NEURAL NETWORK INFERENCE MEASUREMENTS APPLIED TO THE PEBBLE BED MODULAR REACTOR

DIRK WOUTER ACKERMANN M.Eng.

Thesis submitted for the degree Philosophiae Doctor in Electronic Engineering at the Potchefstroomse Universiteit vir Christelike Hoër Onderwys

Promoter: Prof. C. P. Bodenstein

March 2004
Potchefstroom

Abstract

Inference measurements with time-delayed feed-forward neural networks facilitates the inference of unknown variables from known variables in non-linear dynamic systems. This is based only on the mapping data of the known variable and variable to be inferred. For successful inference, several constraints have to be overcome. This is, the neural network should have the correct topology, the training data set characteristics must have inherent attributes to ensure generalisation and the training algorithm must be capable of finding an acceptable local minimum on the error surface. At present, the neural network topology is based on trial and error, while the generalisation capability of the trained neural network is tested by using test and validation sets.

Due to the lack of design methods for the topology of neural networks and the need for independent testing and validation, this thesis endeavours to develop a generalised method to find the optimum topology for accurate inference measurements. The aim is further to develop a method for judging the training set that could lead to generalisation without using test sets or validation sets. For this to be done, the training algorithm should succeed in finding a small enough local minimum on the error surface.

The developed methods are applied to a simulated model of the pebble bed modular reactor (PBMR).

Opsomming

Inferensiemetings met tydvertraagde vooruitvoer neurale netwerke maak dit moontlik om onbekende veranderlikes van bekende veranderlikes in nie-liniêre dinamiese stelsels af te lei. Dit word slegs gebaseer op die data van die bekende veranderlike en dié van die veranderlike wat afgelei moet word. Vir suksesvolle inferensie moet verskeie beperkings oorkom word. Die topologie van die neurale netwerk moet korrek wees, eienskappe van die opleidingsdata moet inherente attribute hê om veralgemening te verseker, en die opleidingsalgoritme moet in staat wees om 'n aanvaarbare plaaslike minimum op die fout oppervlakte te vind. Tans berus neurale netwerktopologie op proefondervinding, terwyl die opgeleide neural network se vermoë om te veralgemeen, met toets- en validasie stelle bepaal moet word.

Vanweë 'n gebrek aan ontwerpmetodes vir die bepaling van die topologie van vooruitvoer neurale netwerke is dit nodig om onafhanklike toetsing en validasie te doen. Hierdie tesis stel hom dus ten doel om 'n algemene metode te vind om die optimum topologie vir akkuraat afgeleide metings te bepaal. Verder beoog hierdie studie ook om 'n opleidingstel te vind wat kan lei tot veralgemening sonder die gebruik van toets- of validasiestelle. Om dit te bewerkstellig, moet die opleidingsalgoritme in staat wees om 'n klein genoeg plaaslike minimum op die fout oppervlakte te vind.

Die ontwikkelde metodes word toegepas op 'n gesimuleerde model van die korrel bed modulêre reaktor (Pebble Bed Modular Reactor).

Acknowledgements

My sincere appreciation is extended to:

Prof. C. P. Bodenstein for his guidance.

MTech Industrial (Pty.) Ltd. for the use of the software package Flownet.

PBMR (Pty.) Ltd. for the use of information.

Mrs. E. Brand for proofreading this thesis.

Amanda for her moral support and motivation.

Nomenclature

A	arbitrary amplitude
B	bandwidth [Hz]
C	capacitance [F], heat capacity [kJ/kg.K]
e, E	electrical potential [V], energy [J]
f	frequency, arbitrary function
$FFNN$	feed-forward neural networks
i, I	electrical current [A], moment of inertia [$kg.m^2$]
k	arbitrary number
K	constant
L	inductance [H]
$\dot{m}$	mass flow rate [kg/s]
M	number of floating point weights
MSE	mean square error
N	number of neurons, number of bits
NN	neural networks
ODE	ordinary differential equation
p	pressure [Pa]
P	power [W]
$PBMM$	pebble bed micro model
$PBMR$	pebble bed modular reactor
Q	electrical charge [C], heat [J]
r	pressure ratio
R	resistance [Ω] , thermal resistance [$^oC/W$]
t	time [s]
T	temperature [K], time constant [s]
$\mathcal{T}$	torque [N.m]
TD	number of time delays
$TDFFNN$	time-delayed feed-forward neural networks
v, V	electrical potential [V]
w	neural network weight
W	work [J]
γ	specific heat ratio
Δ	difference
ΔT	time interval [s]
ζ	damping factor
σ	neural network activation function
θ	rotational angle [rad]
ω	angular velocity [rad/s]

Contents

Chapter 1

Introduction

This chapter gives the motivation for investigating inference measurements and the reason why feed-forward neural networks are used. Taking the shortcomings in the design methods for feed-forward neural networks into account, an overall aim and specific objectives are formulated. This chapter also describes the steps that are taken, with reference to other chapters, to reach these objectives.

1.1 Measurements

Measuring the variables of systems are vital for analysis or control. Measurements are performed by using sensors or transducers in order to derive a model of a system or to determine the system state. Measurements attempt to represent variables as accurately as possible, so that accurate modelling can be performed or the state of a system determined accurately.

For complete information on a system, noiseless ideal transducers, sampled at an infinitely high sampling rate and high resolution, are required. Noise, inherent to all physical systems, is nevertheless captured in the measurement process. For practical measurements, the sampling rate and resolution are limited, as is the length of the data records. It follows that for practical systems, the information is limited by the transducer error, noise inherent to the system, sampling noise, sampling rate and the finite length of the data records. An in-depth discussion on noise can be found in reference books such as [1]. In this thesis pseudo random noise within certain amplitude limits will be used to investigate the effect of noise on inference measurements.

Due to the non-ideal characteristics of the transducers, measurement errors could occur, depending on the characteristics of the specific sensors [2]. Typical characteristics that could result in measurement errors, are for example non-linearity, temperature dependence, hysteresis and a limited frequency response [3]. Many installed transducers, such as current transformers [4], were implemented for the purpose of current measurement at a fixed frequency and acceptable accuracy.

For a noiseless measurement process, the Nyquist sampling rate given in Equation 1.1, where f_s is the sampling rate and B is the bandwidth in Hz, ensures that a bandwidth-limited signal is uniquely represented by the data. Failure to adhere to the critical sampling rate causes aliasing in the frequency domain [5]. Since bandwidth and the rate of change of a signal are related, it is equally true that the sampling rate must be high enough to capture the rate of change of a signal (everywhere) to obtain a sufficiently high accuracy. Based on the limitations of the measurements itself and the processing limitations of the modeller, the modelling of systems has in practice limited accuracy and complexity.

$$f_s \geq 2B \tag{1.1}$$

In order to gather information on a system, it is necessary to have the system excited. It might be required to excite the system externally, or a system in operation could also be in an excited state already. It is important to realise, however, that the information gathered is limited to the operational range. System models are therefore not applicable outside the measurement range.

For a linear system, the Nyquist or critical sampling rate defines the minimum data rate, while the data is valid at any amplitude. For non-linear systems, however, such a simple rule is not sufficient to gather for accurate information on a system.

In the frequency domain, a system with non-linearities results in harmonics, subsequently, harmonic frequencies that are higher than the excitation frequency, exist in the system. This implies that the critical sampling rate must be based on the highest significant harmonic frequencies and not merely on the highest frequency of the system excitation signal. Furthermore, the amplitudes of harmonics generated in the system depend on the absolute amplitudes of the variables.

In the time domain, the rate of change of a variable depends on the rate of change of the stimulus and absolute amplitude of the variable. This implies that samples have to be taken at a sufficiently high sampling rate in order to capture the dynamics (rate of rise) of a non-linear system at various amplitudes.

Both these arguments imply that to map non-linear system dynamics, it has to be sampled at a considerably higher rate than its linear counterpart. Furthermore, the excitation of the system should cause a rate of change (or frequency) at various absolute amplitudes. Although this does not lead to an explicit formulation of excitation waveforms, excitation waveforms are devised in **Chapter 3: Training, test and validation sets**, causing satisfactory mapping of the system.

1.2 Inference measurement

In many cases direct measurement techniques are either inaccurate or impossible, subsequently, other methods of variable estimation are required. Such indirect measurements are referred to as derived or inferred measurements. An inference measurement is an estimate of a variable derived from other directly measurable variables [6]. As an example, Figure 1.1 illustrates a system with various interrelated variables. In this example, the variable D must be derived from the measurement of variables A, B and C. Information of the *system* characteristics enables the construction of the *inference engine*.

Inference measurement techniques can be applied to determine unknown variables or variables that are measured by using transducers with non-ideal characteristics. In general, the *inference engine* maps the input space $\mathbf{x_n}$ to the output space $\mathbf{y_m}$ via an input/output relationship $\mathbf{g_p}$ as shown in Figure 1.2. The vector presentation is shown in Equation 1.2 [7, p16]. Since models or inference requires the mapping of an input to an output via

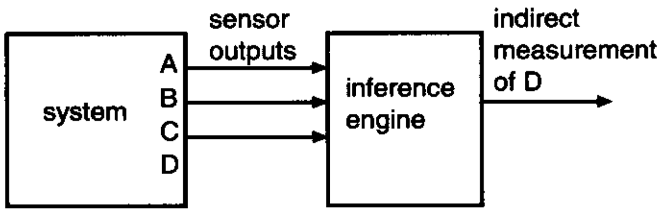

Figure 1.1: *Block diagram of inference measurement process*

some relationship, the same techniques used for modelling can be used for inference. For a transducer, the inference engine is the inverse mapping of the transducer itself.

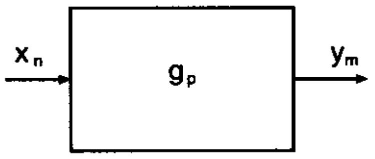

Figure 1.2: *Inference engine as a general system with input/output relationship*

$$\mathbf{g}_p : \mathbf{x_n} \rightarrow \mathbf{y_m} \tag{1.2}$$

Models that are best suited to the compensation problem at hand [8], are often derived. In the following sections several inference measurement techniques are discussed. These techniques are either model-based or information-based.

1.3 Inference measurement techniques

In this section two types of inference measurements techniques are discussed, namely model-based techniques and information-based techniques with neural networks as an example. Since most physical systems can be modelled as non-linear dynamic systems with smooth nonlinearities, this thesis will be limited to the inference measurements applied to such systems.

Variables in systems are interrelated and therefore it is possible, at least in principle, to derive an unknown variable from the other known variables if the unknown variable were observable. If every state variable affects the output of a system, the system is completely observable. If a state variable cannot be observed from the output of a system, the system is unobservable [9, p153]. For the purpose of this thesis, it will be assumed that an unknown variable to be inferred is observable.

1.3.1 Model-based inference measurements

Based on the model of the system, an unknown variable, if observable, can be calculated from the equations that describe the system. Model-based inference of both linear and non-linear dynamic systems will be discussed.

Linear dynamic systems

For linear dynamic systems with an input $x(t)$ and an output $y(t)$, the output/input characteristics of a system can be written in Equation 1.3. For an inference measurement, the output $y(t)$ is the directly measurable variable, while the input $x(t)$ is the variable to be inferred.

$$G(s) = \frac{Y(s)}{X(s)} \tag{1.3}$$

If the transfer function $G(s)$ has been obtained accurately, an accurate estimate of the input can be obtained from the output and the transfer function of the system as expressed in Equation 1.4.

$$X(s) = \frac{Y(s)}{G(s)} \tag{1.4}$$

In state variable form [10, p453] for a system state vector $\mathbf{x}$ and its derivative $\dot{\mathbf{x}}$, a linear system can be written as in Equation 1.5, where $[A]$ is the system matrix, $[B]$ the input matrix and $\mathbf{u}$ the input vector. With known characteristics of such a general multivariable linear system, the input $\mathbf{u}$ can be inferred based on the output of the system.

$$\dot{\mathbf{x}} = [A]\mathbf{x} + [B]\mathbf{u} \tag{1.5}$$

For example, in the system shown in Figure 1.3, with a transfer function $\frac{E_2(s)}{E_1(s)} = G(s)$, the input can be inferred from $E_1(s) = \frac{E_2(s)}{G(s)}$. Similarly, to infer another variable, (e.g. i_6 from i_3), algebraic manipulation results in a system of the same order, in this case, a system with two complex pole pairs.

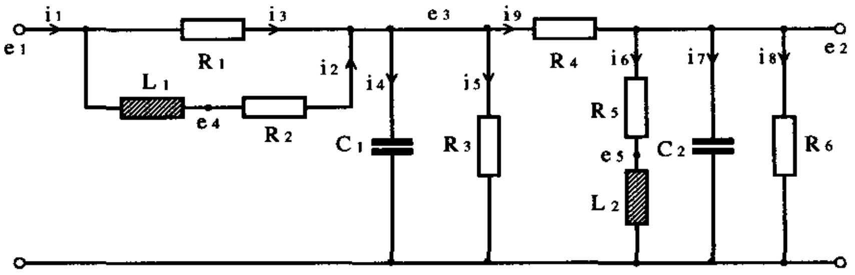

Figure 1.3: *Fourth order linear system*

Nonlinear dynamic systems

In contrast to a linear system with $\dot{\mathbf{x}} = [A]\mathbf{x} + [B]\mathbf{u}$, for non-linear systems, the matrix $[A]$ can no longer be separated [10, p453]. The vector-matrix state differential equations for a nonlinear system is given in Equation 1.6, or more simply, Equation 1.7 [9, p158].

$$\left.\begin{array}{l} \dot{x}_1 = h_1[\mathbf{x}, \mathbf{u}, t] \\ \dot{x}_2 = h_2[\mathbf{x}, \mathbf{u}, t] \\ \cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ \dot{x}_n = h_n[\mathbf{x}, \mathbf{u}, t] \end{array}\right\} \tag{1.6}$$

$$\dot{\mathbf{x}} = \mathbf{f}[\mathbf{x}, \mathbf{u}] \tag{1.7}$$

For a nonlinear sytem, the state equations can be determined from the parametric model of the sytem. Various methods can be used for determining the parametric model [11], applicable to electrical, chemical or mechanical processes. Once such a model has been found by some method, a measurement can be inferred by using state-space equations, and a computationally intensive numerical solution.

Assuming that an accurate parametric model is available, one method of finding a solution is to expand the nonlinear state equations into a Taylor series [9, p159] about a nominal operating trajectory $\mathbf{x}_0$. This should correspond with the nominal input $\mathbf{u}_0$ as shown in Equation 1.8 with $i = 1, 2, ..., n$, ignoring higher order terms. This results in a set of linear equations. Subsequently a variable can be inferred in the same manner as for a linear system. If a high accuracy were required, many terms of the Taylor series expansion are required, resulting in a large number of equations. Such a method is only valid close to the nominally chosen operating point.

$$\Delta\dot{x}_i = \sum_{j=1}^{n} \left.\frac{\delta f_i(\mathbf{x},\mathbf{u})}{\delta x_j}\right|_{\mathbf{x}_0,\mathbf{u}_0} \Delta x_j + \sum_{j=1}^{p} \left.\frac{\delta f_i(\mathbf{x},\mathbf{u})}{\delta u_j}\right|_{\mathbf{x}_0,\mathbf{u}_0} \Delta u_j \tag{1.8}$$

Another method of finding a solution is to use a linearised parameterised model for each applicable frequency. Because of the linearisation, superposition can be applied to find the solution for an arbitrary, frequency decomposable input signal. This method is common in power systems engineering, since specific harmonic frequencies that are sustained over a period of time are dominant.

1.3.2 Information-based inference

In the previous section, inference techniques were discussed, using a model of a system. To create such models usually requires a domain expert for each applicable technical field. Information-based inference does not require a model of the system. Variable inference is based on the data giving the relationship between the known variable and the variable to be inferred. Artificial neural networks are one such information-based inference measurement technique. It was chosen for inference measurement in this thesis because of their advantages as discussed in the following section.

1.3.3 Artificial neural networks

Artificial neural networks are loosely modelled on the interconnected neurons of a biological brain. A neuron of a biological brain could have several inputs of different strengths, which are summed at the neuron. Some threshold function acts on the summation results in the neuron output. With massive interconnection of neurons, a biological brain could process complicated functions. These complicated functions (e.g. the human ability to read) are possible because of the way the neurons are interconnected and the strength of the inputs.

Using artificial neural networks (mimicking parts of a biological brain), similar complicated functions can be achieved by the correct interconnectivity and training or learning. However, the field of artificial neural networks has developed considerably on its own. The similarities between a biological brain and artificial neural networks have become virtually irrelevant in order to understand and use artificial neural networks.

General problem-solving techniques attempt to map a set of inputs to a set of outputs. Mathematically, it is attempted to map an input space $\mathbf{x}_n$ in $\mathcal{R}^n$ to an output space $\mathbf{y}_m$ in $\mathcal{R}^m$. Problem-solving, such as model-based inference measurements, was achieved by programmed computing using a suitable programming language [12, p1]. Neurocomputing, using artificial neural networks, differs from problem-solving techniques, in that it uses programmed computing. These differences are reflected in Table 1.1 [12].

Table 1.1: *Comparison of programmed computing and neurocomputing*

programmed computing	**neurocomputing**
accurate system model required	only input/output data required
algorithm must be developed	network topology must be found
algorithm must be implemented	network must be trained
algorithm must be debugged	network must be tested

The most appealing advantages of neural networks for inference measurements are that:

i A parameterised model for the system is not required. This implies that the neural network inference measurement can be implemented directly from the data of the input/ouput relationship of the system to be modelled, as opposed to first having to derive a model and then finding a programmed solution based on the model.

ii Neural networks are fault-tolerant due to its inherent massive parallelism [7, p10]. In contrast, solving problems by means of programmed computing requires that accuracy and stability should be incorporated into the algorithm and implementation

[13].

The most important disadvantages of a neural network implementation are that:

i No clear rules exist for the design of network topology [7, p10].

ii The internal working of a neural network is hidden, and therefore a black box approach is taken [14, p205],[15].

iii The training procedure might not be successful in finding the global minimum by adjusting weights.

iv Generalisation cannot be guaranteed and must be validated [7, p10].

Neural networks have been successfully applied to system modelling and inferential measurements. Modelling examples are a non-linear dynamic electronic circuit [16], an extruder [17] and a function approximator [18]. Inferential measurements have been applied for nitrous oxide emissions from a gasline pumping station, feedwater flow in a nuclear reactor and sensor validation [19, p266].

In order to solve certain problems, different neural network topologies with applicable learning rules have been developed [20, p114]. The type of problem, the neural network topology that is used and the training algorithm are related as shown in Figure 1.4. A suitable combination of the problem type, neural network topology and training algorithm must thus be found to do inference measurements.

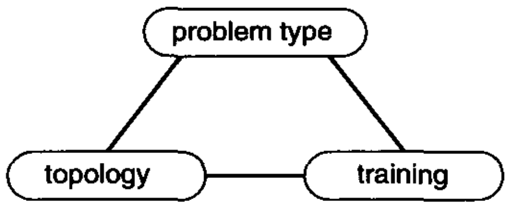

Figure 1.4: *The type of problem, topology and training are related*

Since the type of system used to infer variables is restricted to non-linear dynamic systems with smooth non-linearities, the topology and training should thus be chosen for such systems. Time-delay feed-forward neural networks are universal function approximators [7, p175] and therefore suitable for inference measurements. This will be discussed in the next section and investiged in this thesis.

1.4 Time-delayed feed-forward neural networks

In this section time-delayed feed-forward neural networks (TDFFNN) will be discussed. For successful neural network implementation, the neural network topology must be chosen for a specific modelling application. The training algorithm, using training sets, must be capable of finding a minimum that results in a trained neural network conforming to the inferrence error specification when tested or validated, using test and validation sets.

1.4.1 Neural network topology

The basic building blocks of artificial neural networks are neurons [21, p43]. A neuron, shown in Figure 1.5, has as input the input vector $\mathbf{x} = (x_0, x_1, ..., x_n)$ [20, p46]. The inputs could be bipolar $x_i \in \{-1, 1\}$, binary $x_i \in \{0, 1\}$ or continuous in the intervals $[-1, 1]$, $[0, 1]$, $(-1, 1)$ or $(0, 1)$. The x_0 term is often labelled the *bias* and set to 1 [12, p4]. These inputs $\mathbf{x}$ are multiplied with the weight vector $\mathbf{w} = (w_0, w_1, ..., w_n)$ and the function F is the sum of these products multiplied with an activation function σ so that the neuron output y is written as $y = \sigma \Sigma \mathbf{w}^T \mathbf{x}$.

With the activation function σ linear, only linearly seperable classes of problems can be solved. With σ being non-linear, non-linear classes of problems can be solved. The activation function σ is called the discriminator or squashing function [7, p78]. Several nonlinear functions have been implemented, such as the threshold function (hard limiter), linear function with upper and lower threshold, polynomial function and sigmoid functions [20, p48] [7, p78].

A feed-forward neural network consists of several layers of neurons organised so that the output of a layer is the input of the next layer. Feed-forward neural networks therefore consist of the superposition and composition of non-linear functions. Figure 1.6 shows a

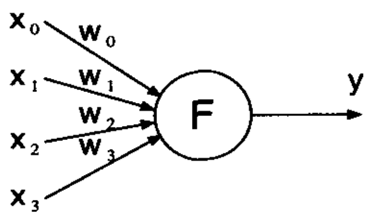

Figure 1.5: *The neuron is the basic building block af an artificial neural network*

three-layer feed-forward neural network. The output of this three layer neural network can be written as

$$\mathbf{y} = \sigma_3 \sum \mathbf{w_3}^T \left[\sigma_2 \sum \mathbf{w_2}^T \left(\sigma_1 \sum \mathbf{w_1}^T \mathbf{x} \right) \right] \tag{1.9}$$

where σ_1 is the activation function for the first hidden layer, σ_2 is the activation function for the second hidden layer and so forth.

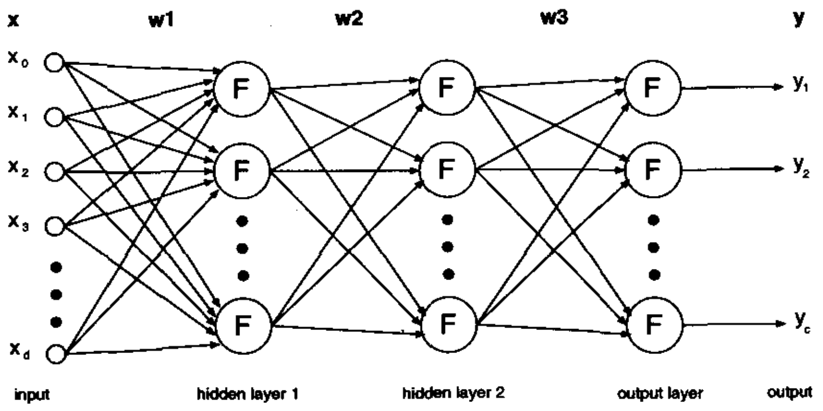

Figure 1.6: *Three-layer feed-forward neural network*

Given the topology shown in Figure 1.6, the question arises what type of functions can be estimated by Equation 1.9. It was shown [22] that a continuous function can be presented as

$$f(\mathbf{x}) = \sum_{0 \le q \le m} \chi[h^q(\mathbf{x})] \tag{1.10}$$

with h^q and χ continuous functions and $m \geq 1$.

Directly applicable to feed-forward neural networks, it was demonstrated rigorously [23] that functions with support in a unit hypercube can be approximated uniformly with a feed-forward network with a single hidden layer. It was shown that even with weights bounded, universal approximation can be achieved [24]. Whether more than one hidden layer is advantageous, was investigated [25]. It was found that, in general, single hidden layer neural networks perform better classification than two hidden layer neural networks [25].

For an input vector $\mathbf{x}$, with activation function σ, with n neurons in the single hidden layer, the output $\mathbf{y}$ of a single hidden layer feed-forward neural network can be written simply as

$$\mathbf{y} = \sum \mathbf{w_2}^T (\sigma \sum \mathbf{w_1}^T \mathbf{x}) \tag{1.11}$$

The number of neurons in the hidden layer required for a function mapping to be approximated within a certain error, depends on the specific function. Generally, with the error $\epsilon \rightarrow 0$, the number of neurons $n \rightarrow \infty$. It was shown [26] that for n neurons in the hidden layer, the approximation error e can decrease up to $e \propto \frac{1}{\sqrt{n}}$. For practical situations the number of neurons n will be limited, subsequently the function mapping can only be approximated. Apart from the practical limitations, it was shown that the required number of neurons n, for mapping an input/output relationship, are bounded by the the number of training patterns p. It was shown that for p training patterns, $p+1$ neurons are required for mapping the input/output relationship [27]. Failure to achieve acceptable approximation is due to faulty training, non-optimal number of neurons per layer or insufficient information captured in the training input/output data [7].

For modelling or inference, other types of neural networks have been implemented successfully, such as recurrent neural networks. A recurrent neural network can perform the same mapping with less memory than a time-delayed feed-forward neural network. However, stability is a difficult issue to deal with [28, p664]. In this thesis, feed-forward networks were chosen for inference measurements, due to their stability and since any function can be approximated within an arbitrarilary small error. Two disadvantages of neural networks are addressed in this thesis, namely that *no clear design rules exist for* *the required topology* and *generalisation cannot be guaranteed.*

In order to capture dynamic characteristics, the input vector **x** to the neural network is the time-delayed values of the variable x. The topology for the time-delayed feed-forward network is formed in such a manner. Figure 1.7 shows such a time-delayed feed-forward neural network with one hidden layer, capable of inferring variables of a non-linear dynamic system.

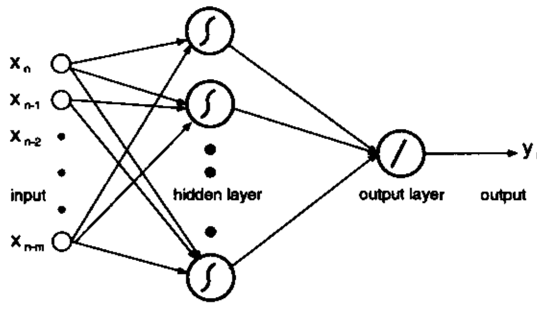

Figure 1.7: *Time-delayed feed-forward neural network with one hidden layer*

The topology of a time-delayed feed-forward neural network is defined by the number of inputs, outputs, layers, neurons per layer, the type of activation functions of the neurons, the way the neurons are interconnected and the number of time delays in these connections.

The number of time delays for the neural network depends on the system's dynamic range that needs to be captured. If a time step ΔT is small, fast dynamics can be modelled. Many time delays in the network result in information regarding the slow-varying dynamics of a system [28, p639]. Should a network be too large (too many time delays, layers or neurons) there is a possibility of overtraining. The network then tends to memorise the training set, rather than generalising or capturing the underlying system characteristics [29, p240]. For successful inference, it is thus desirable to have a neural network that has a sufficient number of time delays and neurons in the hidden layer to perform the required mapping, but limited in such a way that generalization can be accomplished. The minimum (but sufficient) number of time delays and neurons in the hidden layer, results in minimum processing requirements.

1.4.2 Training and testing

Artificial neural network training is done by an algorithm that seeks to minimise the output error with the training set as example. Using the error as feedback for weight adjustment, training is repeated until an acceptably small error has been achieved. Failure to reach the required small error can be caused by a topological limit or should the algorithm iterate indefinitely at a local minimum. After a certain time lapse it is expected that the adjustments to the weights would be small, subsequently a minimum has been reached under these constraints [7, p24].

The purpose of a training algorithm for a feed-forward neural network is to update the weight vector $\mathbf{w}$ to such an extent as to minimise the output error of the neural network. This is shown schematically in Figure 1.8. During training, the neural network is presented with a set of input vectors $\mathbf{x}$ and a set of output vectors $\mathbf{y}$ [12, p4]. During each training trial, the weights $\mathbf{w}$ are modified to such an extent that the neural network output approaches the desired output response.

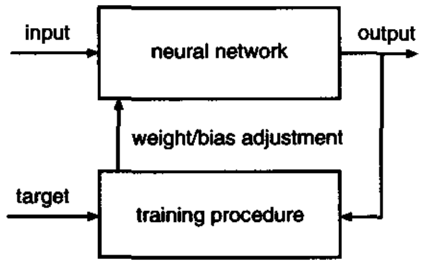

Figure 1.8: *Training procedure for a feed-forward neural network*

For single hidden layer feed-forward neural networks, the backpropagation algorithm is most commonly used [30]. In this algorithm, the error E , defined by

$$E = \frac{1}{2}\sum_{c}\sum_{j}(y_{j,c} - d_{j,c})^2 \tag{1.12}$$

for any set of weights must be minimized. In Equation 1.12, c is the number of input/output training pairs, j the index of output units, y and d are the actual and desired states respectively. For minimizing E by gradient descent, the partial derivative of E, with respect to each weight w in the network, must be found. Doing so, the weights can be adjusted according to

$$\Delta w = -\eta \frac{\delta E}{\delta w} \tag{1.13}$$

where η is the learning rate parameter. For convergence, the choice of the learning rate parameter η is critical. Choosing the learning rate η high, can result in fast learning, but can also lead to a zigzag training path or even to divergence. A low value for the learning rate η results in slow learning. Adjustments can be made either sequentially, or in a batch, where every $\frac{\delta E}{\delta w}$ is calculated before weight adjustments are made.

Gradient descent has proven to be slow, and one method to accellerate learning, is to modify the velocity of the point in weight space according to

$$\Delta w(\tau) = -\eta \frac{\delta E}{\delta w(\tau)} + \alpha \Delta w(\tau - 1) \tag{1.14}$$

for every epoch τ. An epoch is one sweep of adjustments made to the weights. The parameter α determines the contribution of the previous gradient.

Newton's method [28] attempts to minimise the quadratic approximation of the error vector $\mathbf{E}$ at a specific point in weight space. By expanding the error vector $\mathbf{E}$ into a second order Taylor series, adjustments to the weight vector $\mathbf{w}$ is made according to

$$\Delta \mathbf{w} = -\mathbf{H}^{-1}\mathbf{E} \tag{1.15}$$

where $\mathbf{H}$ is the Hessian of $\mathbf{E}$ and defined as $\mathbf{H} = \nabla^2\mathbf{E}$. Newton's method learns faster than gradient descent, but $\mathbf{H}$ must be nonsingular for the inverse to exist, the calculation of $\mathbf{H}$ is computationally expensive and convergence cannot be guarenteed. To overcome the difficulty in calculating the Hessian, other methods such as the conjugate gradient or the quasi-Newton method (which only requires an estimate of the gradient of the vector $\mathbf{E}$) have been developed.

The Gauss-Newton method [28] only makes use of the first order Taylor series. By expanding the error vector $\mathbf{E}$ into a first order Taylor series, an expression for the linearized error $\mathbf{E}$ as a function of the weight vector $\mathbf{w}$ can be found. By minimizing $\mathbf{E}$ for every epoch, an adjustment to the weight vector $\mathbf{w}$ is made [28, p126][31] according to

$$\Delta \mathbf{w} = -[\mathbf{J}^T\mathbf{J}]^{-1}\mathbf{J}^T\mathbf{E} \tag{1.16}$$

In Equation 1.16 the Jacobian $\mathbf{J}$ is the transpose of the gradient of the error $\nabla\mathbf{E}$. For the inverse of the Jacobian to exist, it must be nonsingular.

The Levenberg-Marquardt algorithm [32] is in essence an interpolation of the gradient descent and Gauss-Newton methods. The weight vector $\mathbf{w}$ with every epoch is updated according to

$$\Delta \mathbf{w} = -[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{E} \tag{1.17}$$

For a large value of μ, this algorithm approaches the steepest descent method (of which a step is proportional to the variable $\mathbf{E}$) and, for a small value of μ, the algorithm appoaches the Gauss-Newton method. The inclusion of the factor $\mu\mathbf{I}$ makes the matrix (to be inverted) nonsingular. The constant μ is multiplied by β should a step result in an increase of the sum of the square error. Should a step result in a decrease of the sum of the square error, μ is divided by β. It was shown [31] that the Levenberg-Marquardt algorithm performs well in comparison to other algorithms, and was subsequently chosen as the training algorithm for the feed-forward neural networks in this thesis.

A training algorithm can iterate indefinitely at a local minimum, and thus not be able to find the global minimum. It was proven that the error surface of a linear feed-forward neural network does not have local minima [33] when trained in batch mode. For non-linear feed-forward neural networks, it was found that a neural network capable of mapping a specific function, with the backpropagation training of a single hidden layer neural network, the error surface does not have a local minimum under certain training set constraints [34]. These training set constraints might not be met, and the error surface may have local minima.

The characteristics of the training set and the initialization of the weights can influence the success of a neural network implementation. Concerning the bias/variance dilemma as applicable to neural networks [35] it has been shown that training sets that are biassed, will result in a biassed training of a neural network (and not generalize well). To circumvent this, a training set can be selected that is less biassed, or the number of neurons in the hidden layer can be increased, resulting in more degrees of freedom and the algorithm is less likely to iterate indefinitely at a suboptimal point on the error surface.

A test set is used to establish whether acceptable generalisation and accuracy have been achieved with a trained neural network. Test sets are thus critical for an evaluation of the trained neural network. The trained network is accepted after having passed validation. Validation is an independent test, using an objective validation data set. The training set must span the whole required input space, otherwise certain characteristics of the system under investigation will not be mapped, and subsequently large errors are to be expected. A test set or validation set that spans a larger range than the input training set will not result in reliable output mapping, since neural networks are good at interpolation, but weak in performing extrapolation.

Requiring that the test and validation sets will test the trained neural network within all specified operating conditions, the training set must have certain characteristics to ensure that the trained neural network will pass both test and validation. Since the test or validation set output target is not visible for the modeller, while the test or validation set inputs (or equivalent specification) are, the relationship between the test (or validation) excitation as well as the training set excitation waveform should be established (if possible) to ensure accurate inference under all operating conditions.

1.5 Aim, objectives and benefits

From the previous section it was seen that no generalised design procedure exists for the design of the size of time-delayed feed-forward neural networks. The relationship between training set inputs and test inputs (or validation set) are not defined in concrete terms. The aim of this thesis is accordingly stated as follows:

The overall aim of this thesis is to develop a generalised design method for the implementation of inference measurements, using time-delayed feed-forward neural networks applicable to general non-linear dynamic systems.

From this overall aim, the following specific objectives are formulated, namely to:

1. Develop a design procedure for the number of neurons, layers and time delays that determine the size of the time-delayed feed-forward neural network based on the type of system from which variables must be inferred;
2. develop a design procedure for the characteristics of the training excitation waveform to ensure successful inference measurement implementation based on the test (or validation) excitation waveform characteristics or some operating range specification;
3. apply the design procedures to general non-linear dynamic systems with smooth non-linearities; and
4. demonstrate the neural network inference design method on a simulated model of the Brayton cycle as proposed for the pebble bed modular reactor.

This research has multiple benefits:

1. It will provide a generalised method for designing the size of time-delayed feed-forward neural networks for inference, resulting in an optimum neural network size for a certain specification, without exhaustive trial and error;
2. training excitation waveforms with the desired characteristics, as compared to test (or validation) excitation waveforms could ensure accurate inference measurements using time-delayed feed-forward neural networks without any knowledge of the system in any form; and
3. since the requirements for the neural network inference capability for a certain type of system are the same as the requirements for neural network modelling of such a system, the design procedures can also be used for neural network modelling.

1.6 Discussion

Artificial neural networks can offer an inference method that eliminates actual specific modelling in terms of component or equation parameters. Neural networks do not offer a "model" in an analytical form, and if not required explicitly, eliminates the necessity to find such a model. Furthermore, artificial neural networks can approximate complicated mapping requirements, complicated meaning a collection of energy storage elements and non-linear dissipative elements. Whether using traditional methods or neural networks for modelling, a mapping of the input/output relationship is required for all the expected operational conditions.

Due to the shortcomings in general design procedures for time-delayed feed-forward neural networks, namely the design of the neural network size and training excitation waveforms that ensure accurate inference, the overall aim and specific objectives were formulated.

Chapter 2: Network topology and system characteristics draws a relationship between system characteristics and the appropriate neural network topology for inference. From this relationship, design rules are established that lead to optimum neural network topology for a specific problem. **Chapter 3: Training, test and validation sets** shows that simulated data sets are generated correctly, as well as the effect of noise on the inference error. More importantly, it shows what the characteristics of training and test (or validation) excitation waveforms should be to ensure a small inference error without knowledge of the system input/output characteristics. **Chapter 4: PBMR neural network inference measurements**, applies the methods developed in the thesis to a simulated model of the basic Brayton cycle. Simulated data is generated by using Flownet, the thermodynamic software package used for the design of the PBMM and PBMR. **Chapter 5: Conclusion**, summarises this thesis and makes recommendations for future research. **Appendix A** lists the MATLAB files used in this thesis.

Chapter 2

Network topology and system characteristics

The purpose of this chapter is to develop design rules for a neural network topology that would meet the inference accuracy specification of variables on a non-linear dynamic systems. Based on the input/output mapping of the system, the neural network topology for the problem type can be selected and the size of the neural network determined by using the design rules.

With these methods, feed-forward neural networks for inference can be designed for non-linear dynamic systems, that are based only on the input/output mapping. It will be shown that two distinct topological requirements must be met, namely a layer with several non-linear neurons for performing non-linear static mapping, and time delays for mapping dynamics.

Since the neural network topology, training algorithm and problem type are related, the chosen topology must be capable of modelling the system within an expected error, assuming that such an error target is possible. Assuming that the topology has been chosen correctly, an expected error might not be reached due to the incapability of the training algorithm or due to the low information content of the data sets. Many time delays and many neurons in the hidden layer result in many neural network weights and a large memory requirement. This in turn tends to slow down the training process due to the increase in operations that must be performed during each training epoch. A minimum number of weights required to reach a specified accuracy uses the least computational resources.

The neural network topologies for inference on systems which are nonlinear, linear dynamic or non-linear dynamic were determined. The number of time delays required for dynamic mapping is determined explicitly from the input/output data of a system. The number of neurons in the hidden layer is determined by means of only a few training trials. Inference of a variable of a general non-linear dynamic system requires many weights. Therefore a topology was developed that requires only a fraction of such a number of weights for the same inference capability. Where there is an inference problem, an reduction in the number of weights results in a reduction of training and simulation time. Developed topologies for multivariable inference measurements are extentions of the topology for single input, single output inference of a variable of a general non-linear dynamic system.

2.1 Neural network implementation

The neural network topology and training used in this thesis were implemented by simulation using, MATLAB's neural network toolbox. The static or time-delayed feed-forward neural networks were simulated by using MATLAB's neural network toolbox `newff` function with appropriate time delays at the input as required for a specific experiment. For training, MATLAB offers many implemented training algorithms. From these, the Levenberg-Marquardt algorithm was chosen for its training speed for all neural networks. The implementation of this algorithm `trainlm`, using MATLAB, is given in **Appendix A: Software implementation**. For every simulation the input vectors and targets (the variable to be inferred) were selected from the system variables. These were obtained from solving the state equations using MATLAB's ODE solver `ode45` as described in **Appendix A: Software implementation**. For a specific experiment, a topology was chosen with an appropriate selection of time delays or neurons per layer.

The final training error and test error serve as measures of accuracy for the trained neural network system model. The output error of the model, excited by an input waveform, can be expressed either in mean square error (MSE) or the maximum absolute error in the time domain. With the target y_n and the network ouput $\hat{y}_n$, the mean square error is calculated from $MSE = \frac{1}{ndts}\sum|y_n - \hat{y}_n|^2$. The maximum absolute error in the time domain is given by $err_{max} = \max|y_n - \hat{y}_n|$. Both these measures were used as error measures of the neural network performance.

2.2 Static non-linearities

In this section the type of non-linearity and the required topology size to reach a specific error are compared. From this comparison a relation can be drawn between the non-linear element characteristics and the number of neurons required for accurate inference. Two types of non-linear dissipative elements were used in the experiments. Figure 2.1 shows a non-linear dissipative element, where the voltage over the element is given by $e = f(i)$.

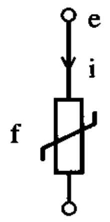

Figure 2.1: *Non-linear element*

The first type of non-linear elements is three tangent functions with input ranges of $[-0.9\ 0.9]$, $[-1.1\ 1.1]$ and $[-1.3\ 1.3]$ respectively. A large input range results in the largest deviation from a linear element. The tangent function was chosen because it is smooth and even and has a dominant spatial frequency of $f_x = 1$cycle/input range. Three sets of data for the $V - I$ characteristic were simulated, using equations 2.1, 2.2 and 2.3. Each tangent function was multiplied with a constant so that the output is a closest fit to linear function $\hat{e} = 0.5i$. These constants were determined by using MATLAB's `polyfit` function (see **Appendix A: Software implementation**).

$$e_5 = 0.458\tan(0.9i) \tag{2.1}$$

$$e_7 = 0.3293\tan(1.1i) \tag{2.2}$$

$$e_9 = 0.2249\tan(1.3i) \tag{2.3}$$

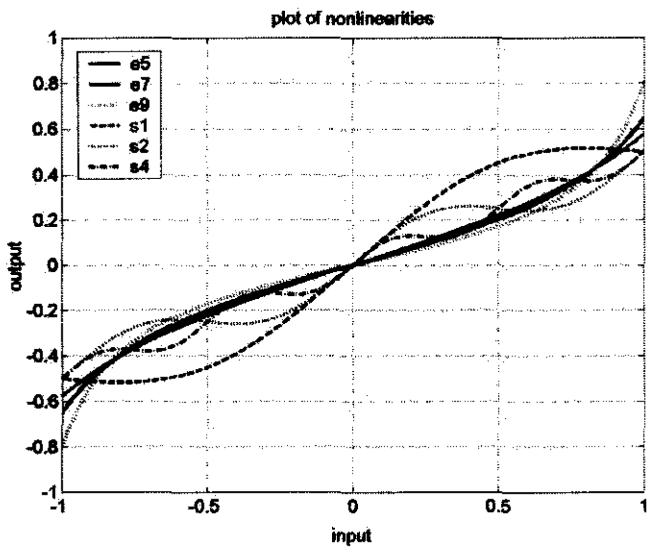

Figure 2.2: *Plot of non-linear elements*

The second type of non-linear components is the superposition of sinusoidal functions and a linear function $f = 0.5i$. The spatial frequencies of the sinusoidal functions were chosen as 1, 2 and $4Hz$cycles/input range respectively, as shown by equations 2.4, 2.5 and 2.6. The amplitudes of the sinusoidal part are scaled so that the maximum rate of change of the three functions are equal. Figure 2.2 shows the plots of the non-linear components. Figure 2.3 shows the spatial frequency components of the non-linear functions.

$$s_1 = 0.5i + 0.2\sin(\pi i) \tag{2.4}$$

$$s_2 = 0.5i + 0.1\sin(2\pi i) \tag{2.5}$$

$$s_4 = 0.5i + 0.05\sin(4\pi i) \tag{2.6}$$

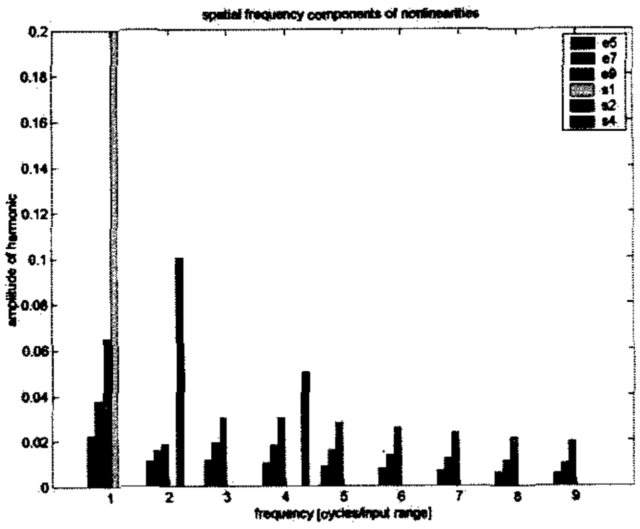

Figure 2.3: *Spatial frequency components of nonlinear functions.*

A network topology consisting of a hidden layer with `tansig` activating functions and a single linear neuron in the output layer was used, as shown in Figure 2.4. Training sets were generated with the independent variable i linearly increasing $-1 \leq i \leq 1$. By increasing the number of neurons in the hidden layer for every training experiment, the mean square error MSE reached after 50 epochs for all the non-linearities were tabulated. The error goal was set at a mean square error of $MSE = 10^{-8}$. The training errors for the non-linear elements with an increasing number of neurons in the hidden layer are represented in Figure 2.5.

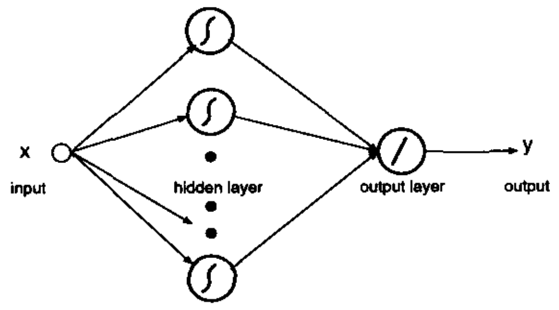

Figure 2.4: *Static feed-forward neural network with a hidden layer*

The characteristics of the non-linear functions and the neurons required for a mean square error $MSE \approx 10^{-8}$ are given in Table 2.1. The first-order polynomial fit for all the non-linear functions is $\hat{e} = 0.5i$. The characteristics for comparison of the non-linear functions are the spatial frequency f_x, the root mean square of the deviation from a linear function

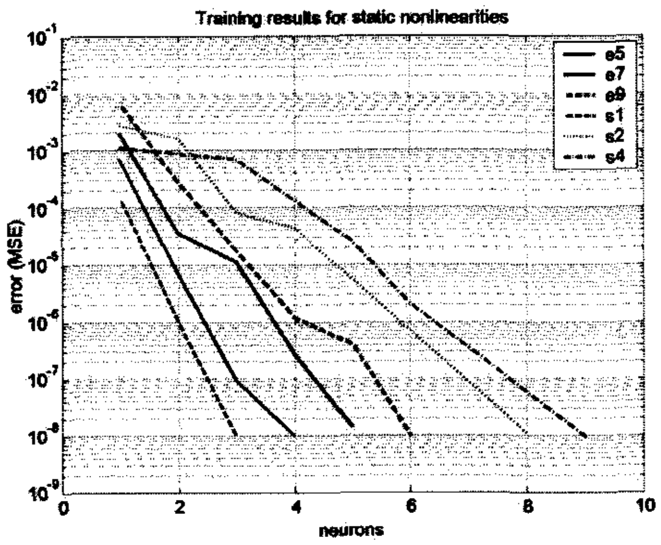

Figure 2.5: *Plot of training results for static non-linear functions*

$E_{RMS} = \sqrt{\sum (e_n - 0.5i_n)^2}$ and maximum rate of change $\max\frac{\Delta e}{\Delta i}$.

Table 2.1: *Characteristics of non-linear elements and required number of neurons N for a mean square error $MSE \approx 10^{-8}$*

function	spatial frequency f_x	RMS deviation $\sqrt{\sum(e_n - 0.5i_n)^2}$	rate of change $\max\frac{\Delta e}{\Delta i}$	neurons N
e_5	1	0.0258	0.2131	4
e_7	1	0.0448	0.3513	5
e_9	1	0.0803	0.8134	6
s_1	1	0.1414	0.2257	3
s_2	2	0.0707	0.2257	8
s_4	4	0.0353	0.2257	9

From Table 2.1 two relationships between the non-linear function characteristics and the number of neurons required for an error goal can be drawn:

1. *The number of neurons increases as the maximum rate of change* $\max\frac{\Delta e}{\Delta i}$ *increases.* Functions e_5, e_7, e_9 and s_1 have spatial frequencies f_x = 1cycles/input range. For the functions e_5, e_7 and e_9, the number of neurons increases as the maximum rates of change $\max\frac{\Delta e}{\Delta i}$ increases. The functions e_5 and s_1 have the same spatial frequency and approximately the same rate of change, but a large difference in RMS deviation

from a linear function. Approximately the same number (3 or 4) of neurons are required to reach the target error. For this reason the maximum rate of change $\max\frac{\Delta e}{\Delta i}$ of functions with the same spatial frequency is regarded as a reasonably characteristic attribute.

2. *The number of neurons increases as the dominant spatial frequency increases.* For the functions s_1, s_2 and s_4, the maximum rates of change $\max\frac{\Delta e}{\Delta i}$ are equal. With an increase in the spatial frequency of the function, the number of neurons required to reach a specific error goal increases, even though the RMS deviation from a linear function decreases.

2.3 Linear dynamic systems

A general transfer function for a linear system can be written in the form of Equation 2.7. The order of the system is the power of s in the denominator. By factorization, such a general transfer function can be written as the product of first and second order systems. Assuming that the dominant response of a system can be modelled as a first or second order system, first and second order systems are investigated in order to establish a design rule for the number of time delays required for accurate inference which is based on the system response.

$$\frac{E_2(s)}{E_1(s)} = \frac{K(s-a_1)(s-a_2)...(s-a_m)}{(s-\gamma_1)(s-\gamma_2)...(s-\gamma_n)} \text{ with } n > m \tag{2.7}$$

Linear dynamic system modelling requires a neural network with time delays and a single linear neuron. Figure 2.6 shows such a neural network. In this section, the relationships between the error as function of the time delays and the system dynamics are established.

Firstly, the time delays required for a neural network to perform integration and differentiation were established. Secondly, linear systems with decay were investigated in order to establish the time delays required for such systems.

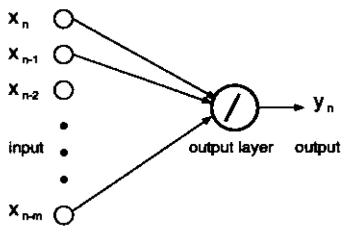

Figure 2.6: *Time-delayed feed-forward neural network*

2.3.1 Integration and differentiation

Integration $y(t) = \int_{-\infty}^{t} x dt$, or in the discrete case, a summation $y_n = \sum_{-\infty}^{n} x_n \Delta T$, is cumulative, meaning that the value y at a specific time is a function of all the previous values of x. This suggests that in order to perform an integration (or cumulative summation), values of the independent variable x are required at an infinite length of time in the past for the accurate evaluation of such a function. Subsequently, for a neural network to perform such cumulative summation, a very large number of input time delays are required for a reasonable estimate.

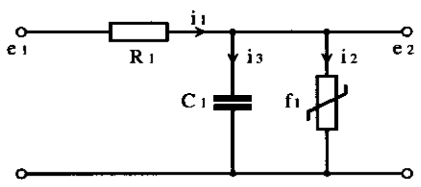

Figure 2.7: *First-order non-linear system*

Differentiation $y = \frac{dx}{dt}$ can be expressed in the discrete case as the difference $y = \frac{\Delta x}{\Delta t}$. In the discrete case, a value of x is required at a time t and a previous value for x at an infinitesimal small time in the past. Thus, it is at least required for the calculation of the difference in the discrete case to have the value for $x(n)$ at a time t and the previous value $x(n-1)$ at a time $t - \Delta T$. It is also true that for slow-varying functions, the difference can be estimated from $x(n)$ at a time t and a more distant previous value $x(n-2)$ at a time $t - 2 * \Delta T$ and so forth. This suggests that the neural network topology for performing difference must have at least one input time delay. The calculation of the difference is highly sensitive to noise or errors in the data set, therefore neural networks with more than one time delay were investigated to model the difference.

$$\dot{x}_1 = \frac{1}{C_1}\left[\frac{(e_1 - x_1)}{R_1} - f_1(x_1)\right] \tag{2.8}$$

The neural network training sets for cumulative summation and difference modelling were simulated by using a first-order non-linear system as shown in Figure 2.7. For the first order non-linear system shown in Figure 2.7, the state variable was chosen as $x_1 = e_2$. Equation 2.8 gives the state space equation for this system. For this system, $C_1 = 0.05F$, $R_1 = 1\Omega$, $i_2 = f_1(e_2) = K_{1a}\tan(K_{1b}e_2)$ with $K_{1a} = 0.2A$ and $K_{1b} = 3.0rad/V$.

The data of the $V - I$ relationship of the capacitor C_1 was used for the training set, from basic principles, by integrating $e_2(t) = \frac{1}{C_1}\int i_3 dt$ and by differentiating $e_2(t) = C_1\frac{d}{dt}i_3(t)$. For the discrete case, the cumulative summation is given by Equation 2.9 and the difference by Equation 2.10.

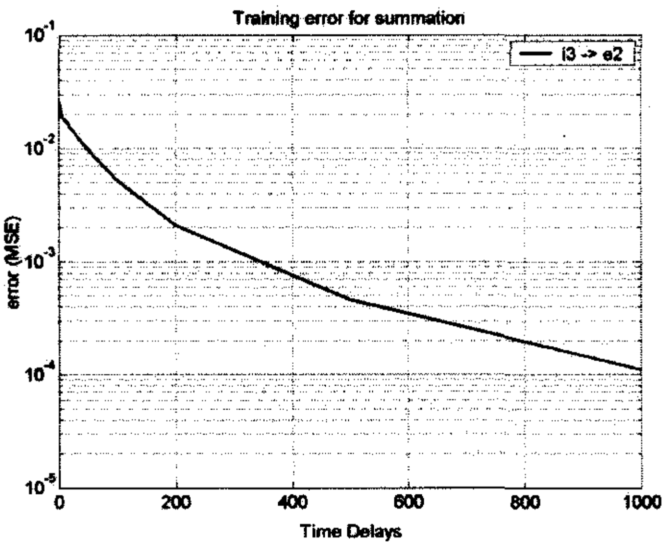

Figure 2.8: *Training error versus time delays for cumulative summation*

$$\widehat{e_2}(n) = \frac{1}{C_1}\sum_{1}^{n} i_3(n)\Delta T \tag{2.9}$$

$$\widehat{i_3}(n) = C_1\frac{e_2(n) - e_2(n-1)}{\Delta T} \tag{2.10}$$

With regard to inference where the target is the cumulative summation of the input, the simulated data for e_2 was used as the target of the neural network and i_2 is the neural network input. By changing the number of time delays for every training experiment, the relationship shown in Figure 2.8 was obtained.

With regard to inference where the target is the difference of the input, the simulated data for i_2 was used as the target and e_2 as the input. By changing the number of time delays for every training experiment, the relationship shown in Figure 2.9 was obtained.

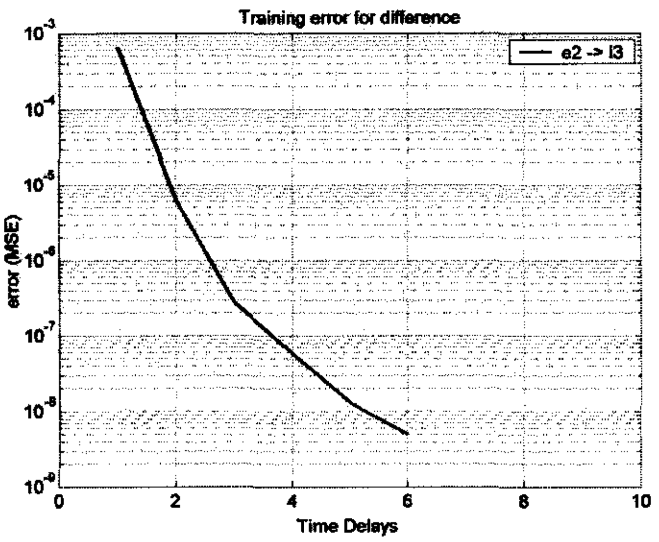

Figure 2.9: *Training error versus time delays for difference*

From Figure 2.8 it can be seen that a large number of time delays are required for inference where the input/output mapping is a cumulative summation, as was predicted. From Figure 2.9 it can be seen that inference where the input/output mapping is a difference requires only a few time delays as was predicted. Systems with poles at the origin, i.e. no time decay, cannot be modelled accurately for a limited number of time delays. Zeros at the origin, on the other hand, require only a few time delays for accurate modelling. Practical stable systems decay with time. The next section deals with linear systems with decay.

2.3.2 Dynamic systems with decay

It was seen from the previous results that systems with zeros at the origin (differentiators) require only a few time delays, hence variables can be inferred accurately. Systems with poles at the origin (integrators) require a very large (infinite) number of time delays.

Practical systems, however, decay, and in this section the experiments will show how many time delays are required for linear systems with no poles at the origin.

A phase lead circuit, a phase lag circuit, a parallel resonant circuit and series resonant circuit were investigated to establish a design rule for the number of time delays required for modelling linear dynamic systems. These systems are shown in Figure 2.10.

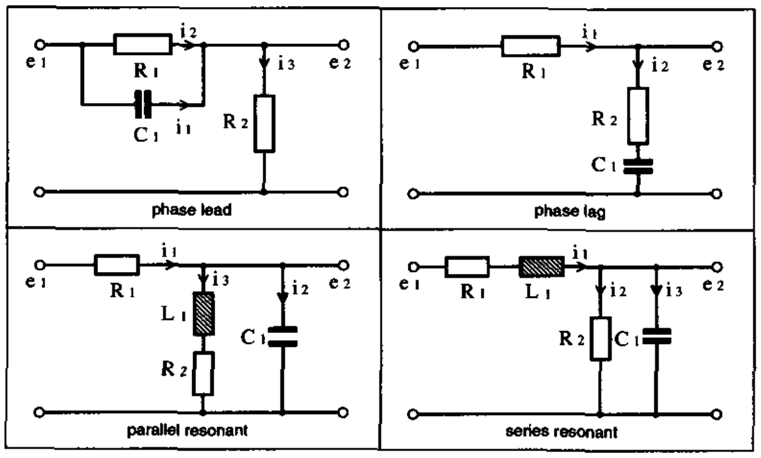

Figure 2.10: *Phase lead, phase lag, parallel resonant and series resonant systems*

A phase lead circuit has a transfer function of $\frac{E_2(s)}{E_1(s)} = \frac{1}{a}\frac{1+aTs}{1+Ts}$ with $a > 1$. The zero is at $-\frac{1}{aT}$ and the pole at $-\frac{1}{T}$. Furthermore, $aT = R_1C_1$ and $T = \frac{R_1R_2}{R_1+R_2}C_1$ with $a = \frac{R_1+R_2}{R_2}$ [9, p536]. For this circuit the chosen values are $R_1 = 1\Omega$, $R_2 = 1\Omega$ and $C_1 = 0.1F$. By choosing the state variable $x_1 = e_1 - e_2$, the state space equation for the phase lead circuit is obtained (Equation 2.11).

$$\dot{x}_1 = \frac{1}{C_1}\left[\frac{e_1-x_1}{R_2} - \frac{x_1}{R_1}\right] \tag{2.11}$$

A phase lag circuit has a transfer function of $\frac{E_2(s)}{E_1(s)} = \frac{1+aTs}{1+Ts}$ with $a < 1$ resulting in a zero at $-\frac{1}{aT}$ and a pole at $-\frac{1}{T}$. Furthermore, $aT = R_2C$ and $T = (R_1 + R_2)C_1$ with $a = \frac{R_2}{R_1+R_2}$ [9, p536]. For this circuit the chosen values are $R_1 = 1\Omega$, $R_2 = 1\Omega$ and $C_1 = 0.025F$. By choosing the state variable $x_1 = e_3$, the state space equation for the phase lag circuit is obtained (Equation 2.12).

$$\dot{x}_1 = \frac{1}{C_1}\left[\frac{e_1-x_1}{R_1+R_2}\right] \tag{2.12}$$

A parallel resonant circuit has a transfer function as given in Equation 2.13, which is of the form shown in Equation 2.14 with complex pole pair at $-\alpha \pm j\omega = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$ and $\alpha = \frac{R_1C_1R_2+L_1}{2R_1C_1L_1}$. $\frac{1}{\alpha}$ is proportional to the time constant of the system [9, p275]. For this circuit, the chosen values are $R_1 = 1\Omega$, $R_2 = 1\Omega$, $C_1 = 0.05F$ and $L_1 = 0.05H$. By choosing the state variables $x_1 = e_2$ and $x_2 = i_3$, the state space equations in Equation 2.15 is obtained.

$$\frac{E_2(s)}{E_1(s)} = \frac{1}{R_1C_1}\left[\frac{s+\frac{R_2}{L_1}}{s^2+\frac{R_1R_2C_1+L_1}{R_1C_1L_1}+\frac{R_1+R_2}{R_1C_1L_1}}\right] \tag{2.13}$$

$$\frac{E_2(s)}{E_1(s)} = K_1\frac{s+z}{s^2+2\zeta\omega_n+\omega_n^2} \tag{2.14}$$

$$\begin{aligned}\dot{x}_1 &= \tfrac{1}{C_1}\left[\tfrac{e_1-x_1}{R_1}-x_2\right]\\ \dot{x}_2 &= \tfrac{1}{L_1}\left[x_1-x_2R_2\right]\end{aligned} \tag{2.15}$$

The series resonant circuit has a transfer function as shown in Equation 2.16, the same form shown in Equation 2.17 with a complex pole pair at $-\alpha \pm j\omega = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$ and $\alpha = \frac{R_1R_2C_1+L_1}{2C_1L_1}$. With the chosen values $R_1 = 1\Omega$, $R_2 = 1\Omega$, $C_1 = 0.05F$ and $L_1 = 0.05H$. By choosing the state variables $x_1 = e_2$ and $x_2 = i_1$, the state space equations are obtained (Equation 2.18). $\frac{1}{\alpha}$ is proportional to the time constant of the system [9, p275].

$$\frac{E_2(s)}{E_1(s)} = \frac{R_2}{L_1C_1}\left[\frac{1}{s^2+\frac{R_1R_2C_1+L_1}{C_1L_1}+\frac{R_1+R_2}{C_1L_1}}\right] \tag{2.16}$$

$$\frac{E_2(s)}{E_1(s)} = K_2\frac{1}{s^2+2\zeta\omega_n+\omega_n^2} \tag{2.17}$$

$$\begin{aligned}\dot{x}_1 &= \tfrac{1}{C_1}\left[x_2-\tfrac{x_1}{R_2}\right]\\ \dot{x}_2 &= \tfrac{1}{L_1}\left[e_1-x_2R_1-x_1\right]\end{aligned} \tag{2.18}$$

The time response of a system is determined by the poles of the characteristic equation. For first order systems, such as the phase lead and phase lag circuits, the response decays at a rate of $e^{-\frac{t}{\tau}}$, where τ is the time constant of the circuit. For the phase lead circuit $\tau_{lead} = \frac{1}{T}$ and for the phase lag circuit $\tau_{lag} = \frac{1}{T}$. Circuits with complex pole pairs, such as the parallel resonant and series resonant circuits with $0 < \zeta < 1$, decay at a rate of $\frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t}$.

Since the output of these systems decay exponentially, it is required to capture the information of the system within the period of decay. An infinite number of measurements will result in an exact description of the system. However, as the output of the system decays, the transient response amplitude becomes smaller and subsequently less relevant. With noisy data, small amplitudes will be masked, and subsequently no useful information can be gathered from small amplitudes. For practical purposes it is thus only required to gather sufficient information, that is when the circuit has decayed to a sufficiently small amplitude.

In terms of neural network modelling, the number of time delays will depend on the system decay and the required accuracy.

In order to make a comparison of the four circuits, time delays for the phase lead and phase lag circuits and $\frac{1}{\alpha}$ for the resonant circuits were chosen as $0.05s$. With a choice of $R_1 = 1.0\Omega$ and $R_1 = 1.0\Omega$, this resulted in $C_1 = 0.1F$ (for the phase lead circuit), $C_1 = 0.025F$ (for the phase lead circuit), $L_1 = 0.05H$ and $C_1 = 0.05F$ (for the resonant circuits). Figure 2.11 shows the transient responses of the four circuits under investigation, with the phase lag circuit output display scaled down by 0.5 in order to have the same final value as the other circuits. It can be seen that the (enveloped) decay of the systems are more or less equal, as the choice of component values indicated.

The neural network for inference of the output variable e_2 from the input variable e_1 is a single linear neuron with different input time delays (see Figure 2.6) for every training experiment. The neural network training results for different values of time delays were plotted in Figure 2.12. It can be seen that, as predicted, the error decreases with the increasing number of time delays. As the number of time delays increased beyond a certain point, no significant improvement of the accuracy was achieved. The accuracy is therefore limited by the data characteristics and training algorithm capabilities.

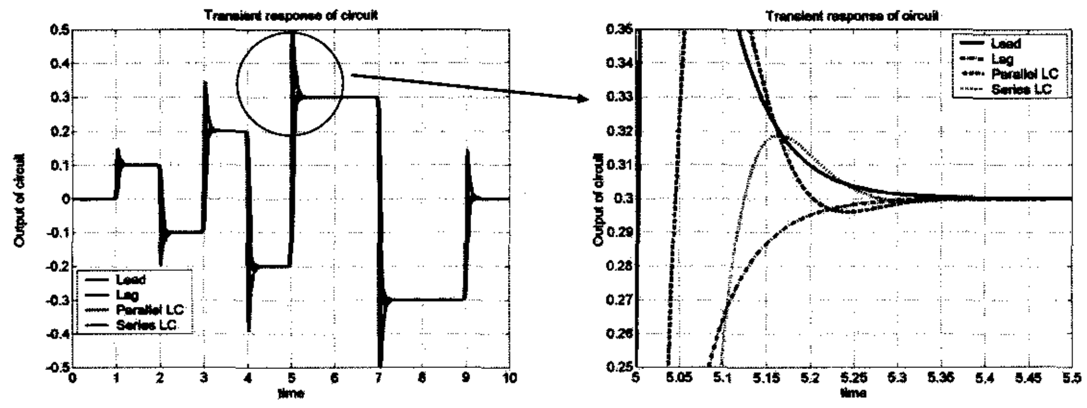

Figure 2.11: *Phase lead, phase lag, parallel resonant and series resonant systems transient responses*

An important consequence of these results are that the number of time delays required can be derived from the time lapsed till a system output decays to a small value, and not on the order of the system. Since the decay of a system response is largely determined by the dominant pole or pole pair, the time delays required can also be found from the time constant of a system, if available. The number of time delays required by a time-delayed feed-forward network for inference can be established from the transient response with a step-like excitation signal for systems such as studied here.

2.4 Nonlinear dynamic systems

The previous sections have dealt with the mapping of non-linear characteristics and the mapping of linear dynamic systems. It was shown that a layer of non-linear activation functions is required for non-linear mapping, while time delays are required for dynamic mapping. For a system with non-linearities and dynamics, both these topological entities are required. A layer of non-linear activation functions is required for non-linear mapping, while time delays are required for capturing the dynamics of a system. A neural network that has both these entities is shown in Figure 2.13. For a general non-linear system, the number of time delays must be established. It must also be established whether such a system has nonlinearities. Two types of dynamic systems will be considered, namely a system with dissipative non-linearities, and a system with storage non-linearities.

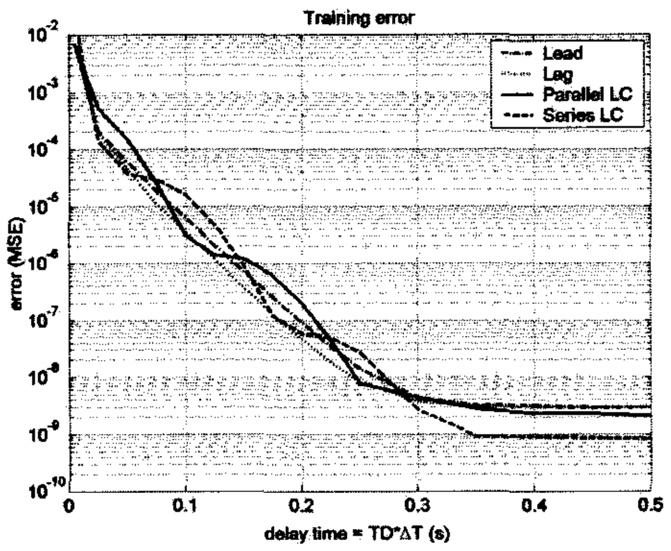

Figure 2.12: *Training error versus NN time delay for phase lead, phase lag and resonant circuits*

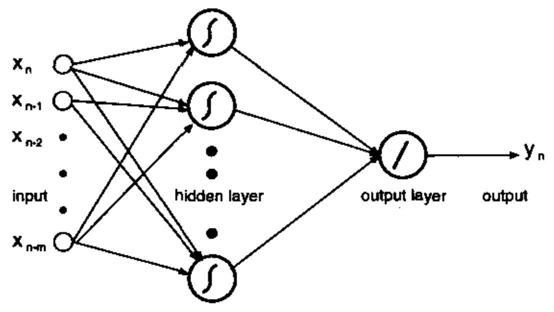

Figure 2.13: *Time-delayed feed-forward neural network with a hidden layer*

2.4.1 Dynamic system with non-linear dissipative elements

As an example of how the neural network topology can be determined for a non-linear dynamic system with non-linear dissipative elements, the second order dynamic system with non-linearities shown in the Figure 2.14 will be used. By choosing the state variables $x_1 = i_5$ and $x_2 = e_2$, the state space equation for this system is obtained (Equation 2.19). For this system, $L_1 = 0.05H$, $C_1 = 0.05F$, $R_1 = 2.0\Omega$, $e_4 - e_2 = f_1(i_5) = K_{1a}\tan(K_{1b}i_3)$ with $K_{1a} = 0.4V$, $K_{1b} = 3.0rad/A$ and $i_2 = f_2(e_2) = K_{2a}\tan(K_{2b}e_2)$ with $K_{2a} = 0.2A$, $K_{2b} = 2.0rad/V$.

$$\begin{aligned} \dot{x}_1 &= \tfrac{1}{L_1}\left[e_1 - f_1(x_1) - x_2\right] \\ \dot{x}_2 &= \tfrac{1}{C_1}\left[\tfrac{e_1 - x_2}{R_1} + x_1 - f_2(x_2)\right] \end{aligned} \tag{2.19}$$

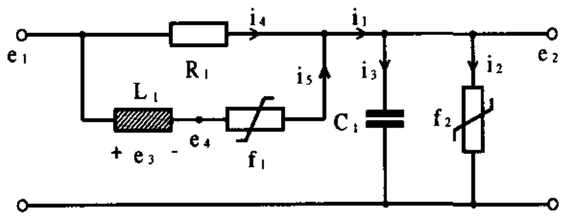

Figure 2.14: *Second order non-linear system*

The number of neurons in the hidden layer cannot be determined from the input/output data, since the non-linearities are not accessible when treated as a black box but hidden in the system. However, by observing the input output data, it can at least it can be shown that there are non-linearities. Two ways of showing that there are non-linearities are presented here.

The first method uses the data generated by applying a stepped input waveform as shown in Figure 2.15. At certain points, after the decay of the transient behaviour (such as at $t = 5.5s$), the input/output relationship is static. By plotting the input/output relationship at these static points (at $0.5s$ after a transient), the existence of non-linearities in the system (reflected at the output) can be observed. Figure 2.15 shows that the input/output at selected static points does not form a straight line. For a linear system superposition is valid and such a plot would result in a straight line.

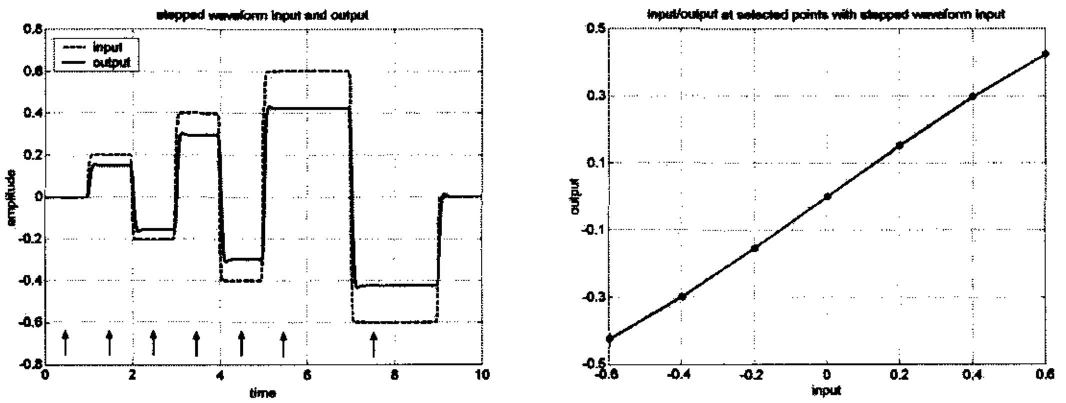

Figure 2.15: *Stepped waveform input and resultant output, and input/output plot at selected points*

When treated as a black box, the input/output relationship plotted in Figure 2.15 cannot reflect the characteristics of the non-linear elements in a system directly. The elements, or the equivalent function of the combined effect of the elements, are hidden. Subsequently the characteristic/neurons relationship derived in a previous section ob static non-linearites cannot be used to predict the number of neurons in the hidden layer that would lead to a specified inference accuracy. If a description of the non-linearities were known as in this experiment, a prediction can to some extent be made. With two non-linearities, it is predicted that a certain number of neurons are required for each non-linearity. By using Table 2.1, it thus predicted that the number of neurons $10 \leq N \leq 20$ should result in a mean square error of $MSE = 10^{-8}$.

The second method proposed here to show that there are non-linearities in the system, uses the fact that a slow-varying excitation waveform approaches static conditions. By plotting the slow-varying part of the input/output relationship shown in Figure 2.16, a distorted oval is formed. In contrast, the mapping of the input/output relationship of a linear system will result in an undistorted oval. Similarly, as with the first method, Figure 2.16 merely shows that there are non-linearities. No judgement can be made though on the actual hidden non-linearities characteristics that require mapping by the hidden layer.

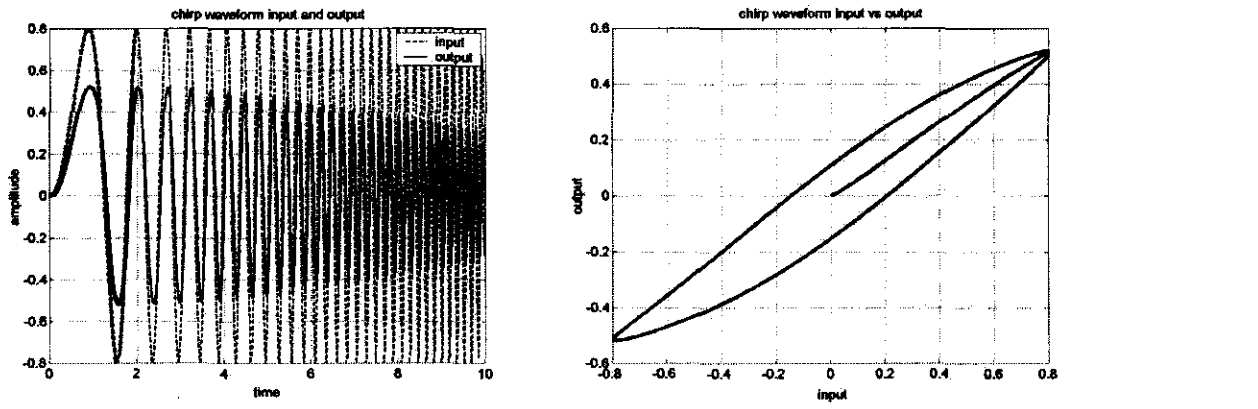

Figure 2.16: *Chirp waveform input and resultant output, and input/output plot of slow-varying part*

From the results of any of these two methods it can be observed that there are non-linearities in the system. Subsequently the neural network used for inference measurements must have one layer with non-linear activation units. Knowing the non-linearities in the system, the number of neurons required can to some extent be predicted.

In order to determine the number of neurons in the hidden layer sufficiently to model the dynamic system, a guess was made based as to the number of neurons required for non-linear mapping as was determined in the section on static non-linearities at the beginning of this chapter. Figure 2.5 shows that 3 to 9 neurons were sufficient to map the non-linearities that were used. Based on this observation and the fact that the equivalent effect of the hidden non-linearities is due to the combination of the separate non-linearities, three neural networks with three different numbers of neurons in the hidden layers were trained. The number of neurons chosen were 10, 20, and 30 respectively.

Modeling a system with dynamic characteristics requires input time delays. The number of time delays required depends on the system's decay in order to obtain a specific accuracy. The time required for decay of the transient response can be obtained from the time plot of the output of the system due to a step response. The stepped input waveform, consisting of several rapid changes, was applied to the system. The simulated data is shown in Figure 2.17, with an enlarged view showing the output decay. From Figure 2.17 it can be seen that after $t \approx 0.25s$, the output transient decayed to an insignificantly small value. Since the neural network input time delay has to capture this time decay, the number of time delays TD can be calculated from $TD = \frac{time\ decay}{\Delta T} = 50$ with $\Delta T = 0.005s$.

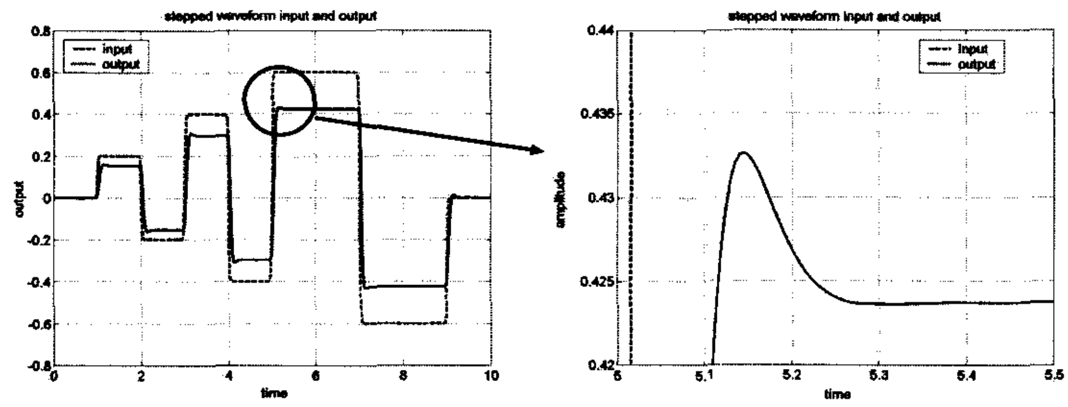

Figure 2.17: *Stepped input waveform and resulting output transient behaviour*

The training results for all three neural networks, differing only in the number of neurons in the hidden layer, are shown in Figure 2.18. From Figure 2.18 it can be seen that the choice of time delays, together with the informed choice of the number of neurons in the hidden layer, resulted in accurate inference with very little trial and error. Having to resort exhaustively to trial and error methods for determining the neural network size in terms of the number of time delays and number of neurons, has up to now been one of the major drawbacks encountered in implementing time-delayed feed-forward neural networks, as was pointed out in **Chapter 1 Introduction.**

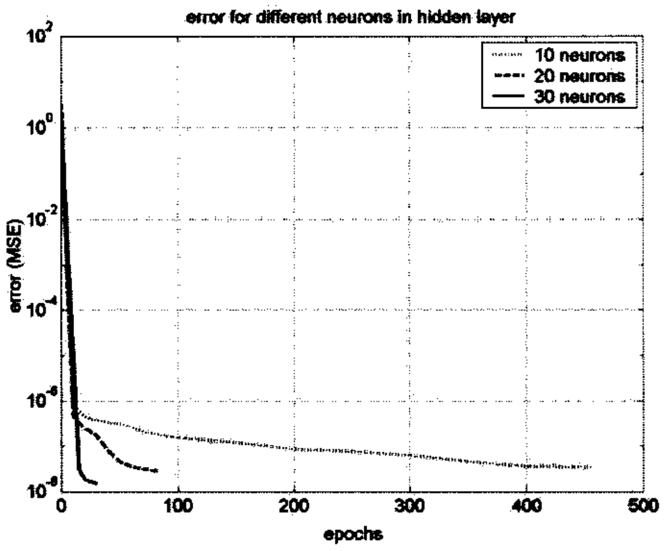

Figure 2.18: *Training results for 10, 20 and 30 neurons in the hidden layer*

2.4.2 Dynamic system with non-linear storage element

The previous section dealt with inference of a variable of a non-linear dynamic system with non-linear dissipative elements and linear storage elements. This section will illustrate that the same method can be used to determine the number of time delays and neurons in the hidden layer for variable inference in a system that includes a nonlinear storage element. Figure 2.19 shows a system with a non-linear storage element. The non-linear capacitor characteristic has been chosen as $C_1 = f_C(e_2) = K_C e_2^3$, since $Q_C = CV = K_C e_2^4$, $i_2 = \frac{dQ_C}{dt} = 4Ke_2^3\frac{de_2}{dt}$. The non-linear dissipative characteristic has been chosen as $f_1(e_1 - e_2) = K_1 tan K_2(e_1 - e_2)$. By choosing the state variable $x_1 = e_2$, the state equation for this system (Equation 2.20) is obtained. The parameters for this system are $K_C = 0.15C/V$, $K_1 = 0.2A$, $K_2 = 2rad/V$ and $R_1 = 2\Omega$.

$$\dot{x}_1 = \frac{1}{4K_C x_1^3}\left[f_1(e_1 - x_1) - \frac{x_1}{R_1}\right] \tag{2.20}$$

By using MATLAB's **ode45** solver with input excitation signals in the range of 0.3 $\leq$

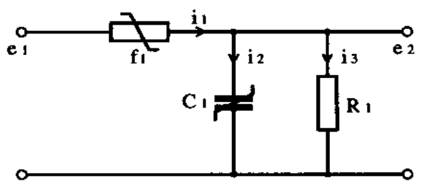

Figure 2.19: *First order non-linear system*

$e_1 \leq 0.9$, input/output data sets were simulated. Using a stepped input excitation signal as in Figure 2.20 and plotting the input/output relationship at static points (after the system has decayed), it can be observed from Figure 2.20 that there is a non-linearity in the system.

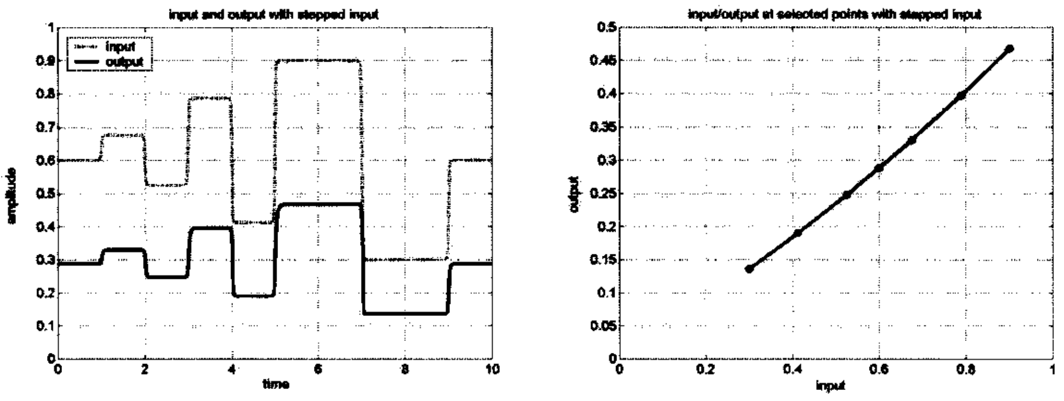

Figure 2.20: *Stepped waveform input and resultant output, and input/output plot at selected points*

By plotting the input/output relationship of the slow-varying part of a chirp input signal as shown in Figure 2.21, a distorted oval is produced. The existence of a non-linear storage element can be observed from Figure 2.21. This is because the input/output relationship produces a distorted oval, indicating that the storage element capacitance increases as the input amplitude increases.

Since the capacitance increases as the input amplitude increases, the time constant, and therefore the decay time of the system increases as the input amplitude increases. In order to find the number of time delays, a stepped input signal is required as well as the observation of the system output decay. This should be observed at a point where the time constant is the longest in order to be able to capture the slowest possible system dynamics that could exist. Since the capacitance increases as the input excitation signal amplitude increases, the output will decay the slowest at the maximum input amplitude.

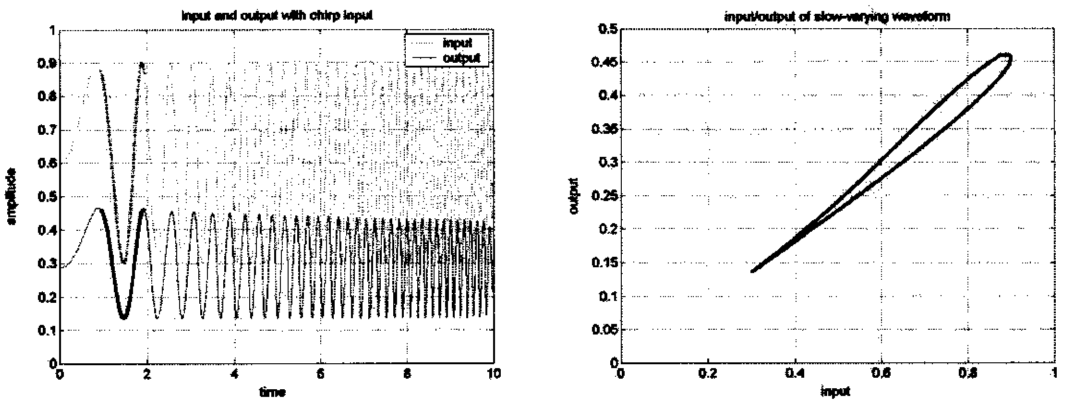

Figure 2.21: *Chirp waveform input and resultant output, and input/output plot of slow-varying part*

In Figure 2.22 the transient response of the system at various input amplitudes is shown with the stepped input signal. The slowest observed decay is at the maximum input amplitude. After a time delay of 0.25s, the output has decayed to a sufficiently small value. As before, with a time interval $\Delta T = 0.005s$, this will required 50 time delays in the input layer.

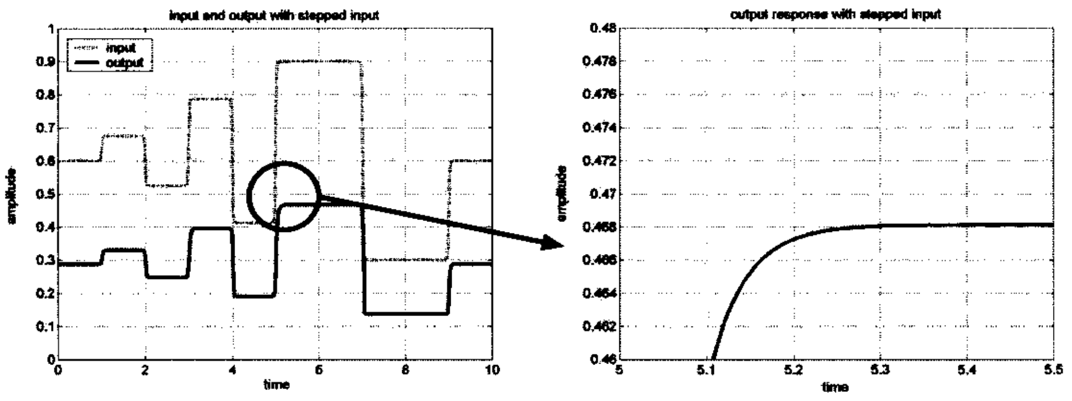

Figure 2.22: *Stepped input waveform and resulting output transient behaviour*

The number of neurons in the hidden layer cannot be obtained explicitly if the nonlinearities of the system were hidden. Each non-linearity of the system must be mapped by the neural network. The total number of neurons in the hidden layer is the sum of the number of neurons required for each non-linear mapping. Assuming that the non-linearities to be mapped have characteristics that fall in the range of the functions in Table 2.1, between 3 and 9 neurons will be required for the mapping of each non-linearity. This results in a number of neurons in the range $6 \leq N \leq 18$. Based on this rough estimate, three neural networks with different numbers of neurons (10, 20 and 30) in the hidden layer were simulated and trained. The training results in an error goal of $MSE = 10^{-8}$ and 50 training epochs, as shown in Figure 2.23 for the chosen number of neurons.

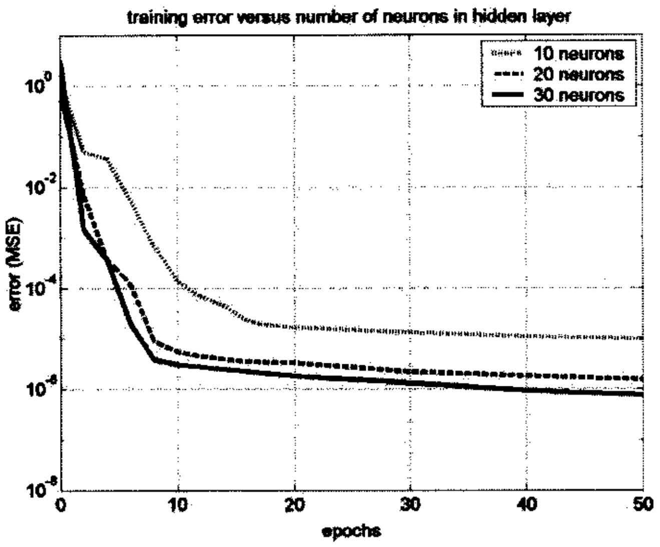

Figure 2.23: *Training results for 10, 20 and 30 neurons in the hidden layer*

Figure 2.23 shows that the range of numbers of neurons in the hidden layer resulted in a small training error. It can subsequently be concluded that:

1. It is possible to apply the methods for determining the number of time delays to a general nonilinear system.
2. An estimate of the number of neurons can be made with the help of some information of the system. If this proves to be impossible, choose progressively larger numbers of neurons in step changes until the error goal has been reached or no significant improvement has been accomplished with the increase in the number of neurons in the hidden layer.

2.5 Network memory

In the previous sections it was shown that the topology required for inference is determined by the problem type. The most general type of problem, a non-linear dynamic system with decay, requires input time delays and a hidden layer with non-linear activation functions. The number of weights of the neural network is determined by the number of time delays and the number of neurons in each hidden layer. Ignoring biases, the number of floating point weights M are given by Equation 2.21 (for Figure 2.4), Equation 2.22 (for Figure 2.6), Equation 2.23 (for Figure 2.13) and Equation 2.24 (for Figure 2.24) respectively.

$$
\begin{aligned}
M_1 &= 2 * N_{L1} &\quad (2.21)\\
M_2 &= TD &\quad (2.22)\\
M_3 &= TD * N_{L1} + N_{L1} &\quad (2.23)\\
M_4 &= TD * N_{L1} + N_{L1} * N_{L2} + N_{L2} &\quad (2.24)
\end{aligned}
$$

For static non-linear mapping, the number of weights M_1 are determined by the characteristics of the non-linear elements. For mapping any of the elements in the section **Static non-linearities**, using 10 neurons in the hidden layer, $M_1 = 20$. For linear dynamic mapping of systems with decay, the number of floating point weights are determined by the input time delays. For 50 input time delays, $M_2 = 50$.

Non-linear dynamic systems using the network topology of Figure 2.13, require a large number of floating point weights, determined mostly by the product of the time delays and number of neurons in the hidden layer.

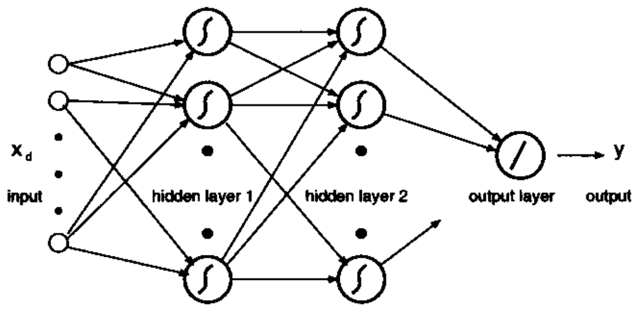

Figure 2.24: *Time-delayed feed-forward neural network with two hidden layers*

From the results in Figure 2.18 and Figure 2.23 it can be seen that the choice of the number of neurons in the hidden layer results accurate modelling. The accuracy achieved with more neurons improves slightly for the same amount of training time. Furthermore, the accuracy achieved for the same number of epochs is better when using more neurons. These results suggest a "more is better" guideline when choosing the number of neurons in the hidden layer. However, using too many neurons has several disadvantages. Firstly, more memory (in terms of weights) are required for a specific modelling requirement. Secondly, the time required for training using back-propagation (e.g. the Levenberg-Marquardt algorithm) is related to the number of weights in the neural network. Thirdly, too many weights could result in overtraining. Due to these factors, a neural network topology that greatly reduces the amount of memory and training time will be discussed.

From the results above, it appears that there are two requirements for the neural network topology. These requirements are a sufficient number of time delays and a layer with a sufficient number of neurons. However, the hidden layer (for non-linear mapping) need not be directly coupled with the input layer. By adding another hidden layer between the input and the layer for nonlinear mapping, an intermediate remapping can be achieved. Such a neural network topology is shown in Figure 2.24. If the first hidden layer has much less neurons than the second layer but is sufficiently large to support the mapping requirement, two advantages are achieved. Firstly, the total amount of memory (in terms of weights) are greatly reduced. Secondly, because of the reduction in the weight matrix size, training time is reduced.

The two hidden layer network can therefore have the same functionality as the single hidden layer network. It is also advantageous if the first hidden layer has less neurons than the second hidden layer does. In order to test whether this is true, the same data sets that were used in the previous section were used to train the two hidden layer neural network. The same number of neurons was used in the second layer, with only 5 neurons in the first layer. The training results are given in Figure 2.25.

For the neural network of Figure 2.13 (ignoring biases) the number of floating point weights $M_3 = 2430$ with 30 neurons in the hidden layer, and for the neural network of Figure 2.24 (ignoring biases) the number of floating point weights $M_4 = 580$ with 30 neurons in the second hidden layer.

When comparing Figure 2.18 and Figure 2.25 it can be observed that the accuracy that

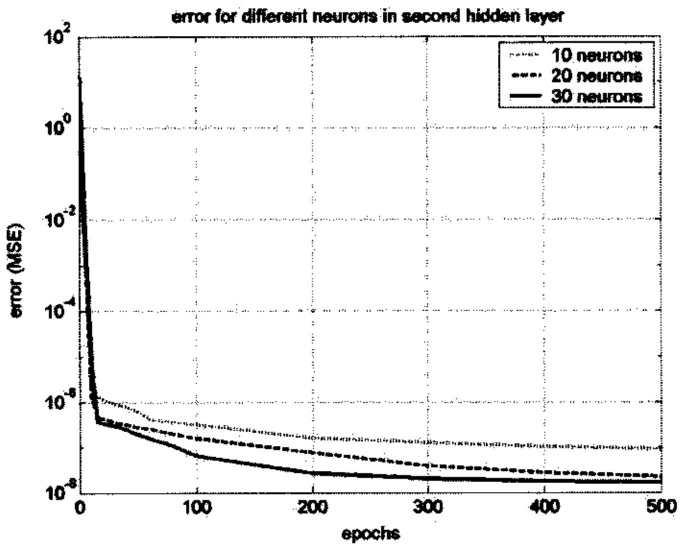

Figure 2.25: *Training results for 10, 20 and 30 neurons in the second hidden layer*

was achieved with the two topologies for the same training data are close. The training time was limited to 10000*s* and 500 epochs. It can be seen that a considerable saving in training time was achieved by using an intermediate hidden layer.

2.6 Multi-variable inference measurements

In the previous sections neural network topologies were derived for the type of relationship (e.g. non-linearity) that exists between the known variable and the variable to be inferred. These neural networks were limited to single input/single output systems. In this section the neural networks required for accurate inference of multiple variables in systems using multiple input variables are derived. Two types of inference problems will be considered: single input, dual output and dual input, single output variable inference.

2.6.1 Single input/dual output

In this section the network topology requirements for the inference of two variables from a single variable are derived. Since there are non-linearities, the system is dynamic. There are two outputs. The neural network topology in Figure 2.26 will be capable of inferring the variables with the correct number of time delays and neurons in the hidden layer. The required number of time delays was established by inspecting both output responses due to a step change by means of the method shown in a previous section on non-linear dynamic systems. The largest number of time delays required for inference of either outputs were chosen. Within $0.3s$, both the outputs decayed to an acceptable small value to ensure accurate inference, therefore the number of time delays were chosen as $TD = 60$.

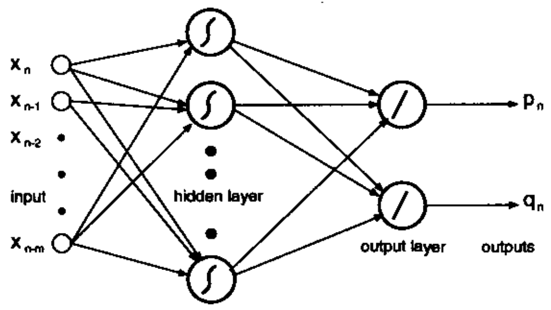

Figure 2.26: *Dual output time-delayed feed-forward neural network with a hidden layer*

The hidden layer is responsible for the mapping of the non-linearities of a system. In order to establish the number of neurons required for the inference of two variables compared to the inference of only variable, the network in Figure 2.26 was trained for two targets with different number of neurons. Two networks such, as in Figure 2.13, were trained for both output targets with a different number of neurons.

For the simulation of data a fourth order system (Figure 2.27) with several non-linearities was used. By choosing the state variables $x_1 = i_2$, $x_2 = e_3$, $x_3 = i_6$ and $x_4 = e_2$, the state space equations for this system was obtained (Equation 2.25).

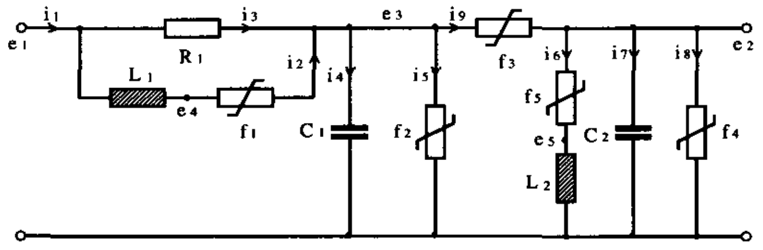

Figure 2.27: *Fourth order non-linear system.*

$$\begin{aligned} \dot{x}_1 &= \tfrac{1}{L_1}\left[e_1 - f_1(x_1) - x_2\right] \\ \dot{x}_2 &= \tfrac{1}{C_1}\left[\tfrac{e_1 - x_2}{R_1} + x_1 - f_2(x_2) - f_3(x_2 - x_4)\right] \\ \dot{x}_3 &= \tfrac{1}{L_2}\left[x_4 - f_5(x_3)\right] \\ \dot{x}_4 &= \tfrac{1}{C_2}\left[f_3(x_2 - x_4) - x_3 - f_4(x_4)\right] \end{aligned} \tag{2.25}$$

For this system, $C_1 = 0.07F$, $C_2 = 0.03F$, $L_1 = 0.07H$, $L_2 = 0.03H$ and $R_1 = 10\Omega$. The non-linear elements' characteristics are $e_4 - e_3 = f_1(i_2) = K_{1a}\tan(K_{1b}i_2)$ with $K_{1a} = 0.8V$, $K_{1b} = 1.3rad/A$, $i_5 = f_2(e_3) = K_{2a}\tan(K_{2b}e_3)$ with $K_{2a} = 0.2A$, $K_{2b} = 2.0rad/V$, $i_9 = f_3(e_3 - e_2) = K_{3a}\tan(K_{3b}(e_3 - e_2))$ with $K_{3a} = 0.2A$, $K_{3b} = 4.0rad/V$, $i_8 = f_4(e_2) = K_{4a}\tan(K_{4b}(e_2))$ with $K_{4a} = 0.1A$, $K_{3b} = 4.0rad/V$, and $e_2 - e_5 = f_5(i_6) = K_{5a}\tan(K_{5b}i_6)$ with $K_{5a} = 0.1V$, $K_{5b} = 5.0rad/A$.

For the dual output target with a single neural network, the input to the neural network is e_1, while the outputs are e_5 and e_2. For comparison, the same targets were used when training two separate networks with only one output target each. The training results for the three neural networks are given in Figure 2.28. The neural network output errors for the three neural networks are expressed in *normalised* mean square error (MSE). Figure 2.28 shows that the number of neurons required for a specific error (irrespective of whether one or two outputs are inferred) are more or less equal. For example, with 15 neurons in the hidden layer, an error of $MSE \approx 10^{-7}$ was achieved for any of the three trained neural networks.

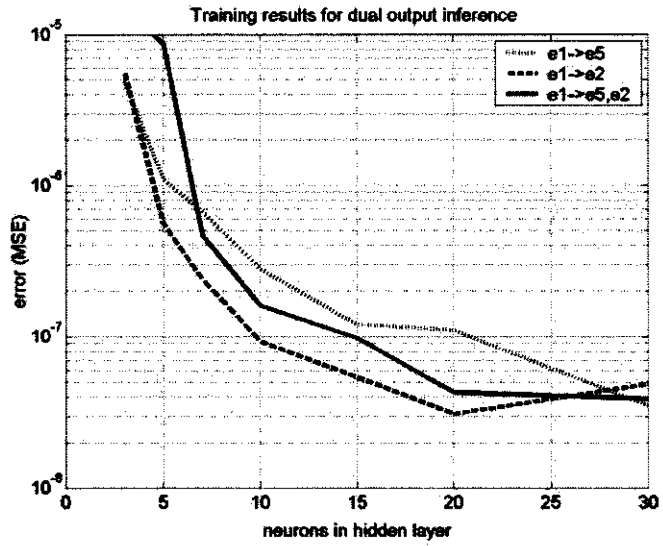

Figure 2.28: *Training results for the inference of two variables with a single input variable*

Ignoring biases, 915 weights were required for a single output inference. For dual output inference, 930 weights were required. The number of weights for these network topologies are mostly determined by the number of input weights. The output layer weights do not contribute significantly to the total number of weights. In this system where the outputs are related, a single neural network uses about half the number of weights for the same capability.

2.6.2 Dual input/single output

For a non-linear dynamic system with two independent directly observable variables from which a single output must be inferred, the neural network topology shown in Figure 2.29 meets the topological requirements with the correct number of time delays and neurons in the hidden layer. The required time delays were established by inspecting the output response decay due to step changes occurring after transients, using the method shown in the previous section non-linear dynamic systems, with both inputs changing. The number of time delays was chosen according to the slowest decay observed at the output due to any step input changes. Within 0.25s the output decayed to an acceptably small value to ensure accurate inference, therefore the number of time delays were chosen as $TD = 50$ with a time step of 0.005s.

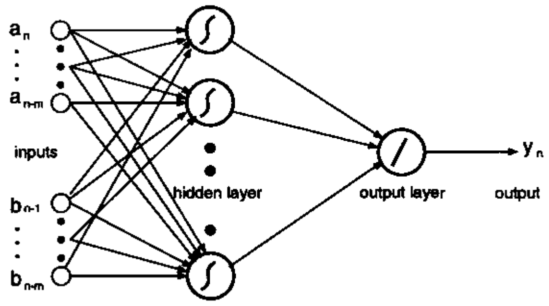

Figure 2.29: *Time-delayed feed-forward neural network with a hidden layer and two outputs*

For the simulation of the data sets, Figure 2.27, a third order system with several non-linearities, was used. By choosing the state variables $x_1 = i_2$, $x_2 = i_4$ and $x_3 = e_2$, the state space equations for this system were obtained (Equation 2.26).

$$\begin{aligned}
\dot{x}_1 &= \tfrac{1}{L_1}\left[e_1 - f_1(x_1) - x_3\right] \\
\dot{x}_2 &= \tfrac{1}{L_2}\left[e_3 - f_2(x_2) - x_3\right] \\
\dot{x}_3 &= \tfrac{1}{C_1}\left[\tfrac{e_1 - x_3}{R_1} + x_1 + \tfrac{e_3 - x_3}{R_2} + x_2 - f_3(x_3)\right]
\end{aligned} \qquad (2.26)$$

For this system, $C_1 = 0.05F$, $L_1 = 0.03H$, $L_2 = 0.07H$, $R_1 = 5\Omega$ and $R_2 = 5\Omega$. The non-

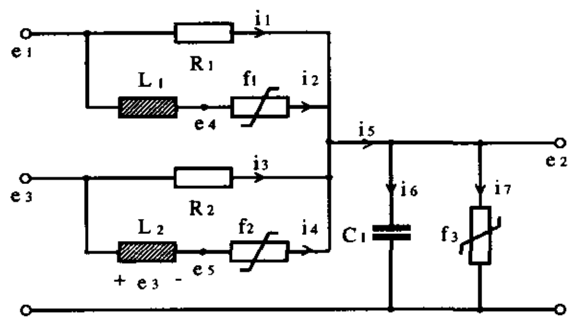

Figure 2.30: *Third order non-linear system*

linear elements' characteristics are $e_4 - e_2 = f_1(i_2) = K_{1a}\tan(K_{1b}i_2)$ with $K_{1a} = 0.3V$, $K_{1b} = 4.0rad/A$, $e_5 - e_2 = f_2(i_4) = K_{2a}\tan(K_{2b}i_4)$ with $K_{2a} = 0.2V$, $K_{2b} = 3.0rad/A$ and $i_7 = f_3(e_2) = K_{3a}\tan(K_{3b}(e_2))$ with $K_{3a} = 0.2A$, $K_{3b} = 3.0rad/V$.

Table 2.2 gives the training results for the inference of a single variable e_2 with two input variables, compared to the training results with only one input excited with either e_1 or e_3 (with the other input at a constant voltage of $0.8V$). A neural network with two inputs, 60 time delays in the input layer and 15 neurons in the hidden layer was used.

Table 2.2: *Training results for the inference of a single variable with two input variables*

network inputs	MSE	maximum error
e_1, e_3	$1.3 * 10^{-6}$	≤ 0.01
e_1, $e_3 = 0.8V$	$3.6 * 10^{-7}$	≤ 0.002
$e_1 = 0.8V$, e_3	$3.4 * 10^{-8}$	≤ 0.001

With only one input variable changing (the other input at a constant value), the values of the variables at the non-linear elements are less than or equal to the values reached with both independant inputs varying. A larger or equal error with both inputs varying as compared to only one input varying is thus expected. The results in Table 2.2 confirm this expectation.

Ignoring biases, 1515 weights are used for the inference of a single variable on a system with two independent input variables. With one input variable only 765 weights were used. At least twice as many weights are therefore required to infer a variable from two input variables than inferring a variable from only one variable.

2.7 Discussion

This chapter has shown which neural network topologies are required for a specific inference problem. For non-linear mapping, a single hidden layer with non-linear activation function are required. Dynamic mapping requires input time delays.

The number of neurons in the hidden layer depends on the characteristics of the non-linear elements to be mapped. If some information of the non-linear elements' characteristics were known, some indication of the number of neurons required for accurate inference can be found. If no information on the non-linear elements are available, a few training trial runs with different numbers of neurons in the hidden layer can be used to establish the required number of neurons that will result in an acceptable small inference error.

Integration (cumulative summation) and differentiation (difference) perform as was theoretically predicted. A transfer function with a pole at the origin is therefore difficult to model when using a neural network. On the other hand, a transfer function with a zero at the origin can be performed accurately by using a neural network.

By using linear dynamic systems it was established that the number of time delays required for inference of a variable of system are related to the time constant of the characteristic equation. For non-linear systems, without knowledge of the time constant, the number time delays required can be calculated from the time the system output response decays to a sufficiently small value with a step input applied to the system.

By introducing a second hidden layer, the neural network memory requirements for the same modelling capability can be reduced significantly with an associated reduction in training and simulation time.

For the inference of two variables from a single variable on the same system, a single neural network resulted in accurate inference of two variables requiring approximately the same number of weights required for two separate neural networks.

For the inference of a variable from two independent variables, the inference error when using a neural network is larger than the inference from only one variable. This is due to the higher level of excitation of the hidden non-linearities. Approximately twice the number of weights are required for inference from two variables than from one variable.

In this chapter a general method for the neural network topology design (with minimal trial and error) for accurate inference measurements was developed. The following procedure summarises these steps:

1. Determine whether the mapping requirement is non-linearly static, linearly dynamic or non-linearly dynamic. Choose the simplest network topology to satisfy the mapping requirement as shown earlier in this chapter. For multivariable mapping, add the appropriate number of inputs or outputs as set out in the problem statement.
2. For dynamic mapping, determine the number of time delays by inspecting the output response decay with a step input.
3. For non-linear mapping, obtain the number of neurons that would satisfy the inference error specification by using a few training trial runs with different numbers of neurons in the hidden layer.
4. Test and validate the neural network, using test and validation sets.

The next chapter will show that, not only is the correct neural network topology necessary for accurate inference, but that the training set inputs must have certain characteristics compared to the test or validation set inputs.

Chapter 3

Training, test and validation sets

This chapter provides a critical evaluation of the validity of the data sets and characteristics of training, test and validation data sets. Two important issues will be addressed in this chapter, namely:

A. Are the data sets that are created by the simulation valid?

B. What are the required characteristics of training sets, as compared to test and validation excitation waveforms, in order to ensure accurate inference?

3.1 Data simulation

Training, test and validation data sets are required for successful implementation of artificial neural networks. Data sets are the applicable system variables with excitation waveforms as input. Figure 3.1 illustrates the data simulation process.

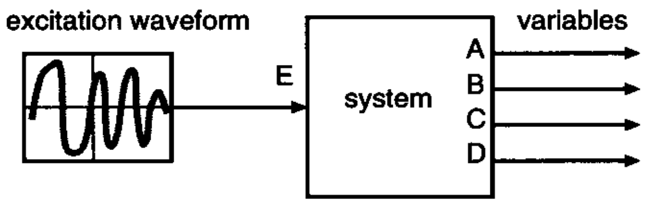

Figure 3.1: *Block diagram of the data simulation process*

In order to generate these training and test sets, MATLAB's ODE solver `ode45` was used to find the solution for the system variables from the state space equations of the systems. **Appendix A: Software implementation** discusses MATLAB's ODE solvers. In most cases the state equations were based on electrical circuits, although the results are equally applicable to mechanical or chemical systems [11], as will be shown in **Chapter 4: PBMR neural network inference measurements**. The variables in the systems that were not solved by the ODE solver were determined explicitly.

A time frame of $10s$ was chosen for all the experiments. The maximum frequency of the excitation waveform applied to the system under investigation was chosen as $f_{max} \approx 10Hz$. The critical sampling rate for the excitation waveforms (with a maximum frequency of $10Hz$) requires a sampling rate of only $20Hz$. However, the non-linear components result in distortion, subsequently harmonics are generated in the system. These harmonics are reflected at any observable point in the system. Therefore it is required to sample any observation at a sufficiently high frequency in order to capture the system characteristics.

Figure 3.2 shows the input and output spectra of a linear chirp signal input e_1 applied to the system under investigation. Because of the non-linearities in the system, harmonics are generated. This can be observed from the higher frequencies in the spectrum of the output e_2.

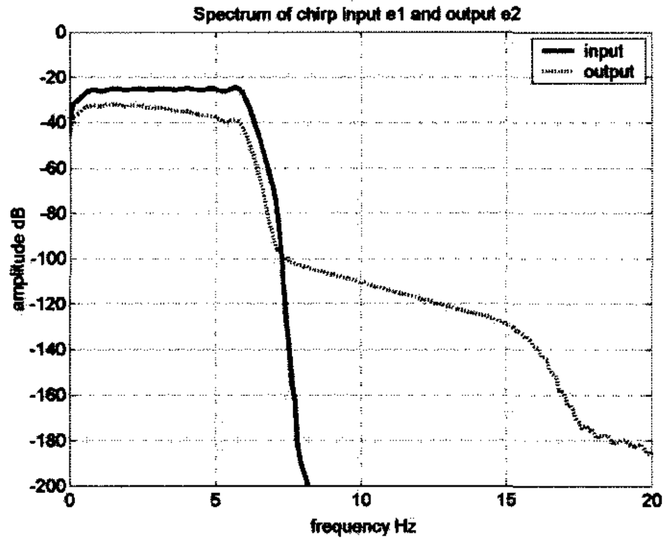

Figure 3.2: *Input and output spectra with linear chirp as input*

In order to reach an error goal, the ODE solver requires a temporal grid that is sufficiently fine. The ODE solver can adjust the temporal grid locally to reach the error goal, or a fixed temporal grid could be set. A fixed temporal grid was chosen since the time-delayed neural network assumes a fixed temporal grid. In order to capture the signals with harmonics and to ensure sufficiently fine temporal resolution for the ODE solver, a sampling rate of $200Hz$ was chosen. This resulted in a time step of $\Delta T = 0.005s$ and 2001 points in each data set. This temporal resolution provided data sets that are accurate enough to lead to sensible experimental results. In order to minimise the effect of absolute scale, the parameters in all the systems were chosen so that the variables are of the same order of magnitude and the amplitude A fall in the range $-1 < A < 1$ so that it is not required to normalise the amplitudes [19, p395].

In the section on data set validation, analysis and inspection methods by means of which data sets can be validated, are presented. These methods assist the modeller to evaluate the accuracy of the data. Inaccurate data will result in inaccurate neural network modelling, irrespective of the neural network topology or training algorithm.

In the subsequent section on excitation waveform characteristics, input waveforms' characteristics are compared. By using the trained neural network output error with test waveform input as measure, the characteristics, by means which the the training waveform input and test waveform input are compared, will enable us to predict whether a training set would result in accurate inference for a test input waveform. By using these input waveforms' characteristics comparison, it will be shown, how to ensure, without neural network simulation with a test waveform input, whether the trained neural will accurately infer a variable for such a test waveform input or not.

To address these issues, the data of the second order non-linear system shown in Figure 3.3 was used. The neural network topology that was used is a time-delayed single hidden layer feed-forward network as shown in Figure 3.4. It has 50 time delays and 30 neurons in the hidden layer. Where used, it is assumed that the neural network topology and training algorithm could infer variables of the system to within a required specification. The issues concerning the design of a neural network topology for a specific system were addressed in **Chapter 2: Network topology and system characteristics**.

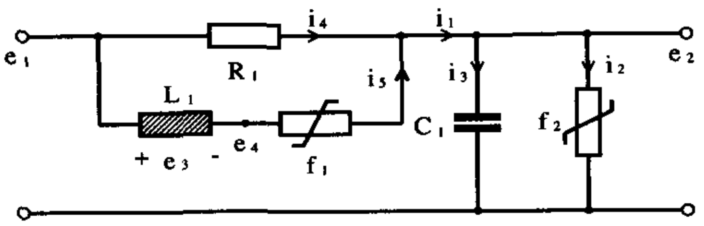

Figure 3.3: *Second order non-linear system*

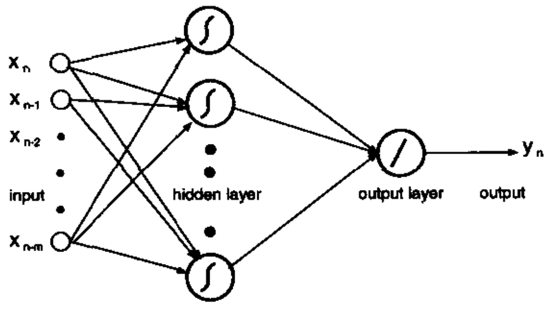

Figure 3.4: *Time-delayed feed-forward neural network with one hidden layer.*

3.2 Data set validation

In this section several tests will be conducted in order to verify the validity of data set simulation. Since all other experiments, results and conclusions rely on the simulated data, it is crucial to verify the accuracy of such data. Figure 3.5 shows the excitation waveform that was used for data simulation in order to validate the data set. Waveform i is an amplitude-modulated linear chirp signal given by Equation 3.1.

$$e_{1i} = 0.8 * \left[1 - e^{(t^2-100)}\right] * \left[\sin(\frac{\pi t}{20})\right] * \left[\sin(2.5 * t^2)\right] \tag{3.1}$$

3.2.1 Energy

For any closed system, the law on conservation of energy states that the energy supplied to a system must be equal to the energy dissipated plus stored energy. Thus, for the system in Figure 3.3, initially at rest, the energy supplied must be equal to the energy dissipated by the lossy elements, i.e. R_1, f_1 and f_2, plus energy stored by the inductor L_1 and capacitor C_1. Since amplitude of the excitation waveform is close to zero at $t = 10$ s, it is expected that the stored energy will be small. Any simulated data set should yield

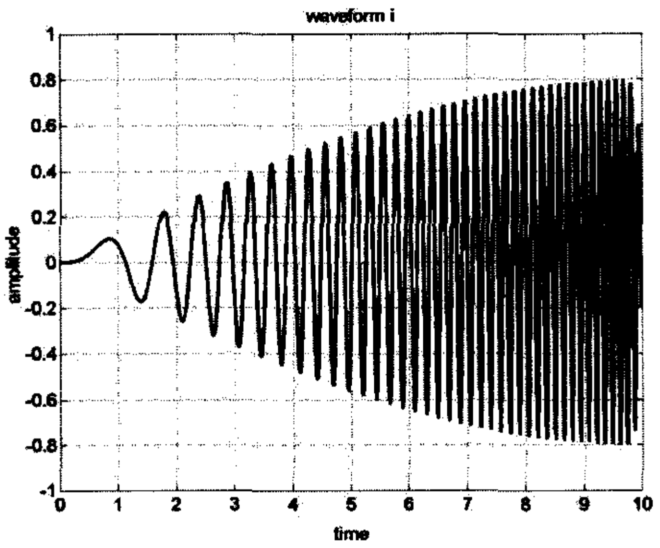

Figure 3.5: *Waveform i used for data set validation*

similar results.

The total energy supplied to the system E_{in} can be calculated from Equation 3.2 and the total energy (dissipated and stored) E_{out} can be calculated from Equation 3.3 with $nts = 2001$ and $\Delta T = 0.005s$. Within a small numerical error, the energy is preserved, while the energy in any storage element is indeed small, as could be expected.

$$\begin{aligned} E_{in} &= \int_0^{10} e_1(t) i_1(t) dt \\ E_{in} &\approx \sum_{n=1}^{n=nts} e_{1n} i_{1n} \Delta T \\ E_{in} &\approx 1.2028J \end{aligned} \tag{3.2}$$

$$\begin{aligned} E_{out} &= E_{R1} + E_{f1} + E_{f2} + E_{L1} + E_{C1} \\ E_{out} &= \int_0^{10} [e_1(t) - e_2(t)]\, i_4(t) dt + \int_0^{10} [e_4(t) - e_2(t)]\, i_5(t) dt + \int_0^{10} e_2(t) i_2(t) dt \\ &\quad + \frac{1}{2} L_1 i_5^2 |_{t=10} + \frac{1}{2} C_1 e_2^2 |_{t=10} \\ E_{out} &\approx \sum_{n=1}^{n=nts} [e_{1n} - e_{2n}]\, i_{4n} \Delta T + \sum_{n=1}^{n=nts} [e_{4n} - e_{2n}]\, i_{5n} \Delta T + \sum_{n=1}^{n=nts} e_{2n} i_{2n} \Delta T \\ &\quad + \frac{1}{2} L_1 i_5^2 |_{nts} + \frac{1}{2} C_1 e_2^2 |_{nts} \end{aligned} \tag{3.3}$$

$$\begin{aligned} E_{out} &\approx 0.6729 + 0.3683 + 0.1614 + 0.000062 + 0.000086J \\ E_{out} &\approx 1.2028J \end{aligned}$$

3.2.2 Dissipative elements

Static dissipative elements such as the resistor R_1 and the non-linear elements f_1 and f_2 should have an accurate V-I relationship for any applied waveforms used to generate a data set. Figure 3.6 shows the V-I relationships of the dissipative elements with waveform i as input for data set generation. Since these plots are lines, showing unique mapping of the V-I relationships, it can subsequently be concluded that these relationships have been computed accurately everywhere.

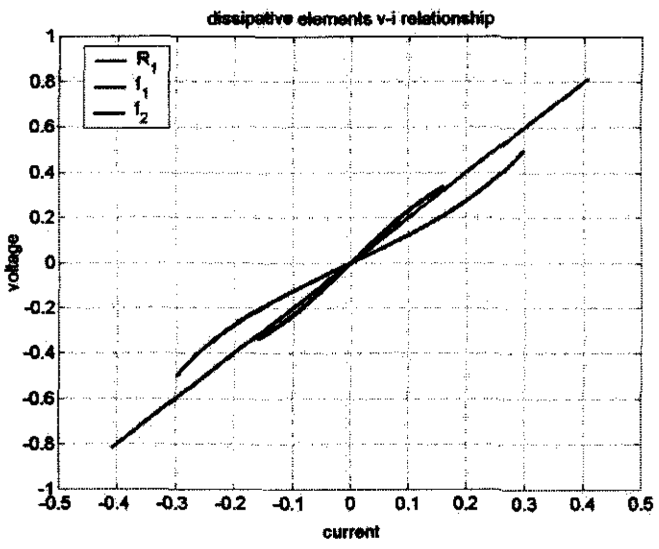

Figure 3.6: *Plot of* V-I *relationship of dissipative elements for excitation waveform*

3.2.3 Storage elements

The voltage over an inductor is given by $v(t) = L\frac{di(t)}{dt}$ and the current through a capacitor is given by $i(t) = C\frac{dv(t)}{dt}$. From these identities, approximations for the voltage $\hat{v}$ over the inductor L_1 and the current $\hat{i}$ through the capacitor C_1 can be written as Equation 3.4 and Equation 3.5 respectively. For accurate data, the plot of the estimate versus the generated data should approach a straight line. The plot of the estimate $\widehat{e_3}$ versus e_3 is shown in Figure 3.7, while the plot of the estimate $\widehat{i_3}$ versus i_3 is shown in Figure 3.8.

Since these plots are thin straight lines, it can be concluded that these relationships have been computed accurately everywhere.

$$\widehat{e_3}(n) \approx L_1 * \frac{i_5(n+1) - i_5(n-1)}{2 * \Delta T} \tag{3.4}$$

$$\widehat{i_3}(n) \approx C_1 * \frac{e_2(n+1) - e_2(n-1)}{2 * \Delta T} \tag{3.5}$$

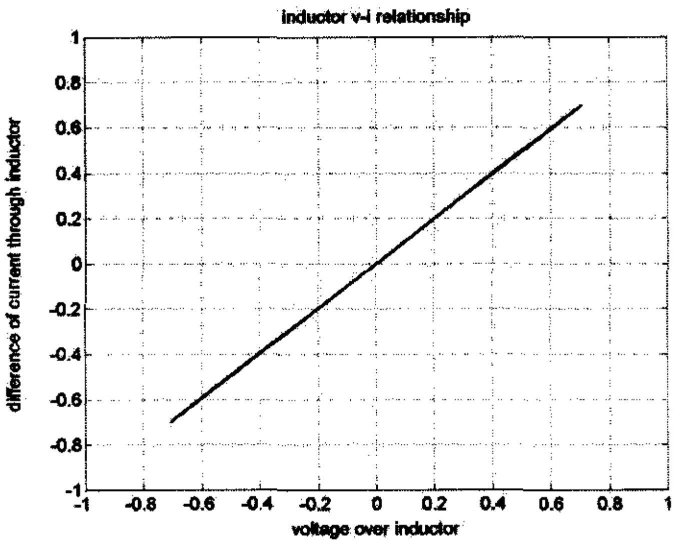

Figure 3.7: *Plot of $\widehat{e_3}$ versus e_3 for excitation waveform*

3.2.4 Noise

This section will determine the effect of noise added to the training set data on the training results. Three experiments with noise added to the training set data were conducted. From these experiments a relationship between the accuracy of the neural network inference with certain scenarios of training set data noise was determined.

In the three experimental scenarios, random noise is added to the training set (I) input, (II) output and (III) input and output. The training set data was simulated by using waveform i. The added pseudo random noise is of the form $noise = A * rand$ where $-1 < rand < 1$ and the amplitude A was varied over a range $0.0001 \leq A \leq 0.1$. The

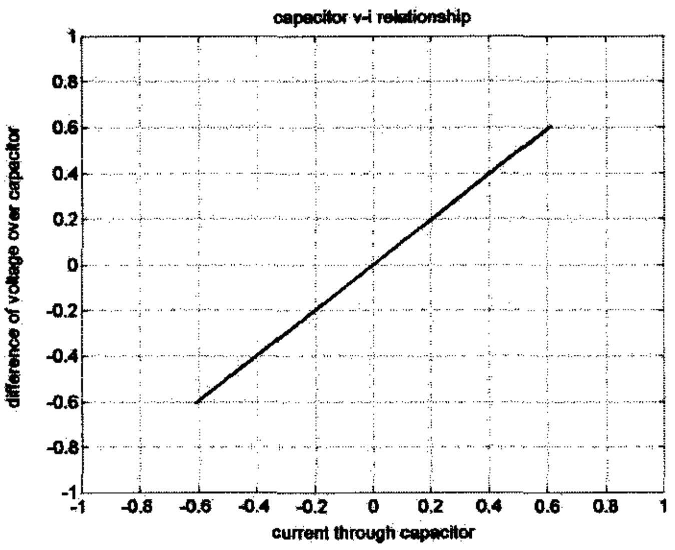

Figure 3.8: *Plot of* $\widehat{i_3}$ *versus* i_3 *for excitation waveform*

random function was implemented by using MATLAB's `rand` function as described in **Appendix A: Software implementation.**

The neural network was trained for each of these scenarios with different amplitudes of the added noise. The training errors achieved with these three scenarios are plotted in Figure 3.9. It can be seen that the training error increases more or less logarithmically with a linear increase in noise amplitude. Assume that the data generation simulation error is random with the same distribution as the added noise, the error of the simulated data can be estimated. Noise added to the input data e_1 indicates that a simulation error of less than ≈ 0.001 could result in a training mean square error of less than $MSE = 2 * 10^{-8}$. For reaching the same training error of less than $MSE = 2*10^{-8}$, the simulation error for the target data e_2 must be less than 0.0002. With $|e_1| \leq 0.8$ and $|e_2| \leq 0.5$, in percentage of maximum amplitude, a randomly distributed error with maximum amplitude of 0.125% added to the input vector, or a randomly distributed error with maximum amplitude of 0.04% added to the output vector, would result in a training error of less than $MSE = 2*10^{-8}$. This indicates that modelling with neural networks is more sensitive to the noise added to the output than to the input.

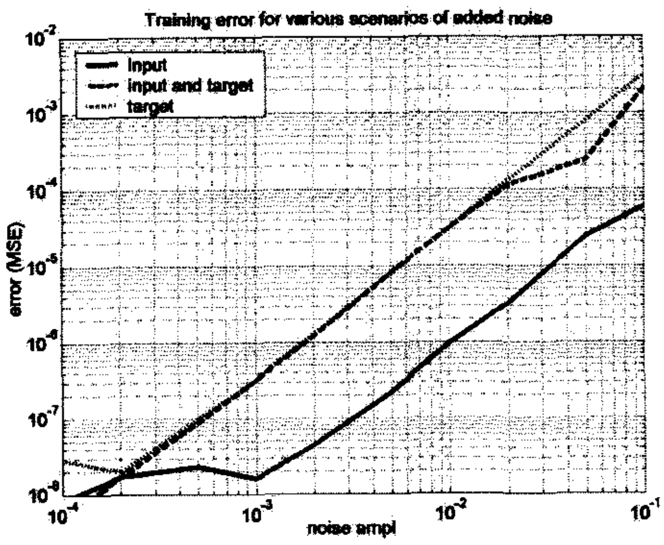

Figure 3.9: *Plot of training error with noise added to* (I) *input,* (II) *input and output,* (III) *input and output*

3.2.5 Finite bit resolution

The measurement data on a system is discrete, with a finite bit resolution. In order to establish the effect of such a finite bit resolution on the neural network inference accuracy, the training data (generated by using waveform i) was converted to sets of data with a finite resolution. With a full-scale input and output range of −1 1, Equation 3.6 and Equation 3.7 convert the training set data to sets of data with finite resolution of N bits.

$$\widehat{e_1} = \frac{\text{round}(1 + e_1)^{N-1}}{2^{N-1} - 1} \tag{3.6}$$

$$\widehat{e_2} = \frac{\text{round}(1 + e_2)^{N-1}}{2^{N-1} - 1} \tag{3.7}$$

For all N bit resolution data sets with $4 \leq N \leq 16$, the neural network was trained and the training error (MSE) tabulated and plotted. Figure 3.10 shows the mean square error (MSE) versus the bit resolution N. It can be seen that the training error decreases as the bit resolution increases, thus more accurate inference could be achieved by using data with a high resolution.

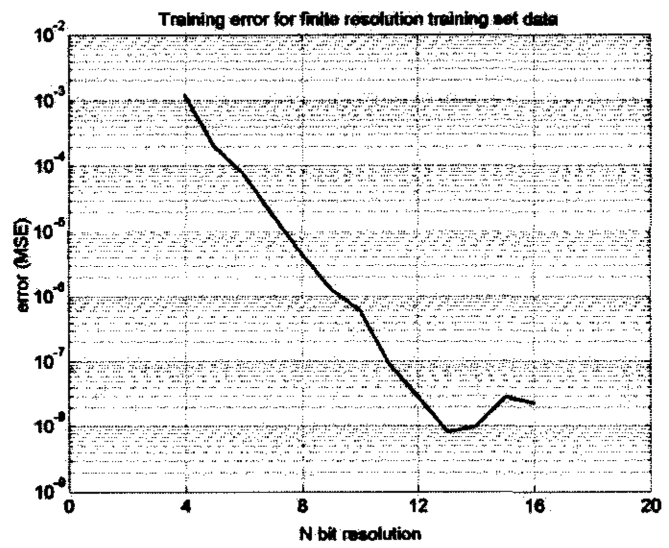

Figure 3.10: *Plot of training error, using input and output training data with finite bit resolution*

3.3 Excitation waveform characteristics

In this thesis, training of the neural network was done with a single set of training input/output data. For accurate inference, such an input/output data set should contain all the information to train the neural network within the required specifications.

Several types of excitation waveforms for obtaining input/output data for system characterisation or neural network training have been used. Commonly used excitation waveforms are waveforms with superimposed harmonics [36], multiple-frequency waveforms [37][19], band limited random waveforms [19] and chirp waveforms [16][38]. For this thesis, variations of chirp, multiple-frequency and step waveforms will be used as inputs to generate the training and test sets. Specifications for the range of variable inference measurements can be expressed as test or validation sets. Test sets will refer to any set that is used for testing the trained neural network. It could be a validation set or a specification expressed as a test set. The comparison of training sets and test sets in this thesis is therefore directly applicable to the comparison of training and validation sets. From this viewpoint, a validation set is just another test set, and both has to fall within the inference specification.

Trained neural networks perform well when doing interpolation, but are not good when performing extrapolation. For this reason it is important to train neural networks with a training set that spans a greater mapping space than a test or validation set. It is expected that the validation set would represent the mapping requirement for the neural network in operation. This section will develop comparitive measures by which training input waveforms' characteristics can be evaluated to predict inference accuracy for any given test or validation input waveforms. A single training set consisting of input/output data is used for training. The input waveform must subsequently excite the system in such a way as to generate a sufficient number of training data examples in a neighbourhood where the required mapping is specified. In order to compare the sufficiency of a training input waveform to a test input waveform for the purpose of predicting whether accurate inference measurements will be achieved, the following hypotheses are presented:

1. *Test or validation input waveforms should not cause variables to have a greater amplitude than the neural network has been trained for.* Black box modelling, such as neural network modelling, could hide such occurrences. Therefore precautions should be taken if an internal parameterised model (that can be inspected) were not available. To ensure that the mapping capabilities of the neural network in terms of the non-linear components are larger than the requirement for the test set, a relationship between the training waveform and test waveform that would result in successful modelling is desired. A simple solution is to ensure that the amplitude A of the training set input is larger than the test set input, thus

   $$A_{train(min)} \leq A_{test} \leq A_{train(max)}$$

   Systems with complex pole pairs, excited with waveforms close to or at the resonant frequency, will experience large variable amplitudes, reaching a maximum when sustained long enough. The smaller the damping factor ζ, the larger the amplitudes reached for a specific input amplitude. For non-linear systems, the resonant frequency changes with the absolute amplitude of the excitation waveform, and therefore the resonant frequency cannot be determined accurately. The resonant frequency is approximately $f_r \approx 5Hz$ and can be estimated from the $-3dB$ cut-off frequency in the spectrum of the output of the system with an input with a flat frequency spectrum such as in Figure 3.2 or from the overshoot with a step input applied. It was shown in the previous chapter that the system under consideration has an overshoot with a step input applied, and therefore the system has complex pole pairs with a damping factor $0 < \zeta < 1$. Since the overshoot is small $\leq 1\%$, the damping factor ζ is just less than 1, and therefore it is not expected that the variable amplitudes would increase significantly if an input waveform has a frequency at the resonance frequency of the system, sustained for a prolonged time.

2. *The training input waveform should span a wider dynamic range than the test input (or validation input), since the trained neural network cannot extrapolate.* In terms of frequency, this implies that

$$f_{train(min)} \leq f_{test} \leq f_{train(max)}$$

Since the frequency of a waveform is related to the rate of change in the time domain, this implies that the rate of change of the test input waveform should be within the rate of change of the training input waveform, or

$$|\Delta A_{train}|_{min} \leq |\Delta A_{test}| \leq |\Delta A_{train}|_{max}$$

Should the neural network be seen as a pattern classifier, where the input consists of the current value and time-delayed copies of the previous values, the patterns presented in the time span of the time delays range from slow-varying patterns to fast-varying patterns. The slowest varying patterns are closest to constant values, ranging from maximum to minimum amplitudes. For the neural network to learn the relationship between the input and output for a constant input, the training input waveform should have constant parts at minimum and maximum amplitudes for at least the time span of the time delays. The highest frequency part of the input waveform produces the fastest varying pattern.

3. *For successful neural network training between the dynamic range limits, the training input waveform should excite the system sufficiently dense.* In terms of frequency components, this means that the training input waveform should have frequency components at regularly spaced intervals, or at least have frequency components with larger amplitudes than that of the test input waveform. The frequency spectrum of the training set input waveform should therefore be sufficiently dense to

enable successful modelling. In the time domain, this implies that the training set input waveform should have different rates of change within the amplitude range. The different rates of change within the amplitude range must be sufficiently dense, in other words, there should not be large domains where there is no mapping of a rate of change at a specific amplitude.

In order to test these hypotheses, experiments were conducted where the input waveforms' characteristics were compared and the output error of the trained neural network was used as measure. The experiments consist of training the neural networks with selected excitation input waveforms and comparing the test input waveform's characteristics with the training excitation waveform's characteristics. Frequency and time domain analyses are performed to quantify the waveforms' characteristics. Using the output error (either the mean square error (MSE) or error in time domain) as inference accuracy measure, a comparison between the characteristics of the training excitation waveform and test excitation waveform is established so that it can be predicted whether inference would be accurate or not, without actual network simulation using test sets.

3.3.1 Excitation waveforms

Variations of the three basic types of excitation waveforms, namely chirp, multi-frequency and step waveforms, were used in the experiments. These three types of waveforms originated from chirp signals used by bats and for radar, waveforms with line spectra that can be used to determine system frequency characteristics and step inputs to determine the step response of systems. One of the waveforms and the resultant output will be used to train the neural network in the experiments. The training input waveform's characteristics will be compared to the characteristics of the other waveforms, using either frequency domain or time domain characteristics.

The numerical values were chosen so that they fall within a range, and can scaled to suit the application as will be shown for the pebble bed micro model (PBMM). These numerical values were chosen for the convenience of having parameter values that can be expressed simply.

Excitation waveforms used for training

Waveforms used for training are shown in Figure 3.11.

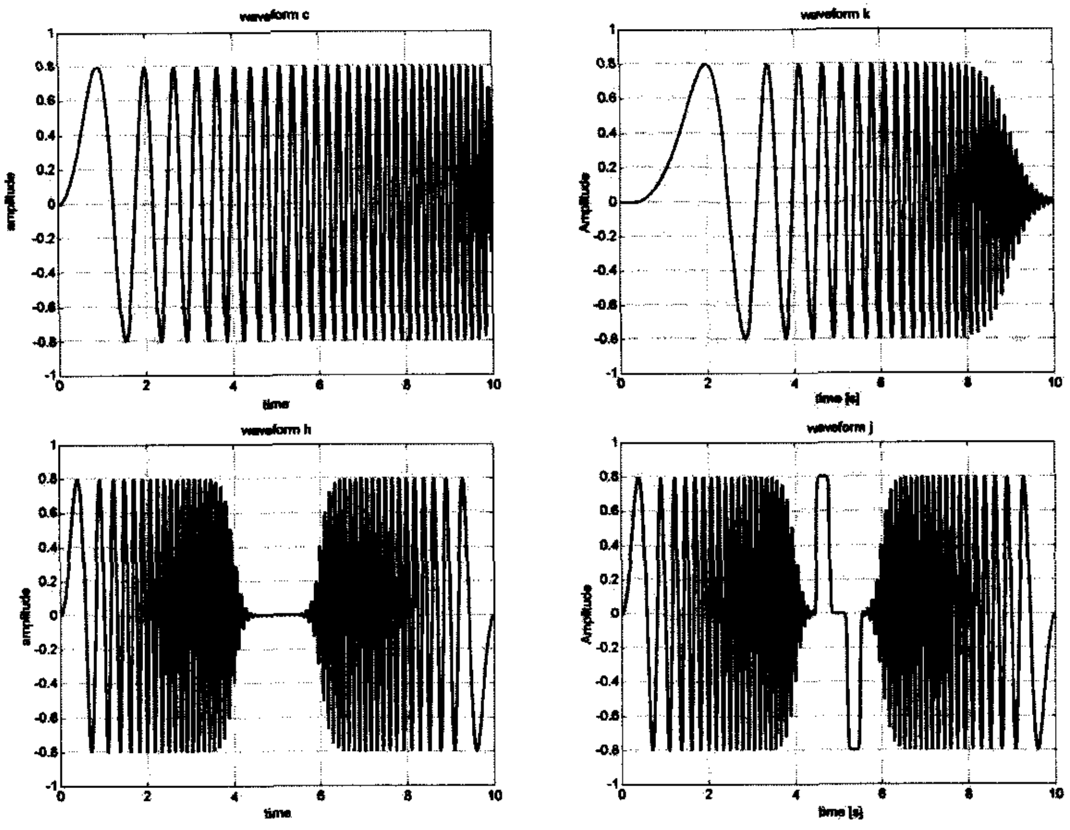

Figure 3.11: *Time plot of waveforms c, k, h, and j*

Waveform c is a linear chirp signal as formulated in Equation 3.8. At high frequencies, the amplitude is modulated to reduce any unwanted end effects.

$$e_{1c} = 0.8 * \left[1 - e^{(t^2-100)}\right] \sin(2t^2) \tag{3.8}$$

Waveform k is a quadratic chirp signal, of which the amplitude is modulated at high frequencies, as given by Equation 3.9

$$e_{1k} = 0.2 * [1 + \tanh(5 * (t - 0.5)) * (1 - \tanh(2 * (t - 9.0))] * \sin(0.2t^3) \tag{3.9}$$

Waveform h is a frequency-modulated signal where the modulation signal is chosen in such a way that the frequency increases up to a maximum and then decreases again. The signal is amplitude modulated at the high-frequency part of the waveform. Equation 3.10 gives the formulation for waveform h.

$$e_{1h} = 0.4 * [2 + \tanh(5 * (t - 6.0)) - \tanh(5 * (t - 4.0))] * \sin[50\pi(1 - \cos(0.1\pi t))] \quad (3.10)$$

Waveform j, given by Equation 3.11, is a modification of waveform h, with the addition of a constant part of maximum absolute amplitude sustained for at least as long as the input time delay of a neural network required for accurate inference.

$$\begin{aligned} e_{1j} &= 0.4 * ([2 + \tanh(5 * (t - 6.0)) - \tanh(5 * (t - 4.0))] * \\ &\quad \sin[50\pi(1 - \cos(0.1\pi t))] + \tanh(50 * (t - 4.5)) - \\ &\quad \tanh(50 * (t - 4.8)) - \tanh(50 * (t - 5.2)) + \tanh(50 * (t - 5.5))) \quad (3.11) \end{aligned}$$

Excitation waveforms used for testing

Waveforms used for testing the trained neural network is given in this section.

Waveform p and q, given by Equation 3.12 and Equation 3.13 are amplitude-modulated linear chirp signals. For waveform p the low-frequency part has the highest amplitude, and for waveform q the high-frequency part has the highest amplitude. These waveforms are shown in Figure 3.12.

$$e_{1p} = 0.8 * \left[e^{\frac{t}{5}}\right] \sin(4t^2) \quad (3.12)$$

$$e_{1q} = 0.8 * \left[e^{\frac{t-10}{5}}\right] \sin(4t^2) \quad (3.13)$$

Waveform a (formulated by Equation 3.14) is a superposition of amplitude-modulated signals, resulting in a line spectrum with a minimum frequency of $0.09Hz$ and a maximum

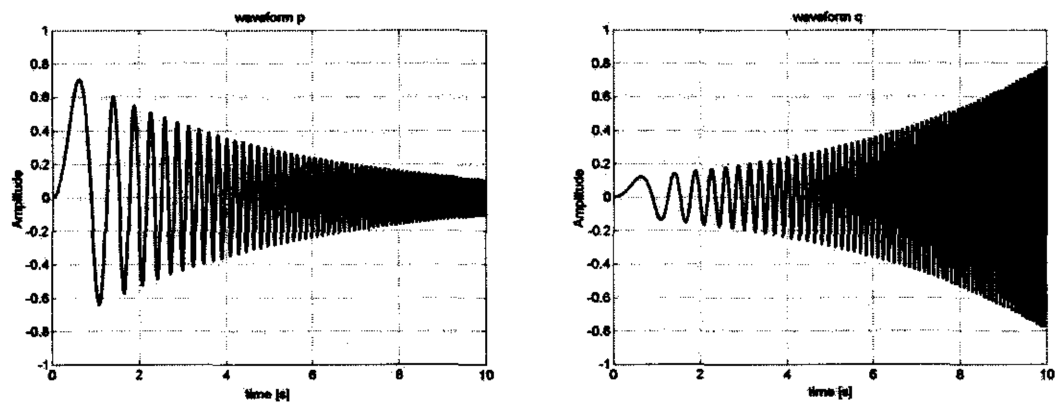

Figure 3.12: *Time plot of waveforms p and q*

frequency of $7.7Hz$. Waveform f (formulated by Equation 3.15) is an amplitude-modulated signal with a fixed carrier frequency of $5Hz$. This frequency was chosen because the resonant frequency of the system under investigation (see Figure 3.3) has a resonant frequency of approximately $5Hz$. Figure 3.13 illustrates the plots of these waveforms.

$$e_{1a} = \sum k_n 0.5\left[\cos(0.1n2\pi ft - \pi) - 1\right]\sin(2n\pi ft) \tag{3.14}$$

with $n = 1, 3, 5$ and 7, $f = 1Hz$, $k_1 = 0.5$, $k_3 = 0.2$, $k_5 = 0.2$, $k_7 = 0.1$.

$$e_{1f} = 0.8 * \left[\sin(\frac{\pi t}{20})\right]\sin(10\pi t) \tag{3.15}$$

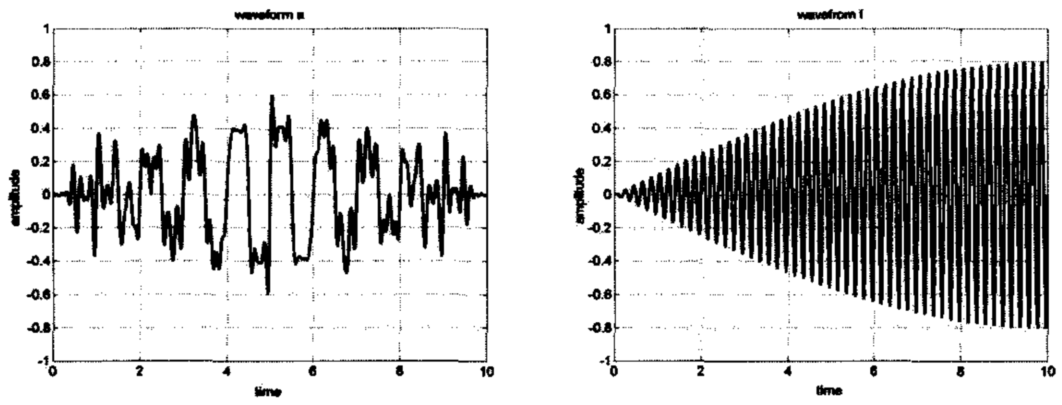

Figure 3.13: *Time plot of waveforms a and f*

The rate of change of a step signal (and the number of frequency components) are infinite in the ideal case. To ensure that the signal is well-presented in the time domain (and frequency domain), the rate of change of a step signal must be limited. In order to accomplish this limited rate of change, pseudo step signals, closely representing true step signals, were constructed, using the hyperbolic tangent function tanh. Waveform e is close to a step signal and is given by Equation 3.16. Waveform m is a multistep signal given by Equation 3.17 Waveform e and m are plotted in Figure 3.14.

$$e_{1e} = 0.4 * [1 + \tanh(50(t-5))] \tag{3.16}$$

$$\begin{aligned} e_{1m} = {} & 0.1 * [\tanh(50(t-1)) - 2\tanh(50(t-2)) + 3.5\tanh(50(t-3)) - \\ & 5\tanh(50(t-4)) + 6.5\tanh(50(t-5)) - 8\tanh(50(t-7)) + \\ & 4\tanh(50(t-9))] \end{aligned} \tag{3.17}$$

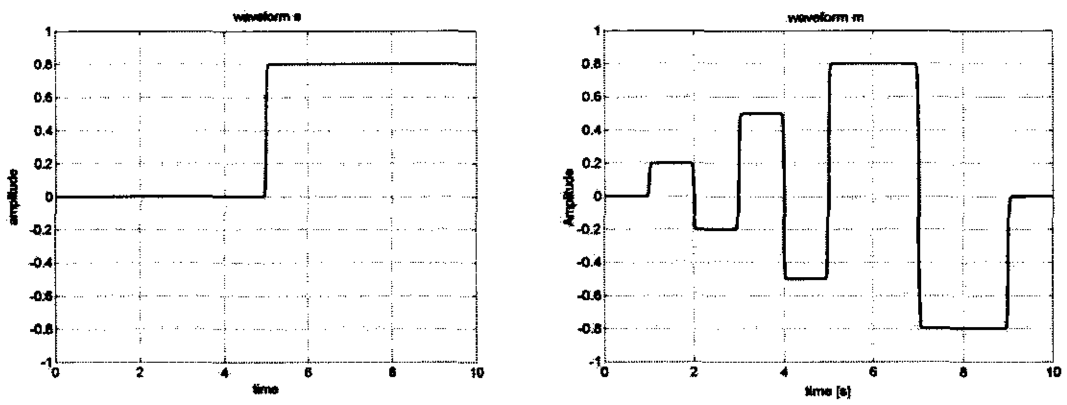

Figure 3.14: *Time plot of waveforms e and m*

3.3.2 Frequency domain analysis

In this section, an attempt will be made to quantify a waveform's characteristics for comparison based on the frequency spectrum. For an estimate of the spectrum of a waveform, MATLAB's `pmtm` function was used as described in **Appendix A: Software implementation**. In a previous section, three characteristics for the comparison of training and test input waveforms were hypothesised, namely, (1) *amplitude*, (2) *dynamic range* and (3) *density within the dynamic range.*

The frequency spectrum gives the frequency components with high confidence, but the amplitude of the spectrum at a specific frequency does not relate directly to the actual amplitude of the waveform, since the amplitude depicts the summation of the components of a specific frequency. When comparing waveform c (a chirp waveform) with a waveform with specific frequency components (such as waveform a or f) that are sustained for a prolonged period with equal maximum amplitude, the amplitude of the spectra will suggest that the amplitude of the waveform with specific frequency components is larger than the amplitude of the chirp waveform. However, this is not true. Subsequently, the spectra cannot be used to compare the amplitudes of waveforms. This can be verified by observing the spectra of waveforms c, a, f and k (given in Figure 3.15).

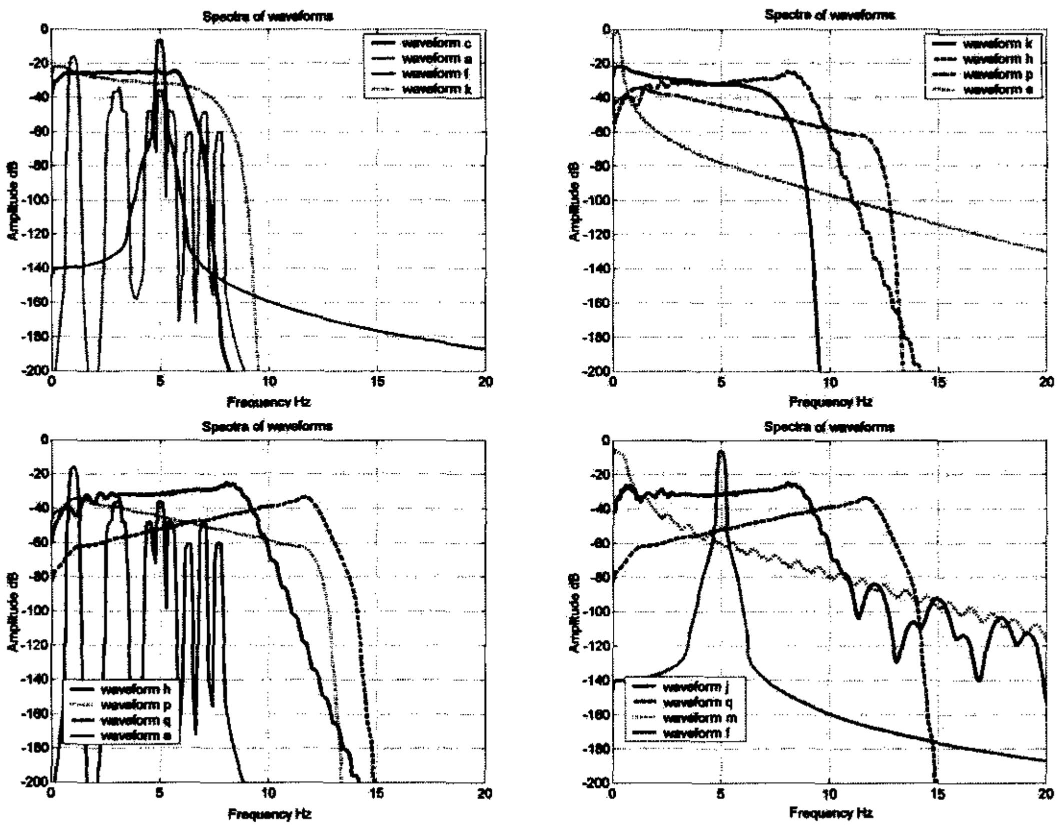

Figure 3.15: *Spectra of input waveforms*

The dynamic range of waveforms, that is the minimum and maximum frequencies, can to some extent be shown by the spectra. For the spectra of waveforms with sharp cut-off at high frequency, such as waveforms k and h (shown in Figure 3.15), the spectra show that waveform h has higher frequency components than waveform k. Similarly, it can be seen that waveform p has higher frequency components than waveform h. For the spectra at low frequency, waveform e has the highest amplitude of the waveforms in Figure 3.15, implying that waveform e has low-frequency components with larger amplitudes than any of the other waveforms. Although the spectrum is computed correctly (compare the Fourier series of a a square wave [39]), the amplitude of the spectrum of the step waveform e does not reflect the amplitudes of distinct frequencies.

Within the dynamic range between the maximum and minimum frequencies, the spectra can to some extent be used to compare the density in the frequency domain. For example, comparing the spectra of waveforms a and h (Figure 3.15), it is observed that waveform h has frequency components where the spectrum of waveform a has no frequency components. From Figure 3.15, according to the same argument, it can be observed that waveform q is more dense in the frequency domain than waveform f is.

From the above discussion it can be seen that the spectra of waveforms can only be used to compare the highest frequencies of waveforms. The amplitude, lowest frequency or frequency density cannot be compared quantitively by using the spectra. It is concluded that in order to have quantified characteristics for waveforms (*amplitude*, *dynamic range* and *density within the dynamic range*), the spectra cannot be used. For this reason, methods to quatify waveform characteristics will be developed in the next section.

3.3.3 Time domain analysis

In the previous section it was shown that the spectra cannot be used to quantify waveforms' characteristics with confidence. In order to compare characteristics quantitively, time domain analyses methods will be developed in this section. The characteristics that ought to be quantified are the *amplitude*, *dynamic range* and *density within the dynamic range.*

It is easy to compare amplitudes of waveforms in the time domain. A simple procedure, directly implementing the requirement that $A_{train(min)} \leq A_{test} \leq A_{train(max)}$, will suffice.

In order to compare the *dynamic range* and *density within the dynamic range* of waveforms, two functions, sum density and rate of change density will be developed.

The average sum over several points of a waveform at a specific amplitude, could indicate the existence of either slow-varying parts of the waveform, or any arbitrary waveform. However, the maximum or minimum average sum at a specific amplitude, will indicate the existence of slow-varying parts of the waveform. In order to indicate the existence of the slowest-varying parts of the waveform relevant to the problem, the average sum was taken over the number of points K of the input pattern of the neural network. In this case it is 51 (the current value and 50 time delays). With K points, the average sum for a waveform x is given by

$$S_n = \frac{1}{K+1} \sum_{k=-\frac{K}{2}}^{k=\frac{K}{2}} x_{n+k}$$

for $\frac{K}{2} \leq n \leq nts - \frac{K}{2}$ where nts is the number of points of the waveform. The sum density function is derived by making the average sum S_n versus amplitude relationship discrete and assigning an 1 if there were an average sum and amplitude on the discrete grid thus formed. For both the axis of the average sum and amplitude, 20 divisions were used. This allows for a maximum of 400 countable features. This number was chosen to be less than the number of weights or the number of training patterns, but suffient to enable comparison of waveforms when using these features. The results will indicate that this was an appropiate choice.

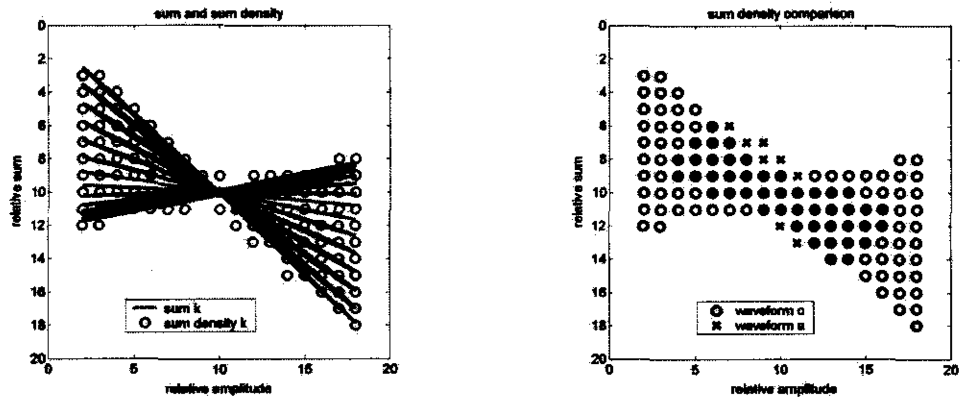

Figure 3.16: *Average sum and sum density of waveform k and sum density of waveforms c and a*

Figure 3.16 shows the relationship between the average sum versus amplitude and the sum density function for waveform k. As an example of how the sum density will be used to compare waveforms, Figure 3.16 shows the sum density functions for waveforms c and a. The sum density of waveform a has 53 features, of which 8 were not mapped by the sum density of waveform c. The percentage sum density features of waveform a not mapped by the sum density features of waveform c will be used as measure, which in this example $\frac{8*100}{53} = 15\%$.

The rate of change of an input waveform to a system will cause a certain output response. For a linear system, rates of change of the input could result in characterising the system at any amplitude. For non-linear systems, system characteristics change with the absolute amplitude of the input. Therefore, to characterise a non-linear system, an input waveform should have different rates of change at several amplitudes. The rates of change of a waveform were determined by making use of a first order polynomial fit (using MATLAB's `polyfit` function). The polynomial fit was taken over 3 points, regarded as sufficiently few to determine the highest rates of change of a waveform. The rate of change density function is the discrete version of the rates of change versus amplitude, using a grid of 20×20. A 1 is assigned to the rate of change density function if a rate of change exists at a specific amplitude. Figure 3.17 shows the rate of change and the rate of change density for waveform m.

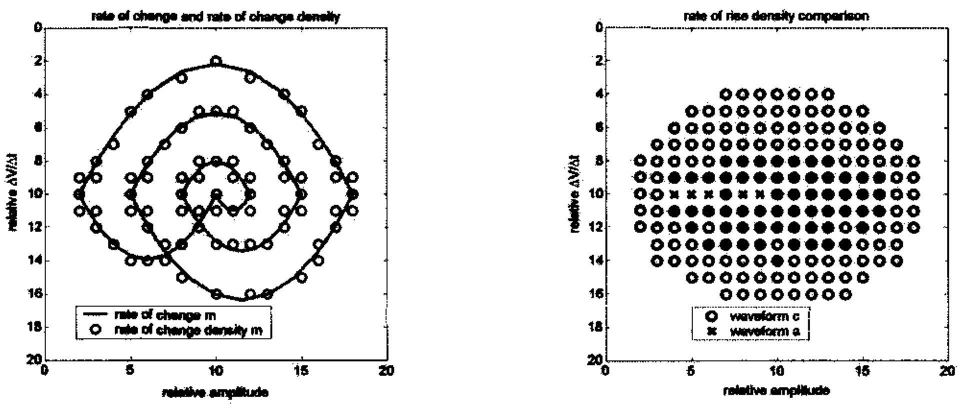

Figure 3.17: *Rate of change and rate of change density for waveform m and rate of change density for waveforms c and a*

As an example of how the rate of change density will be used to compare waveforms, Figure 3.17 shows the rate of change density functions for waveforms c and a. The the rate of change density of waveform a has 64 features, of which 5 were not mapped by the rate of change density of waveform c. The percentage rate of change density features of waveform a not mapped by the rate of change density features of waveform c will be used as measure, which in this example is $\frac{5*100}{64} = 7.8\%$.

In this section, two functions, the sum density function and rate of change density function, have been developed in order to compare waveform characteristics. The unmapped features of these functions, expressed in percentage, will be used to compare waveforms. In the next section, the neural network will be trained by using a training set and tested by using test sets. The test input waveforms are compared to training input waveforms with the sum density and rate of change functions.

3.3.4 Experimental results

Four training experiments were conducted in this section. A time-delayed feed-forward neural network shown in Figure 3.4 with 50 time delays, and 30 neurons in the hidden layer were used for these experiments. The training sets c, k, h, and j were used respectively. The trained neural networks were tested with all the training or test sets. The error, expressed in mean square error (MSE) and the maximum output error in the time domain, in percentage of the maximum output, were used to evaluate the inference accuracy. As an example of how the maximum error in the time domain was obtained, Figure 3.18 shows the error in the time domain with test sets a and m, with set c as training set. With a maximum absolute amplitude of $0.5V$, the errors are $\approx 6\%$ and $\approx 20\%$ respectively. Since the mean square error (MSE) will not show large output errors that occur for a short time interval, the maximum error in the time domain was preferred as error measure, being a more conservative measure. Comparing the errors in Figure 3.18, the large errors with waveform m as input might not be reflected by the mean square error.

Training set c

In this experiment, the neural network was trained by using training set c. The network was simulated for each waveform, and the output error, in mean square error (MSE) and maximum error in the time domain, were determined. Compared to the training waveform c, the sum density features of the waveforms not mapped by the sum density features of the training waveform and the rate of change density features of the waveforms not

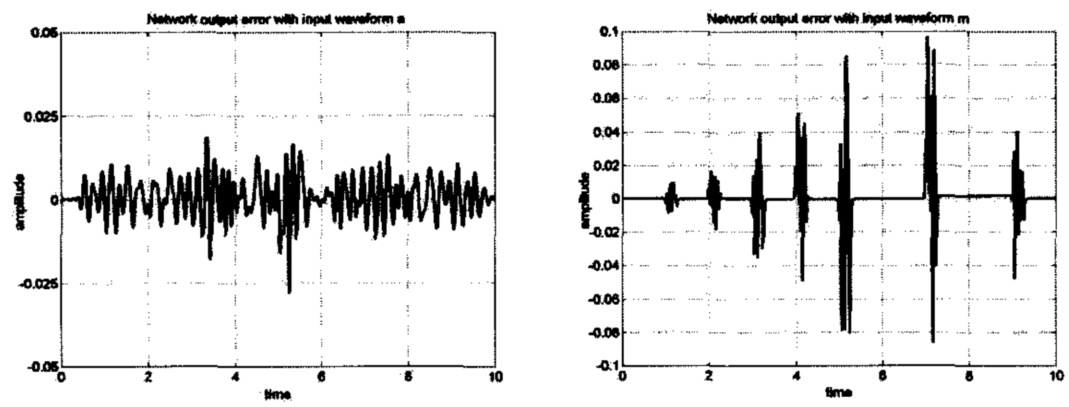

Figure 3.18: *Time domain error with training set c and test inputs a and m*

mapped by the rate of change density features of the training waveform, were determined. The output errors and the percentage features that were not mapped, are indicated in Table 3.1.

Table 3.1: *Training results using training set c, and waveform characteristics*

waveform	MSE	% max error	% unmapped sum density	% unmapped rate of change density
a	$3.6 * 10^{-5}$	6	7.8	15
c	$1.6 * 10^{-8}$	0.05	0	0
e	$2.9 * 10^{-5}$	17	0	29
f	$3.7 * 10^{-5}$	3.2	3.4	0
h	$1.7 * 10^{-3}$	50	26	5.2
j	$1.7 * 10^{-3}$	50	26	13
k	$1.4 * 10^{-5}$	7	2.3	0.9
m	$1.7 * 10^{-4}$	20	9.1	10.1
p	$1.8 * 10^{-4}$	10	9.4	0
q	$1.1 * 10^{-2}$	100	37	0

From Table 3.1, the time domain error, in percentage of output amplitude, versus the percentage unmapped features, are plotted in Figure 3.19 The magnitude of the percentage error in the time domain is shown graphically in Figure 3.19. The radii of the circles in the presentation is approximately proportional to the error in the time domain for the appropriate waveform.

From Figure 3.19 it can be observed that the output error increases as either the rate of

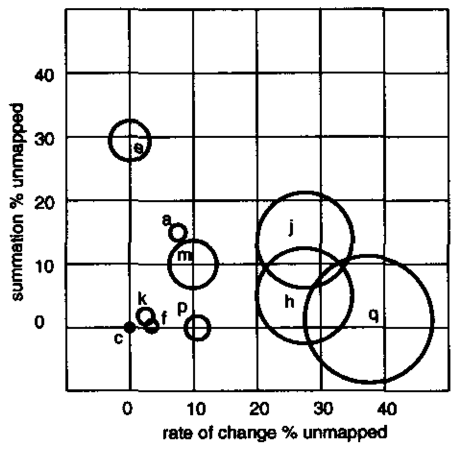

Figure 3.19: *Percentage error in time domain versus unmapped density features with neural network trained with training set c*

change density or the sum density increases.

Training set k

The same procedure was followed for this experiment as with training set c. The network was trained by using training set k and simulated by using all the input waveforms. The results of the simulation errors for all the waveforms and the percentage of unmapped features are listed in Table 3.2. From Table 3.2, Figure 3.20 was plotted, showing the percentage error in the time domain (with the radii approximately proportional to the error) versus the unmapped features.

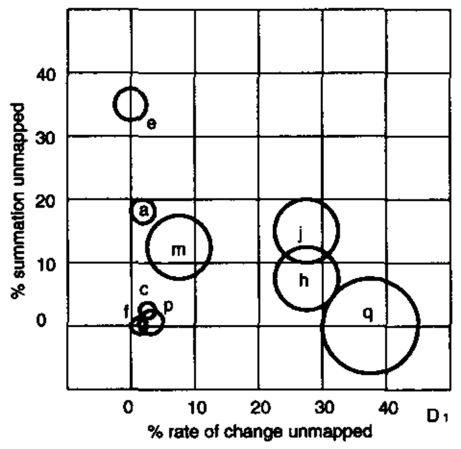

Figure 3.20: *Percentage error in time domain versus unmapped density features with neural network trained with training set k.*

Table 3.2: *Training results using training set k, and waveform characteristics.*

waveform	MSE	% max error	% unmapped rate of change density	% unmapped sum density
a	$1.1 * 10^{-5}$	10	1.6	19
c	$3.6 * 10^{-7}$	1	2	2
e	$4.2 * 10^{-5}$	15	0	35
f	$1.0 * 10^{-5}$	3.3	0.7	0
h	$5.0 * 10^{-5}$	30	26	7
j	$1.2 * 10^{-4}$	30	26	15
k	$9.2 * 10^{-10}$	0.03	0	0
m	$2.1 * 10^{-4}$	30	6.5	12
p	$2.4 * 10^{-5}$	12	1.9	0
q	$7.2 * 10^{-3}$	90	37	0

From Figure 3.20 it can be observed that the output error increases as either the rate of change density or the sum density increases.

Training set h

The same procedure was followed for this experiment as with training set c. The network was trained by using training set h and simulated by using all the input waveforms. The results of the simulation errors for all the waveforms and the percentage of unmapped features are listed in Table 3.3. From Table 3.3, Figure 3.21 was plotted, showing the percentage error in the time domain (with the radii approximately proportional to the error) versus the unmapped features.

From Figure 3.21 it can be observed that the output error increases as either the rate of change density or the sum density increases.

Training set j

The same procedure was followed for this experiment as with training set c. The network was trained by using training set j and simulated by using all the input waveforms. The results of the simulation errors for all the waveforms and the percentage unmapped

Table 3.3: *Training results using training set h, and waveform characteristics.*

waveform	MSE	% max error	% unmapped rate of change density	% unmapped sum density
a	$2.9 * 10^{-6}$	1	0	11
c	$4.2 * 10^{-7}$	0.5	0	2.7
e	$6.3 * 10^{-6}$	5.5	0	41
f	$4.6 * 10^{-6}$	1	0	0
h	$1.6 * 10^{-8}$	0.1	0	0
j	$1.9 * 10^{-5}$	5.4	0	11
k	$9.3 * 10^{-7}$	0.36	0	3.6
m	$5.0 * 10^{-5}$	8.0	0	14
p	$3.4 * 10^{-6}$	1.4	0	1.8
q	$1.4 * 10^{-4}$	14	16	3.1

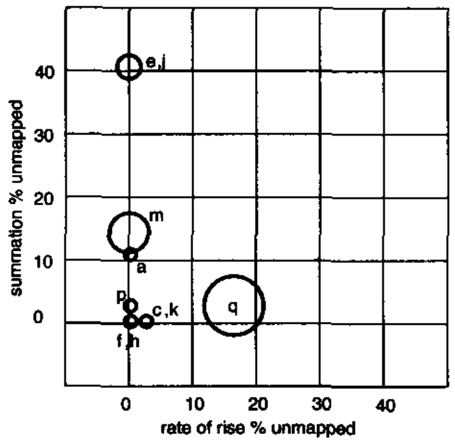

Figure 3.21: *Percentage error in time domain versus unmapped density features with neural network trained with training set h*

features are listed in Table 3.4.

From Table 3.4 it can be seen that the output errors of all the waveforms are small, when the percentage of unmapped features (from the sum density functions and rate of rise density functions) are small. The output error with waveform m as input (with 0 unmapped features) is larger than the output errors of other waveforms, of which the percentage unmapped features are 0. Nevertheless, the results indicate the same general tendencies.

From the results of the four experiments in this section, using training sets c, k, h and

Table 3.4: *Training results using training set j, and waveform characteristics.*

waveform	MSE	% max error	% unmapped rate of change density	% unmapped sum density
a	$1.5 * 10^{-6}$	1	0	0
c	$2.6 * 10^{-7}$	0.7	0	0
e	$5.4 * 10^{-9}$	0.1	0	0
f	$5.0 * 10^{-7}$	0.4	0	0
h	$1.3 * 10^{-8}$	0.04	0	0
j	$2.1 * 10^{-8}$	0.1	0	0
k	$5.2 * 10^{-7}$	0.36	0	0.9
m	$1.5 * 10^{-6}$	2.5	0	0
p	$6.5 * 10^{-7}$	0.56	0	0
q	$1.5 * 10^{-5}$	90	16	0

j, it can be concluded that *the percentage of output errors increases as the percentage of unmapped features increases.* Using the waveform characteristics (the sum density functions and rate of change functions) for determining the unmapped features proved to be an appropiate test to predict whether a trained neural network would have a small error for a test input waveform (or validation input waveform or a specification expressed as an input waveform). This result implies that neural networks need not be tested or validated by using test or validation sets, since the characteristics of test or validation input waveforms can be compared to the training input waveform.

3.4 Multi-variable inference

In the previous sections, inference of a single variable based on a single input variable was discussed. This section looks at two other possibilities, namely the inference of two variables from a single input, and the inference of a variable from two independent input variables. The systems that were used for these two cases and the topologies required, were discussed in the previous chapter. Of concern in this chapter, are the requirements for the training excitation waveforms.

For the single input, dual output, the same requirements for the training set input compared to the test (or validation) input waveforms such as discussed in this chapter are applicable. The results in the previous chapter illustrated this.

For the inference of a single variable from two independent variables, it is required to have training input waveforms of which one input waveform has the whole range of characteristics for any characteristic of the other input waveform. This implies that the training input waveforms will have a considerably longer time span than the single input counterpart for the same sampling rate and frequency range. This is not constructive for creating such input waveforms. However, as experienced in the previous sections when constructing training input waveforms for the single input case, the two input waveforms (for the training set) shown in Figure 3.22 were created. The basis for these waveforms is a waveform that consists of a chirp signal with constant parts at the minimum and maximum amplitude, and assumed to be favourable for mapping the required relationship accurately. Such a waveform basis, for instance, can be seen in Figure 3.22 in time period $15 \leq t \leq 27$ s. These two inputs are given by Equation 3.18 and Equation 3.19 respectively. The numerical values were chosen, as before, for convenience, and can be scaled for a specific application.

It can be seen from Figure 3.22 that for every characteristic, whether a slow-varying pattern or a fast-varying pattern of the one waveform, the other waveform has a range of patterns ranging form slow-varying to fast-varying patterns. The neural network for dual input, single output inference shown in the previous chapter was trained by using the training set data obtained by the same method as discussed before. Table 3.5 shows the training results for dual input, single output inference, and the output errors with either one of the two inputs were held at a constant value of $0.8V$.

$$\begin{aligned}
e_1 \;=\; & 0.4[(\tanh(5(t-13.0)) - \tanh(5(t-49.5)))\sin(t-13.0)^2 \\
& + \tanh(50(t-0.6)) - 2\tanh(50(t-1.2)) + \tanh(50(t-1.8)) \\
& + \tanh(50(t-2.4)) - 2\tanh(50(t-3.0)) + \tanh(50(t-3.6)) \\
& + \tanh(50(t-4.2)) - 2\tanh(50(t-4.8)) + \tanh(50(t-5.4)) \\
& + \tanh(50(t-6.0)) - 2\tanh(50(t-6.6)) + \tanh(50(t-7.2)) \\
& + \tanh(50(t-7.8)) - 2\tanh(50(t-8.4)) + \tanh(50(t-9.0))
\end{aligned}$$

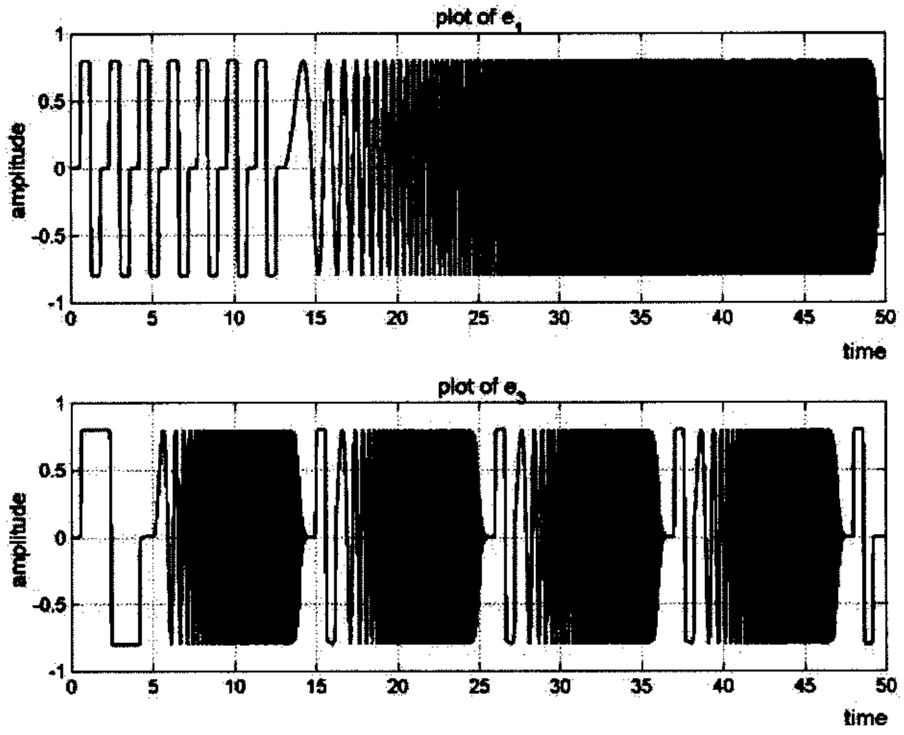

Figure 3.22: *Training set input waveforms for dual input, single output inference*

$$
\begin{aligned}
&+\tanh(50(t-9.6)) - 2\tanh(50(t-10.2)) + \tanh(50(t-10.8)) \\
&+\tanh(50(t-1.4)) - 2\tanh(50(t-12.0)) + \tanh(50(t-12.6))] \qquad (3.18)
\end{aligned}
$$

$$
\begin{aligned}
e_3 \;=\;& 0.4[(\tanh(5(t-5.0)) - \tanh(5(t-14.0)))\sin(4(t-5.0)^2) \\
& \tanh(5(t-16.0)) - \tanh(5(t-25.0)))\sin(4(t-16.0)^2) \\
& \tanh(5(t-27.0)) - \tanh(5(t-36.0)))\sin(4(t-27.0)^2) \\
& \tanh(5(t-38.0)) - \tanh(5(t-47.0)))\sin(4(t-38.0)^2) \\
& +\tanh(50(t-0.6)) - 2\tanh(50(t-2.4)) + \tanh(50(t-4.2)) \\
& +\tanh(50(t-15.0)) - 2\tanh(50(t-15.6)) + \tanh(50(t-16.2)) \\
& +\tanh(50(t-26.0)) - 2\tanh(50(t-26.6)) + \tanh(50(t-27.2)) \\
& +\tanh(50(t-37.0)) - 2\tanh(50(t-37.6)) + \tanh(50(t-38.2)) \\
& +\tanh(50(t-48.0)) - 2\tanh(50(t-48.6)) + \tanh(50(t-49.2))] \qquad (3.19)
\end{aligned}
$$

Table 3.5: *Training results for the inference of a single variable with two input variables*

inputs	MSE	% max error
e_1, e_3	$1.3 * 10^{-6}$	2
e_1, $e_3 = 0.8V$	$3.6 * 10^{-7}$	0.4
$e_1 = 0.8V$, e_3	$3.4 * 10^{-8}$	0.2

3.5 Discussion

This chapter has addressed two important issues, namely whether the simulated data sets used for training or testing are valid, and the characteristics of training input waveforms, compared to test input waveforms that are necassary to ensure accurate inference.

The validity of data sets were based on the conservation of energy and the fundamental relationships of storage elements. Due to the presence of noise in practical systems and the finite bit resolution of digital systems, the influence of noise and finite bit resolution on inference accuracy was determined. From these results it can be concluded that the inference errors in the subsequent sections are not dominated by the inaccuracy of data, but by the specific experimental setup.

A comparison between training input waveforms and test (or validation) input waveform characteristics was made. The purpose of these comparisons were to show whether the hypotheses regarding the *amplitude*, *dynamic range* and *density within the dynamic range* of training inputs compared to test (or validation) inputs were true. Frequency and time domain analysis methods were used. When using the spectra in the frequency domain as waveform characteristics, training input waveforms and test (or validation) input waveforms cannot be compared in order to predict whether inference would be accurate. Because of the inadequacy of spectra to make such a prediction, time domain analysis methods were developed. These time domain methods consist of two functions, namely the sum density function and rate of change density function. Using the sum density function and rate of change function to compare training input waveforms and test (or validation) input waveforms, waveforms could be compared quantitively.

For the sum density function and rate of change density function, test (or validation) input waveforms are compared to the training input waveform. The sum density features or rate of change density features of the test (or validation) input waveforms not mapped by the training input waveform serve as a quantitive measure for comparison. When all the features of the test (or validation) input waveforms have been mapped by the features of the training input waveforms, accurate inference is possible for the test (or validation) input waveforms when the training set was used for training the neural network for inference, subject to certain constraints. These constraints are the topological requirements and the training algorithm capabilities. The topological requirements for accurate inference were derived in the previous chapter. In all cases the training algorithm was capable of finding a local minimum with a sufficiently small error.

This chapter has shown that inference will be accurate (using test or validation sets as acceptability tests) if all the features of the test or validation input waveforms were mapped by the features of the training input waveform. The hypotheses have been shown to be true, and in effect makes it unnecessary to test or validate neural networks after training. These results not only apply to inference measurements, but also to modelling systems. Since a comparison can be made between a training input waveform and test (or validation) input waveforms, it can subsequently be predicted whether accurate inference would be possible. A training input waveform can be constructed to allow accurate inference measurements. Specifications, such as maximum amplitude or frequency range, can be expressed as test input waveforms. Training input waveforms can be constructed that would map all the sum density or rate of change density features of such a test input waveform in order to allow accurate inference measurements (or system modelling) with time-delayed feed-forward neural networks. It is assumed that validation sets will fall within these specifications and that the trained neural network would pass such validation tests.

Assuming the appropiate neural network topology and capable training algorithm, the following is a summary of the steps to be taken to allow accurate inference measurements that should allow the trained neural network to pass test or validation tests:

1. If the specifications for the inference (or modelling) problem were not in the form of test input waveforms, translate them into test input waveforms.

2. Construct a training input waveform that will map all the sum density features and rate of change density features of the test input waveforms. If the sum density function of the training input waveform does not map all the sum density features of the test input waveforms, add slow-varying patterns to the proposed training input waveform at the relevant amplitudes so that all the sum density features of the test waveform inputs are mapped. If the rate of change density function of the training input waveform does not map all the rate of change density features of the test input waveforms, add fast-varying patterns to the proposed training input waveform at the relevant amplitudes so that all the rate of change density features of the test waveform inputs are mapped.
3. Generate the training set in such a way that the training input waveform is the known variable from which some other variable must be inferred.
4. Train the time-delayed feed-forward neural network by using the training set.

The next chapter will apply the methods developed in this and the previous chapter to a simulated Brayton cycle, and the simulated pebble bed micro model (PBMM), which, from a thermodynamic viewpoint, closely resembles the pebble bed modular reactor (PBMR).

Chapter 4

PBMR neural network inference measurements

This chapter applies the inference methods developed in the previous chapters to the Brayton cycle power plant, which is to be used in the pebble bed modular reactor (PBMR). Two examples are presented. The first example is based on a simulation, using MATLAB's ODE solver to solve the variables of the state equations of the Brayton cycle with a single axis. The second example is based on a simulation using Flownet of the three axes Brayton cycle of the pebble bed micro model (PBMM). This is a scaled version of the thermodynamic part of the pebble bed modular reactor (PBMR), designed, built and demonstrated at the Potchefstroom University, South Africa.

4.1 The pebble bed modular reactor

The pebble bed modular reactor (PBMR) is a new type of nuclear reactor currently being developed. PBMR is also the name of the South African company that develops this reactor. However, in this thesis PBMR will refer to the reactor and not the company. In South Africa the PBMR is approaching the construction phase, with the first plant to be constructed in 2005. Also called a high-temperature reactor (HTR), China and Japan have commissioned small HTR reactors [40].

This type of reactor differs from conventional nuclear reactors in many aspects. The most appealing difference is the PBMR's inherent safety in terms of possible core meltdown and subsequent radioactive leakage to the environment. Inherent safety, or passive safety, means that core meltdown is not possible even if all control and safety mechanisms were to fail. To achieve this inherent safety, the enriched uranium fuel is dispersed in billions of particles (with diameter $\approx 0.5mm$) with several high-density coatings [40, p52]. One of the layers is tough silicon carbide ceramic that serves as a miniature pressure vessel capable of containing the fission products up to temperature of $1600°C$. These particles are encapsulated in graphite spheres ($\approx 50mm$ diameter) that can withstand the maximum temperatures reached through convection cooling even if all control systems and safety mechanisms were to fail. There are typically 15000 coated particles encapsulated in a graphite sphere, and several hundred thousand graphite spheres are stacked to form the reactor core. These graphite spheres (or pebbles) have a high specific heat capacity and serves as a moderator. The power density of the pebbles are typically less than one-tenth of that of a conventional nuclear reactor. Fuel replenishment schemes totally refuel after a few years with fuel added during this period (resulting in a growing stack), or continuous refueling with new, partially spent or unfuelled pebbles. The latter scheme has the advantages that the power and temperature distribution can be shaped, adding to the salient safety of the reactor [40, p52].

A single working fluid (helium) in a closed environment is used as working fluid. Helium is both chemically and nuclearly inert and does not interfere with the nuclear moderation process. Because no phase change takes place, the heat transfer and transport are uniform. The use of a single working fluid allows a theoretical higher efficiency than light water reactors, and reduces the number of components with subsequent lower investment cost. Furthermore, the reactor can be much smaller than the existing types of nuclear reactors, with a maximum electrical output of $100MVA$, making it feasible for utilisation by small consumers at localities not served by a power grid. The inlet and outlet temperatures of the working fluid are typically $500°C$ and $900°C$ respectively, operating at a pressure of $\approx 7MPa$.

The thermodynamic cycle is a recuperative Brayton cycle. The Brayton cycle is the ideal for the closed cycle gas turbine unit. The South African PBMR is a recuperative Brayton cycle with three axes. This means that the main turbine/generator, low-pressure compressor/turbine and high-pressure compressor/turbine are mounted on different axes, allowing more degrees of freedom for control, with subsequent increase in efficiency.

The thermodynamic cycle is designed by making use of the thermodynamic simulation software package Flownet [41]. The models of the individual components of a thermodynamic network can be built in software, and when interconnected, the total (closed or open loop) response of a system can be evaluated. Flownet, and similar simulation packages [42] [43] [44] are based on fundamental thermodynamic principles such as the Maxwell equations [45, p367]. In order to validate the design, including the control systems, the pebble bed micro model (PBMM), a scaled down, simplified version of the thermodynamic part of the PBMR, has been designed, built and demonstrated at the Potchefstroom University, South Africa. The PBMM uses nitrogen as working fluid at a much reduced pressure and electrical elements for heat generation.

4.2 The Brayton cycle

For demonstration of neural network inference implementation, a simulated simplified model using the Brayton cycle consisting of a turbine, a compressor and two heat-exchangers, will be used. Such a simplified system consists of all the major components that are found in the complete pebble bed modular reactor. Figure 4.1 shows the T-s diagram and the model for the basic Brayton cycle [46].

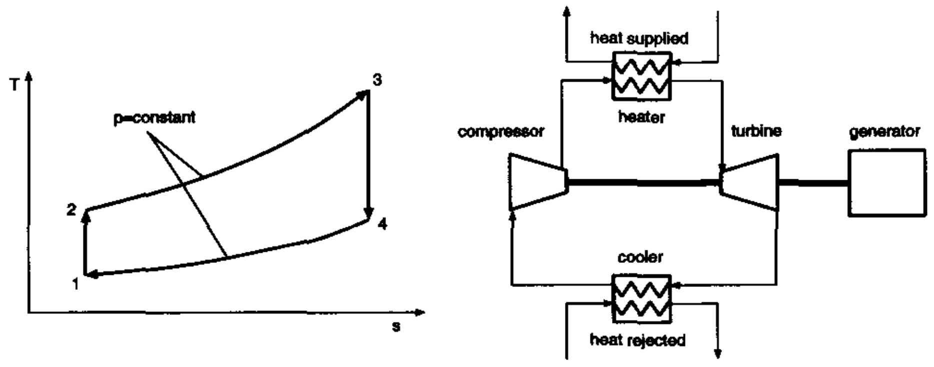

Figure 4.1: *Brayton cycle T-s diagram and basic Brayton cycle*

In order to add dynamics to the basic Brayton cycle, Figure 4.2 shows a diagram of the basic Brayton cycle with heat input Q_{in} and heat output Q_{out}, an input heat exchanger with significant heat capacitance, a load consisting of a flywheel with moment of inertia I_F and varying torque load T_G due to the generator. The output heat exchanger transfers heat to an infinite thermal sink at constant temperature, therefore T_1 is constant.

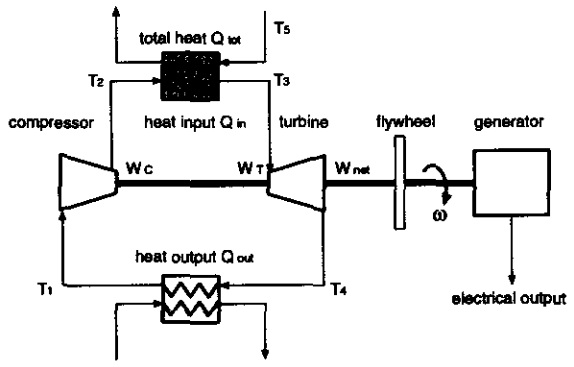

Figure 4.2: *Brayton cycle with heat-exchanger and flywheel*

Assume that the expansion or compression is done isentropically and the heat transfers are at constant pressure p. For steady flow, neglecting velocity changes, Equation 4.1 gives the work and heat transfer for each part of the cycle [46, p156]. The heat capacity C_p of a fluid is the energy per unit mass per unit temperature J/kg.K, therefore the work or heat of Equation 4.1 is in energy per unit mass J/kg.

$$\begin{aligned} W_C &= C_p(T_2 - T_1) \ \ W/kg \\ W_T &= C_p(T_3 - T_4) \ \ W/kg \\ Q_{in} &= C_p(T_3 - T_2) \ \ W/kg \\ Q_{out} &= C_p(T_4 - T_1) \ \ W/kg \end{aligned} \tag{4.1}$$

With r the pressure ratio, γ the specific heat ratio, the netto work $W_{net} = W_T - W_C$ is given by Equation 4.2. Since the work is done isentropically, $\frac{T_2}{T_1} = \frac{T_3}{T_4} = \left(\frac{p_2}{p_1}\right)^{\frac{\gamma-1}{\gamma}}$. At a constant mass flow rate $\dot{m} = 1$ kg/s, from Equation 4.2 the power output to the load P_{net} is given by Equation 4.3.

$$\begin{aligned} W_{net} &= C_p(T_3 - T_4 + T_1 - T_2) \ \ W/kg \\ W_{net} &= C_p\left[T_3\left(1 - \frac{1}{r^{\frac{\gamma-1}{\gamma}}}\right) + T_1\left(1 - r^{\frac{\gamma-1}{\gamma}}\right)\right] \ \ \text{W/kg} \end{aligned} \tag{4.2}$$

$$P_{net} = C_p\left[T_3\left(1-\frac{1}{r^{\frac{\gamma-1}{\gamma}}}\right)+T_1\left(1-r^{\frac{\gamma-1}{\gamma}}\right)\right] \text{ W} \tag{4.3}$$

The total torque $\mathcal{T}_{tot}$ applied to the load is the sum of the generator torque $\mathcal{T}_G$ and the torque applied to the flywheel $\mathcal{T}_F = I_F\ddot{\theta}$. Since $P = \mathcal{T}\dot{\theta}$, the netto power P_{net} in terms of the load is given by Equation 4.4.

$$P_{net} = \dot{\theta}(I_F\ddot{\theta} + \mathcal{T}_G) \text{ W} \tag{4.4}$$

The input heat-exchanger is modelled as a thermal resistance R_t with heat capacitance C_c. Figure 4.3 shows the model for the heat exchanger. The total power P_{tot} is the sum of the power gain of the heat reservoir and the power applied to the thermal cycle, and can be expressed by Equation 4.5, using consistent units [10, p269].

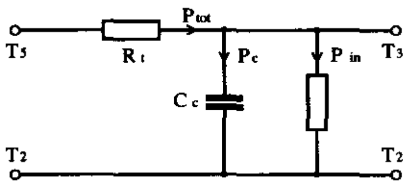

Figure 4.3: *Model for heat-exchanger*

$$P_{tot} = \frac{T_5 - T_3}{R_t} = C_c\dot{T}_3 + P_{in} \text{ W} \tag{4.5}$$

By choosing the state variables $x_1 = \theta$, $x_2 = \dot{\theta}$ and $x_3 = T_3$, from Equation 4.3, Equation 4.4 and Equation 4.5 the state equations are given by Equation 4.6.

$$\begin{aligned}
\dot{x}_1 &= \dot{\theta} \\
\dot{x}_2 &= \frac{C_p}{\dot{\theta}I_F}\left[T_3\left(1-\frac{1}{r^{\frac{\gamma-1}{\gamma}}}\right)+T_1\left(1-r^{\frac{\gamma-1}{\gamma}}\right)\right]-\frac{\mathcal{T}_G}{I_F} \\
\dot{x}_3 &= \frac{1}{C_c}\left[\frac{T_5-T_3}{R_t}-C_p\left(T_3-T_1r^{\frac{\gamma-1}{\gamma}}\right)\right]
\end{aligned} \tag{4.6}$$

4.3 Simulated model of the Brayton cycle

For the simulated model of the Brayton cycle, the turbine characteristic and compressor characteristics were modelled as $r = K_r(1 + A\dot{\theta})$, although more refined thermodynamic modelling will use published compressor and turbine characteristics such as shown in [47, p21]. Practical systems have turbine and compressor characteristics that are more refined than those of the model chosen above. However, with the right pressure ratio at a certain rotational speed $\dot{\theta}$ rad/s, this model is acceptable within a certain operating range. Typical pressure ratios are $5 \leq r \leq 20$ [48, p398]. It was assumed that the mass flow rate is more or less constant (as shown by analysis [43]). The system dynamics are dominated by the effect of the heat capacitance C_c of the input heat exchanger and the moment of inertia I_F of the flywheel.

Temperatures encountered in practical Brayton cycle systems are typically $T_1 = 300\ K$ and $T_3 \approx 900\ K$ [46, p156]. By choosing the fluid diatomic (such as nitrogen), the specific heat ratio $\gamma \approx 1.4$ [45], with a specific heat capacity $C_p = 1.039\ kJ/kg.K$.

The state equations for the Brayton cycle (Equation 4.6) was normalised by multiplying throughout by a constant $A = 0.001$, as shown in Equation 4.7.

$$\begin{aligned}
A\dot{x}_1 &= A\dot{\theta} \\
A\dot{x}_2 &= \frac{C_p}{\dot{\theta} I_F}\left[AT_3\left(1 - \frac{1}{r^{\frac{\gamma-1}{\gamma}}}\right) + AT_1\left(1 - r^{\frac{\gamma-1}{\gamma}}\right)\right] - \frac{AT_G}{I_F} \\
A\dot{x}_3 &= \frac{1}{C_c}\left[\frac{AT_5 - AT_3}{R_t} - C_p\left(AT_3 - AT_1 r^{\frac{\gamma-1}{\gamma}}\right)\right]
\end{aligned} \tag{4.7}$$

The mass flow rate was chosen as $\dot{m} = 1\ kg/s$, the heat capacitance was chosen as $C_p = 1.0\ kJ/kg.K$, the maximum and minimum temperatures were chosen as $T_1 = 300\ K$ $T_5 = 900\ K$ respectively, the moment of inertia was chosen as $I_F = 0.01\ kg.m^2$, the thermal resistance was chosen as $R_t = 0.1\ ^oC/W$, heat capacitance was chosen as $C_c = 0.1\ kJ/kg.K$ and the choice of $K_r = 5$ ensured a pressure ratio $1 \leq r \leq 10$.

Several variables should be monitored on a thermodynamic system. These are pressures, total mass, mass flow rate, fluid temperature, the rotational speeds of the compressors and turbines as well as the power input and output. Since measuring the fluid temperature is problematic and cannot be measured accurately or timeously [49, p252], it was used as an example for demonstrating inference measurements on this model of the Brayton cycle. Temperature measurements of flowing gases are accomplished by using a thermocouple in a thermometer well. Conduction and convection or convection and radiation reduce the accuracy that could be achieved with such a measurement setup.

Training and test set data were generated by using a varying load torque as an independent variable and solving the state equations of Equation 4.6, using MATLAB's ODE solver.

4.4 Inference measurements on the simulated Brayton cycle model

By using the methods developed in the previous chapters, (i) the appropriate neural network topology for inference measurement on the system model was found, (ii) the training and test input waveforms were constructed, (iii) the training and test input waveforms were compared, using the sum density function and the rate of change density function, and (iv) the neural network was trained using the normalised torque AT_G as input and the normalised temperature AT_2 as target.

4.4.1 Neural network topology

In **Chapter 2: Network topology and system characteristics** it was shown which neural network topology to choose for a specific type of system. To do so, it must be established whether the system is dynamic or static, linear or non-linear.

The transient response with a step input applied can be seen in Figure 4.4. The decay of the temperature shows that the system is dynamic, which requires a neural network with time delays. The number of time delays is based on the time when the amplitude has reached the final value within an arbitrary small error. Using Figure 4.4, with the amplitude within 0.05 % of the final value, the time decay was chosen as 0.3s. With time step $\Delta T = 0.005\ s$ the time delays $TD = \frac{time\ decay}{\Delta T} = 60$.

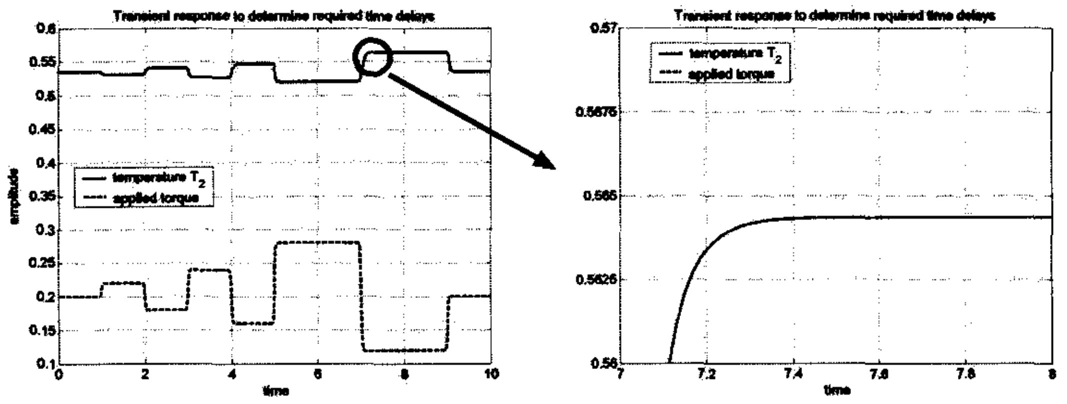

Figure 4.4: *Transient response of normalised output AT_2 with normalised input $A\mathcal{T}_G$*

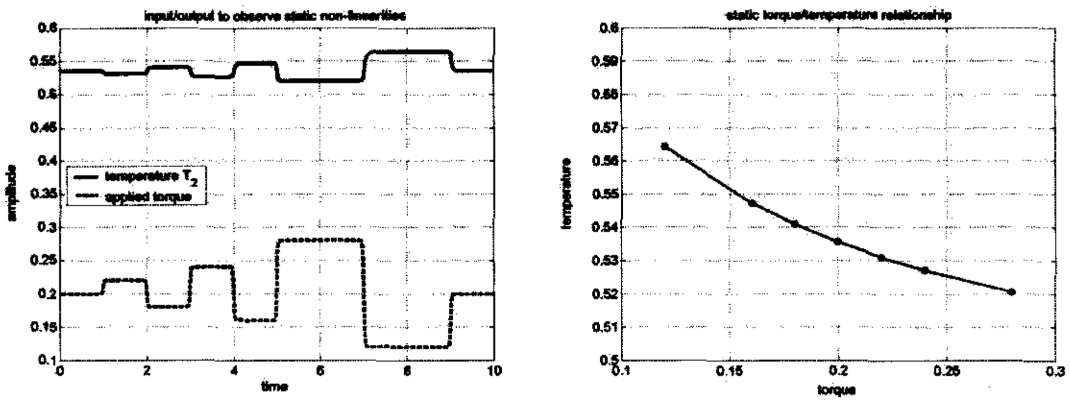

Figure 4.5: *Static input/output relationship of the normalised temperature T_2 vs normalised generator torque $\mathcal{T}_G$*

Two methods were presented by means of which it can be established whether a system is linear or non-linear. The existence of static non-linearities can be shown with a plot of the input/output relationship of static operating points, as shown in Figure 4.5. By plotting the output/input relationship of a slow-varying part of the input and output waveforms, the interpretation of the shape obtained can show whether the system is linear or non-linear. A distorted oval, as shown in Figure 4.6, indicates that the system is non-linear. For modelling non-linearities, the neural network must have at least one layer with non-linear activation functions. A few trial training runs, using different numbers of neurons in the layer, would result in the optimum number of neurons.

From the input/output relationship it was seen that the neural network is required to map a non-linear dynamic relationship. For this type of mapping, a time-delayed feed-forward

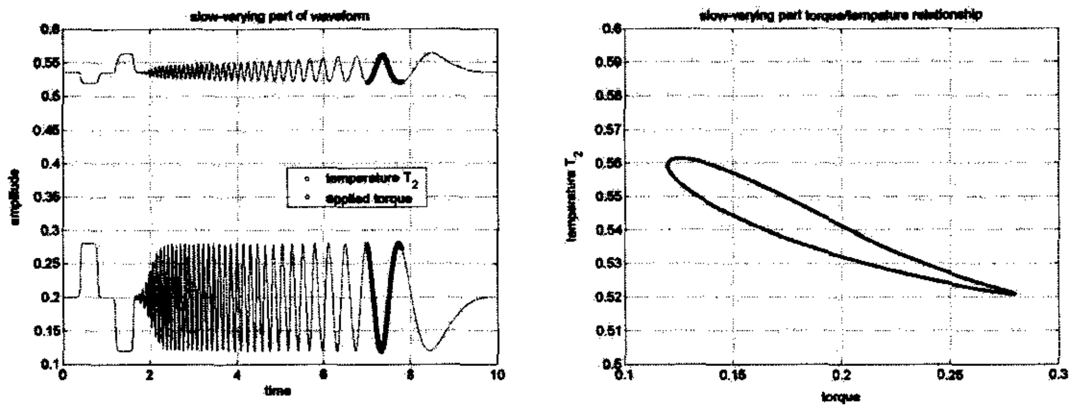

Figure 4.6: *Response of output with slow-varying input and input/output relationship at slow-varying part*

neural network with input time delays and a single hidden layer is required. With a few trial training runs, 30 neurons was selected for inferring the normalised temperature AT_2 from the applied generator load torque $A\mathcal{T}_G$ for the simulated Brayton cycle.

4.4.2 Training and test sets

For the simulation of data sets, the normalised generator torque $\mathcal{T}_G$ was varied according to the exitation signals in Figure 4.7. The data set, using exitation signal n, was used as training set. The data sets from the other exitation signals were used as test sets, or test sets derived from some specification. For example, exitation signal c will represent a frequency range specification of $1\ Hz \leq f \leq 7\ Hz$ for a torque range of $0.12\ N.m \leq A\mathcal{T}_G \leq 0.26\ N.m$.

In the previous chapter, comparisons were made between the training neural network input waveform and the test input waveforms. The sum density functions and rate of change density functions were used for the comparison of input waveforms. It was shown that the neural network simulation error increases as the percentage of unmapped features of a function increases when compared to the proposed training input waveform.

By comparing the characteristics of test inputs to those of the training input before training, the test inputs that are accommodated in the neural network model can be predicted. For example, the sum density functions and the rate of change density functions

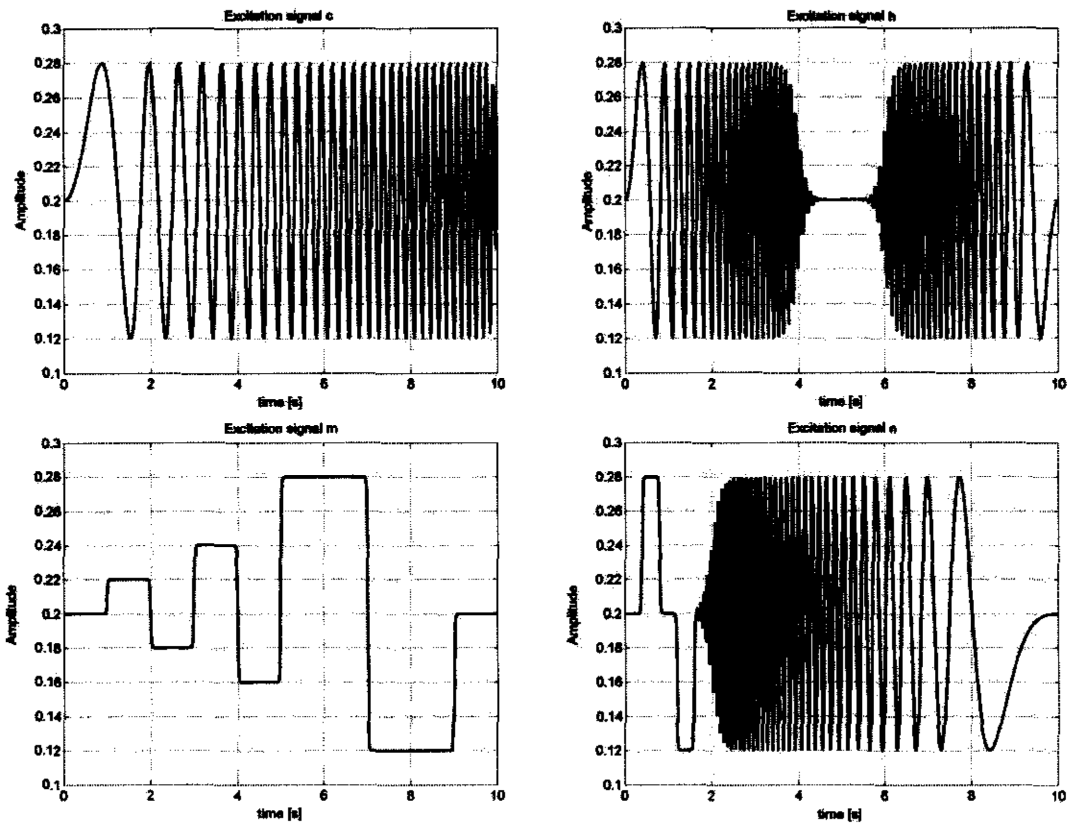

Figure 4.7: *Normalised input AT_G training and test input waveforms for inference of the normalised temperature AT_2*

for input waveforms n and c are shown in Figure 4.8. The percentage of unmapped features of test waveforms not mapped by the features of the training input waveform n were determined for all the input waveforms and listed in Table 4.1.

Based on the comparison of the waveforms and using waveform n as training input, it can be predicted that the neural network should be able to infer the temperature T_2 accurately from the torque T_G for waveforms c and h, if the training algorithm were capable of finding a local minimum with a sufficiently small error. A larger error is expected for inference of the temperature AT_2 from the torque AT_G with input waveform m, because some features of waveform m are not mapped by the features of waveform n.

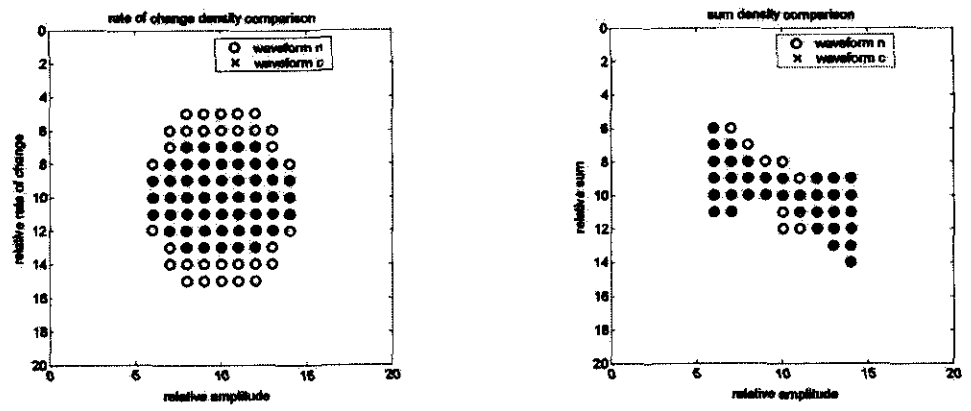

Figure 4.8: *Rate of change density functions and sum density functions for input waveforms n and c*

4.4.3 Neural network training

The time-delayed feed-forward network was simulated and trained by using the training set generated with the torque T_G varying to exitation signal n. The input to neural network is the normalised torque AT_G, and the target is the temperature T_2. The error target was set at a mean square error of $MSE = 10^{-8}$, and trained with the Levenberg-Marquardt algorithm. The training process was stopped after 20 epochs. This number of epochs was chosen based on Figure 2.18, where it can be seen that the training algorithm reaches an error close to the minimum error within 20.

Table 4.1: *MSE and maximum error of output for all input waveforms with set* n *as training set*

waveform	MSE	% max error	% unmapped rate of change density	% unmapped sum density
c	$6.5 * 10^{-8}$	1.2	0	0
h	$4.4 * 10^{-8}$	1.3	0	0
m	$1.2 * 10^{-7}$	3	3.6	0
n	$8.9 * 10^{-8}$	1.5	0	0

The training results expressed in mean square error MSE and maximum percentage error in the time domain are shown in Table 4.1. The maximum percentage error in the time domain is calculated from

$$\% \text{ max error} = \frac{max|e(t)|}{max(T_2) - min(T_2)}$$

where $e(t)$ is the output error in the time domain for a specific input waveform, and T_2 is the temperature. From Figure 4.6, the normalised temperature range is approximately from $0.52 \leq T_2 \leq 0.565$. With a normalisation factor of $A = 0.001$, a 1 % error is 0.45 ^oC.

In this section it was shown that the correct topology can be found from the data from the input/output relationship between the known variable (the torque T_g) and the variable to be inferred. Furthermore, by comparing the characteristics of the training input waveform and the test input waveforms, it could be predicted whether a variable will be inferred accurately. In the next section, the same techniques will be applied to a simulated model of the pebble bed micro model (PBMM).

4.5 Inference measurements on the PBMM

The thermodynamic cycle of the pebble bed modular reactor (PBMR) and the pebble bed micro model (PBMM) are identical, except in size. Therefore the pebble bed micro model (PBMM), having been built, was used to demonstrate the inference measurement techniques developed in the previous chapters. Figure 4.9 shows the three axes recuperative Brayton cycle used for the pebble bed micro model (PBMM) to be used for the South African PBMR [50]. The pebble bed micro model (PBMM) output is rated at $100kW$ with a maximum heat input of $500kW$. The output power is controlled by mass injection and removal. In order to map the input/output characteristics of the model's components, such as the turbines or compressors, mass injection and removal were performed on the Flownet model [51] of the pebble bed micro model. A MATLAB Simulink interface with Flownet [52] enables the use of MATLAB for simulating and training the neural networks for inference.

The generation of data for training and test sets were accomplished in the following way. A static nominal operating point with an output of $36kW$ at constant rotational speed was chosen. Mass was injected or removed at point P of the model shown in Figure 4.9.

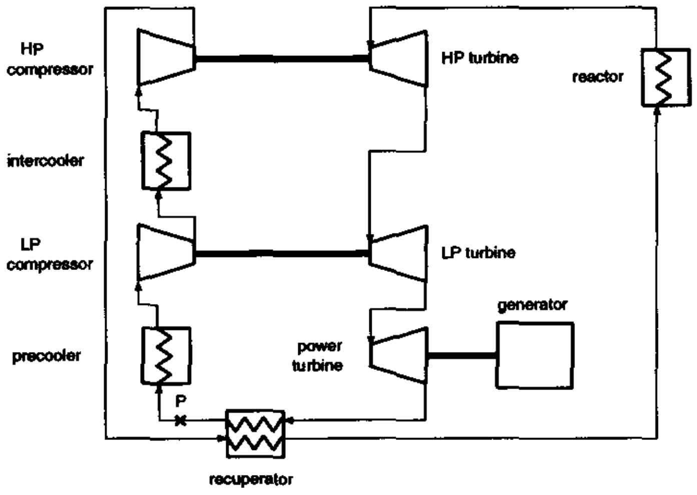

Figure 4.9: *PBMM/PBMR recuperative Brayton cycle with three axes*

Two case studies were done by using data from the simulated pebble bed micro model (PBMM). For the first case study, the mass injection/removal rate was directly varied according to certain training and test input waveforms. In the second case study, the data was generated by using a controller that acts on the mass injection/removal rate so that a variable in the system would vary according to certain training and test input waveforms.

4.5.1 PBMM: case study I

In this case study, the mass injection/removal rate was done according to the training and test input waveforms. A nominal operating point for the generator at a fixed rotational speed of 3000 *rpm* delivering 36 *kW* was chosen. This output level is approximately half of the maximum operating capacity, and therefore a variation around this operating point can be established. The mass injection/removal rate according to the training and test input waveforms resulted in stable operation (with positive power output) of the simulated pebble bed micro model (PBMM). It was assumed that these variations in mass rate change can be realised and will not cause any component to operate beyond its rating or cause instability.

Measuring variables on a system such as the PBMM requires sensors such as pressure sensors, mass flow meters and temperature sensors. Pressure sensors and flow meters, which rely on the pressure drop over a calibrated orifice, are accurate and have good frequency characteristics. Temperature sensors, however, have large time lags, and therefore it is desirable to infer the temperature measurement from other variables. Since the temperature, mass flow rate and pressure in a thermodynamic system are interrelated, the temperature can be inferred from these values. It was therefore decided to use the mapping of the pressure/temperature relationship to infer the temperature from the pressure. For a demonstrative example, the inlet pressure to the power turbine was used for inference of the power turbine temperature.

For successful inference, it was shown that the correct topology and a favourable comparison of test input waveforms and the training input waveform are required. By using the mass injection/removal rate according to the waveforms in Figure 4.10, four sets of input/output data were simulated. The time interval used was $\Delta T = 1\ s$ for the step size of the thermodynamic solver Flownet. The same step size was used for the training and test sets, resulting in 2001 data points for each variable.

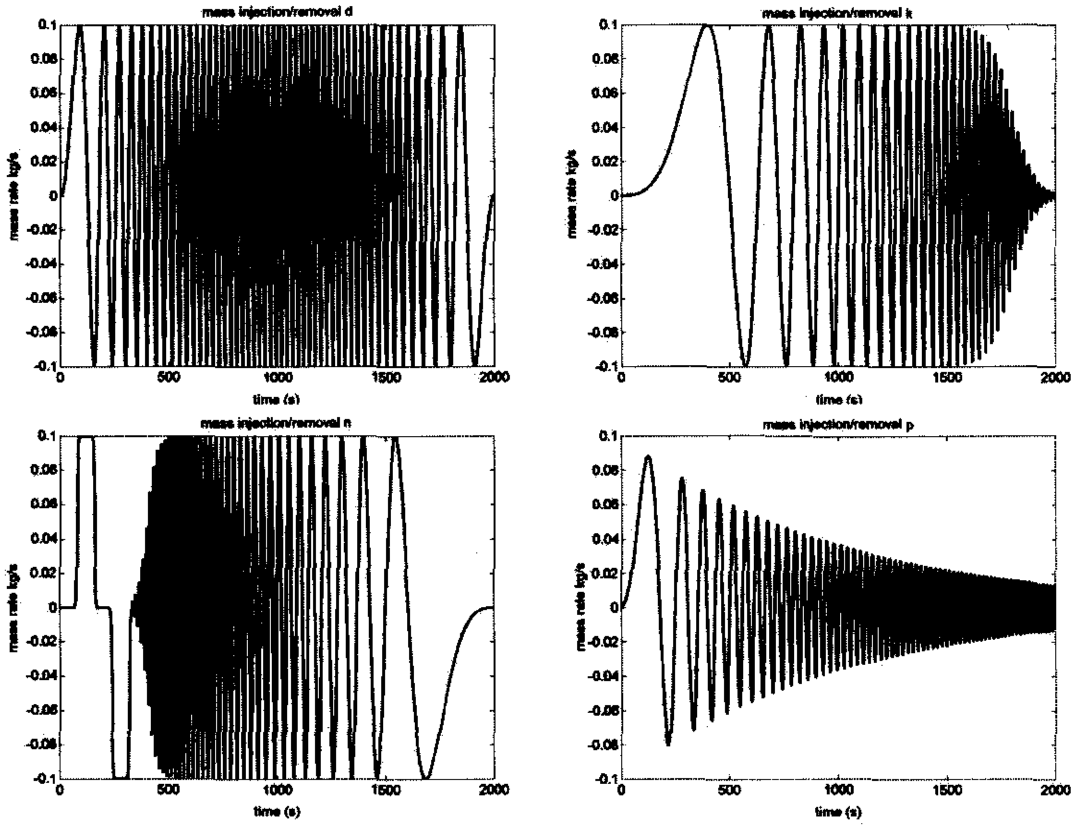

Figure 4.10: *Mass rate injection/removal signals for PBMM inference measurements*

Neural network topology

For the choice of the neural network topology it must be determined whether the input/output relationship to be mapped for an inference measurement is linear or non-linear, and whether the mapping is static or dynamic. Since the power turbine input temperature T_{in} must be inferred from the directly measurable power turbine input pressure P_{in}, this relationship will determine the neural network topology. Figure 4.11 shows the input/output relationship of the power turbine input temperature T_{in} vs the power turbine input pressure P_{in} for the slow-varying part of the waveforms. Since this results in a distorted oval, it can be concluded that the relationship to be mapped is non-linear and dynamic, requiring a time-delayed feed-forward neural network with at least one hidden layer with non-linear activation functions.

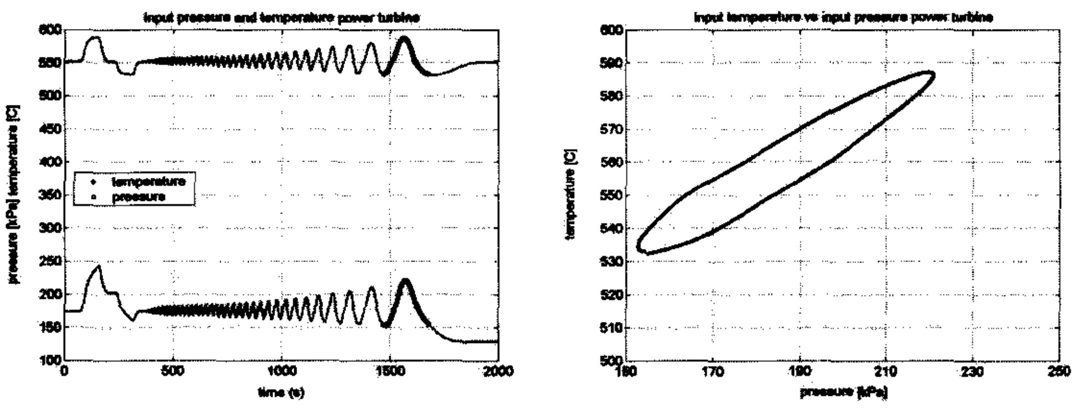

Figure 4.11: *Input pressure vs input temperature of power turbine for slow-varying part of waveform*

The step input at the first part of the mass injection/removal rate according to waveform n enables one to obtain the transient response of the output of a system due to the step input. In the first part of the waveforms in Figure 4.11 showing the power turbine input temperature T_{in} and the power turbine input pressure P_{in}, it can be seen that the step input is not reflected in either of the waveforms, therefore the number of input time delays cannot be determined explicitly. The relationship between the mass injection rate and power turbine inlet temperature is shown in Figure 4.12. In this figure it can be seen that the power turbine inlet temperature decays to a small value within 60 *s* due to step input of the mass injection/removal rate. This led to this number of time delays

$TD = \frac{timedecay}{\Delta T} = 60$ being chosen.

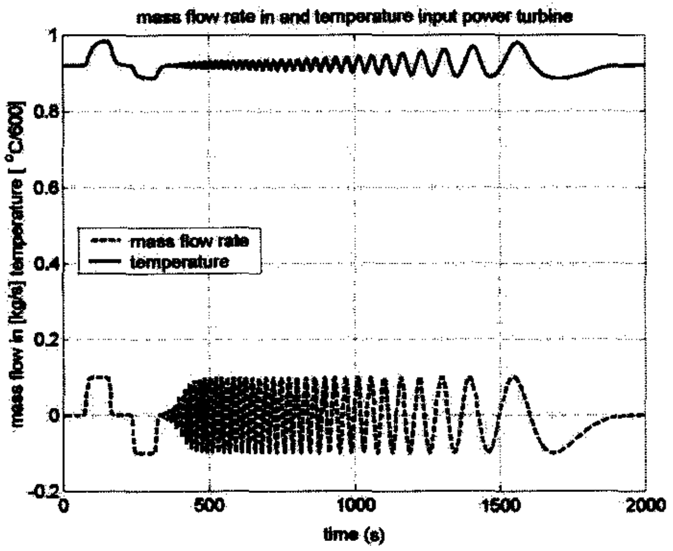

Figure 4.12: *Mass injection rate and resultant power turbine inlet temperature*

Neural network training

For neural network training the variables were normalised. The neural network was trained with the input $\frac{P_{in}}{300}$ and the target $\frac{T_{in}}{600}$. In order to predict whether accurate inference could be accomplished, the input waveform characteristics were compared by using the rate of change density function and the sum density function. As example, the rate of change density functions for the inputs of sets n and d, and the sum density functions of the inputs of sets n and k is shown in Figure 4.13. The percentage unmapped features (features not mapped by the training set input) of the input pressure waveforms are shown in Table 4.2. From the high percentage of unmapped features of the test set inputs, it is predicted that the power turbine input temperature will not be inferred accurately for data sets d and k, even if a small training error were found with input waveform n.

The neural network with 50 input time delays and 20 neurons in the single hidden layer was trained, using normalised data for inference of the power turbine inlet temperature from the power turbine inlet pressure. Figure 4.14 shows the time domain error of the neural network's inferred temperature for the different input waveforms. The mean square error (MSE) of the neural network and the output in the time domain in percentage of

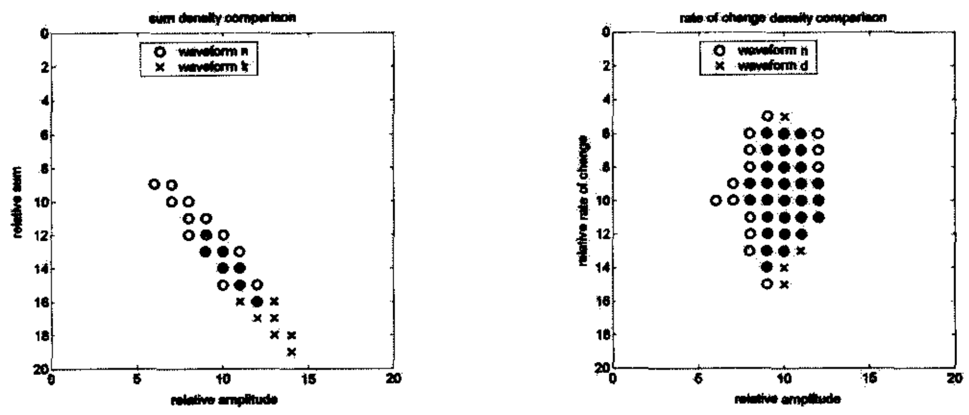

Figure 4.13: *Sum density functions and rate of change density functions for power turbine input pressure waveforms*

the maximum range of T_{in} is shown in Table 4.2.

Table 4.2: *MSE and maximum error of output for all input waveforms with set* n *as training set*

waveform	MSE	% max error	% unmapped rate of change density	% unmapped sum density
d	$1.1 * 10^{-4}$	27	12	9
k	$7.8 * 10^{-4}$	63	53	53
n	$1.5 * 10^{-5}$	25	0	0
p	$1.1 * 10^{-4}$	24	0	0

From Table 4.2 it can be seen that the neural network output error is large, even for the training input waveform. It was expected that the error for the training input waveform would be small, due to the apparently correct topology that was used and assuming that the training algorithm is capable of finding a local minimum with a sufficiently small error. The error due to input waveform p is approximately the same as for the training input, while the error due to input waveform k is larger than for the training input. These are expected results based on the comparison of waveform characteristics.

Due to the large neural network output error in this case study, further investigation revealed that two errors were made. By applying step inputs to the system, it was found that the time decay of the power turbine inlet temperature with a step change of the power input pressure was larger than 200 *s*. With only 50 input time delays used, the system

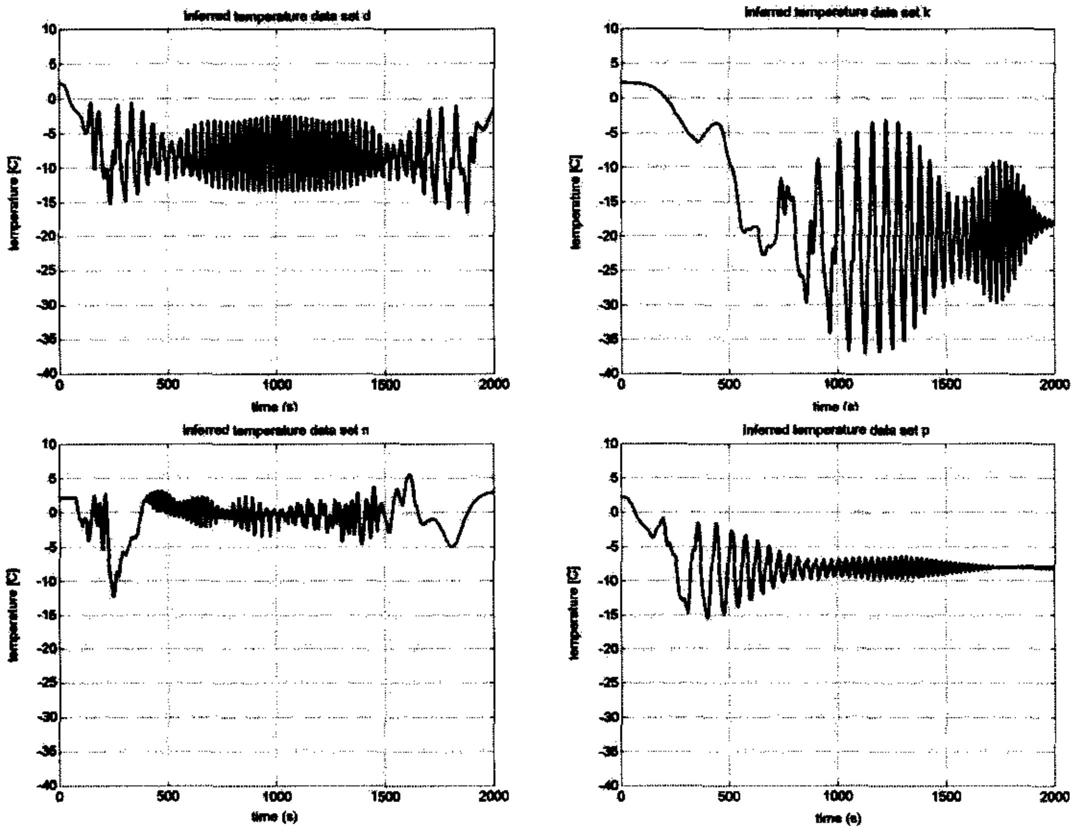

Figure 4.14: *Errors of inferred inlet temperatures of the power turbine*

could not be mapped accurately. For the thermodynamic cycle, the power turbine input pressure around a certain operating point is approximately proportional to the integral of the mass injection rate or $P_{in} \approx K \int \dot{m}\ dt$, where K is either constant for direct proportionality or a non-linear function for non-linear relationship. This can be verified by the triangular shape at the first part of the power turbine input pressure shown in Figure 4.11. This integration causes the pressures of the different data sets to operate at non-overlapping amplitudes. Because of this, the training input waveform (the inlet pressure of the power turbine) is not suitable for accurate inference with pressure inputs d or k. This was shown by compariing of the sum density functions and rate of change density functions as shown in Figure 4.13 and Table 4.2.

In the next section, the second case study shows how to ensure an accurate inference measurement of the temperature at the power turbine inlet by using the directly measurable pressure at the the power turbine inlet pressure.

4.5.2 PBMM: case study II

In order to accomplish accurate inference of the power turbine inlet temperature from the power turbine input pressure, sets of data were generated in order to overcome the shortcomings of the previous case study. A time step $\Delta T = 4\ s$ was used and the input pressure of the high-pressure compressor was controlled by changing the mass injection rate. The desired pressure waveforms are the same as shown in Figure 4.10, but scaled down as shown in Figure 4.16. Figure 4.15 shows the controller, while Figure 4.16 shows the output of the controller (the mass injection rate) for the high-pressure compressor inlet pressure. In Figure 4.18 it can be seen that the power turbine inlet pressure closely approximates a scaled version of the high-pressure compressor inlet pressure.

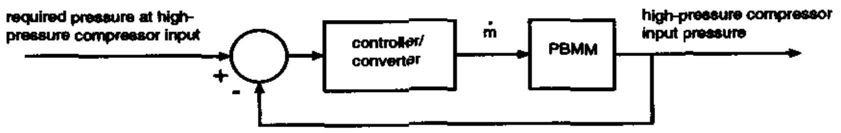
required pressure at high-pressure compressor input
+
-
controller/ converter
$\dot{m}$
PBMM
high-pressure compressor input pressure

Figure 4.15: *Controller for creating data sets*

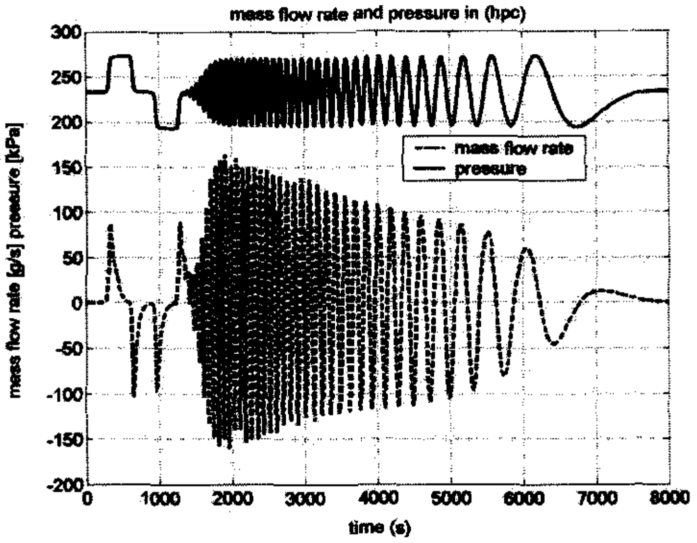

Figure 4.16: *Input pressure of high-pressure compressor and mass injection rate required for such a pressure waveform*

Neural network topology

When choosing the neural network topology, it is necessary to determine whether the required mapping is linear or non-linear, and whether the system is dynamic. Figure 4.17

shows the plot of the slow-varying part of both the power turbine inlet pressure and the power turbine inlet temperature. The distorted oval indicates that the relationship to be modelled is non-linear and dynamic. The required time delays for the neural network can be found when looking at Figure 4.18. This figure shows that the power turbine inlet temperature has decayed to an arbitrary small value within 240 s for a step change of the input, while the required number of time delays can be determined from $TD = \frac{time\ decay}{\Delta T} = 60$.

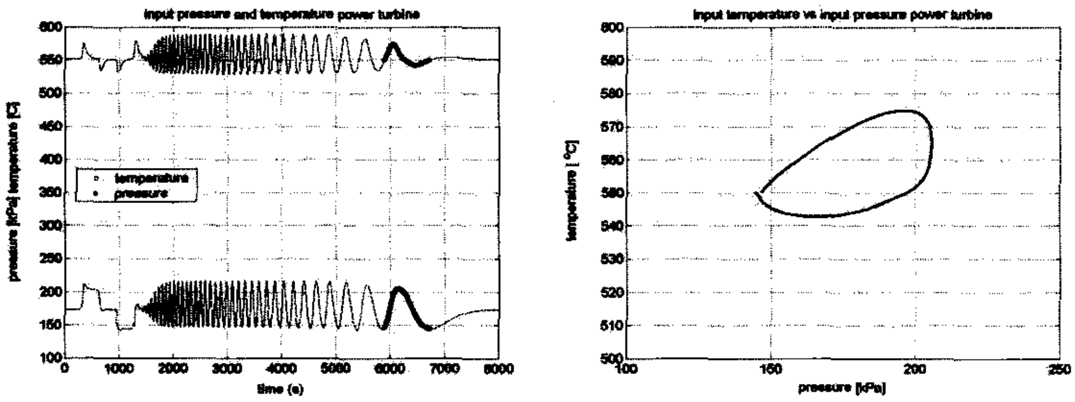

Figure 4.17: *Input pressure vs input temperature of power turbine of slow-varying part of waveform*

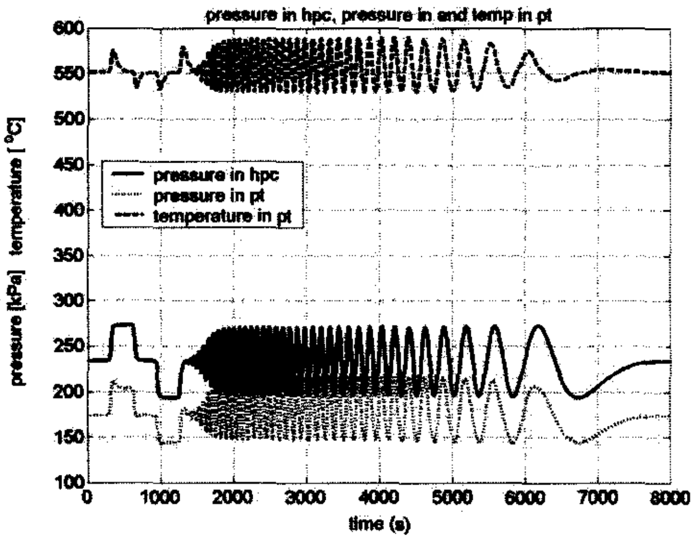

Figure 4.18: *Input pressure of high pressure compressor, input pressure of power turbine and input temperature of power turbine*

Neural network training

A time-delayed feed-forward neural network with 60 input time delays and 20 neurons in the single hidden layer was used to infer the power turbine input temperature T_{in} from the power input pressure P_{in}. When comparing the sum density function and rate of change density function of the input waveforms, it is predicted that the neural network will have a small output error for the training input waveform n. This is because the features of the test waveforms d, k and p are mapped by the features of waveform n. The training results in mean square error (MSE) and percentage maximum error is shown in Table 4.3.

Table 4.3: *MSE and maximum error of output for all input waveforms with set* n *as training set*

waveform	MSE	% max error	% unmapped rate of change density	% unmapped sum density
d	$1.2 * 10^{-6}$	8	0	0
k	$1.3 * 10^{-6}$	7	0	0
n	$4.5 * 10^{-7}$	5	0	0
p	$1.5 * 10^{-6}$	6	0	0

From Table 4.3 it can be seen that the error of the neural network output for a test input is approximately the same magnitude as the error due to the training input. The error is much smaller than was achieved with the previous case study. Assuming that the network topology is correct for this inference problem, a possible cause for the relatively large error of 5 % could be the finite accuracy by means of which the data was generated. From Figure 4.19 it can be seen that for a resolution of more than 10 bits, no improvement of the training errors were reached. It is thus concluded that, if all other factors influencing the training process were to be excluded, that the data was generated with a resolution of approximately 10 bits, and the inference accuracy being limited due to the finite resolution.

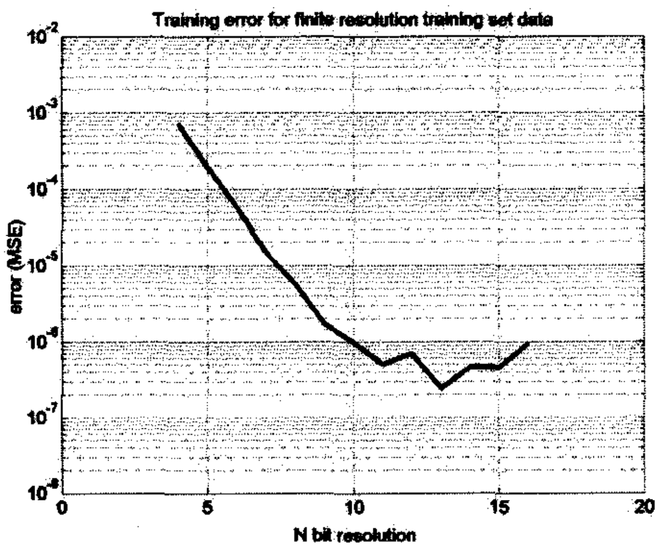

Figure 4.19: *Training error for inference of power turbine input error with finite resolution training set*

4.6 Discussion

For the simulated Brayton cycle, accurate inference measurements were achieved, using the methods developed in the previous chapters. This proves that the developed methods can be applied successfully for a general non-linear dynamic system.

The inference measurements on the PBMM, however, are less accurate. For the first case study, data was generated in such a way that the time delays could not be determined correctly. The neural network training input waveform also does not have the necessary characteristics to ensure accurate inference for the test input waveforms.

In order to improve the inference accuracy on the pebble bed micro model (PBMM), the inlet pressure of the high-pressure compressor was controlled according to the training input and test input waveforms. This enabled one to determine the number of time delays explicitly. It was predicted that the neural network output error with test inputs would be small, or at least as small as the neural network output error with training input. It was also shown that the accuracy is limited by the finite accuracy (in terms of resolution) of the data.

This chapter has shown that variables can be inferred accurately by using the methods developed in this thesis. Care should be taken to ensure that the dynamics are captured by observing the decay of the target with step change of the neural network input. The sum density functions and rate of change density functions enables one to predict whether an inference measurement would be accurate for a specfic test input.

Chapter 5

Conclusion

This aim of this thesis was to find a generalised method for inference measurements, using neural networks. There are several possible types of neural networks that could be used for inference, of which time-delayed feed-forward neural networks were used. Two major issues were addressed, namely the topological requirements for accurate inference and the characteristics of training input waveforms that would allow generalisation if the training algorithm is capable of finding a local minimum.

The topological requirements for accurate inference depend on the system characteristics. The dynamic response determines the number of time delays, since the dynamics of the system must be captured by the neural network. The number of time delays can be found explicitly from the time a system decays to an acceptable small value due to a step input. Should a time delay value larger than required be chosen, hence requiring more time delays, it could cause an unnecessary large neural network that requires longer training times and lends itself to overtraining.

The non-linearity in a system requires a layer with non-linear activation functions. Since the non-linearities are hidden, the number of non-linear neurons in the hidden layer cannot be found explicitly. Nevertheless, two methods were presented by means of which it can be established whether there are non-linearities. By means of this knowledge, a few training runs with variations in the number of non-linear activation functions would result in a choice of an optimum number of non-linear activation functions in the hidden layer.

The steps for finding the topology can be summarised as follows:

1. Determine the number of time delays based on the time response of the system.
2. Determine whether the system is linear or non-linear.
3. If the system is non-linear, find the optimum number of non-linear activation functions in the hidden layer with a few training runs.

In order to allow accurate inference, the training set should map the system sufficiently dense over a wider range than required by the specification to allow generalisation as judged by the test or validation sets. The training algorithm should be able to find a local minimum that is acceptably small. The sampling rate of the training sets for non-linear systems must be higher than depicted by the Nyquist criteria because of the generated harmonics and because the neural network trains on time domain patterns that must be represented accurately.

Assuming that the specification could be translated into excitation signals or test set inputs given as specification, favourable characteristics for the training sets based on the specifications could lead to training sets that would allow generalisation. Two methods to describe these characteristics were presented. The characteristics of the training excitation signal and test excitation signals in both the frequency and time domain were compared.

In the frequency domain the spectrums of the training and test excitation signals do give some indication whether the training set could result in generalisation of the trained neural network. Such a comparison is not reliable. However, comparing the characteristics of the training and test excitation signals in the time domain, has led to a reliable test. It could therefore be said whether the training set would lead to generalisation or not. Passing the test would lead to generalisation. If the training algorithm finds a small enough local minimum, it would pass the test. The test would not result in generalisation if this test was failed, irrespective of the attempts of the training algorithm.

The characteristics of neural network input waveforms were compared by using the sum density functions and the rate of change density functions. It was shown that it can be predicted whether accurate inference would be accomplished based on the features of a test signal not mapped by the features of the training set.

The following steps enable accurate inference for a test input waveform:

1. Compare the test and training set by using the sum density function and rate of change density function. With all the features of the test input waveform mapped by the features of the training input waveform, the error of the neural network output should be small.
2. If some features of a test input waveform were not mapped by the training input waveform, modify the training input waveform until all the features of the test input waveforms have been mapped.
3. Train the neural network by using the training set.

By using this generalised method for designing neural networks, it was shown that time-delayed feed-forward neural networks could be designed from the input/output relationship that must be mapped. A test input waveform can be compared with the training input waveform to predict whether the neural network would be trained for the test input waveform. This was successfully applied to a Brayton cycle and a simulation of the pebble bed micro model (PBMM), applying thermodynamic modelling software package used for its design.

Appendix A

Software implementation

This appendix discusses the implementation of some functions as was used in this thesis, and gives the list of MATLAB files included as a computer medium.

trainlm

`trainlm` is the implementation of the Levenberg-Marquardt backpropagation network training algorithm that updates weight and bias values. `trainlm` can train any network as long as its weight, net input, and transfer functions have derivative functions. At network creation, the training algorithm was set to `trainlm`. Executing `net = train(net,P,T)` trains the network with input `P` and target `T`.

newff

`newff` creates a feed-forward backpropagation network. For example, the command `net = newff([-1 1],[30 1],{'tansig' 'purelin'},'trainlm')` creates the network with input range $[-1, 1]$ with 30 sigmodial tangent neurons in the first layer and 1 linear neuron in the output layer and will use the Levenberg-Marquardt backpropagation network training algorithm.

pmtm

`pmtm` estimates the power spectral density (PSD) of the real time series `x` using the multitaper method (MTM).

ODE Solver

MATLAB implements the automatic step-size Runga-Kutta-Fehlberg integration method and has been used to obtain the solution of the state space equations. `ode45` was used, suggested as first option [53].

polyfit

`p = polyfit(x,y,n)` finds the coefficients of a polynomial `p(x)` of degree `n` that fits the data of `x` and `y` in the least square sense. For a linear fit, `n=1`.

fft

`Y = fft(X)` returns the discrete Fourier transform of vector `X`, computed with a fast Fourier transform (FFT) algorithm.

rand

`Y = rand(m,n)` returns an m-by-n matrix of random entries. `rand('state',sum(100*clock))` will reset the random seed generation to a different state each time.

Table A.1: *Files used for simulation and plots for* **Chapter 2: Network topology and system characteristics**

fsima.m	This file was used for the generation of the data sets for the neural network simulation of non-linear components characteristics.
fsimb.m	This file was used for the generation of the data sets for the neural network simulation of summation and difference. The following files are required: nonlba.m nonlbb.m nonlbc.m nonlbd.m nonlbe.m nonlbj.m
fsimbd.m	This file was used for the generation of the data sets for the neural network simulation of the phase lead circuit. The following files are required: difba.m difbb.m difbc.m difbd.m difbe.m
fsimbi.m	This file was used for the generation of the data sets for the neural network simulation of the phase lag circuit. The following files are required: lagba.m lagbb.m lagbc.m lagbd.m lagbe.m
fsimcr.m	This file was used for the generation of the data sets for the neural network simulation of the parallel resonant circuit. The following files are required: resba.m resbb.m resbc.m resbd.m resbe.m
fsimcs.m	This file was used for the generation of the data sets for the neural network simulation of the series resonant circuit. The following files are required: serba.m serbb.m serbc.m serbd.m serbe.m
fsimt.m	This file was used for the generation of the data sets of the phase lead, phase lag, parallel resonant and series resonant circuits for the plot of the transient responses. The following files are required: difbm.m lagbm.m resbm.m serbm.m
fsimc.m	This file was used for the generation of the data sets for the choice of topology for the inference of variables on a non-linear dynamic system. The following files are required: nonlca.m nonlcb.m nonlcc.m nonlcd.m nonlce.m nonlcf.m nonlcg.m nonlch.m nonlci.m nonlcj.m nonlck.m nonlcl.m nonlcm.m nonlcp.m nonlcq.m
fsimp.m	This file was used for the generation of the data sets for the choice of topology for the inference of variables on a dynamic system with non-linear storage element. The following files are required: nonlpa.m nonlpb.m nonlpc.m nonlpd.m nonlpe.m nonlpj.m nonlpm.m
fsimd.m	This file was used for the generation of the data sets for the choice of a topology for the inference of variables on a non-linear dynamic system for comparison with the inference of two variables using a single neural network. The following files are required: nonlda.m nonldb.m nonldc.m nonldd.m nonlde.m nonldj.m nonldm.m
fsimdm.m	This file was used for the generation of the data sets for the inference of variables on a non-linear dynamic system using a single neural network. The following files are required: nonlda.m nonldb.m nonldc.m nonldd.m nonlde.m nonldj.m nonldm.m

Table A.2: *Files used for simulation and plots for* **Chapter 3: Training, test and validation sets**

fsimc.m	This file was used for the simulation of data sets, the simulation and training of the neural networks, the implementation of the algorithms for comparing the signal characteristics and the plot of waveforms, spectra and signal characteristics. The following files are required: nonlca.m nonlcb.m nonlcc.m nonlcd.m nonlce.m nonlcf.m nonlcg.m nonlch.m nonlci.m nonlcj.m nonlck.m nonlcl.m nonlcm.m nonlcp.m nonlcq.m
fsimk.m	This file was used for the generation of the data sets for the inference of variables on a non-linear dynamic system with two inputs. The following files are required: nonlha.m nonlhd.m nonlhk.m nonlhl.m nonlhm.m nonlhn.m

Table A.3: *Files used for simulation and plots for* **Chapter 4: PBMR neural network inference**

fsimg.m	This file was used for the simulation of data sets, the simulation and training of the neural networks for inference measurements on the Brayton cycle. The following files are required: nonlgc.m nonlge.m nonlgh.m nonlgj.m nonlgm.m nonlgn.m
fsiml.m	This file was used for the reading of the data generated by using Flownet, and training of the simulated neural networks for inference measurements on the PBMM. The data for the different mass injection rate functions are in chirpd.mat chirpk.mat chirpn.mat chirpp.mat data.mat

Table A.4: *Files used for the plots of results*

plotall.m	The documented training results of **Chapter 2: Network topology and system characteristics** and **Chapter 3: Training, test and validation sets** are plotted by using this file.

Bibliography

[1] Ambrózy A., *Electronic Noise*, McGraw-Hill, New York, 1982.

[2] Cooper W. D., *Instrumentation and Measurement Techniques*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[3] Golding E. W., *Electrical Measurements and Measuring Instruments, Third Edition.*, Isaac Pitman, London, 1941.

[4] British Standards Institution, *Specification for Current Transformers: BS 3938*, British Standards Institutio., London, 1973.

[5] Ahmed N. and Natarajan T., *Discrete-Time Signals and Systems*, Reston Publishing, Reston, Virginia, 1983.

[6] McGhee J., Henderson I. A., and Sydenham P. H., "Sensor science - essentials for instrumentation and measurement technology," *Measurement*, , no. 25, pp. 89–113, 1999.

[7] Schalkoff R. J., *Artificial Neural Networks*, McGraw-Hill, New York, 1997.

[8] Sydenham P. H., Ed., *Handbook of Measurement Science, Volume 1: Theoretical Fundamentals*, John Wiley, 1982.

[9] Kuo B. C., *Automatic Control Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[10] Lago G. V. and Benningfield L. M., *Circuit and System Theory*, John Wiley, 1979.

[11] Takahashi Y., Rabins M. J., and Auslander D. M., *Control and Dynamic Systems*, Addison-Wesley, Reading, Massachusetts, 1970.

[12] Hecht-Nielsen R., *Neurocomputing*, Addison-Wesley, USA, 1989.

[13] Higham N. J., *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA., 1996.

[14] Schalkoff R. J., *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley, New York, 1992.

[15] Benítez J. M., Castro J. L., and Requena I., "Are artificial neural networks black boxes?," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, September 1997.

[16] Andrejević M. V. and Litovski V. B., "Nonlinear dynamic network modeling using artificial neural networks," Tech. Rep., Faculty of Engineering, University of Niš, Beogradska 14, 18000 Niš, Yugoslavia.

[17] Wagner M. G., Montague G. A., and Tham M. T., "Neural networks for the steady state modeling of an extruder," *Artificial Intelligence in Engineering*, , no. 11, pp. 375–382, 1997.

[18] Citterio C., Pelagotti A., and Piuri V., "Function approximation - a fast-convergence neural approach based on sprectral analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, July 1999.

[19] Tsoukalas L. H. and Uhrig R. E., *Fuzzy neural approaches in Engineering*, John Wiley, New York, 1997.

[20] Nelson M. M. and Illingworth W. T., *A Practical Guide to Neural Nets*, Addison-Wesley, Reading, Massachusetts, 1991.

[21] Taylor J. G., *Neural Networks*, Alfred Waller, Oxon, 1995.

[22] Sprecher D., "On the structure of continuous functions of several variables," *Transactions of the Americal Mathematical Society*, vol. 115, pp. 340–355, 1965.

[23] Cybenko G., "Approximations by superpositions of sigmoidal functions," *Mathematics of Control, Signals and Systems*, pp. 303–314, 1989.

[24] Stinchcombe M. and White H., "Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights," Department of Economics, U.C. San Diego.

[25] De Villiers J and Barnard E., "Backpropagation neural nets with one and two hidden layers," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 136–141, January 1993.

[26] Hornik K., Stinchcombe M., White H., and Auer P., "Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives," Tech. Rep., Royal Holloway University of London, 1995.

[27] Huang S-C. and Huang Y-F., "Bounds on the number of hidden neurons in multilayer perceptrons.," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 47–55, January 1991.

[28] Haykin S., *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, New Jersey, 1999.

[29] Rojas R., *Neural Networks: A Systematic Approach*, Springer-Verlag, Berlin, 1996.

[30] Rumelhart D. E, Hinton G. E., and Williams R. J., "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 9, pp. 533–536, October 1986.

[31] Hagan M. T. and Menjai M. B., "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, November 1994.

[32] Marquardt D. W., "An algorithm for least squares estimation of nonlinear parameters," *J. Soc. Indust. Appl. Math.*, vol. 11, no. 2, pp. 431–441, June 1963.

[33] Gori M. and Tesi A., "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, January 1992.

[34] Yu X-H., "Can backpropagation error surface not have local minima?," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 1019–1021, November 1992.

[35] Geman S., Bienenstock E., and Dourset R., "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[36] Ackermann D. W., "Current transformer measurements of distorted current waveforms," in *IEEE AFRICON'99*, Sept. 1999, Cape Town, South Africa.

[37] Ackermann D. W. and Case M. J., "The rogowski coil revisited," in *EPE2001*, Aug. 2001, Graz, Austria.

[38] Ackermann D. W. and Bodenstein C. P., "Accurate inference of variables using artificial neural networks," in *IEEE SMC'02*, Oct. 2002, Hammamet, Tunisia.

[39] Spiegel M. R., *Mathematical Handbook of Formulas and Tables*, McGraw-Hill, New York, 1968.

[40] Verkerk E. C. and Kikstra J. F., "Comparison of two models for a high temperature reactor coupled to a gas turbine," *Nuclear Engineering and Design*, , no. 220, pp. 51–65, 2003.

[41] M-Tech Industrial, *Flownet ver 5.9 Manual.*

[42] Kim T. S., Park H. J., and Ro S. T., "Characteristics of transient operation of a dual-pressure bottoming system for the combined cycle power plant," *Energy*, , no. 26, pp. 905–918, 2001.

[43] Shin J. Y., Jeon Y. L., Maeng D. J., Kim J. S., and Ro S. T., "Analysis of the dynamic characteristics of a combined-cycle power plant," *Energy*, , no. 27, pp. 1085–1098, 2002.

[44] Xie Z., Su M., and Weng S., "Extensible object model for gas turbine simulation," *Applied Thermal Engineering*, , no. 21, pp. 111–118, 2001.

[45] Huang F. F., *Engineering Thermodynamics*, MacMillan, New York, 1976.

[46] Eastop T. D. and McConkey A., *Applied Thermodynamics for Engineering Technologists, second edition*, Longman, London, 1970.

[47] Dixon S. L., *Fluid Mechanics, Thermodynamics of Turbomachinery, Third Edition*, Pergamon Press, 1978.

[48] Çengel Y. A. and Boles M. A., *Thermodynamics: An Engineering Approach*, McGraw-Hill, New York, 1989.

[49] Jakob M. and Hawkins G. A., *Elements of Heat Transfer, Third Edition*, John Wiley, 1957.

[50] PBMR, "Pbmr operability summary, document no.: 000574-187 rev. 2d (confidential)," Tech. Rep., PBMR, Centurion, South Africa, 2001.

[51] Greyvenstein G. P., *Options and Variable for Transient Events in Flownet*, M-Tech Industrial, 2001.

[52] Dercksen C. H., *Flownet Simulink Interface Description*, M-Tech Industrial, 2001.

[53] Ashino R., Nagase M., and Vaillancourt R., "Behind and beyond the matlab ode suite," *Computers and Mathematics with Applications*, , no. 40, pp. 491–512, 2000.