NEURAL NETWORKS APPROACH TO PROCESS CONTROL: THE CASE OF PROCESSES WITH LONG DEAD TIMES

C. Meredith McLeod

NEURAL NETWORKS APPROACH TO PROCESS CONTROL: THE CASE OF PROCESSES WITH LONG DEAD TIMES

by

C. Meredith McLeod

Thesis submitted in compliance with the requirements for the Doctor's Degree in Technology: Electrical Engineering, Light Current, in the Faculty of Engineering at Technikon Natal, Durban.

This thesis represents my own work both in conception and execution

C. M. McLeod

APPROVED FOR FINAL SUBMISSION

Professor Vladimir B. Bajić, Pr. Eng., D. Eng. Sc.(E. E.)
Promoter

March 4, 1999
Date

ACKNOWLEDGEMENTS

The tireless encouragement and guidance of my promoter, Professor Vladimir Bajić, is acknowledged with gratitude.

The assistance of Miss Cathy Radhakrishun, Secretary at the Centre for Engineering Development, has been much appreciated, as has been the moral support of fellow-students and the kind encouragement of colleagues.

I acknowledge with thanks assistance received from the Technikon Natal Staff Development programme, the Centre for Research Development, the Centre for Engineering Research, Computer Services and the Library.

Grants in respect of the work represented in this thesis have been received from the Foundation for Research Development, Anglo American & de Beers Chairman's Fund and Technikon Natal.

ABSTRACT

This study relates to applications of static artificial neural networks (ANNs) to two basic problems of process control: (a) process model identification, and (b) optimal controller tuning. The emphasis is on model identification, where several novel techniques are introduced. A review of the use of ANNs for determining optimal controller settings is included as a logical adjunct which would make the complete system suitable for realisation as a portable or networked system.

Three methods for obtaining good approximations for the parameters of first-order processes with long dead time using artificial neural networks (ANNs) are proposed and described. These are termed in this study: time-domain, frequency-domain and model-based methods. In each case the aim was to develop a brief one-shot test that could be applied with minimal disturbance to a closed loop control system. These methods build on existing techniques, but introduce the following novel aspects:

1. The time domain method uses the output waveform arising from a small perturbation pulse but makes use of the entire waveform without preprocessing as input to a static ANN to yield process parameter estimates.

2. The frequency-domain method makes use of the first 81 components of the FFT without further selection as input to a static ANN to yield process parameter estimates.

3. The model-based method uses a simple single-neuron implementation of an ARX model and uses a static ANN to relate process parameter values to the weights of this neuron. In making the analysis, the process input and output are applied repetitively to the neuron model with delays getting progressively larger. Useful effects arising from this are explored.

A technique in which ANN training sets are slightly distorted in a random way during training of a radial basis function is developed as part of the time- and frequency-domain methods. The benefits arising from this technique are demonstrated.

These experimental ANN-based control methods are evaluated by means of simulations in which accuracy in the presence of measurement noise and performance with higher order processes is measured and analysed. Although the main theme of this study is first-order-plus-dead-time (FOPDT) processes, the full autotuning scheme is tested with some representative higher order processes.

Finally, the composition of a complete autotuning scheme is proposed which includes the automatic generation of controller parameters by means of ANNs.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 The Need for Automatic Tuning of Control Systems

It is often the case that plant personnel do not possess enough insight into control loop behaviour to arrive at reasonable settings for the common PID controller. This is borne out by McMillan (1983) who suggests that the answer lies in the adoption of simplified tuning equations and basic rules of thumb for controller parameter selection. A more charitable view is that the problem is one of time rather than lack of ability and that drawn-out procedures are to be avoided (Ruano *et al.* 1992; Yuwana & Seborg 1982). Ultimately, controller tuning is not a trivial operation, and any universal method for automation of this procedure for a large class of processes would clearly be beneficial.

The performance of a control loop in the critical path of plant operations can have a direct impact on production throughput and profitability. Brown (1994) states that in many cases these key loops are not correctly tuned. He presents informal evidence to suggest that the percentage of mistuned control loops in South African plants is at best equivalent to the results of the study by two process control companies in the USA (quoted by Brown 1994) which showed that only 1.4% of 1000 randomly selected

control loops in 100 plants in the USA and Canada were correctly tuned. He goes on to review of the main methods currently used for determining PID controller settings. He dismisses the gain/phase plot as being impractical as a basis for loop tuning because of the mathematical ability required by process operators. The loop tuning procedure based on the ultimate period/ultimate gain method of Ziegler & Nichols (1942) is rejected as not being universally applicable. He points out that a trial and error approach to controller tuning requires an extremely high skill, and is not applicable to slow processes. Self-tuning controllers are reported as not producing adequate results. Brown does however commend a successful autotuning device (unnamed) that is based on the transformation of step response data to the frequency domain.

Regardless of motivation, there is agreement that a rapid, easily applied method for obtaining a good initial controller setting is required. If the scenario described above is true in a general way for all process loops, then the tuning problems are surely multiplied when the process contains significant dead time, even if the designer has provided a suitable compensation device. Shinskey (1979) states that dead time is the ". . . most difficult dynamic element naturally occurring in physical systems". Control systems in which the dead time is larger than the dominant process time constant are often set up with PI controllers because of the difficulty of tuning a Smith Predictor (Hägglund 1992). This avoidance of dead time compensation leads to inherently sub-optimal systems.

The evidence presented above suggests that there is a need for an easy-to-use automatic means of determining suitable controller settings. In the knowledge that significant unavoidable process dead time can be encountered, the proposed method should be valid for dead times that may be many times longer than the dominant time constant of the process. This thesis describes research that was aimed at the prototyping of a system that possesses such characteristics.

1.2 Autotuning vs. Self-Tuning

Automatic tuning can be divided into two categories, autotuning and self-tuning. We adopt the definitions of Shinskey (1995, p.100) who describes a self-tuning controller as "capable of converging stably and promptly to optimum settings without the introduction of an outside disturbance, simply accepting naturally occurring disturbances". In contrast, he defines an "autotuner" as a device that tests the process on demand, using a special test signal, producing a once-only reading. There is no provision for iteration of controller settings towards an optimum.

Autotuning requires the measurement of unknown process parameters and adjustment of controller parameters according to a chosen criterion. The trend towards microprocessor based PID control systems offers opportunities for autotuning methods that were previously impractical or impossible. In process control, artificial neural networks (ANNs) can be used as process models (Warwick *et al.* 1992) and also provide a useful paradigm for the realisation of functions based on empirical or precalculated data that were previously handled by lookup tables and interpolation techniques. An example of the latter can be mapping of process parameter values into the optimal controller parameters (Mamat *et al.* 1995). The ideal autotuning system will function during normal plant operation. This implies leaving the control loop intact and keeping test signals small.

There does not appear to be consensus on which of these two routes is better. Shinskey (1995) makes it plain that he prefers self-tuning controllers. McMillan and Weiner (1987) takes the view that any additions to the traditional stand-alone PID controller should be avoided in order to avoid maintenance problems, but he positively recommends the incorporation of sophisticated control techniques of almost any level of complexity into distributed digital control systems, provided they are incorporated in code that is invisible to the operator.

For the tuning of stand-alone PID controllers an autotuning device could be realised in the form of a portable instrument that is connected to key points in the control system in

which the signals are of electrical form (cf. Brown 1994). The desired control parameters can be visually displayed for reference during manual adjustment. For digital controllers, or in distributed control systems, the functionality can be incorporated in software and invoked as required. This research was directed towards an autotuning method that could be realised in one or more of these three configurations.

1.3 The Research Problem and its Delimitations

We consider the control loop depicted in Fig. 1-1 where r, y and e are the reference signal, output signal and error signal respectively. The aim of the research was to develop an

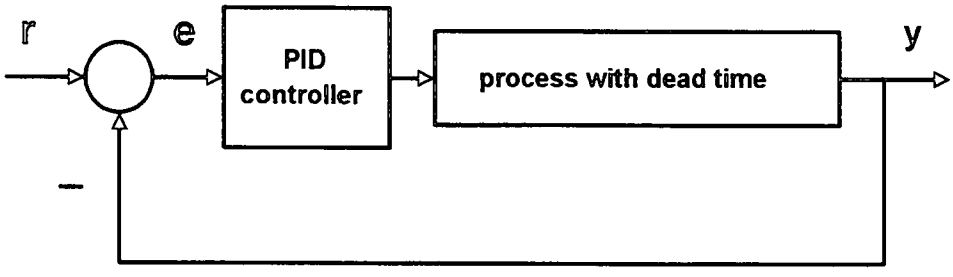

Figure 1-1: Basic PID control system

ANN-based controller tuning method, involving minimal disruption of the process, for first-order processes having long dead time. This method would yield estimates for the three process parameters to a practically acceptable level of accuracy, and the system would also generate estimates for PID controller parameters according to a tuning method selected from several options. Since this research was aimed at developing a method that could be used on demand, the resulting system would be classified as "autotuning".

The research aimed to use static neural networks wherever appropriate. In a quest to keep the approach as simple as possible, pre-processing or selection of the measured data was kept to a minimum. It is acknowledged that considerable prominence is given to dynamic ANNs in control. One of the main factors leading to this has been the

simplicity with which process models or functions can be implemented by on-line training that is not disruptive to the process (Shinskey 1995). It was however decided, after preliminary successes with identification based on static ANNs, to further explore static ANN techniques for processes with long dead time.

The criteria used to judge performance were:

- accuracy
- behaviour in the presence of measurement noise
- applicability to a representative set of self-limiting, monotonic higher order processes.

The resulting system should be capable of realisation in a portable format based on digital signal processing (DSP) technology or as part of a stand-alone or distributed digital control scheme. Successful implementation of such a device would provide economic benefits in terms of improved throughput and product quality in plants where manpower or skills are lacking for tuning control loops by conventional means.

The delimitations that were placed on this study are as follows:

(a) This study attempted to make all measurements in the closed loop because this is in principle less disruptive than breaking the control loop. The approach holds out the possibility of process identification on a system while it is in operation with minimal disruption of production.

(b) The study placed emphasis on time-invariant processes that are modelled in experiments using a first order transfer function and a pure transport delay, i.e. the process was assumed to have a transfer function G_{proc} of the following type:

$$G_{proc}(s) = \frac{K_{proc}}{T_{proc}s+1} e^{-L_{proc}s} \qquad (1)$$

where K_{proc} is the process gain, T_{proc} is the process time constant, and L_{proc} is the dead time. This simplifying assumption was made because many industrial processes exhibit a monotonic and self-regulating open-loop step response that closely resembles the open-loop step response of a first-order system with the exception of a small delay which can

be represented by a dead time component (De Carvalho 1993, p.111). A set of three higher order processes adapted from Hägglund (1992) were used to provide an indication of the generality of the methods.

(c) This study makes use of computer-based simulation of processes, controllers and ANNs. The MathWorks products Matlab 4.2c.1 (1994), Simulink 1.3c (1994) and Neural Network Toolbox 2.0b (1994) were used in conjunction with an Intel Pentium 200MHz microcomputer with 256MB RAM. The simulation of controllers and/or ANNs is fully justified by the intended practical implementation of the techniques presented which in all cases will be microcomputer- or microprocessor-based.

(d) All dead times were assumed to be time-invariant. This simplification is defended by asserting that any method developed to deal with a constant dead time can, in principle, be adapted to slowly-varying dead time or dead time that is functionally dependent on other process parameters.

(e) For convenience, all experiments in this research used times measured in seconds. Restrictions were then placed on the range of values permitted. The methods presented can be adjusted to suit applications of any time scale as long as equivalent ratios of minimum and maximum permitted values are maintained.

It is asserted that these delimitations reduce the complexity of the task whilst impacting minimally upon the validity of the proposed method.

Chapter 2

BASIC CONCEPTS AND DEFINITIONS

2.1 Dead Time

2.1.1 Pure and Apparent Dead Time

In many industrial processes there is a significant delay between a change of applied control signal and the first indication of a corresponding change in the controlled variable. This delay is called *dead time* (McMillan 1983). This effect can generally be ascribed to one or both of two factors: (1) a pure time delay caused by a distance-velocity constraint, off-line measurement, or equivalent; and (2) the presence of second- or higher-order system components that cause the output to remain close to its initial value for such a long time as to produce a close approximation to a distance-velocity lag (Ziegler & Nichols 1942).

In summary, the dead time exhibited by a process can be considered as consisting of an "actual dead period", an "apparent dead period", or a combination of both.

Terms used in the literature to describe these quantities are inconsistent. The following definitions relating to linear processes will be used throughout this document:

Dead time:

If, for purposes under consideration, it is decided to model a process exactly or approximately by a transfer function containing the factor $e^{-L_{proc}s}$ or mathematical equivalent, the process is regarded as exhibiting *dead time* L_{proc}. (Adapted from de Carvalho 1993).

Pure dead time:

If the dead time (or a portion of the dead time) of a process is known to be due to an actual transportation effect or strict equivalent thereof, then that dead time (or portion of dead time) may be referred to as *pure dead time.*

Apparent dead time:

In the context of deriving a process *model,* the retarding effect of higher-order components is often converted to a representative dead time. The dead time introduced for this purpose may be referred to as *apparent dead time.*

Expansion of $e^{-L_{proc}s}$ to a fraction with an infinite series as denominator shows that a system with dead time possesses an infinite number of poles (Palm 1986, p.509). Dead time added to a linear process does not make the process nonlinear. The introduction of a time delay to a linear system does not violate the definition of linearity. This applies to closed- as well as open-loop systems.

Ziegler & Nichols (1943, p.437) draw attention to the S-shape of the open-loop step response of a second-order system consisting of two first-order systems in series and suggest that it be approximated by a first-order-plus-dead-time (FOPDT) model by drawing a tangent to the step response curve at the point of inflexion. The tangent itself embodies the approximation to the first-order time constant, T_{proc}. The intersection of the tangent with the time axis provides the dead time approximation L_{proc}. McMillan (1983) cites Lloyd & Anderson (1971) who showed that specific self-saturating systems consisting of up to four time constants could be modelled with negligible error using first-order plus dead time approximation. This result cannot be generalised, but is strongly supportive of the technique.

McMillan (1983) treats dead time in the categories of process dead time, measurement

dead time, and valve dead time. In the context of a chemical plant at the time of writing he cites the chromatograph as a major cause of measurement dead time having a sampling time of 3 to 30 minutes. He recommends the use of a mass spectrometer which can produce an analysis in 10 to 40 seconds. Specific mention is also made of static mixers, agitated vessels, thermowells, digital controllers, pneumatic tubing and sample transportation as contributing to the dead time of a system.

The presence of pure dead time in a process to be controlled requires a particularly small sampling period for the derivative component of the controller, but derivative control is totally inappropriate for processes with long dead time (De Carvalho 1993).

2.1.2 Analysis of Processes with Dead Time

In spite of the linear behaviour of dead time, its presence in a system complicates the analysis considerably. Palm (1986, p.511) derives a set of root locus plotting guides for the analysis of a generalised linear system with dead time. He suggests (p.516) that if the root locus is found to be cumbersome with higher order systems, the Nyquist theorem should be applied. In the frequency domain, dead time decreases the phase in proportion to the frequency, but leaves the gain plot as it would have been without the dead time. This approach is exploited in the frequency-domain method described in Chapter 3 of this thesis.

2.1.3 Long Dead Time

Long dead time in a process to be controlled is defined here as any process dead time (L_{proc}) that is greater than the dominant time constant (T_{proc}) of the process. This definition is linked to Åström *et al.* (1989) quoted in Hägglund (1992) which reports that dead time compensation should be applied to any process in which $L_{proc}/T_{proc} > 1$.

De Carvalho (1993, p.118) compares the performance of four methods for setting up PID controllers without dead time compensation under conditions of long dead time or short dead time. The tuning methods are those due to Ziegler & Nichols (1942, 1943),

Shinskey (1979), and Cohen & Coon (1953). He cites the instance of a system with the following transfer function:

$$G_p(s) = \frac{e^{-0.880s}}{0.183s+1} \text{ [process A]}$$

In this case $L_{proc}/T_{proc} = 4.81$,which is relatively large.

The performance of the system was compared with

$$G_p(s) = \frac{e^{-0.542s}}{6s+1} \text{ [process B]}$$

which has $L_{proc}/T_{proc} = 0.09$.

It was noted that for process A, none of the controller settings was capable of reducing the initial effect of a unit step disturbance which drove the output to its full uncontrolled extent. Shinskey and Cohen-Coon settings with their lower T_d values did however produce the best recovery from the disturbance to process A.

For process B with its very small dead time component, all four methods produced acceptable results.

Smith (1958, p.324) provides an example based on a continuous flow oil heating system which is some distance from the process making use of the controlled product resulting in a dominant dead time. He recommends the use of two control loops, one for control of temperature at the output of the heater, and another based on a temperature sensor at the input to the process being served.

These examples suggest that if a system exhibits a dominant dead time, efforts should be made to revise the design so as to eliminate its dominance. In this text it will be assumed that the dead times being dealt with are those remaining after all avenues of dead-time reduction have been explored.

2.1.4 The Smith Predictor

A well-accepted solution to problems of controlling processes with inherent long dead time is the Smith Predictor (Smith 1958). This may be classified as a model-predictive control (MPC) device, and is the prime example of its class (Shinskey 1995). Chapter 10 of Smith (1958, pp.299-347) specifically covers the stabilisation and control of systems possessing long dead times. As systems incorporating pure dead time Smith cites "production line delays, the delivery of ordered goods, the prediction of expected sales, and the effect of a change in the tax structure on the economy". He also refers to flow of chemicals through pipes and the flow of water in flumes, and the processes in fractionating towers. In aeronautics, he mentions the transmission of disturbances through the air from the wings to the horizontal stabilisers (p.299). His predictor system can be represented as shown in Figure 2-1.

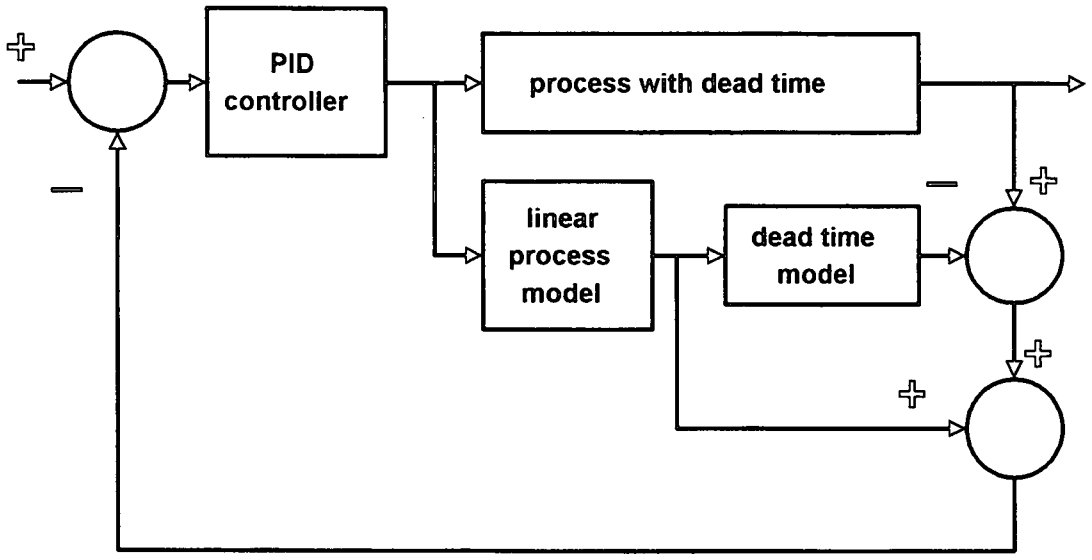

Figure 2-1: Smith Predictor (Smith 1958)

In the current context the following properties of the predictor system have relevance:

(1) With accurate modelling of the process and no load disturbances, the system is capable of applying the same degree of control to a process with dead time as it would

to a process without dead time in a conventional control system.

(2) The effect of a disturbance acting on the process at time t will be measured at the output at any time in the interval $[t, t + L_{proc}]$, depending upon which point within the system the disturbance acts upon. The response of the controller will be retarded by a time period in the corresponding interval $[0, L_{proc}]$.

(3) The only strategy for dealing with random disturbances is to design the system with sufficient inherent stability to limit losses until the controller can correct the error.

Issues raised by O.J.M. Smith in his 1958 publication are summarised below:

(1) Pure dead time (L) has a transfer component of the form e^{-Ls}. This amounts to an "excessive" phase lag with "no attenuation at high frequencies". It can be represented as a "dipole at infinity", or, as a "vertical line of left-side poles and another vertical line of right-side zeros." (Smith 1958, p.112)

(2) The expression e^{-Ls} can be approximated for analytical purposes by means of a suitable power-series expansion, Chebyshev expansion, maximally flat series, or quantised potential functions. (Smith 1958, p.114)

(3) Following on from (2) Smith shows that the above expansions provide a link between dead time and the concept of a distortionless delay line, a lattice network designed to delay a signal for applications in electronics or communications (Smith 1958, p.250).

(4) A dead time block produces infinite phase shift at infinite frequency. (Smith 1958, p.299)

(5) If there are time constants in a system that are small in comparison with other time constants, "they can be added directly to the dead time because their main effect is to contribute to phase shift without attenuation" (Smith 1958, p.307).

(6) There are two common solutions to the instability problems introduced by dead time:

(6a) place an integrating element in cascade with the dead time to provide attenuation that is effective for the first and subsequent resonant poles (Smith 1958, p.303).

(6b) introduce a time constant in cascade that is comparable in magnitude to the dead time. Smith calls this "time-lag stabilisation" (Smith 1958, stability graph on p.306).

(7) Smith considers the effect of a system that is equivalent to an "electrical cable" (i.e. zero inductance, zero conductance). This is in contrast to the analysis of the distortionless transmission line (see above). The Bode diagram on p.314 shows the strong similarity between pure dead time and the distributed lag that takes place in the cascaded RC elements of the cable. (Smith 1958, p.314)

(8) the Smith predictor is depicted in Figure 10-13d on p.326. It shows the familiar components in a format equivalent to Figure 2-2 above, and includes a theoretical load compensating unit that contains the factor $e^{-L_{proc}s}$.

(9) the Smith Predictor method is superior to time-lag stabilisation in that it allows faster response and higher gain. (p.329)

Shinskey (1995, p.99) states "The Smith predictor can be tuned to overcome the sluggish load response by reducing the model gain below the process gain. However there are no known tuning rules for it". It is clear that the use of trained ANNs as a means of (1) identifying and (2) converting process parameters to controller parameters for PID or even more sophisticated control schemes offers opportunities in this area. The next section provides a brief outline of ANNs and related concepts with a view to their use in fulfilling these two functions.

2.2 Artificial Neural Networks (ANNs)

2.2.1 Definition of ANN

An artificial neural network (ANN) is a computational device or construct that uses an interconnection of similar (but not necessarily identical) processing units and which bears some functional resemblance to natural neurological systems, however slight.

ANNs are not intended to be slavish representations of the functioning of the natural

nervous system, but rather draw inspiration from the patterns observed (Fausett 1994). The ANN is primarily a computational device, and is distinct from a scientific model which explores the functioning of biological processes.

The representation of ANNs in software is a natural consequence of the availability of low cost computational power and primary and secondary storage. The identification and tuning methods explored and proposed in this thesis use ANNs implemented in software.

2.2.2 Definitions Relating to ANNs

Neuron

The term "neuron" is used in the context of ANNs to refer to the combinatorial and transfer functions associated with a single processing element. (Kröse & van der Smagt 1993).

Activation Function

The transfer function of the final stage of processing within a neuron is termed the activation function of the neuron. This is normally of the same type within a layer (Kröse & van der Smagt 1993).

Layers

It is common for ANNs to be represented in two dimensions as being made up of successive rows of neurons, the output of each neuron in a row being input to each neuron in the following row. These rows of computing elements are termed "layers". This is the convention followed in this research.

Static ANN

A static ANN is an ANN in which there are no internal or external feedback paths. Data applied to each layer is modified by that layer and transmitted only to the following layer

until the output layer is reached. The output of the system in response to a given input vector is always the same, and is not in any way dependent upon the value of previous input vectors.

Dynamic ANN

A dynamic ANN is an ANN in which data from at least one point in the structure is fed back in such a way as to have the potential of modifying the future value of the data at the output layer of the ANN. The output of the system in response to a given input vector is dependent upon the current input vector and the values of all past input vectors.

2.2.3 A Brief Review of ANN Development

McCulloch & Pitts (1943) proposed a model of the biological neuron and demonstrated that it could be connected into networks that were capable of representing any finite logical function. Eberhart & Dobbins (1990) stated that this work was the first practical exposition of the use of massively parallel architecture, and it provided a basis for the developments that followed.

Hebb (1949) proposed an explanation for the learning process that takes place in biological neurons. An algorithm based on Hebb's learning model was tested in computer simulation and refined by Rochester *et al.* (1956). Hebbian learning is a key concept that was applied successfully in later techniques such as probabilistic ANN models (Hinton *et al.* 1984; Hinton & Sejnowski 1986) and associative memory ANNs (Kosko 1987).

Perceptrons, devices that attempted to mimic biological sensory perception, were proposed and developed by Rosenblatt (1958, 1959, 1960, 1962). He described neural models that used novel training algorithms for both supervised and unsupervised learning. The work included what was probably the first comprehensive simulation of a neural network on a digital computer (Eberhart & Dobbins 1990, p.18). The perceptron learning rule (Rosenblatt 1960) was seen as Rosenblatt's main achievement in this context by

Haykin (1994, p.38).

Widrow & Hoff (1960) produced a learning rule which is known as the least-mean-squares (LMS) algorithm, the delta rule, or the Widrow-Hoff algorithm. This method has been shown to be better than the perceptron learning rule in that it allows the network to respond to input patterns that are similar but not identical to the patterns used in training. Widrow and Hoff initially conceived a computational unit with a signum activation function that they dubbed the ADALINE (Widrow & Hoff 1960). This led to many practical uses in engineering (Widrow *et al.* 1967, Widrow 1987, Tolat & Widrow 1988, Nguyen & Widrow 1989). The concept was extended to multilayer networks with linear elements, MADALINEs, by Widrow & Hoff (1960) and Widrow & Lehr (1990).

Werbos must be credited for publishing the so-called backpropagation algorithm (Werbos 1974), a learning algorithm that can be applied to multilayer ANNs. This work was unfortunately overlooked until the method was discovered independently by Parker (1985) and Le Cun (1986). This method for adjusting the weights of the multilayer ANN became more widely known after it was described and improved upon in Rumelhart *et al.* (1986a, 1986b) and McClelland and Rumelhart (1988).

The backpropagation algorithm, which amounts to descent of an error gradient, suffers from a drawback when used with sigmoidal activation functions. The process can become trapped in a non-optimal local minimum (cf. Kröse & van der Smagt 1993). Various methods have been employed to overcome this difficulty, such as the inclusion of a "momentum" term (cf. Widrow & Lehr 1990) that explores the vicinity of the current solution to look for lower values. The Levenberg-Marquardt algorithm as implemented in Matlab Neural Network Toolbox version 2.0b speeds up the training process by using a combination of gradient descent and Newton's method, but the algorithm is heavy in its use of primary storage during the training phase (Demuth & Beale 1994).

Another practical solution to the local minimum problem can be found in the use of a radial basis function network (RBFN). The existing RBF approach was adapted by Broomhead and Lowe (1988) who proposed a method for the design of feedforward

networks employing radial basis functions. Chen *et al.* (1991) introduced a modification that improved computational efficiency. The RBFN is a two-layer network in which the neurons in the first layer incorporate symmetrical nonlinearities (e.g. gaussian curves) on selected centres. The position and contribution of each curve is adjusted by biases and weights respectively. The neurons in the output layer use linear activation functions and combine the effects of the individual gaussian functions. The algorithm adds neurons to the first layer until the desired function has been mapped to the desired degree of accuracy. The only key parameter that requires selection by the user is the spread constant for the radial basis layer. For this reason the RBFN is often used for prototyping ANN systems (Demuth & Beale 1994). It will usually achieve its objective by using a very much larger number of neurons than the equivalent backpropagation network, which may be problematic if memory capacity is at a premium. A more economical backpropagation network can be developed once viability has been proven.

Associative memory ANNs and self-organising ANNs were studied by Anderson (1968, 1972), Kohonen (1972, 1982), Carpenter & Grossberg (1990).

Hopfield ANNs were introduced by Hopfield (1982, 1984) and developed by Hopfield & Tank (1985, 1986) and Tank & Hopfield (1987).

Overviews of ANN theory are given in Haykin (1994), Anderson *et al.* (1990), and Fausett (1994) and *Proceedings of the IEEE* (1990a, b).

2.2.4 Some Comments on the use of ANNs in Control

Use of Static ANNs

Although there are many ANN schemes for possible use in control problems (cf. Narendra & Parthasarathy 1990; Nijmeyer & van der Shaft 1990; Warwick *et al.* 1992), we will pay more attention to static ANNs and their use in process control (cf. Bajić 1994). Apart from the single neuron ARX model in Chapter 5, all methods proposed in this document make use of static ANNs.

Because static ANNs can approximate arbitrary nonlinearities arbitrarily well (cf.

Fausett 1994; Warwick *et al.* 1992), there is a possibility of using this property to make ANN-based controllers that imitate the conventional nonlinear controllers (cf. Bajić 1994, 1995; Bajić *et al.* 1995). This property can also be applied to provide near-optimal tuning, process identification and noise cancellation (cf. Warwick *et al.* 1992; Fausett 1994). Ljung (1995) points out that the majority of nonlinear black box models used in system identification can be represented by static ANNs. Shinskey (1995, p.100) is somewhat dismissive of neural networks in simple applications that mimic existing methods, but states that they hold great promise in "pattern recognition applications and in high-level optimisation where they can replace models requiring thousands of equations".

The concept of using static ANNs to provide PID settings based on known process parameters was proposed by Ruano *et al.* (1992). This method was developed further in Mamat *et al.* (1995) and has been used, with some modifications, as a component of the system proposed in Chapter 6 of this thesis. The results of extensive time-consuming optimisation experiments or simulations can be stored in the form of ANN weights and biases. The ANN provides a compact self-interpolating representation that holds distinct advantages over tabular representations of empirical loop tuning rules that do not lend themselves to a convenient algebraic representation. This format also lends itself to incorporation in a digitally controlled autotuning scheme.

ANNs Related to Long Dead Time

Our interest is in the use of ANNs for process control purposes, more precisely for control of processes with long dead times. It is widely accepted that such processes are not easy to control (Shinskey 1979; Aström & Hägglund 1988). Shinskey states that ANNs have a promising future in this field (Shinskey 1995).

The problem of converting the time-domain delay into a form suitable for use with a static ANN is addressed in Simmonds *et al.* (1992) quoted by Haykin (1994): "a subsystem of delay lines" is used as part of an ANN-based prototype sonar receiver

based on a biological model of bat sonar. This part is used to determine the echo return delay. Haykin (1994 p35): " . . . there is no reason why a sequential process should not be implemented on a neural architecture . . . " This has relevance in dead time determination. The incorporation of ANNs into the identification and control methods for processes containing long dead times offers the opportunity of exploiting the adaptive and nonlinear characteristics of these structures.

Adaline/Madaline

Widrow and Lehr (1990) make reference to the use of the ADALINE/MADALINE in adaptive control (broom-balancer) in Widrow (1987), adaptive inverse controls in Widrow (1986), nonlinear control and system identification (truck backer-upper) in Nguyen & Widrow (1989).

Adaptive Control

Tao (1994) presents a "critic" and "action" adaptive control scheme which is able to learn how to control completely unknown processes in real time. The "action" ANN issues a control signal which is modified by the "critic" ANN. The "critic" ANN acts as a state estimator. Each ANN is governed by its own learning rule.

2.3 Identification Tests

Identification is the determination of the transfer function or equivalent of the process in question (cf. Schwarzenbach & Gill 1984). This typically involves a practical test in which the response of the process or system to deliberately introduced signals is analysed. This information is then used to decide upon the control strategy required and can be used to determine initial controller settings. This is in contrast to an empirical tuning approach in which the effect of dead time may not receive appropriate attention.

2.3.1 Attributes of Ideal Test

Attention is drawn to three important features of any test used for process identification.

Short Duration

In a practical situation involving testing that requires the plant to be temporarily removed from the production stream it is safe to generalise that the tests should be of short duration. If the process is tested during production using a test signal that is sufficient to affect the product quality, the same reasoning applies.

Small Test Signal

In a practical test situation, it is safe to generalise that a test signal that deviates the output no more than 10% of the steady-state value is preferable to one that drives it to zero or full-scale. Using test signals of this size is not only less disruptive but nonlinearities that may be present in the system are less likely to influence the measurements. As an extension to this technique, very small non-disruptive signals of useful shape or harmonic content can be injected into the system to provide measurements that accurately reflect small signal behaviour. Statistical methods (or their equivalent) are required to correlate the effects of the small test signal from the background noise (cf. Shinskey 1967, Schwarzenbach & Gill 1984) and such methods have been used in different contexts by Hang & Sin (1991) and Bologna *et al.* (1995) amongst others. An equivalent approach is to leave the plant in full or partial production, and to use measurements of actual process signals to infer process parameters (Shinskey 1995).

Closed Loop

Test methods that leave the control loop intact are generally considered to be less disruptive than those that are based on an open-loop test.

2.3.2 The Need for Preliminary Classification of Dead Time

This is a practical issue. Some of the methods presented in this thesis are suitable for measuring systems with restricted, known or zero dead time (Sections 3.2, 4.2 and 5.2 respectively) but the methods which are based on Smith predictor systems (Section 3.3, 4.3 and 5.3) allow for dead times that are almost arbitrarily long. For application of these methods there must therefore be some information available that permits classification of the system within broad parameters so that the method may be selected appropriately. This type of information can generally be obtained from simple step or impulse tests using oscilloscope, chart recorder, stopwatch or computerised methods and is not seen as an obstacle.

2.3.3 Review of Time Domain Methods: Closed Loop

This information is presented as an introduction to Chapter 3 which describes the use of test data that is collected in the time domain and presented to the trained ANNs with virtually no further processing or selection.

Ziegler & Nichols (1942) established the ultimate frequency, ultimate gain method which is still in common use. This has also led to a number of successful automated versions (see Hang *et al.* 1991).

Yuwana & Seborg (1982) reject the widely used approach of Ziegler & Nichols (1942) as being time-consuming, and undesirable because the plant has to be placed in a marginally stable condition. They reject the "process reaction curve" method such as that described by Cohen and Coon (1953) because of the need to open the control loop. They propose instead a quick closed loop test based on a relatively small step change in the setpoint value with the controller set to proportional mode. An underdamped system response is assumed.

The analysis of the response curve in Yuwana & Seborg's method requires determination of five quantities: the first two peak values, the first negative peak value, the duration of the first overshoot, and the steady-state value. The rest follows by calculation

using the given formulas. The method works well in practice, provided the dead time is not dominant. This restriction arises from the Padé approximation used in the analysis. The method is shown in experiment to give good results even if load fluctuations of as much as ±20% from nominal value are taking place during the measurement of the step response.

Lee (1989) modified the method of Yuwana and Seborg (1982) by eliminating the Padé approximation. This was demonstrated to produce good results with dead times as high as double the time constant. A further advantage is that the method allows damped responses to be used for analysis.

Chen (1989) extended the above work to obtain the ultimate gain and ultimate frequency directly without the use of a parametric process model, but process parameter estimates can be derived from these if required. Chen demonstrated the suitability of the method for underdamped systems and with dead time up to twice the time constant.

Ruano *et al.* (1992) successfully applied a neural network technique to the generation of PID controller parameters using as input numerical information extracted from the time domain graphs of process input and output, based essentially on the result of Yuwana & Seborg (1982).

Neural network identification schemes based on preprocessed graphical data have been put forward. Mamat *et al.* (1995) demonstrated that a neural network can form the basis for an autotuning system based on the first-order-plus-dead-time (FOPDT) model. Mamat *et al.* (1995) obtain the closed loop step response of the system with the controller operating in proportional mode and adjusted to produce an underdamped response. The response is analysed to extract time-to-peak and overshoot and undershoot values and these are used together with a trained neural network to yield the FOPDT process parameters.

Bologna *et al.* (1995) demonstrate a method based on minimisation of the difference between actual and model time-domain output in response to a double rectangular set-point perturbation of no more than 5%. This method was shown to work well for first-

and second-order systems, even in the presence of moderate noise.

Ahn (1994) points out that the training of neural networks to model processes is limited in its usefulness because the representation of the process is made in terms of a network structure rather than more familiar terms such as parameters of a transfer function. Ahn notes the structural similarity between the recursive least-squares algorithm used in matching process model parameters to actual measured values and proposes a backpropagation training system which places the ANN to be trained in the test circuit together with the process and the model. The discrepancy between the model and process output forms the error to be used in the backpropagation training process. The parameters corresponding to the chosen model can then be read off from the ANN output.

The time-domain approach taken in the current research (Chapter 3) attempts to obtain useful data by means of a known small perturbation that would certainly be no more disruptive than the method of Yuwana and Seborg (1982) or the Ziegler-Nichols closed-loop method. The use of static ANNs which are, by definition, trained in advance, avoids the need for on-line training. Adaptation of the method of Ahn (1994) allows process parameters to be read off or applied directly within a digital control scheme.

2.3.4 Review of Frequency Domain Methods: Closed Loop

Chapter 4 describes the collection and use of frequency domain data for the identification of processes. This method was developed with due regard to the results reported below.

Hang & Sin (1991) describe an autotuning device based on a pseudo-random binary signal (PRBS). The amplitude of this test signal is kept small enough to restrict output variations due to the test signal to 5% of the full range of variation of the output value. This method applies a modified form of the Ziegler-Nichols tuning formulas (Ziegler and Nichols 1942) by using a Fast Fourier Transform (FFT) and a numerical conversion of the closed loop data to its equivalent open loop form. As in Chen (1989), the method links the frequency domain data to the ultimate gain and ultimate frequency of the system. The method is robust as regards measurement noise and minor load disturbances. During

the test the PID controller parameters may be set to any arbitrary values, provided that the system is stable. However, the PRBS amplitude, bit interval and duration have to be carefully selected to suit the system under test.

Palm (1986 p.517) points out that the Nyquist theorem is particularly useful for systems with dead time. He shows mathematically that long dead time can be considered to have an effect equivalent in some sense to breaking the closed loop. Open loop methods may therefore be applied to the gain curve in the time available, although the dead time does introduce a phase change that must be accounted for.

2.3.5 Review of Use of Neural Process Model

Chapter 5 of this thesis introduces a simulation of the simplest possible configuration of the ARX model by means of a single linear neuron. After training, the weights of this neuron contain the information necessary for computation of the process time constant and gain. The implications of this are examined in Chapter 5 in the light of static ANNs and dead time, but the extension of this investigation to higher order models does not form part of this study. For completeness however, a description of the full ARX model is given below.

In digital simulation or sampled data control, a simple linear difference equation can be used to model the system under test (Pham & Liu 1995). Ljung (1995) refers to this as the ARX model:

$$y(t) = -a_1 y(t-1) - ... - a_{n_a} y(t-n_a) + b_1 u(t-n_k) + ... + b_{n_b} u(t-n_k-n_b+1) \qquad (2)$$

where $y(t)$ is related to a finite number of past inputs $u(t-k)$, and outputs $y(t-k)$. Integers n_a and n_b are the numbers of poles and zeros respectively in the z-domain transfer function of the system. In the context of the current work it is important to note that integer n_k represents the dead time in terms of number of sampling points or integration steps. For zero dead time, $n_k = 1$.

If the delay functions are regarded as external, it is possible to realise this model using a single linear summing element with weighted inputs. The formalisation of this structure as a single linear neuron, identical in all respects to the single neuron used in the output layer of many feedforward ANNs, allows the designer to use this simplifying concept. This allows a uniformity of treatment if the design incorporates other ANN constructs.

The implementation of the delay as a number of sample steps can be done very conveniently within the context of digital control.

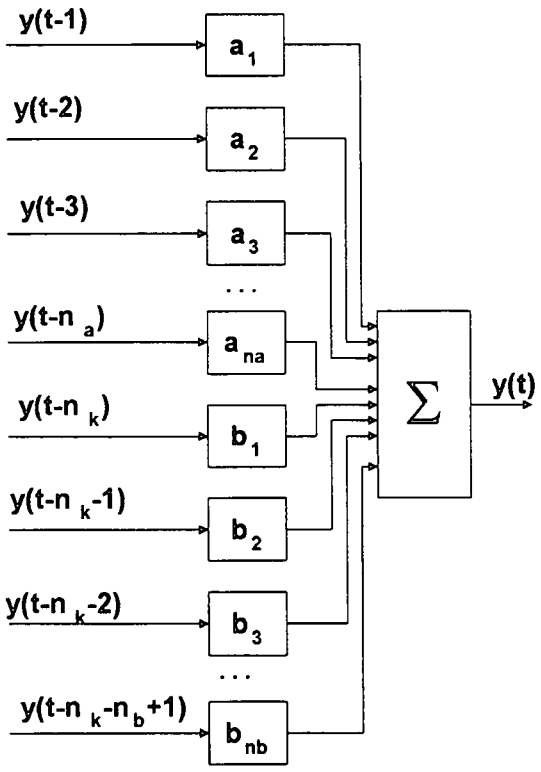

Figure 2-2: ARX model, after Pham & Liu (1995)

With reference to Figure 2-2:

$y(t)$ is the process output sampled at time t.

$y(t-1)$ is the sampled process output immediately preceding the sample at time t.

n_a is the number of poles in the model.

$u(t-1)$ is the process input sampled simultaneously with $y(t-1)$.

n_k is an integer representing dead time in terms of sampling steps. For zero dead time, $n_k = 1$.

n_b is the number of zeros in the model.

Chapter 3

TIME DOMAIN AUTOTUNING USING ANNs

3.1 Introduction

There are well established techniques for system identification that involve the application of calibrated ramp, impulse or step pulses, and the mathematical analysis of the resulting response (see for example Schwarzenbach & Gill 1984). The static ANN methods developed in this chapter makes use of the fact that a closed loop control system with a process that approximates to a first-order-plus-dead-time transfer function $G(s)$ will produce an output response that contains graphical indications of the process gain (K_{proc}), process time constant (T_{proc}) and dead time (L_{proc}):

$$G(s) = \frac{K_{proc}}{T_{proc}s+1}e^{-Ls} \qquad (3.1)$$

This response will of course also be modified to some extent by the controller parameters. The basis of the method, therefore, is to use a calibrated rectangular pulse that deviates the setpoint in both directions briefly, and a "calibrated" PID controller. This calibration of the PID controller is a brief adjustment of the settings to the known stable test values that were originally used as the basis for training the static identification ANNs. In

applying the method, the response is recorded digitally over a fixed period. In order to keep the approach as conceptually simple as possible, this recording is used in its entirety, without selection or modification, as input to trained static ANNs in order to obtain the process parameters (or controller parameters or both) as output. This would allow manual or automatic setting of the PID controller parameters according to any desired tuning method.

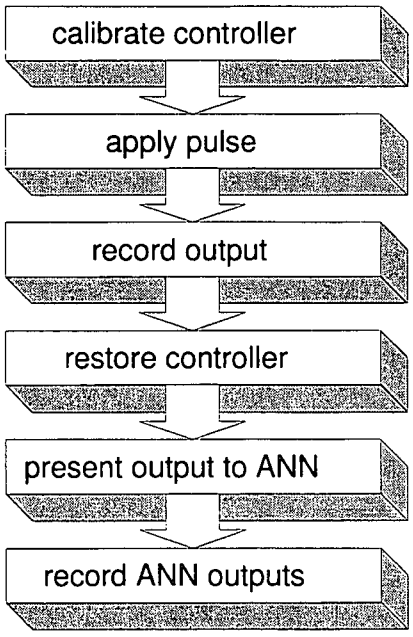

Figure 3-1: Identification test sequence: time domain method

The original scheme called for the direct application to a MIMO ANN of the unmodified samples representing the response waveform at the output of the process in the closed loop. The ANN would then yield a numerical estimate for each of the three process parameters in Equation 3.1.

The amplitude of the output response waveform is related to the process gain, the steepness of the leading and trailing edges is related to the time constant, and dead time introduces a horizontal offset that has no other effect on the waveform. Examples are shown in figures 3-5 to 3-7. This type of pattern lends itself to analysis.

The general principle of applying a step change to the closed-loop system was established by Yuwana and Seborg (1982) who presented a method for inferring the process parameters from such a test. The concept of using extracted features of such waveforms for the training of ANNs to yield process parameters directly was presented in Mamat *et al.* (1995) which applied ANNs to Chen's modification of Yuwana and Seborg's method (Chen 1989; Yuwana and Seborg 1982).

What is different about the approach described in this chapter is that feature extraction is either absent or kept to a simplistic minimum. It is acknowledged that an approach such as Principal Component Analysis of the recording could have been undertaken, but was rejected in favour of exploring the possibilities presented by the raw data. This approach was taken in an attempt to find a simple and universal method without regard to the size of the ANN structure or time taken for training. This view was adopted in the light of falling storage costs and the availability of low cost processing power. An informal review of mass-market computing products has shown that the cost of RAM and disk storage has fallen by two orders of magnitude since 1989, and microprocessor power has increased in the same ratio with no increase in real cost in the same period.

It is asserted that this approach is simpler in concept than the ANN methods described by Valdebenito *et al.* (1995) and Mamat *et al.* (1995) which both involve a degree of preprocessing of the output data.

3.1.1 Outline of the Two Methods Presented in this Chapter

This chapter first describes the simulated prototype resulting from the research referred to above. This was successful in determining FOPDT parameters in the specified ranges. In this method the dead time has an upper limit of $1.5 \times T_{proc}$ for $T_{proc} = 7.8[s]$ and $11.7 \times T_{proc}$ for $T_{proc} = 1.0[s]$. This limitation was imposed by the need to maintain stability in the closed loop control of all combinations of process parameter using the calibrated PID controller settings.

The first method handles dead time, but its restriction to an $L_{proc} : T_{proc}$ ratio of

1.5 under certain conditions disqualifies it as a general method for long dead time. The second method described in this chapter is intended to handle longer dead times than the first, and is therefore presented in the context of a Smith Predictor. This modified method uses ANNs for K_{proc} and T_{proc} only. L_{proc} is estimated by a direct numerical technique. The resulting technique can determine FOPDT process parameters with dead time from $8[s]$ to $80[s]$ and primary time constant from 0.8 to $8[s]$. This permits determination of dead time with upper limit of $10 \times T_{proc}$ for $T_{proc} = 8.0[s]$ and $100 \times T_{proc}$ for $T_{proc} = 0.8[s]$. The technique does however impose an absolute lower limit of $8[s]$ on L_{proc} because the technique depends upon this minimum dead time to allow the recording of response data before it is contaminated by the feedback signal.

The second method is reviewed in terms of its compliance with the original aim and proposed structure. Results of testing under conditions of low-frequency measurement noise are given, and comment about its suitability for use with several examples of higher order process is given.

3.2 Time Domain ANN Method (Restricted Dead Time)

Preliminary research in this direction was published in McLeod & Bajić (1996b) and Bajić & McLeod (1997).

3.2.1 Restrictions on Parameters

This prototype method was developed to determine the three parameters of first-order-plus-dead-time processes that are known to have $K_{proc} \in [0.15, 1.5]$, $T_{proc} \in [1, 7.8]$ and $L_{proc}/T_{proc} \in [0.3, 11.7]$.

3.2.2 Description of Test

The determination is carried out with the control loop intact, but with the controller parameters set to certain predetermined values that are known to provide stable operation for all combinations of process parameters. When the system approximates to a steady state under its temporary PID "test" settings, a known input pulse is applied to the reference input. The response at the process output is recorded digitally for 33.4[s] at intervals of 0.2[s] to yield 168 data points.

3.2.3 Use of MISO ANNs instead of MIMO ANN

The 168 data points that have been recorded could, in principle, then be applied to a single 168-input, 3-output static ANN which had been previously trained to output the three FOPDT parameters corresponding to this time-domain data being applied to the input. The nonlinear approximation capability of static ANNs is ideally suited for use as an encoder of the complex relationships between the hundreds of recorded 168-element samples of output waveform and each of the three process parameters affecting the shape of this waveform. Once trained, the ANN could provide rapid identification based on a single test waveform. In practice however this ideal of a single MIMO ANN was not achieved due to limitations of the available computer resources. In spite of this, the principle was demonstrated successfully using MISO ANNs. As expected, the ANNs conveniently interpolate parameter values for waveforms that fall between pairs of recorded waveforms. The steps taken to achieve an effect equivalent to a MIMO ANN are explained in due course.

3.2.4 Choice of Test Pulse

The standard input pulse described above is shown in Figure 3-2. The step type edges were considered appropriate, and a positive excursion followed by a negative were considered to be less disruptive than an excursion of equivalent amplitude in one direction only.

The longer duration of the second part of the pulse was provided to allow the different responses to develop clearly distinguishable trajectories that would produce a training set with sufficient diversity for good correlation with the process parameters.

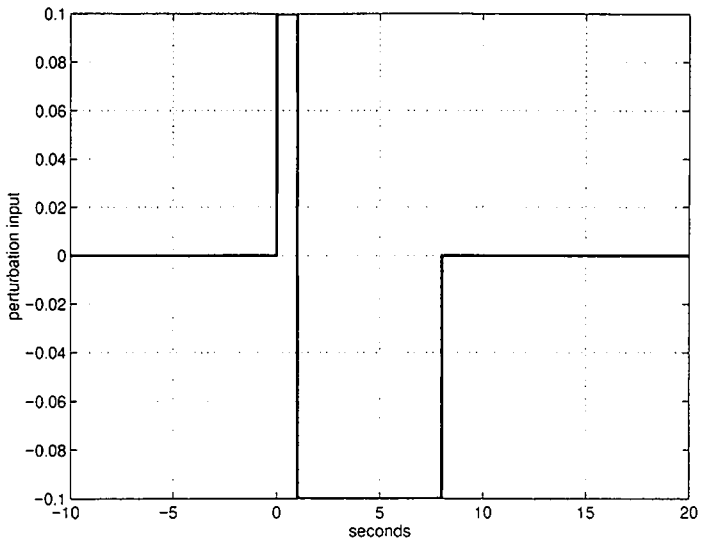

Figure 3-2: Test pulse for time domain testing

3.2.5 Test Circuit

The test circuit incorporating the method for injecting the pulse and the known PID controller is shown in Figure 3-3. It should be noted that the pulse peak amplitude is intended to be a small fraction of the reference value, typically 5%.

3.2.6 Generation of Training Sets

To prepare training sets, a simulation identical to the test setup shown in Figure 3-3 was used in conjunction with a PID controller with known parameters and an FOPDT process model.

The experiments described were based on the following PID controller with set-point weighting:

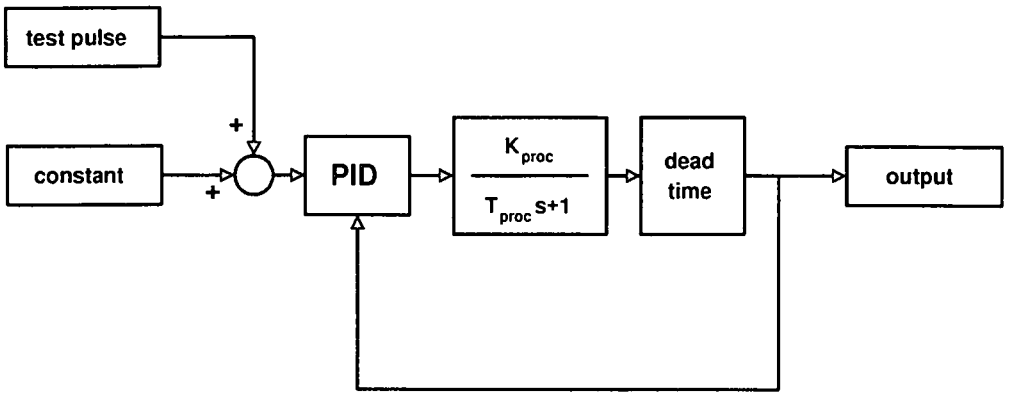

Figure 3-3: Recording of output

$$u = K_p e + \frac{K_p}{T_i s} e - K_p T_d \frac{N}{T_d s + N} y \qquad (3)$$

where $e = r - y$ is the error signal. K_p is proportional gain, T_i is the integral time, T_d the derivative time and N the filtering constant. This scheme is shown in Figure 3-4.

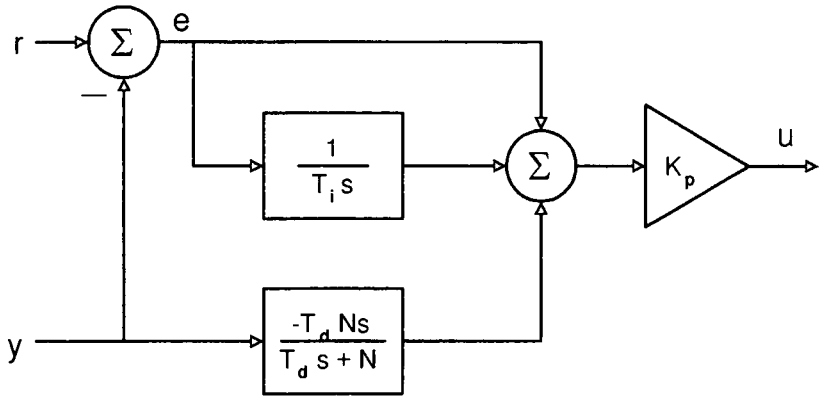

Figure 3-4: PID controller

It was decided to restrict the process parameters to the following ranges:

$$
\begin{aligned}
K_{proc} &\in [0.15, 1.5] \\
T_{proc} &\in [1, 7.8] \\
L_{proc}/T_{proc} &\in [0.3, 1.5]
\end{aligned}
$$

The filtering constant N was set to 10, and values for the other three controller parameters were selected empirically to produce waveforms that were distinctly different as the process parameters were varied. The system was stable for all values chosen. These PID parameter values were:

$$
\begin{aligned}
K_p &= 0.312 \\
T_i &= 10 \\
T_d &= 1.332
\end{aligned}
$$

Each simulation produced a 168-element recording consisting of samples taken every $0.2[s]$ over a period of $33.4[s]$. The training set consisted of two components, the so called "input" training set, which consisted of the 168-element output waveform for each of the n possible combinations of the sets of process variable values selected, and the so-called "output" training set consisting of a $1 \times n$ matrix. This output training set is extracted from the matrix of process parameter values that were used to run the simulation, so that every time the ANN is presented with an input vector, the process parameter that gave rise to that particular waveform is presented as the output value.

A constraint on the proposed method was the number of vectors that could be presented to a neural network for training. This is explained in Section 3.2.7. The initial approach was to use 10 values for each of the 3 process parameters, yielding 1000 combinations. This failed to produce satisfactory results for T_{proc} and L_{proc}/T_{proc} but worked

well for K_{proc}. It was found that 30 values for each of T_{proc} and L_{proc}/T_{proc} produced acceptable results. For this reason the process was divided into two phases:

- A 168×1000 input training set for ANN #1 was generated using 10 values of each of the three parameters, but this ANN would only be for determination of K_{proc}.
- A 168×900 input training set for ANN #2 and ANN #3 for determination of T_{proc} and L_{proc}/T_{proc} respectively used 30 values of T_{proc} and 30 values of L_{proc}/T_{proc}. K_{proc} was set to 1 during this operation.

Examples of the graphs produced are shown in Figs. 3-5, 3-6 and 3-7.

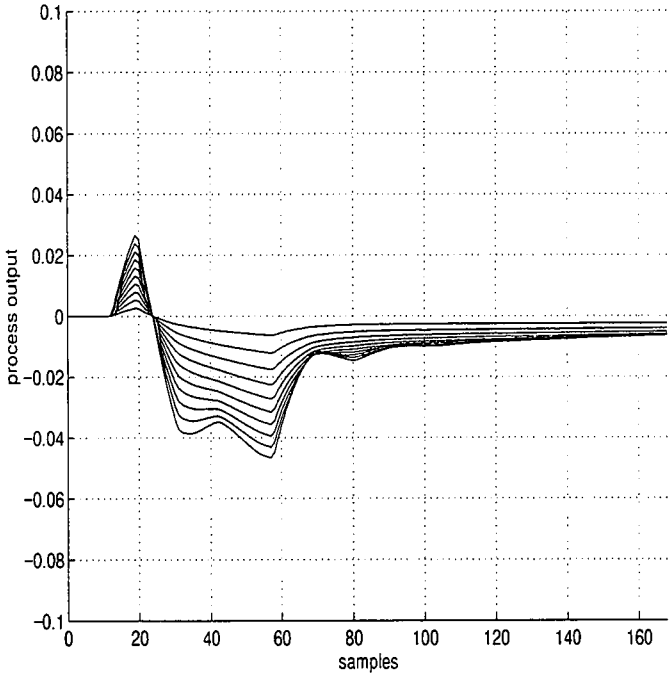

Figure 3-5: Response for different values of K_{proc}

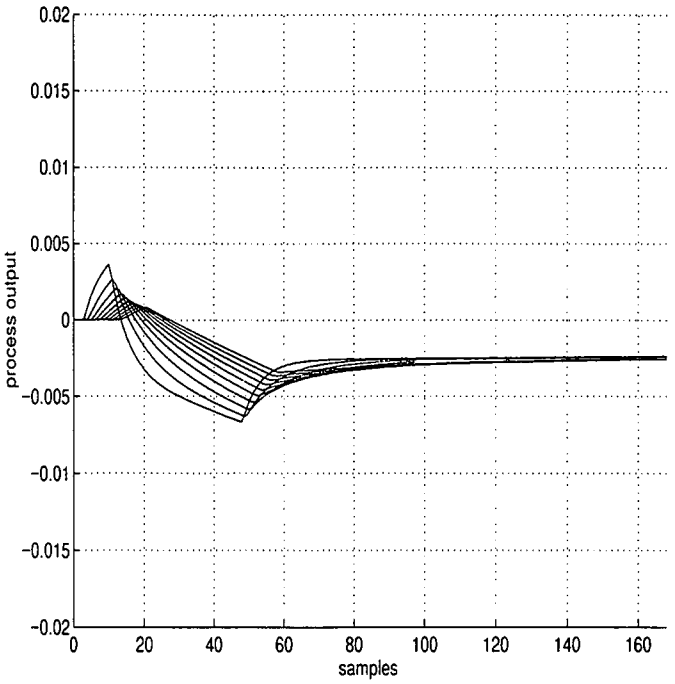

Figure 3-6: Response for different values of T_{proc}

3.2.7 Neural networks - Structure and Training

Choice of ANN Type and Configuration

The choice was narrowed down to radial basis function networks (RBFNs) and the Levenberg-Marquardt Backpropagation Algorithm (BPLM) (see 2.2.3) which are two methods available in the Matlab Neural Network Toolbox 2.0c. BPLM is known to require a great deal of memory and initial experiments showed that BPLM was not viable for this application with the equipment available. Radial basis function ANNs were found to function well and gave consistently good results provided the spread constant was selected correctly. All RBFNs used the standard arrangement of gaussian activation functions in the hidden layer with a linear function in the output layer.

It is, in principle, possible to use one MIMO ANN to produce all three parameters of (1) at once as shown in figure 3-8. This approach was found to be beyond the capability

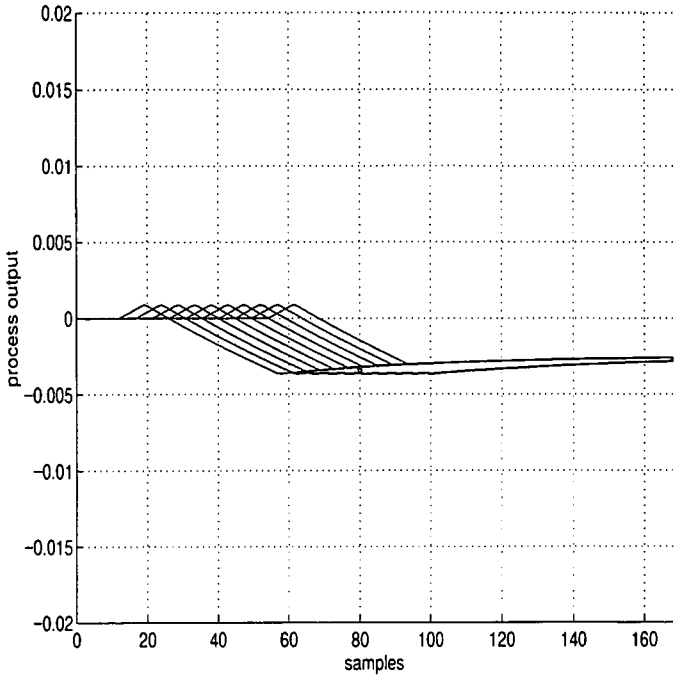

Figure 3-7: Response for different values of L_{proc}

of the PC being used for these experiments when attempting to set up a radial basis function MIMO network. It should be noted that this limitation applies to the training phase only, as the problem is due to the size of the training set rather than the storage requirements for the resulting ANN. This limitation is not seen as a barrier to testing the principles involved.

Training Sets

The test signal of Figure 3-2 was applied 1105[s] after a unit step, when the steady state had virtually been established. The response waveforms were recorded as 168 data points over 33.4[s] from the time of application of the perturbation. The "reference" controller parameters were selected by trial and error so as to produce stable but unique waveforms for all combinations of process parameters.

To generate the input training set for each step value of the parameter which the ANN

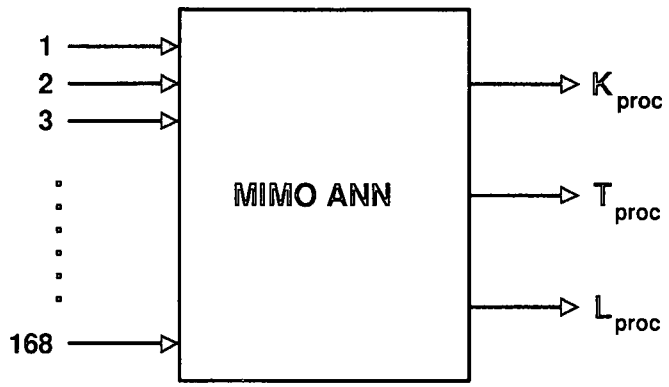

Figure 3-8: Ideal implementation

is being trained to estimate, it is necessary to step systematically through all combinations of all steps of the remaining parameters so as to present examples of output data over the entire range of possible values. Under ideal conditions a single input training set consisting of all combinations of all parameter values could be used. It was initially attempted to use only 10 values of each parameter, and a $10 \times 10 \times 10 = 1000$ vector training set was duly created. Training of ANNs was then attempted for each process parameter in turn, but it was found that the ANN for K_{proc} determination was the only one that converged successfully. The inference drawn from this was that the ANN was able to interpolate the amplitude variations representing different values of K_{proc}, but that the slope and offset variations caused by the steps in T_{proc} and L_{proc}/T_{proc} were too widely spaced to be successfully interpreted by the training process. It was not possible to increase the training set to a size such as $10 \times 30 \times 30$ because the resulting 9000×168 matrix overflowed the $64MB$ primary memory available at that stage. It was therefore necessary to use a separate $30 \times 30 = 900$ vector training set for T_{proc} and L_{proc}/T_{proc}. The problem was taken up in two stages. Firstly, a single radial basis function network (ANN #1) was trained to output the value of K_{proc}. Secondly, ANN #2 and ANN #3 were trained to output T_{proc} and L_{proc}/T_{proc}.

Training of ANN #1

It was found that clearly defined amplitude changes as K_{proc} was varied permitted effective training using as few as ten values of K_{proc} in the range under consideration. Ten values each of T_{proc} and L_{proc}/T_{proc} equispaced over the ranges under consideration were combined with the 10 equispaced values of K_{proc} to produce a 1000-vector output training matrix. The input training matrix was produced by applying these values to the simulation shown in Figure 3-3 using the "reference" PID controller and recording the output waveforms. This produced 1000 168–element vectors. A suitable spread constant was determined by trial and error. The RBFN training took 57 minutes to reach a sum-squared error of 0.0003 with a spread constant of 1. There were 121 hidden layer neurons. The accuracy of the process gain estimates produced by the trained ANN on the basis of recorded system responses was then tested using 100 random sets of values for all three process parameters. This resulted in a maximum absolute error of 1.06% and a standard deviation of the error of 0.234% for the K_{proc} estimates.

Training results for ANN #2. (T_{proc})

The training set was generated by setting process gain to 1.0 and using the 900 combinations of 30 equispaced values each of T_{proc} and L_{proc}/T_{proc} over the ranges under consideration. The RBFN had 39 hidden layer neurons. It trained to a sum-squared error (sse) of 0.03 within 11 minutes using a spread constant of 0.1. Testing with 100 randomly selected values of (T_{proc},L_{proc}/T_{proc}) produced a maximum absolute error of 1.27%, and the standard deviation of the error was 0.204%.

Training results for ANN #3. (L_{proc}/T_{proc})

The waveforms in the training set prepared for ANN #2 were used to train ANN #3 for L_{proc}/T_{proc}. The RBFN had 168 inputs and one output. The spread constant was set to 0.1. Training took 185 minutes to reach $sse = 0.0003$. There were 246 HLNs. Subsequent testing using 100 randomly chosen values of (L_{proc}, L_{proc}/T_{proc}) produced a maximum absolute error of 1.35% and error standard deviation of 0.280%.

3.2.8 Application of the Method for Process Identification

Determination of FOPDT process parameters by means of ANNs requires an on-line test with the control system in closed loop. The experiment follows the steps below which are summarised in Figure 3-9.

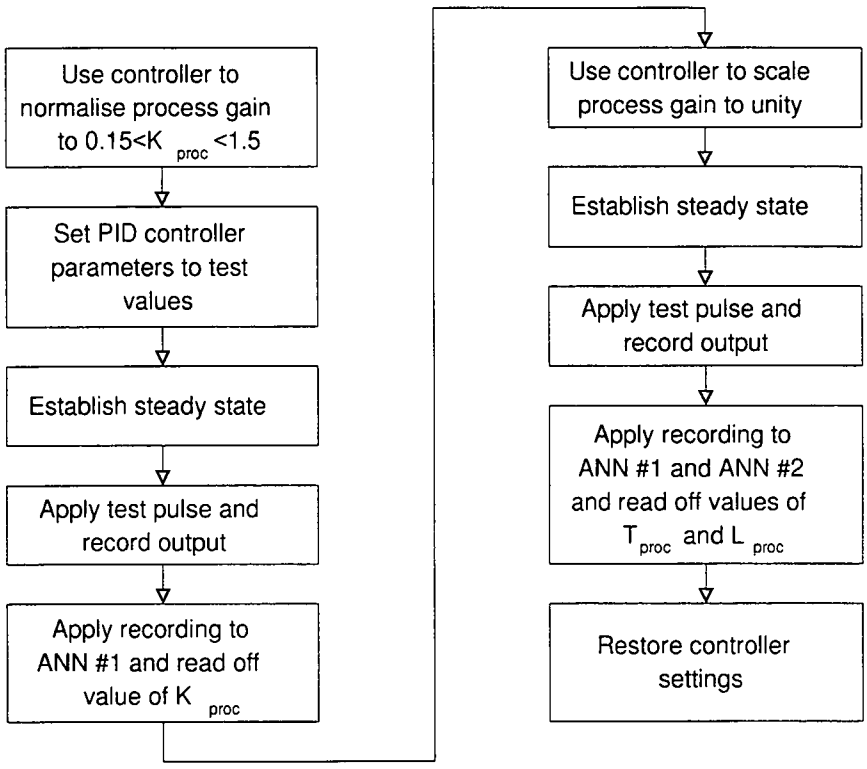

Figure 3-9: Summary of method

Step 1: Determination of K_{proc}

It is assumed that the value of K_{proc} is known to be within the range 0.15 to 1.5 or can be rescaled to this range using, for example, the calibrated gain adjustment on the PID controller. The PID controller parameters are then set as in Table 3.1, and the system is allowed to achieve a steady state.

The test signal shown in Fig. 3-2 is presented to the system input (it is added to the

K_p	T_i	T_d	N
0.312	10	1.332	10

Table 3.1: PID settings for initial test

K_p	T_i	T_d	N
$0.312/K_{proc}$	10	1.332	10

Table 3.2: Setting process gain to unity

signal r). The process response is recorded and presented to ANN #1, which produces the estimated value of K_{proc} directly

Step 2: Determination of T_{proc} and L_{proc}/T_{proc}

As explained, having estimated K_{proc}, the practical limitations of the method make it necessary to conduct a second test for the determination of T_{proc} and L_{proc}/T_{proc} because the test waveforms for training ANN #2 and ANN #3 were generated with $K_{proc} = 1$. At this stage the PID controller is retuned according to Table 3.2 which effectively scales the process gain to unity.

The same test signal as in Step 1 is applied to the system input in the same way. The process response is recorded and presented to ANN #2 which produces an estimation for T_{proc} and to ANN #3 which produces the estimation for L_{proc}/T_{proc}. In this way all three parameters of the process model (1) are estimated.

3.2.9 Evaluation

The accuracy of each stage of the process has been indicated in Section 3.2.7, and the results are consolidated in Table 3.3. These results apply to each individual ANN, and do not reflect the dependence of the T_{proc} and L_{proc} determinations upon the accuracy of the K_{proc} determination.

In order to judge the performance of the system as a whole, ten random sets of FOPDT process parameters were generated in the permissible range and the testing method described in Sections 3.2.8 applied in simulation. The results are given in Table 3.4. It must be emphasised that the results indicated in Table 3.4 represent the propagation of

100 trials	max. abs. err.	standard deviation of error
K_{proc}	1.06%	0.234%
T_{proc}	1.27%	0.204%
L_{proc}	1.35%	0.280%

Table 3.3: Test results for individual ANNs

10 trials	max. abs. error	standard deviation of error
K_{proc}	1.06%	0.334%
T_{proc}	6.20%	1.84%
L_{proc}	4.18%	1.35%

Table 3.4: Results showing propagation of errors

the inaccuracy in the K_{proc} estimate to the subsequent estimates for $Tproc$ and $Lproc$.

In each case, the maximum errors indicated for T_{proc} and L_{proc} were propagated from the K_{proc} value that represented the maximum K_{proc} error over the test.

3.2.10 Conclusion

This method was shown to produce a worst case error of 6.2% for the time constant over 10 random trials with randomly selected FOPDT processes under noise-free conditions for the range of FOPDT process parameters specified. The ideal of a single test was not achieved, and the need for two applications of the test process was seen to have the effect that error in the first measurement causes increased errors during the second measurement. The experiments described do however demonstrate that ANNs can be constructed to handle this type of time domain data without feature extraction. No further tests were done on this system because the dead time is limited to $1.5 \times T_{proc}$ when T_{proc} is at the top of the permitted range.

3.3 Time Domain ANN Method Adapted for Long Dead Time

3.3.1 Description of the Method

The method described in Section 3.2 is not suitable for long dead time because the system becomes unstable when the value of L_{proc} is increased beyond the design limit. The method was modified in order to make it suitable for processes controlled by means of a Smith predictor, with $T_{proc} \in [0.8, 8.0]$, $K_{proc} \in [0.1, 1.1]$ and $L_{proc} \in [8, 80]$.

A key aspect of this method is that a Smith predictor, with the predictor models disabled, can be considered initially to respond to a change of input as it would in open loop for a period equal to the dead time. During this brief period the response to a disturbance pulse is identical to the open loop response of the process.

In applying this method, the first peak resulting from the input spike is detected numerically, and a segment of output data starting at this point and continuing for a period just short of the minimum dead time is extracted. This is equivalent to an open-loop response, unaffected by feedback. With reference to Figure 3-10, measurement of this "open loop" behaviour at the process output commences 1 dead time after the application of the spike. Exactly 2 dead times after the application of the spike, the measurement becomes contaminated by the arrival of the feedback component via the controller. It can also be seen that the recorded response to the disturbance spike is not affected by the PID controller settings.

In testing the given system, the Smith predictor models are disabled and the system is allowed to reach a steady operating state. The PID controller is set to any values that ensure the existence of a steady state. In the tests described here, integral and derivative control elements were disabled, and K_p was set to 0.8.

A sampling period of $0.01[s]$ was selected as the basis for the method. This was found to be necessary to resolve graphical detail associated with the process time constant.

The test pulse, a spike, must deviate the output to a measurable extent without

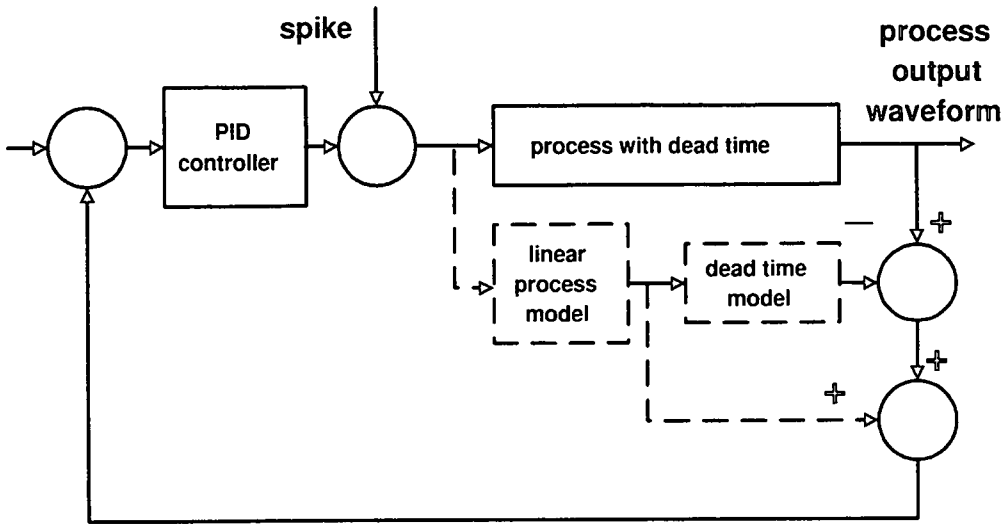

Figure 3-10: Smith predictor control system with model blocks disabled

causing harm to the plant. It is envisaged that this would correspond to an output deviation of no more than 5% of the output steady-state in the worst case. A peak amplitude of 5[*units*] was chosen for development purposes.

In order to capture all meaningful data in the worst case, the recording takes place over a period equal to 160[s], which is equal to twice the longest expected dead time. The output waveform from the spike can therefore vary from a single response spike for high values of L_{proc} to a multiplicity of positive and negative spikes for low values of L_{proc}.

If one were to attempt to use an ANN to produce all three process parameters from this 160[s] waveform without applying feature extraction, it would be necessary to use all 16000 points in order to obtain sufficient resolution of the trailing edge of the waveform for T_{proc} determination. This is considered excessive. For this reason a segment selection technique was decided upon. This technique assumes digital processing facilities to identify and extract a segment of the waveform with a fixed duration selected to be 7.9[s], i.e. just short of the minimum expected dead time of 8[s]. The spike is applied at time zero and the output is recorded over a period of 160[s] at steps of 0.01[s]. The dead time is estimated trivially to the nearest 0.01[s] by using the available computational platform

to identify the highest output value in the recording and determine its time position. A fixed-length 7.9[s] segment of the waveform, starting from this point and decimated to 80 samples, is applied to a single trained ANN to yield approximations for K_{proc} and T_{proc} directly.

3.3.2 Generation of Initial Training Waveforms

Thirty T_{proc} values and 30 K_{proc} values in their specified ranges were used to form a training matrix consisting of the 900 combinations of these values. This constituted the output training set. These values were used in the simulation of Figure 2-4 to generate 900 output waveforms, which were reduced from their original 16000 elements to 80 elements using the method described in 3.3.1. The value of L_{proc} was simply held constant at 40[s] during this process because it has no effect on the shape of the reduced waveform segment.

Samples of the waveforms generated are shown in Figure 3-11. Fig 3-11(a) shows the effect of varying K_{proc} with constant T_{proc}, and Figure 3-11(b) shows the opposite effect.

Equal spacing of the values selected for K_{proc} and T_{proc} was initially chosen, but graphs of the output response corresponding to the different values of T_{proc} did not display sufficient separation for high values and were therefore not considered to be suitable for use as a training set. This problem was solved by using a logarithmic spacing rule to generate 30 values over the desired range so that the spacing was larger for higher values of T_{proc}.

3.3.3 Neural Networks - Structure and Training

The ideal structure of a MIMO ANN with 160 inputs and 2 outputs was realised. A radial basis function ANN was selected for this prototype. The SOLVERB function from the Matlab Neural Network Toolbox 2.0b was used with sum-squared error set to 0.02.

The spread constant of 0.0055 was arrived at by computing the geometric mean of the minimum and maximum euclidean distances between vectors in the input training set.

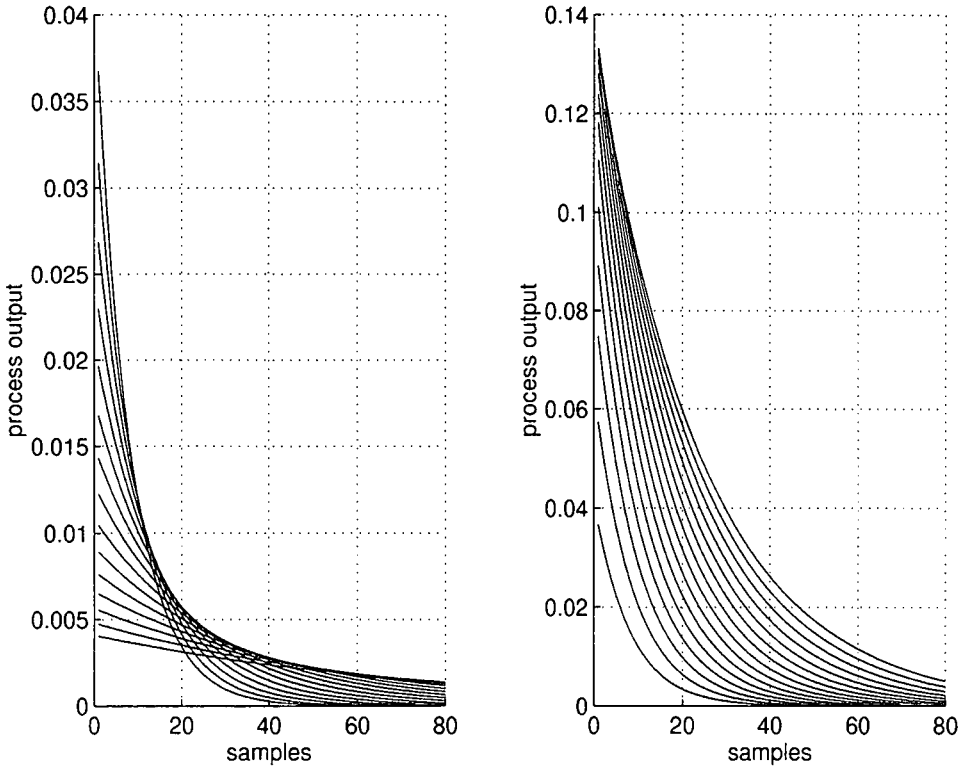

Figure 3-11: Output with (a) K_{proc} constant, T_{proc} varied, and (b) T_{proc} constant, K_{proc} varied.

This value allowed the system to reach the error target in 2.57 hours using 319 neurons.

3.3.4 Testing of the Basic Structure

The results of testing the system with 100 random values of K_{proc}, T_{proc} and L_{proc} are summarised in Table 3.5.

	max. abs. error	standard deviation of the error
K_{proc}	0.633%	0.138%
T_{proc}	0.736%	0.124%
L_{proc}	0.328%	0.0572%

Table 3.5: Test results

3.3.5 Application of the Method

The sequence of steps to be followed in applying this method is given below.

1. Disable Smith predictor models.

2. Set controller to any setting that produces good steady state and can tolerate the small test spike.

3. Wait for steady state to be established.

4. Apply test signal and record output.

5. Restore Smith predictor components and controller settings.

6. Process output waveform to obtain dead time and extract desired segment of output data.

7. Apply the resulting data to the trained ANN to determine approximations for process gain and time constant.

8. Determine controller settings and set controller (see Chapter 6).

This test employs a single measurement using a small pulse. The major disruption to the system is the disabling of the models.

3.3.6 Testing with Measurement Noise

The 100-trial test method introduced in Section 3.2.9 is used as a consistent basis for testing the various methods presented in this thesis. For noise testing, the same method is used, but measurement noise is added to the output of the system using white noise passed through a first-order low-pass RC filter with unity gain at DC and unity time constant. The highest level of white noise that keeps the error of all parameter estimates below 10% provides an indication of the noise tolerance of the method for that parameter. The level of white noise is adjusted in increments of one significant figure only.

The performance of the current method was tested on this basis. The results are shown in Tables 3.6 to 3.9. It should be noted that in this method, L_{proc} is determined by a separate numerical procedure not involving ANNs. The higher noise levels of Tables

100 trials, peak noise 0.0001[*units*]	max. abs. error	*std* of error
K_{proc}	2.79%	0.472%
T_{proc}	7.02%	1.20%
L_{proc}	0.338%	0.0599%

Table 3.6: Standard ANN, noise 0.0001[units] peak

100 trials, peak noise 0.0002[*units*]	max. abs. error	*std* of error
K_{proc}	3.14%	0.480%
T_{proc}	10.10%	1.35%
L_{proc}	0.325%	0.0620%

Table 3.7: Standard ANN, noise 0.0002[units] peak

3.7 and 3.9 are included for comparison with later results. The zero noise results are contained in Table 3.5.

The T_{proc} parameter determination is the most sensitive to noise. To illustrate this, Figure 3-12 shows a plot resulting from tests with process gain, time constant and dead time randomised. K_{proc} and T_{proc} errors are plotted against K_{proc} for a peak noise level of 0.0001[*units*] over 100 trials. The accuracy in each case is lower for low values of K_{proc}. This is attributed to the lower signal-to-noise ratio that arises with low values of output because in this test the noise signal is simply added to the output.

3.3.7 Testing with Higher Order Processes

In order to test the applicability of the method to a more general class of models, several self-saturating processes of order greater than 1 were introduced. The following higher order systems adapted from Hägglund (1992) were used as a basis for further tests:

$$G_1(s) = \frac{0.5e^{-10s}}{(1+s)^2}$$

100 trials, peak noise 0.0003[*units*]	max. abs. error	*std* of error
K_{proc}	5.80%	0.777%
T_{proc}	11.49%	1.82%
L_{proc}	0.268%	0.0593%

Table 3.8: Standard ANN, noise 0.0003[units] peak

100 trials, peak noise 0.0004[*units*]	max. abs. error	*std* of error
K_{proc}	4.42%	0.873%
T_{proc}	17.50%	2.66%
L_{proc}	0.779%	0.0869%

Table 3.9: Standard ANN, noise 0.0004[units] peak

	2nd order (G_1)	3rd order (G_2)	4th order (G_3)
K_{proc} actual value	0.5	0.5	0.5
K_{proc} by ANN (3.3.5)	0.4244	0.4015	0.4114
K_{proc} ANN error	−15.1%	−19.7%	−17.7%

Table 3.10: Kproc estimation for higher order systems

$$G_2(s) = \frac{0.5e^{-10s}}{(1+s)^3}$$
$$G_3(s) = \frac{0.5e^{-10s}}{(1+s)(1+0.5s)(1+0.25s)(1+0.125s)}$$

The test consisted of the following steps of simulation:

(1) Set up the process in a Smith Predictor system with $K_p = 0.8$ and integral and derivative elements disabled.

(2) Use the method of 3.3.5 to obtain FOPDT approximations for the process.

(3) Use a constrained quadratic optimisation technique (CQOT) to obtain a FOPDT approximation for process time constant and dead time for the given process.

(4) Compare the results of (2) and (3) using (3) as the reference.

Results

An optimisation algorithm was employed for step 3 in each case. In this algorithm, the open-loop step response of a given process is compared with the open loop step response of a specified model with dead time and systematically adjusts model parameters until the fit meets the criteria imposed. In this case a FOPDT model was selected with error level of 0.001. The results for each parameter are given in Tables 3.8, 3.9 and 3.10.

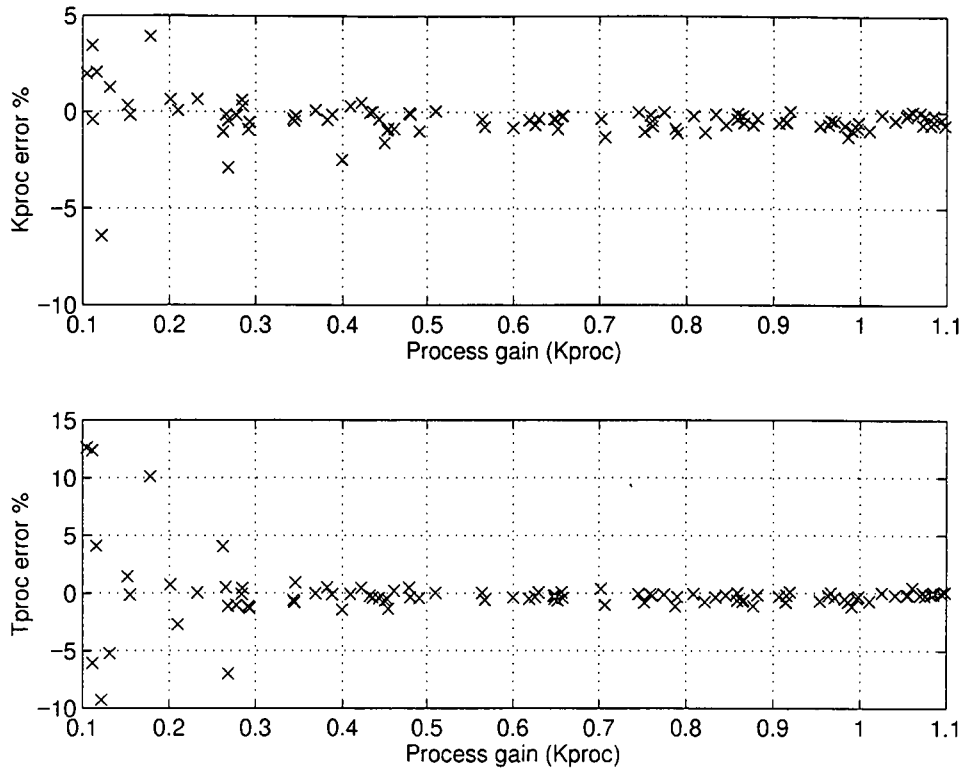

Figure 3-12: (a) K_{proc} and (b) T_{proc} estimation error plotted against K_{proc}.

3.3.8 Use of Randomised Training Set

The method described has very poor immunity to noise as depicted by Figure 3-12(b). The method is also seen to have completely unacceptable results when used with the selected higher order processes. Only three of the nine measurements are within 5%. It was considered that the noise-free training set could have caused this poor performance by allowing the training algorithm to take account of very small differences in the input

	2nd order (G_1)	3rd order (G_2)	4th order (G_3)
T_{proc} by CQOT	1.600[s]	2.003[s]	1.273[s]
T_{proc} by ANN (3.3.5)	1.726[s]	2.041[s]	1.414[s]
T_{proc} ANN error	7.87%	1.90%	11.0%

Table 3.11: Tproc estimation for higher order systems

	2nd order (G_1)	3rd order (G_2)	4th order (G_3)
L_{proc} actual value	10[s]	10[s]	10[s]
L_{proc} by CQOT	10.47[s]	11.13[s]	10.66[s]
L_{proc} by 3.3.5	11.01[s]	12.01[s]	11.16[s]
L_{proc} ANN error	5.16%	7.91%	4.69%

Table 3.12: Lproc estimation for higher order systems

	Without random factor		**factor=0.001**		**factor=0.01**	
	max abs err	*std* of error	max abs err	*std* of error	max abs err	*std* of error
K_{proc}	0.633%	0.138%	0.891%	0.157%	1.31%	0.200%
T_{proc}	0.736%	0.124%	1.152%	0.176%	0.841%	0.202%
L_{proc}	0.328%	0.0572%	0.303%	0.0640%	0.287%	0.0583%

Table 3.13: Error levels of trained ANNs compared

training set that are swamped by the addition of noise or by mild departures from FOPDT characteristics that are displayed by higher order processes. A method suggested by V.B. Bajić was implemented by modifying the Matlab Neural Network Toolbox function SOLVERB which implements the radial basis function algorithm. The following code was added:

```
for counter1 = 1:r
    for counter2 = 1:q
        p(counter1,counter2) = p(counter1,counter2)*(1 + rand*factor);
    end
end
```

The variable `p(a,b)` is the training set. The variable $factor$ is selected at the outset of training. This code has the effect of distorting the graph represented by each vector in a small, random manner without exceeding a ceiling set by $factor$. The training process for this method was repeated using $factor = 0.001$ and $factor = 0.01$. The differences in the training process are summarised in Tables 3.11 and 3.12. It is noted that the introduction of the random factor increased the time taken for training, and resulted in a larger number of neurons in the input layer.

	Without random factor	**factor=0.001**	**factor=0.01**
sse	0.02	0.02	0.02
spread constant	0.0055	0.0055	0.0055
time taken	2.57[h]	2.82[h]	6.29[h]
number of neurons in first layer	319	335	495

Table 3.14: Training outcomes compared

$factor = 0.001$: 100 trials, zero noise	max. abs. error	*std* of error
K_{proc}	0.891%	0.157%
T_{proc}	1.152%	0.176%
L_{proc}	0.303%	0.0640%

Table 3.15: Factor=0.001, zero noise

The two MIMO ANNs which were trained by means of this modified method were tested under circumstances identical to the previous tests. Table 3.11 shows that there is no improvement in accuracy for the modified training method. There is a progressive degeneration in standard deviation for the ANN-estimated parameters. The results are reported below in respect of higher order processes and noise immunity.

3.3.9 Randomised training method tested with noisy measurements

Results for ANN trained using random factor of 0.001:

The results for $factor = 0.001$ are summarised in Tables 3.13 to 3.15. Certain data from earlier results have been repeated for convenience of reference.

Comparison of Tables 3.14 and 3.15 with Tables 3.6 and 3.7 shows that the random factor has produced a better standard deviation and lower maximum error than the

$factor = 0.001$: 100 trials, peak noise 0.0001[$units$]	max. abs. error	*std* of error
K_{proc}	1.49%	0.409%
T_{proc}	5.34%	0.792%
L_{proc}	0.286%	0.0583%

Table 3.16: Factor=0.001, peak noise [0.0001[units]

$factor = 0.001$: 100 trials, peak noise 0.0002[*units*]	max. abs. error	*std* of error
K_{proc}	8.25%	1.06%
T_{proc}	11.55%	1.82%
L_{proc}	0.329%	0.0650%

Table 3.17: Factor=0.001, peak noise 0.0002[units]

$factor = 0.01$: 100 trials, zero noise	max. abs. error	*std* of error
K_{proc}	1.31%	0.200%
T_{proc}	0.841%	0.202%
L_{proc}	0.287%	0.0583%

Table 3.18: Factor=0.01, zero noise

original ANN for K_{proc} and T_{proc} for noise of 0.0001[*units*] peak, this advantage disappears when the noise level is raised to 0.0002[*units*] peak. In comparison with the original ANN, there is no significant improvement in estimation performance in the presence of noise.

Results for ANN trained using random factor of 0.01:

The results for $factor = 0.01$ are summarised in Tables 3.17 to 3.21.

Comparison of Tables 3.17 to 3.21 with Tables 3.6 and 3.7 shows that the random factor of 0.01 has produced a better standard deviation and lower maximum error than the original ANN for K_{proc} and T_{proc} for noise levels up to 0.0003[*units*] peak. This advantage disappears when the noise level is raised to 0.0004[*units*] peak. The introduction of this method allows a peak noise level of 0.0003[*units*] to produce estimates that have error below 10%. This is in comparison with a noise level of 0.0001[*units*] for the original method.

$factor = 0.01$: 100 trials, peak noise 0.0001[*units*]	max. abs. error	*std* of error
K_{proc}	1.63%	0.312%
T_{proc}	2.80%	0.540%
L_{proc}	0.237%	0.0493%

Table 3.19: Factor=0.01, peak noise 0.0001[units]

$factor = 0.01$: 100 trials, peak noise 0.0002[*units*]	max. abs. error	*std* of error
K_{proc}	2.80%	0.467%
T_{proc}	5.53%	1.10%
L_{proc}	0.336%	0.0536%

Table 3.20: Factor=0.01, peak noise 0.002[units]

$factor = 0.01$: 100 trials, peak noise 0.0003[*units*]	max. abs. error	*std* of error
K_{proc}	2.19%	0.542%
T_{proc}	5.77%	1.31%
L_{proc}	0.309%	0.0598%

Table 3.21: Factor=0.01, peak noise 0.0003[units]

3.3.10 Randomised training method tested with higher order processes

Results from Tables 3.8, 3.9 and 3.10 are consolidated in Tables 3.22 to 3.24 together with the results of tests with ANNs trained using random factors of 0.001 and 0.01. L_{proc} is not included as it is not determined by means of an ANN.

From the summary presented in Tables 3.22 to 3.24, it can be seen that the random factor training method increases the error in K_{proc} estimation for the representative higher order systems. It is noted that the ANN trained with $factor = 0.01$ had a beneficial effect on the estimation of T_{proc} in the second and fourth order systems, but in the third order system it had the opposite effect.

$factor = 0.01$: 100 trials, peak noise 0.0004[*units*]	max. abs. error	*std* of error
K_{proc}	11.0%	1.40%
T_{proc}	14.0%	2.26%
L_{proc}	0.338%	0.0662%

Table 3.22: Factor=0.01, peak noise 0.0004[units]

second order	K_{proc} abs. error	T_{proc} abs. error
no random factor	15.1%	7.87%
$factor = 0.001$	26.3%	0.906%
$factor = 0.01$	25.1%	4.21%

Table 3.23: Second order process

third order	K_{proc} abs. error	T_{proc} abs. error
no random factor	19.7%	1.90%
$factor = 0.001$	30.5%	5.15%
$factor = 0.01$	29.9%	9.19%

Table 3.24: Third order process

3.4 Evaluation of the Time Domain Methods Presented

This is presented in terms of the research objectives set out in Chapter 1.

3.4.1 Ability to determine process parameters

For the method of 3.2, the individual ANNs developed for this method give an excellent accuracy under noise free conditions. Because of the two-step approach required, the accuracy of determination of time constant and dead time in the second step depends upon the accuracy of determination of K_{proc} in the first step. Nevertheless, when this two-stage method is tested using random values, it is found that the degradation of accuracy of T_{proc} and L_{proc} determination is tolerable.

For the method of 3.2, when T_{proc} is at the high end of its range, the maximum ratio of dead time to time constant L_{proc}/T_{proc} is 1.5. For the method of 3.3, when the time

fourth order	K_{proc} abs. error	T_{proc} abs. error
no random factor	17.7%	11.0%
$factor = 0.001$	29.3%	0.370%
$factor = 0.01$	28.0%	2.75%

Table 3.25: Fourth order process

constant T_{proc} is at the high end of the range, then the maximum ratio of dead time to time constant L_{proc}/T_{proc} is 10. The method of 3.3 is however restricted in that it is unable to deal with dead times less than 8.0[s]. This restriction cannot be overcome because the principle of operation relies on this brief period. This detracts from its universal applicability.

The method of 3.3 produces excellent accuracy under noise free conditions. Although there is only one measurement, the processing of the recorded waveform takes place in two steps. The determination of K_{proc} and T_{proc} is dependent upon the accuracy of the determination of L_{proc}. In spite of this, simulations show that this effect produces minimal degradation. Under noise-free conditions over 100 trials, a maximum absolute error of 0.633% was noted for K_{proc} and 0.736% for T_{proc}. L_{proc} was estimated with a maximum absolute error of 0.328%.

3.4.2 Use of one test with minimal disturbance to operation

The method of 3.2 uses two short tests. The requirement of a single, short test has been met by the method described in 3.3. Adjustments to the PID controller for method 3.2 are crucial to the accuracy of the measurement. In method 3.3 the adjustments are only to ensure that the process is in a stable condition before application of the test spike.

3.4.3 Application of test in closed loop mode

The main feedback loop is retained in both methods described in this chapter, although the method of 3.3 does require disabling the Smith predictor models.

3.4.4 Use of ANNs wherever possible

The method described in Section 3.2 used ANNs for approximation of all three parameters.

In Section 3.3 ANNs were used for determination of T_{proc} and K_{proc} but not L_{proc}.

This is because the value of L_{proc} is available by inspection to a resolution limited only by the sampling rate, and there is no interpolation involved.

The principle of using ANNs for conversion of process parameters to PID controller settings is developed in Chapter 6.

3.4.5 Use of information from time domain signals

Although the shortcomings of the method are obvious in some respects, an attempt was made to explore the time-domain approach as far as possible. The study has been useful in that it has demonstrated the problems caused by noise, and has highlighted the "open loop" measurement opportunities inherent in a closed-loop system with long dead time.

3.4.6 Applicability to higher order processes

This was only tested for the method of 3.3. To be useful, the method should provide results that deviate from the chosen standard by no more than 10%. A 5% deviation would be entirely acceptable.

The approximation of process parameters for the higher order examples was not satisfactory. The 5% criterion was only met in 2 out of the 9 measurements. The reason for this failure is ascribed to the inability of the trained ANNs to respond to the general similarities that the higher order responses have with the FOPDT responses.

An attempt to train the ANN in such a way as to make it less sensitive to slight differences in the general shape of the test waveforms when compared to the training waveforms was made. The introduction of a small distorting factor during training was investigated using two levels of this factor. These changed the waveform randomly by a factor not exceeding 0.1% in the first instance, and 1.0% in the second. Although some improvement was noted in some particular instances, there was no consistent improvement.

3.4.7 Ability to work with noisy data

The effect of noise was only investigated in respect of the method of Section 3.3. The maximum absolute error of the ANN developed in Section 3.3.3 was degraded beyond 10% when measurement noise of 0.0002[*units*] peak (before filtering) was added to the output waveforms. The effect of noise is more pronounced when the process gain is low.

The strategy of introducing a random factor into the training set was found to have some effect on the ability of the ANN to interpret noisy waveforms. Specifically, a random variation of no more than 1.0% produced an ANN that required a noise level of 0.0004[*units*] peak (before filtering) to drive the error beyond 10%.

3.5 Conclusions

The time domain approach described in this chapter tested the viability of using unprocessed time domain response data from a closed-loop test as input to an ANN for the purpose of process identification. It was demonstrated that static ANNs are indeed able to extract this data to an acceptable level of accuracy under ideal simulated conditions.

An attempt was made to eliminate the effects of noise, and to allow for the variations on the first-order response pattern caused by the presence of higher-order components in the process under test. The effect on higher order processes was not clearly beneficial, but an ANN was produced that was approximately twice as robust in the presence of noise than an ANN trained in the conventional manner. However, in overall review of the method, the time domain method is sensitive to a level of noise that can hardly be seen on a graphical display. This approach was therefore not developed further in this study.

Chapter 4

FREQUENCY DOMAIN AUTOTUNING USING ANNs

4.1 Introduction

The methods described in Chapter 3 of this document used time domain data as input for determination of process parameter values. This approach was found to be sensitive to noise to an extent that would make it impractical in many situations. This led to an investigation of the use of frequency domain data as the basis for identification. It was attempted to perform identification in the closed loop using static ANNs to recognise patterns in the unmodified frequency domain output data resulting from a known test signal in a control configuration with known parameters. Initial work in this area was published as McLeod & Bajić (1997a). The basic approach is summarised in Figure 4-1.

The method developed and proposed in this chapter uses the relationship between process parameters and frequency response as the basis for the numerical determination of those parameter values. Translation of data to the frequency domain before neural network processing was used by Gorman and Sejnowski (1988) in sonar image processing. This approach allowed a significant reduction in the number of input units required when compared to a time domain method, although in the system proposed here this reduction

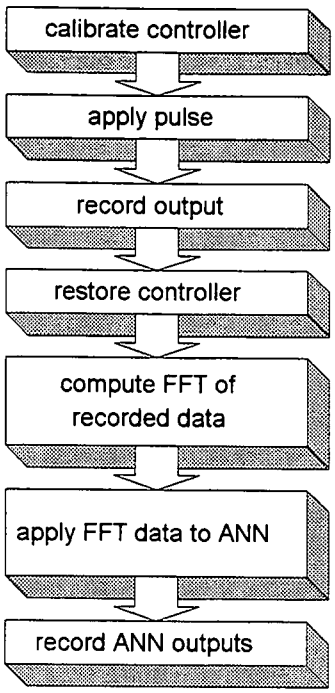

Figure 4-1: Identification test sequence

was not systematically explored.

Approaching the problem in two steps, the issue of dead time is initially omitted. In Section 4.2, the principle of using a frequency domain training set providing process gain and time constant directly is tested. A single neural network provides a compact representation of this mapping. This is extended in Section 4.3 to dead time determination for long dead times

4.2 Frequency Domain ANN Method (Dead Time Excluded)

4.2.1 Overview

The experiment described here was a first approach to frequency domain identification to clarify general principles. It is assumed that dead time is zero, and attention is concentrated upon the process gain and time constant. The PID controller is adjusted to provide proportional gain with integral and derivative elements disabled. In contrast with the methods of Chapter 3, only a single test and a single MIMO ANN are required. The intention is to develop an ANN system for direct reading of the two parameters of a first order system: static gain and time constant. The process parameter values must be available immediately following a brief one-shot closed-loop test that does not disturb the process unduly.

The method uses a closed-loop system under the control of a P-controller with a gain of 0.25 (see Figure 4-2). This gain setting was chosen to ensure that the system under measurement would be stable under all possible values of gain and time constant. The parameter ranges for which training sets were developed are $K_{proc} \in [0.2, 2.0]$ and $T_{proc} \in [0.5, 5]$. The test signal shown in Figure 4-3 is intended to be scaled so that its peak amplitude is one tenth of the constant setpoint value. The system is assumed to be in a steady state when the test pulse is applied.

4.2.2 Generation of the Training Set

Target data:

The target vector set consisted of the 900 combinations of 30 logarithmically spaced values of T_{proc} in the range $[0.5, 5.0]$ and 30 linearly spaced values of K_{proc} in the range $[0.2, 2.0]$. The logarithmic spacing was used so that the time domain waveforms for small values of T_{proc} would be clearly distinguishable from one another. This precaution was

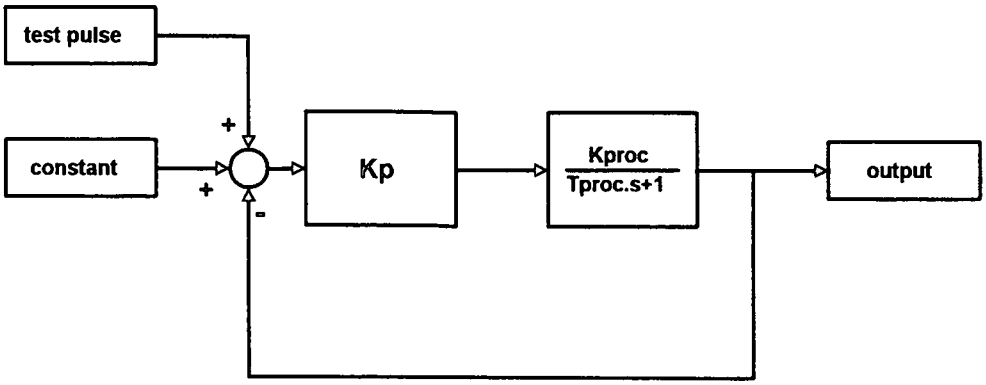

Figure 4-2: Schematic for frequency domain testing

not needed for K_{proc}.

Input data:

A simulation of the system shown in Figure 4-2 was set up. The step size was $0.01[s]$ and the duration $100[s]$. The test signal was timed to start at $45[s]$. This delay allows the system to settle after the initial unit step in the reference input signal.

The simulation was run for each of the 900 input combinations. In each case the output waveform was recorded over a time interval of $100[s]$. Because the output at $40[s]$ was not fully equal to the final steady-state value, an exponential correction was computed and applied to the last $60[s]$ of each recording to make the amplitude exactly zero at the start and finish of the window period. A Fast Fourier Transform was computed on the resulting data. The magnitudes of the first 15 coefficients of the transform in each simulation were then stored for use as the input training matrix. The 15th coefficient corresponds to a frequency of $2.1[radians/s]$. This selection is equivalent to removing higher frequency components from the recorded signal. This restriction was arrived at by means of a graphical computer animation of the data which showed that no significant spectral changes were taking place at higher frequencies.

Selected parts of the training set are shown in Figs. 4-4 and 4-5. Fig. 4-4 shows curves in the time and frequency domain for the two extreme values of T_{proc} with each

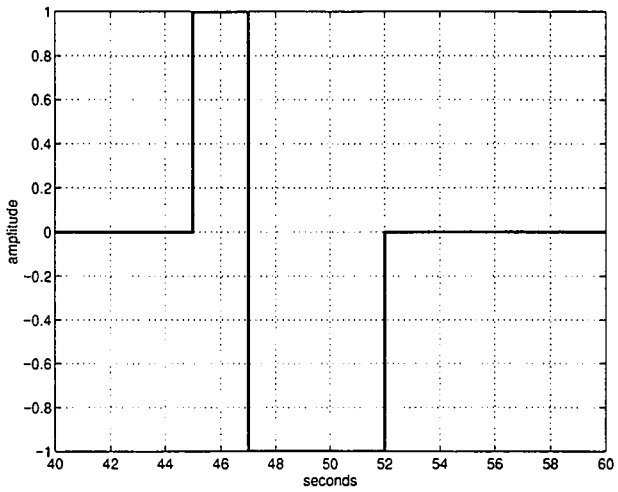

Figure 4-3: Test signal for frequency domain method

	K_{proc}	T_{proc}
maximum absolute error	0.7336%	1.396%
standard deviation of the error	0.0955%	0.1733%

Table 4.1: Training results

of 10 representative values of K_{proc}. Fig. 4-5 shows curves for all values of T_{proc} and one fixed K_{proc} value.

4.2.3 Training

It was decided to build a radial basis function network using the SOLVERB function of the Matlab Neural Network Toolbox version 2.0b (The Math Works Inc.). The sum-squared error goal was set to 0.01 and the spread constant to 1.52. The training produced a network with 238 neurons in the hidden layer. The validity of the training was tested by running 1000 simulations using random values for K_{proc} and T_{proc}. The results are summarised in Table 4.1.

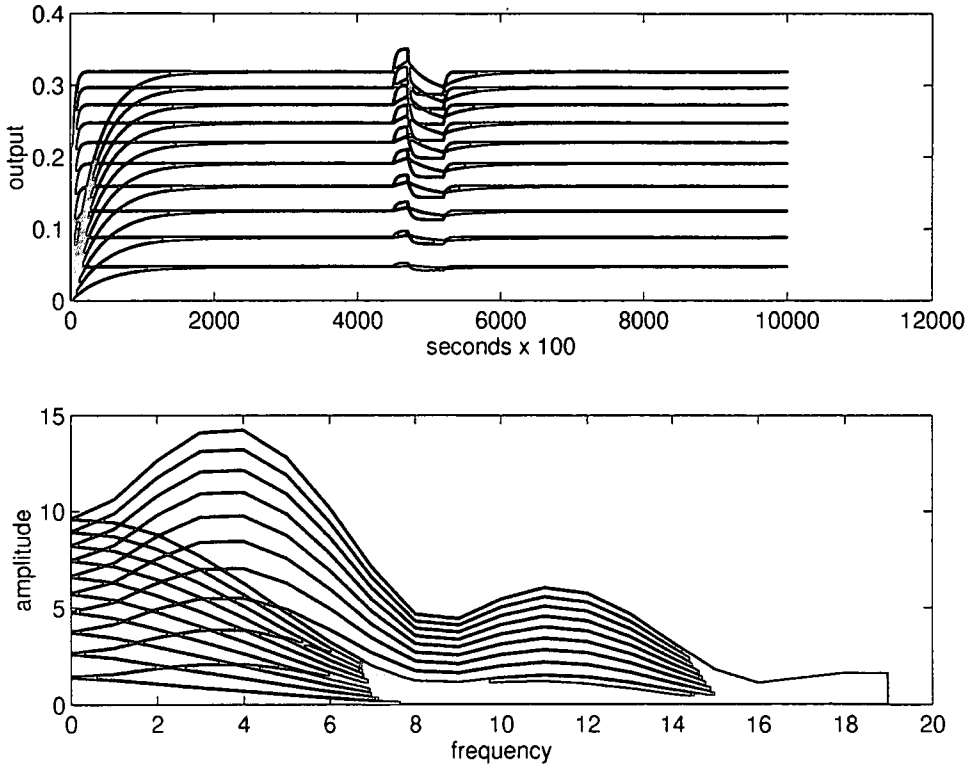

Figure 4-4: Curves for two extreme values of T_{proc} plotted for 10 values of K_{proc}

4.2.4 Application of the Method for Process Identification

This is summarised in Figure 4-6.

The system to be tested has the controller gain K_p set to 0.25 (for reasons explained earlier). A test pulse of the form shown in Figure 4-2 is added to the reference input and the process output signal is recorded. A Fast Fourier Transform is computed on a window of the recording that contains the rest state before the pulse, the pulse itself, and the rest state after the pulse. The recording is processed so that the rest state has an amplitude of zero. The magnitude values of the first 15 harmonics of the transform are applied to a trained NN. This yields close numerical approximations for the values of K_{proc} and T_{proc} directly as in the table above.

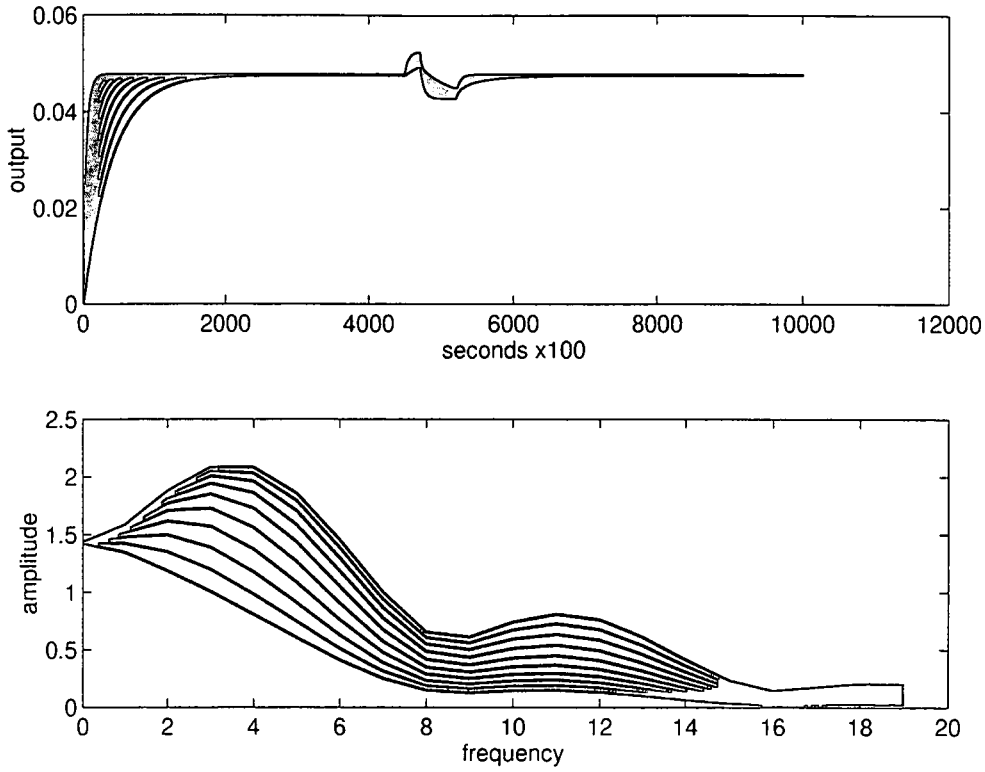

Figure 4-5: Curves for 10 values of T_{proc} with K_{proc} held constant

4.2.5 Evaluation

The method provides uses a single test. The values of gain and time constant are estimated to an accuracy well within 5%. Temporary adjustment of the PID controller is required. The method was not evaluated for noisy data or higher order systems because it was considered to be an evolutionary step towards a fully developed method that could handle processes with long dead times.

4.2.6 Conclusion

The general principles of an ANN-based system of analysis using a contiguous set of FFT magnitude values has been demonstrated. Angular FFT values were not required for determination of gain and time constant.

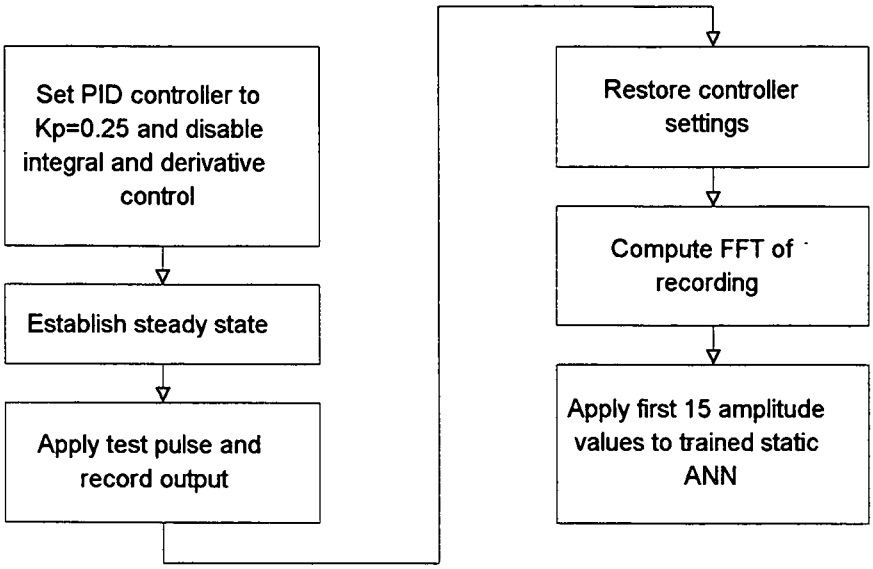

Figure 4-6: Summary of method

4.3 Frequency Domain ANN Method Adapted for Long Dead Time

4.3.1 Overview

In order to make meaningful comparisons between this method and the method of Section 3.3, a testing arrangement functionally equivalent to Figure 3-10 was set up. The same PID controller settings were used ($K_p = 0.25$, $K_i = 0$, $K_d = 0$), but the spike was applied *before* the controller to permit the controller gain to be used for normalisation purposes. The test spike was given an amplitude of $20[units]$ to compensate for the controller gain so that its effect would be equivalent to the effect of the $5[units]$ spike used in Section 3.3. This was done to ensure that comparison of performance in the presence of measurement noise would be meaningful. The schematic is shown in Figure 4-7.

Unlike 4.2, both amplitude and phase information from the FFT is used in the training process.

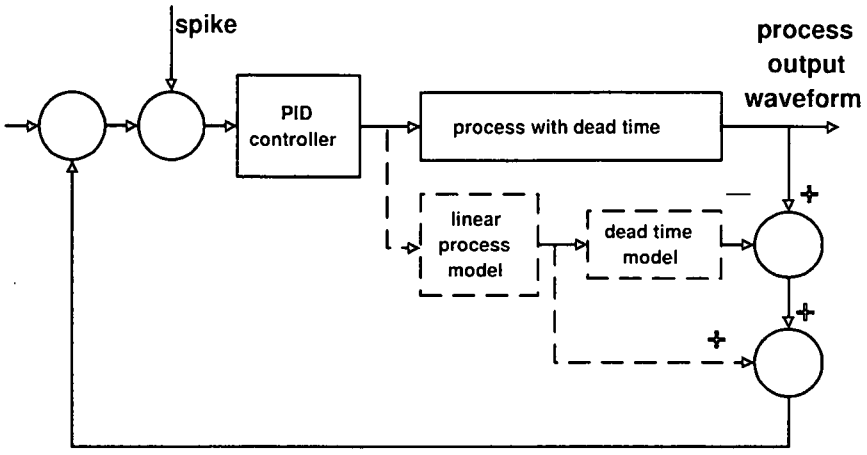

Figure 4-7: Circuit arrangement for frequency domain test

The output is recorded, but in keeping with a purely frequency-domain approach, no attempt is made to detect the position of the first peak. The output waveform is recorded for a period of 163.83[s] using a sampling time of 0.01[s]. This period is too short to incorporate all the transients caused by the impulse, and it is necessary to apply a shaping function to attenuate the recorded waveform towards the trailing edge of the recording to prevent discontinuities from appearing in the time domain data. It was found that a simple straight-line attenuation with zero attenuation of the first sample and maximum attenuation of the final sample produced acceptable results.

A Fast Fourier Transform of these 16384 points is then performed. Animation of the phase and amplitude plots with process parameters systematically varied showed that the first 81 components of the transform contain almost all the variation produced. The modulus and argument of each component are extracted to form a 162-element vector.

For reasons to be explained in the next section, the testing is carried out as a two step process: K_{proc} is estimated by applying the 162-element vector to ANN #1. The process gain is then normalised to 2.0 and the test repeated. The output is shaped, the transform recalculated, and the modulus and argument values applied to ANN #2 and ANN #3 which yield T_{proc} and L_{proc} directly.

4.3.2 Generation of the Training Set

The training set for the K_{proc} ANN used 2500 input vectors produced in simulation using the circuit of Figure 3-10, with 25 equally spaced values of K_{proc} in the range $[0.2, 2]$ and 10 each of $L_{proc} \in [8, 80]$ and $T_{proc} \in [0.8, 8]$, also equally spaced. For each of the remaining ANNs, the input set consisted of 2500 input vectors produced in simulation with $K_{proc} = 2$, and 50 values of each of L_{proc} and T_{proc} from the ranges specified earlier.

It would have been preferable to have used a single ANN trained using 25 values of K_{proc}, 50 values of T_{proc} and 50 values of L_{proc}. This however would have led to a training set consisting of $25 \times 50 \times 50 = 62500$ vectors of 162 elements each. The handling of such a large training set for the radial basis function method chosen was beyond the capacity of the available computer which was equipped with $256MB$ of primary memory.

The major implication of the two-step approach is that the accuracy of K_{proc} estimation affects the accuracy of the subsequent estimations of T_{proc} and L_{proc}. It is therefore desirable to estimate K_{proc} as accurately as possible.

4.3.3 Training

Details of the three RBFN ANNs trained for this experiment are given in Table 4.2. The test results in the table are from a 100-trial test in each repetition of which the process parameters are given random values from the permissible ranges. This is mainly to confirm that the performance of the ANN is satisfactory for input values that do not correspond to the training set. For testing ANN #1, all parameters are randomised. For testing ANN #2 and ANN #3, K_{proc} is held at 2, and only T_{proc} and L_{proc} are randomised.

A set of graphs resulting from settings of $K_{proc} = 0.2$, $T_{proc} = 8.0[s]$ and $L_{proc} = 8.0[s]$ is shown in Figure 4-8. The upper graph of Figure 4-8 shows the time domain plot recorded at the process output before the application of the shaping function. The centre graph gives the amplitude of the 81 non-redundant FFT components, and the lower graph gives their phase. Point 81 of the FFT plot represents a frequency of $0.38[rad/s]$.

	ANN #1 (K_{proc})	ANN #2 (T_{proc})	ANN #3 (L_{proc})
Size of input training set	162×2500	162×2500	162×2500
sum-squared error goal	0.001	0.05	0.02
Number of hidden layer neurons	203	152	370
Spread constant	22	20	10
Maximum absolute error	4.763%	1.36%	1.17%
Standard deviation of error	0.908%	0.313%	0.173%

Table 4.2: Results of training

4.3.4 Application of the Method

The method is implemented by applying the steps described below:

1. Disable Smith predictor models.
2. Set controller to proportional mode only, with $K_p = 0.25$.
3. Wait for steady state to be established.
4. Apply test signal and record output.
5. Apply taper and FFT to recorded output.
6. Apply the resulting data to ANN #1 to determine approximation for process gain, K'_{proc}.
7. Adjust K_p, the proportional gain of the controller, to $K'_p = K_p \times 2/K'_{proc}$.
8. Apply test signal and record output.
9. Apply taper and FFT to recorded output.
10. Apply the resulting data to ANN #2 to determine approximation for process time constant T_{proc}.
11. Apply the resulting data to ANN #3 to determine approximation for process time constant L_{proc}.
12. Determine controller settings and set controller (see Chapter 6).

4.3.5 Testing with Measurement Noise

Using the consistent noise testing basis established in 3.3.5, filtered white noise is added to the process output as before. Each of the three ANNs was tested to determine the

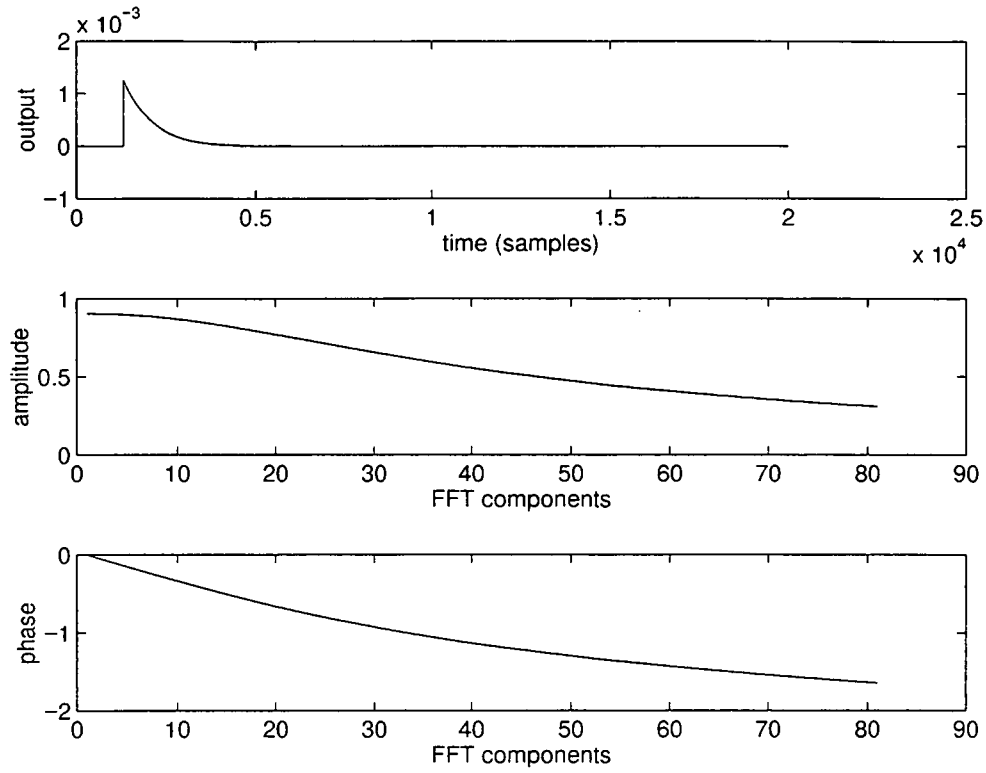

Figure 4-8: Output and FFT data with zero measurement noise

extent of the error introduced by noise. In contrast with Section 3.3.5, the estimate for L_{proc} is now obtained by means of an ANN. The results are shown in Tables 4.4 and 4.5. Table 4.3 contains the zero noise results from Table 3.2 for convenient comparison.

The tables show that the maximum acceptable noise level is $0.001[units]$, a limitation that arises both in the estimation of the process gain and the process time constant. A graph of the output waveform with the white noise generator set at this value, $0.002[units]$ with the process parameters set to $K_{proc} = 0.2$, $T_{proc} = 8.0[s]$ and $L_{proc} = 8.0[s]$ is

100 trials, zero noise	max. abs. error	*std* of error
K_{proc}	4.763%	0.908%
T_{proc}	1.36%	0.313%
L_{proc}	1.17%	0.173%

Table 4.3: Error with zero noise

100 trials, peak noise 0.001[*units*]	max. abs. error	*std* of error
K_{proc}	5.35%	1.59%
T_{proc}	9.50%	1.69%
L_{proc}	0.500%	0.158%

Table 4.4: Error with noise of 0.001[units]

100 trials, peak noise 0.002[*units*]	max. abs. error	*std* of error
K_{proc}	11.32%	2.48%
T_{proc}	10.88%	2.12%
L_{proc}	0.841%	0.319%

Table 4.5: Error with noise of 0.002[units]

shown in Figure 4-9(a). The effect on the portion of the FFT data used for training is clearly visible. The magnitude and phase components are shown in Figure 4-9(b) and (c) respectively. Compare with Figure 4-8 which shows the results of the same experiment without noise.

4.3.6 Testing with Higher Order Processes

The following higher order systems introduced in 3.3.6 were used as a basis for further tests:

$$G_1(s) = \frac{0.5e^{-10s}}{(1+s)^2}$$

$$G_2(s) = \frac{0.5e^{-10s}}{(1+s)^3}$$

$$G_3(s) = \frac{0.5e^{-10s}}{(1+s)(1+0.5s)(1+0.25s)(1+0.125s)}$$

The test consisted of the following steps of simulation:

(1) Set up the process in a Smith Predictor system with a P controller gain set to $K_p = 0.25$.

(2) Use the method of 4.3.1 to obtain FOPDT approximations for the process.

(3) Use the constrained quadratic optimisation technique (CQOT) described in Section 3.3.6 (step 3) to obtain FOPDT approximations for process time constant and dead time for the given process.

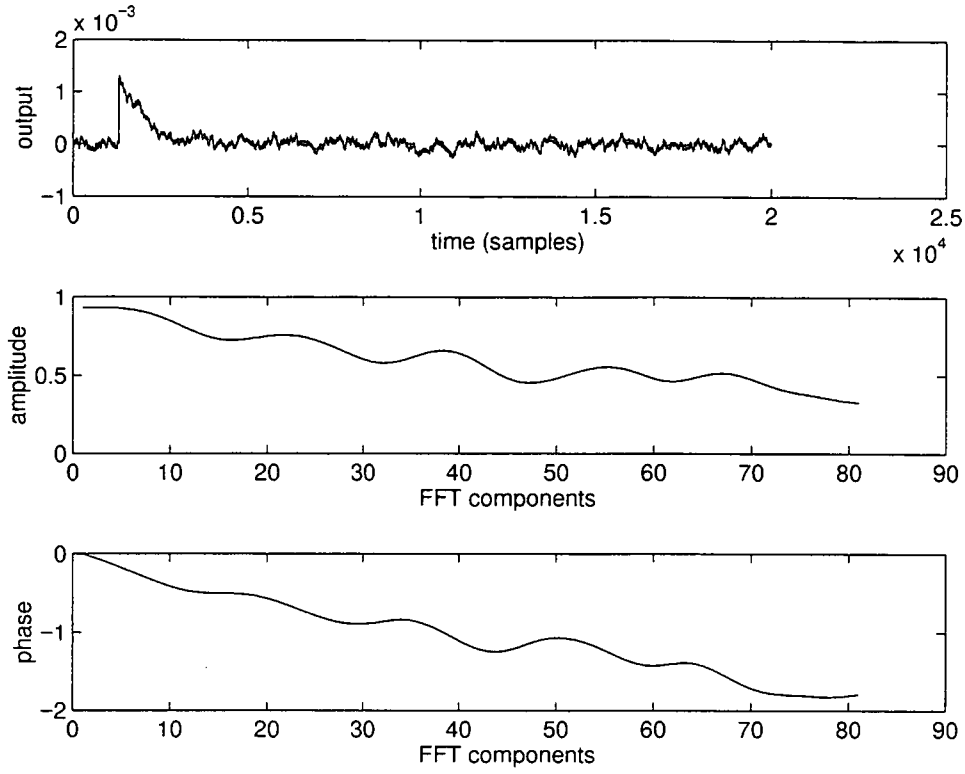

Figure 4-9: Effect of measurement noise of $0.001[units]$ peak on training set

	2nd order G_1	3rd order G_2	4th order G_3
K_{proc} actual value	0.5	0.5	0.5
K_{proc} by 4.3.4	0.5049	0.5051	0.5047
K_{proc} error	0.973%	1.01%	0.944%

Table 4.6: Kproc error for higher order processes

(4) Compare the results of (2) and (3) using (3) as the reference.

Results

The test described above was carried out for each transfer function and the following figures obtained (Tables 4.6 to 4.8). It must be emphasised that the value of K_{proc} was carried forward and used in the determination of the other two parameters according to the requirements of 4.3.1. This has the effect of cascading the error.

	2nd order G_1	3rd order G_2	4th order G_3
T_{proc} by CQOT	1.600[s]	2.003[s]	1.273[s]
T_{proc} by 4.3.4	1.559[s]	1.951[s]	1.284[s]
T_{proc} error	2.58%	2.62%	0.857%

Table 4.7: Tproc error for higher order processes

	2nd order G_1	3rd order G_2	4th order G_3
L_{proc} by CQOT	10.47[s]	11.13[s]	10.66[s]
L_{proc} by 4.3.4	10.41[s]	11.09[s]	10.62[s]
L_{proc} error	0.576%	0.321%	0.385%
Actual dead time	10[s]	10[s]	10[s]

Table 4.8: Lproc error for higher order processes

If a 5% error is acceptable for estimations in the absence of noise, the results above meet this criterion well for all parameters.

Comparisons of the open-loop step responses of the three higher-order models and the models obtained by the current method are given in Figures 4-10(a), (b) and (c).

4.3.7 Use of Randomised Training Set

The accuracy of T_{proc} and K_{proc} estimation was one order of magnitude more sensitive to measurement noise than the estimation of L_{proc} (see tables 4.4 and 4.5). In order to assess the benefits of a randomised training set as applied in section 3.3.8, the training of ANN #1 for K_{proc} and ANN #2 for T_{proc} estimation was repeated using the method of 3.3.8 with a factor of 0.001. The results of the training and the initial testing are shown in column 3 of the tables. The results for the previous ANNs which were trained without a random factor are repeated in column 2 for comparison. The last three rows of Tables 4.9 and 4.10 relate to a test consisting of 100 trials without measurement noise.

The improvement in accuracy and consistency is achieved at the expense of a slight increase in the number of neurons in each case.

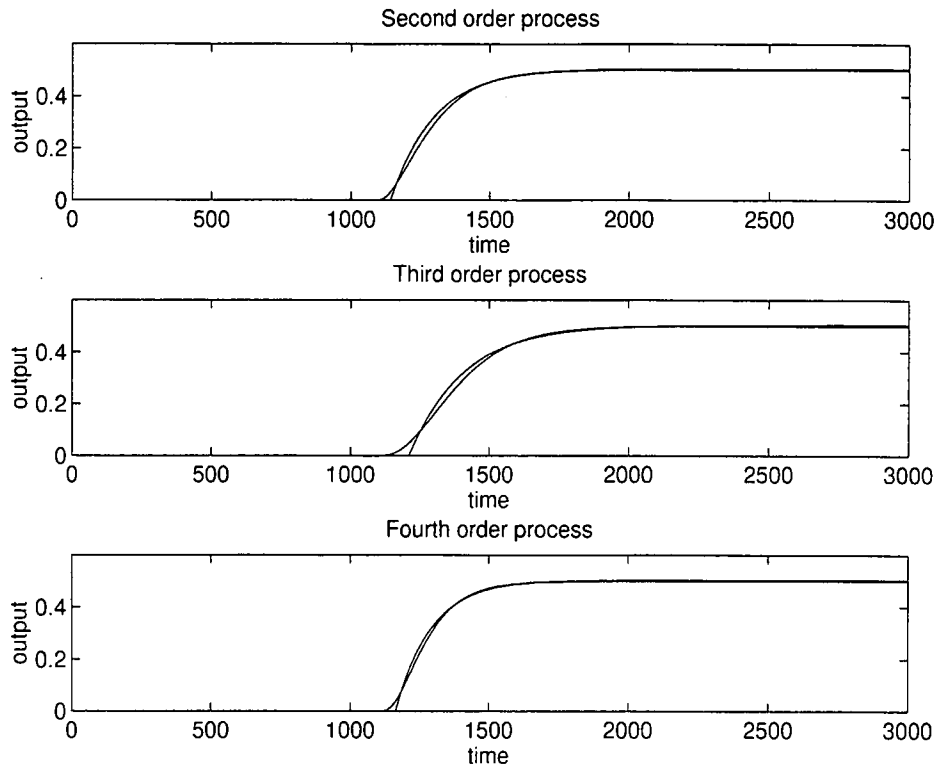

Figure 4-10: Open loop step response for processes of order 2, 3 and 4 and their FOPDT approximations.

The performance of the ANNs trained with the random factor is tested below in respect of the higher order process examples and measurement noise.

4.3.8 Randomised training method tested with higher order processes

The two ANNs produced in Section 4.3.7 above using the randomised training set were applied to the same three higher-order processes in single-shot tests without measurement noise. The results are given in Tables 4.11 to 4.13. L_{proc} estimation with higher order processes was satisfactory using the original ANN and no attempt was made to retrain. A slight improvement in K_{proc} estimation was noted, but the accuracy of T_{proc} estimation

ANN #1, K_{proc} estimator	without random factor	with random factor of 0.001
sum-squared error	0.001	0.001
spread constant	22	22
number of hidden layer neurons	203	221
time taken to train	6.31[h]	7.18[h]
maximum absolute error	4.76%	3.27%
standard deviation	0.908%	0.675%

Table 4.9: Effect of randomising training set for Kproc

ANN #2, T_{proc} estimator	without random factor	with random factor of 0.001
sum-squared error	0.05	0.05
spread constant	20	20
number of hidden layer neurons	152	211
time taken to train	3.92[h]	6.57[h]
maximum absolute error	1.36%	0.786%
standard deviation	0.313%	0.185%

Table 4.10: Effect of randomising training set for Tproc

is impaired. The effect of the more accurate estimates for K_{proc} on L_{proc} estimation is insignificant.

The conclusion is that the randomised training of ANN #1 and ANN #2 did not produce significant improvements in the estimation of the chosen higher order processes.

4.3.9 Randomised training method tested with noisy measurements

The noise tests of Section 4.3.5 are now repeated in respect of K_{proc} and T_{proc} estimation using the ANNs trained with a random factor of 0.001, and in respect of L_{proc} estimation.

random factor = 0.001	G_1	G_2	G_3
K_{proc} actual value	0.5	0.5	0.5
K_{proc} by 4.3.1	0.5018	0.5026	0.5014
K_{proc} error	0.358%	0.522%	0.281%

Table 4.11: Kproc error for higher order processes

random factor = 0.001	G_1	G_2	G_3
T_{proc} by CQOT	1.600[s]	2.003[s]	1.273[s]
T_{proc} by 4.3.1	1.501[s]	1.887[s]	1.212[s]
T_{proc} error	6.21%	5.80%	4.76%

Table 4.12: Tproc error for higher order processes

random factor = 0.001	2nd order G_1	3rd order G_2	4th order G_3
L_{proc} by CQOT	10.47[s]	11.13[s]	10.66[s]
L_{proc} by 4.3.4	10.51[s]	11.16[s]	10.73[s]
L_{proc} error	0.377%	0.301%	0.646%
Actual dead time	10[s]	10[s]	10[s]

Table 4.13: Lproc error for higher order processes

Figures for L_{proc} are not quoted because ANN #3 was not modified.

In comparison with Tables 4.3 to 4.5, the maximum error and standard deviation of the error for both K_{proc} and T_{proc} estimation has been reduced in all cases. It is noted that the maximum permitted measurement noise using the previously established criterion is still 0.001[$units$]. What is significant, however, is that the error with this level of measurement noise has been brought below the 5% mark. This is sufficient justification for use of this technique for this application.

4.4 Evaluation of the Frequency Domain Method

This evaluation consists of a summary of test results and observations from this chapter. It is arranged and presented in terms of the research objectives set out in Chapter 1.

100 trials, zero noise	max. abs. error	*std* of error
K_{proc} ($factor = 0.001$)	3.27%	0.675%
T_{proc} ($factor = 0.001$)	0.786%	0.185%

Table 4.14: Effect of randomised training set

100 trials, peak noise 0.001[*units*]	max. abs. error	*std* of error
K_{proc} $(factor = 0.001)$	4.98%	1.02%
T_{proc} $(factor = 0.001)$	4.30%	0.852%

Table 4.15: Effect of randomised training set on errors introduced by measurement noise

100 trials, peak noise 0.002[*units*]	max. abs. error	*std* of error
K_{proc} $(factor = 0.001)$	9.54%	2.05%
T_{proc} $(factor = 0.001)$	10.15%	1.82%

Table 4.16: Effect of randomised training set on errors introduced by measurement noise

4.4.1 Ability to determine process parameters

The method is able to achieve this under ideal conditions for a FOPDT process with a maximum absolute error of 3.27% for K_{proc}, 1.36% for T_{proc} and 1.17% for L_{proc}. These results were achieved using a training technique which introduced a random variation of 0.1% to each element of each training vector.

The restriction of the lower value of process dead time to 8.0[s] is artificial. Unlike the method of Section 3.3, the lower limit can in principle be extended to zero. This would make the method more widely applicable.

4.4.2 Use of one test with minimal disturbance to operation

In evaluating the disruptive effect that this method might have on a process being measured, it must be reported that the requirement of a single, short test was not met. Two such tests are required for the test developed here. It was however demonstrated in simulation that the test impulse and the adjustments to the PID controller do not produce any instability problems.

4.4.3 Application of test in closed loop mode

The main feedback loop is retained, although the method does require the disabling of the Smith predictor models.

4.4.4 Use of ANNs wherever possible

The method uses ANNs for estimation of all three process parameters. The principle of using ANNs for conversion of process parameters to PID controller settings is developed in Chapter 6.

4.4.5 Use of information from frequency domain

This method has been applied with minimal pre-processing of measured data.

4.4.6 Applicability to higher order processes

It is noted that the randomisation of the training set was not helpful in reducing error for the examples chosen. The original method produced excellent results for all three parameters, the highest individual error being 2.62%.

4.4.7 Ability to work with noisy data

The method has been shown to produce useful results when tested with FOPDT models using a measurement noise level of $0.001[units]$ applied to a standardised low-pass filter. The extreme robustness of this method for determining process gain and dead time in the presence of noise is noted. The frequency domain method of this chapter is able to tolerate measurement noise that is greater by two orders of magnitude when compared to the time domain method of Chapter 3.

Chapter 5

NEURON MODEL-BASED AUTOTUNING USING ANNs

5.1 Introduction

In Section 5.2 we demonstrate the use of a linear neuron as a process model along the lines suggested by Pham & Liu (1995) or Ahn (1994). It is then shown that the functional relationship between the linear neuron weights and the first-order model parameters can be used to provide an estimate of these parameters by means of suitably trained ANNs. For the exploratory technique proposed in Section 5.2, dead time is assumed zero.

In Section 5.3, the determination of dead time is linked to the model approach by using a numerical method that experimentally shifts a selection window along the recorded output waveform so as to simulate the recording of output waveforms with progressively larger delays. These progressively delayed waveforms are applied sequentially to the trained ANNs in a systematic way until results within the permitted range are produced. It is shown that these results give an accurate determination of process gain, and a range of mutually dependent values for process time constant and dead time. The method of 5.3 is able to fix the exact value of L_{proc} given T_{proc}, and *vice versa.*

5.2 Determination of First Order Parameters, Zero Dead Time

5.2.1 Overview

These results are taken from McLeod & Bajić (1997b). They demonstrate in simulation the basis for a scheme in which ANNs may be applied to converting the weights of a linear neuron process model to numerical estimates of process gain and time constant.

5.2.2 Construction of ANN for Determination of K_{proc}

Generation of training set for K_{proc}

A simulation as shown in Figure 5-1 was set up.

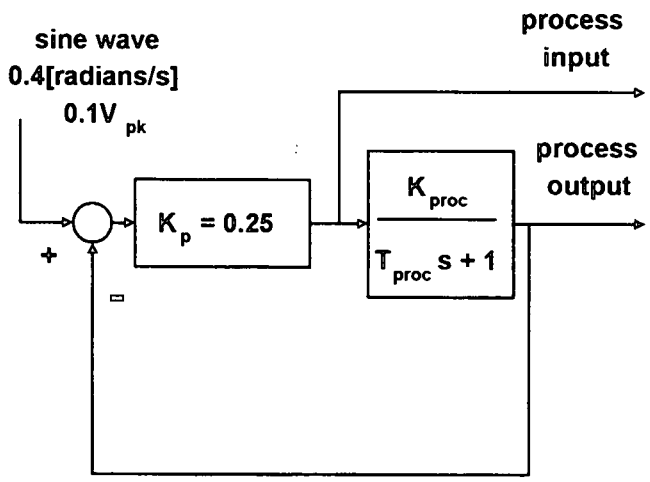

Figure 5-1: Recording of training matrix

One hundred logarithmically spaced values for T_{proc} were selected in the range $[0.05, 5.0]$. Ten linearly spaced values for K_{proc} were selected in the range $[0.2, 2.0]$. The simulation integration step size was set to $0.01[s]$ and the simulation period to $100[s]$. The sinusoidal input signal was set to $0.4[radians/s]$ with a peak value of $0.2[units]$.

The simulation was run for all 1000 combinations of the values selected for K_{proc} and T_{proc}. In each case input and output signals were recorded. The output signal was processed numerically to introduce a time offset of $0.1[s]$. This delayed version of the output and the undelayed input signal formed the input training set. The output signal formed the output training set. The Matlab NN-Toolbox function SOLVELIN was used to construct a two-input, single-output linear neuron. The neuron weights, $W1$ and $W2$, were recorded in each instance together with the process parameter values that produced them. A schematic representation of the linear neuron adapted from Demuth & Beale (1994) is shown in Figure 5-2.

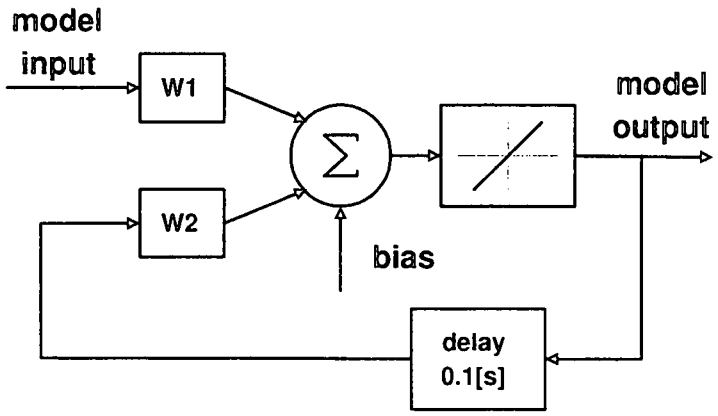

Figure 5-2: Schematic of linear neuron

SOLVELIN also produces a bias weight to model any DC offset that may be present. Since the strategy was to exclude any DC offset from the training set, the bias value generated served as a check on the correct progress of the training. This would have to be essentially zero. Inspection showed that the highest absolute bias level during generation of the training set was 6.27×10^{-7}.

Training ANN #1 for determination of K_{proc}

Graphing of the data showed that K_{proc} is dependent upon $W1$ and $W2$ (see Figure 5-3).

It was decided to develop a two-input, single-output radial basis function network

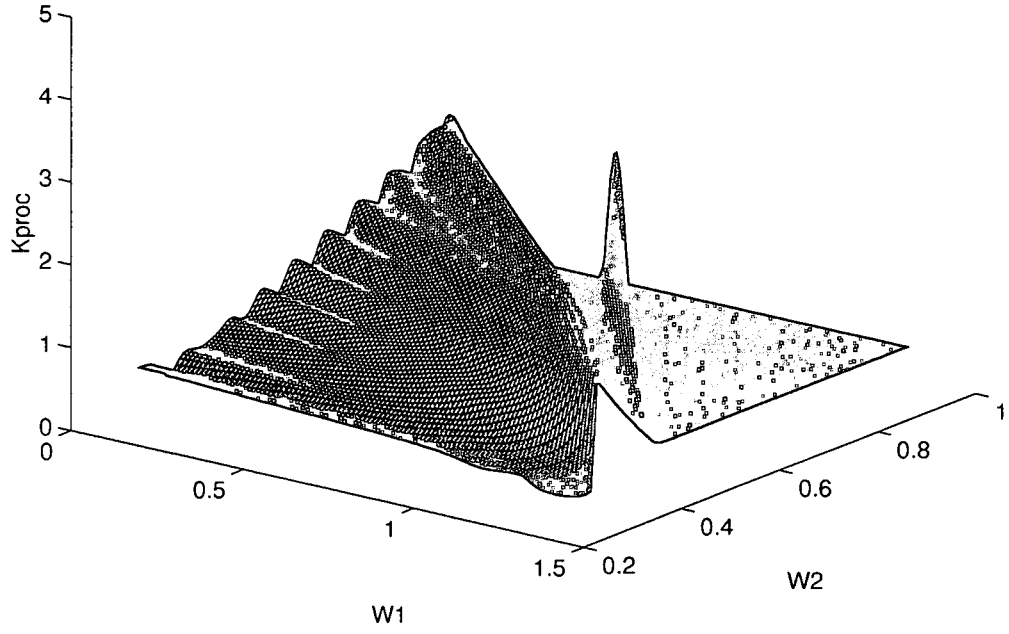

Figure 5-3: Mesh plot of K_{proc} against neuron weights

using the SOLVERB algorithm offered by the Matlab NN-Toolbox. Training to a sum-squared error (sse) of 2×10^{-5} was achieved within 417 epochs using a spread constant of 0.037.

5.2.3 Construction of ANN for Determination of T_{proc}

Generation of NN training set for T_{proc}

The simulation used was the same as for the K_{proc} training set.

Inspection of the data from the K_{proc} training process showed that the values of $W2$ alone could be used to determine T_{proc}. This is depicted in Figure 5-4 which shows two scatter plots, each consisting of 1000 points. In Figure 5-4(a) the values of $W1$ are plotted against the corresponding value of T_{proc} and in Figure 5-4(b) the values of $W2$ are plotted

in the same way.

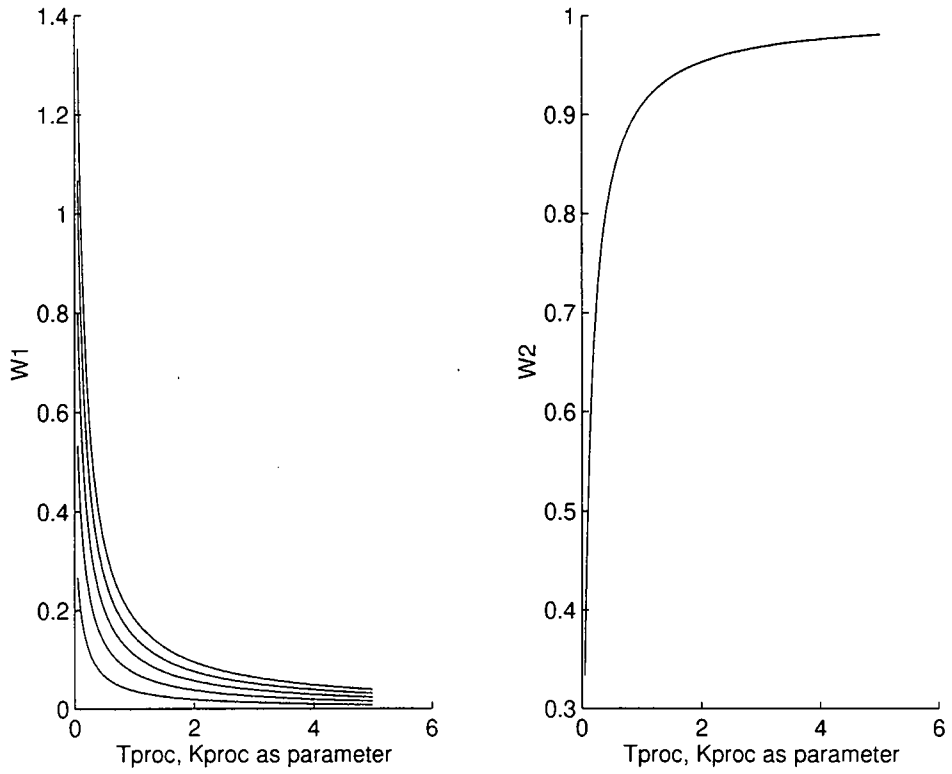

Figure 5-4: Plot of (a) $W1$ and (b)$W2$ against T_{proc}

It can be seen that the five distinct values used for K_{proc} do not have any effect on the $W2$ values. T_{proc} can therefore be uniquely determined by the value of $W2$ alone. The input training set for T_{proc} was therefore constructed by using the first 100 $W2$ values from the training set for K_{proc}. The output training set consisted of the 100 corresponding T_{proc} values.

Training ANN #2 for determination of T_{proc}

It was again decided to use a radial basis function network, and a single-input, single-output network was set up using the same technique as for the K_{proc} network. Training to a sum-squared error of 2×10^{-5} was achieved within 36 epochs using a spread constant of 0.024.

1000 trials	$K_{proc}[\%]$	$T_{proc}[\%]$
max absolute error	2.536	2.794
std of error	0.2398	0.8646
mean error	−0.0634	1.270

Table 5.1: Results of random testing

Network schematic

The determination of process parameters using the trained ANNs can take place simultaneously because neither depends on results from the other. This can be represented as shown in Figure 5-5.

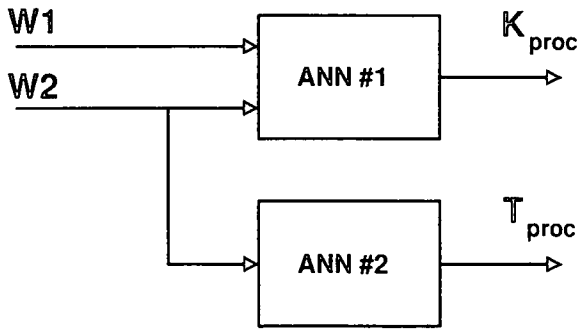

Figure 5-5: Complete network

5.2.4 Testing the Method in Simulation

The simulation of Figure 5-6 was used.

Initial testing showed that error increased to unacceptable levels for lower values of T_{proc}. The range of T_{proc} was truncated to $[0.5, 5]$ to keep the maximum error below 3%. A test consisting of 1000 trials with random values of L_{proc}, K_{proc} and T_{proc} was run. The results are given in Table 5.1.

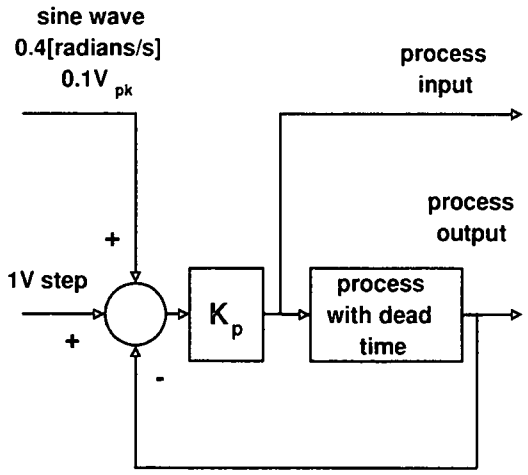

Figure 5-6: Recording of input and output of process to be identified

5.3 Extension of Model Approach to Systems in which Dead Time is Not Known

5.3.1 Overview

In Section 5.2 dead time was assumed known. It was noted that the system was able to tolerate a small error in the dead time and produce usable results in spite of the output graph not being exactly one step out of phase with the input graph. This effect was explored in a systematic manner and led to an unusual method for determining the process gain which is described in this section. It exploits a plateau effect in process gain that is shown to represent the range of mutual accommodation between process time constant and dead time. This was first noticed in a plot similar to the one shown in Figure 5-7 where the time delay between input and output recordings was adjusted experimentally so as to represent the full range of offset due to dead time. In the simulation which generated Figure 5-7, the value of K_{proc} was 0.6 and L_{proc} was 8[s]. Expansion of the scale shows that this value is reached and maintained from points 81 to 85. This phenomenon was investigated and developed into a method for determining upper and lower limits for

the actual dead time.

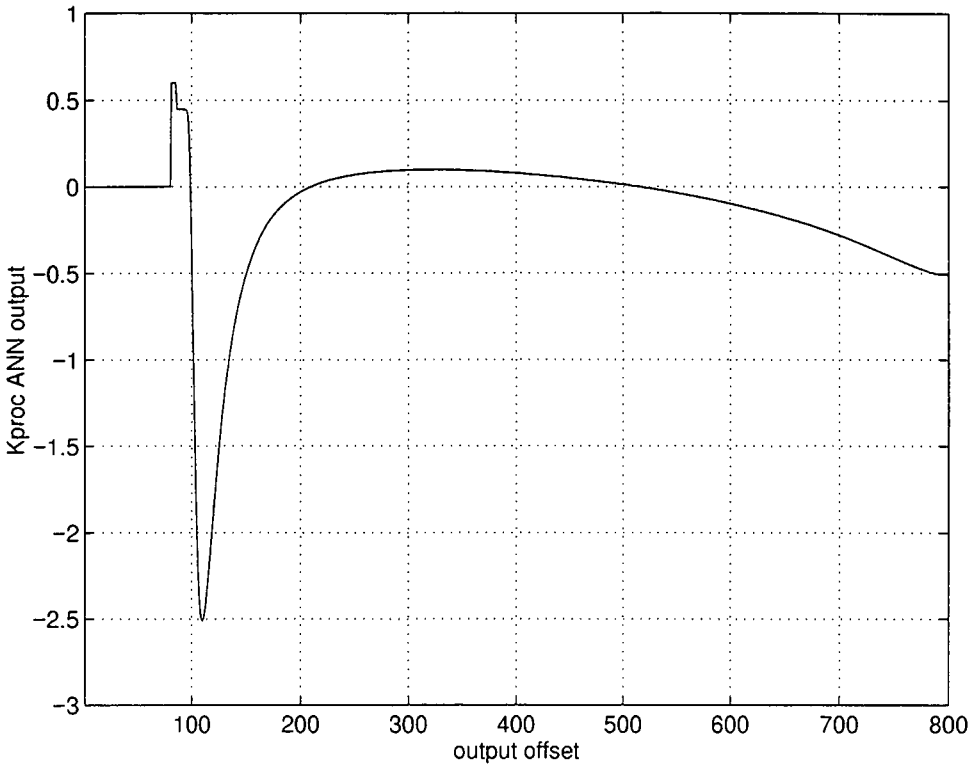

Figure 5-7: Plateau at which $K_{proc} = 0.6$

This extension of the process of Section 5.2 involves the same experimental sliding effect in which the recording of the output is experimentally offered to both the K_{proc} ANN and the T_{proc} ANNs (ANN #1 and #2 of Figure 5-5). In interpreting the outputs logical criteria are applied: if either of the ANN outputs are outside the permitted range of the relevant parameter, the output of both ANNs is suppressed to zero. On completion of this test, adjacent values of the K_{proc} output that have not been suppressed are examined to detect the plateau. The value of K_{proc} at this plateau is the value returned by the system as the K_{proc} estimate. The extent of the plateau is used to calculate the upper and lower bound for L_{proc}. Within this range, the graph of the output of the T_{proc} ANN is a straight line. The method reflects the interdependence of process time constant and dead time. Any dead time, L_{proc}, on the abovementioned plateau may be selected, and the

corresponding value of T_{proc} read off from the graph of the T_{proc} ANN output. It follows that this method is capable of making an entirely valid FOPDT approximation, but that an accurate determination of the actual time constant and dead time is inherently impossible by this route.

The technique described below was designed to determine $K_{proc} \in [0.2, 2]$, $T_{proc} \in [0.05, 5]$ and $L_{proc} \in [8, 80]$ in the context of a Smith predictor. The general arrangement is as shown in Figure 5-8. It is functionally equivalent to Figure 5-6. The difference lies in the numerical processing of the recorded waveform. The method uses a windowing technique to select continuous segments of the process output waveform that are shifted progressively with respect to the time frame of the process input waveform so as to simulate progressively longer delays.

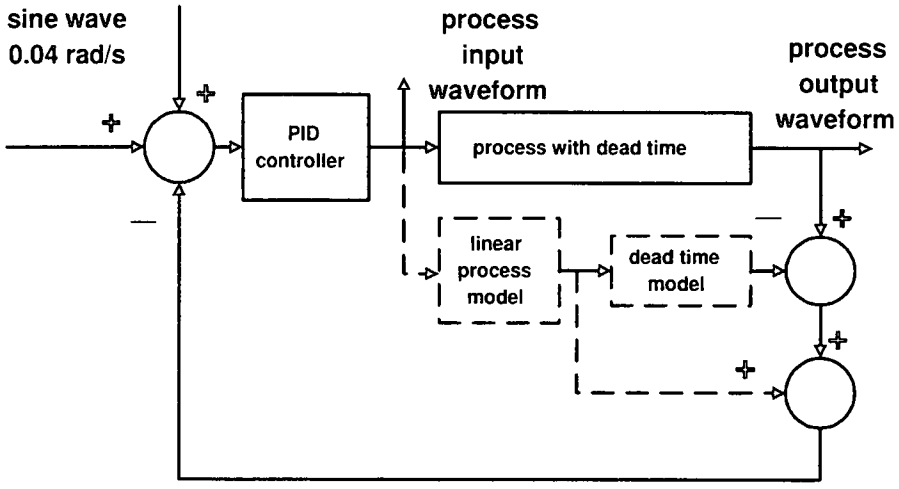

Figure 5-8: Smith predictor scheme for neuron model method

5.3.2 Application of the Method

The following steps are followed in making a test:

1. Set controller proportional gain to 0.25, disable integral and derivative components. Disable Smith Predictor components.

100 trials	K_{proc}
max absolute error	3.258%
standard deviation of the error	1.302%

Table 5.2: Testing with first order processes

2. Apply test waveform. Record input for 80 seconds and output for 160 seconds at 10 samples per second.

3. Repeatedly apply input and output waveforms to a linear 2-input neuron modelled using the SOLVELIN function of MATLAB 4c. In each repetition, increase the delay from a minimum of $8[s]$ to $80[s]$ in steps of $0.1[s]$. In each case apply the resulting weights to the trained ANNs that yield K_{proc} and T_{proc} and store the results.

4. Examine the values of ANN output against the delay steps. Confine attention to delay values that produced legal values of K_{proc} and T_{proc}. Within this confined range, identify on the K_{proc} graph of ANN output a plateau consisting of at least five contiguous values that are within 0.001 of one another. Read this value and return it as K_{proc} for the system.

5. Read off the values of L_{proc} at the start and finish of the viable region of step 4. Return these values as upper and lower bounds for L_{proc} for the system.

5.3.3 Generation of the Training Set

The method uses the ANNs developed in 5.2.

5.3.4 ANN Structure and Training

The method uses the ANNs developed in 5.2.

5.3.5 Testing the Method in Simulation (First Order Processes)

The following results were obtained in 100 simulations, using random values of parameters.

In every instance, the actual value of L_{proc} was within the upper and lower bounds determined by the method. These upper and lower bounds had a maximum range of $4.63[s]$.

The graph of K_{proc} ANN output against offset is shown in Figure 5-9. Segment AB is the plateau referred to in step 4 of 5.3.2. In this case the method returned a value of 0.6025 for K_{proc}. The actual value, indicated by the horizontal line, is 0.5941. The dead time bounds determined by the method are shown by the vertical lines as $[52.5, 57.2]$. The actual dead time is $56.89[s]$.

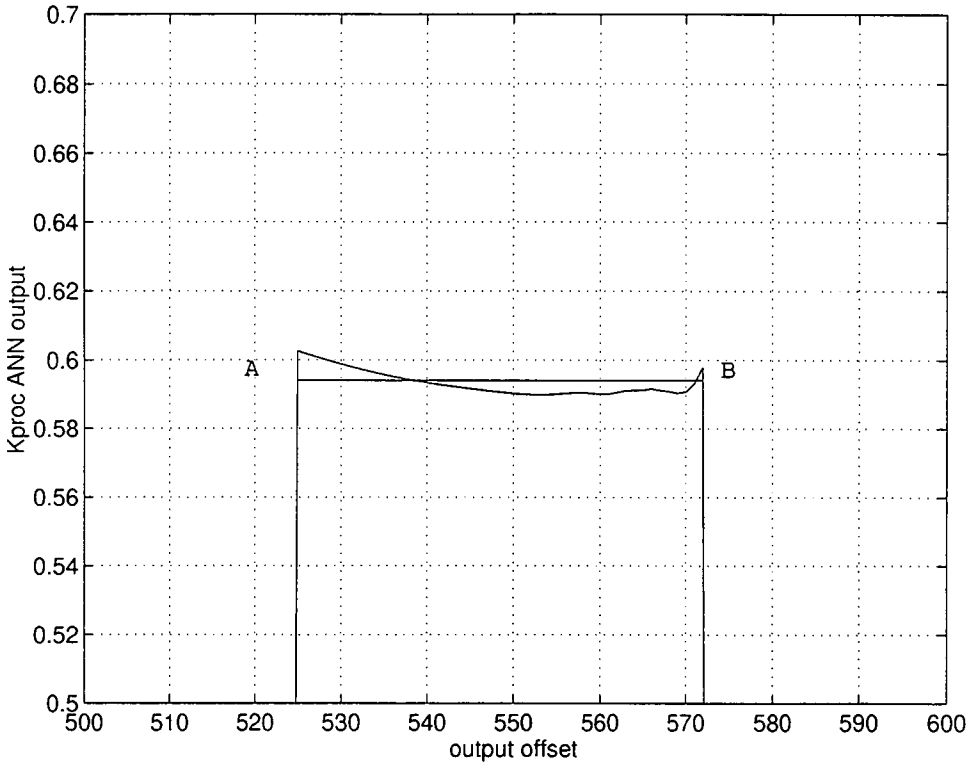

Figure 5-9: K_{proc} ANN output against offset with $T_{proc} = 0.2824[s]$ and $L_{proc} = 56.89[s]$

Figure 5-10 shows a straight line relationship between the output of the ANN that estimates T_{proc} and dead time. This result follows from the elementary nature of the process model. The actual value of T_{proc} (0.2824) is indicated by the horizontal line, and the intersection is seen to be very close to the actual dead time of $56.89[s]$. Figure 5-11,

which shows plots for three different values of T_{proc} (0.1, 1.0, and 3.0) and a constant dead time of 56.89[s], illustrates that the method can be used to determine T_{proc} given L_{proc} or *vice versa*.

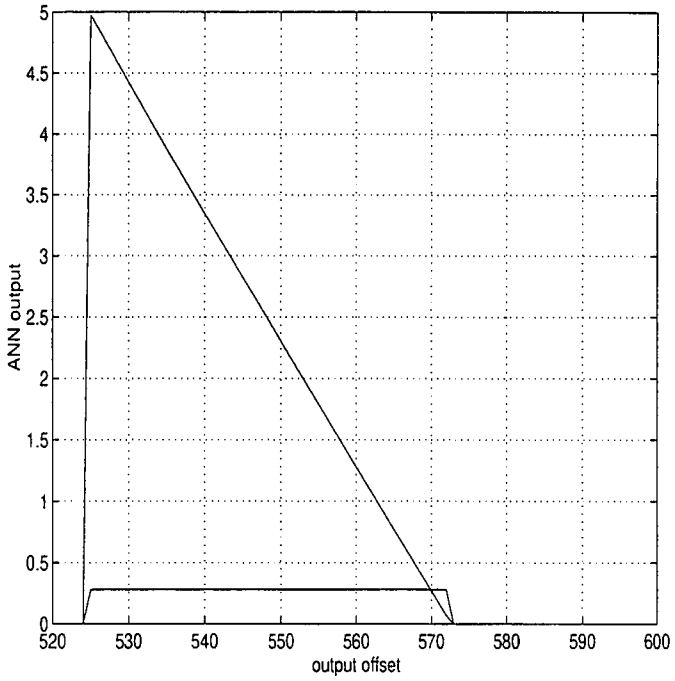

Figure 5-10: T_{proc} ANN output against offset

5.3.6 Testing with Measurement Noise

In line with the tests conducted in Sections 3.3.5 and 4.3.6, the tests of Section 5.3.5 were repeated with noise added to the process output. The sinusoidal perturbation applied to the system was adjusted to 0.024[*units*] peak so as to produce a peak deviation at the output equivalent to the peak deviation obtained with the spike pulses used in sections 3.3.5 and 4.3.6. The results of three 10-trial tests using random values of process parameter are summarised in Table 5.3.

In this system, the noise affects the training of the model rather than the behaviour of the static ANNs as in the time and frequency domain methods of earlier chapters. It

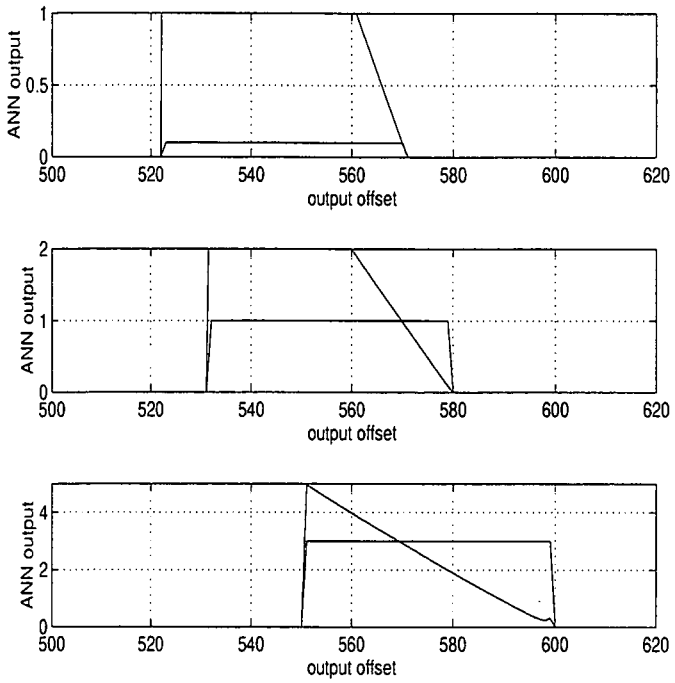

Figure 5-11: T_{proc} ANN output against offset for three different values of T_{proc}

10 trials	noise (peak)	max. abs. error	std of error
K_{proc} determination (zero noise)	$0[units]$	2.85%	1.64%
K_{proc} determination ($error \leq 10\%$)	$0.003[units]$	5.21%	1.92%
K_{proc} determination ($error > 10\%$)	$0.004[units]$	13.48%	5.58%

Table 5.3: Effect of noise

is noted that the performance in the presence of noise is clearly superior with respect to K_{proc} determination when compared to the time domain method. It is comparable to the frequency domain method in this respect. As an aid to visualisation, a segment of the sinusoidal training signal with added measurement noise of $0.003[units]$ is shown in Figure 5.12. The values of process parameter used for Fig. 5.12 are $K_{proc} = 0.2$, $T_{proc} = 5.0[s]$ and $L_{proc} = 8.0[s]$.

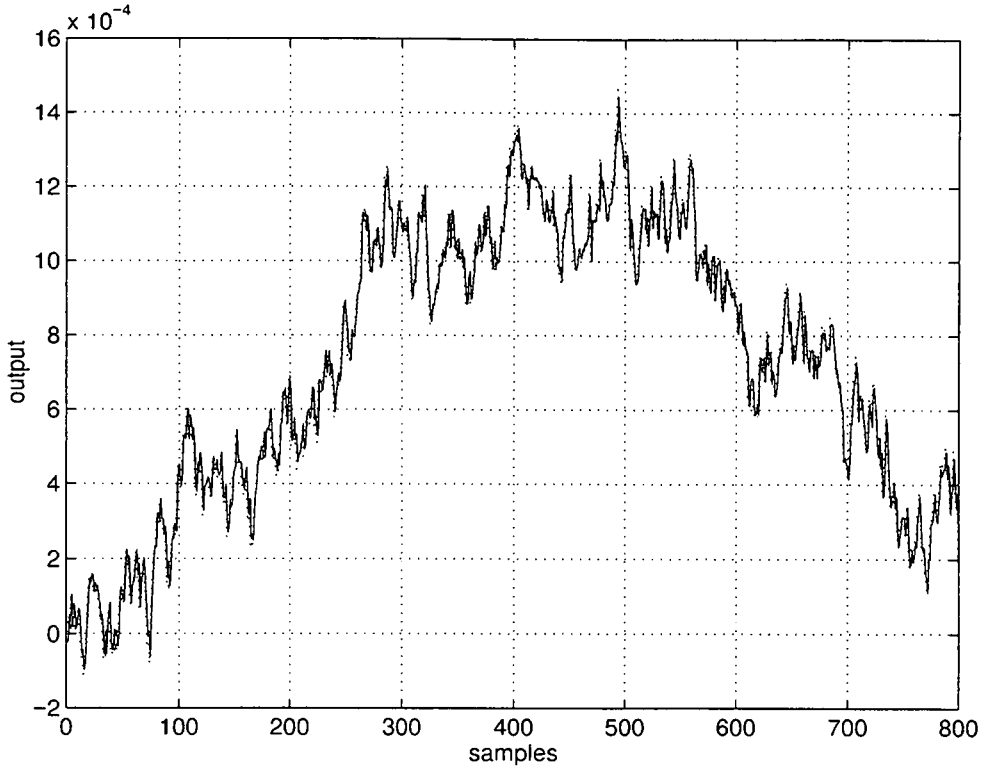

Figure 5-12: Effect of measurement noise of 0.003[*units*] peak

5.3.7 Testing with Higher Order Processes

The following higher order systems introduced in Section 2.3.6 were used as a basis for further tests:

$$G_1(s) = \frac{0.5e^{-10s}}{(1+s)^2}$$

$$G_2(s) = \frac{0.5e^{-10s}}{(1+s)^3}$$

$$G_3(s) = \frac{0.5e^{-10s}}{(1+s)(1+0.5s)(1+0.25s)(1+0.125s)}$$

The test consisted of the following steps of simulation:

(1) Set up the process in a Smith Predictor system with a P controller.

(2) Use the method of 4.3.2 to obtain an approximation for K_{proc} and upper and lower bounds for L_{proc}.

	G_1	G_2	G_3
K_{proc} actual value	0.5	0.5	0.5
K_{proc} by 5.3.2	0.5011	0.5051	0.5013
K_{proc} error	0.216%	1.02%	0.252%
T_{proc} by CQOT	1.600[s]	2.003[s]	1.273[s]
L_{proc} by CQOT	10.47[s]	11.13[s]	10.66[s]
L_{proc} lower bound by 5.3.2	8.1[s]	8.3[s]	8.1[s]
L_{proc} upper bound by 5.3.2	12.0[s]	13.1[s]	11.9[s]
Actual dead time	10[s]	10[s]	10[s]

Table 5.4: Testing with higher order processes

(3) Use the constrained quadratic optimisation technique (CQOT) described in Section 3.3.6 (step 3) to obtain FOPDT approximations for process time constant T_{proc} and dead time L_{proc} for the given process.

(4) Compare the results of (2) and (3) using (3) as the reference.

The results are given in Table 5.4.

The determination of K_{proc} by this method for the higher order models selected is entirely satisfactory with a worst-case error of 1.02%.

Dead time is always within the upper and lower limits returned by this method, and the worst case distance of a limit from the CQOT estimate for L_{proc} is 26%.

5.4 Evaluation of the Model Based Method

Although the underlying computational basis for this method could be set out in purely numerical terms, neural networks provide a useful conceptual structure. Although the method is not by itself able to provide estimates for all FOPDT parameters, the techniques described can, if necessary, be used in combination with other methods.

5.4.1 Ability to determine process parameters

The method of Section 5.3 is able to determine K_{proc} with a worst-case error of 3.3%. Although it cannot give an accurate value for L_{proc}, it is able to determine upper and

lower bounds that place it within a range of about 5 seconds. The method does not allow T_{proc} to be determined unless L_{proc} is known accurately. In such a case, the method of Section 5.2 can be applied.

5.4.2 Use of one test with minimal disturbance to operation

The requirement of a single test was met, and adjustments to the PID controller are not critical, as long as stability is maintained. The duration of the test is $160[s]$, but the test signal can be a small percentage of the reference value.

5.4.3 Application of test in closed loop mode

The main feedback loop is retained, although the method does require the disabling of the Smith predictor models.

5.4.4 Use of ANNs wherever possible

This method makes use of two static ANNs and makes use of the features offered by a linear neuron construct in a commercial software product to model a dynamic process. Further use can be made of ANNs in producing controller parameters. The principle of using ANNs for conversion of process parameters to PID controller settings is developed in Chapter 6.

5.4.5 Use of a neural model method

The method incorporates a simple ARX model based on a single neuron. This forms the basis for the "sliding" technique in that it undergoes training every time the waveform is shifted experimentally by one time step.

5.4.6 Ability to work with noisy data

The primary effect of measurement noise in this method is that it disrupts the training of the ARX model. Filtered white noise with a peak value of $0.004[units]$ was sufficient to render the method unusable. In this aspect it is on a par with the frequency domain method of Chapter 3. The method of randomising the training set developed in Chapter 3 aimed at making a static ANN better able to interpolate inputs that depart from the overall shape of the training set inputs is not applicable in this context.

Chapter 6

USE OF ANNs TO DETERMINE CONTROLLER SETTINGS

6.1 Introduction

The Smith Predictor configuration (Smith 1958), a model-based controller system, is a widely accepted method of dealing with processes with long dead times. In his review of accepted practice for dealing with processes with long dead time, Hägglund (1992) reports that the Smith Predictor is usually used with a PI controller. This is ascribed to the inappropriateness of the derivative control element in this context because it reacts to events that took place too far in the past.

This view is supported by De Carvalho (1993, p.110) who states that derivative action is not beneficial in processes with long dead time, and he also asserts that derivative action is not suitable for any process that is predominantly first order. De Carvalho (1993) gives a thorough treatment of this issue based on the work of Yamanaka & Shimemura (1987) in which he shows that derivative action may be applied to a Smith predictor system in certain instances.

In contrast with the above views, Govender and Bajić (1996) describe the use of a nonlinear modifier in conjunction with standard PID control in the context of a Smith

predictor. This improves setpoint tracking without affecting disturbance rejection, and is capable of tuning out minor model mismatches. This nonlinear modifier effectively uses proportional and derivative action in a nonlinear fashion and is able to control systems with very long dead time. A Fuzzy Logic version of this method is described in Govender & Bajić (1997).

As discussed in McLeod & Bajić (1996a) there are several methods for tuning a PID controller when the model of the process is known. In Section 2.1.3 reference has been made to the methods of Shinskey (1979), Cohen & Coon (1952), and Ziegler & Nichols (1942, 1943). To these can be added the methods of Chrones *et al.* (1952), Yuwana & Seborg (1982), Aström & Hägglund (1988) and Hang & Sin (1991). Any one of these methods may be selected as the basis for training an ANN. It is also possible to apply any desired technique to determine the best controller values in line with specified control optimisation criteria in order to generate a training set. Extending this approach, it becomes possible to create a 'library' of ANNs, each one offering a different control strategy, so that a suitable optimisation method may simply be selected by software.

The use of static ANNs for calculation of controller parameters from formulas such as the Cohen-Coon formulas (McLeod & Bajić 1996b) or the empirically derived formulas of McLeod & Bajić (1996a) when standard computing facilities are available is of questionable value. There is perhaps some justification for this approach when the ANNs are trained to yield controller parameters directly from the input data without any numerical representation of the actual process values.

The static ANN is particularly suited to the representation of highly nonlinear relationships that have been determined experimentally and which are difficult to express in mathematical terms. The representation is compact, and interpolation is automatic. This approach was summarised in McLeod & Bajić (1997b). Details of this project are given below.

6.2 Tuning Data for Optimal Loop Behaviour

The general approach to ANN-based tuning is summarised in Figure 6-1. Block 1 would be implemented using one or more of the methods described in Chapters 3 to 5. Blocks 2 and 3 could be incorporated with block 1 in a portable device that implements block 4 by displaying the settings for manual adjustments. The system could also be implemented as a networked digital controller with a suitable disposition of the storage and processing functions.

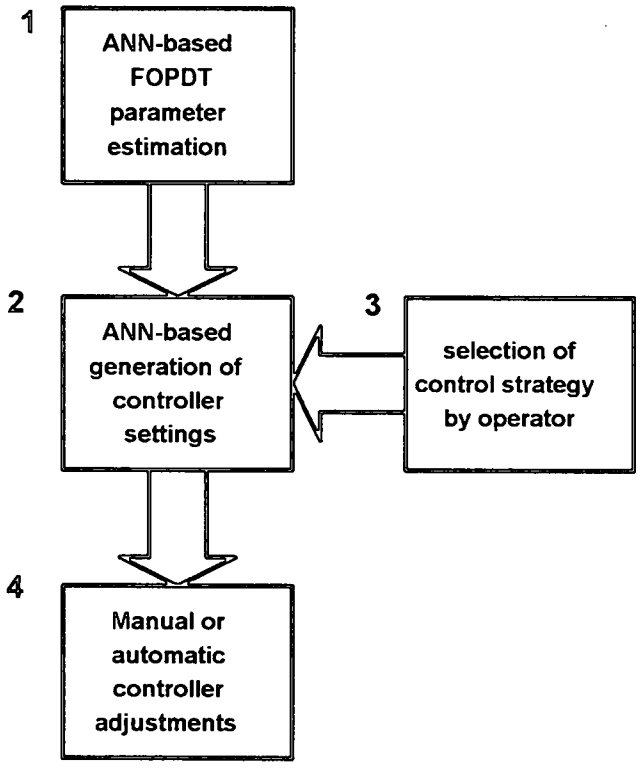

Figure 6-1: Outline of method

Simulation of a PID controller with FOPDT process was used to obtain values for K_p, K_i and K_d for optimal disturbance rejection for 10 linearly spaced values of K_{proc} over the range $[0.2, 2]$ and 10 linearly spaced values of T_{proc} over the range $[0.5, 5]$. PID controller parameters were generated by means of a constrained optimisation method

	K_p	K_d
number of hidden layer neurons	98	70
spread constant	0.3	0.3
sum-squared error (sse)	0.1	0.01

Table 6.1: Results of training

which optimised step type disturbance rejection under the constraint of a maximum overshoot of 20% and restriction of the PID gain values to the interval $[0, 20]$. The criterion function was the sum of absolute errors between the recorded signal and ideal response. The optimisation method yielded an integral gain of 20 in all cases, but the values of K_p and K_d were found to be highly nonlinear functions of K_{proc} and T_{proc}.

These were successfully approximated using radial basis function networks as shown in Table 6.1:

The results are shown in Figure 6-2 and Figure 6-3.

6.3 Review

The files containing the weights and biases representing computer-simulated ANNs are sufficiently compact to be used as a practical means of storing and implementing highly nonlinear empirically derived functions that represent controller settings that meet specific criteria. The low cost of magnetic media allows the data relating to an almost arbitrary number of tuning approaches to be stored and made available to the user. The approach is considered to be more convenient than interpolation from lookup tables, but is not considered as an alternative for tuning schemes that are governed by known algebraic expressions.

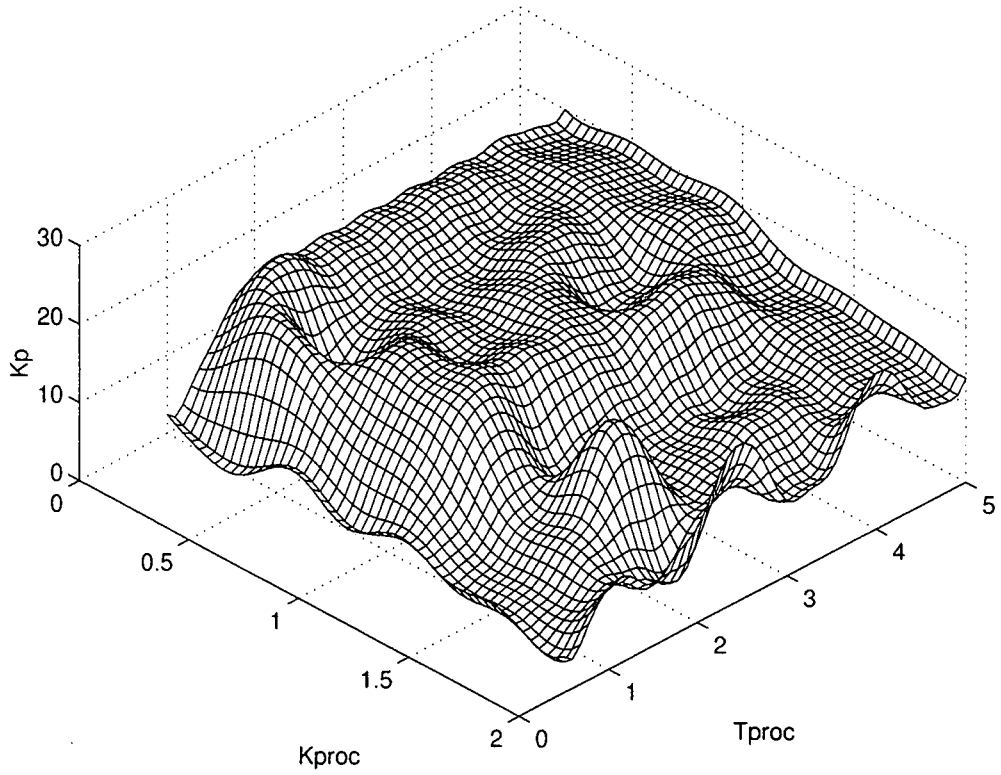

Figure 6-2: Mesh plot of optimal K_p values

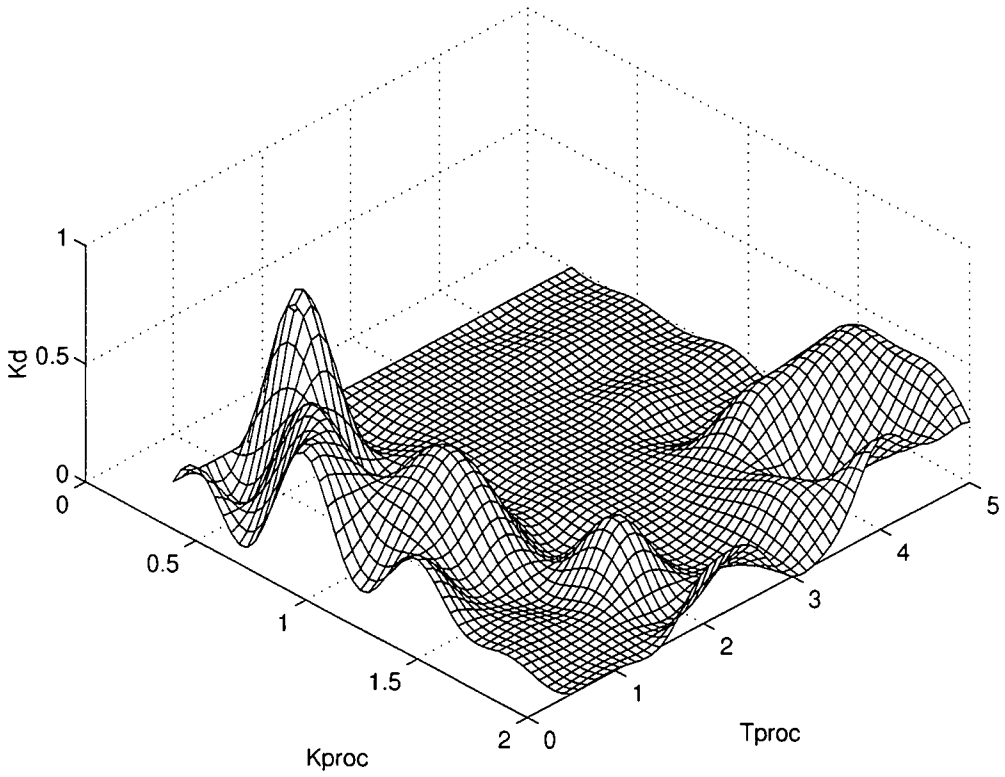

Figure 6-3: Mesh plot of optimal K_d values

Chapter 7

CONCLUSIONS

This study has presented three conceptually different approaches to the problem of process analysis, but each has employed static neural network techniques to a great extent. The intent has been to avoid feature extraction as far as possible in order to point the way towards a generalised method. The accuracy of each method has been tested in simulation under ideal conditions and under conditions of simulated measurement noise. An exposition of methods available for applying neural networks to the setting of controller parameters has been made. The methods described can be used in portable microprocessor-based analysers, or used with networked systems which permit the low-cost disk-based storage of a number of ANNs as part of a larger AI-based tuning system.

The selection of a method is best made on the basis of accuracy and robustness with noisy measurements. A system that can approximate higher order systems satisfactorily would be a bonus.

A white noise generator with a first order low-pass RC filter of unity gain and unity time constant was used as a source for all noise tests. This produced a noise signal with a low frequency component that was considered to be better representative of typical conditions than white noise. As a reflection of the approximate nature of these noise tests, the noise generator output was adjusted in steps of one significant figure only.

	Noise generator level
Time domain method	0.0003[*units*]
Frequency domain method	0.001[*units*]
Model-based method	0.003[*units*]

Table 7.1: Comparison of error levels

	K_{proc}	T_{proc}	L_{proc}
Time domain method	19.7%	11.0%	7.91%
Frequency domain method	1.01%	2.62%	0.576%
Model-based method	0.51%	n/a	n/a

Table 7.2: Error with higher order processes

Levels that permitted operation with a worst-case error of less than 10% for each of the methods presented in sections 3.3, 4.3 and 5.3 are given in Table 7.1.

The suitability of each of the components of each method for higher order systems used as the basis for testing is summarised by quoting, in Table 7.2, the highest value of absolute error for each component.

Both the time- and frequency-domain methods deliver acceptable accuracy for FOPDT systems. The model-based method is excluded at this point because of its inability to determine L_{proc} and T_{proc} in a straightforward manner. From a practical viewpoint, noise performance is seen as a critical overriding factor. On this basis, the frequency domain method is clearly superior.

If one accepts that there are situations in which measurement noise is not significant, then the noise sensitive techniques given here could form the basis for further research.

All methods presented in this study have been tested in lengthy simulations and have been shown to operate consistently and reliably within the parameters specified.

Bibliography

[1] Ahn, I.S. (1994). System parameter identification using artificial neural networks, *Proceedings of the Artificial Neural Networks in Engineering Conference*, November 13-16, St. Louis, Missouri, USA, pp.961-966.

[2] Anderson, J.A. (1968). A Memory Storage Model Utilising Spatial Correlation Functions, *Kybernetics*, Vol. 5, pp.113-119. Reprinted in Anderson, Pellionisz and Rosenfeld (1990) pp.79-86.

[3] Anderson, J.A. (1972). A Simple Neural Network Generating an Interactive Memory, *Mathematical Biosciences*, Vol. 14, pp.197-220. Reprinted in Anderson and Rosenfeld (1988) pp.181-192.

[4] Anderson, J.A., A. Pellionisz and E. Rosenfeld, Eds. (1990). *Neurocomputing 2: Directions for Research.* Cambridge, MA: MIT Press.

[5] Aström, K.J. and T. Hägglund (1988). *Automatic Tuning of PID Controllers*, Instrument Society of America (ISA), Research Triangle Park, NC.

[6] Bajić, V.B. (1994). Intelligent Control, Invited Tutorial Lecture, *Proceedings of the AMSE International Conference: Intelligent Systems - Methodologies & Applications*, September 28-30, Pretoria, RSA, pp.45-61.

[7] Bajić, V.B. (1995). Optimised tuning for feedforward retarded control based on pattern recognition, *Proceedings of International AMSE Conference*, Brno, Czech Republic, July 3-5, Vol. 3, pp.160-164.

[8] Bajić, V.B. and M. McLeod (1997). Process Identification and Assisted Controller Tuning for Industrial Process Loops, *Preprints of the IFIP, IFAC, IMACS Conference on Control of Industrial Systems*, 20-22 May, Belfort, France, Vol. 3, pp.216-221.

[9] Bajić, V.B., C.J. Tait and E.K. Bologna (1995). Is fuzzy-logic PID controller better than the conventional PID controller?, *Proceedings of the International AMSE Conference: Systems-Analysis, Control & Design*, 3-5 July, Brno, Czech Republic, Vol. 3, pp.165-173.

[10] Bologna, E., C. Tait and V.B. Bajić (1995). Closed-loop process identification for autotuning of PID controllers, *Proceedings of the Intelligent Control Symposium*, University of Natal, 20 June, Durban, South Africa.

[11] Broomhead, D.S. and D. Lowe (1988). Multivariable functional interpolation and adaptive networks, *Complex Systems 2*, pp.321-355.

[12] Brown, M. (1994). The state of PID control in South Africa, *S.A. Instrumentation & Control*, August, pp.131-136.

[13] Carpenter, G.A. and S. Grossberg (1990). ART3: Hierarchical Search Using Chemical Transmitters in Self-organising Pattern Recognition Architectures, *Neural Networks*, Vol. 3, No. 4, pp.129-152.

[14] Chen, C.L. (1989). A simple method for on-line identification and controller tuning, *A.I.Ch.E. Journal*, Vol. 35, No. 12, pp.2037-2039.

[15] Chen, S., C.F.N. Cowan and P.M. Grant (1991). Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks, *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, March, pp.302-309.

[16] Cohen, G.H. and G.A. Coon (1953). Theoretical considerations of retarded control, *Transactions of ASME*, Vol.75, pp.827-834.

[17] Demuth, H. and M. Beale (1994). *User's Guide: Neural Network Toolbox for use with Matlab*, The MathWorks Inc.

[18] De Carvalho, J. (1993). *Dynamical Systems and Automatic Control*, Prentice Hall, p.114.

[19] Eberhart, R.C. and R.W. Dobbins Eds. (1990). *Neural Network PC Tools, a Practical Guide,* Academic Press, Inc.

[20] Fausett, L. (1994). *Fundamentals of Neural Networks: Architecture, Algorithms and Applications*, Prentice Hall International, Englewood Cliffs, NJ.

[21] Gorman, R.P. and T.J. Sejnowski (1988). Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets, *Neural Networks 1*, pp.75-89.

[22] Govender, P. and V.B. Bajić (1996). Nonlinear modifier of PID control for processes with long dead time, *Proceedings of the International AMSE Conference on Communications, Signals and Systems (CSS'96)*, Brno, Czech Republic, September 10-12, Vol. 1, pp.247-250.

[23] Govender, P. and V.B. Bajić (1997). Fuzzy Logic Enhancement of Antiwindup PID Control for Servo Systems, in *Advances in Intelligent Systems (*F.C. Morabito, Ed.), IOS Press, pp.474-478.

[24] Hägglund, T. (1992). A Predictive PI Controller for Processes with Long Dead Times, *IEEE Control Systems,* February, pp.57-60.

[25] Hang, C.C., K.J. Aström and W.K. Ho (1991). Refinements of the Ziegler-Nichols tuning formula, *IEE Proceedings-D*, Vol. 138, No. 2, March, pp.111-118.

[26] Hang, C.C. and K.K. Sin (1991). On-Line Auto Tuning of PID Controllers based on the Cross-Correlation Technique, *IEEE Transactions on Industrial Electronics*, Vol.38, No.6, pp.428-437.

[27] Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation,* Macmillan.

[28] Hebb, D.O. (1949). *The Organisation of Behaviour*, New York: John Wiley & Sons. Introduction and Chapter 4 reprinted in Anderson and Rosenfeld (1988) pp.45-56.

[29] Hinton, G.E. and R.J. Sejnowski (1986). Learning and relearning in Boltzmann machines, in *Parallel distributed Processing*, Vol. 1, Ch. 7, D.E. Rumelhart and J.L. McClelland, Eds. Cambridge, MA, M.I.T. Press.

[30] Hinton, G.E., R.J. Sejnowski and D.H. Ackley (1984). Boltzmann machines: Constraint satisfaction networks that learn, *Technical Report CMU-CS-84-119*, Carnegie-Mellon University, Dept. of Computer Science.

[31] Hopfield, J.J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proceedings of the National Academy of Scientists*, Vol. 79, pp.2554-2558. Reprinted in Anderson and Rosenfeld (1988) pp.460-464.

[32] Hopfield, J.J. (1984). Neurons with Graded Response Have collective Computational Properties like Those of Two-state Neurons, *Proceedings of the National Academy of Sciences*, Vol. 81, pp.3088-3092. Reprinted in Anderson and Rosenfeld (1988) pp.579-584.

[33] Hopfield, J.J. and D.W. Tank (1985). Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics*, Vol. 52, pp.141-152.

[34] Hopfield, J.J. and D.W. Tank (1986). Computing with Neural Circuits, *Science*, Vol. 233, pp.625-633.

[35] Kohonen, T. (1972). Correlation Matrix Memories, *IEEE Transactions on Computers*, Vol. C-21, pp.353-359. Reprinted in Anderson and Rosenfeld (1988) pp.174-180.

[36] Kosko, B. (1987). Adaptive bidirectional associative memories,*Applied Optics*, Vol. 26, Dec. 1, pp.4947-4960.

[37] Kröse, B.J.A. and P.P. van der Smagt (1993). *An introduction to neural networks,* University of Amsterdam.

[38] Le Cun, Y. (1986). Learning Processes in an Asymmetric Threshold Network, in E. Bienenstock, F. Fogelman-Souli and G. Weisbuch, Eds. *Disordered Systems and Biological Organisation*, NATO ASI Series, F20, Berlin: Springer-Verlag.

[39] Lee, J. (1989). On-line PID controller tuning from a single, closed-loop test,*A.I.Ch.E. Journal*, Vol. 35, pp.329-331.

[40] Ljung, L. (1995). *System Identification Toolbox User's Guide,* The MathWorks Inc., May.

[41] Mamat, R., P.J. Fleming and A.E.B. Ruano (1995). Neural networks assisted PID autotuning, *Proceedings of the Second AI Conference on Industrial Automation*, Vol. II, pp.849-854.

[42] McCulloch, W.S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, Vol. 5, pp.115-133.

[43] McClelland, J.L. and D.E. Rumelhart (1988). *Explorations in Parallel Distributed Processing*, Cambridge MA: MIT Press.

[44] McLeod, M. and V.B. Bajić (1996a). Neural network assisted tuner for tracking improvement in process control, *Proceedings of the AMSE International Conference CSS'96*, Brno, Czech Republic, September 10-12, pp.243-246.

[45] McLeod, M. and V.B. Bajić (1996b). Neural networks for process parameter identification and assisted controller tuning for control loops, *Proceedings of the SAICSIT Annual Research and Development Symposium*, pp.127-136, Durban, South Africa, September 26-27, pp.127-136.

[46] McLeod, M. and V.B. Bajić (1997a). Measurement of Parameters of First Order System using Digital Signal Processing with Neural Network, in *Advances in Intelligent Systems* (F.C. Morabito, Ed.), IOS Press, pp.63-67

[47] McLeod, M. and V.B. Bajić (1997b). Neural Networks in Control of Processes with Very Long Dead Time, *AMSE: Proceedings of the Third Annual Conference on Modelling and Simulation MS'97*, Melbourne, Australia, 29-31 October, pp.174-179.

[48] McMillan, G.K. (1983). *Tuning and control loop performance*, an ISA Monograph, Instrument Society of America.

[49] McMillan, G.K. and S. Weiner (1987). *How to Become and Instrument Engineer*, Instrument Society of America.

[50] Narendra, K.S. and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, Vol. NN-1, March, pp.4-27.

[51] Nguyen, D. and B. Widrow (1989). The truck backer-upper: an example of self-learning in neural networks, *International Joint Conference on Neural Networks*, Washington, DC. Vol. II, pp.357-363.

[52] Nijmeyer, H. and A. van der Shaft (1990). *Nonlinear Dynamical Control Systems*, Springer-Verlag, New York.

[53] Palm, W.J. (1986). *Control Systems Engineering*, John Wiley.

[54] Parker, D. (1985). *Learning Logic, Technical Report TR-87*, Cambridge, MA: Center for Computational Research in Economics and Management Science, MIT.

[55] Pham, D.T. and X. Liu (1995). *Neural Networks for Identification, Prediction and Control*, Springer-Verlag, London.

[56] *Proceedings of the IEEE* (1990a). *Special issue on Neural Networks, I: Theory and Modeling*, Vol. 78, No. 9.

[57] *Proceedings of the IEEE* (1990b). *Special issue on Neural Networks, II: Analysis, Techniques & Applications*, Vol. 78, No. 10.

[58] Rochester, N., J.H. Holland, L.H. Haibt and W.I. Duda (1956). Tests on a Cell Assembly Theory of the Action of the Brain, Using a Large Digital Computer, *IRE Transactions on Information Theory*, Vol. IT-2:80-93. Reprinted in Anderson and Rosenfeld (1988) pp.68-80.

[59] Rosenblatt, F. (1958). The Perceptron: a Probabilistic Model for Information Storage and Organisation in the Brain, *Psychological Review*, Vol. 65, pp.386-405. Reprinted in Anderson and Rosenfeld (1988) pp.92-114.

[60] Rosenblatt, F. (1959). Two Theorems of Statistical Separability in the Perceptron, *Mechanization of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory, November 1958.* London: HM Stationery Office, pp.421-456.

[61] Rosenblatt, F. (1960). On the convergence of reinforcement procedures in simple perceptrons, *Cornell Aeronautical Laboratory Report VG-1196-G-4*, Buffalo, NY, February.

[62] Rosenblatt, F. (1962). *Principles of Neurodynamics.* New York: Spartan.

[63] Ruano, A.E.B., P.J. Fleming and D.I. Jones (1992). Connectionist approach to PID auto-tuning, *IEE Proceedings - Part D*, Vol.139, No.3, pp.279-285.

[64] Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986a). Learning Internal Representations by Error Propagation. In D.E. Rumelhart and J.L. McClelland, Eds., *Parallel Distributed Processing*, Vol. 1 Ch. 8, reprinted in Anderson and Rosenfeld (1988) pp.696-699.

[65] Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986b). Learning Representations by Back-propagating Error. *Nature*, 323:533-536, reprinted in Anderson and Rosenfeld (1988) pp.675-695.

[66] Schwarzenbach, J. and K.F. Gill (1984). *System Modelling and Control.* Second Edition. Edward Arnold.

[67] Shinskey, F.G. (1967). *Process control systems*, New York: McGraw-Hill.

[68] Shinskey, F.G. (1979). *Process control systems*, 2nd edition. New York: McGraw-Hill.

[69] Shinskey, F.G. (1995). Process control - where have we been, where are we going?, *S.A. Instrumentation & Control Buyer's Guide,* pp.96-101.

[70] Smith, O.J.M. (1958). Feedback Control Systems, McGraw-Hill, New York.

[71] Tank, D.W. and J.J. Hopfield (1987). Collective Computation in Neuronlike Circuits, *Scientific American*, 257:104-114.

[72] Tolat, V.V. and B. Widrow (1988). An Adaptive 'Broom Balancer' with Visual Inputs, *IEEE International Conference on Neural Networks,* San Diego, CA, Vol. II, pp.641-647.

[73] Valdebenito, C., D. Sbarbaro and J.P. Segovia (1995). Gaussian networks for pattern based adaptive control, *Proceedings of the 3rd European Control Conference*, Vol.2, , Rome, September, pp.1185-1190.

[74] Warwick, K., G.W. Irwin and K.J. Hunt (Eds.) (1992). *Neural networks for control and systems*, Peter Peregrinus Ltd., Stevenage, UK.

[75] Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* (Ph.D. thesis). Cambridge, MA: Harvard University Committee of Applied Mathematics.

[76] Widrow, B. (1986). Adaptive inverse control, *Proc. 2nd. International Federation of Automatic Control Workshop*, pp.1-5, Lund, Sweden, July, pp.1-3.

[77] Widrow, B. (1987). The Original Adaptive Neural Net Broom-balancer, *International Symposium on Circuits and Systems*, New York: IEEE, pp.351-357.

[78] Widrow, B. and M.E. Hoff Jr. (1960). Adaptive Switching Circuits, *IRE WESCON Convention Record*, part 4, pp.96-104. Reprinted in Anderson and Rosenfeld (1988) pp.126-134.

[79] Widrow, B. and M.A. Lehr (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation, *Proceedings of the IEEE*, Vol. 78, No. 9, pp.1415-1442.

[80] Widrow, B., P.E. Mantey, L.J. Griffiths and B.B. Goode (1967). Adaptive Antenna Systems, *Proceedings of the IEEE*, Vol. 55, pp.2143-2159.

[81] Yamanaka, K. and E. Shimemura (1987). Effects of mismatched Smith controller on stability in systems with time-delay, *Automatica*, **23**(6), pp.787-91.

[82] Yuwana, M. and D.E. Seborg (1982). A new method for on-line controller tuning, *A.I.Ch.E. Journal*, Vol.28, No.3, pp.434-440.

[83] Ziegler, J.G. and N.B. Nichols (1942). Optimum settings for automatic controllers, *Transactions of ASME*, Vol.64, No.11, pp.759-768.

[84] Ziegler, J.G. and N.B. Nichols (1943). Process lags in automatic control circuits, *Transactions of ASME*, Vol.65, pp.433-444.

Appendix:

GLOSSARY OF ACRONYMS AND ABBREVIATIONS

ADALINE Adaptive linear element

ANN Artificial Neural Network

FFT Fast Fourier Transform

FOPDT First-order-plus-dead-time

K_p Static gain of controller

K_{proc} Static gain of process or model

L_{proc} Dead time of process or model

MADALINE Multilayer adaptive linear element

MIMO Multiple-input, multiple output

PID Proportional-Integral-Derivative [controller]

RBF Radial basis function

RBFN Radial basis function [neural] network

SISO Single-input, single-output

sse Sum-squared error

std Standard deviation

T_d Derivative time for PID controller

T_i Integral time for PID controller

T_{proc} Time constant of first-order process or model