Signal Processing 160 (2019) 316–327

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

A class of multidimensional NIPALS algorithms for quaternion and tensor partial least squares regression

Alexander E. Stott*, Bruno Scalzo Dees, Ilia Kisil, Danilo P. Mandic

Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK

ARTICLE INFO

Article history:
Received 24 October 2018
Revised 8 February 2019
Accepted 3 March 2019
Available online 4 March 2019

Keywords:
Multidimensional signal processing
Partial least squares
Component analysis
Latent variables
Matrix factorisation

ABSTRACT

Quaternion and tensor-based signal processing benefits from exploiting higher dimensional structure in data to outperform the corresponding approaches using multivariate real algebras. Along with the extended range of processing options, these methods produce opportunities for a physically-meaningful interpretation. In this paper, we propose a class of novel partial least squares (PLS) algorithms for tensor- and quaternion-valued data, the widely linear quaternion PLS (WL-QPLS), the higher order nonlinear iterative PLS (HONIPALS) and the generalised higher order PLS (GHOPLS). This enables a regularised regression solution along with a latent variable decomposition of the original data based on the mutual information in the input and output block. The performance of the proposed algorithms is verified through analysis, together with a detailed comparison between quaternions and tensors and their application for image classification.

1. Introduction

Multidimensional signal processing deals with higher dimension variables. The rich structure in direct multidimensional processing offers a better treatment for many types of recorded data compared to interpreting them as separate variables in a multivariate system. Two such techniques are given by quaternions and tensors. The quaternion number system offers a 4D algebra with which to treat data, whereas tensors consider a generic, multilinear approach. Each technology has arose separately, whereby the quaternions are an extension of complex numbers and tensors are an extension of linear algebra concepts. However, there are clear links between the two, a concept we explore in this paper. So far, these methods have found many applications, such as: (i) power grid frequency estimation [1,2], (ii) image classification [3,4] and (iii) sensor network data [5,6]. These types of application are becoming more prevalent due to recent technological advances [7].

One such application, at the core of many data science and signal processing applications, is that of estimation. Specifically, it is desired to predict one dataset from another. In the case of linear regression, the problem is to predict the dependent variables (responses) as a linear combination of the independent variables (predictors). For an extension of this concept with tensor-variate and quaternion-valued data it is necessary to account for the degrees of freedom in the prediction. In this work, this has been achieved through the tensor formalism [8] and by using the widely linear regression for quaternions, which caters for full second-order statistics of a quaternion-valued variable [9].

The standard solution for linear regression, ordinary least squares (OLS), requires that the measured independent variables are full-rank in order to calculate the regression coefficients. Although it is extremely powerful, this is especially problematic for modern problems where data are collected from highly correlated or colinear data channels [7].

The method of partial least squares (PLS) has been developed to tackle the issues of ill-posedness in linear regression, with the nonlinear iterative PLS (NIPALS) algorithm proposed in [10]. This approach implements a dimensionality reduction framework which expresses the independent variables in an orthogonal basis which renders the problem tractable [11]. In contrast to other component analysis methods such as principal component regression (PCR) [12], the decomposition performed through the PLS algorithm is based on the maximal cross-covariance between the input and output blocks [12]. The result, therefore, takes advantage of the natural structure within the data and can be used for further data analysis. These properties have found several applications [13] and recent developments have extended the idea for online data [14].

The PLS has already been extended for multidimensional domains including complex data [15] and tensors [16,17]. These developments have led to new applications in power grid frequency estimation [15], and the processing of batch data [18].

* Corresponding author.
E-mail address: alexander.stott10@imperial.ac.uk (A.E. Stott).

https://doi.org/10.1016/j.sigpro.2019.03.002
0165-1684/

In this paper, we re-examine PLS-regression for multidimensional data. This is achieved through a critical account of tensor and quaternion signal processing methods for each step of the NIPALS algorithm. The required background on PLS-regression, tensors and quaternion algebra is described in Section 2. This is followed by the introduction of the higher order NIPALS (HONIPALS) and the generalised higher order PLS (GHOPLS) algorithms for tensor-variate data and the widely linear quaternion PLS (WL-QPLS) algorithm for quaternion-valued data in Section 3. The comparison and analysis of these algorithms is given in Section 4 using a derived isomorphism between the quaternion and tensor domains and by contrasting the cross-covariance data structures used in each algorithm. In Section 5 the proposed algorithms are implemented for a regression simulation on synthetic data. The WL-QPLS is then shown to be applicable for the problem of quaternion covariance matrix diagonalisation. Finally, the utility of the introduced class of multidimensional PLS (MD-PLS) algorithms is demonstrated through an application for image classification in Section 5.3, which shows their usefulness for subspace identification beyond classical regression applications.

Unless stated otherwise, we use uppercase boldface calligraphic letters for tensors, $\boldsymbol{\mathcal{X}}$, uppercase and lowercase boldface letters for matrices, $\mathbf{X}$, and vectors, $\mathbf{x}$, and lightface, lowercase italic letters, x, for scalars.

2. Background: PLS-regression and multidimensional methods

2.1. Linear regression

Linear regression is a common paradigm in signal processing and can be described by

$$\hat{\mathbf{Y}} = \mathbf{XB} \tag{1}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{N\times p}$ denotes an estimate of the output data matrix, $\mathbf{Y} \in \mathbb{R}^{N\times p}$, which is formed from a linear combination of the columns of the input data matrix $\mathbf{X} \in \mathbb{R}^{N\times m}$ and is determined by the regression coefficients $\mathbf{B} \in \mathbb{R}^{m\times p}$. Note that the data matrices $\mathbf{X}$ and $\mathbf{Y}$ consist of N samples of a vector variate process. The task is then to find the matrix $\mathbf{B}$, and the solution is given by

$$\mathbf{B} = \mathbf{X}^{+}\mathbf{Y} \tag{2}$$

where $(\cdot)^{+}$ denotes the matrix pseudoinverse. The solution is typically obtained through the ordinary least squares as

$$\mathbf{X}^{+} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}} \tag{3}$$

which aims to minimise the squared error between the predicted output variables, $\hat{\mathbf{Y}}$, and the original output matrix $\mathbf{Y}$. Note that this requires the inverse of the input covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

2.2. The two-way NIPALS algorithm

The OLS solution (3) encounters a problem when the data matrix $\mathbf{X}$ is sub-rank and so the matrix inverse $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ cannot be calculated. To address this issue, Wold et al. [10] introduced the PLS algorithm which produces a regression solution for such ill-posed cases. Specifically, the NIPALS algorithm counteracts the ill-posedness of the OLS solution by representing the data matrix $\mathbf{X}$ in a form such that a generalised inverse can straightforwardly be obtained, owing to its factorised structure. This is achieved by calculating successive rank-1 approximations of $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{tp}^{\mathrm{T}} \tag{4}$$

where $\mathbf{t}$ is a "score" vector component of $\mathbf{X}$ and $\mathbf{p}$ is its regression back to the inputs, $\mathbf{X}$, known as the "loadings". This component is selected by first calculating the maximal projection of the cross-covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ summarised by the optimisation problem

$$\mathbf{w} = \underset{||\mathbf{w}||=1}{\arg\max}\, ||\mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Y}||_2^2 \tag{5}$$

The resultant vector, $\mathbf{w}$, is then used to find the score, $\mathbf{t}$, as

$$\mathbf{t} = \mathbf{Xw}$$

As a result, the PLS algorithm determines a basis (described by the vector $\mathbf{w}$) that describes the most significant cross-covariance components between the inputs, $\mathbf{X}$, and the outputs, $\mathbf{Y}$, denoted by the vector $\mathbf{t}$ [19]. As such, it is suitable to also give a rank-1 approximation for $\mathbf{Y}$ as

$$\mathbf{Y} = \mathbf{tc}^{\mathrm{T}} \tag{6}$$

which is known as the "inner-relation" where $\mathbf{c}$ is the regression of $\mathbf{t}$ to $\mathbf{Y}$. An analogous score vector in $\mathbf{Y}$ can be found

$$\mathbf{u} = \mathbf{Yc}$$

This leads to a further rank-1 approximation of $\mathbf{Y}$ as

$$\mathbf{Y} = \mathbf{uq}^{\mathrm{T}} \tag{7}$$

At the end of the iteration the impact of the current score, $\mathbf{t}$, is removed (known as deflation) as

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i\mathbf{p}_i^{\mathrm{T}} \qquad \mathbf{Y}_{i+1} = \mathbf{Y}_i - \mathbf{t}_i\mathbf{c}_i^{\mathrm{T}} \tag{8}$$

where i indicates the number of iterations. This means that the score computed on the following iteration, $\mathbf{t}_{i+1}$, will be orthogonal to that obtained on the current iteration, $\mathbf{t}_i$. This form is known as PLS2. It is also possible to use the relation $\mathbf{Y} = \mathbf{uq}^{\mathrm{T}}$ to deflate the data matrix $\mathbf{Y}$ which is known as PLS Mode-A. However, in this form the scores are not generally orthogonal [19].

At the completion of the iteration the vectors, $\mathbf{t}$, $\mathbf{u}$, $\mathbf{p}$, $\mathbf{c}$ and $\mathbf{q}$ are stored into the respective columns of the matrices, $\mathbf{T} \in \mathbb{R}^{N\times r}$, $\mathbf{U} \in \mathbb{R}^{N\times r}$, $\mathbf{P} \in \mathbb{R}^{m\times r}$, $\mathbf{C} \in \mathbb{R}^{p\times r}$ and $\mathbf{Q} \in \mathbb{R}^{p\times r}$ where r is the number of iterations to be computed. The value of r for a given implementation is obtained through experimental cross-validation.

The data matrices are now expressed with the PLS approximation as

$$\tilde{\mathbf{X}} = \mathbf{TP}^{\mathrm{T}} \qquad \tilde{\mathbf{Y}} = \mathbf{TC}^{\mathrm{T}} \tag{9}$$

Furthermore, the input block $\mathbf{X}$ can be approximated as

$$\tilde{\mathbf{X}} = \mathbf{T}(\mathbf{P}^{\mathrm{T}}\mathbf{W})\mathbf{W}^{\mathrm{T}}$$

where $\mathbf{T}$ is orthogonal, $\mathbf{P}^{\mathrm{T}}\mathbf{W}$ is upper-triangular and $\mathbf{W}$ is also orthogonal. Therefore, the calculation of $\tilde{\mathbf{X}}^{+}$ is straightforward and gives a regularised regression solution (2) as

$$\mathbf{B}_{PLS} = \tilde{\mathbf{X}}^{+}\mathbf{Y} = \mathbf{W}(\mathbf{P}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{Y} \tag{10}$$

The full NIPALS PLS is summarised in Algorithm 1.

2.3. Tensor notation and preliminaries

A tensor is a multi-dimensional array that is a natural extension of a two-way matrix to a general M-way case, e.g. $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M}$, where M specifies the number of its dimensions, also known as *modes*, and is referred to as the *order* of a tensor. Consequently, each element of $\boldsymbol{\mathcal{X}}$ has M indices. For instance, the third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ is composed of scalars x_{ijk}. By fixing all but two indices in $\boldsymbol{\mathcal{X}}$, we obtain a matrix substructure that is called a *slice*, e.g., $\boldsymbol{\mathcal{X}}_{(:,:,k)} = \mathbf{X}_{::k}$. Likewise, a vector substructure, or *fibre*, requires fixing all but one index, e.g., for a third-order tensor is extracted as $\boldsymbol{\mathcal{X}}_{(:,j,k)} = \mathbf{x}_{:jk}$. An operation called *mode-n unfolding* can be used to reshape a tensor into a matrix. This is performed by grouping

Algorithm 1 The two-way NIPALS algorithm [10].

Input: $\mathbf{X}$, $\mathbf{Y}$, r
for $i = 1, \ldots, r$ **do**
  $\mathbf{S}_i = \mathbf{X}_i^{\mathrm{T}}\mathbf{Y}_i$
  $\mathbf{w}_i =$ first left singular vector of $\mathbf{S}_i$
  $\mathbf{t}_i = \mathbf{X}_i\mathbf{w}_i$
  $\mathbf{p}_i = \mathbf{X}_i^{\mathrm{T}}\mathbf{t}_i/\mathbf{t}_i^{\mathrm{T}}\mathbf{t}_i$
  $\mathbf{c}_i = \mathbf{Y}_i^{\mathrm{T}}\mathbf{t}_i/\mathbf{t}_i^{\mathrm{T}}\mathbf{t}_i$
  $\mathbf{u}_i = \mathbf{Y}_i\mathbf{c}_i$
  $\mathbf{q}_i = \mathbf{Y}_i^{\mathrm{T}}\mathbf{u}_i/\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_i$
  $\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{t}_i\mathbf{p}_i^{\mathrm{T}}$
  $\mathbf{Y}_{i+1} = \mathbf{Y}_i - \mathbf{t}_i\mathbf{c}_i^{\mathrm{T}}$
  Store $\mathbf{t}_i$, $\mathbf{p}_i$, $\mathbf{c}_i$, $\mathbf{u}_i$, $\mathbf{q}_i$ and $\mathbf{w}_i$
end for

mode-n fibres of a tensor $\mathcal{X}$ as the columns of the unfolded version $\mathbf{X}_{(n)}$. For example, the mode-n unfoldings of a third order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ are given as

$$\begin{aligned}\mathbf{X}_{(1)} &= [\mathcal{X}_{(:,1,1)} \quad \mathcal{X}_{(:,1,2)}\cdots \quad \mathcal{X}_{(:,1,K)}\cdots \quad \mathcal{X}_{(:,J,K)}],\\ \mathbf{X}_{(2)} &= [\mathcal{X}_{(1,:,1)} \quad \mathcal{X}_{(1,:,2)}\cdots \quad \mathcal{X}_{(1,:,K)}\cdots \quad \mathcal{X}_{(I,:,K)}],\\ \mathbf{X}_{(3)} &= [\mathcal{X}_{(1,1,:)} \quad \mathcal{X}_{(1,2,:)}\cdots \quad \mathcal{X}_{(1,J,:)}\cdots \quad \mathcal{X}_{(I,J,:)}].\end{aligned} \tag{11}$$

Mode-n unfolding permits the multiplication between a tensor $\mathcal{X}$ and a matrix $\mathbf{A}$. This operation is called a *mode-n product* and is defined as follows

$$\mathcal{Y} = \mathcal{X}\times_n \mathbf{A} \quad \Leftrightarrow \quad \mathbf{Y}_{(n)} = \mathbf{A}\mathbf{X}_{(n)}. \tag{12}$$

A tensor by tensor type product is performed by the *mode-n to mode-q contraction product*. For the tensors $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_q\times\cdots\times J_P}$ this is defined as

$$\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}\rangle_{\{n,q\}}, \tag{13}$$

where $\mathcal{S} \in \mathbb{R}^{I_1\times\cdots\times I_{n-1}\times I_{n+1}\times\cdots\times I_M\times J_1\times\cdots\times J_{q-1}\times J_{q+1}\times\cdots\times J_P}$ has the entries

$$\begin{aligned}s_{(i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_M,j_1,\ldots,j_{q-1},j_{q+1},\ldots,j_P)} \\ = \sum_{i_n=1}^{I_n} x_{(i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_M)} y_{(j_1,\ldots,j_{q-1},i_n,j_{q+1},\ldots,j_P)}.\end{aligned} \tag{14}$$

Note that $I_n = J_q$.

Another kind of unfolding operation is given by the *canonical matrix unfolding* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_n\times\cdots\times I_M}$. This is denoted as $\mathbf{X}_{\langle n\rangle} \in \mathbb{R}^{I_1I_2\cdots I_n\times I_{n+1}\cdots I_M}$, which has entries[1] [7,20]

$$\mathbf{X}_{\langle n\rangle(i_1i_2\cdots i_n,i_{n+1}\cdots i_M)} = \mathcal{X}_{(i_1,i_2,\ldots,i_M)}. \tag{15}$$

Note that this is not equivalent to the mode-n unfolding in (11).

2.3.1. Tucker decomposition

The Tucker decomposition of an order M tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M}$ is defined as

$$\mathcal{X} = \mathcal{G}\times_1 \mathbf{A}^{(1)}\times_2 \mathbf{A}^{(2)}\times_3\cdots\times_M \mathbf{A}^{(M)} \tag{16}$$

where $\mathcal{G} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_M}$ is the core tensor and $\mathbf{A}^{(i)} \in \mathbb{R}^{I_i\times J_i}$ are the factor matrices [21]. This gives rise to a new form of rank for tensors known as the multilinear rank. The Tucker decomposition decomposes a tensor into a low multilinear rank form, for the example above this is given as the M-tuple rank-$(J_1, J_2, \ldots, J_M)$, where each entry is defined as the number of non-zero eigenvalues in each factor matrix. This decomposition provides a compression of the original tensor into a smaller core tensor and can be computed by several algorithms, including the higher order SVD (HOSVD) and higher order orthogonal iteration (HOOI) [21].

The mode-n unfolding for a Tucker decomposition is expressed in terms of the core tensor and factor matrices as

$$\mathbf{X}_{(n)} = \mathbf{A}^{(n)}\mathbf{G}_{(n)}(\mathbf{A}^{(M)}\otimes\mathbf{A}^{(M-1)}\otimes\cdots\otimes\mathbf{A}^{(n+1)}\otimes\mathbf{A}^{(n-1)}\otimes\cdots\otimes\mathbf{A}^{(1)})^{\mathrm{T}} \tag{17}$$

Observe that this unfolding has a Kronecker product structure which is an important constraint of tensor decompositions.

2.3.2. Tensor regression

A tensor regression can be expressed [8]

$$\mathcal{Y} = f(\mathcal{X}; \mathcal{B}) \tag{18}$$

where the input $\mathcal{X} \in \mathbb{R}^{N\times I_1\times I_2\times\cdots\times I_M}$ and output $\mathcal{Y} \in \mathbb{R}^{N\times J_1\times J_2\times\cdots\times J_P}$ tensors are related through a tensor $\mathcal{B} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M\times J_1\times J_2\times\cdots\times J_P}$ [8]. Note that the tensors $\mathcal{X}$ and $\mathcal{Y}$ are considered to be N samples of the tensor variate processes. To predict a single, scalar, realisation y from a single tensor observation $\mathcal{X}$ the function $f(\cdot\,;\,\cdot)$ is given in the form of an inner product $\langle\mathcal{X}, \mathcal{B}\rangle$ computed as $y = vec^{\mathrm{T}}(\mathcal{X})vec(\mathcal{B})$. The general tensor-variate regression (where the output is a tensor) is then expressed compactly through a mode-1 matrix unfolding of the tensors as

$$\mathbf{Y}_{(1)} = \mathbf{X}_{(1)}\mathbf{B}_{\langle M\rangle} \tag{19}$$

The regression coefficients, $\mathbf{B}_{\langle M\rangle}$, are computed through

$$\mathbf{B}_{\langle M\rangle} = \mathbf{X}_{(1)}^{+}\mathbf{Y}_{(1)} \tag{20}$$

which is akin to two-way multivariate regression in (2) and requires the generalised inverse of $\mathbf{X}_{(1)}$. The resultant matrices are then folded back into their tensor form.

2.4. Quaternion algebra and notation

Quaternions, $\mathbb{H}$, are both a skew-field over the real numbers, $\mathbb{R}$, whereby every element $g \in \mathbb{H}$ is a quadrivariate vector and an ordered pair of two complex numbers [22,23]. A quaternion scalar, $x \in \mathbb{H}$, is represented as

$$\begin{aligned} x &= x_r + x_i i + x_j j + x_k k\\ \mathrm{Re}(x) &= x_r\\ \mathrm{Im}(x) &= x_i i + x_j j + x_k k\end{aligned}$$

where i, j and k are the imaginary units governed by the multiplication scheme in Table 1. Observe that $ij = k \neq ji = -k$, and as a result, the multiplication of two quaternion numbers, x and y, is in general not commutative, that is

$$xy \neq yx$$

The modulus of a quaternion, x, is defined as

$$|x| = \sqrt{x_r^2 + x_i^2 + x_j^2 + x_k^2}$$

The quaternion conjugate operator negates the imaginary components

$$x = x_r + x_i i + x_j j + x_k k \qquad x^* = x_r - x_i i - x_j j - x_k k$$

Table 1
Quaternion multiplication.

$\times$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

[1] This uses a little endian ordering where the index $i_1i_2\cdots i_n = i_1 + (i_2-1)I_1 + (i_3-1)I_1I_2\cdots(i_n-1)I_1\cdots I_{n-1}$ [7].

analogous to the complex conjugate operator. The involution operators are unique to quaternions and are described as $\mathbf{x}^{\alpha} = -\alpha x \alpha$ which represents a rotation by π of x around α, where α is a pure unit quaternion, that is, a three-dimensional quaternion with a unit norm and has no real component [24]. These involutions have the property $(x^{\alpha})^{\alpha} = x$. Special instances of the quaternion involution around the imaginary axes, where $\alpha \in \{i, j, k\}$, are given as

$$\begin{aligned} x &= x_r + x_i i + x_j j + x_k k & x^i &= x_r + x_i i - x_j j - x_k k \\ x^j &= x_r - x_i i + x_j j - x_k k & x^k &= x_r - x_i i - x_j j + x_k k \end{aligned}$$

and are referred to as i-, j- and k-involutions. These lead to the α-conjugate operations frequently used in quaternion augmented statistics [22].

$$\begin{aligned} x &= x_r + x_i i + x_j j + x_k k & x^{i*} &= x_r - x_i i + x_j j + x_k k \\ x^{j*} &= x_r + x_i i - x_j j + x_k k & x^{k*} &= x_r + x_i i + x_j j - x_k \end{aligned}$$

The complete second-order statistics of a real vector-variate random process, $\mathbf{x} \in \mathbb{R}^m$, are captured by its mean, $E[\mathbf{x}]$, and its covariance matrix, $E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$. However, the mean and the direct extension of the covariance matrix, $E[\mathbf{x}\mathbf{x}^{\mathrm{H}}]$, are not sufficient for a full second-order description of a quaternion-valued vector-variate random process, $\mathbf{x} \in \mathbb{H}^m$ [22,23]. Instead, the consideration of the i-, j- and k-complementary covariance matrices $E[\mathbf{x}\mathbf{x}^{i\mathrm{H}}]$, $E[\mathbf{x}\mathbf{x}^{j\mathrm{H}}]$ and $E[\mathbf{x}\mathbf{x}^{k\mathrm{H}}]$ is also required in order to capture the full augmented quaternion statistics. This is an extension of augmented statistics for complex-valued data (where the complementary pseudocovariance matrix $E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$ is required in addition to the standard covariance) which is now well understood in the literature [25]. In order to conveniently account for the full second-order statistics, a quaternion-valued vector process can be represented in augmented form as

$$\underline{\mathbf{x}} = [\mathbf{x}, \mathbf{x}^i, \mathbf{x}^j, \mathbf{x}^k] \tag{21}$$

An important feature of Quaternion statistics is the notion of circularity which indicates whether all the four components are independent and have balanced powers [22]. Quaternion propriety specifically accounts for up to second-order moments/circularity [23]. The degree of properness is then given by the three circularity measures [26]

$$\rho_i = \frac{E[xx^{i*}]}{E[xx^{*}]} \qquad \rho_j = \frac{E[xx^{j*}]}{E[xx^{*}]} \qquad \rho_k = \frac{E[xx^{k*}]}{E[xx^{*}]} \tag{22}$$

which are generalisations of the circularity quotient for complex-valued data [27].

2.4.1. Quaternion widely linear estimation

In quaternion estimation, it is desired to estimate the output $\mathbf{Y} \in \mathbb{H}^{N\times p}$ from the input $\mathbf{X} \in \mathbb{H}^{N\times m}$ in the same way as the real-valued estimation problem in (1). For the quaternion case, however, the minimum mean square error (MMSE) estimate of the process in $\mathbf{Y}$ is instead linear in terms of $\mathbf{X}$, $\mathbf{X}^i$, $\mathbf{X}^j$ and $\mathbf{X}^k$ expressed as [22]

$$\hat{\mathbf{Y}} = \mathbf{X}\mathbf{G}_1 + \mathbf{X}^i\mathbf{G}_2 + \mathbf{X}^j\mathbf{G}_3 + \mathbf{X}^k\mathbf{G}_4 \tag{23}$$

In this way full second-order statistics are described. The above model is referred to as widely linear quaternion regression [23].

A compact form of the widely linear estimation in (23) can be obtained by representing the quaternion variable $\mathbf{X}$ in augmented form, $\underline{\mathbf{X}} \in \mathbb{H}^{N\times 4m}$, which yields

$$\hat{\mathbf{Y}} = \underline{\mathbf{X}}\underline{\mathbf{B}} \tag{24}$$

where $\underline{\mathbf{B}} \in \mathbb{H}^{4m\times p}$ is the vector of the regression coefficients. This enables the calculation of the regression coefficients analogously to the real-valued case in (2) as

$$\underline{\mathbf{B}} = \underline{\mathbf{X}}^{+}\mathbf{Y}$$

Note that if the data are circular then only the covariance matrix, $E[\mathbf{x}\mathbf{x}^{\mathrm{H}}]$, needs to be considered for processing and so the estimation (23) degenerates to the strictly linear form $\hat{\mathbf{Y}} = \mathbf{X}\mathbf{B}$.

2.4.2. The isomorphism between quaternion and $\mathbb{R}^4$

A quaternion data matrix, $\mathbf{X} \in \mathbb{H}^{N\times m}$ is described as $\mathbf{X} = \mathbf{X}_r + i\mathbf{X}_i + j\mathbf{X}_j + k\mathbf{X}_k$ where $\mathbf{X}_\eta \in \mathbb{R}^{N\times m}$ $\eta = \{r, i, j, k\}$. The isomorphism between the quaternion domain and a four-dimensional real vector space has been established, given through the transform matrix [22,23]

$$\mathbf{\Gamma}_m = \begin{bmatrix} \mathbf{I}_m & i\mathbf{I}_m & j\mathbf{I}_m & k\mathbf{I}_m \\ \mathbf{I}_m & i\mathbf{I}_m & -j\mathbf{I}_m & -k\mathbf{I}_m \\ \mathbf{I}_m & -i\mathbf{I}_m & j\mathbf{I}_m & -k\mathbf{I}_m \\ \mathbf{I}_m & -i\mathbf{I}_m & -j\mathbf{I}_m & k\mathbf{I}_m \end{bmatrix} \in \mathbb{H}^{4m\times 4m}, \tag{25}$$

which provides a link between the augmented data matrix, $\underline{\mathbf{X}} = [\mathbf{X}, \mathbf{X}^i, \mathbf{X}^j, \mathbf{X}^k] \in \mathbb{H}^{N\times 4m}$, and the real-valued matrix $\mathbf{X}_{Re} = [\mathbf{X}_r, \mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_k] \in \mathbb{R}^{N\times 4m}$, given by the relations

$$\mathbf{X}_{Re} = \frac{1}{4}\underline{\mathbf{X}}\mathbf{\Gamma}_m^{*}, \qquad \underline{\mathbf{X}} = \mathbf{X}_{Re}\mathbf{\Gamma}_m^{\mathrm{T}}. \tag{26}$$

This transform is, hence, invertible and is unitary up to a scale factor of 4.

2.4.3. Processing quaternion data in $\mathbb{R}^4$

The transform in (26) converts an augmented quaternion variable to a real-valued representation in $\mathbb{R}^4$. A general matrix product for a row vector $\mathbf{x} = [x_r, x_i, x_j, x_k] \in \mathbb{R}^{1\times 4}$ is given by

$$[y_r, y_i, y_j, y_k] = [x_r, x_i, x_j, x_k]\mathbf{H}.$$

where the entries of the matrix $\mathbf{H} \in \mathbb{R}^{4\times 4}$ can take any real value. The vector $[x_r, x_i, x_j, x_k]$ is equivalently represented as a quaternion scalar x, however, the quaternion product $y = xh$ represented in the real-domain is a special case of the matrix equation, $\mathbf{y} = \mathbf{x}\mathbf{H}$, where the matrix, $\mathbf{H}$, is restricted to the form [9]

$$\mathbf{H} = \begin{bmatrix} h_r & h_i & h_j & h_k \\ -h_i & h_r & -h_k & h_j \\ -h_j & h_k & h_r & -h_i \\ -h_k & -h_j & h_i & h_r \end{bmatrix} \tag{27}$$

where $[h_r, h_i, h_j, h_k]$ are the real and imaginary components of the quaternion number h.

Remark 1. An augmented quaternion variable can be equivalently processed in the real-domain, with both representations having the same degrees of freedom for processing. The quaternion algebra can therefore be viewed as a constrained matrix product in the real-domain indicating that the structure of a quaternion-valued estimation problem will be distinct from a generic real-valued problem [28].

3. Multidimensional PLS-regression algorithms

3.1. A NIPALS algorithm for tensor-variate PLS regression

Tensor-based PLS algorithms have already been developed such as Bro's N-Way PLS [16] and the higher order PLS (HOPLS) of Zhao et al. [17]. These produce a tensor regression between the input data tensor $\mathcal{X} \in \mathbb{R}^{N\times I_1\times I_2\times\cdots\times I_M}$ and the output data tensor $\mathcal{Y} \in \mathbb{R}^{N\times J_1\times J_2\times\cdots\times J_P}$, as introduced in Section 2.3.2. The HOPLS algorithm decomposes $\mathcal{X}$ and $\mathcal{Y}$ into a sum of r rank-$(1, K_1, K_2, \ldots, K_M)$ and rank-$(1, L_1, L_2, \ldots, L_P)$ components respectively. In accordance with (16) we can write

$$\mathcal{X} = \sum_{i=1}^{r} \mathcal{G}_i \times_1 \mathbf{t}_i \times_2 \mathbf{P}_i^{(1)} \times_3 \mathbf{P}_i^{(2)} \times_4 \cdots \times_M \mathbf{P}_i^{(M)}, \tag{28}$$

$$\mathcal{Y} = \sum_{i=1}^{r} \mathcal{D}_i \times_1 \mathbf{t}_i \times_2 \mathbf{Q}_i^{(1)} \times_3 \mathbf{Q}_i^{(2)} \times_4 \cdots \times_P \mathbf{Q}_i^{(P)},$$

where $\mathcal{G} \in \mathbb{R}^{1\times K_1,\times K_2\times\cdots\times K_M}$ and $\mathcal{D} \in \mathbb{R}^{1\times L_1,\times L_2\times\cdots\times L_P}$ are the core tensors of the Tucker and $\mathbf{P}^{(n)} \in \mathbb{R}^{K_n\times I_n}$ $n = 1, 2, \ldots, M$, $\mathbf{Q}^{(n)} \in \mathbb{R}^{L_n\times J_n}$ $n = 1, 2, \ldots, P$ are factor matrices. The derivation of the HO-PLS algorithm is given in Appendix A. On the other hand, the N-way PLS algorithm of Bro [16] produces a decomposition of the tensors $\mathcal{X}$ and $\mathcal{Y}$ whereby each factor matrix is a vector.

Both current tensor-variate PLS algroithms impose strict rules on the latent structure of the data. To that end, we introduce a general tensor-variate NIPALS style PLS-regression algorithm for the input data tensor, $\mathcal{X} \in \mathbb{R}^{N\times I_1\times I_2\times\cdots\times I_M}$, and the output data tensor, $\mathcal{Y} \in \mathbb{R}^{N\times J_1\times J_2\times\cdots\times J_P}$, which instead iteratively calculates any desired low-rank approximation of the original data tensors.

The first step of each iteration of a PLS algorithm is to calculate a cross-covariance structure. One method for tensor-variate data is to calculate the contraction product in (13) along the first mode of the input tensor, $\mathcal{X}$, and the output tensor, $\mathcal{Y}$, to give the structure

$$\mathcal{S} = \langle \mathcal{X}, \mathcal{Y} \rangle_{\{1,1\}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M\times J_1\times J_2\times\cdots\times J_P}. \tag{29}$$

Alternatively, the data tensors can be unfolded along the first mode, as is done in the HOSVD algorithm [21], which yields the matrix cross-covariance structure

$$\mathbf{S} = \mathbf{X}_{(1)}^{\mathsf{T}} \mathbf{Y}_{(1)} \in \mathbb{R}^{I_1 I_2\cdots I_M\times J_1 J_2\cdots J_P}. \tag{30}$$

Note that the mode-wise variations are encoded explicitly in the modes of the contraction product, $\mathcal{S}$. However, for the matrix, $\mathbf{S}$, obtained from the unfolded data tensors, they implicitly integrated.

The next step of the algorithm is to perform an eigen-type decomposition of the cross-covariance data structures in order to determine a spatial basis which describes the maximal cross-covariance between the inputs and the outputs. For the tensor $\mathcal{S}$ obtained from the cross-covariance structure given by the contraction product in (29), the eigen-type decomposition is then given by a rank-$(K_1, K_2, \ldots, K_M, L_1, L_2, \ldots, L_P)$ Tucker approximation

$$\mathcal{S} = \mathcal{G}^C \times_1 \mathbf{P}^{(1)} \times_2 \mathbf{P}^{(2)} \times_3 \cdots \times_M \mathbf{P}^{(M)} \times_{M+1} \mathbf{Q}^{(1)} \times_{M+2} \mathbf{Q}^{(2)} \times_{M+3} \cdots \times_{M+P} \mathbf{Q}^{(P)}. \tag{31}$$

This Tucker decomposition can be obtained using either the HOOI or HOSVD algorithms [21]. When the input and output variables yield a matrix result from (29), the decomposition is instead provided by an SVD.

Now consider the alternative case where the unfolded cross-covariance structure, $\mathbf{S}_{(1)}$, is implemented. The corresponding eigen-problem is then defined as

$$\mathbf{w} = \underset{||\mathbf{w}||=1}{\arg\max} ||\mathbf{w}^{\mathsf{T}} \mathbf{X}_{(1)}^{\mathsf{T}} \mathbf{Y}_{(1)}||_2^2. \tag{32}$$

The result is obtained from the leading eigenvector of the matrix $\mathbf{S}\mathbf{S}^{\mathsf{T}}$, that is, from the SVD of $\mathbf{S}$ in the form

$$\mathbf{S} = \mathbf{W}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{T}}. \tag{33}$$

The resulting PLS scores produced from either the decomposition of (29) or (30) can be represented as a matrix, $\mathbf{T}_{(1)} \in \mathbb{R}^{N\times K}$ where K is the desired number of first mode fibres, given by

$$\mathbf{T}_{(1)} = \mathbf{X}_{(1)} \mathbf{W}_{(1)} \in \mathbb{R}^{N\times K}. \tag{34}$$

For the unfolded cross-covariance data structure, $\mathbf{S}$ in (30), the matrix $\mathbf{W}_{(1)} \in \mathbb{R}^{I_1 I_2\ldots I_m\times K}$ is given by the first K left singular vectors of $\mathbf{S}$ from (33). On the other hand, to compute the scores from the cross-covariance structure $\mathcal{S}$ in (29), the matrix, $\mathbf{W}_{(1)}$, takes the form

$$\mathbf{W}_{(1)} = (\mathbf{P}^{(M)\mathsf{T}} \otimes \mathbf{P}^{(M-1)\mathsf{T}} \otimes \cdots \otimes \mathbf{P}^{(1)\mathsf{T}})^{\mathsf{T}}, \tag{35}$$

which is the Kronecker product unfolded form of the factor matrices obtained from the Tucker decomposition in (31), where $K = K_1 K_2 \ldots K_M$.

Remark 2. The tensor PLS scores are calculated as the mode-1 factor matrix, $\mathbf{T}_{(1)}$, for both choices of cross-covariance structure and their corresponding decompositions. This means that the choice of the structure of the score tensor $\mathcal{T}$ (from the folding of the mode-1 factor matrix $\mathbf{T}_{(1)}$) is flexible and depends on the form suggested by the data. For example, when the scores are calculated from the Tucker decomposition of the contraction product, $\mathcal{S}$ in (31), the rank of the decomposition gives the dimensions of the score tensor $\mathcal{T} \in \mathbb{R}^{N\times K_1\times K_2\times\cdots\times K_M}$. On the other hand, when the scores are calculated from the SVD in (33), the choice of folding is not intrinsic to the calculation.

The score tensor, $\mathcal{T}$, is now regressed to the data tensors, $\mathcal{X}$ and $\mathcal{Y}$, through the parameters $\mathcal{G}$ and $\mathcal{D}$ respectively, to give the relations

$$\mathcal{X} = f(\mathcal{T}; \mathcal{G}), \qquad \mathcal{Y} = f(\mathcal{T}; \mathcal{D}), \tag{36}$$

where the parameters $\mathcal{G}$ and $\mathcal{D}$ are calculated

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{T}_{(1)}^{+}, \qquad \mathcal{D} = \mathcal{Y} \times_1 \mathbf{T}_{(1)}^{+}. \tag{37}$$

Prior to the next iteration, the impact of the score tensor is removed through the deflation process

$$\mathcal{X}_{i+1} = \mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{T}_{(1),i}, \qquad \mathcal{Y}_{i+1} = \mathcal{Y}_i - \mathcal{D}_i \times_1 \mathbf{T}_{(1),i}, \tag{38}$$

where i denotes the current iteration and the tensors $\mathcal{X}_{i+1}$ and $\mathcal{Y}_{i+1}$ are used in place of $\mathcal{X}_i$ and $\mathcal{Y}_i$ in the next iteration. The sum of these tensor regressions provides the tensor version of the low-rank PLS approximations, akin to (4) and (6) in the two-way solution.

Notice that the factor matrices for the Tucker decomposition of the latent variable regression tensors $\mathcal{G}$ and $\mathcal{D}$ are not computed inherently as in HOPLS, where they are obtained from the decomposition of $\mathcal{S}$, and are not required for computation. Here, they are instead obtained by computing a Tucker decomposition (i.e. the HOSVD) for each regression tensor, $\mathcal{G}$ and $\mathcal{D}$ obtained in (37), to give the factor matrices[2] $\mathbf{P}_i^{(n)} \in \mathbb{R}^{K_n\times I_n}$ $n = 1, 2, \ldots, M$ and $\mathbf{C}_i^{(n)} \in \mathbb{R}^{K_n\times J_n}$ $n = 1, 2, \ldots, P$ respectively. This leads to the expressions

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{T}_{(1)} \times_2 \mathbf{P}^{(1)} \times_3 \mathbf{P}^{(2)} \times_4 \cdots \times_{M+1} \mathbf{P}^{(M)}, \tag{39}$$
$$\mathcal{Y} = \mathcal{D} \times_1 \mathbf{T}_{(1)} \times_2 \mathbf{C}^{(1)} \times_3 \mathbf{C}^{(2)} \times_4 \cdots \times_{P+1} \mathbf{C}^{(P)},$$

where $\mathcal{G}$ and $\mathcal{D}$ are now the core tensors within the Tucker decompositions and not the regression tensors. This is a more general decomposition than the HOPLS form (28).

We next obtain a further decomposition of the output data tensor $\mathcal{Y}$ that is analogous to the two-way PLS expression in (7). In this case, the impact of the unfolded latent variables $\mathbf{T}_{(1)}$ in the data tensor $\mathcal{Y}$ is found as

$$\mathbf{C} = \mathbf{T}_{(1)}^{+} \mathbf{Y}_{(1)}, \qquad \mathbf{U}_{(1)} = \mathbf{Y}_{(1)} \mathbf{C}^{\mathsf{T}}.$$

These derived latent variables are related to $\mathcal{Y}$ through the regression tensor, $\mathcal{E}$, calculated as

$$\mathcal{E} = \mathcal{Y} \times_1 \mathbf{U}_{(1)}^{+}.$$

As is the case for tensors, $\mathcal{G}$ and $\mathcal{D}$ in the latent variable regressions in (36), the tensor $\mathcal{E}$ is decomposed using the HOSVD to give the factor matrices $\mathbf{Q}_i^{(n)} \in \mathbb{R}^{K_n\times J_n}$ $n = 1, 2, \ldots, P$ leading to the relation

$$\mathcal{Y} = \mathcal{E} \times_1 \mathbf{U}_{(1)} \times_2 \mathbf{Q}^{(1)} \times_3 \mathbf{Q}^{(2)} \times_4 \cdots \times_{P+1} \mathbf{Q}^{(P)}. \tag{40}$$

[2] Note that these matrices are not the same as in (31).

This process allows the introduction of two algorithms: (i) The Higher Order NIPALS (HONIPALS) and (ii) The Generalised HOPLS (GHOPLS), summarised in Algorithms 2 and 3, respectively. The

Algorithm 2 The HONIPALS algorithm.

Input: $\mathcal{X}$, $\mathcal{Y}$, r, K
for $i = 1, \ldots, r$ **do**
  $\mathbf{S}_{(1)i} = \mathbf{X}_{(1),i}^{\mathrm{T}} \mathbf{Y}_{(1),i}$
  $\mathbf{W}_{(1)i} =$ first K left singular vectors of $\mathbf{S}_{(1),i}$
  $\mathbf{T}_{(1)i} = \mathbf{X}_{(1)i} \mathbf{W}_{(1)i}$
  $\mathcal{G}_i = \mathcal{X}_i \times_1 \mathbf{T}_{(1),i}^{+}$
  Perform HOSVD on $\mathcal{G}_i$ to find factor matrices $\mathbf{P}_i^{(n)}$ $n = 1, 2, \ldots, M$
  $\mathcal{D}_i = \mathcal{Y}_i \times_1 \mathbf{T}_{(1),i}^{+}$
  Perform HOSVD on $\mathcal{D}_i$ to find factor matrices $\mathbf{C}_i^{(n)}$ $n = 1, 2, \ldots, P$
  $\mathcal{X}_{i+1} = \mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{T}_{(1),i}$
  $\mathcal{Y}_{i+1} = \mathcal{Y}_i - \mathcal{D}_i \times_1 \mathbf{T}_{(1),i}$
  Store $\mathcal{G}_i$, $\mathcal{D}_i$, $\mathcal{T}_i$ and $\mathcal{W}_i$
end for

HONIPALS approach is based upon the unfolded cross-covariance structure in (30) and its SVD, whereas the GHOPLS stems from the contraction product in (29) and its Tucker decomposition. These algorithms demonstrate the range of tensor-based options available for each step of a NIPALS style PLS-regression algorithm. For each algorithm, the number of iterations, r, is chosen as the number of latent variable components required to well approximate the data tensors $\mathcal{X}$ and $\mathcal{Y}$, and this number is usually determined through cross-validation.

Remark 3. The solution obtained from the HONIPALS in Algorithm 2 degenerates to the two-way NIPALS in Algorithm 1 when the input and ouptut data tensors $\mathcal{X}$ and $\mathcal{Y}$ are given by the matrices $\mathbf{X} \in \mathbb{R}^{N \times M}$ and $\mathbf{Y} \in \mathbb{R}^{N \times P}$ and $K = 1$.

Remark 4. A symmetric Mode-A type PLS decomposition is also possible for both the HONIPALS and GHOPLS algorithms. This is achieved by replacing the approximation of $\mathcal{Y}$ in terms of the scores, $\mathcal{T}$, used in the deflation step in (38) and instead using the scores derived from the output $\mathcal{U}$ in (40).

Remark 5. The relations (36) are expressed in a general form of tensor regression and are therefore direct extensions of the two-way PLS relations (4) and (6). If desired, they can be expressed as a mode-1 product of the unfolded matrix $\mathbf{T}_{(1)}$ and the respective core tensors. Furthermore, by Remark 4 both algorithms can be used for different types of PLS, similarly to the two-way algorithm. Therefore, the GHOPLS and HONIPALS are both generalisations of the original 2-way NIPALS algorithm harnessing the available tensor-based operations.

Remark 6. The proposed GHOPLS shares the first two steps with the HOPLS algorithm of Zhao et. al. [17], that is, the calculation of the cross-covariance data structure and its decomposition. However, the HOPLS algorithm then only selects a vector component from this decomposition.

3.1.1. Calculation of the regularised tensor regression

The r approximations obtained in each iteration in (39) are summed up to yield the final PLS approximations for the HONIPALS and GHOPLS algorithms in the form

$$\tilde{\mathcal{X}} \approx \sum_{i=1}^{r} \mathcal{G}_i \times_1 \mathbf{T}_i, \qquad \tilde{\mathcal{Y}} \approx \sum_{i=1}^{r} \mathcal{D}_i \times_1 \mathbf{T}_i, \tag{41}$$

where $\mathcal{G}$ and $\mathcal{D}$ result from the tensor regression in (37). This latent variable decomposition can be used to produce a regularised tensor regression solution to predict $\tilde{\mathcal{Y}}$ from $\tilde{\mathcal{X}}$ through a regression coefficient tensor $\mathcal{B}$. To this end, their mode-1 unfolded versions are given as

$$\tilde{\mathbf{X}}_{(1)} = [\mathbf{T}_{(1),1}, \mathbf{T}_{(1),2}, \ldots, \mathbf{T}_{(1),r}][\mathbf{G}_{(1),1}, \mathbf{G}_{(1),2}, \ldots, \mathbf{G}_{(1),r}]^{\mathrm{T}},$$

$$\tilde{\mathbf{Y}}_{(1)} = [\mathbf{T}_{(1),1}, \mathbf{T}_{(1),2}, \ldots, \mathbf{T}_{(1),r}][\mathbf{D}_{(1),1}, \mathbf{D}_{(1),2}, \ldots, \mathbf{D}_{(1),r}]^{\mathrm{T}},$$

where $\mathbf{T}_{(1),i}$ denotes the unfolded version of the ith score tensor $\mathcal{T}$. Based on this, we can now obtain the prediction of $\tilde{\mathbf{Y}}_{(1)}$ from $\tilde{\mathbf{X}}_{(1)}$ as

$$\tilde{\mathbf{Y}}_{(1)} = \tilde{\mathbf{X}}_{(1)}[\mathbf{G}_{(1),1}, \mathbf{G}_{(1),2}, \ldots, \mathbf{G}_{(1),r}]^{+\mathrm{T}}[\mathbf{D}_{(1),1}, \mathbf{D}_{(1),2}, \ldots, \mathbf{D}_{(1),r}]^{\mathrm{T}}, \tag{42}$$

which yields the unfolded regression coefficient tensor, $\mathcal{B}$, in the form

$$\mathbf{B}_{\langle M \rangle} = [\mathbf{G}_{(1),1}, \mathbf{G}_{(1),2}, \ldots, \mathbf{G}_{(1),r}]^{+\mathrm{T}}[\mathbf{D}_{(1),1}, \mathbf{D}_{(1),2}, \ldots, \mathbf{D}_{(1),r}]^{\mathrm{T}} \tag{43}$$

This is analogous to the two-way NIPALS result in (10).

3.2. The quaternion NIPALS algorithm

In this section an extension of the real-valued NIPALS Algorithm 1 is introduced for quaternion-valued data. Previous work on PLS for quaternion-valued data can be found in Via et. al. [23] who shows that the PLS result should have a widely linear form. This PLS method is aimed at data block cross-correlation analysis and is a generalisation of PLS-SB described in [19]. As a result, the obtained decomposition is useful for an analysis of the directions of cross-covariance between the input and output blocks, but the components may not be orthogonal and it is not the most parsimonious for regression. The proposed extension of the NIPALS PLS algorithm for quaternion-valued data, on the other hand, produces a regularised PLS solution for the quaternion widely linear regression problem in (24) based on an orthogonal latent variable decomposition.

The proposed quaternion widely linear PLS (WL-QPLS) algorithm first calculates a vector component $\mathbf{t} \in \mathbb{H}^N$ from the input matrix $\mathbf{X} \in \mathbb{H}^{N \times m}$ which represents the maximal cross-covariance with the output matrix $\mathbf{Y} \in \mathbb{H}^{N \times n}$. The vector component $\mathbf{t}$ is given as

$$\mathbf{t} = \mathbf{X}\mathbf{w},$$

In order to find the vector, $\mathbf{w}$, the augmented versions of the input and output matrices are transformed to the real domain as $\mathbf{X}_{Re} \in \mathbb{R}^{N \times 4m}$ and $\mathbf{Y}_{Re} \in \mathbb{R}^{N \times 4m}$ through the isomorphism introduced in (25). The real-valued vector, $\mathbf{w}_{Re} \in \mathbb{R}^{4m}$, is found from the solution to the optimisation problem

$$\mathbf{w}_{Re} = \underset{||\mathbf{w}_{Re}||=1}{\arg\max} ||\mathbf{w}_{Re}^{\mathrm{T}} \mathbf{X}_{Re}^{\mathrm{T}} \mathbf{Y}_{Re}||_2^2. \tag{44}$$

which yields the leading eigenvector of $\mathbf{X}_{Re}^{\mathrm{T}} \mathbf{Y}_{Re} \mathbf{Y}_{Re}^{\mathrm{T}} \mathbf{X}_{Re}$. This is next cast back into the quaternion-domain through the relation

$$[\mathbf{w}^{\mathrm{T}}, \mathbf{w}^{i\mathrm{T}}, \mathbf{w}^{j\mathrm{T}}, \mathbf{w}^{k\mathrm{T}}]^{\mathrm{T}} = \mathbf{\Gamma}_m \mathbf{w}_{Re}. \tag{45}$$

In this way, the component chosen at each iteration of the WL-QPLS algorithm, $\mathbf{t}$, represents maximal joint second-order information between the $\mathbf{X}$ and $\mathbf{Y}$ blocks. The computation of $\mathbf{w}_{Re}$ in the real domain does not affect the quaternion representation [22,23] and is not equivalent to a real-valued NIPALS implementation, as was the case with the widely linear complex PLS (WL-CPLS) [15].

Following the calculation of the vector, $\mathbf{t}$, its relation to the $\mathbf{X}$ and $\mathbf{Y}$ blocks is found. This relationship is described by a quaternion widely linear regression given in augmented a form as

$$\underline{\mathbf{X}} = \underline{\mathbf{t}}\overline{\mathbf{p}}^{\mathrm{H}}, \qquad \underline{\mathbf{Y}} = \underline{\mathbf{t}}\overline{\mathbf{c}}^{\mathrm{H}}, \tag{46}$$

where the vectors $\overline{\mathbf{p}} = [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4]$ and $\overline{\mathbf{c}} = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4]$ represent the widely linear regression coefficients and are calculated as

$$\overline{\mathbf{p}} = (\underline{\mathbf{t}}^{+}\underline{\mathbf{X}})^{\mathrm{H}}, \qquad \overline{\mathbf{c}} = (\underline{\mathbf{t}}^{+}\underline{\mathbf{Y}})^{\mathrm{H}}. \tag{47}$$

To calculate the next PLS component, the matrices $\mathbf{X}$ and $\mathbf{Y}$ must be deflated, in order to remove the impact of the current score $\mathbf{t}$. The deflation is performed as

$$\underline{\mathbf{X}}_{i+1} = \underline{\mathbf{X}}_i - \underline{\mathbf{t}}_i\overline{\mathbf{p}}_i^{\mathrm{H}}, \qquad \underline{\mathbf{Y}}_{i+1} = \underline{\mathbf{Y}}_i - \underline{\mathbf{t}}_i\overline{\mathbf{c}}_i^{\mathrm{H}}, \tag{48}$$

where the subscript i represents the current iteration number. In the next iteration the matrices $\underline{\mathbf{X}}_{i+1}$ and $\underline{\mathbf{Y}}_{i+1}$ are used in place of $\underline{\mathbf{X}}_i$ and $\underline{\mathbf{Y}}_i$. As such, the latent variables are mutually orthogonal which leads to a parsimonious representation of the joint process in the input and output blocks.

Algorithm 3 The GHOPLS algorithm.

Input: $\mathcal{X}$, $\mathcal{Y}$, r, $K_1, K_2, \ldots, K_M$, $L_1, L_2, \ldots, L_P$
for $i = 1, \ldots, r$ **do**
  $\mathcal{S}_i = \langle \mathcal{X}_i, \mathcal{Y}_i \rangle_{\{1,1\}}$
  Perform rank-$(K_1, K_2, \ldots, K_M, L_1, L_2, \ldots, L_P)$ HOOI decomposition to give $\mathcal{S}_i = \mathcal{G}_i^C \times_1 \mathbf{P}_i^{(1)} \times_2 \mathbf{P}_i^{(2)} \times_3 \cdots \times_M \mathbf{P}_i^{(M)} \times_{M+1} \mathbf{Q}_i^{(1)} \times_{M+2} \mathbf{Q}_i^{(2)} \times_{M+3} \cdots \times_{M+P} \mathbf{Q}_i^{(P)}$
  $\mathbf{T}_{(1)} = \mathbf{X}_{(1)}(\mathbf{P}^{(M)T} \otimes \mathbf{P}^{(M-1)T} \otimes \cdots \otimes \mathbf{P}^{(1)T})^{\mathrm{T}}$
  $\mathcal{G}_i = \mathcal{X}_i \times_1 \mathbf{T}_{(1)i}^{+}$
  $\mathcal{D}_i = \mathcal{Y}_i \times_1 \mathbf{T}_{(1)i}^{+}$
  $\mathcal{X}_{i+1} = \mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{T}_{(1)i}$
  $\mathcal{Y}_{i+1} = \mathcal{Y}_i - \mathcal{D}_i \times_1 \mathbf{T}_{(1)i}$
  Store $\mathcal{G}_i$, $\mathcal{D}_i$, $\mathcal{T}_i$ and $\mathcal{W}_i$
end for

Algorithm 4 The NIPALS algorithm for widely linear quaternion PLS (WL-QPLS).

Initialise: $\underline{\mathbf{X}}_1 = [\mathbf{X}, \mathbf{X}^i, \mathbf{X}^j, \mathbf{X}^k]$, $\underline{\mathbf{Y}}_1 = [\mathbf{Y}, \mathbf{Y}^i, \mathbf{Y}^j, \mathbf{Y}^k]$
for $i = 1, \ldots, r$ **do**
  $\mathbf{X}_{i,Re} = \underline{\mathbf{X}}_i\boldsymbol{\Gamma}_m$, and $\mathbf{Y}_{i,Re} = \underline{\mathbf{Y}}_i\boldsymbol{\Gamma}_n$
  $\mathbf{w}_{i,Re} = \mathrm{Eig}_{\max}\{\mathbf{X}_{i,Re}^{\mathrm{T}}\mathbf{Y}_{i,Re}\mathbf{Y}_{i,Re}^{\mathrm{T}}\mathbf{X}_{i,Re}\}$
  $[\mathbf{w}_i^{\mathrm{T}}, \mathbf{w}_i^{iT}, \mathbf{w}_i^{jT}, \mathbf{w}_i^{kT}]^{\mathrm{T}} = \boldsymbol{\Gamma}_m\mathbf{w}_{i,Re}$
  $\mathbf{t}_i = \mathbf{X}_i\mathbf{w}_i$, $\underline{\mathbf{t}}_i = [\mathbf{t}_i, \mathbf{t}_i^i, \mathbf{t}_i^j, \mathbf{t}_i^k]$
  $\overline{\mathbf{c}_i} = (\underline{\mathbf{t}}_i^{+}\ \underline{\mathbf{Y}}_i)^{\mathrm{H}}$
  $\overline{\mathbf{p}_i} = (\underline{\mathbf{t}}_i^{+}\ \underline{\mathbf{X}}_i)^{\mathrm{H}}$
  $\underline{\mathbf{X}}_{i+1} = \underline{\mathbf{X}}_i - \underline{\mathbf{t}}_i\overline{\mathbf{p}}_i^{\mathrm{H}}$, $\underline{\mathbf{Y}}_{i+1} = \underline{\mathbf{Y}}_i - \underline{\mathbf{t}}_i\overline{\mathbf{c}}_i^{\mathrm{H}}$
  Store $\mathbf{t}_i$, $\overline{\mathbf{p}}_{1,i}$, $\overline{\mathbf{c}}_i$ and $\mathbf{w}_i$
end for

The full WL-QPLS procedure is summarised in Algorithm 4. After every iteration, the vectors $\mathbf{t}$ are stored into the columns of the matrices $\mathbf{T}$, while $\overline{\mathbf{p}}$ and $\overline{\mathbf{c}}$ are stored as $\overline{\mathbf{P}} = [\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4]$ and $\overline{\mathbf{C}} = [\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4]$ which leads to the widely linear relationships

$$\underline{\tilde{\mathbf{X}}} = \underline{\mathbf{T}}\overline{\mathbf{P}}^{\mathrm{H}}, \qquad \underline{\tilde{\mathbf{Y}}} = \underline{\mathbf{T}}\overline{\mathbf{C}}^{\mathrm{H}}. \tag{49}$$

Therefore, the input $\mathbf{X}$ and output $\mathbf{Y}$ are both represented in terms of the new latent variables $\mathbf{T}$. This provides a natural representation of the joint process and offers a useful decomposition to calculate the regression coefficients, $\underline{\mathbf{B}}$, for the widely linear regression

$$\underline{\tilde{\mathbf{Y}}} = \underline{\tilde{\mathbf{X}}}\underline{\mathbf{B}}. \tag{50}$$

This is achieved through

$$\underline{\mathbf{B}} = \underline{\tilde{\mathbf{X}}}^{+}\underline{\tilde{\mathbf{Y}}}.$$

The calculation of the regression coefficients based on the WL-QPLS model means that only shared information is included in the calculation.

In addition to the relations (49), an equivalent version of the real-valued PLS relationship (7) can be found, whereby the score within the data block $\mathbf{Y}$ is calculated as

$$\mathbf{u} = \mathbf{Y}\mathbf{c}_1 + \mathbf{Y}^i\mathbf{c}_2 + \mathbf{Y}^j\mathbf{c}_3 + \mathbf{Y}^k\mathbf{c}_4,$$

with the respective loadings, from the augmented scores $\underline{\mathbf{u}}$, given by

$$\overline{\mathbf{q}} = (\underline{\mathbf{u}}^{+}\underline{\mathbf{Y}})^{\mathrm{H}}.$$

4. Analysis of the MD-PLS algorithms

Having introduced a class of MD-PLS algorithms, including two tensor-variate and one quaternion-valued, this section provides a comparative analysis to contrast each algorithm.

4.1. Analysis of the HONIPALS and GHOPLS cross-covariance structures

The difference between the HONIPALS and GHOPLS algorithms is in the respective cross-covariance structures and the subsequent deflations. In this section, we further elaborate on the consequences of these alternative choices.

Consider the tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_n \times \cdots \times J_P}$ with $I_1 = J_1$. The mode-1 contraction product used by the GHOPLS algorithm is given as $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y} \rangle_{\{1,1\}} \in \mathbb{R}^{I_2 \times I_3 \times \cdots \times I_M \times J_2 \times \cdots J_P}$. On the other hand, the HONIPALS uses the product of the mode-1 unfolded data tensors in the form $\mathbf{S} = \mathbf{X}_{(1)}^{\mathrm{T}}\mathbf{Y}_{(1)} \in \mathbb{R}^{I_2 I_3 \ldots I_M \times J_2 J_3 \ldots J_P}$. Observe that these share the same elements, that is

$$\sum_{i_1 = j_1 = 1}^{I_1} x_{(i_1, i_2, \ldots, i_M)} y_{(j_1, j_2, \ldots, j_P)} = \mathbf{X}_{(1)}[:, i_2 i_3 \ldots i_M]^{\mathrm{T}}\mathbf{Y}_{(1)}[:, j_2 j_3 \ldots j_P].$$

as seen from the definition of the contraction product in (13). The link is provided through the canonical matrix unfolding in (15), whereby the mode-$(M-1)$ canonical unfolding of the tensor $\mathcal{S}$ yields the matrix $\mathbf{S}_{\langle M-1 \rangle} \in \mathbb{R}^{I_2 I_3 \ldots I_M \times J_2 J_3 \ldots J_P}$ which is the same as the matrix $\mathbf{S} = \mathbf{X}_{(1)}^{\mathrm{T}}\mathbf{Y}_{(1)}$.

As the two cross-covariance structures contain the same elements, intuitively, they share the same information. However, the Tucker decomposition of the contraction product form will yield a different result to the SVD of the unfolded cross-covariance matrix, that is, the matrices, $\mathbf{W}_{(1)}$, derived from each decomposition select a different unfolded score matrix as $\mathbf{T}_{(1)} = \mathbf{X}_{(1)}\mathbf{W}_{(1)}$.

Proposition 1. *Consider the tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_n \times \cdots \times J_P}$ with $I_1 = J_1$. The mode-$(M-1)$ canonical unfolding of the Tucker decomposition of the tensor $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y} \rangle_{\{1,1\}}$ (obtained from the mode-1 contraction product) does not yield the equivalent result as the SVD of the matrix $\mathbf{S}_{\langle M-1 \rangle}$, obtained as $\mathbf{S} = \mathbf{X}_{(1)}^{\mathrm{T}}\mathbf{Y}_{(1)} = \mathbf{W}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$.*

Proof. The tensor given by $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y} \rangle_{\{1,1\}}$ admits a Tucker decomposition

$$\mathcal{S} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_{M+P-2} \mathbf{A}^{(M+P-2)},$$

where $\mathcal{G} \in \mathbb{R}^{K_1 \times K_2 \times \cdots \times K_{M+P-2}}$ and $\mathbf{A}^{(i)} \in \mathbb{R}^{K_i \times K_i}$ with $K_i = I_{i+1}$ for $i < M$ and $K_i = J_{i+1}$ for $i \geq M$. The Tucker decomposition of the tensor $\mathcal{S}$ is computed through the HOOI or HOSVD algorithms so that

the factor matrices $\mathbf{A}^{(i)}$, and hence their Kronecker products, are orthogonal. The mode-$(M-1)$ canonical unfolding is then given as [20]

$$\mathcal{S}_{\langle M-1\rangle} = (\mathbf{A}^{(M-1)} \otimes \mathbf{A}^{(M-2)} \otimes \cdots \otimes \mathbf{A}^{(1)})\mathbf{G}_{\langle M-1\rangle} (\mathbf{A}^{(M+P-2)} \otimes \mathbf{A}^{(M+P-3)} \otimes \cdots \otimes \mathbf{A}^{(M)})^{\mathsf{T}}.$$

This yields the left and right multiplication of the unfolded core tensor $\mathbf{G}_{\langle M-1\rangle}$ by two unitary matrices. However, the matrix $\mathbf{G}_{\langle M-1\rangle}$ is in general not diagonal which means that the result is not generally equivalent to the SVD of the matrix $\mathbf{S}_{\langle M-1\rangle} = \mathbf{X}_{(1)}^{\mathsf{T}}\mathbf{Y}_{(1)} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$, where the matrix $\mathbf{\Sigma}$ is diagonal. □

Remark 7. The cross-covariance structures for the HONIPALS, $\mathbf{S}_{(1)} = \mathbf{X}_{(1)}^{\mathsf{T}}\mathbf{Y}_{(1)}$, and GHOPLS, $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}\rangle_{\{1,1\}}$, algorithms have the same entries, linked by a canonical mode-1 unfolding, and thus, they share the same information. However, according to Proposition 1, the Tucker decomposition in (31) which is employed within the GHOPLS algorithm is in general not equivalent to the SVD in (33) used for HONIPALS. As a result, the unfolded score matrices, $T_{(1)}$ in (34), will not be the same in both algorithms.

4.2. The duality between tensor and quaternion PLS

A quadrivariate random variable is readily represented by either a tensor or quaternion data structure. In this section we show how all proposed algorithms process such data equivalently, but in an alternative manner. As a prerequisite, the link between a quaternion variable and its tensor counterpart is first formalised through the derivation of an isomorphism.

4.2.1. The isomorphism between quaternions and tensors

Consider a matrix of quaternion variables, $\mathbf{X}$, which can also be represented as a tensor, $\mathcal{X} \in \mathbb{R}^{N\times 4\times m}$, where each mode-2 slice is constructed as

$$\begin{aligned} \mathcal{X}_{(:,1,:)} &= \mathbf{X}_r, \qquad \mathcal{X}_{(:,2,:)} = \mathbf{X}_i, \\ \mathcal{X}_{(:,3,:)} &= \mathbf{X}_j, \qquad \mathcal{X}_{(:,4,:)} = \mathbf{X}_k. \end{aligned} \tag{51}$$

Hence, the frontal slices, $\mathcal{X}_{(:,:,m)} \in \mathbb{R}^{N\times 4}$ for $m = 1, 2, \ldots, m$, are matrices that contain the four components of a quaternion vector. This can be regarded as an extension of the isomorphism in (25), $\mathbf{\Gamma}_m$, which yields a quaternion to tensor transform. To this end, we shall re-write (26), $\mathbf{X}_{Re} = \frac{1}{4}\underline{\mathbf{X}}\mathbf{\Gamma}_m^*$, as

$$\mathbf{X}_{(1)} = \underline{\mathbf{X}}\mathbf{\Gamma}^*\mathbf{\Phi}, \tag{52}$$

where $\mathbf{X}_{(1)} \in \mathbb{R}^{N\times 4m}$ indicates the mode-1 unfolding of the tensor representation of the augmented quaternion-valued matrix $\underline{\mathbf{X}}$. The matrix $\mathbf{\Phi}$ is a permutation matrix which re-orders the columns of the matrix, $\mathbf{X}_{Re}$, to group the four quaternion vector components, $\mathbf{x}_r$, $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$, of each variable into consecutive columns, that is

$$\mathbf{X}_{(1)} = \mathbf{X}_{Re}\mathbf{\Phi}. \tag{53}$$

Through the right multiplication of (52) by an identity matrix $\mathbf{I}_{(1)} \in \mathbb{R}^{4m\times 4m}$, which is also considered to be in a mode-1 unfolded state we obtain the relation

$$\mathbf{X}_{(1)} = (\underline{\mathbf{X}}\mathbf{\Gamma}^*\mathbf{\Phi})\mathbf{I}_{(1)} \quad \Leftrightarrow \quad \mathcal{X} = \mathcal{I} \times_1 (\underline{\mathbf{X}}\mathbf{\Gamma}^*\mathbf{\Phi}), \tag{54}$$

where $\mathcal{I} \in \mathbb{R}^{4m\times 4\times m}$ is constructed from the $4m \times 4m$ dimensional identity matrix folded such that each mode-3 slice has a 4×4 identity matrix shifted down by 4 from the previous slice. The full quaternion to tensor transform is then given by

$$\mathcal{X} = \mathcal{A} \times_1 \underline{\mathbf{X}}, \tag{55}$$

where

$$\mathcal{A} = \mathcal{I} \times_1 \mathbf{\Gamma}^*\mathbf{\Phi} \in \mathbb{H}^{4m\times 4\times m}, \tag{56}$$

is the complete transform.

The mapping back into the quaternion domain is performed as

$$\underline{\mathbf{X}} = \mathbf{X}_{(1)}\mathbf{A}_{(1)}^{\mathsf{H}}. \tag{57}$$

4.2.2. WL-QPLS as a tensor-variate PLS

The link between a quaternion-valued and tensor-based PLS solution is described through the quaternion to tensor transform in (56). Consider the case where it is desired to calculate a PLS-regression between the quaternion-valued input data matrix, $\mathbf{X} \in \mathbb{H}^{N\times m}$, and the quaternion-valued output data matrix, $\mathbf{Y} \in \mathbb{H}^{N\times p}$. Using the transform $\mathcal{A}$ constructed in (56), we can express the matrices, $\mathbf{X}$ and $\mathbf{Y}$, as third order tensors

$$\mathcal{X} = \mathcal{A}_m \times_1 \underline{\mathbf{X}} \qquad \mathcal{Y} = \mathcal{A}_p \times_1 \underline{\mathbf{Y}}, \tag{58}$$

where $\mathcal{X} \in \mathbb{R}^{N\times 4\times m}$ and $\mathcal{Y} \in \mathbb{R}^{N\times 4\times p}$. These data tensors can be processed by the tensor-valued HOPLS or the proposed GHOPLS and HONIPALS algorithms to compute an alternative PLS solution.

Each iteration of the WL-QPLS in Algorithm 4 computes a widely linear regression between a vector score, $\mathbf{t}_i$, and the input and output blocks $\mathbf{X}_i$ and $\mathbf{Y}_i$. The equivalent regression (with the same degrees of freedom) in the real-domain requires the quaternion vector, $\mathbf{t}$, to be represented as an $N\times 4$ matrix, $\mathbf{t}_{Re} = [\mathbf{t}_r, \mathbf{t}_i, \mathbf{t}_j, \mathbf{t}_k]$, produced by the transform (26). Therefore, the score of an equivalent tensor PLS solution would also be an $N\times 4$ matrix. This score can be computed by the HONIPALS and GHOPLS in Algorithm 2 (with $M = 4$) and Algorithm 3 (with $K_1 = L_1 = 4$ and $K_2 = L_2 = 1$) respectively, however, the HOPLS [17] cannot produce an alternative solution as it only calculates a vector score component. Therefore, for the same data represented as quaternion or tensor-variate, which are related through the transform (58), all the three algorithms, WL-QPLS, GHOPLS and HONIPALS, have enough degrees of freedom to provide a comparable regression result.

4.2.3. Comparison of the WL-QPLS and HONIPALS cross-covariance structures

The quaternion-valued data matrices $\mathbf{X} \in \mathbb{H}^{N\times m}$ and $\mathbf{Y} \in \mathbb{H}^{N\times p}$ can be transformed, through (58), to the data tensors, $\mathcal{X} \in \mathbb{R}^{N\times 4\times m}$ and $\mathcal{Y} \in \mathbb{R}^{N\times 4\times m}$, which, as stated in Section 4.2.2, allow alternative PLS solutions using the WL-QPLS, HONIPALS and GHOPLS algorithms. The data inputs are related as

$$\mathbf{X}_{(1)} = \mathbf{X}_{Re}\mathbf{\Phi}_{\mathbf{X}}, \qquad \mathbf{Y}_{(1)} = \mathbf{Y}_{Re}\mathbf{\Phi}_{\mathbf{Y}},$$

where $\mathbf{X}_{(1)}$ and $\mathbf{Y}_{(1)}$ are the mode-1 unfolded data tensors, $\mathbf{X}_{Re}$ and $\mathbf{Y}_{Re}$ denote the transformed real-valued data matrices through (26) and $\mathbf{\Phi}_{\mathbf{X}}$ and $\mathbf{\Phi}_{\mathbf{Y}}$ are the permutation matrices of the required dimensions needed for the quaternion to tensor transform (56).

Observe that the WL-QPLS algorithm in (45) calculates the cross-covariance structure in the real-domain, given by the matrix $\mathbf{X}_{Re}^{\mathsf{T}}\mathbf{Y}_{Re}$, before transforming back to the quaternion-domain. On the other hand, the HONIPALS algorithm creates the cross-covariance structure from the mode-1 unfolded data tensors, as $\mathbf{S} = \mathbf{X}_{(1)}^{\mathsf{T}}\mathbf{Y}_{(1)}$. These cross-covariance structures are therefore related as

$$\mathbf{S}_{(1)} = \mathbf{\Phi}_{\mathbf{X}}^{\mathsf{T}}\mathbf{S}_{Re}\mathbf{\Phi}_{\mathbf{Y}},$$

while the corresponding SVDs are related as

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}} = \mathbf{\Phi}_{\mathbf{X}}^{\mathsf{T}}\mathbf{U}_{Re}\mathbf{\Sigma}\mathbf{V}_{Re}^{\mathsf{T}}\mathbf{\Phi}_{\mathbf{Y}}.$$

The leading four left singular vectors are chosen for the basis of the HONIPALS score vectors $\mathbf{T}_{(1)}$ in (34) when used to provide an alternative to the quaternion-valued solution. This basis for the HONIPALS algorithm is therefore the same as that calculated in the real-domain for the WL-QPLS, as they are related by

$$\mathbf{T}_{(1)} = \mathbf{X}_{Re}\mathbf{\Phi}_{\mathbf{X}}(\mathbf{\Phi}_{\mathbf{X}}^{\mathsf{T}}\mathbf{U}_{Re})[:, 1:4].$$

However, to calculate the quaternion-valued score vector, $\mathbf{t}$, only the first left singular vector, $\mathbf{U}_{Re}[:, 1]$, is transformed back into the

Table 2
Comparison of the multidimensional PLS algorithms.

Algorithm Step	WL-QPLS	GHOPLS	HONIPALS
Cross-covariance structure	$\mathbf{S}_{i,Re} = \mathbf{X}_{i,Re}^{\mathrm{T}}\mathbf{Y}_{i,Re}$ Transformed to the real-domain	$\mathcal{S}_i = \langle \mathcal{X}_i, \mathcal{Y}_i \rangle_{\{1,1\}}$ Tensor-based contraction product	$\mathbf{S}_i = \mathbf{X}_{(1),i}^{\mathrm{T}}\mathbf{Y}_{(1),i}$ Unfolded to matrices
Eigen-decomposition	$\mathbf{w}_{i,Re} = \mathrm{Eig}_{\max}\{\mathbf{S}_{i,Re}\mathbf{S}_{i,Re}^{\mathrm{T}}\}$ Left singular vectors of $\mathbf{S}_{i,Re}$	HOOI/HOSVD decomposition of $\mathcal{S}_i$	$\mathbf{W}_{(1)i} = \mathrm{Eig}_{\text{K-max}}\{\mathbf{S}_i\mathbf{S}_i^{\mathrm{T}}\}$ First K left singular vectors of $\mathbf{S}_i$
Score calculation	$\mathbf{t}_i = \mathbf{X}_i\mathbf{w}_i$ The projection of $\mathbf{X}_i$	$\mathbf{T}_{(1)} = \mathbf{X}_{(1)}(\mathbf{P}^{(M)\mathrm{T}} \otimes \mathbf{P}^{(M-1)\mathrm{T}} \otimes \cdots \otimes \mathbf{P}^{(1)\mathrm{T}})$ The projection of unfolded $\mathcal{X}_i$	$\mathbf{T}_{(1)i} = \mathbf{X}_{(1)i}\mathbf{W}_i$ The projection of unfolded $\mathcal{X}_i$
Deflation	$\underline{\mathbf{X}}_{i+1} = \underline{\mathbf{X}}_i - \underline{\mathbf{t}}_i\bar{\mathbf{p}}_i^{\mathrm{H}}$ $\underline{\mathbf{Y}}_{i+1} = \underline{\mathbf{Y}}_i - \underline{\mathbf{t}}_i\bar{\mathbf{c}}_i^{\mathrm{H}}$	$\mathcal{X}_{i+1} = \mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{T}_{(1)i}$ $\mathcal{Y}_{i+1} = \mathcal{Y}_i - \mathcal{D}_i \times_1 \mathbf{T}_{(1)i}$	$\mathcal{X}_{i+1} = \mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{T}_{(1)i}$ $\mathcal{Y}_{i+1} = \mathcal{Y}_i - \mathcal{D}_i \times_1 \mathbf{T}_{(1)i}$

quaternion domain, to the vector $\mathbf{w}$ in (45). Moreover, to cast the quaternion multiplication of the score computation, $\mathbf{t} = \mathbf{X}\mathbf{w}$, to the real-domain, the constrained matrix multiplication (27) is required.

Remark 8. The cross-covariance structures for the proposed HONIPALS and WL-QPLS algorithms are linked through a permutation matrix which means that they share the same eigendecomposition computation. The HONIPALS uses the four leading singular vectors to calculate the scores from the unfolded data matrix $\mathbf{X}_{(1)}$. On the other hand, the WL-QPLS uses only the first singular vector of the real-valued cross-covariance structure which is folded back to the quaternion-domain prior to the calculation of the score vector. This operation performed equivalently in the real-domain requires a constrained matrix multiplication. As a result, the scores and the subspace identified in every iteration of the WL-QPLS and HONIPALS algorithms are not the same.

4.3. Comparison of the core components of the multidimensional PLS algorithms

The key characteristics of the NIPALS PLS in Algorithm 1 are that it is iterative, eigenvector-centric and based on capturing the joint second-order information. Each iteration consists of four major steps:

1. Finding the cross-covariance structure,
2. The eigendecomposition of the cross-covariance structure,
3. The calculation of the score,
4. The latent variable decomposition and deflation.

The choice of how these steps are computed dictates the qualities of the solution and therefore the goal of the implementation. For example, PLS Mode-A changes the deflation step to $\mathbf{Y}_{i+1} = \mathbf{Y}_i - \mathbf{u}_i\mathbf{q}_i^{\mathrm{T}}$ for use in correlation analysis applications.

It has been shown that the three algorithms WL-QPLS, GHOPLS and HONIPALS are generalisations of the two-way NIPALS algorithm for performing PLS-regression on higher-order multidimensional data. The choices of the core components with respect to the NIPALS PLS, are summarised in Table 2.

5. Simulations and applications

5.1. Simulation of WL-QPLS and tensor-variate PLS

A simulation on synthetic data is now presented to further examine the WL-QPLS, HONIPALS and GHOPLS algorithms for multidimensional regression. An improper quaternion-valued data matrix, $\mathbf{X} \in \mathbb{H}^{N\times m}$ ($N = 1000$ and $m = 10$), was generated from a mixture of $r = 5$ independent improper sources. Each quaternion axis q_r, q_i, q_j, q_k was an i.i.d. source, sampled from the distribution as $q_n \sim \mathcal{N}(0, 1)$ for $n = r, i, j, k$. The output data matrix, $\mathbf{Y} \in \mathbb{H}^{N\times p}$, was then calculated as $\mathbf{Y} = \underline{\mathbf{X}}\mathbf{B}$, where $\underline{\mathbf{X}} \in \mathbb{H}^{N\times 4m}$ is in the augmented form and $\mathbf{B} \in \mathbb{H}^{4m\times p}$ are the quaternion-valued regression coefficients, where $p = 10$. A noise matrix, $\mathbf{N_Y} \in \mathbb{H}^{N\times p}$, was then created where each quaternion axis was sampled from the distribution $q_n \sim \mathcal{N}(0, \sigma^2)$ for $n = r, i, j, k$, while σ^2 was varied to give a range of SNRs calculated as

$$SNR = 10\log_{10}\frac{\mathrm{Tr}\{E[\mathbf{Y}^{\mathrm{H}}\mathbf{Y}]\}}{\mathrm{Tr}\{E[\mathbf{N}_{\mathbf{Y}}^{\mathrm{H}}\mathbf{N}_{\mathbf{Y}}]\}}.$$

The matrices $\mathbf{X}$ and $\mathbf{Y}$ were transformed to tensors using the quaternion to tensor transform in (56) to give the data tensors $\mathcal{X} \in \mathbb{R}^{N\times 4\times m}$ and $\mathcal{Y} \in \mathbb{R}^{N\times 4\times p}$. The widely linear and tensor regression estimates were calculated using the WL-QPLS, GHOPLS and HONIPALS algorithms for the same data and for a range of components. The tensor GHOPLS and HONIPALS, outlined respectively in Algorithm 3 and Algorithm 2, were implemented to calculate rank-(N, 4, 1) scores. The results were transformed back into the quaternion domain to calculate the error as

$$MSE = \frac{|\mathbf{Y} - \hat{\mathbf{Y}}|^2}{\mathrm{Tr}\{E[\mathbf{Y}^{\mathrm{H}}\mathbf{Y}]\}}, \tag{59}$$

where $\hat{\mathbf{Y}}$ are the estimated dependent variables. The results are shown in Fig. 1 for three different SNRs and include the results for training data and the ensemble average of the estimate on test data, for an ensemble with 50 members. The right hand plots compare the MSE (on the training data) for each algorithm for the range of SNR. For more than $r = 5$ PLS components, the error for the test data degraded for all the algorithms as they were over-fitting. For a general application, the structure of the data within this representation must be considered in order to indicate the most appropriate algorithm.

5.2. Application for quaternion covariance matrix diagonalisation

Applications which invlove covariance matrix diagonalisation are ubiquitous in real-valued signal processing [29]. To that end, the WL-QPLS algorithm can be implemented as a tool for the diagonalisation of the quaternion empirical covariance matrices in Section 2.4, $\mathbf{X}^{\mathrm{H}}\mathbf{X}$, $\mathbf{X}^{i\mathrm{H}}\mathbf{X}$, $\mathbf{X}^{j\mathrm{H}}\mathbf{X}$ and $\mathbf{X}^{k\mathrm{H}}\mathbf{X}$. This result is analogous to that achieved by the WL-CPLS for complex-valued data [15].

In Section 2.4.1 it was shown that to fully cater for quaternion-valued second-order statistics, the four covariance matrices $E[\mathbf{X}^{\mathrm{H}}\mathbf{X}]$, $E[\mathbf{X}^{i\mathrm{H}}\mathbf{X}]$, $E[\mathbf{X}^{j\mathrm{H}}\mathbf{X}]$ and $E[\mathbf{X}^{k\mathrm{H}}\mathbf{X}]$ must be considered. To that end, quaternion covariance matrix diagonalisation requires all four matrices to be simultaneously diagonalised as well as the traditional covariance matrix. An extension of PCA for quaternion-valued data, QPCA, has been developed in [30,31], and was obtained from the quaternion SVD of the quaternion covariance matrix,

$$\mathbf{X}^{\mathrm{H}}\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{H}} \qquad \mathbf{X} = \mathbf{T}\mathbf{U}^{\mathrm{H}},$$

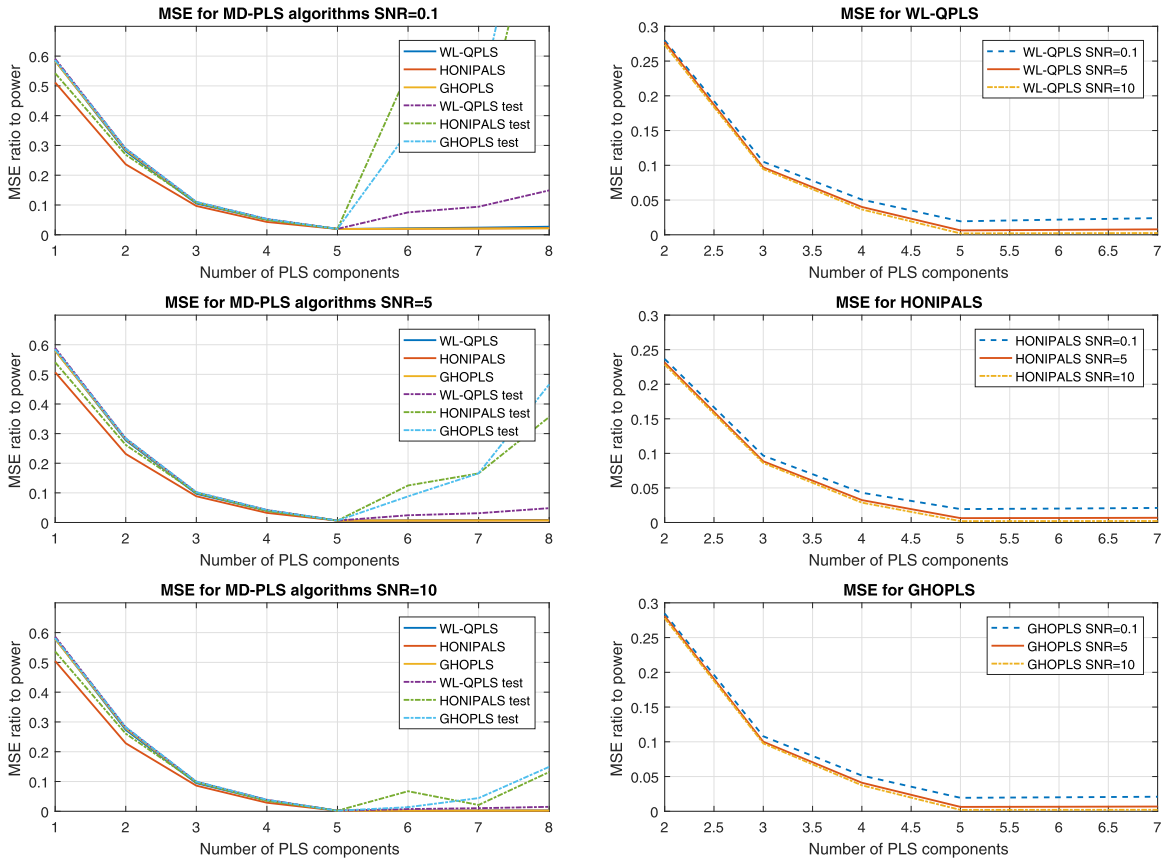

Fig. 1. The MSEs of widely linear and tensor regression estimates from the proposed MD-PLS algorithms, for a range of SNRs and different numbers of PLS components. The MSE was calculated as a ratio of the power in the dependent variables.

as a direct extension of the real-valued PCA. However, in this way the full augmented statistics are not considered and so the components are not fully uncorrelated. Xiang et al. [32] show that an extension of the complex-valued SUT for quaternions, the QUT, can only diagonalise the quaternion covariance matrix and one complementary covariance matrix at a time. Furthermore, the quaternion approximate uncorrelating transform (QAUT) can *approximately* diagonalise the covariance matrix along with the three complementary covariance matrices.

Owing to the ability of the WL-QPLS to calculate orthogonal latent-variables, $\mathbf{T}$, it consequently produces a decomposition of the input variables, $\mathbf{X}$, which has diagonalised all the quaternion covariance matrices. To resolve this outstanding isssue, notice that the WL-QPLS algorithm can be implemented with both the input and output being the same matrix, $\mathbf{X}$, with the aim of producing a diagonalising transform. This diagonalisation is given by

$$\underline{\mathbf{T}} = \underline{\mathbf{X}}\boldsymbol{\Theta}^{+},$$

where the transform matrix $\boldsymbol{\Theta}$ takes the form

$$\boldsymbol{\Theta} = \begin{pmatrix} (\mathbf{P}_1^{\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_2^{i\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_3^{j\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_4^{k\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} \\ (\mathbf{P}_2^{\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_1^{i\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_4^{j\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_3^{k\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} \\ (\mathbf{P}_3^{\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_4^{i\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_1^{j\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_2^{k\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} \\ (\mathbf{P}_4^{\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_3^{i\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_2^{j\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} & (\mathbf{P}_1^{k\mathrm{H}}\mathbf{W})\mathbf{W}^{\mathrm{H}} \end{pmatrix}. \tag{60}$$

Remark 9. The WL-QPLS in Algorithm 4 with the input and output blocks given by the quaternion-valued matrix $\mathbf{X}$ yields a score matrix $\mathbf{T}$ which is fully uncorrelated in terms of quaternion-valued second-order statistics. The augmented covariance matrix, $\underline{\mathbf{T}}^{\mathrm{H}}\underline{\mathbf{T}}$, of the WL-QPLS scores is now block diagonal which means that all four covariance matrices are diagonalised.

Remark 10. If, in each iteration of the WL-QPLS Algorithm 4, the score vector $\mathbf{t}$ is normalised, then the complementary covariance matrices of the score matrix, $\mathbf{T}$, will yield their circularity quotients in (22).

5.3. Application of the MD-PLS algorithms for image classification

The GHOPLS, HONIPALS and WL-QPLS were implemented for the real-world problem of colour image classification, a case where the data are naturally mutlidimensional. To do so we employed the quaternion-based framework developed by Risojević and Babić [33,34] and implemented the MD-PLS algorithms as an alternative pre-processing step.

This methodology was applied to the Brazilian Coffee Scenes dataset [35], a selection of satellite images in the near infrared (NIR), red and green spectrum. Each image is categorised into either an image of a coffee plantation (over 80% of pixels containing coffee plants) or a non-coffee area. The image is then encoded into a quaternion representation as

$$Q(x,y) = iQ_{NIR}(x,y) + jQ_r(x,y) + kQ_g(x,y).$$

As a first processing step, each image was converted into overlapping $p \times p$ pixel patches where $p = 5$. The patches were vectorised as $\mathbf{s} \in \mathbb{H}^{1\times 25}$ and a subset of $N = 10000$ were randomly selected to calculate a set of filters using each proposed MD-PLS algorithm. The WL-QPLS, HONIPALS and GHOPLS were implemented for the case where the input and output were the same data structure, as required for the covariance matrix diagonalisation application in Section 5.2. For the WL-QPLS implementation, the filter

Table 3
Coffee image classification.

Filter	Accuracy
QPCA	88.6%
WL-QPLS	**89.7%**
HONIPALS	88.5%
GHOPLS	88.8%

was the matrix $\mathbf{W}$ of the concatenated score projections in (45). For the tensor-based algorithms, the quaternion-valued patch was transformed to a tensor representation through (56) and the filter was obtained from the GHOPLS and HONIPALS basis in (34). In each case the number of components selected was $r = 15$. The filtered patches were then represented in the quaternion domain as

$$\tilde{\mathbf{s}} = \tilde{\mathbf{s}}^{(0)} + \tilde{\mathbf{s}}^{(1)}i + \tilde{\mathbf{s}}^{(2)}j + \tilde{\mathbf{s}}^{(3)}k.$$

The dictionary learning step employed in [33] was neglected in order to provide a direct comparison between the algorithms considered. Moreover, a filter was developed using quaternion principal component analysis (QPCA) [30], which is calcualted from the SVD of the covariance matrix.A real-valued feature vector for each image was then calcualted via the same process in [33], outlined in Appendix B. Once the feature vector was obtained the images were classified using a support vector machine (SVM) implemented in Matlab. The 5-fold cross validated results are presented in Table 3, demonstrating the validity of the proposed quaternion and tensor-valued representations.

The advantages of quaternion-valued filters for image representation have been established in the literature [30,31,36]. The multidimensional encoding inherent in the algebra enables the relationships between colours to be preserved and so the processing contains physical meaning [31,36]. This ability is lost if the data is transformed to a real-valued vector. Our application shows that tensors also offer a method in which to preserve geometric information in the calculations. The implementation of the MD-PLS algorithms to determine a subspace filter highlights their flexibility. As the data structure is a parameter, the proposed algorithms offer an alternative way in which to represent a subspace in both regression and unsupervised learning tasks.

6. Conclusion

We have introduced a class of novel algorithms for a PLS-regression solution with multidimensional data. The WL-QPLS has been shown to be a generalisation of the real-valued two-way NIPALS algorithm for quaternion-valued data, which makes possible a regularised widely linear regression for both circular and noncircular quaternion data. Two algorithms have been proposed for tensor-based regression, the HONIPALS and the GHOPLS which adopt a different and more general approach to tensor PLS than the existing HOPLS or N-PLS methods. The HONIPALS uses an unfolding approach for the cross-covariance structure, whereas the GHOPLS uses the tensor contraction product. The analysis shows that these methods take into account the same information but produce alternative results. Finally, a quaternion to tensor isomorphism has been introduced to highlight the equivalence for quadrivariate data. The proposed algorithms have been verified through a simulation on synthetic data, the application of the WL-QPLS algorithm for the open problem of quaternion covariance matrix diagonalisation and a case study of image classification of coffee plantations from satellite data.

The proposed MD-PLS algorithms have been shown to represent comprehensive generalisations of a NIPALS style PLS-regression for tensor- and quaternion-valued data. These developments open a new approach for PLS-regression type applications as they allow the dimensionality of the PLS model to be chosen either as the most appropriate parameters from cross-validation or based on some physically meaningful property of the data, whereas conventional applications are limited to purely selecting the number of PLS scores. In particular, the tensor-based framework allows complete flexibility of the data structure dimensions. The proposed HONIPALS and GHOPLS algorithms have been shown to provide alternative methods of representation for such tensorvariate data, which can enhance interpretability of the solution or model. Moreover, the WL-QPLS utilises the quaternion algebra which can be used to represent relationships among data such as rotations. As a result, the class of MD-PLS algorithms has the potential to be employed for a wide variety of regression scenarios and provide in depth data analysis beyond a two-way solution for common applications such as subspace identification.

Conflict of interest

The authors state that there are no conflicts of interest.

Acknowledgements

A. E. Stott was supported by an EPSRC doctoral training award.

Appendix A. The HOPLS algorithm

Zhao et. al. [17] introduced a tensor-based PLS algorithm, HOPLS, which computes the decomposition (28) in order to produce a tensor regression. The full HOPLS process is outlined in Algorithm 5. The first step in each iteration is to compute the contraction product (13) between $\mathcal{X}$ and $\mathcal{Y}$ along the first mode. The orthogonal rank-$(K_1, K_2, \ldots, K_M, L_1, L_2, \ldots, L_P)$ Tucker decomposition of the resultant tensor, $\mathcal{S}$, is then obtained through the HOOI Algorithm [21]. The HOPLS score vector $\mathbf{t}$ is found as the first singular vector of the unfolded matrix $(\mathcal{X}_i \times_2 \mathbf{P}_i^{(1)\mathrm{T}} \times_3 \mathbf{P}_i^{(2)\mathrm{T}} \cdots \times_{M+1} \mathbf{P}_i^{(M)\mathrm{T}})_{(1)}$, where $\mathbf{P}^{(n)}$ for $n = 1, 2, \ldots, M$ are the factor matrices produced from the prior Tucker decomposition of $\mathcal{S}$. The respective core tensors, $\mathcal{G}$ and $\mathcal{D}$, for the HOPLS decomposition of $\mathcal{X}$ and $\mathcal{Y}$ can then be solved for.

Algorithm 5 The HOPLS algorithm [17].

Input: $\mathcal{X}$, $\mathcal{Y}$, r, K_n for $n = 1, 2, \ldots, M$ and L_n for $n = 1, 2, \ldots, P$
for $i = 1, \ldots, r$ **do**
  $\mathcal{S}_i = \langle \mathcal{X}_i, \mathcal{Y}_i \rangle_{\{1,1\}}$
  Perform rank-$(K_1, K_2, \ldots, K_M, L_1, L_2, \ldots, L_P)$ HOOI decomposition to give $\mathcal{S} = \mathcal{G}_i^C \times_1 \mathbf{P}_i^{(1)} \times_2 \mathbf{P}_i^{(2)} \times_3 \cdots \times_M \mathbf{P}_i^{(M)} \times_{M+1} \mathbf{Q}_i^{(1)} \times_{M+2} \mathbf{Q}_i^{(2)} \times_3 \cdots \times_{M+P} \mathbf{Q}_i^{(P)}$
  $\mathbf{t}_i$ is first leading left singular vector of $\left(\mathcal{X}_i \times_2 \mathbf{P}_i^{(1)T} \times_3 \mathbf{P}_i^{(2)T} \cdots \times_{M+1} \mathbf{P}_i^{(M)T}\right)_{(1)}$
  $\mathcal{G}_i = \mathcal{X}_i \times_1 \mathbf{t}_i^{\mathrm{T}} \times_2 \mathbf{P}_i^{(1)T} \times_3 \mathbf{P}_i^{(2)T} \times_4 \cdots \times_{M+1} \mathbf{P}_i^{(M)T}$
  $\mathcal{D}_i = \mathcal{Y}_i \times_1 \mathbf{t}_i^{\mathrm{T}} \times_2 \mathbf{Q}_i^{(1)T} \times_3 \mathbf{Q}_i^{(2)T} \times_4 \cdots \times_{P+1} \mathbf{Q}_i^{(P)T}$
  $\mathcal{X}_{i+1} = \mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{t}_i \times_2 \mathbf{P}_i^{(1)} \times_3 \mathbf{P}_i^{(2)} \times_4 \cdots \times_{M+1} \mathbf{P}_i^{(M)}$
  $\mathcal{Y}_{i+1} = \mathcal{Y}_i - \mathcal{D}_i \times_1 \mathbf{t}_i \times_2 \mathbf{Q}_i^{(1)} \times_3 \mathbf{Q}_i^{(2)} \times_4 \cdots \times_{P+1} \mathbf{Q}_i^{(P)}$
end for

To calculate the tensor regression coefficients for prediction using the HOPLS model, a mode-1 unfolding of the latent variable representation of $\mathcal{X}$ is first performed

$$\mathbf{X}_{(1)} = \mathbf{T}_{(1)}\mathbf{V}, \tag{A.1}$$

where $\mathbf{T}_{(1)}$ is a matrix of r columns which contains the r score vectors obtained by the HOPLS algorithm and $\mathbf{V}$ is a matrix of r rows with each row, $\mathbf{v}_r$, given by

$$\mathbf{v}_r = \mathbf{G}_{(1)r}(\mathbf{P}^{(M)} \otimes \mathbf{P}^{(M-1)} \otimes \cdots \otimes \mathbf{P}^{(1)})^{\mathsf{T}}.$$

A mode-1 unfolding of $\mathcal{Y}$ is performed as [17]

$$\mathbf{Y}_{(1)} = \mathbf{T}_{(1)}\mathbf{Q}^{*\mathsf{T}}, \tag{A.2}$$

where the matrix $\mathbf{Q}^*$ has r rows, $\mathbf{q}_r^*$, given by

$$\mathbf{q}_r^* = \mathbf{D}_{(1)r}(\mathbf{Q}^{(P)} \otimes \mathbf{Q}^{(P-1)} \otimes \cdots \otimes \mathbf{Q}^{(1)})^{\mathsf{T}}.$$

Now the prediction of the mode-1 unfolded dependent variables, $\mathbf{Y}_{(1)}$, can be performed as

$$\mathbf{Y}_{(1)} = \mathbf{X}_{(1)}\mathbf{V}^{+}\mathbf{Q}^{*}, \tag{A.3}$$

where each column $\mathbf{v}_r^+$ of $\mathbf{V}^+$ is calculated as

$$\mathbf{v}_r^+ = (\mathbf{P}^{(M)} \otimes \mathbf{P}^{(M-1)} \otimes \cdots \otimes \mathbf{P}^{(1)})\mathbf{G}_{(1)r}^{+}.$$

Appendix B. Calculation of feature vector for the image classification problem

The process used in [33] is employed to create the feature vector to be used for image classification from the filtered quaternion-valued patch vectors $\tilde{\mathbf{s}}$. The real-valued feature vector, $\mathbf{f}^{(l)} = [f_q^{(l)}, f_{q+r}^{(l)}, f_{q+2r}^{(l)}]$, $q = 1, 2, \ldots, 15$ was obtained from

$$f_q^{(l)} = |\tilde{s}_q^{(l)}|,$$
$$f_{q+r}^{(l)} = \max(0, \tilde{s}_q^{(l)} - \theta),$$
$$f_{q+2r}^{(l)} = \max(0, -\tilde{s}_q^{(l)} - \theta),$$

where $\theta = 1 \times 10^{-10}$, $l = 0, 1, 2, 3$, for each quaternion axis and r is the length of the vectorised patch. These are then pooled for each patch as

$$\mathbf{F}^{(l)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}_i^{(l)},$$

where N is the number of patches within an image. Each element was power-law transformed as $\mathbf{F} = \mathbf{F}^{\alpha}$, where $\alpha = 0.5$ and then normalised through

$$\bar{\mathbf{F}}^{(l)} = \frac{\mathbf{F}^{(l)}}{||\mathbf{F}^{(l)}||_2^2},$$

before being concatenated into one feature vector $\mathbf{F} = [\bar{\mathbf{F}}^{(0)}, \bar{\mathbf{F}}^{(1)}, \bar{\mathbf{F}}^{(2)}, \bar{\mathbf{F}}^{(3)}]$ to be fed into the SVM.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.sigpro.2019.03.002.

References

[1] S.P. Talebi, D.P. Mandic, A quaternion frequency estimator for three-phase power systems, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2015, pp. 3956–3960.

[2] S.P. Talebi, S. Kanna, D.P. Mandic, A distributed quaternion Kalman filter with applications to smart grid and target tracking, IEEE Trans. Signal Inf. Process. Netw. 2 (4) (2016) 477–488.

[3] K. Ilia, G.G. Calvi, A. Cichocki, D.P. Mandic, Common and individual feature extraction using tensor decompositions: A remedy for the curse of dimensionality? in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 6299–6303.

[4] B. Savas, L. Eldén, Handwritten digit classification using higher order singular value decomposition, Pattern Recognit. 40 (3) (2007) 993–1003.

[5] L. Krieger, F. Grigoli, Optimal reorientation of geophysical sensors: a quaternion-based analytical solution, Geophysics 80 (2) (2015) F19–F30.

[6] C.C. Took, G. Strbac, K. Aihara, D. Mandic, Quaternion-valued short-term joint forecasting of three-dimensional wind and atmospheric parameters, Renew. Energy 36 (6) (2011) 1754–1760.

[7] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D.P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization. Part 1: Low-rank tensor decompositions, Found. Trends® Mach. Learn. 9 (4-5) (2016) 249–429, doi:10.1561/2200000059.

[8] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D.P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization. Part 2: Applications and future perszpectives, Found. Trends® Mach. Learn. 9 (6) (2017) 431–673, doi:10.1561/2200000067.

[9] C.C. Took, D. Mandic, J. Benesty, Study of the quaternion LMS and four-channel LMS algorithms, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2009, pp. 3109–3112.

[10] S. Wold, M. Sjöström, L. Eriksson, PLS-Regression: a basic tool of chemometrics, Chemom. Intell. Lab. Syst. 58 (2) (2001) 109–130.

[11] S. Wold, A. Ruhe, H. Wold, W. Dunn III, The collinearity problem in linear regression. the partial least squares (PLS) approach to generalized inverses, SIAM J. Sci. Stat. Comput. 5 (3) (1984) 735–743.

[12] P. Geladi, B. Kowalski, Partial least-squares regression: a tutorial, Anal. Chim. Acta 185 (1986) 1–17.

[13] H. Abdi, Partial least squares regression and projection on latent structure regression (PLS regression), Wiley Interdiscip. Rev. Comput. Stat. 2 (1) (2010) 97–106.

[14] A.E. Stott, S. Kanna, D.P. Mandic, W.T. Pike, An online NIPALS algorithm for partial least squares, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 4177–4181.

[15] A.E. Stott, S. Kanna, D.P. Mandic, Widely linear complex partial least squares for latent subspace regression, Signal Process. 152 (2018) 350–362.

[16] R. Bro, Multiway calibration. multilinear PLS, J. Chemom. 10 (1) (1996) 47–61.

[17] Q. Zhao, C.F. Caiafa, D.P. Mandic, Z.C. Chao, Y. Nagasaka, N. Fujii, L. Zhang, A. Cichocki, Higher order partial least squares (HOPLS): a generalized multilinear regression method, IEEE Trans. Pattern Anal. Mach. Intell. 35 (7) (2013) 1660–1673.

[18] A. Lepore, B. Palumbo, C. Capezza, Analysis of profiles for monitoring of modern ship performance via partial least squares methods, Qual. Reliab. Eng. Int. 34 (7) (2018) 1424–1436.

[19] R. Rosipal, N. Krämer, Overview and recent advances in partial least squares, in: Subspace, Latent Structure and Feature Selection, Springer, 2006, pp. 34–51.

[20] T.G. Kolda, Multilinear operators for higher-order decompositions., Technical Report, Sandia National Laboratories, 2006.

[21] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.

[22] C.C. Took, D.P. Mandic, Augmented second-order statistics of quaternion random signals, Signal Process. 91 (2) (2011) 214–224.

[23] J. Vía, D. Ramírez, I. Santamaría, Properness and widely linear processing of quaternion random vectors, IEEE Trans. Inf. Theory 56 (7) (2010) 3502–3515.

[24] J.B. Kuipers, et al., Quaternions and Rotation Sequences, 66, Princeton University Press, Princeton, 1999.

[25] D.P. Mandic, V.S.L. Goh, Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models, 59, John Wiley & Sons, 2009.

[26] S.P. Talebi, S. Kanna, D.P. Mandic, Real-time estimation of quaternion impropriety, in: Proceedings of the IEEE International Conference on Digital Signal Processing (DSP), IEEE, 2015, pp. 557–561.

[27] E. Ollila, On the circularity of a complex random variable, IEEE Signal Process. Lett. 15 (2008) 841–844.

[28] M. Xiang, C.C. Took, D.P. Mandic, Cost-effective quaternion minimum mean square error estimation: from widely linear to four-channel processing, Signal Process. 136 (2017) 81–91.

[29] I. Jolliffe, Principal component analysis, in: International Encyclopedia of Statistical Science, Springer, 2011, pp. 1094–1096.

[30] N. Le Bihan, S.J. Sangwine, Quaternion principal component analysis of color images, in: Proceedings of the International Conference on Image Processing (ICIP), 1, IEEE, 2003, pp. I–809.

[31] N. Le Bihan, J. Mars, Singular value decomposition of quaternion matrices: a new tool for vector-sensor signal processing, Signal Process. 84 (7) (2004) 1177–1199.

[32] M. Xiang, S. Enshaeifar, A.E. Stott, C.C. Took, Y. Xia, S. Kanna, D.P. Mandic, Simultaneous diagonalisation of the covariance and complementary covariance matrices in quaternion widely linear signal processing, Signal Process. 148 (2018) 193–204.

[33] V. Risojević, Z. Babić, Unsupervised quaternion feature learning for remote sensing image classification, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 9 (4) (2016) 1521–1531.

[34] V. Risojevic, Z. Babic, Unsupervised learning of quaternion features for image classification, in: Proceedings of the 11th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 1, IEEE, 2013, pp. 345–348.

[35] O.A. Penatti, K. Nogueira, J.A. dos Santos, Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 44–51.

[36] L. Shi, B. Funt, Quaternion color texture segmentation, Comput. Vis. Image Underst. 107 (1) (2007) 88–96.