PLS Leads to Different Algorithms for Factor Analysis and Regression

Tyler R. Holcomb

Manfred Morari [*]

Chemical Engineering 210-41
California Institute of Technology
Pasadena CA 91125

Keywords: biased regression, PLS, multivariable regression,
factor analysis, collinearity

CIT-CDS Technical Memo 93-003

May 12, 1993

Abstract

Two multivariable problems of general interest are factor analysis and regression. This paper examines partial least squares (PLS) as a tool for both problems. For single output data sets, the familiar PLS algorithm is applicable to both problems. For multiple output problems the familiar PLS algorithm [1, 2, 3] (called fact-PLS in this paper) is appropriate for factor analysis. However fact-PLS leads to algebraically-inconsistent results for regression problems. To address this issue, a new algebraically-consistent multivariable PLS algorithm, C-PLS, is developed. Unlike fact-PLS, C-PLS does not rely on iterative calculations. Another PLS approach, "one-at-a-time"

[*]Author to whom correspondence should be addressed: phone (818)356-4186, fax (818)568-8743, e-mail mm@imc.caltech.edu

PLS (OAT-PLS), is closely related to C-PLS; however OAT-PLS is also algebraically-inconsistent. A simulation study of these various PLS methods shows C-PLS to have the best estimation and prediction performance.

1 Introduction

This paper examines problems of the form

$$Y = XR + E, \tag{1}$$

where $X \in \Re^{n_s \times n_i}$ is a matrix of measured inputs, $Y \in \Re^{n_s \times n_o}$ is a matrix of measured outputs, $R \in \Re^{n_i \times n_o}$ is an unknown parameter matrix, and $E \in \Re^{n_s \times n_o}$ is an unobservable matrix of errors. For simplicity of development, further assume that the elements of E are zero-mean, independent, and homoscedastic random variables: $\mathcal{E}(E) = 0$, $\mathcal{E}\left(E^T E\right) = n_s \sigma_e^2 I$ and $\mathcal{E}\left(E E^T\right) = n_o \sigma_e^2 I$. The independence and homoscedasticity assumptions can be readily met using a suitable rescaling of the data, as in generalized least squares [4]. Throughout this paper the variables that comprise X will be called the "inputs" and the variables that comprise Y will be called the "outputs." This is merely a convenient nomenclature; these variables can equivalently be called "explanatory variables" and the "dependent variables."

For such data, two problems commonly addressed are: (1) what are the lower dimensional subspaces of $\Re^{n_i}$ and $\Re^{n_o}$ that describe the "relevant" variances of the data, and (2) what is a "good" approximation for the unobservable R? We refer to the former as the factor analysis problem and the latter as the regression problem. The regression problem itself has two interesting variants: the estimation problem, wherein one determines a $\tilde{B}$ that is "close" to R, and the prediction problem, wherein one determines a $\tilde{B}$ that is "good" for predicting futures Y's from future X's. Partial Least Squares (PLS) [1] is useful for all of these problems. However, past investigations have not always carefully distinguished between the factor analysis and regression problems. This paper examines PLS for application to both problems; in doing so, a new PLS algorithm is developed that is more appropriate for multivariable regression.

2 Factor Analysis

In factor analysis one is commonly seeking to model data using equation 1 and the assumption that the "important" variances are described by w_i and c_i, a set of "loading vectors" in the input space and output space, respectively. That is

$$X = t_1 w_1^T + \ldots + t_{n_d} w_{n_d}^T + E_x \quad \text{and} \tag{2}$$

$$Y = t_1 c_1^T + \ldots + t_{n_d} c_{n_d}^T + E_y \tag{3}$$

where $w_i \in \Re^{n_i}$ are the input loading vectors, $c_i \in \Re^{n_o}$ are the output loading vectors, $t_i \in \Re^{n_s}$ are the "scores vectors," and $E_x \in \Re^{n_s \times n_i}$ and $E_y \in \Re^{n_s \times n_o}$ the "residuals" or "unaccounted variations." Each element of a score vector represents the projection of a data sample onto the respective loading vector. Such a description can be useful for interpreting data sets [5] or compactly describing key features of data sets [6]. For chemometrics, a particularly successful algorithm for computing the c_i and w_i is PLS. See Helland [7] for statistical theorems relating PLS to other factor analysis methods. As shown by Höskuldson [2], PLS operates by finding the c and w that maximize the covariance between the input and the output. That is,

$$\{w^{opt}, c^{opt}\} = \arg \max_{||w||=1, ||c||=1} < Yc, Xw >^2 \tag{4}$$

where $< \cdot, \cdot >$ is the inner product.

Such an approach is appealing when one wants to account for common variances *between* the input and output, but one is not interested in describing variances *among* the inputs or the outputs. For illustration, consider a simple example where the inputs are described by two uncorrelated factors, but only one factor affects the output:

$$X = t_1 w_1^T + t_2 w_2^T \tag{5}$$

$$Y = t_1 c_1^T, \text{ and} \tag{6}$$

$$< t_1, t_2 > = 0. \tag{7}$$

Applying PLS, one computes

$$\{w^{opt}, c^{opt}\} = \arg \max_{||w||=1, ||c||=1} < Yc, Xw >^2 \tag{8}$$

$$= \arg \max_{||w||=1, ||c||=1} \left(w^T (t_1 w_1^T + t_2 w_2^T)^T t_1 c_1^T c \right)^2 \tag{9}$$

$$= \arg \max_{||w||=1,||c||=1} (w^T w_1 t_1^T t_1 c_1^T c)^2 \tag{10}$$

$$= \left\{ \frac{w_1}{||w_1||}, \frac{c_1}{||c_1||} \right\} \tag{11}$$

In this example PLS correctly identified the common factor between the inputs and outputs, but disregarded the input factor that did not affect the output (w_2). This is in contrast to the Principal Components Analysis (PCA), which strives to capture the greatest total variation. PCA embraces a variety of methods; two of the more prevalent PCA methods are discussed here. As commonly stated in the chemometrics literature, PCA proceeds by using the eigenvector of greatest eigenvalue of $X^T X$ as the first loading vector [8]. For the above example, PCA would fail to identify the common factor of the input and the output (w_1) if $t_1^T t_1 < t_2^T t_2$. Since this inequality is unrelated to which factor is " relevant" for describing Y, this form of PCA is seen to be less desirable than PLS. In the econometrics literature, a different form of PCA is used in which the "relevance" of the factors for describing variance common to both inputs and outputs is considered [9, 10]. For the above problem, the PCA loading vectors would be found from

$$\{w^{opt}, c^{opt}\} = \arg \max_{||w||=1,||c||=1} \begin{bmatrix} w \\ c \end{bmatrix}^T \begin{bmatrix} X^T X & X^T Y \\ Y^T X & Y^T Y \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} \tag{12}$$

$$= \arg \max_{||w||=1,||c||=1} \begin{bmatrix} w^T \\ c^T \end{bmatrix} \begin{bmatrix} w_1 w_1^T t_1^T t_1 + w_2 w_2^T t_2^T t_2 & w_1 c_1^T t_1^T t_1 \\ c_1 w_1^T t_1^T t_1 & c_1 c_1^T t_1^T t_1 \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} \tag{13}$$

$$= \arg \max_{||w||=1,||c||=1} \begin{array}{c} (w^T w_1)^2 t_1^T t_1 + (w^T w_2)^2 t_2^T t_2 + 2c^T c_1 w_1^T w t_1^T t_1 \\ +(c^T c_1)^2 t_1^T t_1 \end{array} . \tag{14}$$

In this example, the first input factor, w^{opt}, will clearly be affected by the additional factor w_2 that does not account for any of the variance in the output. Notice that PLS is equally facile at removing "irrelevant" factors in both the inputs and the outputs.

After the PLS factors ("loading vectors") have been determined, their effect is removed from X and Y and the process is repeated. The full algorithm then becomes the familiar multivariable PLS algorithm. As described by Höskuldsson [2], the algorithm is:

fact-PLS: multivariable factor analysis

$$X_1 = X \tag{15}$$

```
$Y_1 = Y$                                                        (16)
Do $i = 1$ to $n_d$
    Let $c_i$ be the eigenvector of maximum eigenvalue in
        $\lambda_{c_i} c_i = Y_i^T X_i X_i^T Y_i c_i$                (17)
    $c_i = \frac{c_i}{||c_i||}$                                      (18)
    $w_i = \frac{X_i^T Y_i c_i}{||X_i^T Y_i c_i||}$                  (19)
    $t_i = X_i w_i$                                                  (20)
    $\beta_i = \frac{t_i^T Y_i c_i}{t_i^T t_i}$                      (21)
    $X_{i+1} = (I - \frac{t_i t_i^T}{t_i^T t_i}) X_i$                (22)
    $Y_{i+1} = Y_i - \beta_i t_i {c_i}^T$                            (23)
End Do
```

3 Regression

PLS has also been found effective for regression problems. We first review single output regression problems to clarify several issues. Using these insights, we investigate multiple output regression problems, discover an inconsistency in fact-PLS, and propose an improved PLS algorithm.

3.1 Scalar output problems

Regression problems typically involve minimizing an objective function that measures how well a given regressor either estimates the "true regressor" or predicts outputs. For scalar output problems

$$y = Xr + e, \tag{24}$$

where $y \in \Re^{n_s}$, $e \in \Re^{n_s}$, and $r \in \Re^{n_i}$, the most common regression objective function is $||y - Xb||^2$ for $b \in \Re^{n_i}$ from which

$$\tilde{b} = \arg \min_{b \in \Re^{n_i}} ||y - Xb||^2. \tag{25}$$

A regressor found by such means can be very sensitive to errors if collinearities are present in the input data [5]. Biased regressors can overcome this variance problem by requiring that $\tilde{b}$ be formed from a linear combination of the elements of a set of pre-specified vectors in $\Re^{n_i}$. PLS naturally provides the needed vectors from the loading vectors, w_i. In the scalar output case both regression and factor analysis lead to the scalar PLS algorithm:

scalar-PLS: scalar output problems

```
$W_0 =$ $[0|\cdots|0]^T, \quad W_0 \in \Re^{n_i}$    (26)
DO $i =$ $1, n_d$
    $v = (I - W_{i-1}{W_{i-1}}^T)(X^T X)^{i-1} X^T y$    (27)
    $w_i = \frac{v}{||v||}$    (28)
    $W_i = [w_1|w_2|\cdots|w_i]$    (29)
END DO.
```

This algorithm computes the loading vectors for factor analysis. The loading vectors are used directly in the regression problem. Let $\mathcal{S}$ be the set of all possible linear combinations of $\{w_1, \ldots, w_{n_d}\}$ and $W = W_{n_d}$. Then one computes the biased regressor $\tilde{b}$ as follows:

$$\tilde{b} = \arg\min_{b \in \mathcal{S}} ||y - Xb||^2 \tag{30}$$

$$= \arg \min_{b \in \text{Range}(W)} ||y - Xb||^2 \tag{31}$$

$$= W(W^T X^T X W)^{-1} W^T X^T y. \tag{32}$$

3.2 Multiple output problems

For multiple output problems like equation 1, the ordinary least-squares regression is

$$\tilde{B} = \arg \min_{B \in \Re^{n_i \times n_o}} ||Y - XB||_F^2. \tag{33}$$

As in the scalar case, this regressor can be unreliable if collinearities are present in the input data. One can use the factors from the fact-PLS algorithm to treat these collinearities. First, one must build matrices with orthonormal columns from the input and output loading vectors. That is let $W = [w_1^{opt}|\ldots|w_{n_d}^{opt}]$ and let the orthonormal columns of C be such that $\text{Range}(C) = \text{Span}(\{c_1^{opt}, c_2^{opt}, \ldots, c_{n_d}^{opt}\})$. Moreover, define $C^\perp$ such that $[C|C^\perp]^T\ [C|C^\perp] = I$ and $n_c = \text{Rank}(C)$. Then one requires that all allowable regressors be of the form $B = WVC^T$ where $V \in \Re^{n_d \times n_c}$. Thus, the search for the "optimal" regressor is constrained to considering subspaces spanned by the PLS loading vectors.

One determines V^{opt}, the V to be used for building $\tilde{B}$, using the least-squares objective function:

$$\begin{aligned}
V^{opt} &= \arg \min_{V \in \Re^{n_d \times n_c}} \|Y - XWVC^T\|_F^2 && (34)\\
&= \arg \min_{V \in \Re^{n_d \times n_c}} \| \, Y[C \mid C^{\perp}] - XWVC^T[C \mid C^{\perp}] \, \|_F^2 && (35)\\
&= \arg \min_{V \in \Re^{n_d \times n_c}} \| \, [YC \mid YC^{\perp}] - [XWV \mid 0] \, \|_F^2 && (36)\\
&= (W^TX^TXW)^{-1}W^TX^TYC, && (37)
\end{aligned}$$

from which the fact-PLS regressor is $\tilde{B} = WV^{opt}C^T = W(W^TX^TXW)^{-1}W^TX^TYCC^T$.

Another approach to the multiple output regression problem is to recast the multiple output problem as a scalar output problem and use the scalar-PLS regression method. This approach proceeds by simple reorganization of the data. First, the input data matrix and the output data matrix need to be suitably redefined. Normally,

$$Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_{n_s}^T \end{bmatrix}, \qquad X = \begin{bmatrix} x_1^T \\ \vdots \\ x_{n_s}^T \end{bmatrix}, \qquad \text{and } Y = XR + E. \qquad (38)$$

To conform to equation 24, Y, E, and R must be transformed into column vectors. Considering the columns of $R = [r_1| \cdots |r_{n_o}]$, let

$$y_{stacked} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_s} \end{bmatrix} \qquad \text{and } r_{stacked} = \begin{bmatrix} r_1 \\ \vdots \\ r_{n_o} \end{bmatrix} \qquad (39)$$

where $y_{stacked} \in \Re^{n_s n_o}$ and $r_{stacked} \in \Re^{n_i n_o}$. Create $e_{stacked}$ from E is the same manner that $y_{stacked}$ was created from Y. Moreover, build $X_{stacked} \in \Re^{n_s n_o \times n_i n_o}$ such that

$$X_{stacked} = \overbrace{\left[\begin{array}{c} n_o \left\{\begin{array}{cccc} x_1^T & 0 & \dots & 0 \\ 0 & x_1^T & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & x_1^T \end{array}\right. \\ \begin{array}{cccc} x_2^T & 0 & \dots & 0 \\ 0 & x_2^T & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & x_2^T \end{array} \\ \vdots \\ \begin{array}{cccc} x_{n_s}^T & 0 & \dots & 0 \\ 0 & x_{n_s}^T & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & x_{n_s}^T \end{array} \end{array}\right]}^{n_i \times n_o}. \tag{40}$$

Then equation 1 data can be described by

$$y_{stacked} = X_{stacked}\, r_{stacked} + e_{stacked}. \tag{41}$$

Equation 41 is consistent with equation 24, so scalar-PLS can be used. After the process is completed, $\tilde{b}_{stacked}$ must be "unstacked" in the reverse manner from which $r_{stacked}$ was built in equation 39.

Interestingly, fact-PLS is "algebraically-inconsistent;" if fact-PLS is used for multiple output data, and then applied to the same data formatted as a scalar output problem, different regressors can result. However, the "stacking method" always starts by recasting the data into the scalar output form, so the "stacking" method is "algebraically-consistent" by construction. The "stacking method" will be called C-PLS for the remainder of the paper. Why is fact-PLS inconsistent? Recall how the biased regressor is formed in the scalar problem: $\tilde{b}$ is formed from a linear combination of elements in $\Re^{n_i}$, the space from which r is drawn. Likewise, one would like to produce "factor matrices" in $\Re^{n_i \times n_o}$, and then restrict $\tilde{B}$ to be a linear combination of these "factor matrices." Converting to a scalar output description and using scalar-PLS does precisely this, while fact-PLS does not.

To see the difference more clearly, consider the first "factor matrix." The first "factor matrix" to be used for a biased regressor should have $n_o n_i - 1$ degrees of freedom. In the scalar case the first factor was drawn from $\Re^{n_i}$, and thus had $n_i - 1$ degrees of freedom. (Both lose one degree of freedom to normalization.) C-PLS does in fact have $n_o n_i - 1$ degrees of freedom for the first factor. In contrast, consider the first "factor matrix" produced by fact-PLS: $w_1 c_1^T$. While $w_1 c_1^T \in \Re^{n_i \times n_o}$, this "factor matrix" possesses only $n_o + n_i - 2$ degrees of freedom. Thus fact-PLS invokes constraints relevant for the factor analysis problem that are not necessarily beneficial for the multivariable regression problem.

In addition to being more appropriate for regression problems, C-PLS holds other advantages. As mentioned previously, an iterative eigenvector calculation has been dispatched in favor of a matrix multiplication. Thus C-PLS may lead to faster, simpler computer codes than fact-PLS. However as noted by Manne [11], practitioners are more apt to be interested in PLS for its analytical effectiveness than for its computational efficacy, so this point is not pursued. A more direct benefit is theoretical simplicity. For example, if one denotes the "factor matrices" before orthogonalization and normalization by $F_1, \ldots, F_{n_d}$, then $F_i = (X^T X)^{i-1} X^T Y$ – a direct generalization of equation 27. One can use this fact to build a different objective function for PLS that relates to statistical significance, to develop statistical procedures as alternative to cross-validation, and to compare PLS to other biased regression methods [12]. Also, since the multiple output problem is now described using the scalar output model, the theoretical framework developed for the scalar output problem [3, 7, 13] can now be applied to the multiple output problem without modification.

3.3 Relation to "one-at-a-time" PLS

Some practitioners prefer "one-at-a-time" PLS (OAT-PLS) to fact-PLS. In OAT-PLS, one considers each of the n_o outputs as being an independent problem. Thus, one uses scalar-PLS n_o times, once for each output. This approach is closely related to C-PLS. Let $Y = [z_1 | z_2 | \ldots | z_{n_o}]$. Then for C-PLS, the first loading vector *before* normalization (the result of equation 27 for $i = 1$) is

$$v = \begin{bmatrix} X^T z_1 \\ X^T z_2 \\ \vdots \\ X^T z_{n_o} \end{bmatrix}. \tag{42}$$

Notice that this vector can also be described as the stacking of the single output unnormalized loading vectors; at this point, OAT-PLS and C-PLS are identical. In C-PLS normalization is achieved via scalar multiplication:

$$w_1^{\mathrm{C-PLS}} = \frac{v}{||v||}. \tag{43}$$

With OAT-PLS each piece of the loading vector is normalized separately:

$$w_1^{\mathrm{OAT-PLS}} = \begin{bmatrix} \frac{1}{||X^T z_1||} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{||X^T z_1||} & & & \\ & & & \frac{1}{||X^T z_2||} & & \\ & & & & \ddots & \\ & & & & & \frac{1}{||X^T z_2||} \\ & & & & & & \ddots \end{bmatrix} \begin{bmatrix} X^T z_1 \\ X^T z_2 \\ \vdots \\ X^T z_{n_o} \end{bmatrix} \tag{44}$$

where all off-diagonal elements are zero. Notice that the normalization performed in OAT-PLS alters the direction of v, while C-PLS does not. Thus, the OAT-PLS, like fact-PLS, is algebraically inconsistent in that OAT-PLS can yield a different answer than one gets "stacking" the vector output problem into a scalar output problem. OAT-PLS also faces another problem: why just n_o single output problems? Consider a four output problem ($n_o = 4$). Then one could also consider the four different "three-at-a-time/one-at-a-time" problems and the three different "two-at-a-time/two-at-a-time" problems. To insure that one has chosen the correct grouping of subproblems, one must investigate all of the possible combinations — a considerable effort in computation for an uncertain improvement in performance.

4 Simulation Example

This section presents a comparison between the two multivariable regression methods discussed in the paper. In this study, the examples are simulation studies using purely synthetic data. The data are not claimed to correspond to any particular "real world" process; rather, the data were generated to conform to the model assumptions and to illustrate the relative effectiveness of various methods for problems that satisfy the model assumptions. The "real world" successes of PLS [5, 14, 15] are suggested as evidence of the practical utility of PLS. The regression methods investigated were

- ordinary least squares (OLS),
- the previous multivariable PLS method (fact-PLS),
- "one-at-a-time" PLS (OAT-PLS), and
- the new multivariable PLS method (C-PLS).

All examples had ten inputs and four outputs ($n_i = 10$ and $n_o = 4$). One thousand distinct examples were examined to mitigate sampling effects in the numerical results. Each example was generated by the method presented in appendix B. Since both input variances and the values of the regression parameters varied over five orders of magnitude and since there were typically large variances in the input data that had little effect on the output, this exploration shed light on the relative strengths and weaknesses of the methods for a class of problems that has historically bedeviled OLS.

Two measures were employed to evaluate regressor performance. Since the examples were synthetic, R was known and the estimation error could be computed for each example. The measure was

$$RMS_{\text{MSE}} = \sqrt{\frac{\text{Tr}(\ (\tilde{B} - R)(\tilde{B} - R)^T\ )}{\text{Tr}(RR^T)}}. \tag{45}$$

The $\text{Tr}(RR^T)$ term was included to produce a relative error and allow averaging over all one thousand examples.

The second measure was computed based on the PRESS. For each example an additional one hundred samples (X_{new}, Y_{new}) were generated from the identical distribution as the training data, but the Y_{new} were not corrupted by error ($E_{new} = 0$). Then

$$RMS_{\text{PRESS}} = \sqrt{\frac{\text{Tr}(\ (X_{new}\tilde{B} - Y_{new})^T(X_{new}\tilde{B} - Y_{new})\ )}{400}}. \tag{46}$$

method	$\overline{RMS_{\text{MSE}}}$	$\overline{\text{rank}}$	$\overline{RMS_{\text{PRESS}}}$	$\overline{\text{rank}}$
OLS	520	4.0	0.215	4.0
OAT − PLS	6.0	1.8	0.124	2.3
fact − PLS	5.9	1.7	0.122	1.6
C − PLS	1.0	1.4	0.119	1.6

Table 1: Comparison of PLS methods over 1,000 examples of synthetic data.

Since the data were generated with the constraint

$$\sqrt{\frac{\text{Tr}(Y_{new}^T Y_{new})}{400}} = 1 \tag{47}$$

the RMS_{PRESS} was averaged over the examples without normalization. Note that $n_s \times n_o = 400$ for the test set. Also, for each example the rank (relative performance) of each estimator was recorded: rank = 1 if no other regressor did better for that example, rank = 2 if one other regressor did better, and rank = 3 if two other regressors did better. If two regressors were within 0.1% of each other, they were given the same rank. The average rank with respect to both MSE and PRESS was computed. For all examples, thirty samples were available for training ($n_s = 30$). Ten-way (three-out) cross-validation was used to determine n_d.

The results are shown in Table 1. As discussed in subsection 3.2, fact-PLS, labors under more constraints than C-PLS when determining the best "matrix factors" in $\Re^{n_i \times n_o}$. This limitation is reflected in the MSE results; the $\overline{RMS_{\text{MSE}}}$ for C-PLS was one-fifth that of the $\overline{RMS_{\text{MSE}}}$ for PLS. C-PLS also slightly outperformed fact-PLS as measured by $\overline{RMS_{\text{PRESS}}}$. OAT-PLS, which also uses more constraints than C-PLS when determining the best "matrix factors," performed similar (though slightly inferior) to fact-PLS. Fact-PLS, OAT-PLS, and C-PLS required similar computational effort; however, precise comparisons of computational effort were difficult due to variations in software implementation.

5 Conclusion

This paper examined PLS for factor analysis and regression problems, for both scalar data and vector output data. The multivariable fact-PLS algorithm was seen to be an effective factor analysis method when one wanted to identify covariances between the inputs and outputs while ignoring common variances among the inputs and among the outputs. For scalar regression problems, scalar-PLS was seen to provide useful factors for both factor analysis and biased regression. Likewise, fact-PLS was shown to be applicable to multivariable regression. However, fact-PLS incorporates constraints from the factor analysis problem that are not necessarily relevant for the regression problem. Based on this observation, a new PLS approach to multivariable regression (C-PLS) was put forward wherein the multiple output problem is recast as a scalar output problem and scalar-PLS is used. "One-a-at-a-time" PLS is closely related to C-PLS but still uses additional constraints that are not necessarily relevant for regression. C-PLS was shown to have better estimation performance over one thousand simulated examples. C-PLS opens new avenues for theoretical analysis of multivariable PLS methods and allows results already developed for scalar output problems to be applied directly to vector output problems.

Acknowledgments *Partial support of this research through the Department of Energy, Office of Basic Energy Sciences is gratefully acknowledged.*

References

[1] S. Wold, A. Ruhe, H. Wold, and W. Dunn. The collinearity problem in linear regression: The partial least squares approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, 5(3):753–743, September 1984.

[2] Agnar Höskuldsson. Pls regression methods. *Journal of Chemometrics*, 2:211–228, 1988.

[3] Inge S. Helland. On the structure of partial least squares. *Communications in Statistics - Simulation*, 17(2):1581–607, 1988.

[4] Henri Theil. *Principles of Econometrics.* Wiley, 1971.

[5] Harald Martens and Tormad Næs. *Multivariate Calibration.* Wiley, 1989.

[6] J.V. Kresta, J.F. MacGregor, and T. E. Marlin. Multivariate statistical monitoring of process operating performance. *Canadian Journal of Chemical Engineering*, 69(1):35–47, 1991.

[7] Inge S. Helland. Partial least squares and statistical models. *Scandinavian Journal of Statistics*, 17:97–114, 1990.

[8] Paul Geladi and Bruce Kowalski. Partial least squares regression: A tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

[9] T.W. Anderson. The 1982 wald memorial lectures: estimating linear statistical relationships. *The Annals of Statistics*, 12(1):1–45, 1984.

[10] R. Carter Hill, Thomas B. Fomby, and S.R. Johnson. Component selection norms for principal components regression. *Communications in Statistics A: Theory and Methods*, A6(4):309–334, 1977.

[11] Rolf Manne. Analysis of two partial-least-squares algorithms for multivariate calibration. *Journal of Chemometrics*, 2:187–197, 1987.

[12] Tyler R. Holcomb, Håkan Hjalmarsson, and Manfred Morari. Significance regression: A statistical approach to biased regression and partial least squares. CDS Technical Memo CIT-CDS 93-002, California Institute of Technology, Pasadena, CA 91125, February 1993.

[13] Avraham Lorber, Lawrence Wangen, and Bruce Kowalski. A theoretical foundation for the pls algorithm. *Journal of Chemometrics*, 5:19–31, 1987.

[14] Thor Mejdell. *Estimators for Product Composition in Distillation Columns.* PhD thesis, University of Trondheim, The Norwegian Institute of Technology, November 1990.

[15] N. L. Ricker. The use of biased least-squares estimators for parameters in discrete-time pulse response models. *Industrial and Engineering Chemical Research*, 27:343–350, 1988.

[16] Cleve Moler, John Little, Steve Bangert, and Steve Kleinman. *MATLAB User's Guide*. The MathWorks, 1990.

[17] Ildiko E. Frank and Jerome H. Friedman. A statistical review of some chemometrics regression tools. Technical report, Dept. of Statistics, Stanford University, Stanford, CA 94305, 1992.

A Nomenclature

In general, capital letters represent matrices, lower case letters represent column vectors, and Greek letters represent scalars. Estimates are denoted by a tilde, "˜". The dimensions of matrices are denote by subscripted n's.

dimensional descriptors

variable	description
n_d	is the number of factors to be generated.
n_i	is the number of inputs.
n_o	is the number of outputs.
n_s	is the number of samples.

some vectors and matrices

variable	dimension	description
$\tilde{b}$	$n_i \times 1$	is the biased estimate of r. See equation 32.
$\tilde{B}$	$n_i \times n_o$	is the biased estimate of R.
I	as appropriate	is the identity matrix.
r	$n_i \times 1$	is the "true" regression vector. See equation 24.
$\tilde{r}$	$n_i \times 1$	is the minimum-variance unbiased estimate of r.
R	$n_i \times n_o$	is the "true" regression matrix. See equation 1.
W	$n_i \times n_w$	is the matrix whose range defines the search space for $\tilde{b}$. See equation 32.
x_j	$n_i \times 1$	is the jth input data sample.
X	$n_s \times n_i$	input data; each row corresponds to one input sample. Thus, $X^T = [x_1 \;\; x_2 \;\; \ldots \;\; x_{n_s}]$.
y_i	$n_o \times 1$	is the ith output data sample.
Y	$n_s \times n_o$	is the output data. Each row corresponds to one output sample. Thus, $Y^T = [y_1 \;\; y_2 \;\; \ldots \;\; y_{n_s}]$.

operators

operator	description
$\|\| \cdot \|\|$	is the Euclidean norm. $\|\|a\|\| = \sqrt{< a, a >}$.
$\|\| \cdot \|\|_F$	is the Frobenius (matrix Euclidean) norm. $\|\|a\|\|_F = \sqrt{\sum_{i,j} a_{i,j}^2}$, where $a_{i,j}$ are the components of A.
$[W \ \|V]$	is the matrix formed by placing W and V side-by-side.
$< \cdot, \cdot >$	is the inner product. For vectors a and b, $< a, b > = a^T b$.
$\mathcal{E}(\cdot)$	is the expectation.
Range($\cdot$)	is the range; the span of the column vectors of a matrix.
Span($\cdot$)	is the space defined by all linear combinations of the elements in a set.
Tr($\cdot$)	is the trace, the sum of the diagonal elements of a matrix.

B Generation of Data for Simulation Examples

The simulation exploration was conducted using Matlab [16]. The two Matlab M-files used to generate the data are described below. The parameters used with these routines were: `n_train = 30`, `n_test = 100`, `d = 10`, `o = 4`, `d_ind = 3`, `max_exp = 5`, `min_exp = 0`, and `noise = 0.3`.

The generation routine is specifically designed to produce difficult examples. The "true" regression vectors (columns of R) are drawn from a spherically symmetric distribution about the origin (all directions are equally probable). However, the length of these vectors varies over 5 orders of magnitude. Thus, from a Bayesian viewpoint, the prior distribution for the regression vector is not particularly informative. The X are chosen independently of the R and the singular values (the square roots of the eigenvalues of $X^T X$) also vary over 5 orders of magnitude. Thus, there will be large variances in the X data which do not lie in any of the directions of the columns of R and therefore have little effect on the output. This will trouble principal component regression methods that proceed by examining directions in the order of the value of their singular values (principal components). Lastly, three of the inputs vary independently of all other input variables, but the remaining seven are correlated. This covariance structure can cause difficulties for both variable subset selection methods such as step-wise regression [17] and for scaling methods such as auto-scaling (using "standardized variables") that weight the input data solely on the variance of each individual input variable.

B.1 Routine to generate random regression problems

```
function [X,y,Xt,yt,b] =gen_dat2(n_train,n_test,d,o,d_ind
                                   ,max_exp,min_exp,noise)

% this function generates data for linear regression problems
%
%
%  n_train  is the number of samples to be the training set
%  n_test   is the number of samples to be the testing set
%  d        is the number of inputs
%  o        is the number of outputs
%  d_ind    is the number of inputs NOT rotated
%            and thus "independent"
%  max_exp  the largest order of magnitude contemplated
%  min_exp  the smallest order of magnitude contemplated
%            used for scaling the input data and
%            generating the regression vector
%  noise    std deviation of the normal additive noise
%
%
%  X         is the input training data
%  Xt        is the input testing data
%  y         is the output (noise corrupted) training data
%  yt        is the output (not noise corrupted) testing data



scale = diag(abs(scaled_rand(max_exp,min_exp,d)));
% these b's are for the same direction as singular vectors
```

```
for i=1:o
    b(:,i) = scaled_rand(max_exp,min_exp,d);
end

% need to build random orthogonal matrix
% only rotate d - d_ind columns; let the rest be
% 'approx' independent

d_rot =  d - d_ind;
if  d_ind == d
  v = eye(d);
else
  rand('uniform')
  v = rand (d_rot);
  [u,s,v] = svd(v);
  if d_rot == d
      v = u*v;
  else
      v = [ eye(d_ind), zeros(d_ind,d_rot); zeros(d_rot,d_ind), u*v];
  end

end

% use v as an additional rotation on the data and regression vector

rand('normal')
X =  rand(n_train,d) * scale * v;
Xt =  rand(n_test,d) * scale * v;
b = v'*b;
yt = Xt*b;
%desire RMS of null predictor to be 1
rms = sqrt(trace(yt'*yt)/ (n_test * o) ) ;
```

```
b=b/rms;
yt = Xt*b;
y = X*b + rand(n_train,o)*noise;
```

B.2 Routine to generate "exponential" random numbers

```
function vect =  scaled_rand(u,l,d)


% this function generates a vector of random numbers that are
% 'exponentially' distributed;  that is, the probability of
%  a number having any given  order of magnitude within
%  the valid range is roughly equal
%
%   u     lowest order of magnitude allowed
%   l     highest order of magnitude allowed
%   d     is the dimension of the vector generated
%
%  10^l < number < 10^u
%


rand('uniform');


for i = 1:d
  vect(i) = 10^ ( (u - l) * rand(1,1)  + l);
end


vect = vect';
```