Sample specific prediction intervals in SIMPLS ⋆

Sven Serneels, Pierre J. Van Espen *

Department of Chemistry, University of Antwerp, Belgium

Abstract

In this article, we propose two new differentiated algorithms for SIMPLS. These algorithms allow efficient computation of the Jacobian matrix for the regression vector, taking into account error propagation of eiter X, $\mathbf{y}$ or both X and $\mathbf{y}$, the last case being better known as the errors-in-variables problem. The Jacobian matrices thus computed lead to a new way of calculating a sample-specific prediction error in SIMPLS regression.

Key words: partial least squares regression, Jacobian matrix, prediction error, errors-in-variables.

1 Introduction

In spectrometrical applications, partial least squares regression (PLS) has over the last decade become a standard tool for quantification. The simplicity of the technique makes it most appealing to include it in the commercial software of spectrometer benches, since a single (matrix) multiplication is sufficient to estimate a concentration from a measured spectrum once calibration has been completed.
Albeit estimation of sample specific prediction intervals in PLS in general (or SIMPLS in particular) has been a major topic of research throughout

⋆ Research financed by a PhD grant of the Institute for Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)
* Correspondence to P. Van Espen, Departement Scheikunde, Universiteit Antwerpen, Universiteitsplein 1, 2610 Antwerpen (Belgium)
Email address: piet.vanespen@ua.ac.be (Pierre J. Van Espen).
URL: http://chemometrix.uia.ac.be (Pierre J. Van Espen).

Please cite this publication as:
Sample specific prediction intervals in SIMPLS,
S. Serneels and P.J. Van Espen,
in: PLS and related methods,M. Vilares, M. Tenenhaus, P. Coelho,
V. Esposito Vinzi and A. Morineau (eds.), DECISIA, Levallois Perret (France),
2003, pp. 219-233.

the last decade, in practice the thereunto proposed methods are seldom applied in practice. The most common technique to estimate prediction errors in SIMPLS is the so-called *root mean squared error of prediction (RMSEP)* [1], which is estimated as follows: one uses the regression vector to estimate the concentrations of a set of samples of which the true concentration is known, but which have not been used for calibration. Usually, this set of samples is referred to as the *validation set*. Estimated and true concentrations are then used to compute the RMSEP, which is then supposed to be a measure of the uncertainty of *all* future predictions made by this model.

The RMSEP is an *average* measure of uncertainty. Methods which allow the estimation of a sample specific prediction error have been proposed [2–5]. In this article, we extend the method as proposed by Denham [3] to SIMPLS regression, as it was originally designed for PLS regression based on Helland's algorithm [6]. Because of its computational inefficience compared to the more courantly used SIMPLS algorithm [7], Helland's algorithm has never become popular. One might object to this statement that by means of using modern processors the inefficience of Helland's algorithm is of minor importance. On the other hand, mainly due to its inefficience, Helland's algorithm has never been included in commercial PLS packages. Hence, the methods proposed by Serneels *et al.* [5] and Denham [3] are not immediately applicable for anyone who is accustomed to the more or less "standard" SIMPLS algorithm.

Hitherto we have not explicitly heeded the fact that the calibration spectra may also be prone to a sizeable measurement error. Whenever the measurement error in the spectra is not negligible it may contribute significantly to the prediction error. For example, in X-ray fluorescence spectrometry it is well known that the measurement error for each channel equals the square root of the number of counts due to Poisson statistics. In other types of spectroscopy, the origin of measurement error is mostly unknown, but the error can often not be ignored. Methods in which the measurement error in the spectra is propagated are commonly known as *errors in variables* (EIV) techniques. Faber and Kowalski [4] have proposed expressions which allow to construct sample specific prediction intervals for SIMPLS when the measurement error in the spectra is nonnegligible. In this article, we propose a novel algorithm which enables efficient estimation of the Jacobian matrix thereunto required. The need for a sample specific prediction error for partial least squares regression which is easy to compute, which can be calculated by means of the vectors generated by the SIMPLS algorithm and which can take into account the errors in the spectra led us to the work proposed in this article. In Section 2 we introduce the reader to the notation used throughout this article.

In Section 3 we provide a short introduction to partial least squares regression.

In Section 4 we point out the differences between SIMPLS and the classical PLS algorithms.

In Section 5 we propose an algorithm which allows efficient calculation of the Jacobian matrix with respect to the response vector in SIMPLS regression. This leads to a sample specific prediction error in SIMPLS.

In Section 6 we propose an algorithm which allows efficient calculation of the Jacobian matrix with respect to the predictor matrix in SIMPLS regression. This leads to a sample specific prediction interval in SIMPLS in the errors in variables setting.
The last section concludes; in the Appendix the proofs of the two new algorithms are given.

2 Notation and definitions

Before we can give an introduction to partial least squares regression, we need first define the notation used. The calibration matrix X is a matrix of size $n \times p$ in which the rows are n spectra of standard samples, measured at p channels. Matrices will always be denoted in upper case letters. The corresponding n concentrations of these standard samples constitute the response vector $\mathbf{y}$. Vectors will always be denoted by bold-face lower case letters. When we refer to individual columns of matrices, we shall denote these vectors using the corresponding letter. Throughout this work, we will assume both calibration and respons matrices to be mean-centred. When calibration is completed, spectra of new samples (unknowns or validation set, if used) are denoted by means of the corresponding Greek letter, i.e. a matrix of new spectra is denoted as Ξ. The corresponding (mostly unknown) concentrations are consistently denoted as $\boldsymbol{\upsilon}$. A circumflex accent denotes an estimate, e.g. $\hat{\boldsymbol{\upsilon}}$ the estimated concentrations. Whenever it is necessary to make use of row vectors, they will be represented by lower-case underlined letters. Finally, $\otimes$ and ^{T} denote the Kronecker product and transposition, respectively.
The *vec-operator* is an operator which stacks the rows of a $n \times p$-matrix underneath each other to yield an $np \times 1$-vector. It will be denoted as $\mathrm{vec}(\cdot)$.

3 Partial least squares regression (PLS)

Partial least squares regression is a way to estimate the regression vector $\boldsymbol{\beta}$ in a linear model

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1}$$

In this equation $\boldsymbol{\varepsilon}$ is a constant vector of identically and independently distributed erros with zero expectation and unit variance.
Spectroscopic datamatrices typically consist of few observations compared to the number of spectroscopic variables. Moreover, these spectroscopic variables are often correlated. Hence, ordinary least squares estimates of $\boldsymbol{\beta}$ are often unstable and prone to a high variance in spectrometric applications.

Partial least squares regression is most widely used because it overcomes these problems by extracting a set of uncorrelated so-called *latent variables* from the original data. These variables are constructed according to the following objective function [8]:

$$\hat{\mathbf{a}}_h = \underset{\mathbf{a}}{\operatorname{argmax}} \operatorname{cov}(X\mathbf{a}_h, \mathbf{y}) \tag{2a}$$

under the constraints that

$$\| \mathbf{a}_h \| = 1 \quad \text{and} \quad \mathbf{a}_h^T X^T X \mathbf{a}_i = 0 \text{ for } 1 \leq i < h. \tag{2b}$$

This objective function is a maximization problem under two constraints which can be solved by means of the Langrange multiplier method. All PLS algorithms allow computation of latent variables satisfying the above objective function. However, different PLS algorithms accomplish computation of latent variables as well as the regression vector in a slightly different way.
We will shortly digress on the original PLS approach, as originally defined by Wold [9]. In any algorithm, the first weighting vector must be the dominant eigenvector of the matrix $X^T\mathbf{y}\mathbf{y}^T X$, which will then be or be not scaled, according to the convention imposed. From the second latent variable on, the second constraint becomes important: It requires the following latent variables to be orthogonal (uncorrelated) to the previous ones. Hence, the following weighting vectors will be dominant eigenvectors of the matrix $X^T\mathbf{y}\mathbf{y}^T X$, multiplied by a projection matrix which projects onto the orthogonal complement of the subspace spanned by the previous score vectors. Thus, before scaling, the hth weighting vector will in general be equal to:

$$\hat{\mathbf{a}}_h = X^T \left(I_n - \sum_{i=1}^{h-1} \frac{\mathbf{t}_i \mathbf{t}_i^T}{\mathbf{t}_i^T \mathbf{t}_i} \right) \mathbf{y} \tag{3}$$

The first two factors can be seen as a deflation of the datamatrix X and in this way all classical PLS algorithms are denoted: set $E_0 = X$ and compute the deflated datamatrix as follows:

$$E_h = \left(I_n - \sum_{i=1}^{h-1} \frac{\mathbf{t}_i \mathbf{t}_i^T}{\mathbf{t}_i^T \mathbf{t}_i} \right) X \tag{4}$$

As will be explained in the next Section, SIMPLS differs from classical PLS at this level: the defation will be carried out on the vector $\mathbf{s} = X^T\mathbf{y}$.

4 SIMPLS versus classical PLS

SIMPLS has been proposed in 1993 by Sijmen de Jong [7] as a *Statistically Inspired Modification* of the PLS algorithm. Most classical PLS algorithms

were developed to link the latent variables to the response vector, without former knowledge which criterion one was maximizing. In contrast thereto, de Jong started the deviation of the SIMPLS algorithm from the aforementioned objective (Equation 2).
This leads to a direct calculation of the weighting vectors, which are now called $\mathbf{r}$. Note that computation of the vectors $\mathbf{r}$ in SIMPLS is a way of estimating the weighting vectors $\mathbf{a}$ from the objective and thus need not be denoted with a hat. Algorithmically, one of the major differences between classical PLS and SIMPLS is the way the deflation is done. As stated in the previous section, from the second latent variable on, the weighting vectors are the dominant eigenvectors of $X^T\mathbf{y}\mathbf{y}^TX$, multiplied by a projection matrix. One can deflate the matrix X by means of the projection matrix (classical PLS, Section 3); an alternate choice is to deflate the matrix $X^T\mathbf{y}$. This is what is actually done in the SIMPLS algorithm. The weighting vectors in univariate SIMPLS, which is what we discuss here, are then simply identical to the deflated $X^T\mathbf{y}$-matrix, as can be seen from in algorithm, i.e. in Equations 5a and 5d.
In classical PLS, direct implementation is mostly computationally inefficient. Hence, computational upgrading has been sought for which has led to modifications which are computationally efficient, but which lack transparency. In SIMPLS, the algorithm, as published, implemented directly, (without programmatorical modification) performs as good (or even better) as the classical PLS. Hence, de Jong proposed as an alternative interpretation of the abbreviation "SIM" in SIMPLS: *Straightforward Implementation*.
We will now introduce the reader to a slightly modified SIMPLS algorithm, the proof of which is given in the Appendix.

$$\mathbf{a}_h = \begin{cases} X^T\mathbf{y} & \text{for } h = 1 \\ \left(I_p - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\right)\mathbf{a}_{h-1} & \text{for } h > 1 \end{cases} \tag{5a}$$

$$\mathbf{r}_h = \frac{\mathbf{a}_h}{\sqrt{\mathbf{a}_h^T S \mathbf{a}_h}} \tag{5b}$$

$$\mathbf{p}_h = S\mathbf{r}_h \tag{5c}$$

$$\mathbf{v}_h = \left(I_p - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\right)\mathbf{p}_h \tag{5d}$$

$$\mathbf{b}_h = \mathbf{b}_{h-1} + \mathbf{r}_h\mathbf{r}_h^T\mathbf{s} \tag{5e}$$

In this algorithm, $S = X^TX$ and $\mathbf{s} = X^T\mathbf{y}$, respectively. To initiate the algorithm, set $\mathbf{b}_0 = \mathbf{0}_p$.

5 Sample specific prediction intervals in SIMPLS

5.1 Introduction

Various methods to characterize individual measures by their uncertainty have been reported. On one hand, data driven methods such as the bootstrap can be applied, but as the matrix dimensions increase, computation times rise drastically.
Another approach consists of extending the existing statistical methodology (for ordinary least squares) to PLS. This has been initiated by Agnar Höskuldsson [10] who assumed the PLS estimator to be a approximately linear in $\mathbf{y}$. This assumption is an approximation of zeroth order to the PLS estimator. A more exact way to proceed is to approximate the PLS estimator in the first order. Approximations of the first order, allowing to provide realistic sample specific measures of uncertainty, have been proposed successively by Phatak *et al.* [2], Denham [3], Faber and Kowalski [4] and recently by Serneels *et al.* [5]. The pros and cons of each of those methods have been explained in the Introduction. The linear approximation of the first order consists of a series expansion for the PLS regression vector of which only the first and second terms are retained. Simulation studies by Faber [11] corroborate the assumption that an approximation of the first order is viable.

5.2 Algorithm for $\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial \mathbf{y}}\right)$

In general, the Jacobian matrix for the regression vector in PLS can be obtained in two different ways:

(1) Find a closed-form expression for the regression vector and differentiate it with respect to the variable of interest.
(2) Find an algorithm for the regression vector. Differentiate each step of the algorithm with respect to the variable of interest.

The former appoach was used by Phatak *et al.* [2] for $\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial \mathbf{y}}\right)$. It has been extended to EIV by Faber and Kowalski [4] to yield an expression which may be computationally cumbersome as matrix dimensions n and p increase. The latter appraoch was initiated by Denham [3] and later successfully applied by us in [5] and in the previous Section. Here it will be applied to yield an elegant and computationally efficient algorithm for both Jacobians.
For the sake of simplicity, we will omit in our notation the point about which is being linearized and will denote the Jacobians as, e.g. $\frac{\partial \mathbf{r}_1}{\partial \mathbf{y}}$ instead of $\left(\frac{\partial \mathbf{r}_1}{\partial \mathbf{y}}\right)_{\mathbf{y}_0}$.
The algorithm goes as follows: for h=1: set $\frac{\partial \mathbf{a}_1}{\partial \mathbf{y}} = X^T$ and start the algo-

rithm at Equation 6b. For successive latent variables, follow all steps of the algorithm.

$$\frac{\partial \mathbf{a}_h}{\partial \mathbf{y}} = \frac{\partial \mathbf{a}_{h-1}}{\partial \mathbf{y}} - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\frac{\partial \mathbf{a}_{h-1}}{\partial \mathbf{y}} - \frac{\left(\mathbf{a}_{h-1}^T\mathbf{v}_{h-1}I_p + \mathbf{a}_{h-1}^T \otimes \mathbf{v}_{h-1}\right)\frac{\partial \mathbf{v}_{h-1}}{\partial \mathbf{y}}}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}} + 2\frac{\mathbf{a}_{h-1}^T\mathbf{v}_{h-1}\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T\frac{\partial \mathbf{v}_{h-1}}{\partial \mathbf{y}}}{\left(\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}\right)^2} \tag{6a}$$

$$\frac{\partial \mathbf{r}_h}{\partial \mathbf{y}} = \frac{\frac{\partial \mathbf{a}_h}{\partial \mathbf{y}}}{\sqrt{\mathbf{a}_h^T S \mathbf{a}_h}} - \frac{\mathbf{a}_h\mathbf{a}_h^T X^T X\left(\frac{\partial \mathbf{a}_h}{\partial \mathbf{y}}\right)}{\left(\mathbf{a}_h^T S \mathbf{a}_h\right)^{\frac{3}{2}}} \tag{6b}$$

$$\frac{\partial \mathbf{p}_h}{\partial \mathbf{y}} = S\frac{\partial \mathbf{r}_h}{\partial \mathbf{y}} \tag{6c}$$

$$\frac{\partial \mathbf{v}_h}{\partial \mathbf{y}} = \frac{\partial \mathbf{p}_h}{\partial \mathbf{y}} - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\frac{\partial \mathbf{p}_h}{\partial \mathbf{y}} - \left[\frac{\left(\mathbf{p}_h^T\mathbf{v}_{h-1}I_p + \mathbf{p}_h^T \otimes \mathbf{v}_{h-1}\right)\frac{\partial \mathbf{v}_{h-1}}{\partial \mathbf{y}}}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}} - 2\frac{\mathbf{p}_h^T\mathbf{v}_{h-1}\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T\frac{\partial \mathbf{v}_{h-1}}{\partial \mathbf{y}}}{\left(\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}\right)^2}\right] \tag{6d}$$

$$\frac{\partial \mathbf{b}_h}{\partial \mathbf{y}} = \frac{\partial \mathbf{b}_{h-1}}{\partial \mathbf{y}} + \left(\mathbf{r}_h \otimes \mathbf{s}^T + \mathbf{r}_h^T\mathbf{s}I_p\right)\frac{\partial \mathbf{r}_h}{\partial \mathbf{y}} + \mathbf{r}_h\mathbf{r}_h^T X^T \tag{6e}$$

It is easily seen that this is an exactly differentiated SIMPLS algorithm (of course $\mathbf{v}_0$, $\frac{\partial \mathbf{v}_0}{\partial \mathbf{y}}$ $\frac{\partial \mathbf{b}_0}{\partial \mathbf{y}}$ are zero matrices of appropriate dimensions). It is established using the approach of matrix differential calculus [14]. An elaborate proof of the algorithm can be found in the Appendix. The algorithm needs as inputs the standard weighting, loading and basis vectors from SIMPLS, as well as the non-scaled weighting vectors. Whilst implemeting one should thus see that in the SIMPLS routine the non-scaled weighting vectors ($\mathbf{a}_h$) are stored separately from the scaled weighting vactors ($\mathbf{r}_h$).

5.3 Comparison with Denham's algorithm

We will shortly compare the newly proposed algorithm with the existing technique in terms of computational efficiency. Calculations were carried out in the MATLAB environment (The MathWorks, Natick, MA, USA) on a datamatrix consisting of 24 observations and 6 variables.

[Figure 1 about here]

In figure **??** the flops required in both algorithms are plotted. We can see that for any number of latent variables, the newly proposed algorithm outperforms the existing technique in terms of flop counts. In both cases, a the number of flops required increases linearly with respect to the number of latent variables (in our algorithm, the computation of the Jacobian for a single latent variable is less complicated than successive Jacobians, hence the linearity from the second latent variable on). Moreover, the rate of increase in flops is significantly higher in Denham's algorithm.

5.4 Prediction intervals

An estimate for the degrees of freedom, based on a local linearization of the first order, has been given by Denham [3]:

$$\mathrm{df} = \mathrm{tr}\left[\left(I_n - X\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)^T\left(I_n - X\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)\right] \tag{7}$$

In Section 2 we introduced the oft practiced convention that the individual calibration spectra are the rows of the calibration matrix. To be consistent with this notation, new spectra of unknown samples will also be denoted as row vectors ($\underline{\xi}$). They are proposed to be centred about the mean of X. Whenever an unknown sample is to be quantified, an estimate of the variance of the predicted analyte concentration is given by:

$$\mathrm{var}(\hat{\boldsymbol{\upsilon}}) = \sigma^2\left(\frac{n+1}{n} + \underline{\xi}\left(\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)\left(\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)^T\underline{\xi}^T\right) \tag{8}$$

Note that this equation is only dependent on the Jacobian $\left(\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)$; in the errors in variables setting this will no longer be the case. Equation 8 leads to the construction of a prediction interval specific to this sample:

$$\hat{\upsilon} \pm t_{\alpha/2,\mathrm{df}}\hat{\sigma}\left[\frac{n+1}{n} + \underline{\xi}\left(\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)\left(\frac{\partial\hat{\mathbf{b}}_h}{\partial\mathbf{y}}\right)^T\underline{\xi}^T\right]^{\frac{1}{2}} \tag{9}$$

An expression for $\hat{\sigma}$ has been given by Denham [3]. Note that Equation 9 still has to be corrected for mean-centring.

6 Sample specific prediction intervals for SIMPLS in the EIV model

6.1 Errors-in-variables

As stated in the Introduction, the EIV principle consists of the propagation of measurement error in the spectra. This means that the datamatrix X consists of two terms:$X = \check{X} + E$ where $\check{X}$ is the errorless datamatrix and E denotes a $n \times p$-matrix containing measurement errors. In most cases $\check{X}$ and E are unknown, so all predictions will be based on X. The linear model (Equation 1) changes to:

$$\mathbf{y} = (X - E)\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{10}$$

This change does not affect the estimation of the regression vector: one estimates the regression vector by SIMPLS *knowing that the datamatrix from which it has been computed contains error*. However, its variance will be influenced by the error in the spectra. Propagation of the measurement error of the spectra up to the regression vector has been done by Faber and Kowalski [4]. The variance of the regression vector in the EIV context is given by:

$$\operatorname{var}\hat{\mathbf{b}_h} = \sigma_{\mathbf{y}}^2 \left(\frac{\partial \hat{\mathbf{b}}_h}{\partial \mathbf{y}}\right)\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial \mathbf{y}}\right)^T + \sigma_X^2 \left(\frac{\partial \hat{\mathbf{b}}_h}{\partial X}\right)\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial X}\right)^T \tag{11}$$

Computation of this variance requires computation of the Jacobian with respect to the spectra.

6.2 Algorithm for $\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial X}\right)$

The algorithm goes as follows: for the first latent variable, set:

$$\frac{\partial \mathbf{a}_1}{\partial X} = (I_p \otimes \mathbf{y}^T) \tag{12}$$

Thenceforth, follow each step of the algorithm as stated in Equations 13. From the second latent variable on, follow the whole algorithm, i.e.:

$$\begin{aligned}\frac{\partial \mathbf{a}_h}{\partial X} = &\frac{\partial \mathbf{a}_{h-1}}{\partial X} - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\frac{\partial \mathbf{a}_{h-1}}{\partial X} - \frac{\left(\mathbf{a}_{h-1}^T\mathbf{v}_{h-1}I_p + \mathbf{a}_{h-1}^T \otimes \mathbf{v}_{h-1}\right)\frac{\partial \mathbf{v}_{h-1}}{\partial X}}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}} \\ &+2\frac{\mathbf{a}_{h-1}^T\mathbf{v}_{h-1}\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T\frac{\partial \mathbf{v}_{h-1}}{\partial X}}{\left(\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}\right)^2}\end{aligned} \tag{13a}$$

$$\frac{\partial \mathbf{r}_h}{\partial X} = \frac{\frac{\partial \mathbf{a}_h}{\partial X}}{\sqrt{\mathbf{a}_h^T S \mathbf{a}_h}} - \frac{\mathbf{a}_h \mathbf{a}_h^T X^T \left(X \frac{\partial \mathbf{a}_h}{\partial X} + \mathbf{a}_h^T \otimes I_n \right)}{(\mathbf{a}_h^T S \mathbf{a}_h)^{\frac{3}{2}}} \tag{13b}$$

$$\frac{\partial \mathbf{p}_h}{\partial X} = \mathbf{r}_h^T \otimes X^T + I_p \otimes \left(\mathbf{r}_h^T X^T \right) + S \frac{\partial \mathbf{r}_h}{\partial X} \tag{13c}$$

$$\begin{aligned} \frac{\partial \mathbf{v}_h}{\partial X} = & \frac{\partial \mathbf{p}_h}{\partial X} - \frac{\mathbf{v}_{h-1} \mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T \mathbf{v}_{h-1}} \frac{\partial \mathbf{p}_h}{\partial X} - \left[\frac{\left(\mathbf{p}_h^T \mathbf{v}_{h-1} I_p + \mathbf{p}_h^T \otimes \mathbf{v}_{h-1} \right) \frac{\partial \mathbf{v}_{h-1}}{\partial X}}{\mathbf{v}_{h-1}^T \mathbf{v}_{h-1}} \right. \\ & \left. - 2 \frac{\mathbf{p}_h^T \mathbf{v}_{h-1} \mathbf{v}_{h-1} \mathbf{v}_{h-1}^T \frac{\partial \mathbf{v}_{h-1}}{\partial X}}{\left(\mathbf{v}_{h-1}^T \mathbf{v}_{h-1} \right)^2} \right] \end{aligned} \tag{13d}$$

$$\frac{\partial \mathbf{b}_h}{\partial X} = \frac{\partial \mathbf{b}_{h-1}}{\partial X} + \left(\mathbf{r}_h \otimes \mathbf{s}^T + \mathbf{r}_h^T \mathbf{s} I_p \right) \frac{\partial \mathbf{r}_h}{\partial X} + \mathbf{r}_h \mathbf{r}_h^T (I_p \otimes \mathbf{y}^T) \tag{13e}$$

As a starting values, set $\mathbf{v}_0 = \mathbf{0}_{p \times 1}, \frac{\partial \mathbf{v}_0}{\partial X} = \frac{\partial \mathbf{b}_0}{\partial X} = \mathbf{0}_{p \times np}$. Note that the algorithm for $\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial \mathbf{y}} \right)$ (Equations 6) and the algorithm proposed here are quite similar. Several terms only differ slightly from the correspondent ones in Equations 6. Hence, these algorithms may be implemented together in a single program, which enhances their computational performance.

6.3 Comparison to the existing method

Hitherto, the only formula to compute $\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial X} \right)$ has been given by Faber and Kowalski [4], who supply a closed-form expression for the Jacobian. However, their formula may give rise to numerical instabilities as it requires the Krylov matrix $\left(\mathbf{s} \; S\mathbf{s} \; S^2\mathbf{s} \; \cdots \; S^h\mathbf{s} \right)$ to be computed explicitly.
As a differentiated Helland algorithm with respect to X is up to our knowledge heretofore unreported, we are not able to provide a comparison in flop counts as in the previous section.

7 Conclusions

In practical applications, PLS regression is widely used to compute concentrations from spectra. It is often a part of the software delivered by the manufacturer of the spectrometer bench. The uncenrtainty of the obtained predictions is hardly assessed. In most commercial packages, the RMSEP is seen as the measure for the uncertainty of the predictions. As it is an average measure, sample specific prediction intervals are seldom produced.

The idea of a sample specific prediction interval has raised interest ever since it was introduced by Höskuldsson [10]. For example, Höskuldsson linear approximation of the zeroth order has been adopted by the American Society for Testing and Materials (ASTM) [12]. It has been explained in Section 4 that this method ignores the non-linearity in the PLS estimator. Faber *et al.* [13] argue that application of this method is the result of a trade-off between "statistical rigor" and "user friendlyness".
Thanks to all previous work on this topic as well as to the algorithm proposed here, such a trade-off is no longer necessary. The linear approximation of the first order, which is the most rigourous statistical result yet obtained, can now be computed in reasonable time and can be combined with SIMPLS regression and EIV. Hence, the drawbacks the linear approximation had compared to other methods, are no longer existent. We hope that sample specific prediction intervals for PLS may in the nearby future be included in software packages.

References

[1] Massart, D.L., Vandeginste, B.G.M., Buydens, L.M.C., de Jong, S., Lewi, P.J. and Smeyers-Verbeke, J, 1997. Handbook of Chemometrics and Qualimetrics: Part A. Elsevier, Amsterdam, 282.

[2] Phatak, A., Reilly, P.M. and Penlidis A., 1993. An approach to interval estimation in partial least squares regression. Analytica Chimica Acta, 277 495-501.

[3] Denham, M.C., 1997. Prediction intervals in partial least squares. J. Chemometrics, 11 39-52.

[4] Faber, N.M. and Kowalski, B.R., 1997. Propagation of measurement errors for the validation of predictions obtained by principal component regression and partial least squares. J. Chemometrics, 11 181-238.

[5] Serneels, S., Lemberge, P. and Van Espen, P.J. Calculation of PLS prediction intervals using efficient recursive relations for the Jacobian matrix. submitted to *J. Chemometrics*.

[6] Helland I.S., 1988. On the structure of partial least squares regression. Commun. Stat. – Simul. Comput., 17 581-607.

[7] de Jong, S., 1993. SIMPLS: an alternative approach to partial least squares regression. Chemometr. Intell. Lab. Syst., 42 251-263.

[8] Ter Braak, C.J.F. and de Jong, S., 1998. The objective function of partial least squares regression. J. Chemometrics, 12 41-54.

[9] Wold, H., 1982. Soft modeling: the basic design and some extensions. In: Jöreskog, K.G. and Wold, H. (eds.), Systems under indirect observation, part II, North-Holland, Amsterdam, 1-54.

[10] Höskuldsson, A., 1988. PLS regression methods. J. Chemometrics, 2 211-228.

[11] Faber, N.M., 2002. Uncertainty estimation for multivariate regression coefficients. Chemometr. Intell. Lab. Syst, 64 169-179.

[12] Annual Book of ASTM Standards, Vol. 03.06, E1655, Standard Practices for Infrared, Multivariate, Quantitative Analysis. ASTM International: West Conshohocken, Pennsylvania, USA, 1998.

[13] Faber, N.M., Song, X.-H. and Hopke, P.K. Sample-specific standard error of prediction for partial least squares regression. Trends in Analytical Chemistry, in press.

[14] Magnus, J.R. and Neudecker, H., 1998. Matrix Differential Calculus with Applications in Statistics and Econometrics. Wiley, Chichester.

A Proof of Equations 5

The first difference between our "variant" algorithm and the original SIMPLS algorithm is the by-pass of the vectors $\mathbf{t}_h$. As in the original algorithm the $\mathbf{t}_h$ are computed $\mathbf{t}_h = X\mathbf{r}_h$, whereafter both $\mathbf{t}_h$ and $\mathbf{r}_h$ are divided by the norm of the $\mathbf{t}_h$, it is clear that it is equivalent to write Equation 5b instead. Moreover $\mathbf{p}_h = X^T\mathbf{t}_h$ and thus equals $S\mathbf{r}_h$ as in Equation 5c.
To conclude the proof, the construction of the basis vectors can be re-written as:

$$\begin{aligned}\mathbf{v}_h &= \mathbf{p}_h - V_{h-1}V_{h-1}^T\mathbf{p}_h \\ &= \left(I_p - \sum_{i=1}^{h-1}\frac{\mathbf{v}_i\mathbf{v}_i^T}{\mathbf{v}_i^T\mathbf{v}_i}\right)\mathbf{p}_h \\ &= \left(I_p - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\right)\mathbf{p}_h\end{aligned}$$

B Proof of Equations 6

As stated above, we will use the approach of matrix diffential calculus as proposed by Magnus and Neudecker [14], a work in which they argue that their appraoch is the only sensible approach to correctly define Jacobian matrices.

The Jacobian matrix of a matrix function E with respect to a matrix F is defined as J in the following expression:

$$\mathrm{d}\,\mathrm{vec}(E) = J\,\mathrm{d}\,\mathrm{vec}(F)$$

The authors apply this approach to a myriad of examples which might courantly occur in practice.
For the starting value, it holds:

$$\mathrm{d}\,\mathrm{vec}(\mathbf{a}_1) = X^T\,\mathrm{d}\,\mathrm{vec}(\mathbf{y})$$

We will now proceed with the proof of Equation 6a. The Equation to differentiate is Equation 5a. This yields at first:

$$\frac{\partial \mathbf{r}_h}{\partial \mathbf{y}} = \frac{\partial \mathbf{r}_{h-1}}{\partial \mathbf{y}} - \frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\frac{\partial \mathbf{r}_{h-1}}{\partial \mathbf{y}} - (\mathbf{r}_{h-1}^T \otimes I_p)\frac{\partial\left(\frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\right)}{\partial \mathbf{y}}$$

Only the last term still contains an unknown derivative. At first we can split it up into a derivative of the numerator and a derivative of the denominator:

$$(\mathbf{r}_{h-1}^T \otimes I_p)\frac{\partial\left(\frac{\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}\right)}{\partial \mathbf{y}} = (\mathbf{r}_{h-1}^T \otimes I_p)\left[\frac{\partial\left(\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T\right)/\partial \mathbf{y}}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}} - \frac{\mathbf{v}_{h-1}\otimes\mathbf{v}_{h-1}}{\left(\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}\right)^2}\frac{\partial\left(\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}\right)}{\partial \mathbf{y}}\right]$$

We will treat both terms in this expression successively. From Magnus and Neudecker,we know that the first term yields:

$$\frac{\left(\mathbf{v}_{h-1}\otimes I_p + I_p \otimes \mathbf{v}_{h-1}\right)\frac{\partial \mathbf{v}_{h-1}}{\partial \mathbf{y}}}{\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}}$$

From Magnus and Neudecker, the second term yields:

$$-2\frac{\mathbf{v}_{h-1}\otimes\mathbf{v}_{h-1}\mathbf{v}_{h-1}^T\frac{\partial \mathbf{v}_{h-1}}{\partial \mathbf{y}}}{\left(\mathbf{v}_{h-1}^T\mathbf{v}_{h-1}\right)^2}$$

A basic rule of the Kronecker product states that:

$$(A\otimes B)(C\otimes D) = AC\otimes BD$$

Application of this rule to the previous expressions finally proves Equation 6a.
The proof of Equations 6c is trivial.
The proof of Equations 6b and 6d are analogous to the aforementioned proof of Equation 6a.
Differentiating Equation 6e yields:

$$\begin{aligned}\mathrm{d}\,\mathrm{vec}(\mathbf{b}_h) &= \mathrm{vec}\left(\mathrm{d}(\mathbf{r}_h)\mathbf{r}_h^T\mathbf{s} + \mathbf{r}_h\,\mathrm{d}(\mathbf{r}_h)^T\mathbf{s} + \mathbf{r}_h\mathbf{r}_h^T\,\mathrm{d}\,\mathbf{s}\right)\\ &= \left(\mathbf{s}^T\mathbf{r}_h \otimes I_p\right)\mathrm{d}\,\mathrm{vec}(\mathbf{r}_h) + \left(\mathbf{r}_h^T \otimes \mathbf{s}^T\right)\mathrm{d}\,\mathrm{vec}(\mathbf{r}_h) + \mathbf{r}_h\mathbf{r}_h^T\,\mathrm{d}\,\mathrm{vec}(\mathbf{s})\end{aligned}$$

Hereunto, a basic rule of the vectorization operator has been used:

$$\mathrm{vec}(ABC) = \left(C^T \otimes A\right)\mathrm{vec}(B)$$

Moreover, $\mathrm{d}\,\mathrm{vec}(\mathbf{s}) = \mathrm{d}\,\mathrm{vec}(\mathbf{a}_1)$. This concludes the proof of Equations 6.

C Proof of Equations 13

Again, the approach of matrix differential calculus is used. The algorithm is started at $\mathbf{r}_1$. Since it equals $X^T\mathbf{y}$, the differential is readily found as:

$$\mathrm{d}\,\mathbf{r}_1 = \mathrm{d}\,X^T\mathbf{y}$$

Vectorizing both sides of the equation leads to:

$$\mathrm{d}\,\mathrm{vec}(\mathbf{r}_1) = \mathrm{d}\,\mathrm{vec}\left(I_p X^T\mathbf{y}\right) = \mathrm{d}\,\mathrm{vec}\left(\mathbf{y}^T X I_p\right) = I_p \otimes \mathbf{y}^T\,\mathrm{d}\,\mathrm{vec}(X)$$

Hence, equation 12 has been proved.
Equations 13a, 13b, 13d and 13e are similar to their analogues in the algorithm 6, except for the matrix with respect to which is being differentiated. Hence, the proofs will not explicitly be given here.
For $\mathrm{d}\,\mathbf{p}_h$ it holds that:

$$\begin{aligned}\mathrm{d}\,\mathrm{vec}(\mathbf{p}_h) &= \mathrm{vec}\left(\mathrm{d}\,X^T X\mathbf{r}_h + X^T\,\mathrm{d}\,X\mathbf{r}_h + S\,\mathrm{d}\,\mathbf{r}_h\right)\\ &= \left(I_p \otimes \mathbf{r}_h^T X^T + \mathbf{r}_h^T \otimes X^T + S\frac{\partial \mathbf{r}_h}{\partial X}\right)\mathrm{d}\,\mathrm{vec}(X)\end{aligned}$$

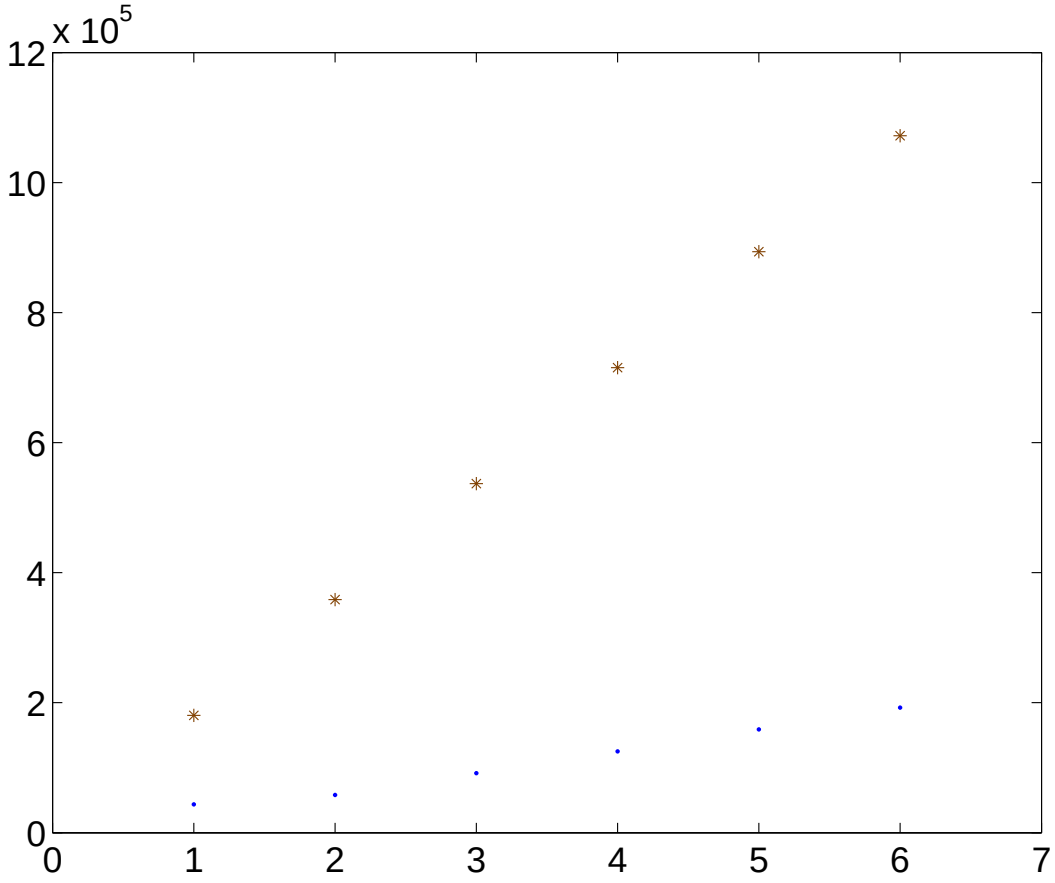

Fig. 1. Number of flops required to compute $\left(\frac{\partial \hat{\mathbf{b}}_h}{\partial \mathbf{y}}\right)$ using Denham's [3] algorithm ($*$) and our algorithm ($\bullet$)