MODELING AND IMPROVEMENT OF PROCESSES WITH HETEROGENEOUS SOURCES OF DATA

A Dissertation
Presented to
The Academic Faculty

By

Mostafa Reisi Gahrooei

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology

August 2019

MODELING AND IMPROVEMENT OF PROCESSES WITH HETEROGENEOUS SOURCES OF DATA

Approved by:

Dr. Kamran Paynabar, Advisor
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Jianjun (Jan) Shi, Advisor
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Nagi Gebraeel
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Yajun Mei
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Bianca Maria Colosimo
Dipartimento di Meccanica
Politenico Milano

Date Approved: May 09, 2019

Become who you are. Do what only you can do. Be the master and the sculptor of yourself.

Friedrich Nietzsche

To my parents and my wife, Mojdeh,

for their continuous love and support.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisors Dr. Kamran Paynabar and Dr. Jianjun Shi for their constant presence and support throughout my PhD studies. During the most challenging times of my doctoral study, they provided the greatest guidance and encouragement for me to stay on the right track and move forward. Dr. Paynabar and Dr. Shi have set excellent examples of a research advisor for me, and I am privileged to have had the opportunity to study and conduct research under their supervision and mentorship. I especially thank Dr. Paynabar for his trust in me that changed my life and the flexibility in his mentorship that empowered me to become an independent researcher.

I would like to thank my committee members Drs. Nagi Gebraeel, Yajun Mei, and Bianca Maria Colosimo, for contributing to my research by important comments, ideas, and suggestions they provided during my work, for providing me with data sets, and for evaluating my dissertation. I would also like to thank our collaborators Dr. Massimo Pacella from university of Salento, Dr. Shawn Mankad from Cornell university, and Dr. Richard Neu from Georgia Tech for their support and interesting applications that they provided.

My transition to industrial and systems engineering would not have been possible without the support that I received from my colleagues: Hao Yan, Xiaolei Fang, Samaneh Ebrahimi, Xiaowei Yue, Ana Maria Estrada Gomez, Andi Wang, Dan Li, Xinran Shi, Yuchen Wen, Sanam Gorgannejad, Mohammad Nebhan, Chen Zhang, Ruizhi Zhang, Juan Du, Zhen Zhong, and my officemate Sara Kaboudvand. Our constructive discussions significantly improved my work, and their friendship warmed my every day in Atlanta.

I am very grateful to my wife, Mojdeh, who has always been present for me. Her intelligence, curiosity, and kindness have inspired me and keep on inspiring me every day. Last but not least, I would like to thank my parents, brothers, and sister for their love and support during my time in graduate school.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Advancements in sensing technology and the shift toward the Internet of Things (IoT) has transformed and will continue to transform data analytics by producing new requirements and more complex forms of data. On one hand, the abundance of data creates a distinct opportunity to design more efficient systems and make near-optimal operational decisions (from data-rich to decision-smart). On the other hand, the structural complexity and heterogeneity of the generated data poses a significant challenge to extracting useful features and patterns for making sense of the data and facilitating decision-making. Therefore, continual research to develop new statistical and data analytical methodologies that overcome these data challenges and turn them into opportunities is required.

Integrating heterogeneous data in an effective manner to construct an efficient model of a system is the main theme of this Thesis. Heterogeneity of data may refer to different levels of accuracy of data, different levels of information that process inputs (specifically functional inputs) may contain in explaining an output, or different forms of data. In this thesis, we will built upon the existing works and methods related to each of these classes of heterogeneity, and introduce new methodologies to address existing challenges in practice.

In several applications, a large amount of low-accuracy (LA) data can be acquired at a small cost. However, in many situations, such LA data is not sufficient for generating a high-fidelity model of a system. To adjust and improve the model constructed by LA data, a small sample of high-accuracy (HA) data, which is expensive to obtain, is usually fused with the LA data. Unfortunately, current techniques assume that the HA data is already collected and concentrate on fusion strategies, without providing guidelines on how to sample the HA data. In Chapter I, we address the problem of collecting HA data adaptively and sequentially so when it is integrated with the LA data a more accurate surrogate model is achieved. For this purpose, we propose an approach that takes advantage of the information provided by LA data as well as the previously selected HA data points and computes an

improvement criterion over a design space to choose the next HA data point. The performance of the proposed method is evaluated, using both simulation and case studies. The results illustrate the importance of intelligent sampling of HA data in reducing the cost and improving the model accuracy.

Learning the relationship between a response variable (e.g., a quality characteristic) and a set of predictors (e.g., process variables) is of special importance in process modeling, prediction, and optimization. In many applications the number of these variables is large, but only a few of them contain information in explaining the output variable. Moreover, in many situations, these variables are high-dimensional (e.g., they are represented by waveform signals). This sparsity of the high dimensional variables (a few of a large number of HD variables contain information) requires a systematic approach to both modeling the relationship between the variables and removing the non-informative input variables. In Chapter II, we propose a functional regression method in which an HD response is estimated and predicted through a set of HD covariates. To deal with the HD variables, the functional regression coefficients are expanded through a set of low-dimensional smooth basis functions, making the estimation tractable while preserving the essential information of the covariates and response. In order to estimate the low-dimensional set of parameters and to deal with a large number of HD variables a penalized loss function with both smoothing and group lasso penalties is defined. The Block Coordinate Decent (BCD) method is employed to develop a scalable, iterative, and computationally tractable algorithm for minimizing the loss function and estimating the regression parameters. The group lasso penalty is modified so that the loss function has a closed-form solution in each block of BCD, which in turn increases the computational efficiency. Through simulations and case studies, the performance of the proposed method is evaluated and compared to benchmarks. The results illustrate the advantage of the proposed method over the benchmark based on the prediction mean square errors.

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. In Chapter III, we address 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 (MTOT) 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. In order 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.

CHAPTER 1
AN ADAPTIVE FUSED SAMPLING APPROACH OF HIGH-ACCURACY DATA IN THE PRESENCE OF LOW-ACCURACY DATA

1.1 Introduction

Accurate modeling of a complex system requires exploration of a large design space. However, this exploration often requires a large number of high-accuracy (HA) simulations or experiments that are too costly or time-consuming to conduct. Alternatively, the model can be built based upon less accurate simulations or experiments that are faster and less costly to undertake and can provide a large number of data points. These approximate simulations, however, compromise the model accuracy and may introduce bias in the model. A practical approach is to fuse the LA data obtained from crude but fast experiments with a few HA data points to remove the bias of the LA model and construct a reasonably accurate and cost-effective model [1, 2, 3, 4, 5]. Nevertheless, the proposed data fusion strategies assume the availability of LA and HA data and do not provide any strategy for sampling expensive HA data using the information that the abundant LA data can provide. The goal of this work is to propose an LA-data-led approach for collecting HA data to be fused with LA data for constructing a more accurate model. Therefore, this work lies at the intersection of design of experiments and data fusion literature. Such LA-data-led sampling approach of HA data can be used in many applications. For example, in geometric inspection and metrology [6, 4], measurements from less accurate metrology devices (e.g., structured light scanner) can lead the sampling trajectory of the HA data obtained from more accurate tools such as traditional Coordinate Measuring Machines (CMMs) that use contact touch probes.

In another example, reducing the number of tests performed on a combustion engine is one of the most important requirements to improve cost efficiency of the engine control

units (ECU) modeling. The ECU includes models of different systems in a vehicle that are constructed based on a large number of physical tests performed at different levels of engine torque and speed. Often, these physical tests have already been performed on other engines with similar specifications when collecting data from a new engine, and can be considered as historical LA data. This LA data can guide the design values at which performing a test on the new engine can significantly improve the model. This intelligent sampling can significantly reduce the number of required experiments on the engine, benefiting the manufacturer both economically and environmentally.

In general, data fusion refers to the process of combining data obtained from different sources with the goal of achieving improved results over what could have been obtained from each source, separately [5]. A data fusion approach may consider one of the following settings: a) Integrating data obtained from computer codes with different levels of accuracy [1], b) Integrating physical experiment data with the data obtained from a computer experiment [7], and c) Integrating data collected from physical experiments with different levels of accuracy [4]. In this chapter, we concentrate on a situation in which measurements contain noise and are collected from two sources, where one source is more accurate but more expensive than the other. We refer to the data from these two sources as low and high accuracy data. As a result, this chapter is more concerned with the third setting, i.e., the fusion of physical experiments data (or stochastic computer experiment data). Nevertheless, our proposed approach can be modified to be used in the deterministic settings. This work will not be concerned with the second setting, which is usually cast as a calibration of a computer model. In order to fuse data from LA and HA experiments, several approaches have been introduced in the literature. In the simplest approach, the two sets of data are directly merged together and are used to construct a surrogate model. For example, in metrology applications, [8] used the CMM machine at the boundaries, where the laser scanner is less accurate, and combined all the data to construct a model for a surface. Co-kriging models have extensively been used to fuse LA and HA experiment data [1, 9]. The co-kriging

approach usually generates very accurate predictive models but are computationally expensive [10]. Another popular framework used for data fusion is the hierarchical approach. [11] proposed a hierarchical method with linear models to integrate physical and computer experiments. [3] introduced Bayesian hierarchical Gaussian process (BHGP) models to fuse multiple sources of data. Their approach can be viewed as the Bayesian formulation of a co-kriging model. [4] used a hierarchical Bayesian model to align and fuse CMM and laser scanner measurements. An alternative method that is tightly related to the Bayesian approaches uses a link function between the LA and HA observations. In fact, both of the co-kriging and hierarchical Bayesian employ a form of link function. [2] proposed a two-step approach in which a surrogate model, referred to as a base model, is first fitted to the LA data. Next, the base model is corrected according to the HA data, using a link function that scales the base model linearly and captures the bias by a Gaussian process model. [4] used a kernel regression link model and employed a hierarchical Bayesian approach for estimating model parameters.

The aforementioned methods focus on data fusion, assuming that the LA and HA experiments have already been conducted and the data is available. They, however, do not provide a strategy on how to collect few expensive HA points given LA data to achieve a more accurate model. Consequently, they may under or over sample from the HA design space that may result in a less accurate model or unnecessary sampling costs. Therefore, it would be essential to devise an adaptive approach that uses LA data to systematically explore the design space and guide the sampling strategy for HA experiments. A large group of works in the field of computer design of experiments provides methods for sampling a set of points from a design space, either in one-step or sequentially, to construct an accurate model of a system. Space-filling designs such as Latin hypercube design [12, 13, 14, 15], distance-based designs [16], uniform designs [17], and sequential versions of these designs (e.g., Sobol sequences) are effective approaches for initial exploration of the surface/model when no information is available. They, however, are constructed based on the assump-

tion that the features of the true model are uniformly distributed across the design space. Criterion-based designs are obtained by minimizing/maximizing a statistical criterion such as mean square prediction error (MSPE) or entropy [18, 19, 20, 21]. These approaches can be converted into sequential designs by selecting the point that minimizes/maximizes the criterion at each step. The issue with these approaches is that they only capture the global behavior of the model using a correlation function but fail to concentrate on the areas of the design space where the model has high local variations. To capture both the global and local features, [22] introduced an expected improvement criterion for a global fit (EIGF). The EIGF at each point of a design space represents the amount of improvement (in the fitted model) that the corresponding point can introduce if added to the design. Unfortunately, none of the foregoing sequential design schemes takes the fusion of HA and LA data into account. They only focus on constructing a design when no prior information (e.g., LA data) is available.

A relevant approach that considers the fusion of LA and HA computer experiment data is the sequential nested Latin hypercube design (LHD) proposed by [23]. In this approach the HA computer experiments are conducted at locations sampled by small LHDs and the LA computer experiments are conducted at points selected by the large LHDs obtained by enlarging the small LHDs. The issue with this method is that when sampling the HA data, the information from the LA data is not used. [24] proposed sampling the LA and HA data using separate designs to calibrate a computer model based on a physical experiment. Although this approach provides flexibility in terms of choice of designs, the LA information remains unused when sampling the HA data. Please note that, although these two works are different in scope (i.e., one focuses on the fusion of data obtained from computer experiments and the other focuses on a calibration problem), both ignore the information from the LA data when sampling the HA data. This limitation is the main focus of this chapter.

The main goal of this chapter is then to introduce a sequential sampling method for

HA data that utilizes LA data to explore the design space and adaptively identify the points where HA data can have the highest impact on improving the accuracy of the surrogate model constructed by fusion of LA data and the sampled HA data. Additionally, as the proposed method samples the design space sequentially, it can choose just enough HA data samples for a pre-specified model accuracy and hence reduce the sampling costs. Our method combines the EIGF experiment design approach with a data fusion method that employs a link function to adaptively select the HA data. Specifically, we first fit a Gaussian process (GP) to LA data to obtain a base surrogate model of the system. This model captures the global behavior of the system but lacks accuracy. To improve this model, we then select an initial set of HA points using a space filling approach. Next, we modify the EIGF criterion based upon both HA and LA data, and employ it to select the next HA points. Because the information is limited early on in the sampling procedure, we propose a simulated annealing procedure to explore more when data is not reliable, and exploit the sampled data as the sampling moves forward and more information becomes available.

The rest of the chapter is organized as follows: In the next section, we review the fusion of the LA and HA data using a Gaussian process model. Section 1.3 describes the adaptive sampling approach and how it can be used for selecting HA data. Next, in Section 1.4, we evaluate the performance of the proposed method using two simulation studies. We also compare the proposed method to three benchmarks based on the model prediction error. Section 3.5 describes two case studies used to assess the proposed method in real-world applications. Finally, we summarize this chapter in Section 3.6.

1.2 Review of the one-step data fusion model using a Gaussian process and link function

In this section, we outline the data fusion method for the situation that both LA and HA data are available. Next, we modify this framework to adaptively select and fuse the HA data. When data from both experiments is already available, the common practice is to follow

a two-step approach for data integration that involves fitting a statistical model (usually a GP model) to the LA data to generate a base surrogate model. Next, this model is adjusted using the HA data to obtain the final surrogate model. Before we describe these two steps, we provide a set of notations used in this chapter.

In this chapter, we consider two sources of data: one obtained from an LA but inexpensive experiment, and one that is acquired (or to be acquired) from HA, expensive experiments. We assume that a design value in both the HA and LA experiments consists of same p factors denoted by $\boldsymbol{v} = (v_1, \cdots, v_p)$. We also denote an experiment data measured at a design value $\boldsymbol{v}$ by $z(\boldsymbol{v})$. For instance, in the metrology example, $\boldsymbol{v} \in \mathbb{R}^2$ is the location of a point in a $x - y$ plane and $z(\boldsymbol{v})$ is the height of the product at that point. In the engine example, $\boldsymbol{v} \in \mathbb{R}^2$ is a vector consisting of torque and rpm, and $z(\boldsymbol{v})$ is the engine performance (e.g., air mass) at a particular design value. We denote an LA point with an index L and an HA point with an index H, i.e., $(\boldsymbol{v}_L, z_L)$ for an LA data and $(\boldsymbol{v}_H, z_H)$ for an HA point. When no subscript is used for a design value $\boldsymbol{v}$, the point is general and can be low or high accuracy. We designate the design set of the LA experiments with M_L runs by $D_L = \{\boldsymbol{v}_{L1}, \boldsymbol{v}_{L2}, \cdots, \boldsymbol{v}_{LM_L}\}$ and the corresponding measurement data by $\boldsymbol{z}_L = [z_{L1}, z_{L2}, \cdots, z_{LM_L}]^{\mathsf{T}}$. Our goal is to find an HA design $D_H = \{\boldsymbol{v}_{H1}, \boldsymbol{v}_{H2}, \cdots, \boldsymbol{v}_{HM_H}\}$ with the corresponding HA measurements $\boldsymbol{z}_H = [z_{H1}, z_{H2}, \cdots, z_{HM_H}]^{\mathsf{T}}$; $M_H \ll M_L$ so that when integrated with $\boldsymbol{z}_L$ a more accurate surrogate model is obtained. That is, for an *unexplored* point $\boldsymbol{v}$, the constructed model returns a value that is almost equal to $z_H(\boldsymbol{v})$. Notice that we assume the HA experiment produces results that agree with the true system and therefore the error in the HA experiment data is negligible.

1.2.1 Fitting a Gaussian process to LA experiment data

The first step of data fusion is to construct a base surrogate model, using the LA data. Mathematically, this can be represented by

$$z_L = f_L(\boldsymbol{v}_L) + \epsilon_L, \tag{1.1}$$

where ϵ_L accounts for the error and is assumed to follow a normal distribution with mean zero and the variance σ_N^2, i.e., $\epsilon_L \sim \mathcal{N}(0, \sigma_N^2)$, and $f_L(\boldsymbol{v}_L)$ is the core part of the model that captures the features and patterns of the surrogate model. In the case of deterministic LA simulations, the term ϵ_L should be removed to reflect the fact that the data is not noisy. Among all possible models to represent $f_L(\boldsymbol{v}_L)$, a GP model is usually considered due to its flexibility and simplicity of parameter estimation. A GP model is a random process in which a joint distribution of any $k < \infty$ observations of the process $\{z_{L1}, \cdots, z_{Lk}\}$ follows a multivariate Gaussian distribution. Such a GP model is denoted by $z_L(\boldsymbol{v}) \sim GP(m_L(\boldsymbol{v}_L), k_L(\boldsymbol{v}_L, \boldsymbol{w}_L))$, where $m_L(\boldsymbol{v}_L)$ denotes the mean function of the Gaussian process evaluated at $\boldsymbol{v}_L$ and $k_L(\boldsymbol{v}_L, \boldsymbol{w}_L)$ is the covariance function of the GP. The output of $k_L(\boldsymbol{v}_L, \boldsymbol{w}_L)$ is the covariance between two random variables $z_L(\boldsymbol{v}_L)$ and $z_L(\boldsymbol{w}_L)$ when $\boldsymbol{v}_L \neq \boldsymbol{w}_L$ and the variance of $z_L(\boldsymbol{v}_L)$, otherwise. In many applications, a constant or a linear function is an appropriate selection for the m_L. In this article, we consider a linear function, i.e., we set $m_L(\boldsymbol{v}) = \boldsymbol{\beta}^{\mathsf{T}}[1; \boldsymbol{v}]$, where $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ is a set of parameters to be estimated. Several functional forms (kernels) have been introduced for covariance functions, such as exponential, squared exponential, and Matern covariance functions that are different in terms of smoothness and differentiability [25]. For a stationary GP, the covariance function depends only on the distance between the two points. The distance is usually defined as $r(\boldsymbol{v}, \boldsymbol{w}) = \sum_{i=1}^{p} \theta_i |v_i - w_i|^{d_i}$, where $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_p)$ and $\boldsymbol{d} = (d_1, \cdots, d_p)$ are the scale and power parameters. When $d_i = 2;\ i = 1, \cdots, p$ the distance is Euclidean. Often, a covariance function is written as $k_L(\boldsymbol{v}, \boldsymbol{w}) = \sigma_L^2 h(r(\boldsymbol{v}, \boldsymbol{w}))$, where

h is a correlation function, and σ_L^2 is the variance of data at any given point $\boldsymbol{v}$. For example, in the squared exponential kernel function $h(\boldsymbol{v}, \boldsymbol{w}) = \exp(-r(\boldsymbol{v}, \boldsymbol{w}; d_i = 2))$. For more information on the form and properties of the covariance functions see [25, 22]. The hyper-parameters $\Theta = \{\boldsymbol{\beta}, \sigma_L, \theta\}$ of the base model are unknown and should be estimated to best fit the observed LA data $\{(\boldsymbol{v}_L, z_L)\}_{i=1}^{M_L}$. The estimation procedure is based on the likelihood maximization of a multivariate normal distribution (see [25] and [2]). Once the hyper-parameters of a GP model are estimated given the available data, the model can be used for prediction of z at any unexplored point $\boldsymbol{v}$. The empirical best linear unbiased predictor (BLUP) is usually adopted for this purpose (see [2]) as follows:

$$\hat{z}_L(\boldsymbol{v}) = [1, \boldsymbol{v}]\,\hat{\boldsymbol{\beta}} + \boldsymbol{k}^{\mathrm{T}}(K + \sigma_N I_{M_L})^{-1}\left(\boldsymbol{z}_L - V\hat{\boldsymbol{\beta}}\right), \tag{1.2}$$

where $\hat{\boldsymbol{\beta}}$ is the vector of estimated parameters, σ_N is the noise effect due to the noise-contaminated measurements, $\boldsymbol{k}^{\mathrm{T}} = [k(\boldsymbol{v}, \boldsymbol{v}_{L1}), k(\boldsymbol{v}, \boldsymbol{v}_{L2}); \cdots, k(\boldsymbol{v}, \boldsymbol{v}_{LM_L})]$ is a vector whose elements are the covariance between $\boldsymbol{v}$ and all sampled points, $K \in \mathbb{R}^{M_L \times M_L}$ is a covariance matrix whose $(i,j)^{th}$ element is $k(\boldsymbol{v}_{Li}, \boldsymbol{v}_{Lj})$; $i, j = 1, \cdots, M_L$, and $V \in \mathbb{R}^{M_L \times (p+1)}$ is the regressor matrix whose i^{th} row is $[1, \boldsymbol{v}_{Li}]$. Note that when the data is deterministic σ_N is zero. The BLUP smoothly interpolates all the observed points to generate a prediction at a new point.

1.2.2 Fusion of the LA and HA data using a link model

In the second step, to integrate the LA and HA data and generate the final surrogate model, [2] proposed the following link function:

$$z_H(\boldsymbol{v}) = \rho(\boldsymbol{v})\,\hat{z}_L(\boldsymbol{v}) + \delta(\boldsymbol{v}) + e,$$

where $z_H(\boldsymbol{v})$ is the observed HA value at point $\boldsymbol{v}$, $\hat{z}_L(\boldsymbol{v})$ is the base model prediction, $\delta(\boldsymbol{v})$ is a GP model that captures the bias, $\rho(\boldsymbol{v})$ is a linear function in $\boldsymbol{v}$, i.e., $\rho(\boldsymbol{v}) =$

$\rho_0 + \sum_{i=1}^{p} \rho_i v_i$, and e is a random noise that follows a normal distribution. In the case of deterministic HA simulations, the term e should be removed to reflect the fact that the data is not noisy. Given a set of HA data, the goal is to estimate the parameters of $\delta(\boldsymbol{v})$, i.e., the mean and the covariance function hyper-parameters and $\boldsymbol{\rho} = \{\rho_0, \rho_1, \cdots, \rho_p\}$. The estimates can be computed by maximizing a likelihood function. For a detailed estimation procedure of these parameters refer to [2]. Next, given $\hat{\rho}_i;\ i = 1, \cdots, p$, one can compute $\boldsymbol{\delta} = [\delta(\boldsymbol{v}_{H1}), \cdots, \delta(\boldsymbol{v}_{HM_H})]$ by

$$\delta(\boldsymbol{v}_{Hi}) = z_H(\boldsymbol{v}_{Hi}) - \hat{\rho}(\boldsymbol{v}_{Hi})\, \hat{z}_L(\boldsymbol{v}_{Hi}).$$

Finally, at an unexplored point $\boldsymbol{v}$, the bias $\delta(\boldsymbol{v})$ can be predicted using a BLUP predictor. In this article, we select a constant value, δ_0, for the mean function. Therefore, the BLUP is constructed as

$$\hat{\delta}(\boldsymbol{v}) = \hat{\delta}_0 + \boldsymbol{k}^{\mathrm{T}} K^{-1} \left(\boldsymbol{\delta} - \hat{\delta}_0 \mathbf{1}_{M_H}\right),$$

where $\mathbf{1}_{M_H}$ is a vector of size M_H whose elements are all one. At a given point $\boldsymbol{v}$ the prediction of the final surrogate model is given by $\hat{z}(\boldsymbol{v}) = \hat{\rho}(\boldsymbol{v})\, \hat{z}_L(\boldsymbol{v}) + \hat{\delta}(\boldsymbol{v})$. Unless it is mentioned otherwise, a Matern covariance function with the scale parameter $\nu = \frac{3}{2}$ (see [25]) is used for $\delta(.)$

Note that the major assumption for this model is that the LA data provides sufficient information about the overall geometric shape of the true surface, otherwise discarding the LA data and using only HA data may provide a better surrogate model.

1.3 Adaptive data fusion using a modified EIGF criterion

Instead of assuming that the HA data is already available, this article aims to sample HA points sequentially to be integrated with the available LA data in order to construct a more accurate final surrogate model. For this purpose, we first describe an expected improvement criterion for a global fit, introduced by [22]. Then, we modify this criterion so that it takes

the LA data into account when selecting an HA point and ultimately constructing an HA design.

1.3.1 Expected improvement for a global fit

An expected improvement (EI) for computing the maximum of a black-box function was first proposed by [26]. [22] extends the EI criterion to a global fit setting, where the goal is to find a design that leads to a better global fit rather than the maximum point. Let $Z(.)$ follow a GP model denoted by $GP(m_Z(.), k_Z(.,.))$, i.e., $Z(\boldsymbol{v})$ is a random variable that follows a normal distribution with mean $m_Z(\boldsymbol{v})$ and the variance $Var(Z(v)) = k_Z(\boldsymbol{v}, \boldsymbol{v})$. Then, the improvement of a point $\boldsymbol{v}$ is defined as

$$I(\boldsymbol{v}) = (Z(\boldsymbol{v}) - z(\boldsymbol{v}_c))^2,$$

where $\boldsymbol{v}_c$ is the nearest observed design value to $\boldsymbol{v}$, and $z(\boldsymbol{v}_c)$ is its corresponding data. Note that one may consider $I(\boldsymbol{v}) = \frac{(Z(\boldsymbol{v}) - Z(\boldsymbol{v}_c))^2}{||\boldsymbol{v} - \boldsymbol{v}_c||^2}$ as an improvement criterion, which effectively gives more weight to the points that are closer to $\boldsymbol{v}_c$. But, due to the major assumption of the Gaussian process that the closer the points the more similar they are, we would like to select a point further away from $\boldsymbol{v}_c$ to obtain more information. Therefore, a normalized $I(\boldsymbol{v})$ may not be a good choice. Because $Z(\boldsymbol{v})$ is a random variable, the expected improvement for a global fit (EIGF) should be considered and is defined as

$$EIGF(\boldsymbol{v}) = E\left((Z(\boldsymbol{v}) - z(\boldsymbol{v}_c))^2\right) = Var(Z(\boldsymbol{v})) + [\hat{z}(\boldsymbol{v}) - z(\boldsymbol{v}_c)]^2,$$

where $\hat{z}(\boldsymbol{v})$ is the prediction value of the GP model at point $\boldsymbol{v}$. In the EIGF equation, a large value of $Var(Z(\boldsymbol{v}))$ indicates that the model is uncertain about its prediction at $\boldsymbol{v}$, and therefore including $\boldsymbol{v}$ in the design can significantly improve the model. Similarly, a large value of $[\hat{z}(\boldsymbol{v}) - z(\boldsymbol{v}_c)]^2$ indicates a high local variation around $\boldsymbol{v}$ and therefore selection of $\boldsymbol{v}$ in the design may help the model in capturing the local behaviors. As a

result, in theory, the EIGF captures both the global and local behavior of the model based on the variance at each point, and by exploiting the observed neighboring points. To obtain a design D_H, one can use the EIGF criterion sequentially. That is, at each step the maximizer of the EIGF over the design space is sampled until a pre-specified design size is reached.

The main issue with the EIGF criterion is that, with no prior knowledge of the surface and with a small design size, $[\hat{z}(\boldsymbol{v}) - z(\boldsymbol{v}_c)]^2$ may not capture the local variations because the prediction value at a point v is an interpolation of its neighboring points (obtained from the BLUP), and therefore it is likely that $[\hat{z}(\boldsymbol{v}) - z(\boldsymbol{v}_c)]^2$ becomes ineffective. Therefore, given the small size of HA experiments, an adaptive sampling approach based on EIGF that only relies on HA data may result in an inaccurate surrogate model. However, when LA data is available, the information from the base surrogate model can be exploited as the prior knowledge to resolve this problem. Beside this issue, the selection of the EIGF maximizer at each step of the sequential sampling, especially at the early steps when only a few data points are explored, may result in excessively greedy choices. To alleviate this problem, an exploration-exploitation routine can be considered.

In the next section, we introduce a modified version of EIGF that computes the expected improvement of a point by considering not only the explored HA neighboring points, but also the available LA data. Furthermore, we design an exploration and exploitation framework that uses the modified EIGF probabilistically to sample the next HA point.

1.3.2 Modified EIGF and data fusion

In this section, we propose an approach for sampling HA points, considering the information of the LA data. For this purpose, we modify the EIGF criterion so that it takes the LA data into account when computing the expected improvement of a design value. The underlying assumption of the proposed method is that the base surrogate model, constructed using the LA data, captures some of the local features of the system, and therefore can lead the sampling path to the areas of the design space with higher local variations. This will

help increase the number of samples collected in these local areas and, consequently, results in a more accurate model. In order to achieve this goal, we define the fused expected improvement for a global fit and data fusion (FEIGF) by considering the link function between the HA and LA, $Z_H(\boldsymbol{v}_H) = \rho(\boldsymbol{v}_H)\,\hat{z}_L(\boldsymbol{v}_H) + \delta(\boldsymbol{v}_H)$, as follows:

$$\begin{aligned} FEIGF(\boldsymbol{v}_H) &= E\left(\left(Z_H(\boldsymbol{v}_H) - z_H(\boldsymbol{v}_{Hc})\right)^2\right) \\ &= E\left(\left(\rho(\boldsymbol{v}_H)\,\hat{z}_L(\boldsymbol{v}_H) + \delta(\boldsymbol{v}_H) - \rho(\boldsymbol{v}_{Hc})\,\hat{z}_L(\boldsymbol{v}_{Hc}) - \delta(\boldsymbol{v}_{Hc})\right)^2\right) \\ &= \left(\rho(\boldsymbol{v}_H)\,\hat{z}_L(\boldsymbol{v}_H) - \rho(\boldsymbol{v}_H)\,\hat{z}_L(\boldsymbol{v}_{Hc})\right)^2 + Var\left(\delta(\boldsymbol{v}_H)\right) \\ &+ \left(\hat{\delta}(\boldsymbol{v}_H) - \delta(\boldsymbol{v}_{Hc})\right)^2, \end{aligned}$$

where $\boldsymbol{v}_H$ is a design value that may be considered for an HA experiment and $\boldsymbol{v}_{Hc}$ is the nearest explored point to $\boldsymbol{v}_H$. In the FEIGF equation, $\rho(.)$ and $\delta(.)$ are assumed to be known or are estimated based on previously observed data (e.g., data from previous sampling steps). Furthermore, $\hat{\delta}(\boldsymbol{v}_H)$ denotes the prediction of the bias GP model, $\delta(\boldsymbol{v}_{Hc})$ represents the bias value computed based on the HA measurement and is assumed to be known, and $\hat{z}_L(.)$ denotes the prediction of the base surrogate model at a given point. The FEIGF consists of three terms: The first term $\left(\rho(\boldsymbol{v}_H)\,\hat{z}_L(\boldsymbol{v}_H) - \rho(\boldsymbol{v}_H)\,\hat{z}_L(\boldsymbol{v}_{Hc})\right)^2$, measures the local behavior of the surface as it is revealed by the LA model. The large value of this term at a design value $\boldsymbol{v}_H$ signals high variations around that point and therefore, including it in the design can improve the accuracy of the final model. The second term, $Var\left(\delta(\boldsymbol{v}_H)\right)$, captures the prediction uncertainty of the bias model, δ, at $\boldsymbol{v}_H$. When $Var\left(\delta(\boldsymbol{v}_H)\right)$ is high, including $\boldsymbol{v}_H$ in the design may improve the predictions of the bias model. The third term, $\left(\hat{\delta}(\boldsymbol{v}_H) - \delta(\boldsymbol{v}_{Hc})\right)^2$, quantifies the local variations of the bias around $\boldsymbol{v}_H$, and therefore, including this point in the design may result in more accurate bias predictions.

In order to adaptively and sequentially sample an HA data point using the FEIGF criterion, we first consider a dense grid, $\mathcal{G}$, over the design space and at each step select the design value that for example maximizes the FEIGF criterion over such a grid. In

other words, at the sampling step t (i.e., when $t-1$ HA points are already collected), we first estimate the GP model of bias (δ_{t-1}) and the scaling parameter (ρ_{t-1}) based on the available LA points and the HA data points sampled up to step $t-1$, using the approach explained in section 1.2.2. Next, given the GP model of bias and the scaling parameter, we calculate the $FEIGF$ over the grid $\mathcal{G}$. Finally to select a point to be added to the design, we may consider the point that maximizes $FEIGF$ criterion. That is, we may sample $\boldsymbol{v}_t = \arg\max_{\boldsymbol{v}\in\mathcal{G}} \left(FEIGF\left(\boldsymbol{v}\right)|\delta_{t-1}, \rho_{t-1}\right)$. However, the problem with sampling the point with maximum $FEIGF$ is that at the early stages of HA sampling when the number of HA points is small, relying on the maximum value of FEIGF may result in excessively greedy choices. The reason is that when the GP estimation is based on only few points the prediction variance (i.e., the second term in FEIGF) is large for many points, increasing the influence of the third term in separating and selecting the HA point. The third term, however, is inaccurate when the bias δ is estimated based on only few points, and results in a poor choice of a design value. To alleviate this problem, we adopt a probabilistic sampling approach in collecting the points so that early on in the process it allows selecting points with smaller-than-maximum FEIGF value (exploration) and as more information becomes available, it high-likely samples design values that maximize the FEIGF criterion. For this purpose, we define the following probability distribution parameterized by α_t over the $FEIGF\left(\boldsymbol{v}\right)$:

$$p_{\alpha_t}\left(FEIGF\left(\boldsymbol{v}\right)\right) = \frac{\exp\left(\alpha_t FEIGF\left(\boldsymbol{v}\right)\right)}{\int \exp\left(\alpha_t FEIGF\left(\boldsymbol{v}\right)\right) dv},$$

where α_t is an increasing sequence that tends to zero as $t \to 0$ and tends to ∞ as $t \to \infty$. In an extreme case, when $\alpha_t = 0$, the distribution $p_{\alpha_t}\left(FEIGF\left(\boldsymbol{v}\right)\right)$ is a uniform distribution and as $\alpha_t \to \infty$, the above distribution places all the probability mass over the $\max\left(FEIGF\left(\boldsymbol{v}\right)\right)$. In this study, we select $\alpha_t = \frac{t}{M_H}$ at the sampling step t. We randomly sample from $p_{\alpha_t}\left(FEIGF\left(\boldsymbol{v}\right)\right)$ at each step t to select the HA data point. Observe that early in the sampling process, we allow selecting points that have smaller values of improvement to explore the design space, and as more information becomes available, we select points

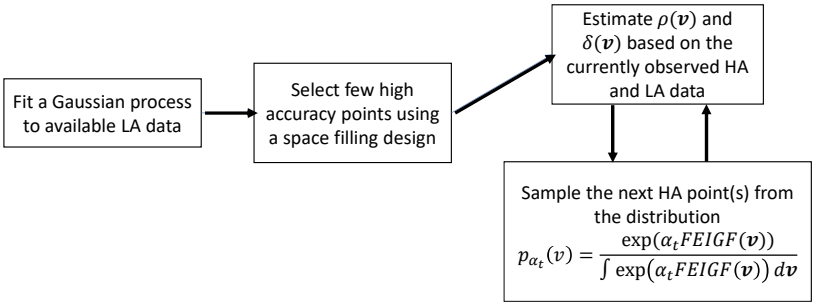


Figure 1.1: Overview of the proposed adaptive data fusion approach

with larger FEIGF. Note that, to compute the probability distribution, we approximate the continuous distribution by a discrete distribution in which the probability mass is located at each point of the grid $\mathcal{G}$, where we calculated the FEIGF. Finally note that, one can sample a batch of points, instead of one, at each sampling process step. However, in our experience, when the batch size is small comparing to the total sampling budget, it will not significantly influence the results. Figure 1.1 illustrates an overview of the proposed adaptive data fusion approach.

1.4 Performance evaluation using simulation study

In this section, we conduct simulations to evaluate the performance of the proposed adaptive data fusion technique. We compare the proposed method, labeled as *adaptive with fusion*, with three other benchmarks: In the first benchmark, we consider a one-step design in which M_H data points from HA experiments are sampled using an LHD and are directly used to construct the final model. We label this benchmark as *one-step* approach. The second benchmark, designated as *one-step-fused*, selects M_H data points from HA experiments using an LHD and fuses them with the available LA data through a link model described in Section 1.2.2. Finally, the third benchmark adaptively samples the HA data using the EIGF without integrating the sampled data with the available LA data. We label this method *adaptive w/o fusion*. Note that we used the standard LHD which is not based on optimizing any criterion. We compare all the methods in terms of prediction accuracy

using standard mean square error, i.e.,

$$SMSE = \frac{\sum_{i=1}^{M} \left(z_i - \hat{z}_i\right)^2}{\sum_{i=1}^{M} z_i^2},$$

where z_i is the true system value, $\hat{z}_i$ is the predicted value of the final surrogate model, and M is the total number of points in a grid over which we sample points. In order to estimate the hyper-parameters of a GP model, we use a MATLAB package developed by [25].

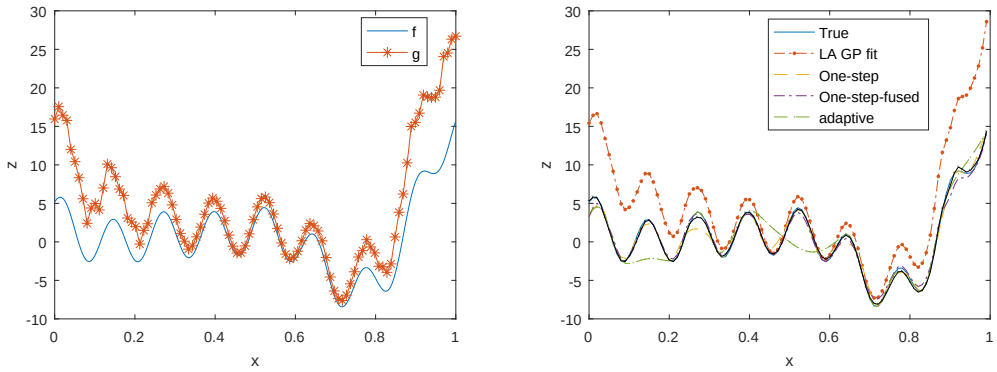
One-dimensional surface simulation: We first consider a one-dimensional non-stationary function

$$z_H = f(v) = (6v-2)^2 \sin(12v-4) + 3\cos(50v-1) + \cos(4v-1)$$

as the true function for $v \in [0,1]$. Next, we generate the LA data from a shifted and scaled version of the function f defined as

$$z_L = g(v) = Af(v) + B(v-0.5)^2 + C + \epsilon(v),$$

where, $A = 1.2$, $B = 40$, and $C = 0.5$, and $\epsilon(v)$ is a normally distributed error with mean 0 and standard deviation $\tau(v) = 4(v-0.5)^2$, i.e., $\epsilon(v) \sim \mathcal{N}\left(0, \tau(v)^2\right)$. Notice that z_L is designed to have larger bias and noise near the boundaries as it is the case in many real applications (e.g., in metrology). Figure 1.2a illustrates the true curve along with the LA data collected over an equidistant grid of size 100. Figure 1.2b illustrates the true curve, the fitted GP to the LA data and the fitted curves obtained by one-step and one-step-fused benchmarks as well as the adaptive with and without fusion methods. The initial number of points for the adaptive approaches is $M_{H,init} = 5$ and the budget, M_H, is 25 for all methods. As described, the one-step benchmarks sample the 25 points at once using an LHD, whereas the adaptive methods first sample $M_{H,init} = 5$ design values by an LHD, and then sequentially sample points from a grid of size $M = 1000$. As illustrated,



(a) (b)

Figure 1.2: (a) Illustration of the true curve and LA data in one dimensional simulation; (b) Example of the fitted curves in comparison to the true curve

the predictions obtained from the proposed method and the one-step-fused benchmark are fairly accurate, indicating the advantage of the data fusion when the LA data captures some of the local and global behaviors of the surface. Between these two methods, the proposed method performs slightly better where local variations exist and the LA data is more biased (e.g., in $[0, 0.1]$ and $[0.9, 1]$ intervals). The adaptive w/o fusion selects several points on the interval $[0.8, 1]$ where the function has a large slope and generates an accurate prediction. However, it fails to capture the local variations in the intervals $[0.1,\ 0.2]$ and $[0.4,\ 0.6]$ because $\hat{z}\left(\boldsymbol{v}\right)$ is an interpolation of its neighbors, which can be located far apart when the sample size is small. As a result, the adaptive without fusion does not perform well in this scenario. The one-step approach performs fairly well but fails to capture some of the local variations due to a lack of prior information. Such information is provided by the LA data when fusion is performed.

To further investigate and compare the performance of the proposed method with the benchmarks, we calculate the SMSE values after sampling of each point in adaptive approaches and compare them to the SMSEs obtained by one-step methods. Because our approach is probabilistic in nature, we consider 100 replications of the simulation and take the average of SMSE values over these replications. For all these simulations, we take the

initial number of points to be 20% of the budget, M_H. Figure 1.3 illustrates the performance of proposed method in comparison to all benchmarks at different values of M_H. For a better visualization, we illustrate the logarithm of SMSE in these figures. For the one-step methods the SMSE represents the error of the fit when M_H points are all selected at once. Observe that in all cases the adaptive approach outperforms the other methods in terms of SMSE. First, with the same number of samples the proposed method produces more accurate predictions. For instance, when $M_H = 20$, the logarithm of the SMSE of the proposed method is about -5, where it is about -4.2 and -2 for one-step-fused and one-step benchmarks, respectively. Second, it reaches the same level of the prediction error with fewer number of samples than the benchmarks. For example, in Figure 1.3c and 1.3d the adaptive approach with fusion obtains the same level of SMSE with about 15 and 20 samples rather than 25 and 30 used by other methods.

In all cases, the SMSE of the adaptive approach drops rapidly at the beginning of the sampling process. The reason is that early samples provide information about the overall behavior of the bias (i.e., they introduce large improvements), but late samples mostly introduce detailed information and provide small improvements. Therefore, after some sampling steps the improvement in SMSE becomes very slow. In practice, this step can be considered as a stopping criterion. For example, in Figure 1.3d after sampling 26 points, no significant improvement can be observed**.**

Two-dimensional surface simulation: As a two-dimensional example, we consider a six-hump camel surface proposed by [27] as the true surface:

$$z_H = f(v_1, v_2) = 4v_1^2 - 2.1v_1^4 + \frac{1}{3}v_1^6 + v_1v_2 - 4v_2^2 + 4v_2^4$$

with $v_1 \in [-2, 2]$ and $v_2 \in [-1, 1]$. We scale and shift the true function to construct an LA

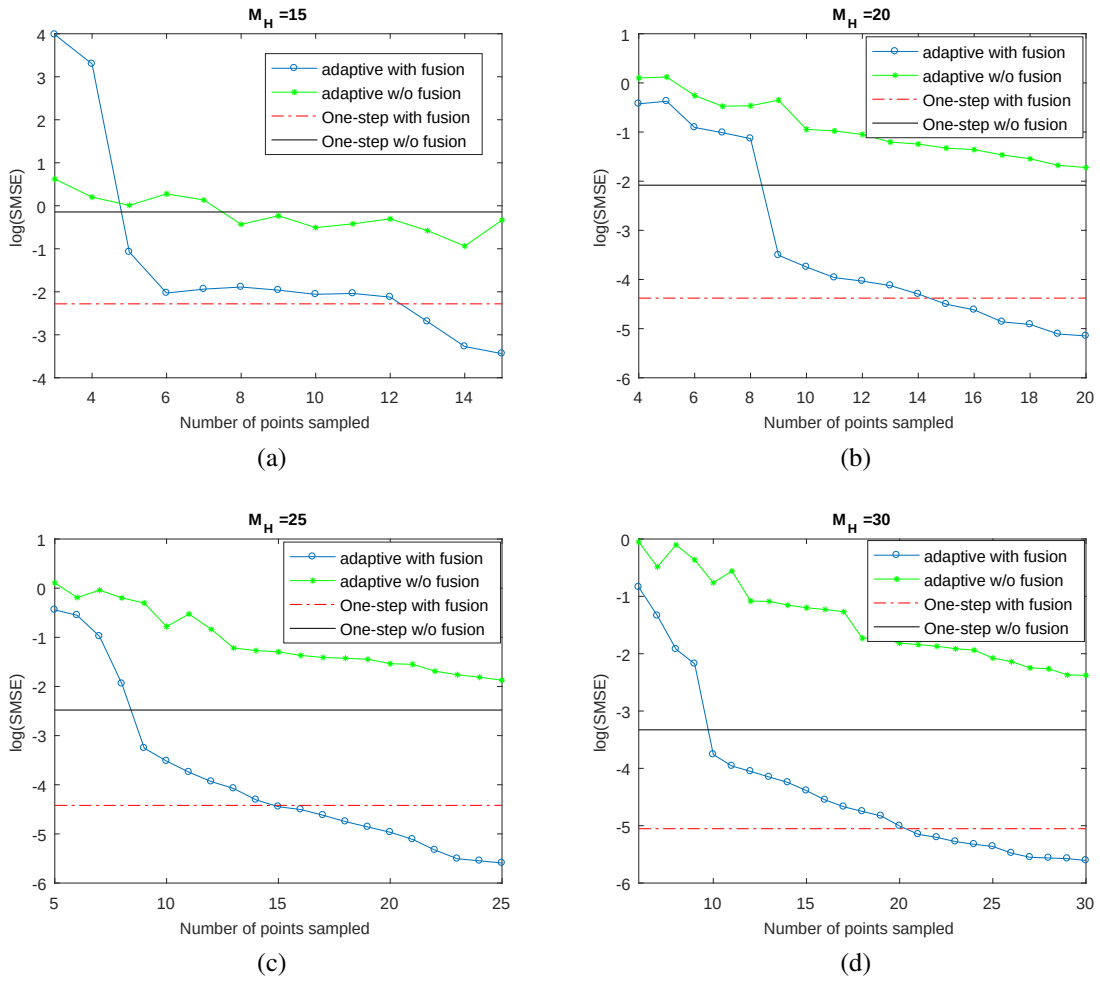


Figure 1.3: Comparison of the proposed method to benchmarks in simulation study I at different values of budget M_H. Observe that the adaptive approach reaches the same level of error as benchmarks with much fewer samples. For example in (c) the proposed approach reaches the same level of error of one-step-fused with only 15 sampled points.

data generator as

$$z_L = g(\boldsymbol{v}) = Af(\boldsymbol{v}) + B_1(v_1 - 0.5) + B_2(v_2 + 0.5) + C + \epsilon,$$

where $A = 0.5$, $B_1 = 1$, $B_2 = 2$, and $C = 2$, and ϵ is a normally distributed error with mean 0 and standard deviation τ, i.e., $\epsilon \sim \mathcal{N}(0, \tau^2)$. Figure 1.4 illustrates the f and g functions evaluated over an equidistant grid of 50×50 with $\tau = 0.2$. We evaluate the performance of the proposed method in comparison to all benchmarks at different values of M_H based upon the SMSE values calculated over an equidistant grid of 50×50 over the design space. Similar to the previous simulation study, the SMSE of the one-step methods represents the prediction error of the final model generated by M_H points that are selected at once using an LHD. Figure 1.5 illustrates the SMSE of the proposed method versus the benchmarks. Because our approach is probabilistic in nature, we consider 100 replications of the simulation and take the average of SMSE values over these replications. As illustrated, in all cases the adaptive approach results in more accurate predictions. For instance, when $M_H = 80$, the logarithm of the SMSE is about -5.5 that outperforms the one-step-fused benchmark (-5), and the other two benchmarks (-3.5). Furthermore, the proposed method can achieve the same level of prediction error with fewer number of samples than the benchmarks. For example, in Figure 1.5d the adaptive approach with fusion obtained the same level of SMSE with about 65 samples rather than 100 that is used by other methods. In addition, the initial SMSE of the proposed method is always smaller than the initial SMSE of the adaptive w/o fusion benchmark. This superiority is due to the information provided by the LA data. Finally, it is worth mentioning the two small jumps in Figures 1.5a and 1.5b. The reason for these jumps is the computational instabilities that may occasionally occur when fitting a GP model. Such instabilities are related to the likelihood optimization procedure that may be trapped in a local maxima. To avoid (or alleviate) these situations, we incorporate a simulated annealing heuristic for function optimization [28] in

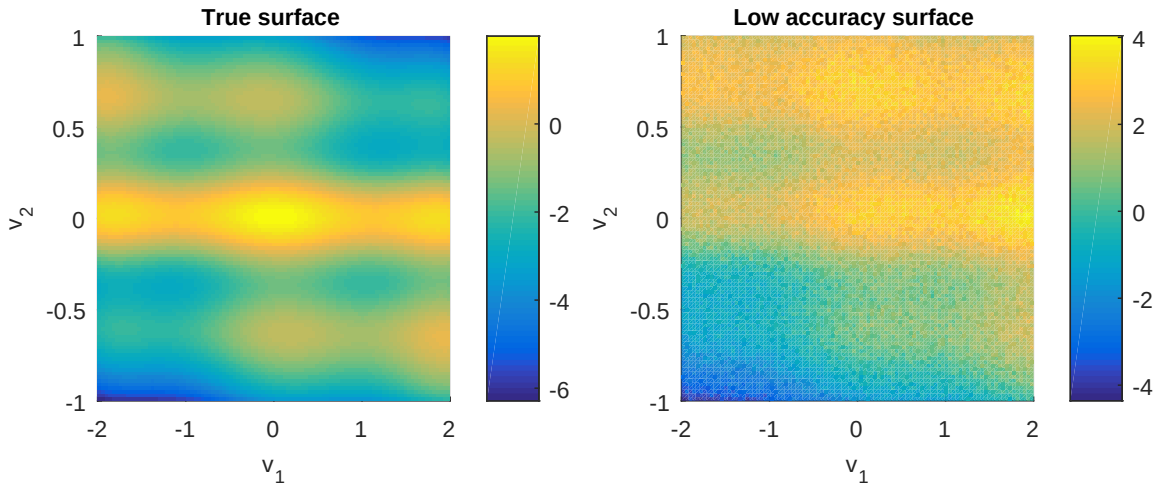


Figure 1.4: Illustration of the true and biased surfaces used for two dimensional simulation

our procedure.

1.5 Case Study

In this section, we consider two case studies to evaluate the performance of the proposed method in real-world applications. First, we consider a metrology example in which a set of HA metrology data are collected using a CMM to improve the LA measurements collected by a structured laser (SL) scanner for reconstructing a free-form surface. In the second case study, we are interested in constructing a surrogate model that predicts the air mass of an engine at a specific torque and revolution speed by performing few new HA experiments on the engine and combining it with previously collected data from another similar engine.

1.5.1 Freeform surface metrology

The freeform surface, illustrated in Figure 1.6, has been used for this case study. The freeform surface is machined from a $100\,mm \times 100\,mm \times 100\,mm$ workpiece. The surface is measured, using both SL scanner and a Coordinate Measuring Machine (CMM) Zeiss "Prismo 5 HTG VAST" equipped with an analog probe head with maximum probing error

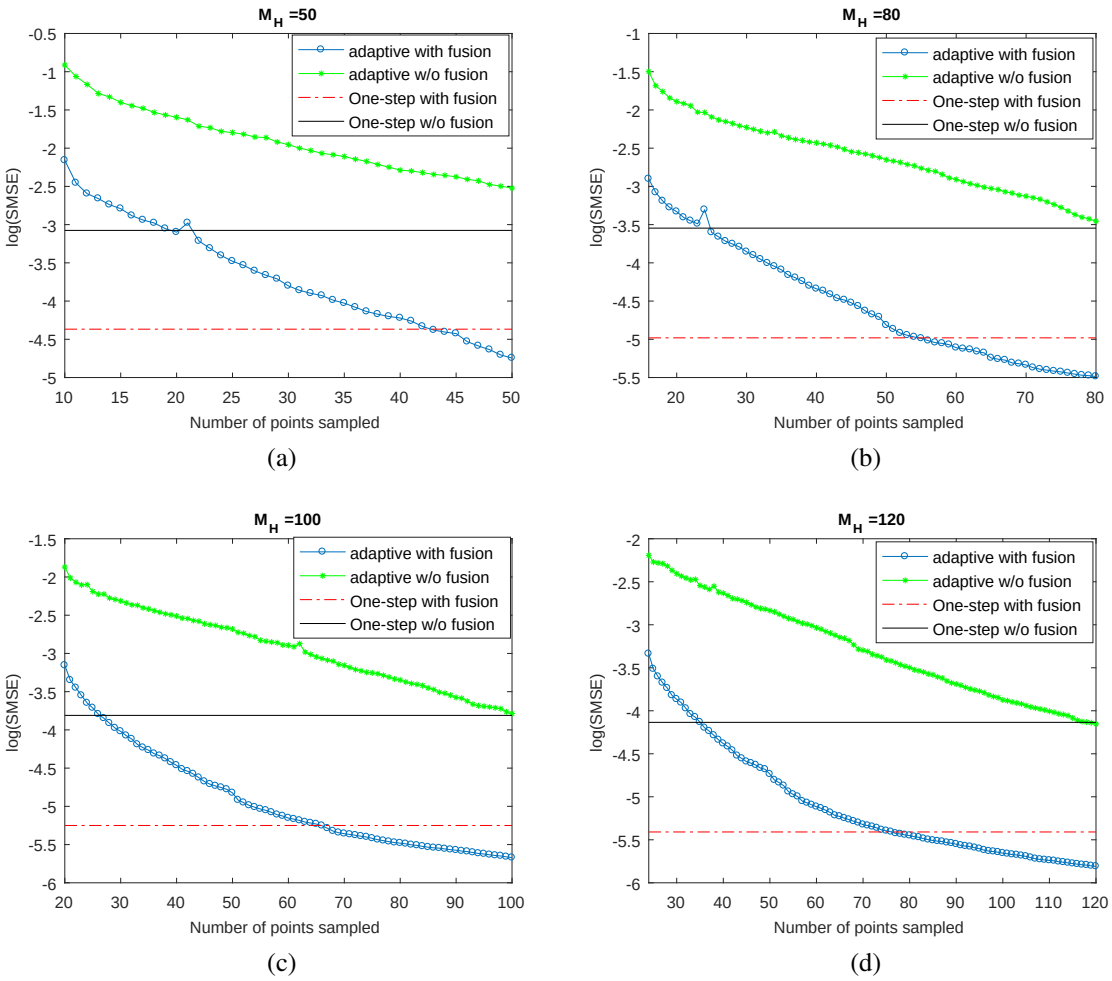


Figure 1.5: Comparison of the proposed method to benchmarks in simulation study II at different values of budget M_H. Note that the adaptive approach can obtain the same level of error as benchmarks with fewer samples. For example in (c), the proposed approach achieves the same level of error as one-step with fusion benchmark with only 65 sampled points rather than 100.

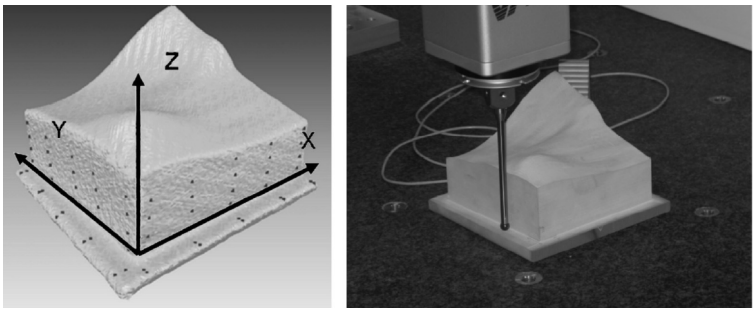


Figure 1.6: The free-form surface used for the case study [5]. Three orthogonal reference surfaces (left panel). CMM sampling (right panel).

of $MPEP = 2\,\mu m$ (according to ISO 10360-2). For more details on the measurement procedure refer to [5]. The point cloud obtained from each machine (SL scanner and CMM) contains 9635 points. The measurements from the SL scanner is considered as the LA data points, and the available data from the CMM is used to construct an emulator of the CMM by fitting a GP model. The goal is to collect a set of HA data from the CMM so when combined with the LA data points a more accurate final surrogate model is achieved.

We use our proposed method to construct a final surrogate model of the surface, and will compare its performance in reconstructing the freeform surface to other benchmarks, i.e., one-step, one-step-fused, and adaptive w/o fusion at different levels of budget M_H. In the case of adaptive approaches, we choose the initial sample size to be 20% of the budget. In order to estimate the link model parameters, we consider the squared exponential covariance function. Before discussing the prediction performance of surrogate models obtained by the proposed method and the benchmarks, it is interesting to look into the location of the points selected by the proposed adaptive approach. Figure 1.7 illustrates the location of the selected points for $M_H = 30$ and 75. As illustrated, more points are selected on the upper right corner and at the lower left corner where more local variations exist. For example, several points are sampled around the boundary of the hump on the lower left area. This indicates the capability of our approach in selecting points at the locations with higher variability. Figure 1.8 illustrates the logarithm of SMSE of each method at different levels of budget. As can be seen from the figures and similar to the simulation results, the

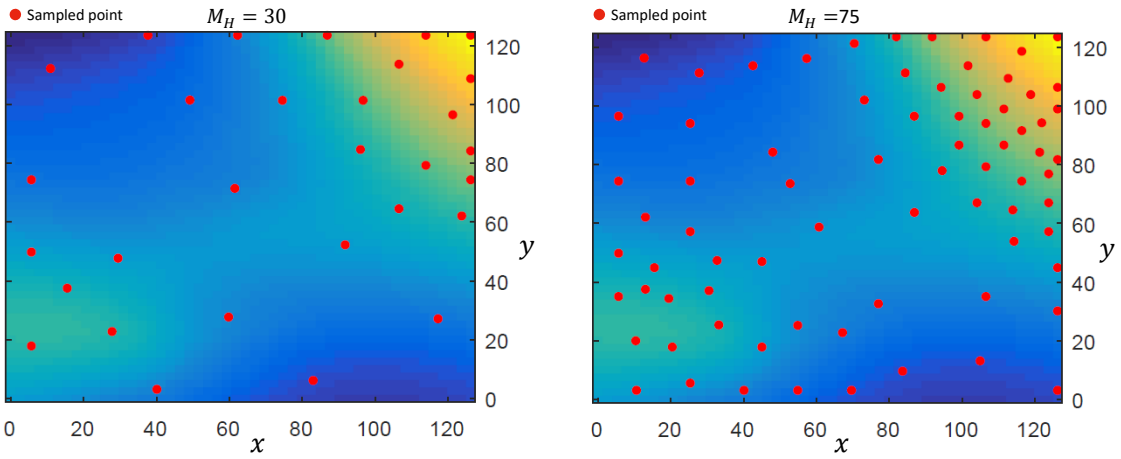


Figure 1.7: Location of the points selected by the proposed method for (a) $M_H = 30$ and (b) $M_H = 75$

proposed method can achieve the same level of SMSE as other approaches, particularly the one-step-fused approach, with much smaller sampled points. For example, when $M_H = 70$, the proposed approach obtains same SMSE as the one-step-fused approach with about 60% of the budget. Observe that the initial SMSE of the proposed method is smaller than both the one-step and adaptive w/o fusion benchmarks. The reason is the good accuracy of the LA data that results in the base surrogate model with small errors. The base surrogate model provides significant information about the surface even without any (or with very few) HA data points available.

1.5.2 ECU calibration

In a modern vehicle, the engine control unit (ECU) ensures the functionality of the vehicle and diagnoses failure for a number of components. The ECU implements surrogate models of complex physical dynamical systems that are constructed based on a large number of tests performed at different levels of engine torque and speed. To reduce the experiment cost, exploiting the performance measurements of another engine (LA data) with similar specifications (e.g., with the same number of cylinder and displacement) may help iden-

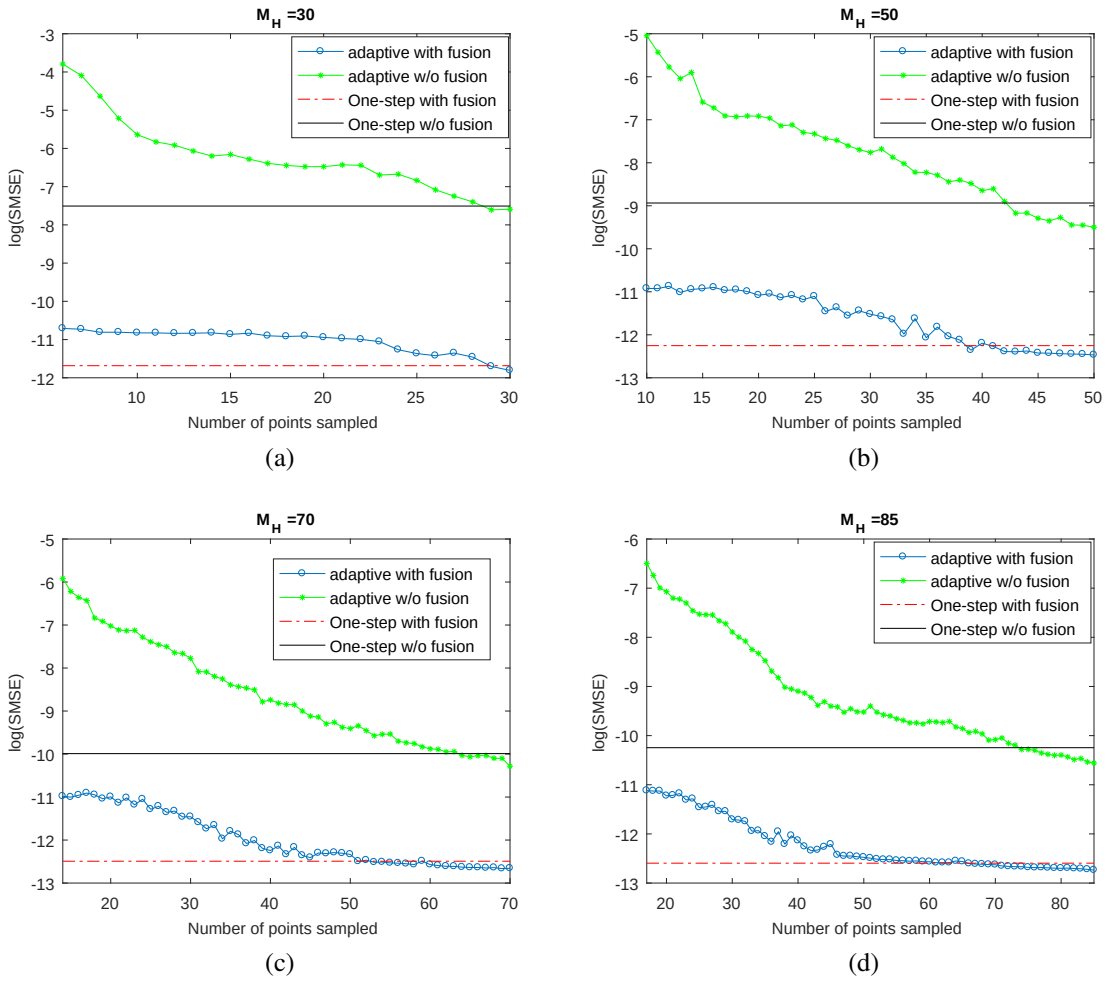


Figure 1.8: Comparison of the proposed method to benchmarks in case study I at different values of budget M_H. Note that the adaptive approach requires fewer samples to obtain the same level of error as benchmarks. For example, in (b) the proposed approach obtains the same SMSE as the one-step-fused approach with about 75% of the budget.

tify few design values at which the engine should be evaluated. The data collected from these tests (HA data) can then be fused to the LA data to construct a surrogate model. For example, for a new engine, the air mass of the combustion process should be measured at different combinations of engine torque and revolution speed (rpm) to ensure the right performance of the engine. Performing these tests over a large number of rpm-torque combinations is not economical. Historical data obtained from a similar engine may illuminate few points where the air mass should be measured.

For this purpose, we consider two diesel engines, both with the displacement of $2000\,cc$ and four cylinders. In ECU calibration of a diesel engine a typical task is to optimize the air mass of the combustion process due to the different engine states, namely different combinations of engine revolutions (rpm) and torque. We designate the engines by E_1 and E_2. We assume that the data from E_1 has already been collected through a series of experiments and are interested in integrating this data with a few experimental data obtained from E_2 to generate a surrogate model that predicts the performance of E_2 at various design values. Therefore, the data in this case study is the air mass measured over a two dimensional space of rpm and torque.

The available data for both engines E_1 and E_2 contains 256 measurements at different combinations of rpm-torque. In this case study, we first generate an emulator of each engine $E_i;\ i = 1,\ 2$ by fitting a GP model with a constant mean and piecewise cubic covariance (PCC) function to the available data. These emulators are then used to replicate the performance of the engines at any given design value. Figure 1.9 illustrates the predictions of both E_1 and E_2 emulators over a grid of size 50×50. Note that the emulator of E_1 can be considered as the base surrogate model, i.e., a GP model fitted to the LA data, but the emulator of E_2 is used as an HA simulator that produces the result of an experiment on E_2 at any pair of rpm and torque.

We compare the accuracy of the final surrogate models generated by the proposed method and the benchmarks for different budgets. For this purpose we calculate the SMSE

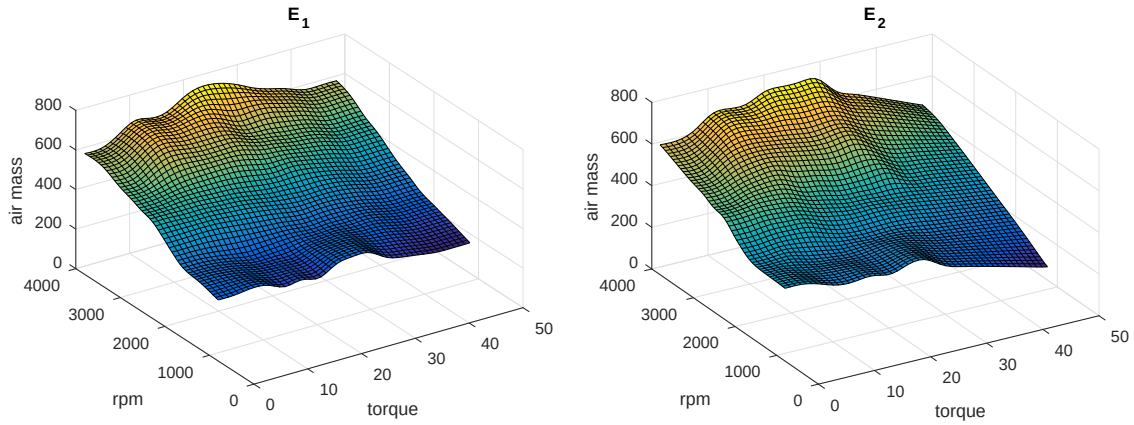


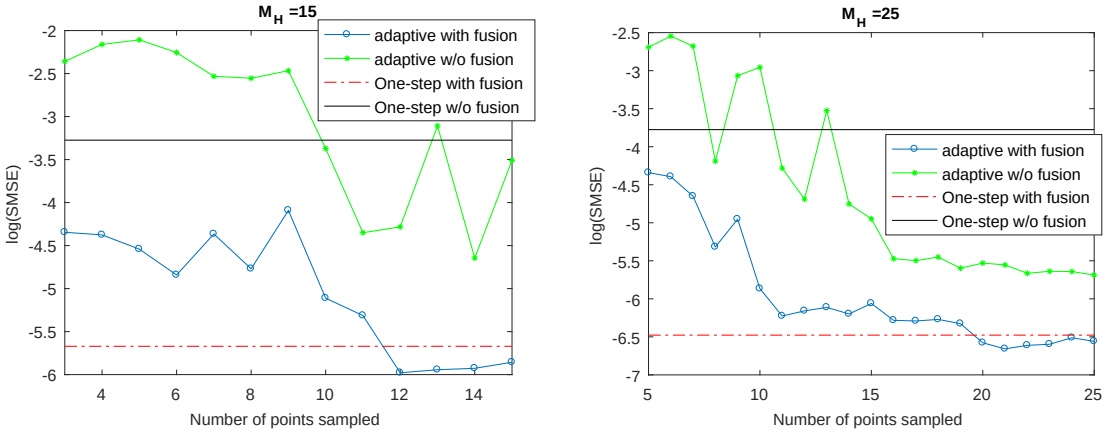
Figure 1.9: Prediction of the engines air mass at different values of rpm-torque pairs obtained by the emulators.

over an equidistant grid of size 100×100. Figure 1.10 illustrates the logarithm of SMSEs of each method at $M_h = 15, 25$. As illustrated, the final surrogate model constructed by the proposed method achieves lower prediction errors. Furthermore, the proposed approach can obtain the same level of accuracy with much fewer sampled points when compared to the benchmarks. For example when $M_H = 25$, the proposed approach obtains the same level of accuracy as the one-step-fused approach with about 80% of the budget.

Examples of the distribution of the points included in the design for different budgets are illustrated in Figure 1.11. Note that in all three cases, the points are mostly selected at the locations with higher local variabilities, especially the upper part of the surface. This is mainly because the proposed method can identify the locations with higher local variations and perform exploration at those areas.

1.6 Discussion

This work mainly assumes that the LA data is already available and utilizes the information from such data to sample HA data. The available LA data could have been collected using any design of experiment approach such as LHD and criterion-based designs. The design of experiment method by which the LA data is collected may or may not have significant influence on the final accuracy of the model. For example, in the situation, in which obtain-



(a) (b)

Figure 1.10: Comparison of the proposed method to benchmarks in case study II for different values of budget M_H. Observe that in both situations, the adaptive approach can reach the same level of error as benchmarks with much fewer samples. For example, in (b) the proposed approach obtains the same SMSE as the one-step-fused approach with about 80% of the budget.

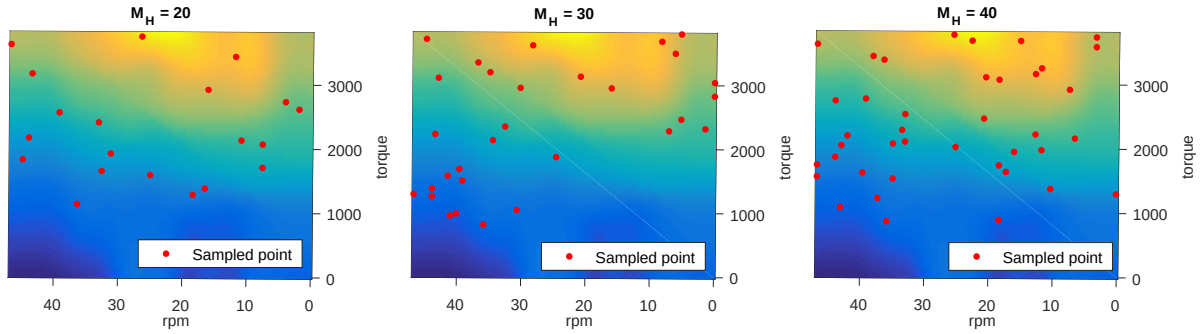


Figure 1.11: Example of the distribution of points included in the design for different budgets.

ing LA data is very inexpensive and a large set of data that effectively covers the sampling space can be collected, any legitimate design of experiment may provide sufficient information about the LA design space. However, when the size of LA data is not very large, a design that better captures the design space, by for example an adaptive approach, can be beneficial.

Given the lowest accuracy data is available, the proposed method may be extended to the situations in which data can be collected from three or more sources with different fidelity levels. For this purpose, our proposed method can be used in a nested or hierarchical fashion to collect data from higher fidelity levels. For example, when three levels of fidelity, low accuracy (LA), medium accuracy (MA), and high accuracy (HA) exists, one can first build a base surrogate model using the LA data and employ our approach to collect MA data and construct a more accurate model (intermediate model) by fusion of LA and MA data. Then the intermediate model can be used as the base model in our approach to collect HA data and the final model.

1.7 Conclusion

In many applications, inexpensive high-density but low-accuracy data is fused with high-accuracy data obtained from expensive experiments to improve the model estimation. Current approaches assume that the HA data is already measured and focus on techniques to integrate the LA and HA data. This chapter considers the problem of selecting high-accuracy data points to be fused with available LA data. In particular, an adaptive approach that takes advantage of the LA data as well as the previously selected HA points is proposed to sequentially select the next HA point. The proposed approach modifies the concept of expected improvement and combines it with the link model used in data fusion to collect as few HA data points as possible so when combined with the available LA data, a more accurate surrogate model is achieved. The proposed adaptive data fusion approach is compared and contrasted with three different benchmarks in several simulation and case studies. The

simulation studies illustrated the strength of our method in modeling a non-stationary surface with a few HA data points when biased LA data of a surface was available. Two case studies illustrated the benefit of our approach in real-world applications. The first case study modeled a freeform surface by collecting (from a CMM) a few HA points and integrating them with a large number of available LA data points measured by a structured laser scanner. The results indicated that our approach selects more points at the locations with larger local variations, which resulted in a more accurate model in terms of prediction errors. In the second case study, the performance measurements of a vehicle engine were exploited to collect HA data from a similar engine to generate an accurate surrogate model for ECU calibration. It was observed that the model generated by the proposed adaptive method produces more accurate predictions compared to the models constructed by the benchmarks.

This work assumes that LA data is already available and utilizes the information from such data to sample HA data. However, when low-accuracy data is not available, one may or may not consider sampling LA data. Decision to sample LA data to be used for sampling HA data depends on the relative cost of sampling and the accuracy of LA and HA data and requires further analysis as a future work.

CHAPTER 2

PROCESS MODELING AND PREDICTION WITH LARGE NUMBER HIGH-DIMENSIONAL VARIABLES USING FUNCTIONAL REGRESSION

2.1 Introduction

In various real-world systems, a response variable (e.g., a quality measure) is influenced, regulated, or related to some input variables (e.g., process variables). Revealing and learning the relationship between the response variable and the set of covariates/predictors is of special importance in system modeling, prediction, and optimization. This is a well-studied problem in regression, design and analysis of experiments, and response surface methodology, especially when the variables are scalar. However, in many applications, system variables are represented in the form of high-dimensional (HD) data, such as waveform signals [29, 30]. Profile-on-profile regression methods have been introduced for modeling systems with profile inputs and output [31, 32, 33, 34, 35, 36, 37, 38, 39]. Nevertheless, these approaches are not suitable for situations in which only a small fraction of inputs are informative. In reality, however, most industrial processes generate a large number of waveform signals from which only a small number is related to the response variable. For example, among a large number of variables measured by sensors, only a few of the internal combustion engine state variables influence the exhaust gas aftertreatment process [40]. The effectiveness of this process is measured by a relative air/fuel ratio (λ) that if falls below an acceptability threshold and remains under the threshold for a given time range (λ-undershoot fault) causes uncontrolled emissions of nitrogen oxides (NOx) during the regeneration phase of a catalyst system. Constructing a model that estimates the lambda signal based on internal combustion engine state variables can be used to better control the lambda values as well as to identify a faulty λ-sensor by comparing the estimates and

sensor readings. Unfortunately, however, a large number of redundant input signals creates a challenge for the existing profile-on-profile regression methods in constructing such a model.

The challenge of high dimensionality in data analytics usually refers to either I) large number of samples, II) large number of variables, or III) high dimensionality of each of the variables (i.e., functional variables). In this chapter, we are dealing with cases II and III. Beside the computational issues, both of these challenges cause sever overfitting in parameter estimation and require dimensionality reduction and sparsity structures to be imposed. Furthermore, capturing the correlation structure within each of HD variables is critical in accurate parameter estimation. A regular regression approach that considers each point of an HD variable as a separate independent scalar variable will significantly fail for ignoring the correlation structure of the points and severely increasing the number of variables ($p \gg n$). Principal Component Analysis (PCA) as a dimensional reduction approach improves process modeling by reducing the dimension of each HD variable while capturing the linear correlation structure of the points. In fact, the PCA replaces the points observed from an HD variable, with smaller number of points that are a linear combination of original ones, and considers these new points as the scalar variables. However, this approach masks the effect of each variable by merging them into new ones and fails to exploit the ordering structure of, for example, a profile variable. To reduce the size of an HD variable and to capture and exploit the correlation structure of the points, functional data analysis (FDA) has been widely used in various application from climate analysis [33] to healthcare [35, 41, 42] and marketing [36]. In this chapter, we focus on functional regression, which refers to a regression model in which either the inputs, output, or both are in a functional form. In the literature of functional regression, a function usually refers to a profile, i.e., a function with a scalar input ($y = f(t)\,;\, t \in \mathbb{R}$). In this work, we interchangeably use profile and waveform signal. The main idea of the functional regression (to deal with the third challenge) is to reduce the dimension of each functional variable by a set of functional

bases such as Fourier and spline bases. The benefit of employing such bases is that unlike the PCA, it captures the nonlinear correlation structure between the points of a function.

In the functional regression analysis, a large body of work focuses on regression models with functional inputs and scalar response [31, 43, 44, 45, 46]. Most of these works consider a linear relationship between the scalar response and the functional input, i.e., $y = \alpha + \int_s \beta(s)\, x(s)\, ds + \epsilon$, where α is an intercept, $x(s)\,;\ s \in \mathbb{R}$ is a profile, $\beta(s)\,;\ s \in \mathbb{R}$ is the profile parameter, and ϵ is a noise term, and some extend the model to a generalized functional linear model [47]. The goal of these works is to estimate the parameter α and the functional parameter $\beta(s)$ by minimizing a loss function that may include smoothing and sparsity penalties such as L_2, L_1, or group lasso penalties [43, 46, 44]. These penalties are usually used to alleviate the overfitting issue. Another group of works considers the case in which a functional response is modeled by a set of scalar inputs, i.e., $y(t) = \alpha(t) + X\beta(t)\,;\ t \in \mathbb{R}$ where X is the design matrix and the goal is to estimate functional parameters $\alpha(t)$ and $\beta(t)$. A complete summary of these works can be found in [48]. Almost all these works focus on the case in which $y(t)$ is a profile, with the exceptions of [49] and [50], which considered images as the response.

A more recently emerged group of works in the field of functional regression estimates a functional response with a functional input [31, 32, 33, 34, 35, 36, 37, 38, 39]:

$$y(t) = \alpha(t) + \int \beta(s,t)\, x(s)\, ds + \epsilon(t)\,,$$

where $\alpha(t)\,;\ t \in \mathbb{R}$ is a functional intercept, $x(s)\,;\ s \in \mathbb{R}$ is a functional input (a profile), $\beta(s,t)\,;\ s,t \in \mathbb{R}$ is a functional parameter, and $\epsilon(t)$ is a random process. Such models are extensively used in healthcare [35, 41, 42] and marketing [36] applications and showed promising performances. Unfortunately, however, most of the available methods are designed for a single functional input to predict a functional response. Furthermore, those that allow for multiple profile inputs (e.g., [36, 39]) fail to handle situations in which only

small portion of the inputs are informative. That is, they are not capable of sensor selection, i.e., selecting the most informative functional inputs. Finally, these works are limited to the profile-on-profile regression (i.e, $s, t \in \mathbb{R}$) and do not discuss the case with higher dimensional inputs or output, for example when $t \in \mathbb{R}^2$. The application of such extension can be found in neuroscience [50]. The main contribution of this work is that we extend previous profile-on-profile regression methods to situations in which a large number of inputs are available, but only small fraction of these inputs are informative. This situation is handled by introducing a functional group lasso penalty that performs variable selection to identify which inputs are most informative. Other contributions of this work are as follows: we extend the profile-on-profile regression to situations with HD functions, and demonstrate the connection between functional regression methods and tensor algebra, and use the tensor calculus to improve the space and computational complexity of the proposed method.

The main goal of this chapter is to develop a functional regression methodology that takes in a large number of functional covariates (relative to the sample size) to estimate an (HD) functional response (e.g., an image or a profile) and is capable of removing non-informative functional inputs. Similar to other functional regression approaches, we will start by using a set of basis functions that jointly represent the response and covariate functions. The use of basis functions will effectively reduce the dimension of the parameter space, making the parameter estimation tractable. To estimate the parameters of the model, we define a penalized least square loss with two sets of penalties: One in the form of a L_2 penalty for global smoothness of the estimations and one in the form of a group lasso (L_1) penalty. The group lasso penalty is in fact used for functional variable selection and handles the large number of inputs challenge. The Block Coordinate Decent (BCD) method is employed to develop an algorithm for minimizing the loss function and estimating the regression parameters. In order to reduce the computational issues caused by HD output functions, we define special structures for penalty functions (both $L_1$and L_2) so that the loss function has a closed-form solution in each block of BCD, and the large matrices are

decomposed into the Kronecker product of several smaller matrices. Next, the closed-form solution will be represented by tensors and computed using tensor algebra, further improving the space and computational efficiency, especially when the output is an HD function.

The rest of the article is organized as follows: In Section 2.2, we first formulate the regression model for the case of profile-on-profile regression and employ the BCD algorithm for estimating the parameters. Then, we extend the method to the case in which both inputs and the output are in higher dimensions. In Section 2.3, we describe two simulation studies, both with profile inputs, but one with a profile response and the other one with an image response. In each simulation study, we compare the performance of the proposed method with benchmarks in terms of mean square prediction errors. Two case studies evaluate the performance of the proposed method in Section 2.4. We first consider the estimation of lambda signal in a vehicle engine based on other sensor measurements such as airmass and engine rotational speed. In the next case study, we investigate the sensor estimation and selection problem based on the body motion data [51]. Finally, we summarize the chapter in Section 2.5.

2.2 Proposed method

In this section, we present a functional regression approach to modeling the relationship between a functional response and a set of functional covariates. For simplicity, we begin with 1D functional data, also known as profiles or waveform signals. Then, we generalize the proposed approach to high dimensional functions. Let $y(t) : \mathbb{T} \to \mathbb{R}$; $\mathbb{T} \subset \mathbb{R}$ denotes a functional response and $x_1(s), \cdots, x_p(s)$ represent a set of p covariate functions, where $x_j(s) : \mathbb{S}_j \to \mathbb{R}$; $\mathbb{S}_j \subset \mathbb{R}$. It is assumed that both $y(t)$ and $x_j(s)$ are smooth functions. In a manufacturing process, $y(t)$ can be a quality measure obtained over time or space, and $\{x_1(s), \cdots, x_p(s)\}$ is a set of process variables that influences the final quality of the product. We assume the functional response can be modeled by a linear combination of

covariates as follows:

$$y(t) = \sum_{j=1}^{p} \int x_j(s)\, \beta_j(s,t)\, ds + e(t), \tag{2.1}$$

where $\beta_j(s,t)$ is the j^{th} 2D functional parameter and $e(t)$ is a random process. The goal is to estimate the functional parameters $\beta_j(s,t)$, given a set of M observations, $\{(y_i(t), x_{i1}(s), \cdots, x_{ip}(s))\}_{i=1}^{M}$. In this work, we assume that the sample size M is of moderate size and does not generate computational challenge. We assume that each function $y_i(t)$ is observed over a grid of size n, and each of the covariates $x_{ij}(s)$ is observed over a grid of size m_j. It is also assumed that the grid size of the response function and each of the covariates do not change from sample to sample. The main challenge in estimating the functional parameters $\beta_j(s,t)$ is that, in theory, they are continuous functions with infinite dimension. To address this issue, following [33], we expand the functional covariates, output, and the parameters using a set of infinite-dimensional smooth basis functions. Let $\boldsymbol{\theta}_j(s) \in \mathbb{R}^{k_{x_j}}$ and $\boldsymbol{\psi}(t) \in \mathbb{R}^{k_y}$ be the vectors of $k_{x_j} \ll m_j$ and $k_y \ll n$ basis functions evaluated at points s and t for the j^{th} functional covariate and the response function, respectively. Using these basis functions, we expand $\beta_j(s,t)$ as $\boldsymbol{\theta}_j^{\mathsf{T}}(s)\, B_j \boldsymbol{\psi}(t)$, where $B_j \in \mathbb{R}^{k_{x_j} \times k_y}$ is a matrix of coefficients of the basis functions that should be estimated. This expansion transforms the functional coefficient with infinite dimensions to a set of finite parameters, B_j. Notice that the use of basis functions significantly reduces the number of parameters to be estimated from $\sum_{j=1}^{p} m_j n$ to $\sum_{j=1}^{p} k_{x_j} k_y$. Defining $\boldsymbol{y}(t) := [y_1(t), \cdots, y_M(t)]^{\mathsf{T}}$ and $\boldsymbol{x}_j(s) := [x_{1j}(s), \cdots, x_{Mj}(s)]^{\mathsf{T}}$, we can rewrite (2.1) as

$$\boldsymbol{y}(t) = \sum_{j=1}^{p} \int \boldsymbol{x}_j(s)\, \boldsymbol{\theta}_j^{\mathsf{T}}(s)\, B_j \boldsymbol{\psi}(t)\, ds + \boldsymbol{e}(t). \tag{2.2}$$

To further simplify (2.1), we define $Z_j := \int \boldsymbol{x}_j(s)\, \boldsymbol{\theta}_j^{\mathsf{T}}(s)\, ds$, where $Z_j \in \mathbb{R}^{M \times k_{x_j}}$, $Y := [\boldsymbol{y}(t_1), \cdots, \boldsymbol{y}(t_n)] \in \mathbb{R}^{M \times n}$, $\Psi = [\boldsymbol{\psi}(t_1), \boldsymbol{\psi}(t_2), \cdots, \boldsymbol{\psi}(t_n)] \in \mathbb{R}^{k_y \times n}$, and finally the combined error as $E = [\boldsymbol{e}(t_1), \boldsymbol{e}(t_2), \cdots, \boldsymbol{e}(t_n)] \in \mathbb{R}^{M \times n}$. Then, the matrix form of (2.1)

is given by

$$Y = \sum_{j=1}^{p} Z_j B_j \Psi + E, \tag{2.3}$$

which can be further simplified by using a Kronecker product as

$$\boldsymbol{y} = \sum_{j=1}^{p} D_j \boldsymbol{b}_j + \boldsymbol{e}, \tag{2.4}$$

where $D_j = Z_j \otimes \Psi^{\mathrm{T}}$ and $\boldsymbol{y}$, $\boldsymbol{b}_j$, and $\boldsymbol{e}$ are $vec\left(Y^{\mathrm{T}}\right)$, $vec\left(B_j^{\mathrm{T}}\right)$, and $vec\left(E^{\mathrm{T}}\right)$, respectively. The $vec(X)$ operator stacks the columns of a matrix X into a vector. To estimate the vector of coefficients $\boldsymbol{b}_j$ in (2.4), we minimize a penalized least square loss function with weighted grouped L_1 and L_2 penalties. The grouped L_1 penalty encourages the model sparsity at the covariate level and hence can be used for identifying important functional variables. This sparsity as well handles the potential issue of overfitting caused by large number of inputs. The weighted L_2 norm will result in smooth estimates for functions $\beta_j\left(s,t\right)$. Such a penalized loss function is given by

$$\begin{aligned} L\left(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_p\right) =& \frac{1}{2}\left|\left|\boldsymbol{y} - \sum_{j=1}^{p} D_j \boldsymbol{b}_j\right|\right|_2^2 \\ &+ \frac{\lambda}{2}\sum_{j=1}^{p} \boldsymbol{b}_j^{\mathrm{T}} P_j \boldsymbol{b}_j \\ &+ \gamma \sum_{j=1}^{p} \sqrt{q_j \boldsymbol{b}_j^{\mathrm{T}} W_j \boldsymbol{b}_j} \end{aligned} \tag{2.5}$$

In this loss function, the second term is an L_2 norm penalty in which $P_j = P_{j,s} + P_{j,t}$ is a matrix that penalizes the roughness of functions $\beta_j\left(s,t\right)$ in both t and s by $P_{j,t}$ and $P_{j,s}$, respectively. We define these two matrices as $P_{j,t} = \left[\int \boldsymbol{\theta}_j\left(s\right)\boldsymbol{\theta}_j^{\mathrm{T}}\left(s\right)\right] \otimes \left[\int L_t\boldsymbol{\psi}\left(t\right) L_t\boldsymbol{\psi}^{\mathrm{T}}\left(t\right)\right]$ and $P_{j,s} = \left[\int L_s\boldsymbol{\theta}_j\left(s\right) L_s\boldsymbol{\theta}_j^{\mathrm{T}}\left(s\right)\right] \otimes \left[\int \boldsymbol{\psi}\left(t\right)\boldsymbol{\psi}^{\mathrm{T}}\left(t\right)\right]$, where L_t and L_s are differentiating operators (e.g., second derivatives) with respect to t and s. In the case of orthogonal bases such as eigenvectors, wavelets, or orthogonal splines, matrices $\left[\int \boldsymbol{\theta}_j\left(s\right)\boldsymbol{\theta}_j^{\mathrm{T}}\left(s\right)\right]$ and

$\left[\int \boldsymbol{\psi}(t)\boldsymbol{\psi}^{\mathrm{T}}(t)\right]$ reduce to identity matrices. Appendix A provides detailed information on how these penalty matrices impose smoothness on $\beta_j(s,t)$. The third term of the loss function performs variable selection similar to the standardized group lasso [52]. That is, it shrinks all elements of $\boldsymbol{b}_j$ to zero if its corresponding covariate function $\boldsymbol{x}_j(s)$ is not significant. In this term, q_j is the size of each group (i.e., the size of vector $\boldsymbol{b}_j$) and $W_j := D_j^{\mathrm{T}} D_j + \lambda P_j$ is a weight matrix. Finally, λ and γ are tuning parameters.

To minimize the loss function in (2.5), one may apply a standard convex optimization method such as interior-points algorithms. However, as shown in Proposition 2.1, the special structure of W_j can transform the loss function in a way that a closed-form solution for each $\boldsymbol{b}_k$ is obtained, given all $\boldsymbol{b}_j;\ j \neq k$, . Therefore, we use a BCD algorithm, in which each group of parameters $\boldsymbol{b}_j;\ j = 1, \cdots, p$ is considered as a block.

Proposition 2.1. *Given all $\boldsymbol{b}_j$'s; $j \neq k$, the minimizer of the loss function in (2.5) has a closed-form solution as*

$$\boldsymbol{b}_k = \left(1 - \frac{\sqrt{q_k}\gamma}{\left|\left|V_k^{-1} D_k^{\mathrm{T}} \boldsymbol{r}_k\right|\right|_2}\right)_+ W_k^{-1} D_k^{\mathrm{T}} \boldsymbol{r}_k, \tag{2.6}$$

where $\boldsymbol{r}_k = \boldsymbol{y} - \sum_{l \neq j} D_l \boldsymbol{b}_l$, $(x)_+ = \max\{x, 0\}$, and V_k is obtained by Cholesky decomposition of W_k, i.e., $W_k = V_k^{\mathrm{T}} V_k$. The solution in each step is soft-thresholding of an equivalent ridge regression problem.

The proof is given in Appendix B. This soft-thresholding solution sets the value of $\boldsymbol{b}_k$ to zero if the k^{th} covariate does not explain the variation in y, and therefore performs variable selection in the context of functional regression. The penalty terms P_j assure the smoothness of the functional parameters $\beta_j(s,t)$ and extend the soft-threshholding solution to under-determined problems.

The BCD algorithm optimizes the loss function with respect to each block, given other blocks, and iterates until convergence. Within each iteration i, the algorithm recalculates

all the p blocks by solving

$$\boldsymbol{b}_k^i = \arg\min_{b_k} L\left(\boldsymbol{b}_1^i, \cdots, \boldsymbol{b}_{k-1}^i, \boldsymbol{b}_k, \boldsymbol{b}_{k+1}^{i-1}, \cdots, \boldsymbol{b}_p^{i-1}\right), \tag{2.7}$$

where $\boldsymbol{b}_l^i$ $(l < k)$ is the value of $\boldsymbol{b}_l$ obtained in the l^{th} step of iteration i, and $\boldsymbol{b}_l^{i-1}$ $(l > k)$ is the value of $\boldsymbol{b}_l$ obtained in the iteration $i-1$. Algorithm 1 summarizes the estimation procedure outlined above. As can be seen, the algorithm contains one inner loop of size p that calculates the values of $\boldsymbol{b}_j$ and one outer loop that checks for convergence. Note that, as the loss function is convex with separable non-differentiable parts, the BCD algorithm has a convergence guarantee.

Algorithm 1 Estimation procedure of the parameters b_j using the BCD algorithm

Set a convergence threshold $\epsilon > 0$
Initilize $\mathbf{b}_j$ for all j
Set i = 1
Set $L_1 - L_0 = \inf$ and $L_0 = -\inf$
while $|L_i - L_{i-1}| > \epsilon$ **do**
    for j = 1: p **do**
        find b_j^i according to (2.6)
    end for
    calculate $L_i = L\left(\mathbf{b}_1^i, \mathbf{b}_2^i, \cdots, \mathbf{b}_p^i\right)$
    calculate $L_i - L_{i-1}$
    i = i + 1
end while

2.2.1 Extension to high dimensional functions

We extend our proposed method to the case in which both response and covariates are HD functions (e.g., 2D functions). Specifically, we assume $y(t) : \mathbb{R}^d \to \mathbb{R}$ and $x_j(s) : \mathbb{R}^{l_j} \to \mathbb{R}$ for $j = 1, \cdots, p$, where $t = \left(t^{(1)}, t^{(2)}, \cdots, t^{(d)}\right) \in \mathbb{R}^d$ and $s = \left(s^{(1)}, s^{(2)}, \cdots, s^{(l_j)}\right) \in \mathbb{R}^{l_j}$. We consider a grid of size $n = n_1 \times n_2 \times \cdots \times n_d$ over the domain of the response function and a grid of size $m_j = m_{j1} \times m_{j2} \times \cdots \times m_{jl_j}$ over the domain of the j^{th} covariate function. Note that n_k and m_{jk} denote the grid size over the k^{th} dimension of space $\mathbb{R}^d$ and $\mathbb{R}^{l_j}$, respectively. Conceptually, The extension requires defining appropriate basis functions

for response, covariate, and functional parameters. Let us define $\boldsymbol{\psi}(t) := \boldsymbol{\psi}_1\left(t^{(1)}\right) \otimes \boldsymbol{\psi}_2\left(t^{(2)}\right) \otimes \cdots \otimes \boldsymbol{\psi}_d\left(t^{(d)}\right)$ and $\boldsymbol{\theta}_j(s) := \boldsymbol{\theta}_{j1}\left(s^{(1)}\right) \otimes \boldsymbol{\theta}_{j2}\left(s^{(2)}\right) \otimes \cdots \otimes \boldsymbol{\theta}_{jl_j}\left(s^{l_j}\right)$ to be the vectors of infinite dimensional basis functions, where $\boldsymbol{\psi}_k\left(t^{(k)}\right)$ and $\boldsymbol{\theta}_{jk}\left(s^{(k)}\right)$ are vectors of basis functions suitable for the k^{th} dimension of the corresponding space evaluated at t and s, respectively. Depending on the behavior of the function, one can select an appropriate basis function (e.g., Fourier or B-spline) in each dimension. For example, if an image has a smooth waveform behavior, Fourier basis may be appropriate. For notation simplicity, we assume that the number of basis functions are k_y and k_{x_j} in all dimensions of the response function y and the covariate function x_j. Therefore, the total number of basis functions for the response and for the j^{th} covariate function are k_y^d and $k_{x_j}^{l_j}$, respectively. Similar to the 1D case, using these basis functions, one can expand the functional parameters and estimate the set of finite parameters by using Algorithm 1. However, when $d > 1$ and $l_j > 1$, the inversion operation on the matrix $W_j \in \mathbb{R}^{k_{x_j}^{l_j} k_y^d \times k_{x_j}^{l_j} k_y^d}$ in (2.6) is extremely expensive as it has the complexity of $O\left(\left(k_{x_j}^{l_j} k_y^d\right)^3\right)$. In order to alleviate this problem, we re-define the penalty matrix P_j in the loss function (2.5) so that matrix W_j can be decomposed as a Kronecker product of $d+1$ matrices, i.e., $W_j = W_{j1} \otimes W_{j2} \otimes \cdots \otimes W_{j,d+1}$. With this decomposition, computation of $W_j^{-1} \in \mathbb{R}^{k_{x_j}^{l_j} k_y^d \times k_{x_j}^{l_j} k_y^d}$ splits into the calculation of $W_{j1}^{-1} \in \mathbb{R}^{k_{x_j}^{l_j} \times k_{x_j}^{l_j}}$ and $W_{ji}^{-1} \in \mathbb{R}^{k_y \times k_y}$ $(i = 2, \cdots, d+1)$, which significantly reduces the computational effort. Proposition 2.2 provides a first step to achieve this goal.

Proposition 2.2. *The matrices $P_{j,s}$ and $P_{j,t}$, $j = 1, 2, \cdots, p$ in loss function (2.5) can be written as the Kronecker product of $d+1$ symmetric positive definite matrices as follows:*

$$P_{j,s} = P_{j,s} \otimes P_{j,s2} \otimes \cdots \otimes P_{j,s(d+1)},$$

$$P_{j,t} = P_{j,t1} \otimes P_{j,t2} \otimes \cdots \otimes P_{j,t(d+1)}.$$

The proof can be found in Appendix A. Now, let us define $W_j := W_{j1} \otimes W_{j2} \otimes \cdots \otimes W_{jd+1}$, where $W_{j1} = \left(Z_j^{\intercal} Z_j + \lambda P_{j,1}\right)$ and $W_{jk} = \left(\boldsymbol{\psi}_{k-1} \boldsymbol{\psi}_{k-1}^{\intercal} + \lambda P_{j,k}\right)$ for $k =$

$2, 3, \cdots, d+1$, with $P_{j,k} = P_{j,sk} + P_{j,tk}$. With this definitions, it can easily be shown that $W_j = D_j^{\mathsf{T}} D_j + \lambda Q_j$ where Q_j is a new penalty matrix comprised of a summation of several penalty terms. For example, for $d = 1$, $Q_j = Z_j^{\mathsf{T}} Z_j \otimes (P_{j,s2} + P_{j,t2}) + (P_{j,s1} + P_{j,t1}) \otimes \boldsymbol{\psi}_1 \boldsymbol{\psi}_1^{\mathsf{T}} + \lambda (P_{j,s1} + P_{j,t1}) \otimes (P_{j,s2} + P_{j,t2})$. The first and second terms in Q_j impose a penalty on covariates and the output, and the third term captures the interactions between the penalties, respectively. Using the new definition of the penalty, and decomposing W_j as a Kronecker product of $d+1$ matrices, we can formulate the BCD closed-form solution as

$$b_j^i = \left(1 - \frac{\sqrt{q_j}\gamma}{\left|\left|\left(V_{j1}^{-1} \otimes V_{j2}^{-1} \otimes \cdots \otimes V_{j(d+1)}\right) D_j^{\mathsf{T}} r_j\right|\right|_2}\right)_+ \times \left(W_{j1}^{-1} \otimes W_{j2}^{-1} \otimes \cdots \otimes W_{j(d+1)}^{-1}\right) D_j^{\mathsf{T}} r_j \tag{2.8}$$

where, V_{ji} is obtained by a Cholesky decomposition of W_{ji}, i.e., $W_{ji} = V_{ji}^{\mathsf{T}} V_{ji}$.

2.2.2 Computational and space complexity

As we are dealing with HD data and the proposed approach requires algebraic operations on large matrices, computational and space complexities are two major challenges in implementing our functional regression method. Let us begin by studying the computational complexity of the BCD solution in (2.8). The cost of inverting W_{j1} is $O\left(\left(k_{x_j}^{l_j}\right)^3\right)$ and the cost of inverting W_{jk}, $k = 2, \cdots, d+1$ is $O\left(k_y^3\right)$. Therefore, the total complexity of matrix inversion operations is $O\left(k_{x_j}^{3l_j} + dk_y^3\right)$. The complexity of computing $W_{j1}^{-1} \otimes W_{j2}^{-1} \otimes \cdots \otimes W_{j(d+1)}^{-1}$, given the inverted matrices, is $O\left(\left(k_{x_j}^{l_j} k_y^d\right)^2\right)$. As a result, the total complexity of calculating $W_j^{-1} = W_{j1}^{-1} \otimes W_{j2}^{-1} \otimes \cdots \otimes W_{j(d+1)}^{-1}$ is $O\left(k_{x_j}^{3l_j} + dk_y^3 + \left(k_{x_j}^{l_j} k_y^d\right)^2\right)$, which is much smaller than the complexity of inverting the original matrix $W_j = D_j^{\mathsf{T}} D_j + \lambda (P_{j,s} + P_{j,t})$ in (2.6), which requires $O\left(\left(k_{x_j}^{l_j} k_y^d\right)^3\right)$ operations. As discussed before, this computational improvement is achieved by reformulating the penalties in the loss function. Finally, the total computational complexity of calculating the parameter $\boldsymbol{b}_j^i$ (i.e., the parameter of the j^{th} predictor in the i^{th} step) is

$O\left(k_{x_j}^{3l_j} + dk_y^3 + \left(k_{x_j}^{l_j}k_y^d\right)^2 + k_{x_j}^{l_j}k_y^d Mn\right)$, where $n = n_1 \times n_2 \times \cdots \times n_d$. Notice that, $n_i \in O(k_y)\,;\, i = 1, \cdots, d$, and therefore the complexity of calculating $\boldsymbol{b}_j^i$ can be written as $O\left(k_{x_j}^{3l_j} + dk_y^3 + \left(k_{x_j}^{l_j}k_y^d\right)^2 + k_{x_j}^{l_j}k_y^{2d} M\right)$. We further reduce this complexity by using tensor algebra.

Besides the computational complexity, the space complexity of both matrices W_j^{-1} and $D_j = Z_j \otimes \boldsymbol{\psi}_1^{\mathsf{T}} \otimes \boldsymbol{\psi}_2^{\mathsf{T}} \otimes \cdots \otimes \boldsymbol{\psi}_d^{\mathsf{T}}$ that are $O\left(\left(k_{x_j}^{l_j}k_y^d\right)^2\right)$ and $O\left(Mnk_{x_j}^{l_j}k_y^d\right)$, respectively generates a new challenge. In both cases the space complexity is exponential in d and l_j and storing these matrices may become an issue. Fortunately, it is not necessary to store these large matrices as we only need to calculate $W_j^{-1}D_j^{\mathsf{T}}r_j$. This observation in fact reduces the computational complexity as well. Let us define $A_{j1} = W_{j1}^{-1}Z_j^{\mathsf{T}} \in \mathbb{R}^{k_{x_j}^{l_j} \times M}$ and $A_{jk} = W_{jk}^{-1}\boldsymbol{\psi}_{k-1} \in \mathbb{R}^{k_y \times n_{k-1}}$, $k = 2, 3, \cdots, d+1$. It is straightforward to show that

$$W_j^{-1}D_j^{\mathsf{T}}r_j = (A_{j1} \otimes \cdots \otimes A_{jd+1})\, r_j. \tag{2.9}$$

In order to calculate $(A_{j1} \otimes \cdots \otimes A_{jd+1})\, r_j$, we may first compute $(A_{j1} \otimes \cdots \otimes A_{jd+1})$ and then multiply it by r_j. However, $(A_{j1} \otimes A_{j2} \otimes \cdots \otimes A_{jd+1})$ has a very large space complexity and its direct computation should be avoided. This can efficiently be done using the equivalency between the Kronecker and tensor products given in Proposition 2.3. We first introduce some preliminaries on tensor and multi-linear algebra. Let $\mathcal{R}$ denote an order-n tensor with $\mathcal{R} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$, where I_i is the dimension of the i^{th} mode of tensor $\mathcal{R}$. We also denote a mode-j matricization of tensor $\mathcal{R}$ with $R_{(j)} \in \mathbb{R}^{I_j \times I_{(-j)}}$, whose columns are the mode-j fibers of the corresponding tensor $\mathcal{R}$, and $I_{(-j)} := I_1 \times I_2 \times \cdots \times I_{j-1} \times I_{j+1} \times \cdots \times I_n$. The mode-$j$ product of a tensor $\mathcal{R}$ by a matrix $A \in \mathbb{R}^{L \times I_j}$ 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 $(\mathcal{R} \times_j A)_{i_1, i_2, \cdots, i_{j-1}, l, i_{j+1}, \cdots, i_K} = \sum_{i_j=1}^{I_j} \mathcal{R}_{i_1, \cdots, i_j, \cdots, i_K} A_{l, i_j}$. Using Proposition 2.3, we can significantly decrease the computational and storage complexity of $A_{j1} \otimes A_{j2} \otimes \cdots \otimes A_{jd+1}$.

Proposition 2.3. *Let $\mathcal{R} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$ be an n-way tensor with dimensions $I_1, I_2, \cdots, I_n$.*

Moreover, let $A_k \in \mathbb{R}^{J_n \times I_n}$ *for* $k = 1, 2, \cdots, d+1$. *Then,*

$$\mathcal{V} = \mathcal{R} \times_1 A_1 \times_2 A_2 \times_3 \cdots \times_{d+1} A_{d+1} \iff$$
$$V_{(d+1)} = A_{d+1} R_{(d+1)} \left(A_d \otimes A_{d-1} \otimes \cdots \otimes A_1\right)^{\mathtt{T}}$$

where V *and* R *are the* $d+1$ *mode matricization of tensors* $\mathcal{V}$ *and* $\mathcal{R}$*, respectively.*

Proof is given in Proposition 3.7 of [53] with minor notation adjustments. According to Proposition 2.3, we only need to store matrices A_{jk} and compute $\mathcal{R}_j \times_1 A_{j1} \times_2 A_{j2} \times_3 \cdots \times_{d+1} A_{jd+1}$. This implies that the space complexity is now $O\left(k_{x_j}^{l_j} M + \sum_{k=1}^{d} k_y n_k\right)$, which is linear in d. In practice, the inputs are usually profiles obtained from sensor readings over time (i.e., $l_j = 1;\ j = 1, \cdots, p$) and the output is either a profile or an image (i,e., d is at most equals two). Assuming $l_j = 1;\ j = 1, \cdots, p$ and $d = 2$, the computational complexity of our approach for calculating $\boldsymbol{b}_j$ in each iteration is about $O\left(k_{x_j}^3 + k_{x_j} k_y^2 M + k_{x_j} k_y^3\right)$. Assuming that $k_y \simeq k_{x_j} = k$, the computational complexity is about $O\left(Mk^3 + k^4\right)$. In this scenario, computing $\boldsymbol{b}_j$ directly from (2.6) requires at least $O\left(k^9\right)$ operations. Note that we are required to compute p parameters separately (i.e., $\boldsymbol{b}_j;\ j = 1, \cdots, p$) and therefore computing all the parameters is linear in p.

Using the tensor algebra, we can transform the BCD closed-form solution in (2.8) and write it in a tensor format as

$$\mathcal{B}_j^i = \omega \mathcal{R}_j \times_1 A_{j1} \times_2 A_{j2} \times_3 \cdots \times_{d+1} A_{jd+1}, \tag{2.10}$$

where

$$\omega = \left(1 - \frac{\sqrt{q_j}\gamma}{\left|\left|\left(V_{j1}^{-1} \otimes V_{j2}^{-1} \otimes \cdots \otimes V_{j(d+1)}\right) D_j^{\mathtt{T}} r_j\right|\right|_2}\right),$$

$\mathcal{B}_j^i \in \mathbb{R}^{k_{x_j}^{l_j} \times k_y \times \cdots \times k_y}$, and $\mathcal{R}_j \in \mathbb{R}^{M \times n_1 \times \cdots \times n_d}$ are tensorized versions of b_j^i and r_j vectors.

As a side note, it is worth mentioning the similarity between the Tucker decomposition of a tensor [54] and (2.10). In fact, one way to interpret (2.10) is that similar to the Tucker

decomposition [54], it decomposes $\mathcal{B}_j^i$ into a core tensor $\mathcal{R}_j$ and a set of basis matrices that are known in advance, and shrinks it using a soft-thresholding coefficient. These basis matrices are related to the covariates, output, and selected basis functions $\boldsymbol{\theta}_j(s)$ and $\boldsymbol{\psi}(t)$.

2.2.3 Choice of tuning parameters

Performance of the proposed approach depends on the choice of tuning parameters λ, γ and the basis functions $\boldsymbol{\psi}(t)$ and $\boldsymbol{\theta}(s)$, as well as the number of basis functions $k_{x_j}, j = 1, \cdots, p$ and k_y. In our simulation and case studies, we use B-splines as the basis functions. The selection of the basis functions depends on the functional forms of inputs and the output and should be done based on domain knowledge or some initial analysis. As suggested by [33], one can choose dense basis matrices (i.e., large k_{x_j} and k_y) and penalize them by roughness penalties $P_{s,j}$ and $P_{t,j}$. A few approaches can be found in the literature for choosing λ and γ values (see [55], for example). For large data sets, one can randomly divide the data into three subsets, i.e., training, validation, and test, and use the training and validation subset for tuning the parameters. Otherwise, the k-fold cross-validation (CV) method can be used [55]. An alternative approach is to choose the set of tuning parameters that minimizes the generalized CV (GCV) or one of information criteria such as AIC or BIC. In this work, we perform k-fold CV to obtain λ and γ. More specifically, we divide our training data into k subsamples and select a grid of λ and γ pairs. Then, for each pair on the grid, denoted by (λ_j, γ_j), we train the model using $k-1$ of the subsamples and test the trained model on the remaining subsample to find the CV error. With this procedure, we obtain k CV errors for each pair of (λ_j, γ_j). We consider the average of the errors as the performance measure for a given pair on the grid. Finally, we choose the point (λ^*, γ^*) as the set of tuning parameters that results in the minimum average CV error. In our simulation studies we select λ from $\{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 0.1, 1, 10\}$ and γ from $\{0, 10^{-6}, 10^{-4}, 10^{-2}, 0.1, 1, 10\}$ and use 5-fold CV.

2.3 Performance evaluation using simulation

In this section, we conduct two simulation studies to evaluate the performance of the proposed method (labeled as HDFFR) and compare it with two other existing methods. In the first simulation, we consider a profile-on-profile regression in which a profile data is predicted based on a large number of profile inputs with a small fraction of informative ones. In the second simulation study, we predict an image data using a set of profile inputs. In both simulations, we compare our method to a method proposed by [39], designated as *Sigcomp*. We implement this benchmark method using the R package *FRegSigCom*. In the second simulation, we also compare our approach to the principle component regression method, designated as *PCR*. In the PCR method, first the PCA is applied to the predictor curves and response to extract low dimensional features. Then, a multivariate regression model is performed between the input and output scores. Specifically, we first perform PCA analysis on the samples of each input x_j and the response y and select the first G_{x_j} and G_y PC scores. Next, we fit G_y regression models, each to estimate one of the PC score of y given the selected PC scores of all covariates. To form the functional prediction of the response, the estimated response scores are multiplied by the transpose of G_y principal components. Notice that because the principle components are orthogonal, fitting a separate regression model to each output score is equivalent to fitting a multivariate regression. In order to select G_{x_j} and G_y, one can perform CV. However, because we do not perform CV to select the number of bases (i.e., $k_{x_j}$and k_y) in our proposed approach, we select $G_{x_j} = k_{x_j}$ and $G_y = k_y$. That is we select same number of bases for both approaches. Please note that other approaches, including functional principle component regression and function partial least square approach are designed for a single input and are not suitable for scenarios in which multiple number of inputs exists. In addition the Sigcomp approach can be viewed as a variant of functional principle component regression that allows multiple inputs.

2.3.1 Simulation I: Profile-on-profile regression

In this simulation study, we evaluate the performance of the proposed method in comparison to Sigcomp in estimating and predicting a profile based on a set of p profiles, where only p_{eff} of them are informative. Because the Sigcomp approach has been shown to outperform the PCR in the profile-on-profile scenarios, we do not consider PCR as a benchmark in this simulation study. We follow the simulation study in [39] to generate the input and output curves. We first randomly generate $\beta_i(s,t)\,;\ i=1,\cdots,p$ as

$$\beta_i(s,t)=\frac{1}{p^2}\sum_{k=1}^{3}\gamma_{ki}(t)\,\psi_{ki}(s)\,,$$

where $\gamma_{ki}(t)$ and $\psi_{ki}(s)$, $k=1,\ 2,\ 3$ and $i=1,\cdots,p$ are Gaussian processes (GP) with covariance function $\Sigma_1(z,z')=e^{-(5|z-z'|)^2}$. Next, we construct functional predictors as follows: first, a set of curves $w_1,\ w_2,\cdots,\ w_p$ are sampled from a Gaussian process with covariance function $\Sigma_2(z,z')=e^{-(3|z-z'|)^2}$. Next, a $p_{eff}\times p_{eff}$ correlation matrix S with diagonal elements equal to one and off-diagonal elements equal to ρ_c is constructed and decomposed as $S=\Delta\Delta^{\mathrm{T}}$, where Δ is a $p_{eff}\times p_{eff}$ matrix. Finally, the predictors at any given point s are obtained by

$$\left(x_1(s),\cdots,x_{p_{eff}}(s)\right)=\left(w_1(s),w_2(s),\cdots,w_{p_{eff}}(s)\right)\Delta^{\mathrm{T}},$$

and

$$\left(x_{p_{eff}+1}(s),\cdots,x_p(s)\right)=\left(w_{p_{eff}+1}(s),\cdots,w_p(s)\right),$$

With this formulation each curve $x_i(s)\,;\ i=1,\cdots,p$ is a Gaussian process with covariance function $\Sigma_2(s,s')$, and for each s, the vector $\left(x_1(s),\cdots,x_{p_{eff}}(s)\right)$ is a multivariate normal distribution with covariance matrix S. Finally, we generate the response curves as

$$y(t)=\sum_{i=1}^{p_{eff}}\int\beta_i(s,t)\,x_i(s)\,ds+\epsilon(t)\,,$$

where $\epsilon(t)$ is generated from a normal distribution with zero mean and $\sigma^2 = 1$. We generate all the input and output curves over $0 < s < 2$ and $0 < t < 1$, and take the samples over an equidistant grid of size $n = 30$.

To evaluate the performance of each method, we generate a set of $M = 30$ samples and randomly divide the data into a training set of size 24 and a test set of size 6. The training set is used to perform CV and to learn the model. With the learned model, we calculate the mean square prediction error (MSPE) and the mean square estimation error (MSEE) using the testing data as follows:

$$MSPE = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} \frac{1}{n} \sum_{k=1}^{n} \left(y_i^{test}(t_k) - \hat{y}_i(t_k) \right)^2$$

and

$$MSEE = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} \frac{1}{n} \sum_{k=1}^{n} \left(y_i^{test}(t_k) - \epsilon(t_k) - \hat{y}_i(t_k) \right)^2$$

We perform this simulation for $p = 10,\ 20,\ 30,\ 40$ with $p_{eff} = 0.1p$. We also take the same number of bases for all the inputs and output, i.e., $k_{x_j} = k_y = 10;\ \forall j = 1, \cdots, p$. We consider two values of ρ_c: When $\rho_c = 0$ the informative input variables are uncorrelated and when it is set to 0.7 they are correlated. In each scenario (a combination of p and ρ_c), we repeat the simulation for 30 times and compute the averages and standard deviations of the MSPE and MSEE. As it is reported in Table 2.1, the proposed approach outperforms Sigcomp for all p and ρ_c. For example, when $p = 20$ and $\rho_c = 0.7$, the MSPE of Sigcomp and HDFFR are 1.874 and 1.471, respectively. The reason is that the irrelevant inputs negatively influence the estimation performance of Sigcomp due to its lack of variable selection capability. The estimation performance of both approaches deteriorate as the number of inputs increases, especially when $p = 40$. Furthermore, both methods benefit from the introduction of correlation (i.e., $\rho_c = 0.7$) among inputs.

Table 2.1: Comparison between the Sigcomp and our proposed method (HDFFR) in terms of MSPE and MSEE. The values within the parenthesis are the standard deviation of the corresponding error calculated over 30 replications.

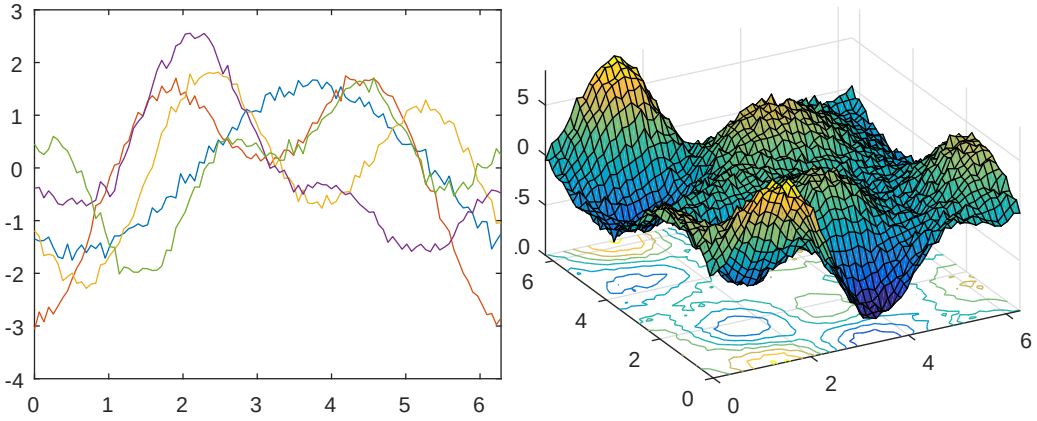
MSPE	ρ_c	*Sigcomp*	*HDFFR*
$p = 10$	0	1.822 (0.247)	1.542 (0.240)
$p = 20$	0	1.992 (0.266)	1.723 (0.374)
	0.7	1.874 (0.202)	1.471 (0.201)
$p = 30$	0	2.358 (0.362)	1.732 (0.340)
	0.7	2.184 (0.317)	1.574 (0.237)
$p = 40$	0	3.357 (0.451)	1.866 (0.432)
	0.7	2.976 (0.385)	1.754 (0.354)
MSEE			
$p = 10$	0	0.802 (0.201)	0.394 (0.157)
$p = 20$	0	0.891 (0.191)	0.630 (0.259)
	0.7	0.831 (0.157)	0.528 (0.198)
$p = 30$	0	1.383 (0.334)	0.773 (0.282)
	0.7	1.131 (0.302)	0.556 (0.257)
$p = 40$	0	2.325 (0.357)	0.899 (0.318)
	0.7	1.969 (0.332)	0.770 (0.315)

2.3.2 Simulation II: Image-on-profile regression

In this simulation study, we evaluate the performance of the proposed method in estimating and predicting an image based on p profile inputs. The data generation model includes functional input variables generated from Fourier bases as follows: We first consider a vector of R Fourier bases $\boldsymbol{f}_R(s) = \left[\sin(s)\ \cos(s)\ \sin(2s), \cdots, h\left(\left\lfloor \frac{R}{2} \right\rfloor s\right)\right]^{\mathsf{T}}$, where $h\left(\left\lfloor \frac{R}{2} \right\rfloor s\right) = \sin\left(\left\lfloor \frac{R}{2} \right\rfloor s\right)$ if R is odd and $h\left(\left\lfloor \frac{R}{2} \right\rfloor s\right) = \cos\left(\left\lfloor \frac{R}{2} \right\rfloor s\right)$, otherwise, and $s \in [0, 2\pi]$. Second, we generate the i^{th} sample of the j^{th} predictor curve as,

$$x_{ji}(s) = \boldsymbol{f}_{R_j}^{\mathsf{T}}(s)\,\boldsymbol{\alpha}_{ji} + w_i(s),$$
$$i = 1, \cdots, M;\ j = 1, \cdots, p$$

where $\boldsymbol{\alpha}_{ji}$ is randomly generated from $N\left(\mathbf{0}, 4I_{R_j}\right)$ and contains the coefficients of the bases, and $w_i(s)$ is an independent random error generated from $N(0, 0.1^2)$. The values



(a) (b)

Figure 2.1: Examples of (a) input profiles and (b) response functions in the first simulation study with $\tau = 0.3$

of s are sampled over an equidistant grid of size 100, i.e., $s_i = \frac{2\pi i}{100}$, $i = 1, \cdots, 100$. After generating the input curves, the following model is used to simulate independent realizations of the response:

$$y_i(t) = \sum_{j=1}^{p} \int x_{ji}(s)\, \beta_j(s,t)\, ds + \epsilon_i(t),$$
$$i = 1, \cdots, M.$$

The functional parameter $\beta_j(s,t)$ is generated using $\boldsymbol{f}_{R_j}^{\mathsf{T}}(s)\, B_j \left(\boldsymbol{f}_{R_{1y}}(t) \otimes \boldsymbol{f}_{R_{2y}}(t) \right)$. The $B_j \in \mathbb{R}^{R_j \times R_{1y} R_{2y}}$ is a matrix whose elements are generated randomly from $N(0,4)$ and R_{1y} and R_{1y} are the number of bases used in each direction of the output image. The values of $t \in [0, 2\pi] \times [0, 2\pi]$ are sampled over an equidistant grid of size 50×50. That is, $t = \left(\frac{2\pi i}{50}, \frac{2\pi j}{50}\right)$, $i = 1, \cdots, 50$ and $j = 1 \cdots, 50$. The independent noises $\epsilon(t)$ are generated from $N(0, \tau^2)$. For the purpose of this simulation study, we take $p = 5$, and $R_j = j + 1$; $j = 1, \cdots, p$, and $R_{1y} = R_{2y} = 3$. Figure 2.1 illustrates a few examples of the covariates and response functions generated through this procedure, with $\tau = 0.3$.

Please note that in this study, data generation step employs Fourier bases, which are

Table 2.2: Comparison between the proposed method (HDFFR) and the benchmarks at different level of noise in the output image. As reported, the Sigcomp method significantly fails because it can only handle curve data and vectorizing an image to a curve eliminates spatial correlation structure of the image. The values in the parenthesis reflects the standard deviation of MSPE.

MSPE (std)	PCR	Sigcomp	HDFFR
$\tau^2 = 0.01$	0.027 (0.052)	0.236 (0.012)	0.011 (0.000)
$\tau^2 = 0.04$	0.096 (0.068)	0.312 (0.071)	0.041 (0.000)
$\tau^2 = 0.09$	0.106 (0.057)	0.385 (0.066)	0.092 (0.000)
$\tau^2 = 0.16$	0.206 (0.098)	0.429 (0.051)	0.163 (0.001)
$\tau^2 = 0.25$	0.294 (0.061)	0.535 (0.044)	0.254 (0.001)

different from the the bases (B-splines) used in model training step. For this study, we generate 200 samples from which we select at random 160 samples for CV and training and the remaining samples as the testing data. The mean square prediction error (MSPE) obtained from the test data is used as the performance criterion. We repeat this procedure 50 times and compute the average and standard deviation of the MSPE. In this simulation, we take $k_{x_i} = 24$ $(i = 1, \cdots p)$ bases for the predictor curves and $k_y = 14$ bases in each direction of the output image.

Similar to the previous simulation study, we compare our proposed method to Sigcomp. However, because the Sigcomp cannot directly handle the image data, we vectorize the output to obtain a curve from an image. We also perform PCR for this simulation study. Table 2.2 reports the mean and standard deviation of MSPE at different values of output noise. As reported the proposed method outperforms both the Sigcomp and PCR. The Sigcomp fails significantly because it requires a smooth curve output and vectrization of an image generates points at which the curve is non-smooth. It also eliminates the spatial correlation structure of the image. Even though the PCR produces low prediction errors, it masks the effect of each input on the output by transforming the data into PC space. Our approach produces the lowest MSPE and solves the issue with the PCR.

2.4 Case study

In this section, we further evaluate the effectiveness of our HD function-on-function regression method using two case studies. In the first study, we use the proposed method to estimate λ signal based upon other engine sensors. In the second case, we employ HDFFR to retrieve a joint motion trajectory based on other joint trajectories and evaluate the performance of the variable selection approach.

2.4.1 Case I: Estimation of lambda sensor

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 to 90 seconds, the electronic control unit of the engine is programmed in order 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 [40]. 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 λ signal based on other engine operation signals can help to better adjust and control the engine operation condition and to identify faulty sensors. Figure 2.2 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.

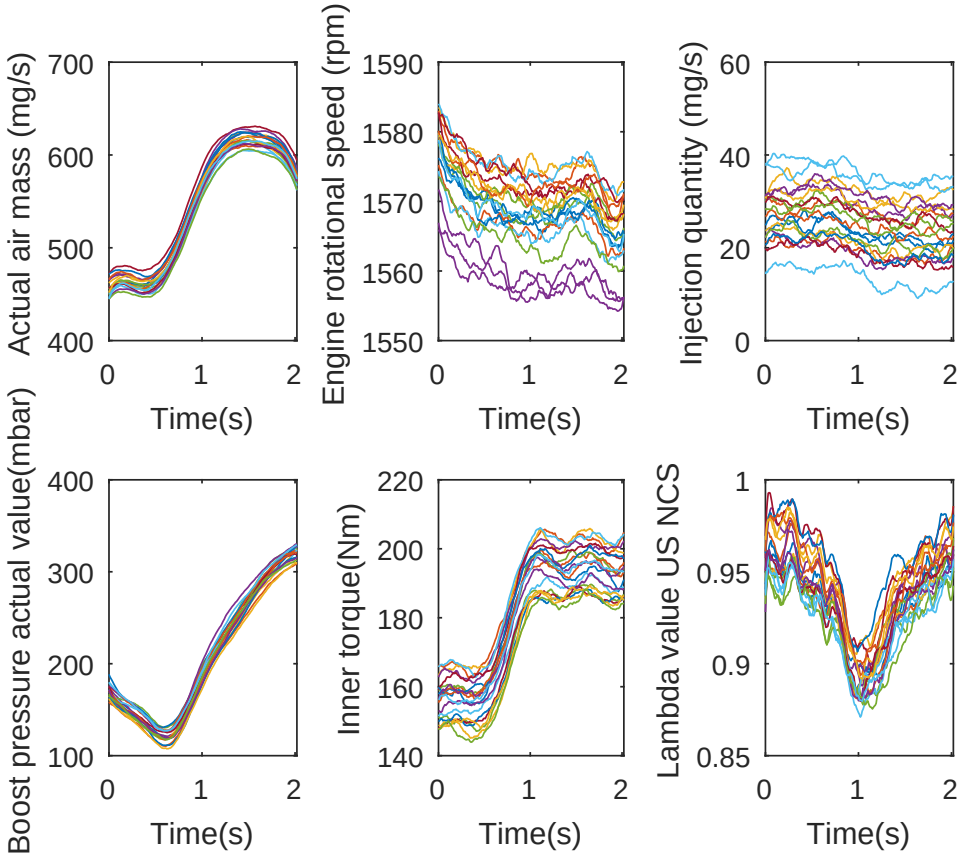


Figure 2.2: Examples of the input signals and the lambda sensor readings during the regeneration phase.

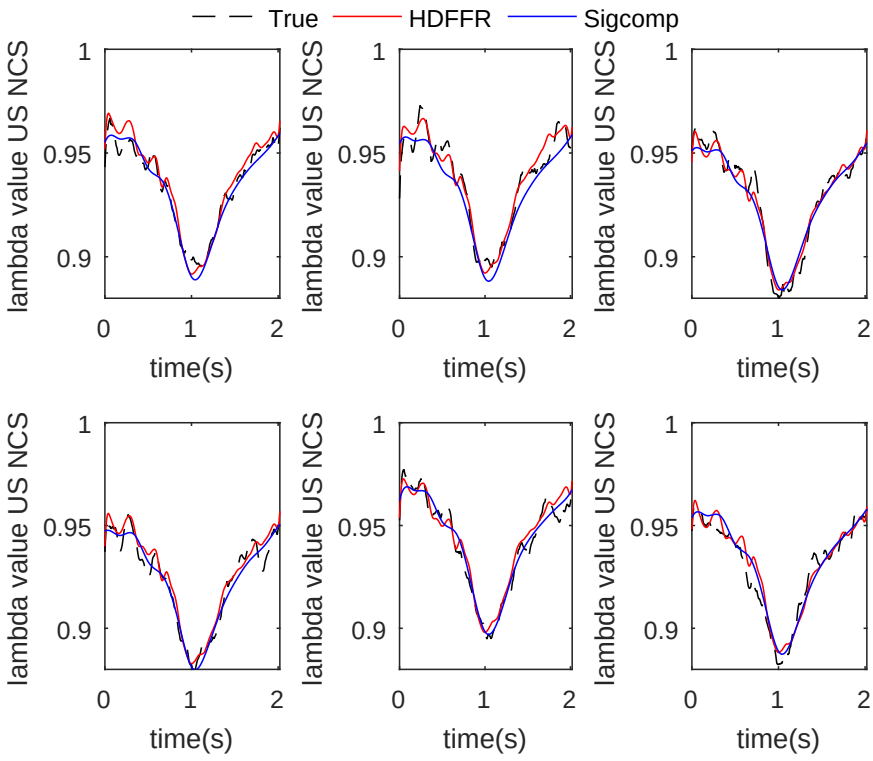


Figure 2.3: Examples of lambda curve predictions

In this case study, we consider 200 samples, each containing 15 input signals and one lambda signal as the output. Among the input signals only five are known to influence the λ signal directly. All signals are measured over a two seconds interval with 203 measurement points. We use the input signals to develop a model that can estimate the lambda value and to identify the informative set of sensors. For this purpose, 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. We repeat this procedure 30 times and calculate the mean and standard deviation of the prediction error for both the proposed method and the Sigcomp approach. For our proposed method, we set $k_{x_i} = k_y = 5$. Figure 2.3 illustrates examples of predicted curves along with the actual ones. As it is illustrated, both approaches construct accurate predictions. Figure 2.4 depicts the boxplot of MSPE obtained over 30 replications of simulation for the proposed method and the Sigcomp. As it is illustrated, the proposed approach outperforms the benchmark in predicting the

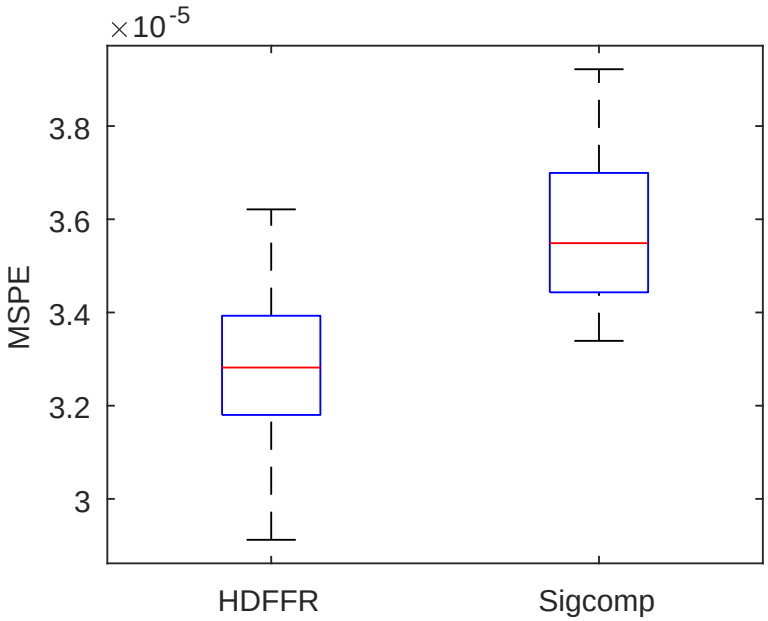


Figure 2.4: MSPE of predicting the lambda curves using HDFFR and the Sigcomp approaches

output curve. The reason is that unlike Sigcomp that combines all the inputs, the proposed approach selects the informative signals and removes the redundant ones when predicting the output curve. To illustrate the performance of our approach in selecting the informative signals, we record the selected signals in each of the 30 iterations. Figure 2.5 illustrates the percentage of times each of the signals has been selected. In this figure, the first five signals are those that are known to influence the lambda curve and are selected in every iteration.

2.4.2 Case II: Joint motion trajectories

This case study is shown to further demonstrate the effectiveness of the proposed method in functional variable selection. Because other benchmarks (e.g., Sigcomp) does not perform variable selection, we only apply our proposed method in this case study without comparing it to other approaches. We apply HDFFR to the body joint motion trajectory data [51]. Figure 2.6b illustrates the motion trajectories of joints located on the hip and neck (sensors 1 and 3), right and left elbows (sensors 6 and 10), and right and left knees (sensors 14 and 18), measured when a person performs a duck gesture (a command in a game) as illustrated

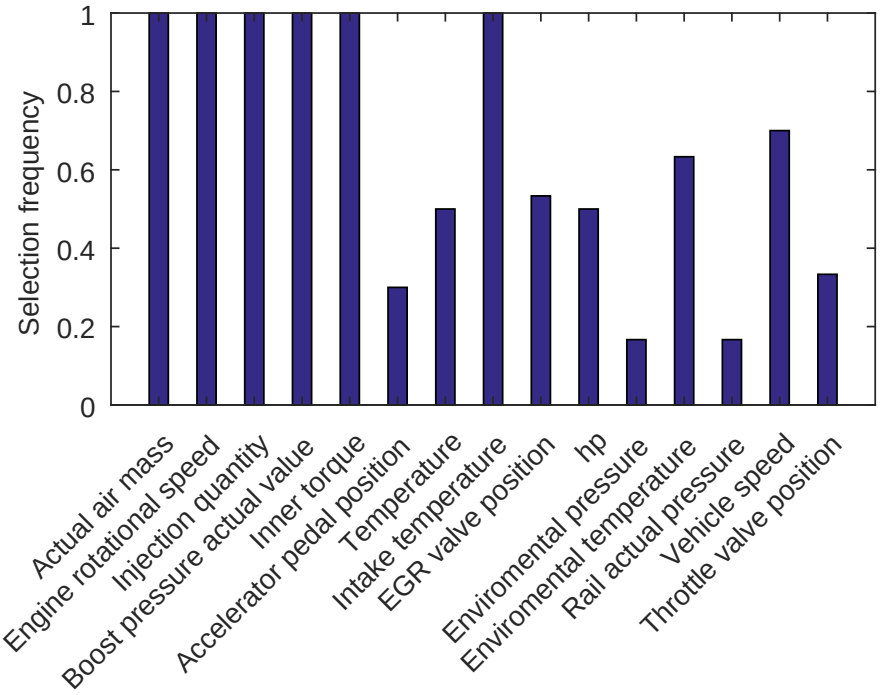
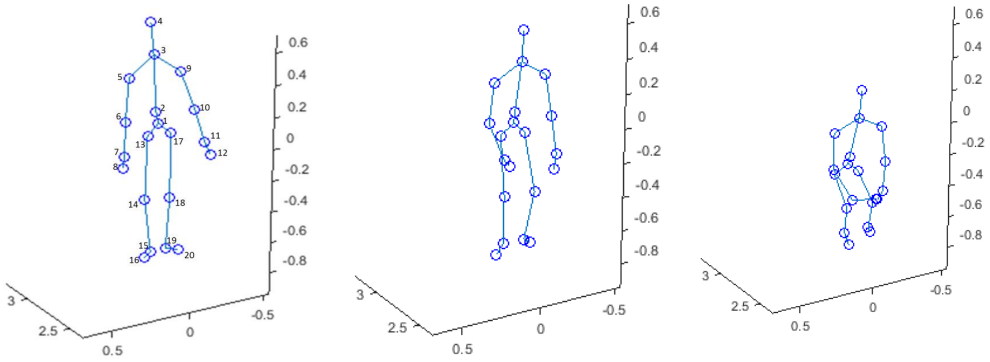


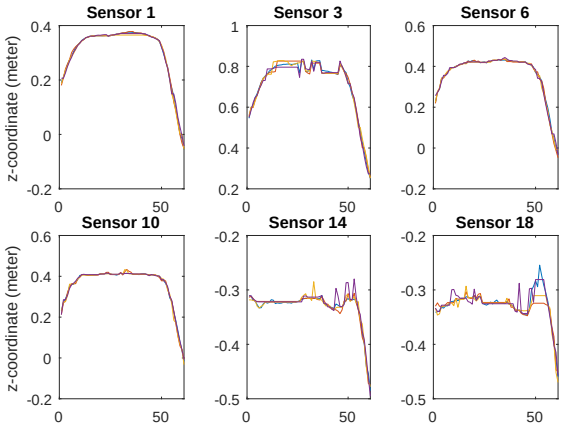
Figure 2.5: Selection frequency of each input curve for prediction of lambda curve. The first five curves from left are known to influence the lambda curve and are selected in every iteration.

in Figure 2.6a. Intuitively, depending on the gesture, the motion trajectory of a joint (e.g., hip) is correlated with the motion trajectories of other joints on the body (e.g., neck).

In a set of experiments, 30 people repeated 12 gestures (e.g., duck and throw), each of which represents a particular command for a game. During each trial, motion trajectories of 20 joints (e.g., neck, elbows, wrists, knees, and ankles) were recorded, with the sampling rate of 30Hz and 2cm accuracy in joint positions. The spatial trajectories were sampled over time in Cartesian coordinates relative to a reference point [51]. The objective of this case study is to identify informative sensors in order to construct a regression model for estimating and retrieving a trajectory given other trajectories' measurements. In this case study, we analyze the duck gesture motion profiles (see Figure 2.6) of a participant who repeated the task nine times. Because the duration of each replication is different, the motion profiles are aligned using the dynamic time warping algorithm [56]. The selected data contains 540 trajectories (20 joint motions $\times$3 xyz coordinates $\times$9 replications). However,



(a)



(b)

Figure 2.6: (a) Position of 20 joints in three different time stamps of a duck gesture; (b) sample measurements at six different joints located on the hip, neck, right elbow, left elbow, right knee, and left knee [51].

we only focus on the z coordinate motions with 180 trajectories. The goal is to identify the sensors that are most informative in estimating the values of the sensor 10 located on the left elbow.

For the purpose of model estimation and validation, we randomly select 8 replications as the training dataset and the remaining one as the test dataset. For the selected training dataset, we further use a 3-fold CV for choosing the tuning parameters and training the prediction models. We repeat this procedure 100 times and in each replication record the sensors selected for prediction of the sensor 10. Figure 2.7 reports the percentage of times each sensor has been selected in the model. The result suggests that the motion trajectory of joints 1,2,3,4,6,7, 8, and 12 corresponding to the hip, back, neck, head, right shoulder, right

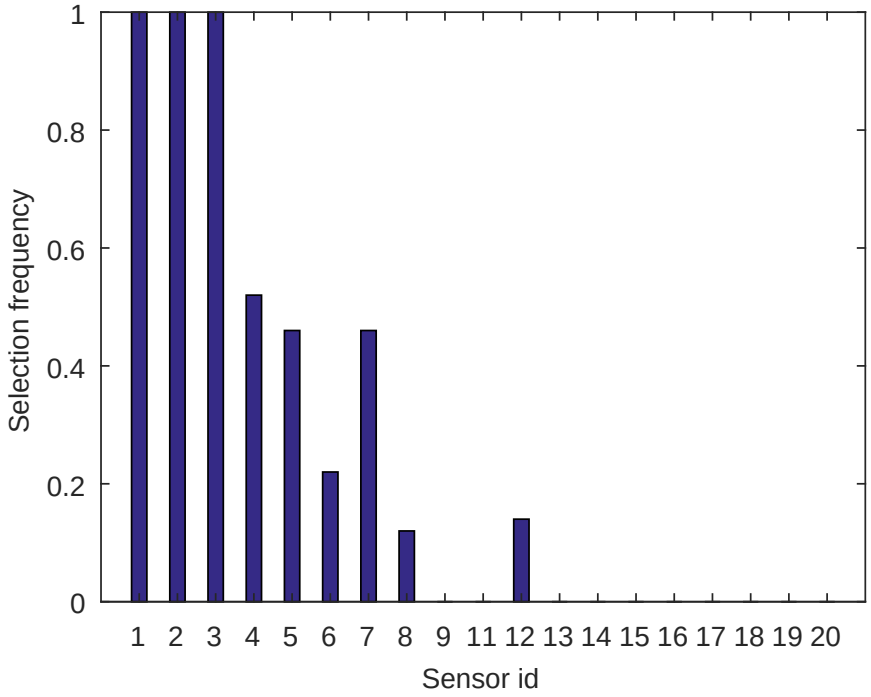


Figure 2.7: Selection frequency of each sensor for prediction of left elbow trajectory. the motion trajectory of joints 1,2,3,4,6,7, 8, and 12 corresponding to the hip, back, neck, head, right shoulder, right elbow, right wrist, right hand, and left hand are the most informative

elbow, right wrist, right hand, and left hand (see Figure 2.6a), are the most informative in estimating the motion of joint 10, i.e., the left elbow. The joints located on the lower body, such as knees and ankles, are excluded, showing no informative power in explaining the motion of joint 10. The exclusion of lower body parts is intuitive, because the trajectory of the left elbow (sensor 10) in the z direction has a larger range than the trajectory of joints in the lower body. Furthermore, note that the left shoulder and wrist are excluded. The reason is that the motion of these parts are highly correlated to the motion of the parts that are selected and therefore are providing redundant information.

2.5 Conclusion

This chapter proposed an HD regression method to predict a functional response (e.g., a profile or an image) through a set of functional predictors. The proposed method expands the functional regression coefficients using smooth basis functions and transforms

the functional problem with infinite dimension into an ordinary least square regression model, which reduces the dimensionality and facilitates parameters estimations. The loss function used for estimating the parameters includes both smoothing and group lasso penalties to handle the under-determined problems, to perform variable selection, and to generate smooth predictions. The BCD algorithm was employed to find the optimal parameters, and it was shown that with a particular structure of group lasso penalty, BCD has a closed-form solution for each block. The smoothing penalty was also modified so that the problem becomes decomposable in each direction of the response function, improving the computational time of the estimation algorithm.

In order to evaluate the performance of the proposed method, we conducted two simulations and two case studies. We compared the performance of the proposed method in the simulation studies to a benchmark method proposed by [39] and another benchmark based on PCA regression. In the first simulation study, the response function was estimated through p profiles. The results show the superiority of our approach when only a small portion of a large number of inputs is informative. In the second simulation study, we considered estimating an image based on $p = 5$ curves. The results showed that the mean square prediction error of the proposed method is significantly smaller than both benchmarks at all noise levels.

We also evaluated the proposed method using data obtained from vehicle engine sensors. The predictions of the lambda sensors. The capability of our approach in variable selection has been reported. Finally, we evaluated the performance of the proposed method in terms of prediction and variable selection using a joint trajectory motion dataset. As expected, the result showed that the motion of the joints on the lower parts of body were not informative in estimation of the left elbow trajectory.

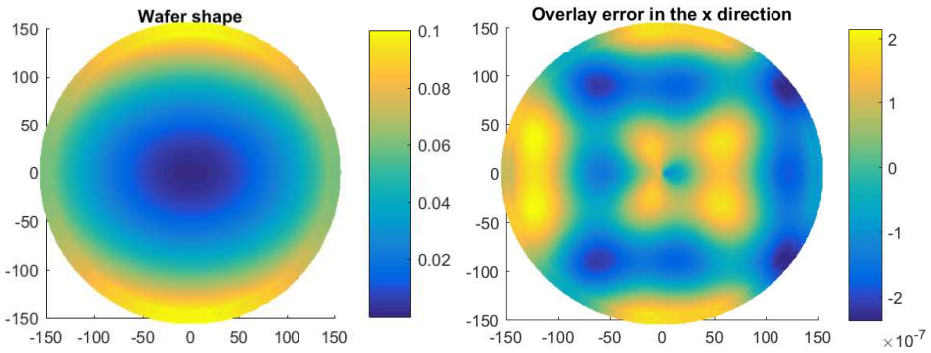
CHAPTER 3

MULTIPLE TENSOR-ON-TENSOR REGRESSION: AN APPROACH FOR MODELING PROCESSES WITH HETEROGENEOUS SOURCES OF DATA

3.1 Introduction

Nowadays, heterogeneous sets of data containing scalars, waveform signals, images, etc. are more and more available. For example, in semiconductor manufacturing, 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 [57, 58], food industries [59], medical decision-making [60], and structural health monitoring [61]. 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 chapter, we refer to non-scalar variables as high-dimensional (HD) variables.

Unfortunately, most works in regression modeling consider scalars and waveform signals [62, 33, 36, 39], and their extension to images or structured point clouds is non-trivial 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, e.g., the temperature and pressure under which a material is operating, and the microstructure of the material [63], 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.



(a) (b)

Figure 3.1: 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. Figure (b) represents 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.

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 3.1. Prediction of the overlay error across a wafer based on the wafer shape can be fed forward to exposure tools for specific corrections [64]. In order 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 an accurate model for a process with a heterogeneous group of variables is twofold: 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 [62, 33, 34, 36, 38, 39]. However, FRM requires a set of basis functions that is usually specified based on domain knowledge rather than a data-driven approach. Recently, [39] 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 [65, 66, 67]. Nevertheless, only a few works in the literature use tensor analysis for regression modeling. [68] 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 [69]. [68] further extended their approach to a generalized linear model for tensor regression in which the scalar output follows any exponential family distribution. [70] 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 [71]. [72] performed the opposite regression and estimated point cloud data as an output using a set of scalar process variables. N-way PLS (N-PLS) is a generalization of PLS offered by [73] 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 [74]. 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, [75] developed a tensor-on-tensor regression (TOT) 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 20000 pixels used in our case study) due to its high space complexity.

The overarching goal of this chapter 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 regression (MTOT). 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 ensure uniqueness, 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 3.2, we introduce notations and review some of the multilinear algebra concepts used in the chapter. In Section 3.3, we formulate the multiple tensor-on-tensor regression model and illustrate the closed-form solution for estimating the parameters. In Section 3.4, we describe four 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). A case study on predicting the overlay errors based on the wafer shape is conducted in Section 3.5. Finally, we summarize the chapter in Section 3.6.

3.2 Tensor Notation and Multilinear Algebra

In this section, we introduce the notations and basic tensor algebra used in this chapter. Throughout the chapter, we denote a scalar by a lower or upper case letter, e.g., a or A; a vector by a boldface lowercase letter and a matrix by a boldface uppercase letter, e.g., $\boldsymbol{a}$ and $\mathbf{A}$; and a tensor by a calligraphic letter, e.g., $\mathcal{A}$. 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 i^{th} 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 $\mathbb{I} = \{I_1, I_2, \cdots, I_l\}$ and $\mathbb{Q} = \{Q_1, Q_2, \cdots, Q_d\}$ be two sets that partition the set $\{I_1, I_2, \cdots, I_l, Q_1, Q_2, \cdots, Q_d\}$, 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, e.g., $||\mathcal{R}||_F^2 = ||\boldsymbol{R}_{(1)}||_F^2$. 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

$$\left(\mathcal{R} \times_j \boldsymbol{A}\right)_{i_1, i_2, \cdots, i_{j-1}, l, i_{j+1}, \cdots, i_K} = \sum_{i_j=1}^{I_j} \mathcal{R}_{i_1, \cdots, i_j, \cdots, i_K} \boldsymbol{A}_{l, i_j}.$$

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 $\boldsymbol{U}_i \in \mathbb{R}^{I_i \times P_i}$ $(i = 1, 2, \cdots, n)$ 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 $\mathcal{A}$, i.e., $P_j \leq I_j$ $(j = 1, 2, \cdots, n)$. 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 $\boldsymbol{B}$:

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

We link the tensor multiplication with the Kronecker product using Proposition 3.1.

Proposition 3.1. *Let* $\boldsymbol{U}_i \in \mathbb{R}^{P_i \times \tilde{P}_i}$ $(i = 1, \cdots, l)$ *and* $\boldsymbol{V}_i \in \mathbb{R}^{Q_i \times \tilde{Q}i}$ $(i = 1, \cdots, d)$*, 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

$$\boldsymbol{T} = (\boldsymbol{U}_l \otimes \boldsymbol{U}_{l-1} \otimes \cdots \otimes \boldsymbol{U}_1)\, \boldsymbol{C}\, (\boldsymbol{V}_d \otimes \boldsymbol{V}_{d-1} \otimes \cdots \otimes \boldsymbol{V}_1)^{\mathtt{T}},$$

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* $\tilde{Q} = \prod_{j=1}^{d} \tilde{Q}_j$*.*

This proposition can be found in [53] (Proposition 3.7, page 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

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

3.3 Multiple Tensor-on-Tensor Regression Framework

In this section, we introduce the multiple tensor-on-tensor (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 $\mathcal{Y}_i \in \mathbb{R}^{Q_1 \times \cdots \times Q_d}$ $(i = 1, \cdots, M)$ and input tensors $\mathcal{X}_{ji} \in \mathbb{R}^{P_{j1} \times \cdots \times P_{jl_j}}$,

$i = 1, \cdots, M$; $j = 1, \cdots, p$, 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:

$$\mathcal{Y}_i = \sum_{j=1}^{p} \mathcal{X}_{ji} * \mathcal{B}_j + \mathcal{E}_i,\ i = 1, \cdots M, \tag{3.1}$$

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. To achieve a more compact representation of the model (3.1), we can combine tensors $\mathcal{Y}_i$, $\mathcal{X}_{ji}$, and $\mathcal{E}_i$ $(i = 1, \cdots, M)$ into one-mode larger tensors $\mathcal{Y} \in \mathbb{R}^{M \times Q_1 \times \cdots \times Q_d}$, $\mathcal{X}_j \in \mathbb{R}^{M \times P_{j1} \times P_{j2} \times \cdots \times P_{jl_j}}$ $(j = 1, 2, \cdots, p)$, and $\mathcal{E} \in \mathbb{R}^{M \times Q_1 \times \cdots \times Q_d}$ and write

$$\mathcal{Y} = \sum_{j=1}^{p} \mathcal{X}_j * \mathcal{B}_j + \mathcal{E}. \tag{3.2}$$

The matricization of (3.2) gives

$$\boldsymbol{Y}_{(1)} = \sum_{j=1}^{p} \boldsymbol{X}_{j(1)} \boldsymbol{B}_j + \boldsymbol{E}_{(1)}, \tag{3.3}$$

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 (3.3) 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 $\hat{\boldsymbol{B}} = \left(\boldsymbol{X}_{(1)}^{\mathsf{T}} \boldsymbol{X}_{(1)}\right)^{-1} \boldsymbol{X}_{(1)}^{\mathsf{T}} \boldsymbol{Y}_{(1)}$ 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 $\mathcal{B}_j$ $(j = 1, \cdots, p)$, we can write

$$\mathcal{B}_j = \mathcal{C}_j \times_1 \boldsymbol{U}_{j1} \times_2 \boldsymbol{U}_{j2} \times_3 \cdots \times_{l_j} \boldsymbol{U}_{jl_j} \times_{l_j+1} \boldsymbol{V}_1 \times_{l_j+2} \cdots \times_{l_j+d} \boldsymbol{V}_d, \tag{3.4}$$

where $\mathcal{C}_j \in \mathbb{R}^{\tilde{P}_{j1} \times \cdots \times \tilde{P}_{jl_j} \times \tilde{Q}_1 \times \cdots \times \tilde{Q}_d}$ $(j = 1, \cdots, p)$ is a core tensor with $\tilde{P}_{ji} \ll P_{ji}$ $(i \leq l_j)$ and $\tilde{Q}_i \ll Q_i$ $(i = 1, \cdots, d)$; $\{\boldsymbol{U}_{ji} : j = 1, \cdots p;\ i = 1, \cdots, l_j\}$ is a set of bases that spans the j^{th} input space; and $\{\boldsymbol{V}_i :\ i = 1, \cdots, d\}$ 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 chapter, we allow $\boldsymbol{U}_{ji}$ to be learned directly from the input spaces. Two important choices of U_{ji} are truncated identity matrices (i.e., no transformation on the inputs) or the bases obtained from the Tucker decomposition of the input tensor $\mathcal{X}_j$, i.e.,

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

In a special case that an input tensor is a matrix, the bases are the principal components (PCs) of that input if one uses Tucker decomposition. Allowing $\boldsymbol{U}_{ji}$ to be determined based on only $\mathcal{X}_j$ is reasonable because learning the core tensors $\mathcal{C}_j$ $(j = 1, \cdots, p)$ and the basis matrices $\boldsymbol{V}_i$ $(i = 1, \cdots, d)$ provides a sufficient degree of freedom to learn $\mathcal{B}_j$. That is, for a given set of basis matrices $\left\{\boldsymbol{U}_{j1}, \cdots, \boldsymbol{U}_{jl_j}\right\}$, we can learn $\mathcal{C}_j$ $(j = 1, \cdots, p)$, so that $\mathcal{C}_j \times_1 \boldsymbol{U}_{j1} \times_2 \boldsymbol{U}_{j2} \times_3 \cdots \times_{l_j} \boldsymbol{U}_{jl_j}$, along with the estimate bases $\{\boldsymbol{V}_1, \cdots, \boldsymbol{V}_d\}$ generate a correct estimation of $\mathcal{B}_j$. Note that depending on what set of bases to be used we may require a larger or smaller value of the ranks $\tilde{P}_{ji}$. Furthermore, fixing $\left\{\boldsymbol{U}_{j1}, \cdots, \boldsymbol{U}_{jl_j}\right\}$ 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:

$$\{\mathcal{C}_j, \boldsymbol{V}_1, \cdots, \boldsymbol{V}_d\} = \arg\min \left|\left| \boldsymbol{Y}_{(1)} - \sum_{j=1}^{p} \boldsymbol{X}_{j(1)} \boldsymbol{B}_j \right|\right|_F^2,$$

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

$$\boldsymbol{V}_i^{\mathrm{T}} \boldsymbol{V}_i = I_{\tilde{Q}_i} \ (i = 1, \cdots, d)\,, \tag{3.5}$$

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 $\boldsymbol{V}_i^{\mathrm{T}} \boldsymbol{V}_i = \boldsymbol{I}_{\tilde{Q}i}$ ensures the uniqueness of both the basis matrices and the core tensors if the problem is identifiable. It should be noted that in general, the problem of estimating functional data through a set of functions may not always be identifiable. [76, 77, 75] discuss the identifiability problem in functional and tensor regression. Because the main purpose of this chapter is to estimate and predict the output, we do not discuss the identifiability issue here, as learning any correct set of parameters $\{\boldsymbol{B}_k : k = 1, \cdots, p\}$ will eventually lead to the same estimation of the output.

In order to solve (3.5), we combine the alternating least square (ALS) approach with the block coordinate decent (BCD) method (designated by ALS-BCD). The advantages of ALS algorithms are conceptual simplicity, noise robustness, and computational efficiency [78]. ALS has also shown great promise in the literature for solving tensor decomposition and regression applications with satisfying results [53]. To be able to employ ALS-BCD, we first demonstrate Proposition 3.2:

Proposition 3.2. *Given $\boldsymbol{U}_{ki}$ $(k = 1, \cdots, p;\ i = 1, 2, \cdots, l_j)$, $\boldsymbol{V}_i$ $(i = 1, 2, \cdots, d)$, and $\mathcal{C}_k$, where $k \neq j$ are known, a reshaped form of the core tensor $\mathcal{C}_j$ can be estimated as*

$$\tilde{\mathcal{C}}_j = \mathcal{R}_j \times_1 \left(\boldsymbol{Z}_j^{\mathrm{T}} \boldsymbol{Z}_j\right)^{-1} \boldsymbol{Z}_j^{\mathrm{T}} \times_2 \left(\boldsymbol{V}_1^{\mathrm{T}} \boldsymbol{V}_1\right)^{-1} \boldsymbol{V}_1^{\mathrm{T}} \times_3 \left(\boldsymbol{V}_2^{\mathrm{T}} \boldsymbol{V}_2\right)^{-1} \boldsymbol{V}_2^{\mathrm{T}} \cdots \times_{d+1} \left(\boldsymbol{V}_d^{\mathrm{T}} \boldsymbol{V}_d\right)^{-1} \boldsymbol{V}_d^{\mathrm{T}}, \tag{3.6}$$

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 (3.4), 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 [79]) to solve the problem. Furthermore, one can efficiently estimate the basis matrices V_i using singular value decomposition according to the following proposition.

Proposition 3.3. *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 3.1 to efficiently calculate the complete matrix using tensor products. Also, unlike the principal component regression (PCR) in which the principal components of the output are learned independent of the inputs, the estimated basis matrices $\boldsymbol{V}_i$ $(i = 1, \cdots, d)$ directly depend on the input tensors, ensuring correlation between the basis matrices and inputs. By combining Propositions 3.2 and 3.3, Algorithm 2 summarizes the estimation procedure for multiple tensor-on-tensor regression. This algorithm, in fact, combines the block coordinate decent (BCD) algorithm

with the ALS algorithm.

Algorithm 2 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: Initilize V_i for all i 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 3.2
7: Estimate V_i for all $i = 1 : d$ using Proposition 3.3
8: Compute B_j for all j and set $w_k = \left|\left|Y_{(1)} - \sum_{j=1}^{p} X_{j(1)} B_j\right|\right|_F^2$
9: **while** $|w_{k+1} - w_k| > \epsilon$

3.3.1 Selection of tuning parameters

The proposed approach requires the selection of values $\tilde{P}_{ji}$ $(j = 1, \cdots, p; i = 1, \cdots, l_j)$ and $\tilde{Q}_k$ $(k = 1, \cdots, d)$. For this purpose, we use the k-fold cross-validation method on a grid of parameters and find the tuple of parameters that minimizes the mean squared error. In order 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 a very accurate solutions. In order to initiate the ranks for the GA, we use the following heuristic: The rank of the i^{th} 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, i.e., $\boldsymbol{A}_{(i)}$. For all studies in the next sections, we perform three-fold cross-validation (CV).

3.4 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 Partial Least Square Regression (*PLS*) and function-on-function regression approach proposed by [39], designated as *sigComp*. The reason we compare our approach to sigComp is that sigComp can handle multiple func-

tional inputs (curves) and learn the basis functions similar to our approach. In the second part, we conduct a set of 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 two benchmarks: The first benchmark is the TOT approach proposed by [75], which can roughly be viewed as a general form of sigComp. Because this approach can only handle a single input tensor, when multiple inputs exist we perform a transformation to merge the inputs into one single tensor. In order to implement TOT, we use R package *MultiwayRegression* provided by [75]. The second benchmark is based on principal component regression (PCR) similar to a benchmark considered in [36]. 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 v 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 $\boldsymbol{X} = \left[\boldsymbol{X}_{1(1)}, \boldsymbol{X}_{2(1)}, \cdots, \boldsymbol{X}_{p(1)}\right]$ 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 chapter, instead of CV over G_x and G_y directly, we perform CV over the percentage of variation the PCs explain, i.e., 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 $SMSPE = \frac{||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2}{||\mathcal{Y}||_F^2}$.

3.4.1 Simulation studies for curve-on-curve regression

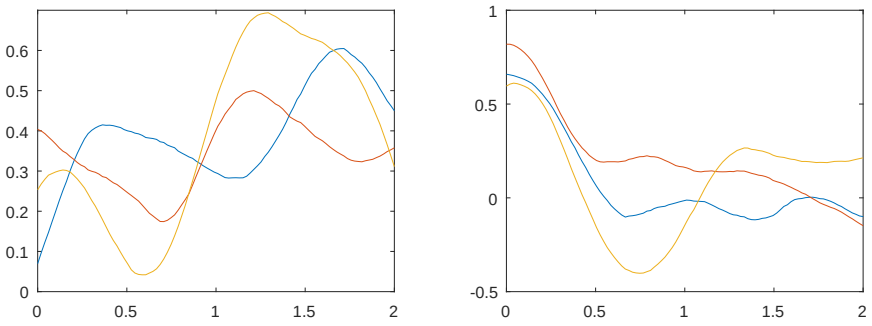
In this simulation, we consider multiple functional (curve) predictors and multiple scalar predictors similar to the simulation study in [39]. We first randomly generate $(\boldsymbol{B}_1, \cdots, \boldsymbol{B}_p)$ as follows:

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

where $\gamma_{ki}(t)$ and $\psi_{ki}(s)$ $(k = 1,\ 2,\ 3;\ i = 1, \cdots, p)$ are Gaussian processes with covariance function $\Sigma_1(z, z') = \left(1 + 20\,|z - z'| + \frac{1}{3}\left(20\,|z - z'|\right)^2\right) e^{-20|z-z'|}$. 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 $\boldsymbol{S} = \boldsymbol{\Delta}\boldsymbol{\Delta}^{\mathsf{T}}$, where $\boldsymbol{\Delta}$ is a $p \times p$ matrix and generate a set of curves $w_1,\ w_2, \cdots,\ w_p$ using a Gaussian process with covariance function $\Sigma_2(z, z') = e^{-(2|z-z'|)^2}$. Finally, we generate the predictors at any given point s as

$$(x_1(s), \cdots, x_p(s)) = (w_1(s), w_2(s), \cdots, w_p(s))\,\boldsymbol{\Delta}^{\mathsf{T}}.$$

With this formulation, each curve of $x_1(s), \cdots, x_p(s)$ is a Gaussian process with covariance function $\Sigma_2(s, s')$, and for each s, the vector $(x_1(s), \cdots, x_p(s))$ is a multivariate normal distribution with covariance $\boldsymbol{S}$. When $\rho_c = 0$, this vector becomes an independent vector of normally distributed variables. Figure 3.2 illustrates examples of the predictors when $p = 3$ for $\rho_c = 0,\ 0.75$. We also generate the scalar predictors $(u_1, \cdots, u_5)$ 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 $\alpha_i(t)$ $(i = 1, \cdots, 5)$ are generated from a Gaussian process



(a) (b)

Figure 3.2: Example of the predictors when (a) $p = 3$, $\rho_c = 0$ and (b) $p = 3$,$\rho_c = 0.75$.

with covariance function $\Sigma(t, t') = e^{\{-5|t-t'|\}^2}$. 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.

For each combination of (p, ρ_c), we compare the performance of the proposed method with the method in [39], designated by sigComp, and partial least square (PLS) regression based on the mean square prediction error (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 [39]. 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 $M_{train} = 400$ that is used to learn the model parameters. Next, we generate a testing sample of size $M_{test} = 100$ and calculate MSPE as

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

Table 3.1: Comparison between the proposed method with PLS and the sigComp method proposed by [39].

		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	0.173 (0.007)	0.076 (0.006)	0.147 (0.005)	0.046 (0.003)	**0.134** (0.003)	**0.035** (0.003)
	0.75	0.165 (0.006)	0.069 (0.005)	0.131 (0.004)	0.032 (0.002)	**0.128**(0.003)	**0.029** (0.002)

and

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

We repeat this procedure 50 times to find the means and standard deviations of the MSPE and MSEE for each method. Table 3.1 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 perform better when the inputs are correlated. Figure 3.3 illustrates prediction examples obtained by each method, along with the true curve for different $p = 3$. As illustrated all of the approaches produce reasonably accurate predictions.

3.4.2 Simulation studies for image and structured point-cloud

We first 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 $\mathcal{X}_{2i} \in \mathbb{R}^{P_{21} \times P_{22} \times \cdots \times P_{2l_2}}$ $(i = 1, \cdots M)$, where M is the number of samples. To generate the input tensors, we define $x_{kmj} = \frac{j}{P_{km}}$ $(k = 1, 2;\ m = 1, \cdots l_k;\ j = 1, \cdots P_{km})$. Then,

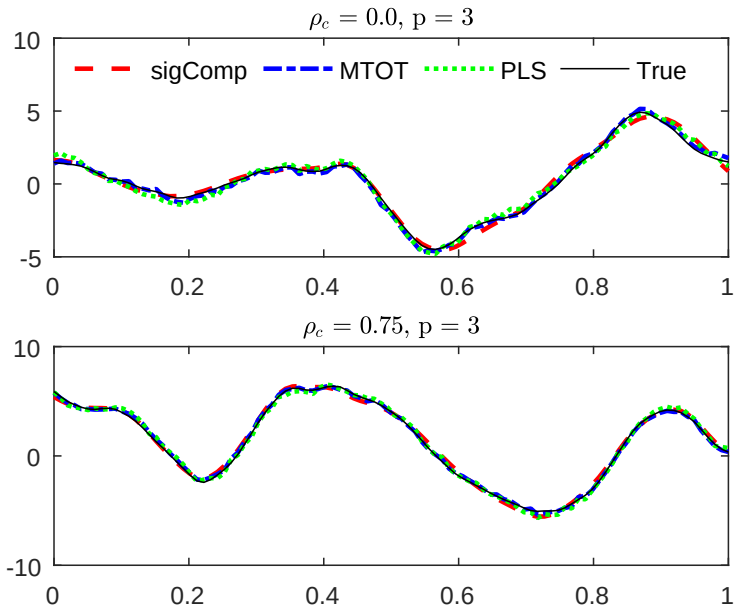


Figure 3.3: Prediction examples of sigComp, PLS, and MTOT.

we set $\boldsymbol{U}_{km} = [\boldsymbol{u}_{km1}, \boldsymbol{u}_{km2}, \cdots \boldsymbol{u}_{kmR_k}]$ $(k = 1, 2;\ m = 1, \cdots l_k)$, where

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

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:

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

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 $\boldsymbol{V}_m = [\boldsymbol{v}_{m1}, \boldsymbol{v}_{m2}, \cdots \boldsymbol{v}_{mR}]$ $(m = 1, \cdots, d)$, where

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

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 $\mathcal{N}(0, \sigma^2)$. 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 $\mathcal{C}_2 \in \mathbb{R}^{3\times 3\times 3}$. Figure 3.4a 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 perform CV and train the model using the training set, then calculate the SMSPE for the proposed method and benchmarks based on the test data. We repeat this procedure 50 times to capture the variance of the SMSPE. In order to prepare data for the TOT approach, 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. We designate this simulation as Case I.

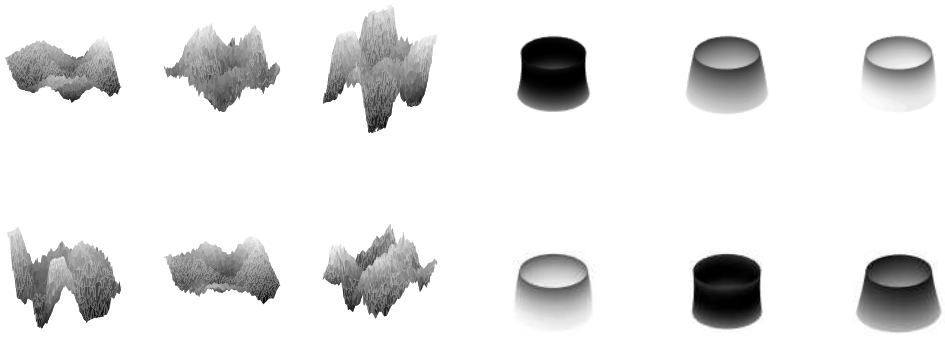
As another simulation case (designated by Case II), we simulate a truncated cone based on a set of scalars and simple profile data in a 3D cylindrical coordinate system (r, ϕ, z), where $\phi \in [0, 2\pi]$ and $z \in [0, 1]$. We first generate an equidistant grid of $I_1 \times I_2$ over the (ϕ, z) space by setting $\phi_i = \frac{2\pi i}{I_1}$ $(i = 1, \cdots, I_1)$ and $z_j = \frac{j}{I_2}$ $(j = 1, \cdots, I_2)$. Specifically,

we set $I_1 = I_2 = 200$. Next, we simulate the truncated cone over the grid by

$$r(\phi, z) = \frac{r_0 + z\tan\theta}{\sqrt{1 - e^2\cos^2\phi}} + c\left(z^2 - z\right) + \epsilon(\phi, z), \tag{3.7}$$

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 $\mathcal{N}(0, \sigma^2)$. Figure 3.4b 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 $x_{4i}(z) = c(z^2 - z)$; $i = 1, \cdots, M$. 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 $X_i \in \mathbb{R}^{81\times 200}$ $(i = 2, 3, 4)$. 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. In order to prepare the data for TOT, 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.

In each case, we compare the proposed method with benchmarks based on the SMSPE calculated at different levels of noise σ. Tables 3.2 and 3.3 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 3.3, 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 seconds to reach the SMSPE of 0.0170, obtained in 1.05 seconds by MTOT. The performance of both PCR 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.



(a) (b)

Figure 3.4: Examples of generated output for the simulation study (a) case I-waveform surface and (b) case II–truncated cone.

Table 3.2: Comparison between the proposed method (MTOT) and the benchmarks in case I with the waveform response. The TOT requires a much larger running time to achieve the same level of prediction error as the MTOT.

	PCR		*TOT*		*MTOT*	
σ	SMSPE	Time (sec)	SMSPE	Time (sec)	SMSPE	Time (sec)
0.1	0.0057 (0.0015)	0.03 (0.00)	0.0046 (0.0011)	154.98 (17.94)	**0.0044** (0.0011)	1.05 (0.03)
0.2	0.0199 (0.0045)	0.04 (0.00)	**0.0170** (0.0039)	147.33 (2.47)	**0.0170** (0.0040)	1.05 (0.01)
0.3	0.0455 (0.0097)	0.04 (0.00)	0.0399 (0.0086)	149.03 (1.36)	**0.0395** (0.0086)	1.05 (0.02)
0.4	0.0773 (0.0233)	0.04 (0.00)	0.0678 (0.0135)	149.13 (0.96)	**0.0673** (0.0212)	1.05 (0.03)
0.5	0.1186 (0.0222)	0.04 (0.00)	0.1036 (0.0231)	146.17 (0.95)	**0.1032** (0.0203)	1.04 (0.01)
0.6	0.1670 (0.0327)	0.04 (0.00)	0.1456 (0.0309)	147.75 (1.81)	**0.1454** (0.0299)	1.03 (0.01)

Table 3.3: Comparison between the proposed method and the benchmarks in case II with truncated cone. Due to the difference between the input and the output rank, the performance of the 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.

	PCR		*TOT*		*MTOT*	
σ	log(SMSPE)	Time (sec)	log(SMSPE)	Time (sec)	log(SMSPE)	Time (sec)
0.01	-5.555 (0.986)	0.05 (0.00)	-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)	-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)	-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)	-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)	-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)	-4.875 (0.956)	37.43 (15.19)	**-6.266** (0.402)	3.95 (0.14)

3.5 Case Study

In semiconductor manufacturing, patterns are printed layer by layer over a wafer in a sequence of deposition, etching, and lithographic processes to manufacture transistors [80].

Many of these processes induce stress variations across the wafer, distorting/changing the wafer shape [81, 64]. Figure 3.5 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 [81]:

$$\begin{cases} \Delta x = T_x - \theta_x y + M_x x & \text{error in x coordinate} \\ \Delta y = T_y + \theta_y x + M_y y & \text{error in y coordinate,} \end{cases} \tag{3.8}$$

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 [81]. 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 [81]. Both the wafer shapes and the overlay errors (in each coordinate, x or y) can be pre-

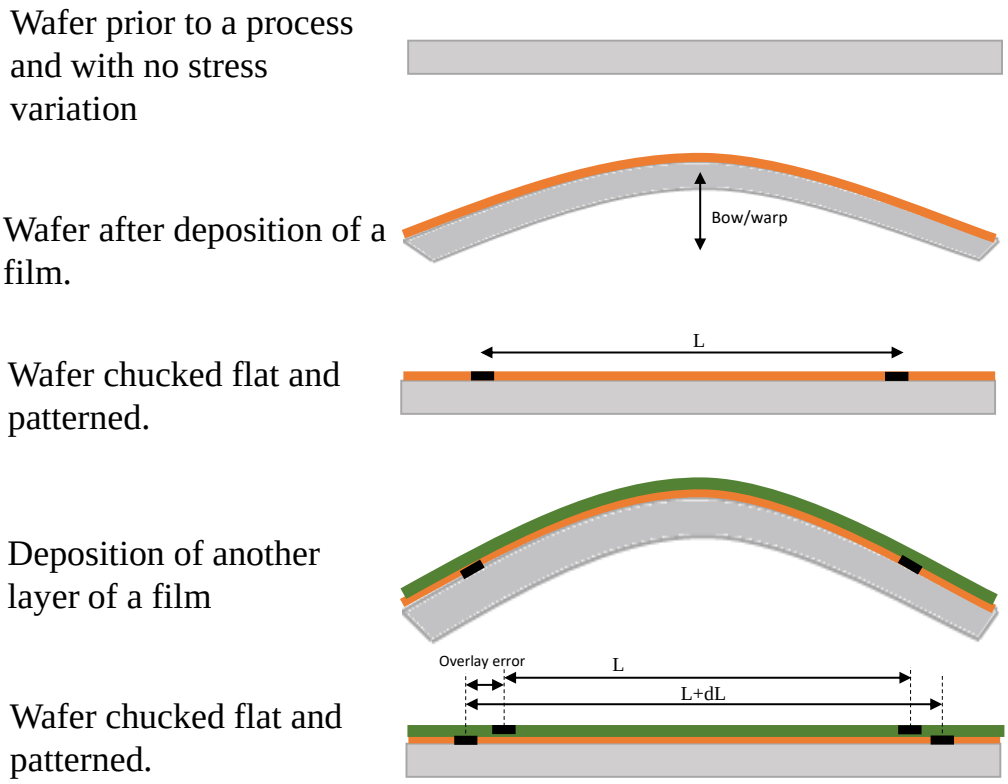


Figure 3.5: Process of a wafer, which causes shape variation and consequently overlay error.

sented 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 [81] to generate surrogate data of overlay errors ($PIR(x,y)$) based on the wafer shape prior to two lithography steps ($w_1(x,y)$, $w_2(x,y)$). 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, i.e., wafer shapes and overlay errors, $\{(w_{1i}(x,y), w_{2i}(x,y), PIR_i)\}_{i=1}^{M=500}$, and employ our proposed method to estimate the PIR_i based on $(w_{1i}(x,y), w_{2i}(x,y))$. Because in our simulated data $w_{1i}(x,y)$ remains fixed, we consider $w_i(x,y) = w_{2i}(x,y) - w_{1i}(x,y)$ as the predictor. We also generate 100 observations as the test dataset. The mean square prediction error 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 [75] failed to run with this size of images due to its high space complexity. Therefore, we only compared our approach with PCR. Figure 3.6 illustrates an example of the original and predicted corrected overlay error image, along with the prediction error. As illustrated, the proposed method predicted the original surface more accurately, with smaller errors across the wafer. Figure 3.7 illustrates the boxplots of the logarithm of the prediction mean square error calculated over the 50 replications in contrast with the benchmark. The results show that the proposed method is superior to the benchmark in prediction of the image. As an example, the average of log(SMSE) over the replications is -8.33 for the proposed method and -7.56 for the PCR approach.

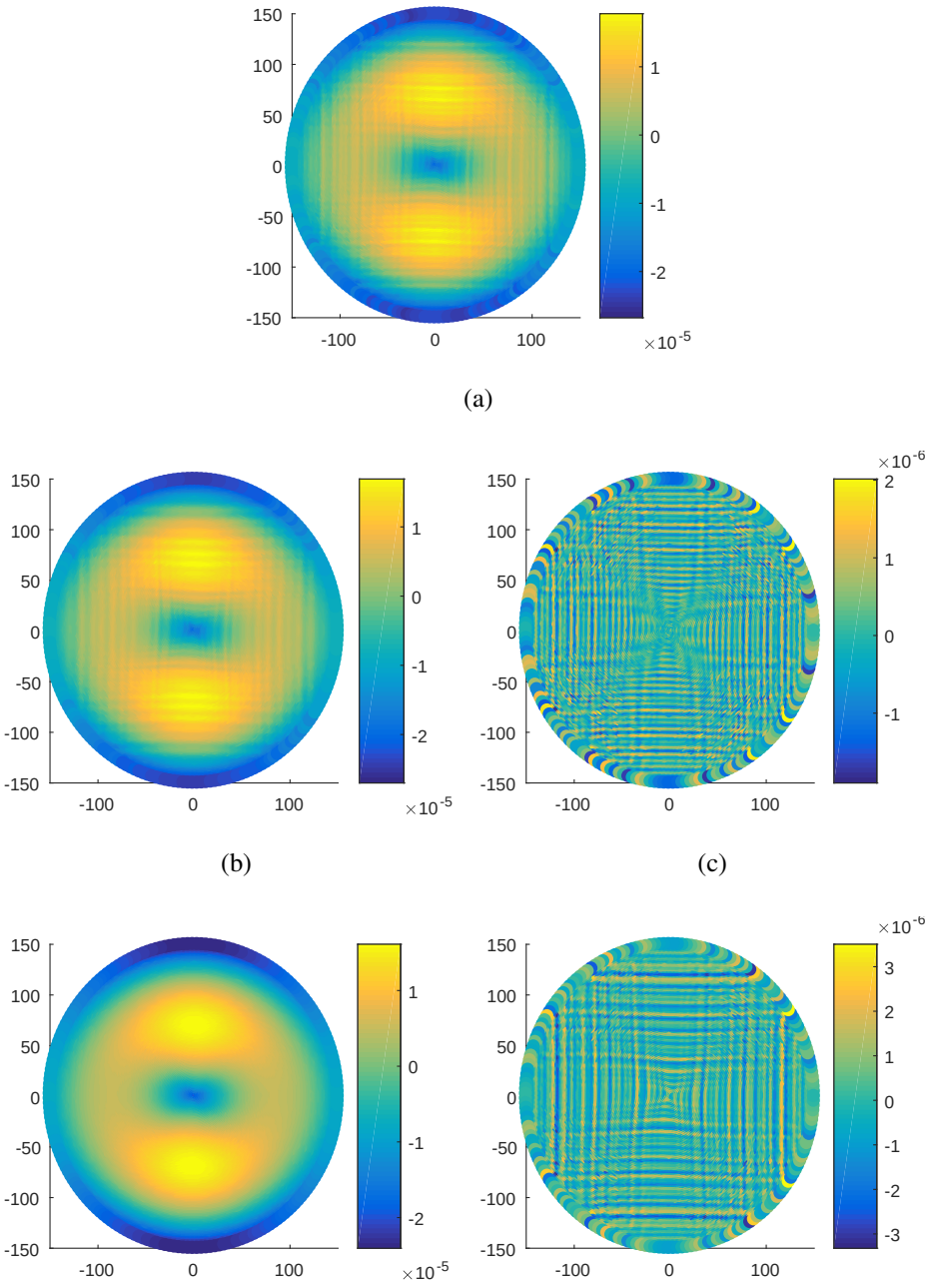


Figure 3.6: Example of (a) the x coordinate overlay error, (b) prediction of MTOT, (c) MTOT prediction error, (d) prediction of PCR, and (e) PCR prediction error.

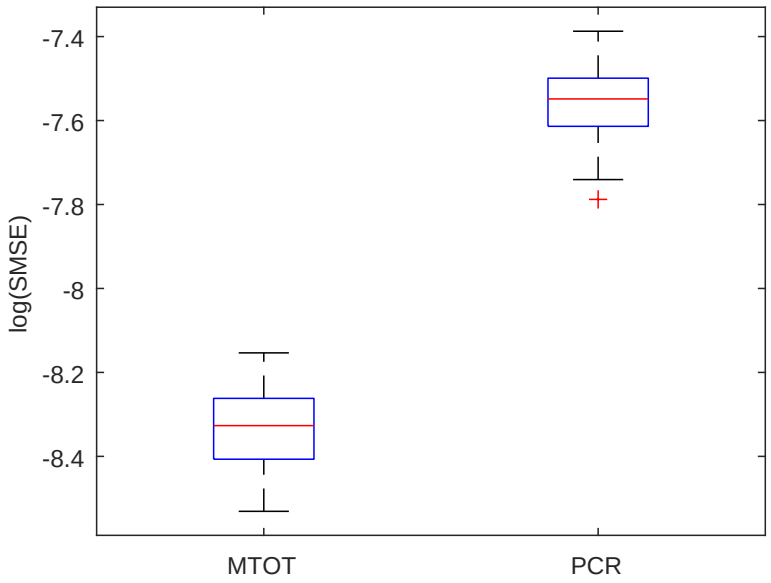


Figure 3.7: Logarithm of the prediction mean square error calculated for the test data over 50 replications. The proposed method illustrates significantly lower standard prediction error than the PCR approach.

3.6 Conclusion

This chapter proposed a multiple tensor-on-tensor 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. In order to estimate the parameters, a least square loss function is defined. In order to avoid overfitting, 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 decent (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.

In order 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 [39]. 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 principal component regression (PCR) and tensor-on-tensor (TOT) regression, and showed superior performance in terms of mean squared prediction error. 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.

Appendices

APPENDIX A

ROUGHNESS PENALTY AND PROOF OF PROPOSITION 2.1

The matrices $P_{j,s}$ and $P_{j,t}$ are the roughness penalties over the functional parameters, $\beta_j(s,t)$. Let L_t (L_s) denotes an differentiating operator (e.g. second derivative) with respect to t (s). Then, we define the roughness penalty over $\beta_j(s,t)$ as follows,

$$\begin{aligned} pen_t(\beta_j) &= \int\int \left[L_t\beta_j^{\mathsf{T}}(s,t)\right]^2 dsdt \\ &= \int\int \left[L_t\left[\eta^{\mathsf{T}}(t)B_j^{\mathsf{T}}\theta_j(s)\right]\right]^2 dsdt \\ &= \int\int \left[L_t\eta^{\mathsf{T}}(t)B_j^{\mathsf{T}}\theta_j(s)\right]^{\mathsf{T}}\left[L_t\eta^{\mathsf{T}}(t)B_j^{\mathsf{T}}\theta_j(s)\right] \\ &= \int\int b_j^{\mathsf{T}}\left[\theta_j(s)\otimes L_t\eta_j(t)\right]\left[\theta_j^{\mathsf{T}}(s)\otimes L_t\eta_j^{\mathsf{T}}(t)\right]b_j \\ &= b^{\mathsf{T}}\left(\left[\int\theta_j(s)\theta_j^{\mathsf{T}}(s)\right]\otimes\left[\int L_t\eta_j(t)L_t\eta_j^{\mathsf{T}}(t)\right]\right)b \end{aligned}$$

where the last equality follows from the property of the Kronecker product that

$$(A\otimes B)(C\otimes D) = AC\otimes BD$$

for matrices A, B, C, D. Moreover, we notice that $\eta_j(t) = \eta_{j1}(t_1)\otimes\eta_{j2}(t_2)\otimes\cdots\otimes\eta_{jd}(t_d)$. Then we can write

$$pen_t(\beta_j) = b_j^{\mathsf{T}}\left(P_\theta\otimes P_{L\eta_1}\otimes\cdots\otimes P_{L\eta_d}\right)b_j,$$

where, $P_\theta = \left[\int\theta_j(s)\theta_j^{\mathsf{T}}(s)\right]$, and $P_{L\eta_k} = \left[\int L_t\eta_{j1}(t_1)L_t\eta_{j1}^{\mathsf{T}}(t_1)\right];\ k = 1,\cdots,d$. Similarly, if we use the operator L_s for the smoothness in the s, we can show that

$$pen_s(\beta_j) = b_j^{\mathsf{T}}\left(P_{L\theta}\otimes P_{\eta_1}\otimes\cdots\otimes P_{\eta_d}\right)b_j,$$

where $P_{L\theta=}\left[\int L_s\theta_j(s)L_s\theta_j^{\mathsf{T}}(s)\right]$ and $P_{\eta_k} = \left[\int\eta_{jd}(t_d)\eta_{jd}^{\mathsf{T}}(t_d)\right];\ k = 1,\cdots,d$

APPENDIX B

PROOF OF PROPOSITION 2.2

Proof I: In this appendix, we derive the (2.6) as a solution to the loss function

$$\begin{aligned} L\left(b_1, \cdots, b_p\right) = & \frac{1}{2}\left|\left|y - \sum_{j=1}^{p} D_j b_j\right|\right|_2^2 \\ & + \frac{\lambda}{2}\sum_{j=1}^{p} b_j^{\mathrm{T}}\left(P_{j,s} + P_{j,t}\right) b_j \\ & + \gamma \sum_{j=1}^{p} \sqrt{q_j b_j^{\mathrm{T}} W_j b_j} \end{aligned}$$

where, $W_j = D_j^{\mathrm{T}} D_j + \lambda\left(P_{j,s} + P_{j,t}\right)$. Note that W_j is positive semi–definite and so Cholesky decomposition of W_k exists, i.e., $W_j = V_j^{\mathrm{T}} V_j$. We now calculate the sub–derivative of L with respect to b_k, which gives,

$$\begin{aligned} \frac{\partial L}{\partial b_k} = & -D_k^{\mathrm{T}}\left(y - \sum_{j=1, j\neq k}^{p} D_j b_j\right) \\ & + W_k b_k + \gamma\sqrt{q_k} V_j^{\mathrm{T}} g\left(b_k\right), \end{aligned}$$

where $g\left(b_k\right) = \frac{V_k b_k}{||V_k b_k||_2}$ if $b_k \neq 0$ and is a vector with $||g\left(b_k\right)|| \leq 1$ if $b_k = 0$. For b_k to be optimal, the sub–differential $\frac{\partial L}{\partial b_k}$ should include zero (i.e. $0 \in \frac{\partial L}{\partial b_k}$) and so we can find optimal b_k by setting $\frac{\partial L}{\partial b_k} = 0$. Let us set $r_k = y - \sum_{j=1, j\neq k}^{p} D_j b_j$ to denote the residuals when the k^{th} covariate is omitted. Then, we have: $\frac{\partial L}{\partial b_k} = -D_k^{\mathrm{T}} r_k + W_k b_k + \gamma\sqrt{q_k} V_j g\left(b_k\right) = 0$. Now if $\left|\left|V_k^{-1} D_k^{\mathrm{T}} r_k - V_k^{-1} W_k b_k\right|\right|_2 < \gamma\sqrt{q_k}$ then $b_k = 0$, otherwise,

$$b_k = \left(1 + \frac{\gamma\sqrt{q_k}}{||V_k b_k||_2}\right)^{-1} W_k^{-1} D_k r_k$$

With some algebra to find $||V_k b_k||$,

$$b_k = \left(1 - \frac{\gamma\sqrt{q_t}}{V_k^{-1} D_k^{\mathtt{T}} r_k}\right) W_k^{-1} D_k r_k$$

Finally, we set $b_k = 0$ when $1 - \frac{\gamma\sqrt{q_t}}{V_k^{-1} D_k^{\mathtt{T}} r_k} < 0$, or we write,

$$b_k = \left(1 - \frac{\gamma\sqrt{q_t}}{V_k^{-1} D_k^{\mathtt{T}} r_k}\right)_+ W_k^{-1} D_k r_k$$

where, $(x)_+ = max\,(0, x)$.

Sketch of an alternative proof using a transformation: Consider the following loss function,

$$L = \frac{1}{2}\left|\left|\boldsymbol{y} - \sum_{j=1}^{p} D_j \boldsymbol{b}_j\right|\right|_2^2 + \frac{\lambda}{2}\sum_{j=1}^{p} \boldsymbol{b}_j^{\mathtt{T}} P_j \boldsymbol{b}_j + \gamma \sum_{j=1}^{p} \sqrt{q_j \boldsymbol{b}_j^{\mathtt{T}} W_j \boldsymbol{b}_j}$$

with $W_j = D_j^{\mathtt{T}} D_j + \lambda P_j$. Let us denote $H_j = \left[D_j\, 0 \cdots \sqrt{\lambda} P_j^{\frac{1}{2}}\, 0 \cdots 0\right]^{\mathtt{T}}$ and $\boldsymbol{z} = [\boldsymbol{y}\, 0 \cdots 0]^{\mathtt{T}}$. Then, we can write,

$$L = \frac{1}{2}\left|\left|\boldsymbol{z} - \sum_{j=1}^{p} H_j \boldsymbol{b}_j\right|\right|_2^2 + \gamma \sum_{j=1}^{p} \sqrt{q_j}\, ||H_j \boldsymbol{b}_j||_2 .$$

Now, consider the QR decomposition of $H_j = U_j Q_j$, where U_j is unitary matrix (i.e. $U^{\mathtt{T}} U = I$) and Q_j is an upper triangle square matrix. With the QR decomposition, the loss function can be written as

$$L = \frac{1}{2}\left|\left|\boldsymbol{z} - \sum_{j=1}^{p} U_j \boldsymbol{c}_j\right|\right| + \gamma \sum_{j=1}^{p} \sqrt{q_j}\, ||\boldsymbol{c}_j||_2$$

where $\boldsymbol{c}_j = Q_j \boldsymbol{b}_j$. The above is a formulation of the loss function with an orthonormal design matrix and has a closed-form soft-thresholding solution. Obtaining the closed-form solution of this new loss function is well-established in the literature [55].

APPENDIX C

PROOF OF PROPOSITION 3.2

For simplicity, we assume only one input tensor exists. Then, we can solve $\mathcal{C}$ by

$$\begin{aligned}\text{argmin}_{\mathcal{C}}\|Y_{(1)} - X_{(1)}B\|_F^2 &= \|Y_{(1)} - X_{(1)}(U_l \otimes U_{l-1} \otimes \cdots \otimes U_1)C(V_d \otimes \cdots \otimes V_1)^T\|_F^2 \\ &= \|vec(Y_{(1)}) - vec(ZC(V_d \otimes \cdots \otimes V_1)^T)\|_2^2 \\ &= \|vec(Y_{(1)}) - (V_d \otimes \cdots \otimes V_1 \otimes Z)vec(C)\|_2^2,\end{aligned}$$

where $vec\,(X)$ stacks the columns of matrix X on top of each other. This is a simple least square regression that gives a closed-form solution as in (3.6) after applying Proposition 3.1 to convert Kronecker products to tensor products.

APPENDIX D

PROOF OF PROPOSITION 3.3

Again assuming a single input tensor, let us define $\tilde{Y} = (V_d \otimes \cdots \otimes V_1 \otimes Z)vec(C)$ in the tensor format as $\tilde{\mathcal{Y}} = \tilde{C} \times_1 Z \times_2 V_1 \times \cdots \times_{d+1} V_d$, which can be written as $\tilde{Y}_{(i)} = V_i \tilde{C}_{(i)}(V_d \otimes \cdots V_{i+1} \otimes V_{i-1} \otimes V_1 \otimes Z)^{\mathtt{T}}$. First note that

$$\begin{aligned}
\text{argmin}_{V_i} \|Y_{(1)} - X_{(1)}B\|_F^2 &= \text{argmin}_{V_i} \|vec(Y_{(1)}) - (V_d \otimes \cdots \otimes V_1 \otimes Z)vec(\tilde{C})\|_F^2 \\
&= \text{argmin}_{V_i} \|Y_{(i)} - \tilde{Y}_{(i)}\|_F^2 \\
&= \text{argmin}_{V_i} \|Y_{(i)} - V_i \tilde{C}_{(i)}(V_d \otimes \cdots V_{i+1} \otimes V_{i-1} \otimes V_1 \otimes Z)^{\mathtt{T}}\|_F^2,
\end{aligned}$$

with $A := \tilde{C}_{(i)}(V_d \otimes \cdots V_{i+1} \otimes V_{i-1} \otimes V_1 \otimes Z)^{\mathtt{T}}$. Then, we want to solve

$$\text{argmin}_{V_i} \|Y_{(i)} - V_i A\|_F^2 \, s.t. \, V_i^{\mathtt{T}} V_i = I.$$

This is an orthogonal procrustes problem and is known to have solution as is stated in Proposition 3.3.

APPENDIX E

SIMULATING THE OVERLAY ERROR

[81] introduced a measure based on in-plane distortion (IPD) called predicted in-plane distortion residual (PIR) to estimate and predict nonuniform-stress-induced overlay errors based on wafer shape. For this purpose, they first illustrate that the IPD is proportional to gradient of wafer shape $w(x, y)$, i.e.,

$$IPD \propto -\nabla w.$$

Then, for two layers, say i and k, to be patterned, they calculate the IPD and subtract them to find the shape-slope difference, i.e., $SSD = IPD_i - IPD_k$. The shape-slope is then corrected based on model (3.8) to find the shape-slope residual (SSR). Finally, [81] calculated the PIR as a factor of SSR. That is,

$$PIR = c \times SSR,$$

where c is a constant that depends on the wafer thickness. In their study, they showed through four differently patterned engineer stress monitor (ESM) wafers that the PIR is linearly correlated by the overlay errors with high R^2 values (e.g., 92%). To perform the experiment, they first deposit a layer of silicon nitride film over a 300mm wafer as a source of stress. This process changes the wafer shape and causes the wafer to curve. The shape of the wafer is measured by a patterned wafer geometry (PWG) tool designed for the metrology of 300mm wafers. After the wafer is exposed by a designed pattern (four different patterns considered in this study), it goes through an etching process that relieves some part of the stress depending on the pattern density. At this stage, and prior to next lithography step, the wafer shape is again measured. After the second lithography step, the overlay

error is measured using standard optical methods. Using the measured wafer shapes, they calculated the PIR and showed a high correlation between the PIR and overlay error.

In our study, we first simulate the wafer shapes and then estimate the overlay errors using the following procedure introduced by [81]. Simulating a wafer shape requires knowledge of the different components in wafer geometry. [81] and [64] consider several wafer shape features that span different ranges of spatial wavelength (λ). At the highest level is the overall shape of the wafer represented by a bow (or warp) in the range of tens of micrometers. Other shape variations are those that are spanned by spatial wavelengths in the range of several meters and waveheight in the micrometer range. Another component is the nanotopography (NT) of the wafer, with λ ranging from few millimeters to 20mm and the wave-height in nanometers. Finally, the roughness of the wafer is defined as variations with $\lambda < 0.2$mm. In this study, we only consider the bow shape and the NT components when simulating a wafer shape. The wafer shapes are simulated as follows: We first assume that a thin layer is deposited over a wafer, which causes only a bow shape geometry in the wafer (that is, we assume no wave patterns). We simulate the bowed wafer geometry using $w_1(x,y) = \frac{b_1\left(0.5x^2+y^2\right)}{R^2}$, where b_1 is the warp or bow size and is assumed to be 100μm, and R is the wafer radius, which is assumed to be 150mm. We then assume that a lithography/etching process is performed and the wafer shape changes in both bow and wavy patterns as follows:

$$w_2(x,y) = \frac{b_2\left(0.5x^2+y^2\right)}{R^2} + \sum_{i=1}^{p} \frac{h_i}{2}\left(1+sin\left(\frac{2\pi x}{\lambda_i}\right)\right) + \sum_{i=1}^{p} \frac{h_i}{2}\left(1+cos\left(\frac{2\pi y}{\lambda_i}\right)\right),$$

where b_2 is the bow size uniformly sampled from 30 to $100\mu m$, and h_i and λ_i are the waveheight and wavelength, respectively. Moreover, p is the number of waveforms assumed. For each wafer (i.e. sample), we first randomly select p from $U(2,10)$ and then select p wavelength from $U(2,20)$ for NT wavelength. Finally, we sample waveheight h_i from $U\left(\frac{\lambda_i}{10^7}, \frac{\lambda_i}{10^6}\right)$ to ensure that the large wavelength has large a waveheight and vice versa. Af-

ter simulating a wafer shape prior to two lithography steps, we calculate the IPD and PIR according to the procedure described previously. Note that we only calculate the values proportional to the original values. Figure E.1 illustrates an example of generated shapes, their associated IPDs in the x coordinates, i.e., $-\frac{\partial w}{\partial x}$, and the difference between the x coordinate IPDs prior to and after correction. In order to correct the IPD values, we consider a second order model:

$$\begin{cases} \Delta IPD_x = k_0 + k_1 x + k_2 y + k_3 x^2 + k_4 y^2 + k_5 xy & \text{error in x coordinate} \\ \Delta IPD_y = k_6 + k_7 x + k_8 y + k_9 x^2 + k_{10} y^2 + k_{11} xy & \text{error in y coordinate,} \end{cases}$$

which is fitted to the calculated values of $\Delta IPD_x = IPD_{x2} - IPD_{x1}$ and ΔIPD_y. Then the fitted model is subtracted from the ΔIPD_x and ΔIPD_y to find the corrected values. The corrected values are associated with the PIR and the overlay error.

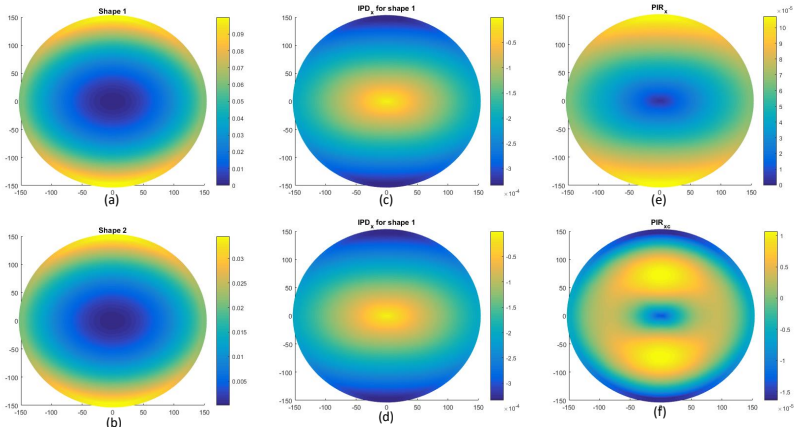


Figure E.1: Illustration of wafer (a) shape prior to first step lithography, (b) shape prior to second step lithography, (c) IPD_x for the first shape, (d) IPD_x for the second shape, (e) PIR prior to correction, and (f) PIR after correction for second order shapes.

REFERENCES

[1] M. C. Kennedy and A. O'Hagan, "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, vol. 87, no. 1, pp. 1–13, 2000.

[2] Z. Qian, C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. J. Wu, "Building surrogate models based on detailed and approximate simulations," *Journal of Mechanical Design*, vol. 128, no. 4, pp. 668–677, 2006.

[3] P. Z. Qian and C. J. Wu, "Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments," *Technometrics*, vol. 50, no. 2, pp. 192–204, 2008.

[4] H. Xia, Y. Ding, and B. K. Mallick, "Bayesian hierarchical model for combining misaligned two-resolution metrology data," *IIE Transactions*, vol. 43, no. 4, pp. 242–258, 2011.

[5] B. M. Colosimo, M. Pacella, and N. Senin, "Multisensor data fusion via gaussian process models for dimensional and geometric verification," *Precision Engineering*, vol. 40, pp. 199–213, 2015.

[6] B. M. Colosimo, G. Moroni, and S. Petrò, "A tolerance interval based criterion for optimizing discrete point sampling strategies," *Precision Engineering*, vol. 34, no. 4, pp. 745–754, 2010.

[7] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[8] C Bradley and V Chan, "A complementary sensor approach to reverse engineering," *Journal of Manufacturing Science and Engineering*, vol. 123, no. 1, pp. 74–82, 2001.

[9] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, vol. 463, 2007, pp. 3251–3269.

[10] L. Le Gratiet and J. Garnier, "Recursive co-kriging model for design of computer experiments with multiple levels of fidelity," *International Journal for Uncertainty Quantification*, vol. 4, no. 5, 2014.

[11] C. S. Reese, A. G. Wilson, M. Hamada, H. F. Martz, and K. J. Ryan, "Integrated analysis of computer and physical experiments," *Technometrics*, vol. 46, no. 2, pp. 153–164, 2004.

[12] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[13] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, 2012.

[14] M. D. Morris and T. J. Mitchell, "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, vol. 43, no. 3, pp. 381–402, 1995.

[15] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. Springer Science & Business Media, 2013.

[16] M. E. Johnson, L. M. Moore, and D. Ylvisaker, "Minimax and maximin distance designs," *Journal of Statistical Planning and Inference*, vol. 26, no. 2, pp. 131–148, 1990.

[17] K.-T. Fang, "The uniform design: Application of number-theoretic methods in experimental design," *Acta Mathematicae Applicatae Sinica*, vol. 3, no. 4, pp. 363–372, 1980.

[18] G. E. Box and N. R. Draper, "A basis for the selection of a response surface design," *Journal of the American Statistical Association*, vol. 54, no. 287, pp. 622–654, 1959.

[19] M. C. Shewry and H. P. Wynn, "Maximum entropy sampling," *Journal of Applied Statistics*, vol. 14, no. 2, pp. 165–170, 1987.

[20] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical Science*, pp. 409–423, 1989.

[21] J. Sacks, S. B. Schiller, and W. J. Welch, "Designs for computer experiments," *Technometrics*, vol. 31, no. 1, pp. 41–47, 1989.

[22] C. Q. Lam, "Sequential adaptive designs in computer experiments for response surface model fit," PhD thesis, The Ohio State University, 2008.

[23] S. Xiong, P. Z. Qian, and C. J. Wu, "Sequential design and analysis of high-accuracy and low-accuracy computer codes," *Technometrics*, vol. 55, no. 1, pp. 37–46, 2013.

[24] A. A. Ezzat, A. Pourhabib, and Y. Ding, "Sequential design for functional calibration of computer models," *Technometrics*, no. just-accepted, 2017.

[25] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT press Cambridge, 2006, vol. 1.

[26] M. Schonlau, "Computer experiments and global optimization," PhD thesis, University of Waterloo, 1997.

[27] F. H. Branin, "Widely convergent method for finding multiple solutions of simultaneous nonlinear equations," *IBM Journal of Research and Development*, vol. 16, no. 5, pp. 504–522, 1972.

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[29] S. J. Ratcliffe, G. Z. Heller, and L. R. Leader, "Functional data analysis with application to periodically stimulated foetal heart rate data. ii: Functional logistic regression," *Statistics in Medicine*, vol. 21, no. 8, pp. 1115–1127, 2002.

[30] P. T. Reiss, L. Huang, and M. Mennes, "Fast function-on-scalar regression with penalized basis expansions," *International Journal of Biostatistics*, vol. 6, no. 1, 2010.

[31] J. O. Ramsay and C. Dalzell, "Some tools for functional data analysis," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 539–572, 1991.

[32] P. C. Besse and H. Cardot, "Approximation spline de la prévision d'un processus fonctionnel autorégressif d'ordre 1," *Canadian Journal of Statistics*, vol. 24, no. 4, pp. 467–487, 1996.

[33] J. Ramsay and B. Silverman, *Functional Data Analysis*. Springer Science & Business Media, 2005.

[34] F. Yao, H.-G. Müller, J.-L. Wang, *et al.*, "Functional linear regression analysis for longitudinal data," *The Annals of Statistics*, vol. 33, no. 6, pp. 2873–2903, 2005.

[35] K. Kim, D. Şentürk, and R. Li, "Recent history functional linear models for sparse longitudinal data," *Journal of Statistical Planning and Inference*, vol. 141, no. 4, pp. 1554–1566, 2011.

[36] Y. Fan, N. Foutz, G. M. James, W. Jank, *et al.*, "Functional response additive model estimation with online virtual stock markets," *The Annals of Applied Statistics*, vol. 8, no. 4, pp. 2435–2460, 2014.

[37] J. Oliva, W. Neiswanger, B. Póczos, E. Xing, H. Trac, S. Ho, and J. Schneider, "Fast function to function regression," in *Artificial Intelligence and Statistics*, 2015, pp. 717–725.

[38] A. E. Ivanescu, A.-M. Staicu, F. Scheipl, and S. Greven, "Penalized function-on-function regression," *Computational Statistics*, vol. 30, no. 2, pp. 539–568, 2015.

[39] R. Luo and X. Qi, "Function-on-function linear regression by signal compression," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 690–705, 2017.

[40] M. Pacella, "Unsupervised classification of multichannel profile data using PCA: An application to an emission control system," *Computers and Industrial Engineering*, vol. 122, pp. 161–169. 2018.

[41] H. Chen and Y. Wang, "A penalized spline approach to functional mixed effects model analysis," *Biometrics*, vol. 67, no. 3, pp. 861–870, 2011.

[42] L. Zhou, J. Z. Huang, and R. J. Carroll, "Joint modelling of paired sparse functional data using principal components," *Biometrika*, vol. 95, no. 3, pp. 601–619, 2008.

[43] H. Zhu and D. D. Cox, "A functional generalized linear model with curve selection in cervical pre-cancer diagnosis using fluorescence spectroscopy," *Lecture Notes-Monograph Series*, pp. 173–189, 2009.

[44] E. R. Lee and B. U. Park, "Sparse estimation in functional linear regression," *Journal of Multivariate Analysis*, vol. 105, no. 1, pp. 1–17, 2012.

[45] J. Goldsmith, L. Huang, and C. M. Crainiceanu, "Smooth scalar-on-image regression via spatial Bayesian variable selection," *Journal of Computational and Graphical Statistics*, vol. 23, no. 1, pp. 46–64, 2014.

[46] G. M. James, J. Wang, and J. Zhu, "Functional linear regression that's interpretable," *The Annals of Statistics*, pp. 2083–2108, 2009.

[47] B. D. Marx and P. H. Eilers, "Generalized linear regression on sampled signals and curves: A p-spline approach," *Technometrics*, vol. 41, no. 1, pp. 1–13, 1999.

[48] J. S. Morris, "Functional regression," *Annual Review of Statistics and its Application*, vol. 2, pp. 321–359, 2015.

[49] J. S. Morris, V. Baladandayuthapani, R. C. Herrick, P. Sanna, and H. Gutstein, "Automated analysis of quantitative image data using isomorphic functional mixed models, with application to proteomics data," *The Annals of Applied Statistics*, vol. 5, no. 2A, p. 894, 2011.

[50] Y. Chen, X. Wang, L. Kong, and H. Zhu, "Local region sparse learning for image-on-scalar regression," *arXiv preprint arXiv:1605.08501*, 2016.

[51] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin, "Instructing people for training gestural interactive systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2012, pp. 1737–1746.

[52] N. Simon and R. Tibshirani, "Standardization and the group lasso penalty," *Statistica Sinica*, vol. 22, no. 3, p. 983, 2012.

[53] T. G. Kolda, "Multilinear operators for higher-order decompositions," Sandia National Laboratories, Tech. Rep., 2006.

[54] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[55] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics New York, 2009.

[56] M. Müller, *Information Retrieval for Music and Motion*. Springer, 2007, vol. 2.

[57] G Szatvanyi, C Duchesne, and G Bartolacci, "Multivariate image analysis of flames for product quality and combustion control in rotary kilns," *Industrial & Engineering Chemistry Research*, vol. 45, no. 13, pp. 4706–4715, 2006.

[58] W. Wójcik and A. Kotyra, "Combustion diagnosis by image processing," *Photonics Letters of Poland*, vol. 1, no. 1, pp. 40–42, 2009.

[59] H. Yu and J. F. MacGregor, "Multivariate image analysis and regression for prediction of coating content and distribution in the production of snack foods," *Chemometrics and Intelligent Laboratory Systems*, vol. 67, no. 2, pp. 125–144, 2003.

[60] E Bellon, J Van Cleynenbreugel, D Delaere, W Houtput, M Smet, G Marchal, and P Suetens, "Experimental teleradiology. novel telematics services using image processing, hypermedia and remote cooperation to improve image-based medical decision making," *Journal of Telemedicine and Telecare*, vol. 1, no. 2, pp. 100–110, 1995.

[61] D. Balageas, C.-P. Fritzen, and A. Güemes, *Structural Health Monitoring*. John Wiley & Sons, 2010, vol. 90.

[62] H. Liang, H. Wu, and R. J. Carroll, "The relationship between virologic and immunologic responses in aids clinical research using mixed-effects varying-coefficient models with measurement error," *Biostatistics*, vol. 4, no. 2, pp. 297–312, 2003.

[63] A. Khosravani, A. Cecen, and S. R. Kalidindi, "Development of high throughput assays for establishing process-structure-property linkages in multiphase polycrystalline metals: Application to dual-phase steels," *Acta Materialia*, vol. 123, pp. 55–69, 2017.

[64] K. T. Turner, R. Ramkhalawon, and J. K. Sinha, "Role of wafer geometry in wafer chucking," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, vol. 12, no. 2, pp. 023 007–023 007, 2013.

[65] J. Sun, S. Papadimitriou, and S. Y. Philip, "Window-based tensor analysis on high-dimensional and multi-aspect streams," in *ICDM*, 2006, pp. 1076–1080.

[66] A. Sapienza, A. Panisson, J. Wu, L. Gauvin, and C. Cattuto, "Detecting anomalies in time-varying networks using tensor decomposition," in *IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015, pp. 516–523.

[67] H. Yan, K. Paynabar, and J. Shi, "Image-based process monitoring using low-rank tensor decomposition," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 216–227, 2015.

[68] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.

[69] H. A. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.

[70] X. Li, H. Zhou, and L. Li, "Tucker tensor regression and neuroimaging analysis," *arXiv preprint arXiv:1304.5637*, 2013.

[71] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in Measuring Change*, vol. 122137, 1963.

[72] H. Yan, K. Paynabar, and M. Pacella, "Structured point cloud data analysis for process modeling and optimization," *Technometrics*, vol. submitted, 2017.

[73] R. Bro, "Multiway calibration. multilinear PLS," *Journal of Chemometrics*, vol. 10, no. 1, pp. 47–61, 1996.

[74] Q. Zhao, C. F. Caiafa, D. P. Mandic, Z. C. Chao, Y. Nagasaka, N. Fujii, L. Zhang, and A. Cichocki, "Higher order partial least squares (HOPLS): A generalized multilinear regression method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1660–1673, 2013.

[75] E. F. Lock, "Tensor-on-tensor regression," *arXiv preprint arXiv:1701.01037*, 2017.

[76] G He, H. Müller, and J. Wang, "Extending correlation and regression from multivariate to functional data," *Asymptotics in Statistics and Probability*, pp. 197–210, 2000.

[77] J.-M. Chiou, H.-G. Müller, and J.-L. Wang, "Functional response models," *Statistica Sinica*, pp. 675–693, 2004.

[78] V. Sharan and G. Valiant, "Orthogonalized ALS: A theoretically principled tensor decomposition algorithm for practical use," *arXiv preprint arXiv:1703.01804*, 2017.

[79] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[80] Y. Nishi and R. Doering, *Handbook of Semiconductor Manufacturing Technology*. CRC Press, 2000.

[81] T. A. Brunner, V. C. Menon, C. W. Wong, O. Gluschenkov, M. P. Belyansky, N. M. Felix, C. P. Ausschnitt, P. Vukkadala, S. Veeraraghavan, and J. K. Sinha, "Characterization of wafer geometry and overlay error on silicon wafers with nonuniform stress," *Journal of Micro/Nanolithography, MEMS, and MOEMS*, vol. 12, no. 4, pp. 043 002–043 002, 2013.

VITA

Mostafa Reisi Gahrooei completed a M.S. degree in computational science and engineering at Georgia Tech, and received M.Sc. degrees in transportation engineering and applied mathematics both from Southern Illinois University Edwardsville. His research interests focus on developing efficient data analytics methodologies for modeling and monitoring complex systems with high-dimensional, heterogeneous data for the purpose of intelligent decision-making to improve system performance. Specifically, he is interested in incorporating tensor and network analysis in modeling, assessing, and improving data-rich systems such as those in manufacturing and healthcare applications. He is the recipient of the INFORMS Data Mining best paper award.