Technometrics

ISSN: 0040-1706 (Print) 1537-2723 (Online) Journal homepage: https://www.tandfonline.com/loi/utch20

Multiple Tensor-on-Tensor Regression: An Approach for Modeling Processes With Heterogeneous Sources of Data

Mostafa Reisi Gahrooei, Hao Yan, Kamran Paynabar & Jianjun Shi

To cite this article: Mostafa Reisi Gahrooei, Hao Yan, Kamran Paynabar & Jianjun Shi (2020): Multiple Tensor-on-Tensor Regression: An Approach for Modeling Processes With Heterogeneous Sources of Data, Technometrics, DOI: 10.1080/00401706.2019.1708463

To link to this article: https://doi.org/10.1080/00401706.2019.1708463

TECHNOMETRICS
2020, VOL. 00, NO. 0, 1–13
https://doi.org/10.1080/00401706.2019.1708463

Taylor & Francis
Taylor & Francis Group

Multiple Tensor-on-Tensor Regression: An Approach for Modeling Processes With Heterogeneous Sources of Data

Mostafa Reisi Gahrooei[a], Hao Yan[b], Kamran Paynabar[c], and Jianjun Shi[c]

[a]Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL; [b]School of Computing, Informatics, & Decision Systems Engineering, Arizona State University, Tempe, AZ; [c]H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA

ABSTRACT

In recent years, measurement or collection of heterogeneous sets of data such as those containing scalars, waveform signals, images, and even structured point clouds, has become more common. Statistical models based on such heterogeneous sets of data that represent the behavior of an underlying system can be used in the monitoring, control, and optimization of the system. Unfortunately, available methods mainly focus on the scalars and profiles and do not provide a general framework for integrating different sources of data to construct a model. This article addresses the problem of estimating a process output, measured by a scalar, curve, image, or structured point cloud by a set of heterogeneous process variables such as scalar process setting, profile sensor readings, and images. We introduce a general multiple tensor-on-tensor regression approach in which each set of input data (predictor) and output measurements are represented by tensors. We formulate a linear regression model between the input and output tensors and estimate the parameters by minimizing a least square loss function. To avoid overfitting and reduce the number of parameters to be estimated, we decompose the model parameters using several basis matrices that span the input and output spaces, and provide efficient optimization algorithms for learning the basis and coefficients. Through several simulation and case studies, we evaluate the performance of the proposed method. The results reveal the advantage of the proposed method over some benchmarks in the literature in terms of the mean square prediction error. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received April 2018
Accepted November 2019

KEYWORDS
Alternating least square algorithm; Blessing of dimensionality; Block coordinate descent algorithm; High-dimensional process modeling; Tucker decomposition

1. Introduction

Nowadays, heterogeneous sets of data containing scalars, waveform signals, images, etc. are more and more available. For example, in semiconductor manufacturing, machine/process settings (scalar variables), sensor readings in chambers (waveform signals), and wafer shape measurements (images) may be collected to model and monitor the process. Statistical models based on such heterogeneous sets of data that represent the behavior of an underlying system can be used in the monitoring, control, and optimization of the system. This can benefit many applications, including manufacturing processes (Szatvanyi, Duchesne, and Bartolacci 2006; Wójcik and Kotyra 2009), food industries (Yu and MacGregor 2003), medical decision-making (Bellon et al. 1995), and structural health monitoring (Balageas, Fritzen, and Güemes 2010). Specifically, regression analysis of such data may lead to the construction of a statistical model that estimates/predicts a high-dimensional (HD) variable based on various types of predictors. In this article, we refer to nonscalar variables as HD variables.

Unfortunately, most works in regression modeling consider scalars and waveform signals (Liang, Wu, and Carroll 2003; Ramsay and Silverman 2005; Fan et al. 2014; Luo and Qi 2017), and their extension to images or structured point clouds is nontrivial if not impossible. However, in several applications, images or structured point clouds can provide rich information about the system performance. For example, materials scientists are interested in constructing a link between process variables, for example, the temperature and pressure under which a material is operating, and the microstructure of the material (Khosravani, Cecen, and Kalidindi 2017), which is often represented by an image or a variation of the microstructure image obtained by two-point statistics. Generating such a linkage model requires regression analysis between scalar and waveform process variables as inputs and an image as an output (Gorgannejad et al. 2019).

As another example, in semiconductor manufacturing, overlay errors (defined as the difference between in-plane displacements of two layers of a wafer) are directly influenced by the shape of the wafer before the lithographic process. In this process, both the wafer shape and the overlay error (in the x and y directions) can be represented as images as shown in Figure 1. Prediction of the overlay error across a wafer based on the wafer shape can be fed forward to exposure tools for specific corrections (Turner, Ramkhalawon, and Sinha 2013). To predict the overlay error based on the wafer shape deformation, an image-on-image statistical model is required to capture the correlation between the wafer overlay and shape.

In addition to the space and computational issues caused by the large size of the HD variables, the challenge of developing

CONTACT Kamran Paynabar kamran.paynabar@isye.gatech.edu H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, 436 Groseclose Bld, 765 Ferst Drive, Atlanta, GA 30332.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/r/TECH.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/TECH.

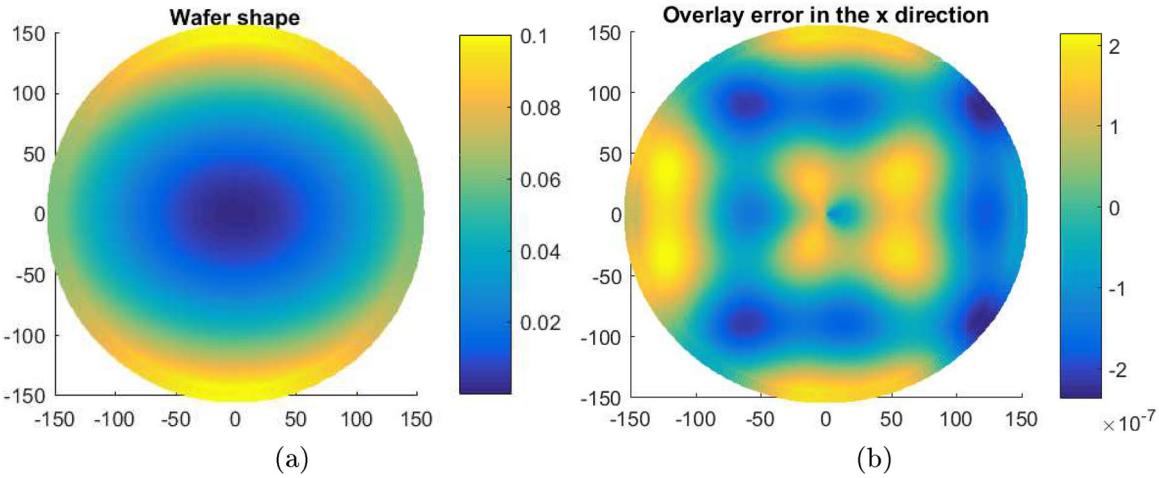

Figure 1. Examples of the (a) wafer shape and (b) *x* coordinate overlay error. All measurements are in millimeters (mm). The wafer in (a) has an overall bow shape, with several high-order wave-form patterns that cannot be seen. (b) The misalignment of two layers in the *x* coordinate. The dark blue represents a large error in the negative direction of the *x* coordinate, and light yellow represents a large error in the positive direction of the *x* coordinate.

an accurate model for a process with a heterogeneous group of variables is 2-fold: Integrating variables of different forms (e.g., scalars, images, curves) while capturing their "within" correlation structure. Mishandling this challenge can lead to an overfitted or inaccurate model. The regular regression approach that considers each observation within an HD variable as an independent predictor excessively increases the number of covariates in comparison to the sample size ($p \gg n$) and ignores the correlation between the observations. Consequently, this method may cause severe overfitting and produce inaccurate predictions. Principal component regression (PCR) and partial least square (PLS) regression alleviate the problem by reducing the dimension of both the input variables and the output. Nevertheless, both PCR and PLS fail to exploit the spatial structure of profiles, images or point clouds**.** Furthermore, PCR determines the principal components (PCs) of the inputs and the outputs separately from each other without considering the interrelationship between them. Functional data analysis, specifically the functional regression model (FRM), has become popular in recent years due to its built-in data reduction functionality and its ability to capture nonlinear correlation structures (Liang, Wu, and Carroll 2003; Ramsay and Silverman 2005; Yao, Müller, and Wang 2005; Fan et al. 2014; Ivanescu et al. 2015; Luo and Qi 2017). However, FRM requires a set of basis functions that is usually specified based on domain knowledge rather than a data-driven approach. Recently, Luo and Qi (2017) proposed an approach that can combine several profiles and scalars to predict a curve, while learning the bases that span the input and output spaces. Nevertheless, it is not straightforward to extend this approach to other forms of data effectively.

In the past few years, multilinear algebra (and, in particular, tensor analysis) has shown promising results in many applications from network analysis to anomaly detection and process monitoring (Sun, Papadimitriou, and Philip 2006; Sapienza et al. 2015; Yan, Paynabar, and Shi 2015). Nevertheless, only a few works in the literature use tensor analysis for regression modeling. Zhou, Li, and Zhu (2013) has successfully employed tensor regression using PARAFAC/CANDECOMP (CP) decomposition to estimate a scalar variable based on an image input. The CP decomposition approximates a tensor as a sum of several rank-1 tensors (Kiers 2000). Zhou, Li, and Zhu (2013) further extended their approach to a generalized linear model for tensor regression in which the scalar output follows any exponential family distribution. Li, Zhou, and Li (2013) performed tensor regression with scalar output using Tucker decomposition. Tucker decomposition is a form of higher order PCA that decomposes a tensor into a core tensor multiplied by a matrix along each mode (Tucker 1963). Yan, Paynabar, and Pacella (2019) performed the opposite regression and estimated point cloud data as an output using a set of scalar process variables. Recently, convex and nonconvex optimization frameworks have been offered to deal with HD multi-response tensor regression problems (Chen, Raskutti, and Yuan 2019; Raskutti, Yuan, and Chen 2019).

Another group of works extends PLS to tensors. N-way PLS (N-PLS) is a generalization of PLS offered by Bro (1996) that uses CP decomposition to capture the lower-rank structure of tensors. To address the limitations of N-PLS, including poor fitness ability, computational complexity and slow convergence when handling multivariate dependent data and higher order ($N > 3$) independent data, higher-order partial least squares (HOPLS) is proposed by Zhao et al. (2013). HOPLS uses Tucker decomposition which is more flexible and has better fitness ability than CP decomposition. The main challenge of the tensor PLS approaches is that they require dealing with the covariance tensor between the input and output tensors. This covariance tensor can become extremely large that makes the computation inefficient and in some situations intractable. Besides, these approaches are only designed for a single input tensor and it is not clear how they can be extended to multiple inputs without artificially increasing the inputs size.

Recently, Lock (2017) developed a tensor-on-tensor (TOT) regression approach that can estimate a tensor using a tensor input while learning the decomposition bases. However, there are some limitations in their proposed TOT. First, TOT uses CP decomposition, which restricts both the input and output bases to have exactly the same rank (say, R). This restriction may cause over- or under-estimation when the input and the output have different ranks. For example, when estimating an image based on a few scalar inputs, the rank of the output can be far larger than the input matrix. Second and more importantly,

this approach can only take into account a single tensor input and cannot be used effectively when multiple sources of input data with different dimensions and forms (e.g., a combination of scalar, curve, and image data) are available. Because the output and the inputs should have the same rank, extending the TOT approach to multiple tensor inputs as well requires all the inputs to have the same rank (which is equal to the rank of the output). However, this means that in certain situations, such as when scalar and image inputs exist, one of the inputs should take the rank of the other, causing a severe underfitting or overfitting problem. Finally, the TOT approach fails to work on tensors of moderate size (e.g., on the image data of size 20,000 pixels used in our case study) due to its high space complexity.

The overarching goal of this article is to overcome the limitations of the previous methods, such as PCR, FRMs, and TOT, by constructing a unified regression framework that estimates a scalar, curve, image, or structured point cloud output based on a heterogeneous set of HD input variables. This will be achieved by representing the output and each group of input variables as separate tensors and by developing a multiple tensor-on-tensor (MTOT) regression. To avoid overfitting due to the estimation of a large number of parameters, we use Tucker decomposition. We obtain the input bases by performing Tucker decomposition on the input tensors, then define a least square loss function to estimate the decomposition coefficients and output bases. To alleviate the uniqueness issues in tensor decomposition, we impose an orthonormality constraint over the output bases when minimizing the loss function and show a closed-form solution for both the bases and the decomposition coefficient in each iteration of our algorithm. This approach not only performs dimension reduction similar to PCR, but it also learns the output bases in accordance with the input space. Furthermore, the use of tensors to represent the data utilizes the spatial structure of a profile, image or structured point cloud.

The rest of the article is organized as follows: In Section 2, we introduce notations and review some of the multilinear algebra concepts used in the article. In Section 3, we formulate the MTOT regression model and illustrate the closed-form solution for estimating the parameters. In Section 4, we describe three simulation studies. The first simulation study combines a profile and scalar data to estimate a profile output. This simulation study is particularly considered to compare MTOT to the available methods in functional regression. The second and third simulation studies contain images or point clouds either as the input or output. In each simulation study, we compare the performance of the proposed method with benchmarks in terms of (standardized) mean square prediction errors (MSPE). Two case studies will be considered in Section 5. First, we employ our approach to predict the overlay errors based on the wafer shape. In the second case study, we evaluate our proposed method in predicting air/gas ratio of an engine given a set of signals obtained from that engine. Finally, we summarize the article in Section 6.

2. Tensor Notation and Multilinear Algebra

In this section, we introduce the notations and basic tensor algebra used in this article. Throughout the article, we denote a scalar by a lower or upper case letter, for example, a or A; a vector by a boldface lowercase letter and a matrix by a boldface uppercase letter, for example, $\boldsymbol{a}$ and $\mathbf{A}$; and a tensor by a calligraphic letter, for example, $\mathcal{A}$. For example, we denote an order-n tensor by $\mathcal{R} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$, where I_i is the dimension of the ith mode of tensor $\mathcal{R}$. We also denote a mode-j matricization of tensor $\mathcal{R}$ as $\boldsymbol{R}_{(j)} \in \mathbb{R}^{I_j \times I_{-j}}$, whose columns are the mode-j fibers of the corresponding tensor $\mathcal{R}$, and $I_{-j} = I_1 \times I_2 \times \cdots \times I_{j-1} \times I_{j+1} \times \cdots \times I_n$. We also define a more general matricization of a tensor $\mathcal{T} \in \mathbb{R}^{P_1 \times \cdots \times P_l \times Q_1 \times \cdots \times Q_d}$ as follows: Let $\mathbb{I} = \{I_1, I_2, \ldots, I_l\}$ and $\mathbb{Q} = \{Q_1, Q_2, \ldots, Q_d\}$ be two sets that partition the set $\{I_1, I_2, \ldots, I_l, Q_1, Q_2, \ldots, Q_d\}$, which contains the dimensions of the modes of the tensor $\mathcal{T}$. Then, the matricized tensor is specified by $\boldsymbol{T}_{(\mathbb{I} \times \mathbb{Q})} \in \mathbb{R}^{P \times Q}$, where $P = \prod_{i=1}^{l} P_i$ and $Q = \prod_{i=1}^{d} Q_i$, and $\left(\boldsymbol{T}_{(\mathbb{I} \times \mathbb{Q})}\right)_{ij} = \mathcal{T}_{p_1 \cdots p_l q_1 \cdots q_d}$, where $i = 1 + \sum_{r=1}^{l} \prod_{n=1}^{r} P_n \left(p_n - 1\right)$ and $j = 1 + \sum_{r=1}^{d} \prod_{n=1}^{r} Q_n \left(q_n - 1\right)$. For simplicity of notation, we will denote $\boldsymbol{T}_{(\mathbb{I} \times \mathbb{Q})}$ as $\boldsymbol{T}$.

The Frobenius norm of a tensor $\mathcal{R}$ can be defined as the Frobenius norm of any matricized tensor, for example, $||\mathcal{R}||_F^2 = ||\boldsymbol{R}_{(1)}||_F^2$. The mode-$j$ product of a tensor $\mathcal{R}$ by a matrix $\boldsymbol{A} \in \mathbb{R}^{L \times I_j}$ is a tensor in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{j-1} \times L \times I_{j+1} \times \cdots \times I_n}$ and is defined as $\left(\mathcal{R} \times_j \boldsymbol{A}\right)_{i_1, i_2, \ldots, i_{j-1}, l, i_{j+1}, \ldots, i_K} = \sum_{i_j=1}^{I_j} \mathcal{R}_{i_1, \ldots, i_j, \ldots, i_K} \boldsymbol{A}_{l, i_j}$.

The Tucker decomposition of a tensor $\mathcal{R}$ decomposes the tensor into a core tensor $\mathcal{C} \in \mathbb{R}^{P_1 \times P_2 \times \cdots \times P_n}$ and n orthogonal matrices $\boldsymbol{U}_i \in \mathbb{R}^{I_i \times P_i}$ $(i = 1, 2, \ldots, n)$ so that $\mathcal{R} = \mathcal{C} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \cdots \times_n \boldsymbol{U}_n$. The dimensions of the core tensor $\mathcal{C}$ is smaller than $\mathcal{A}$, that is, $P_j \leq I_j$ $\left(j = 1, 2, \ldots, n\right)$. Furthermore, the Kronecker product of two matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times s}$ is denoted as $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{mr \times ns}$ and is obtained by multiplying each element of matrix $\boldsymbol{A}$ to the entire matrix $\boldsymbol{B}$:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \ldots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \ldots & a_{mn}\boldsymbol{B} \end{bmatrix}.$$

We link the tensor multiplication with the Kronecker product using Proposition 1.

Proposition 1. Let $\boldsymbol{U}_i \in \mathbb{R}^{P_i \times \tilde{P}_i}$ $(i = 1, \ldots, l)$ and $\boldsymbol{V}_i \in \mathbb{R}^{Q_i \times \tilde{Q}_i}$ $(i = 1, \ldots, d)$, and let $\mathcal{T} \in \mathbb{R}^{P_1 \times \cdots \times P_l \times Q_1 \times \cdots \times Q_d}$ and $\mathcal{C} \in \mathbb{R}^{\tilde{P}_1 \times \cdots \times \tilde{P}_l \times \tilde{Q}_1 \times \cdots \times \tilde{Q}_d}$, then $\mathcal{T} = \mathcal{C} \times_1 \boldsymbol{U}_1 \times \boldsymbol{U}_2 \times_3 \cdots \times_l \boldsymbol{U}_l \times_{l+1} \boldsymbol{V}_1 \times_{l+2} \cdots \times_{l+d} \boldsymbol{V}_d$ is equivalent to $\boldsymbol{T} = \left(\boldsymbol{U}_l \otimes \boldsymbol{U}_{l-1} \otimes \cdots \otimes \boldsymbol{U}_1\right) \boldsymbol{C} \left(\boldsymbol{V}_d \otimes \boldsymbol{V}_{d-1} \otimes \cdots \otimes \boldsymbol{V}_1\right)^{\mathrm{T}}$, where $\boldsymbol{C} \in \mathbb{R}^{\tilde{P} \times \tilde{Q}}$ is an unfold of the core tensor $\mathcal{C}$ with $\tilde{P} = \prod_{j=1}^{l} \tilde{P}_j$ and $\tilde{Q} = \prod_{j=1}^{d} \tilde{Q}_j$ and $\boldsymbol{T}$ is the general matricization of $\mathcal{T}$.

This proposition can be found in Kolda (2006, Proposition 3.7, p. 11). Finally, the contraction product, also known as the Einstein product, of two tensors $\mathcal{B} \in \mathbb{R}^{P_1 \times \cdots \times P_l \times Q_1 \times \cdots \times Q_d}$ and $\mathcal{X} \in \mathbb{R}^{P_1 \times \cdots \times P_l}$ is denoted as $\mathcal{X} * \mathcal{B} \in \mathbb{R}^{Q_1 \times \cdots \times Q_d}$ and is defined as

$$(\mathcal{X} * \mathcal{B})_{q_1 \cdots q_d} = \sum_{p_1, \ldots, p_l} \mathcal{X}_{p_1, \ldots, p_l} \mathcal{B}_{p_1, \ldots, p_l, q_1, \ldots, q_d}.$$

3. Multiple Tensor-on-Tensor Regression Framework

In this section, we introduce the MTOT regression framework as an approach for integrating multiple types of data with

different dimensions and forms to model a process. Assume a set of training data of size M is available and includes response tensors $\mathcal{Y}_i \in \mathbb{R}^{Q_1 \times \cdots \times Q_d}$ $(i = 1, \ldots, M)$ and input tensors $\mathcal{X}_{ji} \in \mathbb{R}^{P_{j1} \times \cdots \times P_{jl_j}}$ $\left(i = 1, \ldots, M; j = 1, \ldots, p\right)$, where p is the number of inputs. The goal of MTOT is to model the relationship between the input tensors and the response using the following linear form:

$$\mathcal{Y}_i = \sum_{j=1}^{p} \mathcal{X}_{ji} * \mathcal{B}_j + \mathcal{E}_i,\ i = 1, \ldots, M, \tag{1}$$

where $\mathcal{B}_j \in \mathbb{R}^{P_{j1} \times \cdots \times P_{jl_j} \times Q_1 \times \cdots \times Q_d}$ is the model parameter to be estimated and $\mathcal{E}_i$ is an error tensor whose elements are from a random process. Please note that in this article, we do not make an assumption over the distribution of the error. One can assume that the error follows a normal distribution with a covariance tensor Σ and use this information to estimate the parameters by maximizing the corresponding likelihood function (see Yan, Paynabar, and Pacella 2019 for more discussion). Nevertheless, we avoid this procedure to achieve a computationally more efficient approach. To achieve a more compact representation of the model (1), we can combine tensors $\mathcal{Y}_i$, $\mathcal{X}_{ji}$, and $\mathcal{E}_i$ $(i = 1, \ldots, M)$ into one-mode larger tensors $\mathcal{Y} \in \mathbb{R}^{M \times Q_1 \times \cdots \times Q_d}$, $\mathcal{X}_j \in \mathbb{R}^{M \times P_{j1} \times P_{j2} \times \cdots \times P_{jl_j}}$ $\left(j = 1, 2, \ldots, p\right)$, and $\mathcal{E} \in \mathbb{R}^{M \times Q_1 \times \cdots \times Q_d}$ and write

$$\mathcal{Y} = \sum_{j=1}^{p} \mathcal{X}_j * \mathcal{B}_j + \mathcal{E}. \tag{2}$$

The matricization of (2) gives

$$\boldsymbol{Y}_{(1)} = \sum_{j=1}^{p} \boldsymbol{X}_{j(1)} \boldsymbol{B}_j + \boldsymbol{E}_{(1)}, \tag{3}$$

where $\boldsymbol{Y}_{(1)}$ and $\boldsymbol{X}_{j(1)}$ are mode-1 unfolding of tensors $\mathcal{Y}$ and $\mathcal{X}_j$, respectively, and the first mode corresponds to the sample mode. $\boldsymbol{B}_j \in \mathbb{R}^{P_j \times Q}$ is an unfold of tensor $\mathcal{B}_j$ with $P_j = \prod_{k=1}^{l_j} P_{jk}$ and $Q = \prod_{k=1}^{d} Q_k$. It is intuitive that the parameters of (3) can be estimated by minimizing the mean squared loss function $L = ||\boldsymbol{Y}_{(1)} - \sum_{j=1}^{p} \boldsymbol{X}_{j(1)} \boldsymbol{B}_j||_F^2$. However, this requires estimating $\sum_{j=1}^{p} \prod_{i=1}^{l_j} P_{ji} \prod_{k=1}^{d} Q_k$ parameters. For example, in the situation in which $p = 1$, minimizing the loss function gives a closed-form solution $\hat{\boldsymbol{B}} = \left(\boldsymbol{X}_{(1)}^{\mathrm{T}} \boldsymbol{X}_{(1)}\right)^{-1} \boldsymbol{X}_{(1)}^{\mathrm{T}} \boldsymbol{Y}_{(1)}$ that requires estimating $\prod_{i=1}^{l} P_i \prod_{j=1}^{d} Q_j$ parameters. Estimating such a large number of parameters is prone to severe overfitting and is often intractable. In reality, due to the structured correlation between $\mathcal{X}_j$ and $\mathcal{Y}$, we can assume that the parameter $\mathcal{B}_j$ lies in a much lower dimensional space and can be expanded using a set of basis matrices via a tensor product. That is, for each $\mathcal{B}_j$ $\left(j = 1, \ldots, p\right)$, we can write

$$\mathcal{B}_j = \mathcal{C}_j \times_1 \boldsymbol{U}_{j1} \times_2 \boldsymbol{U}_{j2} \times_3 \cdots \times_{l_j} \boldsymbol{U}_{jl_j} \times_{l_j+1} \boldsymbol{V}_1 \times_{l_j+2} \cdots \times_{l_j+d} \boldsymbol{V}_d, \tag{4}$$

where $\mathcal{C}_j \in \mathbb{R}^{\tilde{P}_{j1} \times \cdots \times \tilde{P}_{jl_j} \times \tilde{Q}i \times \cdots \times \tilde{Q}_d}$ is a core tensor with $\tilde{P}_{ji} \ll P_{ji}$ $\left(j = 1, \ldots, p;\ i = 1, \ldots, l_j\right)$ and $\tilde{Q}_i \ll Q_i$ $(i = 1, \ldots, d)$; $\left\{\boldsymbol{U}_{ji} : j = 1, \ldots, p;\ i = 1, \ldots, l_j\right\}$ is a set of bases that spans the jth input space; and $\{\boldsymbol{V}_i : i = 1, \ldots, d\}$ is a set of bases that spans the output space. With this low-dimensional representation, the estimation of $\mathcal{B}_j$ reduces to learning the core tensor $\mathcal{C}_j$ and the basis matrices $\boldsymbol{U}_{ji}$ and $\boldsymbol{V}_i$. In this article, we allow $\boldsymbol{U}_{ji}$ to be obtained directly from the input spaces. One important choice of U_{ji} is the bases obtained from the Tucker decomposition of the input tensor $\mathcal{X}_j$, that is,

$$\left\{\mathcal{D}_j, \boldsymbol{U}_{j1}, \ldots, \boldsymbol{U}_{jl_j}\right\} = \arg \min_{\mathcal{D}_j, \{\boldsymbol{U}_{ji}\}} \left|\left|\mathcal{X}_j - \mathcal{D}_j \times_1 \boldsymbol{U}_{j1} \times \cdots \times_{l_j} \boldsymbol{U}_{jl_j}\right|\right|_F^2.$$

Please note that in general the bases obtained from the Tucker decomposition of the inputs are not necessarily the same as the ones in (4). Nevertheless, setting $\boldsymbol{U}_{ji}$ equal to the basis matrices computed from the input spaces creates transformed version of the core tensors, which together with the basis matrices produce the same estimation to the output. This is known as rotation indeterminacy of tensors which indicates that the (Tucker) decomposition of a tensor is not unique due to the possible orthogonal transformations of basis matrices. For more details please refer to Anandkumar et al. (2013) and Yan, Paynabar, and Pacella (2019). Nevertheless, depending on what set of bases to be used we may require a larger or smaller value of the ranks $\tilde{P}_{ji}$ to achieve the same level of prediction accuracy. Furthermore, fixing $\left\{\boldsymbol{U}_{j1}, \ldots, \boldsymbol{U}_{jl_j}\right\}$ improves the computational complexity of our algorithm significantly. Next, we iteratively estimate the core tensors $\mathcal{C}_j$ and the basis matrices $\boldsymbol{V}_i$ by solving the following optimization problem:

$$\begin{aligned} \left\{\mathcal{C}_j, \boldsymbol{V}_1, \ldots, \boldsymbol{V}_d\right\} &= \arg\min \left|\left|\boldsymbol{Y}_{(1)} - \sum_{j=1}^{p} \boldsymbol{X}_{j(1)} \boldsymbol{B}_j\right|\right|_F^2, \\ s.t.\ \mathcal{B}_j &= \mathcal{C}_j \times_1 \boldsymbol{U}_{j1} \times_2 \boldsymbol{U}_{j2} \times_3 \cdots \times_{l_j} \boldsymbol{U}_{jl_j} \times_{l_j+1} \boldsymbol{V}_1 \\ &\quad \times_{l_j+2} \cdots \times_{l_j+d} \boldsymbol{V}_d, \\ \boldsymbol{V}_i^{\mathrm{T}} \boldsymbol{V}_i &= I_{\tilde{Q}_i}\ (i = 1, \ldots, d), \end{aligned} \tag{5}$$

where $\boldsymbol{I}_{\tilde{Q}i}$ is a $\tilde{Q}_i \times \tilde{Q}_i$ identity matrix. The first constraint ensures that the tensor of parameters is low-rank, and the orthogonality constraint $\boldsymbol{V}_i^{\mathrm{T}} \boldsymbol{V}_i = \boldsymbol{I}_{\tilde{Q}i}$ is to restrict the space of possible bases and core tensors in the parameter decomposition. It should be noted that in general, the problem of estimating functional data through a set of functions may not always be identifiable. He, Müller, and Wang (2000), Chiou, Müller, and Wang (2004), and Lock (2017) discuss the identifiability problem in functional and tensor regression. Because the main purpose of this article is to estimate and predict the output, we do not discuss the identifiability issue here, as learning any correct set of parameters $\left\{\boldsymbol{B}_k : k = 1, \ldots, p\right\}$ will eventually lead to the same estimation of the output.

To solve (5), we combine the alternating least square (ALS) approach with the block coordinate descent (BCD) method (designated by ALS-BCD). The advantages of ALS algorithms are conceptual simplicity, noise robustness, and computational efficiency (Sharan and Valiant 2017). ALS has also shown great promise in the literature for solving tensor decomposition and regression applications with satisfying results (Kolda 2006). To be able to employ ALS-BCD, we first demonstrate Proposition 2:

Proposition 2. Given $\boldsymbol{U}_{ki}\ (k = 1, \ldots, p;\ i = 1, 2, \ldots, l_j)$, $\boldsymbol{V}_i\ (i = 1, 2, \ldots, d)$, and $\mathcal{C}_k\ (k \neq j)$ are known, a reshaped form of the core tensor $\mathcal{C}_j$ can be estimated as

$$\tilde{\mathcal{C}}_j = \mathcal{R}_j \times_1 \left(\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{Z}_j\right)^{-1}\boldsymbol{Z}_j^{\mathrm{T}} \times_2 \left(\boldsymbol{V}_1^{\mathrm{T}}\boldsymbol{V}_1\right)^{-1}\boldsymbol{V}_1^{\mathrm{T}} \times_3 \left(\boldsymbol{V}_2^{\mathrm{T}}\boldsymbol{V}_2\right)^{-1}\boldsymbol{V}_2^{\mathrm{T}} \cdots \times_{d+1} \left(\boldsymbol{V}_d^{\mathrm{T}}\boldsymbol{V}_d\right)^{-1}\boldsymbol{V}_d^{\mathrm{T}}, \quad (6)$$

where $\boldsymbol{Z}_j = \boldsymbol{X}_{j(1)}(\boldsymbol{U}_{jl} \otimes \boldsymbol{U}_{jl-1} \otimes \cdots \otimes \boldsymbol{U}_{j1})$ and $\mathcal{R}_j = \mathcal{Y} - \sum_{i\neq j}^{p} \mathcal{B}_j * \mathcal{X}_j$.

The simplified proof of this proposition is given in Appendix A. Furthermore, if $\boldsymbol{V}_i$'s are orthogonal, the core tensor can be obtained efficiently by the tensor product as

$$\tilde{\mathcal{C}}_j = \mathcal{R}_j \times_1 \left(\boldsymbol{Z}_j^{\mathrm{T}}\boldsymbol{Z}_j\right)^{-1}\boldsymbol{Z}_j^{\mathrm{T}} \times_2 \boldsymbol{V}_1^{\mathrm{T}} \times_3 \boldsymbol{V}_2^{\mathrm{T}} \cdots \times_{d+1} \boldsymbol{V}_d^{\mathrm{T}}.$$

Note that $\tilde{\mathcal{C}}_j$ has fewer modes $(d + 1)$ than the original core tensor $\mathcal{C}_j$ in (4), but it can be transformed into $\mathcal{C}$ by a simple reshape operation. Also in the cases where the sparsity of the core tensor is of interest, one can add a lasso penalty over the core tensor, and use numerical algorithms (e.g., iterative shrinkage-thresholding algorithm (Beck and Teboulle 2009)) to solve the problem. Furthermore, one can efficiently estimate the basis matrices V_i using singular value decomposition according to the following proposition.

Proposition 3. Given $\mathcal{C}_j$, $\boldsymbol{U}_{ji}$, and $\boldsymbol{V}_k\ (k \neq i)$, we can solve $\boldsymbol{V}_i$ by

$$\boldsymbol{V}_i = \boldsymbol{R}\boldsymbol{W}^{\mathrm{T}},$$

where $\boldsymbol{R}$ and $\boldsymbol{W}$ are obtained from the singular value decomposition of $\boldsymbol{Y}_{(i)}\boldsymbol{S}^{\mathrm{T}}$, where $\boldsymbol{S} = \sum_{j=1}^{p}\boldsymbol{S}_j$ and $\boldsymbol{S}_j = \tilde{\boldsymbol{C}}_{j(i)}\left(\boldsymbol{V}_d \otimes \cdots \otimes \boldsymbol{V}_{i+1} \otimes \boldsymbol{V}_{i-1} \cdots \otimes \boldsymbol{V}_1 \otimes \boldsymbol{Z}_j\right)^{T}$; and $\tilde{\boldsymbol{C}}_{j(i)}$ is the mode-i matricization of tensor $\tilde{\mathcal{C}}_j$. Note that $\boldsymbol{R}$ is truncated.

The simplified proof of this proposition is shown in Appendix B. Note that we do not require the calculation of the Kronecker product $\boldsymbol{V}_d \otimes \cdots \boldsymbol{V}_{i+1} \otimes \boldsymbol{V}_{i-1} \otimes \boldsymbol{V}_1 \otimes \boldsymbol{Z}$ explicitly to find $\boldsymbol{Y}_{(i)}\boldsymbol{S}^{\mathrm{T}}$. In real implementation, we can use Proposition 1 to efficiently calculate the complete matrix using tensor products. Also, unlike the PCR in which the principal components of the output are learned independent of the inputs, the estimated basis matrices $\boldsymbol{V}_i\ (i = 1, \ldots, d)$ directly depend on the input tensors, ensuring correlation between the basis matrices and inputs. By combining Propositions 2 and 3, Algorithm 1 summarizes the estimation procedure for MTOT regression. This algorithm, in fact, combines the BCD algorithm with the ALS algorithm. In this algorithm assuming that the size of the core tensors and basis matrices (i.e., $\tilde{P}_{ji}\ (j = 1, 2, \ldots, p; i = 1, 2, \ldots, l_j)$ and $\tilde{Q}_k\ (k = 1, \ldots, d)$) are known, we first estimate U_{ji} and initialize V_k by performing Tucker decomposition over each input tensor $\mathcal{X}_j$ and the output tensor $\mathcal{Y}$. Next, by setting a convergence tolerance ϵ we estimate the core tensors and the V_k iteratively. The estimation of ranks are discussed in the next section.

3.1. Selection of Tuning Parameters

The proposed approach requires the selection of the values $\tilde{P}_{ji}\ (j = 1, 2, \ldots, p; i = 1, 2, \ldots, l_j)$ and $\tilde{Q}_k\ (k = 1, \ldots, d)$. For this purpose, we use the k-fold cross-validation method to find a tuple of parameters that minimizes the mean squared error. To minimize the CV-MSE, we use genetic algorithm (GA) with the constraint that the ranks are integer values and they are bounded between one and $P_{ji}\ (Q_k)$. The genetic algorithm is a fast and efficient method for minimizing a black-box function and results in accurate solutions. To initiate the ranks for the GA, we use the following heuristic: The rank of the ith mode of a tensor $\mathcal{A}$ is initiated by the number of principal components that explain 95% of variation of the mode-i matricization of the tensor, that is, $\boldsymbol{A}_{(i)}$. For all studies in the next sections, we perform 3-fold cross-validation (CV) and use the maximum of 30 generations in GA.

Algorithm 1 Estimation procedure for multiple tensor-on-tensor regression

1: Set $\epsilon = 10^{-6}$
2: Estimate U_{ji} using Tucker decomposition of $\mathcal{X}_j$ for all i and j
3: Initialize V_k for all k using Tucker decomposition of $\mathcal{Y}$
4: Compute B_j for all j and set $w_0 = \left|\left|Y_{(1)} - \sum_{j=1}^{p} X_{j(1)}B_j\right|\right|_F^2$
5: **do**
6: Estimate $\mathcal{C}_j$ for all $j = 1 : p$ using Proposition 2
7: Estimate V_k for all $k = 1 : d$ using Proposition 3
8: Compute B_j for all j and set $w_r = \left|\left|Y_{(1)} - \sum_{j=1}^{p} X_{j(1)}B_j\right|\right|_F^2$
9: **while** $|w_r - w_{r-1}| > \epsilon$

4. Performance Evaluation Using Simulation

This section contains two parts. In the first part, we only consider curve-on-curve regression and compare our proposed method to the PLS regression and function-on-function regression approach proposed by Luo and Qi (2017), designated as *sigComp*. The reason we compare our approach to sigComp is that sigComp can handle multiple functional inputs (curves) and learn the basis functions similar to our approach. In the second part, we conduct two simulation studies to evaluate the performance of the proposed method when the inputs or output are in the form of images or structured point clouds. In this part, we compare the proposed method with three benchmarks: The first benchmark is the TOT approach proposed by Lock (2017), which can roughly be viewed as a general form of sigComp. The second approach is HOPLS that is offered by Zhao et al. (2013) and is an extension of PLS to tensors. Because these approaches can only handle a single input tensor, when multiple inputs exist we perform a transformation to merge the inputs into one single tensor. To implement TOT and HOPLS, we use R package *MultiwayRegression* provided by Lock (2017) and the Matlab package provided by Zhao et al. (2013). The last benchmark is based on PCR similar to a benchmark considered in Fan et al. (2014). In this approach, we first matricize all the input and output tensors, then perform principal component analysis to reduce the dimensions of the problem by computing the PC scores of the first few principal components that explain at least ν percent of the variation in the data. Next, we perform

linear regression between the low-dimensional PC scores of both inputs and output. More formally, let $\boldsymbol{X}_{j(1)} \in \mathbb{R}^{M\times P_j}$ and $\boldsymbol{Y}_{(1)} \in \mathbb{R}^{M\times Q}$ denote the mode-1 matricization of the inputs and output, and $\boldsymbol{X} = \left[\boldsymbol{X}_{1(1)}, \boldsymbol{X}_{2(1)}, \ldots, \boldsymbol{X}_{p(1)}\right]$ be a concatenation of all the input matrices. We first compute the first G_x and G_y principal components of X and the response $\boldsymbol{Y}_{(1)}$. Next, the PC scores of the input $\boldsymbol{X}$ are calculated (a matrix in $\mathbb{R}^{M\times G_x}$) and are used to predict the matrix of the scores of the response function (a matrix in $\mathbb{R}^{M\times G_y}$). Then, given the PC scores of the new inputs, we use the fitted regression model to predict the response scores. Finally, we multiply the predicted response scores by the G_y principal components to obtain the original responses. The number of principal components G_x and G_y can be identified through a CV procedure. In this article, instead of CV over G_x and G_y directly, we perform CV over the percentage of variation the PCs explain, that is, v. For this purpose, we take the value of v from {85%, 90%, 95%, 99%, 99.5%} and take the v that minimizes the CV error. The standardized mean square prediction error (SMSPE) is used as a performance measure to compare the proposed method with the benchmarks. The SMSPE is defined as $\text{SMSPE} = \frac{||\mathcal{Y}-\hat{\mathcal{Y}}||_F^2}{||\mathcal{Y}||_F^2}$.

4.1. Simulation Studies for Curve-on-Curve Regression

In this simulation, we consider multiple functional (curve) predictors and multiple scalar predictors similar to the simulation study in Luo and Qi (2017). We first randomly generate $\left(\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_p\right)$ as follows:

$$\boldsymbol{B}_i(s,t) = \frac{1}{p^2}\left[\gamma_{1i}(t)\,\psi_{1i}(s) + \gamma_{2i}(t)\,\psi_{2i}(s) + \gamma_{3i}(t)\,\psi_{3i}(s)\right],$$

where $\gamma_{ki}(t)$ and $\psi_{ki}(s)$ $\left(k = 1, 2, 3;\ i = 1, \ldots, p\right)$ are Gaussian processes with covariance function $\Sigma_1\left(z, z'\right) = \left(1 + 20\left|z - z'\right| + \frac{1}{3}\left(20\left|z - z'\right|\right)^2\right)e^{-20|z-z'|}$. Next, we generate $p = 1, 3, 6$, functional predictors using the following procedure: Let S be a $p \times p$ matrix with the (i,j)th element equal to $\rho_c = 0,\ 0.75$ for $i \neq j$ and equal to one for diagonal elements. Next, we decompose $\boldsymbol{S} = \boldsymbol{\Delta}\boldsymbol{\Delta}^{\mathrm{T}}$, where $\boldsymbol{\Delta}$ is a $p \times p$ matrix and generate a set of curves $w_1, w_2, \ldots, w_p$ using a Gaussian process with covariance function $\Sigma_2\left(z, z'\right) = e^{-(2|z-z'|)^2}$. Finally, we generate the predictors at any given point s as

$$\left(x_1(s), \ldots, x_p(s)\right) = \left(w_1(s), w_2(s), \ldots, w_p(s)\right)\boldsymbol{\Delta}^{\mathrm{T}}.$$

With this formulation, each curve of $x_1(s), \ldots, x_p(s)$ is a Gaussian process with covariance function $\Sigma_2\left(s, s'\right)$, and for each s, the vector $\left(x_1(s), \ldots, x_p(s)\right)$ is a multivariate normal distribution with covariance $\boldsymbol{S}$. When $\rho_c = 0$, this vector becomes an independent vector of normally distributed variables and does not have a lower rank-representation. Figure 2 illustrates examples of the predictors when $p = 3$ for $\rho_c = 0,\ 0.75$. Please note that when $\rho_c = 0.75$ the vector $\left(x_1(s), \ldots, x_p(s)\right)$ have a lower-space representation and dimension reduction is possible. We also generate the scalar predictors $(u_1, \ldots, u_5)$ from a multivariate normal distribution with the mean vector zero and the covariance matrix with diagonal elements equal to 1 and off-diagonal elements equal to 0.5. The coefficients of the scalar variables denoted by $\alpha_i(t)$ $(i = 1, \ldots, 5)$ are generated from a Gaussian process with covariance function $\Sigma\left(t, t'\right) = e^{\{-5|t-t'|\}^2}$. Finally, we generate the response curves as

$$y(t) = \sum_{i=1}^{5}\alpha_i(t)\,u_i + \sum_{i=1}^{p}\int \boldsymbol{B}_i(s,t)\,x_i(s)\,ds + \epsilon(t),$$

where $\epsilon(t)$ is generated from a normal distribution with zero mean and $\sigma^2 = 0.1$. We generate all of the input and output curves over $0 < s < 2$ and $0 < t < 1$ and take the samples over an equidistant grid of size 100. Finally, we combine all p inputs and M samples generated from the explained approach into tensor $\mathcal{X} \in \mathbb{R}^{M\times p\times 100}$ and the output into tensor $\mathcal{Y} \in \mathbb{R}^{M\times 100}$.

For each combination of (p, ρ_c), we compare the performance of the proposed method with the method in Luo and Qi (2017), designated by sigComp, and PLS regression based on the MSPE and the mean square estimation error (MSEE). We do not compare our approach to PCR in this simulation because sigComp has already demonstrated superiority over PCR in simulation studies in Luo and Qi (2017). We implement the sigComp benchmark method using the R package *FRegSigCom* in which we use 50 spline bases for both the inputs and output and a default convergence tolerance. To calculate the MSPE and MSEE, we first generate a sample data of size $M_{\text{train}} = 400$ that is

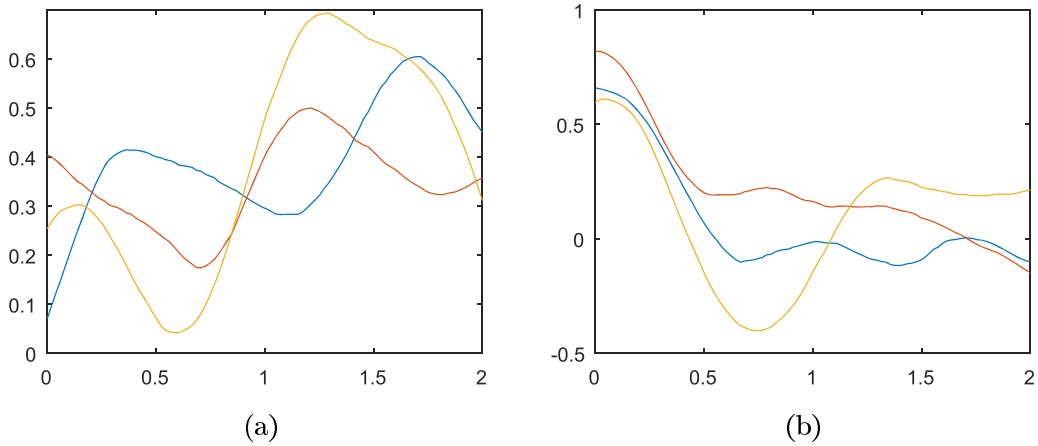

Figure 2. Example of the predictors when (a) $p = 3$, $\rho_c = 0$ and (b) $p = 3$, $\rho_c = 0.75$.

used to learn the model parameters. Next, we generate a testing sample of size $M_{\text{test}} = 100$ and calculate MSPE as

$$\text{MSPE} = \frac{1}{M_{\text{test}}} \sum_{j=1}^{M_{\text{test}}} \left(\frac{1}{100} \sum_{i=1}^{100} \left(y_j^{\text{test}}(t_i) - \hat{y}_j^{\text{test}}(t_i) \right)^2 \right)$$

and

$$\text{MSEE} = \frac{1}{M_{\text{test}}} \sum_{j=1}^{M_{\text{test}}} \frac{1}{100} \sum_{i=1}^{100} \left(y_j^{\text{test}}(t_i) - \epsilon(t_i) - \hat{y}_j(t_i) \right)^2.$$

We repeat this procedure 50 times to find the means and standard deviations of the MSPE and MSEE for each method. Table 1 reports the results at different values of ρ_c and different numbers of predictors, p. As reported, our proposed approach is superior to the sigComp and PLS in terms of MSPE and MSEE for all values of p. For example, when $p = 3$ and $\rho_c = 0.75$, the average MSPE and MSEE of the sigComp are 0.125 and 0.025, which are much larger than the corresponding values (0.113 and 0.013) achieved by MTOT. Please note that the performance of all three approaches deteriorates as the number of input variables increases, but all three methods show improved performance when the inputs are correlated. Figure 3 illustrates prediction examples obtained by each method, along with the true curve for different values of ρ_c and $p = 3$. As illustrated all of the approaches produce reasonably accurate predictions.

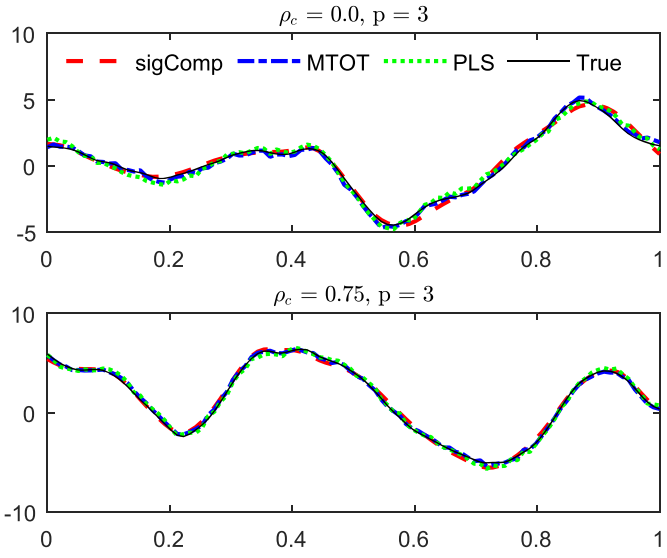

Figure 3. Prediction examples of sigComp, PLS, and MTOT.

4.2. Simulation Studies for Image and Structured Point-Cloud

4.2.1. Case I—Waveform Surface Simulation

We simulate waveform surfaces $\mathcal{Y}_i$ based on two input tensors, $\mathcal{X}_{1i} \in \mathbb{R}^{P_{11} \times P_{12} \times \cdots \times P_{1l_1}}$ and $\mathcal{X}_{2i} \in \mathbb{R}^{P_{21} \times P_{22} \times \cdots \times P_{2l_2}}$ $(i = 1, \ldots, M)$, where M is the number of samples. To generate the input tensors, we define $x_{kmj} = \frac{j}{P_{km}}$ $\left(k = 1, 2;\ m = 1, \ldots, l_k;\ j = 1, \ldots, P_{km}\right)$. Then, we set $\boldsymbol{U}_{km} = \left[\boldsymbol{u}_{km1}, \boldsymbol{u}_{km2}, \ldots, \boldsymbol{u}_{kmR_k}\right]$ $(k = 1, 2;\ m = 1, \ldots, l_k)$, where

$$\boldsymbol{u}_{kmt} = \begin{cases} \left[\cos\left(2\pi t x_{km1}\right), \ldots, \cos\left(2\pi t x_{kmP_{km}}\right)\right]^{\mathrm{T}} & \text{if } t \text{ is odd,} \\ \left[\sin\left(2\pi t x_{km1}\right), \ldots, \sin\left(2\pi t x_{kmP_{km}}\right)\right]^{\mathrm{T}} & \text{if } t \text{ is even.} \end{cases}$$

Next, we randomly simulate elements of a core tensor $\mathcal{D}_{ki}$ from a standard normal distribution. Then, we generate an input sample using the following model:

$$\mathcal{X}_{ki} = \mathcal{D}_{ki} \times_1 \boldsymbol{U}_{k1} \times_2 \cdots \times_{l_k} \boldsymbol{U}_{kl_k} \ (k = 1, 2;\ i = 1, \ldots, M).$$

To generate a response tensor, we first simulate the elements of a core tensor $\mathcal{C}_k$ from a standard normal distribution. Moreover, we set $\boldsymbol{V}_m = [\boldsymbol{v}_{m1}, \boldsymbol{v}_{m2}, \ldots, \boldsymbol{v}_{mR}]$ $(m = 1, \ldots, d)$, where

$$\boldsymbol{v}_{mt} = \begin{cases} \left[\cos\left(2\pi t y_{m1}\right), \ldots, \cos\left(2\pi t y_{mQ_m}\right)\right]^{\mathrm{T}} & \text{if } t \text{ is odd} \\ \left[\sin\left(2\pi t y_{m1}\right), \ldots, \sin\left(2\pi t y_{mQ_m}\right)\right]^{\mathrm{T}} & \text{if } t \text{ is even} \end{cases}$$

and $y_{mj} = \frac{j}{Q_m}$. Next, we define the parameter tensors $\mathcal{B}_k$ using the following expansion:

$$\mathcal{B}_k = \mathcal{C}_k \times_1 \boldsymbol{U}_{k1} \times_2 \cdots \times_{l_k} \boldsymbol{U}_{kl_k} \times_{l_k+1} \boldsymbol{V}_1 \times \cdots \times_{l_k+d} \boldsymbol{V}_d.$$

Finally, we simulate a response tensor as

$$\mathcal{Y}_i = \sum_{k=1}^{2} \mathcal{X}_{ki} * \mathcal{B}_k + \mathcal{E}_i,$$

where $\mathcal{E}_i$ is the error tensor whose elements are sampled from a normal distribution $\mathcal{N}\left(0, \sigma^2\right)$. For simulation purposes, we assume $\mathcal{X}_{1i} \in \mathbb{R}^{60}$, $\mathcal{X}_{2i} \in \mathbb{R}^{50 \times 50}$, and $\mathcal{Y}_i = \mathbb{R}^{60 \times 40}$. That is, we generate a response based on a profile and an image signal. Furthermore, we set $R_1 = 2$, $R_2 = 3$, and $R = 3$. This implies that $\mathcal{C}_1 \in \mathbb{R}^{2 \times 3 \times 3}$ and $\mathcal{C}_2 \in \mathbb{R}^{3 \times 3 \times 3 \times 3}$. Figure 4(a) illustrates examples of generated response surfaces. For this simulation study, we first generate a set of $M = 200$ data points. Then, we randomly divide the data into a set of 160 observations for training and a set of 40 observations for testing. We train the model using the training set, then calculate the SMSPE for the

Table 1. Comparison between the proposed method with PLS and sigComp proposed by Luo and Qi (2017).

		PLS		sigComp		MTOT	
p	ρ_c	MSPE	MSEE	MSPE	MSEE	MSPE	MSEE
1	0	0.125 (0.003)	0.025 (0.002)	0.109 (0.001)	0.009 (0.000)	**0.106** (0.001)	**0.006** (0.000)
3	0	0.158 (0.004)	0.059 (0.003)	0.128 (0.003)	0.029 (0.002)	**0.117** (0.002)	**0.018** (0.001)
	0.75	0.149 (0.004)	0.051(0.003)	0.125 (0.003)	0.025 (0.002)	**0.113** (0.002)	**0.013** (0.001)
6	0	0.253 (0.007)	0.156 (0.006)	0.147 (0.005)	0.046 (0.003)	**0.134** (0.003)	**0.035** (0.003)
	0.75	0.150 (0.004)	0.052 (0.003)	0.131 (0.004)	0.032 (0.002)	**0.128**(0.003)	**0.029** (0.002)

NOTE: The bold indicates the value that is the smallest in each row comparing to its corresponding values in other columns.

proposed method and benchmarks based on the test data. We repeat this procedure 50 times to capture the variance of the SMSPE. Please note that we perform CV only once using one random partition of the data before repeating the simulation procedure. To prepare data for the TOT and HOPLS, three steps are performed: First, because the dimension of the curve inputs (1×60) and the image inputs (50×50) do not match, we randomly select 50 points out of 60 to reduce the curve dimension to 50. Second, we replicate each curve 50 times to generate 50×50 images. Third, for each sample, we merge the image constructed from the curve and the image input to construct a tensor of size $50 \times 50 \times 2$. Combining all of the samples, we obtain an input tensor of size $M \times 50 \times 50 \times 2$, where M is the sample size.

4.2.2. Case II—Truncated Cone Simulation

We simulate a truncated cone based on a set of scalars and simple profile data in a three-dimensional cylindrical coordinate system (r, ϕ, z), where $\phi \in [0, 2\pi]$ and $z \in [0, 1]$. We first generate an equidistant grid of $I_1 \times I_2$ over the (ϕ, z) space by setting $\phi_i = \frac{2\pi i}{I_1}$ $(i = 1, \ldots, I_1)$ and $z_j = \frac{j}{I_2}$ $\left(j = 1, \ldots, I_2\right)$. Specifically, we set $I_1 = I_2 = 200$. Next, we simulate the truncated cone over the grid by

$$r(\phi, z) = \frac{r_0 + z \tan\theta}{\sqrt{1 - e^2 \cos^2 \phi}} + c\left(z^2 - z\right) + \epsilon(\phi, z), \quad (7)$$

where r_0 is the radii of the upper circle of the truncated cone, θ is the angle of the cone, e is the eccentricity of the top and bottom surfaces, c is the side curvatures of the truncated cone, and $\epsilon(\phi, z)$ is process noise simulated from $\mathcal{N}\left(0, \sigma^2\right)$. Figure 4(b) illustrates examples of generated truncated cones. We assume that the parameters of the truncated cone are specific features obtained from a scalar and three simple profile data. In particular, we assume that the scalar predictor is $x_{1i} = r_{0i}$ and the profile predictors are $x_{2i}(z) = z \tan\theta$, $x_{3i}(\phi) = e^2 \cos^2 \phi$, and $x_{4i}(z) = c\left(z^2 - z\right)$; $i = 1, \ldots, M$. That is, the inputs are one scalar and three profiles. We simulate these profiles for training purposes by setting the parameters as follows: We set $r_0 \in \{1.1, 1.3, 1.5\}$, $\theta \in \left\{0, \frac{\pi}{8}, \frac{\pi}{4}\right\}$, $e \in \{0, 0.3, 0.5\}$, $c \in \{-1, 0, 1\}$, and consider a full factorial design to generate 81 samples. That is, for each combination of parameters (e.g., {1.1, 0, 0.3, −1}), we generate a sample containing one scalar value and three profiles. We represent each of the inputs by a matrix (a tensor of order 2) to obtain four input matrices X_1, X_2, X_3, and X_4, where $X_1 \in \mathbb{R}^{81\times 1}$ and $X_i \in \mathbb{R}^{81\times 200}$ $(i = 2, 3, 4)$. Next, we will combine three of the matrices X_2, X_3, and X_4 into one tensor $\mathcal{X} \in \mathbb{R}^{81\times 3\times 200}$. Therefore, for our approach we have two inputs X_1 and $\mathcal{X}$. Finally, we generate the test data by sampling the truncated cone parameters as follows: We assume $r \sim U(1.1, 1.5)$, $\theta \sim U\left(0, \frac{\pi}{4}\right)$, $e \sim U(0, 0.5)$, and $c \sim U(-1, 1)$, where $U(a, b)$ denotes a uniform distribution over the interval $[a, b]$, and sample each parameter from its corresponding distribution. In this simulation, we first train the model using the generated training data. Next, we generate a set of 1000 test data. We predict the truncated cone based on the input values in the test data and calculate the SMSPE for each predicted cone. To prepare the data for TOT and HOPLS, we first replicate the column of X_1 to generate a matrix of size 81×200, then merge this matrix with the other three matrices to construct an input tensor of size $81 \times 4 \times 200$. This tensor is used as an input in the TOT and HOPLS.

In each case, we compare the proposed method with benchmarks based on the SMSPE calculated at different levels of noise σ. Tables 2 and 3 report the average and standard deviation of SMSPE (or its logarithm), along with the average running time of each algorithm for the simulation cases I and II, respectively. In Table 3, we report the average and standard deviation of the logarithm of the SMSPE for better comparison of the values. Note that the SMSPE is a standardized error and should not directly be compared to the variance. In almost all cases, the MTOT has the smallest prediction errors, reflecting the advantage of our method in terms of prediction. Furthermore, with the increase in σ, all methods illustrate a larger SMSPE in all cases.

In the first case, the TOT illustrates a prediction performance comparable to our method at a cost of a much longer running time. For example, when $\sigma = 0.2$, TOT requires about 147.33 sec to reach the SMSPE of 0.0168, obtained in 5.51 sec by MTOT. The HOPLS approach is inferior to both MTOT and TOT in this case. One reason for this inferior performance of HOPLS is the fact that it allows only a single input tensor. Combining the inputs in one tensor causes a challenge for HOPLS to identify a shared subspace between the inputs and the output. Finally, to evaluate the performance of our approach in tuning the

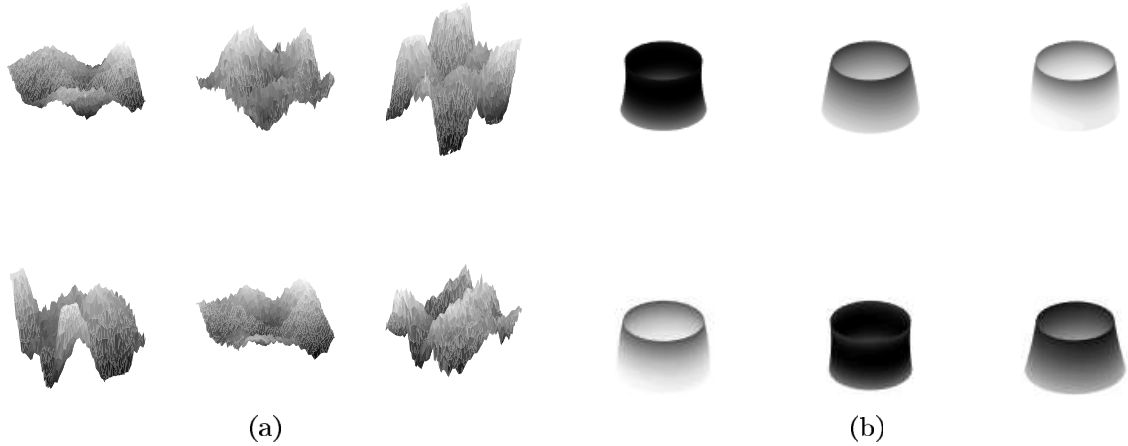

Figure 4. Examples of generated output for the simulation study (a) Case I—waveform surface and (b) Case II—truncated cone.

Table 2. Comparison between the proposed method (MTOT) and the benchmarks in Case I with the waveform response.

	PCR		HOPLS		TOT		MTOT	
σ	SMSPE	Time (sec)	SMSPE	Time (sec)	SMSPE	Time (sec)	SMSPE	Time (sec)
0.1	0.0057 (0.002)	0.03 (0.00)	0.0068 (0.002)	27.61 (1.91)	0.0046 (0.001)	154.98 (17.94)	**0.0041** (0.001)	5.54 (0.05)
0.2	0.0199 (0.005)	0.04 (0.00)	0.0225 (0.006)	24.65 (2.38)	0.0170 (0.004)	147.33 (2.47)	**0.0168** (0.004)	5.51 (0.05)
0.3	0.0455 (0.010)	0.04 (0.00)	0.0528 (0.014)	25.06 (2.29)	0.0399 (0.009)	149.03 (1.36)	**0.0392** (0.009)	5.67 (0.06)
0.4	0.0773 (0.023)	0.04 (0.00)	0.0855 (0.016)	25.27 (1.29)	0.0678 (0.014)	149.13 (0.96)	**0.0669** (0.019)	5.73 (0.06)
0.5	0.1186 (0.022)	0.04 (0.00)	0.1082 (0.023)	26.67 (1.68)	0.1036 (0.023)	146.17 (0.95)	**0.1032** (0.020)	5.84 (0.06)
0.6	0.1670 (0.033)	0.04 (0.00)	0.1596 (0.027)	28.62 (1.95)	0.1456 (0.031)	147.75 (1.81)	**0.1451** (0.029)	5.97 (0.08)

NOTE: The TOT requires a much larger running time to achieve the same level of prediction error as the MTOT. The bold indicates the value that is the smallest in each row comparing to its corresponding values in other columns.

Table 3. Comparison between the proposed method and the benchmarks in Case II with truncated cone.

	PCR		HOPLS		TOT		MTOT	
σ	log(SMSPE)	Time (sec)	log(SMSPE)	Time (sec)	log(SMSPE)	Time (sec)	log(SMSPE)	Time (sec)
0.01	−5.555 (0.986)	0.05 (0.00)	−6.632 (1.19)	4.40 (0.24)	−5.249 (1.326)	23.58 (5.93)	**−8.095** (1.196)	3.82 (0.09)
0.02	−5.509 (0.937)	0.07 (0.00)	−6.507 (0.92)	4.35 (0.25)	−5.197 (1.254)	27.94 (6.11)	**−7.629** (0.869)	3.92 (0.10)
0.03	−5.441 (0.879)	0.08 (0.00)	−6.347 (0.81)	4.36 (0.24)	−5.127 (1.175)	29.11 (8.46)	**−7.215** (0.666)	3.93 (0.12)
0.04	−5.360 (0.819)	0.06 (0.00)	−6.172 (0.72)	4.37 (0.25)	−5.048 (1.097)	33.61 (9.02)	**−6.856** (0.537)	3.95 (0.14)
0.05	−5.269 (0.762)	0.07 (0.00)	−5.994 (0.65)	4.37 (0.26)	−4.963 (1.023)	34.29 (14.55)	**−6.543** (0.454)	3.99 (0.13)
0.06	−5.173 (0.710)	0.07 (0.00)	−5..818 (0.59)	4.36 (0.24)	−4.875 (0.956)	37.43 (15.19)	**−6.266** (0.402)	3.95 (0.14)

NOTE: Due to the difference between the input and the output rank, the performance of TOT is significantly worse than the MTOT. The PCR is very fast in estimation, but the prediction accuracy is not as appealing as the MTOT.

Table 4. Estimated ranks of the two inputs in simulation Case I at different levels of noise.

	0.1	0.2	0.3	0.4	0.5	0.6
$\mathcal{C}_1$	(3, 4, 5)	(3, 4, 5)	(4, 5, 6)	(5, 6, 6)	(6, 8, 6)	(5, 4, 7)
$\mathcal{C}_2$	(3, 4, 4, 5)	(4, 4, 4, 5)	(4, 5, 5, 6)	(5, 6, 6, 6)	(4, 6, 8, 6)	(6, 7, 4, 7)

parameters, we report the selected ranks of each of the inputs and the output (i.e., dimensions of $\mathcal{C}_1$ and $\mathcal{C}_2$) obtained by CV in Table 4. These values can be compared to the true dimensions of the two core tensors given in the data generation description of this simulation case. Please note that, we only perform CV once for a randomly selected partition of the data and we take the estimated ranks for all the repetitions of the simulation. As it is reported in Table 4 the estimated dimensions of the core tensors are increasing as the level of noise increases. The average time (over different levels of noise) to perform CV are 2054.8, 4453.4, 6579.3, and 0.6221 sec for MTOT, HOPLS, TOT, and PCR, respectively.

The performance of all three benchmarks (PCR, HOPLS, and TOT) are significantly worse than MTOT in the second case. The inferior performance of TOT is due to both its restriction on selecting the same rank for both the input and output and the fact that the CP decomposition it uses does not consider the correlation between multiple modes. The HOPLS requires combining X_1 and $\mathcal{X}$ into one tensor that may causes its inferior performance. The parameters of each of the models are obtained by CV. The average time (over different levels of noise) to perform CV are 1528.8, 1459.4, 1019.6, and 1.0678 sec for MTOT, HOPLS, TOT, and PCR, respectively.

5. Case Study

In this section, we present two case studies. In the first study, we employ our approach to predict the overlay errors based on the wafer shape is conducted. The data for this cases study is simulated based on the results in Brunner et al. (2013). In the second case study, we evaluate our proposed method in predicting air/gas ratio of an engine given a set of signals obtained from that engine.

5.1. Simulated Overlay Error Prediction

In semiconductor manufacturing, patterns are printed layer by layer over a wafer in a sequence of deposition, etching, and lithographic processes to manufacture transistors (Nishi and Doering 2000). Many of these processes induce stress variations across the wafer, distorting/changing the wafer shape (Brunner et al. 2013; Turner, Ramkhalawon, and Sinha 2013). Figure 5 illustrates a simplified sequence of processes, causing the overlay error in the patterned wafers. In the first step, a layer is deposited over the wafer and exposed to rapid thermal annealing, causing a curvature in the free-state wafer. The wafer is then chucked flat and patterned in a lithographic process. Next, to generate a second layer pattern, a new layer is deposited, changing the wafer shape. Finally, in the lithography step, the flattened wafer is patterned. Because the wafer is flattened, the first pattern distance increases, but the new pattern is printed with the same distance L, generating a misalignment between patterns. The overlay error caused by lower order distortions can be corrected by most of the exposure tools. For this purpose, the alignment positions of several targets are measured and used to fit a linear overlay error model (Brunner et al. 2013):

$$\begin{cases} \Delta x = T_x - \theta_x y + M_x x & \text{error in x coordinate,} \\ \Delta y = T_y + \theta_y x + M_y y & \text{error in y coordinate,} \end{cases} \tag{8}$$

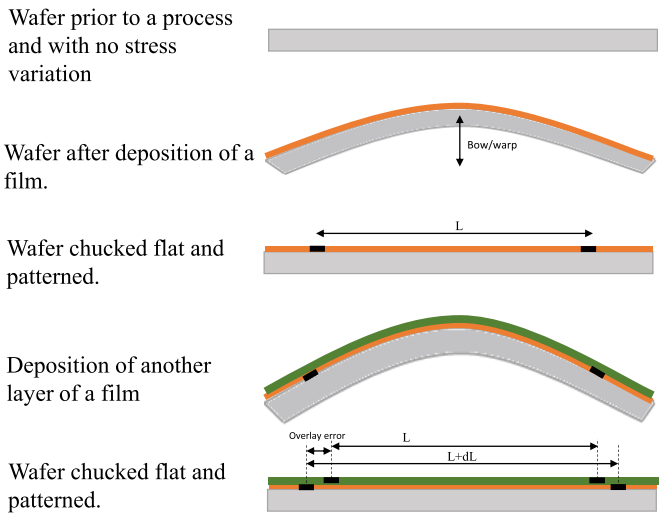

Figure 5. Process of a wafer, which causes shape variation and consequently overlay error.

where x and y identify the position of the target point over the wafer, T_x and T_y are transition errors, θ_x and θ_y relate to rotation error, and M_x and M_y are isotropic magnification errors pertaining to the wafer size change or wafer expansion due to processing. The fitted model is then used to correct the overlay errors. This model, however, can only correct the overlay error induced by a uniform stress field and fails to compensate for overlay errors caused by high-order distortions (Brunner et al. 2013). Therefore, developing a model that can relate the overlay error to higher order patterns in the wafer shape is essential for better overlay correction.

In this case study, we use our proposed method to predict the overlay error based on the wafer shape data. Such predictions can be fed forward to the exposure tools to result in a better correction strategy. In practice, the wafer shape is measured using a patterned wafer geometry (PWG) tool, and the overlay error is measured using standard optical methods (Brunner et al. 2013). Both the wafer shapes and the overlay errors (in each coordinate, x or y) can be presented as image data. In this case study, we follow the procedure and results suggested and verified (through both experiments and finite element [FE] analysis) by Brunner et al. (2013) to generate surrogate data of overlay errors $\left(\text{PIR}\left(x,y\right)\right)$ based on the wafer shape prior to two lithography steps $\left(w_1\left(x,y\right), w_2\left(x,y\right)\right)$. The data generation procedure is elaborated in Appendix C.

Based on the described procedure in Appendix C, we generate a set of 500 training observations, that is, wafer shapes and overlay errors, $\left\{\left(w_{1i}\left(x,y\right), w_{2i}\left(x,y\right), \text{PIR}_i\right)\right\}_{i=1}^{M=500}$, and employ our proposed method to estimate the PIR_i based on $\left(w_{1i}\left(x,y\right), w_{2i}\left(x,y\right)\right)$. Because in our simulated data $w_{1i}\left(x,y\right)$ remains fixed, we consider $w_i\left(x,y\right) = w_{2i}\left(x,y\right) - w_{1i}\left(x,y\right)$ as the predictor. We also generate 100 observations as the test dataset. The MPSE obtained from the testing data is used as the performance criterion. We repeat the simulations 50 times and record the MSPE values. Because our proposed methodology assumes that the shapes are observed over a grid, we transform the data to the polar coordinate prior to modeling. In the polar space, each shape is observed over a grid of 100×200 (100 in the radial direction and 200 in the angular direction, with overall 20,000 pixels). Unfortunately, the TOT approach proposed by Lock (2017) failed to run with this size of images due to its high space complexity. Therefore, we compared our approach with PCR and HOPLS offered by Zhao et al. (2013). Figure 6 illustrates an example of the original and predicted corrected overlay error image, along with the prediction error. As it is illustrated, the proposed method predicts the original surface more accurately, with smaller errors across the wafer. Figure 7 illustrates the boxplots of the logarithm of the standard prediction mean squared error calculated over 50 replications. The results show that the proposed method is superior to both HOPLS and PCR in prediction of the image. As an example, the average of log(SMSE) over the replications is -9.047 for the proposed method and -8.455 and -8.234 for HOPLS and PCR, respectively. Please note that the reported results for the HOPLS was the best feasible one. That is, for some set of parameters the HOPLS fails to run due to the high space complexity.

Furthermore, the amount of time the HOPLS takes to train the model and to predict an output is significantly longer than our proposed method. Table 5 reports the amount of time each method takes to complete training a model (without CV) and create a prediction. As it is reported, MTOT requires significantly less training and prediction time than HOPLS. For example, while it takes 50.96 sec for MTOT to train a model, it takes about 1468.81 sec for HOPLS to complete training.

5.2. Vehicle Engine Lambda Sensor Prediction

The NOx Storage Catalyst (NSC) is a catalyst system by which the exhaust gas is treated in a two-phase process: (i) adsorption: NOx molecules are trapped by an adsorber based on a catalytic converter support, coated with a special washcoat containing zeolites; (ii) regeneration: when the adsorber is saturated, the stored NOx is catalytically reduced. During the regeneration phase, of duration ranging between 30 and 90 sec, the electronic control unit of the engine is programmed to maintain the combustion process in a rich air-to-fuel status. This status is related to the amount of oxygen present during the combustion process. The relative air/fuel ratio (λ) measured upstream of the NSC is assumed as an indicator for a correct regeneration phase. During regeneration, λ signal should assume values in the set-point interval $0.9 \div 0.95$. However, faults could occur, detected by a λ signal falling below an acceptability threshold ($0.8 \div 0.9$). This kind of fault, which is called λ-undershoot, worsens NSC performance during the regeneration phase (Pacella 2018). It is known that the λ value is dependent on several other engine state values such as engine inner torque, rotational speed, and quantity of fuel injected. Developing a model that estimates lambda signal based on other engine operation signals can help to better adjust and control the engine operation condition and to identify faulty sensors. Figure 8 illustrates examples of the input signals and the lambda sensor readings during the regeneration phase. This data are simulated based upon a set of real data to mask the original engine data.

In this case study, we employ our proposed method to estimate Lambda curve and to compare it to the results of HOPLS, sigcomp, and PLS. For this purpose, we consider 200 samples, each containing five input signals and one lambda signal as

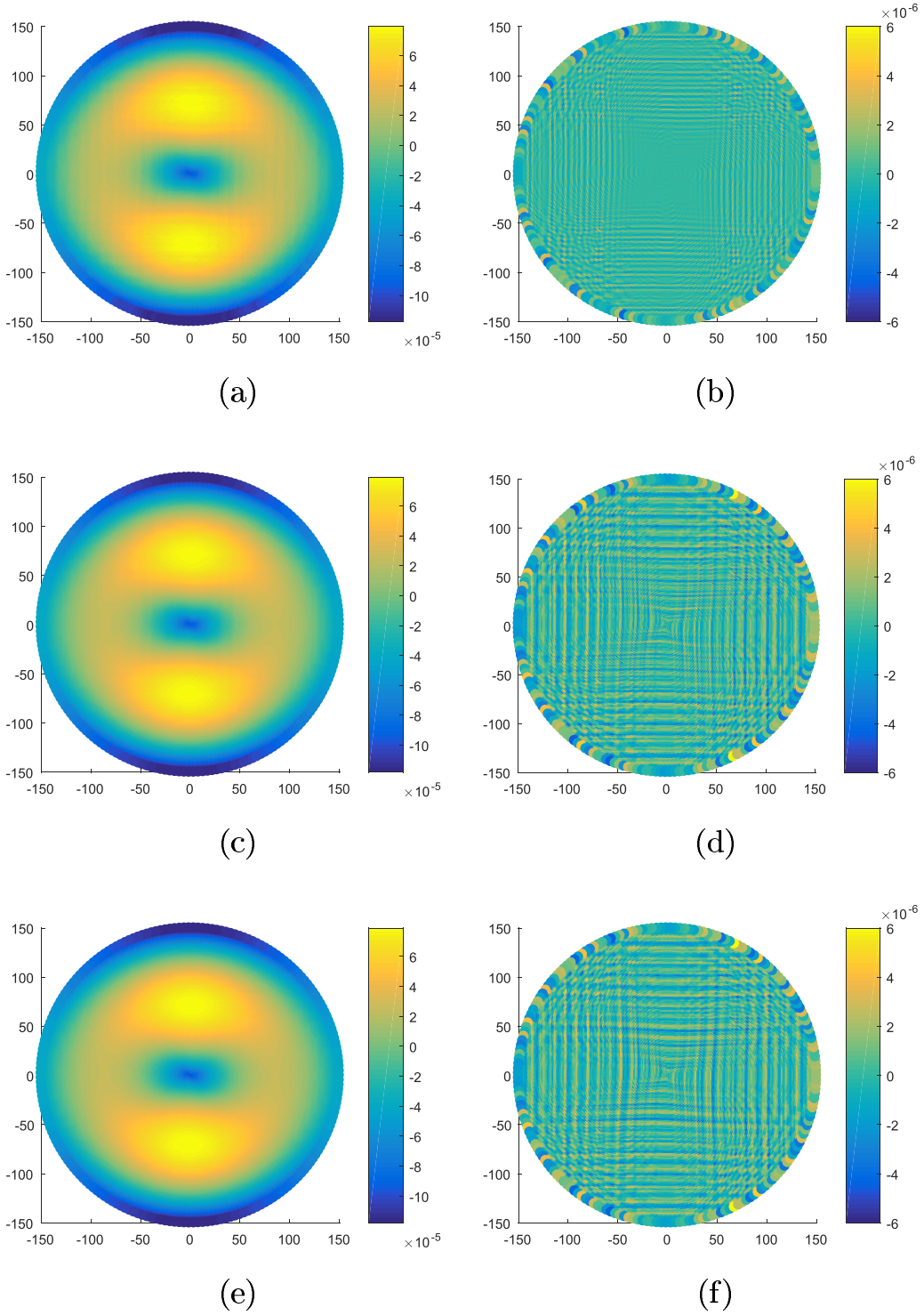

Figure 6. Example of (a) prediction of MTOT, (b) MTOT prediction error, (c) prediction of HOPLS, (d) HOPLS prediction error, (e) prediction of PCR, (f) PCR prediction error of the overlay error.

the output. All signals are measured over a 2-sec interval with 203 measurement points. To develop a predictive model, we randomly partition samples into training (80%) and testing (20%) data. The training data is used for CV and training a model and testing data is employed for model evaluation. For our approach and HOPLS, we create an input tensor of size $160 \times 203 \times 5$ for training and $40 \times 203 \times 5$ for testing. We repeat this procedure 30 times and calculate prediction error for MTOT as well as benchmarks. Figure 9 depicts the boxplot of SMSPE obtained over 30 replications of simulation for the proposed method and the benchmarks. As it is illustrated, the proposed approach outperforms the benchmarks in predicting the output curve. Figure 10 depicts an example of a λ curve along with the predictions obtained by the proposed method and the benchmarks. As it is illustrated, all methods perform well in predicting the curve. Nevertheless, it appeared that all the benchmarks underfit the curve toward the final 0.25 sec of the curve.

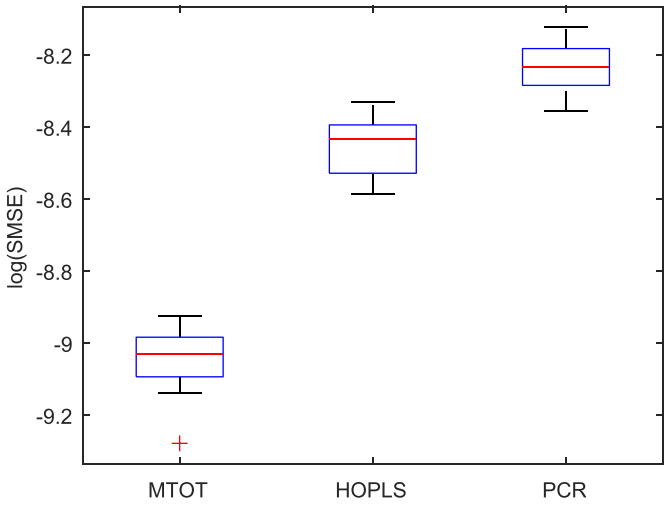

Figure 7. Logarithm of the prediction mean square error calculated for the test data over 50 replications. The proposed method (MTOT) illustrates significantly lower standard prediction error than the benchmarks.

Table 5. Training and prediction time of the proposed method and the benchmarks.

	Training time	Prediction time
MTOT	50.96 (1.09)	8.92 (0.18)
HOPLS	1468.81 (198.57)	85.74 (12.88)
PCR	3.04 (0.06)	0.05 (0.01)

NOTE: The values in the parenthesis report the standard deviation of running time.

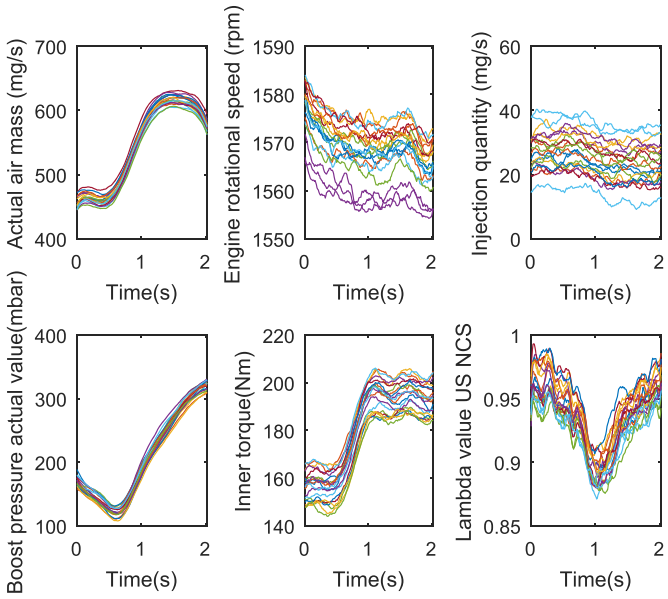

Figure 8. Examples of the input signals and the lambda sensor readings during the regeneration phase.

6. Conclusion

This article proposed an MTOT approach for modeling processes with a heterogeneous set of input variables and an output that can be measured by a scalar, curve, image, or point-cloud, etc. The proposed method represents each of the inputs as well as the output by tensors, and formulates a multiple linear regression model over these tensors. To estimate the parameters, a least square loss function is defined. To avoid overfitting,

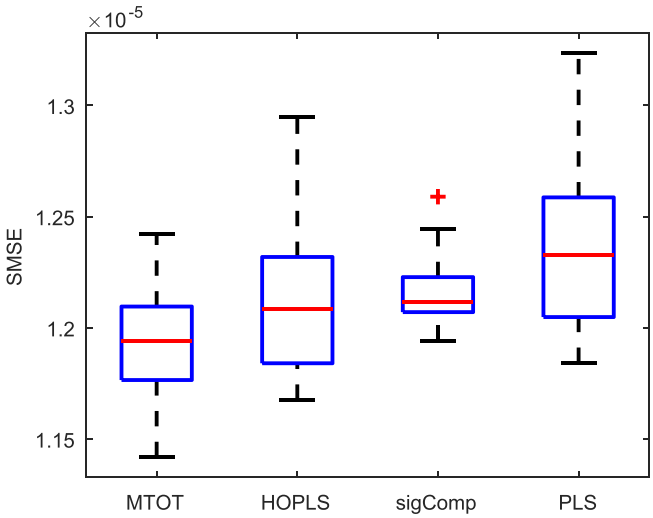

Figure 9. MSPE of predicting the lambda curves using MTOT, HOPLS, sigComp, and PLS.

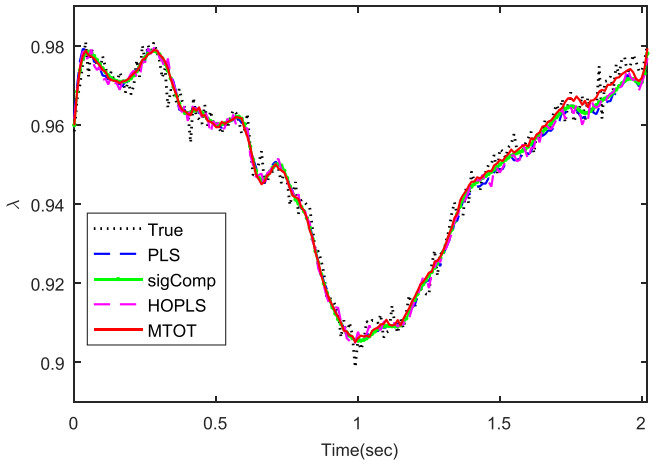

Figure 10. Example of the λ curve prediction by the proposed method and the benchmarks.

the proposed method decomposes the regression parameters through a low-dimensional set of basis matrices that spans the input and output spaces. Next, the basis matrices, along with their expansion coefficients, are learned by minimizing the loss function. The orthogonality condition is imposed over the output bases to assure identifiability and interpretability. To solve the minimization problem, first, a closed-form solution is derived for both the bases and their coefficients. Second, the block coordinate descent (BCD) approach combined with the ALS algorithm is applied. The proposed approach is capable of combining different forms of inputs (e.g., an image, a curve, and a scalar) to estimate an output (e.g., a scalar, a curve, or an image, etc.) as demonstrated in first three simulation studies. For example, in the first and third simulation studies, we combined the scalar and profile inputs to estimate a profile and a point cloud, respectively; and in the second simulation study, a profile and an image are integrated to predict an image.

To evaluate the performance of the proposed method, we conducted four simulation studies and a case study. In our first simulation study, we compared our proposed method with the function-on-function approach proposed by Luo and Qi (2017). This simulation considered scalar and curve inputs since the

benchmark can only handle those form of data. Next, we performed three other simulations to evaluate the performance of the proposed method when the inputs or outputs are images or point clouds. In these simulation studies, the proposed approach was compared with PCR and TOT regression, and showed superior performance in terms of MSPE. We also evaluated our proposed method using a set of surrogate data generated according to the manufacturing process of semiconductors. We simulated the shape and overlay errors for several wafers and applied the proposed method to estimate the overlay errors based on the wafer shapes measured prior to the lithography steps. Results showed that the proposed method performed significantly better than the PCR in predicting the overlay errors.

As a future work, including penalties such as lasso for sparsity and group lasso for variable selection and imposing roughness penalties over the basis matrices may improve the prediction results and can be further studied.

Supplementary Materials

Supplementary materials contain Appendices A–C that show the proofs of the propositions and the data generation approach for overlay error simulation.

References

Anandkumar, A., Hsu, D. J., Janzamin, M., and Kakade, S. M. (2013), "When Are Overcomplete Topic Models Identifiable? Uniqueness of Tensor Tucker Decompositions With Structured Sparsity," in *Advances in Neural Information Processing Systems*, pp. 1986–1994. [4]

Balageas, D., Fritzen, C.-P., and Güemes, A. (2010), *Structural Health Monitoring* (Vol. 90), Hoboken, NJ: Wiley. [1]

Beck, A., and Teboulle, M. (2009), "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, 2, 183–202. [5]

Bellon, E., Van Cleynenbreugel, J., Delaere, D., Houtput, W., Smet, M., Marchal, G., and Suetens, P. (1995), "Experimental Teleradiology. Novel Telematics Services Using Image Processing, Hypermedia and Remote Cooperation to Improve Image-Based Medical Decision Making," *Journal of Telemedicine and Telecare*, 1, 100–110. [1]

Bro, R. (1996), "Multiway Calibration. Multilinear PLS," *Journal of Chemometrics*, 10, 47–61. [2]

Brunner, T. A., Menon, V. C., Wong, C. W., Gluschenkov, O., Belyansky, M. P., Felix, N. M., Ausschnitt, C. P., Vukkadala, P., Veeraraghavan, S., and Sinha, J. K. (2013), "Characterization of Wafer Geometry and Overlay Error on Silicon Wafers With Nonuniform Stress," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, 12, 043002. [9,10]

Chen, H., Raskutti, G., and Yuan, M. (2019), "Non-Convex Projected Gradient Descent for Generalized Low-Rank Tensor Regression," *The Journal of Machine Learning Research*, 20, 172–208. [2]

Chiou, J.-M., Müller, H.-G., and Wang, J.-L. (2004), "Functional Response Models," *Statistica Sinica*, 14, 675–693. [4]

Fan, Y., Foutz, N., James, G. M., and Jank, W. (2014), "Functional Response Additive Model Estimation With Online Virtual Stock Markets," *The Annals of Applied Statistics*, 8, 2435–2460. [1,2,5]

Gorgannejad, S., Gahrooei, M. R., Paynabar, K., and Neu, R. W. (2019), "Quantitative Prediction of the Aged State of Ni-Base Superalloys Using PCA and Tensor Regression," *Acta Materialia*, 165, 259–269. [1]

He, G., Müller, H. G., and Wang, J. L. (2000), "Extending Correlation and Regression From Multivariate to Functional Data," in *Asymptotics in Statistics and Probability*, ed. M. L. Puri, Netherlands: VSP, pp. 197–210. [4]

Ivanescu, A. E., Staicu, A.-M., Scheipl, F., and Greven, S. (2015), "Penalized Function-on-Function Regression," *Computational Statistics*, 30, 539–568. [2]

Khosravani, A., Cecen, A., and Kalidindi, S. R. (2017), "Development of High Throughput Assays for Establishing Process-Structure-Property Linkages in Multiphase Polycrystalline Metals: Application to Dual-Phase Steels," *Acta Materialia*, 123, 55–69. [1]

Kiers, H. A. L. (2000), "Towards a Standardized Notation and Terminology in Multiway Analysis," *Journal of Chemometrics*, 14, 105–122. [2]

Kolda, T. G. (2006), "Multilinear Operators for Higher-Order Decompositions," Tech. Rep., Sandia National Laboratories. [3,4]

Li, X., Zhou, H., and Li, L. (2013), "Tucker Tensor Regression and Neuroimaging Analysis," arXiv no. 1304.5637. [2]

Liang, H., Wu, H., and Carroll, R. J. (2003), "The Relationship Between Virologic and Immunologic Responses in AIDS Clinical Research Using Mixed-Effects Varying-Coefficient Models With Measurement Error," *Biostatistics*, 4, 297–312. [1,2]

Lock, E. F. (2017), "Tensor-on-Tensor Regression," arXiv no. 1701.01037. [2,4,5,10]

Luo, R., and Qi, X. (2017), "Function-on-Function Linear Regression by Signal Compression," *Journal of the American Statistical Association*, 112, 690–705. [1,2,5,6,7,12]

Nishi, Y., and Doering, R. (2000), *Handbook of Semiconductor Manufacturing Technology*, Boca Raton, FL: CRC Press. [9]

Pacella, M. (2018), "Unsupervised Classification of Multichannel Profile Data Using PCA: An Application to an Emission Control System," *Computers & Industrial Engineering*, 122, 161–169. [10]

Ramsay, J., and Silverman, B. W. (2005), *Functional Data Analysis*, New York: Springer. [1,2]

Raskutti, G., Yuan, M., and Chen, H. (2019), "Convex Regularization for High-Dimensional Multiresponse Tensor Regression," *The Annals of Statistics*, 47, 1554–1584. [2]

Sapienza, A., Panisson, A., Wu, J., Gauvin, L., and Cattuto, C. (2015), "Detecting Anomalies in Time-Varying Networks Using Tensor Decomposition," in *IEEE International Conference on Data Mining Workshop (ICDMW)*, pp. 516–523. [2]

Sharan, V., and Valiant, G. (2017), "Orthogonalized ALS: A Theoretically Principled Tensor Decomposition Algorithm for Practical Use," arXiv no. 1703.01804. [4]

Sun, J., Papadimitriou, S., and Philip, S. Y. (2006), "Window-Based Tensor Analysis on High-Dimensional and Multi-Aspect Streams," in *ICDM*, pp. 1076–1080. [2]

Szatvanyi, G., Duchesne, C., and Bartolacci, G. (2006), "Multivariate Image Analysis of Flames for Product Quality and Combustion Control in Rotary Kilns," *Industrial & Engineering Chemistry Research*, 45, 4706–4715. [1]

Tucker, L. R. (1963), "Implications of Factor Analysis of Three-Way Matrices for Measurement of Change," *Problems in Measuring Change*, 15, 122–137. [2]

Turner, K. T., Ramkhalawon, R., and Sinha, J. K. (2013), "Role of Wafer Geometry in Wafer Chucking," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, 12, 023007. [1,9]

Wójcik, W., and Kotyra, A. (2009), "Combustion Diagnosis by Image Processing," *Photonics Letters of Poland*, 1, 40–42. [1]

Yan, H., Paynabar, K., and Pacella, M. (2019), "Structured Point Cloud Data Analysis via Regularized Tensor Regression for Process Modeling and Optimization," *Technometrics*, 61, 385–395. [2,4]

Yan, H., Paynabar, K., and Shi, J. (2015), "Image-Based Process Monitoring Using Low-Rank Tensor Decomposition," *IEEE Transactions on Automation Science and Engineering*, 12, 216–227. [2]

Yao, F., Müller, H.-G., and Wang, J.-L. (2005), "Functional Linear Regression Analysis for Longitudinal Data," *The Annals of Statistics*, 33, 2873–2903. [2]

Yu, H., and MacGregor, J. F. (2003), "Multivariate Image Analysis and Regression for Prediction of Coating Content and Distribution in the Production of Snack Foods," *Chemometrics and Intelligent Laboratory Systems*, 67, 125–144. [1]

Zhao, Q., Caiafa, C. F, Mandic, D. P., Chao, Z. C., Nagasaka, Y., Fujii, N., Zhang, L., and Cichocki, A. (2013), "Higher Order Partial Least Squares (HOPLS): A Generalized Multilinear Regression Method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 1660–1673. [2,5,10]

Zhou, H., Li, L., and Zhu, H. (2013), "Tensor Regression With Applications in Neuroimaging Data Analysis," *Journal of the American Statistical Association*, 108, 540–552. [2]