A Linearization Method for Partial Least Squares Regression Prediction Uncertainty

Ying Zhang, Tom Fearn

Department of Statistical Science, University College London, Gower Street, London, WC1E 6BT, U.K.

Abstract

We study a local linearization approach put forward by Romera to provide an approximate variance for predictions in partial least squares regression. We note and correct some problems with the original formulae, study the stability of the resulting approximation using some simulations, and suggest an alternative method of computation using a parametric bootstrap. The alternative method is more stable than the algebraic approximation and is faster when the number of predictors is large.

Keywords: multivariate calibration, partial least squares regression, mean squared prediction error, linearization, parametric bootstrap

1. Introduction

Attaching a variance to the predictions made by a partial least squares (PLS) regression model is not straightforward because the factor scores on which the linear predictor is based are themselves nonlinear functions of the data. Various approximate methods have been proposed, see Zhang and Garcia-Munoz [1] for a recent review, including at least two different approaches that involve local linearizations of the prediction formula. The method of Denham (Denham [2], Serneels et al. [3], and Phatak et al. [4]) expands about the observed value of the dependent variable. A more recent method, due to Romera [5] expands about the observed variances and covariances of all the variables in the data. This is fundamentally different from Denham's approach in that it takes into account the variability in the predictors as well as that in the response variable. In trying to implement this latter approach as part of a comparative study of methodologies, we discovered some problems with the formulae presented in Romera [5]. The current paper corrects these formulae, studies their stability, and suggests an alternative computational approach using a parametric bootstrap that is more stable and is also faster when the dimension of the explanatory variables is large.

Email addresses: ying.zhang@ucl.ac.uk (Ying Zhang), t.fearn@ucl.ac.uk (Tom Fearn)

2. Theory

Suppose we have calibration and prediction sets of data generated from the following linear models

$$\dot{\mathbf{y}}_c = \beta_0 + \dot{\mathbf{X}}_c\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad (1)$$

$$\dot{\mathbf{y}}_p = \beta_0 + \dot{\mathbf{X}}_p\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad (2)$$

where $\dot{\mathbf{y}}_c$ and $\dot{\mathbf{y}}_p$ are calibration and prediction set response variables, $\dot{\mathbf{X}}_c$ $(n \times k)$ and $\dot{\mathbf{X}}_p$ $(n_p \times k)$ are calibration and prediction explanatory variable matrices, β_0 and $\boldsymbol{\beta}$ $(k \times 1)$ are intercept and regression coefficients, and $\boldsymbol{\epsilon}$ is the error term that has a normal distribution with mean zero and variance σ_ϵ^2. The dot on, for example, $\dot{\mathbf{y}}_c$ denotes an un-centered variable, and its corresponding centered variable is $\mathbf{y}_c$. To apply PLS regression to such data Romera [5] employs an orthogonal scores algorithm.

2.1. Orthogonal Scores Algorithm

The orthogonal scores algorithm by Martens and Næs [6] is simple, stable and widely used. With the number of factors chosen to be a, the i-th step of the algorithm gives the results for the i-th factor, where $i = 1, \cdots, a$.

2.1.1. Calibration

The algorithm starts from the centered calibration data matrix, $\mathbf{X}_{c_1} = \mathbf{X}_c$.

$$\begin{aligned}
\mathbf{w}_i &= \mathbf{X}'_{c_i}\mathbf{y}_c \\
\mathbf{t}_i &= \mathbf{X}_{c_i}\mathbf{w}_i \\
\mathbf{p}_i &= \mathbf{X}'_{c_i}\mathbf{t}_i/(\mathbf{t}'_i\mathbf{t}_i) \\
q_i &= \mathbf{y}'_c\mathbf{t}_i/(\mathbf{t}'_i\mathbf{t}_i) \\
\mathbf{X}_{c_{i+1}} &= \mathbf{X}_{c_i} - \mathbf{t}_i\mathbf{p}'_i
\end{aligned}$$

In the i-th step of the algorithm, the column vector $\mathbf{w}_i$ $(k \times 1)$ is the weight vector defined by the covariance between $\mathbf{X}_{c_i}$ and $\mathbf{y}_c$. The $n \times a$ score matrix $\mathbf{T} = (\ \mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_a\ )$ is orthogonal. The $k \times a$ weight matrix is $\mathbf{W} = (\ \mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_a\ )$, and the $k \times a$ x-loading matrix is $\mathbf{P} = (\ \mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_a\ )$. The y-loadings vector $\mathbf{q}$ is defined as an $a \times 1$ column vector. In the first step, if $\mathbf{w}_i$ were scaled to be of length one, the algorithm would become more stable, and it would be easier to compare scores, but the normalization would not change the regression coefficient estimate. Helland [7] shows that the PLS1 regression coefficient estimates can be written as

$$\hat{\boldsymbol{\beta}} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{q}. \quad (3)$$

The scores can also be written as $\mathbf{T} = \mathbf{X}_c\mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}$.

2.1.2. Prediction

A prediction $\hat{y}_p$ can be produced via the score of $\mathbf{x}_p$ $(1 \times k)$. In contrast to the calibration, where $\mathbf{t}_i$ is a column of $\mathbf{T}$, the predictor score $\mathbf{t}_p$ is a row vector, $\mathbf{t}_p = (\ t_{p_1} \quad t_{p_2} \quad \cdots \quad t_{p_a}\ )$, and the t_{p_i} are computed recursively as

$$\begin{aligned}
t_{p_i} &= \mathbf{x}_{p_i}\mathbf{w}_i \\
\mathbf{x}_{p_{i+1}} &= \mathbf{x}_{p_i} - t_{p_i}\mathbf{p}'_i
\end{aligned}$$

with $\mathbf{x}_{p_1} = \dot{\mathbf{x}}_p - \bar{\dot{\mathbf{x}}}$. Equivalently, $\mathbf{t}_p = \mathbf{x}_p\mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}$. The prediction is $\hat{\dot{y}}_p = \bar{\dot{y}} + \mathbf{t}_p\mathbf{q}$.

2.2. A Random Sampling Model for the Data

We suppose that the $(k+1)\times 1$ vector $\dot{\mathbf{c}} = (\ \dot{y}\quad \dot{\mathbf{x}}\ )'$, comprising dependent and predictor variables from one case from either the calibration or prediction set, is randomly sampled from a distribution for which the covariance of $\dot{y}$ and $\dot{\mathbf{x}}$ is $\boldsymbol{\gamma} = (\ \gamma_1\quad \gamma_2\quad \cdots\quad \gamma_k\ )'$, and the variance matrix of $\dot{\mathbf{x}}$ is $\boldsymbol{\Sigma}$ with elements σ_{ij}, $1 \le i, j \le k$. These parameters can be put in a $k(k+3)/2\times 1$ vector $\boldsymbol{\phi} = (\ \boldsymbol{\gamma}'\quad vecut(\boldsymbol{\Sigma})'\ )'$, where *vecut* denotes an operator that returns a column vector whose elements are taken in order along the rows, including the diagonal elements, from the upper triangular part of a symmetric matrix. Let the $k\times 1$ vector $\mathbf{s}_{xy} = \mathbf{X}_c'\mathbf{y}_c$ and the $k\times k$ matrix $\mathbf{S}_{xx} = \mathbf{X}_c'\mathbf{X}_c$ be the sample sums of squares and products for the calibration set. Then we denote by $\mathbf{b} = \left(\ \mathbf{s}_{xy}'\quad vecut(\mathbf{S}_{xx})'\ \right)'$ the vector random variable made up of these quantities, and by $\mathbf{b}_0$ the actual observed value of the random variable for a particular calibration set. The random variable $\mathbf{b}$ is an unbiased estimator of $(n-1)\boldsymbol{\phi}$.

2.3. Romera's Approach

Romera [5] explores the dependence of regression coefficients $\hat{\boldsymbol{\beta}}$ on $\mathbf{b}$ via the y-loadings $\mathbf{q}$. The estimated y-loadings can be expanded about the observed value $\mathbf{b}_0$ of $\mathbf{b}$ according to the first-order Taylor expansion

$$\mathbf{q}_{\mathbf{b}} \approx \mathbf{q}_{\mathbf{b}_0} + \mathbf{J}(\mathbf{b}-\mathbf{b}_0).$$

The approximate variance of the estimated y-loadings $\mathrm{Var}(\mathbf{q}) \approx \mathbf{J}\mathrm{Var}(\mathbf{b})\mathbf{J}'$, where the Jacobian matrix $\mathbf{J}$ $(a\times k(k+3)/2)$ is the first derivative of $\mathbf{q}$ with respect to $\mathbf{b}$ evaluated at $\mathbf{b}_0$, $\mathbf{J} = (\partial\mathbf{q}/\partial\mathbf{b})_{\mathbf{b}_0}$. Romera [5] then uses $\hat{\boldsymbol{\beta}} = \mathbf{W}\mathbf{q}$ which gives $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \mathbf{W}\mathrm{Var}(\mathbf{q})\mathbf{W}'$, so the approximate variance of $\mathbf{x}_p\hat{\boldsymbol{\beta}}$ becomes

$$\mathrm{Var}(\mathbf{x}_p\hat{\boldsymbol{\beta}}) \approx \mathbf{x}_p\mathbf{W}\mathbf{J}\mathrm{Var}(\mathbf{b})\mathbf{J}'\mathbf{W}'\mathbf{x}_p'.$$

However, there are problems with $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \mathbf{W}\mathrm{Var}(\mathbf{q})\mathbf{W}'$. As shown in Equation (3), $\hat{\boldsymbol{\beta}} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{q}$ for the orthogonal scores algorithm, and not $\hat{\boldsymbol{\beta}} = \mathbf{W}\mathbf{q}$, which is the result of the PLS1 orthogonal loadings algorithm. There is also a second problem, in that the weight matrix $\mathbf{W}$ is dependent on $\mathbf{b}$, so $\mathbf{W}$ cannot be treated as fixed.

2.4. Corrected Formulae

Linearizing around $\mathbf{b}_0$ we have the following approximate formula for the variance of $\mathbf{x}_p\hat{\boldsymbol{\beta}}$ for fixed $\dot{\mathbf{x}}_p$

$$\mathrm{Var}(\mathbf{x}_p\hat{\boldsymbol{\beta}}) \approx \mathbf{x}_p\left(\frac{\partial\hat{\boldsymbol{\beta}}}{\partial\mathbf{b}}\right)_{\mathbf{b}_0}\mathrm{Var}(\mathbf{b})\left(\frac{\partial\hat{\boldsymbol{\beta}}}{\partial\mathbf{b}}\right)_{\mathbf{b}_0}'\mathbf{x}_p' = V_L. \tag{4}$$

To calculate this we need expressions for $\mathrm{Var}(\mathbf{b})$ and for $(\partial\hat{\boldsymbol{\beta}}/\partial\mathbf{b})_{\mathbf{b}_0}$. If we assume that the $\dot{\mathbf{c}}$ defined in Section 2.2 is normally distributed, both the distribution and the variance of $\mathbf{b}$ are known from standard normal theory. Appendix A gives the distribution of $\mathbf{b}$. The algebra for $(\partial\hat{\boldsymbol{\beta}}/\partial\mathbf{b})_{\mathbf{b}_0}$ is in Appendix B.

2.5. Estimating $\mathrm{Var}(\hat{\boldsymbol{\beta}})$ *by a Parametric Bootstrap*

An alternative approach that avoids all the algebra is to use a parametric bootstrap to estimate $\mathrm{Var}(\hat{\boldsymbol{\beta}})$. For the m-th bootstrap sample $(m = 1,\ldots,M)$, a sum of squares and products matrix is drawn from the Wishart distribution in Appendix A and $\mathbf{b}_m$ is extracted from it. Now we need to calculate $\hat{\boldsymbol{\beta}}_m^B$ from $\mathbf{b}_m$, rather than from $\mathbf{X}_c$ and $\mathbf{y}_c$. The formula for doing this were given by Romera [5] and are presented in Appendix C. The variance of regression coefficients from the

bootstrap algorithm is $\text{Var}\,(\hat{\boldsymbol{\beta}}^B) = \frac{n}{n+1}\frac{1}{M-1}\sum_{m=1}^{M}(\hat{\boldsymbol{\beta}}_m^B - \bar{\boldsymbol{\beta}})(\hat{\boldsymbol{\beta}}_m^B - \bar{\boldsymbol{\beta}})'$, where $\bar{\boldsymbol{\beta}} = \frac{1}{M}\sum_{m=1}^{M}\hat{\boldsymbol{\beta}}_m^B$ and the factor $\frac{n}{n+1}$ adjusts for the bias in the bootstrap (See Efron and Tibshirani [8]). The approximate variance of $\mathbf{x}_p\hat{\boldsymbol{\beta}}$ is

$$\text{Var}\,(\mathbf{x}_p\hat{\boldsymbol{\beta}}) \approx \mathbf{x}_p\text{Var}\,(\hat{\boldsymbol{\beta}}^B)\mathbf{x}_p' = V_B. \tag{5}$$

3. Numerical Experiments

In this section, we use simulation studies to investigate how the linearization method and its bootstrap version perform under different conditions. Our purpose is not to carry out an extensive simulation study, but to demonstrate some of the properties of the method using a few simple simulations. Each of the N repetitions in the simulation generates a calibration set of size $n = 200$ and a prediction set of size $n_p = 200$ using the models in Equations (1) and (2) but with $\boldsymbol{\epsilon}$ set to zero in Equation (2). Taking the additive noise component out of the predictions enables the performance of the variance formulae in Equations (4) and (5) to be seen more clearly. The explanatory variables are independently and normally distributed with mean 0 and variances $(\ \sigma_1^2 \quad \sigma_2^2 \quad \cdots \quad \sigma_k^2\ )$ in both calibration and prediction sets. The number of PLS factors is fixed to be a in each of the repetitions. Of course an extensive simulation study would need to explore both correlated predictors and the effect of extrapolation, but our purpose here is just to demonstrate some of the properties of the methods investigated using a few simple simulations.

For each of the $N \times n_p$ predictions in the simulation we calculate a squared prediction error and the estimated variances V_L and V_B given by Equations (4) and (5). These variance formulae neglect the contributions from the variation of $\bar{\mathbf{x}}$ and $\bar{y}$ over repeated drawing of the calibration set. The contribution from $\bar{y}$, σ_ϵ^2/n, was added to each of the estimated variances, so that the Lin variance formula becomes $\sigma_\epsilon^2/n + V_L$, and the Linb variance formula is $\sigma_\epsilon^2/n + V_B$. In practice of course one would need to use an estimate for σ_ϵ^2; the rationale for using the known value here is to focus on the performance of V_L and V_B. The contribution from $\bar{\mathbf{x}}$ is of order k/n^2 and can be neglected for the examples considered here.

To examine the performance of Lin and Linb we plot observed squared error and the two estimated variances against either V_L or V_B after taking averages in 20 bins defined by the x-axis variable. The bins were set up using percentage points of a scaled chi-squared random variable with scale and degrees of freedom chosen so that its first two moments match those of the observed values of either V_L or V_B. This gives roughly equal numbers of observations per bin.

We begin by studying two simulations with $k = 2$ and $a = 1$. In the first the linearization is stable. In the second the linearization approximation performs badly.

3.1. Simulation: $k = 2, a = 1, \sigma_1^2 = 25, \sigma_2^2 = 1, \beta_0 = \beta_1 = 1, \beta_2 = 0, \sigma_\epsilon^2 = 0.25, N = 10000.$

The first predictor variable, which has a non-zero regression coefficient, has a much bigger variance than the second, which has a zero coefficient. Not surprisingly, PLS works rather well, and both Lin and Linb also work well (Figure 1). The plot against V_B looks equally good. Figure 2 shows how the estimated regression coefficients change with $\mathbf{b}$. $\hat{\beta}_1$ is always close to 1 while $\hat{\beta}_2$ depends on two elements of $\mathbf{b}$ in a linear fashion.

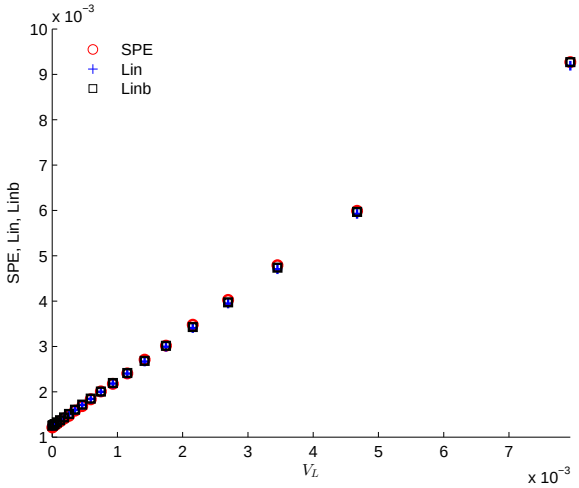


Figure 1: PLS estimated variances and actual squared prediction error versus V_L. $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_0 = \beta_1 = 1, \beta_2 = 0$, $\sigma_\epsilon^2 = 0.25$. SPE: squared prediction error $(\dot{y}_p - \hat{\dot{y}}_p)^2$. Lin: $V_l + \sigma_\epsilon^2/n$. Linb: $V_B + \sigma_\epsilon^2/n$.

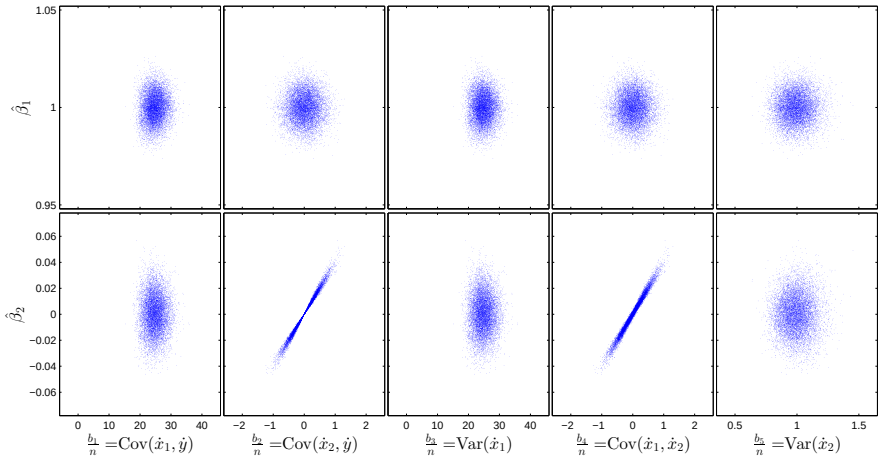


Figure 2: PLS $\hat{\boldsymbol{\beta}}$ against $\frac{\mathbf{b}}{n}$ when $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_0 = \beta_1 = 1, \beta_2 = 0$, $\sigma_\epsilon^2 = 0.25$.

3.2. Simulation: $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_1 = 0$, $\beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$, $N = 10000$.

This is a more difficult case for PLS; the first predictor has the larger variance but has no contribution to the regression, whereas the second, with a smaller variance, is linked to the response variable. The large variance of the first predictor means that even a small sample correlation with the response variable is enough to gain it weight in the PLS factor. The fact that the sign of $\hat{\boldsymbol{\beta}}$ switches along with the sign of this correlation leads to the breakdown of the linearization approximation. Figure 3(a) and Figure 3(b) show that both Lin and Linb fail, though in different ways.

Figure 4 and Figure 5 show why Lin fails. In Figure 4 we can see that the distribution of $\hat{\beta}_1$ as **b** varies is bimodal, with the mode switching as the signs of b_1 and b_4 change. In Figure 5 we see how the local linearization method breaks down for one calibration set. The blue dotted lines were computed by changing b_1 and recalculating $\hat{\boldsymbol{\beta}}$ using the PLS algorithm. They represent how $\hat{\beta}_1$ and $\hat{\beta}_2$ vary with small changes of b_1. The red dashed lines are the linear approximations to the relationships between the estimated regression coefficients and b_1. The are fine locally, but only over a very narrow range.

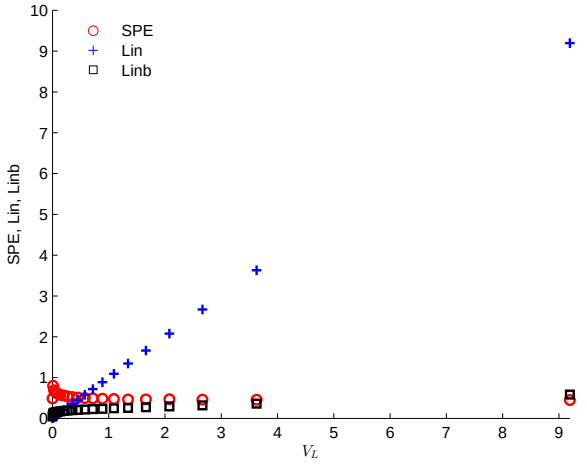
The failure of Linb is less dramatic. It underestimates SPE for two reasons: the bootstrap underestimates the actual variance of the $\hat{\boldsymbol{\beta}}$, and it ignores a contribution from the bias in the PLS $\hat{\boldsymbol{\beta}}$ which is not negligible for this example. The underestimation of the variance of the predictions can be explained by considering Figure 4, and in particular the top left hand panel. The repeated training sets are generated from a joint distribution for response and predictor variables in which b_1 is centred on zero. It can be seen from Figure 4 that the resulting $\hat{\beta}_1$ values will have a bimodal distribution with equal weights in each mode. The bootstrap estimation procedure for any particular training set will be centred on the observed b_1 for that set, which in general will not be zero. The bootstrap $\hat{\beta}_1$ values will usually still have a bimodal distribution but now with unequal weights in the two modes and consequently with smaller variance than that of the $\hat{\beta}_1$'s in the repeated training sets. This accounts for about 20% of the discrepancy between Linb and SPE. The rest is due to a substantial bias in the PLS $\hat{\boldsymbol{\beta}}$.

3.3. Simulation: $k = 3$, $a = 2$, $\sigma_1^2 = \sigma_2^2 = 25$, $\sigma_3^2 = 1$, $\beta_0 = \beta_1 = \beta_2 = 1$, $\beta_3 = 0$, $\sigma_\epsilon^2 = 0.25$, $N = 10000$.

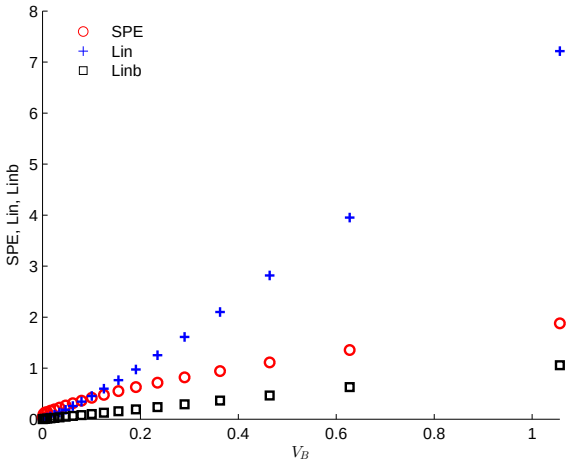
The previous simulation was deliberately chosen to be a difficult case for PLS and it is perhaps not surprising that the linearization fails. Unfortunately however it can also fail in what appear to be innocuous examples. In this simulation we have two predictor variables with big variances and strong correlations with the response, and a third predictor with much smaller variance and no correlation. The bootstrap version, Linb, works well, but the algebraic version, Lin, fails badly for some calibration sets. Figure 6 shows, for one of these calibration sets, how the coefficient vector $\hat{\beta}$ changes with b_4 (the sum of squares of the first predictor) in the vicinity of the observed value. As before, the linear approximation has much too narrow a range of validity and leads to a gross overestimation of the variance of $\hat{\beta}$.

3.4. Simulation: $k = 24$, $a = 7$, $\sigma_1^2 = 64$, $\sigma_2^2 = 49$, $\sigma_3^2 = 36$, $\sigma_4^2 = 25$, $\sigma_5^2 = 16$, $\sigma_6^2 = 9$, $\sigma_7^2 = 4$, $\sigma_8^2 = \cdots = \sigma_{24}^2 = 1$, $\beta_0 = 1$, $\beta_1 = 8$, $\beta_2 = 7$, $\beta_3 = 6$, $\beta_4 = 5$, $\beta_5 = 4$, $\beta_6 = 3$, $\beta_7 = 2$, $\beta_8 = \cdots = \beta_{24} = 1$, $\sigma_\epsilon^2 = 0.25$, $N = 500$.

The simulations so far have involved very small numbers of predictor variables. This one has $k = 24$ variables and a=7 factors. Most of the x-variability and most of the predictive power is in the first 7 variables so this is in some sense an easy problem for PLS. The algebraic method,



(a)



(b)

Figure 3: PLS estimated variances and actual squared prediction error versus (a) V_L and (b) V_B. $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1, \beta_1 = 0, \beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$. SPE: squared prediction error $(\dot{y}_p - \hat{\dot{y}}_p)^2$. Lin: $V_L + \sigma_\epsilon^2/n$. Linb: $V_B + \sigma_\epsilon^2/n$.

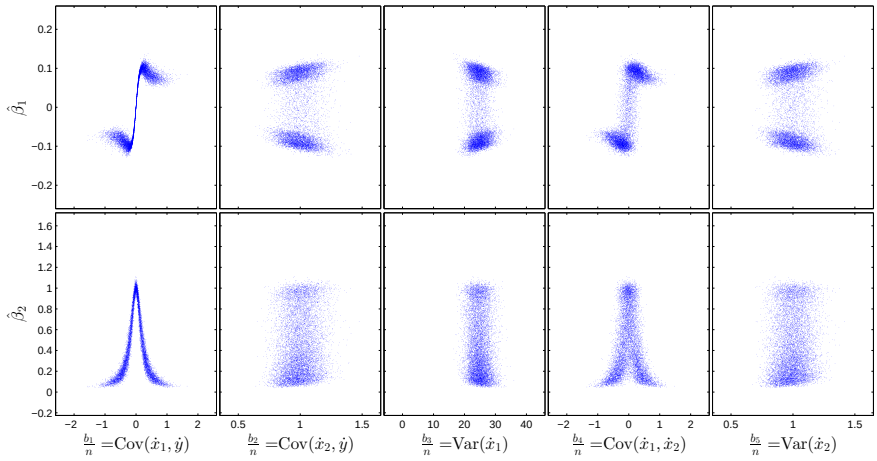


Figure 4: PLS $\hat{\boldsymbol{\beta}}$ against $\frac{\mathbf{b}}{n}$ when $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_1 = 0$, $\beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$.

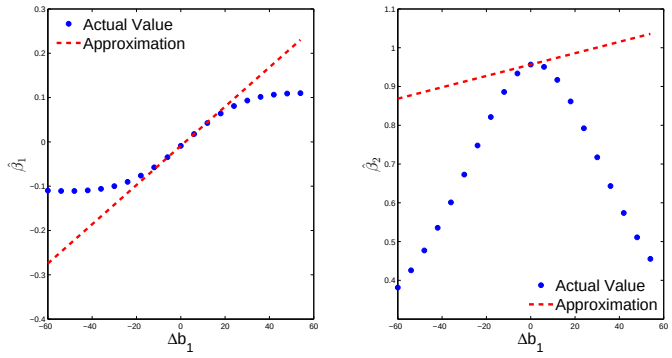


Figure 5: Adequacy of the linearization approximation in the case when $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_1 = 0$, $\beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$. $b_1 = -1.9897$, $b_2 = 215.3367$, $b_3 = 4691.2$, $b_4 = 4.123$, and $b_5 = 224.8093$.

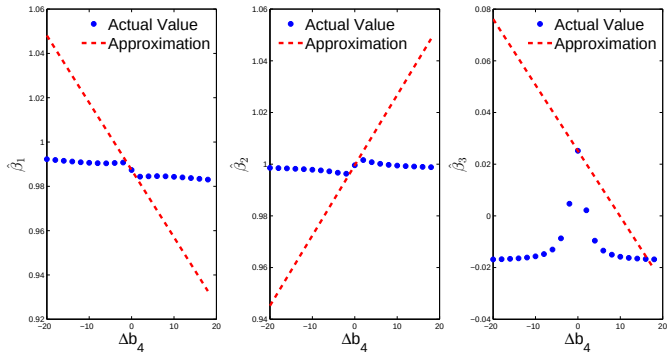


Figure 6: Adequacy of the linearization approximation in the case when $k = 3$, $a = 2$, $\sigma_1^2 = \sigma_2^2 = 25$, $\sigma_3^2 = 1$, $\beta_1 = \beta_2 = 1$, $\beta_3 = 0$, $\sigma_\epsilon^2 = 0.25$. $b_1 = 4145.5$, $b_2 = 4192.6$, $b_3 = -78.6$, $b_4 = 4686.5$, $b_5 = -483.1$, $b_6 = -21.8$, $b_7 = 4674.7$, $b_8 = -61.5$, and $b_9 = 171.3$.

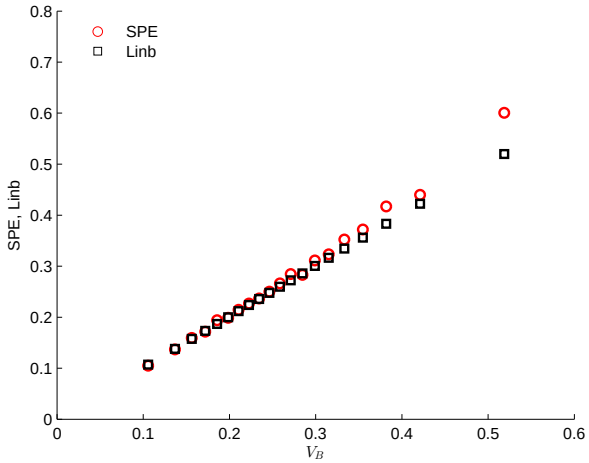


Figure 7: PLS estimated variances and actual squared prediction error versus V_B. $k = 24$, $a = 7$; $\sigma_1^2 = 64$, $\sigma_2^2 = 49$, $\sigma_3^2 = 36$, $\sigma_4^2 = 25$, $\sigma_5^2 = 16$, $\sigma_6^2 = 9$, $\sigma_7^2 = 4$, $\sigma_8^2 = \cdots = \sigma_{24}^2 = 1$; $\beta_0 = 1, \beta_1 = 8, \beta_2 = 7, \beta_3 = 6, \beta_4 = 5, \beta_5 = 4, \beta_6 = 3$, $\beta_7 = 2$, and $\beta_8 = \cdots = \beta_{24} = 0$, $\sigma_\epsilon^2 = 0.25$. SPE: squared prediction error $(\dot{y}_p - \hat{\dot{y}}_p)^2$. Linb: $V_B + \sigma_\epsilon^2/n$.

Lin, breaks down as before, giving extreme overestimates of variance for a small proportion of calibration sets. The bootstrap version, Linb, works reasonably well, as can be seen from Figure 7. It slightly underestimates the average squared errors, especially at the top end of the scale. This time the discrepancy is all due to the neglected bias; the bootstrap makes a good job of estimating the variance of $\hat{\boldsymbol{\beta}}$. Interestingly, with N reduced to 500, which is large enough to give reproducible results, Linb is slightly faster to compute than Lin for this example. This is because the computations for Lin involve matrices of size $(k+1)^2 \times (k+1)^2$, which is 625×625 with $k = 24$. For much larger problems, Linb is a good deal faster than Lin. Not much effort has been put into optimising the code for either calculation, so a detailed comparison of timings would not mean a lot, but it does seem reasonable to conclude that Linb is probably preferable to Lin on grounds of speed as well as stability.

3.5. A Brief Discussion about the Use of a Real Data Set

At this point in the paper one might expect to see the methods illustrated on a real data set. We have applied Linb to real data sets with k up to 100. It gives plausible looking variances in a reasonable amount of computing time. Apart from this there is not much to be learned. With a fixed calibration set it is not possible to evaluate the performance of the method. It would be like trying to evaluate the correctness of, for example, the formula for the variance of a sample mean using a single fixed data set. One might do something with sample splitting, but it would require an enormous data set to get any useful results.

4. Estimating the Missing Components of the Error

Equations (4) and (5) only estimate the variance in the predictions that is a consequence of the variance in $\hat{\beta}$. There are several more contributions to the total error: these come from the variance in $\bar{y}$ and $\bar{\mathbf{x}}$, the observation error that was omitted from the $\dot{\mathbf{y}}_p$ in our simulations, and the bias that results from using PLS rather than multiple linear regression. One possible approach to estimating the joint contribution of all of these would be to use either cross-validation or a separate test set to find an empirical estimate of the average predictive mean squared error and subtract from this the average V_B (or V_L) for the samples predicted. The remainder (truncated at zero should it turn out to be negative) is an estimate of the sum of the missing components, and could be added to the sample-specific V_B to quantify the uncertainty in any future predictions. The main limitation of this approach is that it applies the average squared bias to all predictions, whereas the bias will in general depend on $\mathbf{x}_p$. However, if we had enough data to estimate this dependence we could have used multiple regression rather that PLS in the first place, so it seems unlikely that we can do any better than this in general.

5. Conclusion

Although we have been able to provide a corrected version of Romera's linearization method, we are forced by the simulations to the conclusion that it is probably not a good idea to use this algebraic version in practice. In the simulations it only fails for occasional calibration sets, but when it does fail, it fails badly. For a single real data set there is no simple way of checking whether this is one of the bad cases, and so the risk that the linear approximation is very poor will always be present. The bootstrap version, as well as being much easier to implement, is also more stable and performs reasonably well in the, admittedly very limited, simulations. It is a

variance formula, so neglects bias, but at least the average squared bias can be accounted for as described in the previous section.

Appendix A. The Distribution of b

If we assume the random variable $\dot{\mathbf{c}} = (\ \dot{y} \quad \dot{\mathbf{x}}\ )'$ defined in Section 2.2 has a multivariate normal distribution with variance matrix $\boldsymbol{\psi}$, then the sample sums of squares and products matrix, $\mathbf{G} = \begin{pmatrix} \sum y_i^2 & \mathbf{s}'_{xy} \\ \mathbf{s}_{xy} & \mathbf{S}_{xx} \end{pmatrix}$, based on a sample of size n has a Wishart distribution with parameters $\boldsymbol{\psi}$ and $n-1$. Magnus and Neudecker [9] gives the variance of the column stacked vector $vec(\mathbf{G})$,

$$\mathrm{Var}\{vec\,(\mathbf{G})\} = n(\mathbf{I}_{(1+k)^2} + \mathbf{K})(\boldsymbol{\psi} \otimes \boldsymbol{\psi}),$$

where *vec* denotes the operator that extracts columns from a matrix to form a column vector, and $\otimes$ denotes the Kronecker product. $\mathbf{K}$ is a commutation matrix $\mathbf{K} = \sum_{i=1}^{1+k}\sum_{j=1}^{1+k} \mathbf{M}_{ij} \otimes \mathbf{M}'_{ij}$. $\mathbf{M}_{ij}$ is a $(1+k)\times(1+k)$ square matrix with the (i, j)-th element equal to 1 and all other elements being zero. $\mathrm{Var}\,(\mathbf{b})$ can be obtained by selecting relevant elements from $\mathrm{Var}\{vec\,(\mathbf{G})\}$, because all the elements of $\mathbf{b}$ belong to $vec\,(\mathbf{G})$.

Appendix B. The derivative of $\hat{\boldsymbol{\beta}}$ with respect to b

In this section, we present the algebra needed to calculate $(\partial\hat{\boldsymbol{\beta}}/\partial\mathbf{b})_{\mathbf{b}_0}$ for use in the linearized approximation presented in Equation (4). First we define some notation.

Derivative Let $\mathbf{g}$ be an $l\times 1$ column vector, and $\mathbf{v}$ be an $r\times 1$ column vector. The derivative $\partial\mathbf{g}/\partial\mathbf{v}$ is an $l\times r$ matrix with the (i, j)-th element defined as $\partial g_i/\partial v_j$.

Operators The operator *diag* extracts the diagonal terms from a symmetric matrix as a column vector. The operator *vecut* returns a column vector whose elements are taken in order along the rows, including the diagonal elements, from the upper triangular part of a symmetric matrix.

In what follows the notation w_{il} needs to be interpreted with care: the subscripts do not refer to the element's position in the weight matrix $\mathbf{W}$. Instead, w_{il} is the l-th element of $\mathbf{w}_i$, which is the i-th column of $\mathbf{W}$. Let $\hat{\beta}_l$ be the l-th element of $\hat{\boldsymbol{\beta}}$, $(l = 1, \ldots, k)$. Let $\tilde{\mathbf{w}}_l$ denote the l-th row vector of the weight matrix $\mathbf{W}$, where $\tilde{\mathbf{w}}_l = (\ w_{1l} \quad w_{2l} \quad \cdots \quad w_{al}\ )$, and let $\tilde{\mathbf{R}} = (\mathbf{P}'\mathbf{W})^{-1}$. Then

$$\hat{\beta}_l = \tilde{\mathbf{w}}_l\tilde{\mathbf{R}}\mathbf{q},$$

and

$$(\frac{\partial\hat{\beta}_l}{\partial\mathbf{b}})_{\mathbf{b}_0} = \tilde{\mathbf{w}}_l\tilde{\mathbf{R}}(\frac{\partial\mathbf{q}}{\partial\mathbf{b}})_{\mathbf{b}_0} + \mathbf{q}'(\frac{\partial\tilde{\mathbf{w}}_l\tilde{\mathbf{R}}}{\partial\mathbf{b}})_{\mathbf{b}_0}. \tag{B.1}$$

Appendix B.1 and Appendix B.2 below give the calculations of $(\partial\mathbf{q}/\partial\mathbf{b})_{\mathbf{b}_0}$ and $(\partial\tilde{\mathbf{w}}_l\tilde{\mathbf{R}}/\partial\mathbf{b})_{\mathbf{b}_0}$ respectively.

Appendix B.1. $(\partial\mathbf{q}/\partial\mathbf{b})_{\mathbf{b}_0}$

At the i-th step of the PLS algorithm, we define a working sum of squares and product vector as

$$\mathbf{b}_i = \left(w_{i1} \quad \cdots \quad w_{ik} \quad vecut\,(\mathbf{S}_i)'\right)',$$

where $w_{i1}, \ldots, w_{ik}$ are taken from the i-th column of the weight matrix $\mathbf{W}$, and $\mathbf{S}_i = \mathbf{X}'_{c_i}\mathbf{X}_{c_i}$. Thus $\mathbf{b}_1 = \mathbf{b}$, with subsequent versions having had some variability removed. If we define $\mathbf{A}_i$ by

$$\mathbf{A}_i = \mathbf{I} - \mathbf{S}_i\mathbf{w}_i\mathbf{w}'_i/(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i),$$

then the updating formulae may be written in terms of $\mathbf{A}_i$ as

$$\begin{aligned}
\mathbf{A}_i &= \mathbf{I} - \mathbf{S}_i\mathbf{w}_i\mathbf{w}'_i/(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i) \\
\mathbf{X}_{c_{i+1}} &= \mathbf{X}_{c_i} - \mathbf{t}_i\mathbf{p}'_i = \mathbf{X}_{c_i}(\mathbf{I} - \frac{\mathbf{w}_i\mathbf{w}'_i\mathbf{S}_i}{\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i}) = \mathbf{X}_{c_i}\mathbf{A}'_i \\
\mathbf{w}_{i+1} &= \mathbf{X}'_{c_{i+1}}\mathbf{y}_c = \mathbf{A}_i\mathbf{X}'_{c_i}\mathbf{y}_c = \mathbf{A}_i\mathbf{w}_i \qquad \text{(B.2)} \\
\mathbf{S}_{i+1} &= \mathbf{X}'_{c_{i+1}}\mathbf{X}_{c_{i+1}} = \mathbf{A}_i\mathbf{S}_i\mathbf{A}'_i. \qquad \text{(B.3)}
\end{aligned}$$

At the i-th step, according to the chain rule we have

$$(\frac{\partial q_i}{\partial \mathbf{b}})_{\mathbf{b}_0} = \frac{\partial q_i}{\partial \mathbf{b}_i}\frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}}\frac{\partial \mathbf{b}_{i-1}}{\partial \mathbf{b}_{i-2}}\cdots\frac{\partial \mathbf{b}_3}{\partial \mathbf{b}_2}\frac{\partial \mathbf{b}_2}{\partial \mathbf{b}_1}. \qquad \text{(B.4)}$$

Appendix B.1.1 and Appendix B.1.2 give details of the calculations of $\partial q_i/\partial \mathbf{b}_i$ and $\partial \mathbf{b}_{i+1}/\partial \mathbf{b}_i$.

Appendix B.1.1. $\partial q_i/\partial \mathbf{b}_i$

$$\frac{\partial q_i}{\partial \mathbf{b}_i} = \begin{pmatrix} \frac{\partial q_i}{\partial \mathbf{w}_i} & \frac{\partial q_i}{\partial vecut(\mathbf{S}_i)} \end{pmatrix} = \begin{pmatrix} \frac{\partial q_i}{\partial \mathbf{w}_i} & vecut(\frac{\partial q_i}{\partial \mathbf{S}_i}) \end{pmatrix},$$

where

$$\begin{aligned}
\frac{\partial q_i}{\partial \mathbf{w}_i} &= \frac{\partial}{\partial \mathbf{w}_i}(\frac{\mathbf{w}'_i\mathbf{w}_i}{\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i}) \qquad \text{because } q_i = \mathbf{w}'_i\mathbf{w}_i/(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i) \\
&= \frac{\partial}{\partial \mathbf{w}_i}(\mathbf{w}'_i\mathbf{w}_i)\frac{1}{\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i} + \mathbf{w}'_i\mathbf{w}_i\frac{\partial}{\partial \mathbf{w}_i}(\frac{1}{\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i}) \\
&= \frac{2\mathbf{w}'_i}{\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i} - \frac{\mathbf{w}'_i\mathbf{w}_i}{(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i)^2}\frac{\partial \mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i}{\partial \mathbf{w}_i} \\
&= \frac{2\mathbf{w}'_i}{\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i} - \frac{\mathbf{w}'_i\mathbf{w}_i}{(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i)^2}2\mathbf{w}'_i\mathbf{S}_i, \qquad \text{(B.5)}
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial q_i}{\partial \mathbf{S}_i} &= -\frac{\mathbf{w}'_i\mathbf{w}_i}{(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i)^2}\frac{\partial \mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i}{\partial \mathbf{S}_i} \\
&= -\frac{\mathbf{w}'_i\mathbf{w}_i}{(\mathbf{w}'_i\mathbf{S}_i\mathbf{w}_i)^2}\{2\mathbf{w}_i\mathbf{w}'_i - diag\,(\mathbf{w}_i\mathbf{w}'_i)'\mathbf{I}\}. \qquad \text{(B.6)}
\end{aligned}$$

Appendix B.1.2. $\partial \mathbf{b}_{i+1}/\partial \mathbf{b}_i$

The term $\partial \mathbf{b}_{i+1}/\partial \mathbf{b}_i$ used in the chain rule in Equation (B.4) can be decomposed into four blocks:

$$\frac{\partial \mathbf{b}_{i+1}}{\partial \mathbf{b}_i} = \begin{pmatrix} \textcircled{1} & \textcircled{2} \\ \textcircled{3} & \textcircled{4} \end{pmatrix}.$$

Block (1) is a $k \times k$ matrix:

$$\begin{aligned} (1) \quad \frac{\partial \mathbf{w}_{i+1}}{\partial \mathbf{w}_i} &= \frac{\partial \mathbf{A}_i \mathbf{w}_i}{\partial \mathbf{w}_i} \quad \text{using Equation (B.2)} \\ &= \frac{\partial}{\partial \mathbf{w}_i}(\mathbf{w}_i - \frac{\mathbf{S}_i \mathbf{w}_i \mathbf{w}_i' \mathbf{w}_i}{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i}) \\ &= \mathbf{I} - \mathbf{S}_i q_i - \mathbf{S}_i \mathbf{w}_i \frac{\partial q_i}{\partial \mathbf{w}_i}, \end{aligned}$$

where $\partial q_i / \partial \mathbf{w}_i$ is given in Equation (B.5).

Block (2) is a $k \times \frac{k(k+1)}{2}$ matrix:

$$\begin{aligned} (2) \quad \frac{\partial \mathbf{w}_{i+1}}{\partial vecut(\mathbf{S}_i)} &= \frac{\partial}{\partial vecut(\mathbf{S}_i)}(-\frac{\mathbf{S}_i \mathbf{w}_i \mathbf{w}_i' \mathbf{w}_i}{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i}) \\ &= -\mathbf{w}_i' \mathbf{w}_i \{\frac{\partial \mathbf{S}_i \mathbf{w}_i}{\partial vecut(\mathbf{S}_i)} \frac{1}{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i} - \frac{\mathbf{S}_i \mathbf{w}_i}{(\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i)^2} \frac{\partial \mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}\}. \end{aligned}$$

$\partial \mathbf{S}_i \mathbf{w}_i / \partial vecut(\mathbf{S}_i)$ and $\partial \mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i / \partial vecut(\mathbf{S}_i)$ are given below.

$$\frac{\partial \mathbf{S}_i \mathbf{w}_i}{\partial vecut(\mathbf{S}_i)} = \begin{pmatrix} w_{i1} & 0 & 0 & \cdots & 0 \\ w_{i2} & w_{i1} & 0 & \cdots & 0 \\ w_{i3} & 0 & w_{i1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{ik} & 0 & 0 & \cdots & w_{i1} \\ 0 & w_{i2} & 0 & \cdots & 0 \\ 0 & w_{i3} & w_{i2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & w_{ik} & 0 & \cdots & w_{i2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_{ip} \end{pmatrix}'.$$

As shown in Equation (B.6) of Appendix B.1.1,

$$\begin{aligned} \frac{\partial \mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i}{\partial \mathbf{S}_i} &= 2\mathbf{w}_i \mathbf{w}_i' - diag(\mathbf{w}_i \mathbf{w}_i')' \mathbf{I} \\ &= \begin{pmatrix} w_{i1}^2 & 2w_{i1}w_{i2} & \cdots & 2w_{i1}w_{ik} \\ 2w_{i2}w_{i1} & w_{i2}^2 & \cdots & 2w_{i2}w_{ik} \\ \vdots & \vdots & \ddots & \vdots \\ 2w_{ik}w_{i1} & 2w_{ik}w_{i2} & \cdots & w_{ik}^2 \end{pmatrix}. \end{aligned}$$

Hence, we have

$$\frac{\partial \mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i}{\partial vecut(\mathbf{S}_i)} = \begin{pmatrix} w_{i1}^2 & 2w_{i1}w_{i2} & \cdots & 2w_{i1}w_{ik} & w_{i2}^2 & \cdots & w_{ik}^2 \end{pmatrix}.$$

Block (3) is a $\frac{k(k+1)}{2} \times k$ matrix:

Using Equation (B.3) for $\mathbf{S}_{i+1}$,

$$\textcircled{3} \quad \frac{\partial vecut(\mathbf{S}_{i+1})}{\partial \mathbf{w}_i} = \frac{\partial}{\partial \mathbf{w}_i} vecut\{(\mathbf{I} - \frac{\mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i})\mathbf{S}_i(\mathbf{I} - \frac{\mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i})'\}$$
$$= \frac{\partial}{\partial \mathbf{w}_i} vecut(\mathbf{S}_i - \frac{\mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'\mathbf{S}_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i})$$
$$= -\frac{\partial}{\partial \mathbf{w}_i} vecut(\mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'\mathbf{S}_i)\frac{1}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i} + \frac{vecut(\mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'\mathbf{S}_i)}{(\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i)^2} 2\mathbf{w}_i'\mathbf{S}_i, \tag{B.7}$$

Let $\mathbf{u}_i = \mathbf{S}_i\mathbf{w}_i$, then

$$\textcircled{3} \quad \frac{\partial vecut(\mathbf{S}_{i+1})}{\partial \mathbf{w}_i} = -\frac{\partial vecut(\mathbf{u}_i\mathbf{u}_i')}{\partial \mathbf{u}_i}\frac{\partial \mathbf{S}_i\mathbf{w}_i}{\partial \mathbf{w}_i}\frac{1}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i} + \frac{vecut(\mathbf{u}_i\mathbf{u}_i')}{(\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i)^2} 2\mathbf{w}_i'\mathbf{S}_i$$
$$= -\frac{\partial vecut(\mathbf{u}_i\mathbf{u}_i')}{\partial \mathbf{u}_i}\frac{\mathbf{S}_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i} + \frac{vecut(\mathbf{u}_i\mathbf{u}_i')}{(\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i)^2} 2\mathbf{w}_i'\mathbf{S}_i,$$

where $\partial vecut(\mathbf{u}_i\mathbf{u}_i')/\partial \mathbf{u}_i$ is calculated as follows. At the i-th step, let $\mathbf{u}_i = \begin{pmatrix} u_1 & u_2 & u_3 & \cdots & u_k \end{pmatrix}'$, omitting the i subscript for convenience. Then we have

$$\frac{\partial vecut(\mathbf{u}_i\mathbf{u}_i')}{\partial \mathbf{u}_i} = \begin{pmatrix} 2u_1 & 0 & 0 & \cdots & 0 \\ u_2 & u_1 & 0 & \cdots & 0 \\ u_3 & 0 & u_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_k & 0 & 0 & \cdots & u_1 \\ 0 & 2u_2 & 0 & \cdots & 0 \\ 0 & u_3 & u_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & u_k & 0 & \cdots & u_2 \\ 0 & 0 & 2u_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2u_k \end{pmatrix}.$$

Block $\textcircled{4}$ is a $\frac{k(k+1)}{2} \times \frac{k(k+1)}{2}$ matrix:

$$\textcircled{4} \quad \frac{\partial vecut(\mathbf{S}_{i+1})}{\partial vecut(\mathbf{S}_i)} = \frac{\partial}{\partial vecut(\mathbf{S}_i)} vecut(\mathbf{S}_i - \frac{\mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'\mathbf{S}_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i})$$
$$= \mathbf{I} - \frac{\partial vecut(\mathbf{u}_i\mathbf{u}_i')}{\partial \mathbf{u}_i}\frac{\partial \mathbf{S}_i\mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}\frac{1}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i} + \frac{vecut(\mathbf{u}_i\mathbf{u}_i)}{(\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i)^2}\frac{\partial \mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}.$$

$\partial vecut(\mathbf{u}_i\mathbf{u}_i')/\partial \mathbf{u}_i$ is calculated as in Block $\textcircled{3}$. $\partial \mathbf{S}_i\mathbf{w}_i/\partial vecut(\mathbf{S}_i)$ and $\partial \mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i/\partial vecut(\mathbf{S}_i)$ are obtained as in the calculation of Block $\textcircled{2}$.

Appendix B.2. $(\partial \mathbf{w}_l \tilde{\mathbf{R}}/\partial \mathbf{b})_{\mathbf{b}_0}$

Let the row vector $\mathbf{d} = \tilde{\mathbf{w}}_l \tilde{\mathbf{R}}$, so its element $d_j = \sum_{i=1}^{a} w_{il} \tilde{r}_{ij}$, where w_{il} denotes the l-th element of the column vector $\mathbf{w}_i$, and $\tilde{r}_{ij}$ is the element of $\tilde{\mathbf{R}}$ in the i-th row and the j-th column, $j = 1, \ldots, a$.

$$
\begin{aligned}
(\frac{\partial \mathbf{w}_l \tilde{\mathbf{R}}}{\partial \mathbf{b}})_{\mathbf{b}_0} &= \begin{pmatrix} \frac{\partial d_1}{\partial \mathbf{b}} \\ \frac{\partial d_2}{\partial \mathbf{b}} \\ \vdots \\ \frac{\partial d_a}{\partial \mathbf{b}} \end{pmatrix}_{\mathbf{b}_0}. \\
(\frac{\partial d_j}{\partial \mathbf{b}})_{\mathbf{b}_0} &= (\sum_{i=1}^{a} (\frac{\partial w_{il}}{\partial \mathbf{b}} \tilde{r}_{ij} + w_{il} \frac{\partial \tilde{r}_{ij}}{\partial \mathbf{b}}))_{\mathbf{b}_0},
\end{aligned}
$$

where Appendix B.2.1 and Appendix B.2.2 below show how to calculate $(\partial w_{il}/\partial \mathbf{b})_{\mathbf{b}_0}$, $\tilde{r}_{ij}$ and $(\partial \tilde{r}_{ij}/\partial \mathbf{b})_{\mathbf{b}_0}$.

Appendix B.2.1. $(\partial w_{il}/\partial \mathbf{b})_{\mathbf{b}_0}$

$(\partial w_{il}/\partial \mathbf{b})_{\mathbf{b}_0}$ can be taken as the l-th row vector from $\frac{\partial \mathbf{w}_i}{\partial \mathbf{b}_{i-1}} \frac{\partial \mathbf{b}_{i-1}}{\partial \mathbf{b}_{i-2}} \cdots \frac{\partial \mathbf{b}_2}{\partial \mathbf{b}_1}$, where $\frac{\partial \mathbf{w}_i}{\partial \mathbf{b}_{i-1}} = \begin{pmatrix} (1) & (2) \end{pmatrix}$ and $\frac{\partial \mathbf{b}_{i-1}}{\partial \mathbf{b}_{i-2}}$ are given in Appendix B.1.2.

Appendix B.2.2. $\tilde{r}_{ij}$ *and* $(\partial \tilde{r}_{ij}/\partial \mathbf{b})_{\mathbf{b}_0}$

Manne [10] gives that $\mathbf{R} = \mathbf{PW}$ is an $a \times a$ bidiagonal matrix whose only non-zero elements are r_{ii} and $r_{i(i+1)}$,

$$
\begin{cases}
r_{ii} = & 1 & i = 1, \ldots, a \\
r_{i(i+1)=} & \frac{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_{i+1}}{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i} & i = 1, \ldots, a-1 \\
r_{ij} = & 0 & \text{otherwise.}
\end{cases}
$$

$\tilde{\mathbf{R}} = (\mathbf{PW})^{-1}$ is an $a \times a$ upper triangular matrix, where the upper triangular elements are

$$
\begin{cases}
\tilde{r}_{ij} = 1 & i = j; \\
\tilde{r}_{ij} = -\tilde{r}_{i(j-1)} r_{(j-1)j} / r_{jj} & i \neq j.
\end{cases}
$$

As $\tilde{\mathbf{R}}$ is upper triangular, when $i \geq j$, $\partial \tilde{r}_{ij}/\partial \mathbf{b} = \mathbf{0}$, a row vector with $\frac{k(k+3)}{2}$ elements. Because $r_{jj} = 1$, when $i < j$, the derivative of $\tilde{r}_{ij}$ with respect to $\mathbf{b}_i$ can be calculated by an algorithm as follows

$$
\frac{\partial \tilde{r}_{ij}}{\partial \mathbf{b}_i} = -\{\frac{\partial \tilde{r}_{i(j-1)}}{\partial \mathbf{b}_i} r_{(j-1)j} + \tilde{r}_{i(j-1)} \frac{\partial r_{(j-1)j}}{\partial \mathbf{b}_i}\}.
$$

Then according to the chain rule,

$$
(\frac{\partial \tilde{r}_{ij}}{\partial \mathbf{b}})_{\mathbf{b}_0} = (\frac{\partial \tilde{r}_{ij}}{\partial \mathbf{b}_i} \frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}} \cdots \frac{\partial \mathbf{b}_2}{\partial \mathbf{b}_1})_{\mathbf{b}_0}.
$$

$\partial r_{(j-1)j}/\partial \mathbf{b}_i$ can be calculated in the form of $\partial r_{i(i+1)}/\partial \mathbf{b}_i$ as below. $r_{i(i+1)}$ can be further written as a function of $\mathbf{w}_i$ and $\mathbf{S}_i$,

$$
r_{i(i+1)} = \frac{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_{i+1}}{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i} = 1 - \frac{\mathbf{w}_i' \mathbf{S}_i \mathbf{S}_i \mathbf{w}_i}{\mathbf{w}_i' \mathbf{S}_i \mathbf{w}_i} q_i.
$$

Hence, the derivative can be written as

$$\begin{aligned}
\frac{\partial r_{i(i+1)}}{\partial \mathbf{b}_i} &= -\frac{\partial}{\partial \mathbf{b}_i}\frac{\mathbf{w}_i'\mathbf{S}_i\mathbf{S}_i\mathbf{w}_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}q_i \\
&= -\frac{\partial \mathbf{w}_i'\mathbf{S}_i\mathbf{S}_i\mathbf{w}_i}{\partial \mathbf{S}_i\mathbf{w}_i}\frac{\partial \mathbf{S}_i\mathbf{w}_i}{\partial \mathbf{b}_i}\frac{q_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i} - \frac{\mathbf{w}_i'\mathbf{S}_i\mathbf{S}_i\mathbf{w}_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}\frac{\partial q_i}{\partial \mathbf{b}_i} + \frac{\mathbf{w}_i'\mathbf{S}_i\mathbf{S}_i\mathbf{w}_i}{(\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i)^2}q_i\frac{\partial \mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}{\partial \mathbf{b}_i} \\
&= -2\mathbf{w}_i'\mathbf{S}_i\left(\mathbf{S}_i \quad \frac{\partial \mathbf{S}_i\mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}\right)\frac{q_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i} - \frac{\mathbf{w}_i'\mathbf{S}_i\mathbf{S}_i\mathbf{w}_i}{\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}\frac{\partial q_i}{\partial \mathbf{b}_i} + \frac{\mathbf{w}_i'\mathbf{S}_i\mathbf{S}_i\mathbf{w}_i}{(\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i)^2}q_i\left(2\mathbf{w}_i'\mathbf{S}_i \quad \frac{\partial \mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}\right),
\end{aligned}$$

where $\frac{\partial \mathbf{S}_i\mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}$ and $\frac{\partial \mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i}{\partial vecut(\mathbf{S}_i)}$ are calculated in Block (2) in Appendix B.1.2.

Appendix C. Computing $\hat{\beta}$ from b

In the bootstrap procedure we need to compute the PLS coefficient vector from $\mathbf{b}$. The procedure is as follows. $\mathbf{w}_1$ consists of the first k elements of $\mathbf{b}$ corresponding to $\mathbf{w}_1 = \mathbf{X}_c'\mathbf{y}_c$, and $\mathbf{S}_1$ is a square matrix built by the $(k+1)$-th to $\{k(k+3)/2\}$-th elements of $\mathbf{b}$. When $i \geq 2$, $\mathbf{w}_i = \mathbf{A}_{i-1}\mathbf{w}_{i-1}$, and $\mathbf{S}_i = \mathbf{A}_{i-1}\mathbf{S}_{i-1}\mathbf{A}_{i-1}'$. For $i = 1, \ldots, a$,

$$\begin{aligned}
\mathbf{A}_i &= \mathbf{I} - \mathbf{S}_i\mathbf{w}_i\mathbf{w}_i'/\mathbf{w}_i'\mathbf{S}_i\mathbf{w}_i. \\
\mathbf{v}_i &= \mathbf{w}_i/\sqrt{\mathbf{w}_i'\mathbf{w}_i}. \\
\mathbf{p}_i &= \mathbf{S}_i\mathbf{v}_i/\mathbf{v}_i'\mathbf{S}_i\mathbf{v}_i. \\
q_i &= \mathbf{w}_i'\mathbf{v}_i/\mathbf{v}_i'\mathbf{S}_i\mathbf{v}_i.
\end{aligned}$$

$\hat{\beta}^B = \mathbf{V}(\mathbf{P}'\mathbf{V})^{-1}\mathbf{q}$. The normalization $\mathbf{v}_i = \mathbf{w}_i/\sqrt{\mathbf{w}_i'\mathbf{w}_i}$ is used here because it makes the orthogonal scores algorithm more stable, though it does not change the estimated regression coefficients.

References

[1] L. Zhang, S. Garcia-Munoz, A comparison of different methods to estimate prediction uncertainty using partial least squares (PLS): a practitioner's perspective, Chemometrics and Intelligent Laboratory Systems 97 (2009) 152–158.

[2] M. C. Denham, Prediction intervals in partial least squares, Journal of Chemometrics 11 (1997) 39–52.

[3] S. Serneels, P. Lemberge, P. J. Van Espen, Calculation of PLS prediction intervals using efficient recursive relations for the jacobian matrix, Journal of Chemometrics 18 (2004) 76–80.

[4] A. Phatak, P. Reilly, A. Penlidis, The asymptotic variance of the univariate PLS estimator, Linear Algebra and its Applications 354 (2002) 245–253.

[5] R. Romera, Prediction intervals in partial least squares regression via a new local linearization approach, Chemometrics and Intelligent Laboratory Systems 103 (2010) 122–128.

[6] H. Martens, T. Næs, Multivariate Calibration, new ed., Wiley-Blackwell, 1991.

[7] I. S. Helland, On the structure of partial least squares regression, Communications in Statistics - Simulation and Computation 17 (1988) 581–607.

[8] B. Efron, R. J. Tibshirani, An Introduction to the Bootstrap, Chapman and Hall/CRC, 1994.

[9] J. R. Magnus, H. Neudecker, The commutation matrix: Some properties and applications, The Annals of Statistics 7 (1979) 381–394.

[10] R. Manne, Analysis of two partial-least-squares algorithms for multivariate calibration, Chemometrics and Intelligent Laboratory Systems 2 (1987) 187–197.

Figure 1: PLS estimated variances and actual squared prediction error versus V_L. $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1, \beta_0 = \beta_1 = 1, \beta_2 = 0$, $\sigma_\epsilon^2 = 0.25$. SPE: squared prediction error $(\dot{y}_p - \hat{\dot{y}}_p)^2$. Lin: $V_l + \sigma_\epsilon^2/n$. Linb: $V_B + \sigma_\epsilon^2/n$.

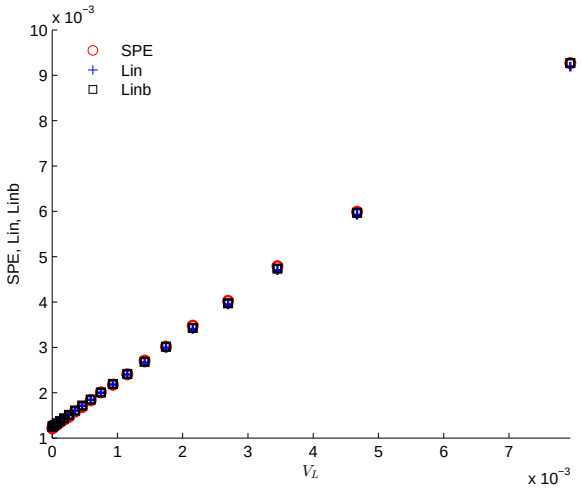

x 10^-3
SPE
Lin
Linb
SPE, Lin, Linb
V_L
x 10^-3

Figure 2: PLS $\hat{\boldsymbol{\beta}}$ against $\frac{\mathbf{b}}{n}$ when $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_0 = \beta_1 = 1$, $\beta_2 = 0$, $\sigma_\epsilon^2 = 0.25$.

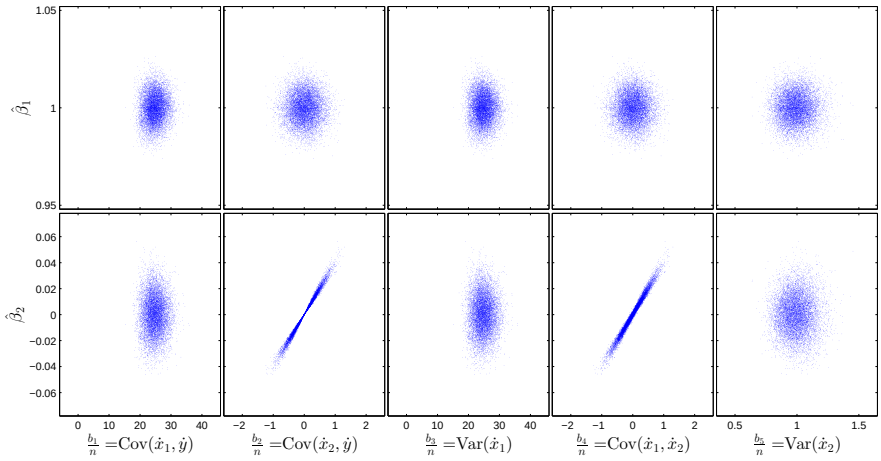
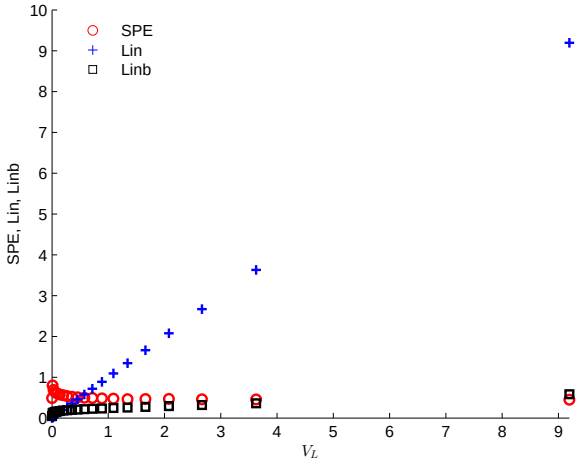
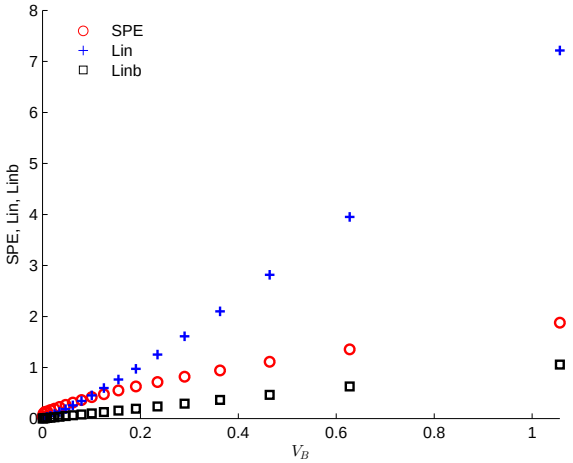

1.05
1
0.95
$\hat{\beta}_1$
0.06
0.04
0.02
0
−0.02
−0.04
−0.06
$\hat{\beta}_2$
0 10 20 30 40
−2 −1 0 1 2
0 10 20 30 40
−2 −1 0 1 2
0.5 1 1.5
$\frac{b_1}{n}$ =Cov($\dot{x}_1, \dot{y}$)
$\frac{b_2}{n}$ =Cov($\dot{x}_2, \dot{y}$)
$\frac{b_3}{n}$ =Var($\dot{x}_1$)
$\frac{b_4}{n}$ =Cov($\dot{x}_1, \dot{x}_2$)
$\frac{b_5}{n}$ =Var($\dot{x}_2$)

Figure 3: PLS estimated variances and actual squared prediction error versus (a) V_L and (b) V_B. $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_1 = 0$, $\beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$. SPE: squared prediction error $(\dot{y}_p - \hat{\dot{y}}_p)^2$. Lin: $V_L + \sigma_\epsilon^2/n$. Linb: $V_B + \sigma_\epsilon^2/n$.



(a)



(b)

Figure 4: PLS $\hat{\boldsymbol{\beta}}$ against $\frac{\mathbf{b}}{n}$ when $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1$, $\beta_1 = 0$, $\beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$.

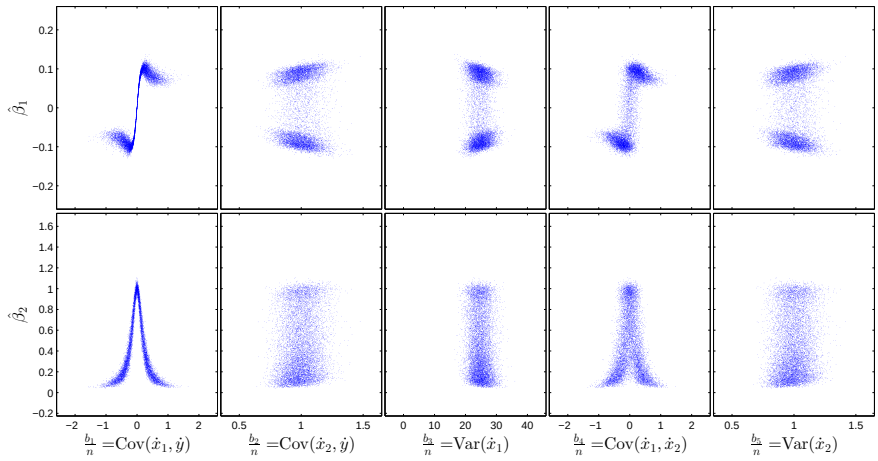

$\hat{\beta}_1$
$\hat{\beta}_2$
$\frac{b_1}{n}$ =Cov$(\dot{x}_1, \dot{y})$
$\frac{b_2}{n}$ =Cov$(\dot{x}_2, \dot{y})$
$\frac{b_3}{n}$ =Var$(\dot{x}_1)$
$\frac{b_4}{n}$ =Cov$(\dot{x}_1, \dot{x}_2)$
$\frac{b_5}{n}$ =Var$(\dot{x}_2)$

Figure 5: Adequacy of the linearization approximation in the case when $k = 2$, $a = 1$, $\sigma_1^2 = 25$, $\sigma_2^2 = 1, \beta_1 = 0, \beta_0 = \beta_2 = 1$, $\sigma_\epsilon^2 = 0.25$. $b_1 = -1.9897, b_2 = 215.3367, b_3 = 4691.2, b_4 = 4.123$, and $b_5 = 224.8093$.

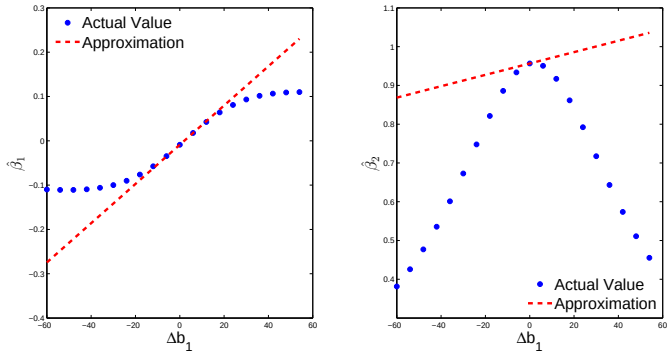

Actual Value
Approximation
$\hat{\beta}_1$
Δb_1
$\hat{\beta}_2$
Δb_1
Actual Value
Approximation

Figure 6: Adequacy of the linearization approximation in the case when $k = 3$, $a = 2$, $\sigma_1^2 = \sigma_2^2 = 25$, $\sigma_3^2 = 1$, $\beta_1 = \beta_2 = 1$, $\beta_3 = 0$, $\sigma_\epsilon^2 = 0.25$. $b_1 = 4145.5$, $b_2 = 4192.6$, $b_3 = -78.6$, $b_4 = 4686.5$, $b_5 = -483.1$, $b_6 = -21.8$, $b_7 = 4674.7$, $b_8 = -61.5$, and $b_9 = 171.3$.

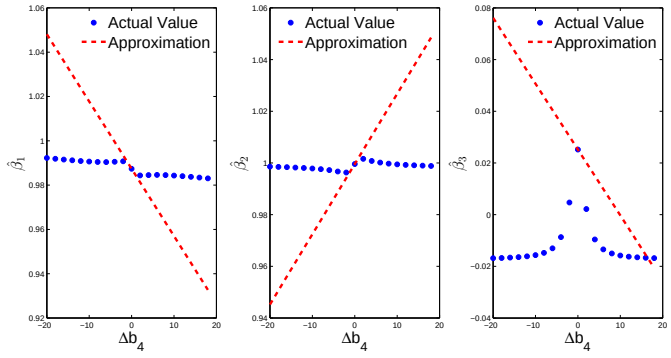
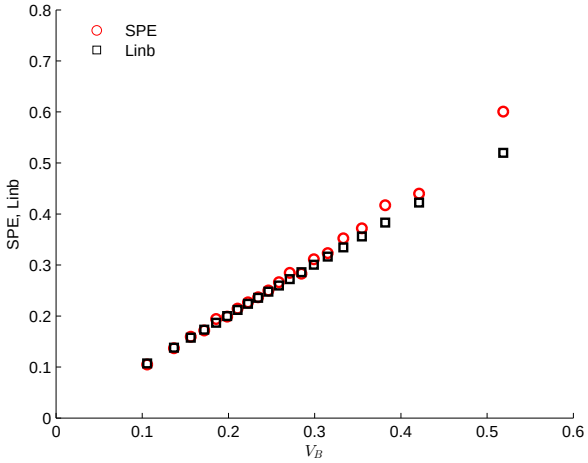

Actual Value
Approximation
$\hat{\beta}_1$
Δb_4
Actual Value
Approximation
$\hat{\beta}_2$
Δb_4
Actual Value
Approximation
$\hat{\beta}_3$
Δb_4

Figure 7: PLS estimated variances and actual squared prediction error versus V_B. $k = 24$, $a = 7$; $\sigma_1^2 = 64$, $\sigma_2^2 = 49$, $\sigma_3^2 = 36$, $\sigma_4^2 = 25$, $\sigma_5^2 = 16$, $\sigma_6^2 = 9$, $\sigma_7^2 = 4$, $\sigma_8^2 = \cdots = \sigma_{24}^2 = 1$; $\beta_0 = 1, \beta_1 = 8, \beta_2 = 7, \beta_3 = 6, \beta_4 = 5, \beta_5 = 4, \beta_6 = 3, \beta_7 = 2$, and $\beta_8 = \cdots = \beta_{24} = 0$, $\sigma_\epsilon^2 = 0.25$. SPE: squared prediction error $(\dot{y}_p - \hat{\dot{y}}_p)^2$. Linb: $V_B + \sigma_\epsilon^2/n$.


SPE
Linb
SPE, Linb
V_B
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8