The Dissertation Committee for Bo Lu
certifies that this is the approved version of the following dissertation:

Improving process monitoring and modeling of batch-type plasma etching tools

Committee:

Thomas F. Edgar, Supervisor

John D. Stuber

Michael Baldea

Roger T. Bonnecaze

Dragan Djurdjanovic

John G. Ekerdt

Improving process monitoring and modeling of batch-type plasma etching tools

by

Bo Lu, B.S.; M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2015

Dedicated to my family

Acknowledgments

First and above all, I would like to thank my adviser, Dr. Thomas F. Edgar, for his patience, wisdom and guidance throughout my graduate studies at the University of Texas. Thank you for always trusting me and encouraging me to persist through challenges. I would also like to thank Dr. John Stuber from Texas Instruments, who provided invaluable engineering knowledge and process insight to help kickstart my project. Next, I gratefully acknowledge the rest of my committee: Dr. Baldea, Dr. Djurdjanovic, Dr. Bonnecaze, and Dr. Ekerdt, who were always available, patient and provided timely feedback.

I would also like to thank my internship mentors and colleagues: Ivan Castillo, Leo Chiang, Swee-Teng Chin, Mary-Beth Seasholtz and Zdravko Stefanov. I am grateful for the opportunity to intern with the chemometrics group at Dow Chemical. I was able to work on topics that closely relate to my research. At the same time, I also gained invaluable industrial experience, which broadened my scope and inspired me in my own research.

I also want to thank my friends from the Edgar and Baldea research groups, who are great sources of knowledge and entertainment that made these years interesting and memorable. The people listed here are by no means exhaustive: Dr. Wesley Cole, Dr. Xiaojing Jiang, Dr. Jong Suk Kim, Shu Xu, Siyun Wang, Ray Wang, Jungup Park, Matt Walters, Ankur Kumar, Vincent Heng, Corey James, and Abigail Ondeck.

Lastly, the pursuit of my doctoral studies would not have been possible without the support of my family: my mother for her support and loving un-

derstanding, my father for his encouragement and critiques, and my girlfriend and best friend, Jiawen Li for her unwavering love, support and empathy.

Improving process monitoring and modeling of batch-type plasma etching tools

Bo Lu, Ph.D.
The University of Texas at Austin, 2015

Supervisor: Thomas F. Edgar

Manufacturing equipments in semiconductor factories (fabs) provide abundant data and opportunities for data-driven process monitoring and modeling. In particular, virtual metrology (VM) is an active area of research. Traditional monitoring techniques using univariate statistical process control charts do not provide immediate feedback to quality excursions, hindering the implementation of fab-wide advanced process control initiatives. VM models or inferential sensors aim to bridge this gap by predicting of quality measurements instantaneously using tool fault detection and classification (FDC) sensor measurements. The existing research in the field of inferential sensor and VM has focused on comparing regressions algorithms to demonstrate their feasibility in various applications. However, two important areas, data pretreatment and post-deployment model maintenance, are usually neglected in these discussions. Since it is well known that the industrial data collected is of poor quality, and that the semiconductor processes undergo drifts and periodic disturbances, these two issues are the roadblocks in furthering the adoption of inferential sensors and VM models.

In data pretreatment, batch data collected from FDC systems usually contain inconsistent trajectories of various durations. Most analysis techniques

requires the data from all batches to be of same duration with similar trajectory patterns. These inconsistencies, if unresolved, will propagate into the developed model and cause challenges in interpreting the modeling results and degrade model performance. To address this issue, a Constrained selective Derivative Dynamic Time Warping (CsDTW) method was developed to perform automatic alignment of trajectories. CsDTW is designed to preserve the key features that characterizes each batch and can be solved efficiently in polynomial time. Variable selection after trajectory alignment is another topic that requires improvement. To this end, the proposed Moving Window Variable Importance in Projection (MW-VIP) method yields a more robust set of variables with demonstrably more long-term correlation with the predicted output.

In model maintenance, model adaptation has been the standard solution for dealing with drifting processes. However, most case studies have already preprocessed the model update data offline. This is an implicit assumption that the adaptation data is free of faults and outliers, which is often not true for practical implementations. To this end, a moving window scheme using Total Projection to Latent Structure (T-PLS) decomposition screens incoming updates to separate the harmless process noise from the outliers that negatively affects the model. The integrated approach was demonstrated to be more robust. In addition, model adaptation is very inefficient when there are multiplicities in the process, multiplicities could occur due to process nonlinearity, switches in product grade, or different operating conditions. A growing structure multiple model system using local PLS and PCA models have been proposed to improve model performance around process conditions with multiplicity. The use of local PLS and PCA models allows the method to handle

a much larger set of inputs and overcome several challenges in mixture model systems. In addition, fault detection sensitivities are also improved by using the multivariate monitoring statistics of these local PLS/PCA models. These proposed methods are tested on two plasma etch data sets provided by Texas Instruments. In addition, a proof of concept using virtual metrology in a controller performance assessment application was also tested.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xii**

List of Figures **xiv**

Chapter 1. Introduction **1**

1.1 Virtual Metrology Motivation 1

1.2 Current Work in Virtual Metrology and Inferential Sensors . . 3

1.3 Dissertation Outline 8

Chapter 2. Data Preprocessing and Variable Selection **10**

2.1 Batch Data Challenges 10

2.2 Trajectory Alignment 13

2.3 Variable Selection for PLS Models 46

2.4 Summary 84

Chapter 3. Maintaining Quality and Performance of Data-driven Models **86**

3.1 Introduction 86

3.2 Integrated Moving Window Update Framework 89

3.3 Growing Self Organizing Map Multiple Model Systems for PLS models 102

3.4 Summary 141

Chapter 4. Industrial Applications **143**
4.1 Introduction 143
4.2 Etch Rate Prediction of a Polysilicon Gate Etch Process 144
4.3 Control Performance Monitoring Based on VM Estimated Process Gains 164
4.4 Metal Etch Relevant Variable Identification 178
4.5 Variable Selection Case Studies using Chemical Plant Data . . 187
4.6 Summary 200

Chapter 5. Summary and Future Recommendation **202**
5.1 Summary of Contributions 202
5.2 Future Work 204

Appendices **206**

Appendix A. Publications **207**

Appendix B. Common Multivariate Methods and Applications **209**
B.1 Principal Component Analysis and Multivariate Monitoring . . 209
B.2 Partial Least Squares Based Prediction and Monitoring 214
B.3 Nonlinear Extensions of PLS and PCA 216
B.4 Batch Process Analysis 220

Appendix C. Closed Loop Run-to-Run Control System Responses Derivation **224**

Bibliography **227**

Vita **245**

List of Tables

1.1 Select data driven applications in virtual metrology and other industries 6

2.1 Summary and assessment the warping methods for trajectory synchronization 15
2.2 The CsDTW algorithm for batch trajectory alignment 35
2.3 Critical overview and assessment of several selected representative variable selection methods in filtering, wrapper and embedded categories 52
2.4 The moving window VIP variable selection algorithm 66
2.5 Simulation conditions and variable selection algorithm settings used in the simulated case study 80
2.6 MW-VIP and VIP filtering variable selection results 82

3.1 Popular data-driven techniques and their adaptation mechanisms 88
3.2 T-PLS decomposition algorithm for single output PLS models 96
3.3 Algorithm details of sample weighted PLS algorithm 114
3.4 Prediction performance of the GSMMS framework in three simulated case studies 132
3.5 Fault detection speed for the Tennessee Eastman Process (lower is faster, blank means failure to detect) 138
3.6 Missed alarm rates in percentages for the Tennessee Eastman Process (lower is better, blank means failure to detect) 139
3.7 False alarm rates in percentages for the Tennessee Eastman Process (lower is better) 140

4.1 Model size, coefficient of determination, and cross-validated root mean squared residual of candidate models 160
4.2 Qualitative evaluation of the candidate models and their implementation difficulty 161
4.3 Simulation parameters used in generating Figure 4.14 167
4.4 Using recursive least squares to estimate ϕ in $(1 + \phi q^{-1})x_k = \epsilon_k$ 173

4.5 Simulation parameters for Case 1 - Plant/model gain mismatch, Case 2 - Improper tuning, Case 3 - Varied metrology delay with gain mismatch, Case 4 - Varied metrology delay with improper tuning 176
4.6 List of relevant variables identified through PLS-VIP Variable Selection 183
4.7 Metal etch PLS model performance summary 186
4.8 Results of the evaluation criteria study calculated using testing dataset I 198
4.9 Results of the evaluation criteria study calculated using testing dataset II 198

B.1 Common kernel functions used in kernel PCA, kernel SVM and kernel FDA 219

List of Figures

1.1 Conceptual illustration of how VM would improve process capability of a semiconductor manufacturing process, figure adapted from [1] 3

1.2 Offline and online stages of soft sensor life cycle 5

2.1 Motivation for performing batch trajectory alignment 14

2.2 Example solution of the Dynamic Time Warping Problem, figure taken from [2] 19

2.3 Illustration showing that only select regions in the batch trajectory are relevant for modeling and monitoring of the batch process 21

2.4 Flow chart of Constrained selective DTW for alignment of a *target* trajectory based on a *reference* trajectory 22

2.5 Trajectory plot and diagnostics from before and after DTW . 27

2.6 Alignment markers (key-feature markers) being assigned to location M1 and M2 on the target trajectory 28

2.7 Determining the marker boundary a^* in subsequence marker identification 30

2.8 Example result of subsequence marker identification using an feature segment on target trajectory 32

2.9 Three local/slope constraints typically seen in DTW. (a) No slope constraint, (b) Slope constraint of $(0.5, 2)$,(c) slope constraint of $(1/3, 3)$ [3]. 32

2.10 Three global constraints: (a) Sakoe-Chiba band with width T, (b) Itakura constraint,(c) path p^* in violation of the Sakoe-Chiba constraint [3]. 33

2.11 Example of the alignment markers being applied as global constraints, Warping is only allowed in the black regions. 34

2.12 Reference and target trajectories to be aligned 36

2.13 Comparison of the five warping methods on (a) simulated trajectories, (b) voltage probe data 38

2.14 Warping alignment diagnostics of DTW versus CsDTW 39

2.15 Re-folded 2 dimensional heatmap of a VIP matrix from a multiway PLS model 43
2.16 Flowchart for iterative marker selection and CsDTW alignment 44
2.17 Illustration of a typical soft sensor model development process 48
2.18 Three categories of PLS variable selection methods: (a) filter, (b) wrapper and (c) embedded 49
2.19 Subwindow Permutation Analysis (SwPA) variable selection algorithm overview 56
2.20 Sparse PLS variable selection algorithm overview 59
2.21 Illustration of the different multiple operating mode dynamics in training data and testing data 61
2.22 MW-VIP score of a 60 variable dataset as a function of variables and moving window sample 62
2.23 Moving window VIP variable selection algorithm overview . . 63
2.24 The moving window data partition scheme where smaller subsets of size N_{sample} offsets neighbouring subset by N_{step} samples 65
2.25 Moving window VIP contour showing different levels of smoothing under different moving window size ($N_{\text{window}} = 500, 1000, 1500, 2000$) 68
2.26 Flowchart showing the steps to calculate residual distribution shift score 75
2.27 Example of residual distribution shifts calculated using testing dataset II (a) Residual moved toward the left after parameter updates, (b) Relatively stationary residuals before and after model parameter updates 76
2.28 Variable correlation map of the training data $\mathbf{X}$ showing two clusters of collinear variables 78
2.29 Simulated training data trajectories of $\mathbf{X}$ and the corresponding quality variable $\mathbf{Y}$ containing two operating modes 79
2.30 Cross-validated Q^2 as a function of variables selected by MW-VIP and VIP filtering methods 82
2.31 Trajectories of the three highest ranked variables from the VIP variable selection and the output variable 83

3.1 Proposed integrated VM framework using T-PLS multivariate screening and moving window model updates 91
3.2 Unfolding of a three-dimension data array($\mathbb{R}^{n\times m\times p}$) into a two-dimensional matrix ($\mathbb{R}^{n\times mp}$) 93

3.3 Moving window data partition of a dataset using two parameters: sample size(N_{sample}) and step size (N_{step}) 94
3.4 (a) PLS Hotelling's T^2 statistic, (b) PLS decomposed subspace process monitoring indices, top-left: output-relevant T^2, top-right: output-orthogonal T^2, bottom-left: output-orthogonal residual T^2, bottom-right: left-over residual Q_r^2, red dashed line is the 95% significance level 99
3.5 Time series plot of the training data at the 1st moving window plus the T-PLS fault diagnostics for the training data 100
3.6 Flowchart of the combined integrated framework for online moving window PLS update subjected to T-PLS outlier filtering . 101
3.7 Example of SOM vs GSOM under different tuning parameters, figure taken from [4] 106
3.8 Parameters that need to be specified in a GSMMS system: The GSOM network, and local model at each node 109
3.9 Basic flow of information in conventional multiple model systems 110
3.10 Overall training procedure for GSMMS models 110
3.11 Topological distance of each node with respect to node m . . . 112
3.12 GSMMS inner loop iterative update of ξ_m as described in Step 3 116
3.13 GSMMS node insertion into region with highest local fitting error 119
3.14 GSMMS local parameter update algorithm 121
3.15 Schematic of the prediction and the online adaptation of proposed GSMMS framework 125
3.16 GSOM node error and codebook location just after initialization 127
3.17 GSOM node error and codebook location after training has been completed 128
3.18 Testing data score plot and the prediction results using the GSOM model 129
3.19 Training and testing results of nonlinear simulated case study 130
3.20 Score plot and GSOM structure for a case with operating mode overlap 131
3.21 Fault signatures (red crosses) plotted on the principal component plane of the PCA_{normal}, green circles denote the normal operating region 134
3.22 GSMMS network trained using simulated Tennessee-Eastman process data 135

3.23 Decision tree of the GSMMS-PCA for detecting faults in the Tennessee-Eastman case study 136

4.1 Schematic of a typical plasma etching system 145
4.2 Summary of etching mechanisms and typical problems in plasma etching 145
4.3 Transmission Electron Micrograph (TEM) of a polysilicon gate geometry showing the quality variables of interest 146
4.4 CD control flow diagram showing an example scheme utilizing test wafer FICD and DICD measurements 147
4.5 raw data plot 149
4.6 Three representative trajectories before and after alignment . 151
4.7 Raw and simplified PLS model performance on training and validation data 152
4.8 VIP information derived from the Raw PLS model used in model simplification 154
4.9 (a) Effect of moving window PLS update on predicting etch rate (bottom) compared against PLS without moving window updates (top), (b) Normalized regression coefficient of top four variables and the bias over the moving window range 155
4.10 Integrated Framework Predicted vs Measured 157
4.11 Gate Etch adaptive GSMMS (a) post-training and (b) final (after testing data update) network structure 159
4.12 Prediction results of the Adaptive GSMMS model for the testing dataset 159
4.13 VM-assisted Controller Performance Assessment Scheme . . . 166
4.14 Simulated profile of a SISO Run-to-Run Process with and w/o EWMA controller with fixed set-point 168
4.15 Internal Model Control Representation of a Discrete Run-to-Run Controller 169
4.16 Comparison of plant model mismatch index $\xi_{mismatch,k}$ with the estimated disturbance difference for mismatch occurring at $k = 300$ 173
4.17 Controller performance assessment of SISO EWMA controller under scenarios listed in Table 4.5 175
4.18 The influence of process gain VM prediction error on the detection speed and false alarm of the CPA VM index 177

4.19 Schematic of the sheet resistance measurement contacts on the device surface from a die on the wafer 178
4.20 Overall metal etch data analysis workflow schematic 180
4.21 Metal etch trajectory alignment using DTW and CsDTW . . . 181
4.22 Score plot of the full multiway-PCA model 181
4.23 Variable Importance in Projection (VIP) map showing key trajectory features that yielded high correlation with the output . 183
4.24 Sheet resistance prediction performance of a metal etch PLS model 184
4.25 Score plot of the reduced PLS model 185
4.26 Reduced dynamic PLS model performance 185
4.27 Multiple operating mode of industrial data visualized in (a) histogram plot and (b) PCA score plot 188
4.28 Cross-validated Q^2 scores of different variable selection methods as a function of the number of parameters included 190
4.29 Time series plot of the measured and predicted quality variable for the training data 192
4.30 Testing data I model validation of the five candidate models (VIP filtering, MW-VIP, SwPA, SwPAi and sPLS) 193
4.31 Testing data II model validation of the five candidate models (VIP filtering, MW-VIP, SwPA, SwPAi and sPLS) 194
4.32 Measured and predicted quality variable from VIP filtering, SwPAi, sPLS and MW-VIP models in testing dataset II 195
4.33 Contribution score plot example of the discrepancies in prediction due to local variables identified by MW-VIP variable selection 197
4.34 Model process monitoring sensitivity (T^2 ratio) as a function of the model fit performance, the ellipse highlights the expected correlation between R^2 and T^2 ratio values 200

B.1 Unfolding of a three-dimension data array($\mathbb{R}^{n\times m\times p}$) into a two-dimensional matrix ($\mathbb{R}^{n\times mp}$) 220
B.2 Tucker3 decomposition of a three-dimensional batch data structure 222

Chapter 1

Introduction

1.1 Virtual Metrology Motivation

There has been an exponential increase in storage capacity, sensor availability and computing power in recent decades. Continuous improvements in energy efficiency, safety, reliability have become performance drivers for many manufacturing industries, where multivariate process monitoring, model-based control and plant wide optimization strategies play important roles [5]. The semiconductor industry in particular provides abundant data and opportunities for process monitoring and data-driven modeling. Integrated circuit manufacturing takes place in semiconductor factories called fabs. In the manufacturing process, a recipe defines a set sequence of unit operations to be applied to the wafer. Some of the example unit operations are oxide growth, chemical-mechanical polishing, film deposition of metals, silicon and dielectrics, dopant ion implantation, lithography, and etching [6]. These unit operations create multiple layers of specifically patterned films on the wafer surface, forming the integrated circuit (IC) that enables device function. Typically, more than a hundred of these unit operations are needed in modern IC devices. The process tools in a fab require large capital expenses and routine maintenance. To maximize their utilization, fabs produce multiple products simultaneously using high-mix threaded manufacturing. The sequence of tools. recipes, and products involved defines the "manufacturing context", which may include

the tool id, product id, layer id, and so on. The steps or runs with the same manufacturing context are considered as "one thread". Advanced scheduling and recipe management tools keep track of the recipe and progress of each wafer. From a fixed viewpoint such as a plasma etch tool, the recipe processed between each run might vary depending on the product type.

To ensure quality and yield, modern fabs typically rely on run-to-run process control with external metrology. Only the most critical processes have metrology on every wafer; typically a dynamic sampling algorithm will adjust the sampling frequency between one wafer per lot and one wafer every three lots. But even in cases where metrology data is collected on every wafer, excursions in product quality are not detected until many more wafers have been processed. The inherent delays in external metrology prolong product cycle time and introduce additional complexity in run-to-run controller tuning. Modern semiconductor fab tools are equipped with fault diagnostic sensors that generate large amounts of real-time data. These trace data have been shown to correlate with quality relevant outputs[7, 8, 9, 10].

Virtual metrology (or inferential sensors) aim to predict end-of-batch outputs using trace data and other end-point sensor readings. This concept is illustrated graphically in Figure 1.1 Having a VM signal would allow the fault diagnostic system to intervene and limit the quantity of wafers processed under excursion conditions. This also reduces false alarms by calling for interventions only in cases that directly impact wafer quality. In addition to excursion prevention, a reduction in the average lot cycle time can be achieved by replacing physical metrology with VM on some fraction of the lots, and the run-to-run system can adjust some recipe set-points to compensate for variations in product quality that do not reach the excursion limit. However, the

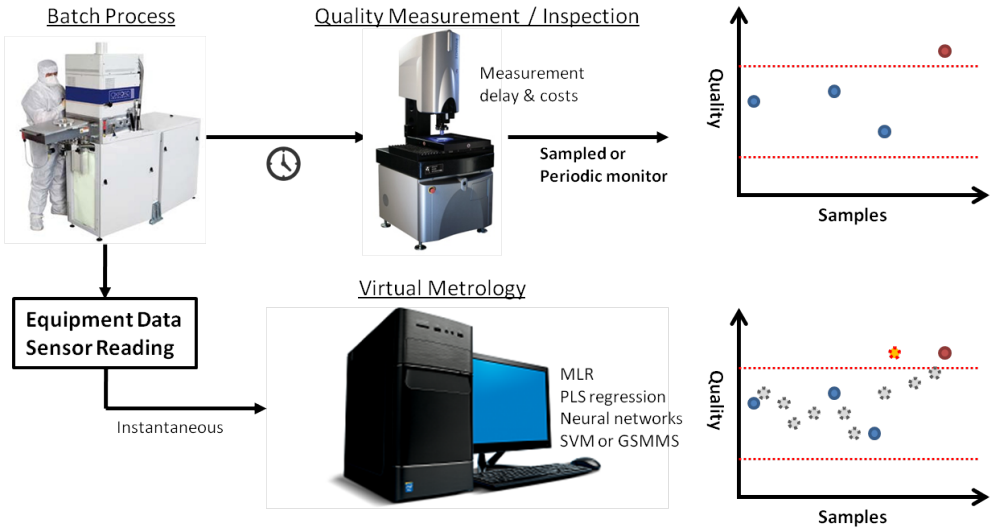


Figure 1.1: Conceptual illustration of how VM would improve process capability of a semiconductor manufacturing process, figure adapted from [1]

raw data collected from the tools are usually low in information. Significant data preprocessing, cleaning, and modeling have to be done. In this dissertation, we focus on improving the development process of data-driven models and their maintenance with special consideration in plasma etch applications.

1.2 Current Work in Virtual Metrology and Inferential Sensors

A wide array of published work are available on virtual metrology and soft sensors in the literature. Table 1.1 summarizes some of the representative works. The approaches used in these works (such as partial least squares, principal component analysis, kernel PCA/PLS, neural networks) are discussed in the Appendix B in more detail and are omitted here for sake of brevity. A common theme in these studies is that the methods are mostly focusing on

the algorithmic aspects of soft sensor development. Success or failure is largely dictated by the prediction performance of the developed techniques. However, as Kadlec et al. pointed out [11], there are inherent trade-offs between model complexity and prediction performance. With literature results advocating for more and more complex models in return for marginal improvement in prediction, it becomes more challenging to maintain the developed models in online environments. Thus, from a life cycle analysis perspective, the actual regression algorithm plays a minor role during the development and deployment processes of a soft sensor model. A typical model development follows the flow chart in Figure 1.2. It should be emphasized that process knowledge and the use of engineering judgement are required in many steps of the model development process; in addition, many of the steps listed are done repeatedly until a candidate model is able to meet the required performance criterion. With these factors in consideration, my research objective is to address the key limiting steps in data-driven modeling of batch processes. Since batch processes are common in many industries, these techniques are fairly general and can be carried over without much modification.

1.2.1 Batch data pretreatment

Data preprocessing is one of the critical steps that impacts the performance and interpretability of the derived models [11, 16, 13]. In continuous processes, data quality issues manifest themselves in instrument precision, sampling frequency, measurement outliers, and missing values in the process data. A wealth of literature and industry experiences are available to address most of these challenges [21]. In batch processes such as plasma etch, a multiway unfolding is usually first performed to convert the measurement at each

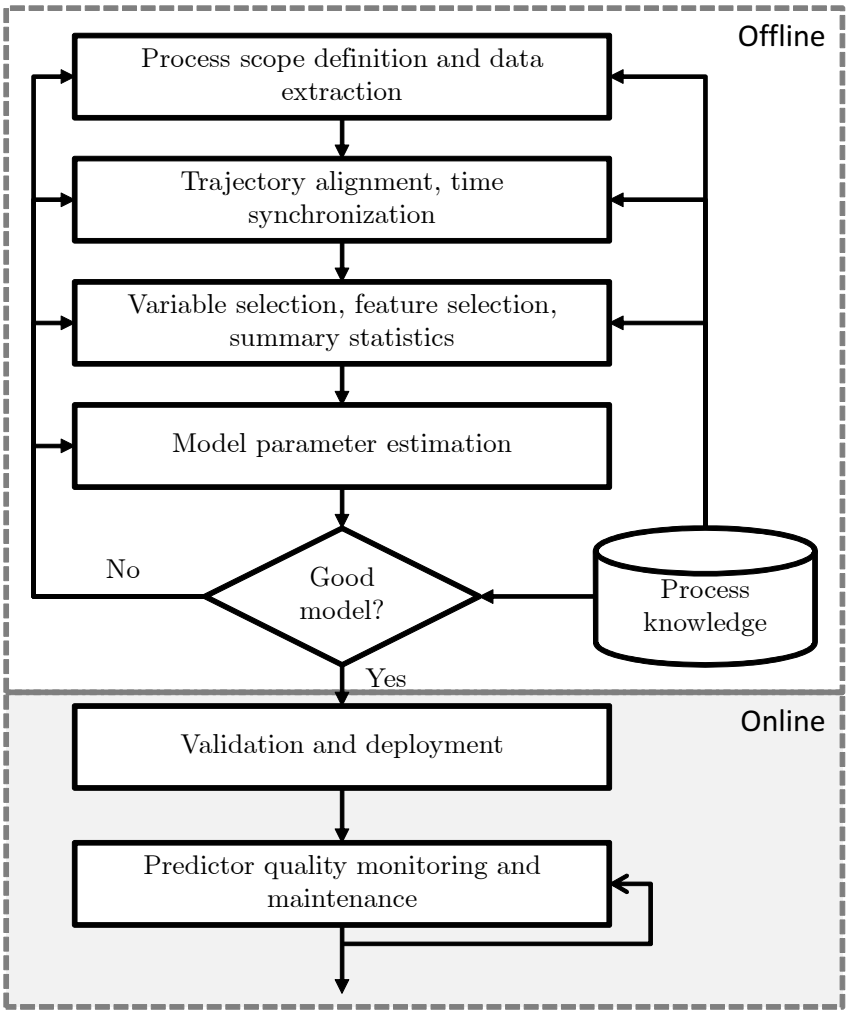


Figure 1.2: Offline and online stages of soft sensor life cycle

Table 1.1: Select data driven applications in virtual metrology and other industries

Authors	Method	Application	Description	Assessment
Gill and Edgar [12]	PLS, static Kalman filtered MLR, moving window PLS	Semiconductor plasma etch	Compares the performance of PLS models with several other adaptive approaches	Data pretreatment procedure was unclear, model not tested across maintenance cycles
Cheng et al. [8]	MLR	Semiconductor CVD	Proposes a mechanism to monitor model quality by looking at the statistical distribution of the prediction outputs	Assumes Gaussian assumption in determining if prediction is reliable, results shown for only 20 wafer samples
Zeng and Spanos [13]	PLS, MLR, neural networks	Semiconductor plasma etch	Reviews the overall model development process and evaluates resulting model (PLS, PCA regression, and BPNN) performances	Demonstrates the need for variable selection and alignment, methods description not detailed enough for reproduction
Lynn et al. [14]	PLS, neural networks, Gaussian process regression	Semiconductor plasma etch	Compares the performance of regression methods listed, also compared cluster, window adaptation, and static model performances	Identifies limitation in data availability for smaller local models, GPR shown with promising results but requires a pre-selected small number of inputs which makes them challenging when variable selection changes
Bleakie et al. [15]	Growing structure systems, MLR	Semiconductor PECVD	Divide the nonlinear drifting operating space into smaller partitions using GSOM	Allows for better adaptation against abrupt process changes, but suffers in higher dimensions with more inputs.
Lin et al. [16]	Dynamic PLS, Hampel Identifier	Rotary kiln NOx monitoring	Applies a combination of recursive PCA outlier detection and a static dynamic PLS model to predict NOx concentration and product concentration in a kiln	demonstrates industrial success of a NOx soft sensor using online data, also shows the need for online outlier removal and data processing for online deployment
Kadlec et al. [17]	Incremental local learning, MLR, PLS	Polymerization reactor catalyst activity	Use fuzzy combination with local experts, partitions based on relative residual change	Works for nonlinear methods but prone to over-fitting, difficult for input with higher dimensions
Fujiwara et al. [18]	Just-In-Time adaptation, PLS	Chemical refining, continuous	Locate the most correlated segment of data and train a model to make prediction	The Just-in-Time allows for better adaptation against abrupt process changes, but has large memory and storage requirements
Galicia et al. [19]	Reduced order, summary statistics, PLS	Paper industry, continuous Kamyr digester	Proposes using time delay estimation first and then introduce lagged variables to better predict system dynamics	Simulated case study showed good performance, but industrial data did not show improvement of the RO-DPLS over traditional methods
Dayal and MacGregor [20]	Weighted recursive PLS	mining process for online DMC control	Multioutput PLS (PLS2) prediction of ten process output variables simultaneously, model is updated using recursive PLS with forgetting factor	The forgetting factor for speed of adaptation was tuned based on testing data. The adaptive algorithms still experienced numerical difficulties resulting in predictions that "blow up" occasionally

time sample into an individual variable [22]. The batch level regression (PLS) or fault detection model (PCA) then attempts to correlate variations in these variables to the batch to batch variation in the outputs. While this process has been demonstrated to work well for a number of problems, the successful case studies usually assume that the batch recipes run for a fixed duration with consistent trajectory profiles that can be compared laterally across different batches. This is usually not the case for semiconductor processes. In addition, to reduce the number of inputs to the regression models, batch process models undergo feature extraction and data summarization. These steps are usually based on intuition and qualitative process knowledge and requires multiple trial-and-error attempts. The discussions in Chapter 2 focus more on developing more rigorous methods of performing trajectory alignment and feature extraction for batch processes.

1.2.2 Model performance degradation

Performance degradation in deployed soft sensors has been a key issue in soft sensor research. Processes are never truly operating at a steady-state when examined from a longer time-scale. Equipment maintenance, sensor degradation, and personnel change all could have an impact on how the system is operated. As a result, over periods of time, the process condition usually deviates from the original process condition when data-driven models are developed. In addition, multiplicity is an challenge to soft sensor models, where switching between product grade, recipe and equipment occurs due to schedule or production constraints. Semiconductor data are characterized by both multiplicity and gradual process drifts.

There have been many work on addressing these challenges [23, 24, 25,

26, 27, 8, 28]. The main approach is to provide an adaptation mechanism to allow automatic re-training of the online models. In these cases, data collection and sometimes lab sampling of important quality variables cannot be displaced entirely and still needs to be performed (maybe at reduced intervals). Selecting what data to use to update the model becomes a difficult question to answer. In addition, depending on the type of soft sensor in place, the difficulty of retraining the model efficiently could range from trivial (simple least squares) to very challenging (support vector machines, neural networks, gaussian process regression). Furthermore, since most of these methods are tested off-line where outlier removal was already done on the entire dataset, these recursive and adaptive methods do not consider the issue of faulty update data. Process multiplicities could also cause problems for soft sensors. Multiple model systems and mixture models have been shown to be effective in dealing with these scenarios. In Chapter 3, the issue with model maintenance and modeling to deal with multiplicity will be examined in more detail. Two novel methods are proposed to complement the existing approaches for model maintenance of batch type processes.

1.3 Dissertation Outline

This next few chapters of this dissertation are organized as follows.

Chapter 2 focuses on the data preprocessing step prior to model development. The first section discusses a new trajectory alignment (data synchronization) method named Constrained selective Dynamic Time Warping (CsDTW). The second section gives a critical overview of existing variable selection techniques specific to Partial Least Squares (PLS) models. Following the critical overview, a new technique utilizing a moving-window approach

is introduced. The proposed new technique aims to address the shortfall of current methods when selecting variables from industrial process data.

Chapter 3 discusses the topic of model maintenance and advanced modeling frameworks. To improve the robustness of adaptive online PLS models, a novel technique using total partial least squares (T-PLS) decomposition to screen out outliers in the update data is proposed. To deal with multiple operating states and process nonlinearity, a novel framework that combines growing self organizing map with local PLS and PCA models is developed.

Chapter 4 presents the results of case studies conducted using data provided by Texas Instruments. The first section discuses the development process of virtual metrology models for a gate etch and a metal etching process. The second section discusses an novel application of the VM results in diagnosing model-plant mismatch and improper tuning of run-to-run controllers. Lastly, a variable selection case study using industrial process data from a chemical refining process is presented. This industrial case study demonstrates the effectiveness of the proposed variable selection method in Chapter 3.

Finally, Chapter 5 details the scientific contributions of this doctoral research along with a recommendation of future work.

Chapter 2

Data Preprocessing and Variable Selection

2.1 Batch Data Challenges

Unlike typical time series data that are collected from a system under continuous operation. There are multiple challenges associated with collecting, pre-processing and analysis of batch data sets. The main challenges listed here are by no means exhaustive:

Data Quality Data quality of batch data collected from industry is usually inferior to its continuous counterparts. This problem has been highlighted by many researchers in the area [29, 30, 31, 5]. Because batch processes usually follow a pre-defined recipe, some of the instrumentation and sensors available during the design stage of the process are not available in production environments. This leads observability problems on some of the critical state variables that are difficult to estimate ex-situ.

Recipe Changes Depending on the specific batch process being analyzed, the recipe settings could remain fixed or be allowed to adapt. This is a typical scenario in plasma etch run-to-run control, where the processing time (or other manipulated variable) will be adjusted depending on final critical dimension measurement feedback from the previous lots [32, 33]. The changing process conditions results in different length of time-series data, which requires alignment and synchronization before modeling.

Furthermore, most of the multivariate statistical analysis techniques make the assumption of steady-state operation and homoscedasticity, meaning that the covariance structure of the data being analyzed is assumed to be relatively constant. The validity of these assumptions are uncertain in cases where there are constant drift and operational changes in the batch processes, leading to performance degradation and sometimes failures of the derived models.

High Dimensionality Batch data can be visualized as three dimensional data cubes(variable, relative sampling time, batch). These three dimensional data structure are difficult to process using existing multivariate statistical analysis and data-mining techniques. Although methods such as PARFAC[34] and ALS[35] can deal with these three-way data structures, the end result and the model are often difficult to interpret and lack transparency. A more common approach is through multi-way analysis which unfolds the data into two dimensions. Since one of the dimensions will be a composite dimension (its size is the product of the two original dimensions), the dimensionality of the unfolded data is high, and frequently exceeds thousands of variables. The high dimensionality of the input dataset introduces problems in parameter estimation, matrix inversion and over-fitting, which must be accounted for in modeling and analysis.

Multiple Phases Having multiple phases in batch processes with unsynchronized phase durations is another problem that needs to be addressed for batch data [31]. Depending on the specific application, a particular phase in a batch process can be configured to be open-loop, where a process is performed for pre-determined period of time, or closed-loop, where a

certain phase termination criteria must be met (for example, tank level has to exceed a certain threshold, or the reactor has to be at a certain temperature). In these instances, many factors could potentially affect the phase evolution of the batch process. For example, if an end-point detection sensor is broken or worn out, then during the downtime, the batch process might be controlled manually and result in drastically different length and properties from runs. In addition, the multiple phases are not equal in significance on the final product quality. Some of the phases are simply preparatory or cleaning-up stages, whereas the reaction phase might have the most impact on the product quality.

Nonlinearity and Non-steady state Since batch processes do not operate at a steady-state, accumulation, depletion or reaction of materials are always taking place in the sampled duration of each batch. As a result, it is difficult to determine features to fully capture the variations or the evolution of the process variable trajectories in a typical batch.

Lack of Quality Measurements In regression problems involving batch data, the quality variable of interest is measured at the end of the batch, where one measurement is made for every batch processed. Furthermore, to save measurement costs, the quality measurement is only done selectively on certain samples in every lot. The lack of available measurement restricts the amount of data available for modeling and analysis.

2.2 Trajectory Alignment

2.2.1 Motivating example

As the previous section has discussed, data collected from batch processes are inconsistent in duration and across different variables and batches. In addition, time series data in general also experiences analogous problems when comparisons are required. In these cases, it is always desirable to make sure the important trajectory features in the data (rise, overshoot, drop-off, peaks, valleys) are aligned. Figure 2.1 illustrates the goal of trajectory alignment and its impact on subsequent batch process modeling. In batch process modeling, it is often desirable to not only align geometric trajectory features (rise, decay, peaks and valleys), but also event boundaries across different phases.

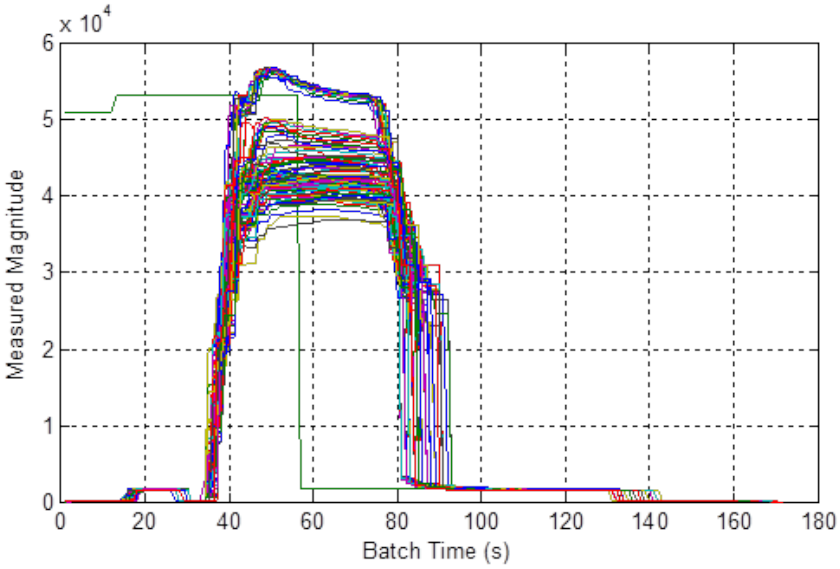
2.2.2 Overview of existing trajectory alignment techniques

There are many methods in practice that aims to synchronize trajectories of that are of different durations. These methods differ in computational complexity and also their alignment objective. Table 2.1 provides a brief summary of the advantages and disadvantages of each method.

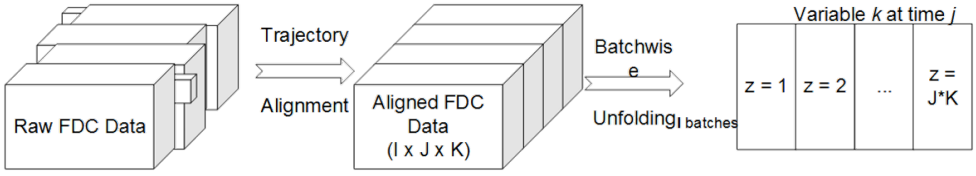
There is no best alignment method that universally applicable. It is often necessary to attempt multiple alignment solutions to determine the most appropriate method for the given problem. Below, we briefly introduce the key concepts and the mechanisms behind each alignment technique.

2.2.2.1 Truncation and padding

In truncation and padding, the alignment technique is very simple and is based mostly on intuition. The assumption made in truncation and padding



(a) Actual trajectory from a batch process showing misalignment



(b) Schematic showing the trajectory alignment workflow

Figure 2.1: Motivation for performing batch trajectory alignment

Table 2.1: Summary and assessment the warping methods for trajectory synchronization

Method	Description	Advantages	Disadvantages	Speed
Truncation and Padding	Delete (longer) the extra time series or pad (shorter) time series data to the same length based on a golden reference trajectory	Easy-to-implement	Does not explicitly align features	Fast
Linear Time Scaling	Use a monotonic batch evolution indicator variable (percent completion, product concentration, tank level) and stretch or shrink linearly to ensure every trajectory is of same length	Straightforward	Performance dependent on good indicator variable	Fast
Dynamic Time Warping	Solves dynamic programming problem to find the most similar path between two trajectories	Explicit feature alignment	Information loss, trajectory distortion, implementation challenge	Medium
Correlation Optimized Warping	Solves dynamic programming problem with suboptimization routines to maximize correlation between trjaectory	Explicit feature alignment, less distortion than DTW	Information loss, complex, implementation challenge	Slow

is that the initial and the end of the time series data are not as important as the middle section of the data. When this assumption is valid, the longer time series trajectory can be simply shortened by removing the head or the tail of the data series. For shorter trajectories, average value from the head or tail of the data series is calculated and then padded on to extend the length of the trajectory being aligned.

This method guarantees that the processed trajectories are of equal length. However, this method does not ensure that the dynamic features within the time series data are properly aligned. As a result, its fidelity is often not as high as other advanced methods. However, due to its simplicity and ease of use, simple truncation or padding could serve as an effective first-pass in assessing the nature of the batch dataset being analyzed.

Lastly, truncation and padding offers the minimal amount of information loss and distortion to the given trajectory, and preserves all the dynamic features provided that the features of interest are not in the truncation region (head or tail region of the trajectory).

2.2.2.2 Linear time scaling

Linear time scaling (LTS) is another popular alignment technique that is easy to execute and very efficient. This alignment method has been made available in many commercial data analysis software packages [22]. In linear time scaling, a batch maturity variable is chosen from the available measurements. This variable is required to be monotonic and should ideally be indicative of the progress of a batch process. For example, if an accumulation reaction takes place inside a CSTR reactor with no outlet stream, then the level of the reactor could be an indicator of the progress of reaction. Since the

level of the tank will always start empty and increases until it is full, each level reading corresponds with an unique one-to-one mapping towards the progress of that particular batch.

To perform alignment, the indicator variable $\mathbf{p}$ is discretized from its initial value to the final value.

$$\begin{aligned} \delta p &= \frac{max(\mathbf{p}) - min(\mathbf{p})}{K-1} \\ p_0 &= min(\mathbf{p}) \\ p_{k+1} &= p_k + \delta p \\ p_K &= max(\mathbf{p}) \end{aligned}$$

The number of discretization points (K) is a controllable parameter, but is usually set to the average number of samples in each batch observation. At each discretized value of the indicator variable p_k, linear interpolation is then performed for every measurement x_j.

$$x_{j,k} = x_j^a + (x_j^b - x_j^a) * \frac{p_k - p^a}{p^b - p^a} \tag{2.1}$$

where superscripts a and b represents the nearest boundary indices that encloses the value p_k. This results in K number of measurement vectors $\mathbf{x}_k$ for $k \in (1, K)$. Since every batch will be interpolated to these K samples after LTS, the trajectories will be aligned and ready for subsequent analysis steps.

The main advantage of LTS is that this is a very efficient technique to align multiple trajectories at the same time. Provided that an accurate linear batch progress indicator variable can be found (meaning that the value of this variable varies linearly from the start to the end). The main drawback

of LTS method is the reliance of a maturity variable. Since this variable is used essentially to replace "time" as an alternative batch progress indicator, improper selection or unavailability of such a variable greatly affects the result of LTS method. However, ways to circumvent this problem using with-in batch PLS models have been reported [36].

2.2.2.3 Dynamic time warping (DTW)

Dynamic time warping (DTW) is an advanced trajectory synchronization technique originally developed for synchronization of sound waves in speech recognition application. In DTW, a optimization problem is formulated to maximize the geometric similarity. An efficient solution of this optimization problem can be found using dynamic programming, hence the name of the warping technique. Kassidas et. al. first [37] applied this method in batch type process modeling. Because DTW explicitly defines the trajectory optimality in the objective function, this method has been very popular in the academic community. In the original formulation for speech recognition, a symmetric alignment is performed, meaning that both the reference and the target trajectory are warped. However, for the purpose of batch trajectory alignment, an asymmetric version of the alignment technique is often used. The asymmetric version of the alignment technique is introduced here in more detail.

2.2.2.4 Correlation optimized warping (COW)

Correlation optimized warping (COW) is a classic segment-wise alignment method. Nielson et al. [38] developed this method for the alignment of wavelength features in chromatography results. DTW without extensive

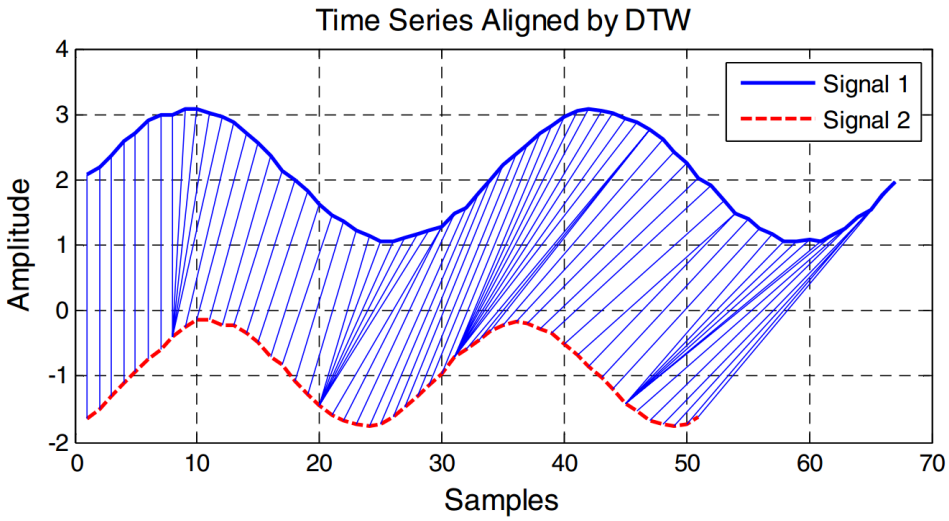


Figure 2.2: Example solution of the Dynamic Time Warping Problem, figure taken from [2]

tweaking and adaptation is not suited for this application because of distortion and artifacts.

To perform COW, the trajectory is first cut into little segments, each segments are stretched or shrunken to optimize the overall objective function value (in this case, the correlation between two trajectories). The correlation of each discretized segment between the reference and the target trajectories are calculated after each warping step, a dynamic programming scheme similar to DTW is employed to perform this comparison from the end to the beginning to select the optimal slack/shift lengths for each segment. Implementation details of COW can be found in [38, 39].

An advantage of COW over DTW is that COW does not require preprocessing, such as calculating the derivative, or applying filters to smooth out the trajectory to be aligned. The tuning parameters of COW also make

intuitive sense and can be selected by gauging the average width of the chromatograph peaks to be aligned. However, the trade-off in COW is that the method is much more computationally intensive, since correlations are used instead of geometric distances in the optimization.

2.2.3 Constrained selective Dynamic Time Warping (CsDTW)

As outlined in Table 2.1, the four main types of alignment techniques all have their advantages and limitations. While LTS and truncation methods are easy to perform because of their simplicity, the alignment results are sometimes unsatisfactory. More advanced methods such as DTW and COW employ optimization to achieve a better alignment, but the algorithms might create distortion and artifacts. They are also computationally expensive. Furthermore, because of the objective functions in these methods only guarantee geometric alignment, the end result do not justify why the features should be elongated or shrunken; thus leading to further scrutiny under first principles based knowledge.

Figure 2.3 shows a hypothetical scenario during warping where the inverse response before the ramp up and the depletion near batch completion are assumed to be correlated to the product quality. In this case, it is desirable to truncate head and tail segments while keeping the key batch features (highlighted in rectangular boxes) free of distortion. In these cases, it would not make sense to apply DTW or COW directly on the trajectory, since these key geometric features will be made identical, thus destroying the information in the variation across multiple batches. On the other hand, if simple truncation is applied, the second feature (the depletion step) would be out of alignment if the initial starting point is truncated, leading to poor alignment affecting

subsequent analysis.

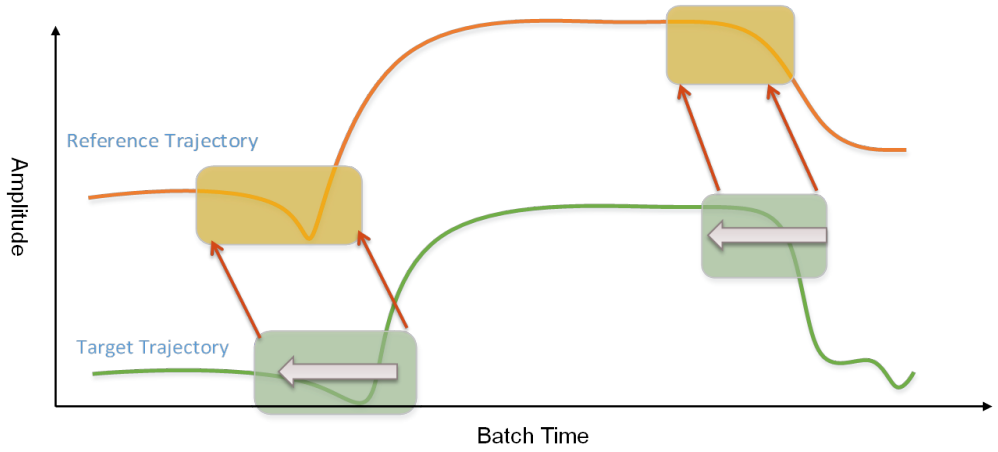


Figure 2.3: Illustration showing that only select regions in the batch trajectory are relevant for modeling and monitoring of the batch process

To address these limitations that are encountered for *batch data trajectory alignment*, a new method based on DTW is developed. The main motivation is to preserve the information within the dynamics of the trajectory as much as possible during alignment. Since any alignment performed on the trajectory will always introduce some distortion and information loss, the new proposed method reduces the distortion on the trajectory by selectively warping the trajectories at regions of least impact on the output quality. As a result, the proposed method is called *Constrained selective Dynamic Time Warping (CsDTW)*. The key assumption in CsDTW is that *certain segments in the batch trajectories are less important and can be allowed to freely warp, while other trajectory segments are important to the process output and should be preserved as much as possible.* Following this assumption, the problem can be divided into four steps as shown in Figure 2.4.

Figure 2.4 presents a flowchart detailing the overall procedure of Cs-

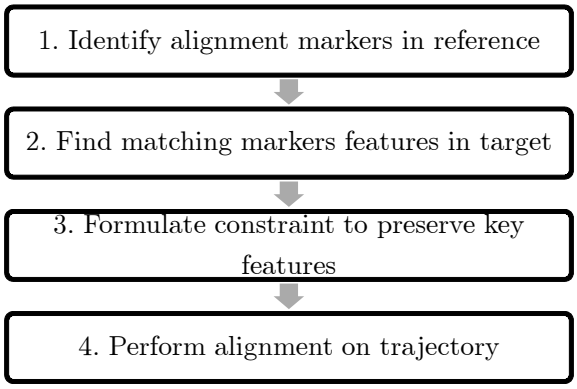


Figure 2.4: Flow chart of Constrained selective DTW for alignment of a *target* trajectory based on a *reference* trajectory

DTW. In the first step, important alignment markers of the reference trajectory first need to be identified. These alignment markers reveals the location of the important trajectory features that need to be preserved in the warping process. First principle knowledge could be incorporated in this step to highlight important features based on process experience. In the absence of first principle knowledge, variable selection and variance ratio analysis could also guide the selection of appropriate alignment markers. The second step aims to validate the existence of similar features in the target trajectory to be warped. Since these key features could occur at different locations, this step requires the efficient search of the entire trajectory length to identify the correct markers. In the last two step of the process, traditional DTW based alignment technique is carried out. The initial alignment markers are formulated into the global constraint of DTW optimization to ensure that the identified features are properly preserved. The rest of the trajectories are then allowed to freely vary and allow for optimal alignment and synchronization of batch trajectories.

2.2.3.1 Mathematical framework of DTW

Before additional details about the proposed warping mechanism can be discussed, the mathematical framework for warping and alignment in DTW is introduced here. The nomenclature and the terminology follow the convention in [3].

First, we define reference trajectory R and target trajectory T of length N and M respectively as follows:

$$R := (r_1, r_2, \ldots, r_N), T := (t_1, t_2, \ldots, t_M) \tag{2.2}$$

We assume that the two trajectories are sampled at the same sampling interval. To measure the quality of alignment, a local cost function (or local distance measure) $\mathbf{c}$ is defined as:

$$c(r_n, t_m) = \|r_n - t_m\| \tag{2.3}$$

This definition assumes that r_n and t_m is of same dimension (which means same number of variables being measured).

Using the local cost function definition, we can define an alignment cost matrix $C \in \mathbb{R}^{N \times M}$

$$C(n, m) := c(r_n, t_m) \tag{2.4}$$

The goal of DTW is to find a warping path through the local cost matrix that gives us the minimal cumulative cost between the two trajectory. The warping path P is a series of pair-wise vectors that map sample indices from trajectory T to sample indices on R, and is defined as $P = (p_1, \ldots, p_L)$ and $p_l = (n_l, m_l) \in [1 : N] \times [1 : M]$. The warping trajectory is constrained by three conditions:

- Boundary condition: $p_1 = (1, 1)$ and $p_L = (N, M)$.
- Monotonicity condition: $n_1 \leq n_2 \leq \ldots \leq n_L$ and $m_1 \leq m_2 \leq \ldots \leq m_L$
- Step-size condition: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L - 1]$

The boundary condition ensures that the first and the last samples map to the beginning and the end of the two trajectories. The monotonicity condition ensures that there will be no backtracking or "folding" of trajectories during warping. The step-size condition ensures that no sample indices will be skipped and limits the amount of warping applied to only three directions (0,1), (1,0) and (1,1).

Given any arbitrary warping path P, the total cost associated with the warping path and the two trajectories in question can be defined:

$$c_p(R, T) := \sum_{l=1}^{L} c(r_{n_l}, t_{m_l}) \tag{2.5}$$

With these definitions in place, we can formulate the objective of DTW warping. The optimal warping path determined through DTW aims to minimize the total cost $c_p(R, T)$.

$$p^*_{DTW} := \arg\min \{c_p(X, Y)\} \tag{2.6}$$

It is not hard to see that it would be very difficult to find p^* for long trajectories since the search space would be of dimension $\mathbb{R}^{N \times M}$. An efficient procedure to solve this problem in polynomial time $O(M \cdot N)$ based on dynamic programming has been devised.

From the local cost matrix $C(R,T)$, a cumulative cost matrix can be defined as:

$$D\left(n,m\right) := DTW\left(R\left(1:n\right), Y\left(1:m\right)\right) \tag{2.7}$$

where each element in the matrix D represents the cumulative cost accrued arriving at (m,n) from the starting point $(1,1)$ by taking the *optimal* path subjected to the three warping constraints.

The elements of the matrix D can be filled out efficiently using the following relationship:

$$\begin{aligned} D\left(n,1\right) &= \sum_{k=1}^{n} c\left(r_k, t_1\right), n \in [1:N] \\ D\left(1,m\right) &= \sum_{k=1}^{m} c\left(r_1, t_k\right), m \in [1:M] \\ D\left(n,m\right) &= \min\left\{D\left(n-1,m-1\right), D\left(n-1,m\right), D\left(n,m-1\right)\right\} + c\left(r_n, t_m\right) \end{aligned} \tag{2.8}$$

Mathematical proof on the optimality on the cumulative cost matrix D can be shown using dynamic programming principles [3]. They are not provided here for brevity.

To perform DTW, the cumulative cost matrix D can be first calculated using rules shown in Eq. 2.8. It takes $O(M \cdot N)$ iterations to populate the entire cumulative cost matrix. After the D matrix is populated, the optimal warping trajectory p^*_{DTW} can be found by back-stepping from the end (N,M) back to the starting point $(1,1)$ using the following rule.

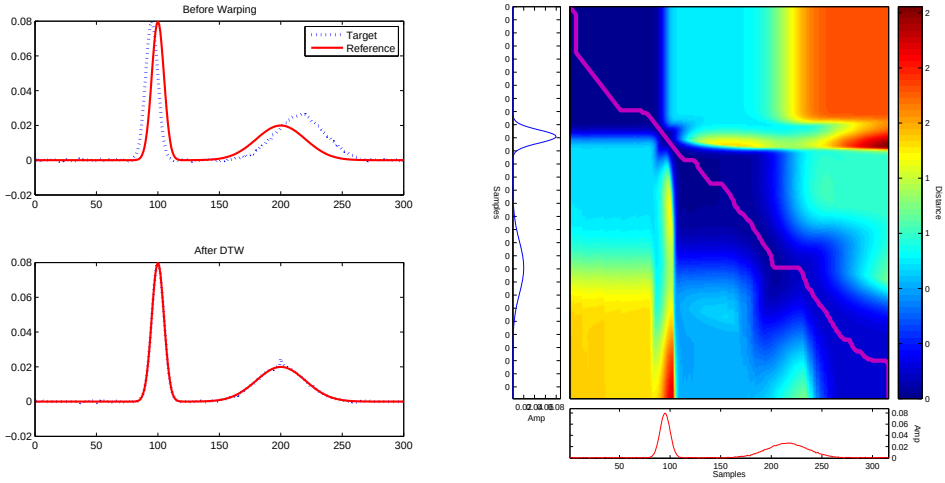
$$p_{l-1} := \begin{cases} (1, m-1) & \text{, if n} = 1 \\ (n-1, 1) & \text{, if m} = 1 \\ \arg\min\left\{D\left(n-1,m-1\right), D\left(n-1,m\right), D\left(n,m-1\right)\right\} & \text{, otherwise} \end{cases} \tag{2.9}$$

Since DTW generates a symmetric warping path (meaning that both the target and the reference trajectory will be warped to give us the best alignment), it is not suitable for process data alignment. For process data alignment, the reference trajectory is usually selected from a set of "golden" batches that do not need to be distorted; thus, only the trajectories to be aligned (target) require warping. A symmetric warping path can be converted to an asymmetric one by applying the following rule:

$$\begin{aligned} R_{asym} &= R \\ t_i^{asym} &= \text{median}\left(t_j^{asym}\right), j \in (p_j | p_r = r_i) \end{aligned} \tag{2.10}$$

Equation 2.10 indicates that if there is a one-to-many correspondence from the points on the reference to the target trajectory, then the median of the target trajectory values will be taken as the reference point value. The length and shape of the reference trajectory itself remains unchanged.

Figure 2.5 shows a result of the asymmetric DTW algorithm on a simple two-trajectory warping problem. Figure 2.5(a) shows that the target trajectory has been warped to conform the reference. In this case, the information loss is apparent in the second peak. The higher amplitude of the target trajectory was reduced to a single "blip" at the maximum point of the second peak, which could potentially be filtered out as an outlier in subsequent analysis. Figure 2.5(b) shows the cumulative cost matrix D for this problem. The darker and more blue color indicates areas of lower cumulative cost. The purple line indicates the optimal path which results in the lowest cumulative cost as defined in Equation 2.5.



(a) Trajectories from before (top) and after (bottom) warping

(b) The cumulative distance matrix D (heat map) and the optimal path (purple line)

Figure 2.5: Trajectory plot and diagnostics from before and after DTW

2.2.3.2 Step 2. Subsequence marker identification

Under the assumption that a series of identified alignment markers from Step 1 in Figure 2.4 has been obtained. Step 2 is then to locate the matching alignment markers on the target trajectory. This requires a fuzzy search of the alignment markers in the target trajectory. Figure 2.6 illustrates the definition of the problem graphically. Segments marker 1 and 2 shown in the figure (the peak and the step change) are first identified in Step 1 either through process knowledge or statistical analysis (will be discussed later). Using these segments, a fuzzy search on the target segment is then performed. Because the reference trajectory segments are most likely to be shorter than the target trajectory to be aligned, this alignment marker search step is called subsequence marker identification.

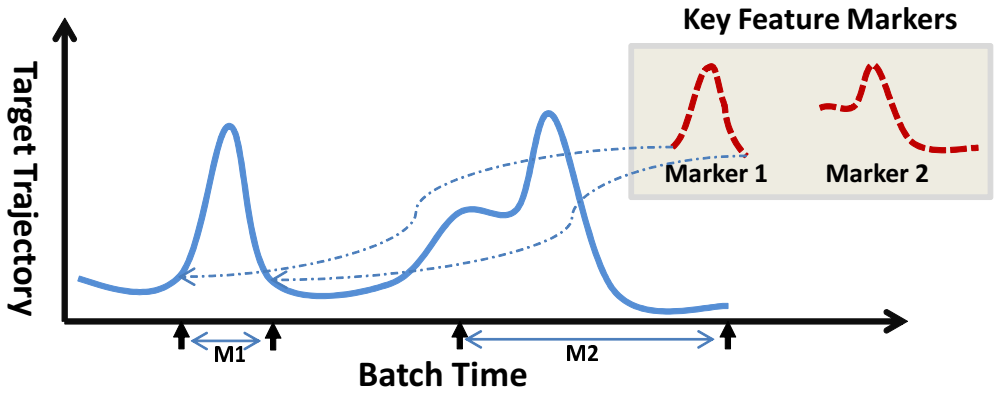


Figure 2.6: Alignment markers (key-feature markers) being assigned to location M1 and M2 on the target trajectory

Fortunately, identifying similar subsequences from another longer source has been a recurring problem in many fields such as speech detection, image processing and DNA-signature matching [3]. As a result, there has been plenty of research in identifying subsequences. One of the solution involves modifying the original DTW formulation and can be solved with high efficiency. The details of this algorithm based on DTW is discussed here.

Define reference alignment marker segment $R = (r_1, r_2, \ldots, r_N)$ and target trajectory $T = (t_1, t_2, \ldots, t_M)$ where $N \ll M$. The goal is to find $T(a^* : b^*) := (t_{a*}, t_{a*+1}, \ldots, t_{b*})$ with $a^* < b^*$ and a^*,$b^* \in [1, M]$.

Using the previous notation introduced, this can be written as a DTW problem as follows:

$$(a^*, b^*) := \underset{(a,b)\in[1,M],a<b}{\operatorname{argmin}} \left(DTW\left(R, T\left(a : b\right)\right)\right) \tag{2.11}$$

The difference between this problem and the original DTW is that we now have additional two degrees of freedom on the boundary of the target

trajectory (a and b). In other words, since the segment R is a continuous vector, *DTW optimization constraints can be relaxed to allow for these additional degrees of freedom.*

To relax DTW constraints, alignment costs for the mismatch at the beginning and the end should be eliminated. This can be done in two phases. First, the boundary condition for the accumulated cost matrix D at the end of the trajectory is modified according to the following rule:

$$\begin{aligned} D(n,1) &:= \sum_{k=1}^{n} c(r_k, t_1), n \in [1:N] \\ D(1,m) &:= c(r_1, t_m) \end{aligned} \tag{2.12}$$

Recall in Equation 2.8, $D(1,m)$ is defined as $\sum_{k=1}^{m} c(r_1, t_k), m \in [1:M]$. The new D definition essentially omits cost incurred by misalignment at the end of DTW. This allows b to vary freely. The rest of the elements in D can still be populated using the same update rule in Equation 2.8. As a result, the optimal boundary location b^* can be found by solving:

$$b^* = \underset{b \in [1:M]}{\operatorname{argmin}} D(N, b) \tag{2.13}$$

After the right boundary b^* has been found, target trajectory can be truncated T to $T' = (t_1, t_2, \ldots, t_{b*})$ since the segment between $[b^*, M]$ do not contain information of interest. To find left boundary a^*, the optimal warping path between the segment R and the truncated trajectory T' need to be computed:

$$p^{a*} = \underset{p}{\operatorname{argmin}} DTW(R, T(1:b^*)) \tag{2.14}$$

To improve computational efficiency, the matrix D found in the first step of subsequence marker identification can be re-used here. Since the only

difference in D occurs during $m \in [b*, M]$, re-computation of D is not required. Let the optimal warping path p^{a*} be $(p_1, p_2, \ldots, p_l)$. This problem can be solved by back-stepping from the end by setting p_l to (N, b^*) and following update rules given in Equation 2.9.

The left boundary a^* of the segment is then the largest index value i that satisfies $p_i^{a*} = (i, 1)$. Figure 2.7 clarifies how the location of a^* is determined. Note that the warping path p has been plotted as connecting lines between the two trajectories. The value of a^* is the largest index before the warping coordinates moves onto the second point (when the warping vector starts showing $(\ldots, 2)$ in the target segment).

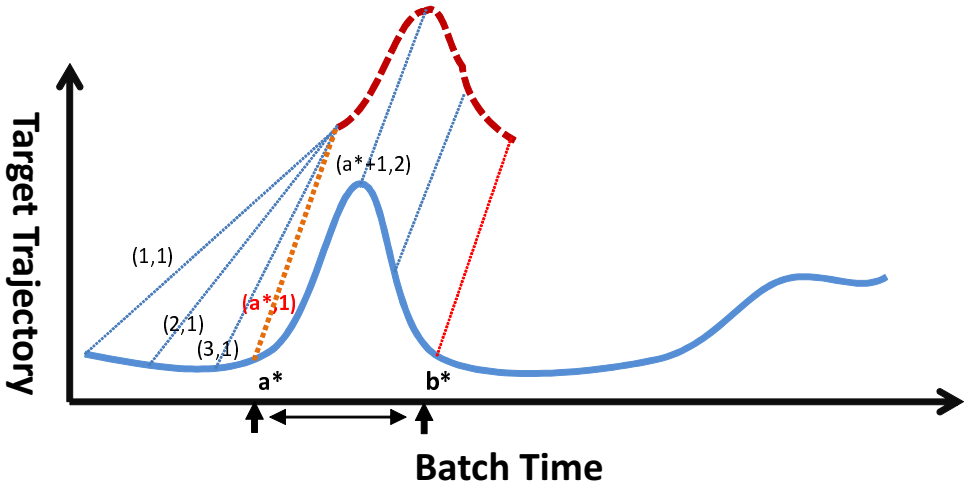


Figure 2.7: Determining the marker boundary a^* in subsequence marker identification

To modularize and simplify subsequent discussion of CsDTW, the subsequence marker identification algorithm introduced in this section can be compressed into a function:

$$(a, b, \Delta) = \textbf{subsequenceDTW}(R, T) \tag{2.15}$$

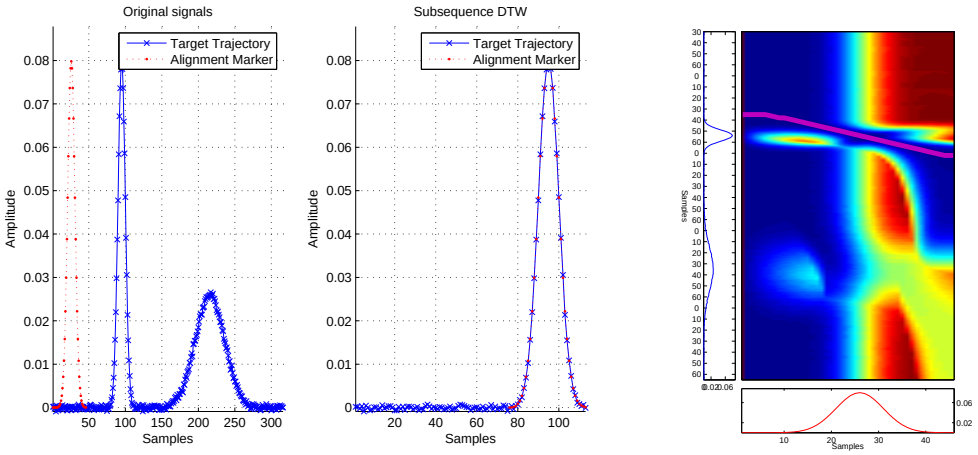
where given a shorter feature segment R and a longer full length trajectory T, a and b are the boundary indices for T that gives us minimal cumulative difference in distance, Δ, between R and $T(a:b)$. The cumulative distance Δ is not used in later alignment steps, but it can be kept as a diagnostic indicator for abnormal batches or runs.

Figure 2.8 shows the subsequence marker identification result of a simulated case study. On the left figure, both the alignment feature (sharp peak highlighted in red) and the the target trajectory (blue dotted line) are shown. In the middle figure, the red alignment marker has been shifted to the location identified through **subsequenceDTW**, which can be visually confirmed to be the correct location. Figure 2.8(b) shows the cumulative cost matrix of the **subsequenceDTW** solution. The thick purple line denotes the trajectory of optimal alignment after solving the DTW problem. It should be noted that the starting point and the ending point of the optimal path do not occur at the diagonal vertices of the matrix, which means that the boundary condition constraint for DTW has been removed and allows for free movement of the alignment marker segment.

2.2.3.3 Step 3. Generating constraints for feature preservation

After Step 2 is completed, segments of target trajectory that are matched against the alignment markers can now be located by their boundary indices (list of as and bs). Since these features should not be distorted during alignment, global bandwidth constraints are formulated to prevent warping during these segments of the trajectory.

Two types of constraints exist in a DTW problem. The first kind is called local or slope constraint, they limit the range of allowed slope for



(a) Alignment markers superimposed on target trajectory (b) Cumulative cost matrix

Figure 2.8: Example result of subsequence marker identification using an feature segment on target trajectory

the optimal path (see Figure 2.9). The second type of constraint is called global or band constraints. These constraints are defined globally by declaring certain zones in the cost matrix D to be illegal during the back-stepping path search. Two popular bandwidth constraints in the DTW literature are shown in Figure 2.10.

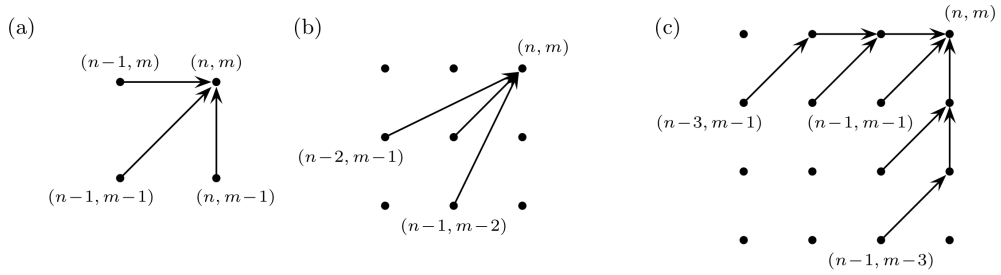


Figure 2.9: Three local/slope constraints typically seen in DTW. (a) No slope constraint, (b) Slope constraint of $(0.5, 2)$,(c) slope constraint of $(1/3, 3)$ [3].

To restrict warping for segments that were identified as alignment mark-

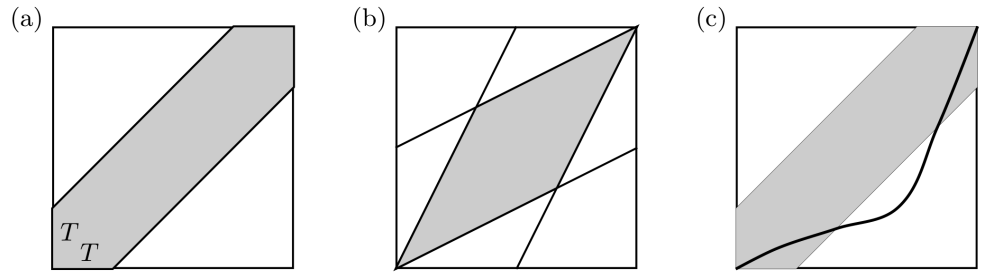


Figure 2.10: Three global constraints: (a) Sakoe-Chiba band with width T, (b) Itakura constraint,(c) path p^* in violation of the Sakoe-Chiba constraint [3].

ers, global constraints can be defined around these key features to restrict the warping path to only proceed diagonally (meaning that there will be no distortion). Global constraints can be defined by declaring the local cost of the illegal zones as infinite:

$$c\,(n,m) := \infty \text{ for } (n,m) \in \text{illegal zone} \tag{2.16}$$

To restrict the warping path to only move diagonally across the region mapping key alignment features, the following modification to the local cost matrix is applied:

$$\begin{array}{l} c\,(n,m) := \infty \text{ for } n \in [\alpha,\beta]\,, m \in [\mathbf{a}^*, \mathbf{b}^*] \\ c\,(n,m) := 1 \text{ for } (\mathrm{n},\mathrm{m}) \text{ in } (\alpha : \mathbf{a}^*, \beta : \mathbf{b}^*) \end{array} \tag{2.17}$$

where α and β are alignment marker bounds on the reference trajectory R and $\mathbf{a}^*$ and $\mathbf{b}^*$ are the subsequence found in target trajectory T. The updated local cost matrix is denoted $C_{\text{constraint}}$ to distinguish it from the original local cost matrix. Making the element of a local cost matrix infinity essentially will remove this element from the backtrack searching of the optimal warping path. Figure 2.11 shows an example of the global constraint generated using

information from subsequence marker identification. The two diagonal lines indicate areas with key trajectory features that should be preserved during alignment.

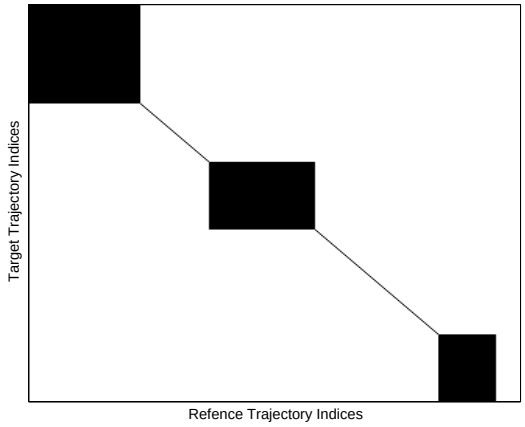


Figure 2.11: Example of the alignment markers being applied as global constraints, Warping is only allowed in the black regions.

2.2.3.4 Step 4. Solving the globally constrained DTW

After completing Steps 1-3, all the pre-requisites for the modified DTW problem are in place. Solution of the modified DTW can be obtained using the same technique as the original DTW problem described in earlier sections. Setting the local cost of the illegal zones to infinity implicitly incorporates the global constraints into the optimization solution. To summarize, detailed algorithm of CsDTW are tabulated here for completeness:

From Table 2.2, the complexity of CsDTW can be shown to be polynomial time. Since each iteration of **subsequenceDTW** function requires executing DTW once, CsDTW will require $(I + 1)$ iterations of the DTW al-

Table 2.2: The CsDTW algorithm for batch trajectory alignment

Step	
1	Given trajectories $R := (r_1, \ldots, r_M)$, $T := (t_1, \ldots, t_N)$, identify key feature marker bounds $(\alpha_i, \beta_i), i \in \mathbb{I}$
2	Calculate a^*,b^* :=**subsequenceDTW**$(R(\alpha_i : \beta_i), T)$ for each segment identified in Step 1.
3	Formulate the global constraint matrix according to Equation 2.17
4	Solve the globally constrained DTW problem for optimal path $p^* = DTW(R, T, C_{\text{constraint}})$
5	Apply asymmetric conversion from warp path p^* using Equation 2.10 to obtain final target trajectory

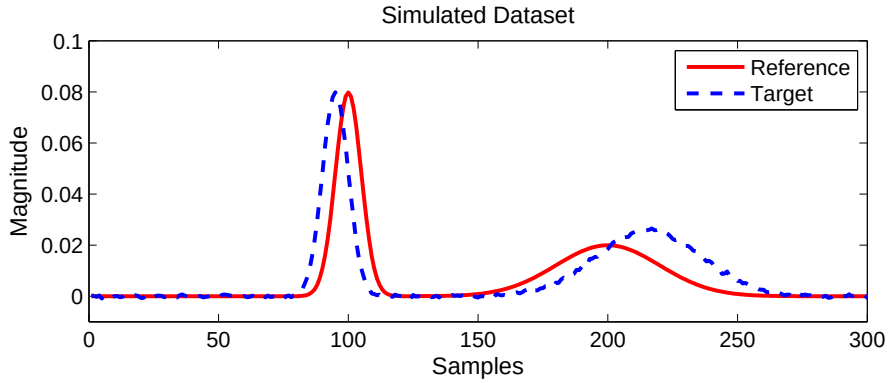
gorithm to complete, where I is the number of feature segments identified in Step 1. As a result, the total complexity of the algorithm is $O(I \cdot M \cdot N)$. In practice, since the number of alignment markers identified in a batch process trajectory typically do not exceed five, the proposed algorithm can be solved just as efficiently as any DTW alignment problem. This efficiency is gained largely through the use of dynamic programming principles in improving the efficiency of searching for the optimal path.

2.2.3.5 Performance comparison with DTW, COW, truncation and interpolation

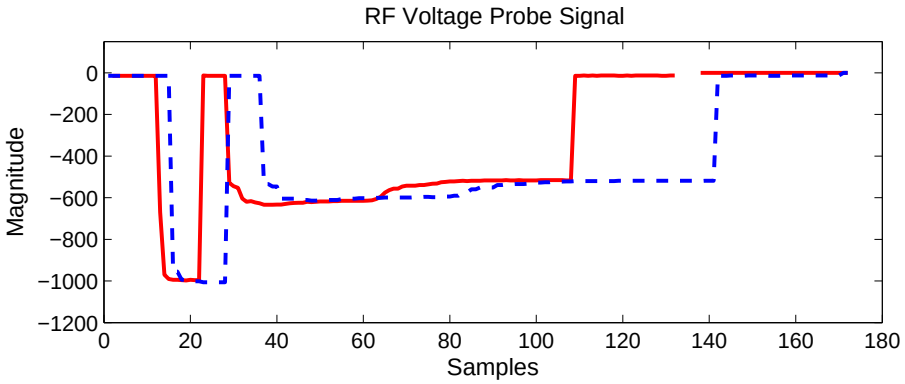
To compare the alignment performance of CsDTW method, all of the alignment methods were tested on a simulated dataset and an actual batch dataset from the semiconductor industry.

The five methods being tested are: simple truncation, LTS (interpolation), DTW, COW and CsDTW. The method for DTW was based on the algorithm described in [37]. COW was performed using MATLAB library provided by [40].

Figure 2.12 shows two sets of trajectories to be aligned. The reference trajectories are solid red lines and the target trajectories to be warped are in blue dashed lines.



(a) Simulated Trajectory



(b) Batch Process Trajectory

Figure 2.12: Reference and target trajectories to be aligned

Figure 2.13(a) shows the comparison results of the five different alignment methods on a set of simulated trajectories.

For the first subplot (simple truncation), both the head truncation and the tail truncation cases are plotted for completeness. In both cases, the truncation only enforced the trajectory to be of same length and did not

result in improvement of the alignment of features. In fact, the head-truncation made the resulting trajectory (green dotted line) even further away from the reference trajectory, leading to information distortion.

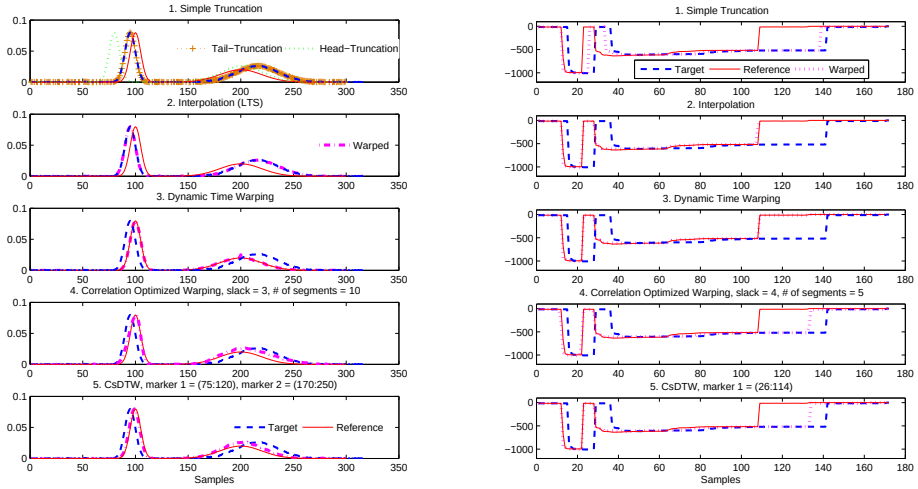
In the interpolation case (2nd subplot), since an indicator variable is unavailable, the batch lengths are simply "compressed" to be of same length as the reference trajectory. The resulting warped trajectory still follows the original target trajectory closely, but the alignment with the reference trajectory did not improve.

In the third case, the DTW warped trajectory is identical to the reference trajectory with the exception of a peak at sample number 200. While the alignment is highly successful, this outcome is undesirable since the between-trajectory information has been lost completely during alignment.

In the fourth case, the COW warped trajectory behaves much better than the previous three candidates, the peaks are aligned at the right location and the trajectory length is now synchronized. However, upon closer inspection, the first peak (sample 80-120) in the warped trajectory exhibits a slightly heavier left tail than the target trajectory, indicating that COW has introduced some slight distortion. In addition, the parameters for COW has been determined through trial and error to achieve the best optimal alignment, which would be a strenuous tuning step for alignment of larger data sets.

In the last case, the alignment markers from segment 1 (75:120) and segment 2 (170:250) are used. The purposed method was able to achieve good alignment with minimal distortion. To show the difference between the CsDTW versus the traditional DTW formulation, Figure 2.14 shows the cumulative cost matrix comparison and the optimal path (purple line) of both methods. In both graphs, lower values are represented using colder colors and

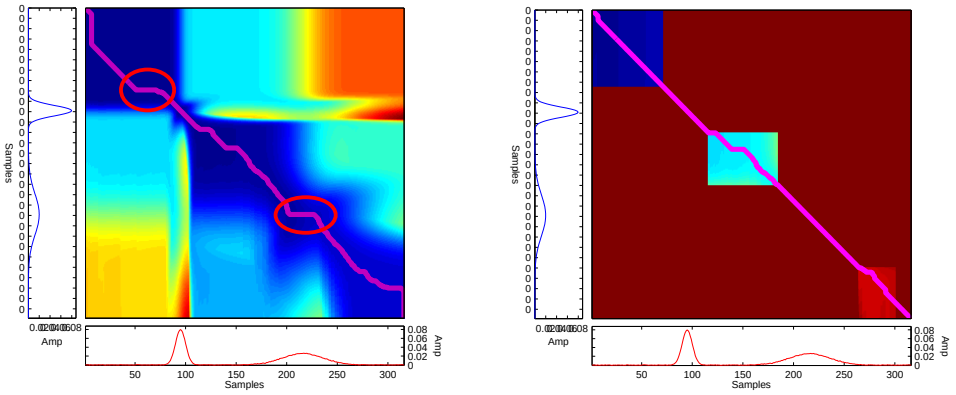
higher values in warmer colors. In Figure 2.14(b), the dark red colored indicate the enforced global constraint zones. The distinctively colored rectangles are regions where the trajectories are allowed to warp. In Figure 2.14(a), there are no restrictions on the warping path; consequently, there are two cases of target trajectory being compressed into a single point (highlighted in red circles). These horizontal lines in the warping path indicate that there are information loss and compression of trajectory taking place; these behaviors are usually undesirable.



(a) Simulated trajectories (b) Voltage probe data

Figure 2.13: Comparison of the five warping methods on (a) simulated trajectories, (b) voltage probe data

Figure 2.13(b) shows the alignment results of the five warping methods using voltage probe data from a batch process. Since the trajectories are zero padded at the beginning and the end to ensure that the entire batch is of same length. Truncation and interpolation alignment will use the first non-



(a) DTW cumulative cost matrix D

(b) Proposed method cumulative cost matrix D

Figure 2.14: Warping alignment diagnostics of DTW versus CsDTW

zero values in the trajectory to align the data. Their results are shown in subplots 1 and 2. Truncation was only able to align the first step change. For interpolation, the target trajectory was compressed according to the reference trajectory and resulted in very good alignment. However, the length of the trajectory has been shortened (the warped trajectory was cut off at 118). The DTW result (in subplot 3) without any constraint also resulted in a similar trajectory profile as interpolation. The last two subplots showing both the COW and the CsDTW gave good alignment results without any changes in the duration of the warped trajectory length. For CsDTW, the alignment marker used was segment (26:114) from the reference trajectory since the voltage probe measures the duration of the plasma chamber activity during this period. Likewise in the simulated case study, a trial and error procedure was performed to estimate the slack and the number of segments for the COW algorithm to optimize for the best alignment results.

In both cases, CsDTW was able to outperform truncation, LTS (or

interpolation) and DTW alignment, while also achieving similar alignment results when compared with the COW method. However, CsDTW excels in computational speed since it only solves a constrained DTW problem and does not require repeated evaluation of the correlation coefficients. In addition, CsDTW offers modeling practitioners an intuitive way to specify the method tuning parameter (just identify key feature markers that should be preserved) and does not require trial and error to find the optimal tuning parameters.

2.2.4 Extensions of CsDTW

Optional extensions of CsDTW will be discussed. These extensions are not required for the main algorithm but can be implemented to allow CsDTW to deal with several special scenarios.

2.2.4.1 Alignment marker selection

It is common that process knowledge or expertise are not available to the model developer. Since CsDTW requires identifying the key trajectory features that need to be preserved, some preliminary analysis can be performed beforehand to help determine the regions of interest. Here two criteria used in assisting identification of key alignment markers will be introduced.

Inter-batch vs Intra-batch variance ratio: Calculating the variance ratio along a batch trajectory is a quick way to filter out non-informative segments in a batch trajectory. The variance ratio is a ratio of the inter-batch variance versus the intra-batch variation at different time segments. The trajectory will be first split into N equal-length segments. For each segment, the inter-batch variance and the inter-batch variance can then be calculated. For inter-batch variance, the average value of each segment is used in calculating the variance. For intra-batch variance, the variance of each segment is first calculated and then averaged. A ratio value of lower than 2 generally indicates that particular segment is not useful for modeling and can be filtered out. The remaining segments can then be plotted and used as feature alignment markers for CsDTW. When there are multiple sensor readings, the variance ratio vector will become a matrix of dimension $\mathbb{R}^{N_{\text{segs}} \times N_{\text{variables}}}$. The correct trajectory alignment markers can be based on the common regions of high variance ratio value.

Variable importance in projection (VIP): A preliminary multiway PLS model can be constructed by first synchronizing the trajectories using trivial alignment methods such as truncation or interpolation. From this preliminary PLS model, the variable- importance-in-projection (VIP) of each variable can be calculated. The VIP criteria is commonly used in PLS model development to filter out uninformative variables to simplify the resulting model. In loose terms, the VIP value is a measure of the contribution of each variable in the identified overall correlation of the model. Section 2.3 discusses the details of VIP and PLS model variable selection. For a batch dataset of dimension $X \in \mathbb{R}^{I \times J \times K}$, where I is batch number, J is variable, and K is the sample time, the VIP vector for the multiway model will be of dimension $\mathbb{R}^{1 \times J \cdot K}$. To improve visualization, the VIP values of each multiway variable can also be folded back into their original two-dimensions and becomes a VIP matrix:

$$VIP_{1d} \in \mathbb{R}^{1 \times J \cdot K} \xrightarrow{refold} VIP_{2d} \in \mathbb{R}^{J \times K} \tag{2.18}$$

Figure 2.15 shows an example VIP matrix heat map plot for a semiconductor process. The red colored cells indicate regions of higher VIP. From this figure, it is apparent that three segments (durations 20-40, 70-80, and 90-100) exhibited the most significant correlation. These segments of the batch trajectory can then be verified visually and used in the CsDTW algorithm for better trajectory alignment.

However, since performing alignment will alter the structure of the original training dataset X, this procedure might need to be carried out iteratively until convergence. Convergence is defined by checking for changes in the alignment markers selected. In practice, the number of iterations required to achieve convergence has never exceeded five. Figure 2.16 shows the flowchart of iterative process.

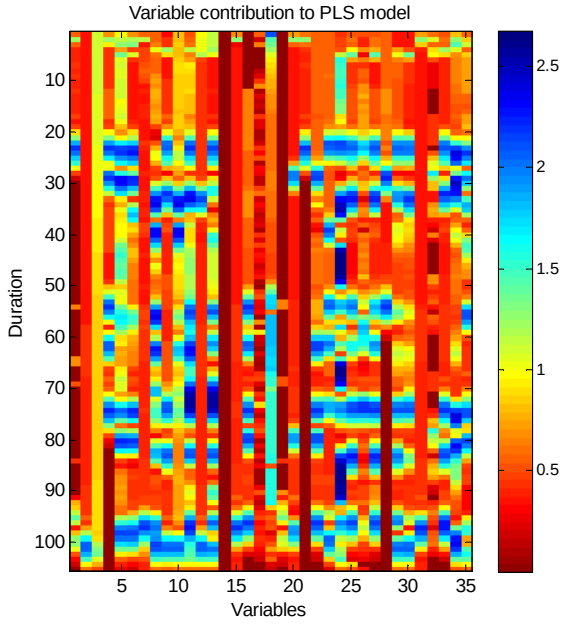


Figure 2.15: Re-folded 2 dimensional heatmap of a VIP matrix from a multi-way PLS model

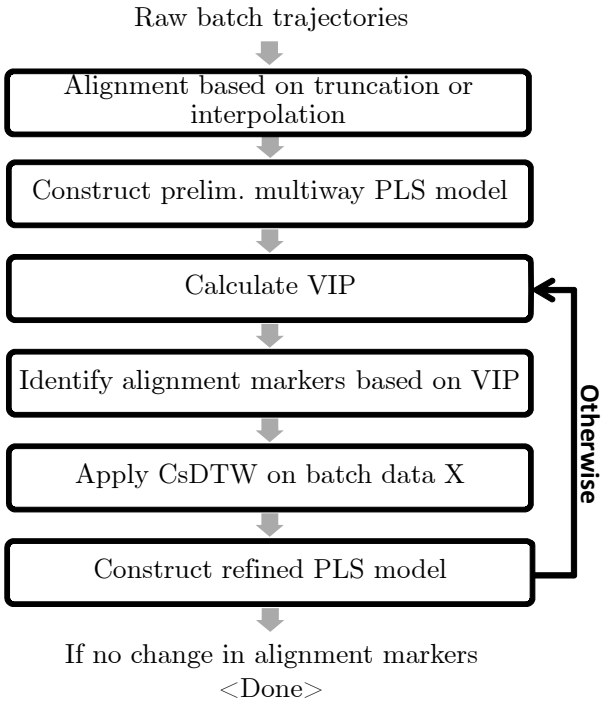


Figure 2.16: Flowchart for iterative marker selection and CsDTW alignment

2.2.4.2 Local constraints and derivative filters for noisy data

DTW based method has been known to suffer from artifacts and distortions in aligning trajectories with excessive measurement noise [41]. As a result, multiple work has been published on dealing with noisy trajectories by taking the derivative of the time series being aligned, or applying filtering to the trajectory derivatives [41, 40, 42]. Since the CsDTW formulation is based on DTW and solves the same optimization problem, these methods can be easily extended to CsDTW as well. Here the robust derivative dynamic time warping (RDDTW) method will be briefly introduced [41].

In RDDTW, the algorithm applies a Savitzky-Golay filter to the derivative of the trajectories before they are aligned using DTW. The SG filter is widely used filter that combines polynomial regression with moving-window [43]. The smoothing strength can be adjusted by changing the window size and the degree of the polynomial. To carry out "robust derivative constrained selective dynamic time warping (RD-CsDTW)", one needs to calculate the derivative of the trajectory and then apply SG filter prior to executing the CsDTW algorithm.

In addition, since CsDTW utilizes global constraint to preserve alignment features in the trajectory. The global constraint will sometimes conflict with band constraints (a type of global constraint) that are commonly used in DTW. Although this scenario is extremely rare as CsDTW is designed to avoid excessive trajectory distortion, local constraints can be implemented in a similar fashion like regular DTW. Figure 2.9 shows three types of local constraints commonly seen in DTW. These local constraints can be implemented by modifying Equation 2.8 to restrict the search space of the optimal path, additional details are not discussed here and can be found in literature [3].

2.3 Variable Selection for PLS Models

2.3.1 Introduction

Data driven soft sensors apply multivariate statistics and machine learning techniques to find empirical correlations between process variables and quality variables. Partial Least Squares (PLS) and Principal Component Analysis (PCA) are two popular techniques for finding these correlations. There are many publications related to the application of these methods for soft sensors and multivariate monitoring [44, 45, 46, 47]. More recently, reduced order dynamic PLS based soft sensors have been developed for the monitoring of processes experiencing large transport delays [19]. PLS regression in particular is suited to deal with high-dimensional data in the presence of collinearity. In practice, performance of the regression models could often be improved when a subset of highly relevant variables is used instead of the whole training dataset [48]. The reduced models are more resilient to measurement noise and are often more interpretable. Thus, a successful variable selection procedure will improve the interpretation and identification of the underlying process conditions.

The popularity of PLS methods has also generated interest in PLS variable selection techniques. Mehmood et al. [49] showed that the number of publications in the field of PLS modeling and related methods has increased exponentially since 1988. The field of application outlined in this review ranged from gene selection data to gene expression data, quantitative structure-activity relationships (QSAR) descriptors selection, spectroscopy wavelength selection, and bio-marker selection. Kalivas and Sutter provided another review of variable selection in the field of QSAR descriptor selection [50]. Saeys et al. reviewed popular selection techniques in the field of bioin-

formatics [51]. While being quite comprehensive in the methods reviewed, these reviews did not cover PLS variable selection in the context of industrial process data that involves multiple operating modes. Since multivariate statistical models are heavily influenced by the properties of the underlying data, the variable selection methods suited for bio-marker selection are likely to be inappropriate for process variable selection. As a result, the goal of this section is to evaluate the existing methods for variable selection of industrial process data, and to present an improved variable selection method appropriate for industrial processes with multiple operating modes.

The structure of this section is organized as follows. First, we evaluate existing variable selection methods; several representative techniques will be implemented and assessed. Second, we will present our improvements to the current methods to address their shortfalls. Last, we will present evaluation results using a set of model-free criteria that help in the assessment of variable selection performance.

2.3.2 Background

2.3.2.1 Data-driven PLS and variable selection methods

An in-depth introduction of PLS methods is available in [52, 53, 36], and thus these methods are not discussed in detail here. A brief introduction of these multivariate methods are also provided in the Appendix B of this dissertation. The soft sensor model development process typically consists of data gathering, pre-processing, variable selection, model development, and model validation. Implementation details of each step will vary depending on the specific application. Kaldec and Sliskovic provided comprehensive reviews of the soft sensor development process for interested readers [11, 54]. For industrial

processes, the model development work flow can be summarized in Figure 2.17. Defining the proper model scope, applying the right pre-processing steps and performing model validation are critical steps in addition to variable selection. Expert knowledge and first principles-based understanding of the process are useful aids in pre-screening variables, transforming nonlinear variables and validating models. Soft sensor modeling practitioners should maximize process knowledge integration to give physical significance to the resulting PLS models.

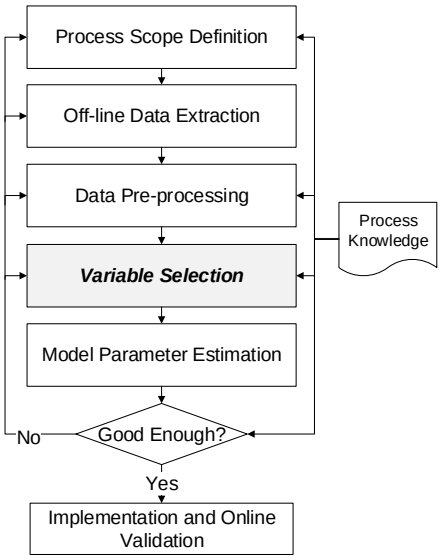


Figure 2.17: Illustration of a typical soft sensor model development process

There are many existing variable selection methods specific to PLS regression. A general observation made from this body of research is that most variable selection methods calculate a variable importance ranking metric and then apply this metric in the subsequent steps to find the optimal variables. Mehmood et al. and Saeys et al. suggested categorizing these methods based

on the mechanism of variable ranking and selection into three types - filter, wrapper, and embedded [49, 51]. Figure 2.18 graphically depicts the major differences among these three types of selection methods.

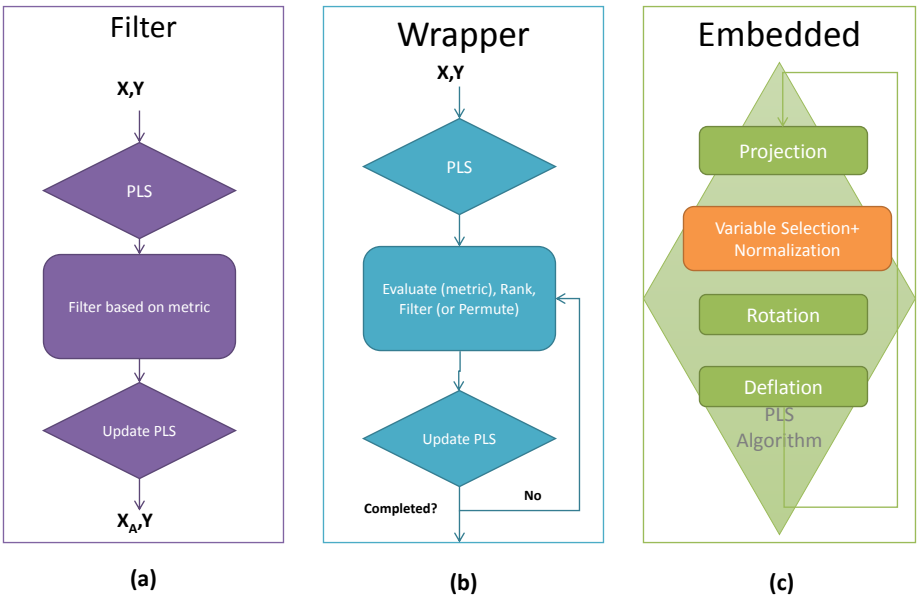


Figure 2.18: Three categories of PLS variable selection methods: (a) filter, (b) wrapper and (c) embedded

2.3.2.2 Selection method categorization

Filter methods, as the name indicates, will first calculate the variable importance ranking criteria and then apply a filtering rule, such as threshold cut-off, to remove the unwanted variables. The threshold calculation and variable importance ranking metrics utilize various filtering methods. The most popular metrics used are Variable Importance in PLS Projection (VIP), out-

put correlation coefficient, and normalized beta coefficients. Filtering based methods are the most common methods used in process chemometrics. Extensions of the filtering methods include subsets selection by Hoskuldsson [55] and interval PLS by Norgaard [56], where intervals or subsets of variables are used instead of individual variables. Filtering methods are implemented in most commercial modeling software packages such as SIMCA, ProMV or PLS-Toolbox; these methods are often preferred for their simplicity.

Wrapper methods and embedded methods are more complex than filter methods. The PLS algorithm itself remains unchanged in wrapper methods. Instead, wrapper methods focus on optimizing a quality criteria, such as the prediction performance, by adjusting the variable selection process. The optimization routine is often iterative and requires repeated application of PLS algorithms. Specific examples of this class of variable selection techniques are stepwise selection (forward and backward) [57, 58], Uninformative Variable Elimination (UVE-PLS) [59], Subwindow Permutation Analysis (SwPA) [60], Genetic Algorithms [61, 62, 63] variable selection, and Competitive Adaptive Resampling (CARS) [64]. The main benefit of using wrapper methods is that more information can be extracted from the variable selection process. Most methods in this category can provide suggestions toward the optimal number of variables. However, wrapper methods are prone to the over-fitting of training data. In addition, it is often more difficult to interpret exactly why the variables are included in the final model without going through a strenuous debugging process.

Lastly, embedded methods perform variable selection directly inside the PLS algorithm by changing the PLS loading weight vectors ($\mathbf{w}$). Examples of methods belonging to this class are sparse PLS (sPLS)[65], Iterative Predictor

Weighting (IPW-PLS)[66], powered PLS (PPLS)[67], and Interactive Variable Selection (IVS)[68]. The main difference between these methods lies in the way that the weight vectors are adjusted. The benefit of embedded methods is their computation speed since variable selection and PLS estimation are performed concurrently. However, a practical issue that was frequently encountered is in the tuning of variable selection behavior. The effect of the tuning parameter (if there is one) on the final variable selection is often unclear and behaves unpredictably, thus making it difficult to apply embedded methods without prior knowledge of the expected variable set.

2.3.3 Overview of current methods

Table 2.3 summarizes the selection metrics, tuning parameters, main advantages and disadvantages of the representative methods from each class. These methods are chosen based on their popularity in literature citations and also their ease of implementation for large data sets. Each method was implemented in MATLAB and tested on an industrial data set. For simplicity, we picked one method from each class for further analysis and summarized the other potential candidates in Table 2.3. Readers can refer to the original articles on these methods for a more comprehensive introduction to the methods and their properties.

2.3.3.1 Filtering method - VIP filtering

Variable importance (or influence) in projection is introduced by Wold to account for the importance of variables in 3D QSAR drug design [36, 69]. This metric has also been used extensively in process chemometrics. The VIP calculation is straightforward once the loading, weights and the scores of the

Table 2.3: Critical overview and assessment of several selected representative variable selection methods in filtering, wrapper and embedded categories

Method Name	Selection Metric	Tuning Parameter	Advantages	Disadvantages
VIP Filtering by Wold et al.	VIP	None	- Simple, results are interpretable, accounts for influence of input variables for both explaining X and Y	- Only a ranking, requires judgment and further analysis - Collinear variables will bias ranking
Beta coefficient filtering	Beta coefficient	None	- Simple, results are interpretable. Can be used in conjunction with regression parameter significance testing	- Same as above - Subjected to scaling issues as well
Subwindow Permutation Analysis (SwPA) by Li et al	Rank-sum P scores	Number of sampling loops, number of sampled variables, percent testing data	- Modeling method independent - Statistical testing based, does not utilize existing weightings or importance rankings native to VIP - Provides graphical visualization of the selection process	- Does not return an optimal set of variables - Effects of tuning parameters unclear - Non-Gaussian residuals (i.e. lots of outliers) will reduce the efficiency of this method
Uninformative Variable Elimination (UVE) by Centner et al.	VIP/Beta coefficient or correlation coefficient	Multiplier to the cut-off threshold reliance level	- Provides a rigorous cut-off criteria for uninformed variables - Simple, does not require iteration or changing the internal PLS algorithm	- Number of variables selected cannot be easily adjusted - Collinear variables will bias the ranking and filtering of other variables
Stepwise Elimination by Frank, Fernandez, and others	VIP or beta coefficient or variable wise-Q^2	Number of variables in the final model	- Simple and interpretable - Tuning parameter directly relevant to the end model - Parameter selection / elimination based on statistical testing	- Can get stuck in local optimal - Requires iterative evaluation of cross-validated performance metrics - Still relies on the same variable ranking metrics, so marginal performance improvement
Competitive Adaptive Re-weighted Sampling by Li et al.	PLS direction weights (w) and Q2 from different sample runs	Number of sample loops, number of re-runs	- Gives recommended set of variables - Visualizes the performance improvement against other possible models	- inconsistent selection results due to random sampling - random search based, difficult to interpret or justify selected variables - Prone to over-fitting the training data
Sparse PLS (sPLS) by Chun and Keles	Weight (w) in PLS loop	Eta (sparsity multiplier used in determining threshold weights)	- Has good theoretical foundation - Tuning by adjusting the sparsity parameter - Requires no iteration - Provides a definitive set of optimal parameters	- Prone to collinearity problems in selected variables - Sparsity tuning parameter is nonlinear - Difficult to select the right combination of PLS latent components and sparsity parameter
Interactive Variable Selection (IVS) By Lindgren at al.	Transformed PLS direction weights (w)	None (tuning parameter is optimized by the algorithm)	- Independent variable selection along each principal component direction - Provides a more rigorous way to define the threshold parameter	- inner cross-validation loop not well justified and explained - each component will have separate set of variables, leading to difficulties in interpreting scores and loadings of the resulting PLS model
Powered PLS (pPLS) by Indahl	Decomposed PLS direction weights (w)	Bounds for optimization of tuning parameter	- Offers an alternative interpretation of PLS loading weights, decomposes into correlation and variation - Tuning parameter optimized over the specified space to maximize correlation	- Complex tuning needed with confounding tuning parameter effects - Performance is poor compared to other available methods

PLS regression have been determined. The VIP measure for the jth variable in PLS regression can be calculated as follows:

$$VIP_j = \sqrt{p \sum_{a=1}^{A} SS_a / SST (w_{aj} / \|w_a\|)^2} \tag{2.19}$$

where p is the number of variables, A is the total number of components, SS_a is the output variance explained by the ath component, SST is the total output variance, $\frac{w_{aj}}{\|w_a\|^2}$ is calculated using loading weight vectors w_{aj} for each component and represents the importance of the variable j for component a.

In Equation 2.19, The VIP is a cumulative measure of the weight of each variable relative to the other variables. The sum of squares explained by the ath component acts as weights to account for the diminishing predictive power of additional principal components.

To select variables using the VIP, the VIP values are sorted in descending order. A cut-off threshold can then be estimated subjectively based on process knowledge or through iteration to optimize for a certain desired performance criteria (in which case the VIP filtering behaves like a wrapper method). The VIP filtering has been the preferred method in most variable selection scenarios because of its speed and efficiency. The results are also easy to interpret and can be easily decomposed to identify anomalies in loading weights or principal components.

While VIP filtering provides a relatively strong performance in most scenarios, the limitations of the VIP filtering method are as follows. First, the VIP filtering method provides multiple methods to set the variable elimination criteria, but it is unclear which method works best in a given situation; previous experience and subjective judgment are often needed to arrive at a

reasonable answer. Second, filtering variables based on their VIP values could be inefficient under the presence of collinear variables. The collinear variables (for instance, redundant temperature measurements) with high correlation to output will lower the priority of other variables during selection. As a result, pre-processing and post-processing are often required. In our scenario, we first pre-processed the dataset to remove extra collinear variables in the input data. The collinear variables are clustered based on the calculated cross-correlation values among the input data, a threshold of 0.85 was used to group the highly correlated variables together. With the exception of one variable (the variable with the smallest variance), all other variables from each clusters are removed. An iterative study was also done to optimize the cut-off threshold in order to maximize the model cross-validation performance.

2.3.3.2 Wrapper method - Subwindow Permutation Analysis (SwPA)

Subwindow Permutation Analysis (SwPA) is a method proposed for selecting biomarkers by Li et al. [60]. Figure 2.19 illustrates the overall procedure of the SwPA method. The distinctive feature of this method relies on using the ranksum test to test for residual distribution shifts. In order to generate a population of residuals for statistical testing, the SwPA method uses two nested loops. In the outer loop, N Monte-Carlo iterations generates N subsets of original data. Each subset is divided into training and testing portions. A PLS model is then constructed using the training data. The number of components for this PLS model is determined through cross-validation. The un-permuted prediction residual R_k for this model can then be calculated using the testing data. Upon entering the inner loop, each variable x_j from the testing subset k is permuted through random shuffling. The variable-wise

permutation results in a new testing dataset from which permuted prediction residual $R^*_{k,j}$ can be calculated. The permutation process scrambles the variable into a noise series with the same mean and variance. If a variable is important to the PLS model, then the permutation process should result in an increase in the residuals. Following this argument, the significance of each variable can then be ranked based on how much each variable permutation shifts the residual distribution.

The population analysis approach taken in SwPA prevents over-fitting of variables selected to training data. In addition, the ranksum test is a non-parametric test for mean-shift, which means SwPA is suitable for cases where residuals are non-Gaussian. The negative log of the p_j-value (probability of rejecting the null hypothesis) calculated from the test statistic is an indicator of how much the residual distribution was affected by the permutation of variable j. SwPA will favor variables that have a large effect on changing the residual distribution. One disadvantage of the SwPA method is the large number of tuning parameters. The number of Monte-Carlo iterations (N), the number of subset variables sampled Q_v, and the variable size of the final model are all tuning parameters. Consequently, sub-optimal selection of these tuning parameters could negatively affect variable selection results.

In addition, the SwPA ranking was found to be affected by noisy variables, where the ranking of minor variables (except the first three or four dominating variables) are different each time the SwPA algorithm was executed. The SwPA method also does not recommend a cut-off point for the model size. To address these short-falls, we presented a slightly improved iterative version of the SwPA method (SwPAi). In SwPAi, backward elimination of variables are added as an additional loop outside the original SwPA algorithm. The

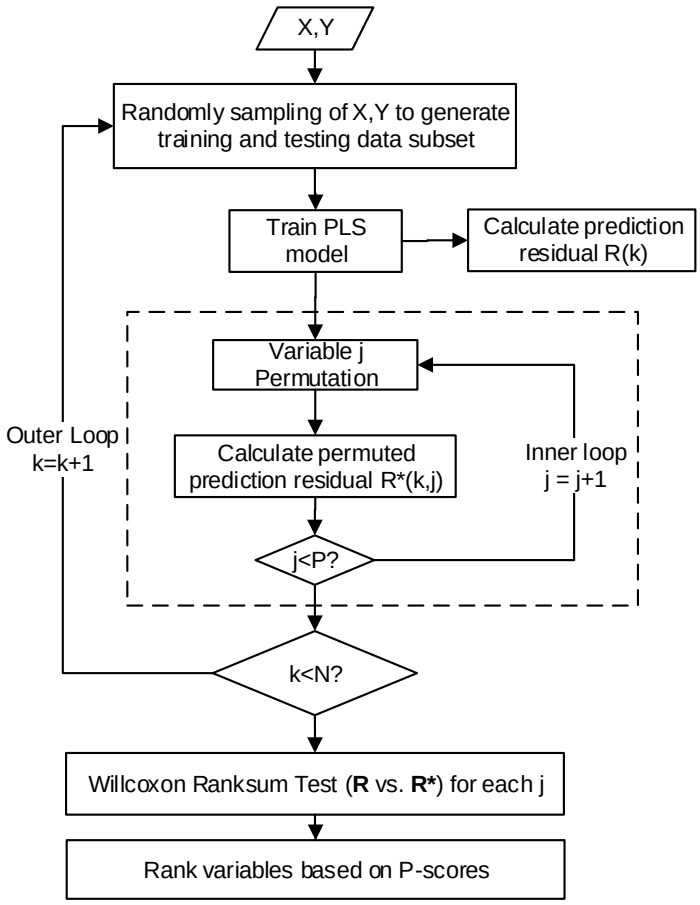


Figure 2.19: Subwindow Permutation Analysis (SwPA) variable selection algorithm overview

least important variable from each iteration is removed in step-wise fashion. Cross-validation is used to determine the cut-off point for the model size. The resulting performance of SwPAi and SwPA are shown in the industrial case study in Chapter4.

2.3.3.3 Embedded method - sparse PLS (sPLS)

Chun and Keles proposed sparse PLS (sPLS) as an embedded method that perform variable selection alongside regression [65]. This method is suitable for problems with more variables than observations ($\mathbf{X} \in \mathbb{R}^{N \times P}, P \gg N$), a common issue in batch data sets. In the Nonlinear Iterative partial least squares algorithm (NIPALS) for PLS regression, the PLS loading weight w is optimized in subsequent iterations to maximize the covariance between the latent scores of $\mathbf{X}$ and $\mathbf{Y}$. This can be formulated as an optimization problem [70]:

$$\begin{aligned} &\mathbf{w}_k = \text{argmax}(\mathbf{w}^T \sigma_{XY} \sigma_{XY}^T \mathbf{w}) \\ &\text{s.t. } \mathbf{w}^T(\mathbf{I}_p - \mathbf{W}_{(k-1)}\mathbf{W}_{(k-1)}^+)\mathbf{w} = 1, \mathbf{w}^T \Sigma_{XX} \mathbf{w}_j = 0 \end{aligned} \tag{2.20}$$

where σ_{XY} is the covariance matrix between $\mathbf{X}$ and $\mathbf{Y}$, w represents the loading weight and W represents the loading weight matrices (of all components) at each iteration k.

The two constraints of Equation 2.20 normalize the length of the loading weight to be equal one and ensure that the loading weights are orthogonal. In sparse PLS, the optimization function for the PLS direction weights (w) is adjusted to include an L-1 norm constraint similar to that of LASSO methods [71] as follows:

$$\begin{aligned} &\mathbf{w}_k = \text{argmax}(\mathbf{w}^T \mathbf{X}^T \mathbf{Y} \mathbf{Y}^T \mathbf{X} \mathbf{w}) \\ &\text{s.t. } \mathbf{w}^T \mathbf{w} = 1, |\mathbf{w}| \leqslant \lambda \end{aligned} \tag{2.21}$$

where λ is a sparsity tuning parameter and $|\mathbf{w}|$ represents the L-1 norm of the loading weight vector. Figure 2.20 shows the details of the sparse PLS algorithm adopted from the original paper. For PLS regression of single outputs, the optimization problem for w has a straightforward solution, where the

loading weights w_i below a threshold are set to 0. Details of the algorithm and proofs of the optimization solution simplification can be found in Chun and Keles [65]. The main advantage of sparse PLS is in its computational speed. In practice, however, the gain in computation speed is offset by the additional processing required to perform two-dimensional cross-validation to determine both the number of components and also the sparsity parameter. The sparsity parameter (0-1) controls the number of variables in the final model, where a higher value means more variables will be eliminated.

One of the drawbacks of the sparse PLS is in selecting the right tuning parameter value for the given problem. Even with the aid of two dimensional cross-validation, several scenarios were found where a higher sparsity parameter value caused more variables to be selected due to interaction effects between a number of components and the sparsity parameter.

2.3.3.4 Method similarities

After reviewing the representative methods from each category as shown previously, several similarities can be identified. First, most of these methods focused on improving the training prediction performance through the metric of cross-validated Q^2 or other measurements of training performance. Second, the variable ranking criteria used to select variables are also similar among these methods. The most common ranking metrics used are VIP, loading weights (w) and correlation coefficients. Lastly, these methods generally assume that the training data originates from a single operating mode, which could be true in applications such as QSAR analysis, spectroscopic calibration or biomarker selection. In the cases of wrapper and embedded methods, the iterative search algorithms will seek to maximize the overall training data

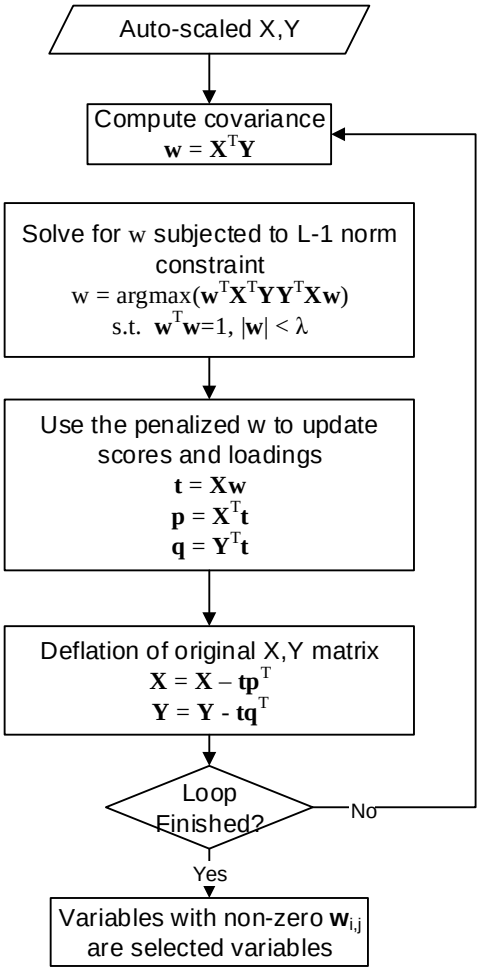


Figure 2.20: Sparse PLS variable selection algorithm overview

prediction performance. However, this assumption is not suited for industrial processes as these processes often experience changes in operation due to varying production demand or changing plant conditions. In our experience, all of the wrapper and embedded methods produced selected variables that are biased by a few globally relevant variables and neglected locally sensitive

variables. Comparisons of standard literature approaches and the prediction performance in testing data will be addressed in Section 2.3.6. Variable selection algorithms for industrial data should consider the changes in operating conditions when performing variable selection.

2.3.4 Moving Window Variable Importance in Projection (MW-VIP)

2.3.4.1 Clustering and variable selection

The two most common approaches to multiple operating mode modeling are local-models and mixed Gaussian or kernel density estimation [5]. In the first approach, each operating mode is modeled independently and then combined later. In the second approach, multiple operating modes are modeled explicitly through additional modal-specific parameters. However, these two approaches have yet to receive wide-spread industry adoption due to a number of limitations, such as difficulty in model maintenance and online model execution. A simpler alternative is to perform periodic model parameter updates based on an original model to correct for drifting biases and correlation changes. In all of the above scenarios, the optimal variable selection becomes more challenging due to changing variable importance associated with different operating modes. A possible variable selection method would be to perform variable selection in each operating mode independently and then combine the selected variables in one model. However, performing variable selection in each operating mode is also challenging. The variable selection process would depend on the prior clustering of data, which is often times unavailable or inaccurate. Second, future data or testing data might experience shifts in clusters which are not captured in individual cluster variable selections. As a result, variable selection for multiple operating mode data should identify all

sensitive variables without relying on clustering labels.

Figure 2.21 visualizes a scenario where multiple operating modes exist in the dataset. In the training data, there are two dominant operating modes (modes 1 and 2). The testing data still contains the same configuration, but the length of each mode has changed (mode 2). Variables that show strong correlation in mode 1 and 2 are favored in conventional variable selection methods, while mode 3 variables are not selected because they are less significant in primary operating modes. In this case, the testing performances of models based on the training data with the overall variable selection scheme would be poor due to the exclusion of important variables in mode 3.

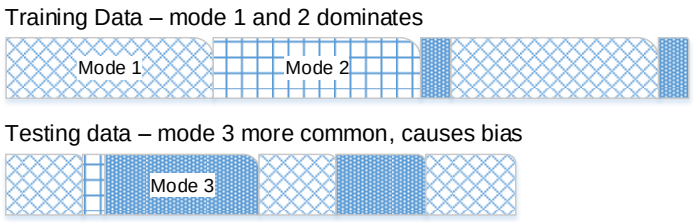


Figure 2.21: Illustration of the different multiple operating mode dynamics in training data and testing data

Figure 2.22 shows an example of the moving window VIP plot in three dimensions, where the VIP values are plotted on the Z-axis. Variables 38 to 51 are the most relevant variables according to their VIP values. However, during samples 40-60, the VIP values of variables 38 to 51 are much lower. Meanwhile, another set of variables demonstrated a range of values higher in VIP during this period. As a result, each operating mode has a set of variables that is responsible for explaining the correlation between $\mathbf{X}$ and $\mathbf{Y}$.

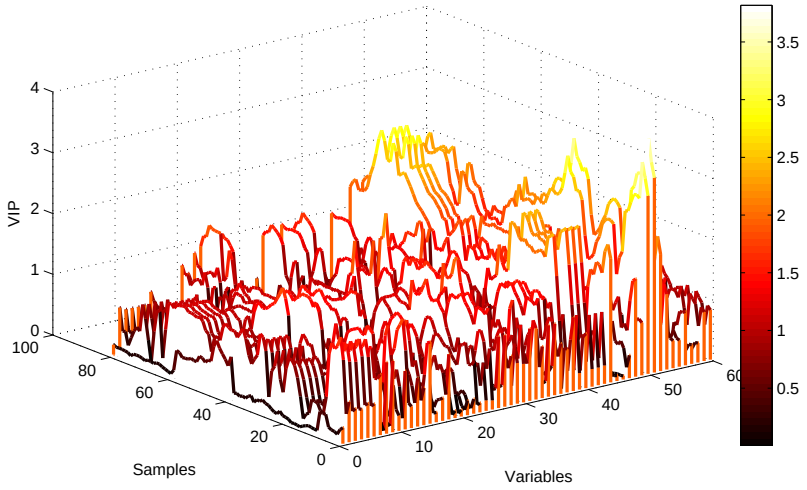


Figure 2.22: MW-VIP score of a 60 variable dataset as a function of variables and moving window sample

The MW-VIP variable selection method aims to identify both globally and locally important correlations that exist in various operating modes in the training dataset. This method belongs to the wrapper method category because of the iterations performed around the PLS regression step. The MW-VIP method differs from previous wrapper methods mainly in its treatment of the training data. This algorithm takes a set of overall training data and produces a set of variables based not only on the global importance ranking of variables, but also on locally sensitive variables. The preference for globally relevant variables in conventional wrapper methods is compensated in the MW-VIP method through a moving-window approach.

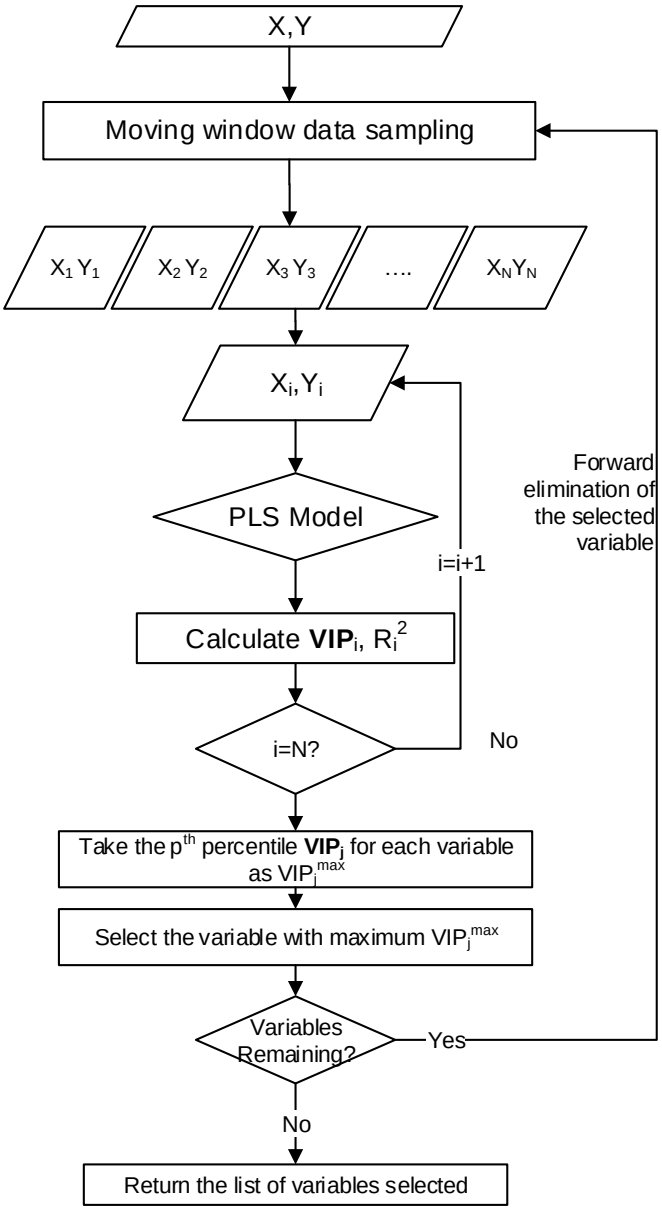


Figure 2.23: Moving window VIP variable selection algorithm overview

2.3.4.2 Moving window VIP variable selection algorithm

Figure 2.23 shows the overview flow chart of the MW-VIP variable selection method. Table 2.4 provides a detailed explanation of each step in the MW-VIP algorithm. The moving window VIP algorithm contains two iteration loops, the outer iteration loop is similar to the forward stepwise elimination loop, where the most important variable calculated from the inner loop is removed from the list of candidate variables in subsequent iterations. The outer iteration loop ensures that the VIP calculation of weaker variables is not biased by the presence of existing strongly correlated variables. In addition, the Q^2 of the selected variables will be calculated for each outer iteration. At the end of the selection process, the Q^2 values can be plotted against model size. The optimal number of variables in the final model can then be determined graphically by selecting the Q^2 . In the inner loop, moving window sampling is performed to divide the original training data into N partially overlapping sub data sets as indicated in Figure 3.3. After the PLS regression estimates are performed on each moving window dataset, the overall dimension of the VIP matrix will be $\mathbb{R}^{(N\times P)}$. The adjusted VIP* for each variable is calculated by taking the p^{th} percentile for all the moving window samples $(VIP(i = 1, 2, ..N, j))$. The percentile p acts as a tuning parameter for prioritizing variables with high correlation to the output during specified time windows or locally. As p approaches 100, variables with locally high correlations will be favored. Conversely, when the p approaches 50, the percentile calculation returns the median VIP value, which makes the selection results approximately equivalent to the result from the regular VIP filtering method. The 85^{th} percentile was used for the industrial case study, providing good results in the selection of variables. After the VIP* is calculated, a simple

ranking and forward selection scheme is then employed to select the appropriate variables.

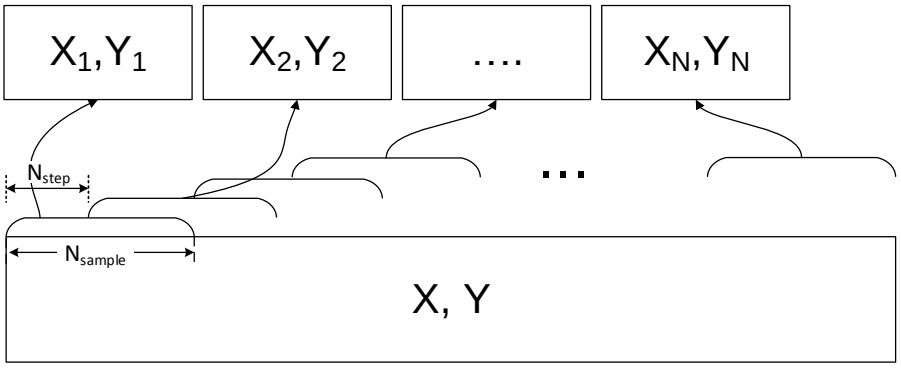


Figure 2.24: The moving window data partition scheme where smaller subsets of size N_{sample} offsets neighbouring subset by N_{step} samples

The MW-VIP outer iteration determines the optimal PLS model size based on the cross-validation performance of the models being tested. However, it is also possible to explicitly specify the size of the final PLS model.

2.3.4.3 MW-VIP tuning parameters

Three tuning parameters can be used to adjust the behavior of the MW-VIP variable selection. They are the moving window size (N_{sample}), the moving window step size (N_{step}), and the percentile value in calculating the adjusted VIP score (p).

Moving window sample size N_{sample}

The moving window sample size (N_{sample}) specifies the number of training data samples to be used in constructing each moving window of the PLS model. Having a smaller N_{sample} implies that the PLS model will increase in sensi-

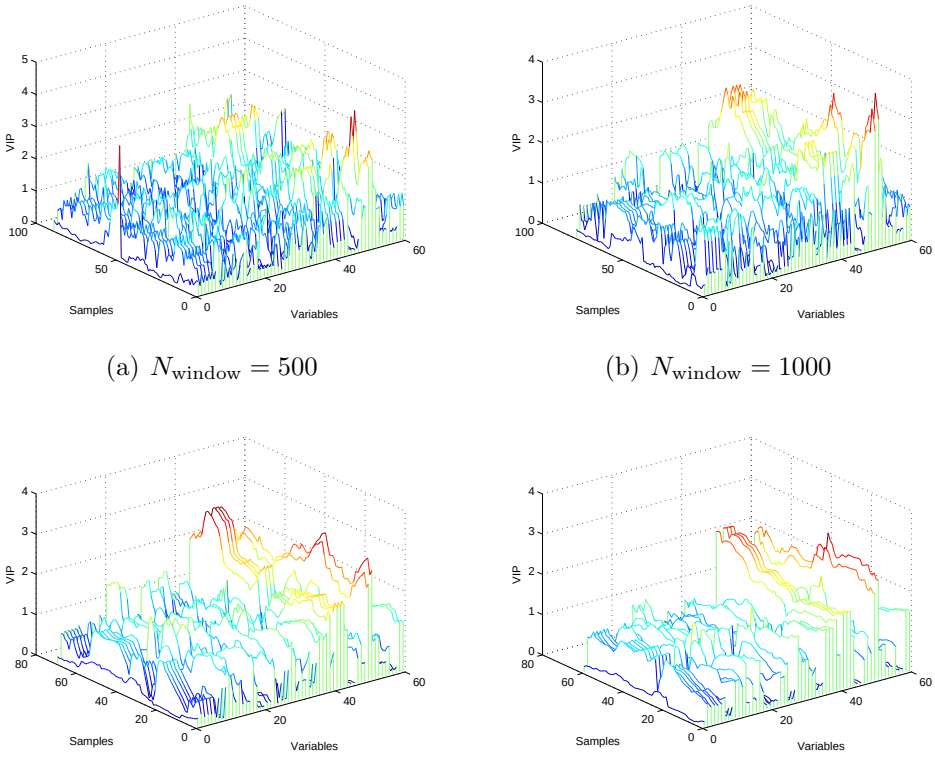
Table 2.4: The moving window VIP variable selection algorithm

Step	
1	Given training data $X \in \mathbb{R}^{M\times P}, Y \in \mathbb{R}^{M\times 1}$, mean center and scale X, Y to unit variance. Define moving window VIP parameters: N_{sample}, N_{step}. Let A denote the initial set of variables, $A = 1, 2, ..P$. Let B denote the selected set of variables, $B = \emptyset$.
2	Divide original data into N smaller sub-dataset pairs (X_i, Y_i) using the moving window partition scheme shown in Figure 3.3.
3	let $k = 1$, start the outer iteration loop.
4	Derive PLS model using standard PLS algorithm for (X_i, Y_i) using variables in set A.
5	Calculate $VIP_{i,j}$ for $j = 1, 2, ..P$ in moving window sample i according to Equation 2.19.
6	Repeat steps 4-5 for $i = 1, 2, ..N$ (inner iteration loop), until $VIP_{i,j}$ is fully populated.
7	Calculate the adjusted VIP_j^* for each variable j by calculating the **p**th percentile value of the vector $VIP(i = 1, 2, ..N, j)$. $$VIP_j^* = \text{percentile}(VIP(i = 1..N, j), p) \quad (2.22)$$
8	Rank variable importance using VIP_j^*, let $B_k = \underset{j}{\text{argmax}}(VIP_j^*)$ denote the most important variable at iteration k. Update current search space A to $A = A \setminus B_k$ (take the set difference).
9	Construct PLS model using B (previously determined peak VIP_j^* variables), calculate cross-validated Q_k^2 for each iteration k (outer iteration loop)
10	$k = k + 1$, Repeat steps 4-9 up to $k = P$.
11	Return B as the variable importance ranking based on the order of addition, the suggested model is the model with minimum Q^2.

tivity for local correlations within the moving window data range. Analogous to a moving-average filter, a larger moving window sample size reduces the sensitivity of the variable selection for local correlations and dampens the impact of noise in creating artificial correlations. The limiting case for increasing N_{sample} occurs when it approaches the total number of training samples. At this point, the MW-VIP algorithm will produce identical results compared to regular VIP filtering. In contrast, having a smaller moving window size will allow the MW-VIP algorithm to pick up more local correlations. However, the undesirable side-effect of higher sensitivity is that many intermittent noisy variables are also selected. Figure 2.25 shows the effect of different moving window sizes on the VIP value of a process variable under the same training dataset. As the number of moving window training samples increases for the MW-VIP method, the VIP trend of the variable being plotted becomes smoother. This smoothing of the trend shows the filtering effect of utilizing a larger moving window sample. The optimal choice of N_{sample} lies between the two limiting cases. The recommended rule of thumb is that the moving window sampling size should not be too large to filter out the quality between operating mode variable dynamics, nor should it be too small to introduce excessive noise. The ideal N_{sample} should allow MW-VIP trends to capture the dynamic in the VIP with less noise. We recommend using a "time-constant" calculated from the operating mode changes as an initial estimate for N_{sample}. This definition of time-constant is based on first order dynamic systems, where the time constant of a system is the time it takes to reach approximately 60% of a new steady-state condition.

Moving window step size N_{step}

The moving window step size is the primary tuning parameter for controlling



(a) $N_{\text{window}} = 500$ (b) $N_{\text{window}} = 1000$

(c) $N_{\text{window}} = 1500$ (d) $N_{\text{window}} = 2000$

Figure 2.25: Moving window VIP contour showing different levels of smoothing under different moving window size ($N_{\text{window}} = 500, 1000, 1500, 2000$)

the number of iterations in the moving window regression process. A larger N_{step} reduces the overlapping data between subsequent moving window samples and results in smaller number of iterations. A smaller N_{step} will result in more overlap and consequently more iterations to loop through the same dataset. Similar to the N_{sample} parameter, N_{step} has a smoothing effect on the calculated VIP trends. N_{step} is bounded between 1 and N_{sample}. Our recommended value for N_{step} is 10% of the N_{sample}. At this value, we found that the MW-PLS algorithm was able to achieve a good balance between computation speed and variable selection sensitivity.

The p value in percentile calculation

As shown in Table 2.4, the p value is the percentile used in deriving the adjusted VIP score (VIP_j^*). The valid range of p falls between 50 and 100, which corresponds to taking the median or the maximum of the VIP values as VIP_j^* respectively. Using a higher p value results in higher sensitivity of MW-VIP to local correlation changes. In practice, our recommended values for p ranges from 85 to 95. Since adjusting the p value and utilizing a moving window sample size achieve the same effect, we recommend fixing the percentile value p and then adjusting the MW-VIP sample size to allow for optimal tuning of moving window filtering effects.

2.3.4.4 Rank deficiency in high dimensional data

For data sets with multiple variables and limited samples (large P small N problem), the applicability of the MW-VIP method could be limited. Although PLS regression is known to be statistically robust against rank-deficient $\mathbf{X}$ matrices [36], the calculated VIPs are less stable and contain excessive noise, which are undesirable in ranking of variables. In this case, the two possible

remedies are as follows:

Apply preprocessing to penalize collinear variables Process expertise, correlation analysis, and signal to noise ratio filtering are useful tools to eliminate variables prior to applying the MW-VIP variable selection. In correlation analysis, clusters of collinear variables can be identified, in subsequent PLS iterations, collinear variables in each cluster can be penalized to prevent them from biasing the VIP calculation towards groups of collinear variables. In our study, this method was found to be very effective to reduce the impact of collinear variables on selection results; over half of the variables in the initial process scope were eliminated by applying these filtering techniques. This leaves a much smaller set of candidate variables for the MW-VIP algorithm.

Apply sparse PLS instead of conventional PLS as the inner algorithm In the current MW-VIP selection method, the inner loop utilizes the conventional PLS and VIP definitions to calculate the local VIP of variables for each moving window sample. This is not suitable for large P small N problems (for example, unfolded batch data with limited samples). To accommodate the rank deficient moving window samples, sparse PLS can be used as the inner iteration loop instead [65]. Sparse PLS is suited for high-dimensional problems and produces a PLS model with a reduced set of variables. Therefore, the VIP score or the selection frequency from sPLS can also be used for variable ranking purposes, since they take into account of local operating modes.

2.3.5 Model evaluation criteria

The current literature in variable selection focuses on evaluating the prediction performance of models created from the selected variables. While

prediction is an important criterion for model performance, it could be biased by the quality of the testing dataset. Four classes of evaluation criteria are proposed here, in order to assess four aspects of PLS regression models: prediction performance, model parsimony, parameter robustness and process monitoring sensitivity.

2.3.5.1 Prediction performance

Model prediction performance is a common metric used to evaluate the effectiveness of a prediction model. The most common prediction criteria are the coefficient of determination (R^2) for testing dataset, testing data set bias, and the root mean squared error of prediction (RMSEP). They are defined as follows:

$$\begin{aligned} R^2 &= 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \\ \%\ \text{Bias} &= \frac{\sum (y_i - \hat{y}_i) \times 100}{\sum y_i} \\ \text{RMSEP} &= \sqrt{\frac{\sum (\hat{y}_i - y_i)^2}{N}} \end{aligned} \tag{2.23}$$

where N is the total number of samples, and $\hat{y}_i$ is the predicted quality variable, y_i is the measured quality variable, and $\bar{y}$ is the mean.

R^2 and RMSEP are regression fit indicators; while the prediction bias indicates if the model consistently under-predicts (positive-sign) or over-predicts (negative sign) the measured variable.

2.3.5.2 Model parsimony

The cut-off for the number of variables selected in the final model varies among the different selection methods. As a result, the trade-off between model complexity and parsimony needs to be taken into consideration in performance evaluation. The Akaike Information Criterion (AIC) is a commonly used model parsimony indicator. The general case of AIC is defined in [72] as follows:

$$AIC = 2k - 2\ln(L) \tag{2.24}$$

where k is the number of parameters in the model and L is the maximized likelihood function value. For least squares estimation with normally distributed errors, the AIC can be expressed as [73]:

$$AIC = n\log\left(\hat{\sigma}^2\right) + 2K \tag{2.25}$$

where n is the number of samples, K is the number of parameters in the model, and $\hat{\sigma}^2 = \frac{\Sigma(\hat{\varepsilon}_i)^2}{n}$ is the estimated variance of the residuals. The most appropriate model would have the smallest AIC value. An interpretation of the AIC score as outlined in Akaike [72] can be expressed as a relative likelihood of model i being a better model (in term of minimizing information loss) than the model with the minimum AIC:

$$p(AIC_i) = \exp\left(AIC_{\min} - AIC_i\right)/2 \tag{2.26}$$

where $p(AIC_i)$ is the probability of model i being better than the model with the minimum AIC.

2.3.5.3 Process monitoring sensitivity (T^2 ratio)

The process monitoring sensitivity is defined as the percentage of incoming data that fall within the T^2 alarm limit (99% confidence). The Hotelling's T^2 statistic monitors the deviation of incoming data from the training data and is calculated using:

$$T^2 = \mathbf{x}^T \mathbf{P} \Lambda^{-1} \mathbf{P}^T \mathbf{x} \tag{2.27}$$

where the T^2 statistic follows a F-distribution with parameters: $\frac{N-l}{l(N-1)} T^2 \sim F_{l,N-l}$ [74], $\mathbf{x}$ is the vector of incoming data, $\mathbf{P}$ is the loading matrix from PCA or PLS, and the Λ is the covariance matrix of $\mathbf{X}_{\text{training}}$ and acts as a normalizing factor.

The process monitoring sensitivity (percentage of T^2 violations) is useful in examining the correlation between model prediction performance and process monitoring sensitivity. Having a model with a strong correlation between prediction performances and monitoring sensitivity is desirable because the process monitoring indices (T^2) can then be used to gauge prediction confidence.

2.3.5.4 Parameter robustness through residual distribution test

Another desirable trait of good variable selection is that the selected variables are robust, meaning that their parameter values do not have to be updated frequently. Inspired by Li's work with population analysis techniques in evaluating variable selection [75], the parameter robustness metric compares how robust the selected variables are when they are updated with incoming data. The proposed metric functions by examining how much a model update

will improve the performance of future predictions.

Figure 2.26 shows the steps to calculate model parameter robustness score. In our proposed method, Monte-Carlo block sampling randomly selects a segment of testing data to be used in updating the model. Twenty percent of the sampled data will be excluded from the model update and to be used as testing data. The block size is also randomized for each sample to mimic different durations of operating mode changes. After the model has been updated, the residual of both the updated model and the original model will be calculated using the testing data. This process is repeated for N times (where N is the number of Monte-Carlo sampling loops performed). The iterative Monte-Carlo procedure generates a population of updated PLS models, the sum of residuals on the testing data are collected from both the updated model and the original model. To compare the population of updated models against the original overall training model, we apply the nonparametric two-sample Kolmogorov-Smirnov test for distribution equality. The null hypothesis of this test is that both residual distributions are from the same distribution. The test statistic p-score can be then used as a measure of the difference in residual distribution between the updated model and the original model. The two-sampled Kolmogorov-Smirnov test statistic is defined as follows [76]:

$$D_{n,n'} = \sup \left| F_{1,n}(x) - F_{2,n'}(x) \right| \tag{2.28}$$

where $F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{X_i<x}$. $I_{X_i<x}$ is an indicator function whose value is 1 if the condition $X_i \leqslant x$ is satisfied and 0 otherwise. The null hypothesis is then rejected at significance level p if $D_{n,n'} > c(p) \sqrt{\frac{n+n'}{nn'}}$, where $c(p)$ is the critical value for two-sampled test and can be looked up from standard statistics reference handbooks. Thus, the residual distribution shift score (RDSS) is defined

as:

$$RDSS = -\log(p) \tag{2.29}$$

A smaller RDSS score reflects a lesser probability of rejecting the null hypothesis, which means that the particular PLS model is more robust than others.

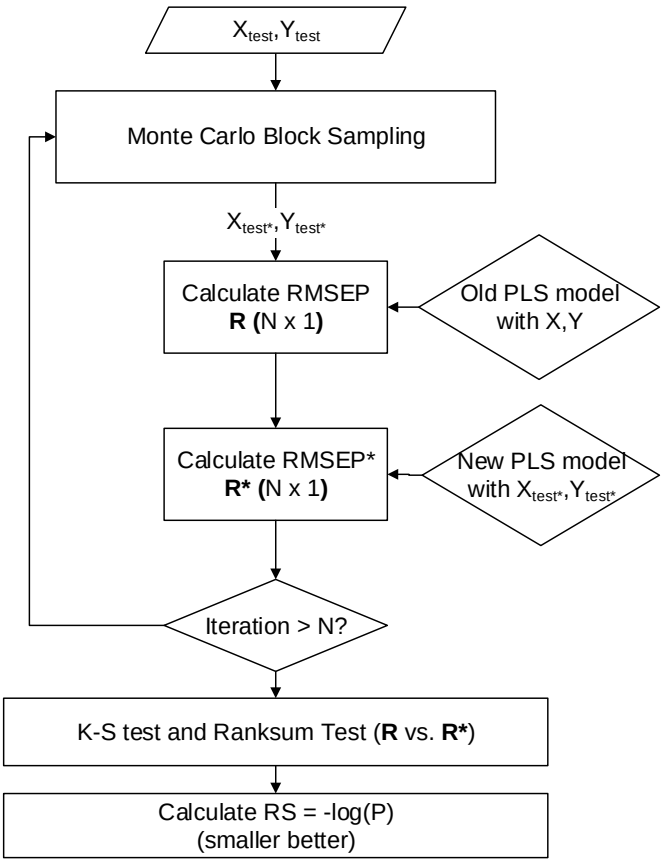
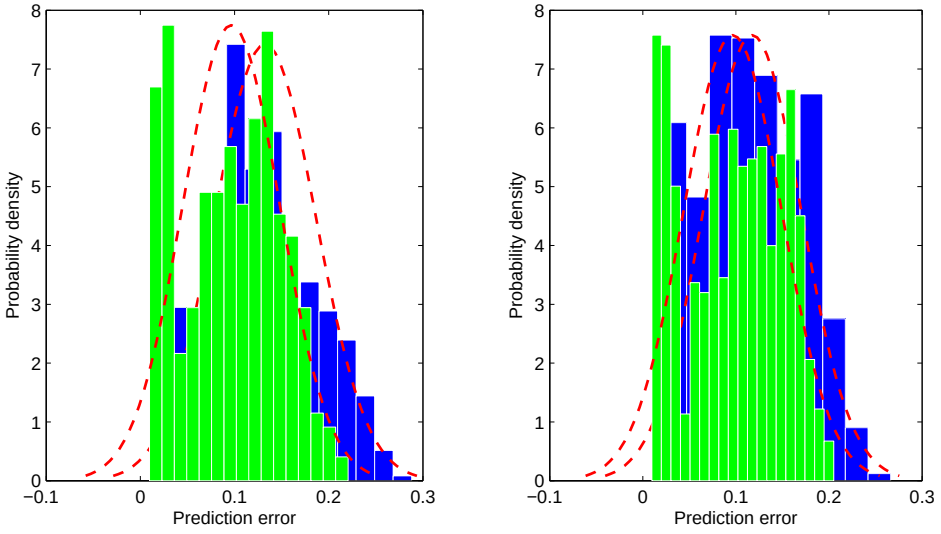


Figure 2.26: Flowchart showing the steps to calculate residual distribution shift score

Figure 2.27 shows an example of the residual distribution test applied

to two PLS regression models. Figure 2.27(b) shows a more robust model with negligible shift in its residual distribution when the model is updated. The RDSS scores calculated using the K-S test for the left and right figures are respectively 185.3 and 121.3. It is important to note that, the RDSS score is a relative measure of model robustness given the same set of testing data (analogous to AIC). The RDSS score cannot be compared for models across different data sets.



(a) RDSS = 185.3, larger residual shift

(b) RDSS = 121.3, smaller residual shift, more robust

Figure 2.27: Example of residual distribution shifts calculated using testing dataset II (a) Residual moved toward the left after parameter updates, (b) Relatively stationary residuals before and after model parameter updates

2.3.6 Results and discussion

2.3.6.1 Simulated case study

A simulated case study was performed to illustrate how MW-VIP variable selection differs from conventional VIP variable selection.

The variable trajectories are generated in MATLAB. Each x variable composes of a random-walk noise series, randomly injected step changes and white noises to mimic real data collected from plant instrumentation. After the initial set of $\mathbf{X}$ variables (that are independent) are generated, collinear variables are created by linearly combining the existing trajectories. The coefficients used in this linear combination are sampled from a uniform distribution with bounds between -2 to 2 ensuring that the redundant variables will be within the same order of magnitude with the existing variable trajectories. In this simulation, six redundant variables are added to the $\mathbf{X}$ matrix as collinear variables. Figure 2.28 shows the correlation map of the $\mathbf{X}$ matrix. The number in each square denotes the correlation value between the two variables. The variables are arranged in descending order of correlation with adjacent variables, so the two clusters of variables in the top left corner represent two groups of redundant variables. Using information from the correlation matrix, the pre-processing step outlined in Section 2.3.4.4 is applied to eliminate the redundant variables prior to applying both the MW-VIP and the VIP filtering methods.

To generate the response variable, two linear models are explicitly defined as follows:

$$\begin{aligned} y_1 &= 10x_5 + 5x_9 + 3x_{10} + x_{13} + \epsilon \\ y_2 &= 9x_5 - 5x_{17} + 4x_{10} + x_7 + \epsilon \end{aligned} \tag{2.30}$$

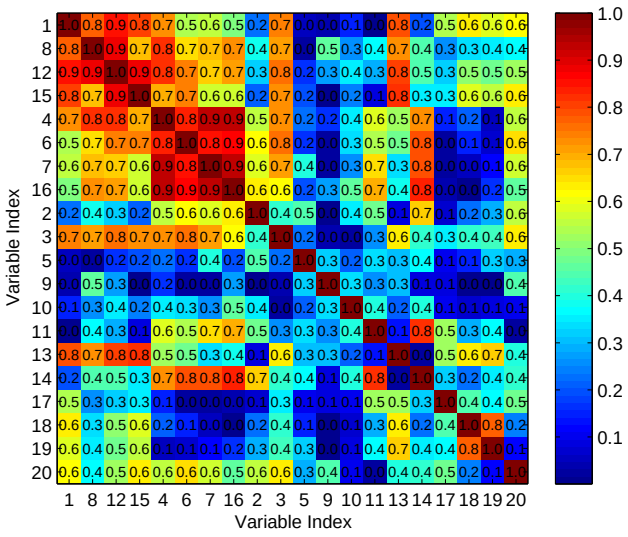


Figure 2.28: Variable correlation map of the training data $\mathbf{X}$ showing two clusters of collinear variables

where y_1 is the process model for operating mode 1, y_2 is the process model for operating mode 2, and ϵ is a white noise sequence. Note that the magnitude of the coefficients in these two operating modes are chosen to contain: two shared variables (x_5, x_{10}) and two sets of variables unique to each operating mode (x_9, x_{12} for mode 1 and x_{17}, x_7 for mode 2). Additional simulation parameters are listed in Table 2.5.

The generated trajectories for the training dataset are shown in Figure 2.29. The sharp drop at sample 2000 marks the boundary that separates operating mode 1 from operating mode 2.

The MW-VIP variable selection was then applied to this dataset using the tuning parameters listed in Table 2.5. Since the MW-VIP method applies a forward elimination scheme to remove the most significant variable from participating in the next iteration, the variables selected and the cross-validated

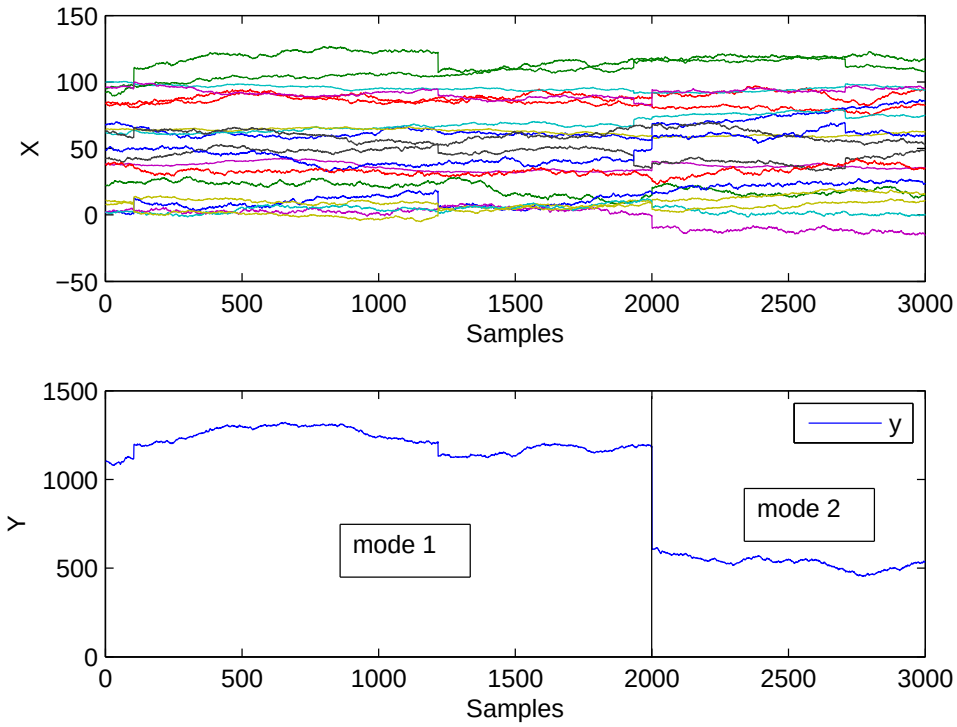


Figure 2.29: Simulated training data trajectories of $\mathbf{X}$ and the corresponding quality variable $\mathbf{Y}$ containing two operating modes

Table 2.5: Simulation conditions and variable selection algorithm settings used in the simulated case study

Simulation setting / tuning parameter	**Value**
Sample size	5000
Number of X Variables	20
Number of operating modes	2 modes
Percentage of redundant variables	30%
Training data mode distribution	1-2000 mode 1 2001-3000 mode 2
Testing data mode distribution	3001-3500 mode 1 3501-5000 mode 2
MW-VIP moving window size	500
MW-VIP step size	50
MW-VIP percentile p value	95
Number of components for the final PLS model	3

Q^2 value at each selection iteration are tabulated in Table 2.6. The rows in Table 2.6 are arranged in descending order of importance such that the more significant variables are shown first. Note that variable x_5, the most significant variable according to Equation 2.30 was first selected. Cross-referencing the order of variable selection with Equation 2.30 shows that the MW-VIP algorithm successfully captures the important variables in both operating modes. The order of variable ranking also corresponds to the magnitudes of the variable coefficients in Equation 2.30. Table 2.6 shows the variable selection results for the VIP filtering method. The same forward elimination scheme was applied to the VIP filtering to determine the model size cut-off. The VIP filtering algorithm was unable to correctly identify the key variables in the original equation. In fact, none of the relevant variables were identified in the top six variables ranked by the VIP filtering. Upon further troubleshooting, we discovered that the variables ranked highly in the VIP filtering selection

are variables from collinear clusters as shown in Figure 2.28. Furthermore, Figure 2.31 plots the trajectories of x_{12}, x_4, x_1 and y to better illustrate the correlation between these variables and the output. Since the VIP filtering method calculates the VIP across the operating mode transition, variables that are correlated with this sharp change during the transition will be favored in the VIP ranking. A closer inspection of the variable x_{12} shows that this trajectory is not highly correlated to the output with the exception of the sharp drop at around sample 2000. In contrast, the MW-VIP selection results were unaffected by collinear variables, even when the same preprocessing steps are applied to both selection methods.

Finally, to determine the cut-off point for the optimal number of variables to be selected, Figure 2.30 shows the cross-validated Q^2 as a function of the number of parameters in the PLS model. The MW-VIP series shows a maximum Q^2 occurring at model 4, which corresponds to six variables selected (x_5,x_9,x_{10},x_{17},x_{13},x_7). This result agrees with the original model in Equation 2.30. On the other hand, the regular VIP method produced similar Q^2 values for the six models, making it difficult to decide which model should be selected.

This simulated case study shows that the MW-VIP algorithm was able to select relevant variables in multiple operating mode scenarios. The cut-off point for the number of variables in the optimal model can also be determined graphically. Finally, the MW-VIP algorithm was also shown to be effective in presence of collinear variables, which was not the case for the conventional VIP filtering technique.

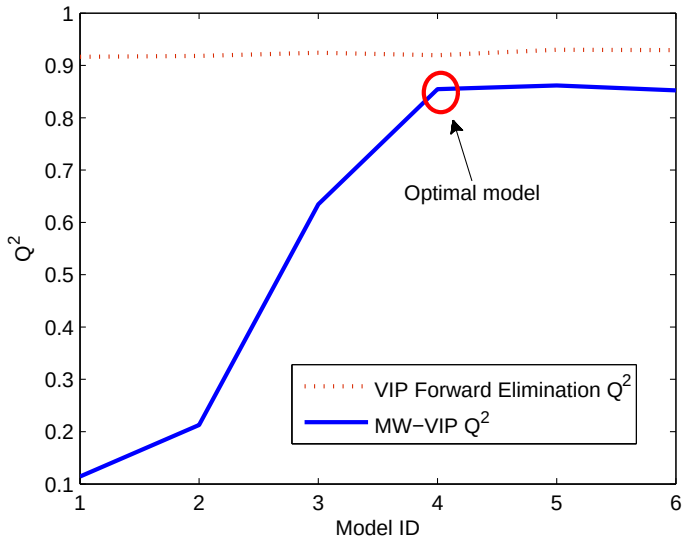


Figure 2.30: Cross-validated Q^2 as a function of variables selected by MW-VIP and VIP filtering methods

Table 2.6: MW-VIP and VIP filtering variable selection results

		Variables in order of importance								Q^2
MW-VIP	Model 1	5	9	10						0.11
	Model 2	5	9	10	17					0.21
	Model 3	5	9	10	17	13				0.63
	Model 4	5	9	10	17	13	7			0.85
	Model 5	5	9	10	17	13	7	14		0.86
	Model 6	5	9	10	17	13	7	14	16	0.85
regular VIP	Model 1	12	4	1						0.92
	Model 2	12	4	1	15					0.92
	Model 3	12	4	1	15	8				0.92
	Model 4	12	4	1	15	8	3			0.92
	Model 5	12	4	1	15	8	3	6		0.93
	Model 6	12	4	1	15	8	3	6	7	0.93

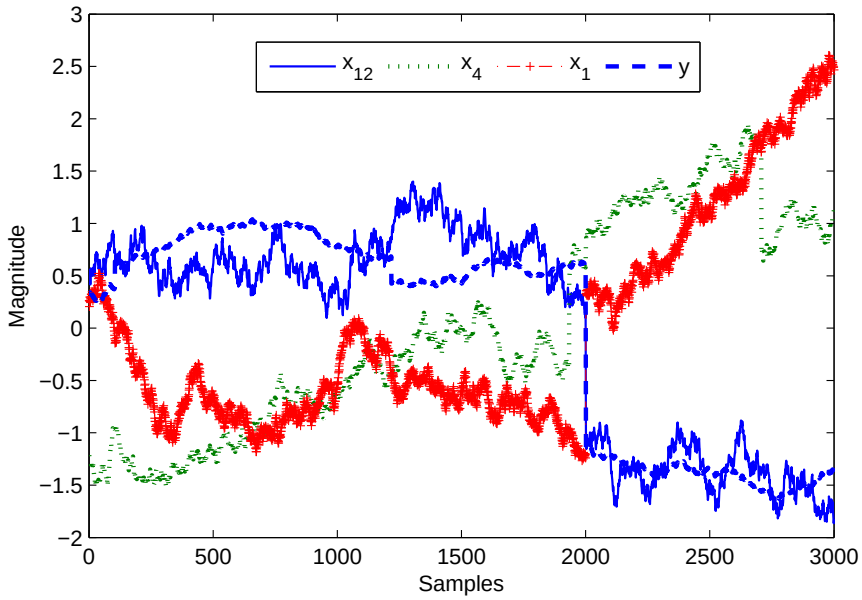


Figure 2.31: Trajectories of the three highest ranked variables from the VIP variable selection and the output variable

2.4 Summary

In this chapter, two topics related to preliminary data processing steps are discussed in detail. These preprocessing steps are critical pre-requisites for a successful data-driven model.

For batch processes, trajectory alignment is one of the first issues that need to be addressed. In this discussion, we showed that the existing methods for batch trajectory alignment are sometimes ineffective or inefficient at aligning trajectories from batch processes. Consequently, we exploited that fact that only selective segments of the batch processes contain useful information. As a result, the constrained selective dynamic time warping (CsDTW) method first performs a fuzzy search of key alignment markers on the trajectories to be aligned. Once identified, these alignment markers will be incorporated as global constraints in formulating a constrained dynamic time warping problem. The resulting problem can be solved with high efficiency. Results from the case studies show that CsDTW exceeded truncation, interpolation, linear time scaling, and traditional dynamic time warping in alignment performance. In addition, CsDTW exceeded covariance optimized warping in computational speed and also ease-of-use.

For variable selection, the common techniques assume only one operating mode exists in the data. This is not a valid assumption for applications where multiple operating points are present. The MW-VIP method exploits the time dependency of process data to maximize selection of relevant variables in each operating mode. Tuning parameters of moving window size and percentile value can be used to tune the sensitivity of the MW-VIP method to local correlations. To evaluate the selection candidate models, four types of performance criteria were proposed. In addition to model performance,

model parsimony can be assessed by calculating the AIC of each model. The residual distribution shift score was developed as an indicator for testing the robustness of candidate models. Lastly, the process monitoring T^2 ratio yields information on the sensitivity of model prediction quality to excursions and abnormalities in process data. These four types of evaluation criteria combined can be used to improve assessment of model performances and variable selection results. Modeling results from the simulated case study and the industrial dataset show that the moving window PLS produced a better model than SwPA, VIP filtering and sparse PLS selection methods. To better deal with collinearity in large data sets, future work on the MW-VIP method will investigate substituting the standard PLS algorithm in MW-VIP with sparse PLS to achieve simultaneous variable selection and sparse VIP calculation.

Chapter 3

Maintaining Quality and Performance of Data-driven Models

3.1 Introduction

With exponential improvement in computer processing power and data storage capacity, more and more industries are starting to look for new ways to utilize the data collected. In the new data-centric era, understanding and utilizing data better than the competition is seen as a critical factor to success. In the context of chemical engineering industries, having large sets of data and better processing power offer us an opportunity to further enhance process understanding, improve process reliability, optimize operation and improve safety.

In the last two decades, data-driven techniques have been applied in developing soft sensors models and multivariate statistical monitoring tools to monitor and predict difficult to measure processes. These approaches have been demonstrated in many real world industrial applications [23, 24, 25, 26, 27, 8, 28]. A comprehensive review of soft sensor and their development approaches is available by Kadlec et al. [11]. However, despite astounding success in both academia and industry, there are still some issues that are unresolved. One of the key challenge in soft sensor is the maintenance of their performance over longer periods of operation.

A intuitive explanation of why performance degradation happens in

data-driven models is that change happens. Processes are never truly operating at a steady-state when examined from a longer time-scale. Equipment maintenance, sensor degradation, and personnel change all could have an impact on how the system is operated. In addition, most systems are also subjected to outside disturbances, such as feedstock changes, weather. As a result, over periods of time, the process condition usually deviates from the original process condition when the data-driven models are developed, and the quality of the data-driven models degrade thereafter.

There has been a lot of work on addressing this issue. The main approach is to provide an adaptation mechanism to allow automatic re-training of the online models. In these cases, data collection and sometimes lab sampling of important quality variables cannot be displaced entirely and still need to be performed (maybe at reduced intervals). Selecting what data to use to update the model becomes a difficult question to answer. In addition, depending on the type of soft sensor in place, the difficulty of retraining model could range from trivial (simple least squares) to impossible (support vector machines). Kadlec provided a comprehensive review of the soft sensor model update mechanisms in [77]. Some of the notable methods and work associated with their update mechanisms are listed in Table 3.1.

In this chapter, we focus on addressing the quality degradation and performance issues of PLS models using two approaches. In the first section, a more robust version of moving-window PLS update technique is proposed. Using Total-PLS quality relevant decomposition, the incoming data are screened for quality issues and outliers that would make the updated PLS more noisy. The end result is that the PLS model update mechanism is more robust and resilient to outliers while also being able to keep track with process condition

Table 3.1: Popular data-driven techniques and their adaptation mechanisms

Method	Description	Advantages	Disadvantages	Speed
Recursive weighing [27, 78, 79, 80]	Augment existing loading matrices with new data to recursively update PLS beta coefficients	Simple, efficient	Usually only updates beta coefficients, does not include diagnostics	Fast
Moving Window [81, 9]	Slide a data window to include new data and discard older data and perform PLS on new window of data	Straightforward, easiest to implement	Prone to noise and data quality issues	Fast
Just-In-Time Adaptive Soft Sensor [18]	Locate the most correlated segment of data with the incoming sample and train a model to make a prediction	allowed for better adaptation against abrupt process changes	Large memory requirement	Slower
Adaptive kernel learning	based on least squares SVM with both forward and backward learning modes	regression and classification	Difficult to implement, too many degrees of freedom	Slow
Incremental local learning [17]	Use fuzzy combination with local experts, partitions based on relative residual change	Nonlinear, disjointed data friendly	Difficult for input with higher dimensions, adaptation limited	Slow
Growing Structure Multiple Linear Regression [82]	Use GSOM topology and multiple linear regression, new partitions created based on SOM growth mechanism	Ensemble-based, nonlinearity, flexible adaptation	Difficult for input with higher dimensions	Slow

changes.

In the second section, we address soft sensors challenges for systems with multiple operating conditions or nonlinear systems exhibiting multi-modal behaviors. In these systems, simple model update mechanisms such as recursive model update or moving window model update is not sufficient to deal with abrupt process changes or transitions. For these scenarios, a new method based on using a divide and conquer approach to train multiple local PLS models simultaneously is proposed. The novelty of our approach resides in using projection based local models (PLS or PCA) with growing self-organizing maps to allow for elastic model complexity tuning during training and online adaptation. This flexible framework can also be used to explore new datasets and rapidly develop model prototypes.

3.2 Integrated Moving Window Update Framework

In this section we present a integrated moving window update framework that aims to improve the robustness of recursive PLS models in the presence of outliers in both input data and output data.

3.2.1 Integrated MW-PLS Model Overview

Moving window PLS is a key part of the proposed integrated framework that aims to address the issue of process drifts in static PLS models. The integrated framework comprises of preprocessing, model development, and model maintenance to ensure that the models constructed are suitable for online adaption. Key problems that this setup aims to address are process drifts and outliers during model update. Figure 3.1 shows the key information flow in the integrated online framework and gives an overview of the proposed con-

figuration. The two major components of the framework are the prediction model (PLS) and the soft sensor maintenance mechanism. In the prediction model, an existing PLS model obtained either off-line or from previous online runs will be used in conjunction with T-PLS update outlier screening. The T-PLS multivariate statistics serves as a prediction quality alarm that will alert users of possible prediction quality issues. In the quality maintenance module, additional metrology data will be synchronized with the corresponding fault detection trace data to carry out the model maintenance and update tasks. The same T-PLS statistics will be used to screen out potential outlier batches that will negatively affect the model update process. Afterwards, the model simplification step removes redundant variables and uninformative variables to reduce memory foot-print. The T-PLS statistics control limits are updated for the new PLS model. The soft sensor maintenance module will run asynchronously from the prediction module, the speed of the model update will depend on the availability of metrology data. However, one can easily implement logic to force an model update in cases where excessive alarms are raised in the prediction module.

3.2.2 Moving Window PLS

Multiway Partial Least Squares (MPLS) Regression Latent variable methods such as partial least squares (PLS) and principal component analysis (PCA) project high-dimensional data onto low dimensional latent subspaces. These methods are widely used to analyze large data sets such as those encountered in plasma etching ([83, 13, 12]) A typical plasma etch dataset contains over 30 sensor measurements, which include gas flow, chamber conditions, and RF circuitry readings. When sensor signals are unfolded,

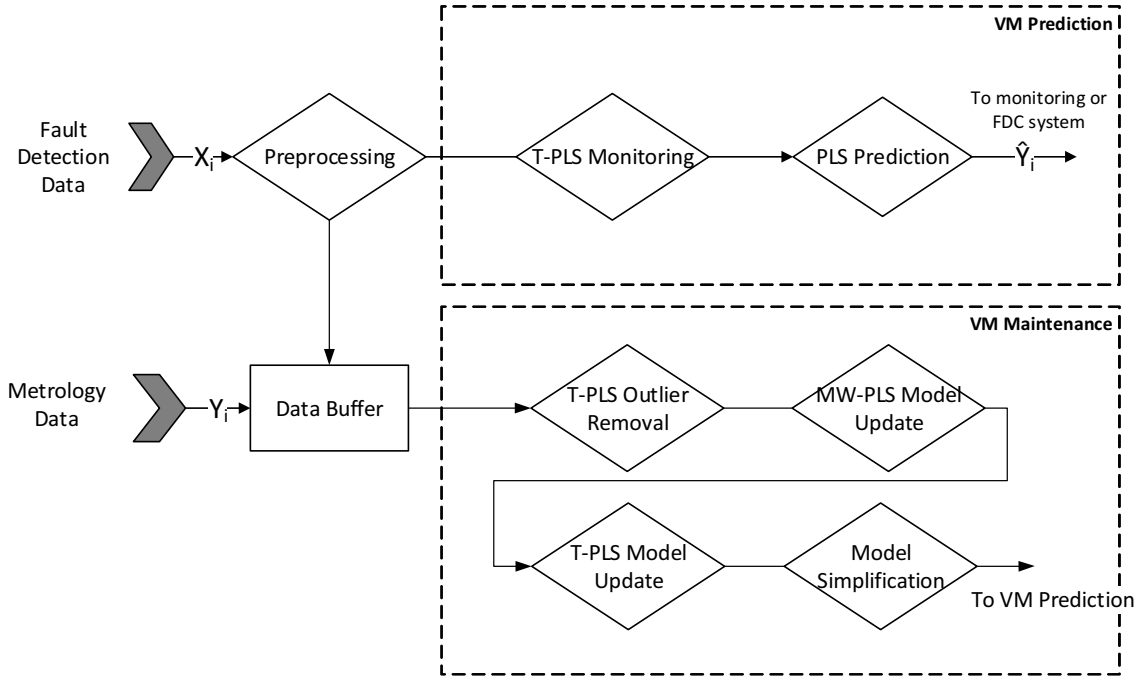


Figure 3.1: Proposed integrated VM framework using T-PLS multivariate screening and moving window model updates

these sensor readings result in over 3000 variables. Unfolding transforms a three-dimensional data array into a two-dimensional matrix for the purpose of performing PCA and PLS [5]. The feasibility of using a multi-way approach (unfolding) for PLS/PCA analysis was demonstrated by Wold et al. and Nomikos et al. in process industry monitoring [84, 85]. Figure B.1 visualizes the transformation of a three-dimensional data cube into a two-dimensional matrix. The unfolded data is then analogous to data from a continuous dataset and can be used directly in PLS modeling. The two mainstream PLS algorithm are the Nonlinear Iterative PLS (NIPALS) and the SIMPLS[53, 52]. Seven additional PLS algorithms are evaluated and reviewed by Andersson [86]. Both PLS algorithms decompose the mean-centered matrices into the

following form:

$$\begin{aligned} \mathbf{X} &= \mathbf{T}\mathbf{P}^T + \mathbf{E} \\ \mathbf{y} &= \mathbf{U}\mathbf{Q}^T + \mathbf{F} \end{aligned} \tag{3.1}$$

where $\mathbf{T} \in \mathbb{R}^{n \times A}$ and $\mathbf{U} \in \mathbb{R}^{n \times A}$are the $\mathbf{X}$ and $\mathbf{y}$ scores respectively, $\mathbf{P} \in \mathbb{R}^{m \times A}$ and $\mathbf{Q} \in \mathbb{R}^{p \times A}$ are the loadings for $\mathbf{X}$ and $\mathbf{y}$, respectively. The number of components in the PLS model is typically determined through cross-validation or through information criterion such as the Akaike Information Criterion (AIC) [87]. The PLS algorithm maximizes the covariance between the X-scores and y-scores; this property leads to PLS requiring fewer components when compared to principal component regression models [87].

To apply the PLS latent structures in regression, given unfolded input data matrix $\mathbf{x}_0$, the output predictions $\hat{\mathbf{y}}_0$ can be calculated linearly using $\hat{\mathbf{y}}_0 = \mathbf{x}_0\hat{\beta}_{pls}$. The $\hat{\beta}_{\text{pls}}$ can be expressed as a function of the latent variables as follows:

$$\hat{\beta}_{\text{pls}} = \mathbf{R}\left(\mathbf{T}^T\mathbf{y}\right) = \mathbf{R}\mathbf{R}^T\mathbf{X}\mathbf{y} \tag{3.2}$$

where $\mathbf{R}$ is the weight matrix in the SIMPLS algorithm. In the NIPALS method, the weight matrix $\mathbf{R}$ can also be calculated iteratively.

PLS faces challenges when the input and output relationship is non-linear. However, multiple options exist depending on the type of nonlinearity encountered. Qin and McAvoy developed neural network PLS (NNPLS) that approximates the latent relationship between the input and output scores using a feed forward neural network[88]. Using this approach, nonlinearity in the latent score space can be approximated to arbitrary precision; however, it is difficult to verify or prevent over-fitting for the latent scores. Kernel PLS maps

original $\mathcal{X}$ space data into a nonlinear feature space $\mathcal{F}$, where the relationship can be represented linearly [89]. However, selecting the right kernel function and requirement for large amount of training data ($n \gg m \times p$) reduces the practical effectiveness of this method [86]. Another method to approximate nonlinearity is by constructing multiple models for each operating regime. The appropriate operating regime can then be selected at run-time based on classification algorithms such as clustering or decision-trees. The detailed merit and challenges of this class of methods will be discussed in more detail in Section 3.3.

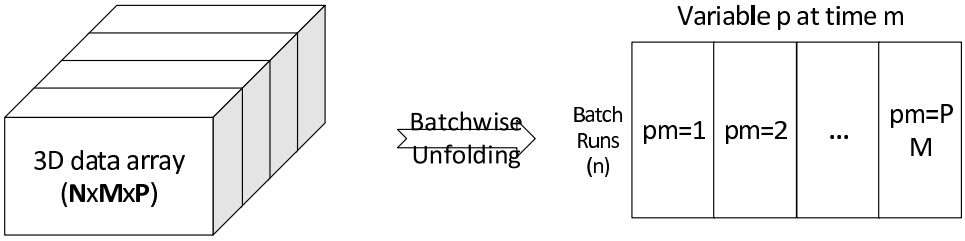


Figure 3.2: Unfolding of a three-dimension data array($\mathbb{R}^{n\times m\times p}$) into a two-dimensional matrix ($\mathbb{R}^{n\times mp}$)

Moving Window Update The moving window and recursive PLS are two adaptation mechanisms to compensate for process drifts in PLS models. Recursive PLS was first introduced by Helland[78] and then improved by Qin [27]. An application of recursive PLS in virtual metrology was demonstrated by Khan et al. [80] using simulated data. However, the drawback of recursive PLS is that the multivariate statistics (Hotelling's T^2 and Q statistics) are more difficult to adapt. Moving window PLS is an alternative formulation to recursive PLS that allows for easy update of the monitoring indices. In MW-PLS, the PLS algorithm itself remains unchanged. As new data are acquired, the window of training data slides along the data-set. The two tuning

parameters in moving window PLS are the window size and the window shift size; these two tuning parameters are graphically illustrated in Figure 3.3.

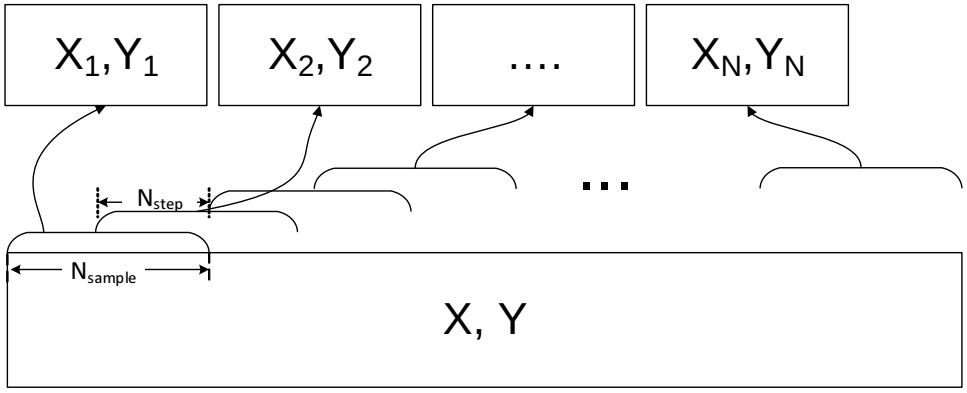


Figure 3.3: Moving window data partition of a dataset using two parameters: sample size(N_{sample}) and step size (N_{step})

In addition, to ensure consistency and resiliency against process noise and disturbance, it might be more desirable to update the model in batches instead of as soon as new data points are available. Therefore, we use a block-wise moving window update approach. Considerations to be taken for a MW-PLS scheme are (1) ensuring incoming data are fault free, and (2) optimal selection of update parameters such as moving window size and update step size.

3.2.3 Total PLS Decomposition

To our best knowledge, current literature on recursive or adaptive PLS models lack an in-depth examination of outlier removal in their implementations. This could be justified since outliers in offline model building can be readily identified and removed using a combination of prior knowledge and statistical screening. However, outlier removal and process monitoring are re-

quired online for both the input data and the lab sampled output data. For batch processes such as plasma etching, there are two levels of outlier removal: the first level focuses on removing spikes/dips temporally; the second level removal focuses on excluding abnormal batch runs from the training set. First level temporal spike removal is done routinely by replacing points outside of the 3σ range with a locally estimated average or median. The second level outlier detection is particularly important for PLS based methods due to its least-squares penalty on residual, where a large deviation in $\mathbf{X}$ and $\mathbf{y}$ can cause excessive bias in model parameters. To detect these abnormal runs that affect VM output, we propose using total projected latent structure (T-PLS) monitoring [90] inspired by its application in industrial chemical processes. For multiple-output PLS regression models, an improved version of T-PLS called concurrent projection to latent structure (CPLS) [91] could also be used as a drop-in replacement for the T-PLS method proposed here.

Total projection to latent structure (T-PLS) decomposition further decomposes the principal and residual subspaces in normal PLS monitoring. The algorithm for T-PLS by Zhou et al. [90] is described here for completeness in Table 3.2.

After decomposition, the quality relevant subspaces are separated from the orthogonal components as follows:

$$\begin{aligned} \mathbf{X} &= \mathbf{T}_y\mathbf{P}_y^T + \mathbf{T}_o\mathbf{P}_o^T + \mathbf{T}_r\mathbf{P}_r^T + \mathbf{E}_r \\ \mathbf{y} &= \mathbf{T}_y\mathbf{Q}_y^T + \mathbf{F} \end{aligned} \tag{3.3}$$

where the subscripts y denote output-relevant, o denote output-orthogonal, r denote output-residual. This decomposition offers a more intuitive interpretation of the PLS decomposed subspaces:

Table 3.2: T-PLS decomposition algorithm for single output PLS models

Step	
1	Center the columns of $\mathbf{X}, \mathbf{y}$ to zero mean and scale them to unit variance
2	Run PLS1 algorithm on $(\mathbf{X}, \mathbf{y})$ to estimate $\mathbf{T}$, $\begin{cases} \mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E} \\ \mathbf{y} = \mathbf{T}\mathbf{q}^T + \mathbf{F} \end{cases}$ where $\mathbf{T} \in \mathbb{R}^{n \times A}, \mathbf{q}, \mathbf{P}, \mathbf{T}$ are obtained by PLS1, A is the cross-validated number of principal components.
3	$\mathbf{t}_y = \mathbf{T}\mathbf{q}^T$.
4	$\hat{\mathbf{X}} = \mathbf{T}\mathbf{P}^T, \mathbf{p}_y = \hat{\mathbf{X}}\mathbf{t}_y / \mathbf{t}_y^T \mathbf{t}_y$.
5	$\hat{\mathbf{X}}_0 = \hat{\mathbf{X}} - \mathbf{t}_y \mathbf{p}_y^T = \mathbf{T}_o \mathbf{P}_o^T$, run PCA on $\hat{\mathbf{X}}_\mathbf{o}$ with $A-1$ components
6	$\mathbf{E} = \mathbf{T}_r \mathbf{P}_r^T + \mathbf{E}_r$, run PCA on $\mathbf{E}$ with A_r components, where $A_r < m - A$ is determined using cross-validation.

Output relevant(spanned by $\mathbf{T}_y \mathbf{P}_y^T$): $\mathbf{T}_y$ represent variations related only to $\mathbf{y}$ from the original PLS model.

Output orthogonal(spanned by $\mathbf{T}_o \mathbf{P}_o^T$): $\mathbf{T}_o$ represent variations orthogonal to $\mathbf{y}$ from the original $\mathbf{T}$.

Output residual(spanned by $\mathbf{T}_r \mathbf{P}_r^T$): $\mathbf{T}_o$ is the majority of the original residual $\mathbf{E}$ and $\mathbf{E}_r$ is the left over residual.

Through this decomposition, the subspaces that are not relevant to process monitoring ($\mathbf{T}_o \mathbf{P}_o^T$ and $\mathbf{T}_r \mathbf{P}_r^T$) are isolated from the output relevant subspaces. The resulting monitoring indices are as follows:

$$\begin{aligned} T_y^2 &= \mathbf{t}_y^T \mathbf{\Lambda}_y^{-1} \mathbf{t}_y \sim \frac{A_y (n^2 - 1)}{n (n - A_y)} F_{A_y, n - A_y} \\ Q_r &= \|\tilde{\mathbf{x}}_r\|^2 \sim \frac{S}{2\mu} \chi^2_{2\mu^2/s} \end{aligned} \tag{3.4}$$

where $\mathbf{t}_y$ and $\tilde{\mathbf{x}}_r$ ae the corresponding T-PLS decomposed output-relevant scores and residual, respectively; S and μ are the sample variance and sample

mean of Q_r. Q_r follows a χ^2 distribution and T_y^2 follows a Fisher distribution, which can be then used to calculate the respective control limits given significance level α.

By monitoring only the quality relevant subspaces shown in Equation 3.4, we can effectively reduce the fraction of false outliers removed to maximize data recovery during training; example illustrating the results of T-PLS compared against traditional Hotelling's T^2 is shown in Figure 3.4.

Comparing Figure 3.4(a) with the top two subplots in Figure 3.4(b), we can see that the original Hotelling's T^2 appears to be a superposition of the output-relevant and output-orthogonal Hotelling's T^2 scores; yet, most of the outliers detected by the conventional Hotelling's T^2 are outliers that are orthogonal to output quality. As a result, using T-PLS will reduce the number of outliers removed while ensuring that the remaining outliers do not affect PLS regression performance.

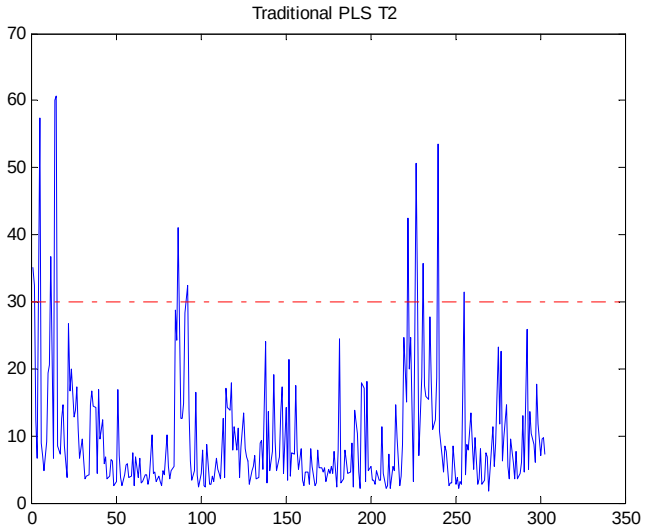
3.2.3.1 Moving Window Update with T-PLS

T-PLS decomposition of an PLS model is done here as an illustrative example, detailed results of this PLS model will be discussed in Chapter 4. The resulting monitoring indices for the training data are plotted in Figure 3.5. The figure shows that two wafer samples are out of the control limit for T_y^2; this outlier can be interpreted as data batches exhibiting abnormal trends in predictor variables that are mostly correlated to response variables. As a result, removing them from the training set improves the PLS model performance. This is a useful property in developing moving window update schemes for PLS models using non-simulated industrial data. Since the data could be contaminated with noise and disturbances, being able to predict them and re-

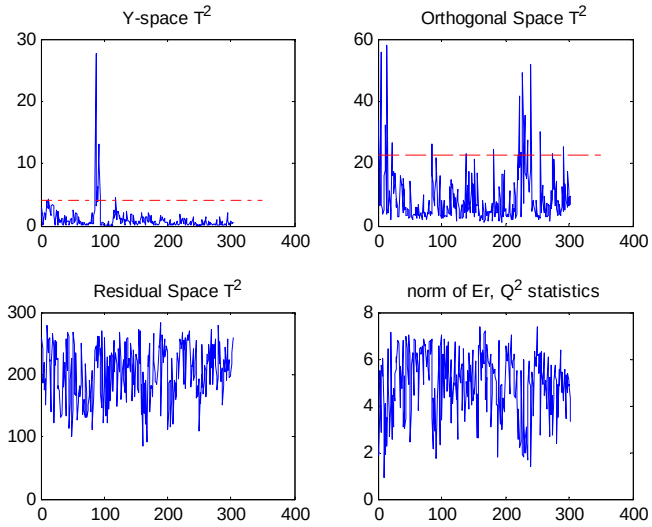
duce their impact in subsequent modeling makes moving window models more robust against noise.

3.2.4 Integrated Framework Results

The exact steps for implementing the integrated VM framework are summarized in Figure 3.6. The framework consists of two phases. In the initialization phase, existing training data is used to create a PLS model and its corresponding outlier detection limits. Following the initialization, the prediction/execution phase of the integrated framework relies only on future data and can be run online. The integrated framework is robust against metrology delays since the T-PLS outlier detection component will provide abnormal data or prediction error warnings prior to collection of actual metrology results. Once metrology results are collected, the data can then be used to update the model. This proposed framework has been applied and tested on a gate etch dataset from a plasma etch tool. The results will be presented in Chapter 4.

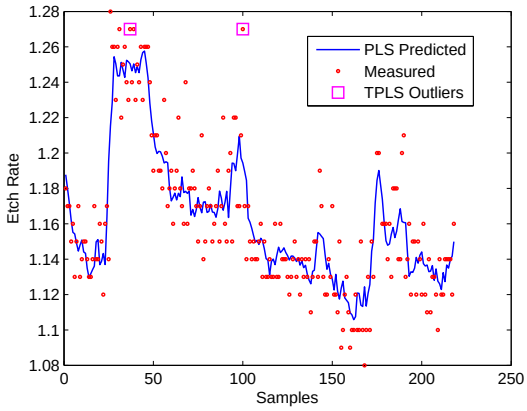


(a)

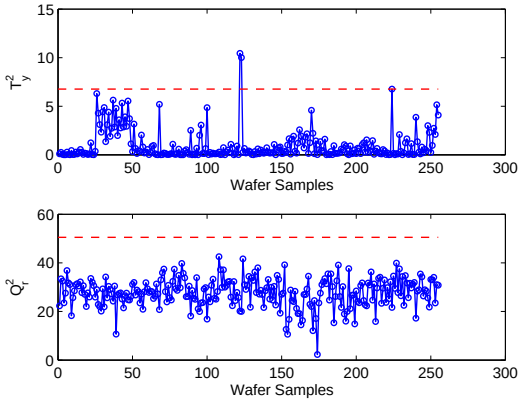


(b)

Figure 3.4: (a) PLS Hotelling's T^2 statistic, (b) PLS decomposed subspace process monitoring indices, top-left: output-relevant T^2, top-right: output-orthogonal T^2, bottom-left: output-orthogonal residual T^2, bottom-right: left-over residual Q_r^2, red dashed line is the 95% significance level



(a)



(b)

Figure 3.5: Time series plot of the training data at the 1st moving window plus the T-PLS fault diagnostics for the training data

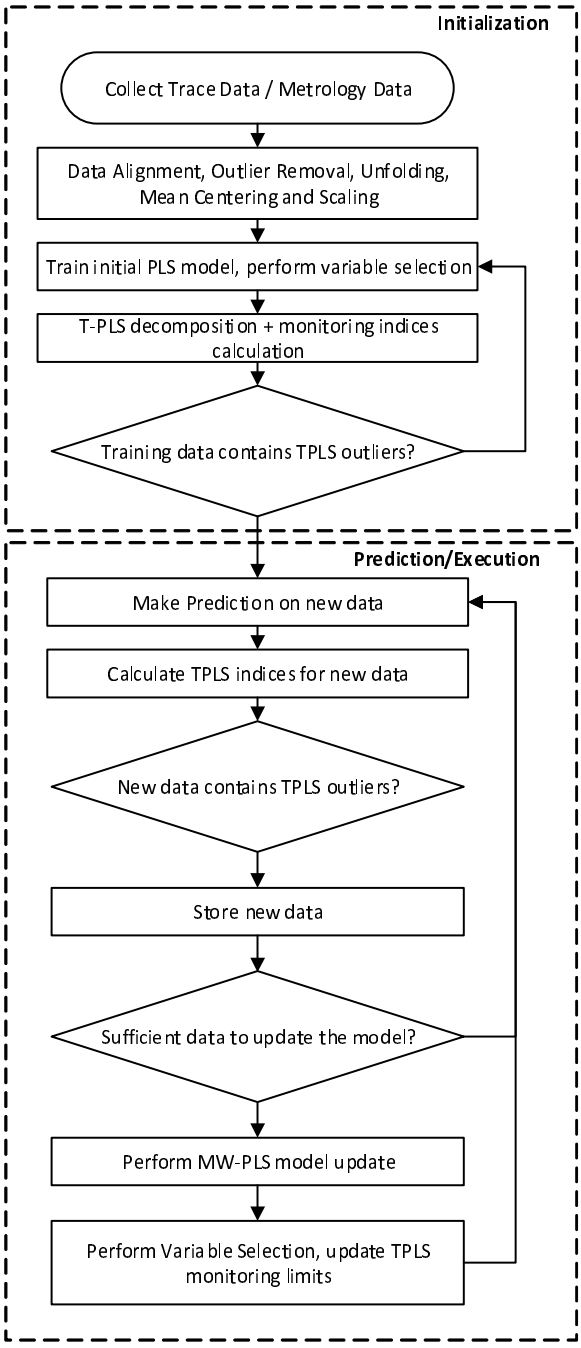


Figure 3.6: Flowchart of the combined integrated framework for online moving window PLS update subjected to T-PLS outlier filtering

3.3 Growing Self Organizing Map Multiple Model Systems for PLS models

3.3.1 Motivation for multiple model systems

Industrial manufacturing systems are becoming increasingly complex and sophisticated due to tighter restrictions on energy efficiency, process integration and equipment utilization. The additional complexity are possible due to advancements in control, real-time optimization and continuous process improvement. As a result, it becomes increasingly challenging to deal with monitoring and diagnostics of faults and events in these systems. Many of these systems operate in multiple production modes which can be characterized by throughput, load, level, recipe and product grades [5]. Traditional methods such as statistical process control are ill-suited for these complex systems. Multivariate data-driven models using traditional techniques such as PCA and PLS also face challenges due to inherent nonlinearity, multiple operating modes and complex fault signatures [77].

Since the data generated from these systems naturally have multi-modal distributions [92]. A "divide and conquer" approach is commonly taken to partition the data into sub-systems that can be approximated using simpler local models. After satisfactory performance is achieved at the local level, the operating mode information can then be used to smooth local model results for better prediction of the overall system. Dunia and Edgar formulated a multi-state PLS framework where transitions between states are smoothed using a k-means clustering smoothing mechanism [26]. Kadlec et al. proposed an incremental learning adaptive soft sensor system that generates operating modes based on prediction residual performance and combines the local model predictions using a Bayesian-inference based validity function approach, the

end result was applied on modeling a catalyst decay problem in a polymerization reactor[17]. There are many published works in this area[93, 94, 16, 95]. Common challenges associated with this approach are:

1. Partitioning of the operating space needs to be determined before local model training
2. Operating modes that do not have sufficient data can lead to numerical instability
3. No simultaneous training of both local model and structure parameters
4. Final model implementation online is a challenge and requires use of heuristics and fuzzy logic rules to combine model results.

In another class of methods, mixing Gaussian distributions and kernel density estimates directly assumes that the underlying data distribution is non-Gaussian and multi-modal *without any easily identifiable class labels*. In these cases, mixture based methods are superior as they do not require apriori definition of a multi-modal structure. Yu and Qin [92] proposed ways a method using finite Gaussian Mixtures. They obtained superior results than conventional multivariate monitoring schemes on the Tennessee-Eastman simulation. Thissen et al. applied finite Gaussian mixture models to industrial data of a fiber spinning process and found much higher sensitivity to faults using Gaussian Mixture Models [96]. Some of the other works using mixtures of distributions are listed here [97, 98, 99, 100]. The main challenges of this approach are:

1. difficult to troubleshoot due to the lack of transparency

2. difficult to implement online due to heavier computational requirements
3. stability of online adaptation of these Gaussian mixtures have not been thoroughly studied.

Batch process data from semiconductor manufacturing are particularly challenging to model due to the following characteristics:

- **high-mix manufacturing**, upstream and downstream processes could change depending on product and production thread. Therefore, unmeasured disturbances could affect process modeling results.
- **large number of measurements**, high dimensional data resulting from unfolding of batch trajectories
- **threaded production**, multiple recipes are ran on the same tool. As a result, some un-popular recipes might have little data available for model building.
- **nonlinearity**, input-output relationships are not linear
- **process degradation**, input-output relationships drift overtime

To improve upon existing work of multiple system model approaches, it is necessary for a multiple model system to satisfy the following requirements:

- Does not rely on class label information
- Does not require large amount of training data to initialize
- Can be easily extended for online adaptation

- Produces interpretable local models and diagnostics information

To this end, the work of growing structure multiple model systems proposed by Liu et al. [94] and Bleakie et al. [15] inspired the development of a hybrid multiple model system that combines latent projection local models (PLS/PCA) models with Growing Self Organizing Map to form a integrated framework suitable for batch data from semiconductor manufacturing.

3.3.2 Growing Self Organizing Map (GSOM)

Growing self organizing map (GSOM) is subclass of self organizing maps (SOM), a popular unsupervised machine learning technique. SOM can be categorized as an artificial neural network. Its purpose is to "map" high dimensional data onto a lower-dimensional space. It is also called a Kohonen map after its creator Teuvo Kohonen [101]. Different from clustering algorithms and other artificial neural networks, a SOM uses a neighborhood function to preserve *topological* properties of the input space. As a result, these topological properties can be used in inferring additional information that could be used in improving prediction, training, and model update. Growing SOM, as the name implies, makes no prior assumption about the size of the map and allows the algorithm to learn the optimal structure at run-time. An example of the topological information in a dataset is given in Figure 3.7, where GSOM is successfully able to identify the spiral topology of the dataset.

Just like other artificial neural networks; GSOM requires training before it becomes useful. The classic GSOM undergoes three stages of training:

initialization where small number of nodes and the topological structure parameters are initialized

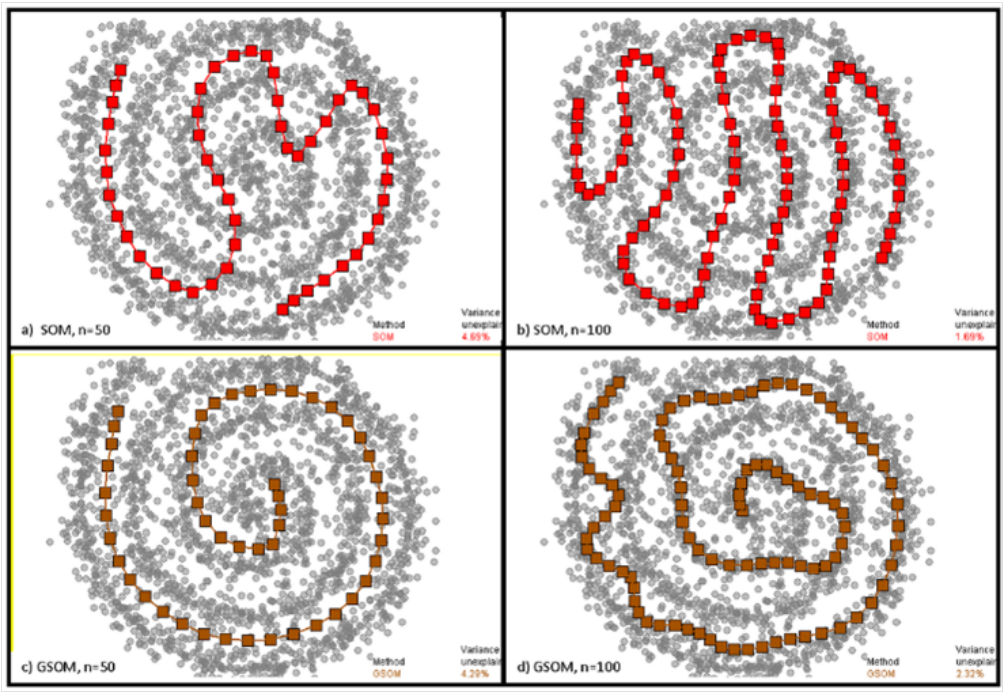


Figure 3.7: Example of SOM vs GSOM under different tuning parameters, figure taken from [4]

growing where training input is presented to GSOM to successively minimize the vector quantization error of the input data. Existing nodes will be relocated to new positions, and new nodes will be added as the algorithm sees fit

fine-tuning final adjustment of node positions according to training data

In the proposed approach, since the structural learning phase (training the GSOM) is integrated with the local learning phase (training PLS and PCA models), both training phases will be discussed together. First, the basic mathematical description of GSOM is discussed.

Given input and output data $X(N \times P)$ and $y(N \times 1)$, we can define a feature vector for each observation

$$\mathbf{s}_i = [s_1\ s_2\ s_3\ s_4 \ldots s_K]_i = [P(\mathbf{x}_i, \mathbf{x}_{i-1}, \ldots), Q(\mathbf{y}_i, \mathbf{y}_{i-1}, \ldots)] \tag{3.5}$$

where $P(\cdot)$ and $Q(\cdot)$ are features selection functions. These functions can either reduce feature space dimension by projecting X onto a lower dimensional space, expand feature space by introducing nonlinear terms or lagged variables. In practice, using only X variables with no transformation is a good starting point for exploratory analysis.

Once feature vectors are defined, SOM is then trained in using the training features. A trained SOM consists of nodes and their topological neighborhood data. This can be represented graphically in Figure 3.8. At each node there is a codebook vector and a local model. The codebook vector contains the position information of the node and has the same dimension as training feature inputs:

$$\xi_m := [s_1\ s_2\ s_3\ s_4 \ldots s_K] \tag{3.6}$$

where $m \in (1 \ldots M)$, M is the number of nodes in the GSOM.

To partition the feature space into sub-regions, the space belonging to node m is defined as:

$$V_m = \{\mathbf{s} : \|\mathbf{s} - \xi_m\| \leq \|\mathbf{s} - \xi_i\|, \forall i \in (1, M), i \neq m\} \tag{3.7}$$

To find the best matching node m (called the best matching unit in SOM terminology) given an feature vector $\mathbf{s}$, we compare the vector quantization error of $\mathbf{s}$ with all possible SOM nodes $\xi_m, m \in (1, M)$ and look for the node with the smallest error:

$$BMU(\mathbf{s}) = \underset{m}{\operatorname{argmin}} \|\mathbf{s} - \xi_m\|, m \in (1, M) \tag{3.8}$$

The topological relationship of the SOM is stored in a symmetric binary matrix, called the adjacency matrix $\mathbb{A}$ $(M \times M)$, where M is the number of nodes in the SOM. A node i is said to be connected to node j if $\mathbb{A}(i, j) = 1$ (in this case, $\mathbb{A}(j, i)$ also equal to 1). The adjacency matrix can be visualized as connections between nodes on a SOM. Figure 3.8 illustrates the adjacency matrix visually using an example with 10 nodes. Each node is numbered with an index. With respect to node 1, nodes 2,3 are its primary neighbors (immediately adjacent) while nodes 4,5 are its secondary neighbors (separated by one node in between).

3.3.3 Growing Structure Multiple Model Systems - GSMMS

Most of multiple modeling techniques divides the training of the model into two independent phases: structural learning and local model learning as shown in Figure3.9. Albeit being straightforward and simple to implement, there are some disadvantages with this approach. For instance, the number of

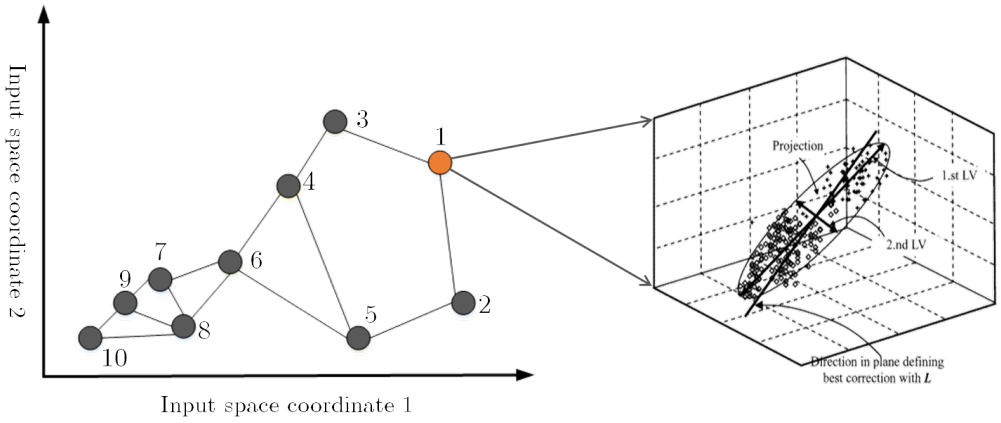


Figure 3.8: Parameters that need to be specified in a GSMMS system: The GSOM network, and local model at each node

local models is usually fixed and cannot be changed easily. In addition, there is no feedback of local model performance to help better guide clustering. However, in our proposed approach, the two training phases are conducted simultaneously as shown in Figure 3.10. The overall algorithm initializes with a small SOM and then iteratively adapts the SOM structure and the corresponding local models in attempt to fit the given input. There are two loops in the proposed training scheme inspired by GSOM training. The outer loop incrementally add nodes until the global fitting error reaches a minimum (corresponding to the growing phase in GSOM training). The inner loop adjusts the codebook of each node (ξ_m) during training to find the optimal placement of node codebooks. We will first discuss the offline batch training of GSMMS in detail and then discuss online implementation adjustments.

3.3.3.1 Initialization

To initialize GSOM training, a initial set of SOM structure parameters need to be generated (the initial node codebooks ξ_m and the initial adjacency

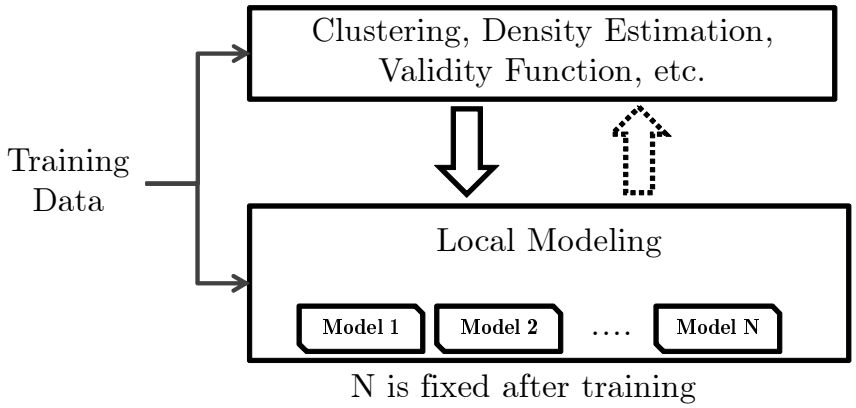


Figure 3.9: Basic flow of information in conventional multiple model systems

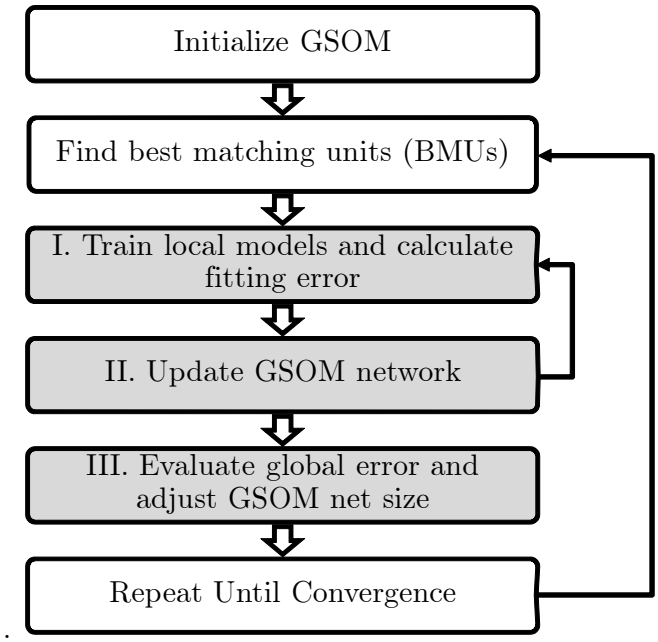


Figure 3.10: Overall training procedure for GSMMS models

matrix $\mathbb{A}$). Three initial nodes with codebook vector ξ_m randomly sampled from the training input is usually used to seed the GSOM. This size is chosen as it offers optimal flexibility in adapting to the optimal network topology. The adjacency matrix $\mathbb{A}$ of this initial network is defined as:

$$\mathbb{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{3.9}$$

This definition of $\mathbb{A}$ connects all three nodes with each other, forming a triangular network.

After initialization, the BMU of each training input is calculated using Equation3.8. This returns a vector of **BMU** for each observation.

$$\mathbf{BMU} = \{BMU_i, i \in (1, N_{\text{samples}})\} \tag{3.10}$$

An observation is said to belong to node m if the BMU of this observation is equal to m. It is important to note that as the SOM undergoes training, the membership association of each observation will change. The vector **BMU** stores the most appropriate node for each observation in the training input at the current iteration. A Gaussian weighing function $\mathbf{w}_m(\mathbf{BMU}, \mathbb{A})$ can be defined to calculate the weight of each observation with respect to the specific node m:

$$w_m\left(BMU_i, \mathbb{A}\right) = \exp\left(\frac{-dis\left(m, BMU_i, \mathbb{A}\right)}{2\sigma^2}\right), m \in [1, M] \tag{3.11}$$

where σ^2 is a parameter that defines the effective range of the weighting function, and $dis(m, BMU_i, \mathbb{A})$ is the topological distance specific to the current network configuration $\mathbb{A}$. This distance is best explained visually through Figure 3.11. For primary neighbors of node m, the value is 1, for secondary

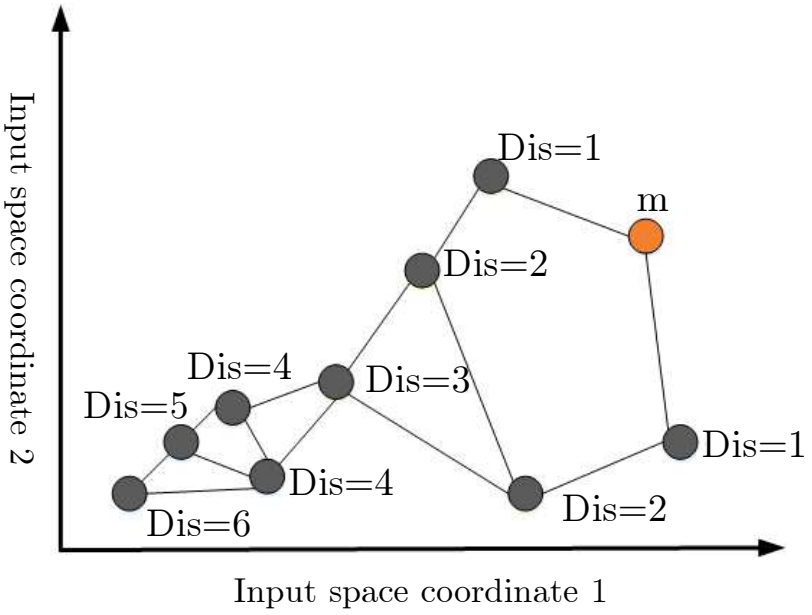


Figure 3.11: Topological distance of each node with respect to node m

neighbors, the value is 2 and so on. This calculation can be carried out efficiently using Breath-first algorithms commonly used in graph theory [102]. The weighting function Equation 3.11 discounts neighbors that are far away from the current node m during training. In addition, the weighting scheme can also be used to vote on final prediction results during model execution.

3.3.3.2 Sample weighted PLS local model fitting

Before going into details of the structural parameter training , the local model fitting algorithm, sample weighted Partial Least Squares, need to be introduced. Each observation and its weight are both inputs to the algorithm. This enables specifying the impact of different training observation on the final PLS model. In GSMMS, since the neighborhood topology carries information

of node membership and neighborhood association, the sample weighted PLS algorithm can be used to integrate these topological information into local model fitting. The sample weighted PLS can be expressed functionally as:

$$[P, T, Q, U, W, \beta] = \text{weighted}PLS\,(X, y, w) \tag{3.12}$$

where $w_i \in [0, 1]$ is a vector of weights that has the same dimension as output variable y.

PLS decomposition is traditionally performed using the Nonlinear Iterative PArtial Least Squares (NIPALS) algorithm; however, it is difficult to account for weighting of observations as it iteratively calculates scores, weights and loadings of each component. Yet, the weighted PCA algorithm accomplishes sample weighting by accounting for the sample weights in calculating the covariance matrix. Following this idea, we discovered that the SIMPLS algorithm for PLS utilizes input-output covariance deflation to arrive at the each successive components [53]. The following modified SIMPLS algorithm shown in Table 3.3 is able to account for the different sample weights in the training input. The sample-weighted PLS requires mean-centering of variables and does not enforce unit variance scaling. However, if the range of magnitudes for the input-output variables is large, the best practice is always to scale the variables to the same order of magnitude prior to applying the PLS algorithm.

3.3.3.3 Combined local model and SOM fitting

After SOM initialization, the inner loop iteration updates the codebook vectors (ξ_m) and the local model parameters ($\mathbf{P}, \mathbf{W}, \beta$ for PLS, $\mathbf{P}$ for PCA) of each node iteratively:

Table 3.3: Algorithm details of sample weighted PLS algorithm

Step	
1	Construct weighting matrix $\Omega = diag(w_1, w_2, \ldots, w_P)$
2	Mean center using weighted mean for X, y matrices as follows: $X_s = X - \bar{X}', \bar{X}' = [1, 1, ..., 1]^T [\bar{x}_1, \bar{x}_1, \ldots, \bar{x}_M]$ $y_s = y - \bar{y}'$ $\bar{x}_m = \sum_{n=1}^{N} w_n x_{nm} / \sum_{n=1}^{N} w_n$ $\bar{y}' = \sum_{n=1}^{N} w_n y_n / \sum_{n=1}^{N} w_n$
3	Compute weighted cross product $\mathbf{S} = \mathbf{X_s^T \Omega y_s y_s^T \Omega X_s}$ Let $a = 1$
4	If $a = 1$, calculate $[\mathbf{r}, \mathbf{s}, \mathbf{c}] = SVD(\mathbf{S})$ If $a > 1$, calculate $[\mathbf{r}, \mathbf{s}, \mathbf{c}] = SVD(\mathbf{S} - \mathbf{P}(\mathbf{P^T P})^{-1}\mathbf{P^T S})$
5	Get weights $\mathbf{r}$ = first left singular vector
6	Compute scores $\mathbf{t} = \mathbf{X_s r}$
7	Compute x-loading $\mathbf{p} = \mathbf{X_s^T \Omega t}/(\mathbf{t^T \Omega t})$
8	Compute y-loading $\mathbf{q} = \mathbf{s} \cdot \mathbf{c}/(\mathbf{t^T \Omega t})$
8	Store $\mathbf{r}, \mathbf{t}, \mathbf{p}, \mathbf{q}$ and $\mathbf{p}$ into $\mathbf{R}, \mathbf{T}, \mathbf{P}, \mathbf{Q}$, respectively
9	Let $a = a + 1$ until $a = A$ and go to Step 4
10	Compute regression coefficients $\beta_{PLS} = \mathbf{RQ^T}/(\mathbf{T^T \Omega T})$

- For $k = 1 : K_{max}$, perform the following:

  1. For each local model m, perform *sample-weighted* PLS regression using the entire training dataset and sample weights calculated from $\mathbf{w}_m(\mathbf{BMU}_i, \mathbb{A})$, the number of latent components in the PLS model is a fixed parameter for all nodes.

  2. Calculate the local modeling error as the root mean squared prediction residual of each local model m from training data subset $j \in \mathbf{J_m} := \{\mathbf{i}|\mathbf{BMU_i} == \mathbf{m}\}$ (all the observations assigned to node m):

$$e_m = \frac{\sqrt{\sum_{j \in \mathbf{J_m}} \left(\hat{y}_j - y_j\right)^2}}{\text{count}\left(\mathbf{J_m}\right)} \tag{3.13}$$

  3. Update the node codebook ξ_m using training data belonging to node m':

$$\xi_m\left(k+1\right) = \xi_m\left(k\right) + \sum_{m'=1}^{M} \alpha_{m'}\left(\mu_{m'} - \xi_m\left(k\right)\right) \tag{3.14}$$

  where K_{max} is a SOM standard parameter that defines the maximum number of passes, $\mu_{m'}$ is the vector mean of training data in subset $J_{m'}$ and $\alpha_{m,m'}$ is the learning rate, the learning rate is defined as the product of the following factors:

  iterative decay: $h_0 \exp\left(-\frac{k}{\lambda}\right)$, this is a standard SOM construct that reduces movement of nodes gradually to eliminate excessive movement due to BMU membership association changes.

  local error ratio: $\frac{MSE_{m'}}{MSE_{m'}+MSE_m}$, this ratio is tailored specifically for soft competitive learning of SOM to assign higher weight for observations that have higher local fitting error.

topological distance: $\exp\left(-\frac{dis(m',m)}{2\sigma^2(m')}\right)$, the topological distance ensures data from neighborhoods further away will have less impact.

neighborhood size ratio: $\frac{N_m}{N_m+N_{m'}}$, this ratio accounts for differences in the number of observations available to each node. A node with large amount of observation samples will have higher weight to attract nearby nodes closer toward itself.

- Go to first step and repeat until convergence or $k == K_{max}$. Convergence is defined as when the change in the ξ_m is less than predefined tolerance value ϵ (a very small value).

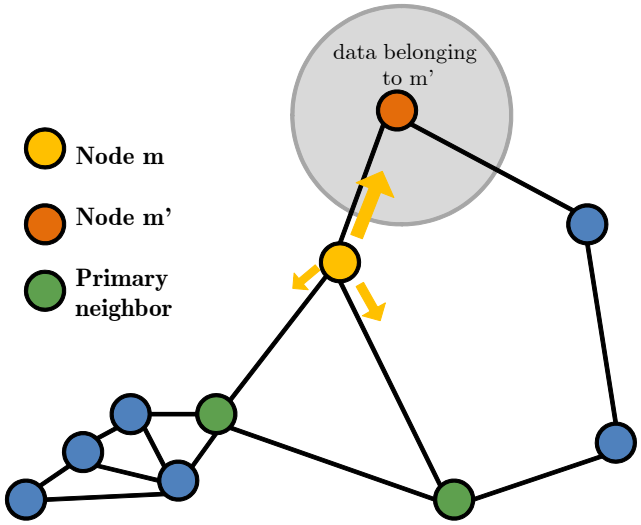


Figure 3.12: GSMMS inner loop iterative update of ξ_m as described in Step 3

In Step 3 of the update algorithm, the learning rate balances the trade offs between training sample size, local model fitting error, neighborhood topology and SOM iteration decay. These factors can be visualized in Figure 3.12.

The orange arrows projecting outward from node m shows directions and magnitudes corresponding to the update vector $\alpha_{m'}\left(\mu_{m'} - \xi_m\left(k\right)\right)$. In a highly nonlinear region such as m' , the local PLS model for m' would have large $MSE_{m'}$, this results in higher weight being assigned in the direction of m'. The higher weight compensates for the high local fitting error around this node by drawing neighboring nodes closer. In a similar fashion, if a local node is assigned many training observations in its partition, the node will carry heavier weight and attract nearby nodes. As a result, data-rich regions (in terms of training data availability) will attract more SOM nodes, leading to finer partitions with better local prediction.

To improve the computational efficiency, $\mathbf{BMU}_k$ from the previous iteration can be stored and monitored for changes in the current iteration. PLS regression only need to be recomputed for local models that had changes in their membership. Minuscule movements and fine tuning of node location ξ_m usually do not result in membership changes. This modification improves the speed of the update step by as much as five times in practice.

3.3.3.4 Network growth mechanism

After the SOM converges, global fitting error is defined as the root mean squared prediction residual of every observation:

$$e_{global} = \frac{\sqrt{\sum_{m=1}^{M}\left(\sum_{i\in\mathbf{J_m}}\left(\hat{y}_i - y_i\right)^2\right)}}{N} \tag{3.15}$$

The global fitting error is used in evaluating the marginal return on prediction performance by having the additional SOM node. To achieve the most parsimonious model, if the improvement in global fitting error does not exceed a

predefined threshold (for example, 5%), then the simpler SOM structure of the previous iteration is preferred over the existing more complex SOM structure. The outer loop then terminates and the SOM training is complete. Otherwise, a new node is added.

Inspired by the GSOM applications in Liu et al. [82], node insertion in our proposed method places new nodes near regions with the highest local fitting error using the following heuristic:

1. Find the node with the largest MSE_m, call this node p
2. Find the furthest node q from its primary neighbors based on Euclidean distance of the codebook vectors: $q = \underset{m'}{\operatorname{argmin}} \|\xi_p - \xi_{m'}\|$, where m' belongs to the set of primary neighbors
3. Insert a new node at the mid-point $\xi_{new} = \frac{\xi_p + \xi_q}{2}$
4. Connect the new node with neighbors of nodes p and q in the adjacency matrix $\mathbb{A}$

The above procedure is illustrated graphically in Figure 3.13, where the 11th node is added and connected to the surrounding neighbors of nodes p and q. After node addition, the algorithm would return the inner loop to update the SOM codebook and the local model parameters.

3.3.3.5 PCA local model modification

While the above discussion of the GSMMS framework focuses on multiple model system using PLS models as local models, this framework can also be easily adapted to use PCA local models for fault detection and process monitoring.

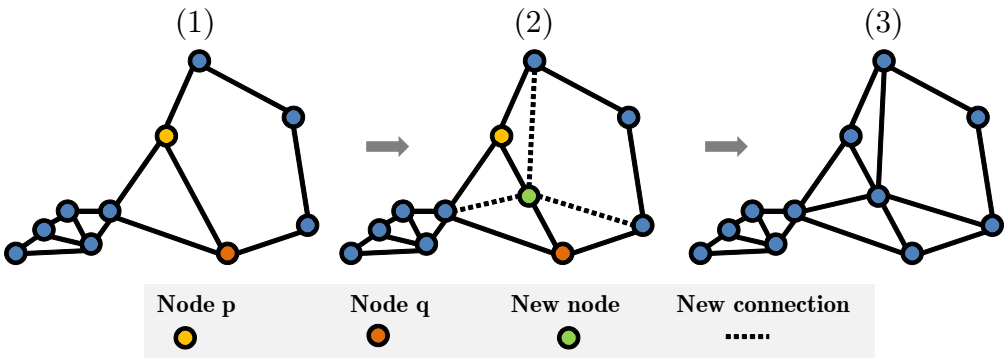


Figure 3.13: GSMMS node insertion into region with highest local fitting error

To train PCA models as local models, the overall training scheme remains unchanged. However, the local model fitting performance for PLS models is measured by the mean squared prediction error of that local region. Since PCA is an unsupervised learning technique, a new local and global fitness measure need to be defined. A PCA decomposition gives the following result,

$$\mathbf{X} = \hat{\mathbf{X}} + \tilde{\mathbf{X}} = \mathbf{T}\mathbf{P}' + \tilde{\mathbf{X}} \tag{3.16}$$

where $\hat{\mathbf{X}}, \tilde{\mathbf{X}}$ are the principal and residual matrices respectively. The magnitude of the residual matrix is inversely proportional to the magnitude of the principal matrix, where the number of components controls the trade-off. Because the number of components for each local model is specified to be the same, the magnitude of the residual matrix $\tilde{\mathbf{X}}$ is a measure of the local fitting performance. We define the local fitting error of node m as the mean Frobenius norm of the residual matrix:

$$e_m = \frac{\left\|\tilde{\mathbf{X}}_m\right\|}{N_m} = \frac{\sqrt{\sum_{i=1}^{N_m}\sum_{j=1}^{N_m}\left(\tilde{x}_{i,j}\right)^2}}{N_m} \tag{3.17}$$

The global fitting error is defined as the weighted mean of the residual matrix norms. The same criteria are applied to ensure that there is acceptable improvement (5%) in global fitting error for the additional node; otherwise, the training terminates at the previous SOM structure.

$$e_{\text{global}} = \frac{\sum_{m=1}^{M} N_m e_m}{N} \tag{3.18}$$

3.3.4 Online update of GSMMS models

One of the key benefits of using GSOM with local modeling is that GSOM offers the flexibility of adaptation. The GSOM construct allows for easy ways to alter the structural of the network (by adding or deleting nodes). However, in GSMMS, the nodes are associated with local models and thus impose additional constraints. The online update of our proposed GSMMS system is based on GSOM but undergoes several modifications.

First, an initial GSMMS model needs to be trained using offline training data. The online update of the GSMMS model therefore takes the following functional form:

$$GSMMS_{k+1} = OnlineUpdateFcn(GSMMS_k, \mathbf{S_{k+1}}, \mathbf{y}_{k+1}) \tag{3.19}$$

where k is a counter for the number of updates performed, $GSMMS_k$ is the total GSMMS system including the SOM structural parameters and all the local models. The $\mathbf{S_{k+1}}$ is the input features of the newly collected online data and $\mathbf{y}_{k+1}$ is the new output vector.

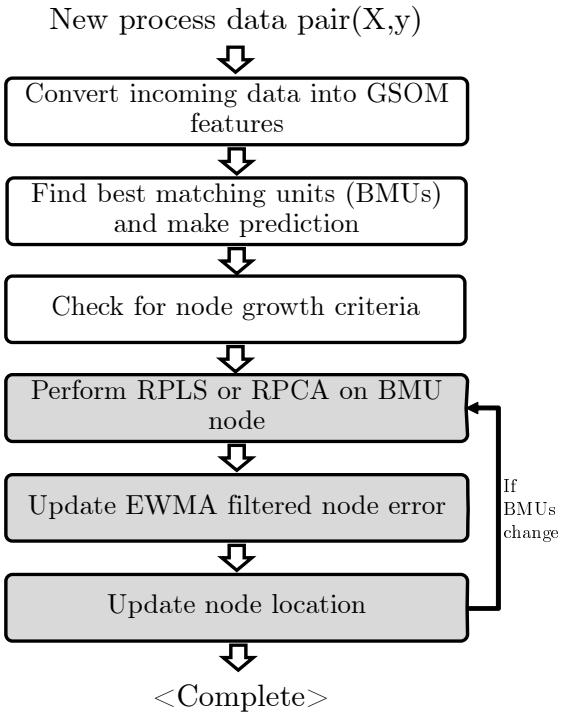


Figure 3.14: GSMMS local parameter update algorithm

3.3.4.1 Local parameter update

Figure 3.14 lists the overall algorithm to update the GSMMS model. The update can be separated into two steps. The first step is to update the local models of nodes that are activated by the new inputs. The local model coefficients can be updated efficiently using recursive PLS and PCA algorithms. Prediction errors for the new feature input are also obtained during the model update. Since the original training data is not available during online adaptation, the new prediction errors are combined with the historical prediction errors using an exponentially weighted moving average filter as follows:

$$e_m^{EWMA}(k) = \lambda w_m(\mathbf{s}(k))\, e_m^{EWMA}(k-1) + [1 - \lambda w_m(\mathbf{s}(k))]\, \|e_m(k)\| \quad (3.20)$$

The forgetting factor λ is tuned to allow the effective sample size of the local node to remain unchanged from the initial training. The effective sample size is related to the forgetting factor λ through the infinite sum identity of a geometric series:

$$N_{eff} = \sum_{k=0}^{\infty} \lambda^k = \frac{\lambda}{1-\lambda}, \lambda \in (0,1) \quad (3.21)$$

Once the local errors are updated, the codebook of each node are then moved to the new location using a modified version of the codebook update equation based on Equation 3.14.

$$\xi_m(k+1) = \xi_m(k) + \tilde{\alpha}(k)\,[\mathbf{s} - \xi_m(k)] \quad (3.22)$$

where $\tilde{\alpha}$ is the recursive learning rate, $\mathbf{s}$ is the input feature vector, and ξ_m is the codebook of the node being updated.

Similar to the batch learning rate α, the recursive learning rate ($\tilde{\alpha}$) is composed of three components:

local error ratio: $\frac{e_{m'}^{EWMA}}{e_{m'}^{EWMA}+e_{m}^{EWMA}}$, this ratio is assigns higher weight to regions with higher local fitting error using the EWMA filtered error measure.

topological distance: $\exp\left(-\frac{dis(m',m)}{2\sigma^2(m')}\right)$, the topological distance ensures data from neighborhoods further away will have less impact.

dampening factor: $\exp\left(-\frac{k}{\tau}\right)$, the k is the number of samples assigned to the node since the last node addition/removal. This is analogous to the offline scenario where the number of passes reduces the learning rate.

3.3.4.2 SOM structural parameter update

The above recursive update rules, when applied, are able to update the local model coefficients and handle slow drifts of operating space over long periods of time. However, in cases where abrupt changes occur in the system, the EWMA filtered local errors and the iterative update equations for the structural parameters are not able to respond quickly enough. In these cases, new nodes need to be inserted near regions with the highest predictive error. Here we briefly discuss two ways this can be implemented, these node insertion heuristics all follow the same general principles that were applied during batch training.

The first method is to monitor the global prediction error as new update samples are made available. A spike in global prediction error of the network would indicate that the current nodes are not able to predict the more recent events in the process. In this case, a new node can be added near the node that is closest to these update inputs. If there is an significant improvement in reducing the prediction error, then the new network structure is kept.

The alternative is to monitor the distance (Euclidean) of new feature

inputs from the existing SOM node codebook vectors. This is an efficient calculation since the distances are already computed during the assignment of the best matching units. The update step then monitors this distance against a threshold. When the threshold is exceeded, it would be likely that this node will not be predicted well by any of the existing local models. In this case, a new node can be inserted at the midpoint between the new observation and its closest neighbor to initialize a new region.

3.3.5 Prediction and fault diagnostics

Combining local model diagnostics and prediction for any multiple model system is a challenging task. It is common that local models carry excessive uncertainty and yield predictions that are unreliable. In addition, transition regions between operating states are also difficult to predict even with multiple model systems because of the dynamic and nonlinear nature of transitions.

As a result, in our proposed framework, the local model prediction and the SOM node information will be combined using a procedure summarized in Figure 3.15. The best neighborhood is selected as the best matching unit and its immediate primary neighbors. The final predictions are then calculated based on heuristics outlined as follows:

- Calculate the T^2 and Q statistics of the best matching unit m for the incoming input.
- If these monitoring indices are below their alarm limits, then use the best matching unit local prediction $\hat{y}_m$ as the final output

- If the monitoring indices are alarming, then use the neighborhood weighted average prediction as follows:

$$\hat{y} = \frac{\sum_{i=1}^{I} w_i \hat{y}_i}{\sum_{i=1}^{I} w_i}$$
$$w_i = \exp\left(-\frac{dis(m,i)}{2\sigma^2(k)}\right) \tag{3.23}$$

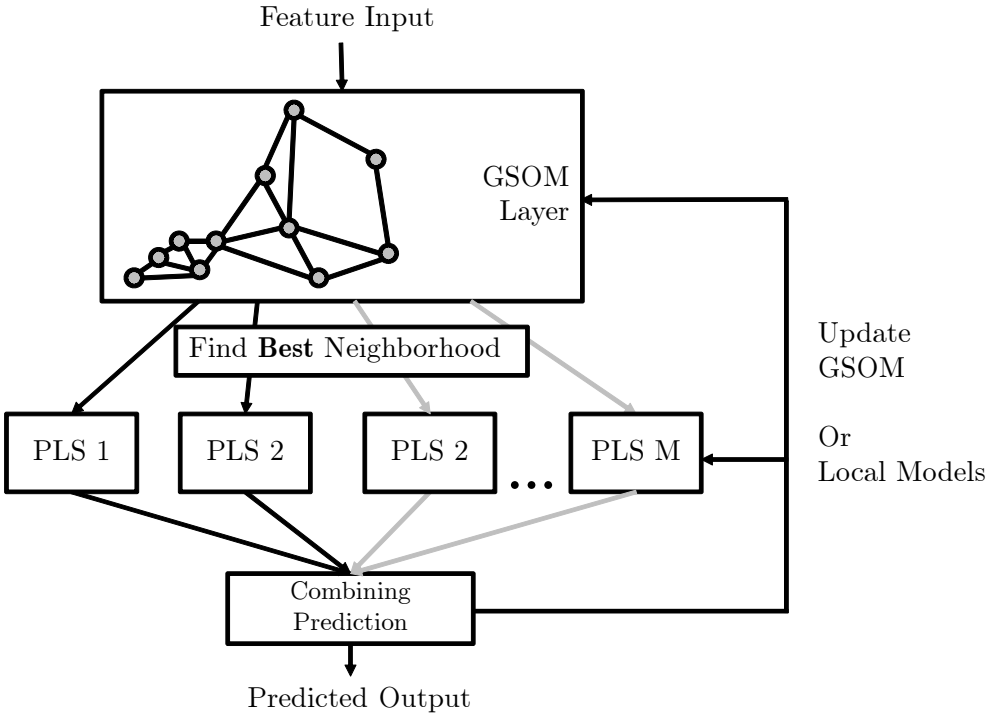


Figure 3.15: Schematic of the prediction and the online adaptation of proposed GSMMS framework

3.3.6 Simulated case studies

Three case studies were performed using simulation to demonstrate the effectiveness of proposed method:

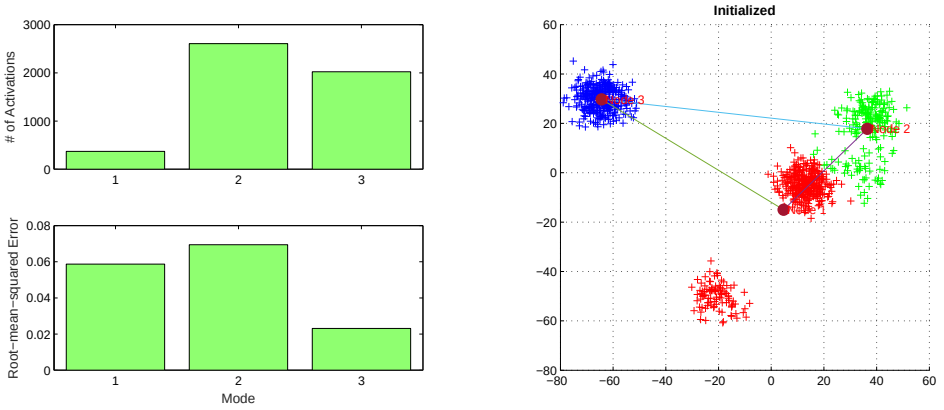
1. Five multiple operating state steady-state input-output data
2. Nonlinear input-output steady-state operating data
3. Five multiple operating state with overlapping boundaries input-output data

In these simulated case studies, we attempt to predict the final output of a simulated process with 10 inputs and one output. The simulated process with multiple operating states is generated in MATLAB. The steady state data is generated using an integrated moving average time series subjected to occasional unmeasured disturbances. Three inputs are collinear variables based on linear combination of the other seven inputs. The final output is generated as a linear function of the inputs under white noise. While this system does not simulate the dynamic behavior during state transitions, it represents a typical set of input-output data for continuous steady-state systems.

The dataset is simulated for 15,000 samples. 5,000 samples are used as training data and the other 10,000 are used as testing inputs. The initial GSOM inputs has been reduced using a global PCA model to the first three components, while the local model inputs still uses the ten raw simulated inputs. This global PCA model captures 85% of the total variance in the process inputs.

Figure 3.16(a) lists the GSOM diagnostics of a model just after initialization, the top plot is the number of activations for each node and the bottom

plot shows the root mean squared error of each node. The node locations are initialized randomly and the local node error are calculated using the membership association at the time of initialization. Figure 3.16(b) shows the current network structure as denoted using the first two scores of a principal component projection for better visualization. Higher dimensional plot is much more difficult to interpret visually.

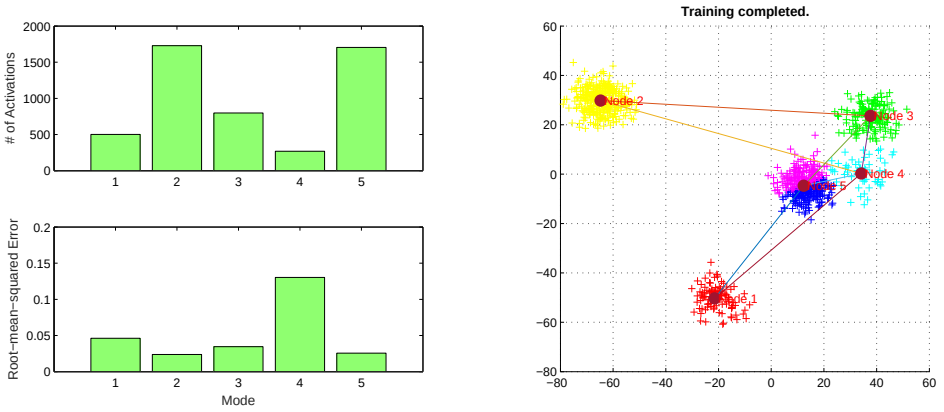


(a) GSOM Diagnostics

(b) GSOM codebook location and training inputs

Figure 3.16: GSOM node error and codebook location just after initialization

After the GSOM undergoes batch offline training, the structural parameters and the diagnostics are shown in Figure 3.17. The algorithm determined that five nodes is required to model the input data, which matches the number of operating modes specified in the simulation. In addition, the number of activations and the root mean squared error of each node are inversely correlated. Nodes with the highest number of activations have lower root mean squared errors, indicating that their membership association have been optimized for prediction.



(a) GSOM Diagnostics

(b) GSOM codebook location and training inputs

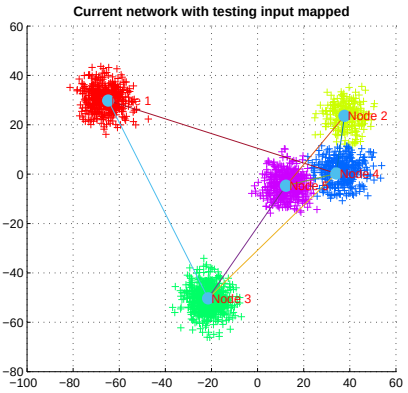
Figure 3.17: GSOM node error and codebook location after training has been completed

Figure 3.18 shows the testing dataset mapped onto the trained SOM. Since this simulated dataset did not include a drifting disturbance, the model is able to accurately predict the results with a R^2 value of 0.91.

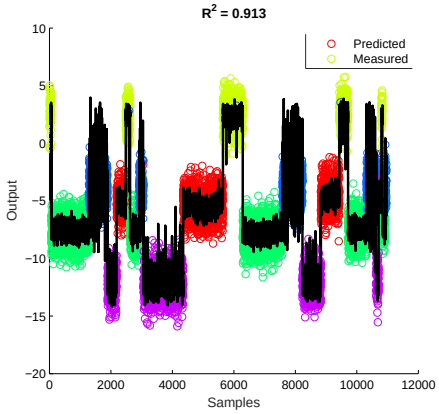
In the second case study, simple steady-state process nonlinearity is also simulated by setting the output as follows:

$$y = \beta_1 \exp\left(-x_1/x_2\right) + \beta_2 x_3 x_4 + \beta_3 {x_5}^2 + \beta_4 x_6 + \beta_5 x_7 + \beta_0 + \varepsilon \qquad (3.24)$$

The input and the generated output are plotted in Figure3.19(a). The trained network structure is shown in Figure 3.19(c), and the final testing dataset prediction is shown in Figure 3.19(d). In this case, the GSOM divided the input space into five nodes to better approximate the nonlinearity in the system. The prediction performance on the testing dataset resulted in a R^2 value of 0.81.



(a) Testing dataset score plot

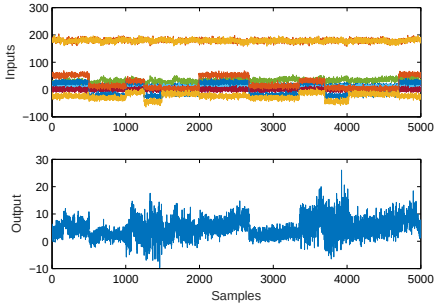


(b) Predicted and actual values of the testing dataset

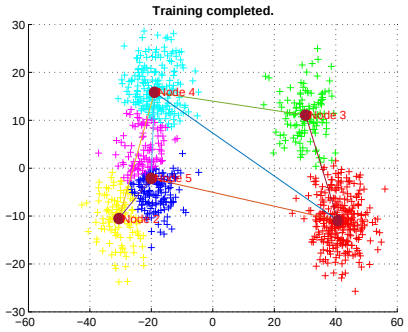
Figure 3.18: Testing data score plot and the prediction results using the GSOM model

The third case study simulates a scenario where there are significant operating mode overlap. The operating mode boundaries are difficult to identify, causing degradation of prediction performance if each mode is treated as independent from other modes. The boundary overlaps are simulated by reducing the variance in the model inputs among different operating modes. The GSMMS trained network structure and the simulated input data are shown in Figure 3.20. The GSMMS training algorithm optimized the network size to be six nodes (greater than five modes used in simulation). The finer partition allows GSMMS to isolate regions of high overlap and improve on prediction in regions with less overlap.

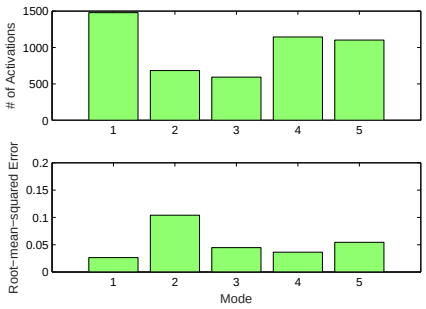
The prediction, training coefficient of determination (R^2)and prediction errors are tabulated in Table 3.4. As references for comparison, two additional model performances are included. The baseline model represents the current industrial practice of assuming contiguous blocks of training data from a single



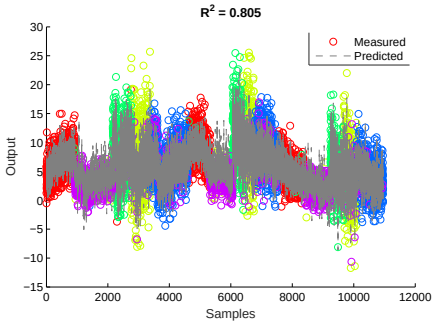
(a) Simulated inputs and nonlinear output time series plot



(b) SOM network structure after training



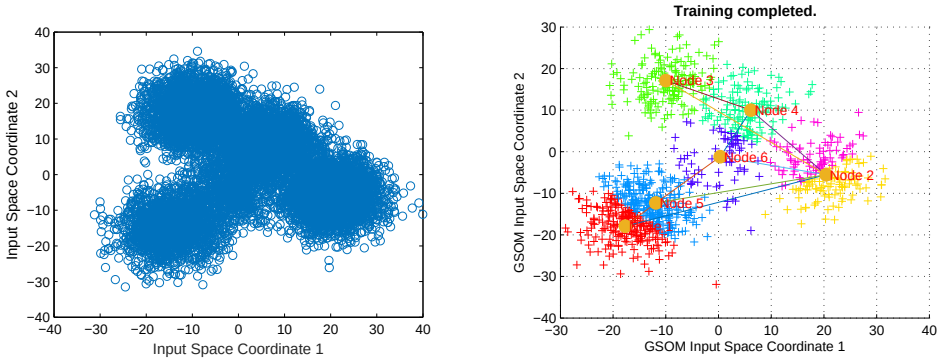
(c) GSOM diagnostics information



(d) Predicted and measured outputs for the testing dataset

Figure 3.19: Training and testing results of nonlinear simulated case study

operating mode and creates only a single PLS model. The ideal case assumes perfect knowledge about the operating mode information and creates a PLS model for each separate mode based on the true operating mode information. The GSMMS model behaves close to the ideal case performance for cases 1 and 2. In case 3 with excessive data overlap, the GSMMS suffers a drop in prediction performance, but it is still performing better than single mode models. As a result, the multiple model system approach is superior if the development process can be simplified and streamlined for practical use in industrial applications.



(a) Score plot of training data (b) SOM partitioning of the input space

Figure 3.20: Score plot and GSOM structure for a case with operating mode overlap

Table 3.4: Prediction performance of the GSMMS framework in three simulated case studies

Model	**# of Nodes**	**Testing R2**	**RMS Error**
	Case 1 - Operating Mode with Clear Separation		
Ideal case	4	0.97	1.01
Baseline	1	0.7	3.04
Piecewise GSMMS	4	0.97	1.01
	Case 2 - Multiple Operating Mode with Nonlinear Output		
Ideal case	4	0.98	0.99
Baseline	1	0.76	3.53
Piecewise GSMMS	5	0.91	2.38
	Case 3 - Multiple Operating Modes with Overlapping Modes		
Ideal case	5	0.96	1.09
Baseline	1	0.52	4.23
Piecewise GSMMS	6	0.85	2.13

3.3.7 Tennessee-Eastman process simulated case study

The Tennessee Eastman Chemical Process simulation developed by Downs and Vogel [103] have been used in many fault detection and process control studies. In this case study, we use the revised TEP code re-written for MATLAB and Simulink developed by Dr. Ricker. In summary, there are 16 standard process measurements of flow rates, temperatures, pressure, compressor power consumption, and tank levels. The simulated plant is under PID closed loop control for some of the key process variables. Faults are then manually injected into the stable steady-state plant. There are a total of 20 fault scenarios. For each fault scenario, the plant was configured to operate normally for the first 300 samples. The fault occurs at sample 300 and then progresses onward until the plant destabilizes and trips. The detailed cause and type of faults injected can be found in [44]. The goal of applying GSMMS is to diagnose and identify the fault at the earliest possible time. Since each fault is injected at sample 300, the fault detection response speed is defined as the elapsed time since sample 300 before the first alarm is generated. In addition, standard fault detection metrics of false alarm and missed alarm rates will be calculated as well.

To apply GSMMS-PCA for the TEP, a month of steady-state normal operating data was first generated. A global PCA model (call this PCA_{normal})was first trained on the normal operating data to define the normal operation. We then apply the loadings of the PCA_{normal} model to the 20 faulty cases to obtain their corresponding scores. Figure 3.21 shows the score plot of the faulty scenarios being projected onto the first two principal components of the PCA_{normal}. The scores of the projection then represents deviation of current behavior relative to the initial normal operating period defined using

PCA_{normal}.

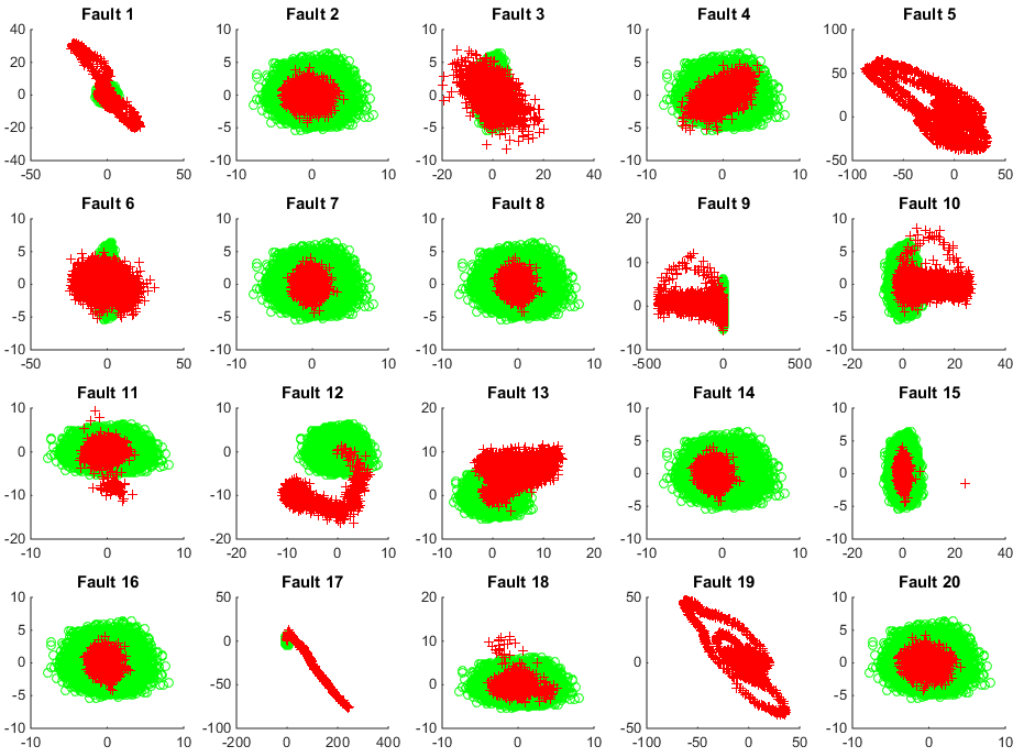
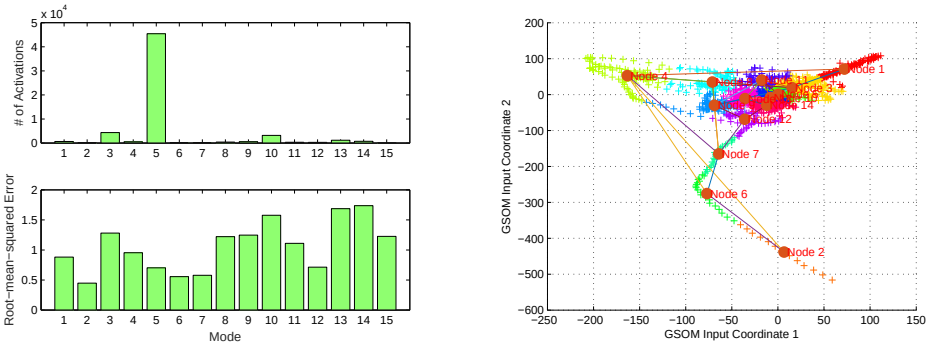


Figure 3.21: Fault signatures (red crosses) plotted on the principal component plane of the PCA_{normal}, green circles denote the normal operating region

A GSMMS framework using PCA as local models was then trained on the faulty data. The number of components for each local model was set to three. The maximum network size was set to 20, which is the true number of fault conditions.

The trained GSMMS framework network structure and the GSOM diagnostics are shown in Figure 3.22. The GSOM arrived at a network size of 15 nodes. Using this trained network representation, we can label each node as either a faulty region or a normal operating region based on the training

data labels available to us from the simulation.



(a) Simulated inputs and nonlinear output time series plot

(b) SOM network structure after training

Figure 3.22: GSMMS network trained using simulated Tennessee-Eastman process data

To apply the trained GSMMS in fault detection. The scores of incoming sample are calculated based on the PCA_{normal} loadings. Using the scores as inputs, the best matching unit and its neighbors can be identified on the existing SOM structure. The T^2 and Q statistics of each local PCA node within the neighborhood can then be calculated. Using these alarm statistics, the decision tree shown in Figure 3.23 is used to arrive at the final decision of whether the input observation is normal or faulty.

The fault detection speeds, the missed alarm rates, and the false alarm rates results are reported in Table 3.5,3.6 and 3.7 respectively. Two literature sources (Russel et al. [104] and Zhang [105]) are used as reference benchmarks to compare the performance of GSMMS-PCA. Russel et al. evaluated the performances of PCA, dynamic PCA and canonical variation analysis (CVA) in detecting these faults, whereas Zhang evaluated nonlinear methods of kernel PCA, kernel ICA and implicit kernel ICA. The false alarm rates are aggregated

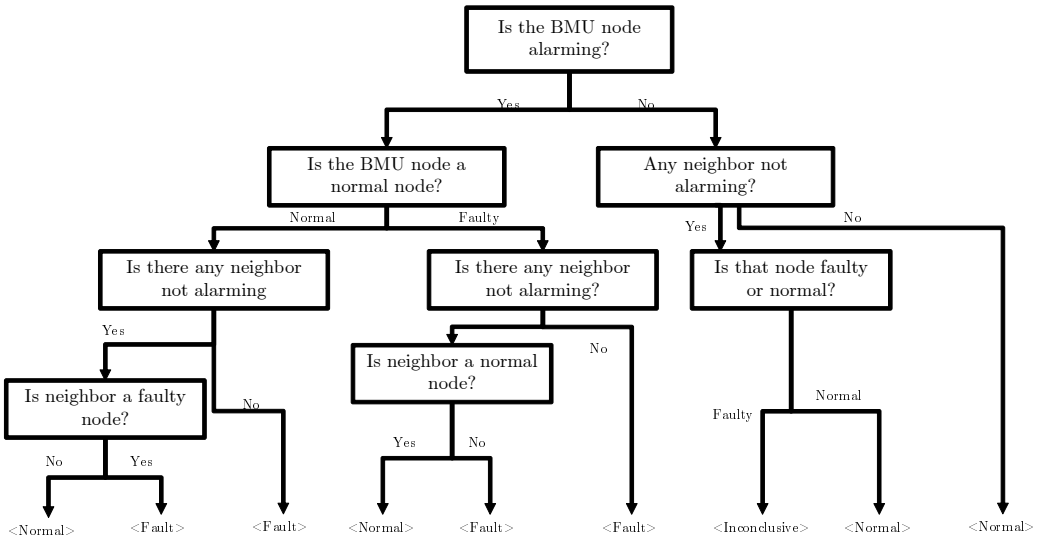


Figure 3.23: Decision tree of the GSMMS-PCA for detecting faults in the Tennessee-Eastman case study

for all 20 faults to maintain the same reporting format found in the literature references.

The fault detection speed of the GSMMS-PCA is on average the same or better than PCA based resources. However, the GSMMS method was unable to detect faults 4, 9, and 19. These faults are difficult to detect due to their similarity to the normal operating data in the principal component space. The fault signatures for these faults are completely overlapped with the normal operating data (as shown in Figure 3.21). The GSMMS-PCA approach increases the sensitivity of the PCA model fault detection by dividing the larger ellipse into many smaller ellipses; but in cases where the normal data is indistinguishable from faulty data, it becomes a challenge to detect these faults. A possible solution would be to try to separate the faulty data from the normal data before applying fault detection methods by introducing lagged

variables, additional measurements, or nonlinear transformations. However, these results are outside of the scope of current work and is thus not discussed here. In addition, as a result of the increased sensitivity in the GSMMS-PCA model, the missed and false alarm rates of the GSMMS-PCA model on average are better than the conventional performance reported in the literature sources as shown in Tables 3.6 and 3.7.

Table 3.5: Fault detection speed for the Tennessee Eastman Process (lower is faster, blank means failure to detect)

			Russel et al. [104]							Zhang [105]		
Fault	GSMMS	Average time	PCA T^2	PCA Q	DPCA T^2	DPCA Q	CVA T^2	CVAT_r^2	CVA Q	KPCA	KICA	Imp. KICA
1	24	13	21	9	18	15	6	9	6	15	9	6
2	240	41	51	36	48	39	39	45	75	30	36	33
3	15											
4		234		9	453	3	1386	3		9	6	6
5	16	8	48	3	6	6	3	3	0	3	3	3
6	27	9	30	3	633	3	3	3	0	3	3	0
7	7	5	3	3	3	3	3	3	0	3	3	0
8	76	64	69	60	69	63	60	60	63	75	69	60
9		899										
10	183	131	288	147	303	150	75	69	132	60	51	42
11	69	271	912	33	585	21	876	33	81	69	57	45
12	162	15	66	24	9	24	6	6	0	9	6	3
13	139	110	147	111	135	120	126	117	129	123	114	99
14	16	6	12	3	18	3	6	3	3	3	3	3
15	323	865		2220			2031			27	27	21
16	328	304	936	591	597	588	42	27	33	27	21	9
17	166	89	87	75	84	72	81	60	69	57	51	51
18	260	242	279	252	279	252	249	237	252	222	198	195
19		140				246		33				
20	228	189	261	261	267	252	246	198	216	177	165	135

Table 3.6: Missed alarm rates in percentages for the Tennessee Eastman Process (lower is better, blank means failure to detect)

		Russel et al. [104]							Zhang [105]		
Fault	GSMMS	PCA T^2	PCA Q	DPCA T^2	DPCA Q	CVA T^2	CVAT_r^2	CVA Q	KPCA	KICA	Imp. KICA
1	1.3	0.8	0.3	0.6	0.5	0.1	0	0.3	0	0	0
2	2.1	2.0	1.4	1.9	1.5	1.1	1.0	2.6	2.0	2.0	
3	73.5	99.8	99.1	99.1	99.0	98.1	98.6	98.5	96.0	94.0	92.0
4		95.6	3.8	93.9	0	68.8	0	97.5	91.0	18.0	19.0
5	3.2	77.5	74.6	75.8	74.8	0	0	0	75.0	71.0	71.0
6	0.9	1.1	0	1.3	0	0	0	0	1.0	0	0
7	13.4	8.5	0	15.9	0	38.6	0	48.6	0	0	0
8	2.8	3.4	2.4	2.8	2.5	2.1	1.6	48.6	3.0	3.0	2.0
9		99.4	98.1	99.5	99.4	98.6	99.3	99.3	96.0	95.0	95.0
10	12.4	66.6	65.9	58.0	66.5	16.6	9.9	59.9	57.0	19.0	20.0
11	43.2	79.4	35.6	80.1	19.3	51.5	19.5	66.9	76.0	19.0	18.0
12	1.5	2.9	2.5	1.0	2.4	0	0	2.1	3.0	3.0	2.0
13	2.7	6.0	4.5	4.9	4.9	4.7	4.0	5.5	6.0	5.0	5.0
14	3.2	15.8	0	6.1	0	0	0	12.2	21.0	0	0
15	83.5	98.8	97.3	96.4	97.6	92.8	90.3	97.9	95.0	95.0	94.0
16	92.5	83.4	75.5	78.3	70.8	16.6	8.4	42.9	70.0	20.0	20.0
17	70.3	25.9	10.8	24.0	5.3	10.4	2.4	16.8	26.0	5.0	5.0
18	24.5	11.3	10.1	11.1	10.0	9.4	9.2	10.2	10.0	10.0	9.0
19		99.6	87.3	99.3	73.5	84.9	1.9	92.3	97.0	25.0	23.0
20	23.8	70.1	57.0	64.4	55.8	44.0	34.2	54.7	59.0	42.0	50.0

Table 3.7: False alarm rates in percentages for the Tennessee Eastman Process (lower is better)

	Methods	False Detection Rate
	GSMMS-PCA	5.7
Russell et al. [104]	PCA T^2	1.4
	PCA Q	16.0
	DPCA T^2	0.6
	DPCA Q	28.1
	CVA T_s^2	8.3
	CVA T_r^2	12.6
	CVA Q	8.7
Zhang [105]	PCA T^2	0.5
	KPCA T^2	1.52
	KICA T^2	0.31
	Improved KICA T^2	0.27

3.4 Summary

In this chapter, we introduce two methods designed to address the issue of model quality degradation in inferential sensors and fault detection models. The two common causes of performance degradation are due to:

- Slow unmeasured process drift with indistinguishable changes in operating mode or product grade.
- Seasonality or cyclic behavior due to process nonlinearity, multiple operating modes or other external influences.

Data-driven models built on these types of processes will decay over time, since the process condition usually deviates from the original condition when the models are developed. To deal with these performance degradations, there are two approaches in literature.

For slow process drifts, a moving window based model update scheme is an effective solution to maintain the model quality overtime. A moving window PLS model is more appealing than recursive PLS models because moving window approaches allow us to re-compute the T^2 and Q diagnostics for the updated model. These diagnostics are also an important part of a projection based data-driven model. They can be used to screen incoming update data for potential outliers. However, in current implementations, it is difficult to distinguish whether an outlier actually constitutes an measurement outlier or a process excursion. Including an measurement outlier (fictitious alarm) can reduce the prediction performance of the data-driven model, while including an process excursion will improve the model prediction. To improve the robustness of the model update, we propose using total projection to latent

structure (TPLS) decomposition as a more refined screening tool to monitor for measurement outliers. T-PLS can be performed on an existing PLS model to divide the incoming data into subspaces that are correlated and orthogonal to the process output. This scheme gives finer control and allow inclusion of variations that could further improve the model without introducing excessive noise.

For processes exhibiting cyclic behavior or multiple operating conditions, "divide and conquer" approaches have been proposed by many researchers in this field. However, most of these methods are not designed to handle high dimensional input data. In addition, detailed process knowledge is often required to select the best features for local modeling. Considerable effort must be taken to determine the appropriate partition scheme to divide the training data. Lastly, to update these methods online, the supervisory layer (where operating mode labels are assigned or how local models are combined) also needs to be considered. To address these problems. We extend a novel framework that combines growing self organizing map with PLS and PCA local models. The proposed framework trains the structural parameters and the local model coefficients simultaneously and can be updated online. The self organizing map acts as the supervisory layer and is able to assign data into a "neighborhood" of models. Using the entire neighborhood for training and prediction smoothes predictions on the boundary of neighborhoods. The self organizing map is able to grow new nodes to accommodate previously unseen process conditions. Two case studies using simulated data have shown that the proposed technique is able to improve the prediction and the fault detection capabilities of PLS and PCA models respectively.

Chapter 4

Industrial Applications

4.1 Introduction

In this chapter, we apply the previously discussed modeling methods in developing two virtual metrology models for plasma etch processes using data provided by Texas Instruments. In addition, a novel use of virtual metrology in controller performance assessment is proposed and tested using a simulated dataset. Lastly, modeling results using industrial data from a liquid-liquid separation process in a chemical plant is also provided to demonstrate why using moving window VIP for variable selection could be beneficial than other existing techniques.

4.2 Etch Rate Prediction of a Polysilicon Gate Etch Process

Plasma etching succeeds traditional wet etching due to its advantages in etch uniformity, selectivity and non-isotropic process [106]. Figure 4.1 shows a schematic of a plasma etch reactor, etchant gases (usually fluorine, chlorine based, halide containing species) enter the reactor through the gas inlet and turns into plasma under the excitation of the RF power. The plasma gas contains free electronics, ionized molecules, neutral molecules, ionized fragments of broken-up molecules and free radicals. In addition to reactive chemical etching (which forms a isotropic etch profile), plasma etch also produces physical etching effects through sputtering of charged particles. Different etch tools have been designed with different emphasis on the reaction mechanism. Figure 4.2 illustrates the different geometry from these etching mechanisms. Isotropic etching could cause undercutting, while physical sputtering can cause trenching; both of which are undesirable geometries. To minimize these effects, the trim time for the etch step has to be controlled precisely, which requires an accurate process model.

Several physics based simulation tools can be used predict the shape and topography of specific etching technologies. The most prominent ones are SPEEDIE[107], Sentaurus Topography[108] and ATHENA[109]. However, these simulator relies empirically derived etch models that contain inputs impossible to measure online in commercial systems, such as the concentration of reactant species. In practice, the etch rate is usually estimated by running test wafers before the a lot of wafers are etched.

Polysilicon gate etch processes define gate transistor structure and produce other geometries that are important in semiconductor devices. The gate

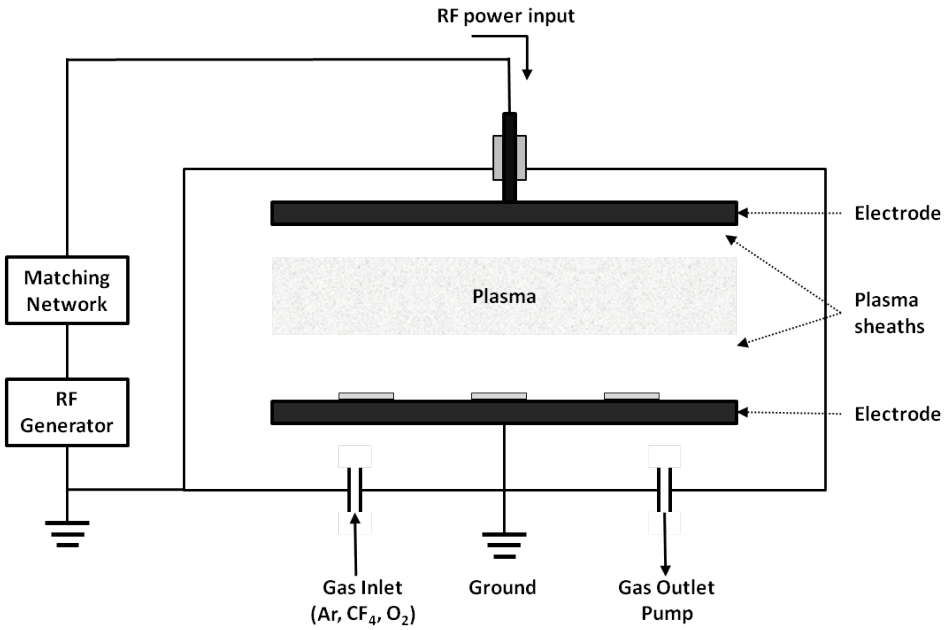


Figure 4.1: Schematic of a typical plasma etching system

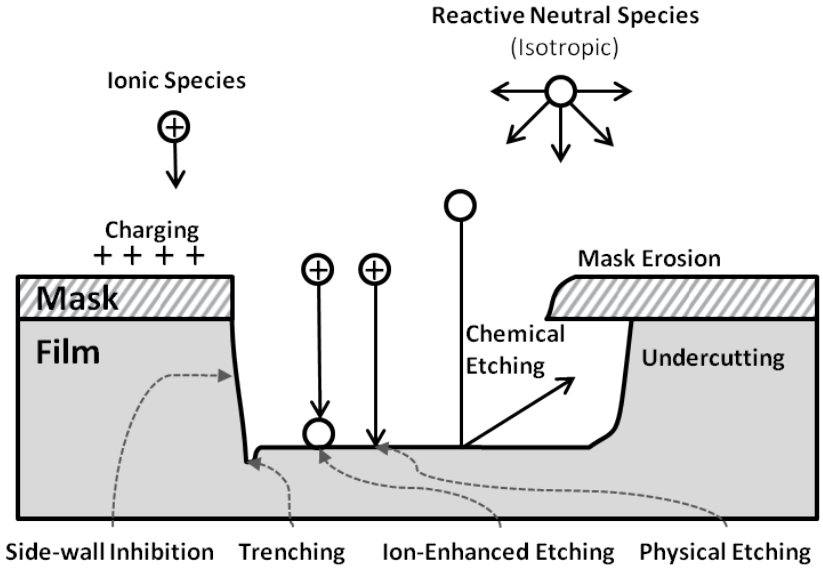


Figure 4.2: Summary of etching mechanisms and typical problems in plasma etching

trim etch process produces a polysilicon line with a width smaller than the minimum photoresist line. Line-width profiles and critical dimension are critical parameters that correlate to the performance of the final device. Metrology stations are used to measure these quality parameters before and after gate etch to ensure wafer quality. These metrology measurements are illustrated in Figure 4.3.

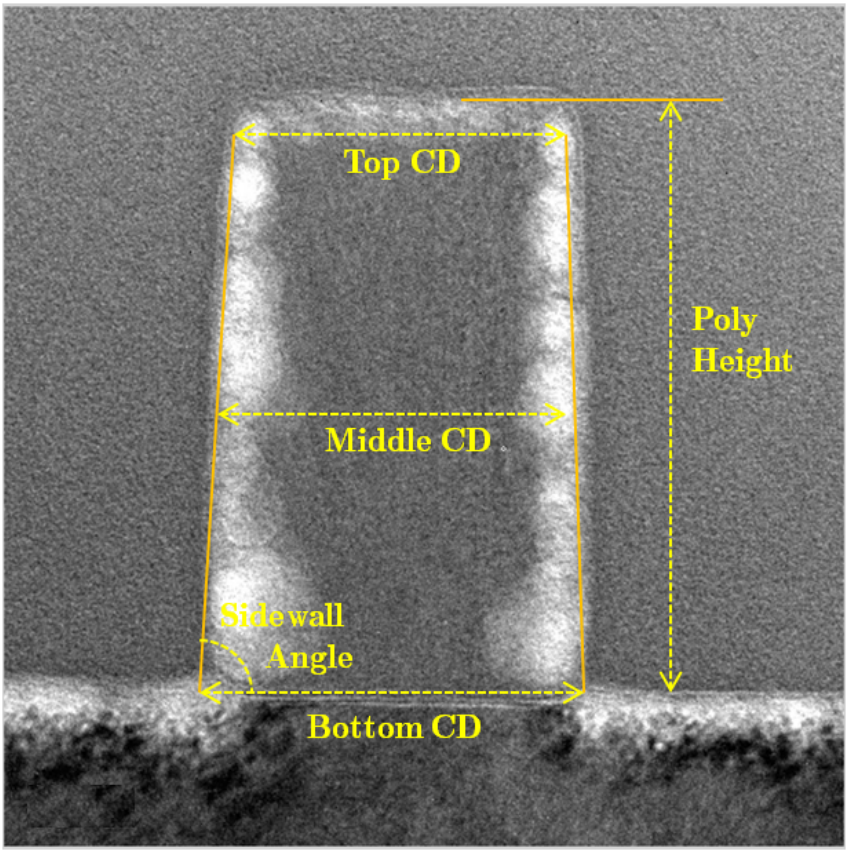


Figure 4.3: Transmission Electron Micrograph (TEM) of a polysilicon gate geometry showing the quality variables of interest

The development inspection critical dimension (DICD) measures the critical dimension of the wafers following the develop step of the photoresist

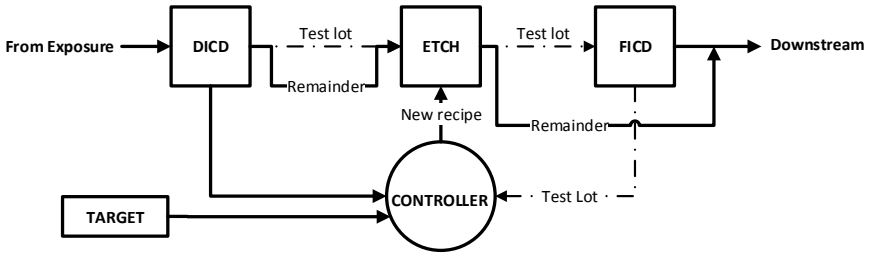


Figure 4.4: CD control flow diagram showing an example scheme utilizing test wafer FICD and DICD measurements

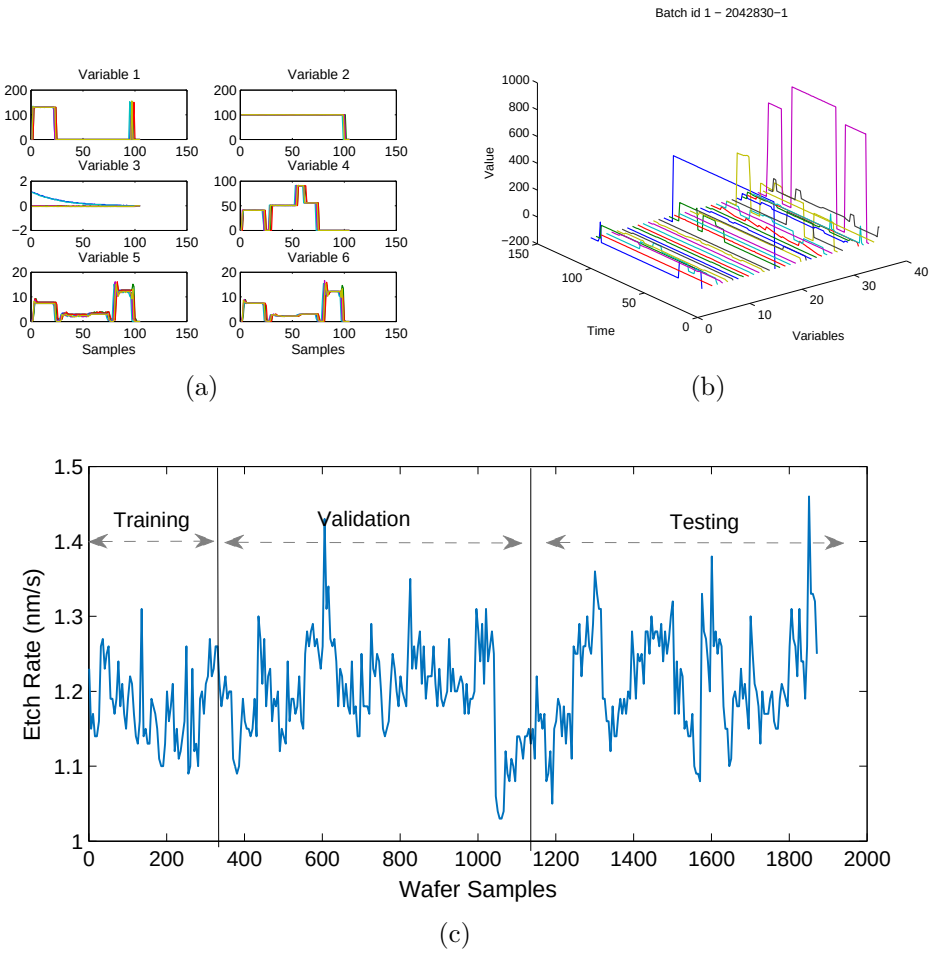
processing. The DICD measurement is usually made on three "pilot" test wafers from a lot of 25 wafers. Using the DICD measurements from the test wafers, the etch time for the gate etch process can be calculated based on a empirical process model. Control is then used subsequently to compensate for process drifts in daily operation. Using the estimated etch times, the test wafers are etched in the plasma etch chamber and then subsequently stripped and cleaned. The critical dimensions of the etched wafers are then measured. This measurement is called the Final Inspection Critical Dimension (FICD). The test wafer FICD and DICD measurements are then used to estimate the etch time for the rest of the wafers in the lot. In some simple control strategies, feedback control of the critical dimensions relies solely on the FICD measurement of the test wafer. In an alternative control scheme, feedforward control can be realized by combining DICD measurements of subsequent lots with FICD measurements of the test wafers. A example control block diagram utilizing DICD and FICD measurements are shown in Figure 4.4

The dataset used in modeling the gate etch process contains 1800 wafers with metrology measurements of Develop-Inspect Critical Dimension of the resist pattern (DICD) and Final Inspect Critical Dimension (FICD). Modeled

outputs could be expressed as either the difference between these measurements (etch bias) or the etch bias divided by the trim etch time as the etch rate.

In this study, *the etch rate is selected as the output variable.* Figure 4.5(c) shows the etch rate plot for the entire dataset, where the first 205 wafers are used as training data; the rest of the dataset are used for validation and testing. The metrology measurements are the response variables in the model. All of the wafer data are from the same manufacturing thread, but the tool did not produce all the wafers continuously.

The trace data (input variables) in the model are collected for each batch in the fault detection and classification (FDC) system of the tool. These measurements can be categorized into RF circuitry readings, etchant gas flow rates, temperature, pressure and key actuator movement readings. The data are collected at 0.5 Hz. Miscellaneous information such as recipe step, process time and EWMA-estimated disturbances are also appended to the original data set. In total, batch trajectories from 39 measurements were used as input variables in attempt to predict the average etch rate of each wafer. An example plot of the raw trace data is also shown in Figure 4.5.



(a) (b)

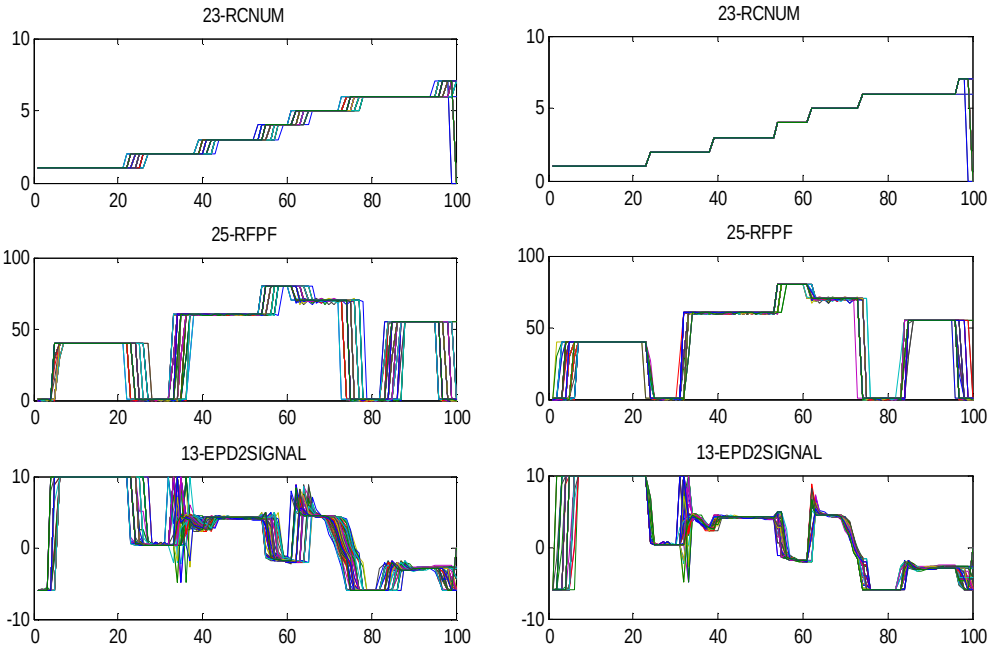
(c)

Figure 4.5: Raw data plot showing (a) the temporal profiles of six variables for five wafer batches, (b) A three-dimensional representation of trajectories from a single batch, and (c) output variable etch rate

4.2.1 Unfolding and alignment

Each dataset was first screened for missing or redundant data. Redundant and irrelevant variables were removed by manual inspection. Also excluded from the models were variables with more than 50% data being blank such as status indicators. Temporal outliers such as spikes in sensor readings were removed by checking against the confidence limits of corresponding variables. After initial cleaning, the dataset was then unfolded into a 2-D array using batch-wise unfolding. The unfolded data then underwent another outlier removal round to remove batch-wise outliers. Process engineers recommended using recipe number as a guide to verify data integrity, so the recipe number trajectory of every run was compared against a prescribed reference trajectory; large deviations in the reference trajectory raised alarms that label the batch as abnormal.

After initial trajectory screening, the trajectories are aligned using robust-derivative dynamic time warping [41] based on the indicator variable "recipe number". Univariate scaling and mean-centering were then applied to the unfolded input-output data to create the $\mathbf{X}$ and $\mathbf{Y}$ matrices. Three representative batch trajectories before and after the alignment are shown in Figure 4.6.

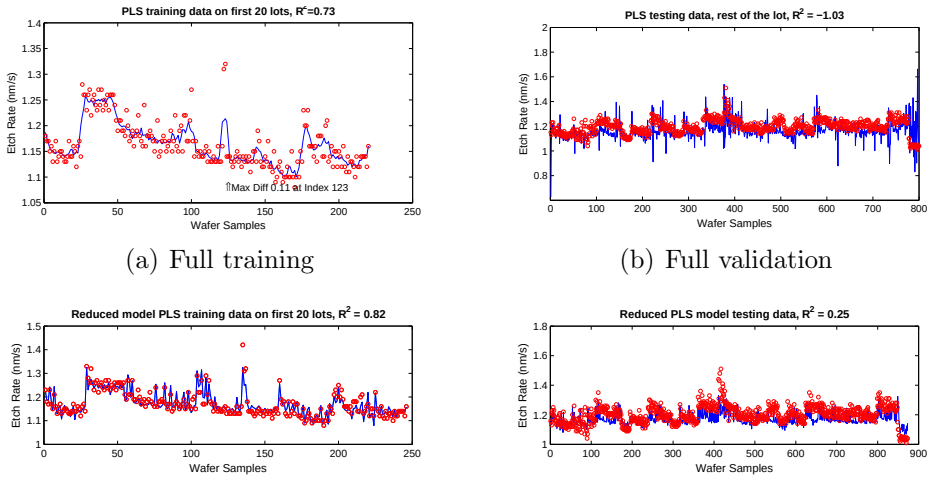


(a) Before Alignment (b) After Alignment

Figure 4.6: Three representative trajectories before and after alignment

4.2.2 Full and simplified PLS models

A preliminary PLS model is built using the unfolded dataset to provide a baseline for PLS model performance. A total of 10 latent components is used in the preliminary model. The number of components is determined by minimizing the cross-validated prediction error. Cross-validation is done by dividing the training data into seven segments of equal length. One of the seven segments is removed from training data and used as testing data against the rest of the six data segments. This process is repeated for every data segment until all the errors are collected. The results are plotted in Figure4.7. From Figure 4.7(a), the PLS variable is able to achieve a R^2 value of higher than 0.5, this indicates that there are significant correlation between the input and output that can be refined with additional variable selection and training data screening.



(a) Full training

(b) Full validation

(c) Simplified training

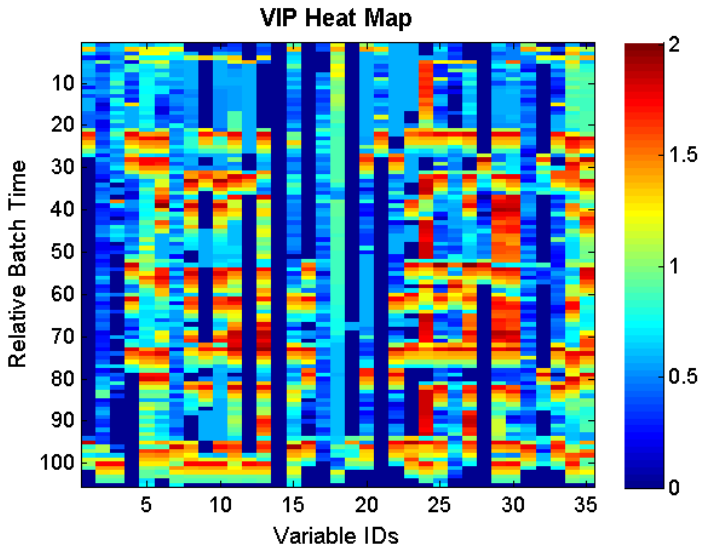
(d) Simplified validation

Figure 4.7: Raw and simplified PLS model performance on training and validation data

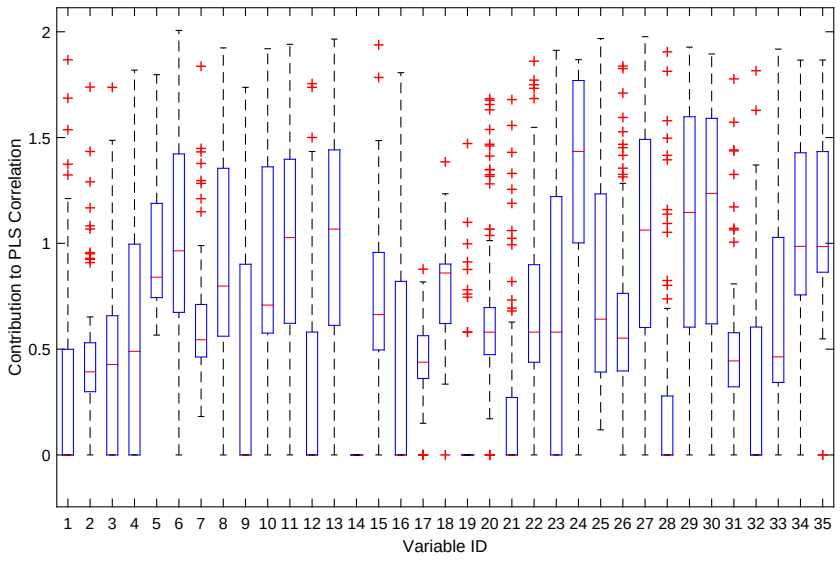
To perform variable selection, the VIP values of each multi-way input is calculated and then refolded back into the original dimensions. The resulting VIP can be visualized graphically in Figure 4.8. The detailed steps to create this plot has been discussed in Section 2.2.4.1. This plot can be examined from two dimensions. From the relative batch time dimension, data from sample periods 20-30, 50-60, 70-80 and 95-105 are relatively important periods that correlate with the output variation. From the variable dimension, it can be seen that variables (4,6,8,11,13,24,29,30,34, and 35) consistently exhibit higher correlation to the output than the rest. The number of multiway variables in the simplified model can then be reduced by applying filters to screen out unimportant time periods and variables. After simplification, the number of multiway input to the PLS model was reduced from 3605 to 652 variables.

The simplified model performed much better than the raw process model during training. However, during validation, the model shows significant degradation of performance as the prediction progresses farther away from the original training period. This indicates that drifts and disturbances are present in the dataset and are accounted for by the model.

Drifts and disturbances experienced in the quality variables of plasma etching can be caused by many factors. For example, having multiple production threads, meaning that each plasma tool is used to manufacture different products using different recipe settings, could contaminate the current production thread with residual deposits from another product. The inherent process nonlinearity of the batch-type dynamic trajectories in plasma etching also makes it difficult to fully model the plasma etching process on different time-scales. As a result, slower time-scale dynamics could then appear as a drifting bias in the etch rate. Preventive maintenance done routinely to clean



(a) Variable Importance in Projection visualized in two-dimensions

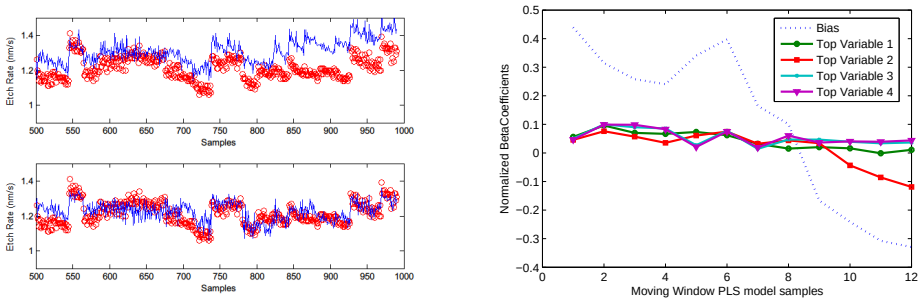


(b) Box plot of the VIP variation (in time) in each variable across the training batches

Figure 4.8: VIP information derived from the Raw PLS model used in model simplification

the plasma chamber of deposits and residues also introduces step change to etch rates that must be accounted for. Lastly, errors from upstream processes could also propagate and affect the etch profile in the plasma etching process. In practice, it is difficult to identify the source of drifts and disturbances, as a result, our goal is to account for the slow drifts in the process through moving window model update and to account for the spike disturbances by selectively removing them according to T-PLS subspace monitoring statistics.

4.2.3 Moving window VIP with T-PLS screening



(a) (b)

Figure 4.9: (a) Effect of moving window PLS update on predicting etch rate (bottom) compared against PLS without moving window updates (top), (b) Normalized regression coefficient of top four variables and the bias over the moving window range

Figure 4.9 shows the comparison between the MW-PLS and the simplified PLS model. In the MW-PLS model, a PLS model with 200 wafer samples is first initialized; subsequent models are updated at five lot intervals and the maximum model training size is fixed at 300 wafer samples. Comparing the moving window update model results with the fixed PLS model, the MW-PLS model handles drifts and disturbances much better and is able to track the

actual metrology measurements (especially after 800 wafer samples) In Figure 4.9(b), the normalized beta coefficient for the top four variables and the bias for the moving window PLS model samples has been plotted. The bias term exhibited the biggest change through performing moving window update. This observation confirms that the process has undergone a slow drift instead of a change in the correlation structure identified in the PLS.

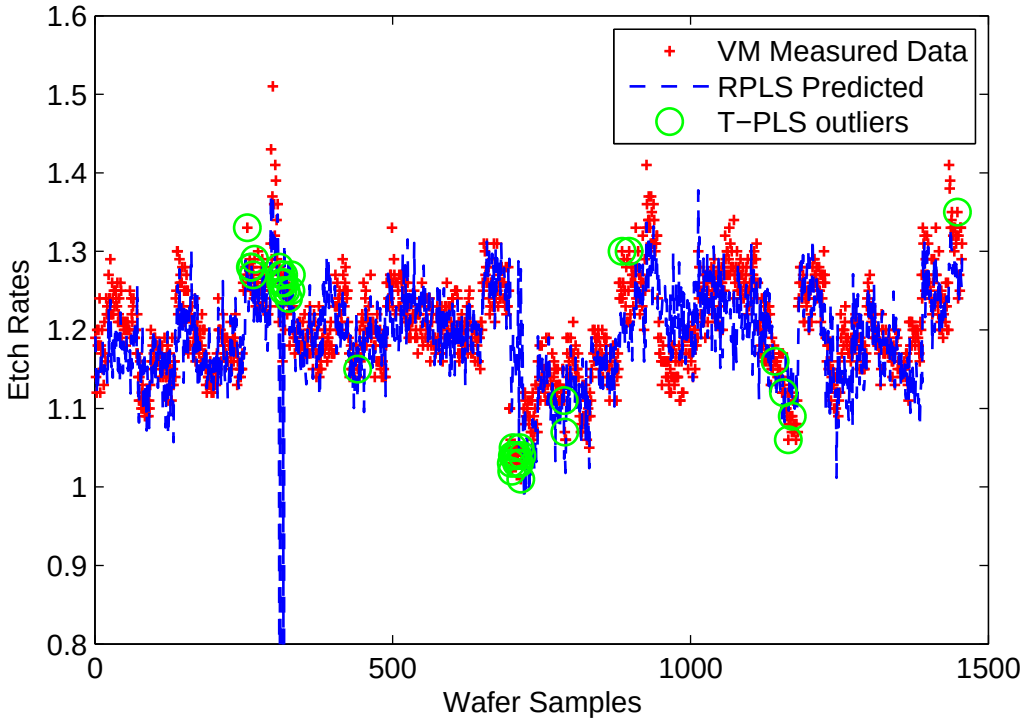


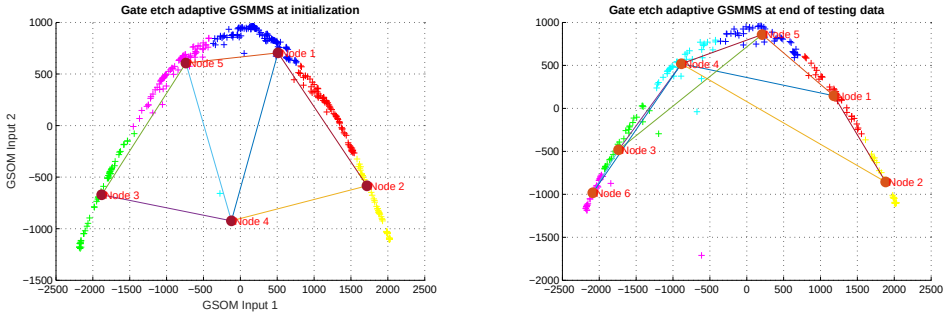
Figure 4.10: Moving Window PLS using TPLS Outlier Screening

As shown in Figure 4.10, in regions with strong outlier presence around wafer samples 380, TPLS detection indices identified the abnormal runs and excluded the highlighted wafers from entering the training data samples. Exclusion of abnormal runs is effective at improving prediction quality when the incoming data quality is poor.

4.2.4 Adaptive GSMMS with PLS local modeling

An adaptive GSMMS model using local PLS models was also trained for this data set following the same data partitioning scheme (200 wafers for initial training, then subsequently updated every five lots). Detailed method

description of the GSMMS framework has been introduced in Section 3.3. To ensure the same comparison reference point, the trajectory alignment and variable selection performed for simplifying the raw PLS models are carried over and applied here for the GSMMS framework model here. The resulting unfolded multi-way dataset contains approximately 650 inputs. The feature input into the GSOM model is reduced to 3 inputs through PCA projection. The initial training returned a network structure with 5 nodes; detailed network structure and the input observations are plotted on the first two principal component planes in Figure 4.11(a). The first two component planes show a parabolic shape trajectory evolution for the training dataset, indicating strong non-Gaussianity in the data. GSMMS-PLS deals with this non-Gaussianity by partitioning into smaller segments that appears more Gaussian. The adaptive GSMMS model update is applied using a sample buffer size of five wafer lots (the same as MW-PLS). The resulting network structure after iterating through the entire testing data is shown in Figure 4.11(b). The network size is increased to six nodes. We also observe an increased density of nodes around the left "tail" of the parabola, indicating that particular region is being activated more frequently than the rest of the network. The prediction performance of the adaptive GSMMS prediction is plotted in Figure 4.12, the adaptive GSMMS predictions behaved as expected and was able to maintain tracking of the etch rate across two maintenance events around sample 220 and sample 800. The prediction error and coefficient of determination (R^2) for these models are compared against other popular modeling techniques in Table 4.1.



(a) (b)

Figure 4.11: Gate Etch adaptive GSMMS (a) post-training and (b) final (after testing data update) network structure

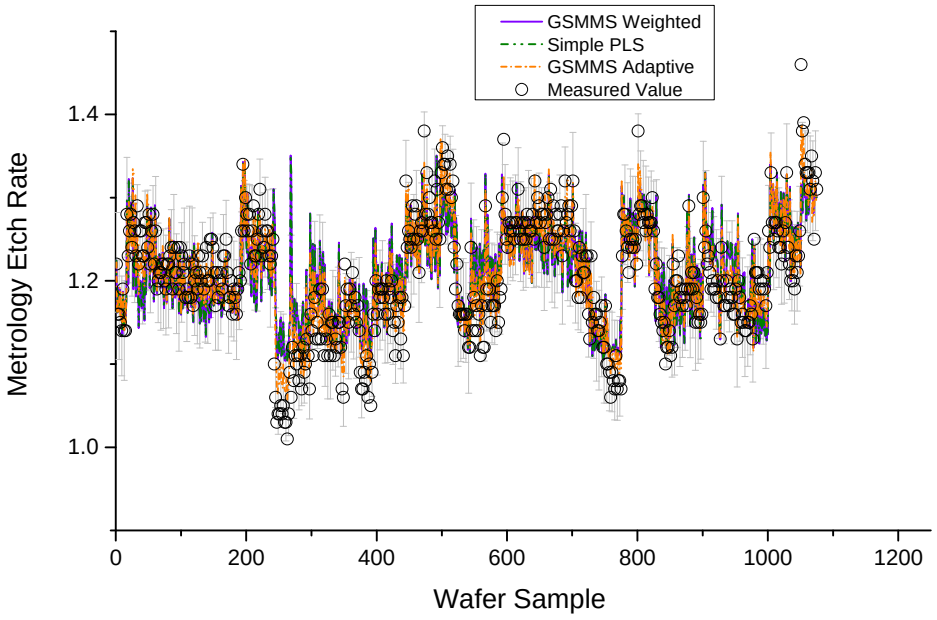


Figure 4.12: Prediction results of the Adaptive GSMMS model for the testing dataset

4.2.5 Comparison with other techniques

Principal component regression (PCA), back-propagated neural network (BPNN) , neural network PLS (NNPLS), and moving window partial least squares without TPLS detection (MW-PLS) were implemented on the same dataset to compare the results with our proposed framework. Table 4.1 lists the comparison results; the three metrics used to evaluate model performances are parameter size, cross-validated coefficient of determination (R^2), and cross validated root mean squared prediction error (RMSPECV).

Table 4.1: Model size, coefficient of determination, and cross-validated root mean squared residual of candidate models

Method	Approx. model size	Testing R^2	RMS residual
PCA regression	2700	0.425	0.0495
BPNN	$2700 \times N_{layers}$	0.72 / 0.32	0.0823
Nonlinear PLS with Neural networks	$600 \times A \times N_{layers}$	0.56 / 0.25	0.0623
Static PLS	600	0.142	0.0998
MW-PLS	600	0.442	0.0598
MW-PLS w/ TPLS	600	0.66	0.029
Adaptive GSMMS-PLS	$600 \times 5 \rightarrow 600 \times 6$	0.812	0.019

Table 4.2: Qualitative evaluation of the candidate models and their implementation difficulty

Method	Development Difficulty	Prediction R^2	Implementation Difficulty
PCA regression	Easy	0.2-0.5	Medium (large memory requirement)
Static PLS	Easy	0.1-0.5	Easy
BPNN	Medium	0-0.7	Difficult(nonlinear activation function)
Nonlinear PLS	Medium	0-0.7	Difficult (nonlinear + large memory requirement)
MW-PLS	Easy	0.2-0.6	Difficult (requires online update)
MW-PLS w/ TPLS	Medium	0.5-0.6	Difficult (requires online update and filtering logic)
Adaptive GSMMS-PLS	Difficult	0.6-0.8	Difficult (requires online update and computationally expensive)

The principal component analysis model used the full dataset (without variable filtering) and was found to be the most effective with 10 components. The back-propagated neural network model took a reduced set of unfolded variables as inputs (650 inputs). The reduced set was chosen using the correlation filtering variable selection method. A two-layer structure with ten neurons in the first layer was adopted to allow for efficient back-propagation. This network configuration resulted in 6210 weights in the BPNN model. Performance degradation was observed for the BPNN model. The testing R^2 in the BPNN model was around 0.72 for about 200 wafers initially, but it decreased quickly to around 0.32 afterwards. For NN-PLS, 10 outer components was used. The inner neural network size varied between 3 to 6, and was trained using the Levenberg-Marquadt back-propagation. The NN-PLS was able to provide reasonable approximation initially but also experienced relatively fast degradation in performance similar to BPNN; the drop in performance indicates that these two models suffer from over-fitting.The PLS with moving window update model also took the same set of reduced inputs as the BPNN model. Simply applying moving window update scheme to PLS models resulted in a much lower R^2 due to the effects of outlier and abnormal runs in future training data. Lastly, the proposed PLS-TPLS with moving window update results was shown. Overall, the principal component analysis regression model and the PLS-TPLS with moving window update model had the best performance across 1300 testing wafers. The BPNN was able to give excellent regression fit but degrades quickly overtime. Alternatively, PLS with moving window update had poor model fit due to impacts of outlier runs that affected the model results. Using the proposed MW-PLS with TPLS, the resulting model was able to maintain its prediction performance across maintenance boundary and abnormal wafers. These properties are valuable in an

online metrology environment where accuracy and certainty of measurement data cannot be guaranteed. The GSMMS adaptive framework gave the best prediction performance among the adaptive techniques (MW-PLS and MW-PLS w/ TPLS). However, the GSMMS model also had the greatest number of inputs among the three models due to the number of local models it generated to account for the non-Gaussianity in the system.

In any practical industrial system, it is also important to take into account the development costs and maintenance costs of an effective data-driven modeling solution. Table 4.2 lists the qualitative assessment of the evaluated models and methods in terms of their development difficulty, deployment difficulty, and modeling performance. These metrics were assessed subjectively based on our practical development experiences. Based on these results, static methods such as the PCA regression or PLS regression would be good candidate methods to attempt as a "first-check". If the preliminary modeling show some correlation, then more advanced techniques can be applied to further improve the performance depending on the level of prediction accuracy required and the deployment environment (online or offline, etc).

4.3 Control Performance Monitoring Based on VM Estimated Process Gains

Performance degradation in controllers can be caused by lack of controller maintenance, drifts in process conditions, or actuator and sensor degradation. Therefore, control performance monitoring and assessment (CPM/CPA) is imperative to maintain controllability and reliability of the manufacturing system. The main objective of CPA is to determine whether current control performance meets specified performance targets or response characteristics. The field of CPA has attracted growing interest since the work of Harris[110] in 1989. There are many published work in the past 23 years, and a few selected work with applications in semiconductor processes are discussed here. A more comprehensive list of those methods and applications in this field can be found in the excellent reviews done by Qin[111], Harris et al.[112] and Jelali [113].

There have been several proposed application of virtual metrology models for advanced process control (APC) and fault diagnostics and classification (FDC) purposes. For example, VM can be used in a dynamic sampling scheme to reduce metrology station utilization [81]. VM-assisted run-to-run control can achieve wafer-to-wafer level control precision as demonstrated through simulation [9]. The VM models can also be used in fault detection and diagnostics [15]. However, deploying these algorithms online faces deployment challenges such as additional IT infrastructure requirements, algorithm stability, model performance degradation, and high memory and computational costs. Alternatively, controller performance assessment and diagnosis (CPA/D) is an area where offline approaches are more commonly used and practical. As a result, it would be worthwhile to examine the possibility of using VM to assist in CPA and CPD.

Previous efforts in run-to-run control performance assessment (CPA) has focused on deriving expressions that compares the current control performance against a benchmark performance such as minimum variance benchmark or the best-achievable performance benchmark (Bode et al.[33], Chen[114], Good and Qin[115], Ko and Edgar[116], Prabhu and Edgar[117], Ma[118]). However, these approaches do not work well under the presence of metrology delay.

Wang et al. [119] and Jiang et al.[120] analyzed the output error of run-to-run EWMA controllers and applied closed-loop identification to identify important process parameters that relates to the control performance. An ARMAX regression method is applied to the process output error, and process parameters are estimated. However, When suboptimal behavior is detected, it is difficult to determine why and how to improve the control performance.

In the gate etch process, a run-to-run controller is used to control the final inspection critical dimension (FICD). The run-to-run controller takes the measured DICD from the previous lot and the current estimate of process gain to calculate the appropriate etch time settings for the next wafer. In this model, the process gain is usually estimated based on the first three test wafers in the lot.

4.3.1 Single-input-single-output Run-to-Run control

Here we will briefly introduce the standard EWMA run-to-run controller used in semiconductor manufacturing. In a standard EWMA controller scheme, the physical process is assumed to be of the form

$$y_k = \alpha_k + \beta u_k \tag{4.1}$$

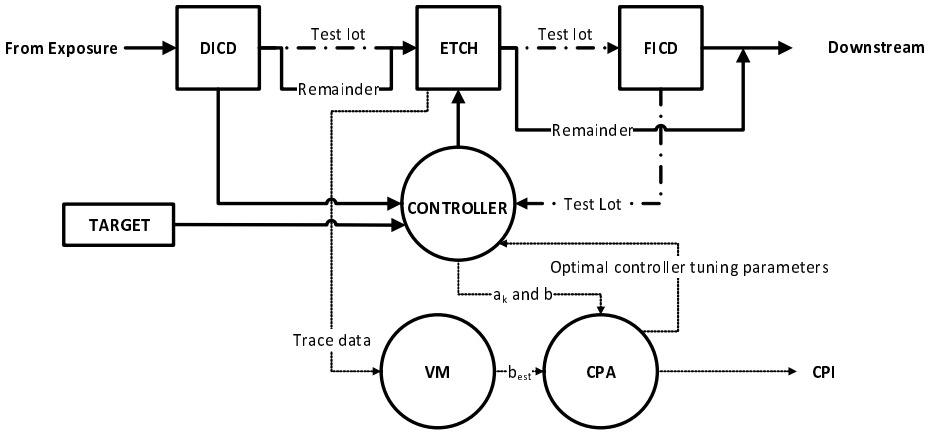


Figure 4.13: VM-assisted Controller Performance Assessment Scheme

where β is the process gain of a unit operation and α_k is a lumped disturbance term to account for unknown disturbance and drifts in the system. The disturbance is often modeled as a IMA(1,1) series of the form:

$$\alpha_k = \alpha_{k-1} - \theta\epsilon_{k-1} + \epsilon_k \tag{4.2}$$

where ϵ is an white noise disturbance with variance σ_0^2 and θ is the first order moving average coefficient. Box and Jenkins shows that the optimal one-step ahead estimator of this disturbance series takes the form:

$$\hat{\alpha}_{k+1} = \theta\hat{\alpha}_k + (1-\theta)\,\alpha_k \tag{4.3}$$

The Greek letter parameters are assumed to be the real, hidden parameters of the system, and their alphabetical counterparts represent the approximations that we use. As a result, the system model approximates the real process as:

$$\hat{y}_k = a_k + bu_k \tag{4.4}$$

where b is the estimate of the process gain, and a_k is an approximation of the real disturbance variable α_k. The approximate disturbance is updated according to the optimal predictor Equation 4.3 as follows:

$$a_{k+1} = (1 - \omega)\, a_k + \omega\, (y_k - b u_k) \tag{4.5}$$

where ω is the EWMA weighting coefficient.

Therefore, after completing run k and measuring output y_k, we can adjust the input to track a target setpoint r_{k+1}:

$$u_{k+1} = \frac{r_{k+1} - a_{k+1}}{b} \tag{4.6}$$

where r is the process target for the next run $k + 1$. r is usually fixed for the same production thread.

We can simulate a standard single-input-single-output (SISO) batch process under EWMA run-to-run control. The simulation parameters are listed in Table 4.3. The trajectories for the output, the true and estimated disturbance, calculated inputs, and the output error are shown in Figure 4.14.

Table 4.3: Simulation parameters used in generating Figure 4.14

Simulation Parameter	**Description**	**Settings**
N	number of runs	500
d	sampling time delay	1
θ	the IMA(1,1) moving average coefficient	0.85
β	the true process gain	2.5
σ	standard deviation of the white-noise in IMA(1,1) series	0.1
b	est process gain in process model	2.5
ω	the forgetting factor for EWMA filter	0.15
r	target setpoint in deviation variable	0

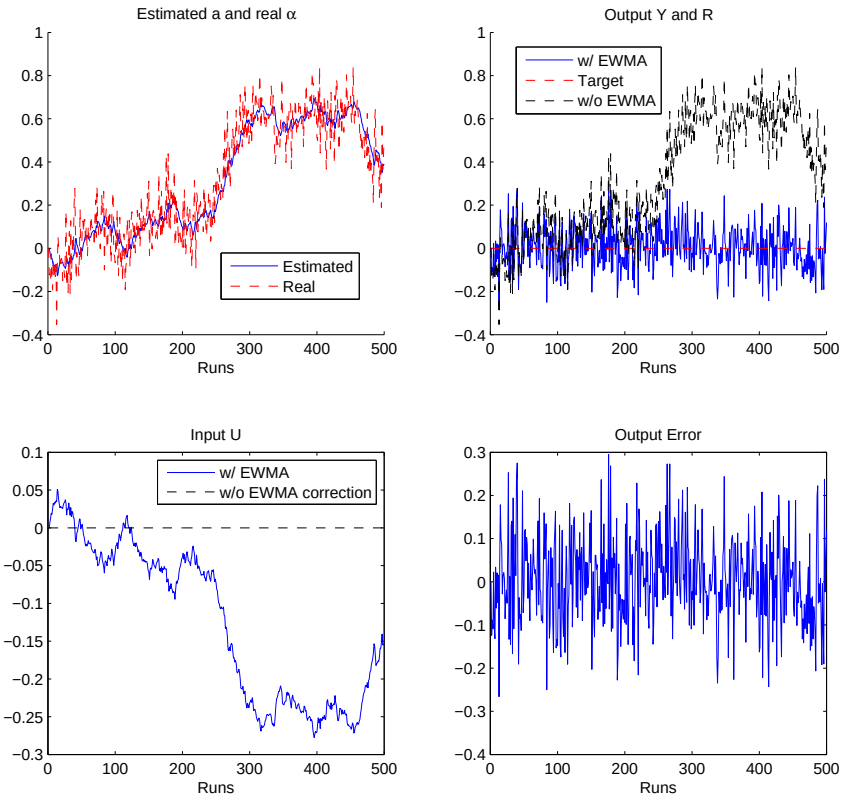


Figure 4.14: Simulated profile of a SISO Run-to-Run Process with and w/o EWMA controller with fixed set-point

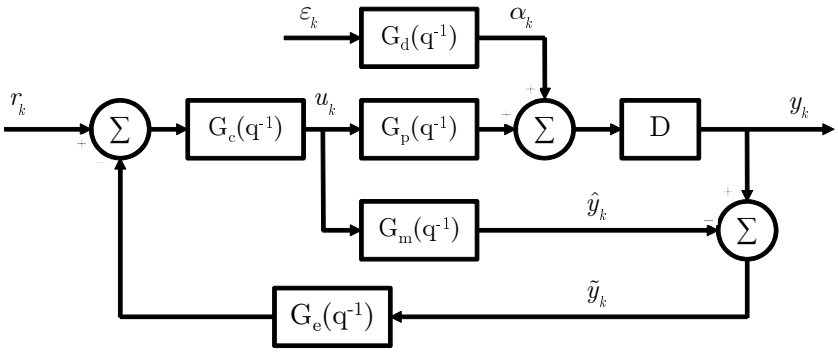


Figure 4.15: Internal Model Control Representation of a Discrete Run-to-Run Controller

4.3.2 Source of error analysis

The internal model representation of the SISO run-to-run control structure is shown in Figure 4.15. The transfer functions are defined as:

$$
\begin{aligned}
G_p &= \beta \\
\hat{G}_p &= b \\
\alpha_k &= \frac{1-\theta q^{-1}}{1-q^{-1}}\epsilon_k \\
G_c &= \frac{1}{b} \\
G_e &= \frac{\omega q^{-1}}{1-(1-\omega)\,q^{-1}}
\end{aligned}
$$

From Figure 4.15, the closed loop response of the system can be written using the individual block transfer functions from system inputs r_k and ϵ_k.

$$y_k = \frac{(\beta/b)\,(1+G_e)}{1+(\beta/b)\,G_e q^{-d}} q^{-d} r_k + \frac{G_d}{1+(\beta/b)\,G_e q^{-d}} q^{-d}\epsilon_k \tag{4.7}$$

We can further re-arrange this equation by substituting the definition of each transfer function block and the output error ($e_k = y_k - r_k$).

$$e_k = \left\{ \frac{\frac{\beta}{b}\left(q^{-d} - q^{-1-d}\right) - \left(1 - q^{-1}\right)}{1 - q^{-1} + \omega\frac{\beta}{b}q^{-1-d}} \right\} r_k + \left\{ \frac{1 - \theta q^{-1}}{1 - q^{-1} + \omega\frac{\beta}{b}q^{-1-d}} \right\} q^{-d}\epsilon_k \quad (4.8)$$

The detailed derivation of this system response is provided in Appendix C.

Equation 4.8 shows that the closed loop system response of the output error is driven by two inputs, the target setpoint r_k and the white-noise disturbance input ϵ_k. Under normal operating conditions, $r_k = r_{k-1}$, so the dynamics due to setpoint changes has settled down to steady-state and can be eliminated.

$$e_k = \left\{ \frac{1 - \theta q^{-1}}{1 - q^{-1} + \omega\frac{\beta}{b}q^{-1-d}} \right\} q^{-d}\epsilon_k \quad (4.9)$$

The ϵ_k is a white noise series and cannot be directly measured. Jiang et al. [120] has shown that an observer can be created to simulate $\hat{\epsilon}_k$ under ideal R2R control. From which, the IMA coefficient θ of the disturbance series can be estimated. The other term $\omega\frac{\beta}{b}$ represents the contribution of improper tuning of the forgetting factor with the process gain mismatch.

From this input-output system, one can observe that small plant mismatch ($\frac{\beta}{b} \neq 1$) can be compensated by changing the EWMA tuning factor ω, as long as the $\omega\frac{\beta}{b}$ remains relatively unchanged. In addition, at optimal controller tuning, the closed-loop residual of this system approaches white noise. Furthermore, if there is a large metrology delay d, the output error will deviate away from white noise behavior due to auto-regressive dynamics that cannot be compensated by the term $\omega\frac{\beta}{b}$.

Following this analysis, given that we have control performance degradation in the process, the possible contributing causes are of the following:

- Process disturbance dynamics has drifted away from the original tuning of the EWMA filter forgetting factor, ω
- The process gain β of the system has drifted away from the process model gain b
- The disturbance series no longer satisfies the assumption of IMA(1,1) time series.
- The metrology measurement delay is too long such that the d-step-ahead predictions are approaching precision limits.

Using strictly system identification based methods (such as ARMAX regression), it is difficult to diagnose the exact sources of the performance error; in addition, since the identification is done in a moving-window fashion, there will be delay in the response speed of such a system.

4.3.3 Virtual metrology assisted CPA

We have shown in the previous section that VM models for gate etch process is able to achieve approximately 0.80 R^2 value with a standard error of $\pm 0.02 nm/s$. Since the etch rate is closely coupled to the process gain of the gate etch model, we assume that after each wafer run k, the process gain for that run $\hat{\beta}_{vm,k}$ can be estimated to within $\pm 0.02 nm/s$ accuracy. In practice, this could be constructed by using a regression model to predict the process gain based on the metrology predicted etch rate.

The EWMA recursive update equation for the estimate of the disturbance series bias can be expanded:

$$\begin{aligned} a_{k+1} &= (1-\omega)\, a_k + (y_k - bu_k) \\ a_{k+1} &= (1-\omega)\, a_k + \omega\,(\beta u_k + \alpha_k - bu_k) \\ a_{k+1} &= (1-\omega)\, a_k + \omega\,[u_k\,(\beta - b) + \alpha_k] \\ a_{k+1} &= (1-\omega)\, a_k + \omega\,[u_k\,(\beta - b) + \alpha_{k-1} - \theta\epsilon_{k-1} + \epsilon_k] \end{aligned} \tag{4.10}$$

Based on this expansion, the estimated disturbance series update is based on the term $[u_k\,(\beta - b) + \alpha_{k-1} - \theta\epsilon_{k-1} + \epsilon_k]$. If there is model plant mismatch, then the term $u_k(\beta - b)$ will be nonzero. We can define the disturbance bias due to plant model mismatch as:

$$\xi_{mismatch,k} = u_k(\hat{\beta}_{vm} - b) \tag{4.11}$$

Using the VM estimated process gain, we can approximate the true process disturbance more accurately using an EWMA filtered estimator a^{vm}:

$$a^{vm}_{k+1} = (1-\omega)\, a^{vm}_k + (y_k - \beta_{vm} u_k) \tag{4.12}$$

Calculating $\xi_{mismatch,k}$ is equivalent to calculating the difference of the time series a_k and a^{vm}_k as shown in Figure 4.16.

When there is no plant-model mismatch, the controller gain (b) is close to the true process gain (β), so the calculated mismatch bias series $\xi_{mismatch,k}$ will be close to zero and behaves like a white noise. When there is process gain mismatch, the disturbance becomes biased due the non-zero term as shown in Equation 4.10, and the series becomes auto-correlated due to controller output (u_k) reacting to minimize the target offset. Therefore, we can exploit

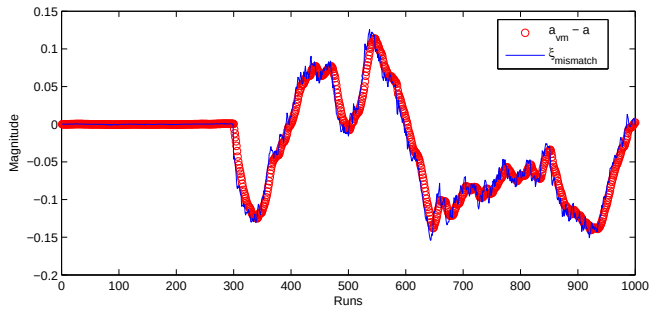


Figure 4.16: Comparison of plant model mismatch index $\xi_{mismatch,k}$ with the estimated disturbance difference for mismatch occurring at $k = 300$

this property to create a controller performance assessment index that checks for the presence of autocorrelation in plant-model mismatch. Inspired by the work from Jiang with recursive least square estimates of ARMAX parameters [120], a quick way to check for auto-correlation is by performing recursive least squares to estimate $\xi_{mismatch,k}$ as a AR(1) time series. If there is auto-correlation, the first order auto-regressive coefficient will be non-zero. The algorithm to perform recursive least square (RLS) estimate of AR(1) coefficient is shown in Table 4.4.

Table 4.4: Using recursive least squares to estimate ϕ in $(1 + \phi q^{-1})x_k = \epsilon_k$

Step	
1	Initialize $\phi_1 = 0, P = 1$
2	For k = 2 to N
a	Calculate the estimation error: $\alpha_k = x_k - x_{k-1}^T \phi_{k-1}$
b	Update recursive gain: $g_k = P_{k-1}x_{k-1}(\lambda + x_{k-1}^T P_{k-1} x_{k-1})^{-1}$
c	Update the parameter covariance matrix: $P_k = (\lambda)^{-1} P k_1 - g_k x_{k-1}^T \lambda^{-1} g_k$
d	Update the new coefficient estimate: $\phi_k = \phi_{k-1} + \alpha_k g_k$

After we obtain a series of ϕ_k corresponding to the estimate of the autoregressive coefficient at time k. The controller performance index is defined

as follows:

$$M = 1 - \phi \tag{4.13}$$

We demonstrate the effectiveness of this approach in simulation case studies. In total, four cases are considered. The first two cases are gain mismatch and improper tuning with metrology delay of one sample. The next two cases are for gain mismatch and improper tuning with randomly sampled metrology delays between 1 and 4 samples. The detailed simulation parameters are listed in Table 4.5. Figure 4.17 lists the resulting simulated controller performance index performance (top subplot) versus the filtered residual CPA II index proposed by Jiang et al. (bottom subplot) [120]. In all four cases simulated, the VM CPA indices are able to detect the controller performance degradation clearly. For the plant model mismatch cases (case 1 and 3), the gain difference between the VM estimated and the controller model results in an instant response in the VM CPA index. The alternative method required additional samples and took longer to respond. In addition, VM CPA index showed a less noisy profile than the CPA II index, making it easier to identify the monitoring cut-off point. In cases with improper control tuning (subplots b and d), the responses of the proposed VM CPA index is a gradual decay instead of a sharp drop. This slower response time is due to the propagation time required for the auto-correlation in the output residual to affect the input series u_k. In these cases, the detection speed of the VM CPA Index is slightly worse than the CPA II index, but the VM CPA index is much more stable and does not show oscillation which may obfuscate the results.

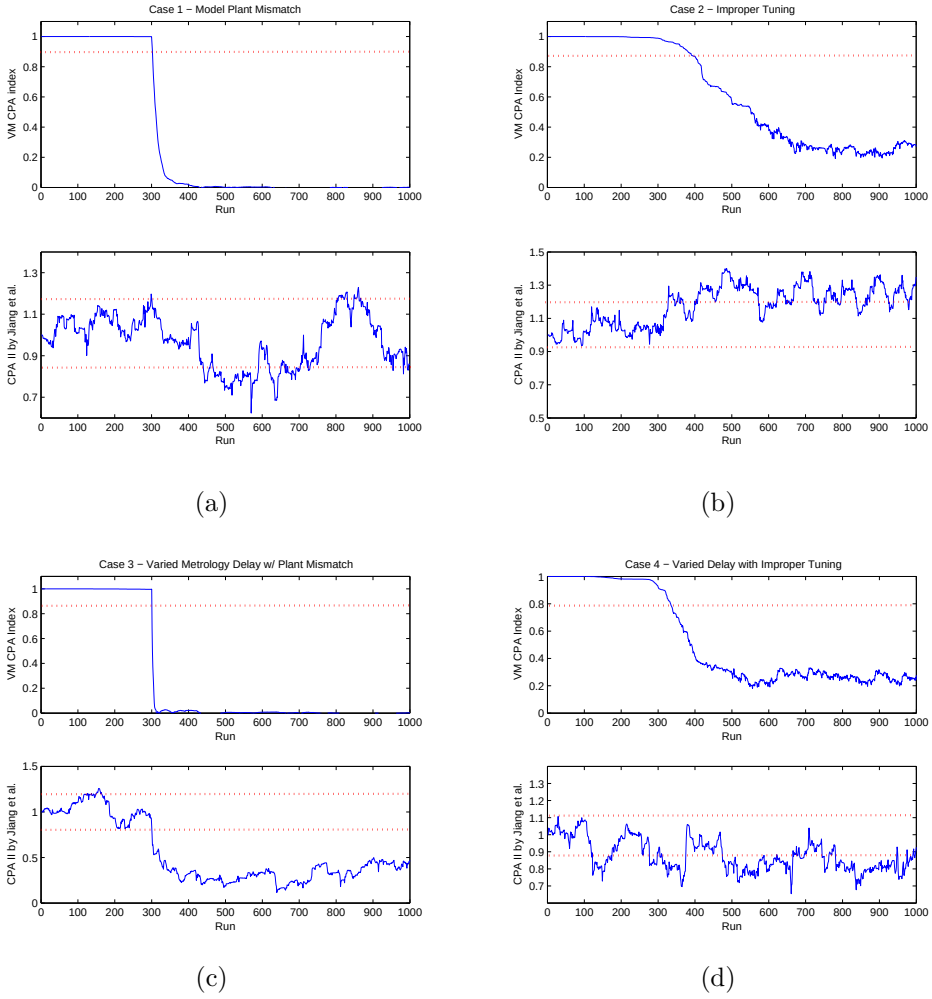


Figure 4.17: Controller performance assessment of SISO EWMA controller under scenarios listed in Table 4.5

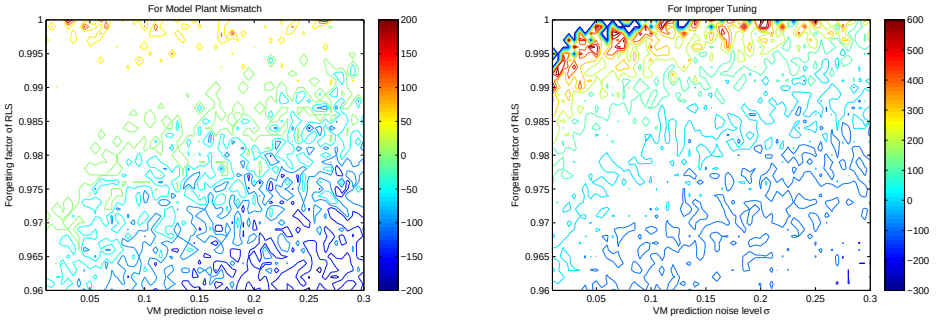
Table 4.5: Simulation parameters for Case 1 - Plant/model gain mismatch, Case 2 - Improper tuning, Case 3 - Varied metrology delay with gain mismatch, Case 4 - Varied metrology delay with improper tuning

	Case 1		Case 2		Case 3		Case 4	
Simulation Parameter	**Initial**	**Faulty**	**Initial**	**Faulty**	**Initial**	**Faulty**	**Initial**	**Faulty**
d	1	1	1	1	1	randi(2,4)	1	randi(2,4)
θ	0.85	0.85	0.85	0.65	0.85	0.85	0.65	
β	2.5	1.5	2.5	2.5	2.5	1.5	2.5	2.5
σ	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
b	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
ω	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
r	0	0	0	0	0	0	0	0

4.3.4 Impact of VM prediction quality on the VM-CPA performance

Furthermore, we investigate the effect of prediction accuracy in the VM estimated process gain on the performance of the proposed VM CPA index. To perform this case study, random noise with pre-defined noise level σ is injected into a baseline VM prediction. The VM CPA index is then calculated based on the noise level and a range of forgetting factor choices (since the forgetting factor in the RLS estimate acts as a filter that smoothes out the responses), the resulting detection time is plotted as a contour plot in Figure 4.18. The colors of the contours represent the detection speed of the VM CPA index. The two cases considered are again: (1) with model/plant gain mismatch and (2) improper tuning. The cold colored contours (cyan - blueish color) represent regions with false alarms reported by VM CPA index (where actual process is still in control). Warmer orange and red colored contours represent delays in the detection response due to excessive filtering, the magnitude (as seen in the colorbar) represents the number of samples elapsed before the out of control event is detected. Ideally, it would be best keep the detection speed as low as possible (without going into negative). Therefore, the cyan/green

colored contours represent regions with the optimal trade-off between lag in detection speed and being overly sensitive. As the noise level in the VM estimated process gain increases (corresponding to a poorer VM model), there appears to be more false alarms. This agrees with our intuition, since the VM CPA index will assign the contribution of noises in the gain estimation toward the auto-correlation of the mnodel-plant mismatch. However, the higher false alarm rate can be compensated by increasing the RLS forgetting factor to remove excessive noise. As a result, depending on the prediction quality of the VM model, the forgetting factor can act as a tuning parameter in the VM CPA index to control the trade-off between detection sensitivity and false alarm rate.



(a) (b)

Figure 4.18: The influence of process gain VM prediction error on the detection speed and false alarm of the CPA VM index

4.4 Metal Etch Relevant Variable Identification

Sheet resistance metrology and the FDC trace data from a metal etch tool were provided by Texas Instruments. The goal of this study is to identify relevant variables that explains the variation seen in the sheet resistance of the processed wafers.

Sheet resistance is a metrology measurement conducted at the end of the metal etch. The sheet resistance is measured at device contact points at multiple site locations on a wafer surface as shown in Figure 4.19. The readings from multiple sites are then averaged to create an average sheet resistance for each wafer. The sheet resistance is commonly measured to characterize the uniformity of a film or coating on the silicon wafer surface, and is often used in inline process control for quality assurance.

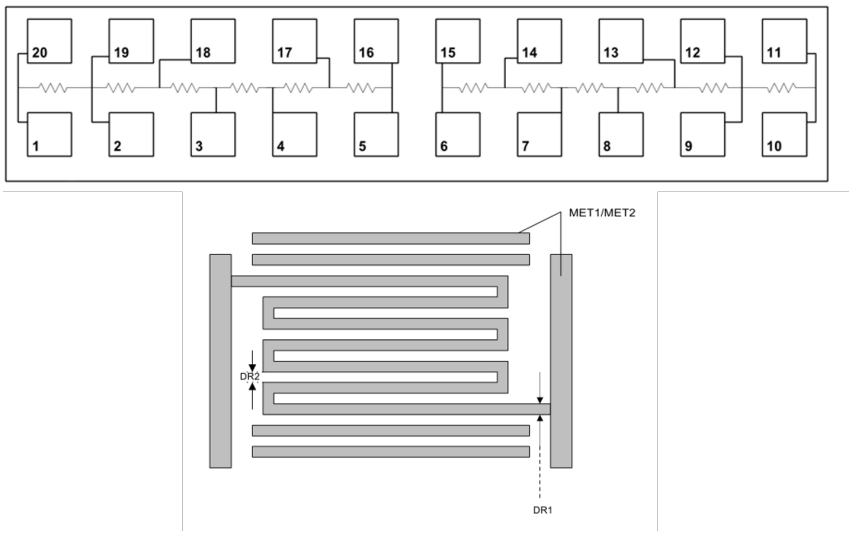


Figure 4.19: Schematic of the sheet resistance measurement contacts on the device surface from a die on the wafer

Different from gate etch where the trim time is adjusted based on run-

to-run control, the trim time in metal etch is based on end-point detection using Optical Emission Spectroscopy signals. As a result, the trim time cannot be used as an independent manipulated input to control the sheet resistance of the device at the end of the etch. In this study, we want to identify candidate variables (trajectory features) of a certain trajectory that could become manipulated variables used to reduce variation in the final sheet resistance of the metal etch process.

The entire dataset comprises of 632 wafers running the same metal etch recipe. There are a total of 13 manufacturing threads (making 13 different devices) and the wafers are produced continuously from January till May of 2013. The data was first exported as DAT files through Infincon Fabguard and then parsed into MATLAB readable CSV files.

Figure 4.20 shows the overall modeling work flow to analyze the metal etch data. An iterative approach similar to that of modeling gate etch is used. In the first step, manual screening is performed to remove poor quality data and non-relevant information:

- Trajectories that do not have any variation across *the batch dimension* are removed
- Trajectories that do not have any variation across *the time dimension* are summarized using the mean value
- Setpoints, recipe settings that are the same across all batches are removed.
- Runs with no matching contexts information are removed (i.e. have no associated metrology measurement)

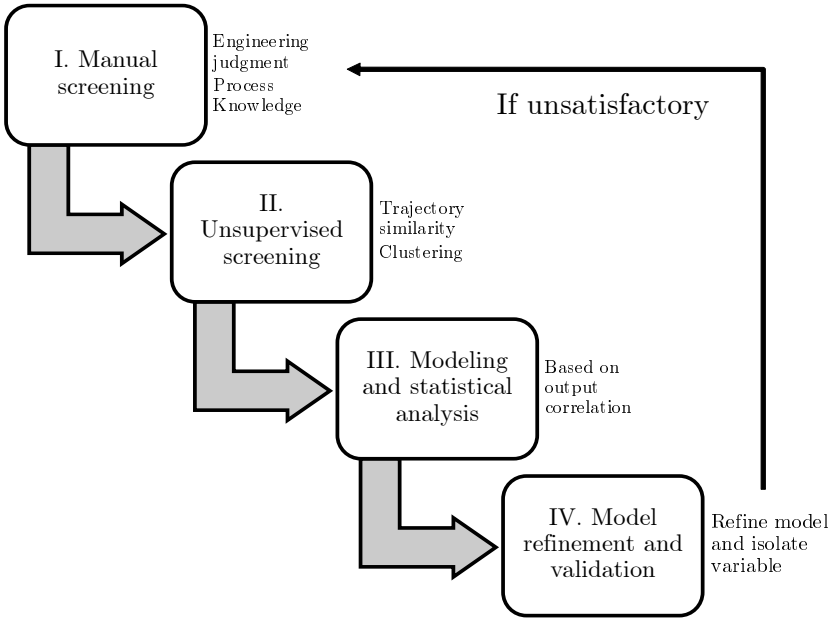


Figure 4.20: Overall metal etch data analysis workflow schematic

After manual screening, the data is then aligned using the recipe step number as the key indicator variable. The before and after alignment recipe number trajectory is shown in Figure 4.21. During dynamic time warping, several batches that showed high cumulative warping costs (the amount of shrinking / expansion performed on a trajectory) are identified and removed, these trajectories were later identified to be following a different etch recipe and is thus outside the scope of this modeling study.

After trajectory alignment, the unfolded data was then used in a full variable full duration multiway-PCA model. Figure 4.22 shows the score plot of the resulting PCA model. The scores are color-coded by the lot ID. Four lots of wafers are away from the rest of the lots, which were confirmed to be from a different chamber. These lots were excluded in subsequent PLS modeling.

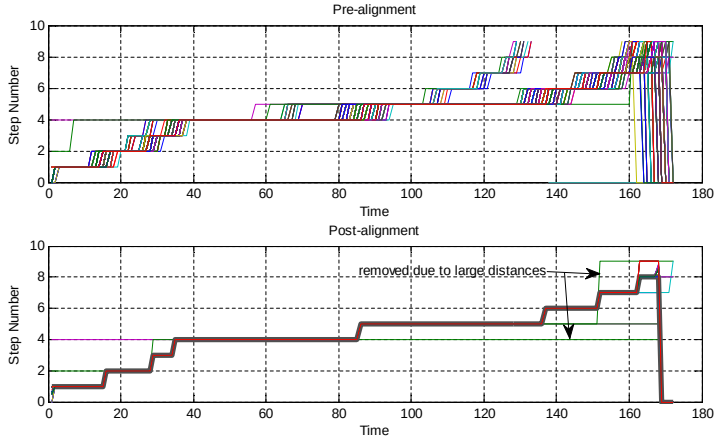


Figure 4.21: Metal etch trajectory alignment using DTW and CsDTW

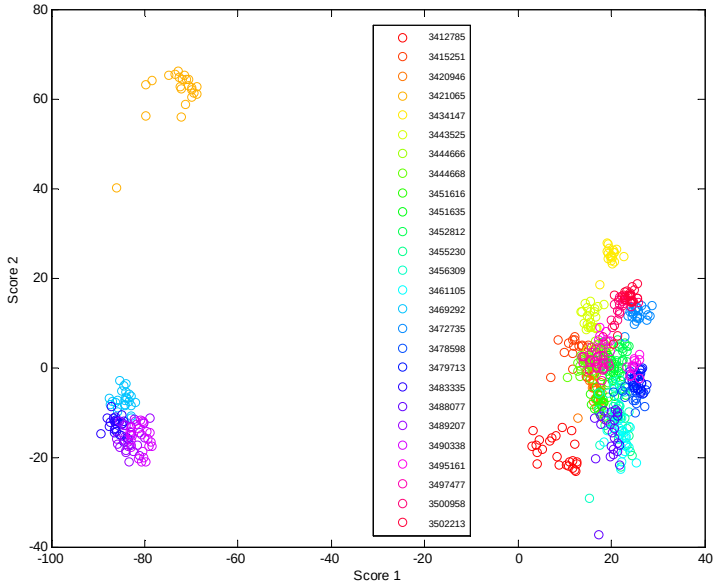


Figure 4.22: Score plot of the full multiway-PCA model

A multiway PLS model utilizing all input variables was first constructed. This preliminary PLS model is used to explore the dataset and not for prediction. Using the trained PLS model, the variable importance in projection (VIP) can be calculated for each input variable. The VIP inputs are then folded back into the original dimensions (batch variable and batch time) and visualized in Figure 4.23. The raw trajectory plots are also shown in the figure to better interpret the results. As shown in the Figure, contiguous time ranges of high VIP correlation indicate that the particular segment of batch trajectory is relevant to the output. These segments are usually ramps, or dips that contain variations across multiple batches. A more refined PLS model can then be built on top of the existing preliminary model by eliminating redundant and uninformative inputs variables, the variables included in the reduced model is shown in Table 4.6. Note that variables such as chamber temperature and pressure are absent from the selected variables listed in Table 4.6. These process variables are usually held constant and controlled very well using standard PID controllers within the tool itself. Batch to batch variations in these process variables are minimal, and thus are identified as not relevant in the subsequent PLS modeling. However, this does not mean that these variables are not important or critical to the operating condition. To fully model the effect of these other influences, design of experiment data or system perturbation studies are required to explore the full operating range of these signals.

Figure 4.24 shows the resulting training and validation performance of the refined PLS model. The right subplot also includes the sheet resistance specification limit of the process. As we can see, the PLS model was able to account for lot-to-lot variations in the sheet resistance but was unable to detect

Table 4.6: List of relevant variables identified through PLS-VIP Variable Selection

Variable Name	VIP value
Tool: ThrottleValvePosition	1.1
Tool: OESB1Value	1.92
Tool: ChuckRFVoltageProbeAIReading (V)	1.2
Tool: ChuckESCCurrentMonitor1Reading (uAmp)	1.67
Tool: ChuckESCBiasVoltageReading (V)	1.53
Tool: BiasRFVoltageProbeAIReading (V)	1.3
Tool: BiasRFCurrentProbeAIReading (amp)	1.2

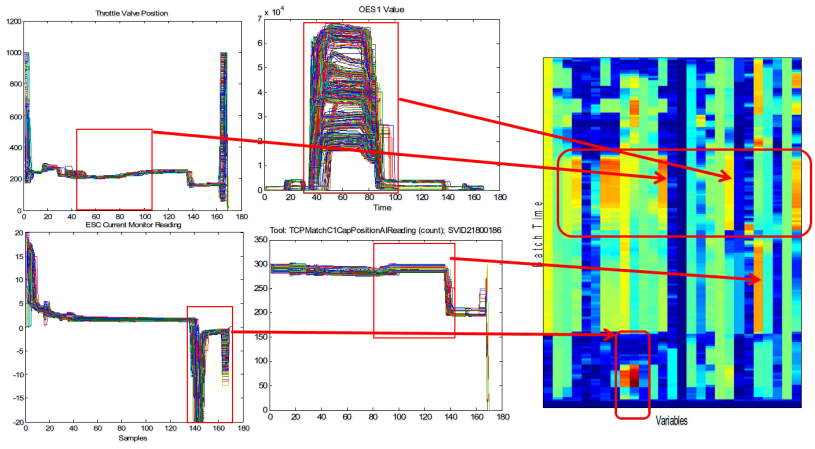
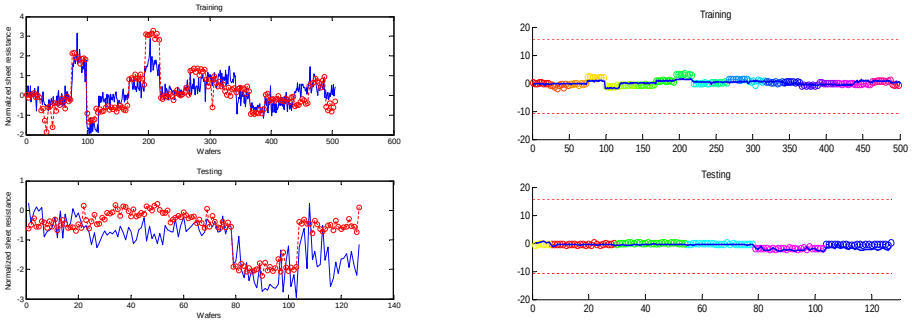


Figure 4.23: Variable Importance in Projection (VIP) map showing key trajectory features that yielded high correlation with the output

variations that takes place within the lot (wafer-to-wafer level variation).



(a) Training vs Testing (b) Training vs Testing with Spec Limits

Figure 4.24: Sheet resistance prediction performance of a metal etch PLS model

The score plot of training and testing data for the reduced PLS model is shown in Figure 4.25. From this plot, we observe that wafers from each lot are clustered together, and the 95% confidence ellipse around the training data encloses all the available data. These observations further confirm that the PLS model is only able to discriminate lot-to-lot differences and does not explain wafer level variations.

To account for wafer-to-wafer level correlation, a finer PLS model for each manufacturing thread needs to be developed, however, this is not possible in a high-mix threaded manufacturing environment since there is not enough training data for each thread. To get around this problem, we can introduce a dynamic component for the reduced model by assuming we have access to metrology measurements from previous runs. A dynamic PLS model can then be created that takes both the trace data and the sheet resistance from previous runs as inputs. The resulting training and prediction performance are shown in Figure 4.26. Table 4.7 lists the training and testing R^2 of all seven models.

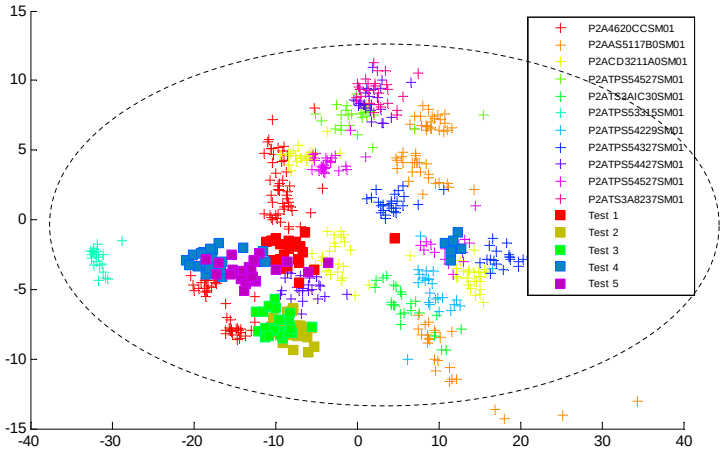


Figure 4.25: Score plot of the reduced PLS model

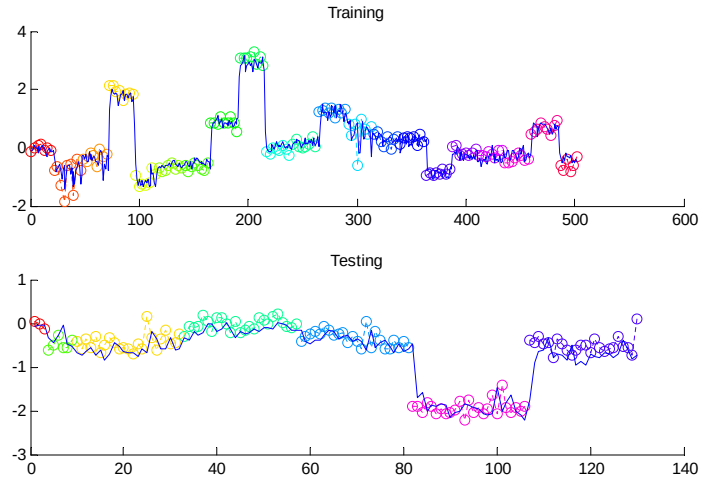


Figure 4.26: Reduced dynamic PLS model performance

To summarize, Table 4.7 lists the performance and description of each model. The reduced and simplified PLS model using summary statistics of the selected trajectory features yielded a testing R^2 of 0.65, which is satisfactory in explaining the lot-to-lot variations in the processed wafers.

Table 4.7: Metal etch PLS model performance summary

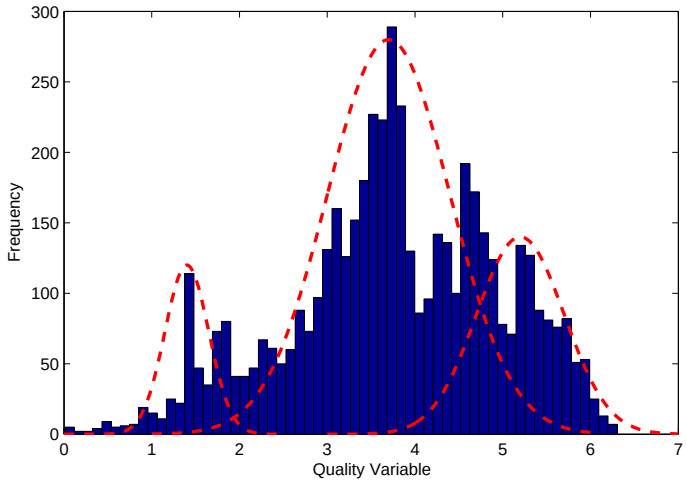
Model #	Model Size	Description	$R^2_{training}$	$R^2_{testing}$
1	2600	Raw inputs	0.53	0.45
2	400	Reduced inputs	0.70	0.65
2	8	Summarized mean	0.67	0.69
3	16	Summarized mean + std	0.72	-0.58
4	16	Summarized mean + max	0.68	0.68
5	16	Summarized mean + sum (integral)	0.67	0.69
6	16	Summarized mean + derivative	0.69	0. 59
7	10	Summarized mean + lagged Rs	0.92	0.91

4.5 Variable Selection Case Studies using Chemical Plant Data

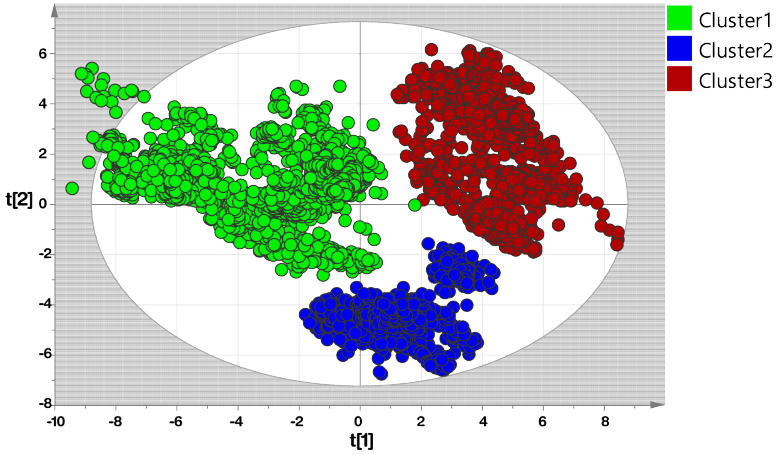
An industrial dataset of a refinery distillation system was provided by The Dow Chemical Company. The dataset contained approximately 12,000 samples of hourly process data. The quality variable was measured at a low sampling frequency through laboratory analysis, which introduced 4-8 hours of time delay into each measurement sample. The quality variable was interpolated to match the sampling frequency of the process variables. A lag analysis by fitting lagged values of quality variables by PLS was done to ensure that sampling lag and interpolation had minimal impact on regression fit. Furthermore, the quality variables are then log-transformed to improve the linear correlation between the input and output. The dataset is divided into three segments. The first segment from year 2011 is taken as training data. The second data segment follows the initial training and lasts for five months (Testing I). The last segment is the second testing dataset (Testing II) and is approximately one year in duration from September 2012 to May 2013.

The Dow dataset provides an example of the multiple operating mode scenario in variable selection. Figure 4.27(a) shows the histogram of the quality variable in the training period, Figure 4.27(b) shows the scatter plot of the PCA decomposed first two principal components. In Figure 4.27(a), three Gaussian distributions are estimated to fit the observed tri-modal distribution. The median for the fitted distributions are 1.5, 3.5 and 5.5 respectively. These three median were verified as the different operating regimes due to changing production priorities. In addition, Figure 4.27(a) indicates that a linear PLS model is not suited to model this process since the resulting prediction will be a single mode Gaussian distribution. In Figure 4.27(b), the scatter plot of the

PCA scores also shows the presence of three clusters. The visible separation between the clusters further indicates that the data are composed of multiple operating modes.



(a) Histogram of the quality variable in training data



(b) Scatter plot of the first two components showing the three operating mode clusters in the training data

Figure 4.27: Multiple operating mode of industrial data visualized in (a) histogram plot and (b) PCA score plot

4.5.1 Optimal model size

Further analysis of prediction performance is based on PLS models with MW-VIP, VIP, SwPA, SwPAi and sPLS variable selection methods. Other methods that show similar results to VIP are excluded for simplicity. Since each model has a different set of tuning rules, comparisons are performed only on the optimized variable selection sets. To determine the optimal model sizes for each method, the cross-validated prediction performance as a function of the number of parameters in each model is plotted in Figure 4.28. The tuning parameter for the number of selected variables is adjusted to return different number of selected variables. A PLS model is built for each set of variables, then the cross-validated Q^2 is calculated and plotted. The tuning parameters adjusted are the filtering threshold for VIP, adjusted VIP ranking for MW-VIP, the COSS score ranking (significance level of ranksum testing) for SwPA and SwPAi, and the sparsity parameter η for sparse PLS. From Figure 4.28, the optimal number of variables is then determined graphically based on visual estimates of the highest Q^2 value. The optimal number of variables for VIP, MW-VIP, SwPA, SwPAi and sPLS were determined to be 6, 7, 7, 10, and 14 respectively. Among these selected variables, three overlapping variables that are highly relevant to the output are present in all selection results. The differences in selection methods therefore resides in the selection of minor correlations. In addition, we observe that all five models exhibit similar trends in how the Q^2 plateaus; therefore, the traditional means of identifying the best performing model by selecting for higher Q^2 alone is challenging.

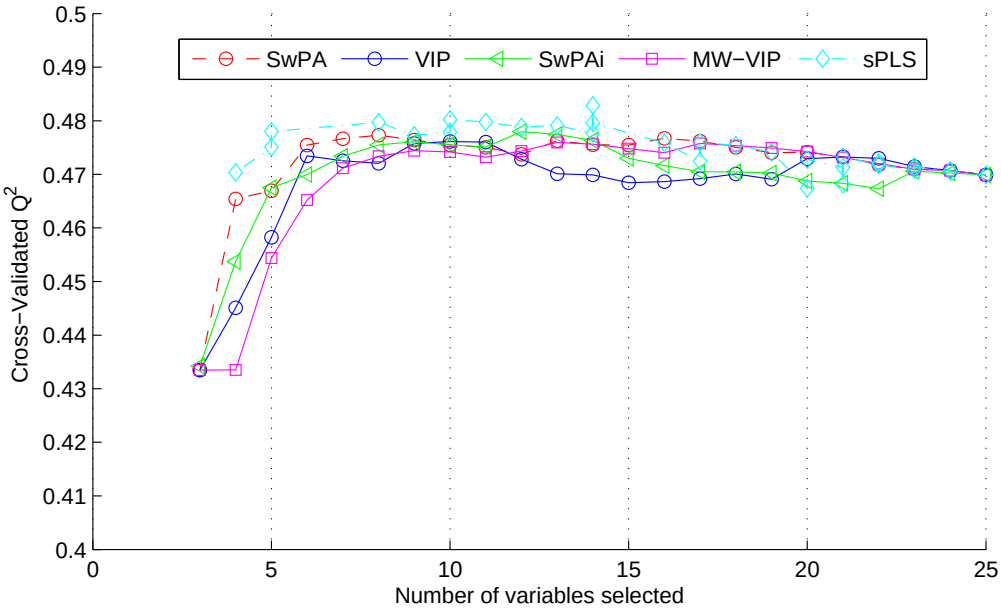
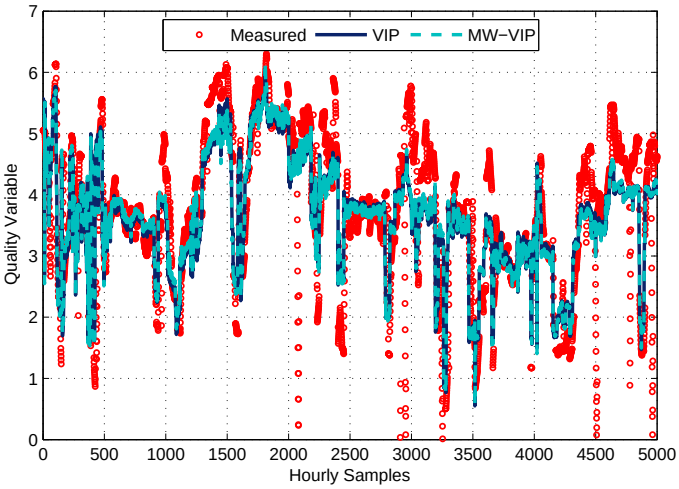


Figure 4.28: Cross-validated Q^2 scores of different variable selection methods as a function of the number of parameters included

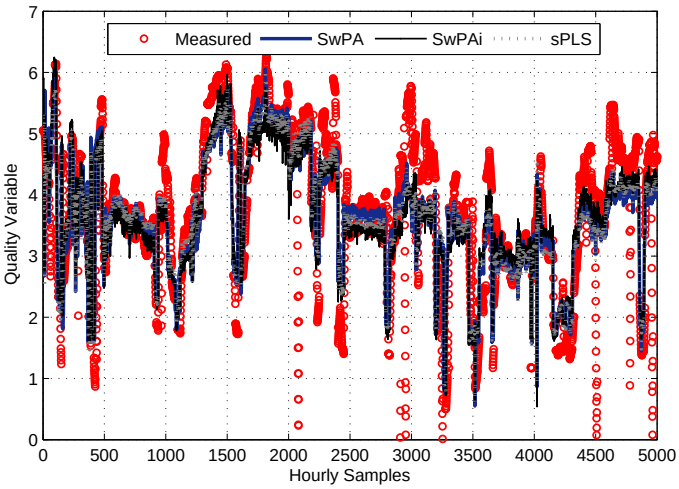
4.5.2 Training and testing results

All of the optimized PLS models are trained using the same training data. The number of principal components are determined individually through cross-validation, the number of components were determined to be around three depending on the number of variables used in modeling. Figures 4.28 and 4.29 show the training performance of the five candidate methods. The unadjusted Q^2 for all methods averaged around 0.50. All PLS models were able to track most of the trends, except in extreme cases where the quality variable was close to zero. Training performances of other methods are similar to VIP and thus are omitted for simplicity.

The calculated models were tested on two testing data sets. Figures 4.30 and 4.31 show time series plots of the five candidate models in testing I and testing II data sets, respectively. In Figures 4.30(a) and 4.30(b), all five candidate models showed bias from the measured value. However, the MW-VIP model was able to reduce the bias and trend the series closer than the other methods. The testing II dataset performance, shown in Figures 4.31(a) and 4.31(b) , contains about 5,500 samples equivalent to about 10 months of production after cleaning. All five candidate models performed better in the testing II dataset than previously during testing I, which indicates that testing II dataset has better resemblance to the original training data. Again, the proposed MW-VIP method behaved better or similar when compared to the other four candidates. In Figure 4.32, the best performing candidate models from each variable selection category are shown. These models are the SwPAi model, the MW-VIP model and the sparse PLS model. From this comparison, it is clear that MW-VIP outperformed other methods in terms of tracking the measured quality variable trajectory.

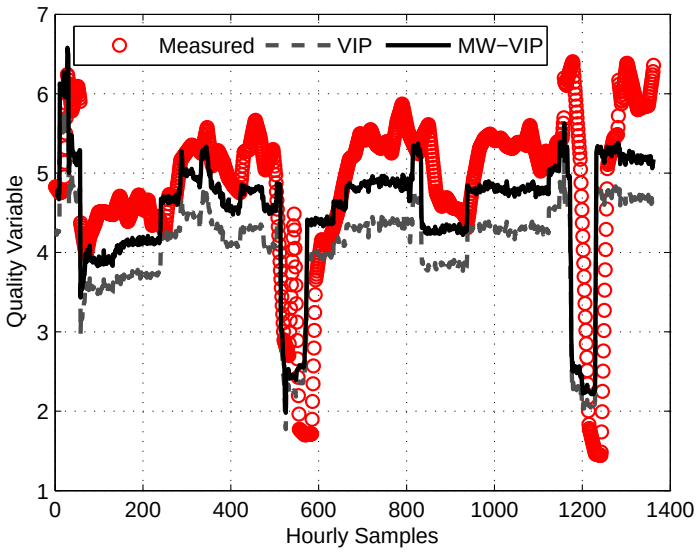


(a) VIP and MW-VIP

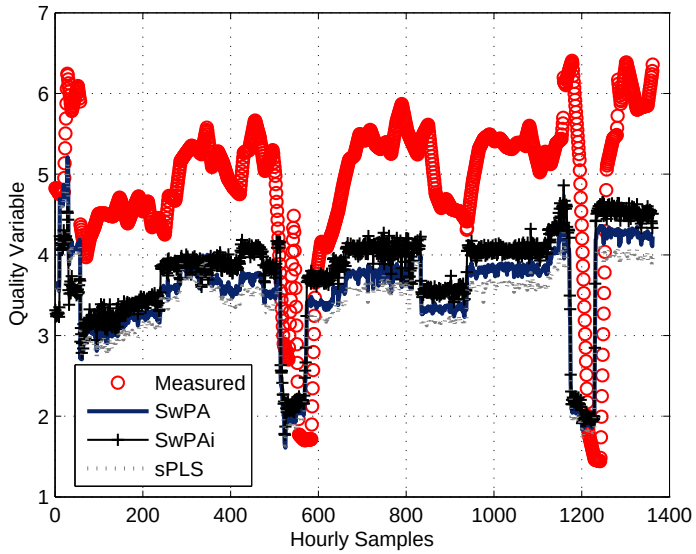


(b) SwPA, SwPAi and sPLS

Figure 4.29: Time series plot of the measured and predicted quality variable for the training data

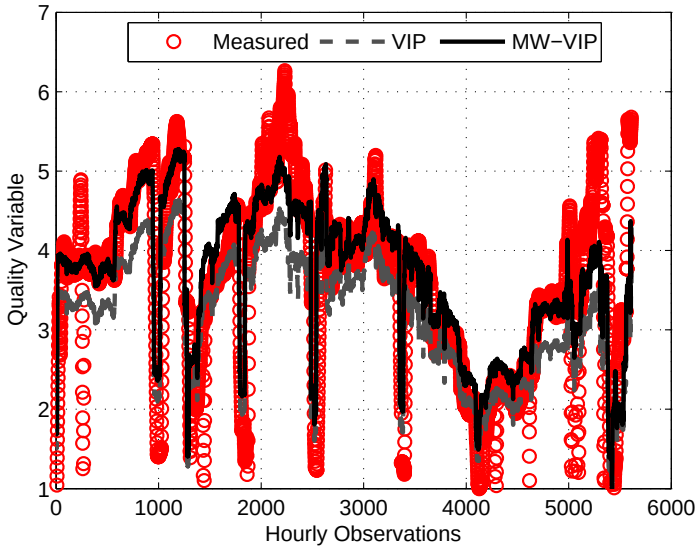


(a) VIP and MW-VIP

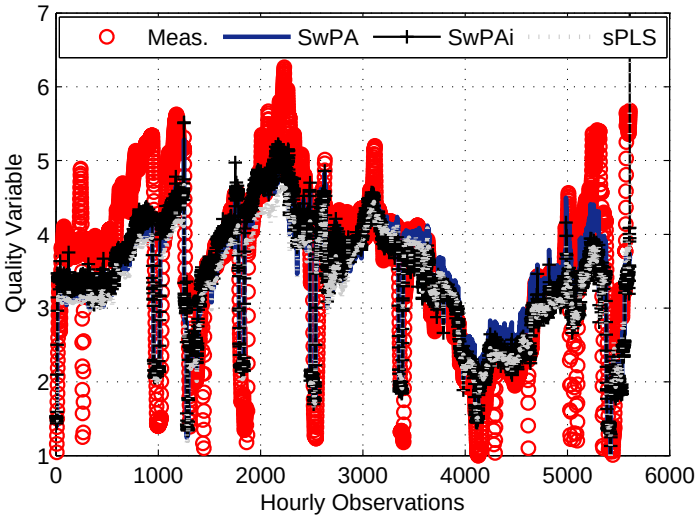


(b) SwPA, SwPAi and sPLS

Figure 4.30: Testing data I model validation of the five candidate models (VIP filtering, MW-VIP, SwPA, SwPAi and sPLS)



(a) VIP and MW-VIP



(b) SwPA, SwPAi and sPLS

Figure 4.31: Testing data II model validation of the five candidate models (VIP filtering, MW-VIP, SwPA, SwPAi and sPLS)

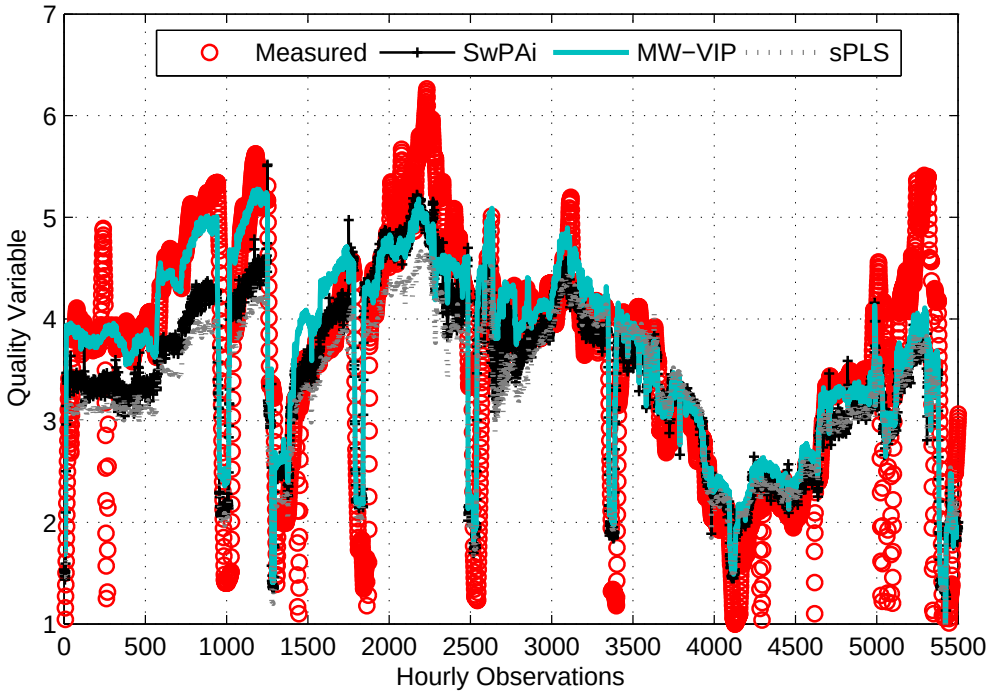


Figure 4.32: Measured and predicted quality variable from VIP filtering, SwPAi, sPLS and MW-VIP models in testing dataset II

To intuitively examine why the MW-VIP model outperformed other methods using similar model structure, the variables selected by each method were further examined. Utilizing the MW-VIP model, the group-to-group contribution plot for two periods of similar quality variable magnitude is shown in Figure 4.33. The MW-VIP approach was able to select variables with high contributions in these specific regions that were not selected in other models. The second and third highest contributing variables are unique to the MW-VIP model (tag 7 and tag 30). The resulting improvement in the model prediction performance shows that selecting locally correlated variables in the overall prediction model will improve the local performance with the advantage of training the model using the full dataset.

The advantage of MW-VIP variable selection is that it is able to filter and select locally correlated variables without performing complex clustering or supervised labeling of training data. As the data from the same operating mode with a particular set of locally correlated variables tends to cluster together, the moving window approach captures the important variables of these correlations by partitioning the entire training data into subsets and benefiting of the time dependency of the process data. This is an effective method in industrial processes since changes in operating conditions can be captured in a simple way utilizing only one model.

4.5.3 Model evaluation results

Tables 4.8 and 4.9 list the evaluation results of the five candidate models for testing I and testing II data sets respectively. The bolded cells in each column highlights the best models for each criterion. In testing I, the MW-VIP model outperformed other models in prediction, AIC, and parameter robust-

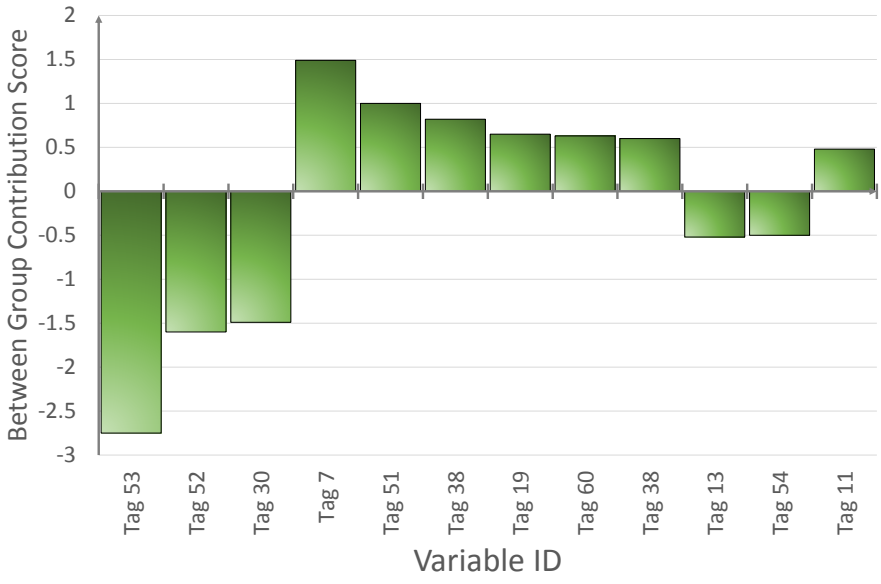


Figure 4.33: Contribution score plot example of the discrepancies in prediction due to local variables identified by MW-VIP variable selection

ness. In testing II, the MW-VIP and SwPA models showed better performance than other models. Although SwPA method had slightly better prediction (R^2 value of 0.46 against 0.45), the robustness score (RDSS) favored the MW-VIP model for its higher robustness. As illustrated in Figure 2.27, the higher robustness score in practice would translate to a model that experiences less degradation in prediction performance over time.

Table 4.8: Results of the evaluation criteria study calculated using testing dataset I

Method	Model Size	Bias	RMSEP	R2	AIC	RDSS	T2 ratio
VIP	6	10.48	1.08	-0.08	-55.344	198.5	*92%*
SwPA	8	19.36	1.9	-0.9	-47.627	204.6	*0%*
SwPAi	10	15.82	1.54	-0.54	-50.488	200.5	*6%*
MW-VIP	9	**3.8**	**0.78**	**0.22**	**-59.725**	**125.8**	*37%*
sPLS	14	22.43	2.3	-1.3	-45.039	204.6	*95%*

Table 4.9: Results of the evaluation criteria study calculated using testing dataset II

Method	Model Size	Bias	RMSEP	R2	AIC	RDSS	T2 ratio
VIP	6	6.65	0.55	0.45	-21.568	50.6	*93%*
SwPA	8	-1.39	0.57	0.43	-21.418	58.1	*86%*
SwPAi	10	**0.38**	**0.54**	**0.46**	**-21.694**	40	*96%*
MW-VIP	9	-5.15	**0.55**	0.45	**-21.632**	**32.2**	*95%*
sPLS	14	3.46	0.56	0.44	-21.461	57	*95%*

The T^2 ratio monitors the multivariate T^2 sensitivity of the trained model for incoming data differences. A high T^2 ratio value corresponds to higher percentage of test data being within the training control limits. Figure 4.34 plots the T^2 ratio as a function of the model prediction performance (R^2) for all three data sets. The expected behavior of the models are outlined

with the ellipse (Shown in Figure 4.34). The sparse PLS and the regular VIP filtering models do not exhibit the expected correlation between their R^2 value and the T^2 sensitivities, which indicates that their correlations are abnormal; these models are unsuited to be utilized in a diagnostic as their T^2 limits are too insensitive to predict changes in both the process variables of the model and the quality variable. In particular, the sparse PLS model has a very low R^2 value due to a constant prediction bias observed in the second testing dataset (see Figure 4.31). The reduced sensitivity of the T^2 indices in this model was caused by the larger model size compared to the rest of the models. The additional variables in the sparse PLS model inflated the T^2 confidence limits, causing abnormal dataset to appear normal. The SwPA and the MW-VIP models fall within the ellipse and follows the expected correlation between R^2 and T^2. Therefore, the MW-VIP and SwPA models have better diagnostic capabilities and are more desirable than the rest of the models developed.

4.5.4 Process experts validation

The model created with the variables selected from MW-VIP and SwPA methods were presented and verified with process operation engineers of the process. Based on a comprehensive evaluation of the material and energy flows inside this process, the run-plant engineering team was able to rationalize and explain how the newly identified variables would affect the process. As a result, the MW-VIP model has been implemented in the plant and utilized to monitor the quality variable online and perform the diagnostic.

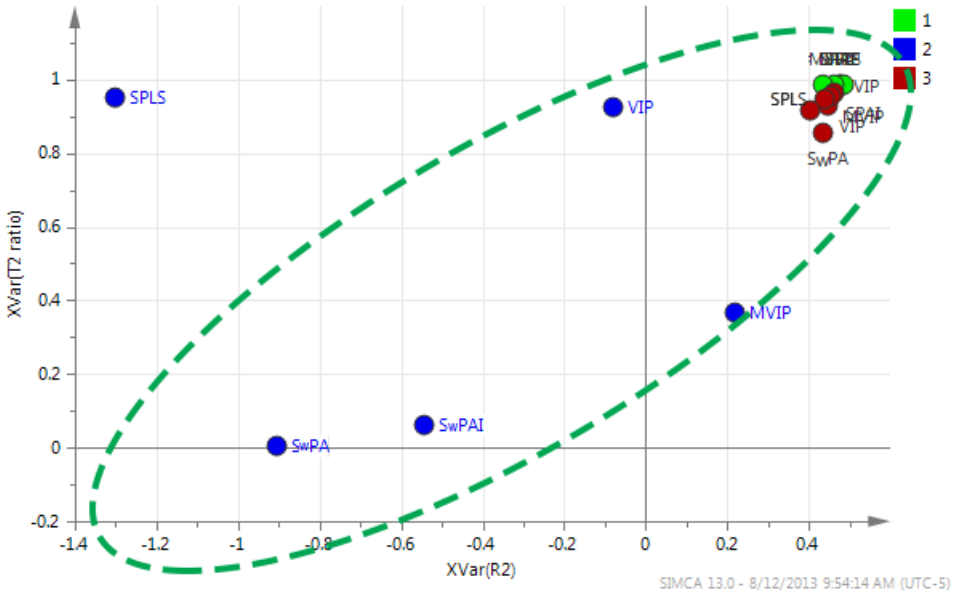


Figure 4.34: Model process monitoring sensitivity (T^2 ratio) as a function of the model fit performance, the ellipse highlights the expected correlation between R^2 and T^2 ratio values

4.6 Summary

In this chapter, we presented three applications of the previously discussed methods using industrial data sets provided by our research collaborators.

In the first application, a set of VM models were developed for a gate etch tool using fault diagnostics trace and metrology data collected on 1800 wafers. The trained model show that multiway PLS based methods were able to predict the etch rate with a R^2 value of 0.5 to 0.7. However, significant process drifts cause performance degradation if stationary models (static PLS) are used. Two adaptive models were therefore implemented to address the problem. The T-PLS moving window PLS model maintained the R^2 value at

around 0.6. The adaptive GSMMS-PLS model was able to maintain the R^2 value at close to 0.8 with less frequent model updates, the adaptive GSMMS-PLS model was also updated only four times instead of over 30 times in the moving window case, making it more friendly for on-line implementation.

Using the predicted VM results, we formulated a VM-assisted control performance assessment criteria that aims to identify model-plant mismatch and improper tuning of EWMA run-to-run controllers. The proposed method makes assumption that the VM model can be used to predict the gain of the process model. A simulated study verified that the proposed technique had faster response times than other performance assessment methods. Furthermore, the proposed method was found to be robust when the VM prediction is corrupted with noise.

In the second application, a VM model for a metal etch process was developed. The VM model for metal etch process aims to identify important contributing variables that correlate to the process output. Through the use of variable importance in projection (VIP), trajectory alignment and refolded VIP visualization methods discussed in previous chapters, the VM model was able to identify the contributing variables that discriminates the output at the lot-to-lot level. Wafer-to-wafer level variations were undetectable because of the limited amount of training data available.

In the third application, data from a chemical process was used to study the various PLS model variable selection techniques. The results demonstrated that the proposed moving window VIP method was able to select a robust set of variables which results in more accurate model predictions while also requiring less frequent updates. The proposed predictor has been validated by process experts and is currently implemented online.

Chapter 5

Summary and Future Recommendation

5.1 Summary of Contributions

With the increasing adoption of data-driven methods in manufacturing industries, the deployment of soft sensors extends beyond the studies of the regression techniques involved in generating the best performing models. This dissertation aims at addressing the issues in data pretreatment and post-model deployment to bridge the gap in wide-spread adoption of data-driven techniques. Although the models developed are for plasma etch datasets, the methods themselves are very general and can be extended for use in other data-driven applications.

First, in dealing with data pretreatment, a new time alignment technique has been proposed in Chapter 2. The new Constrained selective Dynamic Time Warping (CsDTW) algorithm solves a subproblem of identifying key trajectory markers and then utilizes the solution in generating a set of constraints to minimize distortion of the key process trajectories. The warping results in case studies shown are comparable to that of Correlation Optimized Warping (COW) and better than other techniques. Because of the dynamic programming principle used, this polynomial time algorithm is solved much faster than COW and can be implemented online. Chapter 2 also takes a extensive look at variable selection and feature extraction specific to Partial Least Squares models. It was identified that existing techniques usually assume the data set

is stationary and that an optimum combination of variables can be identified through statistical means. The proposed moving window variable importance in projection method does not satisfy this assumption and aims to select the most consistent set of variables for all operating regimes. This selection technique was found to generalize better where future process data correlations are subjected to change.

Chapter 3 presents two new methods developed to improve model maintenance and performance of models with multiple operating states. The first proposed method uses a moving window scheme with Total Projection to Latent Structure (T-PLS) decomposition to screen incoming data. The quality output relevant T-PLS decomposition separates the harmless process noise from the outliers that negatively affects the model. The combination of these two techniques lead to a model update mechanism that is demonstrably more robust. In addition, to reduce the number of model updates and account for multiplicities in the process conditions, a growing structure multiple model system using local PLS and PCA models has been proposed. Previous GSMMS systems uses linear regression local models which limits the number of inputs and suffers some numerical difficulties during training. This multiple model system can handle a much larger set of inputs and overcome typical multiple model system challenges as well. In addition, fault detection sensitivities are increased when using multivariate monitoring statistics of local PLS/PCA models from the GSMMS system.

In Chapter 4, the proposed methods in Chapters 2 and 3 are tested using industrial process data. Using a gate etch dataset, we demonstrated the effectiveness of the proposed trajectory alignment and model update techniques. In addition, the GSMMS model was found to give the best online

prediction with the minimum number of model updates. Using a metal etch data set, variable selection techniques were used to identify the discriminating variables responsible for sheet resistance variations in the final product. Additionally, a simulated case study demonstrates the use of virtual metrology in assessing plant model mismatch and improper tuning of run-to-run controllers. The accuracy versus detection rate trade-off was also investigated.

5.2 Future Work

Despite recent advances in soft sensor and virtual metrology applications, there still remain a number of challenges that should receive further attention. In data preprocessing, the conventional methodology is still relying on multi-way methods by unfolding the process data into two dimensional data arrays. These unfolding techniques requires synchronization, refolding and nonlinear transformations which create additional burden in the analysis process. Statistical pattern analysis has been proposed to summarize the batch profile behavior using summary statistics. But this method targets niche applications where the batch information can be easily described using Gaussian statistics. Alternatively, Tucker3 and other three-way decomposition techniques have been investigated in the context of data visualization and exploration, which do not unfold the data and perform the decomposition in three dimensions. An innovation in regression techniques that could take advantage of three-way decomposed loading matrices would streamline the development process of batch models tremendously.

In terms of selecting features for modeling, we have shown that process correlations change with respect to time. For inferential sensor models that rely on statistics or intuition to reduce the dimensionality, variable selection

of these models needs to be updated in addition to the model coefficients. As plants and manufacturing systems become more heavily instrumented, the number of available inputs variables for models will increase as well; therefore, it is imperative to understand the trade-off and the effects of updating variable selection and model adaptation.

On the topic of multiple model systems, the proposed growing structure multiple model system with local PLS/PCA models does not account for the time dynamics explicitly. Although one can account for the change in process behavior by including age of the data point as an input to the growing self organizing map (GSOM), this solution limits the efficiency of network in grouping similar process conditions together. There has been considerable advances in the field of machine learning in dealing with time-dependent data. Recent works such as neural gas, incremental grid growing, dynamic growing self organizing map and growing cell structures all have different growing mechanisms that account for the time dimension explicitly. By studying these in more detail, the growing structure multiple model system framework can be refined to handle drifting behavior better than the current implementation.

Appendices

Appendix A

Publications

1. Shu Xu, Bo Lu, Michael Baldea, Thomas F. Edgar, Willy Wojsznis, Terrence Blevins, and Mark Nixon. Data cleaning in the process industry. Reviews in Chemical Engineering, *submitted*, 2015

2. Bo Lu, John Stuber, and Thomas F. Edgar. Adaptive linear projection multiple model system modeling of batch semiconductor virtual metrology, AICHE Journal, *to be submitted*, 2015

3. Bo Lu, John Stuber, and Thomas F. Edgar. Constrained selective Dynamic Time Warping for Alignment of Batch Trajectories in Soft Sensor Modeling. Chemometrics, Computers & Chemical Engineering, *to be submitted*, 2015

4. Bo Lu, Ivan Castillo, Leo Chiang, and Thomas F. Edgar. Industrial PLS model variable selection using moving window variable importance in projection. Chemometrics and Intelligent Laboratory Systems, 135:90-109, July 2014.

5. Bo Lu, John Stuber, and Thomas F. Edgar. Integrated Online Virtual Metrology and Fault Detection in Plasma Etch Tools. Industrial & Engineering Chemistry Research, 53(13):51725181, April 2014.

6. Yang Zhang, Bo Lu, and TF Edgar. Batch Trajectory Synchronization with Robust Derivative Dynamic Time Warping. Industrial & Engineering Chemistry Research, 2013.

Appendix B

Common Multivariate Methods and Applications

B.1 Principal Component Analysis and Multivariate Monitoring

B.1.1 Principal component analysis

Principal component analysis (PCA) is a widely used dimensionality reduction and data visualization technique. A detailed discussion on PCA can be found in [121]. Here we will briefly introduce the key ideas in PCA with respect to process monitoring. PCA in process monitoring aims to build a reduced representation of data during normal process operation. PCA is very effective in extracting variable correlation in the process data. The identified correlation can then be summarized into compact multivariate monitoring statistics that can then be used to monitor future data. A change in process condition usually results in a change in the observed correlation and lead to an alarm indicating process abnormality. This idea forms the basic methodology which latent projection based process monitoring utilizes.

Let $\mathbf{x} \in \mathbb{R}^m$ be a sample vector of m measurements from a single time instance. Assume that we have N samples, then the total data collected during this period constitutes $\mathbf{X} \in \mathbb{R}^{N \times m}$. If we convert the measurements in $\mathbf{X}$ into deviation variables with a mean value of 0, then we can apply PCA to decompose $\mathbf{X}$ into a score matrix $\mathbf{T}$ and a loading matrix $\mathbf{P}$. This decomposi-

tion can be performed through NIPALS[86] or singular value decomposition. The decomposition can be expressed as:

$$\mathbf{X} = \hat{\mathbf{X}} + \tilde{\mathbf{X}} = \mathbf{T}\mathbf{P}^T + \tilde{\mathbf{X}} \tag{B.1}$$

where $\hat{\mathbf{X}}$ represents the principal subspace and $\tilde{\mathbf{X}}$ represents the residual subspace.

If we assume the PCA decomposition is full rank (the number of principal component is equal to the number of variables), the following condition holds:

$$\mathbf{S} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^{\mathbf{T}} \tag{B.2}$$

where $\boldsymbol{\Lambda} = \frac{\mathbf{1}}{\mathbf{N-1}}\mathbf{T}^{\mathbf{T}}\mathbf{T} = \text{diag}\left\{\lambda_{\mathbf{1}}, \lambda_{\mathbf{2}}, ...\lambda_{\mathbf{m}}\right\}$. The PCA decomposition basically rotates the point of view to where the variations in the first coordinate direction is maximized. The covariance matrix $\boldsymbol{\Lambda}$ of the latent scores $\mathbf{T}$ is therefore diagonal.

To perform dimensionality reduction, the first A principal components and their associated loadings is kept while the rest are truncated. Since the variance in each latent coordinate direction is sorted in descending order, the dimensionality reduction will maximize the amount of information retained in the remaining principal subspace $\hat{\mathbf{X}}$. The number of principal component A can be determined by calculating percentage of variance explained as follows:

$$\text{variance explained}\,(A) = \sum_{i}^{A} \lambda_i / \sum_{i}^{M} \lambda_i \tag{B.3}$$

B.1.2 Squared prediction error

The squared prediction error measures the magnitude of the projection of an incoming data sample in the residual subspace. It is defined as:

$$\text{SPE} \equiv \|\tilde{\mathbf{x}}^2\| = \|(\mathbf{I} - \mathbf{P}\mathbf{P}^{\mathrm{T}})\|^2 \tag{B.4}$$

A large value of SPE would indicate that the current PCA identified directions is unable to explain the variance in the incoming data sample. The process is only considered normal if

$$\text{SPE} \leq \delta_\alpha^2 \tag{B.5}$$

where δ_α^2 is the upper control limit at significance level α. An expression forδ_α^2 has been developed by Jackson and Mudholkar [122].

$$\delta_\alpha^2 = \theta_1 \left(\frac{c_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_0 \left(h_0 - 1\right)}{\theta_1^2} \right)^{1/h_0} \tag{B.6}$$

where $\theta_i = \sum_{j=l+1}^{m} \lambda_j^i, i = 1, 2, 3, h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2}$. l is the number of principal components and c_α is the normal deviate corresponding to the upper $1 - \alpha$ percentile, λ_j corresponds to the eigenvalue of component j in the PCA decomposition. This control limit assumes the following condition:

- sample vector $\mathbf{x}$ follows a multivariate Gaussian distribution
- The distribution is approximated using a polynomial, which is only valid for large θ_1
- The result is suitable for any number of principal components

An alternative upper control limit has been proposed by Eriksson et. al. [69] called the distance to model residuals in X (DMODX). First the overall

accumulated residual sum of squared for the PCA model is calculated as:

$$S_0 = \sqrt{\frac{\Sigma\Sigma e_{ij}^2}{(N-A-1)(K-A)}} \tag{B.7}$$

where K is the number of variables, A is the number of principal components and N is the number of training samples. The ratio of the sample SPE versus the training total SPE then follows a F-distribution.

$$(s_i/S_0)^2 \sim F(1-\alpha, K-A, (N-A-1)(K-A)) \tag{B.8}$$

where α is the significance level. The actual performance of the DMODX is identical to the SPE statistic in practice.

B.1.3 Hotelling's T^2 statistic

Hotelling's T^2 statistic is complementary to the squared prediction error statistic. It measures the variation within the projected principal subspace of a PCA model. The Hotelling's T^2 is calculated as follows:

$$T^2 = \mathbf{x}^T\mathbf{P}\mathbf{\Lambda}^{-1}\mathbf{P}^T\mathbf{x} \tag{B.9}$$

where $\mathbf{P}$ is the loading matrix and $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues in the PCA decomposition.

Given that the process is normal and the process data follows the assumption of a multivariate Gaussian distribution, the T^2 statistic follows a F-distribution of the following form:

$$\frac{N(N-A)}{A(N^2-1)}T^2 \sim F_{\alpha,A,N-A} \tag{B.10}$$

where $F_{A,N-A}$ is a F-distribution with $(A, N-A)$ degrees of freedom, N is the number of samples and A is the number of principal components. We can

derive the control limit at significance level $(1-\alpha)$ as follows:

$$T^2 \leq T_\alpha^2 \equiv \frac{A\left(N^2-1\right)}{N\left(N-A\right)} F_{A,N-A,\alpha} \tag{B.11}$$

B.1.4 Mahalanobis distance

The Mahalanobis distance is a special case of Hotelling's T^2 statistic that uses the full rank of the input matrix $\mathbf{X}$. It is defined as follows:

$$D = \mathbf{x}^T \mathbf{S}^{-1} \mathbf{x} \sim \frac{m\left(N^2-1\right)}{N\left(N-m\right)} F_{m,N-m} \tag{B.12}$$

where $\mathbf{S}$ is the sample covariance of $\mathbf{x}$. For datasets with singular covariance matrix, a pseudoinverse of $\mathbf{S}$ can be substituted in place and results in the reduced rank Mahalanbois distance measure:

$$D_r = \mathbf{x}^T \mathbf{S}^\dagger \mathbf{x} \sim \frac{r\left(N^2-1\right)}{N\left(N-r\right)} F_{r,N-r} \tag{B.13}$$

where $\mathbf{S}^\dagger$ is the Moore-Penrose pseudoinverse[123] and r is the reduced rank of the covariance matrix $\mathbf{S}$. It can be shown that the Mahalanobis distance is related to the Hotelling's T^2 in the following relationship.

$$D = T^2 + T_H^2 \tag{B.14}$$

where T_H^2 is the Hawkin's T^2 statistics, a symmetric implementation of the regular Hotelling's T^2 in the residual subspace [124].

B.1.5 Combined index

A single index that combines the Hotelling's T^2 and the squared prediction error has been proposed by Yue and Qin [47] that simplifies the need

for two separate control charts in performing multivariate process monitoring. The combined index is defined as:

$$\varphi = \frac{\mathrm{SPE}\,(\mathbf{x})}{\delta_\alpha^2} + \frac{T^2\,(\mathbf{x})}{\chi^2_{A,\alpha}} = \mathbf{x}^T \mathbf{\Phi} \mathbf{x} \tag{B.15}$$

where

$$\mathbf{\Phi} = \frac{\mathbf{P \Lambda^{-1} P^T}}{\chi^{\mathbf{2}}_{\mathbf{A},\alpha}} + \frac{\mathbf{I - P P^T}}{\delta^{\mathbf{2}}_{\alpha}} \tag{B.16}$$

The upper control limit for this monitoring index can be derived by using an approximate distribution as follows:

$$\varphi = \mathbf{x^T \Phi x} \sim \mathbf{g} \chi^{\mathbf{2}}_{\mathbf{h}}, \mathbf{g} = \mathbf{tr}(\mathbf{S\Phi})^{\mathbf{2}} / \mathbf{tr}\,(\mathbf{S\Phi}) \tag{B.17}$$

where h is the degree of freedom of the χ^2 distribution.

Although the combined index simplifies the use of multivariate control chart, it is more difficult to interpret the source of problem. An excursion in T^2 indicates extrapolation of the identified correlation, whereas an excursion in SPE indicates that the incoming data contains a different correlation structure than the training data. The combined index would alarm in both cases, making it difficult to pinpoint the root cause of the problems.

B.2 Partial Least Squares Based Prediction and Monitoring

Partial least squares (PLS) methods are widely used to analyze large data sets such as those encountered in plasma etching ([83, 13, 12]) A typical plasma etch dataset contains over 30 sensor measurements, which include gas flow, chamber conditions, and RF circuitry readings. When sensor signals are unfolded, these sensor readings result in over 3000 variables. Unfolding

transforms a three-dimensional data array into a two-dimensional matrix for the purpose of performing PCA and PLS [5]. The feasibility of using a multi-way approach (unfolding) for PLS/PCA analysis was demonstrated by Wold et al. and Nomikos et al. in process industry monitoring [84, 85]. The two mainstream PLS algorithm are the Nonlinear Iterative PLS (NIPALS) and the SIMPLS[53, 52]. Seven additional PLS algorithms are evaluated and reviewed by Andersson [86]. Both PLS algorithms decompose the mean-centered matrices into the following form:

$$\begin{aligned} \mathbf{X} &= \mathbf{T}\mathbf{P}^T + \mathbf{E} \\ \mathbf{Y} &= \mathbf{U}\mathbf{Q}^T + \mathbf{F} \end{aligned} \tag{B.18}$$

where $\mathbf{T} \in \mathbb{R}^{n \times A}$ and $\mathbf{U} \in \mathbb{R}^{n \times A}$are the $\mathbf{X}$ and $\mathbf{Y}$ scores respectively, $\mathbf{P} \in \mathbb{R}^{m \times A}$ and $\mathbf{Q} \in \mathbb{R}^{p \times A}$ are the loadings for $\mathbf{X}$ and $\mathbf{Y}$, respectively. The number of components in the PLS model is typically determined through cross-validation or through information criterion such as the Akaike Information Criterion (AIC) [87]. The PLS algorithm maximizes the covariance between the X-scores and Y-scores; this property leads to PLS requiring fewer components when compared to principal component regression models [87].

To apply the PLS latent structures in regression, given unfolded input data matrix $\mathbf{x}_0$, the output predictions $\hat{\mathbf{y}}_0$ can be calculated linearly using $\hat{\mathbf{y}}_0 = \mathbf{x}_0\hat{\beta}_{pls}$. The $\hat{\beta}_{\text{pls}}$ can be expressed as a function of the latent variables as follows:

$$\hat{\beta}_{\text{pls}} = \mathbf{R}\left(\mathbf{T}^T\mathbf{Y}\right) = \mathbf{R}\mathbf{R}^T\mathbf{X}\mathbf{Y} \tag{B.19}$$

where $\mathbf{R}$ is the weight matrix in the SIMPLS algorithm. In the NIPALS method, the weight matrix $\mathbf{R}$ can also be calculated iteratively.

The multivariate statistics Hotelling's T^2 and the squared prediction error can also be calculated in PLS decompositions as follows:

$$\begin{aligned} T^2 &= \mathbf{t}_0^T \mathbf{\Lambda}^{-1} \mathbf{t}_0 \sim \frac{A\left(n^2-1\right)}{n\left(n-A\right)} F_{A,n-A} \\ SPE &= \left\| \mathbf{x}_0 - \mathbf{t}_0 \mathbf{p}_0 \right\|^2 \sim g\chi_h^2 \end{aligned} \tag{B.20}$$

where $\mathbf{t}_0$ and $\mathbf{p}_0$ are the PLS decomposed scores and loadings for the data batch being tested, and A is the number of components in the PLS model. $\mathbf{\Lambda}$ is defined by $\mathbf{\Lambda} = \frac{1}{n-1}\mathbf{T}^T\mathbf{T}$. Given significance level α, control limits for T^2 and SPE can be calculated from the Fischer and the χ^2 distributions respectively. The Hotelling T^2 statistic detect mean shifts from PLS/PCA score vectors as an indicator of process operation normality. The SPE estimates the magnitude of model residual for incoming data, where a deviation would indicate degrading model performance or abnormal incoming data.

Re-writing Equation B.18 into $\mathbf{X} = \mathbf{TP} + \mathbf{E} = \hat{\mathbf{X}} + \tilde{\mathbf{X}}$, we denote $\hat{\mathbf{X}}$ as the principal subspace and $\tilde{\mathbf{X}}$ as the residual subspace. Since PLS models maximize the covariance between scores and output, the variations in $\tilde{\mathbf{X}}$ are not minimized in PLS as done in principal component analysis. As a result, the PLS principal subspace contains excess variation that is not relevant to output quality changes. Second, the residual subspace ($\tilde{\mathbf{X}}$) also contains excess variation for the same reason; the excess variation causes inflated control limits and poor detection rates.

B.3 Nonlinear Extensions of PLS and PCA

PCA and PLS faces challenges when the input and output relationship is nonlinear. However, multiple options exist depending on the type of nonlin-

earity encountered. Qin and McAvoy developed neural network PLS (NNPLS) that approximates the latent relationship between the input and output scores using a feedforward neural network[88]. Kernel PCA and PLS maps original $\mathcal{X}$ space data into a nonlinear feature space $\mathcal{F}$, where the relationship can be represented linearly [89]. Another method to approximate nonlinearity is by constructing multiple models for each operating regime. The appropriate operating regime can then be selected at run-time based on classification algorithms such as clustering or decision-trees.

B.3.1 Kernel based PCA/PLS

Kernel based methods allows linear based classifier and regression algorithms to deal with nonlinear data. Many algorithms such as Support Vector Machines (SVM), Fisher discriminant analysis (FDA), PCA and PLS can be modified to use kernels. The main idea of kernel based method is to map a set of inputs into a feature space $\mathcal{F}$ [125]. Inner product operations in this future space can be defined using the input in the original space (called the Kernel trick). For example, for kernel PCA, we first define the feature space covariance matrix as:

$$C = \frac{1}{n}\sum_{j=1}^{n} \Phi\left(\mathbf{x}_j\right) \Phi(\mathbf{x}_j)^T \tag{B.21}$$

where $\Phi(\cdot)$ is a nonlinear kernel function that transforms $\mathbf{x}$ into $\mathcal{F}$.

We can then solve for the eigenvectors in the feature space by solving the eigenvalue problem:

$$\lambda \mathbf{V} = C\mathbf{V} = \frac{1}{n}\sum_{j=1}^{n} \Phi\left(\mathbf{x}_j\right) \mathbf{V} \Phi\left(\mathbf{x}_j\right) \tag{B.22}$$

To avoid computing $\Phi(\cdot)$ explicitly, we note that the eigenvectors span the

feature space, this can be represented as follows:

$$\mathbf{V} = \sum_{i=1}^{n} \alpha_i \Phi\left(\mathbf{x}_i\right) \tag{B.23}$$

Define the inner product in feature space as:

$$K_{ij} := \left(\Phi\left(\mathbf{x}_i\right) \cdot \Phi\left(\mathbf{x}_j\right)\right) = k\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{B.24}$$

We can re-write Equation B.22 by multiplying $\Phi(\mathbf{x_k})$ on both sides.

$$\lambda \Phi\left(\mathbf{x}_k\right) \mathbf{V} = \left(\Phi\left(\mathbf{x}_k\right) C\mathbf{V}\right) \tag{B.25}$$

$$\lambda \left(\Phi\left(\mathbf{x}_k\right) \cdot \sum_{i=1}^{n} \alpha_i \Phi\left(\mathbf{x}_i\right) \right) = \Phi\left(\mathbf{x}_k\right) \cdot C \sum_{i=1}^{n} \alpha_i \Phi\left(\mathbf{x}_i\right) \tag{B.26}$$

$$\lambda \alpha = K\alpha \tag{B.27}$$

The system now only depends on the coefficients of the eigenvectors in the feature space and can be solved like a regular eigenvalue problem. In other words, every linear algorithm that uses scalar products in $\mathcal{F}$ can be computed implicitly without knowing the mapping function Φ, this is called the Kernel trick. To ensure normalized scores and centering of data in feature space, a modified kernel matrix can be used:

$$\hat{K} = K - 1_n K - K 1_n + 1_n K 1_n \tag{B.28}$$

where $(1_n)_{ij} = 1/n$, additional details is provided in [125].

Table B.1 lists some of the commonly used kernel functions:

Kernel methods is not without disadvantages, selecting the right kernel function is usually a trial-and-error process and can be challenging in real world applications [86].

Table B.1: Common kernel functions used in kernel PCA, kernel SVM and kernel FDA

Kernel Type	Expression
Gaussian radial basis function	$k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{c}\right)$
Polynomial	$k(\mathbf{x},\mathbf{y}) = ((\mathbf{x}\cdot\mathbf{y})+\theta)^d$
Sigmoidal	$k(\mathbf{x},\mathbf{y}) = \tanh(\kappa(\mathbf{x}\cdot\mathbf{y}+\theta)$
Inverse Multi-quadratic	$k(\mathbf{x},\mathbf{y}) = \frac{1}{\|\mathbf{x}-\mathbf{y}\|^2+c^2}$

B.3.2 Nonlinear neural network PLS

Another nonlinear extension of PLS is nonlinear neural network PLS (NNPLS) [88]. In NNPLS, the linear latent relationship between the input scores and the output scores is modeled using feedforward neural network. Using this approach, nonlinearity in the latent score space can be approximated to arbitrary precision with increasingly complex neural network structure. A modified NIPALS algorithm was proposed by Qin and McAvoy to train the latent nonlinear neural net. The detailed algorithm is available in [88] and not discussed here.

NNPLS faces challenges when the nonlinearity in data cannot be effectively approximated in the latent projected subspaces. In other words, since the outer projection in NNPLS is still a linear operation, the nonlinear data when projected onto the principal component planes just appears as process noise, and thus cannot be modeled effectively using neural net. This is the case for batch process data from the plasma etch reactor, where the inner relationship appears noisy rather than nonlinear and cannot be improved with the application of NNPLS technique.

B.4 Batch Process Analysis

Batch processes such as the plasma etching usually contains multiple measurements at multiple time samples for multiple batches, the three dimensional data cube cannot be analyzed using conventional multivariate techniques. They need to be converted into two dimensional standard data arrays through the process of unfolding. The unfolding is illustrated graphically in Figure B.1. Modeling techniques utilizing unfolded representation of batch data are called multiway methods.

B.4.1 Data unfolding

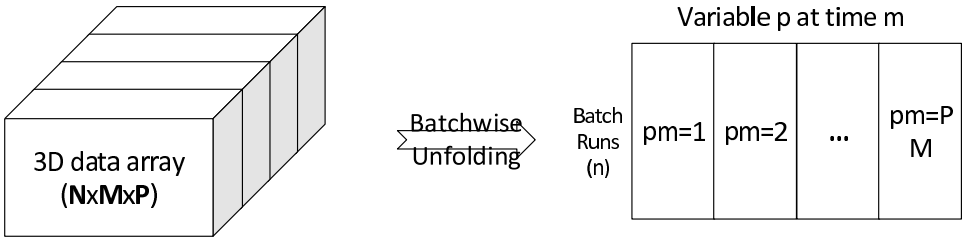


Figure B.1: Unfolding of a three-dimension data array($\mathbb{R}^{n\times m\times p}$) into a two-dimensional matrix ($\mathbb{R}^{n\times mp}$)

Multiway methods have been studied in detail by [126], [127], [128], [31],[129]. Kourti and MacGregor [126] first applied multiway PLS in predicting end-of-batch quality variables for a chemical reactor. Wise et al. [22] applied multiway techniques in modeling a plasma etch process. Chiang et al. [130] assessed the use of multiway techniques and three-way models for industrial fermentation processes.

The process of unfolding in multiway methods increases the number of inputs dramatically, the unfolded $\mathbf{X}$ has $(N\times MP)$ dimensions. The high num-

ber of inputs causes numerous numerical problem in conventional regression and multivariate techniques.

B.4.2 Three-way models

Three-way models aim to bypass the matrix unfolding step that introduces the dimensionality problem in subsequent analysis. The three most common three-way methods are Tucker3, Parallel factor and alternating least squares analysis.

Tucker3

The Tucker3 model with orthogonal factors is also known as three-way PCA, which allows for different number of factors for each of the dimensions. Illustration of the Tucker3 analysis is shown in Figure B.2. The Tucker3 decomposition transforms a three dimensional dataset into several matrices. As a example, given a dataset with dimensionality of N batches, Mvariables and P time samples, Tucker3 give us the following matrices:

- batch mode loading matrix $\mathbf{A}$: $N \times I$
- variable mode loading matrix $\mathbf{B}$: $M \times J$
- time mode loading matrix $\mathbf{C}$: $P \times K$
- core matrix $\mathbf{G}$: $I \times J \times K$
- residual matrix $\mathbf{E}$: $N \times M \times P$

Interpreting the decomposition results requires the analysis of the core matrix $\mathbf{G}$ with respect to the loading matrices of each dimension. Generally, biggest

entry in **G** corresponds to the most important component in the data [131]. There are several applications of Tucker3 models. Garcia et al. used Tucker3 models in determining Clenbuterol concentration through the analysis of gas chromatography and mass spectrometry data [132]. Stefanov et al. analyzed an abnormal process condition called "cockle" formation in paper production using three-way models [133]. De Juan and Tauler assessed the uniqueness of three-way methods using a chemical analyzer dataset [35].

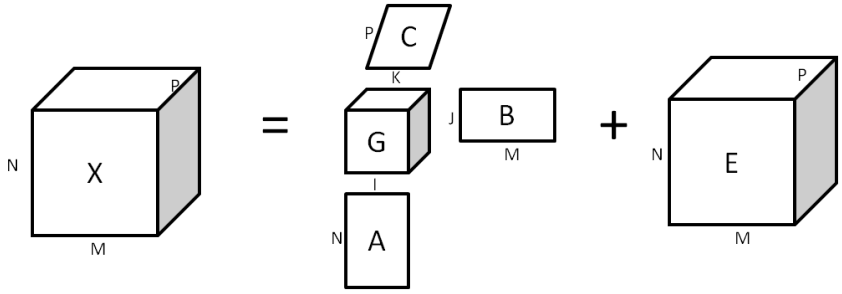


Figure B.2: Tucker3 decomposition of a three-dimensional batch data structure

Parallel factor analysis (PARAFAC)

Parallel factor analysis (PARAFAC) is another three-way data visualization method that decomposes the batch data matrix directly. Krooenberg [134] showed that Tucker3 model is equivalent to a PARAFAC model when the rank of the core matrix is 2. The argument for using PARAFAC as opposed to multiway components is the same as other N-way methods (such as Tucker3) in a sense that this method is truly designed three dimensional datasets. Wise et al. has performed a evaluation of this method in comparison with multiway PCA and regular PCA in monitoring a semiconductor batch dataset [22]. The challenge with PARAFAC and other three-way models is that the statistical properties of the loadings and residuals in the resulting decomposition are not

very well understood. It is difficult to determine the degrees of freedom used in the calculation. As a result, one often needs to make simplifying assumptions (such as Gaussian distributions) that defeats the purposes of using a three-way method.

Appendix C

Closed Loop Run-to-Run Control System Responses Derivation

Given the following definitions:

$$
\begin{aligned}
G_p &= \beta \\
\hat{G}_p &= b \\
\alpha_k &= \frac{1-\theta q^{-1}}{1-q^{-1}}\epsilon_k \\
G_c &= \frac{1}{b} \\
G_E &= \frac{\omega q^{-1}}{1-(1-\omega)\,q^{-1}}
\end{aligned}
$$

and the following internal model control control structure, we can write the output as system based on white noise input disturbance and the target set-point offset.

$$
\begin{aligned}
y_k &= (r_k - \tilde{y}_k)\, G_c G + \alpha_k \\
\tilde{y} &= \left(\frac{y_k - G_c \hat{G} r_k}{1 - G_c \hat{G}} \right) \\
y_k &= \left[r_k - \left(\frac{y_k - G_c \hat{G} r_k}{1 - G_c \hat{G}} \right) \right] G_c G + \alpha_k \\
y_k &= r_k G_c G - \frac{y_k G_c G}{1 - G_c \hat{G}} + \frac{\left(G_c \hat{G} r_k \right) G_c G}{1 - G_c \hat{G}} + \alpha_k \\
y_k \left(1 + \frac{G_c G}{1 - G_c \hat{G}} \right) &= r_k G_c G \left(1 + \frac{G_c \hat{G}}{1 - G_c \hat{G}} \right) + \alpha_k \\
y_k &= \left[r_k G_c G \left(\frac{1 - G_c \hat{G}}{1 - G_c \hat{G}} + \frac{G_c \hat{G}}{1 - G_c \hat{G}} \right) + \alpha_k \right] \times \frac{1}{\left(1 + \frac{G_c G}{1 - G_c \hat{G}} \right)} \\
y_k &= \left[r_k \frac{G_c G}{1 - G_c \hat{G}} + \alpha_k \right] \times \frac{1 - G_c \hat{G}}{1 - G_c \hat{G} + G_c G} \\
y_k &= \frac{G_c G \left(1 - G_c \hat{G} \right)}{\left(1 - G_c \hat{G} \right) \left(1 - G_c \hat{G} + G_c G \right)} r_k + \frac{1 - G_c \hat{G}}{1 - G_c \hat{G} + G_c G} \alpha_k
\end{aligned}
$$

Re-arrange for $r_k - y_k$

$$
y_k - r_k = \frac{G_c G - G_c^2 G \hat{G} - \left(1 - G_c \hat{G} \right) \left(1 - G_c \hat{G} + G_c G \right)}{\left(1 - G_c \hat{G} \right) \left(1 - G_c \hat{G} + G_c G \right)} r_k + \frac{1 - G_c \hat{G}}{1 - G_c \hat{G} + G_c G} \alpha_k
$$

Substituting each respective block using their definition and then further simplify, we obtain the closed loop system error equation:

$$
\begin{aligned}
e_k &= \frac{1 - G_c\hat{G}}{1 + G_c\left(G - \hat{G}\right)} r_k - \frac{1 - G_c\hat{G}}{1 + G_c\left(G - \hat{G}\right)} \alpha_k \\
e_k &= \frac{1 - \frac{\omega q^{-1}}{[1-(1-\omega)q^{-1}]}}{1 + \frac{\omega q^{-1}}{b[1-(1-\omega)q^{-1}]}\left(\beta - b\right)} r_k - \frac{1 - \theta q^{-1}}{1 - q^{-1} + \frac{\beta}{b}\omega q^{-1}} \epsilon_k \\
&= \frac{1 - \left(1 - \omega\right) q^{-1} - \omega q^{-1}}{1 - \left(1 - \omega\right) q^{-1} + \frac{\beta}{b}\omega q^{-1} - \omega q^{-1}} r_k - \frac{1 - \theta q^{-1}}{1 - q^{-1} + \frac{\beta}{b}\omega q^{-1}} \epsilon_k \\
e_k &= \frac{1 - q^{-1}}{1 - q^{-1} + \frac{\beta}{b}\omega q^{-1}} r_k - \frac{1 - \theta q^{-1}}{1 - q^{-1} + \frac{\beta}{b}\omega q^{-1}} \epsilon_k
\end{aligned}
$$

Bibliography

[1] Alexander Q. Bleakie. *Integrated performance prediction and quality control in manufacturing systems*. PhD thesis, The University of Texas at Austin, 2014.

[2] D Zhen, H L Zhao, F Gu, and A D Ball. Phase-compensation-based dynamic time warping for fault diagnosis using the motor current signal. *Measurement Science and Technology*, 23(5):55601, 2012.

[3] Meinard Müller. *Information retrieval for music and motion*. 2007.

[4] Yonggang Liu, Robert H Weisberg, and Ruoying He. Sea Surface Temperature Patterns on the West Florida Shelf Using Growing Hierarchical Self-Organizing Maps. *Journal of Atmospheric and Oceanic Technology*, 23(2):325–338, February 2006.

[5] S Joe Qin. Survey on data-driven industrial process monitoring and diagnosis. *Annual Reviews in Control*, 36:220–234, 2012.

[6] Thomas F Edgar, Stephanie W Butler, W Jarrett Campbell, Carlos Pfei, Christopher Bode, Sung Bo Hwang, K S Balakrishnan, and J Hahn. Automatic control in microelectronics manufacturing :. *Automatica*, 36:1567–1603, 2000.

[7] Dekong Zeng, Yajing Tan, and Costas J. Spanos. Dimensionality reduction methods in virtual metrology. *Proceedings of SPIE*, 6922:692238–692238–11, 2008.

[8] Fan-tien Cheng, Senior Member, Hsien-cheng Huang, and Chi-an Kao. Dual-Phase Virtual Metrology Scheme. 20(4):566–571, 2007.

[9] AA Khan, JR Moyne, and DM Tilbury. An approach for factory-wide control utilizing virtual metrology. *..., IEEE Transactions on*, 20(4):364–375, 2007.

[10] Shane A Lynn, John Ringwood, Senior Member, and Niall Macgearailt. Global and Local Virtual Metrology Models for a Plasma Etch Process. *IEEE Transactions on Semiconductor Manufacturing*, 25(1):94–103, 2012.

[11] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. Data-driven Soft Sensors in the process industry. *Computers & Chemical Engineering*, 33(4):795–814, April 2009.

[12] Gill B. Development of Virtual Metrology in Semiconductor Manufacturing. *Semiconductor Manufacturing*, 2011.

[13] Dekong Zeng and Costas J Spanos. Virtual Metrology Modeling for Plasma Etch. *IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING*, 22(4):419–431, 2009.

[14] Shane Lynn and John Ringwood. Virtual metrology for plasma etch using tool variables. In *Advanced Semiconductor Manufacturing Conference*, number 1, pages 143–148, 2009.

[15] Alexander Bleakie and Dragan Djurdjanovic. Feature extraction, condition monitoring, and fault modeling in semiconductor manufacturing systems. *Computers in Industry*, 64(3):203–213, April 2013.

[16] Bao Lin, Bodil Recke, Jørgen K.H. Knudsen, and Sten Bay Jø rgensen. A systematic approach for soft sensor development. *Computers & Chemical Engineering*, 31(5-6):419–425, May 2007.

[17] Petr Kadlec and Bogdan Gabrys. Local learning-based adaptive soft sensor for catalyst activation prediction. *AIChE Journal*, 57(5):1288–1301, May 2011.

[18] Koichi Fujiwara, Manabu Kano, Shinji Hasebe and Akitoshi Takinami. Soft-Sensor Development Using Correlation-Based Just-in-Time Modeling. *AIChE Journal*, 55(7):1754–1765, 2009.

[19] Hector J. Galicia, Q. Peter He, and Jin Wang. A reduced order soft sensor approach and its application to a continuous digester. *Journal of Process Control*, 21(4):489–500, April 2011.

[20] Bhupinder S. Dayal and John F. MacGregor. Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *Journal of Process Control*, 7(3):169–179, 1997.

[21] Mark Nixon Shu Xu, Bo Lu, Michael Baldea, Thomas Edgar, Willy Wojsznis, Terrence Blevins. Data cleaning in the process industry. *Reviews in Chemical Engineering*, 2015.

[22] Barry M Wise, Neal B Gallagher, E Igenvector R Esearch, and I Nc. Multivariate Modeling of Batch Processes Using Summary Variables.

[23] Bo Lu, Ivan Castillo, Leo Chiang, and Thomas F. Edgar. Industrial PLS model variable selection using moving window variable importance in projection. *Chemometrics and Intelligent Laboratory Systems*, 135:90–109, July 2014.

[24] Gang Li, Baosheng Liu, S Joe Qin, and Donghua Zhou. Quality relevant data-driven modeling and monitoring of multivariate dynamic processes: the dynamic T-PLS approach. *IEEE Transactions on Neural Networks*, 22(12):2262–71, December 2011.

[25] Randolf Reiss, Willy Wojsznis, and Robert Wojewodka. Partial least squares confidence interval calculation for industrial end-of-batch quality prediction. *Chemometrics and Intelligent Laboratory Systems*, 100(2):75–82, February 2010.

[26] Ricardo Dunia, Thomas F. Edgar, Terry Blevins, and Willy Wojsznis. Multistate analytics for continuous processes. *Journal of Process Control*, 22:1445–1456, 2012.

[27] S Joe Qin. Recursive PLS algorithms for adaptive data modeling. *Computers & Chemical Engineering*, 22(4):503–514, 1998.

[28] Shane a. Lynn, Niall MacGearailt, and John V. Ringwood. Real-time virtual metrology and control for plasma etch. *Journal of Process Control*, 22(4):666–676, April 2012.

[29] Salvador Garc, Theodora Kourti, and John F Macgregor. Troubleshooting of an Industrial Batch Process Using Multivariate. *Industrial & Engineering Chemistry Research*, (42):3592–3601, 2003.

[30] Zdravko Stefanov and Leo Chiang. Application of multivariate batch data analysis for troubleshooting of end-point product quality. *American Control Conference (ACC), ...*, pages 1946–1951, 2011.

[31] Theodora Kourti. Multivariate dynamic data modeling for analysis and statistical process control of batch processes, start-ups and grade transitions. *Journal of Chemometrics*, 17(1):93–109, January 2003.

[32] A J Toprac, D J Downey, and S Gupta. Run-to-run control process for controlling critical dimensions, 1999.

[33] C.a. Bode, B.S. Ko, and T.F. Edgar. Run-to-run control and performance monitoring of overlay in semiconductor manufacturing. *Control Engineering Practice*, 12(7):893–900, July 2004.

[34] CA Andersson and Rasmus Bro. The N-way Toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, pages 3–6, 2000.

[35] Anna De Juan and Romà Tauler. Comparison of three-way resolution methods for non-trilinear chemical data sets. *Journal of Chemometrics*, 15:749–772, 2001.

[36] Svante Wold and Michael Sjostrom. PLS-regression : a basic tool of chemometrics. pages 109–130, 2001.

[37] Athanassios Kassidas, John F Macgregor, and Paul A Taylor. Synchronization of Batch Trajectories Using Dynamic Time Warping. *AIChE Journal*, 44(4):864–875, 1998.

[38] Niels-Peter Vest Nielsen, Jens Michael Carstensen, and Jø rn Smedsgaard. Aligning of single and multiple wavelength chromatographic profiles for chemometric data analysis using correlation optimised warping. *Journal of Chromatography A*, 805(1-2):17–35, May 1998.

[39] Wei Jiang, Zhi-Min Zhang, YongHuan Yun, De-Jian Zhan, Yi-Bao Zheng, Yi-Zeng Liang, Zhen Yu Yang, and Ling Yu. Comparisons of Five Algorithms for Chromatogram Alignment. *Chromatographia*, 76(17-18):1067–1078, July 2013.

[40] Giorgio Tomasi, Frans Van Den Berg, and Claus Andersson. Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *Journal of Chemometrics*, 18:231–241, 2004.

[41] Yang Zhang, Bo Lu, and TF Edgar. Batch Trajectory Synchronization with Robust Derivative Dynamic Time Warping. *Industrial & Engineering Chemistry Research*, 2013.

[42] E J Keogh and M J Pazzani. Derivative Dynamic Time Warping. In *Proceedings of the 1st SIAM International Conference on Data Mining*, pages 1–11, 2001.

[43] Abraham Savitzsky and Marcel J. E. Golay. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical chemistry*, 36:1627–1639, 1964.

[44] L H Chiang, R D Braatz, and E L Russell. *Fault Detection and Diagnosis in Industrial Systems*. Advanced Textbooks in Control and Signal Processing. Springer Verlag, 2001.

[45] R Dunia, S J Qin, T F Edgar, and T J McAvoy. Identification of faulty sensors using principal component analysis. *AIChE Journal*, 42:2797–2812, 1996.

[46] James V Kresta, John F Macgregor, and Thomas E Marlin. Multivariate statistical monitoring of process operating performance. *The Canadian Journal of Chemical Engineering*, 69:35–47, 1991.

[47] H Henry Yue and S Joe Qin. Reconstruction-Based Fault Identification Using a Combined Index. *Industrial & Engineering Chemistry Research*, 40(20):4403–4414, 2001.

[48] C. M. Andersen and R. Bro. Variable selection in regression-a tutorial. *Journal of Chemometrics*, 24(11-12):728–737, November 2010.

[49] Tahir Mehmood, Kristian Hovde Liland, Lars Snipen, and Solve Sæ bø. A review of variable selection methods in Partial Least Squares Regression. *Chemometrics and Intelligent Laboratory Systems*, 118:62–69, August 2012.

[50] Jon M JM Sutter and John H JH Kalivas. Comparison of forward selection, backward elimination, and generalized simulated annealing for variable selection. *Microchemical journal*, 47:60–66, 1993.

[51] Yvan Saeys, Iñaki Inza, Pedro Larrañaga, and Pedro Larran. A review of feature selection techniques in bioinformatics. *Bioinformatics (Oxford, England)*, 23(19):2507–2517, October 2007.

[52] A Höskuldsson and Agnar Hoskuldsson. PLS Regression Methods. *Journal of Chemometrics*, 2:211–228, 1988.

[53] Sijmen de Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, (18):251–263, 1992.

[54] Drazen Sliskovic. Methods for Plant Data-Based Process Modeling in Soft-Sensor Development. *HAYATI Journal of Biosciences*, 52(4):306–318, 2011.

[55] Agnar Hoskuldsson. Variable and subset selection in PLS regression. *Chemometrics and Intelligent Laboratory Systems*, 55:23–38, 2001.

[56] L. Norgaard, A. Saudland, J. Wagner, J. P. Nielsen, L. Munck, and S. B. Engelsen. Interval Partial Least-Squares Regression (iPLS): A Comparative Chemometric Study with an Example from Near-Infrared Spectroscopy, 2000.

[57] Ildiko E. Frank. Intermediate least squares regression method. *Chemometrics and Intelligent Laboratory Systems*, 1(3):233–242, July 1987.

[58] Juan Antonio Fernández Pierna, Ouissam Abbas, Vincent Baeten, and Pierre Dardenne. A Backward Variable Selection method for PLS regression (BVSPLS). *Analytica Chimica Acta*, 642:89–93, 2009.

[59] V Centner, D L Massart, O E de Noord, S de Jong, B M Vandeginste, and C Sterna. Elimination of uninformative variables for multivariate calibration. *Analytical chemistry*, 68(21):3851–8, November 1996.

[60] Hong-Dong Li, Mao-Mao Zeng, Bin-Bin Tan, Yi-Zeng Liang, Qing-Song Xu, and Dong-Sheng Cao. Recipe for revealing informative metabolites based on model population analysis. *Metabolomics*, 6(3):353–361, May 2010.

[61] Riccardo Leardi. Application of genetic algorithmPLS for feature selection in spectral data sets. *Journal of Chemometrics*, (March):643–655, 2000.

[62] Riccardo Leardi, Mary Beth Seasholtz, Randy J. Pell, and Mary Beth. Variable selection for multivariate calibration using a genetic algorithm: prediction of additive concentrations in polymer films from Fourier transform-infrared spectral data. *Analytica Chimica Acta*, 461(2):189–200, June 2002.

[63] Leo H. Chiang and Randy J. Pell. Genetic algorithms combined with discriminant analysis for key variable identification. *Journal of Process Control*, 14(2):143–155, March 2004.

[64] Hongdong Li, Yizeng Liang, Qingsong Xu, and Dongsheng Cao. Key wavelengths screening using competitive adaptive reweighted sampling method for multivariate calibration. *Analytica chimica acta*, 648(1):77–84, August 2009.

[65] Hyonho Chun and Sündüz Kele. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 72(1):3–25, January 2010.

[66] M Forina, C Casolino, C Pizarro Millan, and C Pizarro Millan. Iterative predictor weighting (IPW) PLS: a technique for the elimination of useless predictors in regression problems. *Journal of . . .*, 184(November 1998):165–184, 1999.

[67] Ulf Indahl. A twist to partial least squares regression. *Journal of Chemometrics*, 19(1):32–44, January 2005.

[68] Fredrik Lindgren, Paul Geladi, Stefan Rannar, and Svante Wold. Interactive Variable Selection (IVS) For PLS. Part 1: Theory and Algorithms. *Journal of Chemometrics*, 8:349–363, 1994.

[69] L Eriksson and Umetrics AB. *Multi- and Megavariate Data Analysis*. Number p. 1. Umetrics AB, 2006.

[70] Cajo J F ter Braak and Sijmen de Jong. The objective function of partial least squares regression. *Journal of chemometrics*, 12(1):41–54, 1998.

[71] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (...*, 58(1):267–288, 1996.

[72] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 1974.

[73] K. P. Burnham. Multimodel Inference: Understanding AIC and BIC in Model Selection. *Sociological Methods & Research*, 33(2):261–304, November 2004.

[74] J Edward Jackson. *A User's Guide to Principal Components*, volume 43 of *Wiley Series in Probability and Statistics*. Wiley-Interscience, 1991.

[75] Gang Li, S. Joe Qin, and Donghua Zhou. Output Relevant Fault Reconstruction and Fault Subspace Extraction in Total Projection to Latent Structures Models. *Industrial & Engineering Chemistry Research*, 49(19):9175–9183, October 2010.

[76] A Justel, D Pena, and R C Af C Duplicateinen C N P Zamar. A multivariate Kolmogorov-Smirnov test of goodness of fit. *StatisticsProbabilityLetters*, 35:251–259 ST – A multivariate Kolmogorov–Smirnov te, 1997.

[77] Petr Kadlec, Ratko Grbić, and Bogdan Gabrys. Review of adaptation mechanisms for data-driven soft sensors. *Computers & Chemical Engineering*, 35(1):1–24, January 2011.

[78] Kristian Helland, Hans E Berntsen, Odd S Borgen, and Harald Martens. Recursive algorithm for partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 14(1-3):129–137, 1992.

[79] Weihua Li, H.Henry Yue, Sergio Valle-Cervantes, and S.Joe Qin. Recursive PCA for adaptive process monitoring. *Journal of Process Control*, 10(5):471–486, October 2000.

[80] Aftab a. Khan, J.R. Moyne, and D.M. Tilbury. Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares. *Journal of Process Control*, 18(10):961–974, December 2008.

[81] Bo Lu, John Stuber, and Thomas F. Edgar. Integrated Online Virtual Metrology and Fault Detection in Plasma Etch Tools. *Industrial & Engineering Chemistry Research*, 53(13):5172–5181, April 2014.

[82] Jianbo Liu, Dragan Djurdjanovic, Kenneth Marko, and Jun Ni. Growing Structure Multiple Model Systems for Anomaly Detection and Fault Diagnosis. *Journal of Dynamic Systems, Measurement, and Control*, 131(5):051001, 2009.

[83] Gregory Allan Cherry. Methods for Improving the Reliability of Semiconductor Fault Detection and Diagnosis with Principal Component Analysis. *Ph.D Thesis*, 2006.

[84] Svante Wold, Paul Geladi, Kim Esbensen, and Jerker Öhman. Multi-way principal components-and PLS-analysis. *Journal of Chemometrics*, 1(1):41–56, 1987.

[85] Paul Nomikos and John F. MacGregor. Multi-way partial least squares in monitoring batch processes. *Chemometrics and Intelligent Laboratory Systems*, 30(1):97–108, November 1995.

[86] Martin Andersson. A comparison of nine PLS1 algorithms. *Journal of Chemometrics*, 23(10):518–529, October 2009.

[87] N Krämer and M Sugiyama. The degrees of freedom of partial least squares regression. *Journal of the American Statistical ...*, 106(494):697–705, 2011.

[88] S J Qin and T J McAvoy. Nonlinear {PLS} modeling using neural networks. *Computers & Chemical Engineering*, 16(4):379–391, 1992.

[89] Roman Rosipal and N Krämer. Overview and recent advances in partial least squares. *Subspace, Latent Structure and Feature Selection*, pages 34–51, 2006.

[90] Donghua Zhou, Gang Li, and SJ Qin. Total projection to latent structures for process monitoring. *AIChE Journal*, 56(1), 2010.

[91] SJ Qin and Yingying Zheng. Quality-relevant and process-relevant fault monitoring with concurrent projection to latent structures. *AIChE Journal*, 59(2), 2013.

[92] Jie Yu and S Joe Qin. Multimode Process Monitoring with Bayesian Inference-Based Finite Gaussian Mixture Models. *AIChE Journal*, 54(7):1811–1829, 2008.

[93] Jialin Liu. On-line soft sensor for polyethylene process with multiple production grades. *Control Engineering Practice*, 15(7):769–778, July 2007.

[94] Jianbo Liu, Dragan Djurdjanovic, Kenneth a. Marko, and Jun Ni. A divide and conquer approach to anomaly detection, localization and diagnosis. *Mechanical Systems and Signal Processing*, 23(8):2488–2499, November 2009.

[95] Lennart Eriksson, Johan Trygg, and Svante Wold. PLS-trees(R), a top-down clustering approach. *Journal of Chemometrics*, 23(11):569–580, November 2009.

[96] U. Thissen, H. Swierenga, a. P. de Weijer, R. Wehrens, W. J. Melssen, and L. M. C. Buydens. Multivariate statistical process control using mixture modelling. *Journal of Chemometrics*, 19(1):23–31, January 2005.

[97] Jianbo Yu. Fault Detection Using Principal Components-Based Gaussian Mixture Model for Semiconductor. *IEEE Transactions on Semiconductor Manufacturing*, 24:432–444, 2011.

[98] Chang Kyoo Yoo, Kris Villez, In-Beum Lee, Christian Rosén, and Peter A Vanrolleghem. Multi-model statistical process monitoring and diagnosis of a sequencing batch reactor. *Biotechnology and bioengineering*, 96(4):687–701, 2007.

[99] Chunhui Zhao, Yuan Yao, Furong Gao, and Fuli Wang. Statistical analysis and online monitoring for multimode processes with between-mode transitions. *Chemical Engineering Science*, 65(22):5961–5975, 2010.

[100] Shi Jian Zhao, Jie Zhang, and Yong Mao Xu. Monitoring of processes with multiple operating modes through multiple principle component analysis models. *Industrial & engineering chemistry research*, 43(22):7025–7035, 2004.

[101] Teuvo Kohonen. Self-organized formation of topologically correct feature maps, 1982.

[102] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, 3rd edition*. 2009.

[103] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255, 1993.

[104] Evan L. Russell, Leo H. Chiang, and Richard D. Braatz. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 51(1):81–93, May 2000.

[105] Yingwei Zhang. Independent Component Analysis (KICA) and Support Vector Machine (SVM). *Industrial & Engineering Chemistry Research*, 47:6961–6971, 2008.

[106] James Plummer, Michael Deal, and Griffin Peter. *Silicon VSLI Technology: Fundamentals, Practice and Modeling*. Prentice Hall, 2000.

[107] James P McVittie, Juan C Rey, A J Bariya, M M IslamRaja, L Y Cheng, S Ravi, and Krishna C Saraswat. SPEEDIE: a profile simulator for etching and deposition, 1991.

[108] Synopsys. Sentaurus Topography, 2015.

[109] Silvaco. SILVACO Athena Advanced Process Simulation Framework.

[110] Thomas J Harris. Assessment of control loop performance. *The Canadian Journal of Chemical Engineering*, 67(5):856–861, 1989.

[111] S. Joe Qin, Gregory Cherry, Richard Good, Jin Wang, and Christopher a. Harrison. Semiconductor manufacturing process control and monitoring: A fab-wide framework. *Journal of Process Control*, 16(3):179–191, March 2006.

[112] T J Harris, C T Seppala, and L D Desborough. A review of performance monitoring and assessment techniques for univariate and multivariate control systems. *Journal of Process Control*, 9(1):1–17, 1999.

[113] Mohieddine Jelali. An overview of control performance assessment technology and industrial applications. *Control Engineering Practice*, 14:441–466, 2006.

[114] Liang Chen, Mingda Ma, Shi-Shang Jang, David Shan-Hill Wang, and Shuqing Wang. Performance assessment of run-to-run control in semiconductor manufacturing based on IMC framework. *International Journal of Production Research*, 47(15):4173–4199, August 2009.

[115] RP Good and SJ Qin. On the stability of MIMO EWMA run-to-run controllers with metrology delay. *... Manufacturing, IEEE Transactions on*, 19(1):78–86, 2006.

[116] Byung-Su Ko and Thomas F. Edgar. PID control performance assessment: The single-loop case. *AIChE Journal*, 50(6):1211–1218, June 2004.

[117] Amogh V Prabhu and Thomas F Edgar. Performance Assessment of Run-to-Run EWMA Controllers. *IEEE Transactions on Semiconductor Manufacturing*, 20(4):381–385, 2007.

[118] Ma MD, Chang CC, Wong DSH, and Jang SS. Identification of tool and product effects in a mixed product and parallel tool environment. *Journal of Process Control*, 19(4):591–603, 2009.

[119] Jin Wang. Properties of EWMA Controllers With Gain Adaptation. *IEEE Transactions on Semiconductor Manufacturing*, 23(2):159–167, May 2010.

[120] Xiaojing Jiang. *Control Performance Assessment of Run-to-run Control System Used in High-mix Semiconductor Manufacturing*. PhD thesis, 2012.

[121] Christopher Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[122] J Edward Jackson and Govind S Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349, 1979.

[123] Arthur E Albert and Arthur Albert. *Regression and the Moore-Penrose pseudoinverse*, volume 3. Academic Press New York, 1972.

[124] S. Joe Qin. Statistical process monitoring: basics and beyond. *Journal of Chemometrics*, 17(8-9):480–502, August 2003.

[125] Klaus Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[126] Theodora Kourti, Paul Nomikos, and JF MacGregor. Analysis, monitoring and fault diagnosis of batch processes using multiblock and multiway PLS. *Journal of Process Control*, 5(4):277–284, August 1995.

[127] Rasmus Bro. Multiway calibration. Multilinear PLS. *Journal of Chemometrics*, 10(March 1995):47–61, 1996.

[128] Nicolaas Klaas, M Faber, and Rasmus Bro. Standard error of prediction for multiway PLS 1 . Background and a simulation study. *Chemometrics and Intelligent Laboratory Systems*, 61:133–149, 2002.

[129] Jie Yu and S. Joe Qin. Multiway Gaussian Mixture Model Based Multiphase Batch Process Monitoring. *Industrial & Engineering Chemistry Research*, 48(18):8585–8594, September 2009.

[130] Leo H. Chiang, Riccardo Leardi, Randy J. Pell, and Mary Beth Seasholtz. Industrial experiences with multivariate statistical analysis of batch process data. *Chemometrics and Intelligent Laboratory Systems*, 81(2):109–119, April 2006.

[131] Age Smilde, Rasmus Bro, and Paul Geladi. Some Properties of Three-Way Component Models. In *Multi-Way Analysis with Applications in the Chemical Sciences*, pages 89–109. 2005.

[132] Inmaculada García, Luis Sarabia, M. Cruz Ortiz, and J. Manuel Aldama. Three-way models and detection capability of a gas chromatography-mass spectrometry method for the determination of clenbuterol in several biological matrices: The 2002/657/EC European Decision. *Analytica Chimica Acta*, 515:55–63, 2004.

[133] Zdravko I. Stefanov and Karlene A. Hoo. Hierarchical multivariate analysis of cockle phenomena. *Journal of Chemometrics*, 17:550–568, 2003.

[134] P. M. Kroonenberg and J. M F ten Berge. The equivalence of Tucker3 and Parafac models with two components. *Chemometrics and Intelligent Laboratory Systems*, 106:21–26, 2011.

Vita

Bo Lu was born in Henan, China. His family moved to Canada in 2001, where he attended high school and then post-secondary education. He received his Bachelor of Science in Chemical Engineering from the University of Alberta in 2011. During this time, he partook three co-op internships with Dow Chemical, Syncrude Inc., and Suncor Energy Inc. in control engineering and reservoir engineering positions. After obtaining his undergraduate degree, Bo applied and was admitted for graduate studies in Chemical Engineering at the University of Texas at Austin. He began his graduate studies in August 2011 under the supervision of Dr. Thomas F. Edgar. He has accepted a R&D position with Dow Chemical in Freeport, TX after graduation.

Permanent Email address: bo.lu@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.