CALIBRATION TRANSFER METHODS FOR FEEDFORWARD NEURAL NETWORK BASED INSTRUMENTS

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfilment of the Requirements

for the Degree of Doctor of Philosophy

in the Department of Electrical Engineering

University of Saskatchewan

Saskatoon

by

Joseph John Cibere

April 2001

Abstract

The calibration transfer problem examined by this thesis is that of attempting to exploit the knowledge of an initial instrument calibration model so as to obtain a second similar model, having acceptable accuracy, with less data than that used to obtain the initial model. Specifically, this thesis considers instruments whose calibration model is based on a feedforward neural network, FFNN. The recalibration of FFNN based instruments has raised concerns regarding the significant quantity of data needed to perform their recalibration. Calibration transfer methods provide an alternative to recalibration which can reduce the data needed for a recalibration while maintaining acceptable levels of calibration accuracy.

Currently no reported methods of calibration transfer exist for the FFNN based instrument. This thesis develops and introduces a number of calibration transfer methods based on novel concepts and techniques to allow a recalibration using less data than that needed in a recalibration employing conventional backpropagation learning. Using the concept of a similarity map introduced in this thesis, a simple non-learning method of calibration transfer for an FFNN based instrument is introduced. A new calibration transfer method is also developed which is based on using a supervised learning algorithm employing a measure to learn the nth order partial derivatives of the desired calibration model evaluated at the coordinates given by the calibration data. Finally, a simple, but previously unreported idea of initialising the weights of an FFNN so as to begin learning from a point on the error surface that provides the approximation of a previously obtained calibration model is also described.

Using computer simulations, the calibration error associated with using these calibration transfer methods is compared to the error obtained from a standard recalibration using conventional backpropagation learning. The simulations included, changing the numbers of neurons, the number of calibration points, and the similarity between calibration models. The desired calibration models were selected from the classes of 8th order polynomials and bandlimited normal random processes. The simulations indicated that no one method of calibration transfer provides the least calibration error but it is possible to achieve a 2 to 1000 fold decrease in the median calibration error relative to that of the standard recalibration while also reducing the data by a factor of two. The simulation and subsequent analysis also revealed that it is difficult to predict whether a specific set of calibration conditions will achieve this reduction in calibration error.

In conclusion, this thesis introduces, develops, and evaluates a number of previously unavailable methods of calibration transfer for FFNN based instruments. It shows through computer simulations that these methods can achieve lower levels of calibration errors than that achievable from the standard recalibration method employing backpropagation learning.

Copyright

Acknowledgements

I would like to acknowledge the guidance and encouragement provided by my supervisor Dr. H. Wood in this difficult but enlightening thesis topic. I would like to also thank Dr. H. Wood for his financial support through his grants provided by the National Science and Engineering Research Council of Canada. This support, from April to July 1998, was greatly appreciated. In addition, I want to thank Dr. H. Wood for helping me establish contact with Dr. S. Hardy at Simon Fraser University, Burnaby, British Columbia. Completing this thesis while living in Burnaby was made less difficult by having access to both the resources and the faculty members of the school of Engineering and Computer Sciences at Simon Fraser University.

Finally, I would like to acknowledge the generosity of my brother, Anthony Cibere, who through his computer maintenance business, provided the numerous computers and peripherals that were used to complete the many thousands of simulation results reported in thesis.

Dedication

This thesis is dedicated to my family:

my wife Jolanda and our children.

David. Julia. and Samantha.

whose support and encouragement has made this work possible.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Acronyms and Nomenclature

BP	Backpropagation.
DS	Direct standarisation.
$D^{(n)}$-$iDACX$	$D^{(n)}$ Indirect additive calibration transfer.
$FFNN$	Feedforward neural network.
GSS	Generalised Shannon Sampling.
$GSSE$	Generalised sum squared error.
$iDACX$	Indirect additive calibration transfer.
LWR	Local weighted regression.
$MARS$	Multivariate adaptive regression splines.
NIR	Near-infrared.
NN	Neural network.
PDS	Piecewise direct standarisation.
PCA	Principal component analysis.
PCR	Principal component regression.
$PICX$	Prior initialisation calibration transfer.
PLS	Partial least squares.
PPR	Projection pursuit regression.
$RMSECV$	Root mean standard error of cross validation.
$RMSEP$	Root mean standard error of prediction.
SEP	Standard error of prediction.
SSE	Sum squared error.
Std. Cal.	Standard Calibration.

Symbols

a_i	ith input to a neuron.
e	True calibration error.
$\hat{e}$	Estimation of true calibration error.
e_a	Approximation error.
e_D	Error in data values.
e_e	Estimation error.
e_g	Generalisation error.
e_M	Error in slope values.
e_r	Error in using rth hint.
$\mathbf{f}_x$	Similarity map on the space of sensor outputs.
$\mathbf{f}_y$	Similarity map on the space of instrument responses.
$\mathbf{f}_z$	Similarity map on the space of input measurements.
$\mathbf{g}$	Ideal sensor system.
h	Ideal calibration model.
$\hat{h}$	Approximation of the ideal calibration model.
$\tilde{h}$	Approximation of the ideal calibration model using ΔH.
$\hat{h}^*$	Approximation of the ideal calibration model having the least e_g.
$\hat{h}^-$	An approximation of the ideal calibration model that may not have the least e_g.
m_k	Number of neurons in layer k.
m_t	Slope tuning parameter.
o_j	Output from the jth neuron.
s	Ideal similarity function.
$\hat{s}$	Approximation of ideal similarity function.
v_{jk}	kth type of net input to the jth neuron.
w_{ji}^k	Weight connecting the output of the ith neuron in layer $k-1$

	to the input of the jth neuron in layer k.
C	Cost or objective function used in FFNN supervised learning or the error surface used by FFNN supervised learning algorithm.
C^*	Value of global minima of C.
D_i	Finite calibration data set. for $i > 0$.
D_0	Complete infinite calibration data set.
$I(x)$	Identity function $x = I(x)$.
I_a	Information content of object a.
I_F	Improvement Factor.
L	Number of layers in a fully connected FFNN.
N	Number of points in a data set.
T	Test or validation data set.
$\mathcal{D}_0$	True complete calibration data set.
$\mathcal{T}$	Projected sensor output space.
$\mathcal{W}$	FFNN weight space.
$\mathcal{X}$	Sensor output space.
$\mathcal{Y}$	Instrument response space.
$\mathcal{Z}$	Measurement space.
$\mathbf{d}(n)$	Descent direction in $\mathcal{W}$ and on C at the nth descent step.
$\mathbf{w}(n)$	FFNN weight vector at the nth epoch of learning. $\mathbf{w} = [w_{ji}^k]$. $k = 1, 2, \ldots, L$, $i = 1, 2, \ldots m_{k-1}$, $j = 1, 2, \ldots, m_k$.
$\mathbf{w}^*$	FFNN weights vector providing $\hat{h}^*$.
$\mathbf{w}^-$	FFNN weight vector providing $\hat{h}^-$.
$\mathbf{x}$	Sensor output.
$\mathbf{y}$	Instrument output.
$\mathbf{z}$	Input measurement.
$\mathbf{H}$	Hessian matrix of C with respect to $\mathbf{w}$.
$\mathbf{I}$	Identity matrix.

$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$.
$\mathbf{A}^{-}$	Generalised inverse or pseudoinverse of matrix $\mathbf{A}$.
$\mathbf{g}$	Gradient vector of C with respect to $\mathbf{w}$.
η	Step size or learning rate parameter used in FFNN supervised learning.
η^*	Optimum step size.
ν_j	Activation state of the jth neuron.
o_j	Activation function of the jth neuron.
ΔG	Similarity map on the space of sensor systems.
ΔH	Similarity map on the space of calibration models.
$\Delta\mathbf{w}$	Change in the FFNN weight vector.
Φ_j	Output function of the jth neuron.
$\mathbb{N}$	The set of natural number $\{1, 2, 3, \ldots, \}$.
$\mathbb{R}$	The real line.
$\forall$	For all.
$\exists$	There exists.
$\in$	Element of.
$\rightarrow$	Approaches.
$\cap$	Intersection.
$\cup$	Union.
$\subset$	Subset.
$\emptyset$	Empty set.
$\overline{\overline{D}}$	Cardinality of a set D, that is the number of unique elements in a set.
$\{\cdot\}$	Set of elements.
$[\cdot]$	Closed interval or row vector or an array.
$(m \times n)$	Dimensions of a matrix having m rows and n columns.
$\mathbf{0}_{(m \times n)}$	A m row by n column matrix of elements having a value of zero.
$\mathbf{1}_{(m \times n)}$	A m row by n column matrix of elements having a value of one.

Typographic meaning of symbols

$\mathbf{x}$	Boldface lowercase symbols are column vectors.
$\mathbf{X}$	Boldface uppercase symbols are arrays.
X	Uppercase symbols are finite or infinite sets.
x	Lowercase symbols are functions or variables.
$\mathcal{X}$	Calligraphic symbols are vector or normed vector spaces, or operators.

Operators

$d(\cdot,\cdot)$	Distance function or distance metric.
d	Differentiation.
$< \cdot,\cdot >$	Inner or dot product.
$\nabla_{\mathbf{x}}$	Del. $[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_n}]^T$.
$(\nabla_{\mathbf{x}})^k$	$< \nabla_{\mathbf{x}}, \mathbf{1}_{(n\times 1)} >^k = \sum_{n_1=1}^{n} \sum_{n_2=1}^{n} \cdots \sum_{n_k=1}^{n} \frac{\partial^k}{\partial x_{n_1} \cdots \partial x_{n_k}}$.
$\nabla_{\mathbf{x}}^2$	Laplacian. $\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \ldots + \frac{\partial^2}{\partial x_n^2}$.
∂	Partial derivative.
$\circ$	Composite mapping operation.
$\times$	Cartesian product.
$(\cdot)^T$	Matrix transpose operator.
$\|\|\cdot\|\|$	norm.
l_p	l_p-norm. $\sqrt[p]{\sum_i x_i^p}$.
L_p	L_p-norm. $(\int_{x\in\mathcal{X}} \|f(x)\|^p dx)^{\frac{1}{p}}$.
$N(\mu,\sigma)$	Normal distribution having mean μ and variance σ.

1. Introduction

1.1 Overview

1.1.1 Background

The recalibration of analytical instruments is commonly viewed as an inevitable task that is needed to ensure that the level of uncertainty in an instrument's readings is within some set of bounds defined by an application. When the uncertainty in the instrument readings is suspect or known to be outside of these bounds, due to events such as component replacement or drifting of component characteristics, the instrument is recalibrated. Typically, the recalibration of instruments does not raise much of a concern until the cost, in terms of the time and resources needed to perform a recalibration, becomes a significant burden on the application's budget.

The cost burden due to calibration is influenced primarily by three factors associated with calibrating an instrument: the total number of calibration samples or data needed for a calibration, the frequency of calibration, and the cost per calibration sample. The cost per calibration sample, in particular, may hide a number of additional concerns that need to be considered such as the storage, transportation, preparation, collection, and processing of the samples. Estimating a value for these factors then provides a means of assessing both the total cost of calibration and the sensitivity of the application's budget to the cost of calibration.

It becomes clear, given an indispensable instrument with a significant calibration cost, that there is a strong desire to reduce any one or all of the factors that contribute to this cost: the total number of calibration samples, the frequency of calibration, or the cost per calibration sample. This desire has resulted in numerous approaches

that can potentially reduce the cost of calibration. Some of the more obvious approaches, for example in reducing the frequency of calibration, include improving the performance characteristic of the instrument with the use of more stable and accurate components and imposing tighter control on the factors known to influence the instrument response [1]. To reduce the cost per calibration sample, the logistics involved in sample handling represent another obvious approach.

There are also a number of approaches that can be used to reduce the number of calibration samples. These approaches can be seen as methods that attempt to solve the problem of excessive calibration data or alternatively, as attempts to use the data more efficiently. The premise used in applying these methods is that either data exist which do not contribute appreciably to the accuracy of calibration or that the data contain additional information that can be used to increase the accuracy of the calibration.

These data reduction methods can also be placed in one of two contexts. The first is the context of a single isolated calibration where it is desired to optimise the data usage for that calibration. The second context is that of multiple calibrations, either over time or across instruments or both. In this second context, the desire is to optimise the data usage over multiple calibrations. Though not explicitly described in this manner, much of the instrumentation literature concerning the problem referred to as the *calibration transfer problem* [2, 3] can be viewed as a problem in calibration using a small number of samples within the context of multiple calibrations.

The nature of the calibration transfer problem that is the focus of this thesis is that of multiple calibrations, specifically over two calibrations. In particular, the calibration problem examined in this thesis can be characterised as using the assumption that a first calibration results in an instrument having a relatively high degree of accuracy and whose calibration is obtained at great expense. This instrument calibration also results in producing, what is referred to as, an instrument calibration model. The calibration transfer problem examined by this thesis can then be described as

the problem associated with the attempt to exploit the knowledge of the first calibration model so as to obtain a second calibration model which has acceptable accuracy and which is obtained using less data than that used to obtain the first calibration model.

Calibration transfer methods have only recently been introduced into the instrumentation field. Most of these methods are based on linear or piecewise linear techniques where the instrument calibration model is also typically linear. Even more recent is the use of nonlinear multivariate calibration models, such as those provided by the static artificial feedforward neural network, FFNN. Methods of calibration transfer for these nonlinear models are clearly lacking in the literature. Therefore, this thesis partially fills this void in the literature by proposing a number of methods that can be used to perform a calibration transfer for the nonlinear FFNN calibration model.

1.1.2 Purpose

The purpose of this thesis is to propose and evaluate a number of approaches that can be used to reduce the data needed for the calibration, particularly the recalibration, of instruments whose calibration model is based on the static artificial FFNN. In the framework of the current instrumentation literature these proposed approaches represent novel calibration transfer methods for instruments based on a FFNN.

1.1.3 Objectives

The specific objective of this thesis is to determine if any of proposed methods of calibration transfer for instruments using a FFNN as their calibration model can achieve,

for a given number of calibration samples, an improvement in the calibration accuracy as compared to that achievable in a standard recalibration employing

conventional backpropagation learning.

1.1.4 Contributions

This thesis provides the following contributions to the field of instrumentation as it relates to calibration.

1. Introduces the indirect additive calibration transfer, iDACX, and the prior initialisation calibration transfer, PICX, methods for the FFNN based instrument.

2. Develops the $D^{(n)}$-iDACX calibration transfer methods for the FFNN based instrument. This work represents new development.

3. Introduces the concept of a similarity mapping, a unifying framework from which to view calibration transfer problems.

4. Introduces the generalised sum squared error, GSSE, as a measure of the error between calibration models over a finite set of points. The GSSE then forms the bases of the $D^{(n)}$ learning algorithms from which the $D^{(1)}$-iDACX methods are developed.

1.1.5 Chapter overview

The remainder of this chapter begins by first defining the instrument system and describing a number of instrument characteristics that are important to understand in calibration and calibration transfer. The calibration transfer problem is then described in terms of the previous notation introduced for the instrument system. This is followed by discussing the use of the FFNN in instrumentation application. An introduction to the FFNN based instrument calibration transfer problem is then presented. The chapter ends with a summary of the assumptions and limitations used in the thesis and provides a brief thesis outline.

1.2 Instrument Calibration and Concerns

To consider the calibration transfer problem as it applies to instruments with calibration models based on the FFNN it will be convenient to first introduce the appropriate mathematical notation and concept as it relates to calibration and recalibration. This notation will allow for the precise expression of the general problems and concerns in calibration and calibration transfer.

1.2.1 The instrument model and calibration

The concerns raised in calibration and the approaches used to address those concerns can be described concisely by viewing the instrument measurement process abstractly using the mathematical notion of a mapping or functional relationship. This abstraction of the instrument is shown in Figure (1.1).

The process of making an instrument measurement can be viewed as a process that transforms a measurement of a physical phenomenon, $\mathbf{z} = [z_1, z_2, \ldots, z_l]^T \in \mathcal{Z}$, to a desired output reading $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T \in \mathcal{Y}$, where the notation $\mathbf{z}$ refers to a column vector, $\mathbf{z}^T$ refers to a row vector, T is the transpose operator, $\mathcal{Z} \subset \mathbb{R}^l$ represents the input measurement space and $\mathcal{Y} \subset \mathbb{R}^m$ represents the instrument response space. To avoid complicating the discussion and without any loss of generality, the output reading will be confined to be a scalar quantity so that $y \in \mathcal{Y} \subset \mathbb{R}$.

To obtain an output reading, a measurement is typically made by physical devices, such as sensors, which sample the measurement space and produce an intermediate output $\mathbf{x}$ in the sensor output space, $\mathcal{X} \subset \mathbb{R}^n$, $\mathbf{x} \in \mathcal{X}$. The sensor operation can then be viewed as a vector mapping $\mathbf{g} : \mathcal{Z} \rightarrow \mathcal{X}$.

The sensor output $\mathbf{x}$ is typically not useful in its direct form and requires that another process, the calibration model, h, map $\mathbf{x}$ to a point y using $h : \mathcal{X} \rightarrow \mathcal{Y}$. The measurement operation can then be viewed as a composite map $h \circ \mathbf{g}$ or as the functional relationship $y = h(\mathbf{g}(\mathbf{z})) = (h \circ \mathbf{g})(\mathbf{z})$.

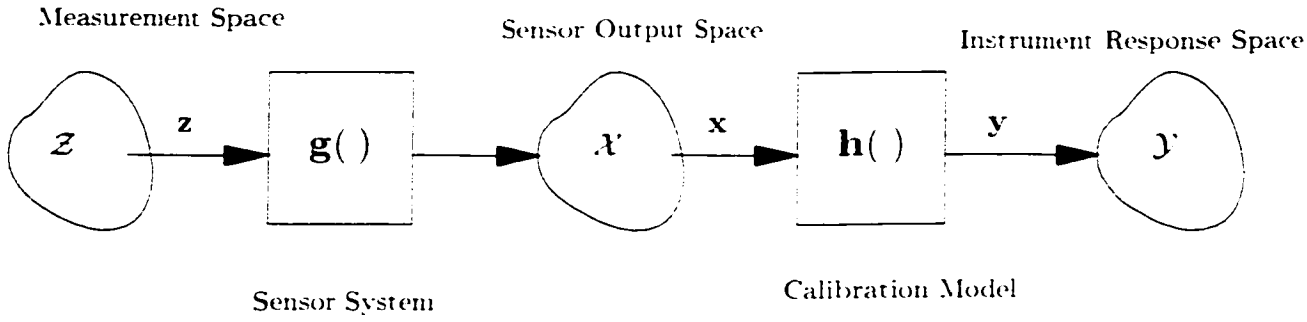


Figure 1.1. A block diagram of an ideal instrument system illustrating the process of taking a measurement $\mathbf{z}$ from $\mathcal{Z}$ and providing a corresponding output reading $\mathbf{y}$.

In many typical instrumentation applications, $\mathbf{g}$ is usually fixed by the sensing system. This implies that in order to obtain a useful reading y, the instrument designer must determine an appropriate form for h. Determining h is referred to as calibration [4] and consists of selecting a model, for example, a linear model, and then estimating the model parameters using, for example, linear regression.

The estimation of the parameters of the model relies on using a set of N calibration samples, $D = \{(\mathbf{g}(\mathbf{z}_i) = \mathbf{x}_i, y_i) : i = 1, 2, \ldots, N\}$, that exemplifies the function h. The set D is referred to as the calibration data set that can be obtained by the common practice of controlled sampling. Controlled sampling requires setting the input $\mathbf{z}$ to a series of known and fixed values so as to obtain a corresponding set of values for $\mathbf{x}$. These $\mathbf{x}$ values are then associated with a corresponding set of y values that themselves where obtained independently, typically from another more accurate, although more costly, instrument measurement of $\mathbf{z}$.

Determining the calibration model h using D obtained by controlled sampling is referred to as a controlled calibration [5]. Alternatively, a random calibration is also possible [5] where $\mathbf{x}$ is not under any direct control. The random calibration case will not be considered in this thesis.

1.2.2 Establishing the accuracy of the calibration

To attempt to reduce the number of calibration samples in D and still provide an acceptable calibration model requires a means to measure the error in the calibration model so as to determine whether it can be deemed acceptable. The acceptability of the calibration model is typically based on measuring the difference between a true or desired model h and its approximation $\hat{h}$ obtained after calibration.

This difference between h and $\hat{h}$ can be expressed using a number of measures. The most frequently reported measures are those based on a distance metric, $d(\cdot,\cdot)$, measuring the difference between two functions, defined as

$$\epsilon = d(h(\mathbf{x}), \hat{h}(\mathbf{x})) = \|h(\mathbf{x}) - \hat{h}(\mathbf{x})\|. \tag{1.1}$$

where ϵ is used to represent the error between the functions and $\|\cdot\|$ is the norm, commonly based on the weighted L_2-norm [6], so as to result in the error ϵ being defined as

$$= \left[\int_{\mathcal{X}} w(\mathbf{x})(h(\mathbf{x}) - \hat{h}(\mathbf{x}))^2 d\mathbf{x}\right]^{\frac{1}{2}}. \tag{1.2}$$

Here $w(\mathbf{x})$ is a kernel type weighting function typically representing the probability density function of $\mathbf{x}$. For the present discussion, $w(\mathbf{x})$ is assumed to be the constant function over $\mathcal{X}$ equal to one, or alternatively, representing a uniform distribution of $\mathbf{x}$ over $\mathcal{X}$.

The difficulty in using the measure given by Equation (1.2) is that in practice h is not known over all $\mathcal{X}$, but only at the values given by the calibration data set D. This requires an estimate $\hat{\epsilon}$ of the error ϵ to be made. Again, a number of measures are available to provide this estimate, a commonly cited measure is given by [4],

$$\hat{\epsilon} = \left[\frac{1}{N_J}\sum_{j=1}^{N_J}[h(\mathbf{x}_j) - \hat{h}(\mathbf{x}_j)]^2\right]^{\frac{1}{2}}. \tag{1.3}$$

where N_J is the number of samples used to determine $\hat{\epsilon}$. Since $\hat{\epsilon}$ is an estimate of ϵ, Equation (1.3) is also referred to as being an estimator of ϵ. The process of obtaining

this estimate is part of a larger non-trivial problem referred to as model validation [4] which has specific implications for calibration and later in calibration transfer.

To appreciate these implications of model validation, assume that a calibration data set D has been collected and a calibration has determined $\hat{h}$. To instill a degree of confidence in the accuracy of $\hat{h}$ requires obtaining $\hat{e}$. Model validation methods approach this problem typically by using a separate disjoint data set T, i.e. $T \cap D = \emptyset$, referred to as the test or validation data set. Pragmatically, this can be achieved by splitting D into two sets and redefining D to be one set and defining T as the other set.

The purpose in splitting the data set is to obtain an estimate $\hat{e}$ using a data set T that is independent of that used to obtain $\hat{h}$. This splitting of the data set results in the well known dilemma in the desire to obtain accuracy in both $\hat{e}$ and $\hat{h}$. To obtain an accurate estimate of $\hat{e}$ requires a large number of samples but this reduces the accuracy of $\hat{h}$ by virtue that there are fewer representative samples of h in D on which to base an approximation. On the other hand, using more samples in D to improve the accuracy of $\hat{h}$ reduces the number of samples in T, hence the accuracy in $\hat{e}$ is reduced. This dilemma can be resolved using a number of methods such as cross-validation [7, 8] which cleverly uses the entire data set to obtain $\hat{e}$ without sacrificing the data used to obtain $\hat{h}$.

When the error is estimated by cross-validation, using an expression similar to that given by Equation (1.3), it is referred to as the root mean standard error of cross validation, RMSECV. If a truly independent test set is used, the error estimate, using Equation (1.3), is referred to as the root mean standard error of prediction, RMSEP [4].

The temptation of naively using all the data for calibration not only brings into question any estimate of $\hat{e}$ but it may also falsely indicate low error estimates. This result is possible, especially for specific types of calibration models such as the FFNN, when $\hat{h}$ is given great flexibility in its approximation capabilities, a characteristic re-

ferred to as over-parameterisation of the model, or an overly complex model [4]. In general, increasing this approximation flexibility in $\hat{h}$ allows it to approach an arbitrarily small error at the data points in D, that is, $\|h(\mathbf{x}_i) - \hat{h}(\mathbf{x}_i)\| \to 0, \forall(\mathbf{x}_i, h(\mathbf{x}_i)) \in D$. If this low error is approached by increasing the approximation flexibility of $\hat{h}$ then this will increase the likelihood of greater errors for $(\mathbf{x}_i, h(\mathbf{x}_i)) \notin D$ [9], a phenomenon often described as over-fitting [4, 9] or under-smoothing [10].

Finally, it should be noted that though the ideal calibration goal is to minimise $\hat{e}$ using an appropriate $\hat{h}$, pragmatically, that goal is not typically realised. Instead, the application sets some upper bound for the estimate of the error and if any $\hat{h}$ has an associated $\hat{e}$ below this bound, then it is considered acceptable.

1.2.3 The need for recalibration

A number of factors, as outlined in Appendix A, influence the accuracy of calibrations. A few of the obvious factors include, data errors, model selection, data selection, and domain representation. All these factors will obviously also affect the accuracy of recalibration. Additional considerations in recalibration result from those factors that affect the error e or $\hat{e}$ after a calibration has occurred. One of these factors is the unpredictable variance of $\mathbf{g}$ either over time or across instruments.

In performing a calibration, the tacit assumption is that $\mathbf{g}$ does not vary during the course of the calibration and, more importantly, does not vary for some period of time after the calibration. Inevitably, $\mathbf{g}$ will change, thereby increasing the errors in y by virtue that $\hat{h} \circ \mathbf{g}$ has changed and will no longer adequately map $\mathbf{z}$ to y. To more precisely refer to these cases, the notation $\mathbf{g}_k$, h_k, and D_k will be used to represent the kth realisation of the sensor system, the calibration model, and the calibration data set, respectively, and where k can represent a specific time interval or instrument.

The lack of invariance of $\mathbf{g}$ is a problem that is especially restricting for multivariate calibration. This restriction results from the cost burden of calibration on the application and occurs primarily in conjunction with a high cost per sample and the

increased dimensionality of $\mathcal{X}$. Of course, any of the factors affecting calibration can also increase the need for data samples and will also contribute to the cost burden.

Again, the pragmatic approach to counter the variation in $\mathbf{g}$ over time is to physically reduce the variation or to account for it explicitly in $\mathbf{g}$. Explicitly accounting for the variation in $\mathbf{g}$ can be accomplished by adding a sensing element that measures the degree of the influence causing the variation. For example, if the sensor system $\mathbf{g}$ is particularly sensitive to temperature and the environment is thermally volatile, then the addition of a temperature sensor will, in principle, allow the calibration model $\hat{h}$ to account for the temperature variation. The drawback with this approach is that the dimensionality of $\mathcal{X}$ is increased, thereby increasing the number of calibration data samples needed to determine an acceptable $\hat{h}$.

The variation of $\mathbf{g}$ over instruments is also a common occurrence due to instrument components which are not identical. Obviously, one approach to reduce the source of this variation is to use instrument components which not only exhibit small variances in their characteristics but also share equally small variances in their sensitivities to influencing factors, such as temperature, humidity, and age. Components with these attributes are typically very costly and not conducive to mass instrument production. Another approach to reduce the variation of component characteristics is to simply avoid exposing the instrument to the factors causing the variations, but this option can often be impractical or very costly.

In other instances the variation in $\mathbf{g}$ is due to the more practical issue of component replacement resulting from failure or due to unacceptable component operating characteristics.

The variation in $\mathbf{g}$ must also be detected and this detection can present a nontrivial and serious problem in deciding when to recalibrate. This problem in deciding when to recalibrate will not be examined in this thesis. Instead, it will simply be assumed that a decision to recalibrate has been made by some means. Appendix A presents a brief discussion on the nature of this problem.

1.3 The Calibration Transfer Problem

In cases where it is not practical or economical to measure or control the source of the variation in **g**, or where the source of the variation is not known, a recalibration is needed to restore the calibration accuracy of the instrument. Then, if the recalibration is of a multivariate instrument having a high cost per calibration sample, recalibration will impose a severe restriction in using the instrument in that the cost burden of calibration threatens the economic viability of the instrument. This provides a strong motivation to seek alternative approaches to a standard recalibration. Though, as discussed previously, there are a number of specific approaches that can be used to reduce either the need or data for recalibration, the approach that is of interest in this thesis is that of calibration transfer.

In this section the problems in calibration transfer are described in terms of the instrument model shown in Figure (1.1). The less treated topic of calibration transfer for instruments using a nonparametric calibration model, such as the FFNN, is then discussed briefly. This leads into the motivation for the use of a FFNN as a calibration model and finally into the specific concerns and problems of calibration transfer for FFNN based instruments.

1.3.1 Describing the problem

Given that a variation in the sensor system **g** has resulted in an unacceptable level of calibration error, the task facing the instrument user is to recalibrate the instrument so as to have an acceptable level of calibration error. The primary concern in performing a recalibration is assumed to be the cost associated with the recalibration. It is further assumed that the cost per calibration sample cannot be altered and, therefore, the only means of reducing that cost of recalibration, besides not performing the recalibration, is to attempt to use less calibration data for the recalibration. Again, to clearly distinguish the second calibration, or recalibration, from the first or initial calibration, the notation $\mathbf{g}_l$, h_l, and D_l will be used to represent the lth

realisation of the sensor system, the calibration model, and the calibration data set, respectively. In this setting, the attempt to use less data for the recalibration, or the lth calibration, is the attempt to have $\overline{\overline{D}}_l < \overline{\overline{D}}_k$, where the over set symbol, $\overline{\overline{(\cdot)}}$, represents the cardinality, or the number of elements, in the set.

Attempting to use less calibration data than what was used for the initial calibration, known to provide an acceptable error, appears to be doomed to fail. This failure is the consequence of the error in the approximation $\hat{h}$ having a tendency to increase with a decrease in data samples. A calibration transfer approach is an attempt to counter this tendency, when using less data, by exploiting the knowledge of the calibration model h_k by *transferring* the model so as to become h_l or alternatively, *transferring* the data obtained from $\mathbf{g}_l$, that is, $\mathbf{x}(l)$, so as to be able to use it with h_k. Here the notation $\mathbf{x}(l)$ is used to indicate that $\mathbf{x}$ is dependent on the lth realisation of the sensor system.

The term calibration transfer appears to have been the result of early attempts to transfer a calibration model from a very accurately and expensively calibrated instrument to another instrument. The intent has been to only perform one expensive calibration and to then use the corresponding calibration model to either obtain additional calibration models for other instruments or to obtain a more accurate calibration model for the same instrument.

To more precisely describe the calibration transfer problem, assume that the true association between $\mathcal{Z}$ and $\mathcal{Y}$ is given by the map $f = h \circ \mathbf{g} : \mathbf{z} \to y$. It is also assumed that this mapping is fixed over all time and instrument realisations. The true *complete* calibration data set is then defined by $\mathcal{D}_0 = \mathcal{Z} \times \mathcal{Y} = \{(\mathbf{z}, f(\mathbf{z}) = y) : \mathbf{z} \in \mathcal{Z}, y \in \mathcal{Y}\}$.

The ideal instrument system then provides a kth realisation of this true association between $\mathbf{z}$ and y, using $h_k \circ \mathbf{g}_k : \mathbf{z} \to y$. Pragmatically, only an approximation of this true association is possible and for this thesis this approximation is given by the map $\hat{h}_k \circ \mathbf{g}_k$. Therefore, the standard calibration problem is to determine $\hat{h}_k$ using a finite calibration data set $D_k = \{(\mathbf{g}_k(\mathbf{z}_i) = \mathbf{x}_i(k), h_k(\mathbf{x}_i(k))) : i = 1, 2, \ldots, N_k\}$.

Given a change in the sensor system from $\mathbf{g}_k$ to $\mathbf{g}_l$ the ideal instrument will need to provide the lth realisation of the true association given by the map $h_l \circ \mathbf{g}_l = h_k \circ \mathbf{g}_k$. By definition, $\mathbf{g}_l$ results in the approximation of the true association having unacceptable errors, that is, the map $\hat{h}_k \circ \mathbf{g}_l$ has unacceptable errors. A recalibration is needed to obtain the approximation $\hat{h}_l$ using the new calibration data set $D_l = \{(\mathbf{g}_l(\mathbf{z}_i) = \mathbf{x}_i(l), h_l(\mathbf{x}_i(l))) : i = 1, 2, \ldots, N_l\}$.

The calibration transfer problem can then be described generally as the problem associated with the attempt to exploit an accurate, and possibly expensively obtained, calibration model $\hat{h}_k$ so as to obtain another calibration model $\hat{h}_l$ having acceptable accuracy using less data than that used to obtain $\hat{h}_k$, that is, to have $\bar{\bar{D}}_l < \bar{\bar{D}}_k$.

Many of the current methods of calibration transfer are based on linear or piecewise linear techniques where the instrument calibration model $\hat{h}$ is also assumed to be linear. These linear methods, as outline in Appendix B, have been shown to achieve a transfer to $\hat{h}_l$ that has acceptable accuracy with a reduction in the number of calibration samples from that used to obtain $\hat{h}_k$. The more recent use of nonlinear, nonparametric, multivariate calibration models, such as those provided by the FFNN have no reported methods analogous to calibration transfer.

1.3.2 Characteristics and concerns in nonparametric calibration transfer

The lack of attention in the literature regarding calibration transfer for multivariate, nonparametric, calibration models is not surprising when some of the more predominant characteristics exhibited by nonparametric methods are considered. In particular, two characteristics, the multivariate nature of the model and its nonparametric approximation capabilities, could demand, as discussed in Appendix A, a significant amount of calibration data. This demand for a significant amount of data tends to portray the use of nonparametric methods in high dimensions as very unfavourable and commonly results in citing the curse-of-dimensionality as a reason for avoiding the use of nonparametric methods in multivariate approximations [11].

However, this is not the case, since recently, the use of nonlinear models, obtained with nonparametric methods, has been investigated as a means to determine $\hat{h}$ [12, 13]. These nonlinear models have been shown, given a fixed number of samples and a clearly nonlinear h, to provide greater accuracy than linear models [12, 13].

In spite of this increased accuracy, nonparametric methods do raise concerns regarding implementation complexity and data needs. These concerns have also been raised by Carey *et al.* [13] in comparing both linear and nonparametric methods in a clearly nonlinear problem. Specifically, Carey *et al.* compared linear methods against the nonparametric techniques of multivariate adaptive regression splines, MARS [14], and projection pursuit regression, PPR [15], in the calibration of nonlinear solid-state sensor arrays. In their investigation Carey *et al.* noted that the nonparametric methods, though providing more than twice the accuracy of the linear methods, require trading-off the needed calibration accuracy with the resources needed to perform the calibration. These resources are primarily affected by the complexity of the calibration process and the quantity of data needed to obtain acceptable accuracy. Consequently, Carey *et al.* convey the impression that there is a strong desire to use nonlinear models but that the complexity and the data requirements associatd with using these models present serious obstacles that need to be addressed.

One specific concern regarding the calibration transfer problem in nonparametric calibration models was also briefly noted by Naes *et al.* [16] in their conclusion regarding the use of a new nonparametric method called local weighted regression, LWR [11]. Naes *et al.* not only claimed that LWR is simpler to use than other nonparametric methods such as PPR but, more importantly, LWR appears to be less sensitive to instrument variations, that is variations in $\mathbf{g}$, than linear methods based on using PCR and PLS. This insensitivity was attributed to LWR's use of only the first few principal components of the calibration data set to form a nonparametric approximation. This is in direct contrast to linear methods based on using PCR and PLS that tend to be used with a number of components, as determined by the user, which account for most of the apparent relevant behaviour of $\mathbf{g}$ over $\mathcal{X}$.

It appears that LWR's nonlinear approximation capabilities allow it to obtain an approximation $\hat{h}$ using fewer principal components than linear methods. It may also be the case that the use of only a few principal components from a dimensionally reduced $\mathcal{X}$ is more effective in removing information that is due to *irrelevant variations* in $\mathbf{g}$, regardless of the fact that the use of more components provides a better account of the behaviour of $\mathbf{g}$ over $\mathcal{X}$. These observations regarding LWR suggest that other nonparametric methods may be able to obtain approximations using fewer principal components than the linear methods, especially if the underlying relationship is highly nonlinear. If this suggestion were true, it would tend to counter the undesirable nonparametric characteristic of requiring more data for an approximation by being able to operate in a lower dimensional space. Of course, the actual data characteristics, which describe the distribution of *relevant variations* of $\mathbf{g}$ over $\mathcal{X}$, would determine the degree of dimensional reduction that is obtainable.

These observations suggest that nonparametric methods are not to be avoided in the case of highly nonlinear $\mathbf{g}$ and, in addition to providing greater accuracy than linear methods, may in fact be more conducive to calibration transfer. This then raises the question of whether other nonparametric methods, such as those based on using a FFNN, are, or can be made to be conducive to calibration transfer.

1.3.3 The need for FFNN calibration transfer

Motivation in using a FFNN calibration model

Very recently, the FFNN has been investigated as an alternative nonparametric method in a wide range of instrumentation applications including predicting product quality [17], measuring the moisture content of processed materials such as coal [18] or wheat [19], measuring the efficiency of suspended solid separation in a fluid [20], measuring the concentration of ambient gases in multicomponent mixtures using either an array of nonselective semiconductor sensors [21–26], or an array of infrared or ultraviolet optical sensors [27–30], measuring and detecting specific odours [31, 32], and numerous other applications [33–39].

The motivation for these investigations are not without theoretical or empirical support. Theoretically, the FFNN has been shown to be a universal approximator in that it is capable of approximating any continuous functional relationship to any required degree of accuracy [40–42]. Though there is a recognised gap between the theoretical and the pragmatic results [9], the FFNN has been shown empirically to provide an accuracy that is better than linear methods and is comparable to or better than other nonlinear methods in many diverse instrument applications [12, 43–45]. In addition, the FFNN has been shown to be free from the dreaded curse-of-dimensionality problem [46, 47], which other nonparametric methods are susceptible to, such as those using local averaging in high dimensional spaces that include kernel methods and spline smoothing [48]. It should be noted that FFNN is not complete free from the curse-of-dimensionality in that conditions on the differentiability of the underlying function to be approximated need to be satisfied ([47], pg 114-119).

An additional motivation in selecting the FFNN for instrumentation applications appears to be its repeated characterisation of possessing the capability of *learning*, thus implying a degree of cognitive superiority over other conventional non-FFNN approximation methods. Though, as described in Chapter 3, FFNN learning can be viewed as a nonlinear optimisation process, characterising the FFNN as being capable of learning attracts attention. Relevant to this thesis, is the attention given by a particular group of instrumentation applications: those having relationships between the measurand and instrument reading that are typified with expressions such as, "being difficult to express analytically" [19, 34, 39, 49], or simply as "being complex, nonlinear, or multivariate" [34, 38]. The successful use of the FFNN in these applications has further characterised the FFNN as a method that is able to handle these difficult and complex application and is relatively simple to apply, when compared to other nonparametric methods. These statement are also supported by the reports from instrumentation researchers who frequently cite these characteristics as a factor in their decision in selecting the FFNN for their investigations [19, 20, 34, 37–39, 49, 50]

Finally, the conclusions of many of these investigations provide another motivating factor by conveying a sense of optimism regarding the use of the FFNN in instrument application: the FFNN represents a viable alternative as a calibration model and has great promise in instrument applications. Though this optimism is clearly present in the conclusion, it is usually tempered by a caveat: that due attention must be paid in implementing the FFNN to avoid its known limitations, which are discussed in Chapter 3.

Characteristics and Concerns in FFNN calibration transfer

Though the FFNN is a popular method for use in many instrumentation investigations, it is still a nonparametric technique. Therefore, it seems reasonable to expect that it will exhibit the nonparametric characteristics that raise a concern in both the cost of calibration and in the calibration transfer problem. As expected, these concerns have been raised by others [51, 52]. In particular, Kermani *et al.* [52] have noted a concern in their investigation that mainly focuses on reducing the dimensionality of a sensor array in an electronic nose. In briefly discussing some practical issues regarding the use of the instrument, Kermani *et al.* point out that it is crucial that the response of replacement sensors be consistent with the original sensor in order for the neural network to provide repeatable results. However, in practice the response of replacement sensors is always different enough to require retraining, i.e. instrument recalibration, using a new set of calibration data. Kermani *et al.* then go on to state that:

> ... Optimal strategies for the calibration of sensor arrays and their impact on NN [neural networks] training/retraining is an important subject, ... **that requires further study**. [emphasis added]

It is evident that the FFNN is playing a role in instrumentation applications but that it exhibits undesirable nonparametric characteristics and has raised concerns regarding calibration transfer. It is also clear, by virtue of the continuing investigations, that there are advantages in using a FFNN in these complex, nonlinear, and

multivariate applications. Though the FFNN has been shown to be capable of obtaining an approximation of these complex relationships with a greater accuracy and with relatively less effort than conventional approximation methods, its requirement for data and the unaddressed problem of calibration transfer threatens the economic viability of the instrument. It therefore seems appropriate to address this problem and provide a solution that can increase the likelihood of the economic viability of instruments using FFNN as their calibration model.

1.4 Assumptions and Limitations

Throughout the discussion in Chapter 1, a number of assumptions regarding the calibration transfer for a FFNN based instrument were stated. Additional assumptions will be made in the remainder of the thesis. For convenience these assumptions are listed here.

In addition to these assumptions, the limitations and scope of the thesis are also enumerated. A number of these items, in either list, are treated or discussed in later chapters but are listed here to provide a sense of the direction taken by the thesis investigation.

1. Assumed calibration conditions.

   (a) The cost of calibration is a significant burden.

   (b) The true or desired calibration model h cannot be adequately described using a linear model.

   (c) An acceptable initial calibration model $\hat{h}_k$ has been obtained.

   (d) The calibration model $\hat{h}_k$ is implemented with a FFNN.

   (e) The true association between $\mathcal{Z}$ and $\mathcal{Y}$ is given by the map $f = h \circ \mathbf{g} : \mathbf{z} \rightarrow y$ that is assumed to be fixed over all time and instrument realisations.

   (f) The kth realisation of the sensor system, represented by $\mathbf{g}_k$, changes in an unpredictable way. This changed state is denoted as $\mathbf{g}_l$.

(g) The sources contributing to the variation in $\mathbf{g}$ are unknown. Alternatively, if the sources of the variation are known, it is not economically viable to directly measure or to estimate these variations.

(h) The decision to recalibrate has been made by some means.

(i) The calibration or data samples used for the kth and lth calibration have negligible error and are obtained using a controlled calibration.

2. Limitations and scope of the thesis

(a) The calibration transfer methods to be proposed by this thesis are limited to the following approaches:

i. An approach based on the indirect similarity map, $\Delta H : \hat{h}_k \rightarrow \hat{h}_l$, that is implemented using $\Delta H(\hat{h}_l) = \hat{h}_k + \hat{s}_l$. This implementation is realised using two methods.

A. A method, referred to as the indirect additive calibration transfer, iDACX, is introduced. This method implements ΔH by summing $\hat{h}_k$, realised by the FFNN, and $\hat{s}_l$, which is obtained using conventional approximation methods.

B. A method, referred to as the $D^{(n)}$ indirect additive calibration transfer, $D^{(n)}$-iDACX, is developed. In this method ΔH is implemented with the supervised learning algorithm used for the FFNN. This supervised learning algorithm uses the generalised sum squared error, GSSE, measure that is limited to incorporating the derivatives, up to order $N_d = 1$ with respect to $\mathbf{x}$, of $\hat{h}_k$ and $\hat{s}_l$, where $\hat{s}_l$ is obtained using conventional approximation methods.

ii. An approach based on a prior initialisation calibration transfer, PICX. This approach initialises the weights of the FFNN to the values that provided the approximation of a prior calibration model, that is, $\hat{h}_k$. This approach is applied to the standard calibration employing back-

propagation learning and $D^{(0)}$ learning. This approach is also used in conjunction with the $D^{(1)}$-iDACX methods.

(b) A hypothetical instrument calibration model will be used. It is not based on an instrument of any particular class or characteristics.

(c) The true underlying calibration model h has no restrictions, other than continuity and a finite *bandwidth*, placed on it. All realisation of h can be essentially considered a bandlimited random process $h(\mathbf{x}, l)$ over $\mathbf{x} \in \mathcal{X}$ and realisation $l = 1, 2, \ldots$.

(d) The simulations are limited to the univariate calibration model, that is, $x \in \mathcal{X} \subset \mathbb{R}$.

(e) The true underlying calibration model h is known over all $\mathcal{X}$ and for all realisations. This is simply intended to allow an accurate estimate of the calibration error without the added complexity of cross-validation. This information is not used to assist in the calibration transfer.

1.5 Thesis Outline

The remainder of the thesis presents the development and results of the FFNN calibration transfer methods. This presentation begins with Chapter 2 where the methods of calibration transfer and the prior and related literature are reviewed. The chapter also introduces the concept of the similarity map from which the FFNN calibration transfer method is introduced. Chapter 3 presents the development of the various FFNN calibration transfer methods. Chapter 4 and Chapter 5 outline and present the methods and results of the calibration transfer simulations. The results of the simulations are discussed in Chapter 6, with concluding remarks provided in Chapter 7. Numerous appendices are provided to support the development and ideas presented in this thesis.

2. Calibration Transfer

2.1 Overview

This chapter begins by reviewing the calibration transfer problem from a historical perspective where, from this historical vantage point, it is seen how the calibration transfer problem arose from the need to recalibrate linear multivariate instruments. This historical perspective also reveals, as was shown in Section 1.3, the lack of attention given to the calibration transfer problem for the nonlinear, nonparametric, multivariate calibration model.

To approach and provide attention to the calibration transfer problem for the nonlinear calibration model, this chapter introduces the notion of a similarity map. The similarity map provides a convenient framework from which to view and examine linear and nonlinear calibration transfer methods. The assumptions and limitations in using the notion of a similarity map are then outlined. The utility in using the similarity map is then demonstrated by reviewing a number of reported calibration transfer methods for linear calibration models and placing them within the framework of a similarity map.

Finally, the FFNN calibration transfer problem is introduced in terms of this similarity map. An approach to solve the FFNN calibration problem is then proposed and a specific methodology to implement the proposed approach is outlined and described. The results from other neural network researchers are then examined and shown to support the approach proposed to solve the FFNN calibration transfer problem.

2.1.1 Historical perspective of calibration transfer

Though the desire to reduce the data for a calibration has always been present, it was the paper by Wang *et al.* [2] that introduced a number of approaches that allowed reducing the data for a calibration by a process described as *transferring calibrations* from one instrument to another, specifically from one spectrometer to another. Wang *et al.* referred to this process of transferring a calibration as an "instrument standardization" and did not refer to the process as a calibration transfer or the problems associated with this process as a calibration transfer problem. It was a paper by Adihetty [1] that actually referred to the process and problems of transferring a calibration with the terms calibration transfer and calibration transfer problem, respectively. Later papers by Wang and Kowalski [3, 53] also began to use these terms and currently the process of transferring a calibration is referred to as either a calibration transfer or an instrument standardisation.

The recognition of the instrument calibration problem appears to have coincided with the maturing of the use of multivariate techniques for instrument calibration. This perception is supported by the opening preface in the frequently cited text by Martens and Naes [4]:

> ... The theory and practice of multivariate calibration has now come far enough that a unified treatment of the topic is needed. ...

It is then evident that with the maturing of multivariate methods in instrument calibration, its use moves from laboratory investigations to practical field and production line applications. This then brings to the forefront the practical problems of using multivariate calibration techniques.

The paper by Wang *et al.* [2] devoted a considerable portion of the introduction to systematically describing the practical problems of multivariate calibration. According to Wang *et al.* [2], there are three instances when calibration can become a problem in NIR spectroscopy. These instances were discussed in general terms in Section 1.2.3. It is instructive to summarise, in terms of Figure (1.1), these problem-

atic instances described by Wang *et al.* [2] simply to provide a sense of realism and importance of the problem of calibration transfer.

According to Wang *et al.* [2], the first problematic instance for calibration occurs when two or more supposedly identical instruments, at the same or different locations, require a calibration. The problem in this instance is that the instruments are, in reality, not identical but have differences, due to sensors or other component variations. These differences, assuming K instruments, translates to the kth and lth instruments having $\mathbf{g}_k \neq \mathbf{g}_l$, $k, l \in \{1, 2, \ldots, K\}$. Typically, one instrument, referred to as the master, for example $\mathbf{g}_1$, is accurately calibrated with great expense to obtain the calibration model $\hat{h}_1$. The problem is that $\hat{h}_1$ cannot be placed inside, or transferred to, the other $k \neq 1$ instruments without incurring a reduction in measurement accuracy, in spite of the instruments being similar.

The other problematic instance occurs during the use of a single instrument over a period of time. In this instance the problem is, as generally described earlier, the result of physically induced instrument changes, caused by factors such as component aging or component replacement. This also results in $\mathbf{g}_k \neq \mathbf{g}_l$, where now k represents a time period and the next period is denoted by l. Again, the use of $\hat{h}_k$ with $\mathbf{g}_l$ will incur a reduction in accuracy so that the only recourse to maintain the initial accuracy is to recalibrate.

The last instance occurs when an instrument is sensitive to a particular set of uncontrollable, and difficult to measure, external factors that can exist during a particular time period, such as during an industrial production run. The problem, in this instance, is that $\hat{h}_k$ obtained during the kth period, or production run, is not adequate given a different set of factors, or values of the factors, that exist over a later time period or during a subsequent production run. This problem is similar to the previous problem except now the change in $\mathbf{g}$ is not due to variation in the characteristics of the internal instrument components but is due to the dependency of $\mathbf{g}$ on the external factors.

In their report, Wang *et al.* [2] also proposed and evaluated four approaches to address the problems in calibration transfer. These approaches are reviewed in detail in Appendix B. Following the report by Wang *et al.* [2], a number of other approaches have been reported. As will be in shown Section 2.2 and in terms of the definitions given in Section 1.3.1, most of the reported approaches use linear multivariate methods where it is also assumed that the calibration model, $\hat{h}$ as well as $\mathbf{g}$, can be adequately described using linear multivariate models. The assumption of linearity then allows the use of well established dimensional reduction methods, such as principal component regression, PCR, which regresses the principal components determined by a principal component analysis, PCA [4], or partial least squares, PLS, [54, 55] to determine the explicit form of the needed calibration model.

The application of these reported solutions, when given nonparametric calibration models or a highly nonlinear $\mathbf{g}$, does not appear to have been given much attention in the literature. This lack of attention also extends to the case of FFNN based calibration models. The void created by this lack of attention has been recognized by others [13, 52] as an area requiring further study.

With the increased use of highly nonlinear sensor systems in instruments, the need for nonparametric modelling methods grows more common. This growth also increases the need to transfer calibrations. Therefore, it seems appropriate to examine the calibration transfer problem for a highly nonlinear $\mathbf{g}$ that is also used in conjunction with a nonparametric calibration model, specifically, a calibration model based on a FFNN.

2.2 Similarity as a Basis for Solving the Calibration Transfer Problem

The idea presented in this section is the concept of exploiting the potential similarity between various realisation of calibration models or sensor systems as a means to frame and approach the problem of calibration transfer. The concept of

exploiting this similarity then leads into the introduction of the abstraction referred to as a similarity map.

The motivation in approaching the calibration transfer problem with the concept of similarity arises from the very common observation that a subjective sense of similarity exists not only between many everyday objects but that this similarity appears between curves representing a set of observations from physical phenomena. These phenomena, for example, include the volume of air expelled as a function of time (spirometer curves) for different individuals [56], human growth curves [10], or the acceleration curves due to the side impact of motor vehicles [10]. The appearance of similarity in these phenomenon is due partly to the phenomenon being governed by some, possibly unknown, set of parameters whose values varying from one instance of the phenomenon to another [57]. It then seems natural to consider the possibility that such similarity also exists between the responses provided by instruments manufactured to measure the same phenomena or between the response of the same instrument at different time intervals.

As will be discussed in later sections, the various reported approaches used to solve the calibration transfer problem can be viewed as attempts to exploit the similarity that may exist between components or systems forming the measurement process, as defined by Figure (1.1). This similarity may exists between various realisations of systems within a single instrument or between systems of multiple instruments. This view of a potential similarity between instrument systems is a broader perspective of another view presented by Anderson and Kalivas [58] where some of the solutions to the calibration transfer problem were placed in the context of Procrustes analysis, a form of multidimensional similarity structure analysis [59].

This section begins by first presenting a framework that encompasses the calibration transfer problem using the concept of a similarity mapping. The underlying assumptions used to exploit the similarity map so as to reduce the data needed for recalibration are then discussed.

2.2.1 The framework of similarity

The observation of similarity between physical phenomena may have motivated the approach to address the calibration transfer problem in near-infrared, NIR, spectrometers. In this case, using an extreme over-simplification of the instrument, it was noticed that the differences between spectral responses, that is, $\mathbf{x}(k) = \mathbf{g}_k(\mathbf{z}_i)$ and $\mathbf{x}(l) = \mathbf{g}_l(\mathbf{z}_i)$, of two supposedly identical instruments for a given measurement $\mathbf{z}_i$, could, over many spectral regions, be approximated adequately using simple linear relationships between $\mathbf{x}(k)$ and $\mathbf{x}(l)$ [53].

The consequence of this realisation results in the potential to represent the linear changes in the spectra using less calibration data than the calibration data needed to represent the entire individual spectra [2]. This realisation then became the impetus for the development of a number of approaches by Wang *et al.* [2] and other investigators to address the calibration transfer problem.

This idea of linearly transforming $\mathbf{g}_l(\mathbf{z})$ into being similar to $\mathbf{g}_k(\mathbf{z})$, parallels many ideas reported earlier by other researchers. One of the closest parallels was noted by Anderson and Kalivas [58] in reporting that many of the calibration transfer solutions can in fact be shown to be identical to Procrustes analysis, a form of multidimensional similarity structural analysis [59], developed much earlier.

Procrustes analysis attempts to *fit* one set of data so as to have it as *similar as possible* to another set under a given series of admissible transforms referred to as similarity transforms [59]. Similarity transforms are those that do not change the ratio of the *distances* between the data points, thereby maintaining the shape or form represented by the data. These transforms include operations such as rotation, reflection, translation, or dilation [59]. The two data sets used in these transforms are typically organized into matrices and matrix algebra and calculus are used to find the least squares fit between the two sets of data.

In calibration transfer, the spectral data, for example, are also expressed as matrices and the required transformation is obtained by estimating, in the least squared

sense, the coefficients of a linear set of equations. These types of calibration transfer solutions were shown by Anderson and Kalivas [58] to be a form of Procrustes analysis.

Another earlier idea motivated by similarities between curves or functions, that is not based on using the matrix transformation of Procrustes analysis, was the *shape invariant model* proposed by Lawton *et al.* [56]. In this case, Lawton *et al.* proposed that any lth curve, that is, function, $f_l : \mathbb{R} \rightarrow \mathbb{R}$ from a set of *related* curves $\mathcal{F}$, i.e. $f_l \in \mathcal{F}$ could be represented using

$$f_l(\theta_l; x) = \theta_{0l} + \theta_{1l} f_0 \left(\frac{(x - \theta_{2l})}{\theta_{3l}} \right). \tag{2.1}$$

where $x \in \mathbb{R}$, θ_{jl}, provides the shifting with $j = \{0, 2\}$, and scaling with $j = \{1, 3\}$ for the lth function, and f_0 is the characteristic shape for the set of curves, which can be any member of $\mathcal{F}$. Lawton *et al.* were interested in determining the *best* parametric curve f_0 for a given set of *similar* curves.

A common approach presents itself when these methods of calibration are examined: a mapping of either a function to another function or the mapping of one set of values to another set of values. For example, the approaches proposed by Wang *et al.* [2] to address the calibration transfer problem can be viewed not only as a linear transformation of $\mathbf{g}_l(\mathbf{z})$ so as to make it similar to $\mathbf{g}_k(\mathbf{z})$ but as a vector mapping operation, $\mathbf{f}_x : \mathbf{g}_l(\mathbf{z}) \rightarrow \mathbf{g}_k(\mathbf{z})$. In this example, the NIR transformation discussed previously, a map $\mathbf{f}_x$ may be given by $\mathbf{g}_k(\mathbf{z}) = \mathbf{g}_l(\mathbf{z})^T \mathbf{s}$, where $\mathbf{s} \in \mathbb{R}^n$ provides the linear scaling of $\mathbf{g}_l(\mathbf{z})$ over $\mathcal{X}$, which represents the n wavelengths measured by the spectrometer.

Stepping back further and viewing the entire instrument operation in conjunction with the change of $\mathbf{g}_k$ to $\mathbf{g}_l$, it seems natural to seek methods that could effectively make it appear as if $\mathbf{g}_l$ were restored back to $\mathbf{g}_k$ using any mapping operation on any component or systems forming the measurement process, as defined by Figure (1.1). With this view and a perspective of calibration transfer, a number of possible mapping strategies become evident.

Since these mapping strategies are attempting to restore the instrument operation so that it is *similar* to that of the kth calibration, they will be referred to as similarity mappings. Then, using Figure (1.1), a number of similarity mappings can be formed.

In expressing these similarity maps it will be convenient to designate the instrument functions and spaces that where obtained with high accuracy and significant cost as the master functions or spaces. These functions and spaces have been previously referred to as comprising the kth realisation of the instrument. It will also be convenient to refer to the changed functions, i.e. $\mathbf{g}_l$, and associated spaces obtained for the lth realisation of the instrument as the slave functions or spaces. The terms master and slave are also used in many of the reported calibration transfer methods reviewed in this thesis.

Now the impetus requiring a recalibration to be performed results from changes in the sensor system, that is, $\mathbf{g}_k$ changes to $\mathbf{g}_l$. The desired similarity mapping operation which can accommodate this change are those which can effectively make it appear to the instrument as if $\mathbf{g}$ had not changed at all. This approach to obtain the similarity maps can be viewed as a direct approach in applying the similarity map, in that it is $\mathbf{g}$ that has directly changed for some physical reason. For reference, a similarity map obtained using this approach will be referred to as a direct similarity map. The direct similarity map can then be viewed as a process that provides a calibration transfer so as to attempt to achieve a transformation or transfer such that it is still possible to use h_k.

To visualise how to achieve this transfer or similarity map, it will be instructive to refer to Figure (1.1) and then set h to h_k and $\mathbf{g}$ to $\mathbf{g}_l$. The idea is to then add any operations or functions to the block diagram of Figure (1.1), such that the resulting instrument operation is equivalent to $h_k \circ \mathbf{g}_k$. The direct similarity map is then defined to consist of the operations or functions added to the block diagram so as to achieved the needed equivalence, $h_k \circ \mathbf{g}_k$. The direct similarity map can then be viewed as allowing the continued use of h_k in the presence of $\mathbf{g}_l$.

Given this definition of the direct similarity map, a number of maps or functions that can achieve the required equivalence become apparent. In identifying these similarity maps, it should be noted that the maps can be considered as functions on A, where A is is defined as a space that depends on the nature of the similarity map. For example, the direct similarity mappings on A that can be formed are listed below.

1. For $A = \{$ **the space of calibration models** $\}$. $\Delta H : h_l \rightarrow h_k$
2. For $A = \{$ **the space of sensor systems** $\}$. $\Delta G : \mathbf{g}_l \rightarrow \mathbf{g}_k$
3. For $A = \mathcal{X}$, **the sensor output space**. $\mathbf{f}_x : \mathbf{x}(l) \rightarrow \mathbf{x}(k)$ or $\mathbf{f}_x : \mathbf{g}_l(\mathbf{z}) \rightarrow \mathbf{g}_k(\mathbf{z})$
4. For $A = \mathcal{Y}$, **the measurement space**. $\mathbf{f}_z : \mathbf{z} \rightarrow (\mathbf{g}_l^{-1} \circ \mathbf{g}_k)(\mathbf{z})$
5. For $A = \mathcal{Z}$, **the instrument response space**. $f_y : y_l \rightarrow y_k$ which after some work is equivalent to

$$f_y : y_l \rightarrow ((h_k \circ \mathbf{g}_k)^{-1} \circ (h_k \circ \mathbf{g}_l))^{-1}(y_l) = y_k \tag{2.2}$$

Here $\mathbf{g}^{-1}$ denotes the functional inverse, i.e. $\mathbf{g}^{-1} \circ \mathbf{g} = \mathbf{g} \circ \mathbf{g}^{-1} = I$, where I is the identity mapping $I(u) = u$, and not $1/\mathbf{g}$. Again, the subscripts and dependencies k and l are used to denote the kth and lth realisations of calibration model or sensor system. It should also be noted that these direct similarity maps are ideal maps in that they result in perfectly replicating the kth realisation of the various instrument systems from the corresponding lth realisation.

An alternative approach to obtain similarity mappings can be achieved with the desire to use the knowledge of $\mathbf{g}_k$, which was obtained during the kth calibration. In other words, given the change in $\mathbf{g}_l$, how can the knowledge of $\mathbf{g}_k$ be used to help estimate h_l.

In terms of Figure (1.1), set h to h_l and $\mathbf{g}$ to $\mathbf{g}_k$. In this case, a similarity mapping is formed by the addition of any admissible operations or functions to the block diagram that will achieve a mapping equivalent to $h_l \circ \mathbf{g}_l$. For reference, similarity maps

obtained in this manner will be called indirect similarity maps, in that the change is viewed to be indirectly occurring through h_l with $\mathbf{g}_k$ still present. The indirect similarity maps that can achieve this equivalence are then identical to those listed previously, except all the k and l subscripts and dependencies should be interchanged.

It should be noted that a number of the listed items are, in terms of instrument calibration, effectively nonsense mappings in that they require knowledge of the changed sensor system $\mathbf{g}_l$, or $\mathbf{g}_l^{-1}$, or calibration model h_l. If $\mathbf{g}_l$ or h_l were known there would be no need to calibrate.

For any valid similarity map, the actual relation or transformation defining the map can be expressed in a number ways, so long as the definition of the actual mapping is still identical to the valid similarity map. For example, the indirect similarity mapping for the space $\mathcal{A}$ of calibration models, ΔH item (1) with the kth and lth subscripts interchanged, can be defined as

$$h_l = h_k + s_l, \qquad \text{where } s_l = h_l - h_k. \tag{2.3}$$

or using

$$h_l = h_k s_l, \qquad \text{where } s_l = h_l / h_k. \tag{2.4}$$

or as

$$h_l = \Delta H(h_k), \qquad \text{where } \Delta H = h_l \circ h_k^{-1}. \tag{2.5}$$

where $s_l \in \mathbb{R}$ is referred to as the similarity function. In the same manner, the direct similarity map for the space $\mathcal{A}$ of sensor outputs, $\mathbf{f}_r$, item (3), can be defined, for example, using

$$\mathbf{x}(k) = \mathbf{g}_k(\mathbf{z}) = \mathbf{g}_l(\mathbf{z}) + \mathbf{s}_l(\mathbf{z}) \qquad \text{where } \mathbf{s}_l(\mathbf{z}) = \mathbf{x}(k) - \mathbf{g}_l(\mathbf{z}). \tag{2.6}$$

where, in this example, $\mathbf{s}_l \in \mathbb{R}^n$, other definitions include

$$\mathbf{x}(k) = \mathbf{g}_k(\mathbf{z}) = \mathbf{g}_l(\mathbf{z})\mathbf{s}_l(\mathbf{z}) \qquad \text{where } \mathbf{s}_l(\mathbf{z}) = \mathbf{x}(k)/\mathbf{g}_l(\mathbf{z}). \tag{2.7}$$

or

$$\mathbf{x}(k) = f_x(\mathbf{x}(l)) \qquad \text{where } f_x = \mathbf{g}_k \circ \mathbf{g}_l^{-1}. \tag{2.8}$$

2.2.2 The underlying assumptions in using similarity maps

In general, a calibration transfer, viewed as a similarity mapping, does not imply, for example, that the lth calibration model can be obtained with less data than the kth calibration model. The similarity maps simply show that it is possible to obtain, for example, the lth calibration model using knowledge of the kth calibration model. The question of whether it is possible, in using a calibration transfer, to reduce the calibration data can be answered by first viewing the calibration transfer as a similarity map.

A similarity map, in the context of a calibration transfer, is simply another function that needs to be determined from a set of data. Having obtained the similarity map then allows a calibration transfer. In other words, the answer to the question of whether it is possible to reduce the data needed for a calibration transfer is given by the answer to the question of whether the data needed to obtain an acceptable similiarity map is less than the data needed to obtain, for example, the lth calibration model without the use of a similarity map.

Qualitative answers to this latter question are not difficult to obtain. Consider the case where the similarity map is as *complex* as, for example, h_l, it becomes apparent that the similarity map will require a quantity of data that is comparable to that needed to obtain h_l. In other words, there is no advantage in using a similarity map in this case. Alternatively, if the similarity map is *simple* relative to h_l, then the data needed to obtain the similarity map will be less than the data needed to obtain h_l, thus allowing a reduction in the data needed for a calibration transfer.

These qualitative answers imply that the degree of *complexity* or *simplicity* of the similarity map is related to the degree of *similarity*, or alternatively, the degree of relevant *dissimilarity*, that exists between the kth and lth realisation of the instrument.

Ignoring for the moment the lack of precise mathematical definitions for terms such as, 'complexity' and 'similarity', it is reasonable to expect that a minimal amount data would be needed to obtain the similarity map in the case, for example, of identical kth and lth realisation of an instrument. In contrast, obtaining either the kth or lth realisation of h or $\mathbf{g}$, regardless of whether they were identical, may require a quantity of data that is greater than the minimal amount data needed to obtain the similarity map. In other words, the assumption in using the similarity map, that is a calibration transfer, is the assumption of *similarity* between the kth and lth realisations of the instrument, such that as the *similarity* increases the amount of data needed to obtain an acceptable similarity map decreases.

Unfortunately, given unpredictable, nonlinear instrument realisations, it is a non-trivial matter to define *complexity* and *similarity* so as to allow determining the degree of data needed to obtain the similarity map or perform a calibration transfer. Further misfortune is provided by realising that even if a measure of similarity were given, it is not possible, unless prior information is used, to obtain an estimate of the degree of similarity without first obtaining the data needed for recalibration. In other words, the data needed to obtain an acceptable estimate of the degree of similarity may exceed the data needed to actually perform the calibration transfer.

Therefore, in spite of the obvious relationship that exists between the similarity in instrument realisations and the data needed to obtain the similarity map, it is not at all obvious how to explicitly determine this relationship. It is also not the intent of this thesis to dwell on the problems associated with determining this relationship. Instead, it is more important, in the context of this thesis, to point out that it is the assumption of similarity between the kth and lth instrument realisations that is relied upon to allow a potential reduction in the data needed for a calibration transfer.

The practical consequence of the assumption of similarity, in the context of a calibration transfer, is that it is used without knowing the degree of *similarity* prior to the calibration transfer. Therefore, the resulting calibration error from a calibration

transfer will vary with the approximation error of the similiarity map whose *complexity* is dependent on the underlying similarity between the kth and lth instrument realisations.

The impact of changes in the underlying similarity between the kth and lth instrument realisations on the calibration error can be reduced by using additional information to assist in obtaining the similarity map. For example, if it were known that the similarity between the unpredictable, nonlinear kth and lth instrument realisations could always be adequately described using a linear relationship, then it would only be necessary to collect two data points to determined this relationship so as to allow an acceptable calibration transfer.

A simple attempt to understand the relationship that similarity has with the data needed for a calibration transfer and impact that the use of prior information has on this relationship can be obtained by viewing either h, $\mathbf{g}$, and the *difference* between the kth and lth realisations as objects. Here the difference between the kth and lth realisations is defined by the similarity map and its implementation that is selected to be used. For example, in using the similarity map ΔH, the *difference* between h_k and h_l can be given by $h_l - h_k$ or by h_l/h_k depending on which implementation is selected for the map. Regardless of the definition of this *difference*, it is the view that allows determining the *information* needed to adequately describe these objects that is of interest. This view is well developed in the field of information theory [60, 61] and it is used here in a very elementary manner without rigorous supporting proofs for statements.

It will be convenient to assume that all possible objects, as defined previously, form a set. It is also necessary to assume that one of the objects can be designated as a reference object. Now assume the existance of a metric I_a which indicates the quantity of information needed to describe the difference between object a and the reference object. In this context, $I_a = 0$ indicates that no additional information is needed to describe object a, that is, object a is equivalent to the reference object. In addition,

let $I_a < I_b$ indicate that the information needed to describe the difference between object a and the reference object is less than the information needed to describe the difference between object b and the reference object. Also, let $I_a = I_b \neq 0$ indicate that object a is equivalent to object b, both of which require the same amount of information to describe their difference between the reference object. Note that the equivalence of object a with b is in terms of the metric I.

Now, let I_k, I_l and $I_{k \rightarrow l}$, denote the information needed to adequately describe the kth and lth objects, and the *difference* between them, respectively. In practical terms, the possibilties that confront a calibration transfer can be then described with the following conditional relationships.

1. $I_{k \rightarrow l} < I_l$.

2. $I_{k \rightarrow l} = I_l$.

3. $I_{k \rightarrow l} > I_l$.

Now assume it is possible to relate the measure given by I to the data needed to describe an object. In this context, only item (1) allows the potential to reduce the data needed for a calibration transfer, that is the data needed to obtain the similarity map which describes the difference object k and l. The degree of potential data reduction is related to $\Delta I = I_l - I_{k \rightarrow l}$, that is, $\Delta I = 0$ would indicate that it is not possible to reduce the data and $\Delta I = I_l$ would indicate that the reduction could potentially be equal to that needed to describe object l. In this case, if the quantity of data is reduced to below the point specified by $I_{k \rightarrow l}$, then the calibration transfer will introduce calibration errors by virtue that the information is insufficient to adequately describe the similarity. Obviously, a further reduction in the data, in this case, would tend to increase the calibration errors. Item (2) indicates that it is not possible to reduce the data needed for a calibration transfer and item (3) indicates that the data needed to perform a calibration transfer will be greater than that needed to perform a standard calibration which obtains $\hat{h}_l$.

The use of *relevant* prior information $0 < I_p \leq I_{k \leftarrow l}$ to assist in obtaining the similarity map will appear as an additional term on the righthand side of the previous conditional relationships. that is. the appearance of the expression $I_l + I_p$ will result. Note it is assumed that the measure of this prior information can not exceed that given by $I_{k \leftarrow l}$. Now. for item (1) the degree of potential data reduction would then be related to $I_l + I_p - I_{k \leftarrow l} > I_l - I_{k \leftarrow l}$. which when given $0 < I_p \leq I_{k \leftarrow l}$ results in $I_l - I_{k \leftarrow l} \leq \Delta I < I_l$. For item (2). the data potential reduction is related to $0 < \Delta I \leq I_{k \leftarrow l}$. When item (3) is true. the reduction in data is only possible when $I_p > I_{k \leftarrow l} - I_l$.

Though the use of a metric given by I allows a very simple understanding of the limits in using similarity maps. in practice the use of I is severely restricted. This restriction results from the difficulty that can be associated with determining an adequate metric I for the objects of interest and to the difficulty in then relating this measure to the data needed to describe the objects. In this context. the importance of the previous discussion is in the awareness it raises regarding the limits of using similarity maps or calibration transfer methods.

2.2.3 Conventional calibration transfer solutions in the context of similarity maps

Instrument standarisation

Wang *et al.* [2] proposed a number of linear and piecewise linear methods to approach the calibration transfer problem. The relationship between these methods and the similarity maps is reviewed in detailed in Appendix B and is briefly described here.

The calibration transfer methods proposed by Wang *et al.* are referred to as a standarisation with the classical calibration model. a standardisation with the inverse calibration model. a direct standarisation. DS. and a piecewise direct standardisation. PDS. Appendix B discusses and shows how the standarisations methods proposed by Wang *et al.* can be placed within the framework of the similarity map. Specifically.

it is shown that the classical calibration model can be placed in the context of an indirect similarity map over the space of sensor systems. Standardisation with the inverse calibration model can also be placed in the context of an indirect similarity map, but the map is now over the space of calibration models. Both DS and PDS can be place in the context of a direct similarity map over the sensor output space. The PDS method differs from DS by forming a mapping $\mathbf{f}_r$ that consists of piecewise series of individual linear maps on various subspaces of the $\mathcal{X}$.

Following the publication of the paper by Wang *et al.* [2], the results of a number of additional investigations were reported that applied the calibration transfer methods described by Wang *et al.* [2]. In particular, the DS and PDS methods were the subject of many of investigations which showed that standarisation, or a calibration transfer, could provide a calibration accuracy comparable to that obtainable from a full instrument recalibration [3], that is, a recalibration not using a calibration transfer method. It was also shown by Wang et al. [62] that the accuracy of a less capable instrument could be improved, over that provided by a full recalibration, by standarising the less capable instrument with an instrument capable of greater accuracy. Wang and Kowalski also applied PDS to account for temperature induced variations of the instrument response [53]. The PDS method was later modified to allow it to correct for baseline offsets, also known as background corrections which corrects for the case of $\mathbf{x} \neq 0$ given $\mathbf{g}(0)$ [63]. A problem with PDS, and a solution, were also reported by Gemperline *et al.* [64], who had noticed the introduction of discontinuities in spectra transferred by PDS. More recently, PDS has been successfully applied to transfer calibrations of spectrometers used with fibre optic probes of various lengths [65].

Other methods of calibration transfer

Calibration transfer is, in a broad sense, an attempt to exploit the information in an accurately calibrated instrument so as to allow the use of another acceptable realisation of the instrument. The essence of this attempt is reflected in the simple

analytical framework provided by similarity maps which also reveals that the number of direct and indirect similarity maps, each with any number of implementations, is not restricted to a handful of realisations. With this insight, it is not surprising that a number of calibration transfer approaches have been reported.

One of the simplest calibration transfer methods that has been reported is a method based on linear slope and bias corrections of the calibration model [66]. In terms of a similarity map, this is given by the indirect similarity map $\Delta H : \hat{h}_k \rightarrow \hat{h}_l$, that is implemented using $\Delta H(\hat{h}_k) = s\hat{h}_k + b$, where $s, b \in \mathbb{R}$, represent the regression coefficients that need to be estimated. Bouveresse *et al.* [66] compared this simple method to PDS using various real data sets and showed that in some cases, slope and bias calibration transfer provided results as good as PDS. Since the slope and bias calibration transfer is simpler than PDS, Bouveresse *et al.* also cautiously proposed a diagnostic tool, using a statistical F-Test on the data, to decide when to use slope and bias correction as opposed to PDS.

Blanco *et al.* [67] proposed an almost identical approach, except in this case the regression coefficients b and s are based on using a few select *reference* points in $\mathcal{X}$ that appeared to provide reasonable accuracy in the calibration after transfer. It should be noted that the selection of values used to estimate the parameters of the similarity map, b and s in this case, is a non-trivial problem.

Another method, proposed by Blank *et al.* [68], is based on using a digital filtering process to transfer the spectra obtained from the slave instrument, $\mathbf{g}_l(\mathbf{z}) = \mathbf{x}(l)$, so as to allow it to be used on the master instrument. The filtering operation was referred to by Blank *et al.* as a mapping of the slave's spectra to the master's spectra. In terms of similarity mappings, this method can be placed in the context of a direct similarity map $\mathbf{f}_x : \mathbf{x}(l) \rightarrow \mathbf{x}(k)$ or $\mathbf{f}_x : \mathbf{g}_l(\mathbf{z}) \rightarrow \mathbf{g}_k(\mathbf{z})$, where the mapping operation is defined by a type of finite impulse response filter. The coefficients of the filter are determined using the actual spectral data from the master instrument. In one respect, the method proposed by Blank *et al.* resembles PDS in that both methods

use a selected set of subspaces from $\mathcal{X}$ to determine a series of maps which are then combined to form $\mathbf{f}_r$. The method proposed by Blank *et al.* also differs from PDS in that the selected set of subspaces is determined with a "moving window", or a "moving subspace" of $\mathcal{X}$.

This "moving window", in terms of the similarity map, can be visualised by first noting that the dimensionality of $\mathcal{X}$ is n, where n also represents the n wavelengths at which the spectrometer performs measurements. Now consider that any sensor system output can be represented as a point $\mathbf{x} \in \mathcal{X}$, which can be expressed as an n-tuple $(x_1, x_2, \ldots, x_n)$ relative to an appropriate basis, and where x_i, for $i = 1, 2, \ldots, n$, are the coordinates of $\mathbf{x}$ with respect to that basis. A "window" of "width" $p_q = a + b$ at the ith "position" can be defined as the subspace $\mathcal{X}_{q_i} \subset \mathbb{R}^{p_q} \subset \mathcal{X}$, whose elements are given by the p_q-tuple $(x_{i-a}, x_{i-a-1}, \ldots, x_i, x_{i-1}, \ldots, x_{i-b-1}, x_{i-b})$, where $p_q \leq n$, $1 \leq i \leq n - p_q + 1$, and $p_q, a, b \in \mathbb{N}$. For each i, a map from $\mathcal{X}_{q_i}$ to $\mathcal{X}_{r_i}$ is formed, where the subspace $\mathcal{X}_{r_i} \subset \mathbb{R}^{p_r} \subset \mathcal{X}$ is a window of width $p_r = 1$. The notion of a "moving window" arises from the practice of setting $i = 1$ and then repeatedly incrementing i by one until its limit is reached. In contrast, it should be noted that PDS may use any set of values for i that is deemed appropriate for the application and DS can be considered in this context by setting $p_q = p_r = n$. Blank *et al.* went on to report that one distinct advantage of their method over PDS is that the mappings can be determined using calibration data samples from only the master instrument as opposed to PDS which requires calibration data samples from both the master and slave instrument in order to determine the maps.

The similarity map $\mathbf{f}_r : \mathbf{x}(l) \rightarrow \mathbf{x}(k)$ given by the DS, PDS, or Blank's *et al.* method is used as a basis for many calibration transfer methods, each with various implementations and approaches to an implementation. Another such method was proposed by Swierenga *et al.* [69] where the idea of preprocessing the spectra, given by $\mathbf{x}$, was reported. Preprocessing of the spectral data can also be viewed as an attempt to transform $\mathcal{X}$ into a domain which is invariant to sensor system changes, thereby allowing a single calibration model to be used for any realisation of a sensor system.

Swierenga *et al.* examined various preprocessing techniques and then compared the resulting instrument calibration error to the error obtained using PDS and concluded that PDS provided slightly better accuracies than the data preprocessing methods. They also suggested that if only a slight improvement in accuracy is obtained, then the use of data preprocessing is preferred over PDS since data preprocessing does not require the calibration set D_t to be measured on both instruments. In their conclusion, Swierenga *et al.* also noted that if a principal component analysis of the spectral data from both instruments "show similar patterns", then it is likely that an acceptable calibration transfer is possible using a reduced quantity of calibration data. This statement by Swierenga *et al.* is also suggesting a potential technique that can provide an answer to the question motivating a Procrustes analysis: are the spectral data sets from the master and the slave instrument in fact the same under admissible transforms?

Another interesting approach in calibration transfer was proposed by Ozdemir *et al.* [70]. In this case, the similarity map is described as $f_x : \mathbf{x} \rightarrow x$. The idea proposed by Ozdemir *et al.* was to use the calibration data from both the master and the slave instruments to determine f_x so that a common calibration model h could be used for both instruments. They referred to the calibration model, determined in this manner, as a hybrid calibration model, HCM. The HCM was itself based on a linear univariate model that was explicitly determined using conventional least squares.

The interesting aspect of the approach reported by Ozdemir *et al.* was their use of a genetic algorithm [71] to not only determine which wavelengths to use to form the spectral response vector $\mathbf{x}$ for f_x, but to also determine an explict expression for f_x. In this case f_x was limited to a class of functions defined as $f_x = \sum_{i=1}^{N} f_i(x_{q_i}, x_{r_i})$, where $N > 2$, $q_i, r_i \in \{1, 2, \ldots, n\}$, $f_i \in \{(x_q + x_r), (x_q - x_r), (x_q x_r), (x_q / x_r)\}$. The genetic algorithm then determined a value for N, and for each i, determined the elements q_i, r_i, and f_i. Note that the spectral response vector $\mathbf{x} = [x_{q_1}, x_{r_1}, x_{q_1}, x_{r_1}, \ldots, x_{q_N}, x_{r_N}]^T$, where it is assumed that for $i \neq j$, $x_{q_i} \neq x_{r_j}$, $\forall i, j = 1, 2, \ldots, N$. By using spectral data from both instruments, the genetic algorithm attempts to find a map f_x such

that the resulting transformed data is invariant to differences in either instrument. The transformed data is then used to build a simple linear univariate calibration model $\hat{h}$, which can be used for either the kth or lth realisation of the instrument.

Another approach to determine $\mathbf{f}_x$ is to use a FFNN, as implemented by Despagne *et al.* [72]. Their method resembles the FIR filtering method of Blank *et al.* [68] in that a moving type "window" is used. This "window" is also defined using $\mathcal{X}_{q_i}$, that is, the window is associated with a "width" given by $p_q = a + b$ and a "position" given by i. A mapping from $\mathcal{X}_{q_i}$ to $\mathcal{X}_{r_i}$ is also determined for each i. Beyond this similarity, the method differs in that mapping is determined using a neural network whose training data set consists of the inputs $\mathbf{i}_{ij}$ and the desired outputs x_{ij}. Specifically, the training data is given by the set $D_l = \{(\mathbf{i}_{ij}, x_{ij}) : i = 1, 2, \ldots, n - p_q + 1, j = 1, 2, \ldots, N_l\}$, where N_l is the number of spectra $\mathbf{x}_j$ used in the calibration. $\mathbf{i}_{ij} = [i, (x_{i-a}, x_{i-a-1}, \ldots, x_i, x_{i-1}, \ldots, x_{i-b-1}, x_{i-b})]^T$ is the input resulting from placing the $\mathcal{X}_{q_i}$ "window" at the ith position over the jth spectra of the slave instrument, and x_{ij} is the desired output obtained from the $\mathcal{X}_{r_i}$ "window" placed at the ith position of the jth spectra of the master instrument. The FFNN is then trained to minimise the SSE, over D_l, between the desired output and the output of the FFNN using the Levenberq-Marquardt algorithm. The resulting approximation determined by the FFNN is the map $\hat{\mathbf{f}}_x$ representing the approximation of $\mathbf{f}_x$.

Despagne *et al.* [72] compared the calibration error of the instrument after standardisation using the FFNN determined f_x and after standardisation using PDS. The calibration error, resulting from using either the FFNN standarisation method or the PDS method, was very sensitive to the selection of the data for D_l. Over all the examined sets of calibration data, the FFNN standarisation method provided lower levels of calibration error than that provided by PDS.

One final method to be mentioned is that proposed by Koehler *et al.* [73]. Again, in terms of a similarity map this is another implementation of $\mathbf{f}_x : \mathbf{x}(l) \rightarrow \mathbf{x}(k)$. Their method is essentially a preprocessing of data in $\mathcal{X}$, which represents a Fourier

transform infrared spectra. The preprocessing in this case consists of using very selective digital filters to isolate specific regions of the spectra. The resulting filtered set of spectral regions is claimed to allow a better discrimination of the signatures of various analytes. The filtering is also claimed to provide a spectral response that has an increased invariance to sensor system changes and to background interference effects. The end result is that the spectral data from any $l \neq k$th instrument can be used with a single calibration model h_k. In terms of the similarity map, this method is again using a map $\mathbf{f}_r$ that transforms the data $\mathcal{X}$ into a domain that is less suseptible to sensor system changes and external interference, thereby allowing a single calibration model to be used.

Summary

The framework of the similarity map allows viewing the calibration transfer problem in a simple unifying manner. As was demonstrated, all the reported calibration transfer methods reviewed could be placed within this framework of similarity maps. This allowed a relatively easy identification of the underlying approach used in these reported methods. In this sense, the framework of the similarity map is just a method of categorising various calibration transfer techniques.

2.3 The FFNN Calibration Transfer Problem

This section introduces the FFNN calibration problem within the framework defined by the terms, concepts, and results presented previously in Section 1.2 through to Section 2.2. Within this framework the problem in FFNN calibration transfer is approached using the notion of similarity maps.

2.3.1 Selecting a similarity mapping approach

Section 2.2.1 outlined, in a simple manner, a number of approaches from which a similarity map could be determined. Determining a suitable mapping strategy and

its implementation is not as easily outlined by virtue that there appears to be a significant number of possible combinations of strategies and implementations. Instead of examining the merits of one particular combination over another, a pragmatic approach is used: select what appears to be the simplest combination in terms of implementation and, more importantly, for later analysis.

Using this pragmatic approach the indirect similarity map for the space of calibration models is selected. This mapping is given by

$$\Delta H : \hat{h}_k \rightarrow h_l. \tag{2.9}$$

Briefly, this similarity map achieves the equivalent mapping requirement $\hat{h}_l \circ \mathbf{g}_l$ so as to allow an attempt to transfer the kth calibration model to the lth model. Also note that the mapping is from $\hat{h}_k$ and not from h_k. This notation is used to indicate that an approximation of h_k has been obtained and that the map ΔH exists allowing h_l to be determined from $\hat{h}_k$.

An additional motivation in selecting the map given by Equation (2.9) is that $\hat{h}_k$ can represent the FFNN approximation of h_k. This implies that ΔH represents the process of approximation that is performed by the FFNN when it attempts to obtain h_l, given that it has already approximated $\hat{h}_k$. In other words, the similarity mapping operation appears to be inherently performed by the FFNN!

This appearance is deceiving in that the actual implementation of ΔH used by the FFNN is not as precisely defined as it was, for example, in the case of the linear the PDS or DS methods. The problem in defining ΔH for the FFNN is due in part to ΔH not being a simple mapping operation. Instead, ΔH can be more accurately viewed as describing the path taken along the FFNN's error surface during its approximation of h_l. As will be shown in Chapter 3, this path along the error surface is difficult to specify and is a complex function of the initial starting point of the path, the calibration data, the learning algorithm, and the FFNN architecture.

The FFNN mapping operation ΔH also hides another crucial view: that the similarity map ΔH, as implemented by the FFNN, does not require knowledge of $\hat{h}_k$

to determine h_l. More precisely, the operation of the FFNN can be described as

$$\Delta H_{\text{FFNN}} : \hat{h} \in \hat{\mathcal{H}}_p \rightarrow \hat{h}_l. \tag{2.10}$$

where $\hat{h}$ is any member of the class of functions, $\hat{\mathcal{H}}_p$, that the FFNN can determine and $\hat{h}_l$ is the approximation of h_l.

The mapping given by Equation (2.10) is not the map ΔH that is desired. The initial appearance of the FFNN inherently performing the similarity map is an illusion. The FFNN actually performs the mapping from any $\hat{h} \in \hat{\mathcal{H}}_p$ to $\hat{h}_l$, which in terms of recalibration, also includes, $\hat{h}_k \in \hat{\mathcal{H}}_p$, thereby providing the illusion that the needed similarity map is being implemented.

Given that the FFNN does not actually perform the desired similarity mapping, a specific implementation of the map needs to be given. Again, simplicity in the implementation is desired, so the desired mapping operation is defined by

$$\Delta H(\hat{h}_k) = h_l = \hat{h}_k + s_l. \tag{2.11}$$

where s_l represents the ideal change that is needed to obtain h_l from $\hat{h}_k$. This mapping is analogous to calibration transfer method of Wang *et al.* [2] referred to as standarisation with the inverse calibration model but also differs by not assuming linearity in $\hat{h}_k$, h_l, or s_l.

2.3.2 The problems

Given the lth set of calibration data, $D_l = \{(\mathbf{x}_i, y_i = h_l(\mathbf{x}_i)) : i = 1, 2, \ldots, N_l\}$, where $N_l \leq N_k$, the lth calibration can be approached either by using D_l to determine an approximation $\hat{h}_l$ or, as described earlier, using D_l and $\hat{h}_k$ to determine the similarity map and then using map, along with $\hat{h}_k$, to obtain $\hat{h}_l$. The former approach will be referred to as a standard recalibration. The latter approach is recognisable as a calibration transfer. Some of the concerns and problems in performing a calibration transfer given a FFNN calibration model $\hat{h}_k$ are now discussed.

Unknown similarity between $\hat{h}_k$ and $\hat{h}_l$

As discussed in section 2.2.2, the advantage in using a similarity map only occurs if an adequate map can be obtained with $N_l < N_k$. To obtain this advantage in using the similarity map requires that either one or both of the following be true: the similarity map is *simpler* in form than the form of $\hat{h}_l$; or that external relevant information can be used to help obtain the similarity map.

The existence of a *simple* similarity map is not known in advance, so it must be assumed that as the *similarity* of h_k and h_l increases, the information, as reflected by the data needed to describe the differences between h_k and h_l, approaches zero. Of course, this assumption may be wrong, in which case reducing the data will incur calibration errors by virtue that the similarity map will become less accurate and therefore $\hat{h}_l$ will become less accurate.

The impact of an incorrect assumption about the simplicity of the similarity map can be weakened by using additional information to help obtain the map. Though, these tactics of using additional information is justified theoretically, implementing a technique that uses this information to help obtain the similarity map presents problems.

Implementation of the similarity map: Approximation of $\hat{s}_l$

Equation (2.11) provides the desired mapping operation needed to obtain a calibration transfer. In terms of an FFNN calibration model, $\hat{h}_k$ represents the current FFNN approximation that is now inadequate and s_l is some unknown additive term that is needed to obtain h_l using $\hat{h}_k$. As used in the approach by Wang *et al.* [2], s_l needs to be approximated. This approximation is simply based on the ideal difference between the true or desired model h_l and the current FFNN approximation of h_k, that is, $\hat{h}_k$. Specifically, this ideal difference is given by

$$s_l = h_l - \hat{h}_k. \tag{2.12}$$

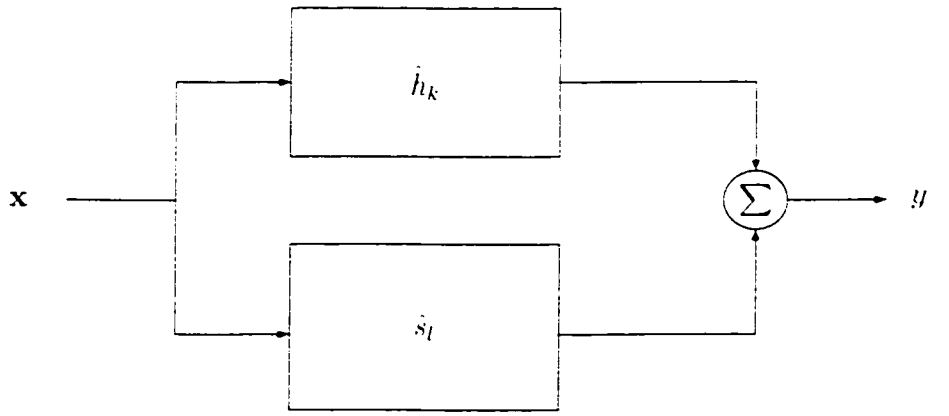


Figure 2.1. An implementation of the direct similarity map $\Delta H(\hat{h}_k) = \hat{h}_l = \hat{h}_k + \hat{s}_l$.

In practice, the function h_l is not known over all $\mathcal{X}$, but only at the data points given in $D_l = \{(\mathbf{x}_i(l), y_i = h_l(\mathbf{x}_i(l))) : i = 1, 2, \ldots, N_l\}$. The FFNN approximation, $\hat{h}_k$, on the other hand, is known over all $\mathcal{X}$ and therefore at any $\mathbf{x}_i(l)$. It then becomes apparent that the approximation of s_l needs to be based on the data set given by $S_l = \{(\mathbf{x}_i(l), h_l(\mathbf{x}_i(l)) - \hat{h}_k(\mathbf{x}_i(l))) : i = 1, 2, \ldots, N_l\}$, where $h_l(\mathbf{x}_i(l)) - \hat{h}_k(\mathbf{x}_i(l))$, represents the difference between the lth calibration data set D_l and the kth calibration model output at the calibration points given in D_l.

From the data in S_l, any method of approximation can be used to determine $\hat{s}_l$. In this sense, determining $\hat{s}_l$ is simply another problem in function approximation.

In an equally simple manner, the straightforward application of the similarity map ΔH, using $\hat{s}_l$, results in the calibration model shown in Figure (2.1). As can be seen from Figure (2.1), it is not necessary to modify the FFNN once $\hat{s}_l$ is determined. All that is needed to implement ΔH is to place the $\hat{s}_l$ in parallel with $\hat{h}_k$ and then sum the outputs. Since the approximation $\hat{s}_l$ is used, this implementation of the similarity map provides the approximation $\hat{h}_l$.

The approach shown in Figure (2.1) represents a viable alternative to calibration transfer not only for calibration models based on a FFNN but any nonparametric calibration model. Note that $\hat{s}_l$ may even be implemented with another FFNN.

However, the use of two FFNNs begs an answer to the question of whether or not it is possible to combine these two FFNN into one FFNN. More generally, the question is whether or not $\hat{s}_l$ can be combined with a FFNN, or in ideal terms, whether or not it is possible to embed knowledge of s_l into the FFNN so as to have it approximate $\hat{h}_k + s_l$.

Constraining the FFNN approximation with similarity

As was discussed previously, the FFNN does not explicitly use $\hat{h}_k$ when determining $\hat{h}_l$. Instead, the FFNN, as commonly used in instrumentation application, uses D_l to determine an approximation, not of h_l, but of the data in D_l. This characteristics is due directly to the process of approximation used by the FFNN.

It should be noted that though the FFNN does not explicitly use $\hat{h}_k$ when determining $\hat{h}_l$, the FFNN can, in a sense, be made dependent on $\hat{h}_k$. This can be achieved by not randomising the initial weights of the FFNN at the start of learning. Instead, if the initial weights are set to the values that provide the approximation $\hat{h}_k$, then in this sense, $\hat{h}_k$ is being used by virtue that the initial state of the FFNN is set to provide the approximation $\hat{h}_k$. This is not required by the FFNN, as it will attempt to determine $\hat{h}_l$ regardless of its initial state. This idea is explored in greater detailed in the following chapters of the thesis.

Now consider that in its approximation process, the FFNN attempts to minimise

$$C = \sum_{i=1}^{N_l} \left(h_l(\mathbf{x}_i) - \hat{h}_l(\mathbf{x}_i) \right)^2. \tag{2.13}$$

where C is referred to as the cost or objective function of the FFNN. Equation (2.13) is also referred to as the sum squared error, SSE.

As can be seen from Equation (2.13), only points at $(\mathbf{x}_i, h_l(\mathbf{x}_i)) \in D_l$ are used to guide the approximation, points $(\mathbf{x}, h_l(\mathbf{x})) \notin D_l$ are not considered. Therefore, if the similarity map defined by Equation (2.11), that is, $h_l(\mathbf{x}_i) = \hat{h}_k(\mathbf{x}_i) + s(\mathbf{x}_i)$, is simply

substituted into Equation (2.13), so as to obtain

$$= \sum_{i=1}^{N_l} \left((\hat{h}_k(\mathbf{x}_i) + s_l(\mathbf{x}_i)) - h_l(\mathbf{x}_i) \right)^2. \tag{2.14}$$

no change in the cost function has occurred. Therefore, in spite of explicitly expressing s_l in the cost function, no advantage is gained as no new information has been added.

The solution to this problem lies in not viewing $h_l(\mathbf{x}_i) = \hat{h}_k(\mathbf{x}_i) + s_l(\mathbf{x}_i)$ simply as a series of data points but as points from a continuous underlying function. The next key realization then follows: that in addition to the data values given by $\hat{h}_k(\mathbf{x}_i)+s_l(\mathbf{x}_i)$, there exists a set of corresponding values given by the data set $D_l^{(r)}$, which represents the rth order partial derivatives of $\hat{h}_k(\mathbf{x}_i) + s_l(\mathbf{x}_i)$ with respect to $\mathbf{x}$ at $\mathbf{x}_i$, where $r = 0, 1, \ldots, N_r$. In this setting $D_l = D_l^{(0)}$.

To obtain $D_l^{(r)}$, requires that $h_l(\mathbf{x}_i) = \hat{h}_k(\mathbf{x}_i) + s_l(\mathbf{x}_i)$ have at least a rth order partial derivative, that is

$$\left. \frac{\partial^r h_l}{\partial \mathbf{x}^r} \right|_{\mathbf{x}=\mathbf{x}_i} = \left(\frac{\partial^r \hat{h}_k}{\partial \mathbf{x}^r} + \frac{\partial^r s_l}{\partial \mathbf{x}^r} \right) \Bigg|_{\mathbf{x}=\mathbf{x}_i}. \tag{2.15}$$

is assumed to exist. For convenience Equation (2.15), can be more concisely expressed using

$$(\nabla_{\mathbf{x}})^r h_l(\mathbf{x}_i) = (\nabla_{\mathbf{x}})^r \hat{h}_k(\mathbf{x}_i) + (\nabla_{\mathbf{x}})^r s_l(\mathbf{x}_i). \tag{2.16}$$

where $(\nabla_{\mathbf{x}})^r h_l(\mathbf{x}_i)$ represents the rth order partial derivative of $h_l(\mathbf{x}_i)$ with respect to $\mathbf{x}$ at the point $\mathbf{x}_i$.

To allow the FFNN to use this additional information requires that cost function be modified so as to incorporate both the $D^{(0)}$ data set as well as the $D_l^{(r>0)}$ data set. This can be done by using

$$C^{(N_r)} = \sum_{i=1}^{N_l} \sum_{r=0}^{N_r} \left((\nabla_{\mathbf{x}})^r (\hat{h}_k(\mathbf{x}_i) + s_l(\mathbf{x}_i)) - (\nabla_{\mathbf{x}})^r h_l(\mathbf{x}_i) \right)^2. \tag{2.17}$$

With $N_r = 0$, Equation (2.17), reduces to the SSE of Equation (2.13), therefore, Equation (2.17) will be referred to as the generalized sum squared error, GSSE, that

is also recognisable as consisting of the sum square of the Sobolev norm for functions in L_2 having derivatives up to order N_r ([47], pg. 118-119).

It can be seen from Equation (2.17), that the gradients of the FFNN's previous approximation $\hat{h}_k$ and its current approximation $\hat{h}_l$, as well as, the gradients of s_l are needed to determine $C^{(N_r)}$. What this means, in terms of a FFNN approximation, is that the FFNN learning algorithm is not only trying to approximate the data points in $D_l^{(0)} = D_l$ but that it is also trying to simultaneously approximate the rth order partial derivatives at those points.

The use of these gradients has achieved the desired result of implementing the mapping ΔH. The FFNN must now use $\hat{h}_k$ to obtain $\hat{h}_l$. Of course, this implies that not only must all the rth order derivatives be used, which may not be bounded, i.e. $N_r \to \infty$, but that precise knowledge of $(\nabla_{\mathbf{x}})^r s_l(\mathbf{x}_i)$ needs to be available. It should be noted that precise knowledge of $(\nabla_{\mathbf{x}})^r \hat{h}_l(\mathbf{x}_i)$ and $(\nabla_{\mathbf{x}})^r \hat{h}_k(\mathbf{x}_i)$ is available by virtue that these expressions represent the partial derivatives of the FFNN calibration model, which is known. Pragmatically, it may be sufficient to use only the first few derivatives and to allow some degree of error in $(\nabla_{\mathbf{x}})^r s_l(\mathbf{x}_i)$. This pragmatic approach raises a number of unaddressed concerns, such as will the use of the GSSE improve the calibration error relative to either the error resulting from using the direct approach of simply adding the output of $\hat{s}_l$ to $\hat{h}_k$, or to the error resulting from using $N_l \leq N_k$ in a standard calibration with conventional FFNN learning methods, such as the method of backpropagation [74].

An additional concern in applying the GSSE is the use of s_l, and not $\hat{s}_l$, in Equation (2.17). In pragmatic terms, $\hat{s}_l$ will need to be used, which then implies that the rth order partial derivatives will be based on $\hat{s}_l$, thereby resulting in unavoidable errors in the derivatives.

As it will be shown in the next section, some guidance into addressing theses concerns can be obtained from the works of neural network researchers, where related ideas of embedding knowledge such as, $\hat{h}_k(\mathbf{x}_i) + s_l(\mathbf{x}_i)$, into a FFNN have been ex-

amined. In addition, the idea of a FFNN approximating the derivatives of a function has also been examined, to some degree, by others.

2.4 Supporting Research and Ideas

It was shown that the use of only D_l to approximate the similarity map, $\Delta H(\hat{h}_l) = \hat{h}_k(\mathbf{x}_i) + s_k(\mathbf{x}_i)$, with a FFNN does not achieve the desired mapping. The FFNN must use information over and above that provided by the calibration data set D_l, such as that provided by $D_l^{(r)}$. This additional information can be viewed as prior information regarding the *smoothness* of the similarity map ΔH at the data points given in D_l.

The explicit exploitation of prior information has been frequently reported in the neural network, NN, literature [75–82] where its implementation, for example, as a set of constraints in learning [80, 82, 83], has been shown to result in a reduction in the data and time needed for learning and in improving generalisation. Obviously, these are the same results that are desired in a calibration transfer. It then seems reasonable to expect, in using a FFNN as a calibration model, that equivalent results will surface when prior information is used in a calibration transfer.

It will be shown that this expectation is more than reasonable by virtue that the FFNN cost function given by Equation (2.17) resembles key features of a number of FFNN learning approaches that were intended to exploit prior information. These particular FFNN learning approaches have been shown to require less data and time to obtain an approximation and to provide improved generalisation. Though a resemblance to these FFNN learning approaches does not ensure comparable performance in a calibration transfer, it does increase the confidence that Equation (2.17) is a viable FFNN cost function that can potentially reduce the data needed for a calibration transfer.

Though this section will point out the similarities and differences between the reported NN cost functions and the proposed FFNN cost function, given in Equa-

tion (2.17). it will also point out that derivative learning is not as well developed as is the standard learning of data values. i.e. zero order derivative values.

2.4.1 Using prior information

One method of using prior information in a FFNN was the idea of hints proposed by Abu-Mostafa [76, 77]. A hint represents information regarding h_l that is obtained independently from D_l. Information such as invariance, monotonicity, and evenness or oddness of a function can be used as a hint or set of hints. In this context, assuming the existence of each nth order derivative of h_l can also be considered a hint regarding the smoothness of h_l.

Hints are represented using, what Abu-Mostafa termed, duplicate and virtual examples. Duplicate examples resemble real samples, say in D_l, and can be used together with the real samples to obtain an approximation, such as $\hat{h}_l$. In other words, each duplicate example has an input value $\mathbf{x}'$ and an associated desired output value $h'_l(\mathbf{x}')$. The duplicate example is generated by both D_l and the hint. For example, given the hint of evenness, then $\forall\,(\mathbf{x}_i, h_l(\mathbf{x}_i)) \in D_l$, $\exists\,(\mathbf{x}', h'_l(\mathbf{x}')) = (-\mathbf{x}_i, h_l(\mathbf{x}_i))$, which represents duplicate examples.

Virtual examples, on the other hand, do not have an associated desired output value. Instead, virtual examples consist of a pair of input values, $(\mathbf{x}_1, \mathbf{x}_2)$ where the NN is to have equal outputs, i.e. $\hat{h}_l(\mathbf{x}_1) = \hat{h}_l(\mathbf{x}_2)$. To incorporate this hint into the NN requires expanding the scope of the cost function used by the NN to not only minimise the error in approximating the data in D_l, but to also minimise the error in the rth hint, using $e_r = \|\hat{h}_{l_r}(\mathbf{x}_i) - \hat{h}_{l_r}(\mathbf{x}_i)\|^2$, where e_r is the NN's error in using the rth hint and $\hat{h}_{l_r}$ is the output of the NN using the rth hint.

The existence of multiple error measures e_r, for $r = 0, 1, \ldots, N_r$, requires a method to combine them into one global cost function. Abu-Mostafa suggests a number of

strategies to combine these error measures. One common method is to simply use

$$C = \sum_{r=0}^{N_r} \alpha_r E_r. \tag{2.18}$$

where α_r is a weighting factor assigning a relative importance to the error and E_r is the estimated error of e_r over all virtual examples of the rth hint.

The similarity of Equation (2.18) to Equation (2.17), the GSSE, is apparent, simply set $E_r = \sum_i \left((\nabla_{\mathbf{x}})^r (\hat{h}_k(\mathbf{x}_i) + \hat{s}_k(\mathbf{x}_i)) - (\nabla_{\mathbf{x}})^r \hat{h}_l(\mathbf{x}_i) \right)^2$, and $\sum_r \alpha_r = 1$. In other words, Abu-Mostafa's error measure for the rth hint has a correspondence to the rth order derivative error in the cost function of Equation (2.17). In terms of the virtual example, this is where the similarity ends, but in terms of the duplicate sample there is an additional likeness. In the same way duplicate examples are generated using D_l and the hint, the derivative data samples need to be *generated* using D_l and a *hint*, that is, the rth partial derivative of $\hat{s}_k$ with respect to $\mathbf{x}$ at each $\mathbf{x}_i$. In this view, the GSSE requires the use of duplicate examples.

Abu-Mostafa then provides both an argument and empirical evidence supporting the claim that virtual and duplicate examples do improve generalisation. In this context, it appears that the GSSE represents a viable cost function by virtue that it requires the use of duplicate examples generated by the hints consisting of the rth order partial derivatives.

Additional support in using duplicate examples is provided by the work of Niyogi, Girosi, and Poggio [75]. In this case, Niyogi *et al.* refers to duplicate examples as virtual examples. To avoid confusion, Abu-Mostafa's term, duplicate example will be used. Essentially, Niyogi *et al.* showed that the generation of duplicate examples from an existing data set, say $D_l = \{(\mathbf{x}_i, h(\mathbf{x}_i) : i = 1, 2, \ldots, N_l\}$, and the use of some *legal* set of transform that resemble Abu-Mostafa's hints was equivalent to a form of regularisation. The importance of this equivalence is that regularisation is a well known technique [81, 84] to solve *ill-posed* problems. In other words, using duplicate examples has a stabilising effect on the approximation solution [9], in the sense that small changes in the data values do not result in large changes in the solution [84].

Another approach in using prior information was suggested by Kramer *et al.* [79], where they proposed to explicitly embed known theoretical models and other constraints into a hybrid FFNN structure. The importance of their work is that the hybrid FFNN structure they proposed has a strong resemblance to the similarity map shown in Figure (2.1). In the Kramer *et al.* FFNN structure, a theoretical model s is placed in parallel with the FFNN model and the output of both models are summed together. Kramer *et al.* refers to the theoretical model s as providing a default behaviour which can be viewed as providing information at points over entire the input space $\mathcal{X}$, including those given by the set D_l. Constraining information is said to represent prior knowledge of the underlying process that is being modeled. Kramer *et al.* combine these ideas by defining the FFNN cost function using $C = \|(h_l(x) - \Gamma(\hat{h}_l(x) + s(x))\|^2$, where $h_l(x)$ is the desired output for a given x, $\hat{h}_l(x)$ is the FFNN approximation, $s(x)$ is the theoretical model output for a given x, and Γ is an auxiliary function selected to satisfy the constraints placed on the model. Similar approaches were also reported by Johansen and Foss [85] and Thompson and Kramer [78].

The difference between the approach used by Kramer and the similarity mapping approach, besides the use of an auxiliary function Γ, is that in Kramer's approach the model s is specified and based on known physical laws describing at least a portion of the underlying phenomenon being modelled. In the similarity mapping approach there is no a prior specification of s_l, it is approximated using only the data D_l and the prior calibration model $\hat{h}_k$. The only a prior specification of s_l that may be employed is to assume some underlying functional form for s_l, such as linearity or piecewise linearity. But the modeling approach used by Kramer *et al.* does suggest the option of specifying a functional form for s_k that can be based on prior knowledge of the underlying process. Finally, an additional difference between the similarity map and Kramer's approach is that s_l in the similarity mapping represents the difference between $\hat{h}_k$ and h_l, whereas in Kramer's approach, s represents a default model of some physical process.

2.4.2 Derivative learning

Another important aspect that needs to be examined is the proposed use of derivatives in the GSSE measure. In support of using derivatives in FFNN learning there are a number of reports [86-95] oriented towards specific applications or pure algorithm derivations. This support is not ideal in that none of the reports directly consider calibration transfer. In addition, only the use of the zero and first order derivative terms, i.e. $\sum_{r=0}^{1} \|(\nabla_{\mathbf{x}})^r h_l(\mathbf{x}_t) - (\nabla_{\mathbf{x}})^r \hat{h}_l(\mathbf{x}_t)\|_2^2$ are considered. For convenience in presentation, conventional learning based on minimising the error between the desired output and the network output data values will be referred to as $D^{(0)}$ learning, whereas learning which uses the GSSE up to and including the nth order derivatives will be referred to as $D^{(n)}$ learning.

In spite of the non-ideal support of the GSSE, all the reports which do provide results, tend to indicate that generalisation improves and learning speed increases with the use of $D^{(1)}$ learning. More importantly, a number of reports [87, 91-93] have indicated that, for a given level of generalisation error, $D^{(1)}$ learning requires less data than that needed in $D^{(0)}$ learning. In addition, results have suggested that the use of only $D^{(1)}$ learning can also provide improvements in generalisation error over the error obtainable with $D^{(0)}$ learning. Though these results are shown empirically, other results have also indicated that $D^{(1)}$ learning does not always provide improvements in generalisation error.

For example, Lee and Oh [95] suggested that the use of $D^{(1)}$ learning may require "tuning" of the step sizes during learning to help insure that improvements in generalisation error occur. Essentially, one step size η_d is used to control the rate of optimising, that is, minimising, the error in data and another step size η_m is used for the same purpose to minimise the error in the first derivatives, i.e. the slope data. Lee and Oh claim that the tuning of the step sizes is required in order to balance the desire to learn the slopes with that of fitting the data. A strategy to properly balance or tune these step size, during learning, was not described.

In one simulation example Lee and Oh attempted to learn a function using noise corrupted slope values. Their strategy in this case was to initially set $\eta_d = 10\eta_m$ and then after a set number of epochs, begin to reduce both step sizes linearly as learning progressed. One important observation is that the reduction in η_m was started sooner, thereby reaching zero much early than η_d. This effectively reverted the learning back to $D^{(0)}$ learning during the latter portion of the FFNN training session. The rationale provided by Lee and Oh in using this particular strategy was based on the idea that the error in the slope data was not conducive in reducing the generalisation error during the "fine-tuning" stage of the learning, i.e. the latter portion of the learning. The impression left by Lee and Oh is that, if η_m had not been reduced to zero, the generalisation error may have exceeded the error resulting from using only $D^{(0)}$ learning.

Evidence that the additional use of slope learning can potentially increase the generalisation error over that of using only data learning was reported by Masuoka [92]. In one series of simulations, ([92], pg..46-48), Masuoka compared the generalisation error of $D^{(0)}$ and $D^{(1)}$ learning as a function of number of training data samples. In this simulation, $D^{(0)}$ learning was shown to have achieved a lower generalisation error than that achieved with $D^{(1)}$ learning, when fewer than four or more than 50 input samples were used. A reason for these results was hypothesised but no supporting analysis or additional empirical results were provided. Essentially, the brief hypothesis, that was provided, was based on the idea of either over constraining the approximation with a large number of points or that slope learning with only a few points tends to dominate data learning so as to sacrifice a small generalisation error for a small slope error. However, it was also pointed out that on average, $D^{(1)}$ learning does reduce the generalisation error over that of $D^{(0)}$ learning.

The importance of these observations is that they show a nondeterministic behaviour for $D^{(1)}$ learning. Though this simply mirrors the known behaviour in $D^{(0)}$ learning, the important difference is that in some instances the advantage of $D^{(1)}$ does not appear to significantly improve the generalisation error over that obtained

with $D^{(0)}$ learning. Currently, no reports have been found conclusively identifying the instances or conditions under which $D^{(0)}$ and $D^{(1)}$ learning provide comparable levels of generalisation error, or more importantly, when the error obtained from $D^{(0)}$ learning is lower than that obtained from $D^{(1)}$ learning.

Though, no conclusive analysis identifies these conditions, it was shown that the estimated average of the generalisation error resulting from $D^{(1)}$ learning is lower than the average generalisation error resulting from $D^{(0)}$ learning. These estimated average errors are measured over various initial starting conditions, i.e. neural weights, while the NN topology, the underlying function, and the number of training data samples is held fixed. It was also briefly noted that the difference between these average errors decreases as the number of sample points increases. In addition, it was shown that as the error in the slope data increased the average generalisation error resulting from $D^{(1)}$ learning increased, where at some variable level of slope error, the generalisation error was greater than that resulting from $D^{(0)}$ learning [91-93].

Finally, one additional observation regarding the use of derivatives in the GSSE should be made. As discussed previously, setting $N_r = 0$ reduces the GSSE to the conventional SSE measure. The SSE effectively provides one measure to estimate the difference between two functions over a set D_l consisting of N_l points. It is shown in Chapter 3 that if the functions are expanded about these points using a Taylor series expansion, then the SSE can be viewed as being obtained by simply truncating the series after the first term and then measuring the distance between the two points representing the functions in a N_l dimensional space. The GSSE, on the other hand truncates the series after the N_r term and measures the distance between two points in a $N_l \sum_{r=0}^{N_r} n^r$ dimensional space, assuming $x \in \mathbb{R}^n$.

Using the Taylor series representation, the GSSE appears to be a *stronger* measure of the difference between two functions by virtue that the GSSE includes higher order terms. A similar measure was also proposed by Bajcsy and Ahuja [96] in the area of image segmentation. Bajcsy and Ahuja proposed a measure of homogeneity to

indicate the degree of similarity that exists between sample points in an image. Each sample point was assigned a nth order feature consisting of the point's nth order derivatives. If the difference in the similarity measures of two points was less than some predefined threshold, then the points were considered to belong to the same image segment.

With this view, the use of the GSSE can be said to more strongly emphasise the error between two functions. Therefore, it is reasonable to expect that reducing the generalisation error in the approximation of $h_l = \hat{h}_k + s_k$, by reducing the the GSSE, will, for a given N_l, result in a tendency for $\hat{h}_l$ to approach, not just h_l, but $h_l = \hat{h}_k + s_k$ more quickly as N_r increases. This is in direct contrast to reducing the SSE which, for a given N_l, will result in obtaining $\hat{h}_l$ without regards to the relationship $h_l = \hat{h}_k + s_k$.

3. FFNN Calibration Transfer Methods

3.1 Overview

The calibration transfer methods described in this chapter assume that an instrument has been previously calibrated to obtain the kth realisation of the calibration model, $\hat{h}_k$. This calibration model $\hat{h}_k$, implemented with a FFNN, also provides instrument readings $\hat{y} = \hat{h}_k(\mathbf{g}_k(\mathbf{z})) = \hat{h}_k(\mathbf{x})$ having acceptable calibration errors. It is then assumed that instead of using $\mathbf{g}_k(\mathbf{z})$ the instrument is confronted with using a changed sensor system $\mathbf{g}_l(\mathbf{z})$ which results in instrument readings having unacceptable calibration errors.

To provide acceptable calibration error, a new calibration model h_l is desired. This new calibration model can be obtained using the conventional approach: performing a standard recalibration using the FFNN with the data set D_l to obtain the needed approximation $\hat{h}_l$.

Alternatively, $\hat{h}_l$ can be obtained by using a calibration transfer method. The calibration transfer methods considered in this thesis can be described by the expression

$$\Delta H(\hat{h}_k) = h_l = \hat{h}_k + s_l. \tag{3.1}$$

Equation (3.1) can be viewed as an additive implementation of the indirect similarity map ΔH which represents an ideal expression in that it assumes the use of an ideal s_l so as to obtain h_l. More realistically, s_l will need to be approximated with $\hat{s}_l$ using a data set S_l based on both D_l and knowledge of $\hat{h}_k$. The approximation $\hat{s}_l$ is then determined using conventional parametric approximation methods and substituted

into Equation (3.1) to obtain

$$\Delta H(\hat{h}_k) = \tilde{h}_l = \hat{h}_k + \hat{s}_l. \tag{3.2}$$

where $\tilde{h}_l$ is used to distinguish this approximation from that obtained with the FFNN. In terms of a calibration transfer. methods based on using Equation (3.1) will be referred to as an indirect additive calibration model transfer. that is identified with the acronym iDACX.

In addition to the indirect additive implementation of ΔH. this thesis also examines the approximation of h_l by the same FFNN which has already approximated $\hat{h}_k$. The FFNN approximation. given by $\hat{h}_l$. is obtained by minimising. with $N_r = 1$. the GSSE between $\tilde{h}_l$ and $\hat{h}_l$. as measured over the data set $G_l^{(1)}$. The data set $G_l^{(1)}$ is defined as $G_l^{(1)} = D_l^{(0)} \cup D_l^{(1)}$. where $D_l = D_l^{(0)}$ and $D_l^{(1)}$ represents an estimate of the first derivatives of $\tilde{h}_l = \hat{h}_k + \hat{s}_l$ with respect to $\mathbf{x}$ at $\mathbf{x}_i$ given $(\mathbf{x}_i, h_l(\mathbf{x}_i)) \in D_l$. For convenience. this approach to calibration transfer will be referred to as a first order approximation of the iDACX. identified with the acronym $D^{(1)}$-iDACX. It should be noted that in general. nth order approximations to the iDACX method can be formed.

In all. three different calibration transfer methods are considered in this chapter. Each method obtains the approximation of h_l using the following approaches:

1. $\hat{h}_l$ representing the FFNN approximation obtained by determining a minimum in the SSE between h_l and $\hat{h}_l$ as measured over the data in D_l. where the weights of the FFNN are initialised to values that provided the previous approximation $\hat{h}_k$. This is method is referred to as a prior initialisation calibration transfer. PICX.

2. $\tilde{h}_l$ representing the approximation obtained using a iDACX method. where $\hat{s}_l$ is obtained with conventional parametric approximation techniques using the data in S_l.

3. $\hat{h}_l$ representing the FFNN approximation obtained using a $D^{(1)}$-iDACX method and the data in $G_l^{(1)}$, where the weights of the FFNN are either randomly initialised or initialised as per the PICX method.

The calibration error resulting from using these transfer methods are, in later simulations, compared to the calibration error resulting from a standard calibration using conventional backpropagation learning of the data D_l.

The calibration error resulting from these approximations is assessed using an error measure, such as the RMSEP defined by Equation (1.3), where the validation data set T satisfies the condition $(T \subset D_0) \cap D_l = \emptyset$ and where D_0 represents the complete calibration data set.

It is apparent that the calibration error, resulting from using any one of the transfer methods, will depend on a number of factors. For example, it is clear that not only will the calibration error depend on the actual calibration data in D_l but also on number of data samples in D_l, $N_l = \bar{\bar{D}}_l$, which is also assumed to be less than $\bar{\bar{D}}_k$. In addition, the error will also depend on factors, such as the underlying *similarity* between h_l and h_k, the specific FFNN architecture, and the learning algorithms.

To appreciate these dependencies, this chapter discusses the details of obtaining the approximation of h_l using the calibration transfer methods listed previously. This discussion begins by briefly reviewing the FFNN and the architecture chosen for this thesis. This is followed by a review of the FFNN learning process. Finally, the introduction and development of the calibration transfer methods are presented.

3.2 The FFNN as an Instrument Calibration Model

In chapter 1 the FFNN was simply described as a nonlinear multivariate mapping $\hat{h}_k : \mathbf{x} \rightarrow y$ that could be used to implement the instrument calibration model. This section will show that the FFNN selected to implement the instrument calibration model can be viewed as a nonlinear approximator. Though the FFNN, viewed

as a nonlinear approximator, is nothing more than another mathematical approximation model, the FFNN selected for this thesis represents a specific class of neural network from a large family of neural network models. This section will not present a detail review of all neural network models as this review is available in a number of texts [9, 47, 97, 98]. Instead, the intent of this section is to provide a sufficient level of detail to allowing identifying the FFNN model chosen for this thesis from within the large family of neural network models and to gain an understanding of the capabilities and limitations of the selected FFNN as it relates to calibration transfer.

3.2.1 The FFNN

The FFNN selected for this thesis represents a specific class from a large family of neural network, NN, models. These NN models are also known as artificial neural network models, connectionist models, parallel distributed processing models, or neuromorphic systems [99]. To appreciate the impetus in using terms such as neural or connectionist to describe these models, it is only necessary to note that the development of the NN models were motivated by the desire to emulate various observed physiological behaviour such as word and object recognition, language comprehension, and muscular motor coordination in tasks as simple as reaching and grasping for an object [100].

To attempt to emulate these physiological behaviours, the NN models were inspired by the vast interconnection of large numbers of mammalian neurons in the brain. This inspiration lead to forming NN models that also exhibited the interconnection of a large number of simple processing units [99, 100], that for obvious reasons, are also commonly referred to as neurons.

Borrowing from the know neurophysiological characteristic of the nervous system, Rumelhart *et al.* [101] proposed a general framework from which most of the specific instances or classes of NN models could be identified as special cases. Though the components of the framework proposed by Rumelhart *et al.* [101] to generalise NN

models were motivated by neurophysiological characteristics, the components of the framework are formed using a simplistic interpretation of these characteristics. It should also be noted that many of the known neurophysiological characteristic ([102], pg. 985-986) are not used in this framework. The simplification and exclusion of neurophysiological characteristics results in Rumelhart and McClelland ([103], pg. 135-138) conceding that at best the NN models are a coarse approximation of neurophysiological processing and the models may fall at a level between the macro structure of cognition and the details of neurophysiological. Essentially, it is likely that the simplification and exclusion of neurophysiological characteristics could mean the difference between providing an accurate emulation of observed physiological behaviour or a poor emulation. Rumelhart *et al.* [101] also cite the works of Kohonen [104, 105], Amari [106], and Feldman and Ballard [107] as those also attempting to generalise a framework for NN models.

The components from the framework proposed by Rumelhart *et al.* [101] that are used to form a NN model are listed below.

1. **A set of processing units**. Processing units, also referred to as neurons, receive inputs from other processing units. Then using these inputs and a set of rules, the processing unit sends an output to other processing units. It has become a common practice to identify units as being either input units, hidden units, or output units. Input and output units typically interface to the environment. Hidden units simply connect to input, output, or other hidden units, thereby, effectively hiding the units from view from the environment.

2. **A state of activation**. Each jth processing unit is associated with being in a state ν_j at time t, where $\nu_j(t) \in \mathbb{R}$, or $\nu_j(t) \in \{-1, +1\}$. Other definitions for the state ν_j are also possible.

3. **An output function**. Given the state ν_j of jth processing unit, a set of rules, is used to determine the output value o_j of the jth unit. Typically a mapping $\Phi_j : \nu_j(t) \rightarrow o_j(t) \in \mathbb{R}$ is used.

4. **A pattern of connectivity**. The individual connection from the output of the ith unit to the input of the jth unit is commonly specified by a weight $w_{ji} \in \mathbb{R}$, where $|w_{ji}|$ represents the strength of the connection and $w_{ji} = 0$ represents no connection.

5. **A propagation rule**. The input to each unit is formed by using propagation rules to obtain the jth unit's net input $\mathbf{v}_j(t) = [v_{j1}, v_{j2}, \ldots, v_{jn_j}]$, where v_{jk} is the kth type of net input and n_j is the total number of net inputs to unit j. Each kth net input is formed by first defining its elements using the set $\mathcal{P}_k = \{P_i = (o_i, w_{ji}) : i \in I_k\}$, where $o_i(t)$ is the output from the ith unit, w_{ji} is the weight connecting the output of the ith unit to the jth unit, and I_k is the kth indexed set such that $\forall i \in I_k$, P_i is a valid output and weight connecting to the jth unit. The propagation rule is commonly a mapping that is, in general, unique for each set $\mathcal{P}_k$, i.e. $v_k : \mathcal{P}_k \to v_{jk}$. Typically, instead of multiple types of net inputs, only one type of net input is used, that is $k = 1$, so that only one propagation rule is needed, such as $\mathbf{v}_j(t) = v_j(t) = \sum_{i=1}^{n} w_{ji} o_i(t)$, which is commonly used given n outputs connecting to the jth unit.

6. **An activation rule**. This rule determines the next state of activation of a unit by using the unit's current state and its net input. This rule is commonly referred to as the unit's activation function o, where $o_j : (\mathbf{v}_j(t), \nu_j(t)) \to \nu_j(t + \Delta t)$, and where $t + \Delta t$ defines the instant of the next state. One important characteristic that is often given to the activation function is a thresholding behaviour where the output of o is bounded over the domain of the argument, i.e. $\sigma^- \leq o(\cdot) \leq \sigma^+$, where σ^- and σ^+ are respectively, the lower and upper bounds of o, and where $-\infty < (\cdot) < \infty$. The thresholding from σ^- to σ^+ usually occurs over some finite subinterval, or point, in the domain of the argument of o.

7. **A learning rule**. The process of adding, removing, or modifying the value of the weights is governed by the learning rule. The learning rule, also referred

to as a learning algorithm [9], is typically designed to adjust the pattern of connectivity to the jth unit in an attempt achieve some objective for the jth unit, such as achieving a specific activation state or output value.

Many learning rules can be formulated as variants of a generalised Hebbian learning rule [101] expressed as $\Delta w_{ji} = l_x(\nu_j(t), d_{\nu_j}(t)) l_y(o_i(t), w_{ji})$, where Δw_{ji} is the change or adjustment for weight w_{ji}, $l_x : (\nu_j(t), d_{\nu_j}(t)) \rightarrow \mathbb{R}$ with $d_{\nu_j}(t)$ representing a desired state for the jth unit, and $l_y : (o_i(t), w_{ji}) \rightarrow \mathbb{R}$. For example, setting $l_x(\nu_j(t), d_{\nu_j}(t)) = \eta_j(t)(d_{\nu_j}(t) - \nu_j(t))$ and $l_y((o_i(t), w_{ji}) = o_i(t)$, results in $\Delta w_{ji} = \eta_j(t)(d_{\nu_j}(t) - \nu_j(t))o_i(t)$ which is known as the delta rule or Widrow-Hoff rule [9, 101], and where $\eta_j(t)$ is the learning rate parameter.

8. **The environment**. The NN operates within some environment. To interact with this environment the NN must accept inputs and provide outputs with specific characteristics suitable for the environment. In addition, the environment can either be stationary, pseudostationary, or not stationary. A stationary environment is one whose statistical measures do not vary with time [9, 108]. A pseudostationary process can be considered stationary over a suitability *short* interval of time ([9], pg. 83-84).

Using this the components of this framework, the FFNN selected for this thesis consists of a set of processing units, which, along with the output function, (item 3), the propagation rule, (item 5), and the activation rule, (item 6), are described in Section 3.2.2. Also, the FFNN chosen for this thesis is a static NN, in that the NN weights are not adjusted after learning has stopped, that is, the weights are independent of time t. Other specific characteristics of the FFNN selected for this thesis include, not using the current state of activation, (item 2) of the processing units to determine the next state, that is,, $o_j(\nu_j) = \nu_j$. In addition, the pattern of connectivity, (item 4), is that of a fully connected single hidden layer FFNN, which is described in Section 3.2.3. The learning rule, (item 7), chosen for the FFNN supports the learning paradigm know as supervised learning, which is described in section 3.3.

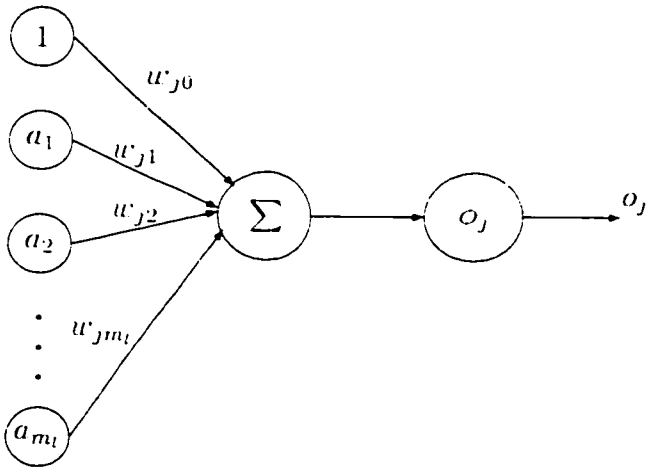


Figure 3.1. A single processing unit or neuron used in the FFNN selected for this thesis. The neuron is based on transforming the weighted sum of its inputs. Note that the bias term w_{j0} has been absorbed into the input.

Finally, for this thesis the environment for the FFNN is assumed to be stationary and is specified over the spaces $\mathcal{X}$ and $\mathcal{Y}$. It should be noted that a learning paradigm refers to a specific example or type of learning that the NN is attempting within its defined environment. In this view there are two commonly recognized learning paradigms, supervised and unsupervised learning ([9], pg. 52).

3.2.2 The processing units

A FFNN consists of set of interconnecting processing units, referred to as as neurons. The neuron used for the FFNN considered in this thesis is shown diagramatically in Figure (3.1).

In terms of the general framework of NN components presented in section 3.2.1, the jth neuron, shown in Figure (3.1), consists of a linear output function, specifically this implies, $o_j = \Phi_j(\nu_j) = \nu_j = o(\nu_j)$. The propagation rule chosen for the jth neuron is simply given by $\nu_j(t) = \sum_{i=0}^{n} w_{ji} a_i(t)$, where a_i is used to denote the ith input to the neuron. It should be noted that the $i = 0$ input is a constant and the

corresponding product $a_0 w_{j0}$ is referred to as the bias for the jth neuron, which can also be denoted as b_j. The bias determines the centre of the thresholding interval of o. This bias term has been transferred to appear as an auxiliary input simply for convenience in analysis.

The operation of the jth neuron, shown in Figure (3.1), can then be expressed using

$$\begin{aligned} o_j &= o_j \left(\sum_{i=1}^{m_l} w_{ji} a_i + w_{j0} \right) . \\ &= o_j \left(\sum_{i=1}^{m_l} w_{ji} a_i + b_j \right) = o_j \left(\mathbf{a}^T \mathbf{w} + b_j \right) . \\ &= o_j \left(\sum_{i=0}^{m_l} w_{ji} a_i \right) = o_j \left(\mathbf{a}_a^T \mathbf{w}_a \right) . \quad a_0 = 1. \end{aligned} \tag{3.3}$$

where $o_j \in \mathbb{R}$ is the output of this jth neuron, $o_j(\cdot)$ is its activation function $o : \mathbb{R}^{m_l} \rightarrow \mathbb{R}$. m_l is the number of inputs, $w_{ji} \in \mathbb{R}$ is the weight connecting the ith input $a_i \in \mathbb{R}$ to this jth neuron, and $w_{j0} = b_j$ is the bias input, i.e. with a constant input $a_0 = 1$. The vector $\mathbf{a}_a^T$ and $\mathbf{w}_a$ are the augmented vectors of $\mathbf{a}^T$ and $\mathbf{w}$ obtained by absorbing the bias term, and where the vectors are normally assumed to be column vectors and the superscript T is the transpose operator.

Of particular importance to the operation of the neuron is the specific properties of the activation function $o_j(\cdot)$. Leshno *et al.* [109] have provided an important result showing that if the activation function can be characterised as a continuous non-polynomial function that is locally bounded, then it is capable of allowing a FFNN to be a universal approximator.

Note that the non-polynomial activation function also requires that it have a finite number of open intervals [47, 109]. Also, a single neuron does not, by itself, form an universal approximator, but a set of neurons, with a reasonably general pattern of connectivity, can approximate any continuous function to an arbitrary degree of accuracy [109], given enough neurons. This results indicates that the universal approximation property is not entirely due to the activation function but is more a

function of the pattern of connectivity ([97]. pg. 239. [47]).

One frequently used class of activation functions with these properties is the class of sigmoidal functions defined by [47]

$$\infty > \lim_{u \to \infty} \sigma(u) = \sigma^+ > \sigma^- = \lim_{u \to -\infty} \sigma(u) > -\infty. \tag{3.4}$$

where σ^+ and σ^- represent the upper and lower bound of $\sigma(u)$. A few of the specific forms from this class of sigmoidal functions are the logistic function.

$$\sigma(u) = \frac{1}{1 + e^{-wu}}. \tag{3.5}$$

or the hyperbolic tangent

$$\tanh(wu) = \frac{e^{wu} - e^{-wu}}{e^{wu} + e^{-wu}} = 2\sigma(2u) - 1. \tag{3.6}$$

which is simply a scaled and offset version of the logistic function.

The results of Leshno *et al.* [109] show that almost any arbitrary activation function can be used to form a FFNN capable of universal approximation. In these terms, it should be noted that the output neuron commonly uses a linear activation function $\phi(u) = u$ that is not bounded and is also a first order polynomial. This linear output neuron is simply used to provide a weighted sum of the output of hidden neurons and does not remove the universal approximation property of a FFNN.

3.2.3 The FFNN architecture

A number of NN architecture can be constructed using neurons having the structure shown in Figure (3.1) [9, 47]. One of the most common is a fully connected multilayer architecture referred to as a FFNN or a multilayer perceptron [9, 47].

The pattern of connectivity for this architecture can be characterised as consisting of L layers of neurons. Each jth neuron in the lth layer accepts the output of all the neurons in the $(l-1)$ layer. The jth neuron in the lth layer then has its output sent to all the neurons in the next $(l+1)$ layer.

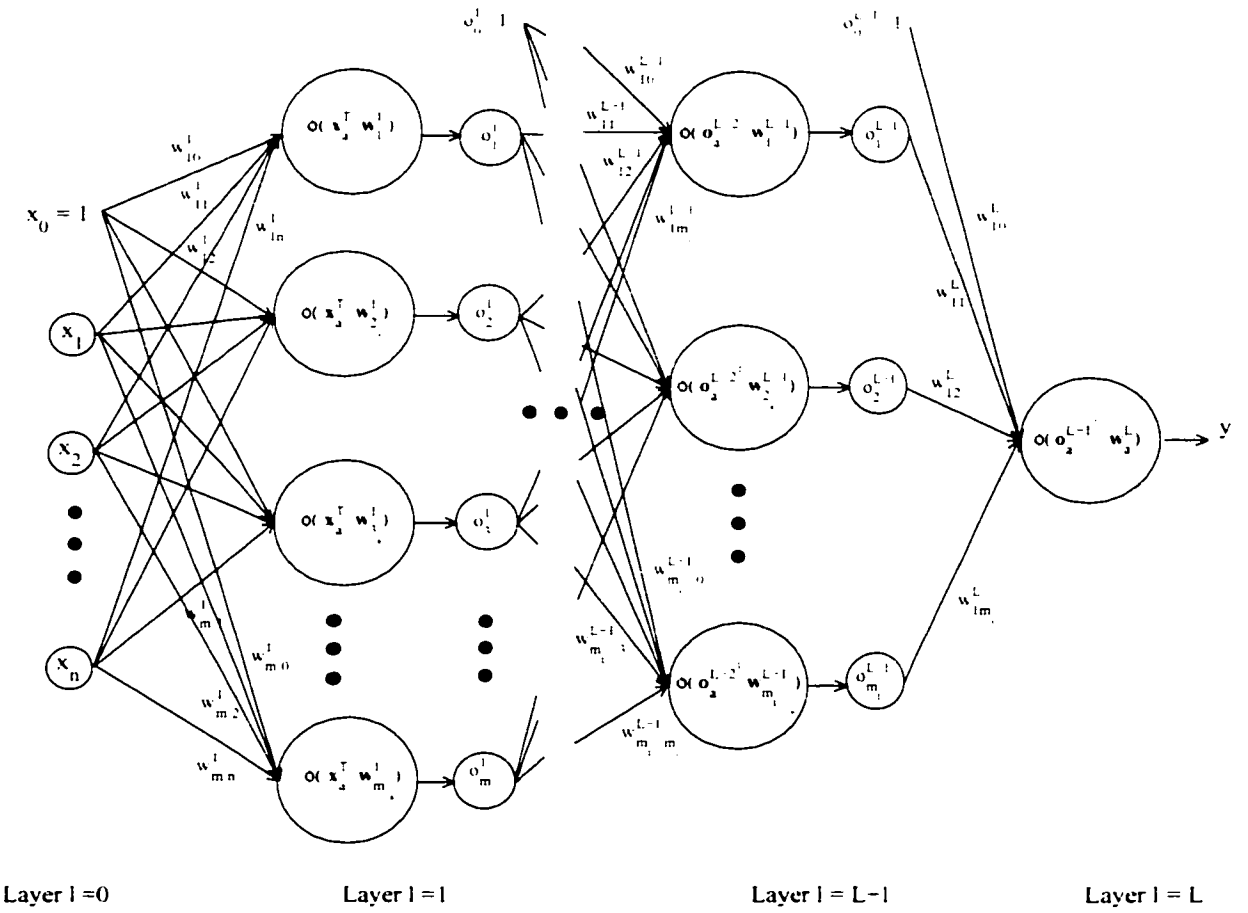


Figure 3.2. A fully connected L layered FFNN based on the neuron structure shown in Figure (3.1). Note that not all the weights are labeled.

This pattern of connectivity for the neurons is shown in Figure (3.2) which, for a single output FFNN, can be expressed as a nested set of activation functions $o(\cdot)$ [9]. Assuming L layers, this nested set of activation functions can be expressed as

$$\hat{y}(\mathbf{x}, \mathbf{w}) = o_L\left(\sum_{k=0}^{m_{L-1}} w_{1k}^{L} O_{(L-1)k}\left(\sum_{j=0}^{m_{L-2}} w_{kj}^{L-1} O_{(L-2)j}\left(\cdots O_{1p}\left(\sum_{i=0}^{m_0=n} w_{pi}^{1} x_i\right)\right)\right)\right). \tag{3.7}$$

where $\hat{y}$ is shown as a function of the input $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ and the vector of network weights $\mathbf{w} \in \mathbb{R}^{n+m_1+m_2+\ldots+m_L+L}$, layer $l = 0$ is the input layer, layer $l = L$ is the output layer, and the layers $l = 1, 2, \ldots, L-1$ are referred to as the hidden layers. The weights w_{ji}^{l} refer to the weight connecting the ith input to the jth neuron in the

lth layer. m_l is the the number of neurons in the lth layer. Though o_{lk} refers to the kth activation function in lth layer, it is common to use the same functional form for all the hidden layer activation functions, thereby allowing the kth subscript to be dropped. It should be noted that in Equation (3.7) the bias terms in the hidden layers are formed with $o_{l0}(\cdot) = 1$, $l = 1, 2, \ldots, L-1$, which then serve as the input to the next, i.e. $l+1$ layer.

In selecting the number of layers L to use for a FFNN, the notion suggested by the precept of Occum's razor has been typically invoked: of all the architecture that can provide the needed level of approximation accuracy, the simplest architecture, that is, the architecture with the fewest number of layers and neurons per layer, should be selected. It is well known that simpler FFNN architectures tend to be less prone to overfitting and to provide improved generalisation when compared to more complex FFNN architectures [9]. In this setting, having $L = 1$, i.e. no hidden layers, represents one of the most simplest architectures, but unfortunately, it has been shown that a FFNN with $L = 1$ is limited in the class of functions it can approximate [9, 47, 97]. Obviously, setting $L = 2$ represents the next simplest architecture and fortunately it has been shown that a FFNN, having an appropriate set of activation functions in the hidden layer and a linear output layer, is capable of approximating any continuous function to any required degree of accuracy [40–42]. For this reason, this thesis uses, as have many other reported applications [18, 26, 43, 110], a FFNN architecture with a single hidden layer.

Again, it should be noted that though a single layer hidden FFNN has been shown to be an universal approximator, it may not be the optimum architecture in terms of the total number of neurons. For example, the use of a FFNN having two hidden layers, $L = 3$, can, in some simple problems, provide an approximation with zero error, as measured by the L_2-norm. In contrast, a FFNN with a single hidden layer having a finite number of neurons, will provide an approximation with some degree of error ([47], pg. 108-109). In addition, it is known that it is possible to find an approximation using a more complex FFNN which results in having a generalisation error

the is equivalent to that provided by a simpler FFNN. This particular result requires determining a suitable initialisation of the network weights and then determining the instance, during learning, when the generalisation error begins to increase, so as to terminate learning [111].

3.2.4 The single hidden layer FFNN with shape-tunable neurons

The architecture of a single hidden layer FFNN is shown in Figure (3.3). The operation of this FFNN can be expressed using

$$\hat{y}(\mathbf{x}, \mathbf{w}) = o_2 \left(\sum_{k=0}^{m_1} w_{1k}^2 o_{1k} \left(\sum_{i=1}^{m_0=n} w_{ki}^1 x_i + w_{k0}^1 \right) \right).$$

where o_2 is selected to be a linear function, so that the operation of the FFNN is then given by

$$\begin{aligned} &= \sum_{k=1}^{m_1} w_{1k}^2 o_{1k} \left(\sum_{i=1}^{n} w_{ki}^1 x_i + w_{k0}^1 \right) + w_{10}^2. \qquad (3.8) \\ &= \vec{o}^T(\mathbf{x}^T \mathbf{w}_k^1 + w_{k0}^1)\mathbf{w}_1^2 + w_{10}^2. \qquad k = 1, 2, \ldots, m_1. \\ &= \vec{o}_a^T(\mathbf{x}_a^T \mathbf{w}_{k_a}^1)\mathbf{w}_a^2. \qquad k = 0, 1, 2, \ldots, m_1. \end{aligned}$$

where $\vec{o}_a^T(\cdot) = [o_{10} = 1, o_{11}(\mathbf{x}_a^T \mathbf{w}_{1_a}^1), \ldots, o_{1m_1}(\mathbf{x}_a^T \mathbf{w}_{m_{1_a}}^1)]$ is the augmented row vector of hidden layer activation functions, $\mathbf{w}_a = [w_{10}, w_{11}, \ldots, w_{1m_1}]^T$ is the augmented weight vector without the superscript identify the layer, and $\mathbf{x}_a = [1, x_1, x_2, \ldots, x_n]^T$ is the augmented FFNN input vector.

Now instead of expressing the operation of the single hidden layer FFNN using Equation (3.8), consider the following alternative, but mathematically equivalent, form

$$\begin{aligned} \hat{y}(\mathbf{x}, \mathbf{w}) &= \sum_{k=1}^{m_1} \left[c_k^1 o_{1k} \left(\sum_{i=1}^{n} w_{ki}^1 x_i + w_{k0}^1 \right) + b_k^1 \right]. \qquad (3.9) \\ &= \sum_{k=1}^{m_1} a_k^1. \end{aligned}$$

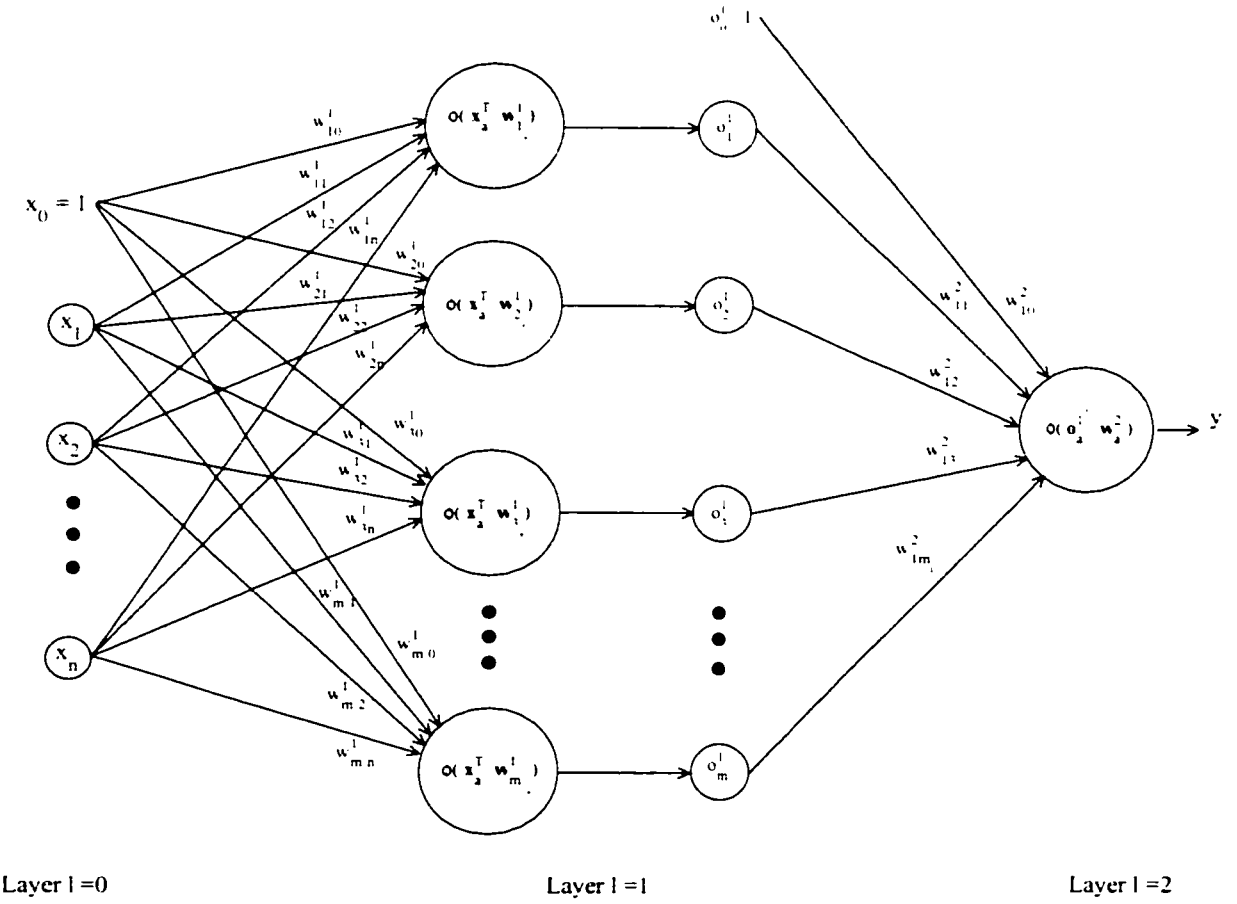


Figure 3.3. A single hidden layer FFNN architecture.

or in vector form

$$
\begin{aligned}
&= [\vec{o}^T(\mathbf{x}^T\mathbf{w}_k^1 + w_{k0}^1) + (\mathbf{b}^1)^T]\mathbf{c}^1. \qquad k = 1.2.\ldots.m_1. \\
&= [\vec{o}^T(\mathbf{x}_a^T\mathbf{w}_{ka}^1) + (\mathbf{b}^1)^T]\mathbf{c}^1. \qquad k = 0.1.2.\ldots.m_1. \\
&= (\mathbf{a}^1)^T\mathbf{k}.
\end{aligned}
$$

where the equivalence in this alternative form is given by setting $w_{1k}^2 = c_k^1$, $w_{10}^2 = \sum_k b_k^1$, and where $\mathbf{k} = [1.1.\ldots.1]^T$ is a $(m_1 \times 1)$ column vector of ones. This alternative expression has taken the bias term for the output neuron and simply distributed it over the outputs of the hidden neurons. In addition, the input weights for the output neuron have been moved to the output of the hidden neurons.

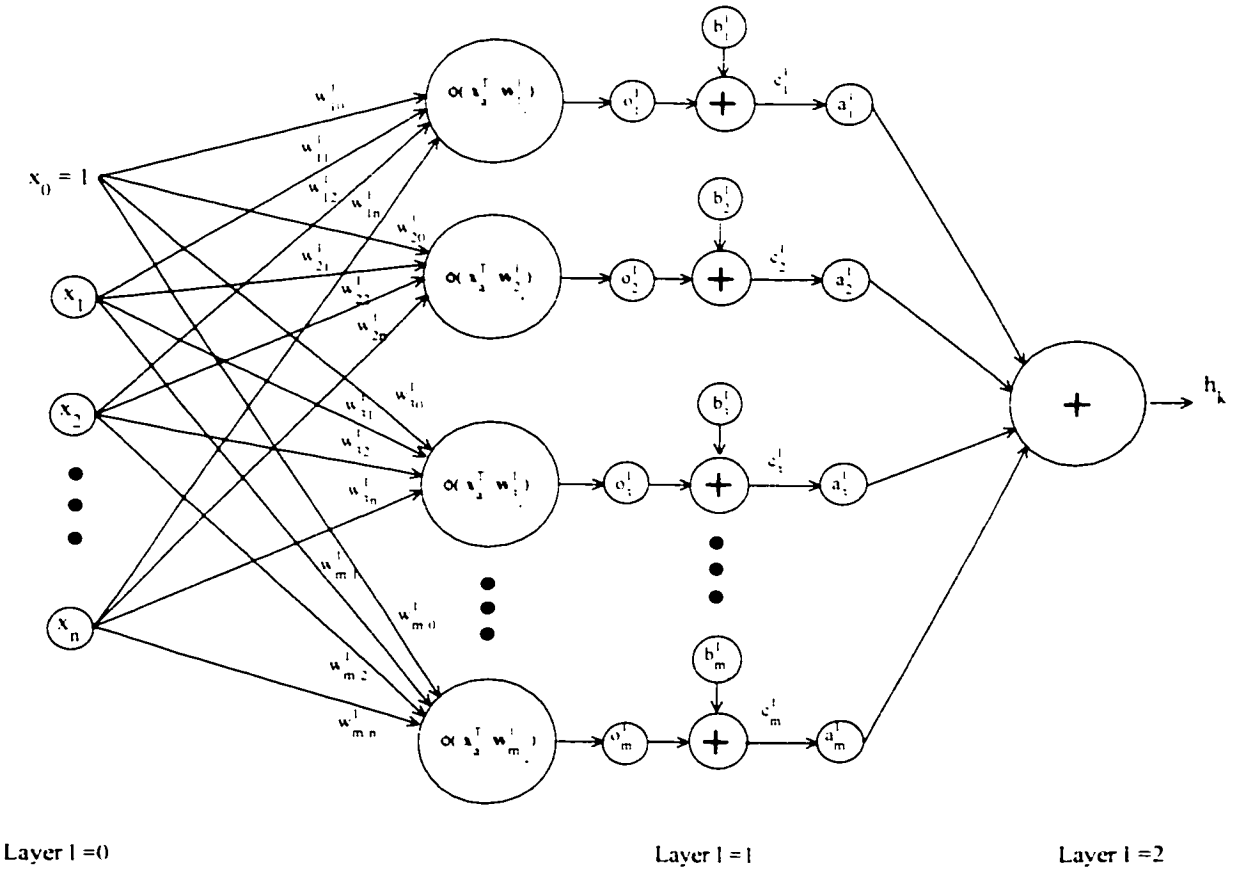


Figure 3.4. The instrument calibration model used in this thesis is based on a single hidden layer FFNN architecture with *shape-tunable* neurons in the hidden layer. Note that this architecture is mathematically equivalent to the single layer architecture shown in Figure (3.3).

Mathematically, Equation (3.8) and Equation (3.9) are equivalent, but the FFNN architecture has changed form slightly to that shown in Figure (3.4). The hidden layer now consists of m_1 neurons whose operation is given by the square bracketed term on the righthand side of Equation (3.9). Note that the output layer now consists of a linear neuron whose input weights, w_k^2, for $k = 1, 2, \ldots, m_1$, are simply set to unity.

These hidden neurons resemble the *shape-tunable* neurons described by Chen *et al.* [112], but appear without a specific slope tuning parameter. The slope tuning

parameter $m_t \in \mathbb{R}$ would appear as an additional argument in the hidden layer activation functions o. that is. for the jth hidden neuron. $o_{1j}(m_t, \mathbf{x}_a^T \mathbf{w}_{j_a}^1)$ [112].

The primary difference between a *shape-tuning* neuron and the conventional hidden neuron can be more precisely expressed by assuming for both neurons the case of $o(\cdot)$ belonging to the class of sigmoidal functions defined by Equation (3.4). In this case. the upper and lower bounds of a conventional neuron's response or output would be given by (σ^-, σ^-). whereas the shape-tuning neuron would have its upper and lower bounds given by $(c\,\sigma^- + b, c\,\sigma^- + b)$. For example. using the hyperbolic tangent function as the activation function. the output bounds of conventional neuron are given by $(\sigma^- = +1, \sigma^- = -1)$. whereas the shape-tuning neuron would have its upper and lower bounds given by $(b+c, b-c)$. Similarly. for the logistic function. the conventional neurons upper and lower output bounds are $(1, 0)$. whereas the shape-tuning neuron has it upper and lower output bounds at $(b+c, b)$. Given that $b, c \in \mathbb{R}$. it is easy to see that for the shape-tuning neuron. the upper and lower bounds can be made be equivalent for either the hyperbolic tangent or the logistic form of the activation function.

The claimed advantage in using the shape-tuning neuron was made empirically by showing that the shaping-tuning neurons. based on functions from the class of sigmoidal functions. appear to have *superior flexibility* and greater *nonlinearity capacity* than the non shape-tuning neurons. In addition. FFNN architectures using these shape-tuning neurons were shown to convergence to an approximate solution more quickly than that of an equivalent architecture using non shape-tuning neurons.

3.3 FFNN Learning

NN learning was briefly characterised in Section 3.2.1 as a process that adjusts the weights of a NN according to a given learning rule. This characterisation of learning encompasses many different forms of NN learning. It is not the intent of this section to review all these forms of learning but to discuss one specific type of

learning algorithm drawn from the learning paradigm known as supervised learning. In particular, this discussion will focus on the features of supervised learning that are relevant to instrument calibration and calibration transfer.

This discussion begins by drawing an analogy between supervised FFNN learning and instrument calibration where it is shown that the generalisation error ϵ_g, used to assess the adequacy of the FFNN approximation, is equivalent to the SEP used in instrument calibration. It is then noted that the SSE in the approximating the calibration data, e_D, has a relationship to ϵ_g that can be described as stochastic. The importance of noting this relationship is formed by the realisation that supervised learning is guided by ϵ_D, but that it is ϵ_g that is used to determine the acceptability of the FFNN approximation of the calibration model. Therefore, it becomes important to understand the factors influencing this error relationship.

To understand this error relationship and its relevance to calibration, this section uses the view that supervised FFNN learning is a nonlinear numerical optimisation process. The process of supervised learning then becomes a search for a minimum over a complex error surface. The techniques and the associated difficulties in finding acceptable minima are then discussed in terms of FFNN learning and calibration. This discussion then allows presenting a framework from which the relationship between ϵ_D and ϵ_g can be more clearly seen.

3.3.1 Supervised FFNN learning

Supervised learning and instrument calibration

In chapter 1 the true instrument calibration model h was viewed abstractly as a mapping $h : \mathbf{x} \rightarrow y$. Pragmatically, h needs to be approximated by $\hat{h}$ which is obtained by performing an instrument calibration. A conventional instrument calibration consists of first selecting a model for $\hat{h}$ and then estimating the model parameters using a set of calibration data $D = \{(\mathbf{x}_j, h(\mathbf{x}_j)) : j = 1, 2, \ldots, N_j\}$. The goal in calibration is to minimise a calibration error $e = d(h(\mathbf{x}), \hat{h}(\mathbf{x}))$, over all $\mathbf{x} \in \mathcal{X}$.

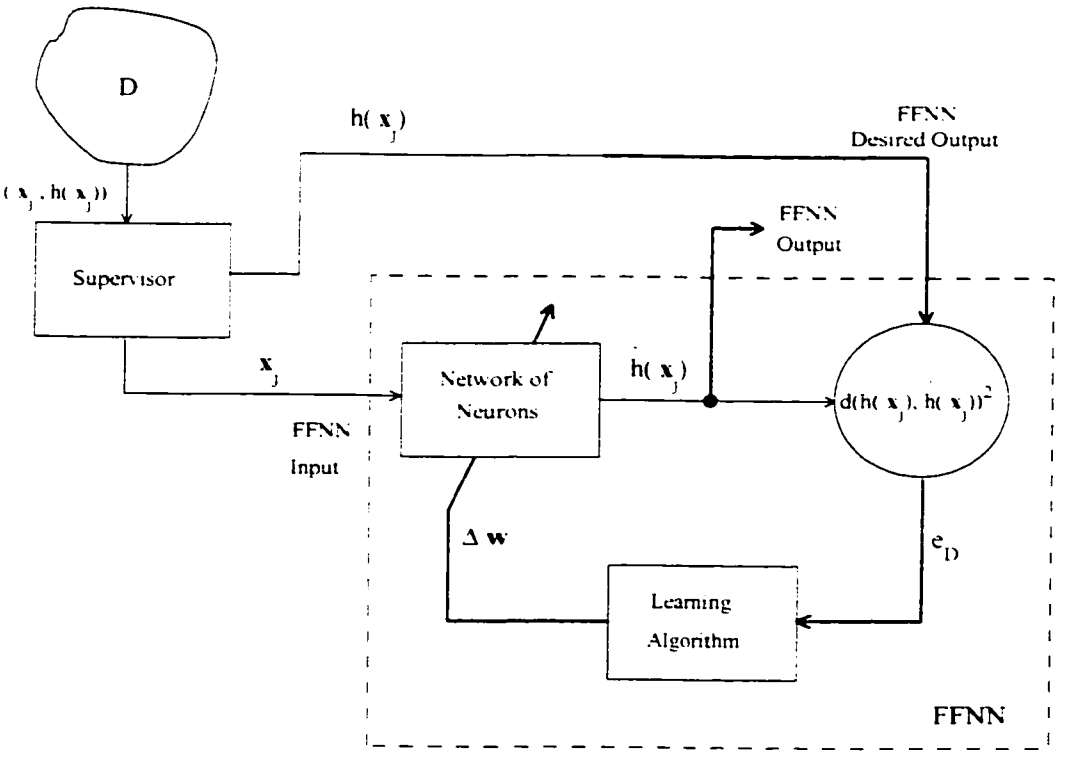


Figure 3.5. A block diagram representing the process of supervised learning for a FFNN.

In this section it will be shown that the calibration process is analogous to the FFNN learning paradigm known as supervised learning. as well. it will be shown that the calibration error ϵ is analogous to the error know as the FFNN generalisation error ϵ_g.

To see these analogies. consider the block diagram of the supervised learning process shown in Figure (3.5). The essential process of supervised learning consists of a supervisor. also referred to as a teacher. that selects a jth training example. $(\mathbf{x}_j, h(\mathbf{x}_j)) \in D$. from the training set D and presents to the FFNN. the example of the jth input $\mathbf{x}_j$ and the corresponding jth desired output $h(\mathbf{x}_j)$. The intent in presenting these examples of input and output values to the FFNN is to *show* the FFNN. by example. the correct association between the inputs and outputs. This presentation of input and output values then allows the FFNN to *learn* the association between all

inputs and outputs as exemplified in D. In other words, the FFNN attempts to learn the mapping $h : \mathbf{x}_j \rightarrow h(\mathbf{x}_j) = y_j$, so as to achieve the minimisation of the data error $e_d = d(h(\mathbf{x}_j), \dot{h}(\mathbf{x}_j))^2$, over all $(\mathbf{x}, h(\mathbf{x})) \in D$ and where $\dot{h}(\mathbf{x}_j)$ represents the output of the FFNN given the jth input $\mathbf{x}_j$. This minimisation is achieved by the learning algorithm appropriately adjusting the weights $\mathbf{w}$ of the network by an amount $\Delta\mathbf{w}$.

Note the similarity and subtle difference between the process of instrument calibration and FFNN learning. The similarity is seen in that both processes are attempting to approximate the true mapping h using only the data provided in the set D. The difference in the processes, as presented here, is that the intent of instrument calibration is reduce the calibration error e over all $\mathcal{X}$, whereas in supervised FFNN learning, only the error e_D is directly reduced at $(\mathbf{x}, h(\mathbf{x})) \in D$.

It is clear that e and e_D are not the same and this difference in e_D and e, given that the FFNN is used as the instrument calibration model $\dot{h}$, then begs an answer to a very important question: what is the magnitude of the error at $(\mathbf{x}, h(\mathbf{x})) \notin D$ given that a FFNN has determined $\dot{h}$ over $(\mathbf{x}, h(\mathbf{x})) \in D$ with an error given by e_D? Fortunately, this question has been examined by others, (Fine [47], chapter 7), and aspects of the answer relevant to calibration are discussed in section 3.3.2. For the present discussion, it suffices to state that, in terms of FFNN learning, the error over $\mathbf{x} \in \mathcal{X}$ is referred to as the generalisation error e_g and is commonly expressed using $e_g = d(h(\mathbf{x}), \dot{h}(\mathbf{x}))^2$, which, ignoring the squaring of the distance metric, resembles the definition of the calibration error e and becomes identical if $d(\cdot, \cdot)$ is defined to be the same metric for both errors. In addition, it should be noted that e and e_g represent *true* errors and in practice estimates are obtained with an appropriately sized validation or test data set [113–115].

As is discussed in section 3.3.2, the relationship between e_g and e_D is complex [116–118] and exhibits random characteristic [119], thereby eluding formulation of a direct analytical relationship. Fortunately again, it is known that as $\overline{\overline{D}} \rightarrow \infty$, $e_D \rightarrow e_g$ [47, 115, 120]. In other words, given sufficient data, the minimisation of e_D will result

in an e_g that should be acceptably close to e_D, that is $d(e_g, e_D) \leq \epsilon$, where $\epsilon > 0$ represents the acceptable degree of closeness using the distance metric $d(\cdot, \cdot)$. Again, some guidance as to the likelihood of achieving $d(e_g, e_D) \leq \epsilon$ for some D has been provided by others [9, 47, 121, 122]. Therefore, the remainder of this section will focus on the process of supervised learning, that is learning to provide the mapping h : $\mathbf{x} \rightarrow h(\mathbf{x}) = y$, for $(\mathbf{x}, h(\mathbf{x})) \in D$ with the understanding that though e_D is being minimised, e_g can be made to be acceptable close to e_D.

Given this relationship between e_g and e_D, the problems in using a FFNN as a calibration model can be already be anticipated. In order to insure an acceptable e_g, or alternatively an acceptable calibration error e, requires a *sufficient* representation in D. The problems, of course, will revolve around answering the question of whether or not D provides a sufficient representation and whether or not the estimate of e_g has acceptable accuracy. Determining an answer to these questions may require using an excessive quantity of data, a prospect that is not looked at favourably in instrument calibration.

The learning error surface

It will be instructive to visualise the process of learning using the concept of a hypersurface surface. A hypersurface is nothing but a higher dimensional version of common surface in a three dimensional space, $\mathbb{R}\times\mathbb{R}\times\mathbb{R} = \{(w_1, w_2, e) : w_0, w_1, e \in \mathbb{R}\}$, such that the *height* of the surface at (w_0, w_1) is given by e. The extension to higher dimensions $\mathbb{R} \times \mathbb{R} \times \ldots \times \mathbb{R} = \{(\mathbf{w}, e) = (w_1, w_2, \ldots, w_p, e) : w_i, e \in \mathbb{R}, i = 1, 2, \ldots, p\}$, does not invalid any insight gained with the three dimensional case.

In the case of supervised learning, the *height* of the hypersurface is given by the error $e_D = d(h(\mathbf{x}_i), \hat{h}(\mathbf{x}_i))^2$, where the *height* is the perpendicular distance from the plane formed by the orthogonal axes $w_1, w_2, \ldots, w_p$ and the point $(\mathbf{w}, e_D)$, where w_i for $i = 1, 2, \ldots p$ represent the p parameters or weights of the FFNN, such that $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^p$. This hypersurface over all $\mathbf{w}$ is known as the error surface of the FFNN. It should be pointed out that even though e_D is a surface over all the weights

of the FFNN. ϵ_D is also dependent on the data set D. If a different D is chosen, ϵ_D will change. This particular point is important in that it will help to understand the complexity of the relationship between ϵ_D and ϵ_g and the role this relationship will have in instrument calibration.

The utility in using the error surface is that the supervised learning process can be visualised as the movement of a point on the surface to another point. Since supervised learning is attempting to minimise ϵ_D by appropriately adjusting the weights by $\Delta\mathbf{w}$, supervised learning can be viewed as a search for the *lowest* point on the error surface. This lowest point, referred to as the global minimum, then corresponds to a set of weights $\mathbf{w}^*$ that results in $\hat{h}^*$ such that

$$\begin{aligned}\epsilon_D^* &= \min_{\mathbf{w}\in W}\left(d(h(\mathbf{x}),\hat{h}(\mathbf{x}))^2\right), \qquad \forall(\mathbf{x},h(\mathbf{x}))\in D,\\ &= d(h(\mathbf{x}),\hat{h}^*(\mathbf{x}))^2 \leq d(h(\mathbf{x}),\hat{h}(\mathbf{x}))^2,\end{aligned}$$

where it should be noted that for a single hidden layer FFNN using a sigmoidal class of activation functions for the hidden layer and a linear output layer, $\mathbf{w}^*$ is not unique [47, 123].

Viewing the process of supervised learning as a search for $\mathbf{w}^*$ over an error surface has been used by others to gain insight into the potential difficulty in locating these global minima. These difficulties, of course, then translate directly into the potential difficulties of instrument calibration and calibration transfer. Therefore, reviewing the insight provided by others will help in understanding the use of a FFNN as an instrument calibration model.

One obvious insight is the realisation that it is the form of the error surface that will contribute to the difficulty in locating these global minima. The existence of multiple global minima [47, 123] implies that the error surface is not at all *simple*. More accurately, it is known that for the FFNN considered in this thesis, that the error surface can be characterised as complex, consisting of long narrow troughs and many regions that are essentially flat [124]. Contributing to this complexity is the existence of multiple local minima. Auer *et al.* [125] have shown that the error surface

for a NN consisting of a single neuron, using a data set with $N = \bar{\bar{D}}$ samples where $\mathbf{x}_i \in \mathbb{R}^n$, will exhibit $\left(\frac{N}{n}\right)^n$ local minima. Given a FFNN with multiple hidden neurons, the existence of large number of local minima is a very real concern that imposes a significant problem for most methods that are searching for acceptable minima in the error surface.

Under certain cases it is possible for an error surface, as given by the SSE, of a FFNN not to have local minima. This has been shown to occur if the FFNN is capable of exact approximation of the data D such that each sample $(\mathbf{x}, h(\mathbf{x})) \in D$ is unique [126]. Exact approximation of D is possible with a single hidden layer FFNN having $\bar{\bar{D}} - 1$ hidden nodes. In addition, the density of local minima, that is the number of minima per unit volume in $\mathcal{W}$, can also be reduced by using a logarithmic error function [127].

Local minima present a problem in that they may correspond to generalisation errors e_g that are unacceptable. In addition, most search methods can not distinguish the difference between a local and global minimum without further extending the search. Fortunately, many of the local minima can correspond to acceptable levels of e_g [47] so as to reduce the motivation to search for a lower minimum. It also becomes apparent that as the level of acceptable error is lowered, the number of acceptable minima will decrease, thereby increasing the difficulty in finding acceptable minima.

As a simple example, Figure (3.6) illustrates the characteristics of a typical error surface for a single neuron for different data sets over the same $h(x)$. As the figure shows, e_D is dependent on the data set D. It should be noted that for any fixed number of data samples, that is $\bar{\bar{D}} = N$, randomly selected from $N \subset \mathcal{X}$, the resulting error surfaces will all be different so long as the elements of any two data sets are different.

Figure (3.6) also shows an important characteristic of the error surface: that the error surface becomes *smoother* as the number of data samples increases. Specifically, Figure (3.6) is graphically showing how the error surface defined by e_D is approaching some asymptotic form. If this error surface is scaled by $1/N$ then Figure (3.6) is more

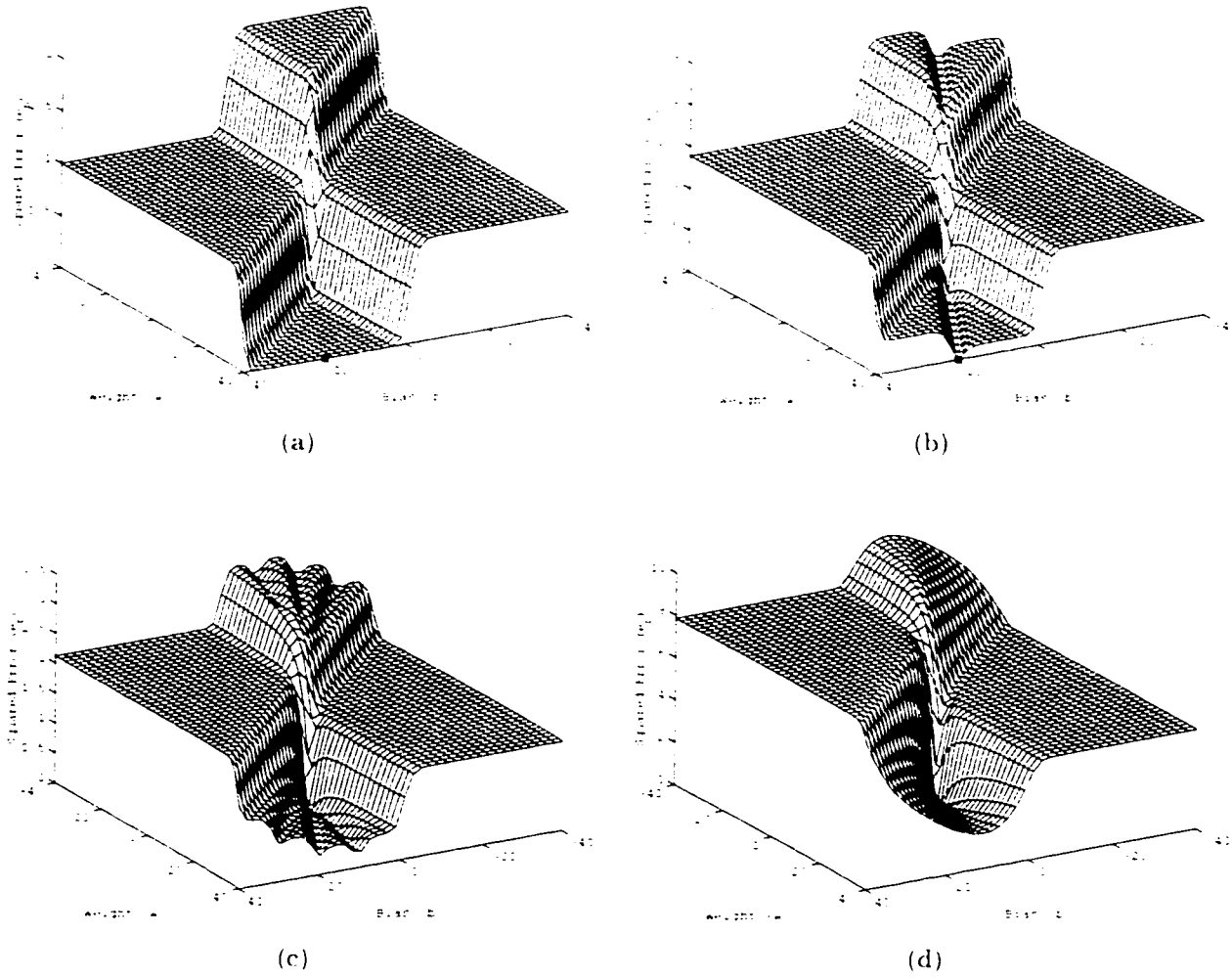


Figure 3.6. Examples of the error surface for a single neuron with a single input having only two weight parameters, w and b the bias. The error surface ϵ_D is given by the squared l_2-norm over the data set $D = \{(x_i, h(x_i)) : i = 1, 2, \ldots, N\}$, and where $h(x) = 3x^3 - 5x^2 + 3x$, $X = [0, 1]$. In (a), $N = 2$, and in (b) $N = 3$, in (c) $N = 5$, and in (d) $N = 20$, where $x_i = (i - 1)/(N - 1)$.

accurately showing how $\epsilon_D \rightarrow \epsilon_g$ as $\bar{\bar{D}} \rightarrow \infty$.

In terms of instrument calibration, the search for a minimum should ideally use the error surface given when $\bar{\bar{D}} \rightarrow \infty$ in that the minimum on the ideal surface corresponds directly to the generalisation error ϵ_g. In reality, $\bar{\bar{D}}$ is finite so that the minimum is searched over a less ideal error surface. By visualising the overlaying of this less ideal error surface with the ideal surface, it becomes more clear as to why the relationship between ϵ_D and ϵ_g is described as being stochastic.

Supervised learning as a nonlinear optimisation

It becomes clear that a successful search for $\mathbf{w}^*$ is highly unlikely in the general case of multilayered FFNN using the SSE as an error measure for ϵ_D. More realistically, a search will find one of many local minima. The success in finding an acceptable local minimum over the error surface will depend not only on the complexity of the error surface but also on the method used to perform the search. Though no *best* method to determine an acceptable minimun [47, 128], for all types of problems and data sets has been established, a number of methods based on numerical or iterative nonlinear optimisation techniques have been developed and used successfully in supervised learning. In particular, the use of descent techniques [129] have become methods of choice when implementing supervised learning algorithms.

Though it is quite common to view the supervised learning problem as a numerical nonlinear optimisation problem, ([9], pg. 234-245, [47], pg. 129-136), descent techniques are not the only means to perform the optimisation, techniques such as direct search methods [129] can also be used.

To be able to place the $D^{(n)}$ learning algorithm within the context of these descent methods the essential elements of these methods will now be considered.

To reduce the ambiguity in describing these descent methods over various error surface, let $C(\mathbf{w}, D_k) = \epsilon_{D_k}$ denote the error surface over $\mathbf{w} \in \mathcal{W}$ for the kth data set $D_k = \{(\mathbf{x}_i, h(\mathbf{x}_i) : i = 1, 2, \ldots, N_k\}$, where $N_k = \bar{\bar{D}}_k$. The notation $C(\mathbf{w})$ will also be

used for the error surface to simplify the notation but where the dependance on D is understood to be present. Since a search for the lowest point on $C(\mathbf{w}, D_k)$ must begin from some point, let $\mathbf{w}_j(n = 0)$ represent the coordinates of the jth starting point. Then let $\mathbf{w}_j(n)$ represents the state of $\mathbf{w}$ at the nth time instant,where $n = 0, 1, 2, \ldots$, given that the search began at the jth starting point $\mathbf{w}_j(0)$. Also, let $\mathbf{w}_l^*$ designate the coordinates of the lth global minimum on $C(\mathbf{w}, D_k)$, i.e. there is more than one global minimum. In addition, given that the search can be terminated before finding the global minimum or that the search may find a local minimum, let $\mathbf{w}_j^+(n)$ represent the coordinates where the search, that began from jth starting point, is stopped at the nth instant. Also, let $\hat{h}(\mathbf{x}; \mathbf{w}_j(n))$ describe the output of the FFNN for $\mathbf{x} \in \mathcal{X}$ given a set of weights set to $\mathbf{w}_j(n)$, and let $\hat{h}^*$ and $\hat{h}^+$ denote the FFNN having its weights set to a $\mathbf{w}^*$ and $\mathbf{w}^+$, respectively, without regard to the lth global minimum or jth starting point. Finally, define

$$\begin{aligned}\mathbf{g}(\mathbf{w}(n)) = \nabla_{\mathbf{w}} C(\mathbf{w}, D_k)|_{\mathbf{w}=\mathbf{w}(n)} &= \left.\frac{\partial C(\mathbf{w}, D_k)}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(n)} \\ &= \left[\frac{\partial C(\mathbf{w}, D_k)}{\partial w_1}, \frac{\partial C(\mathbf{w}, D_k)}{\partial w_2}, \ldots, \frac{\partial C(\mathbf{w}, D_k)}{\partial w_p}\right]^T_{\mathbf{w}=\mathbf{w}(n)}. \end{aligned} \tag{3.10}$$

as the gradient vector representing the first partial derivatives of $C(\mathbf{w}, D_k)$ with respect to the weights w_i evaluated at $\mathbf{w}(n)$. Also define

$$\begin{aligned}\mathbf{H}(\mathbf{w}(n)) = (\nabla_{\mathbf{w}})^2 C(\mathbf{w}, D_k)|_{\mathbf{w}=\mathbf{w}(n)} &= \left.\frac{\partial^2 C(\mathbf{w}, D_k)}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}(n)} \\ &= \begin{bmatrix} \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_1 \partial w_1} & \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_1 \partial w_p} \\ \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_2 \partial w_1} & \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_2 \partial w_2} & \cdots & \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_2 \partial w_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_p \partial w_1} & \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_p \partial w_2} & \cdots & \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_p \partial w_p} \end{bmatrix}_{\mathbf{w}=\mathbf{w}(n)} \end{aligned} \tag{3.11}$$

$$= [H_{ij}]. \qquad i, j = 1, 2, \ldots, p. \tag{3.12}$$

as the Hessian matrix of second order partial derivative of $C(\mathbf{w}, D_k)$ with respect to the weights evaluated at $\mathbf{w}(n)$ and where

$$H_{ij} = \frac{\partial^2 C(\mathbf{w}, D_k)}{\partial w_i \partial w_j} = H_{ji}. \tag{3.13}$$

Descent methods are based on performing the following iterative steps [129] that begin from some initial starting point at $\mathbf{w}_j(0)$:

1. From the current point at $\mathbf{w}_j(n)$, determine a descent direction $\mathbf{d}(n) = [d_1(n), d_2(n), \ldots, d_p(n)]^T$.

2. Determine a descent step length $\eta(n)$.

3. Perform the descent step using $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{d}(n) = \mathbf{w}(n) + \Delta\mathbf{w}(n)$, where $\Delta\mathbf{w}(n) = \eta(n)\mathbf{d}(n)$.

4. Determine if a termination criteria has been met so as to stop the descent at $\mathbf{w}(n+1)$.

An acceptable descent direction is given by any $\mathbf{d}(n)$ in which the error surface decreases, that is

$$C(\mathbf{w}(n+1)) = C(\mathbf{w}(n) + \eta(n)\mathbf{d}(n)) < C(\mathbf{w}(n)). \tag{3.14}$$

The condition under which this inequality is met can be determined by simply differentiating $C(\cdot)$ on both sides of the inequality in Equation (3.14) with respect to $\eta(n)$, that is determine the slope of $C(\cdot)$ with respect to the step length at the point $\mathbf{w}(n)$. This results in, by using the chain rule,

$$(\mathbf{d}(n))^T \mathbf{g}(\mathbf{w}(n)) < 0. \tag{3.15}$$

which is the well known descent condition [47, 129, 130]. The left hand side of Equation (3.15) is also by definition the directional derivative of $C(\mathbf{w}(n))$ in the direction of $\mathbf{d}(n)$ evalutated at the point $\mathbf{w}(n)$.

The direction providing the greatest decrease will depend on the unit of distance used along each of the coordinates axes of $\mathcal{W}$ and on the metric used to measure the distance. The distance metric that is commonly chosen is based on a form of the weighted l_2 norm. Therefore, using this metric, the distance between $\mathbf{w}(n+1)$ and

$\mathbf{w}(n)$ is

$$d(\mathbf{w}(n+1), \mathbf{w}(n)) = \left((\mathbf{w}(n+1) - \mathbf{w}(n))^T \mathbf{A}(n)(\mathbf{w}(n+1) - \mathbf{w}(n))\right)^{\frac{1}{2}}.$$
$$= \left(\Delta(\mathbf{w}(n))^T \mathbf{A}(n) \Delta\mathbf{w}(n)\right)^{\frac{1}{2}}. \tag{3.16}$$

where $\mathbf{A}(n)$ is a $(p \times p)$ matrix referred to as a metric matrix that by definition is required to be definite positive. If the axes are rescaled using $\mathbf{u} = \mathbf{B}\mathbf{w}$, than $\mathbf{A} = \mathbf{B}^T\mathbf{B}$. This rescaling can be performed at each iteration n and is the bases of variable-metric descent methods ([129], pg. 100).

Given the measure of distance provided by Equation (3.16), it can be shown, ([129], pg. 99-101), that the direction of steepest descent is given by

$$\mathbf{d}(n) = -(\mathbf{A}(n))^{-1}\mathbf{g}(\mathbf{w}(n)). \tag{3.17}$$

All descent methods can be differentiated by their choice of $\mathbf{A}(n)$, which determines the descent direction $\mathbf{d}(n)$, and the technique used to determine the descent step length $\eta(n)$ [129, 130]. In this context, some of the more well known descent techniques include those referred to as steepest descent, conjugate gradient, quasi-Newton, and the Levenberg-Marquardt methods. Many of these methods have been used to perform supervised learning in a FFNN [74, 131–137].

Though no *best* method has been established to determine $\mathbf{w}^*$ [47, 128], the use of methods based on steepest descent are frequently employed in supervised learning. The method of steepest descent is one of the simplist descent methods and is based on setting $\mathbf{A}(n) = \mathbf{I}$, where $\mathbf{I}$ is the $(p \times p)$ identity matrix. This results in a descent direction given by

$$\mathbf{d}(n) = -\mathbf{g}(\mathbf{w}(n)) = -\nabla_{\mathbf{w}} C(\mathbf{w}(n)). \tag{3.18}$$

which is simply the negative gradient of $C(\cdot)$ with respect to $\mathbf{w}$.

In selecting the step length $\eta(n)$, an *optimum* value can be determined by searching along the chosen direction $\mathbf{d}(n)$ for the minimum in $C(\cdot)$. In other words, the optimum

step length $\eta^*(n)$ occurs when

$$\left.\frac{\partial C(\mathbf{w}(n+1))}{\partial \eta(n)}\right|_{\eta(n)=\eta^*(n)} = 0.$$

which, from Equation (3.15), results in

$$(\mathbf{d}(n))^T \mathbf{g}(\mathbf{w}(n+1)) = 0. \tag{3.19}$$

which shows that an *optimal* step length results in a gradient of $C(\cdot)$ at $\mathbf{w}(n+1)$ that is orthogonal to the direction used to get to $\mathbf{w}(n+1)$. It can be shown [47, 129] that if $C(\cdot)$ is at least twice differentiable, that differentiating a second order Taylor series expansion of $C(\cdot)$ about the point $\mathbf{w}(n+1)$ with respect to $\eta(n)$ and setting that expression to zero, results in

$$\eta^*(n) = \frac{\mathbf{g}^T(\mathbf{w}(n))\mathbf{A}(n)\mathbf{g}(\mathbf{w}(n))}{\mathbf{g}^T(\mathbf{w}(n))\mathbf{A}(n)\mathbf{H}(\mathbf{w}(n))\mathbf{g}(\mathbf{w}(n))}. \tag{3.20}$$

which for $\mathbf{A}(n) = \mathbf{I}$, as used in the steepest descent method, reduces to

$$= \frac{\mathbf{g}^T(\mathbf{w}(n))\mathbf{g}(\mathbf{w}(n))}{\mathbf{g}^T(\mathbf{w}(n))\mathbf{H}(\mathbf{w}(n))\mathbf{g}(\mathbf{w}(n))}. \tag{3.21}$$

In practice, determining an *optimum* step length using Equation (3.21), is computionally expensive and it has been determined that the information from the Hessian is better used to help determine an improved descent direction [129]. In this light, nonoptimal step lengths are typically chosen where a common and simple choice is to select a fixed step size, that is to set $\eta(n) = \eta$. Other nonoptimal strategies in selecting a step length $\eta(n)$ are also possible. In the context of supervised learning, the step length is referred to as the learning rate parameter or as the step size in learning. In this context, other nonoptimal strategies in setting $\eta(n)$ include performing a line search for η^*, applying learning rate schedules, and using ad hoc adaptive learning rates [47].

The simplicity in selecting a fixed valued for η is traded-off for the possibility of either a very slow descent or an unstable descent. Avoidance of these possibilities is typically approached by trial and error where experience has shown that values in the range of $0.01 \leq \eta \leq 0.1$ are appropriate.

Gradient Evaluation, batch, and on-line learning

The need to determine the gradient $\nabla C(\cdot)$ is present in most descent methods and it can pose a significant challenge when the parameters given by $\mathbf{w}$ are within a $\hat{h}$ having a complex nonlinear structrure such as that of a Equation (3.7) describing a multilayered FFNN. To approach this challenge, the method of backpropagation, as reintroduced by Rumelhart *et al.* [74] to the neural network community and as described by others [9, 47], is an effective method to calculate the gradient vector needed in most descent methods.

In a single hidden layer FFNN, such as that shown in Figures (3.3) and (3.4), the direct application of the chain rule of differentiation will also yield the gradient vector without resorting to the back propagation of errors as is used in the method of backpropagation. For this reason, this thesis does not use the method of backpropagation to evaluate the gradients needed for the $D^{(1)}$ learning algorithm, instead the gradients are determined by direct application of the chain rule of differentiation.

Now consider the error surface $C(\mathbf{w}, D_k) = \epsilon_{D_k}$, based on the l_2-norm, as described in section 3.3.1. The gradient in this case is given by

$$\nabla C(\mathbf{w}(n), D_k) = \frac{\partial C(\mathbf{w}(n), D_k)}{\partial \mathbf{w}(n)} = \frac{\partial e_{D_k}}{\partial \mathbf{w}(n)} = \frac{\partial d(h(\mathbf{x}_i, \hat{h}(\mathbf{x}_i))^2}{\partial \mathbf{w}(n)}. \qquad (\mathbf{x}_i, h(\mathbf{x}_i)) \in D_k.$$

$$= \frac{\partial}{\partial \mathbf{w}(n)} \sum_{i=1}^{N_k} \Big(h(\mathbf{x}_i) - \hat{h}(\mathbf{x}_i)\Big)^2. \qquad (3.22)$$

It is apparent from Equation (3.22) that the entire set of data in D_k is used to determine the true gradient of $C(\mathbf{w}, D_k)$ at $\mathbf{w}(n)$. In terms of NN learning, using all the data to determine the weight adjustment is referred to as batch learning. Also, it should be noted that using all the data in D_k is referred to as a learning epoch.

It is also possible to use an *instantaneous* value or estimate of the gradient based on a single data sample from D_k. More precisely, this method forms N_k data sets $D_{k_i} = \{(\mathbf{x}_i, h(\mathbf{x}_i)\}$ for $i = 1, 2, \ldots, N_k$, and where $\overline{D}_{k_i} = 1$, $D_{k_i} \subset D_k$. This method of gradient evaluation is also referred to as on-line or sequential learning. In this method, a gradient direction and step are determined N_k times for a single learning

epoch, that is a learning epoch still consists of using all the data samples in D_k, but instead N_k descent steps per epoch are taken. In applying this method, it is also recommended that the order, in which the samples are used to estimate the gradient, be randomised for each epoch ([9], pg. 171).

The use of single data samples to estimate the gradient, in conjunction with a possible random ordering for each epoch, results in descent path over the error surface that resembles a random walk. This has prompted the use of an alternative term for the on-line learning algorithm: a stochastic gradient algorithm.

The stochastic nature of the descent in on-line learning makes it difficult to establish convergence results, that is will the algorithm find and settle into a minimum. Recently, Wang *et al.* [138] have suggested that convergence is not guaranteed for on-line learning using the backpropagation algorithm. In spite of this potential non-convergence problem, overviews regarding on-line training have indicated that it can and has been used successfully and provides computational advantages over batch learning [9, 47].

3.3.2 Limitation in FFNN learning

From the previous sections, it is clear that given the complexity of the error surface, limited data, and nonideal search techniques, that finding a minimun on the error surface which corresponds to an approximation with an acceptable generalisation error is not a trival problem. This section presents an alternative view of this nontrival problem that provides additional insight into the limits of FFNN learning accuracy given finite data sets and limited numbers of hidden neurons. It is obvious that this insight helps in understanding the limitations of using a FFNN as as instrument calibration model.

To present this alternative view, the processes used in supervised FFNN learning, instrument calibration, and function approximation are all considered as attempts to reconstruct a continuous mapping [81] or hypersurface, $\hat{h} : \mathcal{X} \rightarrow \mathcal{Y}$, using only a

finite set of samples $D_k = \{(\mathbf{x}_j, h(\mathbf{x}_j)) : j = 1, 2, \ldots, N_k, \mathbf{x}_j \in \mathcal{X}, h(\mathbf{x}_j) \in \mathcal{Y}\}$, that exemplify the true underlying hypersurface h.

To avoid complicating the present discussion, the data set D_k is assumed to have a probability density function for $\mathbf{x}$ that is uniform over all $\mathcal{X}$. In addition, it is assumed $\forall \mathbf{x} \in \mathcal{X}$, that there is only one corresponding value of $h(\mathbf{x})$ that is determined uniquely, that is, there is no associated error or noise. Finally, since the terms mapping, function, and hypersurface are, in the context of this discussion, equivalent they will be used interchangably.

As a problem in hypersurface reconstruction, the ideal goal is to achieve perfect accuracy in reconstructing or approximating h. Alternatively, this goal can be expressed in terms of the error e between h and $\hat{h}$. As introduced in section 1.2.2, e can be defined as a distance measure $d(\cdot, \cdot)$. The ideal approximation would then correspond to distance or error of zero, that is

$$e = d(h(\mathbf{x}), \hat{h}(\mathbf{x})) = 0. \qquad \text{for } \mathbf{x} \in \mathcal{X}.$$

The distance measure is usually taken to be one of the L_p norms $\|\cdot\|_p$, typically the least square or Euclidean norm [6], i.e. $p = 2$, so that the error can then be expressed as

$$\begin{aligned} e &= \|h(\mathbf{x}) - \hat{h}(\mathbf{x})\|_2 = 0. \qquad \text{for } \mathbf{x} \in \mathcal{X}. \\ e &= \left(\int_{\mathcal{X}} (h(\mathbf{x}) - \hat{h}(\mathbf{x})^2 d\mathbf{x} \right)^{\frac{1}{2}} = 0. \end{aligned} \tag{3.23}$$

For convenience in presentation, let $e_g = e^2$, where e_g is the generalisation error. In the context of instrument calibration, e_g is the true squared error in prediction. In instrument calibration, it is the error e_g over $\mathcal{X}$ that is important in that provides an indication of instrument's measurement performance.

Regardless of the context e_g is placed in or its importance, it is apparent that to approach this ideal goal requires addressing a number of concerns. First, it is clear that to achieve a zero error, or to even measure e_g, requires knowledge of h and $\hat{h}$ over all $\mathcal{X}$, which is not available, i.e. only the data in D_k is available. In addition,

having $e_g = 0$ presumes that $\hat{h}$ can in fact approximate h identically over all $\mathcal{X}$. In instances when $\hat{h}$ can not approxmate h, then $e_g \neq 0$, regardless of whether h and $\hat{h}$ is known or not known over all $\mathcal{X}$. These instances can occur when $\hat{h}$ is the wrong model or it does not have sufficient power or complexity to represent h [47, 120].

The nature of these concerns in measuring e_g and approaching $e_g = 0$ can be discussed more precisely by introducing additional abstractions and notation. Let $\mathcal{H}$ be a set representing the family of functions that h belongs to, i.e. $h \in \mathcal{H}$. Let $\mathcal{H}_p$ be the set representing the family of functions having a number of parameters proportional to $p > 0$ and where $\hat{h} \in \mathcal{H}_p$. As an example of a family of functions, $\mathcal{H}_p$ may represent univariate polynomials of degree $(p-1)$ or it may represent a single hidden layer FFNN with p hidden neurons. Finally, it is assumed that $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_p \subset \cdots \subset \mathcal{H}$, in other words, it is assumed that $\mathcal{H}$ does include $\mathcal{H}_p$ as a subset.

Using this abstraction, it is apparent that given some $h \in \mathcal{H}$, the process of attempting to attain $e_g = 0$ requires first selecting a $\mathcal{H}_p$ and then determining a specific $\hat{h} \in \mathcal{H}_p$. Now, assume a $\mathcal{H}_p$ has been selected, it is also clear that to attain $e_g = 0$ requires that $\hat{h}, h \in \mathcal{H}_p$. If $h \notin \mathcal{H}_p$, then $e_g \neq 0$ and the only recourse is to either select another $\mathcal{H}_q$, $q > p$, with the intent of having $\hat{h}, h \in \mathcal{H}_q$, or to accept $\mathcal{H}_p$ and determine a $\hat{h} \in \mathcal{H}_p$ that is acceptably *close* to h.

It is now seen that to select $\mathcal{H}_p$ and to determine a specific $\hat{h} \in \mathcal{H}_p$ requires measuring the *closeness* of h to $\hat{h}$. This *closeness* is essentially what is measured by Equation (3.23), but as was noted previously, it is not possible to directly measure e_g given **only** a finite set of data D_k. Instead, e_g needs to be estimated using an estimator, which in considering Equation (3.23), can be given by [47]

$$\hat{e}_g = \frac{1}{N_j} \sum_{j=1}^{N_j} \Big(h(\mathbf{x}_j) - \hat{h}(\mathbf{x}_j) \Big)^2, \qquad \text{for } (\mathbf{x}_j, h(\mathbf{x}_j)) \in T,\ N_j = \bar{\bar{T}}. \tag{3.24}$$

where $T \subset D_k$, is the test or validation set. In the context of learning, $\hat{e}_g$ is referred to as the empirical generalization error. In the context of instrument calibration, $\sqrt{\hat{e}_g}$ is known as the RMSEP. It should be noted that in spite of defining the true calibration

model h as having no error, the error measure $\hat{\epsilon}_g$ is essentially a random variable. This is true in the sense that T may be randomly selected and that $\hat{h}$ depends on both the data D which may be randomly selected and the search process that may start at some random point on the error surface.

The use of a finite data set must not only be used to estimate ϵ_g but it is also needed to select $\hat{h}$. In selecting $\hat{h}$, the data set D is used, where $D \subset D_k$ and $D \cap T = \emptyset$. Again, the selection of $\hat{h} \in \mathcal{H}_p$ requires a means of measuring the *closeness* of $\hat{h}$ to h using only the data in D. This can be achieved by viewing the set of points given $h(\mathbf{x}_j)$ and $\hat{h}(\mathbf{x}_j)$, for $j = 1, 2, \ldots, N_k$, as two vectors $\mathbf{h} = [h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_{N_k})]^T$ and $\hat{\mathbf{h}} = [\hat{h}(\mathbf{x}_1), \hat{h}(\mathbf{x}_2), \ldots, \hat{h}(\mathbf{x}_{N_k})]^T$ that also represent two points in $\mathbb{R}^{N_k}$. The distance between these two points $d(\mathbf{h}, \hat{\mathbf{h}})$, then provides a measure of the *closeness* of $\hat{h}$ to h, not over all $\mathcal{X}$, but only over $(\mathbf{x}_j, h(\mathbf{x}_j)) \in D$. Again, any measure of distance can be used, but typically, the squared l_2-norm is commonly chosen [9, 47] and can be expressed as

$$C = \sum_{j=1}^{N_k} \left(h(\mathbf{x}_j) - \hat{h}(\mathbf{x}_j) \right)^2, \qquad \text{for } (\mathbf{x}_j, h(\mathbf{x}_j)) \in D,\ N_k = \bar{\bar{D}}. \tag{3.25}$$

which is also referred to as the sum squared error, SSE, measure. Now, since the *closest* distance is desired, the selection of $\hat{h}$ is such that

$$C^* = \min_{\hat{h} \in \mathcal{H}_p} (C). \tag{3.26}$$

is ideally obtained. In the context of this minimisation, Equation (3.25) is also more generally referred to as a quadratic cost, objective, or error function [47, 129, 139] in that the minimum cost or error is the objective.

In a strict sense, it is well known [9, 81, 84] that the selection of $\hat{h}$ using only the data in D is ill-posed in that it is not possible to determine a unique mapping $\hat{h}$ in regions between the data samples, or alternatively, there are an infinite number of mappings over $\mathcal{X}$ that also include the data samples D, i.e. where $C^* = 0$. One simple pragmatic solution is to provide additional prior information, such as assuming that $\hat{h} \in \mathcal{H}_p$, so as to restrict the number of selections for $\hat{h}$. If sufficient prior information

is used along with the data set, then the selection of $\hat{h}$, using Equation (3.25), can be reduced to a single evaluation.

For example, assume that $\hat{h} \in \mathcal{H}_2$, where $\mathcal{H}_2$ is the family of mulitvariate linear hyperplanes, $\hat{h}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$. Now the selection of $\hat{h}$, given D having N_k data samples can, by rewriting C in matrix form, be determined by solving $\partial(\mathbf{h} - \mathbf{X}\mathbf{w})^T(\mathbf{h} - \mathbf{X}\mathbf{w})/\partial\mathbf{w} = \mathbf{0}$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_k}]^T$, $\mathbf{w} = [w_0, w_1]^T$, $\mathbf{x} = [x_0 = 1, x_1]^T$, and $\mathbf{0}$ is a $(N_k \times 1)$ matrix of zero elements. In other words, the selection of $\hat{h}$ using D and Equation (3.25) then reduces to estimating the parameters $\mathbf{w}$ of $\hat{h}$ so as to minimise C. The direct solution for this minimisation is the well known least squares estimate [4], $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{h}$. The resulting selection of $\hat{h}$, achieving C^*, then simply becomes $\hat{h}(\mathbf{x}) = \mathbf{x}^T\hat{\mathbf{w}}$. If the inverse does not exist, the generalised inverse $\mathbf{X}^+$, where $\mathbf{X}\mathbf{X}^+\mathbf{X} = \mathbf{X}$, may be used [4].

In cases where $\mathcal{H}_p$ is a family of functions that are not linear in parameters, the selection of $\hat{h}$ by a direct minimal solution of Equation (3.25) becomes a much more difficult task. In these cases, one common approach, as discussed previously, is to *search* for a set of parameters $\mathbf{w}$ of $\hat{h}$ that achieves a minimisation. Again, the success of the search, i.e finding C^*, is not always assured by virtue that it depends on the complexity of the search space, i.e error surface, and on the technique used to perform the search. More importantly, and as was shown in section 3.3.1, the minimisation of $C = d(\mathbf{h}, \hat{\mathbf{h}})^2$, which is dependent on D, is generally not the same as the desired minimisation of $e_g = d(h(\mathbf{x}), \hat{h}(\mathbf{x}))^2$. In other words, the optimal minimum of C, may not correspond to the optimal minimum of e_g by virtue of using a finite D to represent the h and $\hat{h}$. Second, the optimal minimum of C, referred to as a global minimum, may not be found. Instead, it is likely that due to the complexity of the search space and the limitation of the search technique that the search will find a less than optimal miminisation, referred to as a local minimum.

To emphasis this important idea - that the optimal minimum of C, may not correspond to the optimal minimum of e_g - consider Figure (3.7) which shows the

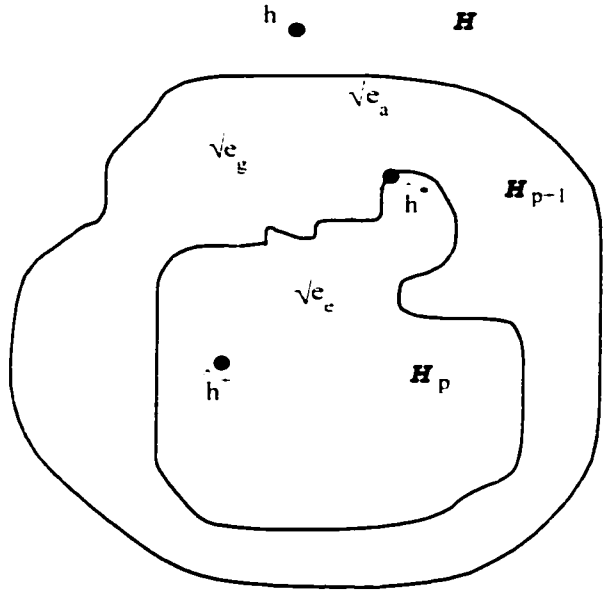


Figure 3.7. The relationship between generalisation error e_g, approximation error e_a, and estimation error e_e. The inner most closed area represents the class of functions in the set $\mathcal{H}_p$, the outer most area represents $\mathcal{H}$. The true calibration model is $h \in \mathcal{H}$ and the *best* approximation available in $\mathcal{H}_p$ is $\hat{h}^*$. A search for $\hat{h}^*$, using C and only D, may find $\hat{h}^+$ instead.

true calibration model h being represented as a point in a plane defined by $\mathcal{H}$. Also, shown is $\mathcal{H}$ encompassing a closed area defined by $\mathcal{H}_{p-1}$, which similarily, encompasses another closed area defined by $\mathcal{H}_p$. This succesive encompassing of areas can be expressed using $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_p \subset \cdots \subset \mathcal{H}$. Now assume that class of functions represented by $\mathcal{H}_p$ has been selected to provide an approximation $\hat{h}$. Using the minimum of $e_g = d(h(\mathbf{x}), \hat{h}(\mathbf{x}))^2$, given by Equation (3.23), to define the *best* approximation, it is clear that the point identified as $\hat{h}^*$, located in $\mathcal{H}_p$, is the best approximation but $e_g \neq 0$ since $h \notin \mathcal{H}_p$. Now further assume that search for $\hat{h}^*$, which is the *closest* approximation to h as measured by $C = d(\mathbf{h}, \hat{\mathbf{h}})^2$, results in a less than ideal ending in that a local minimum C is found at the point identified as $\hat{h}^+$.

As can be seen in Figure (3.7), there are three distances between the points h, $\hat{h}^*$, and $\hat{h}^-$ that are related by $\mathbf{e}_g = \mathbf{e}_a + \mathbf{e}_e$, where $\mathbf{e}$ is an error vector providing both the distance and direction between points in $\mathcal{H}$. The distance given by $e_g = d(h(\mathbf{x}), \hat{h}(\mathbf{x}))^2$ represents the true generalisation, introduced in Equation (3.23), and consists of two components. The first component has a squared magnitude given by $e_a = d(h(\mathbf{x}), \hat{h}^*(\mathbf{x}))^2$ and is referred to as the approximation error which represents the inability of the selected class of functions to approximate h. This error is also known as the bias in $\hat{h}$ and is independent of the data D [4]. The second component has a square magnitude given by $e_e = d(\hat{h}^*(\mathbf{x}), \hat{h}^-(\mathbf{x}))^2$ and is called the estimation error which represents the error due to using both a finite data set D and a search technique that may not find the global minimum of C, that is $\hat{h}^*$ may not be found. Note that random error associated with h, that is, data having zero mean normally distributed noise, $D = \{(\mathbf{x}_j, h(\mathbf{x}_j) + N(0, \sigma)) : j = 1, 2, \ldots, N_j\}$, will also contribute to the estimation error e_e ([4], pg. 261).

As a simple example of these ideas, consider the case of using $\mathcal{H}_1$ representing the univariate linear class of functions, i.e. $\hat{h}(x) = w_1 x + w_0 \in \mathcal{H}_1$, to approximate $h(x) = x^2$. Now it is obvious that $h \notin \mathcal{H}_1$, so that there will always be an approximation error, i.e. $e_a \neq 0$. It is also easily shown the least squares estimate for $\mathbf{w} = [w_1, w_0]^T$ over $\mathcal{X} = [0, b]$, is, regardless of any $D = \{(x_j, h(x_j)) : j = 1, 2, \ldots, N_k\}$, given by $\mathbf{w}^* = [w_1 = b, w_0 = -\frac{1}{6}b^2]^T$. This results in $\hat{h}^*(x) = \mathbf{x}^T\mathbf{w}^* = bx - \frac{1}{6}b^2$ and represents the minimum in the L_2-norm approximation of h. Now assuming that only the data in D is available, the selection of $\hat{h}$ using C will result in the least squares estimate of $\mathbf{w}$ over D given by $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{h}$ which minimises C, i.e. C^* is found. It is apparent that $\hat{\mathbf{w}}$ is dependent on D and if the calibration data is randomly chosen by the instrument user, then it is likely that $\hat{\mathbf{w}} \neq \mathbf{w}^*$ so that selection of $\hat{h}$ results in $\hat{h}^- \neq \hat{h}^*$. This difference between $\hat{h}^-$ and $\hat{h}^*$ is the estimation error e_e and is due entirely to not having sufficient or optimal data provided in D. Also in this case, the search process for $\hat{h}$, which is a direct solution to the minimisation of C, does not contribute to the estimation error since $\hat{\mathbf{w}}$ is always the global minimum of C.

In more complex search spaces using search techniques with limited capabilities, it is also likely that $\hat{\mathbf{w}}$ will correspond to a local minimum of C, thereby contributing additional error to ϵ_e.

3.4 FFNN Calibration Transfer Methods

This section presents the theory and methodology used to develop the calibration transfer method referred to as PICX, iDACC, and $D^{(1)}$-iDACX for a single hidden layer FFNN. This presentation begins by briefly restating the calibration transfer problem and then defining the notation needed to discuss the development of the theory and methods of calibration transfer. Next, the simpler forms of calibration transfer for the given FFNN are discussed, that is, the methods referred to as PICX and iDACX. Finally, the $D^{(1)}$-iDACX method is developed.

3.4.1 Preliminaries

The instrument and the calibration transfer problem

To describe the calibration transfer methods of this section, it will be instructive to briefly review the instrument calibration problem. For convenience, the instrument system, as shown in chapter 1, is reproduced in Figure (3.8).

The true association between $\mathcal{Z}$ and $\mathcal{Y}$ is given by the map $f = h \circ \mathbf{g} : \mathbf{z} \rightarrow y$. It is assumed that this mapping is fixed over all time and instrument realisations. The true *complete* calibration data set is then defined by $\mathcal{D}_0 = \mathcal{Z} \times \mathcal{Y} = \{(\mathbf{z}, f(\mathbf{z}) = y) : \mathbf{z} \in \mathcal{Z}, y \in \mathcal{Y}\}$.

The ideal instrument system provides a kth realisation of this true association between $\mathbf{z}$ and y given by $h_k \circ \mathbf{g}_k : \mathbf{z} \rightarrow y$. Pragmatically, only an approximation of this true association is possible, and for this thesis this approximation is given by the map $\hat{h}_k \circ \mathbf{g}_k$. Therefore, the standard calibration problem is to determine $\hat{h}_k$ from the finite calibration data set $D_k = \{(\mathbf{g}_k(\mathbf{z}_i) = \mathbf{x}_i, h_k(\mathbf{x}_i)) : i = 1, 2, \ldots, N_k\}$.

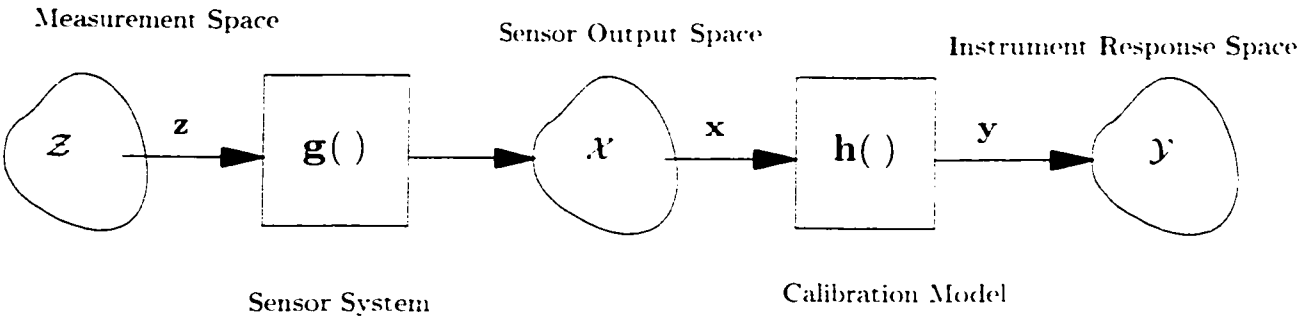


Figure 3.8. A block diagram of an ideal instrument system illustrating the process of taking a measurement $\mathbf{z}$ from $\mathcal{Z}$ and providing a corresponding output reading $\mathbf{y}$.

Given a change in the sensor system from $\mathbf{g}_k$ to $\mathbf{g}_l$, the ideal instrument will need to provide the lth realisation of the true association given by the map $h_l \circ \mathbf{g}_l$. By definition, $\mathbf{g}_l$, used in conjunction with h_k, results in the approximation of the true association, f, having unacceptable errors, that is, the map $h_k \circ \mathbf{g}_l$ has unacceptable errors. A recalibration is needed to obtain the approximation $\hat{h}_l$ using the new calibration data set $D_l = \{(\mathbf{g}_l(\mathbf{z}_i) = \mathbf{x}_i, h_l(\mathbf{x}_i)) : i = 1, 2, \ldots, N_l\}$.

A calibration transfer, in the context of this thesis, consists of an attempt to exploit an accurate, and possibly expensively obtained, calibration model $\hat{h}_k$ so as to obtain another calibration model $\hat{h}_l$ having acceptable accuracy using less data, that is, to have $\bar{\bar{D}}_l < \bar{\bar{D}}_k$.

Notation

To discuss calibration transfer and to compare its accuracy with other methods, various realisation of data sets, calibration models, sensor system, and the corresponding error measures will, at times, be used together. To avoid confusion, it will be necessary to more precisely specify some of the notation introduced early.

Calibration and test sets. The complete calibration data set was defined as $D_0 = \{(\mathbf{x}, h(\mathbf{x})) : \mathbf{x} \in \mathcal{X}, h(\mathbf{x}) \in \mathcal{Y}\}$. A specific finite element subset used as the actual data set for the calibration of h was also denoted as D or as D_k, both of which are subsets of D_0. A finite calibration data set was also defined as $D_k = \{(\mathbf{x}_j, h(\mathbf{x}_j)) : j = 1, 2, \ldots, N_k, \mathbf{x}_j \in \mathcal{X}, h(\mathbf{x}_j) \in \mathcal{Y}\}$. It is now necessary to introduce additional notation to distinguish between the various data and data sets.

The complete calibration data set D_0 is dependent on the instrument system. To see this dependency and its relevance to the calibration transfer, consider that the goal in instrument design is to associate an input measurement $\mathbf{z}_i$ with an instrument reading y_i. The true association is given by the true underlying mapping $f : \mathbf{z} \rightarrow y$. The true complete calibration data set is then defined as $\mathcal{D}_0 = \{(\mathbf{z}, f(\mathbf{z})) = y : \mathbf{z} \in \mathcal{Z}, f(\mathbf{z}) = y \in \mathcal{Y}\}$. More practically, an ith true finite calibration set is drawn from the true complete calibration. This ith true finite calibration set $\mathcal{D}_i$ is defined as $\mathcal{D}_i = \{(\mathbf{z}_j(i), f(\mathbf{z}_j(i))) = y_j(i) : j = 1, 2, \ldots, N_i, \mathbf{z}_j \in \mathcal{Z}, f(\mathbf{z}_j) = y_j \in \mathcal{Y}\}$, where $i = 1, 2, \ldots$, and where it may be necessary to use $\mathbf{z}_j(i)$ and $y_j(i)$ to refer to data belonging to the ith true finite calibration set. It should be noted that f is assumed to be fixed, therefore, the true complete calibration data set $\mathcal{D}_0$ does not change.

Now the instrument is designed to provide a kth realisation of the true underlying mapping $f = h_k \circ \mathbf{g}_k$. In this context, the complete calibration data set is given by $D_{k0} = \{(\mathbf{g}_k(\mathbf{z}) = \mathbf{x}(k), f(\mathbf{z}) = y) : \mathbf{x}(k) \in \mathcal{X}\}$, where the D_{k0} refers to the complete calibration data set generated using the kth realisation of the sensor system and where $\mathbf{x}(k)$ may be used to indicate that the data was generated with the kth sensor system. Also note that the sensor output space $\mathcal{X}$ is defined such that $\mathbf{g}_k(\mathbf{z}) \in \mathcal{X}, \forall k$. More typically, the calibration and test set will be determined by the ith true finite calibration set $\mathcal{D}_i$, so that in its complete form $D_{ki} = \{(\mathbf{g}_k(\mathbf{z}_j(i)) = \mathbf{x}_j(k, i), f(\mathbf{z}_j(i)) = y_j(i)) : j \in \{1, 2, \ldots, N_i\}, \mathbf{x}_j(k, i) \in \mathcal{X}\}$, where $\mathbf{x}_j(k, i)$ may be used to indicate data generated with kth realisation of the sensor system given the jth data point from the ith true calibration data set. Note that $i = 0$ indicates data drawn from $\mathcal{D}_0$.

Given the lth realisation of sensor system $\mathbf{g}_l$, the calibration data set is given by $D_{li} = \{(\mathbf{g}_l(\mathbf{z}_j(i)) = \mathbf{x}_j(l,i), f(\mathbf{z}_j(i)) = y_j(i)) : j \in \{1,2,\ldots,N_i\}, \mathbf{x}_j(l,i) \in \mathcal{X}\}$. As discussed previously, it is always the case that $\forall i$, $D_{li} \subset D_{l0}$ and $T_{li} \subset D_{l0}$, where again $D_{li} \cap T_{li} = \emptyset$.

Finally, it is also possible to draw a number of different calibration and test sets from a given true finite calibration data set $\mathcal{D}_i$. To identify these instances the notation $D_{ki}(m)$ is used to represent the mth draw of data from the ith true finite calibration data generated with the kth sensor system. The coordinates of the data drawn for $D_{ki}(m)$ are given by $\mathbf{x}_j(k,i)$, where $j \in I_m \subset \{1,2,\ldots,N_i\}$ and I_m represents the mth indexed set whose elements specify the values to be used for the jth index. The number of data points in D_{ki} is given by $N_k = \bar{\bar{D}}_{ki} = \bar{\bar{I}}_m$. For $i = 0$, $N_i = \infty$. If the indexed set is not specified then the $D_{ki}(m)$ will be explicitly defined.

Note that this complete and cumbersome notation for the data or data sets will only be invoked when there is a possibility of confusion as to which data or data set is being used. In cases, where it is clear from the context or if it is not necessary to distinguish these cases, the notation introduced earlier will be used. For example, if it is only necessary to identify that the calibration data set is based on the lth sensor system, regardless of which ith true calibration set it is drawn from, then the notation D_l is used, where the number of data samples is given by $N_l = \bar{\bar{D}}_l$.

Error measures. The error measures were used to evaluate the difference between the k realisation of the desired calibration model h_k and its approximation $\hat{h}_k$. The true generalisation error was then given by the $e_g = d(h_k(\mathbf{x}), \hat{h}_k(\mathbf{x}))^2$, where $\mathbf{x} \in \mathcal{X}$. Implicit in this definition is that $\mathbf{x} = \mathbf{g}_k(\mathbf{z})$. In the context of calibration transfer it is also possible to have $\mathbf{x} = \mathbf{g}_l(\mathbf{z})$ and to use this as the argument for either h_l, $\hat{h}_l$, or $\hat{h}_k$. Using $\mathbf{x}(l)$ to indicate that the data is generated by the lth realisation of the sensor system, the generalisation error may be denoted as $e_g(h_l, \hat{h}_l) = d(h_l(\mathbf{x}(l)), \hat{h}_l(\mathbf{x}(l)))^2$, if the meaning is not clear. If the meaning of the generalisation error is clear, the error will be simply denoted as e_g. In the case of using different realisations of instruments,

as when the error in using the kth calibration model with the lth sensor system is measured, the generalisation error will be denoted as $e_g(h_l, \hat{h}_k) = d(h_l(\mathbf{x}(l)), \hat{h}_k(\mathbf{x}(l)))^2$.

The notation becomes cumbersome if finite data sets are used to estimate errors and if it becomes important to distinguish between data sets used to estimate the error. In the most cumbersome case, an estimated error measure $\hat{e}$ will, for example, be denoted as $\hat{e}(h_l, \hat{h}_k, D_{ll}(m))$. This error represents an estimated error measure between h_l and $\hat{h}_k$ over the data set $D_{ll}(m)$ having N_j data samples, and where $\mathbf{x}_j$ is generated by the lth sensor system when given data drawn from the ith true calibration data set $\mathcal{D}_i$ according to the mth indexed set I_m, that is $j \in I_m$ and $N_j = \bar{\bar{I}}_m$.

3.4.2 Calibration transfer using the PICX methods

The objective in performing a calibration transfer is to exploit the calibration model $\hat{h}_k$ so as to obtain another calibration model $\hat{h}_l$ with acceptable accuracy using less data than that used to obtain $\hat{h}_k$. The simplest approach, which moves towards meeting this goal, is to attempt a standard calibration, that is, obtain $\hat{h}_l$ with the data set D_l having $\bar{\bar{D}}_l < \bar{\bar{D}}_k$.

Given that a FFNN is implementing the calibration model, a standard recalibration requires a *relearning* to obtain $\hat{h}_l$ using the data D_l. In using a data set D_l having $\bar{\bar{D}}_l < \bar{\bar{D}}_k$, it is likely that the new calibration model $\hat{h}_l$ will exhibit an increase in the calibration error, that is $\hat{e}(h_l, \hat{h}_l, D_l) > \hat{e}(h_k, \hat{h}_k, D_k)$.

Though, this increase in calibration error appears intuitively obvious, the supervised learning algorithm displays a characteristic that warrants considering it a form of calibration transfer. The particular FFNN characteristic that is of interest can be illustrated using a simple example of a calibration transfer problem.

To define the notation for this example, let $x \in \mathcal{X} \subset \mathbb{R}$ and assume that a FFNN has already determined a $\hat{h}_k(x)$ having an acceptable level of calibration error $\hat{e}(h_k, \hat{h}_k, D_k) \leq e_{\max}$, where $e_{\max}$ represents an arbitrary level of acceptable calibration

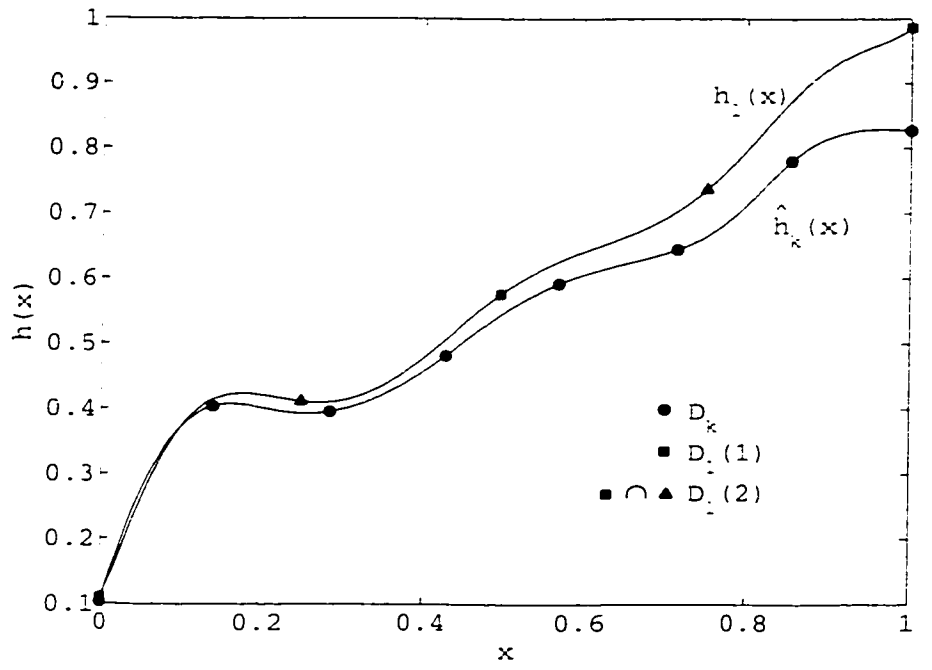
error. The calibration model $\hat{h}_k$ was obtained using the data set $D_k \subset D_0$ having $N_k = \bar{\bar{D}}_k$ calibration data samples. Now assume that a change in the sensor system from $\mathbf{g}_k$ to $\mathbf{g}_l$ has occurred which results in a calibration error $\hat{e}(h_l, \hat{h}_k, D_k) > e_{\max}$. To reduce the calibration error, a new calibration model $\hat{h}_l$, is needed. To obtain $\hat{h}_l$, the calibration data sets $D_l(1) \subset D_0$ and $D_l(2) \subset D_0$ are made available. Also to be consistent, the calibration error $\hat{e}$ will be measured using the empirical generalisation error, that is, $\hat{e}_g = \hat{e}^2$.

Figure (3.9a), illustrates a specific instance of this calibration transfer example, where $\mathcal{X} = [0, 1]$ and $\hat{h}_k(x)$, $h_l(x)$, D_k, $D_l(1)$, and $D_l(2)$ are as shown. Note that $\bar{\bar{D}}_k = 8$, $\bar{\bar{D}}_l(1) = 3$, and $\bar{\bar{D}}_l(2) = 5$, and that $\hat{h}_k$ and h_l are subjectively *similar*. Figure (3.9b) illustrates the same calibration transfer problem of Figure (3.9a), but viewed using only the data sets $D_l(1)$ and $D_l(2)$. It is apparent that without the visual aid of interpolating lines for h_l and $\hat{h}_k$, the approximation of h_l, given only the data $D_l(1)$ or $D_l(2)$, will be difficult, particularly if the level of calibration error is to remain under $e_{\max}$.

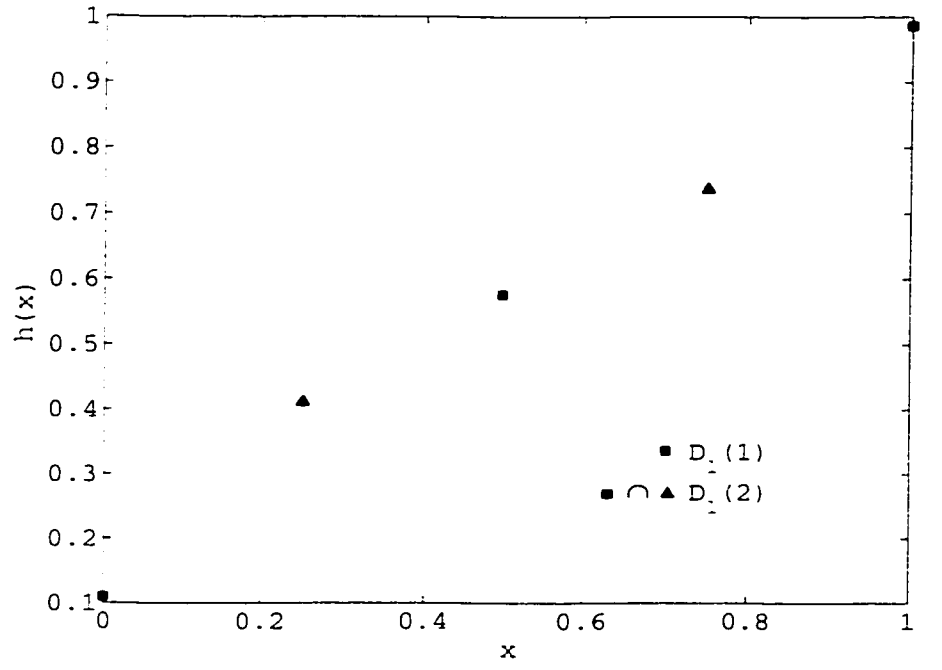
Now, for illustrative purposes and for latter comparisons, consider attempting to perform a standard calibration to obtain $\hat{h}_l$, not by retraining the FFNN, but by more conventional means, such as using a piecewise linear or a cubic spline interpolation [6].

Figure (3.10) and Figure (3.11) provides a sense of both the characteristics of the approximation $\hat{h}_l(x)$ and the magnitude of the squared error over $\mathcal{X}$ using a piecewise linear and a spline interpolation of the data $D_l(1)$ and $D_l(2)$. For reference, the estimated empirical generalisation errors, as defined by Equation (3.24) with $N_j = 200$, are also listed in the figures. In addition, the squared error behaviour of the previously trained FFNN, providing the approximation $\hat{h}_k$, is also shown in Figure (3.10b) and Figure (3.11b) as a solid line.

It is apparent from Figure (3.10) and Figure (3.11) that the use of either a piecewise linear or a cubic spline interpolation method, with $\bar{\bar{D}}_l < \bar{\bar{D}}_k$, causes a significant increase in the calibration error as compared to the error associated with $\hat{h}_k$.

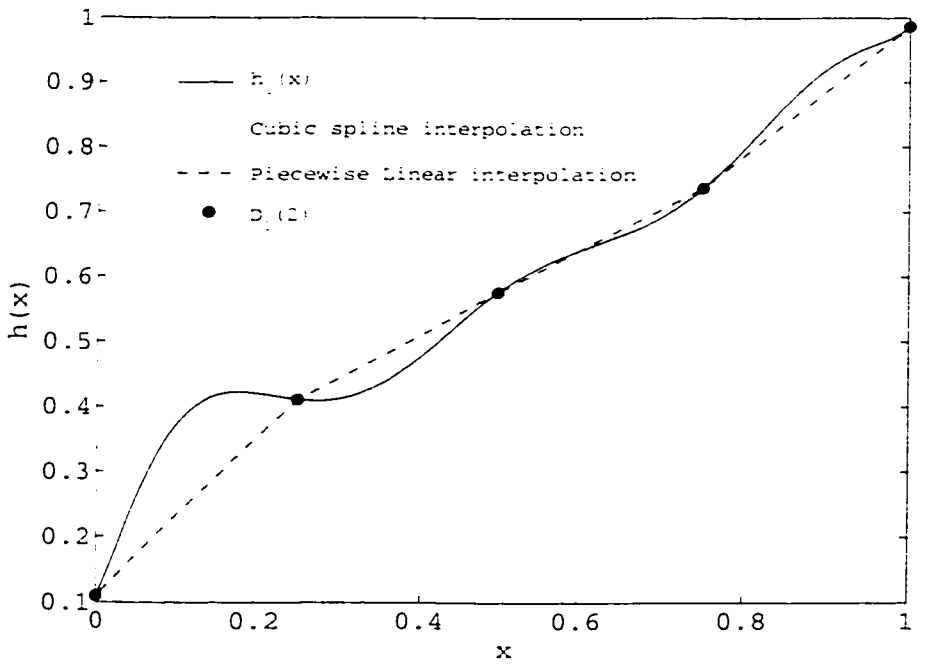


(a)

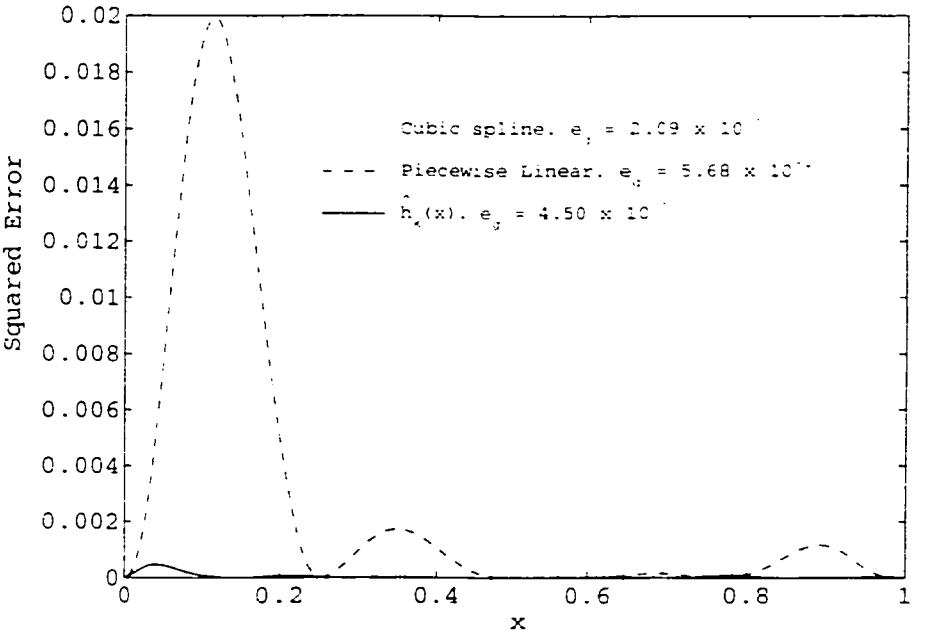


(b)

Figure 3.9. A calibration transfer problem. In (a), a FFNN has determined $\hat{h}_k(x)$ using D_k. A recalibration is required to approximate $h_l(x)$ using the calibration data $D_l(1)$ or $D_l(2)$. In (b), the same calibration transfer problem of (a) without the visual aid of interpolating lines.

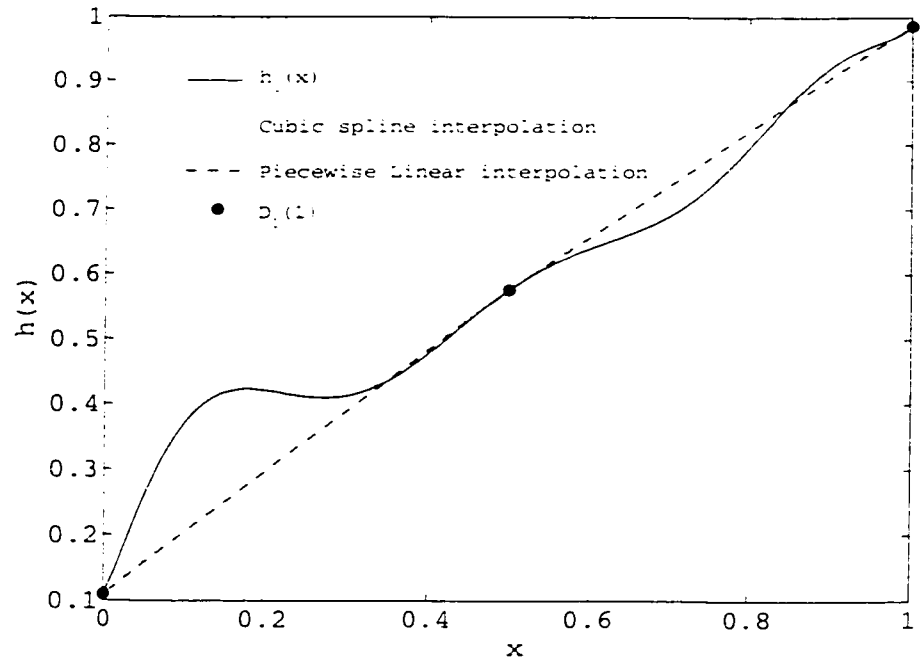


(a)

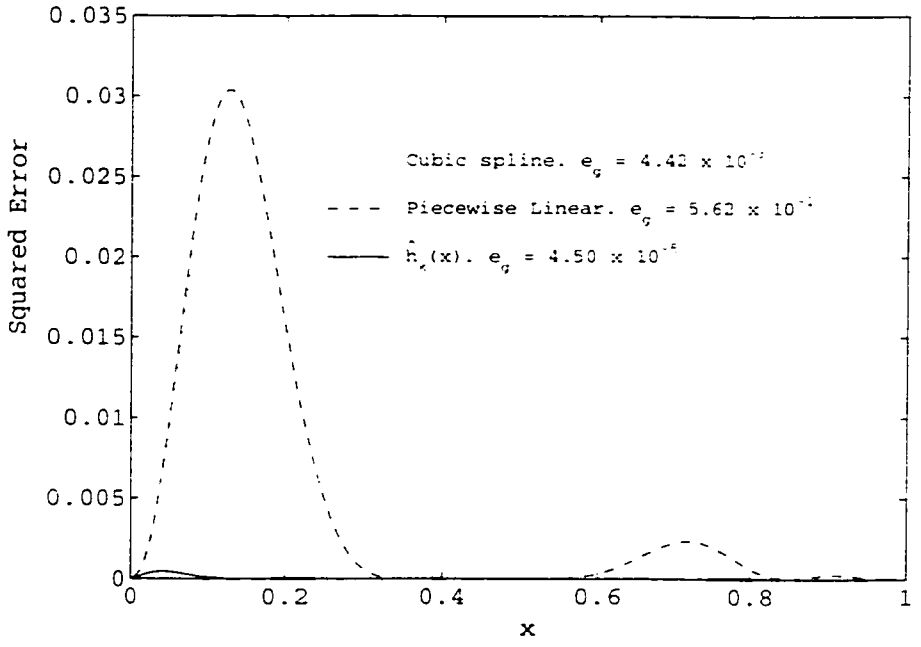


(b)

Figure 3.10. The results of a standard calibration using a piecewise linear and a cubic spline interpolation to obtain $\hat{h}_l$ with five calibration data points, given by $D_l(2)$, are shown in (a), where the true h_l is shown as a solid line. The squared errors between h_l and $\hat{h}_l$, are shown as a function of x in (b).



(a)



(b)

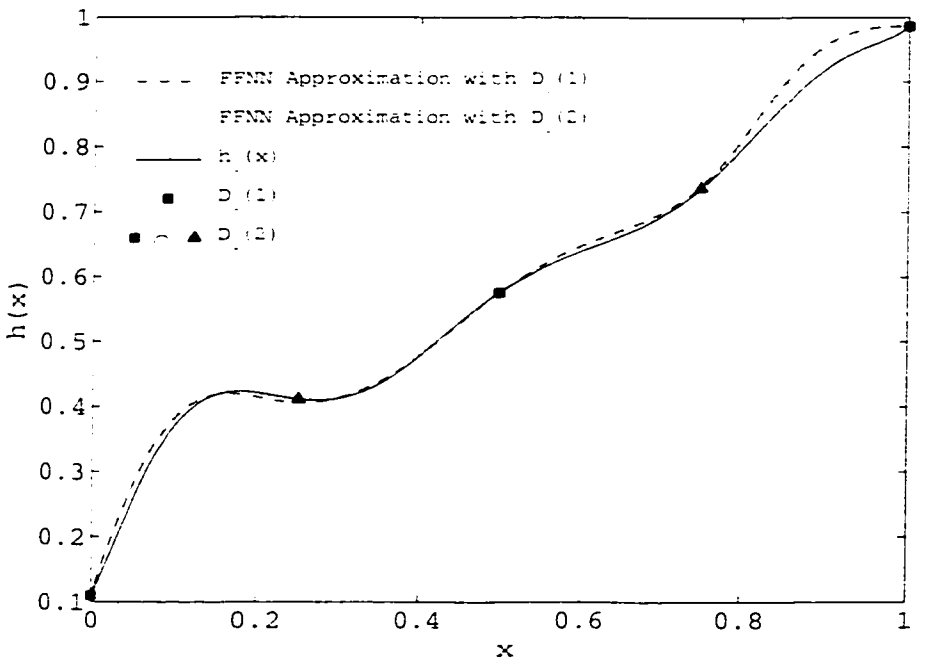
Figure 3.11. The results of a standard calibration using a piecewise linear and a cubic spline interpolation to obtain $\hat{h}_l$ with three calibration data points, given by $D_l(1)$, are shown in (a), where the true h_l is shown as a solid line. The squared errors between h_l and $\hat{h}_l$, are shown as a function of x in (b).

Now given that the FFNN has already learned $\hat{h}_k$, a standard calibration entails a new training, or learning, session so as to have the FFNN obtain the approximation $\hat{h}_l$ using the calibration data given by D_l. As was discussed in section 3.3.1, a FFNN requires its weights to be initialised. One of the most common techniques used to initialise the weights is to randomise them according to some heuristic, such as the Nguyen-Widrow method [140], or by simply using values selected from a zero mean normal random distribution, $N(0, \sigma < 1)$.

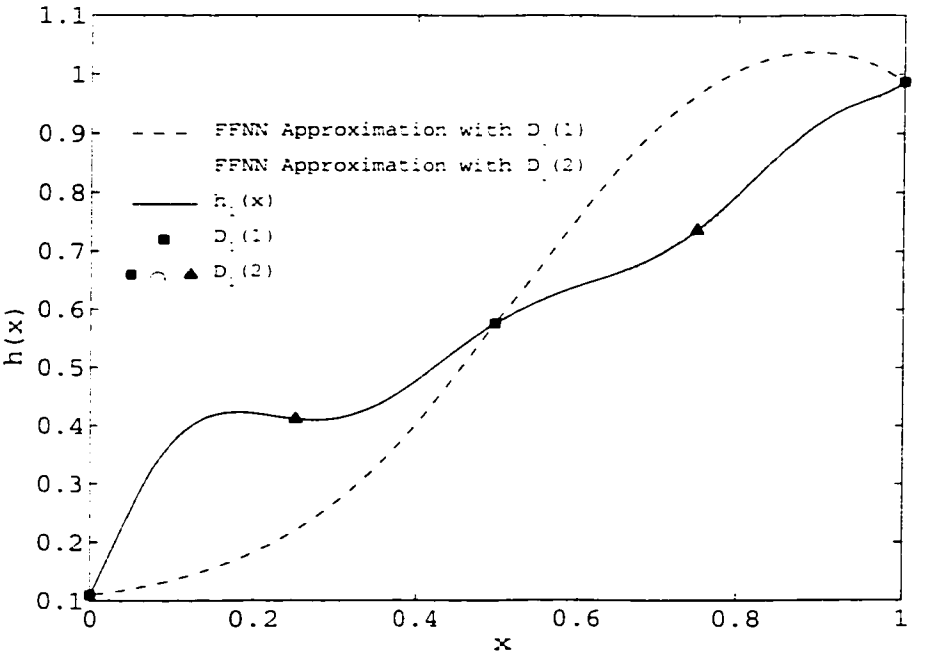
Another possible technique to use to initialise the weights, which is not considered in most FFNN applications, is to set the weights to the values that provided that previous approximation $\hat{h}_k$. The motivation in using the previous weights values is based on the potential similarity between $\hat{h}_k$ and $\hat{h}_l$ and the notion that a degree of information regarding $\hat{h}_k$ may be embedded in the weights. More precisely, given that a *similarity* between $\hat{h}_k$ and $\hat{h}_l$ may exist, there is a possibility that the error surface $C(\mathbf{w}, D_k)$ used to obtain $\hat{h}_k$ may also share a degree of *similarity* to the error surface $C(\mathbf{w}, D_l)$ that will be used to obtain $\hat{h}_l$. More importantly, it may be possible that a minimum on $C(\mathbf{w}, D_k)$, corresponding to $\hat{h}_k$, may be *close* to a minimum on $C(\mathbf{w}, D_l)$ which may correspond to a $\hat{h}_l$ that is *similar* to $\hat{h}_k$.

As an example of this idea, Figure (3.12) shows the approximations resulting from retraining the FFNN using the data defined by $D_l(1)$ and $D_l(2)$. In Figure (3.12a) the weights are initialised to the values which provided the previous approximation, that is, the initial weight values are set to provide the approximation $\hat{h}_k$. Figure (3.12b), shows the approximations resulting from retraining the FFNN using the same data with weights that are randomly initialised using the Nyugen-Widrow method [140].

Though the results shown in Figure (3.12) are but one specific instance in approximating h_l, the results indicate that initialising the weights to provide a previous approximation is likely to provide an approximation of h_l that has less error than an approximation resulting from using a random initialisation. In addition, these results suggest that a random initialisation of the weights can result in approximations hav-



(a)



(b)

Figure 3.12. The results of a FFNN approximating $h_l(x)$, as defined in Figure (3.9), using the data sets $D_l(1)$ and $D_l(2)$. In (a) the FFNN weights are initialisation to the values providing the approximation $\hat{h}_k$. In (b) the initial weight values are randomised.

ing calibration errors that are comparable to, or greater than, the calibration error resulting from a linear piecewise or a cubic spline interpolation. It therefore, seems advantageous to use the weights found from a previous approximation as the initial weight values for a FFNN when performing an instrument recalibration. In particular, it also seems intuitive to expect this advantage to improve as the *similarity* between $\hat{h}_k$ and $\hat{h}_l$ increases. These results are also suggesting that when the FFNN weights are initialised to the weight values providing the approximation $\hat{h}_k$, the FFNN appears to be performing the similarity map $\hat{h}_l = \Delta H(\hat{h}_k)$.

Therefore, this simple example has shown that, for instrument calibration and calibration transfer, it is important to identify how the weights are initialised. In this regard, the complete notation $\mathbf{w}(0, \hat{h}_k^-(\mathbf{x}; \mathbf{w}_j))$ may be used to represent the initial weights set to the values providing the jth approximation of $\hat{h}_k$. The initialisation using $\mathbf{w}(0, \hat{h}_k^-(\mathbf{x}; \mathbf{w}_j))$ will be referred to as a *prior initialisation*. A calibration transfer using this prior initialisation, in conjunction with a learning algorithm using an error measure based only on the data in D_l will be referred to as a prior initialisation calibration transfer, PICX, method. A random initialisation will be denoted as $\mathbf{w}(0, P_i)$, where P_i represents a set of random weight values generated according to some defined ith method, for example, $i \equiv$ NW, would indicate values generated according to the Nguyen-Widrow method [140].

The next approach to calibration transfer to be discussed is the iDACX method which is a very simple non-FFNN based method that attempts to obtain the approximation $\hat{h}_l$ using $\hat{h}_k$ in conjunction with the data set D_l. As will be shown, the calibration errors resulting from using the iDACX and PICX methods are comparable.

3.4.3 Calibration transfer using the iDACX methods

The iDACX method that is considered in this thesis is based on the indirect additive similarity map on the space of calibration models. The ideal mapping is

defined using

$$h_l(x) = \hat{h}_k(x) + s_l(x). \tag{3.27}$$

where it is assumed, without loss of generality, that $x \in \mathcal{X} \subset \mathbb{R}$. The ideal map requires perfect knowledge of s_l which represents the exact difference between the current approximation $\hat{h}_k$ and the desired approximation h_l. Pragmatically, this level of knowledge of s_l is not available, instead it is necessary to approximate this difference in $\hat{h}_k$ and h_l using $\hat{s}_l$, so that the less ideal mapping

$$\tilde{h}_l(x) = \hat{h}_k(x) + \hat{s}_l(x). \tag{3.28}$$

results. It should be noted that $\hat{h}_k(x)$, representing the FFNN approximation, is known over all $\mathcal{X}$.

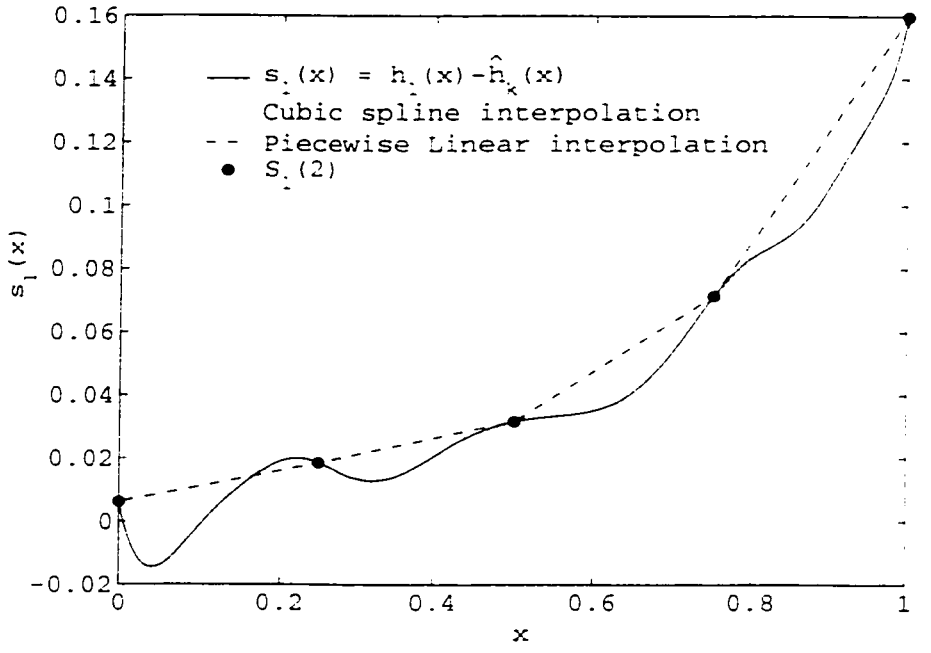
To obtain $\hat{s}_l$, the difference between $\hat{h}_k$ and h_l over the data in D_l is used. Specifically this data set is given by $S_l = \{(x_i, h_l(x_i) - \hat{h}_k(x_i) = s_l(x_i) : i = 1, 2, \ldots, N_l, (x_i, h_l(x_i)) \in D_l\}$. The approximation $\hat{s}_l$ can then be obtained using any number of parametric or nonparametric techniques.

It is not the intent of this thesis to determine an *optimum* technique to obtain $\hat{s}_l$, as it is not likely to be easily determined given an arbitrary $s_l(x)$. Instead, two techniques will be considered, a piecewise linear and a cubic spline interpolation [6].

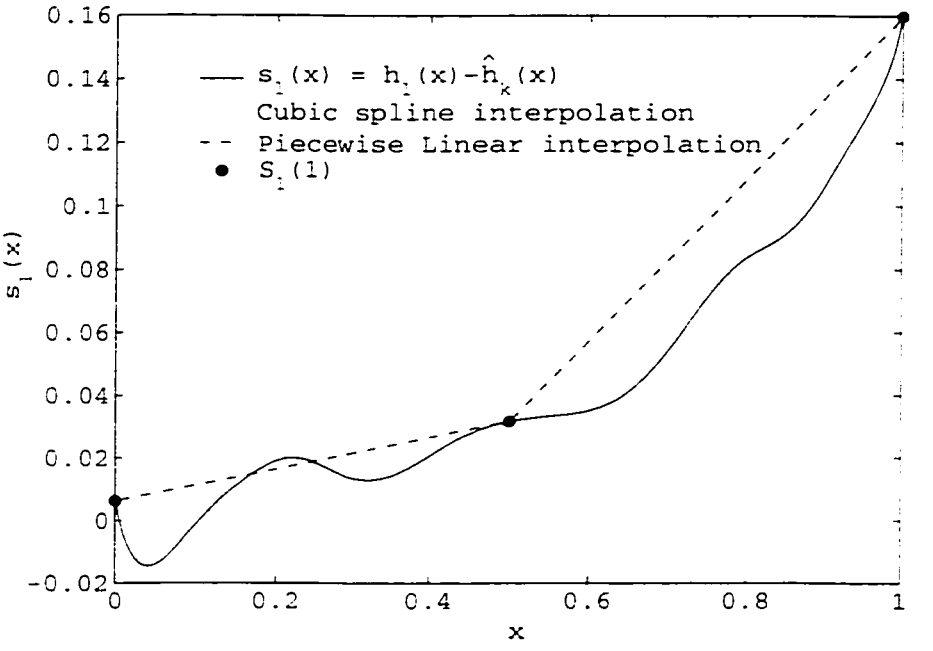
Continuing with the example from section 3.4.2, let $S_l(m) = \{(x_i, h_l(x_i) - \hat{h}_k(x_i) = s_l(x_i) : i = 1, 2, \ldots, N_l = \bar{\bar{D}}_l(m), (x_i, h_l(x_i)) \in D_l(m)\}$, for $m = 1, 2$. The approximations of $\hat{s}_l(x)$ obtained from both a piecewise linear and a cubic spline interpolation using $S_l(2)$ are shown in Figure (3.13a). Figure (3.13b) shows the result of using $S_l(1)$. The true difference, $s_l(x)$, is also shown in both figures.

It is not surprising that using more data provides a more accurate approximation of the difference $s_l(x)$. An important property of the piecewise polynomial approximators is that as the number of samples go to infinity, $\hat{s}_l \to s_l$ [6].

With $\hat{h}_k(x)$ and $\hat{s}_l(x)$ known over all $\mathcal{X}$, Equation (3.28) can be used to obtain $\tilde{h}_l(x)$. The resulting approximations of $\tilde{h}_l$ are shown in Figure (3.14a) and (3.15a).

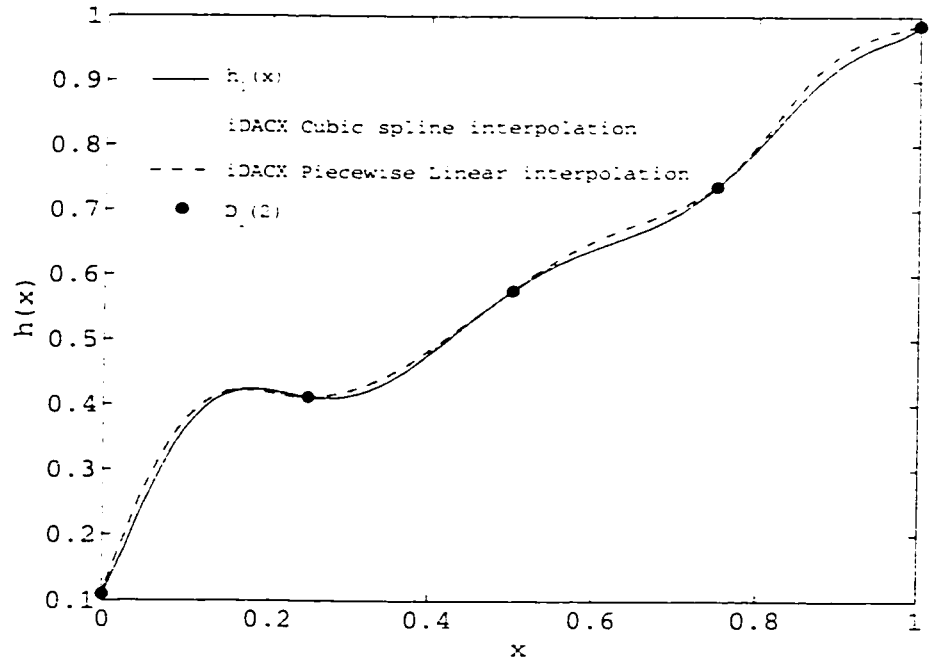


(a)

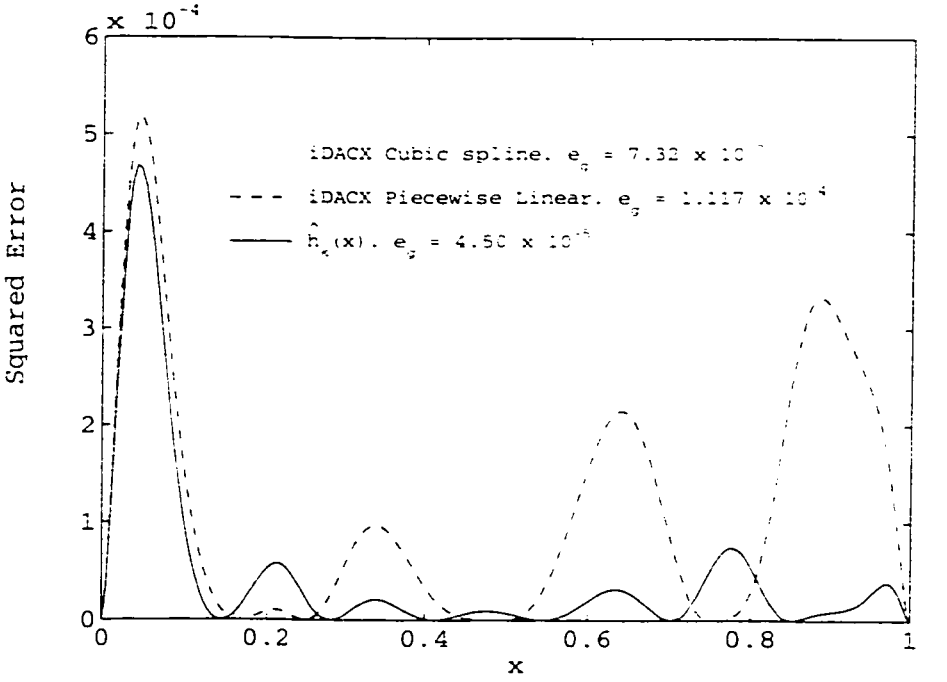


(b)

Figure 3.13. The approximated difference $\hat{s}_l$ between $\hat{h}_k$ and h_l obtained using a piecewise linear and a cubic spline interpolation. The data used to obtain the interpolation is $S_l(2)$ in (a) and $S_l(1)$ (b). The true s_l is shown as a solid line.

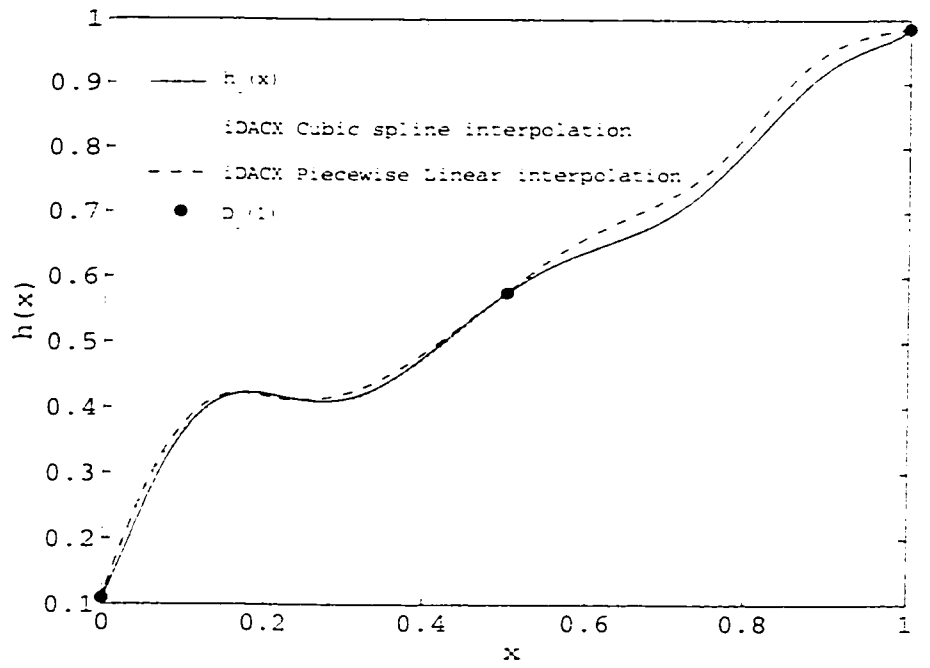


(a)

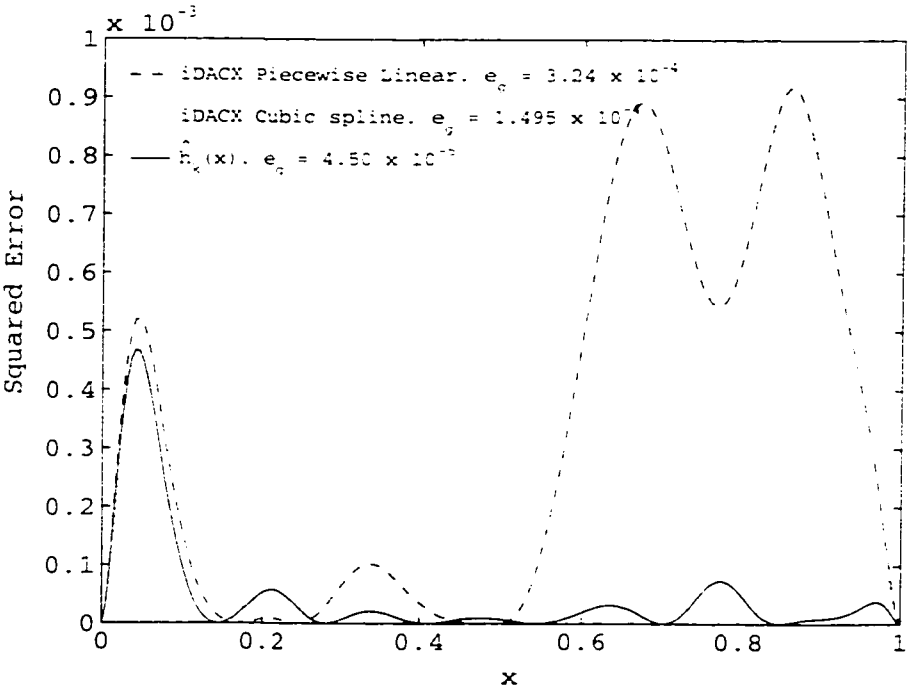


(b)

Figure 3.14. A calibration transfer using iDACX based on the five calibration data points in $D_l(2)$. A piecewise linear and a cubic spline interpolation are used with the data $S_1(2)$ to obtain $\hat{s}_l$. The squared errors in $\hat{h}_l = \hat{h}_k + \hat{s}_l$ are shown (b).



(a)



(b)

Figure 3.15. A calibration transfer using iDACX based on the three calibration data points in $D_l(1)$. A piecewise linear and a cubic spline interpolation are used with the data $S_1(1)$ to obtain $\hat{s}_l$. The squared errors in $\hat{h}_l = \hat{h}_k + \hat{s}_l$ are shown (b).

A plot of the squared errors over $\mathcal{X}$ for these approximations of $\hat{h}_l$ are also shown in Figure (3.14b) and (3.15b). A comparison of these approximation results and that provided by the PICX method can be made by considering the results shown in Figure (3.12a).

It is evident that, in spite of its simplicity, iDACX is highly effective in obtaining approximations that are significantly more accurate than approximations obtained with conventional interpolation methods using only the data in D_l. This effectiveness can be seen by noting that the iDACX method can be considered a form of instrument standardisation, as introduced by Wang *et al.* [2], which has been shown to be a viable method of calibration transfer.

It is also clear that the iDACX method is capable of providing approximations having errors that are comparable to that of the PICX method. The improvement provided by either the iDACX or the PICX methods will, of course, be dependent on the degree of *similarity* that exists between $\hat{h}_k$ and h_l. The degree of *similarity*, in a sense, determines the amount of relevant information that is available to use to obtain $\hat{h}_l$. Of course, the difficulty in quantifying these observations centers on defining similarity and its measure.

3.4.4 Calibration transfer using the $D^{(n)}$-iDACX methods

Given that prior knowledge of $\hat{h}_k$ appears to reduce the calibration error, it is natural to inquire as to the possibility of further reductions in the calibration error by providing the FFNN with the prior knowledge that $\hat{h}_l = \hat{h} + \hat{s}_l$. Section 2.4 described a number of methods that allowed a FFNN to incorporate prior knowledge. Of these methods, it is the method of hints, as reported by Abu-Mostafa [76, 77] and the use of virtual examples, as described by Niyogi, Girosi, and Poggio [75], that provide a point from which to begin to describe the development of the $D^{(n)}$-iDACX methods.

The idea in using the method of hints is to generate, what Abu-Mostafa calls duplicate examples. A straight forward application of this idea is to use the relationship

$\hat{h}_k + \hat{s}_l$ as a hint and to then provide these duplicate examples by generating the data set $V_k = \{(\mathbf{x}_j, \hat{h}_k(\mathbf{x}_j) + \hat{s}_l(\mathbf{x}_j)) : j = 1, 2, \ldots, N_j, \mathbf{x}_j \in [\min(\mathbf{x} \in X_l), \max(\mathbf{x} \in X_l)]\}$, where $X_l = \{\mathbf{x}_j : j = 1, 2, \ldots, N_j\}$. Note that the real data examples, provided by $D_l = \{(\mathbf{x}_i, \hat{h}_l(\mathbf{x}_i)) : i = 1, 2, \ldots, N_l\}$ are obtained physically, whereas the duplicate examples can be generated algorithmically. Thus, any number of duplicate examples can be generated.

The problem with this idea is that the supervised learning algorithm of the FFNN is now attempting to approximate the iDACX method which has already determined $\hat{h}_k + \hat{s}_l$. In other words, the best approximation that the FFNN could achieve, using these duplicate examples, would be that already determined directly by the iDACX method.

The information provided by data, or duplicate examples, generated using the hint that $\hat{h}_l = \hat{h}_k + \hat{s}_l$, $\forall x \in \mathcal{X}$, though representing prior information, does not provide additional information over that already used by the iDACX method. Additional information becomes available when it is realised that h_l is formed by a smooth functional relationship given by $\hat{h}_k + s_l$, $\forall \mathbf{x} \in \mathcal{X}$.

To obtain this additional information, it is necessary to realise that the data in D_l is not only a series of points exemplifying the relationship $h_l = \hat{h}_k + s_l$, but that the series of points are drawn from a continuous underlying function. This realisation then allows the relationship $h_l = \hat{h}_k + s_l$ at D_l to be expressed with a Taylor series expansion about each point in D_l.

To appreciate how this realisation can provide this additional information, consider the n dimensional Taylor series expansion of $h_l(\mathbf{x})$ about the ith point $\mathbf{x}_i = [x_{1i}, x_{2i}, \ldots, x_{ni}]^T \in X_l$. This expansion is given by [141]

$$h_l(\mathbf{x}) = h_l(\mathbf{x}_i) + \sum_{j=1}^{n} \left.\frac{\partial h_l(\mathbf{x})}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_i} \Delta x_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{q=1}^{n} \left.\frac{\partial^2 h_l(\mathbf{x})}{\partial x_j \partial x_q}\right|_{\mathbf{x}=\mathbf{x}_i} \Delta x_j \Delta x_q + \ldots . \tag{3.29}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, $\Delta x_j = x_j - x_{ji}$, and x_{ji} represents the jth component of

the ith point $\mathbf{x}_i$. In vector form Equation (3.29) becomes

$$h_l(\mathbf{x}) = h_l(\mathbf{x}_i) + \frac{1}{1!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^1 h_l(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i} + \frac{1}{2!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^2 h_l(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i} + \ldots$$
$$= \sum_{r=0}^{\infty} \frac{1}{r!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^r h_l(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i}. \tag{3.30}$$

where $< \cdot,\cdot >$ is the inner product, $\Delta\mathbf{x} = [(x_1 - x_{2i}).(x_1 - x_{2i}).\ldots.(x_n - x_{ni})]^T$, and $\nabla_{\mathbf{x}} = [\frac{\partial}{\partial x_1}.\frac{\partial}{\partial x_2}.\ldots.\frac{\partial}{\partial x_n}]^T$. Now use $h_l(\mathbf{x}) = \hat{h}_k(\mathbf{x}) + s_l(\mathbf{x})$ in Equation (3.30) to obtain

$$h_l(\mathbf{x}) = \sum_{r=0}^{\infty} \frac{1}{r!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^r \hat{h}_k(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i} + \sum_{r=0}^{\infty} \frac{1}{r!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^r s_l(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i}. \tag{3.31}$$

Given that $\hat{h}_k$ is described by a FFNN, the terms in its Taylor series expansion about any $\mathbf{x}_i$ can be determined. The terms for the Taylor series expansion of s_l about $\mathbf{x}_i$ are only know for the zero order terms and are given by $s_l(\mathbf{x}_i) = h_l(\mathbf{x}_i) - \hat{h}_k(\mathbf{x}_i)$. The higher order terms are not explicitly known and need to be estimated.

One approach to estimate these higher order terms is to use $\hat{s}_l$, which as described earlier, can be obtained by selecting an approximation model and using $S_l = \{(\mathbf{x}_i, h_l(\mathbf{x}_i) - \hat{h}_k(\mathbf{x}_i)) : i = 1, 2, \ldots, N_l\}$ to estimate the parameters of $\hat{s}_l$. The higher order Taylor series terms of s_l can then either be directly obtained from the higher order Taylor series terms of $\hat{s}_l$ or can be based on some estimator using $\hat{s}_l$. In the context of this estimation of the higher order terms, Equation (3.31) becomes

$$\tilde{h}_l(\mathbf{x}) = \sum_{r=0}^{\infty} \frac{1}{r!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^r \hat{h}_k(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i} + s_l(\mathbf{x}_i) + \sum_{r=1}^{\infty} \frac{1}{r!}\left[< \Delta\mathbf{x}.\nabla_{\mathbf{x}} >^r \hat{s}_l(\mathbf{x})\right]_{\mathbf{x}=\mathbf{x}_i}. \tag{3.32}$$

where $\tilde{h}_l$ is used to represent the approximation of h_l using $\hat{h}_k + \hat{s}_l$. Note that for the zero order terms, $\tilde{h}_l(\mathbf{x}_i) = h_l(\mathbf{x}_i)$.

Equation (3.32) indicates that additional information is available in the higher order Taylor series terms of $\hat{h}_k$, the zero order terms of s_l, and potentially in the higher order terms of $\hat{s}_l$. The information that is of most interest within the terms of the Taylor series expansion, is the information provided at the points $\mathbf{x} = \mathbf{x}_i$. At

these points $\Delta\mathbf{x}$ will be zero, but the values provided by $(\nabla_{\mathbf{x}})^r h_l(\mathbf{x})$ evaluated at $\mathbf{x}_i$ will not be zero. Therefore, at $\mathbf{x}_i$, it is the n dimensional rth order partial derivatives of the functions that are available as additional information.

This additional information regarding the n dimensional rth order partial derivatives, then raises the question of whether or not this information can be used advantageously. Some insight into answering this question can be gained by considering the generalised Shannon sampling, GSS, theorem [142, 143].

Briefly, the GSS theorem states that if samples of a bandlimited process include its derivatives up to order N_r, then the Nyquist sampling interval can be increased by a factor of $N_r + 1$. The bandlimited process can then be uniquely reconstructed using the ideal GSS interpolation formula. For example, assume $h(t)$ is a time t domain process bandlimited to B hertz. If the only samples of $h(t)$ are those given by $D = \{(t_n, h(t_n)) : t_n = nT, n = \ldots, 0, 1, \ldots\}$, where T is the sampling interval in seconds, then the sampling interval must be set to $T \leq (2B)^{-1}$ in order to reconstruct $h(t)$ completely using only the samples in D. If the samples in D also include the first derivative of $h(t)$ with respect to t, then the sampling interval can be increased by a factor of $N_r + 1 = 2$, or $T \leq B^{-1}$. In general if the derivatives up to order N_r are used, the sampling interval can be increased to $T \leq (N_r + 1)/2B$.

The GSS theorem suggests that the derivatives do provide additional information that is useful in reconstructing h. This additional information then allows the separation between sample points to be increased. Alternatively, the GSS theorem suggests that, if the interval between the sample points are fixed, then using derivatives up to order N_r allows h to increase its bandlimit without losing information vital to its reconstruction.

It should be noted that the idea of derivative sampling has been explored and found to be of limited usefulness for functions in the time domain by virtue that the noise in $h(t)$ limits the accuracy of the derivative samples [144], which become more susceptible to noise as the order of the derivative sample increases.

The relevance of this theorem to calibration transfer are two fold. First, the calibration model h_l is based on some underlying physical process which typically exhibits finite rates of change, not over time, but over $\mathbf{x}$, that is, h_l is in a sense bandlimited. Second, the use of higher order derivatives of h_l may provide an effect that is similar to that of decreasing the sampling interval by $N_r + 1$, or alternatively, provide an effect similar to that of increasing the number of samples.

The key point to note in the GSS theorem is the existence of an ideal GSS interpolation formula [142, 143] which allows the effective use of the rth order derivative information to uniquely reconstruct the bandlimited process. A FFNN is not an ideal interpolator in the context of the GSS, therefore, it will likely not use the rth order derivative information as effectively as the ideal GSS interpolation formula. Empirical results from Masuoka [89, 92], Thrun [91, 93], and Lee *et al.* [95] suggest that a FFNN can be made, by changing its learning algorithm, to take some advantage of this information. Specifically, these results have shown that use of partial derivatives up to only the first order can, in most cases, provide improvements in generalisation error and learning speed.

The $D^{(n)}$ and $D^{(1)}$ learning algorithm

The most direct manner to proceed to allow the FFNN to use the information provided by the higher order partial derivatives is to simply extend the approach used in supervised learning. In conventional supervised learning the goal is to

$$\min_{\mathbf{w}\in W} d\left(h_l(\mathbf{x}_i;\mathbf{w}), \hat{h}_l(\mathbf{x}_i;\mathbf{w})\right)^2 \qquad \forall(\mathbf{x}_i \in X_l), i = 1, 2, \ldots, N_l. \tag{3.33}$$

where distance metric $d(\cdot,\cdot)$ is typically defined with the l_2-norm in a N_l dimensional space. The minimisation is then over

$$\begin{aligned} C &= d\left(h_l(\mathbf{x}_i), \hat{h}_l(\mathbf{x}_i)\right)^2, \\ &= \sum_{i=1}^{N_l} \left(h_l(\mathbf{x}_i) - \hat{h}_l(\mathbf{x}_i)\right)^2. \end{aligned} \tag{3.34}$$

which as noted in section 3.3.1, is also referred to as the SSE and defines the error surface over which learning occurs.

To apply the higher order derivative information, the goal would be to

$$\min_{\mathbf{w}\in W} d\left(\tilde{\mathbf{h}}_l^{(N_r)}(\mathbf{x}_i;\mathbf{w}), \hat{\mathbf{h}}_l^{(N_r)}(\mathbf{x}_i;\mathbf{w})\right)^2 \qquad \forall(\mathbf{x}_i \in X_l). \tag{3.35}$$

where, it should be noted that, $\mathbf{h}_l^{(N_r)} = [h_l^{(0)}, h_l^{(1)}, \ldots, h_l^{(N_r)}]^T$ represents the vector with N_r+1 components, where the mth component represents the n dimensional mth order partial derivative of h_l with respect to $\mathbf{x}$, that is $(\nabla_{\mathbf{x}})^m h_l$. In other words, the N_r+1 components of $\mathbf{h}_l$ represent a point in a N_r+1 dimensional space. This allows measuring the distance between $\tilde{\mathbf{h}}_l$ and $\hat{\mathbf{h}}_l$ by simply using the distance between the two points represented by $\tilde{\mathbf{h}}^{(N_r)}$ and $\hat{\mathbf{h}}^{(N_r)}$. Again, the l_2-norm is the basis on which the measure of the distance between $\tilde{\mathbf{h}}^{(N_r)}$ and $\hat{\mathbf{h}}^{(N_r)}$ is made.

Now, instead of a single point at $\mathbf{x}_i$, there are N_l points provided by D_l. Therefore, in the same manner as was done for a single point at $\mathbf{x}_i$, define $\tilde{H}_l^{(N_r)} = [(\tilde{\mathbf{h}}_{l1}^{(N_r)})^T, (\tilde{\mathbf{h}}_{l2}^{(N_r)})^T, \ldots, (\tilde{\mathbf{h}}_{lN_l}^{(N_r)})^T]^T$ and $\hat{H}_l^{(N_r)} = [(\hat{\mathbf{h}}_{l1}^{(N_r)})^T, (\hat{\mathbf{h}}_{l2}^{(N_r)})^T, \ldots, (\hat{\mathbf{h}}_{lN_l}^{(N_r)})^T]^T$ as two vectors that represent two points in a $N_l(\sum_{r=0}^{N_r} n^r)$ dimensional space. Expressing the distance between these points with

$$C^{(N_r)} = d\left(\tilde{\mathbf{H}}_l^{(N_r)}, \hat{\mathbf{H}}_l^{(N_r)}\right)^2.$$

provides the needed cost function to be minimised. Again, using the l_2-norm as the basis for the distance measure, results in the expanded expression

$$\begin{aligned} C^{(N_r)} = {} & \lambda_0 \sum_{i=1}^{N_l} \left(\tilde{h}_l(\mathbf{x}_i) - \hat{h}_l(\mathbf{x}_i)\right)^2 + \lambda_1 \sum_{i=1}^{N_l} \sum_{r_1=1}^{n} \left(\frac{\partial \tilde{h}_l(\mathbf{x}_i)}{\partial x_{r_1}} - \frac{\partial \hat{h}_l(\mathbf{x}_i)}{\partial x_{r_1}}\right)^2 \\ & + \lambda_2 \sum_{i=1}^{N_l} \sum_{r_1=1}^{n} \sum_{r_2=1}^{n} \left(\frac{\partial^2 \tilde{h}_l(\mathbf{x}_i)}{\partial x_{r_1} \partial x_{r_2}} - \frac{\partial^2 \hat{h}_l(\mathbf{x}_i)}{\partial x_{r_1} \partial x_{r_2}}\right)^2 + \cdots \\ & + \lambda_{N_r} \sum_{i=1}^{N_l} \sum_{r_1=1}^{n} \cdots \sum_{r_{N_r}=1}^{n} \left(\frac{\partial^{N_r} \tilde{h}_l(\mathbf{x}_i)}{\partial x_{r_1} \cdots \partial x_{r_{N_r}}} - \frac{\partial^{N_r} \hat{h}_l(\mathbf{x}_i)}{\partial x_{r_1} \cdots \partial x_{r_{N_r}}}\right)^2. \end{aligned} \tag{3.36}$$

where the partial derivatives are evaluated at $\mathbf{x}_i$ and λ represents a weighting factor for each term, such that $\lambda \in [0,1]$ and $\sum_i \lambda_i = 1$. To more conveniently represent

Equation (3.38). the expression

$$C^{(N_r)} = \sum_{i=1}^{N_l} \sum_{r=0}^{N_r} \lambda_r \left((\nabla_{\mathbf{x}})^r \tilde{h}_l(\mathbf{x}_i) - (\nabla_{\mathbf{x}})^r \hat{h}_l(\mathbf{x}_i) \right)^2 .$$
$$= \|\tilde{\mathbf{H}}_l^{(N_r)} - \hat{\mathbf{H}}_l^{(N_r)}\|_2^2. \qquad (3.37)$$

is used. where $(\nabla_{\mathbf{x}})^r h(\mathbf{x}_i)$ is the n dimensional rth order partial derivative of h with respect to $\mathbf{x}$ evaluated at $\mathbf{x} = \mathbf{x}_i$ and $\|\cdot\|_2$ is the l_2-norm. The expression for $C^{(N_r)}$. given by Equation (3.37). is referred to as the generalised sum squared error. GSSE. It should be noted that for $N_r = 0$. the GSSE reduces to the SSE. The GSSE also defines a new error surface over which learning occurs. The use of $C^{(n)}$ as the cost or error function for supervised learning will be referred to as a $D^{(n)}$ learning algorithm.

Based on previous results. as reported by Masuoka [89. 92]. Thrun [91. 93]. and Lee *et al.* [95]. only the terms up to the first order derivative in $C^{(N_r)}$ will be considered in this thesis. that is $N_r = 1$. so Equation (3.37) reduces to

$$C^{(1)} = \lambda_0 \sum_{i=1}^{N_l} \left(\tilde{h}_l(\mathbf{x}_i) - \hat{h}_l(\mathbf{x}_i) \right)^2 + \lambda_1 \sum_{i=1}^{N_l} \sum_{j=1}^{n} \left(\left[\frac{\partial \tilde{h}_l(\mathbf{x}_i)}{\partial x_j} - \frac{\partial \hat{h}_l(\mathbf{x}_i)}{\partial x_j} \right] \right)^2 . \qquad (3.38)$$

The learning algorithm proposed in this thesis. which is essentially a search for a minimum in $C^{(1)}$. is based on steepest descent and will also be referred to as the $D^{(1)}$ learning algorithm. The use of $C^{(0)}$ will be referred to as the $D^{(0)}$ learning algorithm.

To implement these learning algorithms requires explicitly determining the gradients of $C^{(1)}$ or $C^{(0)}$ with respect to the weight parameters of the FFNN. These weight parameters are expressed in the Equation (3.9). which for convenience. is repeated here as

$$\hat{h}_l(\mathbf{x}_i) = \sum_{k=1}^{m_1} \left(c_k^1 o_{1k} \left(\sum_{j=1}^{n} w_{kj}^1 x_j + w_{k0}^1 \right) + b_k^1 \right) .$$

or in vector form. as

$$= [\vec{o}^T(\mathbf{x}_i^T \mathbf{w}_k + w_{k0}) + \mathbf{b}^T]\mathbf{c}. \qquad k = 1.2.\ldots.m_1. \qquad (3.39)$$
$$= [\vec{o}^T(\mathbf{x}_a^T \mathbf{w}_{k_a}) + \mathbf{b}^T]\mathbf{c}. \qquad k = 0.1.2.\ldots.m_1.$$

where, for the single layer FFNN, the superscripts have been drop, the weight parameters are w_{k0}, $\mathbf{w}_k$, $\mathbf{b}$, and $\mathbf{c}$, and where $\mathbf{w}_{k_a} = [w_{k0}, (\mathbf{w}_k)^T]^T$. Also, $\vec{o} = [o_1(\mathbf{x}_a^T \mathbf{w}_{1_a}), o_2(\mathbf{x}_a^T \mathbf{w}_{2_a}), \ldots, o_{m_1}(\mathbf{x}_a^T \mathbf{w}_{m_{1_a}})]^T$. m_1 is the number of hidden neurons, and where

$$o(u) = \frac{1}{1 + e^{-u}}. \tag{3.40}$$

is the logistic function.

Now the resulting gradient of $C^{(1)}$ with respect to the weight parameters $\mathbf{w}_k$, w_{k0}, $\mathbf{b}$, and $\mathbf{c}$ can, as described in appendix C, be given by

$$\frac{\partial C^{(1)}}{\partial \mathbf{w}} = \sum_{i=1}^{N_l} \lambda_0 \left(\frac{\partial \hat{h}_l(\mathbf{x})}{\partial \mathbf{w}} (\hat{h}_l(\mathbf{x}_i) - \tilde{h}_l(\mathbf{x}_i)) + \lambda_1 \sum_{j=1}^{n} \left(\frac{\partial \nabla_{x_j} \hat{h}_l(\mathbf{x}_i)}{\partial \mathbf{w}} \left[\nabla_{x_j} \hat{h}_l(\mathbf{x}_i) - \nabla_{x_j} \tilde{h}_l(\mathbf{x}_i) \right) \right) \right),$$
$$\mathbf{w} \in \{\mathbf{w}_k, w_{k0}, \mathbf{b}, \mathbf{c}\}. \tag{3.41}$$

where $\nabla_{x_j} h(\mathbf{x}_i)$ is the gradient of h with respect to jth component of $\mathbf{x}$, that is, x_j, evaluated at $\mathbf{x} = \mathbf{x}_i$. Equation (3.41) can be more compactly expressed using

$$\frac{\partial C^{(1)}}{\partial \mathbf{w}} = \frac{\partial C^{(0)}}{\partial \mathbf{w}} + \frac{\partial (C^{(1)} - C^{(0)})}{\partial \mathbf{w}}. \tag{3.42}$$

It is clear from Equation (3.41) that there are four gradient components and that each gradient component consists of term due to the difference in the data values and a term due to the difference in slope values.

The explicit expressions for each of the gradient components requires evaluating $\frac{\partial \hat{h}_l(\mathbf{x})}{\partial \mathbf{w}}$, $\nabla_{x_j} \hat{h}_l(\mathbf{x}_i)$, and $\frac{\partial \nabla_{x_j} \hat{h}_l(\mathbf{x}_i)}{\partial \mathbf{w}}$ which is the partial derivative of gradient of $\nabla_{x_j} \hat{h}_l(\mathbf{x}_i)$ with respect to the weight parameter $\mathbf{w}$. These expressions are derived in appendix C.

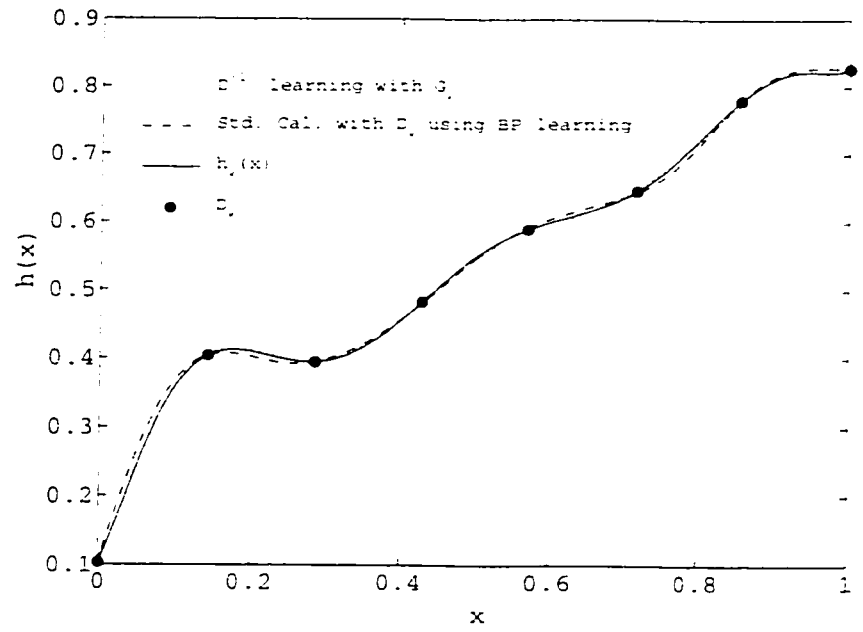
The updating of the weights is then made according to the method of steepest descent

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \Delta \mathbf{w}(n), \\ &= \mathbf{w}(n) - \eta(n) \frac{\partial C^{(1)}}{\partial \mathbf{w}(\mathbf{n})}. \qquad \mathbf{w} \in \{\mathbf{w}_k, w_{k0}, \mathbf{b}, \mathbf{c}\}. \end{aligned} \tag{3.43}$$

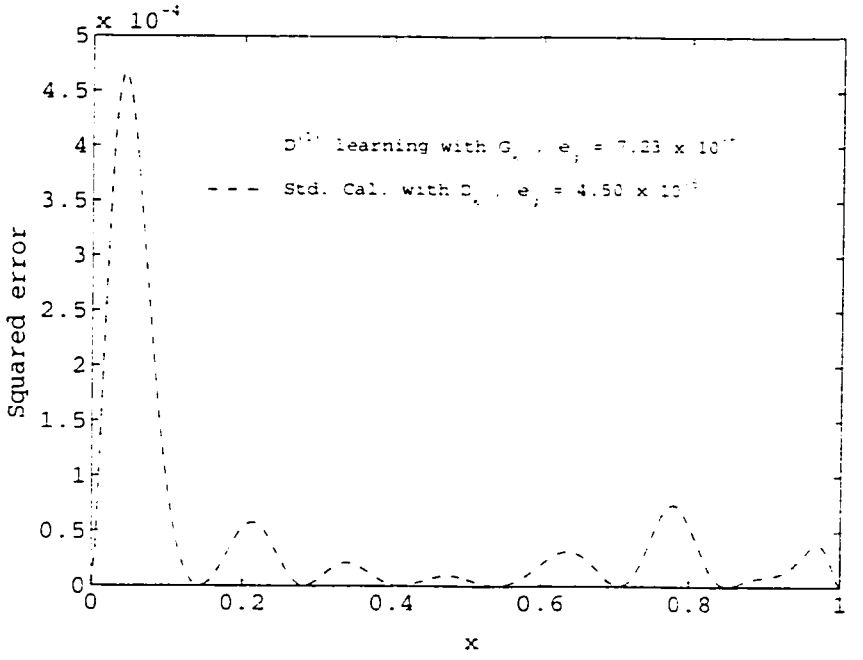
where η is the learning rate parameter that generally depends on the descent progress. Note that η can be distributed over λ_0 and λ_1 resulting in a total of eight learning rate parameters that in general are not equal.

Continuing with the same example used to illustrate the PICX and iDACX methods, consider the initial calibration that obtains $\hat{h}_k$ with the data in D_k using a standard calibration with backpropagation learning and $D^{(1)}$ learning. Figure (3.16) shows the results of this initial calibration obtaining $\hat{h}_k$. As an initial calibration, there is no prior calibration model that can be used to help in the current calibration. Therefore, an initial calibration is not precisely the same as a calibration transfer, in the sense described in this thesis. On the other hand, the initial calibration can be viewed as a calibration transfer where the prior calibration model is not used. Regardless, of the view of the initial calibration, the learning algorithms need to initialise the weights of the FFNN. Given that there is no prior calibration model to use, the initial weight values of the FFNN are randomised using the Nguyen-Widrow method [140] method. In addition, the the $D^{(1)}$ learning algorithm requires a set of desired slope values to learn, that is, the calibration data set for the $D^{(1)}$ learning algorithm is given by $G_k = D_k \cup S_l^{(1)}$, where $S_l^{(1)} = \{(x_i, dh_k(x)/dx|_{x=x_i}) : (x_i, h(x_i)) \in D_k\}$. In other words, for this example the true slope values of h_k are used as the desired slope values. In a more practical setting $S_l^{(1)}$ would to be based on estimated slope values having some degree of error. The influence of this error on $D^{(1)}$ learning is explored in chapter 5. Figure (3.16b) shows the squared error of the resulting approximation over $\mathcal{X}$ and lists $\hat{e}_g$ that is estimated with $N_j = 200$.

Figure (3.16) does suggest that $D^{(1)}$ learning can provide a reduction in the calibration error as compared to the calibration error resulting from a standard calibration using backpropagation learning. Of course, this is but one calibration example and the degree of reduction in the calibration error will change as the calibration conditions change, that is, as conditions such as, the number of calibration data samples and the error in the slope estimates are changed.

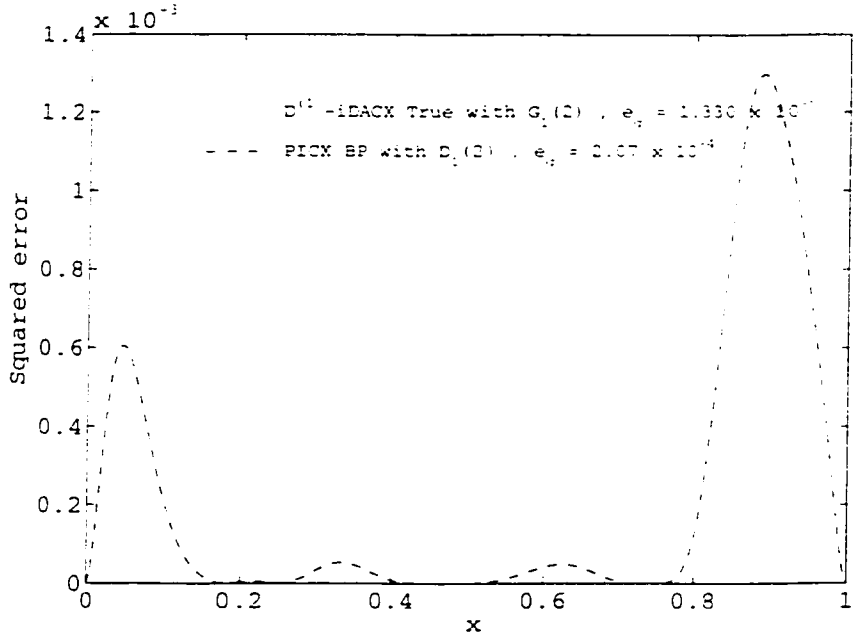


(a)

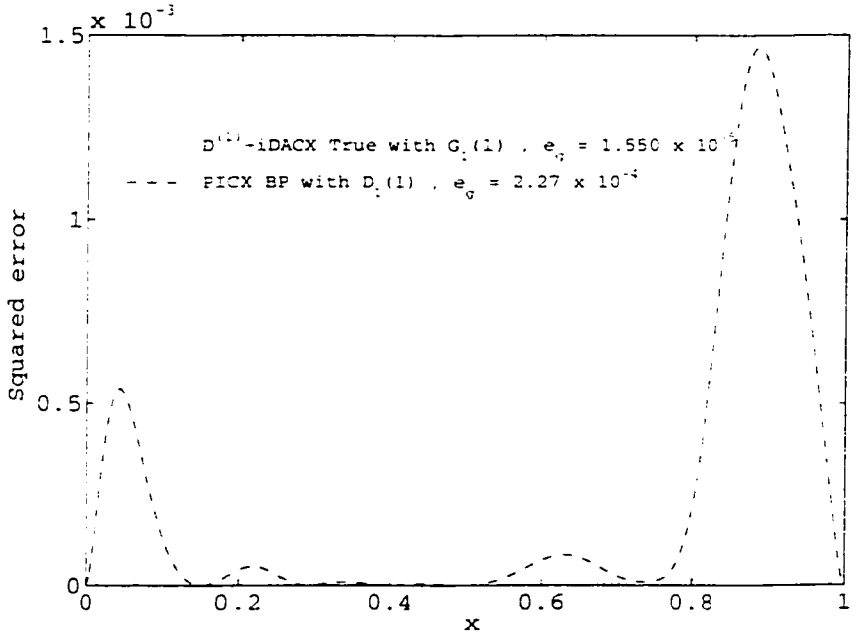


(b)

Figure 3.16. The results of an initial calibration using $D^{(1)}$ learning and standard calibration using backpropagation learning. In (a) the resulting FFNN approximations of h_k using D_k are shown and in (b) the squared error in the approximations over $\mathcal{X}$ are shown.



(a)



(b)

Figure 3.17. The squared calibration error after a calibration transfer from $\hat{h}_k$ to $\hat{h}_l$ using $D^{(1)}$-iDACX True and PICX with back-propagation, PICX BP. In (a), $D_l(2)$ having five calibration data points is used and in (b) $D_l(1)$ having three calibration data points is used. See Figure (3.9) for h_k, h_l and calibration data set.

Now consider performing a calibration transfer using the $D^{(1)}$-iDACX and the PICX methods with the data defined by $D_l(1)$ and $D_l(2)$. In this example, the calibration transfer is from $\hat{h}_k$ to $\hat{h}_l$. Again, the $D^{(1)}$-iDACX, which is based on $D^{(1)}$ learning, requires a desired slope value to learn. For this example the calibration data set used for the $D^{(1)}$-iDACX method is given by $G_l(m) = D_l \cup S_l^{(m)}$, where $S_l^{(m)} = \{(x_i, dh_l(x)/dx|_{x=x_i}) : (x_i, h(x_i)) \in D_l(m)\}$, for $m = \{1, 2\}$. In other words, the true slope values of h_l are used as the desired slope values. In a more practical setting $S_l^{(m)}$ would to be based on estimated slope values having some degree of error. Again, the influence of this error on $D^{(1)}$-iDACX method is explored in chapter 5.

To distinguish the use of different slope estimates for the $D^{(1)}$-iDACX method, the nomenclature given by, for example, $D^{(1)}$-iDACX True, will be used, where the adjective "True" will be replaced with an appropriate adjective describing the estimation technique used to obtain the desired slope values. In a similar manner the learning algorithm used in the PICX method will be identified using, for example, PICX BP, where the adjective "BP" will be replaced with an appropriate adjective describing the learning algorithm used in the PICX method.

It should also be noted that both the PICX BP and $D^{(1)}$-iDACX True methods used weight values that were initialised to the values that provided their corresponding prior approximation of h_k. In other words, the initial weight values for the PICX BP method were set to provide the approximation $\hat{h}_k$ obtained using a standard calibration with backpropagation learning. In the same manner, the initial weight values for the $D^{(1)}$-iDACX True method were set to provide the approximation $\hat{h}_k$ obtained using $D^{(1)}$ learning.

Figure (3.17) shows the squared calibration error over $\mathcal{X}$ resulting from approximating h_l using the PICX BP and $D^{(1)}$-iDACX True methods. Specifically, Figure (3.17a) shows the squared error when using only five calibration data points, that is, the data defined by $D_l(2)$ for the PICX BP method and $G_l(2)$ for $D^{(1)}$-iDACX True method. Figure (3.17b) shows the squared error when using only three calibra-

tion points, again defined by data sets $D_l(1)$ for the PICX BP method and $G_l(1)$ for $D^{(1)}$-iDACX True method.

Again, it is apparent from Figure (3.17) that the use of the $D^{(1)}$-iDACX method can achieve calibration errors that are less than that provided by the PICX BP method. As noted earlier, this reduction in the calibration error will dependent on the calibration conditions such as, the error in the slope estimates, the degree of similarity between h_k and h_l, and the actually data used to perform the calibration transfer. The dependency of the calibration error on these calibration conditions will be explored in chapter 5.

4. Simulation Methods

4.1 Overview

The chapter describes the methodology used to perform the initial instrument calibration and subsequent calibration transfer simulations. The initial calibration simulations of the FFNN based instrument are designed to obtain the calibration model $\hat{h}_k$ that is subsequently used in the calibration transfer simulations. To obtain the initial calibration model, five different learning strategies are used. These strategies, as identified with the acronyms listed below, employ the following methods:

1. BP. Learning with conventional backpropagation using only the data set D_k where the weights are initialised using the Nguyen-Widrow method [140].

2. $D^{(0)}$. Learning with the $D^{(0)}$ learning algorithm using only the data set D_k where the weights are initialised using a modified version of the Nguyen-Widrow method.

3. $D^{(1)}$ True. Learning with the $D^{(1)}$ learning algorithm using the data set G_k that includes the true slope values of h_k at x_i.

4. $D^{(1)}$ Spline. Learning with the $D^{(1)}$ learning algorithm using the data set G_k that includes slope values based on a cubic spline approximation of h_k at x_i.

5. $D^{(1)}$ Linear. Learning with the $D^{(1)}$ learning algorithm using the data set G_k that includes slope values based on a piecewise linear approximation of h_k at x_i.

All the $D^{(1)}$ learning strategies initialise the FFNN weights using a modified version of the Nguyen-Widrow method. The calibration error resulting from the initial

calibration using the methods listed in items (2) through (5) are compared to the calibration error resulting from the BP method, item (1).

The calibration transfer simulations from $\hat{h}_k$ to $\hat{h}_l$ apply the three methods described in chapter 3, that is the methods referred to as PICX, iDACX, and $D^{(1)}$-iDACX. Each method, as identified with the acronym listed below, are applied with specific variations.

1. Std. Cal. BP. A transfer that is simply a standard calibration of the FFNN instrument obtaining $\hat{h}_l$ with conventional backpropagation learning using only the data set D_l where the weights are initialised using the Nguyen-Widrow method [140].

2. PICX BP. A transfer that obtains $\hat{h}_l$ from $\hat{h}_k$ using the data set D_l and a learning algorithm based on conventional backpropagation where the weights are initialised to the state providing the approximation $\hat{h}_k$.

3. PICX $D^{(0)}$. Similar to PICX BP, except the learning algorithm is based on $D^{(0)}$ learning.

4. iDACX Spline. A transfer that obtains $\hat{h}_l$ using $\hat{h}_k + \hat{s}_l$, where $\hat{s}_l$ is based on a cubic spline approximation using the data in S_l.

5. iDACX Linear. A transfer that obtains $\hat{h}_l$ using $\hat{h}_k + \hat{s}_l$, where $\hat{s}_l$ is based on a piecewise linear approximation using the data in S_l.

6. $D^{(1)}$-iDACX True. A transfer that obtains $\hat{h}_l$ using the data set $G_l^{(1)} = D_l \cup D_l^{(1)}$ and the $D^{(1)}$ learning algorithm. The slopes at $x_i \in X_k$ are true slope values of $dh_l(x)/dx$.

7. $D^{(1)}$-iDACX Spline. A transfer that is similar to $D^{(1)}$-iDACX True, except that the slopes at $x_i \in X_k$ are obtained from $d(\hat{h}_k(x) + \hat{s}_l(x))/dx$ where $\hat{s}_l$ is based on a cubic spline approximation using the data in S_l.

8. $D^{(1)}$-iDACX Linear. A transfer that is similar to $D^{(1)}$-iDACX Spline, except the slopes at $x_i \in X_k$ are obtained from $d(\hat{h}_k(x) + \hat{s}_l(x))/dx$ where $\hat{s}_l$ is based on a piecewise linear approximation using the data in S_l.

The calibration error resulting from the calibration transfer using the methods listed in items (2) through (8) are compared to the calibration error resulting from using the Std. Cal. BP method, item (1).

The chapter begins by defining the calibration parameters used for the simulations as well as discussing the rationale used in selecting the values for these parameters. The values used for the FFNN parameters in the initial calibration and calibration transfer simulations are then presented. Finally, the methods used to analyse the calibration errors are discussed. It should be noted that for any given set of values assigned to these parameters constitute what will be referred to as the calibration condition for a simulation trial. A change in the value of any one parameter results in a new calibration condition.

4.2 Calibration Parameter Definitions

4.2.1 Sensor output space $\mathcal{X}$

To more easily visualise, interpret, and understand the results, the simulations are limited to occur over $\mathcal{X} \subset \mathbb{R}$, that is, the calibration model $\hat{h}_k(x)$ is univariate. In using a univariate calibration model, it is hoped that the behaviour of the calibration error resulting from changes in the values of the calibration parameters will be more easily identified over $\mathcal{X} \subset \mathbb{R}$ than over $\mathcal{X} \subset \mathbb{R}^n$. It is assumed that the calibration transfer methods will have analogous behaviour over $\mathcal{X} \subset \mathbb{R}^n$, though the behaviour may be more difficult to recognise and analyse in higher dimensions.

The use of only one dimension will also reduce the computational burden in performing the simulations by virtue that with only $n = 1$ there are less terms to evaluate in the gradient calculations required by Equation (3.41). In addition, with only one

input dimension the number of representative points needed for a calibration or a calibration transfer are also reduced.

It should be noted that, with $\mathcal{X} \subset \mathbb{R}$, it is a simple matter to obtain piecewise linear and cubic spline approximations of either h_k or s_l. Analogous approximation in $\mathbb{R}^n$ will be more difficult and will demand more data to obtain adequate approximations. Note that the actual simulations will be limited to some finite interval defined by the set $X \subset \mathcal{X}$, that is, the set X is subset of the set associated with the space $\mathcal{X}$.

4.2.2 The true calibration model h

It is difficult to determine the characteristics of the true underlying calibration model h that are needed to accentuate or reveal the limitations of the various calibration transfer methods. In this sense, the selection of the true underlying calibration model h is almost arbitrary. Therefore, a practical approach in selecting h is used: the calibration model is selected from a class of functions that are *reasonably* general, in that they include a subset of functions that can represent a wide range of typical instrument calibration models. Of course, the rationale for choosing this approach is that it is assumed that if the calibration transfer methods provide adequate performance for this *general* h, then the transfer methods should provide adequate performance for a less *general* subset of functions.

To help guide the *generalisation* of h, consider some of the characteristics of an instrument calibration model that are desirable. First, in most real instruments h is assumed to be *smooth*, in that it is assumed to possess at least a first order derivative with respect to x over all $\mathcal{X}$. Second, h should possess an inverse, that is, $h \circ h^{-1} = I$, where I is the identity function $I(x) = x$. Also, it is conducive in instrument design to have h be monotonic: $\forall x_1, x_2 \in \mathcal{X}$, $x_1 < x_2$, results in $y_1 = h(x_1) < y_2 = h(x_2)$. Finally, characteristics of h that are in many instances simply taken for granted include h being bounded, that is, $\forall x \in \mathcal{X}$, $|h(x)| \leq B_m < \infty$, and having bounded rates of change over $\mathcal{X}$: $\forall x \in \mathcal{X}$, $|dh(x)/dx| \leq B_r < \infty$. A more insightful concept

in conveying a sense of finite rates of change is that of assigning a *bandlimit* to h evaluated not over time but over $\mathcal{X}$.

Though these characteristics can be attributed to many real instrument calibration models, in this thesis h is simply assumed to be *smooth*, bounded, and having a finite bandlimit. The need for an inverse and monoticity is not relied upon. It is clear that removal of the need to have an inverse and to be monotonic over $\mathcal{X}$ provides a more general h, in that functions that are smooth and have finite bandlimits include, as their subsets, functions which also have inverses and are monotonic.

Another characteristic that is to be attributed to h is in excluding functions that can be precisely approximated with a FFNN using a small finite number, p, of hidden neurons having sigmoidal type activation functions. In other words, it is assumed that to reduce the approximation error e_a, the FFNN's approximation will require more neurons, that is, $e_a \to 0$ as $p \to \infty$. Though this attribute of h is typically satisfied by many functions, it is simply stated to prevent the possibility of all calibration transfer methods exhibiting very small calibration errors simply because $e_a = 0$ for some finite number of neurons. Exhibiting small calibration errors would then make it difficult to determine the relative error performance of the various calibration transfer methods.

Given these characteristics, two classes of functions will be used in these simulations for the calibration model h:

1. The class $\mathcal{P}$ which includes 8th order univariate polynomials defined by

$$h_{P_k}(x) = \sum_{i=0}^{8} a_i x^i, \qquad a_i \in \mathbb{R}, x \in \mathcal{X}. \tag{4.1}$$

   where the notation h_{P_k} is used to identify the kth element from the set $\mathcal{H}_P \subset \mathcal{P}$.

2. The class $\mathcal{R}$ which includes normal random processes defined as a function of $x \in \mathbb{R}$, bandlimited to B_k cycles per unit x and having zero mean and variance of one, $N(0, 1)$. A segment of the process over $\mathbb{R}$ is taken so as to span $\mathcal{X}$. The resulting function over $\mathcal{X}$ is then defined as $h_{R_k}(x)$, where again the notation

h_{R_k} is used to identify the kth element from the set $\mathcal{H}_R \subset \mathcal{R}$. It should be noted that the bandlimit B_x is set such that a number of *cycles* of h_{R_k} span $\mathcal{X}$.

Note that the notation for the calibration model h or $\hat{h}$ will be used if it is not necessary or relevant to the discussion to identify which of the above two classes of functions is being considered.

4.2.3 Calibration model similarity

To generate *similar* functions of h, the selected functions are made to vary over a second dimension that can either be interpreted as time or as other instances of an instrument model. For convenience, this second dimension will be denoted as u. The true calibration model can then be viewed as being described by $h(x, u)$.

To provide a degree of control, the changes in h over u are described using another normal random process that is bandlimited to B_u cycles per unit u. The kth instance of the calibration model h_k is then represented by setting $u = u_k$ and using $h(x, u = u_k)$ for $x \in \mathcal{X}$. To obtain the lth instance of the calibration model h_l, a sample is taken at a different instance of u, that is at $u = u_l$. To obtain instances of h_l that are more similar to h_k, the interval $\tau = u_k - u_l$ is made smaller. With a bandlimit of B_u cycles per unit u, setting $\tau = (2B_u)^{-1}$ will result in instances of h at any $x \in \mathcal{X}$ to be uncorrelated to each other, that is

$$\begin{aligned} R_h(\tau, x_i) &= E[h(x_i, u_k)h(x_i, u_k + \tau)], \qquad x_i \in \mathcal{X}. \\ &= E[h_k(x_i)h_l(x_i)] = 0. \end{aligned}$$

where $R_h(\tau, x_i)$ is the autocorrelation function and $E[\cdot]$ is the expectation operator evaluated over u, and where zero means are assumed. Decreasing τ will increase the correlation of the instances of h and provide an increase in the *similarity* between h_k and h_l.

For the class of function $\mathcal{P}$ describing the 8th order polynomial, the coefficients a_i are set to be independent normal random processes each bandlimited to 0.0625 cycles

per unit u. The polynomial process is then given by

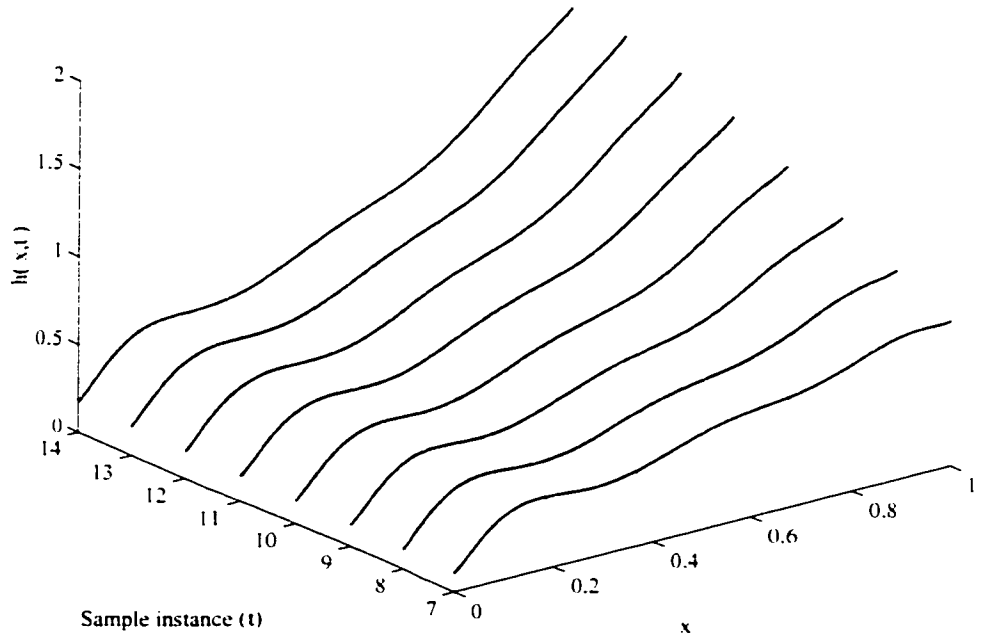
$$h_{P_k}(x, u) = \sum_{i=0}^{8} \mathbf{a}_i(u) x^i, \qquad a_i \in \mathbb{R}, x \in \mathcal{X}. \tag{4.2}$$

Specific instances of h_{P_k} over $\mathcal{X}$ are taken at intervals of one unit of u, so that $u = k = \ldots, 0, 1, 2, \ldots$. Therefore, every $(2B_u)^{-1} = (2(0.0625))^{-1} = 8$th instance of h_{P_k} will have corresponding coefficients $\mathbf{a}_i(k)$ and $\mathbf{a}_i(k+7)$ that, given zero means, will have small, that is, near zero correlation. In the case of non zero means, the correlation will be a constant non zero value. Figure (4.1a) shows the specific realisation of the 8th order polynomial random process over the instance of $u = \{7, 8, 9, \ldots, 14\}$. The specific elements of $\mathcal{P}$ are shown as solid lines over x and u and represent specific realisations of h_{P_k}.

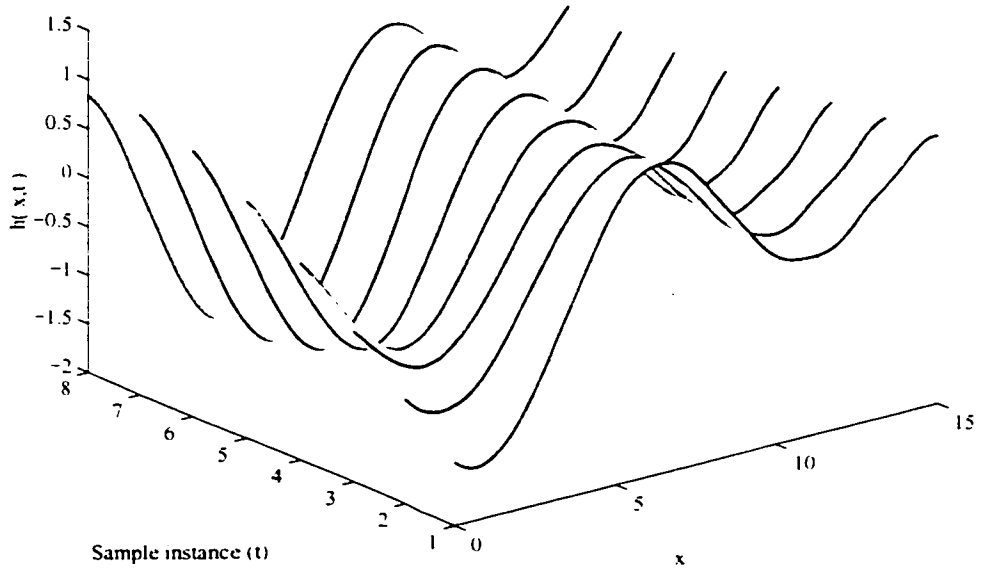
For the class of functions defined by $\mathcal{R}$, which is described by a normal random process bandlimited to B_x cycles per unit x, the variation over u is also defined as another independent normal random bandlimited process defined over u. The specific parameters used to define some of the members of $\mathcal{R}$ include setting $B_x = 0.125$ cycles per unit x and $B_u = 0.0625$ cycles per unit u. The domain over $\mathcal{X}$ is defined to span 16 units of x, that is arbitrarly defined as $X = [0, 15] \subset \mathcal{X}$. Again, instances of the process over u are sampled at intervals of one unit of u. Figure (4.1b) shows the general random process over the instance of $u = \{1, 2, \ldots, 8\}$. The specific elements of $\mathcal{R}$ are shown as solid lines over x and u and represent specific realisations of h_{R_k}.

Figure (4.2) illustrates the degree of *similarity*, over $\mathcal{X}$, that exists between the elements of h for both $\mathcal{P}$ and $\mathcal{R}$.

The similarity between the approximation of h_k and h_l is therefore obtained by using different members from $\mathcal{P}$ or $\mathcal{R}$. The members in either group have been selected to provide, $\forall x_i \in X$, a decreasing correlation between $h_k(x_i)$ and $h_l(x_i)$ as l increases relative to k. For example, the estimated correlations between $h_{P_7}(x = 0)$ and $h_{P_l}(x = 0)$ for $l = \{8, 10, 14\}$ are, given by $\hat{R}_{h_P}(\tau = l - 7, x = 0) \approx \{0.97, 0.8, 0.2\}$. It should be noted that in general the correlation varies with x. For example, the estimate correlation at $x = 1$, given $l = \{8, 10, 14\}$, is $\hat{R}_{h_P}(\tau = l - 7, x = 1) \approx \{0.95, 0.75, 0.05\}$.

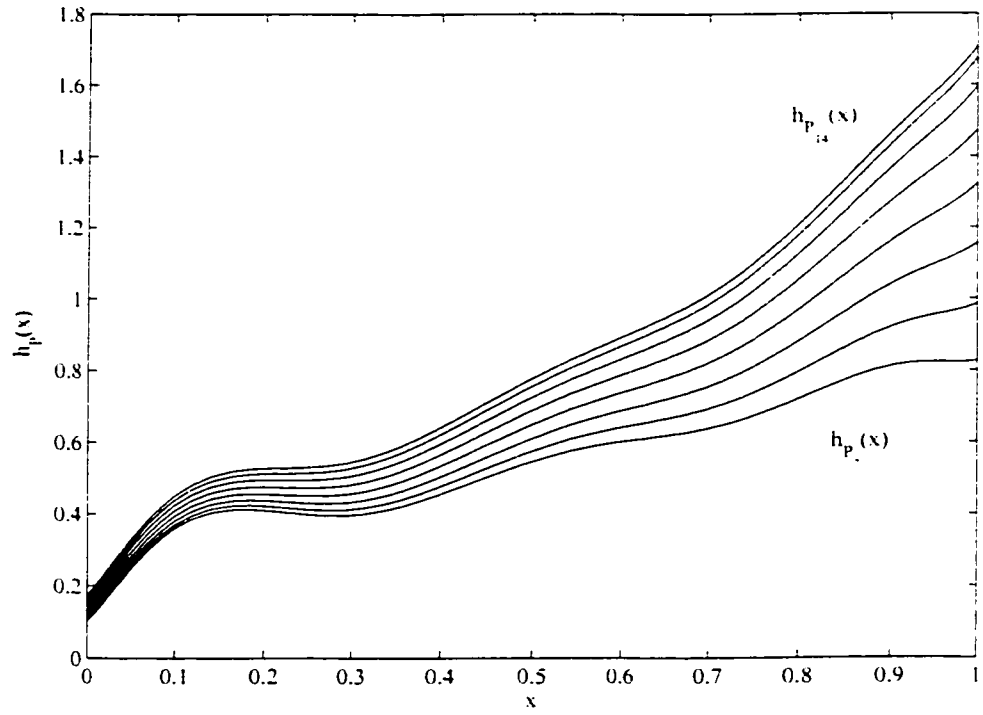


(a)

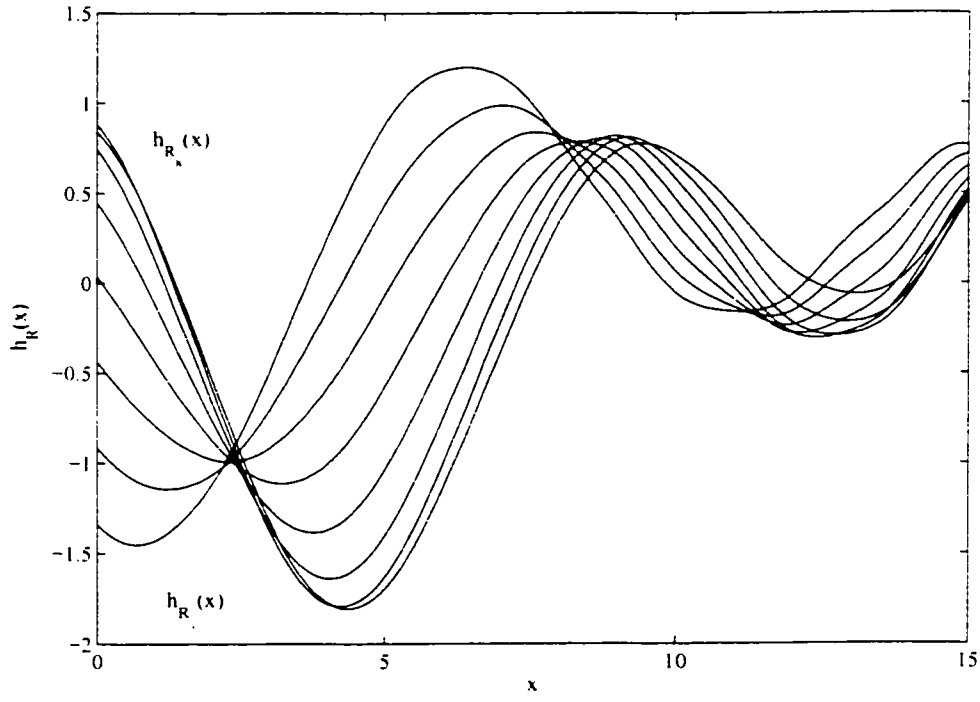


(b)

Figure 4.1. The two sets of true underlying calibration models h used for the simulations shown over X and instances u. In (a) the models are drawn from P and in (b) the models are drawn from $\mathcal{R}$.



(a)



(b)

Figure 4.2. The elements from the set of true calibration models of Figure (4.1) overlayed amongst each other over $\mathcal{X}$. (a) shows elements from $\mathcal{P}$ and (b) shows elements from $\mathcal{R}$.

which also represents the largest variations in the estimated correlation $\hat{R}_{h_P}$ relative to estimated correlation at $x = 0$.

For the initial calibrations, the target calibration model, denoted by h_k, is drawn from the sets defined by $\mathcal{H}_P = \{h_{P_2}, h_{P_8}, h_{P_{10}}, h_{P_{14}}\}$ and by $\mathcal{H}_R = \{h_{R_1}, h_{R_2}, h_{R_4}, h_{R_8}\}$, whose elements are shown in Figure (4.2).

For the calibration transfer simulations, the target calibration model, denoted by h_l, is drawn from the sets defined $\mathcal{H}_P = \{h_{P_8}, h_{P_{10}}, h_{P_{14}}\}$ and by $\mathcal{H}_R = \{h_{R_2}, h_{R_4}, h_{R_8}\}$ whose elements are shown in Figure (4.2). It should be noted that the elements h_{P_2} and h_{R_1} are not part of the set of target calibration models. For the calibration transfer simulations the initial calibration model is defined as $h_k = h_{P_2}$ when using models from $\mathcal{P}$ and as $h_k = h_{R_1}$ when using models from $\mathcal{R}$.

The calibration transfers are therefore performed from $\hat{h}_k$ to $\hat{h}_l$ for $k = \bar{1}$ and $l = \{8, 10, 14\}$ when the calibration model h is drawn from $\mathcal{H}_P$. For calibration transfers using models from $\mathcal{H}_R$, $k = 1$ and $l = \{2, 4, 8\}$. As l increases, the *similarity* between h_k to h_l, at any x_i, decreases, as given by the correlation measure.

4.2.4 Selecting the number, coordinates, and quality of the calibration data

The calibration data values used for calibration are chosen to be noise free to eliminate the possibility of error in the data values contributing to the estimation error e_ϵ in the approximation of h. In other words, the noise in the data values is eliminated to prevent the noise from potentially masking the inherent error behaviour of the calibration transfer methods. Alternatively, the calibration slope values consist of both error free values, that is the true slope values, and slope values having an error due to inaccuracies in the slope estimation based on using piecewise linear and cubic spline interpolation of the calibration data.

To expand on the reason for eliminating the error in the data values, consider that the generalisation error e_g consists of two components, the approximation error

e_a, that is due to the FFNN's inability to precisely approximate h given a finite number of neurons, and the estimation error ϵ_e that results from the inablility to find the *best* FFNN approximation given a finite data set and the use of a non-optimal search process. Therefore, with a fixed set of neurons, sample points, and learning parameters, the observed changes in the generalisation error in approximating h are due entirely to changes in ϵ_e. These changes in ϵ_e are then due entirely to the learning algorithm finding different acceptable approximations of the calibration data given different initial weight values.

Adding error to the values of the data points effectively adds another component to ϵ_e, thereby potentially masking the behaviour exhibited by the learning algorithm. In other words, the question is raised as to whether the observed generalisation error is due to the learning algorithm or to the error added to the data values. The issue of error in the calibration data and slope values is discussed further in chapter 6.

A common calibration strategy is used to select the location of the calibration data points. The points are placed to simply partition $\mathcal{X}$ into equal intervals. It is well known, as discussed in Appendix A, that this strategy may not provide an optimum placement, in that it may influence the potential for $\hat{h}$ to approach $\hat{h}^*$.

The desire in instrument calibration to avoid using significant number of samples needs to be considered when selecting the number of points to use in the simulations. Given that only one dimensional inputs are used, it seems reasonable, for these simulatons, to limit the number of samples to a maximum of sixteen and a minimum of four. It is also apparent that for the calibration models selected for these simulations, the use of sixteen samples provides, to within a given e_e, a potential for $\hat{h}$ to approach $\hat{h}^*$ that is greater than the potential provided by using only four samples.

The specific calibration data values used for the initial calibration simulations are obtained by sampling $h_k(x)$ at N_k points evenly spaced over X. When h_k is member of $\mathcal{H}_P$, $N_k = \{4, 5, 6, 8\}$ and $X = [0, 1]$. When h_k is member of $\mathcal{H}_R$, $N_k = \{5, 6, 8, 10\}$ and $X = [0, 15]$. The specific set of coordinates used for the initial

calibration points are given by $X_k = \{x_i = \Delta X \frac{i}{(N_k-1)} : i = 0, 1, 2, \ldots, N_k - 1\}$, where $\Delta X = \max(X) - \min(X)$.

The various sets of data used in the initial calibration simulations are described by the following definitions. The set of data values is defined as $D_k = D_k^{(0)} = \{(x_i, h_k(x_i)) : i = 1, 2, \ldots, N_k, x_i \in X_k \subset X\}$. The set of slope values used in the $D^{(1)}$ learning algorithms is defined as $D_k^{(1)} = \{(x_i, d\tilde{h}_k(x)/dx\Big|_{x=x_i}) : i = 1, 2, \ldots, N_k, x_i \in X_k \subset X\}$, where, $d\tilde{h}_k(x)/dx$ represents the slope estimate of h_k at x_i obtained using the methods described in section 4.2.5.

Therefore, for the initial calibration simulations the calibration data sets are defined by $G_k = D_k \cup D_k^{(1)}$ for $D^{(1)}$ learning algorithms and by D_k for the BP and $D^{(0)}$ learning algorithms.

For the calibration transfer simulations, the calibration data values used for the simulations are obtained by sampling $h_l(x)$ at $N_l = \{4, 5, 6, 8\}$ points that are evenly spaced over X. When h_l is member of $\mathcal{H}_P$, $X = [0, 1]$ and when h_l is member of $\mathcal{H}_R$, $X = [0, 15]$. The specific set of coordinates used for the lth calibration data set are given by $X_l = \{x_i = \Delta X \frac{i}{(N_l-1)} : i = 0, 1, 2, \ldots, N_l - 1\}$, where $\Delta X = \max(X) - \min(X)$.

The various sets of data used in the calibration transfer simulations are described by the following definitions. The set of data values is defined as $D_l = D_l^{(0)} = \{(x_i, h_l(x_i)) : i = 1, 2, \ldots, N_l, x_i \in X_l \subset X\}$. The iDACX transfer methods use the data set defined as $S_l = \{(x_i, h_l(x_i) - \hat{h}_k(x_i)) : i = 1, 2, \ldots, N_l, x_i \in X_l \subset X\}$, which consists of using the vector defined by the points $h_l(x_i)$ and $\hat{h}_k(x_i)$. The set of slope values used in the $D^{(1)}$-iDACX methods is defined as $D_l^{(1)} = \{(x_i, d\tilde{h}_l(x)/dx\Big|_{x=x_i}) : i = 1, 2, \ldots, N_l, x_i \in X_l \subset X\}$, where, depending on the transfer method used, $\tilde{h}_l(x)$ represents either the approximation of h_l obtained using $\hat{h}_k + \hat{s}_l$ or the true model h_l. It should be noted that in using $\hat{h}_k + \hat{s}_l$, the data set S_l is used to obtain $\hat{s}_l$.

Therefore, the calibration data sets defined for the calibration transfer simulations are given by $G_l = D_l \cup D_l^{(1)}$ for the $D^{(1)}$-iDACX transfer methods, by D_l for the PICX

and Std. Cal. methods, and by S_l for the iDACX methods.

4.2.5 Selecting the slope estimates

It is clear from the discussions in chapter 3 that the accuracy of the slope estimates will influence the $D^{(1)}$ learning algorithm's generalisation performance. Given that there is a lack of published results precisely defining how the slope error influences the generalisation error, the simulations will attempt calibration transfers using slope estimates having various types of error. These error types will range from using an error free or true slope estimate, to estimates based on either a piecewise linear or a spline approximation of s_l or h_k.

The initial calibration simulations use slope estimates based on either the piecewise linear or spline approximation of h_k using the data D_k or the true slope values of h_k at $x_i \in X_k$. The calibration transfer simulations use slope estimates based on either the piecewise linear or spline approximation of s_l using the data S_l or the true slope values of h_l at $x_i \in X_l$. It is clear that the degree of error in these slope estimate will vary with the number of calibration points used for the approximation.

The slope estimates based on the piecewise linear approximation of s_l or h_k technically do not have a defined slope value at the calibration points, that is, at the break points or knots. In this case, the slope at the knot is taken to be average of the slopes of the line segments on either side of the knot. In the case of calibration points on the boundaries of $\mathcal{X}$, or the end points, the slope value is taken to be equal to the slope of the line segment connecting the boundary point to the next adjacent point.

The slope estimates based on the cubic spline approximation are obtained using the slopes from a cubic spline approximation of s_l or h_k. The estimated slope at a calibration point is obtained from the slope of the line formed by two *closely* spaced points on the cubic spline approximation that are centered about the calibration point. The term closely is taken to be a value of approximately 0.5% of the interval

spanned by $\mathcal{X}$ over the appropriate dimension.

4.3 FFNN Parameter Definition

4.3.1 Selecting the number of hidden neurons

Determining the number of hidden neurons to use inorder to obtain approximations with acceptable generalisation error, or calibration error, is in itself a non-trival problem and will not be explored in the proposed simulations. Instead, for the purpose of the proposed simulations, it becomes more important to observe how the error behaviour of the calibration transfer methods is affected as the number of hidden neurons is varied.

The problem then becomes one of selecting the range over which to vary the number of hidden neurons so as to observe changes in the error behaviour of the calibration transfer methods. In this regard, the choice of using a single input dimension helps in determining this range. Since the calibration model h is univariate and known over all $\mathcal{X}$, it becomes a simple matter to plot $h(x)$ over x and to then inspect the functional form of h. This inspection then allows a visually decompositon of $h(x)$ into regions of piecewise segments of sigmoidal functions. Though, the process is highly subjective and crude, it does provide a reasonable value for the lower limit on the number of hidden neurons needed to at least begin to reflect the functional form $h(x)$ over most of its domain.

It should be noted that this visually decomposition of $h(x)$ can be placed in the context of an analytical approach proposed by Huang and Babri [145]. They showed that given N points and $N-1$ hidden neurons in a single hidden layer FFNN, that not only is it possible to precisely interpolate the N points but that it is also possible to directly determine, without learning, the value of the weights needed to provide the needed interpolation of the data points. Visual decomposition of $h(x)$ can be viewed as a non-analytical application of these results, in that the N points are mentally placed at appropriate points on $h(x)$ and one visually places $N-1$ neurons so as to

span the $N-1$ regions.

For example, consider Figure (4.2) showing instances of both sets of $h(x)$, that is, $\mathcal{H}_P$ and $\mathcal{H}_R$. A visual inspection of these sets reveals that four or five neurons will be needed to provide an approximation that begins to follow the functional form of $h(x)$ over its domain.

Increasing the number of neurons above this lower value provides the FFNN with the potential to obtain approximations with generalisation errors that decrease in proportion to the number of hidden neurons in a single layer FFNN [9, 47]. In other words, increasing the number of hidden neurons by a factor of two potentially allows the FFNN to achieve a two-fold decrease in the generalisation error.

The initial calibration model $\hat{h}$ is implemented using $N_n = \{5, 8, 10, 16\}$ hidden neurons to approximate the calibration models in $\mathcal{H}_P$ and by $N_n = \{5, 8, 10\}$ hidden neurons to approximate the calibration models in $\mathcal{H}_R$.

In general, for the calibration transfer simulations, the calibration model $\hat{h}_l$ consists of the same number of hidden neurons as that used to implement $\hat{h}_k$. Again, to limit the number of simulations, the calibration transfers are restricted to calibration models having $N_n = \{5, 8, 10\}$ hidden neurons when $\hat{h}_k$ represents the approximations of members in $\mathcal{H}_P$. When $\hat{h}_k$ represents the approximations of members in $\mathcal{H}_R$, only models using $N_n = \{5, 8\}$ hidden neurons are used.

4.3.2 Weight initialisation for initial calibrations

The initial value of the FFNN weights for the initial calibration simulations are, in the case of using the BP learning algorithm, randomised using the Nguyen-Widrow method [140] or a modified version of the Nguyen-Widrow method. Briefly, the modified method simply ensures that the weights for n hidden neurons are set so as to have their activation functions effectively span, in a *random* manner, one of m non-overlapping regions in $\mathcal{X}$. A total of $N_n = (n)(m)$ hidden neurons are then needed to cover $\mathcal{X}$ and $\mathcal{Y}$. To identify this method, the notation ORm-n is

used. The Nguyen-Widrow method only insures that the activation functions of the $(n)(m)$ hidden neurons *randomly* span $\mathcal{X}$ and $\mathcal{Y}$. Here, random is taken to mean a normal distribution of values with appropriatedly set means and variances to allow the spanning of $\mathcal{X}$ and $\mathcal{Y}$. The $D^{(n)}$ learning algorithms uses weights that are initialised using only the modified version of the Nguyen-Widrow method.

It should be noted that since the resulting FFNN approximation, $\hat{h}_k$, is dependent on the randomised values of the initial weights, the weights will be initialised to new random values a number of times, thus providing a number of different approximations of the initial calibration models $\hat{h}_k$.

4.3.3 Weight initialisation for calibration transfers

The calibration transfer simulations set the initial value of the weights to the values that provide the approximation $\hat{h}_k$ which is determined by the initial calibration simulations. Because of the large quantity of initial calibration models to chose from, the calibration transfer simulations are limited to selecting models, $\hat{h}_k$, obtained using $N_k = 8$ calibration points.

It should be noted that the PICN BP method only draws $\hat{h}_k$ from initial calibrations obtained using BP learning. The PICN $D^{(0)}$, iDACN, and $D^{(1)}$-iDACN methods draw the same $\hat{h}_k$ from initial calibrations obtained using any $D^{(n)}$ learning algorithm.

The Std. Cal. BP method represents a conventional learning trial using backpropagation to approximate the data D_l. Technically, there is no initial calibration model from which to begin learning. Therefore, the FFNN weights for the Std. Cal. BP method are initialised using the Nguyen-Widrow method.

4.3.4 Selecting the learning termination criteria

The termination of the learning algorithm for the initial calibration simulations is set to occur whenever the total number of epochs exceeds 3×10^5 or when the SSE between the target and desired values for either the data or slope falls below 1×10^{-10}.

For the calibration transfer simulations learning is terminated when the total number of epochs exceeds 1.2×10^5 or when the SSE between the target and desired values for either the data or slope falls below 1×10^{-8}. In addition, termination of learning in either simulations occurs whenever either the total SSE begins to increase, indicating an unstable gradient descent, or when the total SSE changes less than 0.1% over 3000 epochs of learning.

In some instances the learning algorithms fail to achieve the specified SSE goals. In these cases, the results of the simulation trials using data obtained from $\mathcal{H}_P$ are excluded from the results if, after termination of learning, the SSE in the approximation of the data values is greater than 0.03 for the BP and $D^{(0)}$ learning methods. For the $D^{(1)}$ learning methods, the simulation results are excluded if the SSE in the approximation of the data values is greater than 0.03 or if the SSE in the approximation of the slope values is greater than 0.25. For the simulation trials using data obtained from $\mathcal{H}_R$, exclusion of the results occurs when SSE in the approximation of the data and slope values is greater than 0.06 and 1.25, respectively. These criteria for excluding results were obtained by simply noting that most of the approximation of h_l or h_k had, subjectively, generalisation errors that were not *excessively* large. It should also be noted that values used for these criteria are not critical to the resulting analysis.

Though the number of epochs appears to be excessive and the SSE threshold very small, they have been selected to allow, or at least attempt to allow, the learning algorithm to achieve a true local minimum on the error surface $C(\mathbf{w}, D)$, where $D = \{D_k, D_l, G_k, G_l\}$ represents the possible data set used for learning. These termination criteria for learning will potentially allow the learning behaviour known as *over-learning* to occur. Over-learning results when the generalisation error ϵ_g begins to increase in spite of a decrease in the SSE measured between the desired data and FFNN's approximation of the data. Again, the consequence of over-learning on the resulting analysis is discussed in Chapter 6, but it should be noted that the effect of over-learning is not crucial in influencing the analysis.

4.3.5 Selecting and adjusting the learning rate parameters

The learning rate parameters are determined by trial and error, in that a number of trials are performed and the set of learning rate parameters that provide reasonably consistent and acceptable results are used for the remaining simulations.

The difficulty in using the trial and error approach for $D^{(1)}$ learning is that there are a total of seven learning rate parameters, or step sizes, that can be varied. Varying each parameter individually to determine an acceptable learning rate can easily result in a significant number of trials. For example using only three values for each learning rate parameter results in $3^7 = 2187$ combinations that would have to be repeated given a change in either the number of neurons, the number of calibration data points, or the true calibration model h. Again, this represents a significant number of trials. Instead, as to be discussed in chaper 6, careful inspection of the learning process using a limited number of trials with different values of learning rate parameters, revealed a range of values that appeared to provide reasonably consistent and acceptable results.

4.4 Calibration Error Analysis

4.4.1 Estimating the calibration error

The effectiveness of the calibration transfer methods is indicated by the magnitude of the calibration error e. For consistency, e^2 will be used as the measure of the calibration error and is defined as

$$e^2 = \hat{e}_g = \frac{1}{N_j} \sum_{j=1}^{N_j} \left(h(x_j) - \hat{h}(x_j) \right)^2 . \qquad (x_j, h(x_j)) \in T_j, N_j = \bar{\bar{T}}_j . \qquad (4.3)$$

where T_j is the test or validation data set and $\hat{e}_g$ is referred to as the empirical generalisation error or standard error of prediction, SEP.

The value of N_j is taken to be approximately 200, that is, 200 points over X, for $\mathcal{H}_P$ and approximately 130 when using $\mathcal{H}_R$. The use of these values of N_j allows an acceptable estimate of the generalisation error. In addition, the use of this number

of sample points to estimate the generalisation error does not overly burden the simulations with excessive computations.

4.4.2 Comparing calibration errors

The resulting calibration error, as measured by $\hat{e}_g$, from each simulation can essentially be viewed as a random variable, in that $\hat{e}_g$ is dependent on the initial weight values used during the initial calibration. Examining a single occurrance of the random variable $\hat{e}_g$ is not of much use; therefore, a set of $\hat{e}_g$ will typically be considered. Each set will consist of anywhere from 4 to 50 independent occurrances of $\hat{e}_g$. These sets will be referred to as a group of simulations, where the group membership is determined by satisfying some pre-selected set of calibration parameters. For example, a simulation group may be defined as all simulations with $N_k = 6$, using 5 hidden neurons, attempting to approximate the data and slope, that is, $D^{(1)}$ learning, of $\mathcal{H}_P$, and using random initial weight values.

The random nature of $\hat{e}_g$ introduces a degree of difficulty in determining whether a particular calibration method, when compared to another method, improves the calibration error for a fixed number of calibration samples or whether for a fixed calibration error less calibration samples can be used. The essential difficulty is that of establishing whether a difference between $\hat{e}_g$ from two groups of simulations actually exists and whether that difference is significant.

The difficulty in determining whether a differences in $\hat{e}_g$ exists and whether it is significant centers on evidence suggesting that the distribution of $\hat{e}_g$ cannot be assumed to follow a normal distribution and is more likely to be multimodal [146]. In other words, reporting the mean and standard deviation of $\hat{e}_g$ for the various calibration transfer methods and then comparing these statistics **may not** reveal any meaningful results.

These difficulties do not prevent the simulations from providing useful results. Instead, it simple means that to compare $\hat{e}_g$ obtained from different groups of simu-

lations requires the use of less powerful measures. In this context, the median values will be used to measure the middle value of a group of simulations and percentiles will be used to provide an indication of the variability of $\dot{e}_g$ within a group.

Standard box plots are used to graphically show the median and 25th and 75th percentile values. Whiskers extending from the ends of the boxes will indicate the nearest data point to the location defined as being 1.5 times the interquartile range from the end of the box. Data points beyond the whiskers are shown as outlier points. The median is indicated by an appropriate symbol within the box.

In addition, a nonparametric statistical test referred to as the Kruskal-Wallis one-way analysis of variance by ranks [147] will be used. This test, when given two or more groups of samples, is used to determine if at least one of the groups represents a distribution with a **median** value different from the medians of the other groups. If the test statistic provided by the Kruskal-Wallis test is greater than the chi-squared value at a particular significance level, $0 < S < 1$, then the test indicates that at least one of the groups has a median which is different from one or more of the other medians with a significance given by S. The test can then be applied pairwise to any two of the groups to determine which of the groups has a median that is significantly different.

In applying the test pairwise to determine which pairs have significant differences in their medians, care must be taken to inadvertently prevent increasing the likelihood of a type I error, that is to declare that there is a significant difference between medians when in fact there is no difference. For example, assume that the type I error level, or the significance level S, is set to 0.05, indicating that it is desired to have less than one chance in twenty of committing a type I error. If k pairwise comparisons are made, then there is approximately a k times 0.05 chance of committing a type I error. This approximation is only accurate for small values of k and S, that is the chance of committing a type one error is actually given by $1 - (1 - S)^k$ so that the adjustment to S is actual given by the factor $1 - \sqrt[k]{1 - S}$ ([147], pg. 340).

To accommodate this problem, the type I error level is adjusted according to the Bonferroni-Dunn method [147] which essentially divides S by the number of comparisons that are to be made. For all the simulations the significance levels for the type I errors are set to $S = \{0.1, 0.05, 0.01, 0.001\}$, which then require to be divided by the number of comparisons. It should be noted that, for these simulations, simply dividing S by k to obtain the adjusted values for S provides a reasonably accurate approximation to the actual adjustment required. If the test statistic, p, returned by the pairwise test is less than or equal to this adjusted significance level, then the pairs have medians that are significantly different. In all these tests the *nondirectional* hypothesis is used, that is, the alternative hypothesis H_1 is that the medians are not equal.

For these simulations the only pairwise comparisons between $\hat{\epsilon}_g$ medians that will be made are those comparing medians from each of the calibration transfer methods with the median associated with the Std. Cal. BP method. A Comparison of the medians associated with the various calibration transfer methods with one another will not be done. For the initial calibrations simulations, the medians associated with each of the $D^{(n)}$ learning algorithms and the median associated with the BP learning algorithm will be compared. Again, comparisons between the medians associated with the various forms of $D^{(n)}$ learning algorithms and the medians of other $D^{(n)}$ learning algorithms will not be done.

The application of the Kruskal-Wallis test requires the following assumptions to be satisfied:

1. Each sample is randomly drawn, that is, each simulation result is random.

2. Each sample is independent of any other sample.

3. The dependent variable, that is, $\hat{\epsilon}_g$ in this case, should be a continuous random variable.

4. The distributions of the groups are the same, though it is not necessary for the

distribution to be normal. It should be noted, as discussed in Sheskin ([147], pg. 397-398), that there is empirical evidence to suggest that the test is not affected by violation of this assumption.

By virtue that each calibration or calibration transfer begins by using a set of weight values that are independently and randomly determined, assumption one and two are met. In addition, assumption three is met since the random variable $\hat{\epsilon}_g$ is continuous. Though it is not known if the distribution of $\hat{\epsilon}_g$ is the same for each group of simulations, the evidence suggesting that the test has a degree of invariance to these differences provides confidence in its applicability for these simulations. Therefore, given that the distribution of $\hat{\epsilon}_g$ from each group of simulations is likely not normal, the Kruskal-Wallis one-way analysis of variance by ranks appears to be an appropriate test.

Given that the medians are significantly different, the ratio of the median error of the Std. Cal. BP method to median error of the calibration transfer methods is defined as the improvement factor I_F. In other words, $I_F > 1$ indicates that the calibration transfer method reduces the median of the calibration error by a factor given by I_F. The improvement factor for the initial calibrations is defined as the ratio of the median error in BP learning to median error of one of the other learning algorithms. Obviously, $I_F < 1$ indicates that the calibration transfer methods or the particular learning algorithm have median errors greater than that provided by the Std. Cal. BP method or the BP learning algorithm.

5. Simulation Results

5.1 Overview

This chapter presents the results of simulations that estimate the calibration error $\hat{e}_g$ of a hypothetical FFNN based instrument after it has been initially calibrated to obtain $\hat{h}_k$ and after it has under gone a calibration transfer from $\hat{h}_k$ to $\hat{h}_l$.

The purpose of the simulations is to show that the proposed methods of calibration transfer achieve the objective of the thesis:

- For a given number of calibration samples, an improvement in the calibration accuracy, as compared to that achievable in a standard recalibration employing conventional backpropagation learning.

The achievement of this objective is tested under various calibration conditions that include, varying the number neurons in the FFNN, using different number of calibration points in D_l, varying the *similarity* between h_k and h_l, and using two different classes of functions for the calibration model h.

The results of this chapter show that under most of the calibration conditions simulated, the calibration error from at least one of the PICN, iDACN, or $D^{(1)}$-iDACN methods achieves the objectives of the thesis. In addition, the results show that the $D^{(1)}$ learning algorithm achieves a median $\hat{e}_g$ that is less than the median $\hat{e}_g$ achieved with the BP learning algorithm.

The chapter first presents the summary of the initial calibration simulation results and then the calibration transfer simulation results.

5.2 Initial Calibration Simulation Results

Figure (5.1) through Figure (5.9) and the accompanying tables, summarises the results of the initial calibration simulations as performed according to the methods described in chapter 4.

The figures are organised to show for each learning algorithm the estimated calibration error, defined using $\hat{e}_g$, as a function of the calibration points, N_k, given a particular number of hidden neurons, N_n. For any specific number of calibration points, number of neurons, and type of learning, the error $\hat{e}_g$ of a series of simulation trials are summarised as a single box plot item that is drawn in the figure. The number of trials used to obtain the specific form of the box plot item is listed directly below the corresponding item.

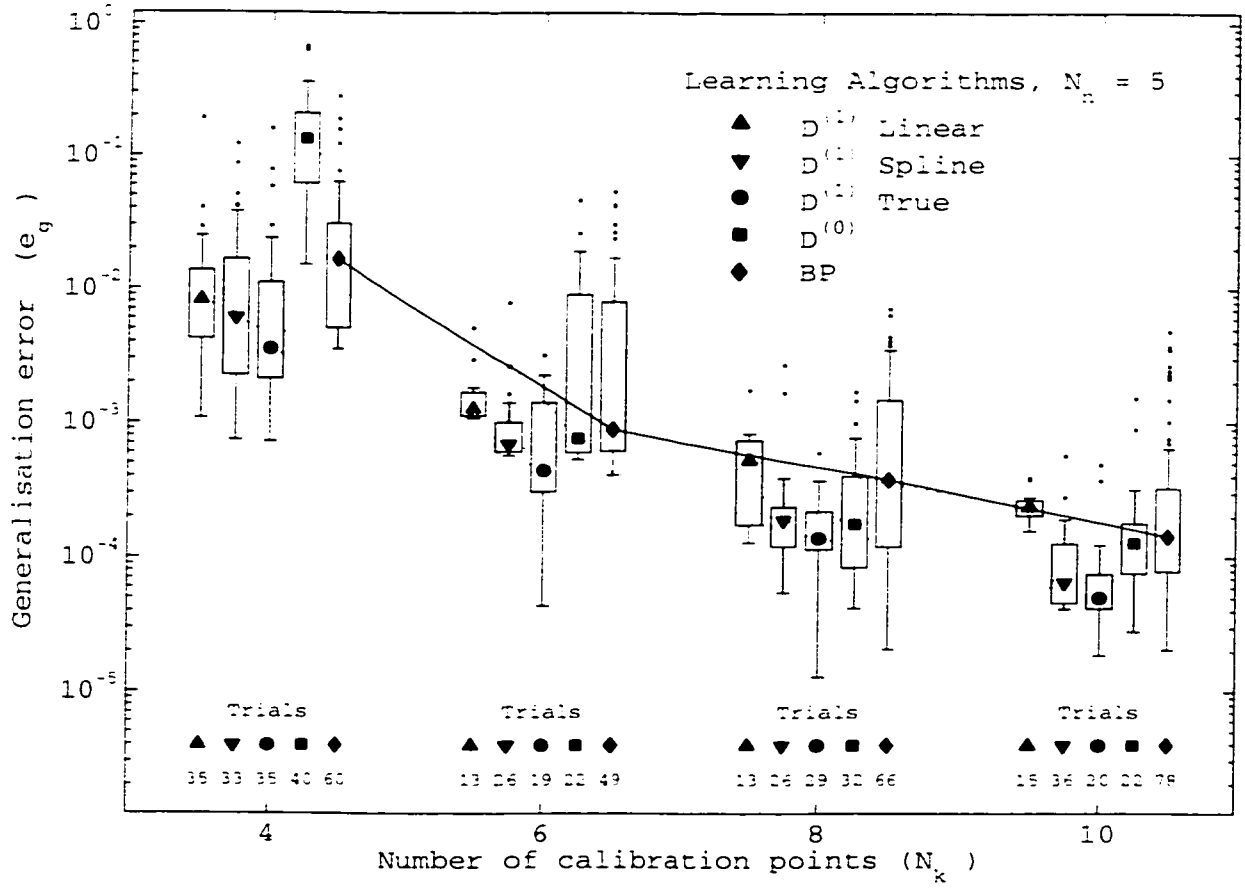
Each box plot item summarises the initial calibration simulation trials for all the members of either $\mathcal{H}_P$ or $\mathcal{H}_R$ as indicated in the corresponding figure caption. For example, the box plot item associated with the $D^{(1)}$ learning algorithm, using a FFNN with $N_n = 5$ hidden neurons, and $N_k = 4$ calibration points, shown in Figure (5.1), summarises the error $\hat{e}_g$ from 35 initial calibration simulation trials. These simulation trials include using h_{P_7}, h_{P_8}, $h_{P_{10}}$, and $h_{P_{14}}$ as the target calibration model.

To help visualise the trend of $\hat{e}_g$ as a function of N_k for any particular learning algorithm, guide lines connecting the error medians at each value of N_k are drawn. The guide line used for the backpropagation, BP, learning algorithm is drawn as a solid line as opposed to the dotted lines used for the other learning algorithms.

Below each figure is a table listing the level of significance, S, the p values, and the improvement factor, I_F, associated with the difference between the median error in BP learning and the median errors in $D^{(n)}$ learning at each value of N_k. For the initial calibration simulations, the significance levels are adjusted for four comparisons using the Bonferroni-Dunn method [146]. It should be noted that only results achieving a p value less than or equal to the adjusted significance level 0.1 are listed in the table.

In other words, unlisted simulation results did not achieve median errors that are significantly different than median error in BP learning.

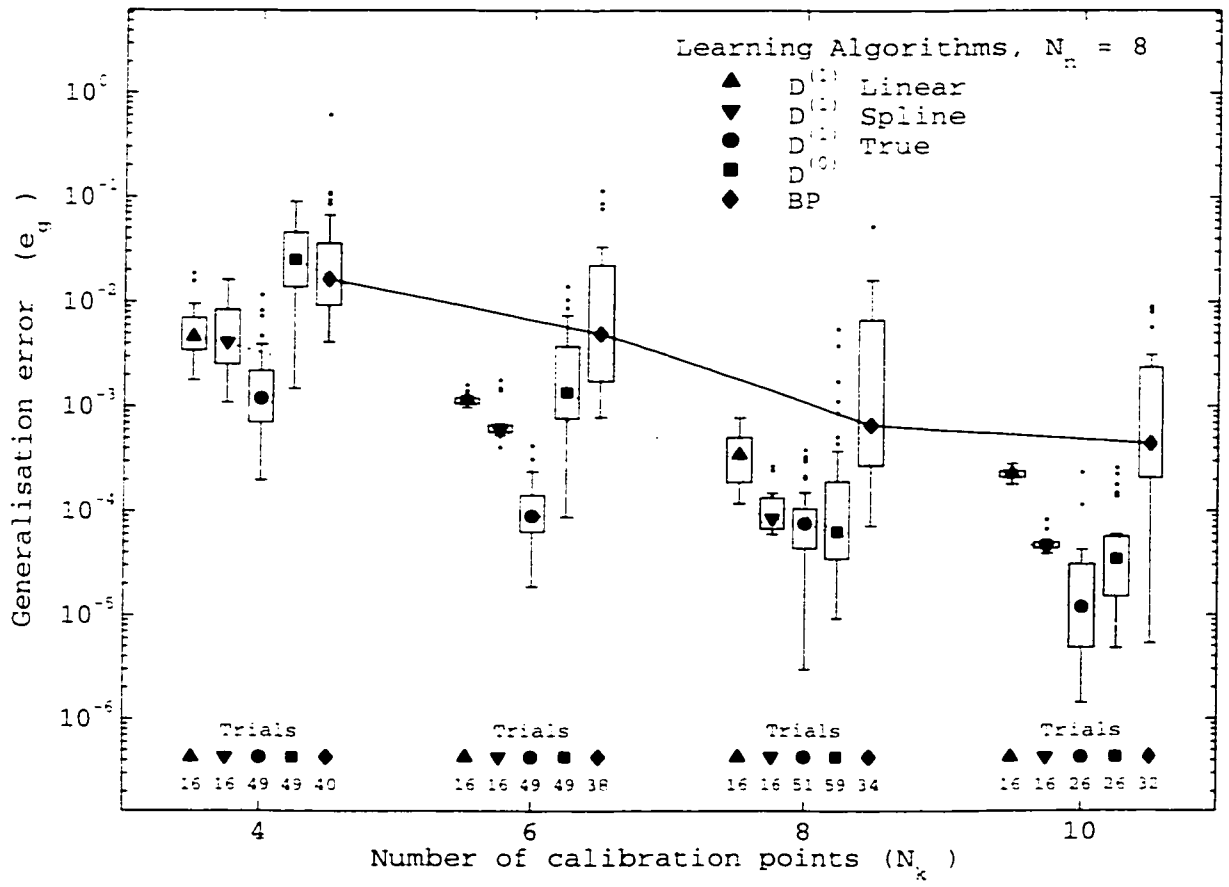
Figure (5.10) through to Figure (5.13) summarises the initial calibration simulations, not as a function of the number of calibration points, but as a function of the number of hidden neurons, N_n for any given number of calibration points.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear												
$D^{(1)}$ Spline	-	0.018	2.8							***	0.00018	2.2
$D^{(1)}$ True	**	0.00047	4.7	**	0.0014	2.0	**	0.00057	2.7	***	< 0.0001	2.9
$D^{(0)}$	***	< 0.0001	0.124									

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method.

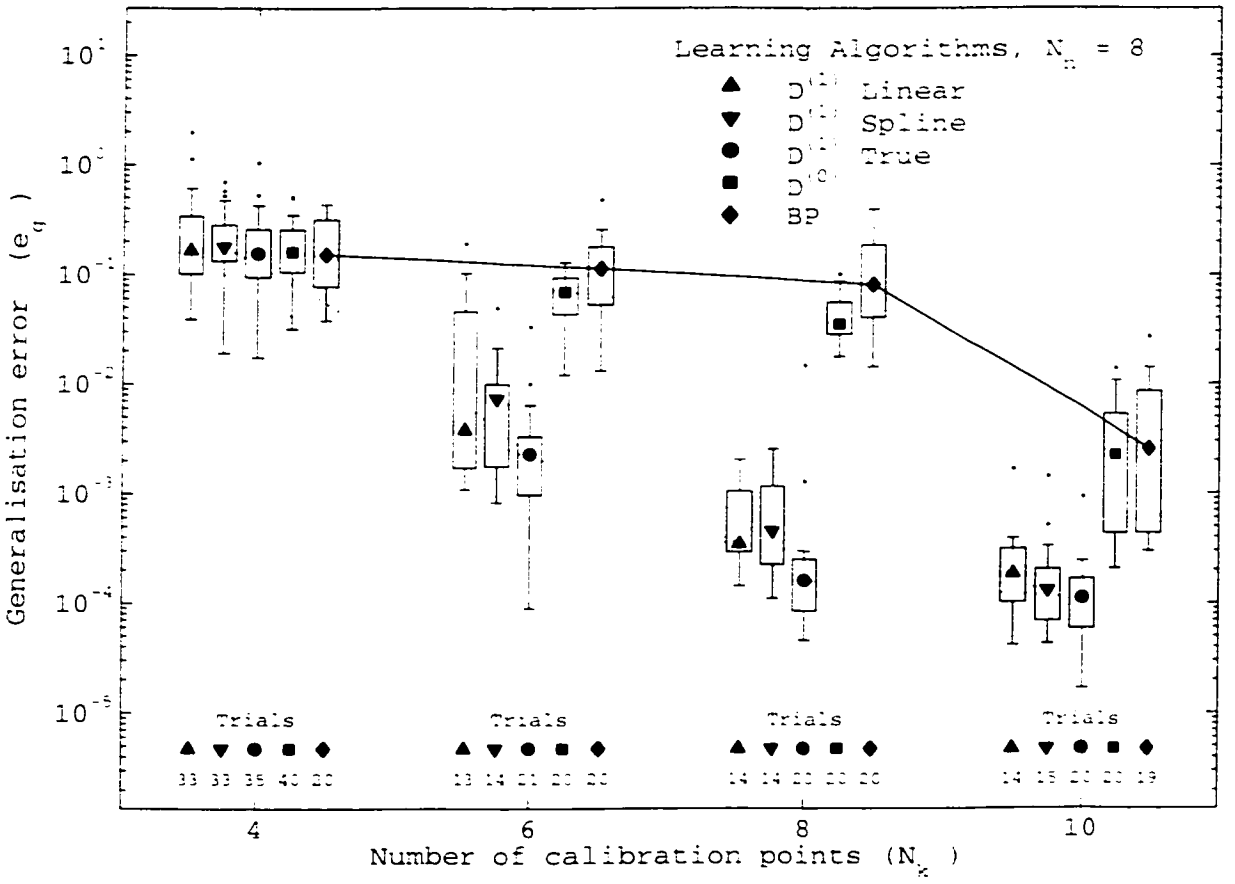
Figure 5.1. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (•) < 0.1, (••) < 0.05, (•••) < 0.01, (••••) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear	••	0.0015	3.6	•	0.0044	4.3						
$D^{(2)}$ Spline	••	0.00051	4.0	•••	< 0.0001	8.4	•••	< 0.0001	7.9	••	0.00042	9.7
$D^{(3)}$ True	•••	< 0.0001	13.8	•••	< 0.0001	55	•••	< 0.0001	8.8	•••	< 0.0001	37
$D^{(0)}$				+	0.0027	3.6	•••	< 0.0001	10.5	•••	< 0.0001	12.7

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method

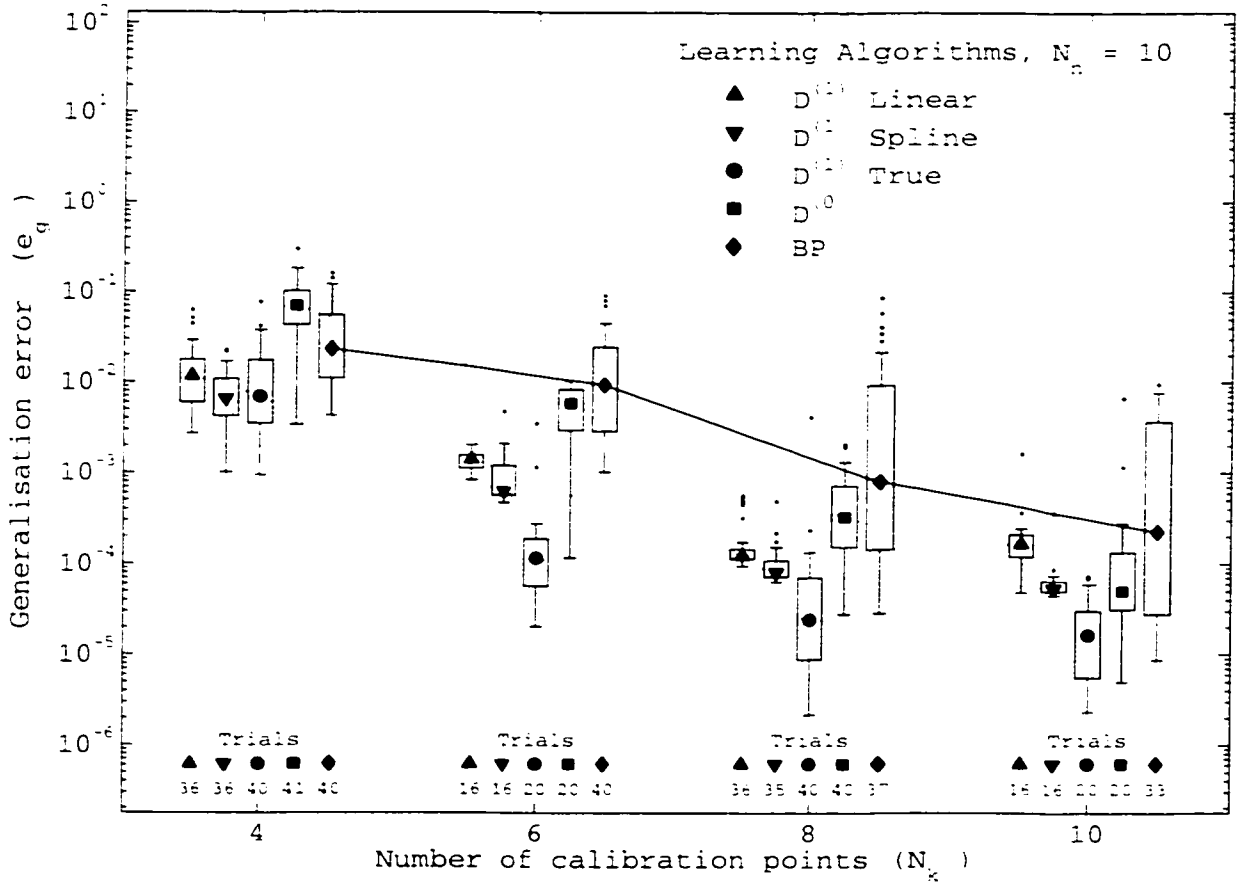
Figure 5.2. The initial calibration errors, as measured by $\dot{e}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = (; Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001)											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear				**	0.00039	30	***	< 0.0001	240	***	0.00014	13.7
$D^{(1)}$ Spline				***	< 0.0001	15.5	***	< 0.0001	180	***	< 0.0001	20
$D^{(1)}$ True				***	< 0.0001	49	***	< 0.0001	510	***	< 0.0001	23
$D^{(2)}$												

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method

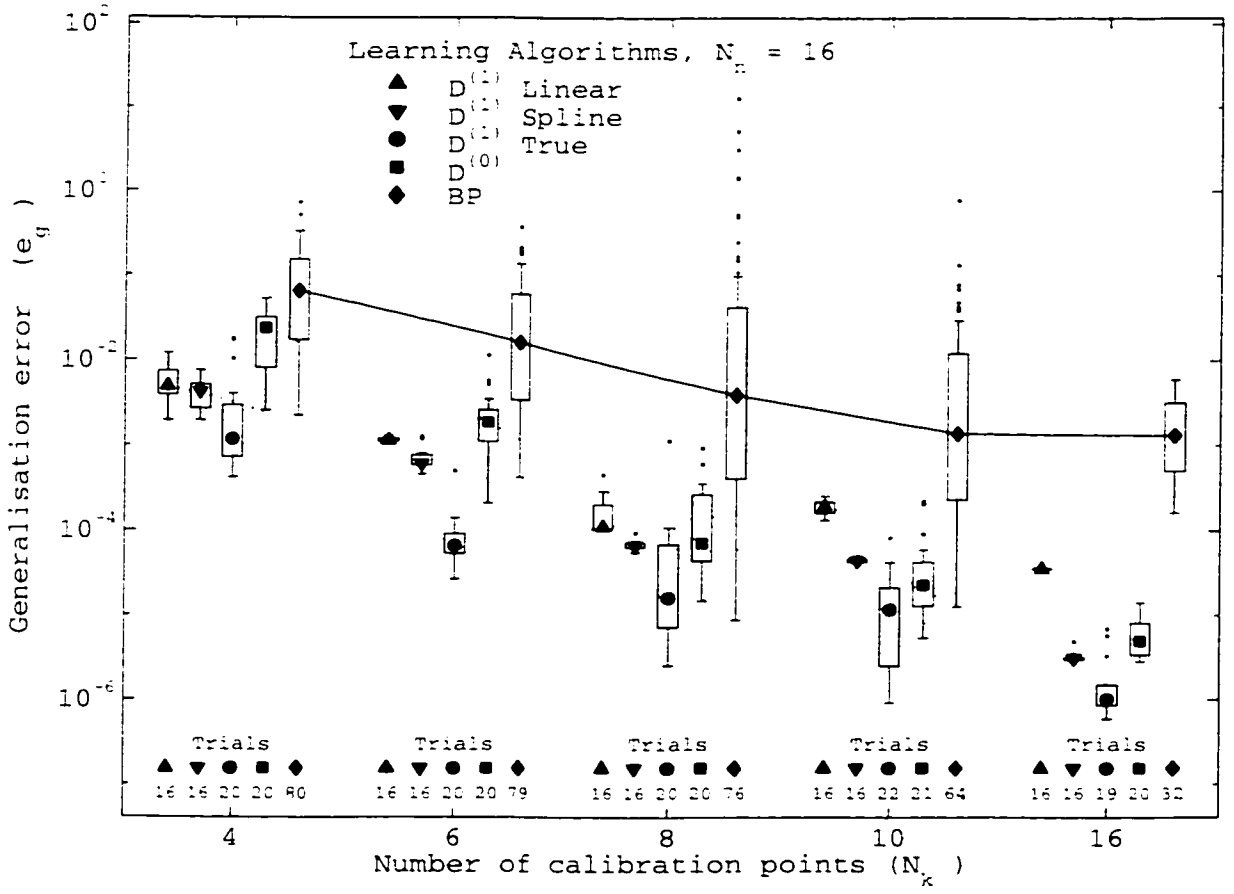
Figure 5.3. The initial calibration errors, as measured by $\dot{e}_q$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons with weights initialised using the OR8-1 method.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_k = 4$			at $N_k = 6$			at $N_k = 8$			at $N_k = 10$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(-1)}$ Linear	**	0.0012	1.98	***	< 0.0001	6.7	*	0.011	6.5			
$D^{(-1)}$ Spline	***	< 0.0001	3.6	***	< 0.0001	15.0	***	< 0.0001	9.9			
$D^{(-1)}$ True	***	< 0.0001	3.4	***	< 0.0001	81	***	< 0.0001	33	***	< 0.0001	13.7
$D^{(0)}$	*	0.0044	0.33									

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method.

Figure 5.4. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 10$ hidden neurons.

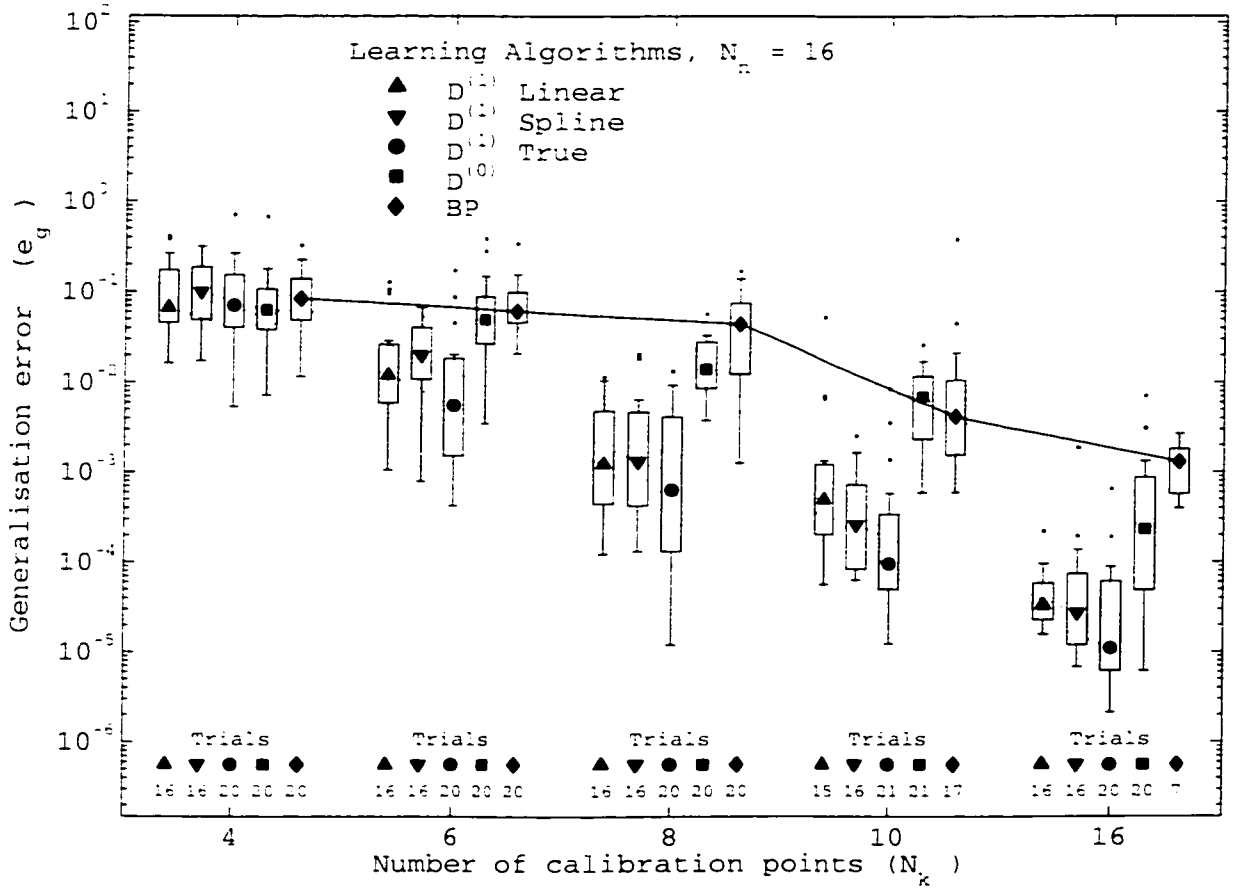


Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear	***	< 0.0001	13.3	***	< 0.0001	14.0	**	0.00032	35			
$D^{(1)}$ Spline	***	< 0.0001	15.3	***	< 0.0001	27	***	< 0.0001	59	***	< 0.0001	32
$D^{(1)}$ True	***	< 0.0001	56	***	< 0.0001	240	***	< 0.0001	240	***	< 0.0001	120
$D^{(0)}$	*	0.0124	2.8	**	0.00035	8.6	***	< 0.0001	55	***	< 0.0001	61

	at N_k = 16		
	S	p values	I_F
$D^{(1)}$ Linear	*	0.0087	37
$D^{(1)}$ Spline	***	< 0.0001	430
$D^{(1)}$ True	***	< 0.0001	1300
$D^{(0)}$	***	< 0.0001	270

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method

Figure 5.5. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 16$ hidden neurons.

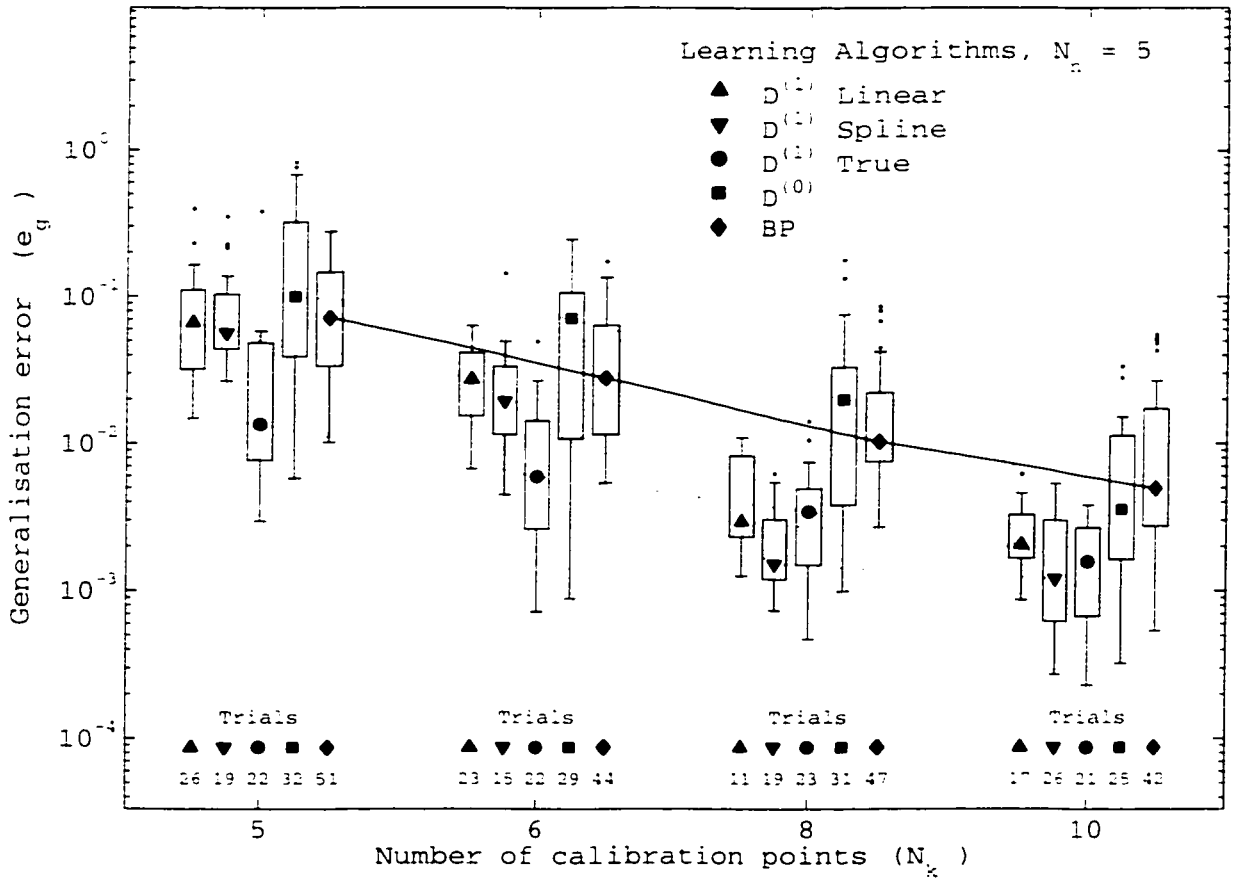


Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (·) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(-1)}$ Linear				***	0.00019	5.1	***	< 0.0001	36	*	0.0061	8.4
$D^{(-1)}$ Spline				**	0.00063	3.1	***	< 0.0001	33	***	< 0.0001	16.0
$D^{(-1)}$ True				***	< 0.0001	11.2	***	< 0.0001	69	***	< 0.0001	44
$D^{(0)}$												

	at N_k = 16		
	S	p values	I_F
$D^{(-1)}$ Linear	**	0.00077	39
$D^{(-1)}$ Spline	**	0.00041	48
$D^{(-1)}$ True	***	< 0.0001	118
$D^{(0)}$			

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method.

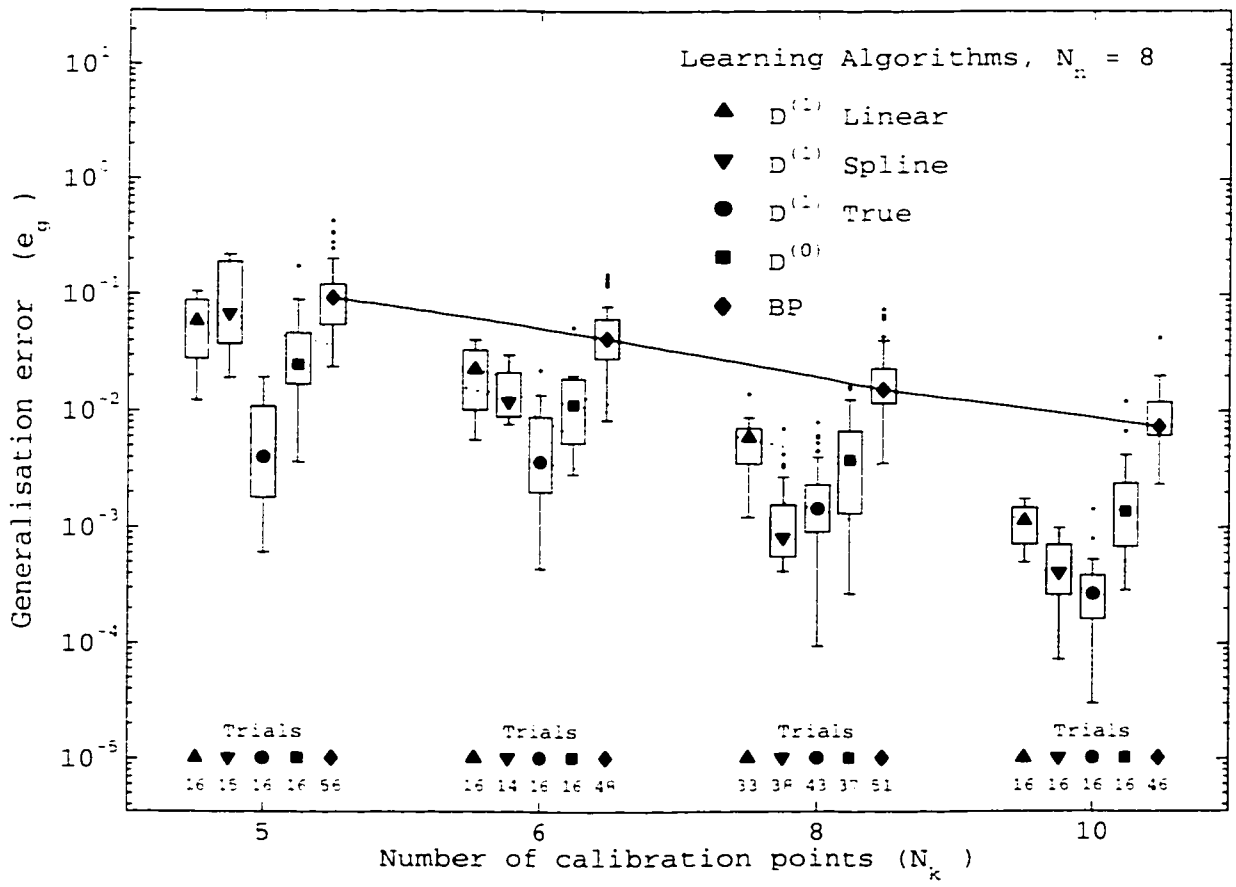
Figure 5.6. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 16$ hidden neurons with weights initialised using the OR8-2 method.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear							**	0.0012	3.5	*	0.0055	2.4
$D^{(1)}$ Spline							***	< 0.0001	6.8	***	< 0.0001	4.1
$D^{(1)}$ True	***	< 0.0001	5.4	***	< 0.0001		***	< 0.0001	3.0	***	< 0.0001	3.1
$D^{(0)}$												

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method.

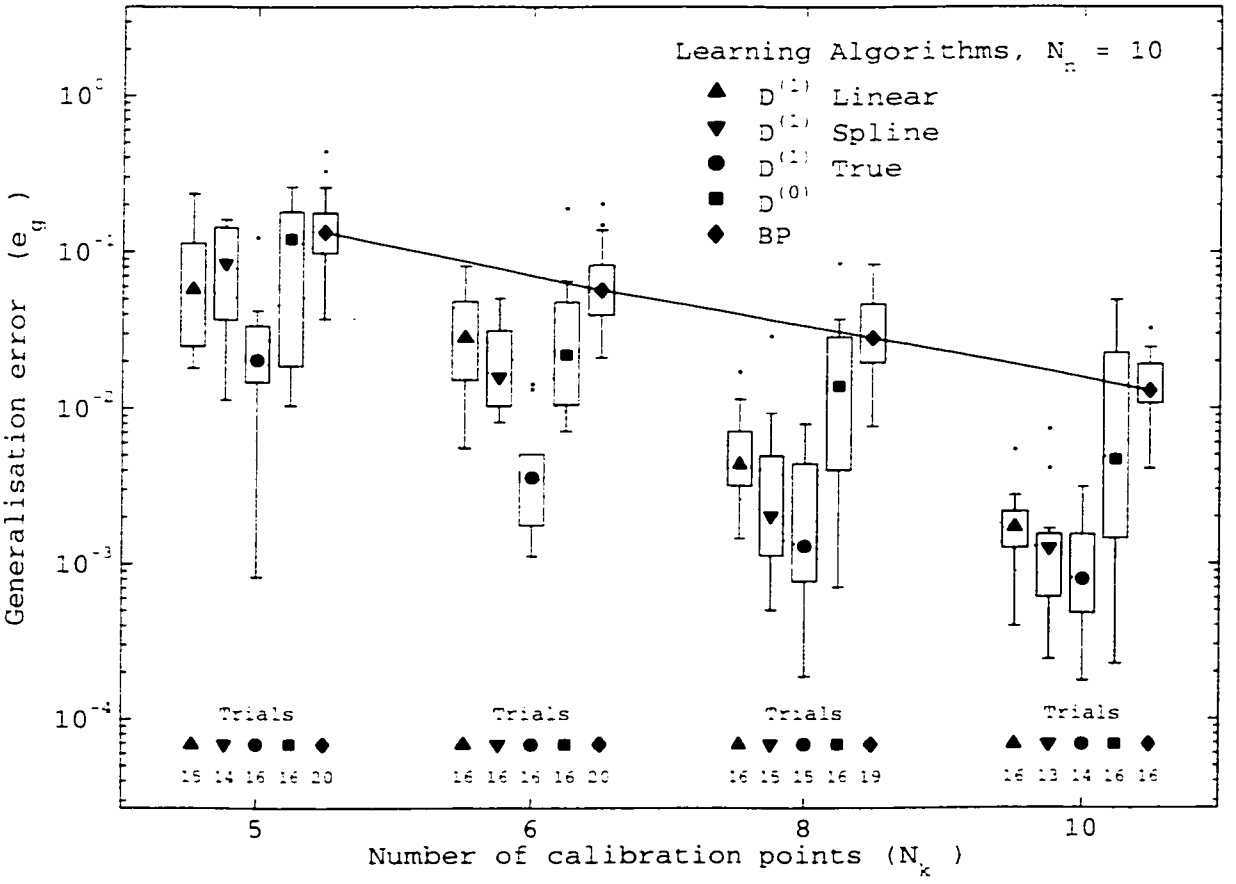
Figure 5.7. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{R_k} \in \mathcal{H}_R$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear				*	0.0054	1.85	***	< 0.0001	2.6	***	< 0.0001	6.4
$D^{(1)}$ Spline				***	< 0.0001	3.5	***	< 0.0001	18.7	***	< 0.0001	17.9
$D^{(1)}$ True	***	< 0.0001	23	***	< 0.0001	11.5	***	< 0.0001	10.5	***	< 0.0001	27
$D^{(0)}$	***	< 0.0001	3.7	***	< 0.0001	3.7	***	< 0.0001	4.0	***	< 0.0001	5.3

Note The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method

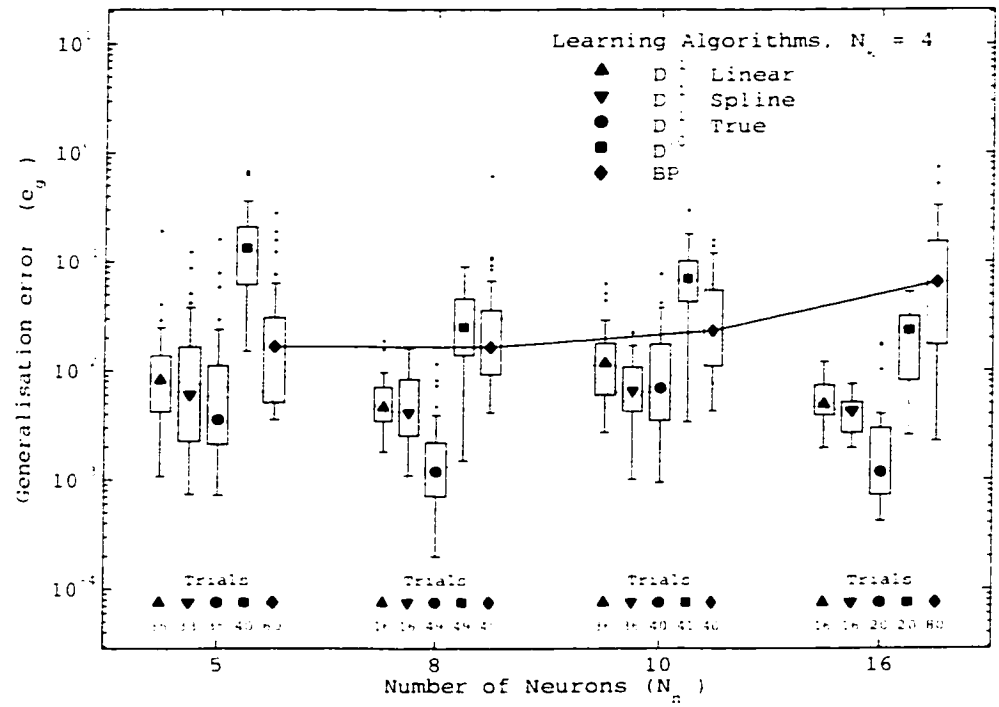
Figure 5.8. The initial calibration errors, as measured by $\hat{\epsilon}_g$, in approximating all $h_{R_k} \in \mathcal{H}_R$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons.



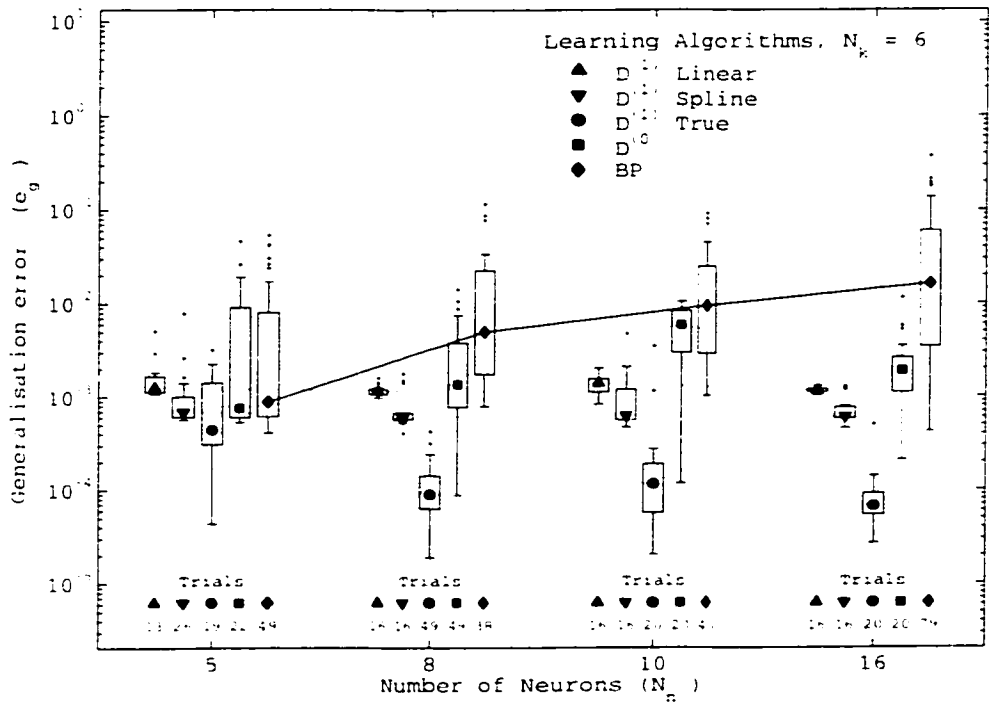
Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_k = 4			at N_k = 6			at N_k = 8			at N_k = 10		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$ Linear	+	0.017	2.3	+	0.013	2.0	***	< 0.0001	6.5	***	< 0.0001	7.5
$D^{(1)}$ Spline				**	0.00052	3.6	***	< 0.0001	13.9	***	< 0.0001	10.3
$D^{(1)}$ True	***	< 0.0001	6.6	***	< 0.0001	15.9	***	< 0.0001	22	***	< 0.0001	16.3
$D^{(0)}$				+	0.0094	2.6				+	0.022	2.8

Note: The significance level, S, is adjusted for four comparisons using the Bonferroni-Dunn method.

Figure 5.9. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{R_k} \in \mathcal{H}_R$ are shown as a function of the number of calibration data points, N_k, for each of the learning algorithms. The calibration model consists of a FFNN having $N_n = 10$ hidden neurons.

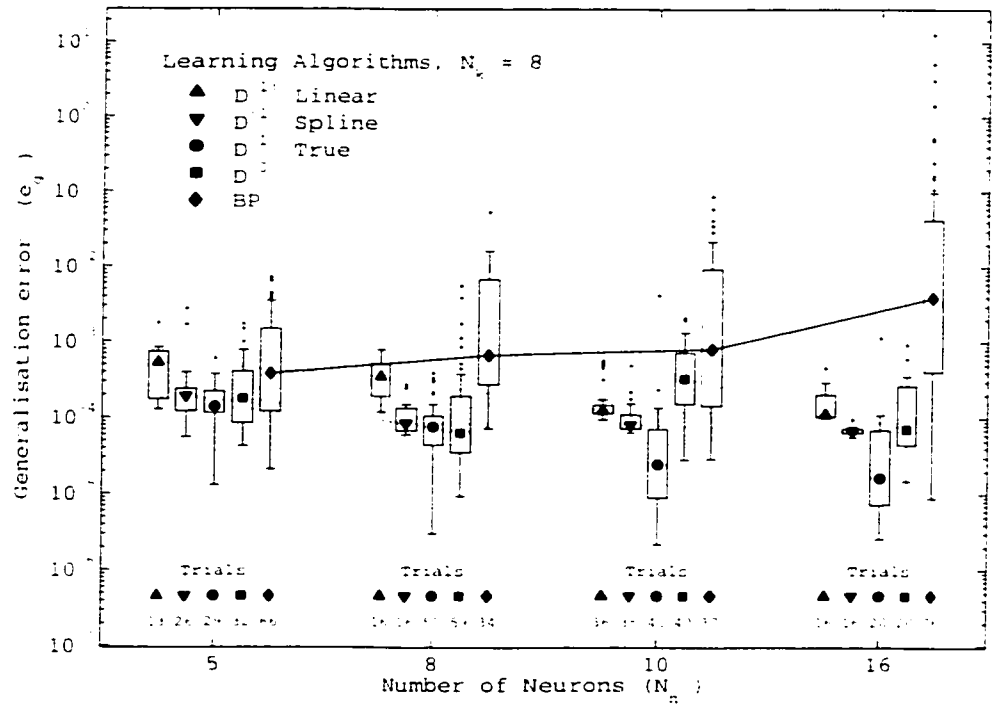


(a)

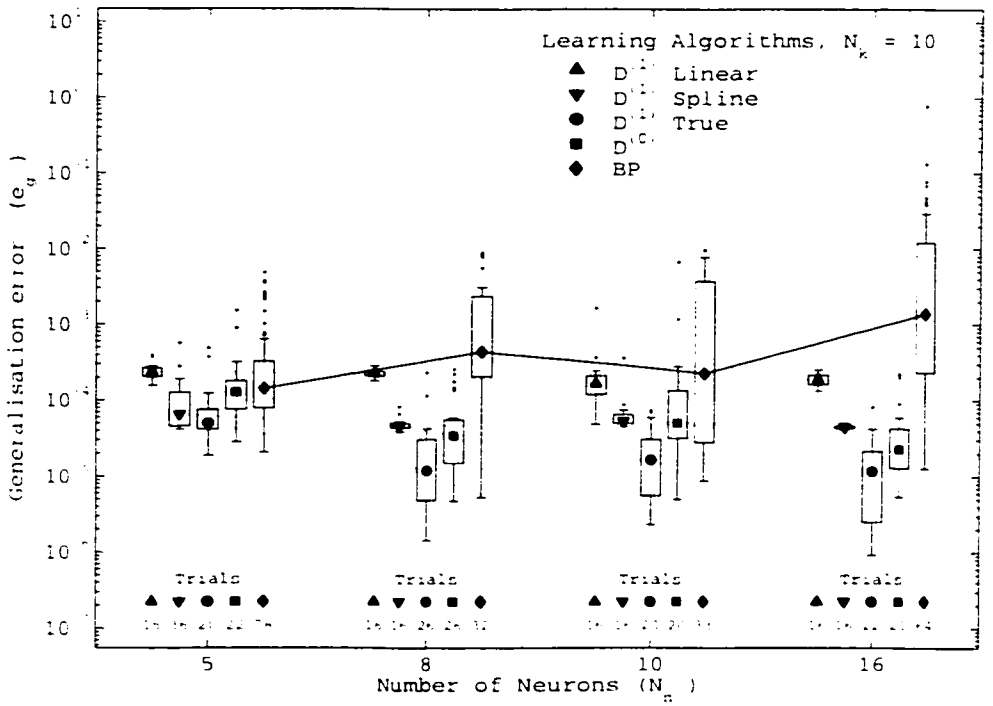


(b)

Figure 5.10. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of hidden neurons, N_n, for each of the learning algorithms.

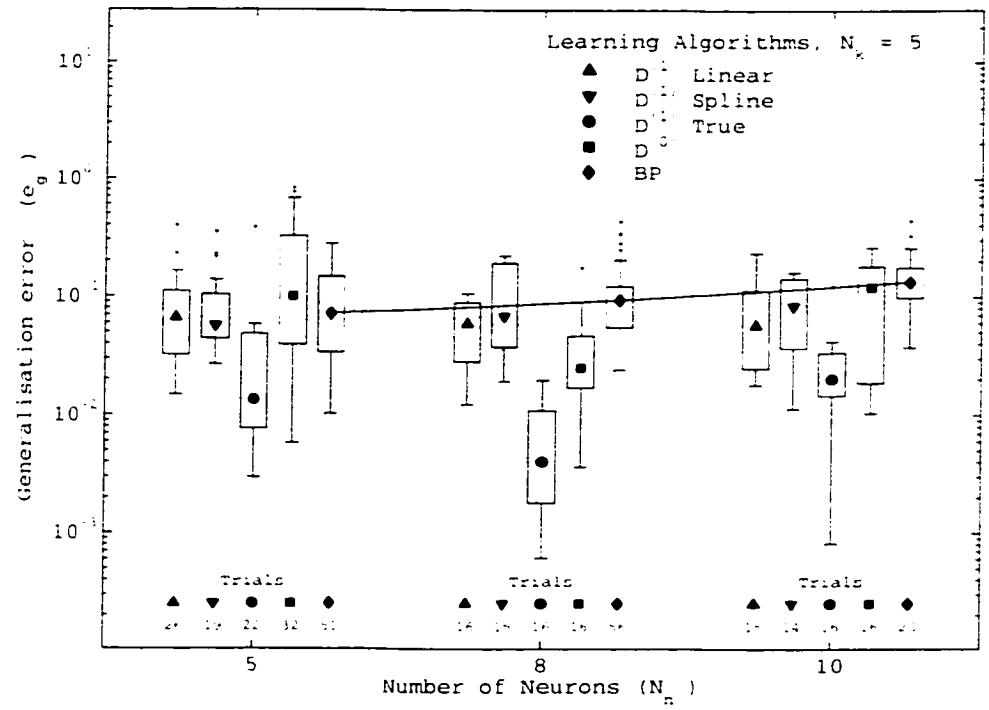


(a)

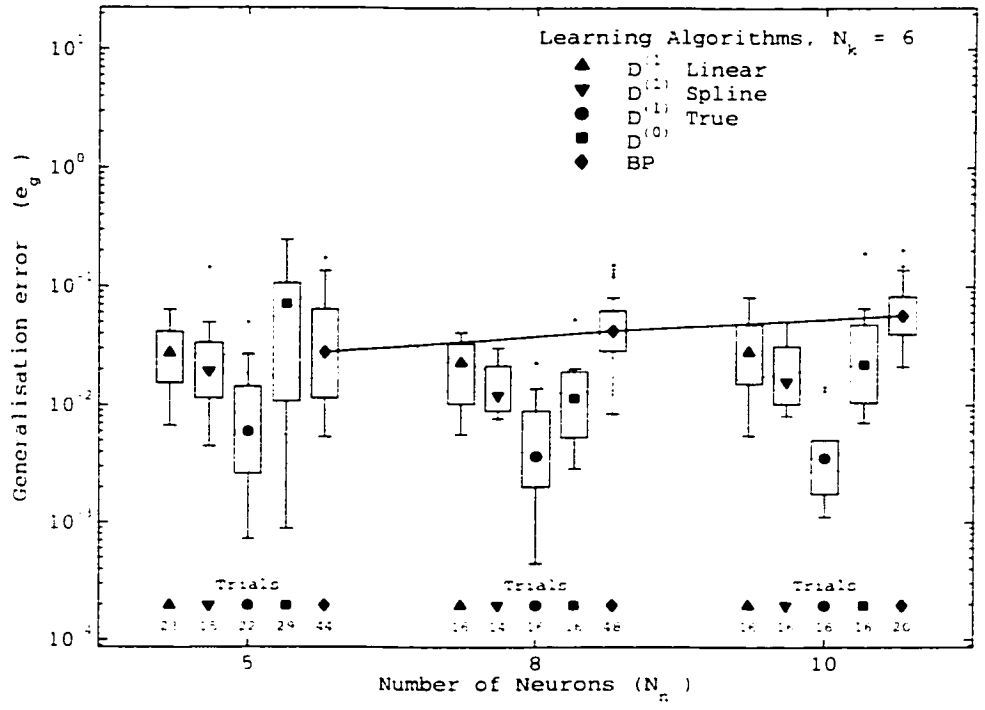


(b)

Figure 5.11. The initial calibration errors, as measured by $\hat{\epsilon}_g$, in approximating all $h_{P_k} \in \mathcal{H}_P$ are shown as a function of the number of hidden neurons, N_n, for each of the learning algorithms.

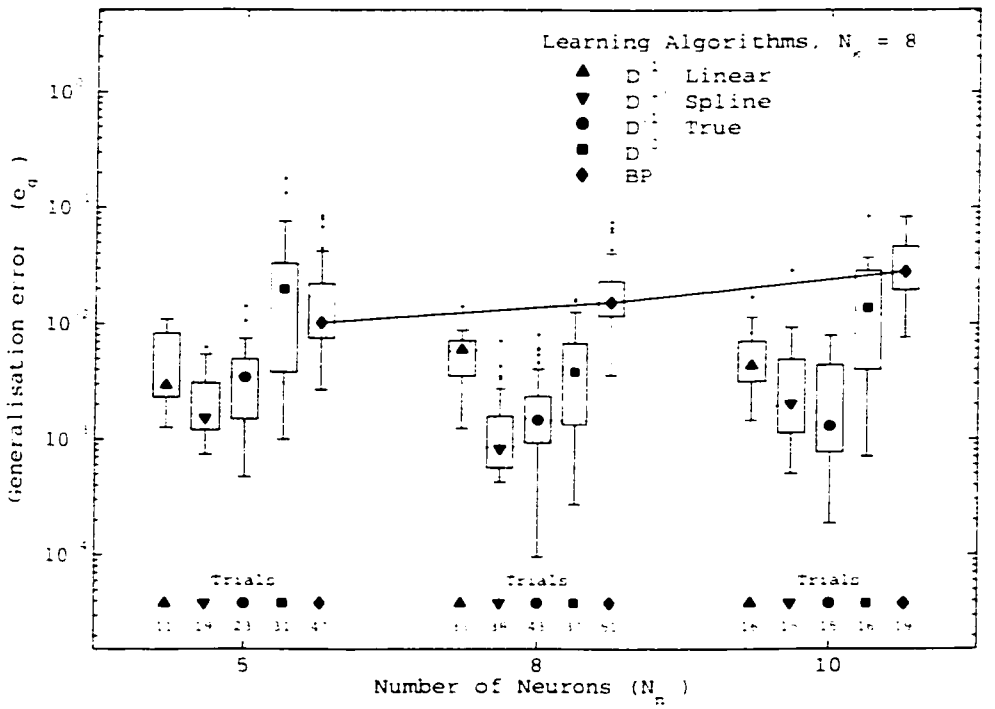


(a)

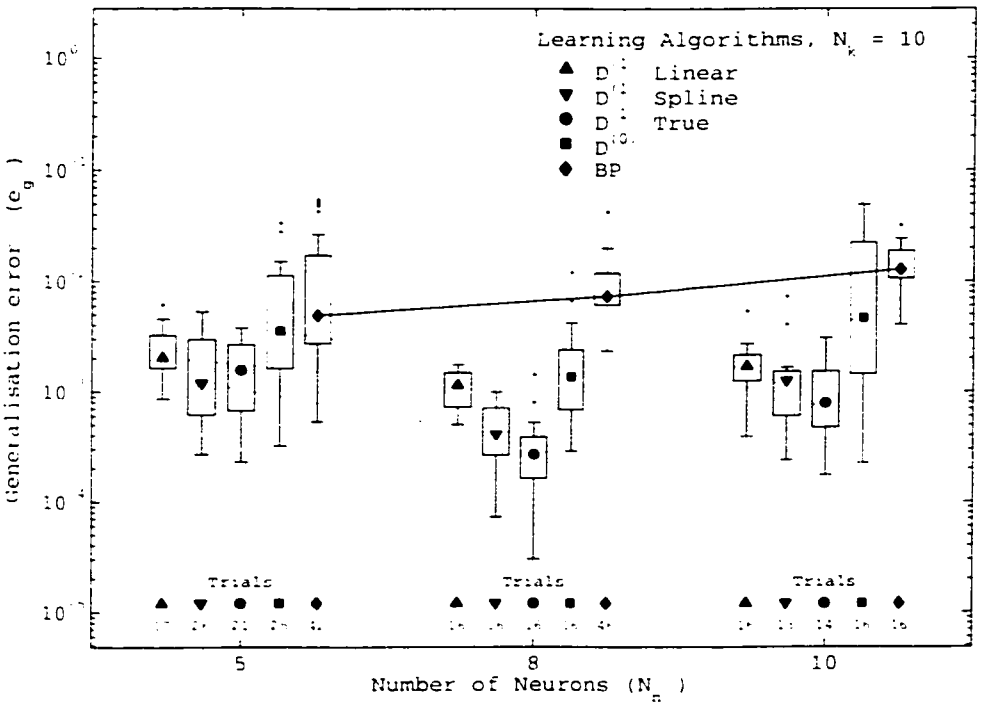


(b)

Figure 5.12. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{R_k} \in \mathcal{H}_R$ are shown as a function of the number of hidden neurons, N_n, for each of the learning algorithms.



(a)



(b)

Figure 5.13. The initial calibration errors, as measured by $\hat{e}_g$, in approximating all $h_{R_k} \in \mathcal{H}_R$ are shown as a function of the number of hidden neurons, N_n, for each of the learning algorithms.

5.3 Calibration Transfer Simulation Results

The calibration transfer simulation results, as performed according the methods described in chapter 4, are summarised with twenty six figures given by Figure (5.14) through Figure (5.39). The presentation of the data is given in the same format as was used to present the initial calibration simulations.

Specifically, the figures are organised to show, for each learning algorithm, the estimated calibration error, measured by $\hat{e}_g$, as a function of the calibration points, N_k, given a particular number of hidden neurons, N_n and the use of a particular initial calibration model. In a manner similar to the initial calibration simulation results, for any specific number of calibration points, number of neurons, and type of calibration transfer method, the error $\hat{e}_g$ of a series of simulation trials are summarised as a single box plot item that is drawn in the figure. The number of trials used to obtain the specific form of the box plot item is listed directly below the corresponding item.

Also, each figure has an associated table listing the level of significance, S, the p values, and the improvement factor, I_F, associated with the difference between the median error in the Std. Cal. BP method and the median errors in the various other transfer methods at each value of N_k. For these calibration transfer simulations, the significance levels are adjusted for the number of comparisons using the Bonferroni-Dunn method [146]. Again, it should be noted that only simulation results that had achieved a p value less than or equal to the adjusted significance level 0.1 are listed in the table.

The overall structure of the presentation of the simulation results is formed by first grouping the results obtained using models drawn from $\mathcal{H}_P$, that is, Figure (5.14) through Figure (5.28). Within this group the results are organised in order of increasing number of hidden neurons. For the results at any specific number of neurons, the results are further organised to first show calibration transfer using models similar to each other and then models that are not similar to each other. Finally, for any one particular result with N_n neurons using models with a specified similarity, the

individual figures represent the calibration transfers using initial calibration models drawn from a particular set.

The next group of simulation results are those using models drawn from $\mathcal{H}_R$. shown in Figure (5.29) through Figure (5.39). The structure of the presentation of these results follows that described for the models drawn from $\mathcal{H}_P$.

As an specific example of this organisation of the results. consider the first three figures. that is. Figure (5.14) through Figure (5.16). Here the results are organised to first show the calibration transfer from $\hat{h}_k = \hat{h}_{P_2}$ to $\hat{h}_l = \hat{h}_{P_8}$. that is. the transfer between the 8th order polynomial models that are similar to each other. All three of these figures show the results of the calibration transfer using a FFNN having $N_n = 5$ hidden neurons. In addition. each figure represents the results of the calibration transfer using a different set of initial calibration models. The particular set of initial calibration models that are used for the results shown in a figure are identified within the figure's caption.

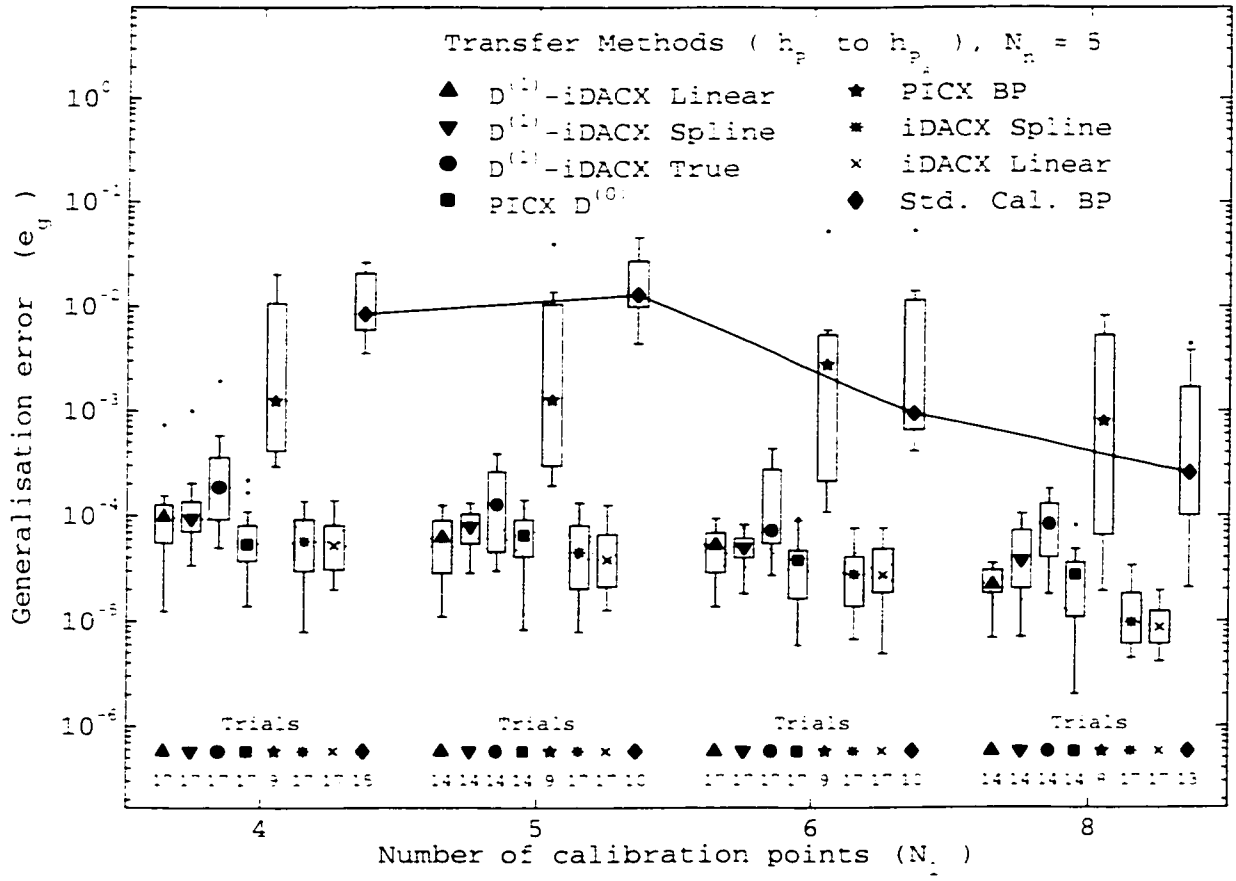
Table (5.1) and Table (5.2) identify the sets from which the initial calibration models are drawn. For each set of initial calibration models. the median error. 25th and 75th percentile values are listed. In addition. the number of members in a specific set are also given.

Number of Neurons used for $\hat{h}_k$	Median, 25th and 75th percentile Error in $\mathcal{H}_{P_{ij}}$		
	$\bar{\bar{\mathcal{H}}}_{P_{1a}}$ median $^{75\%}_{25\%}$	$\bar{\bar{\mathcal{H}}}_{P_{1b}}$ median $^{75\%}_{25\%}$	$\bar{\bar{\mathcal{H}}}_{P_{1c}}$ median $^{75\%}_{25\%}$
5	$\bar{\bar{\mathcal{H}}}_{P_{5a}} = 17$ 1.2×10^{-4} 6.7×10^{-5} 2.7×10^{-5}	$\bar{\bar{\mathcal{H}}}_{P_{5b}} = 10$ 1.4×10^{-4} 6.3×10^{-5} 5.4×10^{-5}	$\bar{\bar{\mathcal{H}}}_{P_{5c}} = 11$ 1.5×10^{-4} 1.2×10^{-4} 7.9×10^{-5}
8	$\bar{\bar{\mathcal{H}}}_{P_{8a}} = 16$ 2.2×10^{-5} 1.2×10^{-5} 5.7×10^{-6}	$\bar{\bar{\mathcal{H}}}_{P_{8b}} = 7$ 1.6×10^{-4} 1.1×10^{-4} 7.0×10^{-5}	$\bar{\bar{\mathcal{H}}}_{P_{8c}} = 11$ 3.0×10^{-5} 1.7×10^{-5} 9.7×10^{-6}
10	$\bar{\bar{\mathcal{H}}}_{P_{10a}} = 12$ 6.1×10^{-5} 2.4×10^{-5} 8.0×10^{-6}	$\bar{\bar{\mathcal{H}}}_{P_{10b}} = 13$ 9.0×10^{-5} 7.4×10^{-5} 7.2×10^{-5}	$\bar{\bar{\mathcal{H}}}_{P_{10c}} = 15$ 3.2×10^{-4} 1.3×10^{-4} 4.9×10^{-5}

Table 5.1. The median, 25th and 75th percentiles associated with the error $\hat{e}_g$ in the sets of the initial calibration models $\hat{h}_k$ used in the calibration transfer simulations involving models selected from $\mathcal{H}_P$.

Number of Neurons used for $\hat{h}_k$	Median, 25th and 75th percentile Error in $\mathcal{H}_{R_{ij}}$		
	$\bar{\bar{\mathcal{H}}}_{R_{1a}}$ $\left(\text{median } ^{75\%}_{25\%}\right)$	$\bar{\bar{\mathcal{H}}}_{R_{1b}}$ $\left(\text{median } ^{75\%}_{25\%}\right)$	$\bar{\bar{\mathcal{H}}}_{R_{1c}}$ $\left(\text{median } ^{75\%}_{25\%}\right)$
5	$\bar{\bar{\mathcal{H}}}_{R_{5a}} = 10$ 5.1×10^{-3} 4.2×10^{-3} 3.6×10^{-3}	$\bar{\bar{\mathcal{H}}}_{R_{5b}} = 10$ 1.3×10^{-3} 7.4×10^{-3} 2.0×10^{-3}	$\bar{\bar{\mathcal{H}}}_{R_{5c}} = 9$ 2.4×10^{-2} 1.5×10^{-2} 5.4×10^{-3}
8	$\bar{\bar{\mathcal{H}}}_{R_{8a}} = 9$ 1.1×10^{-3} 9.4×10^{-4} 8.4×10^{-4}	$\bar{\bar{\mathcal{H}}}_{R_{8b}} = 11$ 1.0×10^{-3} 8.1×10^{-5} 6.2×10^{-5}	$\bar{\bar{\mathcal{H}}}_{R_{8c}} = 8$ 4.0×10^{-3} 1.8×10^{-3} 1.0×10^{-3}

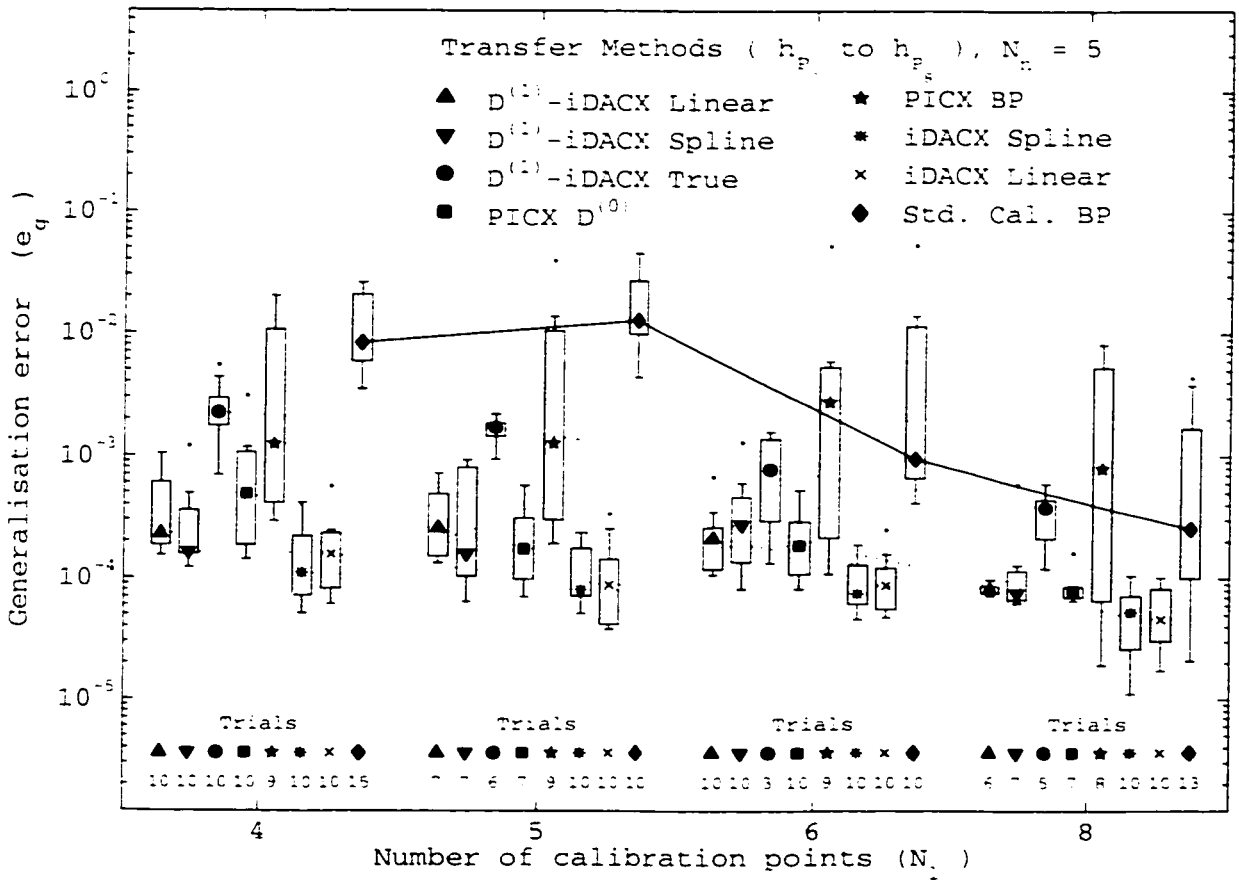
Table 5.2. The median, 25th and 75th percentiles associated with the error $\hat{e}_g$ in the sets of the initial calibration models $\hat{h}_k$ used in the calibration transfer simulations involving models selected from $\mathcal{H}_R$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (–) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	86	***	< 0.0001	210	***	< 0.0001	17.9	**	0.00040	11.6
$D^{(1)}$-iDACX Spline	***	< 0.0001	90	**	0.00025	164	**	0.00018	18.9			
$D^{(1)}$-iDACX True	*	0.0019	45	*	0.0047	100						
PICX $D^{(0)}$	***	< 0.0001	157	***	< 0.0001	197	***	< 0.0001	25	**	0.00043	9.2
PICX BP												
iDACX Spline	***	< 0.0001	148	***	< 0.0001	290	***	< 0.0001	34	***	< 0.0001	26
iDACX Linear	***	< 0.0001	162	***	< 0.0001	340	***	< 0.0001	35	***	< 0.0001	29

Note The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method

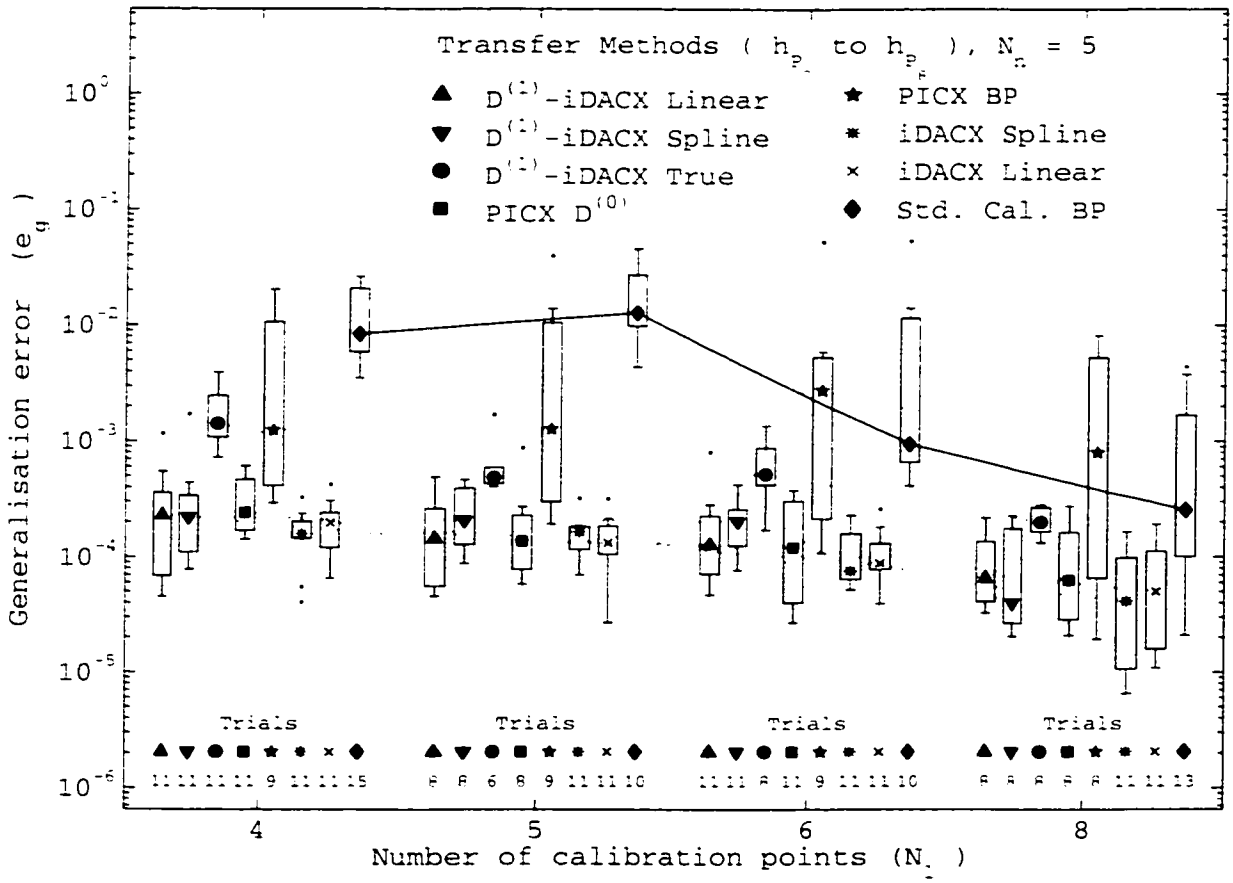
Figure 5.14. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $h_k = h_{P_5}$ to $h_l = h_{P_N}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $h_k \in \mathcal{H}_{P_{5k}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	37	*	0.0031	50	*	0.0043	4.4			
$D^{(1)}$-iDACX Spline	***	< 0.0001	52	**	0.00086	81						
$D^{(1)}$-iDACX True												
PICX $D^{(0)}$	**	0.00040	17.3	**	0.00026	74	*	0.0020	5.1			
PICX BP												
iDACX Spline	***	< 0.0001	76	***	< 0.0001	160	***	< 0.0001	12.5	**	0.00080	4.8
iDACX Linear	***	< 0.0001	54	***	< 0.0001	140	***	< 0.0001	10.8	*	0.0023	3.5

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method

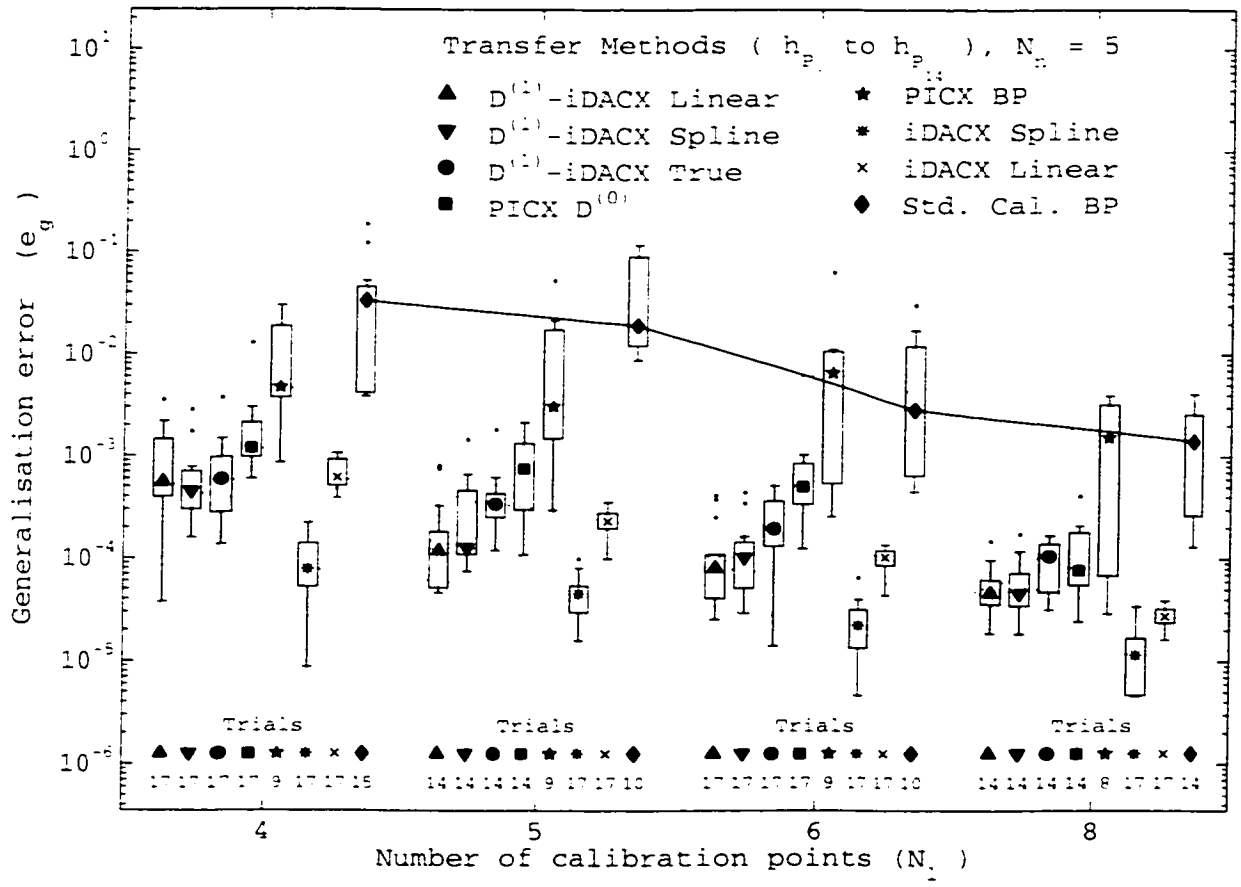
Figure 5.15. The calibration errors, as measured by $\dot{\epsilon}_g$, in performing a calibration transfer from $\dot{h}_k = \dot{h}_{P_4}$ to $\dot{h}_l = \dot{h}_{P_5}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\dot{h}_k \in \mathcal{H}_{P_{5b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	37	***	< 0.0001	89	***	0.00014	7.3			
$D^{(1)}$-iDACX Spline	***	< 0.0001	38	**	0.00089	62	*	0.0028	4.6			
$D^{(1)}$-iDACX True												
PICX $D^{(0)}$	***	< 0.0001	35	***	< 0.0001	94	***	< 0.0001	7.8			
PICX BP												
iDACX Spline	***	< 0.0001	54	***	< 0.0001	78	***	< 0.0001	12.4	**	0.0011	6.1
iDACX Linear	***	< 0.0001	42	***	< 0.0001	96	***	< 0.0001	10.7	*	0.0033	5.1

Note The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

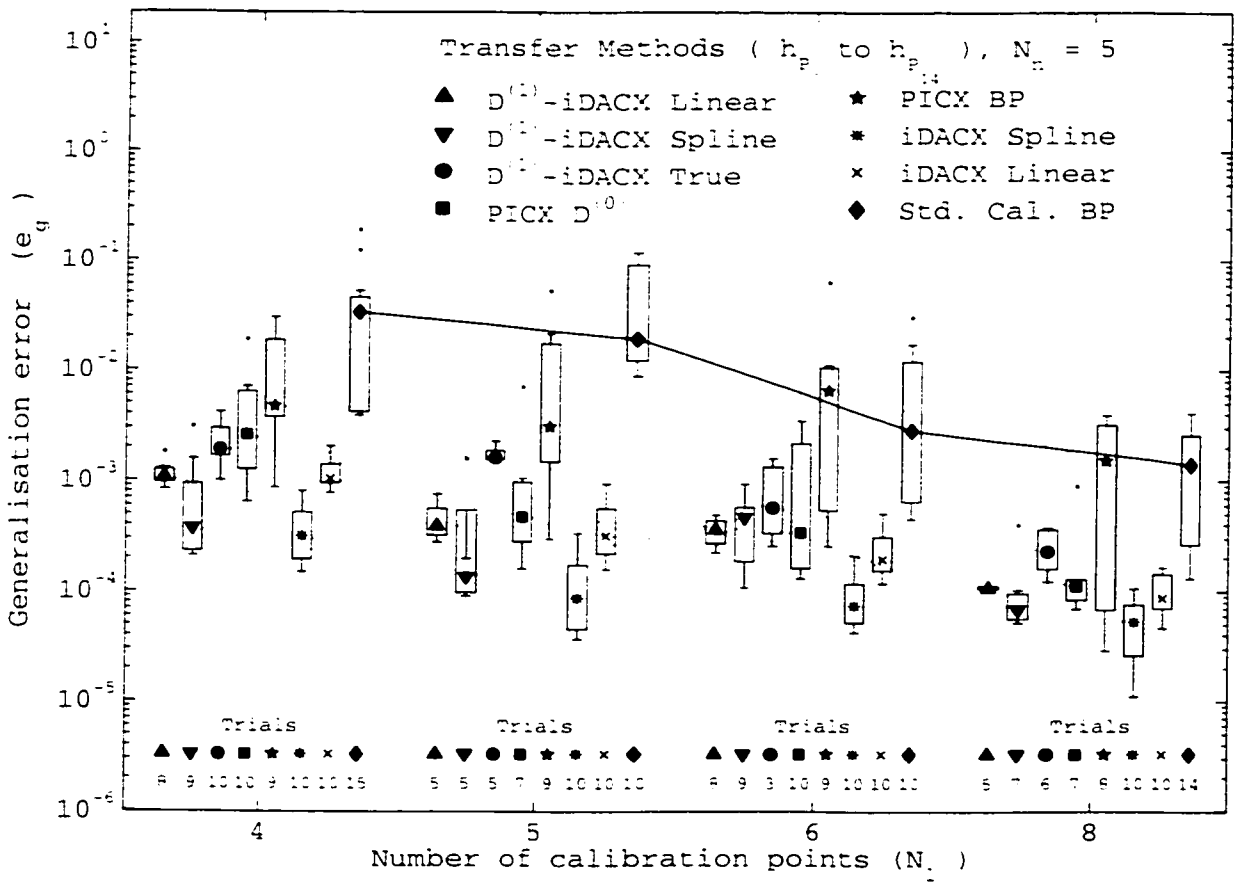
Figure 5.16. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_r}$ to $\hat{h}_l = \hat{h}_{P_s}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{Sc}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	60	***	< 0.0001	161	***	< 0.0001	35	***	0.00013	30
$D^{(1)}$-iDACX Spline	***	< 0.0001	74	***	< 0.0001	151	***	< 0.0001	28	***	0.00014	31
$D^{(1)}$-iDACX True	***	< 0.0001	56	*	0.0027	56	*	0.0051	14.5	+	0.0134	13.2
PICX $D^{(0)}$	+	0.011	28									
PICX BP												
iDACX Spline	***	< 0.0001	420	***	< 0.0001	420	***	< 0.0001	127	***	< 0.0001	121
iDACX Linear	***	< 0.0001	54	***	< 0.0001	82	***	< 0.0001	28	***	< 0.0001	51

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

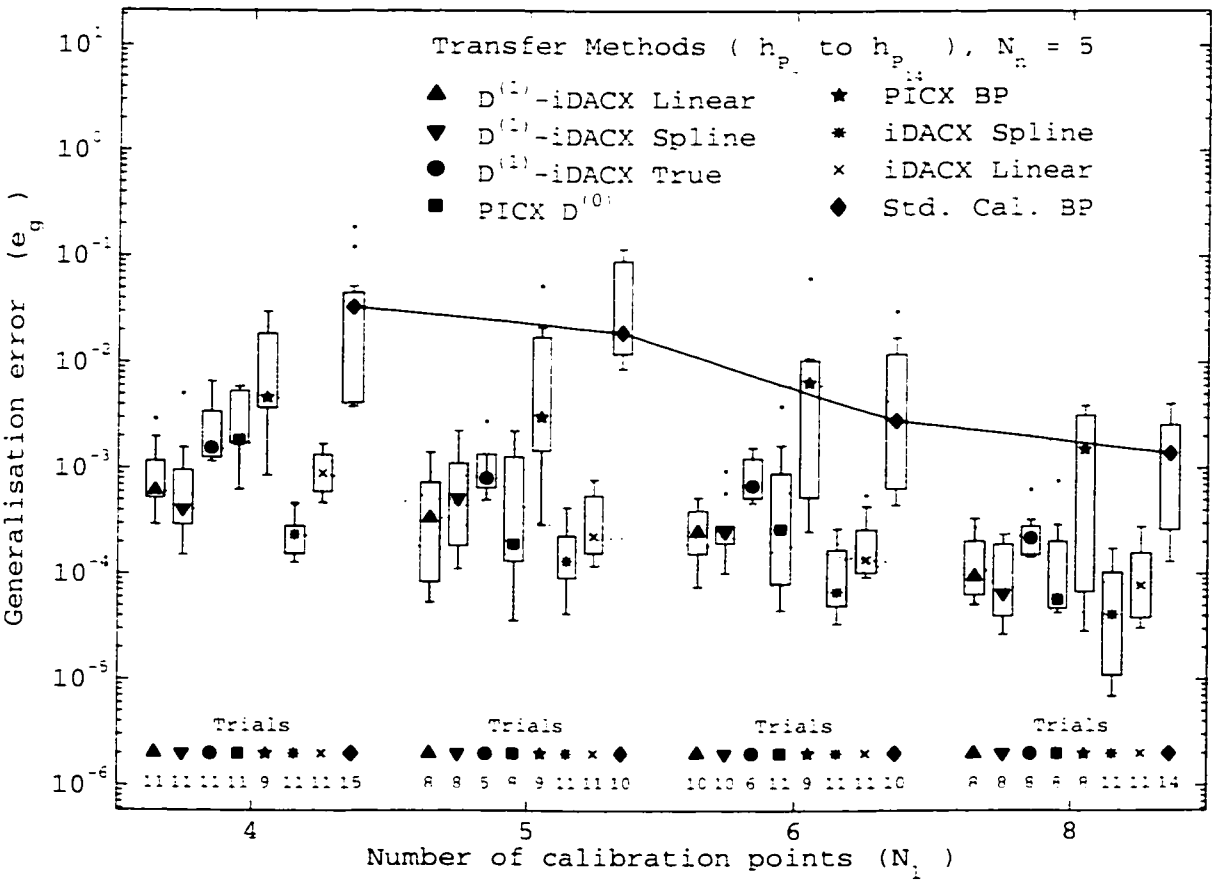
Figure 5.17. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_5}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{5a}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (-) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	**	0.00015	31	*	0.0035	49						
$D^{(1)}$-iDACX Spline	***	< 0.0001	90	***	< 0.0001	142	-	0.0098	6.1	***	0.00011	20
$D^{(1)}$-iDACX True	-	0.011	17.5									
PICX $D^{(0)}$				*	0.0027	41				-	0.0094	12.6
PICX BP												
iDACX Spline	***	< 0.0001	106	***	< 0.0001	220	***	< 0.0001	38	***	< 0.0001	26
iDACX Linear	***	< 0.0001	32	***	< 0.0001	60	***	< 0.0001	14.7	**	0.00029	16

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method

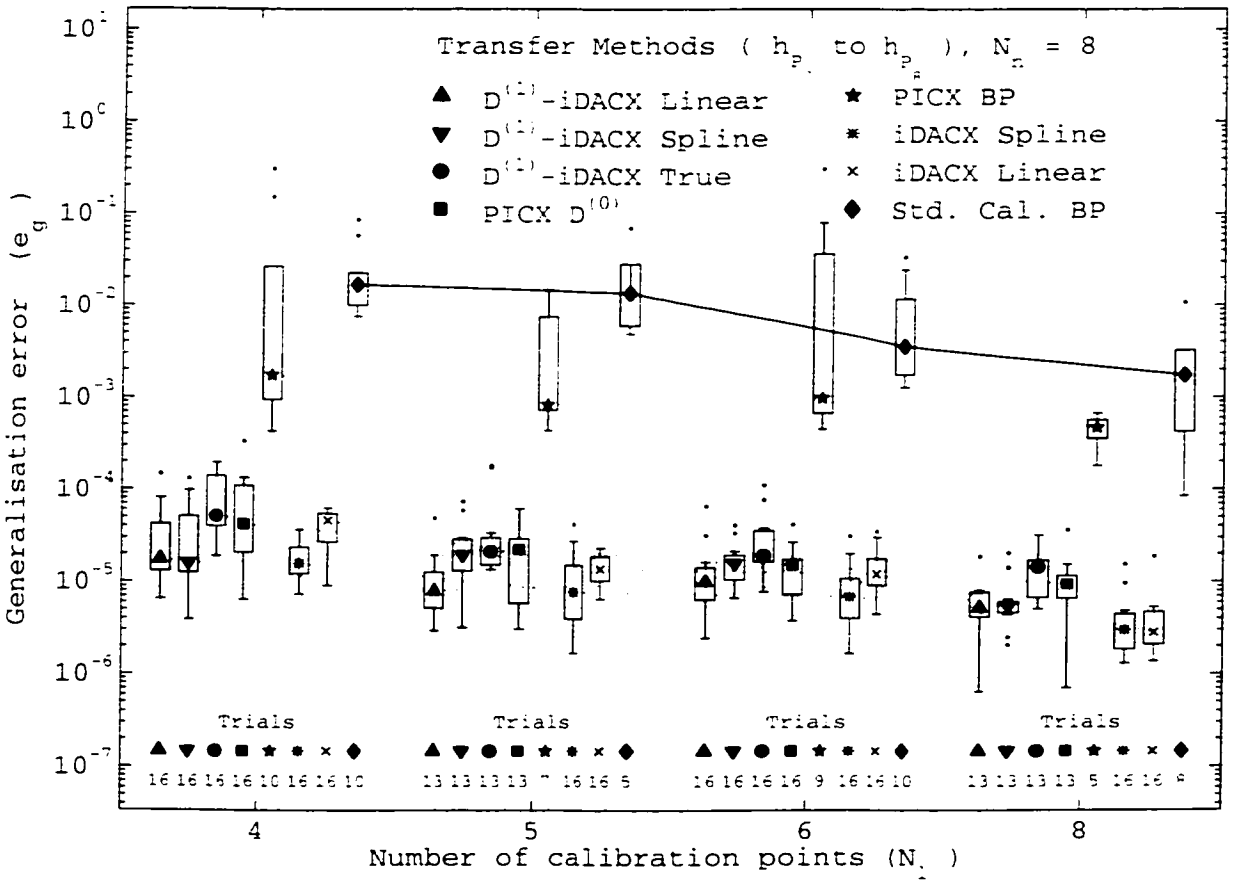
Figure 5.18. The calibration errors, as measured by $\hat{\epsilon}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_5}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{5b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	54	***	< 0.0001	56	*	0.0016	11.2	*	0.0053	14.6
$D^{(1)}$-iDACX Spline	***	< 0.0001	81	*	0.0015	37	*	0.0043	11.1	**	0.00033	21
$D^{(1)}$-iDACX True	+	0.0123	21									
PICX $D^{(0)}$				**	0.00019	99	+	0.0074	10.6	**	0.00098	24
PICX BP												
iDACX Spline	***	< 0.0001	140	***	< 0.0001	142	***	< 0.0001	42	***	< 0.0001	34
iDACX Linear	***	< 0.0001	37	***	< 0.0001	83	***	< 0.0001	21	***	< 0.0001	18.0

Note The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method

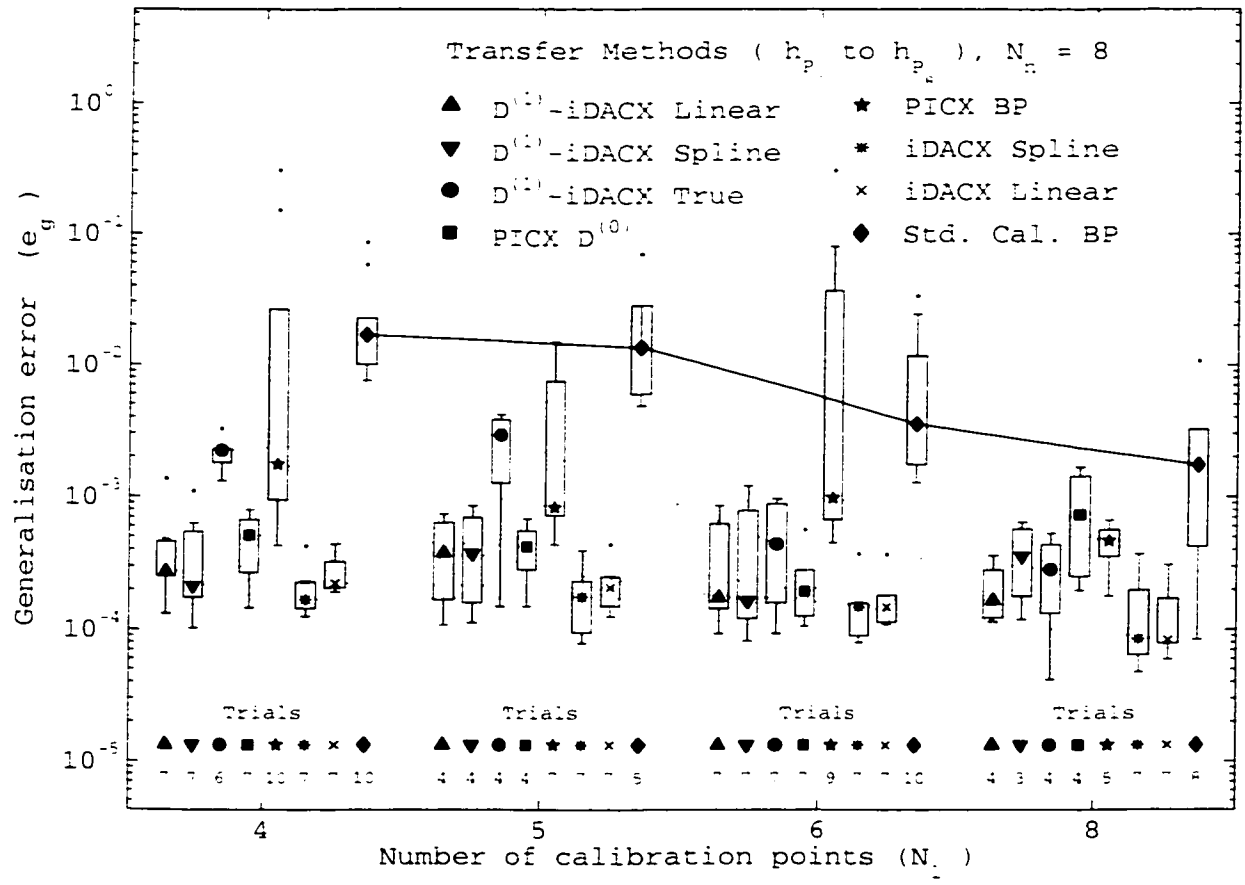
Figure 5.19. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_2}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{5c}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	960	***	< 0.0001	1710	***	< 0.0001	360	***	0.00010	330
$D^{(1)}$-iDACX Spline	***	< 0.0001	1060	*	0.0056	700	**	0.00015	230	**	0.00022	320
$D^{(1)}$-iDACX True	*	0.0059	330				+	0.0095	185			
PICX $D^{(0)}$	***	0.00012	400	*	0.0029	610	***	< 0.0001	230	*	0.0063	183
PICX BP												
iDACX Spline	***	< 0.0001	1070	***	< 0.0001	1800	***	< 0.0001	520	***	< 0.0001	580
iDACX Linear	***	0.00012	360	**	0.00030	990	***	< 0.0001	300	***	< 0.0001	620

Note : The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

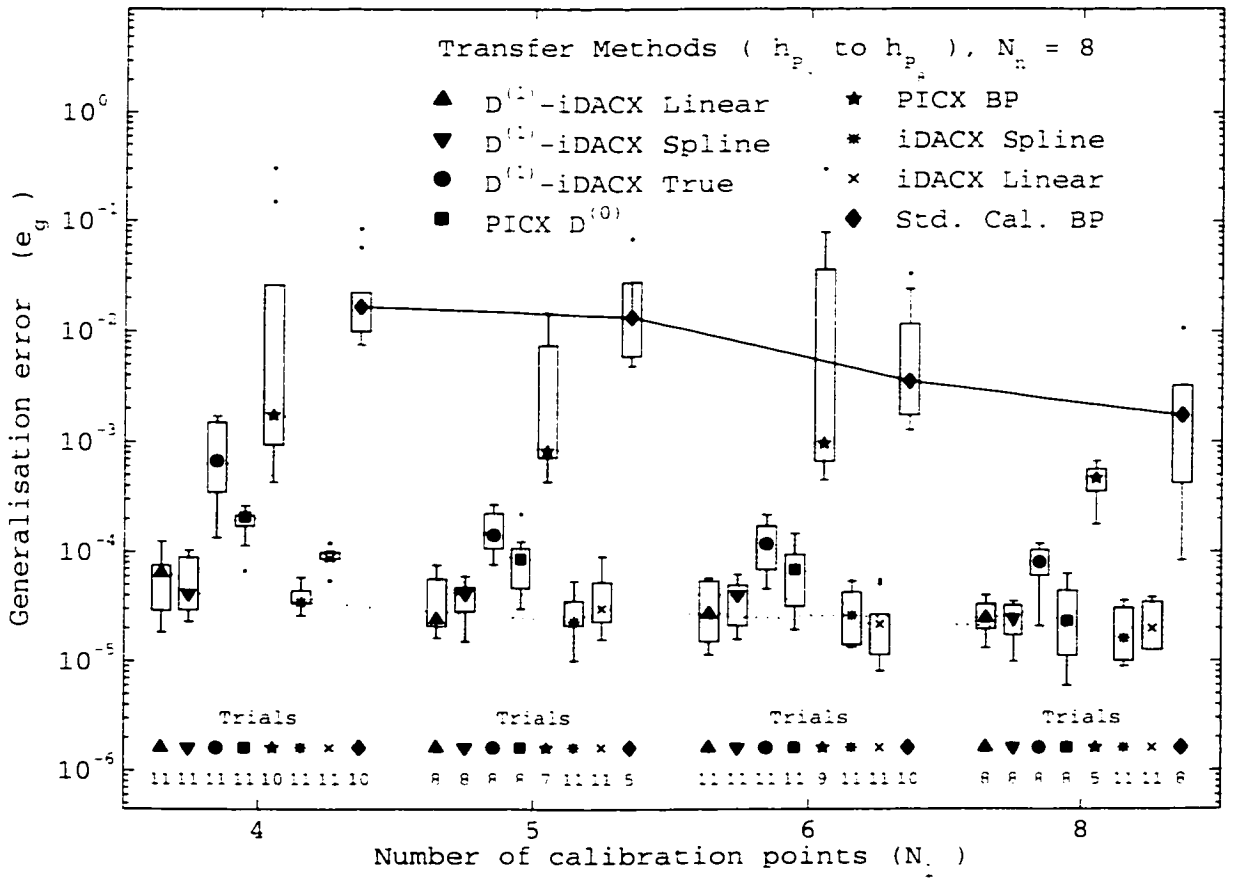
Figure 5.20. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_5}$ to $\hat{h}_l = \hat{h}_{P_8}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{5a}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	**	0.00035	62	+	0.0096	36	**	0.00088	20			
$D^{(1)}$-iDACX Spline	***	< 0.0001	79	+	0.0125	37	**	0.00088	22			
$D^{(1)}$-iDACX True							*	0.0065	8.1			
PICX $D^{(0)}$	**	0.00015	33				**	0.00038	18.4			
PICX BP												
iDACX Spline	***	< 0.0001	100	***	< 0.0001	77	***	< 0.0001	24	**	0.00091	20
iDACX Linear	***	< 0.0001	75	**	0.00038	66	***	< 0.0001	21	**	0.00056	21

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

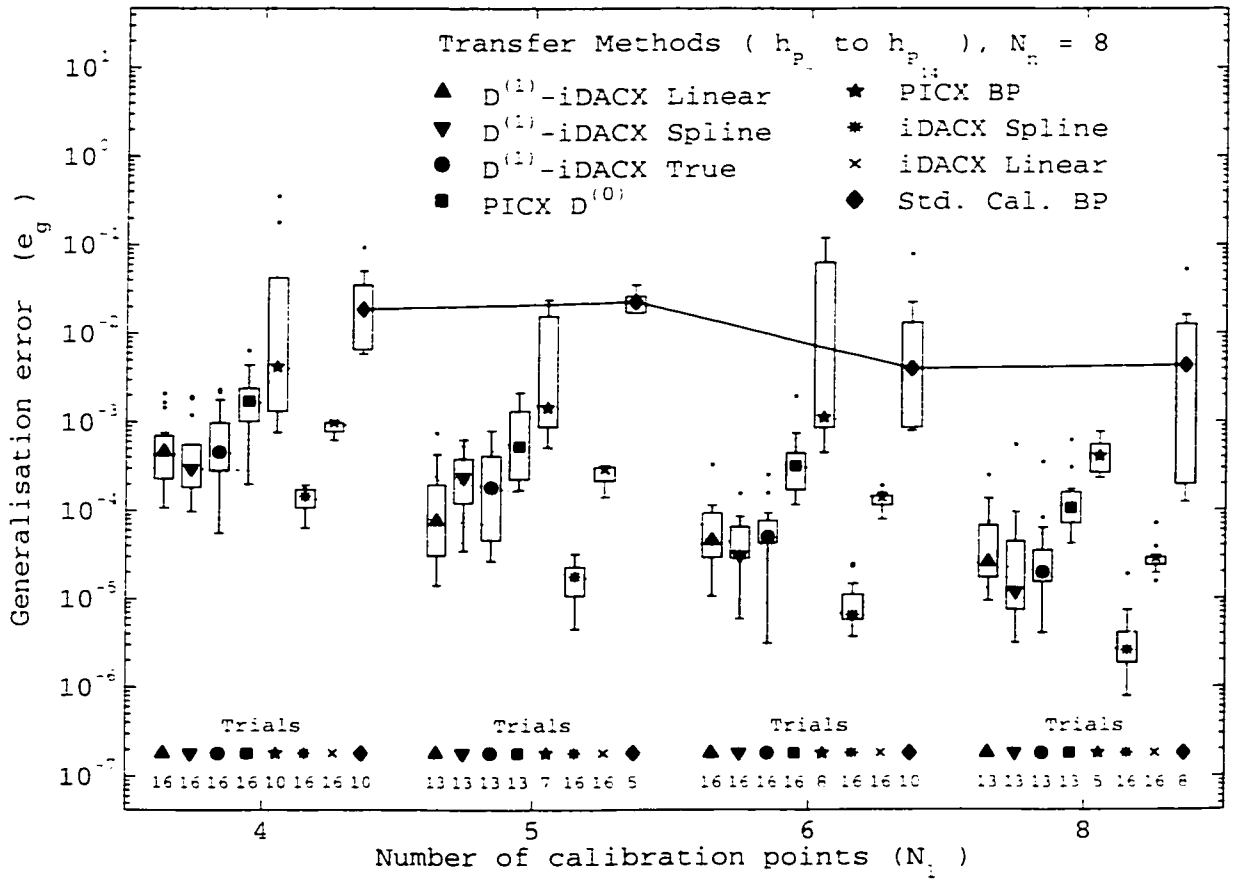
Figure 5.21. The calibration errors, as measured by $\dot{e}_g$, in performing a calibration transfer from $\dot{h}_k = \dot{h}_{P_5}$ to $\dot{h}_l = \dot{h}_{P_8}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\dot{h}_k \in \mathcal{H}_{P_{8b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	260	***	< 0.0001	570	***	< 0.0001	130	**	0.00093	70
$D^{(1)}$-iDACX Spline	***	< 0.0001	410	**	0.00053	330	***	< 0.0001	88	**	0.00022	71
$D^{(1)}$-iDACX True												
PICX $D^{(0)}$	*	0.0069	80				**	0.0013	51	**	0.00015	74
PICX BP												
iDACX Spline	***	< 0.0001	490	***	< 0.0001	590	***	< 0.0001	135	***	< 0.0001	107
iDACX Linear	***	< 0.0001	190	***	< 0.0001	450	***	< 0.0001	163	***	< 0.0001	87

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

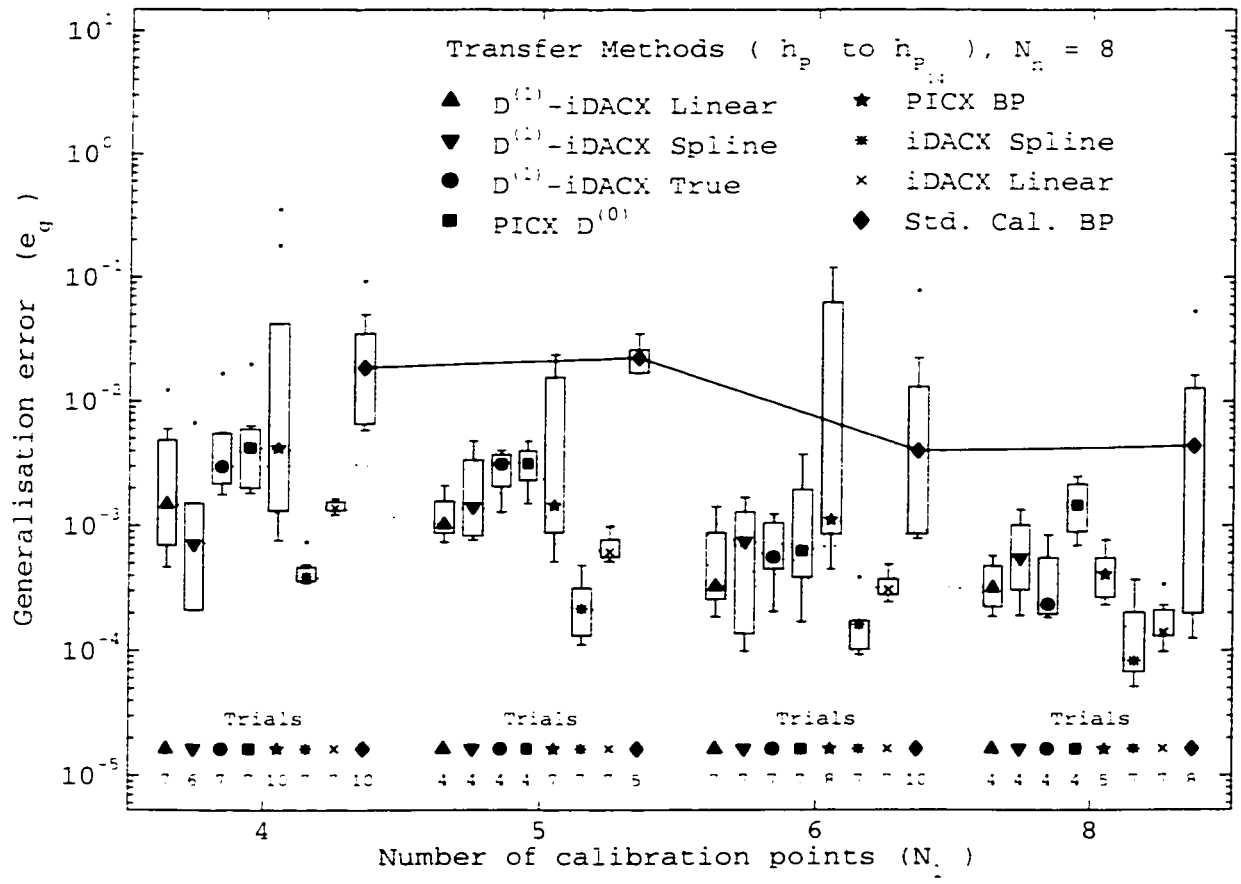
Figure 5.22. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_1}$ to $\hat{h}_l = \hat{h}_{P_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{Sc}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	40	***	< 0.0001	300	***	< 0.0001	88	*	0.0020	168
$D^{(1)}$-iDACX Spline	***	< 0.0001	63	*	0.0037	95	***	< 0.0001	130	***	< 0.0001	370
$D^{(1)}$-iDACX True	***	< 0.0001	41	*	0.0015	124	***	< 0.0001	80	**	0.00017	220
PICX $D^{(0)}$												
PICX BP												
iDACX Spline	***	< 0.0001	129	***	< 0.0001	1270	***	< 0.0001	610	***	< 0.0001	1680
iDACX Linear	*	0.0019	19.4	*	0.0059	79	*	0.0068	29	**	0.00011	155

Note : The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

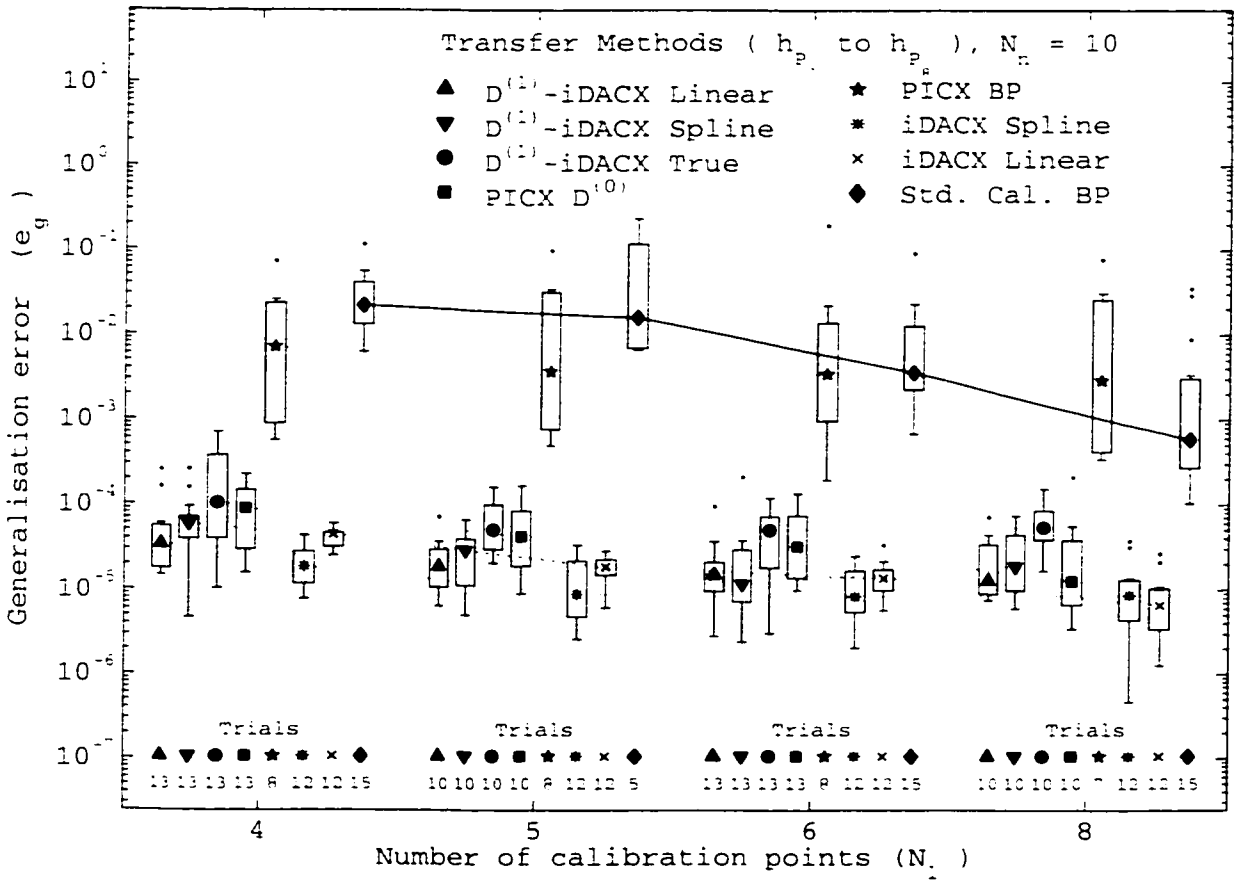
Figure 5.23. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_5}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{sa}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	*	0.0023	12.6				*	0.0069	12.2			
$D^{(1)}$-iDACX Spline	***	< 0.0001	26				+	0.0084	5.3			
$D^{(1)}$-iDACX True												
PICX $D^{(0)}$												
PICX BP												
iDACX Spline	***	< 0.0001	48	***	< 0.0001	105	***	< 0.0001	25	**	0.00085	53
iDACX Linear	**	0.00075	13.6	**	0.00033	37	**	0.00068	13.0	*	0.0053	31

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

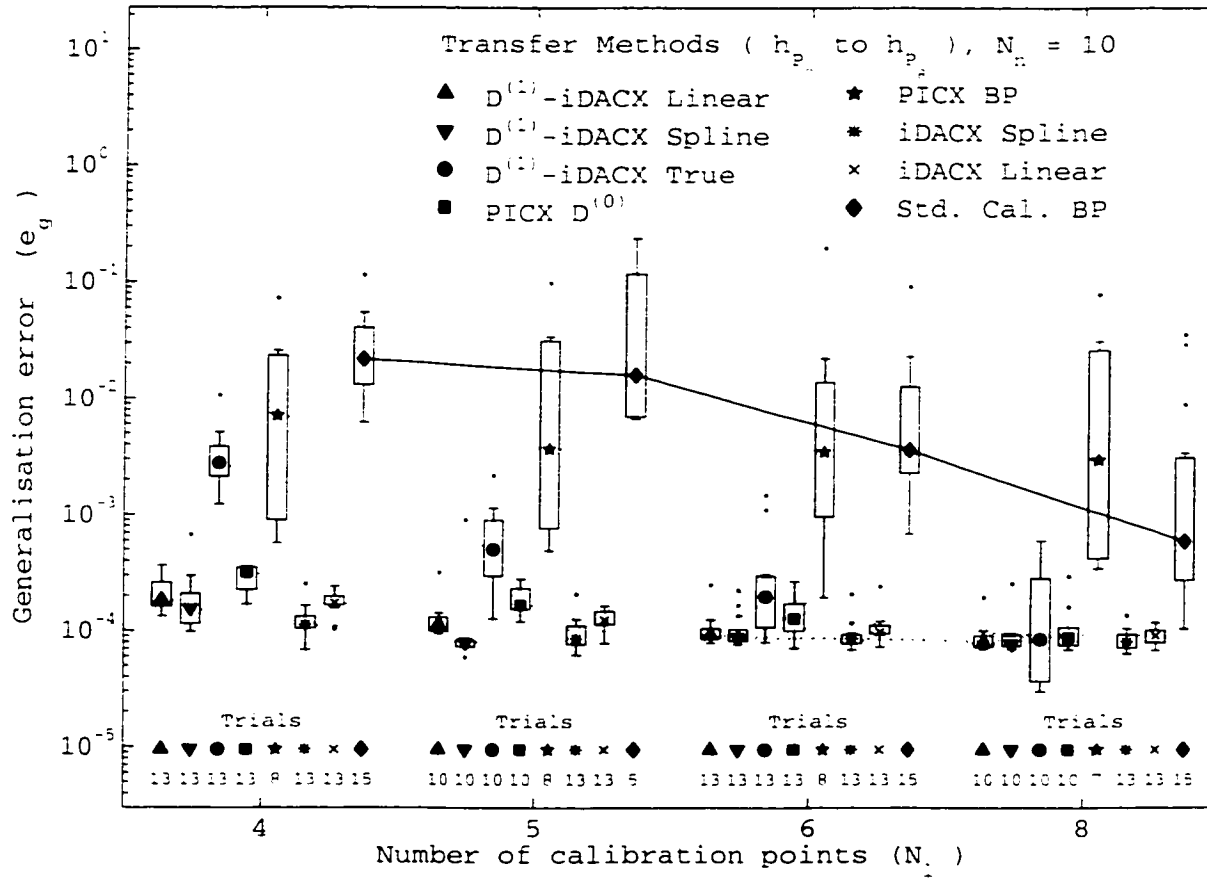
Figure 5.24. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_7}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{sb}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	630	**	0.00041	850	***	< 0.0001	240	**	0.00017	45
$D^{(1)}$-iDACX Spline	***	< 0.0001	370	*	0.0024	550	***	< 0.0001	310	**	0.00050	31
$D^{(1)}$-iDACX True	**	0.00093					**	0.00042	72			
PICX $D^{(0)}$	***	< 0.0001	240	+	0.0074	380	**	0.00054	112	***	< 0.0001	46
PICX BP												
iDACX Spline	***	< 0.0001	1170	***	< 0.0001	1810	***	< 0.0001	430	***	< 0.0001	68
iDACX Linear	***	< 0.0001	490	***	< 0.0001	860	***	< 0.0001	260	***	< 0.0001	89

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

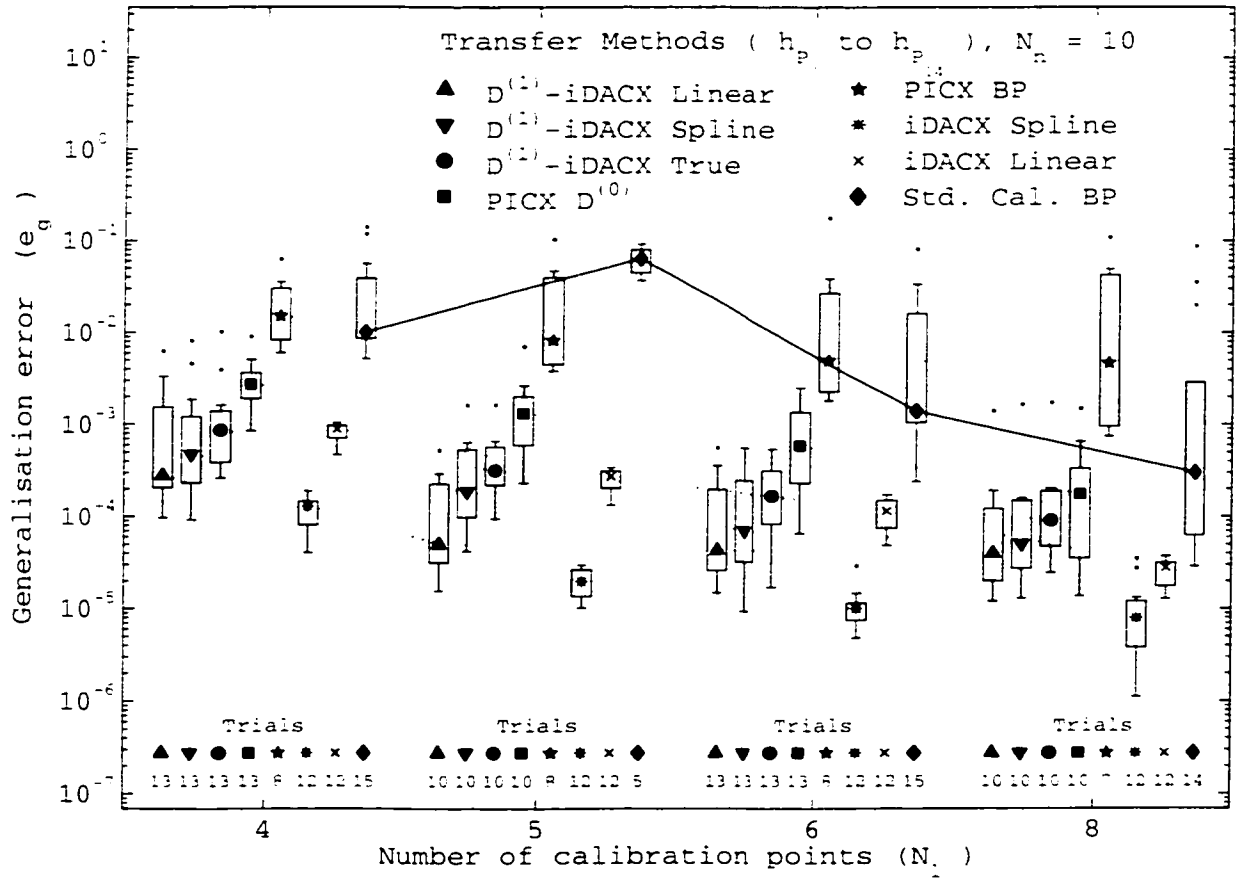
Figure 5.25. The calibration errors, as measured by $\hat{\epsilon}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_i}$ to $\hat{h}_l = \hat{h}_{P_s}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 10$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{10a}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	118	**	0.00057	141	***	< 0.0001	39	***	< 0.0001	7.3
$D^{(1)}$-iDACX Spline	***	< 0.0001	140	***	< 0.0001	200	***	< 0.0001	41	***	< 0.0001	7.6
$D^{(1)}$-iDACX True							+	0.0079	18.2	**	0.00025	6.9
PICX $D^{(0)}$	**	0.00036	69				**	0.00021	28	**	0.00059	6.7
PICX BP												
iDACX Spline	***	< 0.0001	196	***	< 0.0001	185	***	< 0.0001	42	***	< 0.0001	7.2
iDACX Linear	***	< 0.0001	126	**	0.00057	129	***	< 0.0001	37	**	0.00023	6.5

Note : The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

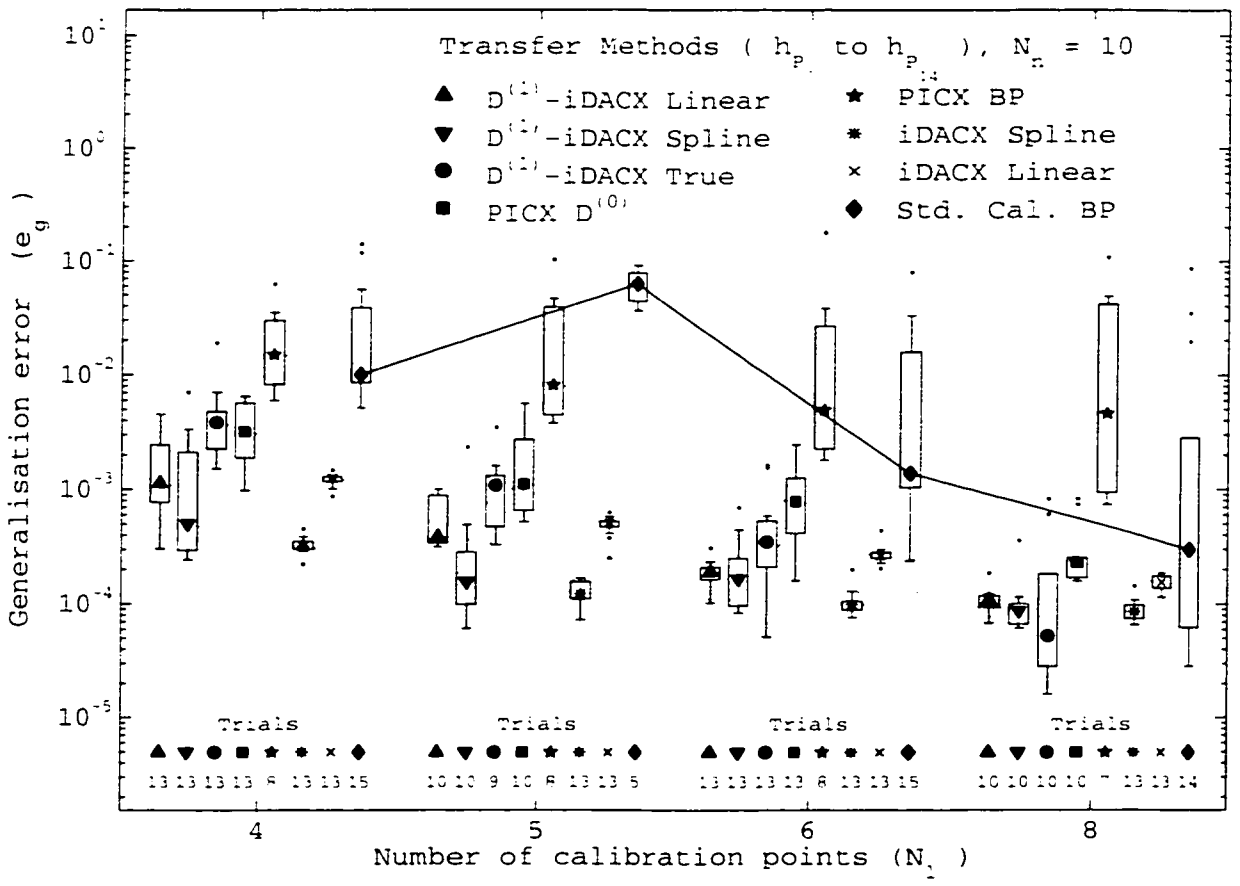
Figure 5.26. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\dot{h}_k = \dot{h}_{P_7}$ to $\dot{h}_l = \dot{h}_{P_8}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 10$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\dot{h}_k \in \mathcal{H}_{P_{10b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	36	***	< 0.0001	1280	***	< 0.0001	32			
$D^{(1)}$-iDACX Spline	***	< 0.0001	21	*	0.0017	340	***	< 0.0001	20			
$D^{(1)}$-iDACX True	***	0.00011	11.6	+	0.0073	200	**	0.00013	8.3			
PICX $D^{(0)}$												
PICX BP												
iDACX Spline	***	< 0.0001	78	***	< 0.0001	3270	***	< 0.0001	138	***	< 0.0001	38
iDACX Linear	***	< 0.0001	11.3	*	0.0018	230	***	0.00014	12.2	**	0.00015	10.8

Note: The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

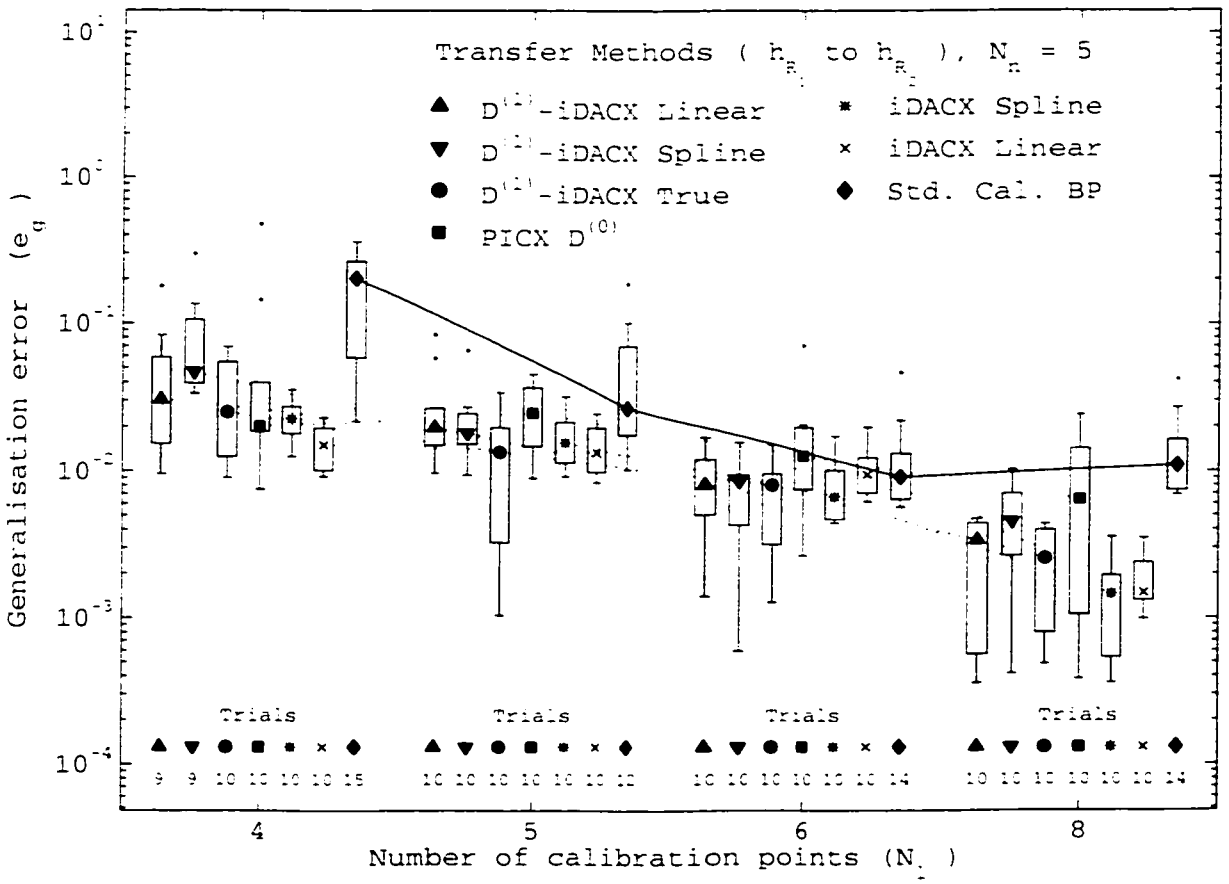
Figure 5.27. The calibration errors, as measured by $\hat{\epsilon}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_7}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 10$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{10_d}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	8.8	*	0.0023	164	***	< 0.0001	7.2			
$D^{(1)}$-iDACX Spline	***	< 0.0001	20	***	< 0.0001	410	***	< 0.0001	8.3			
$D^{(1)}$-iDACX True							*	0.0050	3.9			
PICX $D^{(0)}$	-	0.0078	3.1									
PICX BP												
iDACX Spline	***	< 0.0001	32	***	< 0.0001	520	***	< 0.0001	13.9	-	0.0039	3.5
iDACX Linear	***	< 0.0001	8.0	*	0.0017	125	*	0.0032	5.2			

Note The significance level, S, is adjusted for seven comparisons using the Bonferroni-Dunn method.

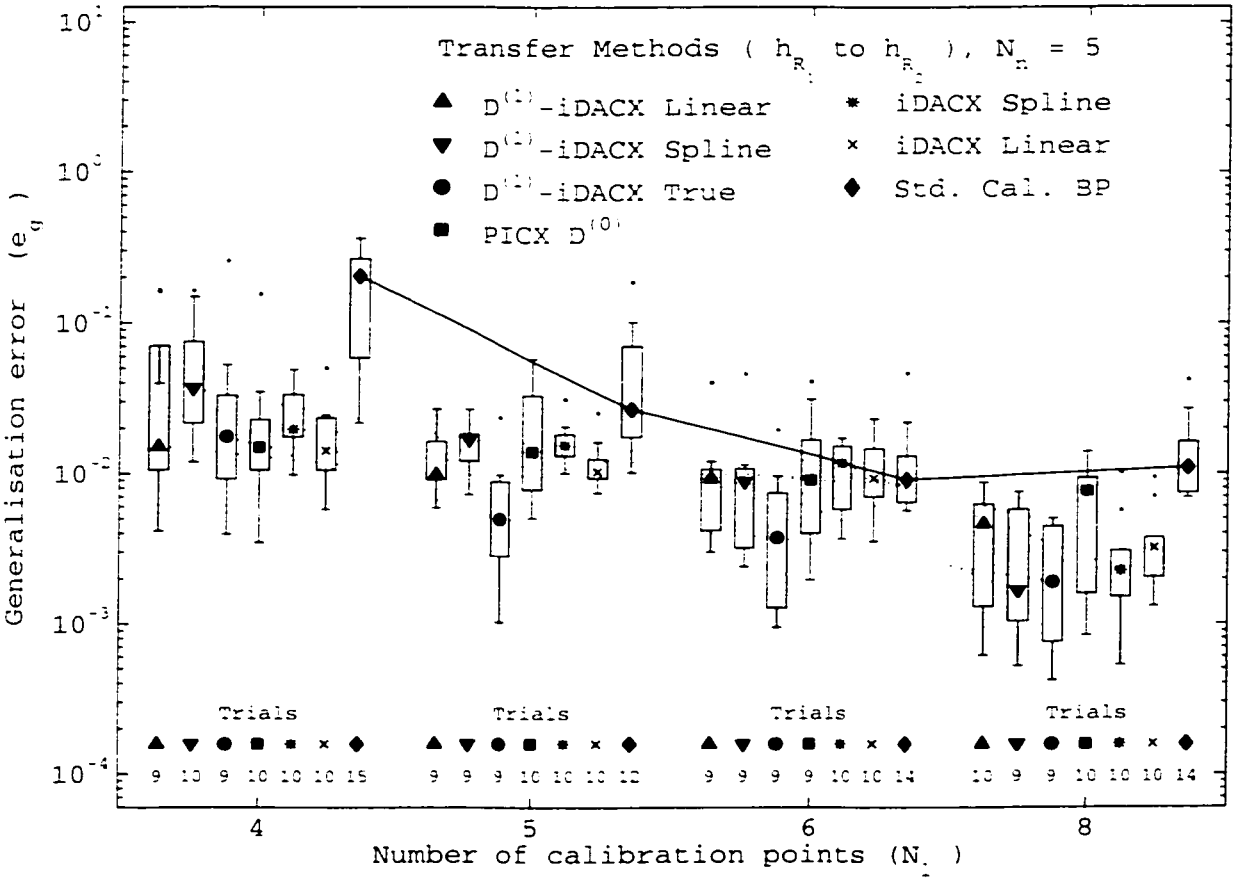
Figure 5.28. The calibration errors, as measured by $\hat{\epsilon}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{P_7}$ to $\hat{h}_l = \hat{h}_{P_{14}}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 10$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{P_{10b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_t = 4			at N_t = 5			at N_t = 6			at N_t = 8		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	+	0.0088	6.7							**	0.00034	3.2
$D^{(1)}$-iDACX Spline										+	0.011	2.4
$D^{(1)}$-iDACX True	**	0.00078	8.1	*	0.0025	2.0				***	0.00010	4.3
PICX $D^{(0)}$	*	0.0052	10.1									
iDACX Spline	**	0.00033	9.0							***	< 0.0001	7.5
iDACX Linear	***	< 0.0001	13.5	*	0.0045	2.0				***	< 0.0001	7.3

Note : The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method

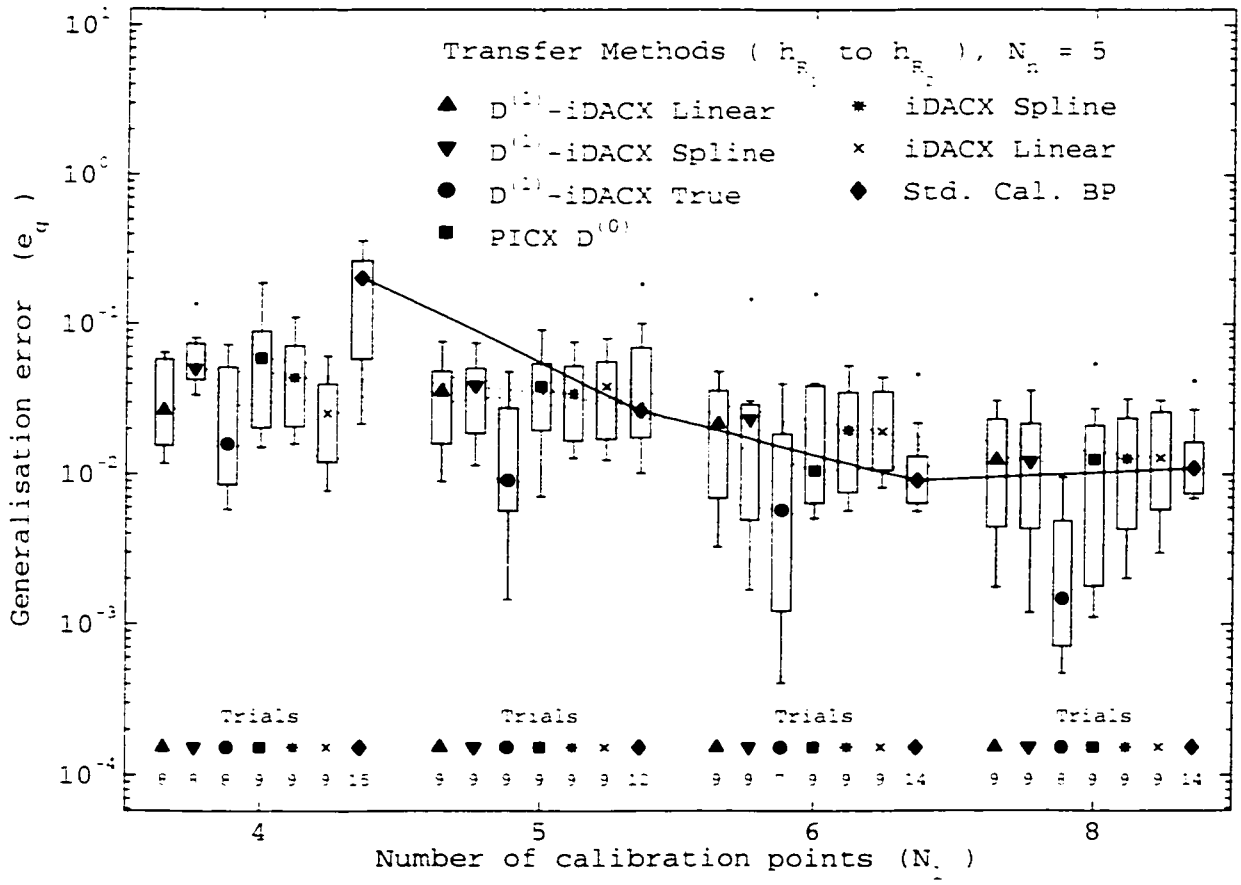
Figure 5.29. The calibration errors, as measured by $\dot{e}_g$, in performing a calibration transfer from $\dot{h}_k = \dot{h}_{R_1}$ to $\dot{h}_l = \dot{h}_{R_2}$ are shown as a function of the number of calibration data points, N_t, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\dot{h}_k \in \mathcal{H}_{R_{5a}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_l = 4			at N_l = 5			at N_l = 6			at N_l = 8		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	**	0.00078	13.4	*	0.0026	2.7				**	0.00013	2.5
$D^{(1)}$-iDACX Spline										***	< 0.0001	6.6
$D^{(1)}$-iDACX True	**	0.00039	11.5	***	< 0.0001	5.4				***	< 0.0001	5.8
PICX $D^{(0)}$	***	< 0.0001	13.6									
iDACX Spline	**	0.00012	10.4							***	< 0.0001	4.8
iDACX Linear	***	< 0.0001	14.3	**	0.00097	2.6				**	0.00011	3.4

Note: The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method.

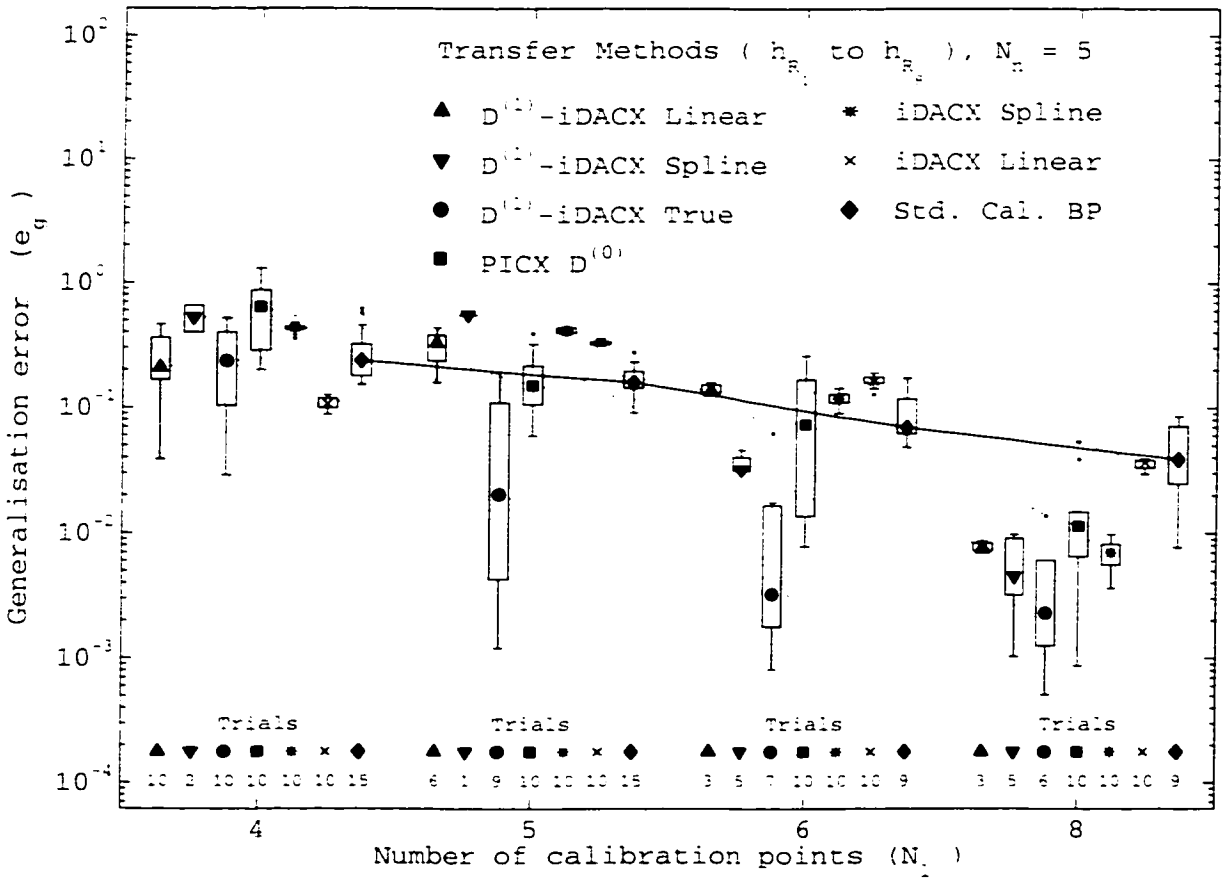
Figure 5.30. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{5b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (–) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_l = 4			at N_l = 5			at N_l = 6			at N_l = 8		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	**	0.0015	7.6									
$D^{(1)}$-iDACX Spline												
$D^{(1)}$-iDACX True	***	0.00014	12.8	–	0.015	2.9				**	0.0016	7.4
PICX $D^{(0)}$												
iDACX Spline												
iDACX Linear	***	< 0.0001	8.0									

Note: The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method.

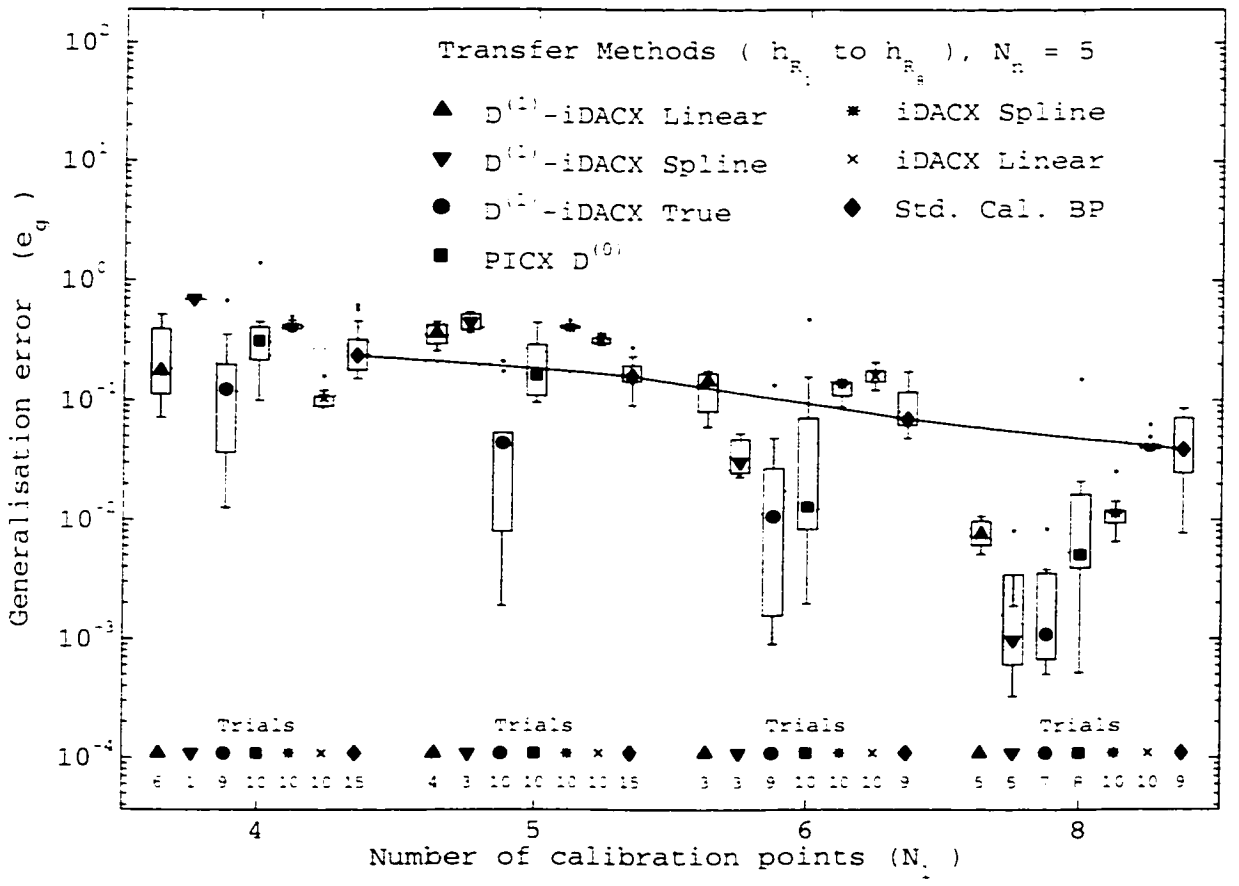
Figure 5.31. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{5c}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_l = 4			at N_l = 5			at N_l = 6			at N_l = 8		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear												
$D^{(1)}$-iDACX Spline										*	0.0026	8.6
$D^{(1)}$-iDACX True							+	0.014	21	***	0.00012	16.8
PICX $D^{(0)}$												
iDACX Spline				***	< 0.0001	0.38				**	0.00074	5.5
iDACX Linear	*	0.0024	2.2	+	0.089	0.48	+	0.0090	0.43			

Note: The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method

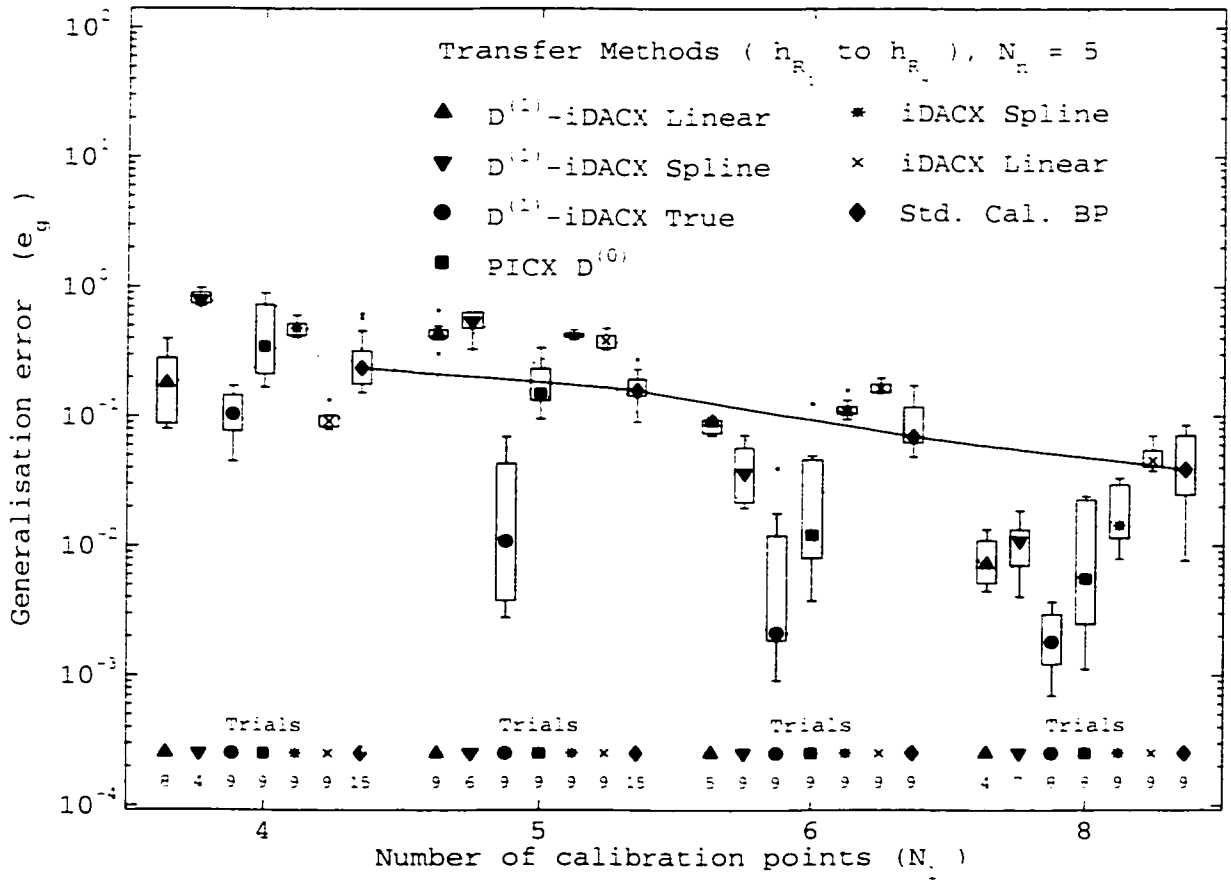
Figure 5.32. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $h_k = h_{R_1}$ to $h_l = h_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $h_k \in \mathcal{H}_{R_{5a}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear												
$D^{(1)}$-iDACX Spline				*	0.0025	0.36				**	0.00030	40
$D^{(1)}$-iDACX True										***	< 0.0001	35
PICX $D^{(0)}$												
iDACX Spline				***	< 0.0001	0.40						
iDACX Linear	**	0.00099	2.3	+	0.0136	0.49						

Note: The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method

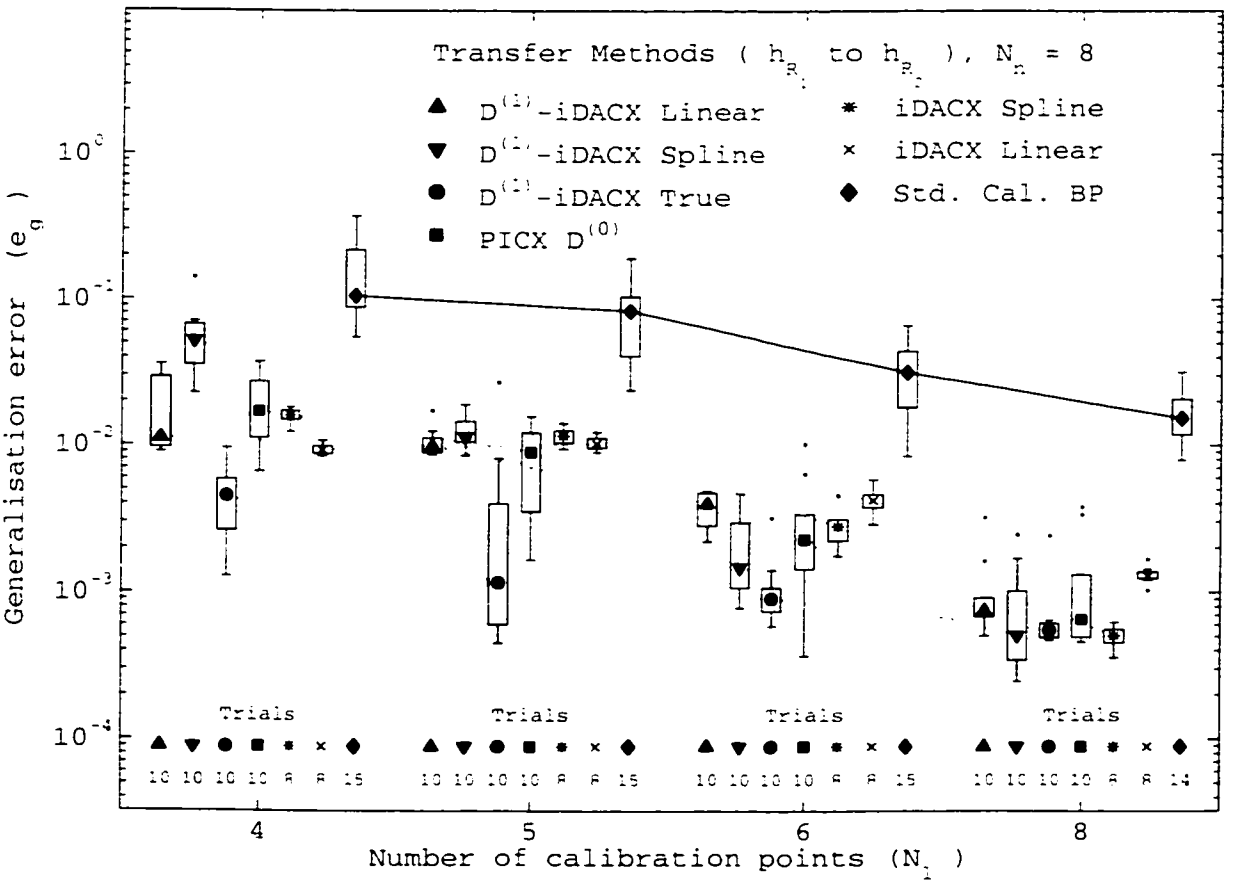
Figure 5.33. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_8}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{5b}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear				**	0.00022	0.37						
$D^{(1)}$-iDACX Spline	+	0.0165	0.30	***	< 0.0001	0.29						
$D^{(1)}$-iDACX True	*	0.0036	2.3				**	0.00052	32	***	< 0.0001	21
PICX $D^{(0)}$										*	0.0047	6.9
iDACX Spline				**	0.00059	0.38						
iDACX Linear	*	0.0017	2.6	*	0.0065	0.42						

Note The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method

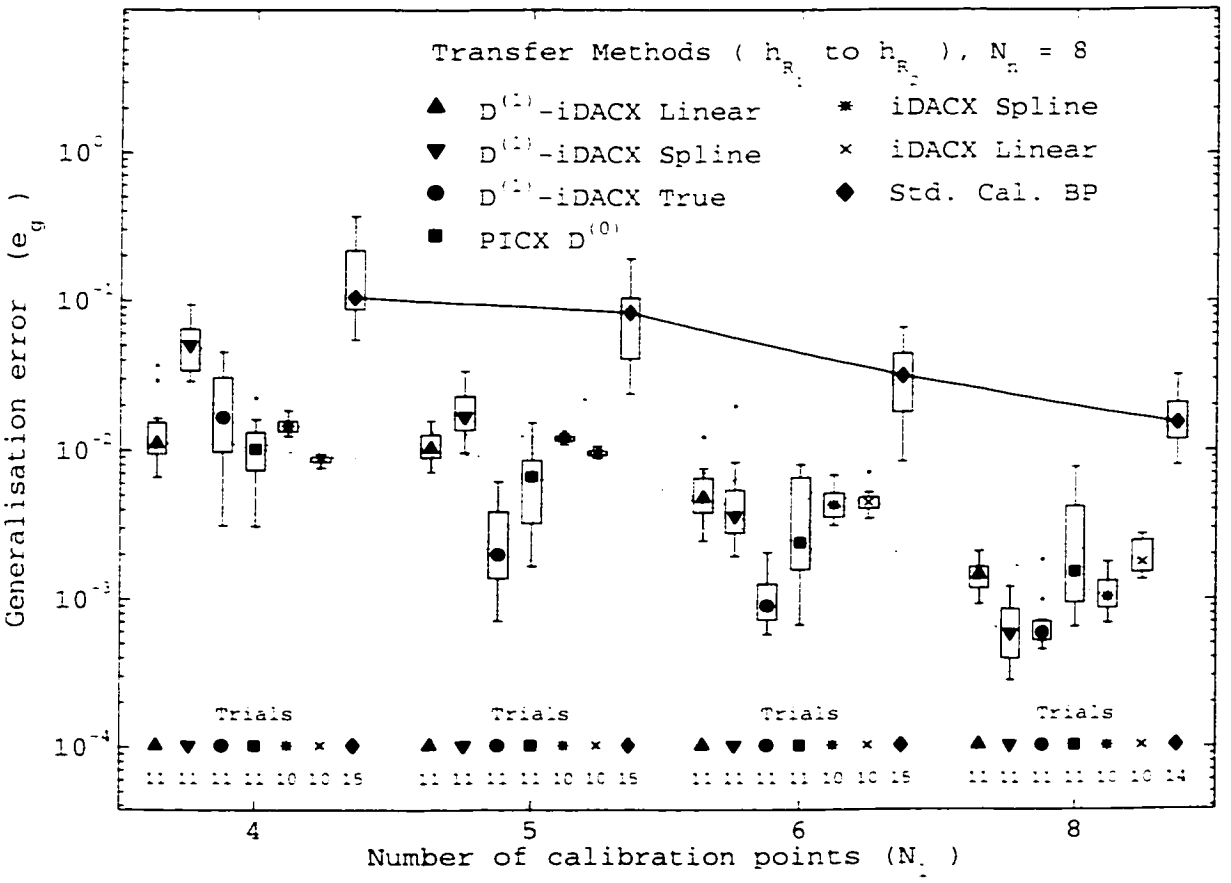
Figure 5.34. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 5$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{5c}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {Blank} = Not significant, (•) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	9.5	***	< 0.0001	8.6	•	0.0042	7.9	**	0.00093	20
$D^{(1)}$-iDACX Spline				•	0.0021	7.3	***	< 0.0001	21	***	< 0.0001	30
$D^{(1)}$-iDACX True	***	< 0.0001	24	***	< 0.0001	72	***	< 0.0001	35	***	< 0.0001	28
PICX $D^{(0)}$	**	0.00031	6.3	***	< 0.0001	9.3	***	< 0.0001	13.9	***	0.00013	23
iDACX Spline	**	0.00099	6.7	•	0.0044	7.2	**	0.00022	11.3	***	< 0.0001	30
iDACX Linear	***	< 0.0001	11.4	**	0.00020	8.2						

Note : The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method.

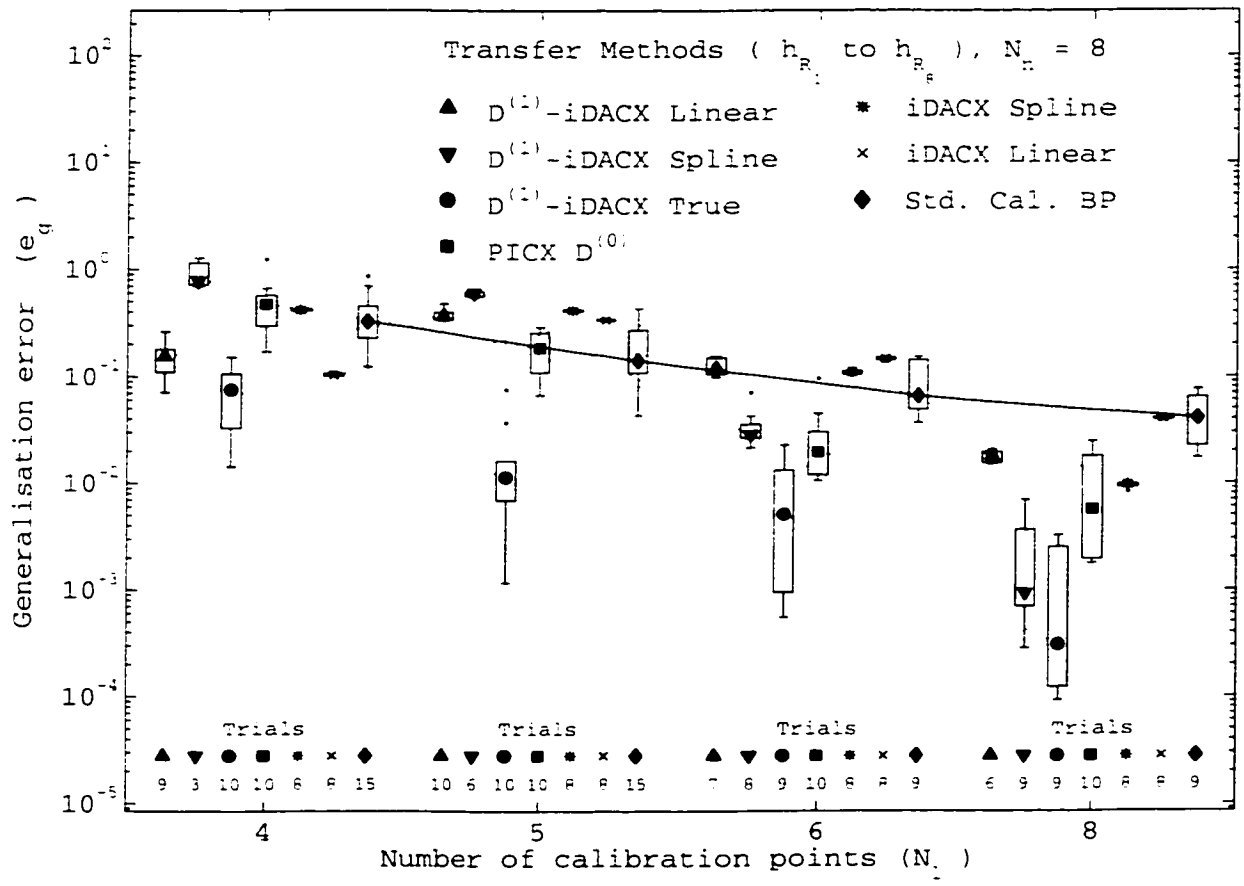
Figure 5.35. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{80}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	***	< 0.0001	9.4	***	< 0.0001	8.1	*	0.0017	6.6	**	0.00087	10.5
$D^{(1)}$-iDACX Spline							***	0.00012	8.7	***	< 0.0001	26
$D^{(1)}$-iDACX True	***	< 0.0001	6.4	***	< 0.0001	42	***	< 0.0001	35	***	< 0.0001	26
PICX $D^{(0)}$	***	< 0.0001	10.3	***	< 0.0001	12.5	***	< 0.0001	13.0	*	0.0022	10.0
iDACX Spline	**	0.00036	7.2	*	0.0039	6.9	**	0.00053	7.4	***	< 0.0001	14.7
iDACX Linear	***	< 0.0001	11.8	***	< 0.0001	8.6	***	0.00085	7.1			

Note: The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method

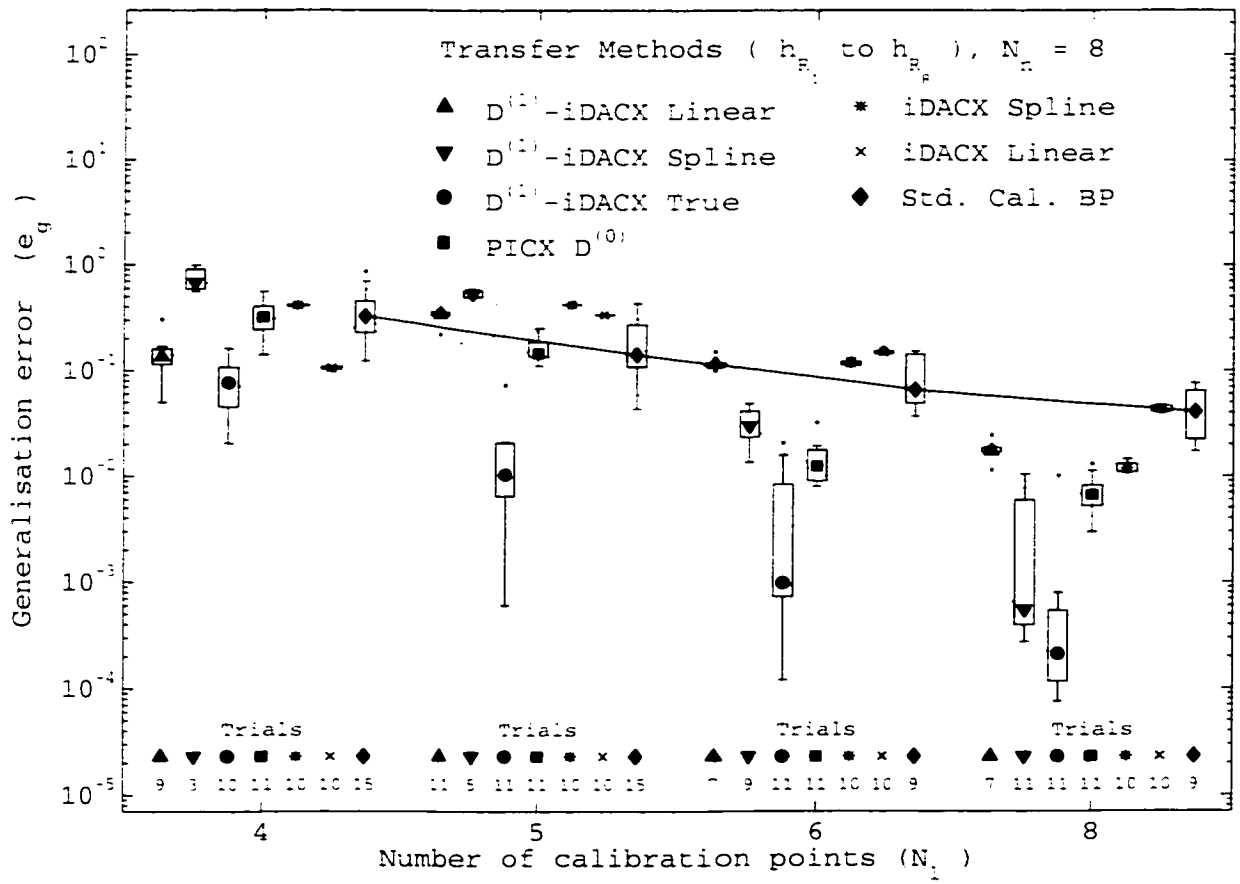
Figure 5.36. The calibration errors, as measured by ϵ_g, in performing a calibration transfer from $h_k = h_{R_1}$ to $h_l = h_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $h_k \in \mathcal{H}_{R_{sb}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear				*	0.0063	0.39						
$D^{(1)}$-iDACX Spline				***	< 0.0001	0.24				***	< 0.0001	44
$D^{(1)}$-iDACX True	***	< 0.0001	4.3	+	0.016	12.5	***	< 0.0001	12.8	***	< 0.0001	132
PICX $D^{(0)}$							*	0.0049	3.3	*	0.0020	7.1
iDACX Spline				**	0.00049	0.34				+	0.0145	4.2
iDACX Linear	**	0.0010	3.2									

Note : The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method.

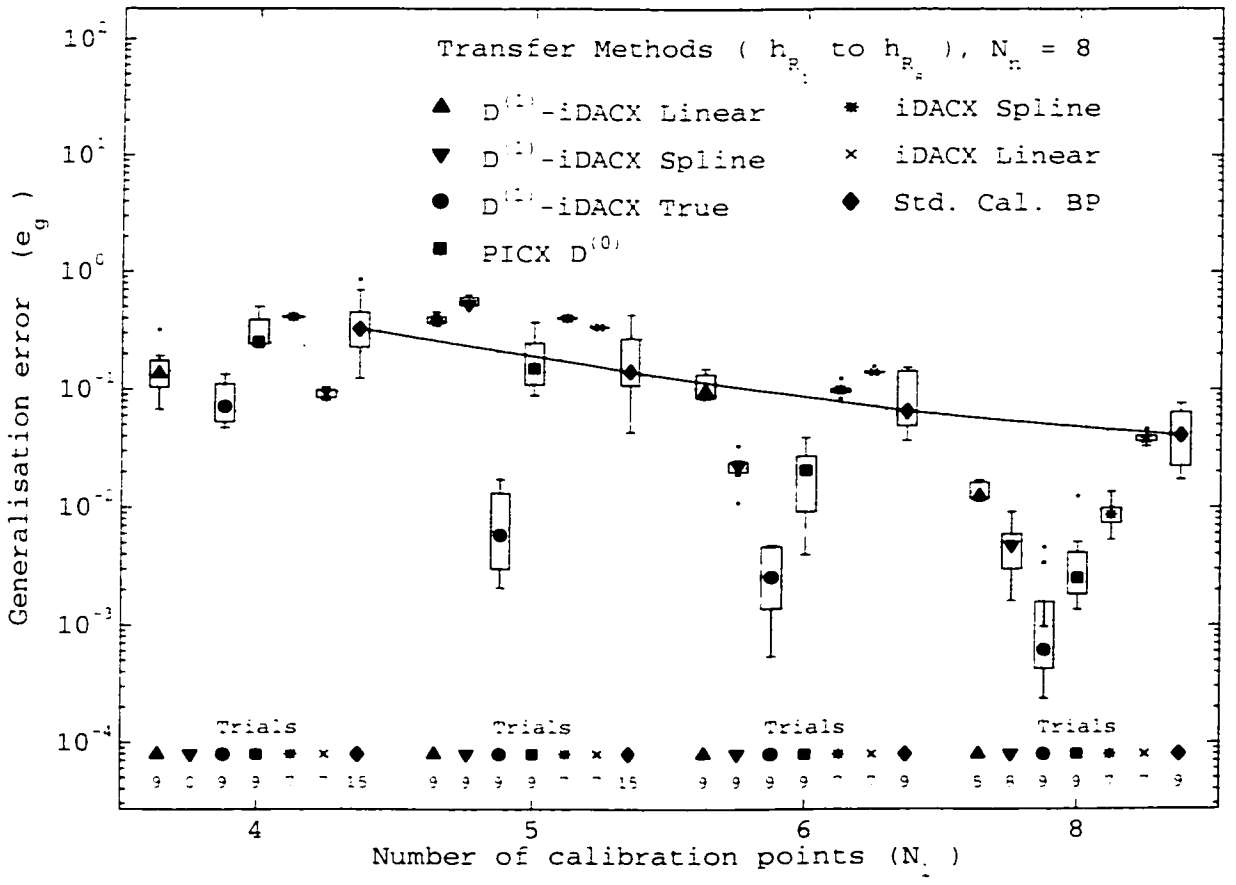
Figure 5.37. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_8}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{8a}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at N_l = 4			at N_l = 5			at N_l = 6			at N_l = 8		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	+	0.013	2.4	+	0.015	0.41						
$D^{(1)}$-iDACX Spline				***	< 0.0001	0.27				***	< 0.0001	74
$D^{(1)}$-iDACX True	***	< 0.0001	4.3	+	0.011	13.7	***	< 0.0001	66	***	< 0.0001	191
PICX $D^{(0)}$							**	0.0013	5.3	**	0.00035	6.1
iDACX Spline				***	< 0.0001	0.34						
iDACX Linear	***	< 0.0001	3.1									

Note: The significance level, S, is adjusted for six comparisons using the Bonferroni-Dunn method.

Figure 5.38. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $h_k = h_{R_1}$ to $h_l = h_{R_8}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $h_k \in \mathcal{H}_{R_{sh}}$.



Median of BP compared to median of	Significance level, S, p values, and Improvement Factor I_F S = {(Blank) = Not significant, (+) < 0.1, (*) < 0.05, (**) < 0.01, (***) < 0.001}											
	at $N_l = 4$			at $N_l = 5$			at $N_l = 6$			at $N_l = 8$		
	S	p values	I_F	S	p values	I_F	S	p values	I_F	S	p values	I_F
$D^{(1)}$-iDACX Linear	*	0.0077	2.4	*	0.0043	0.36						
$D^{(1)}$-iDACX Spline				***	< 0.0001	0.27	+	0.0103	2.9	**	0.00043	8.5
$D^{(1)}$-iDACX True	***	< 0.0001	4.5				***	< 0.0001	26	***	< 0.0001	66
PICX $D^{(0)}$							*	0.0056	3.2	***	< 0.0001	16
iDACX Spline				*	0.0027	0.35						
iDACX Linear	***	0.00015	3.6									

Note: Significance level, S, adjusted for six comparisons, and five for $N_l = 4$, using the Bonferroni-Dunn method

Figure 5.39. The calibration errors, as measured by $\hat{e}_g$, in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ are shown as a function of the number of calibration data points, N_l, for each of the transfer methods. The calibration model consists of a FFNN having $N_n = 8$ hidden neurons that, excluding the PICX BP method, are initialised to values providing $\hat{h}_k \in \mathcal{H}_{R_{SC}}$.

6. Discussion

6.1 Overview

It is apparent from the results shown in Chapter 5 that the calibration transfer methods, when compared to the Std. Cal. BP method, provides a reduction in the error $\hat{e}_g$. Though this reduction in $\hat{e}_g$ occurs under many of the calibration conditions, the simulation results also revealed that the calibration transfer methods do not always provide a significant reduction in $\hat{e}_g$ and under some conditions the methods can actually increase the error over that obtained with the Std. Cal. BP method.

This chapter discusses these results and uses them to provide a series of arguments that not only support the objectives of the thesis but also proposes a set of reasons explaining the instances when the methods do not provide significant improvements in $\hat{e}_g$.

The chapter begins by first discussing the calibration transfer results. The intent of these discussions is to point out that the variability of the simulation results is due to both the characteristics of FFNN supervised learning and the use of a prior calibration model with various degrees of calibration error. The initial calibration results are discussed next. These initial calibration results, though appearing not to be calibration transfer, can be viewed as a special case of a calibration transfer in which there is no prior calibration model to rely on. In this view, the initial calibration results not only provide an indication of the degree of performance, in terms of $\hat{e}_g$, achievable by these methods, but also provide additional insight into the factors that influence the performance of the transfer methods. The limitations of the simulations are then considered and, finally, suggestions for future work are provided.

6.2 Calibration transfer

6.2.1 Summary

The calibration transfer methods do provide the opportunity to improve the calibration error over the standard calibration using backpropagation. These improvement do not occur under every possible calibration condition examined in this thesis. Instead, the calibration conditions under which an improvement can be expected are dependent on a number of factors that include, the nature or complexity of the underlying calibration model, the approximation capability of the FFNN or other approximation method, the degree of similarity between models that are to be transferred, the accuracy of the initial calibration model, and the data set used to perform the transfer.

The simulations highlight how these factors influence the improvement in $\hat{e}_g$ over that of the Std. Cal. BP method. It also becomes clear from these simulations that, because of the complexity of these influences, it is not possible to generalise on the conditions of a calibration transfer that will always provide improvements in $\hat{e}_g$. If any generalisations can be made, they are generalisations that relate to the characteristics of supervised learning in a FFNN. The most important of these generalisations is the need to match the approximation capabilities of the FFNN to the complexity of the underlying function. As shown in the simulations, achieving this match results in increasing the likelihood of improving $\hat{e}_g$ using a calibration transfer.

In place of specific generalisations regarding calibration transfer, the simulation results provide insight into the nature of the influence that a particular factor has on $\hat{e}_g$. In a practical sense, these simulation results, along with the insights gained, suggest that a trial and error approach must be taken when investigating whether a particular transfer method will result in improving $\hat{e}_g$.

Predicting whether a calibration transfer using a FFNN will improve $\hat{e}_g$ is difficult. However, when an improvement in $\hat{e}_g$ is possible, it can be significantly large. These

simulation results have shown that improvements in $\hat{\epsilon}_g$ by a factor as high as 1000 over that of the Std. Cal. BP method are possible using half the number of calibration points of the initial calibration. The potential benefit that these improvements in $\hat{\epsilon}_g$ can provide for an instrument calibration must be weighted against the difficulties imposed in determining whether $\hat{\epsilon}_g$ will improve.

6.2.2 The approach in discussing the results

The discussion regarding the calibration transfer results presents a complex situation in that there are seven calibration transfer methods simulated under more than 100 different calibration conditions. Therefore, to approach these results in a meaningful manner the calibration transfer results are discussed in the following sequence. First, the transfers from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ using $N_n = 5$ and then $N_n = 8$ hidden neurons are considered. These simulated transfers represent the use of models that are very similar to each other, that is, $\forall x_i \in \mathcal{X}$, the estimated correlation is given by, $\hat{R}_{h_R}(\tau = 1, x_i) \approx 0.9$. Next, calibration transfers that occur from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$, again using $N_n = 5$ and then $N_n = 8$ hidden neurons, are discussed. This represents the case of using models that are less similar to each other, that is, there is a low estimated correlation between the models, $\forall x_i \in \mathcal{X}$, $\hat{R}_{h_R}(\tau = 7, x_i) \leq 0.25$.

The calibration transfers using models from $\mathcal{H}_P$ are then discussed in the same sequence as that used to discuss the transfers for models from $\mathcal{H}_R$, that is, first from very similar models using increasing number of neurons to less similar models using increasing number of neurons. It should be noted that the level of detail in the discussion regarding the calibration transfers results using models from $\mathcal{H}_P$ is reduced significantly from the level of detail provided in discussing the calibration transfers results using models from $\mathcal{H}_R$. The reason for this change in the detail of the discussion is simply the consequence of the results of the calibration transfers for models in $\mathcal{H}_P$ exhibiting characteristics that have already been explained and discussed previously. In many instances though, the results of the calibration transfer

for models in $\mathcal{H}_P$ have characteristics that are less accentuated than those provided in the calibration transfer results using models from $\mathcal{H}_R$. This is also the reason for first discussing the calibration transfer results using models from $\mathcal{H}_R$.

To appreciate the meaning of the calibration transfer results, it will be instructive, as noted previously, to examine the first set of simulated calibration transfer results using models from $\mathcal{H}_R$ in detail, after which the remaining results can be interpreted more concisely. In addition, before examining these results it is important to review the context in which the calibration transfers are used in a practical setting.

To consider this context, recall that the intent of any calibration transfer is to attempt to obtain a new calibration model $\hat{h}_l$ using both a prior model $\hat{h}_k$ and data D_l, where $\bar{\bar{D}}_l < \bar{\bar{D}}_k$. It is the goal of the transfer to obtain $\hat{h}_l$ such that its calibration error $\hat{\epsilon}_g$ is either comparable to that of $\hat{h}_k$ or at some other acceptable level. Obtaining $\hat{h}_l$ with comparable or less error than that of $\hat{h}_k$ using less data would be the most desirable outcome of the calibration transfer. More realistically, the calibration transfer method will, with the use of less data, tend to introduce some degree of error over that associated with $\hat{h}_k$. Again, it is the intent of the calibration transfer methods to introduce less error than that resulting from using no calibration transfer.

Therefore, in anticipation of some degree of additional calibration error in $\hat{h}_l$ over that of the prior calibration model $\hat{h}_k$, the model $\hat{h}_k$ can be made highly accurate. Using a highly accurate $\hat{h}_k$ will counter the effect of additional errors being introduced by the calibration transfer method. The end result of this tactic is to provide a final error $\hat{\epsilon}_g$ in $\hat{h}_l$, that though greater than that of $\hat{h}_k$, is still less than what is deemed acceptable to an application.

The calibration transfer results are therefore interpreted within this context, that is, the error in $\hat{h}_l$, after a calibration transfer, is compared to the error that would have been obtained from a standard calibration or recalibration of the instrument. In performing this comparison, it may be the case that the error resulting from the calibration transfer and the standard recalibration are approximately the same. In

this case, there is no point in attempting a transfer since it is clear that the transfer method is either not effectively using the information from a prior calibration model or there is insufficient information from the prior calibration. Therefore, in this case, the tactic in using a highly accurate initial calibration model $\hat{h}_k$, to obtain $\hat{h}_l$ having an acceptable level of error, will not work.

In the case where the calibration transfer method does reduce the error over that of the standard calibration, it is likely that the transfer method is effectively using information from a prior calibration. Whether the degree of reduction of the error has a practical significance depends on the application. For example, if a calibration transfer method provides an improvement in the error by a factor of five over the error provided by the standard calibration, but the standard calibration error is ten times the application's level of acceptable error, then the transfer method does not achieve a practical significance.

Finally, the calibration transfer method may cause the error to increase over that of the standard calibration. In this case, it is apparent that the transfer method is using information that is either insufficient or inaccurate or both.

Given that it is the improvement in the error relative to the error obtained using the standard calibration method that is of importance, the interpretation and discussion of the results is focused towards determining the calibration conditions that tend to produce these improvements.

6.2.3 Calibration transfers from h_{R_1} to h_{R_2} using $N_n = 5$

The first set of calibration results to be discussed are those shown in Figure (5.31) on page 180. Since these results will be considered in more detail than the remaining results, Figure (5.31) is reproduced here and appears in Figure (6.1) for convenience in presentation.

Now recall from the simulation methods, described in chapter 4, that the calibration transfers occur from the initial calibration model $\hat{h}_k$ to $\hat{h}_l$, where for these specific

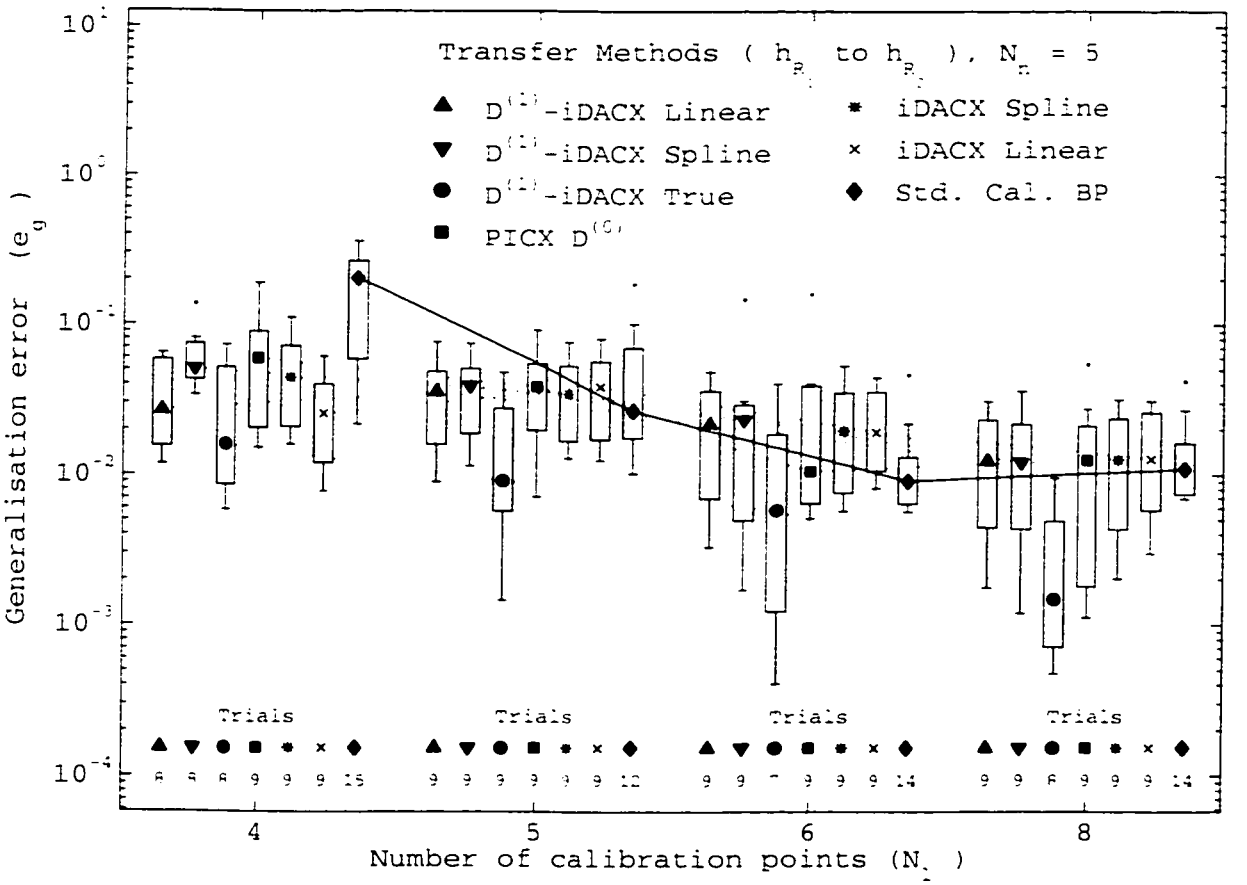


Figure 6.1. The calibration errors in performing a calibration transfer from $\hat{h}_k = \hat{h}_{R_1}$ to $\hat{h}_l = \hat{h}_{R_2}$ as a function of N_l for each of the transfer methods using a FFNN having $N_n = 5$ hidden neurons initialised to values associated with $\hat{h}_k \in \mathcal{H}_{R_{5c}}$.

simulation results $\hat{h}_k = \hat{h}_{R_1}$ and $\hat{h}_l$ is the approximation of $h_l = h_{R_2}$. Consider first the characteristics of the initial calibration models used for these calibration transfer simulations. The median, 25th and 75th percentile values of the error $\hat{e}_g$ of the initial calibration models $\hat{h}_k$ used to obtain the simulations results shown in Figure (6.1), are listed in Table (5.2) as being 1.5×10^{-2}, 2.4×10^{-2} and 5.4×10^{-3}, respectively. These initial calibration models form the set $\mathcal{H}_{R_{5c}}$ having the largest median errors of the sets of initial models used in the simulations for models in $\mathcal{H}_R$. Each of the initial calibration models $\hat{h}_k \in \mathcal{H}_{R_{5c}}$ were obtained with D_k having $N_k = 8$ calibration points using the $D^{(0)}$ learning algorithm. Therefore, given that eight calibration data points are used to obtain the initial calibration model, only eight or less data points are

used in the calibration transfer simulation. The use of more than eight data points no longer meets the spirit of calibration transfer: the intent to find $\hat{h}_l$ with less points with acceptable accuracy.

Now consider $\hat{e}_g$ in $\hat{h}_l$ obtained using the Std. Cal. BP method. that is. the error resulting from a standard recalibration using conventional backpropagation learning. The trend of these errors is shown using a solid line in Figure (6.1). Again. it should be noted that the median errors of all the other transfer methods are compared with the median error resulting from Std. Cal. BP method and that the Std. Cal. BP method can also be viewed as another initial calibration to obtain $\hat{h}_l$. If the median error of the other transfer methods is significantly less than the median error of the Std. Cal. BP method. then the thesis objective is met for those particular set of calibration conditions. The degree of improvement in $\hat{e}_g$. that is. the reduction in $\hat{e}_g$ relative to that of the Std. Cal. BP method. is given by the Improvement factor. I_F. As can be seen in Figure (6.1). the trend of the median error resulting from the Std. Cal. BP method tends to increase as the number of calibration points decreases. This behaviour in the trend is expected from methods based on conventional backpropagation learning.

It should also be noted that the median error in the Std. Cal. BP method at $N_l = 8$. is consistent. that is comparable. with the median error obtained at $N_k = 8$ in the initial calibration simulation results in Figure (5.7) on page 153. that is. both median are approximately 1×10^{-2}. This simply indicates that in setting the data set D_l to be the same as D_k. the BP learning algorithm. in using data sampled from h_{R_2}. finds approximations that have levels of $\hat{e}_g$ that are consistent with approximations obtained using data sampled from any $h \in \mathcal{H}_R$. The reason for this note is that median error for the Std. Cal. BP method at $N_l = 6$ points. which is $\approx 9 \times 10^{-3}$. is not consistent with the median error of 3×10^{-2} at $N_k = 6$ obtained in the initial calibrations simulations using the BP learning algorithm. This anomalous result is due to the inherent variability in finding approximations with supervised learning algorithms given a change in the data set used for learning. In this case. using data sampled from h_{R_2} and data sampled from every $h \in \mathcal{H}_R$ over six sample points

results in a variability in the median error that is larger than that obtained eight sample points. A more detailed discussion on nature and source of this variability in $\hat{e}_q$ is provided in section 3.3.

The comparisons of the median error of $\hat{e}_q$ obtained with the various transfer methods and the Std. Cal. BP method can now be discussed. Also for brevity, a reference to comparisons or differences between medians is always taken to mean a comparison or difference between the median error of a calibration transfer method and the median error in the Std. Cal. method. In addition, reference to a significant difference is defined to be a difference with a statistical significance having $p < S_{0.05}$, where $S_{0.05}$ is the Bonferroni-Dunn adjusted significance level of 0.05.

Now it is clear, from the corresponding table associated with Figure (6.1), listed on page 180, that at $N_l = 8$ calibration points there are, with one exception, no significant differences between the median errors. The one exception is that provided by the $D^{(1)}$-iDACX True method which shows an improvement factor $I_F = 7.4$. This suggests that the transfer methods based on the $D^{(1)}$ learning algorithm, when provided with accurate slope information from h_l and initial weights associated with $\hat{h}_k$, will determine $\hat{h}_l$ that has approximately seven times less error than that obtained with the Std. Cal. BP method. When less accurate slope information is provided, such as that provided in the $D^{(1)}$-iDACX Spline and $D^{(1)}$-iDACX Linear methods, the median error is not significantly different from the median error of the Std. Cal. BP method.

It should also be noted that the slope values used in the $D^{(1)}$-iDACX Spline and $D^{(1)}$-iDACX Linear methods are based on estimating the slope of h_l evaluated at x_i. This slope estimate is given by $d\hat{h}_k(x)/dx + d\hat{s}_l(x))/dx$ at $x = x_i$, where $d\hat{s}_l(x)/dx$ is the slope estimate based on the data given by $S_l = \{(x_i, h_l(x_i) - \hat{h}_k(x_i)) : i = 1, 2, \ldots, N_l, x_i \in X_l\}$. The important point to note is that the slope estimates are also dependent on the accuracy of $\hat{h}_k$, which, in this case, is known to be selected from the set having the largest median error. Therefore, using a more accurate $\hat{h}_k$ should

improve the slope estimate and therefore the resulting error in the approximation $\hat{h}_l$. This observation is supported by the simulation results that use a more accurate $\hat{h}_k$ selected from sets having lower median errors, that is seen in Figure (5.29) and Figure (5.30) on pages 178 and 179, respectively. In using a more accurate initial calibration model $\hat{h}_k$, all the calibration transfer methods, except PICX $D^{(0)}$, have median errors that become significantly reduced.

The iDACX transfer methods are also dependent on the accuracy of $\hat{h}_k$. In these methods $\hat{h}_l$ is given by $\hat{h}_k + \hat{s}_l$, where $\hat{s}_l$ is based on a cubic spline or piecewise linear approximation model whose parameters are estimated using $S_l = \{(x_i, h_l(x_i) - \hat{h}_k(x_i)) : i = 1, 2, \ldots, N_l, x_i \in X_l\}$. Again, increasing the accuracy of $\hat{h}_k$ will improve the accuracy of $\hat{h}_l$, which is shown to occur with simulation results using a more accurate $\hat{h}_k$. Again, see Figure (5.29) and Figure (5.30) on pages 178 and 179, respectively.

Finally, the PICX $D^{(0)}$ method, like the Std. Cal. BP method, is based on a gradient descent search technique over an error surface defined by the sum squared measure $\sum_i^{N_l} (h_l(x_i) - \hat{h}_l(x_i))^2$, for $(x_i, h_l(x_i)) \in D_l$. The only differences between the PICX $D^{(0)}$ method and Std. Cal. BP method are in the initialisation of the weights and the use of a gradient with four components in $D^{(0)}$ learning instead of the two associated with the weights and bias terms in BP learning.

Given that the PICX $D^{(0)}$ method initialises the weights to the values associated with the approximation $\hat{h}_k$, it would seem that improving the accuracy of the approximation $\hat{h}_k$ should also improve the accuracy of $\hat{h}_k$. Unlike the iDACX methods, it is difficult to predict in any specific instance whether the act of increasing the accuracy of $\hat{h}_k$ will increase the accuracy of $\hat{h}_l$. The difficulty in determining the course of this action rests in the difficulty in determining whether increasing the accuracy of $\hat{h}_k$ will correspond to a set weights that provide a location on the error surface that is closer to a minimum representing a more accurate approximation of h_l. This difficulty, as noted previously, is due to the variability observed in $\hat{e}_g$ that results from various attempts with supervised learning to find an approximation $\hat{h}_l$ given different

sets of initial weight values. This variability in ϵ_g that is associated with the PICX $D^{(0)}$ method is also seen in the other transfer methods where, in spite of increasing the accuracy of $\hat{h}_k$, the resulting approximation $\hat{h}_l$ can exhibit either an increase or decrease in error.

For example, consider the median error resulting from using the $D^{(1)}$-iDACX Spline method with $N_l = 8$, that is a median error of about 1.5×10^{-3} indicated in Figure (5.30) on page 179. Now, if the median error in $\hat{h}_k$ is **increased** by a factor of approximately two, the median error in the $D^{(1)}$-iDACX Spline method with $N_l = 8$ increases to about 1.5×10^{-2}, as seen in Figure (6.1). In contrast, if the median error in $\hat{h}_k$ is now **decreased** by a factor of approximately two, the median error of the $D^{(1)}$-iDACX Spline method with $N_l = 8$ still increases to about 5×10^{-3}, as seen in Figure (5.29) on page 178.

The behaviour of the median error as the accuracy of $\hat{h}_k$ changes may be the result of two influencing factors. The first factor is simply the obvious, that is, the error in $\hat{h}_k$. The second factor is less obvious and may be due to the differences in the cost function and data defining the error surfaces used to obtain $\hat{h}_k$ and $\hat{h}_l$. The most accurate set of initial calibration models are those given in $\mathcal{H}_{R_{5a}}$ that were obtained using $D^{(1)}$ True learning, the next most accurate set of initial calibration models are those in $\mathcal{H}_{R_{5b}}$ obtained using $D^{(1)}$ Spline learning, and finally the least accurate models are those in $\mathcal{H}_{R_{5c}}$ obtained using $D^{(0)}$ learning. Now it seems that if the cost function and data used to perform a calibration transfer to $\hat{h}_l$ are *consistent* with that used to obtain the initial calibration model $\hat{h}_k$, then it is more likely that median error in $\hat{h}_l$ may exhibit less variability than if the cost function and data used to obtain $\hat{h}_k$ and $\hat{h}_l$ are not *consistent*.

To illustrate this point, consider again the median error resulting from using the $D^{(1)}$-iDACX Spline method with $N_l = 8$, that is a median error of about 4.5×10^{-3} indicated in Figure (5.29) on page 178. In this result, $\hat{h}_k \in \mathcal{H}_{R_{5a}}$, was obtained using $D^{(1)}$ True learning and $\hat{h}_l$ was obtained using the $D^{(1)}$-iDACX Spline method which

is based on $D^{(1)}$ Spline learning. Now using a less accurate initial calibration model, $\hat{h}_k \in \mathcal{H}_{R_{sb}}$, for this particular transfer method results in an unexpected decrease in median error to about 2×10^{-3}, as seen in Figure (5.30) on page 179. In this case, the initial calibration model $\hat{h}_k$ was obtained using $D^{(1)}$ Spline learning and $\hat{h}_l$ was obtained using the $D^{(1)}$-iDACX Spline which is also based on $D^{(1)}$ Spline learning. In other words, the cost function and data used to define the error surface are *consistent* with each other, in that the cost functions are the same and the data used for learning are obtained using similar methods. This *consistency* in the cost function and the data appears to have counteracted the tendency of an inaccurate $\hat{h}_k$ to increase the median error in $\hat{h}_l$. This result can be explained by considering that it is the cost function and the data that define the error surface over which learning occurs and as the differences in the error surfaces used to obtain $\hat{h}_k$ and $\hat{h}_l$ increases the more likely it is that the variability in the median error between $\hat{h}_k$ and $\hat{h}_l$ will increase.

Now return back to the simulation results shown in Figure (6.1) and note the trend in the median errors as the number of calibration points is decreased. The first apparent observation is the relative insensitivity of the median errors to changes in the number of calibration points. This insensitivity is seen in all the transfers methods, except for the $D^{(1)}$-iDACX True method, which besides having the lowest level of error, also has a greater sensitivity to changes in the number of calibration points. The relative insensitivity to changes in the number of calibration points suggests that, for these particular group of simulations, the transfer methods are relying more on the information in $\hat{h}_k$ than the information provided in D_l.

This suggestion is supported to a degree by the other simulation results, where the use of a more accurate $\hat{h}_k$, selected from sets having lower median errors, tends to increase the differences between the median errors of the transfer methods and the median error of the Std. Cal. BP method, (compare the median errors shown in Figure (5.29) through Figure (5.31) on pages 178 to 180). This behaviour then suggests a dependence on the information in $\hat{h}_k$ by virtue that the median errors in $\hat{h}_l$ vary noticeably with changes in the accuracy of $\hat{h}_k$. Now consider the apparent

decrease in the dependency of the median errors on D_l. It is clear that the information provided by D_l decreases as the number of points in D_l decreases, yet the transfer methods exhibit relatively small changes in the median error compared to the changes exhibited by the Std. Cal. BP method. A possible explanation for this observation, at least for the $D^{(n)}$ learning based transfer methods, is the apparent decrease in the dependency in D_l may in fact be due to the characteristics of supervised learning. This characteristic allows the possibility that the point used to begin learning on the error surface $C(\mathbf{w}, D_l)$, defined by the weight values associated with $\hat{h}_k$, results in the learning algorithm finding $\hat{h}_l$ that is in a region of the weight space where the error surface defining the true generalisation error ϵ_g tends to be relatively constant.

This relatively small increase in the median errors in the transfer methods, as the number of points in D_l decreases, is important when compared to the larger increase in the median error of the Std. Cal. BP method. This difference in the change in the median errors results in an increase in the number of methods having median errors that are significantly different from the median error of the Std. Cal. BP method. For example, in the simulation results reproduced in Figure (6.1), the $D^{(1)}$-iDACX Linear, $D^{(1)}$-iDACX True, and iDACX Linear all achieve significant differences with improvement factors given by 7.6, 12.8, and 8.0, respectively, with $N_l = 4$, (as listed in the table associated with Figure (6.1), given on page 180). Again, it should be stressed that this improvement factor is relative to the median error provided by the Std. Cal. BP method which has increased by a factor of about 20 from the error it provided at $N_l = 8$ points. Therefore, whether the improvement in the error provided by the calibration transfer is of practical significance will be dependent on the application's level of acceptable error.

It then becomes obvious that in decreasing the median error of the initial calibration model $\hat{h}_k$, which *tends* to decrease the median errors in $\hat{h}_l$ obtained by the calibration transfer methods, will also tend to increase the difference these median errors have with the median error resulting from the Std. Cal. BP method. The result is more significant difference in the medians and as shown in Figure (5.29) and

Figure (5.30) on page 178 and page 179, respectively, this is the case.

It should be noted that the median errors in the sets of these initial calibration models, given by $\hat{h}_k \in \mathcal{H}_{R_{5c}}$, $\hat{h}_k \in \mathcal{H}_{R_{5b}}$ and $\hat{h}_k \in \mathcal{H}_{R_{5a}}$ correspond to median errors that are approximately one half and one quarter of the median error associated with $\hat{h}_k \in \mathcal{H}_{R_{5c}}$. In addition, the interquartile range of the distribution of the errors in these initial calibration models also decreases as the median decreases reflecting a distribution that is more *tightly* located about its median. Specifically, these interquartile ranges are approximately 2×10^{-2} to 9×10^{-3} and 1.5×10^{-3} for the sets $\mathcal{H}_{R_{5c}}$, $\mathcal{H}_{R_{5b}}$, and $\mathcal{H}_{R_{5a}}$, respectively.

The results, shown in Figure (5.29) and Figure (5.30) on pages 178 and 179, respectively, show, as discussed previously, that in spite of a factor of two increase in the median error of the initial calibration models, the resulting median error in $\hat{h}_l$ may in fact decrease, remain comparable the same, or increase. The importance of this observation is that it suggests that even if it were possible to have $\hat{h}_k = h_k$, the resulting errors in $\hat{h}_l$ may not change significantly from that shown in Figure (5.30) on page 179. In other words, though the accuracy of $\hat{h}_k$ does seem to influence the resulting error in $\hat{h}_l$, it may be the location in weight space, as indicated by the values of the weights associated with $\hat{h}_k$, that exerts a greater influence on the resulting error in $\hat{h}_l$. A careful inspection of the results also shows that this suggestion only applies to the $D^{(n)}$-iDACX and PICX methods.

The iDACX methods, which do not use supervised learning to determine $\hat{h}_l$, rely more directly on using $\hat{h}_k$ to help determine $\hat{h}_l$. In fact, because of this more direct reliance on $\hat{h}_k$, it is possible to estimate the minimum $\hat{e}_g$ that can be achieved by the iDACX method for a given data set D_l by using exact knowledge of h_k and h_l that is available for these simulations. This minimum $\hat{e}_g$ is determined by using $\hat{h}_l(x) = h_k(x) + \hat{s}_l$ for the approximation, where $\hat{s}_l$ is based on a cubic spline or a piecewise linear approximation model whose parameters are estimated using $S_{l_1} = \{(x_i, h_l(x_i) - h_k(x_i)) : i = 1, 2, \ldots, N_l, x_i \in X_l\}$. The error $\hat{e}_g$ is then given by $\frac{1}{N_j}\sum_j^{N_j}(h_l(x_j) -$

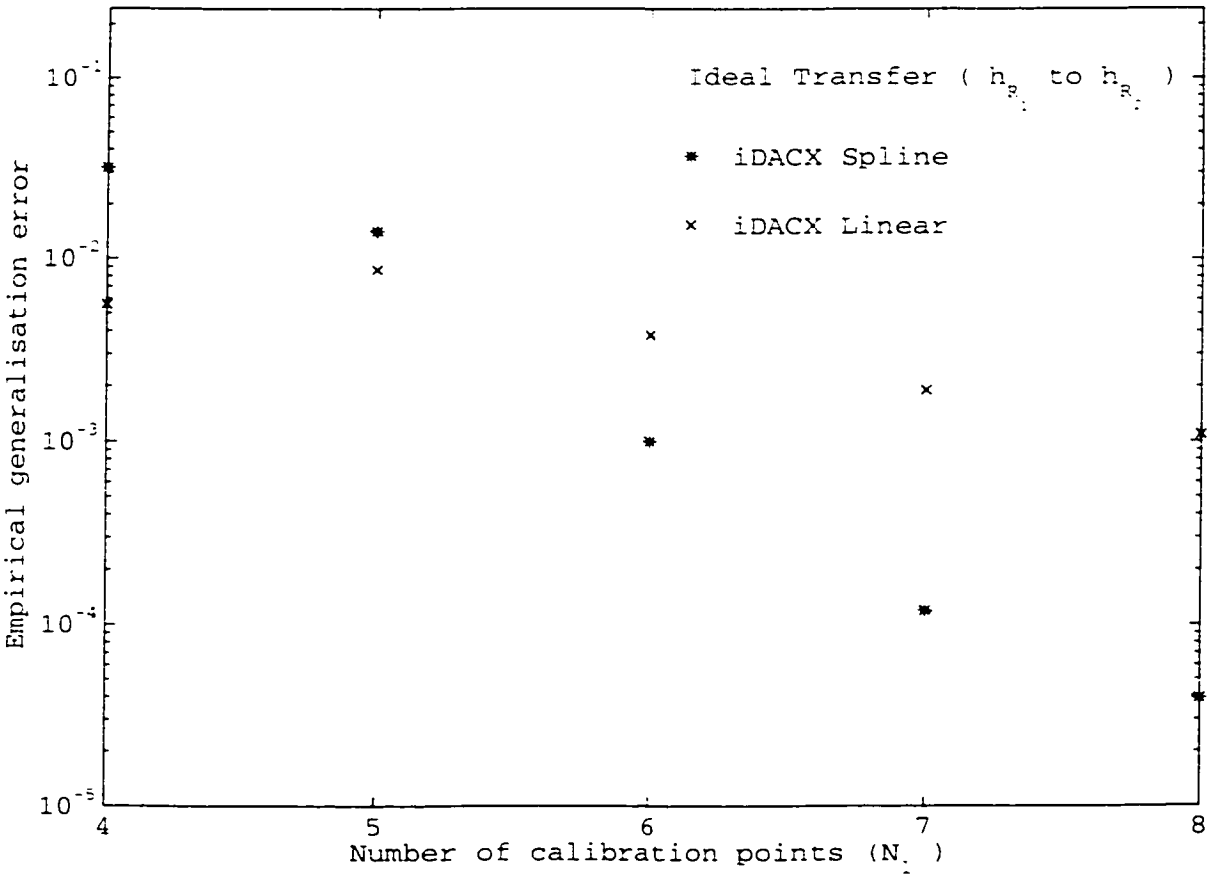


Figure 6.2. The calibration errors, measured by $\hat{e}_g$, in performing an ideal calibration transfer from h_k to $\hat{h}_l$ using $h_k = h_{R_1}$ to $h_l = h_{R_2}$ as a function of N_l for the iDACX Spline and iDACX linear methods.

$\hat{h}_l(x_j))^2$, where $x_j = \frac{15}{(N_j-1)}j$, $j = 0, 1, 2, \ldots, N_j - 1$, and $N_j = 129$. These errors are shown as a function of N_l in Figure (6.2).

It is apparent from Figure (6.2) that the iDACX Spline method can achieve $\hat{e}_g \approx 4 \times 10^{-5}$ at $N_l = 8$ when using an initial calibration model with no errors, that is $\hat{h}_k = h_k$, and the data in D_l. The iDACX Linear method, on the other hand, can only achieve $\hat{e}_g \approx 1 \times 10^{-3}$. If these estimates are compared to the median errors achieved when using $\hat{h}_k = \mathcal{H}_{R_{5a}}$, which from Figure (5.29) on page 178 are approximately 1.5×10^{-3} for both iDACX methods at $N_l = 8$, then it is clear that iDACX Spline method can potentially achieve a 40-fold improvement in the median error if $\hat{h}_k = h_k$ is used to obtain $\hat{h}_l$. The iDACX Linear method, on the other hand, appears to

be at its limit of performance, in that its median error, when using $\hat{h}_k = h_k$, is approximately 1.5×10^{-3}, which is comparable to its limit of 1×10^{-3}. Now consider the use of $\hat{h}_k = \mathcal{H}_{R5c}$, which represents initial calibration models with a median error $\approx 1.5 \times 10^{-2}$, that is also approximately four times that median error associated with $\hat{h}_k = \mathcal{H}_{R5a}$. As seen in Figure (5.31), given on page 180, the resulting median error in $\hat{h}_l$, as expected, increases, but in this case, the median error is also approximately equal to the median error in $\hat{h}_k = \mathcal{H}_{R5c}$. In other words, the error in the initial calibration model appears to dominate the resulting error in $\hat{h}_l$.

It is also clear, that in spite of knowing the limits in $\hat{e}_g$ for the iDAXC methods and the median error in $\hat{h}_k$, this knowledge does appear to be related in an obvious way to the resulting error in $\hat{h}_l$. For instance, note that at $N_l = 4$, the iDACX Linear method has an estimated lower limit in $\hat{e}_g$ of approximately 3×10^{-2}, but Figure (5.29) on page 178 shows at $N_l = 4$, the iDACX linear method achieving errors below this level. This anomalous result is due to the use of $\hat{s}_l$, estimated using a data set, $S_l = \{(x_i, h_l(x_i) - \hat{h}_k(x_i)) : i = 1, 2, \ldots, N_l\}$, having errors that also happens to allow a better approximation to h_l than a data set having no errors, that is, $S_{l_1} = \{(x_i, h_l(x_i) - h_k(x_i)) : i = 1, 2, \ldots, N_l\}$. This result is possible for two reasons. First, it is likely that the coordinates of the data in S_{l_1} are not located at points that provide the minimal error between h_l and h_k. Second the approximation $\hat{h}_k$ may happen to be closer to h_l than h_k. Here minimal error and the meaning of closer are in the sense of the l_2-norm measure of error or distance.

As discussed in section 3.3, many of the characteristics seen in these results are due directly to the fact that the error surface, $C(\mathbf{w}, D_l)$, used for learning is not the same as the error surface, $C(\mathbf{w}, \infty)$, used to measure e_g. Therefore, regardless of the accuracy of $\hat{h}_k$, the weights associated with $\hat{h}_k$, though providing a location on $C(\mathbf{w}, \infty)$ that allows finding minima representing *good* approximations of h_l, may also be providing a location on $C(\mathbf{w}, D_l)$ that is near a set of minima that correspond to poor approximations of h_l on $C(\mathbf{w}, \infty)$.

6.2.4 Calibration transfers from h_{R_1} to h_{R_2} using $N_h = 8$

In these set of calibration transfer simulations the number of hidden neurons in the FFNN is increased to eight from the previous five. The use of additional neurons requires the use of a different set of initial calibration models. These initial models are drawn from three sets $\mathcal{H}_{R_{8a}}$, $\mathcal{H}_{R_{8b}}$, and $\mathcal{H}_{R_{8c}}$. In this case, the sets $\mathcal{H}_{R_{8a}}$ and $\mathcal{H}_{R_{8b}}$ are more similar to each other than were the sets $\mathcal{H}_{R_{5a}}$ and $\mathcal{H}_{R_{5b}}$, that is, the medians, 25th and 75th percentile, and the interquartile range have very similar values as shown in Table (5.2). The only difference between these two sets is that $\mathcal{H}_{R_{8a}}$ was obtained using $D^{(1)}$ True learning and $\mathcal{H}_{R_{8b}}$ was obtained using $D^{(1)}$ Spline learning. The set of initial calibration models defined by $\mathcal{H}_{R_{8c}}$ has a median that is, again, approximately twice that of $\mathcal{H}_{R_{8b}}$ and was obtained using $D^{(0)}$ learning. Overall, the sets of initial calibration models for the eight hidden neuron FFNN have error medians that are approximately five to ten times less than the medians in the sets used with the FFNN having five hidden neurons.

Using the arguments from the previous section, this improvement in the median error of the initial calibration model should not cause much of a change in the median error associated with the various calibration transfer methods. If only five neurons were involved, then this would be a reasonably expectation. Instead, the FFNN now has eight neurons and the most obvious influence this increase in neurons will have on the FFNN is to increase its approximation capabilities.

Before considering the impact of increasing the approximation capabilities of the FFNN on the calibration error resulting from using the calibration transfer methods, consider Figure (5.12) and Figure (5.13) on pages 158 and 159, that show the error $\hat{e}_g$ as a function of the number of hidden neurons. The characteristic to note in these figures is the presence of a minimal point in $\hat{e}_g$. This characteristic is a common feature in approximation methods, regardless of whether the approximation method is based on a FFNN or any other parametric or nonparametric method. For the FFNN, it is known that this optimal point for $\hat{e}_g$ is dependent on number of factors

that include. the specific function that is to be approximated. the number of data points used to obtain that approximation. and the learning algorithm used to find the approximation.

As shown in Figure (5.12) and Figure (5.13). the optimal point for this minimum in $\hat{e}_g$ occurs at about eight neurons over most values of calibration points but only for some of the learning algorithms. The presence of this optimal point at eight hidden neurons introduces another degree of complexity to the behaviour of the errors in the calibration transfer simulations.

Given that a minimal $\hat{e}_g$ exists using eight neurons. the use of more than eight neurons. for any given N_k. or N_l. will tend to increase $\hat{e}_g$. This increase in error is due to the capability of the FFNN to approximate more precisely the data points but at the expense of possibly not approximating the underlying function. This then causes the generalisation error. $\hat{e}_g$. to increase. In contrast. using less than eight neurons will result in the FFNN being less capable of approximating the data points and therefore less able to approximate the underlying function. Again. this will cause $\hat{e}_g$ to increase.

Specifically. for these simulations. this optimal point for $\hat{e}_g$ appears to be present only for methods using the $D^{(1)}$ learning algorithms and occurs at eight neurons. The BP learning algorithm appears to be operating on the increasing tail of the curve representing the behaviour of $\hat{e}_g$ as a function of the number of neurons. These observations suggest that the increase in the number of neurons from five to eight will result in an increase in $\hat{e}_g$ for the Std. Cal. BP method. whereas $\hat{e}_g$ will decrease for the transfer methods based on $D^{(1)}$ learning. The net result will be a further widening of the differences between the error medians. thereby making the transfer methods more attractive to use. Give that the change in the number of neurons appears to only influence the approximation capability of the FFNN. the overall behaviour of the error curves formed by calibration transfer methods. as a function of the calibration points. is not expected to vary significantly from that of the previous section where five neurons where used.

When Figure (5.35) and Figure (5.36). on pages 184 and 185. are compared to Figure (5.29) and Figure (5.30) on pages 178 and 179. this expected behaviour in $\hat{\epsilon}_g$ is seen. The Std. Cal. BP method has an increase in $\hat{\epsilon}_g$ and the calibration transfer methods not only have a reduction in $\hat{\epsilon}_g$ but also possess a more prominent slope indicating a greater dependency on N_l. The existence of a slight difference in the median errors of the various transfer methods. when compared to the median errors resulting from calibration transfer simulations using five neurons. are due to each learning algorithm having slightly different optimal point for $\hat{\epsilon}_g$ as a function of the number of neurons and number of points. This variability can be seen by referring back to Figure (5.12) or Figure (5.13).

One point that needs to be made regarding these simulation results is that the use of two different sets of initial calibration models. having approximately the same error distribution and obtained with different learning approaches. results in a small overall change in the median errors associated with the calibration transfer methods. Specifically. by using the improvement factor as an indication of the change in the median error. the average improvement factor. I_F. over all transfer methods having significant differences in their median error. is approximately 12 for transfers using $\hat{h}_k \in \mathcal{H}_{R_{8a}}$ and approximately 19 for transfers using $\hat{h}_k \in \mathcal{H}_{R_{8b}}$. On the other hand. examining and comparing individual results shows the same behaviour is seen when five hidden neurons were used for the FFNN. that is. the median error is influenced by *consistency* of the costs functions and data associated with obtaining $\hat{h}_k$ and $\hat{h}_l$. (refer to the discussion on page 198).

Though the results from using $\hat{h}_k \in \mathcal{H}_{R_{8c}}$ are not shown. they were consistent with the observed results from the previous section. that is. the improvement factor decreased given increases in the median error of the initial calibration models.

The main observation from these simulations using eight hidden neurons in the FFNN. is that many of the calibration transfer methods now provide a greater degree of improvement in $\hat{\epsilon}_g$ over that provided by the Std. Cal. method. This improvement

is attributed to the better matching of the approximation capabilities of the FFNN using $D^{(1)}$ learning to the underlying model being approximated. This improvement is further accentuated by the corresponding decrease in the matching of the approximation capabilities of the FFNN using BP learning to the underlying model being approximated.

6.2.5 Calibration transfers from h_{R_1} to h_{R_8} using $N_n = \{5, 8\}$

The simulation results using the less similar calibration models at both $N_n = 5$ and $N_n = 8$ are briefly considered. This brevity is imposed by virtue that all the observed behavior in these simulation results can be explained using the arguments presented in the discussion of the previous sections.

As noted earlier, the calibration transfers from $\hat{h}_k = \hat{h}_{R_1}$ to $h_l = h_{R_8}$ represent transfers between less similar calibration models. The result of using a less similar model should therefore present a greater difficulty in performing a calibration transfer. This difficulty arises directly from the greater magnitude of error in estimating either the slopes used in the $D^{(1)}$-iDACX Spline or $D^{(1)}$-iDACX Linear method or $\hat{s}_l$ in the iDACX methods. The decrease in similarity will also present a difficulty for the PICX method in that the location of points on the error surface that correspond to $\hat{h}_k$ and $\hat{h}_l$ will be further apart. In essence, $\hat{h}_k$ will begin to resemble a random starting point that is seen during the initial calibration.

The only transfer method that should have an immunity to this error is the $D^{(1)}$-iDACX True method, which uses the true slope values of h_l at x_i. Of course, it will still need to approximate h_l, which, when given a set of initial weights associated with a less similar $\hat{h}_k$, will begin to resemble an initial calibration attempt.

When Figure (5.32) through Figure (5.34), on pages 181 to 183, are examined, showing the simulation results using five hidden neurons and the initial calibration models defined by the sets $\mathcal{H}_{R_{5a}}$, $\mathcal{H}_{R_{5b}}$, and $\mathcal{H}_{R_{5c}}$, three prominent features surface. First, there exists median errors which are greater, with a statistical significance,

than those of the Std. Cal. BP method. Second, the median error provided by the $D^{(1)}$-iDACX True method is consistently less than the median of the Std. Cal. BP method. Third, that in spite of four-fold increase in the median error of the initial calibration results, there appears to be little change in the overall values of the median errors.

Consider first, the existence of median errors greater than the Std. Cal. BP method. In this case, it is clear that inaccuracy in the slope estimates for h_l at x_l has resulted in the $D^{(1)}$-iDACX Spline and $D^{(1)}$-iDACX Linear methods having median errors that are greater than the median error of the Std. Cal. BP. It is also clear that since the accuracy of $\hat{s}_l$ is dependent on the number of points used to obtain its approximation, that the slope estimates, which are based on $\hat{s}_l$ will also become less accurate as the number of data points from h_l is reduced. Alternatively, increasing the number of data points provides a more accurate slope estimate and, as seen in Figure (5.32) through Figure (5.34), both the $D^{(1)}$-iDACX Spline and $D^{(1)}$-iDACX Linear methods begin to show improvements in the median error as the number of data points is increased. These particular results show that additional *misinformation*, as provided by the slope estimate having errors, can produce approximations that have very poor accuracy. In these cases it would be better to disregard the slope information or at the very least reduce its importance in learning.

The iDACX methods also show median errors greater than the median error of the Std. Cal. BP method, which in this case, is due to both the error in the initial calibration model $\hat{h}_k$ and the greater magnitude of the error in the estimate of $\hat{s}_l$. In these methods it is obvious that, as the number of calibration points increases, the error in $\hat{s}_l$ decreases, thereby providing improved results, which is seen to occur in Figure (5.32) through Figure (5.34).

It should also be mentioned that for the $D^{(1)}$-iDACX Spline and $D^{(1)}$-iDACX Linear methods, the number of trials listed has decreased. This is due to the transfer methods not being able to find an approximation meeting the sum squared error

criteria used to select the successful trials. In particular, if the calibration transfer methods fail to obtain approximations to the data sets having a sum squared error, SSE, that is less than approximately 0.06 for the data values and 1.25 for the slope values, the simulation trial is deemed unsuccessful. Therefore, it is not surprising when given large errors in the slope estimates, that many of the trials did not meet these SSE criteria. Consider that slope values, if inaccurate, may actually increase the apparent complexity of the underlying function. In other words, given only five hidden neurons, it may not be possible to approximate both the data values and arbitrary, that is, incorrect slope values. In a sense, this lack of learning success is hinting at the possibilities that either the network does not have the approximation capabilities to fit both the given data and slope values or that the data or slope or both values are incorrect.

The remaining prominent features from these simulation results, that is, results showing $D^{(1)}$-iDACX True method having median errors consistently less than the median of the Std. Cal. BP method and the small overall change in the median errors, given a four-fold increase in the median error of the initial calibration results, can be explained using arguments raised in the previous sections.

The factor which appears have had the largest influence on these simulation results is the increase in the number of neurons. This increase in the number of neurons has resulted in the calibration transfer methods providing a greater improvement over the Std. Cal. BP method. Again, this is due to both the calibration transfer methods operating near their optimum number of neurons and the BP method moving away from its optimal number of neurons. Specifically, the average improvement factor seen using eight neurons, taken over all methods having significant differences in any one figure, varies between 10 to 40. Whereas, in the case of using five neurons the average improvement factor in any one figure varies between an average of 2 to 16.

6.2.6 Calibration transfers for models in $\mathcal{H}_P$

The calibration transfers using models from $\mathcal{H}_P$ are actually more difficult to interpret if the results from using $\mathcal{H}_R$ were not available for prior examination. The reason for this difficulty can be seen in Figure (5.10) and Figure (5.11), on pages 156 and 157, showing $\hat{\epsilon}_g$ as a function of the number of neurons for all the learning methods given various numbers of calibration points. Instead of exhibiting a clear optimal point for the number of neurons needed in the approximation, the errors tend not to suggest where on the curve of $\hat{\epsilon}_g$ versus N_n the simulations are operating over. The only consistent indication is that of BP learning showing that as the number of neurons increases from five, it becomes less optimal. The $D^{(1)}$ learning algorithms either show little change in the behaviour of $\hat{\epsilon}_g$ as a function of the number of neurons or show a complex behaviour. Specifically, the methods appear to have more difficulty in approximation using five or ten neurons than with eight and sixteen neurons. This behaviour is particular pronounced at $N_k = 4$ neurons. Again, this behaviour may be due to the specific selection of both the data set used for learning and the selection of the particular set of models from $\mathcal{H}_P$ that were used in the simulations.

All the characteristic behaviours seen in the simulation results using models from $\mathcal{H}_R$ are present in these simulations. The extent of any particular behaviour is of course, different. The most prominent difference is the degree of improvement that is seen to be provided by the calibration transfer methods. Whereas in the case of using models from $\mathcal{H}_R$, the average improvement factor varied from about 2 to 40, the improvement factor seen in this group of simulations is dramatically greater, varying from 2 to 1800 for any particular learning algorithm and set of calibration conditions.

This dramatic increase in the improvement factor may be attributed to the models in $\mathcal{H}_P$ being less *complex* than those in $\mathcal{H}_R$. This *complexity* is relative to the approximation capability of the FFNN, thus for the FFNN with the given number of neurons used in this thesis, the models in $\mathcal{H}_P$ appear to be approximated more accurately than models in $\mathcal{H}_R$. This statement is supported by the initial calibration

simulation results which show that the learning algorithm all tend to find approximation of models in $\mathcal{H}_P$ that have less calibration errors than the approximation found for models in $\mathcal{H}_R$. (compare for example, Figure (5.1) on page 147 with Figure (5.7) on page 153). In other words, what this result is also suggesting, is that if the approximation capabilities of the FFNN are matched to the *complexity* of the underlying function, then the calibration transfer methods will more likely provide lower levels of calibration error as compared to the error provided by the Std. Cal. BP method.

Another prominent difference between the simulations involving $\mathcal{H}_R$ and these results is the consistent improvement in $\hat{e}_g$ provided by iDACX methods over that of the Std. Cal. BP method. In many cases, this improvement is much greater than the improvements provided by the other calibration transfer methods. One reason for this result is that the difference between the $\hat{h}_k$ and h_l is essentially another eight order polynomial that may be approximated very well with a lower order polynomial or with cubic splines or with piecewise linear segments. This, is of course, what the iDACX methods perform. In practical terms, these results suggest that calibration transfer cases exist where the FFNN approach does not always provide the better solution.

Finally, another difference that needs to be considered is the use of the PICX BP method for models from $\mathcal{H}_P$ which was not used for models from $\mathcal{H}_R$. The reason for not showing or listing the results from the PICX BP method, when used with models selected from $\mathcal{H}_R$, is that the PICX BP method consistently failed to find approximations, $\hat{h}_l$, that had median errors that were significantly different from the Std. Cal. BP method. One reason that may have contributed to this failure is the choice of using initial calibration models that were obtained only from the BP learning algorithm and not from either $\mathcal{H}_R$ or $\mathcal{H}_P$. The initial calibration models used for the PICX BP method had median errors that are significantly larger than those from either $\mathcal{H}_R$ or $\mathcal{H}_P$. As discussed previously, increasing the error in the initial calibration model results in the initial value of the weights appearing more random. In other words, as the error in the initial calibration model increases, the behaviour

of the calibration transfer method approaches that of an initial calibration. It is then not surprising that the PICX BP method, using initial calibration models with large errors, resulted in median errors that were not significantly different than the median error associated with the Std. Cal. BP method.

6.2.7 Summary of calibration transfer simulations

In summary, the calibration transfer methods can provide a reduction in ϵ_g given classes of functions from either $\mathcal{H}_R$ or $\mathcal{H}_P$. The reductions seen, when using models from $\mathcal{H}_P$, are also much greater than the reductions seen in calibration transfers using models from $\mathcal{H}_R$. This increase in the degree of reduction in ϵ_g is attributed to the approximation capability of the FFNN and the *complexity* of the models in $\mathcal{H}_P$ being matched to a greater degree than the match that exists between the approximation capability of the FFNN and the *complexity* of the models in $\mathcal{H}_R$. The importance of matching the approximating capability of the FFNN with the calibration models is also supported by similar results in using an *optimum* number of neurons, that is, the use of a number of neurons which provide the lowest levels of calibration error. The consequence of using a well match FFNN, with an appropriate calibration data set, is that of increasing the likelihood that a calibration transfer will find an approximation, $\hat{h}_l$ that reduces the calibration error relative to the error resulting from a standard calibration using conventional backpropagation learning.

6.3 Initial calibration

The initial calibration, though not a true calibration transfer, can be viewed as a calibration transfer where it is highly certain that there is no relevant prior calibration model h_k that can be used to assist in obtaining a current model $\hat{h}_l$. In this view, the initial calibration represents a *worst case* scenario for a calibration transfer, in that the methods must disregard any prior information and simply rely on the learning algorithm's capability to find an acceptable approximation given the calibration data.

Overall, the results of the initial calibration simulations showed that all $D^{(n)}$ learning algorithms, for many of the calibration conditions, achieved significantly lower levels of $\hat{e}_g$ than that obtained using BP learning. Equally important, none of the $D^{(n)}$ learning algorithms showed results which were significantly worse than BP learning.

From the results of section 5.2, it is clear that the $D^{(1)}$ True learning algorithm is more likely to find an approximation $\hat{h}_k$ with a lower $\hat{e}_g$ than the BP learning learning algorithm. Specifically, the $D^{(1)}$ True learning algorithm obtained significant differences in the median error in 24 of the 26 different calibration conditions used for $\mathcal{H}_P$ and in all 12 of 12 conditions used for $\mathcal{H}_R$. The degree of improvement in $\hat{e}_g$ over that provided by the BP learning algorithm varied from a factor of 2. ($\mathcal{H}_P$, $N_n = 5$, $N_k = 6$), to as high as a factor of 1300. ($\mathcal{H}_P$, $N_n = 16$, $N_k = 16$). The two remaining cases, at ($\mathcal{H}_P$, $N_n = 8$, $N_k = 4$) and ($\mathcal{H}_P$, $N_n = 16$, $N_k = 4$), which did not show significant differences occurred when the initialisation of the FFNN presented a particularly difficult learning situation. There were no calibration conditions under which the errors were greater than the errors obtained from BP learning.

Both the $D^{(1)}$ Spline and $D^{(1)}$ Linear learning algorithms, though not providing as many instances of significant reduction in the median error over the error associated with BP learning, did show reduction in $\hat{e}_g$ under many calibration conditions. For $D^{(1)}$ Spline learning, improvements were seen in 19 of 26 calibration conditions used for $\mathcal{H}_P$ and in 8 of 12 conditions used for $\mathcal{H}_R$. For $D^{(1)}$ Linear learning, improvements were seen in 16 of 26 calibration conditions used for $\mathcal{H}_P$ and in 7 of 12 conditions used for $\mathcal{H}_R$. Again, there were no calibration conditions under which the errors were greater than the errors obtained from BP learning.

The $D^{(0)}$ learning algorithm showed the least number of reductions in $\hat{e}_g$, achieving significant reductions in only 6 of 26 calibration conditions used for $\mathcal{H}_P$ and 4 of 12 conditions used for $\mathcal{H}_R$. Again, there were no calibration conditions under which the errors were significantly worse than the errors obtained from BP learning.

6.4 Limitations in Interpreting the Simulation Results

Many of the limitations of the simulations were described and discussed in section 4.2. In this section additional concerns regarding the impact that some of these limitations imposed on the simulation are discussed.

One of the first concerns that is raised by these simulation results is whether they are representative enough to provide meaningful insight into the behaviour of the calibration transfer methods. Even though over a hundred different calibration conditions, representing thousands of simulation trials, were considered in these simulations, there is always the presence of the unanswered question of whether the sampling of calibration conditions was sufficient.

The use of two different classes of calibration models, one effectively a random process, does provide some assurance that the behaviour of the calibration transfer methods was captured by the simulations. This observation is supported by the analysis of the simulation results which revealed a consistent and characteristic behaviour in all the calibration transfer methods and learning algorithms as seen with various calibration conditions. In spite of explaining all the significant observed behaviour of the calibration transfer methods, which suggests that the simulation were adequate, it must be acknowledged that it is possible that a subtle and important behaviour exists which was not reveal by the simulations. Of course, this possibility is always present and reflects the nature of attempting to understand any unknown process. For these simulations, it appears that enough calibration conditions were simulated to provide an indication of the nature of factors that influence the calibration error, $\hat{\epsilon}_g$, and to establish that the calibration transfer methods can provide a reduction in $\hat{\epsilon}_g$ relative to that provided by a standard calibration using conventional backpropagation learning.

In terms of factors that could influence the simulation results, it is the practice, used in this thesis, of not stopping the learning at the first indication of over-learning

that needs consideration. This practice is used to prevent the supervised learning algorithm from approaching a minimum on the error surface $C(\mathbf{w}, D)$ that may correspond to a point on $C(\mathbf{w}, \infty)$ that increases the generalisation error.

To implement this idea of preventing over-learning requires estimating the generalisation error during learning, that is, obtaining $\hat{e}_g$. Essentially, the idea is to use $C(\mathbf{w}, D)$ for learning but at the same time obtain information as to the behaviour of $C(\mathbf{w}, \infty)$. In a setting where the number of calibration points are limited the estimation, $\hat{e}_g$, will be difficult and may be less accurate than desired. Therefore, its use, though desirable, may be difficult to implement.

For these simulation results it would have been possible to use an accurate $\hat{e}_g$ during learning to detect when over-learning begins to occur and to then stop the learning algorithm. Although this was not done for these simulations, the effect of doing this would have been to potentially reduce the calibration error resulting from all the simulation trials. Since it is difficult to predict when over-learning occurs, it is difficult to determine how much of an improvement would have been seen. In addition, the importance of this improvement may not have affected the relative performance of the calibration transfer method as it is likely that all the methods would have experienced similar degrees of improvement. Of course, it is possible that one or more methods would have experienced significantly greater improvements than another set of methods.

For these simulations it was deemed more important to see how the methods performed using only termination criteria based on $C(\mathbf{w}, D)$ rather than criteria based on various techniques obtaining estimates of $C(\mathbf{w}, \infty)$. In one sense, using only $C(\mathbf{w}, D)$ to decide when to terminate learning provides a less distorted view of the behaviour of the learning algorithm and the relationship between $C(\mathbf{w}, D)$ and $C(\mathbf{w}, \infty)$. The importance of this view is provided by the realisation that ideally, learning should be using $C(\mathbf{w}, \infty)$, and not $C(\mathbf{w}, D)$, to obtain an approximation with minimal generalisation error. Therefore, providing an unobstructed view of the learning algorithm's

performance using $C(\mathbf{w}, D)$, not only provide an indication on how *close* $C(\mathbf{w}, D)$ is to $C(\mathbf{w}, \infty)$, but also on the variability between $C(\mathbf{w}, D)$ and $C(\mathbf{w}, \infty)$.

It should be noted that for these simulations, knowledge of h_k and h_l was available to allow a very accurate estimate of ϵ_g in a simple manner. This knowledge was only utilised to avoid the error in the estimation of $\hat{\epsilon}_g$ from obscuring the error performance shown by the calibration transfer methods.

Finally, the issue of not directly controlling the slope error or adding error to it needs to be addressed. The issue of data errors was discussed in section 4.2. It is clear from the simulation results that the accuracy of the slope data influences $\hat{\epsilon}_g$ in the $D^{(1)}$-iDACX transfer methods, in that the use of the true slope values tended to provide lower levels of error. It is also clear that the slope estimates with errors, used during the calibration transfer with models that were not similar to each other, resulted in $\hat{\epsilon}_g$ that exceeded that of the Std. Cal. BP method.

This observation regarding the behaviour between the slope error and $\hat{\epsilon}_g$ taken at its face value suggests that as the error in the slope increases from its true value, it will result in an increase in $\hat{\epsilon}_g$. It would then appear to be useful to develop a strategy so as to determine how this increase in slope error influences $\hat{\epsilon}_g$ in order to decide when not to use the slope data or to predict whether a particular slope estimate will prevent achieving a desired level of $\hat{\epsilon}_g$. However this strategy is flawed by virtue that it is currently not even possible to determine the specific level of $\hat{\epsilon}_g$ that will result from using a given set of data D. Therefore attempting to determine the level at which the slope error causes $\hat{\epsilon}_g$ to be unacceptable, will be highly dependent on the data set, the learning algorithm, and the underlying function to be approximated. This argument is also supported by the simulation results that show that in spite of slope error due to the approximations provided by the $D^{(1)}$ Spline or $D^{(1)}$ Linear methods, these methods will actually determine an approximation $\hat{h}_l$ that has less error $\hat{\epsilon}_g$ than the $D^{(1)}$ True method. In other words, in these particular simulations an error in the slope value actually improved $\hat{\epsilon}_g$.

6.5 Suggestions for Future Research

The FFNN calibration transfer methods proposed in this thesis, the subsequent simulation results, and discussion reveals a number of additional avenues that may be investigated in order to improve and better understand the process of calibration transfer in the context of supervised FFNN learning.

6.5.1 Improving the $D^{(n)}$ based calibration transfer methods

The most obvious improvement to the $D^{(n)}$ based calibration transfer methods would be to use learning algorithms based on higher order gradient descent methods such as the Levenberg-Marquardt algorithm. The use of higher order methods will not only speed the learning process, but it should provide a better search strategy and therefore find approximations having values of $\hat{e}_g$ that are lower than those provided by the first order descent methods. The difficulty will be in providing an adaptation of the algorithm that can efficiently incorporate nth order derivative fitting.

Another issue which is important to address is to determine whether the use of higher order derivatives, that is of order $n > 1$, in the supervised learning algorithm would further improve the resulting generalisation or calibration error of the various calibration transfer methods. In this thesis only derivatives to order $n = 1$ were used. The GSS theorem suggests the use of higher order derivatives may further improve the approximation potential of the FFNN, but as noted in the thesis, a FFNN does not use this derivative information as effectively as the ideal interpolation formula in the GSS theorem, so it becomes an open question as to whether higher order derivatives will improve the calibration transfer methods.

Though it was argued that knowledge of the slope error would not help in determining the specific level of $\hat{e}_g$ that would result from a particular learning trial, it may be useful to estimate the slope error and use this information to reduce the weighting factor assigned to the gradients associated with the slope errors. In other words, it may prove useful to use the estimate of the slope error to change the step size of the

slope gradients with respect to the data gradients. This would effectively provide a mechanism of the algorithm to shift from data and slope fitting to only data fitting if the slope estimates appear to have large errors. This would hopefully reduce the likelihood of increasing ϵ_g and provide a more robust algorithm.

The thesis also showed that calibration transfer methods that do not rely on using the FFNN are also possible and that in some cases may actually out perform the FFNN based methods. The extension of the idea of using slope information to help determine an approximation for non-FFNN methods may also prove to be useful. Currently, most approximation methods only use the data values to determine an approximation. The incorporation of nth order derivative fitting in nonparametric approximation methods is not well developed, but as shown in these simulation results, can provide in a significant improvement in the accuracy of an approximation.

Inspite of the thesis discussing the potential impact of adding noise to the data values of the training set, the use of data from real instruments, exhibiting various degrees and types of noise, would provide additional insight into the behaviour of the calibration transfer methods. The interpretation of the results using real data would require a cautionary approach by virtue that the characteristics of both the data and noise would be specific to instruments used to obtain the data. This cautionary approach is imposed by the large variability of the resulting approximation or generalisation error obtained with the calibration transfer methods. Therefore, any generalisation of the behaviour of the calibration transfer methods and the applicability of the methods to other types of instruments must be done carefully.

6.5.2 Additional applications

The thesis focuses on the issue of recalibration of instruments based on the static FFNN. In a more general setting the methods developed in this thesis simply attempt to obtain an approximation $\hat{h}$ of a true model h using a finite set of data, prior knowledge from a previous model, and additional knowledge obtained by estimating

higher order derivative data. It is apparent in this general setting that the methods developed for the recalibation of FFNN based instruments are not the only application that these methods may be used in.

The most obvious application of these methods outside of instrument recalibration occurs in control systems where dynamic neural networks are sometimes used. In this case the adaptability of the neural network can be viewed as a form of recalibration, in that both adaptation and recalibration are attempting to obtain an updated model of the system. The difference, of course, between adaptation and recalibration is that adaptation occurs on a relatively *continuous* basis, whereas, recalibration only occurs at specific instances spread apart by rather large intervals. In addition, the difficulty in applying the methods proposed in the thesis would be in modifying the learning algorithms to include nth order derivative fitting and in determining an appropriate model and estimation technique for the parameters and slopes of $\hat{s}(\mathbf{x})$ or $\hat{\mathbf{s}}(\mathbf{x})$.

Instrument recalibration also has close parallels to system identification or other methods of parametric approximation. In this case recalibration is simply viewed as a process of obtaining a model that may use information from a prior model. Again, the difficulties in applying the methods proposed in this thesis would be in the derivation of expressions that include the use of nth order derivative data and in determining an appropriate model and estimation technique for the parameters and slopes of $\hat{s}(\mathbf{x})$ or $\hat{\mathbf{s}}(\mathbf{x})$.

The benefit in applying the methods proposed in this thesis are the potential to use less data, that is physically obtained data, and the possible improvements in approximation accuracy. These benefits arise directly from the use of the GSSE measure and incorporation of the potential to use information from a prior accurate approximation.

7. Conclusions

This thesis has presented, developed, and investigated a number of previously unavailable methods of calibration transfer for FFNN based instruments. These methods, referred to as $D^{(n)}$-iDACX, iDACX, and PICX, allow the FFNN based instrument to be recalibrated such that for a given number of calibration data, the calibration error can be reduced below that achievable in a standard recalibration employing conventional backpropagation learning, referred to as Std. Cal. BP. Though the calibration transfer methods can provide a reduction in calibration error compared to that provided by a Std. Cal. BP, it is difficult to predict whether a specific set of calibration conditions will result in a reduction in the calibration error.

This difficulty in predicting whether a specific set of calibration conditions will result in a reduction of calibration error results directly from the known difficulty in predicting the generalisation error performance of a specific supervised learning algorithm given a particular set of data, initial weight values, and number of hidden neurons. The calibration transfer methods $D^{(n)}$-iDACX and PICX increase this difficulty in prediction by relying on the prior approximation, that is, the calibration model $\hat{h}_k$, to assist in determining the current approximation, $\hat{h}_l$. In relying on the prior approximation, the existence of various degrees of *similarity* between the underlying models h_k and h_l is the factor that adds to this prediction difficulty.

In practical terms, this difficulty in predicting whether a reduction in calibration error will occur becomes apparent by examining both the variability in the degree of error reduction and its distribution as the set of calibration conditions are changed. Examining the results of the computer simulations comparing the median calibration error of these transfer methods to the median calibration error of the Std. Cal. BP

method support these claims of difficulty. The simulations have shown that not only is it possible to achieve a factor of 2 to 1000 reduction in the calibration error over that of Std. Cal. BP while using half the data of the initial calibration, but that it is also possible to increase the error by as much as a factor of 3 over that of Std. Cal. BP. Each of the calibration transfer methods exhibit this variability, the extent of which is dependent on factors such as the number of hidden neurons, the learning algorithm, the number of calibration data points, the error in the calibration data, the error in approximation of the initial calibration model, the degree of *similarity* between the initial calibration model and the desired model, and the underlying complexity of the calibration models.

For the $D^{(n)}$-iDACX and PICX methods, the governing principle that determines the degree of influence these factors have on the calibration error is given by the principles governing the behaviour of supervised learning in a FFNN. In this context, it is obvious that increasing the quantity of calibration data, as well as its accuracy, will increase the likelihood of the FFNN's learning algorithm finding an acceptable approximation of the desired calibration model. This then translates directly into the increased likelihood of obtaining a reduction in the calibration error.

The iDACX method, though not using a learning algorithm to determine $\hat{h}_l$, still relies on the the FFNN's prior approximation $\hat{h}_k$, as well as the degree of similarity between h_k and h_l, and the calibration data set D_l. Therefore, if $\hat{h}_k$ is accurate, and if the difference between h_l and $\hat{h}_k$ can, with the data D_l, be approximated using the model selected for the iDACX method, then the iDACX methods will perform very well.

Of all the methods considered in this thesis the $D^{(n)}$-iDACX methods incorporates the most information to attempt a calibration transfer. In spite of this, the method does not always provide the most significant reductions in calibration error or even the most improvement in calibration error when compared to the Std. Cal. BP method. The variability displayed by this method is, in part, the result of using in-

accurate information. that is. inaccurate derivative information. However. the nature of supervised learning also plays an important part in this variability. As was seen in the simulation results. the use of perfectly accurate information does not ensure that the method will provide a significant decrease in the calibration error. Alternatively. given inaccurate information the method was seen to provide significant reductions in the calibration error. It was not possible to predict when the method would or would not reduce the calibration error. In spite of this apparent difficulty. when given accurate calibration data. the $D^{(n)}$-iDACX methods tend to provide a reduction in the calibration error relative to the error associated with the Std. Cal. BP method

Taken as a group. all the calibration transfer methods provided in this thesis will allow the FFNN based instrument to be recalibrated such that. for a given number of calibration data. the calibration error can be reduced below that achievable in the Std. Cal. BP method. Therefore. the methods represent novel. viable approaches to calibration transfer for FFNN based instruments that were previously unavailable.

References

[1] Indira S. Adhihetty, Joseph A. McGuire, Boonsri Wangmaneerat, Thomas M. Niemczyk, and David M. Haaland. "Achieving transferable multivariate spectral calibration models: Demonstration with infrared spectra of thin-film dielectrics on silicon." *Analytical Chemistry*, vol. 63, no. 20, pp. 2329–2338, 1991.

[2] Yongdong Wang, David J. Veltkamp, and Bruce Kowalski. "Multivariate instrument standardization." *Analytical Chemistry*, vol. 63, no. 23, pp. 2750–2758, 1991.

[3] Yongdong Wang and Bruce R. Kowalski. "Calibration transfer and measurement stability of near-infrared spectrometers." *Applied Spectroscopy*, vol. 46, no. 5, pp. 764–771, 1992.

[4] Harald Martens and Tormond Naes. *Multivariate Calibration*. John Wiley and Sons, Chichester, 1989.

[5] Philip J. Brown. *Measurement, Regression, and Calibration*. Clarendon Press Oxford, Oxford, 1993.

[6] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press., Cambridge, 1981.

[7] M. Stone. "Cross-validatory choice and assessment of statistical prediction." *Journal of the Royal Statistical Society*. , no. B, pp. 111–133, 1974.

[8] R. D. Snee. "Validation of regression models: Methods and examples." *Technometrics*, vol. 19, no. 4, pp. 415–428, 1977.

[9] Simon Haykin. *Neural Networks, A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, N.J, second edition, 1999.

[10] Wolfgang Härdle. *Applied Nonparametric Regression*. Cambridge University Press., Cambridge, 1990.

[11] William S. Cleveland and Susan J Devlin. "Locally weighted regression: An approach to regression analysis by local fitting." *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 596–610, 1988.

[12] Sonja Sekulic, Mary Beth Seasholtz, Ziyi Wang, and Bruce R. Kowalski. "Nonlinear multivariate calibration methods in analytical chemistry." *Analytical Chemistry*, vol. 65, no. 19, pp. 835a - 845a, 1993.

[13] W. Patrick Carey and Sinclair S. Yee. "Calibration of nonlinear solid-state sensor arrays using multivariate regression techniques." *Sensors and Actuators*, vol. B, no. 9, pp. 113–122, 1992.

[14] J. H. Friedman. "Multivariate adaptive regression splines." *The Annals of Statistics.*, vol. 19, no. 1, pp. 1-141, 1991.

[15] J. H. Friedman and W. Stuetzle. "Projection pursuit regression." *J. Amer. Statist. Assoc.*, vol. 76, pp. 817-823, 1981.

[16] Tormand Naes, Tomas Isaksson, and Bruce Kowalski. "Locally weighted regression and scatter correction for near-infrared reflectance data." *Analytical Chemistry*, vol. 62, no. 7, pp. 664-673, 1990.

[17] Özden Gür Ali and Yu-To Chen. "Design quality and robustness with neural networks." *IEEE transactions on neural networks*, vol. 10, no. 6, pp. 1518-1527, 1999.

[18] Yi Liu. "Calibrating an industrial microwave six-port instrument using the artificial neural network technique." *IEEE transactions on instrumentation and measurement*, vol. 45, no. 2, pp. 651-656, 1996.

[19] P. Bartley, S. O. Nelson, R. W. McClendon, and S. Trabelsi. "Determining the moisture content of wheat with an artificial neural network from microwave transmission measurements." *IEEE transactions on instrumentation and measurement*, vol. 47, no. 1, pp. 123-126, 1998.

[20] Halit Eren, Chun Che Fung, Kok Wai Wong, and Ashok Gupta. "Artificial neural networks in estimation of hydrocyclone parameter $d50_c$ with unusual input parameters." *IEEE transactions on instrumentation and measurement*, vol. 46, no. 4, pp. 908-912, 1997.

[21] Antonio Pardo, Santiago Marco, and Josep Samitier. "Nonlinear inverse dynamic models of gas sensing systems based on chemical sensor arrays for quantitative measurements." *IEEE transactions on instrumentation and measurement*, vol. 47, no. 3, pp. 644-651, 1998.

[22] Gerhard Niebling and Artur Schlachter. "Qualitative and quantitative gas analysis with non-linear interdigital sensor arrays and artificial neural networks." *Sensors and Actuators, B.*, no. 26-27, pp. 289-292, 1995.

[23] Sherif Hashem, Paul E Keller, Richard T. Kouzes, and Lars J. Kangas. "Neural networks based data analysis for chemical sensor arrays." in *Proceedings of SPIE, the International Society for Optical Engineering*, 1995, vol. 2492, pp. 33-40.

[24] Gregory S. Broten and H. C. Wood. "A neural networks approach to analysing multi-component mixtures." *Measurement Science and Technology*, vol. 4, pp. 1096-1105, 1993.

[25] S. W. Moore, J. W. Gardner, E. L. Hines, W. Göpel, and U. Weimer. "A modified multilayer perceptron model for gas mixture analysis." *Sensors and Actuators, B.*, no. 15-16, pp. 344-348, 1993.

[26] H. Sundgren, F. Winquist, I. Lukkari, and I. Lundstrom. "Artificial neural networks and gas sensor arrays: quantification of individual components in a gas mixture." *Measurement Science and Technology*, vol. 2, no. N5, pp. 464-469, May 1991.

[27] Cherly L. Hammer and Gary W. Small. "Artificial neural networks for the automated detection of trichloroethylene by passive Fourier transform infrared spectrometry." *Analytical Chemistry*. vol. 72. no. 7. pp. 1680-1689. 2000.

[28] Husheng Yang and Peter R. Griffiths. "Application of multilayer feed-forward neural networks to automated compound identification in low-resolution open-path FT-IR spectrometry." *Analytical Chemistry*. vol. 71. no. 3. pp. 751-761. 1999.

[29] Chris W. Brown and Su-Chin Lo. "Chemical information based on neural network processing of near-IR spectra." *Analytical Chemistry*. vol. 70. no. 14. pp. 2983-2990. 1998.

[30] Barry J. Wythoff. Steven P. Levine. and Sterling A. Tomellini. "Spectral peak verification and recognition using multilayer neural networks." *Analytical Chemistry*. vol. 62. no. 24. pp. 2702-2709. 1990.

[31] T. Nakamoto. K. Fukunishi. and T. Moriizumi. "Identification capability of odor sensor using quartz-resonator array and neural-network pattern recognition." *Sensors and Actuators*. vol. B. no. 1. pp. 473-476. 1990.

[32] Kouichi Ema. Mamoru Yokoyama. Takamichi Nakamoto. and Toyosaka Moriizumi. "Odour-sensing system using a quartz-resonator sensor array and neural-network pattern recognition." *Sensors and Actuators*. . no. 18. pp. 291-296. 1988.

[33] Robert H. Kewley. Mark J. Embrechts. and Curt Breneman. "Data strip mining for the virtual design of pharmaceutical with neural networks." *IEEE transactions on neural networks*. vol. 11. no. 3. pp. 668-679. 2000.

[34] Haizhuang Kang. Qingping Yang. Clive Butler. Tuqiang Xie. and Fabrizio Benati. "Optimization of sensor locations for measurement of flue gas flow in industrial ducts and stacks using neural networks." *IEEE transactions on instrumentation and measurement*. vol. 49. no. 2. pp. 228-233. 2000.

[35] David J. H. Wilson. George W. Irwin. and Gordon Lightbody. "RBF principal manifolds for process monitoring." *IEEE transactions on neural networks*. vol. 10. no. 6. pp. 1424-1434. 1999.

[36] Peter J. Edwards. Alan F. Murray. Georgios Papadopoulos. A. Robin Wallace. John Barnard. and Gordon Smith. "The application of neural networks to the papermaking industry." *IEEE transactions on neural networks*. vol. 10. no. 6. pp. 1456-1464. 1999.

[37] Daniel Massicotte. Sylvie Legendre. and Andrzej Barwicz. "Neural-network-based method of calibration and measurand reconstruction for a high-pressure measuring system." *IEEE transactions on instrumentation and measurement*. vol. 47. no. 2. pp. 362-370. 1998.

[38] J. M. Dias Pereira. Octavian Postolache. and P. M. B. Silva Girão. "A temperature-compensated system for magnetic field measurements based on artificial neural networks." *IEEE transactions on instrumentation and measurement*. vol. 47. no. 2. pp. 494-498. 1998.

[39] Kuiwei Zhang, Clive Butler, Qingping Lang, and Yicheng Lu. "A fiber optic sensor for the measurement of surface roughness and displacement using artificial neural networks." *IEEE transactions on instrumentation and measurement*, vol. 46, no. 4, pp. 899–902, 1997.

[40] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators." *Neural Networks*, vol. 2, pp. 359–366, 1989.

[41] Ken-ichi Funahashi. "On the approximation realization of continuous mappings by neural networks." *Neural Networks*, vol. 2, pp. 183–192, 1989.

[42] G. Cybenko. "Approximations by superposition of a sigmoidal function." *Mathemat. Control, Signals, Systems*, vol. 2, pp. 303–314, 1989.

[43] Thomas B. Blank and Steven D. Brown. "Nonlinear multivariate mapping of chemical data using feed-forward neural networks." *Analytical Chemistry*, vol. 65, no. 21, pp. 3081–3089, 1993.

[44] Zheng Li. "Nonlinear fitting by using neural networks." *Analytical Chemistry*, vol. 65, no. 19, pp. 393 - 396, 1993.

[45] Paul J. Gemperline, James R. Long, and Vasilis G. Gregoriou. "Nonlinear multivariate calibration using principal components regression and artificial neural networks." *Analytical Chemistry*, vol. 63, no. 20, pp. 2313–2323, 1991.

[46] A. R. Barron. "Universal approximation bounds for superposition of a sigmoidal function." *IEEE Transactions on information theory*, vol. 39, pp. 930–945, 1993.

[47] Terrence L. Fine. *Feedforward Neural Network Methodology*. Springer, New York, 1999.

[48] Vladimir Cherkassky and Hossein Lari-Najafi. "Constrained topological mapping for nonparametric regression analysis." *Neural Networks*, vol. 4, pp. 27–40, 1991.

[49] Gerhard P. Hancke and Ruan Malan. "A modal analysis technique for the on-line particle size measurement of pneumatically conveyed pulverized coal." *IEEE transactions on instrumentation and measurement*, vol. 47, no. 1, pp. 114–122, 1998.

[50] Haizhuang Kang, Qingping Yang, and Clive Butler. "Modeling and measurement accuracy enhancement of flue gas flow using neural networks." *IEEE transactions on instrumentation and measurement*, vol. 47, no. 5, pp. 1379–1384, 1998.

[51] M. Hisham Choueiki and C. A. Mount-Campbell. "Training data development with the d-optimality criterion." *IEEE transactions on neural networks*, vol. 10, no. 1, pp. 56–63, 1999.

[52] Bahram G. Kermani, S. S. Schiffman, and H. T. Nagle. "A novel method for reducing the dimensionality in a sensor array." *IEEE transactions on instrumentation and measurement*, vol. 47, no. 3, pp. 728–741, 1998.

[53] Yongdong Wang and Bruce Kowalski. "Temperature-compensating calibration transfer for near-Infrared filter instrument." *Analytical Chemistry*. vol. 65. no. 9. pp. 1301–1303. 1993.

[54] Paul Geladi and Bruce R. Kowalski. "Partial least squares: A tutorial." *Analytica Chimica Acta*. . no. 185. pp. 1 17. 1986.

[55] Avraham Lorber. Lawrence E. Wangen. and Bruce R. Kowalski. "A theoretical foundation for the PLS algorithm." *Journal of Chemometrics*. vol. 1. pp. 19 31. 1987.

[56] W. H. Lawton. E. A. Sylvestre. and M. S. Maggio. "Self modeling nonlinear regression." *Technometrics*. vol. 13. no. 3. pp. 513–532. 1972.

[57] Grigory Isaakovich Barenblatt. *Scaling. Self-similarity. and Intermediate Asympotics*. Cambridge University Press. Cambridge UK.. 1994.

[58] Chad E. Anderson and John H. Kalivas. "Fundamentals of calibration transfer through Procrustes analysis." *Applied Spectroscopy*. vol. 53. no. 10. pp. 1268–1276. 1999.

[59] I. Borg and J. Lingoes. *Multidimensional Similarity Structure Analysis*. Springer-Verlag. New York. NY. 1987.

[60] Guy Jumarie. *Relative Information. Theories and Applications*. Springer-Verlag. New York. 1990.

[61] Bruce Ebanks. Prasanna Sahoo. and Wolfgang Sander. *Characterizations of Information Measures*. World Scientific Publishing Company. P.O. Box 128. Farrer Road. Singapore. 912805. 1998.

[62] Yongdong Wang. Michael J. Lysaght. and Bruce Kowalski. "Improvement of multivariate calibration through instrument standardization." *Analytical Chemistry*. vol. 64. no. 5. pp. 562–564. 1992.

[63] Ziyi Wang. Thomas Dean. and Bruce R. Kowalski. "Additive background correction in multivariate instrument standardization." *Analytical Chemistry*. vol. 67. no. 14. pp. 2380–2385. 1995.

[64] Paul J. Gemperline and JungHwan Cho. "Appearance of discontinuities in spectra transformed by the piecewise direct instrument standardization procedure." *Analytical Chemistry*. vol. 68. no. 17. pp. 2913–2915. 1996.

[65] Chi-Shi Chen. Chris W. Brown. and Su-Chin Lo. "Calibration transfer from sample cell to fibre-optic probe." *Applied Spectroscopy*. vol. 51. no. 5. pp. 744–748. 1997.

[66] E. Bouveresse. C. Hartmann. and D. L. Massart. "Standardization of near-infrared spectrometric instruments." *Analytical Chemistry*. vol. 66. no. 6. pp. 982–990. 1996.

[67] Marcelo Blanco. Jordi Coello. Hortensia Iturriaga. Santiago Maspoch. and Esther Rovira. "Wavelength calibration transfer between diode array UV-visible spectrophotometers." *Applied Spectroscopy*. vol. 49. no. 5. pp. 593–597. 1995.

[68] Thomas B. Blank, Stephen T. Sum, and Steven D. Brown. "Transfer of near-infrared multivariate calibrations without standards." *Analytical Chemistry*, vol. 68, no. 17, pp. 2987–2995, 1996.

[69] H. Swierenga, W. G. Haanstra, A. P de Weijer, and L. M. C. Buydens. "Comparison of two different approaches toward model transferability in NIR spectroscopy." *Applied Spectroscopy*, vol. 52, no. 1, pp. 7–16, 1998.

[70] Durmus Ozdemir, Matt Mosley, and Ron Williams. "Hybrid calibration models: An alternative to calibration transfer." *Applied Spectroscopy*, vol. 52, no. 4, pp. 599–603, 1998.

[71] John R. Koza. *Gentic Programming*. MIT Press, Cambridge, Massachusetts, 1992.

[72] Frédéric Despagne, Beata Walczak, and Désiré-Luc Massart. "Transfer of calibration of near-infrared spectra using neural networks." *Applied Spectroscopy*, vol. 52, no. 5, pp. 732–745, 1998.

[73] Frederick W. Koehler IV, Gary W. Small, Roger J. Combs, Robert B. Knapp, and Robert T. Kroutil. "Calibration transfer algorithm for automated qualitative analysis by passive Fourier transform infrared spectrometry." *Analytical Chemistry*, vol. 72, no. 7, pp. 1690–1698, 2000.

[74] D. E. Rumelhart, G. E. Hinton, and R.J. Williams. *Parallel and distributed processing: Exploration in the microstructures of cognition*, vol. 1, chapter Learning internal representation by error propagation, pp. 318–328. MIT Press, Cambridge, Massachusetts, 1986.

[75] Partha Niyogi, Federico Girosi, and Tomaso Poggio. "Incorporating prior information in machine learning by creating virtual examples." *Proceeding of the IEEE*, vol. 86, no. 11, pp. 2196–2209, November 1998.

[76] Yaser S. Abu-Mostafa. "Hints." *Neural Computation*, vol. 7, pp. 639–671, 1995.

[77] Yaser S. Abu-Mostafa. "A method for learning hints." in *Advances in Neural Information Processing Systems 5, Proceedings of the 1992 Conference*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds., San Mateo, CA., 1993. Morgan Kaufmann.

[78] Micheal L. Thompson and Mark A. Kramer. "A hybrid modeling methodology to combine prior knowledge and neural networks." in *Proceedings of the International Joint Conference on Neural Networks*, Piscataway, NJ., 1993, vol. 2, pp. 2987–2990. IEEE.

[79] Mark A. Kramer, Michael L. Thompson, and Phiroz M. Bhagat. "Embedding theoretical models in neural networks." in *Proceedings of the American Control Conference*, 1992, pp. 475–479.

[80] Wayne H. Joerding and Jack L. Meador. "Encoding a prior information in feedforward networks." *Neural Networks*, vol. 4, pp. 847–856, 1991.

[81] T. Poggio and F. Girosi. "Networks for approximation and learning." *Proceedings of the IEEE*. vol. 78. pp. 1481–1497. 1990.

[82] Yann le Cun. *Connectionism in Perspective*. chapter Generalization and network design strategies. pp. 143–155. Elsevier Science Publishers. Amsterdam. Holland. 1989.

[83] Dimitris A. Karras and Stavros J. Perantonis. "An efficient constrained training algorithm for feedforward networks." *IEEE transactions on neural networks*. vol. 6. no. 6. pp. 1420–1434. 1995.

[84] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. W.H. Winston. Washington. DC. 1977.

[85] Tor A. Johansen and Bjarne A. Foss. "Representing and learning unmodeled dynamics with neural network memories." in *Proceedings of the American Control Conference*. 1992. pp. 3037–3043.

[86] P. Simard. B. Victorri. Y. LeCun. and J. Denker. "Tangent-prop : A formalism for specifying selected invariances in an adaptive network." in *Advances in Neural Information Processing Systems 4. Proceedings of the 1991 Conference*. J. E. Moody. S. J. Hanson. and R. P. Lippmann. Eds.. San Mateo. CA.. 1992. pp. 895–903. Morgan Kaufmann.

[87] Tom M Michell and S. Thrun. "Explanation based neural network learning for robot control." in *Advances in Neural Information Processing Systems 5. Proceedings of the 1992 Conference*. D. Touretzky and M. Mozer. Eds.. San Mateo. CA.. 1993. Morgan Kaufmann.

[88] Chris M. Bishop. "Curvature-driven smoothing: A learning algorithm for feedforward networks." *IEEE Transactions on Neural Networks*. vol. 4. no. 5. pp. 882–884. 1993.

[89] Ryusuke Masuoka. "Noise robustness of EBNN learning." in *Proceedings of the International Joint Conference on Neural Networks. Nagoya. Japan.*. Piscataway. NJ.. 1993. vol. 2. pp. 1665–1668. IEEE.

[90] Jouko Lampinen and Arto Selonen. "Multilayer perceptron training with inaccurate derivative information." in *1995 IEEE International Conference on Neural Networks Proceedings. Perth. Western Australia.*. Piscataway. NJ.. 1995. vol. 5. pp. 2811–2815. IEEE.

[91] S. Thrun. "Is learning the n-th thing any easier than learning the first." in *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*. D. Touretzky and M. Mozer. Eds.. Cambridge. MA.. 1996. MIT Press.

[92] Ryusuke Masuoka. "Constraining neural networks to fit target slopes." Tech. Rep.. Fujitsu Laboratories. NetMedia. 1015 Kamikodanaka. Nakahara-ku Kawasaki. Japan.. 1995. Internal lab report dated 31 Dec.

[93] Sebastian Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers. Boston. 1996.

[94] Arto Selonen and Jouko Lampinen. "Experiments on regularizing MLP models with background knowledge." in *Artificial Neural Networks - ICANN'97. 7th International Conference Proceedings. Lausanne. Switzerland. October 8-10. 1997.*. Berlin. Germany. 1997. pp. 367-372. Springer.

[95] Jeong-Woo Lee and Jun-Ho Oh. "Hybrid learning of mapping and its jacobian in multilayer neural networks." *Neural Computation.* vol. 9. pp. 937-958. 1997.

[96] Peter Bajcsy and Narendra Ahuja. "A new framework for hierarchical segmentation using similarity analysis." in *Scale-Space Theory in Computer Vision. First International Conference. Utrecht. The Netherlands. July 2-4. 1997.* Bart ter Haar Romeny. Luc Florack. Jan Koenderink. and Max Viergever. Eds.. Berlin. Germany. 1997. Springer.

[97] José C. Principe. Neil R. Euliano. and W. Curt Lefebvre. *Neural and adaptive systems: Fundementals through simulations.* John Wiley and Sons Inc.. New York. 2000.

[98] Steve Ellacott and Deb Bose. *Neural networks: Deterministic Methods of Analysis.* International thomson computer press. London. England. first edition. 1996.

[99] Richard P. Lippmann. "An introduction to computing with neural nets." *IEEE ASSP Magazine.* pp. 4-22. April 1987.

[100] J. L. McClelland. D. E. Rumelhart. and G. E. Hinton. *Parallel and distributed processing: Exploration in the microstructures of cognition.* vol. 1. chapter The appeal of parallel distributed processing. pp. 3-44. MIT Press. Cambridge. Massachusetts. 1986.

[101] D. E. Rumelhart. G. E. Hinton. and J. L. McClelland. *Parallel and distributed processing: Exploration in the microstructures of cognition.* vol. 1. chapter A general framework for parallel distributed processing. pp. 45-76. MIT Press. Cambridge. Massachusetts. 1986.

[102] John B. Best. Ed.. *Best and Taylor's Physiological Basis of Medical Practice.* chapter Sensory Processes. pp. 970-986. Williams and Wilkens. Baltimore. MD.. eleventh edition. 1986.

[103] D. E. Rumelhart and J. L. McClelland. *Parallel and distributed processing: Exploration in the microstructures of cognition.* vol. 1. chapter PDP models and general issues in cognitive science. pp. 110-146. MIT Press. Cambridge. Massachusetts. 1986.

[104] T. Kohonen. *Associative Memory: A System Theoretical Approach.* Springer. Berlin. Germany. 1977.

[105] T. Kohonen. *Self-Organization and Associative Memory.* Springer. Berlin. Germany. 1984.

[106] S. A. Amari. *Systems Neuroscience.* chapter A mathematical approach to neural systems. pp. 67-117. Academic Press. New York. NY.. 1977.

[107] J. A Feldman and D. H. Ballard. "Connectionist models and their properties." *Cognitive Science*. . no. 6. pp. 205-254. 1982.

[108] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice-Hall Inc. Englewood Cliffs. N.J.. 1975.

[109] M. Leshno. V. Lin. A. Pinkus. and S. Schocken. "Multilayer feedforward networks with a polynomial activation function can approximate any function." *Neural Networks*. vol. 6. pp. 861-867. 1993.

[110] Jill L. Card. Debbie L. Sniderman. and Casimir Klimasauskas. "Dynamic neural control for a plasma etch process." *IEEE Transactions on Neural Networks*. vol. 8. no. 4. pp. 883-901. 1997.

[111] Changfeng Wang and Santosh S. Venkatesh. "Temporal dynamics of generalization in neural networks." in *Advances in Neural Information Processing Systems 7. Proceedings of the 1994 Conference*. Gerald Tesauro. David Touretzky. and Todd Leen. Eds.. Cambridge. MA.. 1995. MIT Press.

[112] Chyi-Tsong Chen and Wei-Der Chang. "A feedforward neural network with function shape autotuning." *Neural Networks*. vol. 9. no. 4. pp. 627-641. 1996.

[113] Guyon I. J. Makhoul. R. Schwartz. and V. Vapnik. "What size test set gives good error rate estimates." *IEEE Transactions in Pattern Analysis and Machine Intelligence*. vol. 20. pp. 52-64. 1998.

[114] M. Kearns. "A bound on the error of cross validation using the approximation and estimation rates. with consequences for the training-test split." *Neural Computation*. vol. 9. pp. 1143-1161. 1997.

[115] Jan Larson and Lars Kai Hansen. "Generalization performance of regularized neural network models." in *Neural Networks for Signal Processing. Proceedings of the 1994 IEEE Workshop*. John Vlontzos. Jenq-Neng Hwang. and Elizabeth Wilson. Eds.. NewYork. NY. 1994. pp. 42-51. IEEE.

[116] Vladimir N. Vapnik. "An overview of statistical learning theory." *IEEE Transactions on Neural Networks*. vol. 10. no. 5. pp. 988-999. 1999.

[117] Ester Levin. Naftali Tishby. and Sara A. Solla. "A statistical approach to learning and generalization in layered neural networks." *Proceeding of the IEEE*. vol. 78. no. 10. pp. 1568-1574. October 1990.

[118] H. White. "Learning in artificial neural networks: A statistical perspective." *Neural computation*. vol. 1. pp. 425-464. 1989.

[119] Dilip Sarkar. "Randomness in generalization ability: A source to improve it." *IEEE transactions on neural networks*. vol. 7. no. 3. pp. 676-685. 1996.

[120] Partha Niyogi and Federico Girosi. "On the relationship between generalization error. hypothesis complexity. and sample complexity for radial basis functions." *Neural computation*. vol. 4. pp. 819-842. 1996.

[121] S. Amari, Noboru Murata an Klaus-Robert Müller, Michael Finke, and Howard Hua Yang, "Asymptotic statistical theory of overtraining and cross-validation," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 985–996, 1997.

[122] S. Amari, N. Murata, and K.-R. Müller, "Statistical theory of overtraining - is cross-validation asymptotically effective," in *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, D. Touretzky and M. Mozer, Eds., Cambridge, MA., 1996, pp. 176–182, MIT Press.

[123] V. Kurkova and P. Kainen, "Functionally equivalent feedforward neural networks," *Neural Computation*, vol. 6, pp. 543–558, 1994.

[124] D. R. Hush, B. Horne, and J. M. Salas, "Error surfaces for multilayer perceptrons," *IEEE transactions on Systems, Man, and Cybernetics*, vol. 22, no. 5, pp. 1152–1161, 1992.

[125] P. Auer, M. Herbster, and M. Warmuth, "Exponentially many local minima for single neurons," in *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds., Cambridge M.A., 1996, pp. 316–322, MIT Press.

[126] Xiao-Hu Yu, "Can backpropagation error surface not have a local minima," *IEEE transactions on neural networks*, vol. 3, no. 6, pp. 1019–1021, 1992.

[127] Kiyotoshi Matsuoka and Jianqiang Yi, "Backpropagation based on the logarithmic error function and elimination of local minima," *IEEE International Joint Conference on Neural Networks, Singapore*, vol. 2, pp. 1117–1122, 1991.

[128] Vladmir Cherkassky, Don Gehring, and Filip Mulier, "A comparison of adaptive methods for function estimation from samples," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 969–984, 1996.

[129] S. L. S. Jacoby, J. S. Kowalik, and J. T. Pizzo, *Iterative Methods for NonLinear Optimization Problems*, Prentice-Hall., Englewood Cliffs, New Jersey, 1972.

[130] Yonathan Bard, *Nonlinear Parameter Estimation*, Academic Press, New York, 1974.

[131] R. J. Williams and D. Zipser, "Gradient based learning algorithm for recurrent neural networks and their computational complexity," in *Backpropagation: Theory, architectures, and applications*, Y. Chauvin and D. E. Rumelhart, Eds., Hillsdale, NJ, 1995, pp. 433–486, Springer-Verlang.

[132] Pierri Baldi, "Gradient descent learning algorithm overview: A general dynamical systems perspective," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 182–195, 1995.

[133] P. Patrick Van Der Smagt, "Minimisation methods for training feedforward neural networks," *Neural Networks*, vol. 7, pp. 1–11, 1994.

[134] Stanislaw Osowski, Piotr Bojarczak, and Maciej Stodolski, "fast second order learning algorithms for feedforward multilayer neural networks and its applications," *Neural Networks*, vol. 9, pp. 1583–1596, 1996.

[135] Martin T. Hagan and Mohammad B. Menhaj. "Training feedforward networks with the Marquardt algorithm." *IEEE Transactions on Neural Networks*. vol. 5. pp. 989–993. 1994.

[136] G. Lera and M. Pinzolas. "A quasi-local Levenberg-Marquardt algorithm for neural network training." *The 1998 IEEE International Joint Conference on Neural Networks Proceedings. Anchorage, Alaska*. vol. 3. pp. 2242–2246. 1998.

[137] M. Moller. "A scaled conjugate gradient algorithm for fast supervised learning." *Neural Networks*. vol. 6. pp. 525–533. 1993.

[138] Zi-Qin Wang, Michael T. Manry, and Jeffery L. Schiano. "LMS learning algorithms: Misconceptions and new results on convergence." *IEEE transactions on neural networks*. vol. 11. no. 1. pp. 47–56. 2000.

[139] W. S. Meisel. *Computer-Oriented Approaches To Pattern Recognition*. pp. 146–148. Academic Press. New York. 1972.

[140] Derrick Nguyen and Bernard Widrow. "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights." in *Proceedings of the International Joint Conference on Neural Networks. San Diego, California*. Piscataway, NJ., 1990. vol. 3. pp. 21–26. IEEE.

[141] K. F. Riley, M. P. Hobson, and S. J. Bence. *Mathematical Methods for Physics and Engineering*. Cambridge University Press. Cambridge UK., 1997.

[142] L. J. Fogel. "A note on the sampling theorem." *IRE Transactions in information theory*. vol. IT-1. pp. 47–48. 1955.

[143] D. A. Linden and N. M. Abramson. "A generalization of the sampling theorem." *Information control*. vol. 3. pp. 26–31. 1960. Errata. ibid. vol 4 pp. 95–96. 1961.

[144] Dale H. Mugler and Wolfgang Splettsroesser. "Linear prediction from samples of a function and its derivatives." *IEEE Transactions on Information Theory*. vol. IT-33. no. 3. pp. 360–366. May 1987.

[145] Guang-Bin Huang and Haroon A. Babri. "Upper bound on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions." *IEEE transactions on neural networks*. vol. 9. no. 1. pp. 224–229. 1998.

[146] W. F. Schmidt, S. R. Raudys, M. A. Kraaijveld, M. Skurikhina, and R. P. Duin. "Initialization, back-propagation and generalization of feed-forward classifiers." in *IEEE International Conference on Neural Networks. San Francisco*. IEEE. 1993. vol. 3. pp. 598–604.

[147] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press. Boca Raton. Florida. 1997.

[148] Jeffrey D. Hart. *Nonparametric Smoothing and Lack-of-Fit Tests*. Springer. New York. 1997.

[149] G. K. Shukla. "On the problem of calibration." *Technometrics*. vol. 14. no. 3. pp. 547–553. 1976.

[150] R. L. Ott and R. H. Myers. "Optimal experimental design for estimating the independent variable in regression." *Technometrics*. vol. 10. no. 4. pp. 811-823. 1968.

[151] Friedrich Pukelsheim. *Optimal Design of Experiments*. John Wiley and Sons Inc.. New York. 1993.

[152] R. W. Kennard and L. A. Stone. "Computer aided design of experiments." *Technometrics*. vol. 11. no. 1. pp. 137-148. 1969.

[153] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press. Princeton. N.J.. 1961.

[154] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlang. New York. 1986.

[155] Martin Bilodeau and David Brenner. *Theory of Multivariate Statistics*. Springer.. New York. 1999.

[156] Andrea Bobbio. Patrizia Tavella. Andrea Montefusco. and Stefania Costamangna. "Monitoring the calibration status of measuring instruments by a stochastic model." *IEEE transactions on instrumentation and measurement*. vol. 46. no. 4. pp. 747-751. 1997.

[157] Gerard N. Stenbakken. "Effects of nonmodel errors on model-based testing." *IEEE transactions on instrumentation and measurement*. vol. 45. no. 2. pp. 384-388. 1996.

[158] R. M. Pringle and A. A. Rayner. *Generalized Inverse Matrices With Application to Statistics*. Hafner Publishing Company. New York. 1971.

[159] Max Halperin. "On inverse estimation in linear regression." *Technometrics*. vol. 12. no. 4. pp. 727-736. 1970.

[160] Donal Marquardt. "Generalized inverses. ridge regression. biased linear estimation. and nonlinear estimation." *Technometrics*. vol. 12. no. 3. pp. 591-612. 1970.

A. Factors Influencing Calibration Accuracy

This appendix presents an overview of the factors influencing calibration accuracy and its the dependency on these factors. A brief discussion regarding the methods that have been used to change the degree of these dependencies is also provided

The importance of this overview is that it provides, in terms of the instrument model shown in Figure (1.1), an appreciation of the complexity of calibration and, more importantly, of the calibration transfer problem for the FFNN calibration model. In addition, by having identified the factors influencing calibration accuracy, the limitations and assumption used in the the development of the calibration transfer method presented in this thesis can be more clearly appreciated.

A.1 Influencing factors in calibration

Some of the obvious factors influencing calibration accuracy are easily identifiable: they include the quantity of calibration data, the error in the data, the complexity of the calibration model, and the dimensionality of $\mathcal{X}$. Similarly, the dependency of the calibration accuracy on quantity of calibration data appears intuitively obvious: increase the quantity of data to obtain greater accuracy. Almost equally intuitive is the inter-dependency of the calibration accuracy on the quantity of calibration data, the calibration data errors, complexity of the calibration model, and the dimensionality of $\mathcal{X}$: an increase in errors, or complexity, or dimensionality will demand an increase in the quantity of calibration data in order to maintain a particular calibration accuracy.

What may be less obvious, besides other influencing factors, is the principles at work in forming these dependencies and the methods used to increase or reduce these dependencies.

Another less obvious consideration is the precise meaning of calibration accuracy. As described in section 1.2.2, the estimated error of a calibration is given by $\hat{e}$ that is measured with an appropriate metric, such as that given by Equation (1.3). Being an estimate, $\hat{e}$ will itself possess a degree of variability. This variability in $\hat{e}$ is influenced not only by the data used to obtain $\hat{e}$, but also by its estimator and by the potential variability that exists in obtaining $\hat{h}$. Therefore, to avoid complicating the discussion with the estimated error, the true error between h and $\hat{h}$, given by Equation (1.2), is assumed to be measurable. In this context, improving the accuracy of the calibration is taken to mean reducing e, i.e. $e \rightarrow 0$.

From Equation (1.2), it is apparent that the factors influencing the error e are those which influence the process of obtaining $\hat{h}$. In other words, any factor that influences the approximation of h using the data set D can be viewed as a potential factor influencing e. This simple observation not only results in identifying many of the obvious factors described earlier, it also allows identifying a number of less obvious factors, which are now discussed.

A.1.1 Data errors.

One of the more apparent influence on the calibration accuracy is the existence of random and systematic errors in the calibration data set. It is evident that errors in D will make it difficult to obtain an accurate approximation of h since the data is now no longer exemplifying h precisely.

Reducing the influence of errors on the data can be approached using statistical techniques to obtain optimal, in some sense, estimates of the calibration model's parameters [4, 5, 10]. If the estimates of the model parameters are obtained with a consistent estimator [108], then increasing the number of samples will improve the

estimates of the model parameters. With improved parameter estimates it is assumed that the accuracy in the approximation $\hat{h}$ will improve. Alternatively, reducing the noise should also improve the accuracy of $\hat{h}$ or allow less data to be required for a given level of calibration accuracy.

A.1.2 Model selection.

In some instances of calibration, increasing the data or reducing the error does not improve the calibration accuracy. In these instances it may be that an inappropriate calibration model is being used, such as selecting a linear model for a non linear relationship. In this case, additional data will generally not improve the accuracy of the calibration.

The appropriateness of the model can also be estimated using a number of measures [4, 148].

A.1.3 Data selection.

Calibration accuracy will be influenced by the coordinates or locations selected in $\mathcal{X}$ to sample h. This data selection influence can be optimised, in some measure, during two instances of using calibration data. The first instance occurs when h is assumed to have a particular functional form and data has not yet been collected. Assuming a functional form for h allows a set of optimal, in some measure, sample locations to be determined. For example, if h is assumed to be linear, it is well known that the optimal data sample locations, in terms of the mean squared errors in estimates of the linear parameters of $\hat{h}$, are the points along each coordinate axis of $\mathcal{X}$ that define its boundary [149, 150]. This strategy is also referred to as spanning the input space [4]. Selecting data under this instance is also known as optimal experimental design [151].

As the underlying relationship of h deviates from being linear, the optimum location of the data samples becomes more difficult to determine. In this sense, selecting

the data sample location is intimately related to model validation. As the uncertainty in the assumed functional form of h increases, the strategy is to *spread the data around* so as to allow the model to be validated with greater certainty.

In the case of the linear model, the strategy of spreading the data around, is to simply partition the interval defined by the end points on each coordinate axis into equal regions. The additional points are placed on the partition boundaries. For example, if $\mathcal{X} = [0, 1]$, then the boundary end points would be a $x = 0$ and $x = 1$. Given a partition of $\mathcal{X}$ into two equal regions, the partition boundary point would be the midpoint of interval $[0, 1]$, i.e. $x = 0.5$. The extension into $\mathbb{R}^n$ and multiple partition for each coordinate axis is easily made.

The second instance of data selection occurs when a set of calibration data D is given and the task is to select a subset of data which contribute the most to the accuracy of the calibration. This instance occurs frequently in calibration transfer where the methods described by Wang *et al* [2] or Kennard and Stone [152] have been used to select data samples.

A.1.4 Domain representation.

Another factor influencing calibration accuracy, which is related to data selection, concerns the strategy of *spreading the data around*. In addition to allowing the testing of model validity, spreading the data increases the representation or sampling of the domain of h. The problem with this strategy occurs as the dimensionality of the domain of h increases. To maintain this strategy in higher dimensions requires an exponentially increasing number of sample points to fill an Euclidean space of increasing dimensions [14]. This problem is often referred to as the curse-of-dimensionality, a term coined by Bellman [153].

A pragmatic strategy to counter the curse-of-dimensionality is to simply avoid using a large dimensional space for $\mathcal{X}$. If the dimensionality of $\mathcal{X}$ cannot be controlled or it is not known which dimensional components can be discarded, then an alternative

strategy is to change the domain representation so that original data over $\mathcal{X} \subset \mathbb{R}^n$ is projected onto a space with less dimensions, i.e. $\mathcal{T} \subset \mathbb{R}^p$, where $p < n$. This can be done using, for example, principal component analysis, PCA [4.154.155]. The premise in using these strategies is that there are dimensional components of $\mathcal{X}$ which do not contribute significantly to the model. This premise, of course, is tested in methods such as PCA.

In apply strategies such as PCA, a factor that needs to be considered is the dependency of $\mathbf{x}$ on $\mathbf{g}$, which represents the physical sensing system of the instrument. In other words, it may not be a simple matter to disregard dimensional components of $\mathbf{x}$ or to project $\mathbf{x}$ onto a lower dimensional space without impacting on the interpretation of the actual measurement.

Determining whether increasing or changing the representation of the domain of h improves the accuracy of calibration, as given by Equation (1.2), is a difficult analytical problem, involving the degree of data errors, the type of model selected, and the data selection strategy. Instead of an analytical approach, a more pragmatic approach is to use the estimated calibration error $\hat{e}$, such as given by Equation (1.3), and apply model validation techniques to estimate the influence of the changes in the domain representation on $\hat{e}$.

A.1.5 Unknown g(z).

One of the most difficult type of problems in determining calibration accuracy occurs in cases when $\mathbf{g}$ is highly nonlinear and complex, so much so, that it is difficult to quantify analytically. In these cases, the relationship between $\mathbf{x}$ and corresponding y also becomes difficult to understand, thereby increasing the difficulty in selecting an appropriate parametric model for $\hat{h}$. Therefore, not only is there a problem in model selection, but the problem in selecting the model is result of not having sufficient information to even select an appropriate model.

Again, one pragmatic approach is to assume a simple linear model for h and

estimate the calibration accuracy $\hat{\epsilon}$. Then, through either ad-hoc procedures or trial and error, the model complexity can be increased, i.e. its flexibility, the data selection strategies can be varied, and the domain representation can be changed, until the $\hat{\epsilon}$ becomes acceptable [4]. In the case of varying the model flexibility, the usual approach is to attempt to linearise the problem by either partitioning the input space $\mathcal{X}$ into regions where h can be more adequately approximated with a linear model or to find a nonlinear transformation of the domain of $\mathcal{X}$ so as to have a linear problem.

An alternative approach is to use nonparametric modelling methods, in which case the problem of model selection becomes less of an issue. Of course, an estimate of the calibration accuracy $\hat{\epsilon}$ is still needed and improving $\hat{\epsilon}$ can also be approached using the ad-hoc procedures mentioned previously. The potential advantage of nonparametric modelling lies in its functional flexibility in that it is the data itself that drives the determination of $\hat{h}$ [4, 10]. This, hopefully, lessens the burden of the ad-hoc procedures used to find an acceptable $\hat{h}$.

The use of nonparametric modelling also shifts the concern in calibration accuracy from model selection to that of data selection and domain representation. It is well known [4, 10], that nonparametric modelling methods require more data samples than parametric methods in order to achieve an acceptable approximation, $\hat{h}$.

A.2 The decision to recalibrate

The need to recalibrate was defined in this thesis to be the result of the variation in $\mathbf{g}$ producing an erroneous mapping $\hat{h}(\mathbf{g}) : \mathbf{z} \rightarrow y$. This need to recalibrate will now be defined more precisely in order to emphasise a number of important points.

Let kth calibration use the D_k calibration data set to obtain an acceptable calibration model $\hat{h}_k$. Let $\mathbf{g}_k$ describe the sensor system response during the kth calibration. It is also assumed that the estimated calibration accuracy after the kth calibration,

denoted by $\hat{e}_k$, can be estimated using, for example,

$$\hat{e}_k = \left[\frac{1}{N_j}\sum_{j=1}^{N_j}[h_k(\mathbf{g}_k(\mathbf{z}_{(j,k)})) - \hat{h}_k(\mathbf{g}_k(\mathbf{z}_{(j,k)}))]^2\right]^{\frac{1}{2}}.$$

$$= \left[\frac{1}{N_j}\sum_{j=1}^{N_j}.[y_{(j,k)} - \hat{h}_k(\mathbf{x}_{(j,k)})]^2\right]^{\frac{1}{2}}. \qquad (A.1)$$

is below an application specific bound given by ϵ_{max}. Here $y_{(j,k)}$, $\mathbf{z}_{(j,k)}$, and $\mathbf{x}_{(j,k)}$ denotes the jth calibration point from the kth validation or test set $T_k = \{(\mathbf{x}_{(j,k)} = \mathbf{g}_k(\mathbf{z}_{(j,k)}), y_{(j,k)}) : j = 1, 2, \ldots, N_j\}$.

Assume now for whatever cause, $\mathbf{g}_k$ changes. Let the changed $\mathbf{g}_k$ be denoted by $\mathbf{g}_l$. We define the need for a recalibration to occur whenever the estimated calibration error $\hat{e}_l$ based on using $\mathbf{g}_l$ exceeds $\epsilon_{\max}$, or when

$$\epsilon_{\max} < \hat{e}_l = \left[\frac{1}{N_j}\sum_{j=1}^{N_j}[h_k(\mathbf{g}_k(\mathbf{z}_{(j,k)})) - \hat{h}_k(\mathbf{g}_l(\mathbf{z}_{(j,k)}))]^2\right]^{\frac{1}{2}}.$$

$$= \left[\frac{1}{N_j}\sum_{j=1}^{N_j}.[y_{(j,k)} - \hat{h}_k(\mathbf{x}_{(j,l)})]^2\right]^{\frac{1}{2}}. \qquad (A.2)$$

where $\mathbf{x}_{(j,l)} = \mathbf{g}_l(\mathbf{z}_{(j,k)})$.

There are some practical problems with this definition. The most apparent problem is that the decision to recalibrate requires using the test or validation data set T_k. In most practical applications this is highly unlikely for a number of reasons. First, T_k may not be available at a later time due to sample degradation or other causes. Another reason may be that it is not economically viable to perform this test by virtue that its cost is almost equivalent to a calibration. Finally, and most importantly, there is no mechanism to decide when to administer the test.

These reasons present nontrivial practical problems that are typically handled in the most elementary and pragmatic manner. In some cases, the decision to recalibrate is based on experience with the instrument or the operators recognition of some inconsistency with other instrument's measurements. Other more tractable approaches

are also available, such as monitoring a specific, and likely stochastic, attribute of the instrument which is correlated to the calibration accuracy [156], or developing a separate test model, that, along with a very small number of samples, provides an alternative estimate of calibration accuracy or other instrument performance measures [157].

B. Instrument Standardisation

This appendix presents a detailed review of the methods proposed by Wang *et al.* [2] for solving the calibration transfer problem. These solutions are examined within the framework of similarity mappings.

B.1 Instrument standarisation

Wang *et al.* [2] proposed four solutions to the calibration transfer problem. Though the methods were intended to be used in NIR spectroscopy, they can be applied to other spectroscopies. To review these methods, it will be instructive to work through the first solution, as the remaining solutions are variations of the first. These solutions show directly how calibration transfer is applied and can be viewed as specific linear implementations of the various similarity maps introduced earlier.

It should be noted that typically the number of wavelengths is much greater than the number of samples so that multivariate linear regression is not possible. A common practice to overcome this difficulty is to apply PLS or PCR to reduce the dimensionality of the input space $\mathcal{X}$ so that the number of samples is equal to or greater than the number of principal components. In the following solutions the added complexity of transforming data to a reduced dimensional space and back again using PLR and PCA will not be included.

In reviewing these methods the terms and notations introduced in the thesis are used instead of those used by Wang *et al.* In addition, column vectors will be denoted as lower case boldface letters, i.e. $\mathbf{z}$ and the row vector as $\mathbf{z}^T$, where the superscript T is the transpose operator.

The spectrometer measures a set of samples from the set D_k having N_k samples where each ith sample in D_k consists of l analytes whose concentrations are given by $\mathbf{z}_i^T = [z_1, z_2, \ldots, z_l]$. It is assumed that the spectrometer provides for each ith sample a sensor output response $\mathbf{x}_i^T = \mathbf{g}_k(\mathbf{z}_i^T)$, where $\mathbf{x}_i^T$ is a row vector consisting of reflectance or absorbance values at n wavelengths, $\mathbf{x}_i^T = [x_1, x_2, \ldots, x_n]$ and where $\mathbf{g}_k(\mathbf{z}_i^T) = [g_{k_1}(\mathbf{z}_i^T), g_{k_2}(\mathbf{z}_i^T), \ldots, g_{k_n}(\mathbf{z}_i^T)]$. By assuming a linear mapping for each component of $\mathbf{g}_k$, i.e. $g_{k_j} : \mathbb{R}^l \rightarrow \mathbb{R}: j = 1, 2, \ldots, n$, we have $g_{k_j}(\mathbf{z}_i^T) = g_{k_{1_j}} z_{i1} + g_{k_{2_j}} z_{i2} + \ldots + g_{k_{l_j}} z_{il} = x_j$, where $g_{k_{p_j}}$ is the pth coefficient of the jth component of $\mathbf{g}_k$, and z_{ip} is the pth component of the ith sample, $p = 1, 2, \ldots, l$. In matrix form, over all N_k samples, this can all be simplified to have the sensor output response given by

$$\mathbf{X}_k = \mathbf{Z}\mathbf{G}_k \tag{B.1}$$

where $\mathbf{X}_k$ is a $(N_k \times n)$ matrix (samples by wavelengths) for the kth calibration data set, $\mathbf{Z}$ is a $(N_k \times l)$ matrix (samples by analytes concentration), and $\mathbf{G}_k$ is a $(l \times n)$ matrix (spectrometer analyte sensitivities by wavelengths) for the kth calibration data set.

In a similar manner the instrument output reading for the ith sample is $\hat{y}_i = \hat{h}_k(\mathbf{x}_i^T)$, which would be a predicted value of some property of a specific analyte, typically a predicted concentration value. Again, by assuming a linear mapping for $\hat{h}_k$, we have $\hat{h}_k : \mathbb{R}^n \rightarrow \mathbb{R}$, so $y_i = h_{k_1} x_{i1} + h_{k_2} x_{i2} \ldots + h_{k_n} x_{in}$. It is also possible to let y_i be a row vector $\mathbf{y}_i^T \in \mathbb{R}^m$, in which case there would be m expressions for each component of $\mathbf{y}_i$, each representing a specific predicted property, i.e. concentrations. In this case, the jth component of $\mathbf{y}_i$ would be given by $y_{ij} = h_{k_{1_j}} x_{i1} + h_{k_{2_j}} x_{i2} \ldots + h_{k_{n_j}} x_{in}$. In matrix form this would be expressed as

$$\mathbf{Y}_k = \mathbf{X}_k \hat{\mathbf{H}}_k \tag{B.2}$$

where $\mathbf{Y}_k$ is a $(N_k \times m)$ matrix (samples by predicted analyte properties) based on the kth calibration and $\hat{\mathbf{H}}_k$ is a $(n \times m)$ matrix (wavelengths by regression coefficient for analyte properties) estimated using the k calibration data set.

Given this notation for the operation of the spectrometer, the methods used to solve the calibration transfer problem can now be easily expressed.

1. **Standardization with the classical calibration model.** In terms of the spectrometer, this solution attempts to adjust the response of the sensor system so as to provide an improved spectral response that is as good as the spectral response $\mathbf{ZG}_k$. In other words, given a highly accurate understanding of $\mathbf{G}_k$, and a changed version $\mathbf{G}_l$, the intent is to use $\mathbf{G}_k$ to improve $\mathbf{G}_l$. In terms of similarity mappings, this is an indirect similarity mapping problem.

   To view this solution in terms of Figure (1.1), set h to $\hat{\mathbf{H}}_l$ and $\mathbf{g}$ to $\mathbf{G}_k$ and try to determine a block diagram that is equivalent to the mapping $\hat{h}_l \circ \mathbf{g}_l$, where both functions are linear mappings. The actual mapping implementation that was selected by Wang *et al.* can be viewed as an additive implementation. To see how this was obtained by Wang *et al.* , consider for the kth calibration, the sensor output response, given by

$$\mathbf{X}_k = \mathbf{Z}_k\mathbf{G}_k \tag{B.3}$$

   which represents the accurate calibration obtained with N_k samples. If $N_l = N_k$ samples are also used for the lth realisation, then the lth calibration is potentially as accurate as the kth. The sensor output response to the lth calibration is given by

$$\mathbf{X}_l = \mathbf{Z}_l\mathbf{G}_l \tag{B.4}$$

   that, more importantly, can also be expressed in terms of $\mathbf{G}_k$, using

$$= \mathbf{Z}(\mathbf{G}_k + \Delta\mathbf{G}). \tag{B.5}$$

   where $\mathbf{Z} = \mathbf{Z}_k = \mathbf{Z}_l$. This represents the use of an indirect additive similarity map, given by $\Delta G : \mathbf{g}_k \rightarrow \mathbf{g}_l$, where $\Delta G(\mathbf{g}_k) = \mathbf{g}_l = \mathbf{g}_k + \mathbf{s}_k$, where obviously $\mathbf{g}_k \equiv \mathbf{G}_k$, $\mathbf{g}_l \equiv \mathbf{G}_l$, and $\mathbf{s}_k \equiv \Delta\mathbf{G}$.

Given that the mapping operation needed is $\mathbf{g}_k + \mathbf{s}_k$ and since $\mathbf{g}_k$ is known from the prior accurate calibration, only $\mathbf{s}_k$ is needed, i.e. $\Delta\mathbf{G}$. This can be found using Equation (B.3) and Equation (B.5) to obtain

$$\begin{aligned} \mathbf{X}_l &= \mathbf{Z}(\mathbf{Z}^{-}\mathbf{X}_k + \Delta\mathbf{G}). \\ \mathbf{Z}^{-}\mathbf{X}_l &= \mathbf{Z}^{-}\mathbf{X}_k + \Delta\mathbf{G}. \\ \mathbf{Z}^{-}(\mathbf{X}_l - \mathbf{X}_k) &= \Delta\mathbf{G}. \end{aligned} \tag{B.6}$$

where $\mathbf{Z}^{-}$ is the pseudoinverse or generalised inverse [158-160] of $\mathbf{Z}$. It is now possible to estimate $\mathbf{X}_l$ using

$$\begin{aligned} \hat{\mathbf{X}}_l &= \mathbf{Z}(\mathbf{G}_k + \mathbf{Z}^{+}(\mathbf{X}_l - \mathbf{X}_k)). \\ &= \mathbf{X}_k + \mathbf{Z}\mathbf{Z}^{+}(\mathbf{X}_l - \mathbf{X}_k). \end{aligned} \tag{B.7}$$

This estimate $\hat{\mathbf{X}}_l$, along with $\mathbf{Y}_l$, would then be used to determine $\hat{\mathbf{H}}_l$. In the current presentation there is no data advantage in using $\hat{\mathbf{X}}_l$ over $\mathbf{X}_l$ to estimate $\hat{\mathbf{H}}_l$, where a data advantage is taken to mean the use of less data in the lth calibration than was used in the kth calibration. In fact, since N_k samples were used in determining both $\mathbf{X}_l$ and $\Delta\mathbf{G}$, then $\mathbf{Z}^{+}\mathbf{Z} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. This results in $\hat{\mathbf{X}}_l = \mathbf{X}_l$, or an estimate that is as accurate as if it were obtain using only Equation (B.4).

The potential advantage occurs when $N_l < N_k$ samples are used to obtain $\Delta\mathbf{G}$, in which case Equation (B.7) is written as

$$\hat{\mathbf{X}}_l = \mathbf{X}_k + \mathbf{Z}\overline{\mathbf{Z}}^{+}(\overline{\mathbf{X}}_l - \overline{\mathbf{X}}_k). \tag{B.8}$$

where $\overline{\mathbf{X}}$ denotes quantities obtained with less than $N_l < N_k$ samples. The trade-off in using less data is to accept an increase in error in the estimate for $\hat{\mathbf{X}}_l$ by virtue that the similarity map is being approximated with less data. This translates into increased error in obtaining the approximation $\hat{\mathbf{H}}_l$ and therefore, increased error in $\mathbf{Y}$.

In terms of the previous discussion regarding the information measure I, with the assumption of linearity, the number of elements in the basis of $\Delta\mathbf{G}$ and $\mathbf{G}$ is the same, i.e. they have the same dimensions, which results in $I_{k-l} = I_l$, where object l is a linear $\mathbf{G}$. This condition implies that it is not possible to reduce the data without incurring an error.

It should also be noted that the behaviour of the error in the approximation will depend on N_l, N_k, n, m, l.

2. **Standardization with the inverse calibration model.** In terms of the spectrometer, this solution attempts to modify the calibration model $\hat{\mathbf{H}}_k$ so as to be able to use the spectral data provided by $\mathbf{ZG}_l$.

   In this approach the calibration model $\hat{\mathbf{H}}_k$ is adjusted or corrected to allow it to be used with spectral data given by $\mathbf{ZG}_l$. This is a direct similarity mapping problem. The solution used by Wang *et al.* can be viewed as an additive implementation. This solution approach mirrors that of the previous standarisation technique. In this case, Wang *et al.* begin with

$$\mathbf{Y}_k = \mathbf{X}_k\hat{\mathbf{H}}_k. \tag{B.9}$$

   and seek to use $\hat{\mathbf{H}}_k$ given that $\mathbf{G}_k$ has changed to $\mathbf{G}_l$. In this case, the similarity mapping is given by $\Delta H : \hat{\mathbf{H}}_k \rightarrow \hat{\mathbf{H}}_l$, where $\Delta H(\hat{\mathbf{H}}_k) = \hat{\mathbf{H}}_l = \hat{\mathbf{H}}_k + \mathbf{S}_k$. Again, the task is to obtain an approximation for $\mathbf{S}_k$, which is simply given by $\mathbf{S}_k = \hat{\mathbf{H}}_l - \hat{\mathbf{H}}_k$. Therefore, an approximation for $\hat{\mathbf{H}}_l$ based on using $\hat{\mathbf{S}}_k$ is given by

$$\begin{aligned}\hat{\mathbf{H}}_l &= \hat{\mathbf{H}}_k + \hat{\mathbf{S}}_k. \\ &= \hat{\mathbf{H}}_k + (\hat{\mathbf{H}}_l - \hat{\mathbf{H}}_k). \\ &= \mathbf{X}_k^{+}\mathbf{Z} + (\overline{\mathbf{X}}_l^{+} - \overline{\mathbf{X}}_k^{+})\overline{\mathbf{Z}}.\end{aligned} \tag{B.10}$$

   As simple as this appears, if the estimate of $\hat{\mathbf{S}}_k$ does in fact equal the difference between the calibration models, an approximation of $\hat{\mathbf{H}}_l$, equivalent to that

obtain using $N_l = N_k$ samples is obtained. Again, this only occurs if $N_l = N_k$, in which case $\overline{\mathbf{X}}^{-}\mathbf{X} = \mathbf{I}$. Consequently, there is no data advantage in using $\hat{\mathbf{S}}_k$.

If $N_l < N_j$, then an advantage in using less data is gained but at the expense of incurring approximation errors in $\hat{\mathbf{S}}_k$ by virtue that $I_{a_k \leftarrow a_l} = I_{a_l}$, where object a is a linear map $\mathbf{H}$.

3. **Direct standardization, DS.** In terms of the spectrometer, this solution attempts to transfer the spectral data $\mathbf{X}_l$, provided by $\mathbf{G}_l$, to $\mathbf{X}_k$, thereby allowing the data to be used with $\hat{\mathbf{H}}_k$. This is a direct similarity map $\mathbf{f}_r : \mathbf{g}_l(\mathbf{z}) = \mathbf{x}(l) \rightarrow \mathbf{g}_k(\mathbf{z}) = \mathbf{x}(k)$ implemented using

$$\mathbf{X}_k = \mathbf{X}_l \mathbf{F}_k \tag{B.11}$$

where $\mathbf{F}_k$ is obtained using

$$\hat{\mathbf{F}}_k = \mathbf{X}_l^{-} \mathbf{X}_k \tag{B.12}$$

Again, if $N_l = N_k$, there is no data advantage gained and what occurs is a standard linear mapping or transformation of the spectral data from $\mathbf{ZG}_l$ so as to make it *similar* to the spectral data that would have been obtain with $\mathbf{ZG}_k$. If $N_l < N_k$ samples are used to approximate $\hat{F}_k$,

$$\hat{\mathbf{F}}_k = \overline{\mathbf{X}}_l^{-} \overline{\mathbf{X}}_k \tag{B.13}$$

then a data advantage is gained at the expense of incurring an error in the approximation.

It should be noted, in terms of the calibration, that knowledge of $\mathbf{Z}$, or the concentration of the samples used in the calibration data set, is not needed to determine $\hat{\mathbf{F}}$, whereas it was required in the previous two methods.

4. **Piecewise direct standardization, PDS.** This solution is a piecewise version of the DS method and uses the knowledge that changes in the spectral responses tend to occur in small regions in $\mathcal{X}$. Therefore, DS is applied to those

regions. Not using all the dimensional components of $\mathcal{X}$ results in a dimensional reduction A thereby potentially allowing a data advantage in the lth calibration. In terms of the similarity map, the selection of the regions can be viewed as one of the operation involved in the mapping. This operation expressed as $\mathbf{f}_r : \mathbf{x}(l) \rightarrow \mathbf{x}(k)$ and provides the nonlinear mapping implementing the selection process. The map $\mathbf{f}_r$, though, is assumed to be linear.

The use of PDS results in a piecewise construct of DS over $\mathcal{X}$. Each region will typically have the same number of elements, though a variable number of elements is also possible. For each element a corresponding $\hat{\mathbf{f}}$ is estimated. This estimation of any ith column element of $\mathbf{X}_k$, i.e. the ith wavelength, will be based on a small set of corresponding elements from $\mathbf{X}_l$. This small set includes using the ith column element from $\mathbf{X}_l$ as well as a number of neighboring column elements from either side of the ith column, where the number of neighboring elements from the left side, denoted by i_l, and right side, i_r are not necessarily the same. Therefore, the estimation of the ith column element of $\mathbf{X}_k$, denoted as $\mathbf{r}_{k_i}$, is obtained from the set of column elements in $\mathbf{X}_l$, denoted as $\mathbf{R}_{l_i} = [\mathbf{r}_{l_{i-i_l}}, \mathbf{r}_{l_{i-i_l+1}}, \ldots, \mathbf{r}_{l_i}, \ldots, \mathbf{r}_{l_{i+i_r-1}}, \mathbf{r}_{l_{i+i_r}}]$, using the linear expression

$$\mathbf{r}_{k_i} = \mathbf{R}_{l_i}\mathbf{f}_{k_i}. \tag{B.14}$$

where $\mathbf{f}_{k_i}$ is a $(i_l + i_r + 1 \times 1)$ vector providing the same functionality as $\mathbf{F}$ in Equation (B.11). Therefore, $\mathbf{f}_{k_i}$ also needs to be estimated using any of the usual linear methods, such as

$$\hat{\mathbf{f}}_{k_i} = \mathbf{R}_{l_i}^{+}\mathbf{r}_{k_i}. \tag{B.15}$$

Since each ith column in the selected regions will have its own estimate for $\hat{\mathbf{f}}_{k_i}$, there will be $N_n < n$, estimates, where n is the number of dimensions in $\mathcal{X}$. These estimates can be combined to form $\hat{\mathbf{F}}$.

$$\hat{\mathbf{F}} = \mathrm{diag}((\hat{\mathbf{f}}_{k_1})^T, (\hat{\mathbf{f}}_{k_2})^T, \ldots, (\hat{\mathbf{f}}_{k_{N_n}})^T). \tag{B.16}$$

This combining of estimates requires placing each estimate, a vector, in a specific position in $\hat{\mathbf{F}}$. Given that $\hat{\mathbf{F}}$ is a $(n \times n)$ matrix, (wavelength by wavelength), with most off-diagonal elements equal to zero, the ith estimate $\hat{\mathbf{f}}_{k_i}$, will occupy the ith column of $\hat{\mathbf{F}}$, such that the ith component of $\mathbf{f}_{k_i} = [f_{k_i,1}, f_{k_i,2}, \ldots, f_{k_i,l_l}, f_{k_i,l}, f_{k_i,l-1}, \ldots, f_{k_i,l_r}]$, occupies the ith row of $\mathbf{F}$.

As noted previously, not using regions in $\mathcal{X}$ reduces its dimensionality, thereby allowing a data advantage without appreciably incurring errors. In terms of similarity mapping, dimensional reduction can be viewed as one operation of the similarity map, the other operations can be viewed as a series of local mappings, each given by Equation (B.14)

It should be noted that the data advantage seen in PDS is due in part by the existence of similarity between $\mathbf{X}_k$ and $\mathbf{X}_l$. This similarity is identified using the external knowledge of the user thereby, as discussed in Section 2.2.2, achieving $I_{k \leftarrow l} \approx I_l + I_p$, thus allowing for a potential data advantage.

C. The $D^{(1)}$ Learning Algorithm

This appendix presents the derivation of a first order gradient descent algorithm for a single hidden layer FFNN having N_r *shape tunable* neurons consisting of sigmoidal type activation functions. The resulting algorithm fits the FFNN's output to both a set of target data points and slopes at those points in the least squared sense.

The derivation is also not based on the method of backpropagation. Instead the derivation is based on a direct application of the chain rule of differentiation which can also obtain the error gradients needed for the descent algorithm. In addition, the deriviation does not begin from the GSSE, or cost function $C^{(n)}$ given by Equation (3.37), but, alternatively, begins by using the idea of fitting the neural network's output to both the data and corresponding slope values provided by the training set. The error or cost function that results from this approach is identical to the $C^{(1)}$ cost function given by Equation (3.38). This alternative approach provides another view on the nature of the motivation used to derive the GSSE. Finally, vector notation is avoided simply to show the details of the calculations, though it is acknowledged that the use of vector notation would have provided a more compact presentation.

C.1 Deriving the Gradient Descent Algorithm

We begin with our definitions. Let $\mathcal{X} = \mathbb{R}^n$ be the n dimensional input space and $\mathcal{Y} = \mathbb{R}$ the output space. Let $o(\mathbf{x}, \mathbf{w})$ describe the activation function $o : (\mathbf{x}_i, \mathbf{w}) \rightarrow y_i$. $\mathbf{w} = \{w_i \in \mathbb{R} : i = 0, 1, \ldots, p-1\}$ represents the free parameters of o. $\mathbf{x}_i = [x_{i0}, x_{i1}, \ldots, x_{in}]^T \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Note that the notation x_{ji} refers to the jth coordinate of the ith point.

Now given a data set $D = \{(\mathbf{x}_i, d_i) : d_i \in \mathcal{Y}, i = 1, 2, \ldots, \mathcal{N}\}$, where d_i are the desired or target values, we would like to adjust the free parameters of h so that the error between d_i and the y_i, for all i, is as small as possible. So, let $\mathbf{d} = [d_1, d_2, \ldots, d_{\mathcal{N}}]^T$ and $\mathbf{y} = [y_1, y_2, \ldots, y_{\mathcal{N}}]^T$ be two points in a normed vector space of $\mathcal{N}$ dimensions. Our problem is to minimise the distance, or error, between these two points where we use the square of the l_2-normed distance as our measure of the error

$$\begin{aligned} \epsilon_d^2 &= \|\mathbf{d}, \mathbf{y}\|_2^2 = d(\mathbf{d}, \mathbf{y})^2, \\ &= \left(\left(\sum_{i=1}^{\mathcal{N}} (d_i - y_i)^2 \right)^{\frac{1}{2}} \right)^2 \\ &= \sum_{i=1}^{\mathcal{N}} (d_i - y_i)^2 . \\ &= \sum_{i=1}^{\mathcal{N}} d_i^2 - 2 d_i y_i + y_i^2 . \end{aligned} \tag{C.1}$$

Equation (C.1) is commonly used as an error measure in deriving the backpropagation rule for feedforward neural networks which fits the neural model to a set of target data D in the least squares sense.

What is needed now is a method to also fit the model to a set of slopes at each target data point. We denote these desired slopes using $\mathbf{m_i} = \nabla_{\mathbf{x}} y(\mathbf{x}) = [\frac{\partial h(\mathbf{x})}{\partial x_0}|_{\mathbf{x}_i}, \frac{\partial h(\mathbf{x})}{\partial x_1}|_{\mathbf{x}_i}, \ldots, \frac{\partial h(\mathbf{x})}{\partial x_n}|_{\mathbf{x}_i}]^T$ for $i = 1, 2, \ldots, \mathcal{N}$. In other words, we require y to not only approximate the points in D, but to also provide a specified slope along each coordinate axis at these points. Therefore, we need to measure the difference between the actual slope of y at these points and the target slope $\mathbf{m}_i$.

Let $\mathbf{M} = [(\mathbf{m}_1)^T, (\mathbf{m}_2)^T, \ldots, (\mathbf{m}_{\mathcal{N}})^T]^T$ be the set of desired slopes for the $i = 1$ to $\mathcal{N}$ data points and let $\mathbf{Y} = \nabla_{\mathbf{x}} \mathbf{y} = [(\nabla_{\mathbf{x}} y)_{\mathbf{x}_1}^T, (\nabla_{\mathbf{x}} y)_{\mathbf{x}_2}^T, \ldots, (\nabla_{\mathbf{x}} y)_{\mathbf{x}_{\mathcal{N}}}^T]^T$ be the actual slopes at those $\mathcal{N}$ data points. We can now consider the vectors $\mathbf{M}$ and $\mathbf{Y}$ as two points in the normed vector space of $(\mathcal{N})(n)$ dimensions. Again, we can measure the distance between these two points using the square of the l_2-norm which is also our

measure of error or difference between the vectors .

$$\begin{aligned} e_m^2 &= \|(\mathbf{M} - \mathbf{Y})\|_2^2. \\ &= \sum_{i=1}^{N} \sum_{j=1}^{n} (m_{ji} - \nabla_{x_j} y_i)^2. \end{aligned} \tag{C.2}$$

where the notation $\nabla_{x_j} y_i$ refers to the gradient of y with respect to the jth coordinate of the input space at the ith data point, i.e. at $\mathbf{x}_i$.

Having a measure of fit for both the data and the corresponding slopes, we need to minimise both expressions simultaneously with respect to the parameters of h. This suggests using

$$J = \frac{1}{2} \left(e_d^2 + e_m^2 \right). \tag{C.3}$$

where J is our error or cost function we want to minimise. Note the similarity between the cost function defined by J and $C^{(1)}$ given by Equation (3.38) and that both e_d^2 and e_m^2 will always be greater than or equal to zero, so that J itself will also be greater than or equal to zero. Now expanding J we have

$$\begin{aligned} J &= \frac{1}{2} \sum_{i=1}^{N} d_i^2 - 2 d_i y_i + y_i^2 + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{n} (m_{ji} - \nabla_{x_j} y_i)^2. \\ &= \frac{1}{2} \sum_{i=1}^{N} \left(d_i^2 - 2 d_i y_i + y_i^2 + \sum_{j=1}^{n} (m_{ji} - \nabla_{x_j} y_i)^2 \right). \\ &= \frac{1}{2} \sum_{i=1}^{N} \left(d_i^2 - 2 d_i y_i + y_i^2 + \sum_{j=1}^{n} (m_{ji}^2 - 2 m_{ji} \nabla_{x_j} y_i + (\nabla_{x_j} y_i)^2) \right). \end{aligned} \tag{C.4}$$

To minimise this expression, we differentiate with respect to the parameter w_l, set the resulting expression to zero, and solve for all w_l. Here w_l represents the lth weight parameter of the neural network, where $l = 0, 1, 2, \ldots, p-1$. Instead of solving directly for w_l which presents a number of difficulties in the nonlinear case, we seek an iterative solution using a gradient descent method. Therefore, we proceed by first differentiating our expression of J to obtain

$$\frac{\partial}{\partial w_l} J = \nabla_{w_l} J = \frac{\partial}{\partial w_l} \frac{1}{2} \sum_{i=1}^{N} \left(d_i^2 - 2 d_i y_i + y_i^2 + \sum_{j=1}^{n} [m_{ji}^2 - 2 m_{ji} \nabla_{x_j} y_i + (\nabla_{x_j} y_i)^2] \right).$$

$$l = 0, 1, 2, \ldots, p-1. \tag{C.5}$$

where $\nabla_{w_l} J$ is simply the gradient of J with respect to w_l, so continuing

$$\begin{aligned}
\nabla_{w_l} J &= \frac{1}{2}\sum_{i=1}^{N}\left(\frac{\partial}{\partial w_l} d_i^2 - \frac{\partial}{\partial w_l} 2 d_i y_i + \frac{\partial}{\partial w_l} y_i^2 \right. \\
&\qquad \left. + \sum_{j=1}^{n}\left[\frac{\partial}{\partial w_l} m_{ji}^2 - \frac{\partial}{\partial w_l} 2 m_{ji} \nabla_{x_j} y_i + \frac{\partial}{\partial w_l} (\nabla_{x_j} y_i)^2\right]\right). \\
&= \frac{1}{2}\sum_{i=1}^{N}\left(-2 d_i \frac{\partial}{\partial w_l} y_i + 2 y_i \frac{\partial}{\partial w_l} y_i + \sum_{j=1}^{n}\left[-2 m_{ji} \frac{\partial}{\partial w_l} \nabla_{x_j} y_i + 2(\nabla_{x_j} y_i)\frac{\partial}{\partial w_l}(\nabla_{x_j} y_i)\right]\right). \\
&= \sum_{i=1}^{N}\left(-d_i \frac{\partial}{\partial w_l} y_i + y_i \frac{\partial}{\partial w_l} y_i + \sum_{j=1}^{n}\left[-m_{ji} \frac{\partial}{\partial w_l} \nabla_{x_j} y_i + (\nabla_{x_j} y_i)\frac{\partial}{\partial w_l}(\nabla_{x_j} y_i)\right]\right). \\
&= \sum_{i=1}^{N}\left(\frac{\partial y_i}{\partial w_l}(y_i - d_i) + \sum_{j=1}^{n}\left(\frac{\partial \nabla_{x_j} y_i}{\partial w_l}(\nabla_{x_j} y_i - m_i)\right)\right). \quad l = 0, 1, 2, \ldots p-1.
\end{aligned} \tag{C.6}$$

which requires evaluating the terms $\frac{\partial}{\partial w_l} y_i$, $\nabla_{x_j} y_i$, and $\frac{\partial}{\partial w_l}(\nabla_{x_j} y_i)$ in order to determine the gradient explicitly for a given model. Again, since we are only considering the univariant case, we only have one coordinate, i.e. $j = 1$, so we can drop the j subscript and summation over j and can write the gradient as

$$\nabla_{w_l} J = \sum_{i=1}^{N}\left(\frac{\partial y_i}{\partial w_l}(y_i - d_i) + \frac{\partial \nabla y_i}{\partial w_l}(\nabla y_i - m_i)\right). \quad l = 0, 1, 2, \ldots p-1. \tag{C.7}$$

In the case of a single hidden neuron we have $p = 4$ weight parameters, our current neural network model, has N_r hidden neurons, that is, we have $p = 4N_r$ weight parameters. For convenience we will denote these parameters such that they can be distinguished from each rth hidden neuron, so we write our model as

$$y(x_i) = \sum_{r=1}^{N_r}\left(\frac{c_r}{[1 + \exp(-w_{1r} x_i + w_{0r})]} + b_r\right). \tag{C.8}$$

where N_r is the number of hidden neurons and the free parameters of the rth hidden neuron are denoted w_{0r}, w_{1r}, $w_{2r} = c_r$, and $w_{3r} = b_r$. When we need to identify the output of the rth hidden neuron given the ith input, we will use the notation y_{ir}.

We proceed to determine our gradient components from

$$\nabla_{w_{lr}} J = \sum_{i=1}^{N} \left(\frac{\partial y_i}{\partial w_{lr}} (y_i - d_i) + \frac{\partial \nabla y_i}{\partial w_{lr}} (\nabla y_i - m_i) \right). \quad l = 0, 1, 2, 3. \quad r = 1, 2, \ldots, N_r. \tag{C.9}$$

Note that the expression for the gradient components do not change in form from that derived in the case of the single sigmoidal function.

We now derive the terms $\frac{\partial}{\partial w_{lr}} y_i$, $\nabla_{x_j} y_i$, and $\frac{\partial}{\partial w_{lm}} (\nabla_{x_j} y_i)$ explicitly for our model. We begin with $\frac{\partial}{\partial w_{lr}} y_i$ for $l = 0$.

$$\begin{aligned}
\frac{\partial}{\partial w_{0r}} y_i &= \frac{\partial}{\partial w_{0r}} \sum_{m=1}^{N_r} \left(\frac{c_m}{[1 + \exp(-w_{1m} x + w_{0m})]} + b_m \right). \\
&= (-1) c_r [1 + \exp(-w_{1r} x_i + w_{0r})]^{-2} [\exp(-w_{1r} x_i + w_{0r})]. \\
&= -c_r [1 + \exp(-w_{1r} x_i + w_{0r})]^{-1} [1 + \exp(-w_{1r} x_i + w_{0r})]^{-1} [\exp(-w_{1r} x_i + w_{0r})]. \\
&= -c_r [1 + \exp(-w_{1r} x_i + w_{0r})]^{-1} \frac{-1 + 1 + \exp(-w_{1r} x_i + w_{0r})}{1 + \exp(-w_{1r} x_i + w_{0r})}. \\
&= -c_r [1 + \exp(-w_{1r} x_i + w_{0r})]^{-1} \left(\frac{-1}{1 + \exp(-w_{1r} x_i + w_{0r})} + 1 \right). \\
&= -c_r h_r(x_i)[1 - h_r(x_i)].
\end{aligned}$$

let $D_{ir} = h_r(x_i)(1 - h_r(x_i))$, as this term will appear frequently, so

$$= -c_r D_{ir}. \quad r = 1, 2, \ldots, N_r. \tag{C.10}$$

Now with $l = 1$ we proceed in the same manner as with $l = 0$

$$\begin{aligned}
\frac{\partial}{\partial w_{1r}} y_i &= \frac{\partial}{\partial w_{1r}} \sum_{m=1}^{N_r} \left(\frac{c_m}{[1 + \exp(-w_{1m} x + w_{0m})]} + b_m \right). \\
&= (-1) c_r [1 + \exp(-w_{1r} x_i + w_{0r})]^{-2} (-x_i) [\exp(-w_{1r} x_i + w_{0r})]. \\
&= c_r x_i h_r(x_i)[1 - h_r(x_i)]. \\
&= c_r x_i D_{ir}. \quad r = 1, 2, \ldots, N_r.
\end{aligned} \tag{C.11}$$

with $l = 2$, $w_{2r} = c_r$, we have simply

$$\frac{\partial}{\partial w_{2r}} y_i = \frac{\partial}{\partial w_{c_r}} \sum_{m=1}^{N_r} \left(\frac{c_m}{[1 + \exp(-w_{1m}x + w_{0m})]} + b_m \right).$$
$$= \frac{\partial}{\partial c_r} \sum_{r=1}^{N_r} (c_r h_r(x_i) + b_r) = h_r(x_i), \quad r = 1, 2, \ldots, N_r. \tag{C.12}$$

and finally with $l = 3$, $w_{3r} = b_r$, we have

$$\frac{\partial}{\partial w_{3r}} y_i = \frac{\partial}{\partial w_{3r}} \sum_{m=1}^{N_r} \left(\frac{c_m}{[1 + \exp(-w_{1m}x + w_{0m})]} + b_m \right).$$
$$= \frac{\partial}{\partial b_r} \sum_{m=1}^{N_r} (c_m h_m(x_i) + b_m) = 1. \quad r = 1, 2, \ldots, N_r. \tag{C.13}$$

We now continue by evaluating the term ∇y_i which is simply the gradient with respect to x at the ith point

$$\nabla y_i = \left. \frac{d}{dx} \right|_{x_i} \sum_{m=1}^{N_r} \left(\frac{c_m}{[1 + \exp(-w_{1m}x + w_{0m})]} + b_m \right).$$
$$= \left. \frac{d}{dx} \right|_{x_i} \sum_{m=1}^{N_r} (c_m h_m(x_i) + b_m)$$
$$= \sum_{m=1}^{N_r} c_m w_{1m} h_m(x_i)[1 - h_m(x_i)].$$
$$= \sum_{m=1}^{N_r} c_m w_{1m} D_{im}. \tag{C.14}$$

The final set of terms are given by the partial derivative of ∇y_i with respect to w_{lr}, or $\frac{\partial}{\partial w_{lr}}(\nabla y_i)$. For $l = 0$ we have

$$\frac{\partial}{\partial w_{0r}}(\nabla y_i) = \frac{\partial}{\partial w_{0r}} \sum_{m=1}^{N_r} c_r w_{1m} h_m(x_i)[1 - h_m(x_i)].$$
$$= c_r w_{1r} \left((-1) h_r(x_i)(1 - h_r(x_i))(1 - h_r(x_i)) \right.$$
$$\left. + h_r(x_i)(-1)(-1) h_r(x_i)(1 - h_r(x_i)) \right).$$
$$= c_r w_{1r} h_r(x_i)(1 - h_r(x_i))[-(1 - 2h_r(x_i)) + h_r(x)].$$
$$= c_r w_{1r} D_{ir}[2h_r(x_i) - 1]. \quad r = 1, 2, \ldots, N_r. \tag{C.15}$$

Now we find the expression for $l = 1$

$$\begin{aligned}\frac{\partial}{\partial w_{1r}}(\nabla y_i) &= \frac{\partial}{\partial w_{1r}}\sum_{m=1}^{N_r} c_m w_{1m} h_m(x_i)[1 - h_m(x_i)]. \\ &= c_r h_r(x_i)[1 - h_r(x_i)] + (c_r w_{1r})(x_i) h_r(x_i)[1 - h_r(x_i)][1 - h_r(x_i)] \\ &\quad + (c_r w_{1r}) h_r(x_i)(-1)(x_i) h_r(x_i)[1 - h_r(x_i)]. \\ &= c_r D_{ir} + c x_i w_{1r} D_{ir}[1 - h_r(x_i)] - c_r x_i w_{1r} h_r(x_i) D_{ir}. \\ &= c_r D_{ir}\,(1 + x_i w_{1r}(1 - 2h_r(x_i))\,. \quad r = 1, 2, \ldots, N_r. \end{aligned} \tag{C.16}$$

For $l = 2$, $w_{2r} = c_r$, we have

$$\begin{aligned}\frac{\partial}{\partial w_{2r}}(\nabla y_i) &= \frac{\partial}{\partial c_r}\sum_{m=1}^{N_r} c_m w_{1m} h_m(x_i)[1 - h_m(x_i)]. \\ &= w_{1r} h_r(x_i)[1 - h_r(x_i)]. \\ &= w_{1r} D_{ir}. \quad r = 1, 2, \ldots, N_r. \end{aligned} \tag{C.17}$$

and for $l = 3$, $w_{3r} = b_r$, we have

$$\begin{aligned}\frac{\partial}{\partial w_{3r}}(\nabla y_i) &= \frac{\partial}{\partial b_r}\sum_{m=1}^{N_r} c_m w_{1m} h_m(x_i)[1 - h_m(x_i)]. \\ &= 0. \quad r = 1, 2, \ldots, N_r. \end{aligned} \tag{C.18}$$

Having determined all the required terms, we can now write the gradient with respect to all w_{lr}. We then have for the error gradient with respect to w_{0r}, by direct substitution into Equation (C.9)

$$\begin{aligned}\nabla_{w_{0r}} J &= \sum_{i=1}^{N}\Big[-c_r D_{ir}(y_i - d_i) + c_r w_{1r} D_{ir}[2h_r(x_i) - 1]\Big(\sum_{m=1}^{N_r} c_m w_{1m} D_{im} - m_i\Big)\Big] \\ &= \sum_{i=1}^{N} c_r D_{ir}\Big[(d_i - y_i) + w_{1r}[2h_r(x_i) - 1]\Big(\sum_{m=1}^{N_r} c_m w_{1m} D_{im} - m_i\Big)\Big]. \\ &\qquad\qquad r = 1, 2, \ldots, N_r. \end{aligned} \tag{C.19}$$

Now the error gradient with respect to w_{1r} becomes

$$\nabla_{w_{1r}} J = \sum_{i=1}^{N} \Big[(c_r x_i D_{ir})[y_i - d_i] + c_r D_{ir}\big(1 + x_i w_{1r}[1 - 2h_r(x_i)]\big)\big(\sum_{m=1}^{N_r} c_m w_{1m} D_{im} - m_i\big) \Big].$$

$$= \sum_{i=1}^{N} c_r D_{ir} \Big[x_i(y_i - d_i) + \big(1 + x_i w_{1r}[1 - 2h_r(x_i)]\big)\big(\sum_{m=1}^{N_r} c_m w_{1m} D_{im} - m_i\big) \Big]$$

$$r = 1, 2, \ldots, N_r. \quad \text{(C.20)}$$

The error gradient with respect to $w_{2r} = c_r$ is

$$\nabla_{c_r} J = \sum_{i=1}^{N} \Big[h_r(x)(y_i - d_i) + w_i D_{ir}\Big(\sum_{m=1}^{N_r} c_m w_{1m} D_{im} - m_i\Big) \Big]. \quad r = 1, 2, \ldots, N_r. \quad \text{(C.21)}$$

and finally, the error gradient with respect to $w_{3r} = b_r$ is

$$\nabla_{b_r} J = \sum_{i=1}^{N} \Big[(1)(y_i - d_i) - 0\big(\sum_{m=1}^{N_r} c_m w_{1m} D_{im} + m_i\big) \Big].$$

$$= \sum_{i=1}^{N} [y_i - d_i]. \quad r = 1, 2, \ldots, N_r. \quad \text{(C.22)}$$

Note that the parameter $w_{3r} = b_r$ is independent of r, i.e. $\nabla_{b_r} J$ evaluates to the same value for all r.

The gradient for each component consists of a factor due to the error in the data, $(y_i - d_i)$ and a factor due to the error in the slope $(w_2 w_1 D_i - m_i)$, where in the case of w_3 the factor due to error in the slope is zero. Again, we use

$$\Delta w_{lr} = -\eta_l \nabla_{w_{lr}} J. \quad \text{(C.23)}$$

to change w_l so that it moves towards a minimum error. So our update rule from the nth to the $n + 1$ iteration is then

$$w_{lr}(n + 1) = w_{lr}(n) + \Delta w_{lr}$$

$$= w_l(n) - \eta_l \nabla_{w_l} J(n). \quad l = 0, 1, 2, 3. \quad r = 1, 2, \ldots, N_r. \quad \text{(C.24)}$$

Note that we have assumed the general case of having a different step size for each component of the gradient. Selecting different step sizes for each component of the gradient is essentially equivalent to rescaling the cost function along each of its coordinate axes ([129], page 99).