hammerstein systems using recurrent neural models. In: Tadeusiewicz R., Rutkowski L., Chocjan J. (eds) *Proc. the 3rd Conf. Neural Networks and Their Applications*, Polish Neural Networks Society Press, Częstochowa, 517–522.
76. Janczak A. (1998) Recurrent neural network models for identification of Wiener systems. In: Borne P., Ksouri M., El Kamel A. (eds) *Proc. 2nd IMACS Multiconference, CESA'98*, Nabeul-Hammamet, 965–970.
77. Janczak A., (1998) Gradient descent and recursive least squares learning algorithms for on line identification of Hammerstein systems using recurrent neural network models. In: Heiss M. (ed.) *Proc. Int. ICSC/IFAC Symp. Neural Computation, NC'98*, ICSC Academic Press, Vienna, 565–571.
78. Janczak A. (1999) Fault detection and isolation in Wiener systems with inverse model of static nonlinear element. In: *Proc. Europ. Contr. Conf., ECC'99*, Karlsruhe, F1046-5, CD-ROM.
79. Janczak A. (1999) Parameter estimation based fault detection and isolation in Wiener and Hammerstein systems. *Int. J. Appl. Math. and Comp. Sci.*, 9:711–735.
80. Janczak A. (2000) Least squares identification of Wiener systems. In: Domek S., Kaszyński R. (eds) *Proc. 6th Int. Conf. Methods and Models in Automation and Robotics, MMAR'2000*, Technical University of Szczecin Press, Szczecin, 933–938.
81. Janczak A. (2000) Neural networks in identification of Wiener and Hammerstein systems. In: Duch W., Korbicz J., Rutkowski L., Tadeusiewicz R. (eds) *Biocybernetics and biomedical engineering 2000. Neural networks*, Akad. Ofic. Wyd. EXIT, Warsaw, 419–458, (in Polish).
82. Janczak A. (2000) Parametric and neural network models for fault detection and isolation of industrial process sub-modules. In: Edelmayer M., Banyśsz C. (eds) *Prepr. 4th Symp. Fault Detection Supervision and Safety for Technical Processes, SAFEPROCESS'2000*, Budapest, Hungary, 348–351.
83. Janczak A. (2001) On identification of Wiener systems based on a modified serial-parallel model. In: *Proc. Europ. Contr. Conf., ECC'2001*, Porto, Portugal, 1852-1857.
84. Janczak A. (2001) Training neural network Hammerstein models with truncated back-propagation through time algorithm. In: Kaszyński R. (ed.) *Proc. 7th IEEE Int. Conf. Methods. and Models in Automation and Robotics, MMAR'2001*, Technical University of Szczecin Press, Szczecin, 499–504.
85. Janczak A. (2002) Prediction error approach to identification of polynomial Wiener systems. In: Domek S., Kaszyński R. (eds) *Proc. 8th IEEE Int. Conf. Methods and Models in Automation and Robotics, MMAR'2002*, Technical University of Szczecin Press, Szczecin, 457-461.

86. Janczak A. (2002) Identification of Wiener systems with pseudolinear regression approach method. In: Bubnicki Z., Korbicz J. (eds) *Proc. 14th Polish Conf. Automation, XIV KKA*, Zielona Góra, 413–416, (in Polish).
87. Janczak A. (2002) Training of neural network Wiener models with recursive prediction error algorithm. In: Rutkowski L., Kacprzyk J. (eds) Advances in Soft Computing. Neural Networks and Soft Computing, *Proc. 6th Int. Conf. Neural Networks and Soft Computing*, Physica-Verlag, Heidelberg, New York, 692–697.
88. Janczak A. (2003) Identification of Wiener and Hammerstein systems with neural networks and polynomial models. Methods and applications. University of Zielona Góra Press, Zielona Góra.
89. Janczak A. (2003) A comparison of four gradient learning algorithms for neural network Wiener models. *Int. J. Syst. Sci.*, 34:21–35
90. Janczak A. (2003) Neural network approach for identification of Hammerstein systems *Int. J. Contr.*, 76:1749–1766.
91. Janczak A. (2004) Parametric and neural network models in fault detection and isolation. In: Korbicz J., Kościelny J. M., Kowalczuk Z., Cholewa W. (eds) *Fault diagnosis, models, artificial intelligence, applications*, Springer, Berlin, Heidelberg, New York, 381–410.
92. Janczak A., Korbicz J. (1999) Neural network models of Hammerstein systems and their application to fault detection and isolation In: *Proc. 14th World Congress of IFAC*, Beijing, P.R.C., P:91–96.
93. Janczak A., Mrugalski M (2000) Neural network approach to identification of Wiener systems in a noisy environment. In: Bothe H., Rojas R. (eds) *Proc. Int. ICSC Symp. Neural Computation, NC'2000*, Berlin.
94. Juditsky A., Hjalmarsson H., Benveniste A., Delyon B., Ljung L., Sjöberg J., Zhang O. (1995) Nonlinear Black-box modeling in system identification: mathematical foundations. *Automatica*, 31:1725–1750.
95. Kalafatis A. D., Arifin N., Wang L., Cluett W. R. (1995) A new approach to the identification of pH processes based on the Wiener model. *Chem. Engng Sci.*, 50:3693–3701.
96. Kalafatis A. D., Wang L., Cluett W. R. (1997) Identification of Wiener-type non-linear systems in a noisy environment. *Int. J. Contr.*, 66:923–941.
97. Kang H. W., Cho Y. S., Youn D. H. (1998) Adaptive precomensation of Wiener systems. *IEEE Trans. Signal Processing*, 46:2825–2829.
98. Knohl T., Unbehauen H. (2000) Adaptive position control of electrohydraulic servo systems using ANN. *Mechatronics*, 10:127–143.
99. Knohl T., Xu W. M., Unbehauen H. (2003) Indirect adaptive dual control for Hammerstein systems using ANN. *Contr. Engng Practice*, 11:377–385.
100. Korbicz J., Janczak A. (1996) A neural network approach to identification of structural systems. In: *Proc. IEEE Int. Symp. Industrial Electronics, ISIE'96*, Warsaw, pp. 97–103.
101. Korbicz J., Janczak A. (2002) Artificial neural network models for fault detection and isolation of industrial processes. *Comp. Assist. Mech. and Engng Sci.*, 9:55–69.
102. Krzyżak A. (1989) Identification of discrete Hammerstein systems by the Fourier series regression estimate. *Int. J. Syst. Sci.*, 20:1729–1744
103. Krzyżak A. (1990) On nonparametric estimation of nonlinear dynamic systems by the Fourier series estimate. *Signal Processing*, 52:299–321.
104. Krzyżak A. (1996) On estimation of a class of nonlinear systems by the kernel regressin estimate. *IEEE Trans. Inf. Theory*, 36:141–152.
105. Krzyżak A., Partyka M. A. (1993) On identification of block-oriented systems by non-parametric techniques. *Int. J. Syst. Sci.*, 24:1049–1066.
106. Krzyżak A., Sąsiadek J. Z., Kégl B. (2001) Non-parametric identification of dynamic nonlinear systems by a Hermite series approach. *Int. J. Syst. Sci.*, 32:1261–1285
107. Le Cun Y., Kanter I., Solla S. A. (1990) Optimal brain damage. In: Touretzky D. S. (ed.) *Advances in Neural Information Processing Syst. 2*, Morgan Kaufmann, San Mateo.

108. Leontaritis I. J., Billings S. A. (1985) Input–output parametric models for non-linear systems. Part I: deterministic non-linear systems. *Int. J. Contr.*, 41:303–328.
109. Leontaritis I. J., Billings S. A. (1985) Input–output parametric models for non-linear systems. Part II: stochastic non-linear systems. *Int. J. Contr.*, 41:329–344.
110. Ling W.-M., Rivera D. (1998) Nonlinear black-box identification of distillation column models — design variable selection for model performance enhancement. *Appl. Math. and Comp. Sci.*, 8:794–813.
111. Lissane Elhaq S., Giri F., Unbehauen H. (1999) Modelling, identification and control of sugar evaporation – theoretical design and experimental evaluation. *Contr. Engng Practice*, 7:931–942.
112. Ljung L. (1999) System identification. Theory for the user. Prentice Hall, Upper Saddle River.
113. Lovera M., Gustafsson T., Verhaegen M. (2000) Recursive subspace identifcation of linear and non-linear Wiener state-space models. *Automatica*, 36:1639–1650.
114. Luyben W. L., Eskinat E. (1994) Nonlinear auto-tune identification. *Int. J. Contr.*, 59:595–626.
115. Mak M. W., Ku K. W., Lu Y. L. (1999) On the improvement of the real time recurrent learning algorithm for recurrent neural networks. *Neurocomputing*, 24:13–36.
116. Marciak C., Latawiec K., Rojek R., Oliveira G. H. C. (2001) In: Kaszyński R. (ed.) *Proc. 7th IEEE Int. Conf. Methods and Models in Automation and Robotics, MMAR'2001*, Technical University of Szczecin Press, Szczecin, 965–969.
117. Marmarelis V. Z., Zhao X. (1997) Volterra models and three-layer perceptrons. *IEEE Trans. Neural Networks*, 8:1421–1432.
118. Menold P. H., Allgöwer F., Pearson R. K. (1997) Nonlinear structure identification of chemical processes. *Computers and Chem. Engng*, 21:S137–S142.
119. Mzyk G. (2002) Instrumental variables in Wiener system identification. In: Domek S., Kaszyński R. (eds) *Proc. 8th IEEE Int. Conf. Methods and Models in Automation and Robotics, MMAR'2002*, Technical University of Szczecin Press, Szczecin, 463–468.
120. Narendra K. S., Gallman P. G. (1966) An iterative method for the identification of nonlinear systems using Hammerstein model. *IEEE Trans. Automat. Contr.*, AC-11:546–550.
121. Narendra K. S., Parthasarathy K. (1990) Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1:4–26.
122. Narendra K. S., Parthasarathy K. (1991) Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Trans. Neural Networks*, 2:252–262.
123. Nelles O. (2001) Nonlinear system identification. From classical approaches to neural networks and fuzzy models. Springer, New York, Berlin, Heidelberg.
124. Nešić D. (1997) A note on dead-beat controllability of generalized Hammerstein systems. *Syst. and Contr. Letters*, 29:223–231.
125. Nešić D., Mareels I. M. Y. (1998) Dead-beat control of simple Hammerstein models. *IEEE Trans. Automat. Contr.*, 43:1184–1188.
126. Ninnes B., Gibson S. (2002) Quantifying the accuracy of Hammerstein model estimation.*Automatica*, 38:2037–2051.
127. Nørgaard M., Ravn O., Poulsen N. K., Hansen L. K. (2000) Neural networks for modelling and control. Springer, New York, Berlin, Heidelberg.
128. Norquay S. J., Palazoglu A., Romagnoli J. A. (1998) Model predictive control based on Wiener models. *Chem. Engng Sci.*, 53:75–84.
129. Norquay S. J., Palazoglu A., Romagnoli J. A. (1999) Application of Wiener model predictive control (WMPC) to an industrial C2-splitter. *J. Process Contr.*, 9:461–473.
130. Norquay S. J., Palazoglu A., Romagnoli J. A. (1999) Application of Wiener model predictive control (WMPC) to a pH neutralization experiment. *IEEE Trans. Contr. Syst. Technology*, 7:437–445.

131. Pacut A. (2000) Stochastic modelling at diverse scales. From Poisson to network neurons. Warsaw University of Technology Press, Warsaw.
132. Pacut A (2002) Symmetry of backpropagation and chain rule. In: *Proc. 2002 Int. Joint Conf. Neural Networks*, Honolulu, HA, IEEE Press, Piscataway.
133. Pajunen G. (1992) Adaptive control of Wiener type nonlinear systems. *Automatica*, 28:781–785.
134. Patwardhan R. S., Lakshminarayanan S., Shah S. L. (1998) Constrained nonlinear MPC using Hammerstein and Wiener models. *AIChE J.*, 44:1611–1622
135. Patton R. J., Frank M., Clark R. N. (1990) Fault diagnosis in dynamic systems. Theory and applications. Prentice-Hall, New York.
136. Pawlak W. (1991) On the series expansion approach to the identification of Hammerstein systems. *IEEE Trans. Automat. Contr.*, 36:763–767.
137. Pawlak W., Hasiewicz Z. (1998) Nonlinear system identification by the Haar multiresolution analysis. *IEEE Trans. Circ. and Syst. – I: Fund. Theory and Appl.*, 45:945–961.
138. Pearson R. K., Pottmann M. (2000) Gray-box identification of block-oriented non-linear models. *J. Process Contr.*, 10:301–315.
139. Piché S. W. (1994) Steepest descent algorithms for neural network controllers. *IEEE Trans. Neural Networks*, 5:198–212.
140. Pomerleau D., Hodouin D., Poulin É. (2003) Performance analysis of a dynamic phenomenological controller for a pellet cooling process. *J. Process Contr.*, 13:139–153.
141. Quaglini V., Previdi F., Contro R., Bittanti S. (2002) A discrete-time nonlinear Wiener model for the relaxation of soft biological tissues. *Medical Engng and Physics*, 24:9–19.
142. Rollins D. K., Bhandari N. (2004) Constrained MIMO dynamic discrete-time modeling exploiting optimal experimental design. *J. Process Contr.*, 14:671–683.
143. Roux G., Dahhou B., Queinnec I. (1996) Modelling and estimation aspects of adaptive predictive control in a fermentation process. *Contr. Engng Practice*, 4:55–66.
144. Rutkowski L. (2004) New soft computing techniques for system modelling, pattern classification and image processing. Springer, Berlin, Heidelberg.
145. Schetzen M. (1980) The Volterra and Wiener theories of ninlinear systems. John Wiley and Sons, New York.
146. Sentoni G., Agamennoni O., Desages A., Romagnoli J. (1996) Aproximate models for nonlinear process control. *AIChE J.*, 42:2240–2250.
147. Sieben S. (1996) The Wiener model – an approach by deterministic inputs. In: *Proc. 1st IMACS Multiconference, CESA'96*, Lille, France, 465–469.
148. Sjöberg J., Zhang O., Ljung L., Benveniste A., Delyon B., Glorennec P., Hjalmarsson H., Juditsky A. (1995) Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31:1691–1724.
149. Söderström T., Stoica P. (1994) System identification. Prentice Hall Int., London.
150. Srinivasan B., Prasad U. R., Rao N. J. (1994) Back propagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural models. *IEEE Trans. Neural Networks*, 5:213–228.
151. Stapleton J. C., Baas S. C. (1985) Adaptive noise cancellation for a class of nonlinear systems. *IEEE Trans. Circ. and Syst.*, 32:143–150.
152. Stoica P., Söderström T. (1982) Instrumental-variable methods for identification of Hammerstein systems. *Int. J. Contr.*, 35:459–476.
153. Su H.-T., McAvoy T. J. (1993) Integration of multilayer perceptron networks and linear dynamic models: A Hammerstein modeling approach. *Ind. Engng Chem. Res.*, 32:1927–1936.
154. Sung S. W., Lee J. (2004) Modeling and control of Wiener-type processes. *Chem. Engng Sci.*, 59:1515–1521
155. Śliwiński P., Hasiewicz Z. (2002) Computational algorithms for wavelet-based system identification. In: Domek S., Kaszyński R. (eds) *Proc. 8th IEEE Int. Conf. Methods and Models in Automation and Robotics, MMAR'2002*, Technical University of Szczecin Press, Szczecin, 495–500.

156. Thathachar M. A. L., Ramaswamy S. (1973) Identification of a class of nonlinear systems. *Int. J. Contr.*, 18:741–752.
157. Verhaegen M., Westwick D. (1996) Identifying MIMO Hammerstein systems in the context of subspace model identyfication methods. *Int. J. Contr.*, 63:331–349.
158. Verhaegen M., Westwick D. (1996) Identifying MIMO Wiener systems using subspace model identyfication methods. *Signal Processing*, 52:235–258.
159. Visala A., Pitkänen H., Aarne H. (2001) Modeling of chromatographic separation process with Wiener-MLP representation. *J. Process Contr.*, 78:443–458.
160. Vörös J. (1997) Parameter identification of discontinuous Hammerstein systems. *Automatica*, 33:1141–1146.
161. Vörös J. (1999) Iterative algorithm for identification of Hammerstein systems with two-segment nonlinearities. *IEEE Trans. Automat. Contr.*, 44:2145–2149.
162. Vörös J. (2001) Parameter identifcation of Wiener systems with discontinuous nonlinearities. *Syst. and Contr. Letters*, 44:363–372.
163. Werbos P. J. (1990) Backpropagation through time: What it does and how to do it. *Proc. IEEE*, 78:1550–1560.
164. Westwick D., Verhaegen M. (1996) Identifying MIMO Wiener systems using subspace identification methods. *Signal Processing*, 52:235–258.
165. Wigren T. (1993) Recursive prediction error identification using the non-linear Wiener model. *Automatica*, 39:1011–1025.
166. Wigren T. (1994) Convergence analysis of recursive identification algorithms based on the non-linear Wiener model. *IEEE Trans. Automat. Contr.*, 39:2191–2206.
167. Williams R. J., Zipser D. (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural Computations*, 1:271–280.
168. Xu M., Chen G., Tian Y.-T. (2001) Identifying chaotic systems using Wiener and Hammerstein cascade models. *Math. and Computer Modelling*, 33:483–493.
169. Żurada J. M. (1992) Introduction to artificial neural systems. West Publishing Company, St. Paul.

Index