A Practical Approach to Model Based Neural Network Control

Hannu Koivisto

Tampere University of Technology
Control Engineering Laboratory

P.O. Box 692, FIN-33101 Tampere, Finland

email: hannu.koivisto@ae.tut.fi
fax: +358-31-3162340
phone: +358-31-3162656

Koivisto Hannu J: A Practical Approach to Model Based Neural Network Control
Tampere University of Technology, Tampere, Finland, 1995
Tampere University of Technology Publications 170

Abstract

This thesis presents a practical approach to model based control of nonlinear dynamical systems using multilayered perceptron type neural networks as process models and controllers. A framework for modeling and identification of nonlinear time series models is introduced. Advanced parameter estimation methods are used to identify the weights of the process model and controller networks. A novel stability analysis and practical methods for maintaining stability and generalization capability are introduced.

The identified models are used for model predictive control. Both the direct long range predictive control approach and the dual network control approach are applied. The problems related to the model inverse based IMC design are partially avoided by employing a nonlinear optimal controller within the IMC structure. General guidelines and practical methods for model mismatch and for maintaining stability are introduced and applied.

The neuro-control workstation based on HP9000/425 platform and the software tools for identification and control are also introduced. This forms an efficient tool for neural network identification, control design and real-time control tasks.

The approach is applied in the control of small laboratory scale water and air heating processes, and in the multivariate control of a pilot headbox of a paper machine. The experimental results show that the proposed approach yields good performance characteristics and robust control.

Keywords: feedforward neural nets, recurrent neural nets, nonlinear models, time series models, identification, neurocontrol, intelligent control, predictive control, nonlinear control, adaptive control, stability, computer control, real-time systems

Acknowledgements

The work has been carried out in the Control Engineering Laboratory, Department of Electrical Engineering at Tampere University of Technology during the years 1990-1995.

Many people have given me their time, advice, encouragement and support. One person, Professor Heikki Koivo, stands apart from the others. I wish to express my gratitude to him for his guidance and continuous support during the course of this thesis and my research work in this area.

I am grateful to my friends and co-workers in the Control Engineering Laboratory and especially to the neuro-fuzzy research group, which deserve thanks for all their support during the work. It is my pleasure to sincerely thank Dr. Terho Jussila and Dr. Jukka Lieslehto for many important discussions concerning this work and for reading unfinished versions of the manuscript. I would also thank DI Ari Nissinen, who generously helped in proofreading and in polishing the language.

This thesis was financially supported by the *Academy of Finland, Technology Development Centre in Finland, Foundation for the Advancement of Technology in Finland* and *Wihuri Foundation,* which are gratefully acknowledged.

Finally, my best thanks to my wife Aino-Liisa and daughter Aura for their unending support and patience in the course of my research work.

Tampere, November 1995

Hannu Koivisto

Table of Contents

Abstract i

Acknowledgements ii

Operators and Notational Conventions iii

Abbreviations v

1 Introduction 1

2 Neural Networks as Function Approximators 9

2.1 Neural Networks ... 9
2.2 Global Function Approximation ... 11
2.3 Local Function Approximation ... 16
2.4 Global vs. Local Models ... 20

3 Modelling and Identification 23

3.1 Nonlinear Stochastic Models ... 23
3.2 Nonlinear Predictors ... 29
3.3 Prediction Error Method ... 37
3.4 Stability and Convergence ... 46

4 Model Predictive Control 57

4.1 Control Methods ... 58
4.2 Direct Predictive Control ... 64
4.3 Dual Network Control ... 76

5 Simulation Studies 87

5.1 Experimental Setup ... 87
5.2 Simulation Examples ... 90

6 Control of Water Heating Process 105

6.1 Identification ... 106
6.2 Adaptive Control ... 110
6.3 Dual Network Control - Constant Parameters ... 118

7 Control of Air Heating Process 121

7.1 Identification ... 122
7.2 Control Experiments ... 130

8 Control of Pilot Headbox 135

8.1 Identification ... 137
8.2 Control Experiments ... 142

9 Conclusion 147

References 151

Appendix A Multilayer Perceptron A.1

Operators and Notational Conventions

col $\{A\}$	stacks the columns of the matrix A under each other (a column vector)
det $\{A\}$	determinant of a matrix
diag $\{A\}$	diagonal of a matrix A (a column vector)
diag $\{x\}$	diagonal matrix with the vector x as its main diagonal
$\|\| x \|\|_\Lambda$	weighted norm, Euclidian by default i.e. $\|\| x \|\|_\Lambda^2 = x^T \Lambda x$
$\|\| x \|\|_\infty$	maximum norm
$\|A\|$	element-by-element absolute value of the matrix A
arg min $V(x)$	value of x that minimizes $V(x)$
q, q^{-1}	forward and backward shift operators
$A(q^{-1})$	matrix polynomial in q^{-1}
$A(q^{-1}, t)$	time varying matrix polynomial in q^{-1}
$A(q)$	matrix polynomial in q
roots $(A(q))$	roots of the polynomial $A(q) = 0$
$G(z)$, $G(q)$	transfer function matrix
$A^{(i)}$	ith column of the matrix A
$y^{(t-1)}$	a finite length past history of the sequence $\{y(t)\}$, $y(t-1)$ as newest
$\frac{\partial}{\partial x} f(x)$	Jacobian $\left[\frac{\partial}{\partial x_1} f(x), \ldots, \frac{\partial}{\partial x_n} f(x)\right]$, i.e. a row vector, if f scalar
$f(x, \theta)$ $f(\varphi, \theta)$	model, a function $f : \boldsymbol{R}^n \to \boldsymbol{R}^m$ with the input regression vector $\varphi \in \boldsymbol{R}^n$ or $x \in \boldsymbol{R}^n$ and parameter vector $\theta \in \boldsymbol{R}^{n_\theta}$. A shorthand notation $f(\varphi)$ and $f(x)$ are also used.
$h(\phi, \Theta)$	controller, a function $h : \boldsymbol{R}^n \to \boldsymbol{R}^m$ with the input regression vector $\phi \in \boldsymbol{R}^n$ and parameter vector $\Theta \in \boldsymbol{R}^{n_\Theta}$. A notation $h(\phi)$ also used.
$V'(\theta)$	gradient of $V(\theta)$ w.r.t. to its arguments. This and all other gradients are row vectors if V is scalar.
$V''(\theta)$	second order gradient of $V(\theta)$ w.r.t. to its arguments

$u(t)$	input variable, control signal. Also a shorthand notation for $u(t\|\Theta)$
$u(t\|\Theta)$	a prediction of a control signal using the controller parameters Θ and based on data at time $t-1$
$y(t)$	output variable
$\hat{y}(t\|\theta)$	predicted output at time t, using the model parameters θ and based on data at time $t-1$
$\hat{y}(t\|\Theta)$	predicted output at time t, using the controller parameters Θ and based on data at time $t-1$
$\hat{y}(t+i\|t)$	predicted output at time $t+i$ based on data at time t
$\hat{y}(t)$	a shorthand notation used instead $\hat{y}(t\|\theta)$, $\hat{y}(t\|\Theta)$ or $\hat{y}(t\|t-1)$
$y^*(t)$	target variable (desired output) at time t
$\hat{y}_f(t)$	filtered prediction $\hat{y}(t)$
$\varphi(t)$, $\varphi(t,\theta)$	model regression (data) vector at time t
$\phi(t)$, $\phi(t,\Theta)$	controller regression (data) vector at time t
$\varepsilon(t,\theta)$	prediction error $y(t)-\hat{y}(t\|\theta)$
$\varepsilon(t,\Theta)$	equivalent control error
$\varepsilon(t)$	shorthand notation for $\varepsilon(t,\theta)$ or $\varepsilon(t,\Theta)$
$\Psi(t,\theta)$	gradient of $\hat{y}(t\|\theta)$ w.r.t θ.
$\Psi(t,\Theta)$	negative gradient of equivalent control error $\varepsilon(t)$ w.r.t Θ.
$\Psi_y(t,\Theta)$	gradient of $\hat{y}(t\|\Theta)$ w.r.t Θ.
$\Psi_u(t,\Theta)$	gradient of $u(t\|\Theta)$ w.r.t Θ.
$\Xi(t,\theta)$	gradient of $f(\varphi,\theta)$ w.r.t θ
$\Xi_u(t,\Theta)$	gradient of $h(\phi,\Theta)$ w.r.t Θ
$\Gamma(t,\theta)$	gradient of $f(\varphi,\theta)$ w.r.t φ
$\Gamma_y(t,\Theta)$	gradient of $f(\varphi,\theta)$ w.r.t φ
$\Gamma_u(t,\Theta)$	gradient of $h(\phi,\Theta)$ w.r.t Θ

Abbreviations

ARX	(linear) AutoRegressive with eXogenous input (predictor)
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CMAC	Cerebellar Model Articulation Computer
CSTR	Continuous Stirred Tank Reactor
IMA	Integrating Moving Average (noise)
IMC	Internal Model Control
LM	Levenberg-Marquardt
LRPC	Long Range Predictive Control
MFD	Matrix Fraction Description
MIMO	Multi-Input Multi-Output
MLP	Multi Layer Perceptron (neural network)
MSSE	Mean Sum of Squared Errors
MWGS	Modified Weighted Gram-Schmidt algorithm
NARX	Nonlinear AutoRegressive with eXogenous input (predictor)
NARMAX	Nonlinear AutoRegressive Moving Average with eXogenous input
NARIMAX	Nonlinear AutoRegressive Moving Average with Integrating noise and eXogenous input
NOE	Nonlinear Output Error model
OE	(linear) Output Error model
PRBS	Pseudo Random Binary Sequence
PRS	Pesudo Random Sequnce
RBF	Radial Base Function (neural network)
(R)PEM	(Recursive) Prediction Error Method
SISO	Single-Input Single-Output
TITO	Two-Input-Two-Outpt
w.r.t	with respect to

1 Introduction

This thesis considers practical real-time model predictive control of nonlinear dynamical systems using artificial neural networks (ANN) as process models and controllers. A framework for modelling identification and control of nonlinear time series models is presented. Advanced parameter estimation methods are used to identify the weights of the process model and the controller networks. The performance of the control systems is verified with real-time experiments using laboratory pilot processes.

This study started at the beginning of 1990's, at a time when there was a belief that artificial neural networks as itself include some mysterious “intelligence” and “robustness” and that a combination of an ANN and some “learning” method will show similar adaptiveness and capability to cope with the real world environment as humans do in everyday life. Whether this is true for the massively parallel ANN's we will see in the future - it is true for the human brain with its 10^8 neurons and 10^{12} synapses - but definitely not for the small and moderate sized “artificial neural networks” which engineers use for function approximation and classification tasks. They resemble - and they should resemble - the established methods of statistics and numerical analysis.

The background of the neural computation somewhat reflects to the study, especially in the way that the study started directly as a research towards adaptive neural network control. This was also due to the long expertize of the Control Engineering Laboratory at Tampere University of Technology in linear self-tuning control. Soon it was realized that lessons learnt when applying linear self-tuning control must be taken seriously. The first results were encouraging, c.f. Koivisto (1990) and Raiskila and Koivo (1990). Also the first real-time experiments of the adaptive neural network control proved to be a success, Koivisto *et. al.* (1991a, 1991b). The adaptive approach worked - mostly, but sometimes it failed for unexplained reasons.

Off-line identified neural network models were applied with the direct model predictive control approach - with no advantage. Only after implementation of advanced off-line identification methods and after detailed analysis of the properties of the resulting models and controllers, like stability and extrapolation issues, the reasons for failures were isolated and explained: even small neural network models are often too flexible function approximators. They must be constrained in order to maintain the stability and the generalization capability. This increased the reliability of the approach and now practical results were achieved, c.f. Koivisto *et. al.* (1992, 1993) and Koivisto (1994).

Some History

The history of ANN's can be traced back to the paper of McCulloch and Pitts (1943) introducing a simplified model of one neuron. A good review of the history of neural computation can be found in Widrow (1990) or from the excellent collection of early papers, by Anderson and Rosenfeld (1988). Almost 50 different types of neural network architectures are developed, although only some of them are in common use. There are numerous introductory level books to neural computation. A more detailed presentation and analysis can be found for example in Rumelhart and McClelland (1986) or Herz *et al.* (1991).

The birth of the field as a self-conscious, organized discipline should probably be dated back to June 1987, when the first IEEE Conference on Neural Networks (ICNN) took place in San Diego. Everyone was surprised when almost 2000 people showed up (Werbos 1989). The book of Rumelhart and McClelland (1986) and articles like Hopfield and Tank (1985) had aroused great expectations. A similar hyperbole for neural networks in control can be dated back to the Boston summer school 1988. The abstracts of this conference were published in the first number of Neural Networks (1988).

Since then neural network control has progressed rapidly and also real industrial applications have been reported, e.g. Widrow *et al.* (1994), especially in Japan, e.g. Asakawa and Takagi (1994). A computer survey (INSPEC) considering the experimental or practical applications of the neural network control in 1994 resulted 262 journal articles and 869 conference papers. Of course only a portion of all these were real on-line tests of neural network control. The fuzzy control was excluded from the search to reduce the number of articles, although “Neuro-fuzzy systems” form an entity and it is often hard to distinguish between all the approaches, except by the title.

Neurocontrol

Most promising application areas of the neural network control are robotics, process control, vehicle guidance and teleoperation. A good introduction to the subject is given in the books “Neural Networks for Control” (ed. Miller, Sutton and Werbos, 1990) and “Handbook of Intelligent Control” (ed. White and Sofge, 1992). Artificial neural networks can be used in several fields of control engineering: as process models, as controllers, for optimization and failure detection.

Relevant work on optimization and especially on fault diagnosis has been done, but the scope of this study is restricted to process control applications i.e. modelling, identification and control of nonlinear dynamical systems using artificial neural networks. Critical research issues, when considering neural network control of dynamical systems, were (“Neural Networks for Control”, 1990):

- Integration of neural network models in control architectures
- Neural network representation for dynamical systems
- Optimization of performance over time
- Stability theory for closed loop learning controllers
- Comparison of tools for adaptive/learning control.

The importance of these issues is still the same. Most of them will be considered in this thesis in conjunction with process control.

Neural networks were first applied to robotics, using reinforcement learning schemes, e.g. Barto *et al*. (1983), Anderson (1989), Michie and Chambers (1968) or applying supervised learning to inverse kinematics problem, e.g Psaltis *et al.* (1988), Elsley (1988), Bar-Kana and Guez (1989), Zeman *et al.* (1989). Also on-line experiments were reported quite soon, e.g. Miller (1989), Yabuta *et al*. (1989). The pioneering work of Widrow in the sixties should also mentioned (Widrow 1990).

Robot control applications are not largely considered in this study. The main interest is on process control. Bavarian (1988) presented the first overview of neural network control. Narendra and Parthasarathy (1990) presented a general framework for identification and control of dynamical systems using neural networks. Bhat and McAvoy (1990) presented first demonstrations of neural network control of chemical process systems. Since then numerous articles have considered neural networks for process control. A more detailed survey will be presented in Section 4.1.

The potential usefulness of neural networks in control is due to two main features: learning capability and function approximation capability. Two separate goals can be seen: learning/adaptation and representation of nonlinear models, although these are normally combined to (nonadaptive) nonlinear controller design or to adaptive nonlinear control. The attractive feature of neural network models is that they offer a parametric function which can accurately represent virtually any smooth nonlinear function with similar basic structure.

Nonlinear Control

Systems in the process control area are more or less nonlinear. Most of these processes can be successfully controlled with conventional linear methods like PID control. If the required operation region is large or the process is strongly nonlinear, a linear controller is likely to perform poorly or to become even unstable. Nonlinear controllers can properly compensate the nonlinearities in a system. The application of nonlinear control methods has been limited by the theoretical and computational difficulties associated with the practical nonlinear control design. The representation problem has been another limiting factor and the nonlinear control theory has mainly been concerned with deterministic

continuous time models derived from physical principles like mass and energy balances. Function approximation networks have remarkably reduced this representation difficulty.

From the mathematical point of view, even the control of known nonlinear dynamical system is a formidable problem. This becomes substantially more complex when the system is not completely known, e.g. Levin and Narendra (1992). We are far from being able to optimally design controllers that can handle time varying and uncertain nonlinear systems. Robust control of nonlinear systems is currently one of the most active research areas in the control literature.

Two main approaches to control of nonlinear systems are: differential geometric approach and model based approach. Differential geometric framework allows an *exact* analytic linearization of the nonlinear model using a nonlinear controller, e.g. Isidori (1989) or Slotine and Li (1991), sometimes at a price of robustness. Linear controllers can then be designed for the equivalent controlled linear system. The geometric approach produces excellent results if some basic assumptions hold, mainly the assumption of a deterministic bilinear type nonlinear system. These methods have been extended to more general models, but still some analytical invertibility assumptions must hold. See Henson and Seborg (1990) for an overview.

Neural networks have successfully been applied to adaptive nonlinear control using the geometric approach. Tzirkel-Handcock and Fallside (1992) demonstrate that neural networks can be successfully used to perform an approximate input/output linearization. Notice that a direct continuous time controller is used. Sanner and Slotine (1991) have presented similar results.

Model based approach, on the other hand, normally uses empirically identified linear or nonlinear discrete time models. More emphasis is put on the robustness of control system. The survey paper (Garcia *et al*. 1989) refers to Model Predictive Control (MPC) as that family of controllers in which there is a direct use of explicit and separate identifiable model. The same process model is also implicitly used to compute the control action in a such way that the control design specifications are satisfied.

Control design methods based on the MPC concept have found a wide acceptance in industrial applications due to their high performance and robustness. There are several variants of model predictive control methods, like Dynamic Matrix Control (DMC), Model Algorithmic Control (MAC) and Internal Model Control (IMC). Nonlinear versions of these have also been developed, for example a nonlinear IMC concept, e.g. Economou *et al*. (1986). Another, largely independently developed branch of MPC, called Generalized Predictive Control (GPC), is aimed more for adaptive control, e.g. Clarke and Mohtadi (1989). For the current state-of-the-art, see Clarke (1994).

Model predictive control in this sense is a broad area and some confusion is encountered, because the abbreviation MPC is often used to mean receding horizon (RHPC) or long range predictive control (LRPC), where a model is used to predict the process output several steps into the future and the control action is computed at each step by numerical minimization of the prediction errors i.e. no specific controller is used. This is quite different from the concept where the model is controlled with a implicitly derived specific controller, like in many IMC approaches.

Neural networks have successfully been applied to model based control of nonlinear systems. General guidelines can be found from Nahas *et al.* (1992), Psichogios and Ungar (1991), Hunt and Sbarbaro (1991) and Ydstie (1990). A more detailed survey is presented in Section 4.1.

Adaptive and Learning Systems

Several learning and adaptive methodologies have been studied in the literature to improve the performance of a control system by using the past experience. There is some inexactness in the use of the terms *learning control*, *adaptive control* and *self-tuning control*. Franklin (1989) presented a historical review of the connectionistic learning control. She distinguished between a learning controller with a self-organizing structure and an adaptive controller with a fixed structure (as defined by Landau 1979). In this sense adaptive control is equivalent to self-tuning control.

Fu (1970), an early pioneer in learning control, distinguished between supervised and reinforcement type learning. In supervised learning systems the exact knowledge of the correct responses is known. If not, the system must be able to extract useful information from the environment by a simple reinforcement signal, indicating merely whether the output is currently “good” or “bad”, but not providing any specific information how the output may be improved. Reinforcement learning schemes have a strong connection to the classical conditioning theory. Kentridge (1990) considered the subject on the general level.

Reinforcement type connectionistic learning control has been studied for example by Sutton (1984, 1988), and Anderson (1989). It can be applied to complex real world problems, where the performance of supervised learning is poor in many cases. Most of these learning systems use algorithms with fixed structure, but they can hardly be called only adaptive in the sense of Landau (1979).

Supervised learning is a good choice, if correct responses are known, because it “learns” faster. Only models and controllers with a fixed structure and the supervised learning scheme are considered in this study and “adaptive control” is used as a synonym for “supervised self-tuning control”. Similarly "learning" is often (mis)used instead of "identification".

Organization of the Thesis

As mentioned before, the research results concerning model based neural network control are numerous. What has been especially lacking is the use of efficient identification methods and their analysis, and practical results, not only simulation studies. This thesis addresses both of these issues using the framework of multilayered perceptrons.

The goal of this study was to develop efficient identification and control design methods for neural network based nonlinear control, and to implement them in a real world environment. The performance of the control systems was verified by simulations and with real-time experiments using pilot processes. Thus the study also fills the gap between theory and practice. This wide area is not so popular among neural network researchers, because of its "difficulty" and especially due to its time consuming nature. Practice is anyway the final measure to any control method and the area is of great importance.

Function approximation using artificial neural networks is considered in Chapter 2. The properties of global and localized function approximators are discussed and basic ideas of the nonlinear regression schemes are presented. The multilayer perceptron neural network is selected to be the basic block for representation of nonlinear process models and controllers. As mentioned earlier, the main attractive feature of neural network models is that they offer a parametric function which can accurately represent virtually any smooth nonlinear function with similar basic structure.

A framework for modelling and identification of nonlinear time series models is introduced in Chapter 3. The basic properties of the nonlinear stochastic models are discussed and the structures for nonlinear predictors based on time series approach are introduced. The multivariate case and the multistep prediction are also considered.

An advanced identification scheme for the parameter estimation of these neural network predictors is presented in Section 3.3. The main underlying idea behind the presented *Prediction Error Method* is that the gradient of the identification cost function with respect to the network weights is computed in a proper manner corresponding to the dynamical nature of the predictor. Methods for the stability analysis and for the projection onto the stable domain are introduced in Section 3.4. Also other projection schemes are considered.

Chapter 4 presents the *Model Predictive Control* approach. A short review of the existing neural network control methods and their applications are presented in Section 4.1. The direct predictive control method is considered in Section 4.2. The identified neural network model is used to predict the future process measurements and the control action at each time step is computed by numerical minimization of the predicted control error. The stability issues and the model mismatch are also discussed. The connection with the IMC design is also analysed.

Section 4.3 presents the dual network control approach. A neural network is used as a controller. The problems related to the model inverse based IMC design are partially avoided by employing a nonlinear optimal controller within the IMC structure. The nonlinear control law is approximated by a perceptron network and the cost function associated to the optimal controller design is minimized numerically. The stability of the resulting control system is discussed and practical methods for maintaining the stability are introduced.

Chapter 5 presents results of the simulation studies demonstrating the properties of the proposed neural network identification and control approach. Also the neuro-control workstation based on a HP9000/425 platform is introduced including the software tools for identification, simulation and controller design. The system forms a flexible and efficient tool for identification of nonlinear process models, for development of control systems and for experimental studies.

Chapters 6, 7 and 8 present the experimental results obtained by applying the proposed approach to control of several laboratory scale pilot processes. Chapter 6 considers the real-time control of two small water heating processes. Both direct predictive control and the dual network control are considered. Chapter 7 considers the identification and direct predictive control of a small air heating process. Chapter 8 presents the experimental results obtained applying the multivariate direct predictive control to a pilot head box process of a paper machine.

Contributions

The full nonlinear model based control approach was applied. In this sense it is rather irrelevant that only mildly nonlinear processes were considered. They incorporate many of the features which commonly make control difficult: delays, noise, deterministic disturbances, trends, time varying features etc. Solving the problems associated to these in conjunction of neural networks, both from the theoretical and practical point of view, is the main contribution of the thesis.

A contribution is the development of advanced identification and control design methods, especially for recurrent networks. This work has been made in parallel with other researchers and the use of neural networks as nonlinear time series models and in model based control is nowadays an established field.

The author also has a contribution in this development: guidelines for neural networks as time series models an and application to one-step-ahead predictive control (Koivisto, 1990), an application to Long Range Predictive Control, even with real time experiments (Koivisto, Kimpimäki and Koivo, 1991a,b), and a RPEM based dual network IMC approach (Koivisto, Ruoppila and Koivo, 1992). Identification in presence of trend type of disturbances and identification of multistep-ahead predictors are novel contributions.

A detailed analysis of stability and extrapolation issues resulting in general guidelines and practical methods for maintaining the stability and generalization, and the successful application within the experimental case studies are a novel contributions.

A contribution is also the analysis of the differences and the robustness issues of the LRPC approach using direct predictive models or deterministic recurrent models (IMC approach), and between different implementations of the IMC approach.

Finally, a major contribution is of course that the approach as a whole was put into practice. The experimental results show good performance characteristics and robust control.

2 Neural Networks as Function Approximators

This chapter considers the use of neural networks for approximation of nonlinear functions. A review of existing neural network types is presented and their suitability for function approximation is analysed, especially from a point of view of real time usage and time series modelling.

2.1 Neural Networks

As mentioned earlier, almost 50 different artificial neural network architectures have been developed, although only a part of them is in common use. An overview of different neural network architectures can be found in numerous books, see for example Hertz *et. al.* (1991).

Some of the most common neural networks are: functional mapping nets like multilayer perceptron (MLP) and cerebellar model articulation computer (CMAC) (Albus 1975), adaptive resonance nets (ART) that form input categories from input data, input feature categorization nets of Kohonen (Kohonen, 1984), bilinear associative memories and feedback network of analog neurons (Hopfield 1985). An interpolation technique called Radial Basis Functions (RBF) (Moody and Darken, 1989, Poggio and Girosi, 1990), is normally viewed as a neural network. Fuzzy models are also an efficient function approximation scheme.

The difference between “neural networks” and other approaches is frail because they all apply established statistical methods for classification and function approximation tasks and they are commonly implemented as conventional computer programs.

Neural networks are here considered only from the point of view of function approximation. This requires interpolation, or more generally approximation of the function between the presented data points. This feature is called *generalization* in the neural network terminology.

Most of the neural network architectures are initially motivated by pattern recognition and associative memory tasks. Most of the classification networks can also be used or can be modified for approximating purposes. They categorize directly the input vector space or extract similarities (features) from the input space (Kohonen net, ART, some RBF variants, Principal Component nets (PCA by Oja 1989). The actual functional mapping is approximated separately for each cluster or feature which is relatively easy task compared to the overall approximation of the particular function. This means that the approximation of

a complex function is transferred to a clustering problem which in turn can be quite complex.

For example, Hyötyniemi (1994), and Fox and Heinze (1991) present demonstrations of Kohonen network as (a part of) function approximator. Similar results using ART-2 network is presented by Sørheim (1990). The upper layer consist of several MLP networks, one for each cluster / feature.

On the other hand, some neural networks are directly applicable to function approximation, namely MLP networks, most RBF variants, CMAC and BMAC (B-spline CMAC, *e.g.* Lane *et. al.* 1992). These have successfully been applied to representation of complex nonlinear functions.

Multilayer perceptron is the most used and studied neural network architecture today. It constructs a *global* approximation of a multi-input multi-output function in a similar manner as fitting of a low order polynomial through a set of data points. A rich collection of different learning paradigms has been developed.

CMAC/BMAC and most RBF networks construct a local approximation of multi-input multi-output function analogous to fitting least squares splines through a set of data points using piecewise constant functions (CMAC) or other basis functions (RBF and BMAC).

Neural network architectures have different features and the suitability of particular architecture depends on the application type. When selecting a network type for function approximation, a compromise between several desired features must be made. Some of these are:

- local or global approximation,
- accuracy and generalization capability,
- memory usage, computational load, parallel implementation,
- identification method (speed of convergence, etc.),
- suitability for recurrent use (see Section 2.2),
- on-line specific features (suitability for on-line identification, etc.)

The last two items has a special importance when considering representation of nonlinear time series models. These aspects will be studied in the next section. Special emphasis is paid to real-time specific features.

2.2 Global Function Approximation

The overall goal is to study nonlinear dynamical systems and a general distinction between two different function approximation schemes must be made. The basic functional mapping is of the form

$$y = F(x) \tag{2.1}$$

where $F : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$ is a mapping between an input vector x and output vector y. The other type is a nonlinear difference equation

$$x(t+1) = F(x(t)) \tag{2.2}$$

where the state vector $x(t+1)$ at step $t+1$ is a function ($F : \boldsymbol{R}^m \rightarrow \boldsymbol{R}^m$) of the previous state $x(t)$. These two types and their combinations form the basic building blocks of nonlinear time series models. In both cases there is a functional mapping F which is approximated with a neural network. These are termed as *feedforward* and *recurrent* [1] networks (Fig. 2.1). As will be seen, the recurrent nature of the functional mapping causes a corresponding recurrency to the identification algorithm.

In the neural network terminology the problem of approximating a function is often rephrased as a problem of learning from examples i.e. from data vector pairs

$$\begin{aligned} x(t) &= [x_1(t) \ldots x_n(t)]^T \\ y(t) &= [y_1(t) \ldots y_m(t)]^T \end{aligned} \quad , \qquad t = 1 \ldots N \tag{2.3}$$

which will be called the *training set*. The index t can denote just a sample index or time in time series. The behaviour of the network with the training set does not tell anything about the generalization capability outside the trained region. The approximation should be verified also using a separate *test set*. The term overlearning is used in this context to

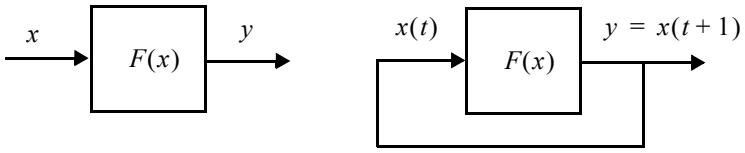


Fig. 2.1. Direct (feedforward) and recurrent functional mapping.

1. The reader should make a distinction between recurrent and recursive. The latter is here used to mean on-line specific features like recursive identification.

mean that the resulting network estimates the trained examples too well and loose its generalization capability.

Training can be made using incremental (*recursive*) minimisation of the selected cost function or using *batch* learning where the cost function over the whole training set is computed before each parameter update pass. The batch learning can be used only as an off-line method, while recursive learning can be used both as an off-line or an on-line method. On-line learning needs recursive methods.

Global Approximation Networks with Fixed Structure

A smooth at least C^1-continuous function $F : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$ can be approximated using some function $f : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$, which implements a global mapping. The prediction is

$$\hat{y}(t) = \hat{y}(t|\theta) = f(x(t), \theta) \tag{2.4}$$

between the input vector $x \in \boldsymbol{R}^n$ and the predicted output vector $\hat{y} \in \boldsymbol{R}^m$ of the true output y. The parameter vector $\theta \in \boldsymbol{R}^{n\theta}$ contains the parameters of the function. The global mapping means that the change of one parameter might affect to the predictions made in the whole input-output domain. The structure of the function f is prespecified and does not necessarily correspond to the true system structure. This type of mapping is termed as *restricted complexity* approximation, e.g. Goodwin and Sin (1984). A consequence of the restricted nature is that there may not exist any vector θ for which $\hat{y}(t) = y(t)$ $t = 1 \ldots N$. Instead one finds values of θ that minimize the approximation error according some selected cost function.

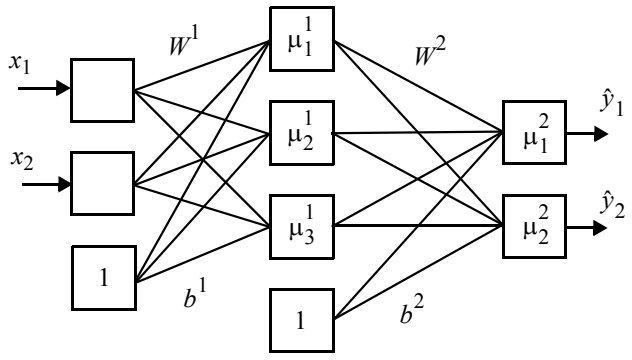


Fig. 2.2. Block diagram of the multilayered perceptron network with one hidden layer and $n = 2, m^1 = 3, m^2 = m = 2$.

Multilayer perceptron network (MLP) implements a global mapping in the sense of (2.4). It consists of input and output layers and one or more "hidden" layers. The block diagram of MLP with one "hidden" layer and $m^1 = 3$ nodes is presented in Fig. 2.2. The superscript denotes the layer number. The lower layer is connected to the upper one through a series of connections called weights (matrices W^1 and W^2). Also threshold connections (bias) are normally used (vectors b^1 and b^2 of appropriate dimension). Each layer performs a diagonal nonlinear mapping $\mu : \boldsymbol{R}^k \to \boldsymbol{R}^k$, where k is the number of nodes in the layer

$$\mu(v) = [\mu_1(v_1) \ldots \mu_k(v_k)]^T \tag{2.5}$$

where v is the input vector to the layer. The scalar activation function μ_j is usually some sigmoid shape function, for example

$$\mu_j(v_j) = 1/(1 + e^{-v_j}) \text{ or } \mu_j(v_j) = \tanh(v_j). \tag{2.6}$$

The outermost activation function can be a linear function, but typically the same nonlinearity is used for all nodes. The network in Fig. 2.2 implements a nested nonlinear function

$$f(x) = \mu^2(W^2\mu^1(W^1x + b^1) + b^2) \tag{2.7}$$

with diagonal activation functions $\mu^1 : \boldsymbol{R}^3 \to \boldsymbol{R}^3$ and $\mu^2 : \boldsymbol{R}^2 \to \boldsymbol{R}^2$.

The parameter vector θ contains the weights of the network (W^1, b^1, W^2, b^2), columns ordered to an $n_\theta = nm^1 + m^1 + m^1m + m$ dimensional vector. Also short-cut connections directly from the input layer to the output layer are commonly used. These provide a convenient way to implement a parallel linear model, if used with linear output activation function. For example with (2.7) one obtains

$$f(x) = (W^2\mu^1(W^1x + b^1) + b^2) + W^Lx \tag{2.8}$$

where W^L is a parameter matrix for the linear part of the function.

Hornik *et al*. (1989) presents a proof that a standard multilayer feedforward network architectures with one hidden layer and arbitrary monotonic activation functions can approximate virtually any smooth and continuous function of interest to any desired degree of accuracy, provided that sufficiently many hidden units are available. In practice, the proof is not very useful, because real networks can not be "sufficiently large" and the restricted complexity assumption must hold.

However, Sontag (1990) presents a proof that for certain problems two hidden layers are required, contrary to what might be in principle expected from the known approximation theorems. The differences are not based on numerical accuracy nor on capabilities for fea-

ture extraction, but rather more basic classification into “direct” and “inverse” problems. The former corresponds to the approximation of continuous functions, while the latter is concerned with the approximation of one-sided inverses of continuous functions.

If large number of weights is used, there is a strong tendency to overfit the data and to perform poorly on unseen data. Methods for structure selection are nowadays available, but they are computationally expensive. In practice the structure of the network is selected by a heuristic trial-and-error method using networks of increasing complexity to minimize the selected cost function and using the test set to monitor the generalization.

The values of the weights are determined during the identification phase. A rich set of different optimization methods have been applied as training methods, like gradient based methods, simulated annealing and genetic algorithms. Most of these are suitable only for batch learning. Only gradient based optimization methods are used in this study, both for batch and on-line learning. In the batch identification, the quadratic cost function

$$V_N(\theta) = \frac{1}{2}\sum_{t=1}^{N} \left\| y(t) - \hat{y}(t|\theta) \right\|^2 \tag{2.9}$$

is minimized iteratively with respect to the parameter vector θ. The $\hat{y}(t)$ is the prediction of the output $y(t)$ for the sample input $x(t)$ according (2.4). In the recursive case the quadratic cost function

$$V(t, \theta) = \frac{1}{2}\sum_{i=1}^{t} \left\| y(i) - \hat{y}(i|\theta) \right\|^2 \tag{2.10}$$

is minimized at each step t, i.e

$$\hat{\theta}(t) = \arg\min_{\theta} V(t, \theta) \tag{2.11}$$

Gradient based minimization methods can be divided in two category:

- general nonlinear optimization methods using values of the scalar cost function, its gradients and possibly its Hessian (the second order derivative matrix).
- nonlinear least squares approach, which can be used only for quadratic cost functions, like (2.9).

At iteration count k the Taylor expansion of the cost function (2.9) around the estimate $\hat{\theta}(k-1)$ gives the well-known Newton's update equation, e.g. Ljung (1987)

$$\Delta\hat{\theta}(k) = -\eta[V''_N(\theta)]^{-1}\,[\,V'_N(\theta)]^T\Big|_{\theta = \hat{\theta}(k-1)} \tag{2.12}$$

where $\Delta\hat{\theta}(k)$ is the parameter update and $\eta \in (0, 1]$ is the step length, which is selected so that a decrease in the cost function is obtained.

On a very general level this approximation is a basis for all gradient based minimization methods, batch or recursive. Different algorithms are obtained depending on the approximation of the Hessian in (2.12). For example, if the Hessian is approximated with a diagonal matrix $\eta \cdot I$, the steepest descent minimization algorithm is obtained.

An important and useful feature of MLP networks is that the gradient of the (2.9) and various other Jacobians of the model (2.4) can be computed efficiently in a hierarchical and parallel manner using the chain rule differentiation. This gradient computation method is called the *generalized delta rule* or *error backpropagation*, e.g Rumelhart *et al*. (1986). This is commonly considered as a minimization method, i.e it includes also the steepest descent parameter update. The terms *forward pass* and *backward pass* are used in this study to denote the computation of the predictions *(forward*) and the gradients *(backward*) only. A detailed description of these algorithms is presented in Appendix A. The recursive and batch type nonlinear least squares approaches are presented in detail in Section 3.4.

Variable metric methods are considered the most efficient general nonlinear (batch) optimization methods. Most common are, e.g. Fletcher *et al.* (1990)

- Broyden-Flecther-Goldfarb-Shanno (BFGS)
- Davidon-Fletcher-Powell (DFP)

Common nonlinear least squares methods are:

- Levenberg-Marquardt (LM)
- Gauss-Newton (GN)

Implementations of these are generally available, for example:

- Matlab Optimization Toolbox (BFGS, DFP, LM, GN)
- Matlab Neural Network Toolbox v2 (LM)
- C- and Fortran-codes: Numerical Recipes (inc. software), IMSL, NAG

Adaptive applications require recursive methods:

- Recursive Error Backpropagation
- Recursive Prediction Error Method (RPE)

The RPE method is a variant of recursive Gauss-Newton algorithm, for details see Chen, Billings and Grant (1990) or Koivisto, Ruoppila and Koivo (1992). The gradient computation according to the cost function is the main effort when applying all these methods. The benefits of these advanced methods are faster adaptation and, more important, better results. The methods are computationally heavier, mainly nonparallel and require some level of expertise to implement.

2.3 Local Function Approximation

Local function approximation methods are alternatives to global ones. There are several approaches for the local approximation. There is no doubt that a local approximation is an efficient method for low dimensional tasks (input vector dimension $n = 1 \ldots 3$). For higher dimensional tasks several difficulties are encountered and the suitability depends heavily on the application. All methods are in general implementations of the spline approach. Most common methods in this category are: RBF networks, CMAC/BMAC networks, normalized B-splines and most fuzzy models. The purpose is not to present a detailed description of local function approximators and only the general properties are overviewed in this section.

An analogy can be seen between learning an input-output mapping and a surface reconstruction from sparse data points $(y(t), x(t))$, $t = 1 \ldots N$. In this sense learning is a problem of hypersurface reconstruction, e.g. Poggio and Girosi (1990) that can be mathematically formulated as a selection of a hypersurface f that solves the variational problem of minimization of the functional ($y(t)$ scalar for notational simplicity)

$$V[f] = \sum_{t=1}^{N} \| y(t) - f(x(t)) \|^2 + \lambda \| Pf \|^2 \tag{2.13}$$

The second term measures the cost associated with the deviation from smoothness. The stabilizer P is usually a differential operator and the regularization parameter λ controls the compromise between the degree of smoothness of the solution and its closeness to the data. If P is an operator with radial symmetry, the solution of (2.13) has following simple form, e.g. Poggio and Girosi (1990)

$$f(x) = \sum_{j=1}^{l} B_j \mu_j(\| x - \xi(j) \|_\Lambda) \tag{2.14}$$

where B is a l-dimensional coefficient vector and $\xi(j)$ is an n dimensional vector which corresponds to the center of the jth radial function $\mu_j(\cdot)$. The function $f(x)$ is a weighted sum of l radial functions, each with its own center $\xi(j)$. In general the norm should be separate for each radial function i.e.

$$\| x - \xi(j) \|^2_{\Lambda(j)} = (x - \xi(j))^T \Lambda(j) (x - \xi(j)) \tag{2.15}$$

where $\Lambda(j)$ is an $n \times n$ dimensional weighting matrix, one for each radial function. Typically the simplest solution is used: one common diagonal matrix $\Lambda(j) = \Lambda = \delta \cdot I$. It is obvious that this type of $\Lambda(j)$ reduces greatly the efficiency of the approach.

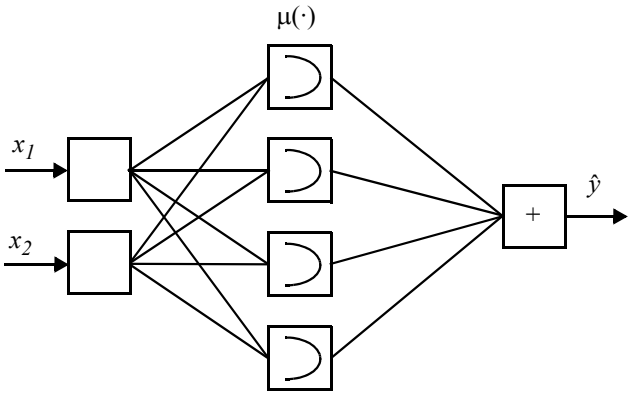


Fig. 2.3. Block diagram of the radial basis function network.

The use of weighting and selection of the radial function varies significantly. One clear difference is whether an n-dimensional radial function is used or whether it is formed as a cartesian product of n one-dimensional "membership functions". A common radial function is the Gaussian function

$$\mu_j(z) = e^{-z^2} \tag{2.16}$$

where z is the distance according to (2.14) or (2.15). The Gaussian function produces a local mapping i.e. $\mu_j(z) \to 0$ as $z \to \infty$. It is commonly used as an n-dimensional radial function although it can be used also as a one-dimensional radial function. A more common one-dimensional radial function is a linear membership function (triangle)

$$\mu_j(z) = \begin{cases} 1 - a_j z, & \text{if } (a_j z \le 1) \\ 0, & \text{otherwise} \end{cases} \tag{2.17}$$

where a_j is a constant tuning parameter. This type of function is typical in fuzzy models.

Consider first the n-dimensional radial function. If $l = N$, the centers are on distinct data points and (2.14) is equivalent to generalized splines with fixed knots, leaving the coefficients B_j and the weight $\Lambda(j)$ to be determined. This approach can also be used for interpolation between prespecified local models (here the constants B_j) leaving only $\Lambda(j)$ to be determined, e.g. Johanssen (1994).

A common application is to identify directly a global function approximator from the data according (2.14), i.e. as a weighted sum of local radial functions; known as Radial Basis Function (RBF) network, see Fig. 2.3. In this case $l \ll N$ and the centers $\xi(j)$ are unknown, which is the main difficulty when applying this approach. This is commonly known as a clustering task. Several methods for off-line clustering are available from the

simple k-means clustering (Moody and Darken 1989) up to the Kohonen network.

Chen *et al.* (1991) introduced an advanced method for off-line training for RBF networks: Orthogonal Least Squares (OLS) learning. The algorithm selects best l regressors (centers in fact) using modified Gram-Schmidt orthogonalization with prespecified and common Λ. The widths $\Lambda(j)$ of the resulting RBF model can then be refined with gradient based minimization. This and other similar methods results in good performance, comparable to the MLP network.

For example a self-organizing neural network (SONN) by Tenorio (1990) constructs a network, chooses the node functions and adjusts the weights. The rule for the node selection is based on a variant of Rissanen's Minimal Description Length (MDL, Rissanen, 1978) information criterion, which provides a trade-off between the accuracy and the complexity of the model.

Although OLS, SONN and other reviewed self-organising systems produce good results, they cannot be used recursively for on-line learning, but only for batch learning. There exist only a couple on-line self-organizing clustering methods.

The well known method of the movable centers, e.g. Moody and Darken (1989) is one possibility, although it cannot be considered as a self-organising one. This method does not produce good results.

The dynamically capacity allocating (DCA) network (Jokinen 1991) is suitable for on-line learning due to its self-organizing and instant learning capability. It is also instantly forgetting and is suitable only for plain predictive purposes, not for modelling recurrent systems.

Modern workstations can store relevant amount of data directly in memory and obviously more efficient on-line clustering methods will be seen in near future.

Cartesian Product Approximator

One way to avoid the clustering task is to use an n-dimensional fixed grid for the region of the interest. The whole operation region is divided into “boxes” each containing its own local model (a constant as the simplest case) which are combined so that continuity and smoothness are achieved (at least should be).

This clustering is commonly done by defining a radial function / membership function separately for each input dimension and combining these to an n-dimensional basis function using the tensor product. The actual function approximation is then obtained as a weighted sum of these basis functions.

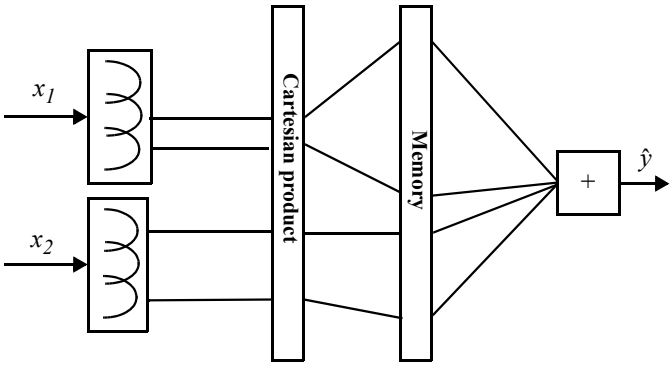


Fig. 2.4. A cartesian product local approximation network.

Common approximators in this category are:

- multidimensional normalized B-splines, e.g. de Boor (1978)
- CMAC, e.g. Albus (1975)
- BMAC (B-spline CMAC), Lane *et al.* (1992)
- most fuzzy models
- RBF network with fixed grid as centers.

The block diagram of this network type is presented in Fig. 2.4. The CMAC network is a typical example of this approach, although it uses binary valued basis functions. The cartesian product requires huge amount of memory in a high dimensional case. Because in practice only a small portion of possible combinations are needed, rehash techniques are used to map the needed memory into much smaller physical memory, e.g. Albus (1975).

Due to the binary basis function, the CMAC does not produce smooth approximations. Higher order basis functions can be used to produce smooth approximations and continuous derivatives (BMAC), but if applied to high dimensional case, the number of parameters in the cartesian product "explodes".

Another version of CMAC which produces smooth approximation is the Associative Memory System (AMS), e.g. Tolle and Ersu (1992), which has been successfully used for on-line learning control within several processes.

The analogy between the approaches can be demonstrated considering the approximator

$$f(x) = \left[\sum_{j=1}^{l} B_j \mu_j(z_j)\right] \Big/ \left[\sum_{j=1}^{l} \mu_j(z_j)\right] \qquad (2.18)$$
$$z_j = \| x - \xi(j) \|_{\Lambda(j)}$$

with triangular μ_j. The basis functions are normalized so that the sum of basis function values at each point is equal to one. This can be viewed as

- a fuzzy model with a certain (most common) defuzzifier
- a first order normalized B-spline
- a special type of RBF network.

This type of localized B-spline approximation is efficient if the input dimension is low. This can be seen from the vast amount of the articles considering fuzzy models and controllers. A minor note is that the use of prespecified fixed grid (fixed knots) does not produce optimal approximation even in one dimensional case.

All local approximators without clustering are suitable for on-line learning. The high dimensional case or the use of very dense grid makes the learning process slow and reduces the extrapolation capability.

2.4 Global vs. Local Models

The main practical difficulty in identification is the lack of good quality data, the measurements do not cover all operation regions of interest. It is a major problem if the model will be used for predictive control. This is typically a model inversion task which is solved iteratively using the model output and the Jacobian of the model w.r.t its input vector. The problem is that the outputs and the Jacobian are often needed also outside the trained region. The prediction is obviously inaccurate outside the trained region but the signs of the Jacobian must still correspond to those of the true process in order to avoid wrong local minima.

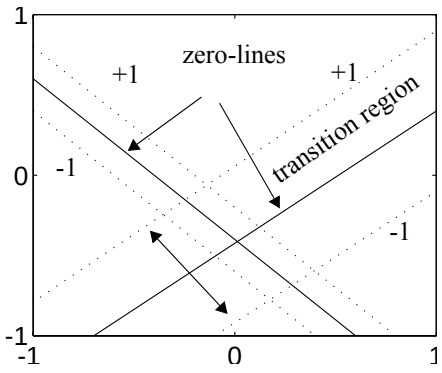


Fig. 2.5. The first hidden layer of the MLP network divides the input space into subre-

Consider first a global function approximator, a MLP network with one hidden layer. The hidden layer produces the activations according to (2.7)

$$p = \tanh(W^1 x + b^1) \tag{2.19}$$

These activations are zero if

$$W^1 x + b^1 = 0 \tag{2.20}$$

This means that the input space is divided into subregions by the hyperplanes defined by (2.20). Near these zero-hyperplanes is a transition region which corresponds to the sigmoid shape of the activation function. The actual model output is a weighted sum of the values of the activations (2.19). A two-dimensional example of these hyperplanes is presented in Fig. 2.5. If the data do not cover the whole region of interest (Fig. 2.6a), the MLP network extrapolates. There is no guarantee that even the sign of the Jacobian w.r.t. the input vector is correct outside the trained region, in fact it can be anything.

The localized function approximator behaves even worse (Fig. 2.6b). When extrapolating, the localized model typically produce the output or at least the corresponding Jacobians ≈ 0. These predictions and Jacobians are anyway needed and serious problems are encountered during the model inversion.

Precautions with both model types must be taken into account to obtain correct results. Ensuring (projecting) the signs of the Jacobian w.r.t the input vector is one possibility to avoid divergence. Straightforward methods for the global approximator will be presented in Section 3.4.

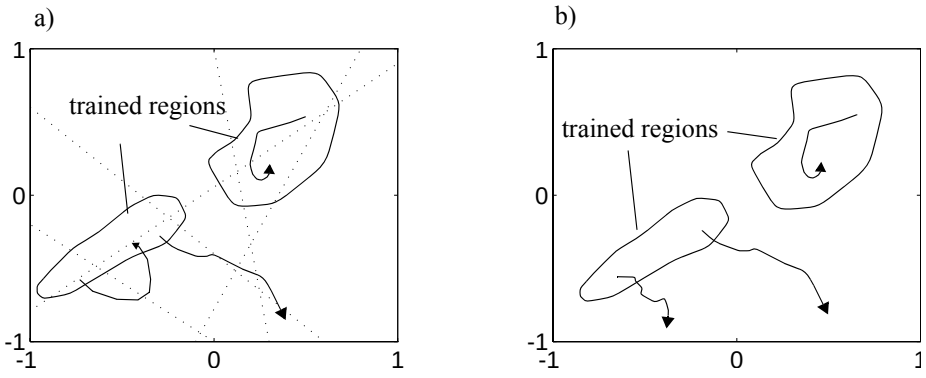


Fig. 2.6. Extrapolating outside the trained region could cause a divergence when inverting the model iteratively, a) MLP network, b) localized approximator.

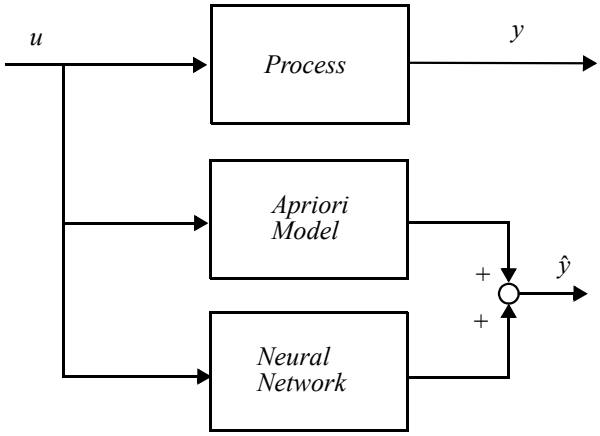


Fig. 2.7. The neural network as an additive process model.

For localized approximators two possibilities for this projection is:

- Ensure the Jacobian values (signs) around the borders of the trained region and ensure that only the trained region is used during the inversion.
- Use a separate and additive global model (Fig. 2.7) obtained using apriori knowledge or via identification. A crude approximation is enough, the signs of the Jacobians are important.

Using these guidelines also the localized approximator can be successfully applied with predictive control. However, the projection operation is more difficult than with the global approximator. Handling the projection within on-line identification is also difficult.

As discussed in Sections 2.2 and 2.3, there are several function approximation networks with different properties. Some of them are suitable for on-line learning, some for localized representation, and some for high dimensional problems. It seems that only global approximators can easily combine all necessary properties. The MLP network is selected for the basic building block to be used in this study although localized representations have some useful properties over the MLP network.

3 Modelling and Identification

Both theory and practice of nonlinear system modelling have advanced considerably in recent years. Neural networks have a remarkable influence on this development. They offer an efficient alternative for modelling of complex nonlinear systems.

This chapter presents a framework for modelling and identification of nonlinear dynamical systems. The multilayer perceptron neural network is used as the basic building block when representing the nonlinear time series models.

3.1 Nonlinear Stochastic Models

Many processes in practice are continuous and models for these are commonly derived from basic principles like mass and energy balances resulting in nonlinear continuous time models. A proper assumption, as normally done in identification is that a discrete time nonlinear system with a fixed unknown structure and constant unknown parameters exists. This assumption is applied throughout this study which means that it is also assumed that an originally continuous time process is discretizable. Also in the continuous case the system is modelled with a discrete time nonlinear model and controlled with a discrete time nonlinear controller.

It is well-known that a nonlinear system can be described by a nonlinear time series model involving nonlinear regression of past data. As indicated by Priestley (1988), a general model of a discrete time noise process takes the form

$$y(t) = \tilde{H}(v(t), v(t-1)\ldots, v(t-k), \ldots) \tag{3.1}$$

where $y(t)$ is a scalar measurement at time t and $\{v(t)\}$ is a white noise sequence. Equation (3.1) represents a general non-anticipative (causal) nonlinear model involving infinite dimensional nonlinear regression of past data. If the $\tilde{H}$ is assumed to be sufficiently well-behaved, it can be expanded in a Taylor series about some fixed point - say $\mathbf{0} = (0, 0, 0, \ldots)$, resulting in

$$y(t) = y_0 + \sum_{i=0}^{\infty} c_i\, v(t-i) + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} c_{ij}\, v(t-i)v(t-j) + ... \tag{3.2}$$

with coefficients $c_i, c_{ij}, \ldots$ obtained from expansion and $y_0 = \tilde{H}(\mathbf{0})$. This expansion is known as a (discrete time) *Volterra series* and it provides an important type of representation for nonlinear models.

The corresponding infinite dimensional nonlinear regression model with a deterministic scalar input sequence $\{u(t)\}$ is

$$y(t) = \tilde{G}(u(t), u(t-1), \ldots, u(t-k), \ldots, v(t), v(t-1), \ldots, v(t-k), \ldots) \qquad (3.3)$$

where $\tilde{G}$ is assumed to be smooth and continuous so that a Volterra series expansion can be derived. The sequences $\{y(t)\}$ and $\{u(t)\}$ are now assumed at least as quasi-stationary bounded sequences (Ljung 1987).

Obviously and as indicated by Billings and Voon (1986), this nonlinear model (3.3) cannot be presented as a deterministic part and a separate noise model. Three blocks are needed: the deterministic part $\tilde{G}^u$, the noise model $\tilde{G}^v$ and the block $\tilde{G}^{uv}$, which represents an interaction between the input u and the noise v (see Fig. 3.1). Depending on the actual physical system some blocks might be missing. This presentation is of practical importance in model validation using correlation tests, e.g. Billings and Voon (1986).

As it stands, (3.3) is infinite dimensional in a sense that it involves a relationship between infinitely many variables. This can be reduced to a finite dimensional form by assuming that the relationship between $y(t)$ and the past history of series can be described in terms of finitely many values of past $\{y(t)\}$ and of past and present $\{u(t)\}$ and $\{v(t)\}$. Assuming also that no direct dependence between $u(t)$ and $y(t)$ exists, one may write

$$y(t) = G(Y^{(t-1)}, U^{(t-1)}, V^{(t)}) \qquad (3.4)$$

where

$$Y^{(t-1)} = [y(t-1)\ldots y(t-l_y)]^T$$
$$U^{(t-1)} = [u(t-1)\ldots u(t-l_u)]^T$$
$$V^{(t)} = [v(t)\ldots v(t-l_v-1)]^T$$

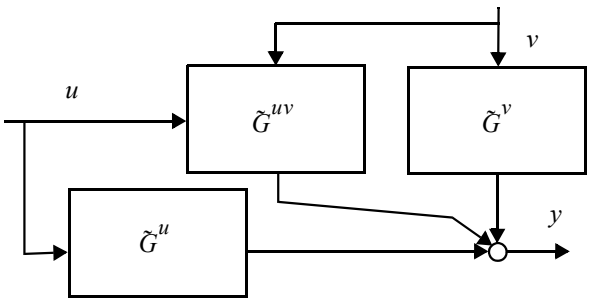


Fig. 3.1. A general nonlinear stochastic dynamical system can be described by three separate parts.

Here G is assumed to be a smooth continuous function and parameters l_y, l_u and l_v determine the amount of the past history needed. This notational simplification is used hereafter to denote a functional relationship between $y(t)$ and *finitely* many values of past history. This is used to emphasize the *variables* involved not the actual number of past values needed.

In the nonlinear system model (3.4) the noise may enter the system internally and cannot always be translated to be additive. In this case there is no simple way to obtain a one-step-ahead prediction $y(t+1|t)$ for the measurement $y(t+1)$ (at time t). If the noise is additive, the model can be presented as

$$y(t) = G(Y^{(t-1)}, U^{(t-1)}, V^{(t-1)}) + v(t) \tag{3.5}$$

where G is not same as G in (3.4). Also $v(t)$ is normalized to have unit coefficient. Now the one-step-ahead predictor can be derived, if the function G and the structure of the regression vectors are known. At time t the best prediction $\hat{y}(t+1|t)$ for the output $y(t+1)$ is the one which predicts the future noise $v(t+1)$ to be zero, resulting in

$$\hat{y}(t+1|t) = G(Y^{(t)}, U^{(t)}, V^{(t)}) \tag{3.6}$$

At time $t+1$ the idealized prediction error $\varepsilon(t+1)$ is

$$\varepsilon(t+1) = y(t+1) - \hat{y}(t+1|t) = v(t+1) \tag{3.7}$$

If the prediction and the estimation of $v(t)$ are made recursively and the system (3.6) is (globally asymptotically) stable, the predictor converges to a quasi-stationaty state, to the heuristically best available predictor. The term stable is here used in rather loose context. More precise definitions will be given in Section 3.4. Also the assumption of quasi-stationary sequences includes an assumption of some sort of stability. A combined representation of this one-step-ahead predictor is

$$\begin{aligned} \varepsilon(t) &= y(t) - \hat{y}(t|t-1) \\ \hat{y}(t+1|t) &= G(Y^{(t)}, U^{(t)}, E^{(t)}) \end{aligned} \tag{3.8}$$

where $E^{(t)} = [\varepsilon(t) \ldots \varepsilon(t - l_\varepsilon - 1)]^T$ denotes finitely many (l_ε) values of the prediction error sequence $\{\varepsilon(t)\}$. As mentioned before, it is not in general possible to obtain a simple closed form for the optimal predictor of the output of a nonlinear system (3.4) . A sensible approach is to seek the best predictor of a given structure, a *restricted complexity predictor* e.g. Goodwin and Sin (1984) and to *assume* the noise to be additive:

$$\begin{aligned} \hat{y}(t+1|t) &= f(\varphi(t+1), \theta) \\ \varepsilon(t) &= y(t) - \hat{y}(t|t-1) \end{aligned} \tag{3.9}$$

where $$\varphi(t+1) = [y(t)\ldots y(t-n_a+1)\ u(t)\ldots u(t-n_b+1)\ \varepsilon(t)\ldots\varepsilon(t-n_c+1)]^T$$

where f is a function with prespecified (selected) structure and constant unknown parameters θ, which are to be identified. The number of past values (model orders n_a, n_b and n_c) in the input data vector $\varphi(t+1)$ should correspond to the true system structure.

This restricted complexity approach can be clarified using the state space point of view. Assume that the single input single output (SISO) system model (3.5) is presented in nonlinear state space form

$$\begin{aligned} x(t+1) &= \Phi(x(t), u(t), \theta_1, v_1(t)) \\ y(t) &= \Gamma(x(t), \theta_2, v_2(t)) \end{aligned} \tag{3.10}$$

where Φ and Γ are known smooth continuous functions with constant parameters θ_1 and θ_2. The sequences $\{v_1(t)\}$ and $\{v_2(t)\}$ are independent white noise sequences. The system is also assumed to be globally asymptotically stable. Assume first that the model parameters θ_1 and θ_2 are known. Referring to the Extended Kalman Filter, e.g. Goodwin and Sin (1984), the one-step-ahead predictor is sought in the form

$$\begin{aligned} x(t+1) &= \Phi(x(t), u(t), \theta_1, 0) + K(\theta_3)(y(t) - \hat{y}(t|t-1)) \\ \hat{y}(t+1|t) &= \Gamma(x(t+1), \theta_2, 0) \end{aligned} \tag{3.11}$$

where K is a selected function with parameters θ_3. In general case the determination of the parameters θ_3 is clearly an identification problem. Note that an additive noise assumption is made and the predictor does not necessarily produce optimal predictions but the best predictions for this (true and known) structure. If only the structure of the functions are known, the situation is the same leaving the parameters θ_1, θ_2 and θ_3 to be determined.

In practice the functions Φ and Γ are not known and they must also be approximated. In that sense the predictors (3.9) and (3.11) are identical. The nonlinear time series model approach (3.9) is clearly simpler and more suitable for predictor identification.

The prediction error approach (Section 3.3) will be used as a parameter estimation tool to identify the unknown parameters θ. The advantage of the prediction error algorithm is that it will provide (locally) the best predictor for a given structure.

Various predictor structures are presented in Section 3.2. If there is apriori knowledge about the noise model structure, the predictor and the identification task might be remarkably simplified.

Structures for Nonlinear Time Series Models

Multilayer perceptrons are selected as the basic building block to represent the nonlinearities in the time series models, because they can approximate virtually any smooth continuous function. Also conventional solutions can be useful, if the general form of the nonlinearities is known in advance. Most conventional models presume some prefixed nonlinear structure with unknown parameters, which are determined during the identification phase.

As indicated before e.g. Priestley (1988) or Monaco and Norman-Cyrot (1986) any nonlinear system can be under certain conditions represented as a corresponding series such as the Volterra series. When the nonlinearity of the true system is unknown, one possibility is to some extent approximate the Volterra series.

Another general (deterministic) representation is Feedbacked-Uryson model (e.g. Vadstrup, 1988)

$$\begin{aligned} A_1(q^{-1})y(t) = & \sum_{k=1}^{l} B_k(q^{-1})[f_k(u(t-d))] \\ & + \sum_{k=1}^{j} A_k(q^{-1})[g_k(y(t))] + c \end{aligned} \tag{3.12}$$

where finite length polynomials in q^{-1} are

$$\begin{aligned} A_1(q^{-1}) &= 1 + a_{11}\, q^{-1} + \ldots \\ A_k(q^{-1}) &= a_{k1}\, q^{-1} + a_{k2}\, q^{-2} + \ldots \qquad k = 2 \ldots j \\ B_k(q^{-1}) &= b_{k0} + b_{k1}\, q^{-1} + b_{k2}\, q^{-2} + \ldots \qquad k = 1 \ldots l \end{aligned}$$

and d is the delay. The functions f_k and g_k must be selected using apriori knowledge. If the functions are selected as polynomials, the well-known Hammerstein model is obtained as a special case of (3.12). Other possibilities are presented for example in the survey paper of Haber and Unbehauen (1990).

This type of model is useful for identification and adaptive control (e.g. Vadstrup 1988), because it is linear with respect to the parameters (linear-in-the-parameters). The key question is whether the selected nonlinearity corresponds to the structure of the true system. The multi-input multi-output (MIMO) case is also quite complex, as it can be seen from the Volterra series (3.2).

The multilayer perceptron network approach has also a fixed structure with unknown parameters, but so much expressing power that it can in principle represent any continuous function. The disadvantage is that MLP is not linear with respect to the parameters and more complicated identification schemes must be applied.

There are several possibilities to simplify the representation, if apriori knowledge about the type of the nonlinearity is available. For example Narendra and Parthasarathy (1990) characterize discrete time nonlinear models (in fact neural networks) with four different classes (assume a deterministic case with $d = 1$ for notational simplicity):

$$y(t+1) = \alpha(q^{-1})y(t) + g(U^{(t)}) \tag{3.13a}$$

$$y(t+1) = f(Y^{(t)}) + \beta(q^{-1})u(t) \tag{3.13b}$$

$$y(t+1) = f(Y^{(t)}) + g(U^{(t)}) \tag{3.13c}$$

$$y(t+1) = f(Y^{(t)}, U^{(t)}) \tag{3.13d}$$

where $\alpha(q^{-1})$ and $\beta(q^{-1})$ are linear polynomials, and f and g nonlinear functions. It is evident that model (3.13d) includes models (3.13a)...(3.13c). However, model (3.13d) is analytically the least tractable and hence for practical applications some other models might prove more attractive. Also the bilinear model

$$y(t+1) = f(Y^{(t)}, U^{(t-1)}) + g(Y^{(t)}, U^{(t-1)})u(t) \tag{3.13e}$$

has attractive features from the control point of view, c.f. Slotine and Li (1991).

3.2 Nonlinear Predictors

Several nonlinear predictor types will be defined in this section. The principles and the naming convention used by Ljung (1987) for linear time series models is adopted. The general form of the one-step-ahead predictor (3.9) is simplified according to apriori noise model assumptions. Only square systems are consired in this study ($y \in \boldsymbol{R}^m$, $u \in \boldsymbol{R}^m$, $v \in \boldsymbol{R}^m$). The data regression vector $\varphi \in \boldsymbol{R}^n$, the original system $h : \boldsymbol{R}^{n_o} \rightarrow \boldsymbol{R}^m$, and the approximation $f : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$ are of course different for each case. Normally $n = n_o$, but for direct d-step-ahead predictors different dimensions are commonly used. The functions f and h are assumed to be at least C^1 continuous and the corresponding dynamical systems globally asymptotically stable. The predictors are defined in the SISO case ($m = 1$), but the extension to the MIMO case is straightforward (page 33).

Nonlinear ARX model (NARX)

The simplest way is to predict future outputs as a function of past measurements and inputs. This nonlinear extension of linear ARX model is termed as *Nonlinear AutoRegressive with eXogenous inputs* (NARX) model. The nonlinear system is assumed to be of the form

$$y(t) = h(Y^{(t-1)}, U^{(t-d)}) + v(t) \tag{3.14}$$

where h is a nonlinear function and d is the input delay. An example of this model is presented in Fig. 3.2. Note how the additive noise enters the system. The one-step-ahead predictor is sought in the form

$$\begin{aligned} \hat{y}(t+1|t) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [y(t)\ldots y(t-n_a+1), \\ &\qquad u(t-d+1)\ldots u(t-d-n_b+2)]^T \end{aligned} \tag{3.15}$$

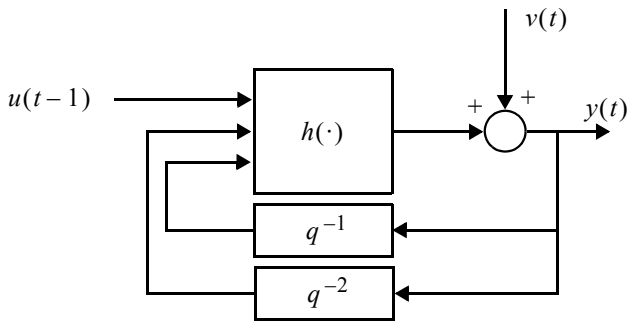


Fig. 3.2. An example of NARX model with $n_a = 2$, $n_b = 1$ and $d = 1$.

where $\hat{y}(t+1|t)$ is the prediction of $y(t+1)$ at time t. The function f has a selected structure with unknown parameters θ. Also the delay d and the model orders n_a and n_b must be determined during the identification phase.

The main advantage of NARX presentation is its simplicity and fast convergence of identification; faster than the predictor types presented in the sequel. Due to this reason it is often used even if the noise model assumption is not correct.

In linear case the controller is commonly designed assuming ARX model to represent deterministic part of process model using the *certainty equivalence principle*. The difference between a NARX model and a deterministic process model (NOE, see below) is conceptually much bigger than in the linear case, because the NARX model cannot be divided in two separate parts as its linear counterpart (Fig. 3.3). If the identified NARX model is used as a process model for controller design, one might get poor results.

If the model assumption (3.14) is correct, one normally obtains good results. A typical situation, where (3.14) does not hold, is the case, when the noise is additive at the output after the dynamical part. Even then one achieves reasonable *predictions*, a nice feature used in many predictive and adaptive algorithms.

If a d-step-ahead prediction $\hat{y}(t+d|t)$ is required, two alternatives exist. One can identify an one-step-ahead predictor (3.15) and use it recursively to obtain future predictions $\hat{y}(t+1|t)\ldots\hat{y}(t+d|t)$. The future predictions are used instead of the future measurements. The other alternative is to identify a direct d-step-ahead predictor i.e.

$$\begin{aligned} \hat{y}(t+d|t) &= f(\varphi(t+d), \theta) \\ \varphi(t+d) &= [y(t)\ldots y(t-n_a+1), \\ &\qquad u(t)\ldots u(t-n_b+1)]^T \end{aligned} \tag{3.16}$$

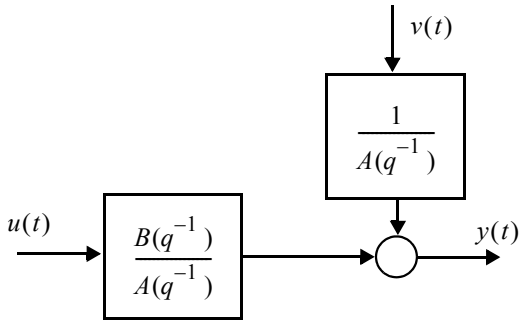


Fig. 3.3. Linear ARX models can be presented in two separate parts: the output additive (coloured) noise and the deterministic part.

In this case the predictor does not correspond to the deterministic process model even if the noise model is correct. Eq. (3.16) can efficiently be used for predictive and adaptive purposes, but not as a process model.

Nonlinear ARMAX model (NARMAX)

The nonlinear innovation model (3.5) will be termed as *Nonlinear AutoRegressive Moving Average with eXogenous inputs* (NARMAX) model. The nonlinear system is assumed to be

$$y(t) = h(Y^{(t-1)}, U^{(t-d)}, V^{(t-1)}) + v(t) \tag{3.17}$$

which is a nonlinear extension of the linear ARMAX model. The one-step-ahead predictor is sought in the form

$$\begin{aligned} \hat{y}(t+1|t) &= f(\varphi(t+1), \theta) \\ \varepsilon(t) &= y(t) - \hat{y}(t|t-1) \\ \varphi(t+1) &= [y(t)\ldots y(t-n_a+1), \\ &\quad u(t-d+1)\ldots u(t-d-n_b+2), \\ &\quad \varepsilon(t)\ldots\varepsilon(t-n_c+1)]^T \end{aligned} \tag{3.18}$$

An analogous representation can be formulated using previous predictions instead of the prediction errors. Now

$$\begin{aligned} \hat{y}(t+1|t) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [y(t)\ldots y(t-n_a+1), \\ &\quad u(t-d+1)\ldots u(t-d-n_b+2), \\ &\quad \hat{y}(t)\ldots\hat{y}(t-n_f+1)]^T \end{aligned} \tag{3.19}$$

where n_f is the number of previous predictions. If the system delay d is greater than one, the predictor equation can be formulated in several ways. See Goodwin and Sin (1984) for a lengthy discussion about indirect and direct d-step-ahead predictors for linear systems. One possible direct nonlinear d-step-ahead predictor is of the form

$$\begin{aligned} \hat{y}(t+d|t) &= f(\varphi(t+d), \theta) \\ \varphi(t+d) &= [y(t)\ldots y(t-n_a+1), \\ &\quad u(t)\ldots u(t-n_b+1), \\ &\quad \hat{y}(t)\ldots\hat{y}(t-n_f+1)]^T \end{aligned} \tag{3.20}$$

Note again that all functions f are different.

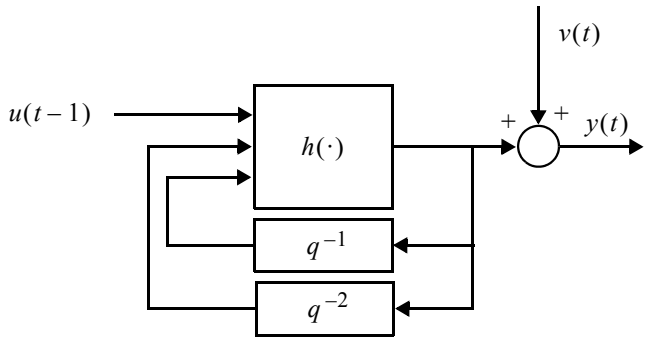


Fig. 3.4. An example of NOE model with $n_f = 2$, $n_b = 1$ and $d = 1$.

Nonlinear Output Error model (NOE)

The *Nonlinear Output Error* (NOE) model is a direct extension of the linear OE model and it is used for identifying the deterministic part of the system. The nonlinear system is assumed to be of the form

$$\begin{aligned} x(t) &= h(X^{(t-1)}, U^{(t-d)}) \\ y(t) &= x(t) + v(t) \end{aligned} \tag{3.21}$$

where $X^{(t-1)} = [x(t-1)\ldots x(t-l_x)]^T$. The noise is assumed to be output additive i.e. additive after the dynamical part (Fig. 3.4). The one-step-ahead predictor is presented in the form

$$\begin{aligned} \hat{y}(t+1|t) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\hat{y}(t)\ldots\hat{y}(t-n_f+1), \\ &\qquad u(t-d+1)\ldots u(t-d-n_b+2)]^T \end{aligned} \tag{3.22}$$

In practice the NARX and the NOE models are the most important representations of nonlinear systems. The NARX model is used mainly for predictive purposes and the NOE model is identified to obtain a model for simulation purposes i.e. the deterministic part of the system model. Of course the selection of the model type depends heavily on the true noise process.

Linear Additive Noise Model

Instead of the NARMAX predictor, one can make an assumption about a linear noise model. Now the system is assumed to be

$$y(t) = h(Y^{(t-1)}, U^{(t-d)}) + \frac{C(q^{-1})}{D(q^{-1})} v(t) \tag{3.23}$$

where $C(q^{-1})$ and $D(q^{-1})$ are polynomials in q^{-1} with appropriate dimensions. Similar assumption can be made with the NOE model. The system is now assumed to be

$$\begin{aligned} x(t) &= h(X^{(t-1)}, U^{(t-d)}) \\ y(t) &= x(t) + \frac{C(q^{-1})}{D(q^{-1})} v(t) \end{aligned} \tag{3.24}$$

The identification of predictors for these systems is not considered much in this study. A special case when $D = 1 - q^{-1}$ is studied in conjunction with predictive control (Chapter 4). In general there are two possibilities: adopt ideas used for identification of linear models, or represent a predictor as a sparse (MIMO) network with linear short-cuts.

Multi Input Multi Output (MIMO) case

The predictors defined above can be straightforwardly extended to MIMO case. For example if the nonlinear system is assumed to be

$$y(t) = h(Y^{(t-1)}, U^{(t-1)}) + v(t) \tag{3.25}$$

where $y \in \boldsymbol{R}^m$, $u \in \boldsymbol{R}^m$, $v \in \boldsymbol{R}^m$ $(m > 1)$ and $\{v(t)\}$ is an independent white noise sequence, the corresponding NARX predictor is

$$\begin{aligned} \hat{y}(t+1|t) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [y^T(t) \ldots y^T(t-n_a+1), \\ &\qquad u^T(t-d+1) \ldots u^T(t-d-n_b+2)]^T \end{aligned} \tag{3.26}$$

The only difference from the SISO case is that different delays can exist between the variables. Now the common delay d must be selected as the minimum delay of all loops and the model orders n_a and n_b as maximum orders of all loops. Because the function f is an MLP network, any delay structure can be implemented by removing a sparse network.

The general complexity of the delay structure selection in the MIMO case is analogous to that of the linear case. The selection is difficult especially if the predictor will be used for control purposes, but it is not a representation problem.

Multistep Prediction

The Long Range Predictive Control (LRPC, see Section 4.2) approach is based on the idea of reducing the controller sensitivity to high frequency noise through an objective function that minimizes the sum of squares of the control error over a finite time horizon (from $t+N1 \rightarrow t+N2$). This requires prediction of the measurement several steps into the future.

The qualitative form of the prediction task is

$$\begin{bmatrix} Predicted \\ Future \\ Outputs \end{bmatrix} = F(\begin{bmatrix} Known \\ Past \\ Data \end{bmatrix}, \begin{bmatrix} Future \\ Inputs \end{bmatrix}) \tag{3.27}$$

where F denotes some functional dependence. At time t the future outputs $y(t+i)$, $i = N1 \ldots N2$ are predicted as a function of available data, but also as a function of *future* inputs. There are three alternatives for this multistep prediction:

- recursive prediction using one-step-ahead predictor
- several separate direct i-step-ahead predictors, i=$N1$...$N2$
- one combined direct MIMO type predictor

The main interest in this study is on recursive prediction, because the resulting models can be used for other purposes like as a simulation model. A good reference for linear recursive multistep prediction is in Holst (1977). Weigend *et al.* (1990) compare combined neural network multistep prediction and recursive approach by identifying autoregressive models for predicting sunspot series and chaotic time series. The results favor recursive methods.

Implementation of recursive multistep prediction is straightforward, for example with a one-step-ahead NARX predictor (3.15)

$$\begin{aligned} \hat{y}(t+i|t) &= f(\varphi(t+i), \theta) \qquad\qquad , i = 1 \ldots N2 \\ \varphi(t+1) &= [y(t) \ldots y(t-n_a+1), \\ &\qquad u(t-d+1) \ldots u(t-d-n_b+2)]^T \end{aligned} \tag{3.28}$$

The future measurements are not known and they are replaced with the predicted ones. An example of a neural network as a recursive multistep predictor is shown in Fig. 3.5.

Due to the delay, the first $d-1$ predictions do not depend on present or future inputs and computational saving could be achieved, if the available but less accurate predictions like $\hat{y}(t+1|t-1)$ are used instead.

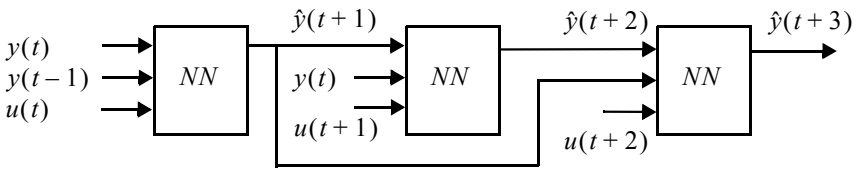


Fig. 3.5. A functional block diagram of the recursive multistep predictor implementation using neural network, *NN*s are identical copies of the predictor network, $n_a = 2$, $n_b = 1$, $d = 1$ and $N2 = 3$.

An example of a combined predictor is shown in Fig. 3.6 (same prediction task as in Fig. 3.5). The identification and the usage of this predictor seems to be straightforward. However, the predictor includes also non-causal correlations like $u(t+2) \rightarrow \hat{y}(t+1)$. The parameters (weights) corresponding to these noncausal correlations seem to exists, if adaptive control is used, i.e. they correspond to the controller. To avoid this, the neural network should have a sparse structure, which complicates the identification algorithm.

This type of combined neural multistep predictor within adaptive long range predictive control is used in Koivisto *et al.* (1991a,b) and Kimpimäki (1990). The results clearly indicate that the non-causal connections should be removed.

The properties of the recursive multistep predictor and the combined neural network predictor within the LRPC approach are discussed for instance by Saint-Donat *et al.* (1991), but with a full (non-sparse) network.

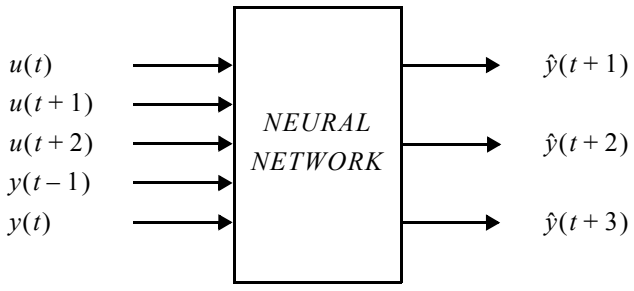


Fig. 3.6. A neural network as a combined multistep predictor with $N1 = 1$ and $N2 = 3$.

An one-step-ahead predictor is normally identified by minimizing the quadratic cost function

$$V_{ID} = \frac{1}{2}\sum_{t=1}^{N} \| y(t) - \hat{y}(t|t-1) \|^2 \tag{3.29}$$

with respect to the model parameters. If the model orders are too low or the noise model assumption is not correct, this predictor will not produce the best multistep predictions and consequently not the best overall control (LRPC). What is needed is an identification method that provides the suitable predictions in the range required by the controller. The effect of unmodelled dynamics in adaptive linear control is studied for example by Shook *et al.* (1991, 1992), Lu and Fisher (1990) and Lu *et al.* (1990). They propose the cost function

$$V_{LRPI} = \frac{1}{2}\sum_{t=1}^{N} \sum_{j=N1}^{N2} \| y(t) - \hat{y}(t|t-j) \|^2 \tag{3.30}$$

i.e. similar to that of the control scheme. This type of identification scheme is termed as identification for control in the literature.

Two approaches can be used to identify models, which minimize this identification cost function:

- minimize (3.30) numerically
- prefilter the data used for identification

These approaches have been shown to minimize the same cost function, when linear systems are considered, see Shook *et al.* (1991). Also, the corresponding prefilter can be determined easily. Prefiltering is not useful when identifying nonlinear models and the cost function (3.30) must be minimized numerically.

3.3 Prediction Error Method

This chapter considers the parameter estimation of nonlinear stochastic time series models, such as those introduced in Section 3.2, in the context of neural networks. The *Prediction Error* (PEM) method and *Recursive Prediction Error* (RPEM) method are introduced. The derivation presented here applies established methods of optimization theory. The identification of neural network models is considered in numerous articles, but only the major references are mentioned here.

The general idea of the Prediction Error Method is adopted from Ljung and Söderström (1983) and the nonlinear case from Goodwin and Sin (1984), where the method is called "sequential prediction error method". The main idea behind these methods is that the gradient of the selected cost function w.r.t. the identified parameters is computed in a proper manner corresponding to the dynamical behaviour of the predictor. Similar results with a different derivation are presented by Narendra and Parthasarathy (1991) and by Nguyen and Widrow (1991), who called it "dynamic back-propagation" or "back-propagation through time". These also include the steepest descent parameter update. Here the gradient computation and the parameter update are strictly separated.

The actual parameter update is to be made with the Levenberg-Marquardt (LM) approach or with the Recursive Gauss-Newton (GN) algorithm. Chen, Billings and Grant (1990) and Chen *et al.* (1990) presented the first Gauss-Newton type algorithm for the identification of NARX type neural network models. Combining the LM or GN method with the proper gradient computation results in the PEM and RPEM approach also for NARMAX and NOE predictors, Koivisto *et. al.* (1992).

Consider the overall cost function

$$V_N(\theta) = \frac{1}{2}\sum_{t=1}^{N} \left\| (y(t) - \hat{y}(t|\theta)) \right\|_{\Lambda}^2 \tag{3.31}$$

where $\hat{y}(t|\theta)$ is the prediction of the measurement $y \in \boldsymbol{R}^m$ and Λ is a diagonal weighting matrix. The notation $\hat{y}(t|\theta)$ instead of the previous $\hat{y}(t|t-1)$ is used to emphasize the dependence between the prediction and the predictor parameters $\theta \in \boldsymbol{R}^{n_\theta}$. The predictor formulas are presented in Section 3.2, generally as

$$\hat{y}(t|\theta) = f(\varphi(t), \theta)\Big|_{\theta = \hat{\theta}} \tag{3.32}$$

where $\hat{\theta}$ is an estimate of the unknown parameter vector θ and $\varphi \in \boldsymbol{R}^n$ is a regression data vector containing the past measurements, past predicted outputs and past inputs. The actual content of the data vector depends on the selected predictor type: NARX, NARMAX or NOE.

Let us introduce the prediction error

$$\varepsilon(t) = \varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta) \tag{3.33}$$

and the gradient of (3.32) w.r.t. θ as

$$\Psi(t) = \Psi(t, \theta) = \left[\frac{\partial}{\partial\theta}\, \hat{y}(t|\theta)\right]^{T}\Bigg|_{\theta = \hat{\theta}} \tag{3.34}$$

i.e. a $m \times n_\theta$ dimensional matrix. The shorthand notations $\varepsilon(t)$ and $\Psi(t)$ instead more precice $\varepsilon(t, \theta)$ and $\Psi(t, \theta)$ are often used for the clarity of the presentation. If the Hessian of the cost function in Newton's algorithm (2.12) is approximated as

$$V''_N(\theta) \approx \sum_{t=1}^{N} \Psi^T(t)\Lambda\Psi(t) \tag{3.35}$$

and notation

$$V'_N(\theta) = -\sum_{t=1}^{N} \Psi^T(t)\Lambda\varepsilon(t) \tag{3.36}$$

is used, one obtains the *nonlinear least squares normal equations* of the Gauss-Newton (GN) algorithm

$$\left[\sum_{t=1}^{N} \Psi^T(t)\Lambda\Psi(t)\right]\Delta\hat{\theta}(k) = \sum_{t=1}^{N} \Psi^T(t)\Lambda\varepsilon(t) \tag{3.37}$$

from which the search direction $\Delta\hat{\theta}(k)$ at iteration count k can be solved. The approximation of the Hessian $V''_N(\theta)$ in (3.35) must be positive definite. For numerical reasons this may not be the case and the equation (3.37) is modified to guarantee the positive definiteness. Before that the equation (3.37) should be written in a more compact form. For this introduce first a $mN \times n_\theta$ dimensional gradient matrix H so that

$$H^T(\theta) = \left[\Psi^T(1)\Lambda^{\frac{1}{2}} \mid \ldots \mid \Psi^T(N)\Lambda^{\frac{1}{2}}\right] \tag{3.38}$$

and mN dimensional error vector e so that

$$e^T(\theta) = \left[\varepsilon^T(1)\Lambda^{\frac{1}{2}} \ldots \varepsilon^T(N)\Lambda^{\frac{1}{2}}\right] \tag{3.39}$$

Now the modified normal equations can be written in a form

$$[[H^T(\theta)H(\theta) + \delta(k)I]\Delta\hat{\theta}(k) = H^T(\theta)e(\theta)]\Big|_{\theta = \hat{\theta}(k-1)} \tag{3.40}$$

where $\delta(k)$ is a positive scalar regularization parameter. For sufficiently large $\delta(k)$ the positive definiteness is granted. When also $\delta(k)$ is updated based on the success/failure of the previous parameter update, one obtains the *Levenberg-Marquardt* (LM) algorithm, e.g. Scales (1985). The use of regularization $\delta(k)$ can be viewed as an adaptive switching between GN search direction (very small $\delta(k)$) and steepest descent direction (large $\delta(k)$) i.e. if the GN direction does not result in a decrease in the cost function value, a turn towards the steepest descent direction and a reduction in the step length is made ($\delta(k)$ is increased).

The solution of (3.40) is the vector $\Delta\hat{\theta}(k)$ that minimizes

$$\left\| \begin{bmatrix} H(\theta) \\ \sqrt{\delta(k)}I \end{bmatrix} \Delta\hat{\theta}(k) - \begin{bmatrix} e(\theta) \\ \mathbf{0} \end{bmatrix} \right\|\Bigg|_{\theta = \hat{\theta}(k-1)} \tag{3.41}$$

e.g. Ljung (1987), which is a linear least squares problem. This in turn is solved using the QR factorization, resulting in a provisional parameter update

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \Delta\hat{\theta}(k) \tag{3.42}$$

When this straight and simple implementation of LM method using QR factorization is applied, a reliable and robust identification algorithm is obtained. The use of (3.41) is computationally somewhat heavier than solving $\Delta\hat{\theta}(k)$ directly from (3.40). Both approaches have been used extensively in this study.

The evaluation of the gradients is not computationally much heavier than the evaluation of the cost function. Hence the line search is not well motivated here. However, a BFGS approach with two sided Armijo line search (Mimoux, 1986) is also implemented. This works well, but the LM approach normally outperforms it in convergence speed. The BFGS method is used only for some constrained optimization tasks like for stability projection.

Recursive Prediction Error Method (RPEM)

The cost function (3.31) can be minimized using recursive methods. Recursive minimization is mainly used in adaptive prediction and adaptive control but it can be used also as an off-line approach. Quite often it is reasonable to assume that the most recent data contains more information than the past data. To discard the old data exponentially, the quadratic cost function

$$V(t, \theta) = \lambda(t)V(t-1, \theta) + \frac{1}{2}\|(y(t) - \hat{y}(t|\theta))\|_{\Lambda}^2 \tag{3.43}$$

where $0 < \lambda(t) \leq 1$, is minimized at each step t w.r.t. θ i.e

$$\hat{\theta}(t) = \arg\min_{\theta} V(t, \theta) \tag{3.44}$$

The exponential forgetting factor $\lambda(t)$ controls the weighting of past and present prediction errors. Typically $\lambda(t)$ is very close to one. The choice of $\lambda(t) \equiv 1$ yields the recursive minimization of the cost function (3.31).

The solution of (3.44) can be written as recursive Gauss-Newton equations (Ljung, 1987).

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + \bar{R}^{-1}(t)\Psi^T(t)\Lambda\varepsilon(t) \\ \bar{R}(t) &= \lambda(t)\bar{R}(t-1) + \Psi^T(t)\Lambda\Psi(t) \end{aligned} \tag{3.45}$$

where $\bar{R}(t)$ is the information matrix. Also here the shorthand notations are used instead of $\Psi(t, \hat{\theta}(t-1))$ and $\varepsilon(t, \hat{\theta}(t-1))$. The $\bar{R}(t)$ must be kept positive definite. An comprehensive literature considering possible methods is available, e.g. Ljung (1987) and Biermann (1977). One possibility is to use (3.45) directly, which is especially suitable for MIMO identification.

A Modified Weighted Gram-Schmidt (MWGS) procedure is used in this study (Biermann, 1977). This method is numerically one of the most accurate and still efficient, producing results close to those obtained with the Householder transformation. The MWGS approach is selected mainly for its availability. Computationally somewhat lighter but still orthogonalizing methods exist for practical implementations.

In the MWGS the information matrix $\bar{R}(t)$ is updated recursively using the factorization

$$\bar{R}(t) = U(t)D(t)U^T(t) = U(t-1)D(t-1)U^T(t-1) + \Psi^T(t)\Lambda\Psi(t) \tag{3.46}$$

where $U(t)$ is a unit upper triangular matrix and $D(t)$ is a diagonal matrix.

For SISO case the "standard" form

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + P(t)\Psi^T(t)\varepsilon(t) \\ P(t) &= \frac{1}{\lambda(t)}\left[P(t-1) - \frac{P(t-1)\Psi^T(t)\Psi(t)P(t-1)}{\lambda(t) + \Psi(t)P(t-1)\Psi^T(t)}\right] \end{aligned} \tag{3.47}$$

is faster than MWGS, at least if coded properly (assume $\Lambda = 1$ for notational simplicity). In practice also (3.47) must be implemented with a numerically more robust method. The UD factorization (Biermann, 1977) is used in this study. Then $P(t)$ is updated recursively using similar factorization as in (3.46).

The recursive algorithm is initialized with small random values or apriori estimates of $\hat{\theta}(0)$. It should be noted that

$$V(0, \theta) = \frac{1}{2}(\theta - \hat{\theta}(0))^T P^{-1}(0)(\theta - \hat{\theta}(0)) \tag{3.48}$$

It is easy to see that the less confidence we have in the initial value of the parameter estimates $\hat{\theta}(0)$ the larger the initial covariance matrix $P(0) = \bar{R}^{-1}(0)$ should be selected. The usual choice is $P(0) = \bar{R}^{-1}(0) = \eta I$, $\eta >> 0$.

If the exponential data weighting is used, the covariance/information matrix must be bounded. This can be done several in ways, e.g. Ljung and Söderström (1983), for example:

- resetting $\bar{R}(t)$ or $P(t)$ periodically
- using constant trace algorithms for $P(t)$
- using regularization i.e adding δI to $P(t)$ at each time step, where δ is a small positive scalar.

It is well understood that a nonlinear global approximation scheme like MLP is not the most suitable for adaptive purposes, because the model is not linear w.r.t. the parameters. This makes the adaptation slower than in the linear case.

There are also several precautions when applying recursive methods, like sensitivity to the initial weights and in the nonlinear case also sensitivity to the data presentation order which may lead to different local minimums. Furthermore, all the aspects and experience gained applying linear self-tuning control must be taken into account. For example, the fiddling with covariance resetting schemes or with a time varying forgetting factor is similar to that of linear recursive identification.

Computing Gradients

The methods presented in the previous pages are general nonlinear least squares algorithms. They are true Prediction Error Methods (PEM) only if the gradient $\Psi(t)$ is computed properly according to (3.32).

Let us first consider gradient computation generally, without fixing the method to be batch or recursive. Define Jacobian matrices of (3.32) w.r.t. θ and φ as

$$\Xi(t) = \Xi(t, \theta) = \left[\frac{\partial}{\partial\theta} f(\varphi, \theta)\right]\Big|_{\theta = \hat{\theta},\, \varphi = \varphi(t)} \tag{3.49}$$

$$\Gamma(t) = \Gamma(t, \theta) = \left[\frac{\partial}{\partial\varphi} f(\varphi, \theta)\right]\Big|_{\theta = \hat{\theta},\, \varphi = \varphi(t)} \tag{3.50}$$

where $\dim(\Xi) = m \times n_\theta$, $\dim(\Gamma) = m \times n$, where n is the dimension of the input data vector φ. These Jacobians are computed during a backward pass of a MLP network, see Appendix A.

Let us first consider the case of a NARX predictor

$$\begin{aligned} \hat{y}(t|\theta) &= f(\varphi(t), \theta)\big|_{\theta = \hat{\theta}} \\ \varphi(t) &= [y^T(t-d)\ldots y^T(t-d-n_a+1) \\ &\qquad u^T(t-d)\ldots u^T(t-d-n_b+1)]^T \end{aligned} \tag{3.51}$$

where n_a and n_b are the model orders. In this case the Jacobian $\Psi(t)$ can be directly computed, because $\varphi(t)$ is not a function of θ, resulting

$$\Psi(t, \theta) = \left[\frac{\partial}{\partial\theta} \hat{y}(t|\theta)\right] = \Xi(t, \theta)\big|_{\theta = \hat{\theta},\, \varphi = \varphi(t)} \tag{3.52}$$

The combination of (3.32) and (3.52)

$$\begin{bmatrix} \hat{y}(t|\theta) \\ \Psi(t, \theta) \end{bmatrix} = \begin{bmatrix} f(\varphi(t), \theta) \\ \Xi(t, \theta) \end{bmatrix}\Big|_{\theta = \hat{\theta},\, \varphi = \varphi(t)} \tag{3.53}$$

is referred to as the extended network model by Chen, Billings and Grant (1990). The stability of the extended network model is of vital importance in any recursive implementation. The set of all θ producing a stable extended network model is denoted as D_θ.

Consider now a NOE predictor

$$\begin{aligned}\hat{y}(t|\theta) &= f(\varphi(t,\theta),\theta)\big|_{\theta=\hat{\theta}} \\ \varphi(t,\theta) &= [\hat{y}^T(t-1|\theta)\ldots\hat{y}^T(t-n_f|\theta), \\ &\qquad u^T(t-d)\ldots u^T(t-d-n_b+1)]^T\big|_{\theta=\hat{\theta}}\end{aligned} \tag{3.54}$$

where n_a and n_f are the model orders. The notation is used to emphasize the fact that previous predictions are also functions of the parameter estimate $\hat{\theta}$. Applying the chain rule differentiation, one obtains

$$\frac{\partial\hat{y}(t|\theta)}{\partial\theta} = \left[\frac{\partial f(\varphi,\theta)}{\partial\theta} + \sum_{i=1}^{n_f}\left[\frac{\partial f(\varphi,\theta)}{\partial\hat{y}(t-i|\theta)}\right]^T\frac{\partial\hat{y}(t-i|\theta)}{\partial\theta}\right]\Bigg|_{\varphi=\varphi(t),\,\theta=\hat{\theta}} \tag{3.55}$$

The presentation can be remarkably simplified using the (left) matrix fraction description (MFD) presentation of a linear MIMO time series model is (Kailath, 1980). A linearized presentation of the model $\hat{y}(t|\theta) = f(\varphi,\theta)$ near a operation point $\theta = \hat{\theta}, \varphi = \varphi(t)$ is

$$A(q^{-1})\hat{y}(t) = q^{-d}B(q^{-1})u(t) \tag{3.56}$$

where the matrix polynomials in q^{-1} are

$$\begin{aligned}A(q^{-1}) &= I - A_1\,q^{-1} - A_{n_f}\,q^{-n_f} \\ B(q^{-1}) &= B_1 + B_2\,q^{-1} + \ldots + B_{n_b}\,q^{-n_b+1}\end{aligned}$$

Here A_i and B_i are $m \times m$ matrices (assuming a square system). The linearization means the calculation of the Jacobian $\Gamma(t) = \partial f/\partial\varphi$ in (3.50). Comparing this to the organization of the data vector $\varphi(t)$ in (3.54) yields

$$\Gamma(t) = [A_1 \mid \ldots \mid A_{n_f} \mid B_1 \mid \ldots \mid B_{n_b}] \tag{3.57}$$

The Jacobian $\Gamma(t)$ contains directly the parameters of a MFD presentation and (3.55) can be written in more consistent form. The reader should note that the Jacobian $\Gamma(t)$ is not computed in a steady state. Thus the resulting model (3.56) is not a true linear version of the nonlinear model, it is just the Jacobian written in a certain form.

The extended network model can now be written as

$$\begin{bmatrix}\hat{y}(t|\theta) \\ A(q^{-1},t)\Psi^{(i)}(t,\theta)\end{bmatrix} = \begin{bmatrix}f(\varphi(t),\theta) \\ \Xi^{(i)}(t,\theta)\end{bmatrix}\Bigg|_{\substack{\theta=\hat{\theta}\\ i=1\ldots n_\theta}} \tag{3.58}$$

where $\Psi^{(i)}$ and $\Xi^{(i)}$ are the ith columns of the corresponding gradient matrices. The notation $A(q^{-1}, t)$ is used to emphasize that the linearization is performed at every time step.

Eq. (3.58) gives an elegant way to analyse the behaviour of the predictor and its gradients. It will also be used for stability and convergence analysis in the next section. A change in the parameter estimate $\hat{\theta}$ in (3.58) requires the whole sequence of predictions and gradients evaluated recursively for time steps $1 \ldots t$. This feature is inbuilt in the batch type identification algorithm, but not in the recursive case, i.e. the recursive algorithms (3.45) and (3.47) are not truly recursive.

In the recursive case the extended network model can be approximated by simpler recursive equations. Because we can expect $\hat{\theta}(t-1)$ to be close to $\hat{\theta}(t-2)$, … in the limit, a reasonable approximation is to replace (3.58) by

$$\begin{bmatrix} \hat{y}(t|\theta) \\ A(q^{-1}, t)\Psi^{(i)}(t, \theta) \end{bmatrix} \approx \left. \begin{bmatrix} f(\varphi(t), \hat{\theta}(t-1)) \\ \Xi^{(i)}(t, \hat{\theta}(t-1)) \end{bmatrix} \right|_{i = 1 \ldots n_\theta} \tag{3.59}$$

where

$$\begin{aligned} \varphi(t) \; = \; & [\hat{y}^T(t-1|\hat{\theta}(t-2)) \ldots \hat{y}^T(t-n_f|\hat{\theta}(t-n_f-1)) \\ & u^T(t-d) \ldots u^T(t-d-n_b+1)]^T \end{aligned}$$

or if the aposteriori predictions are calculated, with

$$\begin{aligned} \varphi(t) \; = \; & [\hat{y}^T(t-1|\hat{\theta}(t-1)) \ldots \hat{y}^T(t-n_f|\hat{\theta}(t-n_f)) \\ & u^T(t-d) \ldots u^T(t-d-n_b+1)]^T \end{aligned}$$

The equation (3.59) is a sequential form of the approximation of the gradient $\Psi(t)$ and with (3.59) the algorithms (3.45) and (3.47) again become truly recursive (Goodwin and Sin, 1984). It is obvious that this extended network model must be stable in order to obtain correct results using the approximation (3.59).

Equation (3.59) can easily be implemented using neural networks, like multilayered perceptrons, only one forward pass, one backward pass and an extra gradient filtering pass is needed. A minor disadvantage is that gradients $\Psi(j), j = t-1 \ldots t-n_f$ must be stored.

Identification of multistep predictor

Identification of a multistep predictor according to the multistep cost function (3.30) can be formulated in several ways depending whether recursive or batch identification is made. The multistep cost function for multistep NARX is ($N1 = 1$ for simplicity)

$$V_{LRPI1} = \sum_{t=1}^{N} \sum_{j=1}^{N2} \| y(t) - \hat{y}(t|t-j) \|^2 \tag{3.60}$$

where $\hat{y}(t|t-j)$ denotes the predictions computed recursively using the actual measurement information at time $t-j$. This can be reordered as

$$V_{LRPI2} = \sum_{t=1}^{N} \sum_{j=0}^{N2-1} \| y(t-j) - \hat{y}(t-j|t-N2) \|^2 \tag{3.61}$$

where $\hat{y}(t-j|t-N2)$ denotes the predictions calculated recursively using the actual measurement information at time $t-N2$. The overall cost functions are identical, they are just reordered for simpler implementation. However, at time t the incremental cost function is different. The first one needs $N2!$ predictions and gradients (forward and backward passes) to be computed at each sample t, while the latter one needs only $N2$. The disadvantage of the reordering is that the measurement information to the data vector is delayed $N2-1$ samples, which can cause problems in adaptive applications.

According (3.61), the prediction $\hat{y}(t)$ is computed recursively using the actual measurement information available at time $t-N2$. The "future" measurements are replaced with the predicted ones. In a similar way, the gradients are calculated according to the dynamic gradient equations (3.58) and by assuming $\Psi(j) = 0, \quad j < t-N2$.

The cost function and its gradients are considered as a MIMO cost function and the least squares normal equations can be directly written according (3.37) and (3.40). The only difference is in gradient computation.

3.4 Stability and Convergence

This section considers the stability analysis of nonlinear time series models. The main goal has been to develop practical methods for ensuring the stability; methods which can be combined with the optimization procedure (model identification or controller design). The stability analysis consist of two parts: instability detection and stability projection, when instability is encountered. The actual projection is made by direct recalculation of the neural network weights or by applying constrained optimization methods. These methods have been applied e.g. Koivisto *et. al*. (1992, 1993) to a neural network control of a SISO heating system.

The need for the stability projection is twofold. It is needed for the convergence of the optimization procedure. The designed control system should also be stable. Applying the stability projection within an optimization procedure means that the stability is ensured with respect to the data used in optimization, which emphasises the selection of the input signal and setpoint sequences.

The analysis is based on the *Contraction Mapping Theorem*, mainly on the work of Holzman (1970), Economou (1985), Li *et. al.* (1990) and Zafiriou (1990). Contraction mapping is a suitable choice, because it can be used without computing the (possible) equilibrium state. The results considering stability issues of the neural network models, e.g. Hernandez and Arkun (1992), Levin and Narendra (1993) and Jin Liang *et. al.* (1994a), are also discussed. Before presenting actual instability detection and stability projection methods, the stability of nonlinear difference equations is shortly reviewed to point out what in fact is done.

Definition 1. (Holzman, 1970) Let a Banach[1] space X contain a closed convex set Ω and let F map Ω into itself. F is said to satisfy a *Lipschitz* condition, if there is a constant γ such that

$$\left\| F(x_a) - F(x_b) \right\| < \gamma \left\| x_a - x_b \right\| \qquad \forall x_a, x_b \in \Omega \tag{3.62}$$

If $\gamma < 1$, then F is a contraction mapping on Ω.

Theorem 1. (Holzman, 1970) If F is a contraction mapping on Ω, then there is a unique fixed point (equilibrium point) x_{eq} of F in Ω such that

$$x_{eq} = F(x_{eq})$$

Furthermore, for all $x \in \Omega$, $\lim_{k \to \infty} F^k(x) = x_{eq}$, where F^k denotes a recursive mapping (k times).

1. Although the Contraction Mapping Theorem is originally presented using Banach spaces, only linear vector spaces are used in this study.

The contraction conditions are often defined within a closed ball

$$U(x_0, r) = \{x \in X \mid \| x - x_0 \| \leq r\}$$

which maps into itself, although any closed convex set Ω will do. When F is differentiable in U, an exact characterization of the contraction can be developed:

Theorem 2. (Economou, 1985) Let the operator $F : U(x_0, r) \to U(x_0, r)$ (into) be differentiable in $U(x_0, r)$. F is a contraction in $U(x_0, r)$ if and only if

$$\| F'(x) \| \leq \gamma < 1 \qquad \forall x \in U(x_0, r) \tag{3.63}$$

where $\|\cdot\|$ is any induced operator norm.

Finding a closed ball which maps into itself is not always easy and the condition can be replaced with another, more suitable for computation.

Theorem 3. (Economou, 1985) If

$$\| F'(x) \| \leq \gamma < 1 \qquad \forall x \in U(x_0, r) \tag{3.64}$$

where

$$r \geq r_0 \equiv \frac{\| F(x_0) - x_0 \|}{1 - \gamma}$$

then F is a contraction in $U(x_0, r)$. Moreover, F has a fixed point x_{eq} in $U(x_0, r_0)$. The fixed point is unique in $U(x_0, r)$ (region of attraction).

By searching a suitable norm and a suitable ball, the contraction conditions (3.64) can be assured. Note that the (possible) equilibrium point does not need to be computed. If the equilibrium is known, the ball can be centered at x_{eq}, resulting $r_0 = 0$, and any r with $U(x_{eq}, r)$ will do, if (3.64) is satisfied. Now also a mapping into itself is found.

Because the norm $\|\cdot\|$ is one suitable Lyapunov function, then F being a contraction mapping in $U(x_{eq}, r)$ denotes that x_{eq} is an asymptotically stable equilibrium in $U(x_{eq}, r)$ in the sense of Lyapunov, e.g. Ogata (1987) or Li *et al*. (1990). The contraction conditions are stronger than those of asymptotically stability (in the sense of Lyapunov). Sharper Lyapunov functions i.e. weaker conditions can be possibly found. Also the contraction conditions can be twisted, by using an invertible function mapping $z = K(x)$ of Ω onto itself (Holzman, 1970), still leaving contraction conditions stronger than those of the Lyapunov stability.

The spectral radius $\rho(F'(x))$ (largest absolute eigenvalue) is the measure for the asymptotical stability of linear systems. It cannot replace (3.64) when F is nonlinear. However, it can be used as an instability detector like in

Lemma 1. Define F to a *not-a-contraction* in $U(x_0, r)$, if

$$\exists x \in U(x_0, r) \quad \text{so that} \quad \rho(F'(x)) \geq 1 \tag{3.65}$$

because there is no induced norm for which conditions (3.64) are satisfied and so F cannot be a contraction in $U(x_0, r)$, e.g. Economou (1985).

Again, if the equilibrium point x_{eq} is known, the condition

$$\rho(F'(x))\big|_{x = x_{eq}} < 1 \tag{3.66}$$

denotes that x_{eq} is an asymptotically stable equilibrium point in some (undetermined) region of attraction. Hernandez and Yarkun (1992) used this for stability analysis of neural network models and controllers. Eq. (3.66) should not be used alone, because it not even (3.65) does not tell whether F is asymptotically stable or unstable in the desired $U(x_{eq}, r)$ (hopefully the whole operation domain). Some possible trajectories of an nonlinear dynamical system illustrate this, see Fig. 3.7.

Consider now a forced difference equation

$$x(t+1) = F(x(t), u(t)) \quad \text{with } t = 0 \ldots N \tag{3.67}$$

which corresponds to state space representation of an open or closed loop model of some nonlinear dynamical system, $u(t)$ being the system input signal. The task is to detect a possible violation of the asymptotical stability. The input $u(t)$ must be considered as a time-varying parameter and thus one has samples of several different mappings, one for each different $u(t)$ and the detection is made using these samples i.e. $\{x(t)\}$, $t = 0 \ldots N$, parameter values $\{u(t)\}$, $t = 0 \ldots N$ and the model F itself.

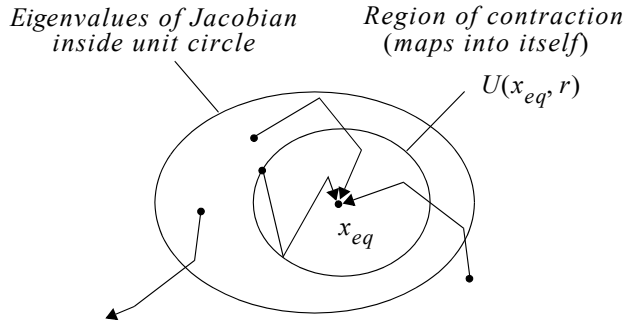


Fig. 3.7. A contraction mapping is asymptotically stable at least in $U(x_{eq}, r)$. Other trajectories are drawn to emphasize that nothing sure can be said about the stability outside.

The not-a-contraction condition and the violation of the contraction conditions are used for the detection of possible "instability", quoted, because the violation of some limit does not necessarily correspond to true instability. The limits for different condition are selected in a pragmatic way, to ensure the stability in practice.

Instability Detection

Consider now a NOE model and assume $d = 1$ for notational convenience. Note also that the system is square. Now according to (3.22)

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1)) \\ \varphi(t+1) &= [\hat{y}^T(t)\ldots\hat{y}^T(t-n_f+1), \\ &\qquad u^T(t)\ldots u^T(t-n_b+1)]^T \end{aligned} \tag{3.68}$$

with $n = n_f + n_b$. The notation $\hat{y}(t+1)$ is here used instead $\hat{y}(t+1|\theta)$. Eq. (3.68) can be presented in state space form

$$\begin{aligned} x(t+1) &= F(x(t), u(t)) \\ x(t+1) &= \begin{bmatrix} [\ \ 0\ \ ] \\ [\ I_A\ ] \\ \quad [0] \\ \quad\ [I_B]\ [0] \end{bmatrix} x(t) + \begin{bmatrix} \tilde{f}(x(t), u(t)) \\ [0] \\ u(t) \\ [0] \end{bmatrix} \\ \hat{y}(t+1) &= [x_1(t+1)\ldots x_m(t+1)]^T \end{aligned} \tag{3.69}$$

Possible I_A and I_B are $m(n_f-1)\times m(n_f-1)$ and $m(n_b-2)\times m(n_b-2)$ dimensional identity matrices. The function $\tilde{f}$ denotes f with a reordered input vector. The state is

$$x(t) = [\hat{y}^T(t)\ldots\hat{y}^T(t-n_f+1)\ u^T(t-1)\ldots u^T(t-n_b+1)]^T \tag{3.70}$$

i.e. $f(\varphi(t+1)) = \tilde{f}(x(t), u(t))$. The Jacobian of (3.69) using the MFD presentation (3.57) is

$$J_1 = F'(x) = \begin{bmatrix} A_1 & \ldots & A_{nf} & B_2 & \ldots & B_{nb} \\ [\ I_A\ ] & & & & & \\ & & [0] & & & \\ & & & [\ I_B\ ] & [0] & \end{bmatrix} \tag{3.71}$$

The corresponding linearized model has transfer function matrix

$$G(z) = z^{-1}A^{-1}(z^{-1})B(z^{-1}) \tag{3.72}$$

The measurements and inputs are typically scaled between $\pm 0{,}5$ and the closed ball $U(0,\ 0{,}5)$ (with the maximum norm $\|\cdot\|_\infty$) is the domain of interest. The maximum norm states that the sum of the absolute values for each row must be less than one. Due to the 1's in some rows, J_1 is in a sense a limiting presentation. It is not easy to find a similarity transform, which gives relaxed conditions, see Zafiriou (1990).

The violation of the contraction condition can be written as

$$\left\| \sum_{i=1}^{nf} |A_i| + \sum_{i=1}^{nb} |B_i| \right\|_\infty \geq 1 \tag{3.73}$$

which is denoted as *AB-condition*. Operator $|\cdot|$ denotes here an element-by-element absolute value. The spectral radius $\rho(J_1)$ gives a less conservative estimator i.e. the system is not-a-contraction if

$$\rho(J_1) \geq 1 \tag{3.74}$$

As shown in Hernandez and Yarkun (1992), the non-zero eigenvalues of J_1 correspond to the eigenvalues of

$$J_2 = \begin{bmatrix} A_1 & \cdots & A_{nf} \\ [& I_A &] \end{bmatrix} \tag{3.75}$$

which can also be reasoned by what is known about the canonical representations for linear time series models. Defining now the ρ*-condition*

$$\rho(J_2) \geq 1 \tag{3.76}$$

which can be used instead of the (3.74). For SISO system this is identical to $|\text{roots}(A(q))| \geq 1$. Also the *A-condition*

$$\left\| \sum_{i=1}^{nf} |A_i| \right\|_\infty \geq 1 \tag{3.77}$$

can be used, because it is stronger than the spectral radius condition.

Also the basic definition (3.62) can be used as a detector, resulting in the *N-condition*

$$\| x(t) - x(t-1) \| \geq \| x(t-1) - x(t-2) \| \tag{3.78}$$

The usability is reduced due to the time-varying nature of the model, see (3.67),and it should be applied with care, because the selection of a suitable norm is even more difficult than in the general case. Using (3.78) causes remarkable savings in numerical load.

All the Jacobians needed for the detection are computed anyway during the identification within the backward pass of the neural network and the detection is computationally light, while the actual projection is not, as it will be seen in the end of the section.

The main differences between these instability detectors are:

- the AB- and N-conditions (3.73) and (3.78) are the most secure taking into account also the numerator of the linearized model.
- the A-condition (3.77) is less secure, neglecting the numerator.
- the ρ-condition (3.76) is the least conservative, a limit for not-a-contraction.

Fulfilling the AB-condition or even A-condition is often very hard, as pointed out by Economou (1985) and the ρ-condition is more useful in practice, especially for SISO systems, where the condition $|\text{roots}(A(q))| \geq 1$ can be transformed to Jury's criterion, resulting in direct limits for parameters of $A(q)$.

The usage of (3.76) or (3.77) contains danger that true instability is not detected. From the pragmatic point of view, Eq. (3.76) can be replaced with

$$\rho(F'(x)) \geq \eta \tag{3.79}$$

where $\eta < 1$ is used as a tuning parameter, if stability problems are encountered, thus causing the eigenvalues to be projected inwards.

As an example consider the model

$$y(t+1) = ay^2(t) + b_1 u(t) + b_2 u(t-1) \tag{3.80}$$

with parameters $\hat{a} = 1$, $\hat{b}_1 = 3{,}1$ and $\hat{b}_2 = 3{,}0$ obtained during the identification. The state space presentation (3.69) is

$$x(t+1) = \begin{bmatrix} y(t+1) \\ u(t) \end{bmatrix} = \begin{bmatrix} \hat{a}\, x_1^2(t) + \hat{b}_2\, x_2(t) \\ 0 \end{bmatrix} + \begin{bmatrix} \hat{b}_1 \\ 1 \end{bmatrix} u(t) \tag{3.81}$$

Assume the current values $y(t) = 0{,}4$, $u(t) = 0$ and $u(t-1) = 0{,}4$, leading to an unstable trajectory, if $u(t)$ is not changed in the future. For $u(t) = u_{eq} = 0$, the system has the equilibrium point $x_{eq} = [\,0\;0]^T$, the current state $x(t) = [0.4\;0.4]^T$ and

$$J_1 = \begin{bmatrix} 2x_1(t) & 3 \\ 0 & 0 \end{bmatrix} \text{ and } J_2 = [\,2x_1(t)] \tag{3.82}$$

Applying the conditions above yields

- the AB- condition : instability detected
- the A -condition : not detected
- the ρ -condition : not detected

This true instability is detected only by the AB -condition which takes into account the numerator of the linearized model, resulting in the parameters a and b_2 to be projected to have smaller values. Note that the zero $= -3/3{,}1$ is stable. The use of the condition $\rho(F'(x)) \geq \eta$ with $\eta < 1$ would also cause the detection of the instability, but resulting parameter a to be projected.

Convergence of RPEM

Ljung and Söderström (1983) present the convergence analysis of the RPEM for linear models. Chen, Billings and Grant (1990) applied this convergence analysis to the NARX type neural network predictors. By using a general method known as the differential equation method for the analysis of recursive parameter estimation algorithms, the convergence of the algorithm (3.45) can be proved. The convergence with the NARMAX and NOE predictors is not proved here, only the practical results are introduced. The main important assumption from the practical point of view is that a projection is employed to keep $\hat{\theta}(t)$ inside the stable region D_θ and some regularity conditions hold.

When looking the extended network model for NARX-predictor (3.53), it is obvious that for a chosen activation function, the D_θ is the whole Euclidian space and the corresponding extended network model is unconditionally stable.

Consider now the extended network model of the NOE predictor (3.59)

$$\begin{bmatrix} \hat{y}(t|\theta) \\ A(q^{-1}, t)\Psi^{(i)}(t, \theta) \end{bmatrix} \approx \left. \begin{bmatrix} f(\varphi(t), \hat{\theta}(t-1)) \\ \Xi^{(i)}(t, \hat{\theta}(t-1)) \end{bmatrix} \right|_{i = 1 \ldots n_\theta} \tag{3.83}$$

This must be kept asymptotically stable to ensure the convergence of RPEM. The dynamics of $\hat{y}(t|\theta)$ and $\Psi^{(i)}(t, \theta)$ cannot be straightforwardly separated, because $A(q^{-1}, t)$ is also a function of the previous estimates $\hat{y}(t-1)\ldots$. Thus the contraction conditions of (3.83) seems to be tighter than those of the predictor only. However, the gradient update equations do not affect each other nor the predictor and they can be considered as separate linear time varying systems

$$A(q^{-1}, t)\Psi^{(i)}(t, \theta) \approx \Xi^{(i)}(t, \hat{\theta}(t-1)) \quad , \quad i = 1 \ldots n_\theta \tag{3.84}$$

The contraction conditions of (3.84) are the same or weaker than those of the predictor i.e. if the predictor is kept asymptotically stable then also the extended network model is asymptotically stable.

Stability Projection

After detection of possible instability, the parameters $\hat{\theta}$ must be projected into a stable domain D_θ. This can be done either with a direct recalculation of the parameters or by constrained minimization of the cost function (3.31).

The recalculation projects the parameters directly into D_θ. Due to the complexity of the function approximator (MLP network) this can be done only in a crude way which somewhat destroys the prediction capability. It is suitable for adaptive applications if high computational load must be avoided. The usage of the constrained minimization is more efficient but also computationally heavier. It is suitable for off-line identification but it can also be used with the RPEM approach.

Consider first the Jacobian J_2 (3.75) which can be used with the A- and ρ-conditions. Multiplying the coefficient matrices A_i with a scalar $k < 1$ so that

$$\tilde{J}_2 = \begin{bmatrix} kA_1 \ \dots \ k^{nf}A_{nf} \\ [\ I_A\ ] \end{bmatrix} \tag{3.85}$$

means that all eigenvalues of $\tilde{J}_2$ are k times the eigenvalues of J_2. For example $k = 0{,}95$ moves the eigenvalues 5 % inwards.

The $\tilde{J}_2$ or $\tilde{A}(q^{-1})$ is used as the target for the parameter projection i.e. where the coefficients in A_i's should be in order to obtain the stability. The multiplier k can be directly determined for low order SISO systems or it can be solved iteratively. A crude but stable approximation is enough.

Consider now the MLP network (Fig. 2.2) with the corresponding Jacobian

$$\Gamma(t) = \left[\frac{\partial}{\partial\varphi} f(\varphi, \theta)\right]\Big|_{\theta = \hat{\theta},\, \varphi = \varphi(t)} \tag{3.86}$$

$$\Gamma(t) = [A_1 \mid \dots \mid A_{n_f} \mid B_1 \mid \dots \mid B_{n_b}] \tag{3.87}$$

Multiply all the weights leaving the particular input nodes of the network corresponding to the A_i's with $\{k, k^2, \dots\}$ and the backward pass of the network model will result in

$$\begin{aligned} \tilde{\Gamma}(t) &= \left[kA_1 \mid \dots \mid k^{nf}A_{n_f} \mid B_1 \mid \dots \mid B_{n_b}\right] \\ \tilde{\Gamma}(t) &= [\tilde{A}_1 \mid \dots \mid \tilde{A}_{n_f} \mid \tilde{B}_1 \mid \dots \mid \tilde{B}_{n_b}] \end{aligned} \tag{3.88}$$

This projects only the Jacobian $\Gamma(t)$ of the linearized network and different value will be obtained if $\Gamma(t)$ is recomputed with a forward-backward pass. This is not a limitation if applied with the RPEM approach and the covariance matrix is reseted or the forgetting factor is decreased after the projection. This approach is successfully applied for on-line identification. This will be denoted as the *direct projection* approach.

A more efficient but computationally more expensive projection method is obtained by applying constrained minimization. Considering the constraint as a hard limit results in unsatisfactory behaviour. The violation of the constraint for example only at one time step overweights its effect. The penalty barrier method is more useful. The cost function for the A-condition can now be written as

$$\begin{aligned} V_N(\theta) &= \frac{1}{2}\sum_{t=1}^{N}\left[\left\|(y(t)-\hat{y}(t|\theta))\right\|_{\Lambda}^2 + \gamma\,\|v(t,\theta)\|^2\right] \\ \text{with } v(t,\theta) &= \begin{cases} (v-1) \text{ if } v = \left\|\sum_{i=1}^{nf}|A_i(t)|\right\|_{\infty} \geq 1 \\ 0 \text{ , otherwise} \end{cases} \end{aligned} \tag{3.89}$$

The cost function is minimized with an increasing scalar weighting γ. The A-condition does not often provide satisfactorily results, it is too conservative. Better results in the SISO case are obtained by combining (3.89) with the ρ-condition i.e. the penalty $v(t,\theta)$ is taken into account only if $|\text{roots}(A(q))| \geq 1$; implemented using the Jury's table. This efficiently pushes the parameters $\hat{\theta}$ into the stable domain D_θ.

For a small order SISO case a straight ρ-condition is applied:

$$\begin{aligned} V_N(\theta) &= \frac{1}{2}\sum_{t=1}^{N}\left[\left\|(y(t)-\hat{y}(t|\theta))\right\|_{\Lambda}^2 + \left\|v(t,\theta)\right\|_{H}^2\right] \\ \text{with } v(t,\theta) &= \begin{cases} \text{col}\,\{[\tilde{A}_1 - A_1 \,|\ldots|\, \tilde{A}_{nf} - A_{nf}]\} & \text{if } \rho(J_2) \geq 1 \\ 0 \text{ , otherwise} \end{cases} \end{aligned} \tag{3.90}$$

where $H = \gamma I$ and $\tilde{A}_i$ is obtained from $\tilde{A}(q^{-1})$ which is determined directly from the Jury's criteria, e.g. Åström and Wittenmark (1990).

The minimization of (3.89) or (3.90) requires the gradient $\partial v(t,\theta)/\partial\theta$ to be evaluated. The computational effort is in the evaluation of $\partial f/\partial\varphi\partial\theta$. Define the matrices

$$\chi(i) = \frac{\partial}{\partial\varphi\partial\theta_i} f(\varphi,\theta)\Big|_{\theta=\hat{\theta},\,\varphi=\varphi(t)}, \quad i = 1\ldots n_\theta \tag{3.91}$$

where θ_i is the ith term in θ and $\dim(\chi(i)) = m \times n$.

The second order gradient $\chi(i)$ can be computed directly for the networks with one hidden layer applying the chain rule differentiation in a similar fashion as done in the normal backward pass. For more complex networks the $\chi(i)$ must be estimated numerically. The secant technique is used in this study, see Appendix A.

These projection methods have been successfully applied within various simulations and real-time experiments. The main drawback is that they are computationally expensive especially due to the numerical evaluation of $\chi(i)$. Because of this computational burden, all conventional methods should be tried in advance, like varying the control specifications in nonlinear controller design. The behaviour of the resulting model, controller or controller-model loop is easy to verify after the design by applying the proposed detection methods.

Input gradient projection

When an identified model is used for predictive control or for controller design, other precautions should also be taken. Because the possible projection task is performed during the identification, the actual conditions are presented also here. A more detailed analysis is presented in Section 4.3. The local existence and uniqueness of the feedback law for the predictors introduced in Section 3.2 necessitates that $\partial\hat{y}(t)/\partial u(t-d)$ is locally invertible i.e.

$$\det\{\partial\hat{y}(t)/\partial u(t-d)\} \neq 0 \qquad \text{(locally)} \tag{3.92}$$

For the SISO case this means that $\partial\hat{y}(t)/\partial u(t-d)$ should be projected off zero. This "input gradient projection" is extremely important for the control point of view. In many cases the sign of this gradient is known in advance, which makes the implementation of the projection algorithm easier. If the input gradient at sample t is near zero, it should be projected away. The actual projection is similar to that used in the stability projection.

Also other apriori information could be taken into account. Assume that the process is known to have a monotonically increasing or decreasing steady-state characteristics and this knowledge is intended to incorporate to the identification procedure, for example because the amount of experimental data is small. An example of this will be presented later, see Chapter 6.

Considering the identified SISO model in the steady state $(\hat{y}_{eq}, u_{eq})$.

$$\begin{aligned} \hat{y}_{eq} &= f(\varphi, \theta) \\ \varphi &= [\hat{y}_{eq}, \hat{y}_{eq}, \ldots u_{eq}, u_{eq}, \ldots]^T \end{aligned} \tag{3.93}$$

From the Inverse Function Theorem, Rosenlicht (1968), local invertibility condition

$$\partial \hat{y}_{eq} / \partial u_{eq} \neq 0 \quad \text{(locally)} \tag{3.94}$$

if applied globally, implies a monotonically increasing or decreasing steady state curve in the SISO case, e.g. Rosenlicht (1968). Applying this to the identified time series model one obtains the condition

$$\sum_{i=0}^{nb-1} \partial \hat{y}(t) / \partial u(t-d-i) \neq 0 \qquad \text{(locally, at the steady state)} \tag{3.95}$$

This condition can be used as a penalty during the identification, i.e. $\forall u(t) \in D_u$, to push the model behaviour into the right direction. It should be used with care because this penalty also restricts the location of the zeroes of the linearized model outside the steady state.

The actual projection is made applying similar penalty barrier as in the stability projection. This can be written (assume positive input gradient here)

$$v(t, \theta) = \begin{cases} (v_{limit} - B_1) & \text{if } (B_1 < v_{limit}) \\ 0 \ , \text{otherwise} \end{cases} \tag{3.96}$$

$$v(t, \theta) = \begin{cases} \left(v_{limit} - \sum_{i=1}^{nb} B_i \right) & \text{if } \left(\sum_{i=1}^{nb} B_i < v_{limit} \right) \\ 0 \ , \text{otherwise} \end{cases} \tag{3.97}$$

where v_{limit} is the lower limit. The former corresponds to the input gradient projection (3.92) and the latter to the condition (3.95).

The data needed for the detection is already available in $\Gamma(t)$. The minimization of these penalties w.r.t. the parameters θ requires the corresponding $\chi(i)$ to be evaluated numerically.

Also a simple heuristic projection method can be applied if the short-cut connections are used and the output activation function is linear, i.e. the network is

$$\hat{y}(t) = f_1(\varphi(t), \theta_1) + \theta_2^T \varphi(t) \tag{3.98}$$

In this case the parameter in θ_2 corresponding to $u(t-d)$ is recalculated so that the input gradient limit is satisfied. This is successfully applied in adaptive on-line control with the RPEM approach.

4 Model Predictive Control

The survey paper (Garcia *et al*. 1989) refers Model Predictive Control (MPC) as that family of controllers in which there is a direct use of explicit and separate identifiable model. The same process model is also implicitly used to compute the control action in such a way that the control design specifications are satisfied.

Control design methods based on MPC concept have found wide acceptance in industrial applications due to their high performance and robustness. There are several variants of model predictive control methods, like Dynamic Matrix Control (DMC), Model Algorithmic Control (MAC) and Internal Model Control (IMC), e.g. Garcia *et al*. (1989). Also nonlinear versions of these are developed, for example the nonlinear IMC concept, e.g. Economou *et al*. (1986). A largely independently developed branch of MPC, called Generalized Predictive Control (GPC), is aimed more for adaptive control, e.g. Clarke and Mohtadi (1989). For the current state-of-the-art of MPC, see Clarke (1994).

Model predictive control in this sense is a broad area and some confusion is encountered because the abbreviation MPC is often used to mean receding horizon (RHPC) or long range predictive control (LRPC) where the model is used to predict the process output several steps into the future and the control action is computed at each step by numerical minimization of the prediction errors, i.e. no specific controller is used. This is quite different from the concept, where the model is controlled with an implicitly derived specific controller, like in many IMC approaches.

In this study the model based approach is divided into two parts: the direct predictive control scheme without actual controllers using neural networks as process models (“Direct Predictive Control”) and the indirect predictive control with actual controller designed using the identified process model (“Dual Network Control”). The third possibility, neural network as a controller, designed or identified without any process model, is omitted in this study and only a short overview is given.

4.1 Control Methods

The existing neural network control approaches can be divided into four categories (Hunt *et al.* 1992): direct inverse control, (direct) model reference control, (direct) predictive control and the IMC approach. The first two ones are in a way similar, a process model is not necessarily required.

The direct inverse control (for example Psaltis *et al.* 1988) utilizes an inverse of the static or dynamic process or its model. It is common in robotics, for example Miller (1989). A general identification scheme is presented in Fig. 4.1 (Hunt and Sbarbaro 1991). The task is to minimize a cost function associated to equivalent prediction error $\varepsilon(t)$. This approach relies heavily on the fidelity of the inverse model and serious questions arise regarding the robustness and the stability.

In the (direct) model reference control the control system attempts to make the plant output match the closed loop reference model asymptotically (Fig. 4.2). The training procedure will force the controller to be a "detuned" inverse, in a sense defined by the reference model. This type of neural network control system is studied for example by Narendra and Parthasarathy (1991), Gupta and Rao (1993) and Jin *et. al*. (1995).

Despite the known potential problems considering robustness and stability, the direct inverse control and (direct) model reference control are the most implemented neural network control methods today. This is obviously due to the straightforward and simple design. It is also true that many processes (models) have a stable inverse and a direct inversion is possible.

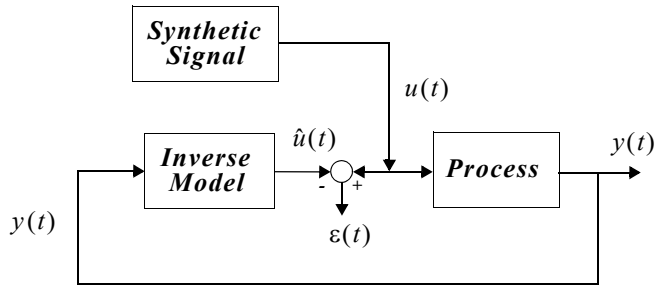


Fig. 4.1. A synthetic signal $u(t)$ is used for identification of the inverse model of the process. The equivalent prediction error is $\varepsilon(t)$.

A practical approach is to use the inferential control scheme, i.e. the process is controlled with a PI or other conventional controller, e.g. Khalid *et al.* (1994b) and Lightbody and Irwin (1995), or with coarsely tuned fuzzy controller, e.g. Khalid *et al.* (1994a). The parallel additive adaptive neural network controller slowly refines the control performance.

Typically the MIT rule (=error backpropagation) is used which is in accordance with the desired slow adaptation speed. Successful practical experiments are presented by Khalid *et. al*. (1992, 1994a) for SISO temperature control, Khalid *et. al.* (1994b) for MIMO furnace control, and Tam (1993) for temperature control.

Some sort of "model" is needed to obtain the gradient of the control error w.r.t. the controller parameters. These "models" vary considerably from a fixed (known) sign of the gradient up to an on-line identified neural network predictor. It is sometimes hard to distinguish between the high end solutions of the adaptive model reference control and the adaptive version of the predictive dual network control. One clear difference is that the former applies the actual control error for controller adaptation while the latter one is adapted according to the predicted control error.

Also many existing CMAC control applications should be classified into this category. The CMAC network is normally identified directly as an inverse of the process (model), i.e. the inferential control approach is not applied. The applications are typically in robotics, for example Miller (1989), but also more noisy applications exist, for example a fuel-injection control (Shiraishi *et. al*. 1995), and a fiber placement composite manufacturing process (Lictenwalner, 1993). Several process control experiments with the LERNAS/AMS have been reported by e.g. Ersu and Tolle (1988), and Tolle and Ersu (1992).

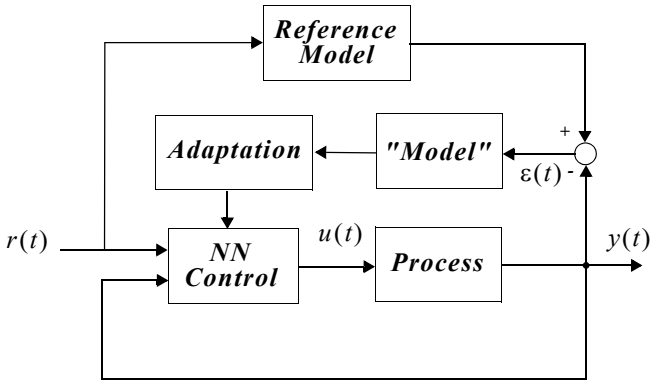


Fig. 4.2. The direct adaptive model reference control system,

The direct model reference approach is also demonstrated for instantaneous adaptive control, i.e. fast initial convergence is required, no inferential scheme and no pretraining are used, e.g. Gupta and Rao (1993) and Jin *et. al*. (1994b, 1995). Also real-time experimental results are reported, e.g. Yabuta *et. al.* (1990). Typically only noise free simulations and experimental results are presented.

Direct model reference control is also common within the geometric approach, e.g. Isidori (1989), because bilinear models are especially suitable for this kind of formulation. Adaptive neural network control studies using the geometric approach are presented by Tzirkel-Hancock and Fallside (1991, 1992), by Rovithakis and Christodoulou (1993), and by Liu and Chen (1993). The off-line approach for a simulated CSTR model can be found in Nikolaou and Hanagandi (1993). These show excellent performance and fast convergence when applied to highly nonlinear (deterministic) simulated process model.

One aspect should be emphasized here. The excellent results were obtained with neural network models which are linear-in-the-parameters. For example Tzirkel-Hancock and Fallside (1991, 1992) used two RBF networks as building blocks for the bilinear model, both with fixed basis functions and with fixed grid as the centers. Sanner and Slotine (1991) present similar results using a preclustered RBF network with fixed basis functions. Comparable experimental results obtained with fully adaptive neural networks have not been found in the literature.

Based on this a clear conclusion is to be made. If fast initial convergence is needed, one should in practice apply linear-in-the-parameters type nonlinear model, neural network or other. The adaptive control of this type of models, like Hammerstein models, is studied widely in the literature, for example Vadstrup (1990).

If the performance during the start-up period is not so important, one can apply the inferential control scheme. The slow adaptation of the (nonlinear-in-the-parameters) neural network controller finally produces good control performance.

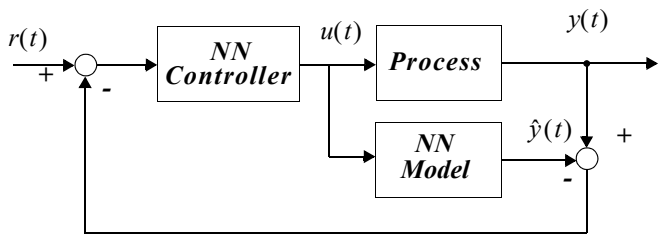


Fig. 4.3. Neural network as a controller in the IMC approach.

If a nonlinear-in-the parameters type model is applied for adaptive control in a way analogical to the linear self-tuning scheme, one must accept that start-up behaviour comparable to the linear case is not typically obtained and that all operation points should be visited at least once before good control performance is achieved. A minor off-line identification is of course a solution to the problems encountered during the initial start-up phase. This reduces the effect of randomly selected initial weights.

Model Predictive Control

The direct predictive neural network control and the dual network control will be studied in detail in Sections 4.2 and 4.3 and only a short overview is given here. In direct predictive control the identified model is used to predict the process output one or several steps into the future and the control action is computed at each step by a numerical minimization of the prediction errors i.e. no specific controller is used.

The other model predictive approach is the dual network control, applied normally within the IMC approach (Fig. 4.3). The control system is commonly implemented with fixed parameters although semi-adaptive and fully adaptive versions are reported. As mentioned earlier, it is sometimes difficult to distinguish between this scheme and direct model reference control. A nonlinear version of LQG control can also be designed. This requires a state estimator design/identification.

In the IMC approach the setpoint is compensated with the prediction error (Fig. 4.3), normally with the filtered one. This introduces an integrator to the control system and no steady state control error is encountered for stepwise disturbances. Another possibility to is to use a PI controller in the outer loop (Fig. 4.4). This approach can of course be used with any controller.

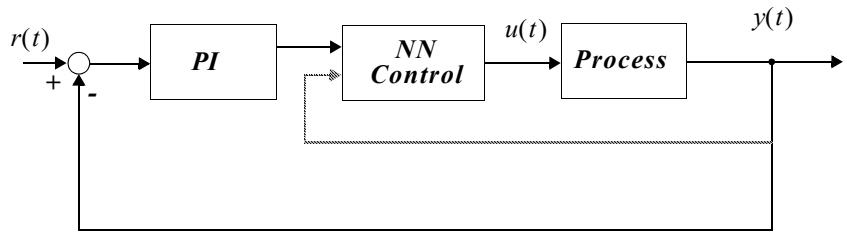


Fig. 4.4. A PI controller can be used to ensure the steady state requirements.

Both the direct predictive neural network control and the dual network control have been studied widely in the literature and successfully applied in the model based control of non-linear systems. General guidelines can be found from Nahas *et al.* (1992), Psichogios and Ungar (1991), Hunt and Sbarbaro (1991), Ydstie (1990), and Bhat and McAvoy (1990).

The main interest seems to be in process control. Most articles consider the control of simulated process models, more or less complex, e.g. a dual network control of a turbogenerator (Wu *et al.* 1992), a LRPC and an IMC control of a steel rolling mill (Sbarbaro-Hofer *et al.*, 1993). The most common applications (simulation models) seem to be pH CSTR's, distillation columns and several fermentors i.e. in the field of the chemical engineering, e.g. Saint-Donat *et al.* (1990), Lee and Park (1992), Hernadez and Arkun (1990), Willis *et al.* (1992) and Chen and Weigand (1994).

Real time tests of model based neural network control have also been reported: control of a small water heating process (Koivisto *et al.* 1992, 1993), a direct predictive control of a fermentor (Willis *et al.* 1992), a CSTR control study (Scott and Ray 1993), a temperature control (Tam 1993), a fed-baker's yeast cultivation process (Ye *et. al*. 1994), a chemical reactor control (Keeler 1994) and a fermentor pressure control (te Braade 1994). Also the work of the Teaching Company Scheme in the University of Newcastle-upon-Tyne (Turner *et. al*. 1994) should be mentioned in this context.

Polynomial models instead of neural network models within the LRPC are applied by Hernandez and Yarkun (1993) and Pröll and Karim (1994). The latter presents also real time experiments with a wastewater pH neutralization process.

Adaptive versions of direct predictive neural network control are studied for example by Cooper *et al.* (1992), Koivisto (1990) and Mills *et al.* (1994). The last introduces an interesting idea of “history stack learning” which clearly seems to outperform “normal” predictive neural network control in controlling a evaporator model. Adaptive real time experiments of direct predictive control or dual network control are not so common. Koivisto *et al.* (1991a, 1991b) applied LRPC for the control of a small water heating process, Tam (1993) applied dual network temperature control and Khalid *et. al.* (1994b) for a MIMO furnace control (direct model reference control).

The difference between direct predictive control and dual network control is merely algorithmic, because the explicit controller can be viewed and designed as a function which produces the control actions similar to those obtained from the direct predictive control, at least in the unconstrained case. This can be done for example by applying the synthetic signal approach (Fig. 4.2).

Soeterboek (1990) has shown that there is very close relationship between both schemes if the model is linear, the criteria is quadratic and there are no constraints. In this case long range predictive control behaves like LQ control or pole-placement control, depending on the selected criteria. This means also that the direct predictive control is not inherently more or less robust than dual network control, but it just can be adjusted more easily for robustness. However, a noise model assumption like NARX which is commonly used within direct predictive control, transforms into a rather complicated state estimator - optimal controller design, when applied within dual network approach.

Also other remarks from the practical point of view can be made. The direct predictive controller is easy to design, only a model identification and fixing the control parameters are required. The implementation of the controller is difficult due to the on-line optimization algorithm. The implemented system can also be numerically heavy especially the long range predictive approach. Because of the on-line optimization, the control signal constraints etc. can easily be taken into account producing efficient control performance. Typically a NARX model is used, simplifying the identification task.

The design of a neural network controller is a numerically heavy off-line optimization task. The resulting controller is easy to implement and only a minor computational on-line load is needed while using the controller. Most of the tuning parameters are fixed during the design and they cannot be altered. The overall control system performance can be somewhat altered with an IMC filter or a state estimator.

4.2 Direct Predictive Control

In the direct predictive control the identified neural network model is used to predict the future process outputs $\hat{y}(t+1), \hat{y}(t+2)\ldots$ and the control signal $u(t)$ is selected so that the predicted control error (cost function) will be minimized. This is formulated to simple one-step-ahead predictive control or to more sophisticated long range (multistep) predictive control (LRPC). Only square systems will be considered i.e. $u \in \mathbf{R}^m$ and $y \in \mathbf{R}^m$.

Model reference is used to make the system behave like a diagonally dominating first or second order linear plant with little or no overshoot. The reference model may be applied in two different ways: filtering the setpoint or filtering the prediction or measurement according to the reference model. The advantage of the latter method is, that it is effective also when the setpoint is constant. These can also be combined to a two-degrees-of-freedom controller.

The system is controlled to follow a linear model

$$\hat{y}(t+d) = \tilde{E}(q)y^*(t+d) \tag{4.1}$$

where the filter $\tilde{E}(q)$ is a diagonal transfer function matrix and $y^*(t+d)$ is the target. Strictly speaking only $\tilde{E}(z)$ is a transfer function matrix. The notation $\tilde{E}(q)$ for the transfer function matrix in difference equations is used instead of $\tilde{E}(q^{-1})$. The target $y^*(t+d)$ is commonly a filtered setpoint $r(t)$ i.e.

$$y^*(t+d) = F_1(q)r(t) \tag{4.2}$$

where $F_1(z)$ is a diagonal transfer function matrix, typically $F_1(z) = I$. Different schemes for model mismatch will be introduced later. The usual selection for (4.1) is

$$\begin{aligned} \tilde{E}(q) &= E^{-1}(q^{-1})\,E(1) \\ E(q^{-1})\hat{y}(t+d) &= E(1)y^*(t+d) \end{aligned} \tag{4.3}$$

where $E(q^{-1}) = I - E_1 q^{-1}$ is a stable matrix polynomial and E_1 is a diagonal matrix (the desired closed loop poles). Also a second order reference model could be used. Using (4.3), the d-step-ahead control cost function can now be written as

$$V(t,u) = \frac{1}{2}\left[\left\| E(1)y^*(t+d) - E(q^{-1})\hat{y}(t+d)\right\|_W^2 + \left\| \Delta u(t)\right\|_\Lambda^2\right] \tag{4.4}$$

where $\Delta = \Delta(q^{-1}) = (1 - q^{-1})I$ and W is a diagonal weighting matrix with positive diagonal elements. The diagonal weighting matrix Λ with positive diagonal elements is used to reduce the control signal variations. By selecting the design parameters, the user can define the properties of the controller. For instance, deadbeat control can be achieved directly with $E(q^{-1}) = I$ and $\Lambda = 0$.

Defining an auxiliary variable $\hat{y}_f(t)$, the cost (4.4) can be written in an analogous form

$$V(t, u) = \frac{1}{2}[\| y^*(t+d) - \hat{y}_f(t+d) \|_W^2 + \| \Delta u(t) \|_\Lambda^2] \tag{4.5}$$

$$E(1)\hat{y}_f(t+d) = E(q^{-1})\hat{y}(t+d)$$

The cost functions (4.4) and (4.5) give identical results (with different W). The selection of the weight W in MIMO case is easier with the former one. The latter is more suitable for analytical treatment and for gradient computation. The quadratic cost function is minimized at each time step with respect to the control $u(t)$ i.e.

$$u(t) = \arg\min_u V(t, u) \tag{4.6}$$

In the *Long Range Predictive Control* (LRPC) (Fig. 4.5) the process model is used to predict future outputs several steps in future and the present and future control signals

$$[u(t) \ldots u(t + Nu - 1)]^T \tag{4.7}$$

are selected so that the predicted control error (cost function) will be minimized.

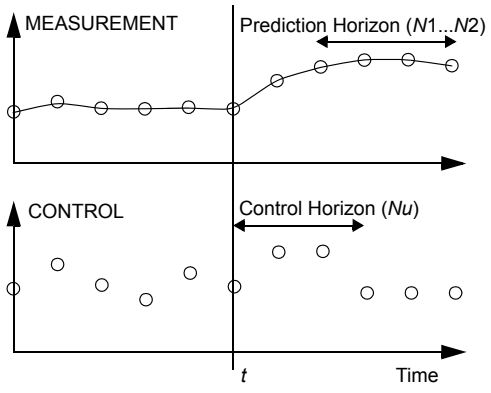


Fig. 4.5. The principle of the long range predictive control.

If the control horizon is shorter than the prediction horizon, the future controls after $Nu-1$ are held constant = $u(t+Nu-1)$.

The quadratic cost function can be written

$$V(t,u) = \frac{1}{2}\sum_{i=N1}^{N2}\left[\left\| E(1)y^*(t+i) - E(q^{-1})\hat{y}(t+i)\right\|_W^2 + \left\| \Delta u(t+i-N1)\right\|_\Lambda^2\right] \quad (4.8)$$

where $N1$ and $N2$ limits the prediction horizon. The selection $N1 = d$ is commonly used in practice.

The cost function (4.8) is minimized at each time instant w.r.t the future controls i.e.

$$[u(t)\ldots u(t+Nu-1)]^T = \arg\min_u V(t,u) \quad (4.9)$$

Only $u(t)$ is used as actual control signal. If $N1 = N2 = d$ one obtains the d-step-ahead controller.

Several gradient based optimization techniques can be used to minimize (4.8). The minimization is generally a constrained optimization task which can be quite complex. Typically at least lower and upper bounds for the control signal are set (u_{min} and u_{max}) and commonly also a maximum allowed change Δu_{max} i.e

$$\begin{aligned} &[u(t)\ldots u(t+Nu-1)]^T = \arg\min_u V(t,u) \\ &u_{min} \le u(i) \le u_{max}, \qquad i = t\ldots t+Nu-1 \\ &\Delta u(i) \le \Delta u_{max}, \qquad i = t\ldots t+Nu-1 \end{aligned} \quad (4.10)$$

A typical selection $Nu = 1$ simplifies considerably the optimization, especially in a SISO case, where one has one cost function value and one free variable $u(t)$. Previous control $u(t-1)$ is also a good initial estimate. The SISO case with $Nu = 1$ can be efficiently handled using parabolic interpolation (Brent's method), e.g. Press (1986). It does not require any gradient information. Typically the minimum is obtained in less than ten iterations. This is efficient because the computational load of one forward pass is not much greater than that of one backward pass.

For complicated tasks a more sophisticated optimizer must be used. Typical methods are based on Sequential Quadratic Programming (SQP). An efficient and reliable implementation called CFSQP (Lawrence, Zhou and Titts 1994) is used in this study. Also the Rosen's Gradient Projection Method (Soeterboek 1990), and the Newton-Raphson method are implemented and applied in some experiments.

As discussed in Section 3.2, there are various possibilities for multistep prediction. The cost function (4.8) is evaluated after computing the predictions. The gradient of the cost function w.r.t. the future controls ($\partial V / \partial u$) is computed according the predictor type by applying the chain rule differentiation. Due to the variety of different multistep predictors and delay structures, procedural formulas are not presented, only the general idea and an illustrative example is given.

Applying the chain rule differentiation is straightforward. A lot of housekeeping is required to build the data vector φ before a forward pass and to separate items from $\partial \hat{y} / \partial \varphi$ after a backward pass. An example using the one-step-ahead NARX predictor is presented in Fig. 4.6. The predictions are computed with a recursive chain of forward passes of the model network. The predicted errors are back-propagated through the recursive chain of the networks to produce $\partial V / \partial u$.

Stability

The stability of various long range and receding horizon predictive control approaches has been studied considerably in recent years. The term "guaranteed stability" is commonly used in this context, e.g. Demircioglu and Clarke (1992). Most properties such as equivalence, stability and internal structures are known for linear systems without I/O constraints. Stability problems are now resolved for linear systems with I/O constraints, e.g. Zafiriou and Marchal (1991) and even for discrete time nonlinear systems, e.g. Alamir and Bornard (1994), as far as a feasible solution exists. For a survey, see Kwon (1994).

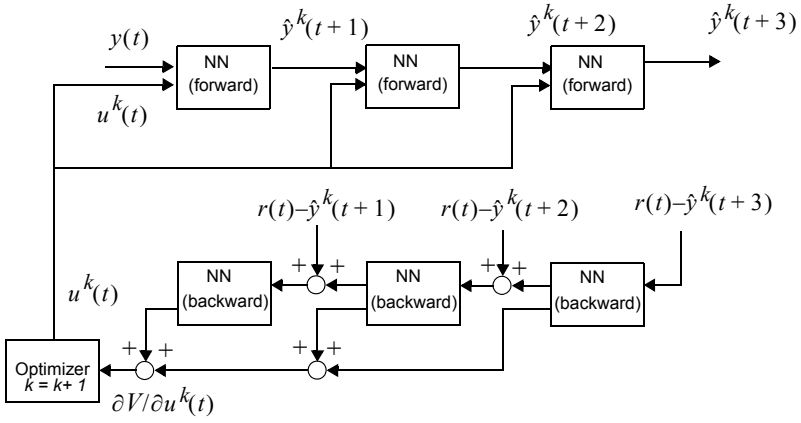


Fig. 4.6. An illustrative example of the gradient calculation. A first order NARX model is used and cost function parameters: $E_1 = 0, \Lambda = 0, N1 = 1, N2 = 3, Nu = 1$. The index k is the iteration counter at time t.

A practical approach to maintain the stability is to add terminal cost or terminal equality constraints, e.g. MacArthur (1993) to the cost function. To avoid a constrained minimization, a terminal cost approach is implemented i.e. the cost function (4.8) is modified to

$$\begin{aligned} V(t,u) &= \frac{1}{2}\sum_{i=N1}^{N2} \left\| E(1)y^*(t+i) - E(q^{-1})\hat{y}(t+i)\right\|_W^2 \\ &+ \frac{1}{2}\sum_{i=1}^{Nu} \left\| \Delta u(t+i-1)\right\|_\Lambda^2 \\ &+ \frac{1}{2}\sum_{i=N2+1}^{N2+N3} \left\| E(1)y^*(t+i) - E(q^{-1})\hat{y}(t+i)\right\|_{W_T}^2 \end{aligned} \tag{4.11}$$

where W_T is the weighting of the terminal cost and $N3$ set the length of the terminal phase. The second term is identical with the (4.8) although defined in a different way.

Model Mismatch

If the direct predictive control scheme is applied with constant model parameters, a steady state control error will be encountered due to the modeling error. This must be removed by incorporating an integrating action to the controller. The approach applied here is to assume that the noise is generated by an *integrated moving average* (IMA) noise process i.e. the model for the additive noise $\{z(t)\}$ is assumed to be

$$\Delta z(t+1) = C(q^{-1})\nu(t+1) \tag{4.12}$$

where $\{\nu(t)\}$ is a m-dimensional independent white noise sequence and C is combined from monic and stable scalar polynomials i.e. a matrix polynomial

$$C(q^{-1}) = I + C_1 q^{-1} + \ldots C_{nc} q^{-nc}$$

with diagonal C_i's.

For the formulation of the predictor, a (NARX + IMA = NARIMAX) noise model assumption is used

$$\begin{aligned} y(t+1) &= \tilde{y}(t+1) + z(t+1) \\ \tilde{y}(t+1) &= f(\varphi(t+1)) \\ \Delta z(t+1) &= C(q^{-1})\nu(t+1) \\ \varphi(t+1) &= [y^T(t), \ldots, y^T(t-n_a+1), \\ &\qquad u^T(t-d+1), \ldots, u^T(t-d-n_b+2)]^T \end{aligned} \tag{4.13}$$

where $\tilde{y}(t)$ is used as a shorthand notation for the predictions obtained from the plain

NARX part. The IMA noise model is just added to the identified NARX predictor to remove the steady state control error. This noise assumption is similar to the T-polynomial used in GPC, e.g. Clarke and Mohtadi (1989). The C-polynomial is not identified, it is used as a tuning parameter to increase the robustness of the control system.

The situation is in general not so straightforward, it depends of the true noise process i.e. whether the IMA noise is actually present in the system. If so, identification problems will be encountered if a plain NARX model is used. In the original GPC formulation, the linear ARIX predictor i.e. ARX with $C(q^{-1}) = I$ is identified, but used with $C(q^{-1}) \neq I$ when solving the control $u(t)$. The identification of an ARIX type predictor also decreases the signal/noise ratio and thus the quality of the prediction and it is questionable whether it is a generally applicable solution especially in adaptive control. The situation is application dependent and the user must select whether to use it or not.

Because the noise model is additive and consists of separate linear SISO models, the minimum variance predictor for each $z_i(t+i)$ can easily be determined by solving the corresponding Diophantine equation, e.g. Åström and Wittenmark (1990), resulting

$$\begin{aligned} C(q^{-1})\hat{z}(t+1|t) &= C(1)z(t) = C(1)(y(t)-\tilde{y}(t)) \\ \hat{z}(t+i|t) &= \hat{z}(t+1|t), \quad \text{when } i>1 \end{aligned} \tag{4.14}$$

where the actual noise $z(t) = y(t) - \tilde{y}(t)$ is calculated backwards using (4.13). The final predictor equations at time t can be presented as a recursive procedure

repeat $i = 1 \ldots N2$

$$\begin{aligned} \tilde{y}(t+i) &= f(\varphi(t+i)) \\ \varphi(t+i) &= [y^T(t+i-1), \ldots, y^T(t+i-n_a), \\ &\qquad u^T(t+i-d), \ldots, u^T(t+i-d-n_b+1)]^T \\ \hat{y}(t+i) &= \tilde{y}(t+i) + \hat{z}(t+i|t) \\ y(t+i|t) &= \hat{y}(t+i) \end{aligned} \tag{4.15}$$

end

where the last equation denotes that the predicted measurements $\hat{y}(t+i)$ are used instead of the missing future measurements $y(t+i)$. When the predictor (4.14)...(4.15) is used with the cost function (4.8), no steady state control error is encountered for stepwise load disturbances. Typically $C(q^{-1}) = I - C_1 q^{-1}$ i.e. the prediction error is simply filtered with a first order filter and added to the actual predictor (4.15).

The extension to the NARMAX or NOE predictor is straightforward, but the NOE case is of special interest. Because the noise is additive to the output after the dynamical part (the NOE model itself), the filtered prediction error can also be subtracted from the target. Now the predictor equations for the NOE + IMA (= NOE-IMA) case can be presented as

repeat $i = 1 \ldots N2$

$$\begin{aligned} \hat{y}(t+i) &= f(\varphi(t+i)) \\ \varphi(t+i) &= [\hat{y}^T(t+i-1), \ldots, \hat{y}^T(t+i-n_a), \\ &\qquad u^T(t+i-d), \ldots, u^T(t+i-d-n_b+1)]^T \\ y^*(t+i) &= r(t) - \hat{z}(t+i|t) \end{aligned} \tag{4.16}$$

end

where $\hat{y}$ now denotes the predictions obtained from the plain NOE part.

Adding the setpoint filtering and adopting the formulation (4.2), one obtains

$$y^*(t+i) = F_1(q)r(t) - F_2(q)[y(t) - \hat{y}(t)] \qquad i = 1 \ldots N2 \tag{4.17}$$

where $F_1(z)$ and $F_2(z) = C^{-1}(z)C(1)$ are diagonal transfer function matrices. Now the overall control system can be presented as an analogous two-degrees-of-freedom *internal model control* (IMC) approach, e.g. Morari and Zafiriou (1989), see Fig. 4.7.

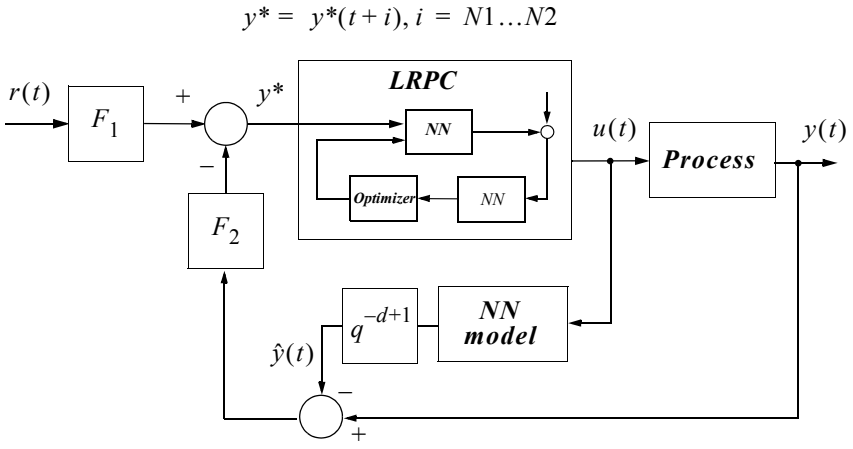


Fig. 4.7. The LRPC with the NOE model can be presented as an IMC approach. The LRPC block represents iterative solving of the control signal $u(t)$ using the identified NOE model.

A practical advantage of this formulation is that the filter F_2 can easily be designed for other types of load changes. This is especially useful because the internal LRPC-NOE loop transforms the system to be controlled to a diagonally dominating linear system and the filter F_2 can be designed for an open loop system of the form $z^{-d}\tilde{E}(z)$, which consists of linear first order SISO models. Straightforward guidelines for the filter design can be found in Morari and Zafiriou (1989).

The proposed nonlinear LRPC approach as a whole has similar features as its linear counterparts, General Predictive Control (GPC), Model Predictive Control and others. The main benefit is efficiency. The design specifications can be set on-line. The exact apriori knowledge of the delay is not so critical as in one-step-ahead controller. LRPC can even handle unstable and non-minimum phase systems.

The LRPC approach is efficient but at a price of high computational on-line load. Because most "true" neural network control systems are anyway targeted to dedicated special tasks, it is reasonable to assume that a computer platform for the LRPC approach is affordable.

Although the LRPC approach has several design parameters, the tuning is straightforward. Some guidelines for tuning is presented below. The proposed scheme is for the LRPC with NARX predictor:

1. Set $F_1(z) = I$, $N1 = N2 = d$, $Nu = 1$ and $\Lambda = 10^{-4} \cdot I$ (or something small)

2. Select $E(q^{-1})$ so that its poles corresponds to the process open loop time constant. Select $C(q^{-1})$ to be same or slower than $E(q^{-1})$

3. Stabilize the system. Increase $N2$ (or increase Λ) while significant ringing can be seen in the control signal.

4. Fine tune $E(q^{-1})$ and $C(q^{-1})$ so that the noise rejection and load disturbance compensation are adequate. This is application dependent and no general guidelines can be given.

5. Finally, fine tune F_1 if setpoint chance causes too heavy control signals variations.

Adaptive LRPC

This section considers an adaptive version of the direct predictive neural network control. The current state-of-the-art and basic problematics associated to this type of control is discussed and also some solutions are proposed. These considerations are also applicable to other types of adaptive nonlinear control.

When the d-step-ahead control law (4.4) or the LRPC control law (4.8) is combined with recursive identification (RPEM) presented in Section 3.3, one obtains an adaptive nonlinear controller (Fig. 4.8). The approach should in principle achieve good control performance, at least at the end. The usage of a nonlinear model makes adaptation slower than in the linear case although efficient recursive scheme like RPEM increase the speed of convergence. This slow adaptation must be combined with a robust control law like LRPC in order to obtain acceptable results. The experimental results are still rare at least if nonlinear-in-the-parameters type models are considered.

The linear adaptive and self-tuning control is studied extensively in the literature, see for example Bitmead *et. al.* (1990) for an excellent analysis of the linear adaptive GPC. Also results considering adaptive nonlinear control with linear-in-the-parameters type models are reported widely.

For example Pröll and Karim (1994) have applied polynomial ARMAX type models for adaptive nonlinear LRPC. They present also real-time experiments using a wastewater pH neutralization process. It is clear that the usage of the linear-in-the-parameters type model makes the identification easier and increases the control system performance during the initial start-up.

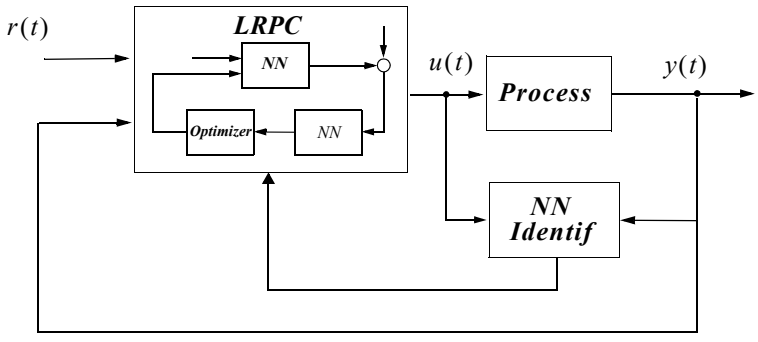


Fig. 4.8. An adaptive direct predictive control scheme.

If linear-in-the-parameters nonlinear models within the LRPC produce good results, why use nonlinear-in-the-parameters type neural network models at all? The reason is clear. If coarser or simpler models are applied, one must typically perform a nontrivial model structure selection according a priori process knowledge. The actual model is then normally easier to identify, it might be even linear-in-the-parameters. On the other hand, neural network models can represent virtually any multivariate nonlinear smooth function with the same basic structure.

If a nonlinear-in-the parameters type model is applied for adaptive control in a way analogical to the linear self-tuning scheme, one must accept that start-up behaviour comparable to the linear case is not typically obtained and that all operation points should be visited at least once before good control performance is obtained.

Thus the usability of the adaptive LRPC with the MLP network or with any similar global approximator depends on, how slow adaptation is acceptable in a particular application. Experimental case studies considering these aspects will be presented in Section 6.2. The examples are selected to discuss basic problematics and typical behaviour of this type of control.

The reliability and repeatability are in practice the main requirements for any adaptive control system, much more than a requirement of an instantaneously good control performance. These are just the same issues where all adaptive control systems have major problems. These problems are emphasized with adaptive neural network control due to the nonlinear nature of the model and/or controller.

A minor off-line identification is of course a solution to the problems encountered during the initial start-up phase. This somewhat reduces the effect of randomly selected initial weights. Sensitivity to the initial weights is obvious but the adaptation is also sensitive to the data presentation order i.e. also to the selected setpoint sequence.

Based on the simulation studies and real time experiments, the adaptation seems also to be quite sensitive to the incorrect noise model assumption, even with the “full” LRPC approach including the C-polynomial to compensate the model mismatch. Naturally this problem is present only in the stochastic case. The adaptive LRPC with nonlinear-in-the-parameters type neural network model is quite capable to control deterministic nonlinear process without any a priori identification. This is verified with extensive simulation studies.

Several precautions must be taken into account to keep the control system stable and to secure the identification algorithm. The usage of (variable) forgetting factor during the start-up phase or resetting the covariance/information matrix periodically reduces the effect of the initial weights during the start-up phase and keeps the identification "alive", but they do not totally remove the problems. One can be "lucky" and obtain excellent start-up behaviour or "unlucky" and obtain a disaster.

The application of stability and input gradient projection schemes secure the control system against total disaster but the resulting control performance is not acceptable for a long period of time. In practice some preliminary off-line identification should be performed to achieve reasonable initial weights.

Ideal model candidate for nonlinear adaptive control should have following properties:

- representation power of an MLP or RBF network
- linear-in-the-parameters
- suitable for high dimensional mappings (input dimension $4 \ldots 20$)
- good extrapolation capability
- only minor apriori model structure selection is necessary

It is apparent that this type of model family is not available and in practice a compromise between the properties is to be made. Some attempts are reviewed below.

Consider the polynomial ARMAX type models, e.g. Hernadez and Yarkun (1993) and Pröll and Karim (1994). The models are simplified versions of the Volterra series representation. A preliminary selection of necessary multiplicative terms is performed before the identification. After that the model itself is linear-in-the-parameters and the identification is easy.

There are also other possibilities for coarser and/or simpler models, for example piecewise linear models. Unfortunately the obvious candidates, B-splines and fuzzy models are suitable only for very low order tasks, not for time series modeling, as discussed in Section 2.2. The use of piecewise linear activation function within an MLP structure, like *hinge* network (Niu and Pucar 1995) results in a coarser model which has many attractive properties. The model is still nonlinear-in-the-parameters and due to non-continuous derivatives the on-line gradient based identification is difficult. The hinge network is thus more suitable for off-line tasks.

An alternative is to use an MLP or RBF neural network with preselected first layer or with preselected radial base functions so that the remaining identifiable part is linear-in-the-parameters. The determination of the first layer weigths or radial basis functions is however nontrivial and in practice some preliminary identification is required.

Pao (1992) formulated the identification task in a different way. An MLP network with one huge hidden layer was used (even hundreds of nodes). The lower layer weights were randomly initialized and fixed after that. Only the weights of the upper layer were identified on-line. Thus the overall model was linear-in-the-parameters and fast initial convergence was obtained. It is however clear that this type of model has extremely low extrapolation capability and it incorporates potential instability etc. problems.

Still another alternative is to identify local linear models within a priori clustered input space and to combine the *predictions* using Gaussian, e.g. Johanssen (1994), or triangular membership functions. In both cases the global model is nonlinear-in-the-parameters.

Mills *et al.* (1994) introduce an interesting idea of "history stack learning" which clearly outperforms "normal" neural network based adaptive LRPC when controlling a evaporator model. The problems with the initial weights and with the nonlinear-in-the-parameters property are reduced by performing a couple of iterations at each time step with an off-line identification scheme using the stored data.

This idea can be developed even further. Modern workstation can store significant amount of data directly in the memory. An on-line measurement data base - a sophisticated history stack which stores only the most relevant process information could be used. Any global or local model could then be refined on-line but using more efficient off-line methods. Theoretically this requires development of better on-line clustering algorithms, algorithms combined with the "one-shot-learning" capability of the ART-2 neural network.

Finally it should be noted that an autotuner property - identification by request - is normally in practice favored instead of any adaptive controller. If the process is nonlinear a nonlinear identification task is still to be performed. Also these problems could be solved using the mentioned sophisticated on-line measurement data base.

Research towards this data base and towards suitable coarser model structures still maintaining the main attractive features of MLP and RBF networks, and the development of the corresponding identification schemes are clearly important issues of the future research.

4.3 Dual Network Control

Several methods have been developed for the model based neural network controller design, see for example Hunt *et. al.* (1992). Internal Model Control (IMC) approach is one way to design and implement robust nonlinear controller. Linear IMC design has been extensively studied in the literature (e.g. Morari and Zafiriou 1989). The main principle of linear IMC design for stable processes is that the process model is factored so that the part containing the time delay and unstable zeros is separated and left uninverted. Consequently, the closed-loop system is stable and the controller is realizable with a suitable chosen filter.

The development of general nonlinear extension of IMC poses serious difficulties due to the inherent complexity of nonlinear systems. Thus the nonlinear extension of IMC design presented by Economou *et al*. (1986) is applicable only to open loop stable systems with a stable inverse. However, they have shown that under certain assumptions, the closed-loop system possesses the same stability and zero-offset properties as the linear IMC structure. This work has been used as a basis by Nahas *et al*. (1992) who have demonstrated that neural networks can be applied straightforwardly in the IMC design.

The problems related to the model inverse based IMC design can be partially avoided by employing a nonlinear optimal controller within the IMC structure (Koivisto *et al.* 1992). This approach is a general extension of the design methods proposed by Economou *et al*. (1986). The nonlinear control law is approximated by a perceptron network and the cost function associated to the optimal controller design is minimized numerically.

The main characteristics of this IMC design are:

- the optimal controller is designed to control the process model w.r.t a desired linear closed loop model
- the designed controller is implemented to control the model not the process itself
- the controller outputs are used as actual process inputs
- the robustness of the control system is increased by using the model mismatch to compensate the setpoint of the internal controller-model loop
- the stability of the internal controller-model loop and the controller is ensured during the controller design phase.

This IMC strategy is applied to the control of a laboratory heating process (SISO), Koivisto *et al*. (1992, 1993).

A more detailed IMC structure is presented in Fig. 4.9. The IMC controller is realized with linear filters F_1 and F_2. The filter F_2 improves the robustness of the control system by reducing the closed-loop gain. The filter F_1 is used for setpoint filtering.

Assume now that the deterministic part of the process is identified, i.e. a d-step-ahead NOE predictor is available

$$\begin{aligned} \hat{y}(t+d) &= f(\varphi(t+d)) \\ \varphi(t+d) &= [\hat{y}^T(t+d-1)\ldots\hat{y}^T(t+d-n_f) \\ &\qquad u^T(t)\ldots u^T(t-n_b+1)]^T \end{aligned} \tag{4.18}$$

and let us introduce a nonlinear parametric feedback law (controller)

$$\begin{aligned} u(t) &= h(\phi(t),\Theta) \\ \phi(t) &= [y^{*T}(t+d)\ldots y^{*T}(t+d-m_c+1) \\ &\qquad \hat{y}^T(t+d-1)\ldots\hat{y}^T(t+d-m_a) \\ &\qquad u^T(t-1)\ldots u^T(t-m_b)]^T \end{aligned}, \tag{4.19}$$

where $\phi(t) \in \mathbf{R}^{(m_a+m_b+m_c)}$ is the data vector and $\Theta \in \mathbf{R}^{n_\Theta}$ is the parameter vector of the controller. Note the specification of the data vector $\phi(t)$, the delay free part of the system is to be controlled. This is identical to the Smith predictor formulation. The target $y^*(t+d)$ is the same as used in the direct predictive IMC control (4.17) i.e.

$$y^*(t+d) = F_1(q)r(t) - F_2(q)[y(t)-\hat{y}(t)] \tag{4.20}$$

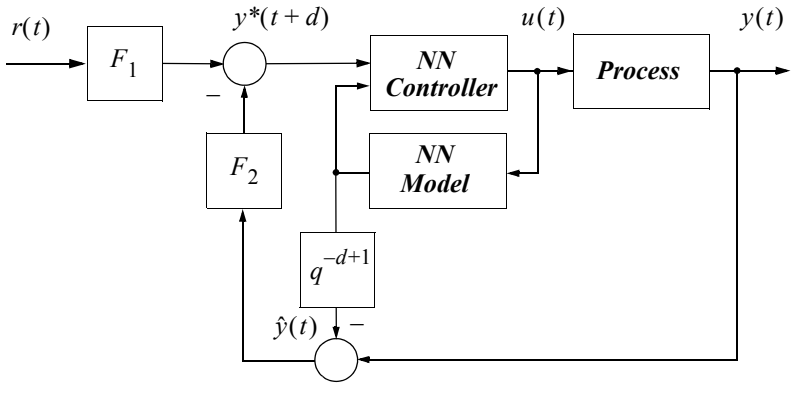


Fig. 4.9. The dual network control, an IMC implementation.

The filters F_1 and F_2 are used only after the implementation of the closed-loop system. During the design of the controller $y^*(t+d) = r(t)$, see Fig. 4.10.

The control law is realized with a MLP network and the parameters, i.e. the weights of the network, are obtained by minimizing the quadratic cost function

$$V(\Theta) = \frac{1}{2}\sum_{t=1}^{N}[\| y^*(t+d) - \hat{y}_f(t+d|\Theta)\|_W^2 + \| \Delta u(t|\Theta)\|_\Lambda^2] \tag{4.21}$$

where

$$E(1)\,\hat{y}_f(t+d|\Theta) = E(q^{-1})\,\hat{y}(t+d|\Theta) \tag{4.22}$$

Notations $\hat{y}(t|\Theta)$, $\hat{y}_f(t|\Theta)$ and $u(t|\Theta)$ are used to emphasize the dependence from the controller parameters. The matrix polynomial $E(q^{-1})$ specifies the desired closed-loop response. It can also be used to reduce controller output variations. By selecting the design parameters, the user can define desired properties of the controller. For instance, the model inverse based approach can be achieved directly with $E(q^{-1}) = I$ and $\Lambda = 0$. The control cost function is similar to that used with LRPC, but only one-step-ahead control is used.

Defining the equivalent control error

$$\varepsilon(t) = \varepsilon(t,\Theta) = [(y^*(t+d) - \hat{y}_f(t+d|\Theta))^T \quad \Delta u^T(t|\Theta)]^T \tag{4.23}$$

and a new weighting matrix

$$Q = \begin{bmatrix} W & 0 \\ 0 & \Lambda \end{bmatrix} \tag{4.24}$$

the cost function can be written

$$V(\Theta) = \frac{1}{2}\sum_{t=1}^{N}\| \varepsilon(t,\Theta)\|_Q^2 \tag{4.25}$$

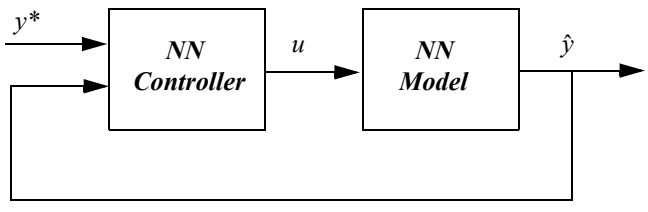


Fig. 4.10. The dual network system during the controller design.

Defining a $2m \times n_\Theta$ dimensional Jacobian matrix

$$\Psi(t) = \Psi(t, \Theta) = \left[\left[\frac{\partial}{\partial \Theta} \hat{y}_f(t+d|\Theta)\right]^T \left[-\frac{\partial}{\partial \Theta} \Delta u(t|\Theta)\right]^T\right]^T \tag{4.26}$$

yields a control design task in the same form as the model identification task. The PEM approach can now be directly applied after computing the gradients.

Computing the gradients

The gradient $\Psi(t)$ is computed in a similar manner as in the identification phase to produce true PEM estimates. Both the model and the controller are assumed to be of NOE type. Let us first define the Jacobian matrices w.r.t. Θ, φ and ϕ as

$$\Xi_u(t) = \Xi_u(t, \Theta) = \left[\frac{\partial}{\partial \Theta} h(\phi, \Theta)\right]\Big|_{\Theta = \hat{\Theta},\, \phi = \phi(t)} \tag{4.27}$$

$$\Psi_u(t) = \Psi_u(t, \Theta) = \left[\frac{\partial}{\partial \Theta} u(t|\Theta)\right]\Big|_{\Theta = \hat{\Theta}} \tag{4.28}$$

$$\Psi_y(t+d) = \Psi_y(t+d, \Theta) = \left[\frac{\partial}{\partial \Theta} \hat{y}(t+d|\Theta)\right]\Big|_{\Theta = \hat{\Theta}} \tag{4.29}$$

$$\Gamma_u(t) = \Gamma_u(t, \Theta) = \left[\frac{\partial}{\partial \phi} h(\phi, \Theta)\right]\Big|_{\Theta = \hat{\Theta},\, \phi = \phi(t)} \tag{4.30}$$

$$\Gamma_y(t+d) = \Gamma_y(t+d, \Theta) = \left[\frac{\partial}{\partial \varphi} f(\varphi, \Theta)\right]\Big|_{\Theta = \hat{\Theta},\, \varphi = \varphi(t+d)} \tag{4.31}$$

where Ξ_u, Ψ_u and Ψ_y are $m \times n_\Theta$ dimensional matrices, $\dim(\Gamma_y) = m \times (n_f + n_b)$ and $\dim(\Gamma_u) = m \times (m_a + m_b + m_c)$. The Jacobians Ξ_u, Γ_y and Γ_u are computed during the backward pass of model and controller networks. Now according to (4.22)

$$\Psi(t, \Theta) = \begin{bmatrix} E^{-1}(1)E(q^{-1})\Psi_y(t+d, \Theta) \\ \Delta\Psi_u(t, \Theta) \end{bmatrix} \tag{4.32}$$

Analogously to the identification phase, recursive formulas for Ψ_u and Ψ_y can be derived by applying the chain rule differentiation.

Applying the MFD presentation also here, one obtains a linearized model

$$\begin{aligned} R(q^{-1})u(t) &= S(q^{-1})\hat{y}(t+d-1) + T(q^{-1})y^*(t+d) \\ A(q^{-1})\hat{y}(t+d) &= B(q^{-1})u(t) \end{aligned} \tag{4.33}$$

where the matrix polynomials in q^{-1} are

$$\begin{aligned} R(q^{-1}) &= I - R_1\, q^{-1} - \ldots - R_{m_b}\, q^{-m_b} \\ S(q^{-1}) &= S_1 + S_2\, q^{-1} + \ldots + S_{m_a}\, q^{-m_a+1} \\ T(q^{-1}) &= T_1 + T_2\, q^{-1} + \ldots + T_{m_c}\, q^{-m_c+1} \\ A(q^{-1}) &= I - A_1\, q^{-1} - \ldots - A_{n_f}\, q^{-n_f} \\ B(q^{-1}) &= B_1 + B_2\, q^{-1} + \ldots + B_{n_b}\, q^{-n_b+1} \end{aligned}$$

and

$$\begin{aligned} \Gamma_y(t+d) &= [A_1 \mid \ldots \mid A_{n_f} \mid B_1 \mid \ldots \mid B_{n_b}] \\ \Gamma_u(t) &= [T_1 \mid \ldots \mid T_{m_c} \mid S_1 \mid \ldots \mid S_{m_a} \mid R_1 \ldots \mid R_{m_b}] \end{aligned} \tag{4.34}$$

These results in the extended network model

$$\left[\begin{array}{rcl} u(t|\Theta) &=& h(\phi(t), \Theta) \\ \hat{y}(t+d|\Theta) &=& f(\varphi(t+d)) \\ R\Psi_u^{(i)}(t, \Theta) &=& S\Psi_y^{(i)}(t+d-1, \Theta) + \Xi_u^{(i)}(t, \Theta) \\ A\Psi_y^{(i)}(t+d, \Theta) &=& B\Psi_u^{(i)}(t, \Theta) \end{array}\right]\Bigg|_{\substack{\Theta = \hat{\Theta} \\ i = 1 \ldots n_\Theta}} \tag{4.35}$$

where the arguments (q^{-1}, t) of the matrix polynomials are omitted and the superscript (i) denotes the ith column of the corresponding Jacobian matrix. The last two rows correspond to the gradient update equations. With the RPEM approach these gradient update equations are used only as an approximation, similarly as in the model identification. The equations (4.32) and (4.35) form now the necessary prediction and gradient computation procedure.

Stability

When considering the stability of the implemented control system (Fig. 4.9), four different stability issues are encountered:

1. Model. The model is assumed to be globally asymptotically stable. This is a necessity only for the RPEM approach.

2. Controller. A parameter projection must be employed to ensure the asymptotic stability of the controller. This is necessary, because the controller tries to cancel the internal dynamics of the process model. The zero dynamics (Isidori, 1989) of the process model can be unstable causing the resulting controller to be unstable or to have a ringing effect.

3. Controller-model loop. The asymptotic stability of the model and the controller does not ensure the stability of the internal loop which consist of the controller and the process model. This stability must be ensured by the parameter projection.

4. Closed loop. The closed-loop gain can always be reduced by selecting a filter F_2 so that the input-output stability of the overall IMC structure is guaranteed, e.g. Economou *et al*. (1986).

The main requirement for the convergence of the RPEM is that $\hat{\Theta}(t)$ is restricted to the stable region D_Θ of the parameter space (Ljung and Söderström, 1983). This condition is equivalent to the stability of the extended network model (4.35). Using similar reasoning as in Section 3.4, the gradient update equations in (4.35) can be neglected, leaving the conditions for the model and controller and their combination to be analyzed. The stability conditions of the nonlinear IMC structure are thus identical to those required by the convergence of the RPEM with one addition: the stability of the overall control system.

Consider now the model (4.18) and the controller (4.19)

$$\begin{aligned} \hat{y}(t+d) &= f(\varphi(t+d)) \\ \varphi(t+d) &= [\hat{y}^T(t+d-1)\ldots\hat{y}^T(t+d-n_f) \\ &\qquad u^T(t)\ldots u^T(t-n_b+1)]^T \end{aligned} \tag{4.36}$$

$$\begin{aligned} u(t) &= h(\phi(t),\Theta) \\ \phi(t) &= [y^{*T}(t+d)\ldots y^{*T}(t+d-m_c+1) \\ &\qquad \hat{y}^T(t+d-1)\ldots\hat{y}^T(t+d-m_a) \\ &\qquad u^T(t-1)\ldots u^T(t-m_b)]^T \end{aligned} \tag{4.37}$$

where $\hat{y}(t+d)$ and $u(t)$ are used as shorthand notations for $\hat{y}(t+d|\Theta)$ and $u(t|\Theta)$.

Assuming a common selection $m_a = n_f$, $m_b = n_b - 1$ and $m_c = 1$ for notational convenience, the system (4.36) ... (4.37) can be presented in the state space form

$$
\begin{aligned}
x(t+1) &= F(x(t), y^*(t+d)) \\
x(t+1) &= \begin{bmatrix} [\quad 0 \quad] \\ [\ I_A\ ] \\ [0] \\ [I_B]\ [0] \end{bmatrix} x(t) + \begin{bmatrix} \tilde{f}(x(t), \tilde{h}(x(t), y^*(t+d))) \\ [0] \\ \tilde{h}(x(t), y^*(t+d)) \\ [0] \end{bmatrix} \\
\hat{y}(t+1) &= [x_1(t+1) \ldots x_m(t+1)]^T
\end{aligned}
\tag{4.38}
$$

where possible I_A and I_B are $m(n_f-1) \times m(n_f-1)$ and $m(n_b-2) \times m(n_b-2)$ dimensional identity matrices. The functions $\tilde{f}$ and $\tilde{h}$ denote f and h with reordered input vectors, simularly as in (3.69). The state is

$$
x(t) = [\hat{y}^T(t+d-1) \ldots \hat{y}^T(t+d-n_f)\ u^T(t-1) \ldots u^T(t-n_b+1)]^T \tag{4.39}
$$

The Jacobian of (4.38) using the parameters of the MFD presentation (4.33) is

$$
J_C = F'(x)
$$

$$
J_C = \begin{bmatrix} (A_1 + B_1 S_1) \ \ldots\ (A_{nf} + B_1 S_{nf})\ (B_2 + B_1 R_1)\ \ \ldots\ (B_{nb} + B_1 R_{mb}) \\ [\ I_A\ ] \\ S_1 \ldots \qquad\qquad \ldots S_{nf} \qquad\quad R_1 \ldots \qquad\qquad \ldots R_{mb} \\ [\ I_B\ ] \qquad\quad [0] \end{bmatrix} \tag{4.40}
$$

This resembles the AB-condition (3.73) i.e. it takes into account also the denominator of the linearized controller and closed loop models. This is unpractical and simpler conditions are needed. Assume, analogously to Hernandez and Yarkun (1992), that a perfect control is applied i.e. $\Lambda = 0$ and

$$
\begin{aligned}
y^*(t+d) &= \tilde{f}(x(t), \tilde{g}(x(t), y^*(t+d))) \\
E(q^{-1}) y^*(t+d) &= E(1) r(t) \\
E(q^{-1}) \hat{y}(t+d) &= E(1) r(t)
\end{aligned}
\tag{4.41}
$$

i.e. $y^*(t) = \hat{y}(t) \quad \forall t$. The local existence and uniqueness of such feedback law can be proved applying the Implicit Function Theorem, e.g. Rosenlicht (1968), resulting in practical conditions

$$
\det \{\partial \hat{y}(t+d) / \partial u(t)\} \neq 0 \quad \text{(locally)} \tag{4.42}
$$

The condition (4.42) should be assured during the identification phase, resulting now in the local existence of an unique feedback law. The use of the closed loop reference model does not affect to this, it just relaxes the assumption of the stable inverse of the process model. Now the Jacobian (4.40) can be written as

$$J_{C_1} = \begin{bmatrix} E_1 \ldots & 0 \ldots & 0 \\ [\ I_A\ ] & & \\ S_1 \ldots \ldots S_{nf} & R_1 \ldots & \ldots R_{mb} \\ & [\ I_B\ ] & [0] \end{bmatrix} \tag{4.43}$$

where E_1 is the poles of the closed loop reference model (4.3). So, if a perfect control is applied, the remaining dynamics of interest are that of the controller, which is closely related to the inverse dynamics or zero dynamics, e.g. Isidori (1989), obtained with $E(q^{-1}) = I$. Different controllers are obtained with different $E(q^{-1})$ and it is clear that the assumption of a stable inverse can be somewhat relaxed by a suitable selection of $E(q^{-1})$.

Analogically to (3.75), it can be shown that the eigenvalues of J_{C_1} consist of zeroes, poles of the diagonal reference model $E(q^{-1})$ and eigenvalues of

$$J_{C_2} = \begin{bmatrix} R_1 \ldots R_{mb} \\ [I_B] \end{bmatrix} \tag{4.44}$$

The instability detectors and the projection cost functions can now be written analogously to that presented in Section 3.4. The AB-condition monitors whether (4.43) is a contraction i.e. whether $\|J_{C_1}\|_\infty \geq 1$. The B-condition monitors whether (4.44) is a contraction i.e. whether $\|J_{C_2}\|_\infty \geq 1$ and the ρ-condition whether the spectral radius $\rho(J_{C_2}) \geq 1$.

In practice the ρ-condition is the most useful, especially because the controller has typically $m_b = 1$ and the detection and the projection is trivial in the SISO case and simple in the TITO case.

This perfect control ($y^*(t) = \hat{y}(t) \quad \forall t$) is not always obtained in practice. Also the RPEM does not produce a perfect control during the convergence. In theory, the stability should be maintained using (4.40) which is not practical. In the SISO case the characteristic polynomial

$$P(q^{-1}) = A(q^{-1})R(q^{-1}) - q^{-1}B(q^{-1})S(q^{-1}) = 0 \tag{4.45}$$

where

$$P(q^{-1}) = I + P_1 q^{-1} + \ldots + P_{nf+mb}\, q^{-nf-mb}$$

can be used instead, resulting in conditions

$$\sum_{i=1}^{nf+mb} |P_i| \geq 1 \tag{4.46}$$

and

$$|\text{roots}(P(q))| \geq 1 \tag{4.47}$$

In the projection phase these are combined together and used with similar conditions for the controller, resulting in the cost function

$$V(\Theta) = \frac{1}{2}\sum_{t=1}^{N} [\|\varepsilon(t,\Theta)\|_Q^2 + \|v(t,\Theta)\|_H^2] \tag{4.48}$$

$$\text{with } v_1(t,\Theta) = \begin{cases} (v_1 - \eta_1) \text{ if } v_1 = \sum_{i=1}^{nf+mb} |P_i(t)| \geq \eta_1 \text{ and } |\text{roots}(P(q))| \geq 1 \\ 0 \text{ , otherwise} \end{cases}$$

$$v_2(t,\Theta) = \begin{cases} (v_2 - \eta_2) \text{ if } v_2 = \sum_{i=1}^{mb} |R_i(t)| \geq \eta_2 \text{ and } |\text{roots}(R(q))| \geq 1 \\ 0 \text{ , otherwise} \end{cases}$$

where H is a diagonal weighting matrix. Originally $\eta_1 = \eta_2 = 1$, but also other values can be used, for example $\eta_2 < 1$ to reduce the ringing effect caused by the controller. Also $\eta_1 > 1$ could be used to relax the first condition. By varying H and especially η the resulting control system behaviour can be remarkably affected. Also in the imperfect case it is often sufficient to project only the controller using either B- or ρ-condition. The closed loop stability should still be verified after the design.

A global existence of the feedback law which produces (4.41) is another case. It requires the system to be globally I/O linearizable. All dynamical systems cannot be I/O- or feedback linearized. Conditions for feedback linearization are presented by Levin and Narendra (1993). In general, the conditions are very hard to verify and one has no other resource in practical situations, except to assume that they are valid and apply the linearization procedure.

For nonlinearizable systems, typically unstable or nonminimum phase systems, the model reference control is not useful. Levin and Narendra (1993) proposed a class of direct stabilizing neural network controllers (feedback laws), which are designed so that (4.38) is a contraction in a selected small closed ball near the equilibrium. The underlying idea is to design a k-step-ahead deadbeat controller. The control error is computed according

$$\varepsilon = \begin{cases} x(k), \text{ if } \| x(0) - x_{eq} \| < r, \text{ or } \| x(k) - x_{eq} \| > \gamma \| x(0) - x_{eq} \| \\ 0, \text{ otherwise} \end{cases} \tag{4.49}$$

where r is the radius of the ball and γ is the contraction coefficient. As it can be seen, the model reference property is removed. The cost function for the minimization is collected with successive simulations starting from randomly selected initial states $x(0)$.

This clearly resembles the N-condition, proposed in (3.78) i.e.

$$\| x(t) - x(t-1) \| \geq \| x(t-1) - x(t-2) \| \tag{4.50}$$

The approach is suitable for stabilizing unstable systems near a constant setpoint, but difficult to apply for a changing setpoint. Remarkable computational savings are achieved compared to the dual network approach (4.48).

Another proposal, (Hernandez and Yarkun 1992), somewhat ties together the dual network approach (4.48), (4.49) and the direct predictive control approach (4.8). The authors suggest a "p-inverse" feedback law, which is a special p-step-ahead predictive controller. The control $u(t)$ is computed by minimizing the unconstrained cost function

$$V(t, u) = \frac{1}{2} \| y^*(t+p) - \hat{y}(t+p) \|^2 \tag{4.51}$$

or more precisely a LRPC with $N1 = N2 = p$, $Nu = 1$, $\Lambda = 0$. Also $E(q^{-1}) = I$, i.e. no model reference is applied.

They proved that near a stable equilibrium point, if certain assumptions hold, mainly (4.42), there always exists (locally) a feedback law (4.37) which stabilizes the system in p steps (with p large enough). This is true regardless of the stability of the inverse dynamics. The selection of p is not straightforward, because the spectral radius (4.44) is not necessarily a monotonically decreasing function of p. The implementation was performed with the LRPC approach. They demonstrated the control system behaviour controlling a nonminimum phase nonlinear CSTR model.

The previous is also one demonstration that neural network controller can be designed according any LPRC cost function, at least in the unconstrained case. This is perhaps unpractical, because one can always first solve the LPRC task with the direct approach and then design a controller according the result. The synthetic signal method can be used for actual controller design (Fig. 4.1). A disadvantage is that the "learning" of constraints is difficult. Also some stability problems may arise.

The proposed dual network IMC approach forms a flexible and robust controller design framework, resulting in a controller which can be implemented with minor computational load. It can thus be applied also to high speed control systems.

5 Simulation Studies

This chapter presents simulation results of neural network identification and control. Simulation is an excellent tool for methodological development and testing. Typically a simple nonlinear process model is used as a test target to get insight, how the selected control/ identification method behaves. Most of the uncertainties of true processes are neglected and their effect is tested separately by adding one type of uncertainty at a time.

As necessary and as useful these simulation studies are, they do not tell the "whole story": how the selected identification or control approach will perform when applied to a real process. Real time tests or at least simulation studies using a detailed and complex process model are also needed.

The amount of the performed simulation experiments during this study is overwhelming, but only a few will be presented here due to the space limitations and because the main interest has been on investigating the performance of real-time neural network control.

Before going into simulation examples, the setup for neural network identification, simulation and for real-time experiments is presented.

5.1 Experimental Setup

The essential properties of the environment for developing, testing and implementing neural network control methods are

- flexible and efficient process simulator
- flexible and efficient mathematical software
- possibility to add own code (C- or Fortran subroutines)
- possibility for real-time use or for real-time extensions.

Several alternatives are available, just to mention two: Matlab/Simulink and Math_X/ Matrix_X. Even a Neural Network Toolbox is available for Matlab, although its suitability for time series modelling is limited.

It is more important to have a good process simulator and a good mathematical software package than a software for neural network representation and simulation. Most neural network types and training methods can be easily be presented using matrix operations and the "neural network" part of the overall environment is a moderately easy task.

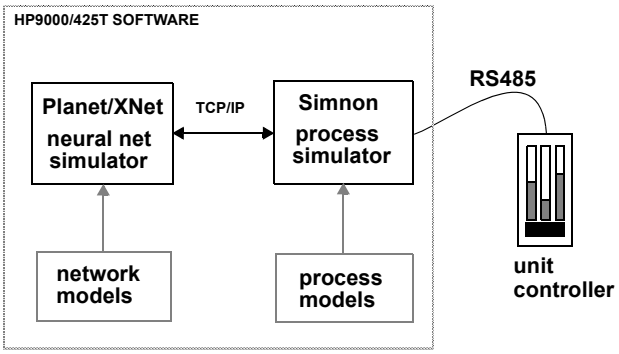


Fig. 5.1. The software block schema.

A UNIX workstation version of a commercial dynamical process simulator Simnon (1991) was selected as a basic software block. Simnon is connected to a neural network software, Planet/XNet (Miyata 1990). The main power of XNet is its flexible nature, all definitions are made using Matlab like matrix notation. Both programs are modified by adding the necessary communication routines (C-code). The software block schema is presented in Fig. 5.1. The communication between Simnon and XNet is implemented using tcp/ip -sockets, which allows the programs to run on separate computers. This forms an efficient environment. Most of the experiments presented in Chapter 6 were performed with this setup.

The lack of higher level mathematical subroutines limited somewhat the *development* of new identification and control methods. Every needed mathematical subroutine has to be added to the source code. In that sense a package like Matlab is superior and immediately after the availability of Matlab mex-file interface and Simulink in HP9000/425 worksta-

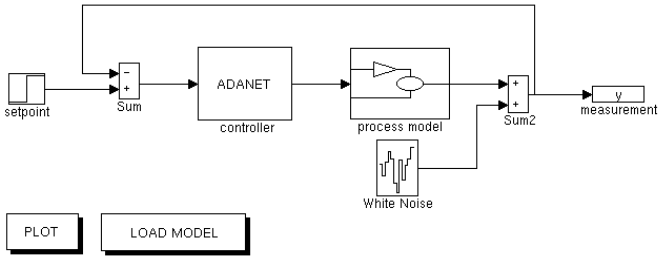


Fig. 5.2. A Simulink model for adaptive neural network control.

tion, the overall neural network environment was transferred and developed within it.

This forms now the main platform of neural network identification, process simulation and control design tasks. Simulink is not used for real-time control experiments, although it is possible. The slow computational speed and the lack of appropriate user interface do not meet the specifications set for real-time control.

The final software package consist of three parts, all within HP9000/720 or HP9000/425T workstations

- Matlab functions (partly mex-files) for off-line tasks like identification, simulation and controller design of MIMO neural network time series models and controllers. Resulting networks can be used as Simulink modules.

- Software (C-code) for adaptive and non-adaptive neural network long range predictive control (LRPC). This package is named to "ADANET" and it can be used as a module in Simulink model (Fig. 5.2) or as a part of real-time experimental setup.

- Real-time experimental setup: "neuro-control workstation" which consists of neural network modules, communication modules for measurement devices controllers (HP-IB and RS485 interfaces) and a commercial user interface tool HP VeeTest.

Both systems have been used efficiently for developing and testing neural network iden-

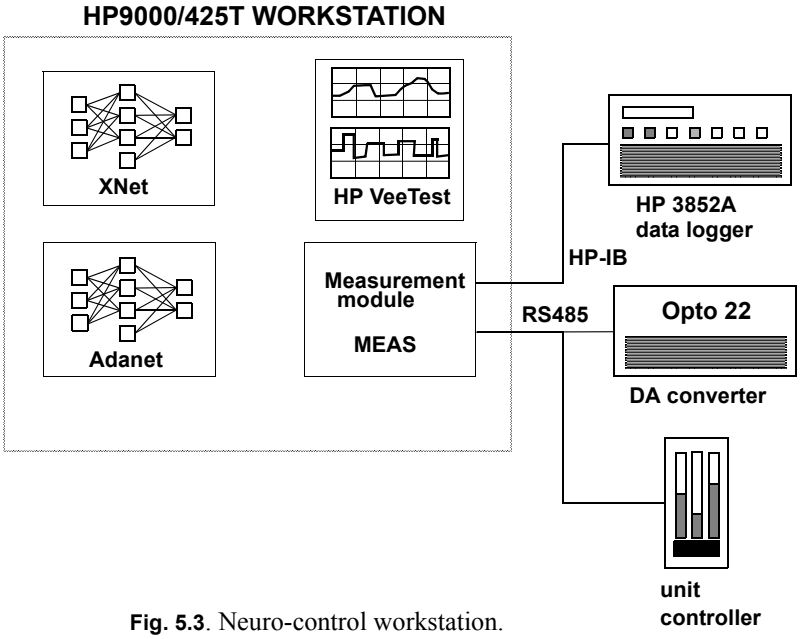


Fig. 5.3. Neuro-control workstation.

tification and control algorithms and for real-time control experiments.

5.2 Simulation Examples

Just a couple of identification examples and one control example are considered, because the main interest is on real-time experiments. Standard identification tasks are omitted. The results reported in the literature clearly show that smooth nonlinearities can be represented with an MLP network. The presented simulation studies consider more complicated tasks, namely identification in the presence of nonstationary noise and identification of a multistep predictor.

The control system simulation example demonstrates the basic properties of the LRPC and dual network control (IMC) approaches, especially from the robustness point of view.

Example 5.1 "Nonlinear Dynamics"

This example considers identification of the deterministic part of the process, especially in the presence of trend like disturbances. Consider the system

$$\begin{aligned} \dot{x}_1(t) &= -\omega_n^2 x_2(t) - 2\omega_n x_1(t)\left|\sin(\pi\omega_n^2 x_2(t))\right| + \omega_n^2(u(t) - 0{,}5) \\ \dot{x}_2(t) &= x_1(t) \\ y(t) &= x_2(t) + 0{,}5 \end{aligned} \tag{5.1}$$

where $\omega_n = 0{,}5$ and $0 \le u \le 1$. The steady state behaviour of the example system is linear but the dynamical behaviour is nonlinear. The model is a second order system, with the "damping factor" $\left|\sin(\pi\omega_n^2 x_2)\right|$ i.e. the open loop poles are on the imaginary axis if

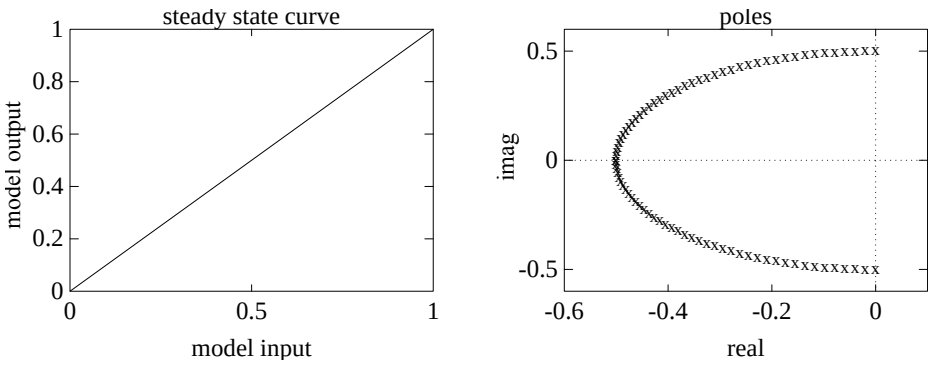


Fig. 5.4. Steady state characteristic curve of the model and the poles of the linearized model, linearized at steady states $0 \le u \le 1$.

$x_2 = 0$ and $y = 0{,}5$, see Fig. 5.4.

This system is stable, but only with a narrow margin at some operation points. It is selected to study how well neural network time series models can cope with nonlinear dynamical behaviour and to give some insight into the stability projection methods.

Consider first the case of stationary measurement noise i.e. uniformly distributed noise in $[-0{,}02,\ 0{,}02]$ is added to the measurement. For the identification, the process is driven with a Pseudo Random Sequence (PRS) type input signal covering all operation regions. The identification data set of 1000 samples with $\Delta T = 1$ sec. is collected. The simulated noiseless open loop step responses are used as the test set. For the identification, the measurement and the input signal are scaled between $\pm 0{,}5$, although original scales are used in most figures. This is a standard procedure in this study, if not otherwise stated. The variables are scaled so that the full operation region corresponds $[-0{,}5,\ 0{,}5]$.

The NOE predictor

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\hat{y}(t)\ \hat{y}(t-1)\ u(t)\ u(t-1)]^T \end{aligned} \tag{5.2}$$

is implemented with an MLP network. After some trials the structure is selected as

layers	2 hidden, 6 nodes each
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	no

The size of this network is quite big because of the complex dynamics of the process. The network weights are initialized with uniformly distributed random values in $[-0{,}1,\ 0{,}1]$. The parameters of the network are identified with the LM method.

The second identification experiment is made by driving the process with similar PRS type input signal as before, but adding an Integrated Moving Average (IMA) noise to the measurement and collecting a data set of 1000 samples. The IMA noise is obtained by integrating normally distributed white noise (variance 10^{-4}). This data set is presented in Fig. 5.5.

As a reference, the plain NOE predictor is identified using the same network structure (5.2) as in the stationary case. The main interest is to identify the deterministic part of the process using appropriate noise model i.e. assuming the process to be of NOE-IMA type, see Section 4.2.

This predictor can be presented as

$$\begin{aligned} \tilde{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\tilde{y}(t)\ \tilde{y}(t-1)\ u(t)\ u(t-1)]^T \\ \hat{z}(t+1) &= y(t) - \tilde{y}(t) \\ \hat{y}(t+1) &= \tilde{y}(t+1) + \hat{z}(t+1) \end{aligned} \tag{5.3}$$

This predictor can be identified by minimizing the cost function

$$V_N(\theta) = \frac{1}{2} \sum_{t=1}^{N} \| y(t) - \hat{y}(t) \|^2 \tag{5.4}$$

which requires the gradient $\partial \hat{y}(t+1) / \partial \theta$ to be computed according the predictor formulation (5.3) to obtain true PEM estimates. Identical, but more convenient approach is to minimize the cost function

$$V_N(\theta) = \frac{1}{2} \| y(1) - \tilde{y}(1) \|^2 + \frac{1}{2} \sum_{t=2}^{N} \| \Delta y(t) - \Delta \tilde{y}(t) \|^2 \tag{5.5}$$

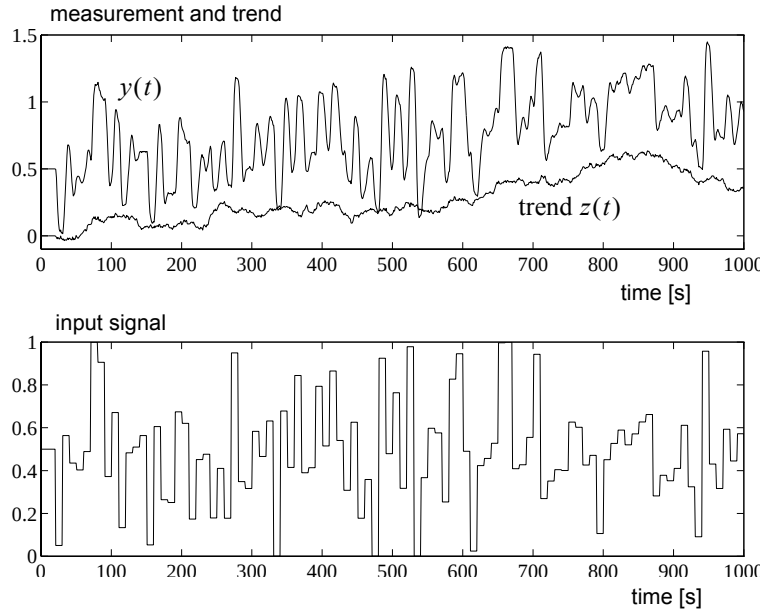


Fig. 5.5. The identification data set. The measurement is corrupted with IMA noise.

which can be implemented with only slight modification of the plain NOE identification procedure. In practice also the initial state $\varphi(0)$ and $\hat{z}(0)$ must be considered as unknown parameters. This is omitted in this simulation example, because the IMA noise starts at zero, see Fig. 5.5.

The NOE-IMA predictor is implemented with a MLP network with the same structure as before and the parameters of the predictor are identified with the LM approach.

The behaviour of the deterministic part of the identified predictors with the test set is presented in Fig. 5.6. As expected the NOE predictor identified with the presence of the stationary noise gives best results (Fig. 5.6a). Also the NOE-IMA predictor identified with the presence of IMA noise performs well (Fig. 5.6c). The plain NOE predictor with IMA noise does not give the correct responses neither in static nor in dynamic sense (Fig. 5.6b). This can more clearly be seen if the prediction is shifted i.e. $\hat{y} - 0{,}3$ is plotted (Fig. 5.6d).

The behaviour of the plain NOE predictor with the presence of the IMA noise is unpredictable, it depends heavily on the actual noise i.e. totally different behaviour could be obtained with different identification data. On the other hand, the NOE-IMA predictor gives the same or almost the same responses despite of the identification data, at least if

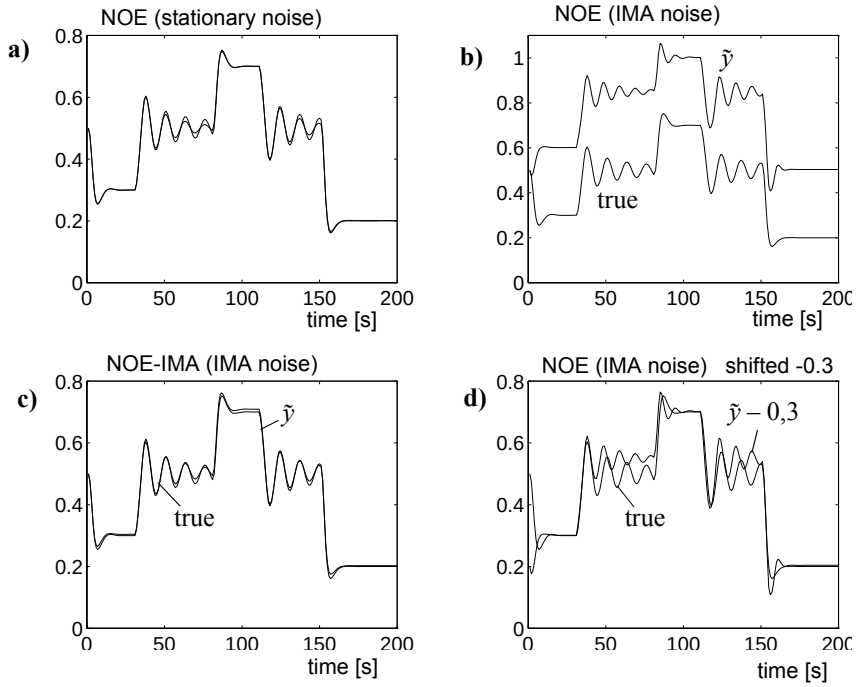


Fig. 5.6. Step responses of the deterministic part of the identified NOE predictors. d) is the same as b), but shifted $\tilde{y} = \tilde{y} - 0{,}3$.

the number of samples is large enough i.e. the true deterministic part of the process is obtained as a result of the identification.

Additive and other trends are common in practice. The solution for the identification problems is simple in the linear case: use incremental measurement and input signals. The presented simulation example demonstrates that also in the nonlinear case the solution is straightforward: use incremental cost function.

This approach must be used with some care, because the gradient based minimization of the cost function (5.5) converges to a local minimum. The NARX or NOE predictors should be used to get adequate initial values for the parameters.

One inconvenient feature is that monitoring the cost (5.5) does not clearly indicate whether the minimization has converged. The overall prediction error $y(t) - \hat{y}(t)$ is small and the error $y(t) - \tilde{y}(t)$ does not tell much about the convergence. It is just an estimate of the trend $z(t)$. Better measures can be obtained by using the correlation analysis of the residuals i.e. if the prediction error does not correlate with the input signal, no further minimization is necessary. The framework for nonlinear correlation analysis can be found in Billings and Voon (1986) and Billings and Zhu (1994). The other possibility is to use a separate test set and stop the identification when the residuals of the test set does not decrease any more.

The correlation measures should be used for validation of any nonlinear model. These have not been used much in this study, because the result of the correlation analysis: plots from 4 to 6 correlation functions, is difficult to interpret to get insight how the model structure or model orders should be changed.

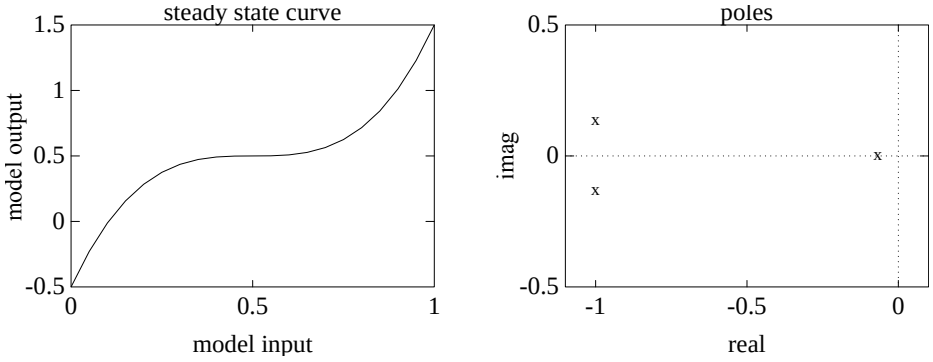


Fig. 5.7. Steady state characteristic curve of the model and the poles of the linear part. (Example 5.2)

Example 5.2 "Unmodelled Dynamics"

The second example addresses the problem of identifying a NARX predictor for multistep prediction, as is normally done in long range predictive control. This example also demonstrates the effect of unmodelled dynamics. Consider the system

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} -2{,}0667 & -1{,}511 & -0{,}0679 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 8 \\ 0 \\ 0 \end{bmatrix} (u(t) - 0{,}5)^3 \\ y(t) &= \begin{bmatrix} 0 & 0 & 0{,}0679 \end{bmatrix} x(t) + 0{,}5 \end{aligned} \tag{5.6}$$

with $0 \le u \le 1$. This system has a static nonlinearity and linear third order dynamics with one dominating time constant (real pole at -0.0667) and lightly damped fast mode (complex pole pair at $-1 \pm 0{,}1339j$), see Fig. 5.7. The system is selected mainly to study the effect of unmodelled dynamics i.e. the system is modelled with a first order nonlinear time series model. The static nonlinearity makes the control difficult, because at steady state $u = 0{,}5$ the gradient $\partial y/\partial u = 0$. The linear part of the model is similar to the well-known "Rohr's example", e.g. Shook *et. al.* (1991), which is used to study the robustness of linear adaptive GPC.

Uniformly distributed noise in $[-0{,}02,\ 0{,}02]$ is added to the measurement. The discretization time is chosen as one second.

The first order NARX predictor

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [y(t)\ u(t)]^T \end{aligned} \tag{5.7}$$

is implemented with an MLP network

layers	2 hidden, 5 nodes each
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	yes

The process model is driven with PRS+PRBS type of input signal in $0{,}1 \le u \le 0{,}9$ and the data set of 3000 samples is collected. First a normal one-step-ahead NARX predictor is identified by minimizing a constrained cost function

$$\begin{aligned} V_N(\theta) &= \frac{1}{2} \sum_{t=1}^{N} \| y(t) - \hat{y}(t) \|^2 \\ & \partial \hat{y}(t)/\partial u(t-1) \ge 0{,}005 \\ & |\partial \hat{y}(t)/\partial \hat{y}(t-1)| \le 1 \end{aligned} \tag{5.8}$$

with RPEM approach i.e. with the stability constraints and input gradient limit. The stability of the NARX predictor is not necessary for the convergence of RPEM. However, ensuring the stability is useful in the case of recursive multistep prediction.

The 8-multistep-ahead NARX predictor is identified with the RPEM approach using the incremental cost function with similar constraints as before

$$\begin{aligned} V_{LRPI} &= \frac{1}{2}\sum_{t=1}^{N}\sum_{i=t-7}^{t} \left\| y(i) - \hat{y}(i|t-8) \right\|^2 \\ &\partial\hat{y}(i)/\partial u(i-1) \geq 0{,}005 \qquad , i = t-7 \ldots t \\ &\left|\partial\hat{y}(i)/\partial\hat{y}(i-1)\right| \leq 1 \qquad , i = t-7 \ldots t \end{aligned} \tag{5.9}$$

where $\hat{y}(i|t-8)$ denotes the prediction $\hat{y}(i)$ calculated recursively using actual measurement at time step $(t-8)$.

The performance of the resulting models is presented in Fig. 5.8, where the input step responses at three different operation points $u = 0{,}5 \rightarrow 0{,}6$, $u = 0{,}65 \rightarrow 0{,}75$ and $u = 0{,}75 \rightarrow 0{,}85$ are presented. The normal NARX predictor does exactly what it should do, predicts one step ahead correctly. The multistep NARX is more efficient in prediction further into the future. The corresponding relative prediction errors, e.g. Grabec (1991)

$$E_{rp}(i) = \frac{1}{Var(y)N}\sum_{t=1}^{N} \left(y(t-i+1) - \hat{y}(t-i+1|t-8)\right)^2 \quad , i = 1 \ldots 8 \tag{5.10}$$

which are computed using a test set of 1000 data points are presented in Fig. 5.9. Again the prediction capability of multistep NARX can be seen. Note that the E_{rp} is not equally distributed w.r.t the prediction horizon $i = 1 \ldots 8$ .There is no reason why relative prediction error should be equally distributed here, the identification procedure just minimizes the cost function (5.9).

Th approach is efficient when identifying too low order models but it can be used successfully also if the model orders are correct. If also the noise model assumption is correct (NARX), no benefits are gained. If not, the approach result in a compromise between plain NARX and a NOE predictor. This compromize is often a suitable model for the LRPC approach.

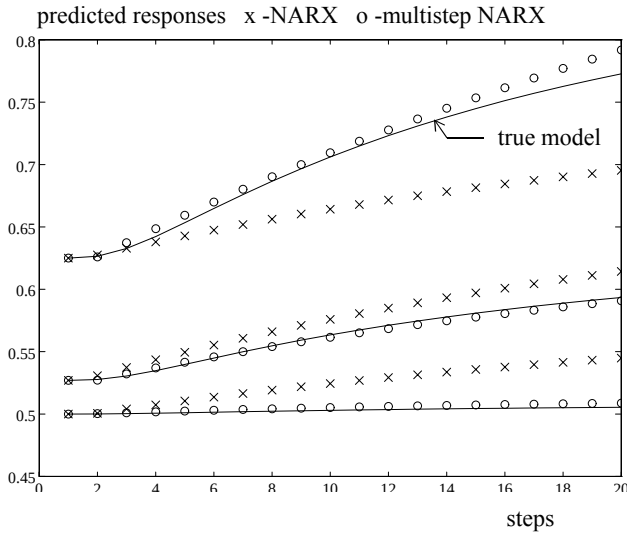


Fig. 5.8. The step responses of the identified NARX predictors.

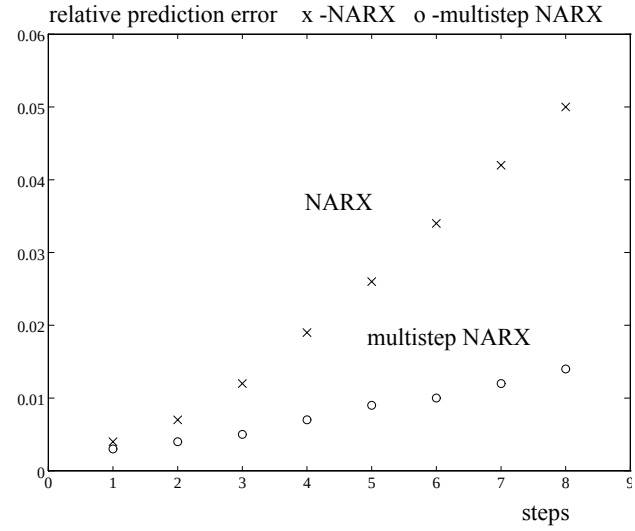


Fig. 5.9. Relative prediction error E_{rp} as a function of the prediction horizon.

Example 5.3

This example considers control of the process in Example 5.1. The identified NOE model is used for the dual network controller design according to the guidelines presented in Section 4.3. The controller of the form

$$\begin{aligned} u(t) &= h(\phi(t), \Theta) \\ \phi(t) &= [u(t-1)\ \hat{y}(t)\ \hat{y}(t-1)\ y^*(t+1)]^T \end{aligned} \tag{5.11}$$

is realized using an MLP network. The network structure is selected as

layers	2 hidden, 6 nodes each
activation function	hidden: *tanh*, output: *tanh*
linear short-cut connections	no

The controller is designed by minimizing the cost function

$$V(\Theta) = \frac{1}{2} \sum_{t=1}^{N} [\| y^*(t+d) - \hat{y}_f(t+d) \|^2 + \| \Delta u(t) \|_\Lambda^2] \tag{5.12}$$

where

$$E(1)\ \hat{y}_f(t+d) = E(q^{-1})\ \hat{y}(t+d) \tag{5.13}$$

with the LM approach without any stability projection. After some trials, the design parameters are selected as $E(q^{-1}) = 1 - 0{,}5q^{-1}$ and $\Lambda = 10^{-3}$.

The ramp type setpoint sequence for the controller design is selected because it removes the steady state control error of the constant setpoint (Fig. 5.10a-b). This is analogical to that a PI-controller, which has zero steady-state error with a constant setpoint, could have a constant steady-state error with a ramp type setpoint. A couple of step wise setpoint changes are added at the end of the design set for instability detection and projection purposes. Note that the activation function of the output layer of the controller is *tanh(·)* i.e. in $[-1, 1]$, but the whole operation region is $[-0{,}5, 0{,}5]$. If a constrained case is needed, the activation function should be changed to 0.5*tanh(·)*.

The performance of the resulting controller-model loop is presented in Fig. 5.10. The eigenvalues of the Jacobian matrix of the controller (controller "poles", Fig. 5.10c) and of the closed loop model (closed loop "poles", Fig. 5.10d) clearly indicate that both the controller and the closed loop system are possibly unstable (see Section 3.4 and Section 4.3). This can be seen more clearly in the input signal behaviour of the test set (Fig. 5.10f). This ringing phenomenon could somewhat be reduced by loosening the desired closed loop behaviour $E(q^{-1})$ and increasing the control signal weighting Λ, but not totally removed.

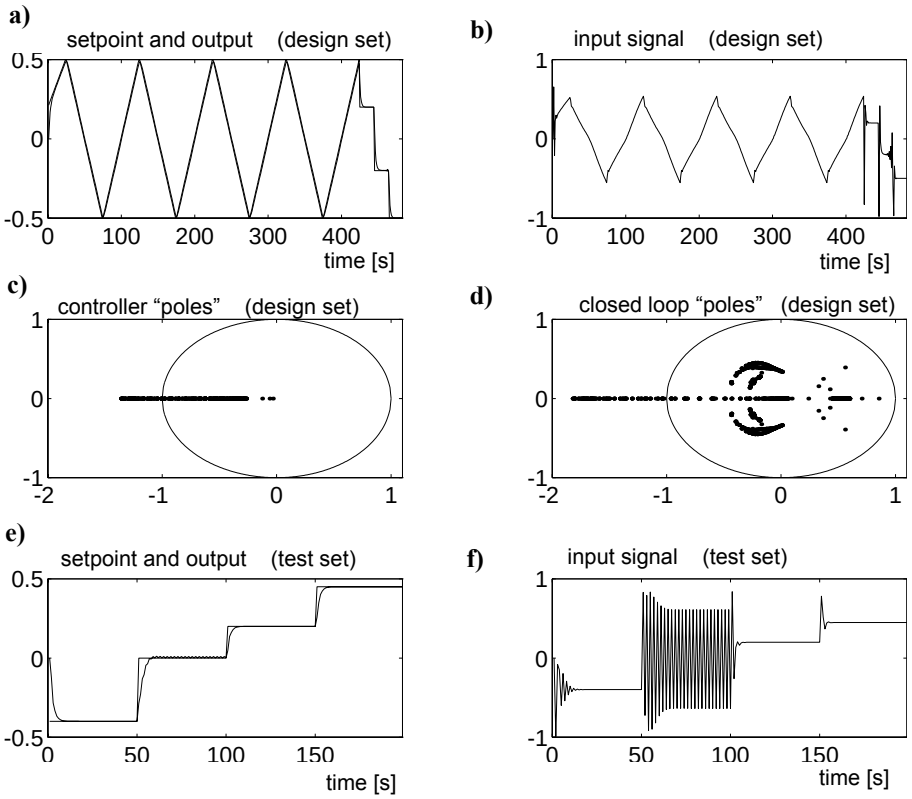


Fig. 5.10. The performance of the dual network control when controlling the identified process model. No stability projection is applied. Note the "internal scaling".

To obtain a stable control system, the closed loop "poles" are projected inside the unit circle using the constrained cost function (4.48) with $H = \text{diag}\{h_1, 0\}$ and $\eta_1 = 1$ i.e. the controller dynamics is not constrained.

The performance of the resulting controller-model loop is presented in Fig. 5.11. Both the controller "poles" and the closed loop "poles" are in the stable domain (Fig. 5.11c-d) and the ringing phenomenon has disappeared from the control signal (Fig. 5.11f).

Incorporating the stability constraints within the optimal controller design seems to be an efficient way to quarantee the stability of the control system still maintaining its control performance. As discussed in Section 4.3, also a justifiable solution is to project only the controller poles or the poles and the zeroes. This detection/projection is much simpler to implement.

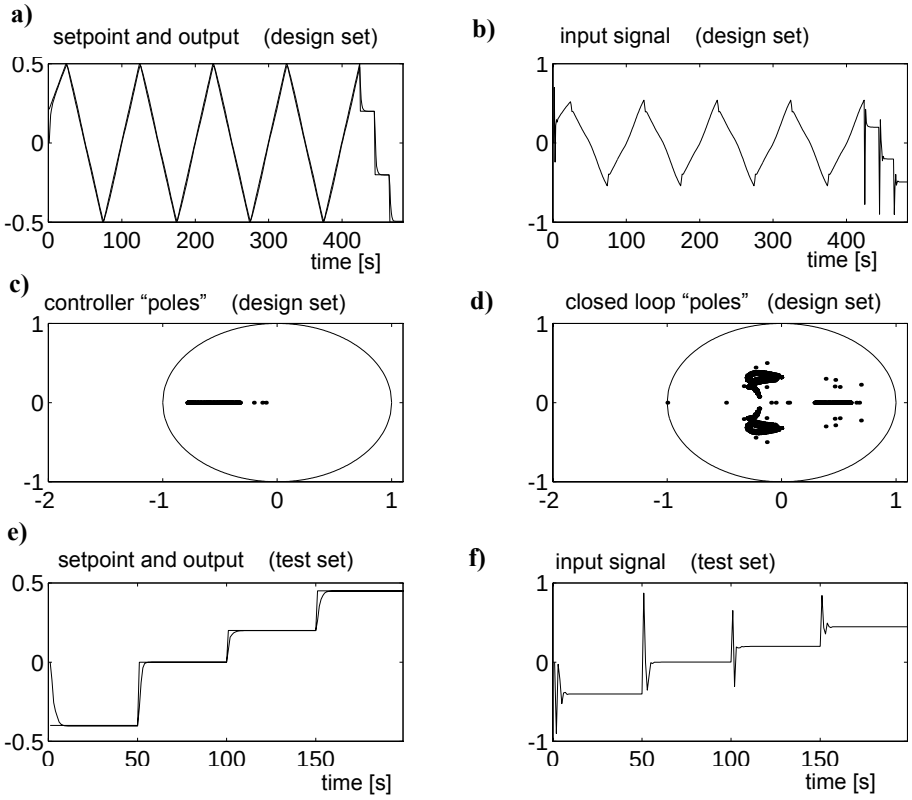


Fig. 5.11. The performance of the dual network control when controlling the identified process model. Closed loop poles are projected inside the unit circle.

To compare the designed dual network controller to direct predictive control, the identified neural network model is also used in the NOE type LRPC approach to control the neural network model using the same design parameters $E(q^{-1})$ and Λ as in the dual network control case and without any model mismatch considerations. As discussed in Section 4.3, the selection $N1 = N2 > 1$ and $Nu = 1$ should give suitable results.

Indeed, the selection $N1 = 1$, $N2 = 2 \ldots 3$, $Nu = 1$ or $N1 = N2 = 2$ $Nu = 1$ gives control performance comparable to Fig. 5.11, but the selection $N1 = 1$, $N2 = 1$, $Nu = 1$ or $N1 = 1$, $N2 = 2$, $Nu = 2$ gives similar control performance as in Fig. 5.10, i.e. heavy ringing phenomenon in the control signal.

These results are in accordance with the "*p*-inverse" feedback law (Hernandez and Yarkun, 1992). Thus the optimal control signals could be solved with the LRPC approach, and after that a neural network controller could be designed so that it produces these control signals. However, based on a couple of trials, this type of identification is rather difficult in a constrained case (input constraints).

Only the behaviour of the controller-model loop has considered so far and especially from the stability point of view. To analyse also the robustness issues, the designed controller is implemented as the dual network IMC approach (Fig. 4.9) with the filters

$$F_1(z) = 1 \text{ and } F_2(z) = \frac{0{,}3z}{z - 0{,}7}$$

The control system is simulated with Simulink using the original continuous time model (5.1) as the process. The robustness of the control system is tested using both input and measurement additive IMA noise obtained by integrating normally distributed white noise with variance 10^{-4}.

The use of trend like disturbances with the process which is near the stablity border posses serious difficulties on control system. The use of input disturbances is also somewhat unfair, because the used IMC filter is designed for output additive noise, see Section 4.2. The results should be considered from the robustness point of view.

The performance of this control system is presented in Fig. 5.12. The control system handles the output disturbances quite well, but it has serious difficulties with the input disturbances near the stability border i.e. near $y \sim 0{,}5$.

This IMC approach is also implemented using the direct NOE-LRPC approach (Fig. 4.9). The selection $N1 = 1$, $N2 = 2$ and $Nu = 1$ with the same design parameters as above gave results comparable to Fig. 5.12. This indicates also that (without input constraints) it is quite the same whether the IMC approach is implemented with the dual network approach or with the direct NOE-LPRPC.

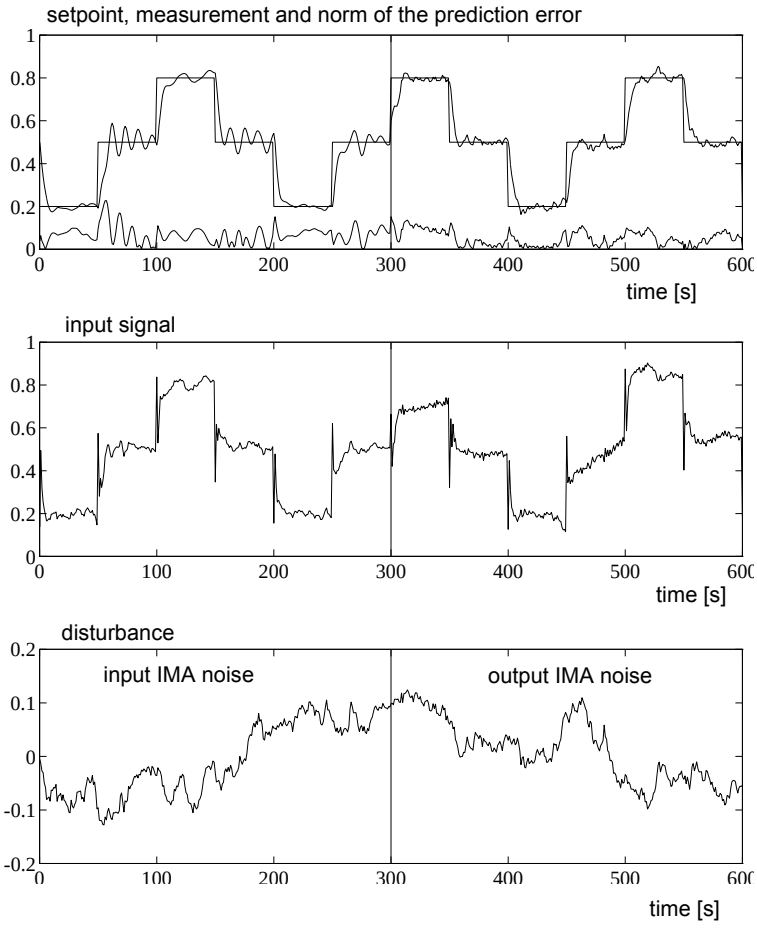


Fig. 5.12. The performance of the implemented control system. Dual network control with first order IMC filter.

The input noise rejection capability can be increased by designing a higher order IMC filter i.e. assuming the equivalent system to be controlles as, see (5.13)

$$z^{-1}\tilde{E}(z) = \frac{0{,}5z^{-1}}{1 - 0{,}5z^{-1}} \tag{5.14}$$

and designing a linear IMC controller for a ramp type disturbance. This results in

$$F_2(z) = (3{,}2 - 4{,}2z^{-1} + 1{,}3z^{-2})\frac{0{,}3}{1 - 0{,}7z^{-1}}$$

The performance of this control system is presented in Fig. 5.13, now with NOE-LRPC approach. Exactly the same noise sequence as in Fig. 5.12 is used to make the comparison easier. The problems with the input disturbances are reduced but not totally removed. This is obvious, because even the second order IMC filter does not correspond to the true noise model. The use of second order filter also increases the control signal variations when the presence of the output disturbances.

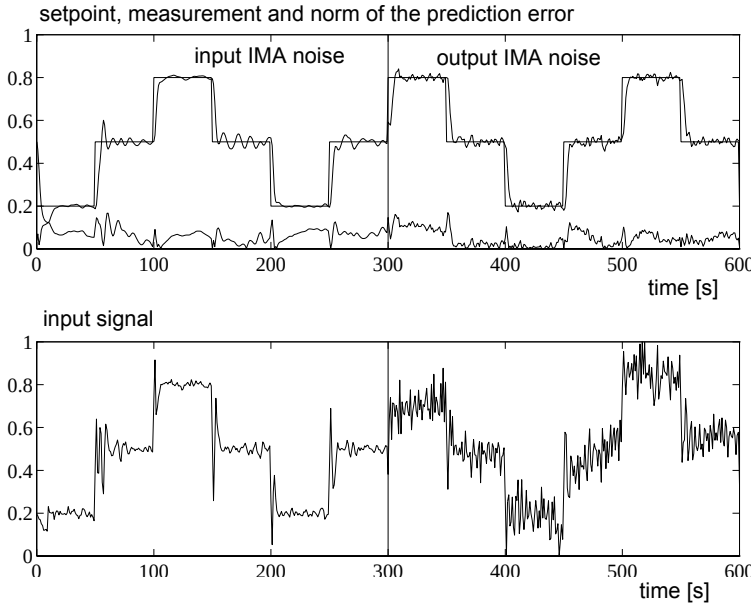


Fig. 5.13. The performance of the implemented control system. The LRPC approach with the NOE model and with the second order IMC filter.

Because the identified NOE model is almost an exact representation of the true process, see Fig. 5.6a, here it can be straightforwardly applied as a NARX predictor. The LRPC approach with NARX predictor demonstrates this. The design specifications are kept same i.e. $C(q^{-1}) = 1 - 0{,}7q^{-1}$. The performance of this control system is presented in Fig. 5.14. The same disturbance sequence as before is used in simulations. The main difference between this and the previous simulations is that the input disturbances are handled more efficiently and the performance under the output disturbances is not worse than those obtained with the IMC approach. It should be noted that also here an incorrect noise model assumption is used.

This is a first example indicating that NARX predictor combined with LRPC approach forms an efficient and robust control system. Other examples will be seen in next Chapters. The NARX predictor is easy to identify and the effect of the design parameters is clear. The only drawback is the high computational on-line load. It is not claimed that a NOE model with carefully designed state estimator would not show similar performance or robustness, but it is claimed that in many cases the LRPC-NARX approach is much easier to apply.

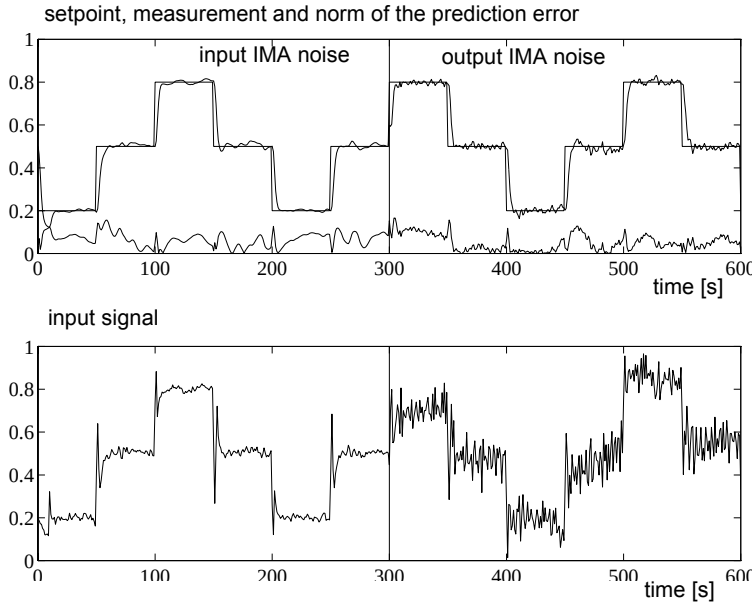


Fig. 5.14. The performance of the implemented control system. The LRPC approach with the NARX predictor.

6 Control of Water Heating Process

Chapters 6, 7 and 8 consider the real-time neural network control experiments performed with several laboratory scale pilot processes. The main goal was to examine the performance of neural network modelling, identification and control algorithms with real processes which incorporate common uncertainties: nonlinearities, model mismatch, time varying features, time delays, noise, load disturbances and trends.

The example processes incorporate all these aspects, with a minor limitation: a heavily nonlinear pilot process was not available. All presented results consider mild nonlinearities. This is not a serious limitation, because it is well known and demonstrated in numerous articles, that neural networks can represent a heavy nonlinearity quite well. More important is how the neural network control system performs with the uncertainties of the real world. The available test processes are quite suitable for this purpose.

This chapter considers identification and control of two slightly different water heating process denoted “Heating process I” (HP I) and “Heating process II” (HP II). The water flows from the domestic water network into an uninsulated 0.4 litre tank through a pipe, which has holes in it, see Fig. 6.1. The water is heated by a resistor element and the temperature of the outlet flow is measured with a Pt-100 transducer. The inlet flow q_{in} can be set between 0 – 3.0 l/min. with a rotameter.

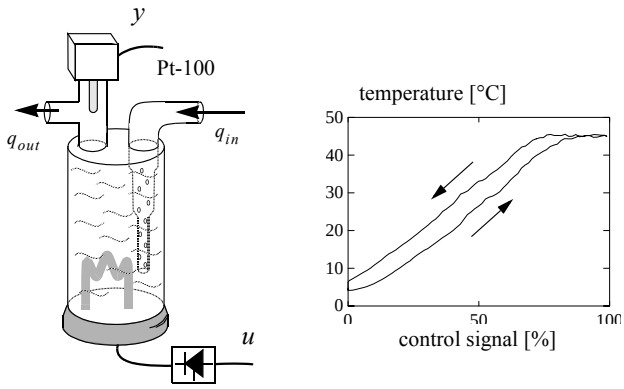


Fig. 6.1. The heating process and the measured open loop response (HP I), obtained by driving the process with a 30 min. ramp both up and down. The measurement $y(t)$ is plotted directly as a function of $u(t)$.

The aim of the control is to drive the temperature of the outlet flow to the desired value. Robustness of the temperature controller is essential since the dynamics of the process are extremely sensitive to the flow changes. Also the time-delay of 12...15 seconds has to be taken into account in control design. The dominating time constants are about 100 seconds (HP I), and 30 seconds (HP II). The main difference between processes is that the process I has a smaller heating element than the process II. From the control point of view, the heating process I is more difficult to control. The experiments were made during several seasons of the year and the water inlet temperature varied from 5°C to 20°C. Thus all test should not be straightforwardly compared.

6.1 Identification

The heating process II was modelled using data gathered from a real time identification run. The system was driven with a PRS type input signal 9000 seconds with $\Delta T = 3$ s., resulting in 3000 samples. The data was divided into an identification set of 2000 samples and a test set of 1000 samples.

The model orders and the delay were determined by identifying a series of linear ARX models. The ARX predictor with $n_a = 2$ and $n_b = 2$ gave the best estimate with the delay $d = 4$. The bias was used in all linear models. The deterministic part of the system is of interest and the corresponding OE model was also identified.

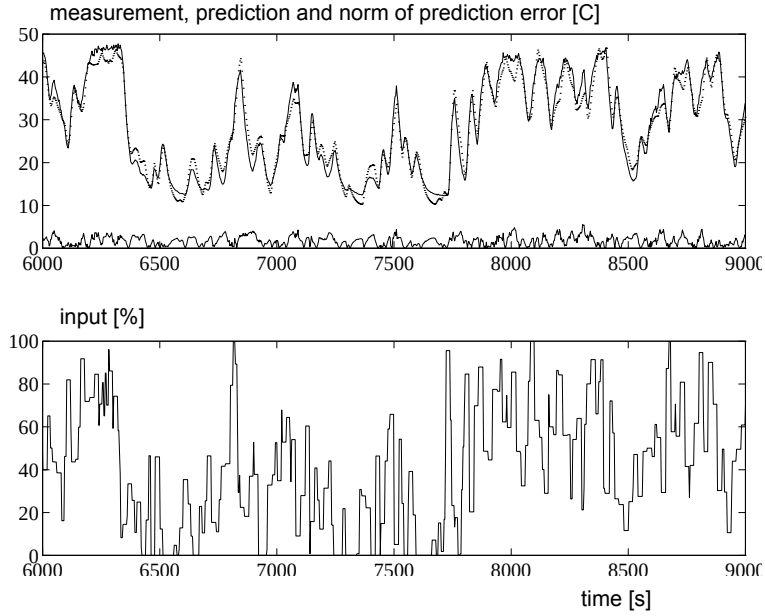


Fig. 6.2.The behaviour of the linear OE predictor with the test set. PRS type input signal.

The resulting behaviour with the test set is presented in Fig. 6.2. The largest prediction error occurs in outer operation ranges which is mainly due to the saturating phenomenon of the thyrisor.

The corresponding d-step-ahead NARX and NOE predictors

$$\hat{y}(t) = f(\varphi(t), \theta) \tag{6.1}$$

with

$$\varphi(t) = [y(t-1)\ y(t-2)\ u(t-4)\ u(t-5)]^T \quad \text{(NARX)}$$

$$\varphi(t) = [\hat{y}(t-1)\ \hat{y}(t-2)\ u(t-4)\ u(t-5)]^T \quad \text{(NOE)}$$

were also identified.

The identification was initially made at the same time as the dual network control experiments (Section 6.3) using a rather large network structure. The models were identified using RPEM approach. Later, the identification was reperformed using the LM method. The goal was to study how small a network is actually needed. The used networks were:

	"large" network	"small" network
layers	2 hidden, 5 nodes each	1 hidden, 5 nodes
activation function	hidden: *tanh*, output: *linear*	hidden: *tanh*, output: *linear*
short-cut connections	yes	no
learning method	RPEM off-line	LM off-line

The overall performance of the identified models seem to be almost identical. The amount of the parameters is significantly different: 65 versus 31 parameters. A remarkable aspect is that the overparametrization does not have such a strong effect as happens when overparametrizing linear models. This is also due to the good quality of the identification data, it covers the whole operation region of interest. If a large MLP network is used outside the trained domain, the performance would probably not be so good.

Table 6.1. Prediction errors (MSSE) for different model structures, "small" network.

Model	Identification set
linear ARX	0.57
linear OE	7.48
NARX	0.34
NOE	0.85 (constrained 0.92)

The results presented here are obtained using "small" networks. The cost function values are presented in Table 6.1. The difference between OE and NOE models is significant which clearly implies that the heating system is nonlinear. This can also be seen from the steady state characteristic curve of the NOE model (Fig. 6.3), which clearly indicates a saturation type nonlinearity. The steady state curve is linear in the input ranges 30...70%, but the dynamical behaviour is not, see Fig. 6.2.

An important aspect is that the steady state curve of the model is not monotonically increasing, contrary as expected from the physical basis of the process. This is due to the lack of the identification data at higher temperatures. Although small, this phenomena causes extreme difficulties, when the identified model is used in predictive control or in controller design.

The constrained cost function

$$V_N(\theta) = \frac{1}{2}\sum_{t=1}^{N} \|y(t) - \hat{y}(t)\|^2$$
$$\text{w.r.t.} \sum_{i=4}^{5} \partial\hat{y}(t)/\partial u(t-i) > 0 \tag{6.2}$$

was minimized to force the steady state curve to be monotonically increasing, successfully, as can be seen from Fig. 6.3. and with only minor increase in the cost function value: 0.85 versus 0.92.

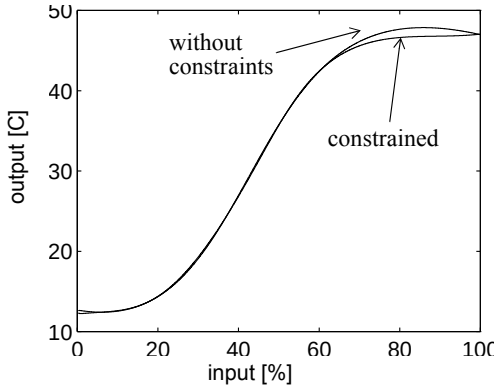


Fig. 6.3. The steady state characteristic curve of the identified NOE models (HP II).

The behaviour of the resulting NOE model with the test set is presented in Fig. 6.4. For comparison, also the NARX predictor was tested as a deterministic process model. The difference between the NOE model and the NARX model is small but clear. This probably indicates that also noise entering the system via NARX style is present. The only clear difference between the "large" and "small" network was in this test. The "large" NARX predictor did not show as good performance when applied as a NOE model.

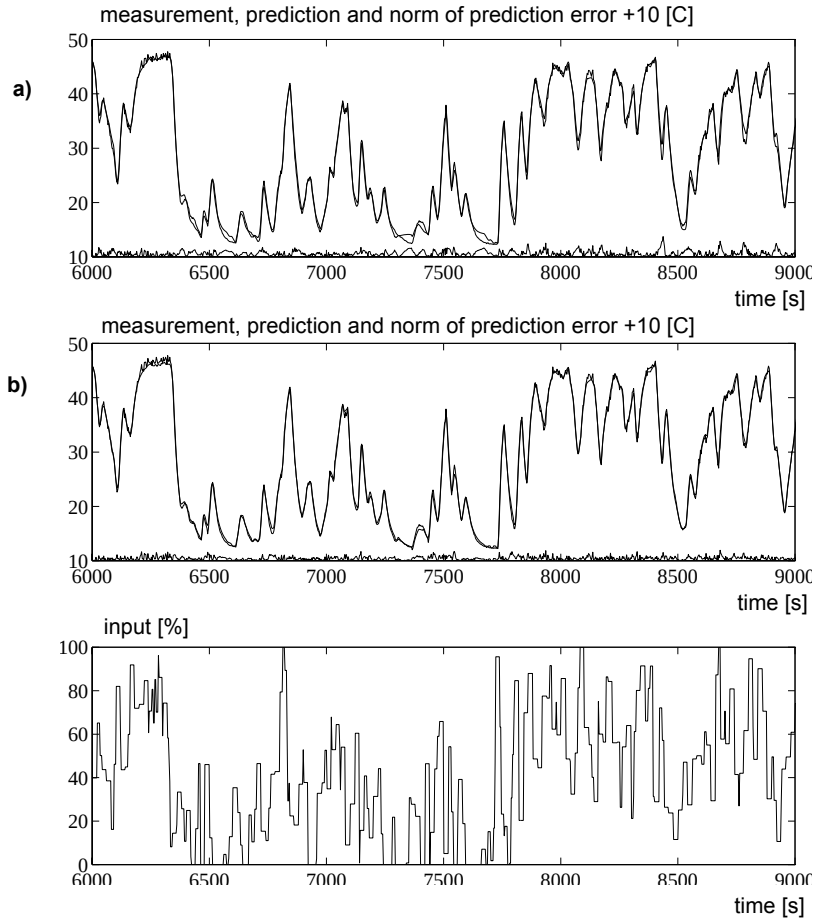


Fig. 6.4. The NARX and NOE predictors as a deterministic process models (HP II). a) NARX as NOE b) NOE.

6.2 Adaptive Control

Several simulation and real-time experiments concerning adaptive nonlinear control have been made. Results obtained when applying adaptive neural network control to the water heating processes will be presented below. These examples are selected to discuss basic problematics and typical behaviour of this type of control. They should not be considered as final solutions.

The first example considers the adaptive LRPC of the heating process I using a direct predictor, i.e. a combined approach, see Section 3.2. The main goal of this part was to study the behaviour of different combined predictors i.e whether a sparse network structure is needed. The identification was performed with the error-backpropagation algorithm. To compensate the slow adaptation, a slow desired response was selected.

If $d > 1$ and a direct predictor is used, the model reference is difficult to apply. One possibility is to take the closed loop dynamics into account already in the identification i.e. identifying directly

$$y_f(t) = E(q^{-1})y(t) \tag{6.3}$$

After selecting the LRPC parameters as $N1 = 4$, $N2 = 6$ and $Nu = 1$, and selecting the desired closed loop response according to $E(q^{-1}) = 1 - 0{,}95q^{-1}$, the predictors can be written as

$$\begin{bmatrix} \hat{y}_f(t+4) \\ \hat{y}_f(t+5) \\ \hat{y}_f(t+6) \end{bmatrix} = f(\varphi(t), \theta)$$
$$\varphi(t) = [y(t)\ y(t-1)\ u(t+2) \ldots u(t-1)]^T \tag{6.4}$$

This predictor was realized with an MLP network with the structure

layers	1 hidden, 33 nodes (total)
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	yes

The non-causal connections were removed and the size of the input data vector for each prediction is balanced to be equal. The combined predictor (6.4) was in fact divided into three separate networks, each with 11 nodes in the hidden layer.

This corresponds to the predictor equations

$$\begin{bmatrix} \hat{y}_f(t+4) \\ \hat{y}_f(t+5) \\ \hat{y}_f(t+6) \end{bmatrix} = \begin{bmatrix} f_1([y(t)\ y(t-1)\ u(t)\ u(t-1)]^T, \theta_1) \\ f_2([y(t)\ y(t-1)\ u(t+1)\ u(t)]^T, \theta_2) \\ f_3([y(t)\ y(t-1)\ u(t+2)\ u(t+1)]^T, \theta_3) \end{bmatrix} \tag{6.5}$$

This predictor was used with the control law

$$V(t,u) = \frac{1}{2} \sum_{i=N1}^{N2} [\| E(1)r(t+i) - \hat{y}_f(t+i) \|^2] + \frac{1}{2} \| \Delta u(t) \|_\Lambda^2 \tag{6.6}$$

with $\Lambda = 0{,}02$ and $\Delta u_{max} = 10\,\%$ (largest allowed change). The control error due to the model mismatch was handled by adaptation, i.e. no $C(q^{-1})$ was used, see (4.14) and (4.15). The cost function (6.6) was minimized using the steepest descent method.

The network was initialized with random weights in $[-0{,}1,\ 0{,}1]$ and the standard error backpropagation with the learning rate of 0.4 and the momentum term of 0.05 was used. A series of setpoint changes covering the whole operation region were performed. A load disturbance test with fixed weights was performed after that to study the robustness of the resulting control system.

A detailed discussion of various sparse and non-sparse predictor structures can be found in Kimpimäki (1990) and a shorter overview in Koivisto *et. al.* (1991). The results indicate that the control system with the sparse predictor structure performs clearly better.

The performance of the adaptive LRPC control with the predictor structure (6.5) is presented in Fig. 6.5. As it can be seen, it took a while before the measurement reached the setpoint. The closed loop behaviour converged almost to the desired response. Some differences can however be seen. The response was slow due to the selected closed loop pole and the overall performance could easily be outperformed by other control methods.

A remarkable aspect is that no steady state control error is present and that the resulting controller shows good robustness properties during the load disturbance test, although the disturbance rejection could be faster.

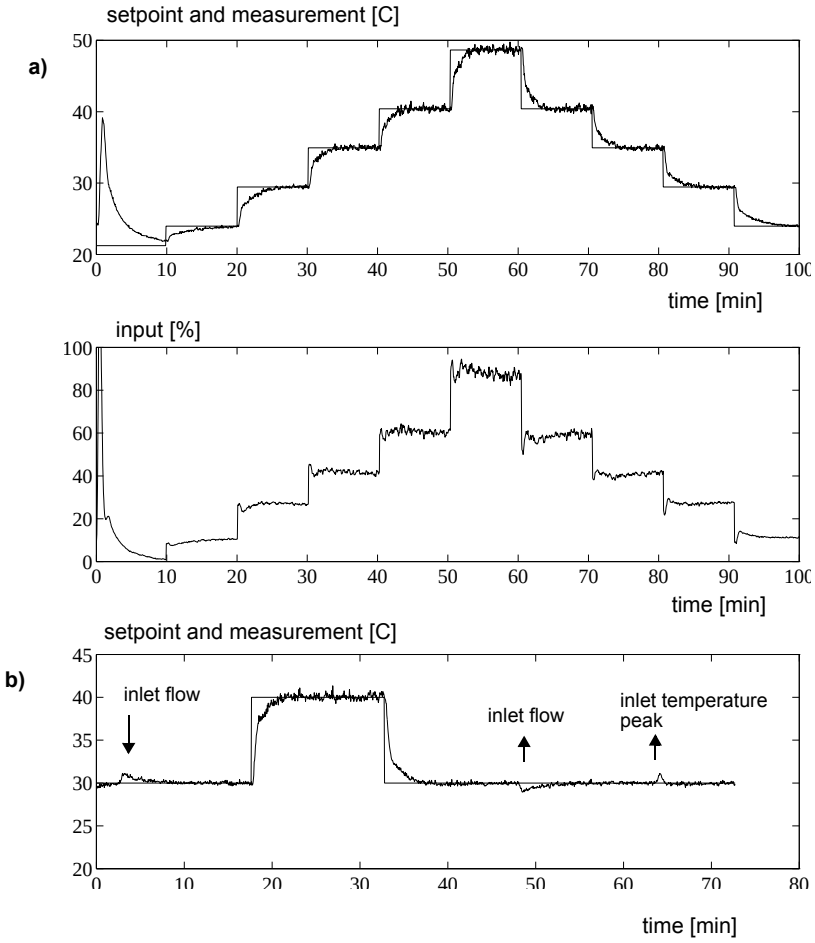


Fig. 6.5. The control system performance with the adaptive LRPC, a direct combined NARX predictor, adaptation with error backpropagation, a) setpoint tracking, b) load disturbance test with fixed parameters. (HP I)

The performance of the adaptive LRPC with a tighter closed loop model is presented in Fig. 6.6, now with a recursive NARX predictor

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [y(t)\ y(t-1)\ u(t-3)\ u(t-4)]^T \end{aligned} \tag{6.7}$$

which was realized with an MLP network

layers	1 hidden, 10 nodes
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	yes

The LRPC control cost function (4.8)

$$V(t,u) = \frac{1}{2} \sum_{i=N1}^{N2} \left[\left\| E(1)y^*(t+i) - E(q^{-1})\hat{y}(t+i) \right\|^2 + \left\| \Delta u(t+i-N1) \right\|_\Lambda^2 \right] \tag{6.8}$$

with $N1 = 4$, $N2 = 9$, $Nu = 2$, $\Lambda = 10^{-3}$ and $E(q^{-1}) = 1 - 0{,}7q^{-1}$, was minimized with Rosen's gradient projection method (Soeterboek 1990).

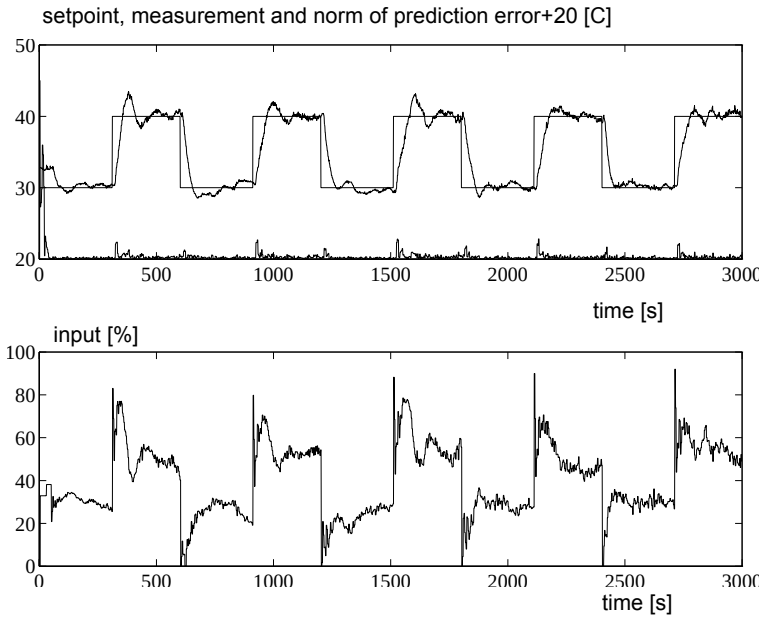


Fig. 6.6. The control system performance with the adaptive LRPC, a recursive NARX predictor, adaptation with the error backpropagation (HP I).

The network starts at random weights in $[-0{,}1,\ 0{,}1]$ The identification was performed using the error backpropagation with the learning rate $0{,}1$ and by filtering the gradients with $0{,}5/(1-0{,}5q^{-1})$.

The use of a recursive predictor caused in this case a strong control signal variation which was reduced with $\Lambda = 0{,}01$ and projecting the gradient $\partial y(t+1)/\partial u(t-3) > 0{,}25$. The gradient limitation reduced also the start-up overshoot.

Tighter specifications ($E(q^{-1})$) changed the overall performance considerably, most important was the decrease of the overall robustness. It took a while (2000 sec = 700 control cycles) to meet the desired closed loop response, when switching only between two setpoint values (Fig. 6.6). The use of a stair wise setpoint sequence as in Fig. 6.5 resulted in much longer adaptation time.

The overall performance (start-up behaviour) was as expected, but not acceptable. The results are presented only to demonstrate the adaptation speed of the error backpropagation algorithm and to get a reference for the RPEM identification, which will be considered next.

RPEM identification

A typical behaviour of the adaptive LRPC approach with the RPEM identification is presented in Fig. 6.7. The NARX predictor

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [y(t)\ y(t-1)\ u(t-3)\ u(t-4)]^T \end{aligned} \tag{6.9}$$

was realized with an MLP network

layers	2 hidden, 5 nodes each
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	yes

The control design parameters were selected as $N1 = 4$, $N2 = 9$, $Nu = 2$, $\Lambda = 10^{-3}$ and $E(q^{-1}) = 1 - 0{,}7q^{-1}$. The gradient $\partial y(t+1)/\partial u(t-3)$ was limited above to $0{,}05$. The network was initialized with random weights $[-0{,}1,\ 0{,}1]$. The UD factored version of (3.47) was used and the initial covariance matrix was set to $P(0) = 100 \cdot I$.

The forgetting factor was computed recursively using the formula

$$\lambda(t) = 0{,}995\ \lambda(t-1) + 0{,}005 \quad \text{and} \quad \lambda(0) = 0{,}95 \tag{6.10}$$

The start-up behaviour in Fig. 6.7 is typical for many adaptive neural network control experiments. All operation points should be visited once before adequate control behaviour is achieved.

The start-up behaviour depends heavily on the ordering of the setpoint sequence and on the initial weights, which is obvious: recursive Gauss-Newton algorithm converges to the nearest local minimum.

The forgetting factor and/or covariance resetting did not remove the problem of the initial weights. The input gradient and stability projection methods helped much but they did not totally remove the problem. Obviously some preliminary off-line identification must be performed in order to achieve reasonable initial weights.

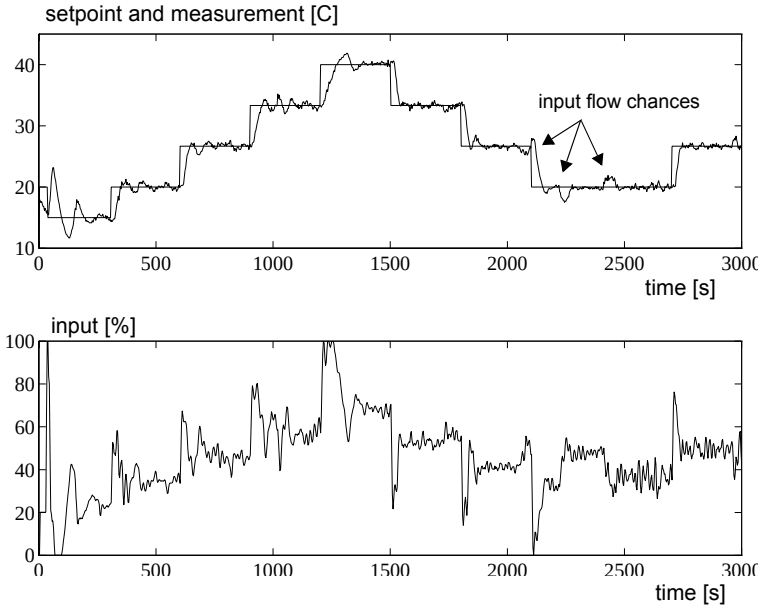


Fig. 6.7. The control system performance with the adaptive LRPC, a recursive NARX predictor, adaptation with the RPEM (HP I).

Adaptive Dual Network Control

This experiment considers the adaptive dual network control, now using Heating Process II. It is presented mainly because the resulting model was used for off-line controller design (Section 6.3), but also to demonstrate that fully adaptive dual network control is possible, even in practice.

The NOE type process model and the NOE type controller were identified using two totally separate and parallel RPEM algorithms. The controller was designed on-line using the approach presented in the Section 4.3. The IMC filters were not used, i.e. the control error due to the model mismatch was handled by adaptation.

The NOE predictor

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\hat{y}(t)\ \hat{y}(t-1)\ u(t-4)\ u(t-5)]^T \end{aligned} \tag{6.11}$$

was realized with an MLP network

layers	2 hidden, 5 nodes each
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	yes

The controller

$$\begin{aligned} u(t) &= h(\phi(t), \Theta) \\ \phi(t) &= [u(t-1)\ \hat{y}(t+4)\ \hat{y}(t+3)\ y^*(t+5)\ y^*(t+4)]^T \end{aligned} \tag{6.12}$$

was realized with an MLP network

layers	2 hidden, 10 nodes each
activation function	hidden: *tanh*, output: *tanh*
linear short-cut connections	yes

Both networks were initialized with random weights in $[-0{,}1,\ 0{,}1]$ and identified with the RPEM approach using similar covariance and forgetting factor as in the previous case. The control cost function was

$$V(\Theta) = \frac{1}{2}\sum_{t=1}^{N} [\| y^*(t+5) - \hat{y}_f(t+5) \|^2 + \| \Delta u(t) \|_\Lambda^2] \tag{6.13}$$

and

$$E(1)\,\hat{y}_f(t+5) = E(q^{-1})\,\hat{y}(t+5) \tag{6.14}$$

where $y^*(t+5) = r(t)$, $E(q^{-1}) = 1 - 0{,}7q^{-1}$ and $\Lambda = 0{,}01$. The gradient $\partial\hat{y}(t+5)/\partial u(t)$ was limited above 0.005 and the pole of the controller was projected between $[-0{,}8,\ 1]$.

Fig. 6.8 shows that the performance of the control system is good. The successful start-up (after the very beginning) is mainly due to the suitable initial weights, but also due to the identification of NOE type model and controller with the RPEM approach using proper gradient computation. Because no IMC filters were used, the noise rejection was not as good as possible. This can clearly be seen from the control signals.

This experiment merely demonstrates the convergence of the RPEM approach under suitable conditions. Also this control system was sensitive to the initial weights and setpoint sequences. Typically one out of five performed this way, one was somewhat a failure, and the rest showed control performance comparable to Fig. 6.7.

The dual network control is computationally light when applied with constant parameters but computationally heavy in an adaptive application. The difference of the computational load of the adaptive LRPC is not big any more. The usage of LRPC is perhaps more motivated in practice.

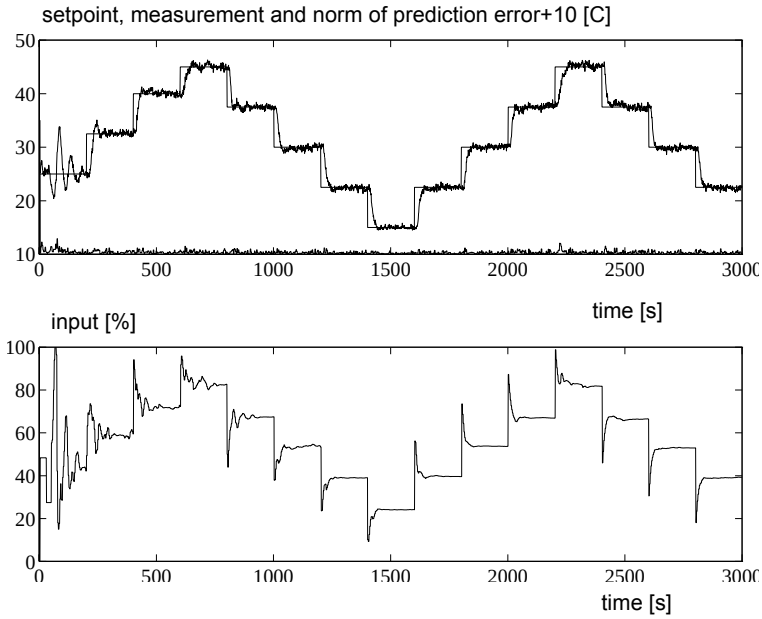


Fig. 6.8. The control system behaviour with the adaptive dual network control, model and controller adaptation with RPEM, HP II.

6.3 Dual Network Control - Constant Parameters

The model and controller obtained from the adaptive dual network control experiment was used as a basis for the redesign of the controller to obtain a tighter performance. The overall control system performance can be made fast or slow by a suitable selection of the IMC filters $F_1(z)$ and $F_2(z)$. However a tight tuning could cause instability problems and in practice only loosening the performance with the IMC filters is safe. Thus the goal of the controller design is to obtain a response as fast as possible within the internal controller-model loop still maintaining stability and adequate control signal variations. After that the overall control system performance can be set with the IMC filters.

The controller redesign was made by changing the pole of the closed loop reference model from $0{,}7 \rightarrow 0{,}2$ i.e. by minimizing the cost function (6.13) with the RPEM approach using $E(q^{-1}) = 1 - 0{,}2q^{-1}$ and the same setpoint sequence as in Fig. 6.8. The controller pole was again limited between $[-0{,}8,\ 1]$. The limit $-0{,}8$ was used instead -1 to avoid ringing effect in the control signal. The redesign gave a stable controller-model loop which was implemented with the filters $F_1(z) = 1$ and $F_2(z) = 0{,}3z/(z - 0{,}7)$

Several weeks after the model identification the performance of the control system was tested in real-time experiments. During this time the properties of the process have been changed due to the variations of the inlet flow temperature and room temperature. This caused a significant modelling error. The performance of the control system is shown in Fig. 6.9. The step responses show that in spite of the mismatch between the process and the model the overshoot is very small and the response is fast. Moreover, the control is insensitive to noise mainly because of the selected IMC filter. Due to the physical limitations of the actuator and the use of the control penalty Λ, the response defined by $E(q^{-1})$ cannot be achieved, it is not even the main target. The effect of these limitations can also be seen by comparing the behaviour of the model with the signal y^* (Fig. 6.9). The behaviour of the controller-model loop shows that the controller satisfies the requirements defined by the cost function.

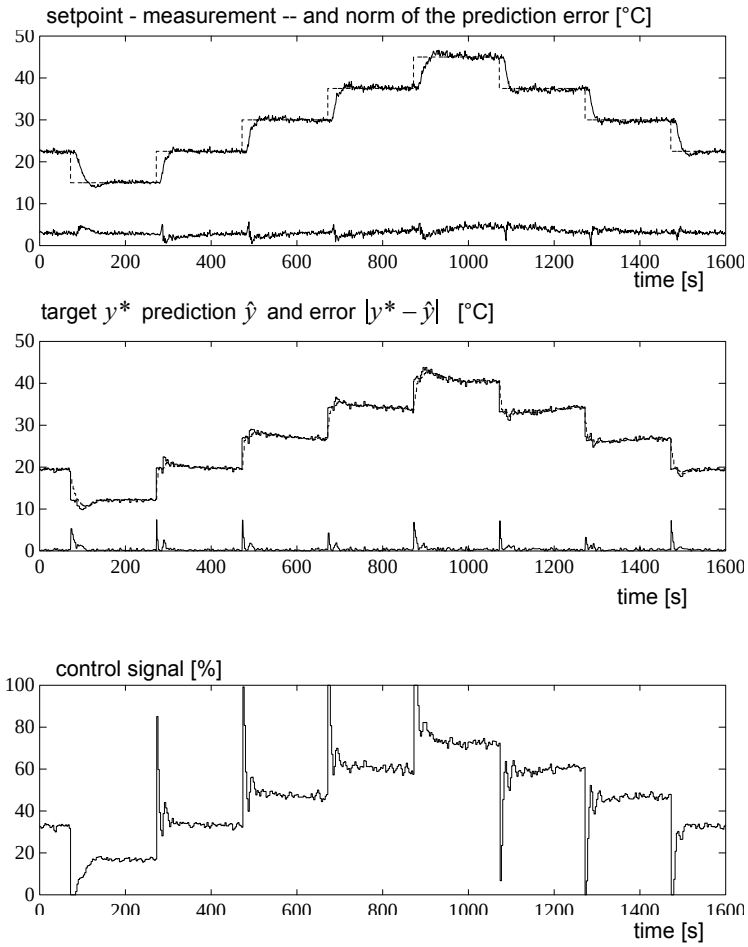


Fig. 6.9.The control system performance with the dual network control during the setpoint tracking test (HP II).

The performance of the control system was also tested with an unmeasured step disturbance. Disturbances are generated by changing the inlet flow according to the following sequence:

$$q_{in}(t) \approx \begin{cases} 1{,}0 \text{ l/min}, & 0 \le t < 80 \text{ s} \\ 1{,}5 \text{ l/min}, & 80 \text{ s} \le t < 300 \text{ s} \\ 0{,}5 \text{ l/min}, & 300 \text{ s} \le t < 1100 \text{ s} \\ 1{,}0 \text{ l/min}, & 1100 \text{ s} \le t \le 1200 \text{ s} \end{cases}$$

The nominal inlet flow is 1.0 l/min. From the behaviour of the prediction error (Fig. 6.10) after the flow change, it can be seen that the load disturbances affect the measurement through slow dynamics. Since the load disturbances have slow internal dynamics and the IMC incorporates disturbances through prediction error feedback, the resulting flow change rejection is sluggish. Better disturbance compensation could be achieved by paying more attention to the selection of the filter F_2. Overall, the IMC controller achieves good performance and the control is robust in the presence of significant modelling errors and disturbances.

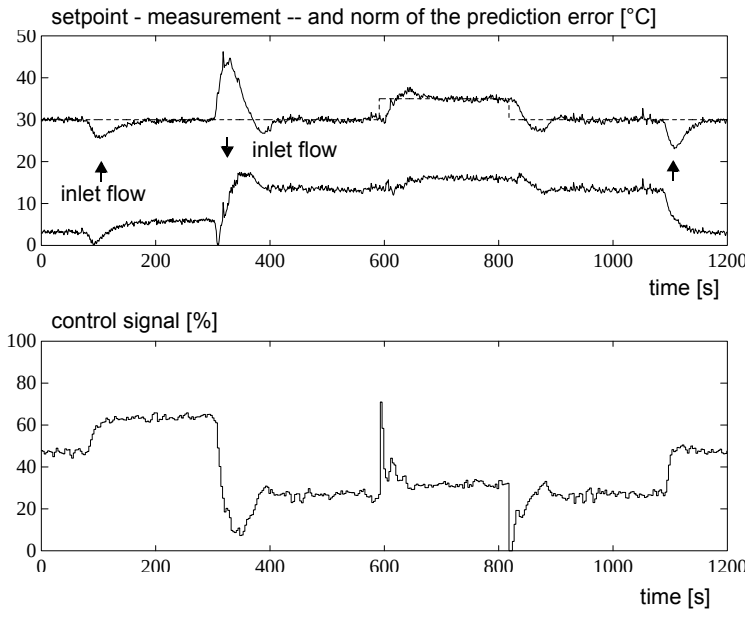


Fig. 6.10. The performance of the control system during the load disturbance test.

7 Control of Air Heating Process

The second case study considers the identification and control of a small laboratory heating process (Fig. 7.1). The air is fanned through a 30 cm plastic tube and heated with a thyristor controlled heating element. The voltage to the thyristor circuit is the actual control signal $u(t)$. The task of the control system is to keep the temperature $y(t)$, measured near the other end of the tube, at the desired setpoint ($20 \ldots 80$ °C). The fan speed can be set manually with another thyristor circuit. This alters the air flow in the tube and thus also the delay and the dynamical behaviour of the process. The identification experiments were made with the fan speed 100% and the 50% fan speed was used to test the robustness of the control system.

The process is only mildly nonlinear mainly due to the thyristor circuit (hysteresis and nonlinearity). Also the delay of $1 \ldots 2$ seconds and the heat transfer through the wall must be taken into account. The process is continuously used in control education and it is considered rather difficult to model, but relatively easy to control. The main interest in this case study is thus in identification, not in control.

The identification results, especially the NOE case, show that this simple process has complex behaviour and properties which require the advanced methods presented in Section 3.4. On the other hand, identification for predictive control was straightforward and the actual control of this process was a moderately easy task.

It should be emphasised that nonlinear modelling, identification, and control framework was applied *an sich*, contrary to what would have been done, if the task would have been just to develop a control system for the process.

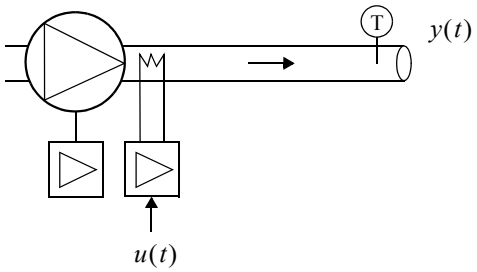


Fig. 7.1 The air heating process.

7.1 Identification

The goal of the identification was twofold:

- to identify a predictor for control purposes,
- to identify a deterministic model for simulation purposes.

The open loop process was first driven with a ramp type of control signal, changing 3 %/min., two times up and down the whole operation region, resulting in a data set of 2900 samples with $\Delta T = 1$ sec. (Fig. 7.2). As it can be seen either very slow dynamics or hysteresis is present. The "steady state" curve is clearly different when the fan speed was altered.

During the actual identification experiment, the process was driven with PRBS + offset type signal, resulting in 3300 samples (Fig. 7.3), which were divided into two parts: the first 2500 samples for an identification set and the rest 800 samples for a test set. The process indeed has a very slow mode and hysteresis is also present. The slow dynamics are due to the heat transfer through the tube wall. Although plastic has a small heat transfer coefficient it also has a significant heat capacity. The hysteresis is due to the thyristor circuit. Some nonlinearity is also assumed to be present, based on a nonlinear characteristic curve for the thyristor circuit measured several years ago during the installation of the process. The effect of this nonlinearity is however small.

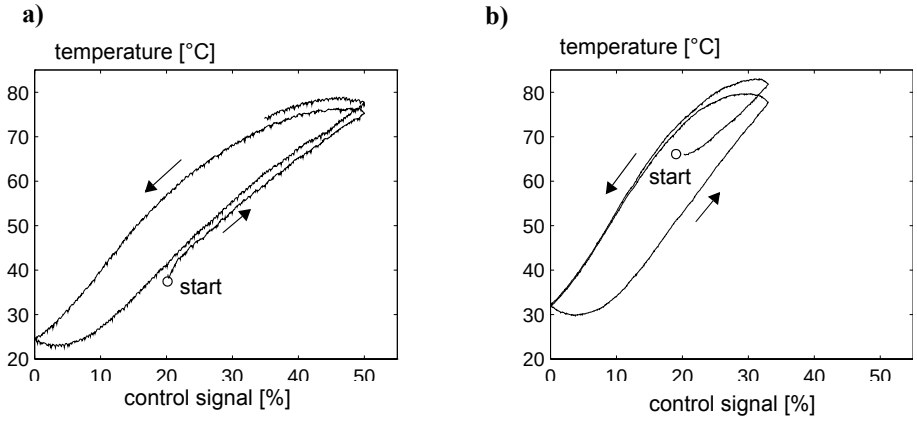


Fig. 7.2. The "steady state" characteristic curve of the open loop process, measured as a slow ramp response. $y(t)$ plotted as a function of $u(t)$, a) fan speed 100%, b) fan speed 50%.

A series of linear ARX models were first identified to determine the delay d and the model orders, resulting in delay $d = 2$. The determination of model orders n_a and n_b were not so straightforward, due to the slow dynamics.

The NARX predictors were identified next, but for the clarity of the presentation, the NOE case is considered first. Due to the amount of different model structures, the abbreviations like NOE(n_f,n_b,d) are used as a shorthand notation, i.e. to denote

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\hat{y}(t)\ldots\hat{y}(t-n_f+1), \\ &\quad u(t-d+1)\ldots u(t-d-n_b+2)]^T \end{aligned} \tag{7.1}$$

First a linear OE(2, 2, 2) i.e. with $n_f = 2$, $n_b = 2$, and $d = 2$ was identified. The model is too simple to represent the slow dynamics of the process and an OE(4, 2, 2) model was identified instead.

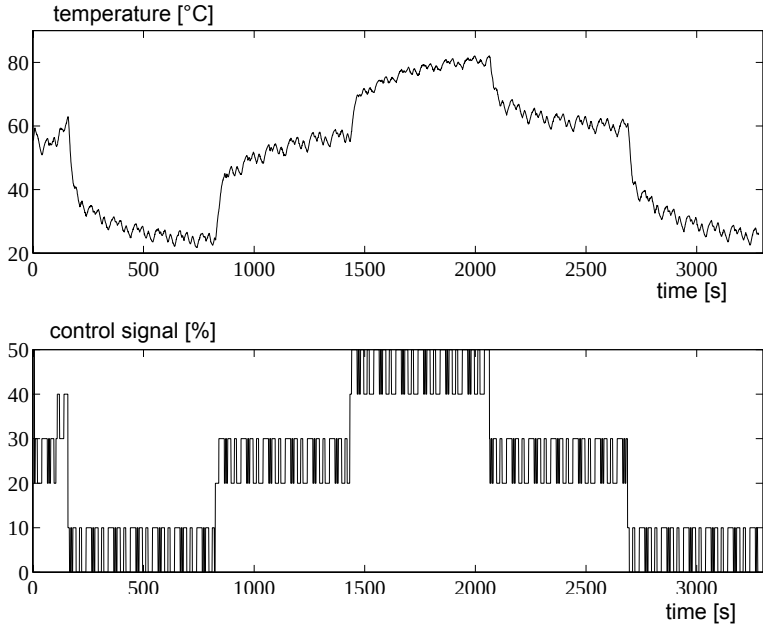


Fig. 7.3. The measurement and the control signal, when driving the process with PRBS+offset type signal (fan speed 100%).

Also the corresponding NOE(2, 2, 2) model, implemented with an MLP network

layers	1 hidden, 6 nodes
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	no

was identified with the LM method. Due to the slow dynamical behaviour, the instability problems were not surprising and the model was reidentified with the constraint (3.89)

$$V_N(\theta) = \frac{1}{2}\sum_{t=1}^{N}\left[\left\|(y(t)-\hat{y}(t|\theta))\right\|^2 + \gamma\left\|v(t,\theta)\right\|^2\right]$$

$$\text{with } v(t,\theta) = \begin{cases} (v-\eta) \text{ if } v = \sum_{i=1}^{nf}\left|A_i(t)\right| \geq \eta \quad \text{and } |\text{roots}(A(q))| \geq 1 \\ 0 \text{ , otherwise} \end{cases} \tag{7.2}$$

A pragmatical selection $\eta = 1{,}7$ resulted in largest spectral radius 1,003 and largest $v(t) = 1{,}701$. It is noticeable that the resulting constrained cost function value is *smaller* than the unconstrained one i.e. the mild constraint helped the identification. The resulting cost functions are compared in Table 7.1. The results seem to justify the use of a nonlinear model, the difference to linear models being a decade. The behaviour of the identified NOE(4, 2, 2) model with the identification set is presented in Fig. 7.4. As it can be seen the prediction error is small.

Table 7.1. Prediction errors (MSSE) for different models.

Model	Identification set	Test set
OE(2, 2, 2)	6.48	9.36
OE(4, 2, 2)	4.32	9.36
NOE(4, 2, 2)	0.50	1.33
NOE(4, 2, 2) (constrained)	0.45	1.25

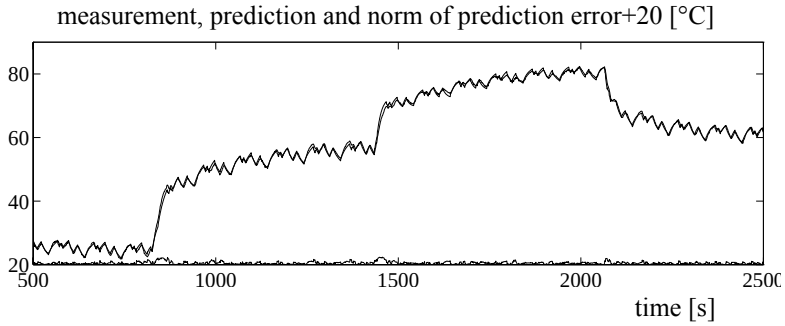


Fig. 7.4. The behaviour of the NOE(4, 2, 2) model with (a part of) the identification set.

The poles and the zero of the "linearized" model (3.56) are presented in Fig. 7.5, computed at every 10 th sample corresponding to Fig. 7.4. The model has a constant pair of complex poles, two poles near +1 and one zero near +1. The result of the identification indicates that IMA noise type of behaviour is present.

Also another conclusion is to be made. The poles and the zero do not vary much i.e. the *dynamics* of the nonlinear model seems to be constant, although the "linearized" model cannot be straightforwardly analysed this way, as discussed in Section 3.4.

The physical explanation for slow dynamics seems anyway to be relevant. However the use of the ramp responses as a validation set showed significant modelling error. It is somewhat questionable whether a deterministic part was obtained or a trend was identified. It is well known that a reliable identification of slow dynamics is difficult, even in the deterministic linear case.

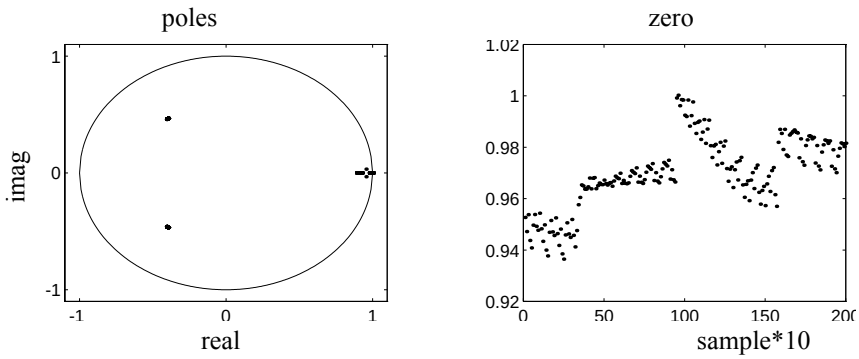


Fig. 7.5. The poles and the zero of the "linearized" model. Plotted for every 10th sample.

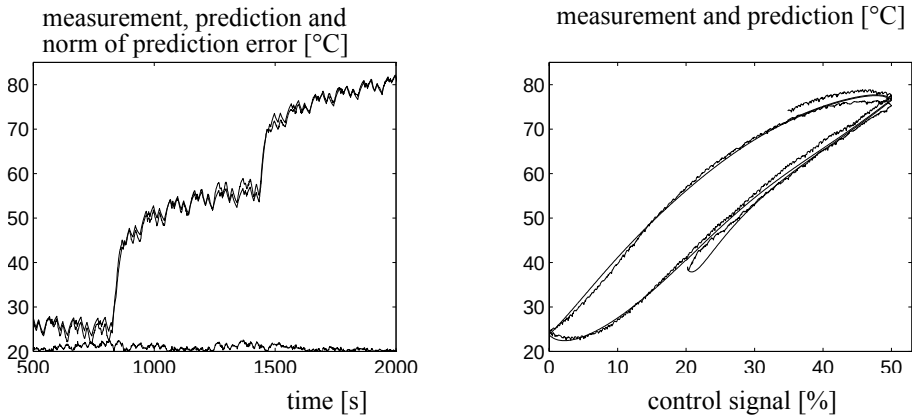


Fig. 7.6. The behaviour of the identified NOE(4, 2, 2) model, identified with all available data (fan speed 100%).

After the control experiments (Section 7.2) the NOE(4, 2, 2) model was refined using all available data for the identification i.e. combining identification, test and ramp set together. The idea was to try whether a deterministic nonlinear model could represent both slow and fast dynamics. The existing hysteresis was a problem because such identification requires much more experimental data than was available. The stability constraints were again a necessity, applied now with the BFGS approach. The resulting behaviour is presented in Fig. 7.6. The fit is quite good, especially for the ramp test, but the residuals with the original identification set, i.e. compared to those in Fig. 7.4, were smaller. Anyway, the resulting model is very useful for simulation purposes.

The identification results presented above do not have much to do with the models aimed for predictive control. For this ARX(2, 2, 2), ARX(4, 2, 2), and NARX(2, 2, 2) predictors were identified using the original 2500 samples as the identification set. The NARX predictor was implemented with an MLP network

layers	2 hidden, 6 nodes each
activation function	hidden: *tanh*, output: *linear*

The resulting models are compared in Table 7.2. The difference between the linear and the nonlinear predictors is quite small, indicating that adequate control performance with the predictive control can be obtained also with linear models.

Table 7.2. Prediction errors (MSSE) for different models.

Model	Identification set	Test set
ARX(2, 2, 2)	0.046	0.046
ARX(4, 2, 2)	0.040	0.040
NARX(2, 2, 2)	0.034	0.037

If the slow dynamics is considered as an external disturbance there are several possibilities to incorporate the IMA noise assumption. The NOE-IMA(2, 2, 2) was applied here i.e. the predictor

$$\begin{aligned} \tilde{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\tilde{y}(t)\ \tilde{y}(t-1)\ u(t-1)\ u(t-2)]^T \\ \hat{z}(t+1) &= y(t) - \tilde{y}(t) \\ \hat{y}(t+1) &= \tilde{y}(t+1) + \hat{z}(t+1) \end{aligned} \tag{7.3}$$

was identified, using the same network structure as NARX(2, 2, 2), This gave the smallest identification cost function of all experiments (Table 7.3). Also the linear OE-IMA performed better than ARX predictors.

The behaviour of the NOE-IMA(2, 2, 2) predictor is presented in Fig. 7.7. The predicted trend $\hat{z}(t)$ is clearly due to the slow dynamics plus some undetermined part due to the hysteresis. It is also obvious that $\hat{z}(t)$ correlates with the input, but the response is very slow. Note also the steady state behaviour of the resulting NOE part ($\tilde{y}(t)$), the gain is clearly linear.

Table 7.3. Prediction errors (MSSE) for different models.

Model	Identification set	Test set
OE-IMA(2, 2, 2)	0.037	0.038
NOE-IMA(2, 2, 2)	0.031	0.031
NOE-IMA(2, 2, 2) (constrained)	0.032	0.032

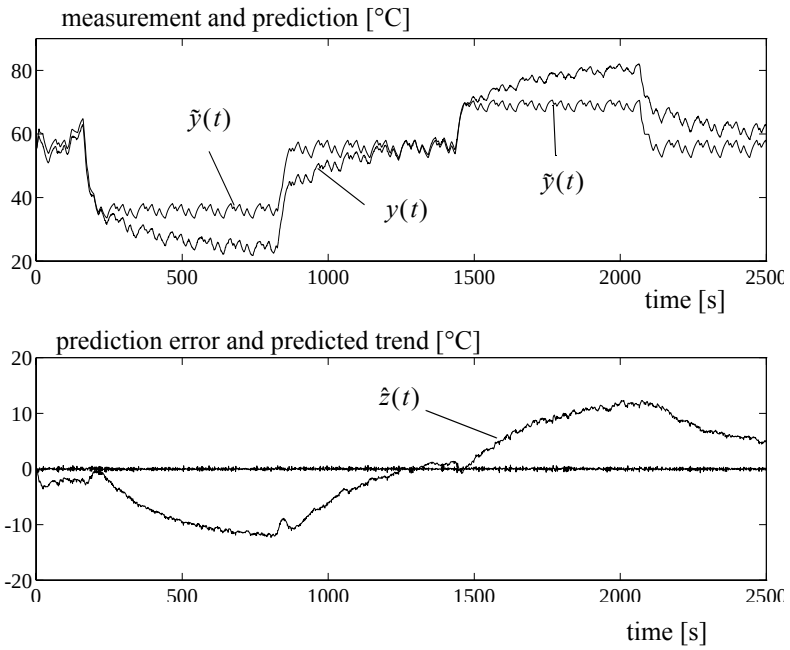


Fig. 7.7. The behaviour of the NOE-IMA(2, 2, 2) predictor with the identification set.

As it can be seen from Fig. 7.2 and Fig. 7.3, the maximum desired temperature 80 °C was achieved with $u(t) = 50$ % i.e the identification data did not include any samples where $u(t) > 50$ %. This might cause problems during the control if a nonlinear model is applied. Indeed this happened during the first control experiments. If the control signal even temporarily rose to 80 % or more, it stayed there. A local minimum of the LRPC cost function was achieved. The control signal should be limited below 50 %. To study the "input gradient" projection this was not applied, except as a test.

For example Fig. 7.8 presents the steady state characteristic curve of the NOE part of the identified NOE-IMA(2, 2, 2) predictor and the corresponding sum of the "input gradients" i.e.

$$\sum_{i=2}^{3} \partial\tilde{y}(t)/\,\partial u(t-i) \tag{7.4}$$

The identified area corresponds to $u(t) \in [0, 50]$ %. Outside that the model extrapolates. It extrapolates in an inconvenient way and the characteristic curve turns back. This can also be seen from the sum of the "input gradients", which crosses zero line approxi-

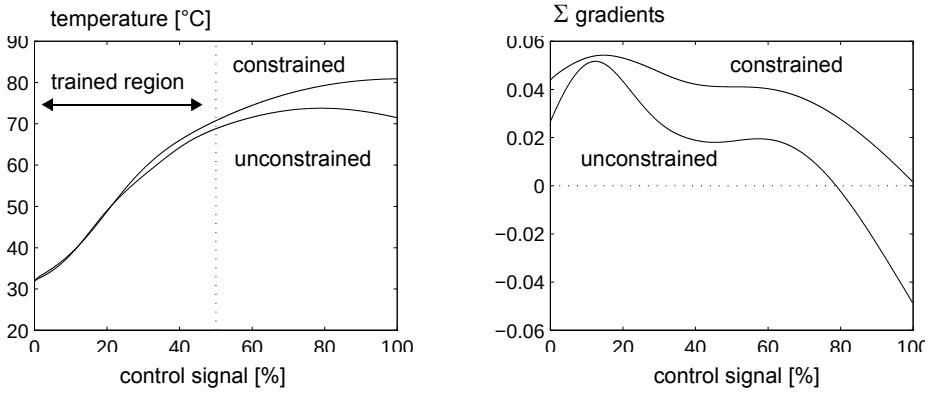


Fig. 7.8. a) the steady state characteristic curve of the NOE part of the identified NOE-IMA(2, 2, 2) predictor, b) the corresponding sum of the "input gradients" , see (7.4).

mately at the same time.

The NOE-IMA(2, 2, 2) predictor was refined using the penalty (3.97) i.e.

$$v(t, \theta) = \begin{cases} (\eta - v) & \text{if } v = \sum_{i=2}^{3} \partial \tilde{y}(t) / \partial u(t-i) < \eta \\ 0 & \text{, otherwise} \end{cases} \tag{7.5}$$

The limit $\eta = 0{,}02$ was selected interactively monitoring the simulated steady state curve and the corresponding sum of the gradients. The performance in the trained area decreased only slightly, from MSSE 0,0032 to 0,0031 (Table 7.3). The steady state characteristic curve is also presented in Fig. 7.8. The problems with non-monotonical characteristic curve have disappeared.

7.2 Control Experiments

In a way all what is actually needed to control the process is presented in Fig. 7.9. The process was controlled with a PID unit controller, SattControl ECA40. The controller has an autotuner option which was used to tune the PID settings. As can be seen the setpoint response is adequate and the overall behaviour is somewhat sluggish i.e. the control system behaves in a way typically desired in many real applications. Tighter tuning would probably decrease robustness.

The identified ARX(2, 2, 2) and NARX(2, 2, 2) predictors were applied in the LRPC approach. The cost function (4.8)

$$V(t,u) = \frac{1}{2}\sum_{i=N1}^{N2}\left[\left\| E(1)y^*(t+i) - E(q^{-1})\hat{y}(t+i)\right\|^2 + \left\| \Delta u(t+i-N1)\right\|_{\Lambda}^2\right] \quad (7.6)$$

was minimized with Brent's method. After some trials, design parameters were selected as $N1 = 2$, $N2 = 5$, $Nu = 1$, $E(q^{-1}) = 1 - 0{,}85q^{-1}$, $C(q^{-1}) = 1 - 0{,}7q^{-1}$, and $\Lambda = 10^{-4}$. Also the limit $\Delta u_{max} = 25\%$ was used in all experiments. The corresponding control system performance are presented in Fig. 7.10 and in Fig. 7.11.

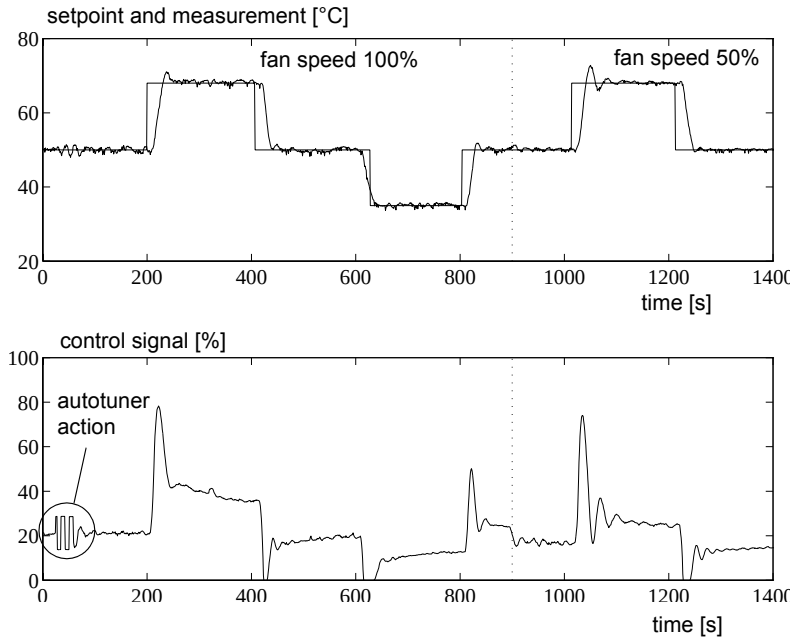


Fig. 7.9. The control system performance, a PID controller.

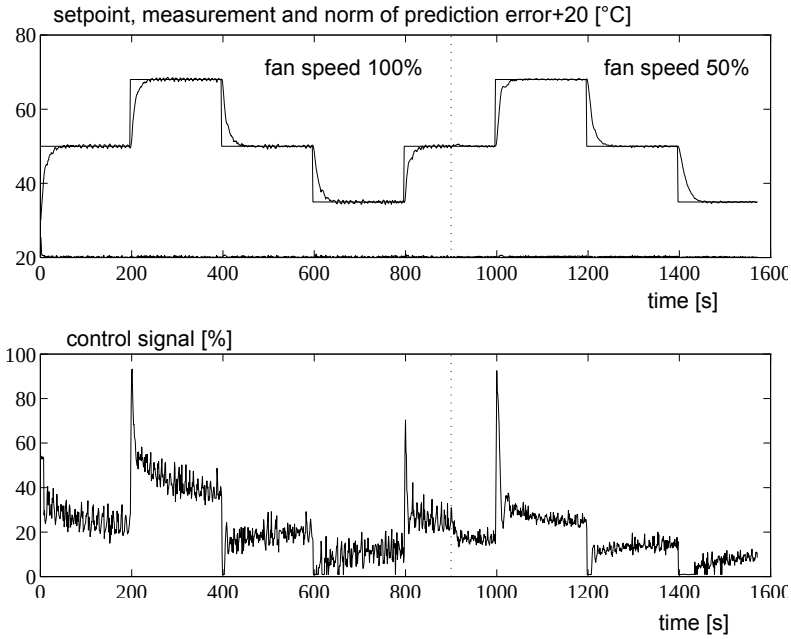


Fig. 7.10. The control performance of the LRPC approach with the ARX predictor.

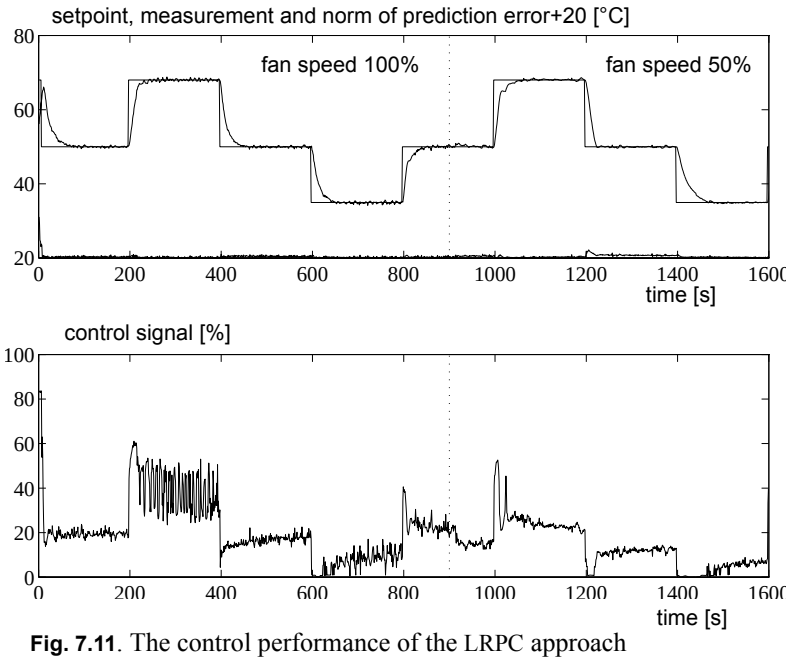


Fig. 7.11. The control performance of the LRPC approach with the NARX predictor.

It is hard to distinguish between the linear and the nonlinear control performance, except by looking at the control signals. This justifies the analysis made during the identification that the process is almost linear. The NARX predictor was not constrained, the "input gradients" were just checked to see whether they met the specifications. They did, but only barely. This can also be seen from the ringing type of behaviour in Fig. 7.10. Different prediction horizons $N2 = 2 \ldots 10$ were tried to remove it. The ringing slightly decreased when the prediction horizon increased, but on the whole almost identical control performance was achieved.

Comparison with the control performance of the PID controller demonstrates the efficiency of the LRPC approach. The noise rejection is clearly better and the robustness to the fan speed is also better. Altering the fan speed from $100 \rightarrow 50\,\%$ did not affect much the overall performance.

The higher order OE$(4, 2, 2)$ and NOE$(4, 2, 2)$ models were next applied within the LPRC-IMC approach, with the first order IMC filter F_2. They both failed, the control system was stable and the desired setpoint was achieved, but the overall performance was poor. This was expected based on the analysis of the identified models (Fig. 7.5.)

The low order models OE$(2, 2, 2)$ and NOE-IMA$(2, 2, 2)$ were tried instead, also within the LRPC-IMC approach. They performed better, especially the NOE-IMA, but failed to achieve the desired steady state setpoint. The first order IMC filter could not compensate the trend disturbance. The filter was redesigned assuming a ramp type disturbance and using a tighter reference model $E(q^{-1}) = 1 - 0{,}5q^{-1}$, resulting in

$$F_2(z) = (2{,}6 - 3{,}7z^{-1} - 1{,}2z^{-2}) \frac{0{,}1}{1 - 0{,}9z^{-1}} \tag{7.7}$$

The other design specifications were same as before. The performance of this control system is presented in Fig. 7.12, now with the NOE-IMA$(2, 2, 2)$ model.

If the assumption of the NOE-IMA type disturbance is correct, then a NOE part of the NOE-IMA$(2, 2, 2)$ model could be applied as a NARX predictor, violating the noise model assumption, but assuming that the representation of f in (7.3) is correct. To test this the NOE-IMA$(2, 2, 2)$ model was used as a NARX predictor within the LRPC approach. After some trials, the design parameters are selected as $N1 = 2$, $N2 = 5$, $Nu = 1$, $E(q^{-1}) = 1 - 0{,}85q^{-1}$, $C(q^{-1}) = 1 - 0{,}7q^{-1}$, and $\Lambda = 10^{-4}$ i.e same as in the original NARX case. The control system performance is presented in Fig. 7.13

The response in both experiments was faster than before, due to tighter $E(q^{-1})$. A significant difference is the better noise rejection, although it cannot be clearly seen from the figures. The control signal variation is smaller in Fig. 7.13.

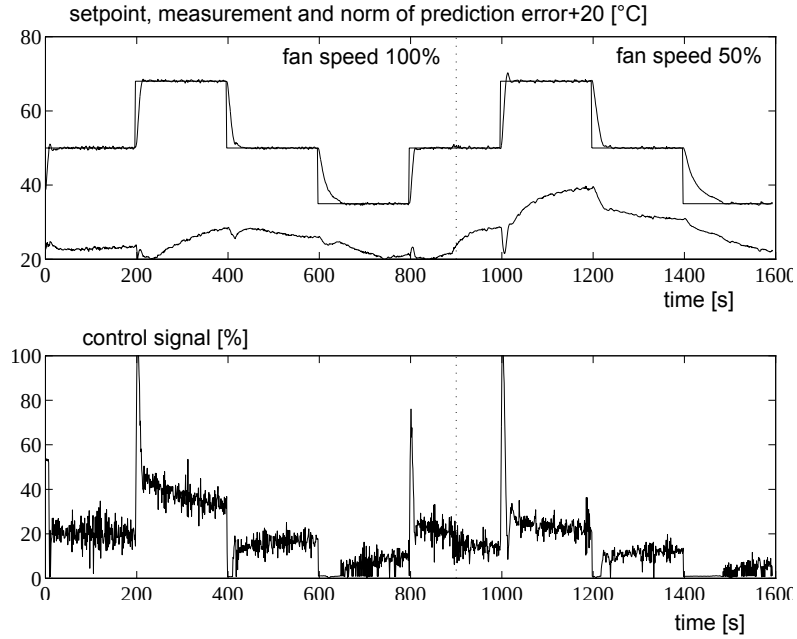


Fig. 7.12. The control performance of the LRPC-IMC approach with the NOE-IMA identified NOE predictor, second order IMC filter.

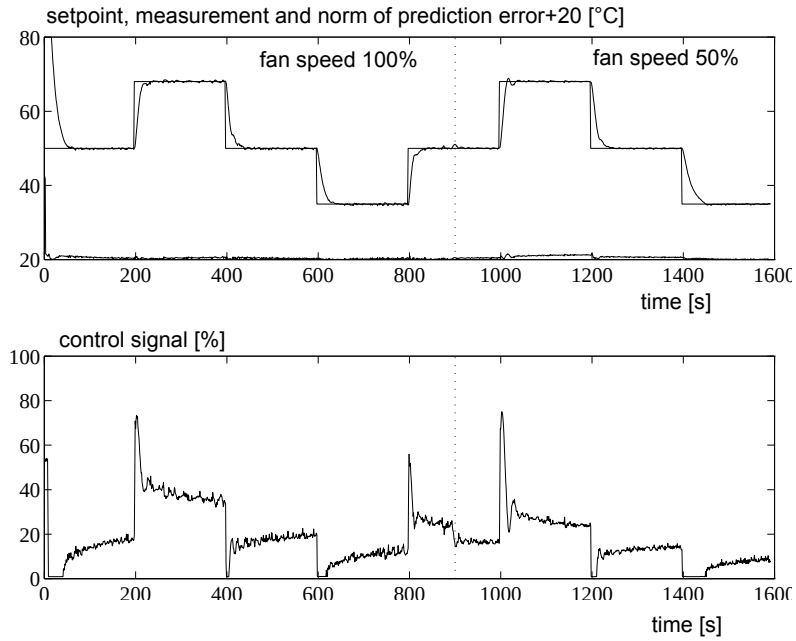


Fig. 7.13. The control performance of the LRPC approach with the NOE-IMA identified model as NARX predictor.

The results of the identification and control experiments are somewhat surprising. If the slow dynamics was considered as an external disturbance, the remaining behaviour were almost linear except for some hysteresis. This can be seen by comparing the identification cost functions of the ARX and NARX predictors, analysing the NOE and NOE-IMA models, and of course from the control experiments because almost identical control performance was obtained using either linear or nonlinear model. The robustness of the LRPC had also some effect on this.

The full nonlinear modelling, identification and control approach was applied, successfully and resulting in reliable models and good control performance. In this sense it is rather irrelevant that the process showed to be linear. Still, this is perhaps the most obscure way to design a linear controller.

The IMA noise assumption within the LRPC approach performs well with this type of disturbances, in fact it is just what the LRPC is supposed to do, see Section 4.2. The NOE-IMA model, applied as a NARX predictor gave the best overall performance if also the control signal variations are taken into account. The practical difficulties were in the identification of such nonlinear predictors. The proposed identification scheme seems to work in simulations and in practice. The convergence properties etc. are not thoroughly analysed and the approach is still at “experimental” stage.

A significant aspect is also that this small and “simple” process incorporates complex behaviour which made the identification difficult. The developed constrained identification scheme was clearly useful for obtaining reliable models.

8 Control of Pilot Headbox

This pilot process simulates some basic features of a real paper machine headbox. The pilot process is a real process not a computer simulation model. It has been extensively used as a test process both for modelling and control purposes in the Control Engineering Laboratory. The main task of a real headbox is to distribute stock across the wire through a slice into a 10 millimeter thick and several meters wide jet. Stock is a mixture of fibres and water, the amount of fibres being only 1% depending on the paper grade. Here pure water is used instead.

The pilot headbox process is presented in Fig. 8.1. The water is pumped from the lower storage tank to the upper tank (the headbox). The speed of the pump (WP) is controlled

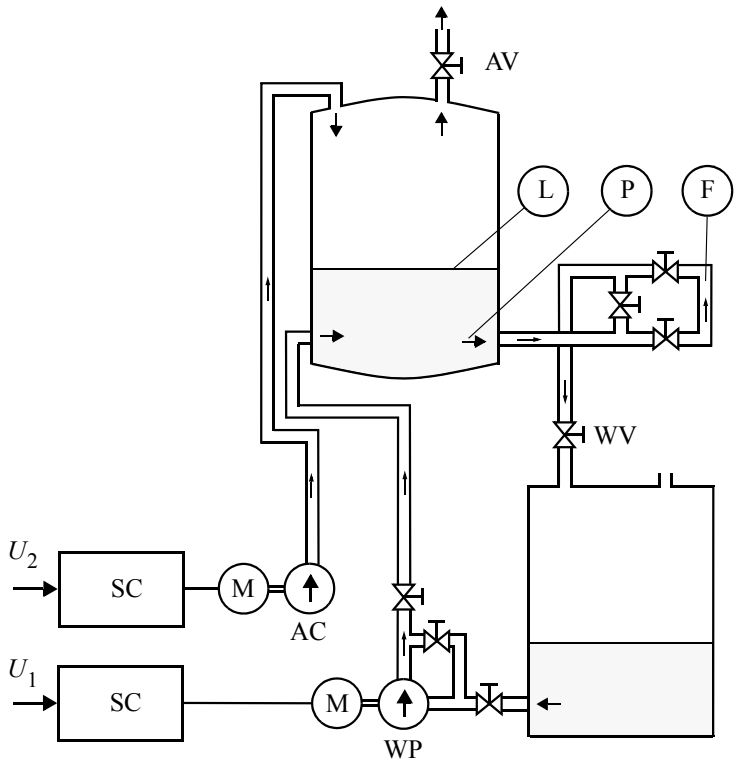


Fig. 8.1 The pilot headbox process.

with a thyristor circuit. The overpressure in the tank is maintained by blowing air to the headbox with a thyristor controlled compressor (AC). The slice is a combination of a pipe and a valve (WV) and the water "jets" through it into the open air pressure. The process characteristic can be set with the water valve (WV) and the air valve (AV). The main task of the pilot headbox is to simulate the dynamical behaviour of the pressure and the water level w.r.t. the controls (water pump and compressor) in the real headbox.

The water level (0…0,6 m) and the pressure at the water outlet point (0…1 bar) are the controlled variables (measurements L and P). The pressure is the sum of water hydrostatic pressure and air pressure. Both measurements are scaled between 0...1. The actual control variables are the voltages (1…5 V) to the speed controllers (SC). The control task is to keep the pressure and the level at desired setpoints.

The effect of the water pump to both measurements is much stronger than the effect of the compressor which makes the control difficult. As typical to all tank processes, a water level includes an integrating feature (not a pure integrator). Without a controller the measurements slowly drift away from the setpoints i.e. the system has a saddle type equilibrium point. This is also a serious difficulty, if one wants to identify a deterministic model of the process using open loop experiments. The system is not heavily nonlinear, but the process dynamics vary depending on the operation point and on the valve settings (WV and AV).

The compressor affects directly the air pressure and thus the overall measured pressure. The air pressure affects in turn the level, but there is no direct relationship between the level and the compressor speed (see Fig. 8.2), a fact which can also be derived from the basic physical principles.

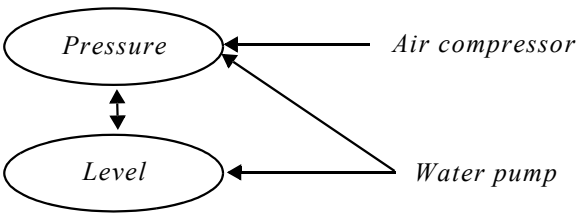


Fig. 8.2. Basic relationships between the measurements and the controls.

8.1 Identification

The goal is to control both the pressure and the level in a wide range of operation points. The main purpose of the identification is twofold:

- to identify a predictor for control purposes,
- to identify a deterministic model for simulation purposes (for controller design).

The former was relatively straightforward and easy, but the latter caused problems, mainly due to the integrating feature which made the identification experiments difficult. Only after applying a coarse controller (PI controller + relay with hysteresis), good quality data were obtained, resulting in two data sets (2500 samples and 4700 samples with $\Delta T = 1$ s.). The outputs of the PI controllers were transformed into a coarsely quantified offset plus a pulse width modulated signals between “high” and “low” with a period of 15 seconds. The identification set was formed by picking several at least 500 second intervals from both data sets, the rest remaining for a test set. The variables were ordered according to

y_1	pressure	u_1	water pump speed
y_2	water level	u_2	compressor speed

The model orders and delays were determined by fitting a series of linear ARX models and selecting the best. The same model orders are used in data vector φ of the nonlinear model. The predictive NARX model is identified separately for both measurements. The level predictor is identified according Fig. 8.2 i.e. air compressor speed is not used. The resulting two separate neural networks are combined to one sparse network i.e. the predictors

$$\begin{aligned}
\hat{y}_1(t+1) &= f_1(\varphi_1(t+1), \theta_1) \\
\hat{y}_2(t+1) &= f_2(\varphi_2(t+1), \theta_2) \\
\varphi_1(t+1) &= [y^T(t), y^T(t-1), u^T(t), u^T(t-1)]^T \\
\varphi_2(t+1) &= [y^T(t), y^T(t-1), u_1(t), u_1(t-1)]^T
\end{aligned} \tag{8.1}$$

are combined to

$$\hat{y}(t+1) = f(\varphi_1(t+1), \theta) \tag{8.2}$$

An MLP network with

layers	1 hidden, 6 nodes
activation function	hidden: *tanh*, output: *linear*
linear short-cut connections	no

is used for both predictors.

Because the NARX predictor is intended to be used with LRPC approach, also a multistep predictor was identified (10-multistep-ahead NARX) i.e. according the cost function

$$V_{LRPI} = \frac{1}{2}\sum_{t=1}^{N}\sum_{j=1}^{10} \left\| y(t) - \hat{y}(t|t-j) \right\|^2 \tag{8.3}$$

and with same network structure as the plain NARX.

Also a deterministic NOE model was identified using two separate predictors with the other measurement as an measured disturbance i.e.

$$\begin{aligned} \hat{y}_1(t+1) &= f_1(\varphi_1(t+1), \theta_1) \\ \hat{y}_2(t+1) &= f_2(\varphi_2(t+1), \theta_2) \\ \varphi_1(t+1) &= [\hat{y}_1(t),\, y_2(t),\, \hat{y}_1(t-1),\, y_2(t-1),\, u^T(t),\, u^T(t-1)]^T \\ \varphi_2(t+1) &= [y_1(t),\, \hat{y}_2(t),\, y_1(t-1),\, \hat{y}_2(t-1),\, u_1(t),\, u_1(t-1)]^T \end{aligned} \tag{8.4}$$

The neural network structure and size is same as in the NARX case. The resulting models were combined to one sparse network i.e to a predictor

$$\begin{aligned} \hat{y}(t+1) &= f(\varphi(t+1), \theta) \\ \varphi(t+1) &= [\hat{y}^T(t),\, \hat{y}^T(t-1),\, u^T(t),\, u^T(t-1)]^T \end{aligned} \tag{8.5}$$

which was reidentified as true MIMO-NOE style (only 30 iterations is needed).

The stability and gradient limits for all resulting predictors were checked, but no projection was needed. This is due to the good quality of the identification data. Especially the gradient projection can be seen as a way to incorporate apriori knowledge to compensate the lack of data.

The cost function values of the resulting models are presented in Table 8.1. The difference between linear OE and NOE model is remarkable, indicating that the true process is non-linear. However, when comparing the linear ARX and NARX models the difference is quite small indicating that a linear predictive control might give adequate performance.

As mentioned before, the NOE model is needed mainly for simulation purposes. A significant aspect is that the amount of data (different operation points) and the computational effort needed for a NOE model was a decade greater than for a NARX model. The resulting NOE model was tested by simulating the data set of 4700 points. Fig. 8.3 shows three 300 second samples of the simulation, corresponding to the best and the worst case situations. This model is capable to predict the whole data set without significant error. This is remarkable because of the integrator in the level causes also an accumulating error.

The identified NOE model indeed revealed some basic features of the process. This does not help much, as will be seen in control experiments, because even a slight change in valve settings causes a cumulative prediction error.

Anyway, the NOE model can be efficiently used for simulation purposes. An example clarifies the quality of the resulting NOE model. The simulation model and the pilot process were controlled with the same predictive controller and with the same tuning parameters. As it can be seen from Fig. 8.4, the simulation model incorporates the dynamical behaviour of the true process up to small details, only minor steady state errors in control signals can be seen.

Table 8.1. Prediction errors (MSSE) for different models.

Model	Identification set	Test set
NOE	5.1e-5	16e-5
linear OE	40e-5	263e-5
NARX	3.0e-5	3.9e-5
linear ARX	3.7e-5	4.6e-5

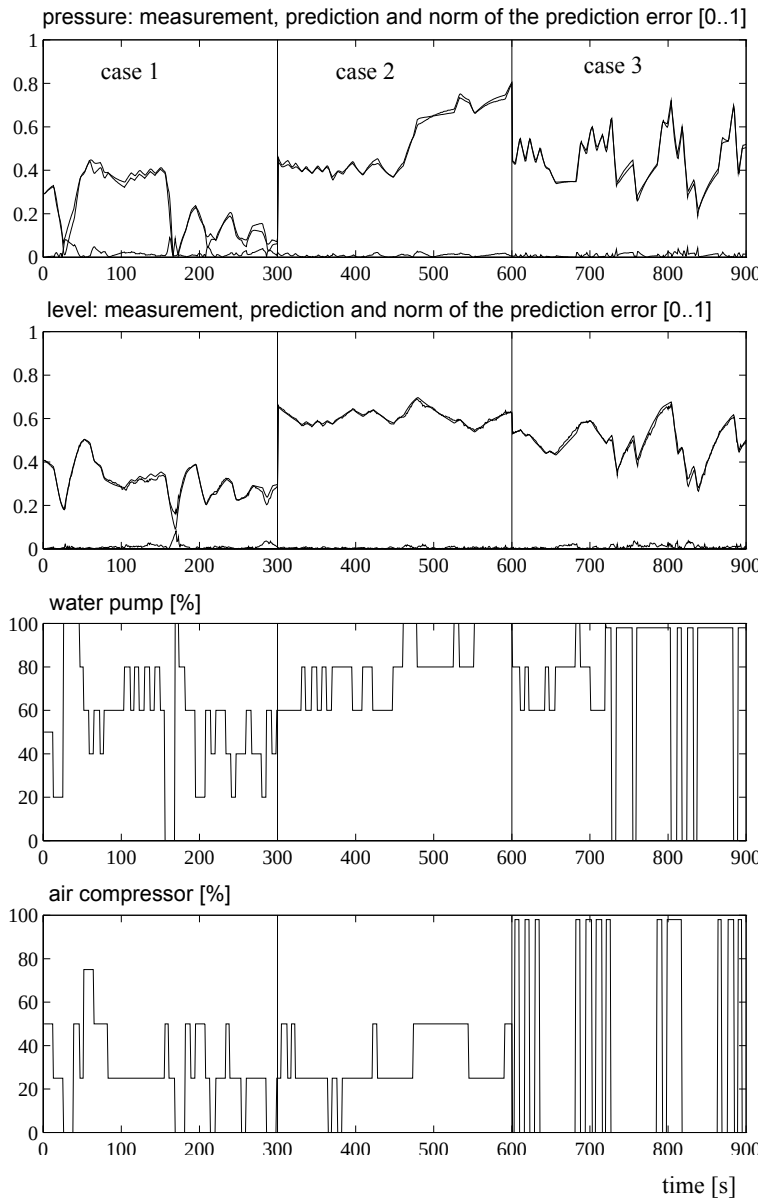


Fig. 8.3. The behaviour of the identified NOE model with the identification / test set of 4700 seconds. Case 1: 0...300 s, case 2: 1100...1400 s and case 3: 4000...4300 s.

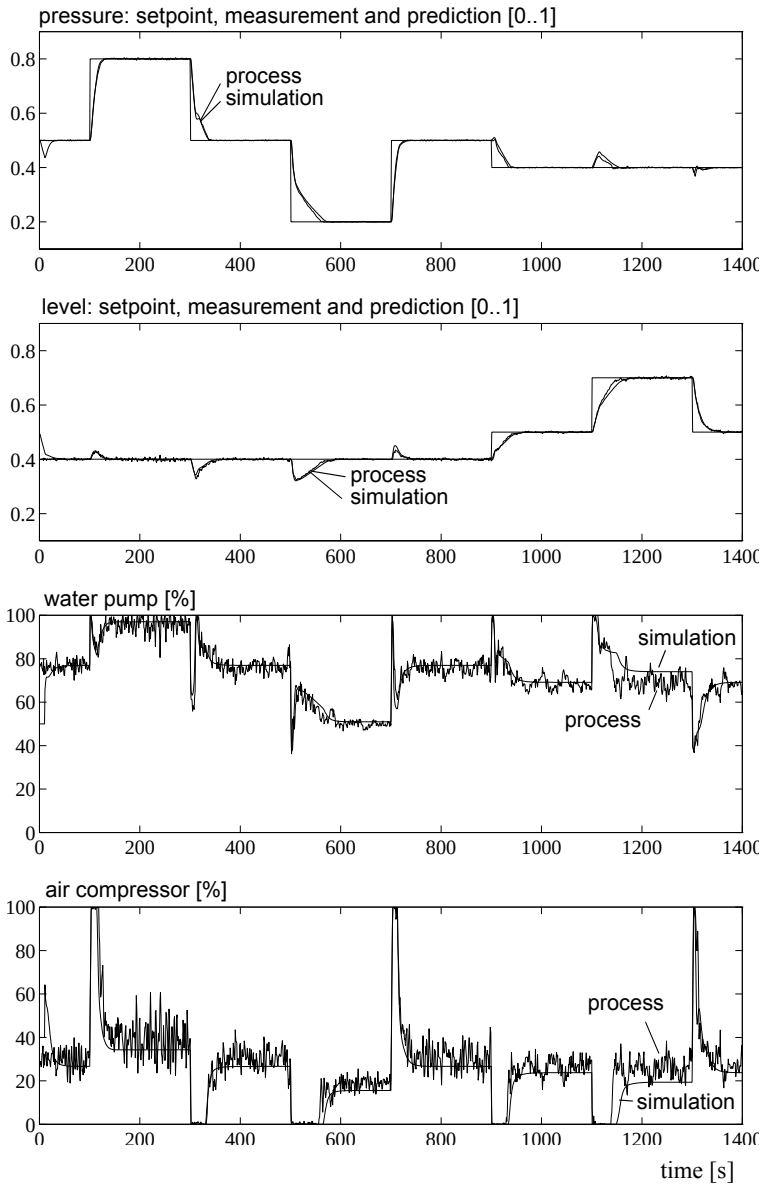


Fig. 8.4. The behaviour of the identified NOE model and the true process, when both are controlled with the similar predictive controller with same tuning parameters.

8.2 Control Experiments

To get a reference for nonlinear and MIMO control, the system was first controlled with two SISO PI controllers; both Foxboro Exact (1989) with an autotuner option. The pressure was controlled with the water pump and the level with the air compressor. Because the water pump dominates the dynamical behaviour, the pressure control is better than the level control and the water pump causes disturbances to the level. If the control pairing would have been reversed, then the level control would have been better but the pressure control worse.

The autotuner option was used to tune the controllers. The desired response was selected to be fast, too tight in fact, as seen from Fig. 8.5. Note the temporal instability at the pressure measurement near time 800 seconds. This process is used extensively in control education and similar performance shown in Fig. 8.5 is seldom achieved using a linear model as a basis for a PI controller design nor by tuning the controllers heuristically.

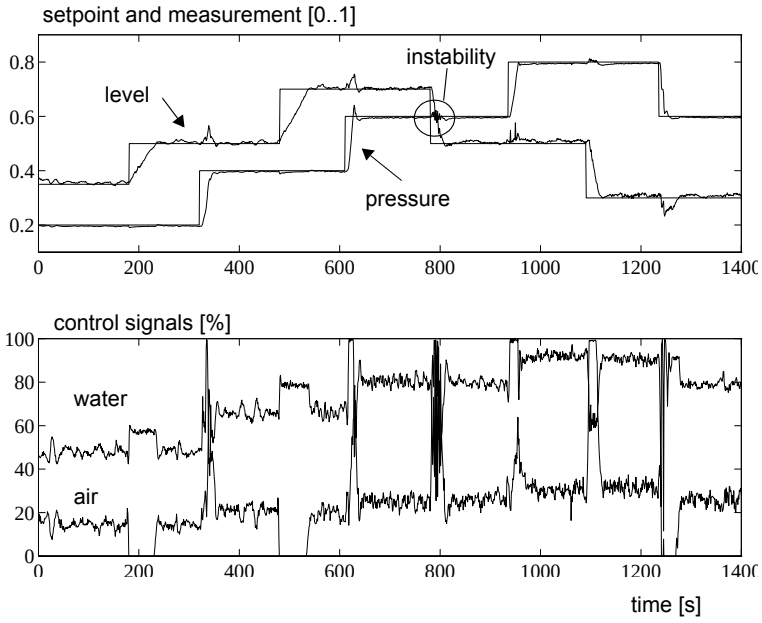


Fig. 8.5. The control system performance, PI controllers.

The identified 10-multistep-ahead NARX predictor was applied with the LRPC approach and the cost function (4.8)

$$V(t,u) = \frac{1}{2}\sum_{i=N1}^{N2}\left[\| E(1)y^*(t+i) - E(q^{-1})\hat{y}(t+i)\|_W^2 + \| \Delta u(t+i-N1)\|_\Lambda^2\right] \quad (8.6)$$

was minimized using CFSQP/FSQP. After some trials, the design parameters were selected as $N1 = 1$, $N2 = 5$, $Nu = 1$, $W = I$, $\Lambda = 10^{-3}\cdot I$ and

$$E(q^{-1}) = I - \text{diag}\{0.7, 0.9\}q^{-1}$$
$$C(q^{-1}) = I - \text{diag}\{0.7, 0.7\}q^{-1}$$

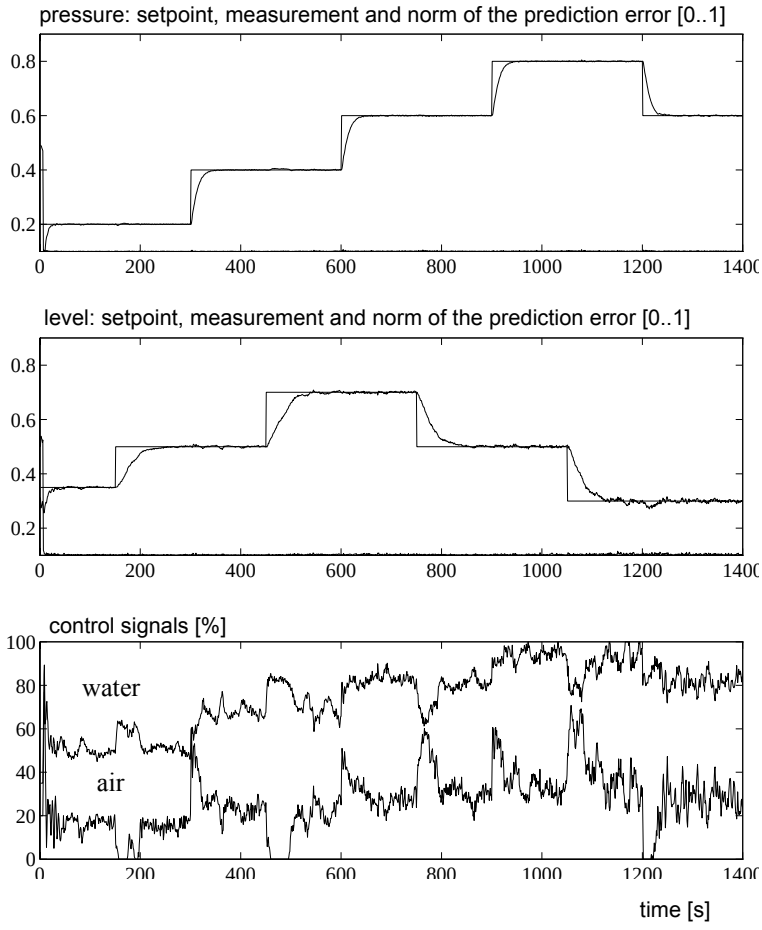


Fig. 8.6. The control performance of the LRPC approach with the NARX predictor.

where the C polynomial is used to filter the prediction error according (4.15). Also the limiting $\Delta u_{max} = I \cdot 25\,\%$ was used in all experiments.

This control system shows excellent performance with only minor interactions which are due to the high and low limits of both control signals. The goal of the control design was to remove also these interactions. The LRPC could quite well compensate them if the control signals were inside their limits. To achieve this the setpoint was filtered with

$$F_1(z) = \text{diag}\left\{ \frac{0{,}1z}{z-0{,}9} \;\; \frac{0{,}05z}{z-0{,}95} \right\}$$

to remove the rest of the interactions. The resulting behaviour of the control system is presented in Fig. 8.6. Hardly no interactions can be seen and the noise rejection is also good.

Using the linear ARX model instead of NARX yields almost comparable control behaviour. This can be reasoned comparing the cost function values of the identified ARX and NARX predictors (Table 8.1). However, the difference between the cost function values of the OE and NOE predictors is significant. One explanation is that the process is nonlinear only at some operation regions like at higher pressure. Another reason for good performance with linear models is obviouly the LRPC approach itself.

Although the NOE model is not initially identified for control purposes, it was anyway tested, first with the same settings for E, W, Λ and F_1 as with the NARX case and with the IMC filter

$$F_2(z) = \text{diag}\left\{ \frac{0{,}3z}{z-0{,}7} \;\; \frac{0{,}3z}{z-0{,}7} \right\}$$

Due to the integrator and especially due to the accumulating model mismatch, the steady state setpoint was not achieved. The IMC filter was redesigned by assuming a ramp type load disturbance and assuming the system to be controlled to be

$$z^{-1}\tilde{E}(z) = \text{diag}\left\{ \frac{0{,}3z^{-1}}{1-0{,}7z^{-1}} \;\; \frac{0{,}1z^{-1}}{1-0{,}9z^{-1}} \right\}$$

which resulted in the IMC filter

$$F_2(z) = \text{diag}\left\{ \begin{array}{l} (5{,}33 - 8{,}06z^{-1} + 3{,}03z^{-2})\dfrac{0{,}3}{1-0{,}7z^{-1}} \\ (3{,}8 - 6{,}22z^{-1} + 2{,}49z^{-2})\dfrac{0{,}07}{1-0{,}93z^{-1}} \end{array} \right\}$$

The performance of this control system is presented in Fig. 8.7. The performance is similar to that of the NARX, but at a price of heavier controller output variations due to the higher order IMC filter. The valve settings were slightly different from the ones of the identification experiments and the prediction error of the level accumulates (40% of the whole operation range after 1400 seconds). On the other hand, the control system shows remarkable robustness for the model mismatch. The small ticks are communication failures not real process disturbances.

It is clear that the IMC approach cannot be straightforwardly applied to this type of process. Sooner or later the model goes out of the trained region. This is a serious limitation, if one wishes to develop a dual network controller for the process. A separate NARX style state estimator must be identified or developed otherwise.

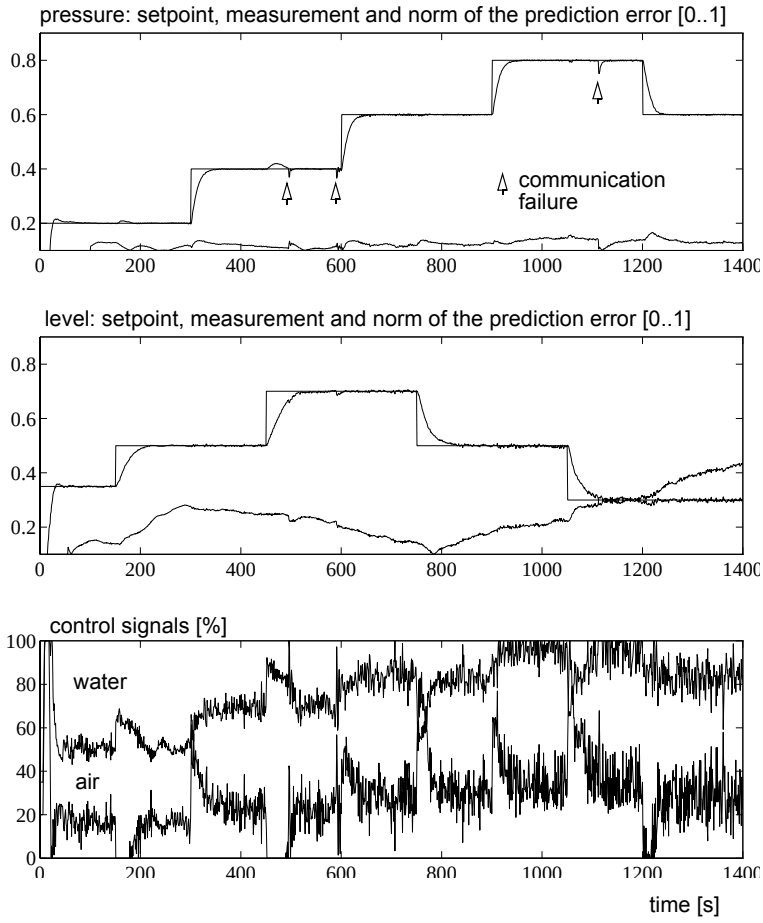
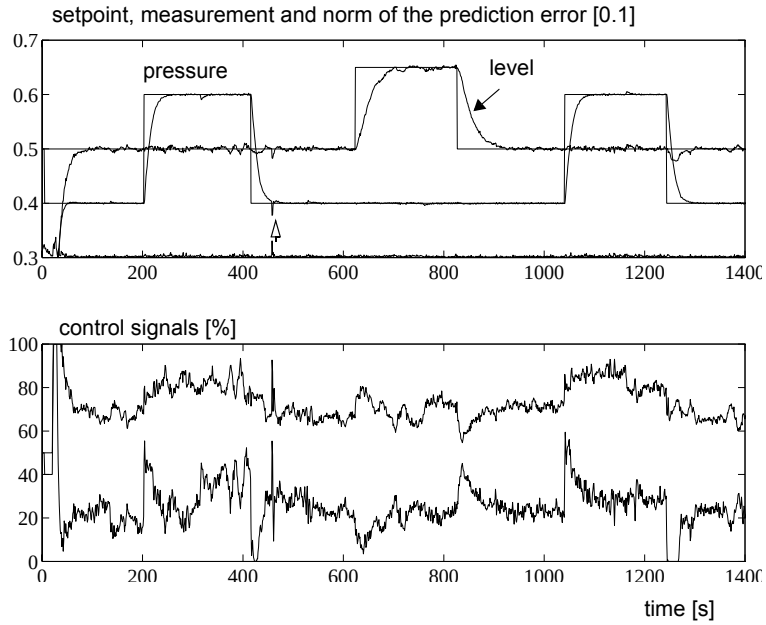


Fig. 8.7. The control system performance of the LRPC based IMC i.e with a NOE model. The small ticks are communication failures not real process disturbances.

The NARX-LRPC approach (Fig. 8.6) was tested also for model mismatch by changing the valve settings from their nominal values. The behaviour of the control system is presented in Fig. 8.8. The process is very sensitive to the air valve position and a 10° change must be considered as "large", the water valve position is not so critical. This can also be seen from Fig. 8.8. Changing the air valve $90° \rightarrow 80°$ and dropping the pressure setpoint brings the level control near the instability border at 400 seconds.

As a conclusion for this case study, a couple of remarks are to be made. Due to the integrator and the accumulating prediction error, the NARX predictor is superior to NOE predictor when considering predictive control. The NOE model can still be efficiently applied for simulation and control design purposes. The LRPC approach shows excellent performance and good robustness both with linear ARX and NARX predictors.



Air valve, nominal 90°

90°	$t < 100$	$t > 555$
80°	$100 < t < 300$	
100°	$300 < t < 600$	

Water valve, nominal 355°

355°	$t < 600$	$900 < t < 1200$
330°	$600 < t < 750$	
380°	$750 < t < 900$	$t > 1200$

Fig. 8.8. The control system performance, when changing the process from its nominal state. The NARX based LRPC approach.

9 Conclusion

A practical approach to model based neural network control is presented in this study. The efficiency of the approach has been verified with simulation studies and real time experiments, yielding good control characteristics and robust control. The general applicability of the proposed approach seems clear.

The goal of this study was to develop efficient identification and control design methods for neural network based nonlinear control, and to implement them in a real world environment. The study thus also fills the gap between theory and practice. Practice is anyway the final measure to any control method and this research area is of great importance.

The multilayer perceptron neural network is selected to be the basic block for representation of nonlinear process models and controllers. The main attractive feature of neural network models is that they offer a parametric function which can accurately represent virtually any smooth nonlinear function with similar basic structure. The results verified this well-known fact.

The advanced identification and control design methods are presented and applied. The PEM approach combined with efficient optimization methods result in reliable models and controllers, clearly better than obtained with conventional methods like error backpropagation. A contribution is the incorporation of the constraints within the model identification and within the controller design to maintain the stability or to include apriori process knowledge. The latter also somewhat compensates the amount of experimental data needed for the identification.

The identified models are used for model predictive control. Both direct long range predictive control approach (LRPC) and dual network control approach are introduced. The problems related to the model inverse based IMC design are partially avoided by employing a nonlinear optimal controller within the IMC structure. The nonlinear control law is approximated by a perceptron network and the cost function, associated to the optimal controller design, is minimized numerically. Practical solutions for the model mismatch for both approaches are applied. General guidelines and practical methods for maintaining the stability are also introduced and applied.

The neuro-control workstation based on HP9000 platform showed to be an efficient and flexible tool for identification of nonlinear process models, for development and simulation of nonlinear control systems, and for real-time experimental studies.

This model based control approach has successfully been applied to the control of laboratory scale water and air heating processes and to the multivariate control of the pilot headbox of a paper machine using the neuro-control workstation. These processes are not heavily nonlinear, but they incorporate many of the features which commonly make control difficult: delays, noise, deterministic disturbances, trends, time varying features etc. Solving the problems associated to these in conjunction of neural networks is the main contribution of the thesis, both from the theoretical and practical point of view. Experimental results indicate that both the direct LRPC and the dual network IMC structure provide robust performance and are clearly good alternatives for controlling nonlinear plants.

The direct LRPC approach with the NARX predictor seems to be extremely efficient and easily applicable control method, if only the on-line computational load is acceptable. In the linear case both ARX and OE models are commonly used for controller design according the *certainty equivalence principle*. The difference between NARX and NOE models is conceptually much bigger than in the linear case. Only NOE model which represents the deterministic part of the system, should be used in the design of IMC like deterministic controller.

The identification of a NOE model is often a formidable task in practice, as can be seen from the case studies. The overall quality of the experimental data must be better. On the other hand, the resulting NOE model is directly applicable as a simulation model for control design purposes, contrary to the NARX predictor.

As shown in simulation Example 5.3, the robustness of the IMC approach is good, but the NARX-LRPC approach outperforms it. Paying more attention to the state estimator design comparable results should in theory be achieved. This feature is however inbuilt to the NARX-LRPC approach.

The simulation studies and experimental results clearly point out that it is quite the same if the IMC approach is implemented within the LRPC or within the dual network approach, the control system behaviour is similar. The capability of handling on-line constraints and the possibility of on-line tuning favors the LRPC approach and the dual network approach is worthwhile only when a minor computational on-line load is needed, like in high speed applications.

The experiments with the adaptive control worked as expected, all operation regions should be visited once before the adequate control performance is achieved. The adaptive experiments should be considered mainly as demonstrations of the problematics encountered with adaptive nonlinear control. A lot of research must be done before real industrial adaptive applications will be seen.

There seems to be two adaptive approaches which might show useful in practice: inferential control and nonlinear (localized) autotuner. The inferential control slowly adapts the parameters of an additional nonlinear controller to the time varying process. The linear autotuner - tuning by request - is commonly favored in industry instead of linear adaptive control. The nonlinear version would tune a nonlinear controller or identify a model locally near the current operation point. This feature is difficult to implement with any global nonlinear function approximator, a localized model must be used instead.

When considering real industrial applications, the proposed off-line identifiable LRPC approach seems directly applicable, excellent performance but sometimes at a price of significant model identification task. On the other hand, a minor identification task would make the development of practical applications easier. The usage of coarser model would probably reduce the control system performance only slightly, the LRPC approach is quite robust as seen from the presented experimental results.

A coarser model, suitable for representation of nonlinear rather high dimensional functions, as normally required in time series models, obtained with minor identification experiments and applying the often available steady state information, would be an ideal practical solution. Combined with the proposed methods for maintaining stability and extrapolation would result in reliable models and controllers. The development of coarser and simpler model structures still maintaining the attractive features of the MLP neural network is clearly an important issue of the future research.

References

[1] Alamir, M. and G. Bornard (1994): New sufficient conditions for global stability of receding horizon control for discrete-time nonlinear systems. *Advances in Model-Based Predictive Control* [20], Oxford University Press, pp. 173-181.

[2] Albus, J. S. (1975): A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC). *Trans. of the ASME. Journal of Dynamic Systems, Measurement and Control.* Sept. 1975, pp. 220-227.

[3] Anderson, J. A. and E. Rosenfeld (Eds.) (1988): *Neurocomputing: Foundations of Research*. Cambridge. MIT Press.

[4] Anderson, C. W (1989): Learning to Control an Inverted Pendulum Using Neural Networks. *IEEE Control Systems Magazine.* April 1989, pp.31-37.

[5] Asakawa K. and H. Takagi (1994): Neural Networks in Japan. *Communications of the ACM,* Vol. 37, No. 3, pp. 106-112.

[6] Bar-Kana, I., A. Guez (1989): Neuromorfic Adaptive Control. *28th IEEE Conference on Decision and Control*, Tampa, Florida, Dec. 1989, pp. 1739-1743.

[7] Barto, A. G., R. S. Sutton and C. W. Anderson (1983): Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man and Cybernetics (SMC),* Vol. 23, pp. 834-846.

[8] Bavarian, B (1988).: Introduction to Neural Networks for Intelligent Control. *IEEE Control Systems Magazine*, April 1988, pp. 3-7.

[9] Biermann, G. J. (1977): *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York.

[10] Billings, S. A. and Voon, W. S. F. (1986): Correlation based model validity tests for non-linear models. *Int. J. Control,* Vol. 44, No. 1, pp. 235-244.

[11] Billings, S. A. and Zhu, Q. M. (1984): Nonlinear model validation using correlation test. *Int. J. Control,* Vol. 60, No. 6, pp. 1107-1120.

[12] Bitmead, R. R., M. Gevers and V. Wertz (1990): *Adaptive Optimal Control, The Thinking Man's GPC*. Prentice-Hall, 1990.

[13] Bhat, N. and T. McAvoy (1990): Use of Neural Nets for Dynamic Modelling and Control of Chemical Process Systems. *Computers Chem. Engng*, Vol. 14 No. 4/5, pp. 573-583.

[14] de Boor, C. (1978): *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[15] te Braade, H., R. Babuska, E. van Can (1994): Fuzzy and neural models in predictive control. *Journal A,* Vol. 35, No. 3, pp. 44-51.

[16] Chen, S., S. A. Billings, P. M. Grant (1990): Non-linear system identification using neural networks. *Int. J. Control,* vol 51, no. 6, pp. 1191-1214.

[17] Chen, S., C. Cowan *et al.* (1990): Parallel Recursive Prediction Error Algorithm for Training Layered Neural Networks. *Int. J. Control,* vol 51, no. 6, pp. 1215-1228.

[18] Chen, S., C. Cowan, P. Grant (1991): Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. *IEEE Trans. on Neural Networks,* Vol. 2, No. 2, March 1991, pp. 302-309.

[19] Chen, Q. and W. A. Weigand (1994): Dynamic Optimization of Nonlinear Processes by Combining Neural Net Model with UDMC. *AIChE Journal,* Sep. 1994, Vol. 40, No. 9, pp. 1488-1497.

[20] Clarke, D. W. (Eds.)(1994): *Advances in Model-Based Control.* Oxford University Press.

[21] Clarke, D. W. and C. Mohtadi (1989): Properties of Generalized Predictive Control. *Automatica.* Vol. 25, No. 6, pp. 859-875.

[22] Cooper D. J., L. Megan and R. F. Hinde Jr. (1992): Comparing Two Neural Networks for Pattern Based Adaptive Process Control. *AIChE Journal*, January 1992, Vol. 38, No. 1, pp. 41-55.

[23] Demircioglu, H. and D. W. Clarke (1992): CGPC with guaranteed stability properties. *IEE Proceedings-D,* Vol. 139, No. 4, pp. 371-380.

[24] Economou. C. G. (1986): *An Operator Theory Approach to Nonlinear Controller Design*, Ph.D. thesis, California Institute of Technology.

[25] Economou, C. G., M. Morari, B. O. Palsson (1986): Internal Model Control 5. Extension to Nonlinear Systems. *Ind. Eng. Chem. Process Des. Dev,* 25, pp. 403-411.

[26] Elsley, R. K. (1988): A Learning Architecture for Control Based on Back-propagation Neural Networks. *Proc. of IEEE International Conference on Neural Networks (ICNN),* San Diego, California, pp. Vol. II 587-594.

[27] Ersü, E., H. Tolle (1988): Learning Control Structures with Neuron-Like Associative Memory Systems. In W. v. Seelen et al. (Eds.)*: Organization of Neural Networks. Structures and Models*, Weinheim (FRG), 1988.

[28] Fletcher, R. (1990): *Practical Methods of Optimization.* John Wiley & Sons, New York.

[29] Fliess, M. and Hazewinkel, M. (Eds.) (1986): *Algebraic and Geometric Methods in Nonlinear Control Theory,* D. Reidel Publishing Company.

[30] Fox, D., V. Heinze *et al.* (1991): Learning by Error-Driven Decomposition. *ICANN-91, International Conference on Artificial Neural Networks*. Espoo, Finland, June 24-28, 1991. pp 207-212.

[31] Foxboro Instruction Book (1989): 761 Series single station micro plus controller, MI-018-848, the Foxboro Company.

[32] Franklin, J. A.(1989): Historical Perspective and State of the Art in Connectionistic Learning Control. *Proc. of the 28th IEEE Conference on Decision and Control.* Tampa, Florida, Dec. 1989, pp. 1730-1736.

[33] Fu, K-S. (1970): Learning Control Systems - Review and Outlook. *IEEE Trans. on Automatic Control,* April 1970, pp. 210-221.

[34] Garcia, C. E., D. M. Prett, M. Morari (1989): Model Predictive Control: Theory and Practice - a Survey. *Automatica,* Vol. 25, No. 3, pp. 335-348.

[35] Goodwin, C. G., K. S. Sin (1984): *Adaptive Filtering, Prediction and Control*. Prentice-Hall.

[36] Grabec, T. (1991): Modelling of Chaos by a Self-Organizing Neural Network. *ICANN-91, International Conference on Artificial Neural Networks*. Espoo, Finland, June 24-28, pp. 151-156.

[37] Gupta, M. M. and D. H. Rao (1993): Dynamic Neural Units with Applications to the Control of Unknown Nonlinear Systems. *Journal of Intelligent and Fuzzy Systems,* Vol. 1(1), pp. 73-92.

[38] Haber, R. and Unbehauen, H. (1990): Structure Identification of Nonlinear Dynamic Systems - A Survey on Input/Output Approaches. *Automatica,* Vol. 26, No. 4, pp. 651-677.

[39] Henson, M. A. and D. E. Seborg (1990): A Critique of Differential Geometric Control Strategies for Process Control. *Preprints of 11th IFAC World Congress,* Tallin, Estonia, Aug. 13-17, 1990, Vol. 8, pp. 1-8.

[40] Hernandez, E., Y. Arkun (1992): Study of the control-relevant properties of back-propagation neural network models of nonlinear dynamical systems. *Computers Chem. Engng*, Vol. 16 No. 4, pp. 227-240.

[41] Hernandez, E., Y. Arkun (1993): Control of Nonlinear Systems Using Polynomial ARMA Models. *AIChE Journal,* March 1993, Vol. 39, No. 3, pp. 446-460.

[42] Hertz, J., A. Krogh, R. G. Palmer (1991): *Introduction to the theory of the neural computation*. Addison-Wesley.

[43] Holst, J. (1977): *Adaptive Prediction and Recursive Estimation*. Ph.D. Thesis, Lund Institute of Technology, Dept. of Automatic Control, Lund.

[44] Holzman, J. M. (1970): Nonlinear system theory - functional analysis approach. Prentice-Hall, 213 p.

[45] Hopfield, J. J., D. W. Tank (1985): "Neural" Computation of Decisions in Optimization Problems. *Biol. Cybern*, 52, pp. 141-152.

[46] Hornik. K., M. Stinchcombe, H. White (1989): Multilayered Feedforward Networks are Universal Approximators. *Neural Networks,* Vol. 2, pp. 359-366.

[47] Hunt, K. J., D. Sbarbaro (1991): Neural networks for nonlinear internal model control. *IEE Proceedings-D,* Vol. 138, No. 5, pp 431-438, Sep. 1991.

[48] Hunt, K. J., D. Sbarbaro *et al.* (1992): Neural Networks for Control Systems - A Survey. *Automatica,* Vol. 28, No. 6 pp. 1083-1112.

[49] Hyötyniemi, H. (1994): *Self-Organizing Artificial Neural Networks in Dynamic Systems Modelling and Control.* Dissertation, Report 97, Helsinki University of Technology, Control Engineering Laboratory, Finland, 136 p.

[50] Isidori, A. (1989): *Nonlinear Control Systems*. Springer-Verlag Berlin, Heidelberg 1985 and 1989.

[51] Jin Liang, P. N. Nikiforuk and M. M. Gupta (1994a): Absolute Stability Conditions for Discrete-Time Recurrent Neural Networks. *IEEE trans. on Neural Networks*, Vol. 5, No. 6, pp. 954-964.

[52] Jin Liang, P. N. Nikiforuk and M. M. Gupta (1994b): Adaptive control of discrete time nonlinear systems using recurrent neural networks. *IEE Proc.-Control Theory Appl.,* Vol. 141, No. 3, pp. 169-176.

[53] Jin Liang, P. N. Nikiforuk and M. M. Gupta (1995): Fast Neural Learning and Control of Discrete-Time Nonlinear Systems. *IEEE trans. on Systems, Man and Cybernetics,* Vol. 25, No. 3, pp. 478-488.

[54] Johanssen, T. A. (1994): *Operating Regime Based Process Modelling and Identification*, Dr. Ing. Thesis, Report 94-109-W, Department of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim, Norway, 224 p.

[55] Jokinen, P. (1991): *Continuously Learning Nonlinear Networks with Dynamic Capacity Allocation.* Ph.D Thesis, Tampere University of Technology, Publications 73, Tampere, Finland.

[56] Kailath, T. (1980): *Linear Systems.* Prentice-Hall, Englewood Cliffs, N. J.

[57] Keeler, J. D. (1994): Prediction and control of chaotic chemical reactor via neural network models. *Journal A,* Vol. 35, No. 3, pp 54-57.

[58] Kentridge, R. W. (1990): Neural networks for learning in the real world: representation, reinforcement and dynamics. *Parallel Computing,* 14, pp. 405-414.

[59] Khalid, M. and S. Omatu (1992): A Neural Network Controller for a Temperature Control System. *IEEE Control Systems,* June 1992, pp. 58-64.

[60] Khalid, M., S. Omatu, R. Yosof (1994a): Adaptive Fuzzy Control of a Water Bath Process with Neural Networks. *Engng. Applic. Artif. Intell.,* Vol. 7, No. 1, pp. 39-52.

[61] Khalid, M., S. Omatu, R. Yosof (1994b): MIMO furnace control with Neural Networks. *IEEE trans. on Control Systems Technology,* Vol. 1, No. 4, pp. 238-245.

[62] Kimpimäki, P. (1990): *Adaptive neural network controller.* Tampere University of Technology, Diploma thesis (in finnish).

[63] Kohonen, T. (1984): *Self-Organization and Associative Memory.* Springer-Verlag, Berlin.

[64] Koivisto, H. (1990): Minimum Prediction Error Neural Controller. *Proc. 29th IEEE Conference on Decision and Control*, Honolulu, 5-7 Dec. 1990, pp. 1741-6 Vol. 3

[65] Koivisto, H., P. Kimpimäki, H. Koivo (1991a): Neural Net Based Control of the Heating Process. *ICANN-91, International Conference on Artificial Neural Networks*. Espoo, Finland, June 24-28, 1991, pp. II-1277-1280.

[66] Koivisto, H., P. Kimpimäki, H. Koivo (1991b): Neural Predictive Control - a Case Study. *1991 IEEE International Symposium on Intelligent Control,* Arlington, Virginia, Aug 13-15, 1991, pp. 405-410.

[67] Koivisto, H., V. Ruoppila, H. N. Koivo (1992): Properties of the neural network internal model controller. *Preprints of 1992 IFAC/IFIP/IFORS International Symposium on Artificial Intelligence on Real-Time Control (AIRTC'92),* June 16-18, 1992, Delft*,* Netherlands, pp. 221-226.

[68] Koivisto, H, V. Ruoppila and H. N. Koivo (1993): Real-time neural network control - and IMC approach. *IFAC World Congress 1993,* July 18-13, Sydney, Australia, pp. IV-47-53.

[69] Koivisto, H. (1994): Neural network control of a pilot heating process - a design example. *International Workshop on Neural Network*s *in District Heating,* 15-16 April 1994, Reykjavik, Iceland.

[70] Kwon, W. H. (1994): Advanced in Predictive Control: Theory and Applications. *'94 Asian Control Conference.* Plenary Talk*,* 41 p.

[71] Landau, Y. (1979): *Adaptive Control, the Model Reference Approach.* Marcel Dekker, New York.

[72] Lane, S. H., D. A. Handelman, J. J. Gelfand (1992): Theory and Development of Higher-Order CMAC Neural Networks. *IEEE Control Systems,* April 92, pp. 23-30.

[73] Lawrence, C., J. L. Zhou, A. L. Titts (1994): *User's Guide for CFSQP Version 2.0: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Ineqality Constraints*. Electrical Engineering Department and Institute for Systems Research ,TR-94-16, University of Maryland.

[74] Lee, M., S. Park (1992): A New Scheme Combining Neural Feedforward Control with Model-Predictive Control. *AIChE Journal,* Feb. 1992, Vol. 38, No 2, pp. 193-200.

[75] Levin, A. U., K. S. Narendra (1993): Control of Nonlinear Dynamical Systems using Neural Networks: Controllability and Stabilization. *IEEE trans. on Neural Networks*, Vol. 4, No. 2, pp. 192-206.

[76] Li, W. C., L. T. Biegler, C. G. Economou and M. Morari (1990): A constrained pseudo-Newton control strategy for nonlinear systems. *Computers Chem. Engng*, Vol. 14, No. 4/5, pp. 451-468.

[77] Lichtenwalner, P. F. (1993): Neural network control of the fiber placement composite manufacturing process. *Journal of Materials Engineering and Performance,* Vol. 2, No. 5, pp. 687-691.

[78] Lightbody G., G. W. Irwin (1995): Direct neural model reference adaptive control. *IEE Proc.-Control Theory Appl.,* Vol. 142, No. 1, pp. 31-43.

[79] Liu C., F. Chen (1993): Adaptive control of non-linear continuous-time systems using neural networks - general relative degree and MIMO cases. *Int J. Control,* Vol. 58, No. 2, pp. 317-335.

[80] Ljung, L. (1987): *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, N. J.

[81] Ljung, L., S. Söderström (1983): *Theory and Practice of Recursive Identification*. MIT Press.

[82] Lu, W., D. G. Fisher (1990): Nonminimal Model Based Long Range Predictive Control. *Proc. of 1990 American Control Conference (ACC-90)*, San Diego, California, May 23-25 1990, pp. 1607-1613.

[83] Lu, W., D. G. Fisher *et al.* (1990): Nonminimal Model Based Output Predictors. *Proc. of 1990 American Control Conference (ACC-90)*, San Diego, California, May 23-25 1990, pp. 998-1003.

[84] MacArthur, J. W. (1993): An End-Time Constrained Receding Horizon Control Policy. *Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control,* Vol. 115, pp. 334-340.

[85] McCulloch, W. S., W. Pitts (1943): A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics* **5**, 115-133, 1943. Reprinted in Anderson and Rosenberg [3].

[86] Michie, D., R. A. Chambers, BOXES: An Experiment in Adaptive Control. *Machine Intelligence,* Vol. 2, pp. 137-152, 1968.

[87] Miller III, W. T. (1989): Real-Time Application of Neural Networks for Sensor-Based Control of Robots with Vision. *IEEE Trans. on Systems, Man and Cybernetics,* Vol. 19, No. 4, July/August 1989, pp. 825-831.

[88] Miller, W., R. Sutton, P. Werbos (Eds.) (1990): *Neural Networks in Control*. MIT Press.

[89] Mills, P. M., A. Y. Zomaya and M. O. Tade (1994): Adaptive model-based control using neural networks. *Int. J. Control,* Vol. 60, No. 6, pp. 1163-1192.

[90] Mimoux, M (1986): *Mathematical Programming, Theory and Algorithms.* John Wiley & Sons.

[91] Miyata, Y. (1990): *A User's Guide to Planet Version 5.6.* Computer Science Department, University of Colorado, Boulder.

[92] Monaco, S. and Normand-Cyrot, D. (1986): Nonlinear systems in discrete time. In *Fliess and Hazewinkel* [29], pp. 411-430.

[93] Moody, J., J. Darken (1989): Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation,* Vol. 1, pp. 281-294.

[94] Morari, M., E. Zafiriou (1989): *Robust Process Control.* Prentice-Hall.

[95] Nahas, E. P., M. A. Henson and D. E. Seborg (1992): Nonlinear Internal Model Control Strategy for Neural Network Models. *Computers chem. Engng,* Vol. 16, No. 12, pp. 1039-1057.

[96] Narendra, K. S., K. Parthasarathy (1990): Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks,* Vol. 1, No. 1, pp. 4-27.

[97] Narendra, K. S., K. Parthasarathy (1991): Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks. *IEEE Transactions on Neural Networks,* Vol. 2, No. 2, pp. 252-263.

[98] Neural Networks (1988), Vol. 1, No. 1.

[99] Nguyen, D. H., B. Widrow (1990): Neural Networks for Self-learning Control Systems. *IEEE Control Systems Magazine,* April 1990, pp.18-23.

[100] Nikolaou, M. and V. Hanagandi (1993): Control of Nonlinear Dynamical Systems Modelled by Recurrent Neural Networks. *AIChE Journal,* November 1993, Vol. 39, No. 11, pp 1890-1994.

[101] Niu S. S. and P. Pucar (1995): *Hinging Hyperplanes for Non-Linear Identification.* Linköping University, Division of Automatic Control, Sweden, 23 p. *ftp://joakim.isy.liu.se/*

[102] Oja, E. (1989): Neural networks, principal components, and subspaces. *International Journal of Neural Systems,* Vol. 1, pp 61-68.

[103] Ogata, K. (1987): *Discrete-Time Control Systems.* Prentice-Hall Inc.

[104] Pao Yoh-Han (1992): The functional link net approach to the learning of real-time optimal control. *Preprints of 1992 IFAC/IFIP/IFORS International Symposium on Artificial Intelligence on Real-Time Control (AIRTC'92),* June 16-18, 1992, Delft, Netherlands, pp. 35-41.

[105] Psichogios, D. C., L. H. Ungar (1991): Direct and Indirect Model Based Control Using Artificial Neural Networks. *Ind. Eng. Chem. Res,* Vol. 30, pp. 2564-2573.

[106] Poggio, T., F. Girosi (1990): Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks. *Science,* Vol. 247, pp 978-982.

[107] Press, W. H *et al.* (1986): *Numerical Recipes, the Art of Scientific Computing.* Cambridge University Press.

[108] Priestley, M. B. (1988): *Non-linear and Non-stationary Time Series Analysis.* Academic Press, San Diego.

[109] Pröll T. and M. N. Karim (1994): Model Predictive pH Control Using Real-Time NARX Approach. *AIChE Journal,* Feb. 1994, Vol. 40, No. 2, pp. 269-282.

[110] Psaltis, D., A. Sideris, A. A. Yamamura (1988): A Multilayered Neural Network Controller. *IEEE Control Systems Magazine*, April 1988, pp. 17-20.

[111] Raiskila, P., H. N. Koivo (1990): Properties of a Neural Network controller. *ICARV '90 International Conference on Automation, Robotics and Computer Vision*, 19-21 Sept. 1990, pp. 1-5.

[112] Rissanen, J. (1978): Modelling by shortest data description, *Automatica,* Vol. 14, pp. 465-471.

[113] Rosenlicht, M. (1968): *Introduction to analysis.* Dover Publications, New York.

[114] Rovithakis, G. A. and M. A. Christodoulou (1994): Adaptive Control of Unknown Plants Using Dynamical Neural Networks. *IEEE trans. on Systems, Man and Cybernetics,* Vol. 24, No. 3, pp. 400-412.

[115] Rumelhart, D. E., G. E. Hinton, R. J. Williams (1986): Learning internal representation by error backpropagation. In Rumelhart and McClelland [116], Vol. 1, Chapter 8.

[116] Rumelhart, D. E., J. L. McClelland (Eds.) (1986): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press.

[117] Saint-Donat, J., N. Bhat, T. J. McAvoy (1991): Neural net based model predictive control. *Int. J. Control,* Vol. 54, No. 6, pp. 1453-1468.

[118] Sanner, R. M. and J-J. E. Slotine (1991): Gaussian Networks for Direct Adaptive Control. *1991 American Control Conference (ACC-91)*, pp. 2153-2159.

[119] Sbarbaro-Hofer, D., D. Neumerkel and K. Hunt (1993): Neural Control of a Steel Rolling Mill. *IEEE Control Systems,* June 1993, pp. 69-75.

[120] Scales, L. E. (1985): *Introduction to Non-linear Optimization* (Macmillan Computer Science Series), Macmillan Publishers Ltd, London and Basingstoke.

[121] Scott, G. M. and W. H. Ray (1993): Experiences with model-based controllers based on neural network process models. *Journal of Process Control,* Vol. 3, No. 4, pp. 179-196.

[122] Shiraishi, H., S. L. Ipri, D. D. Cho (1995): CMAC Neural Network Controller for Fuel-Injection Systems. *IEEE trans. on Control Systems Technology,* Vol. 3, No. 1, pp. 32-38.

[123] Shook, D. S., C. Mohtadi, S. L. Shah (1991): Identification for Long-Range Predictive Control. *IEE Proceedings-D*, Vol. 138, No. 1, Jan. 1991, pp. 75-84.

[124] Shook, D. S., C. Mohtadi, S. L. Shah (1992): A Control-Relevant Identification Strategy for GPC. *IEEE trans. on Automatic Control*, Vol. 37, No. 7, pp 975-980.

[125] *Simnon User's Guide for UNIX Systems Version 3.1.* SSPA Systems. Göteborg, Sweden, 1991.

[126] Slotine, J-J. E. and W. Li (1991): *Applied nonlinear control.* Prentice-Hall.

[127] Soeterboek, R (1990): *Predictive Control - a Unified Approach*. Dissertation, Technical University of Delft.

[128] Sontag, E. D.(1990): Feedback stabilization using two-hidden-layer nets. *Report SYCON-90-11*. Department of Mathematics, Rutgers University, New Brunswick.

[129] Sutton, R. S. (1984): *Temporal Credit Assignment in Reinforcement Learning*, Doctoral Dissertation, COINS Tech. Rept. 84-02, Univ. of Massachusetts.

[130] Sutton, R. S. (1988): Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, Vol. 3, pp. 9-44.

[131] Sørheim, E. (1990): A combined network architecture using ART-2 and backpropagation for adaptive estimation of dynamical systems. *Modelling, Identification and Control,* Vol. 11, No. 4, pp. 191-199.

[132] Tam. Y. (1993): An architecture for adaptive neural control. *Journal A,* Vol. 37, No. 4, pp 12-16.

[133] Tenorio, M. F. (1990): Topology synthesis networks: self organization of structure and weight adjustment as a learning paradigm. *Parallel Computing,* Vol. 14, pp. 363-380.

[134] Tolle, H., E. Ersu (1992): *Neurocontrol.* Springer-Verlag, Berlin, 211 p.

[135] Turner P., G. A. Montague, A. J. Morris (1994): Neural networks in process plant modelling and control. *Computing & Control Engineering Journal,* Vol. 5, No. 3, pp. 131-134.

[136] Tzirkel-Hancock, E. and F. Fallside (1991): *A Direct Control Method For a Class of Nonlinear Systems Using Neural Networks*. CUED/F-INFENG/TR.65, March 1991 Cambridge University Engineering Department, Cambridge, England,.

[137] Tzirkel-Hancock, E. and F. Fallside (1992): Stable Control of Nonlinear Systems Using Neural Networks. *International Journal of Robust and Nonlinear Control,* Vol. 2, pp 63-86.

[138] Vadstrup, P. (1988): *Adaptiv Regulering af Ikke-lineære Systemer.* Licentiatafhandling, Servolaboratoriet, Danmarks Tekniske Højskole.

[139] Weigend, A. S., B. A. Huberman, D. E. Rumelhart (1990): Predicting the Future: A Connectionistic Approach. *International Journal of Neural Systems,* Vol. 1, No. 3, pp. 193-209.

[140] Werbos, P. (1989): Neural Networks for Control and System Identification. *28th IEEE Conference on Decision and Control*, Tampa, Florida, Dec. 89, pp. 260-265.

[141] White, D. A. and D. A. Sofge (Eds.) (1992): *Handbook of Intelligent Control.* Van Nostrand-Rheinhold, New York.

[142] Widrow, B. (1990): 30 years of adaptive neural networks: perceptron, madaline and backpropagation, *proc. IEEE,* Aug. 1990, 27p.

[143] Widrow, B., D. E. Rumelhart and M. A. Lehr (1994): Neural Networks: Applications in Industry, Business and Science. *Communications of the ACM,* Vol. 37, No. 3, pp. 93-105.

[144] Willis, M. J., G. A. Montague *et al.* (1992): Artificial Neural Networks in Process Estimation and Control. *Automatica,* Vol. 28, No. 6, pp. 1181-1187.

[145] Wu, Q. H., B. W. Hogg and G. W. Irwin (1992): A Neural Network Regulator for Turbogenerators. *IEEE trans. on Neural Networks,* Vol. 3, No. 1, pp. 95-100.

[146] Yabuta, T., T. Tujimura, T. Yamada, T. Yasuno (1989): On the Characteristics of the Robot Manipulator Controller using Neural Networks. *Proc. of International Workshop on Industrial Applications of Machine Intelligence and Vision,* April 10-12, 1989, Rappongi, Tokyo, pp. 76-81.

[147] Ydstie, B. E. (1990): Forecasting and Control using Adaptive Connectionistic Networks. *Computers Chem. Engng*, Vol. 4/5, pp. 583-599.

[148] Ye, K., K. Fujioka and F. Shimizu (1994): Efficient control of fed-baker's yest cultivation based on neural networks. *Process Control & Quality,* Vol. 5, No. 4, pp. 245-250.

[149] Zafiriou, E. (1990): Robust Model Predictive Control of Processes with Hard Constraints. *Computers Chem. Engng,* Vol. 14, No. 4/5, pp. 359-371.

[150] Zafiriou, E. and A. L. Marchal (1991): Stability of SISO Quadratic Dynamic Matrix Control with Hard Output Constraints. *AIChE Journal,* Vol. 37, No. 10, pp. 1550-1560.

[151] Zeman, V., R. V. Patel, K. Khorasani (1989): A neural network based control strategy for flexible-joint manipulators. *28th IEEE Conference on Decision and Control*, Tampa, Florida, Dec. 89, pp. 1759-1764, 1989.

[152] Åström, K. J., B. Wittenmark (1990): *Computer Controlled Systems*. Prentice-Hall, 543 pages.

Appendix A Multilayer Perceptron

A multilayer perceptron (MLP) neural network (Fig. A.1) implements a smooth and continuous mapping $f: \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$ i.e the prediction is

$$\hat{y}(t) = \hat{y}(t|\theta) = f(x(t), \hat{\theta}) \tag{A.1}$$

for a sample input $x(t)$, where $\hat{\theta}$ is an estimate of the parameters θ. The index t corresponds to a sample t or to time t, if a time series is considered. Introduce the prediction error

$$\varepsilon(t) = \varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta) \tag{A.2}$$

and the cost function increment

$$V(t, \theta) = \frac{1}{2} \| y(t) - \hat{y}(t|\theta) \|_{\Lambda}^2 \tag{A.3}$$

and its negative gradient $\psi(t)$ with respect to the parameters θ as

$$\psi(t) = \left[-\frac{\partial}{\partial \theta} V(t, \theta) \right]^T \tag{A.4}$$

is an n_θ dimensional vector. A column vector is exceptionally used here.

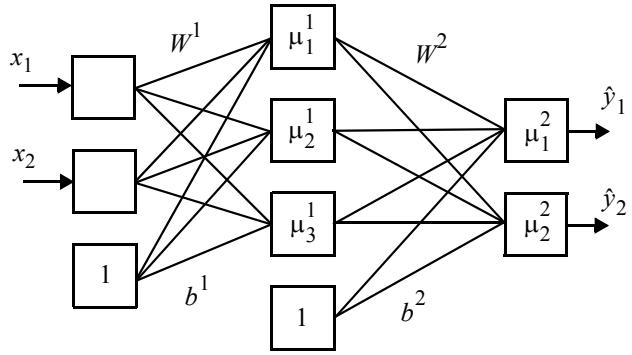


Fig. A.1. Block diagram of the multilayered perceptron network

Define the Jacobian matrices of the (A.1) w.r.t. θ and x as

$$\Xi(t) = \left[\frac{\partial}{\partial\theta} f(x, \theta)\right]\Big|_{\theta = \hat{\theta},\, x = x(t)} \tag{A.5}$$

$$\Gamma(t) = \left[\frac{\partial}{\partial x} f(x, \theta)\right]\Big|_{\theta = \hat{\theta},\, x = x(t)} \tag{A.6}$$

where $\dim(\Xi) = m \times n_\theta$ and $\dim(\Gamma) = m \times n$.

Introduce also a diagonal function $\mu : \boldsymbol{R}^k \rightarrow \boldsymbol{R}^k$

$$\mu(v) = [\mu_1(v_1) \ldots \mu_k(v_k)]^T \tag{A.7}$$

where each $\mu_i(v_i)$ is a scalar function. The Jacobian of the diagonal function is

$$\frac{\partial\mu(v)}{\partial v} = \begin{bmatrix} \frac{d\mu_1(v_1)}{dv_1} & & \\ & \ddots & \\ & & \frac{d\mu_k(v_k)}{dv_k} \end{bmatrix} \tag{A.8}$$

Consider now MLP network with L layers with following definitions (see Fig. A.1):

x is the n dimensional input vector to the network (layer 0)

$\hat{y}$ is the m dimensional output vector of the network (layer L)

v^s is the m^s dimensional input vector to layer s

x^s is the m^s dimensional output vector of the layer s. ($x^0 = x$ and $\hat{y} = x^L$)

W^s is the weight matrix between the layers $s-1$ and s

b^s is the treshold (bias) vector to the layer s

θ is the parameter vector which consists of all weights W^s and bias values b^s ordered to a n_θ dimensional vector

$\mu^s : \boldsymbol{R}^{m^s} \rightarrow \boldsymbol{R}^{m^s}$ is a diagonal activation function of the layer s, where $\mu_i : \boldsymbol{R} \rightarrow \boldsymbol{R}$ is a scalar activation function. Without a loss of generality it is assumed that all activation functions in a layer are same. The activation function is usually some sigmoid shape function, for example $\mu_i(v_i) = 1/(1 + e^{-v_i})$ or $\mu_i(v_i) = \tanh(v_i)$. The outermost activation function can also be a linear function.

The terms *forward pass* and *backward pass* are used to denote the calculation of the predictions *(forward*) and the gradients *(backward*). Different backward pass types will be defined for computing $\psi(t)$, $\Xi(t)$ and $\Gamma(t)$. To maintain the simplicity and the clarity of the presentation, a procedural pseudo code with matrix and vector notation is used. Those fond of successive Σ sentences, see for example Hertz *et al.* (1991).

Forward pass

The task of the *forward pass* is to compute the prediction

$$\hat{y}(t) = f(x(t), \hat{\theta}) \tag{A.9}$$

for a sample input $x(t)$, The parameter vector θ consist of the weight matrices W^s and bias vectors b^s with the columns stacked under each other.

Procedure: ***Forward***

extract all W^s *and* b^s *from the parameter vector* $\hat{\theta}$

set $x^0 = v^0 = x(t)$

repeat for layers $s = 1 \ldots L$

$v^s = W^s x^s + b^s$

$x^s = \mu^s(v^s)$

end

set $\hat{y}(t) = x^L$

Backward passes

The task of the first *backward pass* is to calculate the negative gradient $\psi(t)$ of the cost function increment (A.3) with respect to the parameters θ . The application of the chain rule differentation can be viewed as a forward pass of the linearized and "transposed" network (see Fig. A.1), where:

e^s is the "error" input of the layer s

M^s is the Jacobian matrix of the diagonal activation function w.r.t. its input

Procedure: ***Backward_plain***

set $e^L = \Lambda\varepsilon(t, \hat{\theta})$

repeat from layer $s = L$ *to layer* $s = 1$

$$M^s = \frac{\partial}{\partial v^s}\mu^s(v^s)$$

$$e^s = M^s e^s \quad ^{1)}$$

$$\partial V / \partial W^s = e^s (x^s)^T$$

$$\partial V / \partial b^s = e^s$$

$$e^{s-1} = (W^s)^T e^s$$

end

form $\psi(t)$ *by stacking all* $\partial V / \partial W^s$ *and* $\partial V / \partial b^s$ *with the same order as the parameters* W^s *and* b^s *are in* θ.

This procedure can be applied to all gradient based optimization methods which uses a scalar cost function (like BFGS). It is a part of the standard error backpropagation algorithm, where the parameter update is made according to

$$\Delta\hat{\theta}(t) = \eta\psi(t) + \alpha\Delta\hat{\theta}(t-1) \tag{A.10}$$

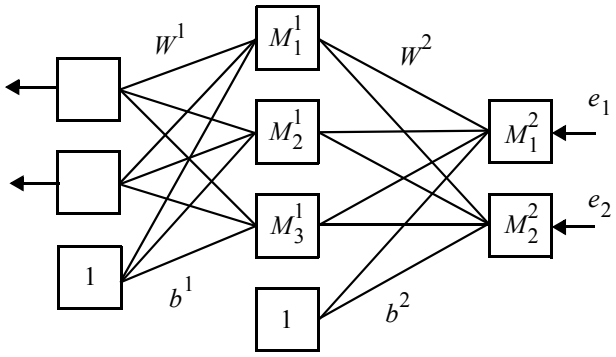


Fig. A.2. Block diagram of the MLP network during the backward phase.

1. In practice implemented with element-by-element multiplication.

where η is the learning rate and $\alpha \in [0, 1]$ is a momentum term, typically $\alpha = 0{,}9$. The (A.10) is a incremental form of the algorithm, suitable for on-line learning. In adaptive applications the $\psi(t)$ can also be filtered with a linear first order filter. A batch version

$$\Delta\hat{\theta}(k) = \eta \sum_{t=1}^{N} \psi(t) + \alpha\Delta\hat{\theta}(k-1) \tag{A.11}$$

where k is the iteration counter, is more effective in off-line applications. These are not used in this study, except in two real-time adaptive control experiments.

Backward pass (type 2)

The task in to compute the Jacobians $\Xi(t)$ and $\Gamma(t)$ with $\dim(\Xi) = m \times n_\theta$ and $\dim(\Gamma) = m \times n$. For nonlinear least squares method, these Jacobians must be computed separately for each output using this backward pass. Two different procedures are presented:

- **backward_only** for computing only $\Gamma(t)$
- **backward_and_gradient** for computing $\Xi(t)$ and $\Gamma(t)$

Procedure: ***Backward_only***

repeat for layers $s = 1 \ldots L$

$$M^s = \frac{\partial}{\partial v^s}\mu^s(v^s)$$

end

repeat for outputs $j = 1 \ldots m$

set $e^L = [0 \ldots 1 \ldots 0]^T$ *with* 1 *at the* j *position*

repeat from layer $s = L$ *to layer* $s = 1$

$$e^s = M^s e^s$$

$$e^{s-1} = (W^s)^T e^s$$

end

set $\Gamma^{(j)}(t) = e^0$ *(to the* j*th column)*

end

Procedure: ***Backward_and_gradient***

repeat for layers $s = 1 \ldots L$

$$M^s = \frac{\partial}{\partial v^s} \mu^s(v^s)$$

end

repeat for outputs $j = 1 \ldots m$

set $e^L = [0 \ldots 1 \ldots 0]^T$ *with* 1 *at the* j *position*

repeat from layer $s = L$ *to layer* $s = 1$

$$e^s = M^s e^s$$

$$\partial f_j / \partial W^s = e^s (x^s)^T$$

$$\partial f_j / \partial b^s = e^s$$

$$e^{s-1} = (W^s)^T e^s$$

end

form $\Xi^{(j)}(t)$ *(*j*th column) by stacking all* $\partial f_j / \partial W^s$

and $\partial f_j / \partial b^s$ *with the same order as parameters in* θ*.*

set $\Gamma^{(j)}(t) = e^0$ *(to the* j*th column)*

end

Backward pass (type 3)

For constrained minimization also the second order Jacobian must be determined. This can be done easily for networks with one hidden layer. For more complicated networks the second order Jacobian $\partial\hat{y} / \partial x \partial\theta$ must be approximated numerically. Define matrices

$$\chi(i) = \frac{\partial}{\partial x \partial \theta_i} f(x, \theta)\Big|_{\theta = \hat{\theta},\, x = x(t)}, \quad i = 1 \ldots n_\theta \tag{A.12}$$

where θ_i is the i th term in θ and $\dim(\chi(i)) = m \times n$.

procedure: ***backward_second_order***

do forward pass with the nominal parameters θ

do backward_only pass and store the result in $\Gamma(t)$

repeat for each parameter $\theta_i \;\; i = 1 \ldots n_\theta$

set $\tilde{\theta} = \theta$ *and* $\tilde{\theta}_i = \theta_i + \Delta\theta$

do forward pass using $\tilde{\theta}$

do backward_only pass and store the result in $\tilde{\Gamma}(t)$

$$\chi(i) = \frac{\tilde{\Gamma}(t) - \Gamma(t)}{\Delta\theta}$$

end