Research Article

JOURNAL OF CHEMOMETRICS

Received: 14 November 2008, Revised: 13 May 2009, Accepted: 18 May 2009, Published online in Wiley InterScience: 17 July 2009

(www.interscience.wiley.com) DOI: 10.1002/cem.1249

A nonlinear partial least squares algorithm using quadratic fuzzy inference system

Araby I. Abdel-Rahman[a] and Gino J. Lim[a]*

We introduce a new nonlinear partial least squares algorithm 'Quadratic Fuzzy PLS (QFPLS)' that combines the outer linear Partial Least Squares (PLS) framework and the Takagi–Sugeno–Kang (TSK) fuzzy inference system. The inner relation between the input and the output PLS score vectors is modeled by a quadratic TSK fuzzy inference system. The performance of the proposed QFPLS method is tested and compared against four other well-known partial least squares methods (Linear PLS (LPLS), Quadratic PLS (QPLS), Linear Fuzzy PLS (LFPLS), and Neural Network PLS (NNPLS)) on various different types of randomly generated test data. QFPLS outperformed competitors based on two comparison measures: the output variables cumulative per cent variance captured by the PLS latent variables and the root mean-square error of prediction (RMSEP). Copyright © 2009 John Wiley & Sons, Ltd.

Keywords: partial least squares; fuzzy inference system; regression analysis; multivariate statistical analysis

1. INTRODUCTION

Statistical modeling and data analysis techniques have been extensively used these days to model experimental and historical data. Characteristics such as high dimensionality and co-linearity in the data are the main challenges to the multivariate statistical data analysis. Partial least squares (PLS) technique is a multivariate regression and data analysis method that addresses these problems [1]. PLS projects the original variables into latent variables and reduces the dimensionality of the data while considering the correlation that exists among the attributes. Therefore, it is easier to predict the output variables in the latent variables domain than in the original variables domain. Originally, the PLS method was introduced as a linear regression technique. This linearity assumption between inputs **X** and outputs **Y** was recognized as a main drawback in the PLS method because the real world data often exhibit nonlinearity [2,3]. Several nonlinear PLS techniques were developed to cope with the nonlinearity. These techniques can be categorized into three groups: transformation techniques, weight updating techniques, and function techniques.

In transformation techniques, new columns are added to the input matrix **X** so that the extended columns can take care of the unknown nonlinearity that the original input data may have. Each added block of columns is a transformed data of the original matrix using a certain nonlinear function such as logarithms, squared values, cross products, binary transformation, to name a few [4–6], i.e., $\mathbf{X} = [\mathbf{X}, f_1(\mathbf{X}), \ldots f_l(\mathbf{X})]$, where l is the number of transformations that was applied to extend the original input matrix **X**. Then the linear PLS method is applied between the extended input matrix and the original output matrix. In general, this approach is simple and easy to use, but it may lead to a poor fitting. Furthermore, it may also lead to fat and short data matrices. Such problems are difficult to handle in data analysis because we now have substantially more input variables (columns) with fixed observations (rows).

The weight updating methods attempt to update the input PLS weights in order to improve the output prediction and to reduce the regression errors. The weight updates can be done either by an iterative method [7–10] or by solving a nonlinear programming problem [11,12]. These methods still rely on either linear or quadratic inner PLS functions to capture unknown nonlinearity in the data.

The main idea of the function methods is to model the inner relationship between the input and the output PLS score vectors in a nonlinear fashion. Several nonlinear versions of the function methods have been developed such as quadratic PLS, spline PLS, and neural networks PLS. Wold *et al.* [4] have pioneered the quadratic PLS techniques and many other methods are variants of their work. They extended the two blocks linear PLS model using a quadratic function instead of a linear function. In spline PLS, a general form of a spline function is used for the inner relation. The spline function can be a B-spline, local polynomial, or additive [13–15]. In neural networks PLS, the nonlinearity is modeled using feed-forward neural networks, radial basis functions (RBF), or an artificial neural network [16–20]. Although there are many methods in the literature focusing on the nonlinear function techniques, some drawbacks include the complexity of implementation and over fitting the regression model parameters (poor interpretability) [21]. For example, quadratic PLS does not provide enough flexibility for modeling the complex nonlinear relationship. It is partly due to the quadratic function that is assumed to fit the PLS inner relation. On the other hand, the spline PLS and neural networks PLS are expected to give adequate flexibility for fitting complex nonlinearity, but the extra flexibility can be a source for over fitting and prediction error.

We attempt to overcome these drawbacks by introducing a new function method called Quadratic Fuzzy PLS (QFPLS). This

* *Correspondence to: G. J. Lim, Department of Industrial Engineering, University of Houston, USA.*
E-mail: ginolim@uh.edu

a *A. I. Abdel-Rahman, G. J. Lim*
Department of Industrial Engineering, University of Houston, USA

method combines the Takagi–Sugeno–Kang (TSK) fuzzy inference system [22,23] with PLS. The motivation for this is because TSK method is known to give a better interpretability [21]. The only attempt in the literature to use fuzzy systems was done by Bang *et al.* [24]. They used a linear TSK fuzzy system and gained some benefits over other nonlinear PLS methods. In this paper we extend their work to a quadratic TSK fuzzy system to model the inner PLS relation.

The rest of the paper is organized as follows. Section 2 describes the proposed method QFPLS in more details. Section 3 gives a comprehensive comparison of the proposed method against four other methods using randomly generated test cases. We give a summary of findings the in Section 4 with recommendations for future work.

2. METHODOLOGY

Our method combines the PLS method and the quadratic TSK fuzzy model. The PLS modeling is a multivariate linear regression technique that reduces the high dimensionality of correlated predictor variables (input matrix **X**) and response variables (output matrix **Y**) by projecting those variables to the input weight (*w*) and output weight (*c*) directions that maximize the covariance between input and output variables. The PLS projection decomposes the high co-linearity variables into one-dimensional variables in the form of input score vector (***t***) and output score vector (***u***). The relation between the score vectors (***t***) and (***u***) is called the inner relation in the PLS procedure and is crucial in the PLS prediction. The decomposition of **X** and **Y** by score vectors is formulated as in equation (1).

$$\begin{aligned} X &= \sum_{i=1}^{s} t_i p_i^{\mathrm{T}} + E = TP^{\mathrm{T}} + E \\ Y &= \sum_{i=1}^{s} u_i q_i^{\mathrm{T}} + F = UQ^{\mathrm{T}} + F \end{aligned} \tag{1}$$

where s is the number of PLS components, **X** is an $n \times K$ matrix and K is the number of input variables, **Y** is an $n \times M$ matrix and M is the number of output variables, n is the number of observations, E and F are the input and output residuals, respectively, ***p*** is the input loading vector, and ***q*** is the output loading vector.

The PLS procedure has two parts: outer PLS and inner PLS. In outer PLS as in equation (1), **T** and **U** are the score matrices, and **P** and **Q** are the loading matrices for the **X** and **Y** data sets (or blocks) respectively. The loading vectors (***p*** and ***q***) are the direction cosines of the dominant directions within the data set. Projection of the **X** and **Y** data sets on the loading vectors will give the score vectors (***t***) and (***u***). In the inner PLS part, the **X** and **Y** matrices are indirectly related through their scores by the inner model which is a function of (***t***) on (***u***). The procedure of determining the scores and loading vectors of the inner relation is continued until the required number of PLS components (*s*) are extracted. The number of PLS components is determined based on the percentage variance explained or using statistically sound approaches such as cross validation.

PLS is linear if the inner relation is linear and we call this LPLS. LPLS can be used for data classification, data mining, and image processing [25]. For example, input weight *w* and output weight *c* can be used to find the contributions of different variables to each score while input score vector ***t*** and output score vector ***u*** can be used to detect outliers. LPLS is limited to modeling linear cases while the real world data often exhibit nonlinearity. Various nonlinear PLS methods have been introduced to overcome the nonlinearity situation. However, each of these approaches inherently carries shortcomings such as complexity, lack of analytical interpretability of regression coefficients, and so on [21]. To overcome such shortcomings, we introduce the QFPLS method in the following section.

2.1. The QFPLS modeling methodology

In the QFPLS method, the TSK model is applied to the PLS inner regression. The PLS outer projection is used to reduce the dimension and to remove the co-linearity, and the TSK fuzzy inner model is used to capture the nonlinearity in the projected latent space. In QFPLS, the data are not used directly to train the TSK model, but are preprocessed by the PLS outer transform, which in turn will decompose the multivariate regression problem into a few univariate regression problems and simplifies the TSK model. Figure 1 shows a schematic of the basic QFPLS method, which uses the PLS outer transform to generate score variables from the data.

Score vectors ($\boldsymbol{t}_h$ and $\boldsymbol{u}_h$) for the hth latent variable are used for the inner TSK fuzzy modeling. The relation between the score vectors can be expressed as

$$\hat{\boldsymbol{u}}_h = f_h(\boldsymbol{t}_h) + e_h \tag{2}$$

where e_h is the regression error.

The general procedure of QFPLS method can be defined as follows:

(1) Preprocess by centering and scaling of the **X** and **Y** matrices. At $h = 1$ assume that $E_0 = \mathbf{X}$ and $F_0 = \mathbf{Y}$. (centering and scaling are to have zero mean and unit variance).
(2) For each h, assume $\boldsymbol{u}_h$ as a column in F_{h-1}.
(3) Perform the PLS outer transformation as in equation (3). This step is called the nonlinear iterative partial least squares algorithm (NIPALS). Iterate this step until it converges over the scores ***u***.

$$\begin{aligned} w_h^{\mathrm{T}} &= \frac{\boldsymbol{u}_h^{\mathrm{T}} E_{h-1}}{(\boldsymbol{u}_h^{\mathrm{T}} \boldsymbol{u}_h)} \\ w_h &= \frac{w_h}{\|w_h\|} \\ \boldsymbol{t}_h &= E_{h-1} w_h \\ c_h^{\mathrm{T}} &= \frac{\boldsymbol{t}_h^{\mathrm{T}} F_{h-1}}{(\boldsymbol{t}_h^{\mathrm{T}} \boldsymbol{t}_h)} \\ c_h &= \frac{c_h}{\|c_h\|} \\ \boldsymbol{u}_h &= F_{h-1} c_h \end{aligned} \tag{3}$$

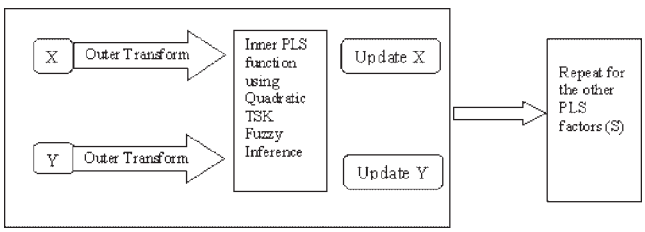

Figure 1. A schematic of the QFPLS method.

(4) Model the inner PLS function $f_h(t_h)$ which predicts the output scores $\boldsymbol{u}_h$ with the input scores t_h. The quadratic TSK fuzzy inference method is used to model this relation as in equation (4). In the next section, this part will be described in detail.

$$f_h(\boldsymbol{t}) = \sum_{i=1}^{L} G_i(b_{i0} + b_{i1}\boldsymbol{t} + b_{i2}\boldsymbol{t}^2) \tag{4}$$

(5) Determine the input and output loading vectors as in equation (5).

$$\begin{aligned} p_h^T &= \frac{\boldsymbol{t}_h^T E_{h-1}}{(\boldsymbol{t}_h^T \boldsymbol{t}_h)} \\ q_h^T &= \frac{\hat{\boldsymbol{u}}_h^T F_{h-1}}{(\hat{\boldsymbol{u}}_h^T \hat{\boldsymbol{u}}_h)} \end{aligned} \tag{5}$$

(6) Calculate the residuals for each h as in equation (6).

$$\begin{aligned} E_h &= E_{h-1} - \boldsymbol{t}_h p_h^T \\ F_h &= F_{h-1} - \hat{\boldsymbol{u}}_h q_h^T \end{aligned} \tag{6}$$

(7) Update $h = h + 1$, then return to step [2] until all PLS factors are calculated.

2.2. The quadratic TSK fuzzy modeling algorithm

A fuzzy inference system is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made. The process of fuzzy inference consists of membership functions, fuzzy logic operators, and if–then rules. Fuzzy inference systems have been successfully applied in the areas such as automatic control, data classification, decision analysis, expert systems, and computer vision. There are usually two types of fuzzy inference systems differing in defuzzification, namely Mamdani-type and Sugeno-type. Mamdani–type inference expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. Rather than integrating across functions to find defuzzified output, it is also possible to use the weighted average of a few data points. Sugeno-type systems support this type of model. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are constant, linear, quadratic, etc.

The fuzzy inference method proposed by Takagi, Sugeno, and Kang, which is known as the TSK model in fuzzy systems field, has been well recognized in both theoretical and practical research for fuzzy modeling and control. The basic idea is to subdivide the input space into fuzzy regions and to approximate the system in each subdivision by a simple model. The main advantage of the TSK model is its capability of describing a highly complex nonlinear system using a small number of simple rules. Typically, a TSK model consists of many IF–THEN rules as in equation (7).

$$R^k = \begin{cases} \text{If} \langle x_1 \text{ is } A_1^k \rangle, \ldots, \text{ and } \langle x_n \text{is} A_n^k \rangle, \\ \text{Then}, y = \alpha_0^k + \alpha_1^k x_1 + \cdots + \alpha_n^k x_n, \end{cases} \quad k = 1, 2, \ldots, L \tag{7}$$

where L is the number of fuzzy rules $\{R^k\}_{k=1,\ldots,L}$, $\{A_n^k\}$ are fuzzy sets characterized by membership functions $A_n^k(x)$, $\{\alpha_i^k\}_{i=0,\ldots,n}$ are real-valued parameters and x_i are the input variables. The overall output of the model is calculated as in equation (8).

$$y = \frac{\sum_{i=1}^{L} \tau_i y_i}{\sum_{i=1}^{L} \tau_i} = \frac{\sum_{i=1}^{L} \tau_i(\alpha_0^i + \alpha_1^i x_1 + \cdots + \alpha_n^i x_n)}{\sum_{i=1}^{L} \tau_i} \tag{8}$$

where τ_i is the firing strength of rule R^i and is defined as in equation (9)

$$\tau_i = A_1^k(x_1) \times A_2^k(x_2) \times \cdots \times A_n^k(x_n) \tag{9}$$

Generally, the Gaussian-type membership function can be used to build the model as in equation (10).

$$A_s^k(x_s) = \exp\left(-\frac{(x_s - c_s^k)^2}{2\sigma_k^2}\right), k = 1, 2, \ldots, L \tag{10}$$

where c_s^k is the center of the k th Gaussian membership function of the sth input variable x_s, and σ_k is the standard deviation of the membership function.

In our QFPLS model, we use a quadratic TSK formulation. The input variable is the PLS input scores vector, and the output variable is the PLS output scores vector. For each PLS component h, the predicted PLS output scores $\hat{u}_h$ is calculated as a function of the PLS input scores t_h as in equation (11).

$$\begin{aligned} &\hat{\boldsymbol{u}}_h = f_h(\boldsymbol{t}_h) + e_h = \sum_{i=1}^{L} G_i(b_{i0} + b_{i1}\boldsymbol{t} + b_{i2}\boldsymbol{t}^2) \\ &\text{where} \\ &G_i = \frac{\tau_i}{\sum_{i=1}^{L} \tau_i} \\ &\tau_i(\boldsymbol{t}) = \exp\left(-\frac{(\boldsymbol{t} - c_i)^2}{2\sigma_i^2}\right) i = 1, 2, \ldots, L \end{aligned} \tag{11}$$

G_i is the normalized firing strength and τ_i is a Gaussian type firing strength for the ith rule. L is the number of fuzzy rules. L should be estimated by minimizing the regression error$_h$ [26]. The other parameters (c_i, σ_i, and b_i) in the previous formulation can be determined as described in the following sections.

(a) Estimation of c_i by Fuzzy c-means (FCM) algorithm

Fuzzy c-means (FCM) is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade. This technique was originally introduced by Bezdek in (27) as an improvement on earlier clustering methods. It provides a method that shows how to group data points that populate some multidimensional space into a specific number of different clusters. It is based on minimization of the following formula:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{L} \mu_{ij}^m \left\| t_i - c_j \right\|^2, \quad 1 \le m \le \infty \tag{12}$$

where m is any real number greater than 1, μ_{ij} is the degree of membership of $t_i$in the cluster j, t_i is the ith of the d-dimensional measured data, c_j is the d-dimension center of the cluster. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown in equation (12). Then the membership grade $\mu_{ij}$and the cluster centers c_j are updated as in equation (13) (assuming $m=2$) and iterate until $\|\mu_k - \mu_{k+1}\| \leq \varepsilon$.

$$c_i = \frac{\sum_{j=1}^{N} \mu_{ij}^2 t_j}{\sum_{j=1}^{N} \mu_{ij}^2}, \quad i = 1, 2, ..., L$$
$$\text{where} \quad \mu_{ij} = \frac{1}{\sum_{k=1}^{L} \left[\frac{|t_j - c_i|}{|t_j - c_k|}\right]^2} \tag{13}$$

(b) Estimation of $\sigma_i$by Moody and Darken's rule [28]

The width of a Gaussian-type membership function, σ_i, can be decided by using the p-nearest neighborhood heuristic suggested by Moody and Darken [28] as in equation (14).

$$\sigma_i = \left[\frac{1}{P}\sum_{l=1}^{P}(c_i - c_l)^2\right]^{1/2} \tag{14}$$

where c_l ($l=1, 2,\ldots, p$) are the p (typically $p=2$) nearest neighborhoods of the center c_i.

(c) Estimation of $b_i$by Global learning algorithm [29]

The parameters,b_i, of a fuzzy rule can be determined by using a global learning algorithm. Global learning chooses the parameters of fuzzy rules that minimize the objective functionJ_G. Equation (15) shows a complete formulation of the global learning algorithm.

$$J_G = \sum_{k=1}^{N}[u(k) - \hat{u}(k)]^2 = (u_G - T_G b_G)^T (u_G - T_G b_G) \tag{15}$$

where

$$u_G = [u(1)\ u(2)\ \cdots\ u(N)]^T$$
$$b_G = [b_{10}\ b_{11}\ b_{12}\ \cdots\ b_{L0}\ b_{L1}\ b_{L2}]^T$$
$$T_G = \begin{bmatrix} G_1(1) & G_1(1)t(1) & G_1(1)t(1)t(1)\cdots & G_L(1) & G_L(1)t(1) & G_L(1)t(1)t(1) \\ G_1(2) & G_1(2)t(2) & G_1(2)t(2)t(2)\cdots & G_L(2) & G_L(2)t(2) & G_L(2)t(2)t(2) \\ & & & & & \\ G_1(N) & G_1(N)t(N) & G_1(N)t(N)t(N)\cdots & G_L(N) & G_L(N)t(N) & G_L(N)t(N)t(N) \end{bmatrix}$$

Applying the singular value decomposition (SVD) to T_G yields

$T_G = USV$
where
$U = [\hat{u}_1\ \hat{u}_2 \ldots\ \hat{u}_N]^T \in R^{N\times N}$
$V = [\hat{v}_1\ \hat{v}_2 \ldots\ \hat{v}_{3L}]^T \in R^{3L\times 3L}$
$S = \text{diag}(\hat{\sigma}_1, \hat{\sigma}_2, \ldots, \hat{\sigma}_{3L})$
$b_G = \sum_{i=1}^{S} \frac{\hat{u}_i^T u_G}{\hat{\sigma}_i}\hat{v}_i$
s is the number of nonzero singular values in S

Computational complexity of this algorithm can be analyzed, but it is beyond the scope of this paper.

3. NUMERICAL EXPERIMENTS

3.1. Experimental setup

We compare the QFPLS method against four other well known PLS methods: LPLS, QPLS, LFPLS, and NNPLS. All methods use the same outer PLS framework, but each method uses a different inner PLS framework (this makes one method different from others). LPLS is the linear partial least squares method proposed by Geladi and Kowalski. In LPLS, the PLS output scores vector ($\boldsymbol{u}$) is predicted using a linear function of the PLS input scores vector ($\boldsymbol{t}$). QPLS is the quadratic partial least squares method introduced by Wold *et al.* [4]. QPLS uses a quadratic function to model the PLS inner relation. LFPLS is the linear fuzzy partial least squares proposed by Bang *et al.* [24]. LFPLS uses a fuzzy inference system called TSK method to model the inner PLS relation. NNPLS is the neural network partial least squares method which was proposed by Qin and McAvoy [16]. NNPLS incorporates the feed-forward neural networks into the PLS modeling. The nonlinear inner relation modeling is performed by a number of single input single output networks and a conjugate gradient learning method is employed to train the network.

In order to make a fair performance comparison among the five methods, we first generated five sets of data that represent five different function types of multi–input–multi-output data sets: linear ($aX + b$), quadratic ($aX^2 + bX + c$), cubic ($aX^3 + bX^2 + cX + d$), sinusoidal ($a \cdot \sin(X) + b$), and exponential ($a \cdot \exp(X) + b$), where a, b, c, and d are the corresponding function coefficients, X is the randomly generated data, i.e., $X_{ij}\tilde{N}(\hat{\mu}_j, \hat{\sigma}_j)$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots \infty, K$, and it is generated by commands *rand* for$\hat{\mu}_j$ in Matlab, 2008b. There are 90 randomly generated test sets for each function type, in which 30 sets are labeled A for small size, 30 sets are labeled B for medium size, and 30 sets are for larger data and labeled C. These three data categories are based on the numbers of the input and the output variables. Therefore, a total of 450 data sets are used for testing the performance of the methods (see Table I). Each data set consists of 100 rows (or observations, $n=100$): $X \in \Re^{100\times K}$ and $Y \in \Re^{100\times M}$. For Category A, K is the number of process (input) variables and M represents the number of product (output) variables. K ranges from 5 to 10 while M ranges from 1 to 5. Each of the data sets may have different values of K and M, and the mean$(\hat{\mu}_j)$. Note that the standard deviation$(\hat{\sigma}_j)$ is set to 1, which reflects white noise to our data, i.e., $X_{ij} = \hat{\mu}_j + \varepsilon_{ij}$, $\varepsilon_{ij}\tilde{N}(0, 1)$, $\forall(i, j)$. Matlab command *randn* is used to generate ε_{ij}. Each data set contains a combination of the input and the output variables whose numbers are listed in Table I.

One can estimate how many PLS components to be included in the model using Cross Validation. However, this is beyond the scope of this paper. In our paper, the maximum number is set to 15, i.e., min (K, 15). This is because it is reported in the literature that 3–5 PLS components will give sufficient information about the data [31]. Furthermore, the well-known software

Table I. Generation of 90 data sets for each function type

Category	*K* (Process variables)	*M* (Product variables)	Combinations
A	5, 6, 7, 8, 9, and 10	1, 2, 3, 4, and 5	30
B	15, 20, 25, 30, 40, and 50	6, 7, 8, 9, and 10	30
C	55, 60, 70, 80, 90, and 100	11, 12, 13, 14, and 15	30

SAS (version 9.1) sets 15 as a default parameter for the maximum number of PLS components in the model. The logic behind this is that adding more than 15 components in the model generally does not add much to the captured variability.

In PLS analysis, two approaches are often used to compare the performance of different PLS methods [32]. The first approach is to use a training set to build the PLS model and then use an independent data set to test the efficiency of the model. The root mean square error of prediction (RMSEP) is an example of this type of performance measure. The second approach is to use a training set only for both "build" and "test." The output (**Y**) variables cumulative per cent variance captured by the PLS latent variables is a typical measure of performance in this approach. In this paper, we use both approaches for comparing the performance of different PLS methods.

First, we use the output (**Y**) variables' cumulative per cent variance captured by the PLS latent variables for comparing the performance of the five PLS methods using all 450 test sets. This has been a standard measure for testing the quality of the PLS model building in the literature [5,11]. It is obtained by dividing the explained sum of squares by the corresponding total sum of squares (i.e., SSY). The more the explained variance, the better is the method. Note that this measure is similar to a cumulative *R*-square in regression analysis.

Second, we use the root mean-square error of prediction (RMSEP) on independent data sets for comparing the five PLS methods. The experiments were performed based on 15 data sets. Each data set has 20 observations. There are three data sets per data type (small, medium, and large), i.e., 15 cases. We use the RMSEP formula given in equation (16).

$$\text{RMSEP} = \left\{ \sum \sqrt{\frac{\sum_{i=1}^{M} (y_i - \hat{y}_i)}{M}} \right\} / 20 \qquad (16)$$

A method with a smaller value of RMSEP is better. We also calculate the average RMSEP per data type (linear, quadratic, cubic, exponential, and sinusoidal) as in equation (17).

$$\text{Average RMSEP (per data type)} = \text{AvRMSEP} = \sum_{j=1}^{3} \text{RMSEP}_j \qquad (17)$$

Our procedure is given below and the results are shown in Table III in Section 3.2.

Procedure RMSEP

Step 1: *Parameter Estimation or Model Building*

We generate two random input training matrices ($100 \times K$) and ($100 \times M$) for building the PLS models. This provides us with the PLS parameters to use in Step 2.

Step 2: *Model Testing*

We randomly generate 20 independent observations that consist of *K* independent variables ($20 \times K$) and *M* dependent variables ($20 \times M$). We use the model in Step 1 to estimate $\hat{y}$(predicted output).

Step 3: *RMSEP Calculation*

$$\text{RMSEP} = \left\{ \sum \sqrt{\frac{\sum_{i=1}^{M} (y_i - \hat{y}_i)}{M}} \right\} / 20$$

for each test case is calculated.

Step 4: *AvRMSEP Calculation*

$$\text{AvRMSEP} = \sum_{j=1}^{3} \text{RMSEP}_j$$

We repeat Steps 1–4. Note that we have five data types. Therefore, we run this procedure five times.

The following section shows an overall comparison results followed by detailed ones for each data type used in this study.

4. RESULTS

Figure 2 shows the captured variability of the first 15 data sets (out of 30) in Category A while Figure 3 shows results based on 15 data sets in Category B. LPLS does not seem to perform well on both of these cases. However, LPLS does better on the exponential data (Figure 3) than the quadratic case (Figure 2). This is not surprising because LPLS is based on linear PLS, which would favor exponential data over quadratic in data fitting. QPLS does a much better job than LPLS on both of these cases. It favors quadratic over exponential because QPLS is simply based on quadratic PLS. It is interesting to see that three other methods (LFPLS, NNPLS, and QFPLS) seem to be robust in predicting the

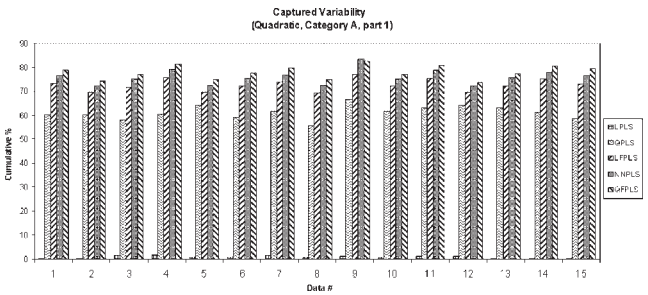

Figure 2. Captured variability (quadratic, category A, part 1).

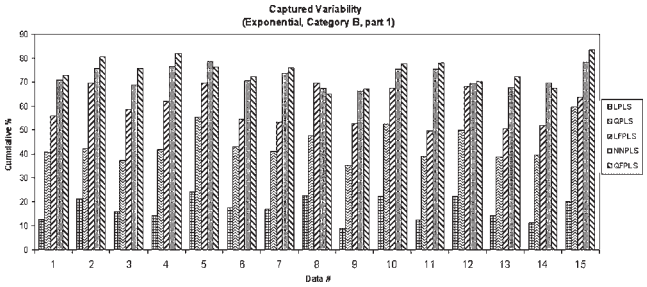

Figure 3. Captured variability (exponential, category B, part 1).

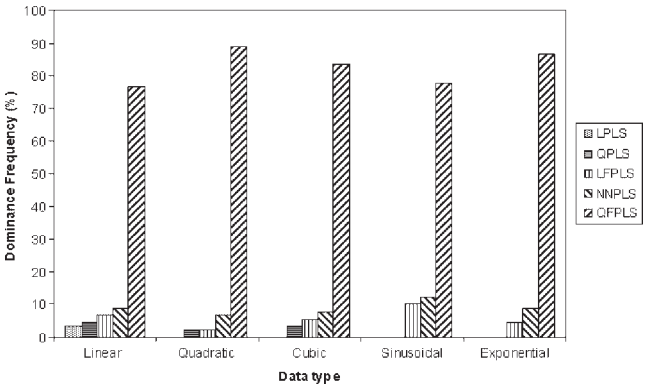

Figure 4. Dominance frequency of QFPLS over other methods.

output. All three methods perform well on all data sets tested. More output analysis follows on the next page.

Figure 4 shows an overall performance comparison among the five methods. It shows the dominance frequency in percentage that one method outperforms all other methods. All five methods performed well in the linear case data. QFPLS outperformed other methods in 76% of the linear cases. Note that when the test cases are linear, all methods performed reasonably well, i.e., the percentage differences among the five methods were relatively small.

As expected, LPLS does not do well for the quadratic case. The other methods (QPLS, LFPLS, NNPLS, and QFPLS) performed better than the LPLS method. The performance difference among LFPLS, NNPLS, and QFPLS was small, yet QFPLS was the better performer in most of the cases. QFPLS outperformed 88% of the quadratic cases. We observed similar results on the cubic data where QFPLS outperformed other methods 83% on all cubic test cases. The results of other methods were similar to those of the quadratic case. In the exponential data, QFPLS took the performance lead in 86% of the test cases. Two of the best performers were NNPLS and QFPLS with a small difference between the two methods. Performance lead of QFPLS continues in the sinusoidal cases with a 77% lead. The performance of LPLS and QPLS in the case of sinusoidal data was extremely poor. Although LFPLS performed better in the case of sinusoidal data than any other data, NNPLS and QFPLS were still the best performers.

So far, we used dominance frequency as a performance measure. We continue our performance comparison using a statistical measure, *the independent-samples t-test*, to show the dominance of QFPLS over the other four methods in the study. This test compares the means of two independent samples. For each pair of methods, we first checked the normality assumption of the dependent variable (the captured variability). If the variances between two methods are approximately equal, then we adopt the results of the independent-samples *t*-test. Otherwise, we cannot apply the *t*-test as is. Therefore, the *Levene's test* in SPSS (33) is used for approximation. The null hypothesis of the independent samples *t*-test is that the means of the two groups are not significantly different. While, the alternative hypothesis is that the means of the two groups are significantly different. By using 95% confidence level, we reject the null hypothesis if the *p*-value is less than 0.05. Our results using SPSS are shown in Table II. It shows the *p* (*p*-value) and *t* (*t*-statistic) values of the independent samples *t*-test between the QFPLS method and the other four methods in our comparison study. All *p*-values are zero or much less than 0.05. This indicates that QFPLS outperforms the other four methods. Note that *t*-values of NNPLS and QFPLS are somewhat small. This indicates that NNPLS performs close to QFPLS.

Table II. Results of the independent samples *t*-test between QFPLS and other methods

Data types	Statistic	Methods			
		LPLS	QPLS	LFPLS	NNPLS
Linear	*t*	5.301	5.403	3.347	2.214
	p	0	0	0.001	0.028
Quadratic	*t*	95.59*	17.125	4.808	2.291
	p	0*	0	0	0.023
Cubic	*t*	88.084*	13.604	4.184	2.098
	p	0*	0	0	0.037
Exponential	*t*	63.928	30.255*	14.362*	3.078
	p	0	0*	0*	0.002
Sinusoidal	*t*	122.202*	51.887	11.504*	6.503
	p	0*	0	0*	0

*We use the Levene's test to approximate the case of no equal variances.

Table III. RMSEP and avrmsep comparison among five PLS methods

	Size		RMSEP $= \left\{ \sum \sqrt{\frac{\sum_{i=1}^{M} (y_i - \hat{y}_i)}{M}} \right\} / 20$				
	K	*M*	LPLS	QPLS	LFPLS	NNPLS	QFPLS
Linear data	8	3	1.01	0.98	0.91	0.88	0.84
	30	8	2.19	2.18	1.99	1.93	1.75
	80	13	1.04	1.03	0.94	0.91	0.92
	AvRMSEP		1.41	1.40	1.28	1.24	1.17
Quadratic data	8	3	2.73	1.65	1.33	0.88	0.86
	30	8	2.44	1.50	1.24	0.94	0.90
	80	13	2.22	1.70	1.58	1.54	1.52
	AvRMSEP		2.46	1.62	1.38	1.12	1.09
Cubic data	8	3	5.16	2.02	1.93	1.43	1.15
	30	8	4.24	3.01	1.24	1.33	1.11
	80	13	3.43	2.45	1.35	1.46	1.53
	AvRMSEP		4.28	2.49	1.51	1.41	1.26
Exponential data	8	3	1.09	0.99	0.86	0.97	0.79
	30	8	1.25	1.32	1.23	1.08	1.17
	80	13	1.14	1.51	1.08	0.95	0.87
	AvRMSEP		1.16	1.27	1.06	1.00	0.94
Sinusoidal data	8	3	8.94	3.99	3.37	3.06	2.31
	30	8	6.50	5.07	3.03	3.11	2.23
	80	13	7.65	3.47	2.97	3.05	2.07
	AvRMSEP		7.70	4.18	3.12	3.07	2.20

We continue our comparison using RMSEP as the measure of performance. The results are displayed in Table III. Overall, QFPLS outperforms the other four PLS methods. The performance difference between QFPLS and NNPLS seems to be marginal in linear, quadratic, and exponential data. In linear data, there is not much difference among the five PLS methods. However, due to the complexity of the sinusoidal data, the RMSEP values of all methods are higher than those of other data types. It is interesting to see that RMSEP depends solely on the PLS method used and it is independent of the data size. Table III also shows the average RMSEP (AvRMSEP) per data type. The results favor QFPLS and NNPLS. On average, QFPLS clearly takes the performance lead in cubic and sinusoidal data.

5. CONCLUSION AND FUTURE WORK

We introduced a new nonlinear partial least squares method (QFPLS). The proposed method uses the PLS framework for the outer relations and uses the TSK fuzzy inference system for the inner relation. We used the quadratic TSK method which uses a quadratic function to model the relation between the input and the output PLS scores vectors. Comprehensive experiments on various types and sizes of data have shown that QFPLS clearly outperformed four other well-known methods (LPLS, QPLS, FPLS, and NNPLS) in the literature.

Our future work includes designing a fuzzy inference system with a higher order mapping function such as cubic or other nonlinear functions. Furthermore, we plan to develop a practical PLS framework in which a library of different PLS methods will be available for different users to choose the appropriate methods for their specific applications. This library will also include well known PLS methods found in the literature.

REFERENCES

1. Doymaz F, Palazoglu A, Romagnoli J. Orthogonal nonlinear partial least-squares regression. *Ind. Eng. Chem. Res.* 2003; **42**: 5836–5849.
2. Wold S, Sjostrom M, Eriksson L. PLS-regression: a basic tool of chemometrics. *Chem. Int. Lab. Sys.* 2001; **58**: 109–130.
3. Li C, Zhang J, Wang G. Batch to batch optimal control of batch processes based on recursively updated nonlinear partial least squares models. *Chem. Eng. Comm.* 2007; **194**: 261–279.
4. Wold S, Wold K, Skagerberg B. Nonlinear PLS modeling. *Chem. Int. Lab. Sys.* 1989; **7**: 53–65.
5. Berglund A, Wold S. INLR: implicit non-linear latent variable regression. *J. Chemometrics* 1997; **11**: 141–156.
6. Berglund A, Kettaneh N, Uppgard L, Wold S, Bendwell N, Cameron D. The GIFI approach to nonlinear PLS modeling. *J. Chemometrics* 2001; **15**: 321–336.
7. Baffi G, Martin E, Morris A. Nonlinear projection to latent structures revisited: the quadratic PLS algorithm. *Comput. Chem. Eng.* 1999; **23**: 395–411.
8. Hassel PA, Martin EB, Morris J. Nonlinear partial least square: estimation of the weight vector. *J. Chemometerics* 2002; **16**: 419–426.
9. Min KG, Han I-S, Han C. Iterative error-based nonlinear PLS method for nonlinear chemical process modeling. *J. Chem. Eng. Japan* 2002; **35**(7): 613–625.
10. Li B, Hassel PA, Morris AJ, Martin EB. A nonlinear nested partial least squares algorithm. *Comput. Stat. Data Anal.* 2005; **48**: 87–101.
11. Tu A, Tian A. A modified QPLS based on nonlinear constrained programming and its applications. *Proceedings of the 6th World congress on Intelligent Control and Automation*, Dalian, China, June 2006; 21–23.

12. Yu X, Huang D, Wang X, Liu B. DE and NLP based QPLS Algorithm. Lecture Notes in Computer Science, (LNAI Vol. 4682), 2007; 584-592.
13. Frank IE. A nonlinear PLS model. *Chem. Int. Lab. Sys.* 1990; **8**: 109–119.
14. Wold S. Nonlinear partial least squares modeling: II Spline inner relation. *Chem. Int. Lab. Sys.* 1992; **14**: 71–84.
15. Durand JF, Sabatier R. Additive splines for partial least squares regression. *J. Am. Stat. Assoc.*, 1997; **92**: 1546–1554.
16. Qin SJ, McAvoy TJ. Nonlinear PLS modeling using neural networks. *Comput. Chem. Eng.* 1992; **16**(4): 379–391.
17. Martin E, Xu L, Morris J. Non-linear projection to latent structures. *13th Triennial World Congress*, San Francisco, USA, 1996.
18. Wilson D, Irwin G, Lightbody G. Nonlinear PLS modeling using radial basis function. *Proceedings of the American Control Conference*, Albuquerque, New Mexico, June 1997.
19. Lee D, Lee M, Woo S, Kim Y, Park J. Nonlinear dynamic partial least squares modeling of a full scale biological wastewater treatment plant. *Process Biochemistry* 2006; **41**: 2050–2057.
20. Zhou Y, Jiang J, Lin W, Xu L, Yu R. Artificial neural network-based transformation for nonlinear partial least squares regression with application to QSAR studies. *Talanta* 2007; **185**: 848–853.
21. Yoo C, Bang Y, Lee I, Rosen C. Application of fuzzy partial least squares (FPLS) modeling nonlinear biological processes. *Korean J. Chem. Eng.* 2004; **21**(6): 1087–1097.
22. Sugeno M, Kang G. Structure identification of fuzzy model. *Fuzzy Set. Sys.* 1988; **28**: 15–33.
23. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. SMC* 1985; **15**(1): 116–132.
24. Bang YH, Yoo CK, Lee I. Nonlinear PLS modeling with fuzzy inference system. *Chem. Int. Lab. Sys.* 2003; **64**(2): 137–155.
25. Kourti T. Application of latent variable methods to process control and multivariate statistical process control in industry. *Int. J. Adapt. Control Signal Process.* 2005; **19**: 213–246.
26. Herrera L, Pomares H, Rojas I, Valenzuela O, Prieto A. TaSe, a Taylor series-based fuzzy system model that combines interpretability and accuracy. *Fuzzy Set. Sys.* 2005; **153**: 403–427.
27. Bezdek JC. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press: New York, 1981.
28. Moody J, Darken C. Fast learning in networks of locally-tuned processing units. *J. Neural Comp.* 1989; **1**: 281–294.
29. Yen J, Wang L, Gillespie CW. Improving the interpretability of TSK Fuzzy models by combining global learning and local learning. *IEE Trans. Sys.* 1998; **6**: 530–537.
30. Geladi P, Kowalski BR. Partial least squares regression: a tutorial. *Anal. Chimica Acta* 1986; **185**: 1–17.
31. Xu D, Albin S. Manufacturing startup problem solved by mixed integer quadratic programming and multivariate statistical modeling. *Int. J. Prod. Res.* 2002; **40**(3): 625–640.
32. Kim K, Lee J, Lee I. A novel multivariate regression approach based on kernel partial least squares with orthogonal signal correction. *Chemomet. Intel. Lab. Sys.* 2005; **79**: 22–30.
33. SPSS. SPSS Statistics 17.0, SPSS, Inc.