JOURNAL OF CHEMOMETRICS, VOL. 9, 91–123 (1995)

A COMBINED THEORY FOR PCA AND PLS

AGNAR HÖSKULDSSON
Danish Engineering Academy, Bldg. 358, DK-2800 Lyngby, Denmark

SUMMARY

We present here an algorithmic approach to modelling data that includes principal component analysis (PCA) and partial least squares (PLS). In fact, the numerical algorithm presented can carry out PCA or PLS. The algorithm for linear analysis and extensions to non-linear analysis applies to both PCA and PLS. The algorithm allows for combination of PCA and PLS types of models and therefore extends modelling to new types of models that involve combination of regression models and 'selection of variation' models, which is the idea of PCA-type models. The fact that the algorithm carries out both PCA and PLS shows that PCA and PLS are based on the same theory. This theory is based on the H-principle of mathematical modelling. The algorithm allows tests for outliers, sensitivity analysis and tests of submodels. These aspects of the algorithm are treated in detail. We compute various measures of sizes, e.g. of components, of the covariance matrix, of its inverse, etc. that show how much the algorithm has selected at each step. The analysis is illustrated by data from practice.

KEY WORDS H-principle; PCA; PLS regression; latent variable models; quadratic models; sensitivity analysis; outlier tests; prediction variances

INTRODUCTION

Principal component analysis (PCA) is popular in applied statistical work and data analysis.[1–5] The reason is that PCA has a good ability to summarize multivariate variation in data. It has an intuitive appeal because it can be given a geometric interpretation in terms of an ellipsoid that encloses the data points. The principal components are the major axes in such an ellipsoid. Also, partial least squares (PLS) regression has been found preferable in many situations because it generally provides more stable predictions than do traditional methods.[6,7] Here we present an algorithmic approach that combines the basic features of PCA and PLS into one algorithm. The algorithm is based on the Heisenberg principle of mathematical modelling, the H-principle. The algorithm is formulated here in terms of linear and quadratic models. The importance of the present algorithm is partly due to the fact that it allows different types of tests such as outlier tests, analysis of sensitivity of samples and tests of submodels as in regression analysis and partly due to that it allows measures of sizes and a geometric interpretation of data as in principal component analysis. Thus the algorithm extends the working techniques of regression analysis and principal component analysis to other areas of modelling. We use the working techniques of regression when working with principal component types of analysis and conversely we apply the working tools of principal component analysis (and factor analysis)

CCC 0886–9383/95/020091–33

Received 25 February 1994
Revised 20 October 1994

when studying regression analysis types of models. Thus we get approximate expressions for variances of parameters and new samples (observations) as in linear regression and the working techniques of PCA when working with regression models, such as plots of loading and score vectors and measures of extracted variation.

Normally PCA and PLS are considered different techniques. PCA is conceptually a simple technique, while PLS is generally considered complicated and somewhat difficult to comprehend. The fact that numerical analysis can be carried out by the same algorithm shows that these two theories are basically the same. This allows for new developments, both theoretical and practical, because each area has been extensively studied. Since basically these theories are the same, we can expect many different results from PCA to be carried over to PLS and conversely. Another important aspect is that non-linear extensions are the same for PCA and PLS.

Principal component regression (PCR) is popular among statisticians, because the data are decomposed according to the variation in the X-matrix. One view of PCA or PCR is that each component is selected from $\mathbf{X}$ such that we get maximal size of the *loading vector* at each step. (In the presentation of PCA a scaling method is often used such that the loading vector has length one.) The disadvantage is that the components need not be good at describing the response vectors. In PLS regression with one response variable the *score vector* is selected such that we get maximal covariance between the score vector and the response variable. From the application point of view we get the important question can we select at each step:

(i) a loading vector that is as large as possible and
(ii) a score vector that has maximal covariance with the response variable?

This would combine the idea of PCA and PLS in each step. We shall show that this is in fact possible. We shall show how to determine a weight vector for rows such that we get a loading vector of maximal size and a weight vector for columns such that the score vector has maximal covariance. This approach can be expected to be important in practice, where there are many variables and where we are not sure whether the training set contains all the relevant variation in data.

The methods presented here are basically regression methods, but we shall show that PCA, both linear and non-linear extensions, can be viewed as regression. The PCA types of methods are those where the response data are chosen as the X-data. This idea gives a new type of model where we mix a PCA-type model and a regression model in the same model.

All the methods described here are carried out by one algorithm. The basic skeleton of the algorithm is given in MATLAB code in Appendix III. The algorithm works with two matrices $\mathbf{X}$ and $\mathbf{Y}$. The algorithm does not mind the contents of $\mathbf{X}$ or $\mathbf{Y}$. If $\mathbf{Y}=\mathbf{X}$, we get PCA types of models. If we place some of the instrumental data (X-data) into $\mathbf{Y}$, we get mixture models. Here we also show how the algorithm handles non-linear models. In practical work it is of fundamental importance to be able to detect non-linearities in data.

Associated with the algorithm is a large collection of measures. These are used to provide with a description of the results of the modelling process. The importance of these measures is due to their ability to identify the noise level. Thus they can e.g. be used to determine the appropriate number of components to use.

In Reference 8 we showed that the H-principle provides the conceptual background for both PCA and PLS and actually unifies them into one theory. In this paper the details are worked out.

NOTATION AND ASSUMPTIONS

Notation

We suppose that there is given a data table $\mathbf{X}$ that is an $N \times K$ matrix. The columns of $\mathbf{X}$ are viewed as the x-variables. We suppose also that there are given M response variables whose values are denoted by $\mathbf{Y}$, which is an $N \times M$ matrix. Matrices are denoted by bold capital letters and vectors by bold lowercase letters. The columns of $\mathbf{X}$ are written as $(\mathbf{x}_j)$ and rows as $(\mathbf{x}^i)$. Thus we write

$$\mathbf{X} = (x_{ij}) = (\mathbf{x}_j) = (\mathbf{x}^i) = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_K)$$

We will sometimes write the product of two matrices as

$$\mathbf{TP}^T = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_1^T + \cdots + \mathbf{t}_K\mathbf{p}_K^T$$

This equation follows from the rules of matrix multiplication. Here $\mathbf{T} = (\mathbf{t}_1, ..., \mathbf{t}_K)$ is an $N \times K$ matrix and $\mathbf{P} = (\mathbf{p}_1, ..., \mathbf{p}_K)$ is a $K \times K$ matrix. We write $\mathbf{T}_A$ for the $N \times A$ matrices containing the first A vectors of $\mathbf{T}$ and similarly $\mathbf{P}_A$ and $\mathbf{R}_A$ for the first A columns of $\mathbf{P}$ and $\mathbf{R}$ respectively.

The algorithm of this paper is basically a regression algorithm of $\mathbf{Y}$ onto $\mathbf{X}$. Some preprocessing of data may be needed before the algorithm is applied. If the matrices $\mathbf{X}$ and $\mathbf{Y}$ are partitioned as $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$ and $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2)$, we sometimes see the following preprocessing tasks:

(a)	$\mathbf{X} \leftarrow (\mathbf{X}, \mathbf{Y})$	$\mathbf{Y} \leftarrow (\mathbf{X}, \mathbf{Y})$
(b)	$\mathbf{X} \leftarrow (\mathbf{Y})$	$\mathbf{Y} \leftarrow (\mathbf{Y})$
(c)	$\mathbf{X} \leftarrow (\mathbf{X}, \mathbf{Y}_1)$	
(d)		$\mathbf{Y} \leftarrow (\mathbf{Y}, \mathbf{X}_1)$

In (a) we want a PCA type of analysis of all the data and in (b) we want it on $\mathbf{Y}$. In (c) we expand the X-matrix with some columns from $\mathbf{Y}$ and in (d) we expand $\mathbf{Y}$ with some columns from $\mathbf{X}$. In (c) we want the components to depend on some of the Y-data and in (d) we want the components also to describe some of the X-data.

We shall be working with *components*. A component is a triple $(\lambda, \mathbf{t}, \mathbf{p})$. The *score* vector $\mathbf{t}$ is an N vector computed as $\mathbf{t} = \mathbf{Xw}$, where $|\mathbf{w}|^2 = (\mathbf{w}^2\mathbf{w}) = \sum w_j^2 = 1$, i.e. a score vector is a linear combination of columns of $\mathbf{X}$, where coefficients make a vector of length one. Similarly a *loading* vector $\mathbf{p}$ is computed as $\mathbf{p} = \mathbf{X}^T\mathbf{v}$, where $|\mathbf{v}|^2 = (\mathbf{v}^T\mathbf{v}) = \sum v_i^2 = 1$, i.e. a loading vector is a linear combination of rows of $\mathbf{X}$ where coefficients make a vector of length one. λ (lambda in the algorithm) is a scaling constant. Note that in the present algorithm neither the score vector $\mathbf{t}$ nor the loading vector $\mathbf{p}$ has length one. The scaling is taken care of by λ.

The following is a list of the important matrices in the algorithm.

$\mathbf{X} = (x_{ij})$	An $N \times K$ matrix containing the original data.
$= (\mathbf{x}^j)$	Columns of $\mathbf{X}$, $j = 1, ..., K$.
$= (\mathbf{x}^i)$	Rows of $\mathbf{X}$, $i = 1, ..., N$.
$\mathbf{Y}$	An $N \times M$ data matrix. Typically response data.
$\mathbf{T}$	An $N \times K$ matrix of score values, computed as $\mathbf{t} = \mathbf{Xw}$.
$\mathbf{P}$	A $K \times K$ matrix of loading values, computed as $\mathbf{p} = \mathbf{X}^T\mathbf{v}$.
$\mathbf{\Lambda}$	A $K \times K$ diagonal matrix of scaling constants $\lambda = 1/(\mathbf{v}^T\mathbf{Xw})$. We have $\mathbf{X} = \mathbf{T\Lambda P}^T$.
$\mathbf{R}$	A $K \times K$ matrix. Transformation matrix. Transforms samples to score space. We have $\mathbf{R}^T\mathbf{P} = \mathbf{\Lambda}^{-1}$.

$\mathbf{S}$	An $N \times K$ matrix. Transformation matrix. Transforms variables to loading space. We have $\mathbf{S}^T\mathbf{T} = \mathbf{\Lambda}^{-1}$.
$\mathbf{W}$	A $K \times K$ matrix of weights of columns of $\mathbf{X}$. The column vectors of $\mathbf{W}$ are scaled to unit length. $\mathbf{P}^T\mathbf{W}$ is upper triangular with $(\mathbf{v}^T\mathbf{X}\mathbf{w}) = 1/\lambda$ in its diagonal.
$\mathbf{V}$	An $N \times K$ matrix of weights of rows of $\mathbf{X}$. The column vectors of $\mathbf{V}$ are scaled to unit length. $\mathbf{T}^T\mathbf{V}$ is upper triangular with $(\mathbf{v}^T\mathbf{X}\mathbf{w}) = 1/\lambda$ in its diagonal.
$\mathbf{Q}$	An $M \times K$ matrix. We have $\mathbf{Q} = \mathbf{Y}^T\mathbf{S}$.
$\mathbf{B}$	A $K \times M$ matrix of regression coefficients. We have $\mathbf{B} = \mathbf{R}\mathbf{\Lambda}\mathbf{Q}^T$.
$\mathbf{X}_{(A)}$	$= \sum^A \lambda_a \mathbf{t}_a \mathbf{p}_a^T = \mathbf{T}_A \mathbf{\Lambda}_A \mathbf{P}_A^T$, an approximation of $\mathbf{X}$.
$\mathbf{X}^+_{(A)}$	$= \sum^A \lambda_a \mathbf{r}_a \mathbf{s}_a^T = \mathbf{R}_A \mathbf{\Lambda}_A \mathbf{S}_A^T$, an approximation of $\mathbf{X}^+$, a generalized inverse of $\mathbf{X}$.

The index a is used for the components selected, while i and j are the elements in the matrices.

Centring data

If in a linear regression model there is a constant term, it is convenient to assume that there is one extra x-variable that consists of ones only.[9] This simplifies the formulae. Another approach is to suppose that $\mathbf{X}$ and $\mathbf{Y}$ have been centred. This means that we have taken the mean value of each column and subtracted it from each element in the corresponding column. When the data have been centred, we can write e.g. the sample covariance matrix of x-variables as $\mathbf{X}^T\mathbf{X}/(N-1)$.

In this paper we suppose that data have been centred, both $\mathbf{X}$ and $\mathbf{Y}$, or that a column of ones has been added to $\mathbf{X}$, if a constant term is wanted in the models.

Statistical model assumptions

The statistical assumptions needed in this paper are the standard assumptions. For linear regression we suppose that $\mathbf{y}$, for given $\mathbf{x}$, are normally distributed with mean $\mathbf{B}\mathbf{x}$ and variance matrix $\Sigma_{y|x}$. When we need assumptions concerning the x-variables, we suppose that $\mathbf{x}$ is normally distributed with mean zero and covariance matrix $\mathbf{\Sigma}$. Since the assumptions are very simple, we shall mainly formulate the algorithms and methods in terms of the sample vectors and matrices. A formulation in terms of population is a straightforward translation of the sample terms.

MATHEMATICAL MODELS AND DATA

Some types of models

It will be convenient to discuss some common types of models which can be handled by the present algorithm. The discussion here will be based on small models to illustrate the ideas. Consider a case where we have three response variables and two explanatory variables or x-variables. A linear regression model involving these variables can be written as

$$\begin{aligned} y_1 &= b_{11}x_1 + b_{12}x_2 + \text{residual}_1 \\ y_2 &= b_{21}x_1 + b_{22}x_2 + \text{residual}_2 \\ y_3 &= b_{31}x_1 + b_{32}x_2 + \text{residual}_3 \end{aligned} \tag{1}$$

In the model (1) we seek to explain the variation in the y-variables by the x-variables. In PCA

we seek to describe the variation by an appropriate subspace model

$$\begin{aligned} y_1 &= p_{11}t_1 + p_{12}t_2 + \text{residual}_1 \\ y_2 &= p_{21}t_1 + p_{22}t_2 + \text{residual}_2 \\ y_3 &= p_{31}t_1 + p_{32}t_2 + \text{residual}_3 \end{aligned} \tag{2}$$

The components (score variables) t_1 and t_2 are computed on the basis of the variation in the y-variables. The model (2) shows that the data values (y_1, y_2, y_3) are situated on a two-dimensional plane. The data values (t_1, t_2) are orthogonal, $\mathbf{t}_1^T\mathbf{t}_2 = 0$, and determine the direction of the two main axes of the ellipse that contains the data values (y_1, y_2, y_3) in the plane. The decomposition (2) is useful when the y-variables are highly correlated and the number of components is small compared with the number of y-variables.

Consider now again the model (1). It happens frequently that the residuals of (1) are highly correlated. The explanatory variables may explain a large part of the total variation in the ys but may not explain the covariation among the ys. Therefore it is often useful to expand (1) by (2) into the model

$$\begin{aligned} y_1 &= b_{11}x_1 + b_{12}x_2 + p_{11}t_1 + p_{12}t_2 + \text{residual}_1 \\ y_2 &= b_{21}x_1 + b_{22}x_2 + p_{21}t_1 + p_{22}t_2 + \text{residual}_2 \\ y_3 &= b_{31}x_1 + b_{32}x_2 + p_{31}t_1 + p_{32}t_2 + \text{residual}_3 \end{aligned} \tag{3}$$

In the model (3) we ask simultaneously for (a) the influence of the explanatory variables and (b) an orthogonal decomposition of the covariance of the ys. The estimation procedure of the bs and ps may look complicated, because the t-variables depend on the ys; however, as shown below, the estimation procedure is simple. Above we have discussed linear models. The algorithm handles linear and quadratic ones. The linear and quadratic regression model extension of (1) is:

$$\begin{aligned} y_1 &= b_{11}x_1 + b_{12}x_2 + c_{1,11}x_1^2 + c_{1,22}x_2^2 + c_{1,12}x_1x_2 + \text{residual}_1 \\ y_2 &= b_{21}x_1 + b_{22}x_2 + c_{2,11}x_1^2 + c_{2,22}x_2^2 + c_{2,12}x_1x_2 + \text{residual}_2 \\ y_3 &= b_{31}x_1 + b_{32}x_2 + c_{3,11}x_1^2 + c_{3,22}x_2^2 + c_{3,12}x_1x_2 + \text{residual}_3 \end{aligned} \tag{4}$$

Each response variable is explained by a linear and a second order surface. Similarly we can expand (2) to a linear and a second-order surface:

$$\begin{aligned} y_1 &= p_{11}t_1 + p_{12}t_2 + q_{1,11}t_1^2 + q_{1,22}t_2^2 + q_{1,12}t_1t_2 + \text{residual}_1 \\ y_2 &= p_{21}t_1 + p_{22}t_2 + q_{2,11}t_1^2 + q_{2,22}t_2^2 + q_{2,12}t_1t_2 + \text{residual}_2 \\ y_3 &= p_{31}t_1 + p_{32}t_2 + q_{3,11}t_1^2 + q_{3,22}t_2^2 + q_{3,12}t_1t_2 + \text{residual}_3 \end{aligned} \tag{5}$$

In (2) we ask for the best possible plane, while in (5) we ask for the best second-order surface that the points (y_1, y_2, y_3) are lying on (apart from small residuals or deviations). Similarly (3) can be expanded into an expression containing second-order terms.

Weighting schemes

The algorithm determines an optimal direction or score vector $\mathbf{t}$ as $\mathbf{t} = \mathbf{X}\mathbf{w}$ with $|\mathbf{w}| = 1$. If we write

$$\mathbf{t} = \mathbf{X}\mathbf{w} = \mathbf{x}_1 w_1 + \mathbf{x}_2 w_2 + \cdots + \mathbf{x}_K w_K$$

we see that we are determining an appropriate weighted average of the columns of $\mathbf{X}$. If $\mathbf{Y} = \mathbf{X}$

and $\mathbf{w}$ can vary freely, we get PCA. If we require $\mathbf{w}$ of the form $\mathbf{w} = (0, .., 0, 1, 0, ... 0)$, we are selecting principal variables. For general $\mathbf{X}$ and $\mathbf{Y}$, we get a PLS type of regression when $\mathbf{w}$ varies freely and a stepwise regression when $\mathbf{w}$ is required to contain a single element one and the others are zero. When $\mathbf{X} \leftarrow (\mathbf{X}, \mathbf{Y}_1)$, we can split the score vector into two parts:

$$\mathbf{t} = (\mathbf{x}_1 w_1 + \cdots + \mathbf{x}_K w_K) + (\mathbf{y}_1 w_{K+1} + \cdots) = \mathbf{t}^{(1)} + \mathbf{t}^{(2)}$$

The component $\mathbf{t}^{(1)}$ refers to $\mathbf{X}$ and the component $\mathbf{t}^{(2)}$ to $\mathbf{Y}^1$. We can treat $\mathbf{w}^{(1)} = (w_1, w_2, ..., w_K)$ and $\mathbf{w}^{(2)} = (w_{K+1}, ...)$ in similar ways, i.e. freely or by fixing certain values.

In some cases a robust procedure for estimating $\mathbf{w}$ must be used. This is the case when a correlation coefficient is invalid for describing the correlation between columns of $\mathbf{X}$ and columns of $\mathbf{Y}$. See Reference 9 for various robust estimation procedures.

The co-ordinates of $\mathbf{w}$ tell us how to look at the columns of $\mathbf{X}$. Similarly we can have a vector $\mathbf{v}$ that tells us how we should treat the rows of $\mathbf{X}$. This is often important in practice when theory or preliminary analysis suggests that some data are more important than others. This aspect is also important when we have reference objects that we want our loading vectors to 'aim' at. We shall show here how such weight vectors can be integrated in the modelling procedure.

DATA FOR ILLUSTRATION

The various results in this paper will be illustrated by an example. The data related to the production of a chemical material for adhesion. The products go through a very specific production process. The main factors determining the quality of the final products are the temperatures and the duration of each temperature level. We consider here 40 temperatures used in production. We do not include here the duration of the temperatures, because this is approximately constant for the chosen temperatures. The quality is measured by two variables, namely peel–adhesion and strength. We consider here 50 samples. Since the quality parameters are directly related to the temperature treatment, we expect a good linear model with a rank that is considerably smaller than the number of temperature measurements, here 40.

We use this example to illustrate the formulae presented. We do not present score, loading, response variable plots and other standard plots that belong to the professional analysis of data.

THE H-PRINCIPLE OF MODELLING DATA

The H-principle is a procedure of how modelling of data should be carried out. The starting point is an $N \times K$ data matrix $\mathbf{X}$ and we wish to use it to describe or explain a data matrix $\mathbf{Y}$ that is an $N \times M$ matrix. The procedure suggested by the H-principle is as follows.

1. Carry out the analysis in steps, selecting one component $(\lambda, \mathbf{t}, \mathbf{p})$ from $\mathbf{X}$ at a time.
2. Determine the improvement in fit, Δfit, i.e. the explained variation in $\mathbf{Y}$, that a component can attain. Compute the associated increase in variance, Δvariance, of the parameter estimates. Select the component that is a solution to the task

$$\text{Maximize } (\Delta\text{fit})/(\Delta\text{variance}) \tag{6}$$

3 Reduce $\mathbf{X}$ by the component selected (rank one reduction):

$$\mathbf{X} \leftarrow \mathbf{X} - \lambda \mathbf{t}\mathbf{p}^{\mathrm{T}}$$

4. Determine whether a new component is to be selected.

The criterion (6) states that the improvement in fit and the increase in model variance are to be given equal weight. A more precise formulation of (6) depends on the model one is working

with and the way one computes the measure of fit. In a regression context the change in the theoretical variance is $\sigma^2/|\mathbf{t}|^2$. If we treat σ^2 as constant here, we get $\Delta\text{variance} \approx 1/|\mathbf{t}|^2$. For a more detailed study of the H-principle see Reference 8.

APPLICATION OF THE H-PRINCIPLE TO PCA

We suppose that we have given a data matrix $\mathbf{X}$ and we want to determine a score vector $\mathbf{t}$ computed as $\mathbf{t} = \mathbf{Xw}$ with $|\mathbf{w}| = 1$. The H-principle suggests that a score vector is determined such that a balance is obtained between the following two terms

(i) reduction in variation in $\mathbf{X}$ due to a component, $|\mathbf{X}^T\mathbf{t}|^2/(\mathbf{t}^T\mathbf{t})$
(ii) squared size of the component, $(\mathbf{t}^T\mathbf{t})$.

Taking the product of these two terms, we get

$$\text{maximize } [|\mathbf{X}_i^T|^2/(\mathbf{t}^T\mathbf{t})]\ (\mathbf{t}^T\mathbf{t}) = \text{maximize } |\mathbf{X}^T\mathbf{t}|^2 = \text{maximize } \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{w} \tag{7}$$

Thus the H-principle suggests that we should work with $\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)\mathbf{X}$. The solution of this maximization task is precisely the PCA solution. In the algorithm we may work with $\mathbf{X}_a^T(\mathbf{X}\mathbf{X}^T)\mathbf{X}_a$ or $\mathbf{X}_a^T(\mathbf{X}_a\mathbf{X}_a^T)\mathbf{X}_a$. The solution will in both cases be the ath PCA component.

APPLICATION OF THE H-PRINCIPLE TO PLS REGRESSION

Here we have given a matrix $\mathbf{Y}$ of data values of the response variables. We are to determine a score vector $\mathbf{t} = \mathbf{Xw}$ that describes $\mathbf{Y}$. The H-principle suggests a balance between the following two terms:

(i) reduction in variation in $\mathbf{Y}$ due to a component, $|\mathbf{Y}^T\mathbf{t}|^2/(\mathbf{t}^T\mathbf{t})$
(ii) squared size of the component, $(\mathbf{t}^T\mathbf{t})$.

Taking the product of these two terms, we get

$$\text{maximize } [|\mathbf{Y}_i^T|^2/(\mathbf{t}^T\mathbf{t})]\ (\mathbf{t}^T\mathbf{t}) = \text{maximize } |\mathbf{Y}^T\mathbf{t}|^2 = \text{maximize } \mathbf{w}^T\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{w} \tag{8}$$

Here the H-principle suggests us to work with $\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X} = \mathbf{X}^T(\mathbf{Y}\mathbf{Y}^T)\mathbf{X}$. This is precisely the criterion of PLS regression. Thus, the H-principle can be considered as the conceptual basis for PLS regression.

INTERPRETATION OF THE H-PRINCIPLE

We see that if we want a data matrix $\mathbf{X}$ to describe another data matrix $\mathbf{Z}$ that has the same number of rows as $\mathbf{X}$, we should work with $\mathbf{X}^T(\mathbf{Z}\mathbf{Z}^T)\mathbf{X}$. If $\mathbf{Z} = \mathbf{X}$, we get PCA; if $\mathbf{Z} = \mathbf{Y}$, we get PLS. This shows that *the same numerical algorithm can be used for all PCA and PLS computations.*

The H-principle suggests that if we want to describe $\mathbf{Y}$ by $\mathbf{X}$, we should work with the matrix $\mathbf{X}^T\mathbf{Y}$. The argument is that it provides a balance between fit and prediction. Let us look more closely at what this means. We can write $\mathbf{X}^T\mathbf{Y} = \sum (\mathbf{x}^i)^T\mathbf{y}^i$. This shows that if a row in $\mathbf{Y}$ is zero or close to zero, it is not used in the computations. The same holds for a row of $\mathbf{X}$. If it is small, it will have little influence. This can be a disadvantage when we use this procedure. The results will depend much on the scale used for the variables. Values close to zero (close to the average value for centred data) will have relatively less influence than larger values. Also, if there are extreme samples, they will tend to determine the components. Therefore we can generally

recommend scaling the data before analysis. It is common to scale the data such that the length or variance of each column is one (sometimes called autoscaling).

The H-principle provides with new perspectives in the modelling of data. PCA is conceptually very simple, while PLS is theoretically complicated, but the fact that we are basically carrying out the same computations allows for new methods and algorithms. For example, the formulae in Reference 5 developed for PCA are principally valid for PLS too. Likewise the approximate results for PLS regression are also valid for PCA. We shall discuss this later. An important aspect is that we do the same interpretation of the different magnitudes computed. For example, the scaling constant λ_a has a clear meaning in PCA as the inverse of the standard deviation, but we make the same interpretation of λ_a in other cases.

THE DECOMPOSITION ALGORITHM ASSOCIATED WITH THE H-PRINCIPLE

Associated with the H-principle is a decomposition algorithm that decomposes X, Y and X^+ into components. The algorithm is presented in Appendix I. The basic ideas of the algorithm are illustrated in Figure 1.

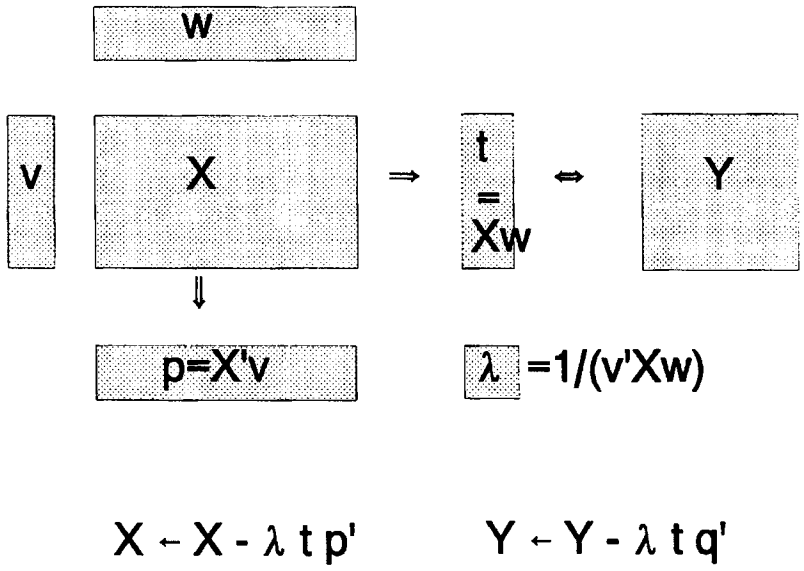

Figure 1. Matrices and vectors in the decomposition algorithm

The figure depicts the situation at each step. We determine a vector **w** that gives a 'good' score vector **t** which is computed as $\mathbf{t} = \mathbf{Xw}$. Similarly a vector **v** is determined such that the loading vector **p** given by $\mathbf{p} = \mathbf{X}^T\mathbf{v}$ is a 'good' one.

The results of the algorithm are approximations $\mathbf{X}_{(A)}$, $\mathbf{X}^+_{(A)}$ and $\hat{\mathbf{Y}}_{(A)}$ to **X**, $\mathbf{X}^+$ and **Y** respectively that are given by

$$\mathbf{X}_{(A)} = \lambda_1\mathbf{t}_1\mathbf{p}_1^T + \cdots + \lambda_A\mathbf{t}_A\mathbf{p}_A^T \tag{9}$$

$$\mathbf{X}^+_{(A)} = \lambda_1\mathbf{r}_1\mathbf{s}_1^T + \cdots + \lambda_A\mathbf{r}_A\mathbf{s}_A^T \tag{10}$$

$$\hat{\mathbf{Y}}_{(A)} = \lambda_1\mathbf{t}_1\mathbf{q}_1^T + \cdots + \lambda_A\mathbf{t}_A\mathbf{q}_A^T \tag{11}$$

What are a good score vector $\mathbf{t}$ and a good loading vector $\mathbf{p}$ in these approximations? We shall discuss this question in the case of PCA and PLS. One way to look at PCA in the present context is as follows. $\mathbf{X}$ and $\mathbf{Y}$ are the same, $\mathbf{X}=\mathbf{Y}$, and $\mathbf{v}$ is determined such that the length of the loading vector is as large as possible:

$$\text{maximize } |\mathbf{p}|^2 = \text{maximize } |\mathbf{X}^T\mathbf{v}|^2 \text{ for } |\mathbf{v}| = 1$$

The vector $\mathbf{v}$ can be determined e.g. by the NIPALS algorithm. $\mathbf{w}$ is then determined as scaled $\mathbf{p}$, $\mathbf{w}=\mathbf{p}/|\mathbf{p}|$. We also get the PCA solution if the score vector $\mathbf{t}$ is determined such that it has maximal length. In the PLS context the mathematical model is a regression model. Here the aim is to determine a score vector $\mathbf{t}$ that describes the matrix of response values $\mathbf{Y}$ as 'well' as possible. Equation (8) is the PLS criterion for determining the weight vector $\mathbf{w}$. When $\mathbf{w}$ has been determined, the weight vector $\mathbf{v}$ is determined as $\mathbf{v}=\mathbf{t}/|\mathbf{t}|$.

Other criteria of good loading and score vectors can be used in the algorithm. The vectors $\mathbf{w}$ and $\mathbf{v}$ can be chosen independent of each other or, as in the case of PCA and PLS, dependent on one another. In many applications where we determine both vectors independently we do it for the first few components. Typically one of the criteria becomes small after a few components have been selected. In these cases we can continue with the algorithm and select $\mathbf{w}$ or $\mathbf{v}$ only, whichever gives a contribution to the modelling task.

THE ALGORITHM IN THE CASE OF PCA

It is useful to look more closely at the algorithm in the case of PCA. The general interpretation of the magnitudes computed is similar to the one we have in PCA. $\mathbf{w}$ in step 1 of the algorithm is determined from

$$\text{maximize } \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{w} \quad \text{for } |\mathbf{w}|$$

The solution is given as an eigenvalue problem

$$\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{w} = \sigma^4\mathbf{w} \quad \text{for } |\mathbf{w}| = 1 \tag{12}$$

The eigenvalue is written in this way because we need $(\sigma^4)^{1/4}=\sigma$. From this and $\mathbf{t}=\mathbf{X}\mathbf{w}$ it is easy to show that

$$|\mathbf{t}| = \sigma, \quad \mathbf{v}=\mathbf{t}/\sigma, \quad \mathbf{p}=\sigma\mathbf{w}, \quad \mathbf{q}=\mathbf{p}, \quad \mathbf{v}^T\mathbf{X}\mathbf{w}=\sigma, \quad \lambda=1/\sigma, \quad \mathbf{r}=\mathbf{w}, \qquad \mathbf{s}=\mathbf{t}/\sigma$$

Thus we have $\mathbf{w}=\mathbf{p}/\sigma=\mathbf{r}$ and $\mathbf{v}=\mathbf{t}/\sigma=\mathbf{s}$. Both $(\mathbf{w}_a)$ and $(\mathbf{t}_a)$ are orthogonal vectors. The criterion (12) is in the form of a squared variance, $\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{w}=(\sigma^2)^2$. This observation is important in relation to the numerical stopping criterion of the algorithm. If we use the size of $\sigma=\mathbf{v}^T\mathbf{X}\mathbf{w}$ as a criterion, we cannot allow σ to be too small. We select $\mathbf{w}$ according to (12), and if σ is too small, the eigenvalue σ^4 may fall below the numerical precision of the computer. If data are autoscaled, it is usually in order to stop when σ becomes smaller than 10^{-4} or σ^2 smaller than 10^{-8}. Note the expressions for $\mathbf{X}$ and $\mathbf{X}^+$. For PCA the vectors $(\mathbf{r}_a)$ and $(\mathbf{s}_a)$ in (10) are of length one, while this is not true in the general case. The lambda's, λ_a, are the reciprocals of standard deviations, $\lambda_a=1/\sigma_a$. Note that (12) is used to determine $\mathbf{w}$. This choice has nothing to do with the numerical accuracy of the algorithm, i.e. the precision of the terms (9)–(11).

THE ALGORITHM IN THE CASE OF PLS

Here $\mathbf{w}_a$ is determined according to (8). The mathematical solution is determined from

$$\mathbf{X}^T(\mathbf{Y}\mathbf{Y}^T)\mathbf{X}\mathbf{w} = \sigma^4\mathbf{w} \quad \text{for } |\mathbf{w}| = 1 \tag{13}$$

Since we do not have a row criterion, we choose $\mathbf{v}_a = \mathbf{t}_a/|\mathbf{t}_a|$. We therefore have $\mathbf{s}_a = \lambda_a\mathbf{t}_a$ and $\lambda_a = 1/|\mathbf{t}_a|$, and one can show that the vectors have the following geometric properties:

$$\mathbf{w}_a^T\mathbf{w}_b = 0 \quad \text{and} \quad \mathbf{t}_a{}^T\mathbf{t}_b = 0 \quad \text{for} \quad a \neq b.$$

These properties are given in Reference 10. They show that $(\mathbf{w}_a)$ is a set of orthonormal vectors and $(\mathbf{t}_a)$ is a set of orthogonal vectors. If all components are selected, $\mathbf{X}$ is of full rank and $N > K$, then from $\mathbf{P}^T\mathbf{R} = \Lambda^{-1}$ we have

$$\mathbf{X}^T\mathbf{X} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \cdots + \mathbf{p}_K\mathbf{p}_K^T = \mathbf{P}\mathbf{P}^T \tag{14}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \lambda_1^2\mathbf{r}_1\mathbf{r}_1^T + \cdots + \lambda_K^2\mathbf{r}_K\mathbf{r}_K^T = \mathbf{R}\Lambda^2\mathbf{R}^T \tag{15}$$

where Λ is a diagonal matrix with the inverse of the length of $\mathbf{t}_a$, $1/|\mathbf{t}_a|$, in its diagonal. Steps 3–5 of the algorithm can be viewed as a decomposition of $\mathbf{X}$ into orthogonal components which are derived from the criterion vectors $(\mathbf{w}_a)$. Equation (15) shows that for a K-vector $\mathbf{x}_0$ we can write

$$\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_0 = (\mathbf{x}_0^T\mathbf{r}_1)^2/(\mathbf{t}_1^T\mathbf{t}_1) + \cdots + (\mathbf{x}_0^T\mathbf{r}_K)^2/(\mathbf{t}_K^T\mathbf{t}_K) \tag{16}$$

Normally we do not extract all components from $\mathbf{X}$. If only A components are selected, K is replaced by A in the above formulae. Note that the formulae (9) and (14)–(16) are the same for any column criterion used. In the general case the weight vectors $(\mathbf{w}_a)$ need not be orthogonal (but $(\mathbf{t}_a)$ always are).

We shall look more closely at (9) and consider how new samples are transformed to score space and conversely. If we consider a row in $\mathbf{X}$, we get

$$\mathbf{x}^i = \mathbf{t}^i\mathbf{P}^T \quad \text{or} \quad (\mathbf{x}^i)^T = \mathbf{P}(\mathbf{t}^i)^T$$

Similarly we have

$$\mathbf{t}^i = \mathbf{x}^i\mathbf{R} \quad \text{or} \quad (\mathbf{t}^i)^T = \mathbf{R}^T(\mathbf{x}^i)^T$$

This shows that the estimated transformations between score space and sample space can be written as

$$\mathbf{x} = \mathbf{P}\mathbf{t} \quad \text{and} \quad \mathbf{t} = \mathbf{R}^T\mathbf{x} \tag{17}$$

Thus if we have a sample vector, the associated score vector can be computed from (17), and conversely if we know a score vector. Thus the algorithm can be viewed as determining transformation matrices that transform samples to scores and conversely.

APPLICATION OF PCA

In PCA we have some working techniques that are useful in the data analysis. We shall consider some of them and show how they can be applied in the present context.

Sizes

In PCA we look at the variance selected at each step. In the PCA context it is σ^2. In the context

of the present algorithm it is what we call H-precision:

$$\text{H-precision} = (\mathbf{v}^{\mathrm{T}}\mathbf{X}\mathbf{w})^2 \tag{18}$$

When we use both column and row criteria, the value of $\mathbf{v}^{\mathrm{T}}\mathbf{X}\mathbf{w}$ need not be positive. However (18) tells us how precise the component is. If it is close to zero, it is time to stop extracting components. In PCA we are extracting the variation in X. Therefore we compute the relative amount of variation selected,

$$\mathrm{tr}(\mathbf{X}_{(A)}^{\mathrm{T}}\mathbf{X}_{(A)})/\mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{X}) \tag{19}$$

and similarly for the inverse of X,

$$\mathrm{tr}(\mathbf{X}_{(A)}^{+\mathrm{T}}\mathbf{X}_{(A)}^{+})/\mathrm{tr}(\mathbf{X}^{+\mathrm{T}}\mathbf{X}^{+}) = \mathrm{tr}((\textstyle\sum^{A}\lambda_a\mathbf{r}_a\mathbf{s}_a^{\mathrm{T}})^{\mathrm{T}}(\sum^{\mathrm{A}}\lambda_a\mathbf{r}_a\mathbf{s}_a^{\mathrm{T}}))/\mathrm{tr}((\textstyle\sum\lambda_a\mathbf{r}_a\mathbf{s}_a^{\mathrm{T}})^{\mathrm{T}}(\sum\lambda_a\mathbf{r}_a^{\mathrm{T}})) \tag{20}$$

Here tr( ) is the trace function. In principle we would like (19) to be as close to one (100 per cent) as possible and (20) as close to zero as possible.

Prediction error variance

In multivariate regression analysis the prediction error variance is given by

$$\mathrm{var}(\mathbf{y}(\mathbf{x}_0) - \hat{\mathbf{y}}(\mathbf{x}_0)) = \mathbf{S}[1 + \mathbf{x}_0^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{+}\mathbf{x}_0] \tag{21}$$

Here S is the residual variance matrix given by

$$\mathbf{S} = (\mathbf{Y} - \hat{\mathbf{Y}})(\mathbf{Y} - \hat{\mathbf{Y}})^{\mathrm{T}}/(N - A) \tag{22}$$

$\mathbf{x}_0$ is a new sample, $\hat{\mathbf{y}}(\mathbf{x}_0)$ is an estimated response value for $\mathbf{x}_0$ and $\mathbf{y}(\mathbf{x}_0)$ is the observed response value. $\hat{\mathbf{Y}}$ is the estimated response matrix for the given model. The matrix $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{+}$ is the (generalized) inverse of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, where X is the matrix used in the analysis. In the PCA context the sizes of the residual variance matrix and $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{+}$, when A components have been extracted, are given by

$$\mathrm{tr}(\mathbf{S}) = (\textstyle\sum_{A-1}(1/\lambda_i^2))/(N - A) \quad \text{and} \quad \mathrm{tr}((\mathbf{X}^{\mathrm{T}}\mathbf{X})^{+}) = \sum^{A}\lambda_i^2$$

When we study the number of components in PCA, we consider the following expression that we call the *H-error*:

$$\text{H-error} = (\textstyle\sum_{A+1} 1/\lambda_i^2)(1 + \sum^{A}\lambda_i^2)/(N - A) \tag{23}$$

The motivation for this function is expression (21). The prediction error variance depends on the sample used. However, it is natural to look at the sizes of the matrices involved, because the matrices are positive definite. In the algorithm the H-error is computed as

$$\text{H-error} = |\mathbf{Y} - \hat{\mathbf{Y}}|^2(1 + \textstyle\sum^{A}\lambda_i^2)/(N - A) \tag{24}$$

Expression (24) is the same as (23) in the case of PCA. We have found it useful to work with (24) when determining the number of components to use in the algorithm. When (24) has approached zero or is at its lowest value, it is time to stop extracting components.

Figure 2 plots (24) against the number of components selected. The minimum value of H-error is at eight or nine components, suggesting that only eight or nine components should be used. Figure 3 shows the increase in the size of $\mathbf{X}_{(i)}$. If nine components are used, we are using some 60% of the total size of X. In Figure 4 we show the relative increase in the size of the inverse up to 35 components. The size of the matrix $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{+}$ increases rapidly when the number of component increases from 35 to 40. If the plot contained all 40 components, we would not be

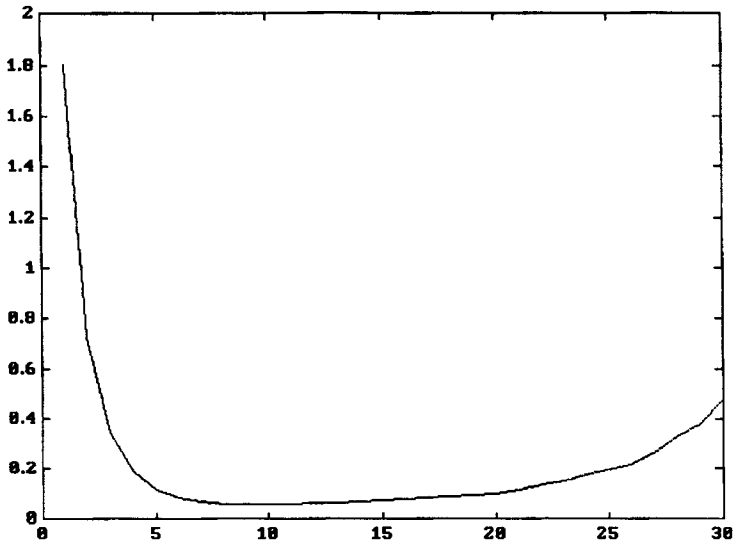

Figure 2. H-error versus number of components

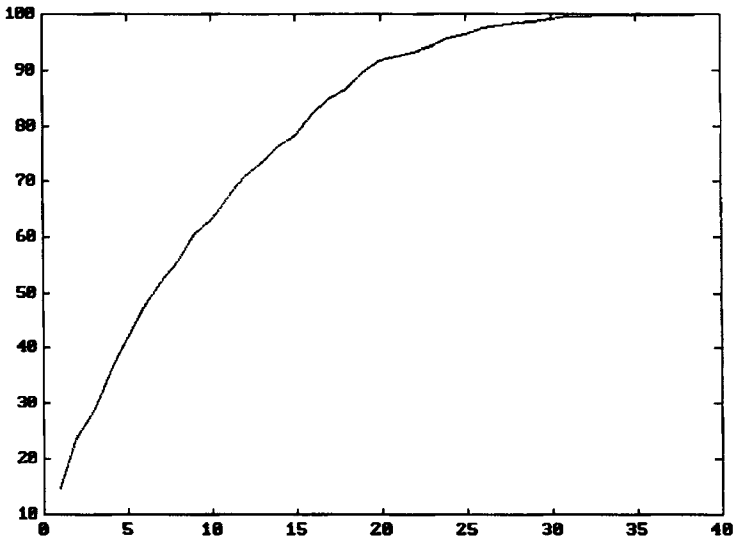

Figure 3. Percentage increase in the size of X $100\mathrm{tr}(X_a^T X_a)/\mathrm{tr}(X^T X)$

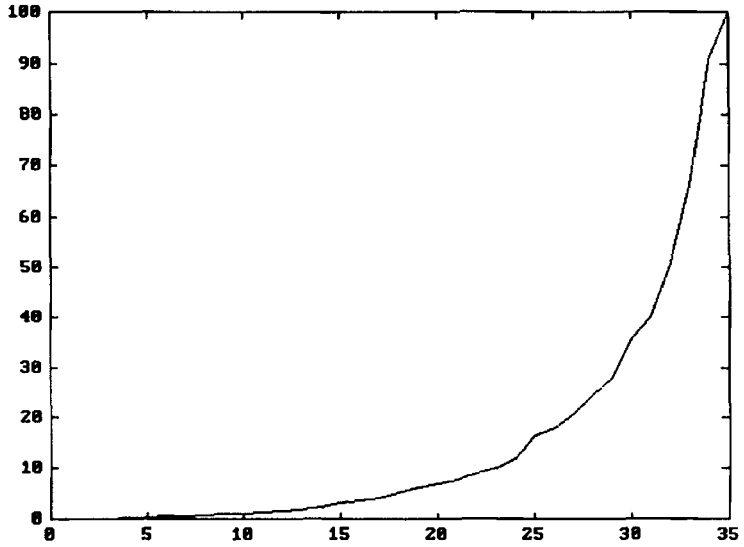

Figure 4. Percentage increase in the size of the inverse $\mathbf{X}^{*}_{(A)}$

able to see anything because of the large value of the last components. From Figure 4 we see that we only use a few per cent of the inverse if we use nine components.

APPLICATION OF PLS REGRESSION

We shall consider here some basic modelling questions that arise when we are carrying out a PLS regression analysis.

The noise level

There are several possible approaches to identify the noise level. The one we shall pursue here is to identify the prediction variance and bias associated with a sample (a row in X). For any random variable we can write the mean square error as the sum of variance and squared bias,

$$E|Y-\hat{Y}|^2 = \text{var}(Y) + (E(Y-\hat{Y}))^2 = \text{variance} + \text{squared bias} \tag{25}$$

When we have a sample $\mathbf{x}_0$, we estimate the predicted value as

$$\hat{y}(\mathbf{x}_0) = x_{10}\beta_1 + x_{20}\beta_2 + \cdots + x_{K0}\beta_K + \varepsilon$$

Instead of the theoretical regression coefficients (β_i), we insert the estimated ones and let $\varepsilon = 0$. If the observed response values associated with $\mathbf{x}_0$ are denoted by $\hat{\mathbf{y}}(\mathbf{x}_0)$, we are interested in the decomposition (25) for $E|\mathbf{y}(\mathbf{x}_0) - \hat{\mathbf{y}}(\mathbf{x}_0)|^2$. The terms in (25) depend on how many components have been extracted by the algorithm. In Reference 8 it is shown that if A components have been selected, then

$$E[(\mathbf{y}(\mathbf{x}_0) - \hat{\mathbf{y}}(\mathbf{x}_0))(\mathbf{y}(\mathbf{x}_0) - \hat{\mathbf{y}}(\mathbf{x}_0))^{\mathrm{T}}] \approx \mathbf{S}(1 + |\Sigma^A \lambda_a (\mathbf{r}_a{}^{\mathrm{T}}\mathbf{x}_0)\mathbf{s}_a|^2) + (\Sigma_{A+1} \lambda_a (\mathbf{r}_a^{\mathrm{T}}\mathbf{x}_0)\mathbf{q}_a)(\Sigma_{A+1} \lambda_a (\mathbf{r}_a^{\mathrm{T}}\mathbf{x}_0)\mathbf{q}_a)^{\mathrm{T}} \tag{26}$$

The first term on the right-hand side of (26) is the variance and the second is the squared bias. Here S is the residual variance (22). The total variance associated with $\mathbf{x}_0$, when A components have been extracted is

$$\text{total variance} = \text{tr}(\mathbf{S})(1 + |\Sigma^A \lambda_a (\mathbf{r}_a^T \mathbf{x}_0)\mathbf{s}_a|^2) = \text{tr}(\mathbf{S})[1 + |(\mathbf{X}_{(A)}^+)^T \mathbf{x}_0|^2] \tag{27}$$

and the total squared bias associated with $\mathbf{x}_0$ at A components is

$$\text{total squared bias} = |\Sigma_{A+1} \lambda_a (\mathbf{r}_a^T \mathbf{x}_0)\mathbf{q}_a|^2 \tag{28}$$

The basic task of the modelling procedure is to determine the number of components where the squared bias has reached the zero level, and the noise level in the data (27) has reached the lowest possible level.

Note that (26) depends on the object or sample used. It is natural to require that the average of (26) over all samples should decrease, when new components are selected. In Reference 11 it is shown that the average value of (26) decreases when

$$\lambda_A^2 (\mathbf{y}_j^T \mathbf{s}_A)^2 (\mathbf{t}_A^T \mathbf{t}_A) - s_{A,j}^2 N/(N-A) \geqslant 0, \quad j = 1, \ldots, M \tag{29}$$

The first term of (29) is the variation in the jth response variable explained by the Ath component; $s_{A,j}^2$ is the residual variance when A components have been selected. If (29) is not satisfied for any response variable, the average of (26) will increase. Therefore if the difference in (29) is close to zero or starts to be negative, it is time to stop modelling.

Figure 5 plots (27) against the number of components. The sample $\mathbf{x}_0$ here is the sample with the largest leverage value. It is seen that the lowest value is reached at some nine to 13 components. Figure 6 plots (28) against the number of components. It shows that the squared bias has reached the zero level at seven components. In Figure 7 we plot the sum of the left-hand side of (29) against A, the number of components. Here $M = 2$, so we sum over two values. The zero level is reached at some nine components. Therefore this figure suggests that

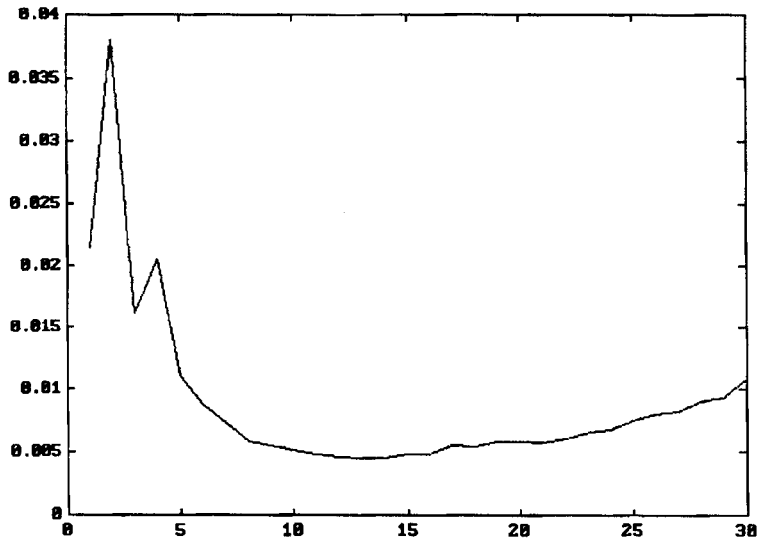

Figure 5. Prediction variance var($y(\mathbf{x}_{max})$) for the most extreme sample

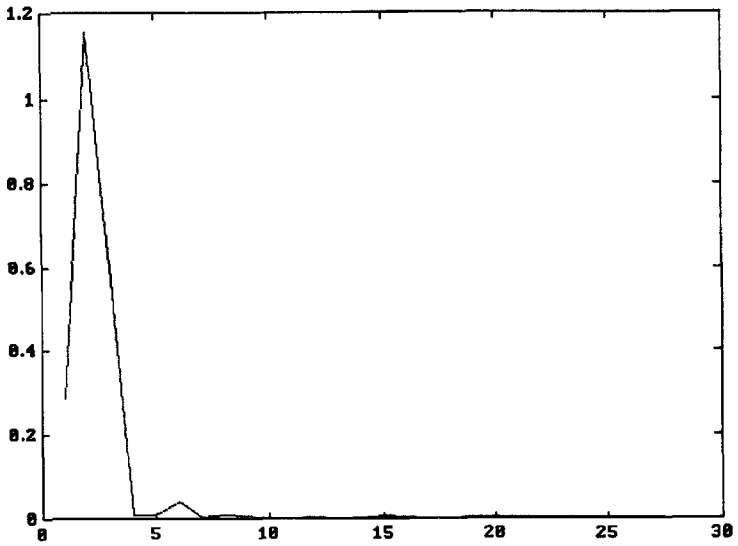

Figure 6. Squared bias for the most extreme sample

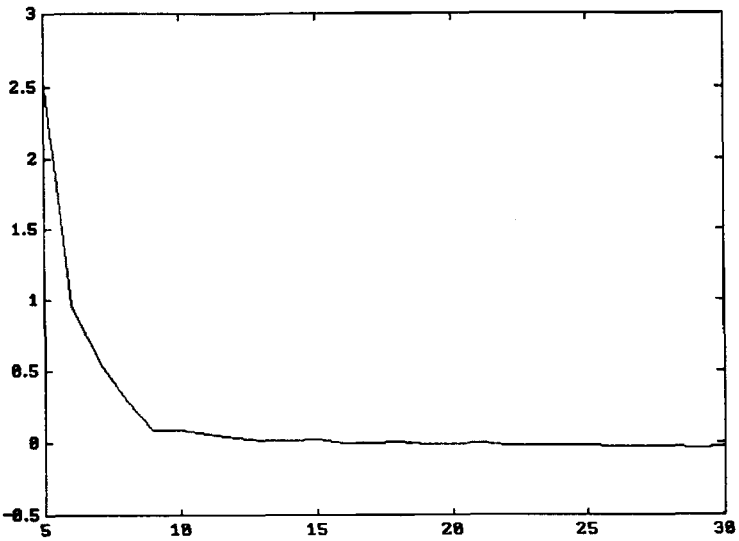

Figure 7. The difference $\lambda_2(\mathbf{t}^T\mathbf{t})(\mathbf{q}^T\mathbf{q}) - s^2N/(N-A)$, a requirement for improvement in prediction

we should select nine components. Figure 7 indicates that some more components could be selected. However, the values of (29) are small for more than nine components. A conclusion from Figure 5–7 is that nine components will be sufficient.

Outliers

We use the regression formulae to determine outliers. If all necessary components have been selected so that (28) can be considered to be zero, we get from (26)

$$E(|\mathbf{y}(\mathbf{x}_0)-\hat{\mathbf{y}}(\mathbf{x}_0)|^2)=\mathrm{tr}(\mathbf{S})(1+|\textstyle\sum^A \lambda_a(\mathbf{r}_a^{\mathrm{T}}\mathbf{x}_0)\mathbf{s}_a|^2) \tag{30}$$

If we apply the normal distribution, we should for 95% of the samples get the following inequality

$$|\mathbf{y}(\mathbf{x}_0)-\hat{\mathbf{y}}(\mathbf{x}_0)| \leq 1{\cdot}96[\mathrm{tr}(\mathbf{S})(1+|\textstyle\sum^A \lambda_a(\mathbf{r}_a^{\mathrm{T}}\mathbf{x}_0)\mathbf{s}_a|^2)]^{1/2} \tag{31}$$

We check (31) for $\mathbf{x}_0=\mathbf{x}^i$, $i=1,\ldots,N$. If some of the samples in the data do not satisfy (31), they are potential outliers. New samples are checked in the same way.

Sensitivity analysis

We use similarly the formulae from regression analysis to compute various sensitivity measures. These sensitivity measures are designed to give information about changes due to leaving one sample out of the computations. An important magnitude in sensitivity analysis is the leverage value

$$h_i=\mathbf{x}^i(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{x}^{i\mathrm{T}} \tag{32}$$

In our computations we compute the leverage value h_i as

$$h_i=|\mathbf{X}_{(A)}^{+\mathrm{T}}\mathbf{x}^{i\mathrm{T}}|^2=|\textstyle\sum^A \lambda_a(\mathbf{x}^i\mathbf{r}_a)\mathbf{s}_a|^2 \tag{33}$$

Expression (32) is only valid for non-singular matrices. Thus we make an error by using (33) instead of (32). However, our experience is that these terms are approximately equal.

In empirical data analysis it is useful to work with 'externally studentized residuals'. We shall explain this term briefly. For the sake of simplicity suppose that there is only one response variable. Let y_i be the observed value of the response variable, $\hat{y}_i$ be the estimated response value, and $\hat{y}_{(i)}$ be the estimated response value, when the ith sample is excluded from the computations. The externally studentized residual is given by

$$|y_i-\hat{y}_i|/[s_{(i)}(1-h_i)^{1/2}] \tag{34}$$

where $s_{(i)}$ is the residual standard deviation that is obtained when the ith sample is excluded from the analysis.

Figure 8 shows the leverage values (33) for $A=9$. In empirical work we often say that we have a large leverage value h_i, if $h_i>2A/N$. Here $2A/N=2\times 9/50=0{\cdot}36$. All 50 leverage values here are smaller than 0·36. In Figure 9 the externally studentized residuals are plotted. They follow approximately a t-distribution with degrees of freedom equal to $N-A-1$. Figure 9 gives the numerical values. It shows that none of them is significant. Thus this analysis confirms that there are no outliers in the data. In the case where such figures suggest that there are clear outliers, the computations should be carried out again without these samples.

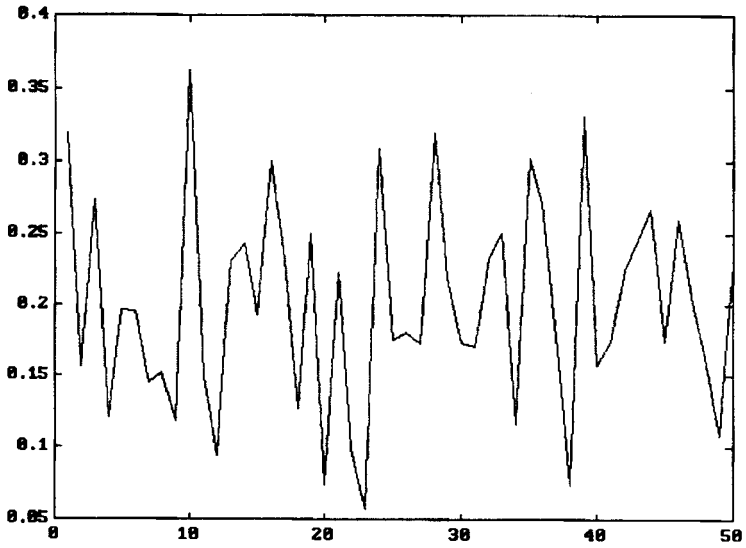

Figure 8. Leverage values $h_i = |\mathbf{X}^{+T}\mathbf{x}^i|^2$

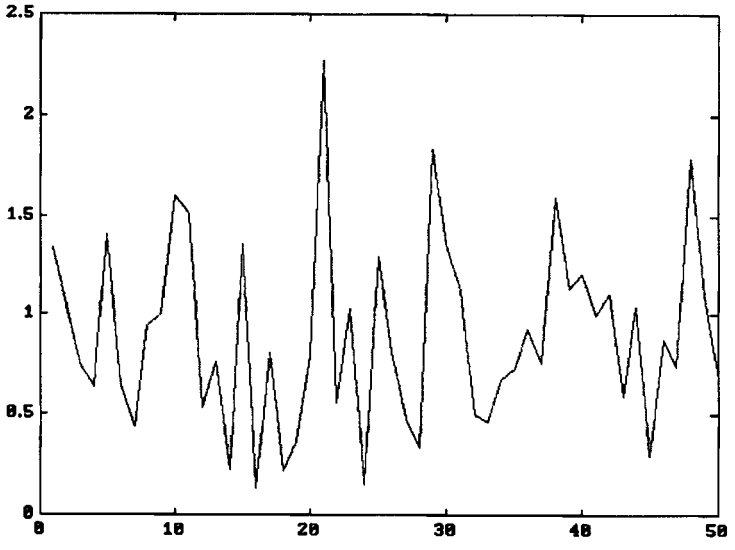

Figure 9. Studentizes residual, ith sample excluded

AN EXAMPLE OF ROW AND COLUMN CRITERIA

We shall consider here an example where we use both a column criterion and a row criterion. Here we combine the criteria of PCA and PLS. We use (7) to determine $\mathbf{v}$ such that we get as large a loading vector as possible. This is the criterion of PCA. We use (8) to determine $\mathbf{w}$, which is the PLS criterion. These choices at each step generate a decomposition (9)–(11) where the vectors $(\mathbf{t}_i)$ *and* $(\mathbf{p}_i)$ are not orthogonal. This decomposition can be studied more closely in various ways. Here we shall only consider the following two figures.

Figure 10 shows a plot of the H-error (24) against the number of components. The minimum is obtained at nine components. Thus here also we should use only nine components as before. The curve is not as smooth as that curve in Figure 2. This is due to the fact that the vectors selected are not orthogonal. It is a general feature that the curves are not smooth when two independent criteria are used. We also see that if we select too many components, we may have problems with numerical accuracy in those cases where independent determinations of $\mathbf{v}$ and $\mathbf{w}$ are required. In the example here the value of $(\mathbf{vXw})^2$ at the 22nd component becomes very small. If we continue beyond the 22nd component, the numerical results become unreliable. If it is desired to go beyond the 22nd component, one should continue using only one of the two criteria, the one that contributes most to the modelling task. Figure 11 shows the squared bias (28) for the most extreme sample. It corresponds to Figure 6 and shows that the squared bias has reached the zero level already at seven components. Considering the squared bias for some other samples (those having the largest distance to the one used in Figure 11) confirms that we should not select more than nine components.

In many cases it is advantageous to have one set of the vectors $(\mathbf{t}_a)$ or $(\mathbf{p}_a)$ an orthogonal set of vectors. If it is required that $(\mathbf{t}_a)$ are orthogonal, $(\mathbf{t}_a^{\mathrm{T}}\mathbf{t}_b)=0$ for $a \neq b$, one can show that $\mathbf{p}_a$ must be computed as $\mathbf{p}_a=\mathbf{X}_{a-1}^{\mathrm{T}}\mathbf{t}_a/|\mathbf{t}_a|$. Similarly, if $(\mathbf{p}_a)$ are required to be orthogonal, $\mathbf{t}_a$ must be computed as $\mathbf{t}_a=\mathbf{X}_{a-1}\mathbf{p}_a/|\mathbf{p}_a|$. There are several ways to resolve this. One way is to optimize

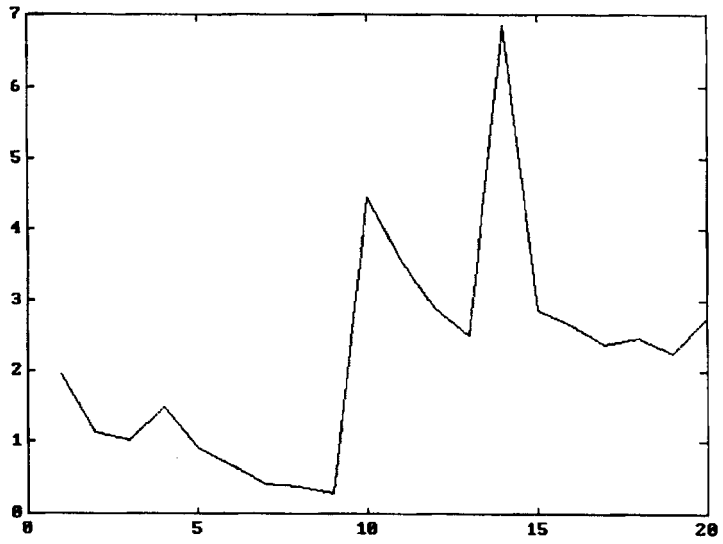

Figure 10. H-error versus number of components

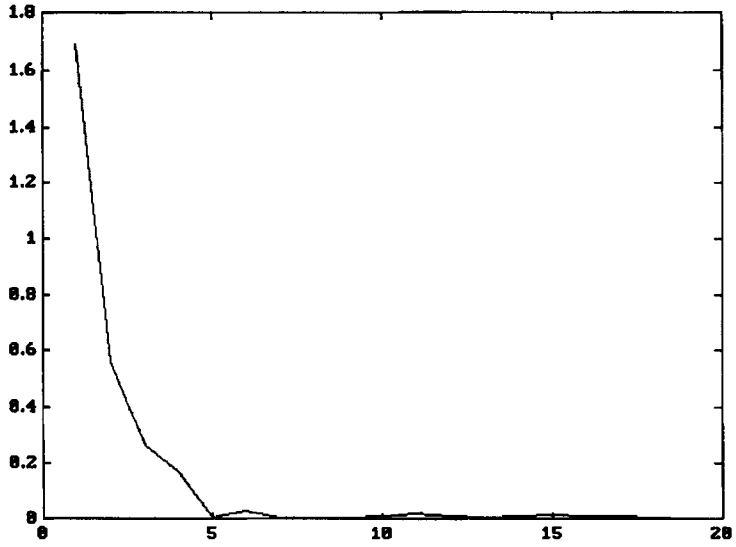

Figure 11. Squared bias for the most extreme sample

a product of the row and column criteria. For example, in the case of PCA for rows and PLS for columns we can maximize the product of the two criteria:

$$\text{maximize } (\mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{v})(\mathbf{w}^T\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{w})$$

If $(\mathbf{t}_a)$ are required to be orthogonal, $\mathbf{v}$ is replaced in this expression by $\mathbf{Xw}$, while if $(\mathbf{p}_a)$ are to be orthogonal, $\mathbf{w}$ is replaced by $\mathbf{X}^T\mathbf{v}$.

LINEAR–QUADRATIC MODELS

Suppose that the theoretical model is given by

$$y_i = \beta_i^T\mathbf{x} + \mathbf{x}^T\mathbf{C}_i\mathbf{x} + \text{residual}, \quad i = 1, 2, ..., M \tag{35}$$

Here $\mathbf{x}$ is a K-vector. If K is equal to 100, we are talking about a model of medium size compared with the ones we meet in practice. In that case the number of elements in each $(\beta_i, \mathbf{C}_i)$ is $100 + 100 \times 101/2 = 5150$. In such cases it is not feasible to estimate the parameters $(\beta_i, \mathbf{C}_i)$ directly. Instead we seek to determine directions or components that can more adequately identify the surface that describes the response values.

Let us present an example. Consider the mathematical model

$$\begin{aligned} y &= 0{\cdot}95x_1 + 2{\cdot}01x_2 + 0{\cdot}5x_1^2 + 2x_1x_2 + 2x_2^2 + \text{residual} \\ &= (x_1 + 2x_2) + 0{\cdot}5(x_1 + 2x_2)^2 + (-0{\cdot}5x_1 + 0{\cdot}01x_2) + \text{residual} \\ &= t_1 + 0{\cdot}05t_1^2 + 0{\cdot}5t_2 + \text{residual} \end{aligned}$$

where

$$t_1 = x_1 + 2x_2 \quad \text{and} \quad t_2 = -0{\cdot}05x_1 + 0{\cdot}01x_2$$

From a mathematical point of view we can work equally well with the t-variables as with the x-variables. However, it can happen that the residual values are so large that the component t_2 is not worth keeping, because the sum of t_2-values and residual values is of the same size as the residual values alone. One can state that in this case the component t_1 alone appropriately identifies the surface, in which case we can state that the data are appropriately described by a parabola in the direction of t_1.

The algorithm determines a transformation matrix $\mathbf{R}$ such that the new components are given by $\mathbf{t} = \mathbf{R}^T\mathbf{x}$; see (17). Inserting this into (35), we get

$$\begin{aligned} y_i &= \beta_i\mathbf{x} + \mathbf{x}^T\mathbf{C}_i\mathbf{x} = \beta_i^T\mathbf{P}\mathbf{R}^T\mathbf{x} + \mathbf{x}^T\mathbf{R}\mathbf{P}^T\mathbf{C}_i\mathbf{P}\mathbf{R}^T\mathbf{x} \\ &= (\mathbf{P}^T\beta_i)^T(\mathbf{R}^T\mathbf{x}) + (\mathbf{R}^T\mathbf{x})(\mathbf{P}^T\mathbf{C}_i\mathbf{P})(\mathbf{R}^T\mathbf{x}) = \delta_i^T\mathbf{t} + \mathbf{t}^T\Theta_i\mathbf{t} \end{aligned} \tag{36}$$

where

$$\mathbf{P} = (\mathbf{R}^T)^{-1}, \qquad \delta_i = \mathbf{P}^T\beta_i \qquad \Theta_i = \mathbf{P}^T\mathbf{C}_i\mathbf{P}$$

If all K components are selected, there is a one-to-one relationship between $\mathbf{t}$ and $\mathbf{x}$. However frequently not all components are selected. This means that $\mathbf{P}$ and $\mathbf{R}$ only have A columns, $\mathbf{P} = \mathbf{P}_A$ and $\mathbf{R} = \mathbf{R}_A$. Note that $\mathbf{t}$ here will be an A-vector.

The criterion for selecting components

We shall now present a criterion for determining 'quadratic' components. This theory is presented in the univariate case in Reference 12 and in the multivariate case in Reference 8. Only the numerical criterion will be presented here and a discussion of the experience in using it.

When the ath component t_a is added to the model, there are in fact $a+1$ variables being added to the model, namely t_a, t_a^2, t_at_1, t_at_2, ..., t_at_{a-1}. In order to simplify the notation, let $\otimes$ denote the elementwise product of two vectors, e.g.

$$\mathbf{t}_1 \otimes \mathbf{t}_2 = (t_{i1} \times t_{i2}) = (t_{11} \times t_{12},\ t_{21} \times t_{22},\ t_{31} \times t_{32},\ \ldots,\ t_{N1} \times t_{N2}).$$

Using this notation, the following matrices are computed:

$$\mathbf{V}_k = \mathbf{Y} \otimes \mathbf{t}_k = (y_{ij} \times t_{ik}), \quad k = 1, \ldots, a-1$$

$$\mathbf{H} = \mathbf{X}^T(\mathbf{Y}\mathbf{Y}^T + \mathbf{V}_1\mathbf{V}_1^T + \cdots + \mathbf{V}_{a-1}\mathbf{V}_{a-1}^T)\mathbf{X}$$

$$\mathbf{G}_j = \mathbf{X}^T\mathrm{diag}(\mathbf{y}_j)\mathbf{X}, \quad j = 1, \ldots, M$$

Here $\mathrm{diag}(\mathbf{y}_j)$ is the diagonal matrix with $\mathbf{y}_j$ in its diagonal. The numerical task is

$$\text{maximize } \mathbf{w}^T\mathbf{H}\mathbf{w} + \textstyle\sum_j(\mathbf{w}^T\mathbf{G}_j\mathbf{w})^2 \quad \text{for } |\mathbf{w}| = 1 \tag{37}$$

This is a non-linear expression in $\mathbf{w}$. It is solved by considering the function

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{H}\mathbf{w} + \textstyle\sum_j(\mathbf{w}^T\mathbf{G}_j\mathbf{w})^2 - \lambda(\mathbf{w}^T\mathbf{w} - 1)$$

Differentiating $f(\mathbf{w})$, we get

$$f'(\mathbf{w}) = 2\mathbf{H}\mathbf{w} + \textstyle\sum_j[2(\mathbf{w}^T\mathbf{G}_j\mathbf{w})\mathbf{G}_j\mathbf{w}] - \lambda 2\mathbf{w}$$

A stationary point $f'(\mathbf{w}) = 0$ is therefore given by

$$\mathbf{w} = (\mathbf{H}\mathbf{w} + \textstyle\sum_j(\mathbf{w}^T\mathbf{G}_j\mathbf{w})\mathbf{G}_j\mathbf{w})/(\mathbf{w}^T\mathbf{H}\mathbf{w} + \textstyle\sum_j(\mathbf{w}^T\mathbf{G}_j\mathbf{w})^2) \tag{38}$$

This shows that (37) is solved by iterating (38) in the same way as in the case of the power method of determining the largest eigenvalue and associated vector. Good initial values can be obtained by first computing the largest eigenvalue and associated eigenvector for **H** and then using the eigenvector as a starting vector on the right-hand side of (38). Numerical experience on several examples, large and small, has shown that convergence is reached with the same speed as the normal power method of computing eigenvalues/vectors, i.e. convergence is normally reached within ten to 30 iterations of (38). Since the iterations of (38) are very fast, the amount of computer time needed to iterate (38) is very small on modern computers. This means that the method can handle a large number of components on even small computers. A MATLAB algorithm carrying out these computations is presented in Appendix III.

As mentioned above, introducing a new component t_a means that there are $a+1$ new variables being added to the model, namely t_a, t_a^2, $t_a t_1$, $t_a t_2$, ..., $t_a t_{a-1}$. Usually not all these variables are wanted in the model. It is important to look at the results and check which ones should be eliminated. This can be done in several ways, e.g. by checking the inequality (29) for each numerical component, i.e. for $\mathbf{t} = \mathbf{t}_a$, $\mathbf{t} = \mathbf{t}_a \otimes \mathbf{t}_a$, $\mathbf{t} = \mathbf{t}_a \otimes \mathbf{t}_1$, $\mathbf{t} = \mathbf{t}_a \otimes \mathbf{t}_2$, ..., $\mathbf{t} = \mathbf{t}_a \otimes \mathbf{t}_{a-1}$. In the case where the inequality (29) is not satisfied, the corresponding term is excluded in the criterion above.

MODEL STRATEGIES

We have presented here a general algorithm for modelling data. The algorithm includes PCA, PLS and various types of extensions. Therefore it is useful to discuss some modelling strategies that one can follow when using the algorithm. We shall discuss here some principal issues of modelling.

(a) It can be useful to combine **X** and **Y** into in one matrix and use PCA on the total matrix. In the algorithm we select

$$\mathbf{X} \leftarrow (\mathbf{X}, \mathbf{Y}) \quad \text{and} \quad \mathbf{Y} \leftarrow (\mathbf{X}, \mathbf{Y})$$

The PCA analysis will reveal outliers in the data via sensitivity analysis.

(b) PCA analysis on **X** separately and **Y** separately will reveal outliers and groups of samples, if they exist. We can also use the non-linear extension of the algorithm to detect whether there are non-linearities in **X**- or **Y**-data. If the rank of the data is much smaller than the number of rows or columns, we should be careful in the subsequent analysis.

(c) PLS analysis shows how **Y** depends on **X**. The non-linear extension can indicate whether there are significant non-linear components.

(d) Plots of score vectors, loading vectors and score–response variables tell us how the data 'behave'.

(e) On the basis of the preliminary analysis we can identify the criteria to be used as row and column criteria. The following are some examples of choosing a row criterion for **v**.

1. Choose **v** to obtain as large a loading vector as possible.
2. Suppose we have a vector $\mathbf{x}_0$ and we want our next loading vector to be as close as possible to it. We determine **v** as: minimize $|\mathbf{x}_0 - \mathbf{X}^T\mathbf{v}|$. An example of such a vector can be 'a target sample', 'a reference sample' or some similar consideration.
3. Sometimes the samples can be ordered in some chronological order. We determine load vectors to reflect this.
4. Sometimes some samples are more important than others, so that **v** is now a weight vector

that reflects this importance. This is the case if we can insert a price or value (e.g. amount of 'gold indicators') associated with each sample.

5. Sometimes certain peaks in the data are more important than others. We let $\mathbf{v}$ reflect this importance and downweight samples far away from the important ones.
6. We compute $\mathbf{v}$ on the basis of the samples, e.g. $\mathbf{v}=(v_i)$ with $v_i=|\mathbf{x}_i|$, $i=1,...,N$, and then we scale $\mathbf{v}$ to have length one. This is an example where we want small occurrences to have smaller weights than do large occurrences.

There are many other possible considerations when determining the row criterion. Note that in the algorithm we cannot say that a loading vector $\mathbf{p}_a$ should be some prescribed vector. However, we can ask for $\mathbf{v}_a$ such that $\mathbf{X}^T\mathbf{v}_a$ resembles or correlates with a prescribed vector as well as possible.

Choosing both a row and a column criterion makes it possible to obtain results that are easier to interpret or more robust than with the standard approach (PCA or PLS). The price one is paying is that we usually need a few more components in these non-standard approaches. The score vectors ($\mathbf{t}_a$) and loading vectors ($\mathbf{p}_a$) will not be orthogonal. In these cases we sometimes see that after several components the λ_as are not positive. Although $\mathbf{X}$ is always reduced by rank one, there will be steps where the size of $\mathbf{X}$ increases. In these cases it may be useful to switch to orthogonal ($\mathbf{t}_a$) or orthogonal ($\mathbf{p}_a$), i.e. choose $\mathbf{t}_a$ such that $\mathbf{X}_{a-1}^T\mathbf{t}_a=\mathbf{0}$ or $\mathbf{p}_a$ such that $\mathbf{X}_{a-1}\mathbf{p}_a=\mathbf{0}$. This will always give a reduction in the size of $\mathbf{X}$.

CONCLUSIONS

We have presented here a general algorithm to decompose data into rank-one components. The algorithm includes as special cases most stepwise regression methods, PLS regression, PCA and different types of extensions. The algorithm allows for extensions that can be formulated as criteria for weighting columns (PLS) and/or rows (PCA). Some non-linear extensions are also included. The algorithm provides the user with tools to develop a model that appropriately reflects the given data and the theory. For each type of decomposition we can apply various measures of fit, sizes, measures of prediction, measures of sensitivity, etc. that can be used in judging the modelling procedure. The MATLAB code for the basic algorithm is given in Appendix III, where some examples of criteria for choosing the weight vectors $\mathbf{w}$ and $\mathbf{v}$ are also included. The basic algorithm for the non-linear extension is also given in Appendix III. The area of application of the present algorithm is very large and therefore only the basic skeleton of the algorithm has been included in the appendix. In applied work the algorithm is used for various tasks: to carry out preliminary data analyses, different types of regression analysis, PCA types of tasks, mixture of regression and PCA types of models, etc. The non-linear extension has been found to be efficient in detecting curvature in regression and PCA types of models.

APPENDIX I: THE DECOMPOSITION ALGORITHM

We shall present here the basic decomposition algorithm of the H-principle. It gives a decomposition of a matrix $\mathbf{X}$ into components in a form resembling that from PCA (or singular value decomposition).

We suppose that $\mathbf{X}$ is an $N\times K$ matrix and $\mathbf{Y}$ an $N\times M$ matrix. Let $\mathbf{X}_0=\mathbf{X}$, $\mathbf{Y}_0=\mathbf{Y}$, $\mathbf{E}_0=\mathbf{I}_K$ and $\mathbf{F}_0=\mathbf{I}_N$, where $\mathbf{I}_K$ and $\mathbf{I}_N$ are identity matrices of order K and N respectively. Let $\mathbf{B}_0=\mathbf{0}$, a $K\times M$ zero matrix. N is the number of objects or observations and K is the number of variables. M is the number of response variables.

The algorithm is as follows. For $a = 1, 2, ...$

1. *Determine the criterion vector* $\mathbf{w}_a$ *for columns*
   Determine $\mathbf{w}_a$ of length one according to some column criterion.
2. *Determine the criterion vector* $\mathbf{v}_a$ *for rows*
   Determine $\mathbf{v}_a$ of length one according to some row criterion.
3. *Compute score and loading vectors and a scaling constant*
   Score vector: $\mathbf{t}_a = \mathbf{X}_{a-1}\mathbf{w}_a$.
   Loading vector: $\mathbf{p}_a = \mathbf{X}_{a-1}^{\mathrm{T}}\mathbf{v}_a$.
   Scaling constant: $\lambda_a = 1/(\mathbf{v}_a^{\mathrm{T}}\mathbf{X}_{a-1}\mathbf{w}_a)$.
4. *Compute transformation vectors*
   Transformation vectors: $\mathbf{r}_a = \mathbf{E}_{a-1}\mathbf{w}_a$, $\mathbf{s}_{a-1} = \mathbf{F}_{a-1}\mathbf{v}_a$.
   Update $\mathbf{E}$ and $\mathbf{F}$: $\mathbf{E}_a = \mathbf{E}_{a-1} - \lambda_a\mathbf{r}_a\mathbf{p}_a^{\mathrm{T}}$, $\mathbf{F}_a = \mathbf{F}_{a-1} - \lambda_a\mathbf{s}_a\mathbf{t}_a^{\mathrm{T}}$.
5. *Reduce* $\mathbf{X}$ *by the selected component*
   New matrix $\mathbf{X}$: $\mathbf{X}_a = \mathbf{X}_{a-1} - \lambda_a\mathbf{t}_a\mathbf{p}_a^{\mathrm{T}}$.
6. *Adjust the response variables*
   Adjust $\mathbf{Y}$: $\mathbf{Y}_a = \mathbf{Y}_{a-1} - \lambda_a\mathbf{t}_a\mathbf{q}_a^{\mathrm{T}}$, $\mathbf{q}_a = (\mathbf{Y}_{a-1}^{\mathrm{T}}\mathbf{s}_a)$.
7. *Update the regression coefficients*
   Compute the improved solution: $\mathbf{B}_a = \mathbf{B}_{a-1} + \lambda_a\mathbf{r}_a\mathbf{q}_a^{\mathrm{T}}$.
8. *Determine whether a new component should be extracted*
   If a new component is to be extracted, replace a by $a + 1$ and start at step 1.

Discussion

The present algorithm is a very general one. It includes as special cases most algorithms in the literature that at each step reduce the rank of $\mathbf{X}$ by one. In theoretical statistics it contains the algorithms of stepwise regression, PLS regression, canonical correlation etc. In numerical analysis the QR algorithm, Gauss elimination,[13] methods of conjugate directions,[14] etc. can be viewed as special cases of this algorithm. We shall briefly discuss the steps of the algorithm.

Step 1

We suppose that there is a column criterion determining $\mathbf{w}_a$. The criterion states how we look at the columns of $\mathbf{X}$. If we use criterion (23), we are selecting principal variables. If we are doing PLS regression (PLS2), we select $\mathbf{w}_i$ according to (8). If $\mathbf{Y}=\mathbf{X}$, we are doing a PCA type of analysis, and determining $\mathbf{w}_a$ according to (7), we are doing PCA. Many criteria are possible which can reflect theoretical or applied views on data. If we do not specify a column criterion, we choose $\mathbf{w}_a = \mathbf{p}_a/|\mathbf{p}_a|$.

Step 2

We suppose that there is a criterion determining $\mathbf{v}_a$, which shows how rows should be looked at. If we only have an X-matrix, $\mathbf{Y} = \mathbf{X}$, and choose $\mathbf{v}_a$ as the eigenvector of $\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{v}_a = \lambda_a\mathbf{v}_a$, (e.g. by the NIPALS algorithm), we are doing PCA. Often $\mathbf{v}_a$ will reflect the weight that is associated with the objects in the analysis. If we do not specify such a row criterion, we choose $\mathbf{v}_a = \mathbf{t}_a/|\mathbf{t}_a|$.

If neither a column nor a row criterion is specified by the user, the algorithm selects $\mathbf{w}_a$ according to the PLS2 criterion (8) and chooses $\mathbf{v}_a$ as $\mathbf{v}_a = \mathbf{t}_a/|\mathbf{t}_a|$.

Step 3

The vectors $\mathbf{t}_a$ and $\mathbf{p}_a$ are the resulting vectors

$$\mathbf{t}_a = \mathbf{X}_{a-1}\mathbf{w}_a = \mathbf{x}_1 w_{1a} + \mathbf{x}_2 w_{2a} + \cdots$$

and

$$\mathbf{p}_a = \mathbf{X}_{a-1}^{\mathrm{T}}\mathbf{v}_a = \mathbf{x}^1 v_{1a} + \mathbf{x}^2 v_{2a} + \cdots$$

These vectors are the best we can select from X and the value of the criterion is $\mathbf{v}_a^{\mathrm{T}}\mathbf{X}_{a-1}\mathbf{w}_a$. We have a similar interpretation of $(\mathbf{v}_a^{\mathrm{T}}\mathbf{X}_{a-1}\mathbf{w}_a)^2$ as of σ_a^2, the variance, in the case of PCA, i.e. the variation extracted from X. The larger the numerical value of this criterion is, the better are the vectors $\mathbf{t}_a$ and $\mathbf{p}_a$. If the value is zero (or close to zero), we cannot reduce X further by using the given row and column criteria.

Step 4

We reduce X by the vectors selected. That this always gives a rank-one reduction of X when $\mathbf{v}_a^{\mathrm{T}}\mathbf{X}_{a-1}\mathbf{w}_a$ is different from zero is a well-known theorem of Laplace. If all possible vectors are selected, we get a decomposition of X that looks like

$$\mathbf{X} = \lambda_1 \mathbf{t}_1 \mathbf{p}_1^{\mathrm{T}} + \lambda_2 \mathbf{t}_2 \mathbf{p}_2^{\mathrm{T}} + \cdots + \lambda_K \mathbf{t}_K \mathbf{p}_K^{\mathrm{T}}$$

Step 5

The transformation vectors $\mathbf{r}_a$ and $\mathbf{s}_a$ are needed in order to compute the generalized inverse associated with the decomposition. $\mathbf{r}_a$ is determined so that $\mathbf{t}_a = \mathbf{X}\mathbf{r}_a = \mathbf{X}\mathbf{E}_{a-1}\mathbf{w}_a$ and $\mathbf{r}_a$ is a linear combination of $\mathbf{w}_1, \ldots, \mathbf{w}_a$. Writing out the expression for $\mathbf{E}_{a-1}$, it is easy to show the recursive equation for $\mathbf{E}_a$ in step 5. Similarly, $\mathbf{s}_a$ is determined so that $\mathbf{p}_a = \mathbf{X}^{\mathrm{T}}\mathbf{s}_a = \mathbf{X}^{\mathrm{T}}\mathbf{F}_{a-1}\mathbf{v}_a$ and $\mathbf{s}_a$ is a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_a$.

Step 6

Here we compute the contribution of the selected component (λ_a, $\mathbf{t}_a$, $\mathbf{p}_a$) in the explanation of the response variables.

Step 7

$\mathbf{B}_a$ represents the estimated regression coefficients. The matrix of estimated response values is $\hat{\mathbf{Y}}_a = \mathbf{X}\mathbf{B}_a$. If all components are selected, we get the linear least squares coefficients as $\mathbf{B}_K$, $\mathbf{B} = \mathbf{R}\Lambda\mathbf{Q}^{\mathrm{T}}$.

Step 8

Here we use some criterion to determine whether more components should be selected.

Geometric aspects of the vectors

The algorithm works with three pairs of vectors. ($\mathbf{w}_a$) and ($\mathbf{v}_a$) determine how we look at the *X*-matrix. The algorithm assumes that we have both a column and a row criterion. Frequently we

will only have one of them. We choose the other one as indicated in the algorithm. For these vectors we have

$$\mathbf{X}_b\mathbf{w}_a = \mathbf{0} \quad \text{and} \quad \mathbf{v}_a^T\mathbf{X}_b = \mathbf{0} \quad \text{for} \quad b \geqslant a$$

This means that we select the best we can at each step and orthogonalize $\mathbf{X}$ with respect to what has been selected. This property is the most important numerical aspect of the H-principle. We choose some vectors $(\mathbf{w}_a)$ and $(\mathbf{v}_a)$, that are good according to the views we have on columns and rows. We then orthogonalize $\mathbf{X}$ with respect to these choices, so that later X-matrices will be orthogonal to what have been chosen previously as ws and vs.

The vectors $(\mathbf{t}_a)$ and $(\mathbf{p}_a)$ reduce $\mathbf{X}$ as shown.

The vectors $(\mathbf{r}_a)$ and $(\mathbf{s}_a)$ reduce $\mathbf{X}^+$ as shown in the following equation:

$$\mathbf{X}^+ = \lambda_1\mathbf{r}_1\mathbf{s}_1^T + \lambda_2\mathbf{r}_2\mathbf{s}_2^T + \cdots + \lambda_K\mathbf{r}_K\mathbf{s}_K^T$$

Thus the algorithm can be viewed as a simultaneous expansion of $\mathbf{X}$ and the associated generalized inverse $\mathbf{X}^+$. This inverse satisfies $\mathbf{X}\mathbf{X}^+\mathbf{X} = \mathbf{X}$. This follows from the following geometric properties of the pairs $(\mathbf{p}_a)$, $(\mathbf{r}_a)$ and $(\mathbf{t}_a)$, $(\mathbf{s}_a)$:

$$\mathbf{s}_a^T\mathbf{t}_b = 0 \quad \text{and} \quad \mathbf{s}_a^T\mathbf{t}_a = 1/\lambda_a \quad \text{for } a \neq b$$

and

$$\mathbf{r}_a^T\mathbf{p}_b = 0 \quad \text{and} \quad \mathbf{r}_a^T\mathbf{p}_a = 1/\lambda_a \quad \text{for } a \neq b$$

In terms of matrices these properties can be written as

$$\mathbf{R}^T\mathbf{P} = \mathbf{\Lambda}^{-1} \quad \text{and} \quad \mathbf{S}^T\mathbf{T} = \mathbf{\Lambda}^{-1}$$

where $\mathbf{R} = (\mathbf{r}_i)$, $\mathbf{P} = (\mathbf{p}_i)$, $\mathbf{S} = (\mathbf{s}_i)$, $\mathbf{T} = (\mathbf{t}_i)$ and $\mathbf{\Lambda}^{-1} = \text{diag}(1/\lambda_i)$. These geometric properties of the three pairs of vectors are given in Reference 15.

In summary, the algorithm works with three pairs of vectors. $(\mathbf{w}_a)$ and $(\mathbf{v}_a)$ are determined from the criteria given by the practical problem in question. $(\mathbf{t}_a)$ and $(\mathbf{p}_a)$ reduce $\mathbf{X}$ by rank one and $(\mathbf{r}_a)$ and $(\mathbf{s}_a)$ reduce $\mathbf{X}^+$ by rank one. When $\mathbf{w}_a$ or $\mathbf{v}_a$ are selected, they can be scaled to be of length one. In the algorithm we do not scale $\mathbf{t}_a$ or $\mathbf{p}_a$. The scaling is taken care of by the constant λ_a. This is done in order to secure numerical stability of the algorithm. All computations in the algorithm are products and inner products except for the computation of λ_a. This means that the algorithm can be programmed to give very precise numerical results. The algorithm has been successfully applied to matrices involving thousands of rows and columns. In these cases it is important to use an appropriate scaling as in the algorithm.

APPENDIX II: SENSITIVITY MEASURES

We shall briefly review here some formulae that are useful for evaluating models and the given data. These formulae and their computational aspects are studied in details in Reference 9. We use the following notation.

$$\mathbf{X}_{(A)} = \lambda_1\mathbf{t}_1\mathbf{p}_1^T + \cdots + \lambda_A\mathbf{t}_A\mathbf{p}_A^T \qquad \text{Approximation of } \mathbf{X}.$$
$$\mathbf{X}_{(A)}^+ = \lambda_1\mathbf{r}_1\mathbf{s}_1^T + \cdots + \lambda_A\mathbf{r}_A\mathbf{s}_A^T \qquad \text{Approximation of } \mathbf{X}^+.$$
$$\hat{\mathbf{Y}}_{(A)} = \lambda_1\mathbf{t}_1\mathbf{q}_1^T + \cdots + \lambda_A\mathbf{t}_A\mathbf{q}_A^T \qquad \text{Approximation of } \mathbf{Y}.$$

Measures of fit

Variation	$d = \lambda^2(\mathbf{t}^T\mathbf{t})(\mathbf{q}^T\mathbf{q})$	Marginal contribution to the variation in $\mathbf{Y}$.
Dif	$d - s^2N/(N-A)$	s^2 is the residual variance. Negative difference indicates that the component is redundant.
R^2	$1 - \lvert\mathbf{Y} - \hat{\mathbf{Y}}\rvert^2/\lvert\mathbf{Y}\rvert^2$	$\hat{\mathbf{Y}}$ is the y-value estimated on the basis of the present model.
Adj R^2	$1 - (1-R^2)N/(N-A)$	A is the number of components in the model and N is the number of samples.
C_p	$\lvert\mathbf{Y} - \hat{\mathbf{Y}}\rvert^2/s^2 + 2A - N$	Mallows' C_p-value based on the given model. s^2 is the residual variance based on the full model.
Cov^2	$100\lambda^2\lvert\mathbf{p}\rvert^2\lvert\mathbf{Y}^T\mathbf{t}\rvert^2/\lvert\mathbf{X}^T\mathbf{Y}\rvert^2$	The percentage amount of covariance of the component.

Influence of observations (samples)

e_i	$y_i - \hat{y}_i$	Residual of response variable for the ith sample.
h_i	$\lvert\mathbf{X}^{+T}\mathbf{x}^i\rvert^2$	The leverage value associated with the ith sample.
h_{yi}		The leverage value when $\mathbf{X}$ and $\mathbf{Y}$ are combined into one matrix.
r_i	$e_i/[s_{(i)}(1-h_i)^{1/2}]$	The standardized value of the residual, here the residual standard deviation $s_{(i)}$ is computed without the ith sample. Called the externally studentized residual.
WK_i	$\lvert\hat{y}_i - \hat{y}_{(i)}\rvert/(s_{(i)}h_i^{1/2})$	The ith sample is also excluded here. Called the Welsh–Kuh distance.
Cov_i^2	$100(\lvert\mathbf{X}^T\mathbf{Y}\rvert^2 - \lvert\mathbf{X}_{(i)}^T\mathbf{Y}_{(i)}\rvert^2)/\lvert\mathbf{X}^T\mathbf{Y}\rvert^2$	$\mathbf{X}_{(i)}$ and $\mathbf{Y}_{(i)}$ are data with the ith sample deleted.

Sizes of figures

H-precision	$(\mathbf{v}^T\mathbf{X}\mathbf{w})^2$	The precision associated with the extracted component.
Size	$\lambda^2(\mathbf{t}^T\mathbf{t})(\mathbf{p}^T\mathbf{p})$	The size of the component extracted from $\mathbf{X}$.
%X	$100\text{tr}(\mathbf{X}_{(A)}^T\mathbf{X}_{(A)})/\text{tr}(\mathbf{X}^T\mathbf{X})$	The cumulative sum of the previous number. The result is given as a percentage of the total sum, which is $\text{tr}(\mathbf{X}^T\mathbf{X})$.
1/Size	$\lambda^2(\mathbf{r}^T\mathbf{r})(\mathbf{s}^T\mathbf{s})$	The size of components of the inverse.
$\%\mathbf{X}^+$	$100\text{tr}(\mathbf{X}_{(A)}^{+T}\mathbf{X}_{(A)}^+)/\text{tr}(\mathbf{X}^{+T}\mathbf{X}^+)$	Similarly, the cumulative sum.

Prediction error variance

H-error	$\lvert\mathbf{Y} - \hat{\mathbf{Y}}_A\rvert^2(1 + \sum^A \lambda_i^2)/(N-A)$	Corresponds to $\mathbf{x}_0$ such that $(\mathbf{r}_i^T\mathbf{x}_0)^2(\mathbf{s}_i^T\mathbf{s}_i) = 1$.

Prediction error variance

$$E\text{-}[(y(x_0)-\hat{y}(x_0))\,(y(x_0)-\hat{y}(x_0))^T] = S(1+\Sigma^A \lambda_i^2(r_i^T x_0)^2) + (\Sigma_{A+1}\,\lambda_i(r_i^T x_0)q_i)\,(\Sigma_{A+1}\,\lambda_i(r_i^T x_0)q_i)^T$$

Squared bias $|\Sigma_{A+1}\,\lambda_i(r_i^T x_0)q_i|^2$ Size of what is left of the regression equation.

APPENDIX III

We shall give here a MATLAB algorithm that carries out the basic algorithm of the H-principle. The algorithm supposes that there are given files X.MAT and Y.MAT that contain the X-matrix and Y-matrix of the data. It also supposes that there is given a file crit.mat that gives the numbers of the functions or procedures which are to be used as column and row criteria in the algorithm. The algorithm also supposes that the user knows the number of components to be selected and the numerical precision to be used. If the user has no knowledge about them, he/she can use nComp = number of columns or rows in X, whichever is smaller, and Eps = 0·00000001. If the values in different columns of X are of different magnitudes, the columns of X should be scaled to be of unit variance. This has been found to be adequate in practice. If the matrices X and/or Y are large, this scaling can be recommended. The algorithm has been applied in several cases where the matrices have had hundreds or thousands of columns and rows and the numerical precision of the algorithm has been found to function well provided that the matrices are appropriately scaled from the beginning.

```
% HDecomp carries out a decomposition of data according to the
% H-principle
% If Y=X, we get PCA types of decomposition
% If Y=matrix of response data, we get multivariate regression
% If some columns of Y are columns of X too, we get mixture
% models
% Variables
%   X         the original X-matrix
%   Y         the original Y-matrix
%   Crit      The column and row criteria to be used
%   Eps       Limit used in determining components
%   nComp     Maximum number of components to select
%   Res       Results from each step of decomposition
%   Ns        Number of components selected
%   B         The resulting regression coefficients
%   W, V, P, T, L, R, S : Matrices in the decomposition
% algorithm
%
% ------------------------------------------------------------
function [Res, Ns, B, W, V, P, T, L, R, S]=HDecomp(X, Y, Crit,
Eps, nComp);
%
[N,K]=size(X);              % Size of original X-matrix
[N1,M]=size(Y);             % Size of original Y-matrix
if N~=N1
  error('X and Y must have equal number of rows');
end
```

```
% ------------------------------------------------------------
%    Initialization of vectors and temporary matrices
B=zeros(K,M);                       % Solution matrix
wi=zeros(K,1);                      % Vectors
vi=zeros(N,1);
ti=zeros(N,1);
qi=zeros(M,1);
pi=zeros(K,1);
ri=zeros(K,1);
si=zeros(N,1);
E=eye(K);                          % Transformation matrix, for ri
F=eye(N);                          % Transformation matrix, for si
% ------------------------------------------------------------
%    Initialization of matrices
Res=zeros(nComp, 5);               % Results from decomposition
W=zeros(K,nComp);                  % Column weight vectors
V=zeros(N,nComp);                  % Row weight vectors
T=zeros(N,nComp);                  % Score vectors
P=zeros(K,nComp);                  % Loading vectors
R=zeros(K,nComp);                  % Transf. vectors, loadings
S=zeros(N,nComp);                  % Transf. vectors, scores
L=zeros(K,nComp);                  % Scaling constants
% ------------------------------------------------------------
%    Initialization of variables
Crw=Crit(1,1);                     % Criterion # for columns
Crv=Crit(1,2);                     % Criterion # for rows
iend=0;
i=0;
Ns=0;                              % Counter # of components
% ------------------------------------------------------------
%
while iend==0                      % Continue to select
% ------------------------------------------------------------
%   Initialization of the selection of a component
  i=i+1;                           % Number of components
%
% ------------------------------------------------------------
%   Step 1: Determination of w, if there is a column criterion
  if Crw>=1                        % Determine w
    [wi, wMax, wIn] = HDecompW(X, Y, Crw);
    ti=X*wi;                       % Compute score vector
  end
%
% ------------------------------------------------------------
%   Step 2: Determination of v, if there is a row criterion
  if Crv>=1                        % Determine v
    [vi, vMax, vIn] = HDecompV(X, Y, Crv);
    pi=X'*vi;                      % Compute loading vector
  end
%
% ------------------------------------------------------------
%   Step 1, 2 and 3: Compute score vector, loading vector and
% lambda
```

```
%
  if (Crw==0) & (Crv==0)
    error('Error: At least one criterion must be specified');
  elseif Crw==0
     a=norm(pi);                  % Length of p
     if a>Eps
       wi=pi/a;                   % Set w=p/|p|
       ti=X*wi;                   % t=Xw
       c=ti'*ti;
       wMax=c*c;                  % Value of column criterion
       wIn=0;                     % Index or information of w
     else
       iend=1;                    % We are finished
     end
  elseif Crv==0
     a=norm(ti);                  % Length of t
     if a>Eps
       vi=ti/a;                   % Set v=t/|t|
       pi=X'*vi;                  % p=X'v
       c=pi'*pi;
       vMax=c*c;                  % Value of row criterion
       vIn=0;                     % Index or information of v
     else
       iend=1;                    % We are finished
     end
  end
  lambda=pi'*wi;                  % lambda=v'Xw=p'w,
  lambda2=lambda*lambda;          % the H-precision
  if lambda2<Eps
     iend=1;                      % Finish. Not num. accuracy
  else
     lambda=1/lambda;             % Scaling constant
  end
% ------------------------------------------------------------
%    Collect results from the decomposition
  Res(i,1)=wMax;                  % The value of column crit
  Res(i,2)=wIn;                   % Register index
  Res(i,3)=vMax;                  % The value of row criterion
  Res(i,4)=vIn;                   % Register index
  Res(i,5)=lambda2;               % The H-precision
%
  if iend==0                      % Only if num accuracy is OK
% ------------------------------------------------------------
%    Step 4: Compute transformation vectors and update transf.
% matrices
     ri=E*wi;                     % Transf. vector, loadings
     si=F*vi;                     % Transf. vector, scores
     E=E-lambda*ri*pi';           % Update transf. matrices
     F=F-lambda*si*ti';           %
% ------------------------------------------------------------
```

```
%    Step 5: Adjust X for the selected component
%
    X=X-lambda*ti*pi';          % X reduced, SVD/PCA-type
% ------------------------------------------------------------
%    Step 6: The solution vector
%
    qi=Y'*si;                   % q=inner product Y with si
    B=B + lambda*ri*qi';        % The approximate solution
% ------------------------------------------------------------
%   Storage of vectors and matrices
    Ns=Ns+1;                    % # components selected
    W(:,i)=wi;
    V(:,i)=vi;
    T(:,i)=ti;
    P(:,i)=pi;
    R(:,i)=ri;
    S(:,i)=si;
    L(i,i)=lambda;
  end
  if (i==K) | (i==nComp)        % Stop decomposing X
    iend=1;
  end;
end
% ------------------------------------------------------------
% General properties of matrices involved for any row and
% column criteria
                              % If all components selected:
%X1=T*L*P'                    % Original X matrix
%X2=R*L*S'                    % Generalized inverse
%X3=X2*X1                     % Identity matrix
%X4=X1*X2*X1                  % Original matrix
%L1=inv(L)                    % Std deviations. Diagonal.
%X5=S'*T-L1                   % Zero matrix, geometric prop
%X6=R'*P-L1                   % Zero matrix,      ---
% Due to Xjwi=0 and Xj'vi=0 for j>i we have:
%X7=P'*W                      % Upper triangular. Diagonal
                              % elements same as L1.
%X8=T'*V                      % Upper triangular. Diagonal
                              % elements same as L1.
%Y1=Y-X1*B                    % Residual matrix. Zero if
                              % complete fit.
```

The function HDecompW.m tells us what criterion is to be used for columns and HDecompV.m what criterion is to be used for rows. Let us look more closely at HDecompW.m. If we are selecting principal variables, it could look like:

```
function [wi, wMax, wIn]=HDecompW(X,Y,Crw);
if Crw == 1
[wMax, wIn] = max(max(X'*Y*Y'*X));
wi=X(:,wIn);
```

If we are carrying out PCA/PLS, the function could look like

```
elseif Crw == 2
[B,D]=eig(X'*Y*Y'*X);
[B1,j]=max(D);
[wMax, wIn]=max(B1);
wi=B(:,wIn);
end
```

There are different situations, where the algorithm can be modified in order to be more efficient. For example, if $\mathbf{Y}$ is a column vector, $\mathbf{X}^T\mathbf{Y}$ has rank one and we can get the eigenvector without calling the function eig. Also, it may be faster to work with $\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}$. In the case of $\mathbf{Y}=\mathbf{X}$ we only need to call HDcompW once, since in this case $\mathbf{W}=\mathbf{B}$. Thus how we carry out the computations depends on the given situation.

The following MATLAB code carries out the estimation of the loading vector for quadratic components. We call this function in the procedure HDcompW.

```
[wi]=qpls(X, Y, T, nComp);
[N,K]=size(X);
[N1,M]=size(Y);
K1=nComp-1;
e1=ones(1,1);
H=Y*Y';                              % WEIGHT MATRIX
for i=1:K1
  R0=Y.*(T(:,i)*e1);                 % Weigh the y-matrix with
                                       % each t-vector
  H=H+R0*R0';                        % New weight matrix for
                                       % linear term
end
% ------------------------------------------------------------
%                              PLS2/PCA LINEAR
C=X'*H*X;
[E1,D]=EIG(C);
[B2,BO]=MAX(D);
[B3,J]=MAX(B2) ;
wi=E1(:,J);                          % THE W-WEIGHT VECTOR
G0=0;
IT=0;
IEND=0;
while IEND==0
% ------------------------------------------------------------
%                              QUADRATIC PLS/PCA
  F2=0;
  IT=IT+1;                           % NUMBER OF ITERATIONS
  w2=C*wi;                           % LINEAR TERM
  F1=wi'*w2;                         % SIZE OF LINEAR TERM
  for J=1:L                          % FOR EACH RESPONSE VARIABLE
    G=X'*DIAG(Y(:,J))*X;             % G MATRIX FOR EACH Y-COLUMN
    A=wi'*G*wi;                      % THE COEFFICIENT IN SQUARE
                                       % TERMS
```

```
    w2=w2 + 2*A*G*wi;          % ADD THE CONTRIBUTION FROM SQ
                                  % TERM
    F2=F2+A*A;                 % SIZE OF SQUARE TERM
  end                          % END OF TREATING THE Y-VECTORS
  F=F1+F2;                     % VALUE OF CRITERION,
                                  % LIN+QUADR.
  w2=w2/NORM(w2);              % SCALE NEW W TO LENGTH 1
  A1=NORM(wi-w2);              % SIZE OF DIFFERENCE OF W-VEC.
  if A1<0.001                  % SIZE MUST BE SMALL
    IEND=1;                    % WE ARE FINISHED IN DETERMING
                                  % A NEW COMP
  else
    wi=w2;                     %NEW CRITERIA VECTOR
  end
  if (IT>100) | (G0>F)         % MAX 100 ITERATIONS. CRITERION
                                  % MUST INCREASE
     w2=w3;                    % USE RESULTS FROM PREVIOUS
                                  % ITERATION
     F1=G1;
     F2=G2;
     IEND=1;
  end
  G0=F;                        % STORE RESULTS FROM LAST
                                  % ITERATION
  G1=F1;
  G2=F2;
  w3=w2;
end                            % END ITERATIONS
```

We do not show here any practical aspects of the algorithm for determining quadratic components, e.g. testing for significance of components, adjustment of the response vectors, revised computations and other relevant 'household' tasks.

REFERENCES

1. S. Jolliffe, S., *Principal Component Analysis*, Springer, Berlin (1968).
2. T. Næs and H. Martens, *Chemometrics Intell. Lab. Sys.*, **5**, 155–168 (1989).
3 T. Næs and H. Martens, *J. Chemometrics*, **2**, 155–167 (1988).
4. O. M. Kvalheim, *Anal. Chim. Acta*, **233**, 53–73 (1989).
5. J. E. Jackson, *Users Guide to Principal Components*, Wiley, New York (1991).
6. S. Wold, H. Martens and H. Wold, in *Matrix Pencils*, ed. by A. Ruhe and B. Kågström, pp. 286–293, Springer, Heidelberg (1983).
7. S. Wold, C. Albano, W. J. Dunn III, U. Edlund, K. Esbensen, P. Geladi, S. Helberg, E. Johansen, W. Lindberg and M. Sjöström, in *Chemometrics, Mathematics and Statistics in Chemistry*, ed. by B. R. Kowalski, Reidel, Dordrecht, p. 17 (1984).
8. A. Höskuldsson, *Chemometrics Intell. Lab. Syst.* **23**, 1–28, (1994).
9. S. Chatterjee, A. S. Hadi, *Sensitivity Analysis in Linear Regression*, Wiley, New York (1988).
10. A. Höskuldsson, *J. Chemometrics*, **2**, 211–228 (1988).
11. A. Höskuldsson, *Prediction Methods in Sciences*, Internal Report, Danish Engineering Academy (1994).

12. A. Höskuldsson, *J. Chemometrics*, **6**, 307–334 (1992).
13. G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD (1989).
14. M. Hestenes, *Conjugate Direction Methods in Optimization*, Springer, Berlin (1980).
15. A. Höskuldsson, *SIAM J. Sci. Comput.*, **15**, 239–262 (1994).