# AN ONLINE NIPALS ALGORITHM FOR PARTIAL LEAST SQUARES

*Alexander E. Stott, Sithan Kanna, Danilo P. Mandic, William T. Pike*

Electrical and Electronic Engineering Department, Imperial College London, SW7 2AZ, UK
E-mails: {alexander.stott10, shri.kanagasabapathy08, d.mandic, w.t.pike}@imperial.ac.uk

## ABSTRACT

Partial Least Squares (PLS) has been gaining popularity as a multivariate data analysis tool due to its ability to cater for noisy, collinear and incomplete data-sets. However, most PLS solutions are designed as block-based algorithms, rendering them unsuitable for environments with streaming data and non-stationary statistics. To this end, we propose an online version of the nonlinear iterative PLS (NIPALS) algorithm, based on a recursive computation of covariance matrices and gradient-based techniques to compute eigenvectors of the relevant matrices. Simulations over synthetic data show that the regression coefficients from the proposed online PLS algorithm converge to those of its block-based counterparts.

***Index Terms***— Latent variables, NIPALS, regression, Rayleigh quotient, online learning.

## 1. INTRODUCTION

Partial Least Squares (PLS) regression is a data analysis tool used to find relationships between blocks of multivariate data. The key attribute of PLS is its ability to deal with collinear and noisy data, an ill-posed problem for traditional ordinary least squares (OLS) solutions [1]. Importantly, unlike standard regression algorithms for colinear data, such as principal component regression (PCR), the PLS does not require the user to choose principal components in the data before finding the relationships between the independent and dependent variables. These benefits have led to successful applications of PLS in many areas such as chemometrics, social sciences and bioinformatics [2], however, the PLS is still rather unexplored in the signal processing community.

The main drawback of conventional PLS algorithms is that their block-based nature requires simultaneous processing of all the available observations. This grossly limits the use of PLS in applications involving streaming data with non-stationary statistics. In addition, block-based methods tend to be computationally prohibitive for large scale problems, where gradient-based and online estimators are becoming standard [3].

Although finding a unified solution to the online PLS algorithm is still an open problem, several efforts in this direction have been made. These include, a recursive solution based on a sliding-window type technique [4], incremental solutions specifically designed for dimensionality reduction [5] or solutions limited to univariate data [6]. None of these methods have exploited ideas from stochastic gradient-based learning which has been shown to routinely outperform other methods in large scale settings [3, 7].

To this end, we propose an online PLS algorithm which is capable of finding a common subspace for multivariate data which are collected in an online fashion. The proposed online PLS algorithm employs well-known ideas, such as recursive computation of empirical cross-covariance matrices and stochastic gradient maximisation of the Rayleigh quotients, to estimate the scores and loading factors of the data matrices. The proposed result has similarities with online manifold learning methods [8].

The rest of this paper is organised as follows. In Section 2, a background of the PLS solution is provided and its extension to an online format is derived in Section 3. Simulation over synthetic data for an online prediction setting is included in Section 4 to verify the performance for the proposed algorithm.

## 2. BACKGROUND

Consider a multivariate regression problem of predicting dependent variables (output), $\boldsymbol{Y} \in \mathbb{R}^{N\times p}$, from independent variables (input), $\boldsymbol{X} \in \mathbb{R}^{N\times m}$, where $N$ denotes the number of observations of the $p-$dimensional dependent variables and $m-$dimensional independent variables. The linear model for this regression problem is given by

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} \tag{1}$$

where $\boldsymbol{B} \in \mathbb{R}^{m\times p}$ is the matrix of regression coefficients. The ordinary least squares solution to (1) is given by

$$\hat{\boldsymbol{B}}_{\text{LS}} = \boldsymbol{X}_{LS}^{+}\boldsymbol{Y} \tag{2}$$

and is performed through calculating the matrix pseudoinverse defined as $\boldsymbol{X}_{\text{LS}}^{+} = (\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathsf{T}}$. Crucially, this requires the matrix $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$ to be invertible, so that for colinear independent variables in $\boldsymbol{X}$, the least squares solution in (2) is ill-posed due to the singularity of the of the matrix $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$.

The partial least squares (PLS) method was introduced by Wold [9] to solve this class of problems. Several methodologies to compute a block-based PLS solution exist, and without loss in generality, we consider the classical nonlinear iterative PLS (NIPALS) algorithm owing to its widespread use and simplicity [1] .

The principle of PLS is to jointly decompose the input matrix, $\boldsymbol{X}$, and the output matrix, $\boldsymbol{Y}$, into the matrices which contain only the mutually relevant components [9]. This is achieved through coupled decompositions in the form

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^{\mathsf{T}} \quad \boldsymbol{Y} = \boldsymbol{T}\boldsymbol{C}^{\mathsf{T}} \tag{3}$$

where $\boldsymbol{T} \in \mathbb{R}^{N\times r}$ is an orthogonal matrix common to both $\boldsymbol{X}$ and $\boldsymbol{Y}$, and is referred to as the matrix of "scores". The matrices $\boldsymbol{P} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{C} \in \mathbb{R}^{p\times r}$ are the so-called "loadings" of the individual input and output matrices,[1] while $r$ refers to the rank of the input matrix and is upper bounded as $r \leq m$. Notice that the PLS

[1] In the full NIPALS solution, $\boldsymbol{Y}$ is also decomposed in terms of another matrix $\boldsymbol{U}$ which are the equivalent scores in the column space of $\boldsymbol{Y}$, but are ommited here for brevity as they are not required for calculations.

decomposition in (3), constrains $\boldsymbol{X}$ and $\boldsymbol{Y}$ to lie in the same column space spanned by the score matrix, $\boldsymbol{T}$, and in that sense it establishes an input-output relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ which can be exploited for prediction.

The decomposition in (3) is accomplished through an iterative computation of the columns of the matrices in $\boldsymbol{T}$, $\boldsymbol{P}$ and $\boldsymbol{C}$. This is achieved by first calculating a component, $\boldsymbol{w} \in \mathbb{R}^{m\times 1}$, that maximises the empirical cross-covariance between the $\boldsymbol{X}$ and $\boldsymbol{Y}$ blocks defined as

$$\boldsymbol{S}_1 \stackrel{\text{def}}{=} \boldsymbol{X}_1^{\mathsf{T}}\boldsymbol{Y}_1 \tag{4}$$

where the subscript "1" in $\boldsymbol{S}_1$ is used to indicate that the PLS is an iterative procedure in which subsequent covariance matrices are calculated using "deflated" versions of the original data matrices, with the intial values $\boldsymbol{X}_1 \leftarrow \boldsymbol{X}$ and $\boldsymbol{Y}_1 \leftarrow \boldsymbol{Y}$.

The first principal vector, $\boldsymbol{w}_1 \leftarrow \boldsymbol{w}$, is found as the solution to the following optimisation problem

$$\boldsymbol{w}_1 = \underset{\|\boldsymbol{w}\|=1}{\arg\max}\ \boldsymbol{w}^{\mathsf{T}}\boldsymbol{S}_1\boldsymbol{S}_1^{\mathsf{T}}\boldsymbol{w} \tag{5}$$

and is given by the maximum eigenvector of the sample cross-covariance matrix (in the same iteration) post-multiplied by its transpose, that is

$$\boldsymbol{w}_1 = \text{Eig}_{\max}\{\boldsymbol{S}_1\boldsymbol{S}_1^{\mathsf{T}}\} \tag{6}$$

where the $\text{Eig}_{\max}\{\cdot\}$ operator extracts the maximum eigenvector of the matrix in its argument [2]. The so obtained $\boldsymbol{w}_1$ is then used to produce the first column of the score matrix, $\boldsymbol{T}$, in (3), in the form

$$\boldsymbol{t}_1 = \boldsymbol{X}_1\boldsymbol{w}_1 \tag{7}$$

To compute the loading matrices, $\boldsymbol{C}$ and $\boldsymbol{P}$, it is convenient to express the PLS decomposition in (3) as a sum of rank-1 matrices, produced through outer products of the columns of the matrix of scores, $\boldsymbol{T}$, and the corresponding columns of the loading matrices, $\boldsymbol{C}$ and $\boldsymbol{P}$, that is

$$\boldsymbol{X} = \sum_{d=1}^{r}\boldsymbol{t}_d\boldsymbol{p}_d^{\mathsf{T}}, \quad \boldsymbol{Y} = \sum_{d=1}^{r}\boldsymbol{t}_d\boldsymbol{c}_d^{\mathsf{T}} \tag{8}$$

Recall that equation (7) produces only the first score, $\boldsymbol{t}_1$, so that the initial rank-1 approximation of (8) is given by

$$\boldsymbol{X} \approx \boldsymbol{t}_1\boldsymbol{p}_1^{\mathsf{T}}, \qquad \boldsymbol{Y} \approx \boldsymbol{t}_1\boldsymbol{c}_1^{\mathsf{T}} \tag{9}$$

where the vectors $\boldsymbol{p}_1$ and $\boldsymbol{c}_1$ from (9) are respectively the first columns of the loading matrices $\boldsymbol{P}$ and $\boldsymbol{C}$ in (3) which are yet to be found. Notice that (9) represents nothing else than a variant of the standard least squares problem in (1) – (2), from which the solutions for $\boldsymbol{p}_1$ and $\boldsymbol{c}_1$ can be found by projecting $\boldsymbol{X}_1$ and $\boldsymbol{Y}_1$ on to the score $\boldsymbol{t}_1$ in (7), to give

$$\boldsymbol{p}_1 = \frac{\boldsymbol{X}_1^{\mathsf{T}}\boldsymbol{t}_1}{\boldsymbol{t}_1^{\mathsf{T}}\boldsymbol{t}_1}, \qquad \boldsymbol{c}_1 = \frac{\boldsymbol{Y}_1^{\mathsf{T}}\boldsymbol{t}_1}{\boldsymbol{t}_1^{\mathsf{T}}\boldsymbol{t}_1} \tag{10}$$

Finally, the remaining columns of $\boldsymbol{T}$, $\boldsymbol{C}$ and $\boldsymbol{P}$ are found by removing the impact of the score $\boldsymbol{t}_1$ from the original data $\boldsymbol{X}$ and $\boldsymbol{Y}$ to produce the "deflated" matrices

$$\boldsymbol{X}_2 \leftarrow \boldsymbol{X} - \boldsymbol{t}_1\boldsymbol{p}_1^{\mathsf{T}}, \quad \boldsymbol{Y}_2 \leftarrow \boldsymbol{Y} - \boldsymbol{t}_1\boldsymbol{c}_1^{\mathsf{T}} \tag{11}$$

The process in (4) – (11) is repeated $r$ times to find all $\boldsymbol{w}_d, \boldsymbol{t}_d, \boldsymbol{p}_d, \boldsymbol{c}_d$, for $d = 1, \ldots, r$, starting from the deflated covariance matrix, $\boldsymbol{S}_d = \boldsymbol{X}_d^{\mathsf{T}}\boldsymbol{Y}_d$ in (4). The eigenvector matrix, $\boldsymbol{W}$, score matrix, $\boldsymbol{T}$, and loading matrices, $\boldsymbol{P}$ and $\boldsymbol{C}$, then become

$$\begin{aligned} \boldsymbol{W} &= [\boldsymbol{w}_1, \ \ldots\ , \boldsymbol{w}_r], \qquad \boldsymbol{T} = [\boldsymbol{t}_1, \ \ldots\ , \boldsymbol{t}_r], \\ \boldsymbol{P} &= [\boldsymbol{p}_1, \ \ldots\ , \boldsymbol{p}_r], \qquad \boldsymbol{C} = [\boldsymbol{c}_1, \ \ldots\ , \boldsymbol{c}_r]. \end{aligned} \tag{12}$$

The eigenvectors, $\boldsymbol{w}_d$, of the iteration wise matrix $\boldsymbol{S}_d\boldsymbol{S}_d^{\mathsf{T}}$, $d = 1, 2, \ldots, r$, are obtained from the NIPALS given in Algorithm 1, and span the subspace of $\boldsymbol{X}$ which is relevant to the regression with $\boldsymbol{Y}$. Hence, they provide a low-rank PLS decomposition of the input matrix $\boldsymbol{X}$ in the form

$$\tilde{\boldsymbol{X}} = \boldsymbol{T}(\boldsymbol{P}^{\mathsf{T}}\boldsymbol{W})\boldsymbol{W}^{\mathsf{T}} \tag{13}$$

The regression coefficients, $\boldsymbol{B}$, are now calculated as

$$\hat{\boldsymbol{B}}_{\text{PLS}} = \tilde{\boldsymbol{X}}^{+}\boldsymbol{Y} \tag{14}$$

where, owing to its structure, $\tilde{\boldsymbol{X}}^{+}$ is straightforwardly calculated as

$$\tilde{\boldsymbol{X}}^{+} = \boldsymbol{W}(\boldsymbol{P}^{\mathsf{T}}\boldsymbol{W})^{-1}\boldsymbol{T}^{\mathsf{T}} \tag{15}$$

**Algorithm 1** The NIPALS Algorithm

1: **Initialise:** $\boldsymbol{X}_1 \leftarrow \boldsymbol{X}$, $\boldsymbol{Y}_1 \leftarrow \boldsymbol{Y}$,
2: **for** $d = 1, \ldots, r$ **do**
3: $\quad \boldsymbol{S}_d = \boldsymbol{X}_d^{\mathsf{T}}\boldsymbol{Y}_d$
4: $\quad \boldsymbol{w}_d = \text{Eig}_{\max}\{\boldsymbol{S}_d\boldsymbol{S}_d^{\mathsf{T}}\}$
5: $\quad \boldsymbol{t}_d = \boldsymbol{X}_d\boldsymbol{w}_d$
6: $\quad \boldsymbol{c}_d = \boldsymbol{Y}_d^{\mathsf{T}}\boldsymbol{t}_d/\boldsymbol{t}_d^{\mathsf{T}}\boldsymbol{t}_d$
7: $\quad \boldsymbol{p}_d = \boldsymbol{X}_d^{\mathsf{T}}\boldsymbol{t}_d/\boldsymbol{t}_d^{\mathsf{T}}\boldsymbol{t}_d$
8: $\quad \boldsymbol{X}_{d+1} = \boldsymbol{X}_d - \boldsymbol{t}_d\boldsymbol{p}_d^{\mathsf{T}}$, $\boldsymbol{Y}_{d+1} = \boldsymbol{Y}_d - \boldsymbol{t}_d\boldsymbol{c}_d^{\mathsf{T}}$
9: **end for**
10: Create matrices $\boldsymbol{W}$, $\boldsymbol{T}$, $\boldsymbol{P}$ and $\boldsymbol{C}$ from (12)
11: Find $\boldsymbol{B}_{\text{PLS}}$ from (14).

## 3. AN ONLINE PARTIAL LEAST SQUARES

The NIPALS algorithm outlined in Algorithm 1 requires block data inputs, $\boldsymbol{X} \in \mathbb{R}^{N\times m}$ and $\boldsymbol{Y} \in \mathbb{R}^{N\times p}$, where $N$ refers to the number of observations. However, in many practical scenarios all the observations, $N$, cannot be obtained simultaneously but are observed as streaming values, that is, at every time instant, $n$, only one row of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is observed.

Therefore, the key difference between the NIPALS formulation in (4) – (11) and the proposed online-NIPALS (OL-PLS) is that in the online setting, at each time instant $n$, we only have access to the $n$-th row of the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, denoted by $\boldsymbol{x}_n \in \mathbb{R}^{1\times m}$ and $\boldsymbol{y}_n \in \mathbb{R}^{1\times p}$, given as

$$\boldsymbol{x}_n \stackrel{\text{def}}{=} \boldsymbol{X}_{(n,1:m)}, \quad \boldsymbol{y}_n \stackrel{\text{def}}{=} \boldsymbol{Y}_{(n,1:p)} \tag{16}$$

The OL-PLS algorithm is next developed through recursive solutions to the steps in (4) – (11). To this end, the empirical cross-covariance matrix in (4) is first expressed as a sum of rank-1 matrices $\boldsymbol{x}_n^{\mathsf{T}}\boldsymbol{y}_n$, for

$n = 1, \ldots, N$, to give

$$\boldsymbol{S}_{1,N} = \sum_{n=1}^{N} \boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{y}_n \tag{17}$$

where the subscript $N$ in $\boldsymbol{S}_{1,N}$ indicates the number of observations used to compute the sample cross-covariance matrix in (17). Then, (17) can be computed recursively as

$$\boldsymbol{S}_{1,n} = \boldsymbol{S}_{1,n-1} + \boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{y}_n \tag{18}$$

and the same procedure can also be used for the deflated cross-covariance matrices, to give

$$\boldsymbol{S}_{d,n} = \boldsymbol{S}_{d,n-1} + \boldsymbol{x}_{d,n}^{\mathsf{T}} \boldsymbol{y}_{d,n}, \quad d = 1, \ldots, r \tag{19}$$

where $\boldsymbol{x}_{d,n}$ and $\boldsymbol{y}_{d,n}$ denote the deflated data vectors which constitute the deflated data matrices in (11). A forgetting factor, $0 \ll \lambda < 1$, can be introduced in (18) and (19) to account for non-stationary data statistics, which yields

$$\boldsymbol{S}_{d,n} = \lambda \boldsymbol{S}_{d,n-1} + (1-\lambda) \boldsymbol{x}_{d,n}^{\mathsf{T}} \boldsymbol{y}_{d,n} \tag{20}$$

The recursion in (20) provides a reliable estimate owing to the inherent averaging of the cross-covariance, which reduces the impact of outliers.

To explain the principle of OL-PLS, recall that the block-based PLS solution in steps (4) – (11) uses the empirical cross-covariance matrix, $\boldsymbol{S}_1$, to identify a basis to describe a subspace in $\boldsymbol{X}$ that exhibits co-variation with $\boldsymbol{Y}$. This basis is used to produce the corresponding latent variables, which are the columns of the score matrix $\boldsymbol{T}$, these in turn provide a robust prediction of the output $\boldsymbol{Y}$ from the input $\boldsymbol{X}$. The most important step for an online PLS algorithm is to produce an estimate of the eigenvectors $\boldsymbol{w}_{d,n}$, $d = 1, 2, \ldots, r$, at each time instant $n$. With an increase in $n$, these estimates should converge to a set of vectors equivalent to the NIPALS solution that describe the same joint subspace. Upon calculating the eigenvector, $\boldsymbol{w}_{d,n}$, the remaining variables can be straightforwardly computed in an online fashion.

The online PLS is now fully equipped to produce estimates of the covariance matrix $\boldsymbol{S}_{d,n}$, which is then used to produce the eigenvectors $\boldsymbol{w}_{d,n}$, at each time instant, $n$, and for every deflation iteration, $d = 1, 2, \ldots, r$. The estimation of $\boldsymbol{w}_{d,n}$ from (6), when performed in an online setting, can therefore be considered as a problem of adaptively estimating the largest eigenvectors of the matrix $\boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}}$.

### 3.1. Adaptive Estimation of the OL-PLS Variables

Consider the Rayleigh quotient, which for a matrix and vector pair $(\boldsymbol{A}, \boldsymbol{w})$ is defined as [10]

$$\mathcal{R}(\boldsymbol{A}, \boldsymbol{w}) \stackrel{\text{def}}{=} \frac{\boldsymbol{w}^{\mathsf{T}} \boldsymbol{A} \boldsymbol{w}}{\boldsymbol{w}^{\mathsf{T}} \boldsymbol{w}} \tag{21}$$

The vector $\boldsymbol{w}$ that maximises (21) represents the maximum eigenvector of the matrix $\boldsymbol{A}$. Therefore, the estimation of the maximum eigenvector, $\boldsymbol{w}_{d,n}$, of the matrix $\boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}}$ in (6) can be accomplished using the maximisation of the Rayleigh quotient in (21) for the matrix-vector pair $(\boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}}, \boldsymbol{w})$ along with the constraint $\|\boldsymbol{w}\| = 1$. This yields the cost function

$$\mathcal{J}_w \stackrel{\text{def}}{=} \frac{1}{2}\mathcal{R}(\boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}}, \boldsymbol{w}) \underset{\text{s.t.}\|\boldsymbol{w}\|=1}{} = \frac{\boldsymbol{w}^{\mathsf{T}} \boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}} \boldsymbol{w}}{2\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}} + \beta(1 - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}) \tag{22}$$

where $\beta$ is a Lagrange multiplier. The Rayleigh cost function in (22) can be maximised recursively, using gradient ascent methodology [11], in the form

$$\boldsymbol{w}_{d,n+1} = \boldsymbol{w}_{d,n} + \mu \frac{\partial \mathcal{J}_{\boldsymbol{w}}}{\partial \boldsymbol{w}}\bigg|_{\boldsymbol{w}=\boldsymbol{w}_{d,n}} \tag{23}$$

where $\mu$ is a step-size which controls the trade-off between convergence speed, stability and steady state error. The gradient $\partial \mathcal{J}_w / \partial \boldsymbol{w}$ is calculated as

$$\frac{\partial \mathcal{J}_{\boldsymbol{w}}}{\partial \boldsymbol{w}} = -\left( \frac{\|\boldsymbol{S}_{d,n}^{\mathsf{T}}\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|^2}\boldsymbol{I} - \boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}} \right) \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|^2} - \beta \boldsymbol{w}$$

The role of the Lagrange optimisation is to keep $\|\boldsymbol{w}\| = 1$ and to ensure the uniqueness of the vector that maximises the Rayleigh quotient. It can be shown that the Lagrange multiplier, $\beta$, is in the form

$$\beta = \frac{\boldsymbol{w}^{\mathsf{T}} \boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}} \boldsymbol{w}}{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}} (1 - \frac{1}{\|\boldsymbol{w}\|})$$

The OL-PLS algorithm follows the direction of this gradient, estimated from the available variables at the time instant $n$, to give the OL-PLS update

$$\begin{aligned} \boldsymbol{w}_{d,n+1} = \boldsymbol{w}_{d,n} - \mu\beta\boldsymbol{w}_{d,n} + \\ \mu \left( \frac{\|\boldsymbol{S}_{d,n}^{\mathsf{T}}\boldsymbol{w}_{d,n}\|^2}{\|\boldsymbol{w}_{d,n}\|^2}\boldsymbol{I} - \boldsymbol{S}_{d,n}\boldsymbol{S}_{d,n}^{\mathsf{T}} \right) \frac{\boldsymbol{w}_{d,n}}{\|\boldsymbol{w}_{d,n}\|^2} \end{aligned} \tag{24}$$

*Remark* 1. The analysis in [11] shows that the gradient ascent step in (24) converges to the largest eigenvector of $\boldsymbol{S}_d\boldsymbol{S}_d^{\mathsf{T}}$. This implies that if the estimate of $\boldsymbol{S}_{d,n}$ converges to the cross-covariance matrices $\boldsymbol{S}_d$ obtained by NIPALS in (4) as $n \to \infty$, then the online estimate, $\boldsymbol{w}_{d,n}$ will also converge to the eigenvectors, $\boldsymbol{w}_d$ in (6).

Upon estimating the eigenvectors, $\boldsymbol{w}_{d,n}$, the scores $t_{d,n}$ from (7) are computed as

$$t_{d,n} = \boldsymbol{x}_{d,n}\boldsymbol{w}_{d,n} \tag{25}$$

The next step is to find the corresponding loading vectors, $\boldsymbol{p}_d$ and $\boldsymbol{c}_d$, necessary for the deflation. Note that the computation in the block-based NIPALS algorithm in (10) gives the loading vectors

$$\boldsymbol{p}_{d,n} = \frac{\boldsymbol{x}_{d,n}}{t_{d,n}}, \quad \boldsymbol{c}_{d,n} = \frac{\boldsymbol{y}_{d,n}}{t_{d,n}} \tag{26}$$

However, the deflation of the input and output data using the loadings in (26) will yield a null-vector at each time instant. This is obvious from the deflation step in (11), modified for the online case, where

$$\boldsymbol{x}_{d+1,n} = \boldsymbol{x}_{d,n} - t_{d,n}\boldsymbol{p}_{d,n}^{\mathsf{T}} = \boldsymbol{x}_{d,n} - t_{d,n}\frac{\boldsymbol{x}_{d,n}}{t_{d,n}} = \boldsymbol{0}$$

and is a consequence of the fact the available data at each time instant are rank-1. Our solution to this issue is through an online estimation of optimal loading vectors, these are found via gradient descent min-

imisation of the following cost functions

$$\mathcal{J}_{\boldsymbol{p}} = \|\boldsymbol{x}_{d,n} - t_{d,n}\boldsymbol{p}^{\mathsf{T}}\|^2, \quad \mathcal{J}_{\boldsymbol{c}} = \|\boldsymbol{y}_{d,n} - t_{d,n}\boldsymbol{c}^{\mathsf{T}}\|^2 \tag{27}$$

for which the gradients are given by

$$\frac{\partial \mathcal{J}_{\boldsymbol{p}}}{\partial \boldsymbol{p}} = -2t_{d,n}\left(\boldsymbol{x}_{d,n}^{\mathsf{T}} - t_{d,n}\boldsymbol{p}\right)$$
$$\frac{\partial \mathcal{J}_{\boldsymbol{c}}}{\partial \boldsymbol{c}} = -2t_{d,n}\left(\boldsymbol{x}_{d,n}^{\mathsf{T}} - t_{d,n}\boldsymbol{c}\right)$$

The online estimates for the loading vectors now take the form

$$\boldsymbol{p}_{d,n+1} = \boldsymbol{p}_{d,n} + \mu t_{d,n}\left(\boldsymbol{x}_{d,n}^{\mathsf{T}} - t_{d,n}\boldsymbol{p}_{d,n}\right) \tag{28}$$

$$\boldsymbol{c}_{d,n+1} = \boldsymbol{c}_{d,n} + \mu t_{d,n}\left(\boldsymbol{x}_{d,n}^{\mathsf{T}} - t_{d,n}\boldsymbol{c}_{d,n}\right) \tag{29}$$

which are standard least mean square adaptive filters. This solution is equivalent to the block NIPALS algorithm which produces rank-1 least squares estimates of $\boldsymbol{X}$ and $\boldsymbol{Y}$, denoted by $\hat{\boldsymbol{X}} = \boldsymbol{t}\boldsymbol{p}^{\mathsf{T}}$ and $\hat{\boldsymbol{Y}} = \boldsymbol{t}\boldsymbol{c}^{\mathsf{T}}$, for each score on each iteration. Recall that $\boldsymbol{t}\boldsymbol{p}^{\mathsf{T}}$ and $\boldsymbol{t}\boldsymbol{c}^{\mathsf{T}}$ are then removed from the data blocks $\boldsymbol{X}$ and $\boldsymbol{Y}$. To this end, the instantaneous score, $t_{d,n}$, within OL-PLS is used to deflate each $\boldsymbol{x}_{d,n}$ and $\boldsymbol{y}_{d,n}$ through the estimates $\hat{\boldsymbol{x}}_{d,n} = t_{d,n}\boldsymbol{p}_{d,n}^{\mathsf{T}}$ and $\hat{\boldsymbol{y}}_{d,n} = t_{d,n}\boldsymbol{c}_{d,n}^{\mathsf{T}}$, with the full deflation scheme given by

$$\boldsymbol{x}_{d+1,n} = \boldsymbol{x}_{d,n} - t_{d,n}\boldsymbol{p}_{d,n}^{\mathsf{T}} \tag{30}$$
$$\boldsymbol{y}_{d+1,n} = \boldsymbol{y}_{d,n} - t_{d,n}\boldsymbol{c}_{d,n}^{\mathsf{T}}$$

In other words, the OL-PLS performs an adaptive estimation of $\boldsymbol{p}_{d,n}$ and $\boldsymbol{c}_{d,n}$ in order to converge to the least squares solution.

As a result, the proposed online extension of PLS adaptively estimates $\boldsymbol{w}_{d,n}$, $\boldsymbol{p}_{d,n}$ and $\boldsymbol{c}_{d,n}$ from the inputs $\boldsymbol{x}_n$ and $\boldsymbol{y}_n$, which provides the instantaneous estimates of the PLS components $t_{d,n}$. To this end, the matrices $\boldsymbol{S}_{d,n}$ are updated recursively based on the deflated inputs $\boldsymbol{x}_{d,n}$ and $\boldsymbol{y}_{d,n}$, which are also estimated adaptively as shown in (30).

The OL-PLS algorithm summarised in Algorithm 2, where the tuning parameters are $\mu$ and $\lambda$.

**Algorithm 2** Proposed online PLS (OL-PLS)

**Inputs at time instant** $n$**:** $\boldsymbol{x}_n \in \mathbb{R}^{1\times m}$ and $\boldsymbol{y}_n \in \mathbb{R}^{1\times p}$
**Initialise:** $\boldsymbol{S}_{d,1} \leftarrow \boldsymbol{x}_1^{\mathsf{T}}\boldsymbol{y}_1$
**Initialise:** $\boldsymbol{p}_{d,1}$, $\boldsymbol{q}_{d,1}$ and $\boldsymbol{c}_{d,1} \leftarrow 0$
**Initialise:** $\boldsymbol{w}_{d,1} \leftarrow$ first column of $\boldsymbol{S}_{d,1}$
**for** $d = 1, \ldots, r$ **do**
  $\boldsymbol{S}_{d,n} = \lambda\boldsymbol{S}_{d,n} + (1-\lambda)\boldsymbol{x}_{d,n}^{\mathsf{T}}\boldsymbol{y}_{d,n}$
  $\boldsymbol{w}_{d,n+1} = \boldsymbol{w}_{d,n} + \mu\frac{\partial \mathcal{J}_{\boldsymbol{S}_{d,n}}}{\partial \boldsymbol{w}_{d,n}^{\mathsf{T}}}$, using (24)
  $t_{d,n} = \boldsymbol{x}_{d,n}\boldsymbol{w}_{d,n}$
  $\boldsymbol{p}_{d,n+1} = \boldsymbol{p}_{d,n} + \mu t_{d,n}\left(\boldsymbol{x}_{d,n}^{\mathsf{T}} - t_{d,n}\boldsymbol{p}_{d,n}\right)$
  $\boldsymbol{c}_{d,n+1} = \boldsymbol{c}_{d,n} + \mu t_{d,n}\left(\boldsymbol{x}_{d,n}^{\mathsf{T}} - t_{d,n}\boldsymbol{c}_{d,n}\right)$
  $\boldsymbol{x}_{d+1,n} = \boldsymbol{x}_{d,n} - t_{d,n}\boldsymbol{p}_{d,n}^{\mathsf{T}}$, $\boldsymbol{y}_{d+1,n} = \boldsymbol{y}_d - t_{d,n}\boldsymbol{c}_{d,n}^{\mathsf{T}}$
**end for**

## 4. SIMULATIONS AND ANALYSIS

Simulations on synthetic data were conducted to verify the performance of the proposed OL-PLS algorithm. A particular emphasis was on the accuracy of OL-PLS in an online prediction setting. To this end, we generated $M = 10$ realisations of a low-rank input input matrix, $\boldsymbol{X} \in \mathbb{R}^{N\times 5}$, with rank $r = 3$, drawn from a unit variance Gaussian distribution. This input matrix was multiplied by a vector $\boldsymbol{b} \in \mathbb{R}^{5\times 1}$ to produce a univariate output $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{b}$. The OL-PLS algorithm in Algorithm 2 was then used to estimate the output $\boldsymbol{y}$ at each time instant. The performance measure was the empirical mean square error (MSE) computed as

$$\text{MSE}_n = \frac{1}{M}\sum_{\ell=1}^{M}|\boldsymbol{y}_n^{(\ell)} - \hat{\boldsymbol{y}}_n^{(\ell)}|^2$$

where $\hat{\boldsymbol{y}}_n^{(\ell)}$ denotes the $\ell$-th realisation of the OL-PLS output estimate at time $n$. Figure 1 (top panel) shows the ensemble average prediction MSE, calculated as a percentage of the total variance in $\boldsymbol{y}$, for $\lambda = 0.9999$ and $\mu = 0.0001$. For successful performance, the OL-PLS solution should be equivalent to that of the NIPALS algorithm, that is, the OL-PLS vectors $\boldsymbol{w}_{d,n}$ should converge to those produced by NIPALS, $\boldsymbol{w}_d$, as $n \to \infty$. To verify this, Figure 1 (bottom panel) shows the ensemble average of the MSE of each vector $\boldsymbol{w}_{d,n}$ (as a percentage of its total variance) and the corresponding vector calculated by NIPALS.

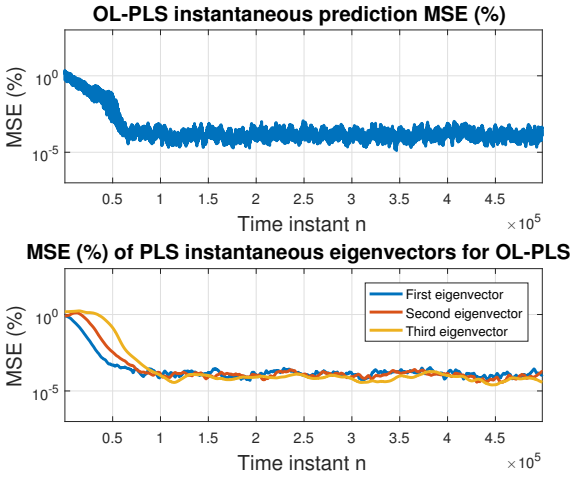


**Fig. 1**: OL-PLS error in predicting $\boldsymbol{Y}$

## 5. CONCLUSION

We have introduced an Online PLS (OL-PLS) algorithm as an extension of the popular NIPALS method for PLS-regression. It has been shown that a recursive estimation of the empirical cross-covariance matrices, $\boldsymbol{S}_d$, provides a suitable instantaneous estimate from which the eigenvectors of $\boldsymbol{S}_d\boldsymbol{S}_d^{\mathsf{T}}$ can be adaptively calculated through gradient ascent of the Rayleigh quotient. This makes it possible for the online version of PLS to adaptively calculate the scores, while the corresponding loadings are estimated through a stochastic gradient descent scheme. We have validated the proposed OL-PLS through illustrative simulations which confirm an accurate, online, estimate of the output $\boldsymbol{Y}$ and the NIPALS vectors $\boldsymbol{w}_d$. The convergence analysis and simulations over a larger number of case studies are the subject of ongoing work.

## 6. REFERENCES


[1] S. Wold, M. Sjöström, and L. Eriksson, "PLS-regression: A basic tool of chemometrics," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 2, pp. 109–130, 2001.

[2] R. Rosipal and N. Krämer, "Overview and recent advances in partial least squares," in *Subspace, Latent Structure and Feature Selection*, C. Saunders, M. Grobelnik, S. Gunn, and J. Shawe-Taylor, Eds. Springer Berlin Heidelberg, 2006, pp. 34–51.

[3] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 161–168.

[4] K. Helland, H. E. Berntsen, O. S. Borgen, and H. Martens, "Recursive algorithm for partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, vol. 14, no. 1, pp. 129 – 137, 1992.

[5] X. Q. Zeng and G. Z. Li, "Incremental partial least squares analysis of big streaming data," *Pattern Recognition*, vol. 47, no. 11, pp. 3726 – 3735, 2014.

[6] L. Qin, H. Snoussi, and F. Abdallah, "Online learning partial least squares regression model for univariate response data," in *Proc. of the 22nd European Signal Processing Conference (EUSIPCO)*, Sept 2014, pp. 1073–1077.

[7] L. Bottou, *Online learning and stochastic approximations*, D. Saad, Ed. Cambridge Univ Pr, 1998.

[8] A. B. Goldberg, M. Li, and X. Zhu, *Online Manifold Regularization: A New Learning Setting and Empirical Study*, W. Daelemans, B. Goethals, and K. Morik, Eds. Springer Berlin Heidelberg, 2008.

[9] H. Wold, "Estimation of principal components and related models by iterative least squares," in *Multivariate Analysis*. Academic Press, New York, 1966, vol. 6, pp. 391–420.

[10] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.

[11] R. Arablouei, S. Werner, and K. Dogancay, "Analysis of the gradient-descent total least-squares adaptive filtering algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1256–1264, 2014.