Non-linear Projection to Latent Structures

by

Giuseppe Baffi

**A Thesis submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy**

Department of Chemical and Process Engineering

The University of Newcastle
1998

Abstract

This Thesis focuses on the study of multivariate statistical regression techniques which have been used to produce non-linear empirical models of chemical processes, and on the development of a novel approach to non-linear Projection to Latent Structures regression. Empirical modelling relies on the availability of process data and sound empirical regression techniques which can handle variable collinearities, measurement noise, unknown variable and noise distributions and high data set dimensionality. Projection based techniques, such as Principal Component Analysis (PCA) and Projection to Latent Structures (PLS), have been shown to be appropriate for handling such data sets. The multivariate statistical projection based techniques of PCA and linear PLS are described in detail, highlighting the benefits which can be gained by using these approaches. However, many chemical processes exhibit severely non-linear behaviour and non-linear regression techniques are required to develop empirical models. The derivation of an existing quadratic PLS algorithm is described in detail. The procedure for updating the model parameters which is required by the quadratic PLS algorithms is explored and modified. A new procedure for updating the model parameters is presented and is shown to perform better the existing algorithm. The two procedures have been evaluated on the basis of the performance of the corresponding quadratic PLS algorithms in modelling data generated with a strongly non-linear mathematical function and data generated with a mechanistic model of a benchmark pH neutralisation system. Finally a novel approach to non-linear PLS modelling is then presented combining the general approximation properties of sigmoid neural networks and radial basis function networks with the new weights updating procedure within the PLS framework. These algorithms are shown to outperform existing neural network PLS algorithms and the quadratic PLS approaches. The new neural network PLS algorithms have been evaluated on the basis of their performance in modelling the same data used to compare the quadratic PLS approaches.

Acknowledgements

I would like to express my gratitude to a number of persons who have contributed to the completion of this work.

First of all, I would like to thank my supervisors, Professor A. Julian Morris, Department of Chemical and Process Engineering, and Doctor Elaine B. Martin, Department of Engineering Mathematics, for they guidance, encouragement and support through this project and during these years.

Financial support is gratefully acknowledged from the Strang Studentship, the European project ESPRIT PROJECT 22281 (PROGNOSIS), and the Centre for Process Analysis, Chemometrics and Control.

A particular thank to my parents Domenico and Antonietta and my sister Francesca who bore my absence from home over this period of time. This work could have not been undertaken without their constant love and encouragement. Many thanks also to my friends in Rome, Daniela, Francesca and Alessandro who have been near to me, even though several thousand miles separated us. I also wish to thank my friends in Drum Din who helped me to relax letting me play percussion with them. Last, but not least, my friend Fifi who patiently bore my presence here in Newcastle and was always with me when I needed someone to talk to, or to have a good time with.

Love many; trust few - and always paddle your own canoe.

Billy Two Rivers, Canadian Mohawk chief

Table of Contents

Notation

Boldface lower case letters indicate vectors (if column or row vectors is specified in the text).

Boldface uppercase letters indicate matrices. Scalar are generally written in *italics*.

A hat (^) or a tilde (~) over a scalar, vector or matrix indicates an estimated (predicted or approximated) value, unless stated otherwise.

Superscripts T denotes the transpose of a matrix or a vector.

Subscripts in *italics* denote row and/or column indexes for scalar variables and vectors, unless stated otherwise.

Products between matrices are defined as row by column products, unless stated otherwise.

Products between vectors are defined as the inner product, unless stated otherwise.

Products between vectors and matrices are defined as row by column products, unless stated otherwise.

Summations between scalar quantities and matrices or vectors are defined as the summation of the scalar quantity and each element of the matrix or vector involved in the calculations.

Products between scalar quantities and matrices or vectors are defined as the product between the scalar quantity and each element of the matrix or vector involved in the calculations.

Linear and non-linear functions applied to vectors are applied to each element of the vectors, unless stated otherwise.

Partial derivatives of linear and non-linear functions applied to vectors are defined as the partial derivative of the functions applied to each element of the vectors.

Glossary for Chapter 2

a	principal component index.
$\boldsymbol{A}$	generic square matrix.
$e_{nm,k}$	approximation error on the n-th observation of the m-th variable for a model based on the use of the first k principal component.
E(·)	Expected value, mean operator.
$\boldsymbol{E}_K$	($N\times M$) residual matrix when K principal components are used to approximate the data matrix.
i	column/row index.
j	column/row index.
k	principal component index.
K	number of principal components retained in a PC model.
$l_{(i)}$	scalar used in power method for calculation of eigenvalue at i-th iteration.
m	column index, $m \in [1: M]$.
M	number of columns (variables) in data matrix.
n	row index, $n \in [1: N]$.
nt_j	score on the j-th principal component for new data sample $\boldsymbol{nx}$.
$\boldsymbol{nt}$	($1\times M$) row vector comprising scores nt_j for new data sample $\boldsymbol{nx}$.
$\boldsymbol{nx}$	($1\times M$) row vector comprising new data sample.

N	number of row (samples) in data matrix.
p_{mj}	loading coefficient for m-th variable on j-th principal component.
$\boldsymbol{p}_j$	($M\times1$) loading (column) vector comprising the loading coefficients for the j-th principal component.
PRESS(k)	sum of squared prediction error for a model based on the use of the first k principal components.
$\boldsymbol{P}$	($M\times M$) loading matrix comprising loading vectors $\boldsymbol{p}_j$.
R	rank of the data matrix.
$R(k)$	function of PRESS(k) for Cross Validation.
s_m	sample standard deviation for m-th variable.
t_j	score on the j-th principal component for single data sample.
t_{nj}	score on the j-th principal component for the n-th data sample.
$\boldsymbol{t}_j$	($N\times1$) score (column) vector comprising the scores on the j-th principal component for a given data matrix $\boldsymbol{X}$.
$\boldsymbol{T}$	($N\times M$) score matrix comprising score vectors $\boldsymbol{t}_j$.
$\boldsymbol{u}_{(i)}$	column vector used in power method for calculation of eigenvectors at i-th iteration.
$\boldsymbol{v}_j$	j-th eigenvector.
V(·)	Variance operator.
w_m	scaling weight factor for m-th variable.
$W(k)$	function of PRESS(k) for Cross Validation.
x_{nm}	measurement value of n-th sample for m-variable.

$\bar{x}_m$	sample mean value for m-th variable.
$\hat{x}_{nm,k}$	prediction of n-th observation of the m-th variable for a model based on the use of the first k principal component.
$\tilde{x}_{nm,k-1}$	approximation of n-th observation of the m-th variable for a model based on the use of the first $k-1$ principal component.
$\boldsymbol{x}_m$	($N\times1$) column vector comprising observations on m-th variable.
$\boldsymbol{x}_m^*$	($N\times1$) column vector comprising scaled values for m-th variable.
$\boldsymbol{X}$	($N\times M$) data matrix (observations by rows, variables by column).
$\hat{\boldsymbol{X}}$	($N\times M$) approximated data matrix.
$\boldsymbol{X}_a$	($N\times M$) deflated $\boldsymbol{X}$ matrix after a principal component have been extracted.
$\|\boldsymbol{p}\|$	Euclidean norm of vector $\boldsymbol{p}$.
λ_j	j-th eigenvalue.
σ_e	variance of the noise introduced by a sensor.
σ_x	variance of the measured value $\boldsymbol{x}$.
Σ	variance-covariance matrix or correlation matrix for data matrix $\boldsymbol{X}$.

Glossary for Chapter 3

A	number of latent variables retained in a generic PLS model.
b	generic regression coefficient between input and output latent variables, as defined in Equation 3-2.
b_j	regression coefficient between j-th input and output latent variables.
$\boldsymbol{B}$	($M\times M$) diagonal matrix comprising regression coefficients b_j.
$\boldsymbol{B}_A$	($A\times A$) diagonal matrix comprising the first A regression coefficients b_j.
$\hat{\boldsymbol{B}}_{PLS}$	($M\times K$) matrix of regression coefficients between $\boldsymbol{X}$ and $\boldsymbol{Y}$ given by PLS.
$\hat{\boldsymbol{B}}_{PLS,A}$	($M\times K$) matrix of regression coefficients between $\boldsymbol{X}$ and $\boldsymbol{Y}$ given by a PLS model based on the use of the first A latent variables.
$\hat{\boldsymbol{B}}_{MLR}$	($M\times K$) matrix of regression coefficients between $\boldsymbol{X}$ and $\boldsymbol{Y}$ given by MLR.
c_k	generic output weight coefficient for k-th output variable.
$\boldsymbol{c}$	column vector comprising output weights c_k.
$\tilde{\boldsymbol{c}}$	generic output weight vector prior to scaling it to unit norm.
$\boldsymbol{e}$	residual column vector for inner regression between input and output latent variables.
$\boldsymbol{E}$	generic ($N\times M$) input residual matrix.
$\boldsymbol{F}$	generic ($N\times K$) output residual matrix.
i	generic index.
j	generic index.

k output variable index, $k \in [1{:}\, K]$.

K number of columns (variables) in output data matrix.

m input variables column index, $m \in [1{:}\, M]$.

M number of columns (variables) in input data matrix.

n row index, $n \in [1{:}\, N]$.

N number of row (samples) in data matrices.

$\boldsymbol{p}$ generic (M×1) input loading (column) vector.

$\boldsymbol{p}_j$ (M×1) input loading (column) vector comprising the loading coefficients for the j-th input latent variable.

$\boldsymbol{P}$ (M×M) input loading matrix comprising input loading vectors $\boldsymbol{p}_j$.

$\boldsymbol{P}_A$ (M×A) input loading matrix comprising the first A input loading vectors $\boldsymbol{p}_j$.

$\boldsymbol{q}$ generic (M×1) output loading (column) vector.

$\boldsymbol{q}_j$ (M×1) output loading (column) vector comprising the loading coefficients for the j-th output latent variable.

$\boldsymbol{Q}$ (K×K) output loading matrix comprising output loading vectors $\boldsymbol{q}_j$.

$\boldsymbol{Q}_A$ (K×A) output loading matrix comprising the first A output loading vectors $\boldsymbol{q}_j$.

$\boldsymbol{R}$ (M×M) projection matrix as defined in Equation 3-36.

$\boldsymbol{R}_A$ (M×A) projection matrix based on the use of the first A latent variables, as defined in Equation 3-40.

$\boldsymbol{t}$ generic input score (column) vector, input latent variable.

$\boldsymbol{t}_j$ (N×1) score (column) vector comprising the scores on the j-th input latent

variable.

$\boldsymbol{T}$ ($N\times M$) input score matrix comprising input score vectors $\boldsymbol{t}_j$.

$\boldsymbol{T}_A$ ($N\times A$) input score matrix comprising the first A input score vectors $\boldsymbol{t}_j$.

$\boldsymbol{u}$ generic ($N\times 1$) output score (column) vector, output latent variable.

$\hat{\boldsymbol{u}}$ generic ($N\times 1$) predicted output score (column) vector.

$\boldsymbol{u}_j$ ($N\times 1$) score (column) vector comprising the scores on the j-th output latent variable.

$\hat{\boldsymbol{u}}_j$ ($N\times 1$) predicted output score vector for j-th latent variable.

$\boldsymbol{U}$ ($N\times K$) output score matrix comprising output score vectors $\boldsymbol{u}_j$.

$\hat{\boldsymbol{U}}$ ($N\times K$) predicted output score matrix comprising predicted output score vectors $\hat{\boldsymbol{u}}_j$.

$\hat{\boldsymbol{U}}_A$ ($N\times A$) predicted output score matrix comprising the first A output score vectors $\hat{\boldsymbol{u}}_j$.

w_m generic input weight coefficient for m-th input variable

$\boldsymbol{w}$ ($M\times 1$) column vector comprising generic input weights w_m.

$\boldsymbol{w}_j$ ($M\times 1$) column vector comprising the input weight coefficients for the j-th latent variable.

$\boldsymbol{W}$ ($M\times M$) input weight matrix comprising weight vectors $\boldsymbol{w}_j$.

$\boldsymbol{W}_A$ ($M\times A$) input weight matrix comprising the first A input weight vectors $\boldsymbol{w}_j$.

x_{nm} measurement value for n-th sample of m-input variable.

$\boldsymbol{x}_m$ ($N\times 1$) column vectors comprising the observations collected on the m-th input variable.

X ($N\times M$) input data matrix, process variables.

$\hat{X}$ ($N\times M$) approximated input data matrix.

$\hat{X}_A$ ($N\times M$)approximated input data matrix given by a model based on the use of the first A latent variables.

y_{nk} measurement value for n-th sample of k-output variable.

y_k ($N\times 1$) column vectors comprising the observations collected on the k-th output variable.

Y ($N\times K$) output data matrix, quality measurements.

$\hat{Y}$ ($N\times K$) predicted output data matrix

$\hat{Y}_A$ ($N\times K$) predicted output data matrix given by a model based on the use of the first A latent variables.

$\|w\|$ Euclidean norm of vector w.

$\|F\|$ norm of the output block residual matrix.

λ_w eigenvalue corresponding to first eigenvector when computing input weight from singular value decomposition of kernel matrices, Equation 3-51.

λ_c eigenvalue corresponding to first eigenvector when computing output weight from singular value decomposition of kernel matrices, Equation 3-52.

λ_t eigenvalue corresponding to first eigenvector when computing input score from singular value decomposition of kernel matrices, Equation 3-53.

λ_u eigenvalue corresponding to first eigenvector when computing output score from singular value decomposition of kernel matrices, Equation 3-54.

Glossary for Chapter 4

b scalar quantity used in input weights updating procedure.

c_0 generic coefficient for polynomial inner regression in Quadratic PLS.

c_1 generic coefficient for polynomial inner regression in Quadratic PLS.

c_2 generic coefficient for polynomial inner regression in Quadratic PLS.

$c_{0,j}$ coefficient for polynomial inner regression in Quadratic PLS for j-th latent variables.

$c_{1,j}$ coefficient for polynomial inner regression in Quadratic PLS for j-th latent variables.

$c_{2,j}$ coefficient for polynomial inner regression in Quadratic PLS for j-th latent variables.

$\boldsymbol{c}$ column vector comprising coefficients for polynomial inner regression in Quadratic PLS.

$\boldsymbol{e}$ residual column vector for inner regression between input and output latent variables.

$\boldsymbol{e}_j$ residual column vector for inner regression between input and output scores on j-th latent variable.

$\boldsymbol{E}$ generic ($N \times M$) input residual matrix.

$f(\cdot)$ generic scalar non-linear function for inner mapping.

f_j scalar non-linear function for j-th latent variable.

$\boldsymbol{f}_{00}$ column vector comprising values given by non-linear function f for known values of the input scores.

$\boldsymbol{F}$ generic ($N \times K$) output residual matrix.

i generic index.

j generic index.

k output variable index, $k \in [1: K]$.

K number of columns (variables) in output data matrix.

m input variables column index, $m \in [1: M]$.

M number of columns (variables) in input data matrix.

n row index, $n \in [1: N]$.

N number of row (samples) in data matrices.

$\boldsymbol{p}$ input loading vector.

$\boldsymbol{P}$ input loading matrix.

$\boldsymbol{q}$ output loading vector.

$\boldsymbol{Q}$ output loading matrix.

$\boldsymbol{s}$ column vector used in the original input weights updating procedure.

t_n generic input score for n-th sample.

t_{nj} input score on j-th latent variable for n-th sample.

$\boldsymbol{t}$ column vector comprising generic input scores.

$\boldsymbol{t}_j$ column vector comprising input scores on j-th latent variable.

$\boldsymbol{t}^2$ column vector comprising square values of scores for generic latent variable.

t_j^2	column vector comprising square values of scores on j-th latent variable.
T	input score matrix.
u	column vector comprising generic output scores.
u_j	column vector comprising output scores on j-th latent variable.
$\hat{u}$	generic column vector comprising predicted output scores.
U	output score matrix.
$\hat{U}$	predicted output score matrix.
v	column vector used in input weights updating procedure.
w	generic column vector comprising input weight.
W	input weight matrix.
x_m	column vector comprising measurements on m-th input variable.
X	input data matrix.
Y	output data matrix.
$\hat{Y}$	predicted output matrix.
Z	matrix comprising known part of Taylor series expansion used for input weights updating procedure.
$\frac{\partial f}{\partial c_i}$	partial derivative of function f with respect to i-th regression coefficient for Quadratic PLS.
$\frac{\partial f}{\partial w_m}$	partial derivative of function f with respect to m-th input weight.
$\frac{\partial f}{\partial c}$	generic partial derivative of function f with respect to regression coefficients c.

$\frac{\partial f}{\partial w}$	generic partial derivative of function f with respect to the input weights w.
$\left.\frac{\partial f}{\partial c}\right\|_{\infty}$	partial derivative of function f with respect to regression coefficients c computed for known values of the input latent variables (Equation 4-6).
$\left.\frac{\partial f}{\partial w}\right\|_{\infty}$	partial derivative of function f with respect to the input weights w computed for known values of the input latent variables (Equation 4-7).
Δc_i	finite increment of i-th regression coefficient (scalar) of inner model for Taylor series expansion.
Δw_m	finite increment of m-th input weight (scalar) for Taylor series expansion.
Δc	column vector comprising finite increment of inner model regression coefficients for Taylor series expansion.
Δw	column vector comprising finite increment of input weights for Taylor series expansion, i.e. input weights updating parameters.
$\|w\|$	Euclidean norm of vector w.

Glossary for Chapter 5

$\boldsymbol{e}$	residual column vector for inner regression between input and output latent variables.
$\boldsymbol{e}_j$	residual column vector for inner regression between input and output scores on j-th latent variable.
$\boldsymbol{E}$	generic ($N \times M$) input residual matrix.
$F(\cdot)$	generic scalar non-linear function for inner mapping.
$\boldsymbol{f}_{00}$	column vector comprising values given by non-linear function f for known values of the input scores.
$\boldsymbol{F}$	generic ($N \times K$) output residual matrix.
i	generic index.
j	generic index.
k	output variable index, $k \in [1: K]$.
K	number of columns (variables) in output data matrix.
m	input variables column index, $m \in [1: M]$.
M	number of columns (variables) in input data matrix.
n	row index, $n \in [1: N]$.
N	number of row (samples) in data matrices.
NC	number of hidden neurones for radial basis function PLS.
$\boldsymbol{p}$	input loading vector.

$\boldsymbol{P}$ input loading matrix.

$\boldsymbol{q}$ output loading vector.

$\boldsymbol{Q}$ output loading matrix.

$\boldsymbol{r_j}$ column vector comprising Euclidean distance between centre $\boldsymbol{c_j}$ and input scores $\boldsymbol{t}$, as defined in Equations 5-18 and 5-25.

$\boldsymbol{r_j^2}$ column vector comprising the square values of the Euclidean distance between centre $\boldsymbol{c_j}$ and input scores $\boldsymbol{t}$.

$\boldsymbol{s_j}$ auxiliary column vector used in error based updating procedure for radial basis function PLS algorithm.

$\boldsymbol{t_n}$ generic input score for n-th sample.

$\boldsymbol{t}$ column vector comprising generic input scores.

$\boldsymbol{t_j}$ column vector comprising input scores on j-th latent variable.

$\boldsymbol{T}$ output score data matrix.

$\boldsymbol{u}$ column vector comprising generic output scores.

$\boldsymbol{u_j}$ column vector comprising output scores on j-th latent variable.

$\hat{\boldsymbol{u}}$ generic column vector comprising predicted output scores.

$\boldsymbol{U}$ output scores matrix.

$\hat{\boldsymbol{U}}$ predicted output cores matrix.

$\boldsymbol{v}$ column vector used in input weights updating procedure.

$\boldsymbol{w}$ generic column vector comprising input weight.

$\boldsymbol{W}$ input weight matrix.

x_k column vector comprising measurements on k-th input variable.

x_m column vector comprising measurements on m-th input variable.

X input data matrix.

Y output data matrix.

$\hat{Y}$ predicted output matrix.

z generic argument for sigmoid neural network.

z_k column vector used in the error based input weights updating procedure.

Z matrix comprising known part of Taylor series expansion used for input weights updating procedure.

β_1 sigmoid neural network first layer bias.

β_2 sigmoid neural network second layer bias.

ρ_j gaussian function width.

$\sigma(\cdot)$ activation function for sigmoid neural network.

$\sigma'(\cdot)$ first derivative of the activation function for sigmoid neural network with respect to its argument.

$\sigma^2(\cdot)$ square value of the sigmoid activation function.

ω_0 radial basis function network bias on output layer.

ω_1 sigmoid neural network first layer weights.

ω_2 sigmoid neural network second layer weights.

ω_j radial basis function network weights.

$\frac{\partial f}{\partial c_i}$	partial derivative of function f with respect to i-th regression coefficient.
$\frac{\partial f}{\partial w_m}$	partial derivative of function f with respect to m-th input weight.
$\frac{\partial f}{\partial w}$	generic partial derivative of function f with respect to the input weights w.
$\left.\frac{\partial f}{\partial w}\right\|_{\infty}$	partial derivative of function f with respect to the input weights w computed for known values of the input latent variables (Equation 4-7).
Δw_m	finite increment of m-th input weight (scalar) for Taylor series expansion.
Δw	column vector comprising finite increment of input weights for Taylor series expansion, i.e. input weights updating parameters.
$\|w\|$	Euclidean norm of vector w.

1. Introduction

1.1 Empirical Modelling

Process analysis, process monitoring and process control relies on the availability of appropriate mathematical models to represent the system of interest. A common, but sometimes very time demanding, approach is to develop a first principles or mechanistic model of the process based upon a detailed knowledge of the chemical and physical phenomena underlying the process operation. In particular, the development of rigorous theoretical models may not be practical (in some cases not even possible) for complex processes if the model requires a large number of differential and algebraic equations with a significant number of unknown parameters. Empirical data-based modelling is a widely used alternative to mechanistic modelling since it requires less specific knowledge of the process being studied than that needed to develop a first principles model. Empirical modelling techniques require experimental data (measurements) which are collected on those variables believed to be representative of process behaviour, and of the quality or properties of the product or system output. Statistical regression techniques and neural networks are now routinely used in the process industries for building empirical models.

Multivariate statistical regression techniques, based upon least squares methodologies, have been used extensively for developing linear empirical models from monitored plant data. However, it is well known that when dealing with highly correlated multivariate problems, the traditional least squares approach can lead to singular solutions or imprecise parameter estimation. This is the case in modern chemical, bio-chemical and food processing plants, where measurements can be collected on-line on a large number of process variables which are generally noise corrupted and highly correlated. Measurement noise, correlated variables, unknown variable and noise distributions, and high data set dimensionality are typical features of data collected in process manufacturing and are known to have a detrimental effect on the performance of both statistical regression models and neural networks. These limitations can be addressed by applying multivariate statistical projection based regression techniques such as Principal Component Analysis (PCA) and Projection to Latent Structure or Partial Least Squares (PLS). These two techniques can overcome both the dimensionality and the collinearity

problems and can also provide filtering of the measurement noise. Projection based techniques can handle highly correlated, noise corrupted data sets since they are based upon the assumption of dependency (correlation) between the variables. Consequently they provide the capability to estimate the main underlying structure in terms of a number of *principal components* or *latent variables* which are a linear combination of the original (process) variables.

Principal Component Analysis is one of the oldest multivariate statistical analysis techniques. It can provide a compact representation of a large data set through a reduced number of linear combinations of the original variables which comprise relevant information about the process and the working conditions corresponding to the data forming the basis of the analysis. Thus PCA can be used for process analysis, monitoring and optimisation. However, although PCA can provide information about linear dependencies characterising the system, it cannot be used as a regression tool.

Projection to Latent Structures has been shown to be a reliable multivariate linear regression technique for the analysis and modelling of noisy and highly correlated data. In particular, it provides a compromise between the approximation to the predictor (process) variables and the prediction of the response (quality or reference) variables. In doing so, it simultaneously reduces the dimensionality of the predictor and response variable spaces by seeking those latent variables of the predictor variable space, and of the response variable space, which are themselves highly correlated. The corresponding latent variables can be used to realise a low-dimensional linear regression model between the predictor and the response variables.

However, many chemical processes exhibit severe non-linear behaviour and linear regression techniques cannot be reliably employed to develop empirical regression models. Multi-Layer Perceptrons have been used extensively to develop non-linear regression models, even though these techniques are known to be sensitive to variable noise and variable correlations. This is mainly due to the algorithms used to build the models which are based upon optimisation routines. These are known to lead to sub-optimal solutions when the training data sets comprise large numbers of noisy and correlated variables.

A number of attempts have been made to incorporate non-linear features within the linear PLS framework to provide a non-linear regression method capable of overcoming these limitations. This requires modifications to the algorithm used to develop the PLS regression model. A major

breakthrough in non-linear PLS modelling was reported in the work of Wold *et al.* (1989) who showed the benefits achievable by using an updating procedure of the parameters of the PLS model when quadratic polynomial expansions provided the non-linear features within the PLS methodology. However, their algorithm (as they stated) is somewhat obscure and fairly complicated. The objective of this Thesis was to address the non-linear PLS problem by initially searching for an improvement to the approach proposed by Wold *et al.* (1989). In particular interest focused on the model parameters updating procedure. A new procedure was developed which was shown to outperform the original algorithm. Furthermore, it is also shown that the procedure can be extended to non-linear regression approaches other than polynomial expansions. It is believed that this represents an important contribution in terms of non-linear PLS modelling, and hence non-linear empirical modelling. In particular the new procedure was extended to integrate sigmoid neural network and radial basis function networks in the PLS inner model in order to enhance the modelling capability of the corresponding neural network PLS algorithms.

It is noted that each Chapter has its own detailed introduction and literature survey which provides a specific and relevant review to each piece of work.

1.2 Outline of the Thesis

The first Chapter of the Thesis introduces the scientific area of interest and gives a general overview of the work carried out, outlining the innovative aspects and main results achieved. An outline of the Thesis is also given.

In Chapter 2, Principal Component Analysis (PCA) is reviewed along with its theoretical background and its properties. This Chapter is justified by the fact that PCA represents the basis of many projection based multivariate statistical regression techniques. Furthermore PCA is generally used as a data pre-screening tool prior to carrying out empirical regressions as well as forming a cornerstone of multivariate statistical process control through the building of a PCA representation of the process of interest.

Chapter 3 focuses upon Projection to Latent Structures or Partial Least Squares (PLS). This Chapter forms the basis of the rest of the Thesis. The major features of the methodology as a multivariate linear regression methodology are presented. The algorithm used to build PLS regression models is described. Modifications to the original linear PLS approach required to embrace complex data systems are also introduced.

In Chapter 4 an overview of existing non-linear PLS algorithms is given. The model parameters (weights) updating procedure proposed by Wold *et al.* (1989) is described in detail and its limitations are outlined, prior to developing a new procedure for updating the weights. The two weight updating procedures are then compared in terms of modelling performance when employed within a quadratic PLS framework. The new procedure is showed to outperform the original procedure proposed by Wold *et al.* (1989).

In Chapter 5, the parameter updating procedure is extended to allow the full integration of sigmoid neural networks and radial basis function networks within the PLS inner model. This approach was motivated by the recent publication of other neural network and radial basis function PLS algorithms which appeared to be promising because of the universal approximation properties of sigmoid and gaussian neural networks. It is then shown how non-linear features can be fully integrated into the PLS methodology in conjunction with the updating procedure for the PLS model parameters. Furthermore it is demonstrated that it is possible to achieve better modelling performances than the existing non-linear PLS algorithms based upon the use of the original linear PLS methodology. The model parameters updating procedure and the algorithms are evaluated on a strongly non-linear mathematical function and a benchmark pH neutralisation system.

Chapter 6 focuses upon conclusions and suggestions for future work are given.

2. Principal Component Analysis

2.1 Introduction

Principal component analysis (PCA) is one of the oldest and best known of the multivariate statistical analysis techniques and in many cases it forms the basis of multivariate data analysis. PCA was originally introduced in 1901 by Karl Pearson who provided the mathematical basis and a geometrical interpretation of the principal component methodology through his pioneering work "*On Lines and Planes of Closest Fit to Systems of Points in Space*". While in 1933, Harold Hotelling developed the modern definition of principal components, that is the directions which sequentially account for maximal variability in a sample ("*Analysis of a Complex of Statistical Variables into Principal Components*"). Pearson and Hotelling produced the core of the theoretical approach to principal component analysis, both from an algebraic and a geometric perspective, and also the underlying mathematics. A third milestone in the history of PCA was the work of C. Radhakrishna Rao ("*The Use and Interpretation of Principal Component Analysis in Applied Research*", 1964). He identified the potential of using PCA as an investigative tool in applied research. In particular, he provided various interpretations for principal components calculated from a set of multiple measurements, and gave a number of generalisations of PCA and their related properties.

Since then the literature relating to PCA has grown and today it has been successfully applied to many systems across a wide range of disciplines. Areas of particular applicability include data compression, cluster analysis, outlier detection and variable selection. A detailed description of the technique, its theoretical basis and references to a wide range of application can be found in Mardia *et al.* (1979), Anderson (1984), Jolliffe (1986) and Wold *et al.* (1987).

The key objective of PCA is to provide an approximation of a data matrix $\boldsymbol{X}$ by means of a linear decomposition of the matrix. In PCA this is achieved by performing a transformation of the original data into a set of *latent variables*, the principal components (PCs), which are particular linear combinations of the original variables.

More precisely each principal component is a normalised linear combination of the original variables which has maximum variance amongst all possible linear combinations of the set of variables and which is uncorrelated with the other PCs. According to the definition of principal components, it can be proved that the coefficients (*loadings*) of each linear combination are the coefficients of the eigenvectors of the variance-covariance matrix of the original data matrix. In particular the j-th PC corresponds to the j-th eigenvalue of the variance-covariance matrix, ranked according to its magnitude with respect to the other eigenvalues. Thus the computation of the principal components of a data matrix reduces to the solution of an eigenvalue-eigenvector problem (Appendix 1). Furthermore, it can be shown that the transformation performed by PCA is an orthogonal linear transformation, and therefore invertible.

From a geometrical point of view, the loadings identify a new system of orthogonal reference axis, and a corresponding *principal component space*, in the multidimensional space defined by the original variables. Each axis of the new reference system corresponds to a sorted direction of greatest variability. The loadings of each PC are the cosines of the angles between the axis corresponding to that PC (*principal axis*) and the axes relating to the original variables. In particular, the orthogonal transformation provided by PCA corresponds to a rigid rotation of the reference basis, i.e. a rotation of the co-ordinate axes which does not stretch the measurement units on the axes, and hence preserves distances between points. The linear combination of the original variables, with coefficients equal to the loadings of one of the PCs, define the projection (*scores*) of the original variables on the axis corresponding to that principal component.

A two dimensional example describing the use of principal component analysis is given in Figure 2.1. A set of 100 data points was randomly generated (Var_1) and a second set of data points (Var_2) was produced by multiplying the first set by two and adding white noise to it (zero mean and unit variance). The two variables were then merged into a single data matrix (set_a) and a principal component analysis was performed on that matrix. In Figure 2.1a the scatter plot of Var_1 versus Var_2 is given with the directions of the corresponding two PCs, which are defined PC_1 and PC_2. The resultant scatter plot for the two score vectors is given in Figure 2.1b. From the two plots it is clear that the first PC identifies the direction of greatest variability, whilst the second components is orthogonal to the first and accounts for minor variations.

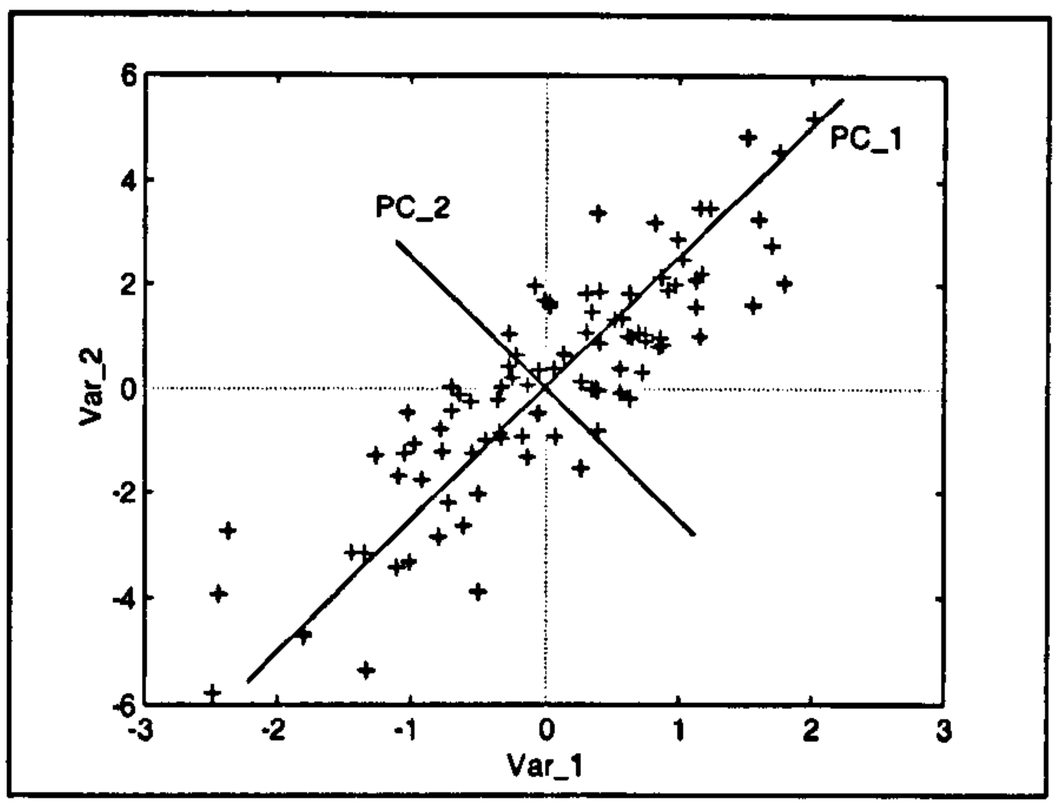

Figure 2.1a: Original variables scatter plot, with directions of the principal axes.

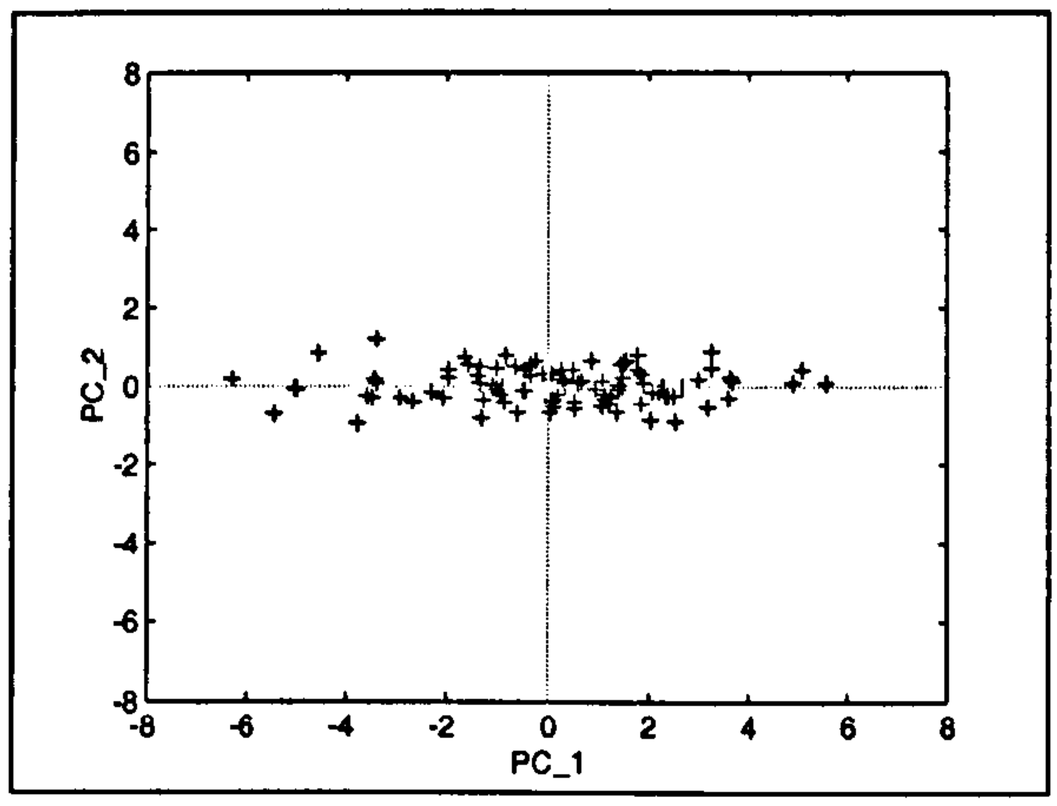

Figure 2.1b: Scores scatter plot for the same points given in Figure 2.1a.

2.2 Scale-Invariance of Principal Components Analysis

Defining each principal component as a linear combination of the original variables leads to one of the major limitations of PCA. In fact, identifying a principal component, and hence the corresponding principal axis, of a given data matrix can be considered equivalent to fitting a straight line to the data points, subject to certain constraints. But this is a least squares procedure (Wold *et al.*, 1987), and consequently the numerical values, i.e. the measurement units, in which the original data are given affect the PC model, since variables with a large variance will have correspondingly large loadings, and will tend to dominate the first few PCs. From a geometrical point of view it implies that variables with a large variance tend to pull the axes of the first principal components in their direction. Furthermore, the variance of a variable varies with the measurement units in which the variable is given, implying that changing the measurement units of one variable in the data set will change the variance-covariance structure of the data matrix and hence the PC model. Mardia *et al.* (1979) observed that the sensitivity of PCA to scaling factors is also a consequence of the eigenstructure (i.e. the eigenvalues and the corresponding eigenvectors) of a matrix not being scale invariant.

The effect of different measurement units on the directions of the principal components can be observed by multiplying Var_1 of set_a (§ 2.1) by 10 in order to generate a new data matrix, set_b. The impact of the scaling factor can be seen comparing the PCA representation of set_a with that of set_b. In Figure 2.2a the original data is indicated by '+' and the modified data set

by 'o'. The scaling factor on Var_1 changes the direction of the first principal axes pulling it towards the original reference axis corresponding to Var_1. The direction of the second PC (although not shown) will also be different, since it is constrained to be orthogonal to the first principal component.

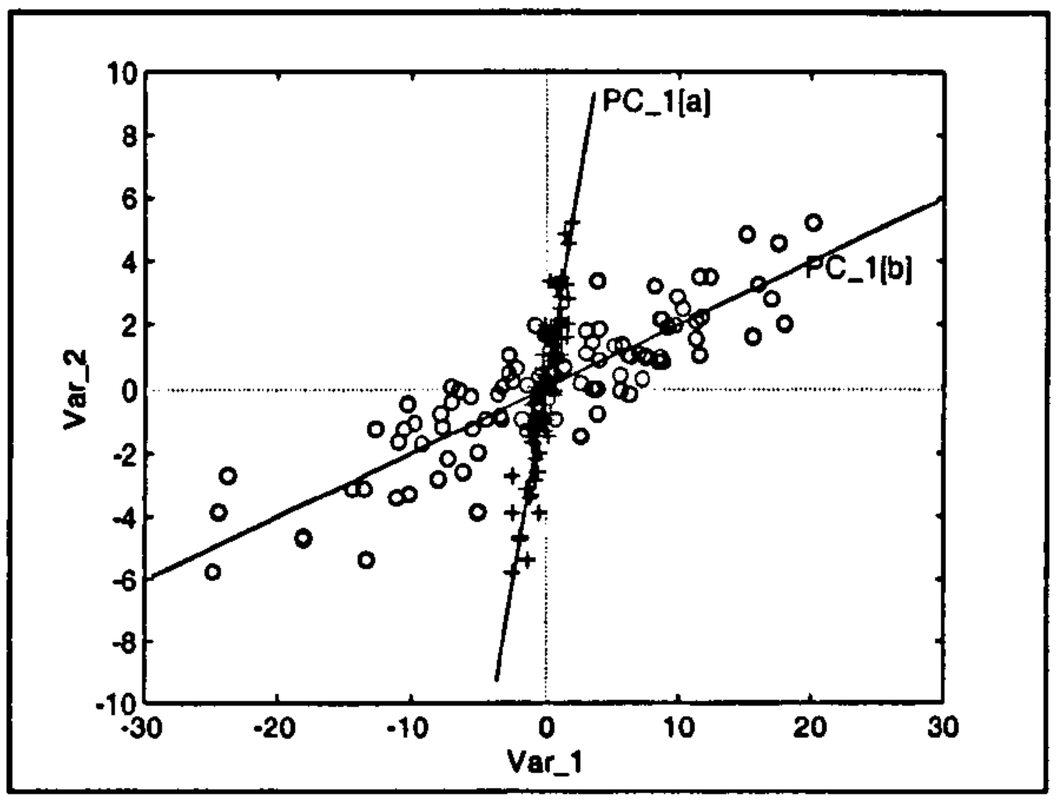

Figure 2.2a: Real variables scatter plot, with directions of the principal axes for PC_1.

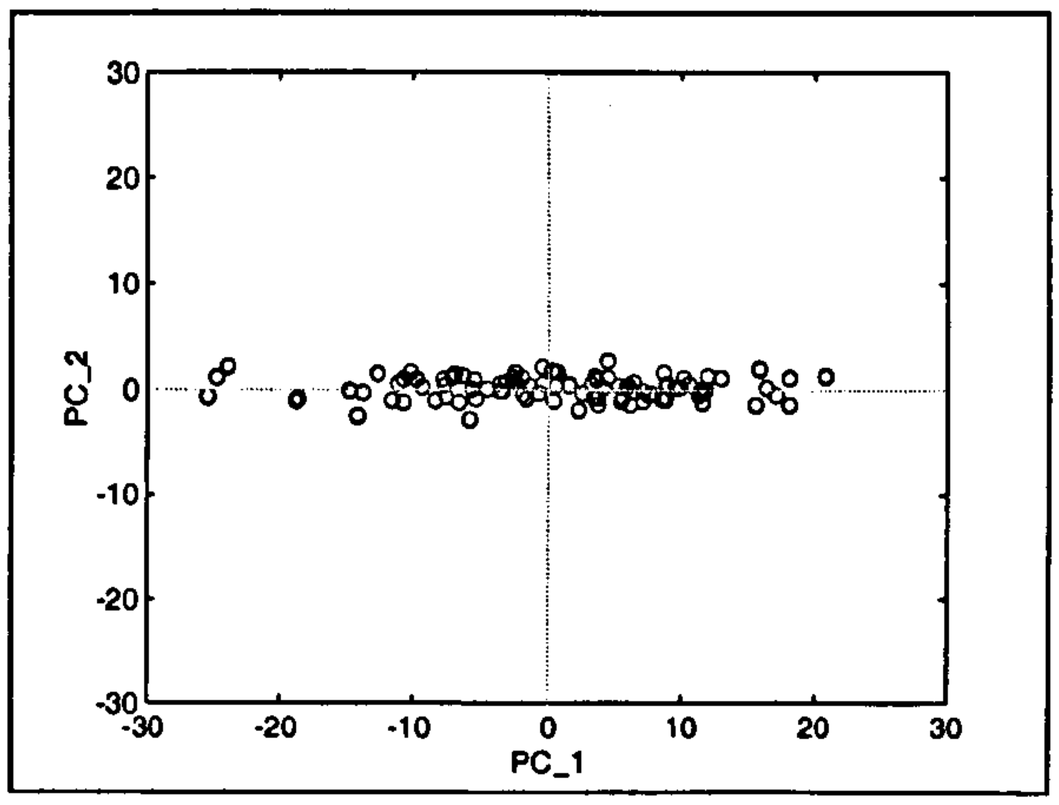

Figure 2.2b: Scores scatter plot for set_b given in Figure 2.2a.

Furthermore comparing Figure 2.2b with Figure 2.1b it can be observed that the scores on the first PC are stretched more for set_b than for set_a. In particular it can be noted the different scales on the axis for Var_1, [-8,8] in Figure 2.1b and [-30,30] in Figure 2.2b. As observed by Jolliffe (1986) this can be related to the magnitude of the corresponding first eigenvalues (5.4318 for set_a and 95.0894 for set_b) which is proportional to the spread of the scores along the corresponding principal axis. It should be also stressed that variance is an univariate statistic and that it does not account for the other variables in the data set, whilst PCA relates to the data set as a total entity and gives a global description of the data matrix. Which implies that variance and PCA are not fully compatible.

To address the issue of scale-invariance of PCA (i.e. reduce the sensitivity of the PCs to scaling factors) some kind of standardisation of the numerical values of the variables can be applied and various approaches exist in literature. One way, as observed by Mardia *et al.* (1979), would be to express all the variables in terms of the same "natural" units. However when the observed values are collected on a physical system which comprises a range of measurement groups it is not

always possible to identify a common "natural" unit which encompasses all the variables. Another way to express all the variables in a common measurement unit could be through the use of dimensionless numbers (although no suggestions were found in the literature with respect of this approach), but also in this case their calculation might not be easy to perform or, in the worst cases, realisable (because some of the required parameters might not be available).

The most common form of standardisation is to scale each variable $\boldsymbol{x}_m$ by a corresponding weight factor w_m:

$$\boldsymbol{x}_m^* = \frac{\boldsymbol{x}_m}{w_m} \qquad \textbf{2-1}$$

where $\boldsymbol{x}_m$ is the column vector comprising a set of N observations x_{nm} ($n \in [1{:}N]$) collected on the m-th variable of the data set, and $\boldsymbol{x}_m^*$ is the corresponding column vector comprising the scaled value for the m-th variable. Jolliffe (1986) suggested the use of scaling factors which reflect some *a priori* idea of the relative importance of the variables. Whilst Wold *et al.* (1987) suggested performing *blockwise scaling*, where blocks of different kind of variables exist and a similar type of scaling is applied to each individual block, so that the total variance is the same for each group of variables. However the most common approach is to mean-centre and scale the data matrix so that it has zero mean and unit variance. This can be achieved by centring each variable around its sample mean value $\bar{x}_m$ and dividing the centred value by the corresponding sample standard deviation s_m of the original variable:

$$\boldsymbol{x}_m^* = \frac{\boldsymbol{x}_m - \bar{x}_m}{s_m} \qquad \textbf{2-2}$$

where:

$$\bar{x}_m = \frac{1}{N} \cdot \sum_{n=1}^{M} x_{nm} \qquad \textbf{2-3}$$

$$s_m = \sqrt{\frac{1}{N-1} \cdot \sum_{n=1}^{N} \left(x_{nm} - \bar{x}_m\right)^2} \,. \qquad \textbf{2-4}$$

This approach leads to identifying the loading vectors with the eigenvectors of the correlation matrix rather than with the eigenvectors of the variance-covariance matrix of the given data set. This implies that the principal components computed from the correlation matrix cannot be directly related to those computed from the variance-covariance matrix.

The effect of using normalised variables (with zero mean and unit variance) can be observed in Figure 2.3 which was generated using the same data as for Figures 2.1 and 2.2. The normalised values of the two data sets (set_a and set_b) are exactly the same, since the only difference between them is a scaling factor applied to the first variable.

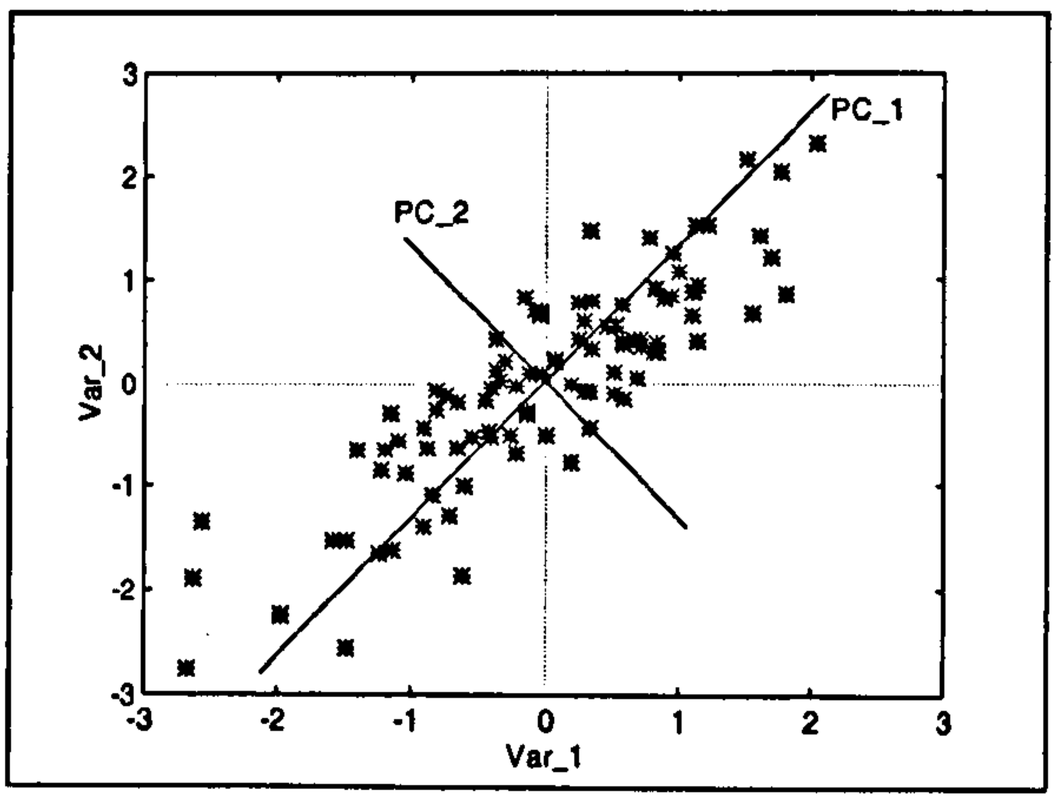

Figure 2.3a: Normalised variables scatter plot, with directions of the principal axes.

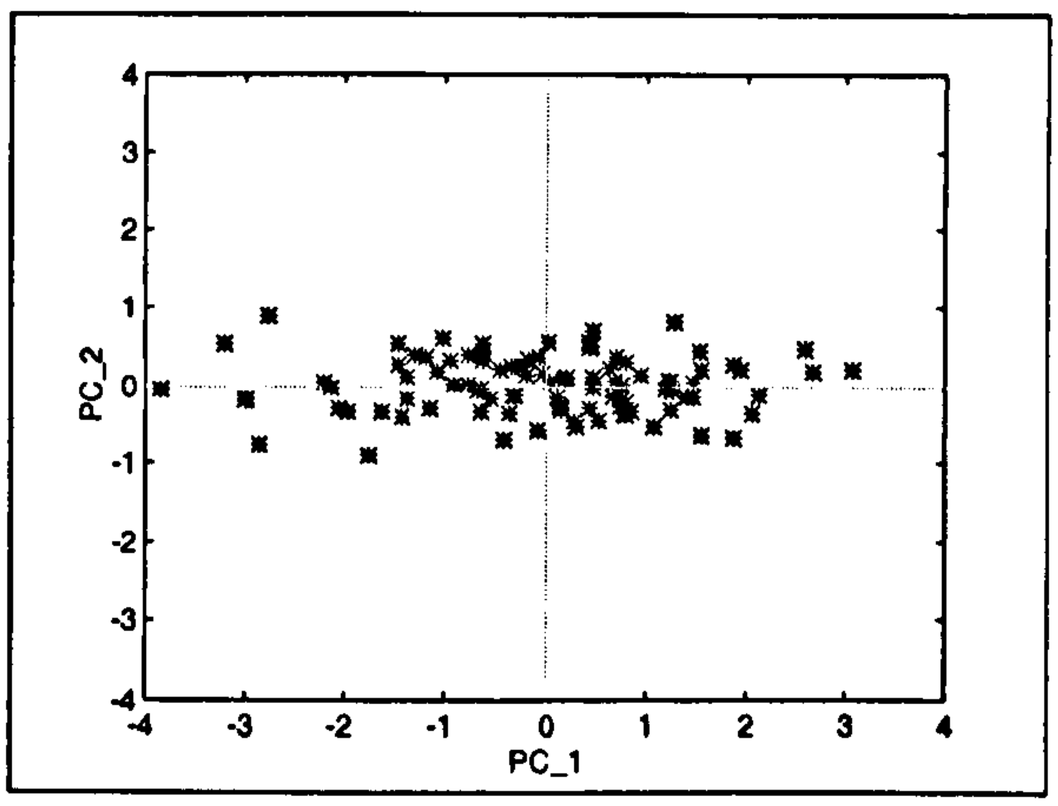

Figure 2.3b: Scores scatter plot for normalised variables given in Figure 2.3a.

Also, Jolliffe (1986) observed that if the variances of the variables are significantly different, performing PCA on the variance-covariance matrix reduces to a sorting algorithm for the original variables, and the PCs will hardly differ from the original variables other than to be rearranged in decreasing order of magnitude of their variances. This implies that the PCA representation would not contain information about the covariance structure of the data matrix. However when using the correlation matrix, the variables are comparable since they are normalised to unit variance. Consequently the representation of the data set given by the PCs computed on the correlation matrix is more informative of the correlation structure of the data matrix than that attained using the PCs computed on the variance-covariance matrix.

Jolliffe (1986) provided an additional argument for performing PCA on variables standardised to zero mean and unit variance. This is, the use of correlation matrices, rather than variance-covariance matrices, to define the PCs, makes the results of the analyses of different sets of data directly comparable. This is mainly due to the fact that when the variances of the variables differ significantly, the variance-covariance matrices are more sensitive than correlation matrices to the numerical values of individual variables.

Wold *et al.* (1987) observed that this form of standardisation (zero mean and unit variance) must be handled with care. In practice it makes all the co-ordinate axes have the same length, implying that in terms of correlation structure each variable has almost the same effect on the PC model. However if a variable has a standard deviation which is significantly "small" with respect to the standard deviation of the other variables (i.e. is almost constant), the standardisation would increase the importance of this variable. In the case where, for example, variation in a particular variable is mainly due to noise, through the standardisation procedure it would have the same importance as the other variables, which potentially carry information relevant to the behaviour of the system. The worst scenario is that it would partially mask the impact of these variables and the resultant model would be less accurate. To avoid this issue, Wold *et al.* (1987) proposed not scaling to unit variance those variables whose standard deviation is four times smaller than their measurement error (sensor noise), but applying scaling (standardisation) to the other variables.

One approach to scaling would be to use the correlation matrix in the first step of the analysis to obtain an understanding of the structures which exist within the data set and at this stage decide whether a different kind of scaling is required, according also to physical knowledge of the system.

For consistency with the most common approach cited within the literature, the data matrix X will be standardised to zero mean and unit variance within the section on PCA, unless stated otherwise. Therefore the loadings of each PC will be identified with the coefficients of the eigenvectors of the correlation matrix Σ:

$$\Sigma = \frac{1}{N-1} \cdot \left(X^T \cdot X \right). \qquad \textbf{2-5}$$

Constant variables are usually removed from the data set since they do not contribute to the covariance of the system and hence would not contribute to the PC model.

2.3 Mathematical Definition of Principal Components

As stated by Anderson (1984), principal component are normalised linear combination (the sum of squares of the coefficients being one) of random or statistical variables which have maximum variance amongst all possible linear combinations of the set of variables and which are themselves uncorrelated. Because of this, the principal components turn out to be the eigenvectors of the variance-covariance (or the correlation) matrix of the set of variables. In statistical practice, principal component analysis is the method used to find the principal components of a set of observations collected on a set of variables.

Thus, denoting the row vector containing the values of a set of M real variables x_m, $m \in [1: M]$, by $\boldsymbol{x}$ and the loading of the m-th real variable on the j-th PC by p_{mj}, the score t_j of the original variables x_m on the j-th PC can be written as:

$$t_j = \sum_{m=1}^{M} x_m \cdot p_{mj} \qquad \textbf{2-6}$$

or, in matrix notation:

$$t_j = \boldsymbol{x} \cdot \boldsymbol{p}_j \qquad \textbf{2-7}$$

where $\boldsymbol{p}_j$ denotes the (M×1) column vector containing the M loadings p_{mj} for the j-th PC.

Similarly, given an (N×M) data matrix $\boldsymbol{X}$ corresponding to a set of N samples collected on M variables x_{nm}, the score t_{nj} of the n-th set of values x_{nm} on the j-th PC can be written as:

$$t_{nj} = \sum_{m=1}^{M} x_{nm} \cdot p_{mj} \qquad \textbf{2-8}$$

and, in matrix notation:

$$\boldsymbol{t}_j = \boldsymbol{X} \cdot \boldsymbol{p}_j \qquad \textbf{2-9}$$

where $\boldsymbol{t}_j$ is the (N×1) column vector containing the N scores values t_{nj} on the j-th PC.

The score and the loading vectors $\boldsymbol{t}_j$ and $\boldsymbol{p}_j$ can be collected into two matrices $\boldsymbol{T}$ and $\boldsymbol{P}$, respectively, and the linear mapping provided by PCA can be defined as:

$$\boldsymbol{T} = \boldsymbol{X} \cdot \boldsymbol{P} \qquad \textbf{2-10}$$

where the (N×M) score matrix $\boldsymbol{T}$ is the representation of the (N×M) data matrix $\boldsymbol{X}$ in the M-dimensional orthogonal space identified by the (M×M) loading matrix $\boldsymbol{P}$. In particular, from a geometrical perspective, each column vector in the score matrix $\boldsymbol{T}$ represents the projection of the data matrix $\boldsymbol{X}$ on the j-th *principal direction* identified by the loading vector $\boldsymbol{p}_j$. The orthogonality of the transformation implies that it is invertible and that the inverse of the loading matrix is the same as its transpose, that is:

$$\boldsymbol{P}^{-1} = \boldsymbol{P}^{T}. \qquad \textbf{2-11}$$

Therefore the linear decomposition provided by PCA can be written as:

$$\boldsymbol{X} = \boldsymbol{T} \cdot \boldsymbol{P}^{T}. \qquad \textbf{2-12}$$

Expanding the matrix multiplication between the score and the loading matrices, the linear decomposition can be rewritten as a summation of the cross products between each pair of corresponding score and loading vectors:

$$X = \sum_{j=1}^{M} t_j \cdot p_j^T \,. \qquad \textbf{2-13}$$

Since the loading vectors p_j comprise the coefficients of the linear combinations corresponding to each PC, and therefore define the directions of the principal axis, the principal components model (PC model) is identified by the loading matrix P. Consequently when referring to the PC model, usually one refers to the loading matrix. Whilst the score matrix T is simply a representation of the data matrix X given by the PC model.

When a new data sample nx is available, it can be compared with the data used to build the PC model by projecting it down onto the PC space defined by the loading matrix P. The projection of the new sample on the j-th PC can be evaluated as:

$$nt_j = nx \cdot p_j \qquad \textbf{2-14}$$

where nx is the ($1 \times M$) row vector containing the new measurements, and a ($1 \times M$) row vector nt containing the scores nt_j can be defined as:

$$nt = nx \cdot P \,. \qquad \textbf{2-15}$$

Therefore the new sample can be compared with the samples in the original data matrix X by comparing its scores nt with the score matrix T, as described later (§ 2.8).

2.4 Some Properties of Principal Components

From the definition of the principal components of a standardised data matrix, a number of properties can be shown to hold for the score and the loading vectors, and the corresponding matrices. In particular, the mean value of each score vector is equal to zero and the variance is equal to the corresponding eigenvalue λ_j:

$$\mathrm{E}(\boldsymbol{t}_j) = 0 \qquad \textbf{2-16}$$

$$\mathrm{V}(\boldsymbol{t}_j) = \lambda_j . \qquad \textbf{2-17}$$

where $\mathrm{E}(\boldsymbol{t}_j)$ denotes the expected (mean) value of the score vector $\boldsymbol{t}_j$, and $\mathrm{V}(\boldsymbol{t}_j)$ its variance.

The constraint for each PC to be uncorrelated with the other PCs necessitates the score and the loading vectors to be orthogonal, thus for any $j \neq i$:

$$\boldsymbol{t}_j^T \cdot \boldsymbol{t}_i = 0 \qquad \textbf{2-18}$$

and:

$$\boldsymbol{p}_j^T \cdot \boldsymbol{p}_i = 0 . \qquad \textbf{2-19}$$

From a statistical perspective, the orthogonality between each pair of score vectors implies zero covariance between the two vectors, since they are mean-centred, and hence that the two PCs are uncorrelated. Whilst, geometrically and algebraically, the orthogonality between each pair of loading vectors implies the orthogonality of the corresponding principal axes and consequently the two PCs are uncorrelated.

By identifying the loadings p_{mj} with the direction cosines of the j-th principal axis, their squared values will sum to one, and hence the norm of the vector $\boldsymbol{p}_j$ will be equal to one:

$$\|\boldsymbol{p}_j\| = \sqrt{\boldsymbol{p}_j^T \cdot \boldsymbol{p}_j} = 1 \qquad \textbf{2-20}$$

which, coupled with the orthogonality of the loading vectors, leads to the orthonormality of the basis of the principal component space, represented by $\boldsymbol{P}$. The orthonormality of the loading matrix $\boldsymbol{P}$ implies the orthogonality of the transformation:

$$\boldsymbol{T} = \boldsymbol{X} \cdot \boldsymbol{P} . \qquad \textbf{2-21}$$

Thus, under the transformation given by $\boldsymbol{P}$, the two matrices $\boldsymbol{X}$ and $\boldsymbol{T}$ are invariant with respect to the *generalised variance*, i.e. the determinant of the variance-covariance matrix or of the correlation matrix, and the sum of the variances of the respective components, i.e. the trace of the correlation matrices (sometimes referred to as *total variation*):

$$\sum_{m=1}^{M} \mathrm{V}(\boldsymbol{x}_m) = \sum_{j=1}^{M} \mathrm{V}(\boldsymbol{t}_j) . \qquad \textbf{2-22}$$

Both these quantities are generally used as a single scalar measure of the spread of a multivariate data set.

This implies that the score matrix $\boldsymbol{T}$ preserves all the information contained in the data matrix $\boldsymbol{X}$, as expected since the transformation is invertible. The only difference is that the original variables may be more or less correlated whilst the PCs are themselves uncorrelated.

In particular, since the sum of the variances of the components $\boldsymbol{x}_m$ and $\boldsymbol{t}_j$ are invariant under the PC transformation, the fraction of variance of the data matrix $\boldsymbol{X}$ explained by each individual PC can be defined as the ratio between the variance of that PC and the sum of the variance of the original variables. However, since the variance of each principal component is equal to the value of the corresponding eigenvalue, the amount of variability explained or captured by each PC can be evaluated as the ratio between the corresponding eigenvalue and the sum of all the eigenvalues:

$$\frac{\mathrm{V}(\boldsymbol{t}_j)}{\sum_{m=1}^{M} \mathrm{V}(\boldsymbol{x}_m)} = \frac{\lambda_j}{\sum_{m=1}^{M} \mathrm{V}(\boldsymbol{x}_m)} = \frac{\lambda_j}{\sum_{j=1}^{M} \mathrm{V}(\boldsymbol{t}_j)} = \frac{\lambda_j}{\sum_{j=1}^{M} \lambda_j} . \qquad \textbf{2-23}$$

This can be simplified to:

$$\frac{\mathrm{V}(\boldsymbol{t}_j)}{M} \qquad \textbf{2-24}$$

where the sum of the variances of the standardised matrix $\boldsymbol{X}$ is equal to the number of variables M, since it is the sum of the auto-correlations of the M original variables. Therefore the amount

(proportion) of total variation accounted by the first K components can be quantified as the ratio between the sum of the first K eigenvalues and the sum of all the eigenvalues, i.e. the ratio between the sum of the first K eigenvalues and the number of variables M:

$$\frac{\sum_{j=1}^{K}\lambda_j}{\sum_{j=1}^{M}\lambda_j} = \frac{\sum_{j=1}^{K}\lambda_j}{M}. \qquad \textbf{2-25}$$

The PCs of a data matrix are arranged in decreasing order of magnitude of their eigenvalues, thus the first *few* PCs correspond to the largest eigenvalues and generally retain most of the variation (i.e. information or signal) present in the data matrix $\boldsymbol{X}$. The remaining PCs typically account for noise and, if some of the lower order eigenvalues are close or equal to zero then there exists a linear (or near-linear) combination (i.e. collinearities, relationships or structures) between the original variables (Jolliffe 1986). If some of the eigenvalues are equal to zero, the correlation matrix is *rank* deficient, i.e. its rank $R < M$ and thus the data matrix has only R linearly independent column vectors (assuming that the number of samples $N > M$). Therefore the total variation of the data matrix can be entirely explained by the first R principal components. From a geometrical point of view, the first few PCs identify the directions of greatest variability or spread of the data, while the lower order PCs identify directions in which there is little variation.

Thus when performing PCA on a data set which comprise a large number of noisy and linearly correlated variables, one of its main features is the ability to reduce the dimensionality of the data set, whilst retaining as much of the information as desirable. This reduction in dimensionality is achieved by means of the *representation* of the data given by the first few PCs, since they usually retain most of the variation and hence can be considered to be representative of the data set. The power of PCA lies in the fact that if the data set is noisy or highly correlated the number of principal components K required to describe the whole data set might be considerably smaller than the number of original variables M (i.e. $K << M$).

Furthermore the "original" data can be reconstructed by inverting the principal component transformation. The PCA decomposition of a data matrix $\boldsymbol{X}$ can be used to approximate the same matrix by means of the first K components:

$$\hat{X} = \sum_{j=1}^{K} t_j \cdot p_j^T \qquad \textbf{2-26}$$

where $\hat{X}$ denotes the approximated data matrix. The overall decomposition can be written as:

$$X = \sum_{j=1}^{K} t_j \cdot p_j^T + E_K \qquad \textbf{2-27}$$

where the residual matrix E_K represents the deviations between the original data matrix and its approximation:

$$E_K = X - \hat{X} = \sum_{j=K+1}^{M} t_j \cdot p_j^T . \qquad \textbf{2-28}$$

Selecting the number of components K to retain to account for most of the *relevant* variation in X is a critical issue in PCA. Several methods are available for this purposes and will be discussed in the section 2.6.

2.5 Sensitivity of PCA to outliers

When considering a data set, any sample which exhibits abnormal values for one or more of the variables it is usually considered an *outlier*, in the sense that it "appears" to be numerically inconsistent with the rest of the data set. This inconsistency might arise from a number of sources, such as recording errors which can lead to numerically incorrect values which do not correspond to real measurements, or genuine anomalous values which correspond to unusual physical conditions of the system.

With PCA being a least square method, outliers can have a major impact on the model, since they tend to influence the direction of the principal axes. This is especially true when the observations are numerically different from the rest of the data set. Therefore before undertaking a detailed analysis of the data, it is first necessary to investigate any anomalous sample(s). In particular, if the abnormality is due to recording errors, then the erroneous values should be

corrected, if possible. While if the measurements values are genuine, then an investigation of the samples is required, to decide whether or not they should be retained in the data set.

This last point is a very important issue in empirical modelling and data analysis. Removing samples from a data set only because they show atypical values with respect to the rest of the data is not advisable, since these samples might contain useful information about the system. The fact that a number of samples appear to be inconsistent with the rest of the data set does not imply that they are "*bad*" samples. Their connotation is to be "*different*" from the others. Only a detailed analysis of the data coupled with specific knowledge of the process (nominal operating conditions and their physical limits) and of the data acquisition system can help to establish the "*goodness*" of these samples with respect to rest of the data. Thus the first and apparently simple operation to perform before undertaking any analysis is to identify and isolate any sample whose values appear to be atypical with respect to the rest of the data set.

This issue is more complicated than it seems to be. In fact, in multivariate data, atypicality of data samples can arise in a number of different ways, it could be due to the abnormal value of a single variable or due to a combination of (apparently) normal values of different variables. Furthermore each form of inconsistency generally requires a different approach to ensure its detection. Therefore in many practical situations, one of the major issues is to define what is to be considered atypical in the data set and for the scope of the analysis.

When considering outliers as those observations that are "*far away*" from the majority of the data set, some kind of distance measurement (e.g. Euclidean or Mahalanobis distance) can be used for their identification. A number of useful references and related statistics for outliers detection can be found in Krzanowski *et al.* (1994) and Jolliffe (1986). Nevertheless, since PCA can be affected by the presence of outliers in the data set and because of this, its sensitivity can be used as a tool for outlier detection (§ 2.8.3)

2.6 Selecting the number of PCs

When dealing with approximation or regression techniques, the objective is to produce a method which is capable of providing a "*valid*" representation of a given data matrix. A suitable measure

for the validity of a model might be the fraction of information of the original data matrix which can or cannot be reproduced by the model. For example the fraction of the total variation of the original data matrix captured by the model, or the sum of squared residuals.

In PCA, the main issue is to determine the number of principal components that are required to "*adequately*" account for the total variation in the data matrix X. Furthermore, the definition of what is an "*adequate*" or "*valid*" representation of the data set is necessary, and this depends upon the purpose of the principal component decomposition. In fact, if the aim is to capture as much of the total variation of the data matrix as possible, the solution would be to use all the components, therefore explaining 100% of the total variation.

However in most applications of PCA this is not the case. The aim of applying PCA is typically to explain a *sufficiently* large amount of the total variation of the data matrix, whilst retaining only a few principal components. Thus the final application of the PC model must be considered when defining what is a "*sufficient*" amount of total variation which should be captured by the PC model and hence choose the appropriate criterion for selecting the number of components to retain in order to capture that level of total variation. If the purpose of the PC model is to compress the data matrix X, it is only necessary to define in advance the amount of total variation which should be retained in the model. However, if the model is to be used as an approximate representation of a system for data analysis or process monitoring it must satisfy a more rigorously defined optimality constraint. A number of suggestions on how to address this problem can be found in Jolliffe (1986). Some of the more common criterion are described hereafter.

A simple way to choose the number of components to retain in the model is to define a cut-off limit for the cumulative percentage of total variation which should be captured by the PC model (in many practical situations it might be somewhere between 70% and 90%). Consequently, the required number of components to retain in the model is the smallest for which the chosen percentage is exceeded. The major drawback of this approach is the *a priori* assumption that the pre-specified amount of total variation matches the application requirements, leaving a certain degree of subjectivity in the criterion. When working with correlation matrices instead of variance-covariance matrices it is possible to define a more rigorous rule for fixing a cut-off limit, this is retaining those components with eigenvalues larger than 1. The idea behind this is

that any PC with a variance of less than 1 contains less information than the original variables, which have been scaled to unit variance. However, it can be argued that a cut-off limit of 1 can result in too few components being retained. Jolliffe (1986) then suggested using a limit of 0.7. The rule can be modified further to reflect the distribution of the eigenvalues of the correlation matrix, selecting the cut-off limit corresponding to the mean value of the eigenvalues, or a suitable fraction of this.

Another fairly subjective criterion involves looking at the plot of the eigenvalues against the number of components, the "*scree*" plot, and deciding at which component the slopes of the lines joining the plotted values of the eigenvalues are "*steep*" to the left and "*not steep*" to the right. The aim is to identify the component at which the line joining the eigenvalues changes slope, and thus where the performance of the model changes drastically. In particular a steep slope implies that that component is still extracting a considerable amount of the total variation, while a not steep (gentle) slope is indicative of components which are extracting only a small amount of information. If more than two steep slopes are present, it will be necessary to decide where to place the cut-off limit. Furthermore, such behaviour is also a symptom of the existence of more than one major underlying structure, and further analysis of the data is required before building the final model.

The selection criteria described so far require a certain degree of subjectivity when selecting the cut-off point. However, in more demanding situations, where the PC model might have to be used for data analysis or process monitoring, it is necessary to establish a more rigorous and objective criterion for selecting the number of components to retain in the model. For example when using the PC model in data analysis, it is necessary to identify the number of components which capture the structural information present in the data set, leaving out random disturbances and noise. In this case the main issue is to provide a different but hopefully more appropriate representation of the data set in terms of the score and loading vectors.

Similarly when developing a PC model with the purpose of monitoring process performance, the main issue is to provide a representation of the underlying structure of the process and, to a certain extent, of the specific structure of the data matrix used to build the model. In this case, the objective of the model is to compare new observations with those used to build the model, which generally correspond to nominal working conditions. Consequently the level of variability

(and hence of specific information) of the training data set that should be retained in the final model depends upon the nature of the data and the specific use of the model.

Another issue is that of avoiding overfitting of the data matrix used to build the model. This is of critical interest for all empirical modelling approaches, since the final model is required to be able to generalise to new data drawn from within the operating region of interest. The data matrix used to build the model comprises information relating to the system but specific to the working conditions in which the data was collected. Therefore, if interest is focused on the underlying structure of the system, the model should not retain information which might be specific to the data matrix, such as measurement noise and process disturbances.

Wold (1978) and Eastman and Krzanowski (1982) proposed the use of *Cross Validation* to address this problem in PCA. This was originally developed by Mosteller and Wallace (1963) and explored in detail by Stone (1974). The idea behind it is simple, but extremely effective in terms of model validation. The standard procedure can be described as follow: the data matrix used to build the model is randomly partitioned into a number of groups and each group is deleted in turn from the original data matrix to form a reduced data matrix. Parameters for different models are estimated from the reduced data set and tested on the deleted subset according to some optimising criterion which gives a statistic or a measure of the "*goodness*" of the parameters. The deleted group is then restored in the data matrix, a new group deleted and the procedure repeated. This search is performed in an exhaustive way, until each group has been deleted once from the data matrix. The statistics calculated for each subgroup are then used to give a total measure of goodness of the different models. Selection of the best model structure is then based upon error measurements.

A "*criterion of goodness-of-fit*" (CGF, Wold, 1978) between the model and the data must first be defined before performing the procedure. Generally the sum of squared prediction error (PRESS) on the unseen data is employed as a measure of the goodness of the model. In particular, in PCA the optimisation parameter is the number of components K to retain in the model to attain a minimum error when using the PC model to approximate unseen data. Thus when performing cross validation in PCA, the data matrix is partitioned providing a number of training data matrices and corresponding testing data matrices. A different PC model is built for each training data matrix and the deleted subset is projected down onto this PC space and predicted. The prediction error is then computed using different number of components for its

approximation. When all the subsets have been taken out of the data matrix and used to test the corresponding model, the total PRESS is computed:

$$\mathrm{PRESS}(k) = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M} e_{nm,k}^2}{N \cdot M} \qquad \textbf{2-29}$$

where $e_{nm,k}$ denotes the mismatch of the i-th observation of the j-th variable given by the model based upon the use of the first k principal components:

$$e_{nm,k} = x_{nm} - \hat{x}_{nm,k}\,. \qquad \textbf{2-30}$$

N and M are the number of observations and the number of variables comprised in the original data matrix, respectively. The final PC model can be built using the original data matrix using the number of components corresponding to the minimum value of the total PRESS (Stone, 1974) or to some value of a function of the PRESS. For example Wold (1978) proposed the use of the ratio:

$$R(k) = \frac{\mathrm{PRESS}(k)}{\sum_{n=1}^{N}\sum_{m=1}^{M}\left(x_{nm} - \tilde{x}_{nm,k-1}\right)^2} \qquad \textbf{2-31}$$

where $\tilde{x}_{nm,k-1}$ denotes the approximation of x_{nm} given by the first (k-1) principal components of a PC model built using the entire data set. This quantity compares the approximation on unseen data given by a model based upon the use of the first k components with the approximation of seen data given by a model based on the first (k-1) components. If $R(k) < 1$ the implication is that a better prediction is achieved using k rather than (k-1) components, so that k PCs should be included in the PC model.

On the other hand, Eastman and Krzanowski (1982) proposed the use of the ratio:

$$W(k) = \frac{\mathrm{PRESS}(k-1) - \mathrm{PRESS}(k)}{\mathrm{PRESS}(k)} \cdot \frac{M \cdot (N-1)}{N + M - 2 \cdot k} \qquad \textbf{2-32}$$

which represents the increase in predictive information given by the k-th component divided by the average predictive information in each of the remaining components. Thus significant components should lead to values of $W(k) > 1$.

When performing cross validation the data matrix can be divided in different ways. The simplest approach, which might be called "*sample-wise*" selection, is to select groups of rows and alternately delete them from the data matrix. In particular, if the data matrix contains samples collected in stationary conditions, and thus which do not show any serial correlation between them, the selection of the subsets can be performed by randomly selecting individual samples, disregarding their sequence in the data matrix. However, if the data contains any dynamic structure which cannot be ignored, then it is necessary to divide the data matrix into a number of subsets of contiguous samples. In fact, by randomly shuffling time-correlated observations the dynamic dependencies within the data are destroyed, leading to models which cannot be considered to be fully representative of the data set.

Eastman and Krzanowski (1982) proposed deleting one element x_{ij} of the data matrix at a time, and hence it might be called "*element-wise*" selection. This implies predicting the missing element using all the information available, whilst remaining strictly independent of the deleted observation, but this leads to the difficulty of building a PC model on an incomplete data matrix. The approach is more complicated and time consuming than the *sample-wise* selection and an efficient algorithm is necessary. Nevertheless the authors propose the use of two singular value decompositions, omitting respectively the i-th row and the j-th column of the data matrix, to predict the missing element. Details can be found also in Krzanowski (1987) and Krzanowski and Marriott (1994).

The model structure (i.e. the number of components to retain) identified by performing cross validation and corresponding to the minimum value of the total PRESS is the "*best*" in terms of the approximation of unseen data. However it would not necessarily correspond to the best model structure when considering the approximation of the original data matrix. This occurrence leads to a major feature of a PC model validated using cross validation. Since cross validation is able to identify the number of components which gives a generic description of the system, the dimension of the PC space identified by cross validation is the one for which the PC model extracts the maximum fraction of the relevant information contained in the data set, without

capturing any of the random disturbances or noises, and hence without overfitting the data matrix. This feature leads to the use of PCA in combination with cross validation as a tool for noise filtering (§ 2.8.5).

2.7 Algorithms for Extracting PCs of a Data Matrix *X*

A number of algorithms are available in the literature to extract the principal components of a data matrix. In particular, because of the analogy between PCA and the eigenstructures of correlation matrices, any algorithm providing either spectral decomposition or singular value decomposition (Appendix 1) can be used to determine the eigenstructure of the correlation matrix Σ, and hence the principal components of the corresponding data matrix X. Both algorithms can be used to evaluate the loading matrix P, whilst the score matrix T has to be computed.

The calculation of the eigenstructure of a data matrix is a laborious computation procedure, which requires the solution of a polynomial equation of the same degree as the dimension of the matrix. Nevertheless, with modern computers, computation time is no more a major limitation in the application of multivariate statistical techniques. Furthermore almost every statistical package for data analysis provides the calculation of the principal components by means of fast and reliable algorithms.

For convenience only the *power method* described by Hotelling (1933) in his paper on PCA and the Non-linear Iterative Least Square (NILES) algorithm proposed by Wold (1966a,b) will be described. However faster and more stable algorithms are available, but the description of these other algorithms is beyond the intent of this work.

2.7.1 The Power Method

The power method was described by Hotelling (1933), and an accelerated version of the algorithm was subsequently presented by Hotelling (1936). It is an iterative technique which simultaneously yields both the eigenvalue and the eigenvector of a square matrix, and hence the loading vectors for the corresponding PC. The algorithm estimates the eigenvalues in order of

magnitude, the largest first. This feature makes it possible to stop the procedure when it is evident that the number of principal components is explaining a satisfactory fraction of the total variation of the original data set. This can be simultaneously checked since the sum of the first K eigenvalues gives the fraction of the total variation of the data set captured by the combination of the first K principal components (§ 2.4).

In its simplest form the power method is an iterative procedure for finding the largest eigenvalue and the corresponding eigenvector of an ($M \times M$) matrix $\boldsymbol{A}$. The general idea behind it is based on the definition of an eigenvector $\boldsymbol{v}$ of a square matrix $\boldsymbol{A}$ as that normalised (to unit norm) M dimensional vector which maps onto itself or multiples of itself under the transformation given by the matrix $\boldsymbol{A}$:

$$\boldsymbol{A} \cdot \boldsymbol{v} = \lambda \cdot \boldsymbol{v} \qquad \textbf{2-33}$$

where the scalar factor λ is the corresponding eigenvalue (Appendix 1).

The algorithm can be implemented by selecting an initial M dimensional vector $\boldsymbol{u}_{(0)}$ scaled to unit norm and starting the iterative procedure (with iteration index i). The algorithm computes a series of vectors $\boldsymbol{u}_{(i)}$ given by the equation:

$$\boldsymbol{u}_{(i)} = \boldsymbol{A} \cdot \boldsymbol{u}_{(i-1)} \qquad \textbf{2-34}$$

which are scaled to unit norm:

$$\boldsymbol{u}_{(i)} = \boldsymbol{u}_{(i)} / \left\| \boldsymbol{u}_{(i)} \right\| \qquad \textbf{2-35}$$

to give a series of corresponding scalar parameters $l_{(i)}$:

$$l_{(i)} = \boldsymbol{u}_{(i)}^{T} \cdot \boldsymbol{A} \cdot \boldsymbol{u}_{(i)} . \qquad \textbf{2-36}$$

The overall sequence of operations is then repeated until the difference between two successive values of $l_{(i)}$ does not fall below a predefined tolerance limit. It can be shown that as the

iteration index i tends towards infinity the sequence of scalar quantities $l_{(i)}$ tends to the largest eigenvalue λ_1, and consequently the sequence of vectors $\boldsymbol{u}_{(i)}$ tends to the corresponding first eigenvector $\boldsymbol{v}_1$.

Hotelling (1933) adapted the power method to sequentially extract the PCs of a data matrix $\boldsymbol{X}$ by applying it to the correlation matrix Σ. In particular, once the first eigenvalue and the corresponding eigenvector have been extracted, the correlation matrix Σ can be *deflated* by subtracting from it the variance explained by the first component ($\lambda_1 \cdot \boldsymbol{p}_1 \cdot \boldsymbol{p}_1^T$):

$$\Sigma_1 = \Sigma - \lambda_1 \cdot \boldsymbol{p}_1 \cdot \boldsymbol{p}_1^T \qquad \textbf{2-37}$$

where Σ_1 is the deflated correlation matrix, i.e. the correlation matrix of the residuals after one component has been extracted (§ 2.4). Hence, the power method can then be applied to Σ_1 to extract the second component and the overall procedure can be repeated until all the PCs have been extracted.

The algorithm is usually fast and works well if the difference between two subsequent eigenvalues λ_j and λ_{j+1} is not small, i.e. if $\lambda_j >> \lambda_{j+1}$, otherwise it will converge slowly. The same problem arises when the data matrix is strongly correlated and some of the final eigenvalues are zero or close to zero. In this case, the method becomes sensitive not only to the similarity between the eigenvalues but also to rounding errors in the calculations.

The computation of the last components can be avoided if not all the PCs are necessary for the PC model. In this case the overall procedure can be stopped when the required fraction of the total variation has been captured. Otherwise it is possible to apply a transformation to the correlation matrix to simplify the calculation of the eigenvectors, e.g. shifting and inverting the correlation matrix (Jolliffe, 1986). A description of the accelerated algorithm can be found in Hotelling (1936) and Jolliffe (1986).

2.7.2 Non-linear Iterative Least Square (NILES) Algorithm

The NILES algorithm was originally proposed by Wold (1966a,b). It is a simple iterative procedure which sequentially estimates one pair of corresponding score and loading vectors at a time. In its simplest form the algorithm can be used to extract the factor corresponding to the largest eigenvalue of a data matrix. However, it is also suitable for extracting the subsequent components by repeating the procedure on the residual matrix obtained after the estimation of the previous components, i.e. to the data matrix deflated by the contribution of the previous components.

To facilitate the description of the algorithm, the deflated X matrix after a components have been extracted will be termed X_a:

$$X_a = X_0 - \sum_{j=1}^{a} t_j \cdot p_j^T \qquad \textbf{2-38}$$

where X_0 corresponds to the original data matrix X. In particular for each component a, the NILES algorithm estimates the score and loading vectors t_a and p_a from the residual matrix obtained after the estimation of the previous (a-1) components, i.e. X_{a-1}.

The procedure starts with a guess for the score vector t_a. During each iteration the estimate of the loading vector is improved by regressing X_{a-1} on the previous score vector estimate:

$$p_a^T = \frac{t_a^T \cdot X_{a-1}}{t_a^T \cdot t_a} \qquad \textbf{2-39}$$

and the estimate of the score vector is improved by regressing X_{a-1} on the improved (previous) loading vector estimate:

$$t_a = \frac{X_{a-1} \cdot p_a}{p_a^T \cdot p_a} \qquad \textbf{2-40}$$

until convergence is achieved. During each iteration the loading p_a vector is scaled to unit norm prior to updating the score vector t_a. Once the algorithm converges to the score and loading

vectors t_a and p_a, for that particular principal component, a new deflated (residual) matrix X_a is computed:

$$X_a = X_{a-1} - t_a \cdot p_a^T \qquad \textbf{2-41}$$

and the overall procedure is applied to it to extract a further component, if required.

Wold (1966a,b) refers to the procedure as "*criss-cross regression*" of the data matrix on both the loading and the score vectors (Equations 2-39 and 2-40).

For each component *a* the overall procedure can be described as follow:

0.	Initialise t_a to some column vector of X_{a-1}, e.g. to the vector corresponding to the variable with the largest variance.	
1.	Estimate the loading vector p_a by regressing the columns of X_{a-1} on the score vector t_a:	$p_a^T = \frac{t_a^T \cdot X_{a-1}}{t_a^T \cdot t_a}$
2.	Scale the loading vector to unit norm	$p_a = \frac{p_a}{\|p_a\|}$
3.	Estimate the score vector t_a by regressing the columns of X_{a-1} on the loading vector p_a:	$t_a = \frac{X_a \cdot p_a}{p_a^T \cdot p_a}$
4.	Check convergence on t_a, if the difference with respect to the previous iteration is larger than a predefined tolerance limit go to step 1 else go to step 5.	
5.	Deflate data matrix X_{a-1}.	$X_a = X_{a-1} - t_a \cdot p_a^T$

Table 2.1: NILES Algorithm for PCA.

If $a = 0$, set X_0 equal to X. If another component is required the overall procedure is repeated using the deflated matrix X_a.

Martens and Næs (1989) observed that the NILES algorithm is based upon the fact that the principal components are orthogonal in terms of both score and loading vectors. This implies that each pair of score and loading vectors account for a different source of variability in the data matrix. Therefore sequentially estimating a new component from the residual matrices implies modelling that part of the total variation which cannot be modelled by the previous components.

2.7.3 Some comments on the power method and the NILES algorithm

The general idea behind the power method and the NILES algorithm is to sequentially extract information (i.e. identify a principal component) from the data matrix, deflate it and perform the same operation on the residual matrix. They both operate a sort of "*peeling*" process of the data matrix, where the end result is to model a certain amount of the information present in the data and subtract it from the data matrix. In both cases, the modelling phase is subject to maximising the fraction of total variation which can be modelled by each individual principal component.

The main difference between the two algorithms lies in the fact that the power method operates on the residual variance-covariance matrices, whilst the NILES algorithm acts on the residual matrices. Consequently one of the major advantages of the NILES algorithm, as observed by Martens and Næs (1989), is the possibility to work directly on the data matrix $\boldsymbol{X}$ and the associated residual matrices, as opposed to using the more abstract cross-product tables, as in the power method approach.

However this feature has a major drawback in terms of computational expense and memory requirements when performing PCA on a computer. When handling data sets which comprise a large number of samples, $N >> M$, the procedure can become unsuitable for low memory machines, since it requires a number of expensive matrix manipulations of an $(N \times M)$ matrix, whilst the power method requires the manipulation of a smaller $(M \times M)$ matrix. Nevertheless this does not represent a real limitation with modern machines.

2.8 The Use of PCA in Data Analysis

Because of its properties, PCA represents an efficient investigative tool for data analysis. In particular, it can be used for data pre-screening before carrying out any analysis or regression on the data. Some of the more common uses of PCA are reviewed hereafter.

2.8.1 Data Compression

In many practical situations, large data sets need to be stored for future applications, but the space required can represent a limitation. However, by retaining the first few score vectors given by PCA, a reliable representation of the data matrix can be obtained. This is particularly true if the data are linearly correlated or if not all of the information comprised in the data set needs to be stored. In this case the first few PCs can be used to compress the original data set into a smaller matrix given by the first few significant score vectors. Furthermore, this reduced score matrix can also be used, in conjunction with the corresponding loading matrix, to approximate the data matrix itself (Equation 2-26). If the data matrix has rank R, the first R principal components (corresponding to the non-zero eigenvalues of the variance-covariance matrix) capture all the information contained in the data matrix. Therefore the data matrix can be entirely represented by means of the linear decomposition given by the first R score and loading vectors:

$$\boldsymbol{X} = \sum_{j=1}^{R} \boldsymbol{t}_j \cdot \boldsymbol{p}_j^T \,. \qquad \textbf{2-42}$$

Consequently, if the original variables are linearly correlated, the number of non-zero eigenvalues will be smaller than the number of variables and it is possible to represent the ($N\times M$) data matrix $\boldsymbol{X}$ by means of the smaller ($N\times R$) score matrix, and the corresponding ($M\times R$) loading matrix, which comprise respectively the first R score and loading vectors.

Furthermore, if only a major fraction of the total variation of the data set needs to be stored, the number of components required can be reduced further. This reduction can be achieved, for example, by fixing the desired cut-off limit on the cumulative percentage of total variation which can be captured by the PC model and accordingly selecting the number of components, or by using cross validation.

2.8.2 Cluster Analysis and Pattern Recognition.

Perhaps one of the most common uses of PCA is as a graphical tool for the investigation of a data matrix through plots of the score and loading vectors. PCA can then be used for identifying relationships between samples or variables by examining trends and clusters in the score or the loading plots, respectively. Each score vector is the projection of the X data matrix onto the principal axis identified by the corresponding loading vector (§ 2.3 and 2.4) and can be used as a univariate representation of the multivariate data set X. The power of PCA as an investigative tool is its ability to provide a simple representation of a multivariate system. Individual time series plots of the score vectors can be used to identify trends in the data matrix which might be difficult to recognise when using individual time series plots of the original variables (Figure 2.4a).

Generally time series plots based upon the first few components can be used to summarise major changes in the data set. Similarly scatter plots can be used to identify clusters of points or multivariate outliers (Figure 2.4b). The scores depicted in Figure 2.4 correspond to a practical application of PCA for analysing an industrial process. The data set used for this analysis consisted of 2036 samples collected on 41 variables. The data were supposed to have been collected during similar working conditions, however the scatter plot of the score for the first two PCs (Figure 2.4b) identified differences within the data set. The first two components captured 41 % of the total variation of the scaled data set. Three clusters of samples and a number of observations which do not belong to any of the three clusters were identified. Similarly the time series plot for the first PC (Figure 2.4a) provides an indication of major changes in the process, with respect to the sample number, and can be used to identify the periods when the process was affected by these changes.

In a similar way the loading vectors can be used to gain a better insight into the data matrix, and consequently into the process. A bar chart representation of the loading coefficients can be very useful to visualise the relative importance of the variables on each PC, especially when the number of original variables in the data matrix is large (Figure 2.5a). The closer the loading coefficient to unity, the great is the effect of the corresponding variable on the PC. Similarly loading scatter plots (Figure 2.5b) can be used to help identify clusters of variables which exhibit similar behaviour. Variables clustering near to the centre of the scatter plot do not have any numerical contribution on that particular PCs. On the other hand, variables clustering away

from the centre are representative of variables exhibiting positive correlation (e.g. variables 18 and 20 in Figure 2.5b). Their influence on the PCs being proportional to their distance from the centre of the scatter plot. Furthermore, variables clustering in diametrically opposite quadrants potentially exhibit negative correlation (e.g. variables 18 and 20 on one side and variable 4 on the other side in Figure 2.5b).

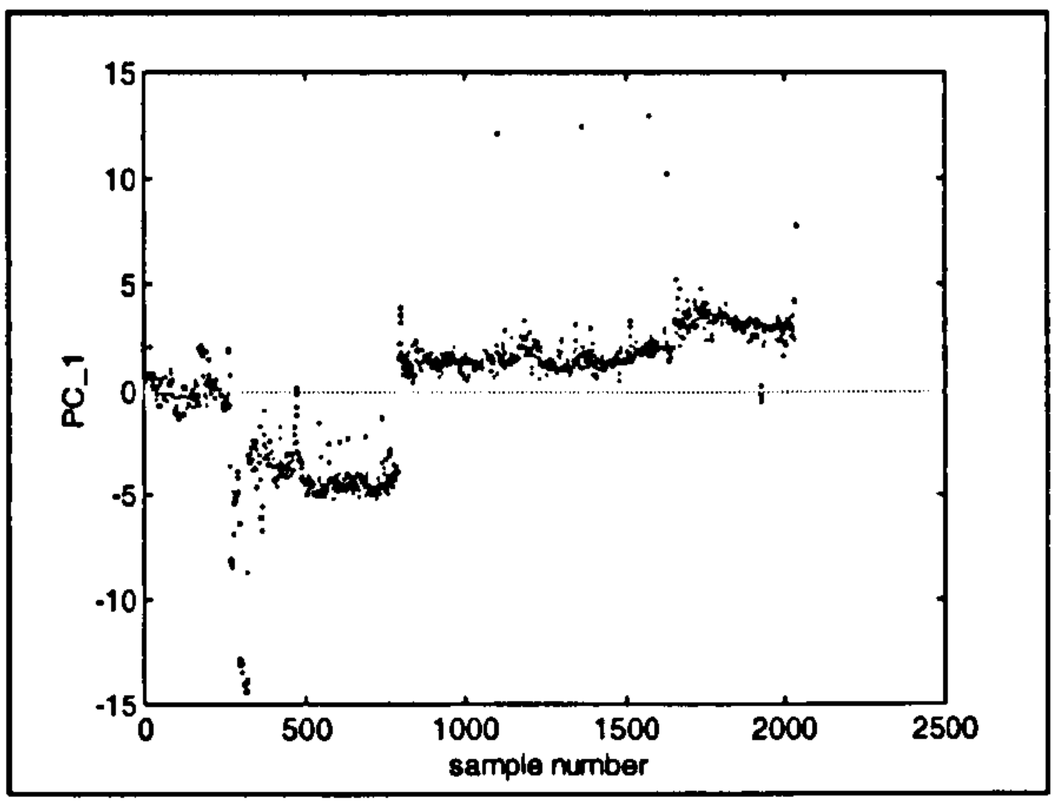

Figure 2.4a: Scores time series plot for PC_1 for industrial data.

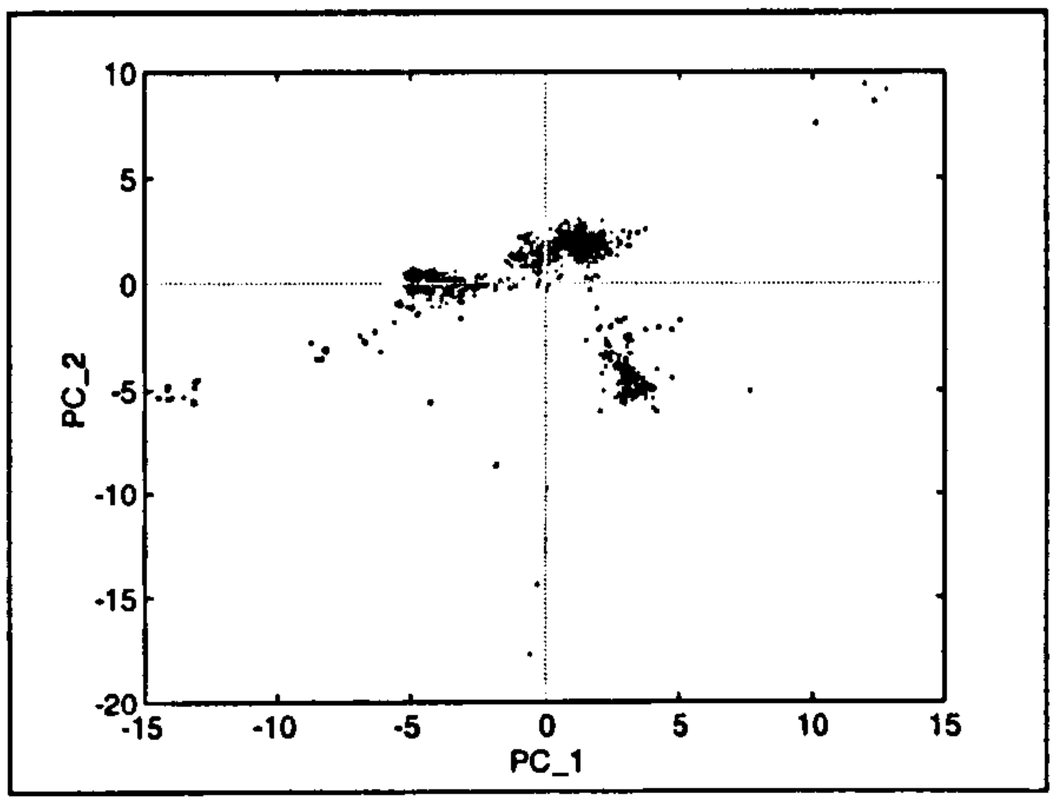

Figure 2.4b: Score scatter plot for PC_1 versus PC_2 for industrial data.

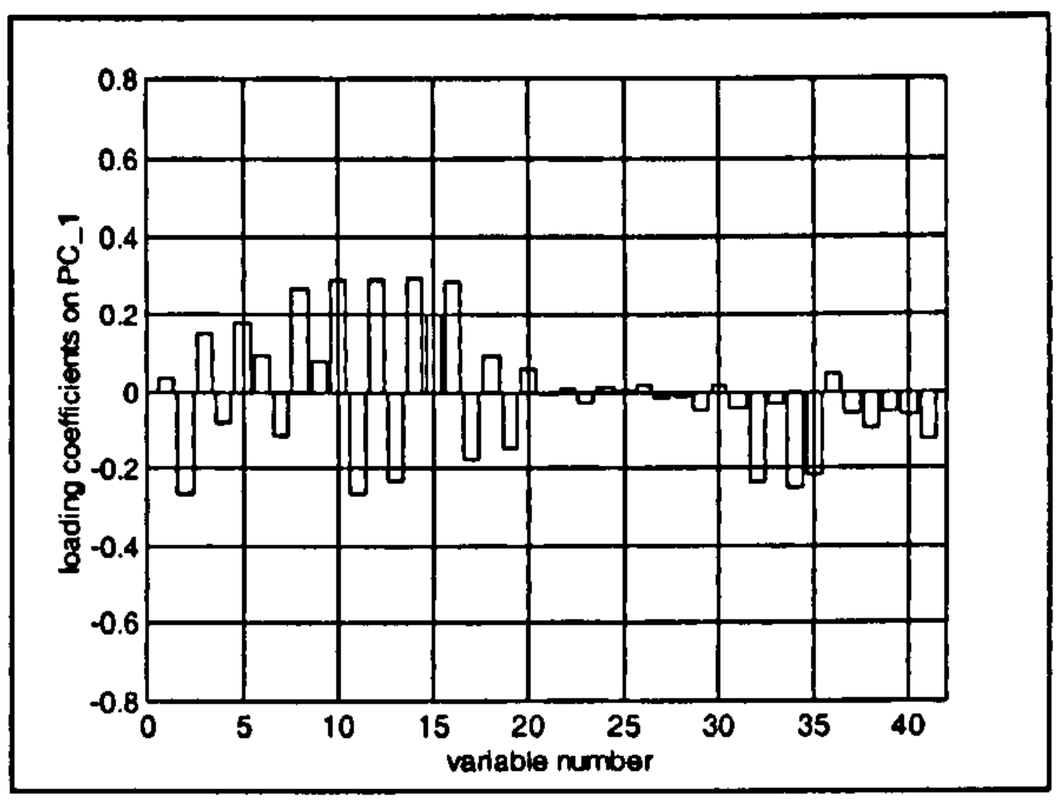

Figure 2.5a: Loadings bar chart for PC_1 for industrial data.

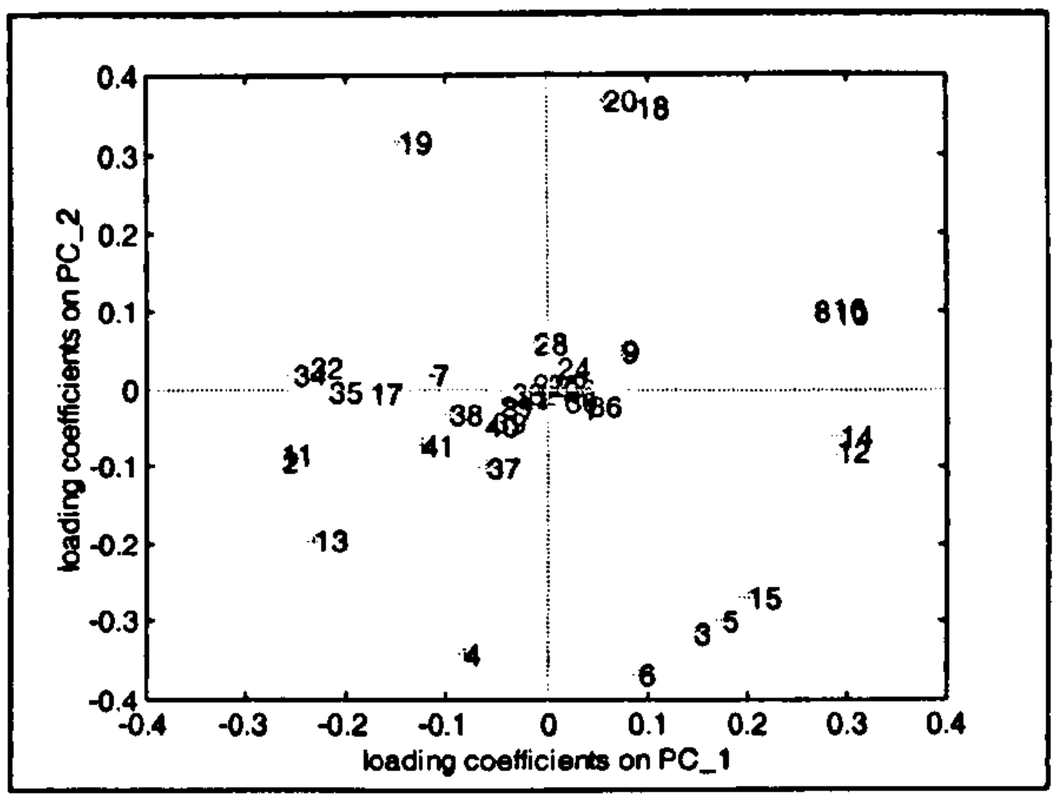

Figure 2.5b: Loadings scatter plot for PC_1 versus PC_2 for industrial data.

2.8.3 Outliers Detection

As observed before (§ 2.5) outliers represent a critical issue in empirical modelling since their presence or absence can affect the structure of the model. However, as Jolliffe (1986) observed, there is no formal and widely accepted definition of what is meant by "outlier". Furthermore outliers can be of different types, which complicates the identification of these atypical observations. Nevertheless PCA is a useful graphical tool for outlier detection and Jolliffe (1986) gave a detailed description of the use of PCA for this purpose.

When defining outliers as those samples which do not belong to any known cluster observed in the data, PCA can be used to identify different types of outliers through score time series plots and score scatter plots. In fact outliers occurring in the first few PCs are different from those which materialise in the lower order PCs. In particular outliers detected from the first PCs are those which *inflate* variances and covariances. They are the cause of a large increase in the variance of one or more of the original variables and can also be detected by looking at the plots of the original variables. Outliers which occur in the lower order components are multivariate outliers and are not apparent with respect to the original variables and consequently cannot be detected by looking at the original variables. Similarly loading scatter plots can be used to identify variables which show atypical behaviour with respect to the other variables of the data set. In both cases ascertaining that a sample or a variable is an outlier with respect to the data matrix does not imply that it should or can be removed from the data set without further investigation.

2.8.4 Variable Selection

In many situations an important issue is to determine the impact of each of the measured variables on the system (e.g. when the behaviour of a process has to be analysed in order to obtain a better understanding, or to design a new control strategy). In such situations PCA represents a useful tool for identifying the relative importance of the variables collected on the system since the loading coefficients are representative of the importance (weight) of each variable on individual components. Furthermore the importance of a variable in a PC model (representation) is indicated by the size of its residual variance, i.e. the variance of its residuals in the residual matrix as defined in Equation 2-28. The smaller the residual variance, the great the importance of the variable in determining the model structure. Generally the influence of each real variable can be investigated by developing PC models based on the first few PCs

(eventually cross validated), which can be considered as representative of the whole data set. It must be bore in mind that the results of the analysis are related to the data matrix which is considered to be representative of the process.

However, not only are the first few PCs important for the description of the data set. Any PC with zero variance (and hence zero eigenvalue) identifies a linear relationship between the original variables (Jolliffe, 1986). This means that some variable is redundant and its value can be determined by solving the linear combination corresponding to that component, i.e. using the coefficients of the corresponding loading vector. Thus one of the variables having non-zero coefficient in the loading vector can be expressed as a linear combination of the other variables, and consequently might be removed from the data set with negligible loss of information. This use of PCA can lead to a major problem which is the criterion to use for selecting the variables to remove from the data set. One approach might be to choose the variable with the smallest coefficient in the first few PCs, since it does not appear to be important in terms of the proportion of total variation explained. Another criterion might be to exclude the variable with the largest residual variance when using a specified number of components to approximate the data matrix, because again it does not appear to affect the PC model. A more rigorous approach would be to verify the loss of the approximation capabilities of the PC model, i.e. variance explained, by removing each of the candidate variables.

However, it can be argued that each variable contributes a certain amount of information to the model. This information may be useful at some point in the analysis. Therefore it does not seem appropriate to exclude variables from the data set solely as a consequence of a statistical analysis without having first tested the effect of its removal in terms of its impact on the performances of the PC model. Nevertheless the choice must be supported by knowledge of the system. Furthermore, since one of the major advantages of PCA is its ability to deal with highly correlated data sets, there is no need to reduce the number of original variables used to build the model even if some of them are redundant.

2.8.5 Data Filtering

Generally data collected on industrial process are corrupted with noise, which might have originated from different sources. In fact, noise can be associated with measurement sensors, electrical equipment used to transmit the measurement to the data gathering system, or with the

process itself. Different techniques have been proposed to reduce noise level in measurements collected on industrial processes. In particular electrically generated noise can be minimised by following established procedures concerning shielding of sensors and cables, and electrical filtering. On the other hand, process and measurement noise can be reduced by signal filtering. This can be achieved by means of analogue or digital filters implemented as hardware or software devices, respectively. An overview of analogue and digital filters and related references can be found in Seborg *et al.* (1989).

When performing empirical modelling, filtering noise corrupted data is a critical issue. In particular one of the major problems is the estimation of how much of the data used for building the model is "*signal*" and how much is "*noise*", because interest is generally focused on identifying and modelling the signal rather than the noise. This implies defining a criterion to discriminate between signal and noise, and hence identifying a suitable measure of the "*goodness*" of the model. Both the discriminating criterion and measure of "*goodness*" depends on the objective of the model (e.g. approximation or regression) and on the definition of signal and noise. According to Wold (1978), the signal corresponds to that part of the data which displays a systematic trend, while the noise comprises the complementary fraction (residuals) of the data which exhibits random or pseudo-random behaviour. In fact the noise could also show a systematic trend (e.g. oscillatory behaviour), but is considered to be negligible with respect to the main signal. In practice it is not straight forward to define a physical (quantitative) limit between signal and noise and cannot be done without specific knowledge of the system. More precisely defining noise and signal involves knowing the accuracy of the sensor, the sensitivity of the process to changes in the measured variable, the desired accuracy of the model and the aim of the model. Furthermore this limit can vary from process to process and between similar sensors which may be located at different points on the piece of equipment or on the same process.

Therefore it is not possible to define a general statistical or mathematical rule to define what is noise and what is signal. However it is always useful to know the noise level introduced by the sensors and by the signal transmitters. This is generally expressed in terms of the ratio between the variance of the measured value σ_x and the variance of the noise introduced by the sensor σ_e and is usually referred to as the signal to noise ratio (s.n.r):

$$\text{s.n.r.} = \frac{\sigma_x}{\sigma_e}. \qquad \textbf{2-43}$$

As stated before, a number of traditional techniques are available for noise reduction. Nevertheless PCA can handle noisy data sets and, furthermore, can be used to separate underlying data structures from noise (i.e. filtering the data). This is achieved by identifying the number of components which extract the underlying signal in the data set while rejecting the noise and can be attained by means of cross validation (§ 2.6). In particular random variations (e.g. noise) are usually captured by the last components, whilst the first few components provide a generalised representation of the data matrix. Therefore, by applying cross validation to develop a PC model it can be assumed that the scores representation of the data matrix is noise free, and consequently the number of components identified by cross validation can be used to approximate the original data matrix leaving out the noise.

2.9 Conclusions

When dealing with multivariate system data analysis, one of the major problems is handling the information contained in the data collected on the system. In fact, with modern data acquisition devices, it is practically possible to collect and record measurements on almost every variable describing the behaviour of a process. As a consequence, these data sets represent a "*gold mine*" because of the information they contain about the process of interest. However, extracting the relevant information is not straightforward. In certain situations, identifying trends or clusters of process measurements corresponding to atypical working conditions of the process, and addressing the causes for these abnormalities can be unfeasible because of the excessive amount of data collected, which cannot be properly analysed by visual inspection or with traditional statistical tools (e.g. traditional statistical univariate control charts).

Principal Component Analysis has been shown to be a useful tool for the analysis of multivariate systems. Generally, it can provide a compact representation of a large data set through the score and loading vectors associated with the first few principal components. These can be used for process analysis, because they contain the relevant information about the process and the working conditions corresponding to the data. Score and loading vectors provide also a compact

graphical representation of the multivariate data system and can be used to identify any typical (or atypical) features of the process (i.e. trends, clusters or variables collinearities) and hence can be used for monitoring (either off-line or on-line) process performance.

Furthermore, if quality measurements are available, principal component analysis can be used for monitoring the process, and comparing different working conditions in terms of product quality. This can be extended to on-line application, if quality data are available on-line.

It is known that in many practical situations, quality measurements are not available on a regular time basis. Typically they are available off-line. As a consequence, applications of principal component analysis for process analysis and monitoring relies on the assumption that process measurements can be used not only to describe the behaviour of the system of interest but also to infer the quality of the product. In this case the working conditions of the process are considered, to a certain extent, to be representative of the quality of the product.

However if the interest is focused upon the product quality, empirical regression models of the system are generally used for prediction (inference). Although principal component analysis can provide information about linear dependencies characterising the system, it cannot be used as a regression tool. In this case, multivariate empirical regression techniques, such as Multiple Linear Regression (MLR) or Projection to Latent Structures (PLS), can be used to develop regression models of the process. These are subject of the next Chapter. Nevertheless, principal component analysis can be used as an investigative tool, in particular for pre-treating the data before carrying out any regression.

3. Projection to Latent Structures

3.1 Introduction

Projection to Latent Structures or Partial Least Squares (PLS) has been shown to be a reliable multivariate linear regression technique for the analysis and modelling of noisy and highly correlated data. In particular it represents an appealing alternative to the classical Multiple Linear Regression (MLR) approach since it has been shown to be more robust than the least squares methodology. In this context, *robust* means that PLS can handle variable noise, variable correlations and high data dimensionality, which are the main drawbacks of MLR. Because of this, PLS has established itself as an important analytic and regressive tool in chemometrics, whilst the chemical and manufacturing processes are gaining benefits from its use, particularly with applications in process analysis, modelling, monitoring and control.

The development of PLS is mainly due to the pioneering work of H. Wold who applied the technique for modelling socioeconometric trends during the late sixties. While the first applications of PLS in the field of physical sciences, more precisely for the development of multivariate calibrations in chemometrics, are due to B. Kowalsky, S. Wold and H. Martens during the late seventies. During the last decade PLS has been successfully used for the modelling, prediction and statistical monitoring and control of the behaviour of a wide variety of chemical process. For example Kresta *et al.* (1991) illustrated the use of PLS for process monitoring, underlying the advantage of using PLS as a graphical investigative tool. In particular they used PLS for monitoring mechanistic simulation models of a fluidized bed reactor and of an extractive distillation column. Similarly MacGregor *et al.* (1991a,b) and Skagerberg *et al.* (1992) applied PLS for the monitoring of a mechanistic model of a Low Density Polyethylene Reactor. Mejdell *et al.* (1991) used PLS for describing the behaviour of a mechanistic simulation model of a high purity distillation column. Meanwhile, MacGregor (1994) and Kourti *et al.* (1994) reviewed a number of multivariate statistical process control charts which can be used in conjunction with PLS for process monitoring.

An early feature of PLS was that it was originally developed as an iterative procedure, the Non-linear Iterative Partial Least Squares (NIPALS) algorithm, for linear regression without a specified statistical or numerical basis, and consequently it was initially treated as an algorithm as opposed to a sound statistical methodology. It is mainly due to this reason that its acceptance has been slow in the field of statistics (De Jong, 1991).

During the eighties, a number of papers were published concerning the mathematical foundations of PLS (Lorber *et al.*, 1987; Manne, 1987; Höskuldsson, 1988) and its relationship to other statistical techniques, such as Multiple Linear Regression (MLR), Ridge Regression (RR) and Principal Component Regression (PCR), (Boardman *et al.*, 1981; Næs *et al.*, 1985; De Jong, 1991; De Jong, 1993b). A review of the development of PLS is given by Geladi (1988), while the work of Geladi and Kowalski (1986a,b) is commonly accepted as one of the most clear and complete tutorial on PLS. However, the paper of Höskuldsson (1988) represents a major breakthrough in PLS modelling and following on from this work, a number of PLS algorithms have been published during the nineties (Kaspar *et al.*, 1993; Lindgren *et al.*, 1993; De Jong, 1993a; De Jong *et al.*,1994). These are a consequence of the fact that PLS generally identifies a projection based approach to empirical modelling and, consequently, a class of corresponding algorithms.

3.2 Linear Regression: From MLR to PLS via PCR

In many practical situations, and generally in empirical modelling, the objective of a multivariate statistical regression analysis is not to explain the variability inherent within the process measurements as in Principal Component Analysis, but to use historical data collected on the working conditions of the process and the quality of the product to predict or infer the values of unavailable quality variables corresponding to new process observations. In fact, the quality measurements usually are not available on a regular and frequent basis or require expensive and time demanding laboratory analysis leading to unacceptable time delays.

The underlying assumption in empirical modelling is that the training data used to build the model contains relevant information to predict the future behaviour of the process. This is true

as far as the process remains within the working conditions corresponding to these data (i.e. same production, same strategy, same set-points for the control loops).

Statistical regression techniques, such as MLR which is based upon least squares methodology, have been extensively used for developing linear empirical models for the prediction (interpolation) of unavailable quality measurements corresponding to process measurements on the basis of historical data. However it is well known that when dealing with noisy and highly correlated multivariate problems, the traditional least squares approach can lead to singular solutions or imprecise parameter estimation, and hence to poor predictions (Wetherill, 1986). To overcome this limitation a common approach is to eliminate some of the process variables, according to heuristic rules or process knowledge. However while removing the numerical ill-conditioning, this approach can lead to unsatisfactory estimates. In fact, removing process variables implies subtracting information which might be useful in terms of prediction. Alternatively Ridge Regression and related Regularisation Methods (Wetherill, 1986) tackle the problem of collinear data by attempting to stabilise the regression coefficients by means of penalty factors. Unfortunately these approaches are cumbersome and computationally intense (because of the choice of the penalty factor) and do not reduce the dimensionality of the problem, which is of importance when the regression model is to be used as an investigative tool for process monitoring.

Limitations due to measurement noise, correlated variables, unknown variable and noise distribution, and data set dimensionality can be overcome by applying multivariate statistical projection based regression techniques such as Principal Component Regression (PCR) and Projection to Latent Structure (PLS). In fact, it is known that projection based techniques are capable of handling highly correlated, noise corrupted data sets since they are based on the assumption of dependency (correlation) between the variables and, consequently, provide the capability to estimate the main underlying structures in terms of a reduced number of linear combination of the original variables. Furthermore, these linear combinations usually represent the true dimensionality of the system of interest (Mejdell 1991).

In particular Principal Component Analysis (PCA) is known to be suited for handling large noisy and highly correlated data sets, and Principal Component Regression (PCR) exploits this capability of PCA by regressing each of the quality measurements on the principal component scores of the input data matrix, which are themselves independent. In this way PCR overcomes

the ill-conditioning problem due to variable collinearities. Furthermore, because of the low-dimensional representation of the predictor variables (i.e. the process measurements) achieved by using only the higher order principal components, PCR reduces the impact of measurement noise on the regression parameters (§ 2.10.5).

However, if there exists more than one output variable, PCR treats them as individual entities, while, in practice, they might constitute a multidimensional system which can provide additional relevant information for the development of the regression model. Furthermore, the space defined by the principal components of the input data matrix is not necessarily the most predictive with respect to the output variables because, as mentioned before (§ 2) in PCA the principal components capture the directions (linear combinations) of greater variability in the data matrix. Therefore, using the principal components of the input data matrix to regress the output variables does not take account of either the actual structure of the output data matrix, or the functional relationships which exists between the process and quality measurements.

These limitations of PCR can be partially overcome by regressing the principal component of the output data matrix on the principal components of the input data matrix. This approach would lead to a more stable MLR solution since the PCA representation of the input and output data automatically removes any variable collinearity and noise which can affect both the input and output data matrices. However it would still treat the two sets of variables independently before performing the least squares regression and, as stated before, it cannot be assumed that the principal components of the input data matrix are the best in terms of predictive capabilities of the structures of the output data matrix, in this case the principal components of the output data matrix.

PLS simultaneously addresses the above problems. Conceptually it is similar to PCA and PCR, since it is a projection based regression technique which provides a low dimensional representation of both the input and the output data matrices. This is achieved by means of particular orthogonal linear combinations (generally known as "*latent variables*") of the original input and output variables. As a consequence PLS compromises between the *approximation* of the predictor (process) variables and the *prediction* of the response (quality) variables.

In fact, similar to the principal components in PCA, the latent variables in PLS define the orthogonal directions of "*great*" variability in each of the two matrices. However they do not

identify the directions of "*greatest*" variability of the two data matrices (as the principal components do). PLS seeks to identify those linear combinations of the input variables which capture most of the variation of the input data set while being more predictive of corresponding linear combinations of the output variables, which capture most of the variation of the output data set. Consequently, the corresponding latent variables can be used to provide a set of orthogonal (i.e. independent) linear regression models between the predictor and the response variables. More precisely, PLS simultaneously reduces the dimensionality of the input and the output data matrices by seeking those latent variables of the predictor variables space and of the response variable space which have maximum variability and which are themselves highly correlated. With respect to this issue, Höskuldsson (1988) observed that in PLS the latent variables are selected to give the *maximal* reduction in the dimensionality of the regression problem which reflects the capabilities of PLS to identify the minimum number of latent variables required to describe the covariance structure existing between the input and the output data matrices.

The advantage of applying PLS, in preference to MLR as a linear regression technique, arises from the fact that in PLS the input latent variables are linearly independent, i.e. they define a set of orthogonal reference axis and each output latent variable is defined to be orthogonal to the previously extracted input latent variables. Consequently each ordinary least squares regression performed between each pair of input and output scores vectors cannot affect the others and cannot be affected by the others. This gives the PLS algorithm the robustness that MLR lacks in terms of sensitivity to variable collinearities. Furthermore when handling linearly correlated or noisy data, the "*validated*" PLS regression model uses fewer latent variables than original observations. This is due to the fact that the first few latent variables typically summarise the major sources of variability of the data set. As a consequence by using only these latent variables to build the regression model the problem of rank deficiency of the input data due to variable collinearities and measurement noise is overcome for both the input and output data matrices. This is also a consequence of the fact that linear correlations and measurement noise is usually associated with the lower order components, which are generally excluded from the final PLS model since they explain only a small proportion of the total variation.

Höskuldsson (1988) addressed PLS also from a geometrical perspective and showed that the objective of the PLS methodology is to find orthogonal rotations of the original matrices such that the angle between the rotated matrices is minimised. Similarly De Jong (1991) demonstrated

how in MLR the direction of the regression model is selected regardless of the variance-covariance structure of the X matrix. Whilst in PCR it is chosen without consideration of the correlation between X and Y. On the other hand, in PLS the direction of the regression model lies between the orientations identified by MLR and PCR, hence maximising the correlation between the input latent variables and the output latent variables.

It should be pointed out that in PLS (like in PCR) no assumption is made concerning the dimension of the data matrices. While for MLR the number of samples is required to be greater than the number of variables, in order to ensure that the inner product of the input data matrix ($X^T \cdot X$) is invertible.

3.3 Linear PLS and the NIPALS algorithm

Given a reference data set of N time consistent ("*time aligned*") samples collected on M regressor variables, x_{nm} (the independent variables) and K response variables, y_{nk} (the dependent variables), these can be arranged into an ($N \times M$) matrix, $\boldsymbol{X}$, and an ($N \times K$) matrix, $\boldsymbol{Y}$, respectively. Typically, in empirical modelling, the X matrix relates to the process variables (inputs), whilst the Y matrix corresponds to the quality or reference variables (outputs). This may comprise also a single quality variable. Standard linear PLS projects the X and Y matrices down onto a subset of latent variables, t and u, which are referred to as the input and output *scores*, respectively. The objective of the procedure is to fit a linear relationship between the independent and the dependent variables by performing an ordinary least squares regression between each pair of corresponding t and u scores vectors:

$$u = t \cdot b + e \qquad \textbf{3-1}$$

where b is the coefficient of the inner linear regression between the input and the output latent variables t and u, and is found by least squares regression:

$$b = \frac{t^T \cdot u}{t^T \cdot t}. \qquad \textbf{3-2}$$

The "*engine*" of the PLS procedure is the Non-linear Iterative Partial Least Squares (NIPALS) algorithm (Wold, 1966a,b; Wold, 1973). The NIPALS algorithm sequentially extracts each pair of corresponding latent variables as linear combinations of the input and output variables (Equations 3-3 and 3-4), prior to fitting the inner linear regression (Equation 3-2), and then evaluating the linear prediction of the output scores (Equation 3-5):

$$t = \sum_{m=1}^{M} x_m \cdot w_m = X \cdot w \qquad \textbf{3-3}$$

$$u = \sum_{k=1}^{K} y_k \cdot c_k = Y \cdot c \qquad \textbf{3-4}$$

$$\hat{u} = t \cdot b\,. \qquad \textbf{3-5}$$

In Equations 3-3 and 3-4, x_m and y_k are the column vectors comprising the observations collected on the m-th input variable and the k-th output variable, respectively, and w_m and c_k are the corresponding coefficients (*weights*) of the linear combinations. w and c denote the column vectors comprising the corresponding weights. Input and output loadings vectors are then computed from the least squares regression of the X matrix on t and of the Y matrix on $\hat{u}$:

$$p^T = \frac{t^T \cdot X}{t^T \cdot t} \qquad \textbf{3-6}$$

$$q^T = \frac{\hat{u}^T \cdot Y}{\hat{u}^T \cdot \hat{u}} \qquad \textbf{3-7}$$

in order to fit the approximation of the input data matrix given by the input scores:

$$\hat{X} = t \cdot p^T \qquad \textbf{3-8}$$

and the prediction of the output data matrix given by the predicted output scores:

$$\hat{Y} = \hat{u} \cdot q^T . \qquad \textbf{3-9}$$

The procedure starts with an initial guess of the output score vector u (typically this is initiated to the variable with the largest variance in the output data matrix), the column vector comprising the coefficients of the linear combination given in Equation 3-3 is then calculated by performing a least squares regression of X on u:

$$w^T = \frac{u^T \cdot X}{u^T \cdot u} \qquad \textbf{3-10}$$

and is normalised to unit norm. Hence the input score vector t is computed:

$$t = \frac{X \cdot w}{w^T \cdot w}. \qquad \textbf{3-11}$$

Similarly the column vector comprising the coefficients of the linear combination given in Equation 3-4 is calculated by performing a least squares regression of Y on t:

$$c^T = \frac{t^T \cdot Y}{t^T \cdot t} \qquad \textbf{3-12}$$

and is normalised to unit norm. A new output score vector u is then computed:

$$u = \frac{Y \cdot c}{c^T \cdot c} \qquad \textbf{3-13}$$

and the overall procedure is reiterated, starting from the regression of the input weights w (Equation 3-10), until convergence on u is achieved. Afterwards, the regression coefficient between t and u is evaluated (Equation 3-2) and the input and output loading vectors p and q are computed (Equations 3-6 and 3-7). The final step in the iterative procedure is to deflate the X and Y matrices:

$$E = X - t \cdot p^T \qquad \textbf{3-14}$$

$$\boldsymbol{F} = \boldsymbol{Y} - \hat{\boldsymbol{u}} \cdot \boldsymbol{q}^T . \qquad \textbf{3-15}$$

If further components are required the data matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are replaced by the corresponding residual matrices $\boldsymbol{E}$ and $\boldsymbol{F}$ and the procedure is repeated.

This last feature represents another characteristic of the NIPALS algorithm (and similarly of the NILES algorithms for PCA), that is to "*peel*" the input and output data matrices by sequentially extracting relevant information from the corresponding residual matrices, after fitting the approximation of $\boldsymbol{X}$ and the prediction of $\boldsymbol{Y}$, respectively. In this way each *latent structure* (i.e. input/output scores) accounts for information which has not been modelled by the previous latent variables. Furthermore, as observed by Geladi *et al.* (1986), Equations 3-10 and 3-12 provide an exchange of information between the two data sets ("*scores exchange*"). This reflects the fact that PLS seeks to compromise between the approximation of $\boldsymbol{X}$ and the prediction of $\boldsymbol{Y}$, but stressing the prediction of the output variables more than the approximation of the input variables.

By regressing the output loading vector $\boldsymbol{q}$ on the (deflated) matrix $\boldsymbol{X}$ and on the predicted output scores $\hat{\boldsymbol{u}}$ (Equation 3-7) it can be shown (§ 3.4) that this is equivalent to the normalised output weight vector $\boldsymbol{c}$. Therefore the calculation of $\boldsymbol{q}$ can be omitted by setting it equal to $\boldsymbol{c}$, and $\boldsymbol{q}$ can then be used to denote both the output weight and loading vectors. In particular, for consistency with other descriptions of the NIPALS algorithm the loading/weight vector $\boldsymbol{q}$ is computed directly from equation 3-12, and the calculation of the weight vector $\boldsymbol{c}$ is omitted. The final algorithm is summarised in Table 3.1.

The score, weight and loading vectors are generally arranged in matrices denoted by $\boldsymbol{T}$ and $\boldsymbol{U}$, $\boldsymbol{W}$ and $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. The inner regression coefficients are similarly grouped into a diagonal matrix $\boldsymbol{B}$, with the off-diagonal elements set equal to zero. The order in which the vectors and regression coefficients are sorted in the corresponding matrix is determined by the order in which they have been extracted by the NIPALS algorithm. The final PLS regression model is then represented by the input weight matrix $\boldsymbol{W}$, the input and output loading matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, and the regression diagonal matrix $\boldsymbol{B}$.

As for PCA, variable scaling (§ 2.2) is a critical issue in PLS and for consistency with the common approach adopted in the literature, both the input and output data matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are standardised to zero mean and unit variance for the course of the description of PLS, unless stated otherwise.

3.4 Some Modifications of the NIPALS Algorithm

In § 3.3, it was stated that regressing the output loading vector $\boldsymbol{q}$ on the (deflated) $\boldsymbol{X}$ matrix and on the predicted output scores $\hat{\boldsymbol{u}}$ (Equation 3-7) gives the equivalent result as for the output weight vector $\boldsymbol{c}$. Furthermore, it can be shown that the inner regression coefficient b is equal to the norm of the output weight vector $\boldsymbol{c}$ prior to scaling it to unit norm, and that those two modifications are related to each other. In fact, considering Equation 3-5, the regression of the output loadings (Equation 3-7) can be rewritten as:

$$\boldsymbol{q}^T = \frac{\hat{\boldsymbol{u}}^T \cdot \boldsymbol{Y}}{\hat{\boldsymbol{u}}^T \cdot \hat{\boldsymbol{u}}} = \frac{b \cdot \boldsymbol{t}^T \cdot \boldsymbol{Y}}{b^2 \cdot \boldsymbol{t}^T \cdot \boldsymbol{t}} = \frac{1}{b} \cdot \frac{\boldsymbol{t}^T \cdot \boldsymbol{Y}}{\boldsymbol{t}^T \cdot \boldsymbol{t}} = \frac{1}{b} \cdot \tilde{\boldsymbol{c}}^T \qquad \textbf{3-16}$$

where $\tilde{\boldsymbol{c}}$ denotes the output weight vector $\boldsymbol{c}$ prior to scaling it to unit norm:

$$\tilde{\boldsymbol{c}}^T = \frac{\boldsymbol{t}^T \cdot \boldsymbol{Y}}{\boldsymbol{t}^T \cdot \boldsymbol{t}}. \qquad \textbf{3-17}$$

Thus writing the normalised output weight vector $\boldsymbol{c}$ as:

$$\boldsymbol{c} = \frac{\tilde{\boldsymbol{c}}}{\|\tilde{\boldsymbol{c}}\|} \qquad \textbf{3-18}$$

and considering Equation 3-13, the regression coefficient b given by Equation 3-2 can be rewritten as:

$$b = \frac{\boldsymbol{t}^T}{\boldsymbol{t}^T \cdot \boldsymbol{t}} \cdot \frac{\boldsymbol{Y} \cdot \boldsymbol{c}}{\boldsymbol{c}^T \cdot \boldsymbol{c}} = \frac{\boldsymbol{t}^T \cdot \boldsymbol{Y}}{\boldsymbol{t}^T \cdot \boldsymbol{t}} \cdot \frac{\boldsymbol{c}}{\boldsymbol{c}^T \cdot \boldsymbol{c}} = \frac{\tilde{\boldsymbol{c}}^T \cdot \boldsymbol{c}}{\boldsymbol{c}^T \cdot \boldsymbol{c}}. \qquad \textbf{3-19}$$

Since:

$$c^T \cdot c = 1 \qquad \textbf{3-20}$$

and from Equation 3-18 the expression for the regression coefficient can be further simplified to:

$$b = \frac{\tilde{c}^T \cdot \tilde{c}}{\|\tilde{c}\|} = \|\tilde{c}\| . \qquad \textbf{3-21}$$

Consequently the output loading vector can be rewritten as:

$$q^T = \frac{\tilde{c}^T}{\|\tilde{c}^T\|} = c^T . \qquad \textbf{3-22}$$

Hence the equalities mentioned before:

$$b = \|\tilde{c}\| \qquad \textbf{3-23}$$

and:

$$q = c . \qquad \textbf{3-24}$$

The NIPALS algorithm can be further simplified if the output weight vector $\boldsymbol{c}$ is not normalised to unit norm. In this case the regression coefficient b can be written as:

$$b = \frac{t^T \cdot u}{t^T \cdot t} = \frac{t^T}{t^T \cdot t} \cdot \frac{Y \cdot c}{c^T \cdot c} = \frac{t^T \cdot Y}{t^T \cdot t} \cdot \frac{c}{c^T \cdot c} \qquad \textbf{3-25}$$

which reduces to:

$$b = \frac{c^T \cdot c}{c^T \cdot c} = 1 \qquad \textbf{3-26}$$

and implies that:

$$\hat{u} = t . \qquad \textbf{3-27}$$

Consequently, if the output loading/weight vector is not normalised to unit norm, the NIPALS algorithm can be used to identify a single latent variable space, corresponding to the input latent variable t, which provides both the approximation and the prediction models for the input and output data matrices, respectively. Additionally it is not necessary to evaluate the inner regression coefficient b because it is equal to one. Furthermore, the PLS regression model is built in such a way that it is easier to interpret from a geometrical point of view, since both the input and the output data matrices are projected down onto the same latent variable (i.e. along the same direction), which is the "*best*" latent direction in terms of both approximation and prediction capabilities. This simplified version of the NIPALS algorithm is given in Table 3.2.

Another simplification which applies to the NIPALS algorithm is when the output data set comprises a single quality variable. In this case, the iterative algorithm reduces to a single iteration procedure for each component and the output loading/weight coefficients for all the components are equal to one. It is generally known as PLS1 algorithm and details are given in Table 3.3. This algorithm follows naturally from the fact that there is only one variable contributing to the linear combination corresponding to the output latent variable. Because of this, the output score vector is the same as the output variable, and no iterations are required. Consequently the algorithm seeks the linear combination of the input variables which provides the best correlation with the output variable, or its deflated values after the first latent variables have been computed.

3.5 PLS Regression Model

As stated previously PLS provides both an approximation model for the input data matrix $\boldsymbol{X}$ and a prediction model for the output data matrix $\boldsymbol{Y}$. The approximation given by the first A latent variables can be written as:

$$\hat{X}_A = \sum_{j=1}^{A} t_j \cdot p_j^T \qquad \textbf{3-28}$$

whilst the prediction of the output data matrix given by the same A latent variables can be defined as:

$$\hat{Y}_A = \sum_{j=1}^{A} \hat{u}_j \cdot q_j^T \qquad \textbf{3-29}$$

where $\hat{u}_j$ denotes the prediction of the scores u_j given by:

$$\hat{u}_j = t_j \cdot b_j . \qquad \textbf{3-30}$$

Thus the prediction of the output data matrix can be rewritten as a function of the input scores:

$$\hat{Y}_A = \sum_{j=1}^{A} t_j \cdot b_j \cdot q_j^T . \qquad \textbf{3-31}$$

Furthermore both the approximation model given by Equation 3-28 and the prediction model given by Equation 3-29 can be expressed in matrix notation as:

$$\hat{X}_A = T_A \cdot P_A^T \qquad \textbf{3-32}$$

$$\hat{Y}_A = \hat{U}_A \cdot Q_A^T \qquad \textbf{3-33}$$

where P_A and Q_A denote the input and output loading matrices comprising the first A loading vectors. Similarly, T_A and $\hat{U}_A$ denote the input and output score matrices made by the first A input and corresponding predicted output score vectors. The predicted output score matrix can also be regressed on the input score matrix:

$$\hat{U}_A = T_A \cdot B_A \qquad \textbf{3-34}$$

where B_A is the diagonal matrix comprising the first A regression coefficients b_j. Consequently Equation 3-31 can be rewritten in matrix notation as:

$$\hat{Y}_A = T_A \cdot B_A \cdot Q_A^T . \qquad \textbf{3-35}$$

However the major objective of PLS as a regression technique is to predict the values of unavailable quality variables corresponding to new process measurements. This is achieved in PLS by computing the new input scores, regressing the corresponding output scores and evaluating the prediction of the output variables. When using PLS as a regression tool, the new process measurements must be consistent with the data used to build the model. This implies that the new data must be scaled according to the scaling procedure applied to the training data, and therefore using the same scaling factors (e.g. mean values and variances).

As observed previously, the NIPALS algorithm extracts one pair of latent input/output variables at a time from the corresponding deflated $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices. Consequently, if any component is required to be calculated it is necessary to deflate the input and the output data matrices up to that component. This is the case when new process data are available and the PLS model is used to infer the value of corresponding quality variables.

However, the scores matrix $\boldsymbol{T}$ can also be related directly to the input data matrix $\boldsymbol{X}$. A projection matrix $\boldsymbol{R}$ can be defined as:

$$\boldsymbol{R} = \boldsymbol{W} \cdot \left(\boldsymbol{P}^{T} \cdot \boldsymbol{W}\right)^{-1} \qquad \textbf{3-36}$$

which maps the input data matrix $\boldsymbol{X}$ onto the input latent variable space (De Jong and Ter Braak, 1994). The input scores can then be computed directly from $\boldsymbol{X}$ as:

$$\boldsymbol{T} = \boldsymbol{X} \cdot \boldsymbol{R} \qquad \textbf{3-37}$$

and the prediction of the output data can be related directly to the input matrix:

$$\hat{\boldsymbol{Y}} = \boldsymbol{X} \cdot \boldsymbol{R} \cdot \boldsymbol{B} \cdot \boldsymbol{Q}^{T} = \boldsymbol{X} \cdot \hat{\boldsymbol{B}}_{PLS} \qquad \textbf{3-38}$$

where:

$$\hat{\boldsymbol{B}}_{PLS} = \boldsymbol{R} \cdot \boldsymbol{B} \cdot \boldsymbol{Q}^{T} \qquad \textbf{3-39}$$

is the ($M \times K$) matrix of the regression coefficients between $\boldsymbol{X}$ and $\boldsymbol{Y}$, i.e. the model matrix for the regression of $\boldsymbol{Y}$ on $\boldsymbol{X}$. In particular, when using the first A latent variables, the projection matrix $\boldsymbol{R}$ can be built using the first A input weight and loading vectors:

$$\boldsymbol{R}_A = \boldsymbol{W}_A \cdot \left(\boldsymbol{P}_A^T \cdot \boldsymbol{W}_A\right)^{-1} \qquad \textbf{3-40}$$

and consequently the model matrix for the regression of $\boldsymbol{Y}$ on $\boldsymbol{X}$ based on the use of the first A latent variables can be evaluated as:

$$\hat{\boldsymbol{B}}_{PLS,\,A} = \boldsymbol{R}_A \cdot \boldsymbol{B}_A \cdot \boldsymbol{Q}_A^T. \qquad \textbf{3-41}$$

Furthermore, it can been shown that if the input data matrix is not singular, by retaining all the latent variables, the coefficients of the linear PLS regression model converge towards the coefficients of the corresponding MLR model:

$$\hat{\boldsymbol{B}}_{PLS,A}\Big|_{A=M} = \hat{\boldsymbol{B}}_{MLR} = (\boldsymbol{X}^T \cdot \boldsymbol{X})^{-1} \cdot \boldsymbol{X}^T \cdot \boldsymbol{Y}\,. \qquad \textbf{3-42}$$

However, if the input data matrix comprises collinear variables and measurement noise, the MLR approach would result in a singular solution or imprecise parameter estimation, and hence poor predictions. In comparison, the PLS regression parameters are unaffected by variable collinearities, even though they are affected by process noise since all the latent variables have been used to build the model.

Thus PLS can be used for regression in one of two ways. If interest is only in the prediction of the quality variables, without any regard to the underlying latent structures, the regression matrix given in Equation 3-41 can be used, and it would not be necessary to compute the input scores. However, if the regression model is to be used for process analysis or monitoring, the input latent variables represent a useful tool (§ 2.10), and the regression model can be split into the various components, i.e. weight, loading and score matrices. The overall PLS procedure corresponds to two linear *outer* mappings between the input and the output variables and the corresponding scores, and a linear *inner* mapping between each pair of latent variables, as summarised hereafter

linear input outer mapping	$X \rightarrow T$	$T = X \cdot R$
linear inner mapping	$T \rightarrow \hat{U}$	$\hat{U} = T \cdot B$
linear output outer mapping	$\hat{U} \rightarrow \hat{Y}$	$\hat{Y} = \hat{U} \cdot Q^T = T \cdot B \cdot Q^T$

3.6 Some Properties of PLS Regression Models

PLS provides an orthogonal bilinear decomposition of the X and Y matrices in a similar manner to that of PCA with a single data matrix. More precisely, the decomposition can be defined as the product between each pair of input score vectors, t_j, and predicted output score vectors, $\hat{u}_j$, and the corresponding input and output loading vectors, p_j and q_j:

$$X = \sum_{j=1}^{A} t_j \cdot p_j^T + E = T_A \cdot P_A^T + E \qquad \textbf{3-43}$$

$$Y = \sum_{j=1}^{A} \hat{u}_j \cdot q_j^T + F = \hat{U}_A \cdot Q_A^T + F \qquad \textbf{3-44}$$

where E and F are the resulting residual matrices for the regressor and response matrices, X and Y respectively, when a model with A latent variables is used for the approximation of the X matrix and the prediction of the Y matrix:

$$E = X - \sum_{j=1}^{A} t_j \cdot p_j^T \qquad \textbf{3-45}$$

$$F = Y - \sum_{j=1}^{A} \hat{u}_j \cdot q_j^T \; . \qquad \textbf{3-46}$$

Furthermore in PLS, as in PCA, each pair of latent variables accounts for a certain amount of the variability in the input and output data. Typically most of the variance of the X and Y blocks can

be accounted for by the first A latent variables, where $A < (M, K)$. From this a compressed representation of the input/output relationship can be achieved by means of a reduced number of latent variables. In PLS, the output latent variables are sorted in descending order with respect to the explained variance of the output data set. However this does not necessary imply that the input latent variables are sorted in descending order, since the PLS algorithm does not focus on explaining the variance of the input data matrix, but on extracting the structures of the input data which are more predictive of the output data. Thus reflecting the fact that the overall optimising criterion is to maximise the amount of variance of the output data set which can be captured by a limited number of input latent structures.

A further advantage of omitting the lower order latent variables is that the random noise, which is generally intrinsic to the data, is filtered out. The lower order latent variables typically explain less of the underlying variability in the data and account either for the measurement noise in the variables, or the linear dependencies which may exist between the regressor variables but which are not strongly related to the underlying input-output relationship.

However, a major difference exists between PCA and PLS in terms of explained variance. In PCA it is possible to explain all the variability in the data matrix, likewise in PLS it is possible to capture all the variance in the input data set, but this is not always true for the output data set. This is because the process variables used to build the regression model may not contain all the information required to predict the quality variables or alternately because the functional relationship between the two sets of variables may be non-linear and consequently, since the PLS methodology is inherently linear, it is less capable of fitting a non-linear relationship.

From an algebraic and geometric perspective, a number of properties hold for the structural input vectors of PLS, i.e. weight, loading and score vectors. A detailed description and proof of these properties is given by Höskuldsson (1988), De Jong (1993a) and Phatak *et al.* (1997).

The most important feature in terms of model robustness and prediction reliability of the PLS regression approach is the orthogonality of the latent variables, since it provides the insensitivity of the PLS regression model to variable collinearities and measurement noise. In particular the input score vectors are mutually orthogonal:

$$t_j^T \cdot t_i = 0 \quad (j \neq i) \qquad \textbf{3-47}$$

while the output score vectors are orthogonal to all the previously extracted input score vectors:

$$u_j^T \cdot t_i = 0 \quad (j > i). \qquad \textbf{3-48}$$

These two properties reflect the fact that each set of input and output latent variables are fitted on the input and output residual matrices $\boldsymbol{E}$ and $\boldsymbol{F}$, Equations 3-45 and 3-46 respectively, and hence on a subspace which is orthogonal (because complementary) to the space spanned by the previous ones. In other words, each new component seeks to model (extract) the information which have not been previously modelled by the earlier latent variables. Similarly the input weight vectors are mutually orthogonal, reflecting the orthogonality of the input scores:

$$w_j^T \cdot w_i = 0 \quad (j \neq i). \qquad \textbf{3-49}$$

Whilst the input loading vectors are orthogonal to all previously extracted input weight vectors:

$$w_i^T \cdot p_j = 0 \quad (j > i). \qquad \textbf{3-50}$$

As observed by Höskuldsson (1988) no special orthogonality properties exist amongst the output weight, loading and score vectors.

Generally the input weight vectors are scaled to unit norm and consequently represent a set of orthonormal vectors, similar to the loading vectors for PCA. However the input weight matrix $\boldsymbol{W}$ (or the input loading matrix $\boldsymbol{P}$) in PLS is not the same as the loading matrix $\boldsymbol{P}$ in PCA. In PCA the loading matrix is used to provide both the projection of $\boldsymbol{X}$ onto $\boldsymbol{T}$ and vice versa. In contrast, in PLS, the weight matrix $\boldsymbol{W}$ is used to provide the projection of the *deflated* $\boldsymbol{X}$ matrices onto the latent variables space, while the loading matrix $\boldsymbol{P}$ is used to project the score matrix $\boldsymbol{T}$ onto $\boldsymbol{X}$. Furthermore, although the projection matrix $\boldsymbol{R}$ (Equation 3-36) can be used to project the input data matrix $\boldsymbol{X}$ directly onto $\boldsymbol{T}$, it cannot be compared with the loading matrix in PCA since it is not orthonormal and cannot provide the inverse mapping from $\boldsymbol{T}$ to $\boldsymbol{X}$. A detailed description of

the PLS regression model in terms of projection matrices, and comparisons with PCR and MLR can be found in Phatak *et al.* (1997).

The orthonormality of the input weight matrix W holds because the weight vectors are orthogonal with unit norm as a result of the NIPALS algorithm. However there are variants of the NIPALS algorithm where the score vectors are scaled to unit norm, implying that although the weight vectors are mutually orthogonal, they no longer constitute a set of orthonormal vectors.

The paper of Höskuldsson (1988) represents a major breakthrough in terms of PLS regression algorithms. He observed that it is difficult to identify numerical properties of PLS by looking at the NIPALS algorithm, but the situation becomes clearer when relating the weight and score vectors at a generic iteration step with the vectors at the previous iteration step. He showed that for each component, the NIPALS algorithm operates in a similar manner to that of the power method (§ 2.7.1) for determining the largest eigenvalue and corresponding eigenvector of a matrix, and hence how the weight and score vectors of the PLS representation can be attained from the singular value decomposition of particular product matrices. In particular he showed that:

$$\left[X^T \cdot Y \cdot Y^T \cdot X\right] \cdot w = \lambda_w \cdot w \qquad \textbf{3-51}$$

$$\left[Y^T \cdot X \cdot X^T \cdot Y\right] \cdot c = \lambda_c \cdot c \qquad \textbf{3-52}$$

$$\left[X \cdot X^T \cdot Y \cdot Y^T\right] \cdot t = \lambda_t \cdot t \qquad \textbf{3-53}$$

$$\left[Y \cdot Y^T \cdot X \cdot X^T\right] \cdot u = \lambda_u \cdot u \qquad \textbf{3-54}$$

where the product matrices (between brackets) in Equations 3-51 to 3-54 are generally known as *kernel* matrices, and w, c, t and u are the corresponding first eigenvectors, i.e. the eigenvectors associated with the largest eigenvalues. Consequently the overall PLS algorithm can be reduced to the calculation of the first eigenvector of the kernel matrices (§ 3.8.2).

3.7 Selecting the Number of Latent Variables

The number of latent variables, A, to be retained in the final PLS regression model can be selected according to different heuristic or statistical tools. If the objective of the PLS analysis is model-building, one possible criterion is to fix a threshold value for the norm of the $\boldsymbol{Y}$-block residual matrix $\boldsymbol{F}$, i.e. $\|\boldsymbol{F}\|$, and then select the number of components corresponding to the first value of $\|\boldsymbol{F}\|$ below that threshold. But it is known that this approach leaves a certain degree of subjectivity in the selection criterion, which can lead to overfitting of the training data.

The simplest criterion is to select the number of components corresponding to the maximum percentage of variance explained on the output block, which implies that all the latent variables are included in the final regression model. However such a selection would incorporate measurement noise, variable correlations and process information which is specific to the training data set, and not to the process in general. Furthermore, it would lead to a model which overfits the training data, without capturing the underlying structure of the process. Thus, if the aim of the analysis is to build a robust model for prediction, cross validation (Wold, 1978; § 2.9) should be used to determine the desired number of latent variables to retain in the model. In this context, robust implies insensitivity to variable collinearities and measurement noise.

When performing cross validation in PLS, the selection of the subsets of data used to build and validate the model is made in a *sample-wise* way (§ 2.9). Hence the training data set is split into a number of subsets, which might comprise blocks of observations or individual samples. Initially, for a model comprising one latent variable, the first subset of data is omitted and a PLS model is built on the remaining data. The prediction error sum of squares (PRESS) on the output variables for the omitted subset of data is computed and the omitted subset restored to the data matrix. The procedure is repeated until every individual subset has been left out once. The individual PRESS's are then summed to give the total PRESS. The procedure is repeated for $i = 2,\ldots, M$ latent variables and the corresponding total PRESS calculated. The optimal number of latent variables is selected to be that which minimises the total PRESS. When a PLS model is validated through cross validation, matrix $\boldsymbol{E}$ in Equation 3-45 contains the part of $\boldsymbol{X}$ which is not useful to describe $\boldsymbol{Y}$, while matrix $\boldsymbol{F}$ in Equation 3-46 contains the part of $\boldsymbol{F}$ which cannot be related to $\boldsymbol{X}$.

3.8 Alternative PLS Algorithms

PLS was originally developed as an iterative procedure for developing linear regression models. It identifies a projection based approach to empirical modelling and, consequently, a class of corresponding algorithms. The NIPALS algorithm can be modified, for example by scaling the score vectors instead of the weight vectors to unit norm, or removing the linear inner regression between input and output scores and using the input latent variables both for the approximation of the input data matrix and the prediction of the output variables. Even though these modifications do not affect the underlying structure of the procedure, they can lead to more versatile PLS regression algorithms. Alternative PLS algorithms have also been proposed, some of which give the same model as that obtained from the NIPALS algorithm even though they are based upon completely different approaches, e.g. the kernel algorithm (§ 3.8.2). However there are also a number of PLS algorithms which cannot be compared to the original NIPALS algorithm since they are based upon different optimising criterion, e.g. the SIMPLS algorithm (§ 3.8.1) or else result in more complex model structures, e.g. the MultiBlock PLS algorithm (§ 3.9).

3.8.1 The SIMPLS Algorithm

An alternative to the NIPALS algorithm is the SIMPLS algorithm proposed by De Jong (1993a). This is an iterative procedure which directly calculates the latent variables as a linear combination of the original variables, and hence avoids deflating the input and output data matrices. However the coefficients of the linear combination calculated from the SIMPLS algorithm are not the same as those obtained using the NIPALS algorithm and arranged in the projection matrix $\boldsymbol{R}$ defined before (Equation 3-36). Furthermore, different orthogonality properties hold for the weight, loading and score vectors for the two approaches. Details can be found in De Jong (1993a) and Phatak *et al.* (1997). In particular De Jong (1993a) claimed that the SIMPLS regression model is easier to interpret, because of the direct mapping between the input variables and the corresponding scores, and hence the fact that the data matrices are not deflated. However, in practice the algorithm appears to be more complex than the traditional NIPALS approach and seems to depart from the traditional NIPALS methodology and is not discussed further in the thesis.

3.8.2 The PLS Kernel Algorithm

Höskuldsson (1988) proposed a completely novel approach to PLS modelling based upon the assumption that for each *latent dimension*, the weight and the score vectors are the first eigenvector of the kernel matrices of the corresponding deflated input and output data matrices (§ 3.6, Equations 3-51 to 3-54). In particular he proposed the use of singular value decomposition (Appendix 1) to compute the input weight vectors, that is the first eigenvector of the corresponding kernel matrices. The input and output score and loading vectors can then be calculated according to Equations 3-11, 3-12, 3-13 and 3-6, thus retaining the numerical and geometrical properties of the NIPALS algorithm. Kaspar *et al.* (1993), Lindgren *et al.* (1993), De Jong *et al.* (1994), Rännar *et al.* (1995) all proposed different variants of the PLS kernel algorithm, whilst Goutis (1997), Dayal *et al.* (1997), Westad *et al.* (1996) exploited the kernel algorithm to provide computationally fast PLS algorithms.

As observed by Lindgren *et al.* (1993), iterative procedures often become inefficient when the data structure to be modelled is large, and the NIPALS algorithm is no exception. It requires the calculation of the input and output score vectors, even though they are not necessary for future predictions. In contrast, if the score vectors do not need to be computed, the kernel algorithms can be used to compute the loading and weight matrices and the regression coefficients which are necessary for future predictions. The different versions of the kernel algorithm were seen to be faster than the traditional NIPALS algorithm (in terms of floating points operations, *flops*) and to require less memory since the calculation of the score vectors can be avoided during the model building phase, and the deflation process is generally performed on the smaller kernel matrices corresponding to the input weight vectors, rather than on the data matrices. The PLS kernel algorithms therefore gained importance as fast and memory parsimonious PLS regression algorithms.

3.9 The MultiBlock PLS Algorithm (MBPLS)

PLS can be used to model multivariate relationships which exist between two sets of observation, i.e. process and quality measurements. However, chemical processes generally consist of interconnected units, each one comprising a number of variables to be monitored. Thus when building a global model by merging together all the units, there could be hundreds of

variables to include in a single model. The advantage of using PLS is that it can handle large numbers of variables, especially when they are linearly correlated. However if the process variables are collected on different units which do not exhibit functional dependencies between themselves, the PLS model can loose accuracy or sensitivity since the weight coefficients (which are scaled to unit norm) are distributed between all the variables. Furthermore global PLS models based upon all the variables are difficult to interpret, because of the excessive number of variables. In this situation MultiBlock PLS (MBPLS) is an alternative tool to build a global regression model based upon a number of sub-models interconnected between them.

The MBPLS approach represents an exception to the family of PLS regression algorithms. In general, PLS regression models provide a direct mapping between the input and the output variables by means of the input and output score vectors which are sequentially linked. Consequently if the process consists of a number of different units, which are interconnected and participate to the quality of different products, all the measurements collected on process and quality variables need to be identified and placed in one of two data matrices (X and Y). In MBPLS, the individual units can be kept distinct. A consequence of this is that different sets of input and output variables can exist within the same model, theoretically providing a multiple-input-multiple-output (MIMO) PLS regression model. In MBPLS the underlying model structure seeks to resemble the structure of the process to be modelled. This is achieved through a number of individual score vectors which are linked in series and/or in parallel through a few global score vectors (the *consensus scores*) in order to reflect the contribution of each set of process measurements from each individual unit to each set of quality variables (Figure 3.1).

Figure 3.1 depicts a situation where three different input data sets (***X1***, ***X2*** and ***X3***) are used to infer the values of two different output data sets (***Y1*** and ***Y2***). In this case it is assumed that all the input data contribute to the value of ***Y2***, while only ***X1*** and ***X2*** are related to ***Y1***. Thus, one individual input score matrices is defined for each input data matrix (***T1***, ***T2*** and ***T3*** respectively). Similarly one individual output score matrices is defined for each output data matrix (***U1*** and ***U2***). Therefore one consensus score matrix is defined for each input/output stream (***Tc1*** and ***Tc2***). In Figure 3.1 the input score matrices ***T1*** and ***T2*** contribute to ***Tc1***, the consensus matrix used to infer ***U1*** (and hence ***Y1***). Then ***Tc1*** and ***T3*** relate to ***Tc2***, which predicts ***U2*** (and hence ***Y2***). It can be noticed that for this configuration the input score matrices

T1 and ***T2*** do not act directly on ***Tc2*** and that their contribution to the prediction of ***Y2*** is carried through ***Tc1***.

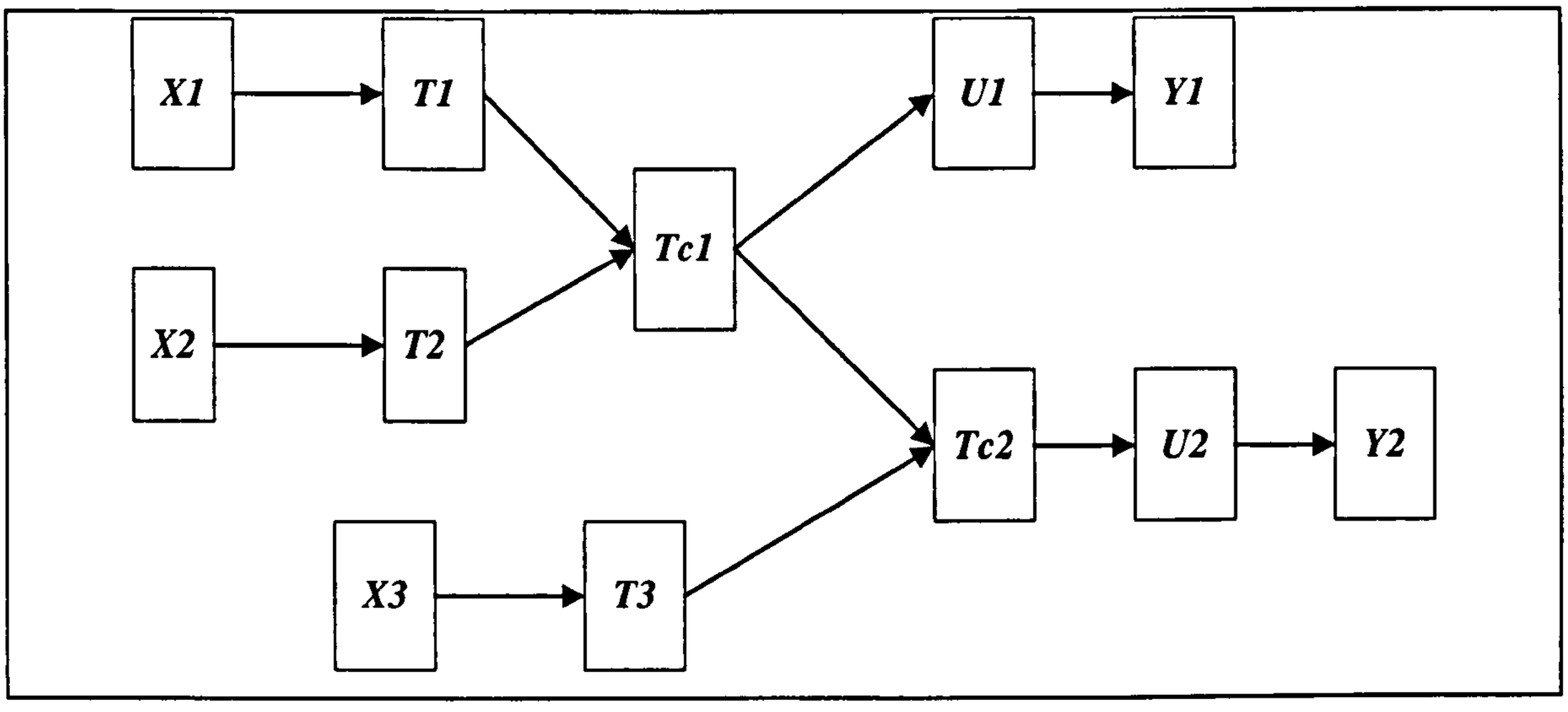

Figure 3.1: MBPLS flow diagram.

Thus a MBPLS model provides the potential to model and monitor the process as a single entity but with the possibility to screen each unit separately. In practice the MBPLS algorithm generates both individual score vectors for each input/output data set which are representative of the corresponding unit, and global score vectors which are representative of the whole process (or parts of it). In particular, individual score vectors are associated with each input unit and are used in a nested PLS algorithm to predict the scores associated with each output unit. Therefore it is possible to build control charts based upon the global model to provide a tool for the identification of non-conforming operation. But it is also possible to build individual control charts for each unit, which will be more sensitive than the equivalent control charts built upon the global MBPLS model or upon the equivalent PLS model for the same system. Therefore if an abnormal situation occurs on the whole process it can be identified from the global model, and a subsequent analysis can be carried out on the individual sections in order to identify which unit is responsible for the occurrence and hence identify the variables indicative of non-conform behaviour.

Therefore MBPLS can be used to build complex and structured PLS regression models, which, to a certain extent, could be considered as hybrid regression models, since it can be used to incorporate the structure of the process within the PLS regression model. However, because of this flexibility, a major limitation intrinsic to MBPLS modelling, and in particular, the building phase of the model is the fact that there is not a general MBPLS algorithm. It has to be built *ad hoc* for each specific process since it has to reflect the underlying structure of the process. Consequently for each process, a different algorithm has to be designed, which implies that specific knowledge of the process and of the mutual dependencies between the individual units is necessary.

An early description of the MBPLS methodology is given by Boardman *et al.* (1981). They referred to it as to an interdependent model with multiple indicators while a more generalised and detailed description of the procedure is given by Wangen *et al.* (1988). MacGregor *et al.* (1994) applied a simplified version of the algorithm for the modelling and monitoring of the behaviour of a two zone low density polyethylene reactor.

Wold *et al.* (1996) proposed a hierarchical multiblock PLS (H-PLS) algorithm which can be modified to provide a hierarchical multiblock PCA algorithm (H-PCA). In H-PLS, as in MBPLS, score vectors are defined for each unit and for the global model. However they do not exist at the same level as in MBPLS. In H-PLS, the score vectors are regressed simultaneously and are *hierarchically* arranged into a *super* and a *lower* level. In particular the low level scores are used to describe individual blocks, while the super level scores relate the input variables to the output variables. Furthermore, to preserve the direct generalisation of the ordinary PLS approach, the score vectors are linked in series, as in PLS, to provide a mapping between the input and the output data. In fact in H-PLS only one input and one output data matrix is used, and indices require to be specified to arrange the variables in different blocks. Consequently a general algorithm can be defined and is given in the work of Wold *et al.* (1996). However, because of this, the structure of the H-PLS algorithm lacks the flexibility of the ordinary MBPLS approach.

0	mean centre and scale X and Y
1	set the output scores u equal to a column of Y
2	regress columns of X on u $\quad w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	normalise w to unit length $\quad w = \frac{w}{\|w\|}$
4	calculate the input scores $\quad t = \frac{X \cdot w}{w^T \cdot w}$
5	regress columns of Y on t $\quad q^T = \frac{t^T \cdot Y}{t^T \cdot t}$
6	normalise q to unit length $\quad q = \frac{q}{\|q\|}$
7	calculate new output scores $\quad u = \frac{Y \cdot q}{q^T \cdot q}$
8	check convergence on u: if YES goto 9 ELSE goto 2
9	calculate the X loadings $\quad p^T = \frac{t^T \cdot X}{t^T \cdot t}$
10	regress u on t $\quad b = \frac{t^T \cdot u}{t^T \cdot t}$
11	calculate input residual matrix $\quad E = X - t \cdot p^T$
12	calculate output residual matrix $\quad F = Y - b \cdot t \cdot q^T$
13	if additional PLS dimension are necessary then replace X and Y by E and F and repeat steps 1 to 13.

Table 3.1: NIPALS Algorithm for PLS.

0	mean centre and scale X and Y
1	set the output scores u equal to a column of Y
2	regress columns of X on u $\quad w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	normalise w to unit length $\quad w = \frac{w}{\|w\|}$
4	calculate the input scores $\quad t = \frac{X \cdot w}{w^T \cdot w}$
5	regress columns of Y on t $\quad q^T = \frac{t^T \cdot Y}{t^T \cdot t}$
6	calculate new output scores $\quad u = \frac{Y \cdot q}{q^T \cdot q}$
7	check convergence on u: if YES goto 8 ELSE goto 2
8	calculate the X loadings $\quad p^T = \frac{t^T \cdot X}{t^T \cdot t}$
9	calculate input residual matrix $\quad E = X - t \cdot p^T$
10	calculate output residual matrix $\quad F = Y - t \cdot q^T$
11	if additional PLS dimension are necessary then replace X and Y by E and F and repeat steps 1 to 11.

Table 3.2: Simplified Version of the NIPALS Algorithm.

0	mean centre and scale $\boldsymbol{X}$ and $\boldsymbol{Y}$	
1	set the output scores $\boldsymbol{u}$ equal to a column of $\boldsymbol{Y}$	
2	regress columns of $\boldsymbol{X}$ on $\boldsymbol{u}$	$w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	normalise $\boldsymbol{w}$ to unit length	$w = \frac{w}{\|w\|}$
4	calculate the input scores	$t = \frac{X \cdot w}{w^T \cdot w}$
5	set $\boldsymbol{q}$ equal to one	$q = 1$
6	calculate the X loadings	$p^T = \frac{t^T \cdot X}{t^T \cdot t}$
7	regress $\boldsymbol{u}$ on $\boldsymbol{t}$	$b = \frac{t^T \cdot u}{t^T \cdot t}$
8	calculate input residual matrix	$E = X - t \cdot p^T$
9	calculate output residual matrix	$F = Y - b \cdot t \cdot q^T$
10	if additional PLS dimension are necessary then replace X and Y by $\boldsymbol{E}$ and $\boldsymbol{F}$ and repeat steps 1 to 10.	

Table 3.3: NIPALS Algorithm for PLS with one output variable (PLS1).

4. Non-Linear PLS Modelling

4.1 Introduction

When modelling data collected from a non-linear system, linear regression techniques may be used to approximate complicated relationships over small intervals of the predictor variables ***X***. This approach is based on the assumption that the underlying non-linear relationship can be locally approximated by a linear model, and hence that over these small intervals (domains of local linearity) the predictor variables ***X*** exhibit linear behaviour which can be approximated by linear regression models. However, in many practical situations, industrial data exhibit non-linear features which cannot be locally linearized. In this case, empirical modelling relies on either transforming the original variables using a non-linear function and applying linear regression to the transformed variables, or else adopting a more complex non-linear regression approach. Included within the framework of non-linear regression techniques are SPLINE functions (Wold, 1974) Sigmoid Neural Networks and Radial Basis Function Networks (Irwin *et al.*, 1995).

A common and simple approach for incorporating non-linear features in empirical modelling is to transform each variable by a suitable function, and then carry out a linear regression on the new set of variables. However, Friedman (1991) observed that by applying transformations to the individual variables, it is assumed that the variables are themselves independent, which is rarely true in practice, since measurements collected on a chemical processes are typically correlated. He stressed also that non-linear models are flexible and that transformations of the individual variables is only reliable when the number of observations is much larger than the number of variables. With respect to this issue Friedman (1991) suggested that the ratio between the number of samples to the number of variables should be at least 10 or 15.

When the number of variables is comparable or larger than the number of samples, projection based techniques, such as PCA and PLS, are appropriate for extracting information from the data. As observed previously, projection based regression techniques are founded on the assumption of correlation between the variables, and hence are particularly suited when the

variables exhibit strong dependencies. They provide the potential to summarise the underlying structures of the process as linear combinations (the latent variables) of the original variables and to relate the predictor and response variables by means of linear regression models between the latent variables. PLS has been shown to be a powerful and reliable approach for empirical modelling when dealing with noisy and highly correlated data and a limited number of observations. However linear PLS is inappropriate for modelling non-linear systems. A number of methods have been proposed to integrate non-linear features within the linear PLS framework, to produce a non-linear PLS algorithm. These include polynomial expansions and SPLINE functions, which are described hereafter, and multi-layer perceptrons which are reviewed in the next chapter.

4.2 Non-linear Projection to Latent Structures

When dealing with real complex chemical and physical systems, linear PLS cannot be used reliably to model the underlying structure of the process of interest since it may exhibit significant non-linear characteristics. In particular when applying linear PLS to non-linear problems, the minor latent variables cannot always be discarded since they may not only describe noise or negligible variance-covariance structures in the data, they may contain significant information about the non-linear nature of the problem. In fact non-linear structures may be modelled using a combination of latent variables which comprise both the higher-order and lower-order latent variables calculated from linear PLS. Although this limitation can be overcome by choosing an appropriate number of latent variables (for example by means of cross validation), the final PLS model might contain too many components to be fit for purpose, e.g. for monitoring and analysing the system of interest. Several attempts have been made aimed at producing non-linear variants of the linear PLS algorithm by integrating non-linear features within the linear PLS framework, and will be reviewed hereafter.

4.2.1 Linear PLS and Individual Non-Linear Transformations

One approach to developing a non-linear PLS regression model without modifying the NIPALS algorithm, is to extend the input matrix by including non-linear combinations of the original process variables such as logarithms, square values, cross-products, etc., and then performing a

linear PLS on the extended input and the original output matrices (Wold *et al.*, 1989; Berglund and Wold, 1997). This approach can be generalised by applying non-linear transformations to the input and the output variables (Mejdell *et al.*, 1991). Here, process knowledge can help in selecting the most appropriate and viable non-linear transformations. However, if there is no *a priori* knowledge about the underlying non-linear relationships that exists between some of the variables, then there is no limitation to the number (and kind) of transformation that might be applied. Thus by pretreating data sets in this way, the number of non-linear terms can increase excessively resulting in large input and output matrices; a consequence of this is that the results become difficult to interpret. Furthermore, the arguments raised by Friedman (1991) with respect to transformations of individual variables also apply in this case, and is even more pertinent since PLS is based on the assumption of dependencies between the variables, whilst individual transformations are not.

4.2.2 Modified NIPALS Algorithms

An alternative and more structured approach to the development of a non-linear PLS model is to modify the NIPALS algorithm by introducing a non-linear function which relates the output scores $\boldsymbol{u}$ to the input scores $\boldsymbol{t}$ (Equation 3.5) without modifying the input and output variables, i.e. retaining the input and output matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ in their original form. This procedure was originally investigated and described in detail by Wold *et al.* (1989) and several algorithms have been subsequently published and are reviewed hereafter.

4.2.2.1 Polynomial (Quadratic) PLS

Wold *et al.* (1989) proposed a non-linear PLS algorithm based on the use of a second order polynomial (quadratic) regression (QPLS) to fit the functional relation between each pair of latent variables, but still retaining the framework of the linear PLS, including the orthogonality of the predictor latent variables. In their paper Wold *et al.* (1989) outlined the main drawbacks of merging non-linear regression techniques within the framework of the linear PLS algorithm and proposed a way of updating the projection parameters of the input variables (weights $\boldsymbol{w}$), to match the mapping of the input variables onto the corresponding scores with the non-linear regression used to predict the output scores. More details about the QPLS approach and the input weights updating procedure are given in section 4.3. Wold *et al.* (1989) showed how their

approach of updating the input weights is suitable for any kind of continuous and differentiable functional relationship between the input and the output scores. They applied the modified algorithm to the analysis of two chemometric data sets, cosmetic data and a beta-receptor agonist example showing the improvement with respect to the linear PLS approach.

4.2.2.2 Linear Quadratic PLS

Wold *et al.* (1989) proposed also a Linear Quadratic PLS algorithm (LQPLS), described as a "*quick and dirty method*", that applies when the curvature in the inner relation between the input and the output scores is not too large. In this case, the weights produced by the linear PLS algorithm will be very similar to the weights produced by the QPLS algorithm and not updating the input weights will have a negligible effect on the final regression model. The LQPLS consists of the standard linear PLS algorithm (NIPALS), to produce the outer relations, followed by the fitting of the quadratic inner relation between the output and the input scores. Although it can be claimed that a quadratic function may not be the best non-linear relation to regress the output scores on the input ones, the problem can be overcome by choosing the non-linear function according to the scatter plot of the input-output scores arising from the linear PLS.

4.2.2.3 Smoothing PLS

To generalise the form of the non-linear inner mapping, Frank (1990) proposed using a smoothing regression procedure (Cleveland, 1979) to fit the relation between $\boldsymbol{t}$ and $\boldsymbol{u}$, while retaining the outer mapping of the linear PLS approach, i.e. without updating the input weights, therefore the algorithm is suitable when the inner relation is slightly non-linear. The local regression function and the size of the neighbourhood of each point, i.e. the smoothness of the regression function, have to be defined in advance, which makes the non-linear PLS algorithm dependent on these two parameters. Furthermore, as Wold (1992) observed, this approach is comparable to a first degree (linear) SPLINE function which is non-smooth when applied to small data sets and hence it is only suitable for large data sets. The method was applied to the analysis of three different data sets from the field of chemistry (modelling of molecular structure-activity, product quality control from analytical chemistry; and atomic emission spectrometry of wine samples from food chemistry) and was shown to perform better than the linear PLS approach. No comparison was given with respect to the QPLS algorithm.

4.2.2.4 SPLINE PLS

Wold (1992) went on and presented a non-linear PLS procedure which uses a smooth bivariate SPLINE function (quadratic or cubic) to fit the inner relation between each pair of latent variables. He claimed that this approach has greater approximation power and flexibility than the QPLS and the smoothing PLS. In this case the input outer weights w are derived from an appropriately weighted correlation of the u scores with the input X matrix in order to give each input variable the same variance as the input scores and therefore to correlate the output scores, i.e. the output Y variables, with the curved transformation of each input variable provided by the SPLINE function. The overall procedure produces an updating of the input weights that improves the fitting of the input variables with the output scores bringing the effects of the SPLINE inner relation into the input outer mapping. Furthermore, Wold (1992) pointed out that the SPLINE-PLS approach includes the ordinary linear PLS approach, and those non-linear PLS approaches based on the use of polynomial inner relations as limiting cases, due to the fact that SPLINE functions have the approximating properties of any continuous function. In particular, he observed that quadratic SPLINES are most suitable for data with no inflexion points, whilst cubic SPLINES assure a general approximation power for continuous data. The only problem arising when using SPLINE functions is the over-fitting capability of higher degree polynomials. For this reason Wold (1992) suggested to use models of lower degree and more knots, to achieve better stability and lower prediction error.

Wold (1992) stated that the SPLINE-PLS is more consistent than the QPLS approach with the PLS principles stressed by Höskuldsson (1988). However the SPLINE-PLS algorithm appears to be more obscure than the QPLS approach and no applications were found after the paper of Wold (1992) where the algorithm was applied to the beta-receptor agonist data. Slightly better performance than the quadratic PLS approach proposed in Wold *et al.* (1989) were attained. Again the degree of the SPLINE function, the number of knots and their placement must be specified in advance.

4.2.2.5 Neural Network PLS Algorithms

All the above non-linear PLS algorithms are based upon the assumption that the relationship between the predictor and the response latent variables can be modelled by means of a polynomial non-linear mapping (or a combination of them in the case of SPLINE functions). Consequently a number of parameters such as degree of the polynomial and SPLINE's knots

placements are required to be defined in advance. To avoid this issue and to take advantage of the fact that a neural network with one hidden layer of sigmoidal units can approximate any continuous function with arbitrarily desired accuracy (Cybenko, 1989), a number of non-linear PLS algorithms which use neural networks to fit the non-linear inner mapping have been proposed in literature and are reviewed in the next Chapter.

4.3 Quadratic Projection to Latent Structures [QPLS]

Wold *et al.* (1989) proposed a non-linear (polynomial) PLS regression algorithm which retains the framework of the linear PLS algorithm, including the orthogonality of the predictor latent variables $\boldsymbol{t}$, but which for each dimension (j) modifies the linear inner relation between the predictor ($\boldsymbol{t}$) and the response ($\boldsymbol{u}$) latent variables to a non-linear relationship:

$$\boldsymbol{u}_j = f_j(\boldsymbol{t}_j) + \boldsymbol{e}_j \qquad \textbf{4-1}$$

where $\boldsymbol{e}_j$ is the mismatch between the output scores $\boldsymbol{u}_j$ and their approximation given by the function $f_j(\boldsymbol{t}_j)$, which is applied to each element t_{nj} of the input score (column) vector $\boldsymbol{t}_j$.

The authors stated that any non-linear function $\boldsymbol{u} = f(\boldsymbol{t}) = f(\boldsymbol{X}, \boldsymbol{w})$ that is continuous and differentiable with respect to the weights $\boldsymbol{w}$, can be used to fit the inner model. In particular, they proposed a quadratic polynomial relation for the inner mapping of the form:

$$\boldsymbol{u}_j = c_{0,j} + c_{1,j} \cdot \boldsymbol{t}_j + c_{2,j} \cdot \boldsymbol{t}_j^2 + \boldsymbol{e}_j \qquad \textbf{4-2}$$

here $c_{0,j}$, $c_{1,j}$ and $c_{2,j}$ denote the coefficients of the quadratic expansion for the j-th component and are found by fitting the polynomial expansion on the input and output score vectors $\boldsymbol{t}_j$ and $\boldsymbol{u}_j$, respectively. In Equation 4-2 $\boldsymbol{t}_j^2$ denotes the column vector which comprises the square values of the input scores, i.e. t_{nj}^2.

In a similar manner to the linear PLS framework, it is possible to describe the algorithm as two linear outer mappings between the input and the output variables and the corresponding scores, and a non-linear inner mapping between each pair of latent variables as summarised hereafter:

linear input outer mapping	$X \rightarrow T$	$T = X \cdot R$
non-linear inner mapping	$T \rightarrow \hat{U}$	$\hat{u} = f(t)$
linear output outer mapping	$\hat{U} \rightarrow \hat{Y}$	$\hat{Y} = \hat{U} \cdot Q^T$

Wold *et al.* (1989) observed that by adopting the use of a non-linear function to fit the inner mapping between the input and the output scores, the NIPALS algorithm can be accordingly modified. By recalling that the projection coefficients of the input block, the weights w, are derived from the covariance of the u scores and the input X matrix (Equation 3.10), using a non-linear function to relate each pair of latent variables, the calculations of the inner mapping as well as those of the outer mapping are affected. To take this into account, Wold proposed updating the weights of the input outer relationship (w) by means of Newton-Raphson linearization of the inner relation, i.e. a first order Taylor series expansion of the quadratic inner relationship, and then solving it with respect to the weights increments Δw. The input weights updating procedure proposed by Wold *et al.* (1989) for a generic iteration and for a generic latent variable can be summarised as follows. Given the non-linear mapping between t and u:

$$u = f(t) + e = f(X, w, c) + e \qquad \textbf{4-3}$$

where $f(\cdot)$ is a continuous function differentiable with respect to w and c, and c is the vector comprising the parameters of the function $f(\cdot)$. X denotes the input data matrix when referring the procedure to the first latent variable, or the deflated input data matrix when referring to subsequent latent variables. The above relationship can be approximated by means of Newton-Raphson linearization:

$$u = f_{00} + \left.\frac{\partial f}{\partial c}\right|_{00} \cdot \Delta c + \left.\frac{\partial f}{\partial w}\right|_{00} \cdot \Delta w \qquad \textbf{4-4}$$

where:

$$f_{00} = \hat{u} = f(t, c) \qquad \textbf{4-5}$$

$$\left.\frac{\partial f}{\partial c}\right|_{00} \cdot \Delta c = \sum_{i=0}^{NC} \left.\frac{\partial f}{\partial c_i}\right|_{00} \cdot \Delta c_i \qquad \textbf{4-6}$$

$$\left.\frac{\partial f}{\partial w}\right|_{00} \cdot \Delta w = \sum_{m=1}^{M} \left.\frac{\partial f}{\partial w_m}\right|_{00} \cdot \Delta w_m \,. \qquad \textbf{4-7}$$

NC is the degree of the polynomial expansion and *M* is the number of input variables. The partial derivatives in Equations 4-6 and 4-7 can be evaluated numerically at each point t_n of the score vector $\boldsymbol{t}$ corresponding to $f_{00} = \hat{u} = f(t, c)$ since $\boldsymbol{t}$ is known and the assumption is made that the function $f(\cdot)$ can be differentiated with respect to $\boldsymbol{w}$ and $\boldsymbol{c}$. Thus the only unknowns in the approximation are Δc_i and Δw_m. The corrections Δw_m can be evaluated as follows:

1) f_{00}, $\frac{\partial f}{\partial c}$ and $\frac{\partial f}{\partial w}$ are initially placed in a matrix $Z = \left[f_{00} \; \frac{\partial f}{\partial c} \; \frac{\partial f}{\partial w} \right]$;

2) a column vector $\boldsymbol{v}$ is then regressed on $\boldsymbol{Z}$ and $\boldsymbol{u}$:

$$v = \frac{Z^T \cdot u}{u^T \cdot u}, \qquad \textbf{4-8}$$

which corresponds to:

$$Z = u \cdot v^T, \qquad \textbf{4-9}$$

3) and normalised to unit norm:

$$\boldsymbol{v} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} \qquad \textbf{4-10}$$

4) another column vector, $\boldsymbol{s}$, is then evaluated according to:

$$\boldsymbol{s} = \boldsymbol{Z} \cdot \boldsymbol{v} \qquad \textbf{4-11}$$

5) the vector $\boldsymbol{u}$ is then regressed on $\boldsymbol{s}$:

$$b = \frac{\boldsymbol{s}^T \cdot \boldsymbol{u}}{\boldsymbol{s}^T \cdot \boldsymbol{s}}, \qquad \textbf{4-12}$$

which corresponds to:

$$\boldsymbol{u} = b \cdot \boldsymbol{s}, \qquad \textbf{4-13}$$

6) the values of $[\, b \cdot \boldsymbol{v} \,]$ corresponding to $\frac{\partial f}{\partial w_m}$ are then assigned to Δw_m.

7) finally, $\boldsymbol{w}$ is updated:

$$\boldsymbol{w} = \boldsymbol{w} + \Delta \boldsymbol{w} \qquad \textbf{4-14}$$

8) and normalised to unit norm:

$$\boldsymbol{w} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \qquad \textbf{4-15}$$

4.4 Some Considerations on the Input Weights Updating Procedure

Wold *et al.* (1989) stated that "*the present algorithm (quadratic PLS) is fairly complicated and converges slowly when the data lack structure. Hopefully this situation may be improved by better algorithms*". The weights updating procedure proposed by Wold *et al.* can be considered to be cumbersome in its approach, and a number of issues that are interesting to address are raised. In particular, three questions arise from the original quadratic PLS algorithm and the input weights updating procedure:

- why is the vector $\boldsymbol{v}$ in Equation 4-8 computed according to $\boldsymbol{Z} = \boldsymbol{u} \cdot \boldsymbol{v}^T$ instead of $\boldsymbol{u} = \boldsymbol{Z} \cdot \boldsymbol{v}$?
- why is the vector $\boldsymbol{v}$ scaled (step 6, § 4.3) by means of the scalar factor b related to $\boldsymbol{u} = b \cdot \boldsymbol{s}$?
- why are the derivatives of $f(\cdot)$ with respect to $\boldsymbol{c}$ included in $\boldsymbol{Z}$ if the corrections Δc_i are not used to update the parameters $\boldsymbol{c}$?

The first two questions can be answered by comparing Equations 4-8, 4-10, 4-11 and 4-12 with steps 2, 3, 4, and 7 in Table 3-3. It can be seen that the weights updating procedure proposed by Wold *et al.* (1989) relies on the PLS1 algorithm (Table 3-3) to compute the weights updating parameters. More precisely, the PLS1 algorithm is nested within the weights updating procedure and is used to compute the first latent variable ($\boldsymbol{s}$) of the PLS regression model which relates $\boldsymbol{Z}$ to $\boldsymbol{u}$. Consequently, the nested input weights $\boldsymbol{v}$ are scaled accordingly to the nested regression coefficient b between $\boldsymbol{s}$ and $\boldsymbol{u}$ and are used to evaluate the input weights updating parameters $\Delta \boldsymbol{w}$. This is based on the assumption that the matrix $\boldsymbol{Z}$ is a linearized expansion of the input scores $\boldsymbol{t}$ with respect to the output scores $\boldsymbol{u}$ and that a single latent variable PLS model is enough to regress the linear mapping between $\boldsymbol{Z}$ and $\boldsymbol{u}$. However the input weights updating parameters $\Delta \boldsymbol{w}$ are not directly related to $\boldsymbol{s}$, $\boldsymbol{u}$ and $\boldsymbol{X}$. Instead they are computed from the input weights of the nested PLS model ($\boldsymbol{v}$). This implies that the nested PLS1 algorithm may have an impact on the correction parameters $\Delta \boldsymbol{w}$ drawn from $[b \cdot \boldsymbol{v}]$. In other words, it may numerically reduce the efficiency of the updating procedure. This can also be seen when considering that the input weights of the nested PLS model are scaled to unit norm (Equation 4-10) and the regression coefficient b (Equation 4-13) lies between -1 and 1.

In order to address the three issues, consider the linear approximation:

$$u = f_{\infty} + \left.\frac{\partial f}{\partial c}\right|_{\infty} \cdot \Delta c + \left.\frac{\partial f}{\partial w}\right|_{\infty} \cdot \Delta w \qquad \textbf{4-16}$$

which can be rewritten as:

$$u = Z \cdot \Delta v \qquad \textbf{4-17}$$

where:

$$Z = \left[f_{\infty} \quad \frac{\partial f}{\partial c} \quad \frac{\partial f}{\partial w} \right] \qquad \textbf{4-18}$$

and:

$$\Delta v = \begin{bmatrix} 1 \\ \Delta c \\ \Delta w \end{bmatrix}. \qquad \textbf{4-19}$$

The vector Δv comprising the corrections Δw_m can be evaluated directly from the regression of u on Z giving:

$$\Delta v = (Z^T \cdot Z)^- \cdot Z^T \cdot u \qquad \textbf{4-20}$$

where $(Z^T \cdot Z)^-$ is the pseudo inverse of the matrix $(Z^T \cdot Z)$. Furthermore, if the corrections Δc_i are not used, then they can be considered constant and the linear approximation can be reduced to:

$$u = f_{\infty} + \left.\frac{\partial f}{\partial w}\right|_{\infty} \cdot \Delta w \qquad \textbf{4-21}$$

which can be rewritten as:

$$u = Z \cdot \Delta v \qquad \textbf{4-22}$$

where:

$$Z = \left[f_{\infty} \quad \frac{\partial f}{\partial w} \right] \qquad \textbf{4-23}$$

and:

$$\Delta v = \left[\begin{array}{c} 1 \\ \Delta w \end{array} \right]. \qquad \textbf{4-24}$$

Again the vector Δv comprising the corrections Δw_m can be evaluated directly from the regression of $\boldsymbol{u}$ on $\boldsymbol{Z}$ giving:

$$\Delta v = (Z^T \cdot Z)^{-} \cdot Z^T \cdot u \,. \qquad \textbf{4-25}$$

The last modification leads to a further interpretation and possible solution of the updating procedure. The mismatch, $\boldsymbol{e}$, between the value of $\boldsymbol{u}$ given by:

$$u = Y \cdot q \qquad \textbf{4-26}$$

and the value given by the non-linear mapping:

$$\hat{u} = f(t, c) \qquad \textbf{4-27}$$

can be denoted by:

$$e = u - \hat{u} \,. \qquad \textbf{4-28}$$

Through the application of a Newton-Raphson approximation, $\boldsymbol{e}$ can be rewritten as:

$$e = u - \hat{u} = u - f_{\infty} = \left. \frac{\partial f}{\partial w} \right|_{\infty} \cdot \Delta w \qquad \textbf{4-29}$$

where the only unknowns are the factors Δw_m. Thus by combining the partial derivatives $\frac{\partial f}{\partial w_m}$ into a matrix $\boldsymbol{Z}$, the mismatch $\boldsymbol{e}$ can be written as:

$$\boldsymbol{e} = \boldsymbol{Z} \cdot \Delta \boldsymbol{w} \qquad \textbf{4-30}$$

and the corrections Δw_m can be regressed directly as follows:

$$\Delta \boldsymbol{w} = (\boldsymbol{Z}^T \cdot \boldsymbol{Z})^- \cdot \boldsymbol{Z}^T \cdot \boldsymbol{e}\,. \qquad \textbf{4-31}$$

This leads to the development of three variants of the quadratic PLS algorithm.

4.5 Variants of the Quadratic PLS algorithm

Starting with the Newton-Raphson linearization of the non-linear inner mapping, it is possible to derive three different procedures for updating the input weights $\boldsymbol{w}$ which are more related to the Taylor Series expansion of the non-linear inner mapping than the procedure originally proposed by Wold *et al.* (1989). This is a consequence of writing the Taylor Series expansion in matrix form and solving it for the correction factors $\Delta \boldsymbol{w}$ by least squares regression of the vector $\boldsymbol{u}$ on the matrix $\boldsymbol{Z}$:

$$\boldsymbol{u} = \boldsymbol{Z} \cdot \Delta \boldsymbol{v}\,. \qquad \textbf{4-32}$$

These three developments of the weights updating procedure differ from each other in the way in which the matrix $\boldsymbol{Z}$, containing the known terms of the Taylor Series expansion of the function $f(\cdot)$, is constructed and consequently the way in which the correction factors $\Delta \boldsymbol{w}$ are regressed. In particular, depending upon the manner in which the matrix $\boldsymbol{Z}$ is constructed, the pseudo inverse of the matrix $\boldsymbol{Z}$ may need to be used. If the matrix $\boldsymbol{Z}$ is rank deficient as a result of some of the partial derivatives of the function $f(\cdot)$ being linearly correlated with themselves, or alternatively with the function $f(\cdot)$ itself, the pseudo inverse is required. For example, when

using a second order (quadratic) polynomial relationship for the inner mapping between the t and u scores:

$$u = c_0 + c_1 \cdot t + c_2 \cdot t^2 + e \qquad \textbf{4-33}$$

the first order Taylor Series expansion around the point f_{00} can be written as:

$$u = f_{00} + \Delta c_0 + \Delta c_1 \cdot t + \Delta c_2 \cdot t^2 + \sum_{m=1}^{M} \left(c_1 + 2 \cdot c_2 \cdot t\right) \cdot x_m \cdot \Delta w_m \qquad \textbf{4-34}$$

and the matrix Z as:

$$Z = \left[f_{00} \quad 1 \quad t \quad t^2 \quad \left[\left(c_1 + 2 \cdot c_2 \cdot t\right) \cdot x_m\right]\right] \qquad \textbf{4-35}$$

in which the third column [t] is linearly correlated with any of the last M columns $\left[\left(c_1 + 2 \cdot c_2 \cdot t\right) \cdot x_m\right]$. In Equations 4-34 and 4-35 the summation between scalar quantities and column vectors is defined as the summation between the scalar quantities and each element of the column vector, while the product between the column vector given by $\left(c_1 + 2 \cdot c_2 \cdot t\right)$ and the column vector x_m, which comprises the measurement on the m-th input variable, is defined as the scalar product between each pair of corresponding elements in the two vectors (element by element product).

By using the pseudo inverse of the matrix Z, numerical problems can arise from the algorithms used to evaluate the pseudo inverse. This issue, however, is beyond the scope of this work since it is not related to the PLS algorithm itself but to the software used to perform the calculations themselves. For simplicity the new three weights updating procedures, are denoted by PLS_A, PLS_B and PLS_C respectively. The major features of the three algorithms are summarised hereafter:

The Modified PLS_A Algorithm

$$Z = \left[f_{\infty} \quad \frac{\partial f}{\partial c} \quad \frac{\partial f}{\partial w} \right] \tag{4-36}$$

$$u = Z \cdot \Delta v \qquad \rightarrow \qquad \Delta v = (Z^T \cdot Z)^{-} \cdot Z^T \cdot u \tag{4-37}$$

$$\Delta v \rightarrow \Delta w \tag{4-38}$$

The Modified PLS_B Algorithm

$$Z = \left[f_{\infty} \quad \frac{\partial f}{\partial w} \right] \tag{4-39}$$

$$u = Z \cdot \Delta v \qquad \rightarrow \qquad \Delta v = (Z^T \cdot Z)^{-} \cdot Z^T \cdot u \tag{4-40}$$

$$\Delta v \rightarrow \Delta w \tag{4-41}$$

The Modified PLS_C Algorithm

$$Z = \left[\frac{\partial f}{\partial w} \right] \tag{4-42}$$

$$e = Z \cdot \Delta w \qquad \rightarrow \qquad \Delta w = (Z^T \cdot Z)^{-} \cdot Z^T \cdot e \,. \tag{4-43}$$

Referring back to the original non-linear PLS algorithm, the three modified non-linear PLS algorithms (PLS_A, PLS_B, PLS_C) arise as a consequence of changing the weights updating procedure whilst retaining the overall framework of the original non-linear PLS algorithm. Of the three procedures, the third approach (PLS_C) is more precise in its formulation since the mismatch between the value of u, given by $u = Y \cdot q$, and the value of $\hat{u}$, given by the non-

linear mapping, $\hat{u} = f(t,c)$, is only related to the input weights w by means of the correction factors Δw. In fact, from an algebraic perspective, i.e. with respect to the product between the matrices and vectors involved in the least squares regression, in PLS_A and PLS_B the correction factors Δw are derived through the vector Δv, as defined in Equations 4-19 and 4-24, which addresses the mismatch between u and $\hat{u}$ not only to w, but also to c and $\hat{u}$. However only the portion corresponding to w is used, whilst the fraction relating to c and $\hat{u}$ is disregarded. As a consequence, in PLS_A and PLS_B the potential of the updating procedure is not fully used (as in the original weights updating procedure proposed by Wold *et al.*). In particular, with respect to PLS_A, the factors Δc embody that part of the mismatch between u and $\hat{u}$ which affects the (matrix) calculation of Δv and hence deprives the factors Δw of their potential. With respect to PLS_B some kind of corrective action can be taken such as scaling the vector Δv in such a way that the element of it corresponding to f_{00} is equal to unity. Differently the PLS_C approach exploits the weights updating procedure in full by addressing the mismatch between u and $\hat{u}$ solely to w (Equations 4-29, 4-30 and 4-31). For this reason only the weights updating procedure corresponding to the PLS_C approach solely is considered in the remainder of the thesis and is referred to as the *error based weights updating procedure*. Consequently the quadratic PLS_C approach is named Error Based Quadratic PLS algorithm. A detailed description of the modified NIPALS algorithm for Quadratic PLS with the input weights updating procedure is given hereafter.

0	mean centre and scale X and Y	
1	set the output scores u equal to a column of Y	
2	regress columns of X on u	$w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	normalise w to unit length	$w = \frac{w}{\|w\|}$
4	calculate the input scores	$t = \frac{X \cdot w}{w^T \cdot w}$
5	fit the non-linear mapping	$c \leftarrow \text{fit}[u = f(t) + e]$

6	calculate non-linear prediction of u	$\hat{u} = f(t,c)$
7	regress columns of Y on $\hat{u}$	$q^T = \frac{\hat{u}^T \cdot Y}{\hat{u}^T \cdot \hat{u}}$
8	normalise q to unit length	$q = \frac{q}{\|q\|}$
9	calculate new output scores	$u = \frac{Y \cdot q}{q^T \cdot q}$
10	compute input weights updating parameters Δw as described above, i.e. Equations 4-29, 4-30 and 4-31.	
11	compute new input weights w	$w = w + \Delta w$
12	normalise w to unit length	$w = \frac{w}{\|w\|}$
13	calculate new input scores	$t = \frac{X \cdot w}{w^T \cdot w}$
14	check convergence on t: if YES goto 15 ELSE goto 5	
15	calculate the X loadings	$p^T = \frac{t^T \cdot X}{t^T \cdot t}$
16	fit the non-linear mapping	$c \leftarrow \text{fit}[u = f(t) + e]$
17	calculate non-linear prediction of u	$\hat{u} = f(t,c)$
18	calculate input residual matrix	$E = X - t \cdot p^T$
19	calculate output residual matrix	$F = Y - \hat{u} \cdot q^T$
20	if additional PLS dimension are necessary then replace X and Y by E and F and repeat steps 1 to 20.	

Modified NIPALS Algorithm for Quadratic PLS with Input Weights Updating Procedure.

4.6 Comparative Application of the Error Based Quadratic PLS Algorithms and the Original QPLS

In this work, the Error Based Quadratic algorithm is compared with the Quadratic PLS algorithm proposed by Wold *et al.* (1989) and the traditional linear PLS algorithm. These algorithms are compared on the basis of their performances when applied to data from a highly non-linear mathematical function and an industrial based pH simulation model.

4.6.1 Case Study 1 - A Synthetic Example.

The data for this example was generated from a non-linear function described by Cherkassky *et al.* (1995) in their paper comparing statistical and neural network methods for function estimation. The function has four uncorrelated random inputs which are uniformly distributed between -0.25 and 0.25 and is given by:

$$y = \exp(2 \cdot x_1 \cdot \sin(\pi \cdot x_4)) + \sin(x_2 \cdot x_3).$$

A data set comprising 500 points was generated and split into two sets: one set of 400 points for model building (training and cross validation) and a set of 100 points for model testing. A direct consequence of the four inputs being uncorrelated and randomly uniformly distributed between –0.25 and 0.25 is that the results, in terms of the amount of variance explained by the X-block are expected to be approximately the same for each latent variable.

The variance of the X-block and Y-block explained by each model is given in Tables 4.1a, 4.1b and 4.1c for linear PLS, the original quadratic PLS algorithm and the error based quadratic algorithm, respectively. From Table 4.1a the inability of the linear PLS algorithm to model the data set is clearly evident. No more than 0.59 % of the variance of the response variable can be explained. Figure 4.1 highlights the inability of linear PLS to fit a non-linear relationship which exists between the t and u scores, especially for the first and third latent variables, Figures 4.1a and 4.1c, respectively.

Tables 4.1b and 4.1c show the improvement that can be achieved by implementing the error based weights updating procedure with the quadratic PLS algorithm. The cumulative amount of

variability explained for the Y-block by the first two latent variables is seen to be greater for the error based quadratic algorithm than for the original quadratic PLS (92.3% compared with 79.9%). Overall the error based quadratic model with two latent variables explains more of the underlying variability in the data than the original quadratic PLS model with four latent variables (86.7 %). This is reflected not only in the value of the mean square prediction error for the training data set, Table 4.2a, and the mean square prediction error for the test data set, Table 4.2b, computed for the response variable y, but also in terms of the values of the mean total predicted error sum of squares (used as a criterion of goodness) given by cross validation performed on the training data set and given in Table 4.3.

The performance of the two quadratic PLS algorithms are comparable in terms of their modelling and predictive ability for the first latent variable. However, from the second latent variable onwards, the different weights updating procedures significantly affects the values of the input and output scores for each of the subsequent dimensions. This can be also seen from the scores scatter plots for each of the four dimensions shown in Figures 4.2 and 4.3 which illustrate the contrast in the two approaches. In particular, when considering Figures 4.2a and 4.3a it can be explained why the two approaches are very similar for the first latent variables. In fact the two narrow edges at the top right and top left of the scatter plot, and the large number of points falling in its central area force the mapping existing between the first set of input/output scores to fit a parabola passing through those three points, i.e. the edges and the centre of the cluster. Apparently the error based updating procedure cannot modify the shape or the spread of this cluster (i.e. the direction of the first latent variable and the projection of the input variables on it) enough to improve the fitting of the inner relation. Whilst comparing Figures 4.2b, 4.2c and 4.2d with 4.3b, 4.3c and 4.3d, the effect of the new updating procedure becomes evident. The u scores are less well spread in the case of the error based quadratic algorithm than in the original quadratic algorithm, (note the different scales on the u axis for Figures 4.2c and 4.2d and Figures 4.3c and 4.3d). This is even more clear when considering Figure 4.4, where the output score residuals ($\hat{u} - u$) for the second latent variable are depicted for the two quadratic approaches. Thus, one of the advantages of using the error based updating procedure appears to be the reduction in dispersion of the score scatter plot which leads to enhanced performance of the algorithm in terms of modelling capabilities. This follows from the fact that the greater the spread in the u, the poorer the fit of the inner model and vice versa.

Figure 4.5 show the actual and the predicted values for the unseen data. In particular Figure 4.5a shows the inability of the linear PLS methodology to predict the output. Figure 4.5b shows actual and predicted values for the original quadratic PLS algorithm, based upon a four latent variable model, whilst Figures 4.5c to 4.5d, illustrate the differences between the two and four latent variable models for the error based PLS quadratic algorithm. There is no real improvement to the results for the test data set by fitting a model with more than two latent variables. These figures clearly demonstrate the improved fit achievable using the error based quadratic model, even with only two latent variables, with respect to the original quadratic model with four latent variables, and confirm the idea that the error based weights updating procedure has better performances than the original procedure proposed by Wold *et al.* (1989).

4.6.2 Case Study 2 - Simulation of an Industrial pH Problem

pH-neutralisation systems are known to be highly non-linear and to exhibit severe time varying behaviour. For this reason they have been used as a benchmark for testing control algorithms (e.g. Henson *et al.*, 1994; Johansen *et al.*, 1997; Kim *et al.*, 1997; Kavšek-Biasizzo *et al.*, 1997). In this work, the dynamic model for the pH process described by Henson *et al.* (1994) is used. The process consists of a tank where a strong acid (HNO_3) is neutralised by a strong base (NaOH) in the presence of a buffer stream ($NaHCO_3$). For the purposes of this work the composition of the inlet streams (acid, base and buffer) have been kept fixed whilst the flowrates have initially been changed randomly and then kept constant until the process reached steady state, i.e. a constant value of the pH for the outlet stream. The flowrate of the outlet stream was also changed according to the inlet flowrates in order to maintain a constant level in the tank. A data set of 999 points was generated by randomly changing the inlet flowrates (Q1, Q2, Q3) and recording the value of the pH on the outlet stream at steady state. No noise was added to the pH measurements. The flowrates Q1, Q2, Q3 and Q4 were used as the predictor variables and the pH measurements as the response variable. In Figures 4.6a to 4.6d scatter plots of the steady state data between each stream and the corresponding pH value on the outlet stream are given. From these plots it can be observed that there are two main clusters of points. This occurrence was expected to lead to some misleading effect in terms of modelling and prediction capabilities of the regression models. The data was split into two subsets, a training data set comprising 799 observations and a test data set of 200 values.

The results of applying the linear PLS, original quadratic and error based quadratic algorithms are presented in Tables 4.4, 4.5 and 4.6 for the percentage of variability explained for the ***X*** and ***Y***-blocks, the mean squared prediction error for training and testing data sets and the mean total predicted sum of squares based upon cross validation, respectively. Overall, the total percentage of variability explained for each of the three methods is similar, although for the error based quadratic PLS algorithm an additional 3% of the variability is explained within the ***Y*** block. The major difference between the algorithms is that the majority of the variability in the training data set can be explained by one latent variable for the error based approach, whilst for the other two approaches two latent variables are required to explain 90% of the total variance. The error based algorithm clearly outperforms both the linear and original quadratic approaches. Even with four latent variables neither the linear nor the original quadratic approaches achieve the same level of variability explained as that achieved by one latent variable for the error based quadratic approach. These results are also reflected in the mean square prediction error for training and testing data for the ***Y***-block given in Tables 4.5a and 4.5b, respectively and in the mean of the total predicted sum of squares calculated using cross validation, given in Table 4.6.

Furthermore the linear PLS algorithm outperforms the original quadratic PLS approach from the second latent variables onwards, uncovering the fact that the weights updating procedure proposed by Wold *et al.* (1989) can have some side effect on the input weights and can reduce the modelling capabilities of the PLS methodology. A possible reason for the poor behaviour of the original quadratic algorithm is that the pH tritation curve is more cubic-shaped in its trajectory hence a linear fit is a better approximation than a quadratic shaped model. This inability to describe the underlying cubic structure is less apparent with the error based quadratic algorithm example indicating that the modified approach provides greater flexibility for modelling compared with the original quadratic algorithm which has a fairly rigid structure.

The difference between the three algorithms in terms of explained variability of the input data set (Table 4.4) should also be noted. The error based quadratic PLS algorithm requires only 10% of the variance of the input data set to capture more than 98% of the variance of the output signal. In contrast the linear PLS algorithm and the original quadratic PLS algorithms require almost 90% of the variance of the input data matrix to explain around 90% of the variability of the output variable. Thus it can be deducted that the error based approach discriminates between the approximation of the input data matrix and the prediction of the output data matrix, putting

more emphasis on the prediction than on the approximation. Furthermore, it can be deducted that the new weights updating procedure modifies the projection of the input variables in such a way that only that part of the input data which is really predictive of the output data is projected on the first latent variable(s), whilst the remaining is left in the residuals.

This feature of the error based updating procedure could not be observed on the non-linear function used before (§ 4.6.1) because it was known that there was no structure relating the input variables. Whilst in this case the error based weights updating procedure shows to enhance the prediction capabilities of the methodology extracting from the input data set only that fraction of variability (10%) required to achieve optimal prediction of the output data. This reflects on the score scatter plots for each of the four dimensions given in Figure 4.7, 4.8 and 4.9. In particular Figures 4.8a and 4.9a show the major feature of the error based updating procedure, that is to change the spread of the projection of the input data seeking for that linear combination of the input variables which is more predictive of the output scores, and hence of the output variables. In other words, the error based approach operates as a filtering tool directly on the latent variables, moving all the signal which are not predictive of the output, but mainly of the structure of the input data set, to the lower order components. Thus, it discriminates between input and input/output structures and enhance the prediction of the output variables more than the approximation of the input variables. Whilst the weights updating procedure proposed by Wold *et al.* (1989) lacks this discriminating feature, even though it does still modify the projection of the input variables onto the latent variable space. Because of this it can happen that the linear PLS algorithm does actually outperform the original QPLS approach, as it was observed in Tables 4.4a and 4.4b. In fact, from the score scatter plots for the linear PLS model it can be seen that the spread of the scores on the second latent variable (Figure 4.7b) around the fitted line is less than the spread of scores around the parabola fitted by the QPLS algorithm on the second latent variable (Figures 4.8b), although the QPLS outperform the linear PLS on the first component (Figures 4.7a and 4.8a). This appears more evidently when comparing the plots of the residuals for the second latent variable for the two algorithms. It can be observed that the mismatch between predicted and actual output scores for the linear PLS model (Figure 4.10a) is smaller than the corresponding one for the original quadratic PLS (Figure 4.10b).

Figures 4.11, 4.10 and 4.12 illustrate the final predictions for the test data set for the three approaches for one, two, three and four latent variable based models. The predictions for the test data set clearly illustrate the improvements that are achievable through the application of the

error based approach. A one latent variable model, using the error based approach, provides a sufficiently robust final model for prediction purposes. This has clear advantages if the final objective of the model is for routine process monitoring purposes at operator level. Thus the error based technique appears to outperform the existing algorithms in terms of explaining the overall variability in the data. This is more clearly evident from the test data set, the results of which clearly indicate that the error based algorithm is not prone to overfitting which is a criticism levelled against many empirical based approaches.

4.7 Discussion and Conclusions

The weights updating procedure proposed by Wold *et al.* (1989) represents a breakthrough in non-linear PLS modelling. The updating of the weights of the input outer mapping in order to make the input outer mapping consistent with the inner mapping is a step forward. However the procedure appears cumbersome and Wold *et al.* (1989) observed that it is "*fairly complicated and slow to converge when the data lack structure*". The original algorithms was subsequently broken down into a number of steps, with each step being interrogated as to how it contributed to the overall modelling ability of the algorithm. From this investigation, three key issues relating to the weights updating procedure were identified. Resulting from these issues, three variants of the original quadratic PLS algorithm were developed.

With respect to the original quadratic PLS algorithm, the three modified non-linear PLS algorithms are intrinsically related to the original quadratic PLS algorithm proposed by Wold *et al.*(1989) in the sense that they are based upon the use of a Newton-Raphson linearization of the inner mapping between the $\boldsymbol{t}$ and the $\boldsymbol{u}$ scores for the updating of the input weights. However, they differ from the approach of Wold in the way the Newton-Raphson approximation is used within the updating procedure. Furthermore the three modified algorithms differ between themselves in the way the mismatch between the value of $\boldsymbol{u}$, given by $\boldsymbol{u} = \boldsymbol{Y} \cdot \boldsymbol{q}$, and that given by the non-linear mapping, $\hat{\boldsymbol{u}} = f(\boldsymbol{t}, \boldsymbol{c})$ impacts upon the input weights. In particular the first two algorithms (PLS_A and PLS_B) are more closely related, both to one another in terms of mathematical structure and to the Wold updating procedure (they indirectly address the mismatch to the input weights). For the third approach (PLS_C), the error based weights updating procedure, the input weights correction factors are directly regressed onto the value of

($\boldsymbol{u} - \hat{\boldsymbol{u}}$). A consequence of this approach is that it leads to a change in the balance of emphasis in terms of the variability explained by the $\boldsymbol{X}$-block and the $\boldsymbol{Y}$-block.

The previous PLS algorithms (linear and original quadratic) tend to strike a balance in terms of the variance explained between the approximation of the regressor variables $\boldsymbol{X}$ and the fit to the response variables $\boldsymbol{Y}$. In contrast, the error based quadratic PLS algorithm is more focused towards explaining the variability associated with the quality variables, the $\boldsymbol{Y}$-block. A consequence of this is that when the "*optimal*" number of latent variables is selected, on the basis of the variability explained for the quality variables, the level of variability explained in terms of the regressor variables is lower for the modified algorithm compared to the original quadratic algorithm. However, since in process modelling, the objective is to infer the values of the quality variables on the basis of the measurements of the process variables, the new PLS algorithm satisfies this criteria. It might be conjectured, that the emphasis the modified algorithm is placing upon the prediction of the response variables is being addressed by the input weights $\boldsymbol{w}$ no longer being directly related to the input matrix $\boldsymbol{X}$ and to the output scores $\boldsymbol{u}$, as in the linear PLS algorithm. Thus the error based quadratic algorithm, omits the direct link (*exchange*) between the input weights and the output scores as a result of the new weights updating procedure and the input weights cease to be directly related to the input matrix $\boldsymbol{X}$. In fact the weights correction factors $\Delta \boldsymbol{w}$ are mainly related to the mismatch between $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$. This does not completely hold true for the original quadratic PLS algorithm where the effect of the weights updating procedure is encompassed within the series of regressions and products between the matrices and vectors, i.e. the PLS1 iteration nested within the weights updating procedure which reduced the efficiency of the weights updating procedure proposed by Wold *et al.* (1989). This reversal in terms of behaviour is clearly seen from the results for the pH simulation model.

The power of the approach is that it can be generalised to any function which is differentiable to the second order, and is shown in the next chapter where it is extended to the case of PLS algorithms base on the use of sigmoid neural networks and radial basis function networks.

	***X*-block**		***Y*-block**	
LV	Single LV	Cumulative	Single LV	Cumulative
1	21.1275	21.1275	0.5825	0.5825
2	24.9204	46.0479	0.0040	0.5866
3	28.6462	74.6942	0.0000	0.5866
4	25.3058	100.0000	0.0000	0.5866

Table 4.1a : Percentage of Variability Explained - Linear PLS Algorithm.

	***X*-block**		***Y*-block**	
LV	Single LV	Cumulative	Single LV	Cumulative
1	21.2623	21.2623	69.5765	69.5765
2	28.0744	49.3366	10.4197	79.9961
3	25.4617	74.7983	6.3379	86.3341
4	25.2017	100.0000	0.3932	86.7273

Table 4.1b : Percentage of Variability Explained - Original Quadratic PLS Algorithm.

	***X*-block**		***Y*-block**	
LV#	Single LV	Cumulative	Single LV	Cumulative
1	21.2049	21.2049	69.7688	69.7688
2	29.0818	50.2867	22.5636	92.3324
3	25.1833	75.4700	1.7808	94.1132
4	24.5300	100.0000	0.3307	94.4439

Table 4.1c : Percentage of Variability Explained - Error Based Quadratic PLS Algorithm.

Table 4.1 : Comparison of Percentage of Variability Explained for the Linear, Quadratic and Error Based Quadratic PLS Algorithms for the Mathematical Function.

LV#	*1*	*2*	*3*	*4*
Linear PLS	0.0135	0.0135	0.0135	0.0135
Wold QPLS	0.0041	0.0027	0.0019	0.0018
Error Based QPLS	0.0041	0.0010	0.0008	0.0008

Table 4.2a : Mean Square Prediction Error (training data set).

LV#	*1*	*2*	*3*	*4*
Linear PLS	0.0207	0.0207	0.0207	0.0207
Wold QPLS	0.0044	0.0032	0.0026	0.0024
Error Based QPLS	0.0044	0.0017	0.0014	0.0014

Table 4.2b : Mean Square Prediction Error (testing data set).

Table 4.2 : Comparison of the Mean Square Prediction Error on Training and Testing Data for the Linear, Original Quadratic and Error Based Quadratic PLS Algorithms for the Mathematical Function.

LV#	*1*	*2*	*3*	*4*
Linear PLS	0.0141	0.0140	0.0140	0.0140
Wold QPLS	0.0048	0.0028	0.0026	0.0022
Error Based QPLS	0.0044	0.0012	0.0009	0.0008

Table 4.3 : Mean Total Predicted Sum of Squares for the Mathematical Function Models (Leave 10% out Cross Validation).

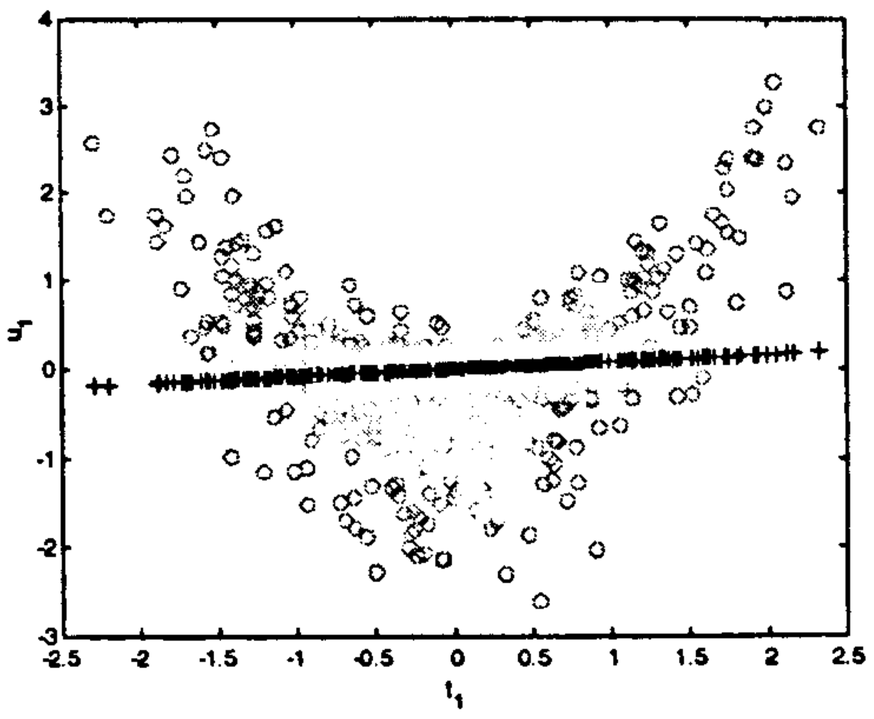

Figure 4.1a : Linear PLS, first latent variable scatter plot.

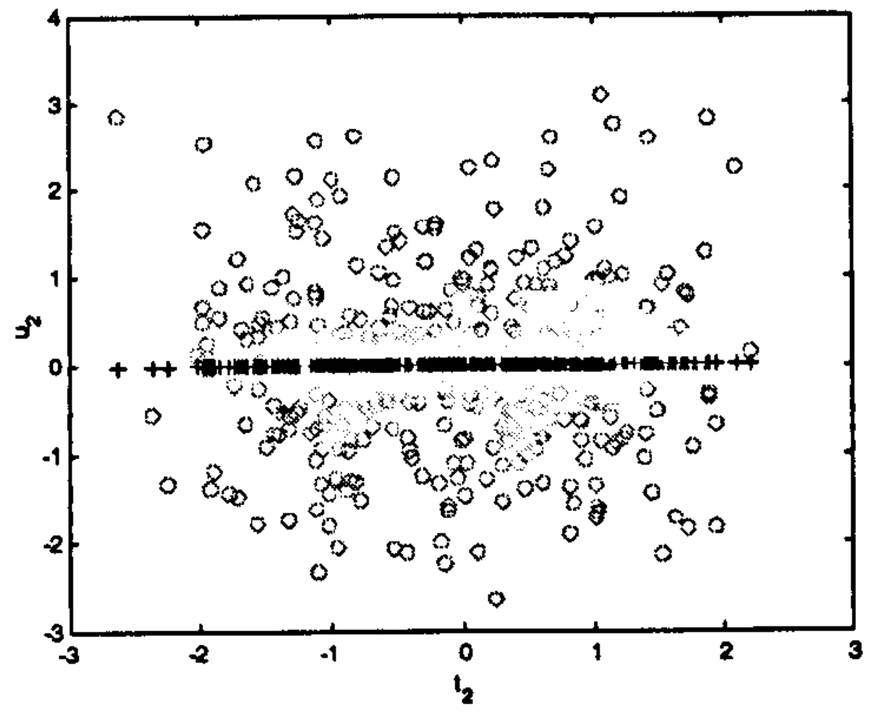

Figure 4.1b : Linear PLS, second latent variable scatter plot.

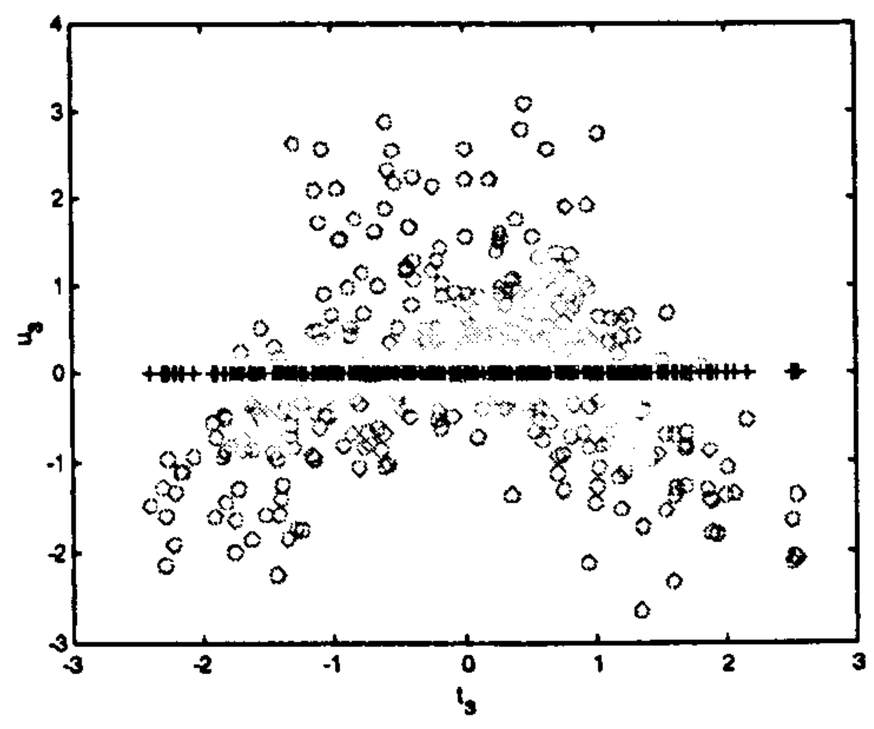

Figure 4.1c : Linear PLS, third latent variable scatter plot.

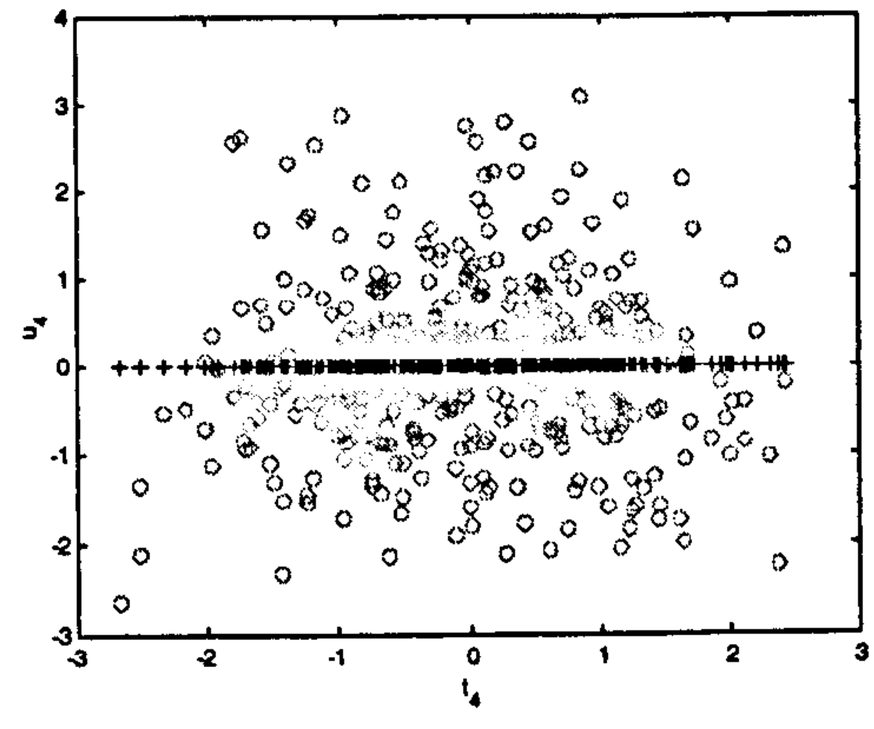

Figure 4.1d : Linear PLS, fourth latent variable scatter plot.

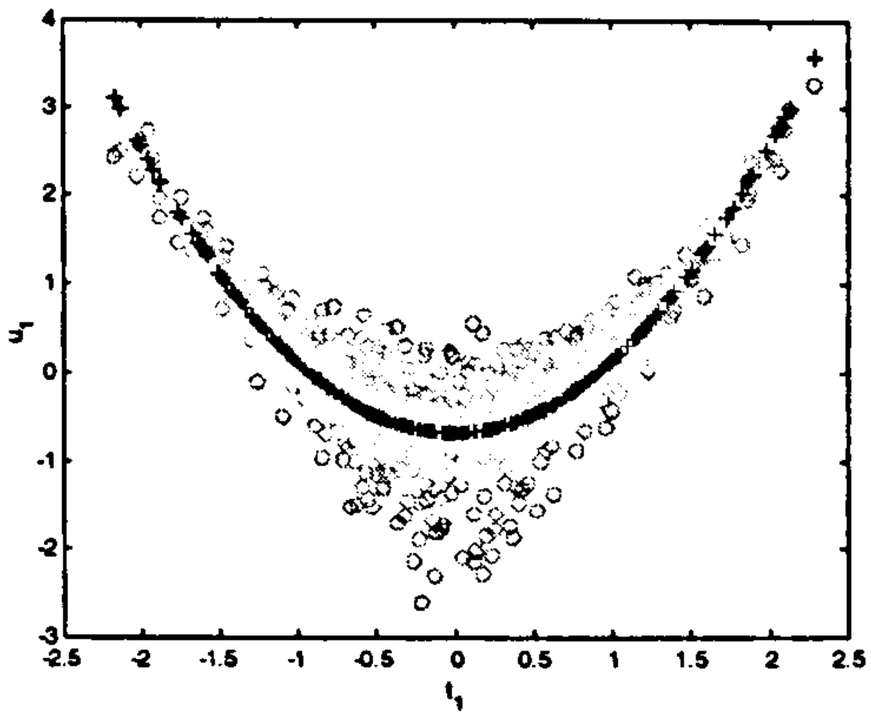

Figure 4.2a : Wold Quadratic PLS, first latent variable scatter plot.

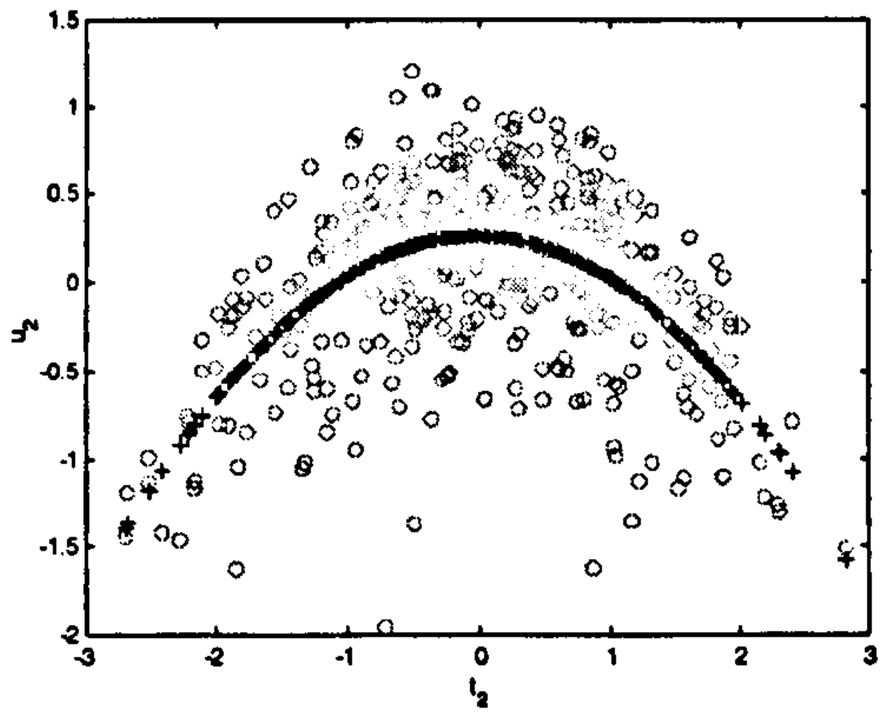

Figure 4.2b : Wold Quadratic PLS, second latent variable scatter plot.

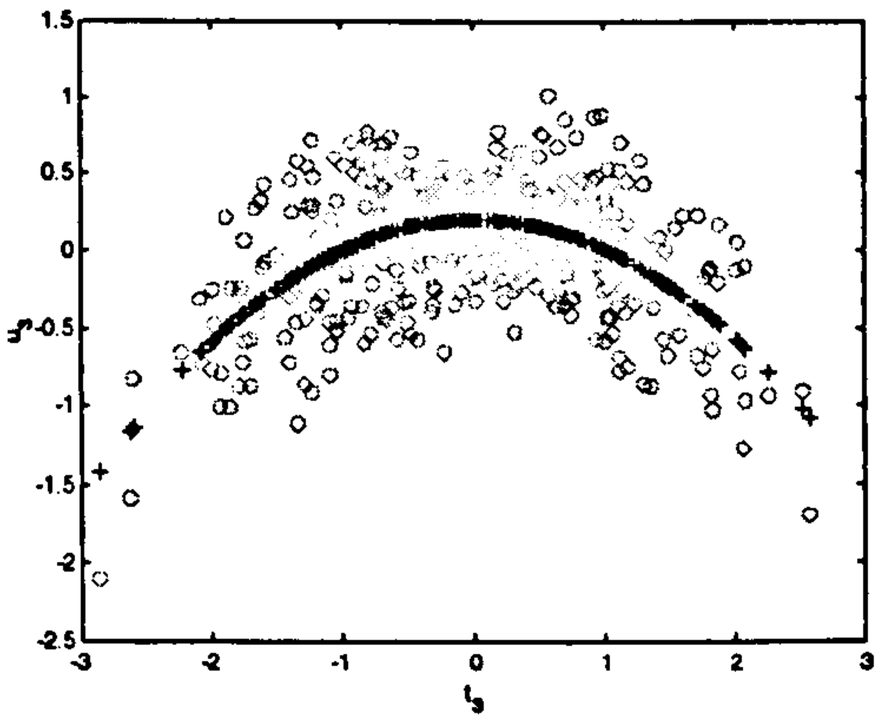

Figure 4.2c : Wold Quadratic PLS, third latent variable scatter plot.

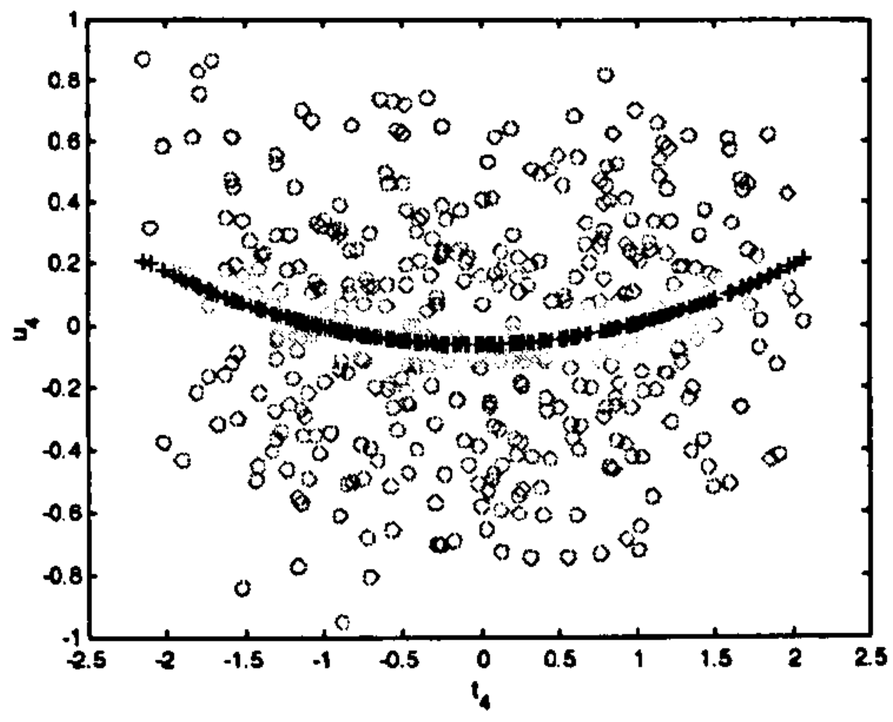

Figure 4.2d : Wold Quadratic PLS, fourth latent variable scatter plot.

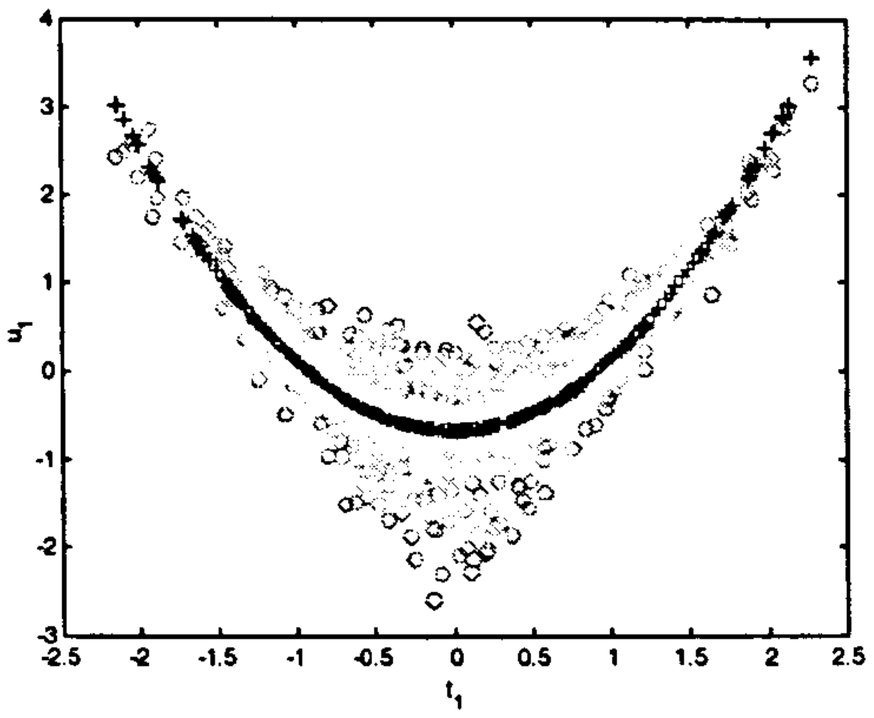

Figure 4.3a : Error Based Quadratic PLS, first latent variable scatter plot.

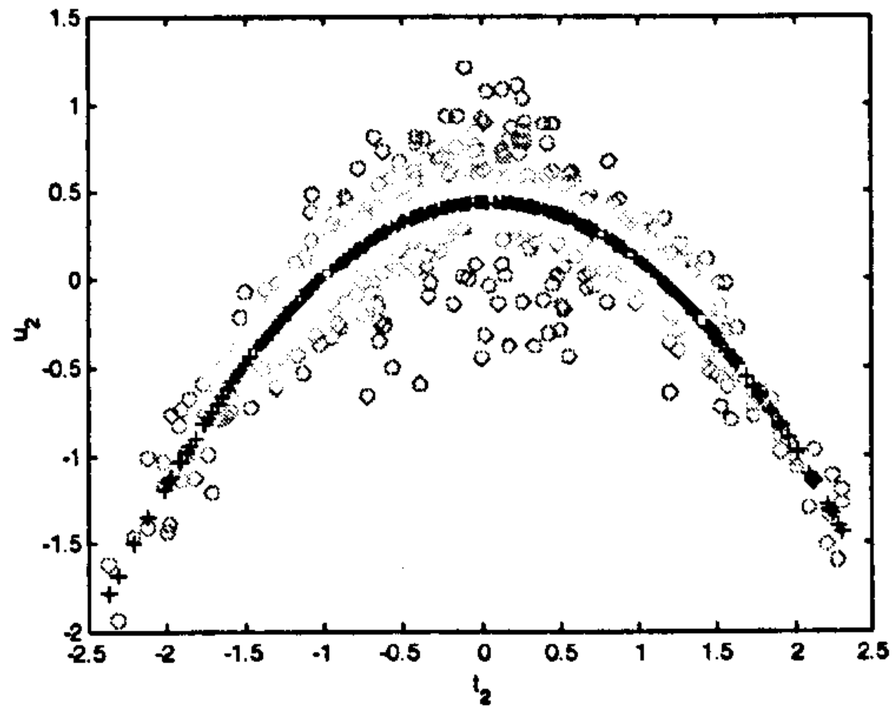

Figure 4.3b : Error Based Quadratic PLS, second latent variable scatter plot.

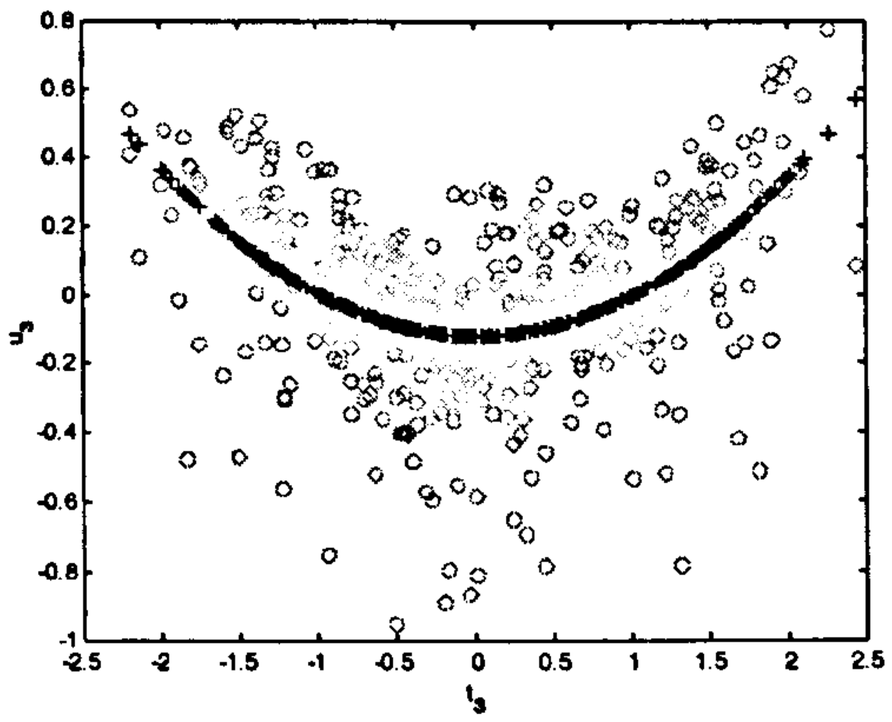

Figure 4.3c : Error Based Quadratic PLS, third latent variable scatter plot.

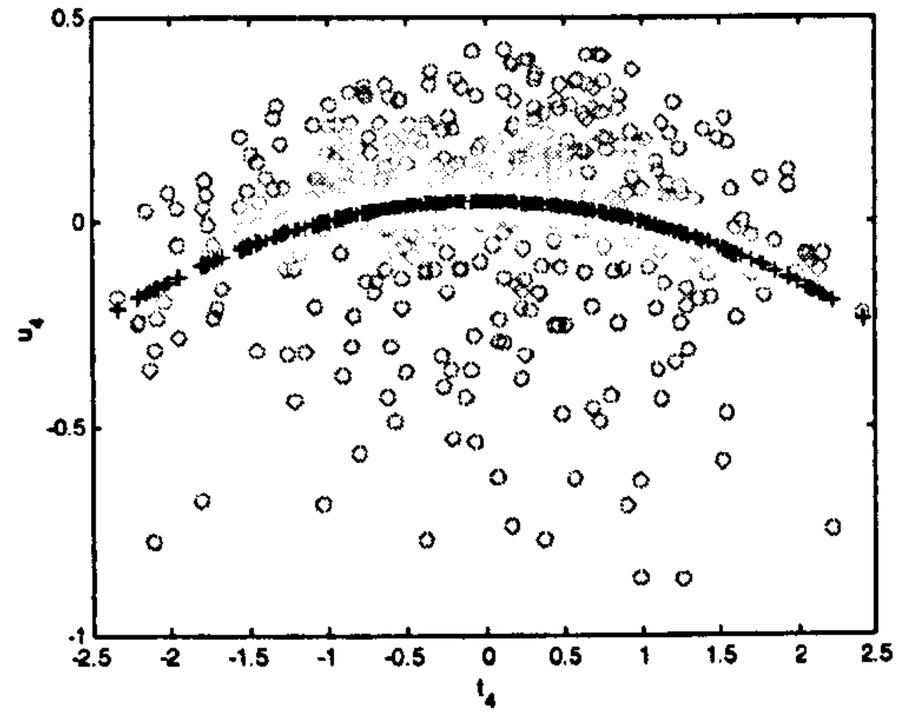

Figure 4.3d : Error Based Quadratic PLS, fourth latent variable scatter plot.

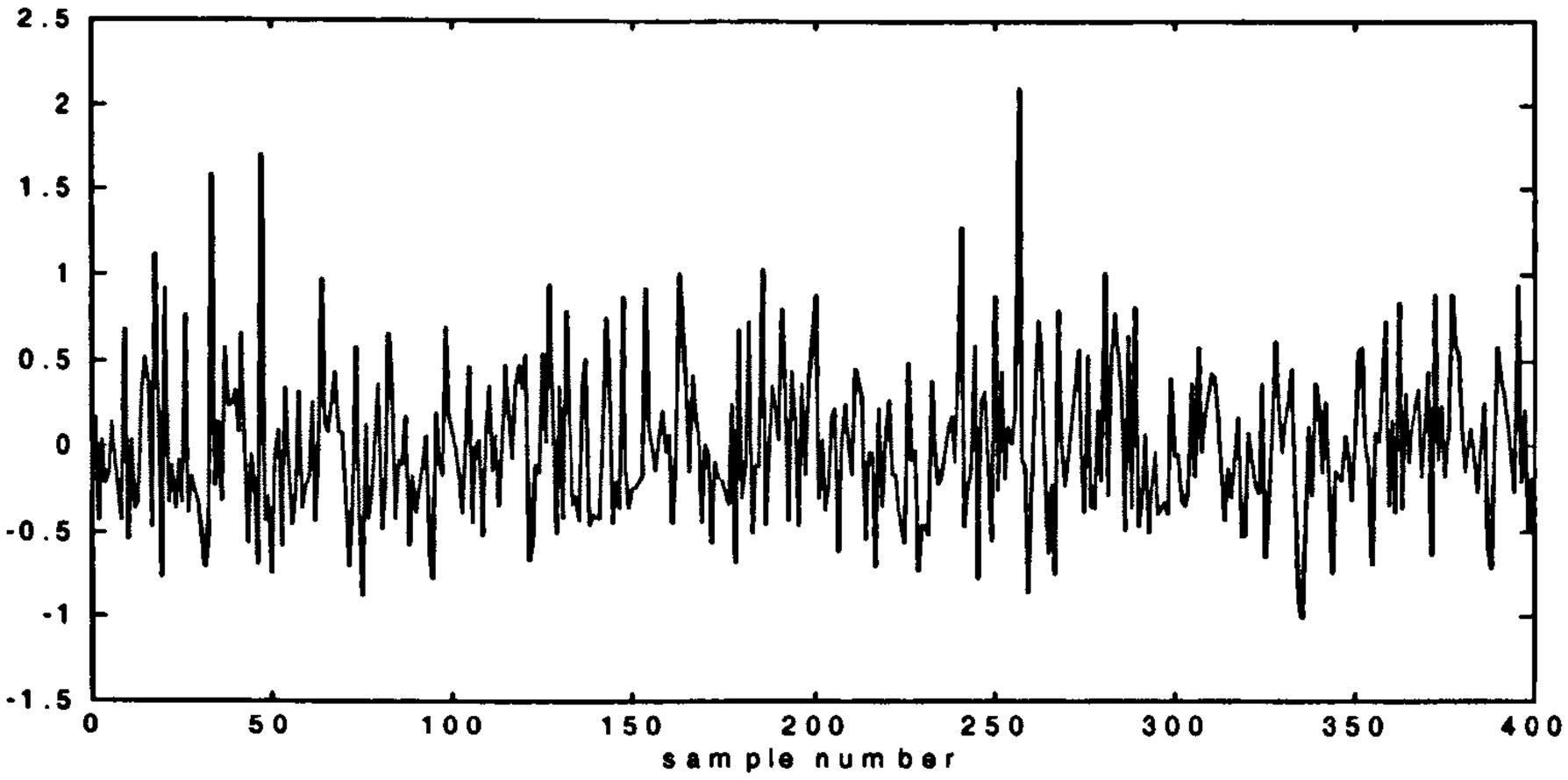

Figure 4.4a : Wold Quadratic PLS, output score residuals plot for 2[nd] latent variable.

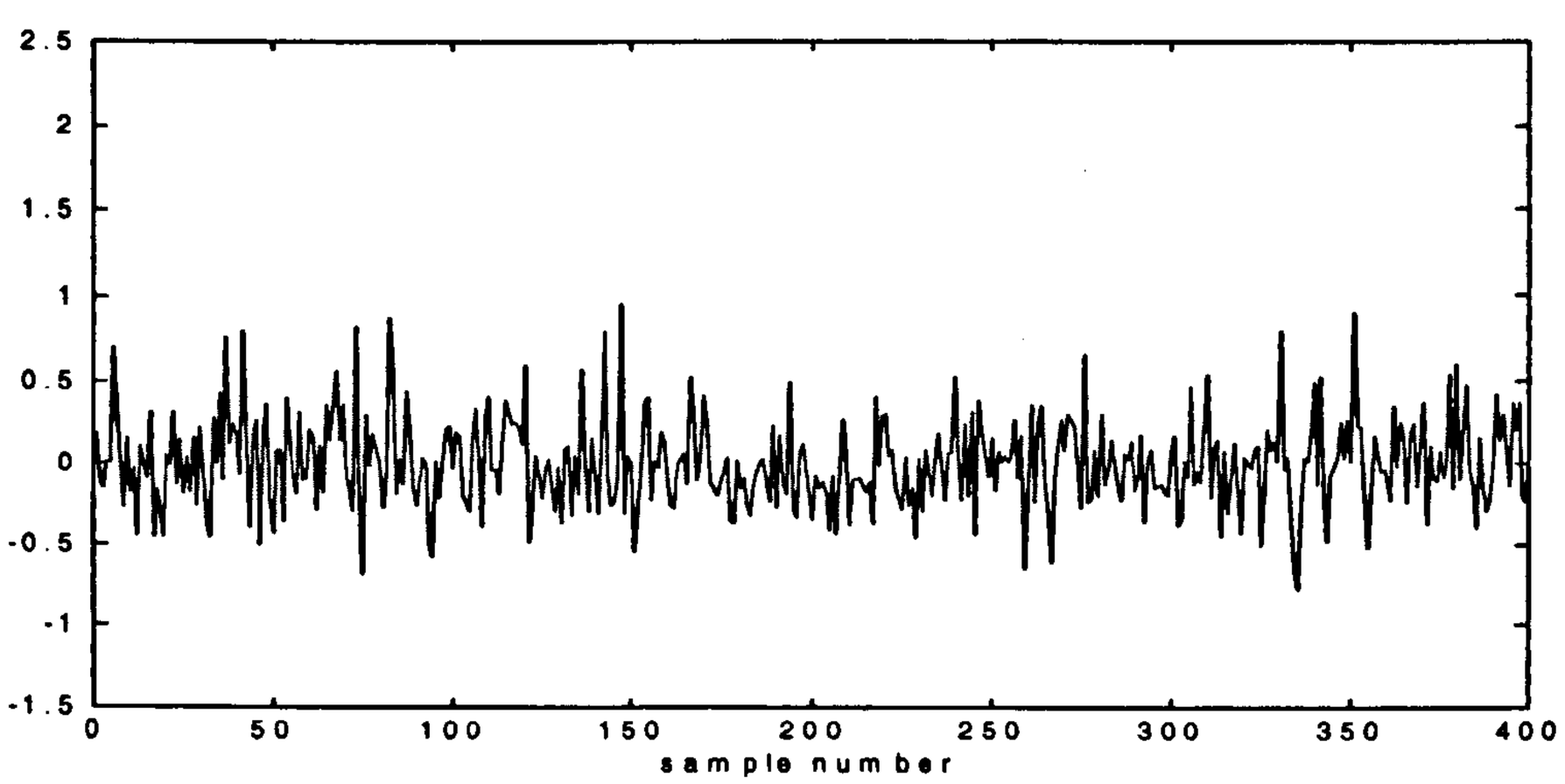

Figure 4.4b : Error Based Quadratic PLS, output score residuals plot for 2[nd] latent variable.

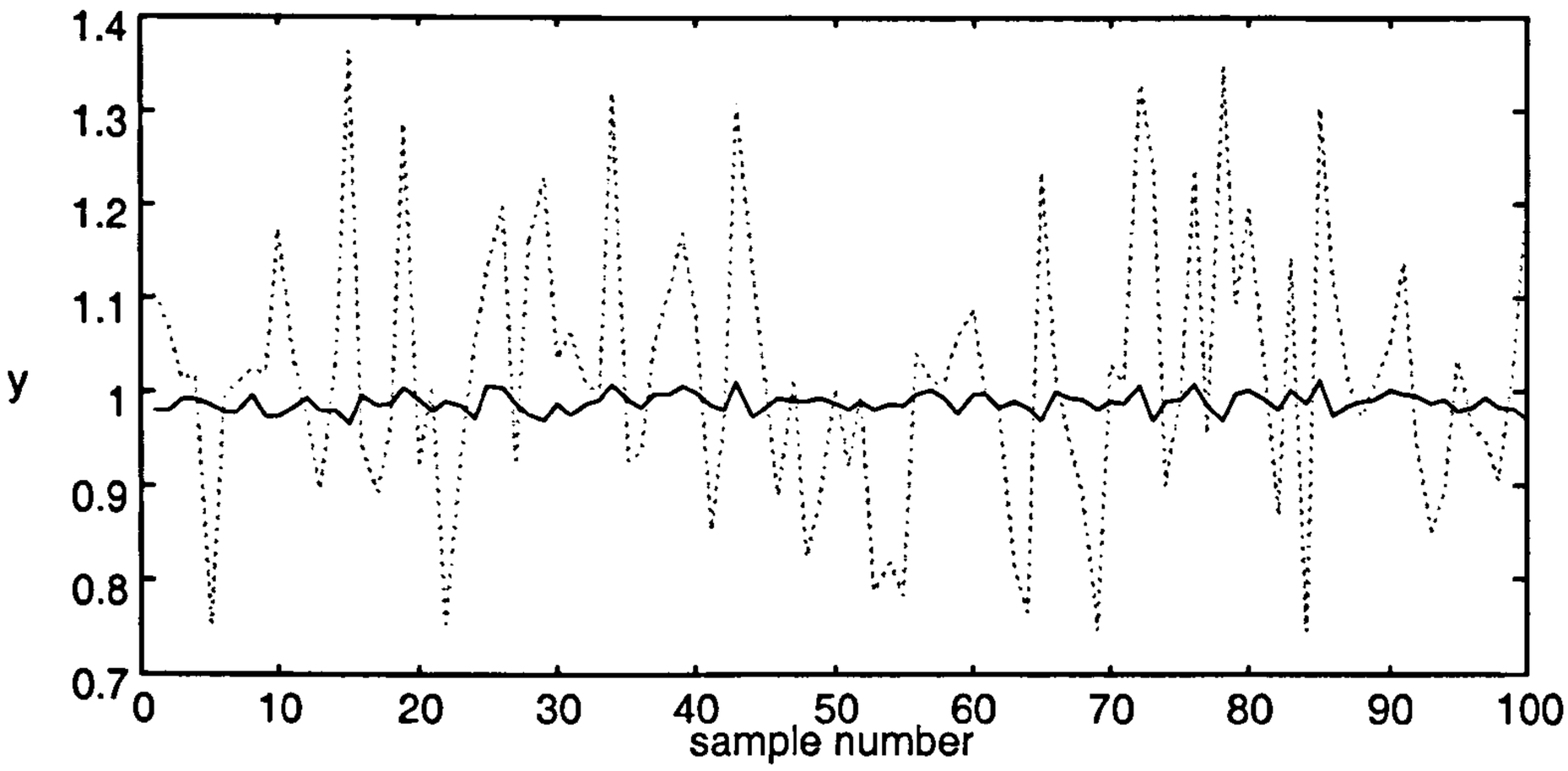

Figure 4.5a : actual (dotted light) versus predicted (full dark) output values for Linear PLS model using four latent variables.

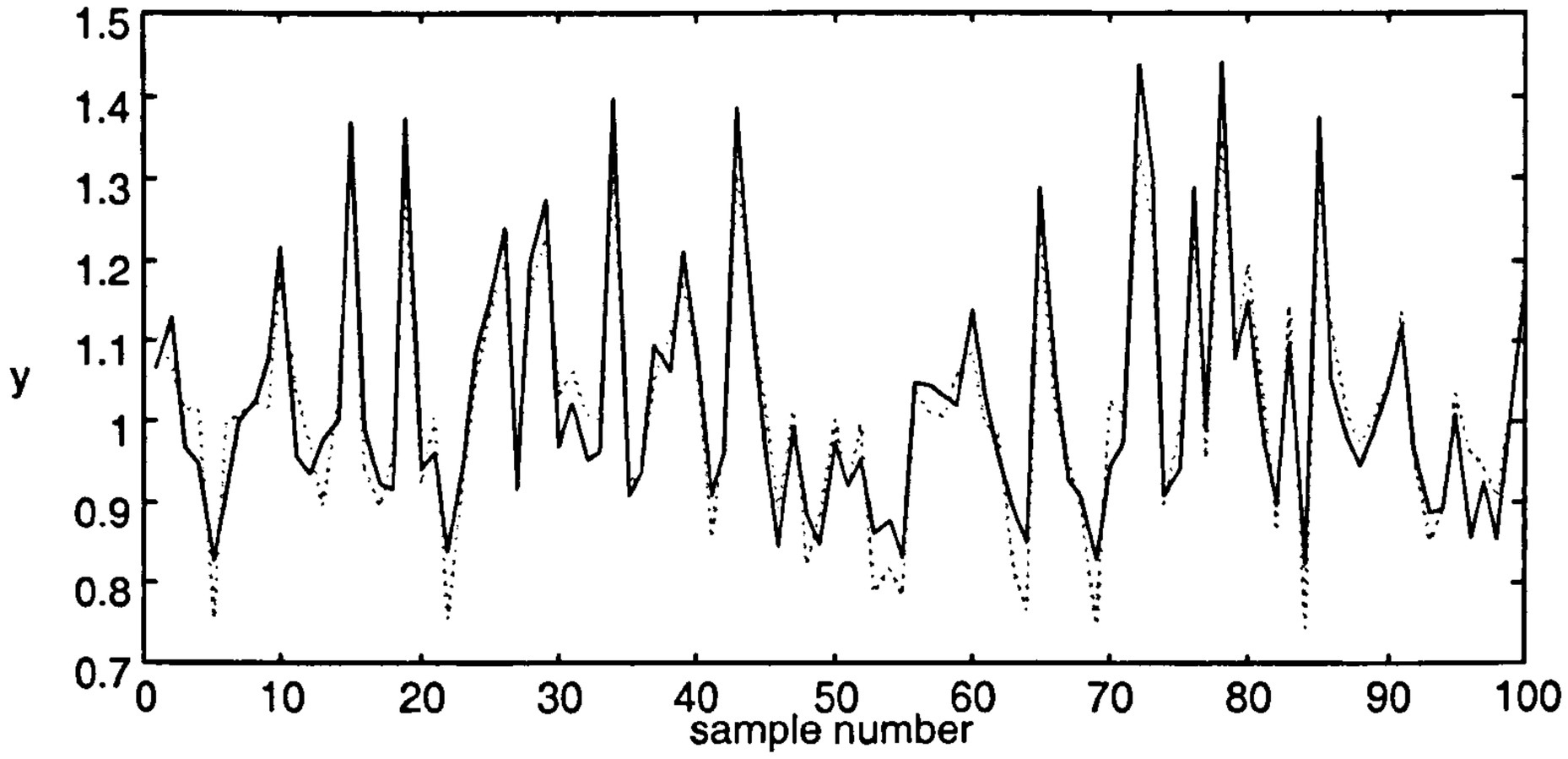

Figure 4.5b : actual (dotted light) versus predicted (full dark) output values for Wold Quadratic PLS model using four latent variables.

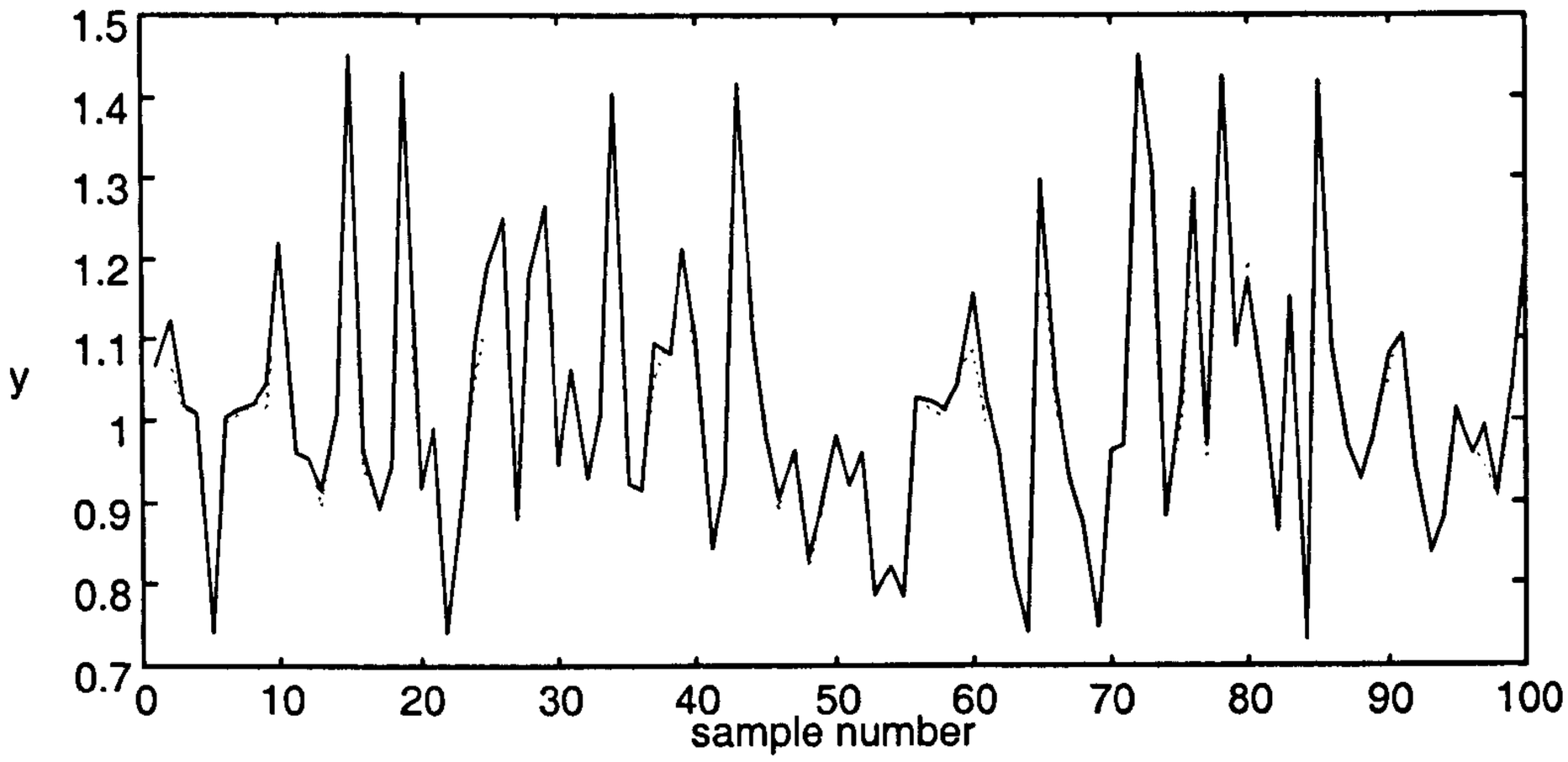

Figure 4.5c : actual (dotted light) versus predicted (full dark) output values for Error Based Quadratic PLS model using two latent variables.

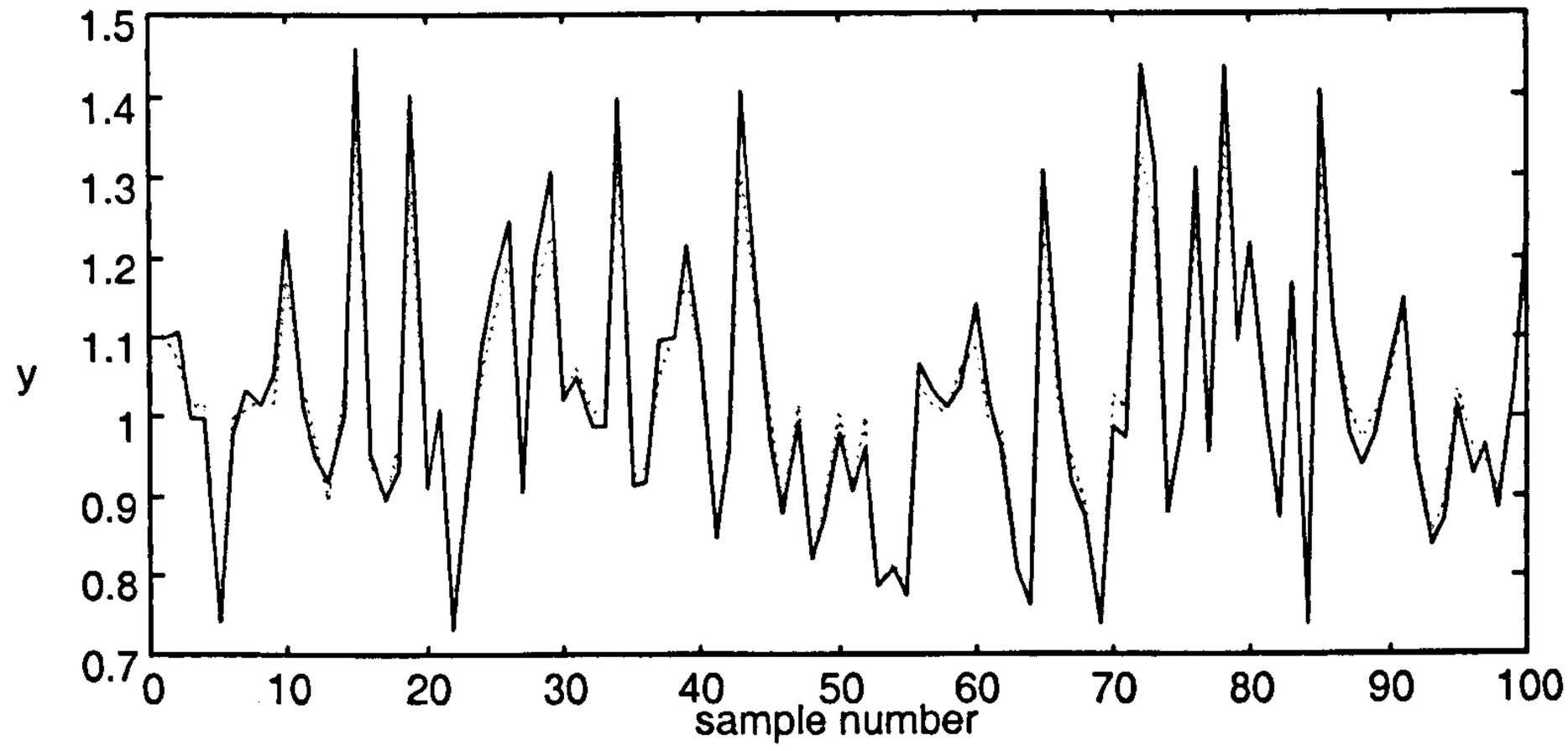

Figure 4.5d : actual (dotted light) versus predicted (full dark) output values for Error Based Quadratic PLS model using four latent variables.

	X-block		Y-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	60.5851	60.5851	42.6297	42.6297
2	31.9094	92.4945	49.6840	92.3138
3	7.5055	100.0000	3.4998	95.8136
4	0.0000	100.0000	0.0018	95.8154

Table 4.4a : Percentage of Variability Explained - Linear PLS Algorithm.

	X-block		Y-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	28.3636	28.3636	79.6772	79.6772
2	61.6078	89.9714	11.0981	90.7753
3	10.0286	100.0000	4.9767	95.7520
4	0.0000	100.0000	0.0018	95.7538

Table 4.4b : Percentage of Variability Explained - Original Quadratic PLS Algorithm.

	X-block		Y-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	10.1552	10.1552	98.7295	98.7295
2	80.8207	90.9760	0.1324	98.8619
3	8.5568	99.5328	0.1784	99.0404
4	0.4672	100.0000	0.0007	99.0410

Table 4.4c : Percentage of Variability Explained - Error Based Quadratic PLS Algorithm.

Table 4.4 : Comparison of Percentage of Variability Explained for the Linear, Quadratic and Error Based Quadratic PLS Algorithms for the pH Data.

LV#	*1*	*2*	*3*	*4*
Linear PLS	0.0372	0.0050	0.0027	0.0027
Wold Quadratic PLS	0.0132	0.0060	0.0028	0.0028
Error Based Quadratic PLS	0.0008	0.0007	0.0006	0.0006

Table 4.5a : Mean Square Error (training data set).

LV#	*1*	*2*	*3*	*4*
Linear PLS	0.0381	0.0054	0.0031	0.0030
Wold Quadratic PLS	0.0133	0.0060	0.0027	0.0027
Error Based Quadratic PLS	0.0009	0.0008	0.0007	0.0007

Table 4.5b : Mean Square Prediction Error (testing data set).

Table 4.5 : Comparison of the Mean Square Prediction Error on Training and Testing Data for the Linear, Original Quadratic and Error Based Quadratic PLS Algorithms for the pH Data.

LV#	*1*	*2*	*3*	*4*
Linear PLS	0.0373	0.0051	0.0027	0.0028
Wold QPLS	0.0132	0.0061	0.0028	0.0028
Error Based QPLS	0.0009	0.0008	0.0007	0.0007

Table 4.6 : Mean Total Predicted Sum of Squares for the pH Data Models (Leave 10% Out Cross Validation).

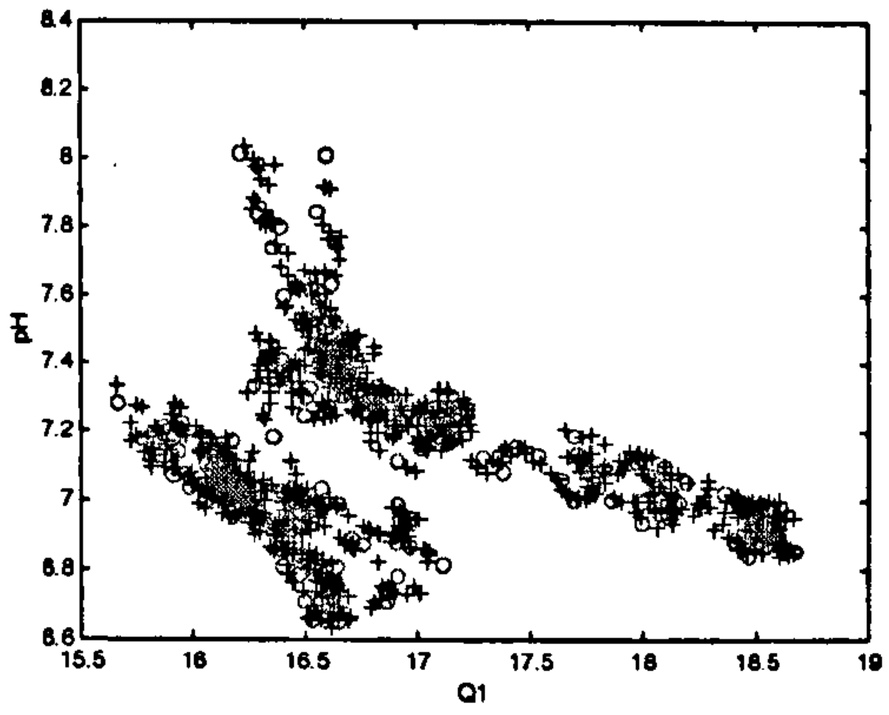

Figure 4.6a : inlet flowrate Q1 versus pH value on outlet stream (light + : training data; dark o : testing data).

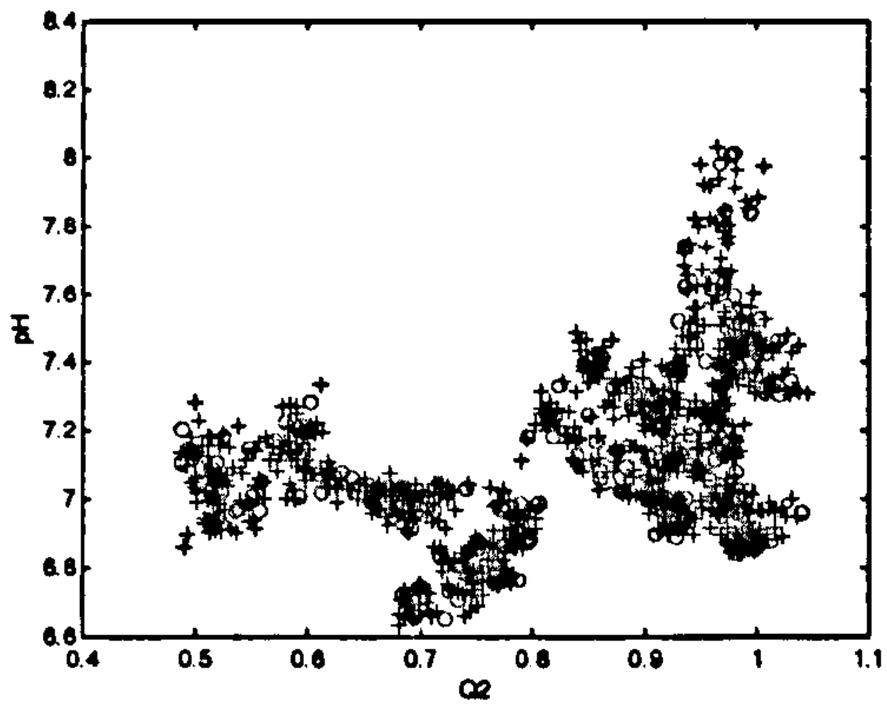

Figure 4.6b : inlet flowrate Q2 versus pH value on outlet stream (light + : training data; dark o : testing data).

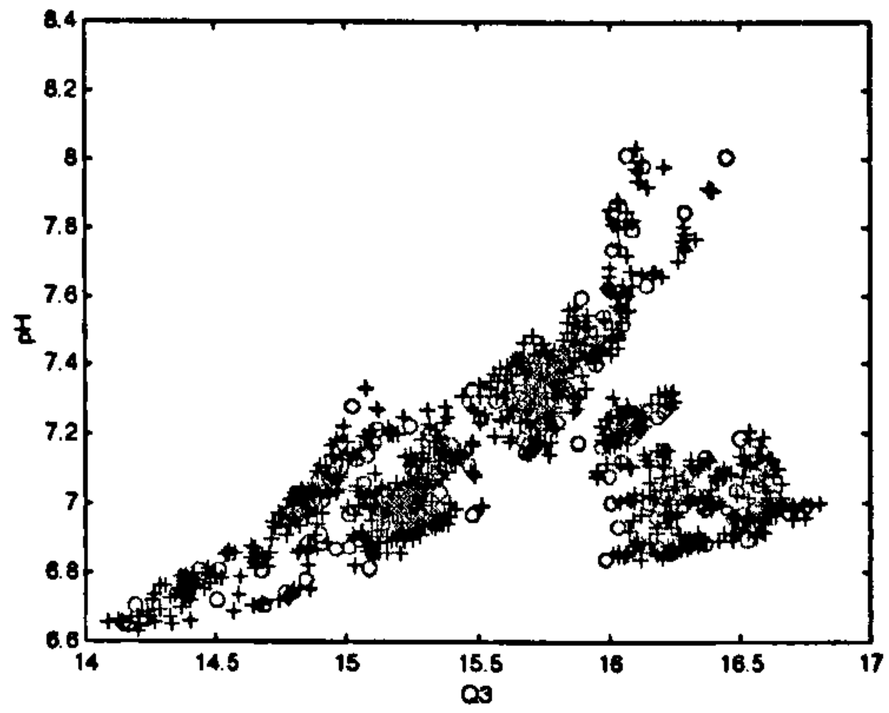

Figure 4.6c : inlet flowrate Q3 versus pH value on outlet stream (light + : training data; dark o : testing data).

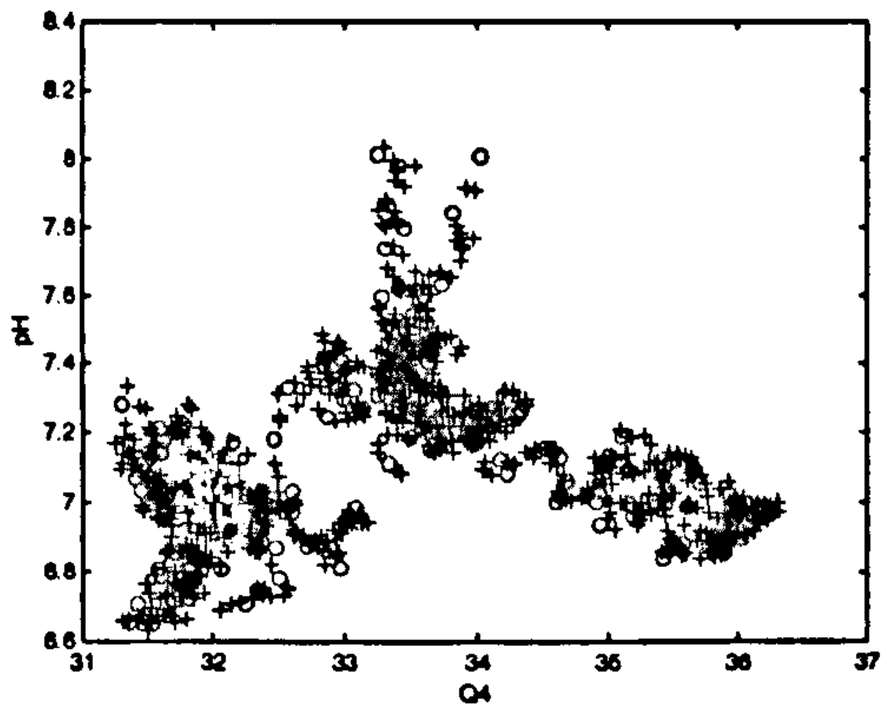

Figure 4.6d : outlet flowrate Q4 versus pH value on outlet stream (light + : training data; dark o : testing data).

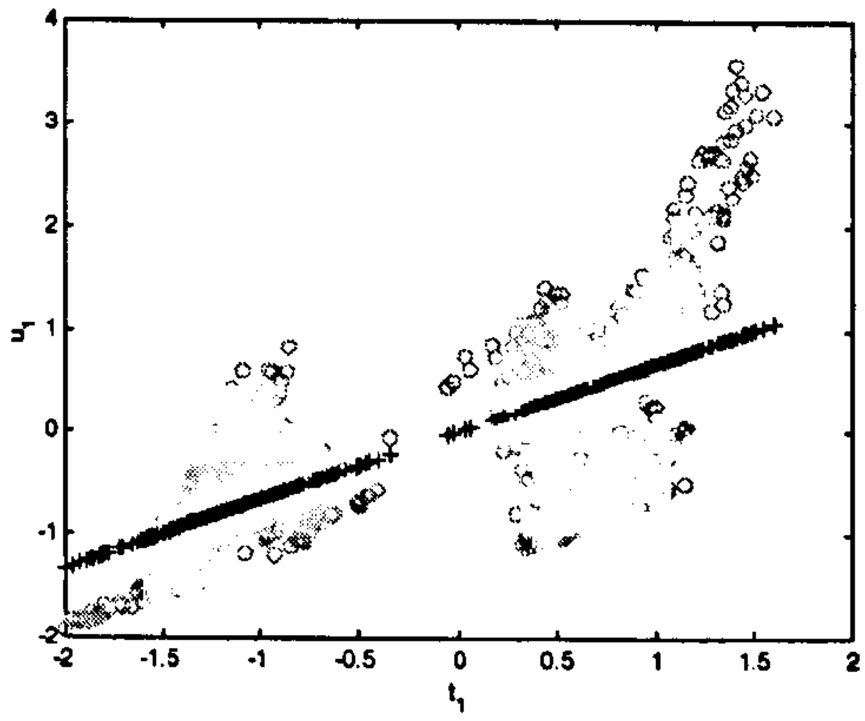

Figure 4.7a : Linear PLS, first latent variable scatter plot.

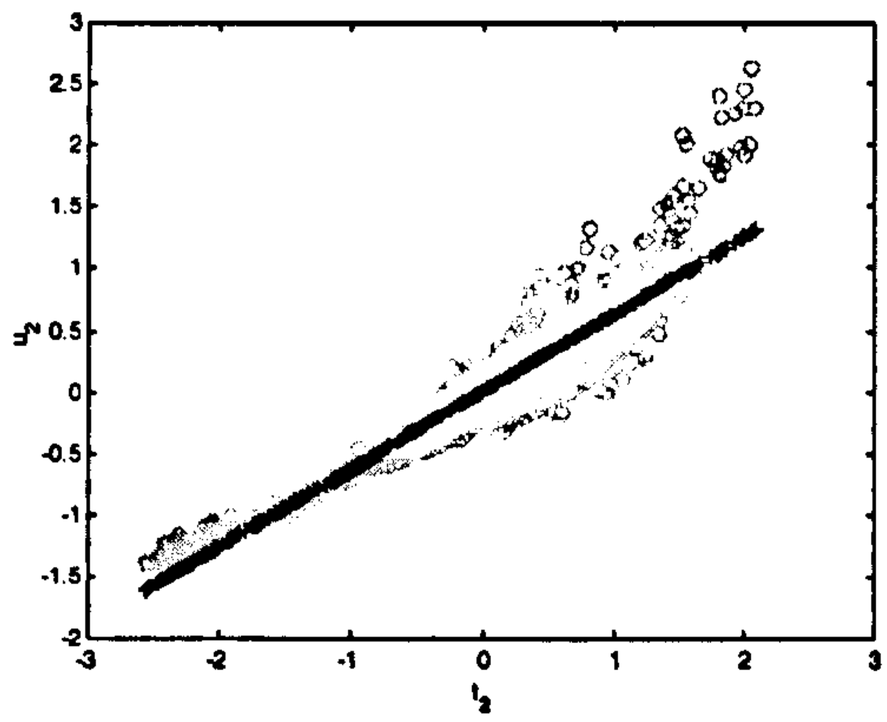

Figure 4.7b : Linear PLS, second latent variable scatter plot.

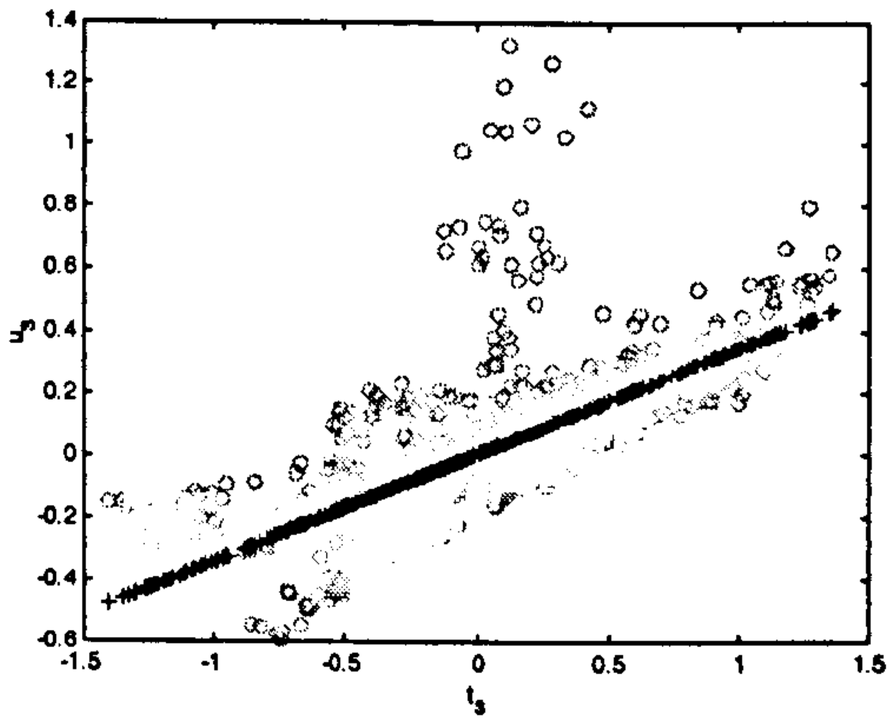

Figure 4.7c : Linear PLS, third latent variable scatter plot.

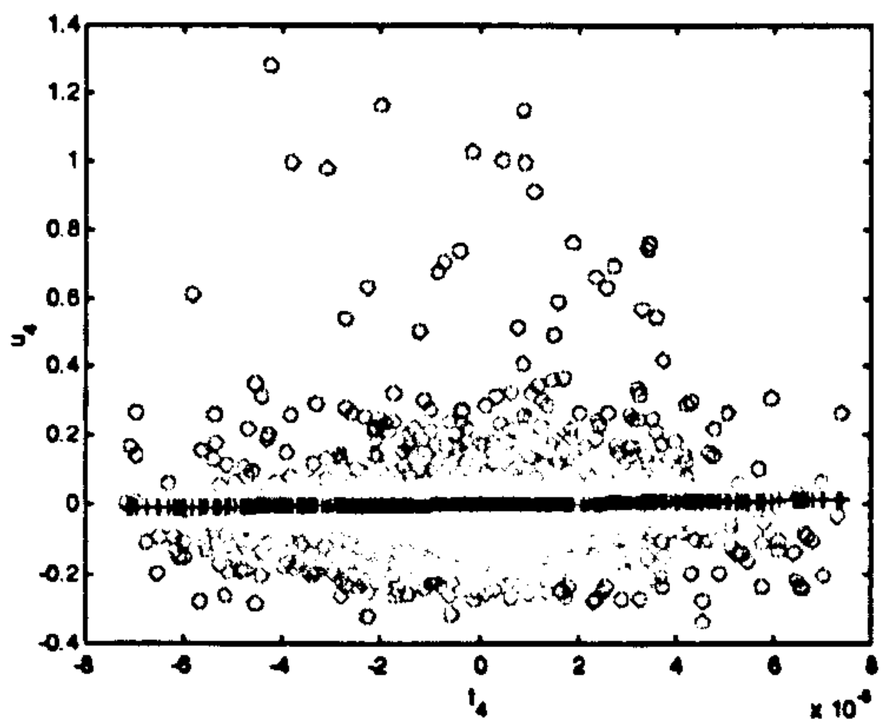

Figure 4.7d : Linear PLS, fourth latent variable scatter plot.

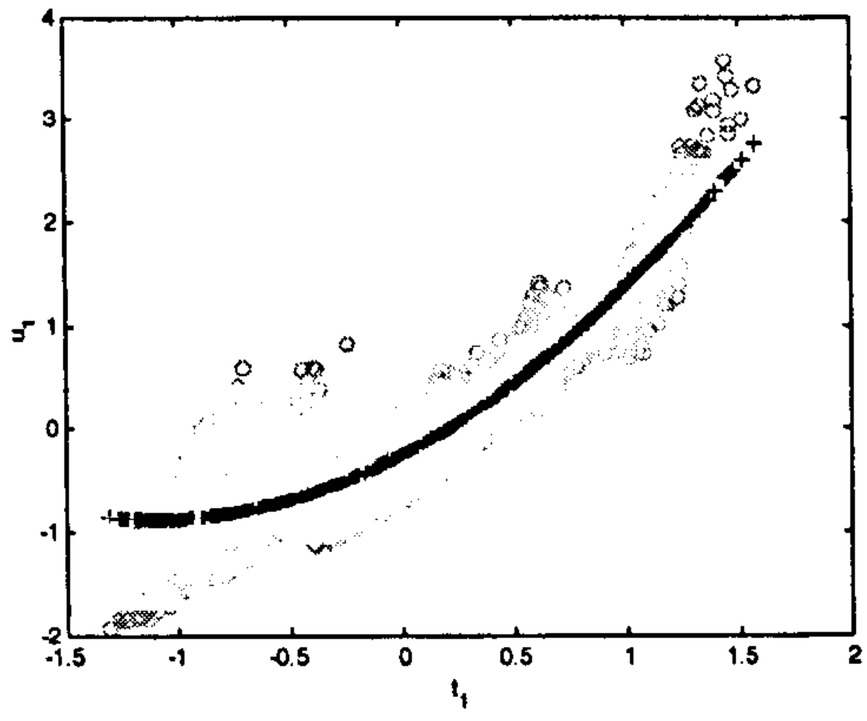

Figure 4.8a : Wold Quadratic PLS, first latent variable scatter plot.

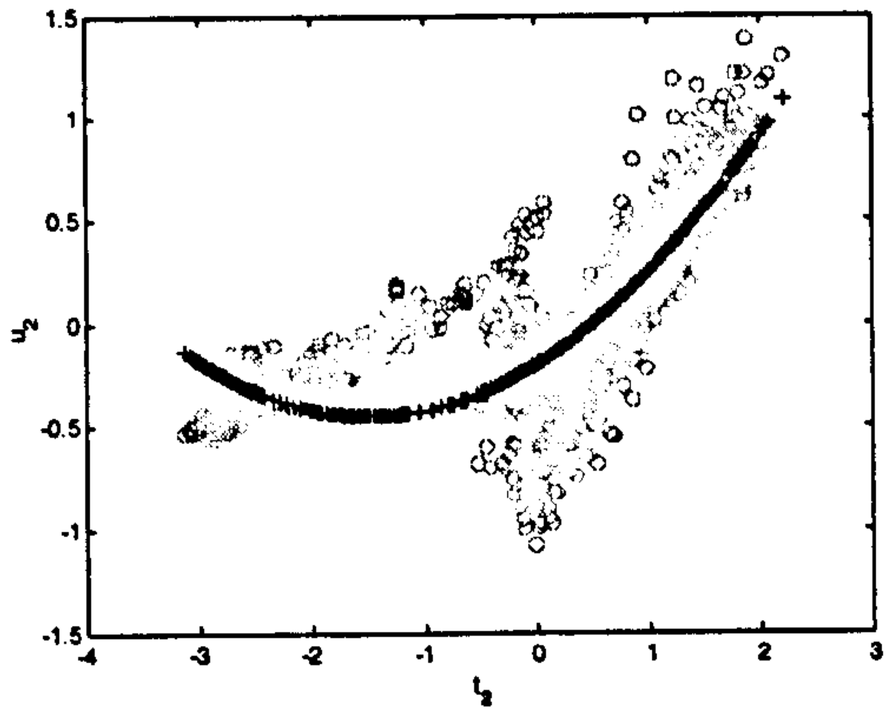

Figure 4.8b : Wold Quadratic PLS, second latent variable scatter plot.

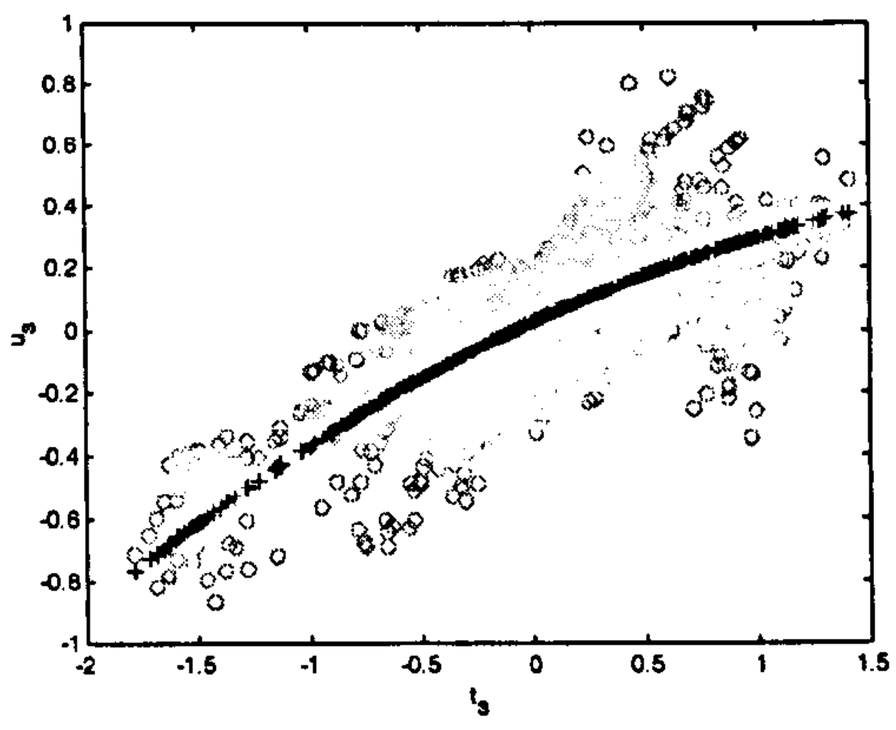

Figure 4.8c : Wold Quadratic PLS, third latent variable scatter plot.

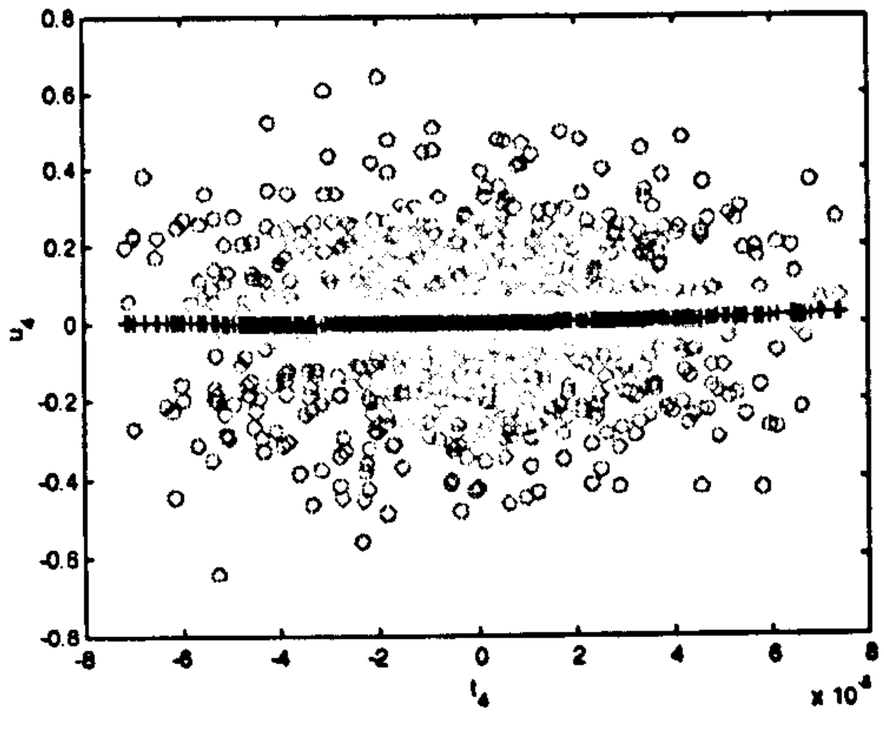

Figure 4.8d : Wold Quadratic PLS, fourth latent variable scatter plot.

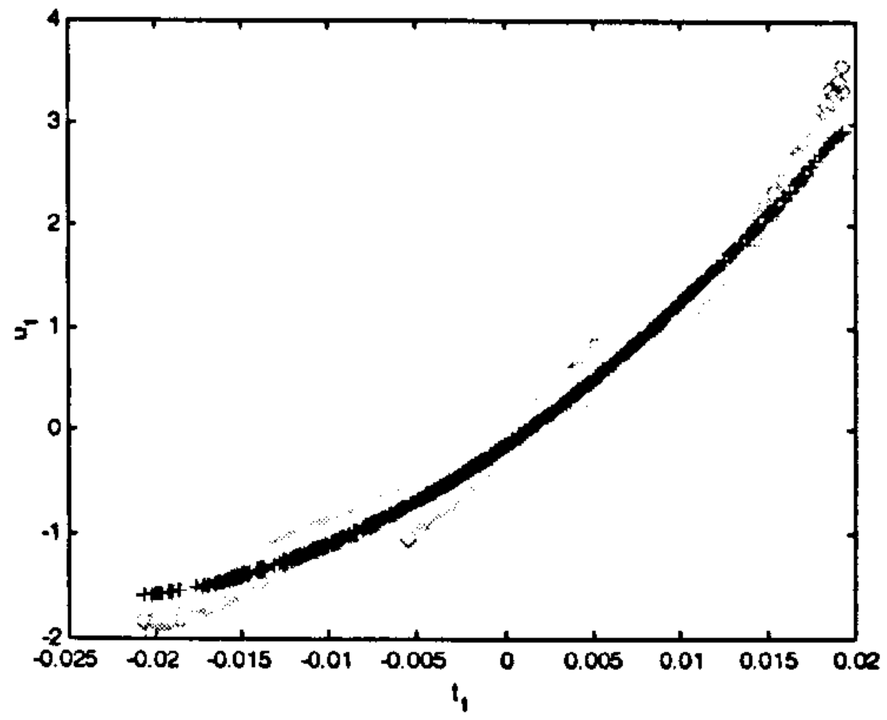

Figure 4.9a : Error Based Quadratic PLS, first latent variable scatter plot.

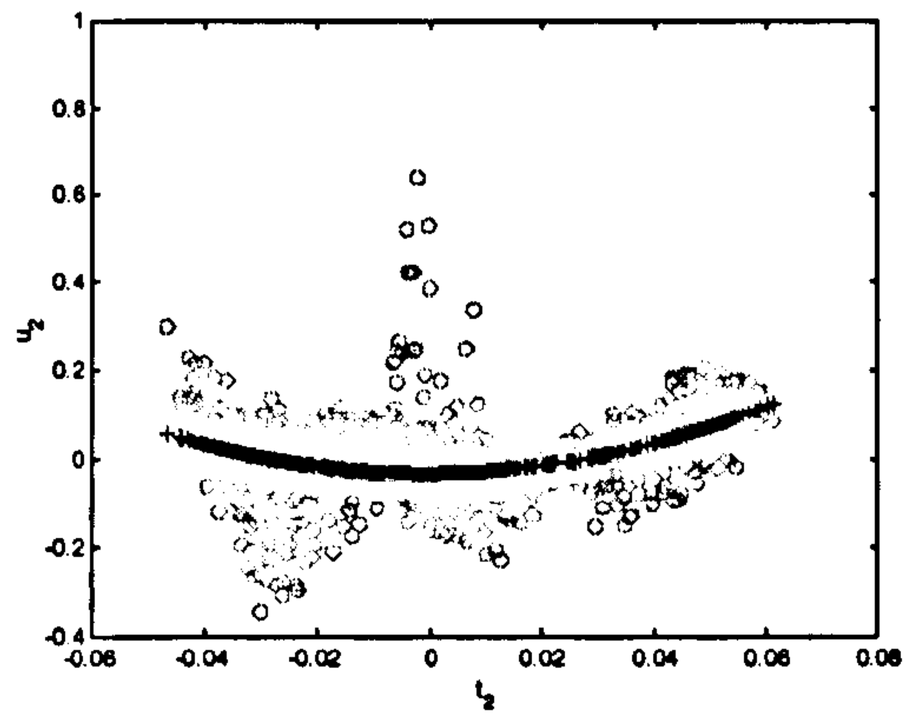

Figure 4.9b : Error Based Quadratic PLS, second latent variable scatter plot.

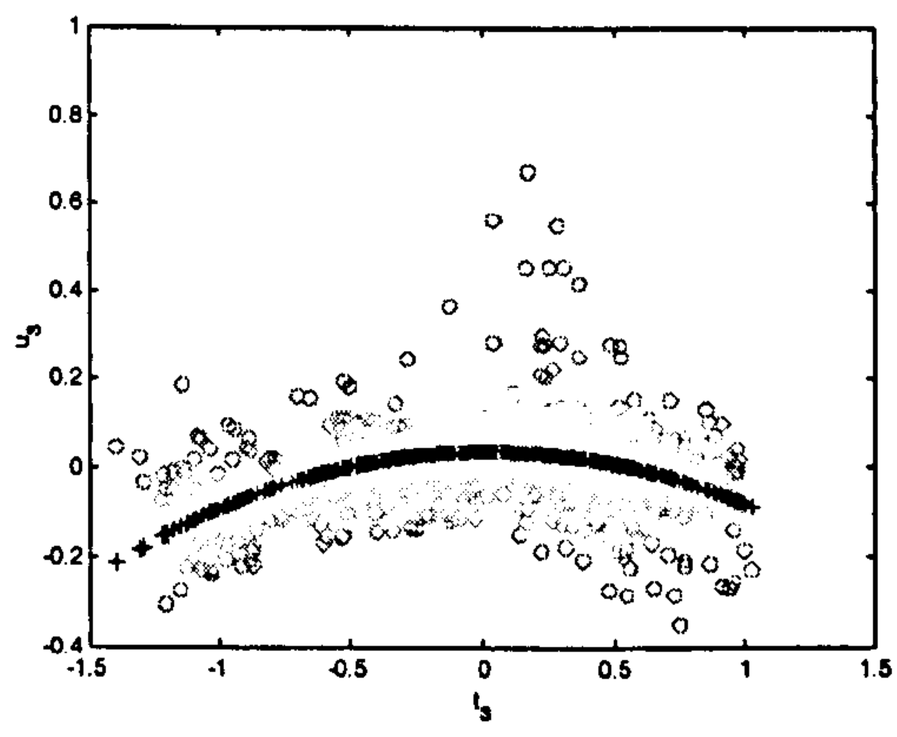

Figure 4.9c : Error Based Quadratic PLS, third latent variable scatter plot.

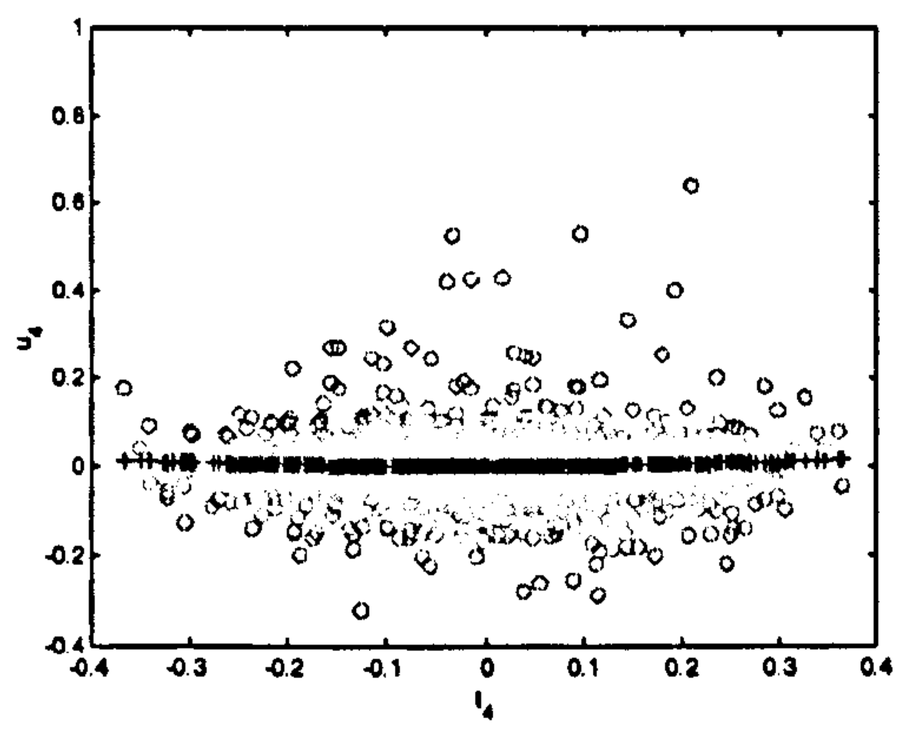

Figure 4.9d : Error Based Quadratic PLS, fourth latent variable scatter plot.

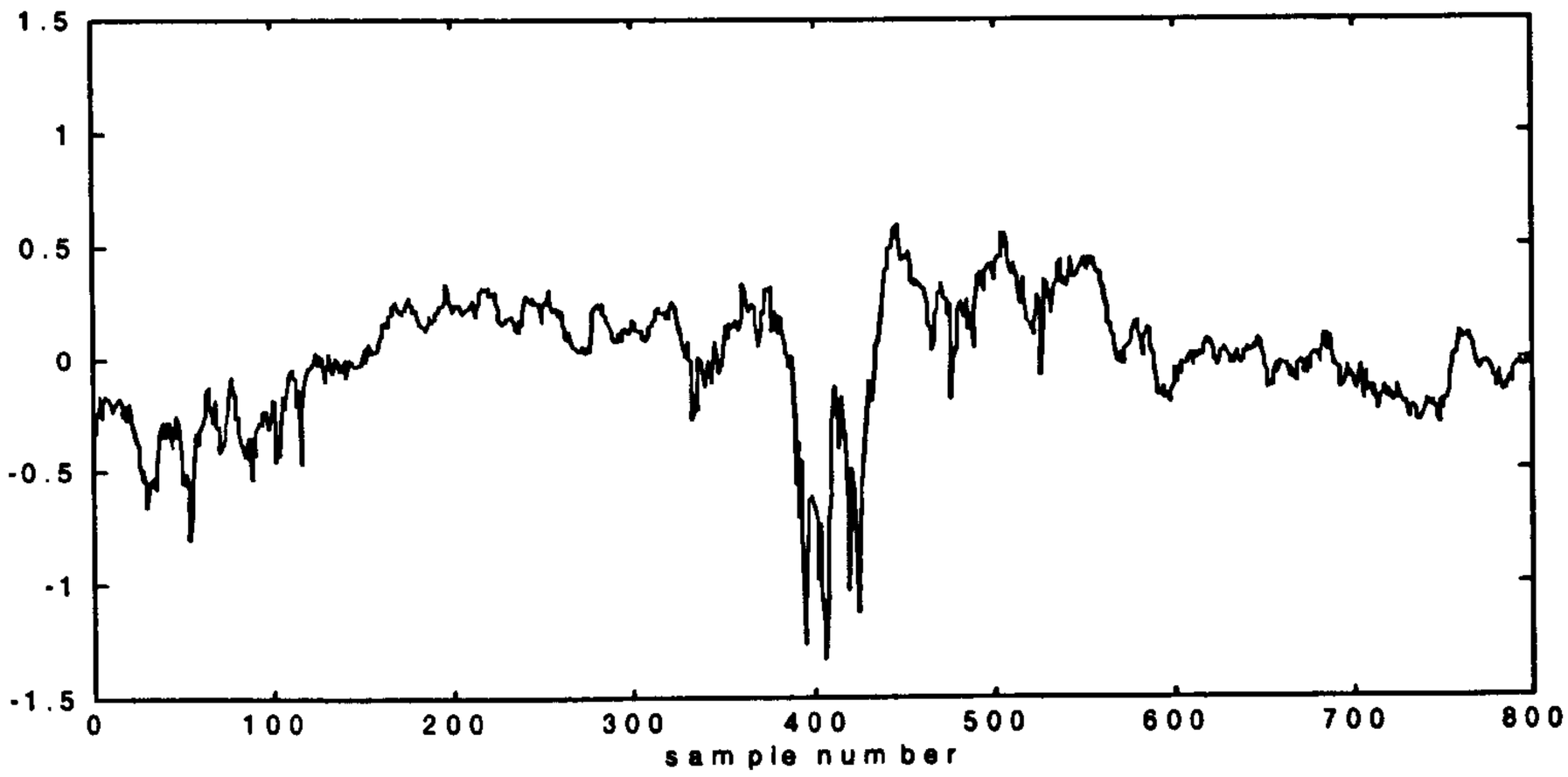

Figure 4.10a : Linear PLS, output score residuals plot for 2[nd] latent variable.

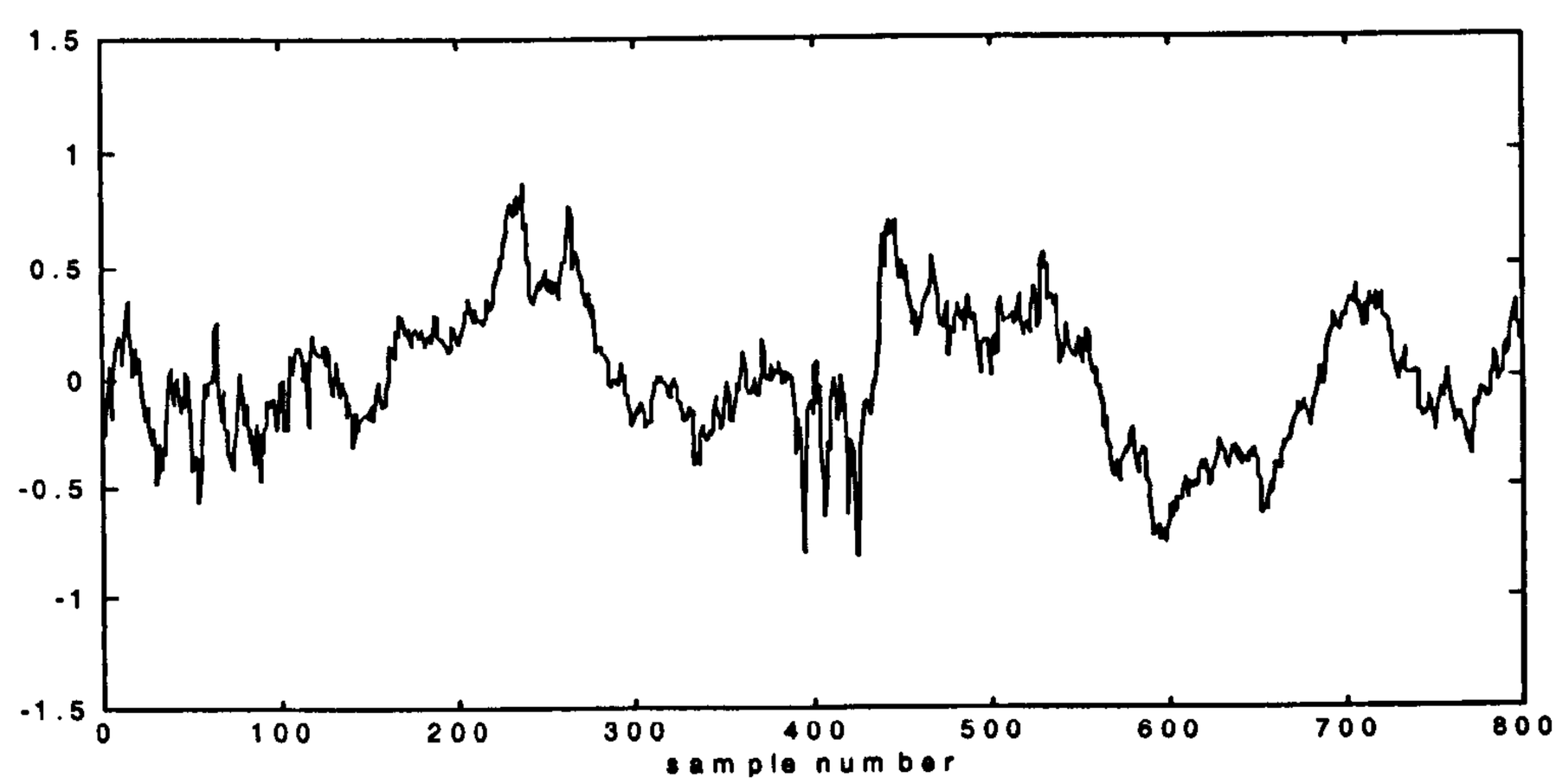

Figure 4.10b : Wold Quadratic PLS, output score residuals plot for 2[nd] latent variable.

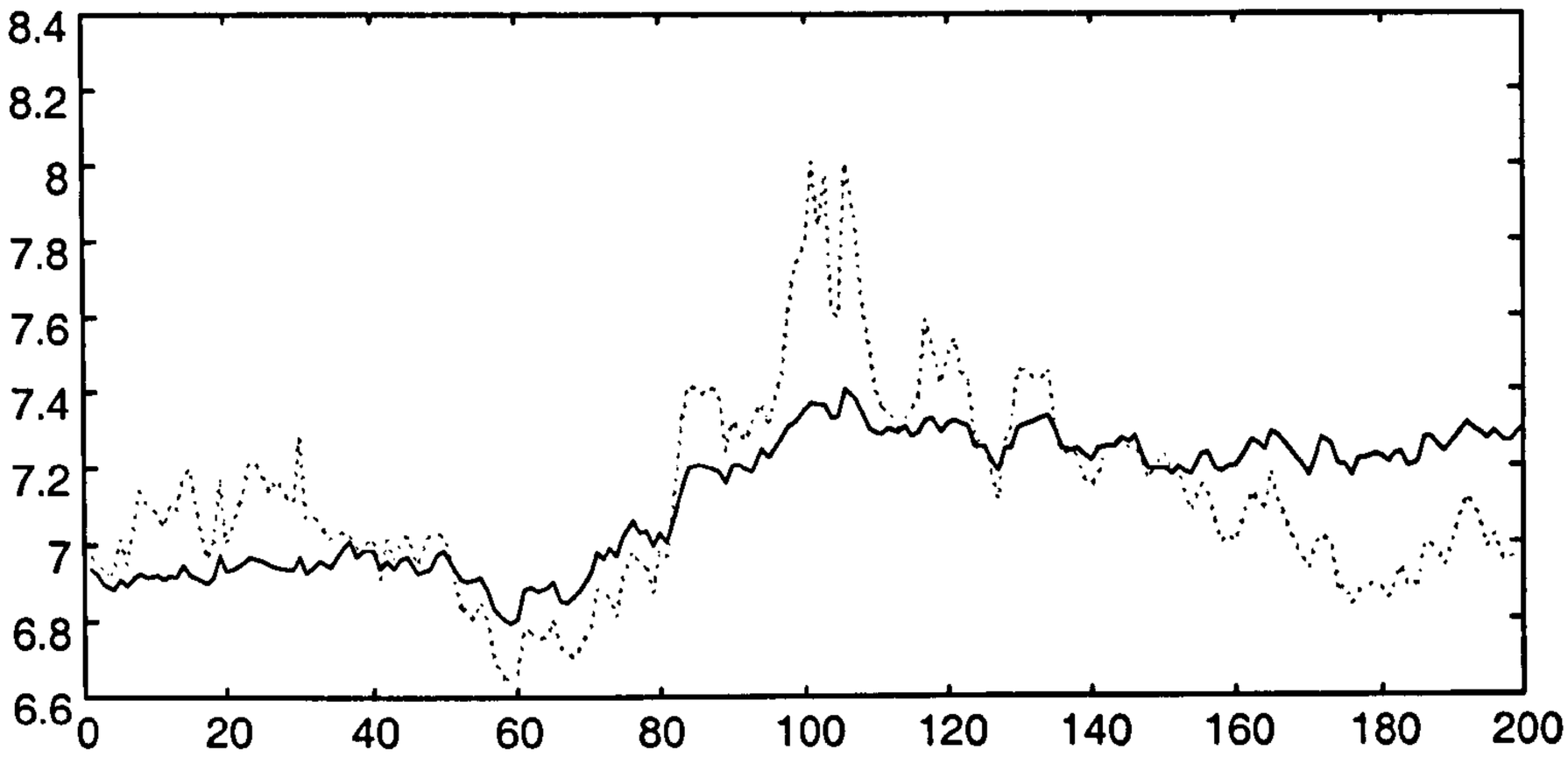

Figure 4.11a : actual (dotted light) versus predicted (full dark) output values for Linear PLS model using one latent variable.

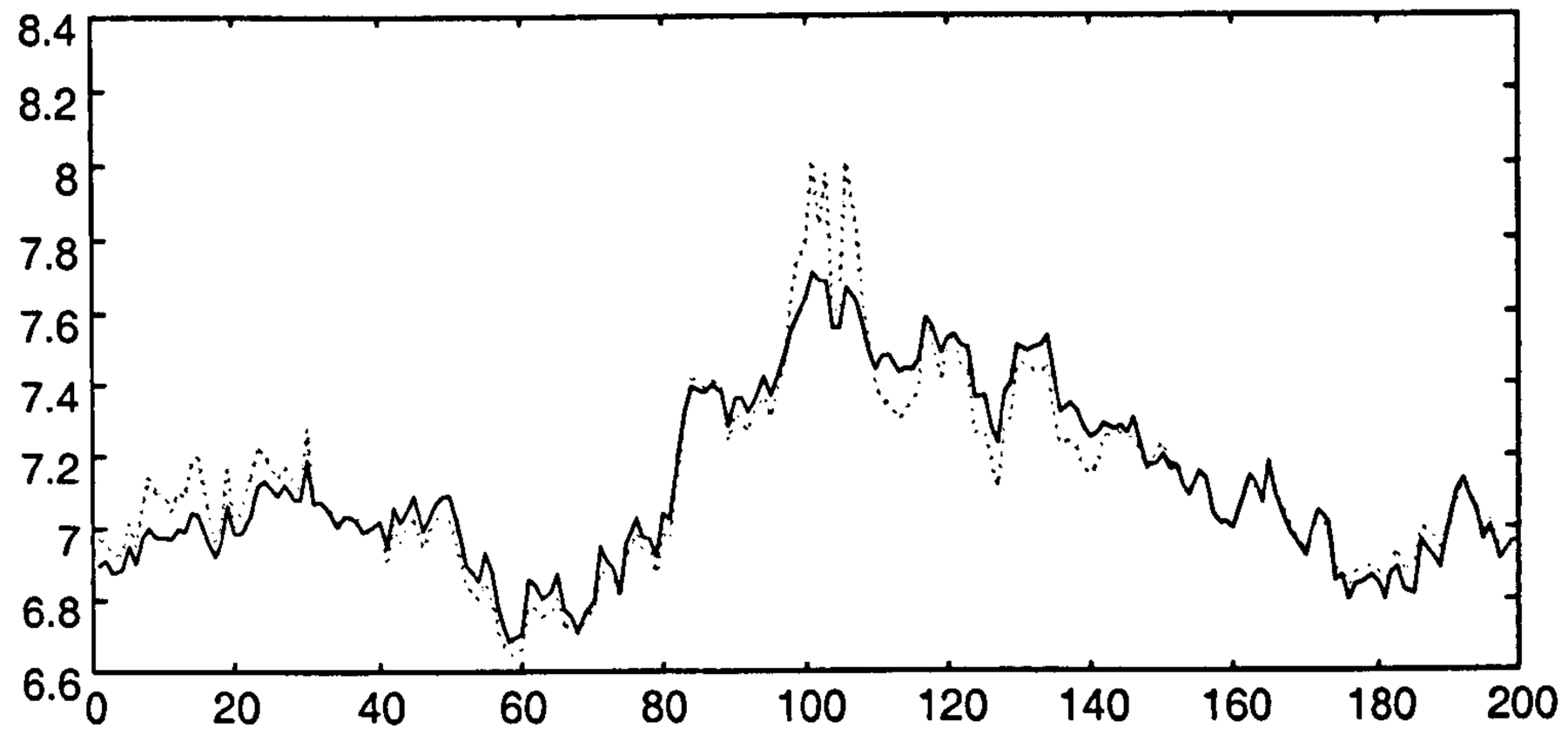

Figure 4.11b : actual (dotted light) versus predicted (full dark) output values for Linear PLS model using two latent variables.

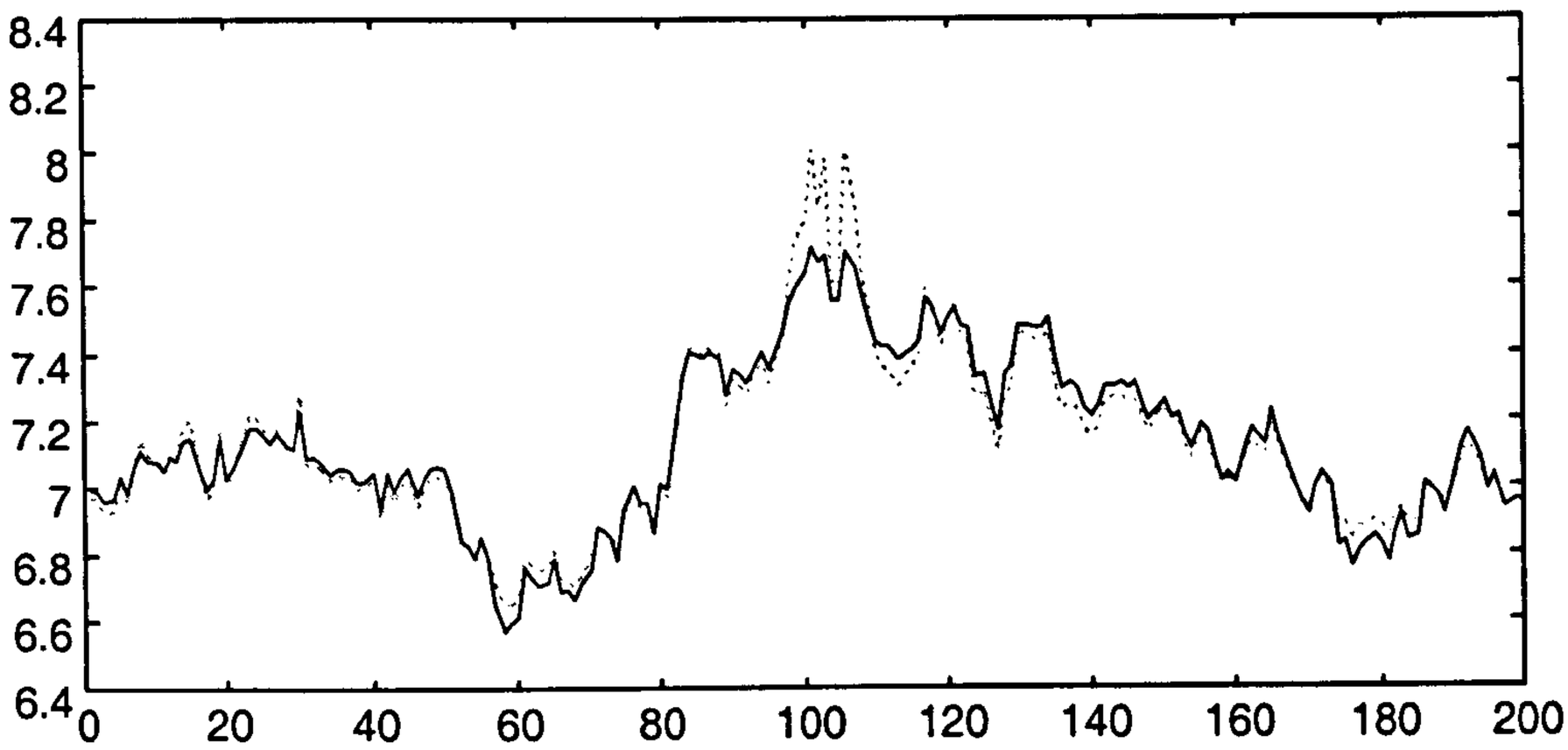

Figure 4.11c : actual (dotted light) versus predicted (full dark) output values for Linear PLS model using three latent variables.

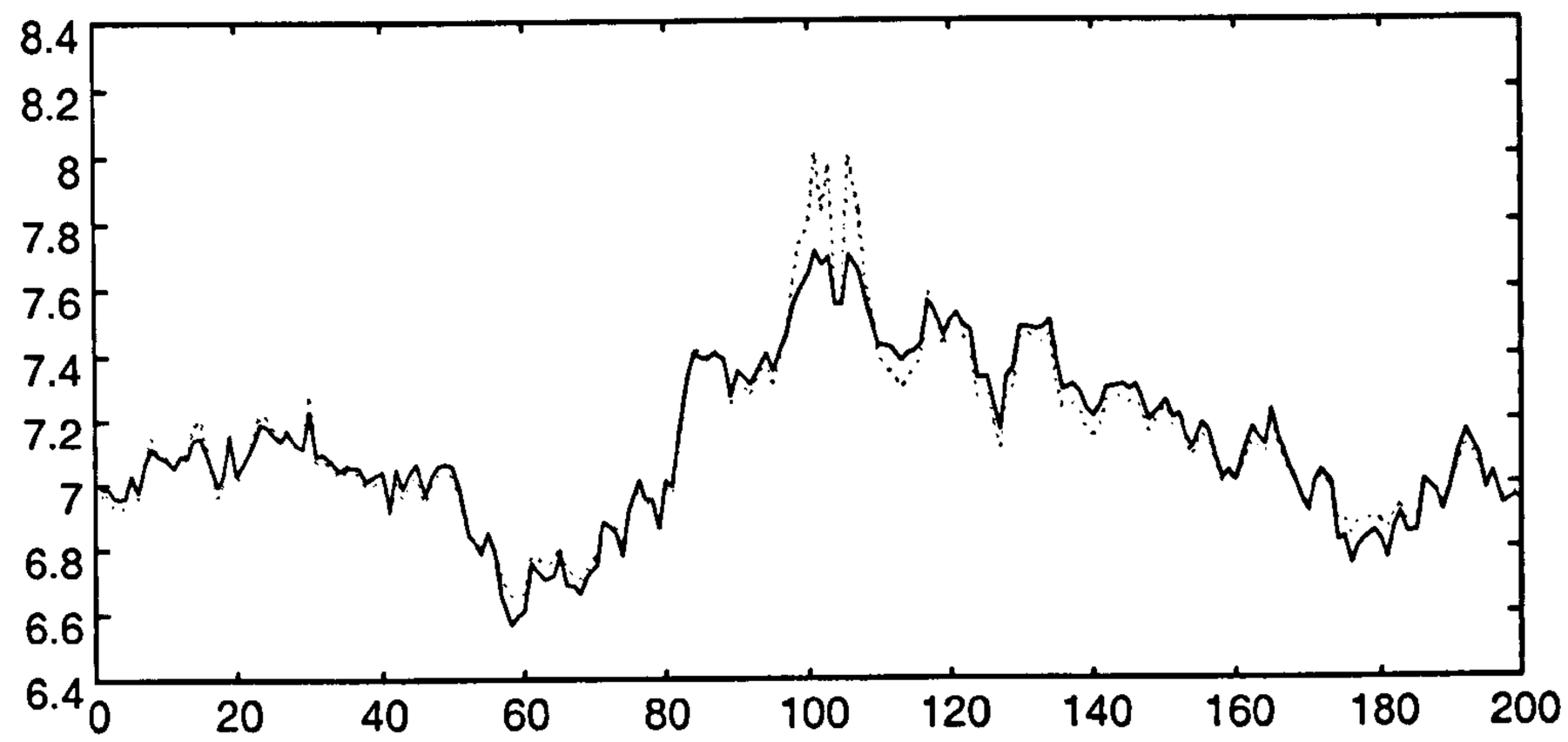

Figure 4.11d : actual (dotted light) versus predicted (full dark) output values for Linear PLS model using four latent variables.

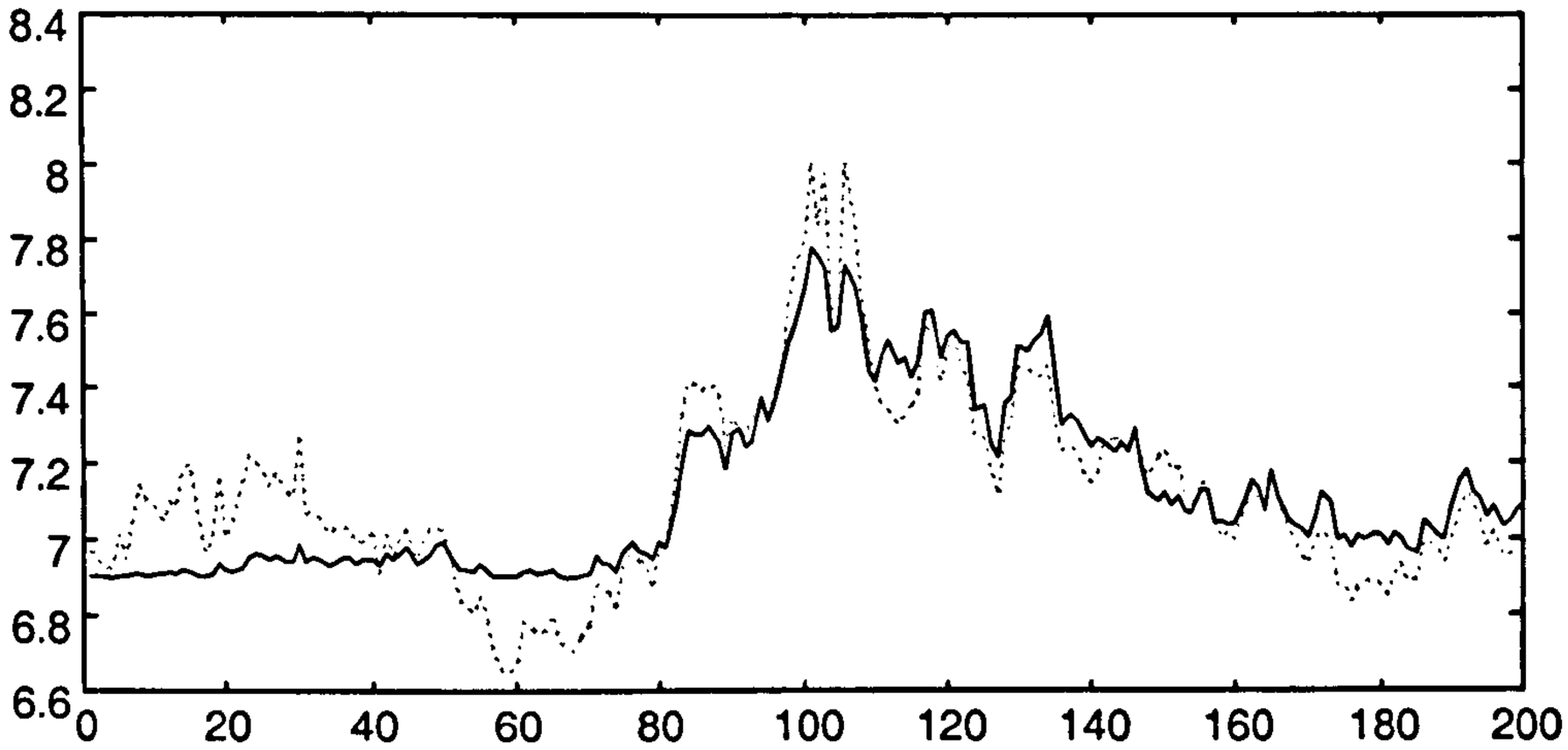

Figure 4.12a : actual (dotted light) versus predicted (full dark) output values for Wold Quadratic PLS model using one latent variable.

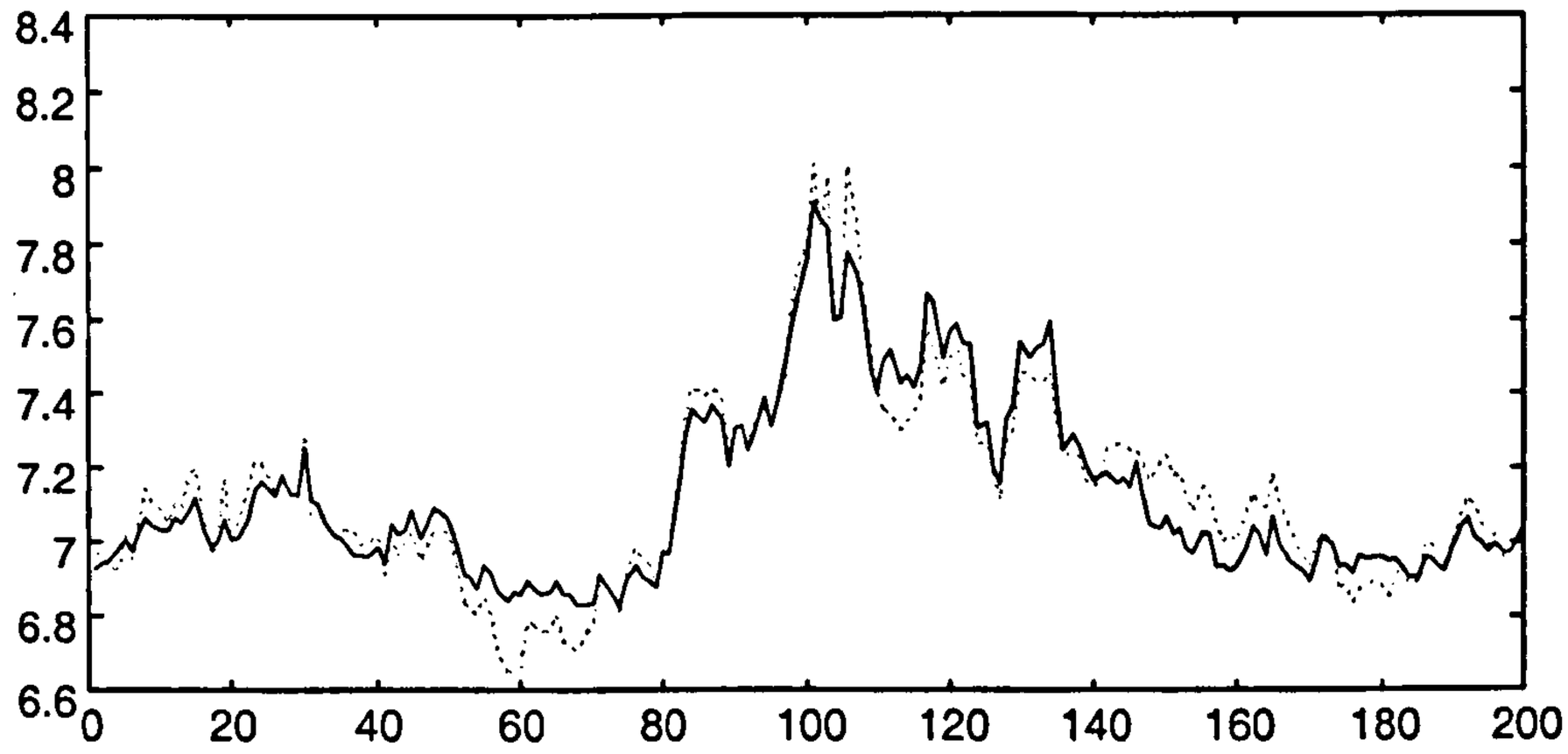

Figure 4.12b : actual (dotted light) versus predicted (full dark) output values for Wold Quadratic PLS model using two latent variables.

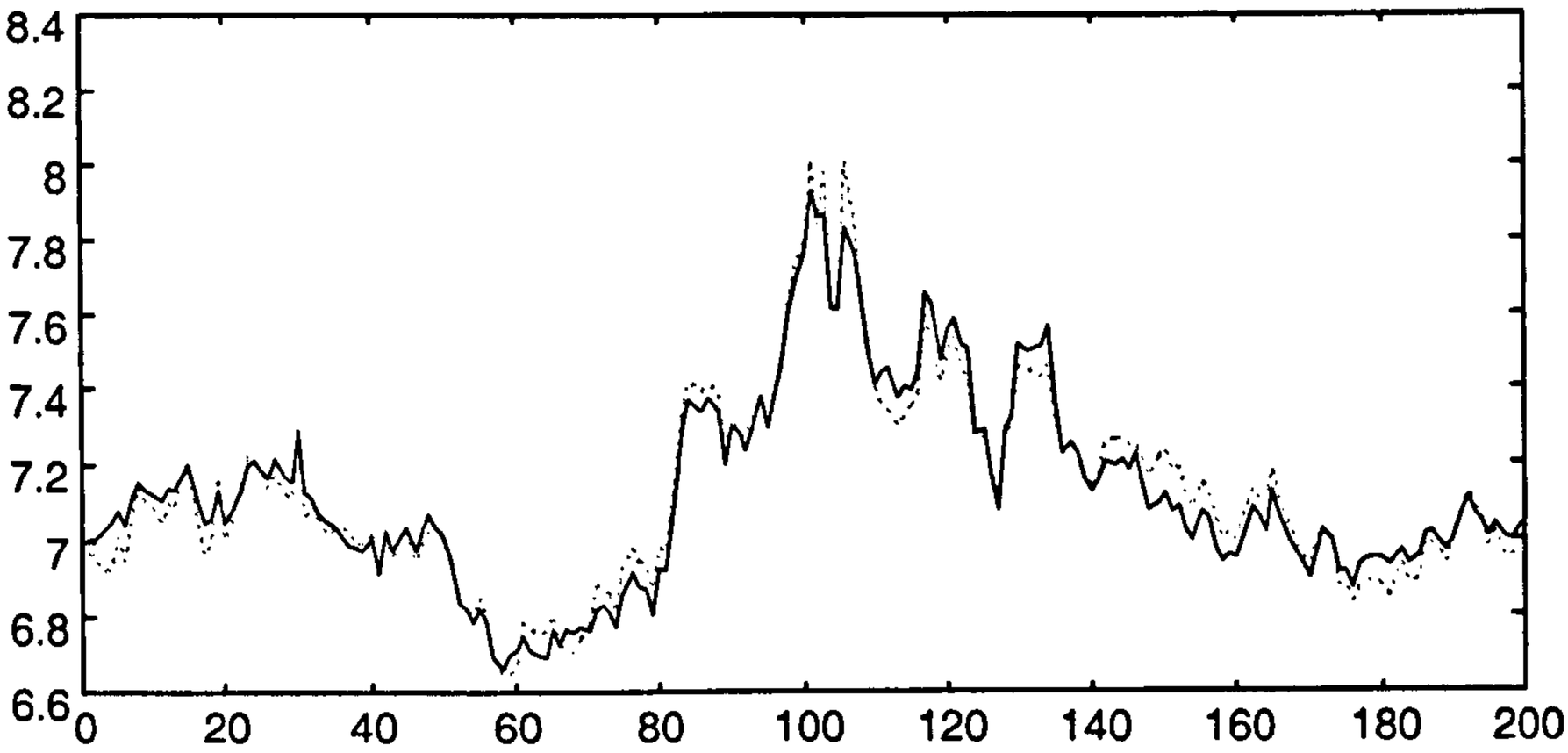

Figure 4.12c : actual (dotted light) versus predicted (full dark) output values for Wold Quadratic PLS model using three latent variables.

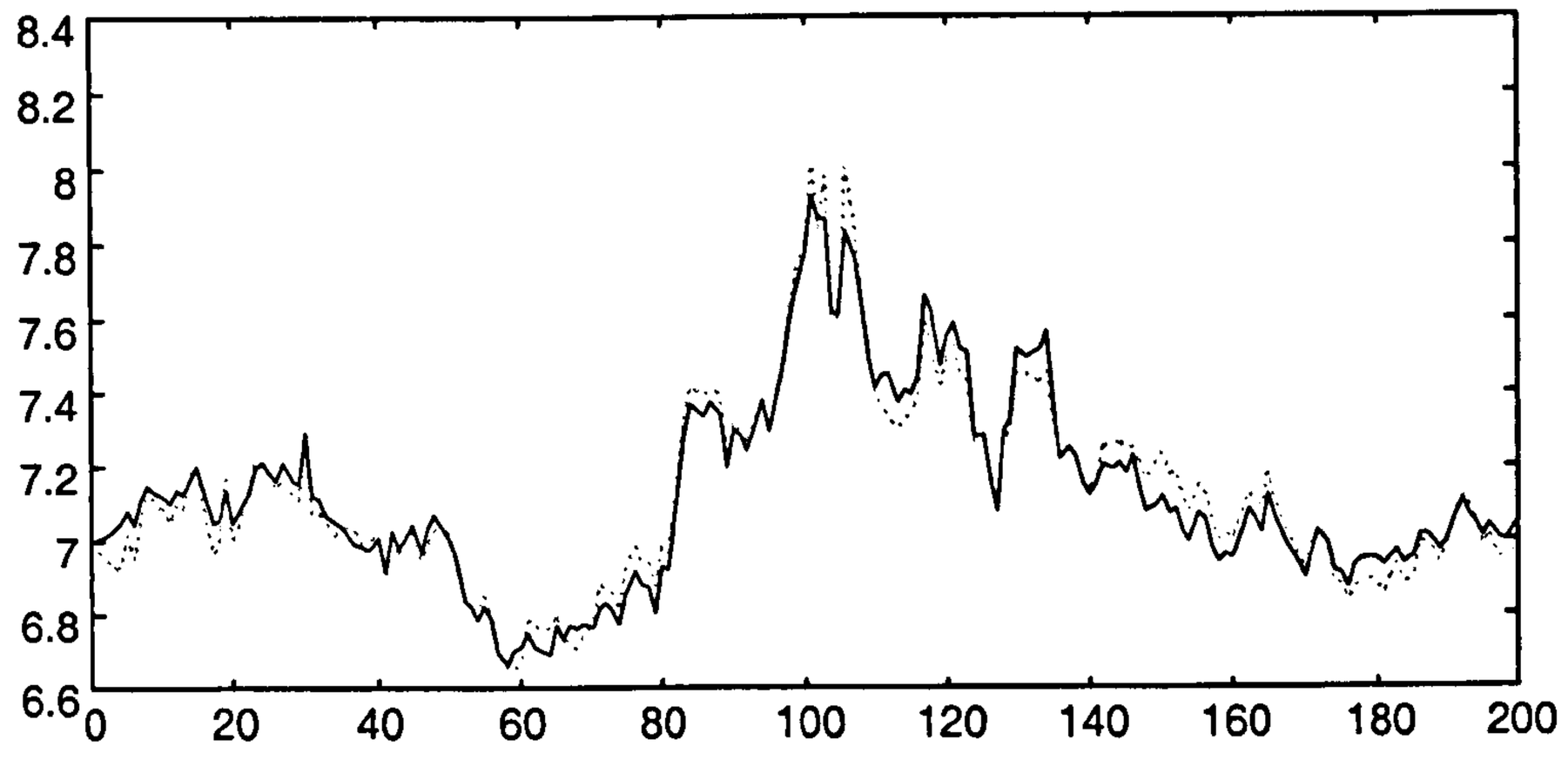

Figure 4.12d : actual (dotted light) versus predicted (full dark) output values for Wold Quadratic PLS model using four latent variables.

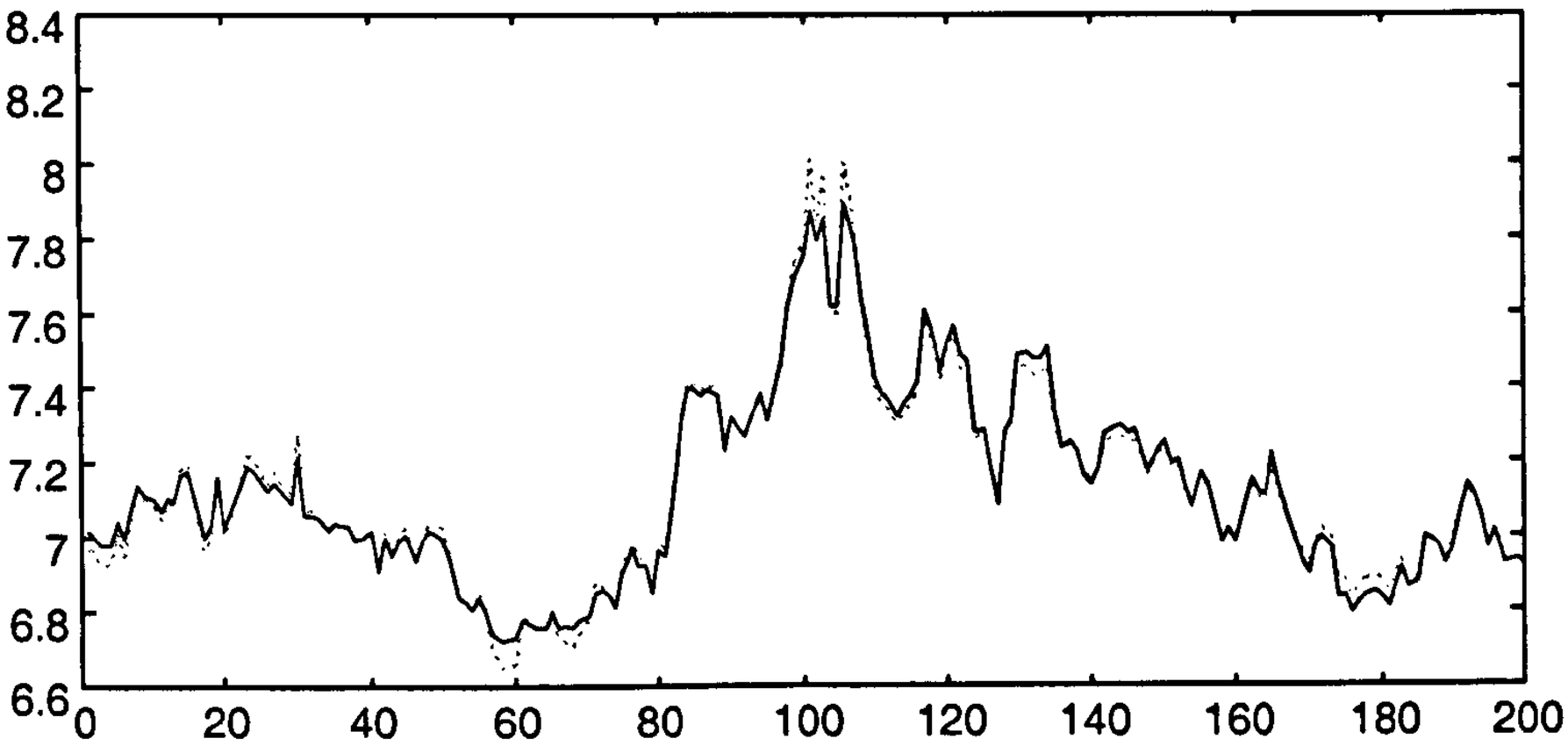

Figure 4.13a : actual (dotted light) versus predicted (full dark) output values for Error Based Quadratic PLS model using one latent variable.

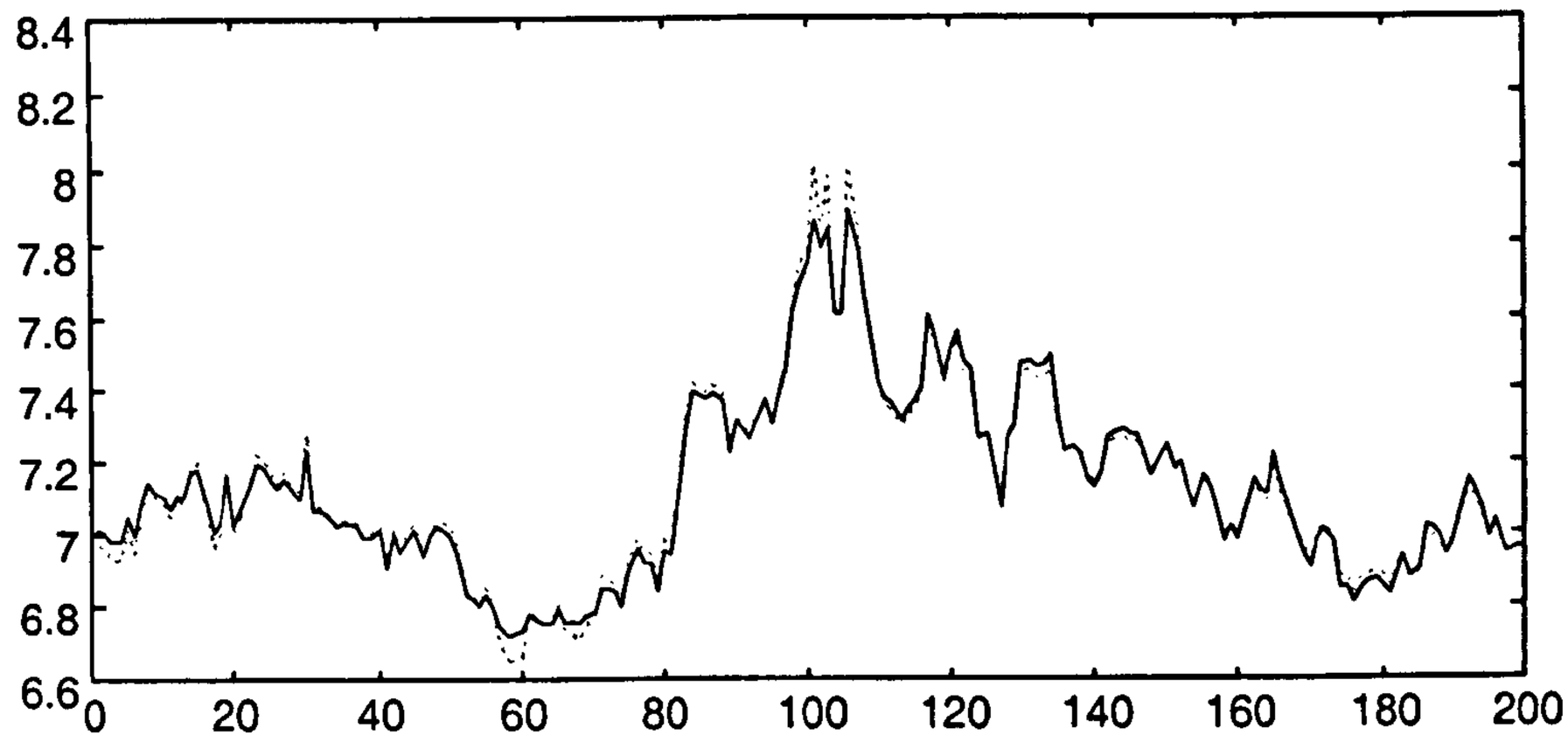

Figure 4.13b : actual (dotted light) versus predicted (full dark) output values for Error Based Quadratic PLS model using two latent variables

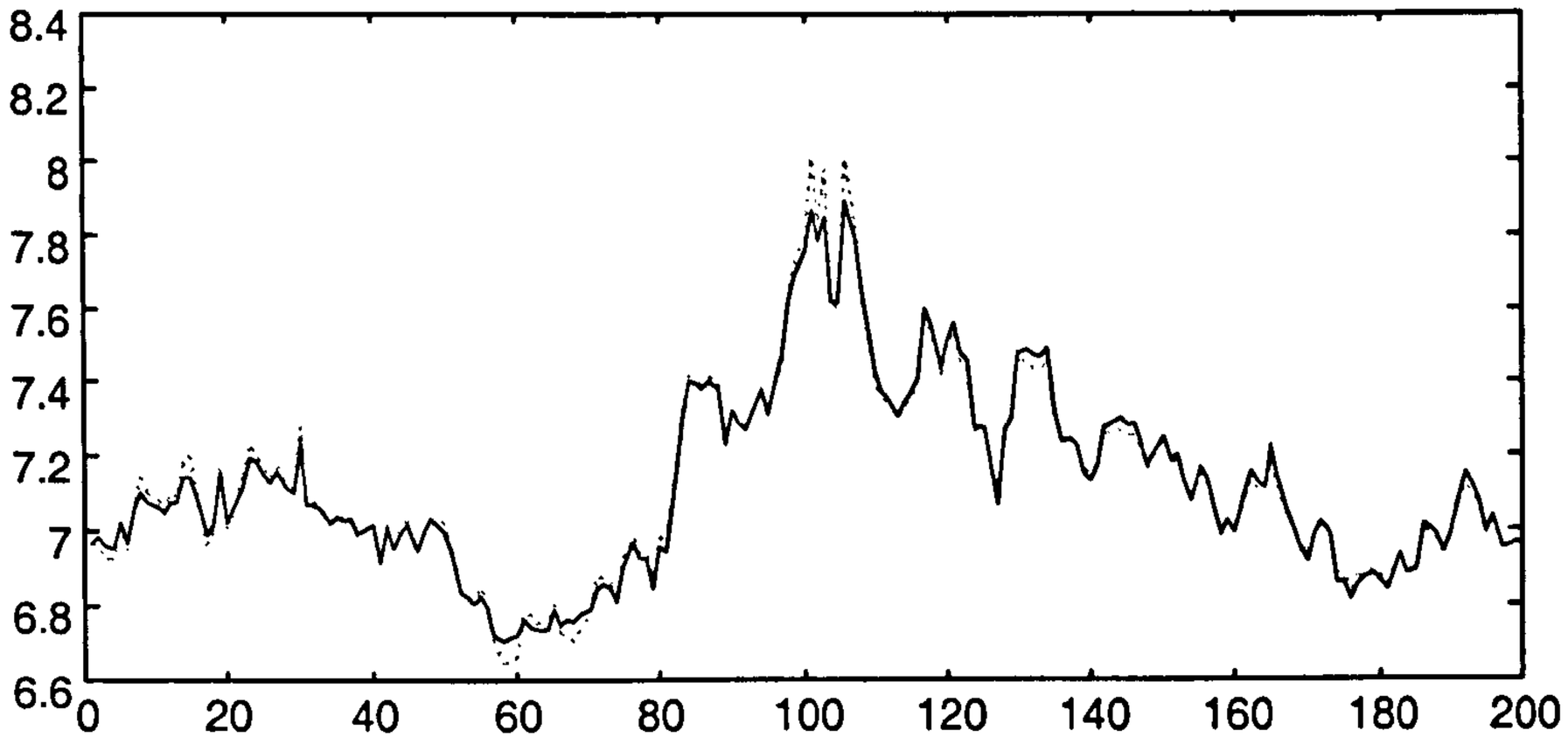

Figure 4.13c : actual (dotted light) versus predicted (full dark) output values for Error Based Quadratic PLS model using three latent variables.

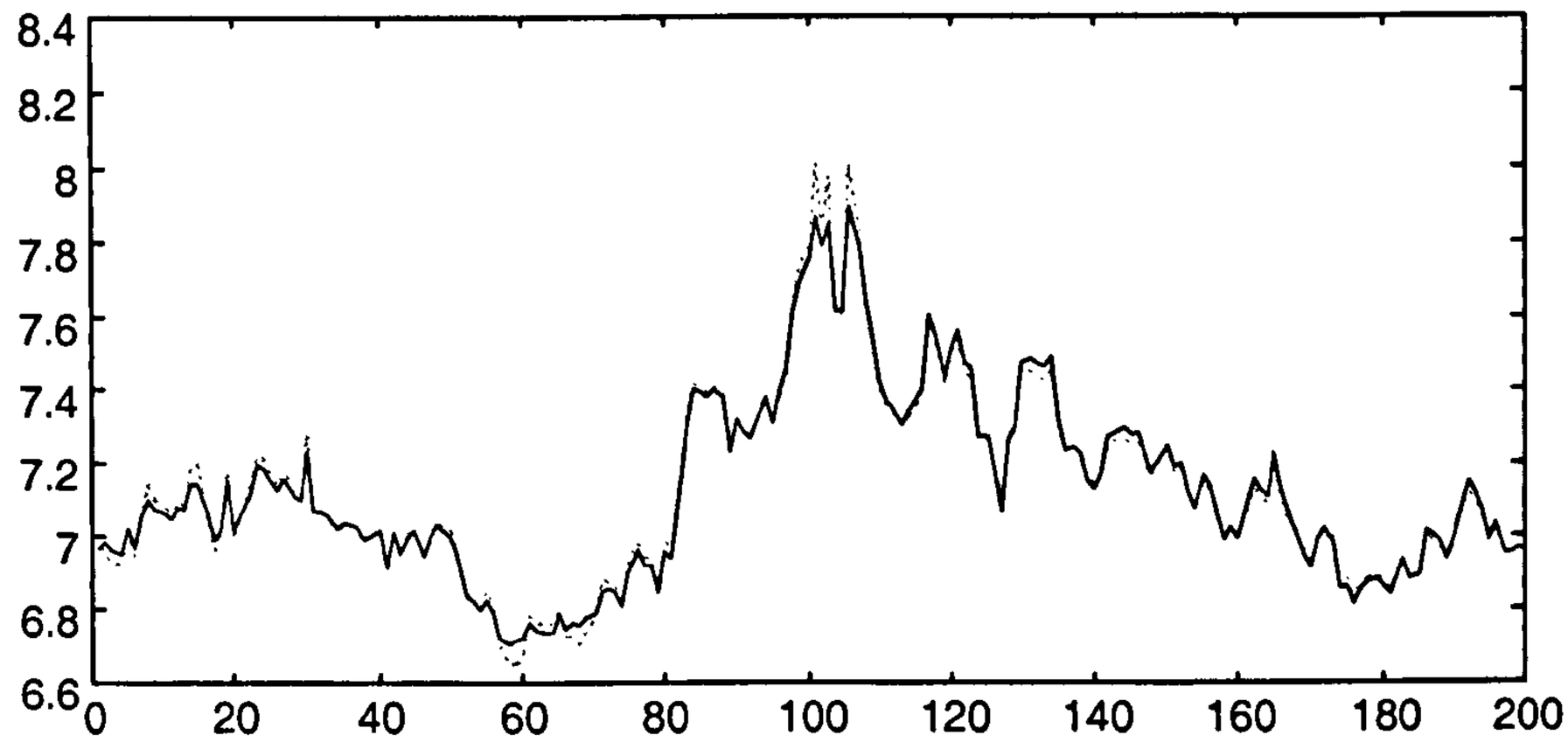

Figure 4.13d : actual (dotted light) versus predicted (full dark) output values for Error Based Quadratic PLS model using four latent variables

5. Neural Network Based Non-Linear PLS Algorithms

5.1 Introduction

Multi-Layer Perceptrons (MLPs) gained acceptance as universal approximators since they have been shown to outperform linear regression techniques when dealing with non-linear problems. Because of their general approximation capabilities (Cybenko, 1989), they have been also shown to provide better representational capabilities than polynomial or SPLINE function approaches. A number of attempts have been made to realise non-linear PLS algorithms which use sigmoidal Feed Forward Neural networks (FFN) or Radial Basis Function (RBF) networks to fit the inner mapping between the input and the output latent variables. However, some of these neural network PLS algorithms are only qualitatively comparable with PLS as a projection based regression technique. This is in contrast to those algorithms which attempt to follow the NIPALS algorithm and seek to tackle the problem of providing a general regression tool for the approximation of the non-linear mapping between the input and output latent variables in PLS. However, these algorithms lack an updating procedure for the weights of the outer input mapping. The error based approach developed in the previous chapter for updating the input weights for the quadratic PLS approach is extended to the neural network and radial basis function PLS algorithms. These showed improved performances over other published neural network PLS algorithms for the two test data sets described in the previous Chapter.

5.2 Neural Network PLS Algorithm [NNPLS]

Qin and McAvoy (1992) and Qin (1993) proposed a "*generic*" non-linear PLS algorithm by combining the universal approximation capabilities of feedforward neural networks with the robustness of PLS regression, leading to a Neural Network PLS (NNPLS) algorithm. In their algorithm, a set of centred sigmoid neural networks are used, one for each latent variable, to fit the non-linear inner regression, while retaining the outer mapping of the linear PLS algorithm, i.e. without updating the weights of the outer input mapping w. In this respect they arguably fall into the *quick and dirty* approach described by Wold *et al.* (1989) and mentioned in Section

4.2.2.2. Each neural network is a two layer feedforward network with one centred sigmoidal hidden layer and a linear output layer, and has one input, the scores t, and one output, the scores u. They demonstrated the use of the NNPLS algorithm by application to the cosmetic data of Wold *et al.* (1989) and to a chemometric data set (fluorescence spectra data for estimating bio-concentration). Better performance was attained with the neural network PLS algorithm than linear PLS. However, Qin and McAvoy (1992) did not use the same performance criteria as Wold *et al.* (1989), consequently it is not possible to compare the results obtained from the neural network PLS approach with those resulting from the application of quadratic PLS to the cosmetic data.

Qin and McAvoy (1992) stated that the use of neural networks as the inner regression model within the PLS framework makes the NNPLS regression approach generic for non-linear modelling, even with respect to the quadratic PLS approach proposed by Wold *et al.* (1989), which uses or imposes a quadratic relation as an the inner regression model. By using a sigmoid neural network, no functional relationship need be assumed *a priori* when building the inner PLS models, thus the non-linear PLS model relies on a general tool with universal approximation capabilities, and by training a sigmoid network for each component, the non-linear relationship between each pair of input-output scores can be approximated by a different network without affecting or being affected by the others. Furthermore, since one of the major advantages arising from the use of the NIPALS algorithm is that each pair of input/output latent variables is uncorrelated with respect to the others, thus training a sigmoidal network for each component ensures that the non-linear relationship between each pair of input/output scores can be approximated by a different network without affecting or being affected by the other networks.

Qin and McAvoy (1992) also showed that all the individual single-input-single-output (SISO) neural network models can be assembled with the weights and the loadings of the outer linear mapping into a single global multiple-input-multiple-output (MIMO) neural network model.

However, as pointed out earlier, their algorithm does not use any updating procedure for the outer input weights w. This is acceptable if the relationship between each pair of latent variables is slightly non-linear. However, if the inner mapping is strongly non-linear the approximation given by this approach is no longer reliable since the use of a non-linear function to relate each

pair of input/output scores affects the calculations of both the inner mapping as well as the outer mappings (Wold *et al.*, 1989).

5.3 Radial Basis Function Network PLS Algorithm [PLS/RBF]

Wilson *et al.* (1997) proposed the use of Radial Basis Function (RBF) networks to regress the mapping between the input and the output score vectors in a similar manner to that of Qin and McAvoy (1992), i.e. without any updating of the input weights $\boldsymbol{w}$. In this respect it represents nothing new in terms of algorithm development, since the only difference between the two approaches is in the kind of non-linear functional relationship used to fit the inner mapping. However, there are a number of advantages that arise from the use of RBF networks to fit the inner mapping in contrast to the use of sigmoidal feed forward networks. Even though the approximation properties of RBF networks are comparable with those of sigmoidal networks, the training algorithms for RBF networks are considerably simpler and faster than those for sigmoidal networks. Wilson *et al.* (1997) applied the PLS/RBF algorithm for fault detection in a benchmark problem describing an industrial overheads condenser and reflux drum plant configuration and showed that it performed better than linear PLS. No comparisons were given with respect to the NNPLS algorithm of Qin and McAvoy (1992).

The PLS/RBF algorithm employs a set of RBF networks with Gaussian activation function, one for each pair of latent variables $\boldsymbol{t}$ and $\boldsymbol{u}$, to fit the non-linear inner mapping. This structure was shown to give acceptable results. However, they did not completely exploit the advantages arising from the use of RBF networks instead of sigmoidal feedforward networks. They used a modified Broyden-Fletcher-Golfarb-Shanno (BFGS) algorithm to train the RBF network. In particular they used a BFGS algorithm to determine the locations of the centres and the distance scaling parameters for the Gaussian activation function, and least squares regression to fix the RBF output weights. Furthermore they did not use any updating procedure for the PLS outer input weights $\boldsymbol{w}$.

5.4 Alternative Neural Network PLS Algorithms

A different approach to neural network PLS modelling arises from the properties of *auto-associative* neural network (ANN; Kramer, 1992). These are four-layer networks consisting of two non-linear *feature layers* (layers 1 and 3), a linear *bottle-neck layer* (layer 2) and a linear *output layer* (layer 4). The input and the output data of auto-associative networks are the same. The two features layers have the same number of units as the number of input/output variables, and provide a non-linear mapping between the outer space (input variables) and the feature space (non-linear transformation of the input variables). The bottle-neck layer is placed in between the first and the second feature layers and consists of a smaller number of units than the feature layers, it forces (project) the output from the first feature layer down onto a lower-dimensional space. In this way auto-associative networks provide a useful tool for projecting information from high dimensional data space down onto lower dimensional non-linear space represented by the output of the bottle-neck layer, and hence can be used to develop non-linear PLS algorithms. This can be achieved by replacing the input data matrix with the output data matrix on the output layer of the auto-associative network, and modifying the structure of the output layer of the network.

Non-linear PLS algorithms based on the use of ANNs might still be classified as projection based regression techniques, even though the projection feature is no longer linear. This is due to the projection feature provided by the activation functions of the network nodes (typically sigmoid, centred sigmoid or gaussian functions). In the case of the auto-associative networks, the predictor and the response latent variables can be identified with the inputs to, and the outputs from, the activation functions of the bottle-neck layer respectively. In addition, in a similar way to PLS, the relationship between the predictor and the response variables can be modelled by means of a reduced number of activation functions. However, from a statistical point of view these algorithms cannot be classified as multivariate statistical regression techniques. Furthermore they are only qualitatively comparable with the family of linear and non-linear PLS algorithms based on the use of the NIPALS algorithm. In practice they lead to PLS models which have a global neural network structure and which are trained using neural network training algorithms, i.e. non-linear optimisation routines.

5.4.1 Integrated Neural Network PLS Algorithm [INNPLS]

Saunders (1992) proposed a fully "Integrated Neural Network PLS" algorithm (INNPLS), which uses a sigmoidal neural network to fit the inner mapping between each pair of latent variables. This partially follows the work of Qin and McAvoy (1992), but modifications were made to the NIPALS algorithm in order to perform updating of the input weights. However the weights updating procedure proposed by Saunders (1992) differs from that proposed by Wold *et al.* (1989) in that the INNPLS algorithm treats the outer input mapping [$X \rightarrow t$] as a linear extension of the inner network model [$t \rightarrow \hat{u}$].

The extended network consists of a standard inner sigmoidal network with an additional input layer of linear units that performs a weighted summation of the input data (without associated bias terms), and acts as the input outer mapping. In the new algorithm, the extended network is trained by alternately freezing the weights of the two layers. Saunders (1992) claimed that freezing the weights of the first layer (input outer mapping) and modifying the weights of the inner sigmoid network improves the fit of the non-linear inner relation, whilst freezing the weights of the inner sigmoid network and modifying the weights of the linear input layer of the extended network improves the fit of the input outer mapping to the non-linear inner regression. Thus the input weights updating procedure is carried out by training the extension of the extended network while keeping constant the weights of the inner sigmoidal network. In practice the INNPLS algorithm reduces to an optimisation procedure for training multi-layer neural networks and in this respect cannot be regarded as being a sound statistical approach to non-linear modelling, since it uses the traditional PLS algorithm only as the first step of the overall optimisation. Consequently INNPLS can only be qualitatively compared to the family of PLS algorithms based upon the NIPALS algorithm. Nevertheless, in terms of projection techniques the approach of Saunders (1992) still leads to a linear projection for the input outer mapping [$X \rightarrow t$], a non-linear projection for the inner mapping [$t \rightarrow \hat{u}$], and a linear mapping for the output outer mapping [$\hat{U} \rightarrow \hat{Y}$]. Furthermore the procedure proposed by Saunders acts in an iterative manner and sequentially extracts one latent variable at a time, hence "*peeling*" the input and the output data matrices of relevant information.

Saunders used the cosmetic data, in addition to other data sets, to test the INNPLS algorithm and a better result was obtained with the INNPLS algorithm than when linear PLS was applied. Although no comparison with the quadratic PLS approach was made, it would appear that the

results of Wold *et al.* (1989) using quadratic PLS were better than those obtained with the INNPLS algorithm.

5.4.2 PLS FeedForward Network Algorithm [PLS/neural]

Holcomb and Morari (1992) proposed a neural network implementation of the PLS algorithm based on the use of a feedforward neural network (PLS/neural), with both linear and centred sigmoid neurone activation functions. They started by considering the linear PLS method as a two layer neural network consisting of a linear *feature layer* and a linear *output layer*. The feature layer is a hidden layer which performs a linear combination of the input data to provide an orthonormal linear mapping from the input space to the feature space, and hence act as a projection device within the network structure in a similar manner to the projection matrix $\boldsymbol{R}$ in the linear PLS algorithm (Equation 3.37). The output layer then performs a linear combination of the output signal arising from the feature layer. The PLS model is subsequently built using an optimisation routine (e.g. backpropagation). Starting from this assumption, they defined a new PLS approach which is fully implemented as a multi-layer feedforward network (FFN). In particular they defined the PLS/neural network structure as a three-layer network consisting of a linear feature layer and a two-layer feedforward network with mixed linear and centred sigmoidal activation functions for the hidden units, together with a linear activation function for the output layer. To train the three-layer network they proposed a hybrid approach based on the use of PCA to select the number of neurones of the feature layer (i.e. to select the number of latent variables to retain in the model). The weights of the feature layer are also initiated using PCA, using the directions of the principal components to initialise the directions of the latent variables. Finally the back-propagation algorithm is used to train the two-layer feedforward neural network and back-propagation again to train the overall three-layer network. The whole approach makes use of different learning parameters for the weights in different layers.

Holcomb and Morari (1992) compared their approach with MLR, linear PLS and the QPLS approach proposed by Wold *et al.* (1989). On the basis of the performance of these algorithms on two sets of experimental data and a tutorial example, they showed that their PLS/neural algorithm out-performed the other three algorithms. However, a major drawback of the PLS/neural algorithm is that the structure of the neural network must be defined in advance. Holcomb and Morari (1992) suggested using PCA to determine the true dimension of the input data set and hence the number of neurones to use in the feature layer. But using PCA to identify

the true dimension of the system can lead to poor performance, since PCA identifies the number of PCs which are required to explain most of the variability of the input data set, and not the latent variables which are required to model the correlation structure between the input and the output data sets. Furthermore, the orthogonality of the non-linear latent variables in the PLS/neural approach is achieved by carrying out a Gram-Schmidt orthogonalization on the feature vectors, and if the number of vectors is large the overall procedure may become time demanding.

The PLS/neural approach does not differ markedly from the INNPLS algorithm proposed by Saunders (1992). In fact the end result of both algorithms is a three-layer neural network model with a first linear hidden layer which performs a linear projection of the input data onto the feature space of the input scores, and a two-layer network to produce the non-linear mapping between the input and the output scores. However, the algorithm proposed by Saunders (1992) is still iterative and extracts one latent variable at a time, whilst in Holcomb and Morari (1992) the structure of the PLS/neural model must be defined in advance.

5.4.3 Auto-associative Networks and the Non-Linear PLS Algorithm [NLPLS]

Auto-associative neural networks with a one-dimensional bottle-neck layer represent the main feature of the non-linear PLS (NLPLS) algorithm proposed by Malthouse *et al.* (1997), which is fully implemented with a neural network. It reflects the properties of the linear PLS algorithms as a projection-based regression technique but with non-linear features that provide both a non-linear PLS model (NLPLS) and a non-linear PCA (NLPCA) representation within the same network. The overall structure consists of an auto-associative network which performs a non-linear compression of the input variables, and the second half of another auto-associative network which performs the decompression of the output of the bottle-neck layer of the full auto-associative network onto the output variables. The second half-auto-associative network is linked to the first full-auto-associative network by means of another mixed linear and non-linear hidden layer. This provides the mapping between the output from the input-variables feature layers to the input to the output-variables feature layer. The overall network structure is trained by means of non-linear optimisation algorithms.

They applied their NLPLS algorithm to the study of a number of tutorial examples and also to a process control application (fabrication of composite materials), comparing several

configurations of their NLPLS and NLPCA models with PCR, PLS, a neural network and projection pursuit regression. They pointed out that although their NLPLS algorithm gave better performances than the other algorithms, it was sensitive to noise and to starting values (being an optimisation-based algorithm) and that attention needed to be paid to the training of the network since it had a tendency to overfit the training data. This is always a danger with neural network, and hence with neural network based PLS algorithms.

5.5 Direct Network Approach and NNPLS

Multi-layer feed-forward networks (FFN) or multi-layer perceptron networks (MLP) are known to be good non-linear function approximators, and have been applied to a wide variety of situations related to chemical process modelling and control. In particular it has been shown that neural networks with sufficient hidden layers and neurones can approximate any continuous function with arbitrary desired accuracy (Hornik *et al.*, 1989; Cybenko, 1989). Furthermore, since feedforward network models can be built using gradient type learning rules like back-propagation and conjugate gradient training algorithms, it would seem (at least from a superficial view) that training a neural network cannot be affected by collinearities between the variables of the training data set, which is the major limitation of least squares regression approaches, where collinearities lead to an ill-conditioned problem due to the required inversion of the regressor matrix. This, however, is not true since neural networks can suffer, in a different way, from the same problem, and can fail to provide a robust solution. Qin (1993) pointed out a number of problems related to the robustness of models based on neural networks trained on collinear data. In particular he observed that when the input variables are correlated there are many combinations of the network weights which can minimise the training error giving almost the same output values. The end result of training neural networks on collinear input data is that when predicting new output values on the basis of new input data corrupted with noise, the network models often give large variance in the prediction. Thus, even if neural networks are not directly affected by ill-conditioning problems, in the presence of collinearities, neural networks models tend to enlarge the noise variance in the predictions.

With respect to this issue, it is stressed that collinear data are not infrequent in databases collected on chemical processes, mainly because of the large number of variables generally

monitored on each process unit, and these all describe the same underlying process. In addition the relationship imposed on variables by closed loop control systems, which often cannot be opened when collecting the data, also results in correlated data.

Qin (1993) observed that neural network models trained on collinear data are only valid for new data where the correlation still holds. Thus when passing to the network model new input data which does not have the same correlation structure as the training data set, the model is no longer valid and needs to be updated. Furthermore, Qin (1993) pointed out that there are other limitations affecting the training of a neural network model on process data, especially when dealing with multivariate systems. In particular in the case of a limited number of available samples, the number of weights of the neural network might be larger than the number of observations and hence some of the weights cannot be uniquely determined from the observed data. This leads to an over-parameterized model which might overfit the data, resulting in poor, if not negligible, generalisation capabilities. The same problem arises with complex input-output relationships, when a considerable number of hidden layers and nodes are required to model the system. In this case too many hidden units again result in an over-parameterized model, whilst too few hidden units leads to an under-parameterized model.

A common approach to reducing the effect of collinearities in the training data set, and the related increase of the output variance, is the introduction of penalty factors in the error function used for training the neural network. This leads to an application of Ridge Regression (Irwin *et al.*, 1995). But introducing penalty factors brings new problems, since they can introduce bias into the prediction (alongside reducing the variance). In this case right (or optimal) values of the penalty factors must be chosen to optimise the reduction in the variance and the increase of the bias in the network prediction.

In the same work Qin (1993) proposed a number of solutions to overcome the limitations mentioned before, in particular the effect of the collinearities present in the input data. First of all, an easy way to remove the collinearities would be the use of PCA: finding the minimum number of principal components to represent the input data set. The neural network is then trained to model the output variables from the PCs which are known to be uncorrelated. A major problem related with this approach is that the underlying criterion in PCA is the location of the directions of greatest variability of the data set on which the analysis is being performed, without considering the correlations that exist between the input and the output variables. However,

some variables might exhibit large variability in the input data set whilst only being weakly correlated with the output variables. A consequence of this would be that those variables which are more highly correlated with the output variables might be masked, that is their loadings on the first PCs might be small, resulting in them being relegated to the lower order PCs, which might not be included in the kernel of PCs required to represent the input data set. This would lead to poor performance of the neural network model, since the underlying linkage existing between the input and the output variables would be lost from the outset. To overcome this problem Qin (1993) proposed the use of PLS to remove the collinearities affecting the input data set since, as commentated previously, it has as a maximising criterion the covariance structure between the input and the output variables. The use of PLS as a pre-processor leads to the Neural Network PLS (NNPLS) approach proposed by Qin and McAvoy (1992) and by Qin (1993), where the data are not directly used to train the network, but are pre-processed by the PLS outer transformation.

Using neural networks within the PLS framework leads to a number of other advantages with respect to the direct neural network approach. When reducing the multiple-input-multiple-output (MIMO) network regression problem to a number of simpler single-input-single-output (SISO) network regression problems, the number of weights to be determined in the NNPLS model is much smaller than the number of weights to be determined in the corresponding direct neural network approach. By reducing the number of weights, the over-parameterization problem is avoided. Furthermore, the number of local minima is expected to be reduced since the structure of each inner network model is much simpler than the structure of the network required in the direct neural network approach. Last, but not least, the NNPLS method is less sensitive than the direct neural network approach to situations in which the number of training samples is small. In this situation the NNPLS approach benefits from the PLS framework for the outer mappings.

Thus the overall effect of using a NNPLS approach instead of the traditional direct neural network modelling approach, is increased robustness of the model and smaller prediction variance, as a result of the reduced sensitivity to collinearities in the input data set provided by the PLS framework.

5.6 Modified Neural Network and RBF PLS Algorithms

The major limitation arising from both the previously published works on the centred sigmoidal network and the RBF network PLS algorithms is that the weights of the PLS outer input mapping are not updated. Wold *et al.* (1989) pointed out how this approach is acceptable when the mapping between each pair of latent variable is slightly non-linear. Wold *et al.* also underlined the necessity of an updating procedure for the input weights, to match the linear outer input mapping with the non-linear inner mapping. The non-linear PLS algorithms proposed here are founded upon the *error based weights updating procedure* described in Sections 4.4 and 4.5 in conjunction with the neural network PLS algorithm proposed by Qin and McAvoy (1992) and the PLS/RBF algorithm proposed by Wilson *et al.* (1997).

The input weights updating can be achieved since the non-linear relationships provided by the centred-sigmoidal neural networks and by the RBF neural networks are differentiable with respect to the weights of the outer input mapping, and a Taylor series expansion of the two neural network models can be used within the error based updating procedure. In this way the new neural network PLS algorithms are thought to provide a more generic approach to non-linear modelling. This arises from the use of two theoretically valid universal approximators (sigmoidal and RBF networks) and the use of the error based weights updating procedure which leads to a more coherent non-linear PLS framework with respect to the previous ones.

5.6.1 The Taylor Series Expansion of the Centred Sigmoidal Neural Network

Here the neural network is a two-layer feedforward network with one centred sigmoidal activation function hidden layer and one linear activation function output layer. The weights and the biases of the hidden layer and of the output layer are denoted by ω_1, β_1, ω_2 and β_2, respectively. The network has one input, the scores t, and one output, the scores u. The centred sigmoid activation function σ is skew symmetric, centred around zero and valued between -1 and 1:

$$\sigma(z) = \frac{1-e^{-z}}{1+e^{-z}} = \tanh\left(\frac{z}{2}\right). \qquad \textbf{5-1}$$

The derivative of the centred sigmoid with respect to its argument z is:

$$\sigma'(z) = \frac{\partial \sigma}{\partial z} = \frac{1}{2} \cdot \left(1 - \sigma^2(z)\right) \qquad \textbf{5-2}$$

where $\sigma^2(z)$ denotes the square value of the centred sigmoid applied to its argument z.

Thus the non-linear inner relationship provided by the neural network can be written in explicit form as:

$$\boldsymbol{u} = \omega_2 \cdot \sigma(\omega_1 \cdot \boldsymbol{t} + \beta_1) + \beta_2 + \boldsymbol{e} \qquad \textbf{5-3}$$

and replacing $\boldsymbol{t}$ with $\boldsymbol{X} \cdot \boldsymbol{w}$, it becomes:

$$\boldsymbol{u} = \omega_2 \cdot \sigma\left(\omega_1 \cdot (\boldsymbol{X} \cdot \boldsymbol{w}) + \beta_1\right) + \beta_2 + \boldsymbol{e}\,. \qquad \textbf{5-4}$$

Where $\boldsymbol{X}$ denotes the input data matrix when referring to the first latent variable, or the deflated input data matrix when referring to subsequent latent variables. In Equations 5-3 and 5-4 the sigmoid activation function σ is applied to each element of the score vector $\boldsymbol{t} = \boldsymbol{X} \cdot \boldsymbol{w}$ and the summation between the scalar quantities β_1 and β_2 and the column vectors $\boldsymbol{t} = \boldsymbol{X} \cdot \boldsymbol{w}$ and $\sigma\left(\omega_1 \cdot \boldsymbol{t} + \beta_1\right)$, respectively, is defined as the summation between the scalar quantities and each element of the column vectors.

Denoting by w_k the outer input weight for the k-th variable $\boldsymbol{x}_k$ on the score $\boldsymbol{t}$, the first order Taylor series expansion of the non-linear function given in Equation 5-4 can be written as:

$$\boldsymbol{u} = f_{00} + \sum_{k=1}^{M} \frac{\partial f(\boldsymbol{X} \cdot \boldsymbol{w})}{\partial w_k} \cdot \Delta w_k \qquad \textbf{5-5}$$

where:

$$f_{\infty} = \hat{u} = f(X, w, \omega_1, \beta_1, \omega_2, \beta_2) \qquad \textbf{5-6}$$

$$\frac{\partial f(X \cdot w)}{\partial w_k} = \omega_2 \cdot \sigma'\left(\omega_1 \cdot (X \cdot w) + \beta_1\right) \cdot \omega_1 \cdot \frac{\partial (X \cdot w)}{\partial w_k} \qquad \textbf{5-7}$$

and:

$$\frac{\partial (X \cdot w)}{\partial w_k} = \frac{\partial}{\partial w_k}\left(\sum_{m=1}^{M} x_m \cdot w_m\right) = x_k \,. \qquad \textbf{5-8}$$

Thus:

$$\frac{\partial f(X \cdot w)}{\partial w_k} = \frac{\partial f(t)}{\partial w_k} = \frac{1}{2} \cdot \omega_1 \cdot \omega_2 \cdot \left(1 - \sigma^2\left(\omega_1 \cdot t + \beta_1\right)\right) \cdot x_k \qquad \textbf{5-9}$$

and the overall Newton-Raphson approximation can be written as:

$$u = f_{\infty} + \frac{1}{2} \cdot \omega_1 \cdot \omega_2 \cdot \sum_{k=1}^{m}\left(1 - \sigma^2\left(\omega_1 \cdot t + \beta_1\right)\right) \cdot x_k \cdot \Delta w_k \,. \qquad \textbf{5-10}$$

In Equations 5-9 and 5-10 the summation between scalar quantities and column vectors is defined as the summation between the scalar quantities and each element of the column vectors, $\sigma^2\left(\omega_1 \cdot t + \beta_1\right)$ denotes the column vector which comprises the square values of the centred sigmoid function applied to each element of the column vector $\left(\omega_1 \cdot t + \beta_1\right)$, while the product between the column vector given by $\sigma^2\left(\omega_1 \cdot t + \beta_1\right)$ and the column vector x_k, which comprises the measurement on the *k*-th input variable, is defined as the scalar product between each pair of corresponding elements in the two vectors (element by element product).

The error based updating procedure can then be applied by defining a matrix $Z = [z_k]$, where each column z_k is set equal to:

$$z_k = \frac{\partial f(t)}{\partial w_k} = \frac{1}{2} \cdot \omega_1 \cdot \omega_2 \cdot \left(1 - \sigma^2\left(\omega_1 \cdot t + \beta_1\right)\right) \cdot x_k . \qquad \textbf{5-11}$$

Arranging the Δw_k updating parameters into a column vector Δw, the Taylor series expansion can be written as:

$$u = f_{00} + Z \cdot \Delta w . \qquad \textbf{5-12}$$

Writing the mismatch between the value of u given by:

$$u = Y \cdot q \qquad \textbf{5-13}$$

and the value given by the neural network model:

$$\hat{u} = \left(\omega_2 \cdot \sigma(\omega_1 \cdot t + \beta_1) + \beta_2\right) \qquad \textbf{5-14}$$

as:

$$e = u - \hat{u} = Z \cdot \Delta w \qquad \textbf{5-15}$$

the PLS outer input weights w can be updated using the updating parameters:

$$\Delta w = (Z^T \cdot Z)^{-} \cdot Z^T \cdot e . \qquad \textbf{5-16}$$

The centred sigmoid neural network PLS algorithm can be implemented similarly to the quadratic PLS algorithms (§ 4). In fact, the only differences between the two algorithms are the training of the centred sigmoid neural network instead of the fitting of the polynomial expansion between the input and the output scores, and the partial derivatives used in the error based input weights updating procedure.

A detailed description of the modified NIPALS algorithm for the centred sigmoid neural network PLS with the input weights updating procedure is given hereafter.

0	mean centre and scale $\boldsymbol{X}$ and $\boldsymbol{Y}$	
1	set the output scores $\boldsymbol{u}$ equal to a column of $\boldsymbol{Y}$	
2	regress columns of $\boldsymbol{X}$ on $\boldsymbol{u}$	$w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	normalise $\boldsymbol{w}$ to unit length	$w = \frac{w}{\|w\|}$
4	calculate the input scores	$t = \frac{X \cdot w}{w^T \cdot w}$
5	train the centred sigmoid neural network between $\boldsymbol{t}$ and $\boldsymbol{u}$	$(\omega_1, \omega_2, \beta_1, \beta_2) \leftarrow (t, u)$
6	calculate the non-linear prediction of $\boldsymbol{u}$	$\hat{u} = \omega_2 \cdot \sigma(\omega_1 \cdot t + \beta_1) + \beta_2$
7	regress columns of $\boldsymbol{Y}$ on $\hat{u}$	$q^T = \frac{\hat{u}^T \cdot Y}{\hat{u}^T \cdot \hat{u}}$
8	normalise $\boldsymbol{q}$ to unit length	$q = \frac{q}{\|q\|}$
9	calculate new output scores	$u = \frac{Y \cdot q}{q^T \cdot q}$
10	compute the input weights updating parameters Δw as described above, i.e. Equations 5-11, 5-15 and 5-16.	
11	compute new input weights $\boldsymbol{w}$	$w = w + \Delta w$
12	normalise $\boldsymbol{w}$ to unit length	$w = \frac{w}{\|w\|}$
13	calculate new input scores	$t = \frac{X \cdot w}{w^T \cdot w}$
14	check convergence on $\boldsymbol{t}$: if YES goto 15 ELSE goto 5	

15	calculate the X loadings	$p^T = \frac{t^T \cdot X}{t^T \cdot t}$
16	train the centred sigmoid neural network between t and u	$(\omega_1, \omega_2, \beta_1, \beta_2) \leftarrow (t, u)$
17	calculate the non-linear prediction of u	$\hat{u} = \omega_2 \cdot \sigma(\omega_1 \cdot t + \beta_1) + \beta_2$
18	calculate input residual matrix	$E = X - t \cdot p^T$
19	calculate output residual matrix	$F = Y - \hat{u} \cdot q^T$
20	if additional PLS dimension are necessary then replace X and Y by E and F and repeat steps 1 to 20.	

Modified NIPALS Algorithm for Centred Sigmoid Neural Network PLS with Input Weights Updating Procedure.

5.6.2 The Taylor Series Expansion of the Radial Basis Function Network

The RBF network used is a two-layer neural network. The hidden layer has *NC* neurones with a Gaussian activation function. The network centres, biases and output weights are denoted by c_j, ρ_j and ω_j, respectively. The network has one input, the scores t, and one output, the scores u. The non-linear mapping provided by the RBF model can be written as:

$$u = \sum_{j=1}^{NC} \omega_j \cdot \exp\left(-\frac{r_j^2}{\rho_j^2}\right) + \omega_0 + e \qquad \textbf{5-17}$$

where:

$$r_j = \|c_j - t\|. \qquad \textbf{5-18}$$

is the column vector which comprises the distances between the centre of the j-th neurone and each input score t_n, and r_j^2 denotes the column vector which comprises the square value of the elements of r_j. Introducing an auxiliary column vector s_j, the signal produced by the Gaussian activation unit:

$$s_j = \exp\left(-\frac{r_j^2}{\rho_j^2}\right) \qquad \textbf{5-19}$$

the RBF model expression can be simplified to:

$$u = \sum_{j=1}^{NC} \omega_j \cdot s_j + \omega_0 + e\ . \qquad \textbf{5-20}$$

Denoting by w_k the outer input weight for the k-th variable x_k on the score t, the first order Taylor series expansion of the non-linear function given in Equation 5-20 can be written as:

$$u = f_{\infty} + \sum_{k=1}^{M} \frac{\partial f(t)}{\partial w_k} \cdot \Delta w_k \qquad \textbf{5-21}$$

where:

$$\frac{\partial f(t)}{\partial w_k} = \sum_{j=1}^{NC} \omega_j \cdot \frac{\partial s_j}{\partial w_k} \qquad \textbf{5-22}$$

and:

$$\frac{\partial s_j}{\partial w_k} = \left(-2 \cdot \frac{r_j}{\rho_j^2}\right) \cdot \exp\left(-\frac{r_j^2}{\rho_j^2}\right) \cdot \frac{\partial r_j}{\partial w_k}\ . \qquad \textbf{5-23}$$

Thus:

$$\frac{\partial f(t)}{\partial w_k} = \sum_{j=1}^{NC} \omega_j \cdot \left(-2 \cdot \frac{r_j}{\rho_j^2}\right) \cdot \exp\left(-\frac{r_j^2}{\rho_j^2}\right) \cdot \frac{\partial r_j}{\partial w_k}\ . \qquad \textbf{5-24}$$

The main problem related with this derivative is the evaluation of the derivative of $r_j = \|c_j - t\|$, since, although the Euclidean distance:

$$r_j = \|c_j - t\| = \sqrt{(c_j - t)^2} \qquad \textbf{5-25}$$

is a positive quantity, its argument can be negative or positive. Therefore, when evaluating the derivative $\frac{\partial r_j}{\partial w_k}$ this must be taken into account by the following conditions:

$$\frac{\partial r_j}{\partial w_k} = \frac{\partial\left(\|c_j - t\|\right)}{\partial w_k} = \frac{\partial\left(\sqrt{(c_j - t)^2}\right)}{\partial w_k} = \begin{cases} \dfrac{\partial t}{\partial w_k} & \text{IF} \quad (c_j - t) < 0 \\ -\dfrac{\partial t}{\partial w_k} & \text{IF} \quad (c_j - t) > 0 \end{cases} \qquad \textbf{5-26}$$

Now, since:

$$\frac{\partial t}{\partial w_k} = x_k \qquad \textbf{5-27}$$

where x_k is the column vector which comprises the observations collected on the k-th input variable, the overall derivative can then be written as:

$$\frac{\partial f(t)}{\partial w_k} = \frac{\partial f(X \cdot w)}{\partial w_k} = \sum_{j=1}^{NC} \omega_j \cdot s_j \cdot \left(-2 \cdot \frac{r_j}{\rho_j^2}\right) \cdot [\pm x_k] \qquad \textbf{5-28}$$

where the sign of $\pm x_k$ depends on the sign of $(c_j - t)$ (Equation 5-26). The overall Newton-Raphson approximation can then be written as:

$$u = f_{00} + \sum_{k=1}^{M} \left(\sum_{j=1}^{NC} \omega_j \cdot s_j \cdot \left(-2 \cdot \frac{r_j}{\rho_j^2}\right) \cdot [\pm x_k] \right) \cdot \Delta w_k \qquad \textbf{5-29}$$

and the error based updating procedure can be applied by defining a matrix $Z = [z_k]$, where each column z_k is set equal to:

$$z_k = \frac{\partial f(X \cdot w)}{\partial w_k} = \sum_{j=1}^{NC} \omega_j \cdot s_j \cdot \left(-2 \cdot \frac{r_j}{\rho_j^2} \right) \cdot [\pm x_k] . \qquad \textbf{5-30}$$

In Equations 5-23, 5-24, 5-28, 5-29 and 5-30 products between column vectors are defined as the scalar products between each pair of corresponding elements in the vectors (element by element products).

Arranging the Δw_k updating parameters into a column vector Δw, the Taylor series expansion can be written as:

$$u = f_{\infty} + Z \cdot \Delta w . \qquad \textbf{5-31}$$

Writing the mismatch between the value of u given by:

$$u = Y \cdot q \qquad \textbf{5-32}$$

and the value given by the RBF network model:

$$\hat{u} = \left(\sum_{j=1}^{NC} \omega_j \cdot s_j + \omega_0 \right) \qquad \textbf{5-33}$$

as:

$$e = u - \hat{u} = Z \cdot \Delta w \qquad \textbf{5-34}$$

the PLS outer input weights w can be updated using the updating parameters:

$$\Delta w = (Z^T \cdot Z)^{-} \cdot Z^T \cdot e . \qquad \textbf{5-35}$$

A detailed description of the modified NIPALS algorithm for radial basis function neural network PLS with the input weights updating procedure is given hereafter.

0	mean centre and scale X and Y	
1	set the output scores u equal to a column of Y	
2	regress columns of X on u	$w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	normalise w to unit length	$w = \frac{w}{\|w\|}$
4	calculate the input scores	$t = \frac{X \cdot w}{w^T \cdot w}$
5	train the radial basis function neural network between t and u	$(c_j, \rho_j, \omega_j) \leftarrow (t, u)$
6	calculate the non-linear prediction of u	$\hat{u} = \sum_{j=1}^{NC} \omega_j \cdot \exp\left(-\frac{\|c_j - t\|^2}{\rho_j^2}\right) + \omega_0$
7	regress columns of Y on $\hat{u}$	$q^T = \frac{\hat{u}^T \cdot Y}{\hat{u}^T \cdot \hat{u}}$
8	normalise q to unit length	$q = \frac{q}{\|q\|}$
9	calculate new output scores	$u = \frac{Y \cdot q}{q^T \cdot q}$
10	compute the input weights updating parameters Δw as described above, i.e. Equations 5-30, 5-34, and 5-35.	
11	compute new input weights w	$w = w + \Delta w$
12	normalise w to unit length	$w = \frac{w}{\|w\|}$
13	calculate new input scores	$t = \frac{X \cdot w}{w^T \cdot w}$

14	check convergence on t: if YES goto 15 ELSE goto 5	
15	calculate the X loadings	$p^T = \frac{t^T \cdot X}{t^T \cdot t}$
16	train the radial basis function neural network between t and u	$(c_j, \rho_j, \omega_j) \leftarrow (t, u)$
17	calculate the non-linear prediction of u	$\hat{u} = \sum_{j=1}^{NC} \omega_j \cdot \exp\left(-\frac{\|c_j - t\|^2}{\rho_j^2}\right) + \omega_0$
18	calculate input residual matrix	$E = X - t \cdot p^T$
19	calculate output residual matrix	$F = Y - \hat{u} \cdot q^T$
20	if additional PLS dimension are necessary then replace X and Y by E and F and repeat steps 1 to 20.	

Modified NIPALS Algorithm for Centred Sigmoid Neural Network PLS with Input Weights Updating Procedure.

5.7 Comparison of the Error Based NNPLS and RBFPLS Algorithms with the Original Published Algorithms

The NNPLS algorithm proposed by Qin and McAvoy (1992) and the RBFPLS algorithm proposed by Wilson *et al.* (1997) are compared with the newly developed algorithms based on the use of the error based weights updating procedure. The performance was compared through their application to the modelling of the data used in Chapter 4 to compare the original quadratic PLS algorithm with the error based quadratic PLS approach.

The centred sigmoidal network and the radial basis functions network were trained using the tools available in the Optimisation and the Neural Network Toolboxes for MATLAB 4.0. In particular the centred sigmoidal networks were trained using the BFGS algorithm with one set of

randomly initialised weights, whilst the radial basis function networks were built using the orthogonal least squares (OLS) approach proposed by Chen *et al.* (1991) (Appendix 2) with a fixed band width for all the network centres. Although it might be argued that this is not the most robust way to build the neural network models (each network was trained with only one set of initial weights) and in the case of the radial basis function network models it would be more reasonable to chose a different band width for each centre, the results obtained in this way did not show any lack of accuracy or robustness in the training procedure.

The major drawback in building both the NNPLS and RBFPLS models was the selection of the network structure. To avoid this inconvenience an exhaustive search (from a simpler to a more complex structure) was carried out to identify the best model structure, in terms of both network model and number of latent variables. The criterion to select the best model was the minimisation of the mean square prediction error (MSPE) on unseen output data, after splitting the data sets into a training subset and a testing subset. In particular the best models have been selected according to the minimum MSPE on unseen data over all the combinations of latent variables and neurones or centres, without any restriction on the number of latent variables to include in the model. A different search could have been done looking for the model based on the use of a predefined number of latent variables only and with the minimum MSPE.

It is noted that an exhaustive cross validation search is highly time demanding in selecting the best model and therefore no formal cross validation was carried out on the training data set to select the best PLS model. Nevertheless, the selection of the model was still based on a minimisation criterion similar to the one used in cross validation and it was accepted as being a suitable compromise. In particular the neural network inner models for the NNPLS algorithms were trained for 1000 and for 2500 iterations using 9 different structures, from 3 up to 11 neurones in the hidden layer, with random initialisation of the network weights. A maximum of 11 nodes appeared sufficient for a first trial and this was confirmed by the results discussed in the following sections. The network weights were drawn from a normally distributed population with values between –0.1 and +0.1. For the RBFPLS algorithms 19 sets of 4 different radial basis function networks were built, changing the number of centres from 3 to 21 hidden units and the band width associated with them in steps of 0.5, 1.0, 1.5 and 2.0, respectively.

When building the radial basis function models, a better design could have been achieved using different band width values for different centres. Unfortunately this was not possible because of a major limitation in the Neural Networks Toolbox of MATLAB 4.0, which does not allow the use of different spread factors for the Gaussian activation function. In any case, the results achieved in this way were sufficiently good to justify the use of a fixed band width, in addition to the savings in terms of training time.

The overall database consists of a number of network topologies built for each PLS model. In particular for each one of the two NNPLS algorithms (with and without weights updating) there were a total of $9 \times 2 = 18$ models; whilst for each one of the two RBFPLS algorithms (with and without weights updating) there were $19 \times 4 = 76$ models. Since the two data sets used to test the reliability of the error based weights updating procedure with the new neural network PLS algorithms consisted of four input variables, the number of models available for each algorithm must be multiplied by 4. A legend with the abbreviations used to designate the different NNPLS and RBFPLS algorithms is given in Table 5.1.

5.7.1 Case Study 1 - A Synthetic Example.

The data for this example were those used for comparing the original quadratic PLS approach with the error based quadratic PLS algorithm in Section 4.6.1. In Table 5.2 the mean square prediction error (MSPE) for the output variable on unseen data is given for each model. The improvement in prediction capabilities of both the NNPLS and the RBFPLS algorithms in conjunction with the error based weights updating procedure when compared to the same algorithms, but without any weight updating procedure, is evident.

It can be observed that the two best models exhibit almost the same capabilities in terms of MSPE. However the RBFPLS approach is arguably more convenient (applicable) than the NNPLS approach, since the training of the neural network inner model of the NNPLS algorithm takes a significantly longer time than building the radial basis function inner model. In fact the network training procedure is iterated several times by the weight updating procedure and computation time becomes a key parameter when choosing the methodology to use. In particular with MATLAB 4.0 on a P200 machine with 128MB RAM, building the NNPLS model with the

error based updating procedure takes almost two hours, whilst building the RBFPLS model with the error based updating procedure takes less than twenty minutes.

In Table 5.3, the MSPE on testing data for the best two new algorithms are compared with those obtained for the linear, original quadratic and error based quadratic PLS models. From these results the improvement achieved using a neural network to model the inner relation is evident. It can be seen that both the error based NNPLS and RBFPLS algorithms perform better than the error based QPLS algorithm. Surprisingly, though, the cumulative variance of the *Y*-block captured by the error based NNPLS algorithm and by the error based RBFPLS algorithm (Tables 5.4a and 5.4b) is less than the cumulative variance for the same output block captured by the error based QPLS (Table 4.1c). In particular both the neural PLS algorithms perform slightly better than the QPLS in terms of output variance explained by the first latent variable, although both algorithms loose their advantage on subsequent latent variables. The reason for this can be deduced by observing the score scatter plots.

Figures 5.1 to 5.4 highlight the effect of the error based weights updating procedure on the NNPLS and RBFPLS algorithms. Both the centred sigmoid network and the radial basis function network are capable of modelling the non-linear relationship between the t and u scores. However, the error based weights updating procedure, in changing the projection parameters modifies the shape of the input/output latent variables scatter plots and enhances the performance of the network model. In general the centred sigmoid network tends to overfit the data. This feature is enhanced for the NNPLS model without weights updating (Figures 5.1a and 5.1b). However, the error based NNPLS approach also shows some tendency to overfit the data (Figure 5.2b). This behaviour can be explained by considering that the neural networks were trained without any attempt to avoid overfitting. The training algorithms were terminated after 1000 or 2500 iterations. The RBFPLS algorithm without weights updating shows greater tendency to overfit the data (Figure 5.3). It is possible to observe (especially from Figures 5.2b and 5.4b) that the model provided by the radial basis function network in conjunction with the error based weights updating procedure is smoother than the model provided by the other approaches. From this it can be concluded that by changing the shape and the spread of the score scatter plot, the error based updating procedure can reduce (if not prevent) overfitting.

These score plots can be compared with that attained with the error based Quadratic PLS (Figure 4.3). It appears that the three error based algorithms perform in a similar way, with the main differences occurring in the inner mapping relationship. The error based NNPLS and RBFPLS algorithms appear to be affected by the distribution of the data more than the error based quadratic PLS. This is evident by observing the scatter plot on the fourth latent variable (Figures 4.3d, 5.2d and 5.4d). The error based QPLS algorithm fits a quadratic relationship between the input and output scores on the fourth latent variable (Figure 4.3d), mainly because the residual from the previous latent variable model contains a parabolic type of structure up to the fourth component. In contrast the neural network model cannot fit more than a straight line between the input and the output scores, even if it is still possible to identify inflexions at the side of the cluster of points, i.e. a sort of parabolic relationship (Figure 5.2d). By comparison the RBF inner model is able to capture the inflexions at the sides of the cluster, but cannot approximate more than a straight line inside the cluster of points (Figure 5.4d). The difference in behaviour between the NNPLS and RBFPLS models can be explained by considering the global modelling capabilities of the centred sigmoidal network in comparison with the local modelling properties of the radial basis function network. The centred sigmoidal network tends to develop an approximation over all the space spanned by the input latent variable, whilst the radial basis function network approximates the input/output scores relationship over small intervals (locally) of the space spanned by the input latent variable. Again, considering the output score residuals of the second latent variable (Figure 5.5) the filtering effect of the error based weights updating procedure can be observed from the reduced spread of the residuals.

Figure 5.6 shows the actual and the predicted values of the output, on unseen data, for the best NNPLS model (Figure 5.6a) and for the best RBFPLS model (Figure 5.6b). Again the two algorithms appear to be very similar in their predictions, and comparable with the error based quadratic PLS algorithm. The same situation holds when considering models based on fewer latent variables.

5.7.2 Case Study 2 - Simulation of an Industrial pH Problem

The data for this example is that used for comparing the original quadratic PLS approach with the error based quadratic PLS algorithm in section 4.6.2.

The results given in Table 5.5 show the improvement in the prediction capabilities in terms of MSPE with the inclusion of the error based updating procedure within both the NNPLS and the RBFPLS algorithms. Furthermore, the effect of not establishing a proper training procedure for the NNPLS algorithm can be observed. It appears not to perform as well as the RBFPLS. When comparing these results with those achieved with the error based QPLS algorithm, the effect of a less than perfect network training procedure appears to impact upon the approximation capabilities of the NNPLS algorithm when the results are compared to those achieved by the QPLS (Table 5.6). In contrast the RBFPLS model shows better predictive capabilities than the QPLS algorithm, even though its first latent variable appears to capture less variance. The cumulative variance captured by the two error based network algorithms is given in Table 5.7. In particular Table 5.7a reflects the weakness of the NNPLS algorithm with respect to the RBFPLS algorithm, whilst Tables 4.4c and 5.7b highlights the main differences between the RBFPLS algorithm and the QPLS algorithm. It can be seen that even though the QPLS approximation model for the first latent variable is better, the RBFPLS model captures a higher percentage of variance from the second latent variable and overall a greater proportion of the variance is explained on the output.

These differences are reflected in the score plots for the three models, given in Figures 5.7 to 5.10. The NNPLS, algorithm without weights updating, tends to fit a combination of a straight line and a sigmoid shape to approximate the inner relationship. The final effect is to loose the actual relationship between the input and the output latent variables. In contrast the error based NNPLS algorithm, even with the improvement due to the weights updating, tends to fit the inner relationship with a straight line. This is symptomatic of the neural network used for the inner model not being trained properly, and confirms the assertion that the network training procedure plays a pivotal role in building the regression model. It is also noted that in the case of the NNPLS algorithm with the weights updating procedure, the best model was that based on an inner network with 11 neurones, requiring excessive training times. Furthermore, it also suggests that the straight line regression results from an overlapping effect of the neurones. In comparison, the RBFPLS algorithm captures the underlying distribution of the input/output scores. Again the effect of the weights updating procedure plays an important role since it dramatically reduces the spread of the scores on the first latent variable and thus enhances the fitting of the regression model. It is also noted, from Table 5.5b, that the best error based RBFPLS algorithm requires eight centres (i.e. eight gaussian units) and this feature is reflected in the plots of Figures 5.10b and 5.10c where the radial basis networks appear to follow the

scores distribution with a fairly oscillatory curve, reinforcing the necessity of sound training algorithms also for the RBF networks.

Particular attention might be paid to Figures 4.9 and 5.10, when comparing the RBFPLS and the QPLS approaches. It can be seen in Figure 5.10a and 4.9a that the first radial basis function model is better than the quadratic polynomial interpolation, but in terms of variance explained the quadratic polynomial interpolation provided a better approximation than the gaussian network. A reason for this behaviour might be found in the fact that in the case of the QPLS algorithm the scores on the first latent variable are less spread around the non-linear prediction than with the RBFPLS algorithms. The score plots on the second and third latent variable (Figures 5.10b, 5.10c, 4.9b and 4.9c) show the reason for the increase in prediction capabilities of the error based RBFPLS algorithm compared to that of the error based QPLS. It seems that the gaussian network is able to fit the non-linear relationship characterising the inner mapping on the second and third components, while the quadratic polynomial expansion cannot. This is probably due to the trend of the input/output scores functional relationship not being truly quadratic. This case highlights the limitation that may be encountered when using predefined models, such as a polynomials expansion of a fixed degree, instead of a universal approximator for modelling purposes. Figure 5.11 illustrates the final predictions for the test data set for the two error based neural network PLS approaches. It is observed that the prediction of the NNPLS algorithm is poor in comparison with the RBF algorithm. However, when comparing the error based RBFPLS algorithm (Figure 5.11b) with the error based QPLS approach (Figure 4.13) the enhanced prediction achieved by combining the error based weight updating procedure with the RBFPLS approach becomes evident.

Sigmoidal network and radial basis function networks are well known to have overfitting tendencies, a criticism levelled against many poorly constructed empirical modelling approaches, especially neural networks. Indeed it is a major concern since the best RBFPLS model uses eight gaussian units. However the results achieved on the testing data set clearly indicate that the error based algorithm is not prone to overfitting, and that the combination of the error based weight updating procedure with the RBF network model provides a reliable non-linear regression tool. The same argument holds for the NNPLS algorithm, even though it requires a rigorous training procedure to build the final model which may lead to unacceptable training times. This is particularly true when considering that in the error based NNPLS the network training procedure is nested within the iterative updating of the input weights. In

contrast, even if a RBF training procedure is nested within the same iterative procedure, it can be computed in a straightforward way using faster training algorithms which are not based upon non-linear optimisation methods.

5.7.3 Case Study 3 : Simulation of an Industrial pH Problem

The error based updating procedure was further evaluated on the pH neutralisation simulation model used previously, but with a different set of data. The simulation was run in the same way as described in Section 4.6.2, randomly changing the flowrates of the inlet streams, changing outlet stream flowrate in order to maintain the level in the tank constant, and collecting the values of the pH on the outlet stream at steady state. The only difference with respect to the data set used in the previous example is that the base flowrate (Q3) was changed in order to allow pH values to cover the range from pH 3 to pH 11 (Figure 5.12). From Figures 5.12c and 5.12d the effect of the base flowrate on the pH values is evident, and how the outlet flowrates (Q4) reflects the relationship between Q3 and the pH values is clearly shown. These two plots also show the negligible effect of the other two streams on the pH value. Furthermore from the relationship depicted in Figures 5.12c and 5.12d, it is expected that a single non-linear relationship should be able to model the system. Also in this case a data set of 999 samples was generated and was split into a training data set of 700 samples and a testing data set of 299 observations.

Six different algorithms were tested on these data: the original quadratic PLS algorithm, the neural network and the radial basis function PLS algorithms, and the corresponding algorithms enhanced with the error based weights updating procedure. Also in this case the selection of the best NNPLS and RBFPLS models was carried out by an exhaustive search as described in Section 5.7. The results of this exhaustive search for the NNPLS and RBFPLS algorithms are given in Table 5.8. Again the weights updating procedure enhanced the performances of the two algorithms. Furthermore it can be observed that a large number of neurones is required for both algorithms. This was to be expected because of the shape of the curve shown in Figures 5.12c and 5.12d which exhibits a hyperbolic-tangent trend with four different inflexion points. This feature also reflects upon the performance of the original QPLS and the error based QPLS algorithms. The two algorithms are seen not to differ too much in their modelling performance with the error based QPLS performing slightly better (Table 5.9). This can be also observed from the plot for the first input/output latent variables (Figure 5.13). Both the quadratic algorithms seek to fit the inner mapping with a quadratic relationship, which is not the most

suitable. Nevertheless, the error based QPLS approach shows a reduction in the spread of the scores which was also observed in the other examples.

Both the original NNPLS and RBFPLS algorithms appear capable of approximating the true shape of the pH tritation curve, including the inflexion points. This occurs directly in the first input/output latent variable mapping (Figure 5.13c and 5.13d). The NNPLS algorithm, with 9 neurones, appears to be more stable than the RBFPLS algorithm, with 18 neurones, which oscillates and appears to overfit the data. This is mainly due the local modelling properties of the radial basis functions. The same situation is shown in the corresponding plots for the error based NNPLS and RBFPLS algorithms (Figure 5.13e and 5.13f). These plots provide evidence that the radial basis function inner model is more capable than the centred sigmoid network inner model of approximating the steep changes in the pH curve. However, the sigmoid network based PLS algorithm appears to be more stable than the radial basis function at the edges of the pH curve. This is mainly due to the RBFPLS using a large number of neurones (21), each with the same bandwidth (0.5). This topology provide good modelling capabilities where the curve exhibits rapid changes (at the centre, for the input scores lying between -1 and 1), with a tendency to overfit the mapping where the curve shows an almost constant slope (at the edges, with input scores smaller than-1 and bigger than 1). Nevertheless, the error based RBFPLS algorithm results in an overall smaller MSPE than the error based NNPLS algorithm on unseen data (Tables 5.8 and 5.9). Furthermore, as discussed previously in Section 5.7, improved training algorithms for radial basis function networks are available for selecting more appropriate centre locations and bandwidths.

5.8 Discussion and Conclusions

The use of sigmoid and gaussian networks represents a major step forward in the development of non-linear PLS algorithms. In particular, they provide the universal approximation capabilities that polynomial expansions or SPLINE functions lack. Because of this, neural network based PLS algorithms represent an appealing alternative to polynomial PLS algorithms and the traditional direct neural networks approach. However, whilst for the quadratic PLS approach proposed by Wold *et al.* (1989) and the SPLINE-PLS approach proposed by Wold (1992) an updating procedure for the input weights had been proposed, this was not the case for the neural

network based PLS algorithms. Furthermore, as observed before (§ 5.4), not all the neural network PLS algorithms can be classified within the family of the PLS approaches which rely upon the NIPALS algorithms. More precisely, it was observed that even though the INNPLS algorithm proposed by Saunders (1992) includes an updating procedure, this is performed by means of an optimisation routine which regards the mapping between the input variables and the corresponding input latent variables as an extension of the network used to fit the inner non-linear model between the input and output latent variables. Consequently, the INNPLS algorithm can be arguably considered as a NIPALS based non-linear PLS approach.

Amongst the neural network PLS approaches reported in the literature, the neural network PLS algorithm proposed by Qin and McAvoy (1992) and the radial basis function PLS algorithm proposed by Wilson *et al.* (1997) appear to be most consistent with the NIPALS approach. In fact, they fall into the class of "*quick and dirty*" algorithms described by Wold *et al.* (1989) which use a non-linear function to fit the mapping between the input and output latent variables, regardless of the relationship which exists between the input weights and the output scores. As observed by Wold *et al.* (1989) this approach can be safely employed as far as the relationship between the input and the output latent variables is *slightly* non-linear, which implies that the effect of an updating procedure of the input weights would be almost negligible. However, when dealing with highly non-linear systems, these algorithms cannot be exploited in full. In fact, even though the inner mapping is enhanced by the use of the universal approximation properties of the sigmoid or the gaussian network, they both lack optimisation of the input weights, which is required to maintain the linkage between the input outer mapping and the inner mapping.

To overcome this limitation, the error based updating procedure proposed in the previous Chapter (§ 4.5) has been extended to the case of the sigmoid and gaussian neural network PLS approaches. This was possible because both the sigmoid and the gaussian function are continuous and differentiable with respect to the input weights, and hence can be linearized by means of a Taylor series expansion (§ 5.6.1 and 5.6.2).

The effect of combining the error based weights updating procedure with the universal approximation properties of the sigmoid and the gaussian networks is evident when comparing the performances of the different approaches. In particular, the use of sigmoid and gaussian networks to fit the inner mapping between the input and the output latent variables provides the generality that the quadratic expansion lacks. This is particularly evident when the shape of the

inner mapping is not parabolic (§ 5.7.3). Furthermore, the use of the error based weights updating procedure appears to enhance the performance of the sigmoid and gaussian networks, since it can reduce the spread of the input/output scores around the underlying functional relationship. This feature of the error based weights updating procedure had previously been observed in Chapter 4 when comparing the two quadratic PLS approaches (§ 4.6), although it did not appear to significantly improve the performances of the quadratic fitting. However, when using neural networks to fit the inner mapping, the error based updating procedure appears to be capable of reducing the possibilities for the networks training algorithms to overfit the data (§ 5.7). This is a well known drawback of optimisation routines for sigmoid and radial basis function networks.

In this Chapter the necessity for a sound training strategy is clearly evident. Both the sigmoid and the gaussian networks suffer from sub-optimal training procedures. In particular in training the sigmoid networks, no stopping criteria was used. Instead, the algorithms were terminated when 1000 or 2500 iterations were reached. As a consequence, the networks parameters, might not be optimal, or alternatively the algorithm might have been stopped before it had reached its minimum. Similar problems hold for the gaussian network basis function, since the algorithm used to train the RBF networks did not provide the feature to select different bandwidths for different centres. in practice it would have been better to have the option of using different bandwidths for different centres, or activation functions other than the gaussian (Appendix 2). Nevertheless, the error based sigmoid neural network and radial basis function PLS algorithms exhibit the general approximation properties expected of them, and it is anticipated that if improved training algorithms had been used, the results may have been more satisfactory in terms of perceived benefits.

Model	Description
net_1000	NNPLS without updating procedure, 1000 training iterations
net_2500	NNPLS without updating procedure, 2500 training iterations
up_net_1000	NNPLS with updating procedure, 1000 training iterations
up_net_2500	NNPLS with updating procedure, 2500 training iterations
rbf_0.5	RBFPLS without updating procedure, band width 0.5
rbf_1.0	RBFPLS without updating procedure, band width 1.0
rbf_1.5	RBFPLS without updating procedure, band width 1.5
rbf_2.0	RBFPLS without updating procedure, band width 2.0
up_rbf_0.5	RBFPLS with updating procedure, band width 0.5
up_rbf_1.0	RBFPLS with updating procedure, band width 1.0
up_rbf_1.5	RBFPLS with updating procedure, band width 1.5
up_rbf_2.0	RBFPLS with updating procedure, band width 2.0

Table 5.1 : Legend for the PLS model names.

Model	Neurones	Latent Variables	**MSPE**
net_1000	6	1	8.5870E-03
net_2500	11	1	8.3414E-03
up_net_1000	9	4	1.2980E-03
up_net_2500	7	4	**1.0518E-03**

Table 5.2a : MSPE for the mathematical function for the NNPLS algorithm (testing data).

Model	Centres	Latent Variables	**MSPE**
rbf_0.5	4	4	4.1603E-03
rbf_1.0	8	4	4.9091E-03
rbf_1.5	8	4	5.5315E-03
rbf_2.0	4	4	5.4743E-03
up_rbf_0.5	9	4	1.2644E-03
up_rbf_1.0	5	3	1.1531E-03
up_rbf_1.5	5	4	1.1203E-03
up_rbf_2.0	4	3	**1.0473E-03**

Table 5.2b : MSPE for the mathematical function for the RBFPLS algorithm (testing data).

Table 5.2 : Performance comparison between the different neural network PLS algorithms.

LV#	*1*	*2*	*3*	*4*
Linear PLS	**2.0679E-02**	2.0687E-02	2.0687E-02	2.0687E-02
Wold QPLS	4.3999E-03	3.1792E-03	2.6038E-03	**2.4380E-03**
EB QPLS	4.3888E-03	1.7475E-03	1.4021E-03	**1.3841E-03**
up_net_2500	4.1299E-03	1.3982E-03	1.0590E-03	**1.0518E-03**
up_rbf_2.0	4.0449E-03	1.3152E-03	**1.0473E-03**	1.0551E-03

Table 5.3 : Performance comparison between the different non-linear PLS algorithms.

	X-block		***Y***-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	21.2320	21.2320	70.7082	70.7082
2	29.0746	50.3066	20.6115	91.3197
3	25.1634	75.4699	1.9854	93.3051
4	24.5301	100	0.0116	93.3168

Table 5.4a : Percentage of Variability Explained - Error Based NNPLS Algorithm.

	X-block		***Y***-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	21.2158	21.2158	70.7224	70.7224
2	29.0726	50.2884	20.7178	91.4402
3	25.1707	75.4591	1.9763	93.4166
4	24.5409	100	0.2313	93.6478

Table 5.4b : Percentage of Variability Explained - Error Based RBFPLS Algorithm.

Table 5.4 : Comparison of Percentage of Variability Explained for the Error Based Neural Network and Radial Basis Function PLS Algorithms for the Mathematical Function.

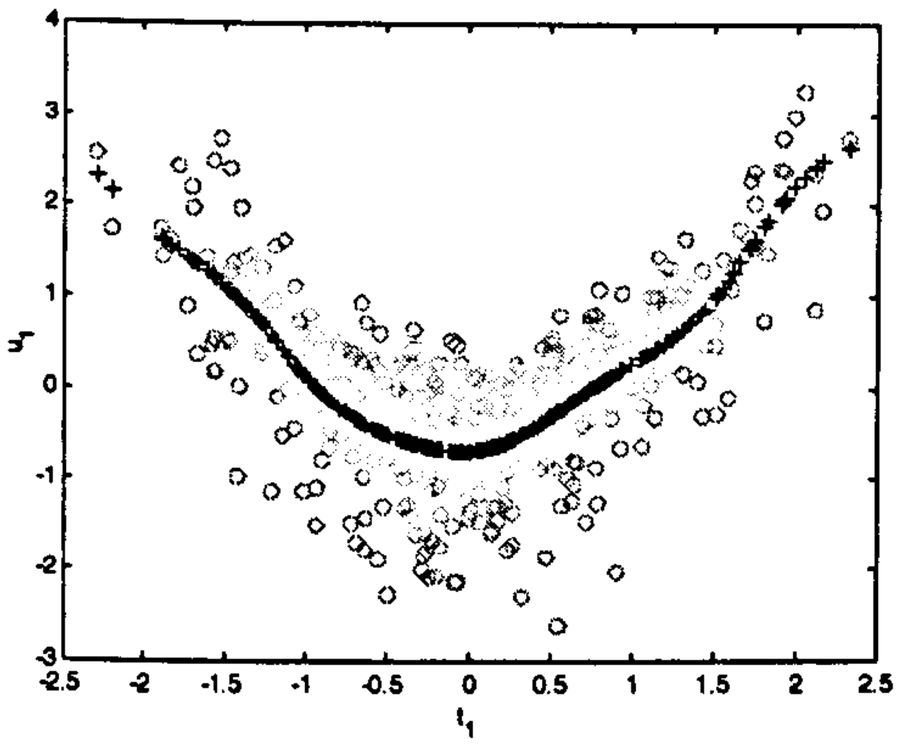

Figure 5.1a : NNPLS, first latent variable scatter plot.

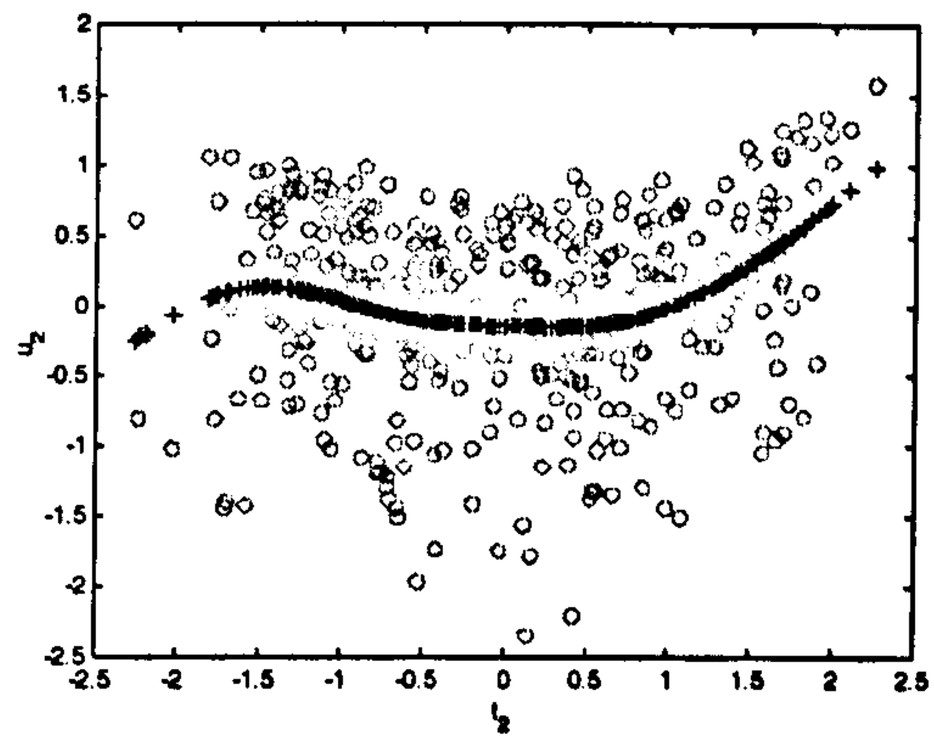

Figure 5.1b : NNPLS, second latent variable scatter plot.

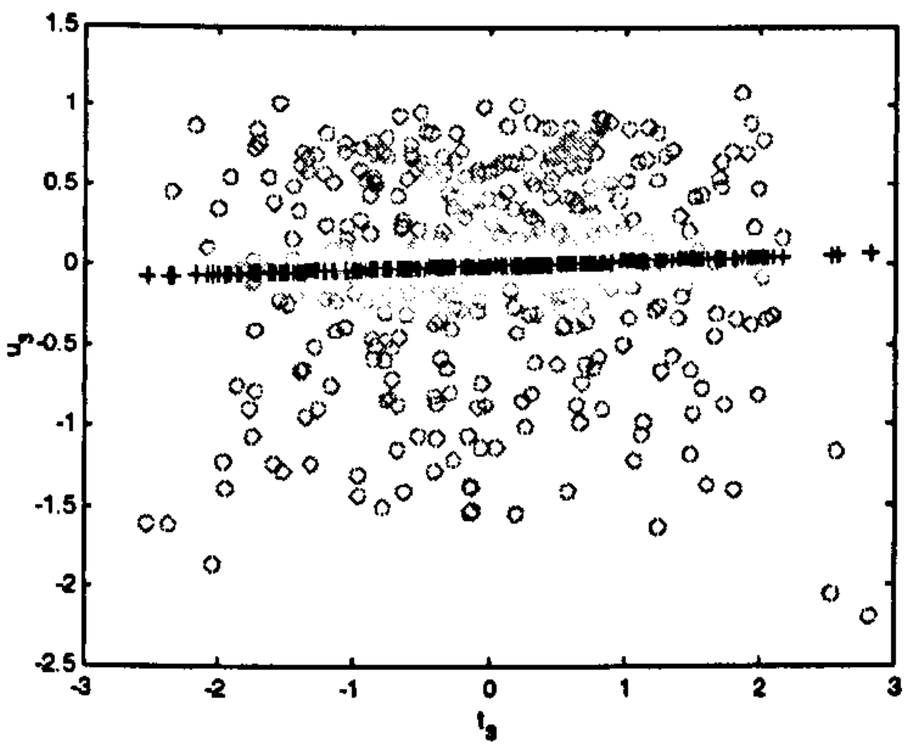

Figure 5.1c : NNPLS, third latent variable scatter plot.

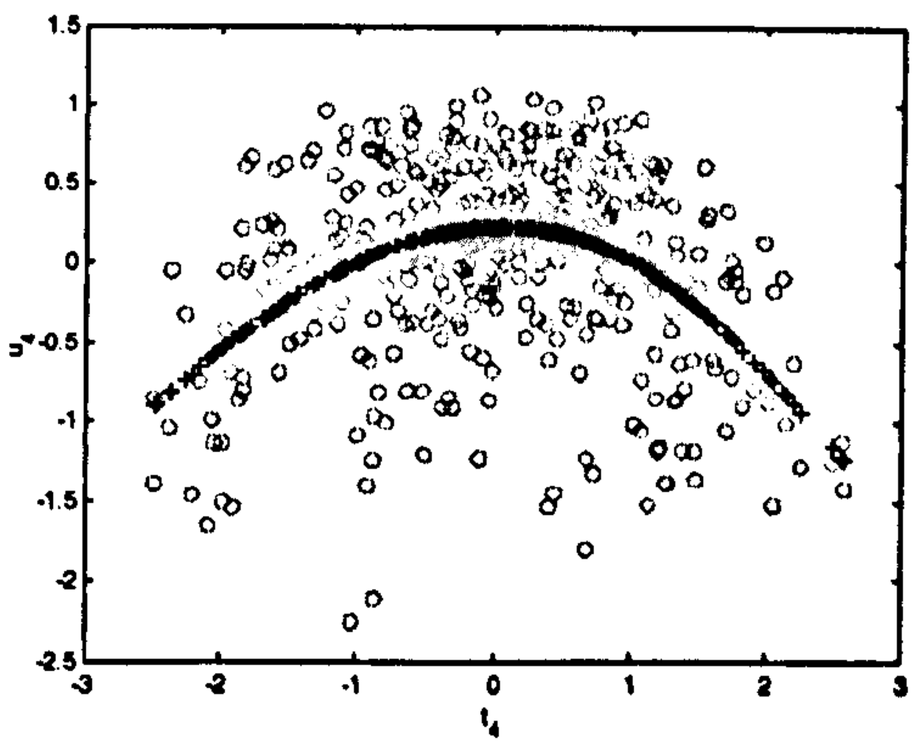

Figure 5.1d : NNPLS, fourth latent variable scatter plot.

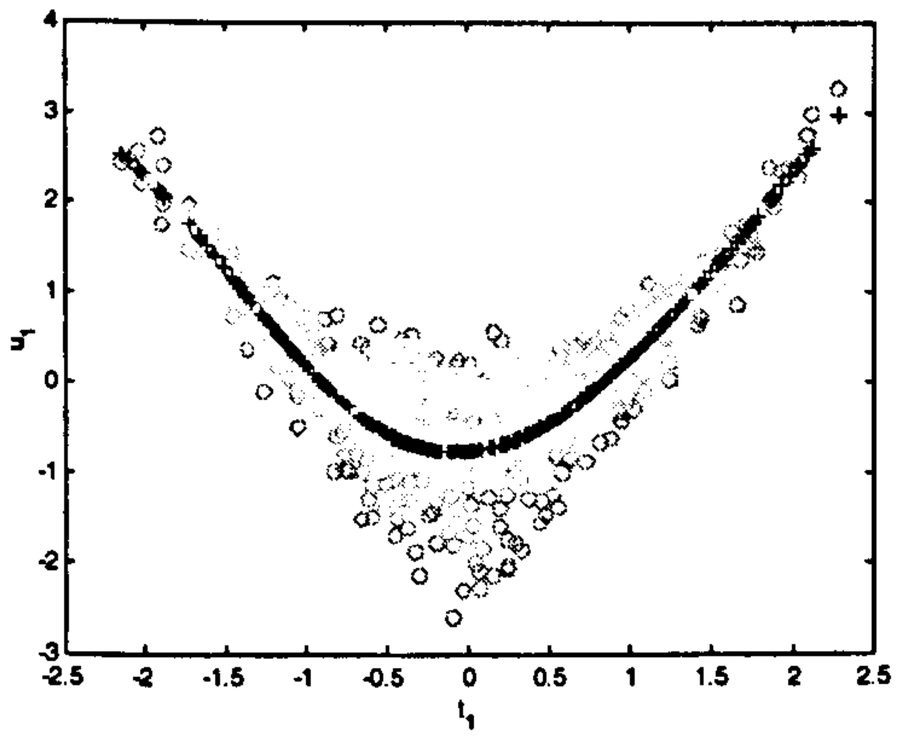

Figure 5.2a : Error Based NNPLS, first latent variable scatter plot.

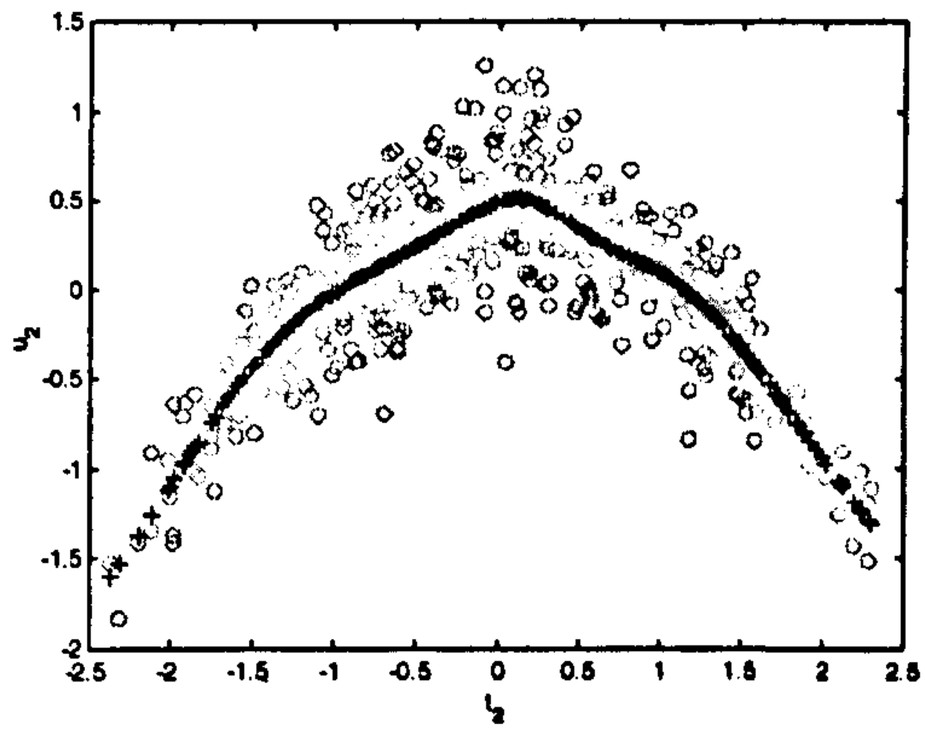

Figure 5.2b : Error Based NNPLS, second latent variable scatter plot.

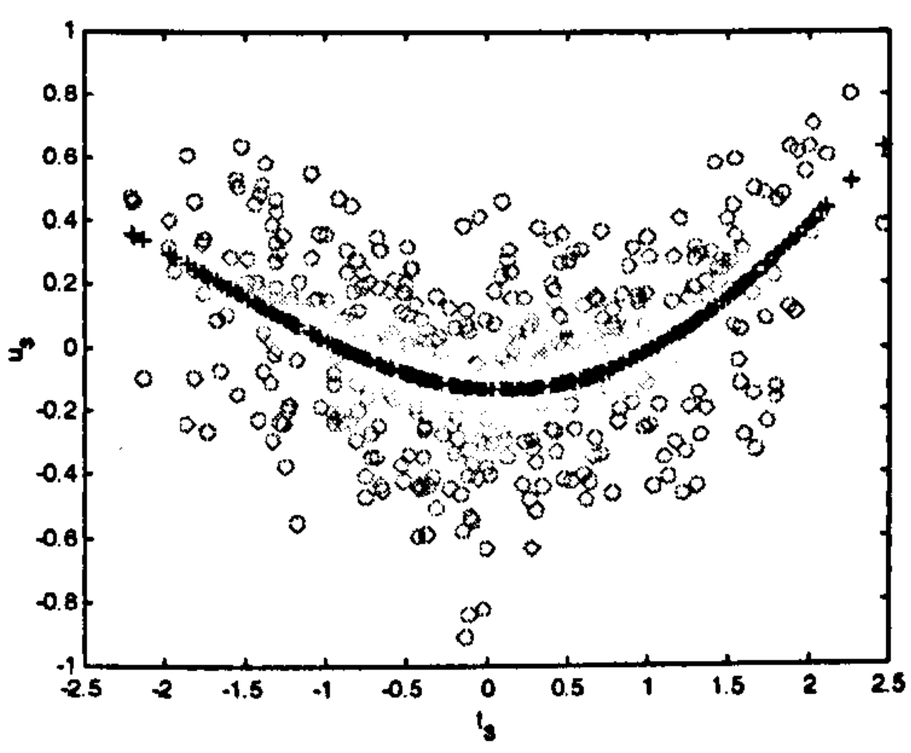

Figure 5.2c : Error Based NNPLS, third latent variable scatter plot.

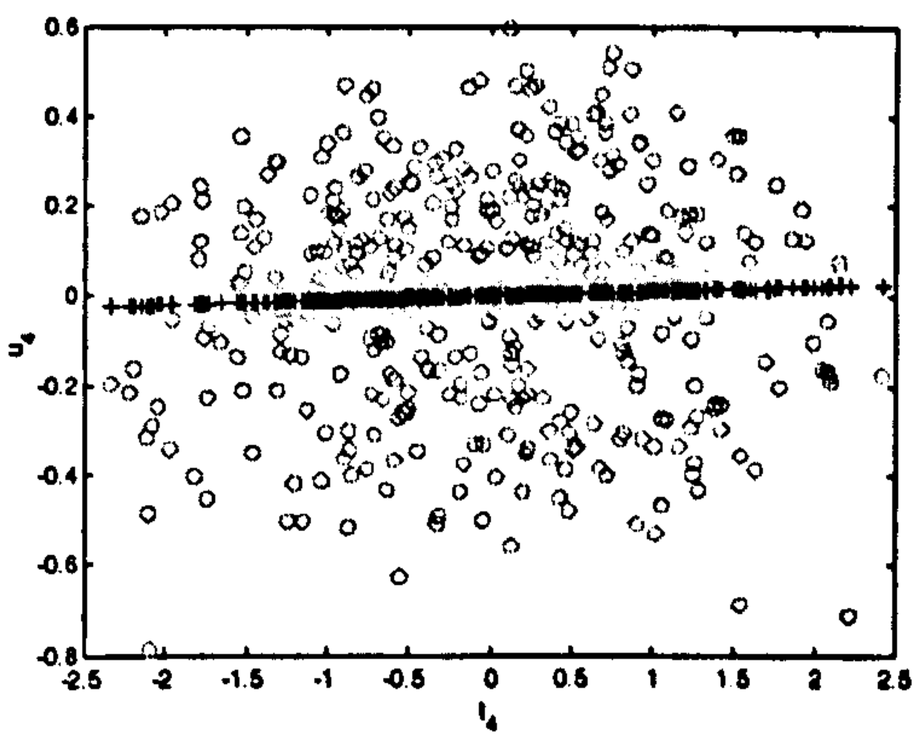

Figure 5.2d : Error Based NNPLS, fourth latent variable scatter plot.

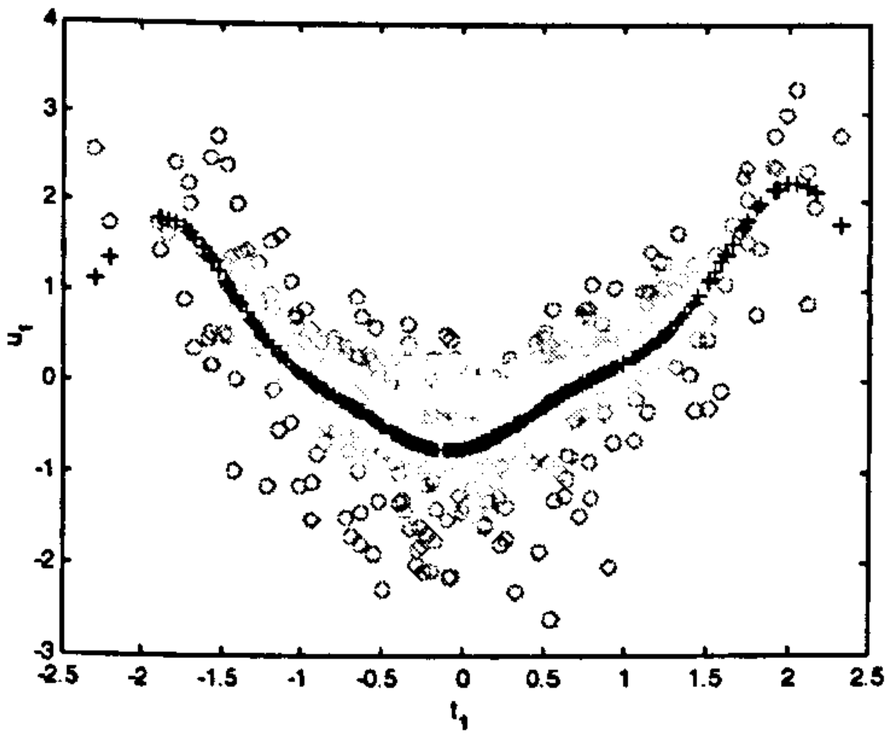

Figure 5.3a : RBFPLS, first latent variable scatter plot.

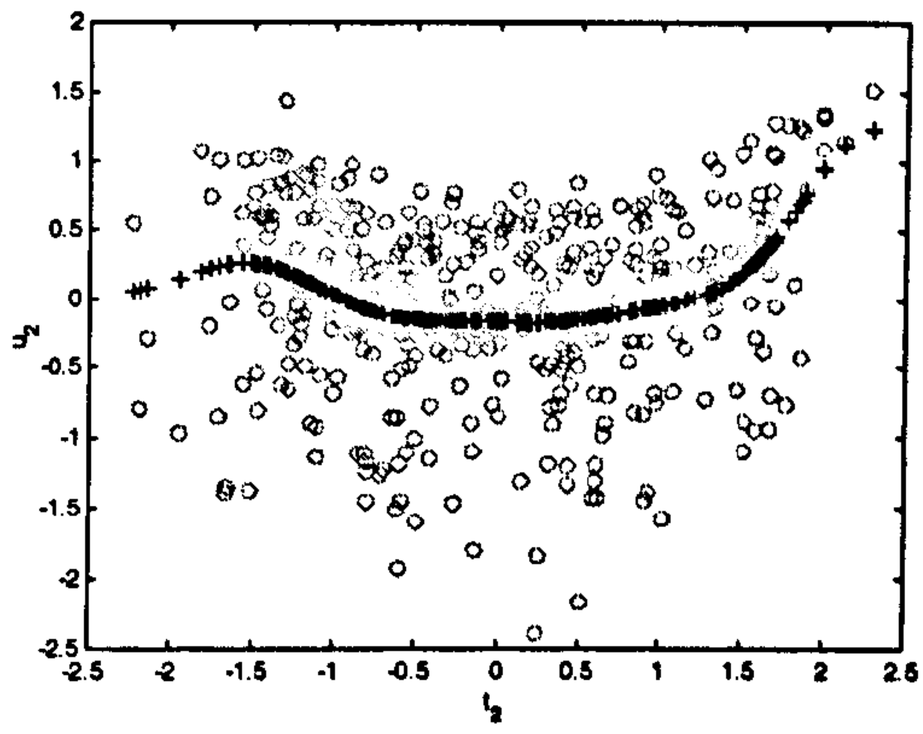

Figure 5.3b : RBFPLS, second latent variable scatter plot.

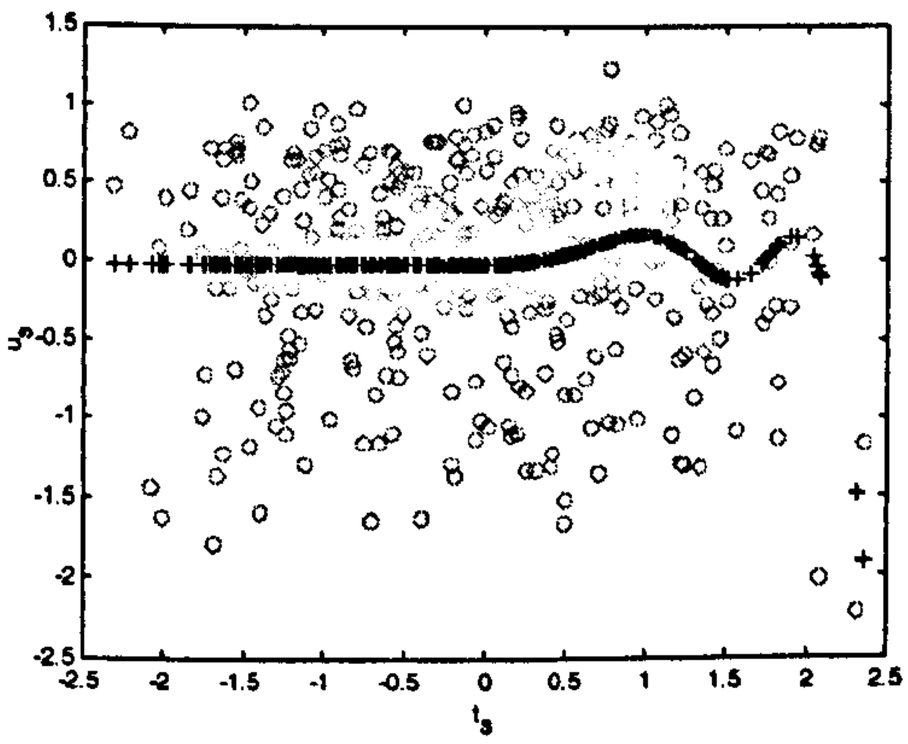

Figure 5.3c : RBFPLS, third latent variable scatter plot.

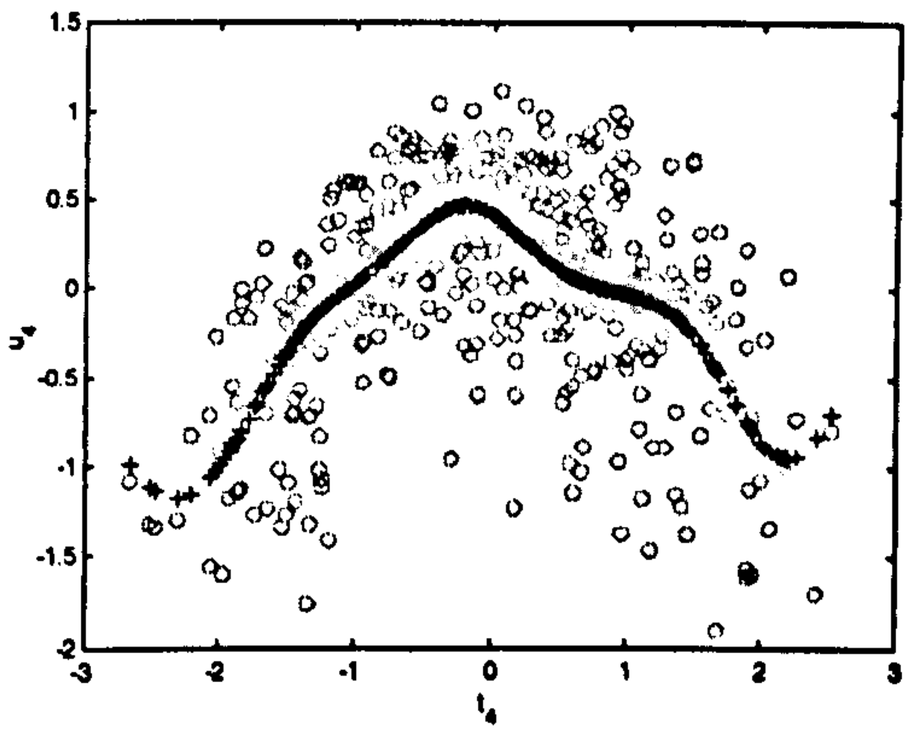

Figure 5.3d : RBFPLS, fourth latent variable scatter plot.

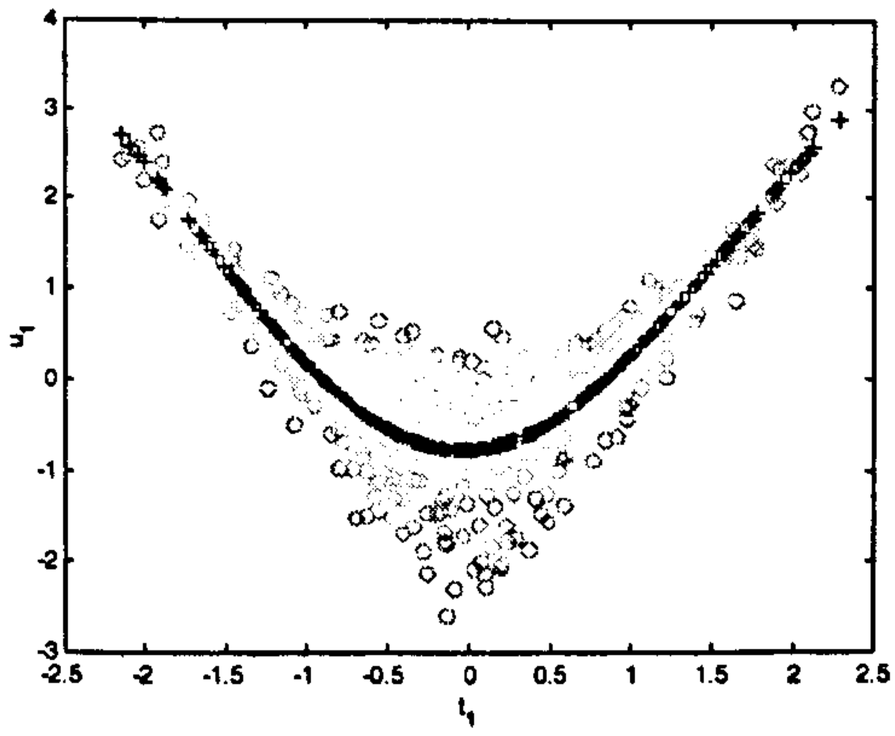

Figure 5.4a : Error Based RBFPLS, first latent variable scatter plot.

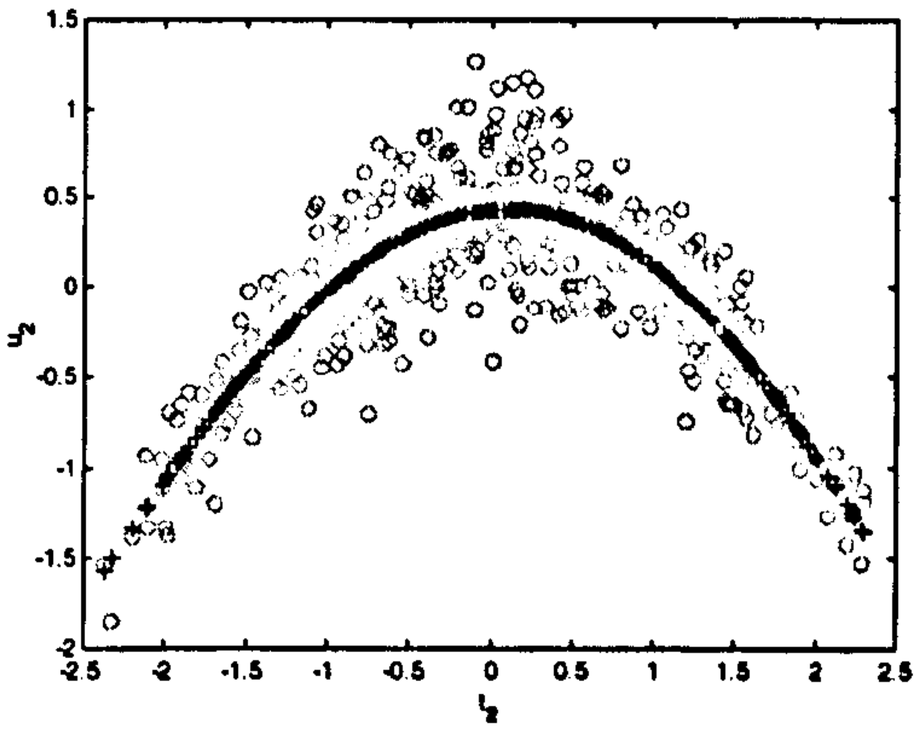

Figure 5.4b : Error Based RBFPLS, second latent variable scatter plot.

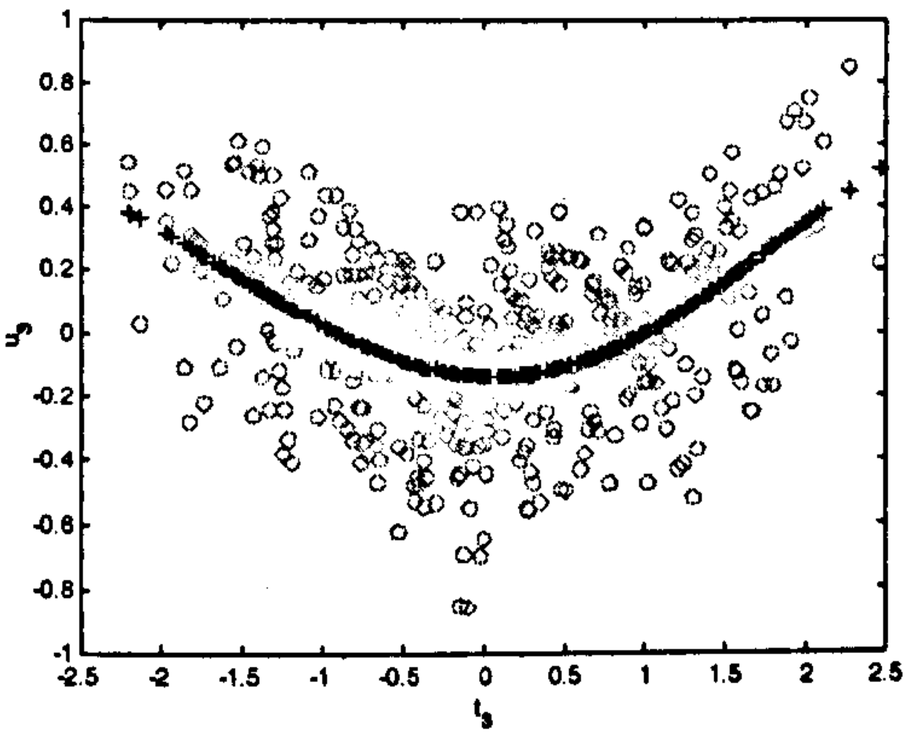

Figure 5.4c : Error Based RBFPLS, third latent variable scatter plot.

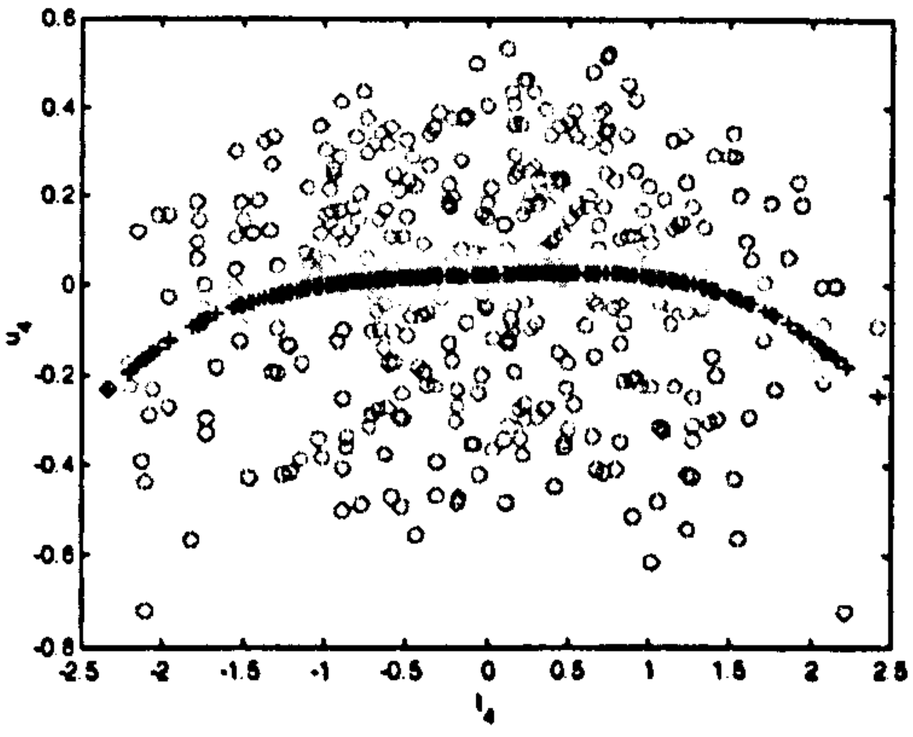

Figure 5.4d : Error Based RBFPLS, fourth latent variable scatter plot.

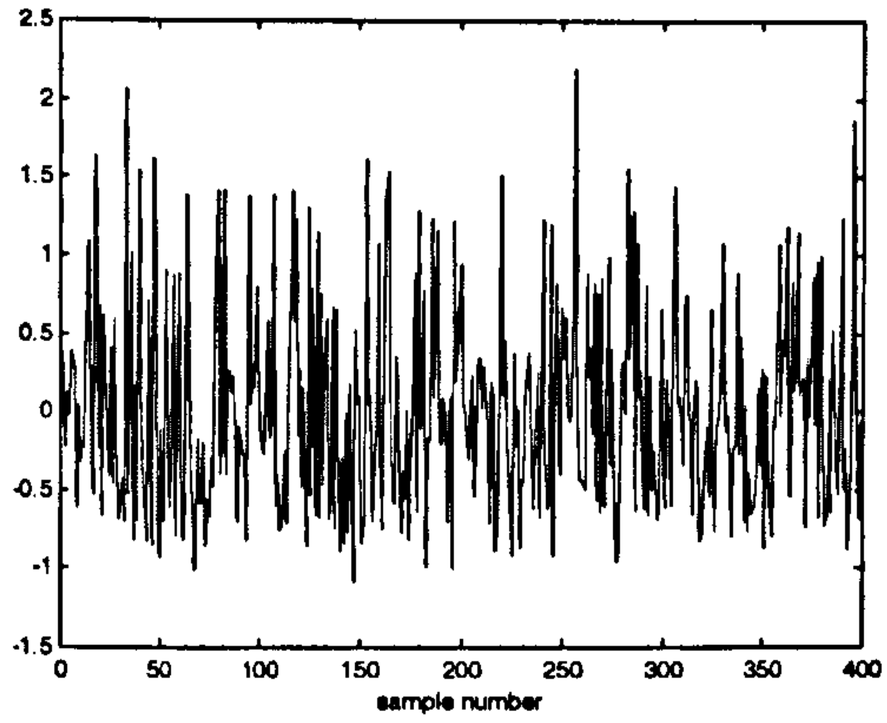

Figure 5.5a : NNPLS output scores residual plot, 2nd latent variable.

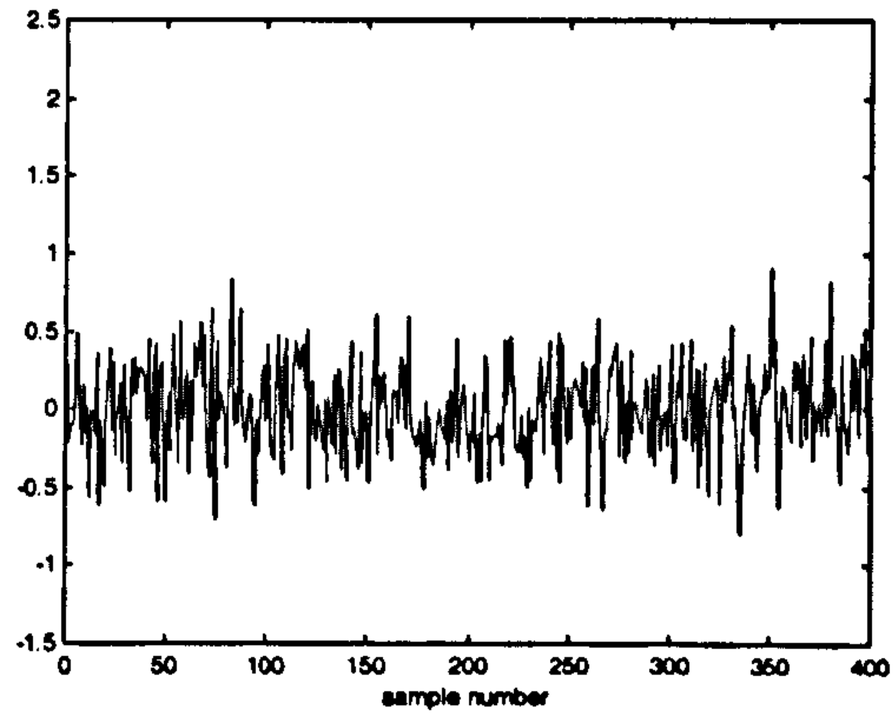

Figure 5.5b : Error Based NNPLS output scores residual plot, 2nd latent variable.

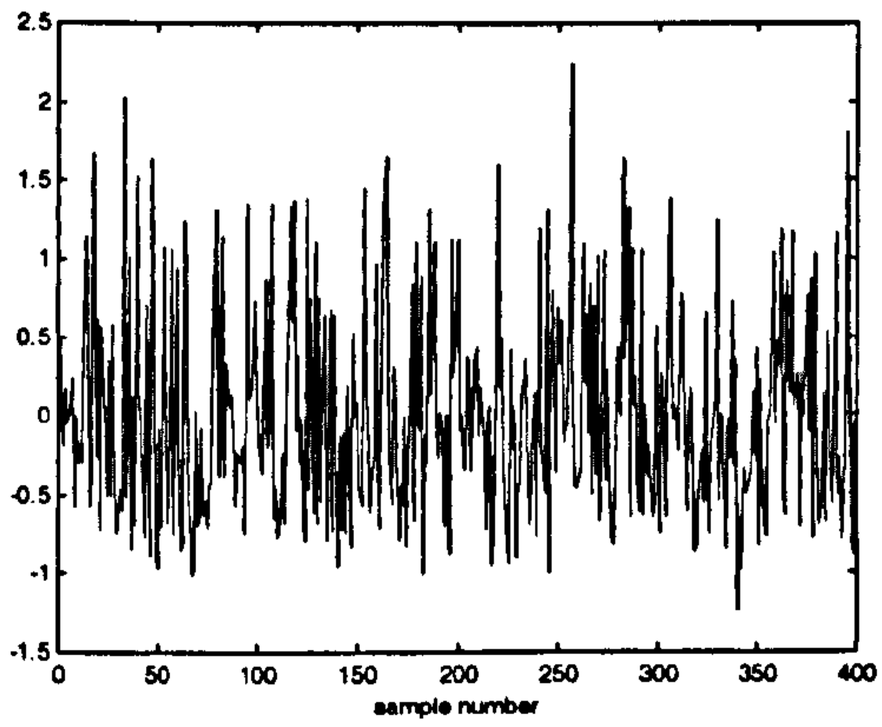

Figure 5.5c : RBFPLS output scores residual plot, 2nd latent variable.

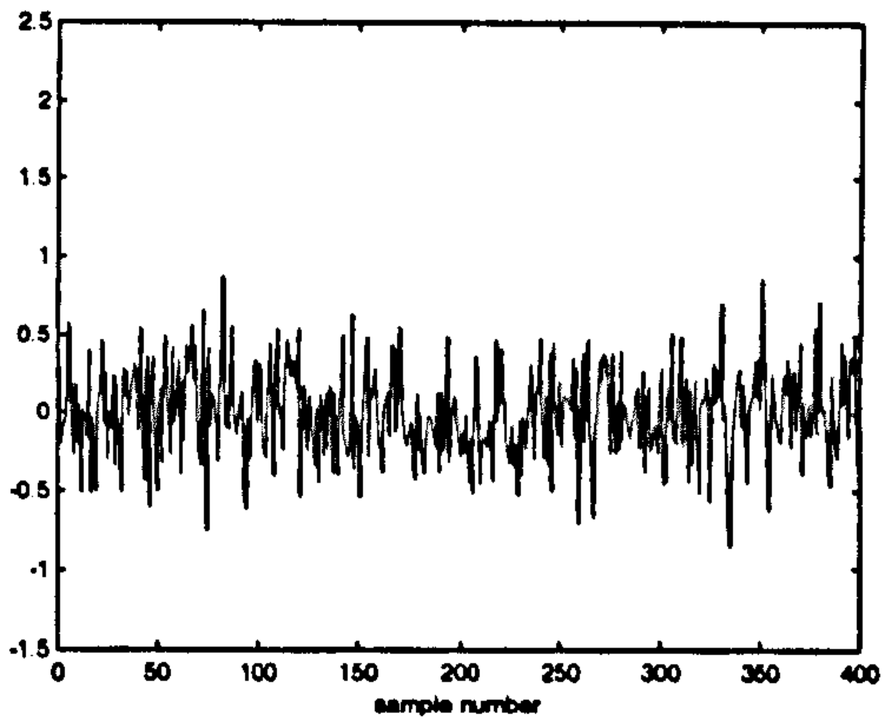

Figure 5.5d : Error Based RBFPLS output scores residual plot, 2nd latent variable.

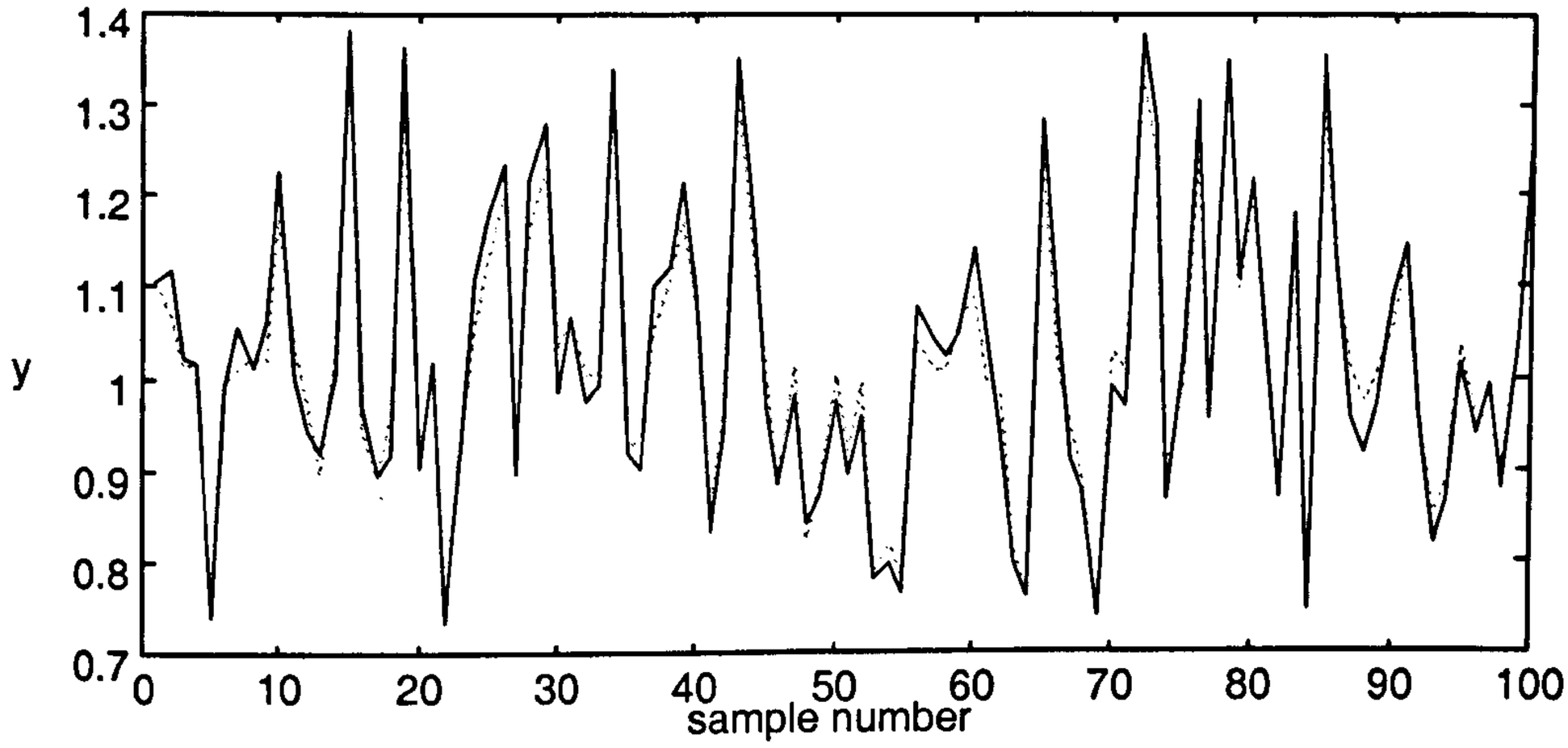

Figure 5.6a : actual (dotted light) versus predicted (full dark) output values for best Error Based NNPLS model using four latent variables.

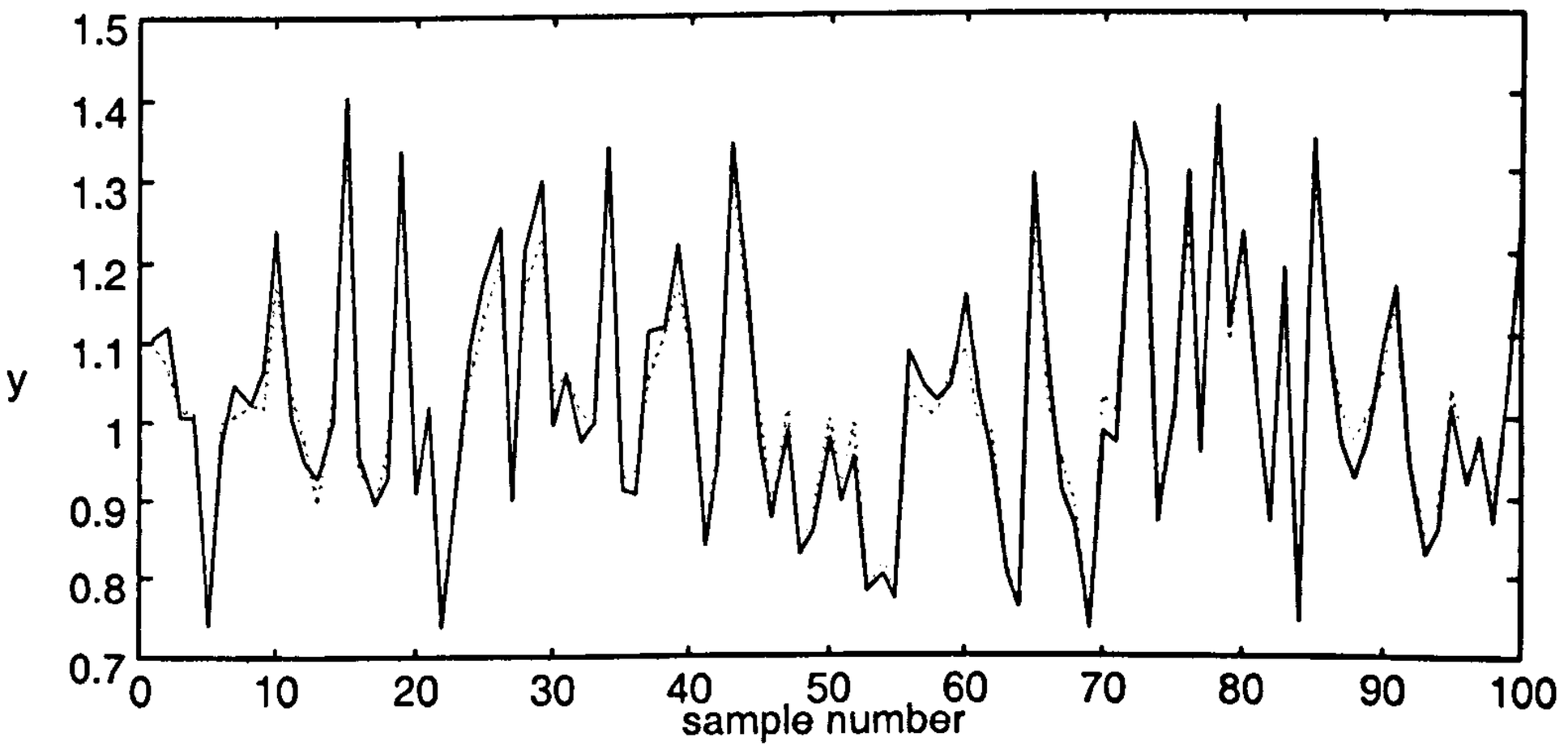

Figure 5.5b : actual (dotted light) versus predicted (full dark) output values for best Error Based RBFPLS model using three latent variables

Model	Neurones	Latent Variables	**MSPE**
net_1000	3	2	1.0627E-02
net_2500	9	1	3.8555E-02
up_net_1000	10	2	3.8392E-03
up_net_2500	11	4	**1.6931E-03**

Table 5.5a : MSPE for the pH data for the NNPLS algorithm (testing data).

Model	Centres	Latent Variables	**MSPE**
rbf_0.5	4	3	4.5766E-03
rbf_1.0	4	3	5.5004E-03
rbf_1.5	8	3	5.5566E-03
rbf_2.0	4	4	3.3008E-03
up_rbf_0.5	8	4	**2.2826E-04**
up_rbf_1.0	7	4	2.5269E-04
up_rbf_1.5	6	4	2.6572E-04
up_rbf_2.0	5	4	2.9767E-04

Table 5.5b : MSPE for the pH data for the RBFPLS algorithm (testing data).

Table 5.5 : Performance comparison between the different neural network PLS algorithms.

LV#	*1*	*2*	*3*	*4*
Linear PLS	3.8070E-02	5.3955E-03	3.0545E-03	**3.0460E-03**
Wold QPLS	1.3345E-02	5.9654E-03	**2.6959E-03**	2.7075E-03
EB QPLS	8.6075E-04	7.7841E-04	**6.6343E-04**	6.6569E-04
up_net_2500	1.7430E-03	1.6937E-03	1.6934E-03	**1.6931E-03**
up_rbf_2.0	1.2006E-03	2.9128E-04	2.3416E-04	**2.2826E-04**

Table 5.6 : performance comparison between the different non-linear PLS algorithms.

	X-block		***Y***-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	10.7095	10.7095	90.3678	90.3678
2	24.4008	35.1103	0.0102	90.378
3	64.7892	99.8995	-0.0008	90.3772
4	0.1005	100	-0.0004	90.3768

Table 5.7a : Percentage of Variability Explained - Error Based NNPLS Algorithm.

	X-block		***Y***-block	
LV	Single LV	Cumulative	Single LV	Cumulative
1	14.6674	14.6674	98.0479	98.0479
2	75.3645	90.032	1.4387	99.4866
3	9.9442	99.9761	0.095	99.5816
4	0.0239	100	0.0035	99.5851

Table 5.7b : Percentage of Variability Explained - Error Based RBFPLS Algorithm.

Table 5.7 : Comparison of Percentage of Variability Explained for the Error Based Neural Network and Radial Basis Function PLS Algorithms for the pH system.

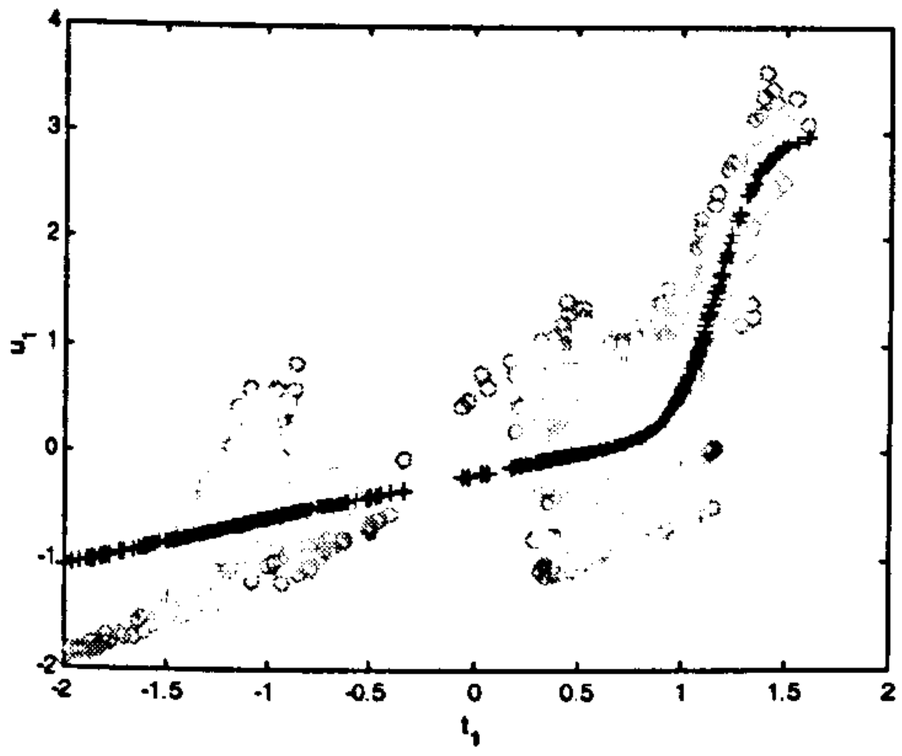

Figure 5.7a : NNPLS, first latent variable scatter plot.

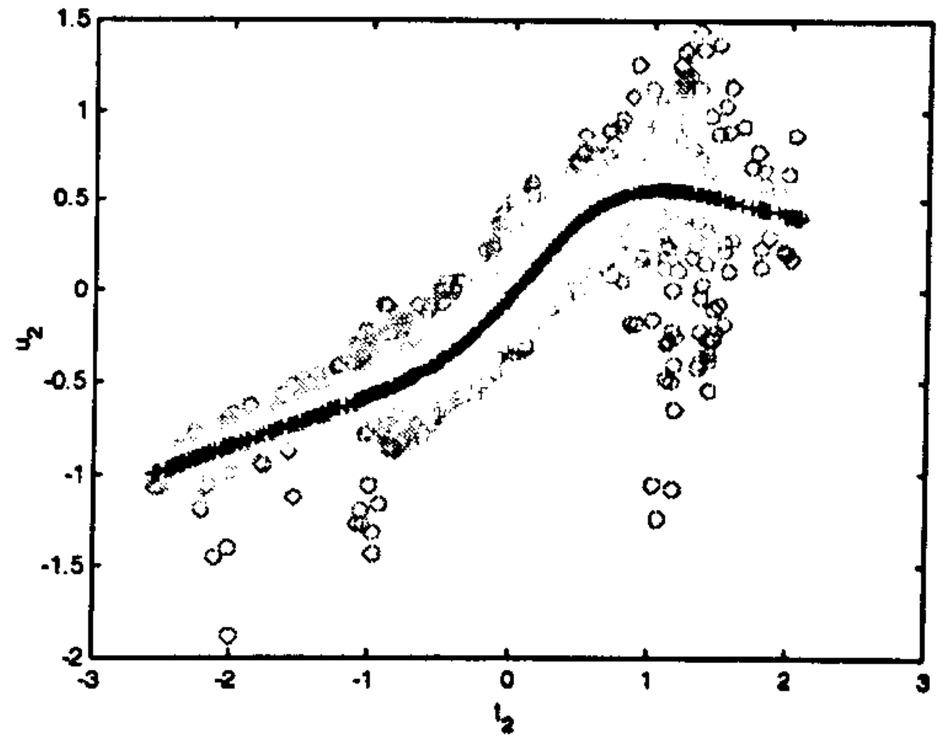

Figure 5.7b : NNPLS, second latent variable scatter plot.

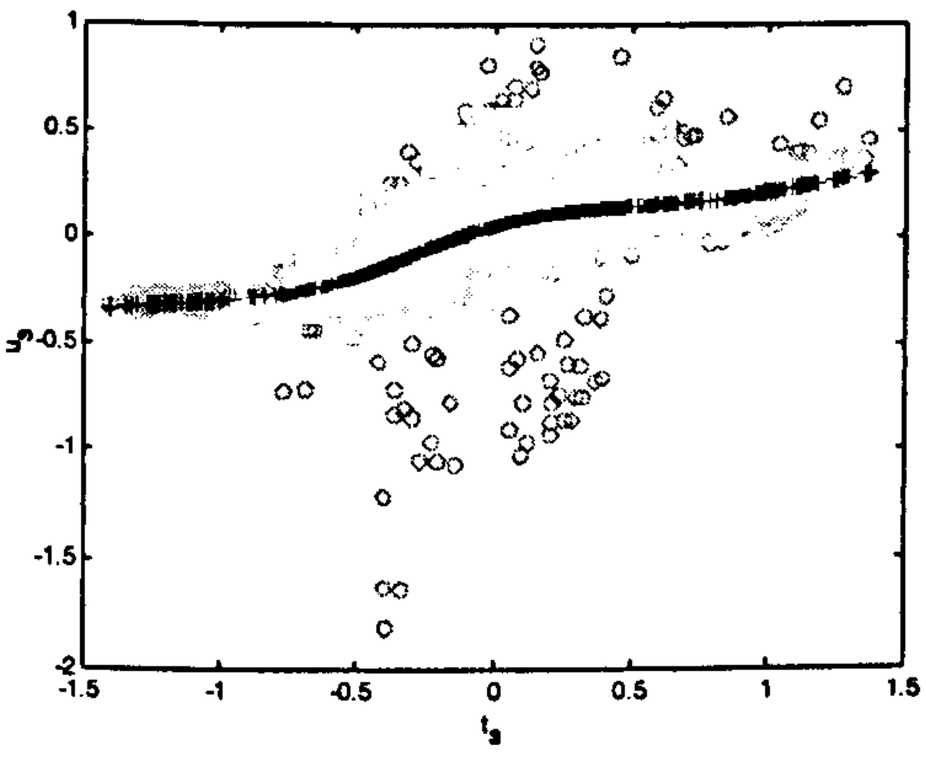

Figure 5.7c : NNPLS, third latent variable scatter plot.

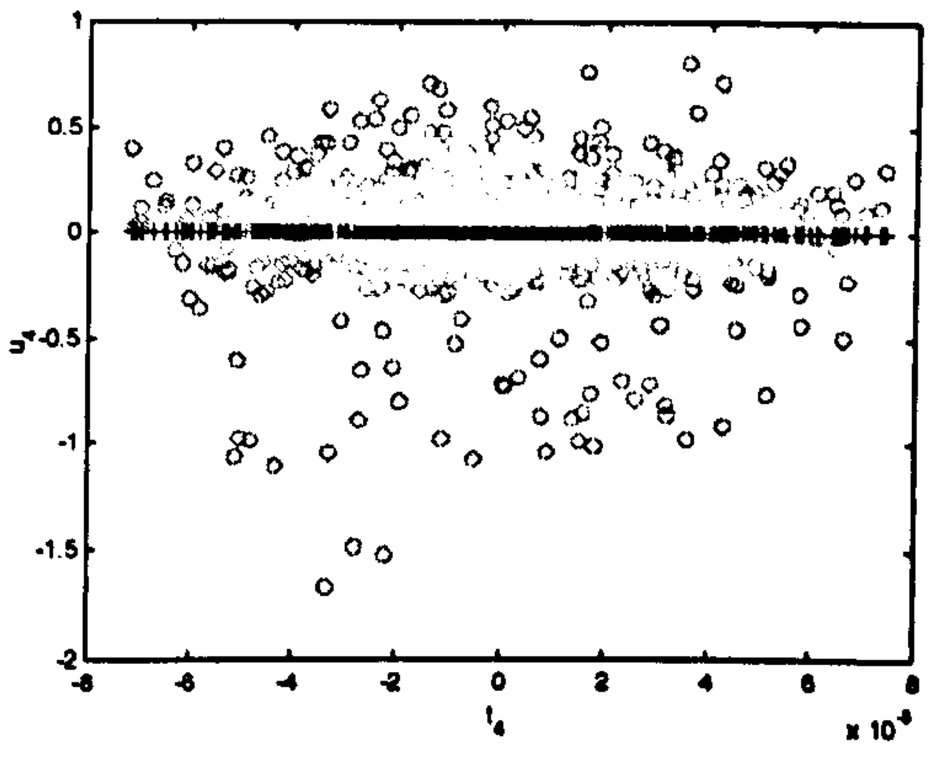

Figure 5.7d : NNPLS, fourth latent variable scatter plot.

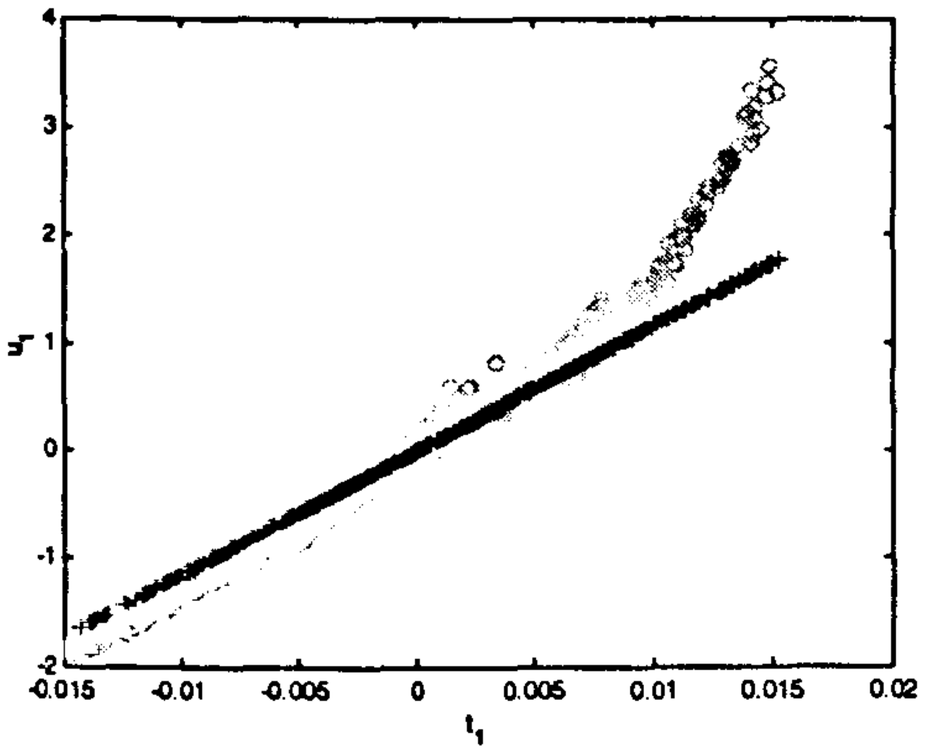

Figure 5.8a : Error Based NNPLS, first latent variable scatter plot.

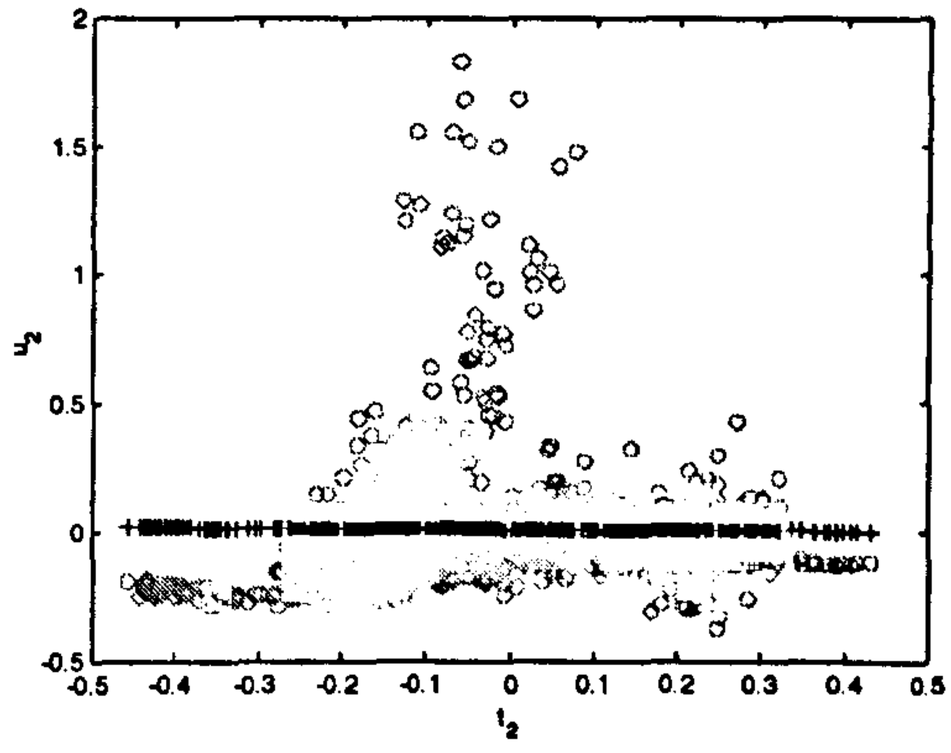

Figure 5.8b : Error Based NNPLS, second latent variable scatter plot.

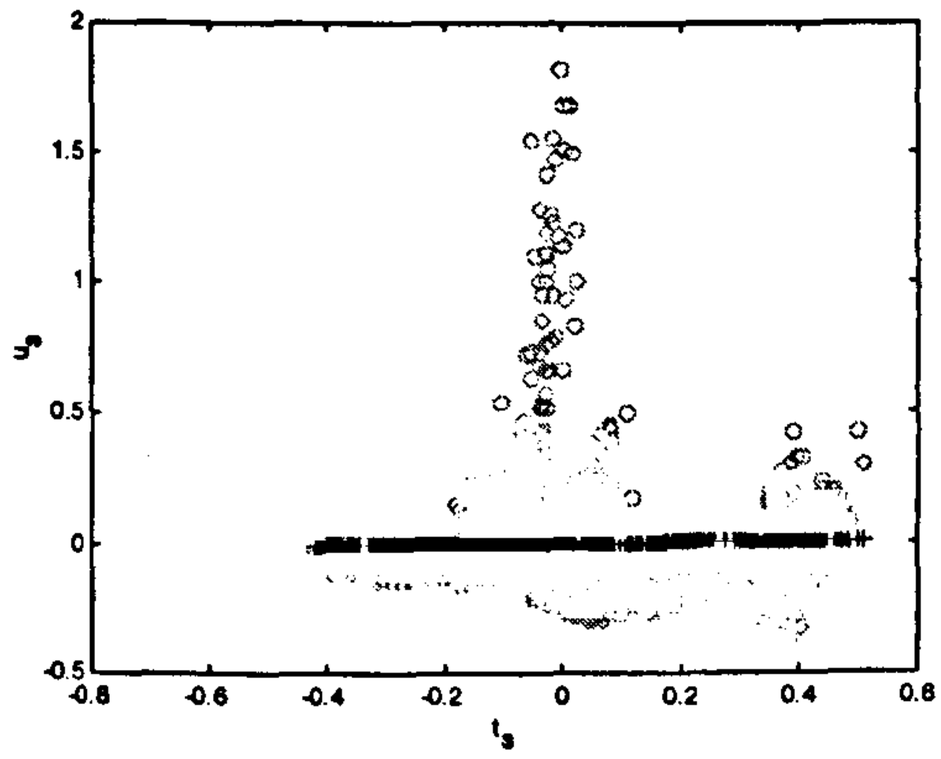

Figure 5.8c : Error Based NNPLS, third latent variable scatter plot.

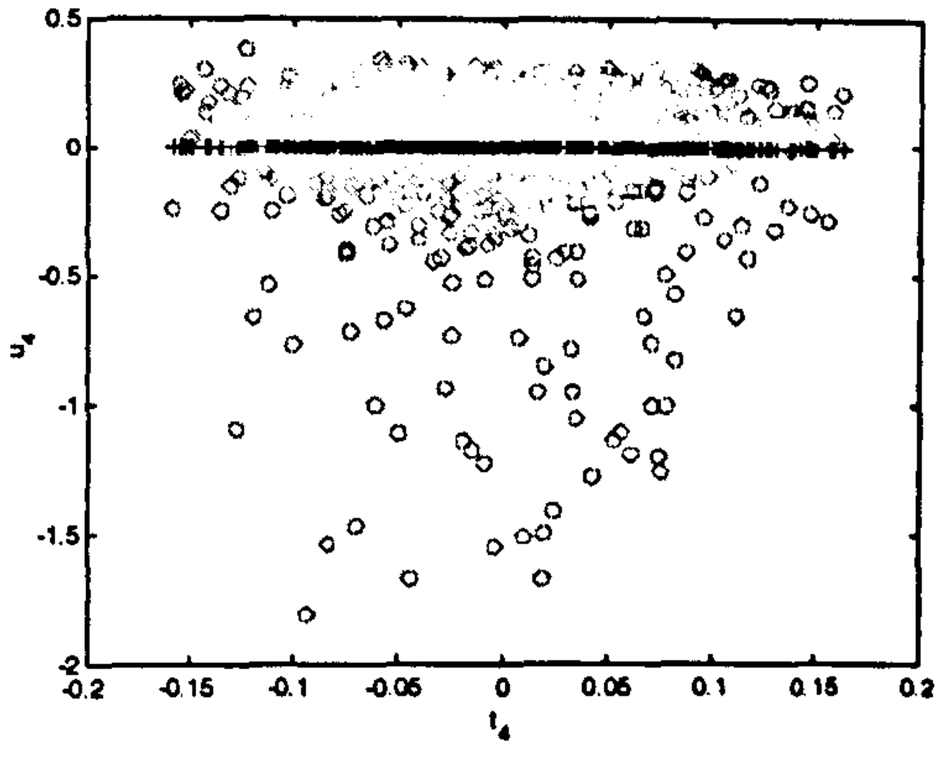

Figure 5.8d : Error Based NNPLS, fourth latent variable scatter plot.

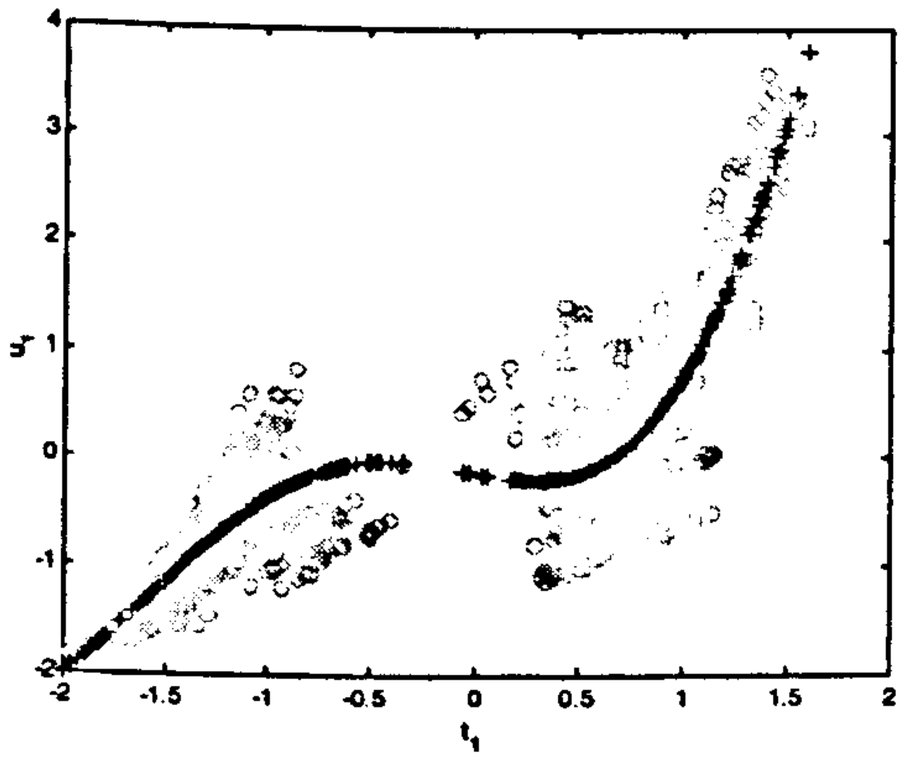

Figure 5.9a : RBFPLS, first latent variable scatter plot.

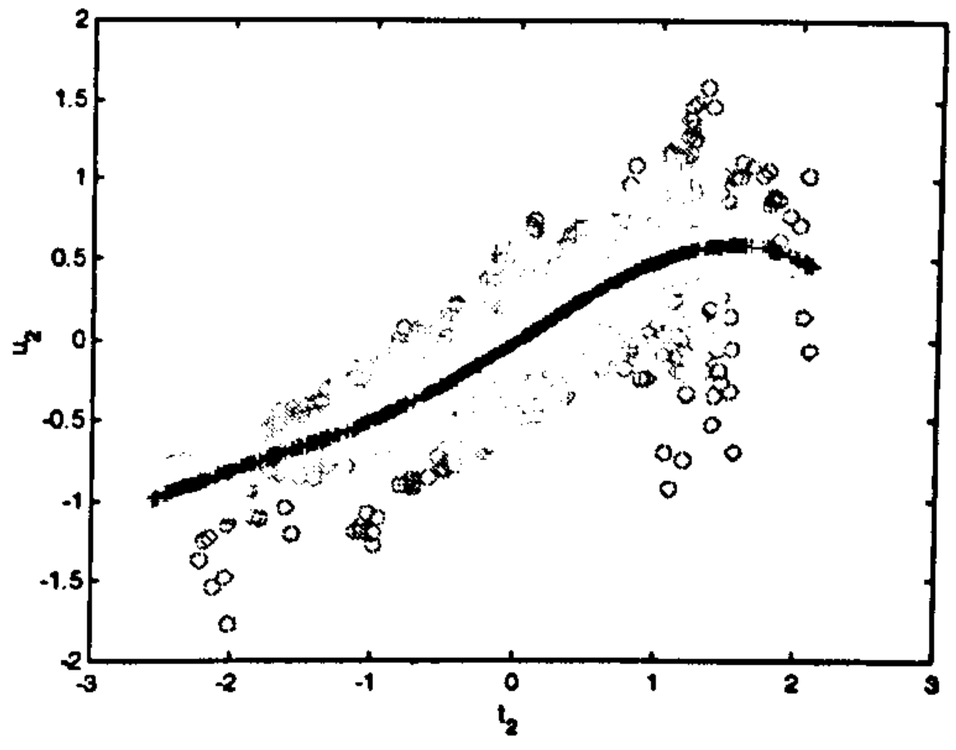

Figure 5.9b : RBFPLS, second latent variable scatter plot.

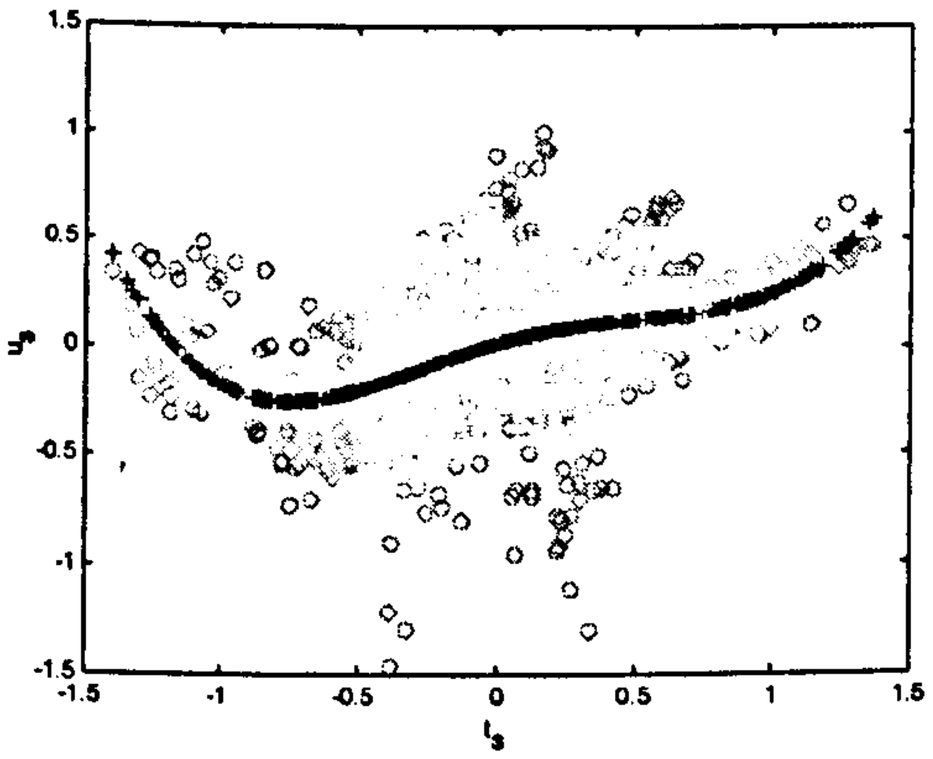

Figure 5.9c : RBFPLS, third latent variable scatter plot.

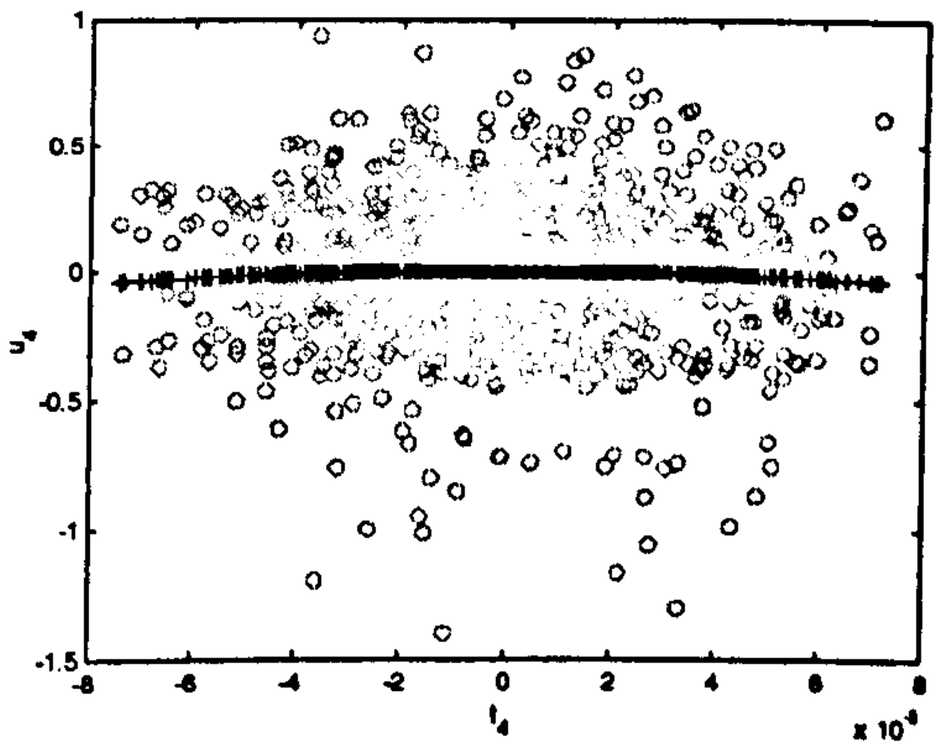

Figure 5.9d : RBFPLS, fourth latent variable scatter plot.

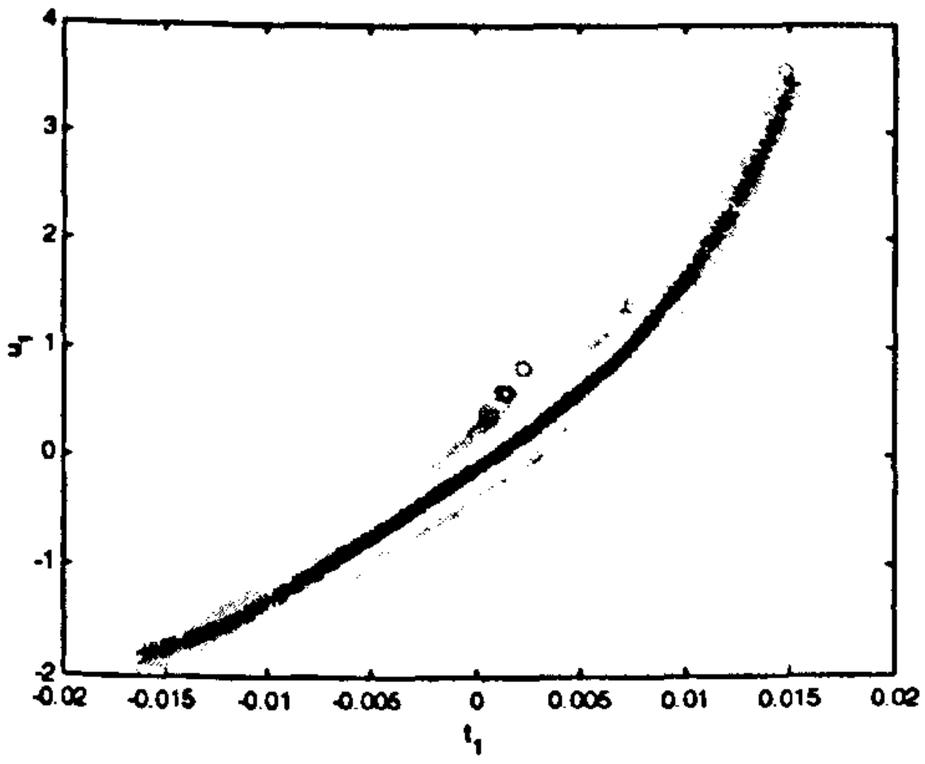

Figure 5.10a : Error Based RBFPLS, first latent variable scatter plot.

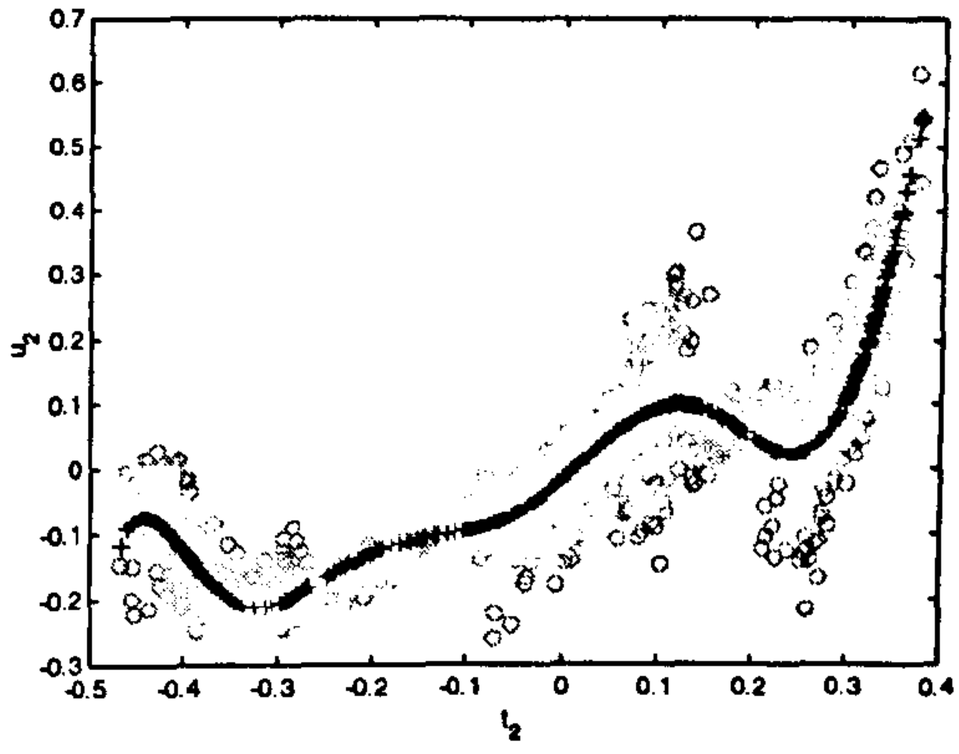

Figure 5.10b : Error Based RBFPLS, second latent variable scatter plot.

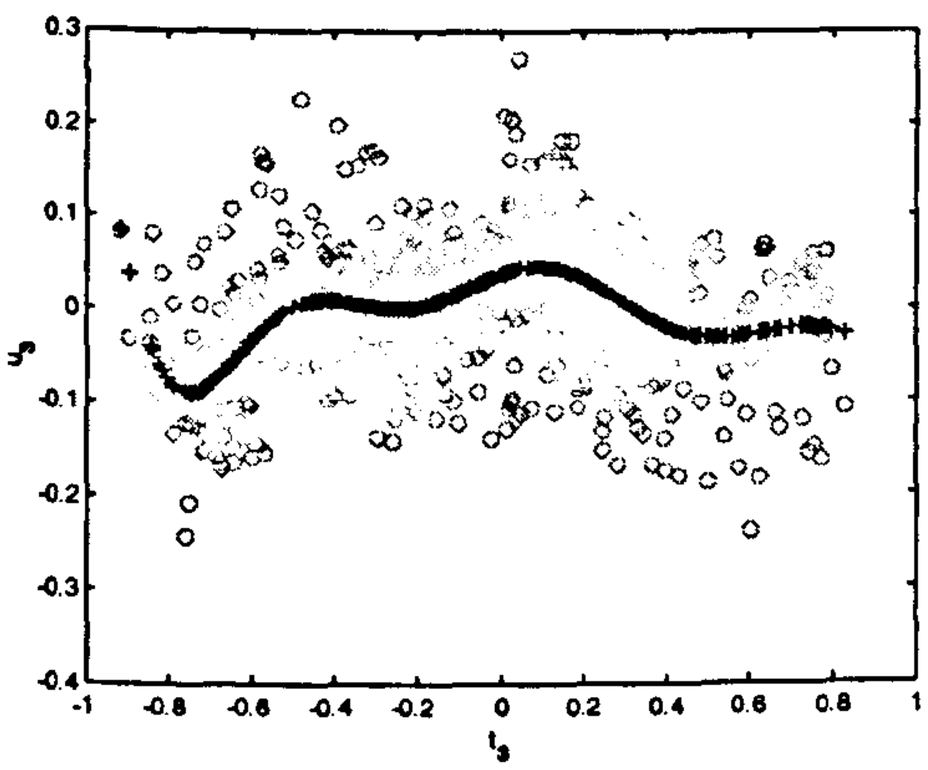

Figure 5.10c : Error Based RBFPLS, third latent variable scatter plot.

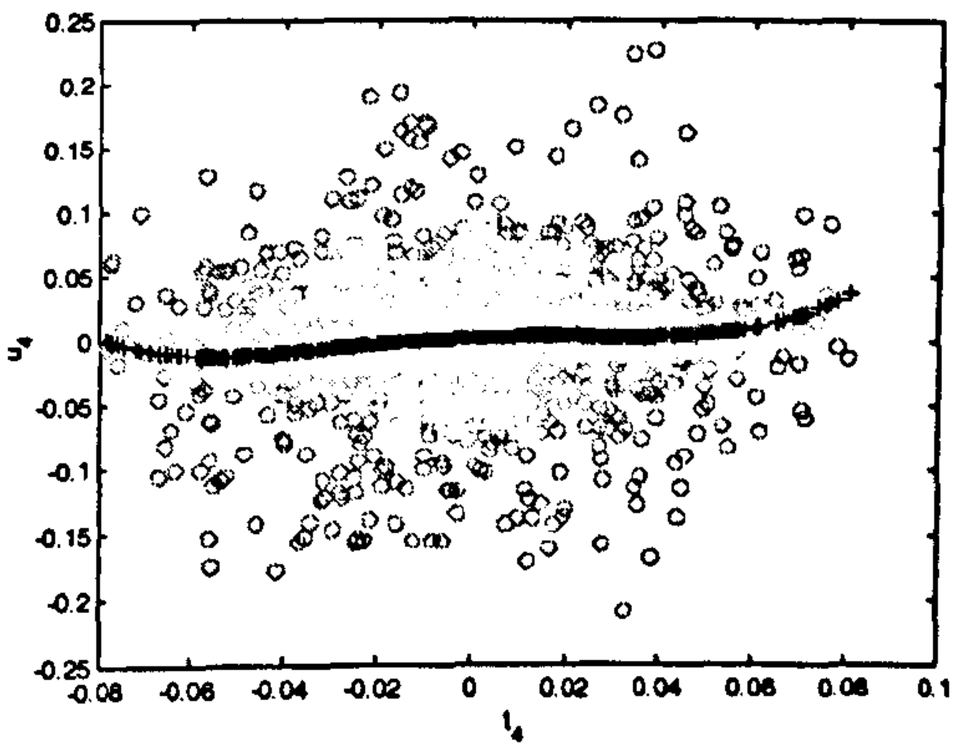

Figure 5.10d : Error Based RBFPLS, fourth latent variable scatter plot.

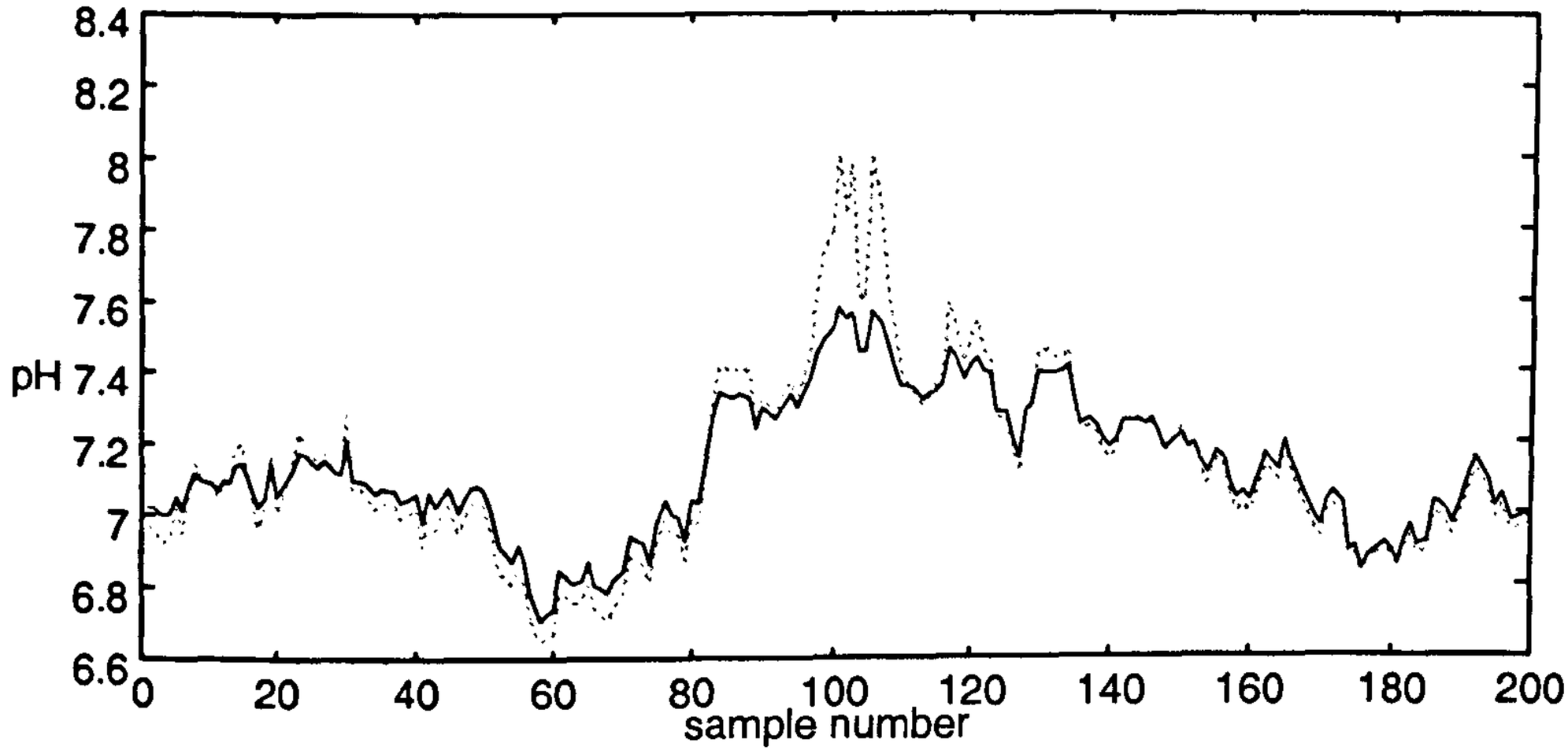

Figure 5.11a : actual (dotted light) versus predicted (full dark) output values for best Error Based NNPLS model using four latent variables.

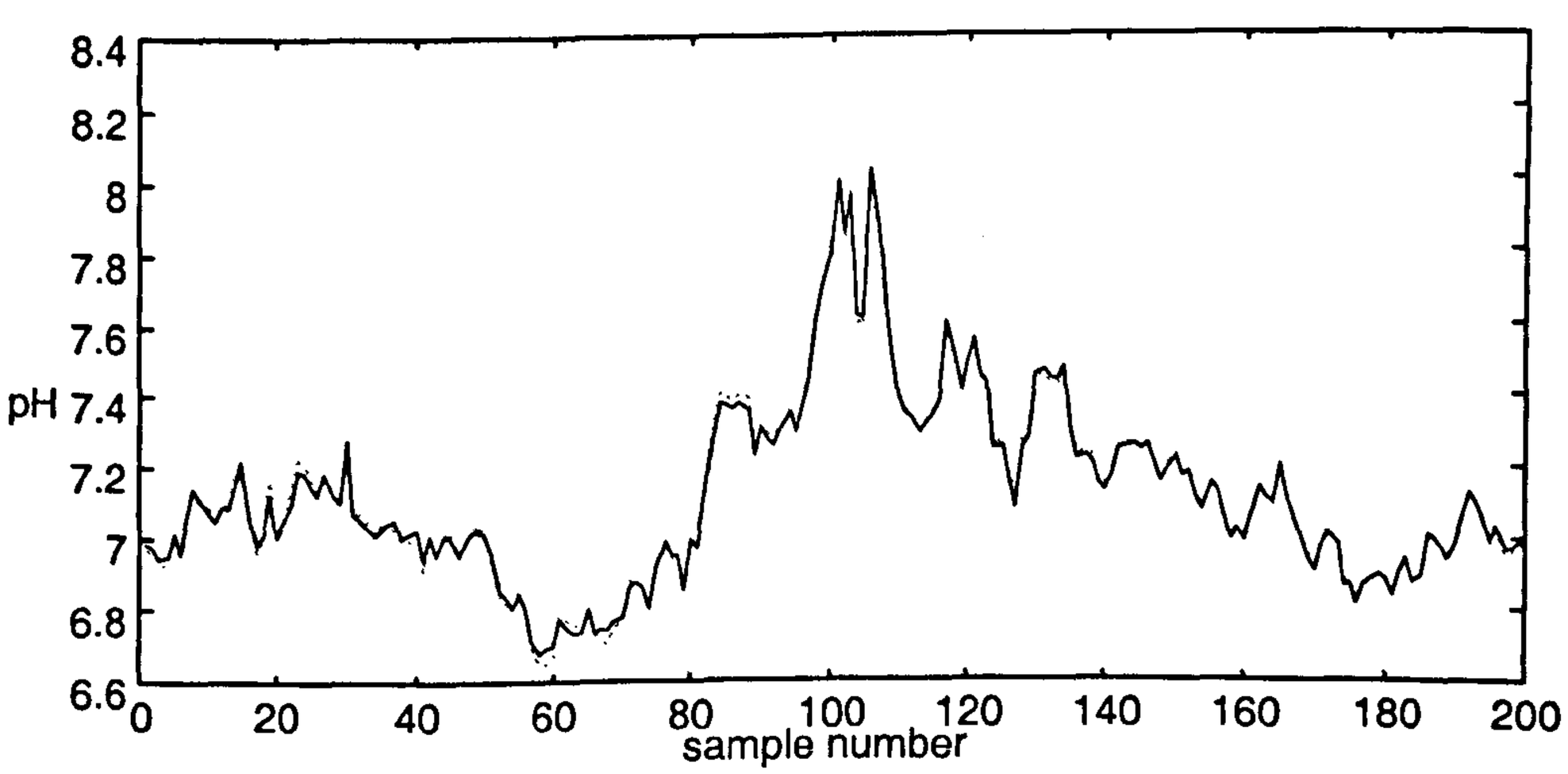

Figure 5.11b : actual (dotted light) versus predicted (full dark) output values for best Error Based RBFPLS model using four latent variables.

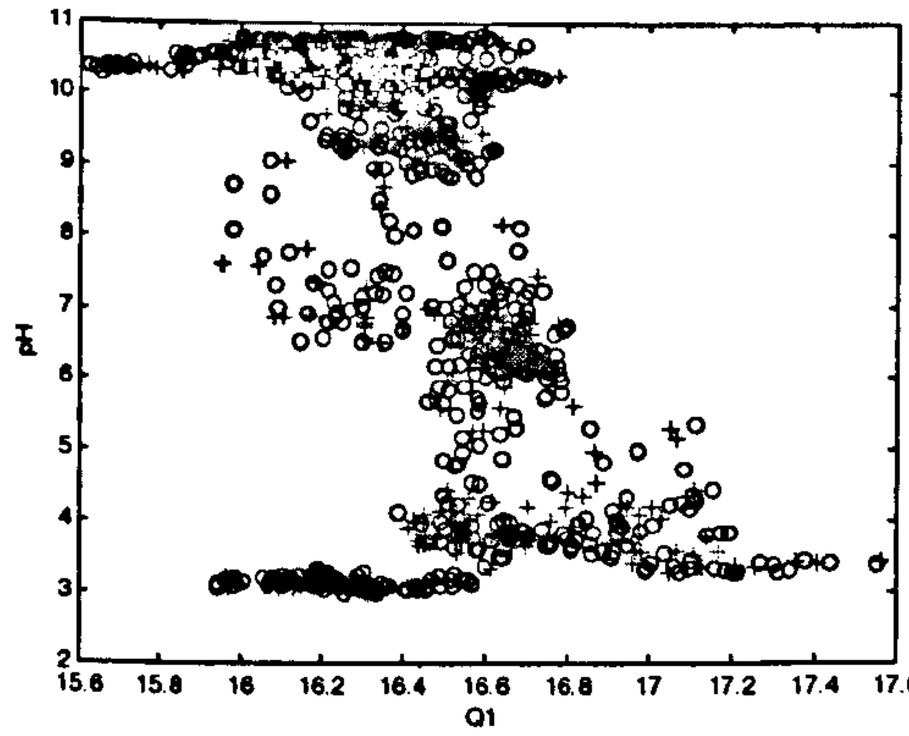

Figure 5.12a : inlet flowrate Q1 versus pH value on outlet stream (light + : training data; dark o : testing data).

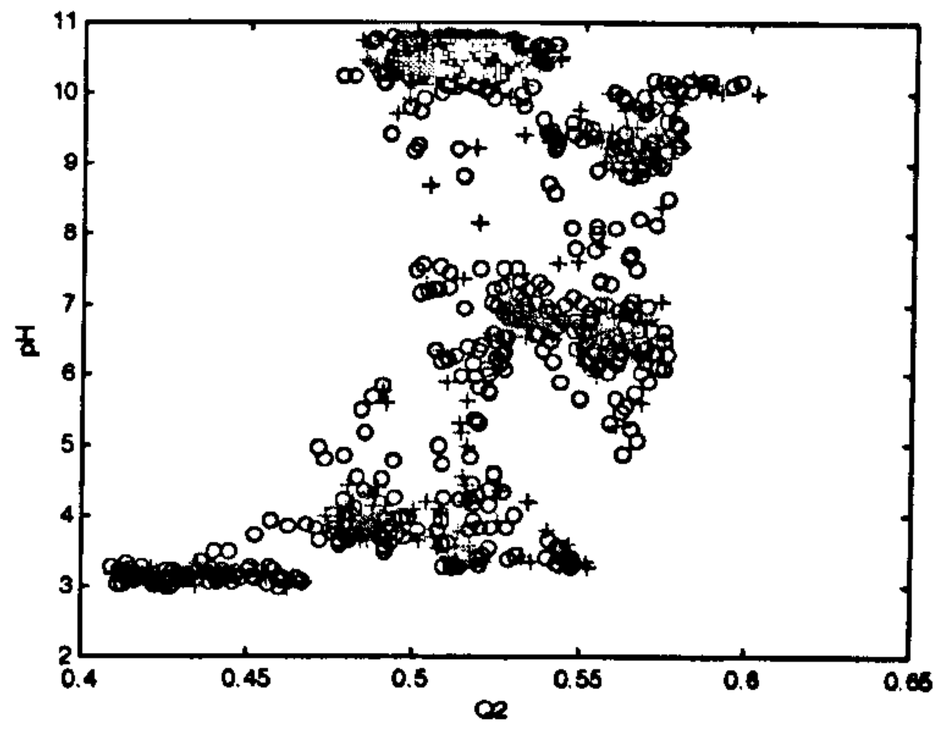

Figure 5.12b : inlet flowrate Q2 versus pH value on outlet stream (light + : training data; dark o : testing data).

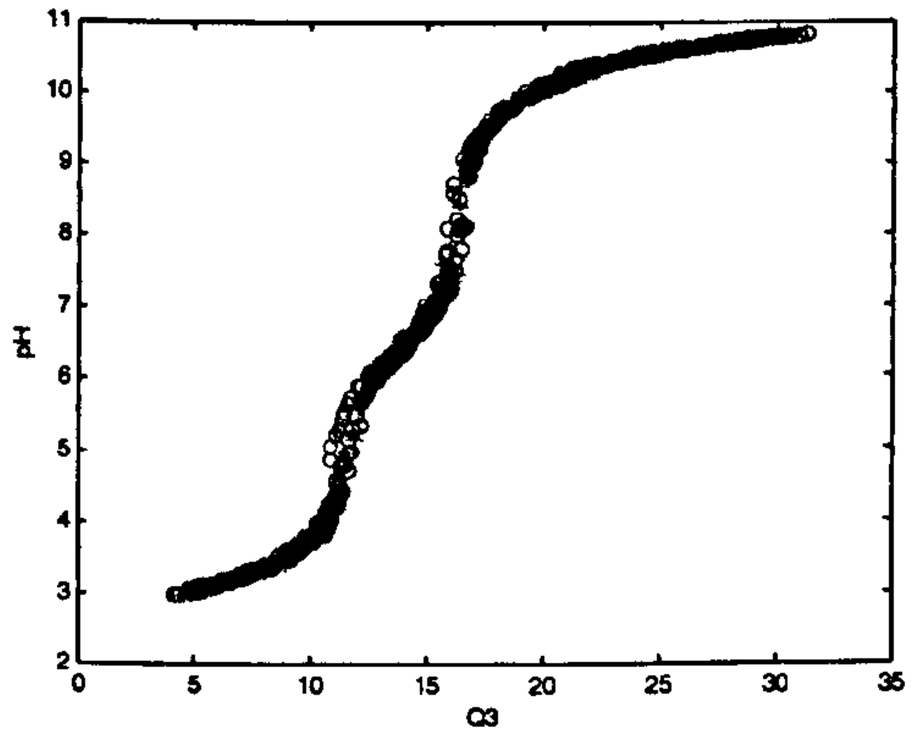

Figure 5.12c : inlet flowrate Q3 versus pH value on outlet stream (light + : training data; dark o : testing data).

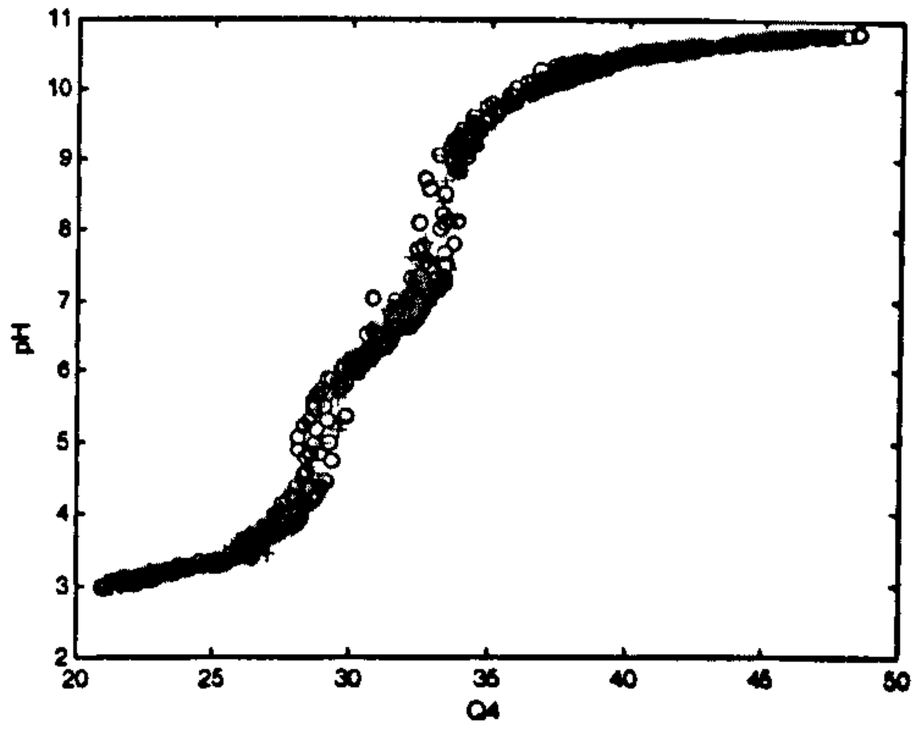

Figure 5.12d : outlet flowrate Q4 versus pH value on outlet stream (light + : training data; dark o : testing data).

Model	Neurones	Latent Variables	MSPE
net_1000	11	4	3.6035E-01
net_2500	9	2	2.9577E-01
up_net_1000	5	2	5.2014E-02
up_net_2500	8	1	1.5932E-02

Table 5.8a : MSPE for the second pH data set for the NNPLS algorithm (testing data).

Model	Centres	Latent Variables	MSPE
rbf_0.5	18	4	7.0625E-02
rbf_1.0	14	4	7.1450E-02
rbf_1.5	15	4	7.2169E-02
rbf_2.0	20	3	7.4487E-02
up_rbf_0.5	21	2	6.3886E-03
up_rbf_1.0	17	2	9.8937E-03
up_rbf_1.5	13	4	2.7474E-02
up_rbf_2.0	20	3	3.2549E-02

Table 5.8b : MSPE for the second pH data set for the RBFPLS algorithm (testing data).

Table 5.8 : Performance comparison between the different neural network PLS algorithms.

LV#	*1*	*2*	*3*	*4*
Wold QPLS	4.8572E-1	4.6596E-1	4.6393E-1	4.6746E-1
EB QPLS	3.7259E-1	3.2691E-1	3.1886E-1	3.1898E-1
up_net_2500	1.5931E-2	1.5940E-2	1.5940E-2	1.5941E-2
up_rbf_2.0	6.9493E-3	6.3885E-3	6.5832E-3	6.7263E-3

Table 5.9 : performance comparison between the different non-linear PLS algorithms.

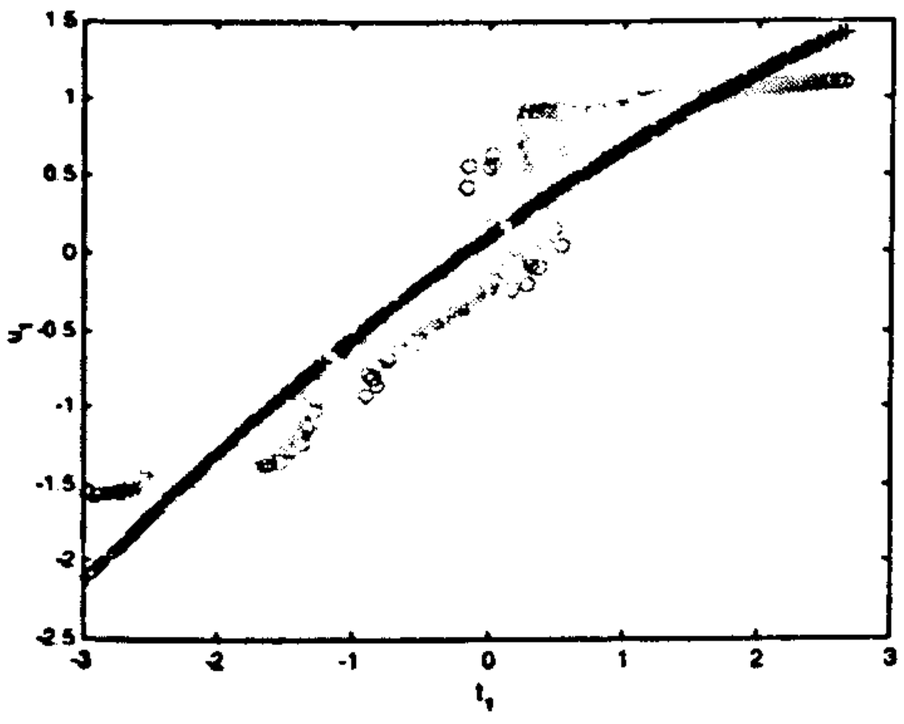

Figure 5.13a :Wold QPLS, first latent variables scatter plot.

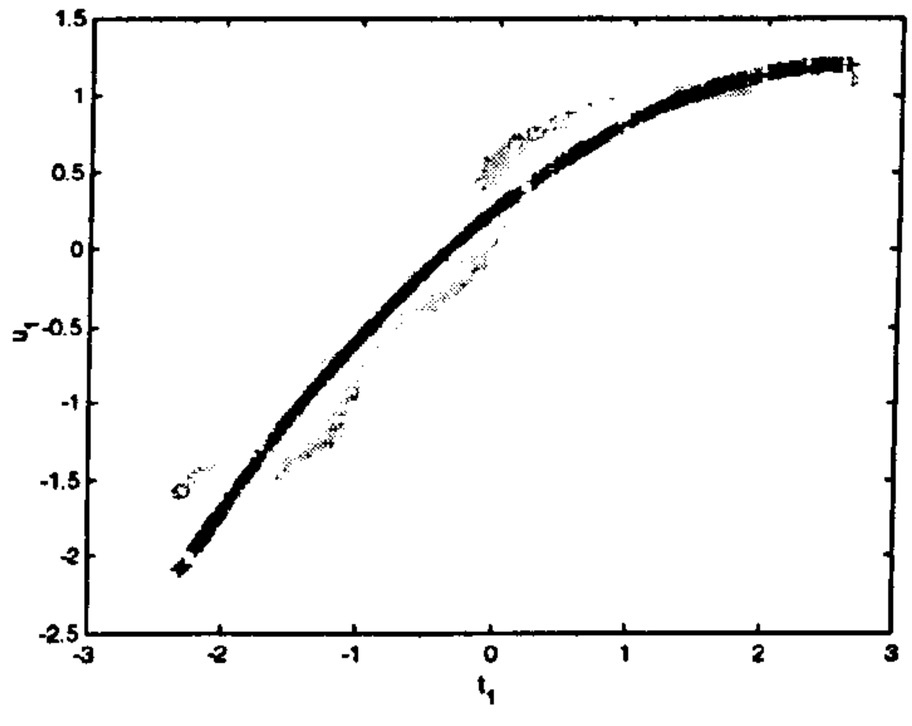

Figure 5.13b :Error Based QPLS, first latent variables scatter plot.

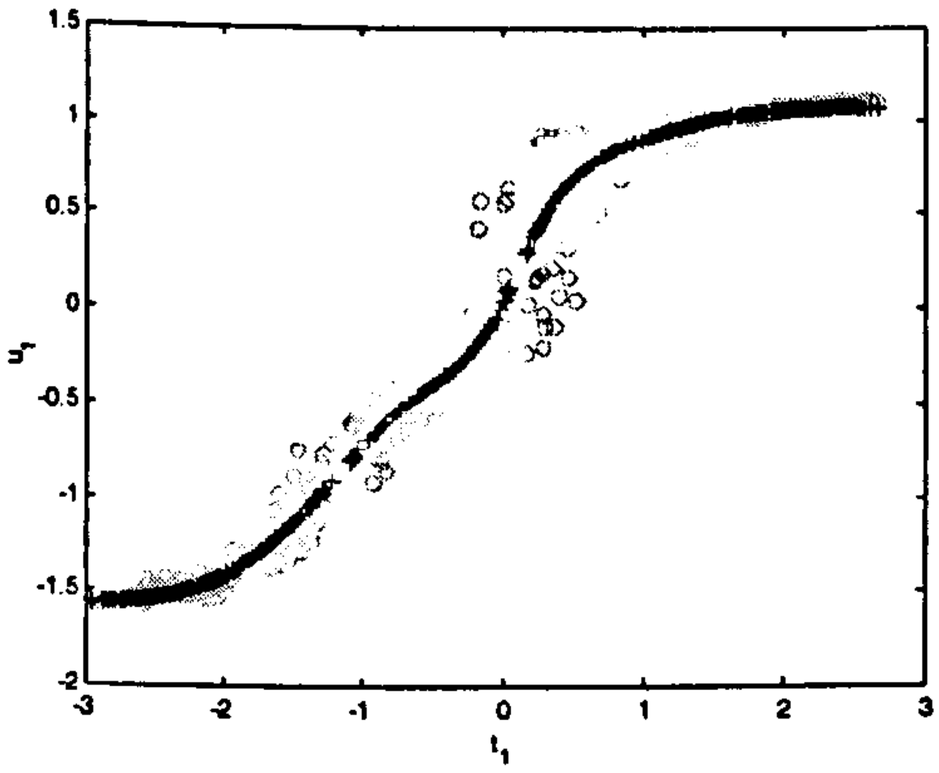

Figure 5.13c : NNPLS, first latent variables scatter plot.

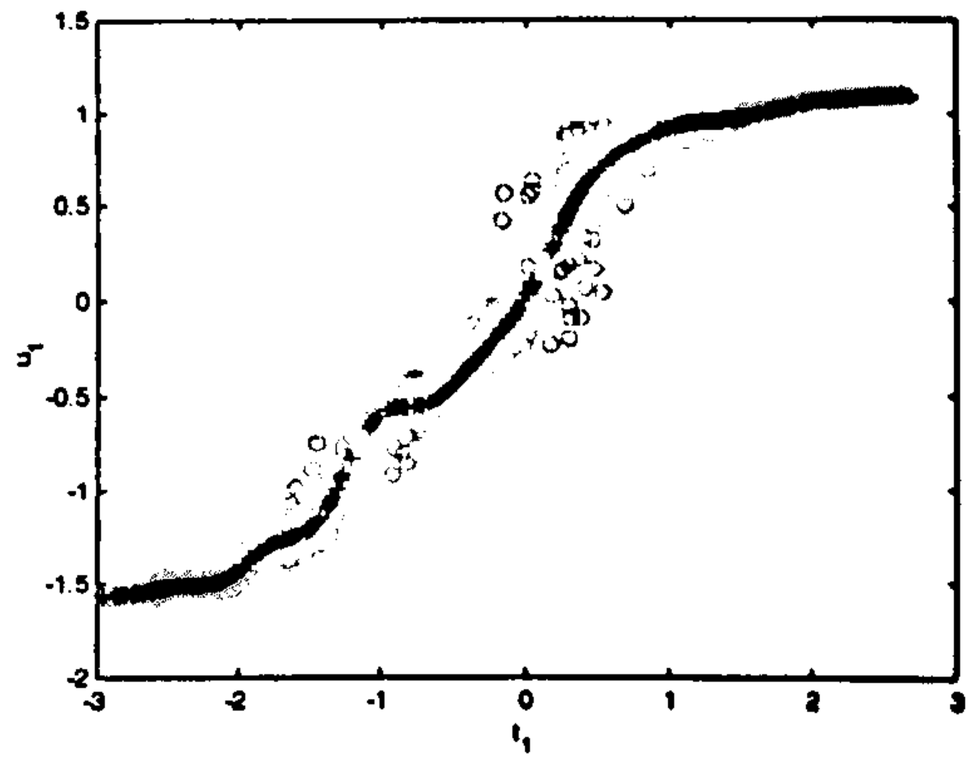

Figure 5.13d : RBFPLS, first latent variables scatter plot.

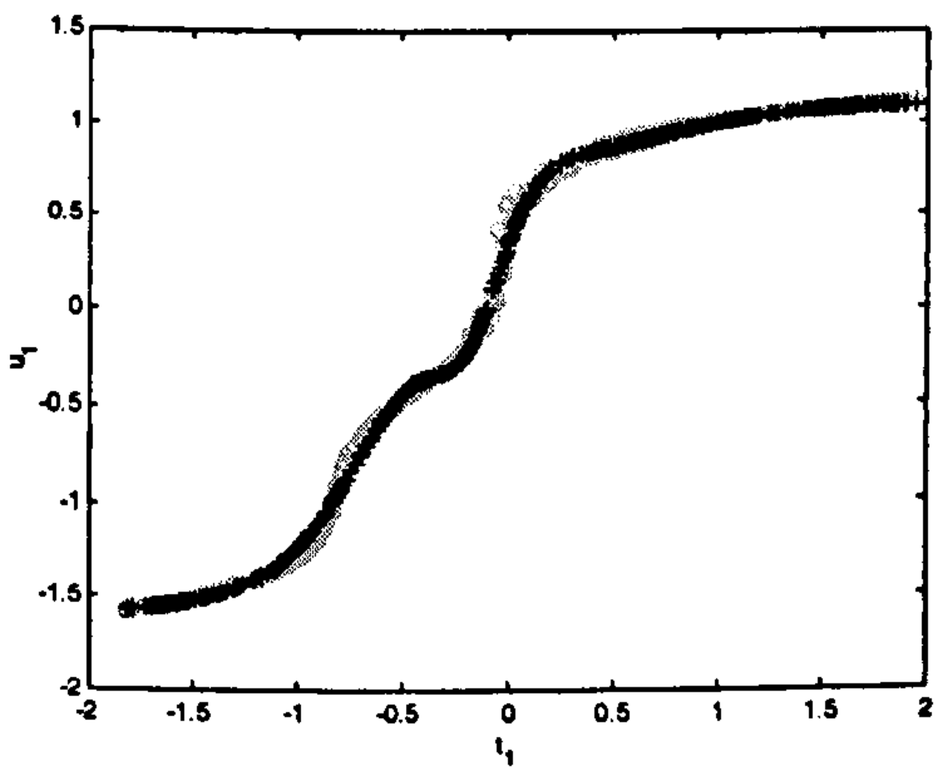

Figure 5.13e :Error Based NNPLS, first latent variables scatter plot.

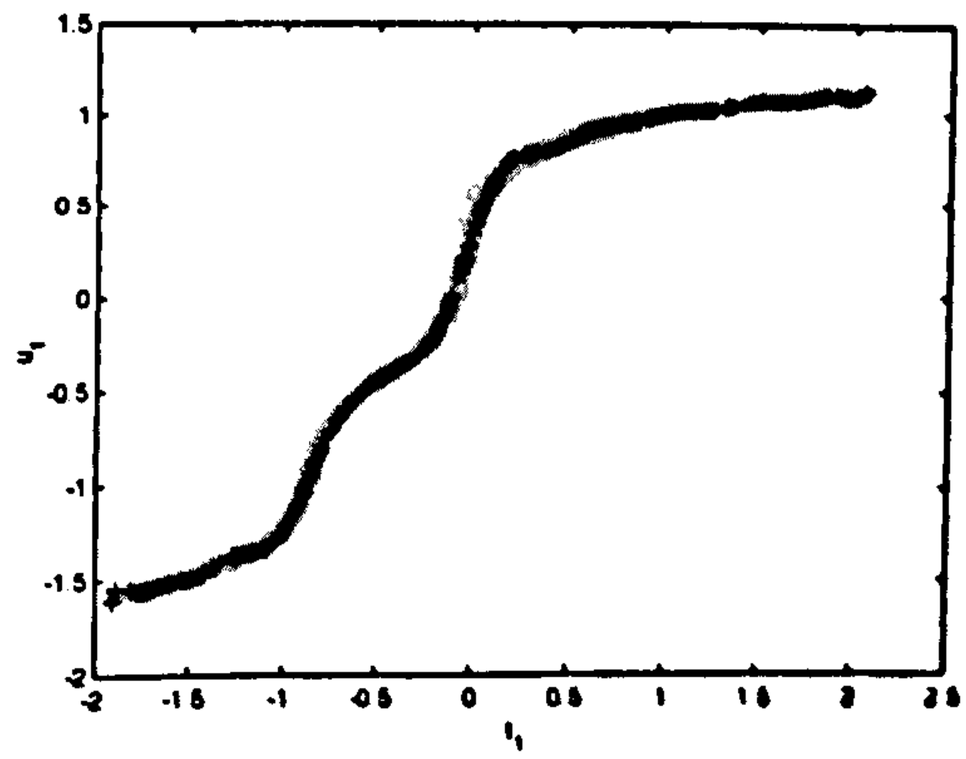

Figure 5.13f :Error Based RBFPLS, first latent variables scatter plot.

6. Conclusions and Further Work

6.1 Conclusions

The objective of the research carried out was the study and development of non-linear regression tools based upon multivariate statistical techniques for empirical modelling. The motivation for developing empirical models of chemical processes was established in Chapter 1.

Linear regression techniques such as Multiple Linear Regression (MLR) are widely used for modelling data (process data and spectral data) collected on real chemical processes. Measurements collected on modern chemical manufacturing plants exhibit strong variable correlations and often non-linear behaviour. Linear regression techniques based upon a least squares methodology are known to be affected by measurement noise and variable correlations, whilst projection based regression techniques such as Principal Component Regression (PCR) and Projection to Latent Structures or Partial Least Squares (PLS) have gained increasing acceptance as linear regression tools because of their tolerance to noise and correlated data structures.

Principal Component Analysis (PCA) is probably one of the oldest projection based multivariate statistical analysis technique, and provides a good illustration of the ideas underpinning projection based multivariate empirical modelling methodologies. It was described in detail in Chapter 2, including its statistical properties and its use as a tool for multivariate data analysis. Similarly, Projection to Latent Structures (PLS) was introduced in Chapter 3, again highlighting its numerical properties and the advantages which can be gained when it is used for modelling collinear and noisy data over the traditional least squares approach. However, being a linear regressive tool, PLS lacks reliability when modelling non-linear data.

Multi-layer Perceptrons (sigmoidal or gaussian neural networks) are used extensively to develop non-linear empirical models. However, it is known that variable collinearities and measurement noise can affect the performance of non-linear regression models developed through applications of these techniques. Furthermore, they are known to be prone to overfitting the data used to

build the model and to rely on optimisation routines which are time consuming and which do not always lead to optimal (or best) solutions. This is particularly true when dealing with highly correlated multivariable data. In this case regularisation features are usually incorporated within the optimisation algorithms to avoid convergence to local minima and hence resulting in a poor model.

An appealing approach for reducing the impact of measurement and process noise, variable correlations, high data dimensionality and data distribution on non-linear empirical modelling is to integrate non-linear features within the linear PLS methodology. This provides a projection based non-linear regression algorithm. A number of different non-linear PLS algorithms have been proposed and were reviewed in Chapter 4. These are usually based on the integration of polynomial expansions, SPLINE functions or neural networks within the inner model of the PLS framework. However when incorporating non-linear features within the linear PLS algorithm it is important that a number of modifications are made to ensure that the PLS algorithm provides a robust non-linear regression method. With respect to this issue, the work of Wold *et al.* (1989) represented a major breakthrough in non-linear PLS modelling. It showed the benefits achieved by using an updating procedure of the parameters of the PLS model when a quadratic polynomial expansion is used as the inner model to provide the non-linear feature within the PLS methodology. Nevertheless, the algorithm proposed appeared rather obscure and complicated and was therefore studied in detail. This analysis led to the development of a new procedure for updating the PLS model parameters.

Both the original algorithm proposed by Wold *et al.* (1989) and the new procedure were discussed in detail in Chapter 4 and compared on the basis of their performances in modelling a non-linear mathematical function and a benchmark pH neutralisation system. In both cases, the new procedure was shown to outperform the original Wold algorithm. In particular it is shown to be capable of changing the projection parameters of the input variables in such a way that the projection of the input variables exhibit less dispersion around the underlying non-linear relationship between the input and the output latent variables. This leads to the enhanced performance of the algorithm used for the development of a non-linear model, in this case, the quadratic interpolation. Furthermore, the new updating procedure is shown to initially extract the information in the input variables which is required to optimise the regression of the output variables. In this way it departs from the original PLS methodology which attempts to achieve a balance between the approximation of the input variables and the regression of the output

variables. It may be argued that this latter feature is a limitation of the proposed non-linear PLS methodology, since the approximation of the input variables is regarded as being less important than the regression of the output variables. However this particular aspect was a consequence of the need to address the development of an improved non-linear PLS algorithm for empirical modelling of input/output relationships between the input (process) variables and the output (quality) measurements. In this context, the regression of the output variables was assumed to be statistically more important than the approximation of the input variables. Furthermore, it is known that PCA can be reliably used for approximation purposes of individual data sets.

In Chapter 5 the new weight updating procedure was extended to include inner models based upon sigmoid neural network and radial basis function networks. It is believed that this represents an important innovation in terms of non-linear PLS modelling. Although several neural network and radial basis function PLS algorithms have been proposed in the literature, either they do not include any updating procedure for the input projection parameters, or they significantly depart from the PLS methodology leading to complex neural network models. In this context, including sigmoid or gaussian networks in the PLS framework does not represent anything new (e.g. Qin *et al.*, 1992; Wilson *et al.*, 1997). However, the use of a sound procedure to update the projection parameters and make the PLS model consistent with the non-linear regressor used to fit the input/output non-linear mapping is innovative. In fact, even though the neural network PLS algorithm proposed by Saunders (1992) performed an updating of the projection parameters, it is observed (§ 5.4.1) that the updating procedure is performed by means of an optimisation routine and is included within the training scheme of the inner neural network model. This unfortunately leads to a non-linear PLS algorithm which significantly departs from the traditional PLS approach.

The advantages arising from the use of PLS in conjunction with neural networks have been demonstrated. In this way a universal approximator, the sigmoid or gaussian network, is used to provide a non-linear mapping between each pair of latent variables. This implies that instead of training a neural network between all the input and output variables, a simpler single-input-single-output network is optimised. As a consequence, the limitations associated with multiple-input-multiple-output neural network modelling are overcome. In particular, the risk of the algorithm falling into a local minima, and hence potentially providing poor parameter estimation is expected to be significantly reduced. Furthermore, the comparison between the use of the sigmoid or gaussian network indicates that the radial basis function network is preferable to the

sigmoid network. As observed in Chapter 5, the radial basis function network appears to outperform the sigmoid network inner model. This is mainly due to the computational strategy adopted for building the RBF network models as well as the computational speed and convergence reliability of the RBF network.

6.2 Further Work

The error based weights updating procedure proposed in this Thesis has been shown to enhance the performance of non-linear PLS algorithms based upon the use of a polynomial (quadratic) expansion or a sigmoid or a gaussian neural network to fit the mapping between the input and output latent variables. Furthermore, the error based non-linear PLS algorithms have been shown to be capable of modelling highly non-linear systems. In particular the sigmoid neural network and the radial basis function network PLS algorithms were shown to provide general approximation capabilities for the non-linear PLS methodology. However, both methodologies require appropriate algorithms for training the model parameters. In particular the sigmoid neural networks can be trained using an early stopping criteria to avoid overfitting and to help the optimisation routine to reach the optimal solution (minimum). On the other hand, even though the radial basis function network does not need an improved non-linear optimisation routine (Appendix 2) the modelling performance might be improved by using activation functions other than the gaussian function (*e.g. the thin plate spline*), or else using different bandwidths for the gaussian function.

A further development of the proposed non-linear PLS approaches (which was not investigated in this work) is within the framework of dynamic empirical modelling. A number of dynamic PLS algorithms have been proposed in the literature aimed at merging dynamic features within the linear PLS framework (Ricker, 1988; Qin, 1993; Kaspar *et al.*, 1993). These have been shown to give better performance than traditional discrete (finite response) models such as FIR and ARX/ARMAX (Ljung, 1987). In particular FIR and ARX/ARMAX frameworks based upon the use of linear PLS regression have been shown to outperform similar approaches which are based upon the traditional MLR. In this context, the error based non-linear PLS algorithms can be used as regression tool for FIR or ARX/ARMAX models from which to develop non-linear dynamic regression algorithms. However, the validation of such models may become excessively

time demanding because of the large number of parameters requiring validation. When building finite response dynamic models, the time lags for the input and output variables need to be included in the number of parameters to be optimised.

In the Thesis the algorithms have been tested and compared on the basis of their performance in modelling data produced from a non-linear mathematical function and the simulation of a real non-linear system. It would be interesting (and challenging) to compare the performances of the algorithms when modelling real data. Such an application would provide useful information about the behaviour of the error based weights updating procedure in the presence of measurement and process noise and time correlations between the variables.

Bibliography

Anderson, T. W., (1963). Asymptotic Theory for Principal Component Analysis. *Ann. of Math. Statist.*, 34, 123-151.

Anderson, T. W., (1984). *An Introduction to Multivariate Statistical Analysis*. Wiley, New York.

Baffi, G., Martin, E. B., and Morris, A. J., (1998). Non-linear Projection to Latent Structures Revisited (The Quadratic PLS Algorithm). *Computers Chem. Engng*. (in press).

Baffi, G., Martin, E. B., and Morris, A. J., (1998). Non-linear Projection to Latent Structures (The Neural Network Inner Model). (in preparation).

Berglund, A., and Wold, S., (1997). INLR, Implicit Non-Linear Latent Variable Regression. *J. Chemometrics*, 11, 141-156.

Bisani, M. L., Faraone, D., Clementi, S., Esbensen, K. H., and Wold, S., (1983). Principal Component and Partial Least-Squares Analysis of the Geochemistry of Volcanic Rocks from the Aeolian Archipelago. *Anal. Chim. Acta*, 150, 129-143.

Bishop, C., (1991). Improving the Generalisation Properties of Radial Basis Function Networks. *Neural Computation*, 3, 579-588.

Boardman, A. E., Baldwin, S. H., and Wold, H., (1981). The Partial Least Squares-Fix Point Method of Estimating Interdependent Systems with Latent Variables. *Comm. Statist. Theor. Meth.*, A10(7), 613-639.

Chen, S., Cowan, C. F. N., and Grant, P. M., (1991). Orthogonal Least Squares Algorithms for Radial Basis Function Networks. *IEEE Trans. Neural Networks*, 2(2).

Cherkassky, V., and Gehring, D., (1996). Comparison of Adaptive Methods for Function Estimation from Samples. *IEEE Transactions on Neural Networks*, 7, 969-984.

Cleveland, W. S., (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *J. Amer. Statist. Assoc.*, 368, 829-836.

Cybenko, G., (1989). Approximation by Superpositions of a Sigmoidal Function. *SIAM J. Sci. Stat. Comput.* 5, 175-191.

Dayal, B. S., and MacGregor, J. F., (1997). Improved PLS Algorithm. *J. Chemometrics*, 11, 73-85.

De Jong, S., (1991). Chemometrical Applications in an Industrial Food Research Laboratory. *Mikrochim. Acta*, 2, 93-101.

De Jong, S., (1993a). SIMPLS: an Alternative Approach to Partial Least Squares. *Chem. Intell. Lab. Sys.*, 18, 251-263.

De Jong, S., (1993b). Short Communications: PLS Fits Closer Than PCR. *J. Chemometrics*, 7, 551-557.

De Jong, S., and Ter Braak, C. J. F., (1994). Short Communications: Comments on the PLS Kernel Algorithm. *J. Chemometrics*, 8, 169-174.

Eastman, H. T., and Krzanowski, W. J., (1982). Cross-Validatory Choice of the Number of Components From a Principal Component Analysis. *Technometrics*, 24(1), 73-77.

Frank, I. E., and Kowalski, B. R., (1984). Prediction of Wine Quality and Geographic Origins from Chemical Measurements by Partial Least-Squares Regression Modelling. *Anal. Chim. Acta*, 162, 241-251.

Frank, I. E., (1990). A Non-linear PLS Model. *Chemometrics and Int. Lab. Syst.*, 8, 109-119.

Friedman, J. H., (1991). Multiple Adaptive Regression SPLINES. *The Annals of Statistics*, 19, 1-67.

Geladi, P., and Kowalski, B. R., (1986a). Partial Least-Squares Regression: a Tutorial. *Anal. Chim. Acta*, 185, 1-17.

Geladi, P., and Kowalski, B. R., (1986b). An Example of 2-block Predictive Partial Least-Squares Regression with Simulated Data. *Anal. Chim. Acta*, 185, 19-32.

Geladi, P., (1988). Notes on the History and Nature of Partial Least Squares (PLS) Modelling. *J. Chemometrics*, 2, 231-246.

Gemperline, P. J., Long, J. R., and Gregoriou, V. G., (1991). Nonlinear Multivariate Calibration Using Principal Component Regression and Artificial Neural Networks. *Anal. Chem.*, 63, 2313-2323.

Gittins, R., (1969). The Application of Ordination Techniques. *Ecological Aspects of the Mineral Nutrition of Plants* (British Ecol. Soc. Symp. 9), I. H. Rorison, ed., Blackwell Scientific Publications, Oxford

Goutis, C., (1997). A Fast Method to Compute Orthogonal Loadings Partial Least Squares. *J. Chemometrics*, 11, 33-38.

Henson, M. A., and Seborg, D. E., (1994). Adaptive Non-linear Control of a pH Neutralisation Process. *IEEE Transaction on Control Systems Technology*, 3, 169 - 183.

Holcomb, T. R., and Morari, M., (1992). PLS/Neural Networks. *Computers Chem. Engng.*, 16, 393-411.

Hornik, K., Stinchcombe, M., and White, H., (1989). Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*, 2(5), 359-366.

Höskuldsson, A., (1988). PLS Regression Methods. *J. Chemometrics*, 2, 211-228.

Hotelling, H., (1933). Analysis of a Complex of Statistical Variables into Principal Components. *J. Educ. Psychol.*, 24(6), 417-441.

Irwin, G. W., Warwick, K., and Hunt, K. J., (1995). *Neural Network Applications in Control.* The institution of Electrical Engineers.

Johansen, T. A., and Foss, B. A., (1997). Operating Regime Based Process Modelling and Identification. *Computers Chem. Engng.*, 21, 159-176.

Jolicoeur, P., and Mosimann, J. E., (1960). Size and Shape Variation in the Painted Turtle. A Principal Component Analysis. *Growth*, 24(4), 339-354.

Jolliffe, T. I, (1986)
Principal Component Analysis.
Springer-Verlag.

Kaspar, M. H., and Ray, W. H., (1993). Partial Least Squares Modelling as Successive Singular Value Decompositions. *Computers Chem. Engng.*, 17, 985-989.

Kaspar, M. H., and Ray, W. H., (1993). Dynamic PLS Modelling for Process Control. *Chemical Engineering Science*, 48(20), 3447-2461.

Kavšek-Biasizzo, K., Škrjanc, I., and Matko, D., (1997). Fuzzy Predictive Control of Highly Non-linear pH Process. *Computers Chem. Engng.*, 21, Suppl., S613-S618, 1997.

Kim, S. J., Lee, M., Park, S., Lee, S. Y., and Park, C. H., (1997). A Neural Linearising Control Scheme for Non-linear Chemical Processes. *Computers Chem. Engng.*, 21, 187-200.

Kohonen, T., (1990). The Self-Organising Map. *Proceedings of the IEEE*, 78(9), 1464-1480.

Kourti, T., and MacGregor, J. F., (1994). Multivariate SPC Methods for Monitoring and Diagnosing of Process Performance. *Proceedings of PSE '94*, 739-746.

Kramer, M. A., (1992). Autoassociative Neural Networks. *Computers Chem. Engng.*, 16(4), 313-328.

Kresta, J. V., MacGregor, J. F., and Marlin, T. E., (1991). Multivariate Statistical Monitoring of Process Operating Performance. *Canad. J. Chem. Eng.*, 69, 35-47.

Krzanowski, W. J., (1987). Cross-Validation in Principal Component Analysis. *Biometrics*, 43, 575-584.

Krzanowski, W. J., and Marriott, F. H. C., (1994). *Multivariate Analysis.* Edward Arnold, London

Lennard, J. A., and Kramer, M. A., (1991). Radial Basis Function Networks for Classifying Process Faults. *IEEE Trans. Control System Mag.* April 1991, 31-38.

Lindberg, W., Persson, J., and Wold, S., (1983). Partial Least-Squares Method for Spectrofluorimetric Analysis of Mixtures of Humic Acid and Ligninsulfonate. *Anal. Chem.*, 55, 643-648.

Lindgren, F., Geladi, P., and Wold, S., (1993). The Kernel Algorithm for PLS. *J. Chemometrics*, 7, 45-59.

Ljung, L, (1987). *System Identification- Theory for the User.* Prentice-Hall.

Lorber, A., Wagner, L. A., and Kowalski, B. R., (1987). A Theoretical Foundation for the PLS Algorithm. *J. Chemometrics*, 1, 19-31.

MacGregor, J. F., Marlin, T. E., Kresta, J. V., and Skagerberg, B., (1991a). Multivariate Statistical Methods in Process Analysis and Control. *AIChE Symp. Proc. Of the Fourth International Conference on Chemical Process Control*, South Padre Island, Texas, February 17-22, 79-99, AIChE Publ. No. P-67, New York.

MacGregor, J. F., Skagerberg, B., and Kiparissides, C., (1991a). Multivariate Statistical Process Control and Property Inference Applied to Low Density Polyethylene Reactors. *IFAC Symp. ADCHEM'91*, Toulouse, France, Oct. 1991, Pergamon Press, 131-135.

MacGregor, J. F., (1994). Statistical Process Control of Multivariate Processes. IFAC, ADCHEM'94, Kyoto, Japan, May 24-27, Pergamon Press.

MacGregor, J. F., Jaeckle, C., Kiparissides, C., and Koutoudi, M., (1994). Process Monitoring and Diagnosis by Multiblock PLS Method. *Process System Engineering*, 40(5), 826-838.

Malthouse, E. C., Tamhane, A. C., and Mah R. S. H., (1997). Non-linear Partial Least Squares. *Computers Chem Engng.*, 21, 875-890.

Manne, R., (1987). Analysis of Two Partial Least Squares Algorithms for Multivariate Calibration. *Chem. Intell. Lab. Sys.*, 2, 187-194.

Mardia, K. V., Kent, J. T., and Bibby, J. M., (1987). *Multivariate Analysis*. Academic Press, London.

Martens, H., and Næs, T., (1989). *Multivariate calibration*. John Wiley & Sons Ltd.

Mejdell, T., and Skogestad, S., (1991). Estimation of Distillation Composition from Multiple Temperature Measurements Using Partial-Least-Squares Regression. *Ind. Eng. Chem. Res.*, 30, 2543-2555.

Moody, J., and Darken, C. J., (1989). Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, 1, 281-294.

Mosteller, F., and Wallace, D. L., (1963). Inference in an Authorship Problem. *J. Amer. Statist. Assoc.*, 58, 275-309.

Næs, T., and Martens, H., (1985). Comparison of Prediction Methods for Multicollinear Data. *Comm. Statist.-Simula. Computa.*, 14(3), 545-576.

Okamoto, M., (1969). Optimality of Principal Components. *Multivariate Analysis II*, 673-685, Krishnaiah, P. R.

Phatak, A, and De Jong, S., (1997). The Geometry of Partial Least Squares. *J. Chemometrics*, 11, 311-338.

Pearson, K., (1901). On Lines and Panes of Closest Fit to Systems of Points in Space. *Phil. Mag.*, Ser. 6, 2(11), 559-572.

Qin, S. J., (1993). A Statistical Perspective of Neural Networks for Process Modelling and Control. Proceedings of the 1993 International Symposium on Intelligent Control, Chicago, Illinois, USA- August 1993

Qin, S. J., (1993). Partial Least Squares Regression for Recursive System Identification. Proceedings of the 32nd Conference on Decision and Control, San Antonio, Texas, USA, December 1993

Qin, S. J., and McAvoy, T. J., (1992). Non-linear PLS Modelling using Neural Networks. *Computers and Chemical Engineering*, 16, 379-391.

Ricker, N. L., (1988). The Use of Biased Least-Squares Estimators for Parameters in Discrete-Time Pulse Response Models. *Ind. Eng. Chem. Res.*, 27, 343-350.

Rännar, S., Geladi, P., Lindgren, F., and Wold, S., (1995). A PLS Kernel Algorithm for Data Set with Many Variables and Few Objects. Part II: Cross Validation, Missing Data and Examples. *J. Chemometrics*, 9, 459-470.

Rao, C. R., (1964). The Use and Interpretation of Principal Component Analysis in Applied Research. *Sanhkya*, Ser. A, 26, 329-358.

Rao, C. R., (1966). Generalized Inverse for Matrices and its Application in Mathematical Statistics. *Research Papers in Statistics, Festschrift for J. Neyman* (F. David, ed.), 263-279, Wiley, New York.

Saunders, A. C. G., (1992). Process Monitoring and Non-Linear Observers. PhD Thesis, University of Newcastle.

Seborg, D. E., Edgar, T. F., and Mellichamp, D. A., (1989). *Process Dynamics and Control.* John Wiley & Sons, Inc.

Skagerberg, B., MacGregor, J. F., and Kiparissides, C., (1992). Multivariate Data Analysis Applied to Low-Density Polyethylene Reactors. *Chem. Intell. Lab. Syst.*, 14, 341-356.

Stone, M., (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *J. Roy. Statist. Soc.*, B, 36, 111-113.

Wangen, L. E., and Kowalski, B. R., (1988). A Multiblock Partial Least Squares Algorithms for Investigating Complex Chemical Systems. *J. Chemometrics*, 3, 3-20.

Westad, F., Diepold, K., and Martens, H., (1996). QR-PLS: Reduced Rank Regression for High-Speed Hardware Implementation. *J. Chemometrics*, 10, 439-451.

Wetherill, G. B., (1986). *Regression Analysis with Applications.* Chapman and Hall.

Wilson, D. J. H., Irwin, G. W., and Lightbody, G., (1997). Nonlinear PLS modelling Using Radial Basis Functions. American Control Conference, Albuquerque, New Mexico, June 4-6 1997.

Wold, H., (1966a). Nonlinear Estimation by Iterative Least Squares procedures. *Research Papers in Statistics, Festschrift for J. Neyman* (F. David, ed.), 411-444, Wiley, New York.

Wold, H., (1966b). Estimation of Principal Components and Related Models by Iterative Least Squares. *Multivariate Analysis II*, 391-420, Krishnaiah, P. R.

Wold, H., (1973). Nonlinear Iterative Partial Least Squares (NIPALS) Modelling: Some Current Developments. *Multivariate Analysis III*, 383-407, Krishnaiah, P. R

Wold, S., (1974). Spline Functions in Data Analysis. *Technometrics*, 16(1), 1-11.

Wold, S., (1978). Cross-Validatory Estimation of the Number of Components in Factor and Principal Component Analysis. *Technometrics*, 20(4), 397-405.

Wold, S., Wold, H., Dunn, W. J., and Ruhe, A., (1980). Report UMINF-83, Department of Chemistry, University of Umeå, Sweden.

Wold, S., Esbensen, K., and Geladi, P., (1987). Principal Component Analysis. *Chem. Intell. Lab. Syst.*, 2, 37-52.

Wold, S., Kettaneh-Wold, N., and Skagerberg, B., (1989). Non-linear PLS Modelling. *Chemometrics and Int. Lab. Syst.*, 7, 53-65.

Wold, S., (1992). Non-linear Partial Least Squares Modelling II. Spline Inner Function. *Chemometrics and Int. Lab. Syst.*, 14, 71-84.

Wold, S., Kettaneh, N., and Tjessem, K., (1996). Hierarchical Multiblock PLS and PC Models for Easier Model Interpretation and as an Alternative to Variable Selection. *J. Chemometrics*, 10, 463-482.

Zheng, G. L., and Billings, S. A., (1994). Radial Basis Function Network Training Using a Fuzzy Clustering Scheme. Personal Communication (Research Report No. 505).