CONNECTIONIST FEEDFORWARD NETWORKS FOR CONTROL OF NONLINEAR SYSTEMS

A DISSERTATION

SUBMITTED TO THE FACULTY OF ENGINEERING

OF GLASGOW UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Daniel G. Sbarbaro Hofer

October 1992

Acknowledgements

I wish to express my gratitude to my supervisor Prof. P.J. Gawthrop for proposing this field of research, for his valuable help, and guidance during the development of this work. To the Chilean Government and University of Concepción for funding this research. I am also indebted to my family for their support and encouragement.

I wish to thank Dr. K. Hunt for his support during difficult stages in the development of this work, Mr. Rafał Żbikowski for the many suggestions to improve the final version of this thesis and for proof-reading the manuscript, Dr. P. Larcombe for reading part of the manuscript, and finally to the Control Group for their friendship and help.

Thanks to Daimler-Benz (Forschungsinstitut-Berlin) for giving me the opportunity to study some interesting problems. Thanks to Rod Murray-Smith and Dr. Frieder Lohnert for their help during my stay in Berlin.

I wish also to thank Gloria, the Chilean five a side team, and *ma friens* in Glasgow and abroad for their support.

Finally, I would like to thank Maria and her parents for their help, kindness and for making me feel at home in Scotland.

Difficult things in the world must needs have their beginnings in the easy; big things must needs have their beginnings in the small.

Lao Tzu, "Tao Te Ching, LXIII-149a"

Abstract

The control of nonlinear systems is addressed from a new perspective, which makes use of several concepts and techniques developed in the area of Artificial Neural Networks. In particular, this work explores the potential of connectionist representations which consist of only feedforward connections of sigmoids or gaussian units. A unified review demonstrating the capabilities of these structures to approximate continuous nonlinear functions is given. The use of the Fourier transform and the properties of kernel functions are exploited in order to demonstrate some properties of gaussian networks.

The adjustment of the different parameters plays a significant role in the relevance of these structures in control. For sigmoid networks a new learning algorithm is proposed, its main feature being the use of the forgetting factor and pseudoinverse. For gaussian networks several algorithms using different techniques, such as: the Fourier Transform, gradient approach, and/or clustering algorithm, are explored and compared.

The representation of a dynamic system by means of a static nonlinear function, and consequently, by a connectionist representation, is addressed. Concepts such as controllability, observability, and invertibility, which are needed to develop any control structures, are put forward for the nonlinear case.

Four control structures using connectionist models to generate the control signal are proposed, and its potential analysed. For each approach a simple example is presented to illustrate their performance. A summary of their main characteristics is also given.

Two industrial applications have been tackled, and solutions developed, illustrating not only the differences between different techniques, but also the potential and limitations of the ideas pursued in this work.

Finally, some suggestions are given, which may engender further research in the field of control and artificial neural systems.

Preface

0.1 Motivation and Scope

Controlling nonlinear systems is very important from an industrial point of view, because all of the industrial processes are nonlinear. The trend in the industry is towards imposing of more and more demanding constraints on the production process to satisfy a competitive market which requires high quality goods at low prices. From this point of view nonlinear control techniques are highly desired, because they can provide a tighter control than their linear counterparts. On the other hand, Artificial Neural Networks, one of the most promising new technologies emerging, with inputs from different fields which range from psychology to physics, provides a new framework to develop nonlinear structures amenable for control systems. The main aim of this thesis is to explore the capabilities of a subset of Artificial Neural Networks (feedforward networks) within the nonlinear control framework.

0.2 Contributions

The main contributions of this work can be summarised as involving the following:

- The use of a nonlinear estimation algorithm instead of Back-Propagation, establishing proper links between learning algorithms and the system identification theory.

- The use of a forgetting factor and pseudoinverse is proposed within the context of multilayer networks and learning.
- A comparison of learning algorithms for gaussian networks and their simplification for real applications.
- The use of Frequency Approach to analyse regular gaussian networks. Extensions of basic properties to more general representations such as Hermite networks.
- The setting up of a general formalism to integrate discrete nonlinear systems and control structures.
- The use of parametric models to represent nonlinear relations and the extension of linear control structures to deal with nonlinear systems.
- The establishment of the limitations and advantages of four different nonlinear control approaches using connectionist representations.
- The presentation of application examples with a clear industrial orientation to illustrate the performance and limitations of the different approaches analysed.

0.3 Thesis Outline

The thesis is divided into seven chapters, starting with an introduction, which contains a brief historical perspective and a general description of the basic ideas behind this work.

The remaining chapters are organised in order to give a coherent exposition of the work that has been completed. We start from a general description of the different feedforward architectures used as parametric representations of nonlinear functions and their training algorithms. Then, the problem of representing a nonlinear dynamical system using a nonlinear function of delayed inputs and outputs is addressed. Following these results, different nonlinear control structures are generated, which are analysed and applied to two industrial processes.

The work starts with Chapter 2, giving the necessary background material needed to develop the basic representation of a nonlinear system using a connectionist representation. The main contributions of this chapter are the unified presentation of the different structures, and the use of Fourier transform and properties of kernel functions to demonstrate the approximation capability of a regular gaussian network.

Chapter 3 gives a comparison among different algorithms for feedforward networks using two different activation functions. The main contributions of this chapter are a new algorithm based on linearisation, and the introduction of a forgetting factor and pseudoinverse within the learning framework for sigmoid networks, together with the use of the Fourier transform to analyse gaussian networks, and a new version of a known scheme to train gaussian networks with different algorithms working in parallel.

Chapter 4 introduces both the feedforward architecture needed to represent nonlinear dynamic systems, and presents the concepts of controllability and observability for nonlinear system, which are used in later chapters. This chapter addresses fundamental questions with regard to realisation, i.e., which kind of system can be modelled with a feedforward network, plus the invertibility of this model.

The usage of nonlinear models to control a nonlinear system is discussed in Chapter 5. The approaches analysed are Direct Inverse Control, Model Reference Control, Internal Model Control and Receding Horizon Control. Every approach is analysed, and its behaviour under different conditions is illustrated by basic simulation examples. The original contribution of this chapter is to highlight the problems of some known schemes (Inverse Control and Model Reference), and to propose other control strategies (Internal Model Control and Receding Horizon

Control), which overcome the difficulties encountered with Inverse Control and Model Reference.

In Chapter 6 two industrial applications are described in detail. The first one is the control of a pH plant, and the second one is the control of a steel mill strip thickness.

This work is concluded with Chapter 7, where appropriate conclusions are given, and several extensions to the methodology are also suggested .

Contents

List of Tables

List of Figures

Chapter 1

Introduction

SUMMARY

Firstly, in this introduction there is a description of the basic framework used to establish the relationship between connectionist representation and control. Secondly, a brief historical review of both fields is given, illustrating the many instances when these two disciplines have complemented each other.

1.1 Basic Framework

As J.L. Borges describes in Averroes' Search [9], our research is bounded by our knowledge. He narrates the process of a defeat. He tells of the case of a man who set himself a goal which is not forbidden to others, but is to himself. He wrote:

> I remembered Averroes who, closed within the orb of Islam, could never know the meaning of the terms tragedy and comedy. ... I felt that Averroes, wanting to imagine what a drama is without ever having suspected what a theatre is, was no more absurd than I, wanting to imagine Averroes with no other sources than a few fragments from Renan, Lane and Asín Palacios.

This leads me to picture myself, trying to write a Thesis on Neural Networks and control just with a limited reference of a few hundred papers. This is the reason why in this thesis some very simple ideas from the Artificial Neural Network have been taken and applied to control problems. The aim of this thesis is not to give physiologically plausible architectures and algorithms, but rather to give ones that are useful in engineering applications. Obviously, the theory is not yet complete, nor have its implications and possible applications been fully explored. It should be regarded as only a first step in contributing to the joint analysis of connectionist systems and control systems, clarifying the potentialities and the limitations of these new developments within the system theory framework. To expect something different would be unrealistic.

Figure 1.1 illustrates the basic structure used in this work, in which the control problem is *decomposed* in two subproblems: representation and control.

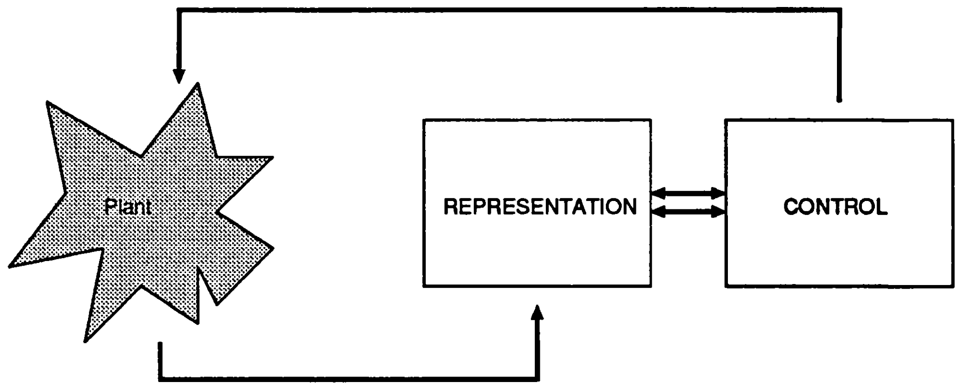


Figure 1.1: Decomposition of a controller in representation and control.

In order to control a system it is necessary to know something about it. This knowledge is used within a *representation* framework, which is itself a model of the reality. Once this representation is available, it is possible to generate strategies or control signals in order to drive the system to a desired operational point or trajectory. This concept can be found in the more traditional Artificial Intelligence framework, as it is described explicitly by Newell [62] :

> An intelligent agent is embedded in a *task environment; a task statement* enters via a *perceptual* component and it is encodeded in

> an initial *representation.* Whence starts a cycle of activity in which a *recognition* occurs ... of a method implemented to undertake the problem. The method draws upon a memory of *general world knowledge.* ... It is clear to us all that representation is in this picture. It is a data structure that holds the problem and will be processed into a form that makes the solution available. Additionally, it is the data structure that holds the world knowledge, and will be processed to acquire parts of the solution or to obtain guidance in constructing it.

As the first stage, this functional decomposition is conducive to tackling two different problems individually, and has the additional advantage of permitting incremental refinement, i.e., if a better representation is found then it can be included in the system without formulating the control scheme again. Under this scheme, as the representation accumulates more information about the unknown plant, the system's control will be altered according to the updated information in order to improve the system's performance.

The relationship between the control scheme and representation is usually chosen by the designer. The alternative approaches used throughout the design of the controller section are:

- Methods based on the use of nonlinear functional analysis, in which the system components are represented as general nonlinear operators.
- Control design methods for nonlinear systems employing optimisation techniques.

There is a third approach based on linearisation of the nonlinear system. This, however, was not pursued here.

The concept of representation and memory are closely related. A simple definition of memory is "preservation of the past experience for future use" [33]. Hence, it is possible to regard a representation as the manner in which we preserve the past evolution of the plant. One of the contributions of Artificial Neural Networks has been the provision of different representations to serve as memories. The main aim of the work in this thesis, however, involves the exploration of some of the possibilities uncovered when a connectionist representation is employed to control systems.

1.2 Historical Background

This brief summary about neural research is based on the compilation of papers done by Anderson and Rosenfeld [4].

In the early 1940's, the pioneers of the field—MacCulloch and Pitts—studied the potential and capabilities of the interconnections of several basic components based on the model of a neuron. Others, such as Donald Hebb, were concerned with the adaptation laws involved in natural neural systems. Rossenblatt coined the term of Perceptron, and devised an architecture whose potentialities arose out of random connections between elements. In the sixties, Minsky and Papert [56] introduced a rigorous analysis of the perceptron; establishing the computational cost of what perceptrons can do as a function of increasing problem size, they demonstrated rigorously many properties and limitations of the model. In the seventies, the work of S. Grossberg became prominent. His work, based on psychological and biological evidence, proposed several architectures of a nonlinear dynamic system with novel characteristics. Hopfield applied a particular nonlinear dynamic structure to solve technical problems, for example, optimisations. In 1986, the Parallel Distributed Processing (PDP) group published a series of results and algorithms [53]. This gave a strong impetus to the area, and provided the catalyst for much of the subsequent research in this field.

Parallel to all this development in the control field, some algorithms and concepts were developed and analysed. The first person who saw the control theory as a broader discipline involving Artificial Neural Networks was N. Wiener (1948).

One of the approaches utilised to deal with variations in the parameters of a linear representation of a time-invariant nonlinear system was the adaptation of the parameters. From an Artificial Neural Network perspective these adaptive systems can be regarded as systems which are able to modify their representation of their environment and therefore capable of learning.

The early work in the 1950s on adaptive systems came from the studies of simple adaptive laws designed using the sensitivity approach (MIT rule), and later, in the 1960s, using Lyapunov theory, which has the advantage of embodying the notion of stability within the design of adaptive systems.

Kalman (1958) [37] not only envisaged the integration of a self tuning controller with explicit identification of the parameters, but also introduced the state space representation as a powerful tool to design a controller for a complex linear system; he also introduced the key system concepts of observability and controllability.

In the 1960s, Bellman made important contributions to the understanding of dynamic programming, and Tsypkin [87] proposed several learning schemes based on the stochastic approximation idea. Widrow [91], working in the area of signal processing, also proposed a learning rule for the single layer perceptron. This work was the only one that explicitly related these fields so far.

It is worth mentioning the work done by Aizermann during the 60's [1] in the field of potential functions in nonlinear representation, which can now be seen as one aspect of Artificial Neural Networks research.

In the 1970s and 1980s, most work was concentrated on self-tunning regulators and also on the development of tools to analyse these algorithms. Critical examination and evaluation of the main results to date was also undertaken. Issues such as stability, convergence and robustness were addressed.

The book by Padulo and Arbib [63] set up the current trends in system theory, unifying different branches of this general body of concepts and techniques which are used to analyse and design systems. This unified approach shows that many concepts are available for nonlinear systems, as well as for linear ones.

Following the ideas developed by Kalman and others [38], Hammer [31] developed his own ideas for controlling discrete systems, extending the algebraic approach to nonlinear input-output models. Billings and his collaborators [45] [46] concentrated on the practical side of obtaining a nonlinear model from experimental data, and finally, Sontag [83] developed his theory of polynomial maps to deal with nonlinear representations.

Modern nonlinear control theory, particularly the differential geometric approach, has emerged during the seventies and with a rather successful endeavour to deal with basic questions in the state space formulation of nonlinear control systems, also including the key problems of controllability, observability, and (minimal) realisation theory.

Modelling inaccuracy, which can come from the actual uncertainty about the plant or from purposeful choice of a simplified representation of the dynamical system, has a strong adverse effect on the nonlinear control system. Hence, any practical design must address them explicitly. Two major and complementary approaches to dealing with uncertainty are robust control and adaptive control. One way in which to produce a robust control is by the variable structure approach, which deals with discontinuous control. The main advantages of this technique are: order reduction, decoupling design procedure, disturbance rejection, and robustness to parameter variation. Its final implementation is simple, and resembles the basic characteristic of neurons. Some researchers have already investigated the possibility of using connectionist representations within this area [75] [94].

Chapter 2

Feedforward Networks

SUMMARY

This Chapter describes the basis of connectionist representations using sigmoid and the gaussian functions and their link with Artificial Neural Networks. The capabilities of these representations to approximate any nonlinear continuous function are proved using the Stone-Weiertrass Theorem. In addition, the concept of spatial frequency is introduced and the capabilities of a regular gaussian network are proved using properties of kernel functions. Extensions to Hermite networks are also described.

2.1 Artificial Neural Networks

The field of Artificial Neural Networks studies the properties, characteristics, representations, and the underlying adaptive mechanisms of the real neural system. As described by D. Ballard [5], one of the objectives is to find useful abstract descriptions of the computation performed by the neural system without reducing them to anatomy. The level of formulation is in terms of symbolic constraints and methods for solving them. These useful abstract descriptions generally are parallel and non-algorithmic, sometimes they model a function rather than a physiological aspect of the brain. The term *connectionist* has come to mean the fact that these essential components of abstract level can be described in terms of synaptics connections of networks of neurons. The main potential of the different structures stems from the use of simple basic processing units connected together, which gives origin to an *architecture*. The basic model of an element is not unique; different neurons in different parts of the brain seem to represent information in very specific ways. For example, the activation function, i.e. type of response between the firing rate and the value of the stimulus, for the ocular-motor system neurons has a *sigmoidal* characteristic and for those cortical neurons, from the visual areas, has a *gaussian* characteristic [5]. This suggests that there are certain kinds of functions which can be approximated more efficiently using a suitable activation function.

In this work only these representations which consider feedforward connections with no implicit dynamics are studied.

At a given time t, the ith layer of a multilayer network, Figure 2.1, can be represented by

$$y_i = W_i^T x_i, \tag{2.1}$$

where x_i is given by

$$x_i = \begin{pmatrix} \breve{x}_i \\ 1 \end{pmatrix}, \tag{2.2}$$

and $\breve{x}_i$ by

$$\breve{x}_{i+1} = \bar{f}_i(y_i). \tag{2.3}$$

The last element of x_i corresponds to a scalar offset term when multiplied by the appropriate weight vector. Equation (2.1) describes the *linear* part of the layer, where x_i is a vector of dimension $((n_i + 1) \times 1)$ containing n_i inputs to the layer together with 1 as the last element, W_i is a matrix of dimension $((n_i+1)\times(n_i+1))$, which maps the input to the linear output of the layer, and $\breve{W}_i$ $(n_i \times (n_i + 1))$, a submatrix of W, does not contain the offset weights. Equation (2.3) describes the *nonlinear* part of the layer, where the function $\bar{f}_i$ maps each element y_i, the linear output of the layer, to each element $\breve{x}_{i+1}$, the inputs to the next layer. A special case of the nonlinear transformation of equation (2.3) occurs when each element of the vector $\breve{x}_{i+1}$ is obtained from the corresponding element y of the vector y_i by the same nonlinear function

$$\breve{x}_{i+1} = f(y_i). \tag{2.4}$$

Typical functions $f(y)$ are the *sigmoid* function [53]

$$f(y) = \frac{1}{1+e^{-y}}, \tag{2.5}$$

and the *hyperbolic tangent* function

$$f(y) = \tanh(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}. \tag{2.6}$$

The former is appropriate to asymmetric scaling between 0 and 1; the latter is appropriate to symmetric range between of -1 and 1. The derivative of the function in equation (2.5) is

$$f'(y) = \frac{dx}{dy} = f(y)(1 - f(y)), \tag{2.7}$$

and that of the function in equation (2.6) is

$$f'(y) = \frac{dx}{dy} = (1 + f(y))(1 - f(y)) = 1 - f(y)^2. \tag{2.8}$$

Another useful activation function, but with a rather different characteristic, is the *gaussian* function,

$$f(y) = e^{-\|y-m\|_A^2}, \tag{2.9}$$

where y is a vector of dimension $(n \times 1)$, $\|\cdot\|$ is the weighted norm defined as

$$\|y\|_A^2 = y^T A y,$$

A is a symmetric positive definite matrix of dimension $(n \times n)$. In the simple case of a diagonal A, the diagonal elements a_{ii} assign a specific weight to each

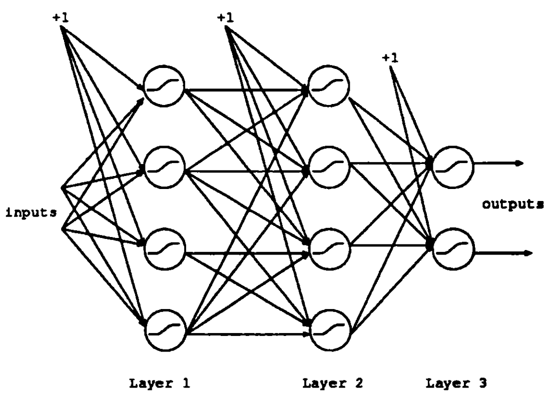


Figure 2.1: Multilayer perceptron.

input coordinate, and the standard Euclidean norm is obtained when A is set to the identity matrix. The centre of the function is defined by the vector m of dimension $(n \times 1)$.

The model for a gaussian network is shown in Figure 2.2 and described by the following equation

$$F(x) = \sum_{i=1}^{N} c_i e^{\|x-m_i\|^2_{A_i}}, \tag{2.10}$$

where N is the number of units, c_i is the weight of unit i, m_i is the centre of unit i, and A_i is related with the bandwidth and orientation of the unit i. This representation can be regarded as a conceptualization of the receptive field[1]. Here, the computation is performed by *gaussian receptive fields* and their *linear combination* . In this case the multidimensional gaussian function could be synthetesised by combining lower dimensional receptive fields, possibly in multiple stages [68]. The synthesis of gaussian functions in many dimensions is easier if they are factorizable, for example

$$e^{-\|x-m_i\|^2_{A_i}} = e^{-a_{11i}(x^1-m_i^1)^2} e^{-a_{22i}(x^2-m_i^2)^2}.$$

where a_{11i} and a_{22i} represent the values of the diagonal matrix A_i. The gaussian function itself can be synthetesised directly by appropriately weighted connections from the sensor arrays, which transduce the implicit position of the stimuli in the sensor array into the activity of the unit.

[1]A receptive field is defined as the response to all neurons' inputs, some of which may be feedback connections from neurons in more abstract cortical areas [5].

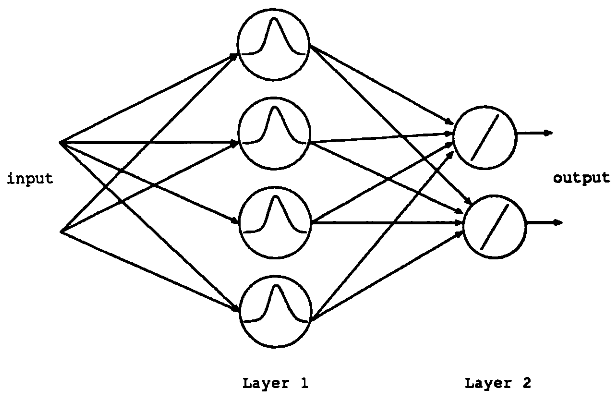


Figure 2.2: Gaussian network.

2.2 Representation Problem

The representation problem addresses the question of whether a determined structure can represent an arbitrarily continuous function. The mathematical tools most commonly used to answer this kind of question fall in two categories: those involving algebra of functions (leading to the Stone-Weierstrass arguments) and those involving translation invariant subspaces [15]. In this work the first approach was followed.

In order to use the theorem some definitions are necessary [72].

Definition:

If U is a metric space, the set of continuous functions mapping U to the real line, R, is denoted by $C[U]$. Given an arbitrary set U the family $\mathcal{A}$ of functions f which map $U \rightarrow R$ is an algebra if f_1, $f_2 \in \mathcal{A} \Rightarrow \alpha f_1 + \beta f_2 \in \mathcal{A}$ and $\alpha f_1 f_2 \in \mathcal{A}$, for all α and $\beta \in R$ □

Definition:

Let $\mathcal{A}$ be the family of functions $f : U \rightarrow R$. Then $\mathcal{A}$ is said to *separate points* on U if for any $u_1 \neq u_2$ of U there exists a function $f \in \mathcal{A}$ such that $f(u_1) \neq f(u_2)$ □

Definition:

$\mathcal{A}$ *vanishes* at no point of U if for any point $u \in U$ there exists a function $f \in \mathcal{A}$, such that $f(u) \neq 0$ □

2.3 The Stone-Weierstrass Theorem

Theorem 1 *Let U be a compact metric space and $\mathcal{A}$ an algebra of $C[U]$. If $\mathcal{A}$ separates points on U and does not vanish at any point of U, then $\mathcal{A}$ is dense in $C[U]$ (i.e. its closure is the whole $C[U]$).*

Proof: [72].

This theorem provides the theoretical framework to set up the potentialities of feedforward networks, but provides neither a training algorithm, nor a useful framework to compare different structures, because the property of approximation is shared by many architectures.

2.3.1 Sigmoids and Multilayer Perceptron

The approximation property of a single layer perceptron is given by the following theorem.

Theorem 2 *Let $s(x)$ be any continuous sigmoidal function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j s(\beta_j^T x + \gamma_j), \quad \beta_j, x \in R^n, \quad \alpha_j, \gamma_j \in R$$

are dense in $C[U]$, where $U \subset R^n$.

In other words, given any $f \in C[U]$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$\|G(x) - f(x)\| < \epsilon \qquad \forall x \in U.$$

Proof: see [15], [34].

Since a hidden layer is adequate to approximate any continuous function, it follows that the same property is also valid for networks with more than one layer [15].

This theorem shows the capability of sigmoid networks to approximate continuous functions. Even though it has been shown that such representation is possible with just one hidden layer, from the practical point of view it may be inconvenient to use only one layer. In fact, the number of units to approximate certain functions (for example, a function peaking at $[0,0]^T$ and equal to 0 everywhere outside of the unit circle) can be very large [13]. This fact was noted by Cybenko [15] and Funahashi [22], as well.

In some cases the coefficients can grow exponentially, and this has serious consequences both practically and conceptually, as pointed out by Minsky and Papert [56]. Questions such as how the coefficients scale with the number of hidden layers to solve the same problem, or what is the optimal number of layers and units to approximate a given function are open.

2.3.2 Gaussian Networks

The Stone-Weierstrass Theorem applies directly to gaussian networks. The fact that the gaussian function is defined in terms of the Euclidean norm implies the closure of the function space under multiplication.

Theorem 3 *Let $U \subset R^n$ be a convex compact set. Let the family $\mathcal{G}$ of functions corresponding to gaussian functions be defined by*[2]

$$g_{m_i,a_i} : U \to R$$

$$g_{m_i,a_i}(x) = e^{-a_i\|x-m_i\|^2}, \quad a_i > 0, m_i \in U, x \in U.$$

Then a finite linear combination with real coefficients of elements of $\mathcal{G}$ is dense in $C[U]$.

Proof:

[2] $\|\cdot\|$ means $\|\cdot\|_I$, where I is the identity matrix.

Multiplying two elements of $\mathcal{G}$ one obtains an element of $\mathcal{G}$ as

$$g_{m_i,a_i}(x)g_{m_j,a_j}(x) = D_{ij}\ g_{m_{ij},a_i+a_j}(x),$$

where m_{ij} is a convex linear combination of m_i and m_j:

$$m_{ij} = c_i m_i + c_j m_j \quad c_i = \tfrac{a_i}{a_i+a_j} \quad c_j = \tfrac{a_j}{a_i+a_j}, \tag{2.11}$$

where D is a constant given by

$$D_{ij} = e^{\frac{-a_i a_j}{a_i+a_j}\|m_i-m_j\|^2},$$

and the centres of a new gaussian are still in U, if U is convex. Let $\mathcal{L}$ be the set of all finite linear combinations with real coefficients of elements of $\mathcal{G}$. It is obvious that $\mathcal{L}$ is closed with respect to the sum and multiplication by scalar. The multiplication of two elements of $\mathcal{L}$ is a linear combination of product of gaussians. Such products, as shown above, are gaussians times a scalar and hence $\mathcal{L}$ is closed with respect to multiplication. Thus $\mathcal{L}$ is an algebra of gaussians on U. Further, $\mathcal{G}$ separates points and does not vanish at any point in U. Hence, it follows by the Stone-Weierstrass Theorem that linear combination of gaussians is dense in $C[U]$ [34]. $\diamond\diamond$

2.3.3 Approximation Properties

As was mentioned before, the Stone-Weierstrass Theorem does not provide hints about the characteristics of the solution. From the point of view of approximation theory, the property of approximating continuous functions arbitrary well is not enough to characterise a good approximation scheme. The *best approximation* concept guarantees that the approximation problem has a solution. An approximation scheme has the best approximation property if in the set A of approximating functions (for instance the set of $F(W,x)$ defined as (2.10), which is spanned by the set of parameters defined as $W = \{m_i, c_i, A_i | i = 1, \ldots, N\}$) there is one that has minimum distance from any given function of a larger set Φ.

The set A is called the *existence set* if, for each $f \in \Phi$ there is at least one best approximation to f from A. It is clear that the approximation problem, i.e. given $f \in \Phi$ and $A \subset \Phi$ find a best approximation to f from A, has a solution if and only if A is an existence set. The key property of every existence set is that it is closed. Poggio and Girosi [68] have shown that the set defined by

$$\mathcal{S} = \{f \in C[U] \mid f(x) = \sum_{j=1}^{N} \alpha_j s(\beta_j^T x + \gamma_j), \quad \beta_j, x \in R^n, \quad \alpha_j, \gamma_j \in R\}$$

for $N \geq 2$ is not closed, and therefore is not an existence set. On the other hand, the condition of compactness is sufficient for a set to be an existence set.

When the approximating function is a finite linear combination of basis functions, such as gaussians functions, the set that is spanned by these basis functions is compact, hence, it is an existence set for $C[U]$. Depending on the norm that is chosen in $C[U]$, the best approximating element can be unique. In particular, it has been shown for gaussian networks that the best approximation exists and is unique [68].

As was pointed out in [68] the lack of the best approximation property for the multilayer perceptron is related to possible practical degeneracies of the solution. For example under certain circumstances, when the unit is operating in the saturation zone, a change in the parameters of a sigmoidal unit does not affect the output.

2.4 Spatial Frequency

A nice property of the gaussian function is that its Fourier transform exists and is also gaussian. This result can be obtained applying the following properties of the Fourier transform [86].

If τ_m denotes *translation* by $m \in R^n$ (this means the operator mapping the function $g(x)$ into the function $g(x-m)$) then

$$\mathcal{F}(\tau_m g(x)) = e^{-j\omega^T m}\mathcal{F}(g(x)), \tag{2.12}$$

where $\mathcal{F}(\cdot))$ denotes the $n-$dimensional Fourier transform. If $a > 0$ and δ_a denotes *dilation* by $a \in R$, i.e. the operator δ_a mapping the function $g(x)$ into the function $g(ax)$, then

$$a^n \mathcal{F}(\delta_a g(x)) = G(a^{-1}x) \tag{2.13}$$

where $G(\cdot)$ is the Fourier transform of $g(\cdot)$. The FT of a gaussian function defined by $g(x) = e^{-\|x\|^2}$ is

$$\begin{aligned} \mathcal{F}(g) = \int_{R^n} g(x) e^{-j\omega^T x} dx = \quad & \textstyle\prod_{i=1}^n \int_{-\infty}^{\infty} e^{-x_i^2} e^{-j\omega_i x_i} dx_i \\ & = \textstyle\prod_{i=1}^n \sqrt{\pi} e^{-\frac{\omega_i^2}{4}} = \pi^{\frac{n}{2}} e^{-\frac{\|\omega\|^2}{4}}. \end{aligned} \tag{2.14}$$

Defining g_{σ_i,m_i} as

$$g_{\sigma_i,m_i} = e^{\frac{-\|x-m_i\|^2}{\sigma_i}}, \tag{2.15}$$

and applying the translation and dilation operators

$$G_{\sigma_i,m_i}(\omega) = \mathcal{F}(g_{\sigma_i,m_i}) = \int_{R^n} g_{m_i,\sigma_i}(x) e^{-j\omega^T x} dx$$

gives

$$G_{\sigma_i,m_i}(\omega) = \sigma_i^{\frac{n}{2}} \pi^{\frac{n}{2}} e^{-\frac{\sigma_i}{4}\|\omega\|^2} e^{-j\omega^T m_i},$$

where $\omega \in R^n$ represents the spatial frequency. It is worth noting that the Fourier transform is a basic gaussian function, which has a phase that is linear with respect to m_i, and the gain is given only by σ_i which fixes the characteristic of smoothing. The Fourier transform of $e^{-4\pi\alpha\|x\|^2}$ is called the *Gauss-Weierstrass kernel* .

2.4.1 Regular Gaussian Networks

A *regular* gaussian network is a network with its centres distributed uniformly over the input space forming a grid of units. This kind of network offers simplicity with good approximating capabilities at the expense of more units. Applying spatial frequency concept it is possible to prove the following theorem, which is based on function's kernel [8] [28] [23] and sampling theory [67].

Theorem 4 *Given $\epsilon > 0$ and an L_2 function $F : [0,1]^d \subset R^d \rightarrow R$, and N points (measurements) y_i from the system characterized by (2.16)*

$$y_i = F(x_i) + z_i, \tag{2.16}$$

where x_i are equidistant points, z_i is a noise with the following properties: $var(z_i) = \varsigma$ and $E(z_i) = 0$. Then there exists a regular gaussian network that can approximate F to within ϵ mean square error accuracy.

Proof:

Consider the $d-$dimensional space $Q_d = \{x \in R^d | x \in [0,1]^d\}$, and let $N = n^d$,n an integer and $k = 1, \ldots, d$ and $i = 1, \ldots, N$. Partition the unit interval on the kth axis into $n-1$ equal subintervals of length $\Delta(x^k)$ each, defining the volume of an elementary cube as

$$V(Q_d) = \Delta(x^1)\Delta(x^2)\ldots\Delta(x^d), \tag{2.17}$$

and $\hat{F}$ as the representation of F by means of a gaussian network defined by

$$\hat{F}(x) = \sum_{i=1}^{N} y_i V(Q_d) g^n{}_{m_i,\sigma}(x), \tag{2.18}$$

where $g^n{}_{m_i,\sigma}(x)$ is the normalised gaussian defined by $g^n{}_{m_i,\sigma}(x) = \sigma^{\frac{-d}{2}} \pi^{\frac{-d}{2}} g_{m_1,\sigma}(x)$, such that

$$\int_{R^d} g^n{}_{m_i,\sigma}(x) dx = 1,$$

and the centres are located in each x_i, i.e. $m_i = x_i$, $i = 1, \ldots, N$. Replace the index i, in m_i, with l representing a $d-$tuple $[l] = \{l_1, \ldots, l_d\}$ to emphasise the dependence of the value of m_i on the position of the *ith* unit in the grid. The same change of index applies to y_i. This can be expressed in terms of the canonical basis vector e for R^d as $m_l = l_1 e_1 \Delta(x^1) + \ldots + l_d e_d \Delta(x^d)$, where l belongs to $[l_{Q_d}] = \{l \in Z^d | m_i \in Q_d\}$. Then equation (2.18) can be written as

$$\hat{F}(x) = \sum_{[l_{Q_d}]} y_l V(Q_d) g^n{}_{m_l,\sigma}(x),$$

where $\sum_{[l_{Q_d}]}$ means a multiple sum over the multiple index defined by the $d-$tuple $[l]$.

The mean squared error can be expressed as $(variance) + (bias)^2$ [27],

$$E(\hat{F} - F)^2 = var(\hat{F}) + b^2(\hat{F}), \tag{2.19}$$

where b is

$$b(\hat{F}) = E(\hat{F}) - F. \tag{2.20}$$

Since $V(Q_d)$ is constant

$$var(\hat{F}) = \varsigma^2 V^2(Q_d) \sum_{[l_{Q_d}]} ({g^n}_{m_l,\sigma}(x))^2, \tag{2.21}$$

and

$$E(\hat{F}) = \sum_{[l_{Q_d}]} F(m_l)V(Q_d){g^n}_{m_l,\sigma}(x), \tag{2.22}$$

or

$$E(\hat{F}) = (2\pi)^{-d} \int_{-\infty}^{\infty} (\sum_{[l_{Q_d}]} F(m_l)V(Q_d)e^{-j\omega^T m_l})e^{j\omega^T x}G(\omega\sqrt{\sigma})d\omega, \tag{2.23}$$

where $G(\omega\sqrt{\sigma})$ is the Fourier transform of $g_{0,1}(\frac{x}{\sqrt{\sigma}})$. and $\phi_F(\omega)$ is the Fourier transform of $F(x)$. Defining a window function $W(x)$ such that

$$\phi_F(\omega) = \int_{R^d} e^{-j\omega^T x} F(x)W(x)dx = \int_{Q_d} e^{-j\omega^T x} F(x)dx, \tag{2.24}$$

to account for the indefiniteness of F outside Q_d. Then the bias b can be expressed as

$$b = (2\pi)^{-d} \int_{-\infty}^{\infty} \left[[\sum_{[l_{Q_d}]} F(m_l)V(Q_d)e^{-j\omega^T m_l}]G(\omega\sqrt{\sigma}) - \phi_F(\omega) \right] e^{j\omega^T x}d\omega. \tag{2.25}$$

Noting that

$$G(\omega\sqrt{\sigma}) \sum_{[l_{Q_d}]} F(m_l)V(Q_d)e^{-j\omega^T m_l} \tag{2.26}$$

can be considered as an approximation to $\phi_F(\omega)$, and using the Poisson's formula, $\sum_{[l]} F(m_l)V(Q_d)e^{-j\omega^T m_l}W(m_l) = \sum_{[k]} \phi_F(\omega + n_k)$, where n_k is a vector defined by $m_l^T n_k = 2\pi$ if $l = k$, otherwise $m_l^T n_k = 0$, it follows that

$$|[G(\omega\sqrt{\sigma}) \sum_{[k]} \phi_F(\omega + n_k)] - \phi_F(\omega)| \leq \alpha. \tag{2.27}$$

replacing b in (2.19) by (2.25) we get

$$E(\hat{F}-F)^2 \leq \varsigma^2 V^2(Q_d) \sum_{[l_{Q_d}]} ({g^n}_{m_l,\sigma}(x))^2 + \alpha^2 \tag{2.28}$$

and finally

$$E(\hat{F}-F)^2 \leq \epsilon.$$

$\diamond\diamond$

Comments:

- This proof depends on certain regularity conditions, like the same values of σ and volume $V(Q_d)$ for all the units. Extension to more general case can be done.
- α, see inequality (2.27), takes into account the effect of a finite number of units, the lack of compact support, and filtering characteristic of the gaussian units. Further implications are analysed in Chapter 3.
- From this analysis it can be seen that the coefficients are related to different properties of the approximation, for example the width of the gaussian is related to the bandwidth of the function, the linear coefficients to the output of the functions, and the centres to a grid, which defines a sampling structure over the input space to recover the functions.

2.5 Other Networks Using Gaussians

Another useful function is defined by

$$gl_i(x) = (b_i + c_i^T x)^l g_{m_i,a_i}(x),$$

where $l \in Z, a_i > 0, m_i \in U, x \in U$, and $b_i \in R$ and $c_i \in R^n$ are such that $b_i + c_i^T x \neq 0$.

The central idea of this kind of function is try to generate a window and then approximate the function within the window with a polynomial expression.

Theorem 5 *Let $U \subset R^n$ be a convex compact set. Let the family $\mathcal{G}l$ of functions corresponding to gaussian functions defined by*

$$gl_{b_i,c_i,m_i,a_i} : U \to R.$$

Then a finite linear combination with real coefficients of elements of $\mathcal{G}l$ is dense in $C[U]$.

Proof:

Let $\mathcal{L}$ be the set of all finite linear combinations with real coefficients of elements of $\mathcal{G}l$. Multiplying two elements of $\mathcal{G}l$ one obtains an element of $\mathcal{L}$ as,

$$gl_{b_i,c_i,m_i,a_i}(x) gl_{b_j,c_j,m_j,a_j}(x) = (b_i + c_i^T x)_i^l (b_j + c_j^T x)_j^l \; D_{ij} \; g_{m_{ij},a_i+a_j}(x),$$

where m_{ij} is a convex linear combination of m_i and m_j, D_{ij} a constant, and the centres of the new gaussians are still in U, if U is convex. The right-hand side of the equation can be represented as

$$(b_i + c_i^T x)_i^l (b_j + c_j^T x)_j^l D_{ij} g_{m_{ij},a_i+a_j}(x) = \sum_{k=0}^{l_i+l_j} (b_k^* + c_k^{*T} x)^k D_{ij} g_{m_{ij},a_i+a_j}(x).$$

It is obvious that $\mathcal{L}$ is closed with respect to the sum and multiplication by scalar. The multiplication of two elements of $\mathcal{G}l$ is a linear combinations of the product of elements of $\mathcal{G}l$. Such products, as shown above, are elements of $\mathcal{G}l$ times a scalar, and hence $\mathcal{L}$ is closed under multiplication. Thus, $\mathcal{L}$ is an algebra on U, as $\mathcal{L}$ is closed with respect to multiplication. Furthermore, $\mathcal{G}l$ separates points and does not vanish at any point in U. Hence it follows by the Stone-Weierstrass Theorem that linear combination of function $\mathcal{G}l$ is dense in $C[U]$. $\diamond\diamond$

The above theorem is a special case of the one stated in [34]. This representation is related to the Hermite Theory [49]. In fact, gaussian and its derivatives belong to $\mathcal{G}l$. Using this, it is possible to find a closed form for the coefficients of the expansion as explained in [49]. It has been shown that in practice the expansion requires a few terms of the polynomial expression in order to approximate a function due to the fact that the frequency peaks for high orders move very close together, so that successive terms give very little additional information.

Other approximating schemes such as CMAC and splines are encompassed by the gaussian networks and can be included in a general model of memory as described in [81].

In statistics the properties of additive models such as the one described in the previous sections have been investigated by several authors [18], [32]. The difference between gaussian and sigmoids from a statistical point of view is that the former is related to *kernel* functions (this characteristic was fully explored in the previous section) and sigmoid is related to the *projection* based schemes [18]. In statistics it is known that the latter representation can reduce the "curse of dimensionality" [18]. This refers to the tendency for nonparametric regression procedure to perform very badly when the sample data is limited, due to the fact the high dimensional space is mostly empty.

2.6 Conclusions

The basic idea of approximation using structures derived from Artificial Neural Networks was described, and the main theorems demonstrating the possibility of reaching certain approximation were stated and proved for two cases. Further, connections with other fields were pointed out and possible extensions analysed. The approximation of an unknown function can be accomplished by different architectures, but the theorems used here, which are based on algebra of functions, do not give the characteristics of the final solution and how to get it. In the next chapter a description of how to obtain solutions to the approximation problem is given.

Chapter 3

Learning Algorithms

SUMMARY

This Chapter describes some learning algorithms to train different networks. We start by analysing the multilayer perceptron and designing a new training algorithm for it. It is closely related to the stochastic approximation version, but uses a forgetting factor; the performance of this algorithm is shown by several simulations. For gaussian networks several algorithms are compared. The spatial frequency concept is used to solve completely (at least for low dimensions) the training problem through the changing format approach. A general least square problem is analysed and the decomposition of the training problem for gaussian networks into smaller subproblems is pursued with a multi-algorithm approach. In addition, some guidelines to train regular gaussian networks are given. Finally, a comparison among the different algorithms is done.

3.1 Introduction

Learning is defined in behavioural terms as an increase in performance index over a specified time interval. In terms of designing a control system, the problem of learning may be viewed as the problem of estimation or successive approximation of unknown quantities of a functional which represents the controlled process under study [21]. From this perspective, this problem is related from the engineering viewpoint to system identification problems and techniques [48], and from the statistical viewpoint with the Theory of Generalised Additive Models [32].

The problem can be stated as the approximation of a multivariable function $f(X)$ by an approximation function $F(\theta, X)$ having a finite number of parameters θ. For the choice of a specific architecture and units of a connectionist representation, which means a specific F, the goal is to find the best (in some sense) set of parameters such that F approximates f on the set of "training data". The training set contains n_t input-output values representing the function f. Formally the approximation problem is stated as:

Approximation Problem:

If $f(X)$ is a continuous function defined on a set $X \subset R^n$,and $F(\theta, X)$ is an approximation function which depends continuously on $\theta \in P \subset R^p$ and X, the approximation problem is to determine the parameters θ^*, such that

$$\rho[F(\theta^*, X), f(X)] \leq \rho[F(\theta, X), f(X)] \tag{3.1}$$

for all $\theta \in P$. ρ represents the distance function which is induced by a norm, i.e. the distance between f and F is induced by the L^p norm defined as

$$\|f - F\|_p = \left\{\int_C |f - F(\theta, X)|^p dX\right\}^{\frac{1}{p}},$$

where f is the unknown function to be approximated. In this work the standard L_2 norm is used □

The calculation of the norm is based on the training set, so for practical purposes equation (3.1) can be expressed by

$$\sum_{l=1}^{n_t}[F(\theta^*, X_l) - f(X_l)]^2 \leq \sum_{l=1}^{n_t}[F(\theta, X_l) - f(X_l)]^2,$$

where l represents an index over the training set. From an engineering point of view, the problem can be relaxed such that

$$\sum_{l=1}^{n_t}[F(\theta, X_l) - f(X_l)]^2 \leq \zeta, \tag{3.2}$$

where $\zeta \in R$ is a known constant. This means that θ^* is not of interest but instead any $\theta \in S$ suffices, where $S \subset P$ represents the set of θs which satisfy (3.2). In practice is not necessary to get always the best solution and a suboptimal solution can be used instead.

The solution to the problem is approached from an engineering, rather than a physiological, perspective. In particular, no claim of physiological relevance is made nor the solution is constrained by the conventional loosely coupled parallel distributed processing architecture. The aim is to find a good learning algorithm; the appropriate architecture for implementation is a subject for further investigation.

For each particular problem, the network must be trained; that is, the weights governing the strengths of the connections between units must be varied in order to get the target output corresponding to the input presented.

3.2 Learning Algorithms for Nets Using Sigmoids

The most popular method for training networks using sigmoids is 'Back Propagation' (BP) [53], but this training method requires a large number of iterations before the network generates a satisfactory approximation to the target mapping. Improved rates of convergence arise from heuristics to vary certain parameters of the basic algorithm, the main problem with the last approach is the difficulty to prove correctness and completeness of the heuristic set.

The problem is viewed within a standard framework for the optimisation of nonlinear systems: the "Stochastic Approximation" (SA) algorithm of Albert and

Gardner [3][2].

The method based on SA is related to the celebrated BP algorithm [53]; and this relationship will be explored in this work. However, the simulations presented in this work indicate that our method converges to a satisfactory solution much faster than the BP algorithm [53]. More simulations can be found in [79] [26].

As Minsky and Papert [56] have pointed out, the BP algorithm is a hill climbing algorithm with all the problems implied by such methods. Our method can also be viewed as a hill climbing algorithm; but it is more sophisticated than the standard BP method. In particular, unlike the BP method, the cost function approximately minimised by our algorithm is based on past as well as current data.

Perhaps even more importantly, our method provides a bridge between Artificial Neural Network approaches and well developed techniques arising from control and systems theory. One consequence of this is that the powerful analytical techniques associated with control and systems theory can be brought to bear on Artificial Neural Network algorithms. Some initial ideas are sketched out in section 3.2.4.

3.2.1 Backpropagation

The general index to be minimised is the sum of the squared error over the training patterns, which is defined as follows

$$J = \frac{1}{2}\sum_{t=1}^{n_t} e_t^2,$$

where n_t is the number of training points, $e_t = (Y_t - y_N)$, and y_N is the output of the network at presentation of the input data t, To minimise this index the gradient algorithm is proposed

$$W_i(k) = W_i(k-1) - \alpha\frac{\partial J}{\partial W_i(k-1)}, \tag{3.3}$$

where k is the index for the algorithm iterations, α is the learning rate, and the sensitivity of J with respect to $W_i(k-1)$ is given by

$$\frac{\partial J}{\partial W_i(k-1)} = \frac{1}{2}\sum_{t=1}^{n_t} \frac{\partial e_t^2}{\partial W_i(k-1)}. \tag{3.4}$$

The calculations of $\frac{\partial e_t^2}{\partial W_i}$ can be done in several steps, unfolding a certain structure for the calculations themselves.

As a first step, define the *incremental error* matrix E_i relating the ith layer to the error evaluated for a given set of weights and net inputs

$$E_i = \frac{\partial y_N}{\partial \breve{x}_i} e_t. \tag{3.5}$$

Using the chain rule, it follows that

$$E_i = E_{i+1} \frac{\partial \breve{x}_{i+1}}{\partial y_i} \frac{\partial y_i}{\partial \breve{x}_i}, \tag{3.6}$$

and using equations (2.1) and (2.3)

$$E_i = E_{i+1} \bar{f}'(y_i) \breve{W}_i^T. \tag{3.7}$$

Equation (3.7) will be called the *error backpropagation algorithm.* By definition

$$E_N = 2\frac{\partial y_N}{\partial \breve{x}_N} e_t. \tag{3.8}$$

Applying the chain rule once more

$$\frac{\partial e_t^2}{\partial W_i} = \frac{\partial y_N}{\partial \breve{x}_i} \frac{\partial \breve{x}_i}{\partial W_i}. \tag{3.9}$$

Substituting from equation (3.5), equation (3.9) becomes

$$\frac{\partial e_t^2}{\partial W_i} = E_i \frac{\partial \breve{x}_i}{\partial W_i}. \tag{3.10}$$

To calculate an update $\frac{\partial e_t^2}{\partial W_i}$ these equations must be solved backwards, which gives the name to the algorithm. Having computed $\frac{\partial e_t^2}{W_i}$, we get $\frac{\partial J}{\partial W_i}$ by using equation (3.4).

There are several heuristic procedures commonly used to increase the speed of convergence of the algorithm:

- *Momentum:* This extra feature offers the possibility of increasing the learning rate without producing oscillation [53], a typical value used for β is 0.9, and the rule is modified in the following way

$$W_i(k) = W_i(k-1) - \alpha \frac{\partial J}{\partial W_i(k-1)} + \beta(W_i(k-1) - W_i(k-2)). \quad (3.11)$$

  In this case the parameters are updated in each presentation of a training value.

- *Adaptive learning rate:* The learning rate is varied according to whether or not an iteration decreases the performance index (the total error for all training patterns)[88], [6]. If a step results in reduced total error then the learning rate is increased. If an update produces an index greater than the previous one, all the changes to the weights are rejected, and the learning rate is decreased, the momentum term is set to zero, and the step is repeated. When a successful step is then taken, the momentum is set to its original value.

The summary of the momentum algorithm is as follows

1. Set the elements of the weight matrices W_i to small random values.
2. Set α and β.

At each time step:

1. Present an input $x_1 = X_t$.
2. Propagate the input *forwards* through the network using equations (2.1) and (2.3) to give x_i and y_i for each layer.
3. Given x_i for each layer, propagate the incremental error matrix E_i *backwards* using the gain back propagation of equations (3.7) and (3.8).
4. Compute the partial derivatives $\frac{\partial J}{\partial W_i}$ using equations (3.10) and (3.4).

5. Compute the output error e_t from equation

$$e_t = (Y_t - y_N). \tag{3.12}$$

6. Update the parameter vector W using equation (3.11).

3.2.2 The Stochastic Approximation Algorithm

Parametric Linearisation

The stochastic approximation technique deals with the general nonlinear system of the form

$$Y_t = F(\theta, X_t), \tag{3.13}$$

where θ is the parameter vector ($n_p \times 1$) containing the n_p system *parameters*, X_t is the input vector ($n_i \times 1$) containing the n_i system *inputs* at (integer) time t. Y_t is the system output vector at time t. $F(\theta, X_t)$ is a nonlinear, but differentiable, function of the two arguments θ and X_t.

In the context of the layered neural networks discussed in this work, Y_t is the network output and X_t the input (training pattern) at time t and θ contains the network *weights*. Following Albert and Gardner [3], the first step in the derivation of the learning algorithm is to find a local linearisation of equation (3.13) about a nominal parameter vector θ^0.

Expanding F around θ^0 in a first order Taylor series gives

$$F(\theta^0 + \tilde{\theta}, X_t) = F(\theta^0, X_t) + F'(\theta^0, X_t)\tilde{\theta} + \epsilon, \tag{3.14}$$

where

$$F' = \frac{\partial F}{\partial \theta}, \tag{3.15}$$

$\tilde{\theta} = \theta - \theta^0$, and ϵ is the approximation error representing the higher terms in the series expansion. Defining

$$\tilde{Y}_t = F(\theta^0 + \tilde{\theta}, X_t) - F(\theta^0, X_t) + F'(\theta^0, X_t)\theta^0 \tag{3.16}$$

and

$$\tilde{X}_t = F'(\theta^0, X_t)^T \tag{3.17}$$

then equation (3.14) becomes

$$\tilde{Y}_t = \tilde{X}_t^T \theta + \epsilon. \tag{3.18}$$

This equation forms the desired linear approximation of the function $F(\theta, X_t)$ witn ϵ representing the approximation error, and forms the basis of a least-squares type algorithm to estimate θ.

Least-squares and the Pseudoinverse.

This section describes an application of the standard non-recursive and recursive least squares algorithm to the estimation of θ in the linearised equation (3.18). There is however, one modification: a *pseudoinverse* is used to avoid difficulties with a non-unique optimal estimate for the parameters, which manifests itself as a singular data dependent-matrix [2].

The standard least square cost function with exponential discounting of data is

$$V_T(\theta) = \frac{1}{T} \sum_{t=1}^{T} \lambda^{T-t} [\tilde{Y}_t - \tilde{X}_t^T \theta]^2, \tag{3.19}$$

where the *exponential forgetting factor* λ gives different weights to different observations and T is the number of presentations of the different training sets.

Using standard manipulations, the value of the parameter vector $\hat{\theta}_T$ minimising the cost function is obtained from the linear algebraic equation

$$S_T \hat{\theta}_T = \sum_{t=1}^{T} \lambda^{T-t} \tilde{X}_t \tilde{Y}_t, \tag{3.20}$$

where the matrix S_T $(n_p \times n_p)$ is given by

$$S_T = \sum_{t=1}^{T} \lambda^{T-t} \tilde{X}_t \tilde{X}_t^T. \tag{3.21}$$

When applied to the layered neural networks discused in this work, S_T will usually be singular (or at least nearly singular), thus the estimate will be non-unique. This non-uniqueness is of no consequence when computing the network output, but it is vital to take it into account in the computation of the estimates.

Here the minimum norm solution for $\hat{\theta}_T$ is chosen as:

$$\hat{\theta}_T = S_T^+ \sum_{t=1}^{T} \lambda^{T-t} \tilde{X}_t \tilde{Y}_t \tag{3.22}$$

where S_T^+ is a pseudoinverse of S_T .

In practice, a *recursive* form of (3.22) is required. Using standard manipulations

$$\hat{\theta}_{t+1} = \hat{\theta}_t + S_t^+ \tilde{X}_t e_t, \tag{3.23}$$

where the *error* e_t is given by

$$e_t = \tilde{Y}_t - \hat{\theta}_t^T \tilde{X}_t, \tag{3.24}$$

but, using (3.16) and (3.17),

$$\tilde{Y}_t - \hat{\theta}_t^T \tilde{X}_t \approx Y_t - F(\hat{\theta}_t, X_t) \tag{3.25}$$

and finally (3.23) becomes

$$\hat{\theta}_{t+1} = \hat{\theta}_t + S_t^+ \tilde{X}_t (Y_t - F(\hat{\theta}_t, X_t)). \tag{3.26}$$

S_t can be recursively updated according to

$$S_t = \lambda S_{t-1} + \tilde{X}_t \tilde{X}_t^T. \tag{3.27}$$

Indeed, S_t^+ itself can be updated directly[2], but the details are not pursued further here.

Data is discarded at an exponential rate with time constant τ given by

$$\tau = \frac{1}{1-\lambda}, \tag{3.28}$$

where τ is in units of samples. This feature is necessary to discount old information corresponding to estimates far from the convergence point and thus inappropriate to the linearised model about the desired nominal parameter vector θ^0.

The Multilayer Perceptron

To apply the algorithm given by equation (3.26), however, it is necessary to find an expression for $\tilde{X}_t = F'(\theta^0, X_t)$ as in equation (3.18). Because of the simple recursive feedforward structure of the multi-layer perceptron, it is possible to obtain a simple recursive algorithm for the computation of the elements of $\tilde{X}_t$. This algorithm is, not surprisingly, related to the BP algorithm.

As a first step, define the *incremental gain* matrix G_i relating the ith layer to the net output evaluated for a given set of weights and net inputs.

$$G_i = \frac{\partial x_N}{\partial \breve{x}_i}. \tag{3.29}$$

Using the chain rule, it follows that

$$G_i = G_{i+1} \frac{\partial \breve{x}_{i+1}}{\partial y_i} \frac{\partial y_i}{\partial \breve{x}_i} \tag{3.30}$$

and using equations (2.1) and (2.3)

$$G_i = G_{i+1} \bar{f}'(y_i) \breve{W}_i^T. \tag{3.31}$$

Equation (3.31) will be called the *gain backpropagation algorithm* or GBP. By definition

$$G_N = \frac{\partial \breve{x}_N}{\partial \breve{x}_N} = I_{n_N}, \tag{3.32}$$

where I_{n_N} is the $n_N \times n_N$ unit matrix.

Applying the chain rule again

$$\frac{\partial x_N}{\partial W_i} = \frac{\partial x_N}{\partial \breve{x}_{i+1}} \frac{\partial \breve{x}_{i+1}}{\partial W_i}. \tag{3.33}$$

Substituting from equations (2.1), (2.3) and (3.29), equation (3.33) becomes

$$\frac{\partial x_N}{\partial W_i} = G_{i+1} \frac{\partial \breve{x}_{i+1}}{\partial W_i}. \tag{3.34}$$

The GBP algorithm

Initially:

1. Set the elements of the weight matrices W_i to small random values and load the corresponding elements into the column vector $\hat{\theta}_0$.

2. Set λ between .95 and .99 and the matrix S_0 to $s_0 I$ where I is a unit matrix of appropriate dimension and s_0 a small real number.

At each time step:

1. Present an input $x_1 = X_t$

2. Propagate the input *forwards* through the network using equations (2.1) and (2.3) to give x_i and y_i for each layer.

3. Given x_i for each layer, propagate the incremental gain matrix G_i *backwards* using the gain back propagation of equations (3.31) and (3.32).

4. Compute the partial derivatives $\frac{\partial x_N}{\partial W_i}$ using equation (3.34) and load the corresponding elements into the column vector $\tilde{X}$.

5. Update the matrix S_t using (3.27) and find its pseudoinverse S_t^+ (or update S_t^+ directly [2]).

6. Compute the output error e_t from equation (3.24).

7. Update the parameter vector $\hat{\theta}_t$ using equation (3.26).

8. Reconstruct the weight matrices W_i from the parameter vector $\hat{\theta}_t$.

3.2.3 Simulations

The simple multi-layer perceptrons discussed in section 3.2.2 were implemented in Matlab [57] and a number of simulation experiments were performed to evaluate the proposed algorithm and compare its performance with the conventional BP algorithm.

The XOR Problem

To illustrate the method the XOR problem was considered. The architecture for solving the XOR problem with two hidden units and no direct connections from inputs to output is shown in Figure 3.1.

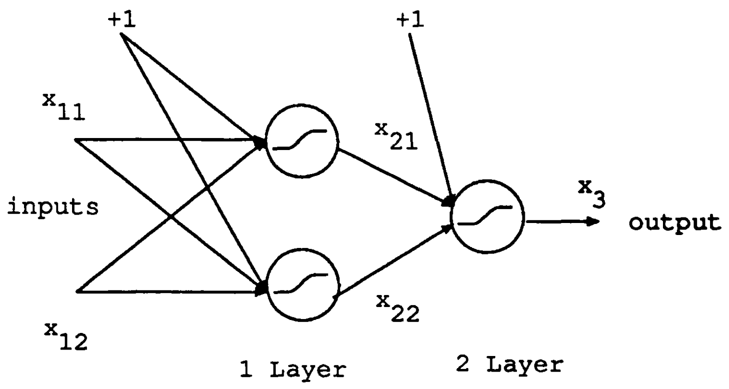


Figure 3.1: Architecture to solve the XOR problem.

In this case, the weight and input matrices for layer 1 are of the form

$$W_1 = \begin{pmatrix} w_{111} & w_{112} \\ w_{121} & w_{122} \\ w_{131} & w_{132} \end{pmatrix}, x_1 = \begin{pmatrix} x_{11} \\ x_{12} \\ 1 \end{pmatrix}. \tag{3.35}$$

The last row of the matrix in equation (3.35) corresponds to the offset terms. The weight and input matrices for layer 2 are of the form

$$W_2 = \begin{pmatrix} w_{211} \\ w_{221} \\ w_{231} \end{pmatrix}, x_2 = \begin{pmatrix} x_{21} \\ x_{22} \\ 1 \end{pmatrix}. \tag{3.36}$$

Once again, the last row of the matrix in equation (3.36) corresponds to the offset term.

Applying the GBP algorithm (3.31) gives

$$G_2 = \bar{f}'(y_2)\check{W}_2^T = \bar{f}'(y_2)\begin{pmatrix} w_{211} \\ w_{221} \end{pmatrix}^T. \tag{3.37}$$

Hence, using (3.34) and (3.32),

$$\frac{\partial x_3}{\partial W_2} = x_2\bar{f}'(y_2) = x_2x_3(1-x_3) \tag{3.38}$$

and using (3.34) and (3.37)

$$\frac{\partial x_3}{\partial W_1} = G_2 \frac{\partial \breve{x}_2}{\partial W_1}. \tag{3.39}$$

Thus, in this case, the terms $\tilde{X}$ and θ linearised equation (3.18) are given by

$$\tilde{X} = \begin{pmatrix} x_{11}x_{21}(1-x_{21})x_3(1-x_3)w_{211} \\ x_{12}x_{21}(1-x_{21})x_3(1-x_3)w_{211} \\ x_{21}(1-x_{21})x_3(1-x_3)w_{211} \\ x_{11}x_{22}(1-x_{22})x_3(1-x_3)w_{221} \\ x_{12}x_{22}(1-x_{22})x_3(1-x_3)w_{221} \\ x_{22}(1-x_{22})x_3(1-x_3)w_{221} \\ x_{21}x_3(1-x_3) \\ x_{21}x_3(1-x_3) \\ x_3(1-x_3) \end{pmatrix}, \theta = \begin{pmatrix} w_{111} \\ w_{121} \\ w_{131} \\ w_{112} \\ w_{122} \\ w_{132} \\ w_{211} \\ w_{221} \\ w_{231} \end{pmatrix}. \tag{3.40}$$

The training patterns used are shown in table 3.1

x_1	x_2	y
0	0	0.9
1	0	0.1
0	1	0.1
1	1	0.9

Table 3.1: Patterns for the XOR problem.

The conventional backpropagation (BP) and GBP algorithms were simulated. Figure 3.2 shows the performance of the BP algorithm and Figure 3.3 shows the performance of the GBP algorithm.

The reduction in the number of iterations is four fold for the same error.

The effect of not discounting past data is shown in Figure 3.4, where the forgetting factor was equal to 1.

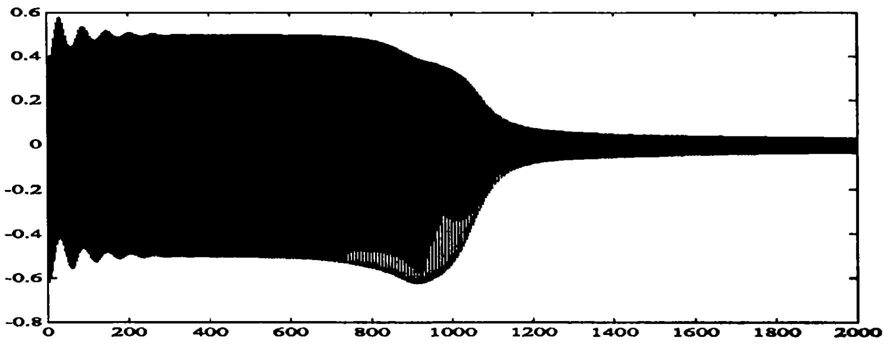


Figure 3.2: Error for the BP algorithm.

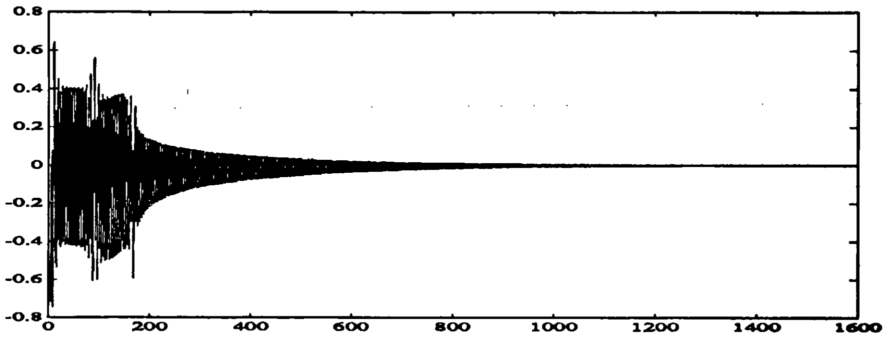


Figure 3.3: Error for the GBP algorithm.

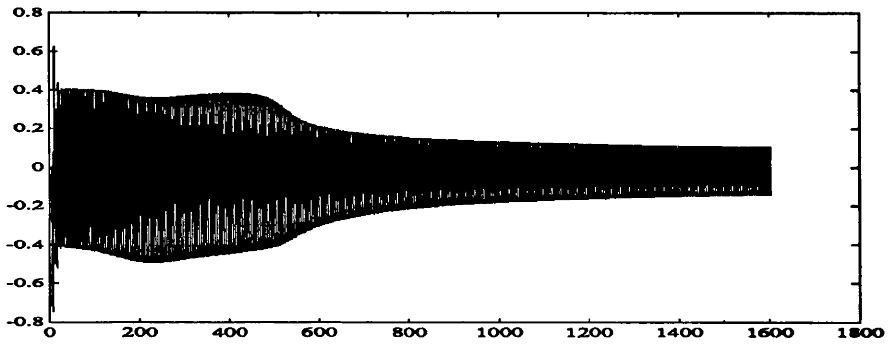


Figure 3.4: Error for the GBP algorithm , with forgetting factor λ=1.

Discontinuous Function Approximation

This example illustrates the performance of the algorithm approximating a discontinuous function defined by

$$Y(l) = f(X_1(l), X_2(l)) = \begin{cases} 1.0 & , X_1(l) > 5 \text{ or } \mathrm{X}_2(1) > 5 \\ .2 & , \text{otherwise,} \end{cases} \tag{3.41}$$

where X_i, $i = 1, 2$ varies in the interval $[0, 10]$. The architecture is the same as the one used in the XOR problem. The training set consists of 25 patterns with inputs values distributed uniformly over the input space. The evolution of the squared error is shown in Figures 3.5 and 3.6. The difference in the convergence time is considerable. This is due to appropriate setting of the initial conditions for the network.

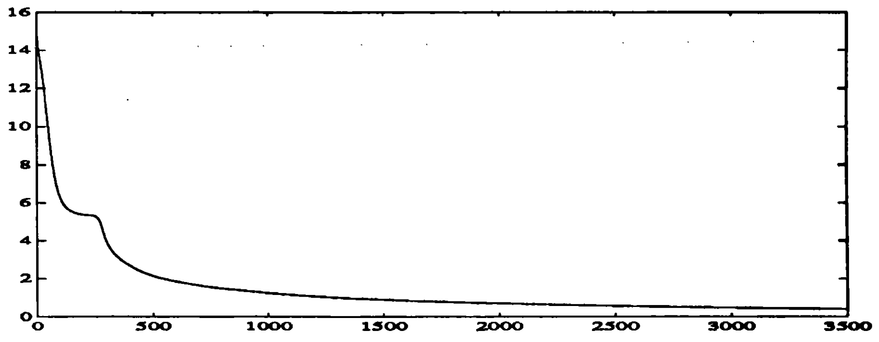


Figure 3.5: Square Error for the BP Algorithm.

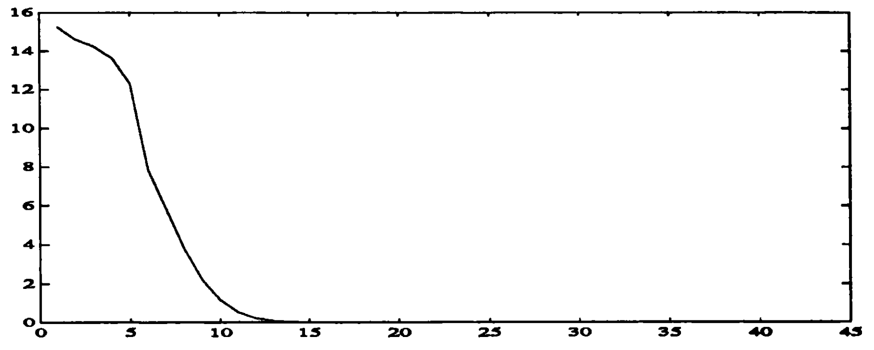


Figure 3.6: Square Error for the GBP Algorithm.

The characteristic of the final solution can be seen in Figure 3.7, in this case is obvious that the system can approximate this kind of surface just with a very limited number of units.

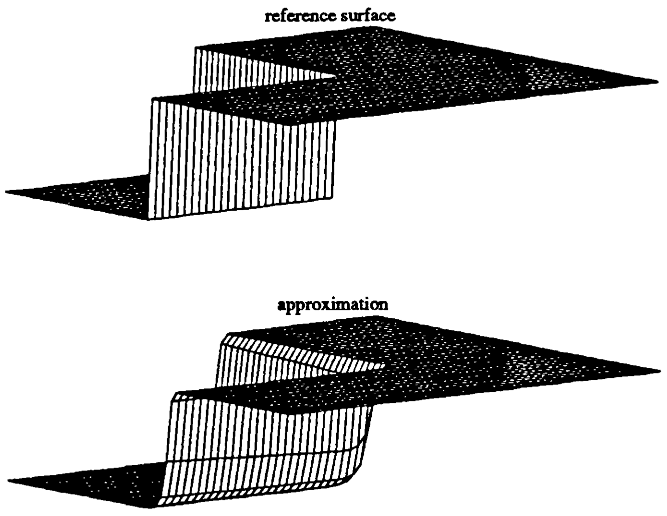


Figure 3.7: Reference surface and final approximation.

Transformation of Coordinates

The Cartesian endpoint position of a rigid two-link manipulator (Figure 3.8) is given by

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \tag{3.42}$$

$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \tag{3.43}$$

where $[x, y]$ is the position in the plane, $L_1 = 0.3m$, $L_2 = 0.2m$ are the lengths of the links. The joint angles are θ_1, θ_2.

Equations (3.42) and (3.43) show that it is possible to produce this kind of transformation using the structure shown in Figure 3.9.

The linearisation is similar to that of the XOR network except that W_1 is 2×10, W_2 is 10×1 and input vector x_1 is 10×1.

The connectionist representation was trained to do the coordinate transformation in the following range of θ_1 =[0, π] and θ_2=[0, π], from this region a grid of 33 training values were used. The evolution of the square error (measured in

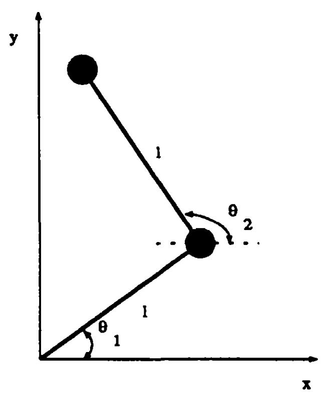


Figure 3.8: Two-link manipulator in cartesian coordinates.

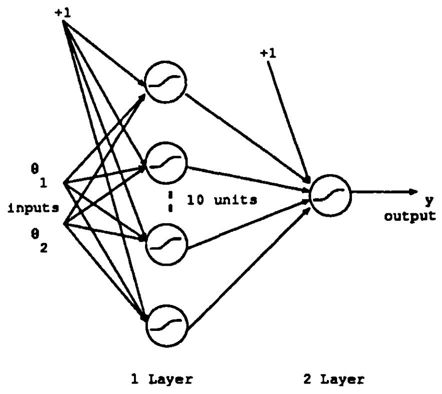


Figure 3.9: Architecture to solve the coordinate transformation problem.

m) using BP and GBP is shown in Figures 3.10 and 3.11. The GBP algorithm converges faster than the BP algorithm and reaches a lower minimum.

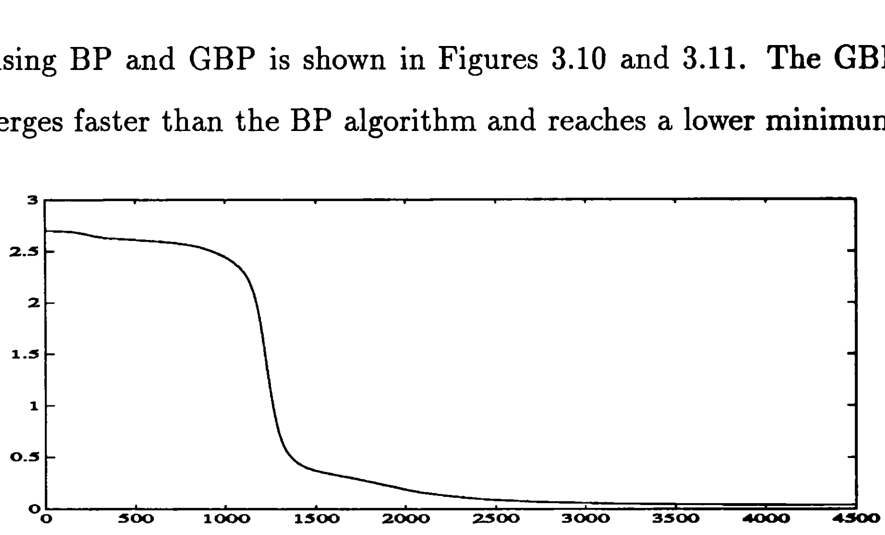


Figure 3.10: Square Error for the BP algorithm.

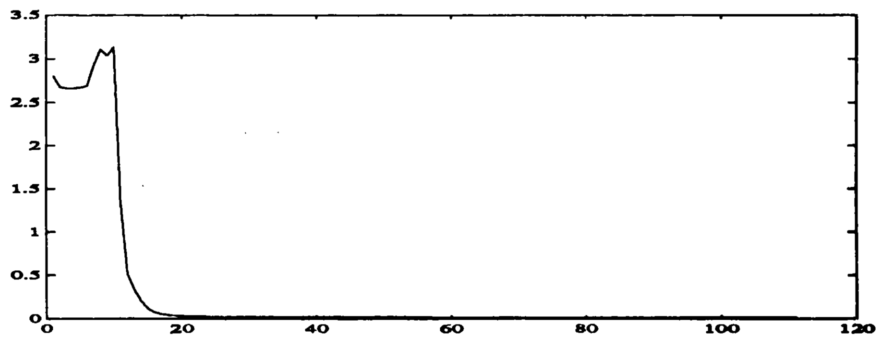


Figure 3.11: Square Error for the GBP algorithm.

Table 3.3 shows the number of presentations required for both algorithms to reach a sum square error of 0.02.

Summary of Simulation Results

The values of the parameters used in each problem are shown in Table 3.2

problem	gain(BP)	momentum(BP)	λ (GBP)
XOR	.5	.9	.95
Disc. Trans.	.05	0	.95
C. Transform	.02	.95	.99

Table 3.2: Parameters used in the simulations.

The results for the four sets of experiments appear in Table 3.3.

problem	N(BP)	N(GBP)
XOR	300	70
D. Transform	2,520	15
C. Transform	3,450	24

Table 3.3: Simulation results: number of presentation needed for convergence.

3.2.4 Comments About Convergence

Following the derivations of Ljung [47], if there exists a θ^0 such that

$$e_t = Y_t - F(\theta^0, X_t) = \epsilon_t, \tag{3.44}$$

where ϵ_t is a white noise, and introducing

$$\tilde{\theta}_t = \hat{\theta}_t - \theta^0 \tag{3.45}$$

in equation (3.23), the following are obtained

$$\begin{aligned} S_t\tilde{\theta}_{t+1} &= S_t\tilde{\theta}_t + \tilde{X}_t e_t \\ &= \lambda S_{t-1}\tilde{\theta}_t + \tilde{X}_t\tilde{X}_t^T\tilde{\theta}_t + \tilde{X}_t e_t \end{aligned} \tag{3.46}$$

Using the expression

$$S_t = \sum_j \beta(j,t)\tilde{X}_j\tilde{X}_j^T \tag{3.47}$$

and setting

$$\lambda^{t-j} = \beta(j,t) \tag{3.48}$$

(3.46) can be transformed to

$$S_t\tilde{\theta}_{t+1} = \beta(0,t)S_0\tilde{\theta}_0 + \sum_j \beta(j,t)\tilde{X}_j[\tilde{X}_j^T\tilde{\theta}_j + e_j]. \tag{3.49}$$

In general

$$e_j = Y_j - F(\theta_j, X_j), \tag{3.50}$$

but its linear approximation around θ^0 is

$$e_j \approx \epsilon_j - \tilde{X}_j^T\tilde{\theta}_j. \tag{3.51}$$

Hence, the equation (3.49) becomes

$$S_t\tilde{\theta}_{t+1} = \sum_j \beta(j,t)\tilde{X}_j\epsilon_j, \tag{3.52}$$

$$\tilde{\theta}_{t+1} = S_t^+ \sum_j \beta(j,t)\tilde{X}_j \epsilon_j. \tag{3.53}$$

Then $\tilde{\theta}_t \to 0$ as $t \to \infty$, representing a global minimum if

C1. S_t has at least n_p nonzero eigenvalues ;

C2. $\sum_{k=1}^{\infty} \|\tilde{X}_k\| < \infty$ and some of the following conditions are met:

C3. the input sequence X_t is such that $\tilde{X}_t$ is independent of ϵ_j and

C4. (a) ϵ_j is a white noise and is independent on what has happened up to the time $n-1$ or

(b) $\epsilon_j = 0$.

Comments

1. This analysis is based on two assumptions: firstly that $\tilde{\theta}_0$ is close to θ^0, that is the analysis is only local, and secondly that an architecture of the NN can approximate the behaviour of the training set. Even though these assumptions seem quite restrictive, it is possible to have some insight into the convergence problem.

2. The conditions C1. and C2. are the most important from the solution point of view, because if they are not met it is possible to have a local minimum.

3. The condition C2. means that the derivatives must be different from 0, that is all the outputs of the units can not be 1 or 0 at the same time. In other words, if the input is bounded it will be necessary to bound the parameters. For example one way to accomplish this is to change the interpretation of the output [53].

4. If S is not invertible, the algorithm will not give a unique solution. It will have a 'valley' and certain linear combination of the parameters will not converge or converge an order of magnitude more slowly [48]. There

are many causes for the non-existence of S inverse, such as: the model set contains 'too many parameters' or when experimental conditions are such that individual parameters do not affect the predictions [48]. These effects arise during the learning process, because only some units are active in some regions contributing to the output.

5. The condition C4.b) arises when the structure of the connectionist representation can match the structure of the system.

6. Condition C3. shows that the order of presentation can be important to get convergence of the algorithm.

3.2.5 Summary

The stochastic approximation algorithm presented can successfully adjust the weights of a multilayer perceptron using many less presentations of the training data than required for the BP method. However, there is increased complexity in the calculation.

A formal proof of the convergence properties of the algorithm in this context has not been given; but, instead, some insights into the convergence problem, together with links back to appropriate control and systems theory, are given. Moreover, our experience indicates that the GBP algorithm is better than the BP method at performing continuous transformations. This difference stems from the fact that in the continuous mapping the transformation is smooth and it is possible to have a great number of patterns (theoretically infinity). Only a small number is used to train the connectionist representation, so it is possible to choose an adequate training set for a smooth representation, giving a better behaviour in terms of convergence.

A different situation arises in the discrete case, where it is possible to have some input patterns that do not differ very much from each other, and yet produce a very different system output. In terms of our interpretations, this means a rapidly varying hypersurface and therefore a difficult one to match with a few components.

3.3 General Learning Problem Using Gaussian Networks

As was described in Chapter 2 a gaussian network is represented by the following equation:

$$F(x) = \sum_{i=1}^{N} c_i g_{m_i,\sigma_i}(x), \tag{3.54}$$

where N is the number of units, $x \in R^d$ is an input vector, and $g_{m_i,\sigma_i}(x)$ is a gaussian function defined as in Chapter 2.

Here, the problem is to select the parameters c_i,m_i,σ_i, and N given a certain amount of input points contained in a region $C \subset R^d$, and their corresponding outputs so that the index $J = \|f - F\|_2^2 < \zeta$, ζ represents a tolerance. As Parzen [66] has shown, when the number of terms increase and as the variance term decreases to zero, the index vanishes. However, from a practical point of view a finite number of units N is desired; so it seems reasonable to attempt to minimise the index given a certain N. The same problem can be stated in matrix notation, defining

$$\begin{aligned} Y_l &= [f(x(1))\dots f(x(l))]^T, \\ G_l &= [g_1 \dots g_l],^T \\ g_i &= [g_{m_1,\sigma_1}(x(i)), \dots, g_{m_N,\sigma_N}(x(i))], \quad 1 \le i \le l \\ C &= [c_1 \dots c_N]^T, \end{aligned} \tag{3.55}$$

then for the training set containing n_t points equation (3.54) can be written in the compact form

$$Y_{n_t} = G_{n_t} C, \tag{3.56}$$

and the index as

$$J = \|f - F\|_2^2 = (Y_{n_t} - G_{n_t}C)^T (Y_{n_t} - G_{n_t}C).$$

3.3.1 Changing Format Approach

In Chapter 2 it has been shown that any continuous function on a compact set can be approximated by a finite sum of gaussians functions

$$\hat{f}(x) = \sum_{i=1}^{N} c_i g_{m_i,\sigma_i}(x). \tag{3.57}$$

The following approach is based on Medgyessy's work on decomposition of Functions [54].

Take the Fourier Transform (FT) of (3.57)

$$\hat{F}(\omega) = \sum_{i=1}^{N} c_i G_{m_i,\sigma_i}(\omega), \tag{3.58}$$

where $G_{m_i,\sigma_i}(\omega)$ is the FT of $g_{m_i,\sigma_i}(x)$. Then if the format of g is changed, that is if the variance is reduced by λ

$$g_{m_i,\sigma_i-\lambda}(x) = e^{\frac{-\|x-m_i\|^2}{\sigma_i-\lambda}}, \tag{3.59}$$

the FT of this function is

$$G_{m_i,\sigma_i-\lambda}(\omega) = \sigma_i^{\frac{-d}{2}} \pi^{\frac{-d}{2}} e^{-\frac{\sigma_i-\lambda}{4}\|\omega\|^2} e^{-j\omega^T m_i}. \tag{3.60}$$

This results in

$$G_{m_i,\sigma_i-\lambda}(\omega) = e^{\frac{\lambda}{4}\|\omega\|^2} G_{m_i,\sigma_i}(\omega), \tag{3.61}$$

where $0 < \lambda < \sigma_i$. If λ is near to σ_i, it is possible to distinguish the contribution of each unit to the building of the final function, in a way changing the format is like filtering the original function. To recover the filtered function it is necessary to apply the inverse FT

$$f_\lambda(x) = \mathcal{F}^{-1}\Big(\sum_{i=1}^{N} c_i e^{\frac{\lambda}{4}\|\omega\|^2} G_{m_i,\sigma_i}(\omega)\Big). \tag{3.62}$$

If $f_\lambda(x)$ is examined it is possible to select the number of units and position of these units, a simple example (see Figure 3.12) is given but a further systematisation is needed.

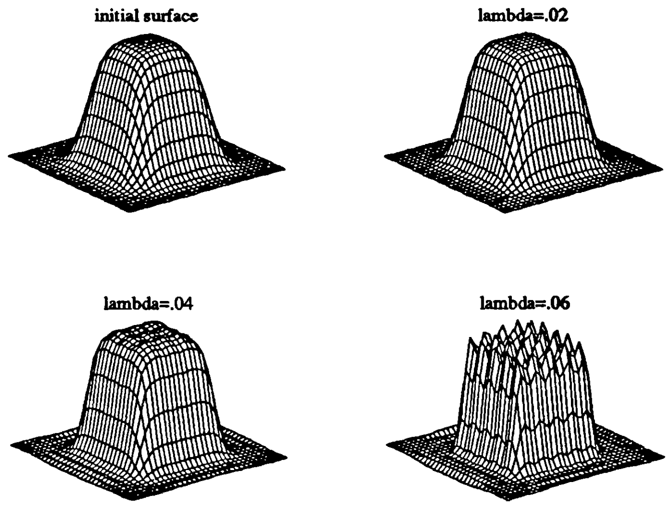


Figure 3.12: Different outputs of the network for different values of λ.

This example illustrates the filtering properties of the changing format approach, each figure shows the output of the system for different values of λ. Note that for $\lambda = 0.06$ it is possible to distinguish the structure of this function. In fact, from the last figure it is possible to see that this function can be synthesised with a grid of 25 units, with centres located at the maxima of the function $f_{\lambda=.06}$.

3.3.2 General Optimisation Approach

The objective of this approach is the minimisation of $||f-F||_2^2$ over all the training points with respect to the parameters c_i,m_i, and σ_i.

For the training set it is possible to write the index in the following way

$$||GC - Y||_2^2, \tag{3.63}$$

where $G \in R^{n_t} \times R^N$, $C \in R^N$ contains all the linear parameters, and $Y \in R^{n_t}$ all the desired outputs. As Golub and Pereyra [29] have shown, minimising J is equivalent to minimising

$$\hat{J} = ||P_G^{\perp} Y||_2^2, \tag{3.64}$$

where the matrix $P_G^{\perp}$ projects orthogonally onto the null space of G, that is $P_G^{\perp} = (I - GG^{+})$, where G^{+} is the pseudoinverse of G.

The jth column of the Jacobian matrix (see Lemma 4.1 [29]) of (3.64) is given by

$$\varphi_{\cdot,j}(\theta) = -[(P_G^{\perp}\frac{\partial G}{\partial \theta_j}G^{+}) + (P_G^{\perp}\frac{\partial G}{\partial \theta_j}G^{+})^T]Y, \tag{3.65}$$

where $\theta \in R^n \times R^{d\ n}$ is a vector containing all σ_i and m_i.

The gradient $\nabla \hat{J}$ is given by $\nabla \hat{J} = \varphi^T P_G^{\perp} Y$, in order to calculate the Jacobian, G can be factorised as

$$G = Q \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} Z, \tag{3.66}$$

where Q and Z are orthogonal matrices, $R \in R^r$ is an upper-triangular matrix of full rank $r \leq n_t$

$$P_G^{\perp} = Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n_t - r} \end{bmatrix} Q^T, \tag{3.67}$$

where I_{n_t-r} is the $(n_t - r) \times (n_t - r)$ identity matrix, and of course

$$G^{+} = Z^T \begin{bmatrix} R^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q^T. \tag{3.68}$$

This algorithm is efficient but very time consuming, due to the factorisation. Another drawback is that all samples must be available to train the network, enabling only batch operation.

A much simpler algorithm is the gradient descent over the original index J. Define the error at a step l as

$$e(l) = y(l) - F(x(l)), \tag{3.69}$$

where $y(l)$ represents the value of the function at $x(l)$, that is $y(l) = f(x(l))$. Then the following expressions for the gradient evaluated in each $[y(l), x(l)]$ are obtained:

$$\begin{aligned} \frac{\partial e(l)}{\partial c_i(l)} &= -g_{m_i,\sigma_i}(x(l)), \\ \frac{\partial e(l)}{\partial m_{ij}(l)} &= -c_i g_{m_i,\sigma_i}(x(l))\frac{(x_j - m_{ij})}{\sigma_{ij}^2}, \\ \frac{\partial e(l)}{\partial \sigma_{ij}(l)} &= -c_i g_{m_i,\sigma_i}(x(l))\frac{(x_j - m_{ij})^2}{\sigma_{ij}^3}, \end{aligned} \tag{3.70}$$

where m_{ij} represents the element j of m_i, x_j the element j of x, and σ_{ij} represents element j of σ_i. The algorithm accumulates the changes of the parameters over the training set

$$\begin{aligned} \nabla c_i &= \nabla c_i - \alpha_c e(l) \tfrac{\partial e(l)}{\partial c_i(l)}, \\ \nabla m_{ij} &= \nabla m_{ij} - \alpha_{M_i} e(l) \tfrac{\partial e(l)}{\partial m_{ij}(l)}, \\ \nabla \sigma_{ij} &= \nabla \sigma_{ij} - \alpha_\sigma e(l) \tfrac{\partial e(l)}{\partial \sigma_{ij}(l)}, \end{aligned} \tag{3.71}$$

and once all the training points have been presented all the parameters are updated by

$$\begin{aligned} c_i(ep) &= c_i(ep) + \alpha_c \nabla c_i + \beta_c \Delta c_i(ep), \\ m_{ij}(ep) &= m_{ij}(ep) + \alpha_M \nabla m_{ij} + \beta_M \Delta m_{ij}(ep), \\ \sigma_{ij}(ep) &= \sigma_{ij}(ep) + \alpha_M \nabla \sigma_{ij} + \beta_\sigma \Delta \sigma_{ij}(ep), \end{aligned} \tag{3.72}$$

where ep is the index to the current *epoch*, an epoch is the presentation of the complete training set, l represents just an index over the training set, and the operator Δ is defined by $\Delta x(ep) = x(ep) - x(ep-1)$ and represents a *momentum* term.

In addition to the basic algorithm a heuristic procedure to change the gradient steps α_c, α_M, and α_σ and the *momentum* is added [88] [6]. This leads to the following:

```
• set $l = 0$, $E = 0$, $ep$, $tol$ as a level of tolerance for the sum of the squared
  error.
• $ne_{max}$ = maximum number of epochs,
• $ns$ = number of cycles required to increment the gradient steps.
• while $ep < ne_{max}$ or $E > tol$ do
    – $ep = ep + 1$;
    – set $\nabla c_i = \nabla m_{ij} = \nabla \sigma_{ij} = 0$;
    – while $l < n_t$ do
        * $l = l + 1$
        * apply equation (3.70) and accumulate changes using (3.71)
        * calculate $E = E + e(l)^2$
    - end;
    - if *there is reduction of the index by ns consecutives steps* then *double the step;*
    - if *there is not a reduction in $E$* then
        * *reduce the step by half*
        * *discard all updates*
        * *set $\beta$ to 0*
    - else
        * *set $\beta$ to its original value*
        * *update parameters using (3.72)*
    - end;
• end.
```

This algorithm is easily implementable in parallel computers and an on-line version, where the parameters are updated at every step, can be implemented.

3.3.3 Multi-algorithm Approach

The idea of having several algorithms working in parallel in order to solve a problem is worth investigating. As was described in Chapter 2, each parameter is associated with some characteristic of the function to be approximated. To reach a certain level of the error each parameter is changed by means of the following algorithms:

- **Clustering:** The centres of the gaussian functions are computed by a clustering algorithm, in this way the location of the centres is decided by the

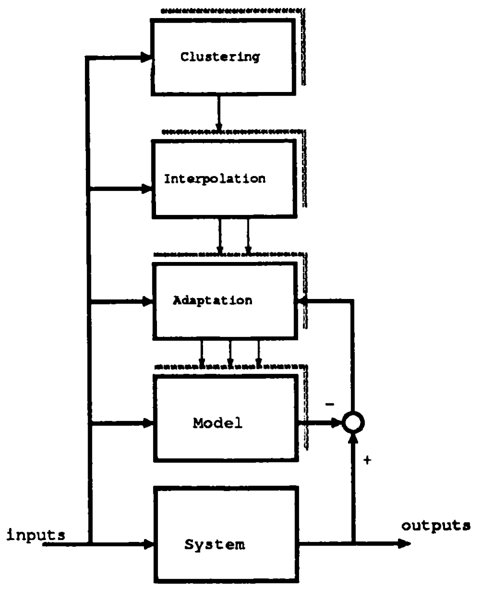


Figure 3.13: Hierarchical algorithms.

characteristic of the inputs. In general it is desired to have more units in those regions where more data is available.

- **Interpolation:** The widths are adjusted to reach a certain degree of overlaping between the centres.

- **Adaptation:** the linear coefficients are calculated in order to minimise the difference between the output of the network and the desired output.

These algorithms are organised in an hierarchical way as shown in Figure 3.13. The whole scheme can be embedded in a temporal framework in which the time scale varies according to the hierarchical position of the algorithm, that is the *adaptation* procedures should work at relative higher speed than the *interpolation* algorithm, and *clustering* a lower speed than the algorithm downstream. This defines a series of time operational envelopes for the different algorithms.

Clustering

Given a number of units it is necessary to distribute their centres in a certain way over the input space. In this case, it is possible to use the *Kohonen algorithm* to adjust the centres of the function m_i. The asymptotic values of the weight vectors constitute one kind of vector quantisation of the input space, which depends on the statistical characteristic of the input. The procedure has the following steps:

1.- select an input of the input variable $x(l)$;

2.- select the unit index i with centre M_i which $\min_i \|x(l) - m_i\|$;

3.- update the parameter $m_i = m_i + \alpha(x(l) - m_i)W_j$, where W_j defines a topological neighbourhood described by

$$W_j = \begin{cases} 1, & \forall j \in [i - N_c, i + N_c]; \\ 0, & otherwise. \end{cases} \tag{3.73}$$

N_c defines the size of the topological neighbourhood, which is a function of time, and α is a slowly decreasing function sequence < 1 [40];

4.- if there is no more improvement in the values of m_i, stop, otherwise go to part 1.

Another algorithm is the *k-means algorithm* which computes n centres representing a local minimum of the total squared euclidean distances between the n_t examples of $x(l)$ and the nearest of the n centres m_i:

$$E = \sum_{i=1}^{n} \sum_{l=1}^{n_t} M_{il}(x(l) - m_i)^2.$$

Here M_{il} is the cluster membership functions, which is a $n \times nt$ matrix of 0's and 1's with exactly one 1 per column which identifies the processing units which a given exemplar belongs.

The local minimisation of E can be performed as an interative process over the whole training set or as an adaptive incremental process [58].

The batch approach can be summarised as follows:

1.- the centres of each cluster are initialized to different training points;

2.- for each training example: Each training example $x(l)$ is assigned to the unit nearest to it;

3.- for each the unit: The average position of the training points for each units is found and unit's centres is moved to that point;

4.- go to 2 until it converges.

The on-line version follows the following procedure

At each training step:

1.- select an input of the input variable $x(l)$;

2.- select the unit index i with centre m_i which $\min_i ||x(l) - m_i||$;

3.- update the parameter $m_i = m_i + \alpha(x(l) - m_i)$;

4.- if there is no more improvement in the values of m_i, stop, otherwise go to part 1.

This algorithm has a decay characteristic that makes it sub optimal in its use of past data; but under some circumstances it will be able to adapt to changes in the system. As a direct consequence of decay in the estimator the variance does not approach zero, but as l increases the expected mean value of m_i approaches the expected mean value of $x(l)$ as a limit [56]. In fact, seting $x_i^s(l)$ as the value assigned to the nearest centre m_i, the learning algorithm can be written as

$$m_i = (1 - \alpha)z^{-1}m_i + \alpha x_i^s.$$

This means that the value of the output of this system is equal to the mean value of the input and the variance is given by

$$var(m_i) = \frac{\alpha^2}{(1 - (1 - \alpha)^2)} var(x_i^s).$$

The main advantage of this algorithm is that it requires less memory than the batch approach, but has the disadvantage of retaining a certain variance in the solution.

There is a surprising analogy between the Kohonen algorithm and the gradient algorithm,

Gradient rule: $m_{ij}(l+1) = m_{ij}(l) - \alpha c_i g_{m_i,\sigma_i}(x(l))(x_j - m_{ij})e(l)$,

Kohonen rule: $m_{ij}(l+1) = m_{ij}(l) + \alpha(l) W_j (x_j - m_{ij}) e(l)$.

From these equations it is possible to see that the Kohonen algorithm is like the gradient rule but the error has not been considered and a decreasing sequence for the learning rate has been introduced. In the gradient rule the neighbourhood is given by the response of the gaussian unit, which acts as a topological neighbourhood.

Finally, another common approach is to distribute the centres uniformly along the input space without taking into consideration the characteristic of the input signal at all.

Interpolation

An interpolation condition is used to adjust σ_i, such that a continuous interpolation over all the input region is obtained. A similar approach has been suggested in [58]. The interpolation condition is expressed by the following functional

$$J = \sum_{l=1}^{n_t} (1 - \frac{\sum_{i=1}^{n} g_{m_i,\sigma_i}(x(l))}{\bar{K}})^2 \tag{3.74}$$

where $\bar{K}$ is defined as

$$\bar{K} = \frac{\sum_{l=1}^{n_t} \sum_{i=1}^{n} g_{m_i,\sigma_i}(x(l))}{n_t},$$

and (3.74) is minimised with σ_i. It is interesting to note that if the input variables x are normalised and the centres m_i are distributed uniformly, then $\sigma_i (= \sigma)$ is constant for all units, so the minimisation of J is reduced to a one-dimensional search which contains one global minimum. These ideas are similar to the one used in [85].

In both cases the parameter updates depend only on the inputs. The result obtained with this procedure is shown in Figure 3.14, which considers a grid of 121 units.

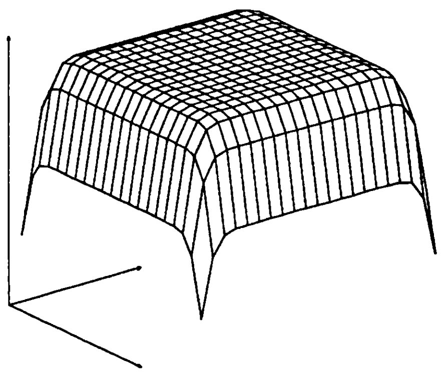


Figure 3.14: Output of the network.

There are other heuristics that can be used, the most popular is the $P-$ *nearest neighbour heuristic* [58]:

$$\sigma_i = \sqrt{\frac{1}{P}\sum_{j=1}^{P} \|x(j) - m_i\|},$$

where $x(l)$ is the P-nearest neighbors of m_i; a common value for P is 2 and as a general guideline P should be much smaller than n_t.

Adaptation

To adjust the weights C a linear least square approximation problem must be solved. Using the pseudoinverse the solution to this problem is

$$C = G^+ Y. \tag{3.75}$$

The solution of the above equation can be performed in a recursive form. If the solution to the system with n data is given by

$$C_n = G_n^+ Y_n. \tag{3.76}$$

and then a new observation is obtained

$$C_{n+1} = G_{n+1}^+ Y_{n+1}, \tag{3.77}$$

where

$$G_{n+1} = \begin{bmatrix} G_n \\ g_{n+1} \end{bmatrix}. \tag{3.78}$$

It turns out that C_{n+1} is related to C_n [2] by

$$G_{n+1}^{+} = [(I - H_{n+1} g_{n+1}) G_n^{+} \vdots H_{n+1}], \tag{3.79}$$

where

$$H_{n+1} = \begin{cases} \frac{(I - G_n^{+} G_n) g_{n+1}^T}{g_{n+1}(I - G_n^{+} G_n) g_{n+1}^T} & \text{if } (I - G_n^{+} G_n) g_{n+1}^T \neq 0 \\ \frac{G_n^{+} G_n g_{n+1}^T}{1 + g_{n+1}(I - G_n^{+} G_n) g_{n+1}^T} & \text{otherwise.} \end{cases} \tag{3.80}$$

Thus equation (3.77) has the form

$$C_{n+1} = (I - H_{n+1} g_{n+1}) G_n^{+} Y_n + H_{n+1} y_{n+1}, \tag{3.81}$$

and this is

$$C_{n+1} = C_n + H_{n+1}[y_{n+1} - g_{n+1} C(n)], \tag{3.82}$$

where the initial conditions for C and H are 0. Here the recursion takes the form of a predictor; where the error between the prediction and the measurement is used to correct the parameters.

Taking the basic structure of (3.82), but with a diagonal constant matrix H, it is possible to arrive at the same result saving a lot of memory, because there is no need to store a square matrix, which can be prohibitive when the number of units is very large.

There are two versions, a batch version where the changes are summed over the whole training set and the sum is applied to modify the parameters after each iteration over all the training set, and an on-line version where the training set is presented at random and the parameters are updated in each presentation.

In the batch algorithm the parameters are updated as

$$C(k) = C(k-1) + \alpha[Y_{n_t} - G_{n_t} C(k-1)] G_{n_t}^T, \tag{3.83}$$

where the matrices C, Y_{n_t}, and G_{n_t} are defined as (3.55).

Theorem 6 *Let $C(0) = 0$, and $0 < \alpha < \frac{2}{c}$, where $c = \max(\alpha_1^2, \alpha_2^2, \ldots, \alpha_n^2)$, eigenvalues of G_{n_t}. Then the learning law (3.83) converges to the solution*

$$C_\infty = G^{+} Y.$$

Proof:

Write (3.83) as a function of $C(0)$

$$C(k) = (I - \alpha G^T G)^k C(0) + \alpha \sum_{k=j}^{k} (I - \alpha G^T G)^j G^T Y.$$

As $k \to \infty$ the term $\alpha \sum_{j=0}^{k} (I - \alpha G^T G)^j G^T$ tends to G^+ (**Theorem 3.5.1** [71]), then the result follows $\diamondsuit\diamondsuit$

In [65] there is a complete analysis of convergence based on Lyapunov functions and a description of different results that can be obtained using this algorithm.

For the on-line version the use of the following theorem shows that the algorithm defined by

$$C(n) = C(n-1) + \alpha[y_n - g_n C(n-1)]g_n^T, \tag{3.84}$$

or equivalently

$$C(n) = \prod_{j=1}^{n} (I - \alpha g_j g_j^T) C(0) + \sum_{i=1}^{n} \prod_{j=1}^{i} (I - \alpha g_j g_j^T) \alpha g_i y_i$$

can solve equation (3.56).

Theorem 7 *Let the sequence $\{i_r\}_{r=1}^{\infty}$ be such that there is a uniformly bounded number of elements between any two successive occurrences of each fixed integer i of the set $\{1, \ldots, M\}$. Let:*

(i) $\{b_1, \ldots, b_m\}$, with $M \geq 1$, be a set of nonzero linearly independent vectors of finite norm in R^N;

(ii) $T_k = \prod_{r=1}^{k} (I - \alpha_r b_{i_r} b_{i_r}^T)$, where α_r are scalars;

(iii) $0 < \alpha_r < 2\|b_{i_r}\|^{-2}$ for every r, where $\|.\|$ is the Euclidean norm.

Then the sequence of matrices $\{T_k\}_{k=1}^{\infty}$ converges to $P_B = I - B^T(BB^T)^{-1}B$, where $B = [b1, \ldots, b_m]$.

Proof: see [41].

With the aid of Theorem 7 it is possible to prove the following theorem

Theorem 8 *The learning law (3.84) under Theorem 7 conditions converges to the solution*

$$C_\infty = (I - G^+G)C(0) + G^+Y,$$

which is the least square solution of the equation $Y = GC$.

Proof: [41].

The cost involved in the use of a much simpler algorithm is the speed of convergence. For the recursive version of the pseudo-inverse the convergence is given by the number of samples, i.e. if there are n independent vectors the algorithm converges to the optimal solution in n steps. This should be compared with the gradient algorithms whose convergence depends on the parameter α.

Multialgorithm in Continuous Time

Extensions to continuous time analysis have been done [80]; a basic gradient law has been used to adjust the parameters, and Lyapunov arguments have been used to show that this adaptation law is robust enough against modelling errors.

3.3.4 Training a Regular Gaussian Network

Suppose that the system can be represented by

$$\hat{f}(x) = \sum_{[l]} c_l {g^n}_{m_l,\sigma_l}(x),$$

where g^n is the normalised gaussian function as defined in Chapter 2, m_i represents a grid defined as

$$m_l = l_1 e_1 \Delta + \ldots + l_d e_d \Delta,$$

where $\{e_1, \ldots, e_d\}$ is a basis vector for R^d, and $[l] = \{l \in Z\}$. Then the error can be expressed as

$$\hat{f} - f = \frac{1}{2\pi^d} \int_\Omega (\sum_{[l]} c_l e^{-\omega^T m_l} G(\omega\sqrt{\sigma}) - \phi(\omega)) e^{j\omega^T x} d\omega.$$

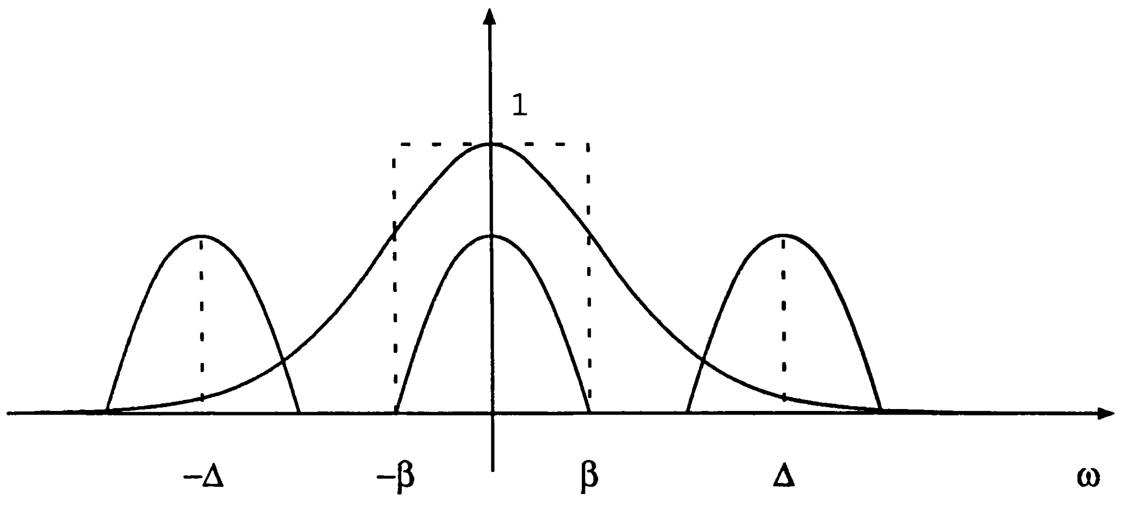


Figure 3.15: Components of the error in the approximation (1-dimensional case).

Noting that $c(x)$ can be calculated as $\frac{1}{2\pi^d}\int_\Omega \phi(w)e^{jw^Tx}dw$ or equivalently as was shown in Chapter 2 $c_l = f(m_1)V(Q_d)$ such that

$$\sum_{[l]} c_l e^{-\omega^T m_l} = \sum_{[k]} \phi_F(\omega - n_k),$$

then

$$\hat{f} - f = \frac{1}{2\pi^d}\int_\Omega (G(\omega\sqrt{\sigma})\sum_{[k]}\phi_F(\omega - n_k) - \phi(\omega))e^{j\omega^T x}d\omega.$$

The error can be approximated by two components , the first one due to the filtering properties of G and the second one due to the lack of compact support of the gaussian function as it is shown in Figure 3.15, where is it visible that a compact support is lacking.

Defining

$$\epsilon_1 = \frac{1}{2\pi^d}\int_\Omega (1 - G(\omega\sqrt{\sigma}))\phi(\omega)e^{j\omega^T x}d\omega,$$

and

$$\epsilon_2 = \frac{1}{2\pi^d}\int_\Omega \sum_{[k^0]}\phi(\omega - n_k)G(\omega\sqrt{\sigma})\phi(\omega)e^{j\omega^T x}d\omega,$$

where $[k^0] = \{k \in Z | k \neq 0\}$. From these expressions it can be seen that there is an obvious compromise between these two errors. The centres of the units are related with the sampling theorem and must be chosen at least such that $\Delta_i > 2\beta$, where

β represents the bandwidth of the function to be approximated, as the radius of the smallest d-cube, centered at the origin which completely encloses the support of $\phi(w)$. The width σ must be chosen such that $\epsilon_1 + \epsilon_2$ is minimum. A heuristic approach adopted in [75] is to select sigma, such that

$$e^{-\frac{\sigma}{4}(2\pi\beta)^2} = .3678,$$

that is $\sigma\pi^2\beta^2 = 1$ and then the sampling interval is chosen, such that the exponent almost vanishes to that position, that is $\sigma\pi^2\frac{1}{\Delta^2} = 4$.

There is a third source of error, the one produced for a finite set of units. If the function is written as

$$\epsilon_3 = \sum_{[l_Q]} c_i {g^n}_{m_i,\sigma_i}(x) + \sum_{[l_{\bar{Q}}]} c_i {g^n}_{m_i,\sigma_i}(x),$$

then the error is

$$\epsilon_3 = \int_Q |\sum_{[l_{\bar{Q}}]} c_i {g^n}_{m_i,\sigma_i}(x)| dx.$$

Here $[l_{Q_d}]$ is defined as $[l_{Q_d}] = \{l \in Z | m_i \in Q_d\}$, and $[\bar{l}_{Q_d}]$ is such that $[l_{Q_d}] \cup [\bar{l}_{Q_d}] = Z$. Due to the gaussian characteristic this error is only important in the borders of the approximation.

Data Driven Smoothing Approach

It is natural to try to use the training data itself to determine an appropriate degree of smoothing. This general approach is simple and leads to a batch algorithm as described below.

1.- the centres are fixed such that they meet the conditions of the sampling theorem;

2.- calculate σ such that $\min_\sigma J = \|(I - GG^+)Y\|_2^2$;

3.- calculate $C = G^+Y$.

In step 2 there is an implicit solution of the equation $Y = GC$ obtained by $C = G^+Y$, and for this reason the method has been named *data driven approach.*

3.3.5 Simulations

In order to illustrate the differences in the approximations reached by the different algorithms two examples in two dimensions were selected.

Discontinuous Function

The first example illustrates the performance of the algorithm approximating a discontinuous function defined by

$$Y(l) = f(x_1(l), x_2(l)) = \begin{cases} 10 & , x_1(l) > 5 \text{ or } x_2(l) > 5; \\ 2 & , \text{otherwise} \end{cases} \tag{3.85}$$

where x varies in the interval $[0, 10]$. Figure 3.16 shows the desired surface, generated by the training set, containing 121 uniformly distributed points. Figure 3.17 shows the solution reached by the *optimisation* approach after 1000 epochs. The solution reached is quite good and reflects the fact that σ_i has been varied to meet the different requirements in the slope of the surface.

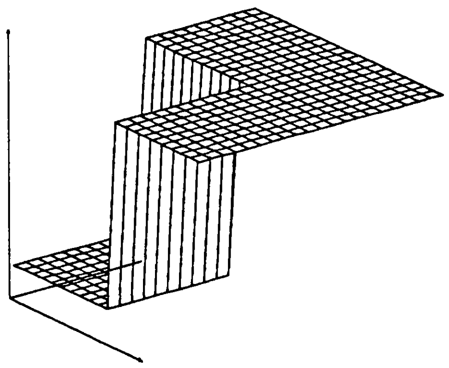


Figure 3.16: Training surface.

Figure 3.18 shows the solution reached by *multi-algorithm* approach using 121 units. Note the oscillatory effect near the discontinuity.

Coordinate Transformation

A network of 121 units was used to approximate a function representing a coordinate transformation, as explained in section 3.2.3. The *general optimisation*

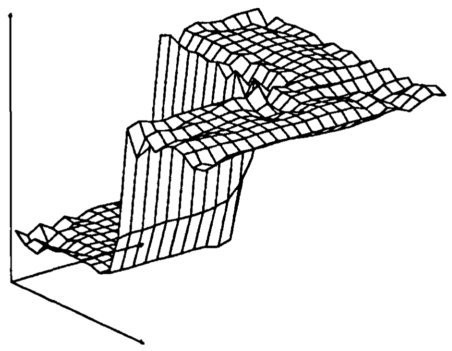


Figure 3.17: Surface generated by the network after training **using** ***optimisation*** approach.

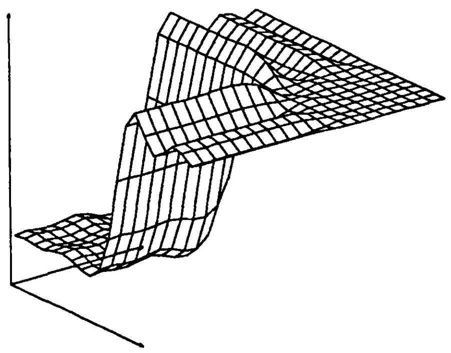


Figure 3.18: Surface generated by the network after training with ***multi-algorithm*** approach.

algorithm converges in 380 epochs to a solution with a sum squared error less than 0.0613, the curve of the error is shown in Figure 3.19. The surface used as a training surface is shown in Figure 3.20 and final surface reached by the algorithm is shown in Figure 3.21.

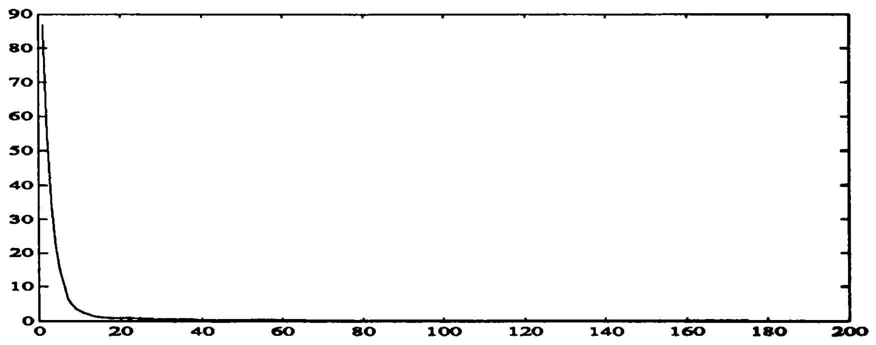


Figure 3.19: Sum squared error for the optimisation approach.

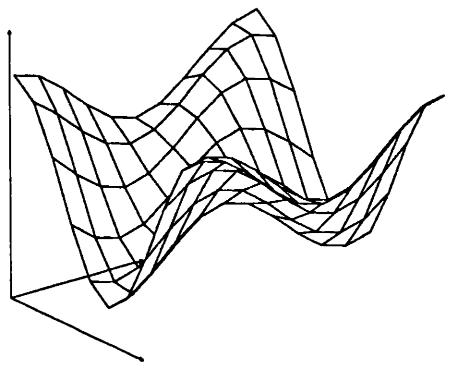


Figure 3.20: Surface representing the coordinate trasformation between θ_1, θ_2, and y.

The *multi-algorithm* approach produces an approximation shown in Figure 3.22, with a sum squared error of 3.2×10^{-4}.

3.3.6 Summary

To summarise the characteristics of the algorithms a comparison of the use of different input-output variables for each of them is given in table 3.4.

Under the actual implementations it is not possible to compare processing times, mainly because their implementations are different. Instead, a general

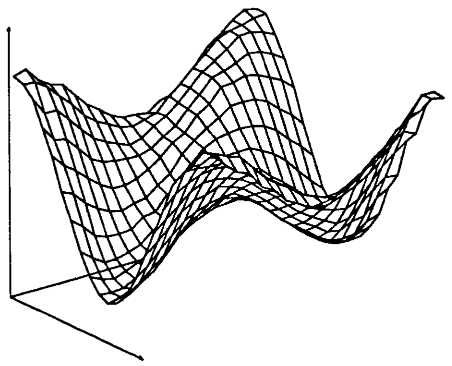


Figure 3.21: Surface representing the solution after 380 epochs.

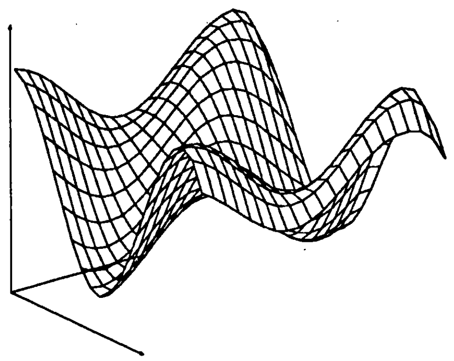


Figure 3.22: Surface representing the solution reached by the *multi-algorithm* approach.

Algorithm	input data	output data
Optimisation	n, training data	c_i, M_i, σ_i
Multi-algorithm	n, training data	c_i, M_i, σ_i

Table 3.4: Input-Output data for the different algorithms.

comparison is given.

The *optimisation approach* has the drawback that the estimates can become trapped at a local minimum of the cost function during the updating process. In general, the optimisation procedure is sensitive to initial parameters. The following hints can be helpful in overcoming the difficulty: distribute M_i uniformly over the input space, set σ_i to small random values, and set c_i to zero. From the user point of view there is a lot of parameters (α's β's) that must be specified.

The *multi-algorithm* approach is simple and the interpolation among points is embedded in the algorithm. A general drawback is that it is necessary to use more units for functions with big variations in their slopes, this effect is produced because all units have the same σ, fixing in this way the maximum variation of the function which can be approximated. This approach reduces the problem to a linear in the parameter minimisation problem having global minima only.

Further examples and descriptions of other algorithms can be found in [78].

Gaussian networks offer high flexibility in the approximation of nonlinear functions, each parameter has a clear influence in different aspects of the outputs, so it is easy to devise several algorithms to train them. Among the algorithms analysed the *multi-algorithm* method is the simplest and offers a good compromise between complexity and accuracy in the approximation. On the other hand, *optimisation* offers the capability of approximating very complex functions adjusting each σ_i individually.

3.4 Conclusions

The solutions reached with sigmoid and gaussian networks have different characteristics. From the simulations done it is possible to see that gaussians are more suitable for approximating discontinuous functions with small training sets. On the other hand, gaussians can approximate well any continuous function with any desired degree of accuracy if they are provided with enough training data. The advantage of using gaussian stems from the convergence of their learning algorithm, which is simple and easy to implement.

Chapter 4

System Representation and Concepts

SUMMARY

This Chapter describes the general conditions under which a nonlinear system can be represented as a nonlinear function of a finite set of past input-output values, establishing the link between the connectionist repesentation of a nonlinear functions and the representations of a nonlinear dynamical systems. Then different-key concepts like observability, controllability, and invertibility of a nonlinear dynamic system are defined. Finally, the problem of identification of a dynamic system and its inverse with connectionist model is analysed.

4.1 Nonlinear Dynamic Systems Representation

The class of systems considered in this study consists of time-invariant single-input single-output nonlinear dynamic systems in discrete time. The output of the system can be expressed as a function of a finite number of values of the input and past values of the output. The relationship between the output y and the input u is given by the following equation

$$\Sigma : \qquad y(k+1) = f(y(k)\ldots, y(k-n), u(k), \ldots, u(k-m)), \quad (4.1)$$

where m and n are fixed integers, $f : R^{n+1} \times R^{m+1} \rightarrow R$ is a multivariable function, and $m < n$. In this case, the system can be regarded as a mapping between the input defined by the vector $[y(k)\ldots, y(k-n), u(k), \ldots, u(k-m)]$ and the output at next sampling time. The integer n is called the ***principal degree*** of the recursive equation (4.1).

Defining the vector

$$x(k) = [y(k)\ldots, y(k-n), u(k-1), \ldots, u(k-m)]^T$$

the system represented by (4.1) can be expressed in the state space form as

$$x(k+1) = \begin{bmatrix} 0 & \ldots & 0 & 0 & \ldots & 0 \\ & I_n & & \vdots & & \vdots \\ & & & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & & \vdots & & I_{m-1} & \\ 0 & \ldots & 0 & & & \end{bmatrix} x(k) + \begin{bmatrix} f(x(k), u(k)) \\ 0 \\ \vdots \\ u(k) \\ 0 \\ \vdots \end{bmatrix}, \quad (4.2)$$

and the output is defined by

$$y(k) = [1, 0, \ldots, 0]x(k).$$

The following question arises: what kind of systems can be represented by the recursive equation (4.1)?. The answer to this question was given by Leontaritis and Billings [45], [46]. A summary of the discussion is given below.

Define a general discrete time-invariant system S

$$\begin{aligned} x(t+1) &= g[x(t), u(t)], \\ y(t) &= h[x(t), u(t)], \end{aligned} \quad (4.3)$$

where $t \in Z$, $x(t+1), x(t) \in X \subset R^n$ the state set of dimension n, $u(t) \in U \subset R^1$ the input set of dimension 1, $y(t) \in Y \subset R$ the output set of dimension 1, $g : Z \times X \times U \rightarrow X$ the one step-ahead or local state transition function and $h : Z \times X \times U \rightarrow Y$ the output function [63].

Obviously, this state space model is more general than the input-output model described by equation (4.2), because all the states are nonlinear functions of the previous states.

The many steps transition function Φ, describes the dynamics for all times $t + k$, and can be found by repeated application of the function g to (4.3)

$$\begin{aligned} x(t+2) &= g[x(t+1), u(t+1)] \\ &= g[g[x(t), u(t)], u(t+1)] \\ &= \Phi[x(t), u(t+1), u(t)] \\ x(t+3) &= \dots \end{aligned} \tag{4.4}$$

Let U^* be the set of all finite input sequences of members of the set U

$$U^* = \{u(t+k-1), \dots, u(t+1), u(t) |\ k \geq 0 \text{ and } u(t+j) \in U, j = 0, 1, \dots, k-1\}.$$

The empty sequence denoted by e is included in U^* and corresponds to $k = 0$. The set of all sequences minus the empty sequence e is the set U^+.

The state transfer function can now be defined as

$$\Phi : X \times U^* \rightarrow X. \tag{4.5}$$

If $w = [u(t+k-1) \dots u(t+1)\ u(t)]$ then

$$x(t+k) = \Phi[x(t), w] \quad \text{for} \quad k > 0 \tag{4.6}$$

and

$$x(t) = \Phi[x(t), e] \quad \text{for} \quad k = 0. \tag{4.7}$$

When the system is at an equilibrium state $x(t)$, its behaviour from that state can be described by a function $f_{x(t)}$ called the *input-output map* of the system, which is defined as

$$f_{x(t)} : U^{+} \rightarrow X, \tag{4.8}$$

where

$$f_{x(t)} = h(\Phi[x(t), u(t+k-1) \ldots u(t+1), u(t)]). \tag{4.9}$$

Realisation Problem

In general the only thing known is the function $f_{x(0)}$, which is obtained as a result of measurements upon the system of interest, but whose dynamic equations are not known. Obviously, the function $f_{x(0)}$ is not practical to be used as a model, because the amount of memory needed to store $f_{x(0)}$ increases linearly with the time scale. A more pragmatic approach is to find a function S which represents the input-output behaviour of the system, requiring just a finite number of past data. This defines *the realisation problem*, which can be stated as:

> Given a function $F : U^{+} \rightarrow Y$ find the system S such that it has a particular state $x(0)$, which realises F, i.e. such that the response function of the system $f_{x(0)}$ of S is equal to the function F.

Under this definition, there is a possibility that more than one system S satisfies the realisation definition, but of all these the simplest or minimal in some sense must be chosen [63]. The criterion for minimality commonly demanded is that the realisation S should be reachable and observable, which corresponds to the minimal dimension criterion in the case of linear systems.

Hammer [31] found that the minimal recursive representation can be obtained from an arbitrary given recursive representation of the system through a step by step reduction procedure, which is based on two concepts *minimal domain* and *globally degenerate domain.* Define the input-output space D_0 as the *minimal* *domain* over which the recursion function f has to be defined in order to characterise the input-output relationship induced by the system. It is said that D_0 is *globally degenerate* if for every point $x(k) \in R^{n+1} \times R^m$, the one step ahead continuation of $x(k)$ is uniquely determined.

The procedure starts from a given recursive representation Σ (see equation (4.1)) with principal degree n, then the reduced recursive representation Σ_1 is obtained, having principal degree $(n-1)$ and input-output space D_0^1. If D_0^1 is still globally degenerate the same procedure can be applied to Σ_2. After k such steps, where k is at most $(n+1)$, a recursive representation Σ_k of the system having principal degree $(n-k)$ is obtained, for which the recursion does not depend on the output variable, i.e. $n-k=-1$, or its input-output space is no longer degenerate. This procedure does not rely on any special characteristic, like continuity and/or invertibility, of function f to obtain a minimal representation, being in this way more general than the approach described next.

If the state space model is considered, and if $x(k)$ can be solved in terms of the values of the output and input, then it not only implies observability but also that a recursive input-output equation can be derived for the nonlinear system. The function that describes the input-output behaviour of the system is very important because this is all that an external observer can obtain.

To illustrate the relation of input-output representation and observability, firstly a linear system is considered

$$\begin{aligned} x(t+1) = &\ Ax(t) + Bu(t), \\ y(t) = &\ Cx(t), \end{aligned} \tag{4.10}$$

where $x \in R^n$, $u(k) \in R$, A is a $[n \times n]$ matrix, B is a $[n \times 1]$ vector, and C is a $[1 \times n]$ matrix. Write the future values of y in terms of $x(k)$ and $u(k)$

$$\begin{aligned} y(t) = &\ Cx(t) \\ y(t+1) = &\ CAx(t) + CBu(t) \\ &\vdots \\ y(t+l) = &\ CA^l x(t) + \textstyle\sum_{j=1}^{l} CA^{j-1}Bu(t+j-1) \end{aligned} \tag{4.11}$$

or, in short notation,

$$Y_{t,l} = [CA]_{t,l}x(t) + [CAB]_{t,l}U_{t,l}.$$

From this equation it follows that the observability depends on the characteristic of matrix $[CA]_{t,l}$. As rows in $[CA]_{t,l}$ cannot be independent of previous rows if $l > n - 1$ because of the Cayley-Hamilton theorem, this means $\text{rank}([CA]_{t,l}) \leq n, \forall l$. If $[CA]_{t,l}$ is invertible, i.e has full rank for $l = n - 1$, then the system is observable and has the input-output representation given by

$$y(t+n) = C[CA]_{t,n-1}^{-1}(Y_{t,n-1} - [CAB]_{t,n-1}U_{t,n-1}) + \sum_{j=1}^{n} CA^{j-1}Bu(t+j-1).$$

Clearly the input in this case is not relevant if we assume that all values of $u(k)$ are available in the computation of $x(t)$. This representation cannot be minimal, because the reachability condition has not been imposed.

For the nonlinear system, the many step transition function for the system (4.3) can be written as

$$x(t+k) = \Phi_k(U_{t,k-1}, x(t)),$$

where U^k is the input sequence of length k and $\Phi_k : U^k \times X \rightarrow X$, then at any time k

$$\begin{aligned} y(t) = &\; h[\Phi_0(x(t), u(t))], \\ y(t+1) = &\; h[\Phi_1(x(t), u(t))], \\ &\;\vdots \\ y(t+k-1) = &\; h[\Phi_{k-1}(x(t), u(t+k-2), \ldots, u(t))]. \end{aligned} \tag{4.12}$$

The function (4.12) can be written as

$$Y_{t,k} = G_k(x(t), U_{t,k-1}), \tag{4.13}$$

where $G_k : U^k \times X \rightarrow Y^k$.

The necessary condition for $x(t)$ to be solved in terms of the the output and input values is given by the Implicit Function Theorem [72], which states roughly speaking, that a continuously differentiable mapping $f(x, y)$ can be solved for y in terms of x at any point (a, b) at which $f(a, b) = 0$ and $\frac{\partial f}{\partial y} \neq 0$.

Theorem 9 (Implicit Function) *Suppose f(x,y) is a continuously differentiable mapping of an open set $E \subset R^{n+m}$ into R^n, such that $f(a,b) = 0$ for some point $(a,b) \in E$.*

Put $A = f'(a,b)$ and assume that $A_x = \frac{\partial f}{\partial x}|_{(a,b)}$ is invertible.

Then there exist open sets $U \in R^{n+m}$ and $W \subset R^m$, with $(a,b) \in U$ and $b \in W$, having the following property:

To every $y \in W$ corresponds a unique x such that

$$(x,y) \in U \quad \text{and} \quad f(x,y) = 0.$$

If this x is defined to be $g(y)$, then $g(y)$ is a continuously differentiable mapping of W into R^n, $g(b) = a$,

$$f(g(y),y) = 0 \quad (y \in W)$$

and

$$g'(b) = -(A_x)^{-1}A_y.$$

Proof: [72].

The assumption that A_x is invertible means that the n by n matrix

$$\begin{bmatrix} D_1 f_1 & \dots & D_n f_1 \\ \vdots & \ddots & \vdots \\ D_1 f_n & \dots & D_n f_n \end{bmatrix}, \tag{4.14}$$

evaluated at (a,b), defines an invertible linear operator in R^n; in other words, its column vectors should be linearly independent. Furthermore, if $f(x,y) = 0$ holds when $x = a$ and $y = b$, then the conclusion of the theorem is that $f(x,y) = 0$ can be solved for x in terms of y for every y near b, and these solutions are continuously differentiable functions of y. Naturally, the conditions are only valid locally. In reference [74] can be found the necessary and sufficient conditions under which it is possible to globally and uniquely solve $f(x,y) = 0$ for x in terms of y, with the solution map continuous.

If the conditions of Theorem 9 are satisfied then there exists a continuous function G^*_{k-1} such that $x(t) = G^*_{k-1}(Y_{t,k-1}, U_{t,k-2})$. Using equation (4.12) the following relation is obtained

$$y(t+k) = h[\Phi_k(G^*_{k-1}(Y_{t,k-1}, U_{t,k-2}), U_{t,k-1})],$$

and from here it follows that

$$y(t+k) = f(Y_{t,k-1}, U_{t,k-1}),$$

where f is a nonlinear function.

This means that $\max(\text{rank}(D_x G_{t,l}))$ must be n, which is equivalent to the linear case where a realisable system must have a $[CA]^l_t$ matrix of maximum rank equal to n for any l and t. Of course this condition is satisfied by a system that can be described by state space equations in a finite dimensional space.

The other condition is that the system can be approximated around an equilibrium point by the linearised system at that point of operation. This is translated to the following conditions on $G_{t,l}$: there exist k and t, such that the derivative $D_x G_{t,l}(x(t), U_{t,l})$ has rank n.

These conditions mean that $x(t)$ can be solved in terms of the output and input values, the system must be finite time observable if the input-output equation is to exist.

Theorem 10 *Let the nonlinear system (4.3) satisfy the following conditions*

a) $\max(\text{rank}(D_x G_{t,l})) = n$ *for* $U_{t,l} \in U^l, x(t) \in X$ *and for any* $l = 1, 2, \ldots$ *and* $t = 1, 2, \ldots$*;*

b) $\text{rank}(D_x G_{t,l}(x(t), U_{t,l})) = n$ *for some* t *and some* l*.*

Then there exists a nonlinear function q*, such that*

$$y(t+l) = q(Y_{t,l}, U_{t,l-1})$$

for a restricted region of operation around the $(x(t), U_{t,l-1})$ *point.*

Proof:

Let the response vector $Y_{t,l}$ be

$$Y_{t,l} = G_l(x(t), U_{t,l-1}),$$

applying condition *a*) the system has a finite representation, and for some k and t, $x(t)$ can be solved in terms of $Y_{t,l}$ and $U_{t,l-1}$. Applying condition *b*) and the Implicit Function Theorem, it follows that there exists a continuous function $G^* : Y^{n-1} \times U^{n-1} \rightarrow X$, such that

$$x(t) = G^*(Y_{t,n-1}, U_{t,n-2}),$$

and to every $Y_{t,l}$ and $U_{t,l-1}$ there corresponds a unique $x(t)$ such that

$$y(t+n) = h[\Phi_n(G^*(Y_{t,n-1}, U_{t,n-2}), U_{t,n-1})]$$

Hence the result. $\diamond\diamond$

There are some special structures commonly used to represent the nonlinear system, which admit an input-output model, among them Wiener models and Hammerstein models.

The Wiener model consists of a cascade of a linear system followed by a static nonlinearity, this model can be described by the following equations

$$\begin{aligned} x(k) &= z^{-d}\tfrac{B(z)}{A(z)}u(k), \\ y(k) &= f(x(k)), \end{aligned} \tag{4.15}$$

where $B(z)$ is a polynomial defined as $B(z) = b_1 z^{-1} + \ldots + b_m z^{-m}$, and $A(z)$ is defined as $A(z) = 1 + a_1 z^{-1} + \ldots + a_n z^{-n}$. If the inverse of f exists then it is possible to replace $x(k)$ with $f^{-1}(y(k))$ in the equation above to get

$$\begin{aligned} y(k) &= f(a_1 f^{-1}(y(k-1)) + \ldots + a_n f^{-1}(y(k-n))) \\ &+ b_1 u(k-1-d) + \ldots + b_m u(k-m-d). \end{aligned} \tag{4.16}$$

This input-output representation is globally valid if the nonlinearity is an invertible function.

A Hammerstein model consists of a cascade of a static nonlinearity followed by a linear system, as is described by the following equations

$$\begin{aligned} u^*(k) &= f(u(k)), \\ y(k) &= z^{-d}\tfrac{B(z)}{A(z)}u^*(k). \end{aligned} \tag{4.17}$$

In this case the input-output representation is given by

$$y(k) = z^{-d}\frac{B(z)}{A(z)}f(u(k))$$

and is valid globally.

The combination of nonlinearity and dynamic system cannot always be represented by finite input-output maps, even if the nonlinearity is invertible [46], [64]. The following example of a real mechanical system will illustrate a case. A drive system with backlash in the mechanical transmission, is described by

$$\begin{aligned} J_1\ddot{\varphi}_1 + f_1\dot{\varphi}_1 &= M_m - M, \\ J_2\ddot{\varphi}_2 + f_2\dot{\varphi}_2 &= M, \end{aligned} \tag{4.18}$$

with M defined as

$$\begin{aligned} M =& \ k_y\delta, \\ \delta =& \begin{cases} \varphi_1 - \varphi_2 - \epsilon & \text{for } \varphi_1 - \varphi_2 > \epsilon \\ 0 & \text{for } |\varphi_1 - \varphi_2| \le \epsilon \\ \varphi_1 - \varphi_2 + \epsilon & \text{for } \varphi_1 - \varphi_2 < -\epsilon \end{cases} \end{aligned} \tag{4.19}$$

The block diagram for this system is shown in Figure 4.1

The aim is to find an expression like $\varphi(k) = f(\varphi(k-1), \ldots, \varphi(k-n), M_m(k-1), \ldots, M_m(k-m))$. Obviously the system operating in different regions admits a finite representation, but the problem lies in the fact that some of the states are not observable in the dead zone region.

4.2 System Concepts

In order to design a control system it is necessary to consider the key concepts of observability, controllability, and invertibility. The observability condition is of a

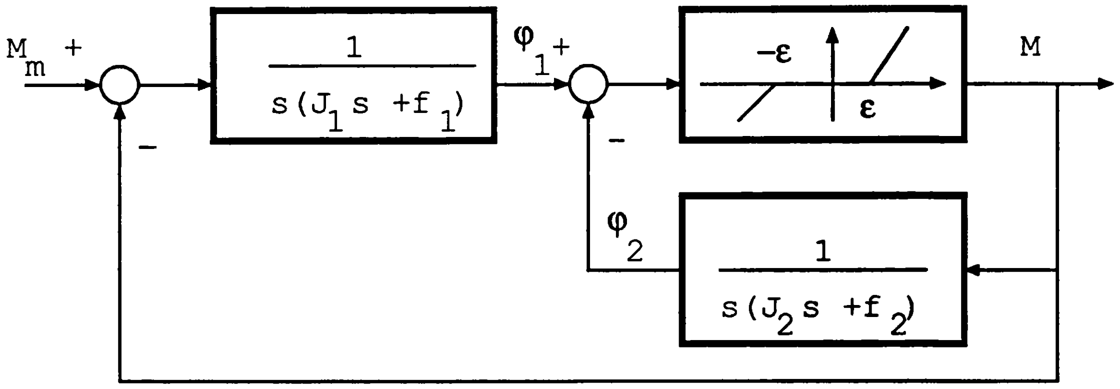


Figure 4.1: Drive system with backlash.

paramount importance, because when we are using the input-output model this condition is implicit in the model. As the input-output models are used in the development of the control algorithm, the observability condition is not imposed as a requirement for the control design because it is implicit in the model. On the other hand, the controllability condition is needed, as presented here, to prove some of the theorems stated in the next Chapter.

Even though some of these concepts are widely known and used in linear systems their nonlinear counterparts are less known and more difficult to test, as described in the following sections.

4.2.1 Observability

Two states x_1 and x_2 are called *indistinguishable* if for any sequence $w \in U^+$ it holds that $f_{x_1} = f_{x_2}$. That means that no input-output experiment can determine which of the two states the system is at.

A system is called *observable* when there are no indistinguishable states in the set state X of the system.

A state x_0 is observable at t_0 if, given any control u, there is a time $t_1 > t_0$, such that knowledge of $u_{(t_0,t_1]}$ and the output $y_{(t_0,t_1]} = \hat{g}(x_0, u_{(t_0,t_1]})$ is sufficient to determine x_0.

For a linear SISO time invariant system it is possible to find a relation between the structural properties of the system (or the algebraic properties of the pair of matrices $\{A, C\}$) and observability. This relation is given by the observability matrix defined as $[C, CA, \ldots, CA^{n-1}]^T$ which must have rank n for the system to be completely observable.

If every state x_0 is observable at every time t_0 in the interval of definition of the system, then we say that the system is *completely observable.*

For a linear time-varying system, the conditions are translated into the definition of uniformly completely controllable. A time-varying system defined by the pair $\{A_k, B_k\}_{k\geq 0}$ is uniformly completely controllable if exists a $\mu_o > 0$ and a positive integer N_o, such that

$$\sum_{j=1}^{i+N_o-1} \Phi(i+N_o, j+1)C_j^T C_j \Phi(i+N_o, j+1) \geq \mu_o; \quad i > 0 \quad x \in R^n.$$

A time invariant system is uniformly completely observable if and only if it is completely observable [42].

It is convenient at this point introduce the function class

$$\mathcal{M} := \{f : R \rightarrow R | \; f \text{ monotone increasing homeomorphism of } R^+ \rightarrow R^+\}.$$

For nonlinear systems it is not possible in general to find conditions on the structure of the system relating it to observability without imposing certain condition like differentiability and continuity, which are usually restrictive. A more general notion of observability is given by a relation between the norm of the input-output sequences and the norm of the initial state [39]. System (4.3) is said to have *property O* if there exists a positive integer N_0 and an $\mathcal{M}$-function W_0, with $W_0(0) = 0$, such that given any $i > 0$ and sequence triple $\{(x(k), y(k), u(k))\}_{k>i}$ that satisfies (4.3) and the following holds

$$\sum_{k=i}^{i+N_0-1} \|(y(k), u(k))\| \geq W_0(\|x(i)\|). \tag{4.20}$$

The $u(k)$ appears in (4.20) because of the nonlinearities of g and h in equation (4.3). This allows to account for the fact that the $u(k)$ might be chosen in such a way that the $y(k)$ are small independent of $x(i)$. For $u(k) = \mathbf{0}, k \geq i$, the property O is an uniform observability property [39]; that is observing the outputs $y(k)$ for $i \leq k \leq i + N_o - 1$, it is possible determine a bound on the size of the initial state $x(i)$.

4.2.2 Controllability

The term controllability means that it is possible to drive any state of the system to the *origin* in a finite time, using bounded piecewise continuous controls.

A state x is said to be *reachable* from the state $x(0)$ if there exists an input sequence $w \in U^*$ such that $x = \Phi_n(x(0), w)$. If the state $x_1 = 0$ is reachable from x_0 at t_0, then x_0 is *controllable* at t_0. In particular, the subset X is (n-step, globally) *controllable* (to zero) if for each state x in X there is an input sequence w of length at most n for which $\Phi_n(x, w) = 0$.

If every state x_0 is controllable at t_0, then we say that the system is *completely controllable* at t_0.

The definition of controllability is set up in such a way that it is an obvious necessary and sufficient condition for the design of a regulator. The task of a regulator is to move an arbitrary initial state x of Σ to a fixed desired state x, assumed to be zero.

For a linear SISO time invariant system it is possible to find a relation between the structural properties of the system (or the algebraic properties of the pair of matrices $\{A, B\}$) and the controllability of Σ. This relation is given by the controllability matrix defined as $[B, AB, \ldots, A^{n-1}B]$ which must have rank n for the system to be completely controllable.

For a linear time-varying system, the conditions are translated into the definition of uniformly completely controllable. A time-variant system defined by the pair $\{A_k, B_k\}_{k\geq 0}$ is uniformly completely controllable if there exists a $\mu_c > 0$ and a positive integer N_c, such that

$$x(t)^T(\sum_{i=1}^{i+N_c-1} \Phi(i+N_c, j+1)B_jB_j^T\Phi(i+N_c, ji+1))^Tx(t) \geq \mu_c\|x\|^2; \;\; i > 0 \;\; x \in R^n.$$

A time invariant system is uniformly completely controllable if and only if it is completely controllable [42].

And finally, for a nonlinear system we have the notion of property C.

The system (4.1) is said to have the property C if there exists a positive integer N_c and an $\mathcal{M}$-function W_c, with $W_c(0) = 0$, such that given any $i \geq 0, a \in R^n$ there exists a sequence pair $\{(x(k), u(k))\}_{k=1}^{i+N_c-1}$ which satisfies (4.1), $x(i) = a, \;\; x(i + n_c) = 0$ and

$$\sum_{k=1}^{i+N_c-1} \|(x(k), u(k))\| \leq W_c(\|a\|).$$

Under this definition with $W_c(s) = p_cs, \; p_c > 0$, it is possible to see that if a time-varying linear system is uniformly completely controllable, then it satisfies the property C [39].

4.2.3 Invertibility

The inversion of nonlinear operators plays a central role in the development of nonlinear controllers. Here we explore the invertibility of nonlinear dynamic systems represented by (4.1).

Suppose two systems S_1 and S_2 are given, such the output space Y_1 of the first equals the input space U_2 of the second system. Then the output lines of S_1 to the input lines of S_2 are connected to obtain the *series* connection S_s. To specify the state of S_s it is necessary to specify the state of each S_1 and S_2, thus $X_s = X_1 \times X_2$. If S_1 is in state x_1 and S_2 is in state x_2 then $x_s = [x_1^T|x_2^T]^T$ denotes the state of S_s. Defining

$$\begin{aligned} S_1: \;\; x_1(t+1) = & \;\; f_1(x_1(t), u_1(t)) \\ y_1 = & \;\; h_1(x_1(t)), \end{aligned} \tag{4.21}$$

and

$$S_2: \begin{aligned} x_2(t+1) &= f_2(x_2(t), u_2(t)) \\ y_2 &= h_2(x_2(t)), \end{aligned} \tag{4.22}$$

then

$$S_s: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1(x_1(t), u_1(t)) \\ f_2(x_2(t), h_1(x_1(t))) \end{bmatrix}, \quad y_s = h_2(x_2(t)). \tag{4.23}$$

A SISO discrete system S is said to be *k-right invertible* (*k-RI*) around $u(k)$ if there exists a system, say SI, such that the series connection $S \circ SI$ is equivalent to the k delays for any state of S and any $u \in \Omega_{u(k)}$, where $\Omega_{u(k)}$ is a suitable neighbourhood of $u(k)$.

S is said to be *right invertible* if the above mentioned property holds true around any $u(k)$.

The existence of the inverse of a system is related with the concept of reachability. In fact, we can view the problem as finding an input sequence, which drives the system S_2 to the desired state, which represents the input of another system connected in series with S_1. The formalisation of this idea is given by the following theorem.

Theorem 11 *The system S is k-RI if and only if, for each state x reachable from x_0, and for each pair of input sequences $w = u(0)\ldots u(k)$ and $w^* = u(0)^* \ldots u(k)^*$, the equality $\Phi(x, w) = \Phi(x, w^*)$ implies that $u(0) = u^*(0)$.*

Proof : [83].

For an input-output description the system Σ is *invertible* if there exists a system ΣI satisfying $\Sigma \circ \Sigma I = I$, and ΣI is called the generalised right inverse.

The structure defined by the input-output representation Σ assumes that the system is observable and reachable, and this leads to the following result.

Theorem 12 *A recursive system $\Sigma : R^n \times R^m \rightarrow R$ has a recursive RI $\Sigma : R^n \times R^m \rightarrow R$.*

Proof:[31]

The idea motivating the introduction of the right inverse of the system is to get the control signal $u(t)$ to reach a desired value of the output. That is if the system is defined as

$$y(k+1) = f_2(y(k)\dots, y(k-n), u_2(k), \dots, u_2(k-m)),$$

and the inverse f_1 is defined by

$$u_2(k) = f_1(y(k)\dots, y(k-n), u_2(k-1), \dots, u_2(k-m), y(k+1)),$$

the series connection is reduced to the composition of two static nonlinear functions. Obviously, $y(k+1)$ can be replaced by a reference signal, and in this way a 1-RI is obtained. Notice that the inverse is a recursive relation on $u_2(k)$, as it was stated by Theorem 4. In Chapter 5 a general discussion about the stability of this recursive relation is presented.

The inverse in this case depends on the past output of the plant, the desired value of the output and past values of the controller. For this configuration f_2 is 1-RI invertible if the function f_2 is invertible with respect to $u(k)$. If the recursive system $\Sigma : R^n \times R^m \to R$ is a bijection, then its inverse map is uniquely determined.

The system Σ is called *singular* if for any

$$[y^p(k), \dots, y^p(k-n), u(k-1), \dots, u(k-m)]^T \in A$$

, and for any distinct inputs $u^1(k)$, $u^2(k)$, the resulting outputs are equal:

$$\begin{aligned} f(y^p(k), \dots, y^p(k-n), u^1(k), \dots, u(k-m)) &= \\ f(y^p(k), \dots, y^p(k-n), u^2(k), \dots, u(k-m)). & \end{aligned} \tag{4.24}$$

A particular case is presented when the system is monotonic with respect to the control variable.

Theorem 13 *If $f(y^p(k),\ldots,y^p(k-n),u(k),\ldots,u(k-m))$ is monotonic with respect to $u(k)$ then the system is 1-RI $[y^p(k),\ldots,y^p(k-n),u(k-1),\ldots,u(k-m)]^T$ for all $u(k)$.*

Proof:

If $f(y^p(k),\ldots,y^p(k-n),u(k),\ldots,u(k-m))$ is monotonic and $u^1(k) > u^2(k)$ then for the same point $[y^p(k),\ldots,y^p(k-n),u(k-1),\ldots,u(k-m)]^T$

$$f(y^p(k),\ldots,y^p(k-n),u^1(k),u(k-1),\ldots,u(k-m)) >$$
$$f(y^p(k),\ldots,y^p(k-n),u^2(k),u(k-1),\ldots,u(k-m))$$

In the same way,

$$f(y^p(k),\ldots,y^p(k-n),u^1(k),u(k-1),\ldots,u(k-m)) <$$
$$f(y^p(k),\ldots,y^p(k-n),u^2(k),u(k-1),\ldots,u(k-m))$$

if $u^1(k) < u^2(k)$. The system is therefore invertible. $\diamond\diamond$

Clearly, 1-RI is related very much to the global characteristic of the function f. In general it is possible to use arguments of the inverse function theorem to arrive to the same conclusion [73]. In many cases the conditions of invertibility can be satisfied just locally.

4.3 Identification

In Chapter 3 we have outlined the architectures which can be used for approximating nonlinear functions, and in Section 4.1 we have shown that a nonlinear system can be represented by a nonlinear function of the past values of the output and input.

In this section, the learning algorithms used for training the networks are presented, i.e. the algorithms necessary to adjust the parameters of the network in order to approximate a dynamic system. Even though the performance of the algorithms presented in this section is not as good as more sophisticated algorithms they provide a base to outline the basic problems and potentials in this area. A note of caution is necessary before starting the presentation of the algorithms, the basic requirement for a system to have a representation using a network is that it must be described by an *injective* function. Obviously functions such as the solution of $x^2 = y$ cannot represented by a network such as the gaussian network. As pointed out by Sontag [84] a solution (nonunique) to this problem requires the use of discontinuous mappings, and these cannot be approximated as a superposition of maps which are constant on halfspaces or which have kernel characteristics. In this section inverse means 1-RI inverse.

4.3.1 Modelling a Dynamic System

The dynamic system is modelled using a gaussian network, which represents the recursive input-output model of the system by

$$y^m(k+1) = \sum_{i=1}^{N^m} c_i^m g_{\mu_i,\sigma}^m(x^m(k)). \tag{4.25}$$

Here, the superscript m indicates the system's model. We choose the structure of the system model to be the same as that of the plant (4.1), i.e. the model output is a nonlinear function of the present and past system outputs, and the present and past system inputs. The model input vector $x^m(k)$ is thus given as

$$x^m(k) = [y^p(k), \ldots, y^p(k-n), u(k), \ldots, u(k-m)]^T,$$

where the superscript p indicates the plant.

We denote the centre of the Gaussian function of hidden unit i as

$$\mu_i^m = [y_{i,0}, \ldots, y_{i,n}, u_{i,0}, \ldots, u_{i,m}]^T.$$

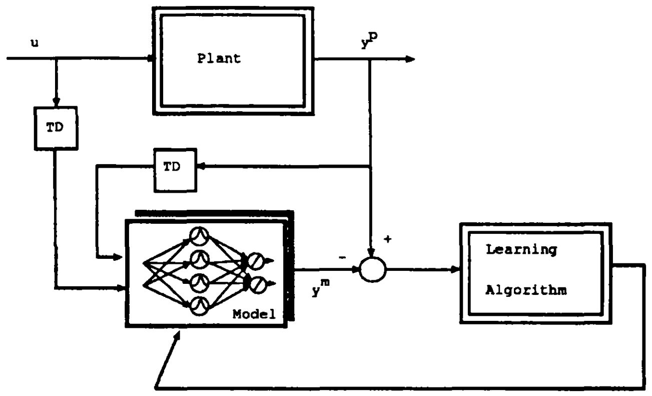


Figure 4.2: Plant identification structure.

The parameters μ_i^m and σ^m are adjusted to meet the interpolation conditions, i.e. the μ_i^m are distributed uniformly over the input space and σ^m is adjusted such that $\sum_{i=1}^{N^m} g_{\mu_i,\sigma}^m(x^m(k))$ =const over the input space. There are other possibilities as explained in Chapter 3.

The parameter vector c_i^m is adjusted to minimise the mean square error between the real plant and the model. That is,

$$c_i^m(k+1) = c_i^m(k) + \alpha g_{\mu_i,\sigma}^m(x(k))(y^p(k+1) - y^m(k+1)), \qquad (4.26)$$

where, α is a gain parameter. Figure 4.2 represents the structure used, where the TD block represent tapped delay lines.

Using standard linear systems theory, as described in Chapter 3, it can be shown that if the function f is continuous, then it can be approximated with any degree of accuracy by (4.25), given the appropriate number of units, and the least mean square solution can be found by (4.26) [80].

4.3.2 Inverse Model Identification

If the model of the plant is invertible, then the inverse of the plant can be approximated in a similar way to the plant. This model can then be used as the controller. A second network described by

$$u(k) = \sum_{i=1}^{N^I} c_i^I g_{\mu_i,\sigma}^I(x^I(k)) \tag{4.27}$$

is utilised. Here, the superscript I indicates the inverse. The inverse of the function f in Equation (4.1) (required to obtain $u(k)$) depends upon the future plant output value $y^p(k+1)$. In order to obtain a realisable approximation we replace this value by the desired input value r. Finally, we define the inverse network input vector $x^I(k)$ as

$$x^I(k) = [y^m(k), \ldots, y^m(k-n), r(k+1), u(k-1) \ldots, u(k-m)]^T.$$

Here, the future value $r(k+1)$ is obtained at time k . The centre of the Gaussian function of hidden unit i is given by

$$\mu_i^I = [y_{i,0}, \ldots, y_{i,n}, r_i, u_{i,1} \ldots, u_{i,m}]^T.$$

Non-iterative Methods

To adjust c_i^I the architecture shown in Figure 4.3 is used.

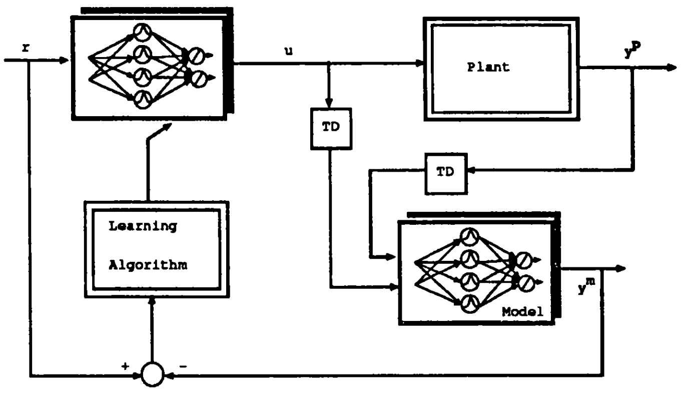


Figure 4.3: Using the model to identify the plant inverse.

This architecture is similar to the specialised learning architecture presented by Psaltis *et al* [70] (the difference being that here we use the plant model, rather than the plant itself). The parameters in c_i^I are adjusted to minimise the mean square error between the output of the model and the input of the controller. This leads to the following learning algorithm:

$$c_i^I(k+1) = c_i^I(k) + \alpha g_{\mu_i,\sigma}^I(x^I(k))(r(k+1) - y^m(k+1))\frac{\partial y^m(k+1)}{\partial u(k)}. \quad (4.28)$$

Here, if the real plant were used in the learning procedure then $\frac{\partial y(k+1)}{\partial u(k)}$ would have to be estimated. This can be done using first order differences [70] changing each input to the plant slightly at the operating point and measuring the change at the output. By using the plant model, however, the derivatives can be calculated explicitly. From Equation (4.25) we obtain

$$\frac{\partial y^m(k+1)}{\partial u(k)} = -2\sigma^m \sum_{i=1}^{N^m} c_i^m g_{\mu_i,\sigma}^m(x^m(k))(u(k) - u_{i,0}). \quad (4.29)$$

Theorem 14 *The learning algorithm defined by Equation (4.28) converges to a global minimum of the index defined by*

$$J = \sum_j (r(j+1) - y^m(j+1))^2, \quad (4.30)$$

if the system is monotonically increasing with respect to $u(k)$, where $r(k)$ is a bounded sequence corresponding to the desired output of the system.

Proof:

Consider the mean square error defined over all the possible inputs $r(k)$ described by (4.30). Its change ΔJ under an adaptation step (4.28) resulting from an input r^* is

$$\Delta J(r^*) = -2\sum_j (r(j+1) - y^m(j+1)) \sum_i \frac{\partial y^m(j+1)}{\partial u(j)} g_{\mu_i,\sigma}^I(x^I(j))\Delta c_i^I$$

Inserting (4.28) and averaging over all inputs r_k^*

$$\begin{aligned} < \Delta J > \; &= \; -2\alpha \sum_l \sum_j (r(j+1) - y^m(j+1))(r^*(l+1) - y^{m*}(l+1)) + \\ & \sum_i \frac{\partial y^m(j+1)}{\partial u(j)} g_{\mu_i,\sigma}^I(x^I(j)) \frac{\partial y^{m*}(l+1)}{\partial u(l)} g_{\mu_i,\sigma}^I(x^{I*}(l)). \end{aligned}$$

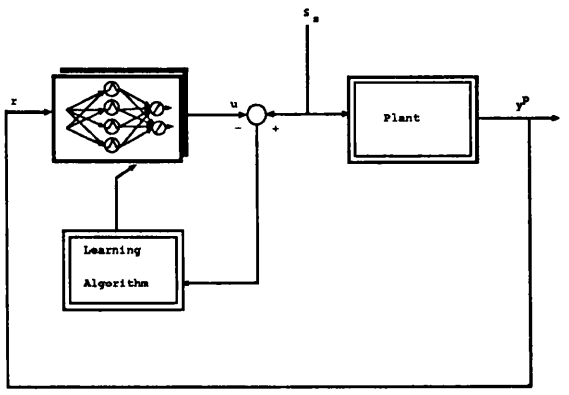


Figure 4.4: Use of the synthetic signal to identify plant inverse.

$$< \Delta J >= -2\alpha \sum_i [\sum_l \frac{\partial y^m(l+1)}{\partial u(l)} g^I_{\mu_i,\sigma}(x^I(l))(r(l+1) - y^m(l+1))]^2 \qquad (4.31)$$

Clearly, this quantity can only be negative or zero. For a monotonic function it cannot be zero. From (4.31), if the function is not monotonic the algorithm can reach a local minimum when $\frac{\partial y}{\partial u}$ is zero. $\diamond\diamond$

If the sign is changed in (4.29), the same result follows for decreasing functions.

Another approach involves the use of a synthetic signal [90]. This leads to the so called general learning architecture as shown in Figure 4.4.

In this case the adaptation law for the weights does not depend on the derivatives of the plant:

$$c^I_i(k+1) = c^I_i(k) + \alpha g^I_{\mu_i,\sigma}(x^I(k))(S_s - u(k)). \qquad (4.32)$$

Here, S_s is the synthetic signal.

Theorem 15 *If the system is 1-RI, the algorithm defined by Equation (4.32) converges to the best approximation of the inverse in the least square sense under the conditions of Theorem 2, Chapter 3.*

Proof:

If the system is 1-RI then there exists an injective mapping which represents the inverse. Thus, by Theorem 2 in Chapter 3 the algorithm defined by (4.32) converges to the least squares error [80]. $\diamond\diamond$

Iterative Methods

Iterative methods make use of a plant model to calculate the inverse. In this case, a recursive method is used to find the inverse of the model at each operating point. This method is useful for systems which satisfy the invertibility conditions, only locally and not for the whole operating space. This approach can also be used when it is necessary to have small networks due to memory or processing limitations. In this case, the restricted accuracy of the trained network can be enhanced by using the network to provide stored initial values for the iterative method, establishing a compromise between speed of convergence and storing capacities.

At time k, the objective is to find an input u which will produce a model output $y^m(k+1)$ equal to $r(k+1)$. It is possible to use the method of successive substitution:

$$u^{n+1}(k) = u^n(k) + \gamma(r(k+1) - y^m(k+1)),$$

where γ is a weight to be chosen.

According to the small gain theorem [17], the inverse operator is input-output stable if the product of the operator gains in the loop is less than 1

$$\|I\|\|I - \gamma f\| \leq 1.$$

The initial value $u^0(k)$ can be stored in a connectionist network.

Practical Issues

During the design of an identification set-up not only some of the parameters of the model, like the order of the system, must be selected beforehand, but also the experimental conditions necessary to extract the information of the system. This means a sequence of inputs that is both sufficiently varied to reveal the function's form and sufficiently repetitive to allow the convergence of the learning algorithm. This leads to the concept of *persistency of excitation* .

Also, the noise and its characteristics must be considered during the modelling stage. If a measurement noise is included, the model is described by the following equations

$$\begin{aligned} y(t) = & \; y^s(t) + e(t), \\ y^s(t) = & \; f(y^s(t-1), \ldots, y^s(t-n), \\ & \; u(t-1), \ldots, u(t-m)), \end{aligned} \tag{4.33}$$

but if the noise enters to the system, the model is given by the following recursion

$$\begin{aligned} y(t) = & \; f(y(t-1), \ldots, y(t-n), \\ & \; u(t-1), \ldots, u(t-m), \\ & \; e(t-1), \ldots, e(t-l)) + e(t). \end{aligned} \tag{4.34}$$

The effect of measurement noise upon the parameter identification experiment must also be considered. For linear systems it is known that sensitivity of the identification process is governed by the degree of excitation present in the input signal. In the models described in the last section, where a linear in the parameters model was used, the noise immunity or convergence rate will diminish in direct proportion to an increase in the condition number of the matrix defined as the persistent excitation matrix $\mathcal{P}$.

Any identification procedure should consider a model validation stage, which should reveal the model's deficiencies. The validation is done with data that has not been used during the identification process, but it is contained within the boundaries of the training set. For prediction models the validation is done over the residual error, that is if the model structure components are correct then the prediction error sequence should be nodeterministic for all linear and nonlinear combinations of past inputs and outputs [12].

Using the series-parallel structure, Figure 4.5, the output of the system is available to identify the parameters, this means that the model is updated with the past output of the real plant.

In this case effectively a one step ahead predictor is identified, that is the error obtained with this algorithm refers just to one step ahead error.

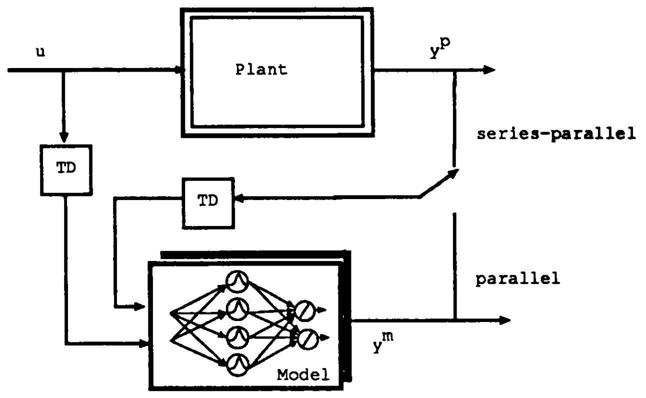


Figure 4.5: The series-parallel and parallel structures.

Under recursive operation of series-parallel model the propagation of the error is over the nonlinear system [30], and therefore its potentials diminish over the prediction horizon, and even worse the model can drift to a different operational region.

Another alternative is the use of parallel structures like the one shown in Figure 4.5. In this case, the model has its own feedback. A different training algorithm must be used to train this network, the concepts of Optimal Control theory play a key role [95]. Once the network is trained over the whole set of possible inputs and initial conditions, the network can be connected in parallel with the system. The disadvantage of using this kind of structure in a nonlinear context is the effect of the initial conditions. In fact as the dynamics of the system depends on the initial condition there exists the posibility that the error betweeen the output of plant and the model increases without bound, even though the model is perfect. On the other hand, when the model and the plant are asymptoticaly stable the effect of the initial condition dies as the time runs.

The error in this case is related to the trajectories, i.e. the solution of a differential equation, of the systems and the model, but in a series-parallel model the error is related to the approximating function.

Chapter 5

Control Structures

SUMMARY

Four approaches to control a nonlinear system, given its connectionist representation, are described. Each approach is analysed and a basic example illustrating the design procedure is provided. Some of the properties are proved and the limitations of each scheme are pointed out. A summary of the methods is given with their main potentials and limitations.

5.1 Direct Inverse Configuration

The objective of the design procedure is to obtain a nonlinear operator, which represents the inverse of the plant. To obtain this operator the ideas described in Chapter 4 can be followed. An off-line procedure which utilises a copy of the plant model, established in the way outlined in Chapter 4, is used. To achieve control of the plant the model-inverse is simply cascaded with the plant. This forces the dynamic response between the reference signal and the plant output to equal the desired reference model, by cancelling out the dynamics and nonlinearities.

As was outlined in Chapter 4 a major problem with inverse identification arises when many inputs produce the same output, i.e. the plant's inverse is not well-defined. In this case, the network will attempt to map the same network input to many different target responses.

The implementation of the inverse can be done with two structures: the feedback structure, which is obtained when the past values of the output of the plant are used to generate the control signal, and the feedforward structure obtained when the past values of the reference signal are used. Figure 5.1 shows the architectures in both cases for a linear plant.

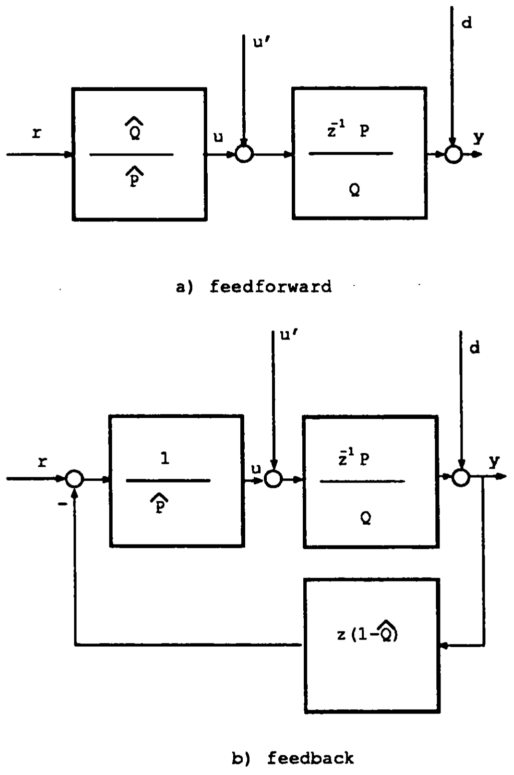


Figure 5.1: Inverses of a linear plant: a) feedforward, b) feedback.

In order to illustrate the differences between these two structures a discrete-time linear system described by the following transfer function

$$\frac{y}{u} = \frac{z^{-1}P(z^{-1})}{Q(z^{-1})}$$

is used, where $P(z^{-1})$ is polynomial such that $P(0) \neq 0$ and $deg(P) = m$ and $Q(z^{-1})$ is a polynomial such that $Q(0) = 1$, $deg(Q) = n$, and $m \leq n$.

In this case the transfer function of the feedforward inverse is given by

$$\begin{pmatrix} y \\ u \end{pmatrix} = \begin{pmatrix} \frac{z^{-1}\hat{Q}(z^{-1})P(z^{-1})}{\hat{P}(z^{-1})Q(z^{-1})} & \frac{z^{-1}P(z^{-1})}{Q(z^{-1})} \\ \frac{\hat{Q}(z^{-1})}{\hat{P}(z^{-1})} & 0 \end{pmatrix} \begin{pmatrix} r \\ u' \end{pmatrix}, \tag{5.1}$$

where $\hat{P}(z^{-1})$ and $\hat{Q}(z^{-1})$ are the estimated values of $Q(z^{-1})$ and $P(z^{-1})$. If $\hat{P}(z^{-1}) = P(z^{-1})$ and $\hat{Q}(z^{-1}) = Q(z^{-1})$, the error due to different initial conditions (u') follows the dynamic of $Q(z^{-1})e = 0$, where $e(t) = r(t) - y(t)$. If the system is stable this effect dies out as $t \to \infty$. Any disturbance in the output of the system does not affect any other signal in the system.

The case when the output of the system is used to generate the inverse, the response of the system is given by

$$\begin{pmatrix} y \\ u \end{pmatrix} = \begin{pmatrix} \frac{z^{-1}P(z^{-1})}{(1-\hat{Q}(z^{-1}))P(z^{-1})+\hat{P}(z^{-1})Q(z^{-1})} & \frac{z^{-1}\hat{P}(z^{-1})P(z^{-1})}{(1-\hat{Q}(z^{-1}))P(z^{-1})+\hat{P}(z^{-1})Q(z^{-1})} \\ \frac{Q(z^{-1})}{(1-\hat{Q}(z^{-1}))P(z^{-1})+\hat{P}(z^{-1})Q(z^{-1})} & \frac{P(z^{-1})(1-\hat{Q}(z^{-1}))}{(1-\hat{Q}(z^{-1}))P(z^{-1})+\hat{P}(z^{-1})Q(z^{-1})} \end{pmatrix} \begin{pmatrix} r \\ u' \end{pmatrix}. \tag{5.2}$$

In this case the closed loop is stable if $(1 - \hat{Q}(z^{-1}))P(z^{-1}) + \hat{P}(z^{-1})Q(z^{-1})$ is stable, and the effect of the disturbance, is given by

$$\frac{y}{d} = Q(z^{-1}).$$

In both cases if the system is non-minimum-phase its inverse is unstable. For a linear system by checking the zeros of the transfer function it is possible to see if the system is minimum-phase or not. Generalising the notion of zeros for a nonlinear system is not a trivial problem. The linear transfer function cannot be defined globally for nonlinear systems. As a first approximation, the extended linearisation over the recursive equation f can be used, and then the zeros of the linearised system under different operational points can be investigated.

Another concept that has proved to be useful, especially for affine systems, is zero-dynamics, which is defined to be the internal dynamics of the system when the system output is set to zero by the input. In linear system the poles of the zero dynamic are exactly the zeros of the system. Similarly to the linear case a system whose zero dynamics is asymptotically stable is called asymptotically minimum-phase system. The zero dynamic is an intrinsic property of a nonlinear system. If the zero dynamic is stable then the inverse of the system is stable around the origin. In fact, using the following form of the recursive function Σ

$$x(t) = [y(t-n)\dots, y(t), u(t-m), \dots, u(t-1)]^T,$$

$$x(t+1) = \begin{bmatrix} & & 0 & \dots & 0 \\ & I_n & \vdots & & \vdots \\ 0 & \dots\ 0 & 0 & \dots & 0 \\ 0 & \dots\ 0 & & & \\ \vdots & \ \ \vdots & & I_{m-1} & \\ 0 & \dots\ 0 & 0 & \dots & 0 \end{bmatrix} x(t) + \begin{bmatrix} \vdots \\ 0 \\ f(x(t), u(t)) \\ \vdots \\ 0 \\ u(t) \end{bmatrix}, \qquad (5.3)$$

the control $u(t) = f^{-1}(x(t), y(t+1))$ linearises the system. The zero dynamic is defined as the strongly unobservable dynamic when this control signal is used. The zero dynamic is obtained setting $y(t+1)$, $x_i(t)$, $i = 1, \dots, n$, constrained to zero

$$\begin{aligned} x_i(t) &= 0, \quad i = 1, \dots, n, \\ x_{n+1}(t) &= x_{n+2}(t), \\ &\vdots \\ x_{n+m} &= f^{-1}(x(t), 0). \end{aligned} \qquad (5.4)$$

If this system is stable then the system is asymptotically minimum-phase.

Another problem arises when the output of the system is constrained for the non-linearities of the system and the space defined by the inputs [44]. This means that there is a feasible region for the output transition ($FOTR$), which is defined by

$$FOTR = \{(y(t+1), \bar{y}(t)) \ \ |y(t+1) = f(\bar{y}(t), \bar{u}(t)) \text{ and } \bar{u}(t) \in U \subset R^{m+l}\},$$

where $\bar{y}(t) = [y(t), \dots, y(t-n)]^T$, and $\bar{u}(t) = [u(t), \dots, u(t-m)]^T$. In Chapter 6 further implications of $FOTR$ are described.

These issues signal the nontrivial problems of building inverse operators for nonlinear dynamical systems.

In an open-loop system, however, the disturbances, noise and approximation errors appear directly at the plant output. Their effects can be minimised if the adaptive procedure is operating with the system or an extra feedback is added.

The first approach considers the learning algorithm as a compensator, in this case the effects of the disturbances are compensated through the adaptation loop. The second approach includes a linear controller in cascade with the inverse, here the nonlinear inverse acts as a linearising transformation.

5.1.1 Simulations

The plant to be considered in these examples is described by

$$y(t+1) = f_1(y(t)) + f_2(u(t)) = \frac{y(t)}{1+y(t)^2} + u(t)^3. \tag{5.5}$$

The system is monotonic with respect to $u(t)$ and therefore invertible. This system was previously considered by Narendra and Parthasarathy [61].

Using a synthetic signal and an architecture of one network $\mathcal{N}$, see section 4.3 Chapter 4, with 2 inputs and 441 units and a training region defined over the interval $y(t) \in [-9, 9]$ and $u(t) \in [-2, 2]$, it is possible to calculate the inverse of the system as

$$u(t) = \mathcal{N}(y(t), y(t+1)),$$

where after training $y(t+1)$ is replaced by the desired output value.

Figure 5.2 shows the final values of the parameters for this regular network with 441 units, and the response of the system for a sinusoidal input.

Using the prior information one can decompose the system into two parts:

$$y^m(t+1) = \mathcal{N}_1(y(t)) + \mathcal{N}_2(u(t)),$$

where $\mathcal{N}_1$ and $\mathcal{N}_2$ represent regular gaussian networks with 40 and 20 units, which

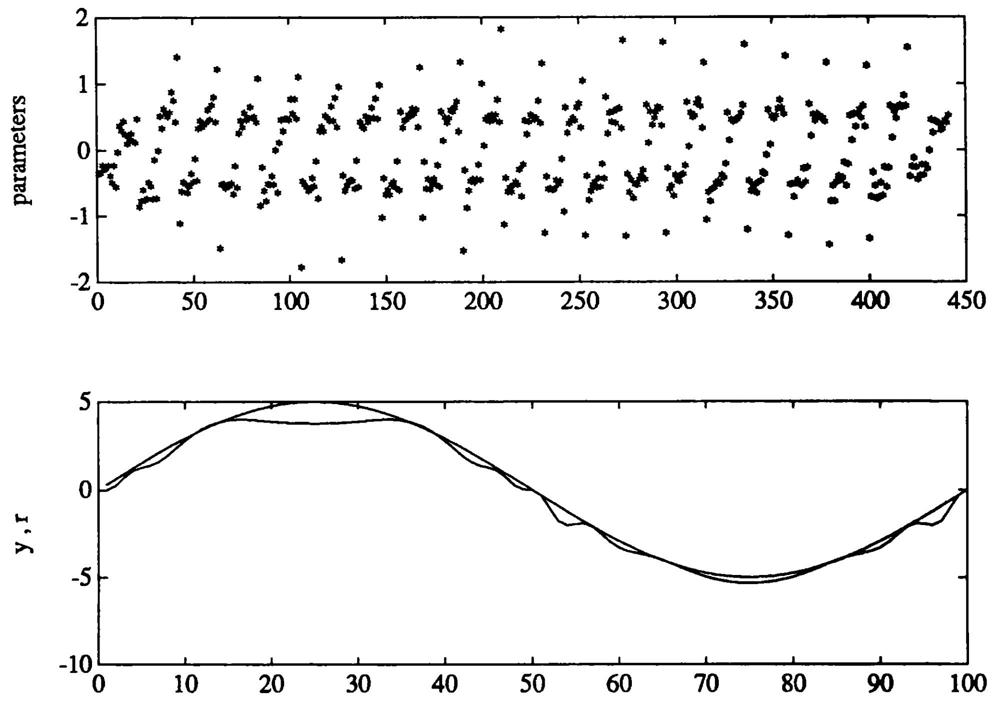


Figure 5.2: Linear coefficients and error between model and real response.

represent f_1 and f_2 over an interval defined over $y(t) = [-9, 9]$ and $u(t) = [-2, 2]$. The result of these approximations is shown in Figure 5.3.

Then the inverse is calculated as

$$u(t) = \mathcal{N}_3\left(-\mathcal{N}_1(y(t)) + r(t)\right),$$

where $\mathcal{N}_3$ is a network with 40 units representing the inverse of $\mathcal{N}_2$. The results obtained with this inverse are shown in Figure 5.4. Notice the oscillation produced in the approximation; this effect is due to the fact that the standard deviation of the gaussians units have been fixed as too high.

Using prior information about the system it is possible to obtain representations with smaller number of units and better accuracy.

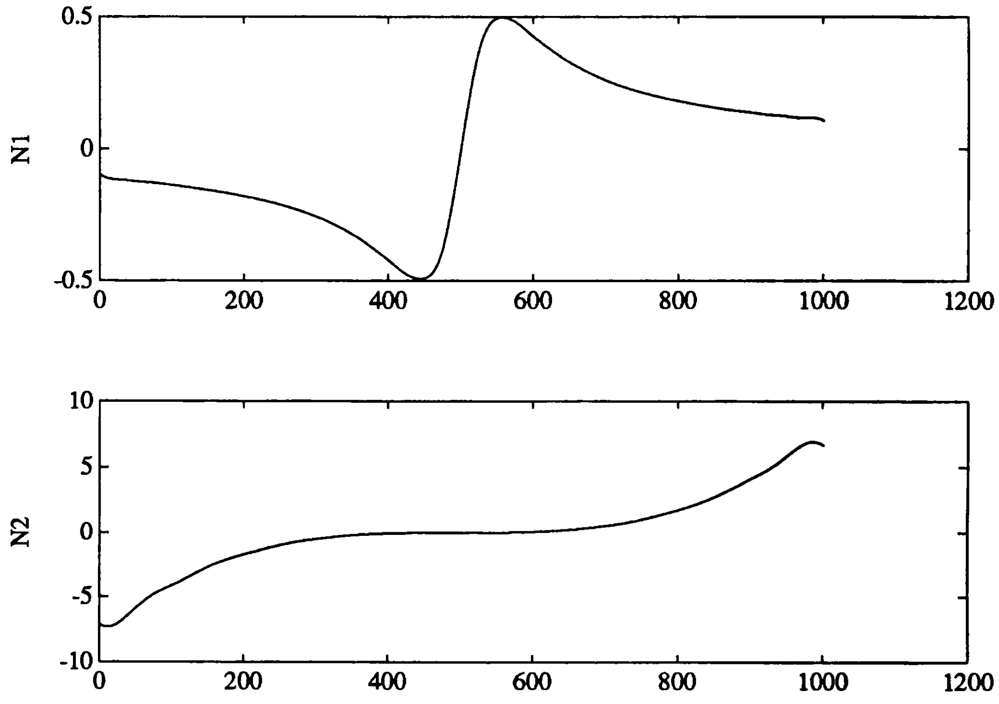


Figure 5.3: Final curves obtained.

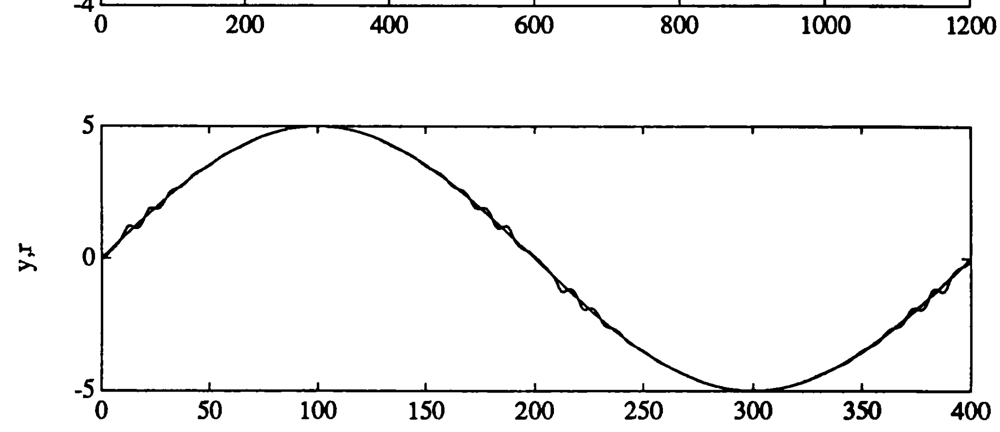


Figure 5.4: Estimate of the inverse and response of the controlled system.

5.2 Model Reference

A schematic diagram of the fundamental idea used in this approach is given in Figure 5.5.

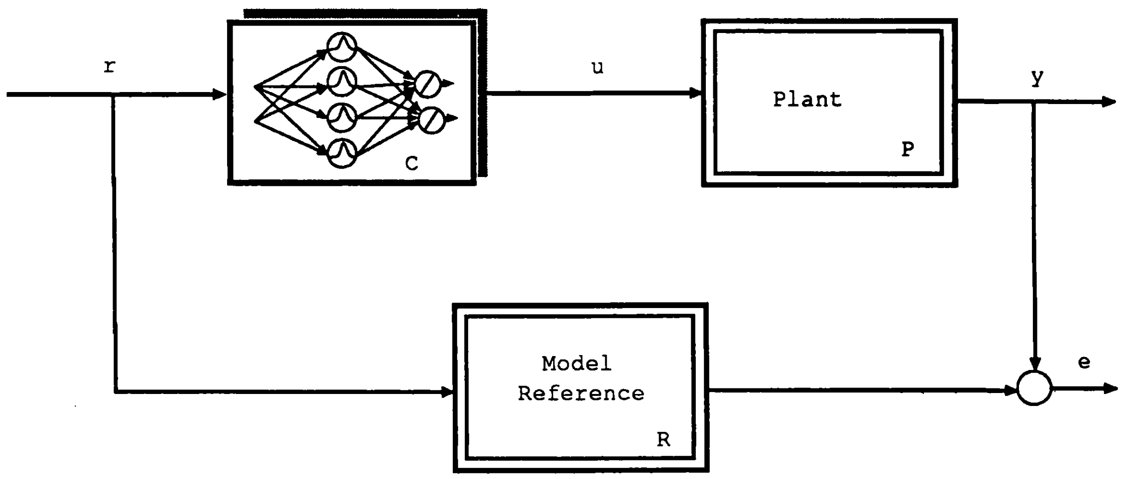


Figure 5.5: Model reference architecture.

The double lines used in the block diagram emphasise that the operators are nonlinear and that the usual block diagram manipulations do not hold.

The plant to be controlled can be represented by a nonlinear operator P and the desired performance of the closed-loop system is specified through a stable reference model, which in general can be a nonlinear system defined by its input-output pair $\{r(t), y^r(t)\}$ or by a nonlinear operator R. The control system attempts to make the plant output $y(t)$ match the reference model output asymptotically, i.e.

$$\lim_{t\to\infty} \|y(t) - y^r(t)\| \leq \epsilon$$

for some specified constant $\epsilon > 0$.

It is necessary to construct an operator C, which is represented by a connectionist network, such that

$$C \circ P = R. \tag{5.6}$$

The solution of equation (5.6) for the controller depends upon the existence conditions provided by the Implicit Function Theorem. Instead of solving a functional equation, the problem is transformed to a parametric problem, where the unknown function depends on several parameters of a known representation, in this case connectionist networks. There are two approaches to find the parameters of this representation such that satisfy the functional equation (5.6); these are: the direct and indirect approach. In the direct approach, the adjustment of the parameters of the representation is done in order to reduce the norm of the output error, that can be expressed as $\|C \circ P - R\|$. The indirect approach assumes certain knowledge of the plant, namely how the functional representation of the plant is related with the functional description of the controller. In this case, the operator representing the plant is estimated, and then the controller is chosen assuming that the identified operator represents the true operator. Figure 5.6 shows the structures used in these cases.

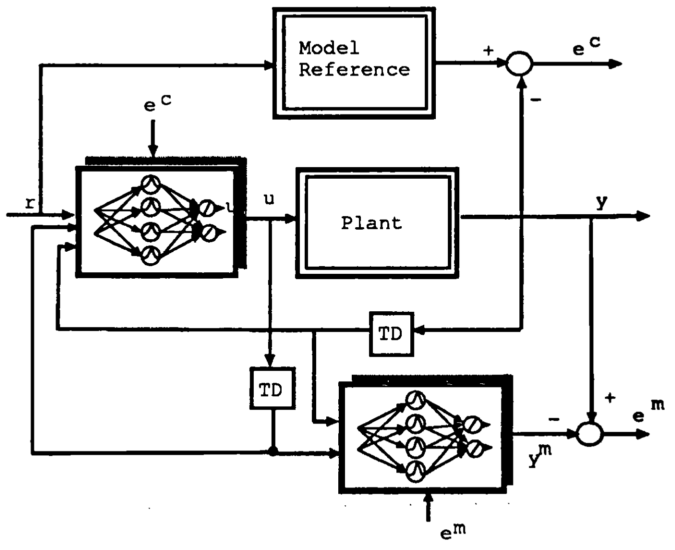
The capability of specifying nonlinear reference system is a practical and powerful tool to shape the response of the closed loop system.

5.2.1 General Structure

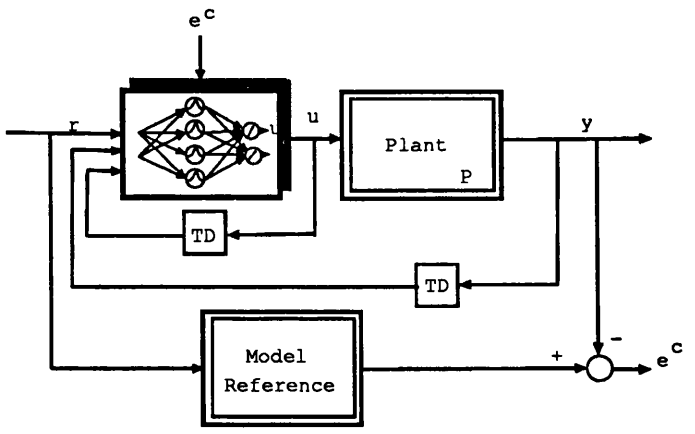
In general, the model reference is represented by a nonlinear function, but in some specific situations can be represented by a linear equation. Let the model reference be described by

$$y^r(t+1) = R(\bar{y}^r(t), \bar{r}(t)) = A\bar{y}^r(t) + B\bar{r}(t), \tag{5.7}$$

where $\bar{y}^r(t) = [y^r(t), \ldots, y^r(t-n^r)]^T, \bar{r}(t) = [r(t), \ldots, r(t-m^r)]^T, A$ is a $(1 \times n^r+1)$ matrix, and B is a $(1 \times m^r + 1)$ matrix. The system P is defined by the recursive



a) Indirect approach



b) Direct approach

Figure 5.6: Model reference architectures: direct and indirect approach.

nonlinear function

$$y(t+1) = P(\bar{y}(t), \bar{u}(t)), \tag{5.8}$$

where $\bar{y}(t) = [y(t), \ldots, y(t-n)]^T$, $\bar{u}(t) = [u(t), \ldots, u(t-m)]^T$, and $\hat{u}(t-1) = [u(t-1), \ldots, u(t-m)]^T$, then the controller performs the nonlinear mapping

$$u(t) = \mathcal{N}(\bar{y}(t), \bar{r}(t), \hat{u}(t-1)). \tag{5.9}$$

In (5.9) $\mathcal{N}$ represents a gaussian network defined as

$$u(t) = \sum_{i=1}^{N} c_i g_{\mu_i,\sigma_i}(\bar{y}(t), \bar{r}(t), \hat{u}(t-1)),$$

or in vector notation as

$$u(t) = CG(\bar{y}(t), \bar{r}(t), \hat{u}(t-1)),$$

where C is the vector formed by the linear coefficients and G is a vector of the regressors g_{μ_i,σ_i}; note that the model is linear in the parameters. The objective of training is to solve the following functional equation

$$P\left(\bar{y}(t), \mathcal{N}\left(\bar{y}(t), \bar{r}(t), \hat{u}(t-1)\right), \hat{u}(t-1)\right) = R(\bar{y}^r(t), \bar{r}(t)). \tag{5.10}$$

In order to have a continuous solution on $\mathcal{N}$ the conditions of the Implicit Function Theorem must be satisfied, this means that P must be continuously differentiable and its derivatives non zero in the domain of interest; such that equation (5.10) holds. Some precautions must be taken before implementing the controller, because solving this equation does not guarantee that the final controller is stable. The algorithm to adjust the controller based in a gradient scheme is given by the following set of equations

$$\begin{aligned}
y(t+1) &= P\left(\bar{y}(t), \mathcal{N}\left(\bar{y}(t), \bar{r}(t), \hat{u}(t-1)\right), \hat{u}(t-1)\right), \\
e(t) &= y^r(t) - y(t), \\
c_i(t+1) &= c_i(t) - \gamma \tfrac{\partial e(t)}{\partial c_i} e(t), \\
\tfrac{\partial e(t)}{\partial c_i} &= \tfrac{\partial P}{\partial u(t)} \tfrac{\partial u(t)}{\partial c_i} = \tfrac{\partial P(t)}{\partial u(t)} g_{\mu_i,\sigma_i}(\bar{y}(t), \bar{r}(t), \hat{u}(t-1)),
\end{aligned} \tag{5.11}$$

where $C = [c_1, \ldots, c_N]^T$ is the vector of parameters of the network. Under the same conditions of Theorem 3 Chapter 3, the system converges to a global minimum of the mean squared error. If $\frac{\partial P(t)}{\partial u(t)}$ is unknown, then it can be calculated on line or use a model to get an estimate of it, as was explained in section 4.3 of Chapter 4.

5.2.2 Effect of the Disturbances

Consider the difference between the model and the controlled plant for a linear model

$$y^r(t+1)-y(t+1) = A\bar{y}^r(t)+B\bar{r}(t)-P\left(\bar{y}(t),\mathcal{N}\left(\bar{y}(t),\bar{r}(t),\hat{u}(t-1)\right),\hat{u}(t-1)\right)+d \tag{5.12}$$

where d is the disturbance, and $\hat{u}(t-1) = [u(t-1), \ldots, u(t-m)]^T$. If $\mathcal{N}$ represents the inverse of P, i.e. $\mathcal{N} \circ P = R$, the above equation simplifies to

$$y^r(t+1) - y(t+1) = A\bar{y}^r(t) + B\bar{r}(t) - A\bar{y}(t) - B\bar{r}(t) + d, \tag{5.13}$$

replacing $e(t+1) = y^r(t+1) - y(t+1)$ and $\bar{e}(t) = \bar{y}^r(t) - \bar{y}(t)$ in equation (5.13) the dynamic of the error is then given by

$$e(t+1) = A\bar{e}(t) + d. \tag{5.14}$$

From this equation it is possible to see that the disturbances have a direct effect on the output of the system.

5.2.3 Separable Structures

In this case the plant can be represented by the following structure

$$y(t+1) = P(\bar{y}(t), \bar{u}(t)) = f(\bar{y}(t)) + g(\bar{u}(t)), \tag{5.15}$$

therefore the controller is given by

$$u(t) = \hat{g}^{-1}\left(-\hat{f}(\bar{y}(t)) + M(\bar{y}(t), \bar{r}(t)), \hat{u}(t-1)\right), \tag{5.16}$$

where $\hat{g}^{-1}$ represents the inverse of the estimate of g and $\hat{f}$ represents the estimate of f.

Defining a parametric representation for each function,

$$\begin{aligned} \hat{g} &= C_g G_g(\bar{u}(t)), \\ \hat{f} &= C_f G_f(\bar{y}(t)), \\ \hat{g}^{-1} &= C_{g^{-1}} G_{g^{-1}} \left(-\hat{f}(\bar{y}(t)) + M(\bar{y}(t), \bar{r}(t)), \hat{u}(t-1)\right), \end{aligned} \tag{5.17}$$

where C_g, C_f, and $C_{g^{-1}}$ are vectors containing the parameters, and G_g, G_f, and $G_{g^{-1}}$ are vectors containing the regressors.

As a first step, the model of the system represented by equation (5.15) must be identified. To achieve this the following algorithm based on the gradient approach is used

$$\begin{aligned} \hat{y}(t+1) &= \mathcal{N}_f(\bar{y}(t)) + \mathcal{N}_g(\bar{u}(t)), \\ e(t) &= y(t) - \hat{y}(t), \\ C_f(t+1) &= C_f(t) - \gamma_1 e(t) G_f^T, \\ C_g(t+1) &= C_g(t) - \gamma_2 e(t) G_g^T. \end{aligned} \tag{5.18}$$

Then, the inverse of g can be identified by the following algorithms

$$\begin{aligned} g^{-1} &= \mathcal{N}_{g^{-1}} \left(R(\bar{y}(t), \bar{r}(t)) - \mathcal{N}_f(\bar{y}(t)), \hat{u}(t-1)\right), \\ e(t) &= R(\bar{y}(t), \bar{r}(t)) - \mathcal{N}_f(\bar{y}(t)) - \mathcal{N}_g(\bar{u}(t)), \\ C_{g^{-1}}(t+1) &= C_{g^{-1}}(t) - \gamma_3 \frac{\partial \mathcal{N}_g}{\partial u(t)} e(t) G_{g^{-1}}^T, \end{aligned} \tag{5.19}$$

and finally the control signal is calculated as

$$u(t) = \mathcal{N}_{g^{-1}} \left(R(\bar{y}(t), \bar{r}(t)) - \mathcal{N}_f(\bar{y}(t)), \hat{u}(t-1)\right). \tag{5.20}$$

The stability of the controller depends on the characteristic of the function g. This scheme belongs to the indirect approach, because the model of the system is needed to build the controller.

5.2.4 Affine Structures

In this case the plant can be represented by the following structure

$$y(t+1) = P(\bar{y}(t), \bar{u}(t)) = f(\bar{y}(t), \hat{u}(t-1)) + g(\bar{y}(t), \hat{u}(t-1))u(t). \tag{5.21}$$

This system is always invertible if $g(\bar{y}(t), \hat{u}(t-1))$ is different from zero, so given this condition it is possible to find a controller defined by

$$u(t) = \frac{1}{\hat{g}(\bar{y}(t), \hat{u}(t-1))} \left(-\hat{f}(\bar{y}(t), \hat{u}(t-1)) + R(\bar{y}(t), \bar{r}(t))\right), \tag{5.22}$$

where $\hat{g}$ represents the estimated of g and $\hat{f}$ represents the estimate of f. Both are connectionist representations defined by

$$\begin{aligned} \hat{g} &= C_g G_g(\bar{y}(t), \hat{u}(t-1)), \\ \hat{f} &= C_f G_f(\bar{y}(t), \hat{u}(t-1))). \end{aligned} \tag{5.23}$$

The first step in the indirect approach is to identify the model of the system represented by equation (5.15). To achieve this the following algorithm is used

$$\begin{aligned} \hat{y}(t+1) = &\ \mathcal{N}_f(\bar{y}(t), \hat{u}(t-1)) + \mathcal{N}_g(\bar{y}(t), \hat{u}(t-1))u(t), \\ e(t) = &\ y(t) - \hat{y}(t), \\ C_f(t+1) = &\ C_f(t) - \gamma_1 e(t) G_f^T, \\ C_g(t+1) = &\ C_g(t) - \gamma_2 u(t) e(t) G_g^T. \end{aligned} \tag{5.24}$$

Then, the control signal can be calculated as

$$u(t) = \frac{1}{\mathcal{N}_g(\bar{y}(t), \hat{u}(t-1))} \left(R(\bar{y}(t), \bar{r}(t)) - \mathcal{N}_f(\bar{y}(t), \hat{u}(t-1))\right). \tag{5.25}$$

This model can work on line, but at the beginning when there is no information about the plant either an identification period must be considered or an extra controller that can work with little information about the plant must be included. A dead zone has been introduced in the updating laws to take into account modelling errors [11].

Another alternative in this case is the direct approach. Here, no information about estimated functions of the plant is used. The basic assumptions needed to prove the stability of the procedure are:

A1 The model reference is given by a stable minimum-phase linear system defined by

$$y^m(t+1) = A\bar{y}^m(t) + br(t), \tag{5.26}$$

where A is a vector defined by $A = [a_0, \ldots, a_n]^T$, and b is a constant.

A2 There exists an operational region $O \subset R^{n+m+1}$ and connectionist representations $\mathcal{N}_1$ and $\mathcal{N}_2$ such that $\delta_1 = \mathcal{N}_1(\bar{y}(t), \hat{u}(t-1)) - \frac{f(\bar{y}(t),\hat{u}(t-1))}{g(\bar{y}(t),\hat{u}(t-1))}$ and $\delta_2 = \mathcal{N}_2(\bar{y}(t), \hat{u}(t-1)) - \frac{b}{g(\bar{y}(t),\hat{u}(t-1))}$, where δ_1 and δ_2 are functions of $\bar{y}(t)$ and $\hat{u}(t-1)$ and $(\bar{y}(t), \hat{u}(t-1)) \in O$.

A3 $g_{min} < \|g(\bar{y}(t), \hat{u}(t-1))\| < g_{max}$, where g_{max} and g_{min} are real constants different from 0.

A4 The sign of $g(\bar{y}(t), \hat{u}(t-1))$ is known.

Defining the control signal as

$$u(t) = \mathcal{N}_1(\bar{y}(t), \hat{u}(t-1)) + \mathcal{N}_2(\bar{y}(t), \hat{u}(t-1))r(t) + A\bar{y}(t) \tag{5.27}$$

the error between the model and the system is $e(t+1) = y(t+1) - y^m(t+1)$,

$$\begin{aligned} e(t+1) = \; & f(\bar{y}(t), \hat{u}(t-1)) - g(\bar{y}(t), \hat{u}(t-1))\mathcal{N}_1(\bar{y}(t), \hat{u}(t-1)) + \\ & g(\bar{y}(t), \hat{u}(t-1))\mathcal{N}_2(\bar{y}(t), \hat{u}(t-1))r(t) + A\bar{e}(t), \end{aligned} \tag{5.28}$$

where $\bar{e}(t) = \bar{y}^m(t) - \bar{y}(t)$. Assuming first that δ_1 and δ_2 are zero, this means that there exist C_1^* and C_2^* such that $\frac{f(\bar{y}(t),\hat{u}(t-1))}{g(\bar{y}(t),\hat{u}(t-1))} = C_1^* G_1$ and $\frac{b}{g(\bar{y}(t),\hat{u}(t-1))} = C_2^* G_2$. Defining

$$R(t) = \begin{bmatrix} \Phi(t) \\ \Psi(t) \end{bmatrix} = \begin{bmatrix} C_1^* - C_1(t) \\ C_2^* - C_2(t) \end{bmatrix}, \tag{5.29}$$

the dynamic of the error can be written as

$$e(t+1) - A\bar{e}(t) = \begin{bmatrix} \Phi(t+1) & \Psi(t+1) \end{bmatrix} \begin{bmatrix} G_1 g \\ G_2 g r(t) \end{bmatrix}. \tag{5.30}$$

Setting $z(t+1) = e(t+1) - A\bar{e}(t)$ such that

$$z(t) = R(t)w(t-1), \tag{5.31}$$

where $w(t-1) = [G_1^T g \quad G_2^T g r(t-1)]^T$, and letting the updating laws for the parameters be

$$\begin{aligned} C_1(t+1) = \; & C_1(t) - \epsilon(t)z(t)\text{sgn}(g(\bar{y}(t), \hat{u}(t-1)))G_1^T, \\ C_2(t+1) = \; & C_2(t) - \epsilon(t)z(t)\text{sgn}(g(\bar{y}(t), \hat{u}(t-1)))r(t-1)G_2^T \end{aligned} \tag{5.32}$$

the following result is obtained.

Theorem 1 *The origin of the system formed by $z(t) = R(t)w(t-1)$ and equations (5.32) is asymptotically stable under assumptions A1, A2 with $\delta_1 = \delta_2 = 0$, A3, and A4.*

Proof:

Consider the following Lyapunov function

$$V(t) = \Phi^T\Phi + \Psi^T\Psi.$$

Then

$$\Delta V(t) = V(t+1) - V(t),$$

and upon substitution of (5.32) to (5.29), we get

$$\begin{aligned}\Delta V(t) = \; & -2\epsilon(t)z(t)\tfrac{\operatorname{sgn}(g)}{g}(\Phi G_2 g + \Psi G_1 g r(t-1)) \\ & + \epsilon(t)^2(\operatorname{sgn}(g))^2(G_1^T G_1 + G_2^T G_2 r(t-1)^2)z(t)^2.\end{aligned} \tag{5.33}$$

Further, replacing $z(t)$ with $\Phi G_1 g + \Psi G_1 g r(t-1)$, yields

$$\Delta V(t) = \; -z(t)^2(2\epsilon(t)\tfrac{\operatorname{sgn}(g)}{g} - \epsilon(t)^2(G_1^T G_1 + G_2^T G_2 r(t-1)^2)), \tag{5.34}$$

where $\epsilon(t)$ is defined as

$$\epsilon(t) = \frac{\alpha}{(|g|(G_1^T G_1 + G_2^T G_2 r(t-1)^2))}, \tag{5.35}$$

with α a constant such that $0 < \alpha < 2$. It follows that the system converges to 0 as $t \to \infty$, and as $z(t)$ represents the input of a stable linear system, then implies $e(t) \to 0$ as $t \to \infty$. $\diamond\diamond$

If there is an error in modelling, i.e. $\delta_1 \neq 0$ and $\delta_2 \neq 0$, and replacing $\delta = \delta_1 + \delta_2$ and $z(t) - \delta = \Phi G_1 g + \Psi G_1 g r(t-1)$ in the Lyapunov function given in (5.33), then

$$\Delta V(t) = \; -z(t)^2\alpha + z(t)\tfrac{2\alpha\delta}{(|g|(G_1^T G_1 + G_2^T G_2 r(t-1)^2))}. \tag{5.36}$$

An interesting interpretation of this equation is that the interaction between the quadratic term and the effect of the modelling error. The term in $z(t)$ acts as disturbance force, pulling the error away from the origin, i.e. increasing the value of the Lyapunov function, but the first term is proportional to the square of $z(t)$, while the second one is of order to $z(t)$, so when $z(t)$ is large the first term dominate bringing $z(t)$ to the origin [76]. In this way $z(t)$ is captured in a zone defined by

$$-z(t)\alpha + \frac{2\alpha\delta}{(|g|(G_1^T G_1 + G_2^T G_2 r(t-1)^2))} = 0. \tag{5.37}$$

A bound of $z(t)$ is given consequently by

$$\|z(t)\| \leq \frac{2\|\delta\|_\infty}{|g_{min}|G_1^T G_1}. \tag{5.38}$$

By definition $G_1^T G_1 > 0$ for all $(\bar{y}(t), \hat{u}(t-1)) \in O$.

A practical problem arises if the system is unstable and the learning algorithm has a small learning constant, α, the output of the system can grow much faster than the necessary control action required to maintain the system operating within the range of operational region O, where the network has been defined.

5.2.5 Linear Case

This case can be regarded as a special case of the previous section. The plant can be represented by a structure like

$$y(t+1) = P(\bar{y}(t), \bar{u}(t)) = f(\bar{y}(t)) + B(z^{-1})u(t), \tag{5.39}$$

where $B(z^{-1})$ is a polynomial defined by $B(z^{-1}) = b_0 z^{-1} + \ldots + b_m z^{m-1}$. This system is always invertible if b_0 is different from zero, and the inverse is stable if the polynomial $B(z^{-1})$ has no roots outside of the unit circle. Given these conditions, it is possible to find a stable control defined by

$$u(t) = \frac{1}{b_0}\left(-\hat{f}(\bar{y}(t)) + R(\bar{y}(t), \bar{r}(t)) - b_1 u(t-1) \ldots - b_m u(t-m)\right), \tag{5.40}$$

where $\hat{f}$ represents the estimated f, defined as $\hat{f} = C_f G_f(\bar{y}(t))$. The following algorithm based on the gradient is used to update the parameters:

$$\begin{aligned} \hat{y}(t+1) &= \mathcal{N}_f(\bar{y}(t)) + \hat{B}\bar{u}(t), \\ e(t) &= y(t) - \hat{y}(t), \\ C_f(t+1) &= C_f(t) - \gamma_1 e(t) G_f^T, \\ \hat{B}(t+1) &= \hat{B}(t) - \gamma_2 e(t)\bar{u}(t)^T. \end{aligned} \tag{5.41}$$

The model can be written in the following way

$$\hat{y}(t+1) = [C_f \quad B] \begin{bmatrix} G_f \\ \bar{u}(t) \end{bmatrix}. \tag{5.42}$$

If a bound for b_0 is known the direct approach can be applied.

5.2.6 Simulations

System linear in the control signal

The system is described by the following equation

$$y(t+1) = f(y(t), y(t-1)) + u(t) = \frac{y(t)y(t-1)}{1 + y(t)^2 + y(t-1)^2} + u(t). \tag{5.43}$$

The identified connectionist model has the following structure

$$y(t+1) = \mathcal{N}_l(y(t), y(t-1)) + u(t), \tag{5.44}$$

where $\mathcal{N}_l$ represents a regular gaussian network with 400 units, approximating f_1 over a defined space, i.e. $y(t) \in [-6, 6]$ and $y(t-1) \in [-6, 6]$. The control objective is to drive the system such that it follows the linear reference model described by

$$y^r(t+1) = .6\ y^r(t) + .2\ y^r(t-1) + .4\ r(t). \tag{5.45}$$

The control signal is given by

$$u(t) = .6\ y(t) + .2\ y(t-1) + .4\ r(t) - \mathcal{N}_l(y(t), y(t-1)). \tag{5.46}$$

In this case, the learning algorithm can be applied directly to the system. Under these circumstances the parameters of the controller are adjusted to drive the error to zero. As the network has a localised characteristic just the parameters of the neighbourhood of that trajectory are adjusted without disturbing the global map. The result of this scheme is shown in the first curve in Figure 5.7.

To apply the indirect approach, firstly, the plant equation was identified assuming different initial conditions in the state space and with zero input. Secondly, the signal control was generated using the identified system. The response of the controlled system is shown in Figure 5.7.

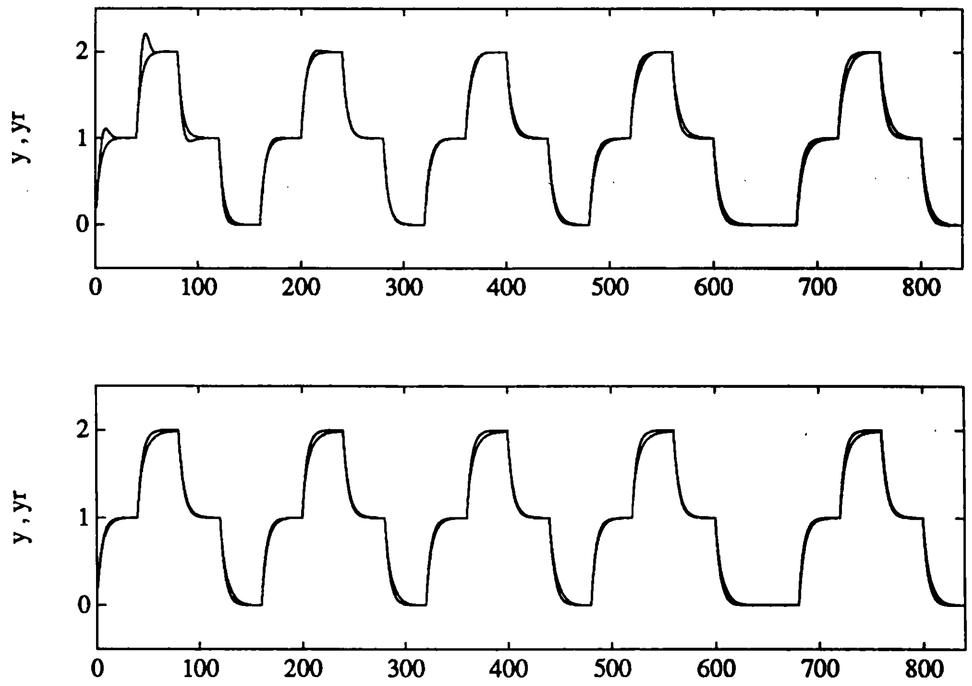


Figure 5.7: a) Direct approach: response of controlled system. b) Indirect approach: response of the controlled system.

Fully nonlinear system

The plant to be considered in these examples is described by

$$y(t+1) = f_1(y(t)) + f_2(u(t)) = \frac{y(t)}{1+y(t)^2} + u(t)^3 \tag{5.47}$$

and the model of the system is given by

$$y(t+1) = \mathcal{N}_1(y(t)) + \mathcal{N}_2(u(t)), \tag{5.48}$$

where $\mathcal{N}_1$ and $\mathcal{N}_2$ represent gaussian nets with 40 and 20 units, representing f_1 and f_2 over an specific defined by $y(t) \in [-9, 9]$ and $u(t) \in [-2, 2]$.

For all the simulations a simple first order linear reference model given by

$$y^r(t+1) = 0.6y^r(t) + 0.4r(t)$$

was used. The system (5.48) is monotonic with respect to $u(t)$ and therefore invertible. Then the control signal can be calculated as

$$u(t) = \mathcal{N}_3\left(0.6y(t) + 0.4r(t) - \mathcal{N}_1(y(t))\right),$$

where $\mathcal{N}_3$ represents the connectionist inverse of $\mathcal{N}_2$.

Figure 5.8 shows the response of the system without control and with control for an input defined as $r = \sin(\frac{2\pi t}{400})$.

Multivariable system

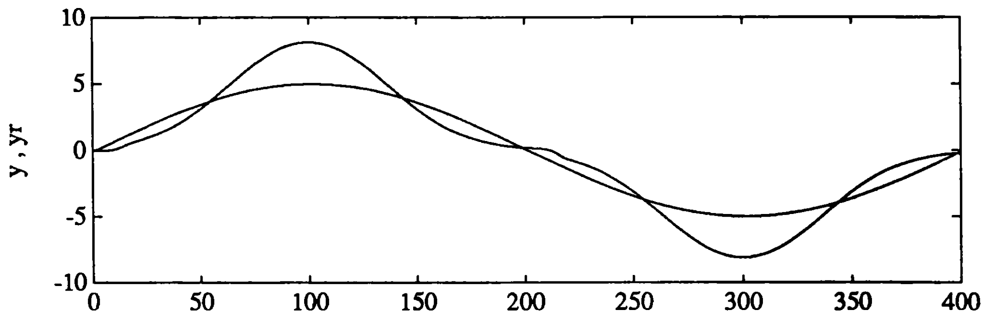
This example illustrates an extension to deal with multivariable cases. The system is described by the following equation

$$\begin{bmatrix} y_1(t+1) \\ y_2(t+1) \end{bmatrix} = \begin{bmatrix} \frac{y_1(t)}{1+y_2(t)^2} \\ \frac{y_1(t)y_2(t)}{1+y_1(t)^2} \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}. \tag{5.49}$$

The control objective is to follow the reference model given by

$$\begin{bmatrix} y_1^r(t+1) \\ y_2^r(t+1) \end{bmatrix} = \begin{bmatrix} .6 & .2 \\ .1 & -.8 \end{bmatrix} \begin{bmatrix} y_1^r(t) \\ y_2^r(t) \end{bmatrix} + \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix}. \tag{5.50}$$

In this case two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ containing 400 units each, were trained using random input signal uniformly distributed in the interval $[-1, 1]$, the final model used is the result of 1200 epochs.



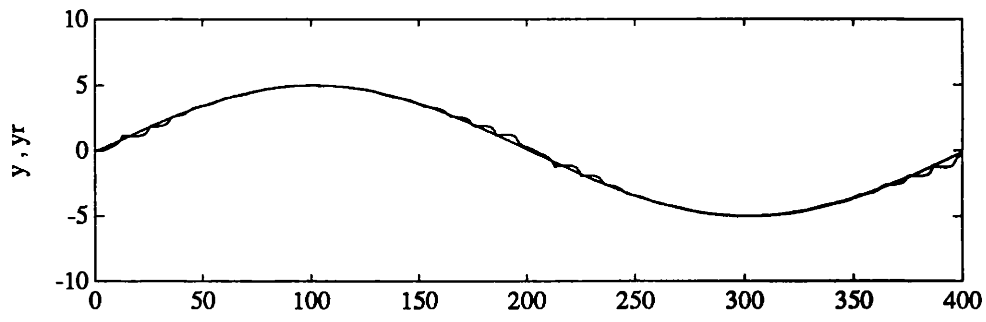


Figure 5.8: a) Open loop response; b) response of the controlled system.

The control variables are calculated by the following equation

$$\begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} .6 & .2 \\ .1 & -.8 \end{bmatrix} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} + \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix} - \begin{bmatrix} \mathcal{N}_1(y_1(t), y_2(t)) \\ \mathcal{N}_2(y_1(t), y_2(t)) \end{bmatrix}. \qquad (5.51)$$

The result achieved with the controller is shown in Figure 5.9. Note the effect of the initial conditions at the begining of the response.

5.3 Internal Model Control

The idea followed here is based upon the possibility of training networks to learn both a system's input-output relationship and the corresponding inverse relationship. A suitable control strategy which directly incorporates the plant model (and the corresponding inverse model) is provided by the Internal Model Control (IMC) [24, 59]. The applicability of IMC to nonlinear systems control was demonstrated by Economou *et al* [19]. The inverse of the nonlinear operator modelling the plant was shown to play a crucial role in the implementation of nonlinear IMC. These

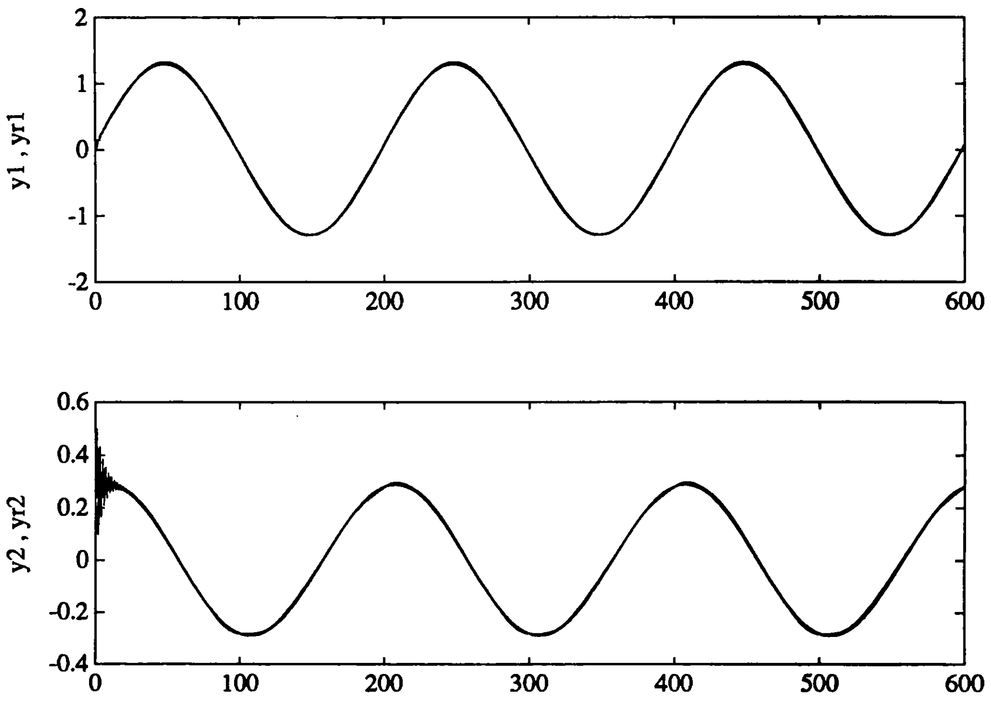


Figure 5.9: a)Response variable y_1; b) response variable y_2.

authors studied analytical and numerical methods for the necessary construction of nonlinear operator inverses. In this work connectionist networks for the construction of plant models and their inverses are employed, and the use of these networks directly within the IMC control structure is pursued.

The IMC structure is now well known and has been shown to underlie a number of control design techniques of apparently different origin [24]. IMC has been shown to have a number of desirable properties; a deep analysis is given by Morari and Zafiriou [59]. Here, a brief summary of these properties is given.

The nonlinear IMC structure is shown in Figure 5.10 (this section follows Economou *et al* [19]). Here, the nonlinear operators denoted by P, M and C represent the plant, the plant model, and the controller, respectively. The operator F denotes a filter, to be discussed in the sequel.

The important characteristics of IMC are summarised with the following properties:

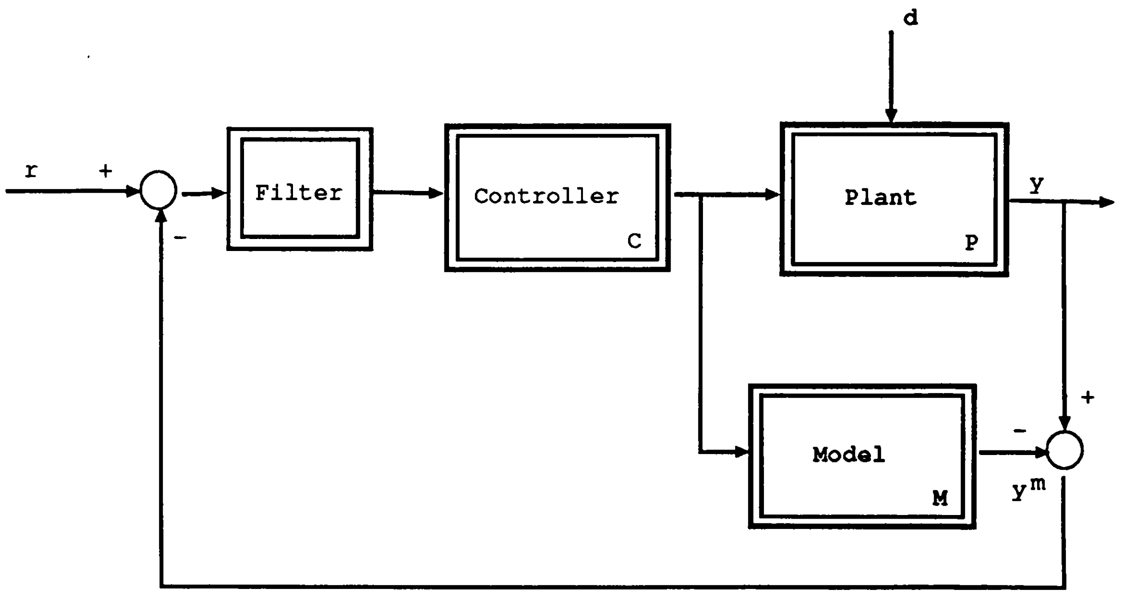


Figure 5.10: Nonlinear IMC structure.

P.1 Assume that the plant and controller are input-output stable and that the model is a perfect representation of the plant. Then the closed-loop system is input-output stable.

P.2 Assume that the inverse of the operator describing the plant model exists, that this inverse is used as the controller, and that the closed-loop system is input-output stable with this controller. Then the control will be perfect, i.e. $r = y$.

P.3 Assume that the inverse of the steady-state model operator exists, that the steady-state controller operator is equal to this, and that the closed-loop system is input-output stable with this controller. Then offset free control is attained for asymptotically constant inputs.

The IMC structure provides a direct method for the design of nonlinear feedback controllers. According to the above properties, if a good model of the plant is available, the closed-loop system will achieve exact setpoint following despite unmeasured disturbances acting on the plant.

The discussion so far has considered only the idealised case of a perfect model, leading to perfect control. In practice, however, a perfect model can never be obtained. In addition, the infinite gain required by perfect control would lead to sensitivity problems under model uncertainty. The filter F is introduced to alleviate these problems. By suitable design, the filter can be selected to reduce the gain of the feedback system, thereby moving away from the perfect controller. This introduces robustness into the IMC structure. A full treatment of robustness and filter design for IMC is given in [59].

The significance of IMC, in the context of this work, is that the stability and robustness properties of the structure can be analysed and manipulated in a transparent manner, even for nonlinear systems. Thus, IMC provides a general framework for nonlinear systems control. Such generality is not apparent in alternative approaches to nonlinear control.

A second role of the filter is to project the signal e into the appropriate input space for the controller.

5.3.1 Nonlinear IMC Using Connectionist Networks

A two step procedure for using connectionist networks directly within the IMC structure is proposed. The first step involves training a network to represent the plant response. This network is then used as the plant model operator M in the control structure of Figure 5.10. Here, the error signal between the model and the plant is used to adjust the network weights. Thus, the network is forced towards copying the plant dynamics. Full details of the learning law used here are given in Chapter 4.

Following standard IMC practice (guided by Property P.2 above) the plant inverse model is selected as the controller. The second step in the procedure is therefore to train a second network to represent the inverse of the plant. For inverse modelling the error signal used to adjust the network is defined as the difference between the (inverse modelling) network input and the plant model output. This tends to force the transfer function between these two signals to unity i.e. the network being trained is forced to represent the inverse of the plant model. Having obtained the inverse model in this way this network is used as the controller block C in the control structure of Figure 5.10.

The final IMC architecture incorporating the trained networks is shown in Figure 5.11.

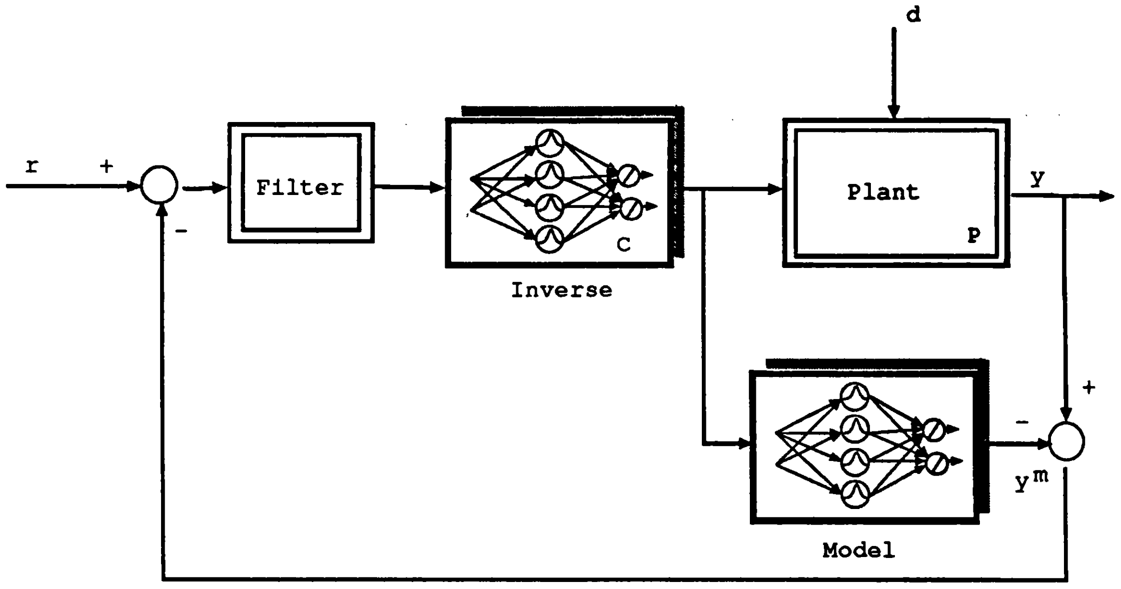


Figure 5.11: Nonlinear IMC structure incorporating connectionist models.

5.3.2 Effect of the Disturbances

For the IMC control, structure shown in Figure 5.10, follows the relationship

$$u = [I + FC(P - M)]^{-1}FC(r - d) \tag{5.52}$$

and the output is

$$y = P[I + FC(P - M)]^{-1}FC(r - d) + d \tag{5.53}$$

If the controller is equal to the inverse of the model ($C = M^{-1}$) and the closed loop is stable, then $y(t) = r(t)$ for all $t > 0$ and all disturbances $d(t)$.

5.3.3 Stability of the IMC Structure

In practice there is always an error between the plant and the model, therefore it is essential to known how the system behaves under this discrepancies. Define the gain of an operator M, which maps the domain D_M into the range R_M, as the supremum over all $u \in D_M$ of the ratios of the norm of the operator output of the associated input, i.e.

$$g(M) = \sup_{u \in D_M} \frac{\|Mu\|_p}{\|u\|_p}.$$

A very conservative bound to assure stability of the closed loop system is given by the small gain theorem [17]. For the IMC structure, this translates to

$$g((P - M)C) < 1,$$

or the sufficient condition

$$g(C)g(P - M) < 1. \tag{5.54}$$

When the plant model mismatch is large the gain of the controller must be small and usually this will not be satisfied if the ideal controller is used. Therefore the filter F can be introduced to satisfy the stability condition, and equation (5.54) becomes

$$g(CF)g(P - M) < 1,$$

and shows that as long as C and $P - M$ are stable, there will always be an F satisfying this inequality. In this case the if $P = M$ the closed loop response is given by

$$y = F(r - d).$$

As in the model reference approach the use of nonlinear filters can lead to an improvement in response without deterioration of the stability of the closed-loop system. However, at present guidelines how to design such a filter are not available.

5.3.4 Simulations

The plant to be considered in these examples is described by

$$y(t+1) = f_1(y(t)) + f_2(u(t)) = \frac{y(t)}{1+y(t)^2} + u(t)^3. \qquad (5.55)$$

The system is monotonic with respect to $u(t)$ and therefore invertible. For all the simulations a simple first order linear filter F was used. In this case the main objective of the filter was to map the error into the input space defined for the controller.

Separable Case

First, the prior information to decompose the system into two parts is used:

$$y^m(t+1) = \mathcal{N}_1(y(t)) + \mathcal{N}_2(u(t)),$$

where $\mathcal{N}_1$ and $\mathcal{N}_2$ represent connectionist networks with 40 and 20 units, representing f_1 and f_2 over a defined interval. Notice that y is used here for training the network; when training is complete the network can be used independent of the plant with $y^m(t)$ as input to $\mathcal{N}_1$. This approach is followed below.

The structure of the inverse of the model can be deduced from the above equations; we require the input of the controller r to equal the model output:

$$r(t+1) = \mathcal{N}_1(y^m(t)) + \mathcal{N}_2(u(t)).$$

Thus, the control signal u is obtained as

$$u(t) = \mathcal{N}_2{}^{-1}(r(t+1) - \mathcal{N}_1(y^m(t)),$$

where $\mathcal{N}_2^{-1}$ is a network with 40 units, representing the right inverse of $\mathcal{N}_2$.

Figure 5.12 shows the response to reference changes for the IMC structure and the response to disturbances and references changes, the system in this case is capable of eliminating the step disturbance.

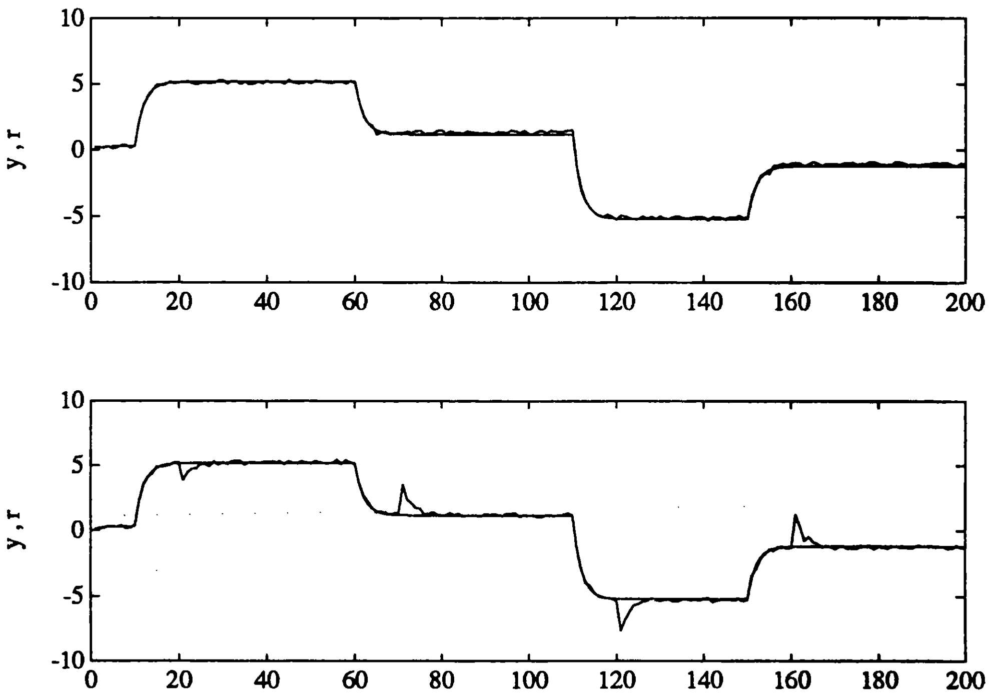


Figure 5.12: a) Response to changes in the reference signal; b) effect of disturbances.

Single Network

Alternatively, the structure of the non-linearity may be ignored. In this case a single connectionist network represents the system:

$$y^m(t+1) = \mathcal{N}(y^m(t), u(t)).$$

The inverse is represented by

$$u(t) = \mathcal{N}^{-1}(y^m(t), r(t+1)).$$

This represents the right inverse of $\mathcal{N}$. Figure 5.13 shows the response using the IMC structure to changes in the reference signal using a mesh of 100 units, and the Figure above shows the relation between the input to the inverse and the output of the model, that is $\mathcal{N}(y^m(t), \mathcal{N}^{-1}(y^m(t), r(t+1)))$.

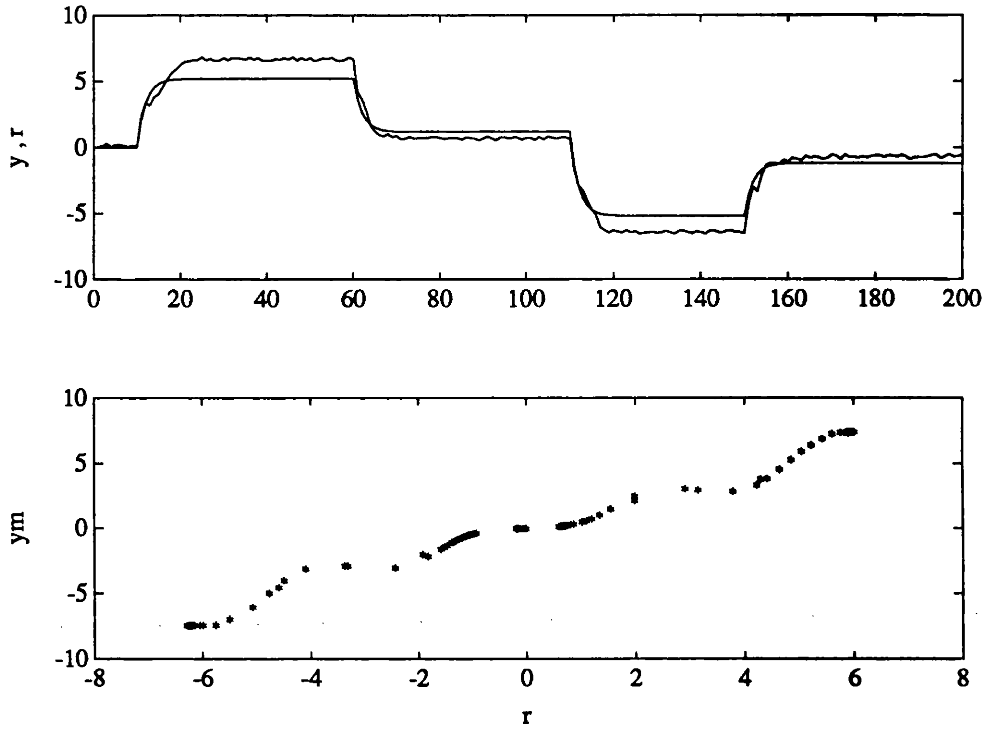


Figure 5.13: Response to changes in reference signal using 100 units and relation between input to the controller and output of the model.

Figure 5.14 shows the response to changes in the reference signal using a mesh of 400 units; in this case the error is much smaller compared with the 100 units case. Additionally, it shows the relation between the input to the network that represents the inverse of the model and the output of the model, for this case. Theoretically, it is possible to increase the number of units to obtain the desired behaviour.

Iterative Calculation

The fidelity of the approximation is limited by the number of units. In order to increase the accuracy without increasing the number of units it is possible to add a refinement procedure over the first approximation using the iterative procedure outlined in Section 4.3.2 of Chapter 4.

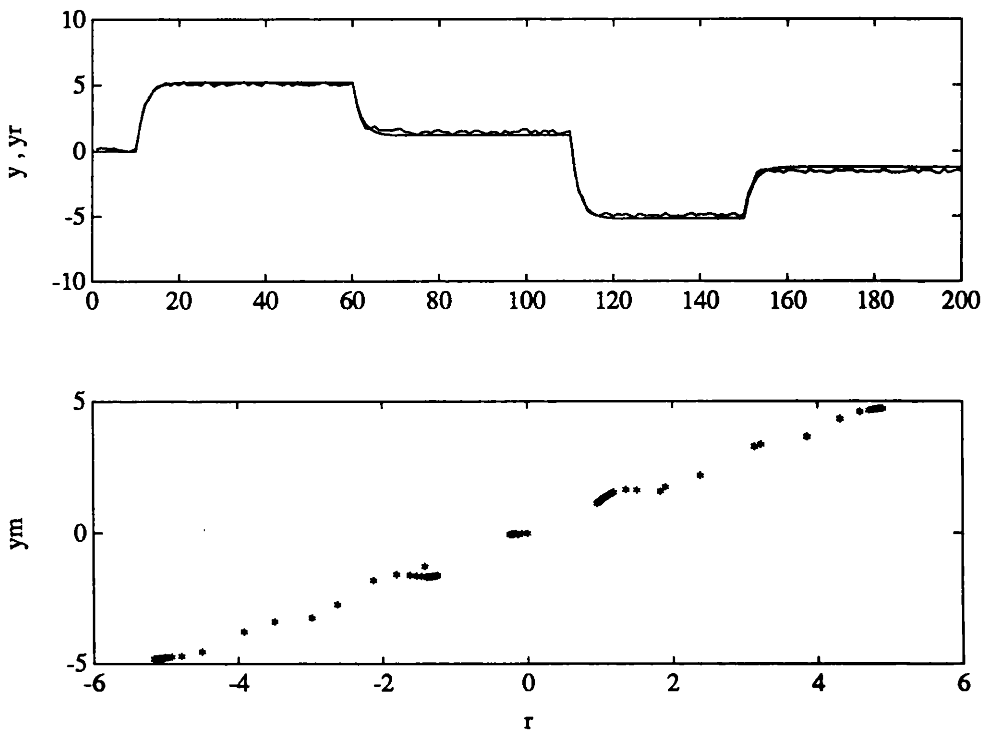


Figure 5.14: Response to changes in reference signal using 400 units and relation between input to the controller and output of the model.

The method used in the refinement, as described before, is the successive substitution. Figure 5.15 shows the results obtained with this procedure as a complement to a network with 100 units. Notice the increased accuracy as a result of the improvement in the inverse model. The bottom curve shows the relation between the input to the inverse and the output of the model for this case.

5.4 Controller Based On Prediction Models

This approach makes use of the model to calculate the control based on an optimisation of a certain cost function. The general structure is given in Figure 5.16.

Depending upon the cost function chosen several algorithms can be obtained.

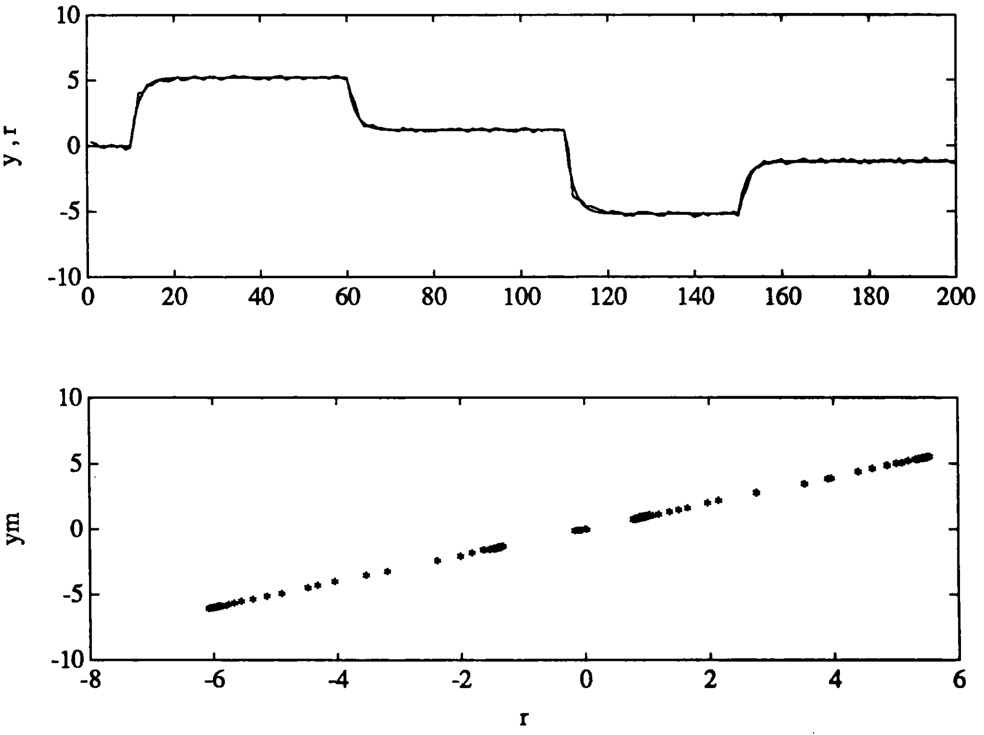


Figure 5.15: Response to changes in reference signal using iterative scheme and relation between input to the controller and output of the model.

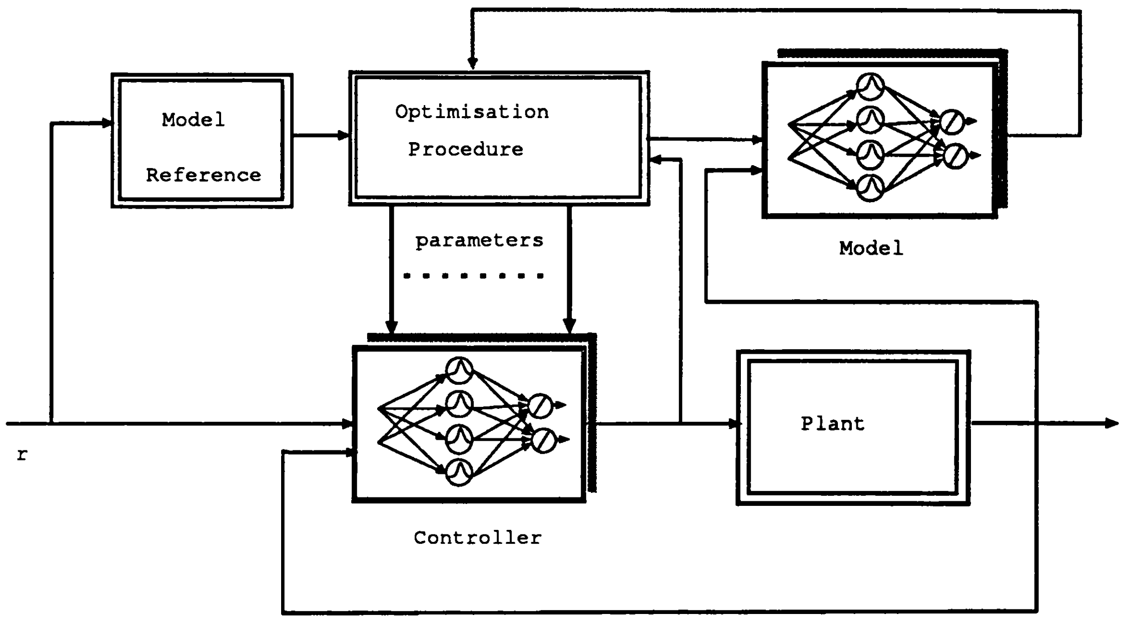


Figure 5.16: Predictive control structure.

5.4.1 Receding Horizon Control

The method, can be summarised as follows:

1. Predict the system output over the range of future time intervals.
2. Assume that the future desired outputs are known.
3. Choose a set of future controls, $\hat{u}$, which minimise the future errors between the predicted future output and the future desired output.
4. Use the first element of $\hat{u}$ as a current input and repeat the whole process at the next instant.

It has been shown that this technique can stabilise linear systems [16] and non-linear systems, as well [52].

The objective is to calculate the control such that the error over a future horizon is minimised, i.e.

$$\begin{aligned} J_{N_1,N_2,N_u}(t) &= \tfrac{1}{2}\textstyle\sum_{i=N_1}^{N_2} Q(i)(y^r(i+t) - y^m(i+t))^2 \\ &+\tfrac{1}{2}\textstyle\sum_{i=1}^{N_u} R(i)\Delta u(i+t-1)^2, \end{aligned} \tag{5.56}$$

where y^r represents the output of the reference model and y^m the output of the plant's model . The first term of the cost functions is a measure of the distance between the model prediction and the desired future trajectory. The second term penalises excessive changes of the manipulated variable.

This procedure generates an *approximate inverse* with almost any desired characteristic by solving this optimisation problem. The process model can be employed to predict the outputs resulting of a series of inputs. Alternatively, desired outputs can be prescribed and the inputs can be calculated such that the predicted output follows the predescribed output in some optimal manner. If one requires that the predicted output agrees with the prescribed one exactly, the system inputs resulting from the solution of this matching problem will be the same as would be obtained by inversion of the process model. On other hand, if one requires the predicted values only to be close to the desired values in the least square sense, the solution of the optimisation problem will provide an approximate inverse of the process model. The characteristic of the approximate inverse can be affected by the choice of N_1, N_2, and the weighting constants $R(i)$ and $Q(i)$ in the cost function.

5.4.2 Some General Results

In this section we summarise some results about the characteristics of the receding horizon controller. This summary is based on references [39],[52] and [51].

Statement of the Problem

Consider a plant described as

$$\begin{aligned} y(t+1) = f(y(t),\dots,y(t-n+1),u(t),\dots,u(t-m+1)), \quad & t > i, \quad i > n, \\ (y(t),\dots,y(t-n+1),u(t),\dots,u(t-m+1)) \in Z_t \subset R^{n+m}, \quad & t > i, \qquad (5.57) \\ y(i) = a, \quad & \qquad (5.58) \end{aligned}$$

where f is a function defined as $f : R^{n+m} \rightarrow R$, a defines the initial condition of the system, $n \geq m > 0$, and Z_t defines the admissible region for the control and state signals, such that the *moving horizon cost* at time i is

$$J_i = \frac{1}{2}\sum_{k=N_1}^{N_2} \|e(k+i)\|^2_{Q(k)} + \frac{1}{2}\sum_{k=1}^{N_2} \|\Delta u(k+i-1)\|^2_{R(k)}, \qquad (5.59)$$

Here, $e(t)$ is defined as $e(t) = y^r(t) - y(t)$, $\Delta u(t) = u(t) - u(t-1)$, $\|\cdot\|$ defines the L_2 norm, N_2 defines the horizon, and N_1 defines the minimum prediction horizon. The reference model is described by

$$\begin{aligned} & y^r(t+1) = g(y^r(t),\dots,y^r(t-n^r+1),r(t),\dots,r(t-m^r+1)), \\ & (y^r(t),\dots,y^r(t-n^r+1),r(t),\dots,r(t-m^r+1)) \in Z^r_t \subset R^{n^r+m^r}, \quad t > i, \\ & y^r(i) = b. \qquad (5.60) \end{aligned}$$

Initially, the model and the plant are in different steady states, that is $y^r(t) = y^r(t-1) = \ldots = y^r(t-n^r) = b$ and $y(t) = y(t-1) = \ldots = y(t-n) = a$.

The problem is to determine a sequence $\{u(k)\}_{k=i}^{i+N_2}$ which minimises the cost given by (5.59), subject to (5.58), (5.60) and the additional constraints

$$i \leq t \leq i + N_2 \quad , \quad y(i+N_2) = y^r(i+N_2). \tag{5.61}$$

This means that the reference model must have a settling time less than or equal to N_2, that is $y^r(i+N_2) = y^r_{ss}$, where the subscript ss stands for steady-state. The demonstration of existence of the solution, stability of the closed loop system, and robustness are based on references [39], [52], and [51].

The minimum prediction horizon N_1 is usually chosen zero, but it is useful to choose $N_1 > 0$ when the system has a time delay or when it is asymptotically non-minimum-phase. The maximum prediction horizon N_2 has an effect on the rise-time of the response, and must agree with the rise time of the model. If the system is minimum-phase, the minimum value of N_1 should be greater than the time required by the system to cover the greater positive going part [16].

Assumptions and Notation

For $i \geq 0$, $b - a \in R$, let $P(i, b-a)$ be the problem of minimising the cost J_i in (5.59) subject to (5.58), (5.60), and (5.61). A sequence $\{(e(k), \Delta u(k))\}_{k=i}^{i+N_2}$ is admissible to $P(i, b-a)$ if it satisfies (5.58). For $i \geq 0$, let $Y(i) = \{b-a : P(i, b-a)$ has an admissible sequence for which J_i is finite $\}$. Let $V(i, b-a, u^*, N_2)$ be the optimal cost of $P(i, b-a, u^*, N_2)$, and $h_k = \frac{1}{2}\|e(k+i)\|^2_{Q(k)} + \frac{1}{2}\|\Delta u(k+i)\|^2_{R(k)}$, then the index (5.59) can be written as

$$J_i = \sum_{k=1}^{N_2} h_k(e(k+i), \Delta u(k+i)),$$

where $Q(k) = 0$ for $1 \leq k \leq N_1$.

The basic assumptions adopted here are:

A.1 For each $k \geq 0$ Z_k is closed. The functions f and g are continuous and the function h_k is lower semicontinuous.

A.2 $\exists\ H_1 \in C^{\infty}$, such that

$$h_k(e(k), \Delta u(k)) \geq H_1(\|e(k), \Delta u(k)\|)$$

with $k > 0$ and $(e(k), \Delta u(k)) \in R^2$.

A.3 $\exists\ H_2 \in C^{\infty}$, such that

$$h_k(e(k), \Delta u(k)) \leq H_2(\|e(k), \Delta u(k)\|)$$

with $k > 0$ and $(e(k), \Delta u(k)) \in R^2$.

A.4 $N_2 \geq N_c$ where N_c is the index in the definition of property C, see section 4.2.2.

Main Results

Theorem 2 Suppose **A.1, A.4** hold and $b - a \in Y_i$. Then $P(i, b-a, N_2)$ has a solution.

Proof:

Assume **A.1** and **A.4** are satisfied so that, $\forall\ i \geq 0\ a \in Y_i(k)$, there is an optimal control sequence $\{\hat{u}(k)\}_{k=i}^{i+N_2}$ for the problem $P(i, b-a, N_2)$, which is a finite dimensional problem that corresponds to minimising a nonnegative, lower semicontinuous function (J_i) with respect to the variables $\{(e(k), \Delta u(k))\}_{k=i}^{i+N_2}$ on a closed set $(\prod_{k=i}^{i+N_2} Z_k)$ subject to equality constraints involving continuous function $e(i) = b - a$ and $y(i + N_2) = y_{ss}^r$). This together with **A.4** is sufficient to imply the existence of a solution [39]. $\diamond\diamond$

The existence of a bounded closed region in $y(t)$ and $u(t)$ spaces, and the assumption that f is continuous is then enough to assure the existence of a solution. Any admissible choice of control variable automatically yields and upper bound for J.

The problem of uniqueness of the minimising control requires some convexity conditions over Z_k and h_k, which play a major role in determining the structure of the solution [7].

Theorem 3 Suppose (5.58) satisfies property C and assumptions **A.1 - A.4** hold, then for all $i \geq 0$ and $b-a \in Y_i$ $\lim_{k\to\infty} e(k, b-a) = 0$.

Proof:

Consider [39]

$$V(t, e(t), u_t^*, N_2) = h_t(e(t), \Delta u_k^*(t)) + \sum_{k=t+1}^{t+N_2} h_k(e(k), \Delta u_k^*(k)), \tag{5.62}$$

define

$$V(t+1, e(t+1), \bar{u}_k^*, N_2 - 1) = \sum_{k=t+1}^{t+N_2} h_k(e(k), \Delta u_k^*(k)),$$

where $\bar{u}_t^*$ is the sequence of length $N_2 - 1$ defined by $\{u_t^*(k)\}_{k=t+1}^{t+N_2}$, and with the sequence $\hat{u}_t^*$ given as

$$\hat{u}_t^*(l) = \begin{cases} \bar{u}_t^*, & t+1 \leq l \leq N_2; \\ \bar{u}_t^*(T), & l = N_2 + 1. \end{cases} \tag{5.63}$$

By definition of the index and by the fact that $u^*(t)$ is not necessarily optimal for $P(t+1, e(t+1), N_2)$, we have

$$V(t+1, e(t+1), \bar{u}_t^*, N_2 - 1) = V(t+1, e(t+1), \hat{u}_t^*, N_2) \geq V(t+1, e(t+1), u_{t+1}^*, N_2). \tag{5.64}$$

Considering (5.62) and (5.64) the following expression is obtained

$$V(t, e(t), u_t^*, N_2) - V(t+1, e(t+1), u_{t+1}^*, N_2) \geq h_t(e^*(t), \Delta u_t^*(t)).$$

Then,

$$V(i, e(i), u_i^*, N_2) \geq V(i, e(i), u_i^*, N_2) - V(j, e(j), u_j^*, N_2) \geq \sum_{k=i}^{j-i} h_k(y(k), \Delta u_i^*(k))$$

for $j > i$. Since j can be arbitrarily large

$$V(i, e(i), u_i^*, N_2) \geq \sum_{k=i}^{\infty} h_k(e(k), \Delta u_i^*(k)) \geq \sum_{k=i}^{\infty} h_k(e(k+i), \Delta u_i^{*\infty}(k+i)),$$

where $u_i^{*\infty}(k)$ is the optimal control with infinite horizon, then, as $V(i, e(i), u_i^*, N_2)$ is a finite quantity, **A.4** implies $H_1(\|e(k), \Delta u(k)\|) \to 0$ as $k \to \infty$. $H_1 \in C^\infty$ implies $\|e(k), \Delta u(k)\| \to 0$ as $k \to \infty$. $\diamond\diamond$

Theorem 4 Suppose that assumptions **A.1 - A.4** are satisfied. Then there exists an admissible system model discrepancy, such that, for the same reference model g, there is a positive constant γ such that

$$V(t, e^s(t), u_t^*, N_2) - V(t+1, e^s(t+1), u_{t+1}^*, N_2) < \gamma,$$

where $e^s(t) = y^r(t) - y^s(t)$, and $y^s(t)$ represents the real output of the system.

Proof:

Consider

$$V(t, e^s(t), u_t^*, N_2) = h_t(e^s(t), \Delta u_t^*(t)) + \sum_{k=t+1}^{t+N_2} h_k(e^s(k), \Delta u_k^*(k)). \tag{5.65}$$

Define

$$V(t+1, e^s(t+1), \bar{u}_t^*, N_2 - 1) = \sum_{k=t+1}^{t+N_2} h_k(e^s(k), \Delta u_k^*(k)),$$

where $\bar{u}_t^*$ is the sequence of length N_2 defined by $\{u_t^*(l)\}_{l=t+1}^{t+N_2}$, and the sequence $\hat{u}_t^*$ as

$$\hat{u}_t^*(l) = \begin{cases} \bar{u}_t^*, & t+1 \le l \le N_2; \\ \bar{u}_t^*(T), & l = N_2 + 1. \end{cases} \tag{5.66}$$

By definition of the index and by the fact that $u_t^*(t)$ is not necessarily optimal for $P(t+1, e(t+1), N_2)$, we have

$$\begin{aligned} V(t+1, e^s(t+1), \bar{u}_t^*, N_2 - 1) &= V(t+1, e^s(t+1), \hat{u}_t^*, N_2) - h_{t+N_2+1}(e^s(N_2+1), 0) \\ &\ge V(t+1, e^s(t+1), u_{t+1}^*, N_2). \end{aligned} \tag{5.67}$$

Considering (5.65) and (5.67) the following expression is obtained

$$\begin{aligned} &V(t, e^s(t), u_t^*, N_2) - V(t+1, e^s(t+1), u_{t+1}^*, N_2) \ge \\ &h_t(e^s(t), \Delta u_t^*(t)) - h_{t+N_2+1}(e^s(t+N_2+1), 0). \end{aligned} \tag{5.68}$$

Letting $\gamma = h_t(e^s(t), \Delta u_t^*(t)) - h_{t+N_2}(e^s(t+N_2), 0)$ the desired result is obtained. $\diamond\diamond$

Ideally $e^s(t+N_2+1)$ is equal to zero, by calculation of u, but as an effect of the mismatch between the model and the real system there is an error at the final time expressed by $e^s(t+N_2+1) = y^s(t+N_2+1) - y^m(t+N_2+1)$.

This proposition shows that the system can deal with model discrepancy, but does not give the lower bound for this discrepancy that results in a stable closed loop system. The second term in γ reflects the mismatch between the model and the real system. The effect of not meeting the final condition in the optimisation process yields to the same result, and therefore the closed loop stability cannot be guaranteed if there is a big discrepancy.

5.4.3 Minimising the Functional

To minimise the functional (5.59) a simple gradient algorithm was used, although more efficient, but at the same time more complex, algorithms can be applied [39], [69].

Defining the vector

$$x(k) = [y(k)\ldots, y(k-n), u(k-1), \ldots, u(k-m)]^T$$

the system represented by (5.58) can be expressed in a state space equation as

$$x(k+1) = \begin{bmatrix} 0 & \ldots & 0 & 0 & \ldots & 0 \\ & I_n & & \vdots & & \vdots \\ & & & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & & \vdots & & I_{m-1} & \\ 0 & \ldots & 0 & & & \end{bmatrix} x(k) + \begin{bmatrix} f(x(k), u(k)) \\ 0 \\ \vdots \\ u(k) \\ 0 \\ \vdots \end{bmatrix} \qquad (5.69)$$

with the output defined as

$$y(k) = [1, 0, \ldots, 0]x(k).$$

Define A, F, and C such that

$$\begin{aligned} x(t+1) &= Ax(t) + F(x(t), u(t)) \\ y(t) &= Cx(t), \end{aligned} \tag{5.70}$$

and consider the quadratic index

$$J(x(0), u|_k) = J_a + J_b = .5[\sum_{k=1}^{N_2} Q(k)(y^r(k) - y(k))^2 + \sum_{k=1}^{N_2} \Delta u(k-1)^2 R(k)],$$

where $\Delta u(k) = u(k) - u(k-1)$.

Taking the derivative of J_a

$$\frac{\partial J_a}{\partial u(k)} = \sum_{j=1}^{N_2} Q(j)(y^r(j) - y(j))C\frac{\partial x(j)}{\partial u(k)},$$

with $\delta_{a,k+1}$ given by

$$\delta_{a,k+1} = \sum_{i=k+1}^{N_2-1} \left[\left[\prod_{j=k+1}^{i-1} \frac{\partial x(j+1)}{\partial u(j)} \right] Q(j)((y^r(j) - y(j)) \right]$$

results in

$$\frac{\partial J_a}{\partial u(k)} = C\frac{\partial x(k+1)}{\partial u(k)}\delta_{a,k+1}.$$

The quantity $\delta_{a,k+1}$ can be calculated in the recursive form

$$\delta_{a,k-1} = \frac{\partial x(k)}{\partial u(k-1)}\delta_{a,k} + Q(j)((y^r(k) - y(k)), \tag{5.71}$$

with

$$\delta_{a,N_2-1} = Q(N_2)((y^r(N_2) - y(N_2)), \tag{5.72}$$

and

$$\prod_{j=j_1}^{j_2} \frac{\partial x(j+1)}{\partial u(j)} = I \;\; \forall \;\; j_2 < j_1. \tag{5.73}$$

The derivative of J_b is

$$\frac{\partial J_b}{\partial u(k)} = R(k)\Delta u(k) - R(k+1)\Delta u(k+1).$$

Finally,

$$\frac{\partial J}{\partial u(k)} = \frac{\partial J_a}{\partial u(k)} + \frac{\partial J_b}{\partial u(k)}, \qquad k = 0, \ldots, N_2 - 1.$$

In practice the derivatives are calculated using a connectionist model,

$$\frac{\partial x(k+1)}{\partial u(k)} = \frac{\partial F}{\partial u(k)} = \begin{bmatrix} \frac{\partial f}{\partial u(k)} \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}. \tag{5.74}$$

As u can have a parametric representation, the problem can be transformed to the adjustment of the parameters of that representation.

For a first order plant defined as

$$y(t+1) = f(x(t)),$$

with $x(t) = [y(t), u(t)]$, the derivatives of the output against an input variable are estimated from the model and used to calculate the gradient in each iteration:

$$u|_{i+k}^{i+N_2} = u|_{i+k}^{i+N_2} + \eta(\nabla_{u|_{i+k}^{i+N\_2}} I_y + \nabla_{u|_{i+k}^{i+N_2}} I_u), \tag{5.75}$$

where η fixes the step of the gradient, $J_a(i) = \sum_{k=i+N_1}^{i+N_2} Q(k)(y^r(i+k) - y^m(i+k))^2$, $J_b = \sum_{k=1}^{N_2} R(k)\Delta u(i+k)$. The gradients are calculated as

$$\nabla_{u|_{i+k}^{i+N_2}} J_a = \begin{pmatrix} \frac{\partial y^m(i+N_1)}{\partial u(i)} & \dots & \frac{\partial y^m(i+N_2)}{\partial u(i)} \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{\partial y^m(i+N_2)}{\partial u(i+N_2)} \end{pmatrix} \begin{pmatrix} (y^r(i+N_1) - y^m(i+N_1))Q(N_1) \\ \vdots \\ (y^r(i+N_2) - y^m(i+N_2))Q(N_2) \end{pmatrix}, \tag{5.76}$$

$$\nabla_{u|_{i+k}^{i+N_2}} J_b = \begin{pmatrix} R(1) & -R(1) & 0 & \dots & 0 \\ 0 & R(2) & -R(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & R(N_2-1) & R(N_2-1) \\ 0 & 0 & 0 & \dots & R(N_2) \end{pmatrix} \begin{pmatrix} \Delta u(i) \\ \Delta u(i+1) \\ \vdots \\ \Delta u(i+N_2-1) \\ \Delta u(i+N_2) \end{pmatrix}, \tag{5.77}$$

where $\Delta u(t) = u(t) - u(t-1)$. The model can be represented as a gaussian network, with $y^m(t+1)$ given by

$$y^m(t+1) = \sum_{i=1}^{N^m} c_i^m g_{\mu_i,\sigma_i}(x(t)). \tag{5.78}$$

Here, N^m represents the number of units in a connectionist network, c_i a linear coefficient, and $g_{\mu_i,\sigma_i}(x(t))$ is a gaussian function defined as $e^{\frac{\|x-\mu_i\|^2}{\sigma_i}}$.

To calculate the partial derivatives the following formulae were used for each control action $u(k)$ with $k = 1,\dots,N_2$ and $j = N_1,\dots,N_2-1$. The partial derivative $D_{k,j} = \frac{\partial y^m(i+j)}{\partial u(k)}$ can be calculated as:

$$D_{k,j} = \begin{cases} \sum_{l=1}^{N} c_l^m g_{\mu_i,\sigma_i}(x(t)) \frac{(u^m(k)-m_{2l})}{\sigma_l}, & k=j; \\ D_{k,j-1} \sum_{l=1}^{N} c_l^m g_{\mu_i,\sigma_i}(x(t)) \frac{(y^m(i+j)-m_{1l})}{\sigma_l}, & k<j; \\ 0, & k>j. \end{cases} \tag{5.79}$$

At the end of the optimisation process it is necessary to verify that the condition $y^r(t+N_2) = y^m(t+N_2)$ is met.

For every starting point in the optimisation process, $x^m(t) = [y(t),\dots,y(t-n),u(t),\dots,u(t-m)]$, there is an associated optimal value of u_k. Thus, u_k can be expressed as

$$u_k = \mathcal{N}_c(x_k, r_k), \tag{5.80}$$

where $\mathcal{N}_c$ is an operator to be represented by a connectionist network. This network becomes the controller C in the overall control structure.

5.4.4 Effect of the Disturbances

It is important to include a reasonable representation of likely disturbances since the correct controller structure can then be deduced. As it has been pointed out in [25], it should not be necessary to force the controller to have integral action, but rather this structure should arise naturally from reasonable assumptions about the dynamics of the controlled system.

In this case, the general prediction model is used to calculate the control signal including an estimate of a step-like disturbance. The general model is given by

$$y^m(t+1) = \sum_{i=1}^{N^m} c_i^m g_{\mu_i,\sigma_i}(x(t)) + \hat{d}(t),$$

where

$$\hat{d}(t+1) = \begin{cases} \hat{d}(t) & ,k > i \\ (y^s(t) - y^m(t)) & ,k = i \end{cases} \tag{5.81}$$

where k is the predictive index, which starts from i.

The disadvantage of this approach is that it assumes independent states of the model, this assumption can be dangerous in nonlinear systems as was explained in Chapter 4.

5.4.5 Comments on Differences Between Linear and Nonlinear Receding Horizon Formulations

The approach presented here is an extension of the linear case studied by Kwon and Pearson [43] and further developed to the nonlinear case [55], [39], and [10]. From this point of view it is closely related with GPC [14], the main difference is that the former approach specifies a constraint at the final time, the effect of this extra specification is that the stability of the closed loop is established. On the other hand, in the GPC approach to get stabilising closed loop controllers a constraint on the increment of $u(t)$ is imposed, but the problem with this strategy is that it is almost impossible to ascertain how to restrict the increments of $u(t)$ to achieve the desired result [60]. In the nonlinear case the analysis is more difficult, because it is not possible to solve the control law for a general model, under this condition the introduction of an scheme like GPC can be dangerous in that the conditions to get closed loop stable system are almost impossible to obtain in a general way. As a general recommendation it is preferable to use a condition in the final state, in the same way as in the linear case it is possible to establish the closed loop stability of the system at the expense of more calculations.

5.4.6 Simulations

The plant to be considered in these examples is described by

$$y(t+1) = f_1(y(t)) + f_2(u(t)) = \frac{y(t)}{1+y(t)^2} + u(t)^3. \qquad (5.82)$$

The system is monotonic with respect to $u(t)$ and therefore invertible. For all the simulations a simple first order linear reference model given by

$$y^r(t+1) = .6y(t) + .4u(t)$$

was used.

First, we make use of the prior information to decompose the system into two parts:

$$y^m(t+1) = \mathcal{N}_1(y(t)) + \mathcal{N}_2(u(t)),$$

where $\mathcal{N}_1$ and $\mathcal{N}_2$ represent connectionist networks with 40 and 20 units, which representing f_1 and f_2 over a defined interval. Notice that y is used here for training the network; when training is complete the network can be used independent of the plant with $y^m(t)$ as input to $\mathcal{N}_1$. Defining the index as

$$J_{1,3,3} = \sum_{k=1}^{3} Q(i)(y^r(i+k) - y^m(i+k))^2 + \sum_{k=1}^{3} R(i)(u(i+t-1) - u(i+t-2))^2$$

the following are obtained

$$\nabla_{u|_{i+1}^{t+2}} J_a = \begin{pmatrix} \frac{\partial y^m(i+1)}{\partial u(i)} & \frac{\partial y^m(i+2)}{\partial u(i)} & \frac{\partial y^m(i+3)}{\partial u(i)} \\ 0 & \frac{\partial y^m(i+2)}{\partial u(i+1)} & \frac{\partial y^m(i+3)}{\partial u(i+1)} \\ 0 & 0 & \frac{\partial y^m(i+3)}{\partial u(i+2)} \end{pmatrix} \begin{pmatrix} (y^r(i+1) - y^m(i+1))Q(1) \\ (y^r(i+2) - y^m(i+2))Q(2) \\ (y^r(i+3) - y^m(i+3))Q(3) \end{pmatrix}, \qquad (5.83)$$

$$\nabla_{u|_{i+1}^{i+2}} J_b = \begin{pmatrix} R(1) & -R(1) & 0 \\ 0 & R(2) & -R(2) \\ 0 & 0 & R(3) \end{pmatrix} \begin{pmatrix} u(i) - u(i-1) \\ u(i+1) - u(i) \\ u(i+2) - u(i+1) \end{pmatrix}. \qquad (5.84)$$

Figure 5.17 shows the error response between the output of the system and the reference trajectory for $\lambda = 0$, $N_1 = 1$ and $N_2 = 6$. In this case a perfect

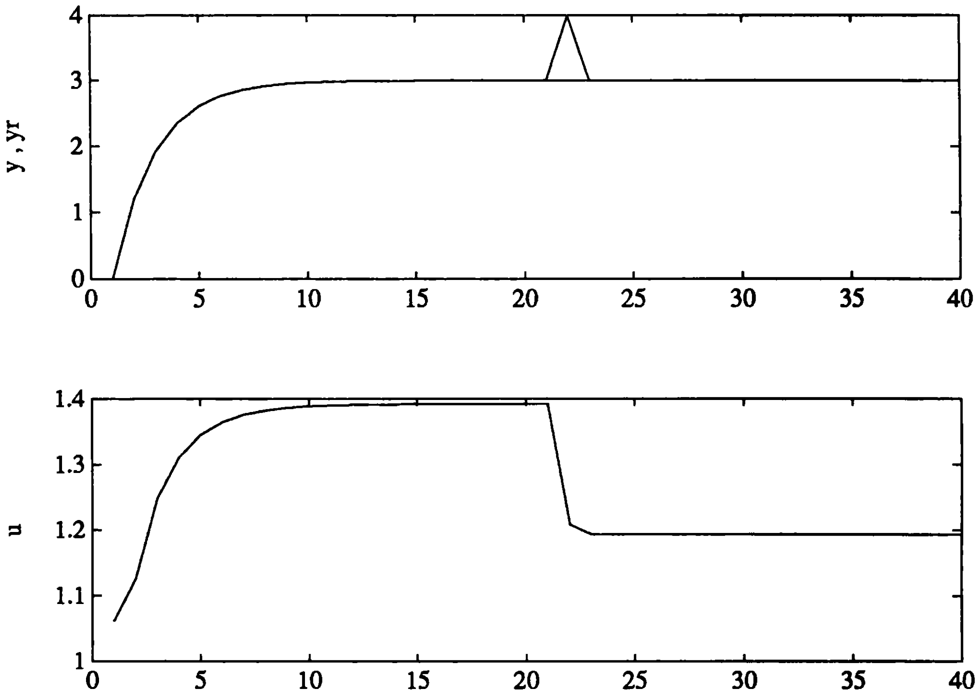


Figure 5.17: Error between response of the system and reference trajectory and control signal for $\lambda = 0$.

tracking of the reference is obtained, which means that the controller is acting as the inverse of the plant plus the reference model.

Figure 5.18 shows the response of the system and the reference trajectory for $\lambda = 30$, $N_1 = 1$ and $N_2 = 6$. In this situation it is possible to have a certain error in the output, so the controller is not the inverse of the plant plus the reference system but some "detuned" approximation.

5.5 Summary

In this section we compare the different approaches according to their basic characteristics and the type of systems for which the are applicable, in particular we answer the following questions for each method.

$Q1$. Has a scheme any robustness to unmeasured disturbances and modelling errors?

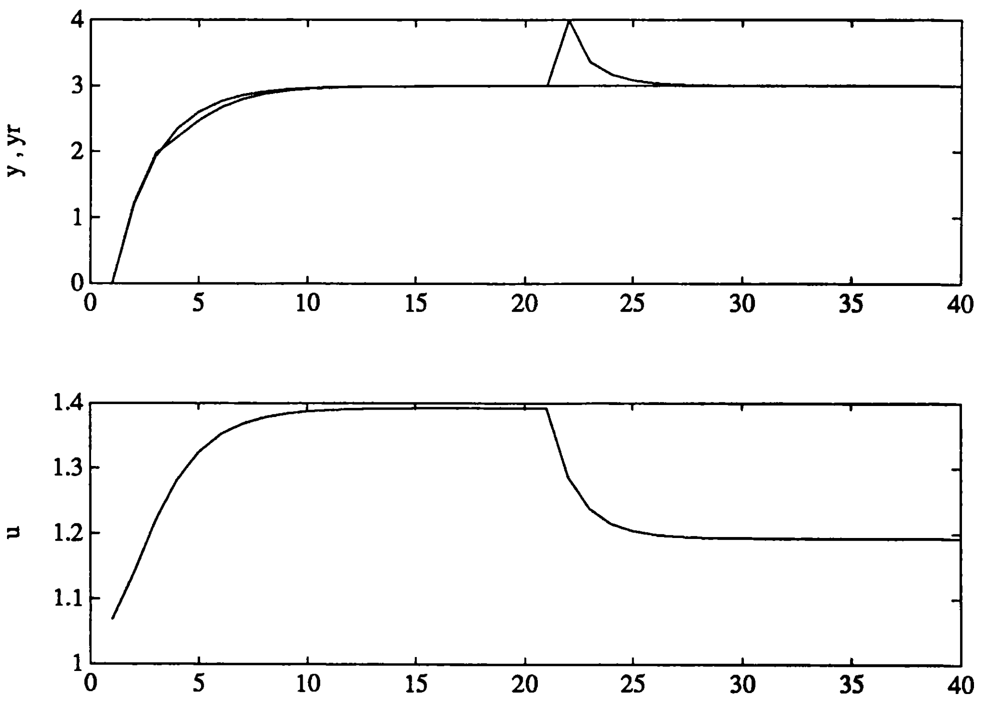


Figure 5.18: Error between response of the system and reference trajectory and control signal for $\lambda = 30$.

$Q2$. Does the method consider the rejection of disturbances?

$Q3$ Can the method be applied to control unstable systems?

$Q4$ Is the closed-loop internal stability guaranteed for non-minimum phase systems?

$Q5$ Is the closed-loop stability guaranteed ?

The answer to these five questions are summarised in Table 5.1, an (*) indicates that the available theory is limited, and an (–) indicates that question is not relevant for that structure.

5.6 Conclusions

This Chapter has outlined different approaches known in linear control theory, which have been extended to deal with nonlinear systems. It seems that the same

Method	$Q1$	$Q2$	$Q3$	$Q4$	$Q5$
Inverse	no	no	no	no	–
Model Reference	no*	no	no	no	no
Internal Model	yes	yes	no	no	yes
Predictive Control	yes*	yes	yes	yes	yes

Table 5.1: Summary of characteristics.

limitations and advantages of these approaches in linear system are carried over to the nonlinear counterpart. Firstly, the inverse control approach was investigated pointing out the limitations due to minimum-phase characteristic and reachable regions, and how to overcome them. The main disadvantage of this approach is its open-loop nature, which can be overcome using a linear controller. This approach is similar to the linearising approach [82] in that the linear controller is designed just considering linear dynamics. The main difference stems from the fact that using an inverse there is no need of calculating the derivatives of the functions in each operational point.

Secondly, the Model Reference control approach was described as the problem of solving a functional equation. From this point of view several algorithms were derived and the convergence for a particular case was proved.

The use of connectionist networks for nonlinear Internal Model Control was also explored, where the model of the plant is used directly in the loop to estimate the effect of disturbances.

And finally, the use of connectionist representations in the Receding Horizon control approach has been addressed. The results show that this approach is feasible for application and the method also has the flexibility to deal with some practical issues such as disturbances and modelling error.

Chapter 6

Applications

SUMMARY

In this Chapter two potential industrial applications are described. The first one is the classical pH control problem, and the second one is the control of a rolling mill. For each application a comparative study among the different nonlinear strategies and linear solutions is done.

6.1 pH Control

In this section, the control of a pH plant using connectionist representation is addressed. Four nonlinear control schemes based on connectionist representation of the controlled plant are developed and compared. It is shown that it is possible to achieve better results with the connectionist approach than with a linear control scheme. Some limitations imposed by the nonlinearity of the plant and the degree of approximation of the model over the different connectionist approaches are also studied.

6.1.1 Basic Model

The neutralisation is a chemical process involving significant nonlinearities. This feature can be so severe that classical linear feedback control does not always lead to a satisfactory performance. In the simplest form the process consists of a stirred tank in which waste water from a plant is neutralised using a reagent, usually a strong acid or a strong base (see Figure 6.1).

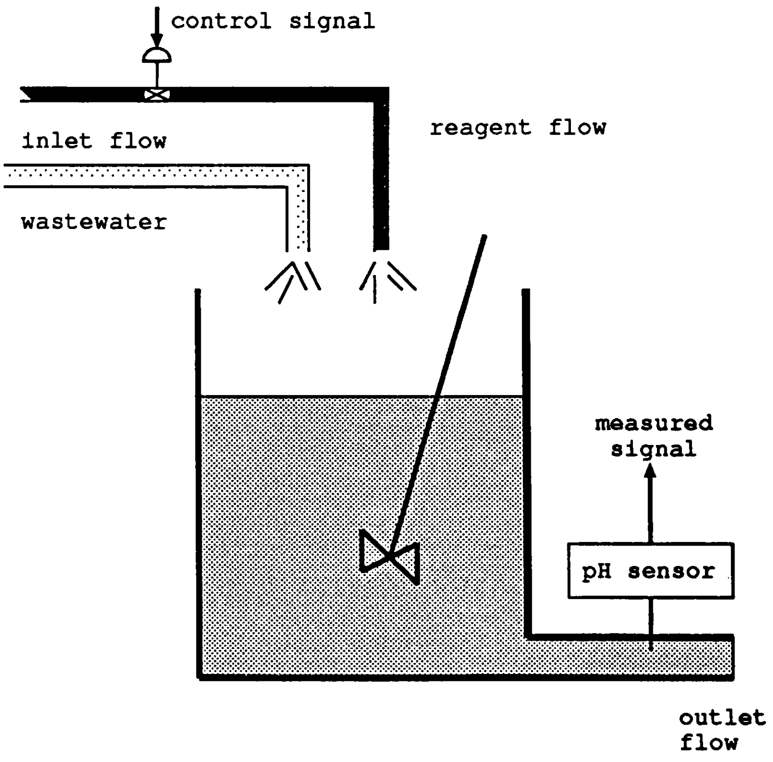


Figure 6.1: Process diagram for the pH plant.

The following assumptions are used:

- The inlet flow of waste water is constant and much greater than the control flow.
- The dynamics of the valve are negligible.
- The chemical reactor is well stirred.
- The chemical reactions are much faster than the mixing dynamics.

Let $X^a(t)$ be the concentration of a substance a in the reactor tank, $X_{in}(t)$ the inlet concentration of a, q_{in} the inlet flow, and V the volume of the waste water in the reactor tank. A mass balance for the substance a gives

$$V\frac{dX^a(t)}{dt} = X^a_{in}(t)q_{in}(t) - q_{out}X^a(t),$$

where q_{out} is the outlet flow. A mass balance of the reagent in the reactor gives

$$V\frac{dc(t)}{dt} = -c(t)q_{out} + c_{in}q_{in}(t).$$

The basic relation between the concentration $c(t)$ and pH is given by the titration curve. In this example, a process studied in [92] will be used. It concerns neutralisation of waste water, containing amonia and sodium hydroxide, by hydrochloric acid.

The pH is measured with a pH-sensor that is assumed to be faster than the dynamics of the process. This means that the measurement equation is

$$y(t) = pH(c(t)) + w(t),$$

where $w(t)$ represents the noise. From this equation it can be seen that it is reasonable to describe this process by a Wiener model. The titration curve used in the simulations is shown in Figure 6.2.

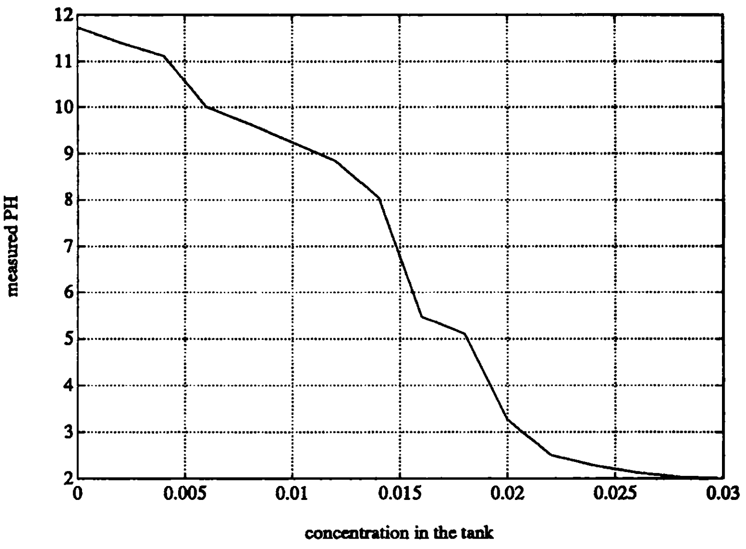


Figure 6.2: Titration curve.

The discrete transfer function used is given by

$$H(z^{-1}) = \frac{0.003626z^{-1}}{1 - 00.8187z^{-1}}.$$

6.1.2 Linear Control

A linear discrete PI controller is described by

$$u(t) = K(1 + \frac{T_s}{T_I}\frac{1}{1 - z^{-1}})e(t);$$

T_s denotes sampling time, K and T_I the parameters of the controller, and $e(t) = r(t) - y(t)$. The system is controlled by this conventional controller with no compensation for the static nonlinearity.

The system was first linearised around an operating point where the slope of the static linearity is

$$\frac{\partial pH(c(t))}{\partial c(t)} = -200$$

resulting in the following transfer function

$$\hat{H}(z^{-1}) = -200\frac{0.003626z^{-1}}{1 - 0.8187z^{-1}}.$$

The gain K and the integration time T_I were determined to give the linearised, closed loop system a double pole at $z = 0.8$ [92]. This leads to the following parameters

$$K = -.0246, \qquad T_I = 89.4.$$

The closed-loop responses for different setpoints with this linear regulator are shown in Figure 6.3. Notice the oscillations in those zones where the gain is too high, producing limit cycles.

6.1.3 Identification

The model of the plant in this case is a Wiener model, i.e. a linear dynamical model followed by a static nonlinearity (see section 4.1, page 73, formula (4.15)). This model has a recursive input-output representation provided that the nonlinearity is invertible [45]. The structure of the model is given by a nonlinear relation between the past value of the input and past value of the output,

$$P: \qquad y(t+1) = f(y(t), u(t)).$$

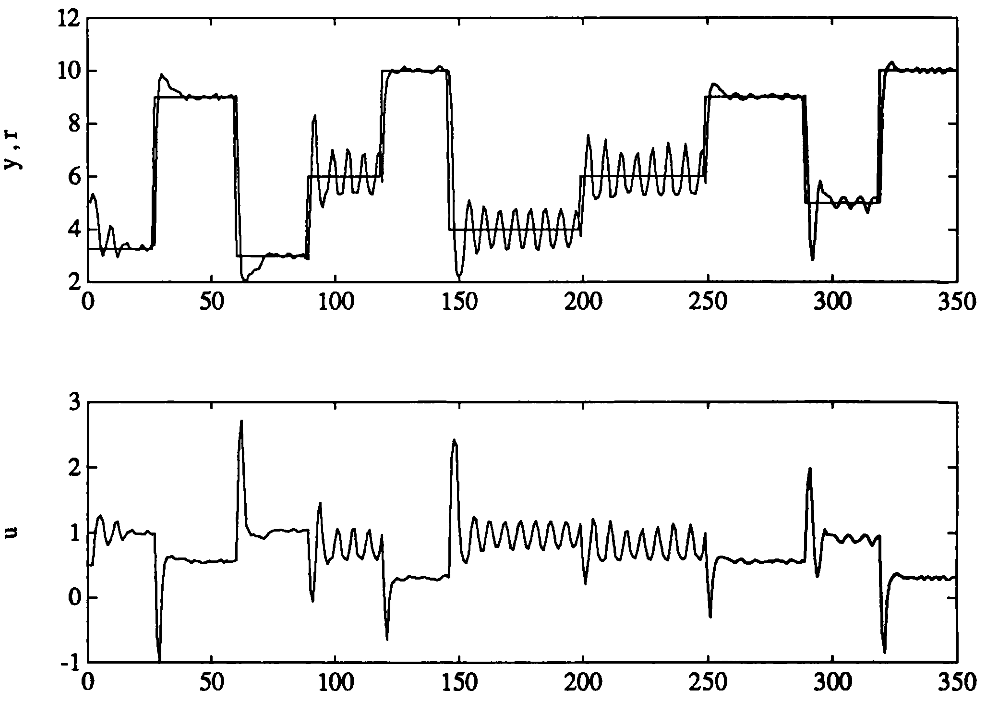


Figure 6.3: Response of the system under conventional PI controller.

A connectionist model was used to model the system. The model uses gaussian units and is described by

$$\mathcal{N}: \qquad y(t+1) = \sum_{i=1}^{N} c_i g_{m_i,\sigma_i}(y(t), u(t)), \tag{6.1}$$

where $g_{m_i,\sigma_i}(y(t), u(t)) = \exp(\frac{-(y(t)-m_i^y)^2-(u(t)-m_i^u)^2}{\sigma_i})$, and m_i^y, m_i^u, and σ_i are parameters to be estimated. In this case the noise has not been modelled.

The connectionist representation was trained to learn the relation between the actual output $y(t)$ and past output $y(t-1)$ and the control signal $u(t-1)$. The identification was done using a batch algorithm and presenting a set of training data consisting of 120 points distributed all over the space of $[y(k-T), u(k-T)]$. In order to obtain a representative training set not only a sufficient excited signal in time but also in magnitude is needed. The maximum deviation between the system and the model of 5% was obtained.

6.1.4 Inverse Control

The first strategy to be considered employs a connectionist representation trained as the inverse function of the plant and used as a controller. There are two approaches to get the inverse. The direct approach assumes the plant to be invertible and the input-output data is used to build the inverse. In a indirect approach, a model is used instead of the real plant in order to generate the inverse [35].

The right inverse of the system $\mathcal{N}_{\mathcal{I}}$ is defined by a gaussian network described by the following relation

$$\mathcal{N}_{\mathcal{I}}: \qquad u(t) = \sum_{i=1}^{N} c_i g_{m_i,\sigma_i}(y(t+1), y(t)),$$

where $g_{m_i,\sigma_i}(y(t+1), y(t)) = \exp(\frac{-(y(t+1)-m_i^y)^2-(y(t)-m_i^y)^2}{\sigma_i})$ was identified using a synthetic signal and 121 units.

In this case the inverse is not defined in the whole domain. In fact there is a reachable region defined by those states which can be reached by one step control [44]. This region is limited by the nonlinearities and the saturation of the controller. The reachable region for the system is shown in Figure 6.4. This means that a regular gaussian network, with its centres fixed in a rectangular grid, is not the most suitable architecture to approximate the inverse.

Once the parameters of the inverse representation have been adjusted, the inverse can be cascaded with the plant. The response of the system is shown in Figure 6.5. As the inverse is not perfect some steady-state errors are present in the response.

To eliminate steady-state errors due to modelling errors and effects of disturbances a PI controller was added as shown in Figure 6.6. The adjustment of the controller only alters the response to disturbances, because the feedforward path gives the desired performance for changes in the set point.

The closed loop response of the system plus inverse is given in Figure 6.7.

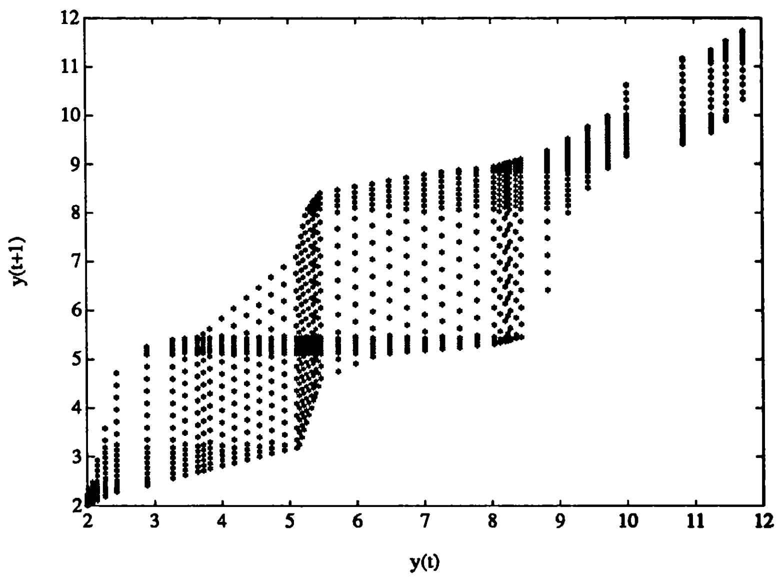


Figure 6.4: Reachable region for the system.

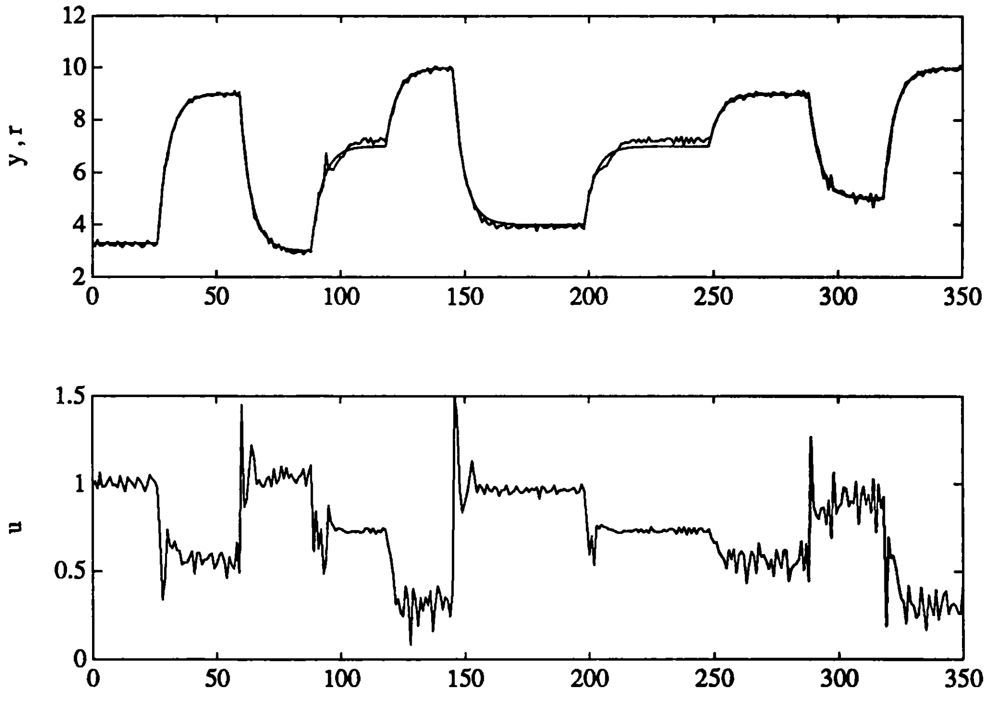


Figure 6.5: Response of the system under inverse controller.

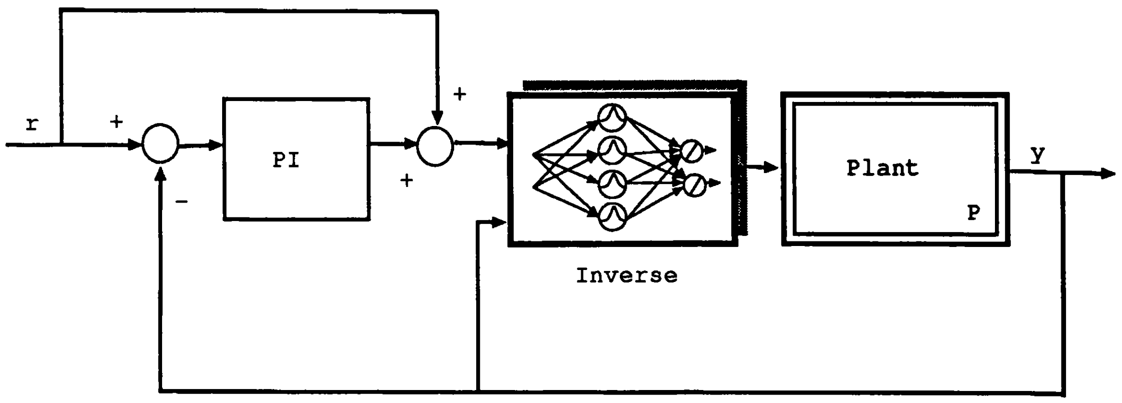


Figure 6.6: Inverse control plus PI.

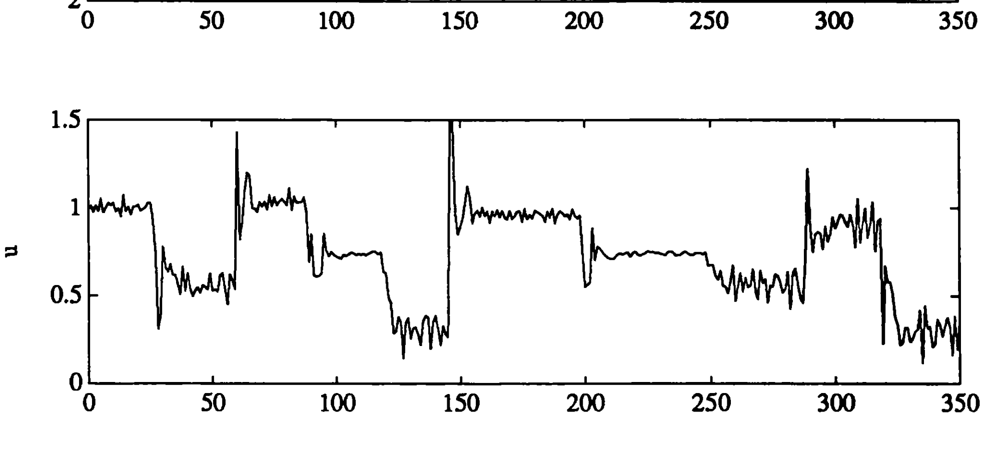


Figure 6.7: Response of the system under inverse system plus PI controller.

6.1.5 Model Reference Control

The second control scheme tested was Model Reference Control. As in [61] the plant P with an input-output pair $\{u(k), y(k)\}$ is given. A stable reference model R is specified by its input-output pair $\{r(k), y^r(k)\}$, where $r(k)$ is a bounded function. The aim is to determine the control input $u(k)$ for all $k \geq k_0$, so that

$$\lim_{k\to\infty} |y(k) - y^r(k)| \leq \epsilon$$

for some specified constant $\epsilon \geq 0$.

Here an indirect approach was used [61], i.e. a model of the system is used to estimate the inverse. The model reference is given by

$$y^r(t+1) = R(y^r(t), r(t)) = 0.8y^r(t) + 0.2u(t).$$

The output error is defined as $e(k) = y(k) - y^r(k)$, the aim of the control is to determine a bounded control signal $u(k)$ such that $\lim_{k\to\infty} e(k) = 0$. If the inverse of the system $\mathcal{N}_{\mathcal{I}}$, that is $\mathcal{N}_{\mathcal{I}} \circ \mathcal{P} = \mathcal{I}$, is known exactly the control signal can be generated as

$$u(t) = \mathcal{N}_{\mathcal{I}}(0.8y(t) + 0.2u(t), y(t)).$$

If the inverse is perfect the dynamics of the error is given by

$$e(t+1) = 0.8y^r(t) + 0.2u(t) - P\left(\mathcal{N}_{\mathcal{I}}((0.8y(t) + 0.2u(t), y(t)), y(t)\right)$$

and simplifying

$$e(t+1) = 0.8e(t), \tag{6.2}$$

which is a stable system. As some modelling errors are always going to be present, the output error $e(t)$ is never going to be zero. Modelling errors act as a forcing function for the system described by equation (6.2).

The structure of the scheme is shown in Figure 6.8. The model reference must be such that the inverse can follow the input signal without falling outside the reachable region.

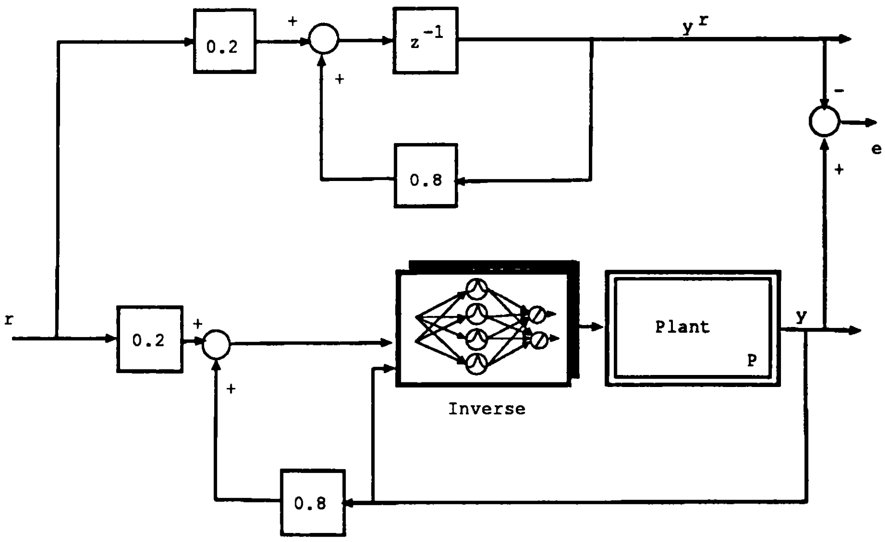


Figure 6.8: Model reference scheme.

The response of the system including noise is shown in Figure 6.9. As can be seen from Figure 6.9 the performance of the controller is the worst in those zones where there is a poor approximation to the inverse, leading to responses with steady-state errors.

6.1.6 Internal Model Control

The third connectionist strategy considered was Internal Model Control. In the structure of this approach two aspects are considered, the control of the plant due to changes in the reference signal and compensation of disturbances through estimation. The general scheme is shown in Figure 6.10, where two networks are included, one representing the model of the plant and the other the inverse of the model. Once the model was obtained; it can be included in a control loop. The inverse of the model must be obtained, to get the inverse a network is trained to learn the inverse relation between the variables. The filter F plays two important roles in the design process: reducing the sensitivity loop against perturbations and unmodelled dynamics, and mapping the input signal to the inverse operator according to the reachable states. There is no theory yet developed to design the filter for nonlinear systems. In this case the filter was chosen as

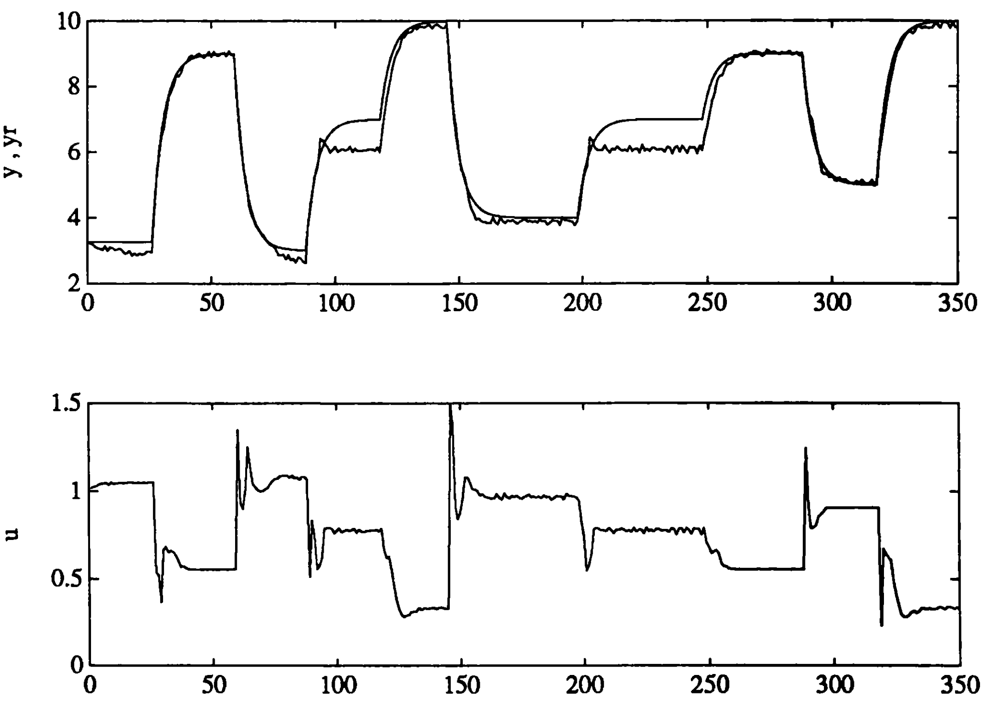


Figure 6.9: Response of the system under model reference approach.

$$F(z^{-1}) = \frac{0.4z^{-1}}{1 - 0.6z^{-1}}.$$

If the gain of F is equal to one and if in steady-state $\mathcal{N}_{\mathcal{I}} \circ \mathcal{N} = I$, then $y = r$, no matter if there are discrepancies between the model (M) and the plant (P).

The Figure 6.11 shows the response of the system following different output set-points, and the control signal. The saturation of the input signal was introduced considering the range of operation and invertibility of the relations.

The main drawback of this scheme is that it can only be applied to a stable plant. In addition, for a linear non-minimum-phase system only the minimum-phase part of the plant can be considered for building the inverse. For nonlinear systems, the extension of this concept is not trivial and includes the zero dynamics as mentioned in section 5.1 of Chapter 5 .

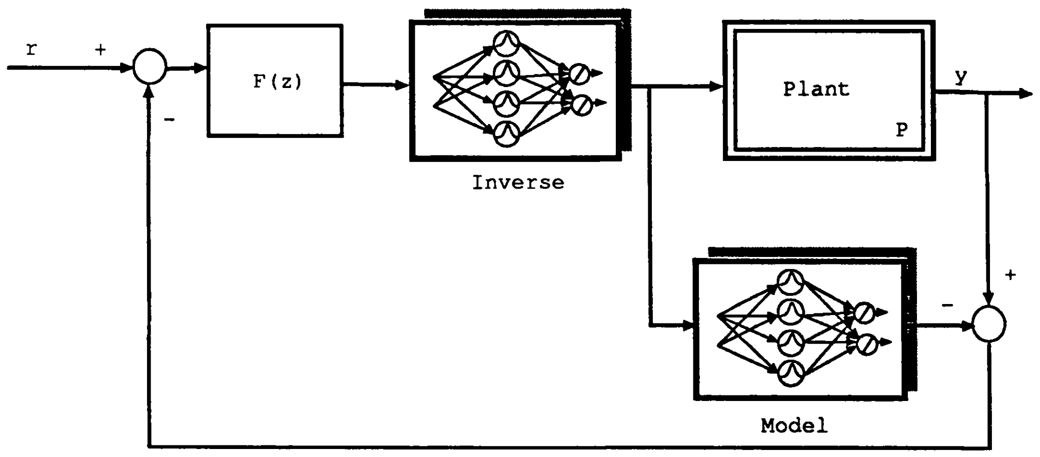


Figure 6.10: Internal model control scheme.

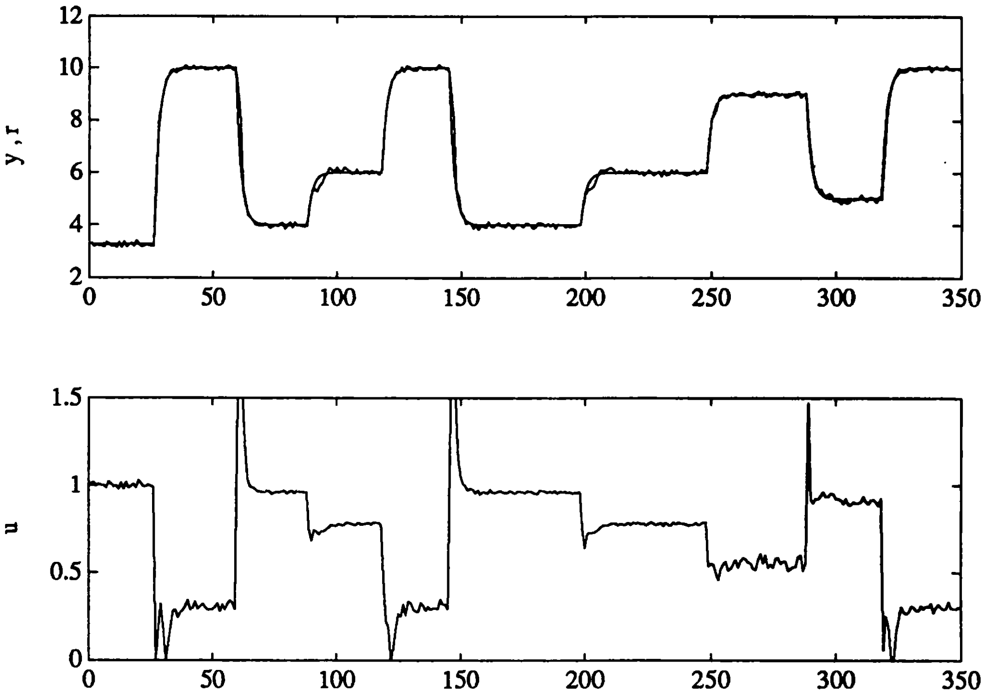


Figure 6.11: Response of the system under internal model control.

6.1.7 Receding Horizon Control

The last strategy investigated was Receding Horizon Control. In this case no inverse is needed to control the system, the control signal is calculated on line. The index to be minimised in this case is a quadratic index defined by

$$J(t,N)=\sum_{i=1}^{N}(y^r(t+i)-y^m(t+i))^2Q(i)+R(i)(u(t+i-1)-u(t+i-2))^2.$$

As there are discrepancies between the model and the real plant an estimation of the disturbance is introduced to assure steady-state error between the reference model and the plant. The stability of the closed loop system is based on a final constraint in the optimisation index [39], this constraint is expressed in terms of the following equality

$$y^r(t+N)=y^m(t+N).$$

The gradients are calculated as

$$\nabla_{u|_t^{t+N}}J=\begin{pmatrix}\frac{\partial y^m(t+1)}{\partial u(t)} & \cdots & \frac{\partial y^m(t+N)}{\partial u(t)}\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \frac{\partial y^m(t+N)}{\partial u(t+N-1)}\end{pmatrix}\begin{pmatrix}(y^r(t+1)-y^m(t+1))Q(1)\\ \vdots\\ (y^r(t+N)-y^m(t+N))Q(N)\end{pmatrix}+$$

$$\begin{pmatrix}R(1) & -R(1) & 0 & \cdots & 0\\ 0 & R(2) & -R(2) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & R(N-1) & R(N-1)\\ 0 & 0 & 0 & \cdots & R(N)\end{pmatrix}\begin{pmatrix}\Delta u(t)\\ \Delta u(t+1)\\ \vdots\\ \Delta u(t+N-2)\\ \Delta u(t+N-1)\end{pmatrix}, \tag{6.3}$$

where $\Delta u(t)=u(t)-u(t-1)$. If the model is a gaussian network, $y^m(t+1)$ is given by equation (6.1), and is a differentiable function.

Figure 6.12 shows the response of the system and the control signal following different output set-points. Introducing a weighting factor to the deviation of the control signal, it is possible to obtain the response shown in Figure 6.13 with less abrupt changes in the control signal.

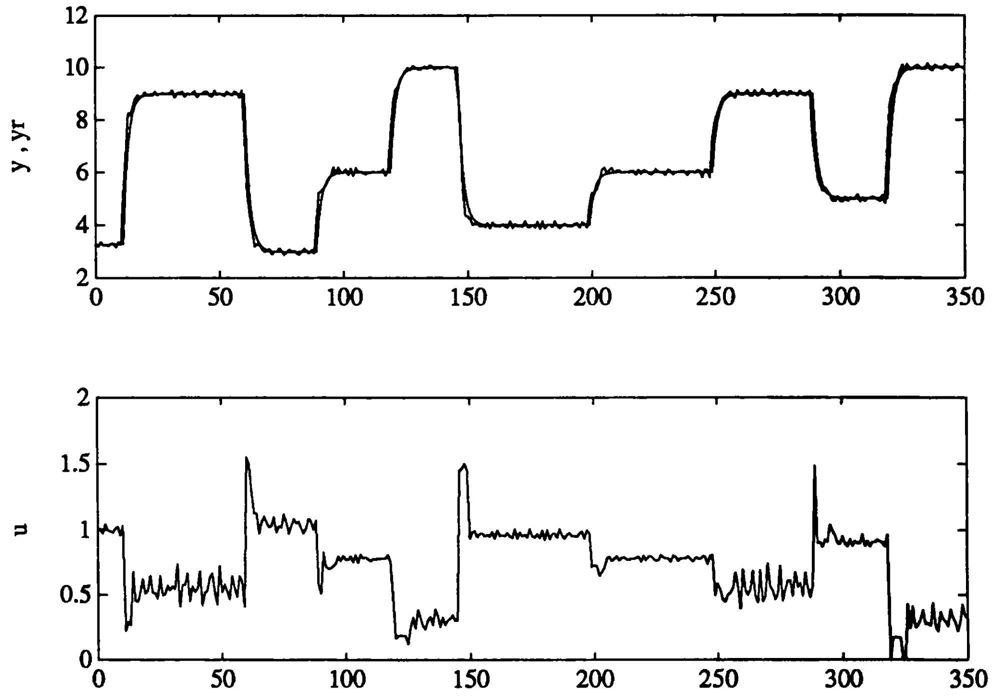


Figure 6.12: Response of the system under receding horizon control, $R = 0$.

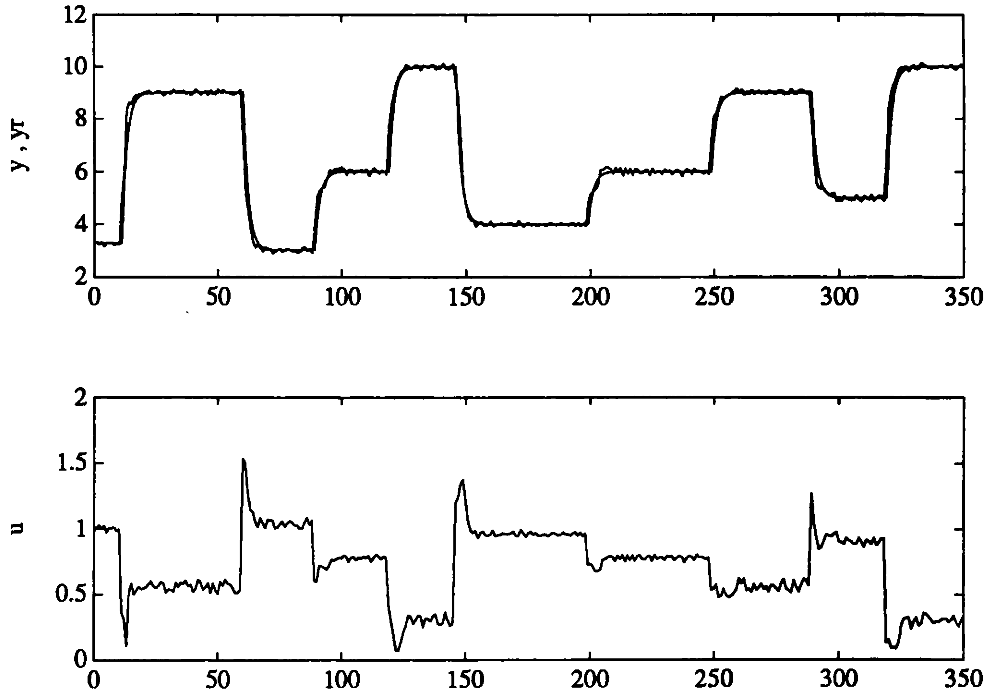


Figure 6.13: Response of the system under receding horizon control, $R = 1$.

6.2 Summary and Conclusions

Even though the pH process is very simple in terms of its dynamic characteristic it is highly nonlinear, which makes it difficult to control with conventional techniques, but amenable to be modelled by a connectionist representation. In particular, this application has revealed some practical issues like the problem of reachable states due to saturation of actuators, and the range of operation for the variables involved when control with connectionist representation is used. Some other issues related with the dynamic characteristic of the process like non-minimum-phase property and time delays need to be addressed.

A regular gaussian net is not efficient for the approximation of the inverse, when the plant is nonlinear and has restrictions on the actuators.

Inverse Control presents an attractive and simple design procedure, but lacks capabilities to cope with disturbances. To overcome this difficulty an external linear controller can be included to improve the performance of the scheme.

Model Reference Control in the form described in this work does not provide the necessary conditions to deal with disturbances. Also, the reference model must be chosen considering the possible reachables states.

Internal Model Control makes an efficient use of the model and its inverse to provide a clear design procedure to control nonlinear systems, overcoming the effect of modelling error by introducing the model within the loop. As this scheme uses the inverse of the system, the restrictions imposed by the reachable states must also be considered.

Finally, Receding Horizon controller gives the necessary degrees of freedom in the design to cope with the problems mentioned before, and additionally the responses are satisfactory.

6.3 Control of Steel Mill Strip Thickness Using Connectionist Models

This section describes different methods of controlling the thickness in a simulated steel mill. The unknown nonlinear relation between the variables has been represented by a connectionist network, which has been trained with input-output data from the simulated plant. In comparison with conventional methods this approach gives better performance against disturbances. Three approaches are considered: PI, Internal Model Control, and Receding Horizon Control.

6.3.1 Rolling Mill Model and Control Problem

The basic physical representation of the process and the relevant variables involved in the control problem are given in Figure 6.14, and summarised here:

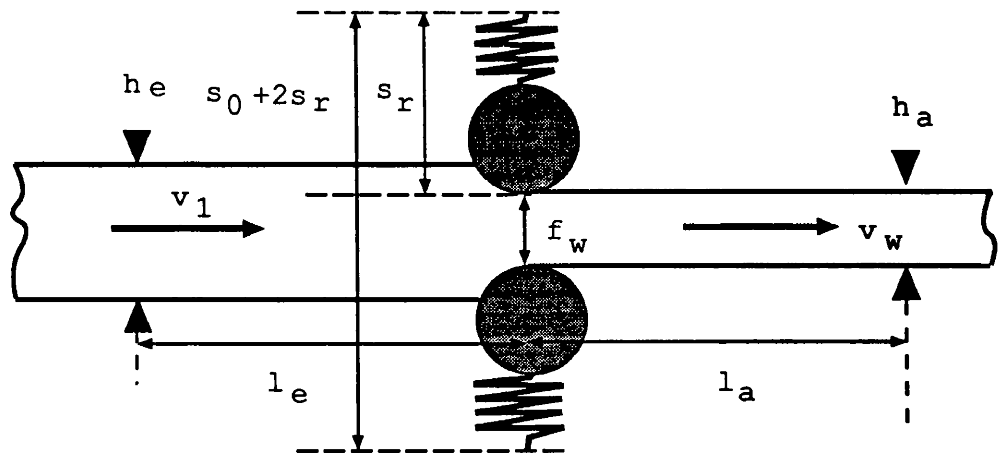


Figure 6.14: General diagram of a rolling-mill.

h_e : strip thickness at entry to stand

h_a : strip thickness at output of stand

s_0 : roll gap with zero rolling force

s_r : dimensions of the roll stand with zero rolling force

v_1 : stand input velocity

v_w : stand output velocity

f_w : rolling force

M' : spring constant

The measured variables are h_e, h_a, v_1, f_w and v_w and the control variable is s_0. As shown in Figure 6.14 the roll dynamics can be viewed as a damped spring system with an effective displacement input of s_0 (the roll gap).

Basic equations - nonlinear model

The basic equation which establishes the relationship among the variables is given by the equilibrium point determined by the characteristics of the material and the working curve of the stand.

The characteristics of the stand can be represented by the linear relation given by

$$f_w = M'(h_a - s_0), \tag{6.4}$$

where M' is constant. The characteristic of the material is given by

$$f_w = f(\hat{K}, \sigma, \mu, R)\sqrt{(h_e - h_a)} + f_e, \tag{6.5}$$

where

$\hat{K}$: mean yield stress $[\frac{M'}{m^2}]$

σ : mean applied tension stress $[\frac{M'}{m^2}]$

μ : coefficient of friction between work roll and strip in roll gap

R : deformed roll radius [m]

f_e : output elastic recovery [N].

In order to simulate the system a simplified version of (6.5) was used,

$$f_w = V_s\sqrt{(h_e - h_a)}, \tag{6.6}$$

where V_s is a function of v_1,

$$V_s = v\frac{3v_1^{max}}{2v_1^{max} + v_1}. \tag{6.7}$$

Here, $v = 111803399[\frac{N}{\sqrt{m}}]$ and $v_1^{max} = 5[\frac{m}{s}]$. The equilibrium point, with $s_0 = s_0^*$, $v_1 = v_1^*$, $h_e = h_e^*$ constants, is given by the intersection of two curves, namely equations (6.6) and (6.4), as shown in Figure 6.15.

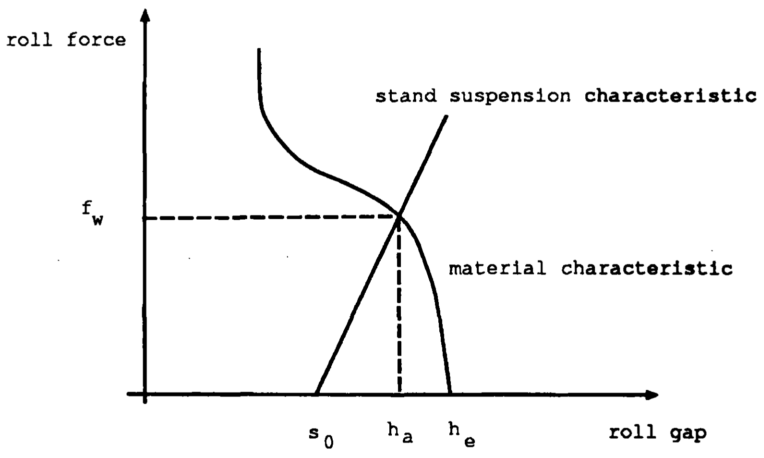


Figure 6.15: Nonlinear relation between roll-force and effective output width.

As equation (6.6) is quadratic it is possible to obtain a closed solution for the intersection point (h_e). In fact, considering

$$M'(h_a - s_0) = V_s\sqrt{(h_e - h_a)} \tag{6.8}$$

and solving for $ds = h_a - d_0$, gives

$$ds = \frac{-b^2}{2M'^2} + \sqrt{c}, \tag{6.9}$$

where

$$\begin{aligned} b &= V_s, \\ c &= \frac{b^4}{4M'^4} - q, \\ q &= \frac{b^2}{M'^2}(s_0 - h_e). \end{aligned} \tag{6.10}$$

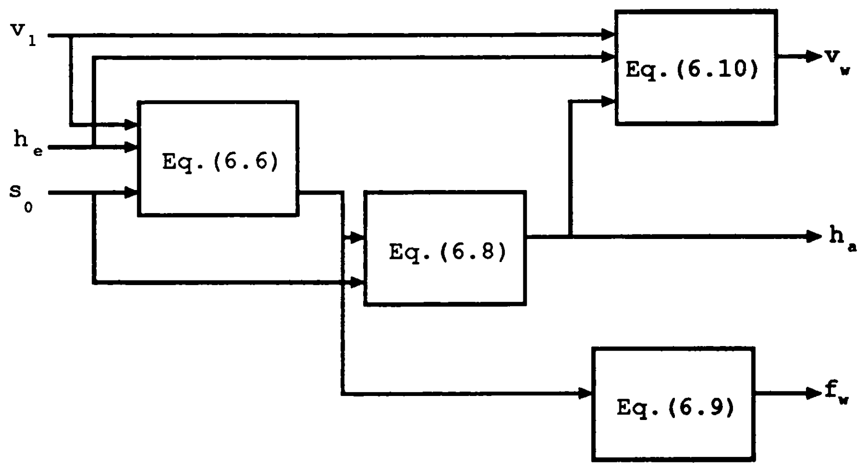


Figure 6.16: General sequence for the solution of the equations.

Then

$$h_a = s_0 + ds \tag{6.11}$$

and

$$f_w = M'ds. \tag{6.12}$$

In addition, the relation between the velocities is given by

$$v_w = \frac{h_e v_1}{h_a}. \tag{6.13}$$

The general simulation scheme is shown in Figure 6.16.

In any realistic simulation it is necessary to include the effect of measurement limitations, namely dead time and noise. If the sensor is located l_a metres from the rolling mill, the dead time associated with the measurements is $T_a = \frac{l_a}{v_w}$. On the other hand, for h_e there is associated a delay related to v_1 given by $T_e = \frac{l_e}{v_1}$. The noise effect was not included in the simulations. All dynamic effects from the actuators and sensors have been ignored.

Linear model

To design a linear controller it is necessary to have a linear model of the plant. The design of the linear model is based on the following linear approximation of equation (6.8):

$$M'(h_a - s_0) = Q(h_e - h_a), \tag{6.14}$$

where Q is the slope of the curve representing the characteristics of the material evaluated at the working point. For practical purposes it must be estimated from the operational conditions.

Defining $k = \frac{Q}{M'}$, the thickness of the output can be expressed as

$$\Delta h_a = \Delta h_e \frac{k}{1+k} + \Delta s_0 \frac{1}{1+k}. \tag{6.15}$$

Defining the constants V and U as

$$V = \frac{\partial h_a}{\partial s_0} = \frac{1}{1+k} \tag{6.16}$$

$$U = \frac{\partial h_a}{\partial h_e} = \frac{k}{1+k} \tag{6.17}$$

the system can be represented by a block diagram as shown in Figure 6.17. In the Figure the measurement delay T_a is included. The operator Δ defines small deviation from the working point.

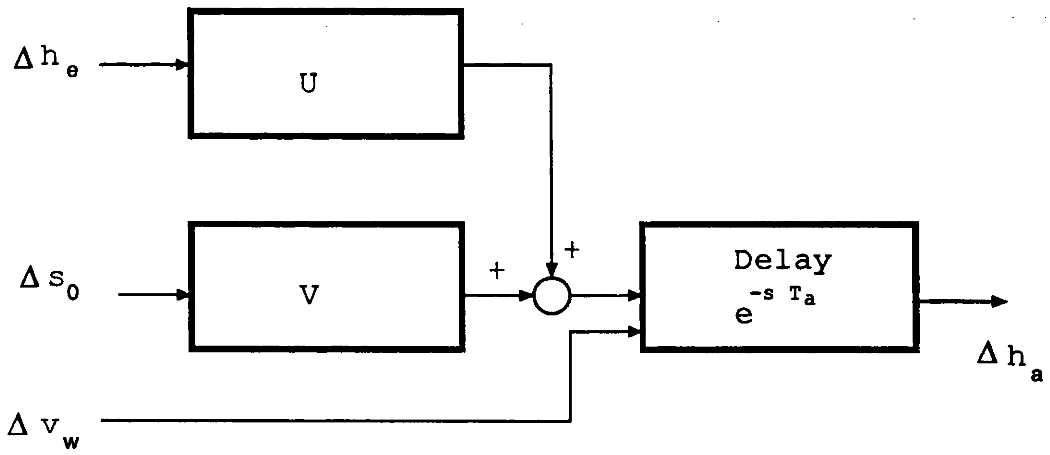


Figure 6.17: Block diagram for the linearised system.

Control problem

The basic control objective is to keep the thickness of the output material (h_a) as close as possible to a reference value. The control variable is s_0 and the measurements available are h_a, h_e, f_w, v_1, and v_w. The evaluation index is the integral of the square error over the time elapsed, that is

$$I = \sum_{t=0}^{t_{max}} (h_a^{ref} - h_a(t))^2. \tag{6.18}$$

The input velocity of the strip is given by a curve with a certain period of acceleration and deceleration as shown in Figure 6.18.

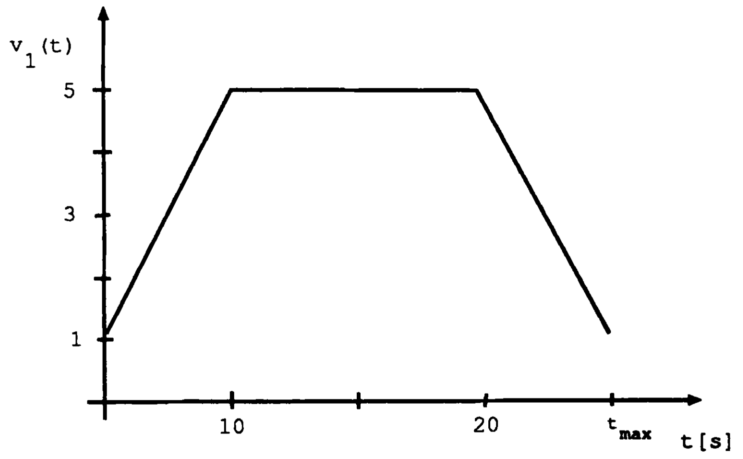


Figure 6.18: Velocity profile.

Simulations

The simulations were carried out using *SIMULAB*. The simulation scheme of the plant including the functions v_1t and h_et, which model the speed profile and the disturbance for h_e, is shown in Figure 6.19.

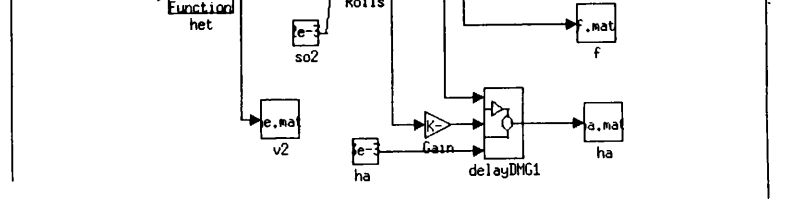


Figure 6.19: Simulation scheme of the plant.

The block called *Rolls* is shown in Figure 6.20.

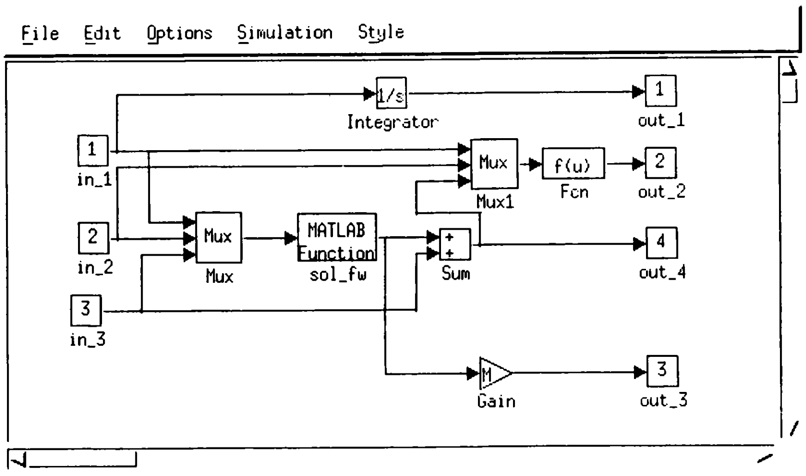


Figure 6.20: *Rolls* block.

6.3.2 Current Control Techniques

The current techniques are based on the linear model described in section 6.3.1. The basic structure has two parts, a feedback controller and a feedforward controller, as shown in Figure 6.21.

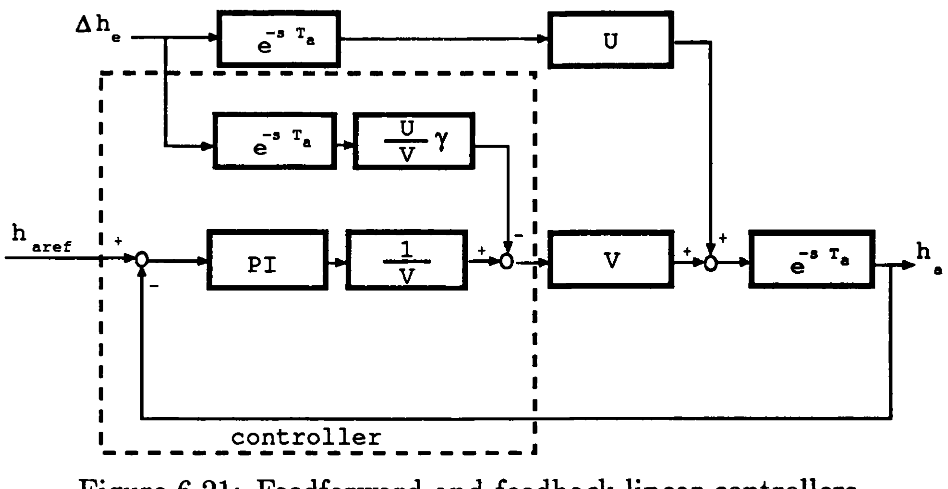


Figure 6.21: Feedforward and feedback linear controllers.

The constant γ takes into account the fact that the cancellation of the measured disturbance terms is not exact. A brief description of the design of the conventional control system is included in order that the connectionist approach developed later can be compared.

6.3.3 Design of the Controller

The basic diagram for the PI controller is shown in Figure 6.22.

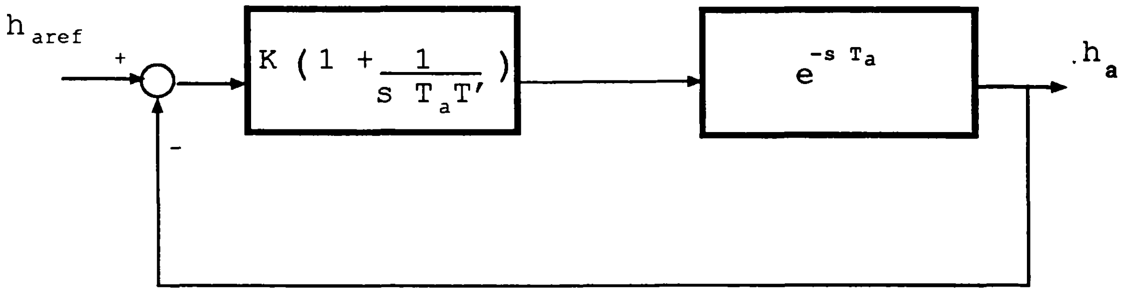


Figure 6.22: Simple PI controller.

The Nichols diagram for the open loop transfer function is shown in Figure 6.23.

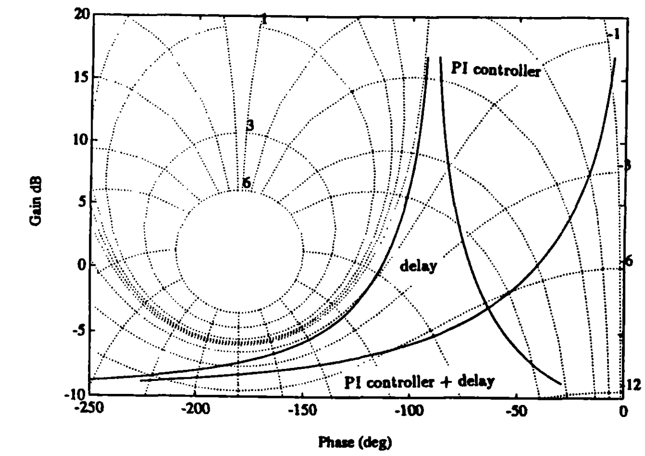


Figure 6.23: Nichols diagram of the system.

In order to have a phase margin of 65^o, the zero of the controller was placed at a frequency of 2.2 $[\frac{rad}{sec}]$ and that gives $T = .454$ and $K = .316$ $(-10dB)$. Considering a delay of $T_d = \frac{m}{v_w}$ with $m = 2$ and a sampling period of 0.01[s], the integral time is $T^* = \frac{T_s}{T} = .006954$ [93].

The response of the system including feedforward controller as well as feedback controller is shown in Figure 6.24

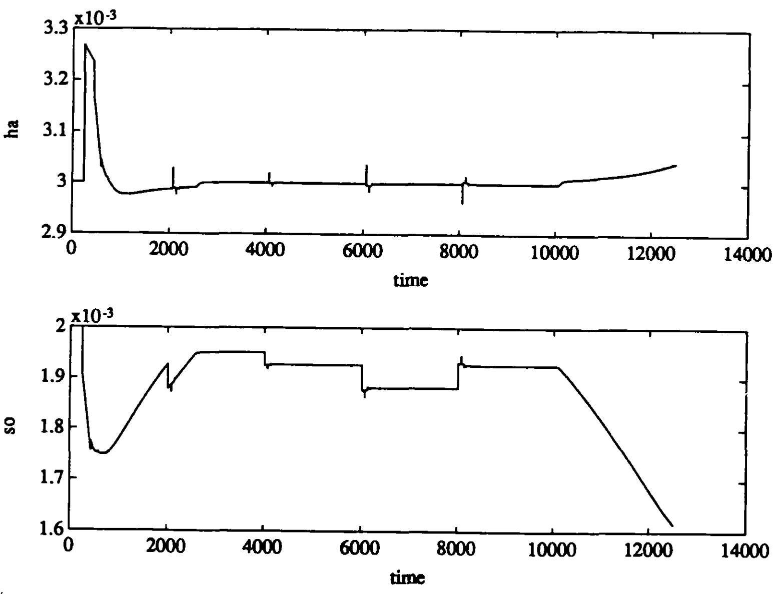


Figure 6.24: Closed-loop response.

On line adaptation

The value of Q can be estimated on-line using an input-output model of the process [93]. As the identification algorithm (an orthogonal recursive least-squares with forgetting factor) takes some time to converge, it is necessary to store Q in a table indexed by the operational conditions. In this way there is effectively a slower adaptation time. The data used for identification are variations of the measurements of f_w and $(h_a - h_e)$.

6.3.4 Connectionist Approach

6.3.5 Modelling

The characteristics of the plant can be respresented by a connectionist network, in this case a Gaussian network described by

$$y = \sum_{i=1}^{N} c_i g_{m_i,\sigma_i}(x), \tag{6.19}$$

where c_i, M_i, and σ_i are the parameters to be adjusted, and x is a vector containing the input variables.

There are several algorithms available to train the network, see Chapter 3. In this case, the data driven smoothing approach was followed. The units were distributed uniformly along the axis of each input variable, then a linear optimization problem was solved several times to find c_i for different $\sigma_i = \sigma$. The tuple $\{c_i, \sigma\}$, which minimizes the square error over the training pattern, was selected. This procedure can be summarized as follows

1 Distribute the centres of units uniformly in the following intervals: $vl \in [1,5]$, $h_a \in [10^{-3}, 6 \times 10^{-3}]$, and $s_0 \in [10^{-3}, 6 \times 10^{-3}]$.

2 Choose a small initial value for σ.

3 Using the training set defined by a set of input-output values $\{X,Y\}$, calculate the c_i as

$$C^* = G^+ Y. \tag{6.20}$$

4 Calculate the residual error $E = (Y - GC^*)^T (Y - GC^*)$

5 If the residual error has decreased then increase σ and go to 3, otherwise end.

In this application two nets with 125 units each were trained to model the plant and its inverse. The first one has the structure shown in Figure 6.25,

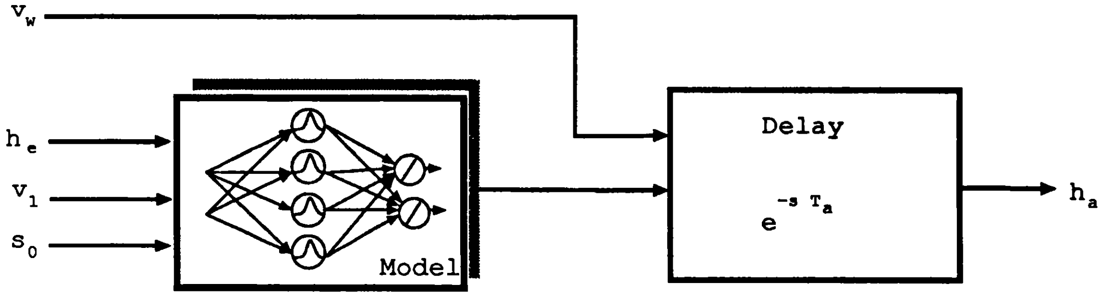


Figure 6.25: Connectionist model of the system.

The second net represents the realizable inverse of the system (in this case the static relation among the variables) as shown in Figure 6.26.

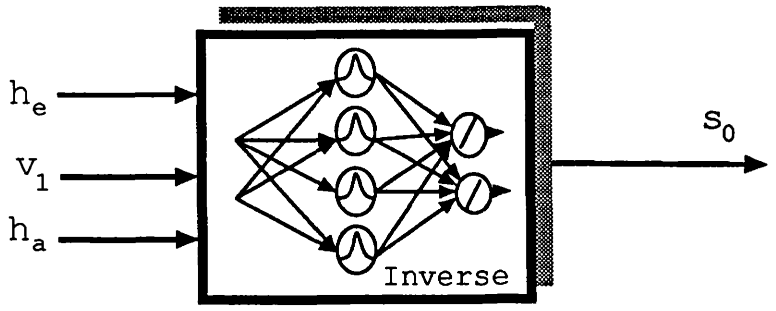


Figure 6.26: Connectionist model of the inverse.

The results of this stage are shown in Figures 6.27 and 6.28. The percentage error in these cases is less than 0.3% for the training patterns.

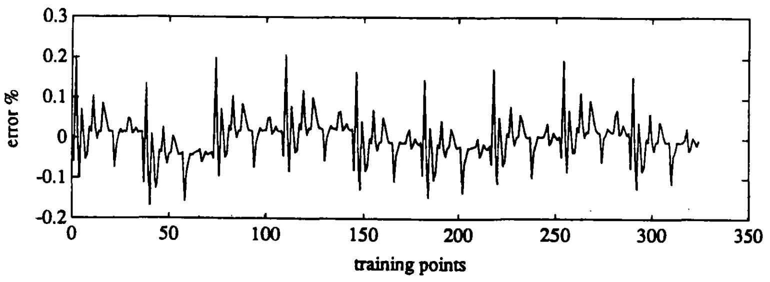


Figure 6.27: Errors for the training set of the model.

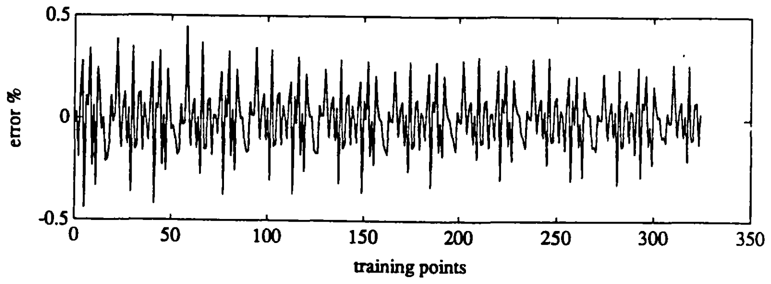


Figure 6.28: Errors for the training set of the inverse model.

Basic structure using PI controller

In this scheme the main idea is to compensate the onlinear relation among the variables by introducing the nonlinear inverse model of the system in the loop. In this sense the PI controller regards the plant as a system with unit gain. If the inverse of the model is not perfect the PI controller helps to reduce the sensitivity of the whole system against this type of error and provides zero steady-state error. The general structure is shown in Figure 6.29.

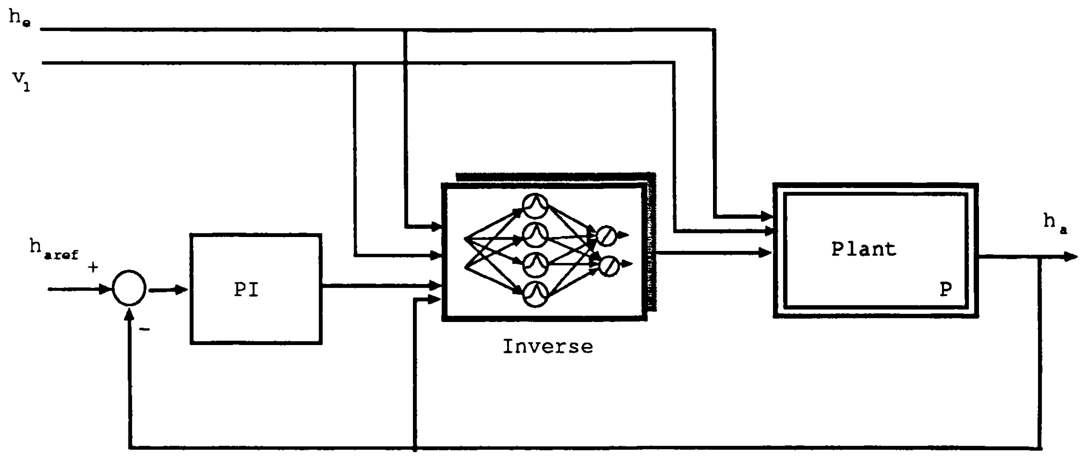


Figure 6.29: PI controller and inverse of the system.

Figure 6.30 shows the results obtained with this approach. It is worth noting that the main difference from the conventional approach lies at the beginning and at the end of the trial. This difference is due to the fact that the knowledge acquired during the training period has been used. The parameters of the PI controller are the same as those used in the linear case, section 6.3.3.

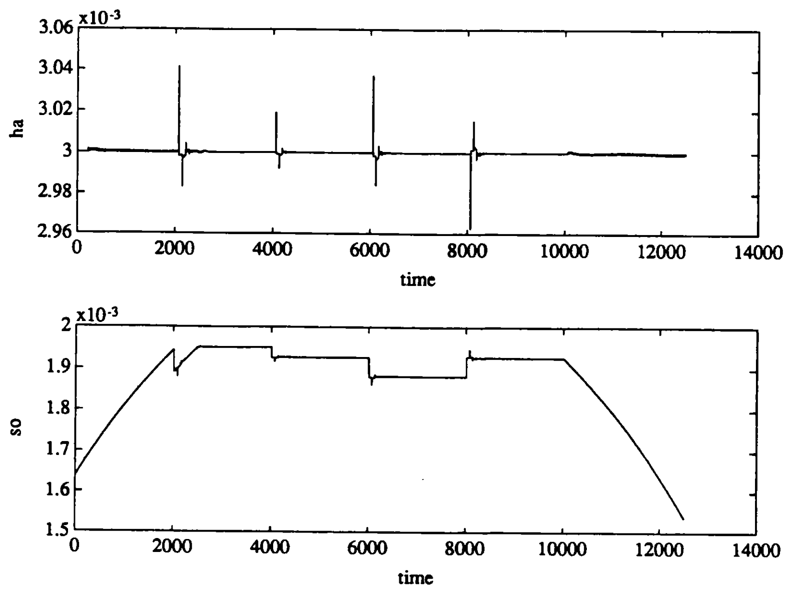


Figure 6.30: Response of the system with PI controller and Inverse model.

Internal Model Control

The basic structure of IMC is shown in Figure 6.31.

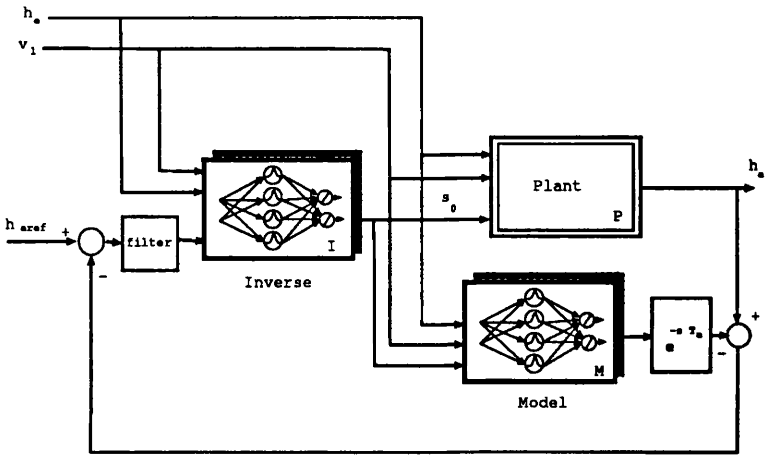


Figure 6.31: Internal Model Control structure.

In this case the model is used in parallel with the plant. In this way the difference takes into account the uncertainty of the model, which is feedback to the system. The design of the filter takes the estimated disturbances into consideration. Typically the filter is a first order filter with unit gain and frequency response tuned to eliminate high frequency noise introduced by measurement devices. As the connectionist inverse model is not perfect there is a steady-state error (as can be seen in Figure 6.32).

In order to eliminate this error it is necessary to have a better inverse. Two possibilities are available: the first one is to increase the number of units for the connectionist inverse representation, and the second one is to include an iterative process, as explained in [36]. The result of this latter modification is shown in Figure 6.33.

Receding Horizon control

The main aim in this approach is to use the solution of a finite dynamic optimization problem in order to generate closed loop control laws. To reach this objective the structure shown in Figure 6.34 is used.

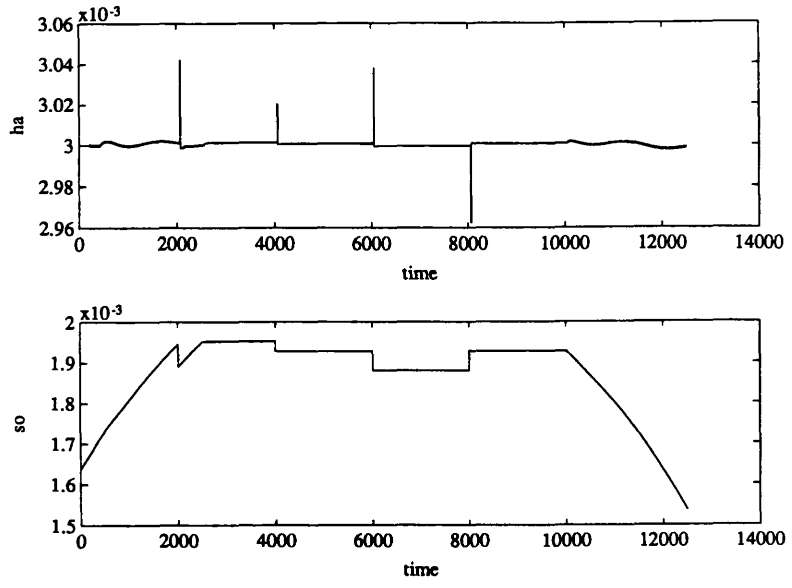


Figure 6.32: Response of the system with IMC.

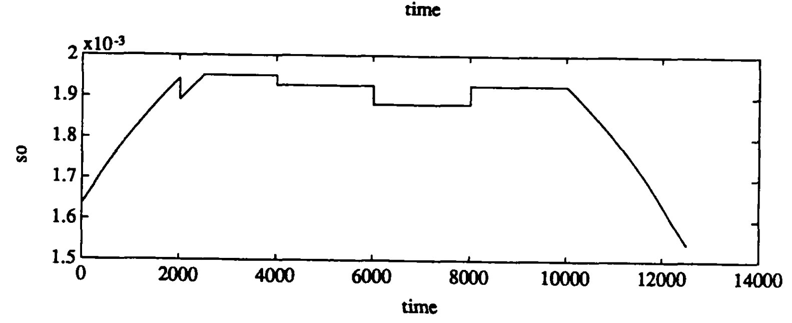


Figure 6.33: Response of the system with IMC including an iterative process.

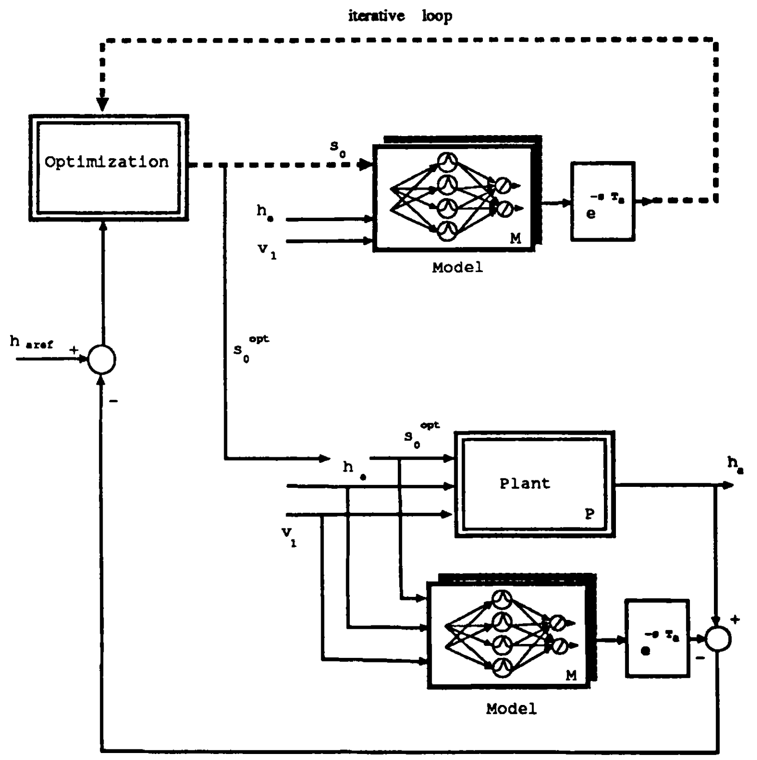


Figure 6.34: Receding Horizon Control structure.

The index to be optimised is

$$J = \sum_{i=j}^{T_h+j} Q(i)(h_{aref} - y_m(i))^2 + R(i)(u(i) - u(i-1))^2, \tag{6.21}$$

where r is the reference value for the output of the system, T_h is the length of the horizon, which must be bigger than the time delay of the system, R and Q are weighting factors, and $y_m(i)$ is the output of the model [77].

A special case of the index defined by

$$J(j) = (h_{aref} - y_m(T_h + j) + d(j))^2 + R(u(j) - u(j-1))^2 \tag{6.22}$$

was used. Additionally, if an estimate of the disturbance $d(j) = y_m(j) - y(j)$ is included in the index, zero error in steady-state is obtained. In fact, considering that the system is in steady-state, that is $u(j) - u(j-1) = 0$ and $y_m(T_h + j) = y_m(T_h + j - 1) = \ldots = y_m(j)$, and replacing these conditions in equation 6.22 we have

$$J(j) = (h_{aref} - y_m(T_h + j) + y_m(j) - y(j))^2 \tag{6.23}$$

and if the minimum of $J(j)$ is 0 then $h_{aref} = y(j)$.

Figure 6.35 shows the result obtained with $R = 10^{-4}$. In this case the response against disturbances is slower than the case when $R = 10^{-6}$, which is shown in Figure 6.36.

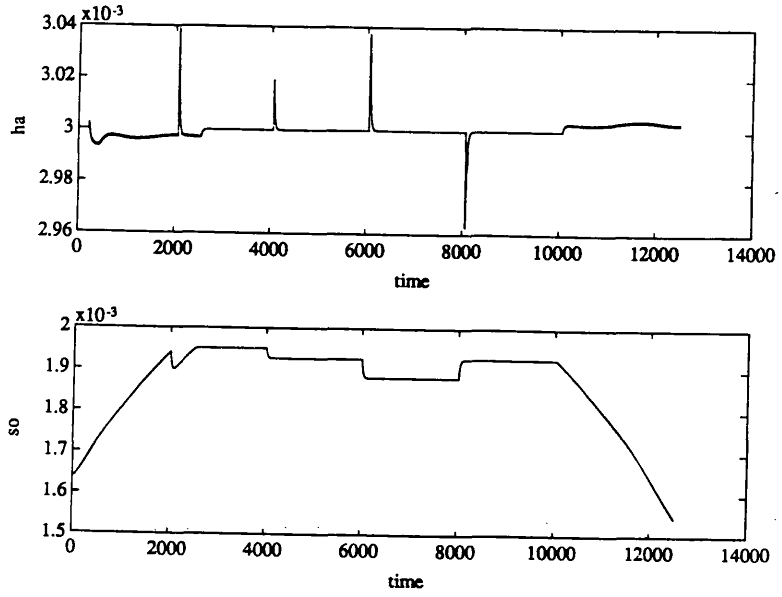


Figure 6.35: Receding Horizon control with $R = 10^{-4}$.

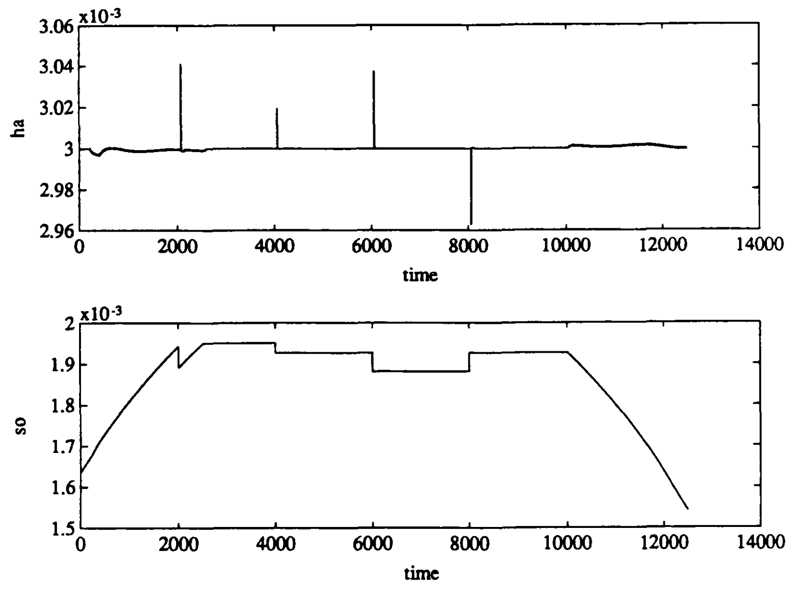


Figure 6.36: Receding Horizon control with $R = 10^{-6}$.

method	Index (eq. 6.18)
PI controller	$1.5989e^{-5}$
PI and inverse model of the plant	$3.1144e^{-8}$
Internal Model Control	$3.6382e^{-8}$
Internal Model Control with iterative procedure	$2.8688e^{-8}$
Receding Horizon Control with $R = 10^{-6}$	$2.8688e^{-8}$
Receding Horizon Control with $R = 10^{-4}$	$1.2017e^{-7}$

Table 6.1: Performance index for the different approaches.

Discussion of results

The results obtained so far are summarized in the table 6.1.

All the methods that use connectionist representations performed better than the traditional PI controller. This result is due to the fact that the information about the speed v_1 was included in the model and, secondly, a compensation for the system working in different operational points is automatically included (without requiring on line identification). The better results obtained with IMC and RHC are due to the fact that a model is used within the loop eliminating the effect of the delay. Both approaches get the same result if the weighting function $R(i)$ for the predictive index is small.

As a general remark, the connectionist approach does not need to differentiate signals and does not need a measurement of f_w.

6.3.6 Further Work

Some of the topics that can be addressed in the future are:

- On line adaptation of the connectionist representations: this extra feature would allow a continuous update of the model to take into account temporal variations.

- Evaluation in noisy environments: as is well known Gaussian functions have some smoothing properties that can be useful to obtain smooth representations from noisy data. Furthermore, it is possible modify adaptation laws to introduce Kalman filter features, i.e. the knowledge of the noise.

- Modelling capabilities to get better estimates of linear coefficients for the traditional PI approach: in this framework the connectionist representation is used to build a nonlinear model among the variables and the derivatives are obtained from the model. This should reduce the effect of the noise on the measured values because there is no differentiation of the incoming data.
- Fusion of information: the operator can have a direct influence on to the connectionist representation to set up certain inputs that can have relevant information for control. One of these inputs can be, for example, the type of steel to be processed.
- Other architectures: the inclusion of dynamics can bring more information and inputs to the Gaussian network. In this case, if there is a explosion in the number of units an alternative architecture must be used, for example, Connectionist Normalized Linear Spline Net [20], Wiener models or others.

6.4 Conclusions

This Chapter has shown that the connectionist approach can be used to address current practical nonlinear control problems with successful results. The main advantage of using this kind of technology is that it provides a general framework to tackle nonlinear control problems. The results obtained with a simulated pH plant and a steel mill are encouraging. In both applications there were structural dependencies among the variables, and therefore it was possible to enbody them into a connectionist representation. In such a case where the parameters or relation between the variables vary with the time, an on line adaptation algorithm is needed. This is a difference compared with traditional adaptive algorithm which were used to compensate not only time variation but also nonlinearities of the system.

Chapter 7

Further Work and Conclusions

SUMMARY

In the final part of this work some suggestions about future research, together with a speculative thought about our potential to understand the organisation of the brain, is given. Some general conclusions are also given.

7.1 Directions for Future Research

The directions for further research are both numerous and wide, but from the perspective of this work it is possible to identify the following areas for future research.

One of the main assumptions used in this work is that the controller uses a functional representation between the variables, but there is another possibility, based on probabilities by which to characterise the system—in this case the modelling effort is placed upon distribution estimation.

Further research is needed to explore the relations between the technique of projection pursuit within the multilayer perceptron, and other well known statistical strategies to deal with multidimensional, nonlinear regression.

Among the control techniques a deeper analysis of the receding horizon approach is required, bearing in mind that the robustness issue is worthwhile pursuing.

Further use of multidimensional signal processing techniques should be fruitful in the future, for instance, a direct application could be the use of Hermite representation for nonlinear dynamical systems.

The use of recurrent networks to model dynamical nonlinear systems must be addressed in some detail, since not only can feedback improve the prediction capabilities of a dynamic connectionist representation, but can also improve the precision of a static map.

Continuous time architectures and learning algorithms can be developed to control and model nonlinear dynamical systems. In this case the tapped delays must be substituted by time operators.

Time scale plays an important role in dynamic systems, and in particular, in adaptive systems. It is generally recognised that in a hierarchical system, the variations in the variables involved in the lower level of the structure have a dynamic response which is much faster than those of the upper levels. This paves the way for a new area of research, where different algorithms can work under different scales of time in order to reach a global objective– as an example of this approach we have the approximation of surfaces using three different algorithms: Kohonen algorithm, interpolation, and adaptation of the linear coefficients. An extension of the latter approach is the use of reinforcement learning as an upper level in a control structure.

From the simulations done it is possible to realise that the convergence time for useful global models are prohibitive in real time. Hence, it will be necessary to combine different schemes to produce on-line learning algorithms, involving the adjustment of the parameters while using them for controlling the system. All the schemes described in this work can be classified as long-term learning, in that they require a long time to reach an acceptable solution. On the other hand, a short-term learning algorithm should deal with the initialisation of the system and with rapidly changing conditions. From this perspective short-term learning algorithms will be mainly local in space as well as in time [6].

7.2 A Speculative Thought

Given that more information from the biological sciences will be available in the future, the people involved in applying this information to solve technological problems are going to be faced with the challenge of recognising the real potentials of these new paradigms to solve problems of this ilk. Perhaps a more formidable challenge is going to be the development of a theory which allows us to recognise the organisation of the brain, or even more generally, to recognize the organisation of a living system, and answer questions like: given a dynamic system, what relations should be observed between its concrete components to determine whether or not they participate in processes that make it a living system?

The following comments not only question the usefulness of the anatomical descriptions, but also the very basis of how to deal with the study of living systems.

The importance of the anatomical descriptions in the understanding of the brain function was conjectured by Kalman [38]:

> It could be true that it is hopeless to try to understand brain functions solely on the basis of anatomy (wiring diagrams). Perhaps the problem will become relatively transparent only after developing a theory powerful enough to give us the main feature of anatomy.

and explained by Maturana [50] in the following way:

> In a strict sense, although the nervous system has anatomical components it does not have functional parts, since any mutilation leaves a functioning unit. Each component of the nervous system that an

observer describes is defined in the domain of interactions of his observations, and is such, that it is alien to the system which it is supposed to integrate.

Neurons are the anatomical units of the nervous system, but are not the structural elements of its functioning. The structural elements of the functioning nervous system have not yet been expressed in terms of invariants of relative activities between neurons.

Models derived from anatomical descriptions can give us some useful perception, but they will not give us a real understanding of the organisation of the brain functions.

The possibility of having different types of representations leads to the question: what is the meaning of the information carried in the brain?. Is it possible for us to understand the "real" meaning of these signals? What type of information is encoded in the signals? All of these questions are related to the key issue of the approach presented in this work, i.e. the representation of the environment. In spite of the advances in Neural Science, our capacity to understand the underlying natural neural processes is biased by our knowledge and concepts. As a limiting factor, perhaps just man's capabilities made by man (including mathematics, logic and linguistic) may be no more than the basis for communication and learning between individuals [89].

Even though, if we have the technology to examine and record all the signals and their different properties (not only voltage, but also concentration of chemicals, magnetic fields, etc.) in the brain, the question of meaning and interpretation of those signals, i.e. the role of the observer, is a more formidable task, because this underlines the basic assessment problem, where the observer is part of the observed phenomena. As Maturana [50] has pointed out, there are clear differences between the role of the observer in the process of explaining the organisation of a man-made system and a living system.

> In a man-made system the system relations (the theory) that integrates the parts that the describer (the observer) defines is provided by him, and as a consequence, these relations appear so obvious to the observer that he treats them as arising from the observation of the parts, and deludes himself, denying that he provides the unformulated theory that embodies the structure of the system which project onto them. In a living system the situation is different: the observer can only make a description of his interactions with parts that he defines through interactions, but these parts lie in his cognitive domain only.

This brief analysis and a few quotations included, not only give a general idea about the deep philosophical implications and problems involved in the study of real living systems and in any development of a theory of the brain organisation, but also help to set up this work in a broader perspective.

7.3 Conclusions

This work has shown that the field of Artificial Neural Network can contribute to the solution of some problems in nonlinear control, providing functional representations and algorithms to adjust the parameters of these functional representations in order to approximate nonlinear systems.

The use of conventional mathematical tools can give not only more insight into the properties of the different representations, but also into the algorithms used to solve the underlying learning problems.

From this perspective, some linear control architectures have been extended to deal with nonlinear systems, finding that the same limitations of the linear versions are apparent in the nonlinear cases. From simulations, and the brief theoretical analysis presented, the Receding Horizon approach seems the most promising and general.

In this work, some concrete applications have been analysed, demonstrating that the approach presented can yield new ways in which to design nonlinear controllers which have better performance than their linear counterparts.

Bibliography

[1] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundation of the potential function method in pattern recognition learning. *Avtomatika i Telemekhanika*, 25:1917–1936, 1964.

[2] A. E. Albert. *Regression and the Moore-Penrose Pseudoinverse.* Academic Press, New York, 1972.

[3] A. E. Albert and L. A. Gardner. *Stochastic Approximation and Nonlinear Regression.* The MIT press, Cambridge, Mass., 1967.

[4] J. A. Anderson and E. Rosenfeld. *Neurocomputing: Foundations of Research.* MIT Press, Cambridge, Massachusetts, 1988.

[5] D. Ballard. Cortical connections and parallel processing: Structure and function. In M. Arbib and R. Hamson, editors, *Vision, Brain, and Cooperative Computation*, chapter 18, pages 665–701. MIT Press, Cambridge, Mass, 1990.

[6] D. F. Bassi. *Connectionist dynamic control of robotic manipulators.* D.Phil. thesis, University of Southern California, 1990.

[7] R. Bellman. *Mathematical Theory of Control Processes.* Academic Press, New York, 1971.

[8] J. K. Benedetti. On the nonparametric estimation of regression functions. *J. Roy. Static. Soc. B.*, 39:248–253, 1977.

[9] J. L. Borges. *Labyrinths.* Penguin Books, London, 1970.

[10] C. C. Chen and L. Shaw. On receding horizon feedback control. *Automatica*, 18:349–352, 1982.

[11] F. Chen. A dead-zone approach in nonlinear adapative control using neural networks. In *Proc. of the 30th IEEE Conference on Decision and Control*, pages 156–161, Brighton, England, December 1991.

[12] S. Chen, S. A. Billings, C. F. N. Cowan, and P. M. Grant. Practical identification of NARMAX models using radial basis functions. *International Journal of Control*, 52(6):1327–1350, 1990.

[13] D. Chester. Why two hidden layers are better than one? In *IEEE Int.Joint Conf. on Neural Networks, IJCNN'90*, pages I 265–268, 1990.

[14] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalised predictive control. part 1: The basic algorithm. OUEL report 1555/84, Oxford University Engineering Laboratory, 1984.

[15] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signal Systems*, 2:303–314, 1989.

[16] H. Demircioglu and P. J. Gawthrop. Continuous-time generalised predictive control. *Automatica*, 27(1):55–74, January 1991.

[17] C. A. Desoer and M. Vidyasagar. *Feedback Systems : Input-Output Properties.* Academic Press, London, 1975.

[18] D. L. Donoho and I. M. Johnstone. Projection-based approximation and duality with kernel methods. *The Annals of Statistics*, 17(1):58–106, 1989.

[19] C. G. Economou, M. Morari, and B. O. Palsson. Internal model control. 5. Extension to nonlinear systems. *Ind. Eng. Chem. Process Des. Dev.*, 25:403–411, 1986.

[20] R. D. Jones et. al. Function approximation and time series prediction with neural networks. In *Proc. IJCNN*, pages 649–665, San Diego, California, July 1990.

[21] K. S. Fu. Learning control systems - review and outlook. *Trans. IEEE on Automatic Control*, 16:210–221, 1970.

[22] K. I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[23] T. Galkowski and L. Rutkowski. Nonparametric recovery of multivariate functions with applications to system identification. *Proceedings of the IEEE*, 73:942–943, 1985.

[24] C. E. Garcia and M. Morari. Internal model control —1. A unifying review and some new results. *Ind. Eng. Chem. Process Des. Dev.*, 21:308–323, 1982.

[25] P. J. Gawthrop. *Continuous-time Self-tuning Control. Vol 1: Design.* Research Studies Press, Engineering control series., Lechworth, England., 1987.

[26] P. J. Gawthrop and D. G. Sbarbaro. Stochastic approximation and multilayer perceptrons: The gain back-propagation algorithm. *Complex System Journal*, 4:51–74, 1990.

[27] A. A. Georgiev. Nonparametric kernel algorithm for recovery of functions from noisy measurements with applications. *Trans. IEEE on Automatic Control*, 30:782–784, 1985.

[28] A. A. Georgiev. Fitting of multivariate functions. *Proceedings of the IEEE*, 75:970–971, 1987.

[29] G. H. Golub and V. Pereyra. The differentiation of peseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10(2):413–432, 1973.

[30] R. W. Gunderson and J. H. George. Error estimates for approximate dynamic systems. *International Journal of Control*, 20(6):971–976, 1974.

[31] J. Hammer. Non-linear systems: stability and rationality. *International Journal of Control*, 40(1):1–35, 1984.

[32] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models.* Chapman and Hall, London, first edition, 1990.

[33] A. S. Hornby. *Oxford Student's dictionary of current English.* Oxford University Press, Oxford, 1978.

[34] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[35] K. J. Hunt and D. Sbarbaro. Connectionist approach to non-linear Internal Model Control using Gaussian approximation. In *Proceedings American Control Conference, Boston, USA*, 1991.

[36] K. J. Hunt and D. Sbarbaro. Neural networks for non-linear Internal Model Control. *Proc. IEE Pt. D*, 138:431–438, 1991.

[37] R. E. Kalman. Design of a self-optimizing control system. *Trans. ASME*, 80:468, 1958.

[38] R. E. Kalman, P. L. Falb, and M. A. Arbib. *Topics in Mathematical System Theory.* McGraw-Hill, New York, 1969.

[39] S. S. Keerthi and E. G. Gilbert. Moving-horizon approximations for a general class of optimal nonlinear infinite-horizon discrete-time systems. In *Proc. 20th Annual Conference Information Science and Systems, Princeton University*, pages 301–306, 1986.

[40] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1987.

[41] T. Kohonen, E. Oja, and M. Ruohonen. Adaptation of a linear system to a finite set of patterns ocurring in a arbitrarily varying order. *Acta Polytechnica Scandinavica*, pages 5–15, 1974.

[42] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems.* Wiley, 1972.

[43] W. H. Kwon and A. E. Pearson. A modified quadratic cost problem and feedback stabilization of a linear system. *Trans. IEEE on Automatic Control*, 22:838–842, 1977.

[44] S. Lee and R. M. Kil. Nonlinear system control based on gaussian potential function network. In *Proc. of the* 1991 *IEEE International Symposium on Intelligent Control*, pages 423–429, Arlington, Virginia, U.S.A, August 1991.

[45] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear system. Part I: deterministic non-linear systems. *International Journal of Control*, 41(2):303–328, 1985.

[46] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear system. Part II: stochastic non-linear systems. *International Journal of Control*, 41(2):329–344, 1985.

[47] L. Ljung. *System Identification — Theory for the User.* Prentice-Hall, Englewood cliffs, New Jersey, USA, 1987.

[48] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification.* MIT Press, London, 1983.

[49] J. Martens. The Hermite transform-theory. *IEEE Transaction on Acoustics, Speech, and Signal Processing*, 38(9):1595–1656, 1990.

[50] H. R. Maturana and F. J. Varela. *Autopoiesis and Cognition.* D. Reidel, Dordrecht, first edition, 1980.

[51] D. Q. Mayne and H. Michalska. An implementable receding horizon controller for stabilization of nonlinear systems. Report IC/EE/CON/90/1, Imperial College of Science, Technology and Medicine, 1990.

[52] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *Trans. IEEE on Automatic Control*, 35:814–824, 1990.

[53] J. L. McClelland and D. E. Rumelhart. *Explorations in Parallel Distributed Processing.* The MIT press, Cambridge , Mass., 1988.

[54] P. Medgyessy. *Decomposition of Superpositions of Density Functions and Discrete Distribution.* Bristol Hilger, Bristol, first edition, 1977.

[55] H. Michalska. *Design of nonlinear control systems: Theory and algorithms.* PhD thesis, Imperial College of Science, Technology and Medicine, 1990.

[56] M. L. Minsky and S. A. Papert. *Perceptrons (Expanded Edition).* The MIT press, Cambridge, Mass., 1988.

[57] C. Moler, J. Little, and S. Bangert. *MATLAB user's Guide.* Mathworks Inc., 1987.

[58] J. Moody and C. Darken. Fast-learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[59] M. Morari and E. Zafiriou. *Robust Process Control.* Prentice-Hall, Englewood Cliffs, 1989.

[60] R. R. Bitmead nad M. Gevers and V. Wertz. Optimal control redesign of generalized predictive control. In *IFAC Symposium, Adaptive systems in control and signal processing*, volume 1, pages 129–134, 1989.

[61] K. S. Narendra and K. Parthasarathy. Identification and control of dynamic systems using neural networks. *IEEE Transactions on Neural Networks*, 1:4–27, 1990.

[62] A. Newell. The Knowledge Level. *Artificial Intelligence*, 18:87–127, 1982.

[63] L. Padulo and M. A. Arbib. *System Theory.* W.B. Saunders Company, Philadelphia, first edition, 1974.

[64] S. R. Parker and F. A. Perry. A discrete arma model for nonlinear system identification. *IEEE Transaction on Circuits and Systems*, 28(3):224–233, 1981.

[65] P. C. Parks and J. Militzer. Convergence properties of associative memory storage for learning control system. *Avtomatika i Telemekhanika*, 50:158–184, 1989.

[66] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33:1065 – 1076, 1962.

[67] D.P. Petersen and D. Middleton. Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and control*, 5:279–323, 1962.

[68] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497, 1990.

[69] E. Polak and D. Q. Mayne. Design of nonlinear feedback controllers. *Trans. IEEE on Automatic Control*, 26:730–733, 1981.

[70] D. Psaltis, A. Sideris, and A. A. Yamamura. A multilayered neural network controller. *IEEE Control Systems Magazine*, 8:17–21, 1988.

[71] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications.* John Willey and Sons Inc., New York, first edition, 1971.

[72] W. Rudin. *Principles of Mathematical Analysis, 3rd Edition.* McGraw-Hill, Auckland, 1976.

[73] I. W. Sandberg. Global inverse function theorems. *IEEE Transaction on Circuits and Systems*, 27(11):998–1004, 1980.

[74] I. W. Sandberg. Global implicit function theorems. *IEEE Transaction on Circuits and Systems*, 28(2):145–149, 1981.

[75] R. M. Sanner and J. E. Slotine. Gaussian Networks for Direct Adaptive Control. In *Proceedings of 1991 American Control Conference, Boston, Massachusetts, USA*, 1991.

[76] S. S. Sastry and M. Bodson. *Adaptive Control: Stability, Convergence, and Robustness.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[77] D. Sbarbaro and K. J. Hunt. A nonlinear receding horizon controller based on connectionist models. In *Proceedings IEEE Conference on Decision and Control, Brighton, England*, 1991.

[78] D. G. Sbarbaro. A comparative study of different learning algorithms for gaussian networks. In *IFAC symposium on Intelligent Components and Intruments for Control Applications. Malaga, Spain*, pages 301–305, 1992.

[79] D. G. Sbarbaro and P. J Gawthrop. Learning complex mappings by stochastic approximation. In *Int. Joint Conference on Neural Networks, IJCNN90*, pages I 569–572, 1990.

[80] D. G. Sbarbaro and P. J. Gawthrop. Self-organization and adaptation in gaussian networks. In *9th IFAC/IFORS symposium on identification and system parameter estimation. Budapest, Hungary*, pages 454–459, 1991.

[81] G. J. Simmons. Iterative storage of multidimensional functions in discrete distributed memories. In J.T. Tou, editor, *Proceedings of the Second Symposium on Computer and Information Sciences - held at Battele Memorial Institute*, chapter 18, pages 261–281. Academic Press, New York, 1967.

[82] J.E. Slotine and W. Li. *Applied Nonlinear Control.* Prentice-Hall, Englewood Cliffs, 1991.

[83] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transaction on Automatic Control,* 26(2):346–356, 1981.

[84] E. Sontag. Feedback Stabilization Using Two-hidden-layer Nets. Technical Report SYCON-90-11, Rutgers University, Department of Mathematics, 1990.

[85] H. W. Sorenson and D. L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica,* 7:465–479, 1971.

[86] E. M. Stein and G. Weiss. *Introduction to Fourier Analysis on Euclidean Spaces.* Princeton University Press, Princeton, New Jersey, first edition, 1971.

[87] Ya. Z. Tsypkin. *Adaptation and Learning in Automatic Systems.* Academic Press, New York and London, 1971.

[88] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of the back-propagation method. *Biological Cybernetics,* 59:257–263, 1988.

[89] I. White. The limits and capabilities of machines — A review. *IEEE Transaction on systems, man, and cybernetics,* 18(6):917–938, 1988.

[90] B. Widrow. Adaptive inverse control. In *Preprints of the 2nd IFAC workshop on Adaptive Systems in Control and Signal Processing, Lund, Sweden,* pages 1–5, 1986.

[91] B. Widrow and M. E. Hoff. Adaptive switching circuits. *1960 IRE WESCON Convention Record, New York:IRE,* pages 96–104, 1960.

[92] T. Wigren. *Recursive Identification based on the nonlinear Wiener model.* PhD thesis, Uppsala University, 1990.

[93] B. Wohler. *Antriebsregelung für eine Waltzenanstellung.* Diplomarbeit, TU-Berlin, Berlin, 1987.

[94] S. H. Żak. Robust tracking control of dynamic systems with neural networks. In *IEEE Int.Joint Conf. on Neural Networks, IJCNN'90*, pages II 563–566, 1990.

[95] R. Żbikowski and P. J. Gawthrop. A Survey of Neural Networks for Control. In K. Warwick, G. R. Irwin, and K. J. Hunt, editors, *Neural Networks for Control and Systems: Principles and Applications*, Control Engineering Series, pages 31–50. Peter Peregrinus, 1992.