INVESTIGATION OF A MULTI-LAYER PERCEPTRON NETWORK TO MODEL AND CONTROL A NON-LINEAR SYSTEM

JOHN THOMAS EVANS, B.Eng.(HONS), A.M.I.E.E.

A thesis submitted in partial fulfilment of the
requirements of Liverpool John Moores University
for the degree of Doctor of Philosophy

September 1994

School of Electrical and Electronic Engineering
Liverpool John Moores University

INVESTIGATION OF A MULTI-LAYER PERCEPTRON NETWORK TO MODEL AND CONTROL A NON-LINEAR SYSTEM

Ph.D Thesis JOHN THOMAS EVANS September, 1994

ABSTRACT

This thesis describes the development and implementation of an on-line optimal predictive controller incorporating a neural network model of a non-linear process. The scheme is based on a Multi-Layer Perceptron neural network as a modelling tool for a real non-linear, dual tank, liquid level process. A neural network process model is developed and evaluated firstly in simulation studies and then subsequently on the real process. During the development of the network model, the ability of the network to predict the process output multiple time steps ahead was investigated. This led to investigations into a number of important aspects such as the network topology, training algorithms, period of network training, model validation and conditioning of the process data. Once the development of the neural network model had been achieved, it was included into a predictive control scheme where an on-line comparison with a conventional three term controller was undertaken. Improvements in process control performance that can be achieved in practice using a neural control scheme are illustrated. Additionally, an insight into the dynamics and stability of the neural control scheme was obtained in a novel application of linear system identification techniques. The research shows that a technique of conditioning the process data, called spread encoding, enabled a neural network to accurately emulate the real process using only process input information and this facilitated accurate multi-step-ahead predictive control to be performed.

ACKNOWLEDGEMENTS

Many thanks are due to my supervisors, Dave Williams and Barry Gomm (School of Electrical and Electronic Engineering, Liverpool John Moores University) and Paulo Lisboa (Department of Electrical Engineering and Electronics, Liverpool University), for their advice, guidance and support throughout the research project. Gratitude is also duly given to the following:

the Science and Engineering Research Council and Liverpool John Moores University for their financial support and;

the staff of the Control Systems Research Group and School of Electrical and Electronic Engineering, Liverpool John Moores University for their general help throughout the research duration.

Finally, many thanks to my Mother and Father for their encouragement and support.

CONTENTS

SYMBOLS

ADC	Analogue to Digital Converter
ai ; i=1, 2, ...	Class intervals
ANN	Artificial Neural Network
ARMAX	Auto-Regressive Moving Average eXogenous
ARX	Auto-Regressive eXogenous
C_1, C_2	Cross-sectional areas of process tanks
D/P	Differential Pressure
DAC	Digital to Analogue Converter
e(t)	Discrete white noise sequence
e_c	Error signal
e_f	Filtered error signal
f()	Non-linear function
G(s)	Process transfer function in s
GMV	Generalised Minimum Variance
GPC	Generalised Predictive Control
$\dot{h}_1$,$\dot{h}_2$	Derivatives of h_1 and h_2
h_1	Height of liquid in tank 1
h_{10}, h_{20}	Steady states at operating point for h_1 and h_2
h_2	Height of liquid in tank 2
I/P	Current to Pressure
IAE	Integral Absolute Error
IMC	Internal Model Control
ISE	Integral Square Error
j	$\sqrt{-1}$

J()	Cost function
k, τ_d	Process dead time
K_c	Proportional gain
K_p	Static gain
K_v	Process valve gain
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
N	Number of data values in correlation test
N1, N2	Prediction horizons
n_a, n_b, n_k	Model orders
NARMAX	Non-linearAuto-Regressive Moving Average eXogenous
NARX	Non-linear Auto-Regressive eXogenous
NU	Control horizon
O	Output vector from output layer
P/I	Pressure to Current
PI	Proportional plus Integral
PID	Proportional plus Integral plus Derivative
PRBS	Pseudo-Random Binary Sequence
psi	Pounds per Square Inch
q_0	Liquid flow rate into tank one
q_1	Liquid flow rate into tank two
q_2	Liquid flow rate out of tank 2
R_1, R_2	Restrictances of tanks 1 and 2
RAS	Random Amplitude Signal
s	Laplacian operator
SEDM	Spread Encoded Data Mapped

SNDM	Single Node Data Mapped
SNR	Signal-to-Noise Ratio
T_d	Derivative action time
T_i	Integral action time
Ts	Sampling time
u	Process input
u(n)	Digital controller output
u(t)	Sampled process input
U_s	Controller bias signal
V	Output vector from hidden layer
W	Weight vector
w_0	Bias weight
w_{ij}	Weight connections between input and hidden layer
w_{jl}	Weight connections between hidden and output layer
X	Input vector
$\bar{x}$	Sample mean
X_{max}	Maximum data value
X_{min}	Minimum data value
X_s	Scaled data value
y'(t)	Predicted process output
y(t)	Sampled process output
Y_r	Vector of desired output
z^{-1}	Backward shift operator
$\hat{\theta}$	Parameter estimate vector
θ	Parameter vector
ψ	Excitation level
β	Filter coefficient

η	Learning rate coefficient
α	Momentum coefficient
ϕ	Residual
σ	Sample standard deviation
τ	Time constant
λ	Weighting factor
$\delta(t)$	Dirac delta function
$\varepsilon(t)$	Prediction error
$\Phi(t)$	Regression vector
σ_e	Process noise standard deviation
σ_{h2}	Process output standard deviation
τ_i ; i=1, 2, ...	Lag

CHAPTER 1

INTRODUCTION

1.1 PROJECT BACKGROUND

In many present day industries there is a requirement for the optimal control of processes for financial, economical and political reasons, which places a very high requirement on the availability of models describing the dynamic behaviour of processes. Process models play an important part since the design of efficient process controllers relies heavily on the accuracy of the models, and in the majority of cases, without an adequate model of the process to be controlled, the synthesis of a control algorithm is not possible (FASOL and JORGL, 1980).

In many industries, such as the aerospace industry, processes are usually well defined and hence the construction of accurate process models to enable design of controllers is simplified. This research project concentrates on chemical, biotechnological and food manufacturing industries where it is typical to have complex processes that exhibit non-linearities, time variations, disturbances and uncertainties. Under these conditions the development of accurate dynamic process models from first principles is a considerably difficult, time consuming and costly exercise. The resulting models may also be very complex thus, rendering the subsequent task of controller design an onerous one.

The following sections, 1.2 and 1.3, give a general review of process modelling and different control strategies available to the process engineer.

1.2 REVIEW OF PROCESS MODELS

Mathematical models of processes can be developed in basically three different ways, these being either purely theoretical, purely empirical or a combination of both theoretical and empirical methods. When developing a model via the theoretical route, the development of the model will be dependent on natural and physical laws governing the process, whereas the empirical route will be based on experimental studies.

Both theoretical and empirical modelling methods have their advantages and disadvantages and in many cases a process model will be constructed using the combination of the two methods. If the process under investigation does not already exist, then empirical modelling is not possible and theoretical modelling is the only way forward to investigate the performance of *a priori* control structures. This is also the only way forward if experiments on an existing process cannot be carried out, for various reasons such as safety. A disadvantage of theoretical modelling when applied to complex processes, such as those found in the chemical industry, is that it usually requires an expert with an in depth knowledge of the physical phenomena occurring in the process in order to obtain a realistic model. For experimental process investigations, the knowledge is also necessary but not that important (FASOL and JORGL, 1980). One of the main disadvantages with the empirical approach is that the resulting models are generally in the form of input-output equations, and hence this results in a 'black box' model of the process which is difficult to relate back to the original process.

The majority of systems are frequently modelled under the assumption that the process can either be assumed linear over the operational region of interest or at

least linearised about the operating point, the resulting process models are then termed linear models. However, in certain circumstances it is not realistic to assume the process is linear over its entire operating range and it is advantageous to obtain a non-linear model description. In this section a review of the most commonly encountered linear and non-linear models along with any associated problems is given.

1.2.1 Linear process models

When theoretical modelling is performed on a linear or linearised system, the resulting dynamic equations will be either ordinary or partial differential equations which will usually reduce to algebraic and ordinary differential equations when only steady state cases are of interest.

When process modelling is performed by experimentation, the resulting linear model or models will be either parametric or nonparametric. The most common parametric models considered are step response models and time series analysis models, which result in models of the auto-regressive integrated moving average, ARIMA, type, and other similar linear model structures, such as the ARX and ARMAX.

For any of the empirical methods there are basically three steps to be taken in order to obtain a process model:

i) Apply a suitable test signal to the input of the process and collect the input-output response data;
ii) Decide on a suitable form of the model in order to represent the process;
iii) Determine the parameters of the postulated model by fitting the model to the

input-output data.

The step response method is a very simple technique but suffers from the problem that if the process is non-linear then the size of the step input should be kept as small as possible so as to maintain an accurate linear model. A disadvantage associated with the other techniques is that of model order selection, which for the realistic case of the process measurements contaminated with noise, can become complex. However, model order selection techniques are available to aid in the design (SODERSTROM and STOICA, 1989).

The popular nonparametric process models are described by the process weighting function or frequency response. As with the step response method, obtaining a model from the frequency response of a process is largely dependent on the process being linear over the range of interest, if this is not the case then this method will result in an inaccurate process model. The weighting function of a process can be determined if the auto-correlation function of the input signal and the cross-correlation function of the input and output signal are known. However, considerable problems occur in interpreting the cross-correlation function in terms of a process weighting function when there are non-linearities present (GODFREY, 1980).

It is evident from the above that if the process under investigation contains non-linearities then each of the techniques is only capable of modelling the process over a small linearised region. In certain cases, however, the process non-linearities are only mild and hence, a linear model can adequately characterise the process. The advantage of this is that the great wealth of linear systems theory, which is well understood, can be used to design a suitable controller. When the process non-

linearities are such that a linear model is only accurate over a small region of the total operating range, then linear models can still be used to represent the process over the whole range by constructing a number of linear models throughout the operating conditions. An alternative approach is recursive identification where a linear model of the process is estimated at each sample period. This technique is usually coupled to a self tuning control structure, discussed in section 1.3 .

Although many processes are modelled on the above techniques it is, however, a fact that non-linear process models are a more suitable alternative when constructing models of real processes.

1.2.2 Non-linear process models

The main problem with non-linear systems theory is that there are no general methods for the analysis and synthesis of non-linear processes. Unlike linear processes, the analysis of many non-linear systems results in complex mathematical models that bear little or no resemblance to each other and hence, this inhibits a widespread methodology for control system design.

As with linear process models, a theoretical approach can be taken to obtain a non-linear model of a process and if this is the case then the experimental approach may not be necessary. However, a number of models have been proposed that use input-output data from the process to construct the model.

The analysis of a wide range of non-linear processes can be based on the use of functional series. The Volterra series (VOLTERRA, 1931) can be used to calculate the process output of a non-linear process. The analysis results in a nonparametric model and suffers from the problem that the solution is extremely difficult when

the input to the process is assumed stochastic (BILLINGS, 1980). A discrete Volterra series was developed from the continuous series (ALPER, 1965) and used to analyse non-linear sampled data systems. Both the models proposed by Volterra and Alper are nonparametric process models and have the disadvantage of a large number of kernels to be estimated which in turn require a very large number of input-output process data to be collected from the process (BILLINGS, 1980). A parametric model has been derived (HABER and KEVICZKY, 1974) which is capable of approximating a discrete second order Volterra series. The advantages of the model are that it has a finite number of parameters and is also linear in the unknown parameters. From the parametric Volterra model a number of other models, suitable for the modelling of non-linear processes, have been devised, such as the generalised Hammerstein and the simple Hammerstein model (ISERMANN et al., 1992). The simple Hammerstein model consists of a linear transfer function in series with a non-linearity and both Hammerstein models are of the parametric type.

Another class of important non-linear process models is based on non-linear differential equations and results in a model under the name of the non-linear differential equation, (NDE), model. This model description is of importance since non-linear differential equations often result from natural physical laws governing the process. Another form of non-linear model structure is termed prediction error models for non-linear stochastic processes (BILLINGS and LEONARITIS, 1985). These models are referred to as non-linear auto-regressive moving average exogenous, NARMAX, models and are a generalisation of the ARMAX model used in linear system identification. The main problem with these non-linear process models is that the order of the model has to be chosen correctly and this is not always an easy task. Also, in the realistic situation of noise disturbing the

process measurements, an over parameterised model will not give a better description of the process but will give a better fit to the process noise.

1.3 REVIEW OF PROCESS CONTROLLERS

The development of linear controllers for linear systems has been established for many years and a wide range of designs are available. By far the most widely used in practice is the three term, PID, controller. Other controllers devised over the years include (OGATA, 1970):

(i) The pole-assignment controller
(ii) The model reference controller
(iii) The cancellation controller
(iv) The deadbeat controller

The main disadvantages with all the above mentioned controllers, when applied to real processes, is that they are not optimal, and their accuracy is dependent on the process being linear and time invariant. In reality the majority of processes are non-linear and the process parameters may vary with time. In an attempt to overcome the disadvantages, the area of self tuning control was introduced. The above mentioned controllers can be used in a self tuning structure and a number of other typical self tuning controllers include (WELLSTEAD and ZARROP, 1991)

(i) The minimum variance controller
(ii) The generalised predictive controller
(iii) The linear quadratic Gaussian controller

In a self tuning control scheme the controller will adapt according to the changes in the process and environment in an attempt to produce an optimal or a desired control. The majority of self tuning schemes still work on the principle of using a linear model of the process, since this enables the great wealth of linear systems theory to be applied in the design. A linear model, representing the region in which the process is operating, is usually estimated at each sample time and then the controller is updated based on the required control. Disadvantages of the adaptive control scheme are that it usually requires a considerable amount of on-line computations, the accuracy of the model is also constrained within certain bounds of the operating point and if the process is required to be operated at another set point a new model must be estimated. Also, if the process returns to an original operating point the model at that point must be estimated again.

A number of non-linear controller designs have been proposed that use the information from the non-linear model representing the process in order to select the controller parameters. However, most of the designs have been based on a specific process and the issue of excessive computations to be performed within the sample time of the process can cause problems.

1.4 MOTIVATION FOR AN ANN APPROACH

It is evident from the above that the modelling and control of non-linear processes has proven difficult and, consequently, although the subject of specific research projects, the application of non-linear controllers to real plant is rare. It has been shown (NARENDRA and PARTHASARATHY, 1989; ELSELY and SHENG, 1991; NGUYEN and WIDROW, 1990) that neural networks may be used for both process modelling and control of non-linear systems. They are naturally suited to

describe the behaviour of systems containing non-linearities, and indeed have been shown to utilise their own non-linear characteristics to provide robustness against noisy data. Since the modelling of processes is a very important step in the understanding of plant operation for process operators, plant designers and control engineers as well as in the design of controllers, a great deal of interest towards artificial neural networks for process modelling and control has been shown.

1.4.1 Types of Artificial Neural Networks

A large number of artificial neural networks have been proposed over the past few years and only a list of the most popular types will be mentioned. Many of the networks not mentioned are found to be closely related to these popular types.

Artificial neural networks can be classed as either supervised or unsupervised networks, and this refers to whether the network is presented with target output values during training or if the network is self organising respectively. The basic difference between the different types of networks is in the architecture and the algorithms used for their training. Some of the characteristics of the different network architectures are that processing units within a layer can be either fully interconnected or not interconnected at all, the networks can have a single layer or a multi-layer of processing units and the flow of data can be from the input to the output of the network only or from a processing unit back to itself, in which case the network is termed recurrent. Of the unsupervised neural networks the most popular are:

(i) Kohonen network (KOHONEN, 1972),

(ii) Adaptive resonance theory networks ART1 and ART2 (CARPENTER and GROSSBERG, 1985)

(iii) Boltzmann machine neural networks (HINTON and SEJNOWSKI, 1983)

(iv) Bi-directional associative memory (BAM) networks (KOSKO, 1987)

The Kohonen and ART2 networks are used with continuous valued data whilst the ART1, Boltzmann and BAM networks are used with binary valued data.

The most widely accepted supervised neural networks are:

(i) The neocognitron (FUKUSHIMA et al., 1983)

(ii) Multi-layer perceptron (RUMELHART et al., 1986)

(iii) Radial basis function (RBF) network (MOODY and DARKEN, 1989)

(iv) Cerebellar model articulation controller (CMAC) (ALBUS, 1975)

(v) Discrete and continuous Hopfield networks (HOPFIELD, 1982; HOPFIELD and TANK, 1985)

The Hopfield network differs from the other supervised networks in that it is an auto associative network.

The most important choice for the neural network used in this research was the ability of the network to represent non-linear functions and this influenced the network chosen to be the multi-layer perceptron. It has been proven that the multi-layer perceptron can approximate any non-linear function (HORNIK et al., 1989; FUNAHASHI, 1989; CYBENKO, 1989) and also, being a supervised network, it is ideally suited to the task of modelling from input-output process data. A thorough description of this type of neural network along with its associated training algorithm is presented in Chapter 2.

1.5 PROJECT DESCRIPTION

1.5.1 Problem Statement

From the review in section 1.2, it is evident that linear process models have their limitations when used to model real processes and it is advantageous to use a non-linear model. However, the complexity behind the majority of the non-linear models and the problems in choosing the correct structure can cause difficulties when constructing a process model. Hence, the research project undertaken was to investigate an alternative method of obtaining a non-linear process model and then using the model for control purposes. The project was split into two sections, firstly to investigate the ability of artificial neural networks to model a non-linear process over a wide operational region, using sampled input-output data obtained from exciting the process. The second part was then to implement the resulting neural network model into a suitable control strategy and investigate control of the non-linear process using the artificial neural network model based control structure. The neural network approach was taken in order to examine its feasibility and benefits when applied to non-linear process modelling and control.

1.5.2 Aims

(i) The overall aim was to develop an on-line artificial neural network model based control strategy that could be used to accurately control a non-linear process when subjected to set point changes covering a wide operational region. The control scheme was to be implemented on a dual tank liquid level process which had characteristics similar to many industrial processes.

(ii) To develop a methodology for obtaining an accurate process model using a neural network. This will enable the accurate modelling of other processes by following the same procedures for determining the optimum network topology, training time, etc.

(iii) To investigate different types of control structures where a neural network process model may be used, and ultimately choose a structure that will exploit the capabilities of the neural network model.

1.6 OVERVIEW OF THESIS

An introduction to the artificial neural network used throughout the research is presented in Chapter 2 along with a number of algorithms available for training the network. The selection of the network topology and length of training is also considered, and finally a novel method of conditioning the process data presented to the neural network along with a standard data conditioning method is discussed. Conditioning the process data is investigated in an attempt to achieve an accurate process model that could operate independently from the process whilst still accurately predicting process behaviour. Chapter 3 describes the process used during the research and practical aspects of obtaining a neural network process model are discussed. The liquid level process was used since it is a practical test bed on which to investigate the proposed techniques before applying them to more complex processes. An assumed mathematical model of the process was investigated initially and then a mathematical model of the real laboratory process was obtained. All the initial process modelling studies were investigated in simulation for cases of noise free process data and then for the more realistic case of simulated process noise included in the process measurements. The Chapter

presents a comparison between the proposed data conditioning method and the standard method of encoding the process data to ascertain if the proposed technique was a viable alternative. The insights gained in simulation studies are then applied to model the real laboratory scale dual tank liquid level process in Chapter 4 and a correlation based method is used to validate the resulting neural network model along with cross-validation on a number of test signals not used during network training.

Once the development of an accurate neural network process model was considered to be achieved, investigations in including the model into a control strategy were undertaken. Chapter 5 describes a number of control strategies that incorporate a process model as an integral part and the inclusion of a neural network model into the control strategies is described. Simulation results of controlling the process using one of the neural network based strategies are then presented. The investigations of on-line neural network control of the process are then presented in Chapter 6.

Finally, conclusions obtained from the research and suggestions for further work, are presented in Chapter 7.

CHAPTER 2

THE MLP NEURAL NETWORK AND RELATED TOPICS

2.1 INTRODUCTION

The first objective of the research was to investigate and develop techniques to model a non-linear process over a wide non-linear operating region using a neural network. Such a neural network model has a wide application for simulation studies of process performance, and more importantly can be used to improve control strategies that utilise models for predicting future controlled process responses. The Multi-Layered Perceptron was studied throughout this research because this network is suitable to model non-linear relationships and is straight forward to implement and train. This Chapter describes the structure and the parameters associated with this type of network and various algorithms that can be used for training the network. Once the neural network topology has been chosen, the next stage in the development of a neural network model is to train the network. Research has shown that it is possible for these networks to be 'over trained', hence a method has been developed to determine when a network has had an adequate amount of training.

In many real processes, dynamic relationships exist between process inputs and outputs. The MLP neural network is a static network and consequently the characteristics of the process dynamics need to be introduced into the network.

A novel method of conditioning the data presented to the neural network was considered, which involved taking a single data value and mapping it across a number of nodes. The coding of the data is described and the accuracy of neural

networks trained using this method is compared to neural networks trained using the standard method widely used in the field of neural networks.

2.2 THE MULTI-LAYER PERCEPTRON (MLP) NEURAL NETWORK

The MLP neural network has been used for many years successfully in the area of pattern recognition (GORMAN et al., 1988; SEJNOWSKI and ROSENBERG, 1987), and recently has been used for the purpose of modelling and controlling non-linear processes (NARENDRA and PARTHASARATHY, 1989; ELSELY and SHENG, 1991; NGUYEN and WIDROW, 1990). Interest in the MLP network developed because the network is well suited to 'learning' non-linear relationships when little *a priori* knowledge is available. It has been proven that a network with just one hidden layer can approximate any non-linear function with arbitrary accuracy provided enough hidden processing units are used (HORNIK et al., 1989; FUNAHASHI, 1989; CYBENKO, 1989). The basic processing unit of the MLP neural network is based on a model proposed by McCulloch and Pitts, (McCULLOCH and PITTS, 1943), and is illustrated in Figure 2.1.

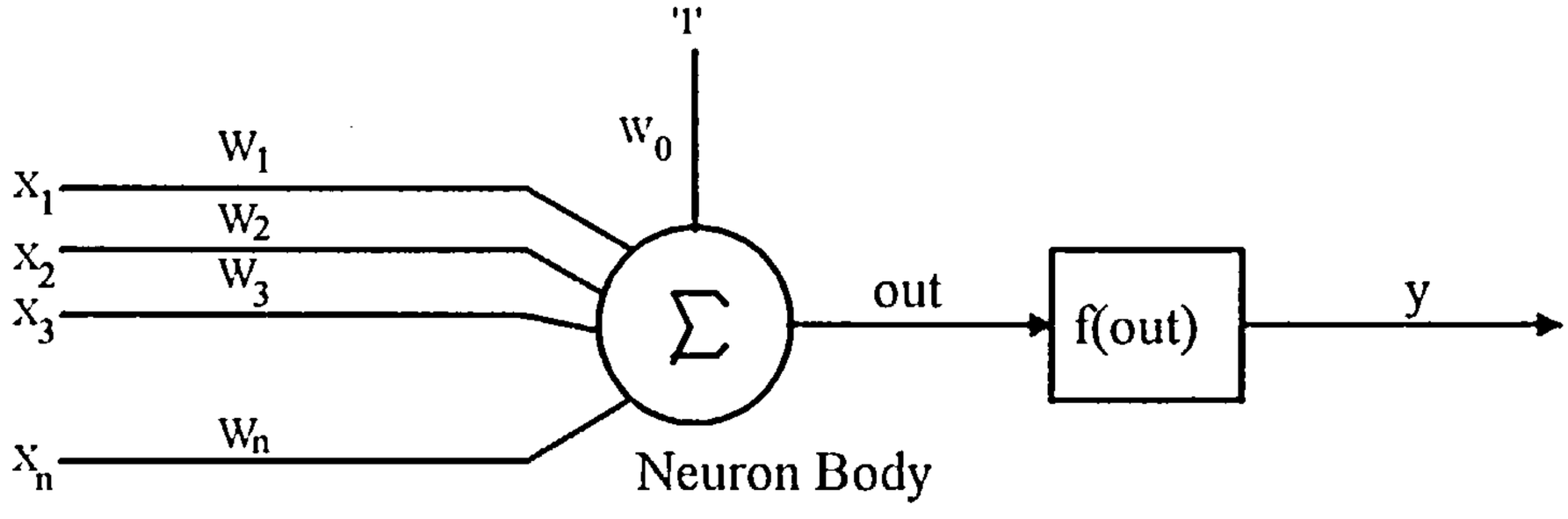


Figure 2.1 McCulloch-Pitts neuron model

Each input X_i, i=1..n, to the neuron body is multiplied by a corresponding weighting value W_i, i=1..n, and the products are summed:

$$Out = \sum_{i=1}^{n} X_i W_i + W_0 \qquad ...(2.1)$$

Equation 2.1 has an extra weight W_0, and this is the weight associated with the bias term in the processing unit, which is used to set the reference level of each processing unit, and it has also been shown (WRAY and GREEN, 1991) that the MLP network without bias is unable to approximate certain functions. The output of the unit, y, is obtained by passing the result of equation 2.1 through a non-linear activation function, as described by equation 2.2:

$$y = f(Out) \qquad ...(2.2)$$

f() can be any differentiable continuous function (CYBENKO, 1989), the most commonly used being the sigmoid illustrated in Figure 2.2, which also illustrates other commonly used functions.

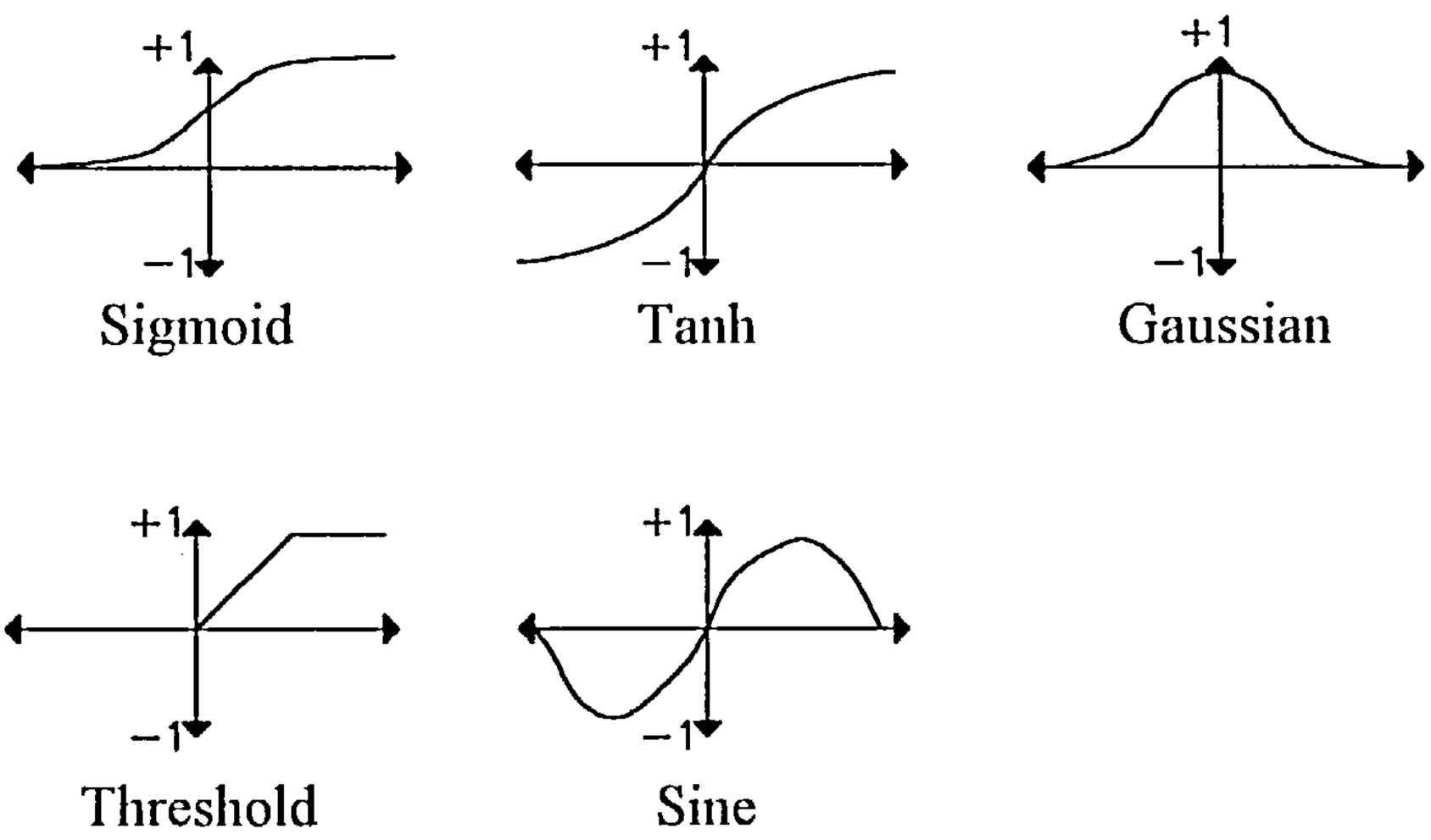


Figure 2.2 Common activation functions

A single processing unit on its own has limited power, and the real usefulness comes when processing units are connected together in certain configurations. The configuration of the MLP neural network with one hidden layer is shown in Figure 2.3.

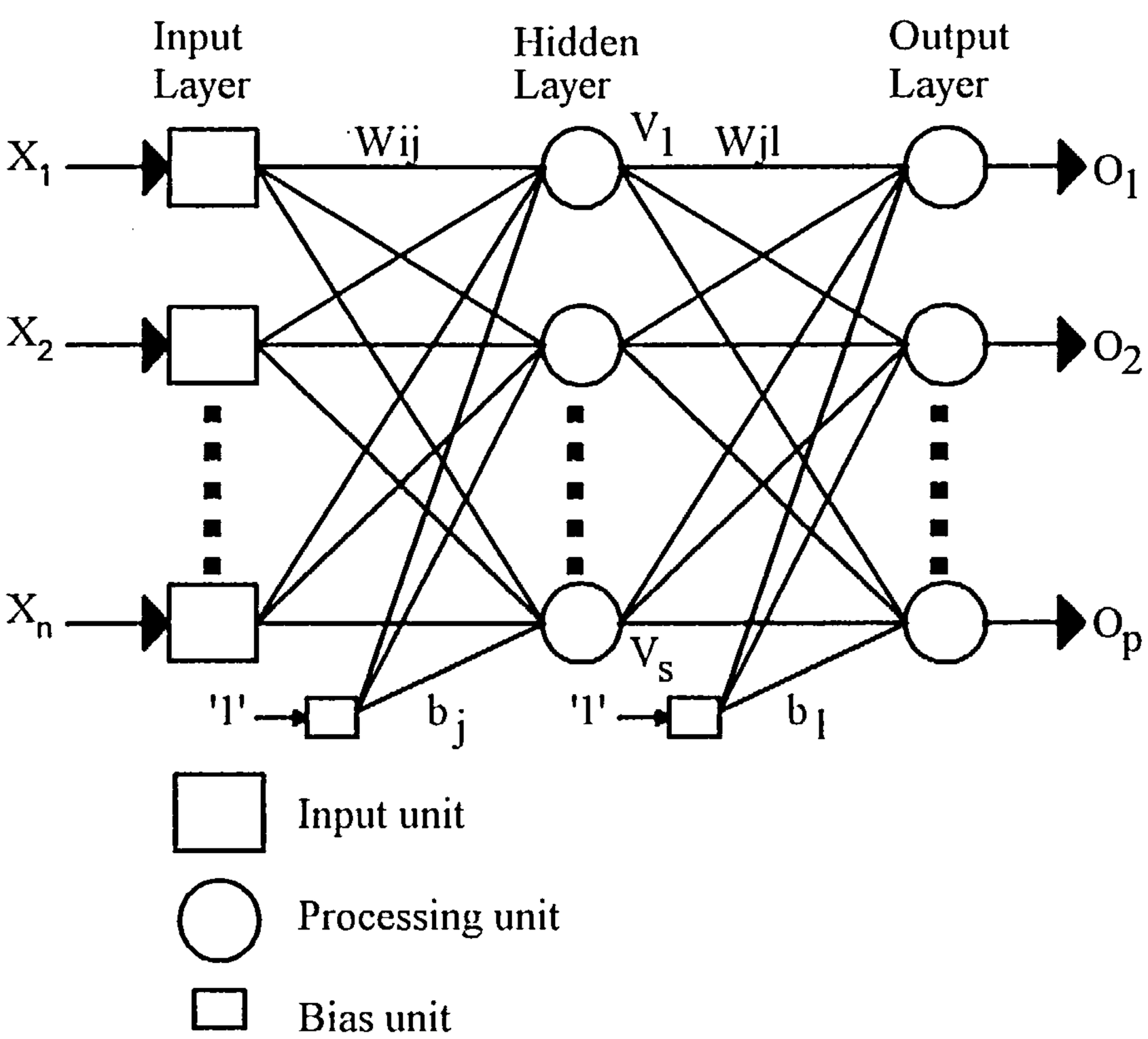


Figure 2.3 Multi-layer perceptron with one hidden layer

The MLP network consists of at least three layers of nodes, normally referred to as an input layer, one or more hidden layers and an output layer. The input layer is passive in nature and performs no processing on the data it receives, it serves to pass the input data to the first hidden layer via the connection weights. The hidden and the output layers are both active, and each processing unit in these layers performs the operations described by equations 2.1 and 2.2. The outputs of each unit in the hidden layers are passed onto the next layer, through to the output layer.

It is the presence of the hidden layer(s) and the use of non-linear activation functions that permits algorithms used to train the MLP neural network (discussed in section 2.3) to be able to model highly non-linear functions. The number of input and output units used in a MLP neural network is generally defined by the problem being considered. Choices have to be made on how many hidden layers and processing units in these layers should be used. To date there is no established theory to determine these choices, and they are usually determined empirically.

The MLP is also known as a feed forward network since data is presented to the network at the input layer, and propagates forward through the hidden layers to the output of the network, and no data is fed back from one processing unit to another processing unit in the previous layer or to itself. Once the network structure has been defined, the next objective is to train the MLP network.

2.3 TRAINING ALGORITHMS

When training the MLP neural network, the objective of the training algorithm is to adjust the networks weights so as to minimise a pre-specified cost function. In recent years by far the most popular training algorithm is the 'back propagation', which has certain disadvantages associated with it. To overcome these disadvantages, modifications to this and other algorithms, for training layered networks, have been developed. The back propagation algorithm was used throughout this project due to its proven track record. The back propagation algorithm is discussed in detail below, and modifications to the algorithm along with other training algorithms are discussed in section 2.3.2.

The development of back propagation is due to Werbos (1974), Parker (1985), Rumelhart et al., (1986). The algorithm is based on gradient descent and is described with reference to Figure 2.3. The network input is a vector X with dimension n, the output of the hidden layer is stored in a vector V with dimension s, and the network output is a vector O with dimension p. The training data is represented as follows:

$$X^i(k) = [x_1(k), x_2(k), x_3(k), ..., x_n(k)] \qquad ...(2.3)$$

$$Y_r^i(k) = [y_{r1}(k), y_{r2}(k), y_{r3}(k), ..., y_{rp}(k)] \qquad ...(2.4)$$

Where X(k) is the sample input vector at time k, and $Y_r(k)$ is the required network output associated with X(k), i is used to index which training sample from the total training samples is being presented to the network. The algorithm can be described in three stages the first of these is to initialise all the network weights randomly to 'small' values. The second stage involves passing a training sample through the network and producing a corresponding network output vector O, this vector is computed using equations 2.6, and 2.7, where it is assumed the activation function for the hidden and output processing units is a sigmoid, equation 2.5

$$f(x) = \frac{1}{1+e^{-x}} \qquad ...(2.5)$$

$$v_j(k) = f\left(\sum_{i=1}^{n} w_{ij} x_i(k) + b_j\right) \quad j = 1,2,...,s \qquad ...(2.6)$$

$$o_l(k) = f\left(\sum_{j=1}^{s} w_{jl} v_j(k) + b_l \right) \quad l = 1,2,\ldots,p \qquad \ldots(2.7)$$

where w_{ij} and w_{jl} are the weight connections between the input and hidden layer, and the hidden and output layer respectively; b_j and b_l represent the bias weights connected to a constant unit input and associated with the hidden and the output layer processing units respectively. The third and final stage of the algorithm involves adjusting the weights in the network. This is achieved by comparing the network output vector, O, with the required output vector, Y_r, to produce an error. This error is back-propagated through the network to determine the adjustments to be made to the network weights using the following gradient descent equations 2.8 and 2.9. These apply to the weight updates between the hidden and the output layer, and equations 2.10 and 2.11 to update the weights between the input and the hidden layer.

$$w_{jl}(k+1) = w_{jl}(k) + \eta \delta_l o_l \qquad \ldots(2.8)$$

where

$$\delta_l = (y_{rl} - o_l) * \bar{f}\left(\sum_{j=1}^{s} w_{jl} v_j + b_l \right) \qquad \ldots(2.9)$$

$$w_{ij}(k+1) = w_{ij}(k) + \eta \delta_j o_l \qquad \ldots(2.10)$$

where

$$\delta_j = \bar{f}\left(\sum_{i=1}^{n} w_{ij} x_j + b_j \right) * \sum_{l=1}^{p} \delta_l w_{jl} \qquad \ldots(2.11)$$

$\bar{f}$ is the first derivative of the sigmoidal function, and is easily shown to be given by

$$\bar{f}(v) = f(v) * [1 - f(v)] \qquad ...(2.12)$$

The bias weights are also updated in the same manner. η is an adjustable parameter, known as the learning rate, which affects the size of the step taken in the search direction. The choice of a small step size will result in a network taking a long time to train whereas too large a step size may result in oscillations about the error surface of a minimum. When the first training example has been passed through the network and the network weights have been adjusted, another training sample is presented to the network, and the weights adjusted according to the output error. This sequence continues for the whole of the training data, at the end of which the mean square error (MSE) between the network predictions and the required predictions can be computed over the training set. If the resulting MSE is acceptable then training of the network can be terminated, otherwise the complete data set is presented to the network again as above.

It is well known that gradient descent methods are inefficient when the search approaches a minimum, since the back propagation algorithm assumes that the error surface around the learning rate is linear. At points of significant curvature the algorithm can display divagating behaviour. To prevent this from happening the learning rate should be set to a small value, resulting in the network taking a long time to train. In order to overcome this problem an adjustable parameter known as momentum is added to equations 2.8 and 2.10, resulting in equations 2.13 and 2.14

$$w_{jl}(k+1) = w_{jl}(k) + \eta \delta_i o_l + \alpha (w_{jl}(k) - w_{jl}(k-1)) \qquad ...(2.13)$$

$$w_{ij}(k+1) = w_{ij}(k) + \eta \delta_j o_l + \alpha (w_{ij}(k) - w_{ij}(k-1)) \qquad ...(2.14)$$

α is the momentum term, and this helps to improve the convergence speed.

2.3.1 Alternative training algorithms

The computations involved in the back propagation algorithm are simple and the algorithm is parallel in structure. The main draw back of the algorithm is its slow convergence properties and in some cases it can get trapped in local minima, even with the addition of the momentum term. Attempts have been made to speed up the back propagation algorithm (FAHLMAN, 1988; SAMAD, 1989), which involved modifying the derivative, and adding some proportion of the error to the activation value of each processing unit.

Other algorithms have been developed for training multi-layered networks in order to overcome the disadvantages associated with back propagation. Bremermann and Anderson (1989) developed an algorithm referred to as chemotaxis. This algorithm adjusts the weights in the network by adding Gaussian distributed random values to the existing weight values. If the error at the output of the network is reduced when the random values are added, then the new weights are retained, otherwise, another set of random values is generated and added to the original weights and the test repeated. The addition of these Gaussian distributed random values is continued until the network prediction errors over the training data set are at an acceptable level. However Bremermann and Anderson concluded from their studies that although the chemotaxis algorithm gives much better convergence properties for large networks, it performed less favourably than the back propagation algorithm for small sized networks. Another algorithm, the directed random search (MATYAS, 1965; BABA, 1977) can also be used for training the standard feed forward network. It operates on the same principles as the chemotaxis algorithm by

adding randomly generated values to existing weights and monitoring the new overall network error.

A parallel recursive prediction error algorithm was developed by Chen et al., (1990) which updates the weights in the network in a parallel manner, similar to the back propagation algorithm. It has been shown to have superior convergence properties to that of the back propagation algorithm although this was at the expense of a more computationally complex algorithm.

2.4 DYNAMIC STRUCTURES AND TRAINING FOR THE MLP NEURAL NETWORK

The MLP network is a static network, which simply performs a non-linear mapping from its input layer through to the output layer. Hence, if dynamics are not introduced into the network the ability to accurately model a process will be impeded. One method of representing the dynamic characteristics suggested by Terzuolo et al., (1969) and implemented by Willis et al., (1991) is that of using a first order lowpass filter connected at the output of each processing unit in the neural network, in order to change the output of the unit. The change in the processing units output is governed by the filter time constants, of which suitable values cannot be chosen prior to network training. The method used by Willis et al., is to estimate the time constants in a similar manner to that of the network weights as training proceeds. This method is an attractive one since it enables process knowledge to be incorporated directly into the network structure. However, the inclusion of the filter does not allow the networks to be trained using the standard back propagation algorithm although alternatives such as the chemotaxis algorithm are suitable.

Another method of introducing dynamics into the network is to use recurrent networks described by Narendra and Parthasarathy (1988). By far the most popular method of introducing network dynamics is to use past input and output data from a process as the inputs to the neural network. This enables the neural network to be trained using the standard back propagation algorithm, and was chosen as the means of introducing network dynamics. Using past input and output data results in a Non-linear AutoRegressive eXogenous (NARX) model configuration for the neural network, which is described in section 2.4.1. The choice of how many past inputs and past outputs to be used can seriously affect the performance of the resulting neural network model. In an attempt to ovecome this a method for determining a suitable amount is described in section 2.5.1.

2.4.1 Non-linear AutoRegressive eXogenous (NARX) model structure

The NARX model is a subset of the Non-linear AutoRegressive Moving Average eXogenous (NARMAX) model (BILLINGS and LEONARITIS, 1985), which in turn is a generalisation of the AutoRegressive Moving Average eXogenous (ARMAX) model for linear systems. The NARMAX and the NARX models are described mathematically by equations 2.15 and 2.16 respectively.

$$y(t) = F\left(y(t-1),..,y(t-n_a),u(t-k),..,u(t-k-n_b),e(t-1),..,e(t-n_e)\right)+e(t) \quad ...(2.15)$$

$$y(t) = F\left(y(t-1),...,y(t-n_a),u(t-k),...,u(t-k-n_b)\right)+e(t) \quad .....(2.16)$$

Where:

y(t) is the sampled process output data,

u(t) is the sampled process input data,

e(t) is a Gaussian distributed white noise sequence,

n_a, and n_b is the number of past output and input data respectively used,

n_e is the number of past noise values,

k is the process dead time,

F() is some unknown non-linear function.

The NARX model structure was chosen over the NARMAX structure since it does not depend on past values of the noise sequence and is thus easier to implement. Equation 2.16 indicates that past process input and output values are used in some unknown non-linear function to estimate the process output at the next sample time, the objective here being to use the MLP neural network to approximate the unknown non-linear function.

2.4.2 Representation of the MLP network in the NARX structure

The implementation of the MLP neural network into the NARX configuration is illustrated in Figure 2.4 where n_a past outputs and n_b past inputs are used as the inputs to the neural network and the output of the neural network is the predicted process output at the next sample time; z^{-p} is the backward shift operator, where for example $z^{-p}u(t) = u(t-p)$. Figure 2.4 also indicates how the neural network can be implemented into two different structures. When the switch is placed in position 1, then the neural network will be configured as a predictor, and similarly in position 2 the neural network is configured as a model. It is possible to develop neural network process models in either configuration, and the relative methods of each have been investigated for incorporation in an on-line control strategy. The two configurations can be defined mathematically by equations 2.17 and 2.18.

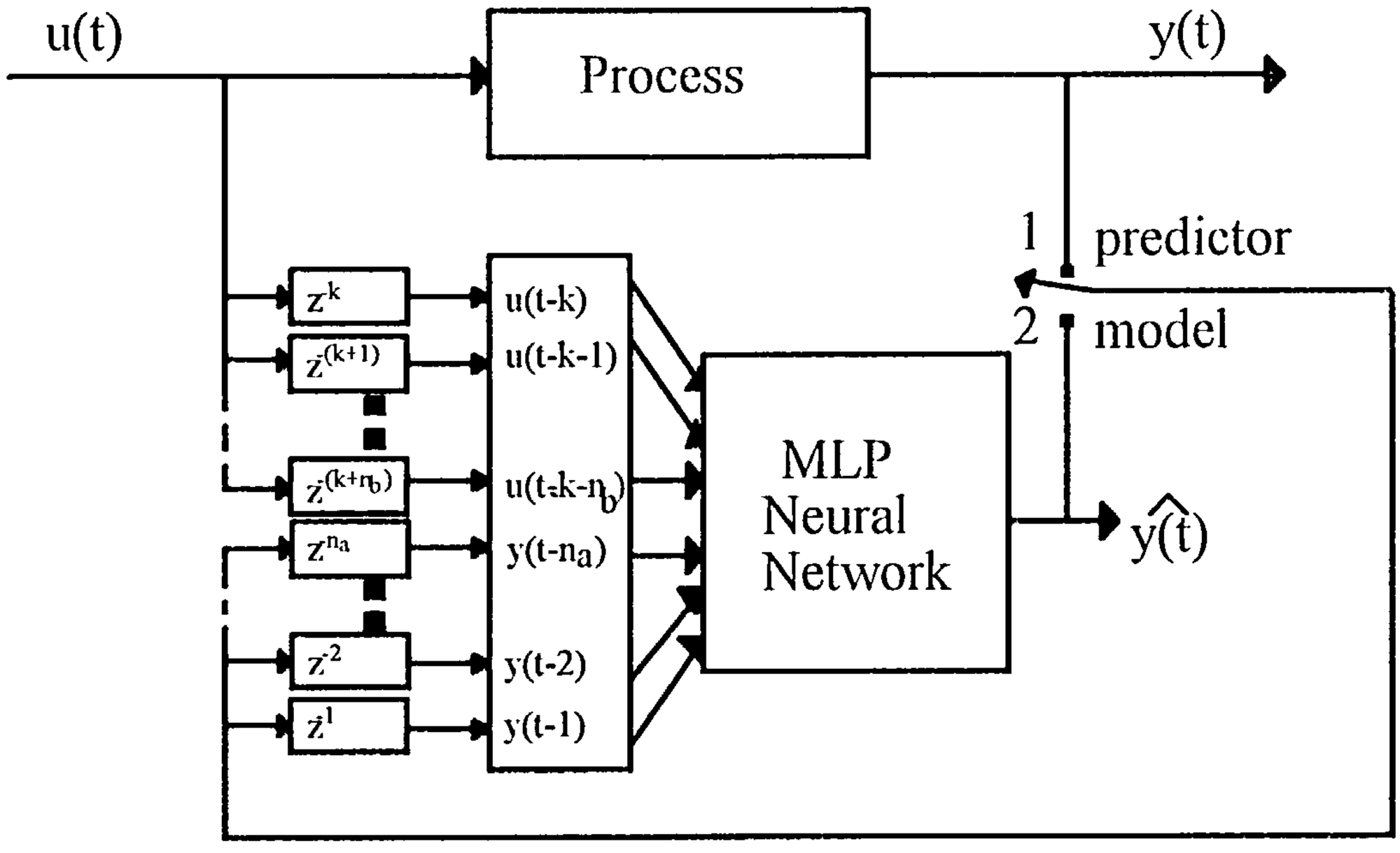


Figure 2.4 Multi-layer perceptron in NARX configuration

$$\hat{y}(t) = F\left(y(t-1),\ldots,y(t-n_a),u(t-k),\ldots,u(t-k-n_b)\right) \qquad \ldots(2.17)$$

$$\hat{y}(t) = F\left(\hat{y}(t-1),\ldots,\hat{y}(t-n_a),u(t-k),\ldots,u(t-k-n_b)\right) \qquad \ldots(2.18)$$

The difference between the two configurations is in the use of feedback from the output of the network to its input for the model, and no feedback for the predictor. Both the predictor and model configurations were investigated by Narendra and Parthasarathy (1990) where they were referred to as the series parallel model for the case of the predictor structure and the parallel model for the structure represented by model. In their work they also considered a number of other

structures which were combinations of linear and non-linear functions of the input and output of the process. Equation 2.17 incorporates all the models suggested by Narendra and Parthasarathy. These two different structures lead to two different methods for training the neural network to emulate the non-linear process. The neural network can be trained in a recursive manner where the switch in Figure 2.4 is in position 2 and hence the neural networks output is fed back as input during training. The approach used for training the neural networks was to train in the predictor configuration with one step ahead prediction at each sample time. This structure was used since straightforward back propagation can be applied as a training algorithm, whereas if training in the recursive mode, new training algorithms are required (WILLIAMS and ZIPSER, 1989; WERBOS, 1988). A comparison of four neural network learning methods for the identification of dynamic systems is given in QIN et al., (1992) where the two methods mentioned above are considered in what is called batch mode and pattern learning mode.

Pattern learning mode is the term used in this research when the weights in the network are updated after each pass of one sample of data from the training set, whereas batch mode refers to updating the weights in the neural network after each pass of the whole training data set.

Once the neural network has been trained to emulate the process, it is highly desirable to be able to use the neural network in the model configuration allowing the network to operate recurrently to predict the process output from only the input from the process. This is advantageous since the neural network could then be used to improve existing control strategies that employ multi-step-ahead predictions. Furthermore, the network can be operated independently from the process as a simulation tool to study process performance and characteristics, since it only

depends on process input values, and not process output values as in the case of the predictor. It has been found that a major factor affecting the model prediction accuracy is the method by which the data is presented to the neural network. In the model configuration, any error in the predicted output is fed back to the network inputs. Thus, an accumulation of error can occur which significantly degrades the model performance. To overcome this, a novel method of encoding the data presented to a neural network has been developed (section 2.7.2) that enables the neural network to be trained in the predictor configuration and once trained it can be implemented in a model configuration.

2.5 NETWORK TOPOLOGY SELECTION

Choosing the topology of the neural network is an important part in the development of a neural network representation of a process. The choices that have to be made are:

(i) The number of network inputs and outputs,

(ii) The number of hidden layers,

(iii) The number of processing units in the input, output and hidden layers,

(iv) What activation function should be incorporated into the processing units.

The theorem of Kolmogorov (KOLMOGOROV, 1957) which stated that any continuous function of N variables can be represented by using linear summations and non-linear continuously increasing functions of one variable, was first thought to give an indication of how many hidden layers and processing units in these layers should be used. However Girosi and Poggio (1989) contended that the usefulness of this theorem is nebulous since it would require a different non-linear

activation function for each processing unit in the neural network, and the functions in the hidden layers have to comply to certain undesirable conditions. One hidden layer was used in the neural network since Hornick et al., (1989), Funahashi (1989), and Cybenko (1989) proved that one hidden layer is adequate.

The sigmoidal activation function was used in both the hidden and output layers of the neural network. It was chosen due to its current popularity and reliability and it was the assumed activation function in the proof by Cybenko (1989). The use of the sigmoidal function in the output layer of the neural network, requires scaling the data presented to the network. This point is discussed in section 2.7, where data conditioning techniques are discussed in detail. The chosen number of processing units used in the hidden layer was found through experimentation. If too many processing units were used in the hidden layer, then training the neural network was slow, and its generalisation properties deteriorated due to over parameterisation. Furthermore, too few hidden processing units were not able to represent the required non-linear mapping between the input and the output data from the process, and the neural network did not converge.

In order to select a suitable number of hidden layer processing units, the network was successively trained down to a pre-specified suitable MSE for a range of processing units. It is seen in Figure 2.5 by using this method, that as the number of hidden processing units is increased then the number of iterations required to reach the specified MSE reduces, until after (in this case) six hidden units when the number of iterations increases.

One problem with this method is that if at the start of training the initial random network weights are such that the network output error is very close to the desired

error then this will result in only a few iterations of the data set, although this happens rarely.

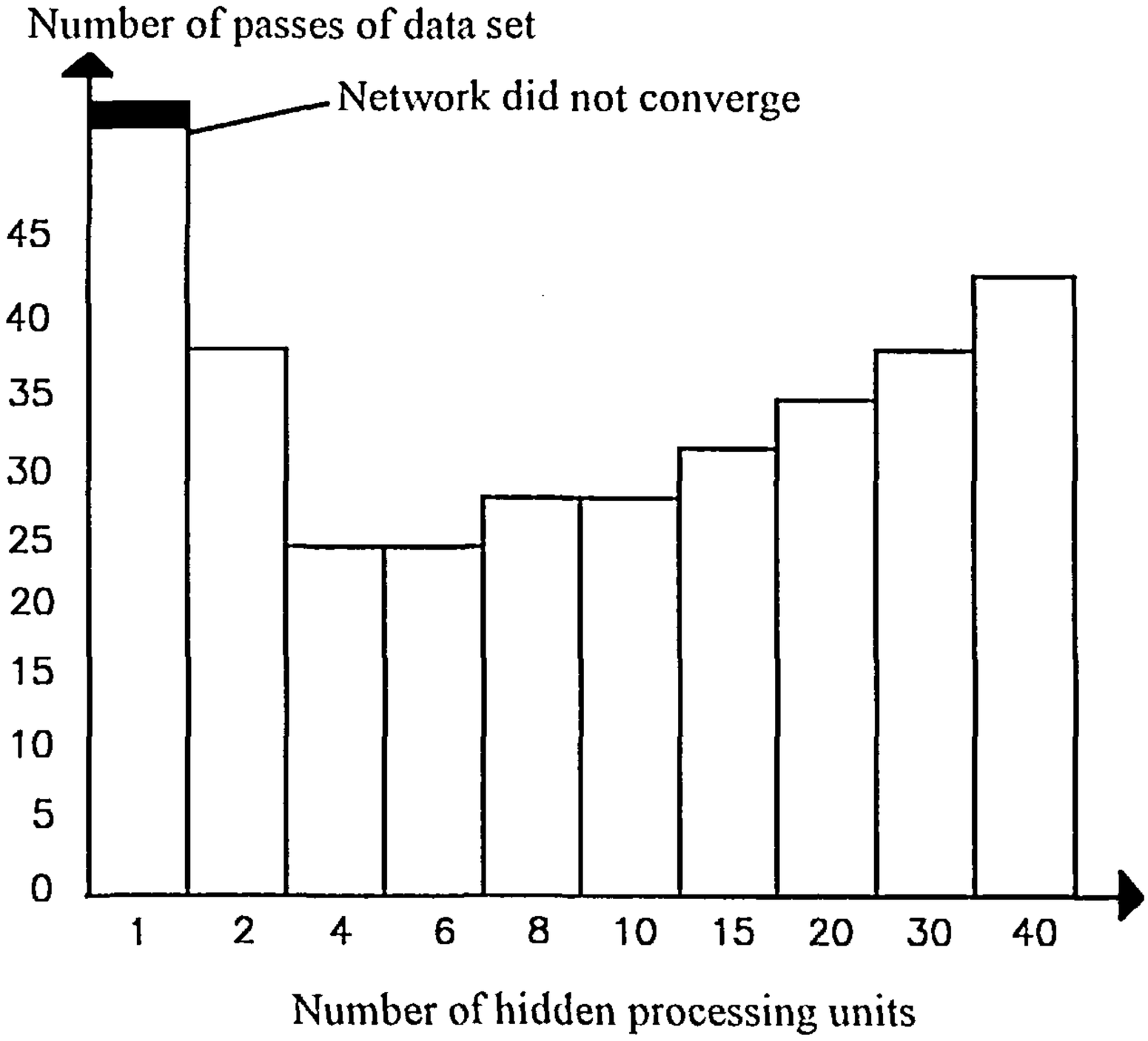


Figure 2.5 Selection of the number of hidden processing units

To overcome this possibility a number of sample runs with the same number of hidden processing units were performed, and the average number of iterations of the training data was the value noted. The assignment of the network input nodes is not a straight forward decision, and certain tests were used to decide upon the input structure.

2.5.1 Selection of input nodes using a correlation based technique

When developing a neural network model the output structure is specified by the application being considered. However, the choice of the number of input nodes is

not so easy to specify, and the wrong choice can affect network performance. The approach is similar to the NARX model order selection i.e. choosing the values for n_a, n_b and k in equation 2.17. It is suggested here that process knowledge can be used to provide an initial starting point, since a process known to exhibit predominantly n^{th} order characteristics can be represented in the NARX structure by n past inputs and outputs. As pointed out by (BILLINGS, 1992), the MLP neural network does not generate components of higher order lagged process inputs and outputs which are not specified in the network input nodes, because the output of the neural network is just a static non-linear expansion of the input nodes. Hence if a dynamic model of a system is represented by certain lagged inputs and outputs that are not included as input to the neural network then the network will be unable to generate the missing dynamic terms and the resulting neural network model will be poor. For many industrial processes obtaining a model may be neither cost effective or practical and in some cases may not be possible, hence the number of lagged inputs and outputs to be used as inputs to a neural network will not be straight forward. To overcome this problem correlation based tests devised by Billings and Voon (1986) were used. These tests detect if the resulting neural network model provides an adequate fit to the process being modelled. If a model of a system is adequate then the residuals, equations 2.19, 2.20, 2.21, 2.22, 2.23, should be un-correlated from all linear and non-linear combinations of past inputs and outputs. For an adequate model the following tests should hold:

$$\phi_{\varepsilon\varepsilon}(\tau) = E\left[\varepsilon(t-\tau)\varepsilon(t)\right] = \delta(\tau) \qquad \ldots(2.19)$$

$$\phi_{u\varepsilon}(\tau) = E\left[u(t-\tau)\varepsilon(t)\right] = 0, \quad \forall\tau \qquad \ldots(2.20)$$

$$\phi_{u^2\varepsilon}(\tau) = E\left[(u^2(t-\tau) - \overline{u}^2(t))\varepsilon(t)\right] = 0, \quad \forall\tau \qquad \ldots(2.21)$$

$$\phi_{u^2\varepsilon^2}(\tau) = E\left[(u^2(t-\tau) - \overline{u}^2(t))\varepsilon^2(t)\right] = 0, \quad \forall\tau \qquad \ldots(2.22)$$

$$\phi_{\varepsilon(\varepsilon u)}(\tau) = E\left[\varepsilon(t)\varepsilon(t-1-\tau)u(t-1-\tau)\right] = 0, \quad \tau \geq 0 \qquad \ldots(2.23)$$

The sampled correlation function between two sequences is calculated using equation 2.24

$$\phi_{v_1v_2}(\tau) = \frac{\sum_{t=1}^{N-\tau} \psi_1(t)\,\psi_2(t+\tau)}{\sqrt{\left[\sum_{t=1}^{N} \psi_1^2(t) \sum_{t=1}^{N} \psi_2^2(t)\right]}} \qquad \ldots(2.24)$$

The tests are improved if all the data is normalised to zero mean and unit variance. For these tests a 95% confidence interval is specified since the correlations will never be exactly zero, given by equation 2.25

$$\overline{x} \pm 1.96\frac{\sigma}{\sqrt{N}} \qquad \ldots(2.25)$$

with normalised data equation 2.25 results in $\pm$ 1.96/$\sqrt{N}$. N is the amount of data being used in the test. If the resulting neural network model is a good one then the correlations should be within the 95% confidence limit bands. The tests are not guaranteed to detect all possible non-linear terms that may be part of a neural network model, but simulation and practical results have indicated that the tests provide a good indication and are a powerful tool to use.

2.6 DURATION OF NETWORK TRAINING

When the overall network topology of the neural network has been selected, the final stage in development of the non-linear process model is to train and test the neural network representation. Training the neural network involves many epochs of the complete training data set through the network. After each epoch the weights in the neural network are adjusted in an attempt to map the given input data to the required output data. It has been found that if a neural network is trained for too long on the training data set then the final neural network model will be excellent at predicting over the range of the training set. However, when required to predict over a set of data not used to train the network, predictions can be poor. In many control applications, this aspect of network training is not addressed, and it is even suggested to continue network training until all the weights have fully converged (BHAT et al., 1990).

Figure 2.6 illustrates the effect on the output MSE of a network if over trained. Initially the MSE falls but, with further training, the MSE on the test data set increases, while the neural network still continues to predict accurately over the training set. The reason for the above phenomena is that if the weights in the network carry on being adjusted too long, eventually the weights will be such that the network has started to memorise the input-output data set pairs, instead of having weights that describe the overall relationship between the input-output data in the training data set. It is of paramount importance that the model of a process, be it a neural network or some other form of model, must be able to predict not only the data that it has been trained on, but also a wide range of different data. The approach that has been taken to overcome this problem is to use two different sets of data generated from the non-linear process being modelled. One set of data,

is used to train the neural network, and the other set of data is used to evaluate the networks generalisation capability at various stages of the training. The MSE of the testing data set is noted at each stage, and from this information a suitably trained network is selected.

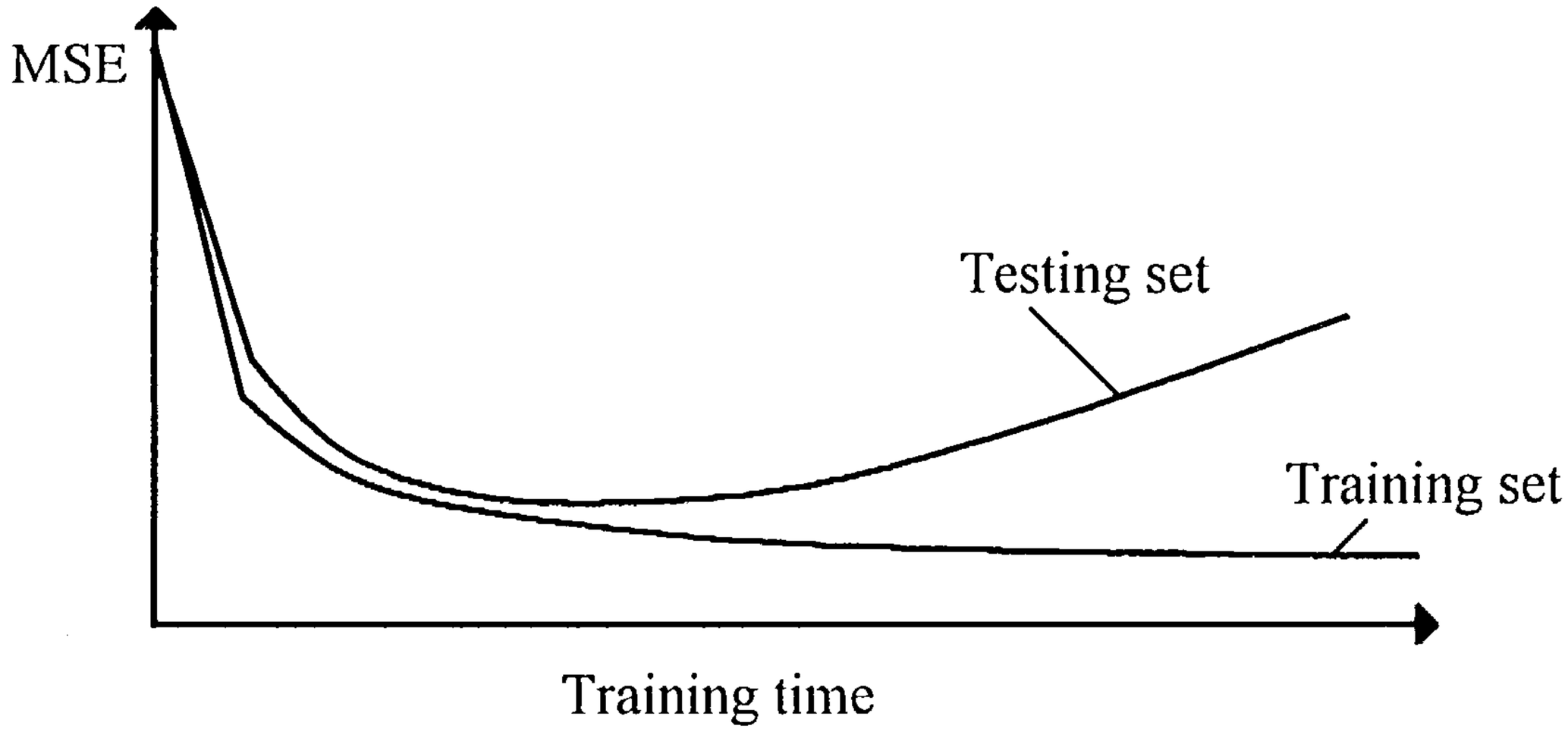


Figure 2.6 Illustration of generalisation during network training

2.7 DATA CONDITIONING

Before the sampled input-output data obtained from the process is presented to the neural network it should be checked to remove any undesirable features associated with real data. Such features include outliers and disturbances caused by random fluctuations in process operation. The input-output data is then normalised between upper and lower specified limits. When using a neural network the limits for the encoded data is usually dependent on the type of activation functions that are being used. The sigmoidal function has an output range of between zero and one, consequently the output layer has maximum and minimum outputs of zero and one. An obvious choice is to encode the sampled input-output data between these values, but doing so reduces the networks ability to learn the relationship between

the input and the output data (HOSKINS and HIMMELBLAU, 1988). This is explained by considering equation 2.5, which indicates that f(x) only reaches its boundaries as $x \rightarrow \pm \infty$ so that if the data presented to a neural network during the training stage is encoded between zero and one, then in order to produce a one or a zero output the weights between the hidden and output layer must be increased appropriately. The weights between the hidden and output layer are updated by equation 2.8 which is dependent on the first derivative of the sigmoidal function, and since as $f(x) \rightarrow 0$ or 1, $\bar{f}(x) \rightarrow 0$, the updating of the weights stop, and hence the learning process is difficult. This problem was overcome by setting the lower and upper limits of data encoding to 0.1 and 0.9 respectively. Another method is to use a linear activation function for each processing unit in the output layer.

The method of conditioning the data presented to the neural network is also crucial when the network is to be used recurrently as a model, since errors produced at the output of the network will be fed back into the input of the network, and will result in poor network performance. In an attempt to obtain an accurate neural network model representation two data conditioning methods have been investigated. One method of Single Node Data Mapping (SNDM), which is used by the majority of workers in the neural network field, was compared to a novel method of encoding the data which has been called Spread Encoding Data Mapping (SEDM).

2.7.1 Single node data mapping (SNDM)

This method of presenting the data to the neural network is widely accepted, and consists of applying each input variable to a single neural network input node, as illustrated in Figure 2.7. A single node is also used for each network output variable. The data is mapped linearly using a straight line relationship as given by equation 2.26.

$$(X_s - 0.1) = m(X - X_{\min}) \qquad ...(2.26)$$

where

$$m = \frac{0.9 - 0.1}{X_{\max} - X_{\min}} \qquad ...(2.27)$$

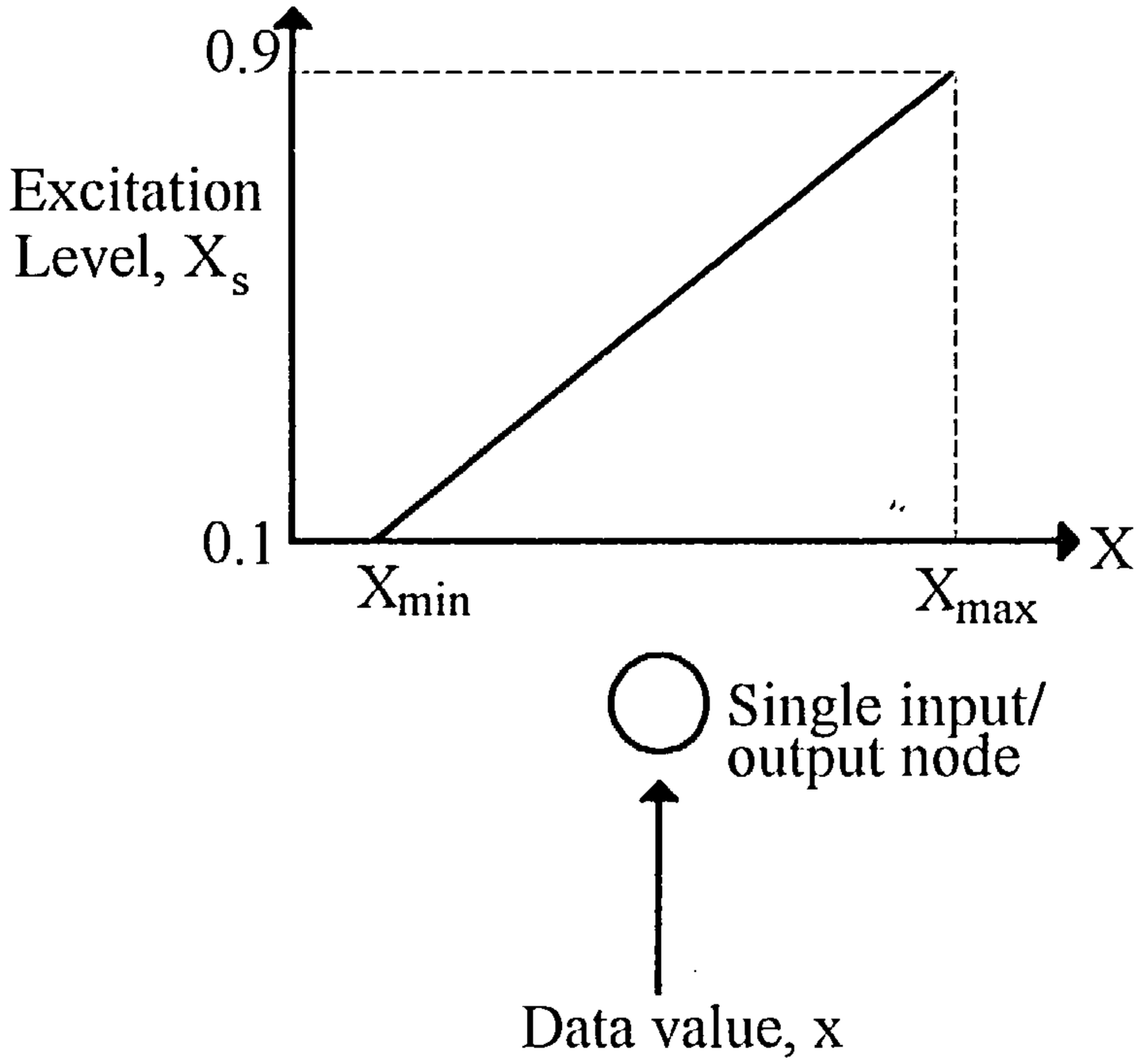


Figure 2.7 Single node data mapping, SNDM

combining equations 2.26 and 2.27 results in the overall data mapping equation given by 2.28

$$X_s = \frac{0.8(X - X_{\min})}{X_{\max} - X_{\min}} + 0.1 \qquad ...(2.28)$$

where X_S is the scaled value of X and, X_{max} and X_{min} are the maximum and minimum values of X.

To obtain the true value from the encoded value at the output of the neural network equation 2.28 is rearranged to give equation 2.29

$$X = \frac{(X_s - 0.1)(X_{max} - X_{min})}{0.8} + X_{min} \qquad ...(2.29)$$

2.7.2 Spread encoding data mapping (SEDM)

This form of coding is illustrated in Figure 2.8 where a variable, X, with a finite range [X_{min}, X_{max}], is mapped onto a sliding Gaussian activation pattern of N network nodes, with additional nodes either side to contain any overspill resulting from the use of a mapping function with wide support.

The level of activation for each node is confined as in the previous SNDM method to the range (0.1 to 0.9). The first stage of the coding is the scaling of the original data range into a normalised data range represented by $r \in (0, N\text{-}2N_a)$ which is achieved using equation 2.30

$$r = \frac{N - 2N_a}{X_{max} - X_{min}} * (X - X_{min}) \qquad ...(2.30)$$

where N_a is the number of nodes either side of the variable range and X is the data value to be coded over the N network nodes.

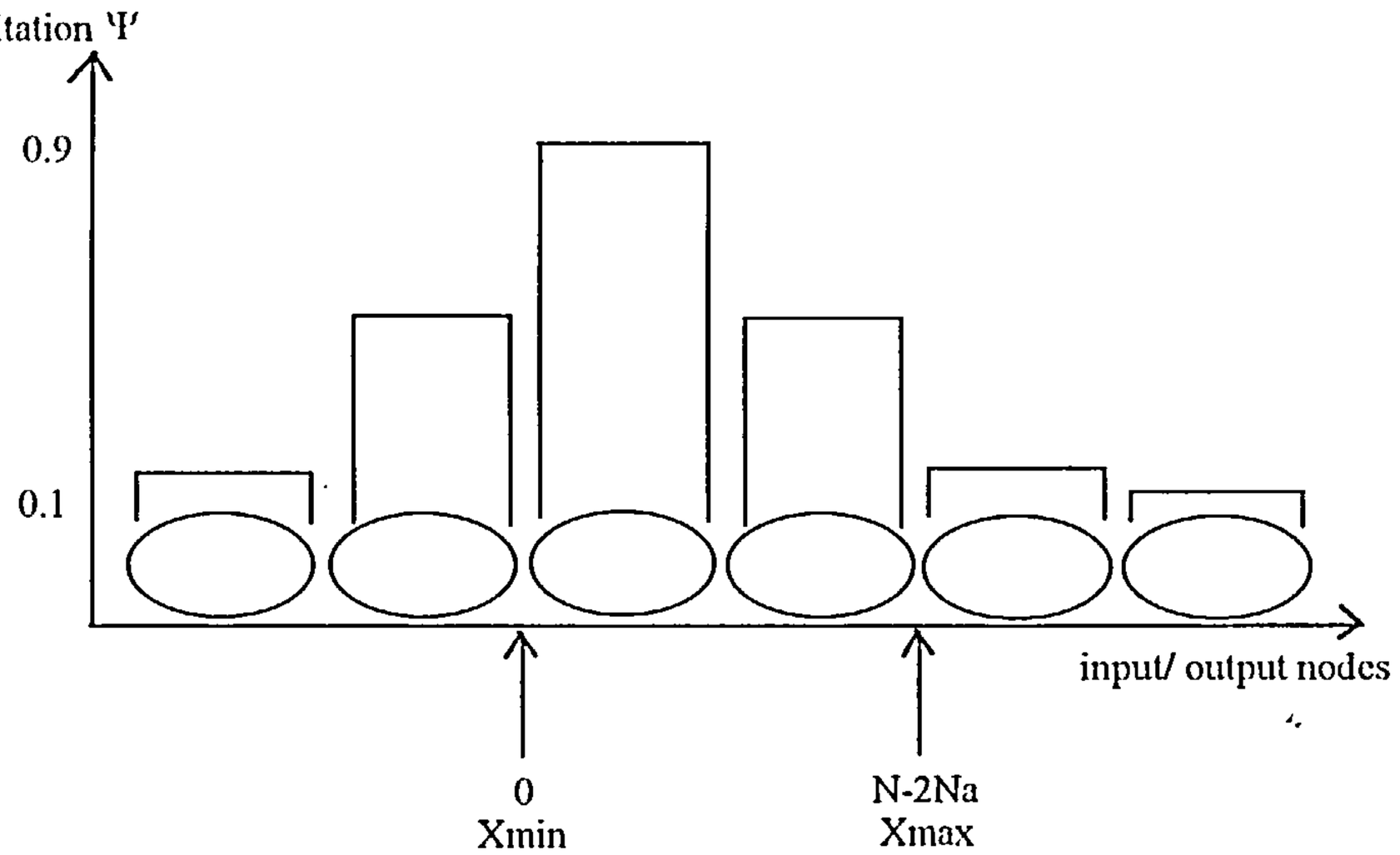


Figure 2.8 Spread encoding data mapping, SEDM

Once the data has been normalised the excitation level of each node is then found. The excitation of each node is defined by

$$\psi_i = \frac{\int_{a_i-\delta/2}^{a_i+\delta/2} a\,\phi(a)\quad da}{a_i} \qquad \ldots(2.31)$$

which satisfies the requirement that

$$\sum_{i=1}^{N} a_i \psi_i = \int a\phi(a)da = \bar{a} \qquad \ldots(2.32)$$

Each node is assigned a class interval, a_i, linearly spaced by a distance δ and a discrete map is created which represents the mean value of a continuous probability distribution, $\phi(x)$ within each class interval.

The activation of each node is evaluated from equation 2.31 by the use of integration by parts which leads to equation 2.33

$$a_i \psi_i = \left[a\,\Phi(a) \right]_{a_i-\delta/2}^{a_i+\delta/2} - \int_{a_i-\delta/2}^{a_i+\delta/2} \Phi(a)\, da \qquad \ldots(2.33)$$

where $\Phi(a)$ is a cumulative Gaussian distribution function with $\phi(a)=\Phi'(a)$. The cumulative Gaussian function was approximated by the sigmoid function. Throughout the research investigations the integral term in equation 2.33 was approximated using the first two terms in the trapezium rule which resulted in

$$\psi_i = \Phi(a_i + \frac{\delta}{2}) - \Phi(a_i - \frac{\delta}{2}) \qquad \ldots(2.34)$$

which was found to provide sufficient accuracy. The steps necessary to code a data value are now listed

Step 1: Scale the data value to the normalised range using equation 2.30

Step 2: Code the data to the N network nodes using equation 2.34

where

$$a_i = i - N_a - c$$

and

$$\Phi(z) = \frac{1}{1+e^{-\beta(z-r)}}$$

β inversely controls the width of the node excitations and c is an offset term which shifts the position of the range limits on the nodes.

Step 3: Scale the excitation of each node to the range (0.1,0.9) using a linear relationship such as equation 2.28.

The method used to retrieve the original coded value is to first apply the inverse scaling procedure used in step 3 above and then to decode the network output back to the normalised range using equation 2.35

$$r = \frac{\sum_{i=1}^{N} a_i \psi_i}{\sum_{i=1}^{N} \psi_i} \qquad \ldots(2.35)$$

Finally apply the inverse of the scaling relationship given by equation 2.30 to obtain the decoded network output.

In the studies performed throughout this research N=6, N_a=2, c=0.5, δ=1 and β=2 were found to provide sufficiently accurate coding and decoding of the process data.

The spread encoding data mapping technique was investigated in order to achieve better modelling accuracy. Also, in many areas including measurement and process control the physical variables usually span a wide range of values. When the range of these values is compressed to a single node, the nodal response to small changes in input is limited. The spread encoding method improves the network accuracy by representing variations in input data values as changes in the excitation of several

input nodes. Also using this method, generation of confidence, or error measures can be defined by the difference between the width of the activity patterns of the output nodes and that of the original Gaussian function used to map the target values. Noise reduction in the reconstruction of the output signal is also achieved, this reduction is by the central limit theorem, in proportion to the reciprocal of the square root of the number of active nodes.

2.8 SUMMARY

The Multi-Layered Perceptron (MLP) neural network has been described, and a number of algorithms have been considered to train this type of network. The standard back propagation algorithm was used in this research, due to the wide applicability and simplicity of the algorithm. All the neural networks constructed consisted of an input an output and one single hidden layer of processing units. The number of hidden processing units is determined empirically, and each of these units as well as the processing units in the output layer has associated with it a sigmoidal activation function. Choice of the number of output units is dictated by the problem under consideration, and the number of input nodes was determined by the use of process knowledge and correlation based methods.

Dynamics have to be introduced into the structure of the MLP neural network since the MLP network is a static network and will not itself learn dynamic relationships between process inputs and outputs. Various methods have been described to introduce these dynamics, and the method chosen is based on time histories of past process input-output data.

Two different methods of training and implementing a trained neural network have been discussed, namely the model and predictor configuration. The predictor configuration was chosen for training the MLP neural network because it leads to more stable convergence properties than the model configuration, when using the back propagation algorithm.

It is important to ascertain the required degree of training the neural network must undergo. It has been shown that over training reduces the generalisation properties of the neural network. The optimum degree of training was determined by using a testing data set to determine the networks prediction capabilities at various stages of the training period.

Conditioning the data presented to the neural network has been considered in detail, and a novel method termed "Spread Encoding Data Mapping", (SEDM) has been described, whereby each single data value presented to the neural network is spread over a fixed number of network input nodes using a Gaussian spreading function. This novel method has been considered in an attempt to achieve better network prediction accuracy when using the trained neural network in a recurrent model configuration, over the standard method of conditioning the data. The standard method of conditioning network data has also been described, and this has been termed Single Node Data Mapping, (SNDM).

CHAPTER 3

PROCESS MODELLING AND VALIDATION IN SIMULATION

3.1 INTRODUCTION

This chapter describes the procedures taken in obtaining an accurate model of a non-linear process using a neural network and the techniques used in validating the resulting neural network model. The initial modelling studies were carried out in simulation, firstly without noise and then for a more realistic case of simulated noise added to the appropriate variables. The simulations were performed in order to speed up development time and obtain indications on the feasibility of the techniques being used, before attempting to model, using a neural network, a real laboratory scale non-linear process.

The computer package ACSL (Advanced Continuous Simulation Language) was used for the simulation studies. ACSL has a range of standard algorithms for solving non-linear differential equations, and simulating noise. ACSL is based on the Fortran computer language, and user specific subroutines required at run time were implemented in this language. The non-linear differential equations are solved using ACSL in order to provide data for the neural network package NeuralWorks Professional 2, which was used for training and validating the neural networks. Neural networks of various sizes can be easily constructed using this software package, and many training algorithms, including the back propagation are built in. There is also the facility for writing user specific routines in the C programming language which was used in the development of the neural network model.

The two methods of conditioning the data (SNDM and SEDM, section 2.7), presented to the neural network were implemented and a comparison was made by validating neural networks trained using the two different coding methods on different test signals to the network training signal. The neural networks were validated in both one-step-ahead predictor and recursive model configurations.

3.2 PROCESS DESCRIPTION AND MATHEMATICAL MODELS

Figure 3.1 illustrates a dual tank non-interacting liquid level process. Although the process is not particularly complex, it was chosen because it exhibits features that are characteristic of many real industrial processes; non-linearity (due to the square root relationship between the tank outflows and the height of liquid in each tank), and also the height of liquid in tank one, h_1, is assumed to be unmeasurable, which is typical of many industrial processes in which variables are not, or cannot be measured. The process provides a good test bed on which to investigate the modelling capability of a neural network. The flow rate of the liquid into tank 1 is controlled via the pneumatic input to the process valve, and the output of the process is the height of liquid in tank two, h_2. The process is easily mathematically modelled from first principles and this enabled a thorough comparison of how well the trained neural network could emulate the process.

Simulation studies were carried out on two different but related processes. Initial investigations were performed on a first principles model obtained from a study by Gomm (1991), which will be referred to as first principles model 1 (FPMODEL1). A first principles model was then obtained for the actual physical liquid level process in order to perform simulation studies on the more realistic model. This model will be referred to as first principles model 2 (FPMODEL2). Both first

principles models are of similar structure, the differences being in the parameters used in the models.

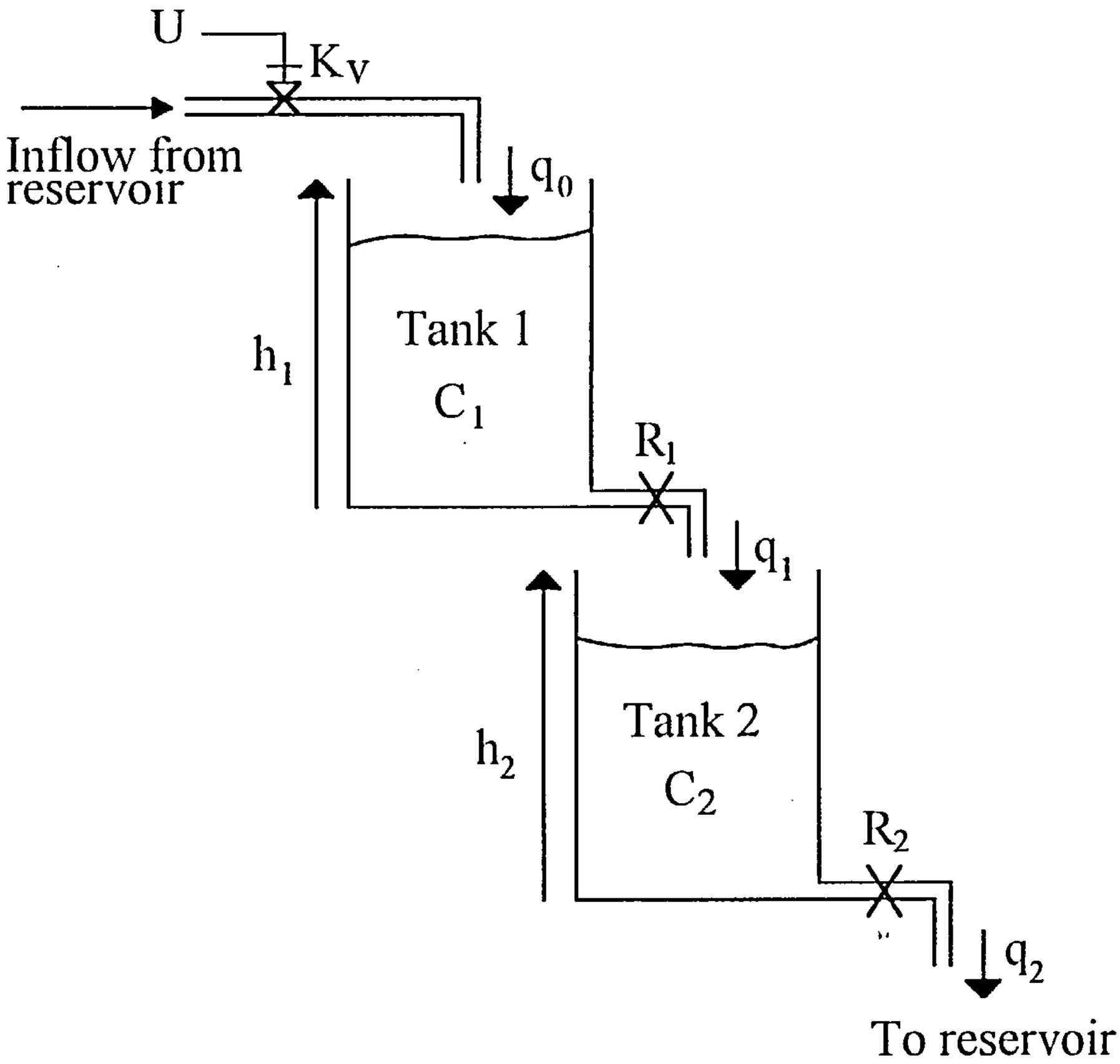


Figure 3.1 Dual tank liquid level process

3.2.1 First principles model one (FPMODEL1)

FPMODEL1 describing the dual tank non-interacting liquid level system (Figure 3.1) is described by the well established following non-linear differential equations:

$$C_1 \frac{dh_1}{dt} = (q_0 - q_1) \qquad ...(3.1)$$

$$C_2 \frac{dh_2}{dt} = (q_1 - q_2) \qquad ...(3.2)$$

where

$$q_0 = k_v u \qquad ...(3.3)$$

$$q_1 = \frac{\sqrt{h_1}}{R_1} \qquad ...(3.4)$$

$$q_2 = \frac{\sqrt{h_2}}{R_2} \qquad ...(3.5)$$

q_0 is the liquid flow rate into tank 1; q_1, q_2 are the liquid flow rates into and out of tank 2 respectively, h_1 and h_2 are the liquid levels in tanks 1 and 2 respectively, u and h_2 are the process input and output, k_v is the valve gain, C_1, C_2 are the cross-sectional areas of tanks 1 and 2 respectively and R_1, R_2 are the outflow pipe restrictances of the tanks. The initial conditions for the process were h_{10} and h_{20} for tanks 1 and 2 respectively, and u_0 for the input. The values of the initial conditions were chosen as h_{10} = 1.0 m, h_{20} = 5.0 m and u_0 = 0.5. The cross-sectional areas of both tanks had equal values of 1.0 m^2, and the restrictances were chosen as 5.0 $s/m^{5/2}$ and 1.0 $s/m^{5/2}$ for tanks 1 and 2 respectively and the value of the valve gain was set to 10. The values of the restrictances were chosen so that tank 1 would have a faster reaction time than tank 2. These values were implemented in the ACSL simulation of the mathematical model.

3.2.2 First principles model two (FPMODEL2)

The real laboratory scale dual tank liquid level process is illustrated in Figure 4.1. First principle methods were performed on the process, and the resulting mathematical model describing the process dynamics was of similar structure as FPMODEL1 described by equations 3.1 and 3.2. It was necessary to experimentally determine the values of the parameters R_1, R_2, C_1, C_2, and k_v of the real process, so that a mathematical model of the real process could be obtained for simulation studies. Using FPMODEL1 gave an insight into the nature of the two tank system, and the capabilities of using neural networks for modelling. However having a mathematical model of the real process gave a more realistic situation.

The cross-sectional areas of tanks 1 and 2 are equal, and were simply calculated from measuring the radius of the tanks, the radius of each tank is 0.0685 m which resulted in values for C_1 and C_2 of 0.01474 m^2. Values for the restrictances of both tanks R_1 and R_2 were obtained by plotting the square roots of the heights of liquid in each tank against the flow rate for a number of steady states, illustrated in Figure 3.2. The restrictances were calculated to be 3671.17 s/m^2 for tank 1 and 3065.58 s/m^2 for tank 2. The equations representing q_1 (the liquid flow rate into tank 2) and q_2 (the liquid flow rate out of tank 2) were calculated from the data in Figure 3.2 by fitting a polynomial of degree one in a least-squares sense to the data, and this resulted in equations 3.6 and 3.7:

$$q_1 = \frac{\sqrt{h_1} + 0.4939}{3671.17} \qquad \ldots(3.6)$$

$$q_2 = \frac{\sqrt{h_2} + 0.1469}{3065.58} \qquad ...(3.7)$$

The inflow rate into tank 1 q_0, given by equation 3.3 was obtained in a similar manner to above by plotting the inflow rate to tank 1 against the value of the input u. The result of this is illustrated in Figure 3.3. The input u to the process valve is a digital number that takes on integer values between a minimum of 100 and a maximum of 200, for a digital number of 100 the process valve was closed and the liquid inflow q_0 was cut off, and for a digital number of 200 the valve was fully open. The choice of the range of the digital numbers and more detail are given in section 4.2.1 where the real laboratory process is discussed. The liquid flow rate into tank 1 was calculated in the same manner as for equations 3.6 and 3.7 and is described by equation 3.8:

$$q_0 = -0.00599 * 10^{-6} u^2 + 3.42483 * 10^{-6} u - 85.83927 * 10^{-6} \qquad ...(3.8)$$

A second order polynomial was used to fit the data for q_0 as this resulted in a significant increase in accuracy over a first degree polynomial. Initial conditions for the height of liquid in tank 1, h_{10}, and height of liquid in tank 2, h_{20}, were set at 0.34 m and 0.56 m respectively, which corresponded to an initial digital number input, u_0, of 150. The parameters for simulating the FPMODEL2 were then implemented into an ACSL program.

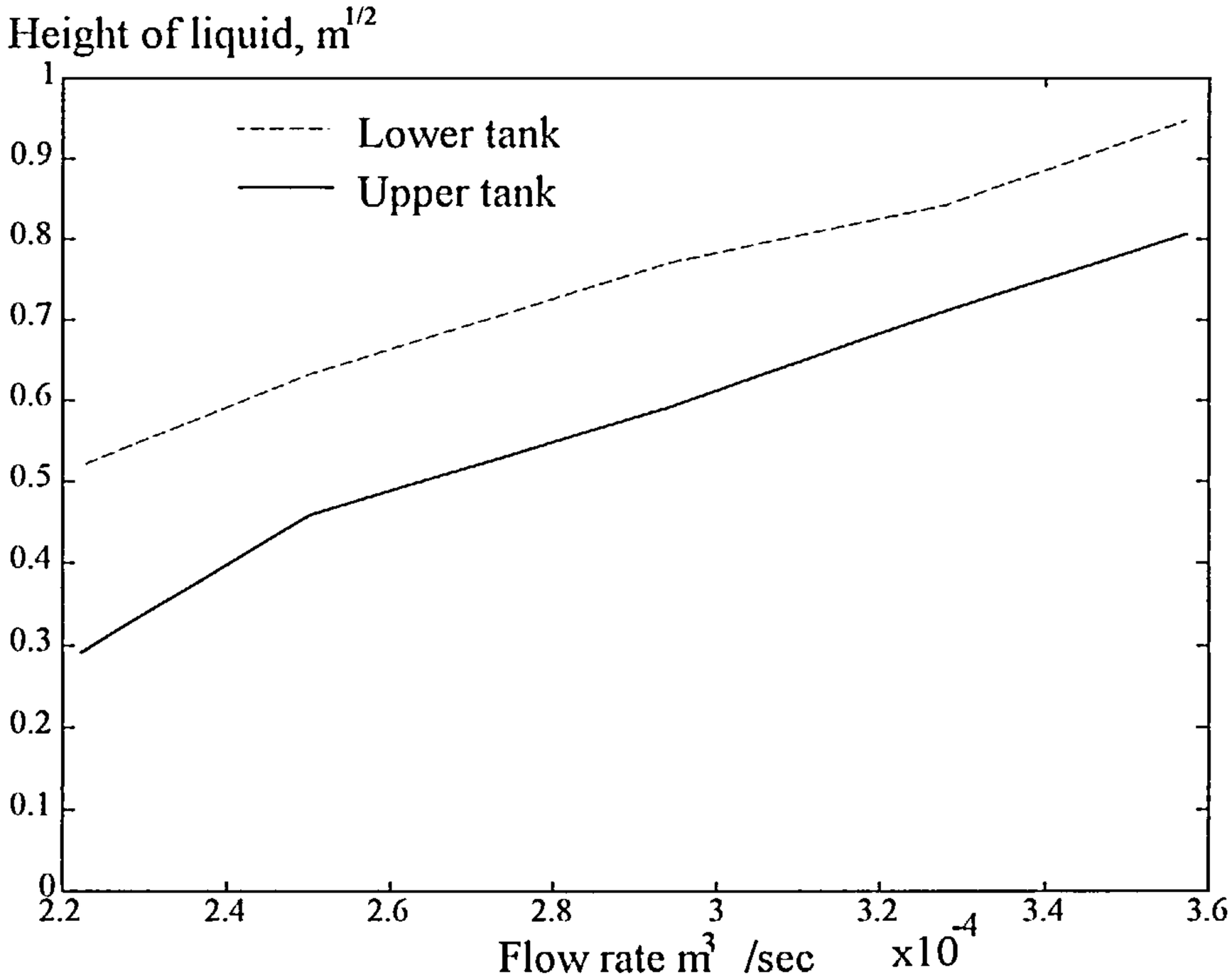


Figure 3.2 Height of liquid in tanks one and two versus liquid flow rate

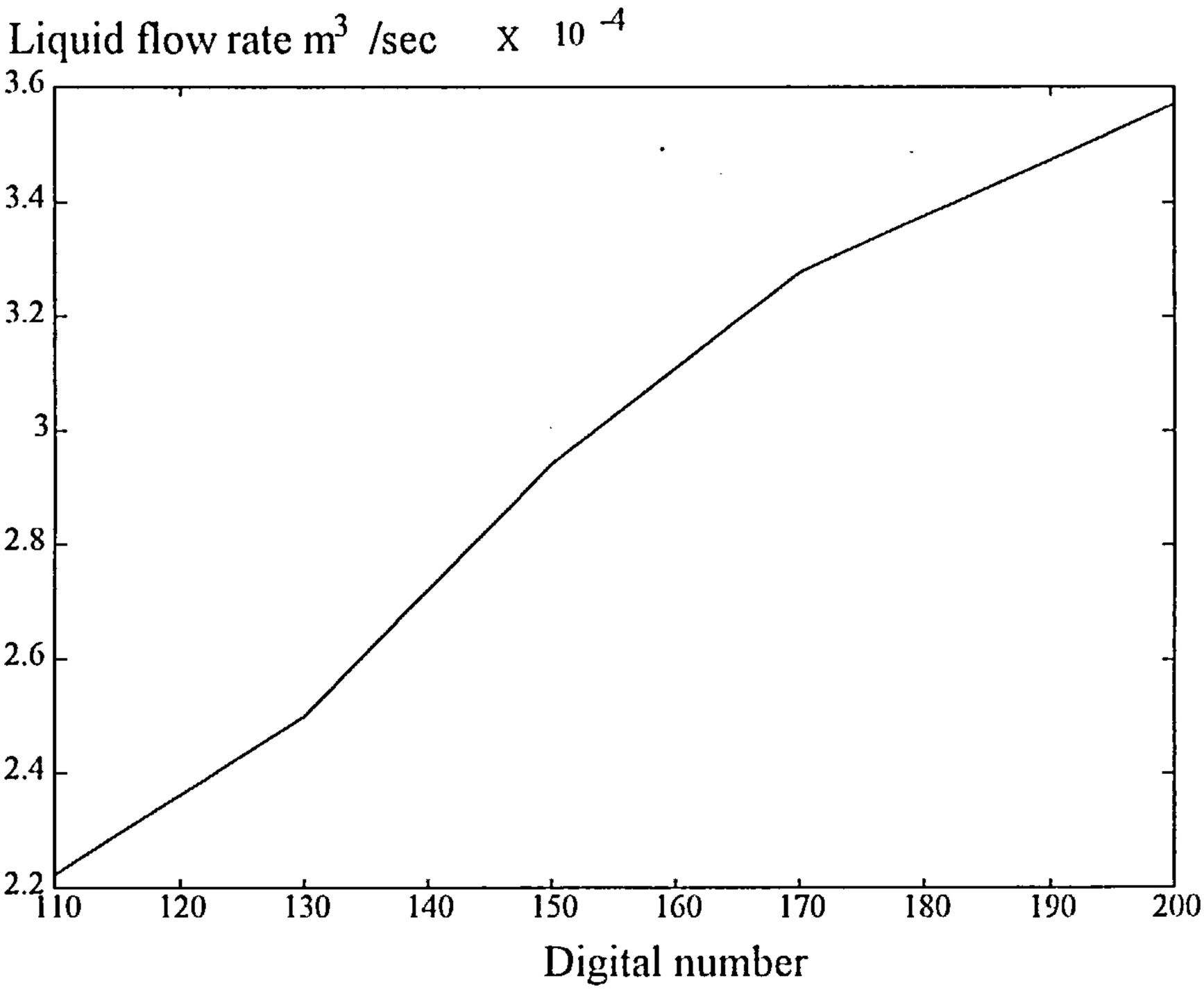


Figure 3.3 Liquid flow rate from pump versus digital number to process valve

3.2.2.1 Comparison of the laboratory process against FPMODEL2

To test the accuracy of FPMODEL2 the response of the real process to a set of input signals was compared to that of the mathematical model for the same inputs. If FPMODEL2 is a 'good' representation of the laboratory process then the response of the model should be very similar to that of the process. Two signals were chosen to evaluate the performance of FPMODEL2, these were a PRBS input and a set of step inputs to test the steady state performance of the model. The response of the real process to the PRBS input is shown in Figure 3.4 along with the response of FPMODEL2. It is clearly seen that the output of the model accurately follows the liquid level response (the height of liquid in tank 2, h_2), there are slight but acceptable differences in the accuracy of the model between the upper and lower levels of the PRBS response. The performance of the real process together with that of FPMODEL2 for a set of step inputs is illustrated in Figure 3.5, Again the transient behaviour of the model is in very good agreement with that of the process, and the lower steady state is captured very accurately. A slight offset occurs at the upper steady state levels, the maximum offset between the process output and the model being 0.021 m, however, considering that the real process is subject to time varying dynamics and external disturbances this is a small acceptable error.

The performance of the FPMODEL2 compared to that of the laboratory scale dual tank liquid level process indicates that an accurate model of the process has been obtained. The use of this model will aid considerably in the simulation studies, and techniques developed for modelling FPMODEL2 using a neural network should be readily applied to the real process without too much difficulty.

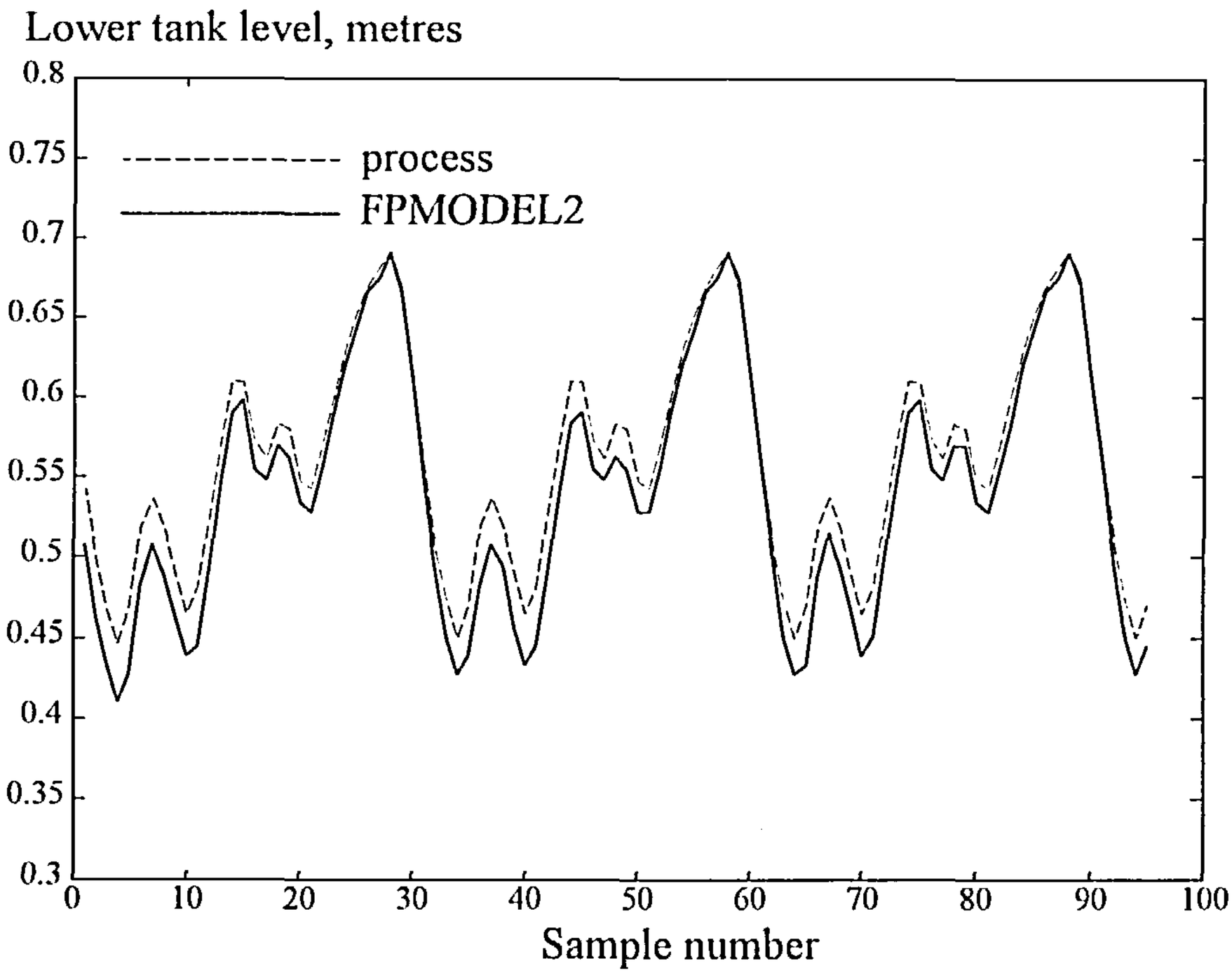


Figure 3.4 Response of process and calculated model for PRBS input signal

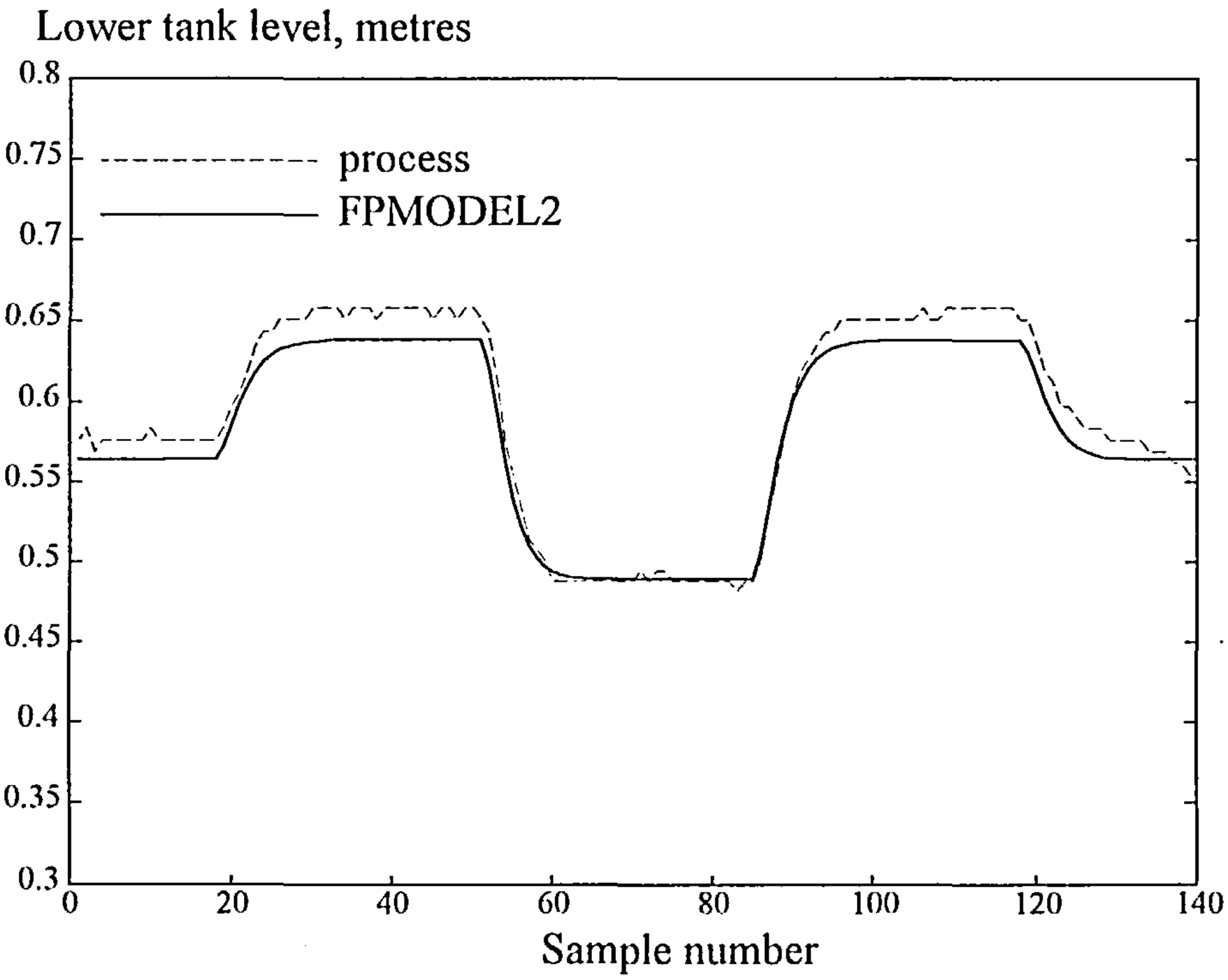


Figure 3.5 Response of process and calculated model for step input signal

3.3 PRACTICAL ASPECTS OF PROCESS MODELLING

Before simulation studies were performed with FPMODEL1 and FPMODEL2 certain practical aspects were considered, these are now discussed.

3.3.1 Sampling time selection

The frequency at which the input, u, and output, h_2, data should be collected from the process is an important factor. Data should be collected at a sufficient rate to avoid aliasing at the maximum frequency of interest. There are a number of guidelines for choosing the sampling time, one method (ASTROM and WITTENMARK, 1984) states:

$$T_S \langle 0.38 * t_{\min} \qquad \ldots(3.9)$$

where T_S is the sampling interval and t_{min} is the smallest time constant of interest. A method suggested by (SEBORG et al., 1986), states that the sampling time should be approximately one fifth to one tenth of the dominant time constant. Another guideline for T_S is between a fifteenth and one fifth of the ninety five percent settling time of the process to a step input (GUSTAVSSON, 1975).

In all of these guidelines the time constants of the process play an important part in selecting an appropriate sampling interval. Since the process being considered is non-linear in nature there is no single time constant for each of the tanks as there would be in the case of a linear model. The response time and gain of the process are dependent upon the operating region of the process. Knowledge about the

process indicated that the bottom tank (tank 2) had a much slower response time than the top tank, and hence tank 2 contains the dominant time constant.

For a linear first order system the time constant can be obtained by measuring the time taken for the measured variable to reach 63.2% of its final value after the application of a step input (BURGHES and GRAHAM, 1980). The approach taken was to use this fact and small step inputs (u) were applied to the process over a number of different operating regions. The time taken for the process output h_2 to reach 63.2% of its final value at the different operating regions was monitored. For small step changes in the input, the process can be assumed to be linear around its operating point, and the time constant of tank 2 can be estimated. The above was performed on FPMODEL1 and also on the real laboratory scale process resulting in a number of time constants for each of the different operating points. From the range of time constants obtained the choice for the real process was 3 minutes, and for the FPMODEL1 5 minutes. The sampling period was taken to be a fifth of the dominant time constant, resulting in FPMODEL2 having a sampling rate of T_S=0.6 minutes, and for FPMODEL1 T_S=1 minute.

3.3.2 Process excitation

In order to identify the parameters, represented by the weights in the neural network, of a dynamical process model, the training data must contain sufficient information about the process under consideration. Also the training data set should be representative of the entire class of inputs that the eventual process model will be subjected to (NARENDRA, 1990). This will ensure that the process model will respond in the desired fashion even when an input, not included in the training set, is applied to it. In general terms, the input signal to the process should persistently excite the process in all modes of operation. In linear systems theory a

signal that satisfies the above requirements is a Pseudo Random Binary Sequence (PRBS) signal (LJUNG and SODERSTROM, 1983), and this was the signal that was initially used to excite the process in simulation. FPMODEL1 was excited around the initial conditions, with small deviations and a neural network was used to identify the process around these conditions. The resulting neural network model gave excellent process predictions within the bounds of which it had been trained on. However as the range over which the process was excited increased the trained neural networks prediction performance deteriorated (LISBOA et al., 1991). The neural network was able to predict well on the PRBS signal that it had been trained on, but when required to predict process performance to other PRBS signals not used during training, the network failed completely. This was believed to be due to an inappropriate range covered by the input excitation signal, since using the PRBS, the input is always either fully on, or off. Consequently a signal with uniform random amplitude (RAS) was used to obtain an input-output data set for training purposes. This signal is sufficiently rich in frequency content and excited the process over different modes of operation. The RAS is described in section 3.4.1 where modelling of the process using a neural network is thoroughly investigated.

3.3.3 Amount of training data

An important aspect that should be considered when obtaining training data from a process is how many samples are required in order to train a neural network. This point has not been addressed by many workers in the neural network field even though it is an important one. For linear system identification using an ARX modelling approach, the number of observations, K, taken from the process should be much larger than the number n_a+n_b of parameters to be estimated (STREJC, 1981). This seems a reasonable approach to take when obtaining data from the

process to estimate the weights in the neural network. Another guideline suggested by Hall and Smith (1992) is to obtain as much data as possible, the more data, the better the solution, and that a reasonable heuristic for many problems is that of 100 data points for each node in the hidden layer. They also point out though that many acceptable results have been demonstrated with relatively small training data sets. When obtaining data from the simulation of a process model, the amount of data available to train and test a neural network is of no problem. However when a real process is being investigated the amount of data available may be restricted. These points should be considered when constructing a neural network process model.

Hence, for all the simulation studies conducted 1000 input-output data points were obtained for training the neural networks.

3.3.4 Network validation

Trained neural networks were validated on a number of test signals in both one-step-ahead predictor and recursive model configurations. The test excitation signals chosen to evaluate network performance were:

(a) a different RAS to that used for training;

(b) a PRBS;

(c) a set of step inputs.

The input levels of each of these signals were such as to excite the liquid level process over a wide operational region.

Another method used to examine how well the network had learned the process dynamics, was to obtain frequency responses around central operating points for

both the process simulations and the trained neural networks, using spectral analysis techniques (LJUNG, 1987).

3.4 NOISE FREE SIMULATIONS

The initial noise free investigations were carried out using FPMODEL1 and then on the calculated mathematical model of the physical process, FPMODEL2. For both models, the two forms of conditioning the process data, namely SNDM and SEDM (section 2.7), were implemented, and the resulting trained neural networks were validated in both the one-step-ahead predictor and model configurations. The realisation of the neural network models using the different coding techniques is now discussed.

3.4.1 FPMODEL1 using SNDM

FPMODEL1 was implemented into an ACSL program in which the process was excited by a RAS applied to the process valve, in order to control the liquid inflow rate into tank one, q_0. The RAS was produced by adding a random value, generated using a function available in ACSL, between -0.5 and 0.5 to the steady state value of the process input. The chosen values enable the process output, h_2, to be excited over the desired non-linear region.

Two data sets were generated in the simulation, one of them "data set one" Figure 3.6 was used for training the neural network, and the second "data set two", Figure 3.7 was used to test the neural networks generalisation capabilities at various stages of the training period. Different random amplitude signals were generated by

initialising the random number generator in ACSL to distinct values before the start of the simulation.

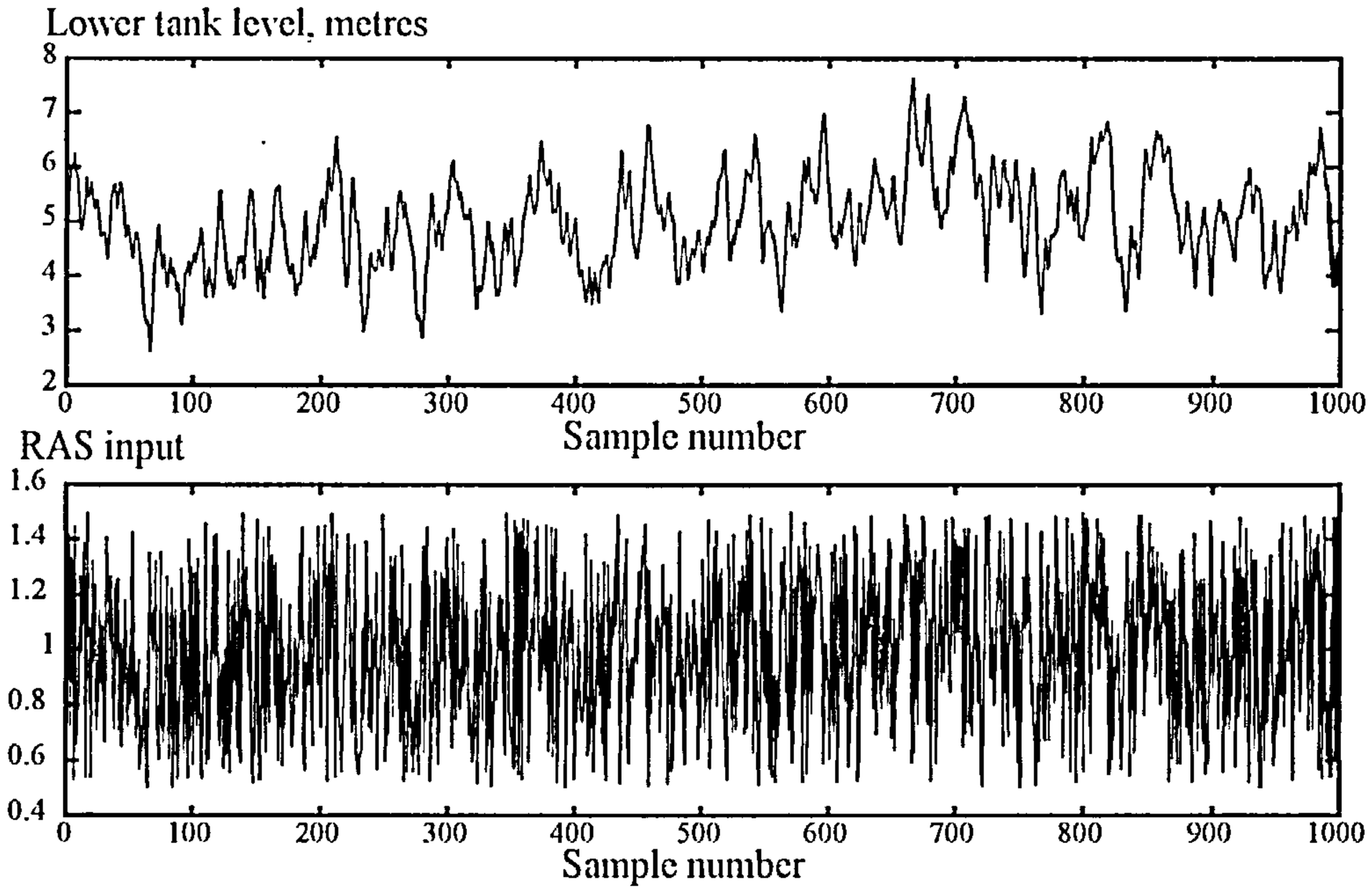


Figure 3.6 Process data set used for training

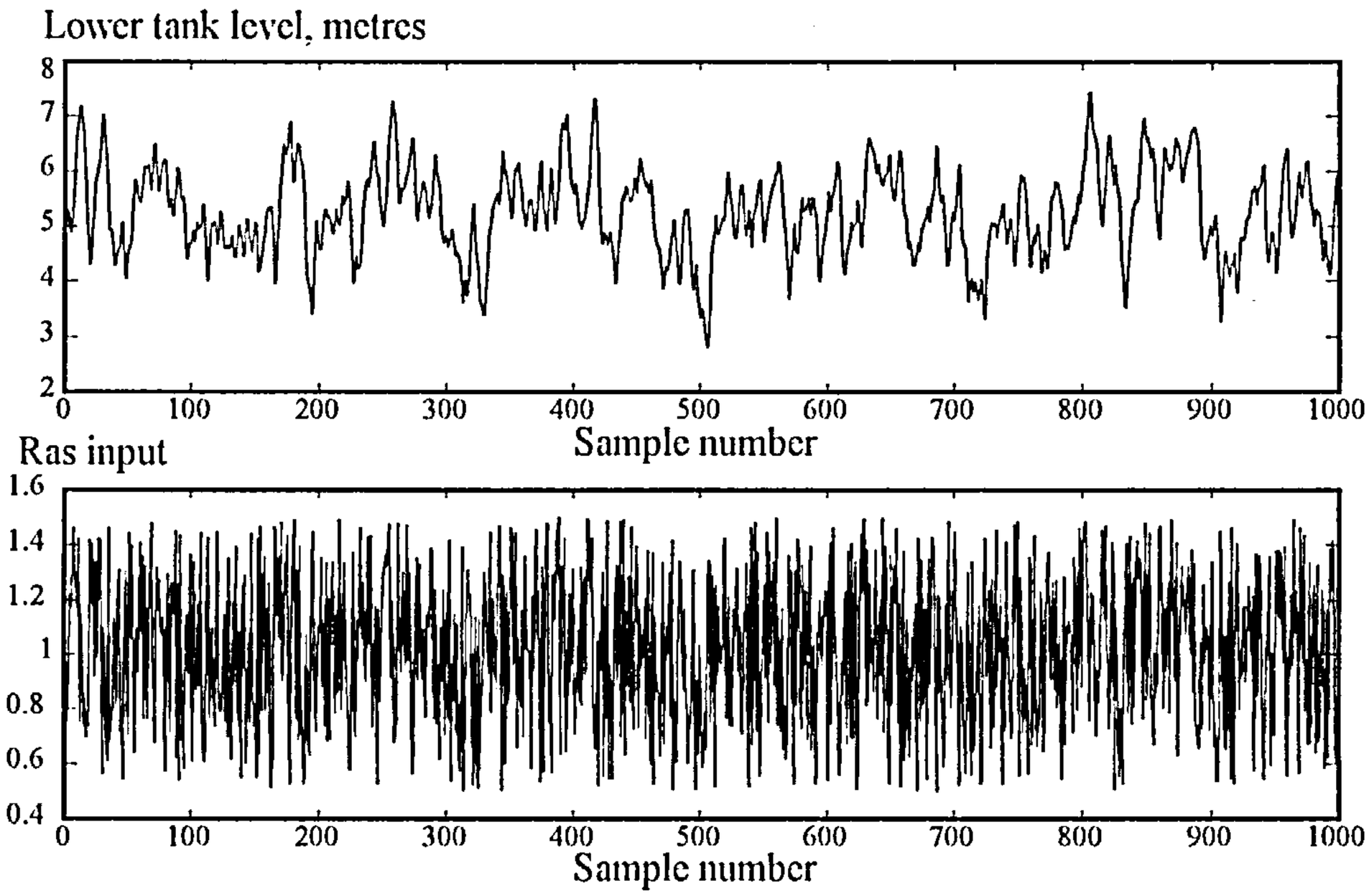


Figure 3.7 Process data set used for testing

3.4.1.1 Neural network structure

As described in Chapter 2, the choice of the number of input nodes can severely affect the network performance. Since it is known that the process when linearised around an operating point has second order dynamics, two past inputs and two past outputs were used as input to the neural network configured in the NARX structure. The objective was to develop a neural network that could predict the height of liquid in the lower tank, h_2, and hence this required one processing unit in the output layer. The number of processing units in the hidden layer was chosen by experimentation. For each case, the neural network was trained on "data set one" until the MSE for the complete data set was less than a pre-specified value of 0.0004 which was deemed to be a suitable choice. The number of complete passes of the data set was noted for each of the networks and is illustrated in Figure 3.8.

It can be seen that the neural network converged to the specified MSE for each of the number of hidden processing units specified, the optimum number chosen was eight. The overall network topology was as shown in Table 3.1.

Number of input nodes	Number of hidden nodes	Number of output nodes
4	8	1

Table 3.1: Neural network topology for FPMODEL1 using SNDM

With this network topology there was a total of 49 network weights to be evaluated.

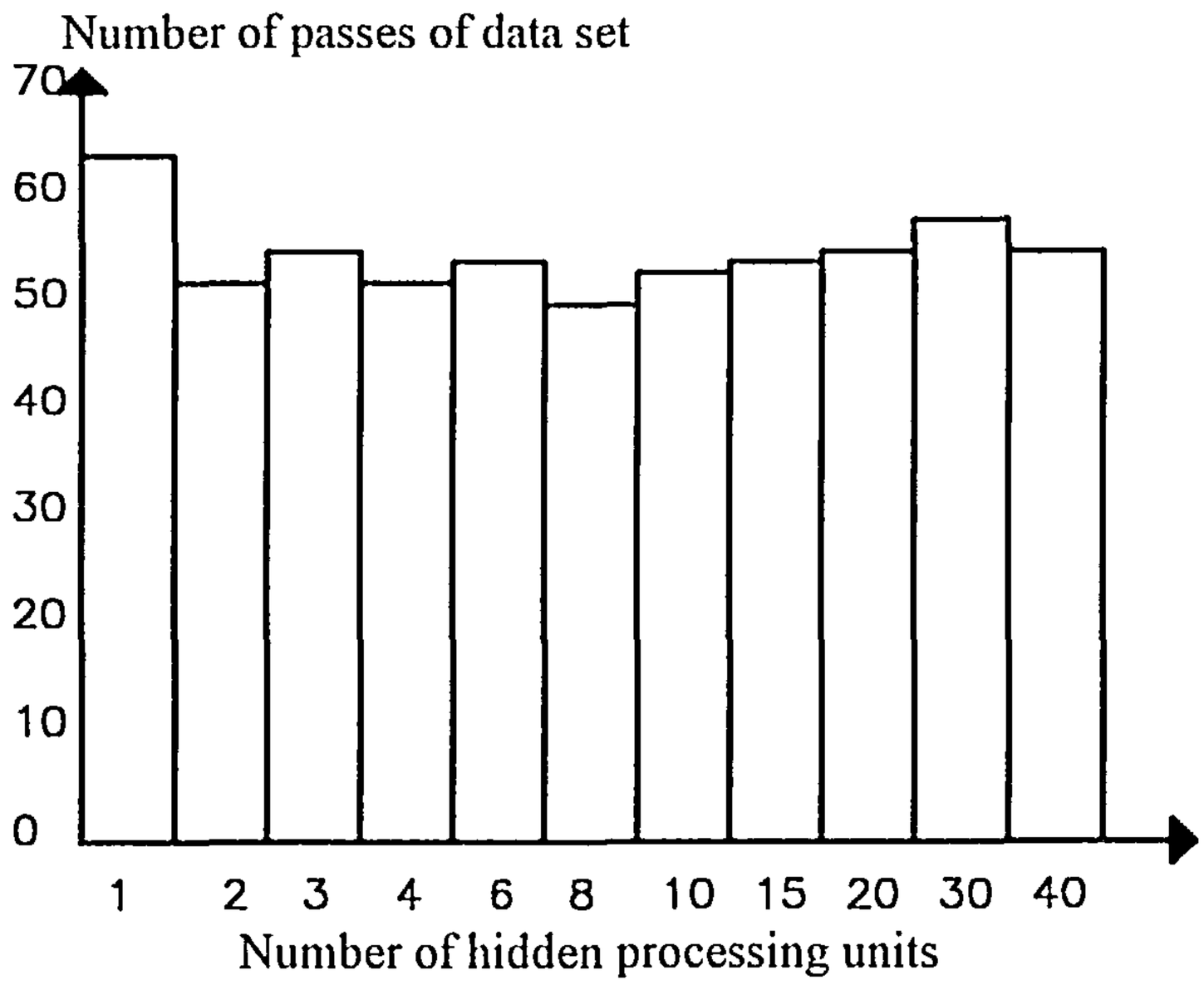


Figure 3.8 Selection of the number of hidden processing units

3.4.1.2 Training the neural network

Once the overall topology of the neural network had been decided upon, the next stage was to train the neural network on the data obtained from the simulation of the process. The two data sets were conditioned using SNDM so that the range of the data presented to the neural network was within the bounds of the sigmoidal function.

The weights in the neural network were initialised to random values between -0.1 and 0.1, and this was the case for every network trained. The initial values of the learning rate and momentum terms in the back propagation algorithm were set to values close to 1.0. After a certain amount of training, the values were reduced to improve the networks convergence. The initial, near unity values enable faster learning, however, if kept at these values, this will cause oscillations in the network output error as the network approaches its minimum. The values of the learning rate and momentum used are illustrated in Table 3.2.

No. of training passes	Learning rate η	Momentum α
< 70	0.9	0.6
> 70	0.4	0.15

Table 3.2: Values of learning rate and momentum during network training

During the initial simulation studies it was discovered that when the neural network was tested on "data set two" whilst configured in the predictor configuration the MSE between the predictions of the network and the required process output kept on reducing as the training proceeded, Figure 3.9.

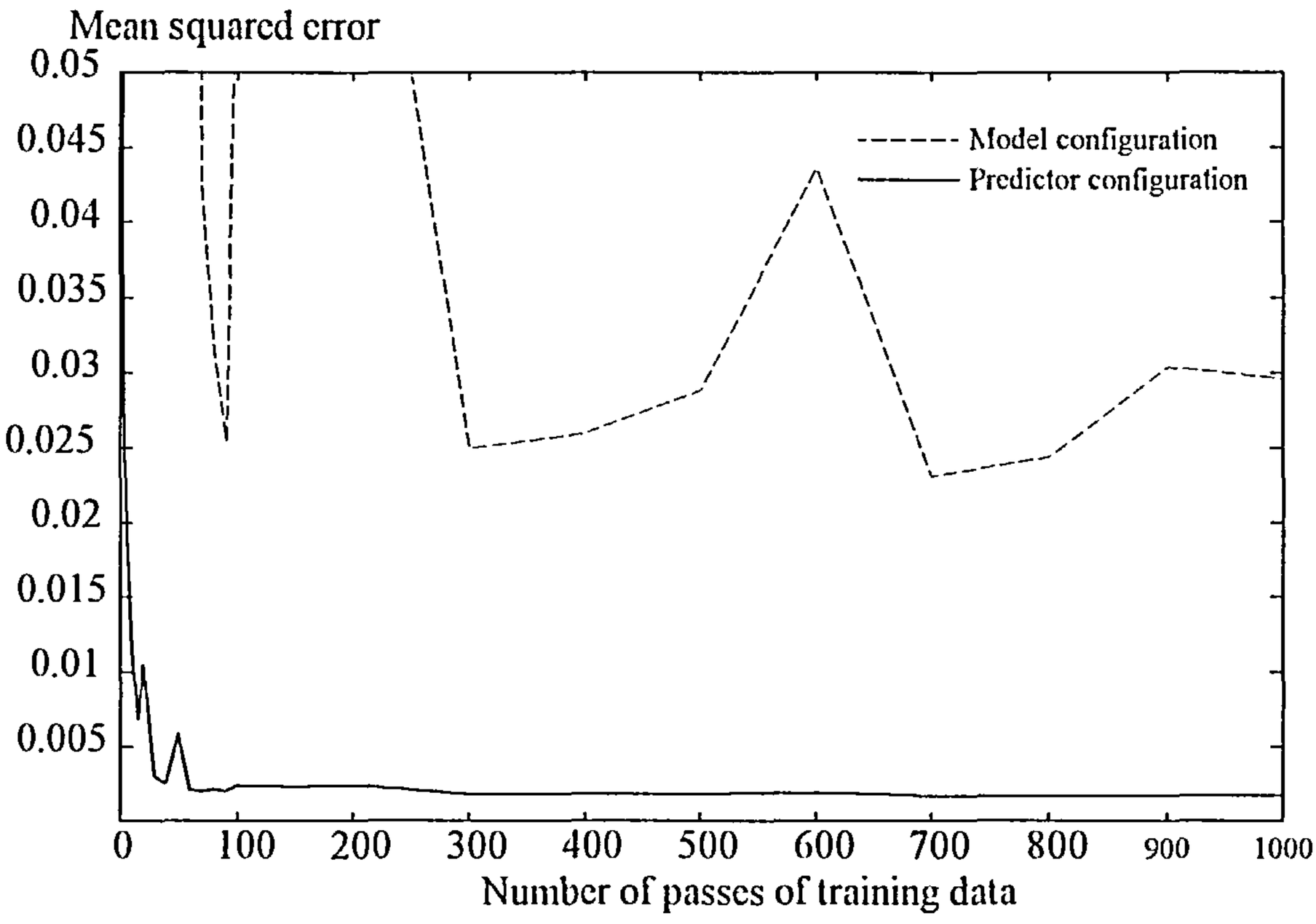


Figure 3.9 Termination of network training

In order to evaluate the networks generalisation it should be tested in the model configuration at various stages of the training, the result of this is also shown in Figure 3.9, which indicates that there is an optimum stage at which to stop network

training after which the MSE starts to increase. The neural network was considered to be optimally trained after 700 complete passes of the training data set and network training was terminated.

3.4.1.3 Validation of the trained neural network

The trained neural network was validated on the test signals described in section 3.3.4. Results for the trained network operating in the one-step-ahead predictor configuration will firstly be presented.

The performance of the trained neural networks predictions compared to the output of the process, when tested on the training RAS was very accurate as expected. The prediction performance when tested on a different RAS is illustrated in Figure 3.10. Only the first 100 points of the response have been shown in order to improve clarity. The response of the network accurately matches that of the process, and this is supported by a small MSE of 0.0017.

Similar accuracy was obtained when the neural network was subjected to a PRBS input signal, Figure 3.11. When the input signal applied to the network was a set of steps, there were offsets between the neural network and the required process output for two of the steady state values although the network had captured the two other steady state levels accurately, Figure 3.12. The network follows the response of the process to the set of steps very well and this suggests that the network has learned the transient behaviour of the process.

The network was then configured in the recursive model configuration and the signals used above were reapplied to the neural network to evaluate the performance as a recursive model.

Figure 3.13 illustrates the response of the process and the neural network predictions for the test RAS. As a model, the accuracy of the network has deteriorated, and this is shown by large errors in the networks response.

When tested on the PRBS signal the accuracy was not as good as when the neural network was configured as a predictor, as the neural network fails to reach the upper limits of the PRBS response and there are slight errors throughout the predictions, Figure 3.14. The performance on the set of steps, Figure 3.15, was very poor, with large offsets occurring between the required steady state values of the process.

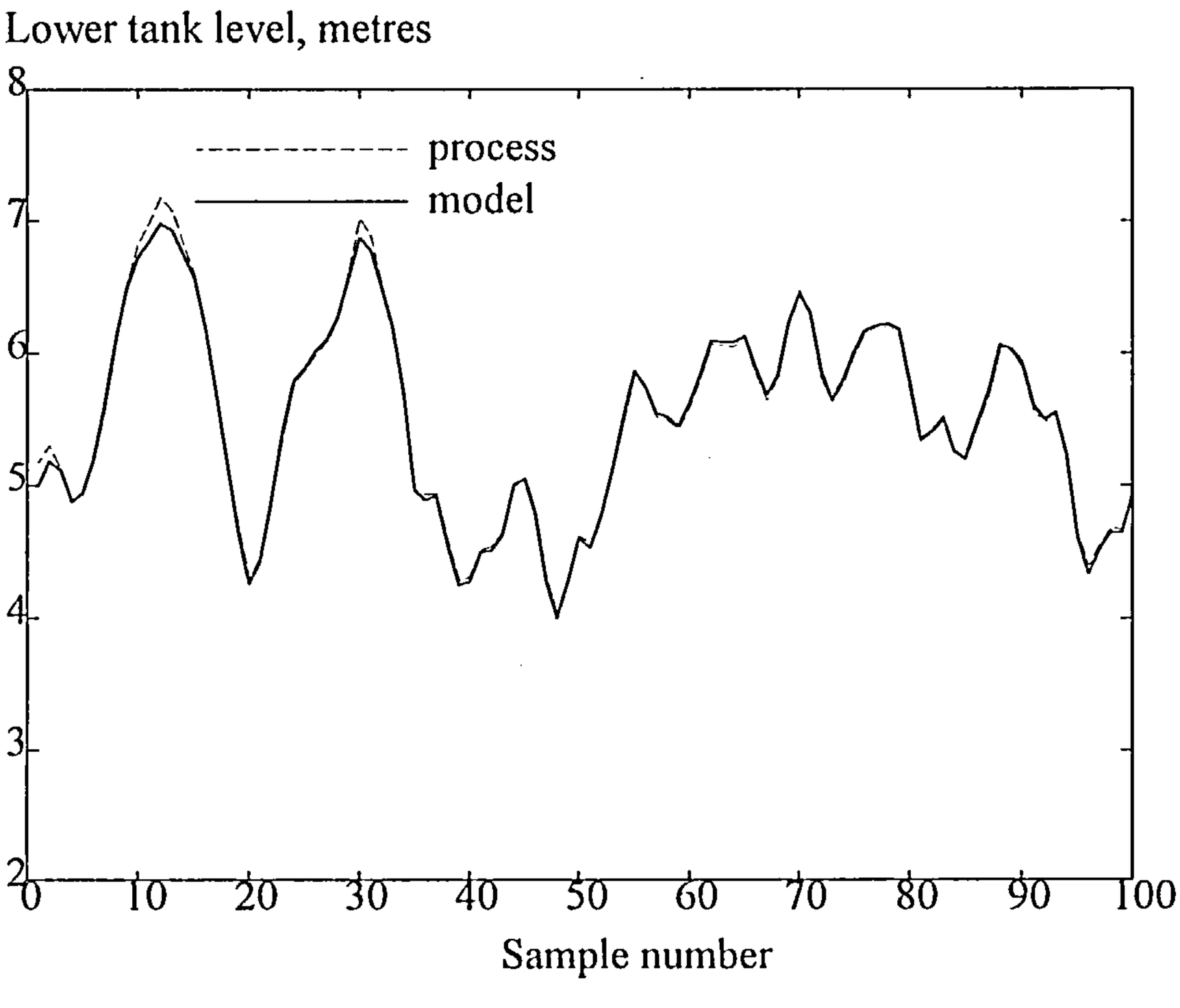


Figure 3.10 Response of SNDM network tested as a predictor on a random amplitude signal

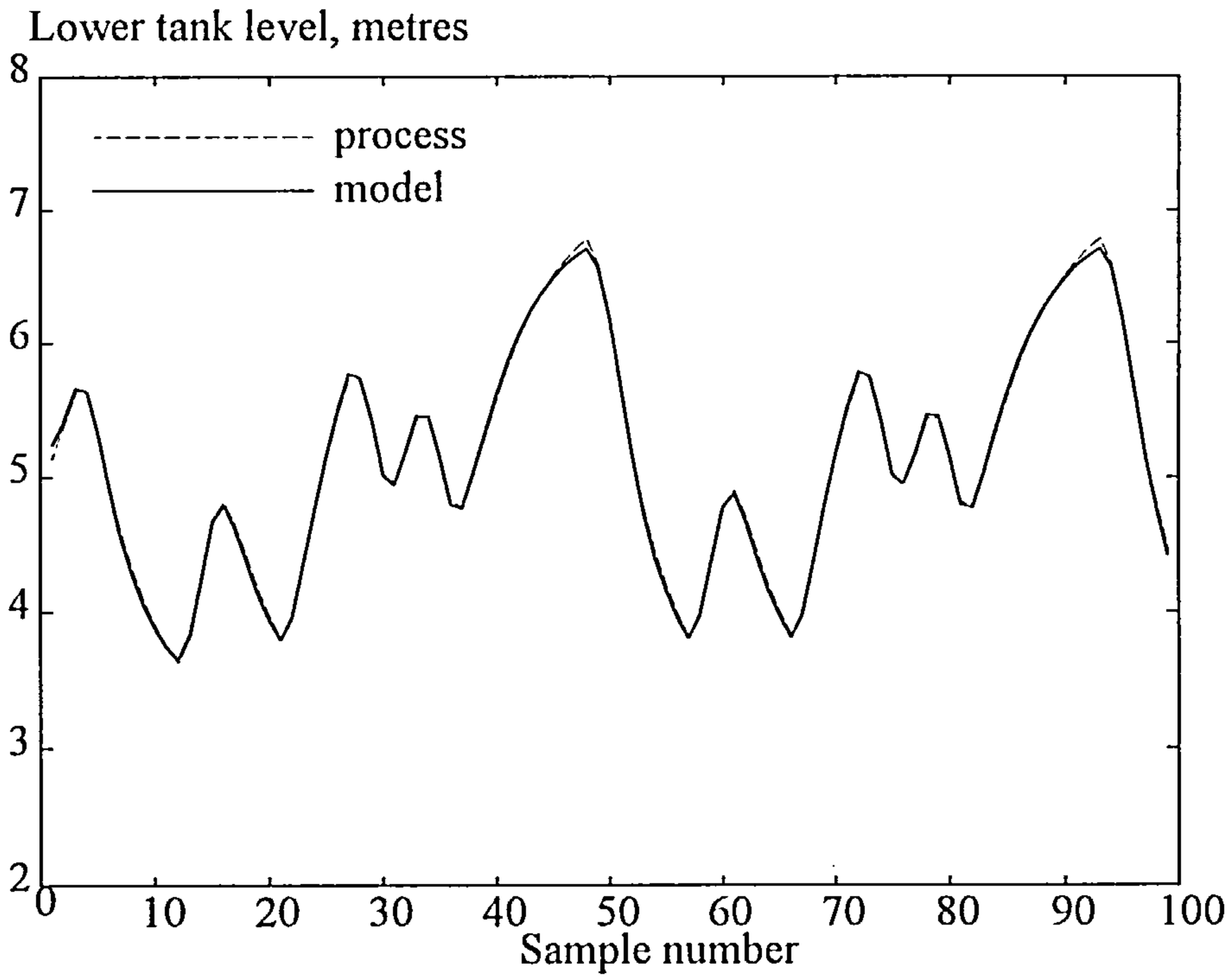


Figure 3.11 Response of SNDM network tested as a predictor on a PRBS signal

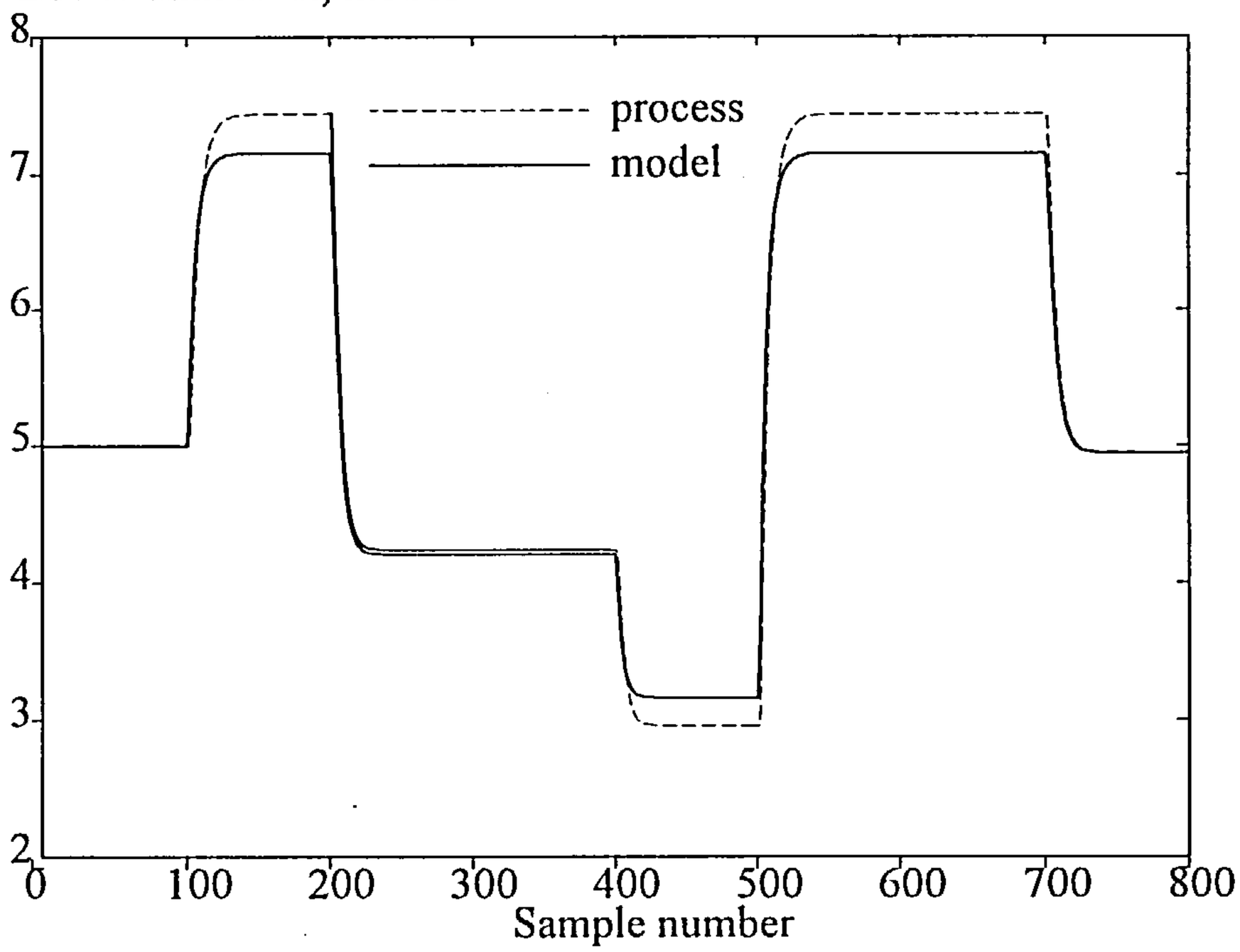


Figure 3.12 Response of SNDM network tested as a predictor on a set of step input signals

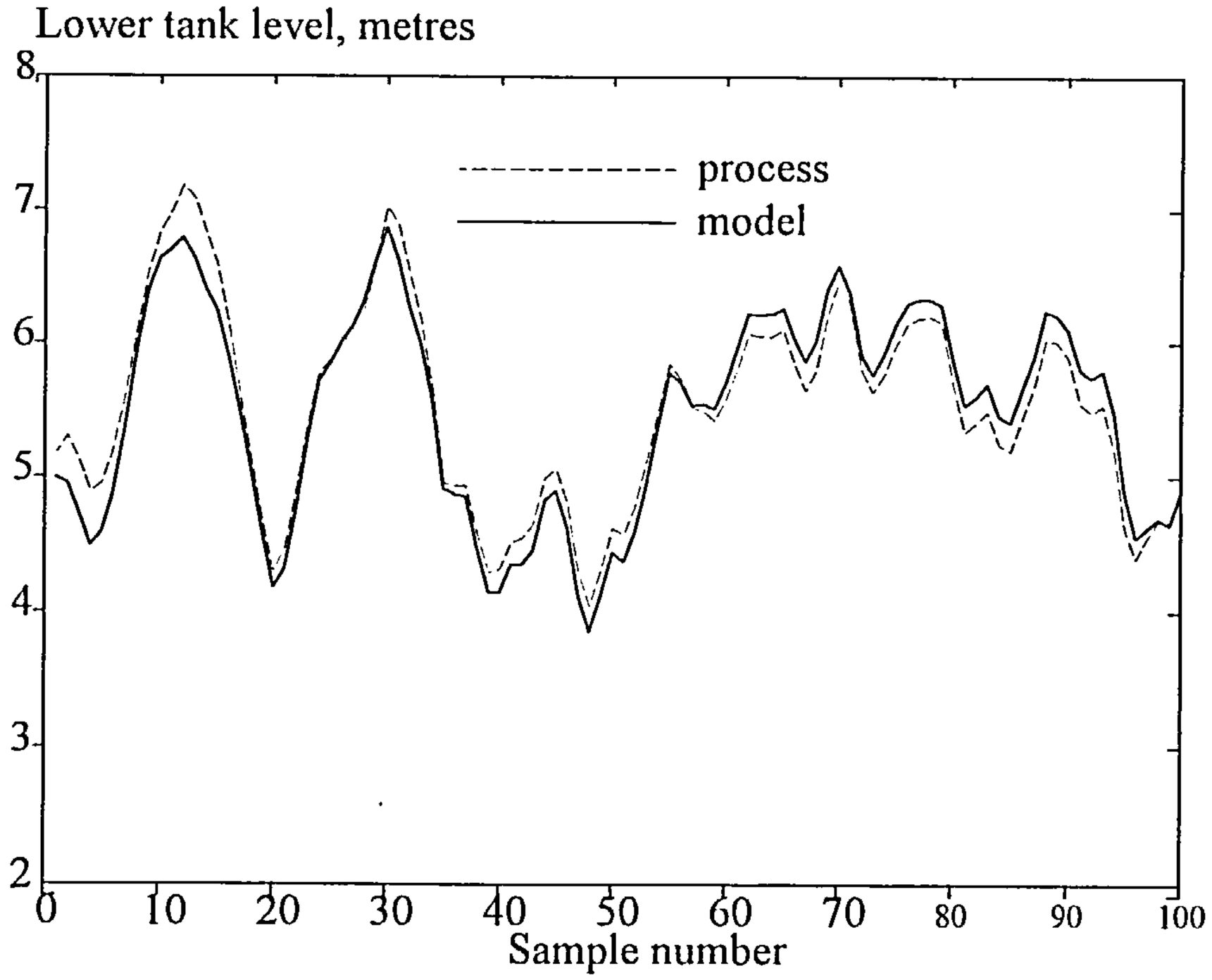


Figure 3.13 Response of SNDM network tested as a model on a random amplitude signal

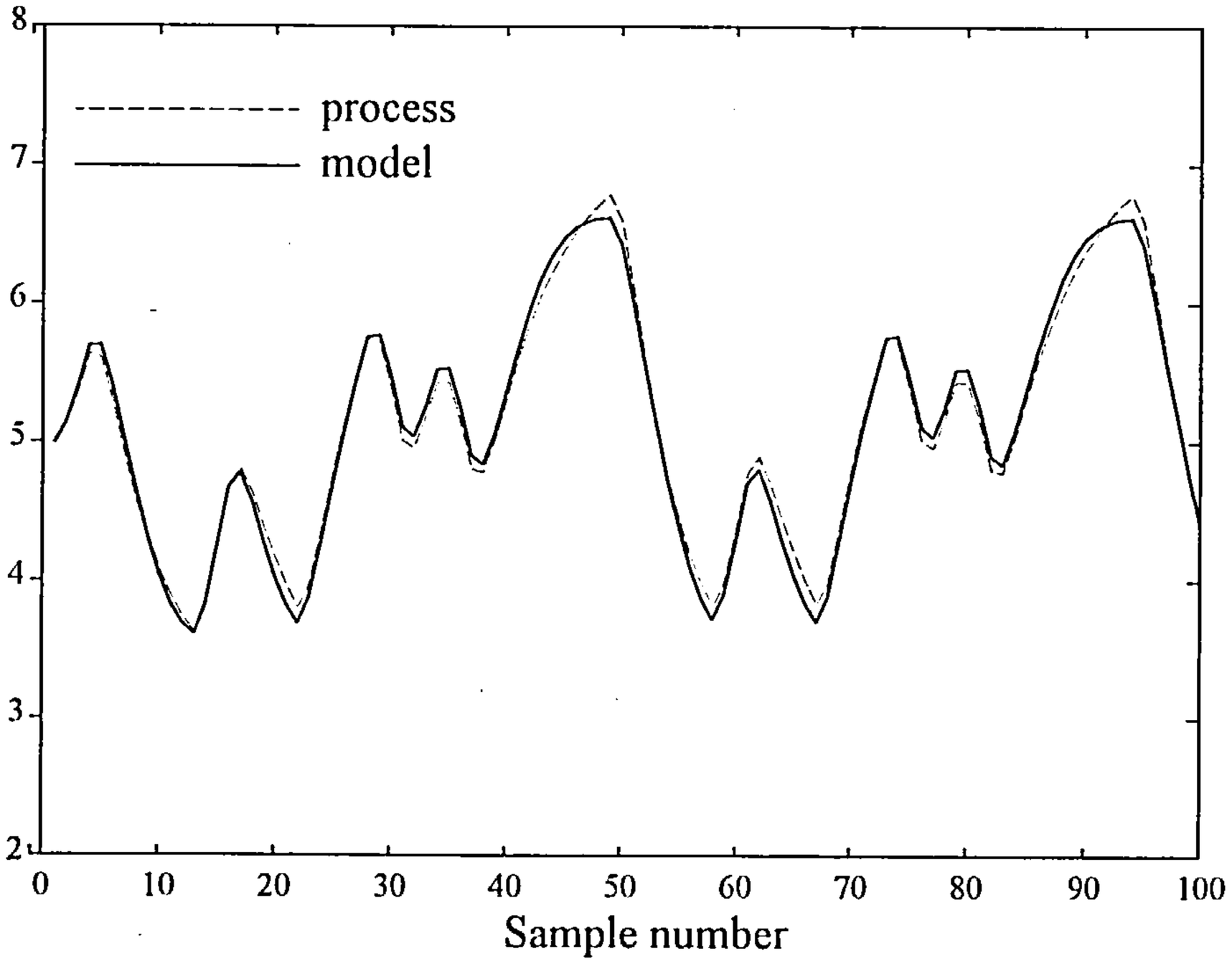


Figure 3.14 Response of SNDM network tested as a model on PRBS signal

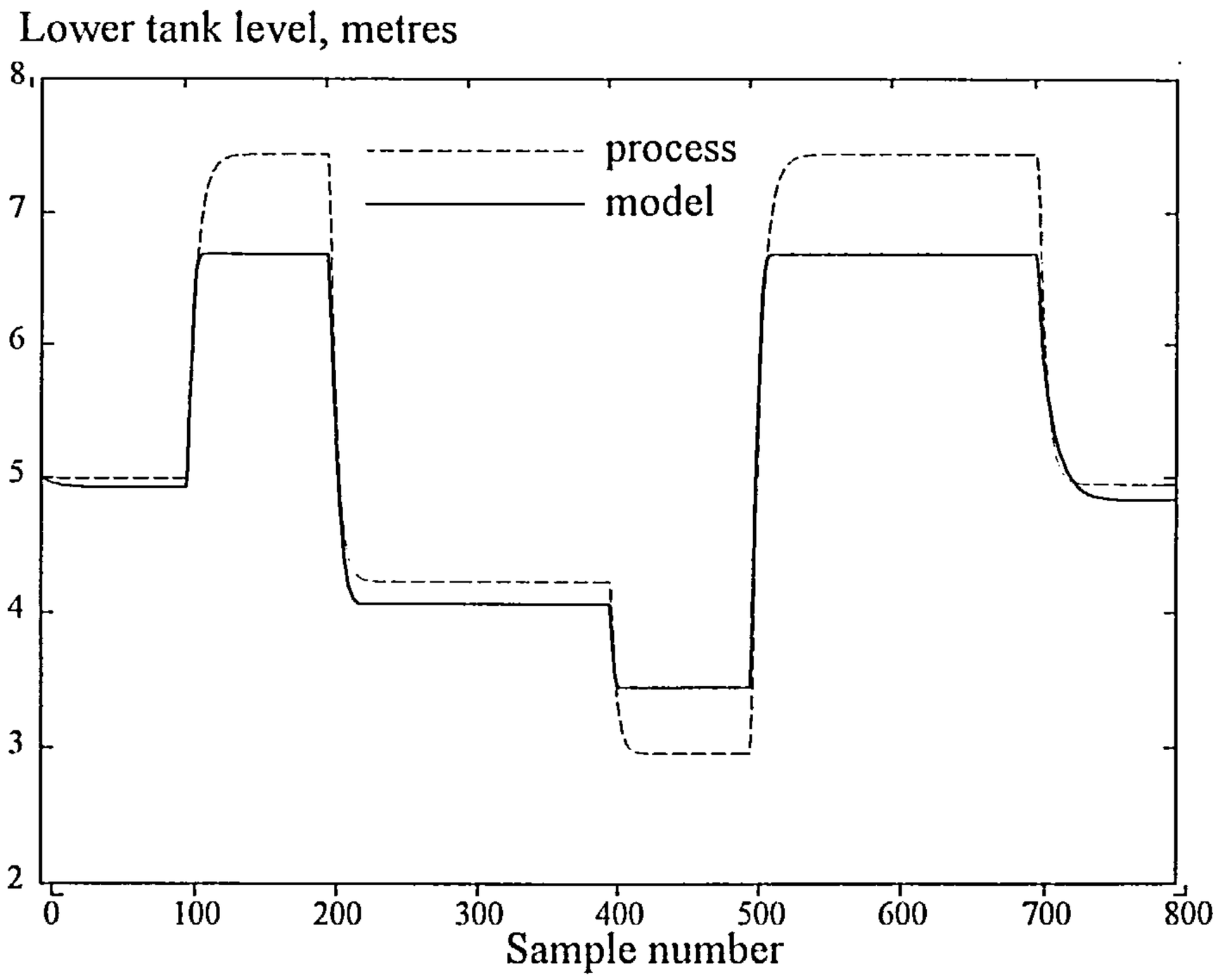


Figure 3.15 Response of SNDM network tested as a model on a set of step Signals

3.4.2 FPMODEL1 using SEDM

The same 1000 input-output data points introduced in section 3.4.1 were again employed, the difference being that the data, to be presented to the neural network, was spread over 6 nodes using the SEDM technique.

3.4.2.1 Neural network structure

As for FPMODEL1 the number of input nodes used was four in order to cater for the two previous inputs and outputs from the process being used. One output node was required and since each data value is spread over six nodes the resultant network had 24 input nodes and six output nodes. Six nodes were used in the hidden layer of the network. The overall neural network topology is shown in Table 3.3.

Number of input nodes	Number off hidden nodes	Number of output nodes
24	6	6

Table 3.3: Neural network topology for FPMODEL1 using SEDM

The network had a total of 192 weights (including bias weights) to be estimated and the 1000 input-output data points used for training was adequate.

3.4.2.2 Training the neural network

The learning rate and momentum parameters were set to the same values used for the network trained using SNDM conditioning, Table 3.2. Figure 3.16 indicates that the optimum amount of training to give the network was that of 300 complete passes of the training data set, and network training was terminated at this point. Figure 3.16 also illustrates that if the network is tested in the predictor configuration during network training that the MSE continues to reduce as was illustrated for the SNDM network.

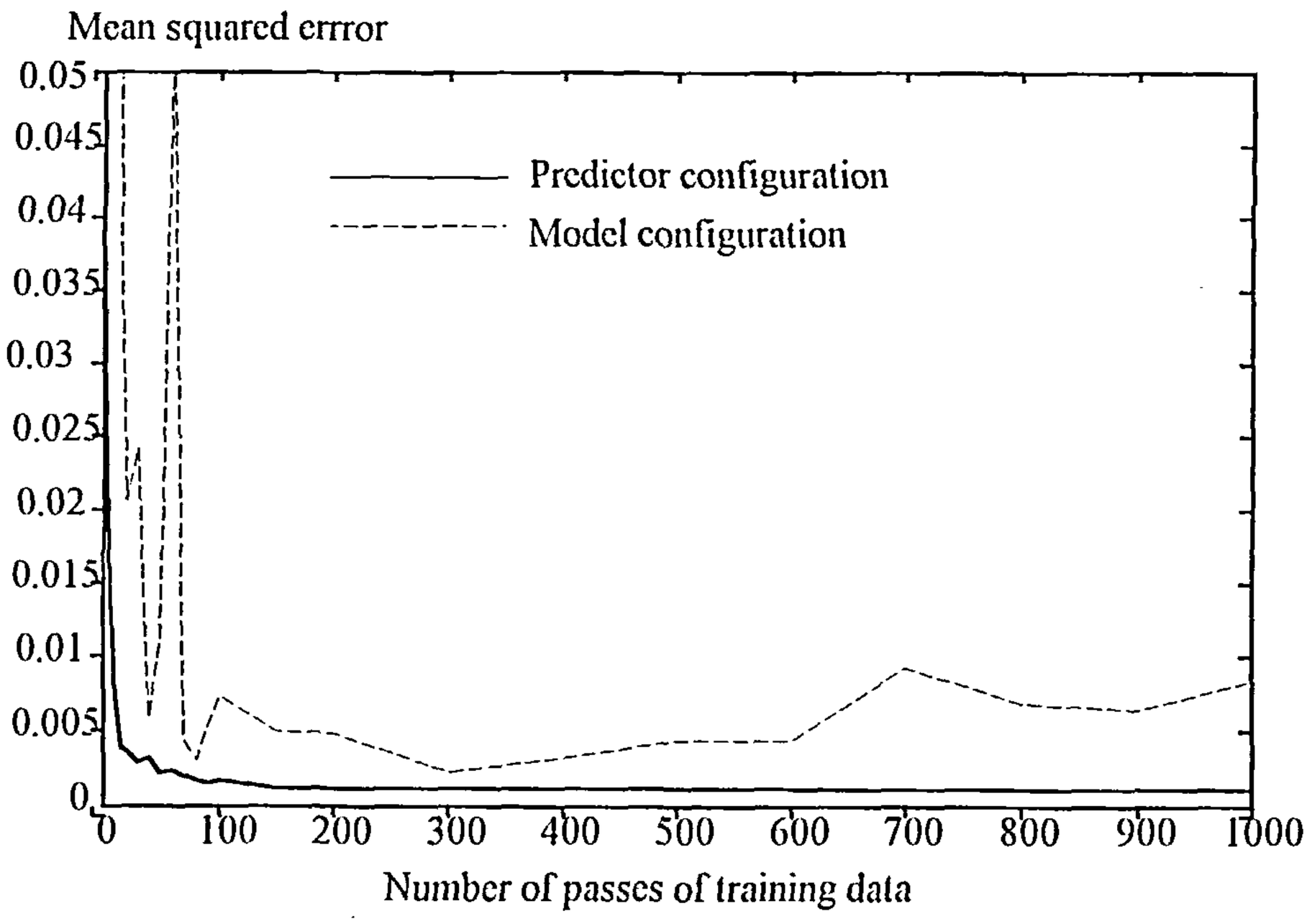


Figure 3.16 Termination of network training

3.4.2.3 Validation of the trained neural network

When tested on the same RAS that the network was trained on, the predictions of the neural network were very accurate as expected.

Figures 3.17 and 3.18 illustrate the accuracy of the neural network when tested on a different RAS and a PRBS signal respectively. In both cases the networks predictions accurately match the response of the simulated dual tank liquid level system, with respective MSE of 0.0012 for the RAS and 0.00096 for the PRBS.

The network's output for the set of step inputs is shown in Figure 3.19. The result is very similar to the one obtained for the SNDM network. Two steady state values have been learned accurately, but at steady state values at the top and bottom of the waveform there is slight errors between the predicted and actual process steady state levels, although these errors are smaller than was observed for the SNDM

network. Figure 3.20 indicates that in the model structure and tested on a new RAS the predictions are very good.

Figures 3.21 and 3.22 illustrate that the response of the network in the model structure, when tested on the PRBS and set of steps, is virtually the same as when the network was tested in the predictor structure with a high degree of accuracy obtained. The steady state offsets for the set of steps Figure 3.22, are much smaller than that which was observed for the SNDM network in the model configuration.

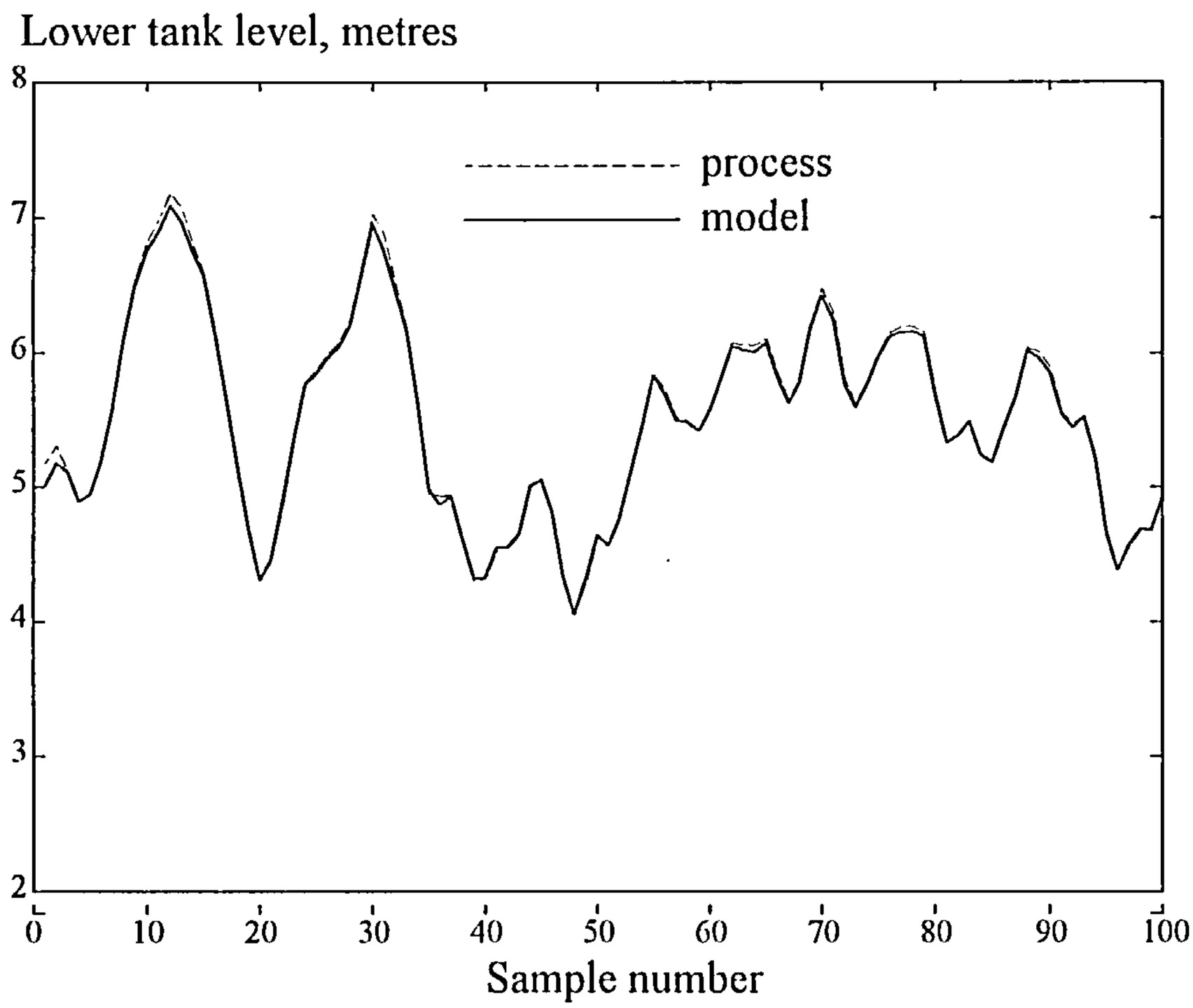


Figure 3.17 Response of SEDM network tested as a predictor on a random amplitude signal

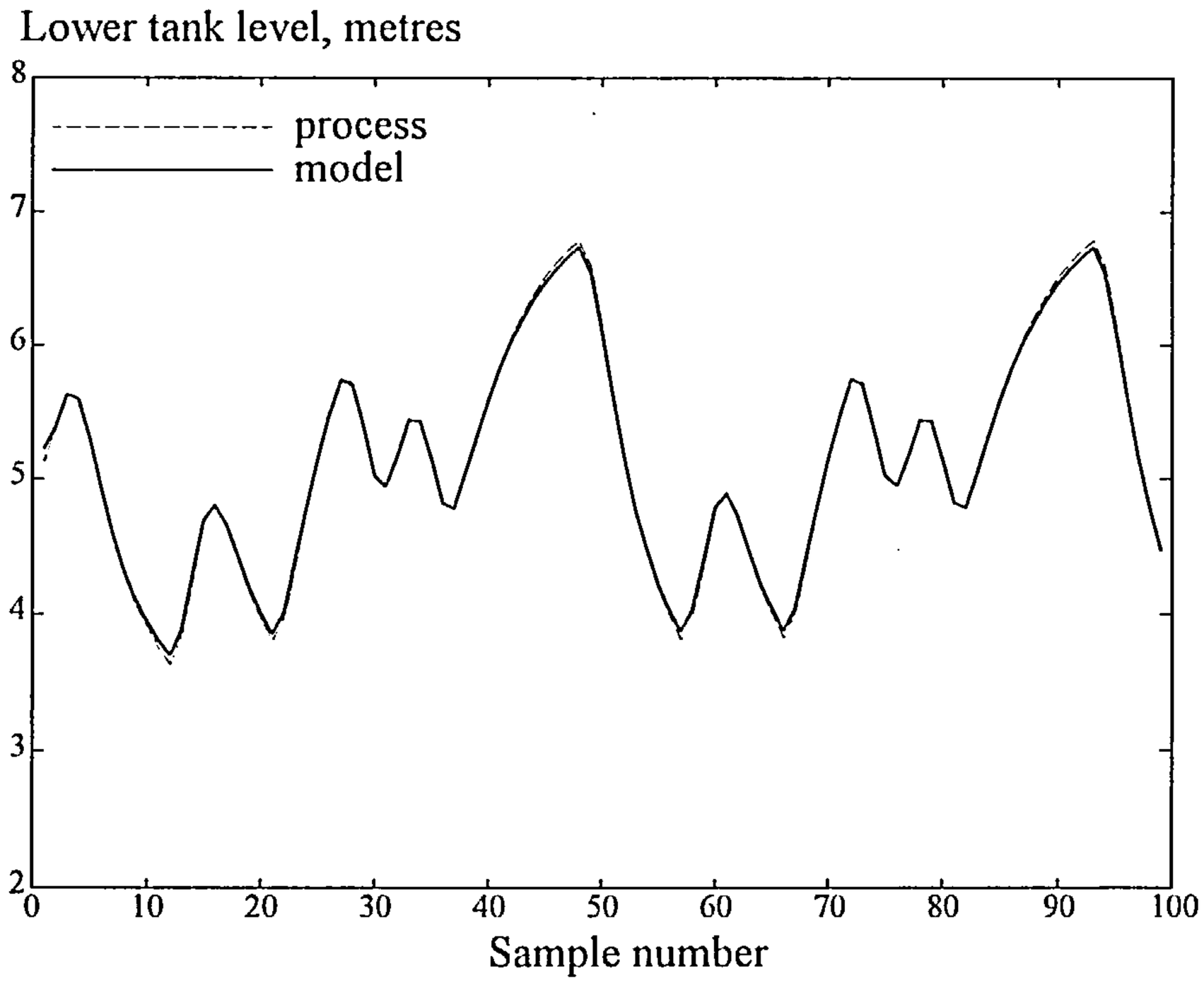


Figure 3.18 Response of SEDM network tested as a predictor on PRBS signal

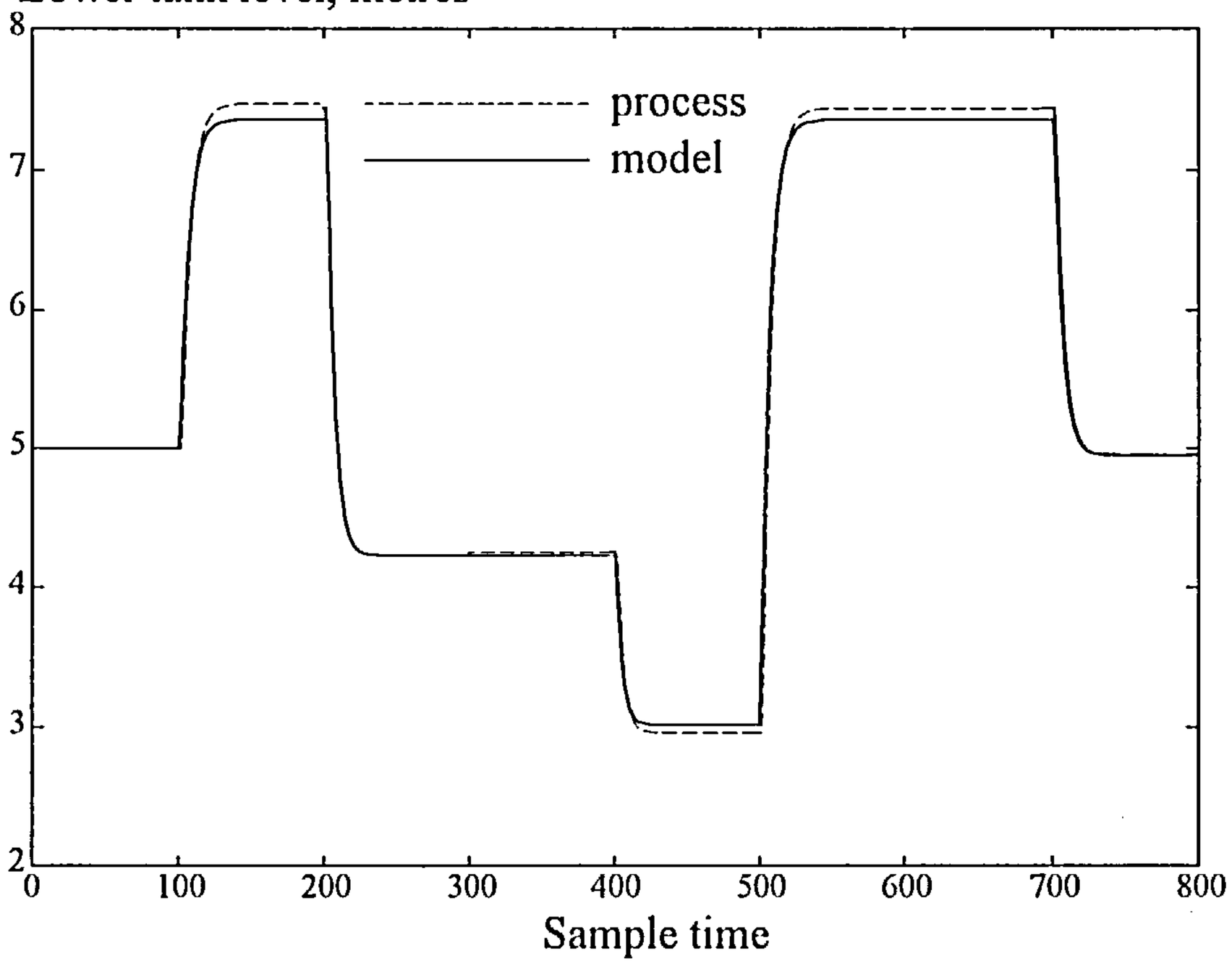


Figure 3.19 Response of SEDM network tested as a predictor on a set of step input signals

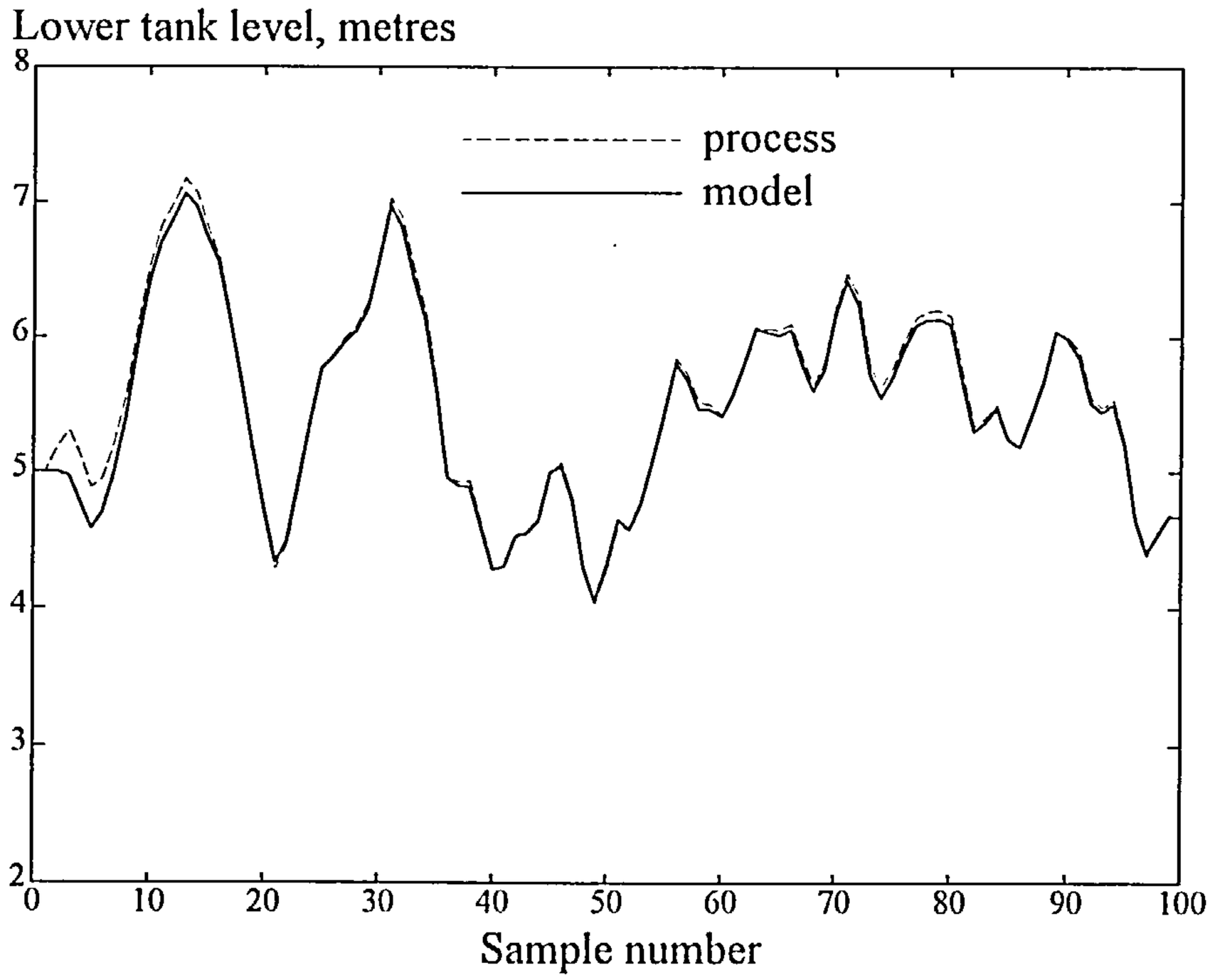


Figure 3.20 Response of SEDM network tested as a model on a random amplitude signal

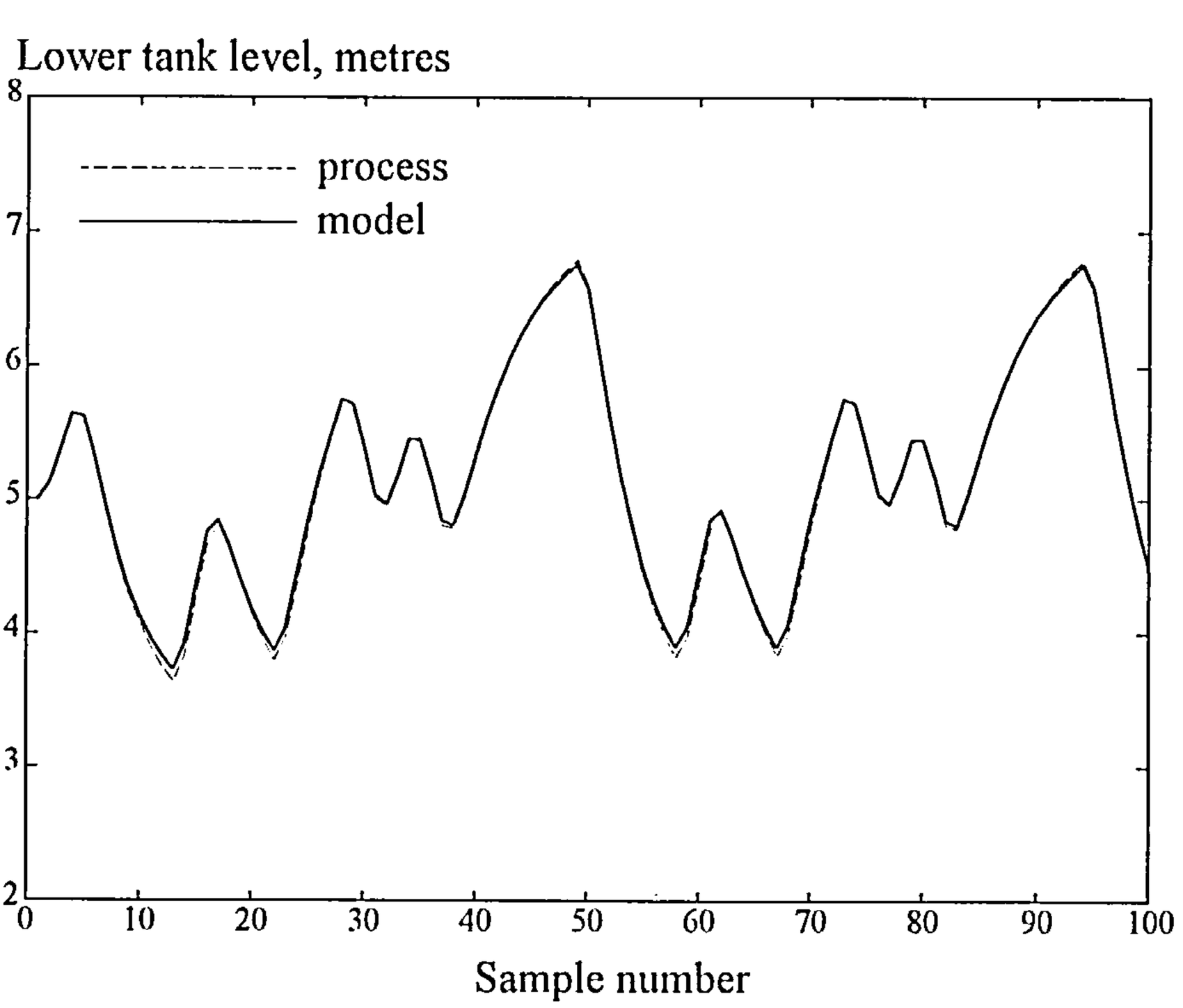


Figure 3.21 Response of SEDM network tested as a model on a PRBS signal

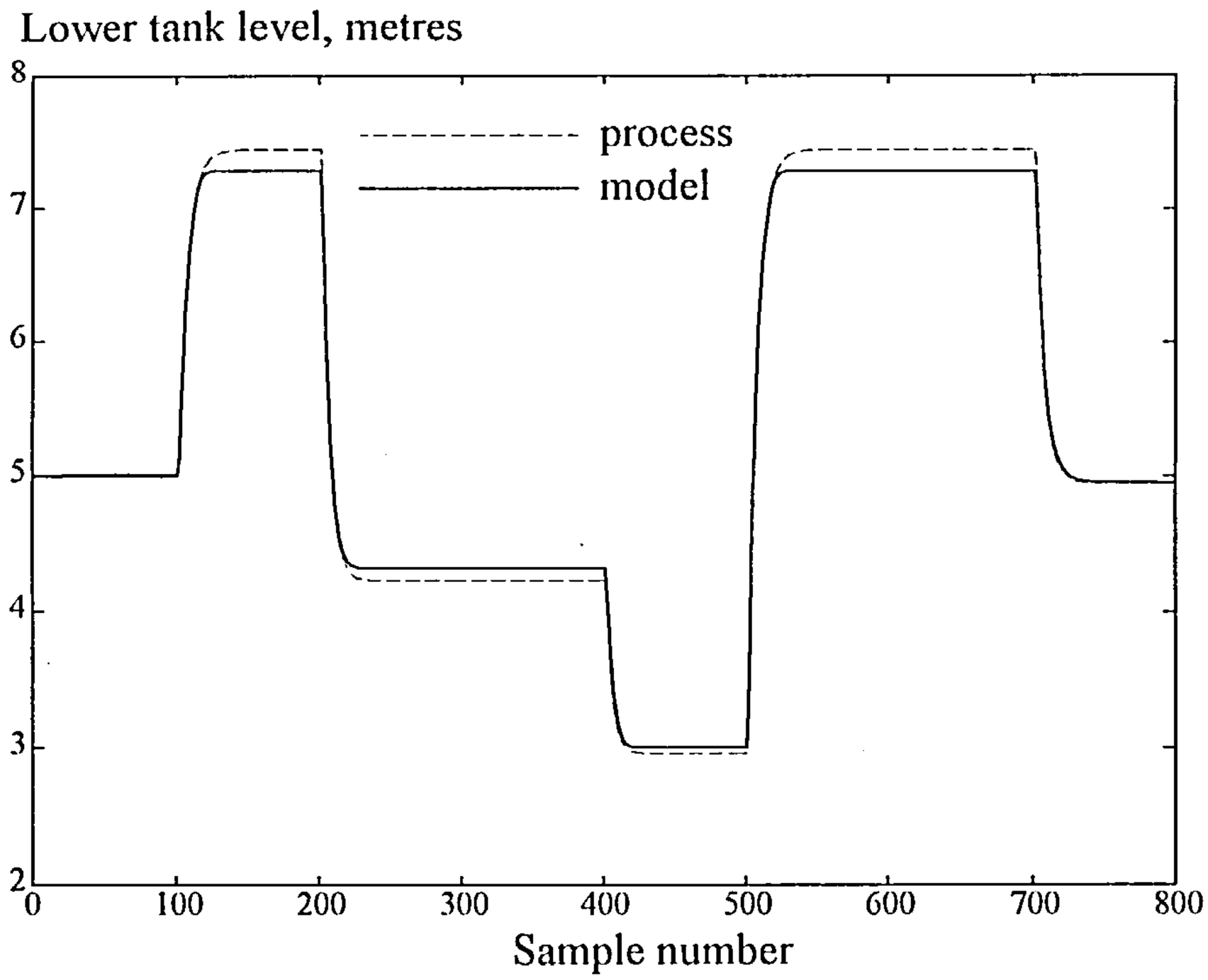


Figure 3.22 Response of SEDM network tested as a model on a set of step inputs

3.4.3 Summary

The ability of a neural network to accurately capture the dynamics of the dual tank liquid level process has been investigated. Two forms of conditioning the data presented to the neural network namely SNDM and SEDM were investigated, and the ability of the trained neural networks to predict the process output in both the predictor and recursive model configurations were illustrated. It is concluded from the above that when the neural networks are configured as a predictor, both the SNDM and SEDM techniques result in networks that are accurate at predicting the process performance. However, when configured as recursive models the neural network trained using the SNDM fails to be able to accurately predict the process performance, whereas the network trained using the SEDM technique has comparable results to when it was tested in the predictor structure. Since the use of

the neural network in the model configuration is highly desirable, then using the SEDM technique is advantageous. Although network complexity is increased using this method of coding, the network required fewer hidden nodes than the SNDM network. Also, as shown in Figures 3.9 and 3.16 the network's MSE using the SEDM method is much smaller than that of the SNDM network, and that the network had converged in 300 passes compared to 700 passes for the SNDM network. In view of the above points all future process identification carried out uses the SEDM method.

The mathematical model of the real laboratory scale process (FPMODEL2) was then simulated in ACSL to obtain training and testing data sets, and the procedures described in the above sections to obtain an accurate neural network process model using SEDM were applied.

3.4.4 FPMODEL2 using SEDM

FPMODEL2 was implemented into an ACSL program and simulation studies were carried out. This model was simulated since it has been demonstrated that it is a close match to the real process (section 3.2.2.1), and hence if a neural network is capable of modelling FPMODEL2 then transfer to the real process should be smooth. The RAS used to excite the process was generated in the same way as in the previous simulations, with random values between -50 and +50 added to the steady state valve input of 150. This enabled the digital number to the valve to range between 100 (fully closed) and 200 (fully open). The period between successive updates of the RAS was set at one sample.

3.4.4.1 Neural network structure

The neural network structure used in section 3.4.2.1 was retained, Table 3.3 shows the number of processing units used.

3.4.4.2 Training the neural network

Data presented to the neural network was conditioned using the SEDM method and the parameters of the back propagation algorithm were set at 0.9 for the learning rate and 0.6 for the momentum, and as training proceeded these values were reduced as shown in Table 3.2. The network's generalisation started to deteriorate after approximately 350 presentations of 'data set one' and network training was terminated at this point.

3.4.4.3 Validation of the trained neural network

The neural network was tested as before on the set of test signals listed in section 3.3.4, and the prediction performance in both the model and one-step-ahead predictor configurations was noted. Figures 3.23 to 3.25 illustrate the accuracy of the neural network when tested in the predictor configuration. In each of the three figures the neural network predictions for the RAS, PRBS and step input are in very accurate agreement with the response of the process output.

Configured in the recursive model structure the neural network's predictions of the process output are illustrated in Figures 3.26 and 3.27 for the RAS and PRBS respectively and Figure 3.28 for the step input. The accuracy of the predictions is not as accurate as when configured in the predictor structure as indicated by the MSE of each of the responses, but is significantly higher than would be obtainable using a SNDM network.

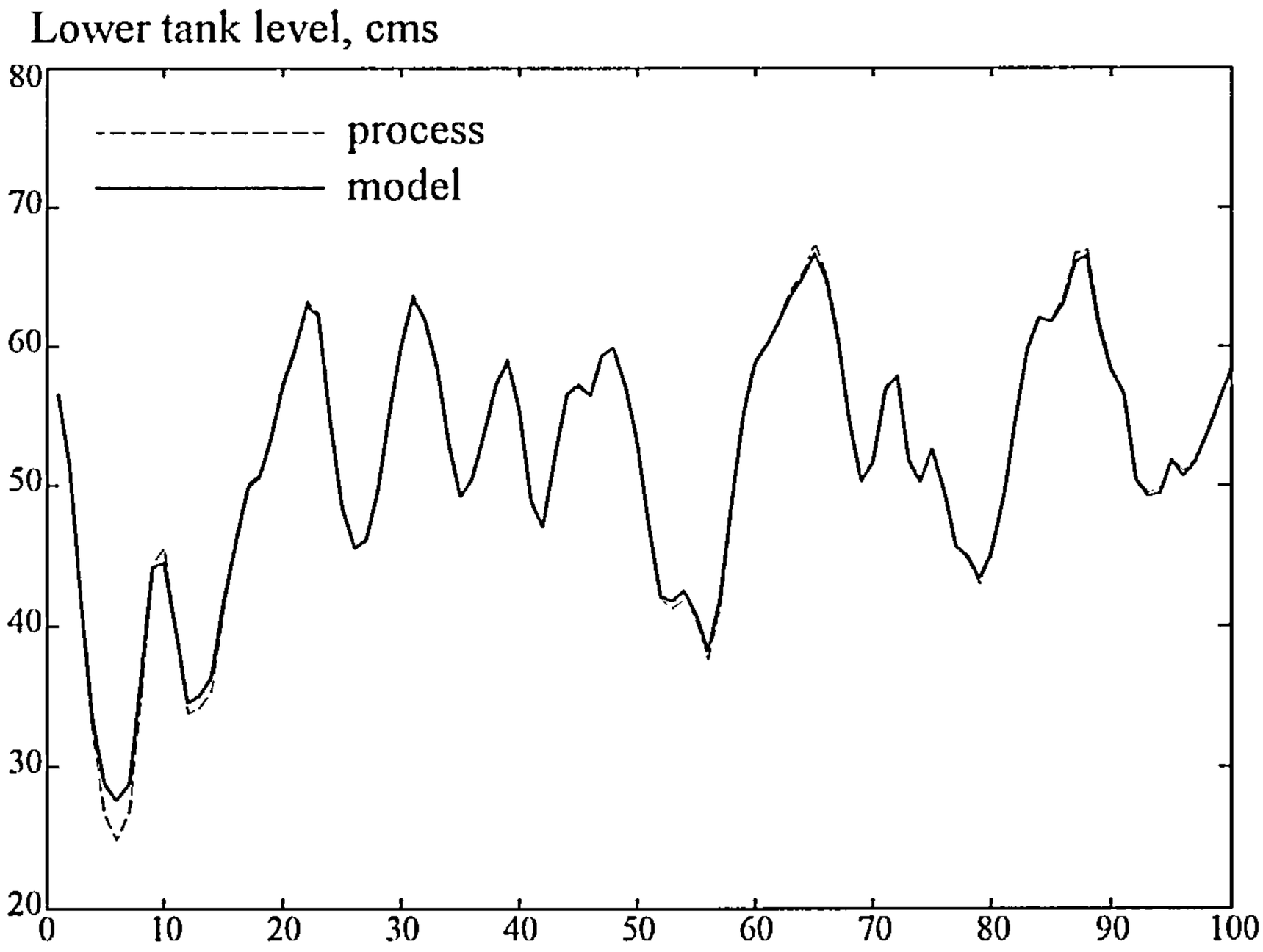


Figure 3.23 Response of SEDM network tested as a predictor on a random amplitude signal

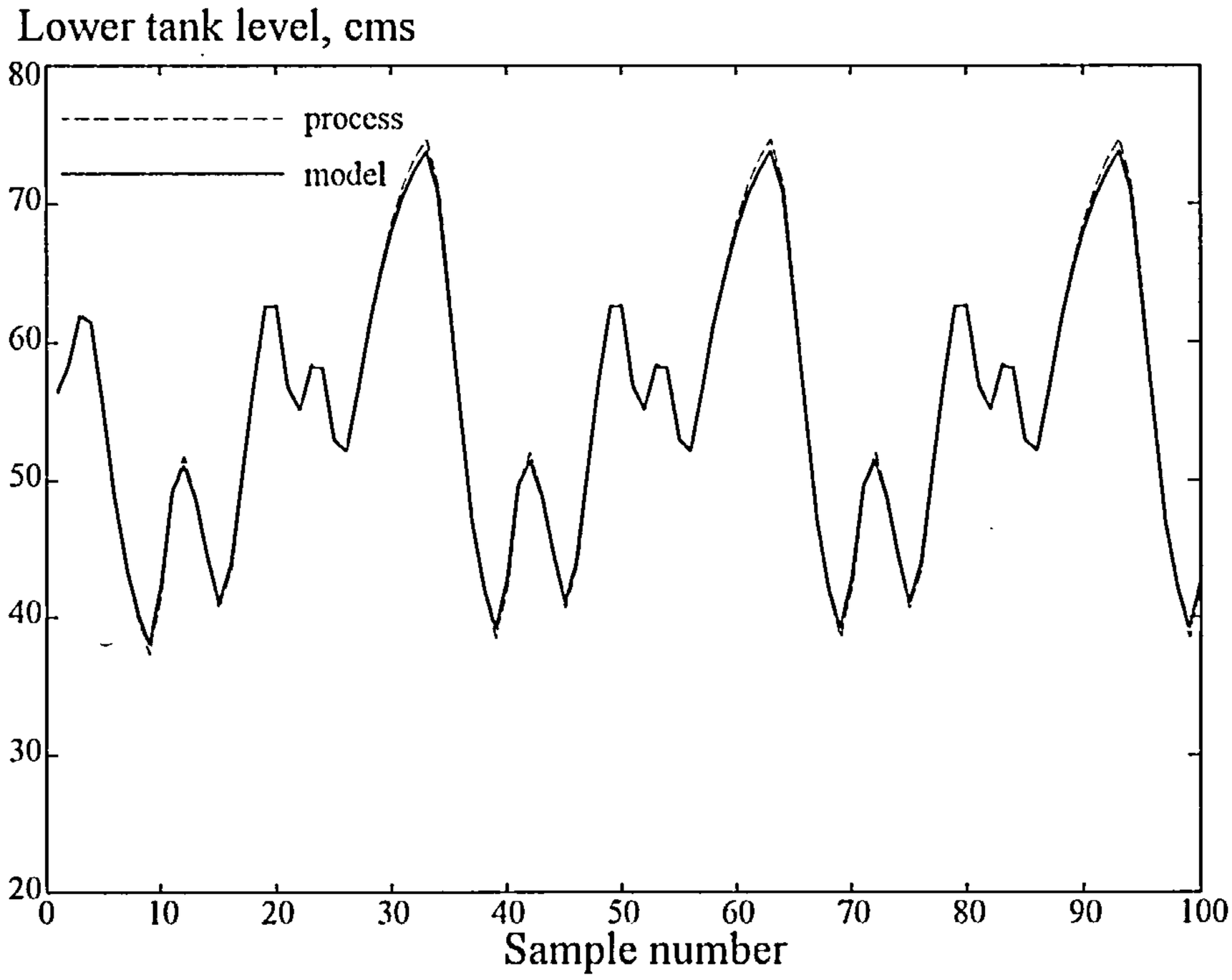


Figure 3.24 Response of SEDM network tested as a predictor on a PRBS signal

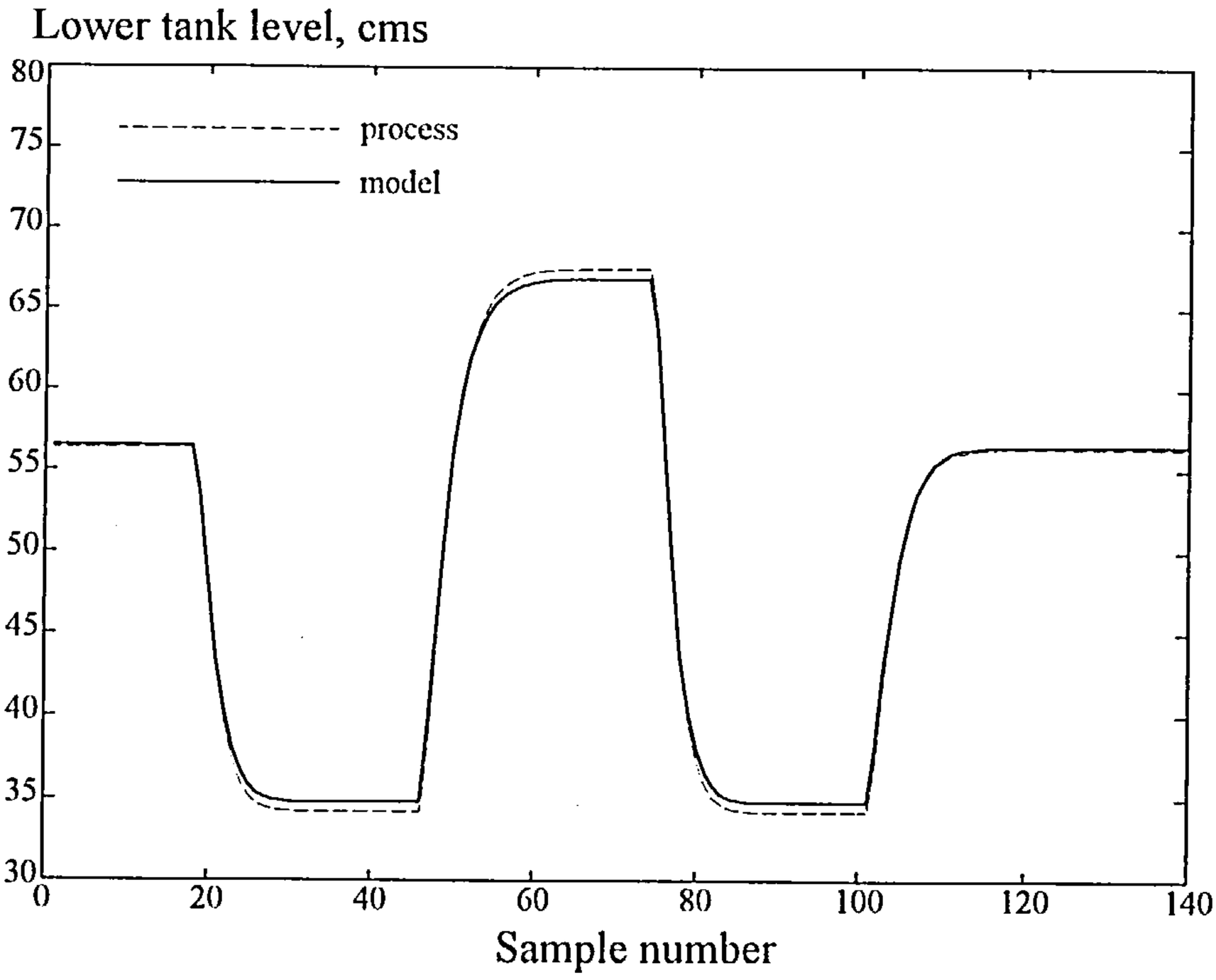


Figure 3.25 Response of SEDM network tested as a predictor on a set of step input signals

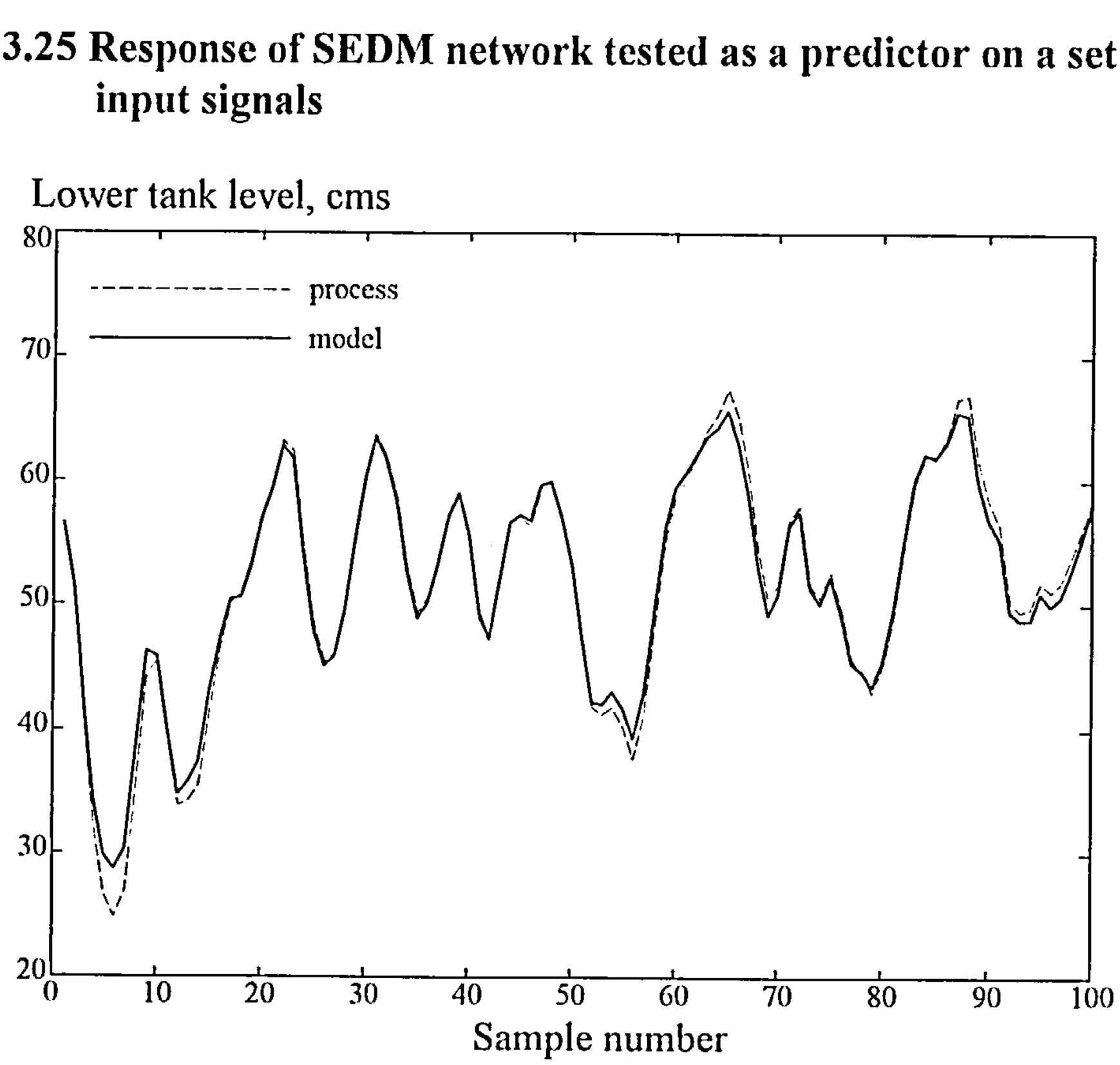


Figure 3.26 Response of SEDM network tested as a model on a random amplitude signal

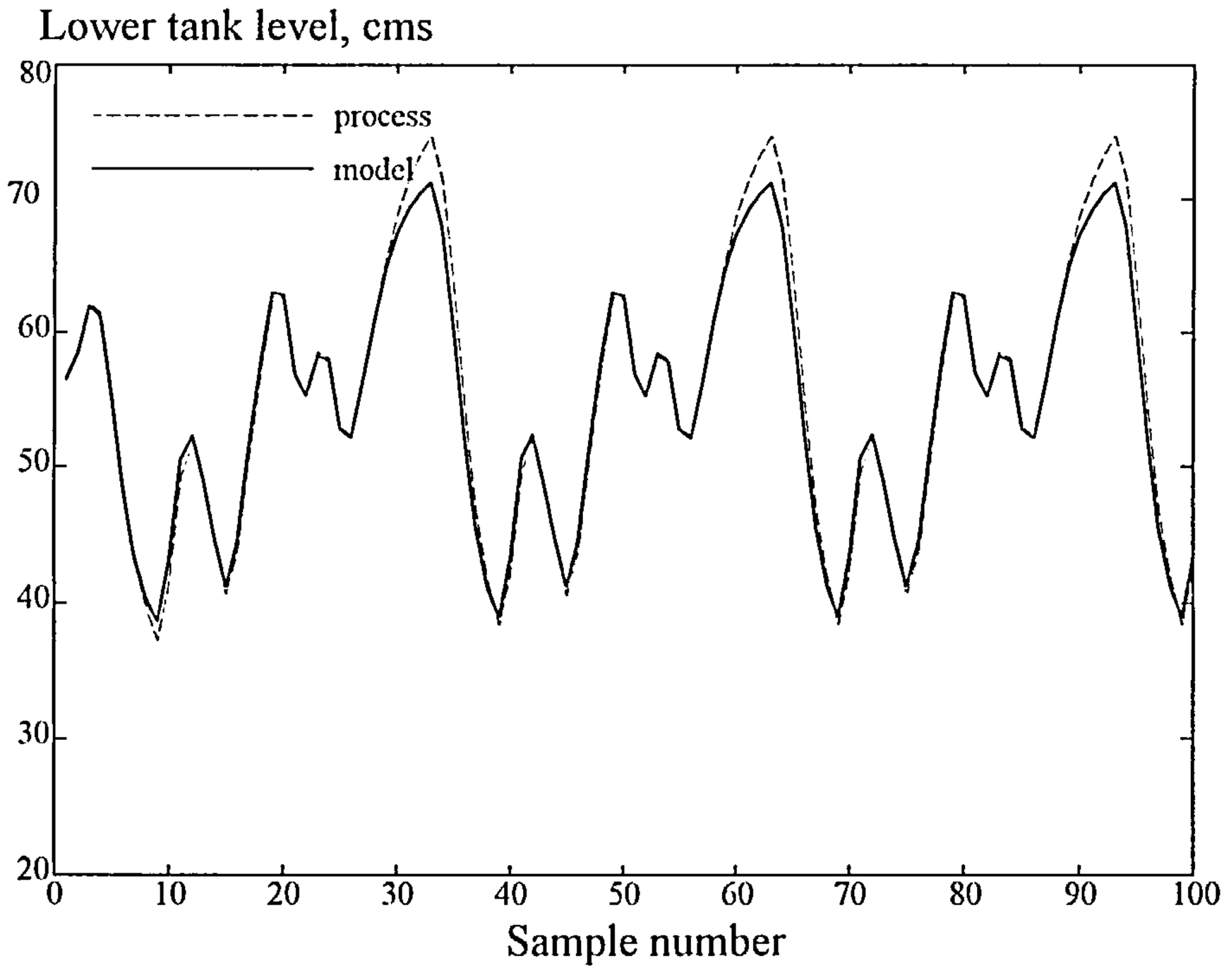


Figure 3.27 Response of SEDM network tested as a model on a PRBS signal

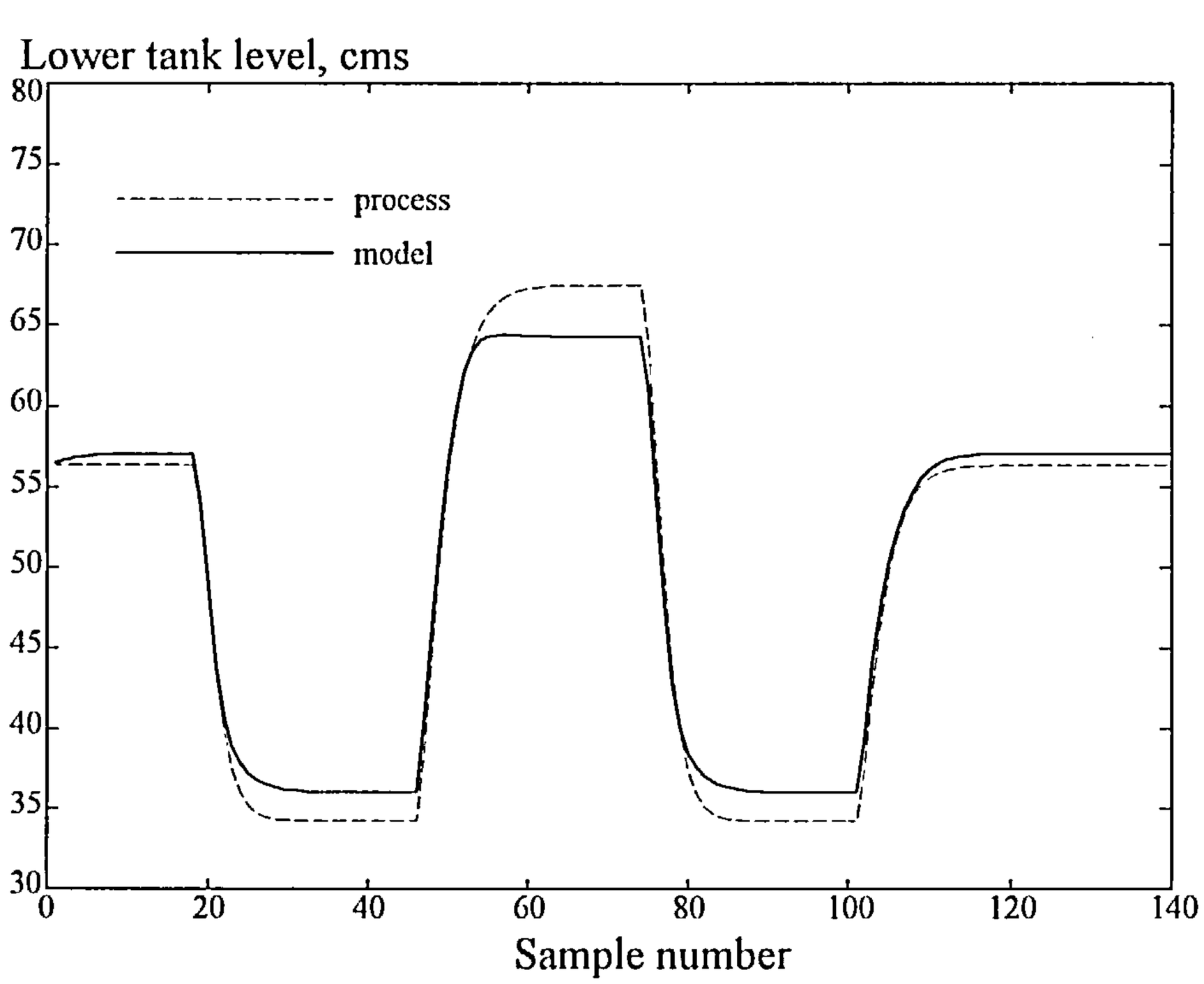


Figure 3.28 Response of SEDM network tested as a model on a set of step input signals

3.5 PROCESS SIMULATIONS WITH MEASUREMENT NOISE

The presence of simulated process noise added to the measurement of the height of liquid in tank 2 makes the problem of modelling the process more realistic, since in practice the measurements taken from the laboratory scale liquid level process, and in general from most processes will contain noise as an integral part of the measured value.

The simulated noise added to the process output was Gaussian in nature and was generated using a function available in ACSL. The signal to noise ratio (SNR), was calculated using equation 3.9

$$SNR = 20\log_{10}\frac{\sigma_{h2}}{\sigma_e} dBs \qquad ...(3.9)$$

where σ_{h2} and σ_e are the standard deviations of the process output h_2 and the noise. The standard deviation of the noise was chosen so as to obtain a SNR of 20dBs. The studies were carried out on FPMODEL1 using SEDM coding of the data.

3.5.1 FPMODEL1 using SEDM

The simulation data used in the previous sections was generated again and at each sample period the process output was disturbed with noise. The standard deviation of the noise in both data sets was the same but the random noise values were made different by the initialisation of the seed of the random number generator available in ACSL.

3.5.1.1 Neural network structure and training

The topology of the SEDM network used in section 3.4.2.1., Table 3.3, was retained. Testing the neural networks generalisation capability at various stages of the training resulted in the neural network having the training data passed through it 600 times, Figure 3.29.

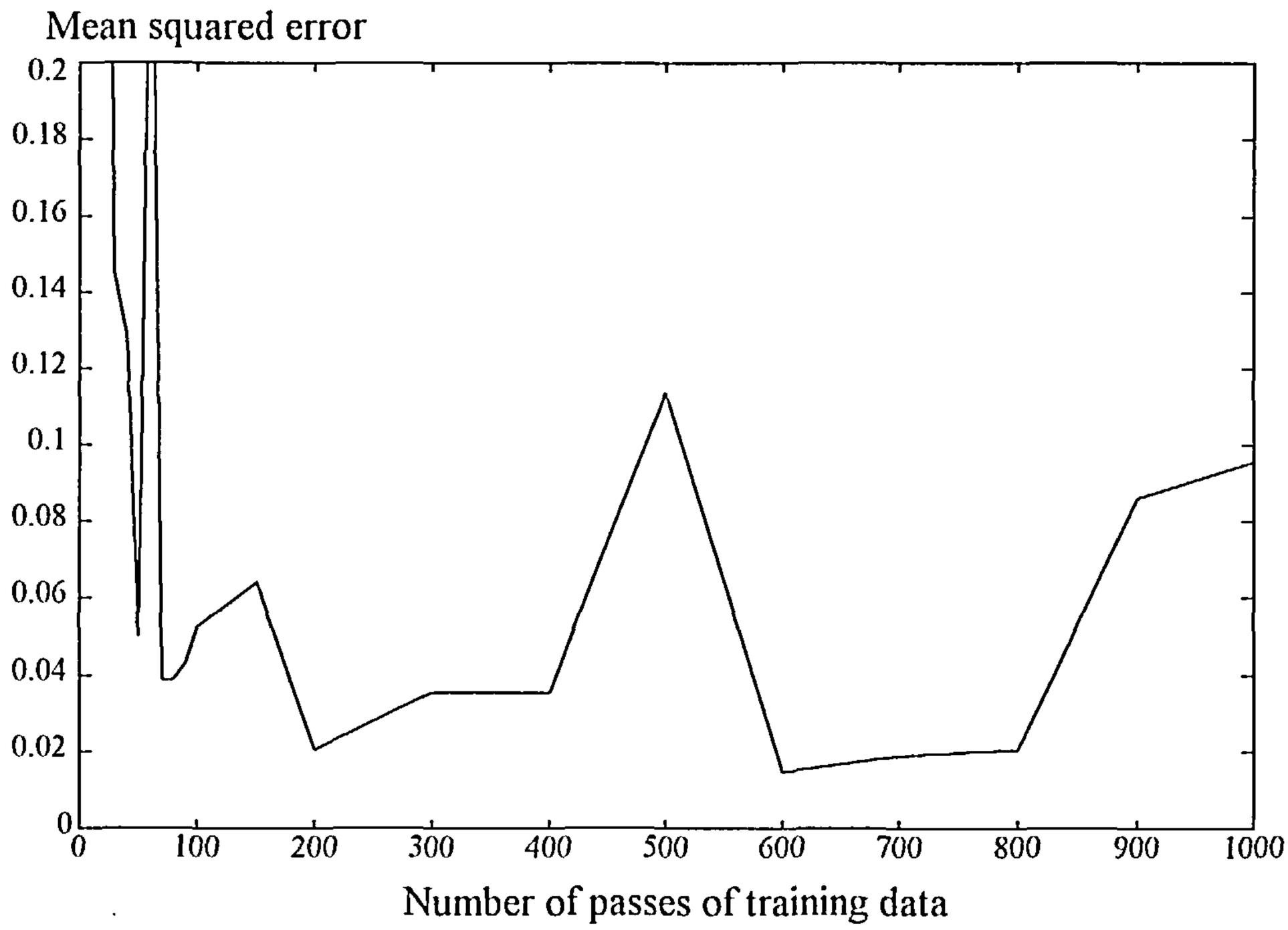


Figure 3.29 Termination of network training for network trained with noisy data

3.5.1.2 Validation of the trained neural network

The neural network trained with measurement noise superimposed onto the process output was evaluated in both the predictor and model configurations, on the test signals. In addition the frequency response of the neural network around linearised operating points was also investigated and compared to the frequency response of FPMODEL1 around the same operating points.

Figures 3.30 to 3.32 show the response of the neural network compared to that of the true process output (the process output without the noise superimposed), when in the predictor structure. The predictions of the neural network are slightly less accurate than the neural network trained without noise, (section 3.4.2), but are still acceptable.

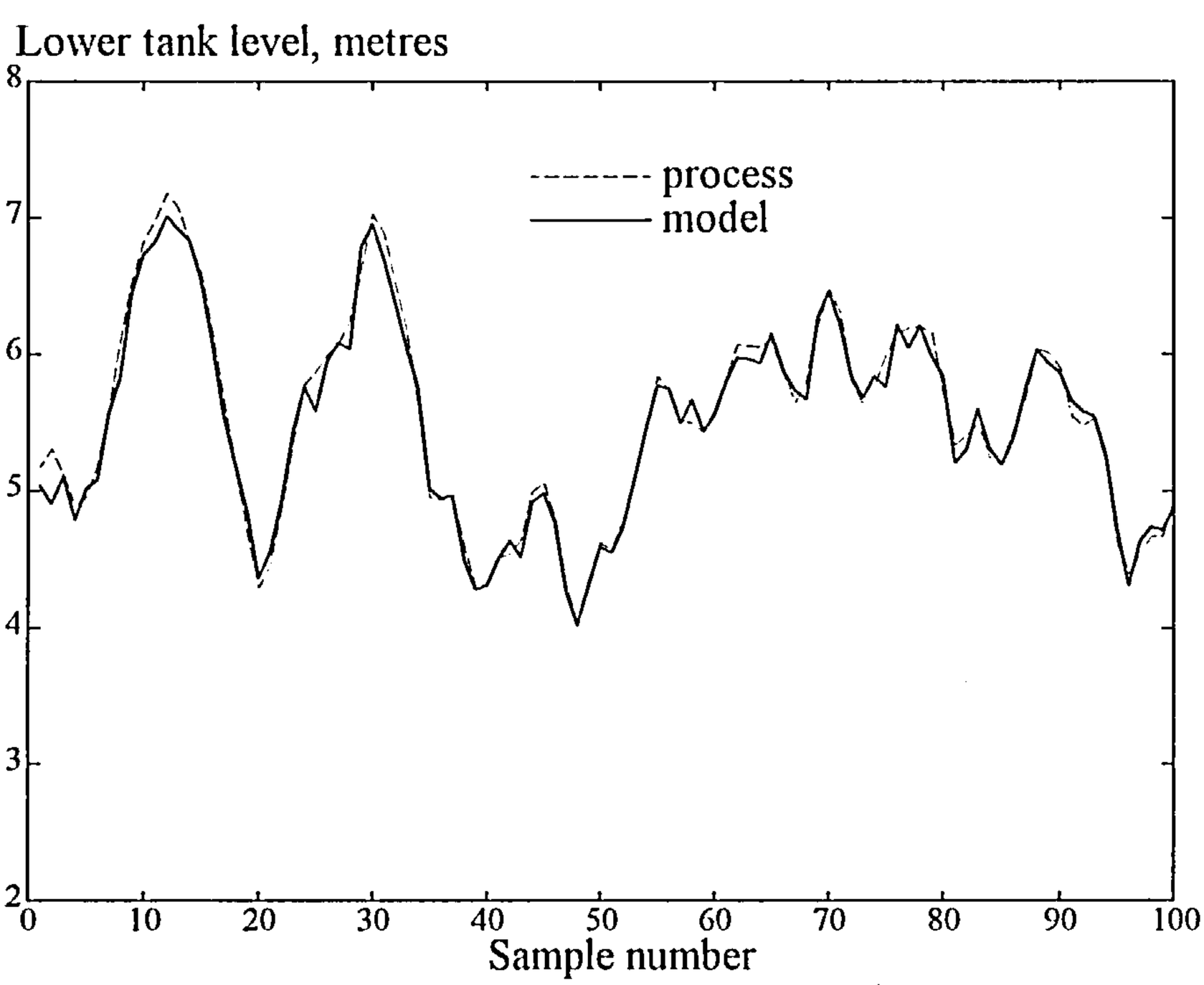


Figure 3.30 Response of SEDM network tested as a predictor on a random amplitude signal

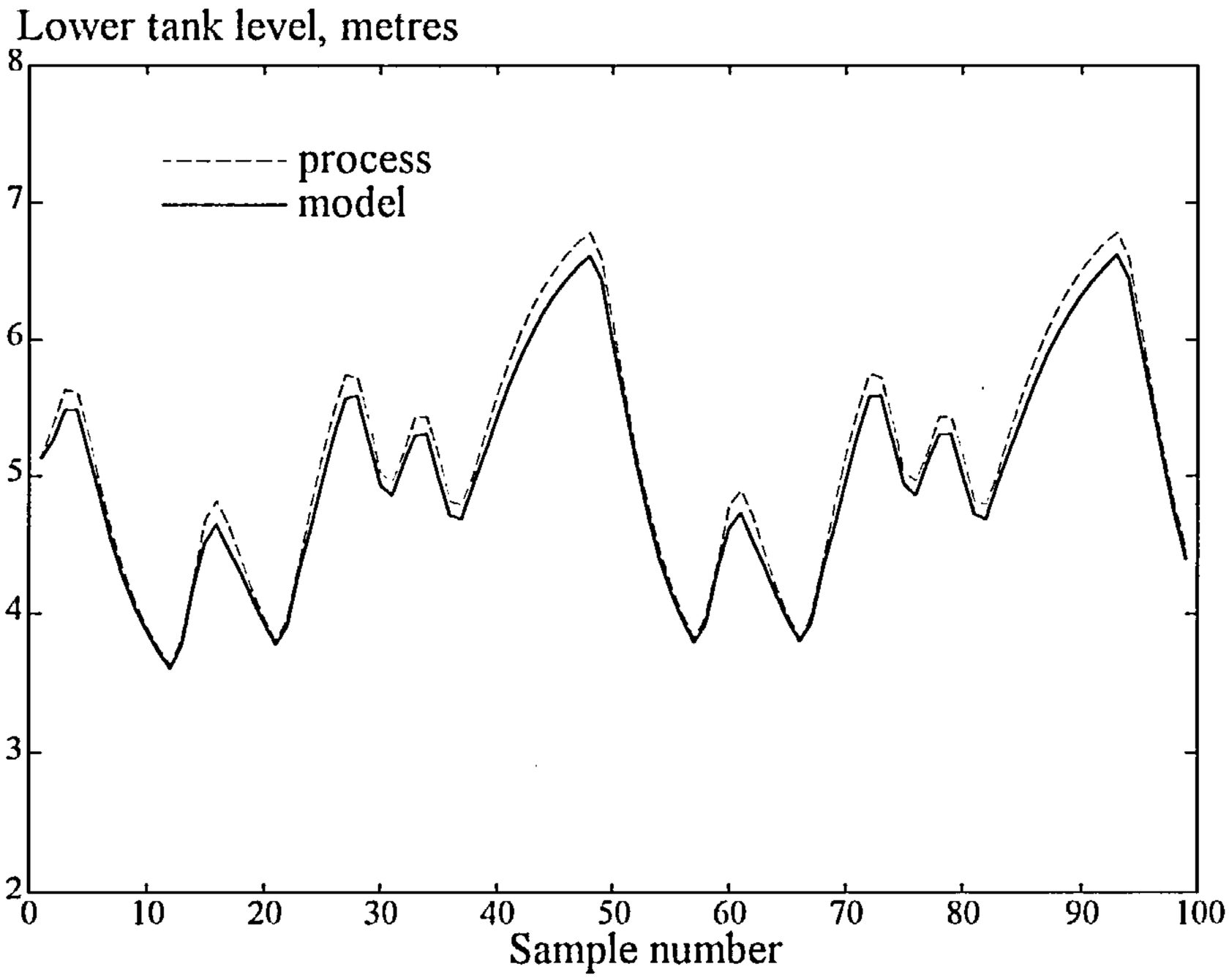


Figure 3.31 Response of SEDM network tested as a predictor on a PRBS signal

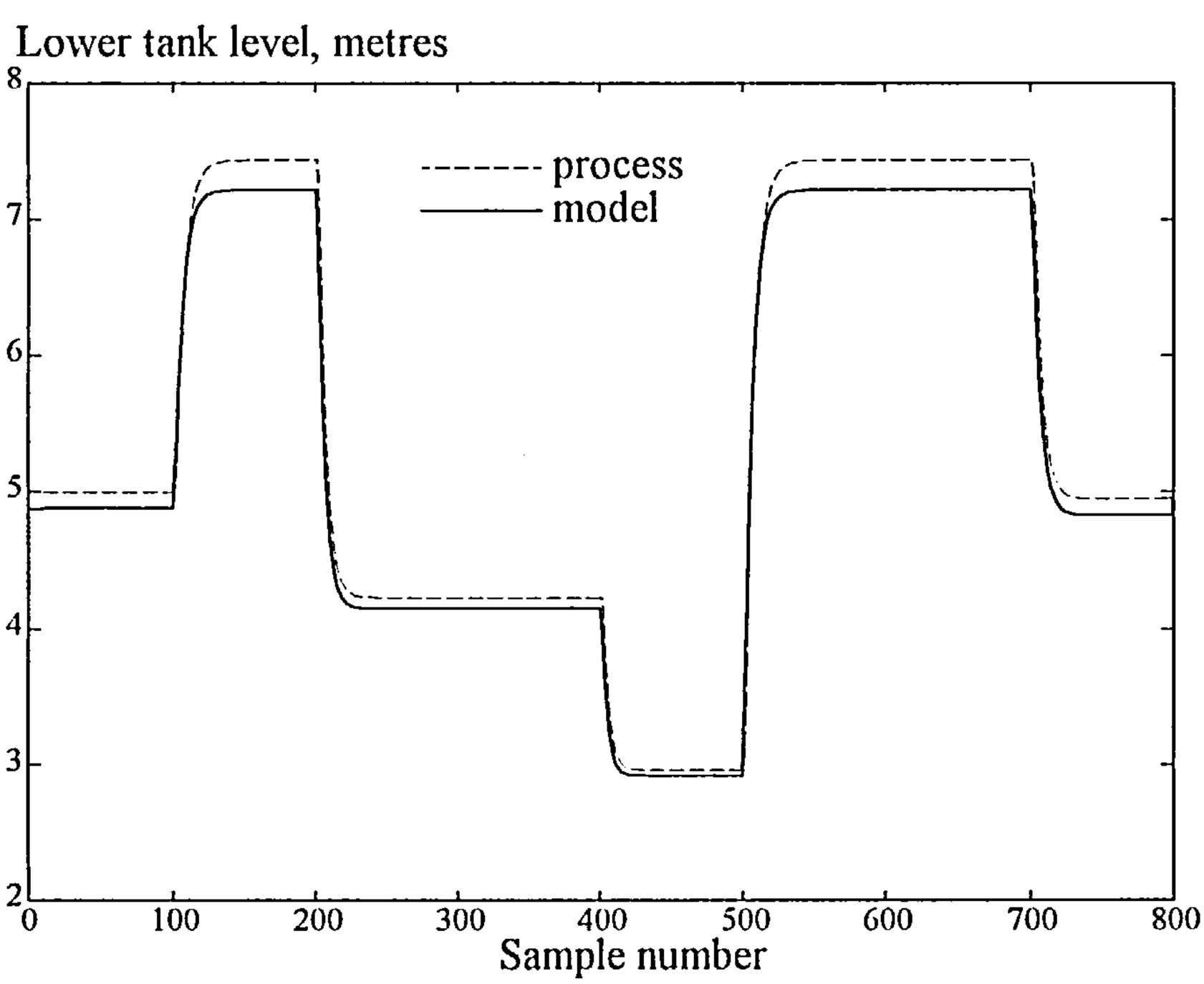


Figure 3.32 Response of SEDM network tested as a predictor on a set of step input signals

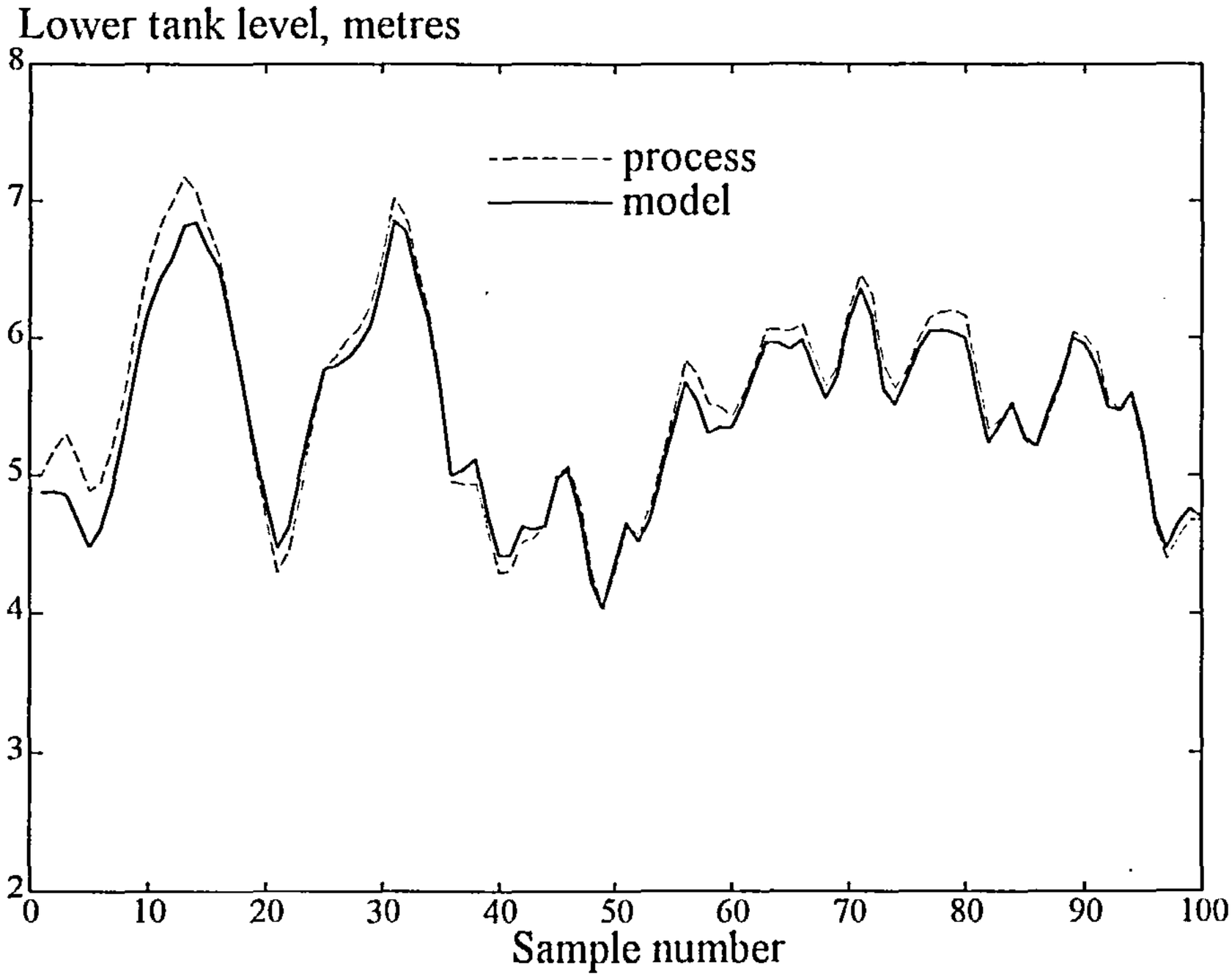


Figure 3.33 Response of SEDM network tested as a model on a random amplitude signal

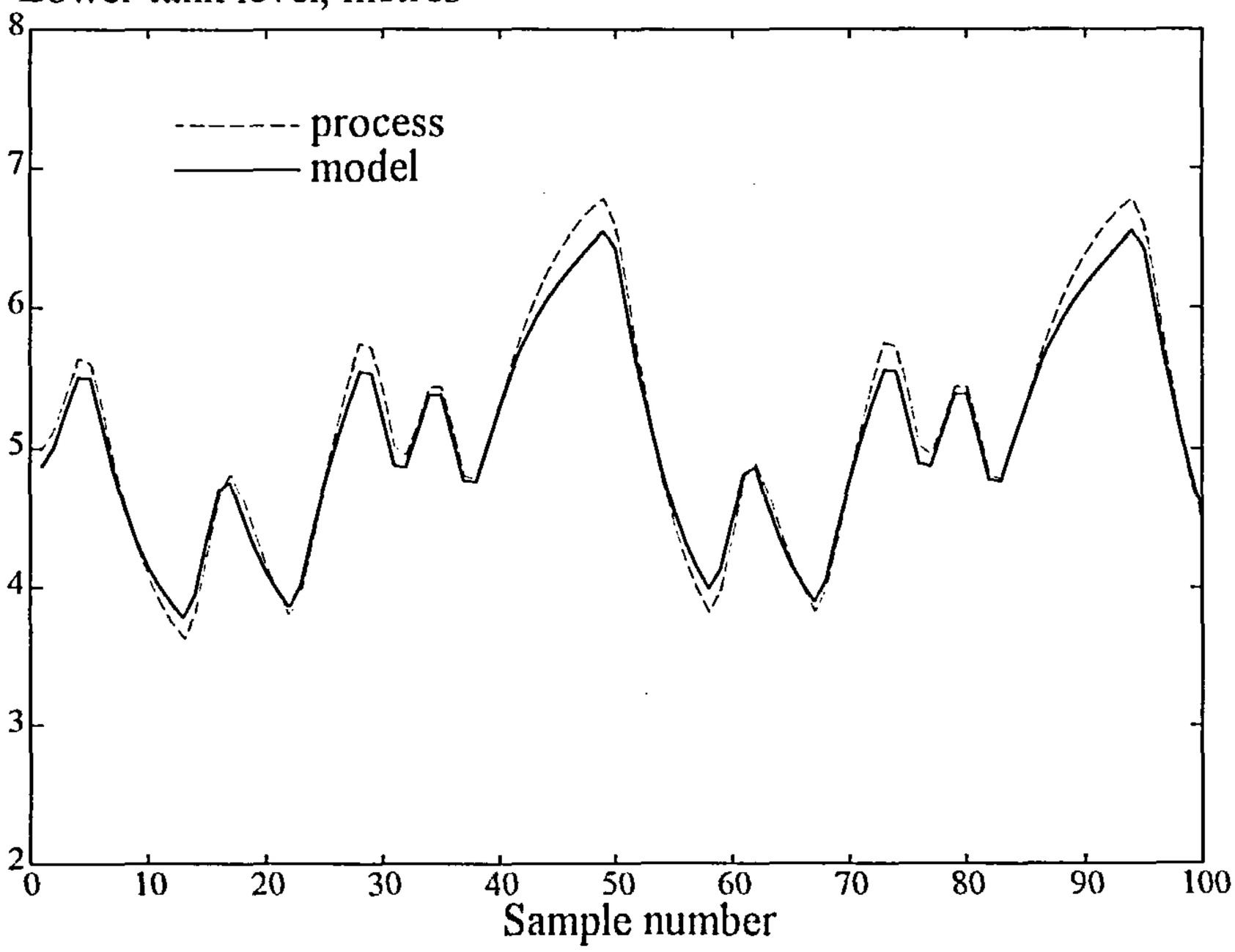


Figure 3.34 Response of SEDM network tested as a model on a PRBS signal

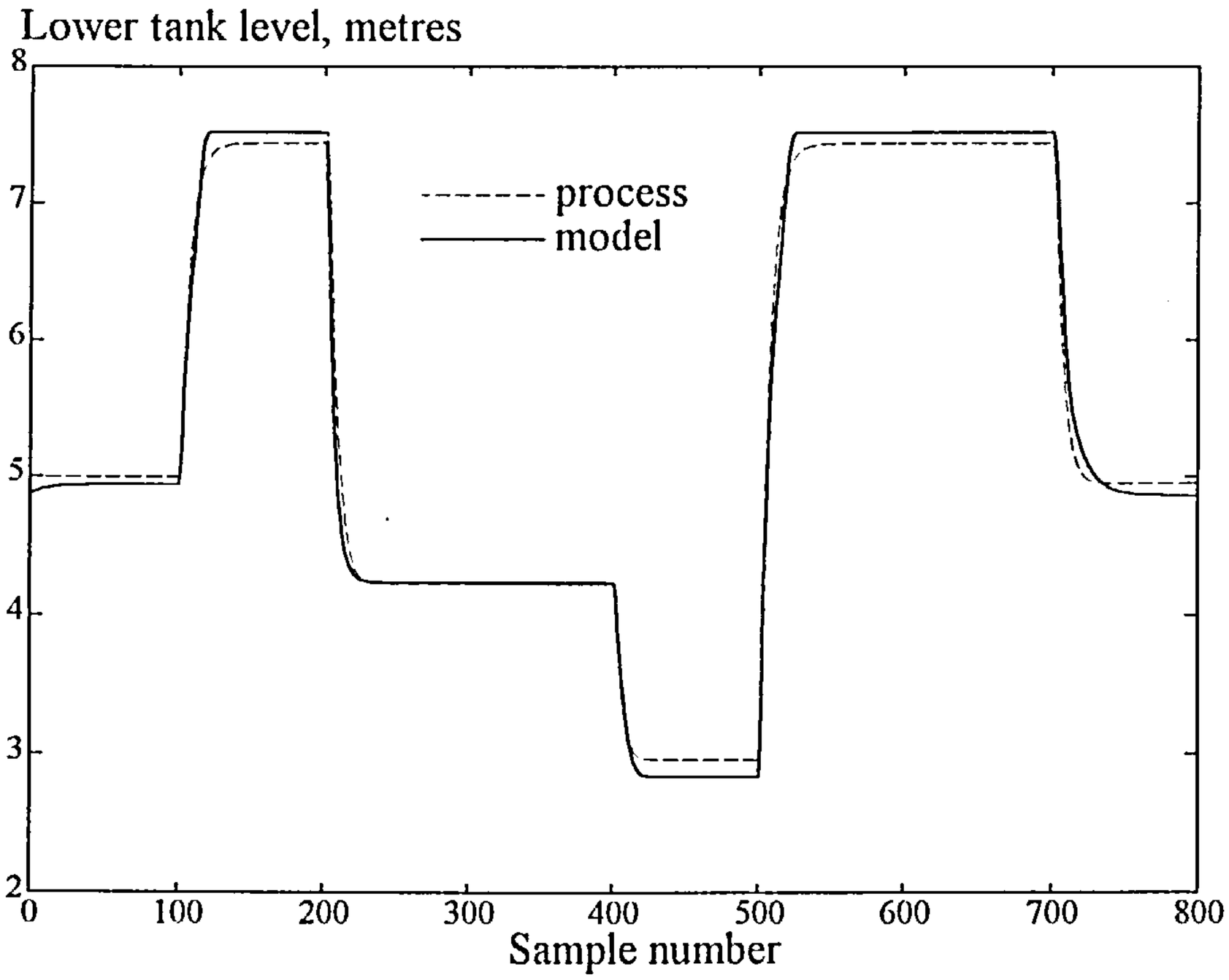
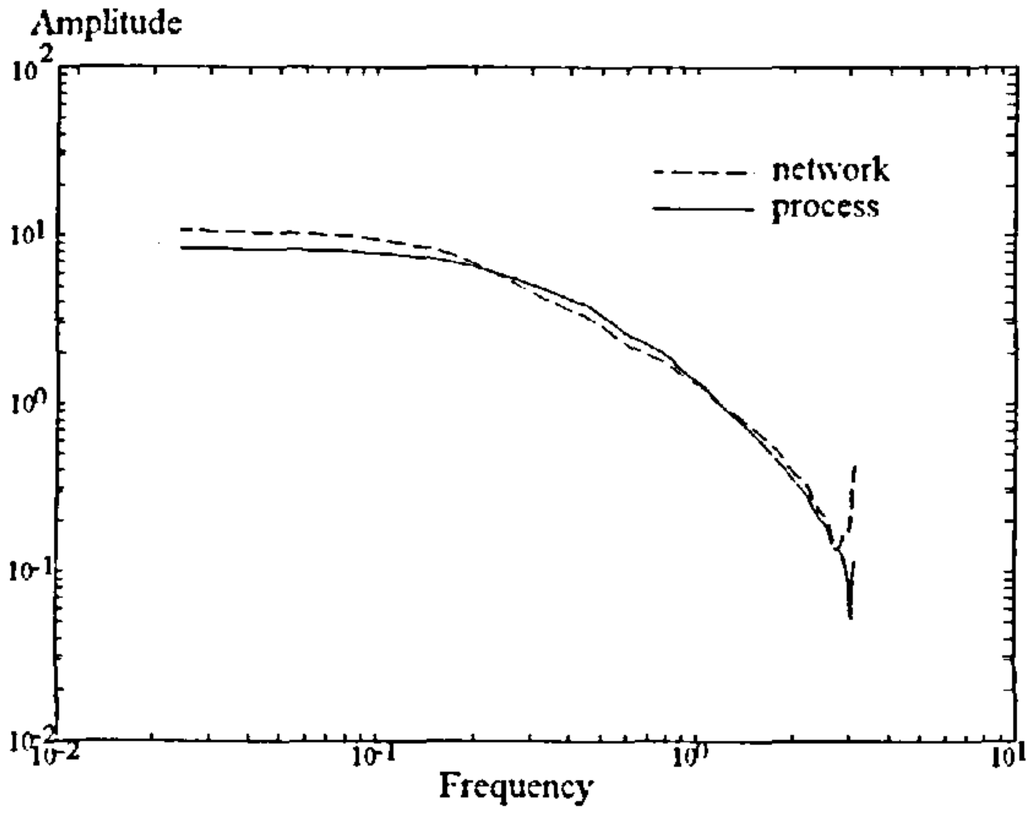
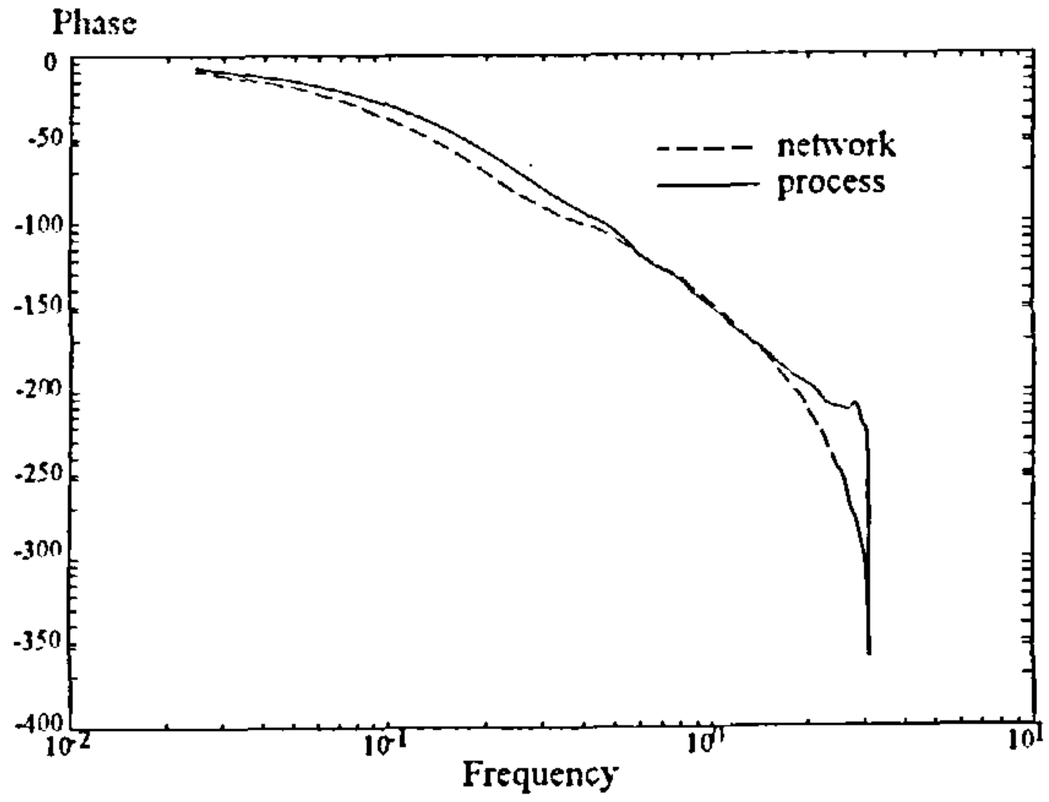


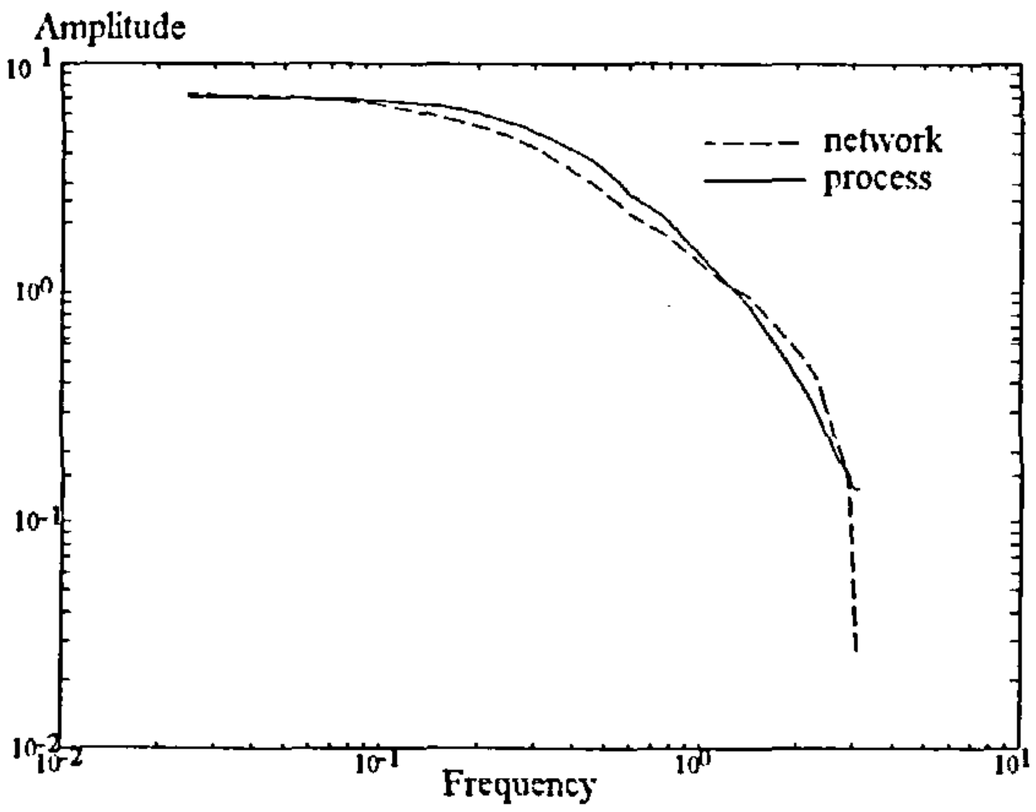
Figure 3.35 Response of SEDM network tested as a model on a set of step input signals

As a model, the response of the neural network to the same signals is illustrated in Figures 3.33 to 3.35. The maximum error between the predicted and actual output for these tests was 4% which is well below the level of noise in the data and illustrates the good noise rejection properties of neural networks.

The frequency responses of FPMODEL1 and the neural network around two operating regions are shown in Figure 3.36. Figure 3.36 illustrates the amplitude and phase responses for the liquid level in tank two at 5m and also when the liquid level was perturbed around 3.2m. The figure illustrates that both the phase and amplitude of the neural network are in good agreement with that of the process, and this further supports the ability of neural networks to identify the non-linear process.







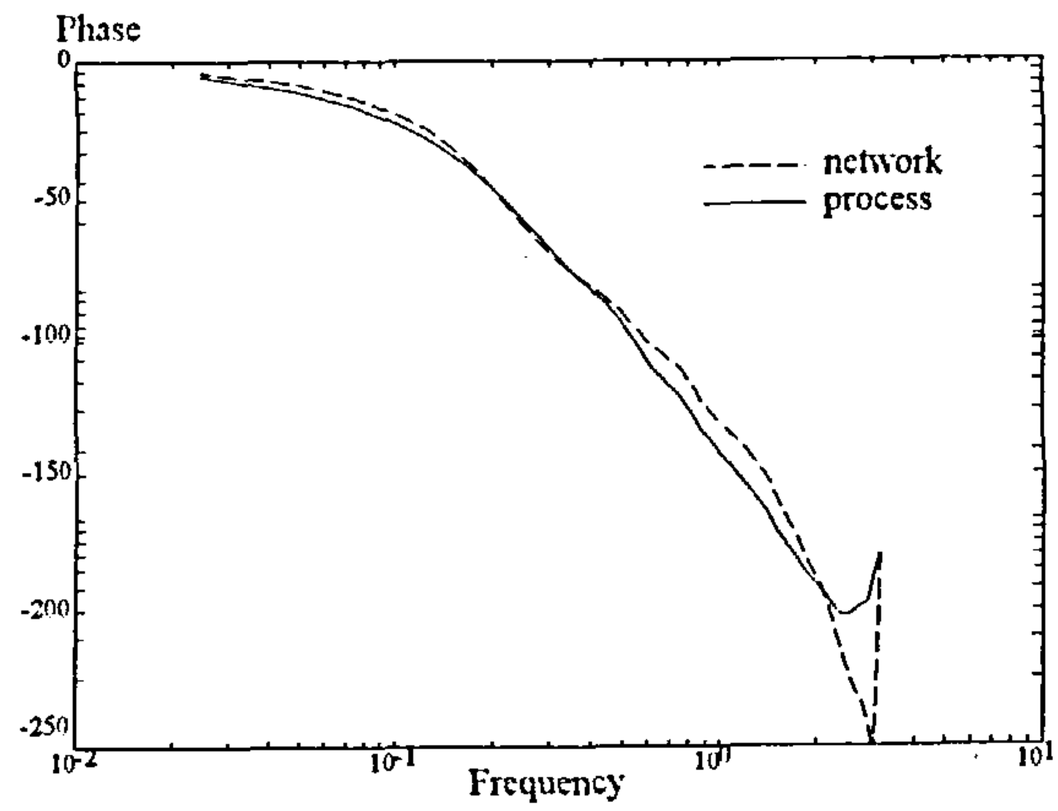


Figure 3.36 Frequency and phase responses about linearised operating points

3.6 SUMMARY

The ability of a neural network to represent a simulation of a non-linear dual tank liquid level system has been demonstrated. It has been shown that SEDM has greater accuracy over SNDM. A first principle model of the laboratory process was obtained and validated against the laboratory process by subjecting both to a number of input test signals and comparing their responses.

A number of practical aspects have been discussed, involving the sampling time and the process excitation signal required in order to successfully identify a process

using a neural network. The amount of training data required to train a neural network was also considered, with techniques used in linear systems identification being deliberated.

The more realistic case of process noise disturbing the reading of the height of liquid in tank 2 was investigated, and it was demonstrated that a neural network trained with noisy data using the SEDM technique was still capable of making accurate predictions in both the one-step-ahead predictor and recursive model configurations. Frequency responses around linearised operating regions for both the trained neural network and FPMODEL1 were taken, and the results indicated that the neural network had learned the process dynamics accurately.

CHAPTER 4

ANN MODELLING OF THE LABORATORY PROCESS

4.1 INTRODUCTION

Chapter 3 demonstrated the feasibility of using the MLP neural network, in conjunction with the SEDM method of conditioning the data to obtain an accurate model of the non-linear dual tank liquid level system over a wide operational region in simulation. This Chapter applies the techniques used and the insights gained in Chapter 3, to obtain a neural network model of the real dual tank liquid level system available in the control systems laboratory.

The dual tank liquid level process rig and necessary equipment used in conjunction is described. Characteristics associated with real processes, including the liquid-level process, not present in the simulations are also considered.

The prediction performance of trained neural networks is evaluated using the three test signals used in Chapter 3. The correlation based tests described in section 2.5.1 are also implemented to further justify the overall network performance. The practical problem of outliers in the real process data, and the effects that these can have on the correlation based model validity tests are also addressed.

4.2 THE LABORATORY PROCESS

The dual tank liquid level system available in the laboratory for on-line investigations is illustrated in Figure 4.1. Both of the process tanks are made of plexiglass cylinders 1.2 m in height, cross-sectional area 0.01474 m^2 and a liquid

capacity of 20 litres. All of the pipe work is standard 25.4 mm copper, with manual valves inserted at the outlets of each of the tanks (m3+m4), and also at the outlet of the process pump (m1) in order to set liquid flow rate at these points as required. The process rig includes industrial standard actuator and sensor equipment which is described in the following sub-section.

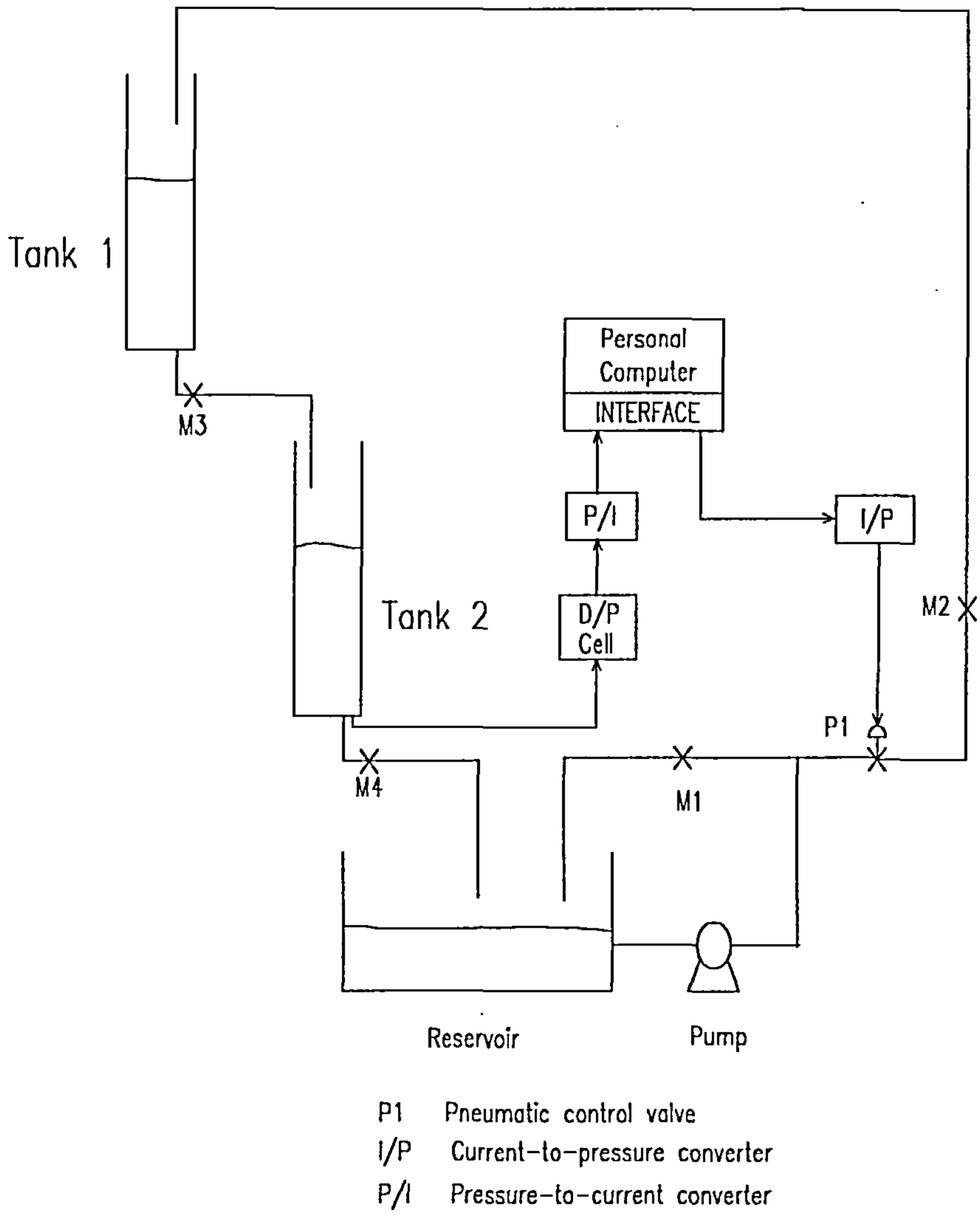


Figure 4.1 The laboratory dual tank liquid level process

4.2.1 Process equipment

Only the height of liquid in tank two is measured, the height of liquid in tank one is assumed to be unmeasurable. A differential pressure (D/P) cell measures the pressure at the bottom of tank two, which is linearly related to the height of liquid. The pressure reading output from the D/P cell is fed into the input of a pressure to current (P/I) converter, and the current is then transformed to a voltage level between 1volt and 5volts before application to the input of the analogue to digital converter (ADC) situated in the computer.

The output from the computer is a digital number which is converted via the DAC into a voltage between 1 volt and 5 volts. This is then changed into a current in order to drive the current to pressure (I/P) converter. The pressure from the I/P converter is used to activate the pneumatic control valve, which controls the flow of liquid into the top tank. The I/P converter and process valve were calibrated such that a digital output number of 100 from the computer closed the process valve, and a number of 200 fully opened the process valve. The digital numbers output to the process valve were integer values between this range.

The liquid, contained within a reservoir, is pumped into the system by a 'Stuart' pump which can be operated continuously for a maximum of 3 hours within any 24 hour time period. This manufacturer's time limit on the use of the pump restricts the amount of data acquisition that can be made in a single process run, and the implications of this are discussed in section 4.2.3.

4.2.2 Features of the real process

The first principle mathematical model obtained of the real process in Chapter 3, and used in simulation studies into obtaining a neural network model of the process gives a good insight into the capabilities of the MLP, but cannot fully justify the ability of the neural network to accurately model a real non-linear process. This is because for real processes certain phenomena exist that are rarely included in simulation studies, and in some instances it may be very difficult to represent these phenomena in a simulation program. The dual tank liquid level process, although not complex in nature, exhibits features that are characteristic of many industrial processes. These include:

1) Noisy and quantised measurement of the bottom tank level,
2) Non-linearities. These are mainly due to a square root relationship between the outflow rate and level of liquid in each tank, and valve stiction,
3) Disturbances, e.g. occasional fluctuations in the pump speed,
4) Time variations on a run-to-run basis, e.g. the same process input can cause different steady state levels in the tanks,
5) Industrial standard actuator and sensor hardware,
6) An unmeasured process state, the level of liquid in the top tank.

The above features enabled a thorough practical investigation of the ability of the MLP neural network to model real processes.

4.2.3 Data acquisition

Software for data acquisition was written in the Microsoft Quick BASIC programming language. Routines were developed to transform the digital number

read into the computer to the corresponding liquid level in the bottom tank, and also so that the required input test signals could be applied to the process. The technique used to excite the process over a wide operational region in order to obtain training data for a neural network was the same as used in section 3.4.4. The steady state level of the liquid in tank 2 was 0.56 m for a digital number of 150 output to the process valve from the computer, and this value was used for the initial start up of the process. The RAS used to excite the process was generated by the uniform random number generator function available in the Quick BASIC language. The range of random numbers generated was between -50 and +50, which were added to the steady state digital output number at each sample time. The range of numbers generated covered the operating range of the process valve (100 to 200) and enabled the process to be thoroughly excited in its operational range.

Selection of the sampling period for measuring the height of liquid in the bottom tank and applying the input test signal was as described in section 3.3.1, and resulted in a sampling period of 0.6 minutes.

The amount of data that the process could provide in order to form both a training and a testing data set was restricted by a maximum operational time. Unlike the simulation studies where there was no fixed limit on the amount of data obtainable, the real dual tank liquid level process could only be operated for a maximum duration of 3 hours in the fixed period due to the continuous running time rating of the process pump. The implication of this is that the amount of data available from the process in a single run is a function of the sampling period. Since the sampling period was set at 0.6 minutes, the upper limit on the amount of data that could be collected for any single process run was 300 input-output sampled data points.

4.3 DEVELOPMENT OF THE NEURAL NETWORK MODEL

Two sets of data were obtained from the process in order to develop a neural network model of the process. The data sets were obtained by exciting the process with two different RAS's on consecutive days. As with the simulation studies one data set was used to train a neural network and the other to test the networks generalisation capabilities at various stages of the training to determine an optimal point to terminate network training. The overall structure, training and validation of the neural network model of the liquid level process is described in this section.

4.3.1 Neural network structure

The input-output structure of the neural network was kept the same as that used in the simulation studies described in Chapter 3, which was 24 input nodes to accommodate two past process inputs and two past process outputs spread over six nodes each, and six processing units in the output layer of the network, the overall network topology is shown in Table 4.1.

Number of input nodes	Number of hidden nodes	Number of output nodes
24	6	6

Table 4.1: Neural network topology

The structure of the neural network was retained since it was demonstrated in the simulation studies that an accurate mathematical model of the process rig had been obtained, and the above mentioned structure was the optimum one. Following a similar approach to that described in section 3.4.1.1, the 'best' number of

processing units used in the hidden layer was found through experimentation. When only one processing unit was used in the hidden layer the network failed to converge and above fifteen units the networks convergence time increased significantly. Six processing units in the hidden layer, which was the same amount in the neural network structure used to model FPMODEL1, when the SEDM technique was employed (section 3.4.2), was found to give the best results.

4.3.2 Training the neural network

The application of the first RAS to the process valve on the laboratory liquid level process excited the liquid level in tank 2 over a range of 35.83 to 81.00 cms, and the second RAS applied to the process valve excited the level over the range 35.34 to 83.72 cms. The input-output data set obtained using the second RAS referred to as 'data set two' was used to train the neural network and the generalisation of the network at various stages of the training was evaluated using the input-output data set obtained from the introduction of the first RAS to the process, referred to as 'data set one'. Data set two was chosen as the training data set since it spanned a wider range than data set one. Both input-output data sets were conditioned to lie in the range 0.1 to 0.9 using the Spread Encoding Data Mapping technique before presentation to the neural network.

As in all the previous neural networks used in the simulation studies, the weights were all initialised to random values before network training proceeded. The values of the gain and momentum terms in the back propagation algorithm were set as shown in Table 3.2.

Figure 4.2 illustrates the MSE of the network output predictions when tested in the model configuration on data set one at various stages of the network training. The

neural network training was terminated after 400 passes of data set two which was considered to be an optimum point.

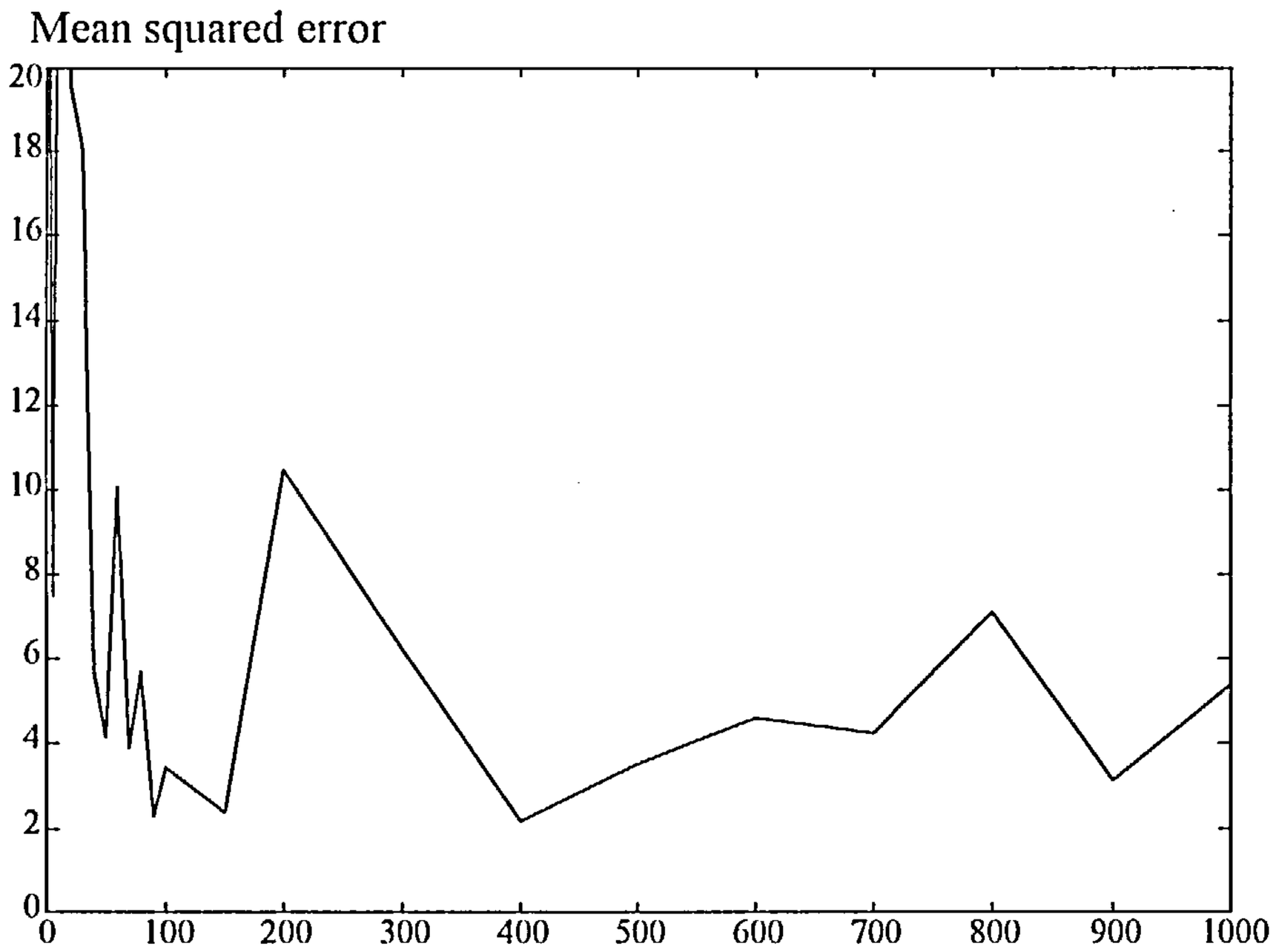


Figure 4.2 Termination of network training

4.3.3 Network validation

Two types of validation of the neural network model were undertaken to fully verify the network's ability to emulate the liquid level process dynamics. As was conducted in the simulation studies, the model was subjected to the set of test signals discussed in section 3.3.4 and the network predictions in both the one-step-ahead predictor and recursive model configurations were compared to the response of the actual process. An alternative form of validation was also performed which would detect if the resulting neural network model provided an adequate fit to the liquid level process. The test, a correlation based one, was fully described in section 2.5.1 and is based upon the idea that if the neural network model of the

process is an adequate one then the tests performed by equations 2.19 to 2.23 should hold.

4.3.3.1 Validation using the test signals

Figures 4.3 to 4.5 illustrate the predictions of the neural network model when configured in the one-step-ahead predictor mode and tested on the RAS, PRBS and set of step inputs. Only the first 100 points of the total 300 data from the experiment are shown for clarity. In all of the three cases the response of the neural network accurately matches that of the liquid level process output.

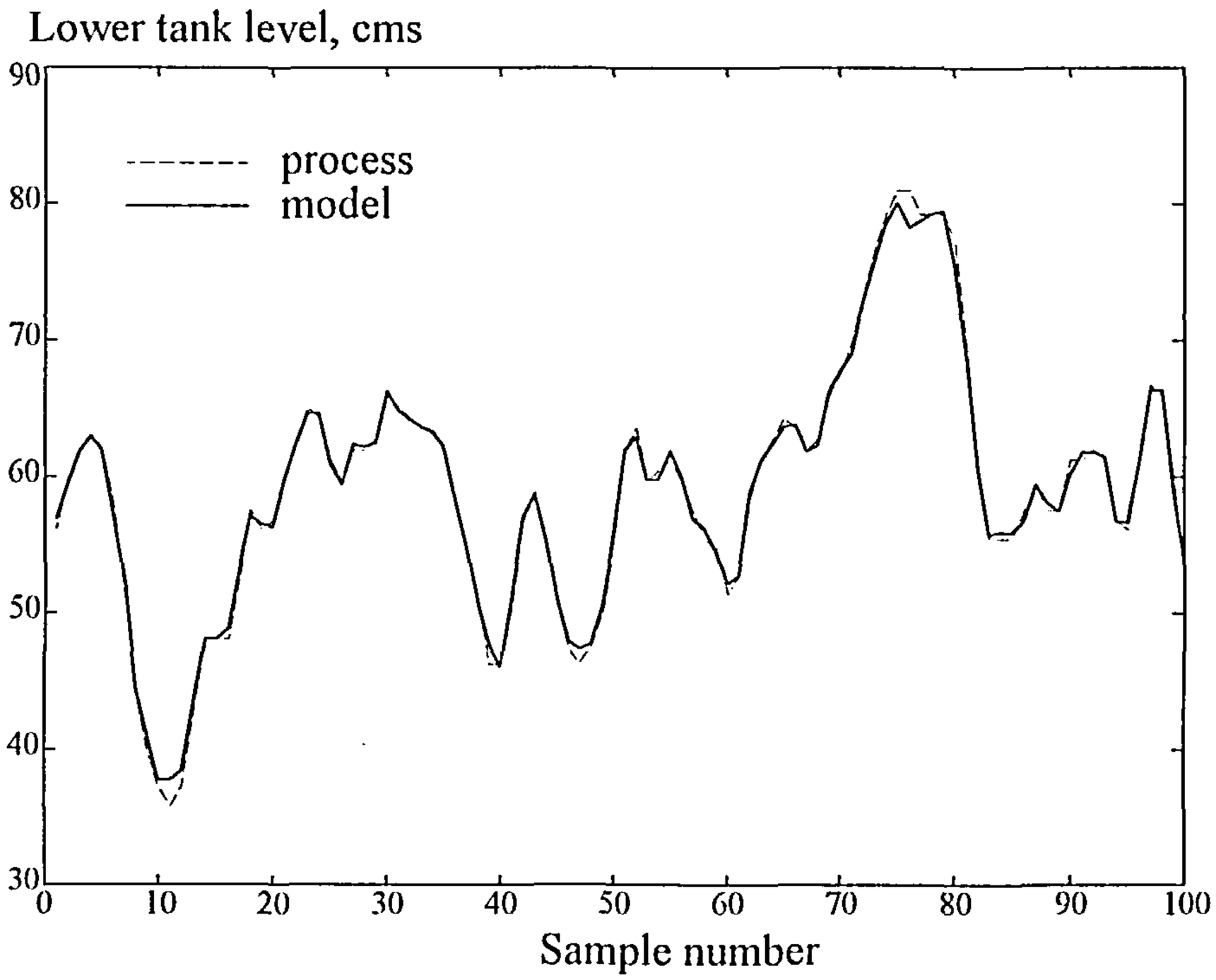


Figure 4.3 Response of network tested as a predictor on a random amplitude signal

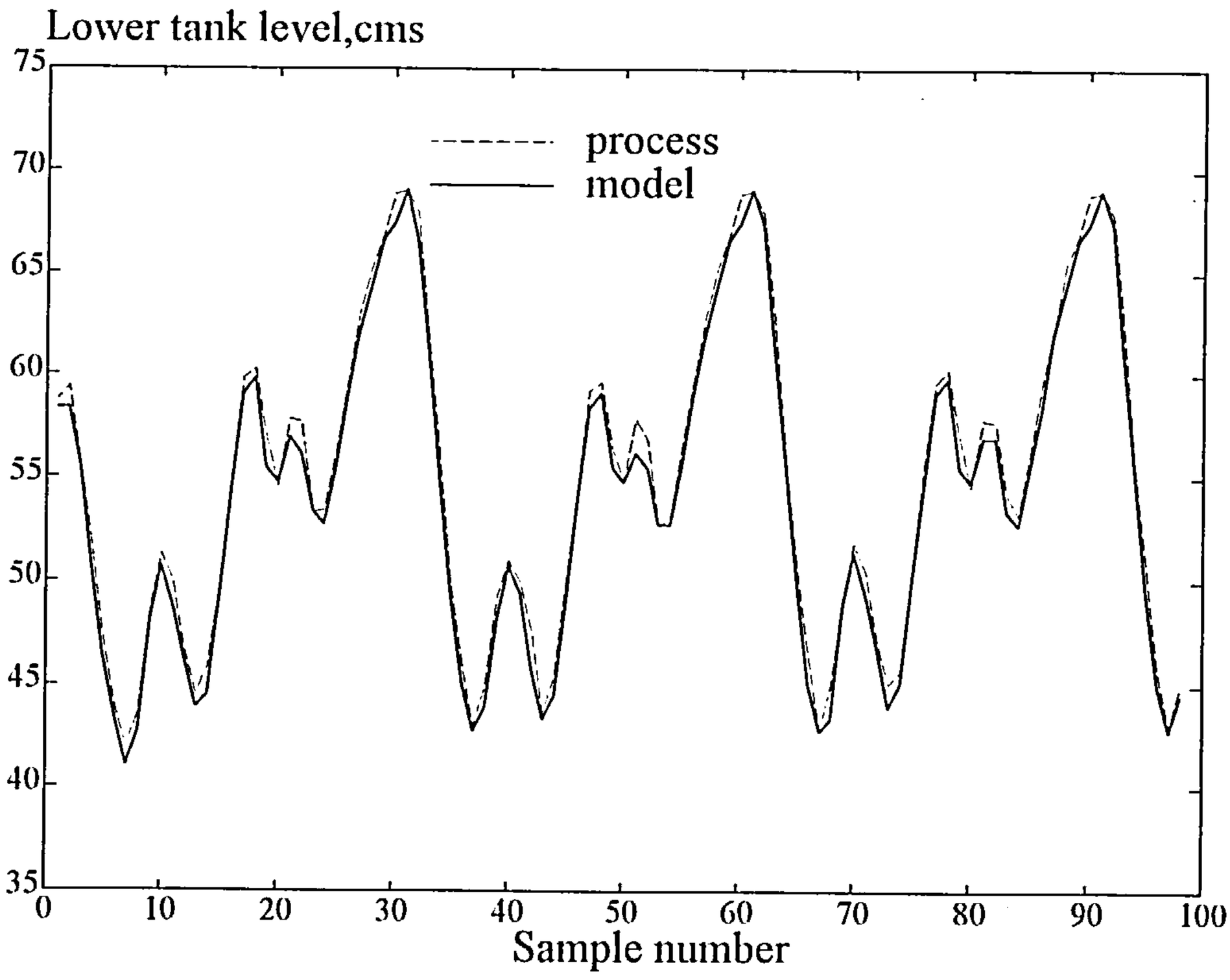


Figure 4.4 Response of network tested as a predictor on a PRBS signal

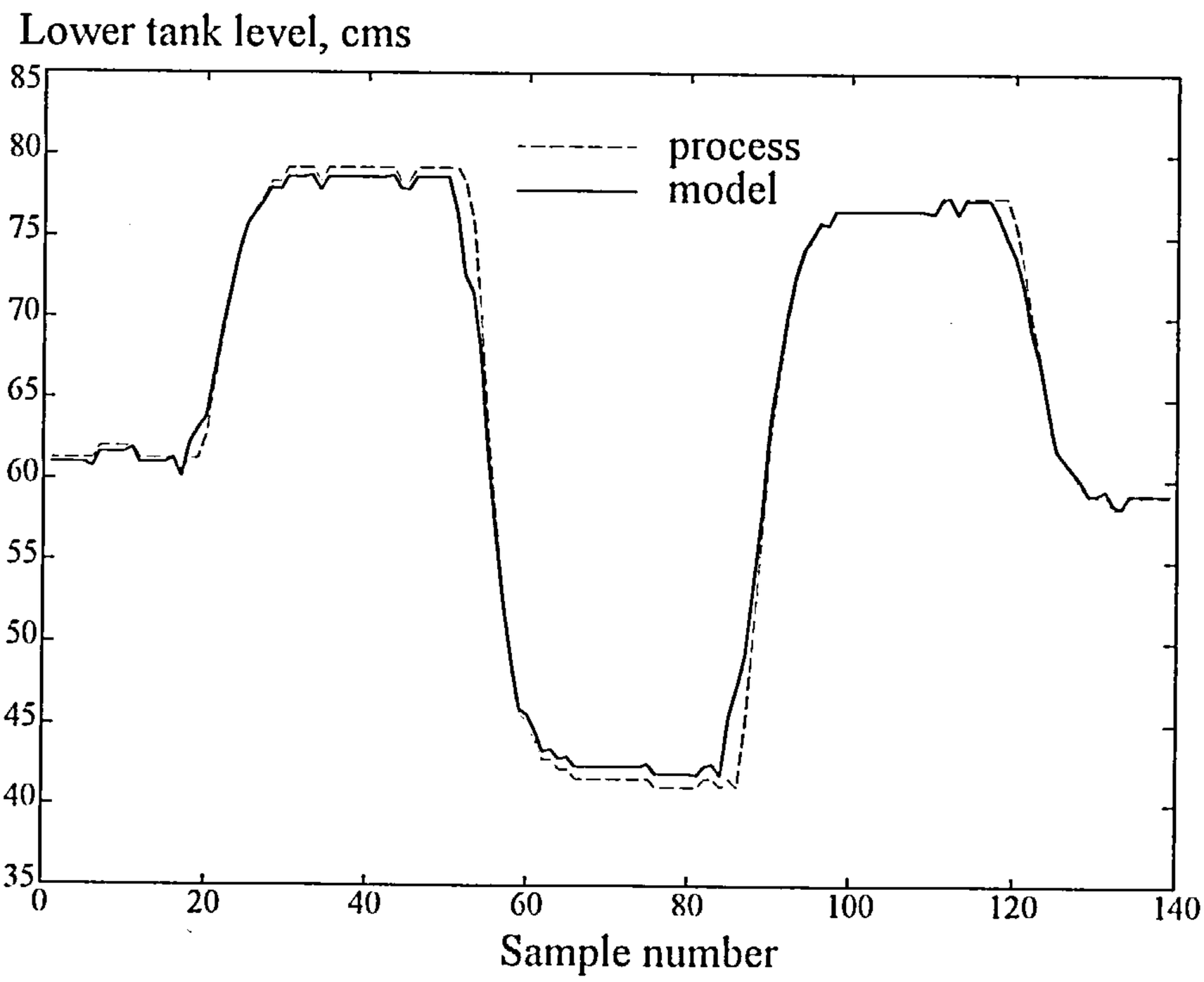


Figure 4.5 Response of network tested as a predictor on a set of step input signals

Configured in the recursive model mode the response of the neural network to the RAS input is shown in Figure 4.6. The prediction accuracy was slightly less than that of Figure 4.3 but was still acceptable.

Figure 4.7 shows the responses of the process and the neural network model when subjected to the PRBS signal, it can be seen that the model response is quite accurate. The results of the step response test are illustrated in Figure 4.8, and again it is demonstrated that accurate network predictions have been achieved.

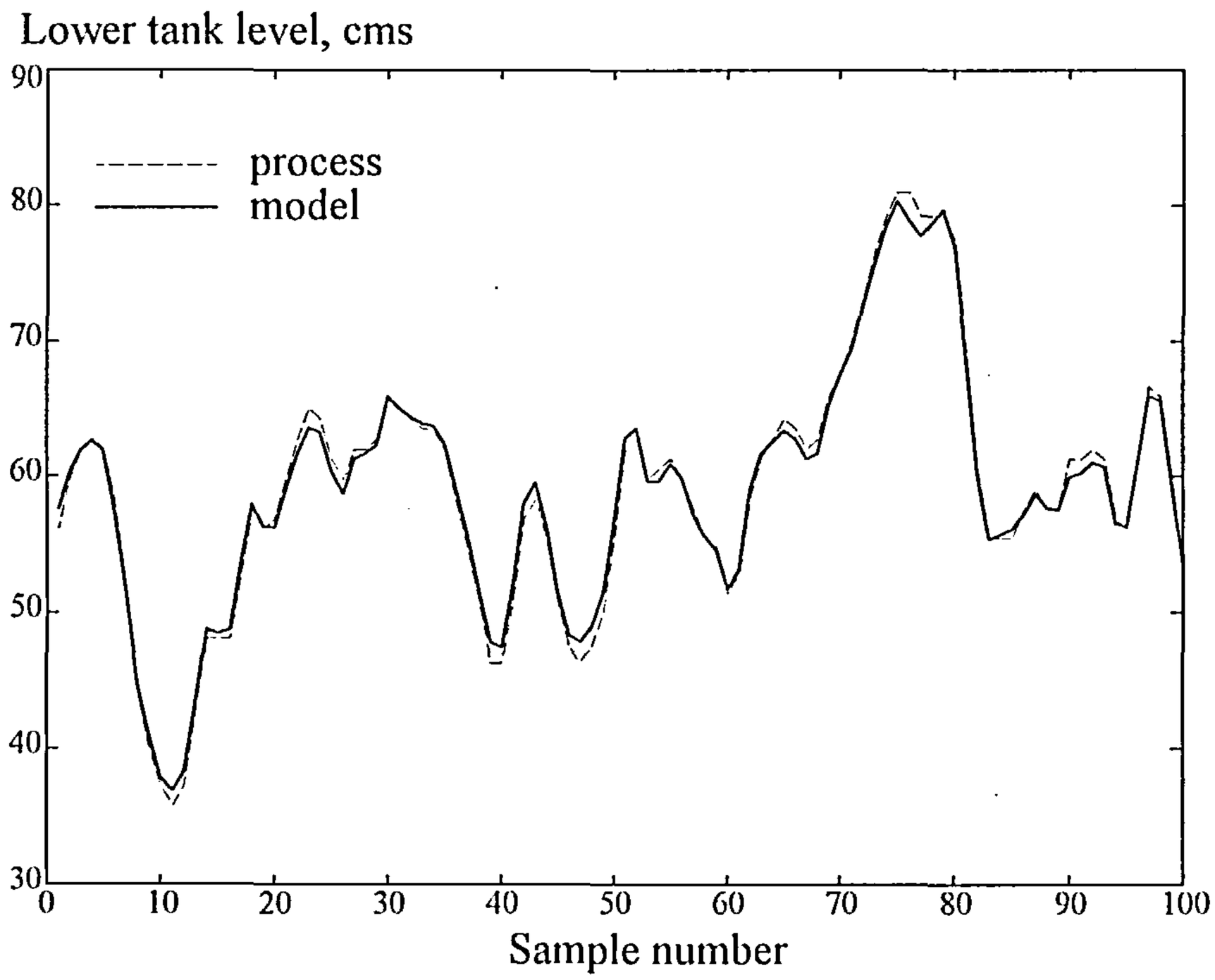


Figure 4.6 Response of network tested as a model on a random amplitude signal

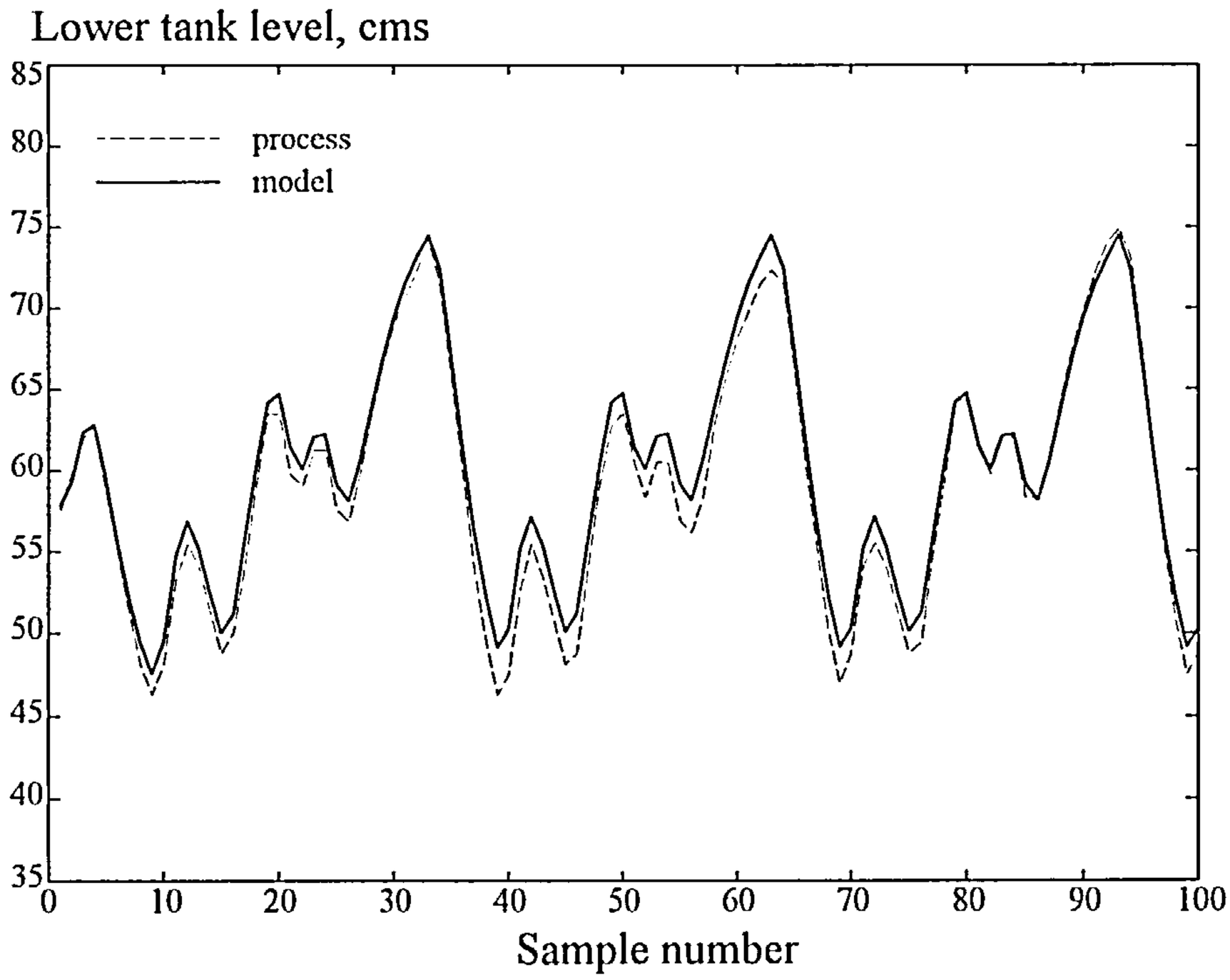


Figure 4.7 Response of network tested as a model on a PRBS signal

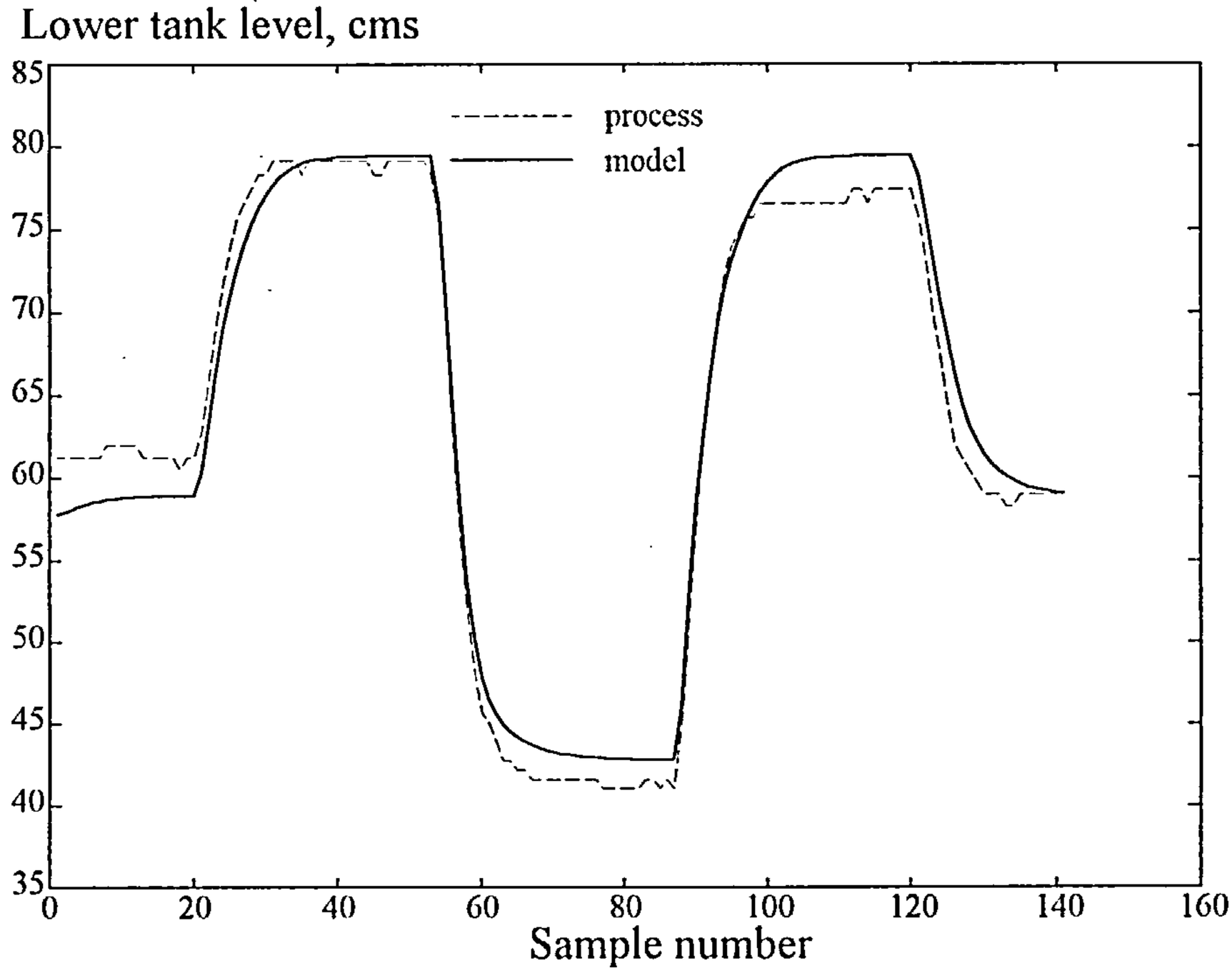


Figure 4.8 Response of network tested as a model on a set of step input signals

Figures 4.7 and 4.8 also serve to illustrate the time varying dynamics of the liquid level process. These figures show that for both test signals the height of liquid in tank 2 reaches different levels for the same signal level input, and is particularly apparent at the extreme levels attained by the process. In contrast the neural network model consistently reaches the same level for the test inputs which is as expected from a static process model.

4.3.3.2 Validation using correlation based techniques

These tests indicate if the neural network model is an adequate representation of the underlying process. The tests can also indicate if the network model structure contains the correct input node assignments, i.e. that no past process input or output samples have been disregarded from the model that should have been included (BILLINGS et al., 1992). It was found in this study that the tests are sensitive to data anomalies that occur in practice due to process disturbances. These data anomalies, commonly known as outliers, are characterised by large spikes in the prediction error sequence. If the tests are performed without taking outliers into consideration then misleading results may be obtained as demonstrated below.

The first test, $\phi_{\varepsilon\varepsilon}(\tau)$ described by equation 2.19, was performed on the prediction errors obtained by testing the neural network in the one-step-ahead predictor mode on the training RAS. Figure 4.9 illustrates the result of this test where it is clearly seen that the correlation is outside the 95% confidence limits at lags τ=-2, -1, 1, 2 indicating that the resulting model fit is inadequate. On examination, the prediction errors were found to have several points where it was considered that an outlier had occurred. The method used to remove outliers was to set a threshold error level,

with any errors above the threshold being set to zero. The threshold level used was 3 times the standard deviation of the prediction error sequence and this removed five outliers from the 300 points in the data set. Figure 4.10 illustrates the result of the first test $\phi_{\varepsilon\varepsilon}(\tau)$ after the outliers had been removed which clearly shows that the test is now satisfied.

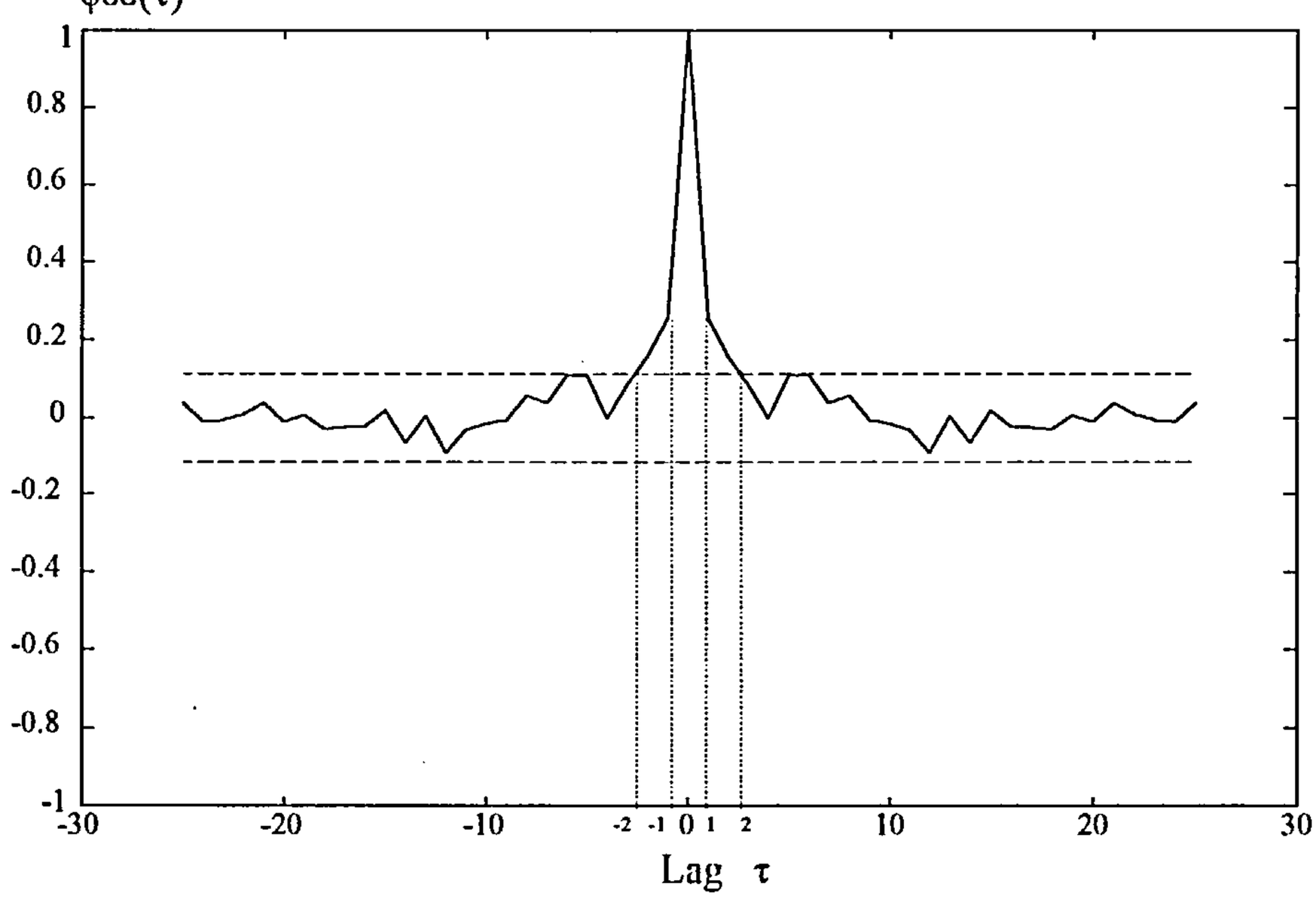


Figure 4.9 Second order model test result for $\phi_{\varepsilon\varepsilon}(\tau)$

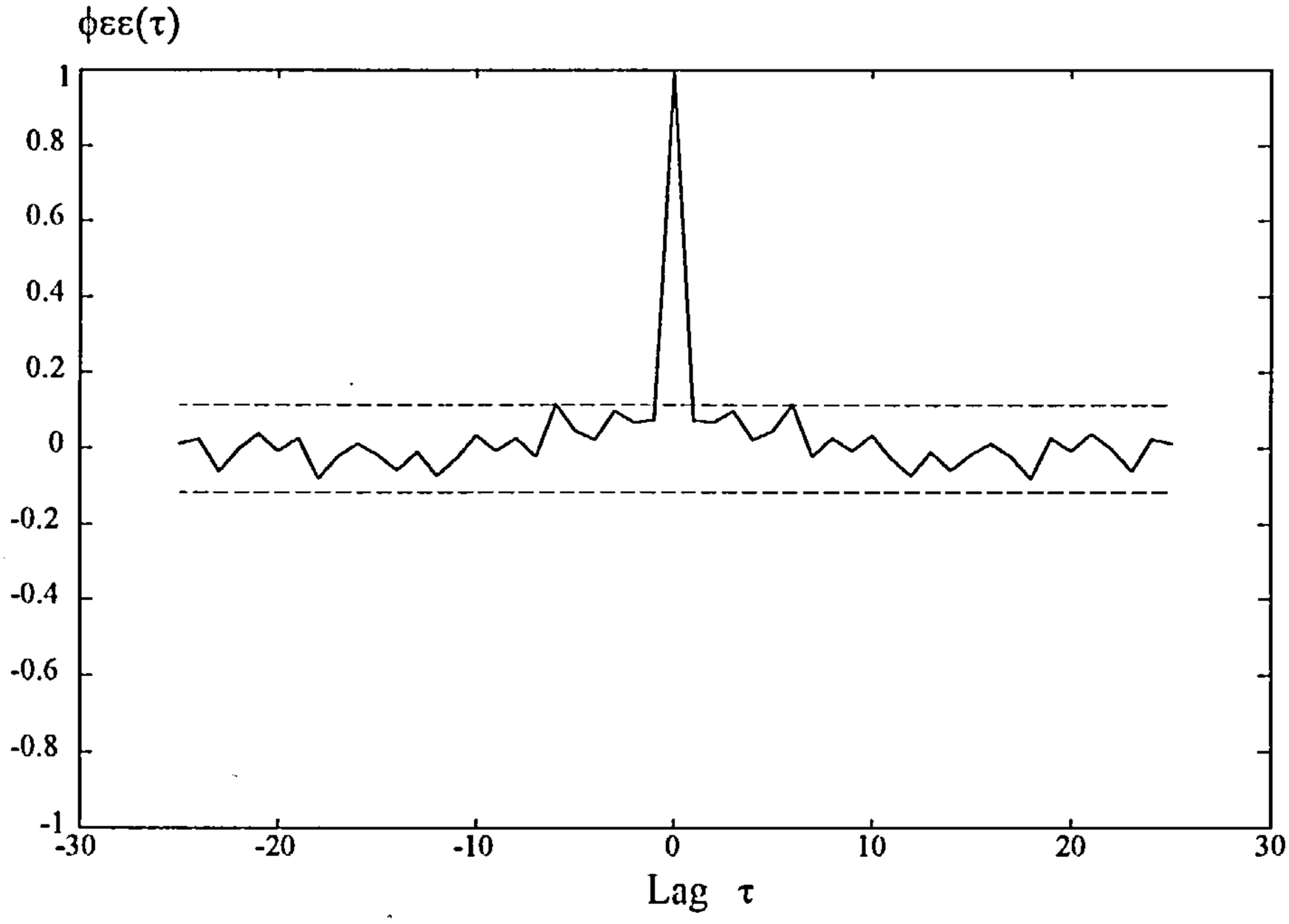


Figure 4.10 Second order model test result for $\phi_{\varepsilon\varepsilon}(\tau)$ after removal of outliers

The remaining four tests, equations 2.20 to 2.23, were carried out with the outliers removed from the prediction error sequence and the results of the tests are shown in Figures 4.11 to 4.14. In each case, the trained neural network satisfied the requirements of the test further supporting that the neural network model structure chosen was adequate for representing the input-output dynamical structure of the dual tank liquid level process.

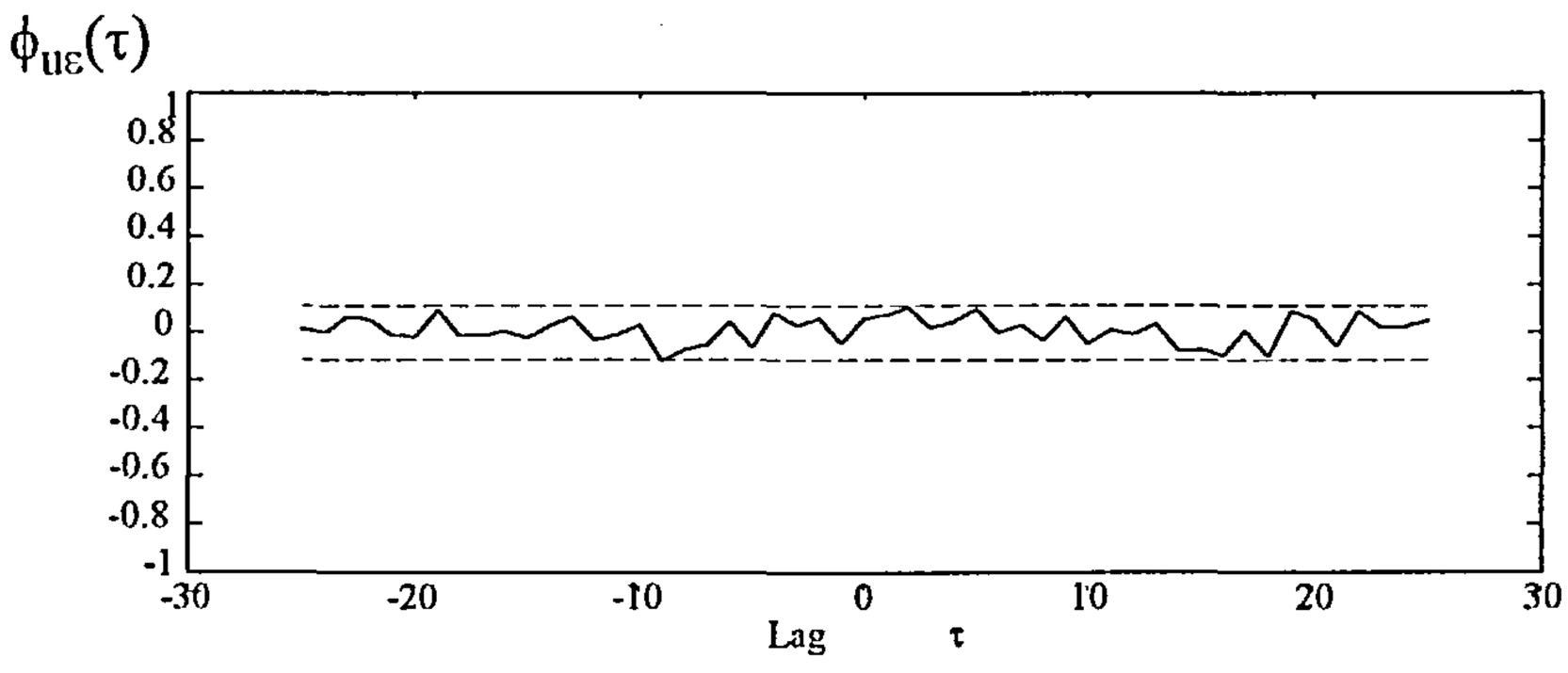


Figure 4.11 Second order model test result for $\phi_{u\varepsilon}(\tau)$

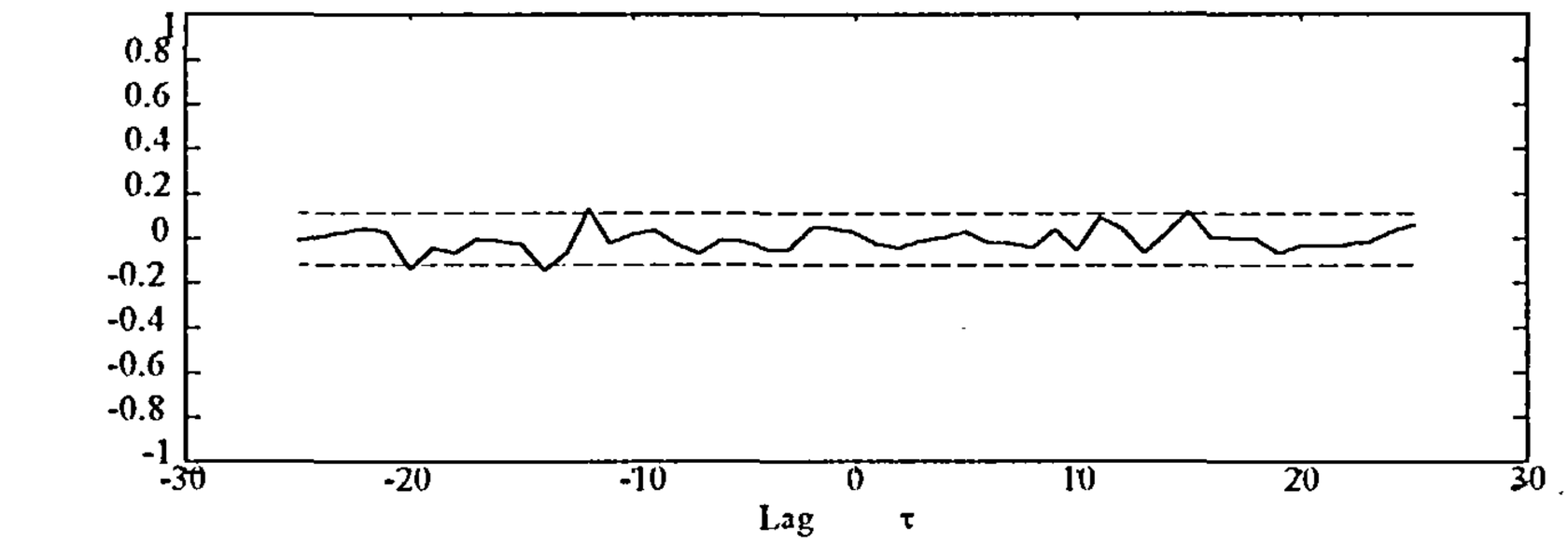


Figure 4.12 Second order model test result for $\phi_u{}^2{}_\varepsilon(\tau)$

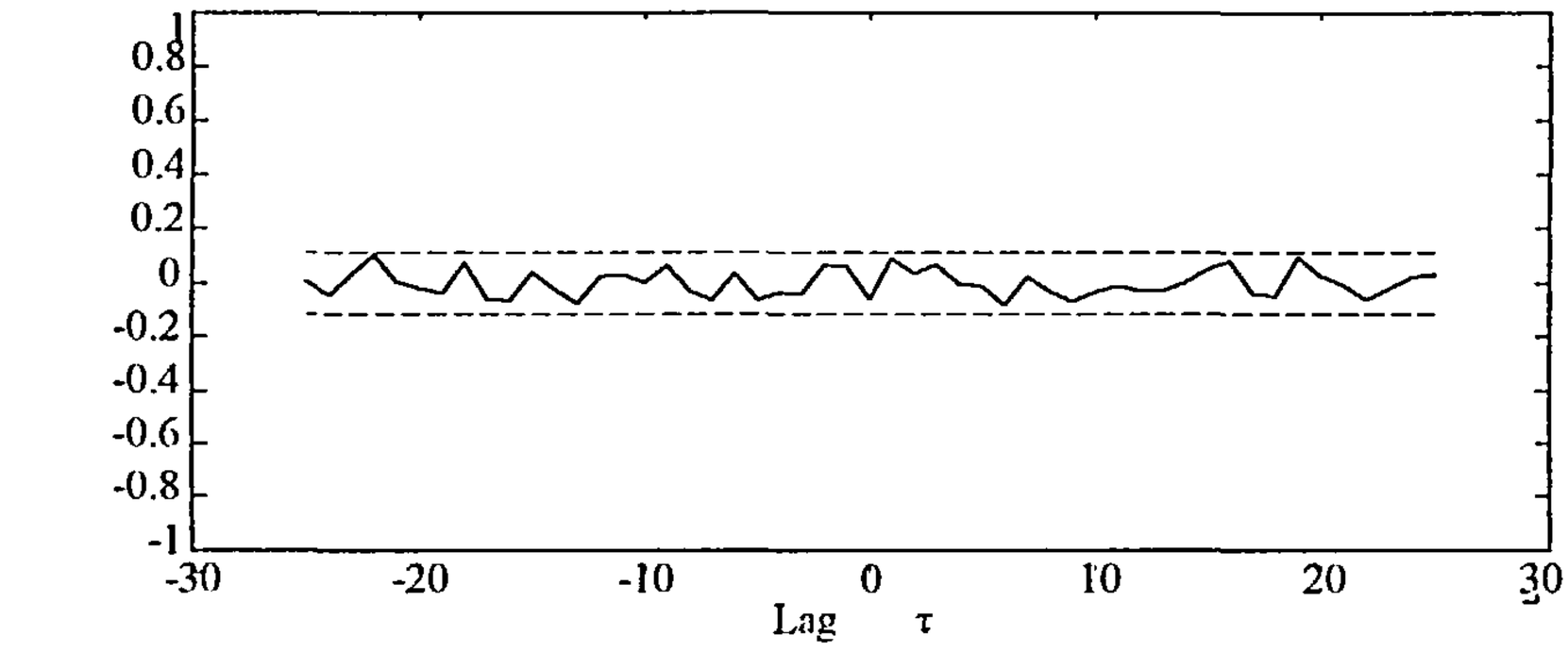


Figure 4.13 Second order model test result for $\phi_u{}^2{}_\varepsilon{}^2(\tau)$

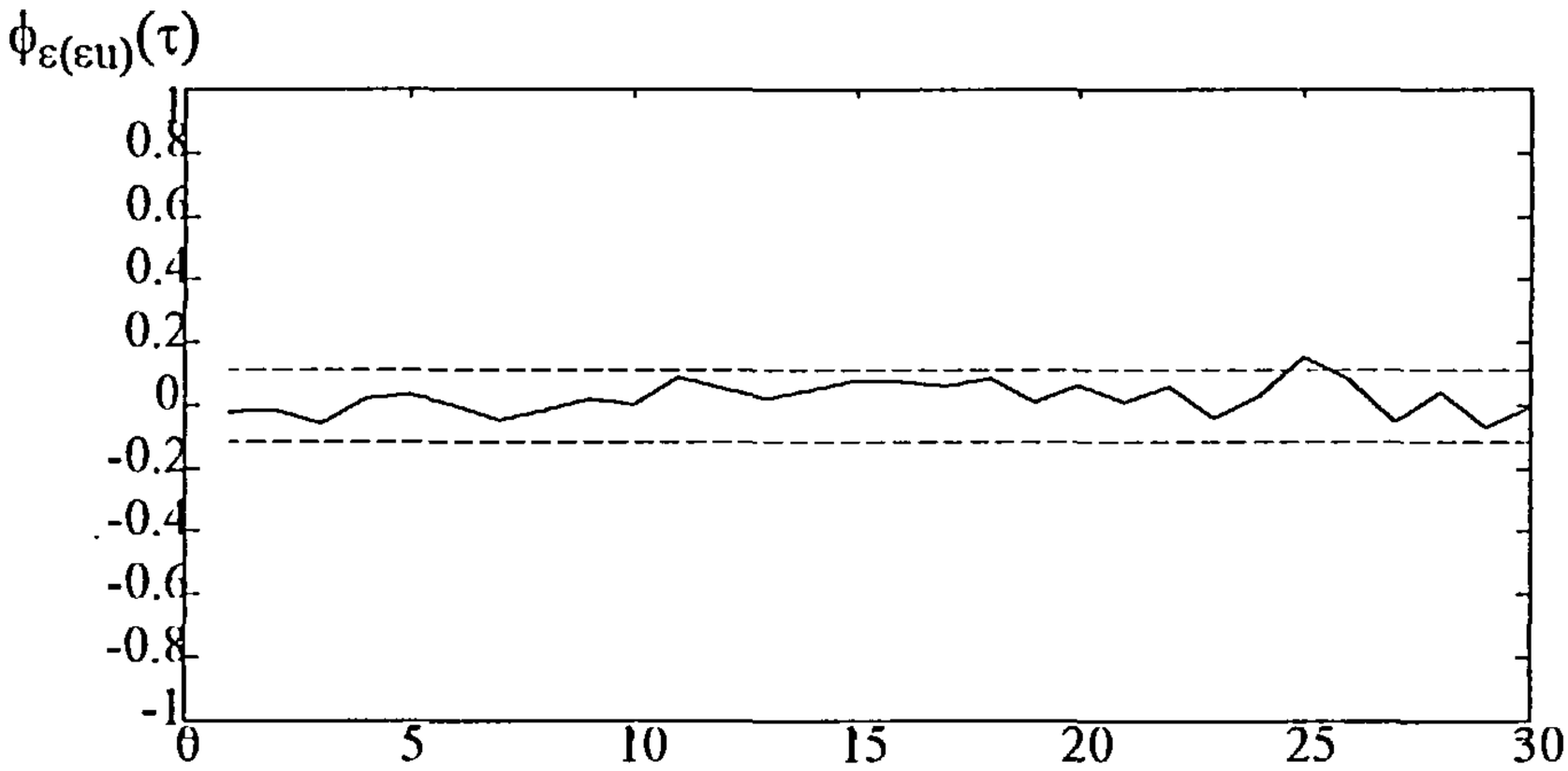


Figure 4.14 Second order model test result for $\phi_{\varepsilon(\varepsilon u)}(\tau)$

In order to fully justify the use of two past process inputs and outputs as the choice of inputs to the neural network, two other structures were trained and tested on "data set two" obtained from the process. The number of inputs to the neural networks were chosen on the basis of a first order model and a third order model. The remaining network topology was retained, which was six processing units in the hidden layer and six processing units in the output layer. The networks training was terminated when both networks gave acceptable MSE's when tested on "data set one". The prediction errors and process input data for each model were then formed in order to perform the correlation based tests. Both of the data sets were normalised and any outliers in the prediction error sequence were removed.

The model structure of the neural network configured as a third order model was firstly evaluated. Figures 4.15 to 4.19 illustrate the results for the third order model when the five correlation based tests were performed, it can be seen that all the tests were satisfied, indicating that the resulting neural network model is an adequate representation of the liquid level process. The network was also tested on the set of test signals discussed in section 3.3.4 and accurate network predictions were obtained.

When the neural network configured as a first order model was subject to the correlation based tests the indication was that the model was an inadequate representation of the process. The results of the tests are shown in Figures 4.20 to 4.24. Although three of the tests are within the 95% confidence limits, Figures 4.21 and 4.23 are well outside of these limits. When the neural network was tested on the set of test signals, the prediction accuracy of the network was very poor in the one-step-ahead prediction configuration which also confirmed the results of an inadequate model fit.

Figure 4.15 Third order model test result for $\phi_{\varepsilon\varepsilon}(\tau)$

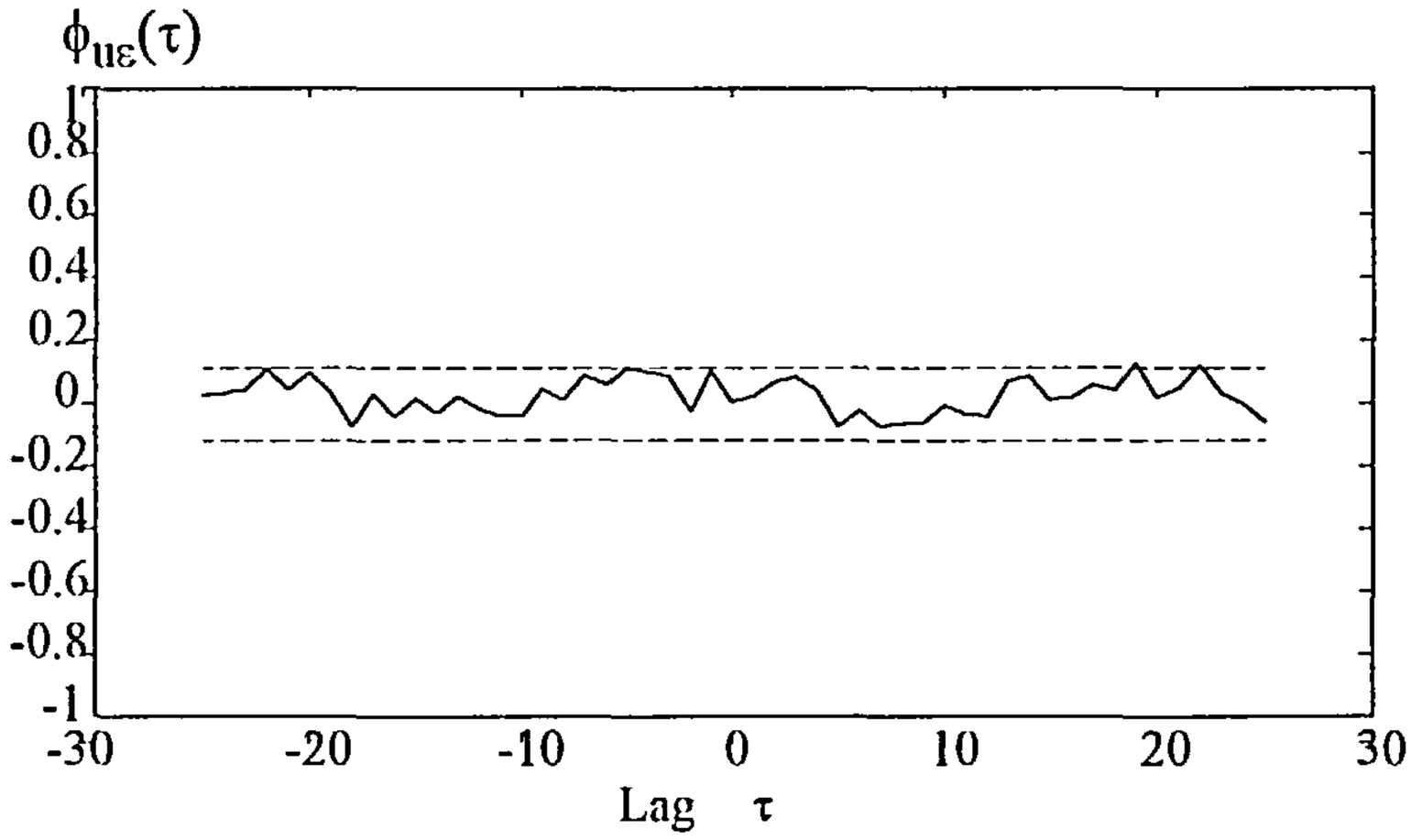


Figure 4.16 Third order model test result for $\phi_{u\varepsilon}(\tau)$

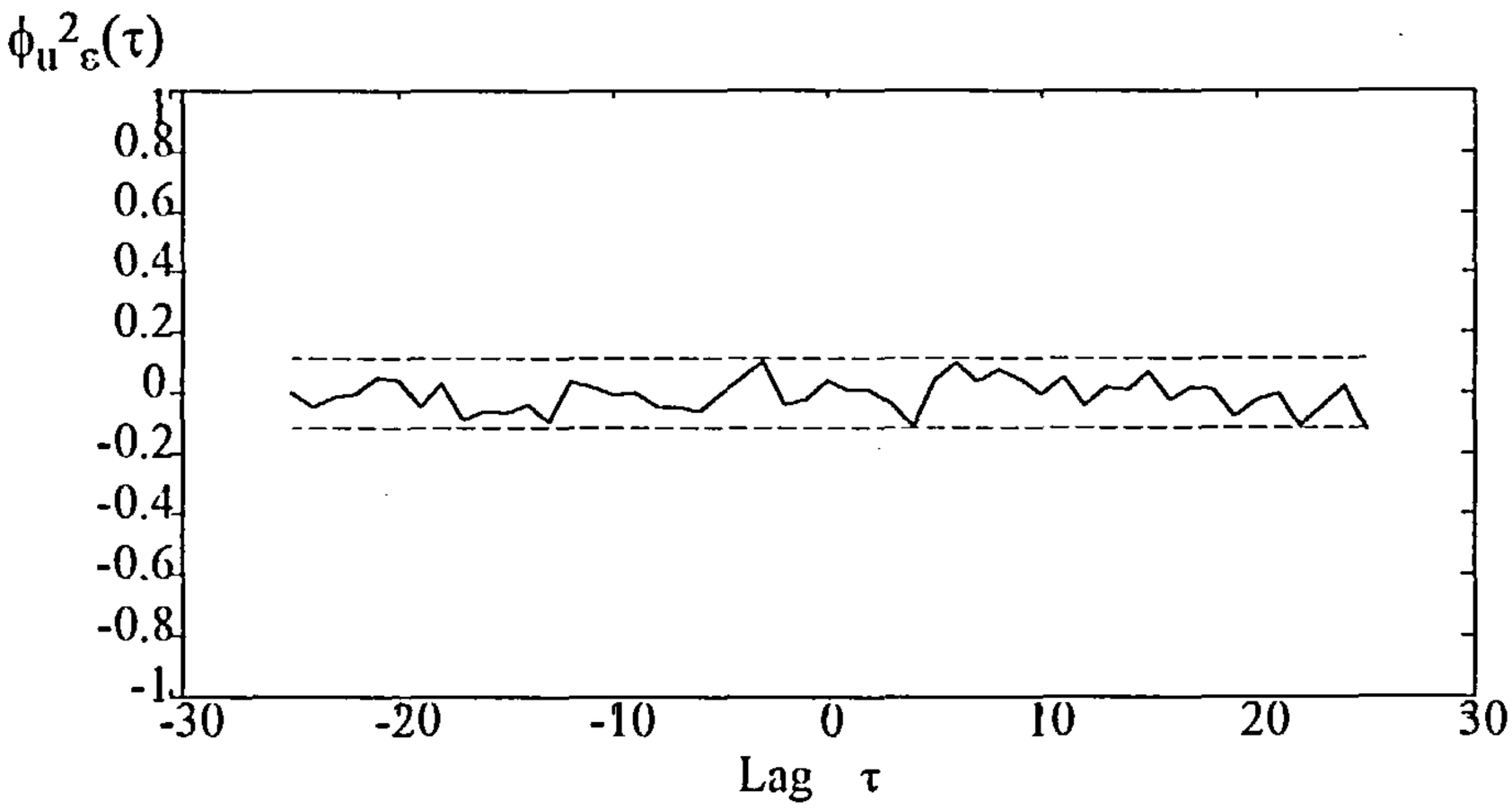


Figure 4.17 Third order model test result for $\phi_u{}^2\varepsilon(\tau)$

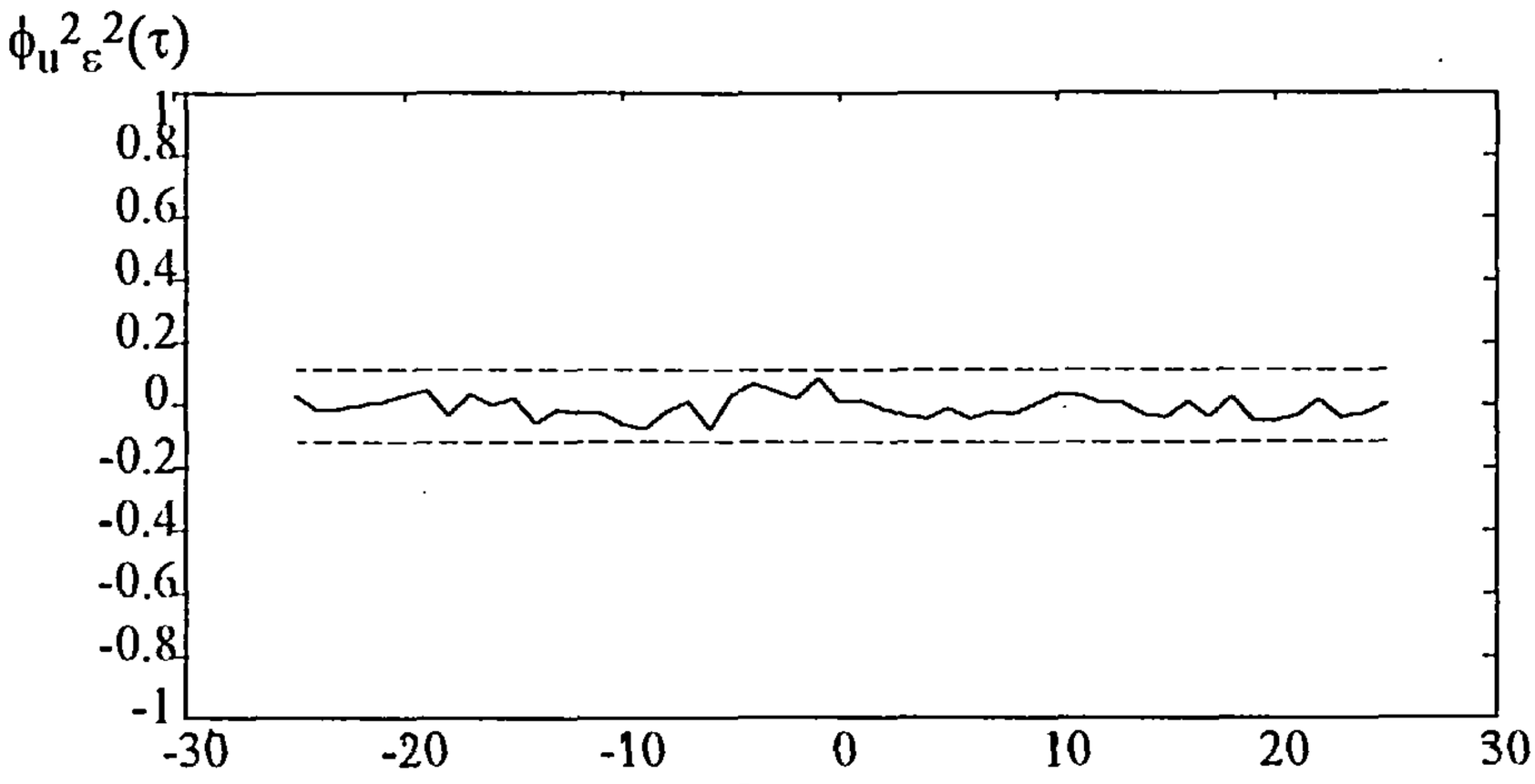


Figure 4.18 Third order model test result for $\phi_u{}^2\varepsilon^2(\tau)$

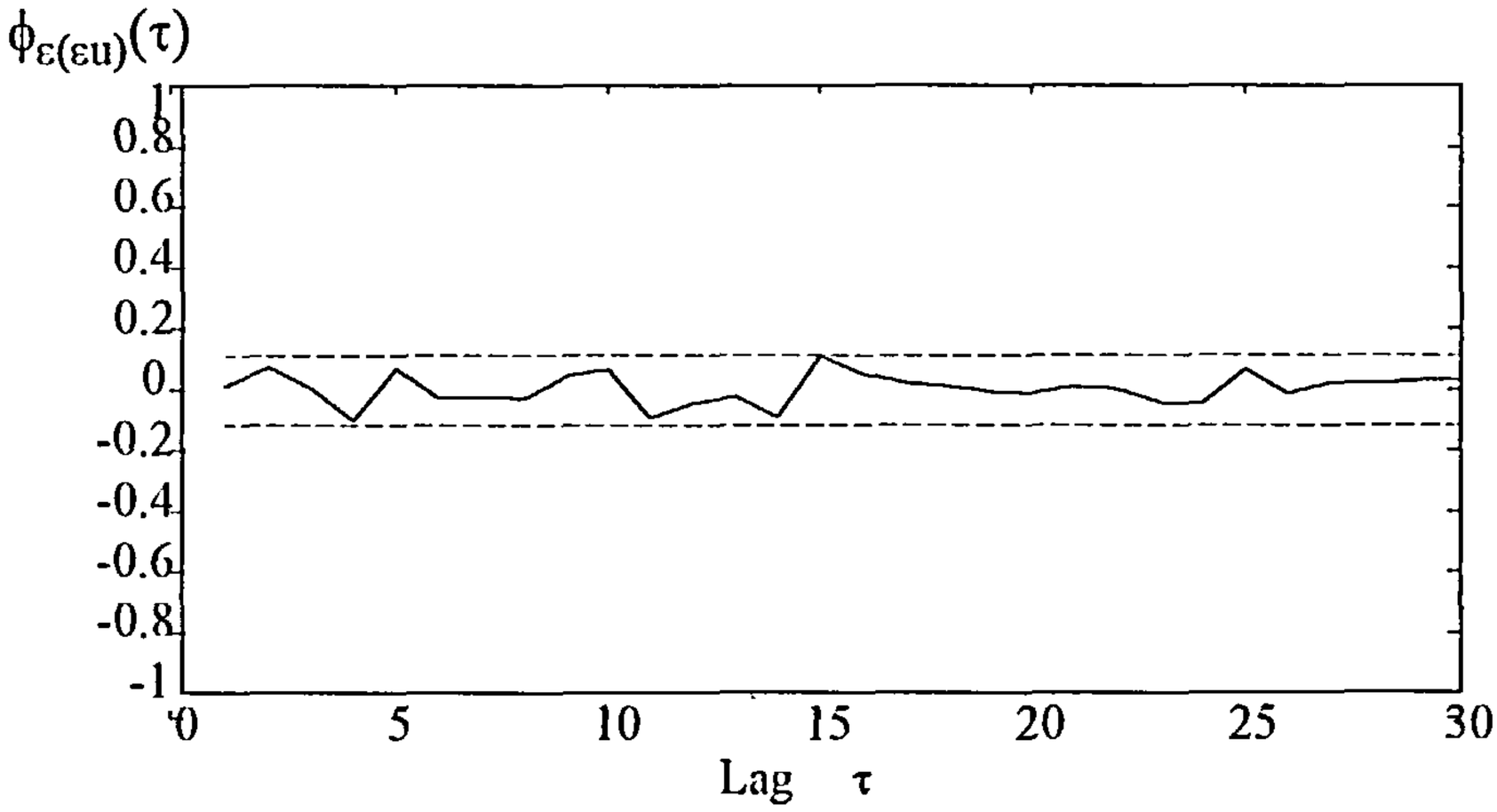


Figure 4.19 Third order model test result for $\phi_{\varepsilon(\varepsilon u)}(\tau)$

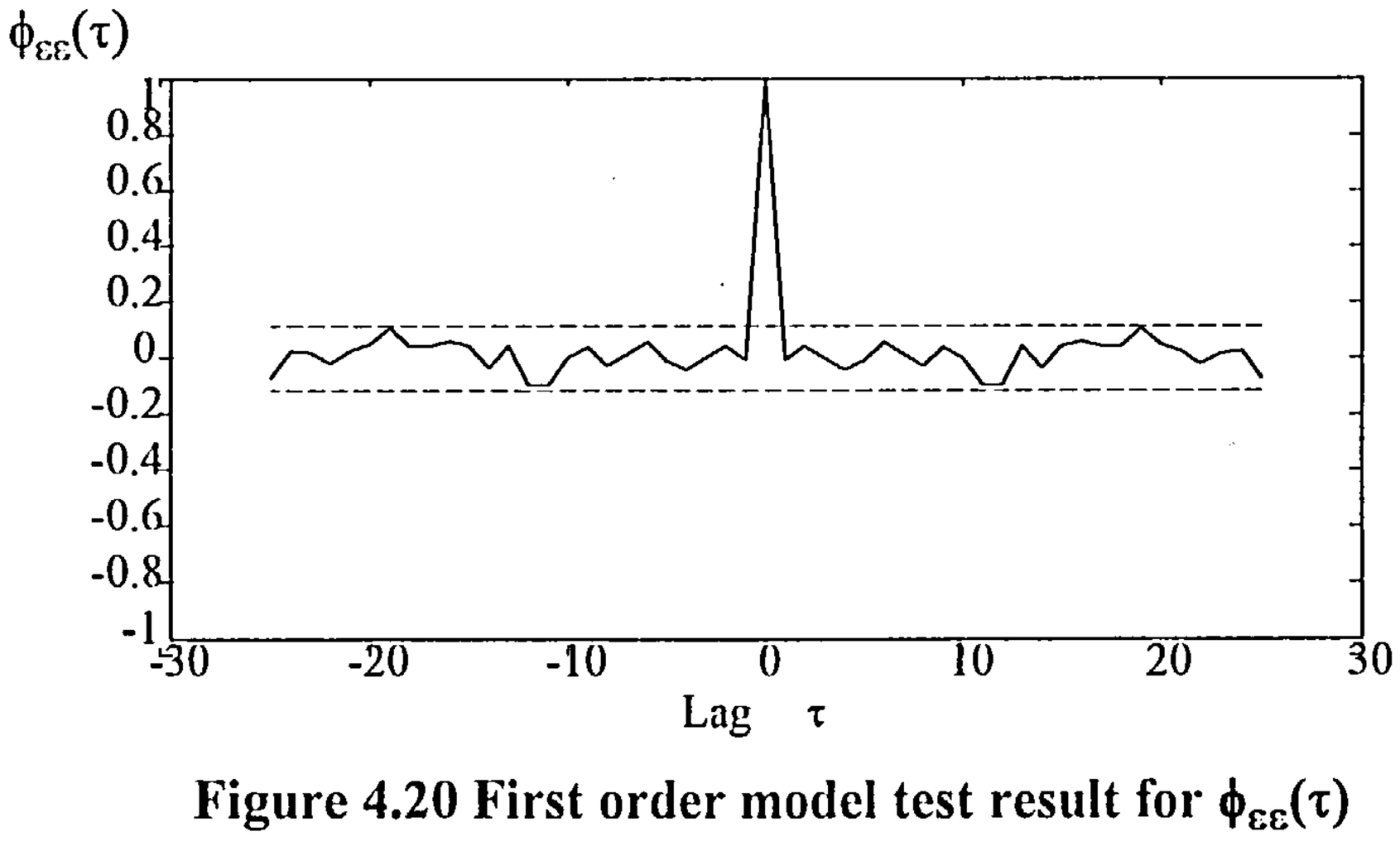


Figure 4.20 First order model test result for $\phi_{\varepsilon\varepsilon}(\tau)$

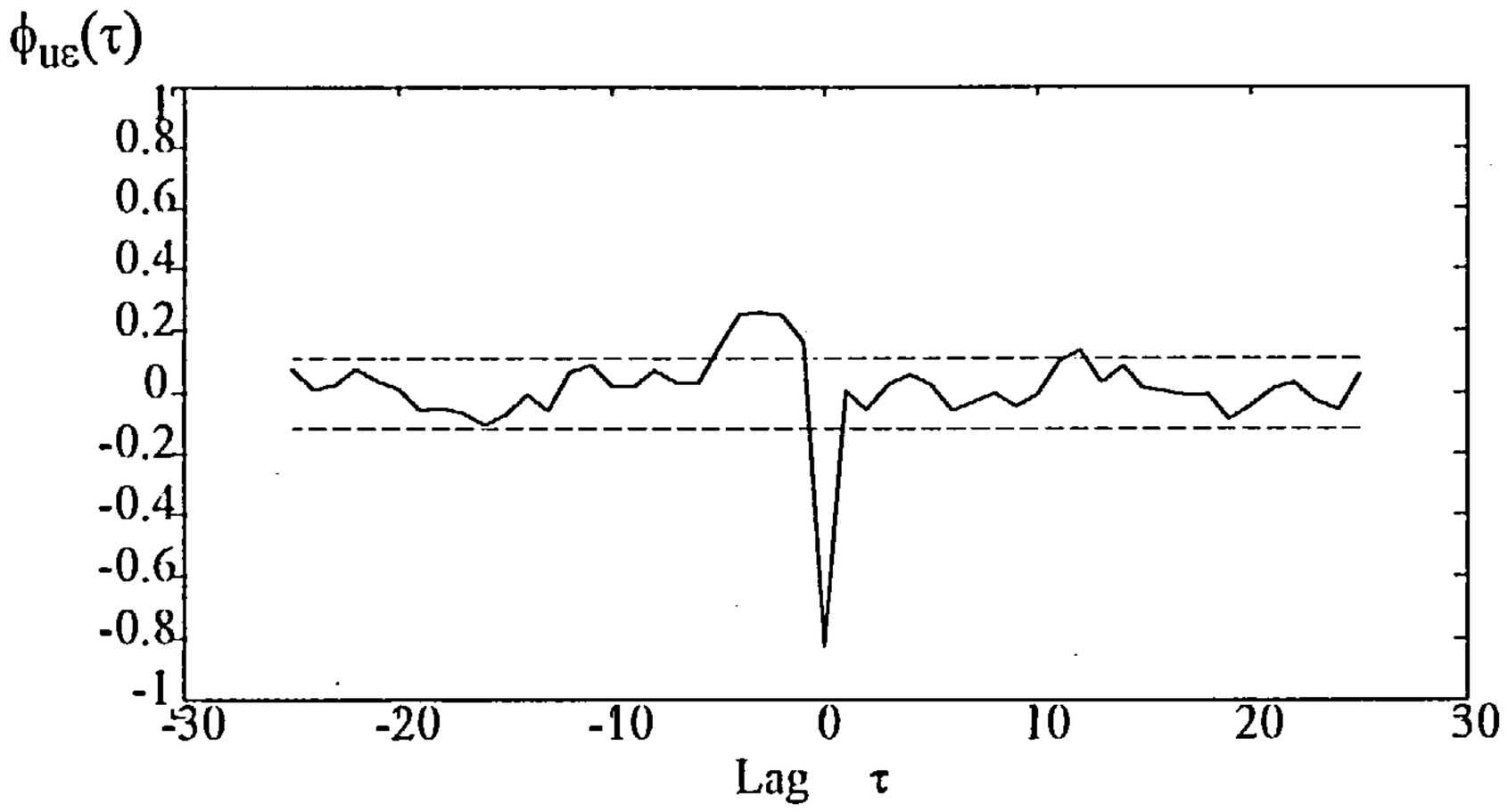


Figure 4.21 First order model test result for $\phi_{u\varepsilon}(\tau)$

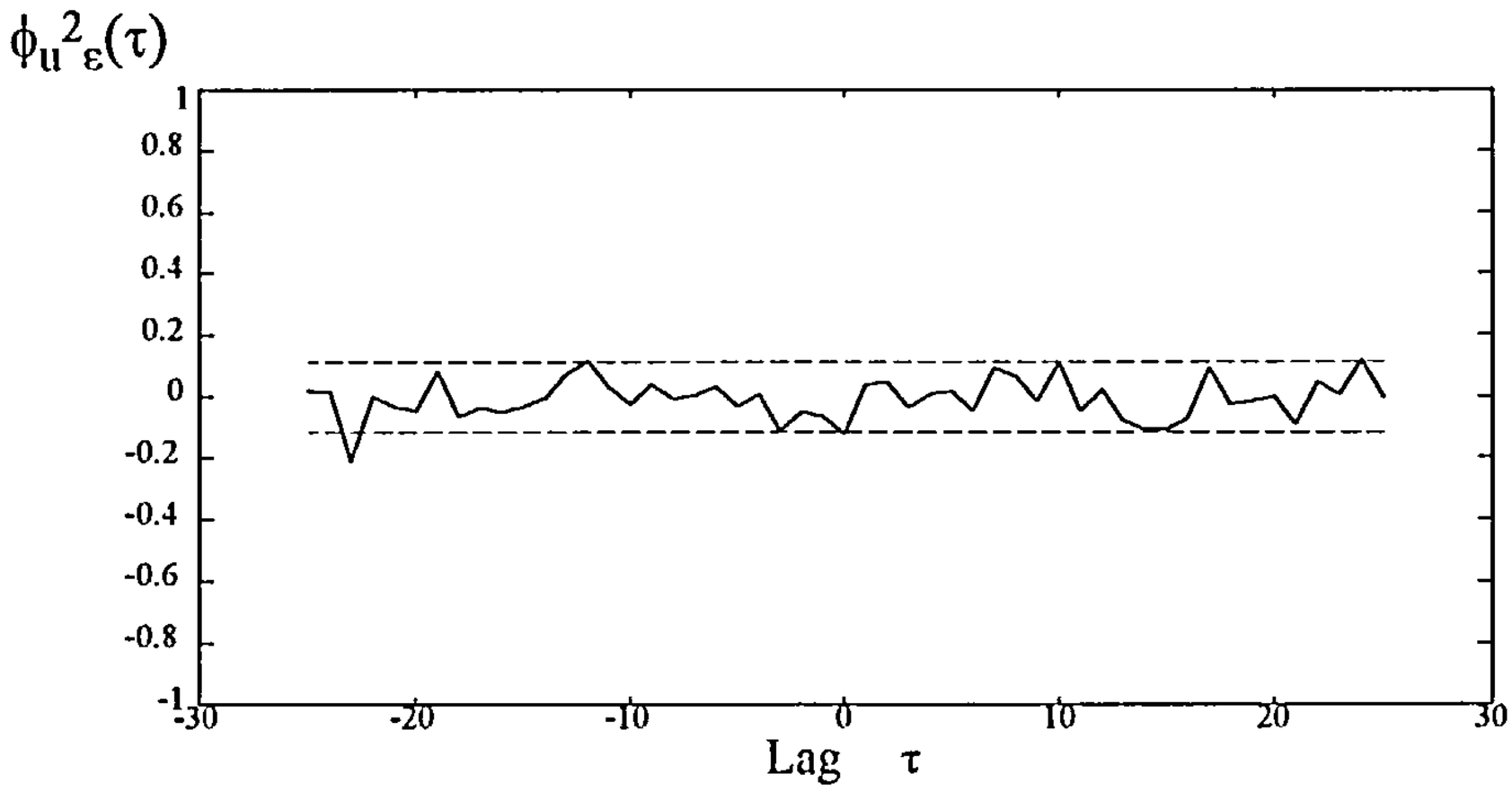


Figure 4.22 First order model test result for $\phi_u{}^2{}_\varepsilon(\tau)$

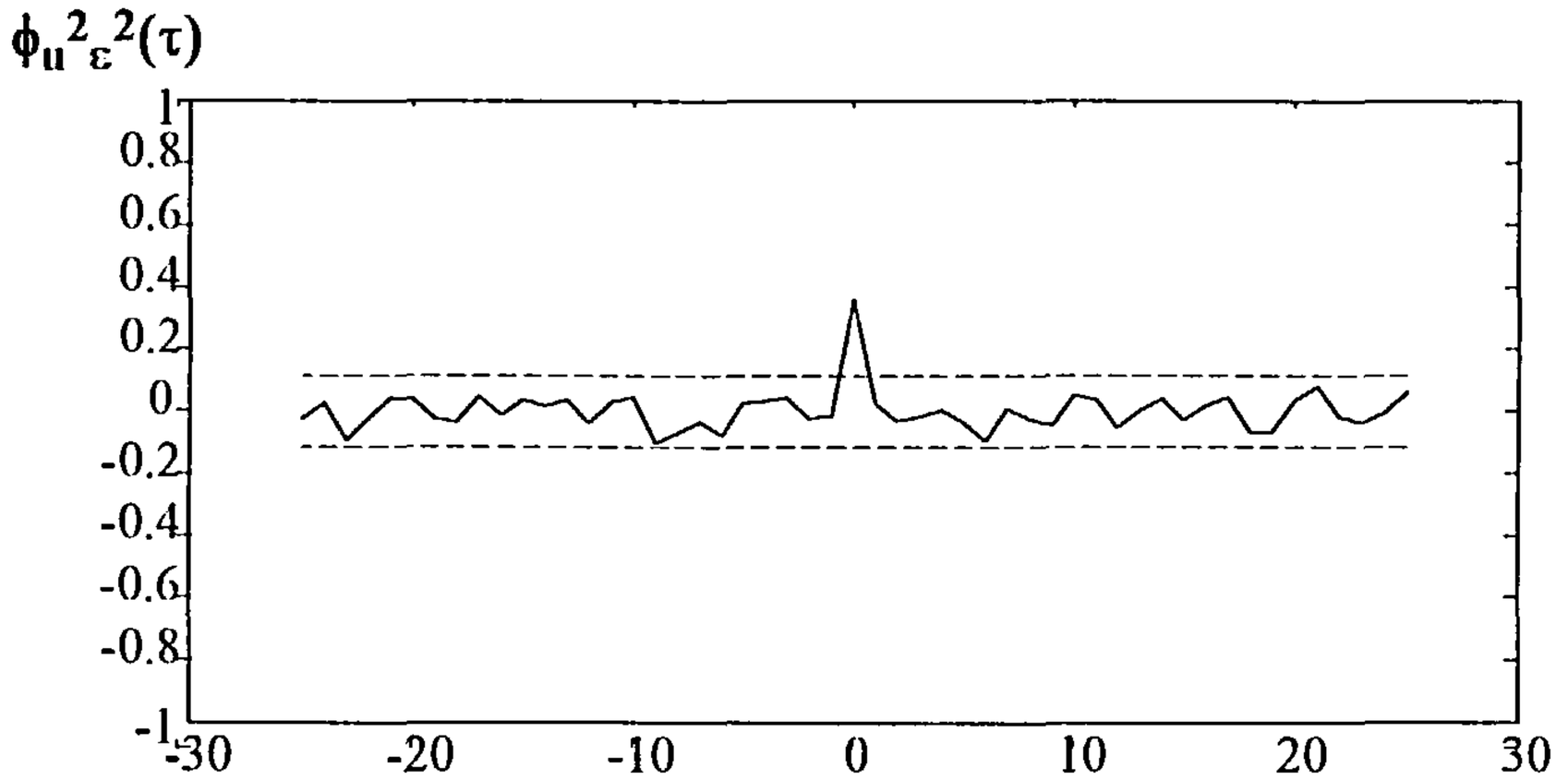


Figure 4.23 First order model test result for $\phi_u{}^2\varepsilon^2(\tau)$

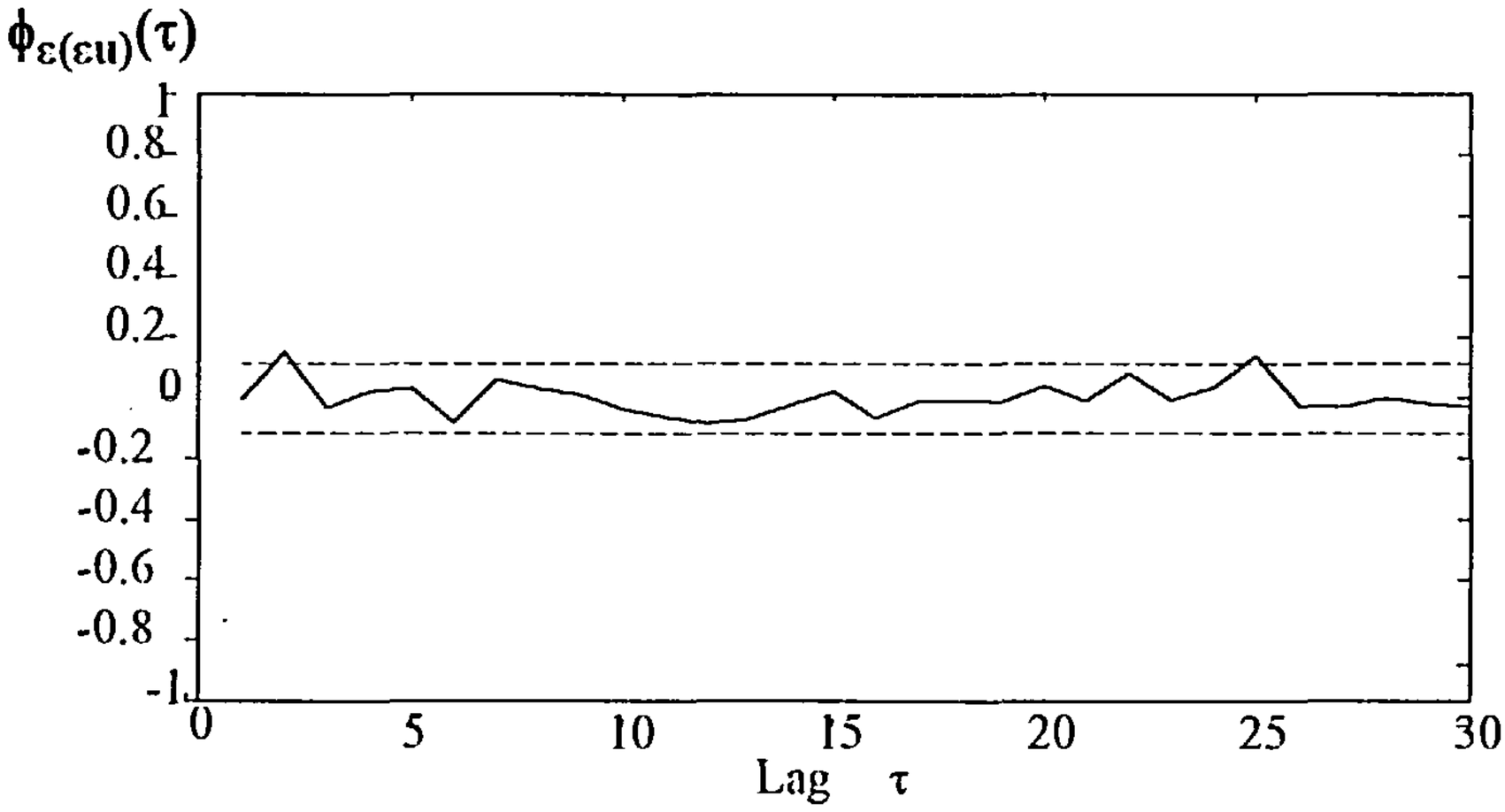


Figure 4.24 First order model test result for $\phi_{\varepsilon(\varepsilon u)}(\tau)$

When the correlation tests were performed on these two networks it was expected that the neural network configured as a third order input model would pass the tests. This was because the correlation tests detect if any lagged inputs are missing from the model and it had already been shown that a network configured as a second order input model was adequate for modelling the process, and these inputs were present in the third order model.

The results of these tests serve to demonstrate that any initial knowledge about the process under investigation should be used to formulate the input structure of a neural network model, as this will lead to the faster development of a neural network model capable of predicting accurate process performance.

4.4 SUMMARY

The feasibility of using a neural network to model a real laboratory scale process has been demonstrated. The data was obtained from the process and then used to train a neural network off-line. The non-linear process considered contained industrial standard sensor and actuator equipment making the process an ideal test bed on which to investigate the application of a neural network.

Although only 300 input-output data points were obtainable from the process in any one experimental run, compared to the 1000 used in the simulation studies, this did not cause any problems in the training of the neural network.

The neural network model of the process was thoroughly evaluated against the laboratory process using a set of test signals in both one-step-ahead predictor and recursive model configurations and it has been shown that, for both modes of operation, acceptable prediction accuracy was achieved. The process model input structure was also shown to contain the correct number of lagged process inputs and outputs using a correlation based technique. When using this method it is important to consider any outliers present in the prediction error sequence as these can have adverse affects on the results as was demonstrated.

The results presented in this chapter are very encouraging and demonstrate the ability of the MLP neural network to model a physical process. The use of this non-linear model should aid considerably in improving the control of the process and this is investigated in Chapters 5 and 6.

CHAPTER 5

CONTROL STRATEGIES AND CONTROL SIMULATION STUDY

5.1 INTRODUCTION

This chapter describes the inclusion of the neural network process model, developed in the previous chapters, into a control scheme. Several strategies exist that utilise a process model as an integral part and a choice has to be made as to which is most suited. The first section of this chapter reviews the main control schemes available to the designer and reported applications of the methods. In a number of the control strategies an inverse model of the process plays an important role. Hence, a brief section on obtaining inverse models via a neural network approach is presented along with a discussion on the viability of an inverse model. A detailed discussion of the control strategy that was implemented in this research is then presented along with results obtained in the process control simulation studies. Finally comparisons between a conventional three term (PID) controller and the chosen neural network based control strategy are investigated.

5.2 CONTROL STRATEGIES

Many control strategies exist where a parametric model of a process is included as a main part of the overall control structure. Recently a number of control strategies have been proposed where, in place of the standard parametric models (e.g. ARX and NARX), a neural network is used. The parameters then to be estimated are the weights within the neural network. It has also been suggested and demonstrated that, as well as replacing the process model with a neural network, the controller

can also be represented with a neural network (PSALTIS et al., 1990; CHEN and PAO, 1989; UNGAR et al., 1990; NARENDRA, 1990; BARTO, 1990).

In a number of control strategies the neural network model of the process is used to obtain the parameters (weights) of the neural network controller. When this method of obtaining the controller is used, the requirement of obtaining an accurate neural network process model is of paramount importance since if a poor process model is used in the chosen control strategy then an inadequate controller can be expected, resulting in overall poor control performance. In this work only the main control structures using neural networks that have been proposed were considered.

5.2.1 Model reference control

This control structure has been studied extensively by Narendra and Parthasarathy (1989) and is illustrated in Figure 5.1.

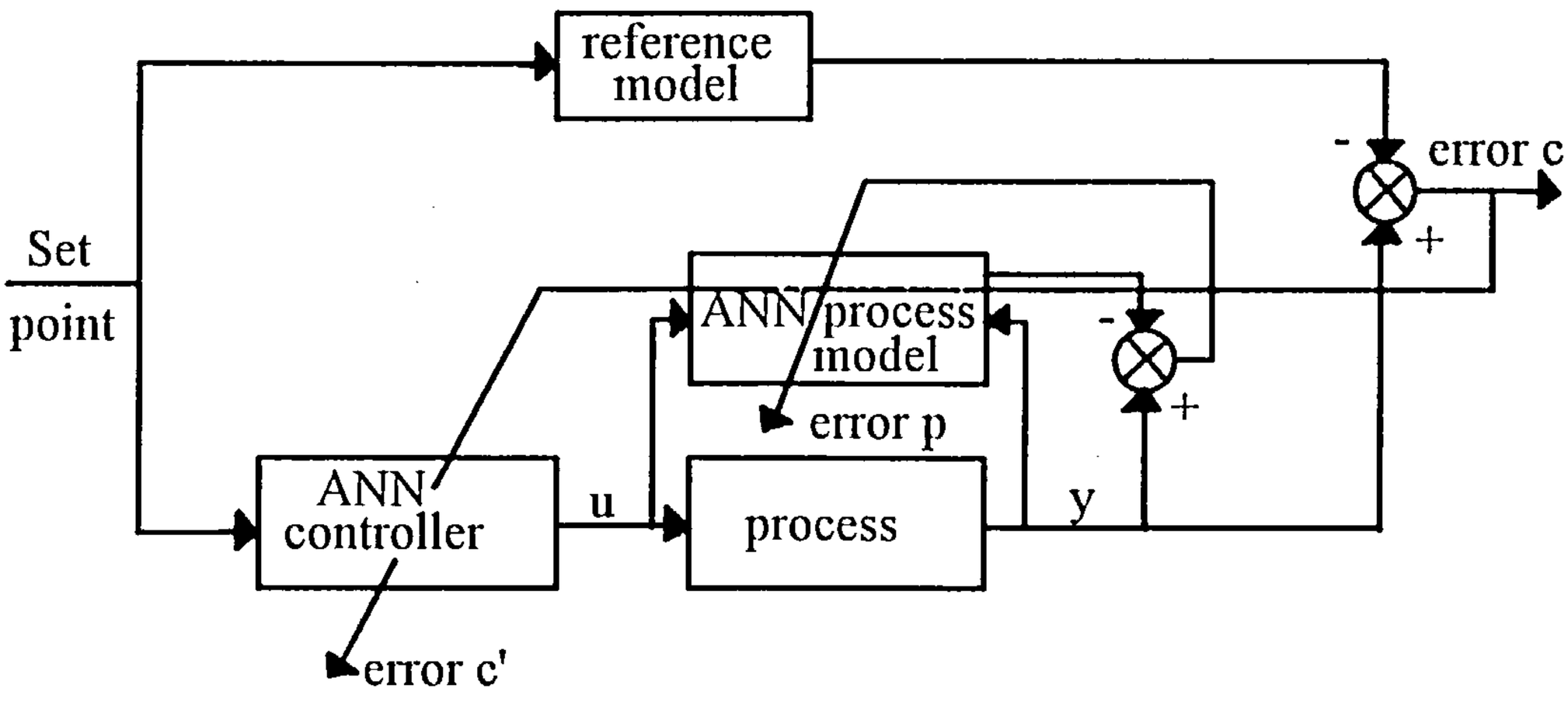


Figure 5.1 Model reference control structure

The objective is to train the neural network controller so that the control signal presented to the process causes the process output, y(t), to match the output of a pre-specified reference model.

The first step in implementing the control scheme is to obtain a neural network model of the process. This model is then used to find the controller error during training which in turn is used to update the controllers weights. This method of obtaining a neural network controller is known as indirect control and is required since there are no methods at the present time of updating the controllers weights based solely on the output error between the process and the reference model. The overall approach involved will attempt to force the neural network controller to be an inverse model of the process modified by the choice of reference model. This leads directly to the reliability of obtaining an inverse process model which is considered in section 5.3. The research by Narendra and Parthasarathy was conducted on a number of assumed single input, single output discrete time models of processes. The process models used were taken from general models which have been used for the identification of linear systems and were generalised to non-linear systems.

Sartori and Antsaklis (1992) also used a model reference control strategy on a number of simulated control problems. The method used to obtain the neural network controller was specialised inverse modelling described in section 5.3.2.

5.2.2 Direct Inverse control

Figure 5.2 illustrates the basic idea behind direct inverse control. The objective is to train a neural network to be the inverse of a non-linear process and then use this network in place of a standard controller as illustrated. If the neural network is a

true inverse of the process then an identity mapping will exist between the input and output of the overall control scheme.

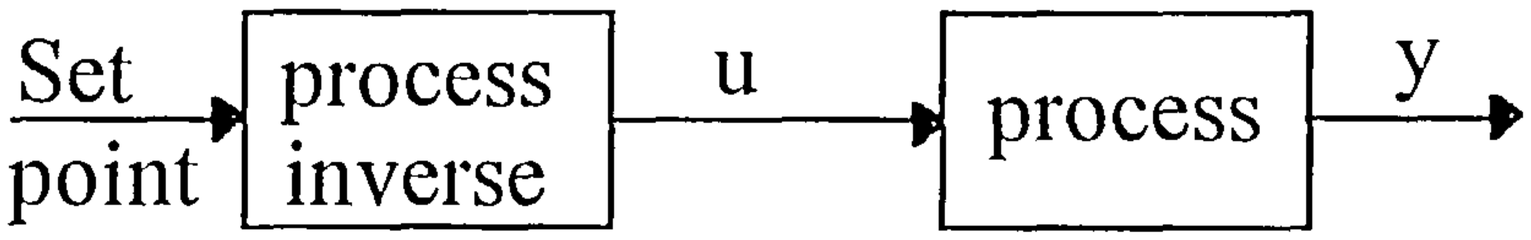


Figure 5.2 Direct inverse control structure

When obtaining the neural network inverse model of the process either the process itself can be used or if a neural network model of the process has already been obtained then this can be used in place of the actual process. Direct inverse control relies on an inverse model being obtainable and, since the scheme does not include any feedback this leads to questions about the overall robustness of the control structure.

Direct inverse control was investigated in simulations by Psaltis et al. (1988) and Elsley and Lan (1988), the former applying it to a robot in order to convert polar co-ordinates to Cartesian and the latter to problems of controlling a robot arm. The technique has also been applied on-line to a temperature control system (KHALID et al., 1992). Khalid et al. investigated the response of the control scheme to set point changes in temperature and also when load disturbances were artificially introduced to the process. Overall good set point tracking was obtained along with good load rejection properties when the range of the disturbance was $\pm 3^{o}C$.

5.2.3 Internal model control

This control structure relies on both a model of the process, sometimes called a forward model, and also an inverse process model. The control structure has been studied extensively for single input, single output linear discrete time systems (GARCIA and MORARI, 1982) and the ANN structure is illustrated in Figure 5.3. The approach of applying Internal Model Control, IMC, to the control of non-linear processes by replacing the standard linear models with neural networks has been investigated by Ungar (1990) who applied IMC to a simulation of a continuous flow stirred tank reactor.

Hunt and Sbarbaro (1991 and 1992) investigated IMC applied to the control of a pH process, and also to a discrete time process model proposed by Narendra and Parthasarathy (1990), the process model chosen was monotonic with respect to the process input and hence was invertible. Both of these investigations were performed in simulation.

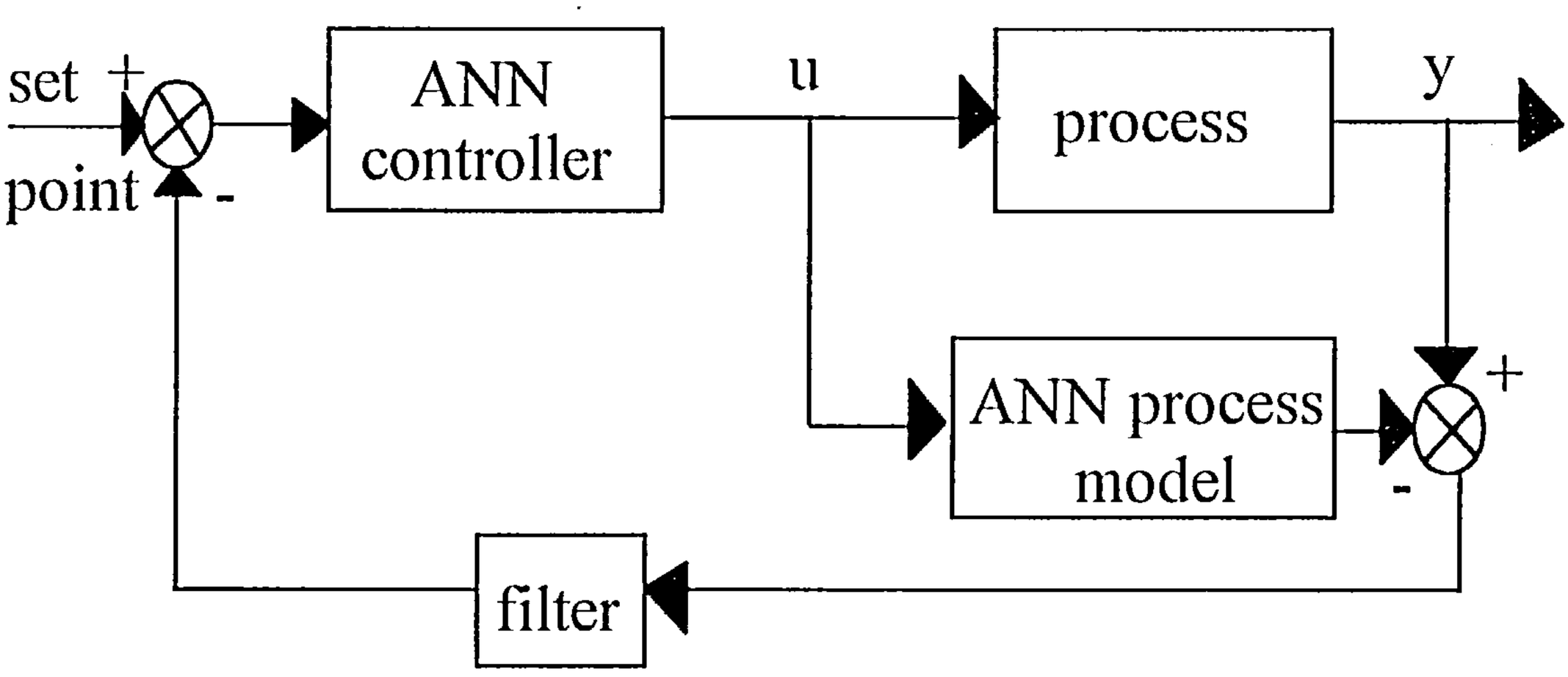


Figure 5.3 Internal model control structure

Two steps are usually involved in the method, the first step involves obtaining a neural network model of the non-linear process and the second step is to train a network to be the inverse of the process. The inverse model is then used as the controller. Either the process or the neural network model of the process obtained in the first step can be used to obtain the inverse model. In some situations it is advantageous to use the neural network model, as discussed in section 5.3.

In general the model of the process will not be an exact replica and process model mismatch will occur, this is accommodated for by the use of a filter in the control scheme, as shown in Figure 5.3, which makes the system robust to the mismatch (MORARI and ZAFIRIOU, 1989).

5.2.4 Predictive control

A neural network predictive control scheme is illustrated in Figure 5.4 and is seen to consist of a neural network process model, a non-linear optimiser, delay and filter blocks. The objective is to use an algorithm in the non-linear optimiser that computes future controller outputs which minimise deviations between the process output and the required set point. The structure of the control scheme enables multi-step-ahead predictions to be performed which has the advantage of anticipating where the process output is 'heading'. Thus the scheme enables early corrective action to process disturbances and good set point tracking to be achieved.

Bhat et al., (1989) investigated the technique in order to find process inputs that would minimise the error between required and predicted pH values in a simulated CSTR. The predictive control scheme was also used to obtain a controller capable of backing a simulated trailer truck to a loading dock (NGUYEN and WIDROW,

1990). Willis et al., (1991) used a neural network model of a non-linear exothermic CSTR studied by Economou and Morari (1986), to investigate the control scheme.

The control of an inverted pendulum via a neural network model approach was examined using the predictive control scheme by Fong and Loh (1991). Multi-step-ahead predictions were performed with a prediction horizon of 100. The simulation studies revealed slight offsets in the required upright position but overall good control performance was obtained.

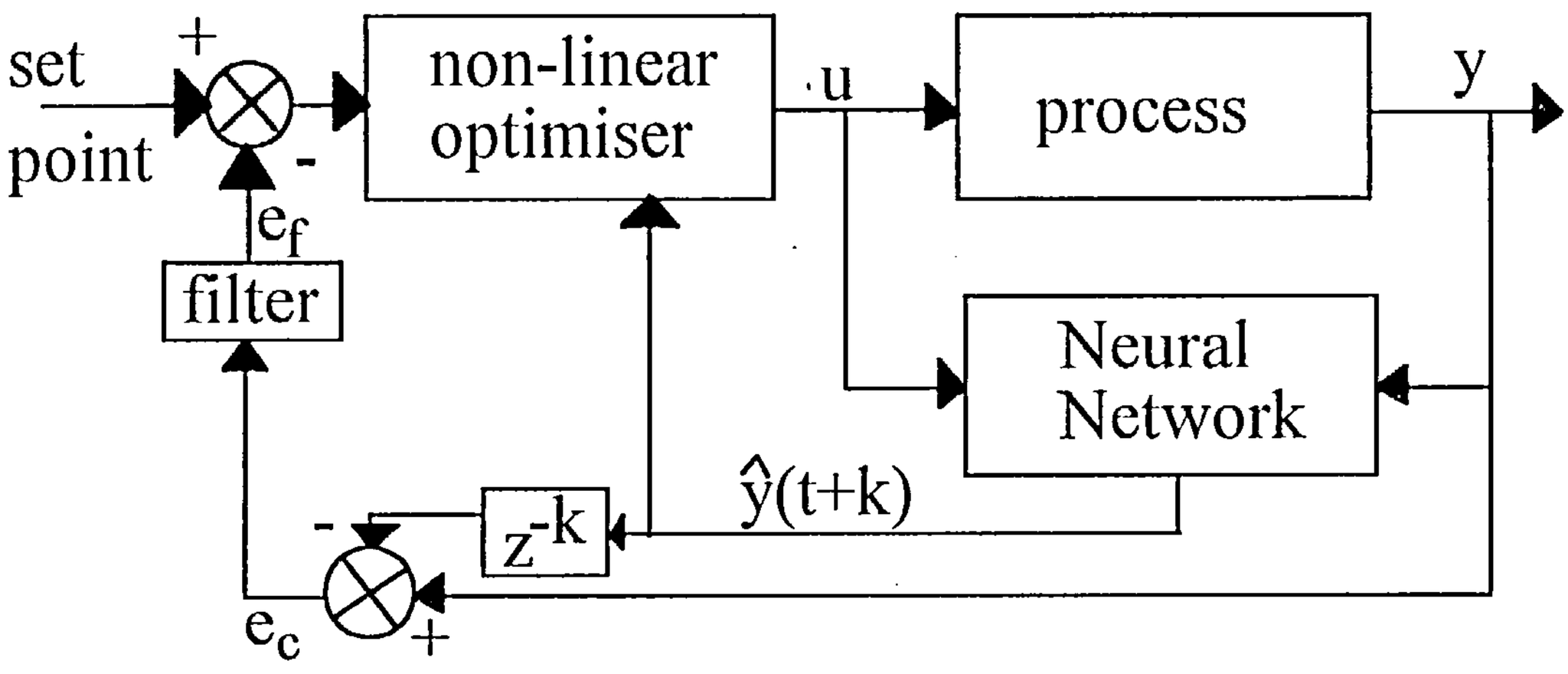


Figure 5.4 Neural predictive control structure

The main advantages of this control scheme are that an inverse model of the process does not have to be found and multiple-step-ahead predictions can be performed. Process constraints can also be included in the scheme by appropriate modifications of the optimisation cost function.

5.3 INVERSE MODEL IDENTIFICATION

From the control schemes described in section 5.2 it is evident that the majority of the schemes are dependent upon an inverse model of the process being obtainable. In order to obtain an inverse model two methods have been proposed and these methods are now discussed.

5.3.1 Direct inverse modelling

This approach to obtaining an inverse process model using a neural network is the simplest approach and is illustrated in Figure 5.5.

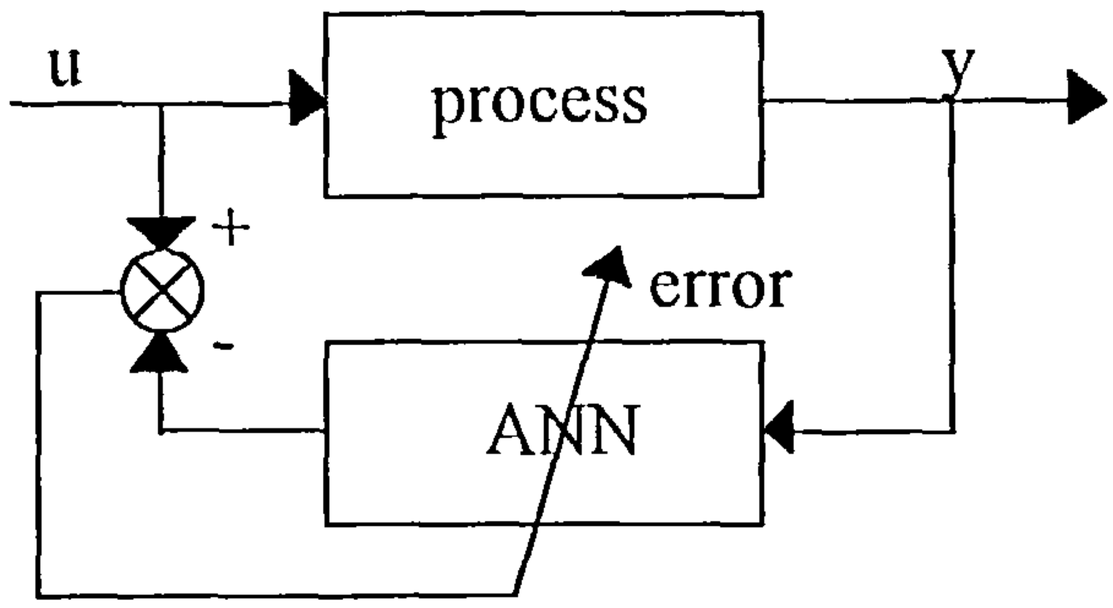


Figure 5.5 Direct inverse modelling

An input is applied to the process and a corresponding output is produced, this output is then fed to the input of a neural network together with lagged input-output data and the output of the network is compared to the process input to produce an error signal. This error signal is then used to update the weights of the neural network. The scheme will clearly attempt to force the neural network to be the inverse of the process. The disadvantage of this method is that the scheme is not goal directed (JORDAN and RUMELHART, 1991), that is to say that the

training input data for the network does not correspond to the actual required response.

5.3.2 Specialised inverse modelling

Specialised inverse modelling (PSALTIS et al., 1988) overcomes the disadvantage associated with direct inverse modelling by placing the neural network inverse model ahead of the process and using the error between the process output and the required output to update the weights. Using the method proposed by Psaltis et al., requires a knowledge of the plant Jacobian in order to adjust the network weights. A scheme proposed by Jordan (1988) incorporates a neural network model of the process into the learning scheme and is illustrated in Figure 5.6.

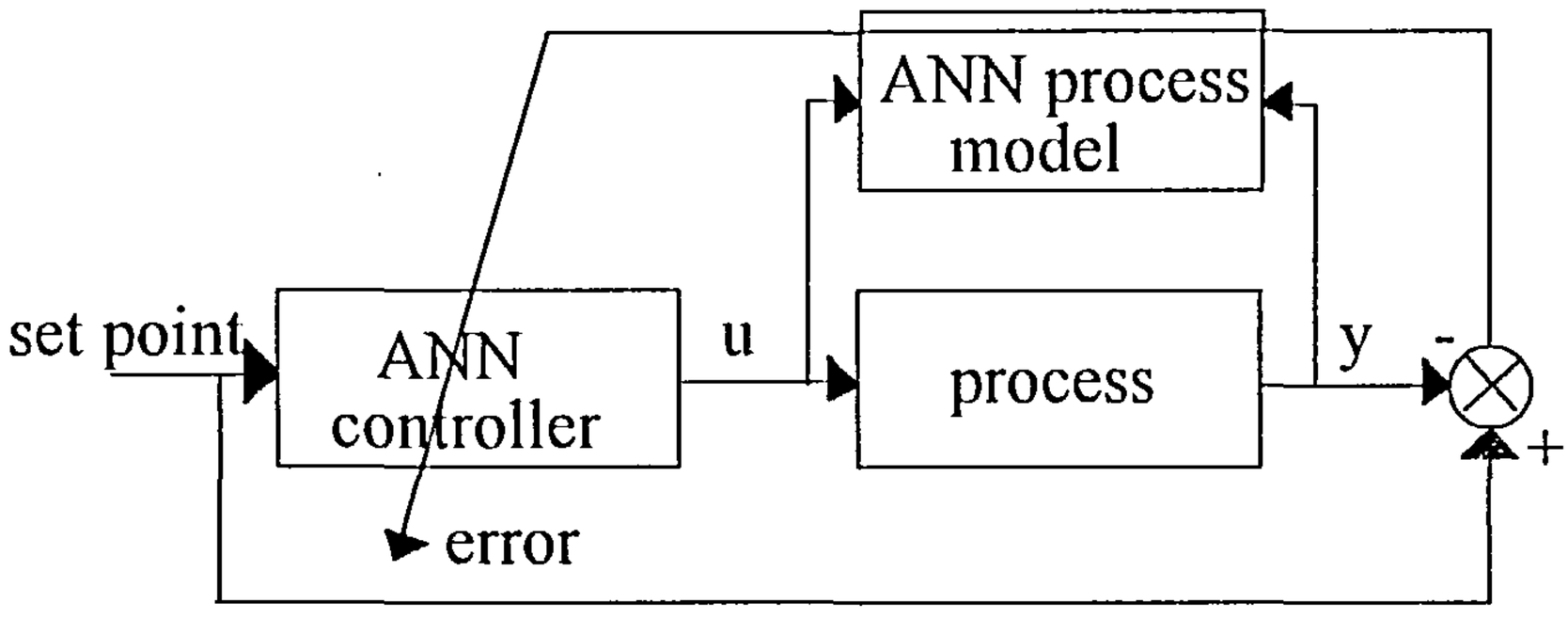


Figure 5.6 Specialised inverse modelling

The method then used to obtain the error in order to update the controller weights is to back-propagate the overall process output error back through the forward neural network process model. This then results in the error at the output of the controller which in turn is used to update the controllers weights. This method is the same as that proposed by Werbos (1974) which was called 'back propagation

through the model'. This observation leads to another advantage of specialised inverse modelling over the direct inverse approach in that it can be used on-line to train a controller. This scheme is very similar to the model reference control structure (compare Figures 5.6 and 5.1) and in fact is identical if the reference model in Figure 5.1 is the identity mapping.

5.3.3 Viability of an inverse model

When obtaining an inverse process model it is very important to consider the process being modelled. Since the overall performance of the control scheme will be a function of the accuracy of the inverse model the ability of obtaining the process inverse has to be considered. If the process under investigation does not have a one to one mapping then an incorrect inverse model will result leading to incorrect control.

5.4 CHOICE OF ANN CONTROL STRATEGY

As described in section 5.2 there are a number of control strategies within which a neural network model of a process can be used. In this research the control structure to be investigated was neural predictive control. There were a number of reasons for this choice which are now described.

The neural predictive control strategy only requires a neural network model of the process which had already been obtained. It has also been illustrated that using the SEDM method of conditioning the process data, when training the neural network model of the process, that accurate process predictions were obtained when the

network was used recursively. Hence, the network will be ideally suited in the control structure when multi-step-ahead predictions are performed.

The requirement of not having to obtain an inverse process model also suits the control structure to control highly non-linear processes as occurs, for example, in the chemical industries. The use of feedback in the predictive control strategy accommodates for plant-model mismatch which makes the scheme more robust than other schemes without feedback such as direct inverse control.

A possible disadvantage of the neural predictive control scheme is that a non-linear optimisation algorithm is required to solve the required input to the process, and this possibly makes the scheme more computationally demanding over the other methods described. However, there are a number of algorithms available for solving the optimisation problem and with the computing power available today this should cause little inconvenience.

5.5 THE ANN PREDICTIVE CONTROL STRUCTURE

The predictive control scheme investigated in this research is illustrated in Figure 5.4. The optimiser functions as the controller in the scheme and produces the control signal to manipulate the process input. The optimisation algorithm was used to minimise the following conventional cost function:

$$J(N1,N2,Nu)=\sum_{i=t+N1}^{t+N2}\left[y_r(i)-y_m(i)\right]^2+\sum_{i=t}^{t+Nu}\lambda(i)\left[u(i)-u(i-1)\right]^2 \qquad ...(5.1)$$

where y_r is the required process output or set point, y_m is the neural network model predicted output, u is the process input, N1 and N2 define the prediction horizon, Nu is the control horizon and λ is a sequence of control weighting factors. The process data used within the cost function is firstly normalised so that both terms have equal weighting in the evaluation of J(). Without this step problems will occur when the input and output data span a different range. A suitable choice for N1 is to make it equal to the process delay between input and output, since for a process with a delay of k samples between input and output there will be no corresponding change in the output for this duration and these computations are unnecessary (CLARKE et al., 1987). The parameter N2 is then set to define the prediction horizon beyond this point. The optimisation algorithm is considerably simplified if the control horizon Nu is set equal to zero and this was the case throughout these investigations. The control weighting factor λ is used to penalise large changes in the control input and is usually set at a constant in the range [0,1].

The feedback loop in the control scheme accommodates for any differences between the process output and the model that will occur in practice. The filter used in the feedback loop was a unity gain, low pass digital filter as described by:

$$\frac{e_f}{e_c} = \frac{1-\beta}{1-\beta z^{-1}} \qquad \ldots(5.2)$$

where e_c is the process-model mismatch error, e_f is the filtered error signal, β is a constant parameter ranging between zero and one which affects the time constant of the filter, and z^{-1} is the unit delay operator. Equation 5.2 was implemented by the difference equation:

$$e_f[t] = (1-\beta)e_c[t] + \beta e_c[t-1] \qquad \ldots(5.3)$$

It is seen from equation 5.3 that as $\beta \rightarrow 0$ the filtering action is attenuated. As well as improving the robustness of the scheme to noise and other disturbances, the filtered error is also used to correct the predictions from the neural network model, for use in the cost function, to achieve zero steady state offset in the control as described by Willis et al., (1992). The value of β was chosen empirically from simulation studies to achieve a stable closed loop response without unduly slowing down the process response to set point changes. This resulted in a suitable value for β=0.8 which corresponded to a filter time constant of approximately 1/(1-β) or five sample periods.

The operation of the neural predictive control strategy is as follows. At each time step the process output is sampled and the error and filtered error between the process output and the neural network process model is computed. The neural network inputs are initialised to current and previous values of the process output and input and a chosen value of u(t). The next N2 outputs of the neural network model are obtained and the cost function J() is computed. Network initialisation and calculation of J() are repeated with different values of u(t), obtained by the optimisation algorithm, and the value of u(t) that minimises the cost function is selected and applied to the process input. The sequence of actions is summarised below

Step 1: Use the neural network model of the process to generate the next N2 predicted process output values.

Step 2: Obtain the required set point values for the next N2 sample times.

Step 3: Calculate the predicted deviations from the required set point using the

values obtained in steps 1 and 2

Step 4: Find the optimal control input u(t) by minimising the cost function J.

Step 5: Apply u(t) to the process and go back to step 1 at the next sampling instant.

The neural predictive control strategy is an extension of the generalised predictive control algorithm, GPC, (CLARKE et al., 1987), which in turn is an extension of the generalised minimum variance algorithm, GMV, (CLARKE and GAWTHROP, 1975, 1979). GMV and GPC are used in self tuning control systems where a linear model of the process is estimated at each sample time. The extension of GPC to non-linear systems is implemented by replacing the linear model estimated at each sample time with the neural network model of the process.

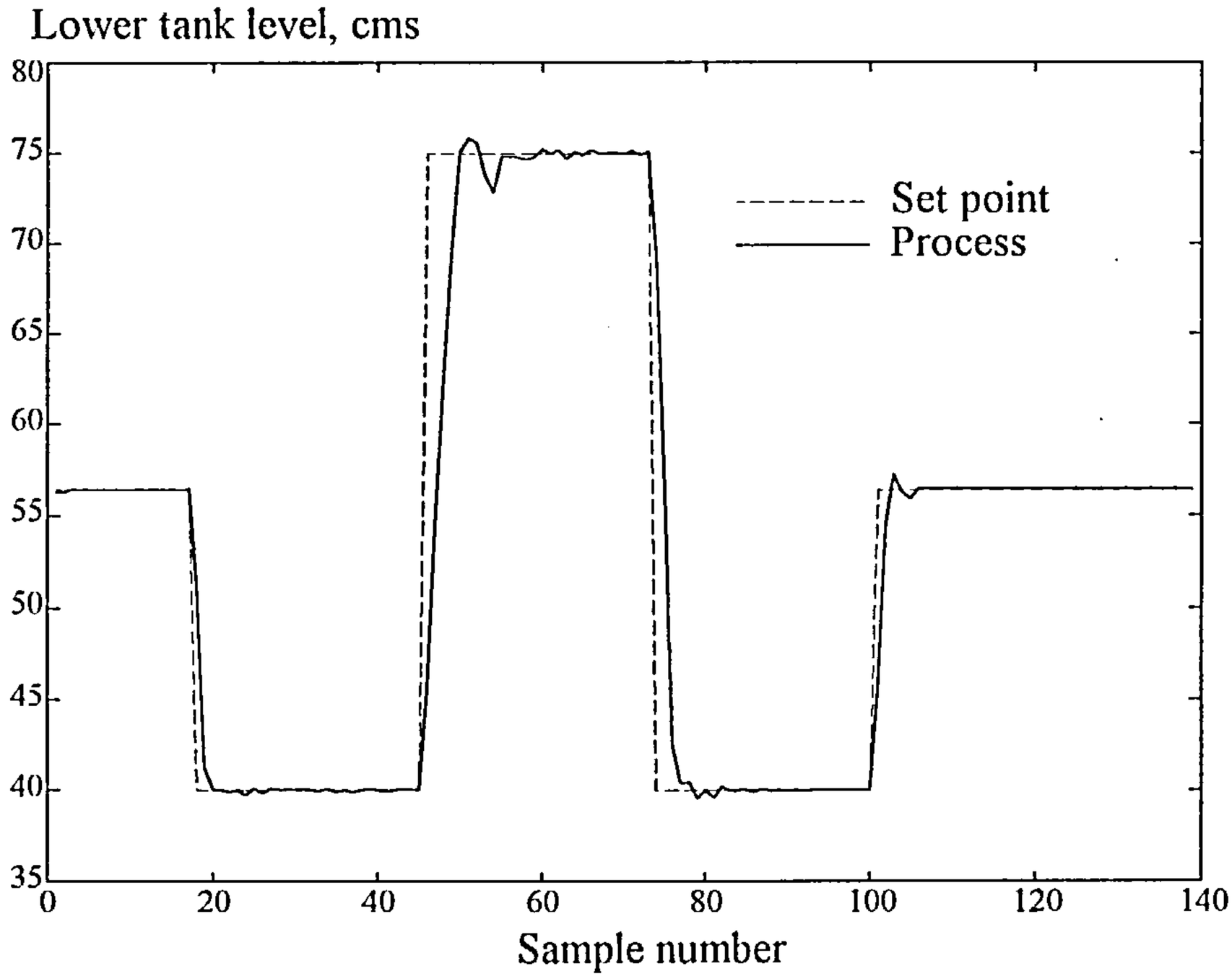
The non-linear optimisation algorithm employed to find the value of control input that minimises the cost function was an interval halving one. The algorithm used is simple and easy to implement and it is also very stable (LUYBEN, 1973). Other non-linear optimisation algorithms exist which are suitable for this task such as the Fibonacci search, golden section search and Nelder-Mead (ADBY and DEMPSTER, 1974) however the interval halving technique was found to be adequate.

As mentioned in Chapter 3, the range of the digital signals from the computer to the process valve are from 100 to 200. If an exhaustive search were performed to minimise the cost function then this would require the 101 different values of u to be implemented in the cost function and the cost function evaluated for each value. Using the interval halving algorithm results in a maximum of eight values of u before the minimum is found for an initial step size of sixteen which obviously speeds up the computations involved considerably.

5.6 NEURAL PREDICTIVE CONTROL SIMULATION STUDIES

The neural predictive control scheme was firstly implemented in an ACSL program for FPMODEL2 where investigations into the control performance for changes in the set point were performed. The set point changes in liquid height were such as to force the process over a wide region of operation within the range of the identified neural network model. The performance of a tuned PID controller was also investigated and the ability of the PID algorithm to control the liquid level process is presented in section 5.7.

Initial simulations were performed for one-step-ahead prediction, resulting in the value of N2 in the cost function being set to one, and the effect of changes in the weighting factor λ on the process response was observed. Figures 5.7 and 5.8 show the results of the simulation of the process, and the control input, when the height of liquid in tank two was required to follow a set of step inputs, with λ equal to zero and one respectively. With the value of λ set equal to one the ability of the process to track the set point becomes more sluggish compared to the response of the process when λ was set equal to zero. The reason for this can be seen when the control input to the process valve is observed for both cases. With λ equal to zero there is no constraint on the control input between its upper limit (200) and its lower limit (100), Figure 5.7. However, with λ set at one, large movements of the control input are penalised in the cost function resulting in a high cost value, Figure 5.8. The resulting control input to the process valve is nevertheless a more desirable one compared to the control input with λ equal to zero where it is observed that movement in the valve is erratic. A range of different values for λ between zero and one were investigated and the process response and control effort were monitored with N2 equal to one.



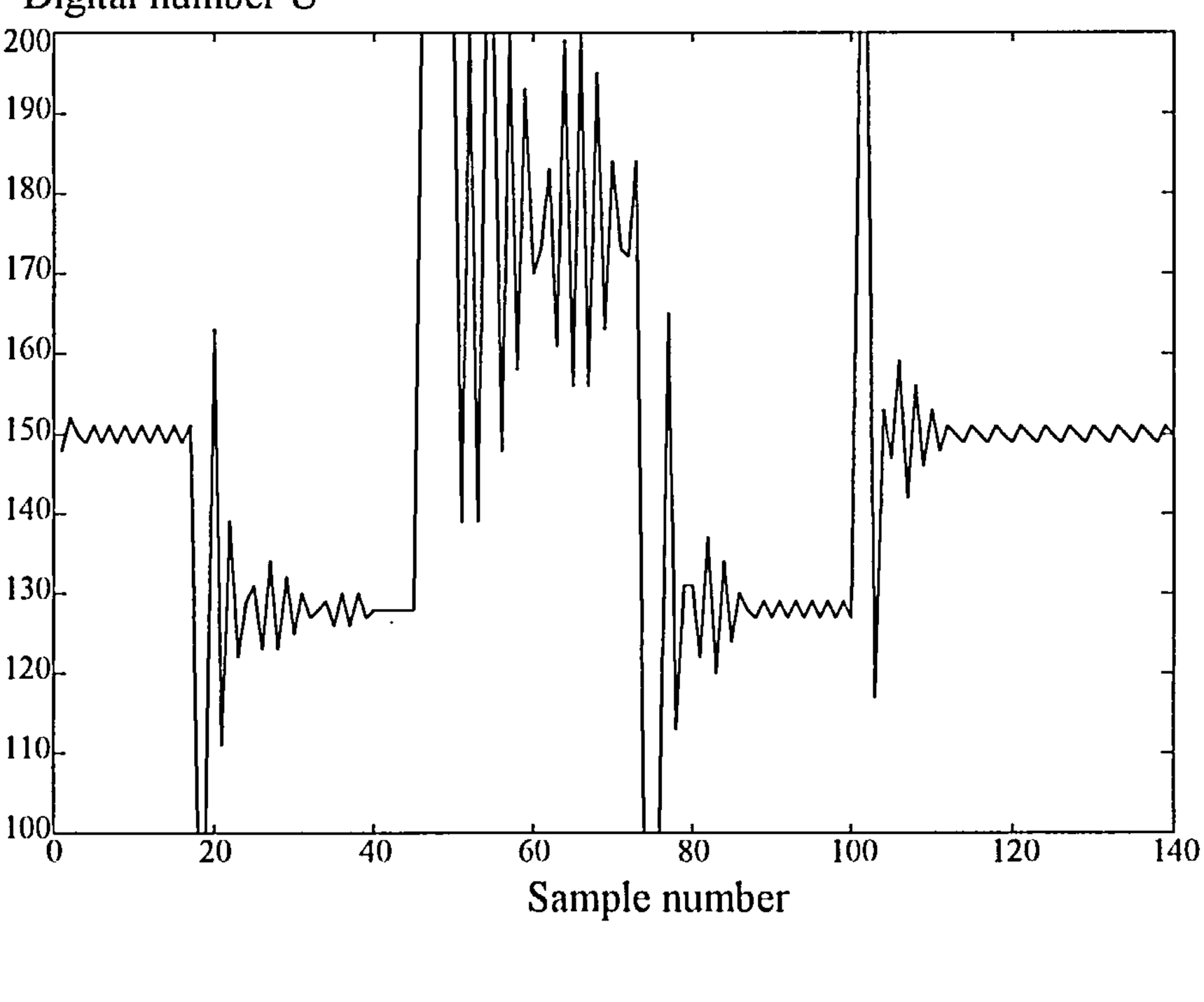
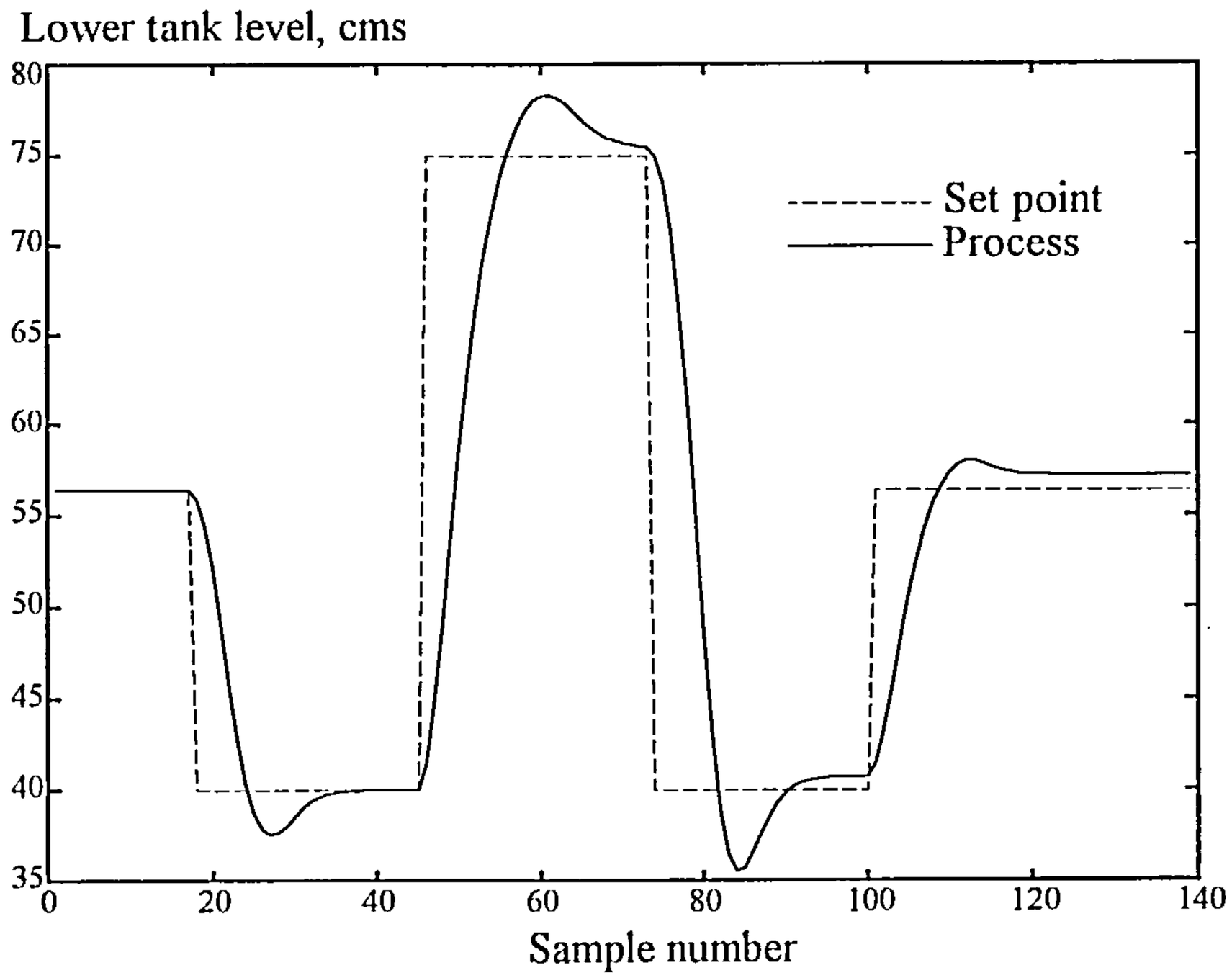


Figure 5.7 Process response and control input, λ=0 and N2=1



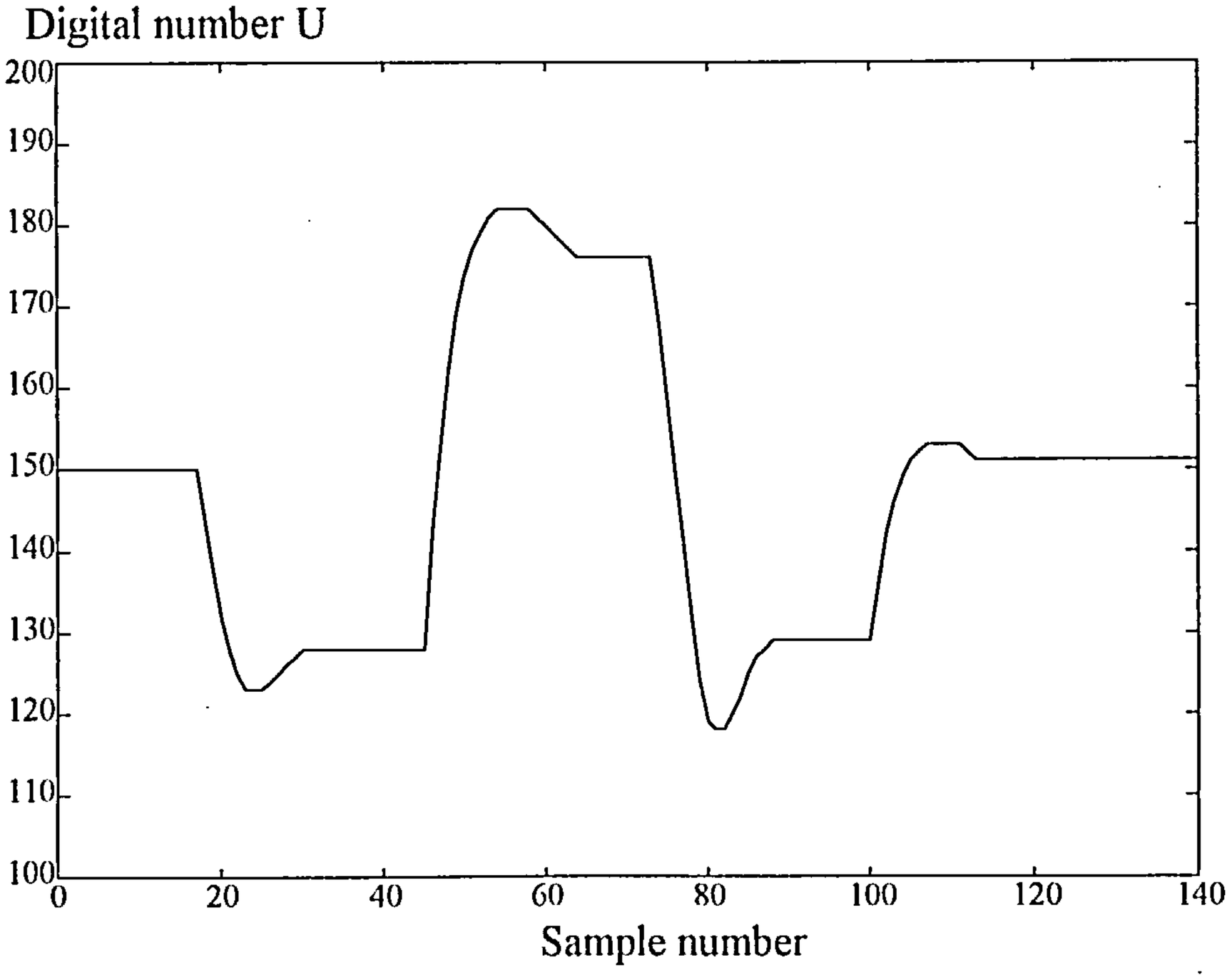


Figure 5.8 Process response and control input, λ=1 and N2=1

It was found that as the value of λ was increased from zero to one the process response time deteriorated and the control signal to the process valve became smoother. Table 5.1 shows the value of λ used and the corresponding movement in the control input. A measure of the movement in the control input was taken to be the control effort (CE) calculated by:

$$CE = \sum_{t=2}^{n} |U(t) - U(t-1)| \qquad \ldots(5.4)$$

where n is the number of samples in the simulation experiment The simulations were each of 140 samples duration.

λ	control effort,CE
0.0	1788
0.2	670
0.4	392
0.6	333
0.8	239
1.0	187

Table 5.1 Control effort versus weighting factor, λ, for N2 equal to one

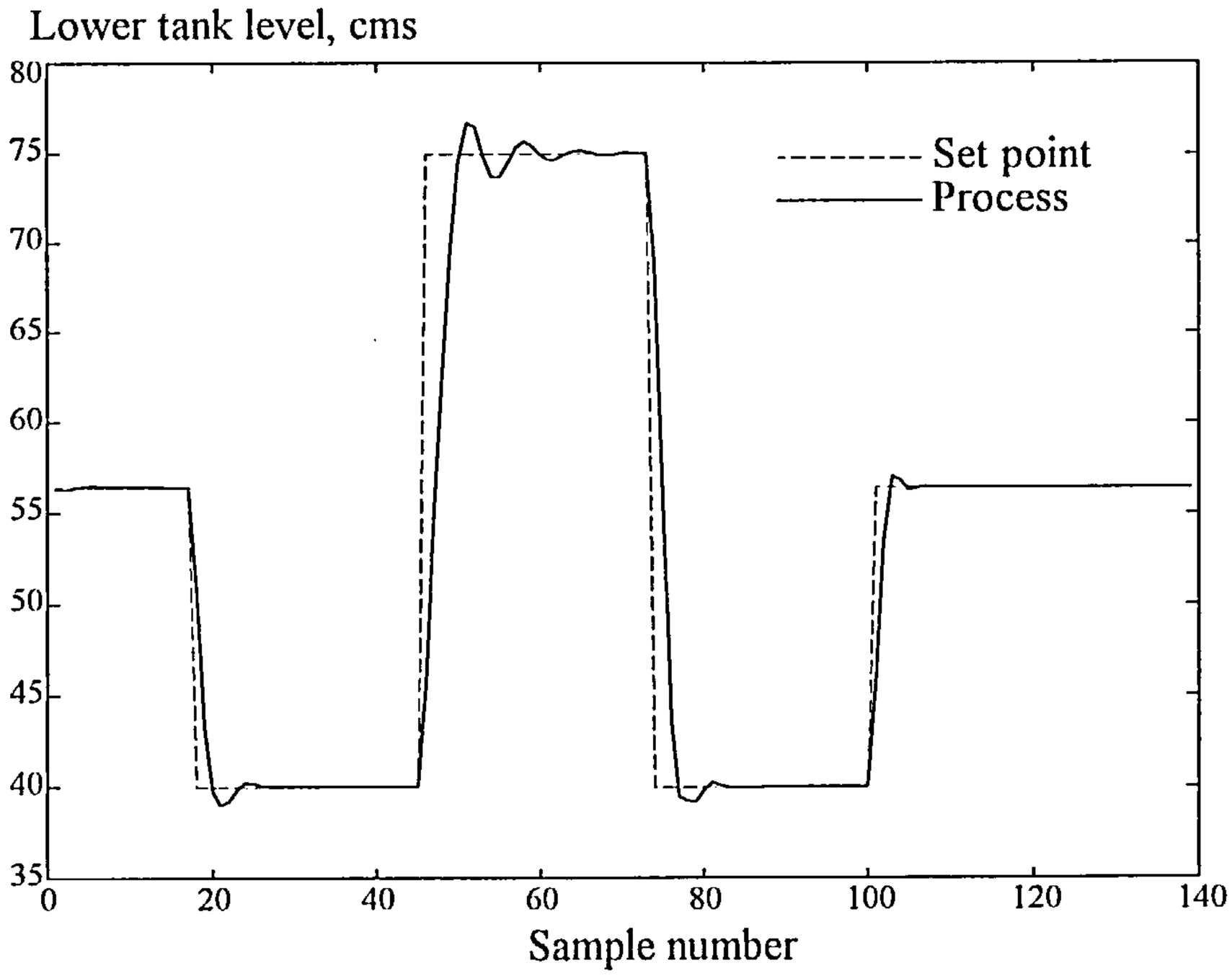
Control of the liquid level process was then investigated with the neural network model of the process used to predict multiple time steps ahead. The value of λ was kept at zero and the prediction horizon N2 was set at values of 2, 3, 4 and 7 sample periods corresponding to 1.2 minutes, 1.8 minutes, 2.4 minutes and 4.2 minutes respectively. The required set point was the same set of steps as used previously

for the one-step-ahead prediction results. Figures 5.9 to 5.12 illustrate the required set point for the height of liquid in tank 2, h_2, and the corresponding response of the process for the different prediction horizons as well as the control signal for each test. In each case it can be seen that very accurate set point tracking is achieved in the steady states and that the process rise time for prediction horizons of 2, 3 and 4 sample times are all very similar. With N2 equal to 7 the response of the process is more sluggish and similar in nature to when one-step-ahead prediction was performed with λ equal to one. From Figures 5.9 to 5.12 it is seen that as the prediction horizon increases the control input becomes smoother in nature and that, even with N_2 equal to two, the control effort is greatly reduced compared to when N2 was set at one, whilst still maintaining an accurate response to set point changes.

When λ was set equal to one and the same prediction horizons of 2, 3, 4 and 7 were used the ability of the process to follow the set point is illustrated in Figures 5.13 to 5.16 where the corresponding control input signal is also shown. Table 5.2 shows the control effort, calculated using equation 5.4, for each of the prediction horizons with λ set equal to one and zero respectively.

N2	Control effort λ=0	Control effort λ=1
2	529	316
3	375	338
4	295	315
7	205	207

Table 5.2 Control effort versus λ for various prediction horizons



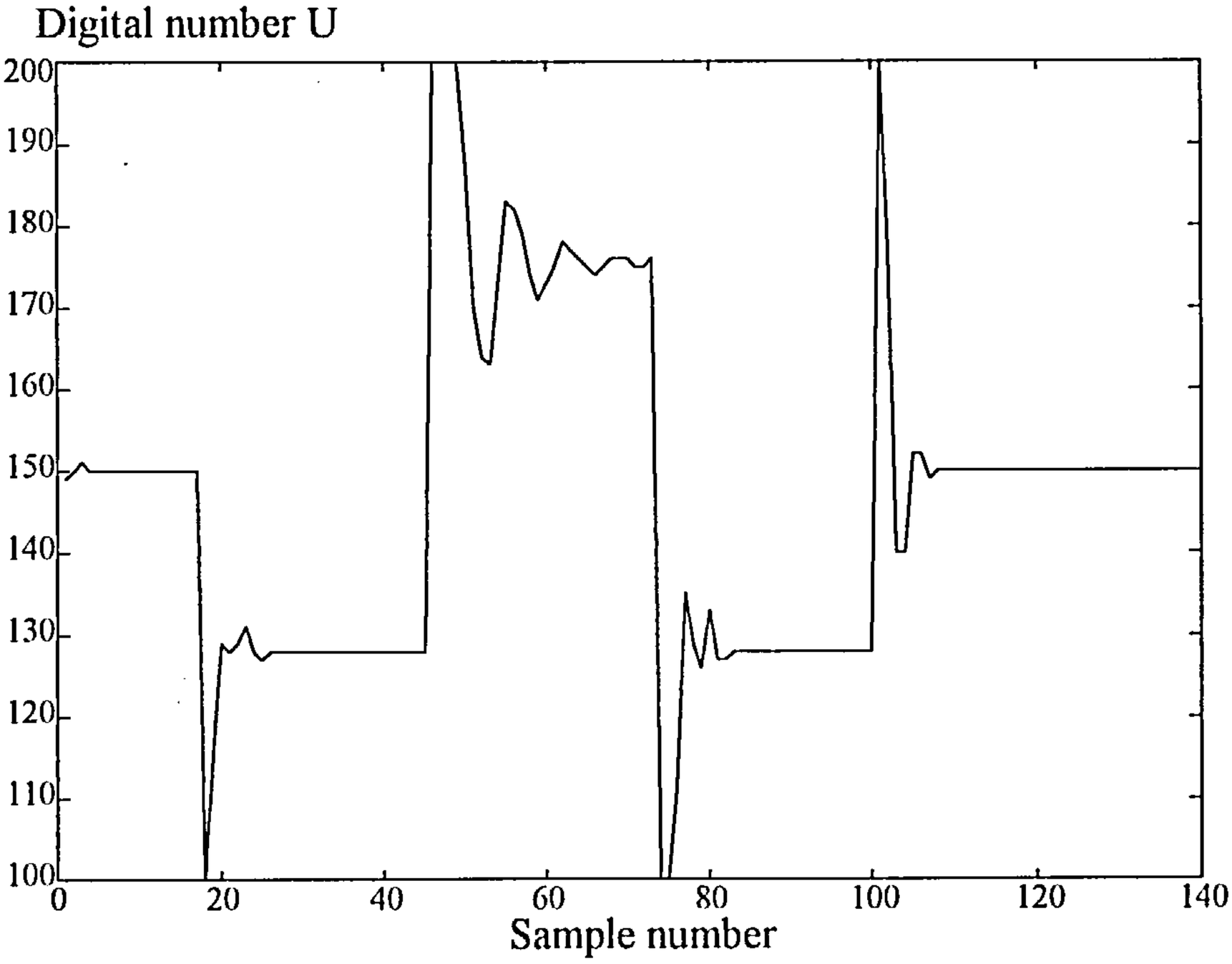
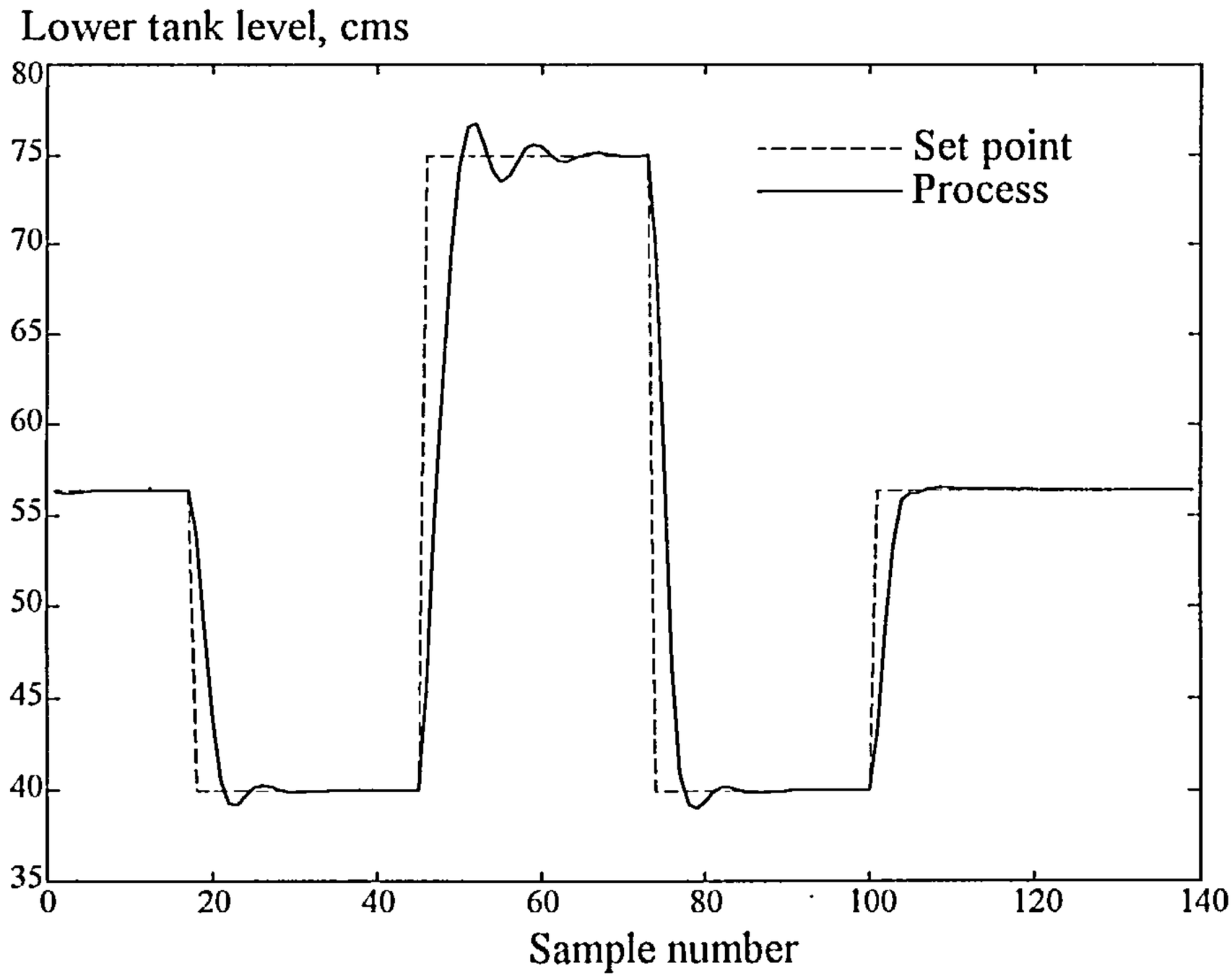


Figure 5.9 Process response and control input, λ=0 and N2=2



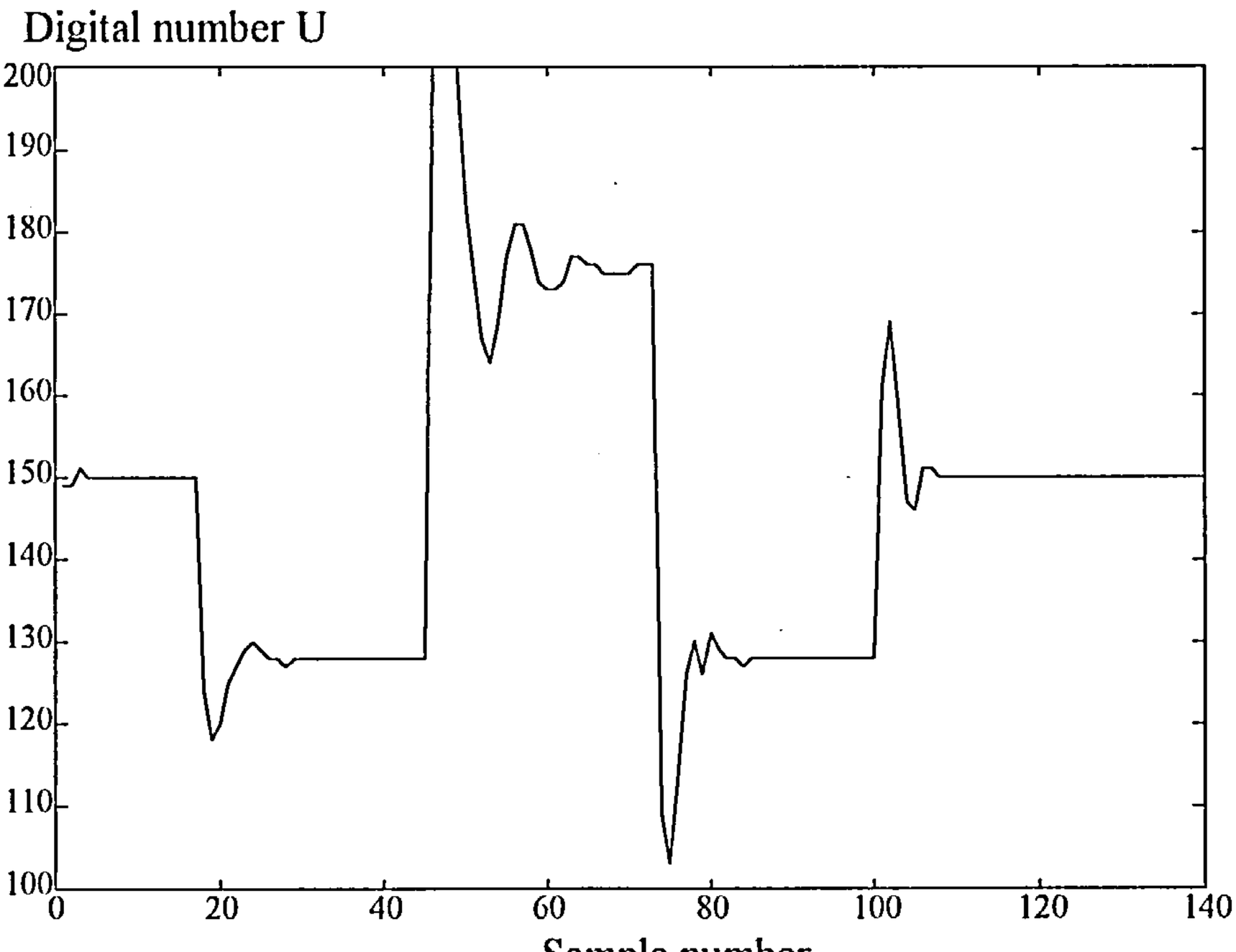
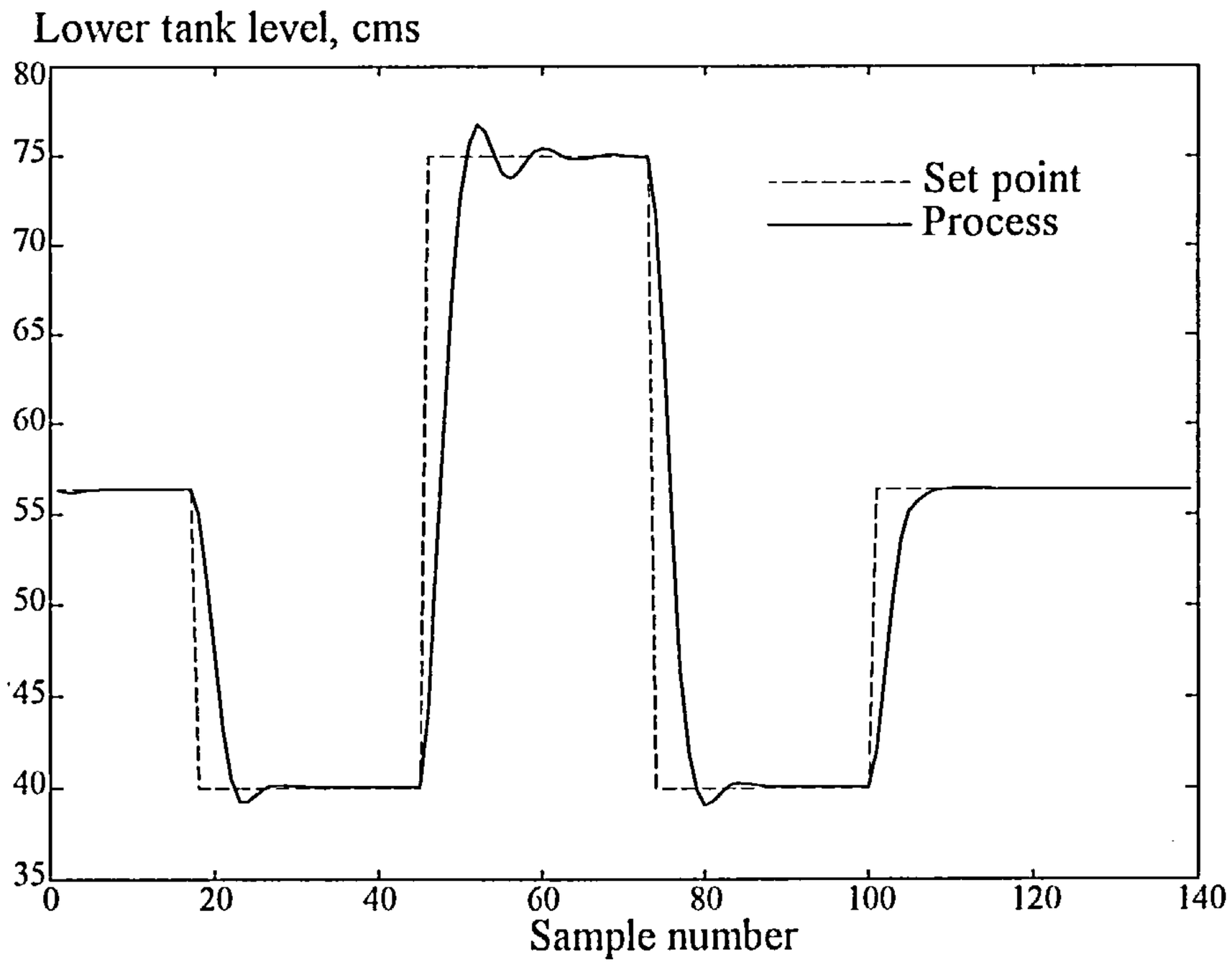


Figure 5.10 Process response and control input, λ=0 and N2=3



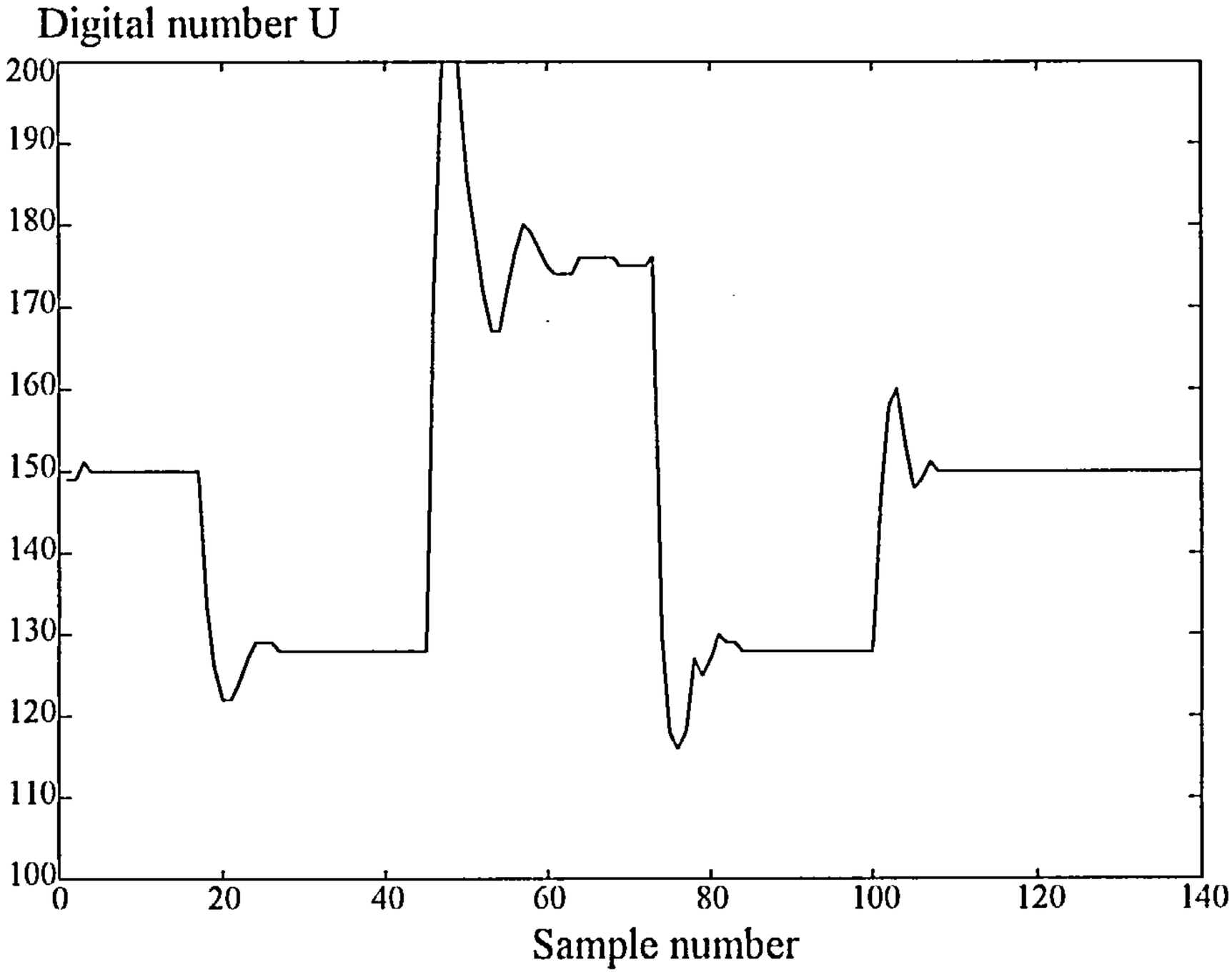
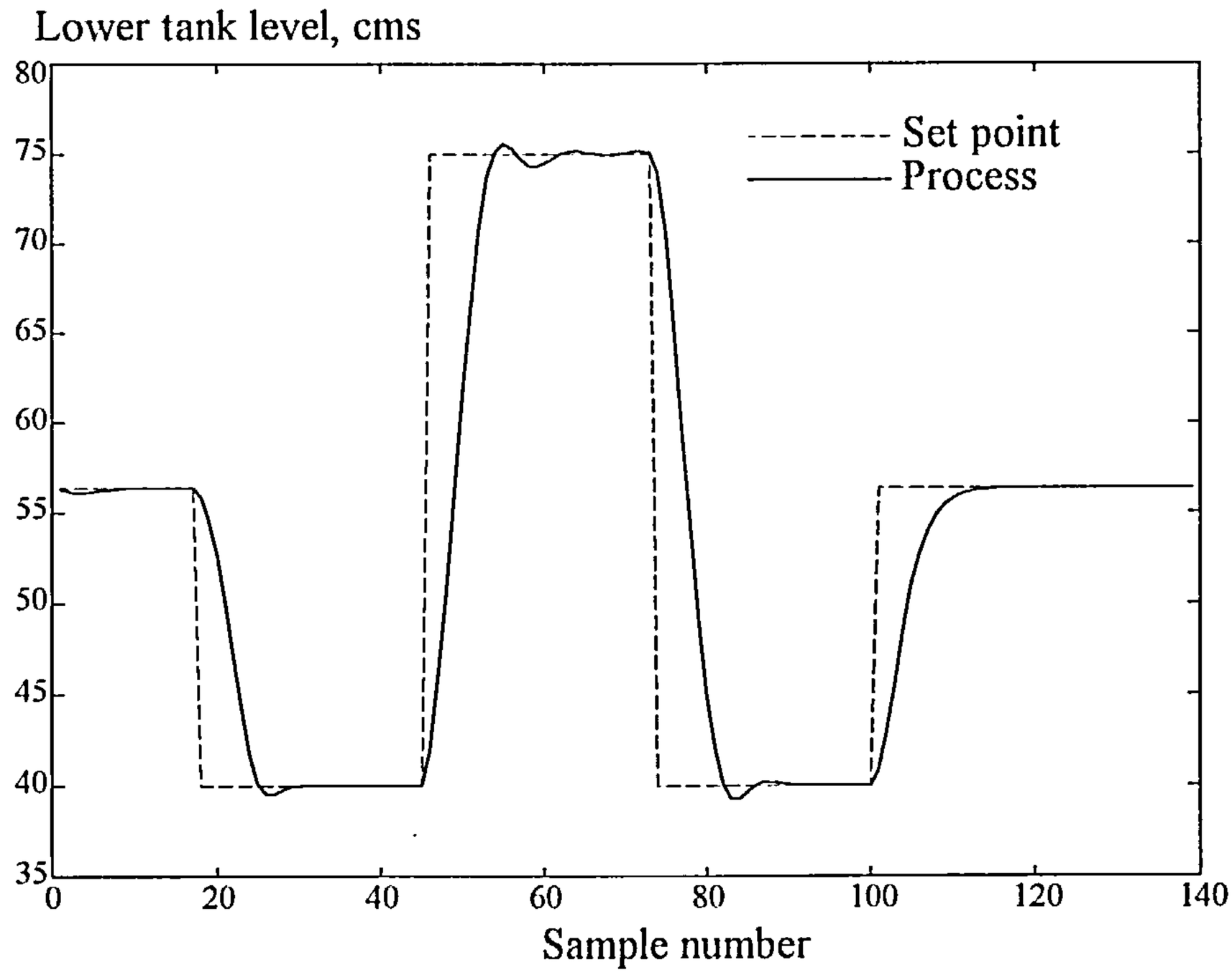


Figure 5.11 Process response and control input, λ=0 and N2=4



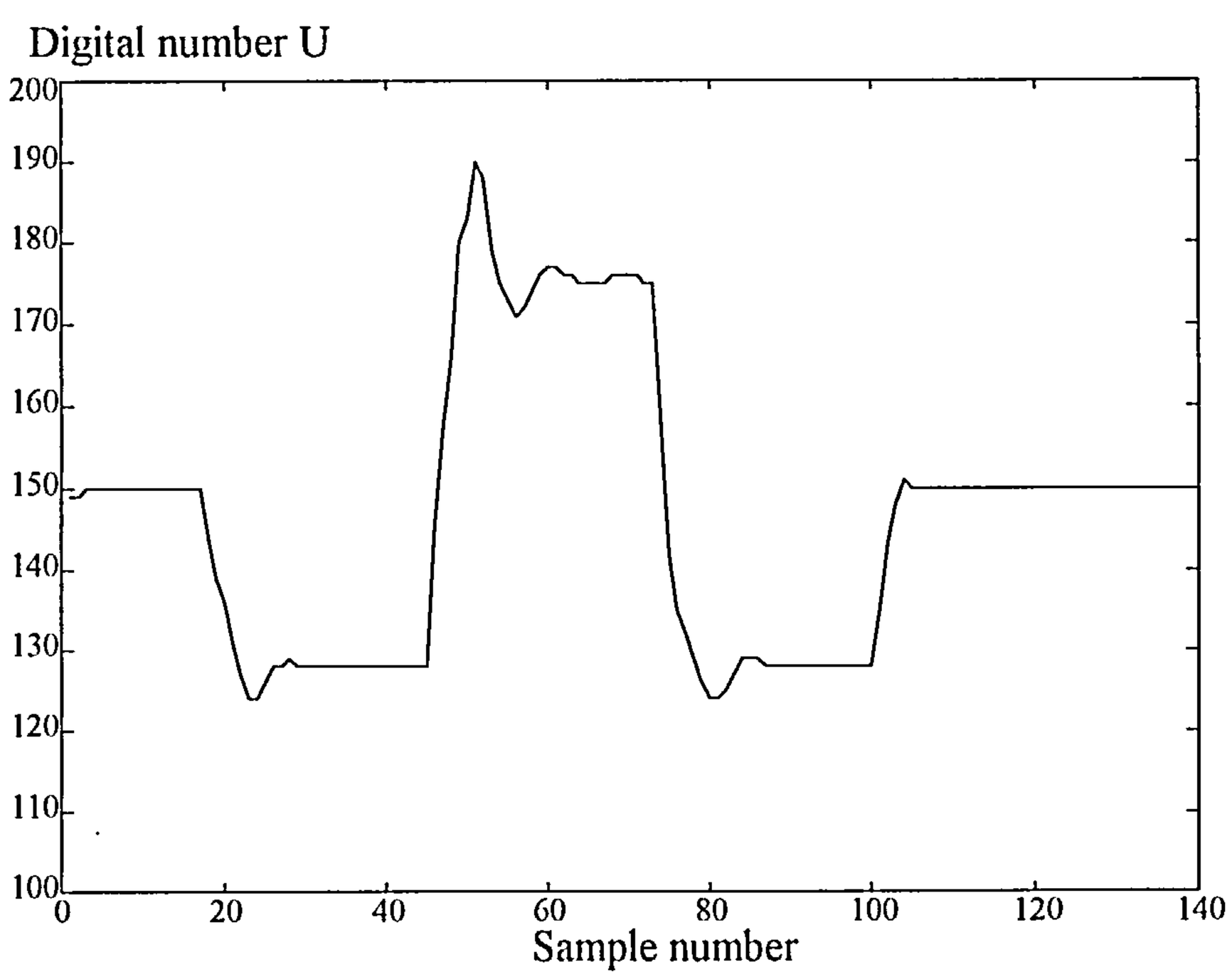
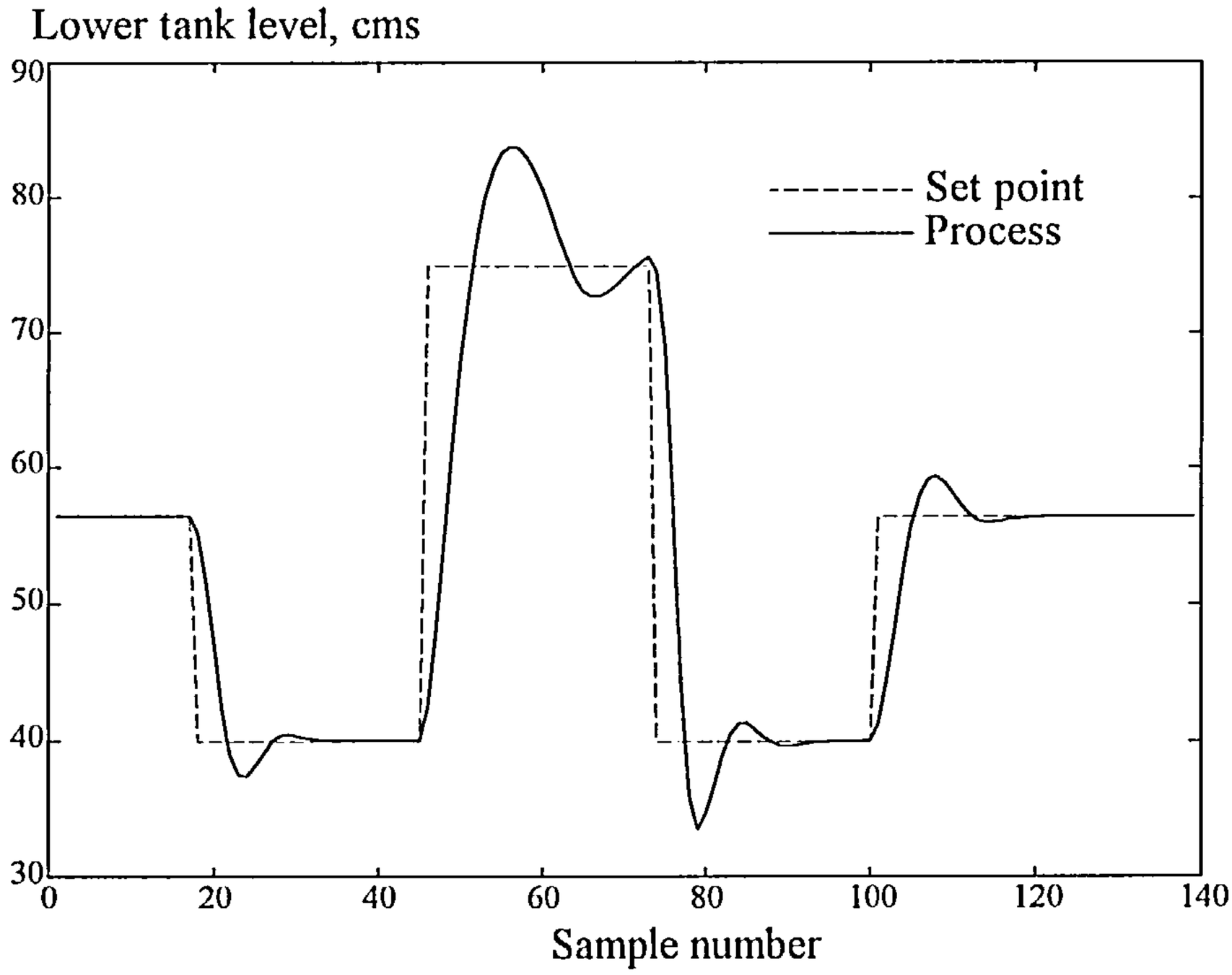


Figure 5.12 Process response and control input, λ=0 and N2=7



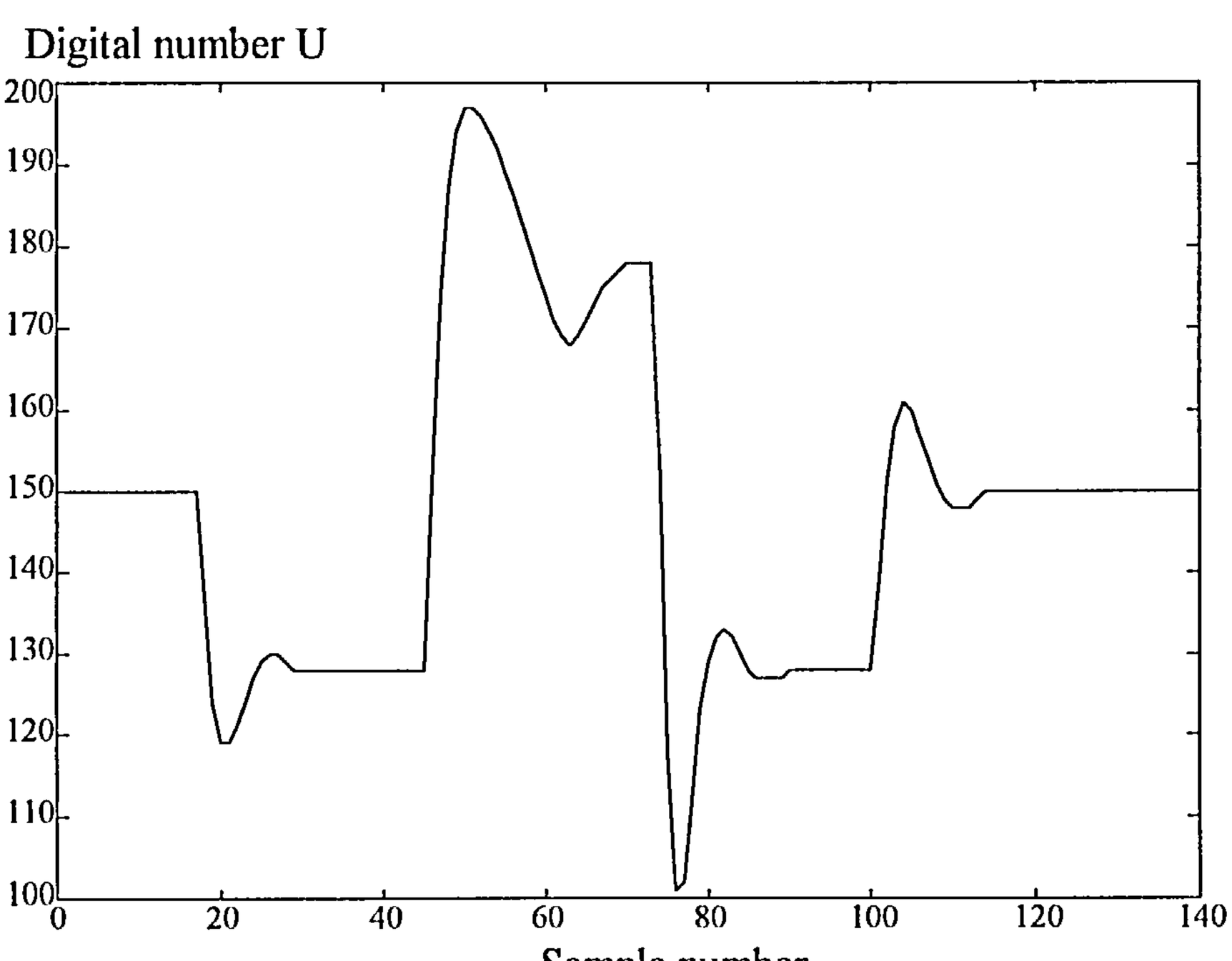
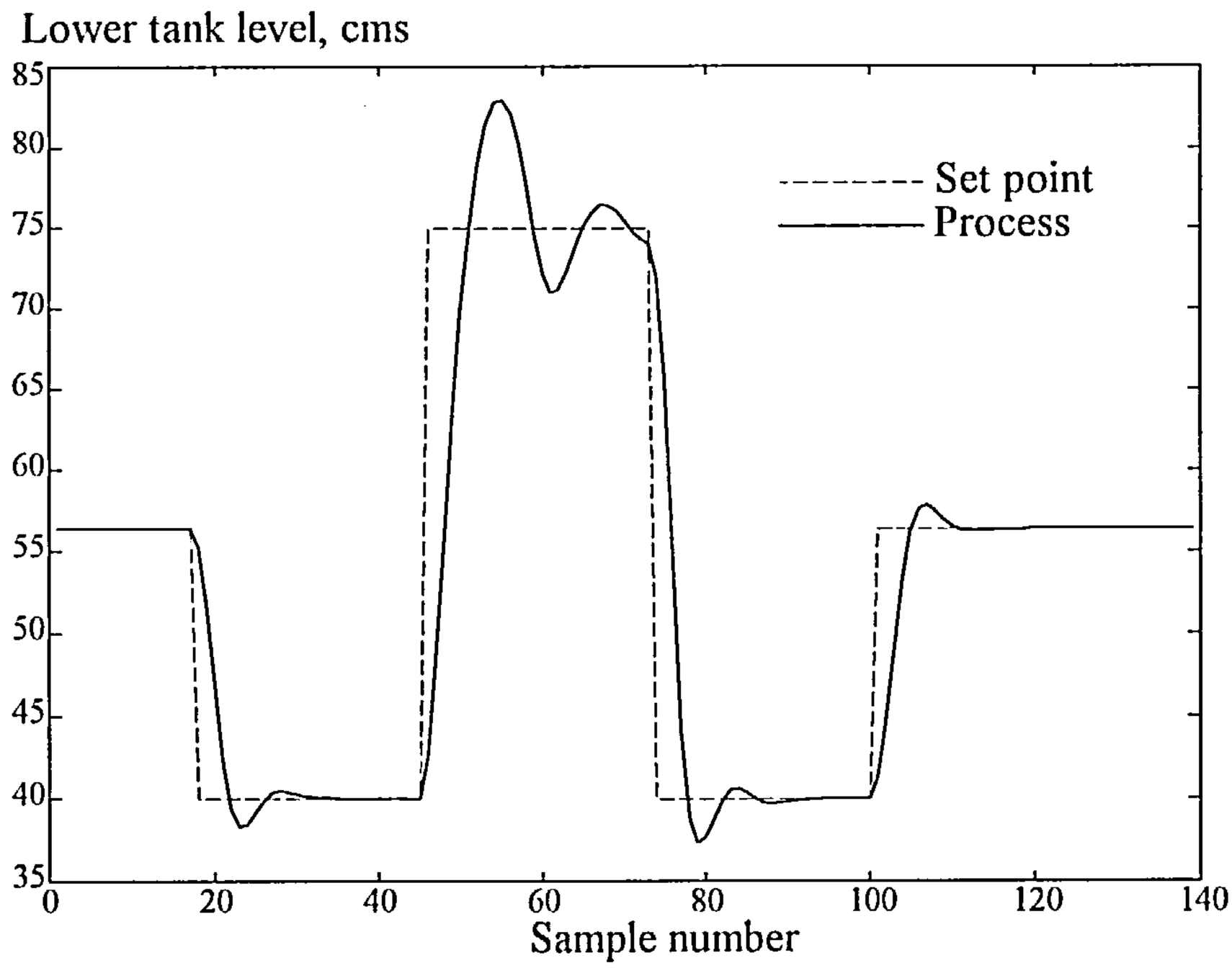


Figure 5.13 Process response and control input, λ=1 and N2=2



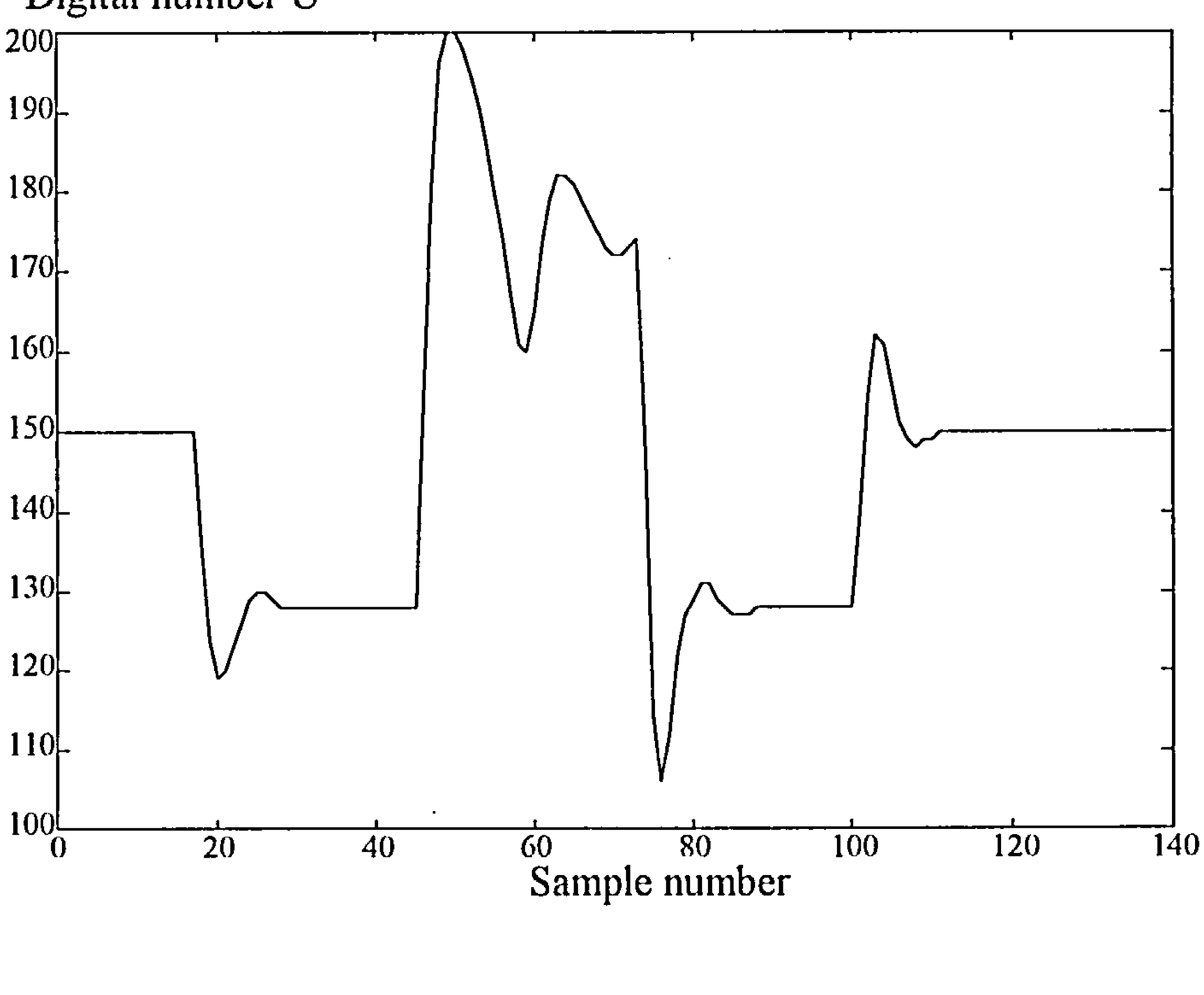
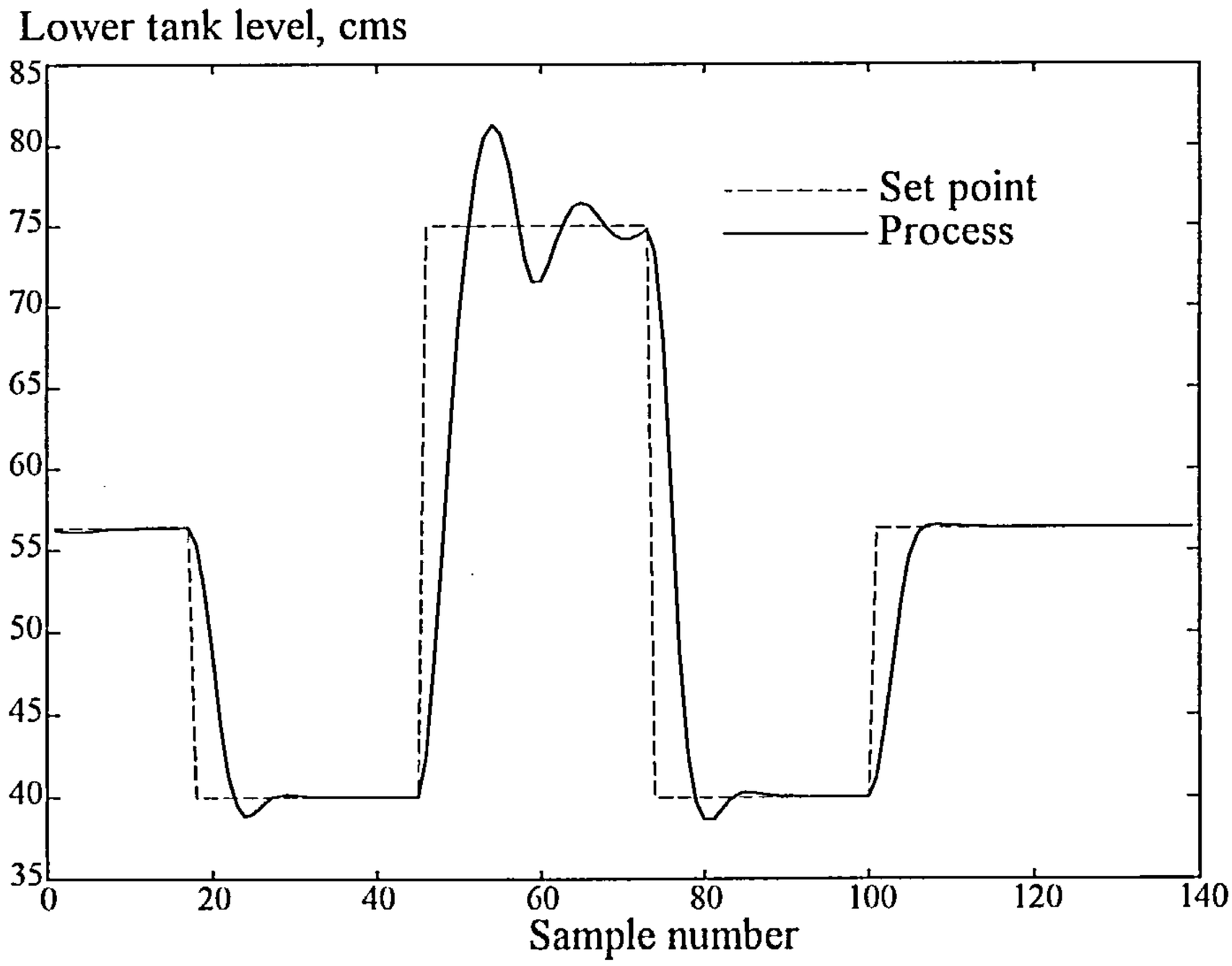


Figure 5.14 Process response and control input, λ=1 and N2=3



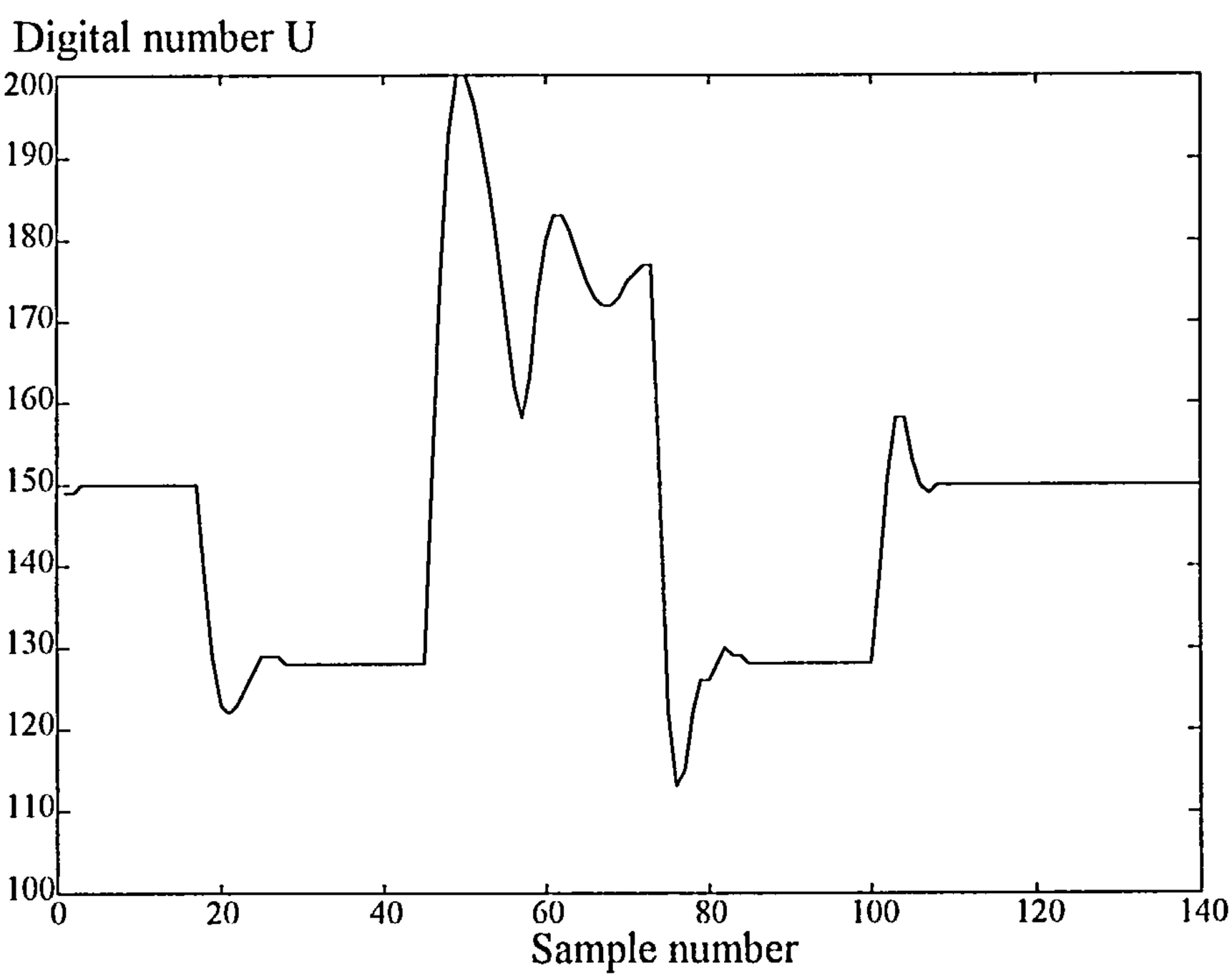


Figure 5.15 Process response and control input, λ=1 and N2=4

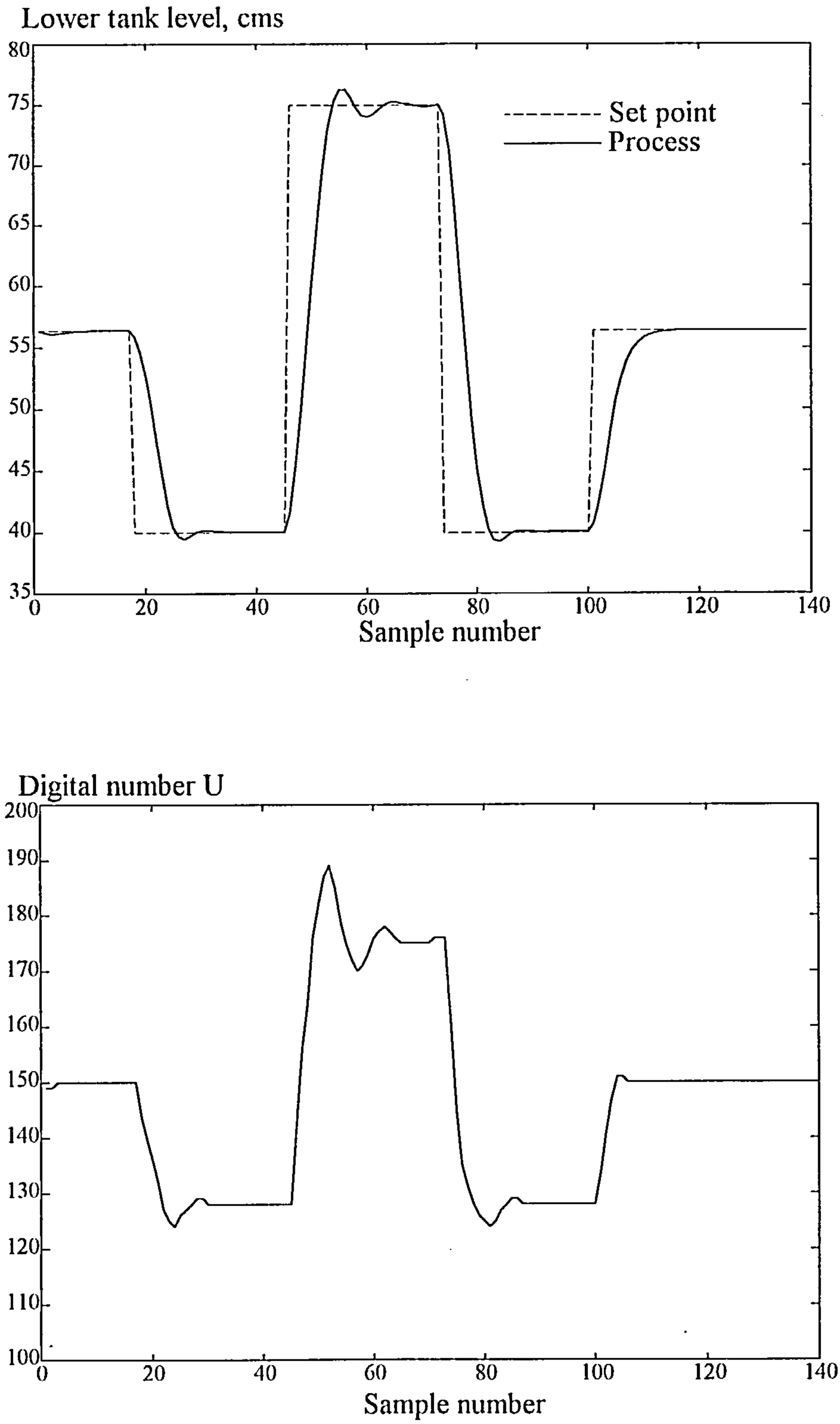
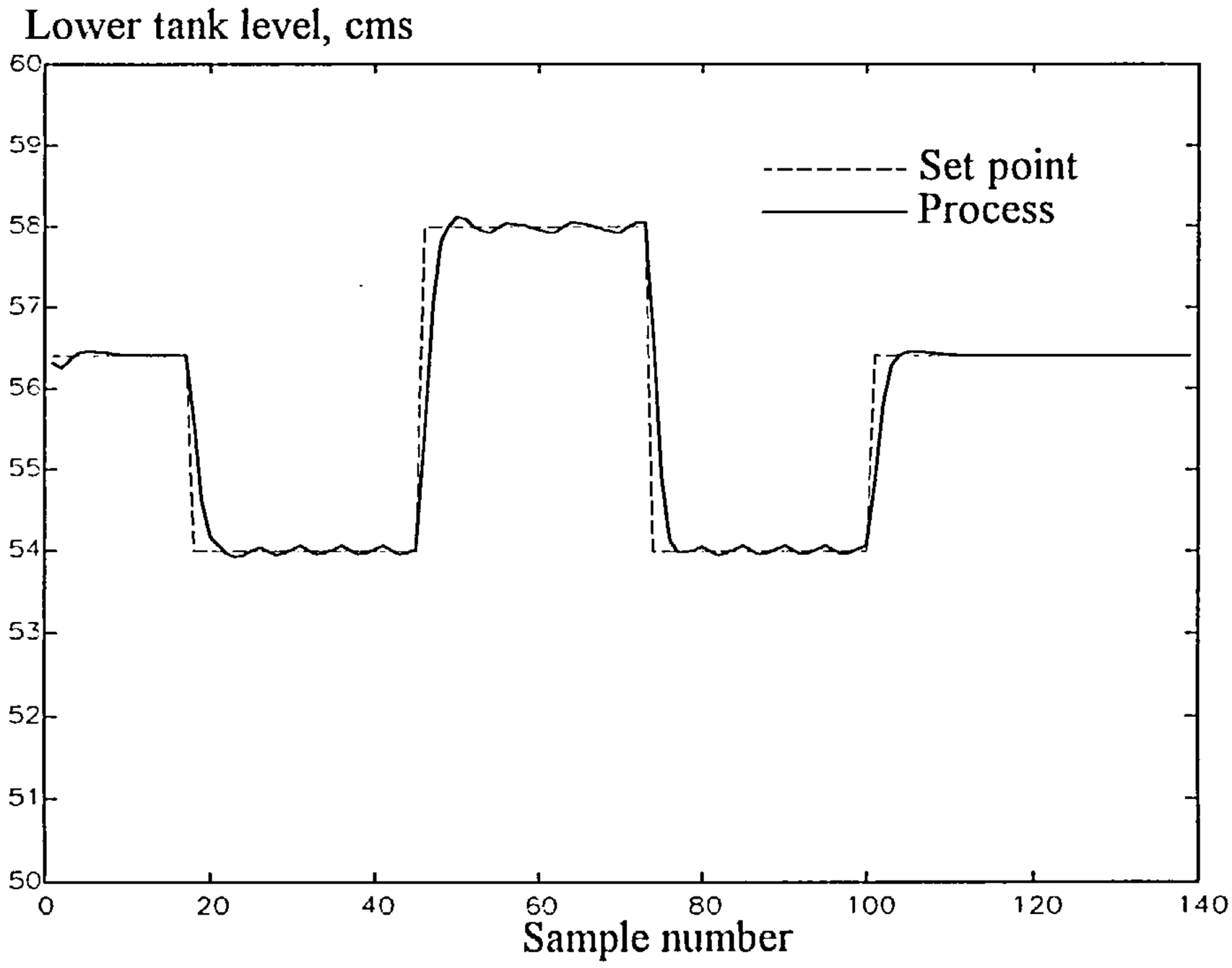


Figure 5.16 Process response and control input, λ=1 and N2=7

With λ set equal to zero a reduction in the control effort is seen as the prediction horizon is increased, however with λ set equal to one there is only a very small reduction in the control effort for prediction horizons 2 to 4 and with the prediction horizon set at 3 the control effort increases. It is also observed from Table 5.2 that, for prediction horizons 3, 4 and 7, the control effort is similar for both λ set at zero and one. This can be explained by examination of the cost function in equation 5.1. As N2 is increased the set point term in the cost function becomes larger than the control weighting term with Nu set equal to zero. Hence, the cost function is minimised mainly to achieve the required control. The reductions in the control effort with λ equal to zero occur as a result of equipping the scheme with long range prediction enabling anticipation of the process direction. However, the process response for λ set equal to zero is more desirable than when λ was set at one and hence, when using multi-step-ahead prediction the value to set λ at is recommended as zero.

The control scheme was then tested with small set point changes around the steady state operating point of 56 cms. The reason for this was so that a comparison with a well tuned PID controller described in the next section could be made. Figure 5.17 shows the result for N2=2 and λ equal to zero, it is seen that very good set point tracking is obtained and that the control signal required (Figure 5.17) is also desirable.



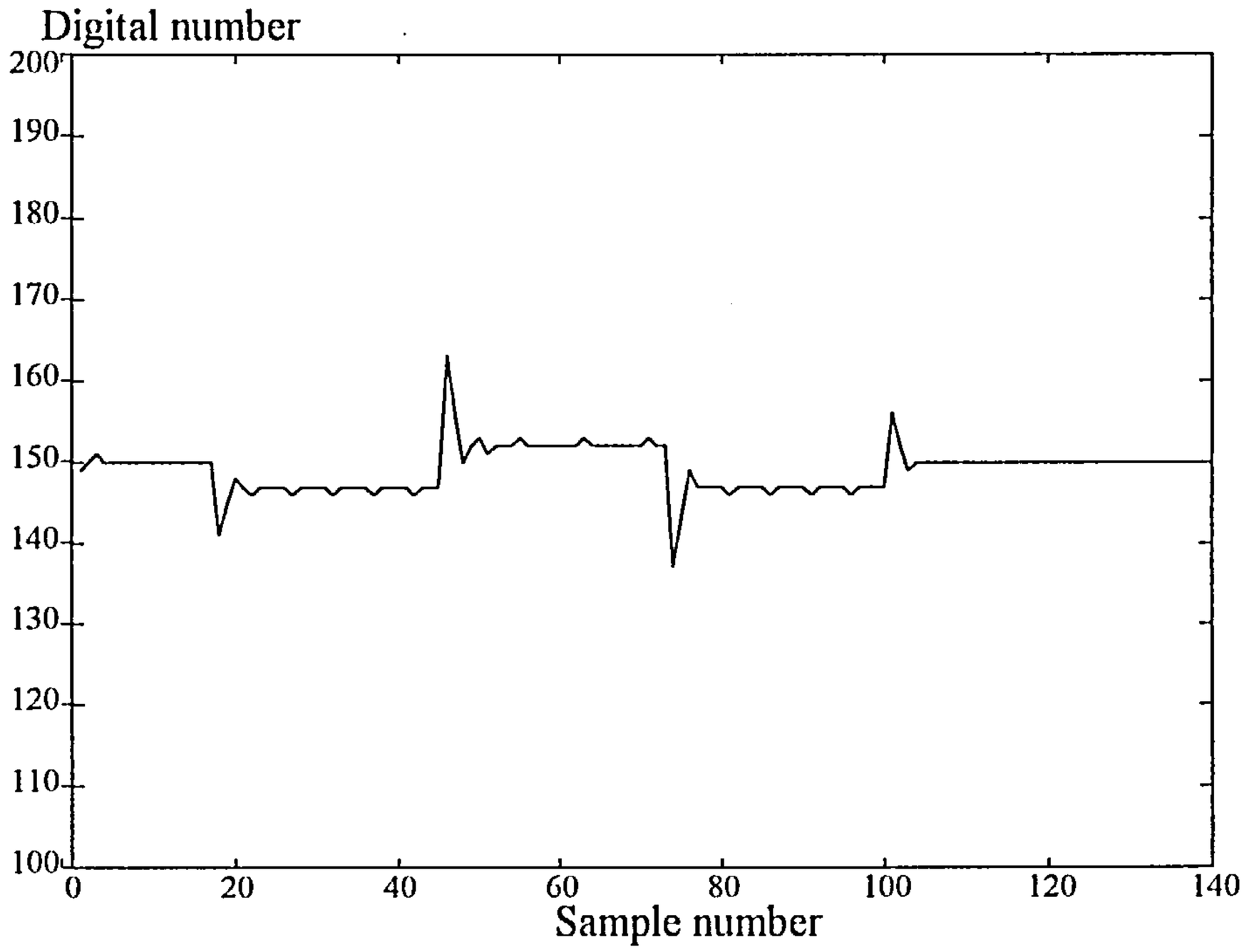


Figure 5.17 Process response and control input for small set point changes, λ=0 and N2=2

5.7 CONTROL USING A STANDARD PID CONTROLLER

A PID algorithm was implemented into a standard feedback control strategy and the ability of the control scheme to track the same set point changes as were used in the neural predictive control strategy was investigated.

PID control has been used in many industrial control systems with success for over the past fifty years and any new techniques for process control should be compared with standard proven methods in order to gauge their relative performance. Section 5.7.1 presents the PID algorithm used and the method used to obtain the parameters of the algorithm. Results using the standard controller are presented in section 5.7.2.

5.7.1 The PID algorithm

Equation 5.5 illustrates the PID algorithm used to calculate the control signal to the process valve:

$$u(n) = k_c\left[e(n) + \frac{T}{T_i}\sum e(n) + \frac{T_d}{T}(e(n) - e(n-1))\right] + u_s \qquad \ldots(5.5)$$

where T is the sampling time, e(n) is the input to the PID controller and represents the error between the required process set point and the process output (the height of liquid in tank 2, h_2), K_c is the proportional gain, T_i is the integral action time and T_d is the derivative time. u_s is the controller bias signal which, in this study, was 150 corresponding to a steady state operating point of 56 cms for the height of

liquid in tank 2. u(n) is the output of the controller that operates on the process valve and can range between 100 and 200, as described previously in section 4.2.1.

When using the PID algorithm the values of the parameters K_c, T_i and T_d have to be chosen. Many techniques have been proposed for calculating these parameters. One method is to use simple process performance criteria such as the one-quarter decay ratio, minimum settling time, etc. Another method that has been proposed is to use time integral performance criteria such as integral of absolute error (IAE) or integral of the square error (ISE). Both of these methods are adequate but result in multiple solutions or can be time consuming. The tuning method adopted in this study was to use empirical tuning methods such as those proposed by Ziegler and Nichols (1942) and Cohen and Coon (1953) since these methods give adequate solutions and are easier to perform.

The empirical method chosen was the Cohen and Coon reaction curve method. This method involves breaking the feedback loop between the process output and the controller input and applying a step change to the input of the open loop process. From the resulting process response to the step input an approximate first order process model can be estimated of the form

$$G(s) = \frac{k_p e^{-\tau_d s}}{\tau s + 1} \qquad \ldots(5.6)$$

where k_p represents the static gain, τ_d the process dead time and τ the process time constant. The size of the step change around the steady state operating point is important and it should not be so large that the process is driven into non-linear regions which would then invalidate the linear model fit.

Using the values of k_p, τ_d and τ calculated for equation 5.6 from the observed process step response the parameters for the PID algorithm can then be calculated using expressions derived by Cohen and Coon (1953). The parameters calculated are not optimal but give a good starting point after which, fine tuning can be performed manually on the process if required. Initially a proportional controller was used and hence only the value of Kp was determined, however this resulted in unreasonable offset between the process output and set point, as expected, hence for a fairer comparison to be achieved a PI controller was considered. The PI controller removed the process offset as was expected, but the overshoot was now considered unacceptable. Finally a PID controller was investigated and a reasonable process response was obtained when the parameters in the algorithm were fine tuned manually. The resulting values for the PID controller were K_p=5, T_i=30 seconds and T_d =80 seconds.

5.7.2 PID controller results

The well tuned controller was then used to control the process when subjected to a set of small step changes around the operating point. The set point changes were the same as those presented to the neural predictive control strategy shown in figure 5.17. The response of the process when controlled using the finely tuned PID controller is illustrated in figure 5.18 along with the corresponding control signal. For the small set point changes the PID controller compared favourably with the neural predictive control strategy in terms of the process response, however it did require a larger control effort as illustrated by the brief saturation of the input signal in Figure 5.18.

When subjected to the large set point demands, which required process operation over a wide non-linear region the PID control performance deteriorated, (Figure

5.19), and the neural predictive control strategy provided a more accurate control together with a much lower control effort (Figure 5.17).

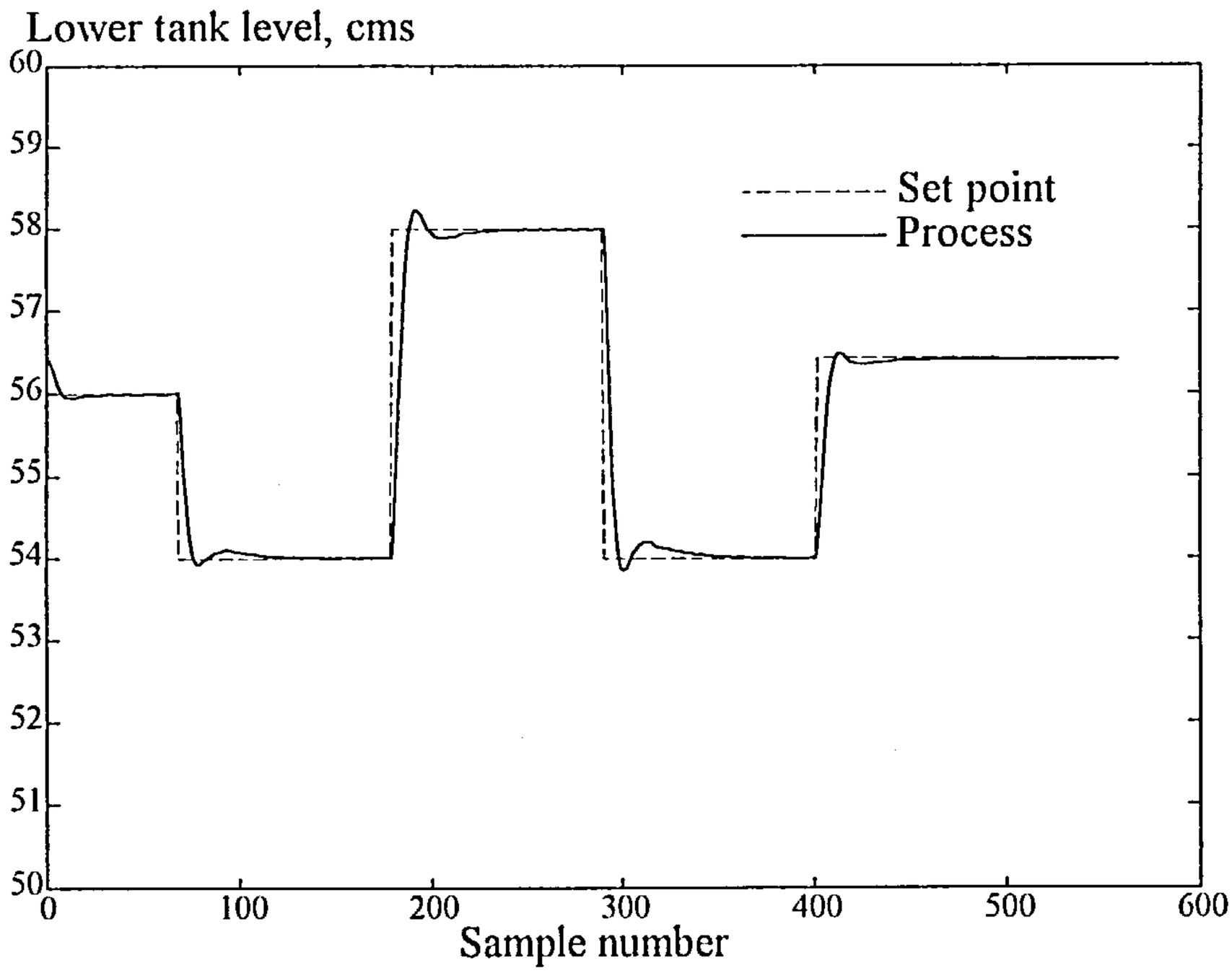
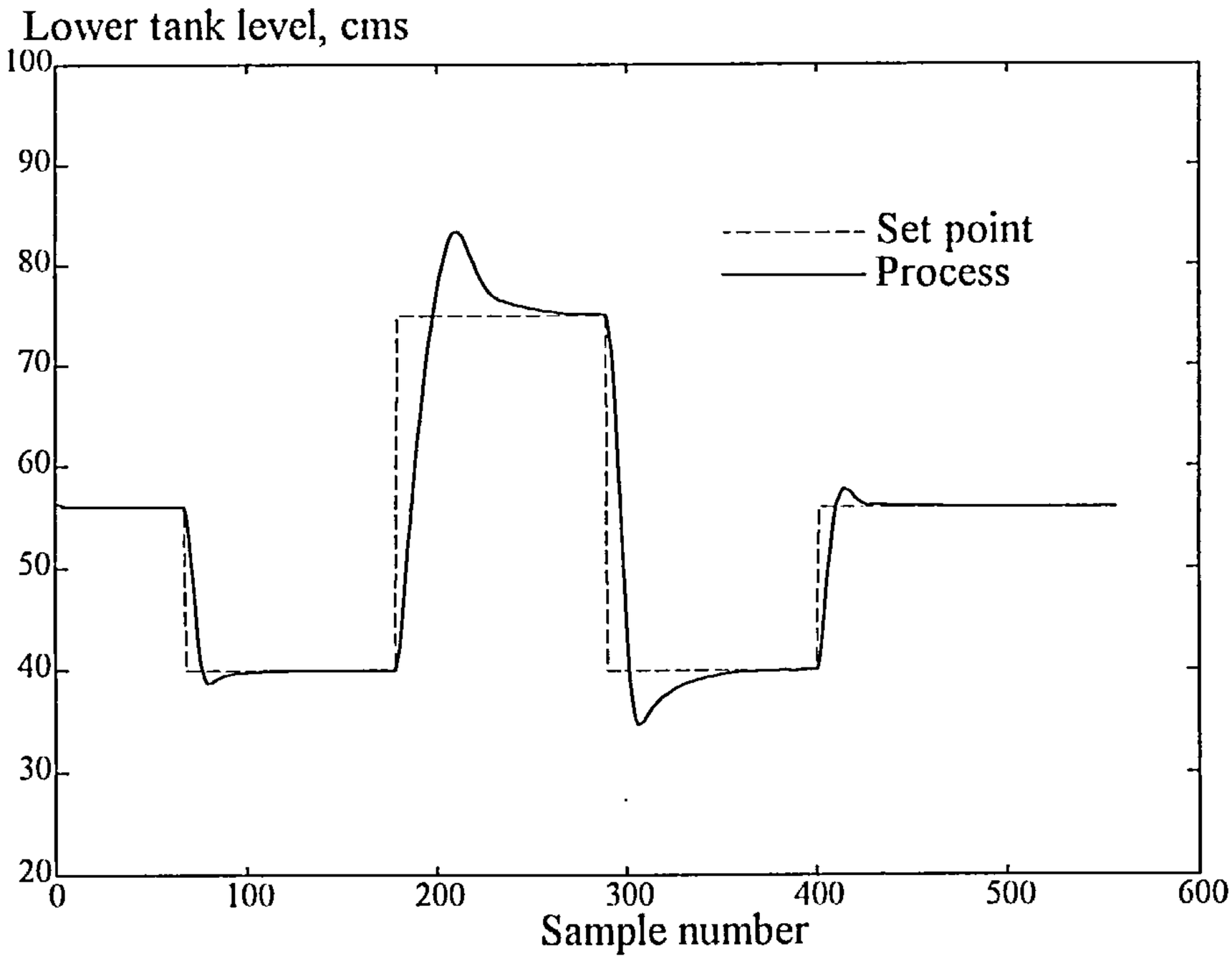


Figure 5.18 Process response and control input for small set point changes under PID control



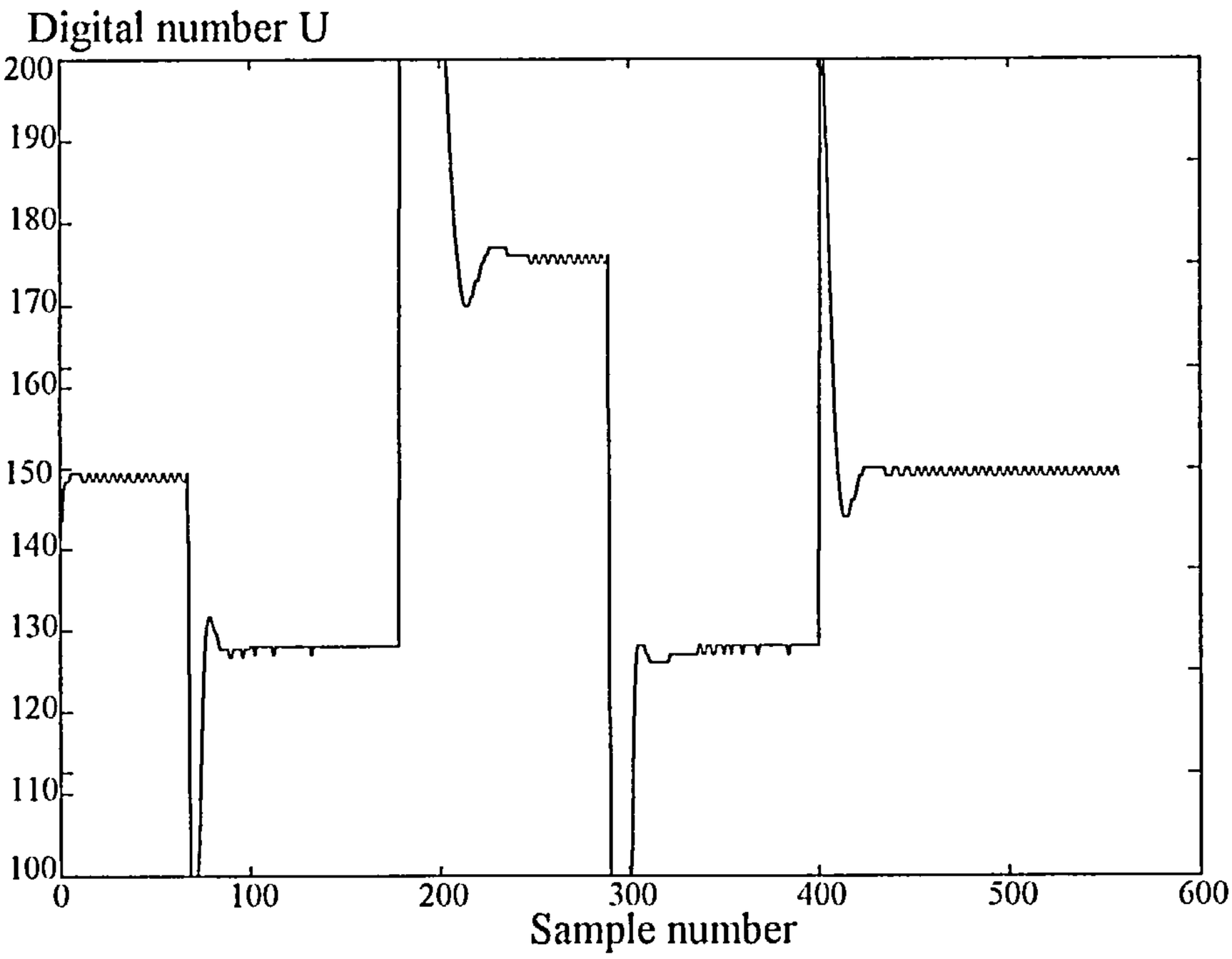


Figure 5.19 Process response and control input for large set point changes under PID control

5.8 SUMMARY

A number of control strategies that contain neural networks as either process models, controllers or both have been described and advantages and disadvantages of each have been presented. One of the major factors, in several of the control schemes, is that an inverse model of the process is required. This led to investigations into the neural predictive control strategy since an inverse process model is not a requirement of the scheme and also multi-step-ahead predictions can be performed which can result in an improved control performance.

The neural predictive control strategy is based around minimising a suitable cost function to produce an optimum process input that causes the response of the process to follow a predetermined set point. Simulations of both one-step-ahead and multi-step-ahead predictions were performed and the effect of different weighting factors, λ, on changes in the control input signal were investigated. When using one-step-ahead prediction, the closer λ became to unity the smoother the control signal became but this was at the expense of a sluggish process response. Using multi-step-ahead predictive control overcame this disadvantage with λ set at either zero or one, but as the prediction horizon increased it was found that λ set equal to zero gave the best results.

A standard PID controller was implemented to control the liquid level process and a comparison was made with the neural predictive control scheme. The PID parameters were initially determined using well established empirical methods and were then finely tuned manually. It was observed that the PID controller compared in performance with the neural predictive control scheme for small excursions about the operating point. However, as the set point demands became larger, the

performance of the PID controller deteriorated and was significantly worse than the neural predictive control both in terms of set point tracking and control effort.

CHAPTER 6

ON-LINE ANN CONTROL

6.1 INTRODUCTION

Chapter five demonstrated through simulation studies the feasibility of using a neural network process model in a predictive control strategy to facilitate control of the dual tank liquid level process. Simulation plays a very important part in the development and investigation of process models and control strategies and can give a valuable insight into potential problem areas. However, simulation studies are usually performed with an assumed or calculated mathematical model of the process and simulated under ideal conditions. In many cases this simulation environment does not accurately represent the behaviour of the real process in practice. This chapter investigates the on-line application of the neural network control scheme to the control of the dual tank liquid level process, available in the laboratory, under what can be assumed as practical conditions.

As in the previous chapter, the performance of the neural network control scheme is compared with that of a standard conventional control algorithm. Also in this chapter, an investigation into the stability of the predictive control scheme is presented using techniques taken from linear system identification.

6.2 CONTROL SCHEME IMPLEMENTATION

To investigate the on-line performance of the neural predictive control strategy and to make comparisons with a standard conventional controller, a computer program was written that contained the required control algorithms and the neural network

model of the liquid level process. The program was written using the Microsoft C language which has many built in functions that can be used to set accurate sample times and produce graphical output on the computer VDU if required. As with the simulation studies, the initial digital output number from the computer was 150 and this corresponded to a steady state height of liquid in tank 2 of 59 cms.

The values used for the PID control algorithm were taken from those used in the simulation studies, but initial investigations resulted in the derivative action being removed completely and the integral action Ti being increased from 30 to 38 seconds in order to obtain accurate set point following over small excursions around the process steady state.

6.3 TESTS CARRIED OUT ON THE PROCESS

Three different tests were carried out on the liquid level process in order to investigate the performance of the neural predictive control strategy.

The first test involved applying a set of step input signals as the required set point and monitoring the ability of the process to track the set point when different constraints were imposed on the cost function, equation 5.1.

The second test involved applying a step change to the set point (required height of liquid in tank 2) and monitoring the process rise time, maximum overshoot, integral square error (ISE) between the required set point and the measured process output, and the control effort (equation 5.4) for four different conditions as listed below

Condition A: $N2=1$, $\lambda(t)=0.0\ \forall\ t$

Condition B: N2=1, $\lambda(t)=1.0\ \forall\ t$

Condition C: N2=2, $\lambda(t)=0.0\ \forall\ t$

Condition D: N2=2, $\lambda(t)=1.0\ \forall\ t$

The final test investigated the process disturbance rejection property when the process was operating under regulatory control and subjected to large disturbances in the height of liquid in tank 2.

6.4 NEURAL PREDICTIVE CONTROL RESULTS

Figure 6.1 shows the response of the predictive controller when operated on-line and subjected to a set of large step input signals as the required set point with λ set equal to zero and the prediction horizon, N2, equal to one. It can be seen that accurate set point tracking is achieved with the corresponding control signal to the process valve shown in Figure 6.1. The on-line neural predictive control scheme was then implemented with the prediction horizon set at three whilst the weighting function λ was maintained at zero. Figure 6.2 illustrates the time response of the liquid level process and the control signal under these conditions. Again it is observed that overall good control performance is obtained. The corresponding control signal to the process valve illustrates an improvement over the control signal for one-step-ahead prediction with λ set at zero, Figure 6.1.

The control strategy was then tested using the same prediction horizons as above, namely one and three, but with the weighting factor λ set at one, hence applying full weighting constraint on the process valve movement. Figure 6.3 shows the response of the process to the set point changes and the control signal with the prediction horizon set at one. The control signal to the process is clearly much

smoother when compared to tests with λ set equal to zero, (Figure 6.1), whilst the ability of the process to track the set point changes is maintained. The set point and process response along with the control signal input for the prediction horizon set at three is illustrated in figure 6.4 where an acceptable control signal and process response is again observable.

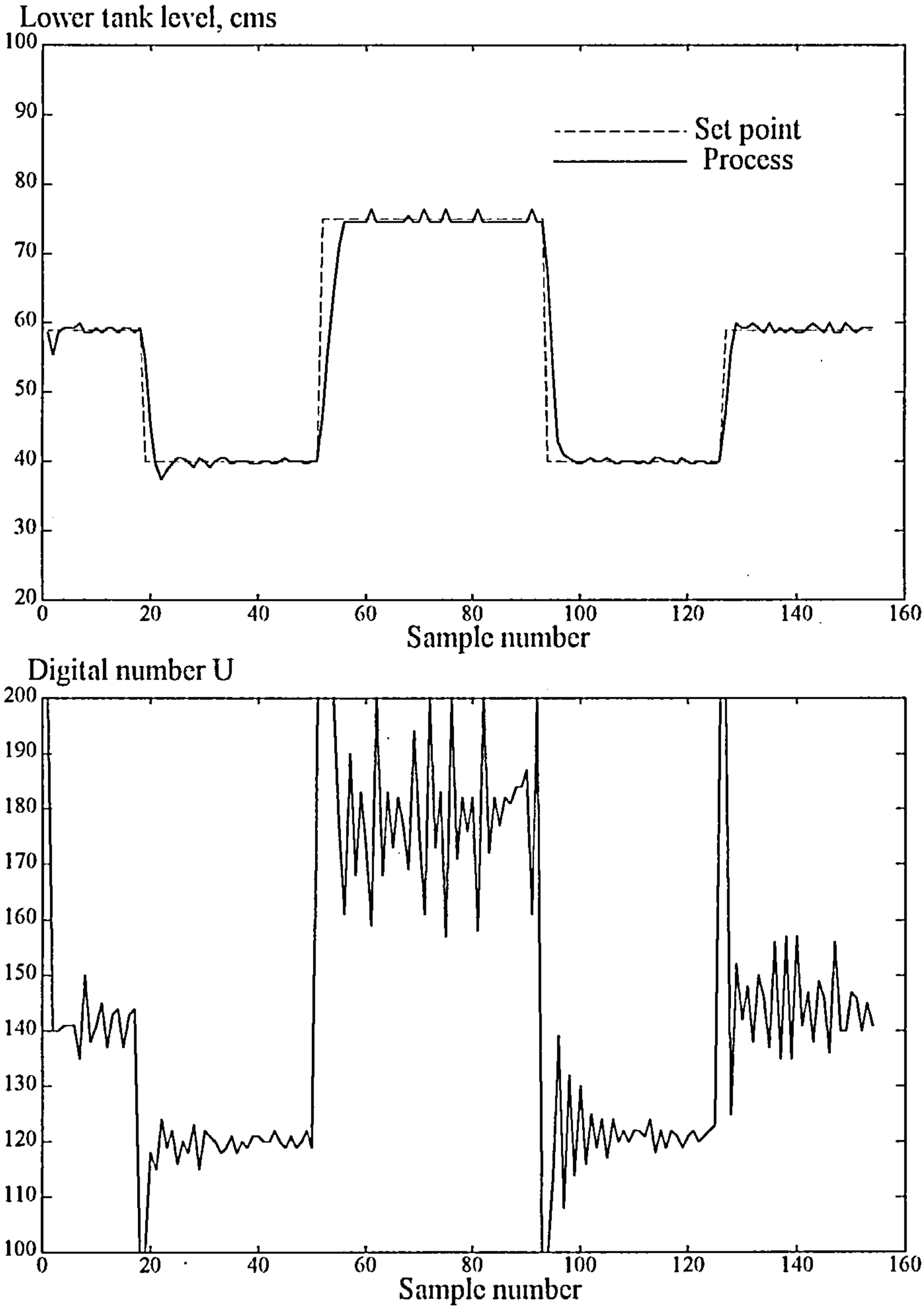


Figure 6.1 Process response and control input, λ=0 and N2=1

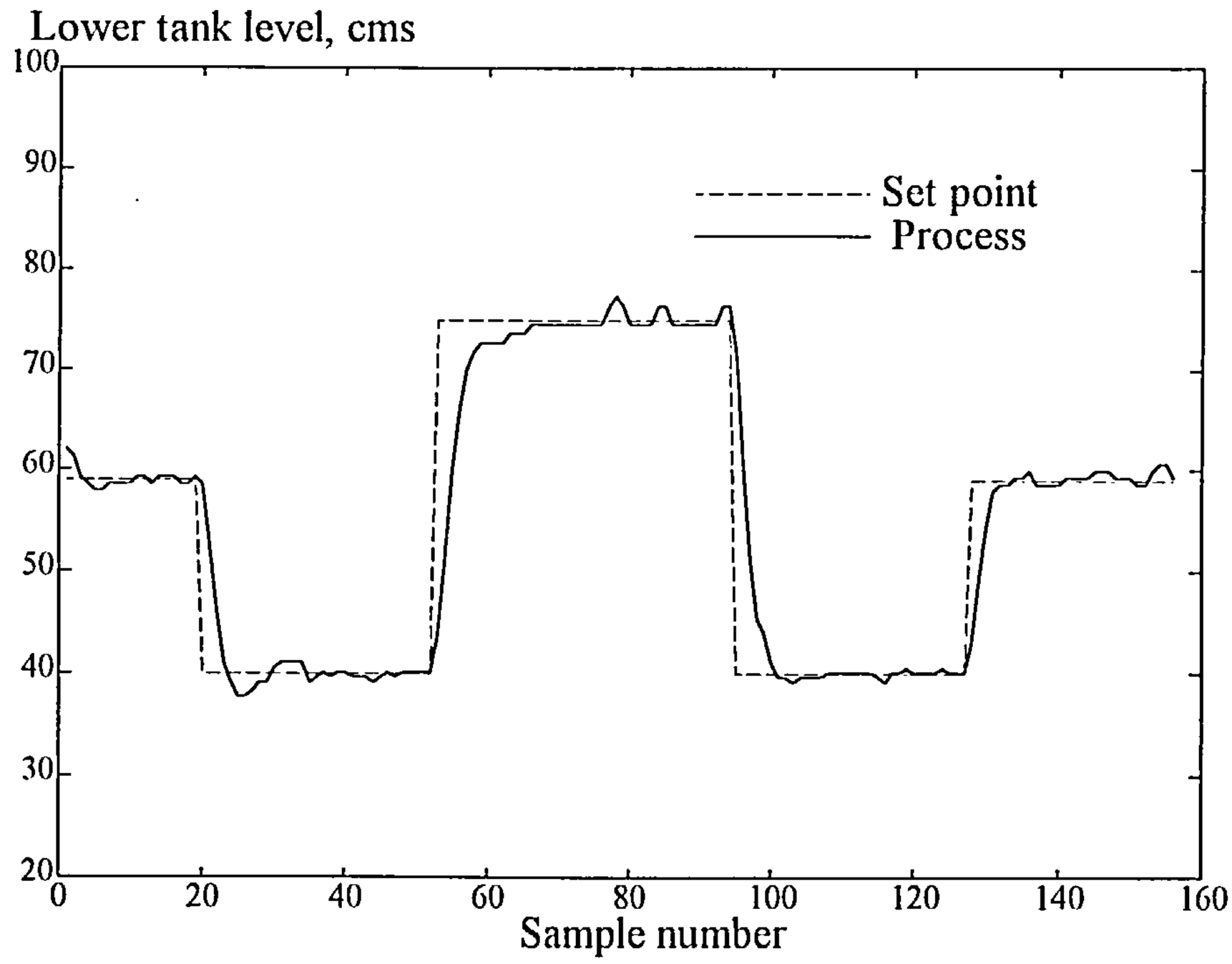
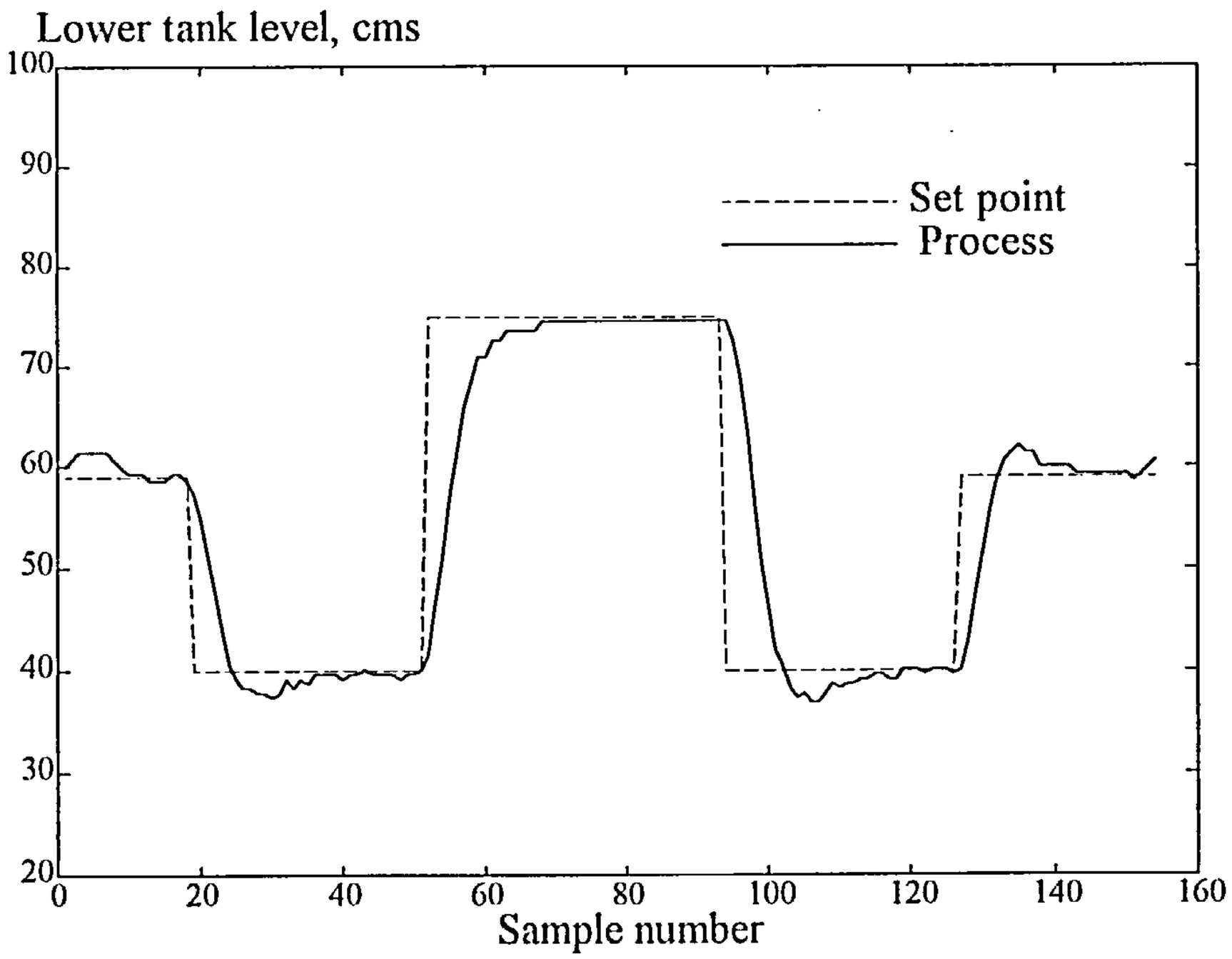


Figure 6.2 Process response and control input, λ=0 and N2=3



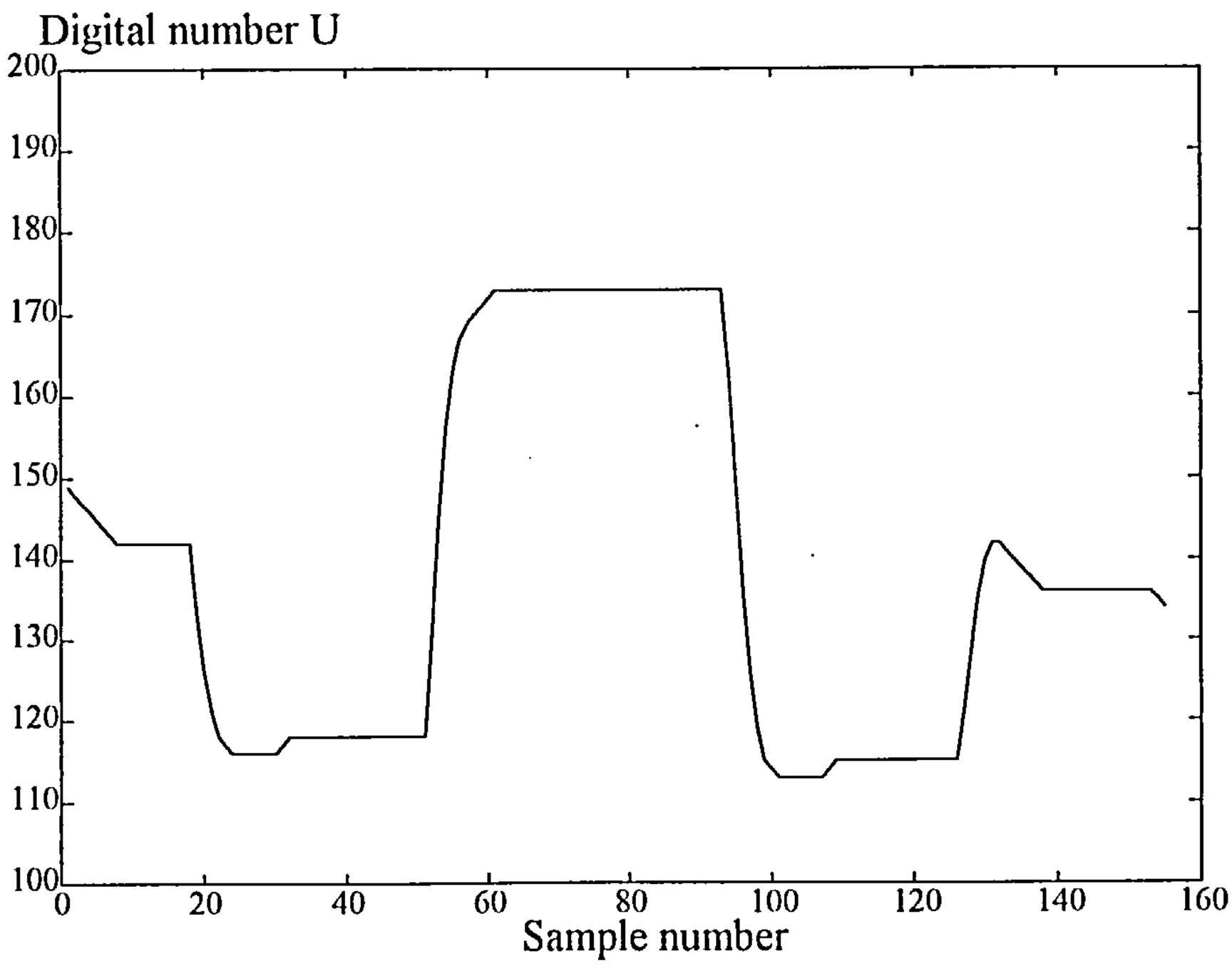
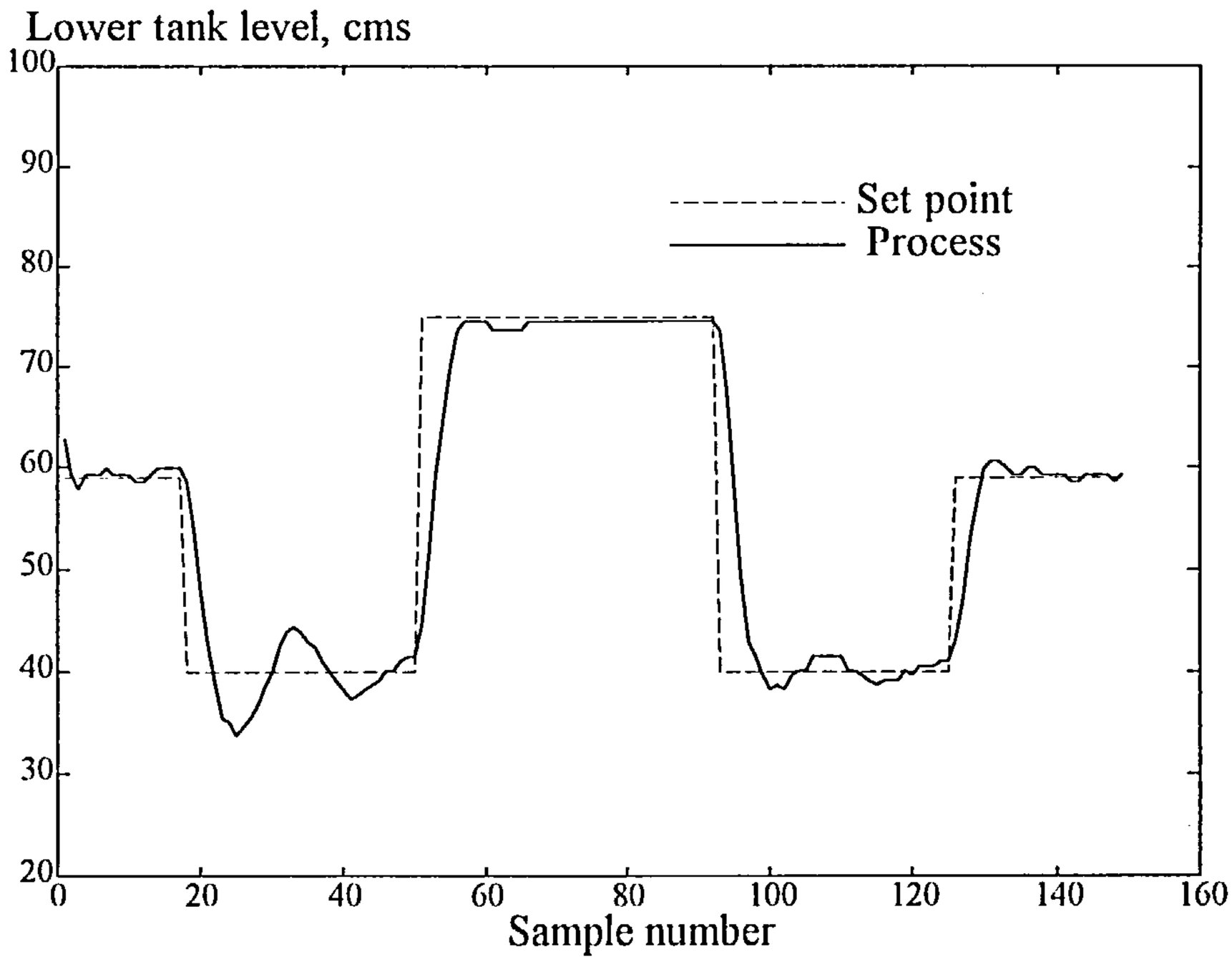


Figure 6.3 Process response and control input, λ=1 and N2=1



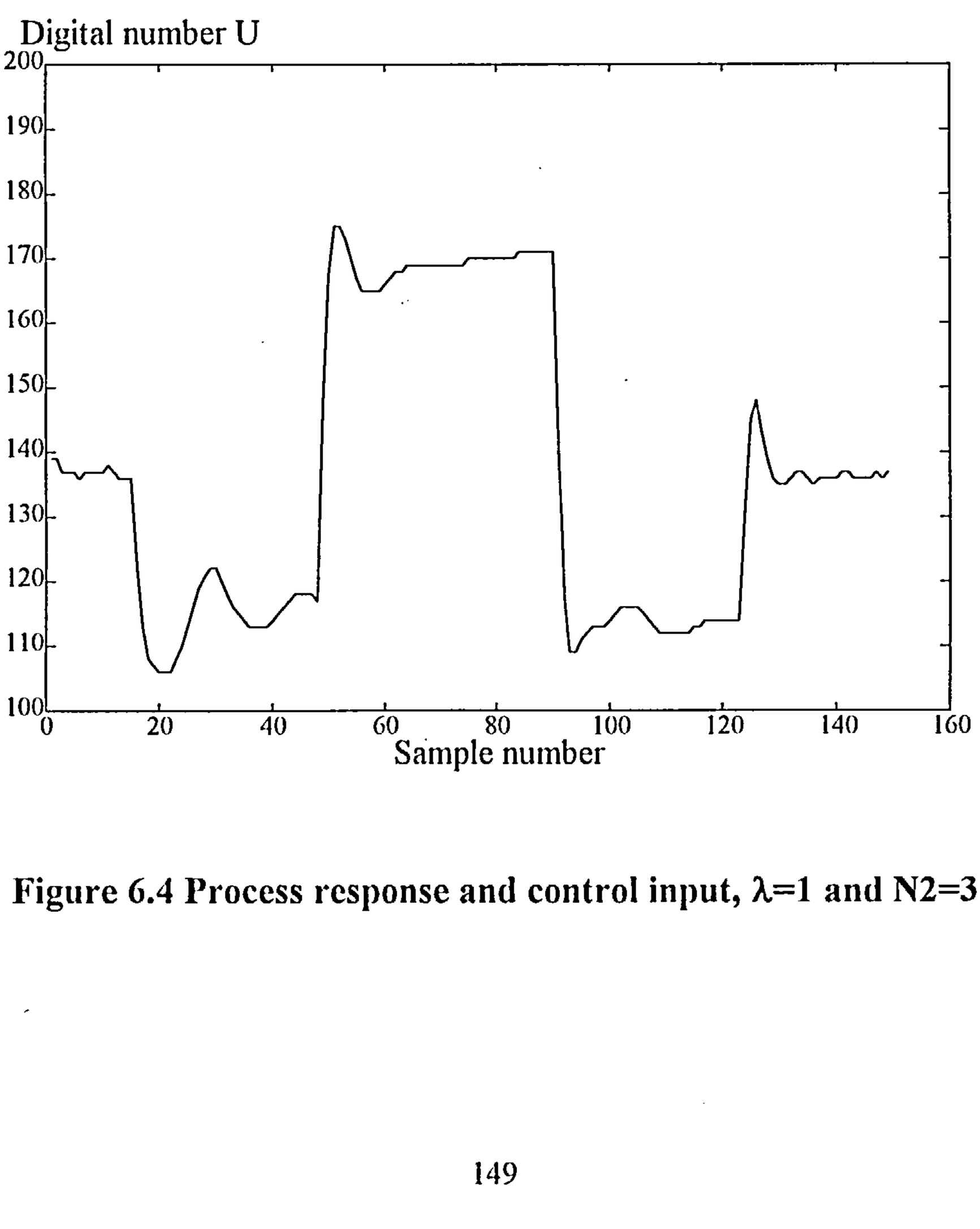


Figure 6.4 Process response and control input, λ=1 and N2=3

When the second test was conducted, in order to investigate the process characteristics, such as the rise time, under neural predictive control the results that were obtained for each of the conditions A, B, C and D are shown in Table 6.1 together with the performance of a conventional PI controller subjected to the same set point changes.

	Condition A	Condition B	Condition C	Condition D	PI control
Rise Time (min)	2.4	7.19	2.4	3.6	2.4
Maximum Overshoot (cms)	1.74	0.0	4.23	5.96	4.14
ISE ($\times 10^3$)	1.466	3.446	1.347	1.816	3.709
Control Effort	842	55	335	92	420

Table 6.1: Test results for the neural predictive controller and PI controller

The test results in Table 6.1 illustrate that with one-step-ahead prediction and λ =0.0 (Condition A), good performance is obtained but this is at the expense of large movements in the control input. These large changes in the control signal are significantly attenuated with λ=1.0 (Condition B) , resulting in no overshoot. However, it can be seen in the table that the rise time is much greater and this results in a larger integral square error. Increasing the prediction horizon to 2 also has the effect of reducing the control effort, even with λ=0.0, at the main expense of a larger overshoot (Conditions C and D). The control movement is again significantly reduced with λ=1.0 (Condition D). The results for Condition C in particular show an improved performance over the PI controller, with a similar rise time and maximum overshoot but, much lower ISE and control movement.

In order to test the disturbance rejection property of the neural predictive control scheme the process was operated under regulatory control at a set point of 59 cms, and disturbances were simulated in the bottom tank. The disturbances involved adding more liquid to the bottom tank to simulate an increase in the liquid inflow rate and also decreasing the amount of liquid in the bottom tank simulating a decrease in the liquid inflow rate. The rejection properties were examined with the prediction horizon set at one and λ at zero. Figure 6.5 shows the required set point and the response of the process for both the disturbances mentioned. Both disturbances were of a significant magnitude and the predictive control strategy is seen to be able to cope very well, and brings the height of liquid in tank 2 back to the required set point in approximately five sample periods. The control signal required to perform this disturbance rejection is also illustrated in Figure 6.5.

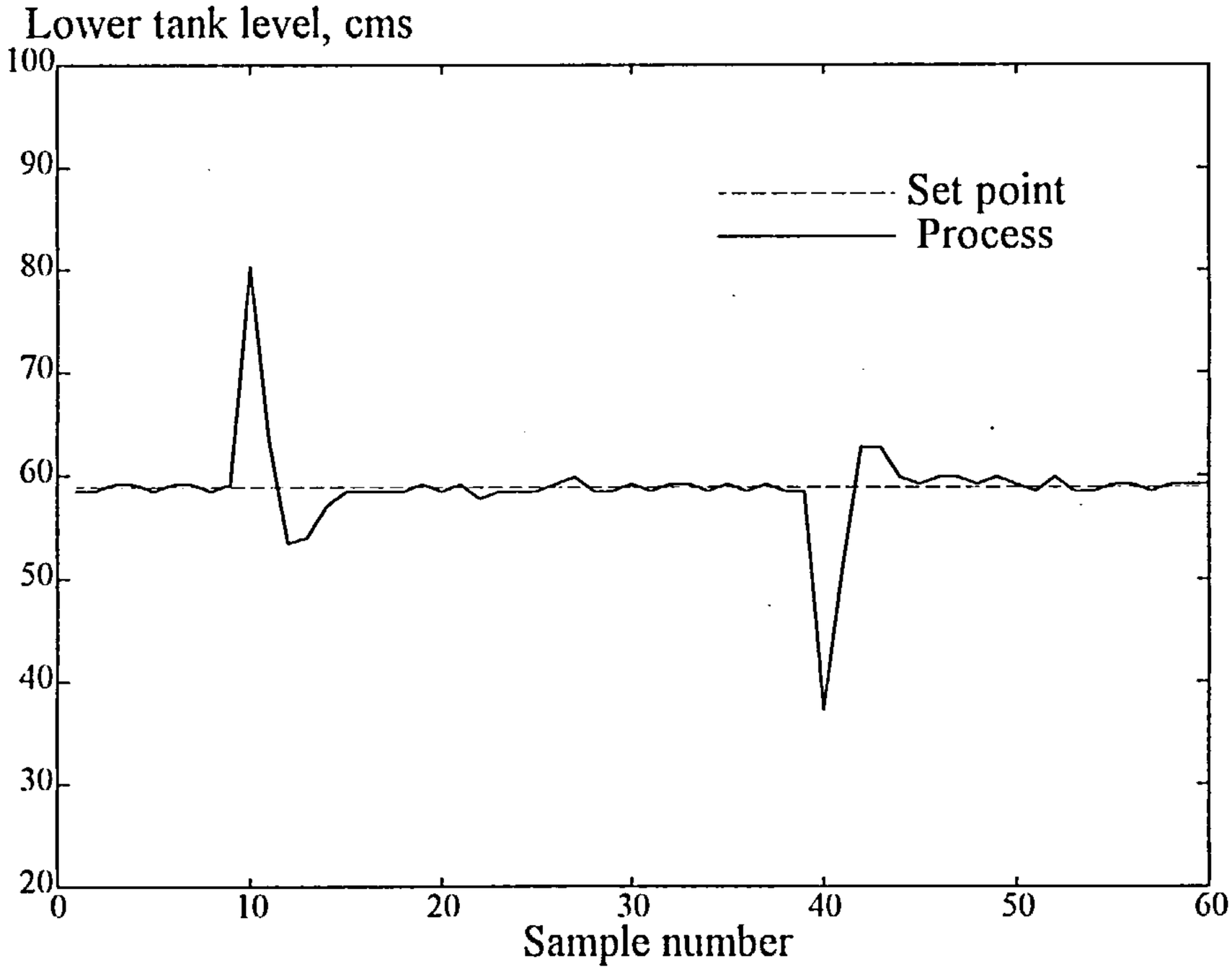


Figure 6.5 Process disturbance rejection, λ=0 and N2=1

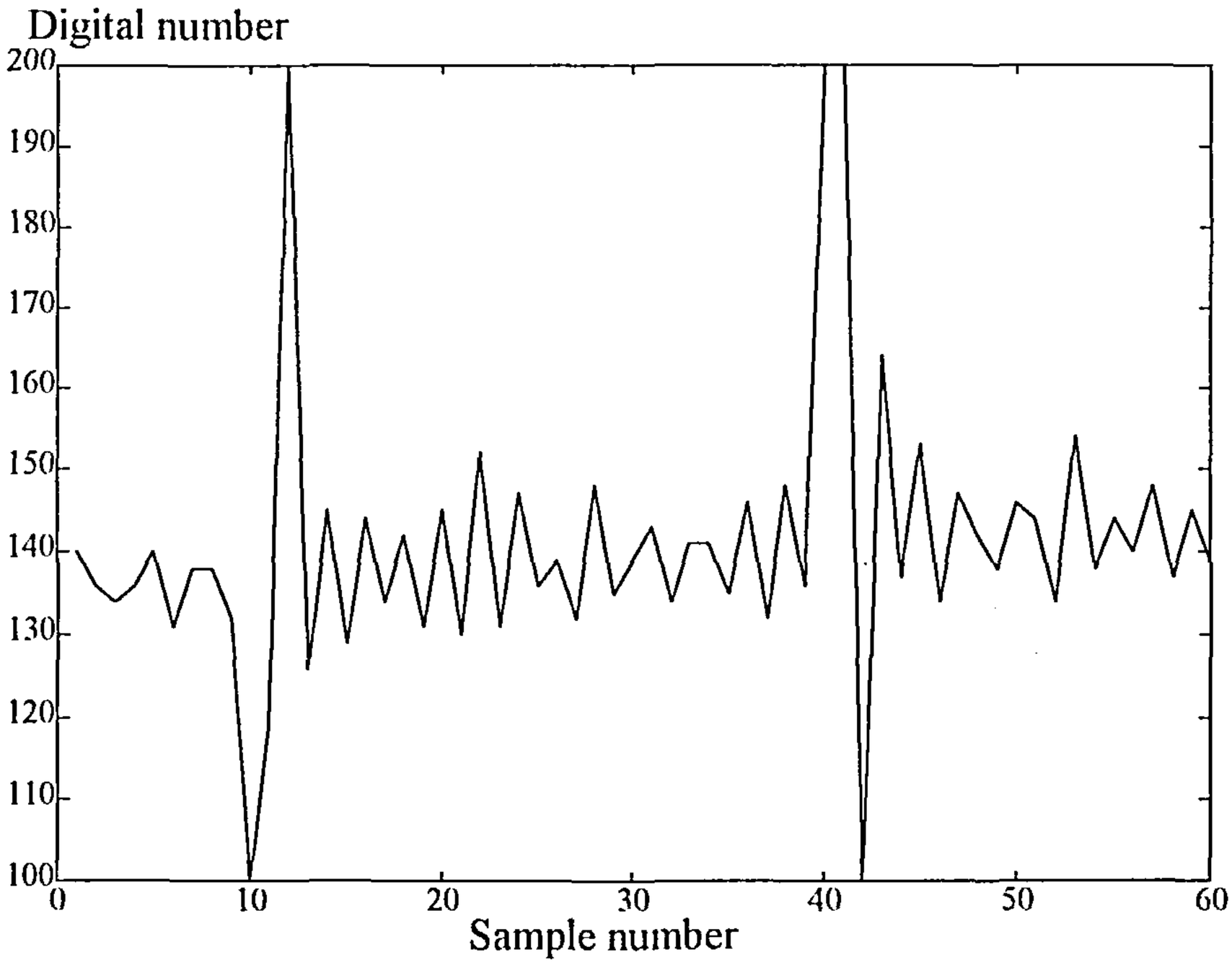


Figure 6.5 Control signal, λ=0 and N2=1

6.5 STABILITY ANALYSIS OF THE PREDICTIVE CONTROL SCHEME

An important consideration during and after the design of a process control system is that of the overall stability of the control structure. By stability, it is implied that the process signals remain bounded and converge so that the desired behaviour of the process is reached asymptotically. For the case of the liquid level process it is required that the height of liquid in the bottom tank follows the desired set point whilst the control input signal and height of liquid in the top tank remain within acceptable and realistic levels.

The condition for stability of closed loop controllers for linear systems is well documented and requires that the position of the roots of the closed loop characteristic polynomial lie in the left half plane of the imaginary axis in the Laplacian domain or within the unit circle in the discrete z-domain. Many methods

exist for determining the position of the poles for linear time-invariant systems such as the Routh-Hurwitz criterion and the Jury criterion. However, for non-linear and time varying closed loop systems, the stability analysis is not as straightforward.

The direct method of stability analysis proposed by Liapunov can be used as a tool in the analysis of overall system stability for non-linear and time-varying systems. Two methods for stability analysis were proposed by Liapunov, indirect and direct, of which the direct method is the most useful since it does not require the solution of the differential or difference equations that describe the overall system and it can also be applied to non-linear time-varying systems (OGATA, 1987). The main disadvantage of the direct method is that obtaining successful results is not an easy task and experience and imagination is often required.

When a neural network is included in a control scheme the analysis of the overall closed loop stability becomes even more of a difficult task since the neural network structure bears no resemblance to that of physical systems. It could be argued that the equations representing the neural network could be written down and an overall closed loop system equation obtained, but even with a neural network containing a small number of processing elements the resulting equation would be extremely complex and difficult to work with.

To overcome the difficulties involved in using any of the above mentioned methods another method is proposed whereby an insight into the stability of the neural predictive control structure can be examined.

6.5.1 Proposed stability analysis method

The proposed method is based on the fact that when the neural network model of the process is introduced into the control structure it compensates for the non-linearity of the process, thus making the process behave in a pre-specified manner. It is therefore reasonable to assume that in many cases a closed loop neural network control system can be reasonably approximated by a linear model between set point, $y_r(t)$, and process response, $y(t)$. Determination of a linear model describing this response would then be useful for investigating the control systems dynamics and hence, its stability.

A linear process model was obtained from data records of past process responses to set point changes and the subsequent application of system identification techniques to estimate an optimal linear model fit to the data was applied as follows.

The response of the closed loop neural predictive control strategy was assumed to be represented by the linear ARX model

$$y(t) = \theta^T \Phi(t) + e(t) \qquad ...(6.1)$$

where

$$\theta^T = \left[a_1,...,a_{nl},b_1,...,b_{nl}\right] \qquad ...(6.2)$$

$$\Phi^T(t) = [-y(t-1),..,-y(t-n_l),y_r.(t-n_k-1),..,y_r.(t-n_k-n_l)] \qquad ...(6.3)$$

n_l and n_k are the order and the dead time of the linear model respectively and e(t) is a discrete white noise sequence. Using the data record an estimate of the parameter vector $\hat{\theta}$ can be obtained using the least squares method as was the case in this work. The order of the model and dead time were calculated by applying linear model order selection techniques (LJUNG, 1987).

The above technique was applied to the on-line neural predictive control response shown in Figure 6.1 where the prediction horizon N2 was set at one and λ was set equal to zero. Using linear model order selection techniques, the best model fit to the data was achieved with the dead time n_k=0 and model order n_l=2. The response of the estimated linear model output and that of the actual process output is shown in Figure 6.6 which clearly illustrates the ability of the linear model to represent the closed loop dynamics accurately.

An insight into the stability of the closed loop process was then obtained by examining the poles of the identified linear model, since the transfer function denominator of the model is an estimate of the closed loop characteristic polynomial. The poles of the identified linear model were found to be positioned at $0.3783 \pm j0.2275$, Figure 6.7, which are well within the unit circle of the z-plane indicating closed loop controller stability. Zero steady state offset in the control action was also confirmed by the identified linear model exhibiting unity steady state gain.

The response of the process when the prediction horizon N2 was set at three and λ was equal to zero, Figure 6.2, was also used to illustrate the above method. As before a model order of n_l =2 and dead time n_k=0 was found to give the best linear model fit. Figures 6.8 and 6.9 show the response of the actual process versus the

linear model fit and the position of the closed loop poles of the model respectively. Again, as in the previous example, the position of the poles, 0.4464 ± j0.1869, indicates that the closed loop system is stable. It was also observed that the identified linear model exhibited unity steady state gain as before.

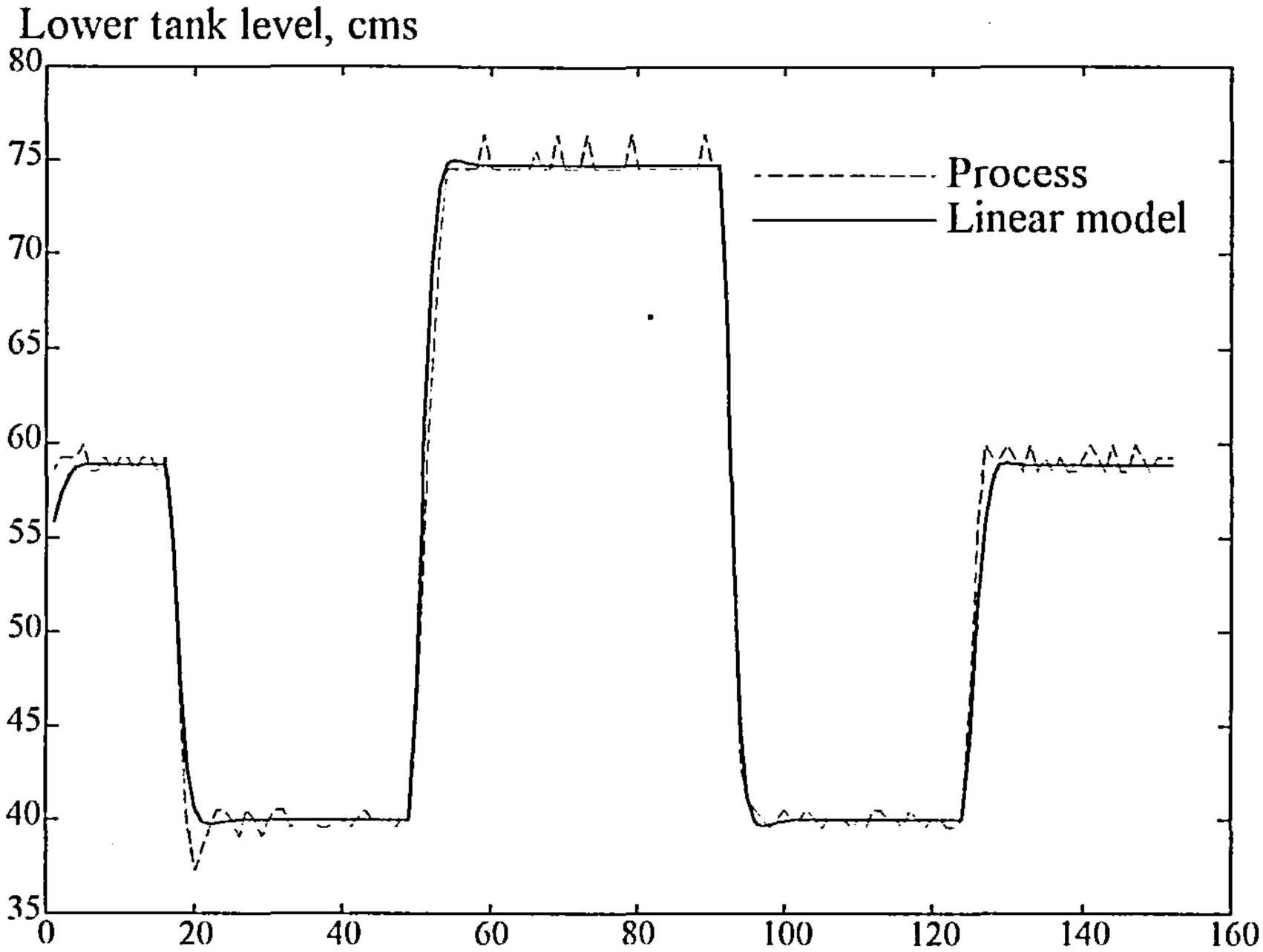


Figure 6.6 Comparison between process response and linear model fit, λ=0 and N2=1

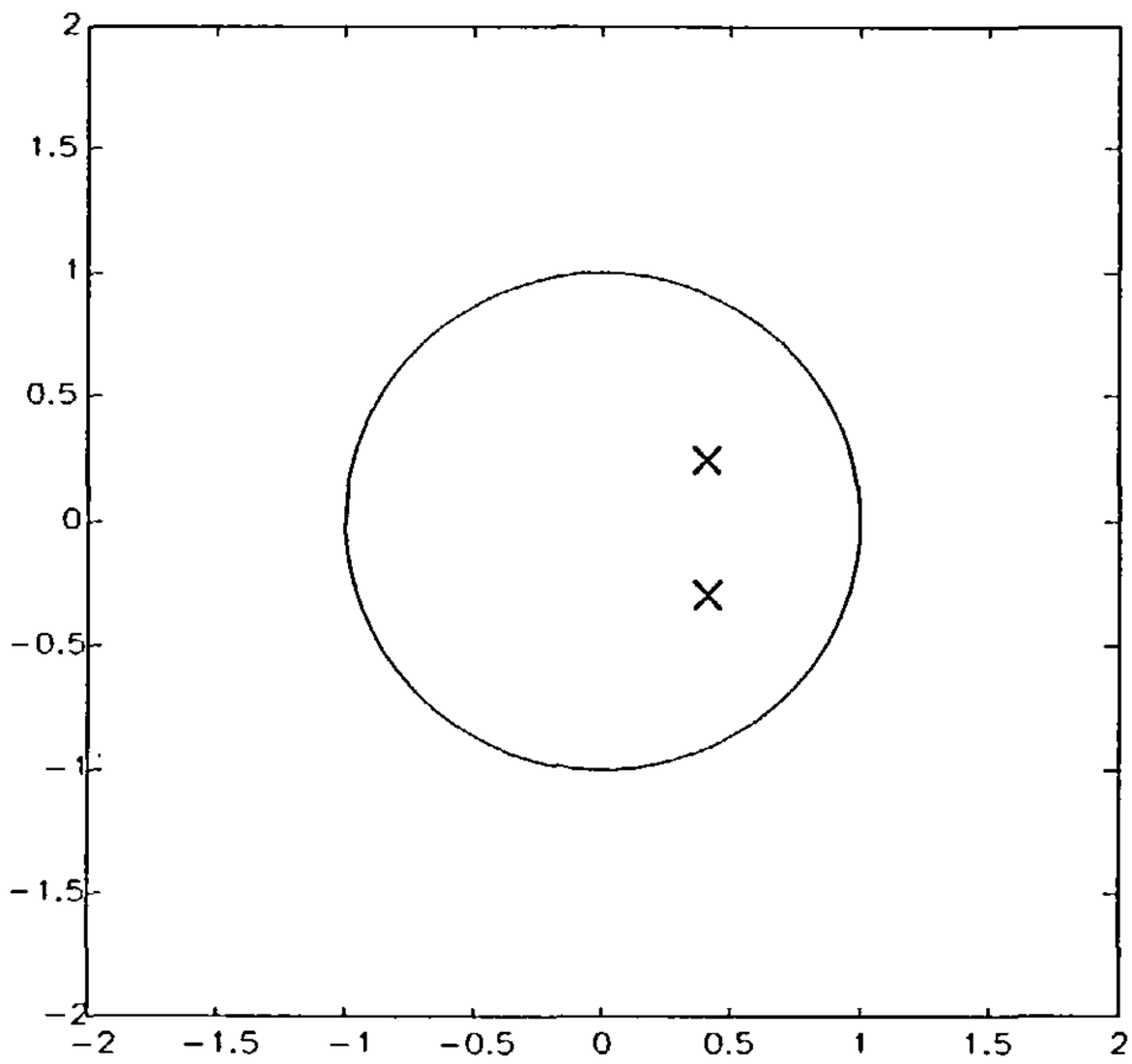


Figure 6.7 Location of poles for identified linear model

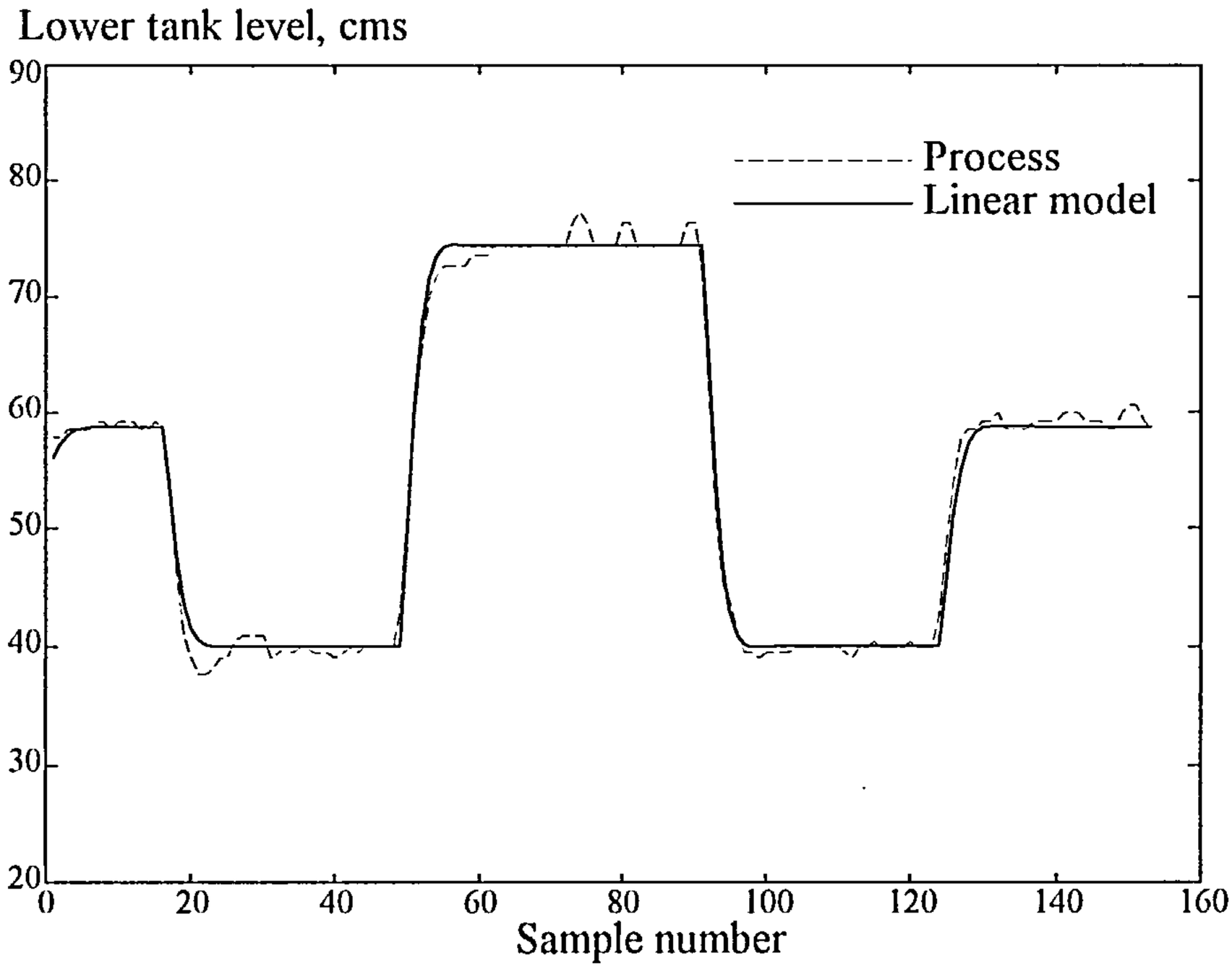


Figure 6.8 Comparison between process response and linear model fit, λ=0 and N2=3

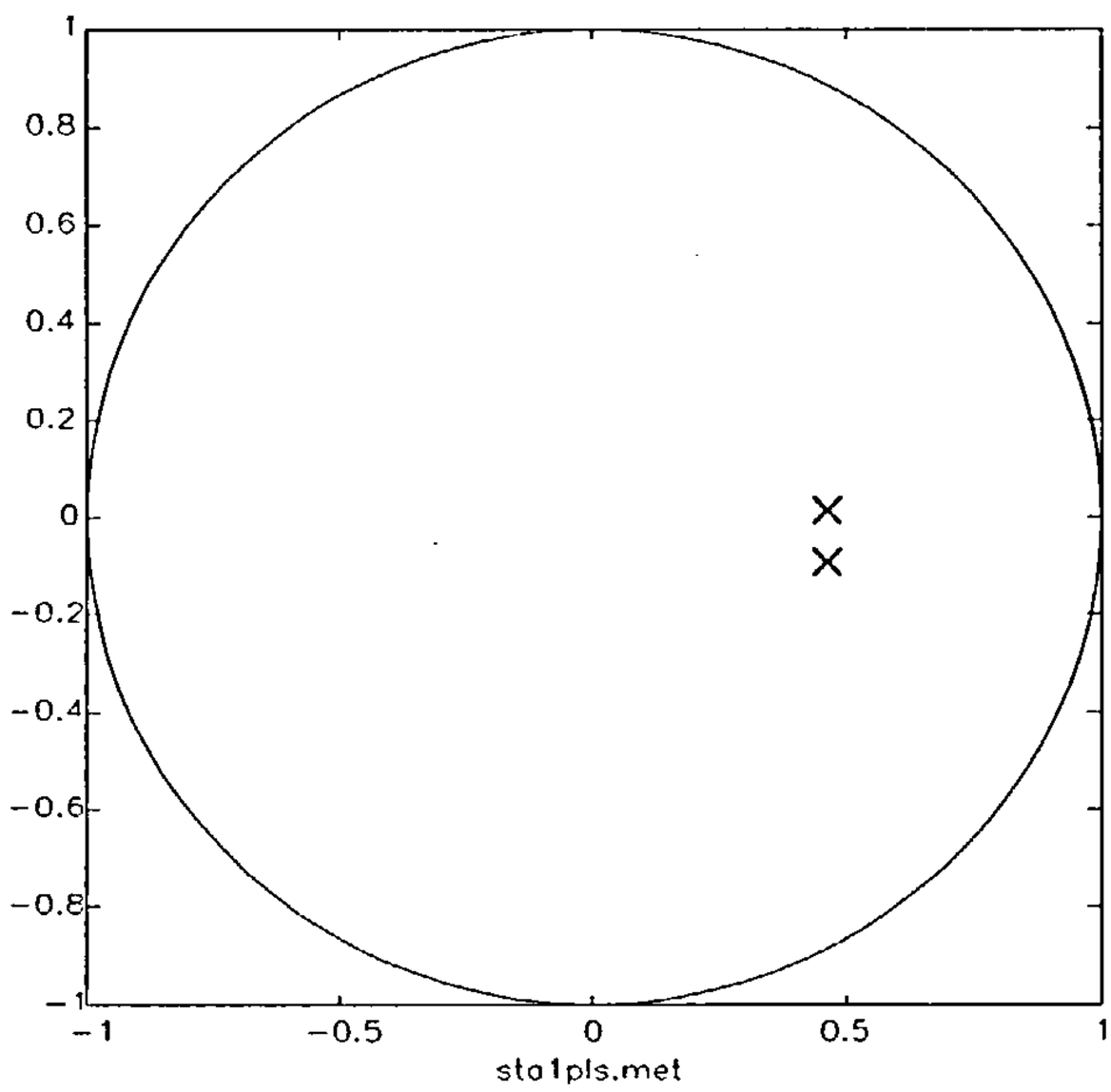


Figure 6.9 Location of poles for identified linear model

6.6 SUMMARY

The application of a neural predictive control strategy has been demonstrated on-line to the control of a non-linear dual tank liquid level process. The insights gained in the simulation studies in Chapter five were applied and this resulted in a smooth transfer to the real process. The technique was shown to be a viable one and comparisons made with a conventional PI controller illustrated an improvement in the tracking capabilities of the controller when subjected to a set of step inputs covering a wide non-linear operating region. Other commonly specified criteria for assessing control performance, such as process rise time, overshoot and control effort, were also investigated to assess the neural predictive control scheme when the prediction horizon was set at one-step-ahead and multi-step-ahead. Investigations showed that increasing the prediction horizon, N2, had a number of desirable effects. Good set point tracking was maintained whilst the control effort was considerably reduced. Results also showed an improved performance over the PI controller when multi-step-ahead prediction was performed.

An insight into the stability of the neural network control structure has been presented using techniques taken from linear system identification and this approach confirmed that, for both one-step-ahead and multi-step-ahead prediction, the control structure was input-output stable for step changes in the set point.

CHAPTER 7

CONCLUSIONS

An artificial neural network model based control strategy was investigated and developed firstly in simulation studies, using a mathematical model of the process obtained through theoretical modelling, and then applied on-line to a real non-linear, dual tank, liquid level laboratory scale process. The research aims of developing a neural predictive control scheme capable of controlling the process in real-time were achieved.

The research demonstrated that the multi-layer perceptron neural network can be used as a general tool for modelling non-linear dynamical processes, and in particular it was found that conditioning the process data, using a technique called spread encoding, (Chapter 2), greatly increased the network's prediction accuracy when operating recursively as a model when compared to the standard method adopted for conditioning the process data.

Efficient procedures for training an MLP neural network and to optimise its performance as a dynamic modelling technique were described (Chapter 3). The importance of choosing the correct network input structure was addressed and a correlation based technique was used to give indications on the suitability of a chosen network input structure. If an incorrect network input structure is used then this can seriously affect the ability of the neural network to accurately predict process behaviour.

Practical aspects were addressed in Chapter 3 such as the need for a correct sampling period when collecting the input-output data from the process and the type of excitation signal required to excite the process over its operational region.

When obtaining process data, in order to train the neural network, it is important that the excitation signal is spread over the whole range of the input space. Initial studies using a PRBS excitation signal, commonly used in linear system identification, revealed that good identification of the process was not possible. Consequently a signal with a uniform random amplitude was used to excite the process. When using the RAS, care should be taken to ensure that the clock period of the excitation signal is adequately matched with the bandwidth of the process dynamics so that the process has sufficient time to respond to each change in input.

A number of control strategies that could be used with a neural network model as an integral part were described in Chapter 5, and the final choice of control structure implemented was neural predictive control. The reasons for this choice were that the neural network model of the process could be fully exploited to predict the process output multiple time steps ahead and that the neural predictive control strategy does not require an inverse process model, as in the case of a number of the other strategies. For both the simulation studies and the on-line control studies, the neural network model based control structure was compared to a conventional PID controller. For small set point changes, around the operating point that the conventional controller had been tuned, the PID control performance was very similar to the predictive control results. However, as the set point signal was extended over a wider non-linear region of process operation the performance of the conventional controller deteriorated.

When using the neural predictive control strategy to control the process, good set point tracking was achieved with the control weighting factor, λ, set at zero and the prediction horizon, N2, equal to one but this was at the expense of a large control effort. Increasing the value of λ to unity resulted in a much smoother and more

desirable control input signal but at the expense of a slower process response time. Results indicated that the use of multi-step-ahead predictions enabled both good set point tracking and a low control effort for both λ set at zero and one.

When an artificial neural network model of a process is included into a control strategy the main disadvantage of the approach is that the analysis of the overall closed loop stability is difficult since neural network models bear no resemblance to physical systems. A method has been proposed to overcome this difficulty using techniques taken from linear system identification. The technique was demonstrated and shown to provide an insight into the stability of the control scheme by monitoring the position of the poles of an estimated linear model between the process set point and output.

7.1 RECOMMENDATIONS FOR FURTHER WORK

7.1.1 On-Line Neural Network Learning

Throughout the investigations into modelling the process using a neural network (Chapters 3 and 4) the input-output data used for training the network was collected from the process and the neural network was trained off-line. An area that should be investigated is training the neural network on-line. This would require some sort of recursive training algorithm, as used in the present adaptive identification systems, in order to update the network weights. The neural network used in this work, namely the MLP, would not be suitable for this task since the learning process is globally oriented, therefore an alternative network structure would have to be investigated. A possibility for on-line adaptation could be the radial basis function network as this consists of local processing units in the hidden

layer and is thus more appropriate to on-line learning. Another advantage of the radial basis function network is that the training algorithm is standard recursive least squares which is guaranteed to converge. However, on-line learning is by no means an easy task and would require considerable investigation.

7.1.2 Application To Other Processes

The research investigated was limited to the modelling and control of a single input-single output dual tank liquid level process. However, the methods investigated and techniques developed were intended to be applicable to a wide range of processes. The techniques developed should be applied to a range of other processes with a range of non-linearities in order to fully justify how generally applicable the techniques are. In particular, the application of neural networks to modelling non-linear multivariable processes is a challenging area that needs investigating. One of the main points would be how to excite a multivariable process in order to capture all the non-linearities that are present in the process, which may be difficult due to the interactions within the process.

7.1.3 Comparisons with other control structures

During this research project neural predictive control was investigated although, as mentioned in Chapter 5, a number of other control strategies are available for inclusion of a neural network process model. A comparison should be made with these other control strategies to thoroughly evaluate the use of the chosen control structure. This would also involve the use of inverse process models which would be obtained via a neural network approach.

REFERENCES

ADBY, P.R. and DEMPSTER, M.A.H. (1974) Introduction to optimisation methods. Chapman and Hall.

ALBUS, J.S. (1975) A new approach to manipulator control: the Cerebellar Model Articulation Controller (CMAC). Trans. ASME J. Dyn. Sys., Meas. and Control, vol.97, pp.220-233.

ALPER, P. (1965) A consideration of the discrete Volterra series. IEEE Trans. Aut. Control, AC 10, pp.322-331.

ASTROM, K.J. and WITTENMARK, B. (1984) Computer controlled systems-theory and design. Prentice Hall, Englewood Cliffs, NJ.

BABA, N. (1989) A new approach for finding the global minimum of error function of neural networks. Neural Networks, vol.2, pp.367-373.

BARTO, A.G. (1990) Connectionist learning for control: An overview. In: Miller, W.T., Sutton, R.S. and Werbos, P.J. (eds), Neural networks for control, Ch.1, pp.5-58. MIT Press.

BHAT, N., MINDERMAN, P. and MCAVOY, T. (1989) Use of neural nets for modelling of chemical process systems. Proc. IFAC conf. on Dynamics and Control of chemical reactors, DYCORD, pp.169-175.

BHAT, N.V., MINDERMAN, P.A., McAVOY, T. and WANG, N.S. (1990) Modelling chemical process systems via neural computation. IEEE Control Systems Mag. vol.10, no.3, pp.24-30.

BILLINGS, S.A. (1980) Identification of non-linear systems-a survey. IEE Proc-D, vol.127, no.6, pp.272-285.

BILLINGS, S.A., JAMALUDDIN, H.B. and CHEN, S. (1992) Properties of neural networks with applications to modelling non-linear dynamical systems. Int. J. Control, vol.55, no.1, pp.193-224.

BILLINGS, S.A. and LEONARITIS, I.J. (1985) Input-Output parametric models for non-linear systems. Part I: Deterministic non-linear systems. Int. J. Control, vol.41, no.2, pp.303-328.

BILLINGS, S.A. and LEONARITIS, I.J. (1985) Input-Output parametric models for non-linear systems. Part II: Stochastic non-linear systems. Int. J. Control, vol.41, no.2, pp.329-344.

BILLINGS, S.A. and VOON, W.S.F. (1986). Correlation based model validity tests for non-linear models. Int. J. Control, vol.44, no.1, pp.235-244.

BREMERMANN, H.J. and ANDERSON, R.W. (1989) An alternative to back-propagation: A simple rule for synaptic modification for neural net training and memory. Internal report, Department of Mathematics, University of California Berkley.

BURGHES, D. and GRAHAM, A. (1980) Introduction to control theory including optimal control. Ellis Horwood.

CARPENTER, G.A. and GROSSBERG, S. (1985) Category learning and adaptive pattern recognition, a neural network model. Proc. third Army conference on Applied mathematics and computation, ARO Report 86-1, pp.37-56.

CHEN, S., COWAN, C.F.N., BILLINGS, S.A. and GRANT, P.M. (1990) Parellel recursive prediction error algorithm for training layered neural networks. Int. J. Control, no.6, pp.1215-1228.

CHEN, V.C. and PAO, Y.H. (1989) Learning control with neural networks. Proc. IEEE Int. Conf. on Robotics and Automation, vol.3, pp.1448-1453.

CLARKE, D.W. and GAWTHROP, P.J. (1975) Self tuning controller. IEE Proc-D, vol.122, no.9, pp.929-934.

CLARKE, D.W., MOHTADI, C. and TUFFS, P.S. (1987) Generalised predictive control Part 1. The basic algorithm. Automatica, vol.23, no.2, pp.137-148.

CLARKE, D.W., MOHTADI, C. and TUFFS, P.S. (1987) Generalised predictive control Part 2. Extensions and interpretations. Automatica, vol.23, no.2, pp.149-160.

COON, G.A. (1956) How to find controller settings from process characteristics. Control Eng. vol.3, no.5, 66.

CYBENKO, G. (1989). Approximation by superposition's of a sigmoid function. Mathematics of Control, Signals and Systems, vol.2, pp.303-314.

ECONOMOU, C.G. and MORARI, M. (1986) Internal model control part 2, extension to non-linear systems. Ind. Eng. Chem. Process Des. Dev, vol.25, pp.403-411.

ELSELY, R.K. and LAN, M.S. (1991) Applications of neural networks to adaptive control. Proc. 22nd Asilomar conference on Signals, systems and control, vol.2, no.2, pp.517-522.

FAHLMANN, S.E (1988) An empirical study of learning speed in back-propagation networks. CMU Technical report, CMU-CS-88-162.

FASOL, K.H. and JORGL, H.P. (1980) Principles of model building and identification. Automatica, vol.16, pp.505-518.

FONG, K.F. and LOU, A.P. (1991) Discrete time optimal control using neural nets. IJCNN Singapore, vol.2, pp.1355-1360.

FUKUSHIMA, K., MIYAKE, S. and ITO, T. (1983) Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE Trans. Sys. Man. and Cybernetics, vol.13, pp.826-834.

FUNAHASHI, K. (1989). On the approximate realisation of continuous mappings by neural networks. Neural Networks, vol.2, pp.183-192.

GARCIA, C.E. and MORARI, M. (1982) Internal model control. 1. A unifying review and some new results. Ind. Eng. Chem. Process Des. Dev, vol 21, pp.308-323.

GIROSI, F. and POGGIO, T (1989) Representation properties of networks: Kolmogorov's theorem is irrelevant. Neural Computation, vol 1. pp.465-469.

GODFREY, K.R. (1980) Correlation Methods. Automatica, vol.16, pp.527-534.

GOMM, J.B. (1991) On-line detection of fault conditions in controlled industrial processes. Ph.D. thesis, Liverpool John Moores University, UK.

GORMAN, R.P. and TERRENCE J.S. (1988) Learned classification of sonar targets using a massively parallel network. IEEE Trans. Acoustic, Speech and Signal Processing, vol.36, no.7, pp.1135-1140.

GUSTAVSSON, I. (1975) Survey of applications of identification in chemical and physical processes. Automatica, vol.11, pp.3-24.

HABER, R. and KEVICZKY, L. (1974) Non-linear structures for system identification. Periodica Polytechnica, Electrical Engineering, 18, pp.393-404.

HALL, C. and SMITH, R. (1992) Pitfalls in the application of neural networks for process control. In: Warwick, K., Irwin, G.W. and Hunt, K.J. (eds), Neural networks for control and systems, Chapter 12, pp.243-256, Peter Peregrinus Ltd.

HINTON, G.E. and SEJNOWSKI, T.J. (1983) Optimal perceptual inference. Proc. IEEE Conference on Computer vision and pattern recognition, Washington, DC, pp.448-453.

HOPFIELD, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. Proc. of the National Academy of Scientists, vol.79, pp.2554-2558.

HOPFIELD, J.J. and TANK, D.W. (1985) Neural computation of decisions in optimisation problems. Biological Cybernetics, vol.52, pp.141-152.

HORNICK, K., STINCHCOMBE, M. and WHITE, H. (1989). Multistage feed forward networks are universal approximators. Neural Networks, vol.2, pp.359-366.

HOSKINS, J.C. and HIMMELBLAU, D.M. (1988). Artificial neural network models of knowledge representation in chemical engineering. Comput. Chem. Engng., vol.12, no.9, pp.881-890.

HUNT, K.J. and SBARBARO, D. (1991) Neural networks for non-linear internal model control. IEE Proc-D, vol.138, no.5, pp.431-438.

HUNT, K.J. and SBARBARO, D. (1992) Studies in neural network based control. In: Warwick, K., Irwin, G.W. and Hunt, K.J. (eds), Neural networks for control and systems, Chapter 6, pp.94-121. Peter Peregrinus Ltd.

ISERMANN, R., LACHMANN, K.H. and MATKO, D. (1992) Adaptive control systems. Prentice Hall.

JORDAN, M.I. (1988) Sequential dependencies and systems with excess degrees of freedom. Coins Technical report 88-27, Department of Computer and Information Science, University of Massachusetts, Amherst.

JORDAN, M.I. and RUMELHART, D.E. (1991) Forward models: supervised learning with a distal teacher. Technical report no.40, Centre for Cognitive Science, MIT.

KHALID, M. and OMATU, S. (1992) A neural network controller for a temperature control system. IEEE Control Systems Magazine, vol.12, no.3, pp.58-64.

KOHONEN, T. (1972) Correlation matrix memories. IEEE Trans. on Computers, C- 21, pp.353-359.

KOLMOGOROV, A.N. (1957) On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. Dokl. Akad. Nauk SSSR, vol.114, pp.953-956.

KOSKO, B. (1987) Bidirectional associative memories. IEEE Trans. on Sys. Man. and Cybernetics, vol.5, pp.312-322.

LISBOA, P.J.G., GOMM, J.B., EVANS, J.T., WILLIAMS, D. and TO, Q.S. (1991) Investigation into the application of neural network models for system identification. Proc. IEEE colloquium on Neural networks in control and modelling of industrial processes, Polytechnic of Central London, 17 April, 1991, pp.8/1-8/5.

LJUNG, L. (1987) System identification: Theory for the user. Prentice-Hall.

LJUNG, L. and SODERSTROM, T. (1983). Theory and practice of recursive identification. MIT Press.

LUYBEN, W.L. (1973) Process modeling, simulation, and control for chemical engineers. McGraw-Hill.

MATYAS, J. (1965) Random optimisation. Automation and Remote Control, vol.26, pp.246-253.

McCULLOCH, W.S. and PITTS, W.H. (1943) A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophy., vol.5, pp.115-133.

MOODY, T.J. and DARKEN, C.J. (1989) Fast learning in networks of locally tuned processing units. Neural Computation, vol.1, pp.151-160.

MORARI, M. and ZAFIRIOU, E. (1989) Robust process control. Prentice Hall, Englewood Cliffs, NJ.

NARENDRA, K.S. and PARTHASARATHY, K. (1988) Neural networks and dynamical systems part 1: A gradient approach to Hopfield networks. Internal Report no.8820, Yale University.

NARENDRA, K.S. and PARTHASARATHY, K. (1989) Neural networks and dynamical systems Part 3: Control. Internal report no.8909, Yale University.

NARENDRA, K.S. (1990) Adaptive control using neural networks. In: Miller, W.T., Sutton, R.S. and Werbos, P.J. (eds), Neural networks for control, Ch.5, pp.115-142. MIT Press.

NARENDRA, K.S. and PARTHASARATHY, K. (1990) Identification and control of dynamical systems using neural networks. IEEE Trans. on Neural Networks, vol.1, no.1, pp.4-27.

NGUYEN, D.H. and WIDROW, B. (1990) Neural networks for self learning control systems.IEEE Control Systems Magazine, vol.10, no.3, pp.18-23.

OGATA, K. (1970) Modern control engineering. Prentice-Hall.

OGATA, K. (1987) Discrete time control systems. Prentice-Hall.

PARKER, D.B. (1985) Learning logic. Report TR-47, Cambridge, MA: Massachusetts Institute of Technology, Centre for computational research in economics and management science.

PSALTIS, D., SIDERIS, A. and YAMAMURA, A.A. (1988) A multi-layered neural network controller. IEEE Control Systems Magazine, vol.10, no.3, pp.17-20.

QIN, S.Z., SU, H.T. and McAVOY, T.J. (1992) Comparison of four neural net learning methods for dynamic system identification. IEEE Trans. on Neural Networks, vol.3, no.1, pp.122-130.

RUMELHART, D.E., HINTON, G E. and WILLIAMS, R.J. (1986). Learning internal representations by error propagation. In: Rummelhart D.E and McClelland J.L. (eds), Parallel Distributed Processing: Explorations in the Micro structure of Cognition, pp.318-362. MIT Press.

SAMAD, T. (1989) Back propagation extensions. Honeywell SSDC Technical Report, 1000 boone Ave. N., Golden Valley, MN 55427.

SARTORI, M.A. and ANTSAKLIS, P.J. (1992) Implementations of learning control systems using neural networks. IEEE Control Systems Magazine, vol.12, no.2, pp.49-57.

SEBORG, D.E., EDGAR, T.F. and SHAH, S.L. (1986) Adaptive control strategies for process control: a survey. AIChE J., vol.32, no.6, pp.881-913.

SEJNOWSKI, T.J. and ROSENBERG, C.R. (1987) Parallel networks that learn to pronounce English text. Complex Systems, vol.1, pp.145-168.

SODERSTROM, T. and STOICA, P. (1989) System identification. Prentice-Hall.

STREJC, V. (1981). Least squares parameter estimation. In: Isermann, R. (ed) (1981) System identification. Tutorials presented at the 5th IFAC symposium on identification and system parameter estimation. F.R. Germany, September 1979. Pergamon Press. pp.535-550.

TERZUOLO, C.A., McKEEN, T.A., POPPELE, R.E., and ROSENTHAL, N.P. (1969) Impulse trains coding and decoding. In: Terzuolo, C.A., Systems analysis to neurophysiological problems, University of Minnesota, Minneapolis, pp.86-91.

UNGAR, L.H. (1990) A bioreactor benchmark for adaptive network based process control. In: Miller, W.T., Sutton, R.S. and Werbos, P.J. (eds), Neural networks for control, Chapter 16, pp.387-402. MIT press, Cambridge, MA.

UNGAR, L.H., POWELL, B.A., and KAMENS, S.N. (1990) Adaptive networks for fault diagnosis and process control. Computers chem. Engng, vol.14, no.4/5, pp.561-572.

VOLTERRA, V. (1931) Theory of functionals and integrals and integro-differential equations. London: Blackie and Son Ltd.

WELLSTEAD, P.E. and ZARROP, M.B., (1991) Self-Tuning Systems, Wiley Int.

WERBOS, P.J. (1974) Beyond regression: new tools for prediction and analysis in behavioural sciences, Ph.D. thesis, Harvard University.

WERBOS, P.J. (1988) Generalisation of back propagation with application to a recurrent gas market model. Neural Networks, vol.1, no.4, pp.339-356.

WILLIAMS, R.J. and ZIPSER, D. (1989) A learning algorithm for continuously running fully recurrent neural networks. Neural Computation, vol.1, pp.270-280.

WILLIS, M.J., Di MASSIMO, C., MONTAGUE, G.A., THAM, M.T., and MORRIS, A.J. (1991). Artificial neural networks in process engineering. IEE Proc-D vol.138, no.3, pp.256-266.

WILLIS, M.J., MONTAGUE, G.A., Di MASSIMO, C.,THAM, M.T., and MORRIS, A.J. (1992). Artificial neural networks in process estimation and control. Automatica, vol.16, pp.1181-1187.

WRAY, J. and GREEN, G.G.C. (1991) Analysis of networks that have learnt control problems. Proc. IEE International conference CONTROL '91, Edinburgh, UK.

ZIEGLER, J.G. and NICHOLS, N.B. (1942) Optimum settings for automatic controllers.Trans. ASME, vol.64, no.11, pp.759-768.

APPENDIX A

PUBLISHED WORK

P.J.G. Lisboa, J.B. Gomm, J.T. Evans, D. Williams and Q.S. To, "Investigation into the application of neural network models for system identification," in Proc. IEEE Colloquium on Neural Networks in Control and Modelling of Industrial Processes, PCL London, 1991.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "Identification of non-linear input-output process models using neural networks," presented at Application of Neural Networks to Modelling and Control, Liverpool, March 1992.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "Non-linear process modelling with artificial neural networks," in Proc. 24th SCS Summer Computer Simulation Conference, Reno, USA, 27-30th July, 1992.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "Modelling a non-linear process using artificial neural networks," in Proc. Sixth IMA Conference on Control: Modelling, Computation, Information, Manchester, UK, 2-4th September, 1992.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "Modelling non-linear process dynamics using back-error propagation," Presented at Workshop on Neural Networks: Techniques and Applications, Liverpool, 7-8th September, 1992.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "A practical application of neural modelling and predictive control," In: Application of Neural Networks to Modelling and Control, G.F. Page, J.B. Gomm, D. Williams (Eds.), Chapter 6, Chapman & Hall, UK, 1993.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "An on-line predictive controller using a neural network process model," presented at Neural Networks and Fuzzy Logic Colloquium, Liverpool, March 1993.

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To, "An on-line application of a neural network based predictive control strategy," presented at Non-linear Control using Structural Knowledge and System models: Design and Application, IEE May 1993.

D. Williams, J.T. Evans, J.B. Gomm and P.J.G. Lisboa, "Application of an artificial neural network to on-line control of a process," presented at AIENG 93, Toulouse, France 29 June-July, 1993

J.T. Evans, J.B. Gomm, D. Williams and P.J.G. Lisboa, "Identification and on-line control of a non-linear process via an artificial neural network approach," Presented at Workshop on Neural Networks: Techniques and Applications, Liverpool, 13-14th September, 1993.

J.B. Gomm, J.T. Evans, D. Williams and P.J.G. Lisboa,"Development of a neural network model based controller for a non-linear process application" In: P.J.G. Lisboa and M.J. Taylor (Eds), Proc. Workshop on Neural Network Applications and Tools, Liverpool, England, IEEE Computer Society press, CA, 1994, pp.109-117.

J.T. Evans, J.B. Gomm, D. Williams and P.J.G. Lisboa, "Implementation and performance evaluation of an on-line neural network control scheme," presented at IEE Control 94, Coventry, 21-24th March, 1994.

J.T. Evans, J.B. Gomm, D. Williams and P.J.G. Lisboa, "On-line modelling and predictive control using a neural network," presented at Modelling, Simulation And Control in the Process Industry, Ottawa, Canada, 25-27th May, 1994.

J.B. Gomm, P.J.G. Lisboa, D. Williams and J.T. Evans, "Accurate multi-step-ahead prediction of non-linear systems using the MLP neural network with spread encoding," Accepted for publication in Trans. Inst. M.C., August 1994.