The Dissertation Committee for Kevin Andrew Chamness
certifies that this is the approved version of the following dissertation:

Multivariate Fault Detection and Visualization in the Semiconductor Industry

Committee:

Thomas Edgar, Supervisor

S. Joe Qin

Grant Willson

Gyeong Hwang

Glenn Masada

Multivariate Fault Detection and Visualization in the Semiconductor Industry

by

Kevin Andrew Chamness, 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

December 2006

Dedicated to Sarah and to my Mom and Dad.

For always being there for me and motivating me to finally put all my thoughts down and finishing.

Acknowledgments

I wish to thank the multitudes of people who helped me complete my research at the University of Texas and the multitude of companies that supported it.

First, I would like to thank Dr. Edgar for his guidance, his flexibility for letting me take my own path with my research, his infinite patience, and his advice on all my work.

I would also like to thank the fellow graduate students Juergen, Tyler, and Sebastien, who early in my doctoral career helped me understand what graduate school was about and how everyone's path is different.

I would like to thank Greg, Rick, and Elaine, also fellow graduate students and dear friends, who continued to push me to continue my work and my dissertation. A very special thank you goes out to Greg, who is a coworker at AMD as well and worked on very similar work with his RPCA development. He provided great insights, a sounding board for ideas, and he built the simulation toolbox that I was able to later integrate my work into. Without this infrastructure, my degree would have taken even longer to complete. Another special thank you goes to Rick who reminded me for three years that the best dissertations are the completed ones.

I would like to thank all the people at Tokyo Electron (TEL), Advanced

Micro Devices (AMD), and Spansion who helped me along the way and to the companies themselves for supporting me during my research. TEL gave me the time and flexibility to learn multivariate analysis, apply it on etch process data, and explore its applications. In addition, at TEL I learned a lot about software development and the necessity for efficient user interfaces to make FDC systems usable. At AMD I was given the tools and resources to integrate existing FDC techniques with electrical test data and to test out the FDC methods and my vision for visualization. In addition, I was given time and encouragement to develop the KNN methods to supplement the already existing RPCA technology. A very special thank you goes to Ernest of Spansion, who really understood my needs to get the dissertation done and gave me the flexibility I needed to complete it.

Finally, I would like to thank Sarah, my fiancé, and my parents for helping me and motivating me to complete my goal of getting through the doctoral program.

Multivariate Fault Detection and Visualization in the Semiconductor Industry

Publication No. _____

Kevin Andrew Chamness, Ph.D.
The University of Texas at Austin, 2006

Supervisor: Thomas Edgar

The semiconductor industry provides vast opportunities for process monitoring and multivariate fault detection. Most of the multivariate methods currently used in the industry are statistically-based techniques. These methods are also extended to monitor batch processes such as the process tools used in semiconductor manufacturing.

In this dissertation, the existing statistical fault detection methodologies are discussed and compared to non-parametric modeling techniques for multivariate outlier detection. Inspired by these non-parametric modeling techniques, a new k Nearest Neighbor (KNN) multivariate fault detection method is proposed to augment the existing statistical methods. In this technique, instead of pre-computing a model, only a window of historic reference data is retained. The fault detection performance metric used in this algorithm provides universal scaling and confidence limits for the overall metric value,

the block contributions, and individual variable contributions. It also has the flexibility to be tuned for local or global sensitivity when multiple populations are present within the reference data.

This new KNN method also is extended to monitor batch processes. Two applications of the KNN method are created by simply unfolding the batch data or by selecting only reference data similar in batch time for each individual trace sample. Both KNN batch methods are compared against other existing batch methods to detect induced faults using a plasma etch experiment. The trace sample method performs among the best of all investigated batch techniques.

This dissertation also introduces additional methods for monitoring systems with multivariate models. A complete software architecture is presented for reporting and visualization of multivariate results. This method takes advantage of block and variable contributions to guide users to the process variables with the most extreme and most frequent excursions. This system is applied to monitor final wafer electrical test data. In addition, methods are presented which assist the monitoring of drifting processes. A simple technique to recursively adapt the centering and scaling coefficients of a principal component analysis (PCA) model is presented. Movement metrics are also introduced to monitor the changes in these coefficients over time. These movement metrics allow visibility into the process changes which caused the model to adapt.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Fault detection

In the chemical process industries (and most any manufacturing environment), monitoring an ever growing number of processes and instruments is an ongoing challenge. Many methods have been developed for monitoring individual variables and for simultaneous multivariate analysis. In general, the first step of process monitoring is detecting an excursion. Then, more sophisticated monitoring systems can be used to identify the root cause of the excursion, estimate the excursion, and possibly reconstruct excursion-free sensor values.

This field of fault detection varies from application to application based on how well-known the process is to be monitored, the amount of effort that is available to model and monitor a system, the risk involved in the system itself, and the time required to make calculations between process samples. Many methods have been developed and are applicable to a variety of systems.

The most basic statistical process control (SPC) systems monitor a single process variable and signal an alarm when statistically-based limits are exceeded. The models for these systems are simple to compute and are based

on only a mean and standard deviation of each variable within a historical period. From this model, limits are computed based on probability levels. Engineering time and effort are required to maintain limits for all the individual variables to be monitored.

Multivariate techniques have been developed to take historic data including a number of process variables and look at the normal range of each process variable and also the relationship between the variables. The general hypothesis assumed in these methods is that a historical data set defines a characteristic region of the n-dimensional space that represents normal operation. Any new data outside this region is considered an outlier and hence is characterized as an excursion.

There are many statistical modeling techniques to capture this normal space and most involve using the correlation matrix within the data set or principal component analysis (PCA) to capture the most important directions within the correlation among the variables. With this statistical model, PCA can determine both an estimated probability that the model can accurately predict the new data point and that the point is within the normal operating space of the model. The statistical models are precalculated based on a known set of reference data, and as new data become available, the performance statistics can quickly be calculated based on the model parameters, without requiring a historical reference data set. These statistical methods have also been extended to monitor batch processes such as processing tools in the semiconductor industry.

Other existing techniques are non-parametric as apposed to the existing parametric model-driven techniques, and focus on simply detecting outliers within the reference data set. Instead of estimating a set of model parameters based on modeling assumptions of the reference data, the data itself are retained as the model. When a new point is collected, a fault metric is determined by clustering of the data, kernels, or by nearest neighbor analysis. These calculations are generally more expensive than ones based on a pre-computed model, but they are more flexible and potentially easier to maintain, only requiring the retained historic data.

After every new sample, the results of the fault detection metric and any contributions to the metric are captured (regardless of the method used to generate them). When excursions occur, automated responses can be set up to quickly notify operators and shut down abnormally operating systems. In addition, the results from these techniques are made available for root cause analysis by operators and engineers to quickly diagnose the problem, make corrective action, and get the faulting equipment operational again. The availability of the analysis results in an easy to navigate and informative medium is a key to making any monitoring system usable and effective.

1.2 Microelectronic fabrication introduction

This work focuses on the application of fault detection techniques to the semiconductor manufacturing industry, but by no means does this prevent the application of these methods and techniques to other industries or prob-

lems. The semiconductor industry does pose a special set of problems where fault detection can be readily applied. The semiconductor industry is one of the most capital-intensive industries compared to revenue and only recently has attention been focused on control, optimization, and system-wide fault detection required for competitive operation of state of the art factories[84].

The manufacturing of integrated circuits involves the creation of several layers of specifically patterned films on the surface of a silicon wafer. Each silicon wafer is divided into a large number of functional die. Specific electrical characteristics are achieved in each die by chemically altering the different films and layers and by the specific layout of the patterns. These films and layers are created by generally over a hundred batch operation steps. These steps include various operations of crystal growth, oxidation, depositions of dielectrics, silicon, and metals, physical and chemical vapor deposition, dopant diffusion, dopant ion implantation, photolithography, etch, and chemical-mechanical polishing[33]. Once the wafer is complete (after usually a month or more of processing), it undergoes many testing steps. Usually the measurement of electrical parameters on test structures occurs when the final layer of metal is attached to the wafer. At this stage all the underlying circuits can be verified indirectly using test structures built into the wafer, and the quality of the entire process can be monitored.

Each unit operation in the factory takes place in an isolated process tool. Each tool, if it is properly equipped, can report all the data from onboard sensors during or after each process run. As device technologies shrink and

the size of the wafers increase, a growing number of software applications are being created in the industry to monitor this process data[84].

Many statistical methods exist for building multivariate models which could be applied to final wafer electrical test data and for building batch models for analyzing process tool data. This dissertation presents a new method that improves on existing non-parametric techniques that can be applied for analyzing both types of data sets. In addition, a method for communicating and navigating the results of any multivariate monitoring system is presented.

1.3 Dissertation outline

The next chapter contains a review of the current literature on fault detection techniques, visualization of these techniques, and different applications of these techniques to semiconductor tool trace data. This chapter includes a discussion on both statistic approaches for static and batch processes and non-parametric approaches for anomaly detection.

Chapter 3 discusses a new k-nearest neighbors (KNN) multivariate fault detection method attempting to bridge the non-parametric approaches with a similar statistical performance metric. This chapter provides results of many test simulations examining the properties of the method.

Chapter 4 focuses on a generic technique for organizing multivariate analysis results and presenting them in a way that provides quick investigations into fault analysis regardless of the multivariate method. This chapter

discusses the reports and charting necessary to navigate through the monitoring results and provides the results of this system implemented to monitor the final wafer electrical tests of a semiconductor factory.

Chapter 5 focuses on application of multivariate analysis on semiconductor process tools. The first half of the chapter discusses techniques for monitoring drifting processes and investigates a new adaptation method with simulations on summary data from an etch process tool. The second half of the chapter discusses two different applications of the KNN algorithm to batch processes and provides a comparison of these methods to existing batch methods based on an industrial etch experiment with induced faults from the literature.

In the final chapter a summary of the work is presented along with recommendations for future work.

Chapter 2

Fault monitoring background

The field of process monitoring is essentially the study of methods to make decisions from data sets. The first step in all monitoring is fault detection, indicating whether or not a process excursion has occurred. After an excursion is identified, many other techniques can be used to classify the fault, identify the fault, estimate the fault magnitude, and even reconstruct the fault-less process data. A survey of the full field of statistical process monitoring can be found in [19, 66, 83, 110]. Obviously, as more analysis is performed on the faults, the more complicated these methods become and the more a priori knowledge of the system is required.

The focus of this work is simply on the fault detection step, generally applied to semiconductor manufacturing. In this industry specifically, the manufacturing process and products change rapidly on the scale of months, and each processing tool sees a high mix of manufacturing operations, products, and tool states. All monitoring applications have to be as automatic as possible and adapt to the changing conditions. Operators and engineers are typically already burdened with maintaining process operations and do not have much time for maintaining the monitoring system itself. The addition of

new monitoring tools should help operators and engineers understand where exceptions are occurring within the factory, but without adding the burden of large sets of limits to maintain and charts to monitor. The goal of this work is to provide additional methods for fault detection, provide a system to communicate and effectively navigate results, and provide results from semiconductor applications.

2.1 Univariate fault detection

The simplest case of process monitoring is monitoring a single variable and sampling its population by taking measurements from the manufacturing process. From the historic samples, an estimate of parameters describing the population can be made. The more data that are collected the better the population can be estimated but at a higher cost to sample the data. When a new piece or set of data arrives, it is compared against the known population and a decision is made whether or not the new data contains enough proof that the process is no longer operating satisfactorily. Statistically speaking, that is to say there is the null hypothesis that the process is good and this is assumed to be true until proven otherwise. The new piece of data is evaluated to attempt to prove the null hypothesis false. To make this decision the confidence in the population is evaluated with respect to the new data, and a risk must be taken in classifying the new data. If the process has not really changed and it is identified as a fault (a "false positive"), wasted time and effort could be used troubleshooting or needlessly shutting down a process tool. If the process

has really changed but it is not identified as a fault, then the risk is the tool will continue to operate and create product off specification.

Based on this simple premise, Statistical Process Control (SPC) has been successfully used for monitoring univariate data particularly of product quality variables, but also continuous process variables. Originally introduced by Shewhart[95], SPC operates by estimating the mean and variation of an individual variable based on data from a period of time when the process is known to be operating free from faults. Any change in the statistical nature indicates an excursion which should be investigated. Many rule sets have been developed for identifying conditions that have a small probability of occurring naturally in a normal distribution, and the most prevalent is the Western Electric rules[106]. These rules indicate the process is out of control if:

- a single data point is outside the mean plus or minus three standard deviations,
- two out of three consecutive points are outside the mean plus or minus two standard deviations,
- four out of five consecutive data points are outside the mean plus or minus one standard deviation, or
- eight consecutive points are on one side of the center line.

Any of these events have a small probability of occurring randomly in a normal distribution, so when they occur, notifications can be automatically created

in the monitoring system. More detailed explanations of SPC are included in [75].

2.2 Multivariate fault detection

To improve upon univariate fault detection, methods exist that consider a set of process variables simultaneously and detect not only if the process variable goes outside its normal range, but also if the historical correlation between the process variables has been violated. When process variables are highly correlated, the values can simultaneously change within the normal univariate limits but still break from the normal correlation[66]. In addition, these methods have multivariate performance metrics that can be monitored in lieu of all the individual single variable control charts, as discussed below.

2.2.1 Mahalanobis distance and Hotelling's T^2

The Mahalanobis distance was introduced by Mahalanobis in 1936 (and discussed in [68]) as a metric to detect similarity between a new vector and a sample set. Given a vector of quality measurements $x = (x_1, x_2, x_3, ..., x_p)$, the vector of means from the sample set $\mu = (\mu_1, \mu_2, \mu_3, ...\mu_p)$ and the covariance matrix Σ of the sample set, the Mahalanobis distance can be given as:

$$D_M = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}. \tag{2.1}$$

An upper control limit can be constructed for the D_M metric based on a central Chi-squared distribution with p degrees of freedom[66]. The $\chi^2_\alpha(q)$

can be calculated with α level of significance for performing the test. This test works well with a small number of process variables, but can have rank-deficient Σ matrices when the process variables are highly correlated. This formulation also matches the Hotelling's T^2 statistic with an estimate of the sample variance used in place of the correlation matrix Σ.

2.2.2 Principal component analysis

The use of principal component analysis (PCA) forms the basis of multivariate data analysis and has shown great utility as a Multivariate Statistical Process Control (MSPC) tool when samples can be considered independent observations. PCA as described by Jackson[52] and Wold *et al.*[117] was introduced by Pearson[81] in 1901 and then developed by Hotelling[50] in 1933[54]. PCA is the method of reducing a large multivariate data set into a set of latent variables that captures most of the variation within the data set. This allows the estimation of a covariance matrix based on the principal directions while removing the influence of noisy directions.

To compute the PCA analysis according to Qin[83] and many others, for a new sample vector of m sensors, let $x \in \Re^m$ denote the sample vector. Assuming there are N samples for each sensor, a data matrix $X \in \Re^{N \times m}$ is composed with each row representing a sample. For correlation-based PCA the matrix X is scaled to zero-mean and unit variance, and for covariance-based PCA the matrix X is simply scaled to zero-mean. The matrix X can be decomposed into a score matrix T and a loading matrix P, whose columns

are the right singular vectors of X,

$$X = TP^T + \tilde{X} = TP^T + \tilde{T}\tilde{P}^T = [T \;\; \tilde{T}][P \;\; \tilde{P}]^T, \tag{2.2}$$

where $\tilde{X} = \tilde{T}\tilde{P}^T$ is the residual matrix. Since the columns of T are orthogonal, the covariance matrix is

$$\Sigma \approx \frac{1}{N-1} X^T X = [P \;\; \tilde{P}]\Lambda[P \;\; \tilde{P}]^T \tag{2.3}$$

where

$$\Lambda = \frac{1}{N-1} [T \;\; \tilde{T}]^T[T \;\; \tilde{T}] = \text{diag}\{\lambda_1, \lambda_2, ..., \lambda_m\} \tag{2.4}$$

$$\lambda_i = \frac{1}{N-1} t_i^T t_i = \text{var}\{t_i\} \tag{2.5}$$

and t_i is the i^{th} column of T and λ_i are the eigenvalues of the covariance matrix in descending order. For variance-scaled X, Equation 2.3 gives the correlation matrix R.

Once the model is generated, a sample vector x can be projected on the principal component space (S_p) and the residual space (S_r) as,

$$\hat{x} = PP^T x \in S_p \tag{2.6}$$

$$\tilde{x} = \tilde{P}\tilde{P}^T x = (I - PP^T)x \in S_r. \tag{2.7}$$

Since (S_p) and (S_r) are orthogonal,

$$\hat{x}^T \tilde{x} = 0 \tag{2.8}$$

and

$$\hat{x} + \tilde{x} = x. \tag{2.9}$$

In order to produce the optimal PCA model for a given data set, the general goal is to choose the number or principal components such that $\hat{x}$ (the PC space) contains mostly information and $\tilde{x}$ (the residual space) contains mostly noise. In order to calculate the model given a certain number of principal components, either the NIPALS[117] algorithm or singular value decomposition (SVD) is used. Many different algorithms are used to choose the optimal number of principal components. Wold's work on cross validation[115] was one of the first methods appearing in 1978. Recently Valle *et al.*[102] and Qin and Dunia's work[85] suggest the variance of reconstruction method that includes an optimization function that guarantees the global minimum for determining the number of principal components. A complete discussion of methods for principal component selection can be found in [103].

2.2.2.1 PCA fault detection metrics

For process monitoring, historically two metrics are calculated for a new sample vector x, the squared prediction error (SPE) and the Hotelling's T^2. The SPE statistic measures the projection of the sample vector on the residual space, indicating how well the sample conforms to the model,

$$SPE = \|\tilde{x}\|^2 = \|(I - PP^T)x\|^2. \tag{2.10}$$

The Hotelling's T^2 measures the variation in each sample within the principal component space as the sum of the normalized squared scores,

$$T^2 = t_i \Lambda^{-1} t_i^T = x_i P \Lambda^{-1} P^T x_i^T. \tag{2.11}$$

In both of these equations the loadings matrix P and the eigenvalues Λ are the only first l values, where l is the number of principal components chosen in the model.

Using these metrics, an excursion is hypothesized when either

$$SPE \geq \delta^2 \tag{2.12}$$

or

$$T^2 \geq \chi^2_\alpha(l) \equiv \tau^2 \tag{2.13}$$

where δ^2 and τ^2 are the control limits for the SPE and T^2 metrics, given a $1-\alpha$ confidence level[17]. The SPE control limit was derived by Jackson and Mudholkar[53] assuming that x follows a multivariate normal distribution. Given the same assumptions, T^2 follows a χ^2 distribution with l degrees of freedom[121]. A full formulation for the limits can be found in [22].

A third process monitoring metric with its associated limit was introduced by Yue and Qin[121]. This metric combines the two above metrics into a combined index weighting each metric against its respective limit as

$$\varphi = \frac{SPE(x)}{\delta^2} + \frac{T^2(x)}{\chi^2_\alpha(l)} = x^T \Phi x \tag{2.14}$$

where

$$\Phi = \frac{I - PP^T}{\delta^2} + \frac{P\Lambda^{-1}P^T}{\chi^2_\alpha(l)}. \tag{2.15}$$

The limit for this metric takes the form of

$$\varphi \geq g\chi^2_\alpha(h) \equiv \zeta^2 \tag{2.16}$$

where g and h are estimated by Yue and Qin based on φ being a quadratic function of the x vector and Φ being positive-definite.

2.2.2.2 Automatic adaptation techniques

Multivariate analysis using PCA over a long period of time requires a stable process. In practice, especially in the semiconductor manufacturing industry, the tools are relatively stable but drift over the maintenance cycle of the tool, giving limited life to a single static model. A number of attempts have been made to adapt PCA models to slowly changing systems, while maintaining a valid model.

Wold discussed exponentially weighted moving principal component analysis in 1993. He established a solution for the model updates that balanced a number of competing objectives for the model[116]; the model

- should be a good summary of the local process data,
- should be stable against unwarranted rotation,
- should have an optional long term memory, and
- should be robust against spikes.

His solution is mathematically complicated and iterative but consists of a weighted average between the present model and some golden reference data (reference data that is known to be normal). The mean, standard deviations,

and correlation update have contributions from the historical data and the current model.

Gallagher *et al.*[38] studied monitoring a semiconductor etch process with varying methods for adaptation. They investigated simply moving the means of the centering step for PCA and not changing the standard deviations or correlation structure. They also investigated using an exponentially weighted moving covariance where the covariance matrix was adapted element by element with an EWMA filter. They found that although the adaptation methods performed better than the static model, neither case created a robust model that could last through preventative maintenance, additional cleaning, new equipment installs, or extended tool drifts.

Li *et al.*[64] proposed the first recursive algorithms for PCA model updated based simply on the old vector of means, vector of standard deviations, the correlation matrix, and the new sample vector of data. They included a discussion of how to update the number of principal components and confidence limits recursively as well. This algorithm has been implemented by Cherry[16]. Cherry's implementation of recursive PCA is used throughout this dissertation when RPCA is mentioned.

2.2.2.3 Robust monitoring issues

One issue of high importance is how to create robust models and how to keep the model robust when incomplete or corrupt data is encountered. The initial steps for building a PCA model require that an estimate of the

mean and standard deviation for each process variable within the reference data. Chiang *et al.*[22] discusses robust methods for estimating non-biased centering and scaling factors for the PCA model based on medians and other methods less sensitive to outliers. They also introduce a robust calculation of the correlation matrix. These methods allow a more robust model to be created to initialize the monitoring system even when outliers exist in training data.

Another concern for a robust monitoring system using PCA is how the performance metrics are calculated when a full vector of sample data is not available. When monitoring univariate data, when a sensor fails or a variable is not available, only the single SPC chart is affected. For multivariate monitoring in the case of a missing sensor value, the monitor must continue to operate and if possible give an estimate of the missing value for the sensor based on the model. The first proposed method of data reconstruction based on simultaneous multiple components was proposed by Cleason[23]. The first PCA-based method introduced by Wold for reconstruction used a single principal component[114]. Wise and Ricker[111], Nelson *et el*[77], and Dunia *et al.*[32] all present methods that are similar for data and fault reconstruction. All these methods are able to estimate a missing variable in the sample vector. A more recent work by Nelson *et al.*[76] also gives a method for estimating the confidence region for the reconstructed variable.

2.2.2.4 Nonlinear PCA methods

Another area of research to improve the application of PCA to real processes is the incorporation of nonlinear components into the model. PCA assumes Gaussian distributions, normal density estimates, and that the key directions within the reference data are linear throughout the whole reference data space. Methods have been introduced that relax these assumptions and attempt to build models that allow more complicated data sets to be monitored with PCA techniques.

The details of most nonlinear methods can be found in [28]. Initial methods by Oja[80] and others that followed used Hebbian networks to compute principal components with unsupervised networks with nonlinear activation functions. These networks did not have a geometrical interpretation to extend the PCA monitoring metric concepts to these new components. This was followed by autoassociative neural networks, which use a reduced number of hidden layer nodes in the neural network to predict the output values from the input values. This method works to reduce the number of dimensions in the data but leads to very complex nonlinear optimization problems that can lead to local optimum[61]. Further work looked at generating principal curves (or surfaces) with iterative calculations to capture the structure of the data. These curves are made principal components by mapping each point in the reference data to the closest point on the curve and finding the curve where each point on the curve is the average of all the data points projected onto it[29, 48]. One limitation of this approach is that the number of principal com-

ponents has to be specified before creating the nonlinear model. The other main difference between these nonlinear methods and PCA is that in PCA the principal components have a decreasing amount of information, while in these nonlinear methods the amount of information seemed to be evenly distributed between the components[18].

Later methods of nonlinear PCA involved the use of kernel functions to give nonlinear estimates of the dot product internal to the PCA calculation to create a kernel principal component analysis (KPCA) model. In this method all properties of the PCA model still apply in the mapped nonlinear space and do not necessarily require an expensive neural optimization[94]. Another nonlinear extension for PCA is the use of radial basis functions (RBF) to create principal manifolds for process monitoring[44, 109]. A detailed description of RBF can be found in [10].

Most recently, attempts have been made to improve the neural network mapping to predict orthogonal nonlinear principal components, and then to add a linear principal component model step. Improvements to the last step of the neural network have been added to insure that the output of the neural network has orthogonal components, prior to the linear PCA model step[71].

Many nonlinear methods have been successfully applied for monitoring real industrial processes. Chen *et al.*[14] used Kernel PCA and Yoo *et al.*[120] used adaptive fuzzy nonlinear multivariate analysis to monitor waste water plants. Martin *et al.*[70] used similar Kernel PCA to monitor a batch methyl methacrylate polymerization reactor. Anomaly detection with RBF neural

nets has been used successfully in Iraq to detect faults in the swashplate ball bearings of Apache helicopters[9].

2.2.3 Non-parametric approaches

To this point in this chapter, the concept of fault detection has taken the form of a preconstructed statistical model built from historical data, then each new sample is compared to the model and its limits. In this section, the concept of fault detection simply answers the question: is a new sample point an outlier given a history of known good reference data? Outlier detection in multivariate data sets has a long history and is not covered in detail in this document. Historically outlier detection has mostly been a binary property. Most outlier detection algorithms were not designed for outlier detection in itself, but to remove outliers only to make another method more robust such as in classification. In some cases there have been methods that generate a metric that characterize a degree to which data is an outlier and in a few cases a limit representing a confidence limit for the data.

Hawkins[49] defines an outlier as "an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" which is exactly the goal of statistical-based fault detection. A review of multivariate outlier detection can be found in [4]. Many different outlier tests have been created based on distributions, whether the distribution parameters are known, the number of expected outliers, and the expected types of outliers. Most of the outlier tests are simply univariate and most

are statistical in nature. The most used outlier test for multivariate data is the Mahalanobis distance (as described in Section 2.2.1) based on covariance measures[37, 41].

Markou and Singh have done an extensive review of statistical approaches for novelty detection[69]. The detection of novelties in the field of machine learning is the identification of new or unknown data or signals that the system was not made aware of during training. This class of methods has been used successfully for fault detection in industrial applications. These included statistical parametric methods and nonparametric classification, clustering, and nearest neighbor techniques.

2.2.3.1 Classification and clustering using distance metrics

Statistical classification is the procedure in which new samples are placed into groups based on a training set of previously labeled reference data. A historical review of classification can be found in [25]. Fisher discriminant analysis is a two-way classifier that attempts to find the linear combination of features which best separates two classes. Support vector machines (SVM) also search for a decision surface that best separates two classes but they optimize the margin between the two classes[104]. The Bayes naive classifier is a probability-based training method that weighs the mixed probabilities of each class[74]. Neural nets can also be trained to predict the classification vector[51]. Finally, k-nearest neighbor classification classifies a sample point as the class which is most common among its k nearest neighbors[27, 31].

In all these approaches, the attribute used to classify the data is included in the training reference data. When the reference data is not pre-classified, an unsupervised classification algorithm must be used first to cluster the data and assign classes to each point. A number of methods can be used for clustering[5]. By far the most popular is k-means clustering developed by Hartigan and Wong[47]. In this method, k clusters are specified each with a corresponding centroid, assuming each point belongs to the nearest centroid. Iterations calculate the centroids and the centroid membership until the algorithm converges. Improvements to this approach have been developed which help the computations of the algorithm and have different weights on the feature space[36].

2.2.3.2 Anomaly detection using clustering

Algorithms that identify anomalies have been generated from unsupervised clustering algorithms. The problem of anomaly detection is more difficult than classification. In classification only boundaries between different populations are constructed which extend infinitely in space. In anomaly detection methods separate populated space from anomalous space. Portnoy[82] introduced a method of outlier detection using unlabeled data based on clustering to detect network intrusions in computer networks. This method uses a k means clustering technique to build a model of normal reference data of network use. New incoming data is classified based on the nearest cluster using a Euclidean distance. If the point is farther from the nearest centroid than

the cluster width of that centroid (the farthest distance between two points within the centroid's cluster), then the point is considered an outlier.

Eskin *et al.*[34] introduce two methods for anomaly detection using classification and clustering. The first method simply counts the number of reference data points N within a sphere of radius w from the sample point. If N is below a certain threshold, then the sample point is considered an outlier (given a certain width w). The second method computes the sum of the distances from the sample point to its k nearest neighbors. This algorithm is made computationally efficient for a fixed reference data set using canopy clustering. The third method Eskin describes from [93] first maps the reference data into a second feature space with a radial basis kernel function and then uses support vector machines to isolate a region of space near the reference data from the origin. This is done by simultaneously maximizing the distance from a separation hyperplane to the origin while penalizing any reference point not separated from the origin by the hyperplane.

Chan *et al.*[13] introduced their clustering for anomaly detection (CLAD) algorithm that relates to k nearest neighbor algorithms. It attempts to identify local as well as global outliers. In this method, clusters are considered to have a fixed radius and clusters are allowed to overlap. This method suggests the fixed radius by sampling 1% of the reference data and calculating the pair-wise distance between the points, then taking the average of the shortest 1% of the pair-wise distances as the cluster width. Since each cluster is of fixed radius, the density of a cluster is determined by the number of points within

a cluster. The distance to other clusters for a particular cluster (inner cluster distance, ICD) is calculated as the average of the distance to all other clusters. An outlier is considered a cluster that is both distant and sparse. The cluster is considered distant (and a global outlier) if its ICD is more than a standard deviation away from the average ICD. The cluster is considered sparse (and a local outlier) if the count of points within the cluster is more than one median absolute deviation (MAD)[43] smaller than the median count.

2.2.3.3 Anomaly detection metrics

In a number of algorithms, the detection of an outlier is a binary decision[3, 59, 90]. In this section a number of anomaly detection criteria and anomaly metrics are introduced.

Knorr and Ng formally define an outlier as an object O in a data set T is a $DB(p, D)$-outlier (a distance-based outlier given p and D) if at least a fraction p of the objects in T lies greater than distance D from O[58]. They show this is consistent with Hawkins' definition of an outlier, and this generalization is consistent with statistical definitions such as the normal distribution with three standard deviations with $p_o = 0.9988$ and $D_o = 0.13\sigma$. This formulation represents a global method for outlier detection. They present two algorithms that attempt to evaluate this metric on a data set (with reference database buffering for large data sets) that stop searching each point once the p fraction of points have been found.

Ramaswamy *et al.*[87] investigate the problem finding up to n outliers

within a reference data set. They characterize outliers by ranking each point within the reference data by its distance to its k^{th} nearest neighbor (using any distance metric). The largest n points on the list are considered outliers. Their algorithm efficiently partitions the large input space and prunes partitions that cannot contain the top outliers.

Breunig *et al.*[6] introduce what they claim is the first metric for the degree to which a point is local outlier. The local outlier factor (LOF) uses a local perspective and locates outliers with respect to the density in the neighboring region of the reference data. This work functions similarly to the algorithm generated in Chapter 3 so it is described in detail. First the k-distance of an object p in the reference data set, $d_k(p)$, is defined as the distance from p to its k^{th} nearest neighbor. The k-distance neighborhood of an object p, $dn_k(p)$, is defined as all the reference data points within the k-distance of p (and hence are defined as the k-nearest neighbors of p). The third definition is the reachability distance of object p with respect to object o, $rd_k(p, o)$, as the maximum of the k-distance or the actual distance between p and o. This reachability distance allows points that are very close to the reference point to be considered at the k-distance apart for smoothing purposes. So given a tunable minimum number of points z, the local reachability density of p, $lrd_z(p)$, is defined as

$$lrd_z(p) = \left[\frac{\Sigma_{o \in dn_z(p)} rd_z(p, o)}{|dn_z(p)|} \right]^{-1}. \tag{2.17}$$

Finally, the local outlier factor of p is defined as

$$LOF_z(p) = \frac{\Sigma_{o \in dn_z(p)} \frac{lrd_z(o)}{lrd_z(p)}}{|dn_z(p)|}. \tag{2.18}$$

The LOF determines the degree to which a point is an outlier. For most objects within a cluster of data the LOF is approximately 1. Given this metric, there is no specification on a generic upper bound on LOF for "fault detection", but there are examples of the metric on simulated data that give confidence in its ability to measure outliers. Breunig *et al.* also discuss rules for picking values of z that relate to the selecting a number of k nearest neighbors. They suggest a minimum value of 10 to remove statistical fluctuations, a value greater than the threshold of minimum points that define a cluster, and an upper bound of the number of "close by" objects that can potentially be local outliers.

Harmeling *et al.* recently proposed three metrics for outlier detection[45]. Their Kappa index is simply the distance from a point to its k^{th} nearest neighbor. Their Gamma index for a sample point is the average distance to its k nearest neighbors. Their Delta index is the length of the mean of vectors pointing from the sample point to its k nearest neighbors. This Delta index works well since an outlier generally is the same direction from its nearest neighbors (if all the neighbors lie within the same cluster). Rieck[89] presents another metric based on the above Gamma metric that attempts to remove the density dependence of the Gamma metric. The Gamma metric does not take on similar values for points in and around clusters of different density. Rieck's improvement is the Zeta metric which is the Gamma metric

value minus the average inner-clique distance of its neighbors:

$$\zeta_k(x) = \frac{1}{k}\Sigma_{i=1}^{k} d(x, nn_i(x)) - \frac{1}{k(k-1)}\Sigma_{i=1}^{k}\Sigma_{j=1}^{k} d(nn_i(x), nn_j(x)). \quad (2.19)$$

This metric performs well as a global or local method depending on the value of k selected, but there is no limits given for Zeta metric values to be considered an outlier.

2.3 Fault detection visualization

One of the critical pieces in a fault detection system is the communication of the excursions to engineers and operators. In some cases every excursion can be directly linked to shutting down a process line, process tool, or a process chamber. In most cases, though, there are far too many excursions to intervene directly with the process on every alarm. The goal is to alert engineers and operators and present all the information about the excursion as quickly and as completely as possible. In addition, the goal is to present trends and contributions to turn the fault detection results into meaningful information so that an engineer can take corrective action as soon as possible.

Many works have presented basic visualization suggestions for PCA-based fault detection techniques[15, 65, 72, 105, 107]. In most cases the first set of charts is a time series plot of each T^2 and SPE along with its respective limit. In the case of Yue and Qin[121], these are replaced with a single combined index with its corresponding limit. Some authors have made the suggestion of scaling the performance metrics to their limit so that a constant

value (usually unity) is used as the limit value for all charts[15, 72]. These charts function like SPC for the whole set of process variables included in the model. Like traditional Shewhart SPC charts, the T^2 and SPE can also be placed on cumulative sum (CUSUM) and exponentially weighted moving average (EWMA) control charts[66, 107].

Another method for understanding the principal components of a PCA model is to plot scatter plots of the various scores. Since the first two principal components capture the largest percentage of the information in a PCA model, it is common to plot a scatter of the first score value and the second score value. From this plot, outliers and clustering within the reference data can clearly be identified and a model can be refined and tuned. In addition to the scores, the loading vectors can be plotted on a scatter plot as well to visualize which input process variables dominate the principal components corresponding to the selected loadings. The individual scores can also be displayed on a time series chart to identify trends (possibly along with calculated confidence limits for the individual scores[79]).

When excursions occur in the performance metrics, information as to which input process variables contributed to the excursion are needed. Contributions are calculated for the T^2, the SPE, and the combined index. The calculation for contributions vary among many authors for a multitude of reasons which are beyond the scope of this summary[24, 72, 86, 105] and are even patented by Hopkins *et al.*[92]. The contribution calculations used for PCA throughout this work are based on the work of Cherry *et al.*[15]. Using any

contribution method, a bar chart for each performance metric can be created for an individual sample. Each process variable bar represents the contribution to the performance metric from the corresponding input process variable. These contribution charts are simple to calculate but do not necessary point to the variables that are the root cause of the fault. They simply indicate those variables that are inconsistent with the normal operating condition. Miller *et al.* suggest referring to the univariate SPC charts with limits to quickly inspect the major contributors[72].

A major improvement on contribution plots was the addition of the block contributions[16, 65, 86]. To compute block contributions, only a subset of the process variables is used to calculate the performance metrics. This gives an indication if any of the subset of process variables has a univariate fault or there is a break in correlation between the subset of process variables. The block contributions allow a knowledgeable engineer to break the process variables into logical groups by function or process subsystem. Multiple blocks can be used to indicate subsystems where the excursions may occur. These blocks values can help reduce the number of bars in a contribution plot to an understandable number for systems with a large number of process variables.

Beyond time series plots of performance metrics, scores and loading plot, contribution and block contribution plots, and univariate time series plots, few other concepts exist in the literature for visualization of faults. Chiang and Braatz [20] introduced the modified distance and modified causal dependency for continuous process monitoring when a causal map is available.

Their work focuses on the correlation of causal steps by examining the paired contributions between pairs of variables. They use the paired contributions (similar to block contributions) to assign causal relationships between process variables that are indicating faults and those that are the root cause to the fault. The set of excursions and pairing are used to indicate on a control panel the variables in alarm and the probable causes. Examples of this work are given for the Tennessee Eastman Process[30]. Another interesting visualization method was the presentation of fault specific control charts introduced by Goodlin *et al.*[39]. This work used Fisher linear discriminants to pick fault directions that discriminated samples from the particular fault classes from the normal process and from other fault classes. The normal process data is then projected onto these fault directions and the results are used to create a control chart for the particular fault class.

Even with all the existing multivariate fault detection visualization techniques, improvements are still needed to communicate excursions to operators and engineers and help convince them that there is a problem. As more analysis methods are added, care must be taken not to add additional false alarms which cause wasted time investigating normal processes. As more signals are added for operators and engineers to analyze, additional verification tests must be placed at their fingertips. All methods and calculation steps must be convincing and available in one place.

2.4 Algorithms for batch process analysis

A number of algorithms have been developed to extend principal component analysis to monitor batch processes. In the semiconductor industry these methods have proved very useful for monitoring the wealth of trace data from process tools. In typical PCA process monitoring, data is typically available in a two-dimensional matrix where process variables are represented in columns and time-based samples are represented as rows. In batch analysis, each batch is a particular run in a piece of equipment and has its own matrix of process variables and time-based samples. In general batch processes are represented by a three-dimensional array ($I \times J \times K$), where I is the number of batches, J is the number of process variables, and K is the number of time-based observations made of each variable during a batch. This data array is then decomposed differently in a number of methods.

2.4.1 Multi-way PCA

Multi-way principal component analysis was introduced by Wold *et al.*[118] and similarly introduced by Nomikos and MacGregor[78, 79], and has been used extensively[110]. In this method the batch array is decomposed into a two-dimensional matrix by slicing the array in the time dimension and setting each matrix of batches (I) by variables (J) side by side for each time sample (K). This decomposition maintains a row in the data matrix for each batch and creates a column for each combination of variable and time sample (creating a I by $J * K$ matrix). This matrix can then be mean-centered and variance

scaled in order to perform PCA analysis. Centering the data is particularly meaningful because this results in subtracting the mean trajectory of each variable. The scaling of each column handles the differences in units between each measured variable and can be performed on each column individual or globally for each process variable. Once the data is unfolded, centered, and scaled, then traditional PCA analysis can be applied. This method allows the normal trajectory of the batch to be monitored along with the correlation between process variables and different time periods within the batch.

MPCA has the limitations that each batch is required to have the same number of time samples and that the results for a batch can not be analyzed until the batch is complete. Nomikos and Macgregor[79] discuss methods for online batch monitoring for calculating current results when the batch is being processed. Improvements to using MPCA without making assumptions about the rest of the batch are introduced by Rannar *et al.*[88]. Kosanovich *et al.* give a detailed description of MPCA and show its effectiveness for monitoring a batch reactor and improving process understanding [60].

2.4.2 Three-way models for batch data

Another set of methods for analyzing the array of batch data are three-way models. Smilde *et al.*[96, 99] present a thorough summary of these methods, although an introduction is presented here. A number of methods have been developed to improve upon the "unfolding" used in MPCA including parallel factors (PARAFAC) (also known as CANDECOMP) and Tucker mod-

els[11, 46, 97, 101, 123].

In the PARAFAC (and trilinear) model, each element of the data array is predicted by

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} + e_{ijk} \tag{2.20}$$

where a_{ir}, b_{jr}, and c_{kr} are the elements of the loading matrices A ($I \times R$), B ($J \times R$), and C ($K \times R$) respectively, e_{ijk} is the residual error, and R is the number of PARAFAC components. The differences between PARAFAC and trilinear decomposition (TLD) are simply in the way the models are determined. A complete method for constructing a PARAFAC model can be found in [7]. The original method for estimating a TLD model can be found in [91, 108] with a discussion on improving convergence in [73]. Kiers discusses methods for visualizing the results of three-way models[56].

Two extensions to PARAFAC have since been proposed to remove the constraint that batches must have the same number of samples as originally defined. These methods also allow monitoring of a batch while it is still in progress. Wold *et al.*[119] propose a two level model for batch analysis. The inner model simply models each time sample individually, but instead of simply building a PCA model on all the traces, a projection to latent structures model using partial least squares (PLS) is used to predict a batch "maturity metric". This maturity metric can simply be time or a process variable that is monotonically changing indicating the progression of the batch. This inner model is monitored while a batch is progressing at each time period. Once a batch is complete, a batch health metric is calculated based on the scores of

the inner model. This method is referred to as MPLS. The other extension to PARAFAC is PARAFAC2 as introduced by Kiers *et al.*[57] and Bro *et al.*[8]. In this method the K direction is given a different set of loadings for each different size of k. The set of outer loading matrices must be proportional and their matrix cross product must be a constant for every value of k in order to create a unique model. When analyzing the MSPC limits for PARAFAC2 the squared residuals are scaled by the number of samples to provide consistent results[113].

The other approaches to use three-way models are based on Tucker models. In the Tucker3 model, each element of the data array is predicted by

$$x_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} a_{ip} b_{jq} c_{kr} g_{pqr} + e_{ijk} \tag{2.21}$$

where a_{ip}, b_{jq}, and c_{kr} are the elements of the loading matrices A $(I \times P)$, B $(J \times Q)$, and C $(K \times R)$ respectively, g_{pqr} is an element of the array G $(P \times Q \times R)$, e_{ijk} is the residual error, and P, Q, and R are the number of Tucker components in the three modes. If only two modes are reduced, the Tucker model becomes the Tucker2 model and is predicted by

$$x_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} a_{ip} b_{jq} g_{pqk} + e_{ijk} \tag{2.22}$$

similar to the Tucker3 except the array G is now extended to $(P \times Q \times K)$. Similarly, other models could be created by instead reducing the first and third or the second and third modes and still result in a Tucker2 model. If

only the first mode is reduced, then a Tucker1 model can be created by

$$x_{ijk} = \sum_{p=1}^{P} a_{ip} g_{pjk} + e_{ijk} \tag{2.23}$$

where a_{ip} is the elements of the loading matrix A, g_{pjk} is an element of the further extended array G ($P \times J \times K$), and e_{ijk} is the residual error[99]. The Tucker1 model is identical to the multi-way PCA model (PCA with unfolding). Methods for building the various Tucker models can be found in [98].

Throughout the literature there are a number of comparisons of these different methods. To date there is no clear winner; they all seem to perform well depending on the application. One of the most complicated concerns using these multi-way models is the application of preprocessing steps. For batch processes centering or scaling across more than one direction simultaneously can destroy trilinear structure and multilinear structure[46]. In most cases centering is performed across the batch mode. This method essentially removes the average nonlinear batch trajectory for each process variable, focusing the three-way model on deviations from this mean trajectory[99]. Scaling is generally performed similarly in the batch mode, but the results of the trilinear methods are less sensitive to scaling choices.

2.4.3 Batch process analysis applications

A variety of industrial applications exist implementing these previously described batch process analysis methods. MPCA has been used successfully in a number of applications[67, 122].

Wise *et al.*[112] apply MPCA, PARAFAC, and TLD to the trace data from a plasma etch process for chamber monitoring. In this investigation PARAFAC appeared to work best, followed closely by both the normal PCA model on the step means and TLD. Wise *et al.* conclude that the single largest effect on the ability to detect faults is handling sensor drift. Wise *et al.*[113] in another work also apply the PARAFAC2 model to the same data set and find they gained the flexibility of handling batches of various size but see slightly decreased performance. This etch process and the results are described more in Chapter 5. These results are used as a benchmark for the new algorithms described in this dissertation.

Dahl *et al.*[26] compare MPCA and PARAFAC on a polymerization batch reactor and discuss a thorough investigation of the scores and loadings of the polymerization reactor. They found that PARAFAC was able to identify clearly three of the four main process zones in the clustering of the identified model loadings. They also found that both methods were able to find the increased variability in the beginning of the process, which agreed with the process operators' knowledge. In the end they recommended MPCA over PARAFAC for robust factory applications due to its handling of missing data and the ability to interpret less ambiguously the model information.

Chiang *et al.*[21] compare MPCA, MPLS, and Tucker3 analysis on a batch fermentation process. They found that overall all three were complementary to each other, and that persons who are trained and understand each technique should find all three methods useful. In their work MPCA seemed

more sensitive to overall batch variation, MPLS seemed more sensitive to the localized batch variation, and Tucker3 was a good balance of both. They did find the MPLS was most helpful at identifying bad batches during the model building process, but once models were built with all methods while excluding the rogue batches, all methods were able to clearly identify the excursions.

In very recent work Aguado *et al.*[1] applies a number of batch methods to a sequencing batch reactor for wastewater treatment. They compare MPCA, MPLS, and PCA applied samplewise to data that was unfolded, scaled, and centered like MPCA then stacked back to variable-wise unfolding. Their analysis looks at the methods both for offline monitoring of batches as a whole and also online monitoring of batches as they evolve. Overall they found that all three methods were able to describe the sequencing batch reactor, detect most of the faulty batches, and identify the responsible variables. However, for offline analysis, they preferred the MPCA approach because it was straightforward and slightly more consistent. They took some issues with the results of the online MPLS approach, especially with a single model for the whole process, although it had normal performance when the batch process was modeled with three phases. The final contribution of this work was the suggestion that online and offline monitoring should be coupled, and since this batch reactor was non-stationary, a moving window of good batches detected by batch analysis should be used to build adapting models for online realtime analysis.

2.4.4 Nonlinear extensions for batch analysis

All of the previously discussed algorithms for batch analysis have been linear. These algorithms generally center the data and remove the majority of the nonlinearities within the batch dynamics. Recently there have been a few applications to extend batch monitoring with nonlinear techniques. Lennox *et al.*[63] predicted that neural nets would be added to multi-way methods. They started with using a RBF neural net to predict the "maturity" metric during the evolution of a batch with success. The biomass concentration of an industrial fed-batch fermentation system was predicted.

Lee *et al.*[62] introduced multiway kernel principal component analysis (MKPCA) based on the kernel PCA methods discussed in Section 2.2.2.4. In this method, the batch array is treated identically to MPCA where the array is unfolded, the mean trajectory of each variable is removed, and each variable at each time is variance scaled. At this point, instead of calculating a normal PCA model, a KPCA model is constructed and the KPCA equivalents of T^2 and SPE are used for batch to batch monitoring. For within batch monitoring, Lee *et al.* uses the same methods suggested in the original MPCA work[78, 79], which include filling in the future values with zero, filling in the missing values with the current value, or using the ability of the PCA to predict the missing data subject to the PCA model and the known data. This algorithm was applied to a simulation of a fed-batch penicillin fermentation system and showed improved results for MKPCA over MPCA.

2.5 Existing software for fault detection

Many software packages have been developed to perform offline and online multivariate process monitoring. Most of these software packages have implemented the statistical MSPC and multiway batch methods. The Umetrics® company grew out of the work by Svante Wold, Rolf Carlsson, and others and now offers a commercially mature multivariate and batch analysis tools and online process monitoring in SIMCA-P+®[1]. The McMaster Advanced Control Consortium has put together a full monitoring system BatchSPC® based on the work of John MacGregor, Theodora Kourti, Paul Nomikos, and others[2]. The work of Barry Wise, Neal Gallagher, Rasmus Bro, Jeremy Shaver, and others is captured in the Eigenvector Research® PLS_Toolbox®[3], which is a very popular toolbox for MATLAB®. This toolbox has a very large array of multivariate analysis techniques including all the previously discussed multiway batch monitoring methods. The CAMO® company also offers a mature product for multivariate analysis called Unscrambler® which seems to have wide use in academia and industry[4]. Emerson Process Management® has integrated the MDC Technology® batch process monitor MSPC+ into their DeltaV® system for industrial monitoring[5].

Two industrial implementations of non-parametric modeling techniques

[1] www.umetrics.com
[2] macc.mcmaster.ca
[3] software.eigenvector.com/toolbox/pls_toolbox
[4] www.camo.com
[5] www.mdctech.com/products/mspc/mspc.htm

exist. Triant®'s product ModelWare® was one of the first industrial products to robustly monitor semiconductor process tools[6]. Their technique uses a method called universal process monitoring (UPM®), which is a nearest neighbor-based algorithm based on the work of Jack Mott[2]. In this method, the model simply maintains a reference set of historical process data. In order to evaluate a new point, the k nearest neighbors are located within the reference set. A weight is determined for each neighbor and the neighbors are linearly combined to determine prediction for the new point. Signal health metrics (contributions) and a system health metrics (overall performance) are calculated by comparing the residual of the prediction to normal ranges within the reference data.

The MASA Group® and STMicroelectronics® published an article introducing BlueKaizen® WaferFit® with advanced statistical learning concepts[35][7]. Their method has a simple hypothesis for novelty detection: using the learning set L, build a model able to decide whether a test point w is novel with respect to the elements of L. Their method uses a nonlinear abnormality score and calculates its distribution for all the data within the reference set. If a new test points has a metric outside a threshold of this distribution, then it is considered novel.

[6] www.triant.com
[7] www.bluekaizen.com

2.6 Conclusions

This chapter presents a summary of the literature for fault detection methodologies. The literature is dominated by statistical approaches which mostly utilize linear models. These methods are fairly mature and include all the attributes required for robust fault detection monitoring. They provide multivariate performance metrics with statistically generated limits, provide process variable contributions (and block contributions) also with statistically generated limits, handle missing data, and can provide updated models in most cases. In addition, nonlinear extensions exist for these methods, although generally the use of the nonlinear implementations in practice is limited due to the model complexity.

Non-parametric modeling methods for multivariate outlier or anomaly detection were also introduced. The multivariate fault detection problem described in the statistical-based literature is identical to the anomaly detection problem. The non-parametric techniques have the advantage of simply storing reference data and making less assumptions about the structure of the data. They function well even when data is nonlinearly distributed and can have local and global properties for outlier metrics. One shortcoming of the existing literature is generating generic limits on a non-parametric metric without processing the entire reference set and making sure that the same value of the metric is equally an outlier in all areas of space even with different local population densities. In the next chapter a k nearest neighbor algorithm is developed that attempts to solve the issues with generating limits and creat-

ing a metric with similar values around different population densities. This method is also extended to have all the robust monitoring properties discussed for statistical approaches.

Methods in the literature for visualization of the various MSPC methods were also discussed in this chapter. The current methods for MSPC visualization focus on a single model and the results of a single analysis. This background sets the framework for the integrated visualization architecture that is described in Chapter 4.

In most tool monitoring applications in the semiconductor industry, batch monitoring methods are used. The batch monitoring methods presented in this chapter help motivate the extension of the k nearest neighbor to tool trace data. In Chapter 5 the extension of the k nearest neighbor method to monitor batches are described and the results are compared to existing batch methods.

Chapter 3

k-Nearest Neighbor Fault Detection Algorithm Development

This chapter describes the development of a k-Nearest Neighbor (KNN) fault detection algorithm based on local distance distributions. The first section of this chapter gives the motivation and all the details of the algorithm. The second section provides examples showing results for a variety of different simulated data sets in order to validate the new algorithm and its properties.

3.1 Algorithm background

All fault detection algorithms have a hypothesis about the data set they are trying to analyze and a metric that gives an indication of an agreement with the hypothesis. In this method, the fundamental hypothesis is that the density distribution within the reference data set can be used to determine if a new data point is characteristic of the reference data set. Instead of characterizing model parameters from a selected training data set (such as in PCA), the method simply uses a set of samples from the training set as its model. For each decision about a new sample point, only the nearest region of the reference set to the sample point is analyzed, and a decision is made

about the sample point. This analysis makes no linearity assumptions about the data set and can have different detection properties for different regions of the reference data set.

In order to characterize the new sample point, the sample point population is created by calculating the distances from the point to its k-nearest neighbors within the reference set. This sample point distribution by itself can not give a sense if a point is representative of the data set. In order to characterize the local region of the data set, this same calculation of the distance from the point to its k-nearest neighbors is performed for each of the n nearest points (from the sample point) within the reference set. If there is a constant characteristic distribution from each point to its k-nearest neighbors, as n increases this distribution should not change significantly. All of the distances from n nearest points to their k-nearest neighbors are pooled into a single population to form the characteristic population. Finally, the distributions are compared to identify how characteristic the sample point is to the rest of the reference set.

To give a visual explanation to motivate this analysis, one thousand normally distributed points (with a mean of zero and a standard deviation of 2.5) were taken as a reference set. Then for various sample points from -10 to 10, the local characteristic population, the sample point population, and the estimated gamma probability distributions were generated. For this analysis, a k value of 32 and an n value of 334 was used. Figures 3.1 through 3.4 show the results for the sample points -8, -3, 0, and 3. In Figure 3.1 the sample

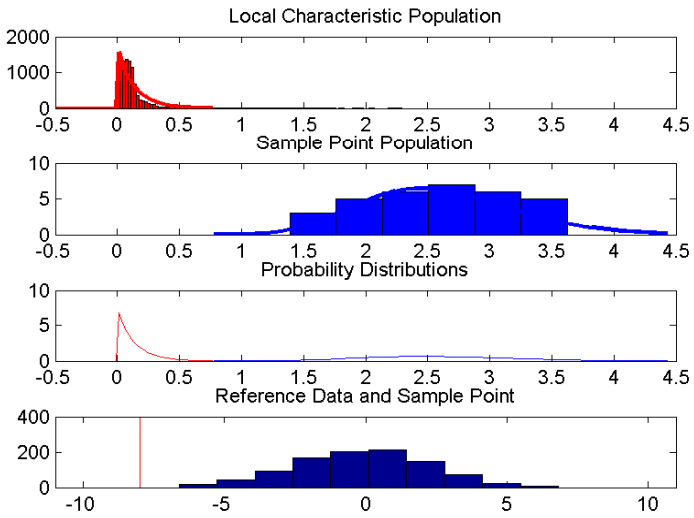


Figure 3.1: Characteristic and sample distributions from a normal data set with sample point at -8.

point is over 3 standard deviations from the mean of the data set and there is essentially no overlap of the distributions. As the sample point moves inside the reference population for the following 3 figures, the local characteristic population noticeably changes shape as the neighbors that are being used also become internal points of the reference population. Also, for all three values from -3 to 3 (just over one standard deviation from the mean), the sample point distribution has a high overlap with the local characteristic distribution. These figures show that the gamma distribution model does not perfectly fit these truncated populations, but it gives a good indication of the overlap of the populations.

From these observations on the four simulations, it is clear when the

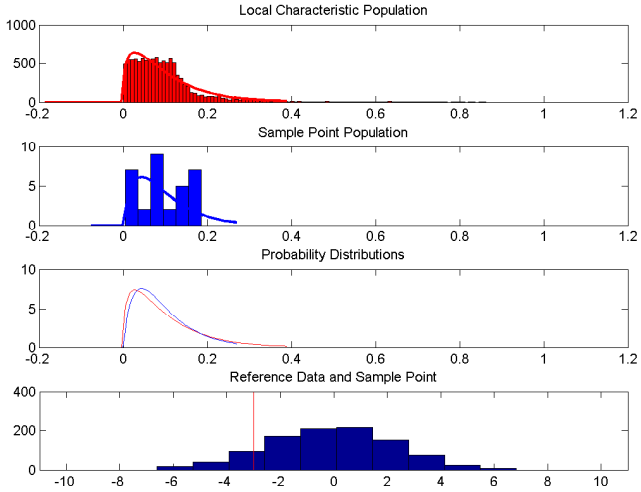


Figure 3.2: Characteristic and sample distributions from a normal data set with sample point at -3.

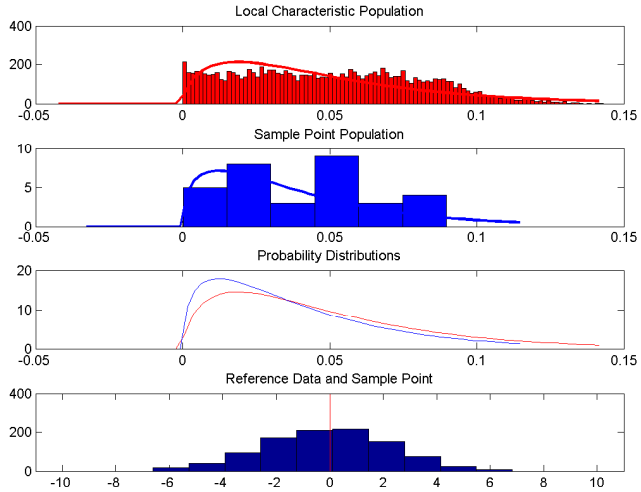


Figure 3.3: Characteristic and sample distributions from a normal data set with sample point at 0.

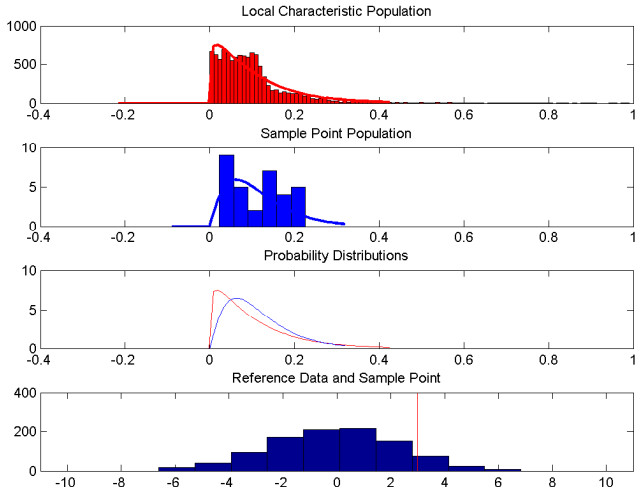


Figure 3.4: Characteristic and sample distributions from a normal data set with sample point at 3.

sample point lies far outside the reference set, there is little overlap between the two probability distributions. Also, when the sample point is internal to a population of the reference data, there is a much higher overlap in the probability distributions. The following subsections go through the details of fitting the distributions, determining where confidence limits should be placed to segregate data that is determined to be statistically normal, creating a performance index metric, and extending the method to multiple dimensions.

3.1.1 Modeling the distributions

In order to model the two populations in this work, the gamma distribution was used as described in [42]:

$$f(x;\eta,\lambda) = \begin{cases} \frac{\lambda^\eta}{\Gamma(\eta)} x^{\eta-1} e^{-\lambda x}, & x \geq 0, \quad \lambda > 0, \eta > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Fitting the populations to a statistical distribution provides smoothing of the population data and allows a continuous function to be integrated in the performance index metric calculation. The gamma distribution may not be technically appropriate for these two populations since they are truncated distributions (k is not the length of the entire data set), but this class of distributions fit adequately for a large range of cases. The gamma distribution allows for a changing shape and scaling through η and λ for a wide range of populations. In addition, it fits the long tails in these populations much better than approximating the populations with a characteristic mean and standard deviation based on a normal distribution and allows continuous parameterization as compared to the discrete number of degrees of freedom in the χ^2 distribution. This class of distributions was also chosen because it is appropriate in an extreme case as well. If we allowed the value of k to approach the full length of the reference set, then the characteristic distribution should approach a χ^2 distribution for normally distributed data, which is a subclass of the gamma distribution (when $\lambda = \frac{1}{2}$ and η is a multiple of $\frac{1}{2}$).

In order to fit the populations to the gamma distribution the method of maximum likelihood was used as implemented in the MATLAB® Statistics

Toolbox and described in [42] based on the original work by Greenwood and Durand[40].

3.1.2 Non-overlap probability calculation

Once these two distributions are characterized, they are used to evaluate how likely it is that the sample point population is representative of the local characteristic population. It was decided to calculate the non-overlay probability as the area above the characteristic probability distribution function and below the sample point probability distribution function starting from the 50th percentile of the characteristic probability distribution function to positive infinity. The non-overlap probability P_{no} is calculated as

$$\begin{aligned} &P_{no} = \textstyle\int_{\overline{C}}^{\infty} f(x)dx \\ &\text{where } f(x) = \begin{cases} S_{PDF}(x) - C_{PDF}(x), & S_{PDF}(x) > C_{PDF}(x) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.2}$$

where $\overline{C}$ is the distance corresponding to the 50th percentile of the characteristic cumulative distribution function, $S_{PDF}(x)$ is the sample point probability distribution function, and $C_{PDF}(x)$ is the characteristic probability distribution function. The 50th percentile of the characteristic CDF was used as a starting point so that having a large portion of the sample point PDF inside the average of the cumulative population was not penalized.

To illustrate this calculation, the non-overlap probability was calculated for a sample point at +7, -4, and 0 using the same reference data set used before in Figures 3.1 through 3.4 with 1000 normally distributed points with zero mean and a standard deviation of 2.5. The respective non-overlap probabilities

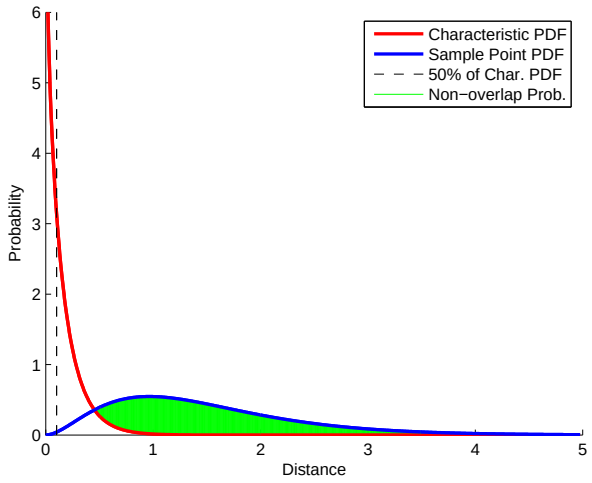


Figure 3.5: Non-overlap probability calculation region for a sample point at 7, $P_{no} = 0.8802$

were 0.8802, 0.1336 and 0.0998, and the calculations are illustrated in Figures 3.5, 3.6, and 3.7 respectively. The non-overlap probability will be asymptotic to 1.0 as the sample points moves away from the reference data and will have values closer to zero for the sample points near the densest regions of the reference data.

3.1.3 Threshold estimation based on a specified confidence limit

Unfortunately, the non-overlap probability values calculated in the previous section do not correspond to confidence limit values. For instance a value of 0.995 does not relate to the 99.5% confidence limit of the normal distribution that was used to generate the reference set. In order to find a relationship between the non-overlap probability values and the confidence limit, a sim-

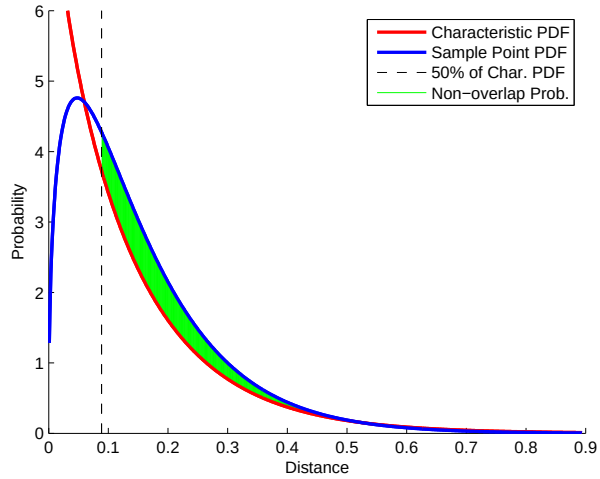


Figure 3.6: Non-overlap probability calculation region for a sample point at -4, $P_{no} = 0.1336$

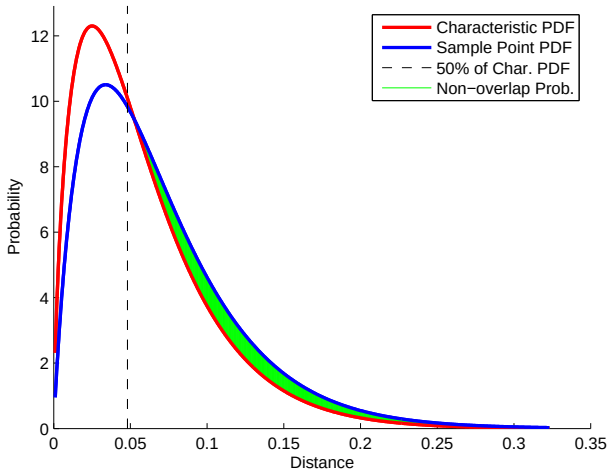


Figure 3.7: Non-overlap probability calculation region for a sample point at 0, $P_{no} = 0.0998$

ulation approach was used. There may exist an theoretical solution to this problem, but it is beyond the scope of this research and could be a topic of future research.

In order to find a relationship between the non-overlap probability threshold value and a given confidence limit, simulations were performed for a series of different reference data sizes (s), values of k, and values of n all using different data generated from the same normal distribution. Then the values of the non-overlap probability were calculated at the sample points corresponding to the different confidence limits (L values of 0.999, 0.997, 0.975, 0.9, 0.8, and 0.85) calculated from the normal distribution. All combinations of reference data of size (s) of 40, 100, 300, 800, 1500 entries, n values of 10, 20, 35, 66, 150, 500, and s, and k values of 10, 20, 33, 40, 60, 150, and 250 were used where $k < n < s$. It is worth pointing out that simulations with low values of s, k, and n provided very noisy results (as would be expected with the statistics involved with small sample sizes). Since this area is not where the algorithm will be used in the future, these values were not given as much weight in the fitting analysis.

The initial observation from the results (see simulation experiment data in Appendix A) reveal that the relationship does not seem to be based on the value of k. On the other hand there is a strong relationship between the threshold value and the ratio of n/s for each confidence limit. The final model that was generated to estimate the non-overlap probability threshold value

P_{crit} for a given n/s ratio and confidence limit is

$$\begin{aligned} P_{crit} &= 1 - ((1-B)e^{\frac{An}{s}} + B) \\ B &= 0.1617394 \log(1-L) + 1.1575 \\ A &= -0.685645(1-L)^{-0.612522} \end{aligned} \tag{3.3}$$

where L is the confidence limit (0.9975 for a 99.5% confidence limit). The fit of this model to the simulation data is show in Figure 3.8. Although there is significant scatter in these simulated data sets, the model form requires the threshold to go through 1 at a value of n/s equal to zero and then decay to an asymptote that varies by the confidence limit and decays at a rate that is a function of the confidence limit as well.

3.1.4 Scaling the results to form a performance index

Once this threshold value is determined, the non-overlap probability could be scaled by this threshold in order to determine a performance index for this method (similar to the combined index used in PCA[15] and the tool health used in Triant®'s ModelWare® package). Unfortunately, if the ratio of the probability and threshold values is used, the value of this ratio varies for different problem sizes after the critical value of 1. To relate this metric to the PCA combined metric, the goal is to have the same metric values regardless of the values of k, n, and s and also to have a value with a consistent meaning for different population densities. In order to satisfy these goals the performance index C was chosen to be the ratio of the χ^2 inverse of the non-overlap probability to the χ^2 inverse of the threshold with the number of

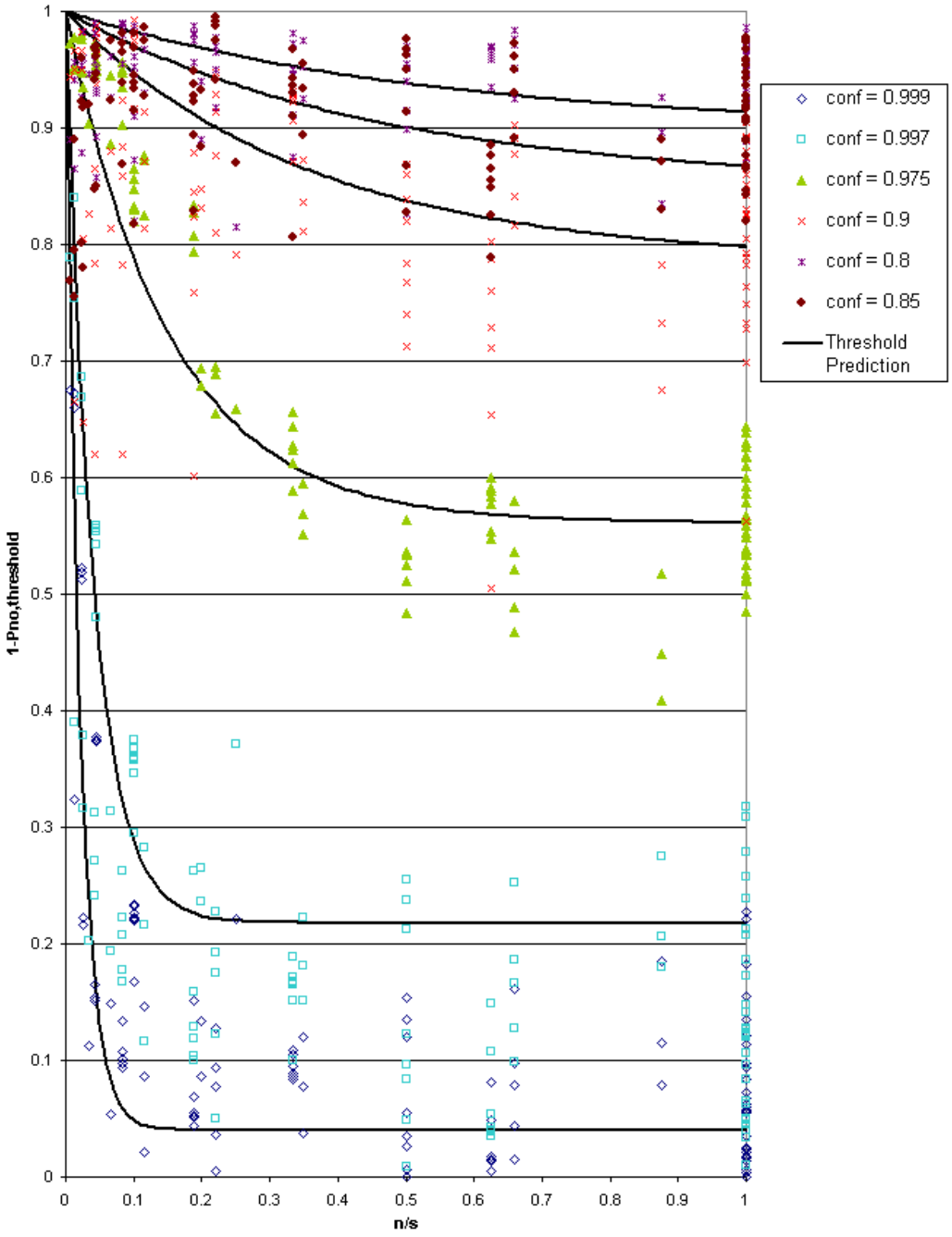


Figure 3.8: 1 minus non-overlap probability versus n/s ratio for various confidence limits with model fit to simulation experiments

degrees of freedom ν tuned for the problem size.

$$C = \frac{F^{-}1(P_{no}|\nu)}{F^{-}1(P_{crit}|\nu)} \tag{3.4}$$

where

$$F^{-}1(p|\nu) = \{x : p = F(x|\nu) = \int_0^x \frac{t^{(\nu-2)/2}e^{-t/2}}{2^{\nu/2}\Gamma(\nu/2)}dt\} \tag{3.5}$$

as implemented by the MATLAB® statistical toolbox and ν is the number of degrees of freedom corresponding to the problem parameters.

In order to determine the ν as a function of the problem parameters, again simulations were used. Simulations were designed to cover the broad range of reference data length (s = 42, 98, 308, 812, 1512), a full range of characteristic depths ($n = s/7, 2s/7, ..., s$), various values of nearest neighbors ($k = 2s/14, 3s/14, ... , n$) and four different confidence limit values (L = 0.999 0.9975 0.975 0.9). For each simulation, the value of ν was calculated such that the metric would have a value of 10 at a sample point that was $\sqrt{10}$ times the distance from the mean as the confidence limit location. This should provide a two point calibration of the overall performance index C fitting the value of 10 and the value of 1 (the confidence limit). Data from these simulations are tabulated in Appendix B.

From these experiments, the value of ν was determined to be a strong function of $\sqrt{k/s}$, n/s, s, and the confidence limit L. Each influence was individually investigated, then merged into a nonlinear model and fit with nine coefficients simultaneously. The resulting model took the form

$$\nu(k, n, s, L) = e^{\frac{\sqrt{k/s}-(x_1\ln(1-L)+x_2 s+x_3)}{x_4(1-L)+x_5 s+x_6}} + x_7(n/s)^2 + x_8(n/s) + x_9 \tag{3.6}$$

where the optimized values for x_1 through x_9 are displayed in Table 3.1.

Table 3.1: Fitting parameters for degrees of freedom(ν) calculation

Parameter	Value
x_1	-0.080919
x_2	-3.4359E-05
x_3	0.64843
x_4	-0.87983
x_5	-3.8949E-05
x_6	-0.27710
x_7	-13.502
x_8	20.230
x_9	-8.1403

3.1.5 Extension to multiple dimensions

The algorithm development thus far has been achieved by simulating data in strictly one dimension. The goal in the development of the distance distributions was to reduce the multivariate problem into a univariate-based distribution, thus allowing the algorithm to be extended to multiple dimensions with only minor changes. Therefore, the calculation of the non-overlap probability and its threshold value remains the same for the multivariate case.

The main difference is that the distances are calculated in multiple dimensions. In order to compare different variables with different units, all variables are mean centered and scaled by the standard deviation of the variable within the reference data set. The sample points are then analyzed after applying the same transformation.

3.1.6 Block contributions

Once an overall metric is calculated in any fault detection algorithm, an engineer or operator is interested in the contribution of each variable or of each group of variables. For this algorithm, a blocking structure was implemented similar to that in [86]. It allows for a blocking matrix with multiple levels. At each level, variables can be blocked into arbitrary groups without any necessary relationships between the levels. Usually the last level has each variable in its own block in order to calculate the univariate contributions to each variable. When a blocking structure is configured for a problem, in addition to calculating the performance index for the overall problem with all input variables, the entire algorithm is repeated for each block in each level. For a block contribution, only the variables within the block are used in the sample point and characteristic distance calculations. In this way, the block performance index is based only the subspace defined by the variables within the block. Finally the overall index and all the block contributions are reported as a part of the algorithm results for each sample.

3.2 Algorithm validation simulations

To this point this method may seem very empirical, and there may exist an analytical solution that better implements the goals of looking at the local distance probabilities and comparing them to a sample point. In order to justify moving ahead with this fault detection method, a series of simulation results are presented to convince the reader of the power of this method.

3.2.1 One-dimensional problem with various configurations

The first set of seven simulations builds various reference sets by creating one-dimensional data from normal distributions with seven different means and standard deviations. The performance index is then evaluated for sample points from eight standard deviations below each mean to eight standard deviations above each mean. Table 3.2 describes the different configurations for each of the seven experiments. The resulting performance index are plotted in Figure 3.9 overlaying the experiments by the number of standard deviations away from the mean. For very different reference data sets and model parameters, there are very few false negatives or false positives using the limit of 1.0, and the overlap above this limit is acceptable. These results validate the scaling that was use to generate the threshold of the non-overlap probability and the scaling used to form the performance index, C.

Table 3.2: Configuration for one-dimensional simulations

Exp.	Mean	Std. Dev.	s	k	n	L
1	5	2	1000	30	300	0.995
2	10	20	500	25	200	0.995
3	100	1	750	25	200	0.995
4	50	5	1200	35	400	0.995
5	-2000	10	750	40	300	0.995
6	-2000	10	750	150	400	0.995
7	-2000	10	1250	200	400	0.995

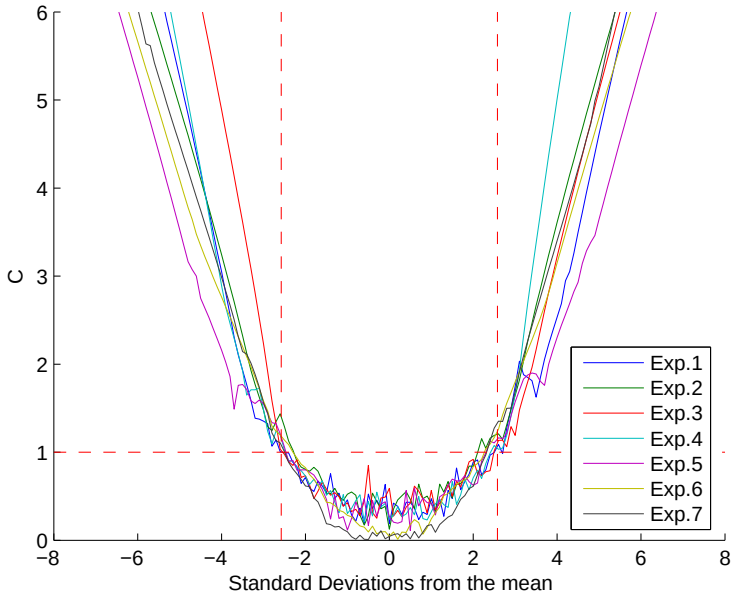


Figure 3.9: Performance metric plotted versus standard deviations from the mean for multiple simulations

3.2.2 One-dimensional non-normal distributions

Another set of simulations evaluates the performance of the KNN method when applied to non-normal distributions and evaluates the effect of the number of reference data used on the performance of the C metric. For this simulation, ten probability distributions were analyzed with the given distribution parameters,

- Normal distribution, $\mu = 0$, $\sigma = 2.5$
- Exponential distribution, $\mu = 5$
- χ^2 distribution, $V = 16$
- Rayleigh distribution, $b = 2.5$
- Uniform distribution, -4 $< x <$ 4
- Weibull distribution, $A = 2$, $B = 1$
- Lognormal distribution, $\mu = 1$, $\sigma = 0.2$
- f distribution, $v_1 = 5$, $v_2 = 8$
- Poisson distribution (discrete), $\lambda = 20$
- Geometric distribution (discrete), $P = 0.3$.

Reference data and the theoretical 99% confidence limits were generated based on each distribution for each simulation using the MATLAB® Statistics Toolbox. For each simulation, reference data were generated and used to create a

KNN model (simply the reference data) and a PCA model (mean and standard deviation estimate). Using these two models, 150 samples points were evaluated from 20% less than the lower confidence limit (or zero for the one-sided distributions) to 20% higher than the upper confidence limit. For each distribution this simulation was repeated five times for each of 8 different reference data sizes (33, 100, 200, 400, 850, 1250, 2500, and 5000). For each simulation, the k and n values were selected as $k = \sqrt{s}$ and $n = s/3$, where s is the number of reference data within the model.

The results for these simulations are illustrated in Figure 3.10 through Figure 3.17 for each of the eight different reference sizes. For each distribution and each number of reference data, the C metric for each sample point is plotted as a function of the sample point value with all five simulations combined, and C values above 5 are shown at the value of 5. Also, a histogram which pools the reference data of the five simulations together is overlaid. For the two-sided distributions, the 0.5 and 99.5 percentile theoretical limits are plotted, and for the one sided distributions the 99.0 percentile theoretical limit is plotted. Finally a horizontal line is placed at $C = 1$ to show the warning limit of the KNN model.

In order to analyze the performance of the KNN method to identify excursions from non-normal distributions, the counts of the Type I and Type II errors for each distribution were summed for each of the five simulations (for a total of 750 sample points). In Figure 3.18, the Type I and Type II errors (along with the total count) are plotted for the KNN and PCA model

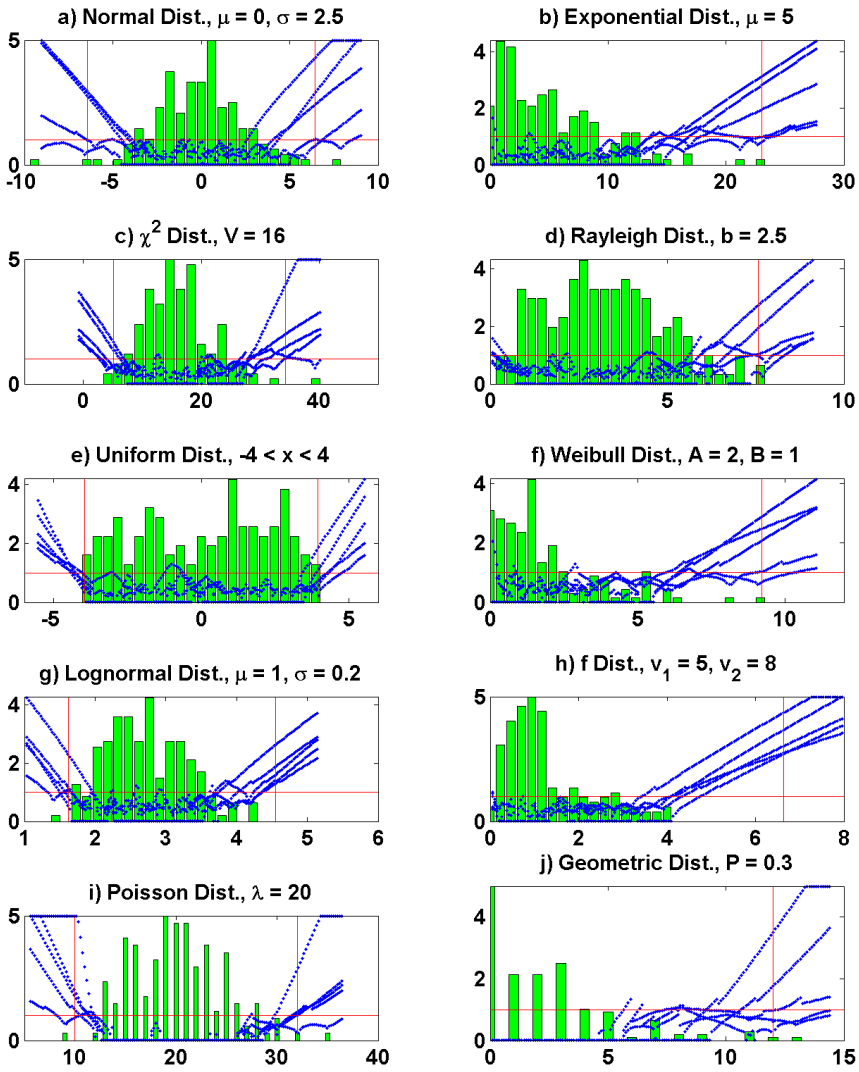


Figure 3.10: Non-normal distribution simulations using 33 reference points

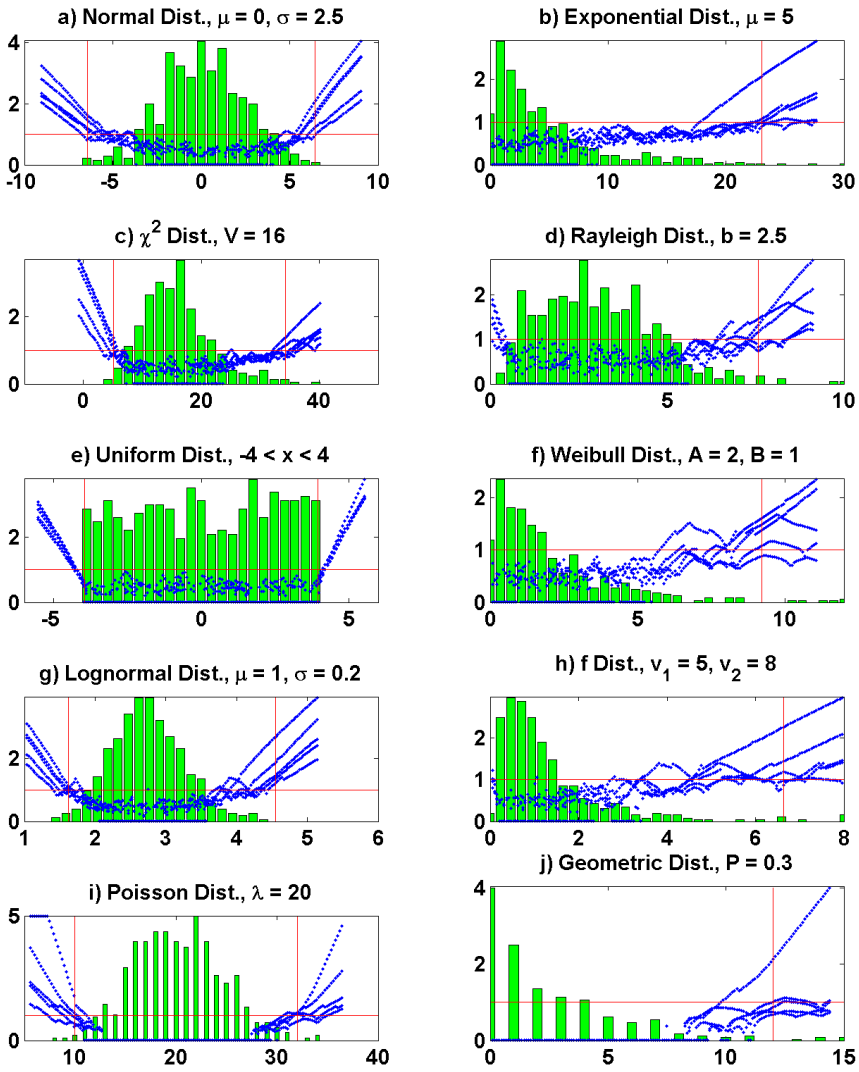


Figure 3.11: Non-normal distribution simulations using 100 reference points

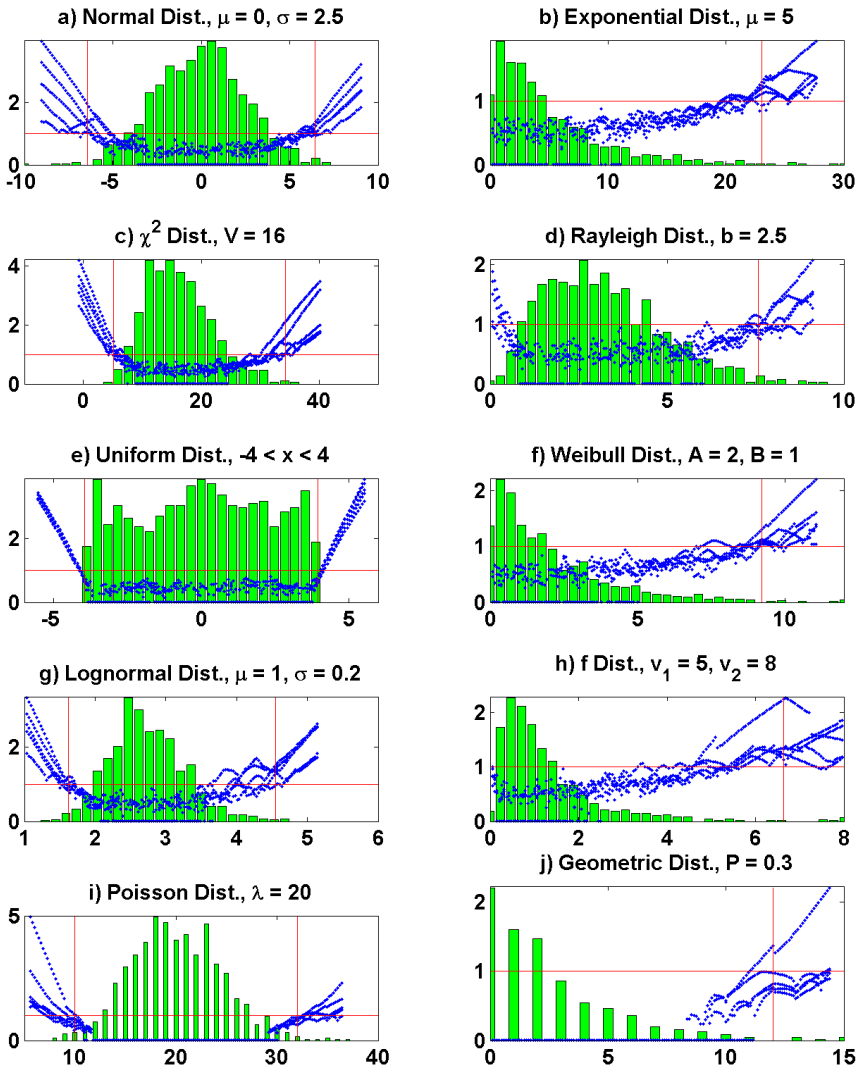


Figure 3.12: Non-normal distribution simulations using 200 reference points

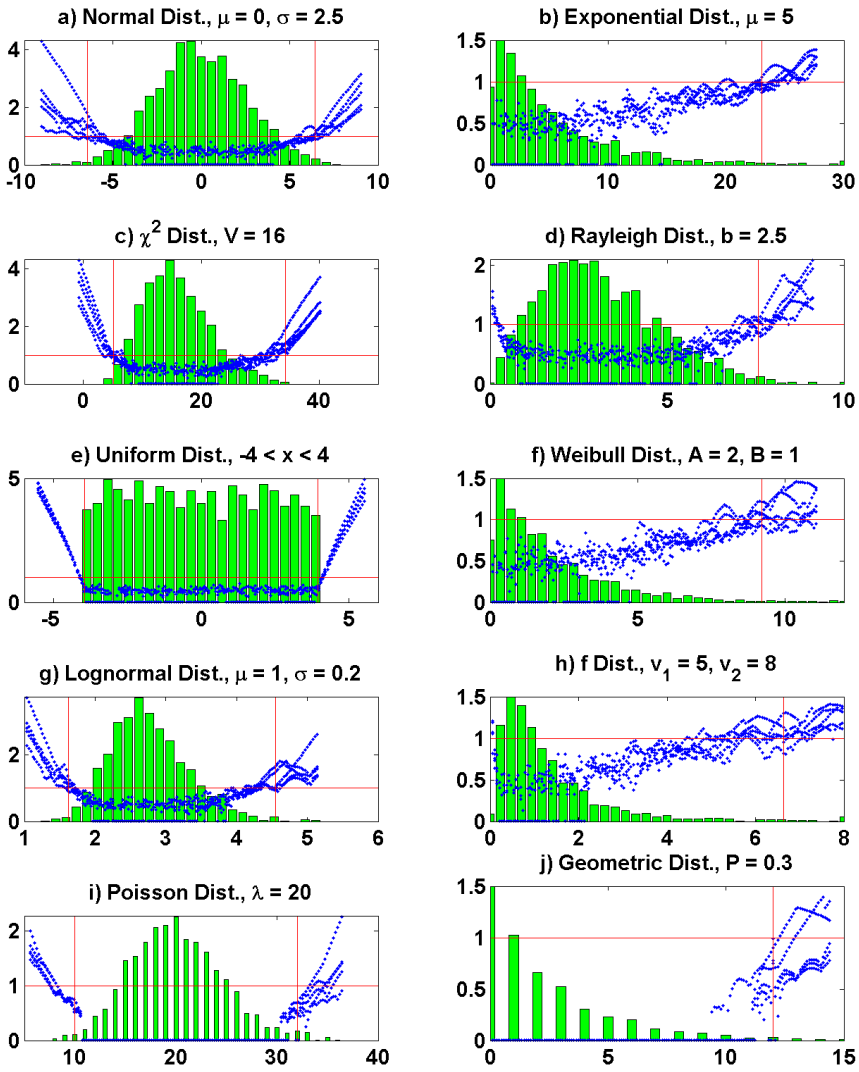


Figure 3.13: Non-normal distribution simulations using 400 reference points

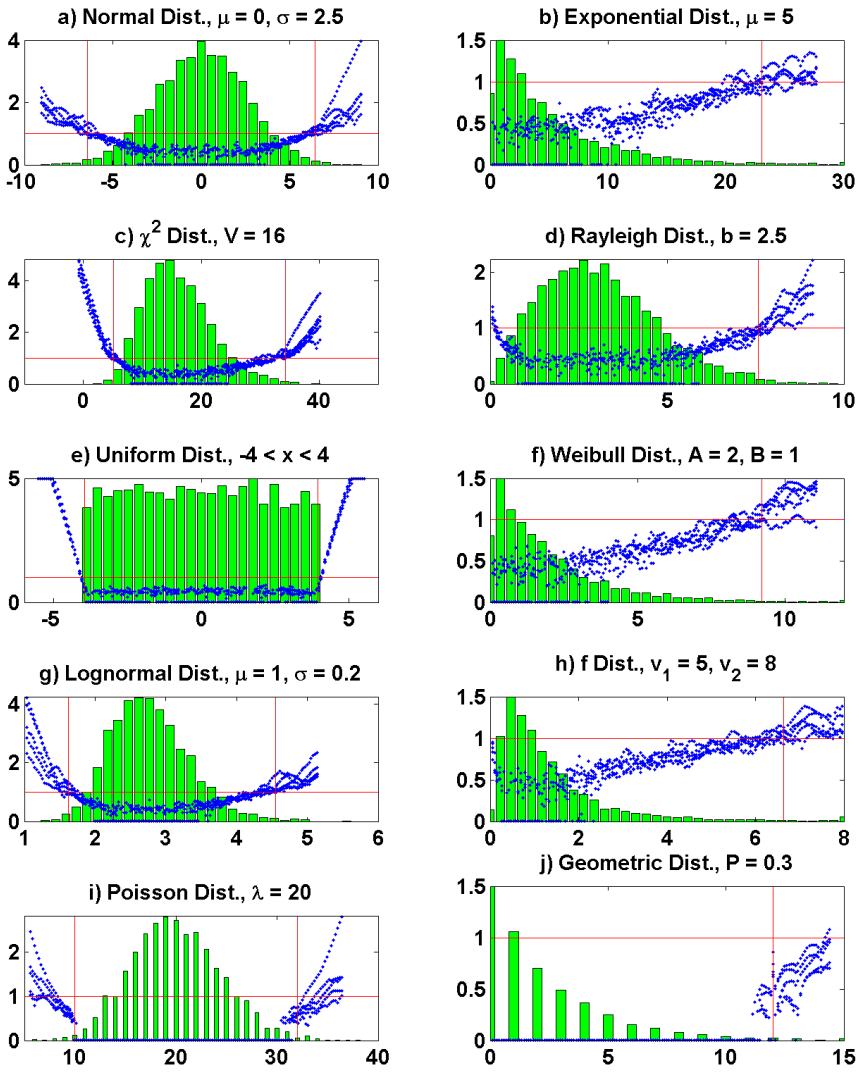


Figure 3.14: Non-normal distribution simulations using 850 reference points

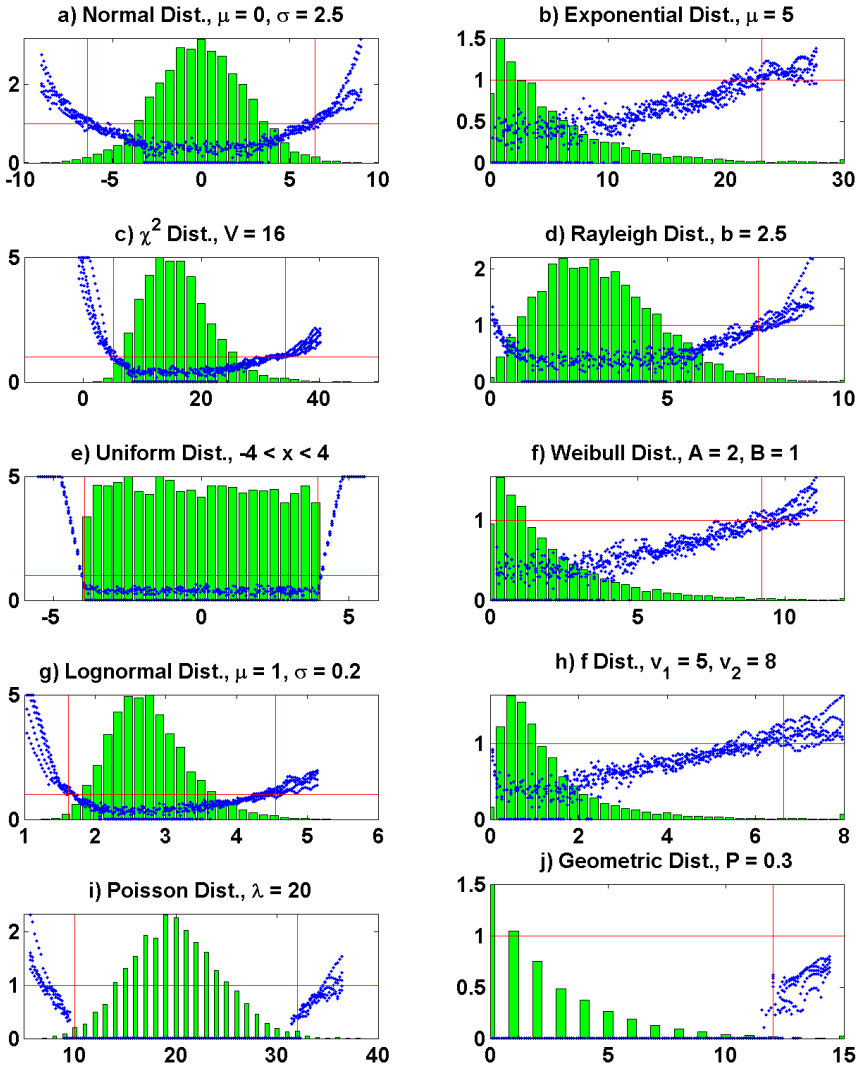


Figure 3.15: Non-normal distribution simulations using 1250 reference points

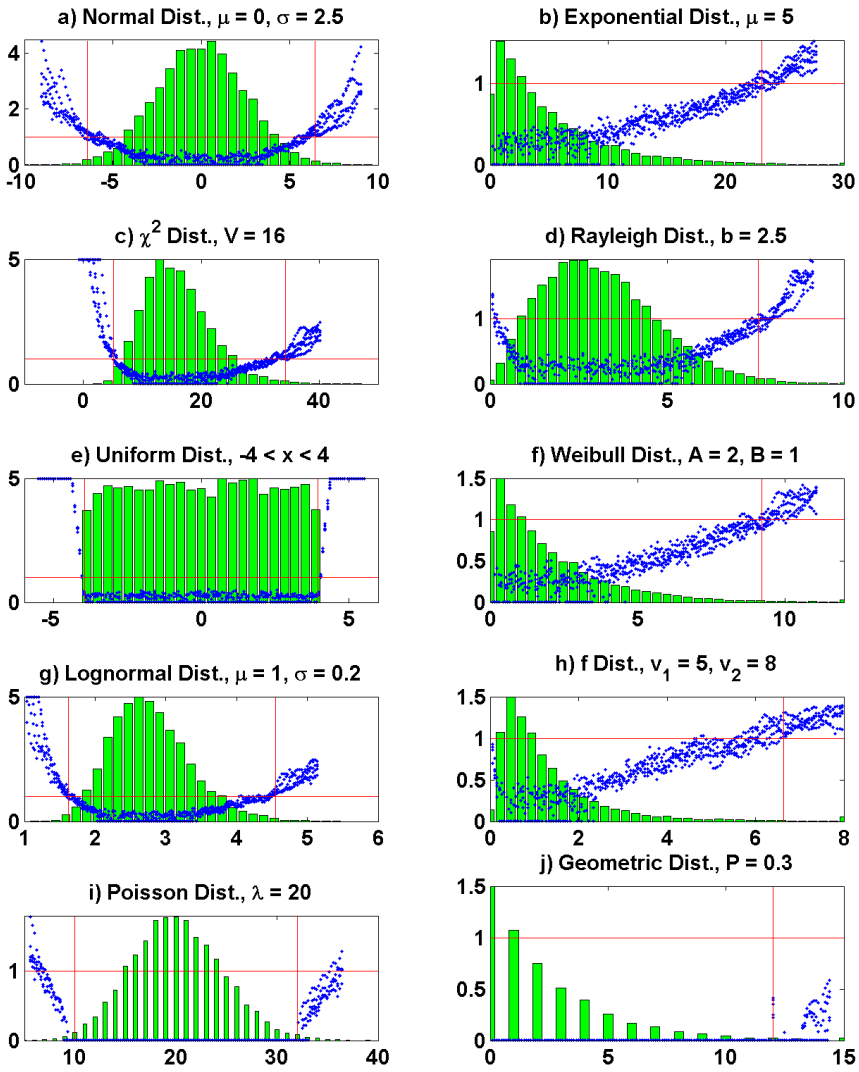


Figure 3.16: Non-normal distribution simulations using 2500 reference points

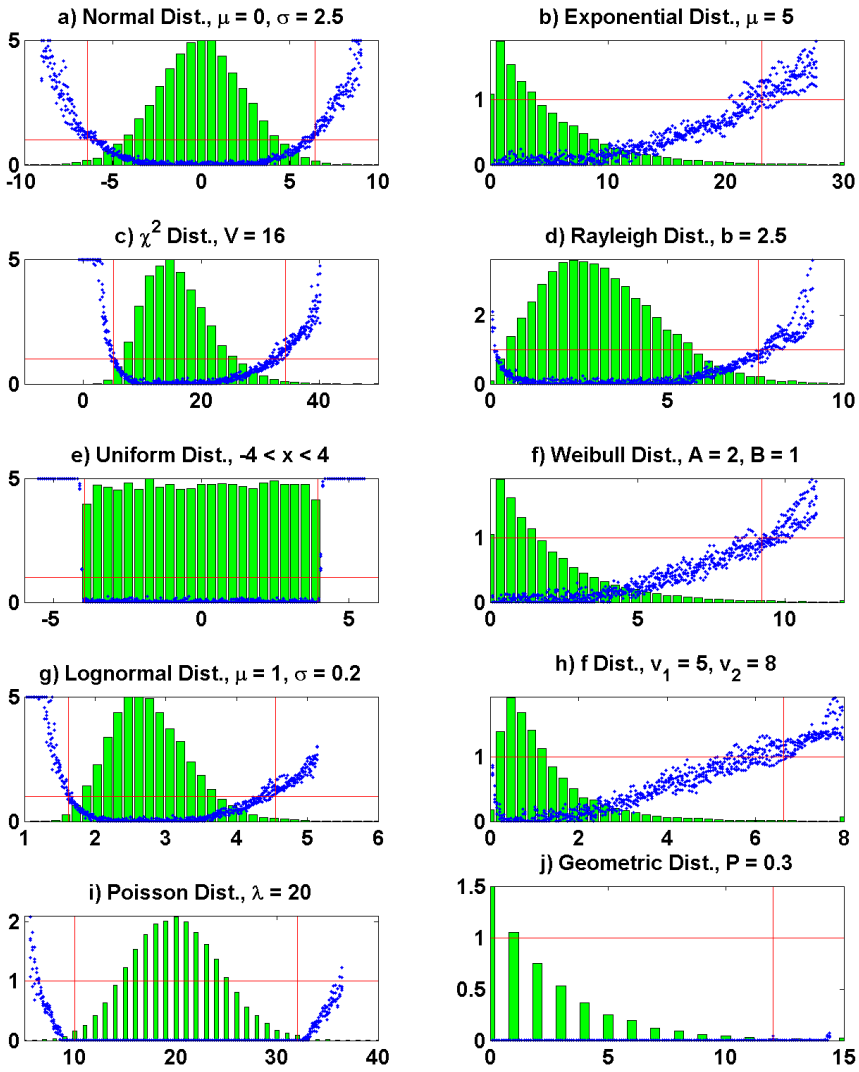


Figure 3.17: Non-normal distribution simulations using 5000 reference points

for each distribution as a function of the number of reference data that was used. In Figure 3.18.a, the normal distribution is compared. For the normal distribution (and most others), as the number of reference data increases, the error counts for the KNN model decrease initially then generally approach an asymptote. For the normal distribution, the KNN model does not perform as well as the PCA model (simply a Gaussian model), but does decrease to an error count of 30 at 850 reference data points. In the exponential, χ^2, uniform, Weibull, log normal, and f distributions the KNN method performs quite well reaching error counts of 50 or less at 850 reference points, and it clearly outperforms the Gaussian model. For the Rayleigh distribution in Figure 3.18.d, both the KNN and PCA methods perform similarly.

For the discrete Poisson and geometric distributions, as the number of reference data increases, the error counts for the KNN method actually increase. Internally to the KNN calculation, the estimation of the sample point distribution and characteristic distribution break down when there is a high number of repeated identical values. When all the k nearest neighbors for a sample point have an identical distance and when each of the k nearest neighbors of n nearby points all have identical values, there is no way to estimate the distributions. For robustness in the algorithm, as long the sample point is within one discrete step from n reference points which each have at least k identical values, then the C metric is defined to be zero. As the number of sample points increases, each value within the discrete distribution simply gets an increasingly larger population, and a larger range of sample points are

assigned a value of zero, which leads to the increase in Type II errors for these distributions.

In all the different continuous distributions, Figure 3.18 shows that there are limited performance gains for models above 400 to 1250 reference data points. Also, models with reference data sizes below 100 or 200 reference data points are significantly improved when larger reference data sizes are used.

In addition to looking at the performance of the KNN algorithm, the total calculation time for each simulation of all ten distributions was collected for each reference data size. The calculation times were scaled by time to calculate the simulation with 33 reference data points and the resulting relative values are displayed in Table 3.3. These relative calculation times are plotted versus the number of reference data used in the simulation in Figure 3.19. From this calculation time data, it was determined that the KNN algorithm (when analyzing data for a one-dimensional problem) is $O(s^2)$, where s is the number of reference data within the model.

Further timing experiments were run varying the number of variables within the reference data and the number of reference data in the model. Reference data was generated by creating random data from a mean vector and covariance matrix for each problem size. The number of variables was varied using values of 1, 2, 4, 7, 12, 25, 50, 100, 200, 250, 650, 1000, and 1250. The number of reference data points was varied using values of 33, 100, 200, 400, 850, and 1250. In each simulation the overall C value was calculated

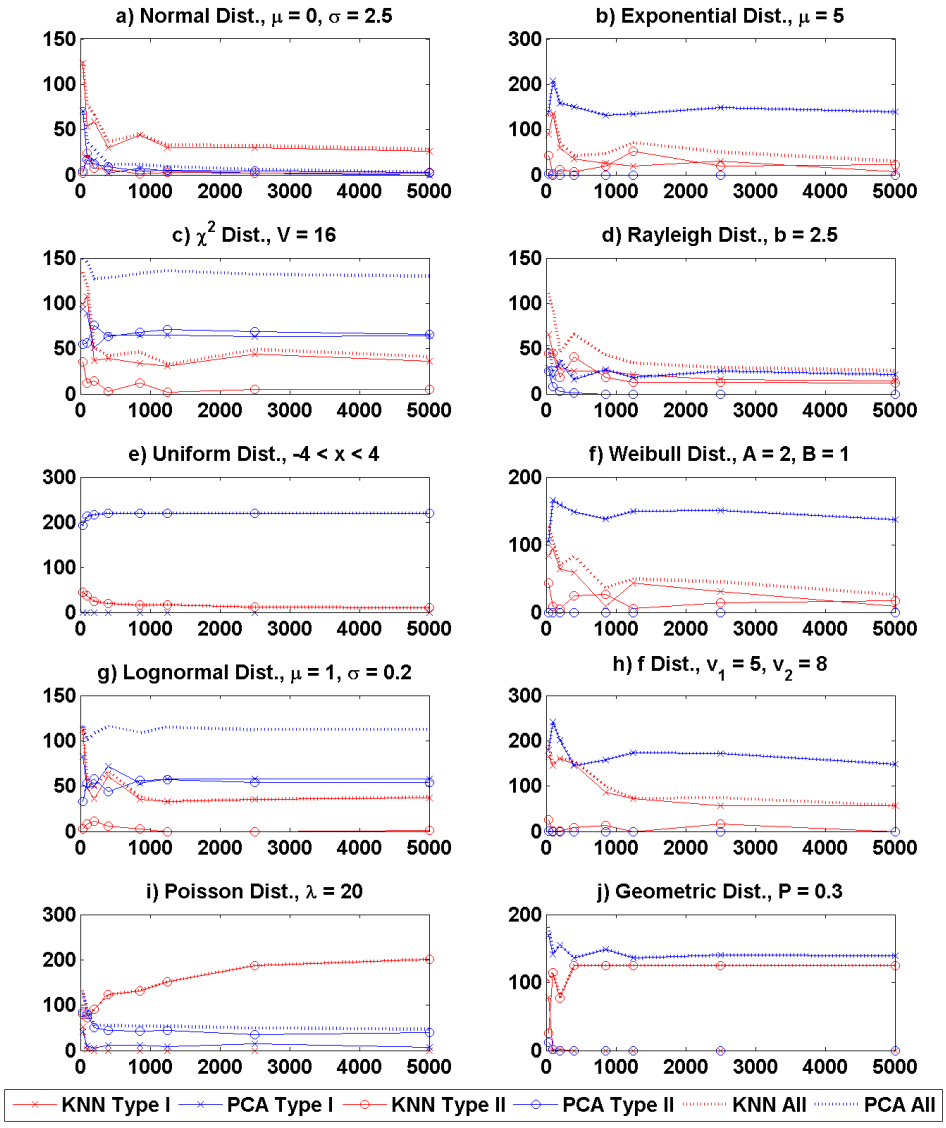


Figure 3.18: Type I errors, Type II errors, and combined errors for non-normal distribution simulations using various number of reference points for both KNN and PCA models

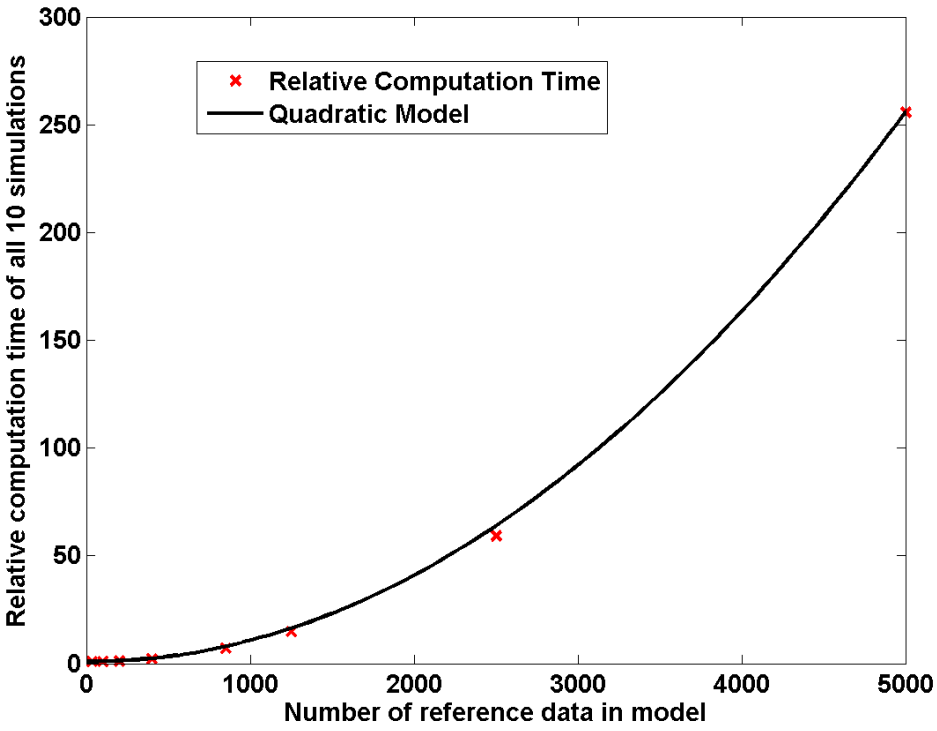


Figure 3.19: Relative computation time for all KNN simulations for each number of reference points

Table 3.3: Relative computation time for each simulation

# Ref Data	Relative Computation Time
33	1.0
100	0.9
200	1.1
400	2.1
850	7.1
1250	14.8
2500	59.2
5000	255.8

along with C value for each univariate contribution. It was determined that the relative computation time could be predicted by

$$t_r(s, v) = (v + 1)(1.43e^{-6}s^2 + 0.349) + 3.39e^{-7}(v + 1)^3 \tag{3.7}$$

where s is the number of reference data within the model and v is the number of process variables within the model. For models with less than around 400 process variables the algorithm is $O(vs^2)$. For models in this range, the algorithm scales linearly with the number of process variables, which is reasonable since the algorithm calculation repeats for each block contributions in addition to the overall metric calculation. For models with more process variables, the algorithm goes to $O(vs^2 + v^3)$. For these very large models, there is more than a linear increase in computation time as the number of process variables within the model increases.

3.2.3 Multiple dimensional problem with induced faults

In this example, 3000 sample points are simulated for 5 process variables (p_1 through p_5) with some internal correlations. The first 500 samples are considered normal, have no faults, and are used for the model building step. The next 250 samples also have no faults. Starting at sample 751, each set of 150 samples has a different fault. The fault types used are ramp disturbances, step disturbances, and changes in variation. In addition, some are applied before the correlation and some are after the correlation. All the simulated data are plotted as a time series in Figure 3.20. Scatter plots of the modeling data for the first 500 points showing the internal correlation are shown in Figure 3.21. In these figures it is clear that there is correlation between process variables 1, 3 and 5 and also between process variables 2 and 4. Scatter plots for the remaining test data are shown in Figure 3.22. Faults can be seen in this figure that go outside the previous range of data and that clearly violate the previous correlation of the data.

The reference data for the KNN fault detection method was taken as the first 500 points. Then, each of the data points (including the first 500) was treated as a sample point and the overall, block contributions, and univariate contributions were calculated and are shown in Figure 3.23. The values for k, n, and L that were used were 23, 167, and 0.995 respectively. The blocking structure included a block for process variables 1, 3, and 5 and a second block for process variables 2 and 4.

This data set has many interesting features that illustrate the properties

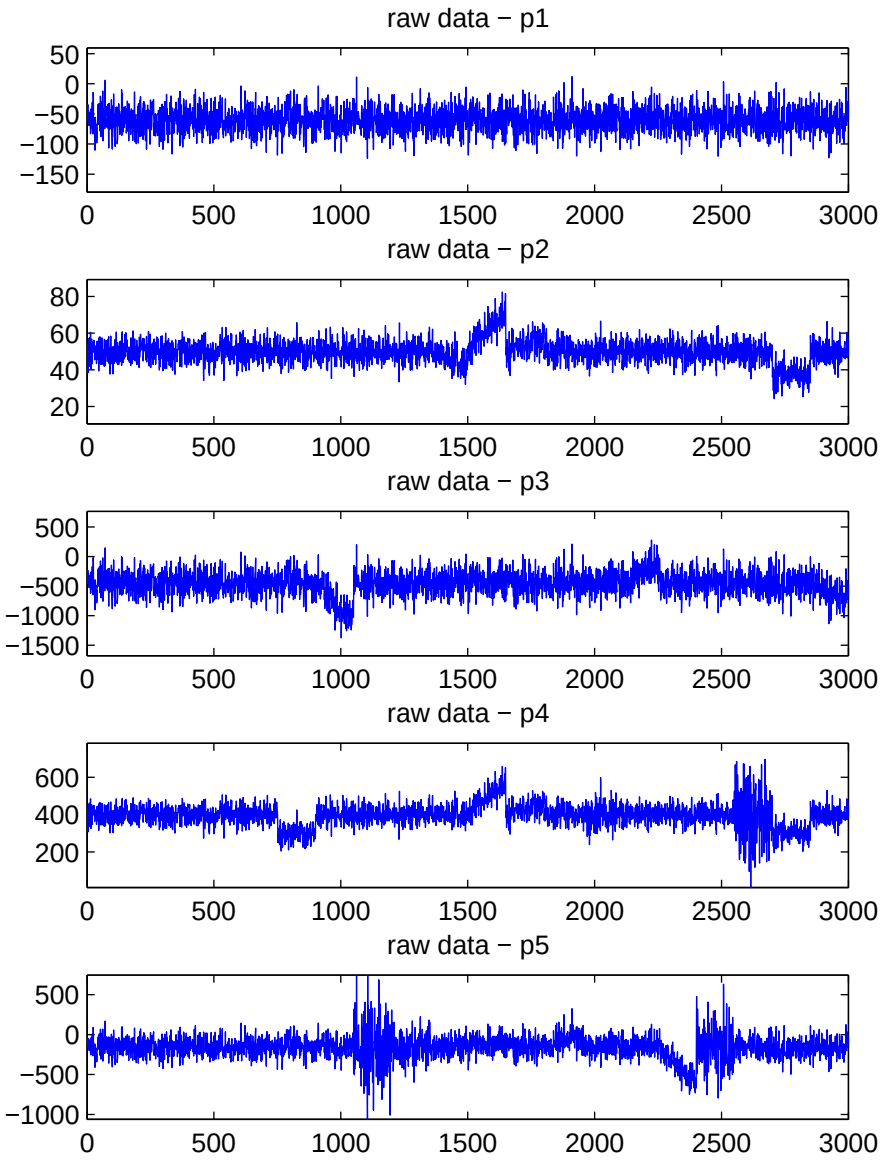


Figure 3.20: Simulated data for 5 variables used for multivariate test

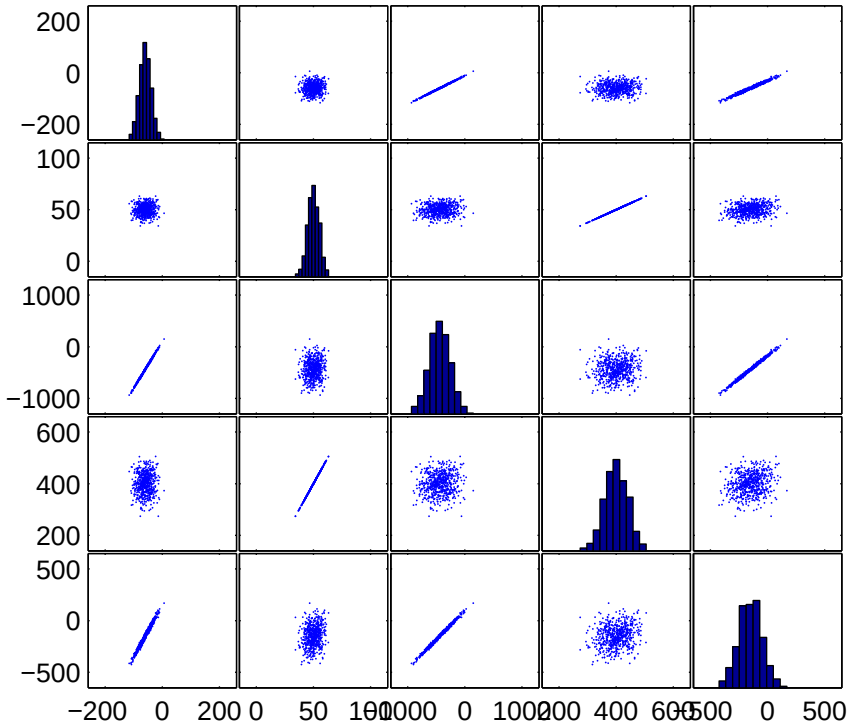


Figure 3.21: Scatter plots of modeling set for 5 variables

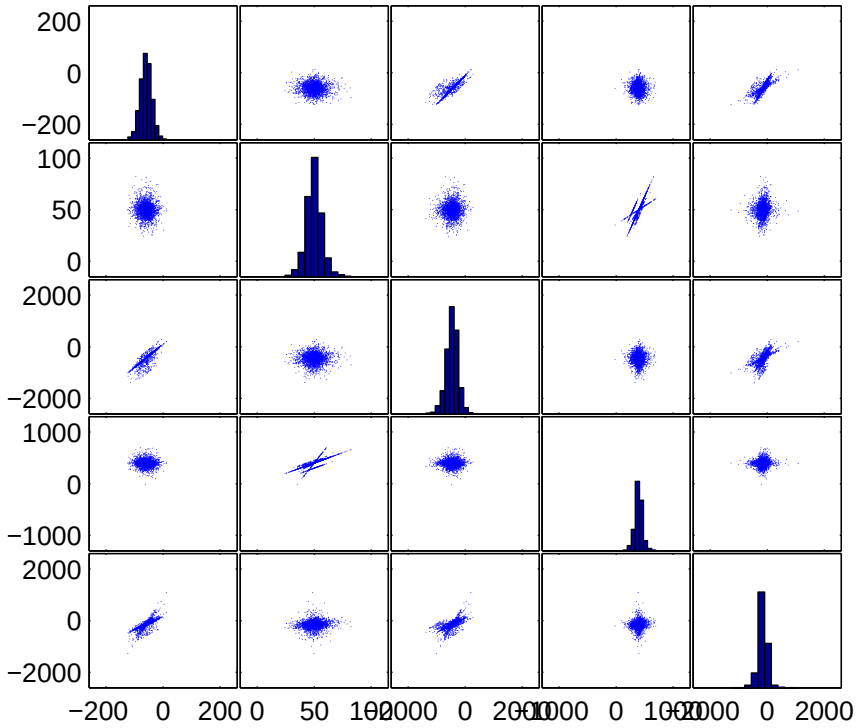


Figure 3.22: Scatter plots of testing set for 5 variables

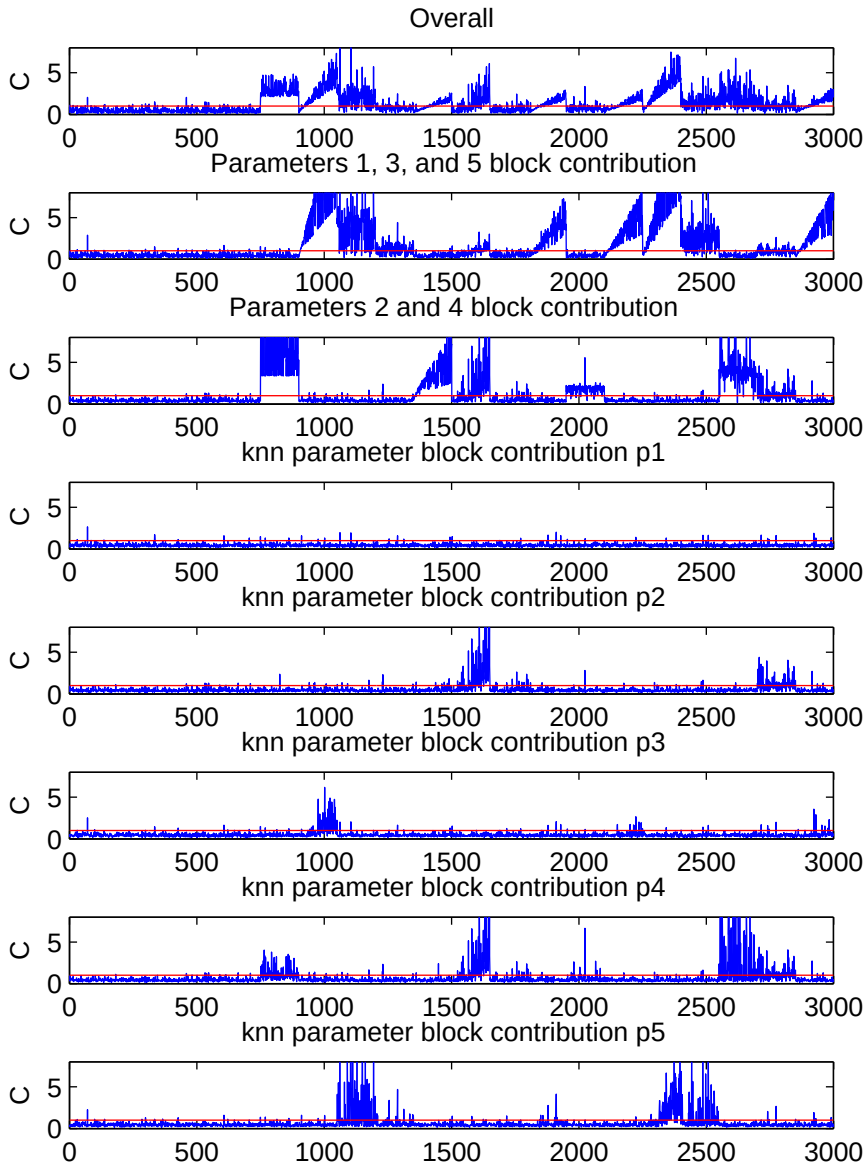


Figure 3.23: Performance index and contributions for simulated fault data

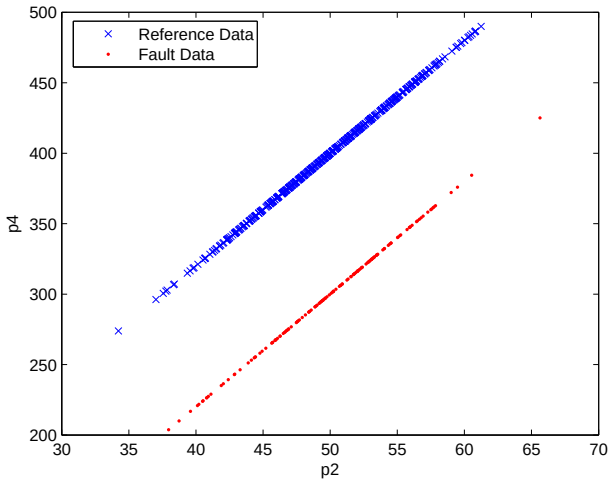


Figure 3.24: Scatter plot of process variable 4 versus process variable 2 with reference and fault data from samples 751 to 900

of the KNN method. In the first set of faulty data from samples 751 to 900 as shown in Figure 3.24, process variable 4 has an output step disturbance of 2.5 standard deviations. The univariate contribution identifies it as seen in Figure 3.23.g. This same fault is also amplified in the block contribution in Figure 3.23.c. This block contribution for process variables 2 and 4 has an even larger magnitude because this fault also violates the strong correlation between process variables 2 and 4. In addition, the overall contribution shown in Figure 3.23.a still has a significant fault signal although it is muted from the block contribution signal.

In the faulty samples from 1501 to 1650 there is a ramp disturbance that is seen in both process variables 2 and 4 as shown in Figure 3.25. Here

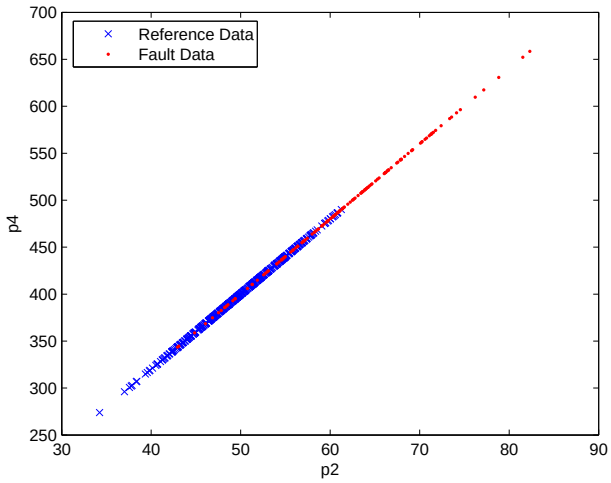


Figure 3.25: Scatter plot of process variable 4 versus process variable 2 with reference and fault data from samples 1501 to 1650

this fault follows the correlation of 2 and 4 but moves outside the modeled range, resulting in a fault. The magnitude of the block contribution in Figure 3.23.c and the individual contributions in Figures 3.23.e and 3.23.g are the same, signifying that there is no violation of the correlation structure, but still a faulty block.

A third type of fault that is identified is a ramp disturbance in process variable 4 that occurs from sample 1351 to 1500 and is shown in Figure 3.26. This fault is of small enough magnitude that the individual contribution of process variable 4 in Figure 3.23.g remains below its threshold for this period. The fault does represent a clear violation of the correlation structure, and the block contribution and overall index show the appropriate violations in figures

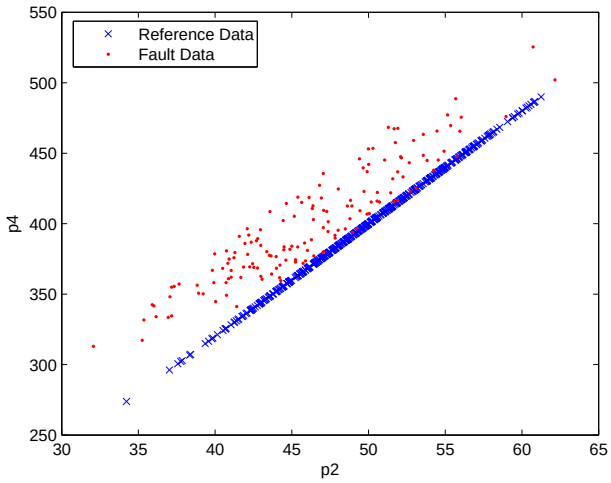


Figure 3.26: Scatter plot of process variable 4 versus process variable 2 with reference and fault data from samples 1351 to 1500

3.23.b and 3.23.a respectively.

This multivariate simulation validates the use of this method in multiple dimensions. The threshold value remains appropriate for the five-dimensional problem, the two and three-dimensional problems for the block contributions, and the one-dimensional problems for the individual contributions. In addition, the overall contributions seem to reflect the faults in the individual process variables and the relationships between the variables. The block contributions clearly show violations in the individual variables and the relationships between the variables within the block. The individual process variable contributions seem to also be appropriate univariate fault indexes, as they are the same as the one-dimensional problems discussed earlier.

3.2.4 Two-dimensional problems with multiple populations

The final simulations are designed to show the method's ability to segregate multiple populations, ability to tolerate isolated outliers within the reference set, and the ability to tune the algorithm to be sensitive to local densities. In the first simulation, the reference data set is built with two 500 point populations generated with different linear correlations that have a slight overlap. In addition, 30 points are randomly generated with a uniform probability over the entire region and were added to the reference set. Sample points were taken over a 100 by 100 grid, and the overall performance index was calculated for each point. A contour plot of the performance index calculated with k, n, and L equal to 33, 344, and 0.995 respectively is plotted in Figure 3.27. The performance index calculation was repeated with this same reference set with n as 688 and 172 and plotted in Figures 3.28 and 3.29 respectively.

In the all of these figures, the threshold limit of 1 captures a reasonable boundary around the two correlated populations. The uniformly distributed (outlying) points do affect the value of the metric to some degree, but the method does allow points within the reference set to have values greater than 1 if they do not fit the distributions assumed within the method. This feature of the method allows it to be tolerant of outliers getting into the reference data set. When enough points from an outlying region cluster within the reference data set, the performance index will go below the threshold limit and create another acceptable region.

Also, this simulation helps show that as the value of n decreases, the

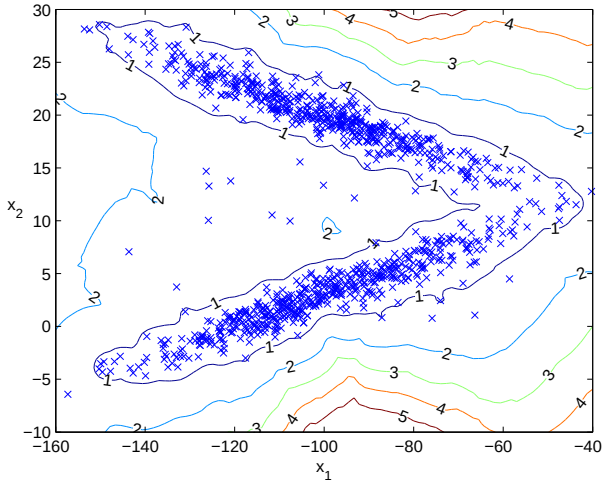


Figure 3.27: Contour plot of performance metric for two-dimensional problem, n=344

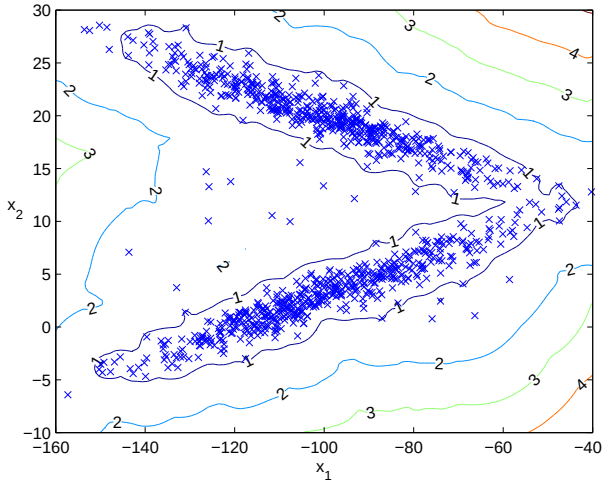


Figure 3.28: Contour plot of performance metric for two-dimensional problem, n=688

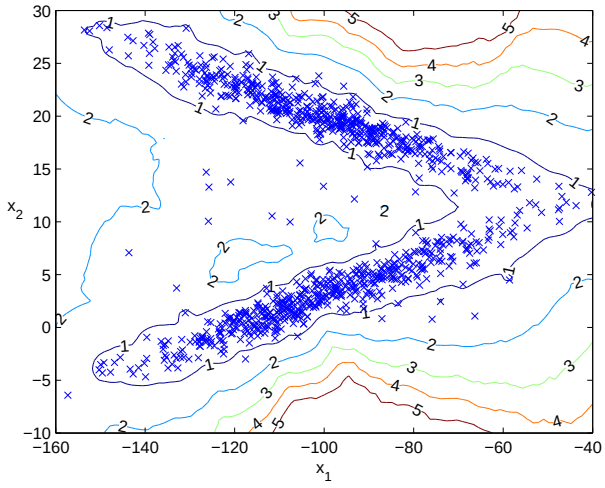


Figure 3.29: Contour plot of performance metric for two-dimensional problem, n=172

metric becomes more sensitive to local populations. In Figure 3.28 with an n value of 688 (two thirds the population size), the contours above and below the two data sets are essentially parallel to the correlations. As the value of n decreases to 344 and then to 172 in Figures 3.27 and 3.29, the contours get closer together near the dense regions of the reference data indicating an increase of the gradient of the performance metric. These lower values represent a higher sensitivity of the performance index in dense regions than in the sparse regions. The n value can be thought of as the tuning knob for this sensitivity to balance allowing multiple populations with different distance distributions versus having smoother global expectations of the distance distributions.

In order to validate this observation about local to global sensitivity, a final simulation was performed. In this simulation, 200 points were generated

for each of 5 populations in a two dimensional space. Each population used a separate standard deviation from the mean that was applied in the both dimensions to create a reference data set with variable population densities. The locations of and the standard deviations for each population are described in Table 3.4. The contours of the performance index are plotted in Figures 3.30 through 3.33 for this reference set using a k value of 32, an L value of 0.995, and n values of 167, 334, 667, and 1000 again for sample points covering the two-dimensional space.

Table 3.4: Location of and standard deviation for each population in the 5 population simulation

Center x_1	Center x_2	Std. Dev.	Population
-5	5	1	200
5	5	0.5	200
0	0	0.75	200
-5	-5	2	200
5	-5	0.1	200

In this simulation, it is clear that the gradient of the performance index is very different for different values of n. For the lowest value of $n = \frac{s}{6}$, the performance index is very dependent of the local population as seen in Figure 3.30. In this graph, the region near (5,-5) has very steep gradients reflecting the very dense and isolated local population. As n increases, the contours broaden and more reflect the global population density and distributions. As n approaches the s value for the reference set, the region of values less than 2 in Figure 3.33 is entirely continuous and is approaching a convex region.

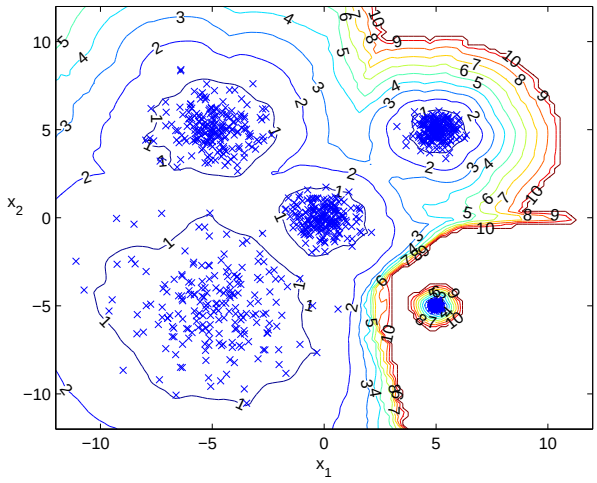


Figure 3.30: Contour plot of performance metric for 5 population problem, n=167

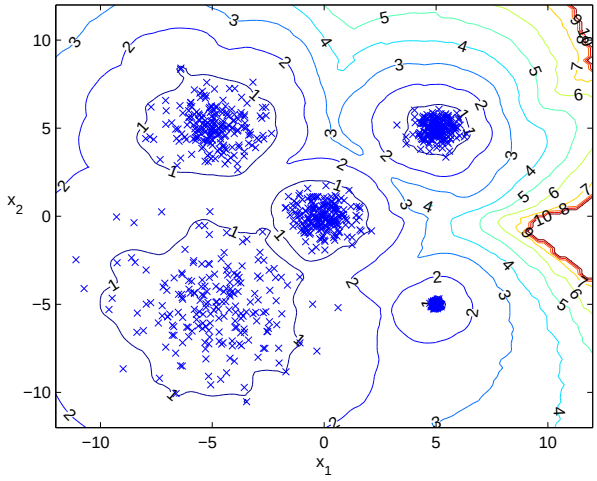


Figure 3.31: Contour plot of performance metric for 5 population problem, n=334

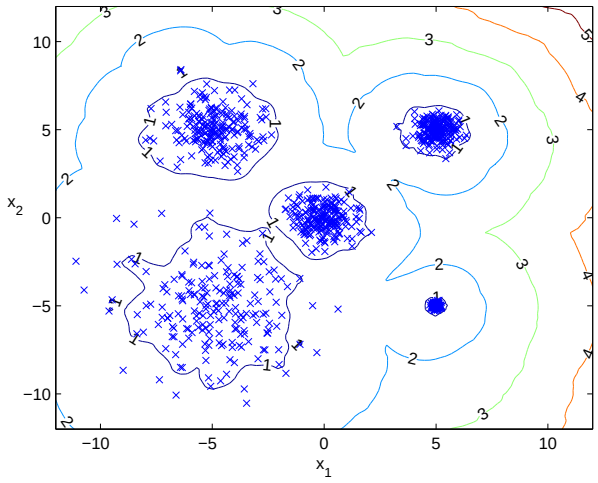


Figure 3.32: Contour plot of performance metric for 5 population problem, n=667

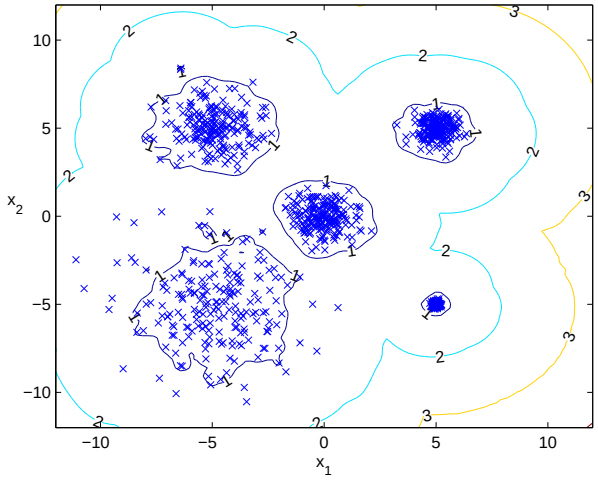


Figure 3.33: Contour plot of performance metric for 5 population problem, n=1000

These last two simulations show the value of having an available tuning knob to balance between local and global contributions. If n is too high, the performance metric may not indicate local gross faults between population centers, and if the n value is too low, the global shape of contours may not be the shape expected around continuous population areas.

3.3 Conclusions

In this chapter a new method for performing fault detection based on nearest neighbor distributions is introduced and evidence of its performance is presented. The concept of the sample point distribution and the characteristic distribution for a sample point is explained and illustrated. Then, based upon these distributions a new fault detection metric is defined with three tunable parameters (the number of nearest neighbors to search for each point (k), the number of nearest neighbors to use in the characteristic distribution (n), and the confidence level (L)). This new method's performance index is shown to have a threshold limit similar to the specified confidence limit for normal distributions, and the method is flexible enough to have similar limits even when multiple populations or non-normal distributions are present. The performance index also scales similarly for different problem sizes and different values of k and n. This method is shown to generalize in multiple dimensions and provides appropriate individual variable and block contributions values.

Chapter 4

Multivariate Analysis Reporting and Visualization

This chapter describes a new general methodology to monitor a steady state system (or one with slow dynamics) with a multivariate fault detection method. The goal is to leverage an adaptive multivariate fault detection method (such as PCA or KNN) and its contribution results organized into a hierarchy to monitor a set of related process variables (or electrical test parameters as is used in this chapter) while allowing easy navigation of the important results. This visualization and navigation system takes advantage of the fault detection results, the parameter blocking structure created by the end user, the adaptive model updates, the dynamically calculated univariate limits, and any existing available fixed limits. It enables the user to navigate from the highest level performance metric for a given sample or time period and quickly find the most extreme and most frequent excursions, identify the parameters blocks in which they occur, and quickly visualize the parameter charts with all the known control limits.

In addition to describing this general methodology, the last section of this chapter describes the results of applying this resulting system to monitor

electrical test data at Spansion's Fab25 (Austin, TX) with use of the RPCA method based on the work by Cherry[17].

4.1 Motivation

The goal of fault detection for any given process is to provide notification when a process excursion occurs and to diagnose the root cause of the problem as quickly as possible. Historically, monitoring and visualizing product quality values was predominantly performed by statistical process control (SPC) charts. SPC charts utilize statistical analysis to monitor a single parameter at a time. They can notify the user if a statistically significant excursion occurs based on set limits or an estimated mean or standard deviation. SPC charts provide the security that any time a parameter exceeds the specified control limits, proper notification will occur. They also provide a historical record of the parameters over time and can visually identify trends and biases in the data. Unfortunately, if a large number of parameters are to be monitored, the number of SPC charts and the number of manually configured limits can become unmanageable. In addition, when an excursion occurs, all the individual parameter alarms and their corresponding charts need to be examined to identify a root cause.

In order to advance from this basic monitoring system, improvements can be made by taking advantage of multivariate fault detection results. A first requirement is that the system must retain the individual parameter chart views similar to SPC charts because end users are comfortable using these for

troubleshooting. These parameter charts are also supplemented with many different types of limits such as high and low static and dynamic limits. The second requirement is to incorporate multivariate performance metrics that examine the correlation between all the parameters and within parameter blocks. The third requirement is to provide a structure to these block results so the user can obtain a high level overview of excursions and then drill down to the individual results.

The final component of the visualization system is to provide reports that highlight to the user the individual parameters and blocks of parameters where the most extreme and most frequent excursions are occurring over a recent time period. This is accomplished by monitoring high and low limits on both static and dynamic control limits for each parameter and multivariate block. All these checks are then summarized in a periodic summary report that has a rolling historical window and in a sample summary report that gives all the results for a single sample or small group of samples.

4.2 Analysis methods and results

This section describes the necessary organization of the parameters into logical groupings and the results that need to be calculated and retained for each parameter and each grouping in order to enable the multivariate fault detection system.

4.2.1 Parameter blocking structure

In order to keep the monitoring system manageable as the number of parameters increases, it is important to break the parameters into logical blocks. These parameter blocks are to some extent arbitrary, but should have meaning to the engineer in charge of monitoring the system. These blocks are formed into a hierarchy so that the user can have a high level view of the system and see where the excursions are located. This is done by making sure that each level in the blocking structure forms parent-child relationships in the form of a branching tree from the overall node down to the parameter contribution nodes. As more detail is required within the results, the user can also drill down deeper into the parameter hierarchy until the individual parameters are reached. An example of a parameter hierarchy is illustrated in Figure 4.1. In this example 18 parameters are being monitored. They are divided into six level 1 contributions, three level 2 contributions, and an overall performance node. This flexible architecture allows for an arbitrary number of blocking levels between the parameters and the overall node and allows for an arbitrary number of blocks within each level. Results are provided not only for each of the 18 univariate parameters, but also for each block within the hierarchy.

4.2.2 Univariate analysis

In the overall monitoring scheme there are a series of static limits that are used to monitor the system. These static limits parallel traditional SPC

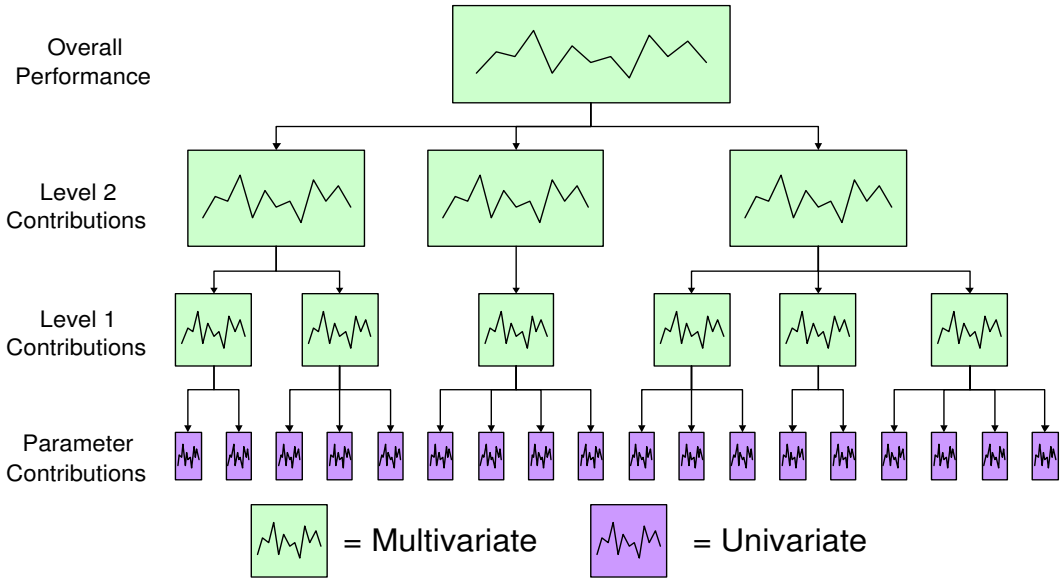


Figure 4.1: Example parameter hierarchy[12]

limits. These must be manually maintained by an engineer if the process happens to be changing frequently. Not all parameters are required to have every limit, and depending on the importance of the parameter to the system (or to achieve quick detection of a fault of the system) some limits can be omitted.

The first limit applied to the data is a physical significance limit. This is simply a check to see if the signal is a reasonable measurement at all. These are usually fairly wide and cover the entire range of the measurement device. Any data outside this range is treated as missing data. If a small percentage of the total number of parameters is missing, then the missing data can be reconstructed using the multivariate fault detection algorithm. This concept

takes advantage of the redundancy of having multiple well correlated sensors.

The next limit is the static fault limit. This is usually a specification limit that should rarely be exceeded. Any parameter values beyond this will be reported as a very high priority alarm. It is important to set these specification limits at levels so that the range has equally "high" importance for each parameter.

The final static limit is the warning limit. The warning limit should be set as the traditional control limit with a value of plus or minus three standard deviations or at a specific confidence limit from the expected mean for each parameter. These static warning limits provide an early warning that a parameter is beginning to move significantly and provide a good safety net when there is a heavy reliance on dynamic limits that may slowly drift from the actual process window.

For each univariate parameter, the result of the static warning and static fault check are saved. For each hierarchy node, the node value is simply the logical "or" of the results of all the child parameters within the span of the node. So unlike multivariate results (as introduced in the next section), if a parent hierarchy node has an excursion, then at least one of it children parameters had an excursion.

4.2.3 Multivariate analysis

In order to implement a multivariate analysis method, in most cases the parameters to be analyzed will be preprocessed. In most methods, some

attempt will be made to put all parameters into a common unit system. Most commonly this is done by performing mean centering and standard deviation scaling based on the mean and standard deviation of each parameter. This translation of the data allows parameters with arbitrary units to be compared in a multivariate sense.

In addition, historical data will be processed to build an appropriate model for the multivariate technique. In the case of RPCA used in this work, one of the model building steps is to remove the parameters that do not have a normal or near-normal distribution from the multivariate model. Parameters that take on few distinct values can cause a serious problem if they are left in the model, because small (normal) changes in the parameter can lead to large changes in the performance metric. For these parameters the system will simply suggest fixed univariate limits, and remove the parameter from the multivariate model.

Once the parameter selection is complete, a multivariate model of historical reference data can be performed as described in section 2.2.2. For each new sample, the overall performance metric is calculated, along with each block contribution, and the block contribution for each univariate parameter. In addition, the equivalent limits for the univariate parameters in their original parameter space are calculated. After the calculation is complete, a model update is calculated (in the case of RPCA, by updating the parameters mean, standard deviation, and correlation structure as described in 2.2.2.2). This model update allows the multivariate model to provide dynamic limits in both

the multivariate space and specific limits in the univariate original parameter space.

For each node in the parameter hierarchy, the performance metric is compared to the current warning and fault limit. For PCA, the performance metric is scaled so that a value above 1.0 is a warning (outside the 99.5% confidence limit) and a value 10 times that represents a very serious fault.

Due to the multivariate nature of the performance metric, a parent node within the parameter hierarchy may have a fault when none of its child parameters have a fault and also may not have a fault even when all the child parameters have a fault. An example of a two parameter block contribution appears in Figure 4.2. In this example there are four classes of points; points that

- fail both univariate and multivariate limits,
- pass univariate and multivariate,
- pass univariate but fail multivariate limits, and
- fail univariate but pass multivariate limits.

4.3 Charting overview

This section describes the three main charts that best visualize results from monitoring a set of parameters with univariate and multivariate fault

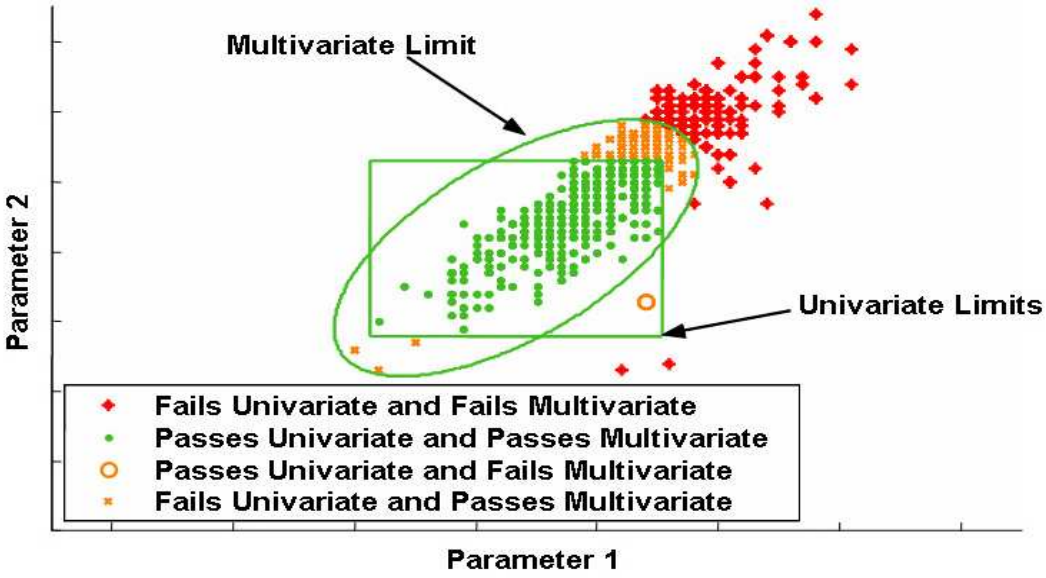


Figure 4.2: Multivariate and univariate limits on a 2 parameter block[12]

detection. The first section describes the most basic single parameter chart. The next section describes highest level overall performance metric chart. The third section describes the middle layer visualization of block level contribution plots.

4.3.1 Single parameter charts

The most basic chart in the visualization system is the parameter chart. This chart view is very similar to an SPC chart and provides the history of a parameter for a certain historical period of time. Along with the historical time series, all the univariate limits for a certain parameter are overlaid. An example parameter chart is shown in Figure 4.3. In this chart there are static

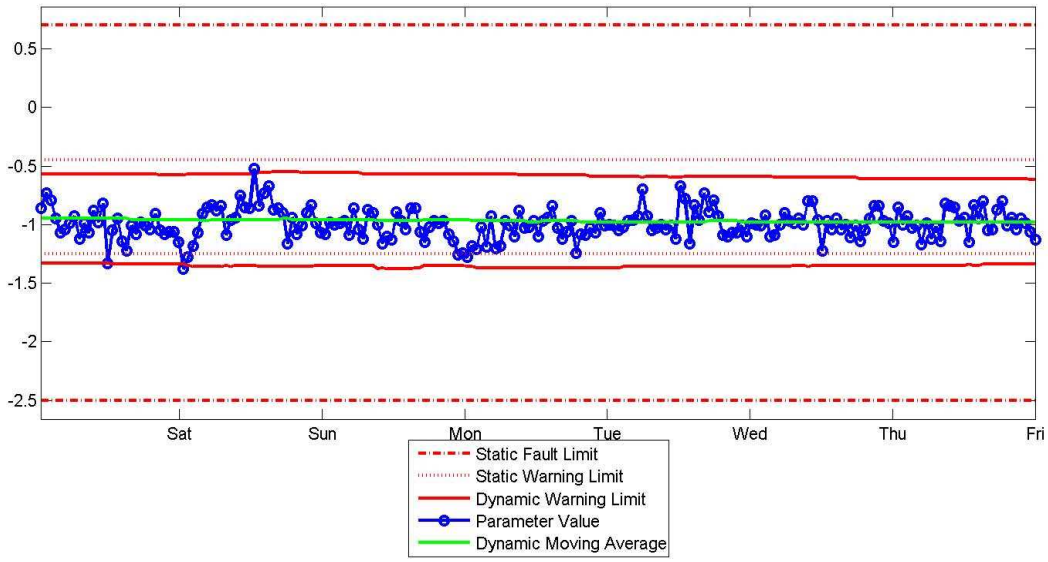


Figure 4.3: Example parameter chart of normal data

warning and fault limits, dynamic warning limits, and the dynamic moving average plotted to give the user a reference to the quality of the historical data. In the case of the parameter in Figure 4.3, it is clear that the parameter is well within its normal limits, and there is little or no change in the mean and standard deviation over the time period.

In a second example of a parameter chart in Figure 4.4, an excursion is identified that clearly violated both the static and the dynamic warning limits, but is not a major fault by either the static or dynamic fault limits. Also in this figure, there is a consistent offset between the static and dynamic limits suggesting a possible need to change the static limits.

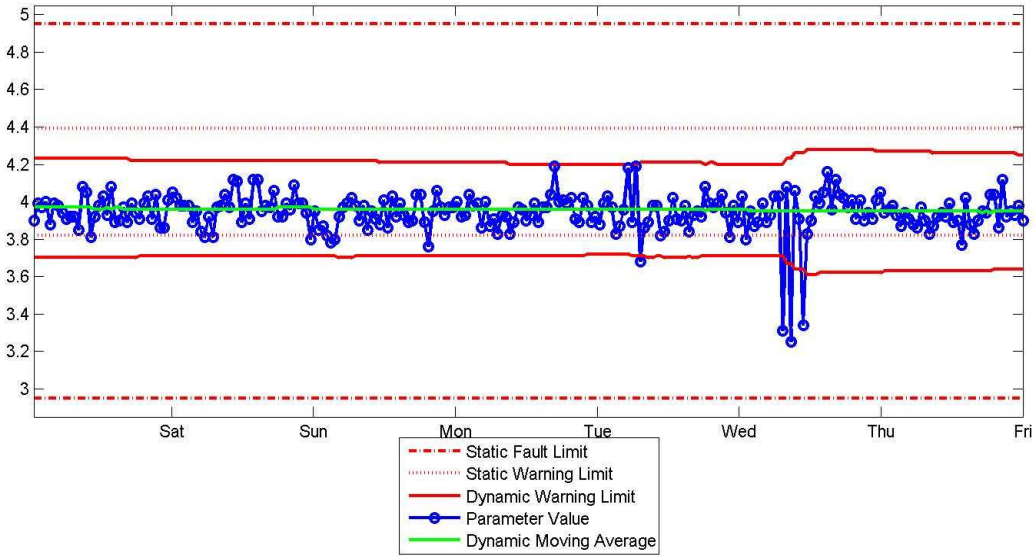


Figure 4.4: Example parameter chart with warning

4.3.2 Overall sample performance charts

The highest level chart for monitoring a particular multivariate analysis model is the overall performance chart. In this chart the value of the overall performance metric is plotted along with its corresponding limits for a historic time period, say the most recent seven days as illustrated in Figure 4.5. This chart gives the high level view of if there is any issue with any of the model input parameters or the internal correlation between the parameters. This chart gives a visual indication if there are any excursions and shows the magnitude of the excursions. Once excursions are identified, more information can be acquired by looking at block contributions and single parameter charts.

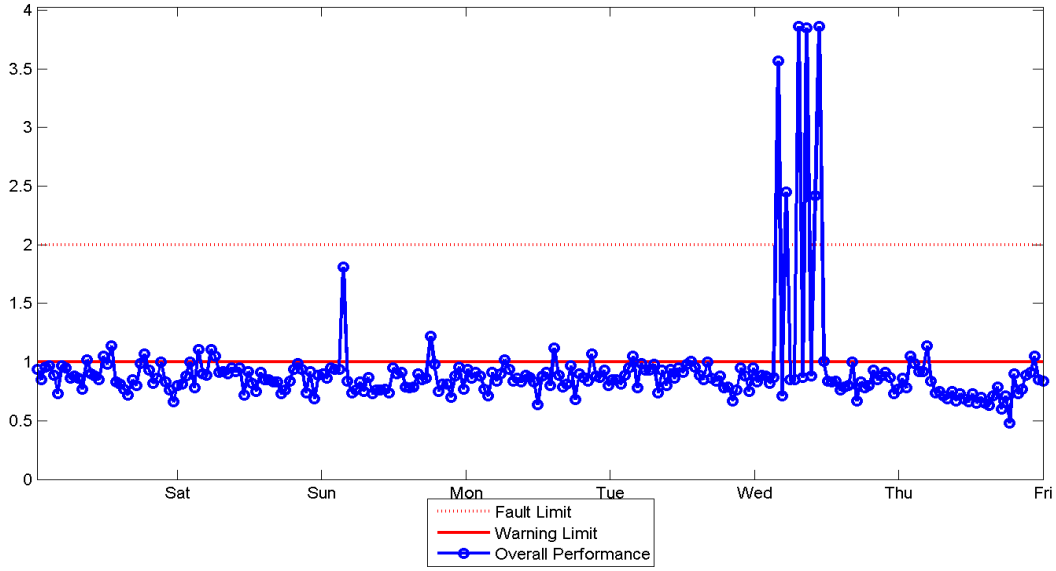


Figure 4.5: Example overall performance chart with fault and warning limits

4.3.3 Contribution charts

The simplest way to display the contribution information is to plot a bar chart for a single sample. In this chart there is a bar for each block contribution parameter on each level of the blocking structure. Since the multivariate analysis metrics are scaled to a common warning limit of 1.0, this line can be added (along with other fault limits if necessary) to quickly identify the parameters contributing to the excursion. An example of a contribution bar chart is displayed in Figure 4.6 for a single sample.

In order to provide this same information, but observe these contributions over a window of multiple samples, a contribution symbol chart can be displayed as in Figure 4.7. This chart has the samples on the x-axis sim-

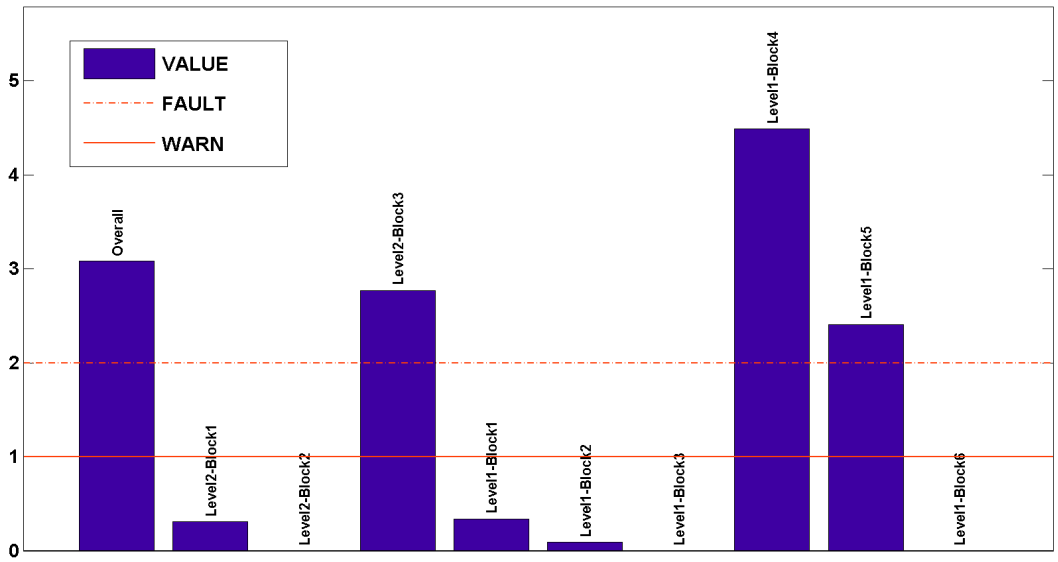


Figure 4.6: Example contribution bar chart with fault and warning limits

ilar to the overall sample chart, then has a row on the y-axis for each block contribution. On the chart there is a symbol representing if the sample is normal, a warning, or a fault. This contribution symbol chart allows a very quick overview of the faults identified in a window of time and which blocks contributed to the excursion. The chart in Figure 4.7 shows the results for the blocks as described in Section 4.2.1.

4.4 Reports for navigation

In order to make the charts in the previous section useful for an engineer to quickly diagnose a particular problem or see a periodic issue with a particular parameter, summary reports are provided. The sample summary

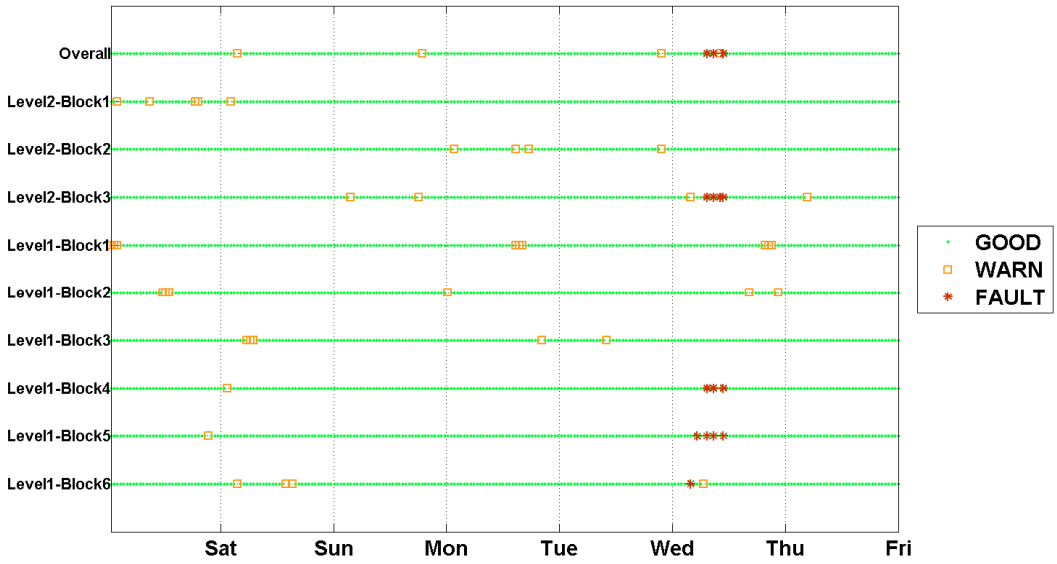
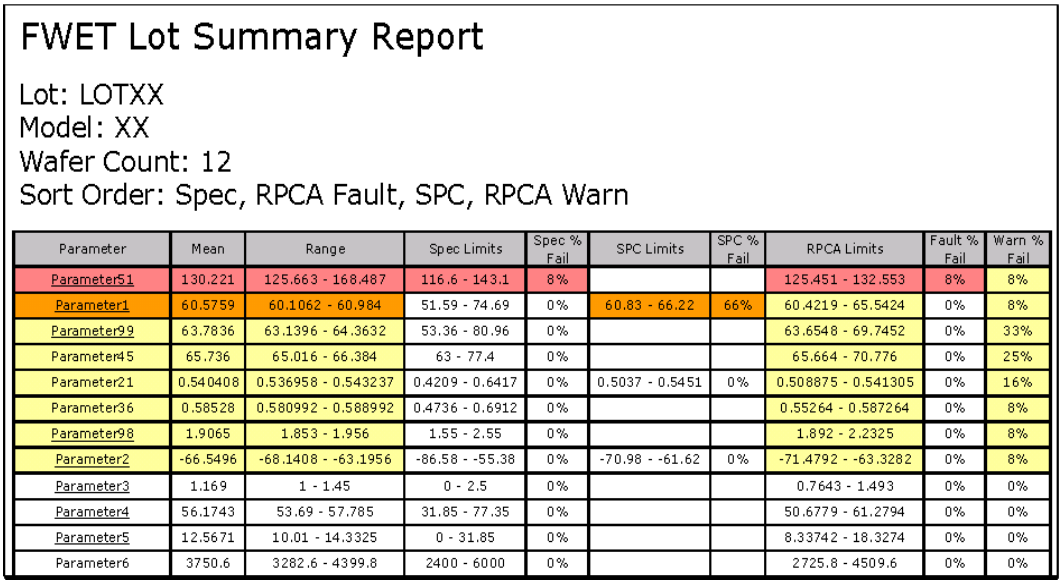


Figure 4.7: Example symbol block contribution plot with fault and warning indications

report merges the results for a group of samples (such as wafers into their parent lot in the semiconductor industry). This report summaries the univariate parameter checks and visually indicates the problem parameters within the group. The period summary report takes a moving window of the most recent data and counts the number of faults for each parameter and each block contribution. It summarizes the results such that the most extreme and most frequent violations are highlighted for the monitoring engineer. In both of these reports direct links are provided to the corresponding charts to give the engineer a clear picture of the data behind the summary statistics.

4.4.1 Sample summary reports

Sample summary reports summarize individual parameter excursions for a single sample or a group of samples. This report is used when there is an issue with the particular sample, and the engineer would like to quickly examine which parameters were abnormal. For each parameter the univariate mean and range are displayed along with the limits for each check. The rows are sorted so that the most severe faults are displayed first, followed by the warnings, and then the normal parameters. The parameter values and the checks that are in violation are highlighted. Figure 4.8 shows the first twelve rows of a sample report of a model with 100 parameters. This report summarizes the twelve samples that were in a specific group (LotXX). Parameter 51 had one wafer (8% of the group) that violated specification limits, the RPCA fault limits, and the RPCA warning limits. Since this was the only specification violation, this parameter was sorted to the very top of the sample summary report and colored red. The next parameter had eight samples (66% of the group) that failed the configured SPC limit and one that failed the RPCA warning limit. This parameter is then sorted as the second row because SPC limits were given preference over RPCA warnings and is colored orange to indicate an SPC violation. The next six parameters are colored yellow to indicate a RPCA warning violation in at least one sample and are sorted in descending number of samples that had a violation. The rest of the report (the four remaining lines and the other 88 lines that were not displayed) include the parameters that have no violation and are simply sorted by pa-



FWET Lot Summary Report

Lot: LOTXX
Model: XX
Wafer Count: 12
Sort Order: Spec, RPCA Fault, SPC, RPCA Warn

Parameter	Mean	Range	Spec Limits	Spec % Fail	SPC Limits	SPC % Fail	RPCA Limits	Fault % Fail	Warn % Fail
Parameter51	130.221	125.663 - 168.487	116.6 - 143.1	8%			125.451 - 132.553	8%	8%
Parameter1	60.5759	60.1062 - 60.984	51.59 - 74.69	0%	60.83 - 66.22	66%	60.4219 - 65.5424	0%	8%
Parameter99	63.7836	63.1396 - 64.3632	53.36 - 80.96	0%			63.6548 - 69.7452	0%	33%
Parameter45	65.736	65.016 - 66.384	63 - 77.4	0%			65.664 - 70.776	0%	25%
Parameter21	0.540408	0.536958 - 0.543237	0.4209 - 0.6417	0%	0.5037 - 0.5451	0%	0.508875 - 0.541305	0%	16%
Parameter36	0.58528	0.580992 - 0.588992	0.4736 - 0.6912	0%			0.55264 - 0.587264	0%	8%
Parameter98	1.9065	1.853 - 1.956	1.55 - 2.55	0%			1.892 - 2.2325	0%	8%
Parameter2	-66.5496	-68.1408 - -63.1956	-86.58 - -55.38	0%	-70.98 - -61.62	0%	-71.4792 - -63.3282	0%	8%
Parameter3	1.169	1 - 1.45	0 - 2.5	0%			0.7643 - 1.493	0%	0%
Parameter4	56.1743	53.69 - 57.785	31.85 - 77.35	0%			50.6779 - 61.2794	0%	0%
Parameter5	12.5671	10.01 - 14.3325	0 - 31.85	0%			8.33742 - 18.3274	0%	0%
Parameter6	3750.6	3282.6 - 4399.8	2400 - 6000	0%			2725.8 - 4509.6	0%	0%

Figure 4.8: Example sample summary report

rameter name and included for completeness. Each line in the report has a link to the parameter chart (as described in section 4.3.1) corresponding to the parameter and a start and end time (a number of days before and after the sample was obtained). This report provides a quick health of group of samples across all parameters and provides links to understand the univariate parameter violations.

4.4.2 Periodic summary reports

The period summary report attempts take all data from a recent window of results and identify the most severe and frequent excursions that are caught from the multivariate blocks and the univariate parameters. The re-

port is created by querying the results for a single model from a recent time period such as the last week's worth of data. Then the number of faults that occurred for each parameter and block contribution are counted for each limit check. Instead of simply sorting each parameter by the number of faults, the block hierarchy is sorted to highlight the blocks that have the most severe and frequent excursions. First the top level of the hierarchy is sorted, then the child nodes within the next level are sorted. This process is repeated until the parameter level is sorted (the lowest level nodes of the parameter hierarchy).

When nodes are sorted there is an order of precedence between the different limit checks. The order used from most severe to least severe is specification faults, RPCA faults, SPC limit violations, and finally RPCA warnings. If two nodes have the same number of specification faults, for example, then they would be sorted based on the number of RPCA faults, and this process would be repeated until the order of the rows are resolved.

Once the entire parameter hierarchy tree is sorted, rows are created within the report starting with the highest level within the tree for the node that sorted highest. Then a row is created for its highest child, and again for its highest child, until the parameter level is reach. Once all the parameters in this one node are added, then the next rows are filled in with all the blocks and their children, essentially flattening the sorted tree into rows within a table as illustrated in Figure 4.9. In each row of the table, the node name is printed under the appropriate node level (in this case Overall, Layer/Device, Scrap Code, or Parameter), then the count of each type of violation that occurred

Model Hierarchy					Wafer Counts					
Overall	Layer Device	Scrap Code	Parameter	Parameter Description	RPCA Warn	RPCA Fault	SPC Warn	Spec Fault	Sanity Fault	Total
Overall					16	3	152	7	27	166
	Diode				6	1	1	3	0	7
		COREDIODE			4	1	.	3	0	5
			COREDBV	Core Drain Diode	3	1	.	3	0	3
			CORESBV	Core Source Diode	0	0	.	0	0	0
		DIODE			7	1	1	1	0	7
			HVP+SDBV	Hi-Volt P+S/D Diode	9	1	1	1	0	9
			HVN+SDBV	Hi-Volt N+S/D Diode	3	1	.	1	0	3
			LVNBV	Lo-Volt N Diode	2	1	.	0	0	2
			LVPBV	Lo-Volt P Diode	1	1	.	0	0	1
	Core				5	1	29	2	27	56
		CORETRAN			3	1	28	1	27	54
			COREIDsat	Core Saturated Drain Current	3	0	.	0	0	3
			COREVTlin	Core Linear Vt	2	0	.	0	0	2

Figure 4.9: Example periodic summary report

during the window of time the report covers.

This report accomplishes the goals of identifying the highest level blocks that have the most severe and frequent faults by putting together the fault counting, block sorting, and the fault severity ordering. In Figure 4.9 the "Diode" Layer/Device is sorted higher than the "Core" Layer/Device because the "Diode" block contained three specification violations compared to the two specification violations in the "Core" block. Within the "Diode" Layer/Device, the "COREDIODE" Scrap Code comes before the "DIODE" Scrap Code also because it has more specification violations. Within the "DIODE" Scrap Code, the first two parameter each have one specification violation, each have one RPCA fault, neither have SPC checks, but the first has nine RPCA warnings

so it was placed first in the sorting.

4.4.3 Navigation

In order to provide meaningful navigation for a factory wide deployment of multivariate monitoring based on the system that has be described in this chapter, the final step was to give every engineer a common starting point. Every day a periodic summary report is created for every model that is online and has results. Then the top "Overall" line from each report is taken to create a model summary list. This list is also sorted similarly by severity and frequency and sent to every engineer in an email (and posted on a common website). From this list the engineer can click on the model and see the individual periodic summary report for the last week.

From the periodic summary report the user can drill down to any of the charting types described in Section 4.3. The "Overall" column header is linked to the overall sample performance chart to quickly view the value of the performance metric over the period of the report. The block contribution nodes in the report such as "Diode", "Core", "COREDIODE", "CORETRAN", etc. are links to the symbol block contribution plots which contain rows within the graph of all the children of the selected node and have a time range of the period of the report. The parameter names in the final level of the hierarchy link directly to the univariate parameter chart with all the limits overlaid for the period of the report.

From any of these three types of charts, the user can also select a data

point which corresponds to a particular sample. From the sample, a link is provided to the sample report to review all the other parameters for the sample group.

These navigation options allow an engineer to become completely empowered to discover the excursions within a huge data set and understand them very quickly. Dozens of models each with hundreds of parameters can be summarized into detailed information, and for the important parameters, the engineer can drill down and learn the root causes of concern. Unlike a simple SPC system, where individual charts (and hence each parameter) is monitored individually, this system allows all the parameters to be integrated and sorted. This system takes advantage of static limits along with dynamic limits, warning levels along with fault levels, and multivariate FDC metrics to sort parameter blocks. This sorted result leads the engineer directly to the most extreme and most frequent excursions within the data set and puts the important data right at their fingertips.

4.5 Results for electrical parameter monitoring

This software architecture for monitoring a multivariate FDC method was implemented in Spansion's Fab25 in Austin, TX to monitor final electrical wafer test results. The system was originally implemented with results from the RPCA algorithm based on the work by Cherry[17]. This implementation was then extended to also use the KNN method as described in Chapter 3.

Near the end of the manufacturing line for semiconductors, many cir-

cuits are measured at many sites on a selection of wafers. For wafers that are measured, over 700 circuit measurements are made at each selected site. The manufacturing system summarizes the site measurements into a single measurement for each parameter for each measured wafer.

Prior to this new monitoring application, the monitoring engineers were only able to monitor 40 parameters (of the 700) per device with the SPC system. The SPC system would require maintaining SPC limits for each parameter for each device and responding to each excursion for which they received an email notification. Working with the team responsible for monitoring the system, parameter hierarchies with meaningful block contribution nodes were built. In most cases, around 150 parameters were included in the parameter hierarchies, which is a significant improvement in coverage from the SPC system. In all cases the same parameters and hierarchy were shared by all the devices within the same technology. This allowed additional summary reports that included counts for the same parameters and nodes across multiple devices within the same technology.

For each device, a RPCA model was built using an offline utility and placed into the system. The system was integrated with the factory to get a daily update of SPC, physical significance, and specification limits for each device to keep these up to date.

This system is currently running on multiple electrical testing process steps with dozens of device models. Now that the system is installed, verified, and demonstrated, about 20 engineers have signed up to get the daily reports

emailed to their inbox and others routinely consult the web page for results information.

4.5.1 Application results

Using this technique to monitor the results of electrical test data proved very successful. It provided significant information to engineers, and made this information much easier to access.

The combination of fixed and adapting limits made the system more robust. Generally the dynamic RPCA warning limits are tighter than the manual fixed SPC limits as illustrated in Figure 4.10. In this case the RPCA limits provide earlier detection that something has shifted, but also change with the data, so they require no maintenance as compared to the SPC limits. Also, occasionally the RPCA limits are outside the manual fixed warning limits as illustrated in Figure 4.11. In this case the SPC limits provide warnings before the parameters move past known fixed boundaries. Together both types of limits maintain the security of not moving past a fixed boundary while also providing notification of tighter, maintenance-free limits that reflect the changing mean and standard deviation of the data.

The combination of two severity levels of each limit type also adds information in the results and robustness to the model updates. Observing the application, in general the RPCA fault alarms correlated highly with the specification violation alarms. Both fault indicators from the dynamic RPCA model and the static factory limits would both identify extreme violations.

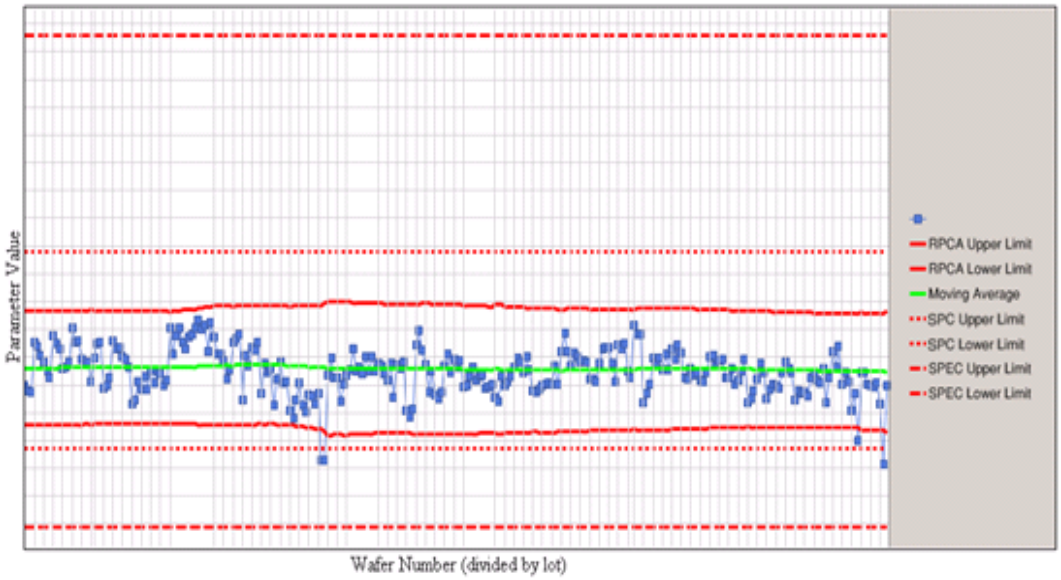


Figure 4.10: Typical parameter chart with RPCA limits tighter than SPC limits

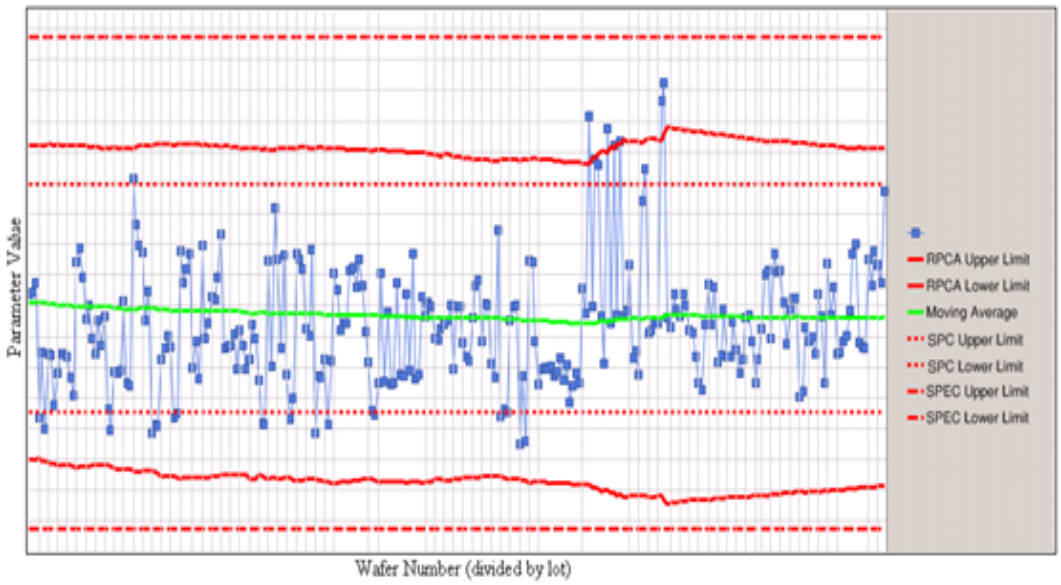


Figure 4.11: Parameter chart with tighter SPC limits than RPCA limits

Occasionally the RPCA fault alarm would trigger even with data inside the manual specification limit providing early indication of an extreme excursion. Having two severity levels allowed the system to point out the gross excursions in the model reports. This was valuable in quickly pointing out the worse samples in the period summary reports and in identifying a sample list for the previous day that engineers can go troubleshoot. The identification of gross errors was also utilized in the RPCA model updating. Samples with gross errors in overall performance were not used to update the model. Also parameters that had gross errors in parameter contribution performance were reconstructed again before model updates were performed. This kept the models updates robust, so that parameter estimates did not become unstable, and kept the acceptable region within the model from growing unreasonably large.

With a robust underlying multivariate monitor for each device combined with the static factory limits and two levels of alarms, the reporting infrastructure was successful. Many engineers use the system because all the alarm information is together in one place and they feel they have access to valuable information. They also realized the value and importance of the block contribution naming as they recognized the groups they had specified early in the model building phase. Frequently the monitoring application was pointing out exactly the parameters in the devices that the engineers were getting notified about from the factory operators.

4.6 Conclusions

In this chapter a novel monitoring infrastructure has been described that can be used to monitor the results of any multivariate analysis method along with its application to electrical parameter monitoring. This infrastructure took advantage of a hierarchical tree structure for block contributions with labels provided by the monitoring engineers. These custom block labels were critical to communicating effectively the results of very large data sets. The reporting system was effective at highlighting the most severe and most frequent problem areas within the data sets. The incorporation of dynamic and static limits on parameters helped the monitoring system stay effective and robust for data sets with realistic factory variations. Integrating this application into the existing factory system for fixed limits and using self updating models made the application maintenance-free.

For the specific application that was developed, the reduction in limit maintenance allowed monitoring engineers expand the scope of the number of parameters they could continuously monitor with their limited time. Overall, engineers were very engaged with this application from its first roll-out, and the application highlighted important problems in the data sets that were being worked on. It also provides some assistance in visually triggering engineers to adjust the SPC limits or specification limits that may no longer be accurate. Overall this investigation showed that monitoring of semiconductor electrical test data is feasible with the developed analysis tools and visualization system.

Chapter 5

Semiconductor Process Tool Analysis

In this chapter work is presented relating the application of multivariate analysis to monitoring semiconductor process tools. In the first section an alternative approach is proposed for updating a PCA model to monitor a non-stationary process. To complement monitoring with adaptive models, two metrics are presented to monitor the movement of process variables between two points in time when a PCA model is constantly being adapted. Results for these methods are presented on an oxide etch process.

The second section of this chapter describes the extension of the KNN algorithm to monitoring batch processes. Two separate methods for applying the algorithm are presented. In addition, the two methods are compared along with PCA, MPCA, TLD, PARAFAC, and PARAFAC2 on an induced fault experiment available in the literature[112, 113] from a metal etch process.

5.1 Monitoring dynamic processes

Monitoring most semiconductor process tools historically has been a very challenging task. In many process tools, especially in plasma etch and other plasma-based processes, the chambers undergo frequent maintenance

cycles. Each cycle begins with a cleaned and conditioned chamber, and from this point the tool process variables may drift and vary until the chamber is shut down again. When the tool is idle, key parts are cleaned or replaced. This process is repeated in order to avoid incorrectly processing wafers or creating defects on the wafer surface.

In order to monitor such a drifting system, a PCA model can be used as described in Section 2.2.2. When a static PCA model is applied to a drifting system, eventually the process variables will drift beyond the model limits, and every new run will be considered an excursion. Existing methods that attempt to solve this issue are discussed in Section 2.2.2.2 and in Wold[116], Gallagher *et al.*[38], and Li *et al.*[64]. In this work, a method for model updates to provide local fault detection is proposed that simply updates the mean and standard deviation coefficients used to center and scale the input data after each run. This update method is computationally simpler and easier to implement than periodically building a new PCA model[38] or using the recursive PCA technique[64].

In this method the centering and scaling coefficients are updated using a recursive scheme. The means of each summary statistic are updated with an EWMA filter

$$\bar{x}_n = \lambda \bar{x}_{n-1} + (1 - \lambda) x_n \tag{5.1}$$

where $\bar{x}_n$ is the estimated mean after n points, x_n is the new data point, and λ is the filter coefficient. In this investigation λ was set to 0.92 in order to

produce very slow adaptations. The scaling coefficient for each variable summary statistic is updated with the following exact recursive standard deviation formula

$$\sigma_n = \sqrt{\frac{n-2}{n-1}\sigma_{n-1}^2 + \frac{1}{n}(x_n - \bar{x}_{n-1})^2} \tag{5.2}$$

where σ_n is the estimated standard deviation, $\bar{x}_{n-1}$ is used instead of the actual previous mean value, and n is the number of points that have been used previously in the standard deviation calculation. In order to create a recursive filter from this expression, a fixed value of n of 500 is used representing a long window length.

When a drifting process is monitored by an adapting model, there is no indication of the variables contributing to the drift from the multivariate model performance metrics. In order to communicate the variables which are changing between two reference runs, the relative change in the centering and scaling coefficient for each variable can be calculated between two runs. This calculation is performed by subtracting the mean estimate at an initial run from the mean estimate at a final run for each variable, then scaling each difference by the corresponding standard deviation used for scaling in the initial run. The mean movement metric $M_{\bar{X}}$ is calculated between run a and run b as

$$M_{\bar{X}}|_{a,b} = \left| \frac{\bar{X}_b - \bar{X}_a}{\sigma_a} \right|. \tag{5.3}$$

The scaling movement metric M_σ is calculated as the difference in standard

deviations scaled with the mean used for centering that step statistic

$$M_{\sigma}|_{a,b} = \left| \frac{\sigma_b - \sigma_a}{\bar{X}_a} \right|. \tag{5.4}$$

The components of these movement metrics are then displayed in a Pareto or contribution chart to identify the variables that exhibited the largest relative change in mean and standard deviation during the period. A typical fault contribution plot indicates the local deviation from a particular run to the model at the current time. These movement metric contributions indicate the variables with changes over a period of time in which the model has adapted. The movement metrics are an important supplement to the typical contribution plot. In the periodic summary reports discussed in Section 4.4.2, in each process variable row the mean and standard deviation movement metric can be provided to indicate if the variable has changed significantly within the reporting period. This addition allows the system to highlight that process variables are changing, although they may not be changing fast enough to get detected by the performance metrics of the adapting model.

5.1.1 Plasma etch process monitoring

The plasma etch process is used in semiconductor manufacturing to precisely and selectively remove material from the wafer surface. Prior to the etch process, a uniform film of photoresist is placed on the wafer. This film is then exposed with a precise pattern with a lithography tool, developed, and then the exposed material is removed. This process leaves behind gaps in the photoresist, which expose the sites on the surface to be etched. This work

focuses on the etching of silicon oxide for trenches and vias (wiring lines and vertical connections between wiring layers within the chip). The etch process uses plasma above the wafer surface to generate high energy ions that are accelerated through the plasma sheath to the surface of the wafer. Due to the voltage potential of the sheath being perpendicular to the wafer surface, the high energy ions that strike the surface in areas that are not protected by photoresist and then react to create essentially vertical notches in the surface. These notches grow until either the plasma is extinguished or the reaction at the surface is quenched. The goal of the oxide etch process is to etch a notch in the silicon oxide material and stop the etch before it proceeds into the next material while making sure to clear the oxide in all notches throughout the wafer without much variation. The use of chemically selective etchant gases such as fluorinated and chlorinated species provide a tool to etch one material at much higher rates than the underlying material, stopping the etch propagation into the new layer.

The industrial plasma etcher is a modular production tool. A cassette of wafers, usually holding 25 wafers, is loaded into the tool. Then the wafers are individually removed from the cassette, aligned, and placed into the process chamber. Within the process chamber a preprogrammed recipe of many process steps is executed. Usually the recipe consists of sets of steps beginning with a stabilization step, a powered etch step, and then a powered over-etch step. The stabilization step starts the flow of the process gases into the chamber an continues until the pressure and temperature within the

chamber stabilize. Once stabilized, the powered step begins and the power is turned on. Quickly the matching network adjusts the reactivity of the delivered power, and after this stabilizes, the rest of the step is essentially a steady state process with constant power, temperature, pressure, and gas flow rates. This step lasts for a fixed time or until an optical sensor detects the endpoint by a change in composition within the chamber. After this endpoint detection, there are a few seconds of "over-etch" time in the final step to make sure the entire wafer has uniformly reached endpoint. These three step types can be repeated within a single recipe to etch different materials on the surface of the wafer with different gases and power settings.

5.1.2 Adaptation and movement metric results

Process data were acquired from Tokyo Electron Unity II DRM CCP chambers that operate an oxide etch with C_4F_8/CO/Ar + O_2 chemistry. This tool operates in a batch mode with a fixed process recipe for each lot. Typically a single recipe is used lot after lot for a particular process step in the manufacture of a device. The same tool frequently is used for many different device layers and steps, but for each process step, the recipe is the same. Data collected include the chamber pressure, applied power, various temperatures and process gas flow rates, and other variables relating to the power and temperature control. A complete list of variables used in this investigation is given in Table 5.1. The process recipe used in this study has three main steps: a photoresist cleaning, the main etch step, and a photoresist stripping step. The

scope of this study is limited to the main etch step, but there may be significant diagnostic value in monitoring the other steps as well. For each process step, the mean and standard deviation of each tool variable were calculated from roughly 160 samples for each wafer. The beginning portion of the step where the RF power increases was ignored in these statistical calculations in an attempt to remove the variation due to matching network when the power is turned on. The resulting mean and standard deviation summary statistics were used to monitor the chamber on a wafer-by-wafer basis. The mean summary statistics for variables with direct control (such as the gas flows) were omitted from the model, and only the step standard deviations were used.

In this investigation a principal component analysis model was created for the first 500 wafers using the same recipe in a single chamber. The standard PCA methods implemented in MATLAB® were used with mean centering and unit variance scaling as described in [52]. Also, the standard Q residuals (SPE) and Q contributions were calculated using the Eigenvector Research® PLS_Toolbox®.

The PCA model built from the first 500 wafers in the first chamber was applied to all 3200 wafers from this chamber. Figure 5.1 shows that the resulting Q statistic exceeds the 95% confidence limit in the model in less than 250 wafers after the model was built and never returns to below that level. In addition, distinct outliers and distinct step-like changes are apparent. When the centering and scaling constants are adapted, the Q statistic is much more stable across all the remaining wafers. Figure 5.2 shows that after the first

Table 5.1: Tool Variable Descriptions

Area	Variable	Descriptions
Gas flow and Pressure	PRESSURE	Chamber Pressure
	APC	Throttle Valve Angle
	Ar	Ar Flow Rate
	C4F8	C_4F_8 Flow Rate
	CO	CO Flow Rate
Power and Matching	RF-FORWARD-LO	Lower Electrode Power
	C1-POSITION-LO	Matching Network Capacitor 1
	C2-POSITION-LO	Matching Network Capacitor 2
	ESC-CURRENT	Electrostatic Chuck Current
	ESC-VOLTAGE	Electrostatic Chuck Voltage
	MAGNITUDE	Matcher Magnitude
	PHASE	Matcher Phase
	RF-VDC-LO	Lower Electrode DC Voltage
	RF-VPP-LO	Lower E. Peak to Peak Voltage
Temperature and Cooling	LOWER-TEMP	Lower Electrode Temperature
	UPPER-TEMP	Upper Electrode Temperature
	WALL-TEMP	Wall Temperature
	COOL-GAS-FLOW1	He Edge Cooling Flow Rate
	COOL-GAS-FLOW2	He Center Flow Rate
	COOL-GAS-P1	He Edge Cooling Gas Pressure
	COOL-GAS-P2	He Center Cooling Gas Pressure

500 wafers, the data predominantly stay inside the same limit.

Wafer 1492 has the largest Q value in the adaptive case. The residual contribution plots for both the static and adaptive cases for this wafer as shown in Figure 5.3. This figure indicates that C1-POSITION-LO mean, RF-VPP-LO mean, LOWER-TEMP mean, and ESC-CURRENT mean are the extreme contributors. The arbitrarily scaled summary statistics for the ESC-CURRENT mean and the RF-VPP-LO mean are plotted in Figure 5.4. These

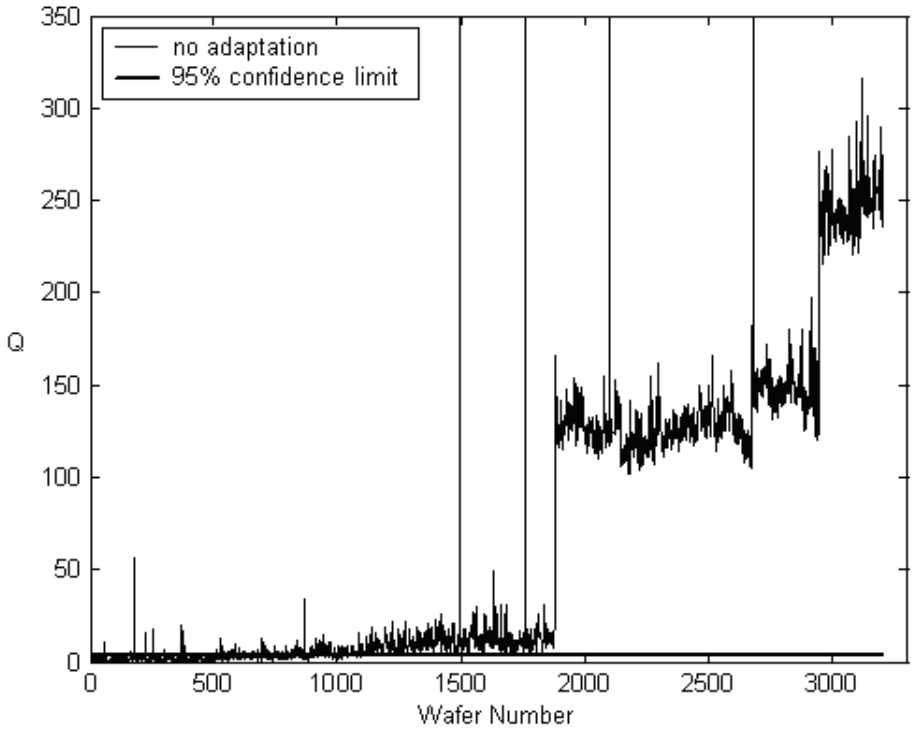


Figure 5.1: Calculated Q for static PCA model of first 500 wafers

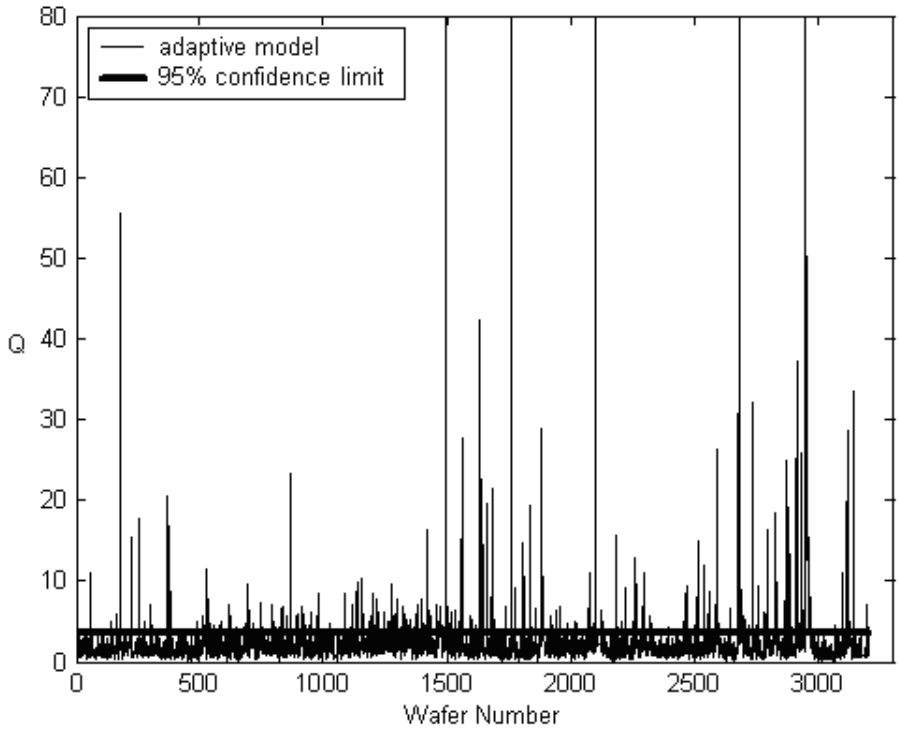


Figure 5.2: Calculated Q for PCA model with adaptive mean and centering

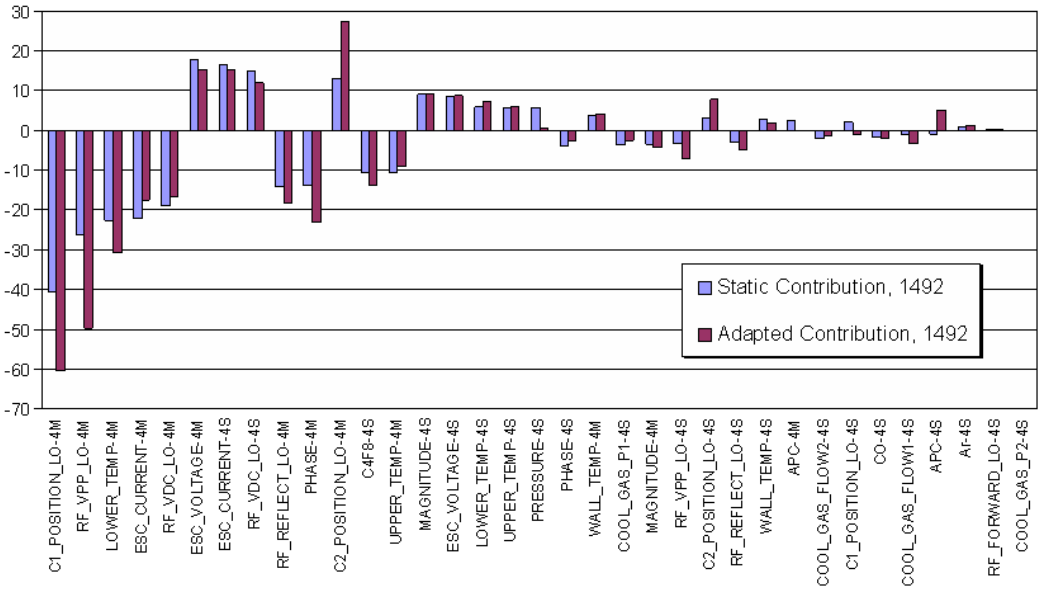


Figure 5.3: Q contributions for extreme outlier, wafer 1492

four variables account for the large spikes in the data at four points, which could indicate an issue with the matching system. This type of outlier is clear in both the static model and adaptive model Q charts, but only the adaptive case allows for a fixed limit for all time.

The next type of excursion that can be identified is a step-like change that is observed in the input summary data. In the static case such changes are clearly evident in the Q chart, although automating detection of these changes with a static limit would prove quite difficult. In the adaptive case there are only 4 periods where the Q statistic violates the confidence limit for more than five consecutive wafers (starting at wafers 1880, 2535, 2683,

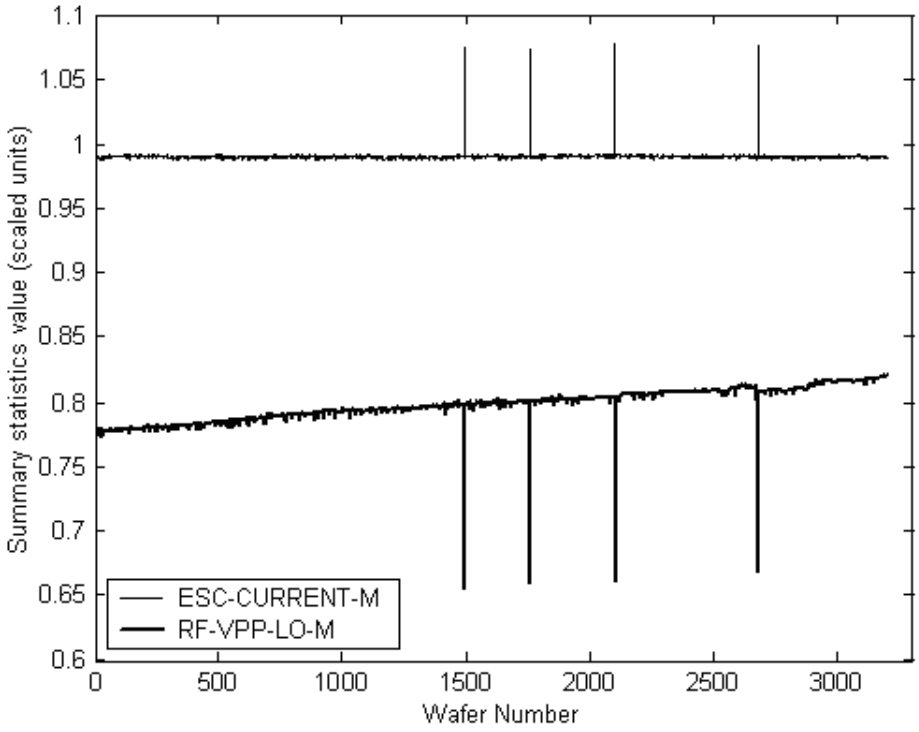


Figure 5.4: Summary statistics for process variables leading to the highest Q deviations for wafer 1492

and 2948). When the mean movement metric is calculated about each of these four periods (from the wafer before the period to the wafer after the period), the most extreme contribution values occur for wafers 1880 and 2946 on variables C1-POSITION-LO mean and LOWER-TEMP mean, respectively. The mean movement metric contributions for these two periods are shown in Figure 5.5. The arbitrarily scaled summary data for the two variables are displayed in Figure 5.6. The two major changes in the Q statistic seem to be dominated by these two variables. The shift in these variables may have been caused by a tool cleaning, where the replacing of key parts could have changed the electrical or heat transfer characteristics of the chamber. Although the temperature is regulated in the process chamber, this is performed only at the upper electrode and walls. The lower temperature is not controlled and could be affected by different materials or part configurations in the chamber. The fault contribution plots for the static case and the adaptive case for wafer 1880 both are dominated by the C1-POSITION-LO. For 2948, LOWER-TEMP is the dominant contribution in the adaptive case, but in the static case it is only slightly larger than the C1-POSITION-LO value (which does not change at this run).

After looking at the major changes over time for one chamber, the same model from the first 500 wafers is applied to a set of 800 wafers from a second chamber. Figure 5.7 shows that the plot of the Q statistic for the static model is many orders of magnitude greater than the confidence limit (around 4.0) for the model. Figure 5.8 shows the same model with adaptive means and

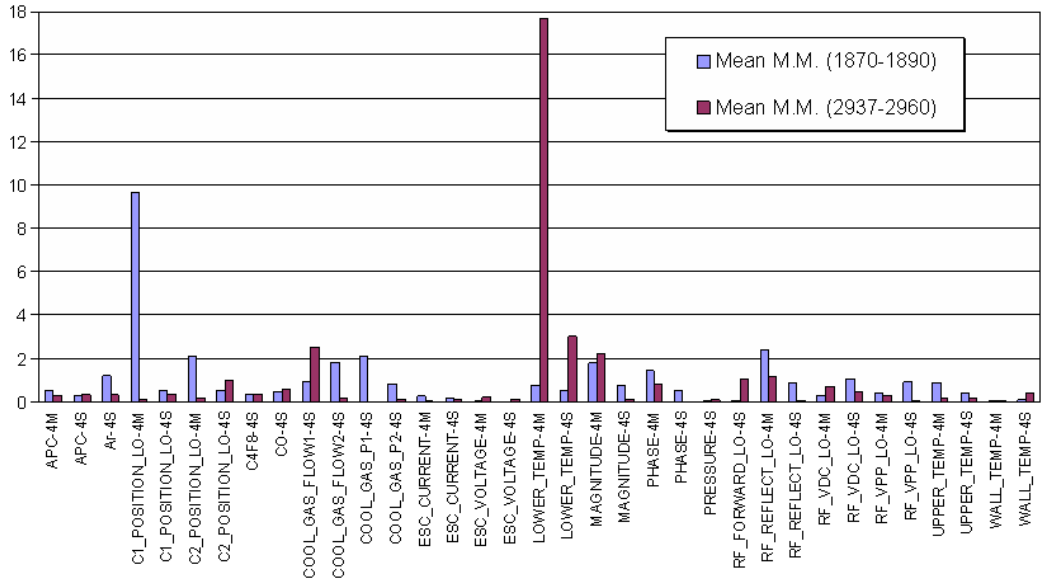


Figure 5.5: Mean movement metric contributions for the first chamber from wafers 1870-1890 and from wafers 2937-2960

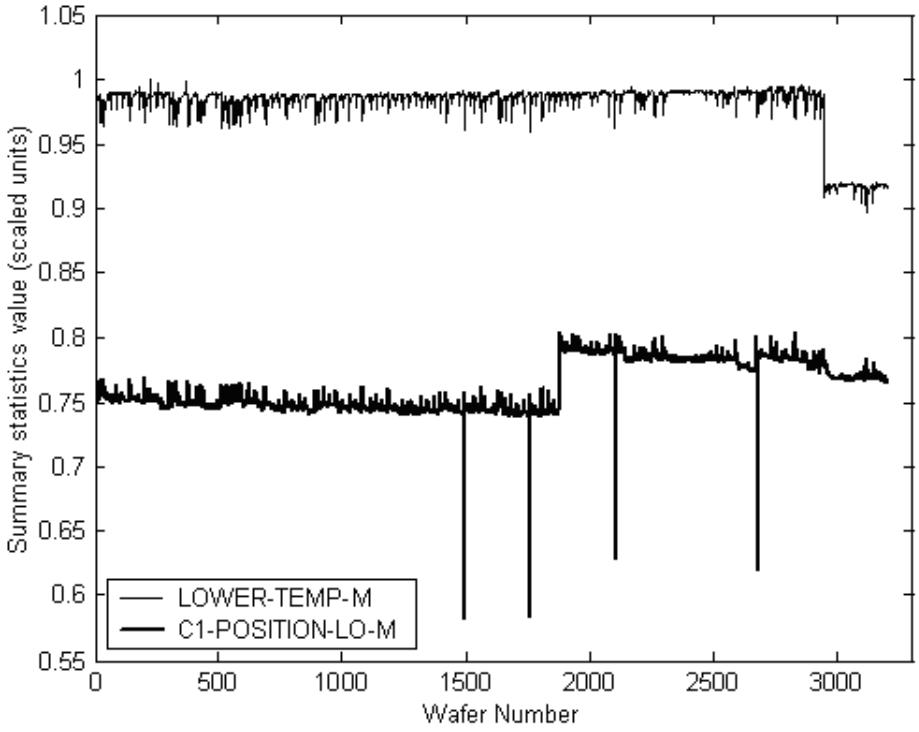


Figure 5.6: Process variable charts for largest mean movement metric contributors LOWER-TEMP-M and C1-POSITION-LO-M

centering applied. The performance metric value is below the model confidence limit after only 25 wafers (the typical load for a single cassette). After the initial adaptation period, the model is still able to indicate the same gross outliers, but also it can highlight other variations such as the region between wafer 445 and 455. With this same model applied to a second chamber, a contribution plot can again be used to identify the cause of the single point excursions. The fault contribution plot for the extreme excursion at wafer 422 is shown in Figure 5.9. The contribution plot based on the static model for this wafer compared to the contribution based on the adaptive scheme no longer point necessarily to the same contributors. The adaptive contributions most likely indicate the local faults in the second chamber. The contributions from the static model are a mix of the contributions to the excursion and to the difference between the operating conditions between the two different chambers.

In order to investigate the sudden shifts of the system, periods of consecutive violations were noted from the data in Figure 5.8. Three different regions had more than five consecutive points exceeding the confidence limit (occurring at wafers 1, 91, and 446). The movement metric for the first 22 wafers, where the model was adapting to the new chamber values, highlighted significant changes in RF-VPP-LO mean, ESC-VOLTAGE mean, C2-POSITION-LO mean, ESC-CURRENT mean, and RF-FORWARD-LO standard deviation, indicating that many of the electrical characteristics have offset between the two chambers. The period beginning with wafer 91 has two large spikes

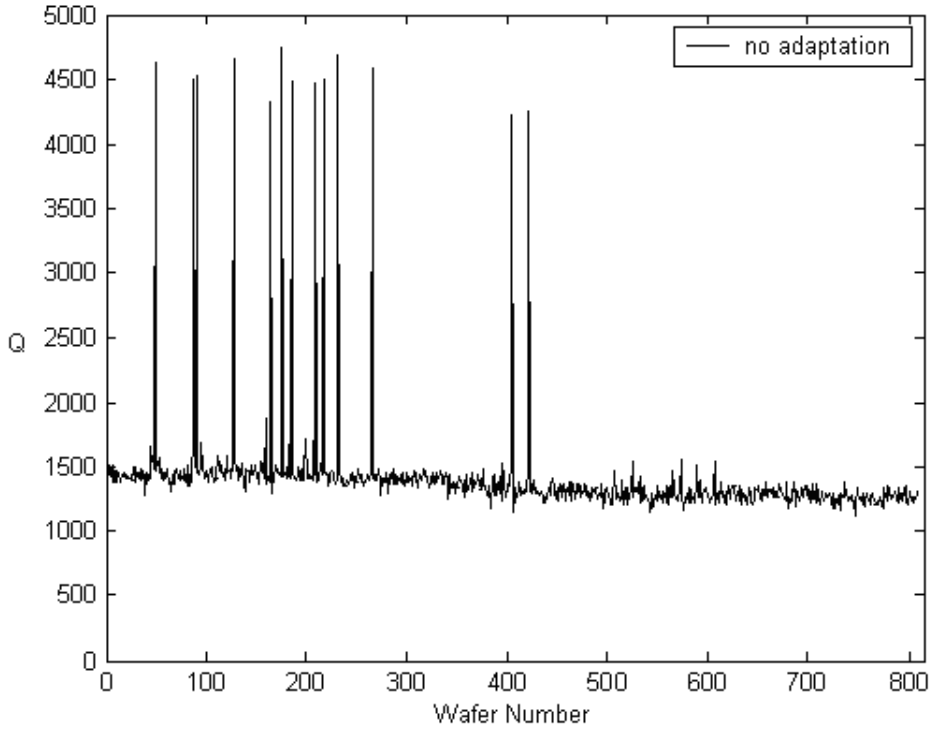


Figure 5.7: Calculated Q for initial model applied to second chamber in a static mode

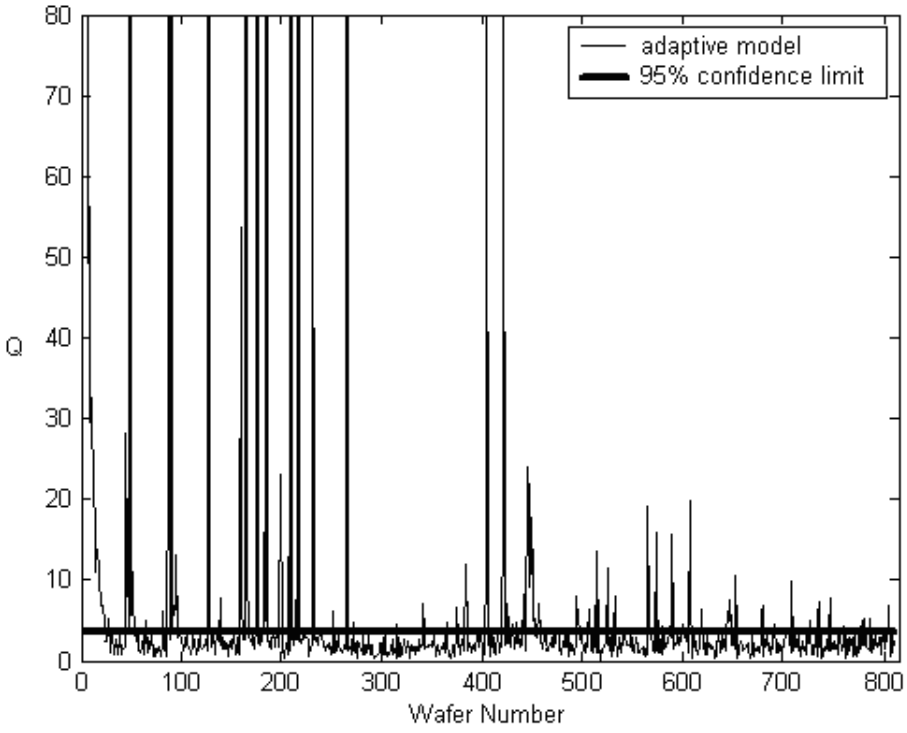


Figure 5.8: Calculated Q for initial model applied to second chamber with adaptive mean and centering

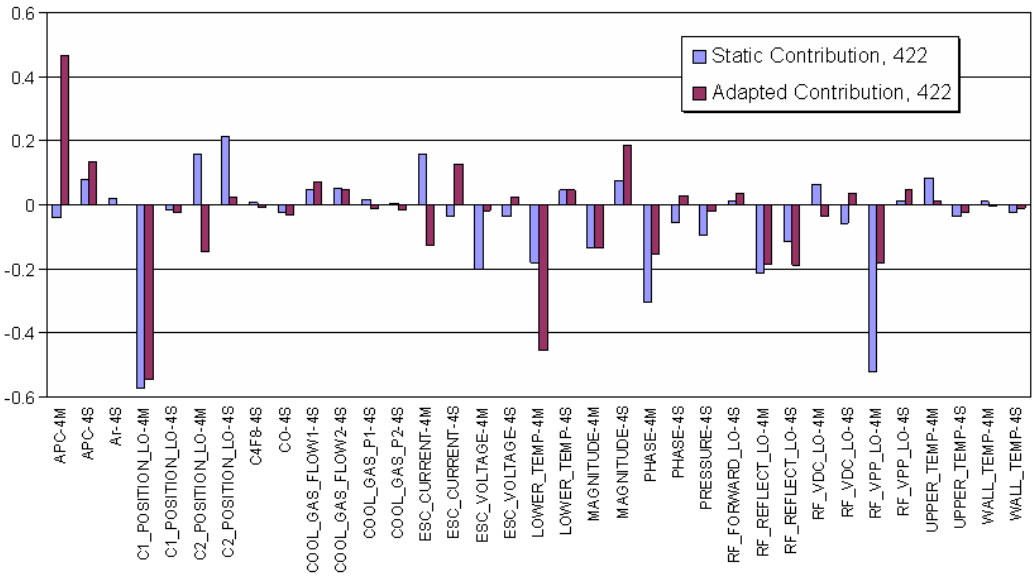


Figure 5.9: Q contributions for extreme outlier on second chamber, wafer 422

within five runs, causing the movement metric to identify the adaptation of the model to the outliers. In the final region starting at wafer 446, the metric contributions shown in Figure 5.10 points to a mean change in APC mean and COOL-GAS-FLOW1 standard deviation and points to a standard deviation change in COOL-GAS-FLOW1 standard deviation and the UPPER-TEMP standard deviation. The wafer summary data for these variables exhibit a clear jump at wafer 446 at this time as seen in Figure 5.11. Further analysis would be necessary to speculate on a type of problem that would be characterized by a shift in the throttle valve angle used to control pressure, in the variability in the helium flow used to control temperature on the lower electrode, and in the variability in the upper temperature.

5.1.3 Adaptation methodology compared

Spitzlsperger *et al.*[100] compared this proposed method for adaptive centering and scaling without correlation updates with a number of other fault detection methods. In their work, the authors similarly looked at a plasma etch tool utilizing at ArO_2C_4-F_8 chemistry. They analyzed the step means for only the RF system variables, the temperature variables, and the pressure control valve actuator. For multivariate monitoring in this study, the Hotelling's T^2 was used simply on the correlation matrix (and a PCA reduction was not performed).

Spitzlsperger *et al.* showed clearly that the etch process was not stable over a series of five wet cleans (over 10,000 runs). They compare the method

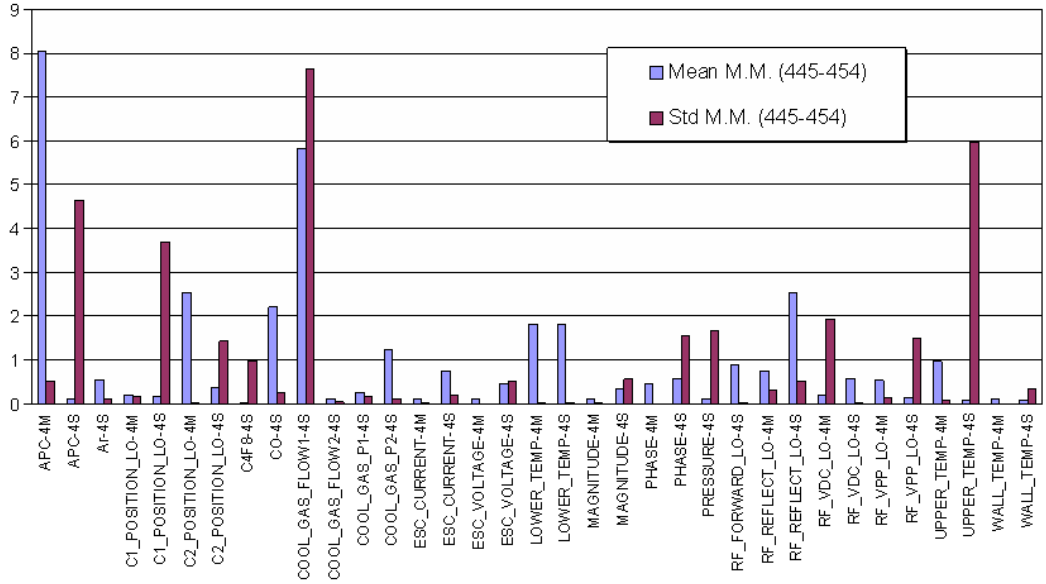


Figure 5.10: Mean and standard deviation movement metric contributions for the second chamber from wafers 445-454

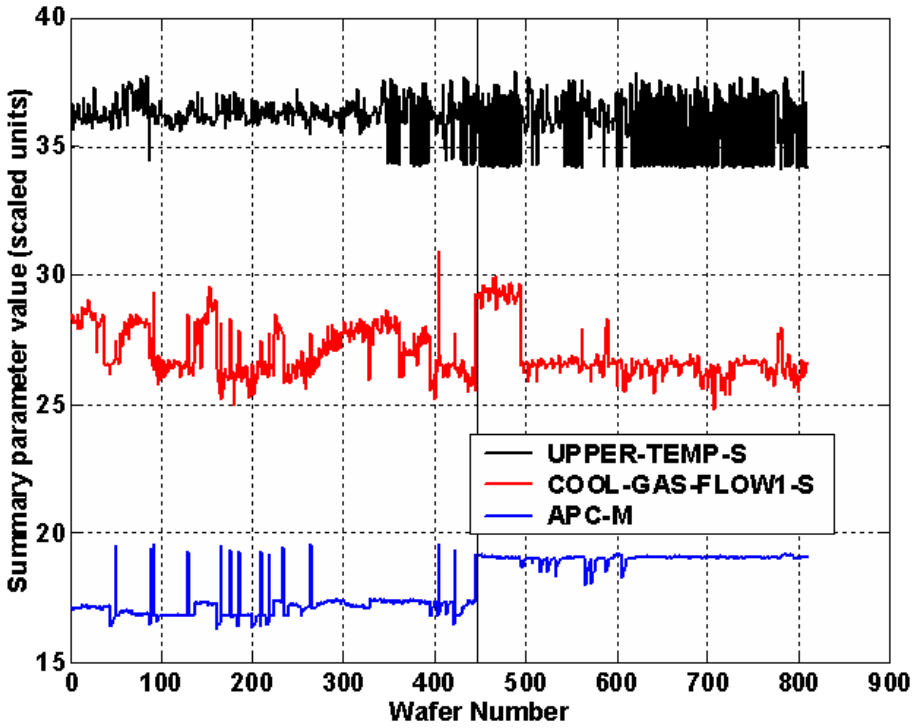


Figure 5.11: Process variable charts for largest mean movement metric contributors at wafer 445, UPPER-TEMP-S, COOL-GAS-FLOW1-S, and APC-M

used by Gallagher *et al.*[38], which utilizes continuous model rebuilding from recent windows of data, and the previously proposed method of simply recursively updating the means and standard deviations. In addition, they incorporated process knowledge by selecting variables typically without drifts and chose not to update these variables. They also looked at limiting updates when the T^2 value exceeds its current limits.

They concluded that all methods were able to identify the extreme step changes at the beginning of a wet clean cycle. All methods also identified RF voltage instabilities during one of the clean cycles. During an excursion where the temperature clearly drifts, the fully adapting model tracks with the drift and does not detect the fault. In their partially updating model (where temperature is considered a known non-drifting variable and is hence not updated), the drift can be clearly identified, as expected. In the model with a moving window model update and in the model with an update that uses all the data since the beginning of the experiment, a similar performance is shown as the fully updating model. In all the adapting cases drifts are difficult to identify.

Spitzlsperger *et al.*[100] also looked at a number of values for the tuning parameters λ and n used in the recursive updates for the mean and standard deviation, respectively. They found that for values of λ between 0.5 and 0.999 and values of n between 200 and 800, the Hotelling's T^2 value for both full updating and partially updating models did not change significantly. They concluded that the exact values of the tuning parameters are not critical to

the performance of the T^2 metric.

Spitzlsperger *et al.* concluded that all the adaptive methods successfully keep the T^2 statistic inside its fixed control limit, which is a major improvement over a static model. As a side effect of removing known uncritical process drifts, these adaptive methods also remove slowly rising excursions. They also concluded that adding computational effort to perform remodeling of the correlation structure did not improve fault identification performance. For the best monitoring results, they suggest the incorporation of engineering knowledge to allow some process variables to adapt while others remain fixed. Overall they found the method of recursively updating the means and centers very useful and practical.

5.1.4 Dynamic process monitoring summary

In this section it was shown that a static PCA model can be inadequate for monitoring a etch process chamber for a long period and cannot be effectively applied from one chamber to other chamber without updates. The adaptive centering and scaling of a PCA model did allow monitoring of a drifting process without rebuilding the model, and was able to utilize the original model limits. Even with the adaptation scheme, contribution plots of the squared prediction error, Q, were able to discriminate the root cause variables of the local deviations instead of global changes. In addition, the contribution plots of the proposed movement metrics discriminated variables that have sharp changes during a period of successive excursions or between

any two points that were monitored.

The work of Spitzlsperger *et al.* also validated the utility of this adaptation scheme for monitoring drifting processes without performing more complicated and computationally expensive model updates. Improvements could be made to incorporate process knowledge to selectively determine which variables are allowed to adapt. In any case, the burden of monitoring many univariate control charts and continuously updating acceptable limits are removed, and multivariate analysis techniques can be applied to monitor dynamic semiconductor process tools.

5.2 KNN algorithm extended to trace data

In this section two methods for applying the KNN method (described in Chapter 3) to tool trace data are proposed. In Section 2.4 algorithms for batch process monitoring are described. A three-dimensional array ($I \times J \times K$) is analyzed for model building, where I is the number of batches, J is the number of process variables, and K is the number of time-based samples made of each variable during a batch. The KNN methods will be applied to these same data arrays. The nonlinear properties of the underlying KNN methods will create a novel multiway method. In addition, the second KNN method will take advantage of the non-parametric aspect of the method to dynamically build a model for each time sample by using only reference data with similar step times.

5.2.1 Unfolding

One method to implement the KNN algorithm for trace data is to simply unfold the data in the time dimension for each batch data set, similar to multiway PCA as discussed in Section 2.4.1. This transformation takes the value of each process variable at each time as an entry in the sample vector. If a batch has 20 process variables being monitored for each of 30 time samples (for example a 1 minute process sampled every two seconds), this would result in a sample vector for the batch of 600 variables. For the KNN analysis method, a history of these sample vectors would be collected to create the reference data set.

For an overall performance metric, C is calculated as in Equation 3.4 for a multivariate problem. In order to obtain individual process variable, individual time sample, and step level contributions, block contributions are used as described in Section 3.1.6. One level in the block contributions is dedicated to individual process variables with all times of that variable sharing a block. Another level in the block contributions is used for each time sample, with all process variables sampled at that time sharing a block. A third level corresponds to each process step within the batch (if there is more than one) with a block for all the time samples within that step with the complete set of process variables. A final level is dedicated to all the univariate (process variable and time combinations) contributions with each element in the sample vector having its own contribution. Other custom levels can then be added if there are process variable groupings that of interest to a user. For each batch

sample, the overall metric is calculated and the analysis is repeated for each block within each level of the contributions. The results are a useful set of contributions to diagnose the root cause of an identified excursion.

One major drawback of this unfolding technique is that each batch must have an identical number of samples representing the exact same periods in time for each batch. In reality, data collection systems do not necessarily provide data at exact periodic rates, and fab processes do not necessarily always have the exact same duration on every run. In order to accommodate these realistic constraints, system data must be preprocessed into a fixed number of time samples that represent a similar time (or event) within the process step. The simplest method is to simply pick a fixed number of time periods that correspond to a fixed set of step times. Then for each time period interpolate the values from the process data at these times. Other more sophisticated methods for particular processes have been devised that attempt to look at the signals and align them such as dynamic time warping[55], but these were not the focus of this research.

The other drawback of this technique is its computational complexity. This method requires dynamically generating the relationship between points in a very large dimensional space. This technique must maintain enough reference data to characterize a region of space at every evaluation. As the number of dimensions of the space increase (both due to the number of process variables and the number of time samples per batch), this adds to the number of samples that are required. As the number of samples and the number of

dimensions grow, the computation time grows quickly.

5.2.2 Trace sample analysis

An alternative approach to the unfolding method takes advantage of the fact the KNN method does not have to build a fixed model ahead of time. The trace sample analysis technique assumes that reference data around the same time period within the batch (or process step) should be characteristic of a single sample within a new batch. To analyze a single sample trace sample within a batch, the algorithm simply queries all the reference data traces with step times within $\pm m$ seconds from the sample trace. The results of the query become the reference data for the individual sample trace, and the KNN algorithm is executed resulting in a performance value C for the single trace and process variable contributions for the single trace. This process is repeated for each trace sample within the batch to be analyzed. To obtain the overall batch performance metric, a weighted average of the time-based C metric is calculated as

$$C_{batch} = \log_{10}(\frac{1}{n}\sum_{i=1}^{n} 10^{C_i}), \tag{5.5}$$

where n is the number of traces within the batch. Similarly, the individual process variable contributions are also calculated as the weighted average of the time-based contributions. This exponentially weighted average was chosen so that faults which are extreme, but only occur during a small fraction of the process time, still result in a larger than average batch metric. Additionally, they still have the property that values between zero and one are normal

values and values of greater than one are excursions. Values of C above four are considered to be extreme faults.

This technique also adds a tuning parameter m, which is the number of seconds to scan within the reference data to find reference trace samples. The parameter m adds flexibility to use the KNN algorithm to monitor traces from a process that have variability in the timing of a particular feature. For instance in monitoring a plasma etch tool, the etch process may clear the main etch layer at varying times (based on a slowly drifting etch rate). This would cause the endpoint signals for monitoring emissions to have a transient earlier or later within a step. Since the KNN algorithm can model multiple populations within the reference data, if the m time window is wide enough to cover this variation in the time of a transient, these types of processes can be monitored without having unacceptably large windows around the transient period.

In order to illustrate the usefulness of m, a simulation of three variables was performed for a number of batches. To generate data with random variations in sample rate and time-based dynamics, a time counter and three process variables were generated as

$$t_{i+1} = t_i + 1/3 + 0.01\omega_i \tag{5.6}$$

$$x_{i+1} = \begin{cases} 5 + 0.5\omega_i, & i < 6 \\ x_i + ((i-6)/2)^2 + 0.25\omega_i, & 6 \leq i < 17 \\ 81 + 0.5\omega_i & i \geq 17 \end{cases} \tag{5.7}$$

$$y_{i+1} = y_i + 0.1x_i + 0.5\omega_i \tag{5.8}$$

$$z_{i+1} = z_i + x_i - y_i + 0.5\omega_i \tag{5.9}$$

where the initial vector of $[t_0, x_0, y_0, z_0] = [0, 5, 0, 0]$ and ω is random sequence of normally distributed values with mean zero and unity standard deviation. This system was simulated 35 times and the example traces are shown in Figure 5.12. From these 35 batches, many static KNN trace models were built on x, y, and z using various values of m. An additional 35 test batches were created by simply adding time delays from -0.8 seconds to +0.8 seconds to the existing simulated batches and are shown in Figure 5.13. Each of the models with various values of m were then applied to the test batches, and the KNN trace performance metric versus time delay for the various models is shown in Figure 5.14. For models with a small value of m, even when there is a small change in the time delay, there is a large increase in the performance metric. As the m value increases, the sensitivity of the performance metric to the change in time delay decreases. For processes that are known to shift in time, the m value can be tuned to allow this normal variation, but still will provide an alarm as the shift becomes too great. MPCA and KNN models using unfolding were also built with the simulated batches. Similarly these models were then applied to the same 35 test batches with various time delays. The resulting overall performance metric for each method with respect to the time delay is shown in Figure 5.15. The methods using unfolding have no inherent tuning capability to allow for varying time delay. These models could be rebuilt with data reflecting various time delays, but reference data with future time delays may not be available when the model is built.

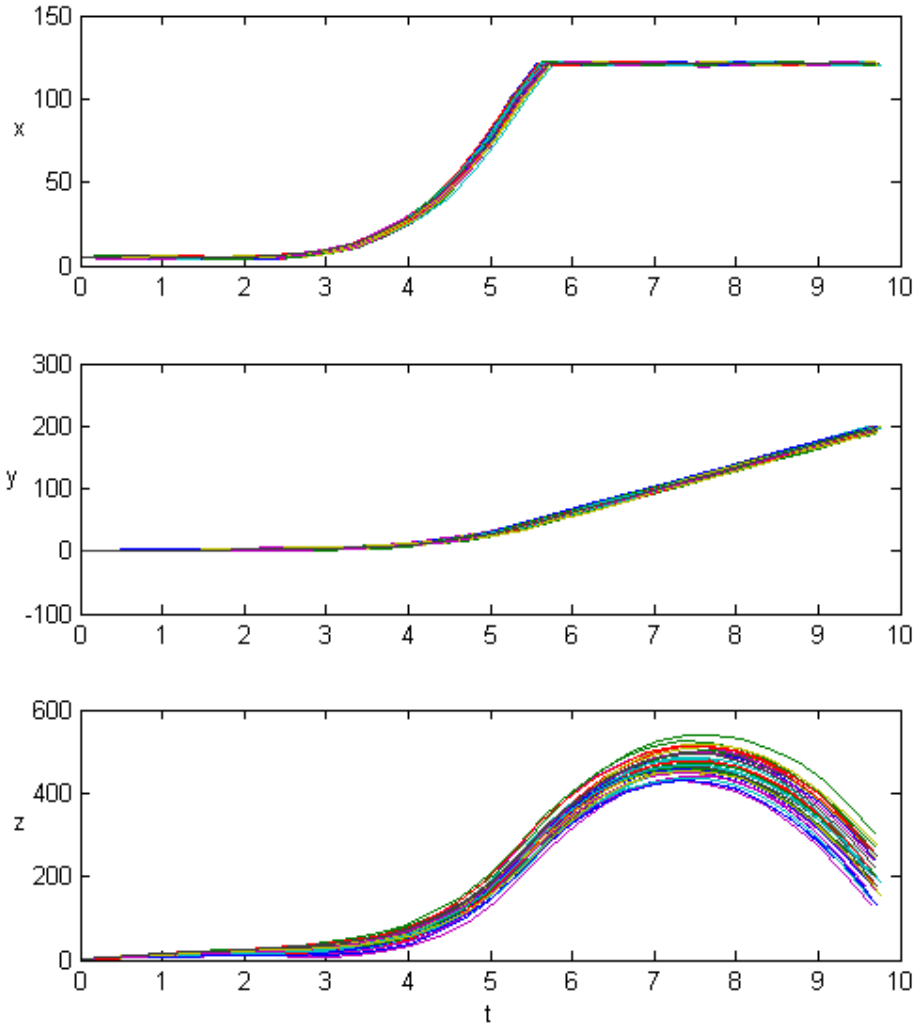


Figure 5.12: Simulated data for model building

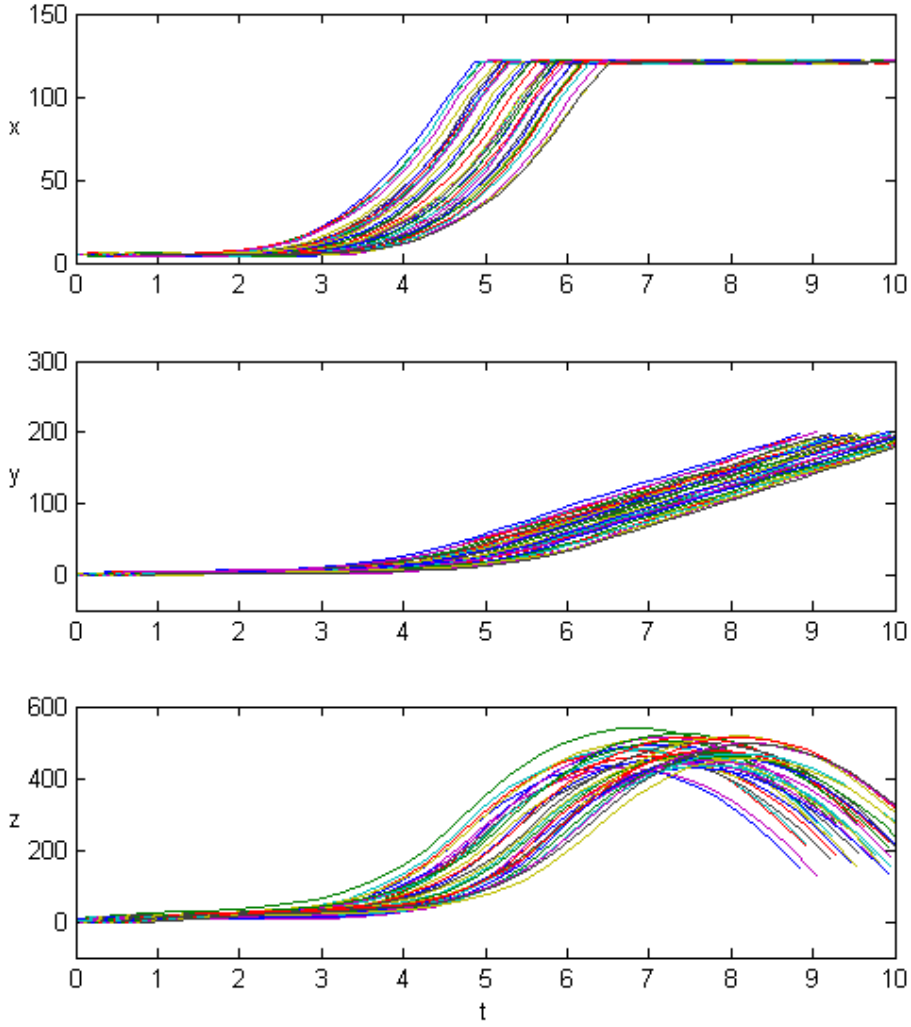


Figure 5.13: Simulated data with time delay of ± 0.8s for testing

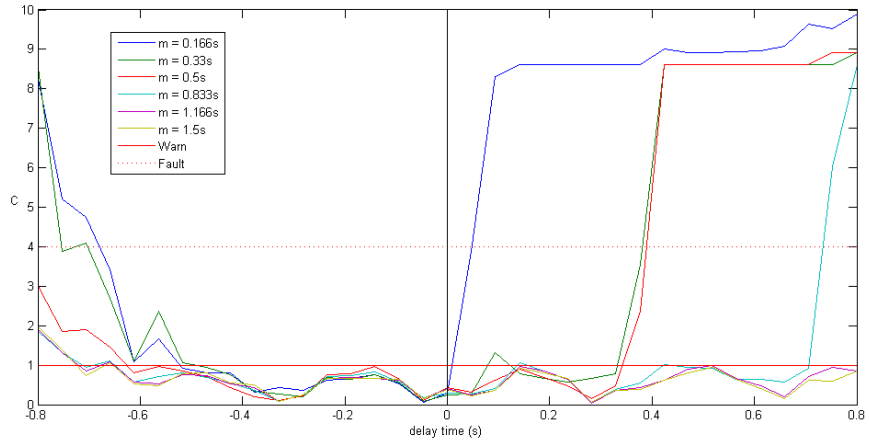


Figure 5.14: KNN trace performance metric C versus delay time for models built with varying values of m

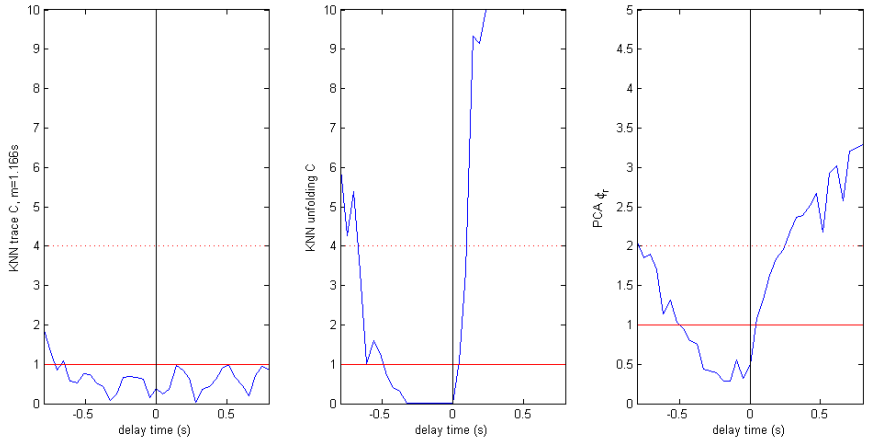


Figure 5.15: Performance metrics versus time delay for KNN methods and PCA

One disadvantage of this technique is that it no longer models the autocorrelation within the process step. In this technique reference data is used from a time period around each sample and only the correlation between variables during the time range is modeled. In the unfolding techniques, the correlation of the same variable at all different time periods is explicitly modeled. One major advantage this technique has over unfolding is that there is no longer a restriction that each batch must have the same number of time samples or even have the same sampling rate. In practice, tool sampling rates can vary and process steps terminated with endpoint systems will lead to very different batch durations. This advantage combined with the ability to tune the algorithm for accommodating time delay makes this a very useful algorithm for monitoring batch processes.

5.2.3 Algorithm comparison on plasma etch with induced faults

In order to compare the proposed KNN algorithms applied on trace data with existing algorithms, the machine data from a metal etch process with induced faults was used from the investigation of Wise *et al.*[112]. This data set is publicly available[1] in a MATLAB® format.

This investigation attempts to monitor an aluminum stack etch process performed on a Lam® 9600 plasma etch tool. The detailed process information and sensor information can be found in [112]. Each batch consists of 19 tool sensor values and has nominally 70 time samples. Three independent

[1] software.eigenvector.com/Data/Etch/

experiments were performed (29, 31, and 33) where faults were intentionally induced by changing the TCP power, RF power, pressure, Cl_2 or BCl_3 flow rate, and He chuck pressure. These experiments consisted of a total of 129 wafers with 21 faults. To make detection of these faults a non-trivial task, the sensor values for the induced faults were reset back to their original values in the data sets. This induced method simulates a sensor fault where the onboard controllers believe there is a change in a variable, when in reality the sensor reading is simply biased.

The three experiments were run at widely spaced intervals, each a month apart. Within each of the three experiments, trace data were collected for a number of normal runs, then the selected induced fault runs, and finally more normal runs. Between the experiments extreme tool drifts are apparent, and the univariate means of the three periods are very different. The data can be easily separated into three groups, one for each experiment. Because of this separation, a local model can be built for each of the three periods and a global model can be built from the entire data set.

Wise *et al.*[112] compared a number of methods in their investigation, which are discussed in Section 2.4. They looked at PCA models built (1) on the step means (with both global and local models), (2) on each raw data point (both global and local models), and (3) on mean-centered data (global model only). They created local and global multiway PCA models with unfolding along with a global model with mean-centering. They also created trilinear decomposition and PARAFAC models for each local region and the global

model. In Wise *et al.*[113] the authors also analyzed the same data set with the PARAFAC2 method, although they did not provide results of each individual fault. A summary of the previous results from these two investigations for each fault for the PCA, MPCA, TLD, PARAFAC, and PARAFAC2 models are shown in Table 5.2.

In order to compare the KNN methods, the raw data from the etch process was preprocessed. Similarly to [112], all the trace samples were ignored except for the last 25 samples from step 4 and the first 45 samples of step 5. For the KNN method with unfolding, the variables were then autoscaled. A KNN unfolding and a KNN trace model were built for each of the three experiments based on the normal data within the experiment and for a global model based on the normal data for all the experiments. For the KNN model with unfolding, the reference data for the local model consisted of 34 to 37 batches with 1330 unfolded variables (19 process variables by 70 samples per batch). With so few samples compared to variables, relatively high values of k and n of 20 and 30 were used. For the unfolding KNN global model, every other reference data point was used instead of all the reference data to speed up computations. For the KNN trace model, a value of m of 3.5 seconds (for the 1 sample per second sampling rate) was used. In addition, a reference data size is used which is long enough for each process step to cover all the reference data used in each experiment (and for all the experiments for the global model). The k and n values used in this model were configured to vary with each trace sample computation, and were computed as $k = \sqrt{s}$ and $n = s/3$, where s is the

		Straight PCA**					MPCA**			TLD**		PARA FAC**		PARA FAC2+		KNN (Unfold)		KNN Trace	
Exp	Induced Fault	Global on Means	Local on Means	Global on Raw Data	Local on Raw Data	Global on MC Data	Global	Local	Global on MC	Global	Local	Global	Local	Global	Local	Global	Local	Global	Local
29	TCP +50	o	*	o	o			o		x	x	x	x		o	o	*	*	*
29	RF +10		o		o						x		x		o		*		
29	Pr +3	*	*	*	*	o	o	*	o	x	x	x	x		*	*	*	*	*
29	TCP +10		o								x		x				*		o
29	BCl3 +5		o										x				o		*
29	Pr -2	*	*	*	*		o	*	o	x	x	x	x		*	*		*	*
29	Cl2 -5		*							x	x	x	x			o	*	o	*
29	He Chuck Pr																		
31	TCP +30	o	*	o	o		o	o		x	x	x	x			*	*	o	*
31	Cl2 +5									x		x	x				*		
31	BCl3 -5			*	*	*	*	*	*							*	*	*	*
31	Pr +2	*	*	*	*		o	o	o	x	x	x	x			*	*	*	*
31	TCP -20	o	*		o					x	x	x	x			*	*	o	*
33	TCP -15		o								x		x			o	*	o	*
33	Cl2 -10	o	o							x	x	x	x			o	*	o	o
33	RF -12	o	o										x			o	*		o
33	BCl3 +10	o	o	o	o	o	o	o		x	x	x	x			*	*	*	*
33	Pr +1	o	*	o	o			o		x	x	x	x			o	*	o	*
33	TCP +20		o		o						x	x	x			*	*	o	*
	Total	10	16	8	11	3	6	8	4	11	14	12	17	7	13	14	17	13	16

o identified in 99% limit or warning limit
* identified 5x over 99% limit or exceeded fault limit
x identified (reference did not distinguish 5x or not)
not included in reference

Table 5.2: Simulation results for various algorithms on fault runs, **[112], +[113]

number of reference data points selected in the particular trace calculation.

For each local model, the overall C metric was calculated for each induced fault run that occurred during that experiment. The global model was then applied to all the induced fault runs. For both KNN methods, C values greater than one are considered warnings and values greater than four are considered extreme faults. The results for the KNN with unfolding and KNN trace are shown in Table 5.2 along with the results of the other methods.

Overall, the KNN with unfolding method successfully identified many of the faults using local models. It performs similar to PARAFAC for detecting the most (17) of the 19 faults. The global model outperforms all other global methods. Unfortunately, the KNN model with unfolding also identifies some of the training data as excursions. Six of the 107 normal runs were identified as warnings by the local models and the global model identified seven reference data as extreme faults. So although this method was very sensitive to the induced faults, it did not validate satisfactorily with normal data. With only 34-37 batches used as reference data within the local KNN models, the KNN method does not have enough data to adequately characterize the distributions within the reference variables. In Section 3.2.2, simulations were performed on a number of different distributions with varying reference data sizes. For the normal distribution simulations with 33 reference data points as shown in Figure 3.10.a, the performance of the KNN method with this few reference data is poor even for a single reference variable. Figure 3.18.a shows that an increase in the number of reference data significantly improves the performance

of the method when the number of reference data is this low.

The KNN trace method also identified the induced faults well. The local models for this method captured 16 of the 19 faults, performed the same as PCA on local means, and identified only one fault less than PARAFAC. The global model for KNN trace was able to identify 13 of the faults, also exceeding any other global model from an existing method. The KNN trace models also validated much better than the KNN unfolding method. In each of the four KNN trace models, at most one normal run was identified as an extreme fault, and in the global model only two normal runs exceeded the warning limit.

The KNN methods both exceeded all methods in identifying the induced faults with a global model covering three distinct populations. This result helps to verify the ability of the KNN method to identify excursions even when the reference data is clustered into different populations. All the other methods are based on linear assumptions that only allow one continuous region of "normal" operating space. The KNN methods allow for separate populations and have the ability to identify faults between the data populations, compared to the linear methods that just expand the "normal" operating space to include all the populations within the modeled data.

5.3 Conclusions

In this chapter two new methods for monitoring semiconductor batch processes were presented. The first half of the chapter was dedicated to mon-

itoring drifting processes with adapting models. A method for updating the means and standard deviations used to center and scale the process variables within a PCA model after each run was presented. This method was shown to bring the squared prediction error (Q) calculated by the model back under the limit calculated for the model and maintain it there even as the process drifts. In addition, when the model is applied to a second process chamber, the Q value returned to below the model confidence limit in only 25 wafers. Spitzlsperger *et al.*[100] also compared this method to others and found that for monitoring a drifting system it performed comparably to other methods, including complete model rebuilds at every run.

The introduction of the mean and range movement metrics is another improvement to assist monitoring systems with drifting processes. These metrics allow the process variables contributing to process changes to be identified between two points in time when the model has adapted to them. This information is important to engineers monitoring the process because it is not available from normal fault contributions when models are configured to automatically adapt to changing conditions. This method also allows engineers to implement time-saving automatically adapting methods without concerns about missing slow adaptations.

The second half of this chapter discusses two techniques for applying the KNN algorithm developed in Chapter 3 to tool trace data. The KNN algorithm can be applied using unfolding similar to MPCA or it can be applied by searching ranges of trace data to create separate time-based reference data

for each time sample. The detailed implementations of these methods were discussed. The new tuning parameter in the time-based algorithm was shown to allow monitoring of batches with variable time delay without modeling reference data with all the possible time delays.

These two methods were compared to a large number of existing batch methods using the induced fault etch experiments provided by Wise *et al.*[112]. The KNN methods both performed quite well and ranked among the best of the methods for identifying the induced faults. Unfortunately, the KNN method with unfolding failed to validate normal runs as normal. Not much can be concluded about this approach since the reference data available in this experiment is smaller than what is needed for the KNN method. The KNN trace method performed very well and was able to validate normal runs. It showed that a KNN global model could be used to model multiple populations with only a slight loss of sensitivity compared to local models, and a better sensitivity than any other global model type. In addition, the KNN trace method includes flexibility for batches with varying sample rates, varying step durations, and a tuning parameter for treating variable time delays within the batch.

Chapter 6

Conclusions and Recommendations

6.1 Summary of work

In this work a number of improvements to multivariate fault detection are introduced and applied to semiconductor monitoring applications. The background for many existing statistically-based multivariate fault detection algorithms is presented. These mature algorithms have all the attributes required for robust fault monitoring. They include performance metrics with statistically calculated limits, include variable and block contributions, they handle missing data, and they provide options for automatic model updating. Existing methods for visualizing the results from these algorithms are also discussed. In addition, existing methods for analyzing batch processes using statistical multivariate algorithms are presented in detail.

Outside from the parametric statistical methods, non-parametric modeling methods for fault detection are also presented that come from the field of outlier and anomaly detection within multivariate data sets. These methods have identical goals as the parametric statistical fault detection methods. They have the advantage of simply storing reference data and make fewer assumptions about the structure of the data. These non-parametric methods

even function well when data is nonlinearly distributed and can have outlier metrics with local and global properties. Unfortunately, when all the existing non-parametric metrics are applied to data sets with populations of varying densities, the metrics take on different values near the different populations. These outlier metrics do not have uniform outlier limits that parallel the statistically generated confidence limits in all regions of the reference data.

Based on these non-parametric concepts, a k nearest neighbor algorithm for fault detection is proposed and evaluated in Chapter 3. This algorithm is based on comparing the sample point distribution to the characteristic distribution for a new sample point. From the overlap of these distributions, a fault detection metric C was created, with the tuning parameters of k, n, and the confidence limit of the model. The C metric scales such that all problem sizes yield a similar response of the metric and have the same confidence limit of unity. In addition, this univariate metric is applied to multiple dimensions to generate an overall performance metric and block contributions. This technique provides a non-parametric method with all the attributes required for robust fault monitoring and allows for clustered populations within the reference data.

In Chapter 4 a novel infrastructure is introduced to monitor any multivariate analysis method. To expand on the existing overall performance metric and contribution charts, periodic and sample reports are added. Periodic reports such as a daily report summarizing the weeks worth of model results provide significant information to monitoring engineers. Based on the variable

hierarchies created by the engineer, these reports have the ability to highlight the most extreme and most frequent excursions in both the multivariate analysis technique and the univariate factory limits, to identify the contributing variables or blocks of variables, and to put the supporting historical data right at an engineer's fingertips. They utilize two levels of alarms for both fixed and dynamic monitoring and require very limited maintenance by the monitoring engineer. This system was applied to monitor final wafer electrical test data for all products within Spansion's Fab25 and results are presented.

Methods to help monitor drifting processes with multivariate analysis techniques are discussed in Chapter 5 with applications to monitor semiconductor process tools. A method to recursively calculate the centering and scaling coefficients used in a PCA model is proposed. This method allows the model to incorporate the slow moving drifts and provide meaningful contributions relative to the current state of the process without the need to recalculate or recursively update the PCA model. It also allows a model built for one process chamber to be applied to another process chamber. Along with the method for adaptation, movement metrics are proposed that allow monitoring of the centering and scaling coefficients in adaptive models. When adaptive models are used, slow drifts in the process and model limits may go unnoticed. The movement metrics yield the relative process variable contributions to changes in the means and standard deviations between the two reference times. These can be used in periodic monitoring reports to highlight variables contributing to drifts over the period or can be used to highlight

the variables that contribute to the difference between two different chamber models. These methods are illustrated on an oxide etch process with variable drifts.

The final contribution of this work is two different approaches of applying the k nearest neighbor algorithm to batch systems. The first method applying the KNN algorithm is identical to the unfolding algorithm used in MPCA. The second method analyzes traces separately based only on reference traces within a window of time within a process step, essentially creating a separate model for each time period within the reference data set. This second method introduces a tuning parameter that adds flexibility for monitoring processes that have time delays within a batch. This method also removes the restriction that each batch must have the same sample rate and the same number of samples within each batch. Both the unfolding method and the trace sample method are applied to a metal etch process with induced faults[112]. The fault detection results are compared to a number of other batch techniques that have been used to analyze this same data set.

6.2 Summary of results

The development of the k nearest neighbor algorithm adds a new option for multivariate fault detection. In Chapter 3 this method was shown to have a similar response on various one-dimensional sample problems with different reference data population sizes, means, and standard deviations and when different values of k and n tuning parameters are applied. On a simulated

five variable problem with induced faults, the algorithm was able to identify faults, the blocks that contributed to the faults, and the individual variables that contributed to the faults. In both these sets of simulations, when the performance metric was applied in a different number of dimensions, the use of a common confidence limit of unity was shown to be appropriate for all cases. It was also shown that the n tuning parameter could be used to tune the global or local response of the algorithm in two-dimensional simulations with multiple populations of varying densities. These simulations also verify that the algorithm is tolerant to outliers within the reference data set, and is capable of identifying reference data as an outlier. This fault detection method has been shown to have all the attributes of a robust fault detection algorithm and provide sensitivity to multiple populations within the reference data set.

This algorithm was also extended to analyze batch process tool data in Chapter 5. The unfolding method and the trace sample method are each evaluated and the trace sample method is shown to be a viable candidate for monitoring batch processes. When the KNN algorithm with unfolding is applied to the metal etch process to identify induced faults, the method is able to identify the most faults, but it did not always validate known normal data. This method struggles from the small reference data size available in the experiment. When the KNN algorithm with trace sample analysis is used to identify the induced faults, it performs very well and (unlike the unfolding method) it is able to validate the known normal data. This algorithm ranks among the top methods using local models and has the highest detection rate

of any of the global models for this induced fault experiment. This result helps to show the usefulness of non-parametric algorithms to model processes with multiple data populations. It was also shown that this method's configuration parameter allows it to be robust when used to monitor processes with variable time delay within batches.

In addition to the creation of a new fault detection method, improvements for visualization of any multivariate method and additional methods for monitoring adapting methods are introduced in this work. A software infrastructure is proposed in Chapter 4 for monitoring an existing system with the addition of a multivariate method. The proposed reports with all their integrated charts and limits have proven to be very useful and informative in industrial practice. The periodic reports are used daily for multiple final wafer test steps by a number of engineers. They provide instant access to an overview of the number of excursions, a sorted list of the most important variables and variables blocks, and links directly to the historical data for the period. This application was able to expand the number of test parameters that are monitored by engineers because of the low maintenance that is required to use such an automated system.

In order to improve upon monitoring systems with slow drifts, a method to update the centering and scaling coefficients of a PCA model is proposed in Chapter 5 without requiring the model to be periodically rebuilt or recursively updated. When a static PCA model is applied to a drifting process, the squared prediction error quickly exceeds the limit and never returns to below

the limit. When the recursive method was applied, the squared prediction error for the model was kept generally below the confidence limit associate with the model. This adaptation method was also applied successfully to use a model from one process chamber to monitor the performance of process data from another chamber. In both cases the contributions of excursions were shown to better represent the changes in the sample from recent runs instead of the differences of the sample to the runs used to build the model.

Another improvement to monitoring systems with adapting models was the introduction of relative movement metrics. When adapting models are used, information is lost concerning the dynamics to which the model is adapting. The movement metrics are shown to identify the variables that contribute to drifts and changes in variance of the system. They can also be used effectively to identify the differences between two models applied to two different process chambers. Overall, these metrics identify to engineers the signals that are causing the models to adapt. This information allows engineers to confidently use adaptive models without worrying that the models will adapt to unknown disturbances and never generate alarms.

This work has made a number of improvements to existing fault detection methods. The k nearest neighbor algorithm has the advantage of the flexibility of simply retaining reference data without pre-computing a model, and it has unique properties for detecting local outliers in reference data with multiple populations. This algorithm applied to trace data extends the existing batch process monitoring techniques. The method of trace sample analy-

sis provides improved performance of global models and provides additional flexibility of time delays and variable batch lengths. The visualization infrastructure adds to the existing simple charting methods that currently exist and provides a summary over a specified duration to identify both extreme and frequent excursions. The method of adaptive means and centering coefficients and the movement metric calculations both assist in the monitoring of a drifting process by accommodating change without losing the information contributing to the change. All of these improvements help in performing multivariate fault detection, especially in the semiconductor manufacturing industry.

6.3 Recommendations for future work

Based on this work, many areas of improvement could be investigated in the future. In the KNN algorithm, better rules could be developed for selecting values of k and n to use. The square root rule for k and the one third rule for n work well for large reference data sizes, but for small reference data sizes, k and n should most likely have minimum thresholds. In addition, the value of n is tuned so that the algorithm can be sensitive to multiple populations. For robustness, this feature should not be used until the reference data size reaches a certain minimum size.

Also, since the KNN algorithm seeks to create continuous distributions, when discrete or rounded process data are included that only take on a few individual values, the selection of nearest neighbors and the estimation of

distributions are difficult. Discrete exception handling could be investigated to replace the continuous distributions when there are many repeated values within the reference data.

In order to make the KNN algorithm adaptive in this work, a simple moving window of reference data is used. In this case, only the model length is available to tune the dynamic memory of the model, the robustness of the non-parametric estimates, and the computation time of the method. An improvement could be made to enhance the moving window and attempt to keep memory by selecting which sample of the old reference data to disregard when a new sample is added. Similar to Wold's [116] method, key reference data could be maintained while other reference data could be removed. Alternatively, quick calculations such as the Zeta metric[89] could be performed on the reference data to rank the normality of each sample. Then traces could be selected to be removed based on probability, in order to balance keeping the reference data on the periphery but also not allowing the acceptable region defined by the reference data to only grow over time.

Finding simpler calculations that have similar properties could be investigated. This task could be performed along with simplifying the models used in Equations 3.3 and 3.6 to unify the metric for all values of k, n, s, and L. In addition, more theoretical approaches could be used to fitting these models to improve upon the empirical approach used in this work.

For the KNN algorithm applied to batch data, two areas for improvement could be investigated for the trace sample analysis method. First, the

time variable used to search for trace samples within the reference data could be replaced with a monotonically changing "batch maturity metric", if one exists. Similar to the PLS method used in Wold *et al.*[119], this change would allow the current trace to be compared with traces at a similar maturity instead of necessarily in the same time period. Another current problem with using the process recipe step time is that there is no support for variable length batches when the sample batch time exceeds the longest batch time in the reference data set plus m. Samples after this time period have no reference data and simply generate alarms. Providing some performance index calculation for samples beyond this time would be a valuable improvement to the algorithm.

Appendices

Appendix A

Non-overlap threshold raw data sets

As discussed in section 3.1.3, in order to find a relationship between the non-overlap probability threshold value and a given confidence limit, simulations were performed for a series of different reference data sizes (s), values of k, and values of n all using different data generated from the same normal distribution. Then the values of the non-overlap probability were calculated at the sample points corresponding to the different confidence limits (L values of 0.999, 0.997, 0.975, 0.9, 0.8, and 0.85) calculated from the normal distribution. All combinations of reference data of size (s) of 40, 100, 300, 800, 1500 entries, n values of 10, 20, 35, 66, 150, 500, and s, and k values of 10, 20, 33, 40, 60, 150, and 250 were used where $k < n < s$. It is worth pointing out that simulations with low values of s, k, and n provided very noisy results (as would be expected with the statistics involved with small sample sizes). Since this area is not where the algorithm will be used in the future, these values were not given as much weight in the fitting analysis. The raw data from these experiments are in the following two tables.

Table A.1: Table of experimental results for confidence limit overlap probability threshold, part 1

Experiment Settings				Overlap Threshold Value for various L						Percent Absolute Prediction Error					
Exp.	s	k	n	0.999	0.997	0.975	0.9	0.8	0.85	0.999	0.997	0.975	0.9	0.8	0.85
1	40	10	10	0.222	0.371	0.659	0.791	0.815	0.871	82	41	2	13	18	8
2	40	10	20	0.119	0.238	0.483	0.713	0.827	0.828	66	8	19	18	13	9
3	40	20	20	0.154	0.255	0.511	0.768	0.899	0.868	74	15	13	9	4	4
4	40	10	35	0.078	0.181	0.409	0.674	0.835	0.831	49	21	38	19	10	5
5	40	20	35	0.115	0.206	0.449	0.733	0.896	0.872	65	6	25	10	2	0
6	40	33	35	0.185	0.275	0.518	0.783	0.927	0.89	78	21	9	3	1	2
7	40	10	40	0.121	0.239	0.485	0.728	0.888	0.865	67	9	16	10	3	0
8	40	20	40	0.155	0.257	0.513	0.783	0.929	0.907	74	15	9	2	2	4
9	40	33	40	0.222	0.317	0.567	0.826	0.949	0.923	82	31	1	3	4	6
10	100	10	10	0.168	0.375	0.865	0.916	0.82	0.969	71	23	9	3	20	0
11	100	10	20	0.086	0.236	0.678	0.831	0.89	0.884	53	5	0	9	9	7
12	100	20	20	0.133	0.265	0.694	0.847	0.94	0.933	70	15	2	7	3	2
13	100	10	35	0.038	0.152	0.552	0.811	0.925	0.894	7	44	10	7	3	3
14	100	20	35	0.077	0.181	0.568	0.836	0.952	0.934	48	20	6	4	0	1
15	100	33	35	0.12	0.222	0.595	0.872	0.975	0.956	66	2	2	1	2	4
16	100	10	66	0.015	0.099	0.467	0.817	0.925	0.891	173	120	21	0	0	1
17	100	20	66	0.044	0.128	0.489	0.841	0.951	0.929	9	70	16	3	2	5
18	100	33	66	0.079	0.166	0.521	0.878	0.971	0.95	49	31	9	7	4	7
19	100	40	66	0.097	0.186	0.537	0.892	0.976	0.961	59	17	6	8	5	8
20	100	60	66	0.161	0.253	0.58	0.902	0.984	0.973	75	14	2	9	6	9
21	100	10	100	0.024	0.12	0.5	0.825	0.927	0.89	71	81	12	3	1	3
22	100	20	100	0.056	0.148	0.518	0.851	0.949	0.927	28	47	8	6	4	6
23	100	33	100	0.093	0.187	0.549	0.89	0.967	0.948	57	17	2	10	6	8
24	100	40	100	0.114	0.208	0.565	0.905	0.972	0.959	65	5	1	12	6	10
25	100	60	100	0.182	0.278	0.61	0.926	0.981	0.974	78	22	8	14	7	11
26	300	10	10	0.112	0.203	0.904	0.826	0.946	0.92	113	180	1	19	5	8
27	300	10	20	0.054	0.194	0.887	0.814	0.926	0.924	51	94	5	18	7	6
28	300	20	20	0.149	0.313	0.945	0.879	0.961	0.975	45	20	11	10	3	0
29	300	10	35	0.021	0.116	0.825	0.814	0.93	0.927	110	129	7	15	5	4
30	300	20	35	0.086	0.216	0.873	0.871	0.968	0.974	49	23	12	8	1	1
31	300	33	35	0.146	0.282	0.877	0.913	0.979	0.986	70	6	13	3	0	2
32	300	10	66	0.005	0.05	0.655	0.81	0.918	0.942	740	340	2	11	5	0
33	300	20	66	0.037	0.122	0.689	0.876	0.95	0.975	10	81	4	3	2	3
34	300	33	66	0.077	0.175	0.695	0.913	0.965	0.988	48	27	4	1	0	5
35	300	40	66	0.093	0.192	0.695	0.928	0.97	0.991	57	15	4	3	0	5
36	300	60	66	0.127	0.228	0.695	0.947	0.977	0.995	68	3	4	5	1	5
37	300	10	150	0	0.009	0.525	0.74	0.916	0.914	18998	2291	10	13	2	1
38	300	20	150	0.006	0.049	0.536	0.783	0.94	0.95	523	343	8	7	0	5
39	300	33	150	0.026	0.084	0.537	0.821	0.951	0.962	55	160	8	2	1	6
40	300	40	150	0.035	0.096	0.535	0.839	0.955	0.967	14	126	8	0	2	7
41	300	60	150	0.056	0.122	0.533	0.86	0.964	0.977	28	78	8	3	3	8
42	300	150	150	0.135	0.213	0.564	0.869	0.97	0.95	70	2	2	4	3	5
43	300	10	300	0	0.009	0.526	0.733	0.929	0.919	29180	2211	7	9	2	6
44	300	20	300	0.006	0.049	0.537	0.789	0.942	0.943	606	342	5	1	3	8
45	300	33	300	0.025	0.084	0.537	0.825	0.951	0.954	58	161	4	3	4	9
46	300	40	300	0.035	0.096	0.536	0.842	0.954	0.958	16	127	5	5	4	9
47	300	60	300	0.055	0.122	0.534	0.865	0.961	0.967	27	79	5	8	5	10
48	300	150	300	0.135	0.213	0.564	0.88	0.976	0.977	70	2	1	9	6	11
49	300	250	300	0.228	0.308	0.618	0.891	0.986	0.977	82	29	9	10	7	11
50	800	10	10	0.324	0.389	0.95	0.666	0.956	0.795	77	105	2	49	4	25
51	800	10	20	0.216	0.316	0.935	0.647	0.96	0.78	55	105	0	52	4	27
52	800	20	20	0.222	0.379	0.948	0.805	0.983	0.917	51	71	2	22	1	8
53	800	10	35	0.151	0.241	0.946	0.62	0.957	0.847	7	103	6	57	4	16
54	800	20	35	0.154	0.271	0.972	0.784	0.981	0.942	6	81	8	24	1	5
55	800	33	35	0.165	0.312	0.972	0.865	0.991	0.969	2	57	8	13	0	2
56	800	10	66	0.094	0.167	0.903	0.62	0.955	0.868	36	95	10	54	3	12
57	800	20	66	0.098	0.178	0.935	0.782	0.98	0.939	39	83	13	22	1	4
58	800	33	66	0.101	0.208	0.948	0.859	0.989	0.965	41	56	14	11	0	1
59	800	40	66	0.107	0.222	0.95	0.883	0.99	0.972	44	46	14	8	0	0
60	800	60	66	0.133	0.262	0.956	0.924	0.986	0.981	55	24	15	3	0	1

Table A.2: Table of experimental results for confidence limit overlap probability threshold, part 2

Experiment Settings				Overlap Threshold Value for various CL						Percent Absolute Prediction Error					
Exp.	s	k	n	0.999	0.997	0.975	0.9	0.8	0.85	0.999	0.997	0.975	0.9	0.8	0.85
61	800	10	150	0.044	0.1	0.794	0.601	0.956	0.829	9	125	13	52	2	15
62	800	20	150	0.051	0.104	0.827	0.758	0.98	0.893	21	117	17	20	1	6
63	800	33	150	0.053	0.119	0.833	0.823	0.981	0.922	23	90	17	11	1	3
64	800	40	150	0.055	0.129	0.833	0.844	0.981	0.927	27	75	17	8	1	2
65	800	60	150	0.068	0.158	0.833	0.878	0.975	0.937	41	43	17	4	0	1
66	800	150	150	0.151	0.262	0.807	0.927	0.987	0.948	73	14	15	2	2	0
67	800	10	500	0.005	0.035	0.591	0.505	0.935	0.789	692	518	4	63	1	13
68	800	20	500	0.013	0.038	0.601	0.653	0.967	0.825	198	469	5	26	4	8
69	800	33	500	0.015	0.04	0.588	0.711	0.971	0.849	174	449	3	16	4	5
70	800	40	500	0.015	0.042	0.584	0.729	0.968	0.855	164	416	3	13	4	4
71	800	60	500	0.018	0.054	0.578	0.76	0.959	0.864	125	303	2	8	3	3
72	800	150	500	0.048	0.108	0.553	0.788	0.961	0.876	17	102	3	4	3	1
73	800	250	500	0.081	0.149	0.547	0.803	0.963	0.885	50	46	4	2	3	0
74	800	10	800	0.004	0.034	0.638	0.563	0.952	0.846	832	545	12	42	4	3
75	800	20	800	0.014	0.04	0.644	0.699	0.963	0.842	190	451	13	14	5	3
76	800	33	800	0.016	0.044	0.63	0.749	0.966	0.868	151	390	11	7	5	0
77	800	40	800	0.017	0.049	0.626	0.764	0.967	0.876	136	341	10	4	6	1
78	800	60	800	0.021	0.065	0.619	0.793	0.968	0.889	89	233	9	1	6	2
79	800	150	800	0.06	0.128	0.593	0.83	0.966	0.905	33	70	5	4	5	4
80	800	250	800	0.097	0.172	0.586	0.845	0.969	0.916	59	26	4	6	6	5
81	1500	10	10	0.676	0.789	0.973	0.944	0.891	0.769	10	12	1	5	12	30
82	1500	10	20	0.66	0.754	0.953	0.95	0.866	0.756	16	4	1	4	15	32
83	1500	20	20	0.673	0.84	0.977	0.966	0.941	0.89	18	7	1	3	6	12
84	1500	10	35	0.513	0.589	0.949	0.95	0.879	0.801	30	13	1	4	13	24
85	1500	20	35	0.518	0.668	0.976	0.966	0.927	0.922	31	1	4	2	7	8
86	1500	33	35	0.523	0.686	0.98	0.983	0.954	0.96	31	3	4	0	4	3
87	1500	10	66	0.374	0.48	0.957	0.938	0.858	0.85	57	2	7	4	16	16
88	1500	20	66	0.374	0.543	0.966	0.967	0.893	0.946	57	10	8	1	11	4
89	1500	33	66	0.375	0.554	0.95	0.978	0.93	0.961	57	12	6	0	7	3
90	1500	40	66	0.375	0.556	0.949	0.981	0.933	0.967	57	12	6	1	6	2
91	1500	60	66	0.377	0.558	0.954	0.988	0.951	0.97	57	12	7	1	4	2
92	1500	10	150	0.233	0.296	0.857	0.942	0.873	0.817	79	2	8	1	13	19
93	1500	20	150	0.226	0.347	0.847	0.975	0.91	0.915	78	17	7	3	8	6
94	1500	33	150	0.223	0.357	0.833	0.985	0.946	0.933	78	19	5	4	4	4
95	1500	40	150	0.221	0.359	0.831	0.982	0.952	0.938	78	20	5	4	3	3
96	1500	60	150	0.22	0.36	0.83	0.981	0.961	0.944	78	20	5	3	2	3
97	1500	150	150	0.234	0.368	0.82	0.993	0.987	0.981	79	22	4	5	0	1
98	1500	10	500	0.108	0.1	0.657	0.871	0.875	0.806	63	117	7	0	9	14
99	1500	20	500	0.095	0.151	0.644	0.906	0.911	0.91	58	45	5	4	5	1
100	1500	33	500	0.089	0.165	0.628	0.922	0.945	0.93	55	32	3	6	1	1
101	1500	40	500	0.087	0.167	0.623	0.925	0.946	0.936	54	31	2	6	1	1
102	1500	60	500	0.084	0.168	0.612	0.932	0.951	0.943	52	30	0	7	0	2
103	1500	150	500	0.088	0.171	0.589	0.929	0.973	0.967	54	27	4	6	2	5
104	1500	250	500	0.105	0.189	0.589	0.931	0.981	0.967	62	15	4	7	3	5
105	1500	10	1500	0.083	0.062	0.6	0.806	0.874	0.819	52	249	6	1	4	6
106	1500	20	1500	0.068	0.106	0.578	0.843	0.923	0.909	41	105	3	5	1	5
107	1500	33	1500	0.062	0.121	0.558	0.86	0.919	0.917	36	80	1	7	1	5
108	1500	40	1500	0.06	0.123	0.552	0.865	0.923	0.917	33	77	2	8	1	5
109	1500	60	1500	0.057	0.124	0.539	0.875	0.931	0.919	30	76	4	9	2	6
110	1500	150	1500	0.06	0.126	0.513	0.889	0.95	0.928	33	73	9	10	4	7
111	1500	250	1500	0.072	0.141	0.511	0.893	0.958	0.936	44	55	10	11	5	7

Appendix B

Scaling the KNN metric for problem size

As discussed in section 3.1.4, in order to determine the ν as a function of the problem parameters simulations were used. Simulations were designed to cover the broad range of reference data length (s = 42, 98, 308, 812, 1512), a full range of characteristic depths ($n = s/7,\ 2s/7,\ ...,\ s$), various values of nearest neighbors ($k = 2s/14,\ 3s/14,\ ...\ ,\ n$) and four different confidence limit values (L = 0.999 0.9975 0.975 0.9). For each simulation, the value of ν was calculated such that the metric would have a value of 10 at a sample point that was $\sqrt{10}$ times the distance from the mean as the confidence limit location. This should provide a two point calibration of the overall performance index C fitting the value of 10 and the value of 1 (the confidence limit). The raw data from these experiments are in the following ten tables.

Table B.1: Table of experimental results for scaling the KNN metric by predicting ν, part 1

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
1	42	6	6	0.999	11.514	13.796	20
2	42	6	12	0.999	12.858	15.859	23
3	42	9	12	0.999	8.757	10.797	23
4	42	12	12	0.999	5.796	7.575	31
5	42	6	18	0.999	14.688	17.371	18
6	42	9	18	0.999	10.113	12.309	22
7	42	12	18	0.999	6.942	9.087	31
8	42	15	18	0.999	5.418	6.858	27
9	42	18	18	0.999	4.38	5.233	19
10	42	6	24	0.999	16.735	18.332	10
11	42	9	24	0.999	12.357	13.27	7
12	42	12	24	0.999	8.595	10.049	17
13	42	15	24	0.999	6.635	7.819	18
14	42	18	24	0.999	5.309	6.194	17
15	42	21	24	0.999	4.442	4.965	12
16	42	6	30	0.999	18.584	18.742	1
17	42	9	30	0.999	13.789	13.68	1
18	42	12	30	0.999	9.74	10.459	7
19	42	15	30	0.999	7.563	8.229	9
20	42	18	30	0.999	6.025	6.604	10
21	42	21	30	0.999	4.988	5.375	8
22	42	6	36	0.999	17.514	18.601	6
23	42	9	36	0.999	13.561	13.539	0
24	42	12	36	0.999	9.75	10.318	6
25	42	15	36	0.999	7.627	8.088	6
26	42	18	36	0.999	6.134	6.463	5
27	42	21	36	0.999	5.105	5.234	3
28	42	6	42	0.999	16.961	17.909	6
29	42	9	42	0.999	10.408	12.847	23
30	42	12	42	0.999	7.465	9.626	29
31	42	15	42	0.999	5.842	7.396	27
32	42	18	42	0.999	4.713	5.771	22
33	42	21	42	0.999	3.956	4.542	15
34	98	14	14	0.999	11.363	13.229	16
35	98	14	28	0.999	14.739	15.293	4
36	98	21	28	0.999	11.293	10.412	8
37	98	28	28	0.999	8.552	7.298	15
38	98	14	42	0.999	16.431	16.805	2
39	98	21	42	0.999	14.453	11.924	18
40	98	28	42	0.999	10.922	8.81	19
41	98	35	42	0.999	8.223	6.652	19
42	98	42	42	0.999	6.455	5.076	21
43	98	14	56	0.999	18.414	17.766	4
44	98	21	56	0.999	16.404	12.885	21
45	98	28	56	0.999	12.824	9.772	24
46	98	35	56	0.999	9.612	7.613	21
47	98	42	56	0.999	7.465	6.037	19
48	98	49	56	0.999	5.954	4.844	19
49	98	14	70	0.999	18.584	18.176	2
50	98	21	70	0.999	17.588	13.295	24
51	98	28	70	0.999	14.063	10.182	28
52	98	35	70	0.999	10.607	8.023	24
53	98	42	70	0.999	8.275	6.447	22
54	98	49	70	0.999	6.627	5.254	21
55	98	14	84	0.999	18.533	18.035	3
56	98	21	84	0.999	18.276	13.154	28
57	98	28	84	0.999	14.611	10.041	31
58	98	35	84	0.999	10.823	7.882	27
59	98	42	84	0.999	8.338	6.306	24
60	98	49	84	0.999	6.599	5.113	23
61	98	14	98	0.999	18.535	17.343	6
62	98	21	98	0.999	15.537	12.462	20
63	98	28	98	0.999	11.704	9.349	20

Table B.2: Table of experimental results for scaling the KNN metric by predicting ν, part 2

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
64	98	35	98	0.999	8.727	7.19	18
65	98	42	98	0.999	6.784	5.614	17
66	98	49	98	0.999	5.407	4.42	18
67	308	44	44	0.999	7.602	11.316	49
68	308	44	88	0.999	10.945	13.38	22
69	308	66	88	0.999	6.99	9.103	30
70	308	88	88	0.999	5.051	6.353	26
71	308	44	132	0.999	14.219	14.892	5
72	308	66	132	0.999	9.107	10.615	17
73	308	88	132	0.999	6.574	7.866	20
74	308	110	132	0.999	5.086	5.946	17
75	308	132	132	0.999	4.041	4.536	12
76	308	44	176	0.999	17.005	15.853	7
77	308	66	176	0.999	10.856	11.576	7
78	308	88	176	0.999	7.793	8.827	13
79	308	110	176	0.999	5.988	6.907	15
80	308	132	176	0.999	4.749	5.497	16
81	308	154	176	0.999	3.862	4.424	15
82	308	44	220	0.999	18.584	16.263	12
83	308	66	220	0.999	12.631	11.987	5
84	308	88	220	0.999	9.04	9.237	2
85	308	110	220	0.999	6.923	7.318	6
86	308	132	220	0.999	5.466	5.908	8
87	308	154	220	0.999	4.415	4.834	9
88	308	44	264	0.999	18.533	16.122	13
89	308	66	264	0.999	13.544	11.846	13
90	308	88	264	0.999	9.636	9.096	6
91	308	110	264	0.999	7.319	7.177	2
92	308	132	264	0.999	5.747	5.766	0
93	308	154	264	0.999	4.624	4.693	1
94	308	44	308	0.999	15.555	15.43	1
95	308	66	308	0.999	9.949	11.153	12
96	308	88	308	0.999	7.137	8.404	18
97	308	110	308	0.999	5.494	6.484	18
98	308	132	308	0.999	4.365	5.074	16
99	308	154	308	0.999	3.552	4.001	13
100	812	116	116	0.999	5.157	7.79	51
101	812	116	232	0.999	7.566	9.853	30
102	812	174	232	0.999	5.052	6.658	32
103	812	232	232	0.999	3.748	4.568	22
104	812	116	348	0.999	10.149	11.365	12
105	812	174	348	0.999	6.728	8.17	21
106	812	232	348	0.999	4.947	6.08	23
107	812	290	348	0.999	3.801	4.6	21
108	812	348	348	0.999	3.098	3.498	13
109	812	116	464	0.999	12.699	12.326	3
110	812	174	464	0.999	8.387	9.131	9
111	812	232	464	0.999	6.135	7.042	15
112	812	290	464	0.999	4.696	5.561	18
113	812	348	464	0.999	3.742	4.459	19
114	812	406	464	0.999	3.085	3.61	17
115	812	116	580	0.999	14.887	12.736	14
116	812	174	580	0.999	9.774	9.541	2
117	812	232	580	0.999	7.103	7.452	5
118	812	290	580	0.999	5.409	5.971	10
119	812	348	580	0.999	4.273	4.869	14
120	812	406	580	0.999	3.497	4.02	15
121	812	116	696	0.999	16.563	12.595	24
122	812	174	696	0.999	10.699	9.4	12
123	812	232	696	0.999	7.67	7.311	5
124	812	290	696	0.999	5.793	5.83	1
125	812	348	696	0.999	4.541	4.728	4
126	812	406	696	0.999	3.673	3.879	6

Table B.3: Table of experimental results for scaling the KNN metric by predicting ν, part 3

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
127	812	116	812	0.999	11.374	11.903	5
128	812	174	812	0.999	7.549	8.708	15
129	812	232	812	0.999	5.538	6.618	19
130	812	290	812	0.999	4.247	5.138	21
131	812	348	812	0.999	3.437	4.036	17
132	812	406	812	0.999	2.813	3.187	13
133	1512	216	216	0.999	3.95	4.528	15
134	1512	216	432	0.999	5.947	6.592	11
135	1512	324	432	0.999	3.874	4.351	12
136	1512	432	432	0.999	2.92	2.855	2
137	1512	216	648	0.999	8.179	8.104	1
138	1512	324	648	0.999	5.312	5.863	10
139	1512	432	648	0.999	3.878	4.367	13
140	1512	540	648	0.999	3.078	3.288	7
141	1512	648	648	0.999	2.489	2.472	1
142	1512	216	864	0.999	10.109	9.065	10
143	1512	324	864	0.999	6.533	6.824	4
144	1512	432	864	0.999	4.718	5.328	13
145	1512	540	864	0.999	3.665	4.249	16
146	1512	648	864	0.999	2.963	3.433	16
147	1512	756	864	0.999	2.475	2.796	13
148	1512	216	1080	0.999	12.078	9.475	22
149	1512	324	1080	0.999	7.783	7.234	7
150	1512	432	1080	0.999	5.601	5.738	2
151	1512	540	1080	0.999	4.259	4.659	9
152	1512	648	1080	0.999	3.422	3.843	12
153	1512	756	1080	0.999	2.856	3.206	12
154	1512	216	1296	0.999	13.457	9.334	31
155	1512	324	1296	0.999	8.578	7.093	17
156	1512	432	1296	0.999	6.093	5.597	8
157	1512	540	1296	0.999	4.57	4.518	1
158	1512	648	1296	0.999	3.632	3.702	2
159	1512	756	1296	0.999	2.982	3.065	3
160	1512	216	1512	0.999	9.097	8.642	5
161	1512	324	1512	0.999	5.906	6.401	8
162	1512	432	1512	0.999	4.287	4.905	14
163	1512	540	1512	0.999	3.363	3.826	14
164	1512	648	1512	0.999	2.759	3.01	9
165	1512	756	1512	0.999	2.306	2.372	3
166	42	6	6	0.9975	8.79	9.109	4
167	42	6	12	0.9975	10.51	11.172	6
168	42	9	12	0.9975	7.253	7.353	1
169	42	12	12	0.9975	5.194	4.92	5
170	42	6	18	0.9975	12.916	12.684	2
171	42	9	18	0.9975	9.188	8.866	4
172	42	12	18	0.9975	6.808	6.432	6
173	42	15	18	0.9975	4.946	4.746	4
174	42	18	18	0.9975	3.883	3.516	9
175	42	6	24	0.9975	15.084	13.645	10
176	42	9	24	0.9975	10.483	9.827	6
177	42	12	24	0.9975	7.625	7.393	3
178	42	15	24	0.9975	5.555	5.707	3
179	42	18	24	0.9975	4.372	4.477	2
180	42	21	24	0.9975	3.679	3.546	4
181	42	6	30	0.9975	16.198	14.055	13
182	42	9	30	0.9975	11.741	10.237	13
183	42	12	30	0.9975	8.733	7.803	11
184	42	15	30	0.9975	6.463	6.117	5
185	42	18	30	0.9975	5.119	4.887	5
186	42	21	30	0.9975	4.281	3.956	8
187	42	6	36	0.9975	16.278	13.914	15
188	42	9	36	0.9975	12.682	10.096	20
189	42	12	36	0.9975	9.108	7.662	16

Table B.4: Table of experimental results for scaling the KNN metric by predicting ν, part 4

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
190	42	15	36	0.9975	6.581	5.976	9
191	42	18	36	0.9975	5.138	4.746	8
192	42	21	36	0.9975	4.274	3.815	11
193	42	6	42	0.9975	13.368	13.222	1
194	42	9	42	0.9975	9.389	9.404	0
195	42	12	42	0.9975	6.983	6.97	0
196	42	15	42	0.9975	5.14	5.284	3
197	42	18	42	0.9975	4.02	4.054	1
198	42	21	42	0.9975	3.366	3.123	7
199	98	14	14	0.9975	7.608	8.712	15
200	98	14	28	0.9975	9.255	10.775	16
201	98	21	28	0.9975	6.414	7.085	10
202	98	28	28	0.9975	4.663	4.728	1
203	98	14	42	0.9975	11.695	12.288	5
204	98	21	42	0.9975	8.057	8.597	7
205	98	28	42	0.9975	5.967	6.24	5
206	98	35	42	0.9975	4.588	4.604	0
207	98	42	42	0.9975	3.516	3.408	3
208	98	14	56	0.9975	14.406	13.249	8
209	98	21	56	0.9975	9.753	9.558	2
210	98	28	56	0.9975	7.097	7.201	1
211	98	35	56	0.9975	5.407	5.565	3
212	98	42	56	0.9975	4.146	4.369	5
213	98	49	56	0.9975	3.304	3.463	5
214	98	14	70	0.9975	15.901	13.659	14
215	98	21	70	0.9975	10.89	9.968	8
216	98	28	70	0.9975	7.989	7.611	5
217	98	35	70	0.9975	6.125	5.975	2
218	98	42	70	0.9975	4.705	4.779	2
219	98	49	70	0.9975	3.746	3.873	3
220	98	14	84	0.9975	16.615	13.518	19
221	98	21	84	0.9975	11.841	9.827	17
222	98	28	84	0.9975	8.579	7.47	13
223	98	35	84	0.9975	6.442	5.834	9
224	98	42	84	0.9975	4.896	4.638	5
225	98	49	84	0.9975	3.864	3.732	3
226	98	14	98	0.9975	12.401	12.826	3
227	98	21	98	0.9975	8.643	9.135	6
228	98	28	98	0.9975	6.448	6.778	5
229	98	35	98	0.9975	4.966	5.142	4
230	98	42	98	0.9975	3.822	3.946	3
231	98	49	98	0.9975	3.05	3.04	0
232	308	44	44	0.9975	5.236	7.366	41
233	308	44	88	0.9975	7.48	9.429	26
234	308	66	88	0.9975	4.952	6.168	25
235	308	88	88	0.9975	3.675	4.069	11
236	308	44	132	0.9975	10.194	10.941	7
237	308	66	132	0.9975	6.722	7.68	14
238	308	88	132	0.9975	4.956	5.581	13
239	308	110	132	0.9975	3.837	4.115	7
240	308	132	132	0.9975	3.033	3.036	0
241	308	44	176	0.9975	12.153	11.902	2
242	308	66	176	0.9975	8.003	8.642	8
243	308	88	176	0.9975	5.87	6.543	11
244	308	110	176	0.9975	4.52	5.076	12
245	308	132	176	0.9975	3.566	3.997	12
246	308	154	176	0.9975	2.888	3.176	10
247	308	44	220	0.9975	14.469	12.312	15
248	308	66	220	0.9975	9.488	9.052	5
249	308	88	220	0.9975	6.928	6.953	0
250	308	110	220	0.9975	5.312	5.486	3
251	308	132	220	0.9975	4.175	4.407	6
252	308	154	220	0.9975	3.359	3.586	7

Table B.5: Table of experimental results for scaling the KNN metric by predicting ν, part 5

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
253	308	44	264	0.9975	16.09	12.171	24
254	308	66	264	0.9975	10.45	8.911	15
255	308	88	264	0.9975	7.526	6.812	9
256	308	110	264	0.9975	5.694	5.345	6
257	308	132	264	0.9975	4.433	4.266	4
258	308	154	264	0.9975	3.545	3.445	3
259	308	44	308	0.9975	11.154	11.479	3
260	308	66	308	0.9975	7.358	8.219	12
261	308	88	308	0.9975	5.415	6.12	13
262	308	110	308	0.9975	4.182	4.653	11
263	308	132	308	0.9975	3.304	3.574	8
264	308	154	308	0.9975	2.677	2.753	3
265	812	116	116	0.9975	3.329	4.85	46
266	812	116	232	0.9975	5.24	6.913	32
267	812	174	232	0.9975	3.48	4.433	27
268	812	232	232	0.9975	2.55	2.809	10
269	812	116	348	0.9975	7.101	8.425	19
270	812	174	348	0.9975	4.689	5.945	27
271	812	232	348	0.9975	3.416	4.321	26
272	812	290	348	0.9975	2.619	3.169	21
273	812	348	348	0.9975	2.134	2.311	8
274	812	116	464	0.9975	8.987	9.387	4
275	812	174	464	0.9975	5.922	6.906	17
276	812	232	464	0.9975	4.303	5.282	23
277	812	290	464	0.9975	3.287	4.13	26
278	812	348	464	0.9975	2.617	3.272	25
279	812	406	464	0.9975	2.154	2.611	21
280	812	116	580	0.9975	10.734	9.797	9
281	812	174	580	0.9975	7.04	7.316	4
282	812	232	580	0.9975	5.085	5.692	12
283	812	290	580	0.9975	3.862	4.54	18
284	812	348	580	0.9975	3.057	3.682	20
285	812	406	580	0.9975	2.469	3.021	22
286	812	116	696	0.9975	12.03	9.656	20
287	812	174	696	0.9975	7.784	7.175	8
288	812	232	696	0.9975	5.559	5.551	0
289	812	290	696	0.9975	4.189	4.399	5
290	812	348	696	0.9975	3.291	3.541	8
291	812	406	696	0.9975	2.636	2.88	9
292	812	116	812	0.9975	8.044	8.964	11
293	812	174	812	0.9975	5.309	6.483	22
294	812	232	812	0.9975	3.862	4.859	26
295	812	290	812	0.9975	2.955	3.707	25
296	812	348	812	0.9975	2.361	2.849	21
297	812	406	812	0.9975	1.949	2.188	12
298	1512	216	216	0.9975	2.868	2.477	14
299	1512	216	432	0.9975	4.297	4.54	6
300	1512	324	432	0.9975	2.943	2.762	6
301	1512	432	432	0.9975	2.397	1.575	34
302	1512	216	648	0.9975	5.814	6.052	4
303	1512	324	648	0.9975	3.945	4.275	8
304	1512	432	648	0.9975	2.987	3.087	3
305	1512	540	648	0.9975	2.511	2.23	11
306	1512	648	648	0.9975	2.085	1.581	24
307	1512	216	864	0.9975	7.379	7.013	5
308	1512	324	864	0.9975	4.83	5.236	8
309	1512	432	864	0.9975	3.54	4.048	14
310	1512	540	864	0.9975	3.023	3.191	6
311	1512	648	864	0.9975	2.474	2.542	3
312	1512	756	864	0.9975	2.054	2.035	1
313	1512	216	1080	0.9975	8.749	7.423	15
314	1512	324	1080	0.9975	5.623	5.646	0
315	1512	432	1080	0.9975	4.141	4.458	8

Table B.6: Table of experimental results for scaling the KNN metric by predicting ν, part 6

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
316	1512	540	1080	0.9975	3.235	3.601	11
317	1512	648	1080	0.9975	2.765	2.952	7
318	1512	756	1080	0.9975	2.278	2.445	7
319	1512	216	1296	0.9975	9.78	7.282	26
320	1512	324	1296	0.9975	6.106	5.505	10
321	1512	432	1296	0.9975	4.51	4.317	4
322	1512	540	1296	0.9975	3.405	3.46	2
323	1512	648	1296	0.9975	2.962	2.811	5
324	1512	756	1296	0.9975	2.411	2.304	4
325	1512	216	1512	0.9975	6.609	6.59	0
326	1512	324	1512	0.9975	4.422	4.813	9
327	1512	432	1512	0.9975	3.217	3.625	13
328	1512	540	1512	0.9975	2.732	2.768	1
329	1512	648	1512	0.9975	2.269	2.119	7
330	1512	756	1512	0.9975	1.903	1.612	15
331	42	6	6	0.975	3.388	1.075	68
332	42	6	12	0.975	3.179	3.138	1
333	42	9	12	0.975	2.188	1.514	31
334	42	12	12	0.975	1.641	0.459	72
335	42	6	18	0.975	3.711	4.65	25
336	42	9	18	0.975	2.579	3.026	17
337	42	12	18	0.975	1.975	1.972	0
338	42	15	18	0.975	1.63	1.23	25
339	42	18	18	0.975	1.295	0.68	47
340	42	6	24	0.975	4.661	5.612	20
341	42	9	24	0.975	3.231	3.987	23
342	42	12	24	0.975	2.484	2.933	18
343	42	15	24	0.975	2.037	2.191	8
344	42	18	24	0.975	1.621	1.641	1
345	42	21	24	0.975	1.294	1.22	6
346	42	6	30	0.975	5.144	6.022	17
347	42	9	30	0.975	3.576	4.397	23
348	42	12	30	0.975	2.698	3.343	24
349	42	15	30	0.975	2.169	2.601	20
350	42	18	30	0.975	1.713	2.051	20
351	42	21	30	0.975	1.366	1.63	19
352	42	6	36	0.975	5.525	5.881	6
353	42	9	36	0.975	3.808	4.256	12
354	42	12	36	0.975	2.797	3.202	14
355	42	15	36	0.975	2.189	2.46	12
356	42	18	36	0.975	1.694	1.91	13
357	42	21	36	0.975	1.333	1.489	12
358	42	6	42	0.975	2.339	5.189	122
359	42	9	42	0.975	1.679	3.564	112
360	42	12	42	0.975	1.304	2.51	93
361	42	15	42	0.975	1.078	1.768	64
362	42	18	42	0.975	0.893	1.218	36
363	42	21	42	0.975	0.74	0.797	8
364	98	14	14	0.975	2.327	0.945	59
365	98	14	28	0.975	3.422	3.008	12
366	98	21	28	0.975	2.215	1.426	36
367	98	28	28	0.975	1.604	0.396	75
368	98	14	42	0.975	4.095	4.52	10
369	98	21	42	0.975	2.687	2.938	9
370	98	28	42	0.975	1.98	1.909	4
371	98	35	42	0.975	1.562	1.183	24
372	98	42	42	0.975	1.273	0.645	49
373	98	14	56	0.975	5.31	5.481	3
374	98	21	56	0.975	3.466	3.899	13
375	98	28	56	0.975	2.557	2.87	12
376	98	35	56	0.975	2.012	2.144	7
377	98	42	56	0.975	1.615	1.607	0
378	98	49	56	0.975	1.32	1.194	10

Table B.7: Table of experimental results for scaling the KNN metric by predicting ν, part 7

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
379	98	14	70	0.975	6.19	5.891	5
380	98	21	70	0.975	3.974	4.309	8
381	98	28	70	0.975	2.917	3.28	12
382	98	35	70	0.975	2.282	2.554	12
383	98	42	70	0.975	1.814	2.017	11
384	98	49	70	0.975	1.467	1.604	9
385	98	14	84	0.975	7.089	5.75	19
386	98	21	84	0.975	4.506	4.168	7
387	98	28	84	0.975	3.283	3.139	4
388	98	35	84	0.975	2.537	2.413	5
389	98	42	84	0.975	1.982	1.876	5
390	98	49	84	0.975	1.57	1.463	7
391	98	14	98	0.975	4.709	5.058	7
392	98	21	98	0.975	3.079	3.476	13
393	98	28	98	0.975	2.257	2.447	8
394	98	35	98	0.975	1.774	1.721	3
395	98	42	98	0.975	1.417	1.183	16
396	98	49	98	0.975	1.16	0.771	34
397	308	44	44	0.975	0.776	0.493	36
398	308	44	88	0.975	1.24	2.557	106
399	308	66	88	0.975	0.947	1.118	18
400	308	88	88	0.975	0.802	0.176	78
401	308	44	132	0.975	1.774	4.069	129
402	308	66	132	0.975	1.287	2.631	104
403	308	88	132	0.975	1.075	1.689	57
404	308	110	132	0.975	0.922	1.021	11
405	308	132	132	0.975	0.785	0.523	33
406	308	44	176	0.975	2.297	5.03	119
407	308	66	176	0.975	1.599	3.592	125
408	308	88	176	0.975	1.315	2.65	102
409	308	110	176	0.975	1.118	1.982	77
410	308	132	176	0.975	0.95	1.484	56
411	308	154	176	0.975	0.808	1.1	36
412	308	44	220	0.975	2.787	5.44	95
413	308	66	220	0.975	1.894	4.002	111
414	308	88	220	0.975	1.515	3.06	102
415	308	110	220	0.975	1.28	2.392	87
416	308	132	220	0.975	1.078	1.894	76
417	308	154	220	0.975	0.913	1.511	66
418	308	44	264	0.975	3.207	5.299	65
419	308	66	264	0.975	2.119	3.861	82
420	308	88	264	0.975	1.648	2.919	77
421	308	110	264	0.975	1.371	2.251	64
422	308	132	264	0.975	1.146	1.753	53
423	308	154	264	0.975	0.958	1.37	43
424	308	44	308	0.975	1.919	4.607	140
425	308	66	308	0.975	1.411	3.169	125
426	308	88	308	0.975	1.185	2.227	88
427	308	110	308	0.975	1.014	1.559	54
428	308	132	308	0.975	0.865	1.061	23
429	308	154	308	0.975	0.739	0.677	8
430	812	116	116	0.975	1.816	-0.391	122
431	812	116	232	0.975	2.754	1.672	39
432	812	174	232	0.975	1.814	0.509	72
433	812	232	232	0.975	1.303	-0.264	120
434	812	116	348	0.975	3.781	3.185	16
435	812	174	348	0.975	2.467	2.022	18
436	812	232	348	0.975	1.765	1.248	29
437	812	290	348	0.975	1.327	0.693	48
438	812	348	348	0.975	1.045	0.274	74
439	812	116	464	0.975	4.721	4.146	12
440	812	174	464	0.975	3.044	2.983	2
441	812	232	464	0.975	2.166	2.209	2

Table B.8: Table of experimental results for scaling the KNN metric by predicting ν, part 8

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
442	812	290	464	0.975	1.625	1.654	2
443	812	348	464	0.975	1.265	1.235	2
444	812	406	464	0.975	1.022	0.909	11
445	812	116	580	0.975	5.511	4.556	17
446	812	174	580	0.975	3.54	3.393	4
447	812	232	580	0.975	2.509	2.619	4
448	812	290	580	0.975	1.882	2.064	10
449	812	348	580	0.975	1.454	1.645	13
450	812	406	580	0.975	1.16	1.319	14
451	812	116	696	0.975	6.159	4.415	28
452	812	174	696	0.975	3.879	3.252	16
453	812	232	696	0.975	2.702	2.478	8
454	812	290	696	0.975	2.01	1.923	4
455	812	348	696	0.975	1.533	1.504	2
456	812	406	696	0.975	1.211	1.178	3
457	812	116	812	0.975	4.195	3.723	11
458	812	174	812	0.975	2.719	2.56	6
459	812	232	812	0.975	1.961	1.786	9
460	812	290	812	0.975	1.47	1.231	16
461	812	348	812	0.975	1.154	0.812	30
462	812	406	812	0.975	0.942	0.486	48
463	1512	216	216	0.975	1.58	-1.287	181
464	1512	216	432	0.975	2.209	0.776	65
465	1512	324	432	0.975	1.486	-0.119	108
466	1512	432	432	0.975	1.126	-0.725	164
467	1512	216	648	0.975	2.957	2.289	23
468	1512	324	648	0.975	1.937	1.393	28
469	1512	432	648	0.975	1.471	0.787	46
470	1512	540	648	0.975	1.18	0.345	71
471	1512	648	648	0.975	0.976	0.007	99
472	1512	216	864	0.975	3.808	3.25	15
473	1512	324	864	0.975	2.359	2.354	0
474	1512	432	864	0.975	1.774	1.748	1
475	1512	540	864	0.975	1.419	1.306	8
476	1512	648	864	0.975	1.169	0.968	17
477	1512	756	864	0.975	0.98	0.701	28
478	1512	216	1080	0.975	4.493	3.66	19
479	1512	324	1080	0.975	2.789	2.764	1
480	1512	432	1080	0.975	2.043	2.158	6
481	1512	540	1080	0.975	1.62	1.716	6
482	1512	648	1080	0.975	1.327	1.378	4
483	1512	756	1080	0.975	1.107	1.111	0
484	1512	216	1296	0.975	4.972	3.519	29
485	1512	324	1296	0.975	2.988	2.623	12
486	1512	432	1296	0.975	2.159	2.017	7
487	1512	540	1296	0.975	1.709	1.575	8
488	1512	648	1296	0.975	1.391	1.237	11
489	1512	756	1296	0.975	1.153	0.97	16
490	1512	216	1512	0.975	3.421	2.827	17
491	1512	324	1512	0.975	2.153	1.931	10
492	1512	432	1512	0.975	1.623	1.325	18
493	1512	540	1512	0.975	1.295	0.883	32
494	1512	648	1512	0.975	1.069	0.545	49
495	1512	756	1512	0.975	0.898	0.278	69
496	42	6	6	0.9	1.245	-2.064	266
497	42	6	12	0.9	1.919	-0.001	100
498	42	9	12	0.9	1.351	-0.717	153
499	42	12	12	0.9	1.062	-1.204	213
500	42	6	18	0.9	2.697	1.511	44
501	42	9	18	0.9	1.91	0.796	58
502	42	12	18	0.9	1.467	0.309	79
503	42	15	18	0.9	1.154	-0.048	104
504	42	18	18	0.9	0.925	-0.322	135

Table B.9: Table of experimental results for scaling the KNN metric by predicting ν, part 9

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
505	42	6	24	0.9	3.629	2.472	32
506	42	9	24	0.9	2.564	1.757	31
507	42	12	24	0.9	1.934	1.27	34
508	42	15	24	0.9	1.508	0.913	39
509	42	18	24	0.9	1.196	0.639	47
510	42	21	24	0.9	0.946	0.422	55
511	42	6	30	0.9	4.226	2.883	32
512	42	9	30	0.9	2.922	2.167	26
513	42	12	30	0.9	2.156	1.68	22
514	42	15	30	0.9	1.654	1.323	20
515	42	18	30	0.9	1.311	1.049	20
516	42	21	30	0.9	1.043	0.832	20
517	42	6	36	0.9	4.503	2.742	39
518	42	9	36	0.9	3.075	2.026	34
519	42	12	36	0.9	2.157	1.539	29
520	42	15	36	0.9	1.574	1.182	25
521	42	18	36	0.9	1.209	0.908	25
522	42	21	36	0.9	0.947	0.691	27
523	42	6	42	0.9	2.789	2.049	27
524	42	9	42	0.9	2.016	1.334	34
525	42	12	42	0.9	1.488	0.847	43
526	42	15	42	0.9	1.09	0.49	55
527	42	18	42	0.9	0.849	0.216	75
528	42	21	42	0.9	0.678	-0.001	100
529	98	14	14	0.9	0.439	-2.108	580
530	98	14	28	0.9	0.673	-0.044	107
531	98	21	28	0.9	0.45	-0.747	266
532	98	28	28	0.9	0.348	-1.226	452
533	98	14	42	0.9	1.018	1.468	44
534	98	21	42	0.9	0.666	0.765	15
535	98	28	42	0.9	0.491	0.286	42
536	98	35	42	0.9	0.39	-0.066	117
537	98	42	42	0.9	0.32	-0.336	205
538	98	14	56	0.9	1.375	2.429	77
539	98	21	56	0.9	0.879	1.726	96
540	98	28	56	0.9	0.626	1.247	99
541	98	35	56	0.9	0.49	0.896	83
542	98	42	56	0.9	0.398	0.626	57
543	98	49	56	0.9	0.33	0.412	25
544	98	14	70	0.9	1.585	2.839	79
545	98	21	70	0.9	1.014	2.136	111
546	98	28	70	0.9	0.72	1.657	130
547	98	35	70	0.9	0.556	1.306	135
548	98	42	70	0.9	0.451	1.036	130
549	98	49	70	0.9	0.375	0.822	119
550	98	14	84	0.9	1.748	2.698	54
551	98	21	84	0.9	1.104	1.995	81
552	98	28	84	0.9	0.768	1.516	97
553	98	35	84	0.9	0.583	1.165	100
554	98	42	84	0.9	0.47	0.895	90
555	98	49	84	0.9	0.388	0.681	76
556	98	14	98	0.9	0.971	2.006	107
557	98	21	98	0.9	0.688	1.303	90
558	98	28	98	0.9	0.524	0.824	57
559	98	35	98	0.9	0.424	0.473	11
560	98	42	98	0.9	0.352	0.203	42
561	98	49	98	0.9	0.295	-0.011	104
562	308	44	44	0.9	0.283	-2.261	899
563	308	44	88	0.9	0.438	-0.197	145
564	308	66	88	0.9	0.333	-0.856	357
565	308	88	88	0.9	0.245	-1.307	632
566	308	44	132	0.9	0.59	1.315	123
567	308	66	132	0.9	0.447	0.656	47

Table B.10: Table of experimental results for scaling the KNN metric by predicting ν, part 10

Exp.	s	k	n	L	ν	$\hat{\nu}$	$\%Abs.Pred.Error$
568	308	88	132	0.9	0.336	0.206	39
569	308	110	132	0.9	0.26	-0.127	149
570	308	132	132	0.9	0.21	-0.383	282
571	308	44	176	0.9	0.729	2.276	212
572	308	66	176	0.9	0.548	1.618	195
573	308	88	176	0.9	0.416	1.167	181
574	308	110	176	0.9	0.326	0.835	156
575	308	132	176	0.9	0.266	0.579	117
576	308	154	176	0.9	0.221	0.375	70
577	308	44	220	0.9	0.85	2.686	216
578	308	66	220	0.9	0.626	2.028	224
579	308	88	220	0.9	0.472	1.577	234
580	308	110	220	0.9	0.37	1.245	237
581	308	132	220	0.9	0.301	0.989	228
582	308	154	220	0.9	0.251	0.785	213
583	308	44	264	0.9	0.891	2.545	186
584	308	66	264	0.9	0.651	1.887	190
585	308	88	264	0.9	0.491	1.436	193
586	308	110	264	0.9	0.384	1.104	187
587	308	132	264	0.9	0.313	0.848	171
588	308	154	264	0.9	0.26	0.644	148
589	308	44	308	0.9	0.618	1.853	200
590	308	66	308	0.9	0.479	1.195	149
591	308	88	308	0.9	0.367	0.744	103
592	308	110	308	0.9	0.289	0.412	42
593	308	132	308	0.9	0.237	0.156	34
594	308	154	308	0.9	0.196	-0.048	124
595	812	116	116	0.9	0.347	-2.578	844
596	812	116	232	0.9	0.522	-0.514	199
597	812	174	232	0.9	0.4	-1.083	371
598	812	232	232	0.9	0.323	-1.476	558
599	812	116	348	0.9	0.719	0.998	39
600	812	174	348	0.9	0.552	0.43	22
601	812	232	348	0.9	0.446	0.037	92
602	812	290	348	0.9	0.372	-0.256	169
603	812	348	348	0.9	0.313	-0.483	254
604	812	116	464	0.9	0.906	1.959	116
605	812	174	464	0.9	0.682	1.391	104
606	812	232	464	0.9	0.55	0.998	81
607	812	290	464	0.9	0.458	0.706	54
608	812	348	464	0.9	0.386	0.479	24
609	812	406	464	0.9	0.329	0.297	10
610	812	116	580	0.9	1.04	2.369	128
611	812	174	580	0.9	0.792	1.801	127
612	812	232	580	0.9	0.637	1.408	121
613	812	290	580	0.9	0.529	1.116	111
614	812	348	580	0.9	0.444	0.889	100
615	812	406	580	0.9	0.378	0.707	87
616	812	116	696	0.9	1.134	2.228	96
617	812	174	696	0.9	0.853	1.66	95
618	812	232	696	0.9	0.68	1.267	86
619	812	290	696	0.9	0.559	0.975	74
620	812	348	696	0.9	0.467	0.748	60
621	812	406	696	0.9	0.394	0.566	43
622	812	116	812	0.9	0.793	1.536	94
623	812	174	812	0.9	0.609	0.968	59
624	812	232	812	0.9	0.493	0.575	17
625	812	290	812	0.9	0.412	0.283	31
626	812	348	812	0.9	0.349	0.056	84
627	812	406	812	0.9	0.298	-0.126	142

Bibliography

[1] D. Aguado, A. Ferrer, J. Ferrer, and A. Seco. Multivariate SPC of a sequencing batch reactor for wastewater treatment. *Chemometrics and Intelligent Laboratory Systems*, Article In Press, 2006.

[2] C. Ambrozic, R. Bunkofske, and M. Sanders. The use of multivariate statistical techniques in semiconductor manufacturing. *AEC/APC Symposium*, 2001.

[3] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. A linear method for deviation detection in large databases. *Second International Conference on Knowledge Discovery and Data Mining*, pages 164–169, 1996.

[4] V. Barnett and T. Lewis. *Outliers in statistical data*. John Wiley, 1994.

[5] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. *Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX*, 2000.

[7] Rasmus Bro. PARAFAC. tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38:149–171, 1997.

[8] Rasmus Bro, Claus A. Andersson, and Henk A. L. Kiers. PARAFAC2 Part II. Modeling chromatographic data with retention time shifts. *Journal of Chemometrics*, 13:295–309, 1999.

[9] Tom Brotherton. Anomaly detection for prognostic and health management system development. *IEEE Computational Intelligence Society San Diego Meeting*, February 2006.

[10] Martin D. Buhmann and M. J. Ablowitz. *Radial Basis Functions: Theory and Implementations*. Cambridge University, 2003.

[11] J. D. Carroll and J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. *Psychometrika*, 359:283–31, 1970.

[12] Kevin Chamness, Gregory Cherry, and Daniel Kadosh. Electrical test parameter monitoring: Use of parallel fault detection analysis methods with integrated visualization. *AEC/APC Symposium XVII*, 2005.

[13] P. Chan, M. Mahoney, and M. Arshad. *Learning Rules and Clusters for Anomaly Detection in Network Traffic: In Managing Cyber Threats: Issues, Approaches and Challenges*. Springer, 2005.

[14] Q. Chen, R. J. Wynne, P. Goulding, and D. Sandoz. The application of principal component analysis and kernel density estimation to enhance process monitoring. *Control Eng. Practice*, 8:531–543, 2000.

[15] Gregory Cherry, Richard Good, and S. Joe Qin. Semiconductor process monitoring and fault detection using recursive multi-way PCA based on a combined index. *AEC/APC Symposium XIV.*, September 2002.

[16] Gregory A. Cherry and S. Joe Qin. Multiblock principal component analysis based on a combined index for semiconductor fault detection and identication. *Texas-Wisconsin Modeling and Control Consortium Proceedings*, February 2004.

[17] Gregory A. Cherry and S. Joe Qin. Multiblock principal component analysis based on a combined index for semiconductor fault detection and diagnosis. *IEEE Transactions on Semiconductor Manufacturing*, 19(2):159–172, May 2006.

[18] C. J. Chessari, G. W. Barton, and P. Watson. On the development of a neural network based orthogonal nonlinear principal component algorithm for process data analysis. *in: Proceedings, IEEE International Conference on Neural Networks*, 1995.

[19] L. H. Chiang and E. L. Russell. *Fault Detection and Diagnosis in Industrial Systems.* Springer, London, 2001.

[20] Leo H. Chiang and Richard D. Braatz. Process monitoring using causal map and multivariate statistics: Fault detection and identification. *Chemometrics and Intelligent Laboratory Systems*, 65:159–178, 2003.

[21] 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*, 81:109–119, 2005.

[22] Leo H. Chiang, Randy J. Pell, and Mary Beth Seasholtz. Exploring process data with the use of robust outlier detection algorithms. *J. Process Control*, 13:437–449, 2003.

[23] T. C. Cleason and R. Staelin. A proposal for handling missing data. *Psychometrika*, 40:229–252, 1975.

[24] A. K. Conlin, E. B. Martin, and A. J. Morris. Confidence limits for contribution plots. *Journal of Chemometrics*, 14:725–736, 2000.

[25] R. M. Cormack. A review of classification. *J. Royal Statistical Society, Series A (General)*, 134(3):321–367, 1971.

[26] Kenneth S. Dahl, Michael J. Piovoso, and Karlene A. Kosanovich. Translating third-order data analysis methods to chemical batch processes. *Chemometrics and Intelligent Laboratory Systems*, 46:161–180, 1998.

[27] Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. McGraw-Hill Computer Science Series. IEE Computer Society Press, Las Alamitos, California, 1991.

[28] K. I. Diamantaras and S. Y. Kung. *Principal component neural networks*. Wiley, New York, 1996.

[29] D. Dong and T. J. McAvoy. Batch tracking via nonlinear principal component analysis. *AIChE Journal*, 42(8):2199–2208, 1996.

[30] J. J. Downs and E. F. Vogel. A plant-wide industrial-process control problem. *Computers & Chemical Engineering*, 17:245–255, 1993.

[31] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley Interscience, 2nd edition, 2000.

[32] R. Dunia, S.J. Qin, T.F. Edgar, and T.J. McAvoy. Identification of faulty sensors using principal component analysis. *AIChE Journal*, 42(10):2797–2812, 1996.

[33] T.F. Edgar, S.W. Butler, J. Campbell, C. Pfeiffer, C. Bode, S.B. Hwang, K.S. Balakrishnan, and J. Hahn. Automatic control in microelectronics manufacturing: Practices, challenges, and possibilities. *Automatica*, 36(11):1567–1603, 2000.

[34] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *In Data Mining for Security Applications*, 2002.

[35] Rodrigo Fernandez, Jerome Kodjabachian, Martial Baudrier, Thomas Cochet, Florence Majorel, and Franois Pasqualini. Multivariate density estimation applied to abnormal wafer detection at parametric tests. *5th European AEC/APC Conference - Dresden*, April 2004.

[36] Giancarlo Ferrari-Trecatea, Marco Musellic, Diego Liberatid, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39:205–217, 2003.

[37] Wing-Kam Fung. Outlier diagnostics in several multivariate samples. *The Statistician*, 48(1):73–84, 1999.

[38] N. Gallagher, B. Wise, S. Butler, D. White, and G. Barna. Development and benchmarking of multivariate statistical process control tools for a semiconductor etch process: Improving robustness through model updating. *IFAC ADCHEM*, pages 78–83, 1997.

[39] Brian E. Goodlin, Duane S. Boning, Herbert H. Sawin, and Barry M. Wise. Simultaneous fault detection and classification for semiconductor manufacturing tools. *201st Meeting of the Electrochemical Society, International Symposium on Plasma Processing XIV*, 413, May 2000.

[40] J. A. Greenwood and D. Durand. Aids for fitting the gamma distribution by maximum likelihood. *Technometrics*, 2(55), 1960.

[41] Ali S. Hadi. Identifying multiple outliers in multivariate data. *J. Royal Statistical Society. Series B (Methodological)*, 54(4):761–771, 1992.

[42] Gerald J. Hahn and Samuel S. Shapiro. *Statistical Models in Engineering*. John Wiley & Sons, Inc, 1994.

[43] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[44] Mohamed-Faouzi Harkat, Gilles Mourot, and Jose Ragot. Nonlinear PCA combining principal curves and RBF-networks for process monitoring. *Proceedings of the 42nd IEEE Conference on Decision and Control Maui, Hawaii USA*, December 2003.

[45] Stefan Harmeling, Guido Dornhege, David Tax, Frank Meinecke, and Klaus-Robert Muller. From outliers to prototypes: Ordering data. *Neurocomputing*, 69:1608–1618, 2006.

[46] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers Phonet*, 16:1–84, 1970.

[47] J. Hartigan and M. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[48] T. Hastie and W. Stuetzle. Principal curves. *JASA*, 84:502–516, 1989.

[49] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[50] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.*, 24(6):417–441, 1933.

[51] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *Signal Processing Magazine, IEEE*, 10(1):8–39, January 1993.

[52] J. Edward Jackson. *A User's Guide to Principal Components*. Wiley-Interscience, New York, 1991.

[53] J. Edward Jackson and Govind S. Mudeholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349, August 1979.

[54] I. T. Jolliffe. *Principal component analysis*. Springer-Verlag, 1986.

[55] Eamonn J. Keogh and Michael J. Pazzani. Derivative dynamic time warping. *in First SIAM International Conference on Data Mining, Chicago, IL*, 2001.

[56] Henk A. L. Kiers. Some procedures for displaying results from three-way methods. *Journal of Chemometrics*, 14:151–170, 2000.

[57] Henk A. L. Kiers, Jos M. F. Ten Berge, and Rasmus Bro. PARAFAC2 Part I. A direct fitting algorithm for the PARAFAC2 model. *Journal of Chemometrics*, 13:275–294, 1999.

[58] Edwin Knorr and Raymond Ng. Algorithms for mining distance-based outliers in large datasets. *Proceedings of the VLDB Conference*, 1998.

[59] Edwin M. Knorr and Raymond T. Ng. Finding intensional knowledge of distance-based outliers. In *The VLDB Journal*, pages 211–222, 1999.

[60] K.A. Kosanovich, K.S. Dahl, and M.J. Piovoso. Improved process understanding using multiway principal component analysis. *Industrial and Engineering Chemistry Research*, 35:138, 1996.

[61] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991.

[62] Jong-Min Lee, Chang Kyoo Yoo, and In-Beum Lee. Fault detection of batch processes using multiway kernel principal component analysis. *Computers and Chemical Engineering*, 28:1837–1847, 2004.

[63] B. Lennox, G. A. Montague, H. G. Hiden, G. Kornfeld, and P. R. Goulding. Process monitoring of an industrial fed-batch fermentation. *Biotechnology and Bioengineering*, 74(2):125–135, 2001.

[64] Weihua Li, H. Henry Yue, Sergio Valle-Cervantes, and S. Joe Qin. Recursive PCA for adaptive process monitoring. *J. Process Control*, 10(5):471–486, October 2000.

[65] J. F. MacGregor, C. Jaeckle, C. Kiparissides, and M. Koutodi. Process monitoring and diagnosis by multiblock PLS methods. *AIChE Journal*, 40:826–828, 1994.

[66] J.F. MacGregor and T. Kourti. Statistical process control of multivariate processes. *Control Engineerig Practice*, 3(3):403–414, 1995.

[67] John F. MacGregor, Honglu Yu, Salvador Garcia Munoz, and Jesus Flores-Cerrillo. Data-based latent variable methods for process analysis, monitoring and control. *Computers and Chemical Engineering*, 29:1217–1223, 2005.

[68] K. V. Mardia. Mahalanobis distances and angles. In P. R. Krishnaiah, editor, *Multivariate Analysis - IV*, pages 495–511. North-Holland, 1977.

[69] Markos Markou and Sameer Singh. Novelty detection: A review: Part 1: Statistical approaches. *Signal Processing*, 83:2481–2497, 2003.

[70] E. B. Martin, A. J. Morris, M. C. Papazoglou, and C. Kiparissides. Batch process monitoring for consistent production. *Computers & Chemical Engineering*, 20:S599–S604, 1996.

[71] A. Maulud, D. Wang, and J. A. Romagnoli. A multi-scale orthogonal nonlinear strategy for multi-variate statistical process monitoring. *J. Process Control*, 16:671–683, 2005.

[72] P. Miller, R. Swanson, and C. Heckler. Contribution plots: The missing link in multivariate quality control. *Appl. Math. and Comp. Sci.*,, 8:775–792, 1998.

[73] Ben. C. Mitchell and Donald. S. Burdick. Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies. *Journal of Chemometrics*, 8:155–168, 1994.

[74] Tom Mitchell. *Machine Learning*. McGraw Hill, 1996.

[75] D. C. Montgomery. *Introduction to Statistical Quality Control*. John Wiley, New York, 2001.

[76] Philip R. C. Nelson, John F. MacGregor, and Paul A. Taylor. The impact of missing measurements on PCA and PLS prediction and monitoring applications. *Chemometrics and Intelligent Laboratory Systems*, 80:1–12, 2005.

[77] Philip R. C. Nelson, Paul A. Taylor, and John F. MacGregor. Missing data methods in PCA and PLS: Score calculations with incomplete observations. *Chemometrics and Intelligent Laboratory Systems*, 35:45–65, 1996.

[78] P. Nomikos and J.F. MacGregor. Monitoring of batch processes using multi-way principal component analysis. *AIChE Journal*, 40:1361–1375, 1994.

[79] P. Nomikos and J.F. MacGregor. Multivariate SPC charts for monitoring batch processes. *Technometrics*, 37(1):41–59, February 1995.

[80] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biology*, 15:267–273, 1982.

[81] K. Pearson. On lines and planes of closest fit to systems of points in space. *Phil. Mag.*, 2(11):559–572, 1901.

[82] Leonid Portnoy. Intrusion detection with unlabeled data using clustering. Master's thesis, Columbia University, 2000.

[83] S. Joe Qin. Statistical process monitoring: Basics and beyond. *Journal of Chemometrics*, 17:480–502, 2003.

[84] S. Joe Qin, Gregory Cherry, Richard Good, Jin Wang, and Christopher A. Harrison. Semiconductor manufacturing process control and monitoring: A fab-wide framework. *J. Process Control*, 16:179–191, 2006.

[85] S. Joe Qin and Ricardo Dunia. Determining the number of principal components for best reconstruction. *J. Process Control*, 10:245–250, 2000.

[86] S. Joe Qin, Sergio Valle, and Michael J. Piovoso. On unifying multiblock analysis with application to decentralized process monitoring. *Journal of Chemometrics*, 15:715–742, 2001.

[87] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *In Bell Laboratories*, pages 427–438, 2000.

[88] Stefan Rannar, John F. MacGregor, and Svante Wold. Adaptive batch monitoring using hierarchical PCA. *Chemometrics and Intelligent Laboratory Systems*, 41:73–81, 1998.

[89] Konrad Rieck. The zeta anomaly score. `http://ida.first.fraunhofer.de/~rieck/zeta/`, 2006.

[90] Ida Ruts and Peter Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, 23:153–168, 1996.

[91] E. Sanchez and B. R. Kowalski. Tensorial calibration: II. Second-order calibration. *Journal of Chemometrics*, 2:265–280, 1988.

[92] John J. Scheible, Ronald E. Swanson, Paige Miller, and Robert W. Hopkins. Method of controlling a manufacturing process using multivariate analysis. Patent 5442562, August 1995.

[93] Bernhard Scholkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Technical Report MSR-TR-99-87, Microsoft Research*, 1999.

[94] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[95] W.A. Shewhart. *Economic Control of Quality of Manufactured Product*. Van Nostrand, Princeton, N.J., 1931.

[96] A. Smilde, R. Bro, and P. Geladi. *Multi-way Analysis: Applications in the Chemical Sciences*. John Wiley & Sons, 2004.

[97] A. K. Smilde. Three-way analyses: Problems and prospects. *Chemometrics and Intelligent Laboratory Systems*, 15:143–157, 1992.

[98] A. K. Smilde, Y. Wang, and B. R. Kowalski. Theory of medium-rank second-order calibration with restricted-tucker models. *Journal of Chemometrics*, 8:21–36, 1994.

[99] Age K. Smilde. Comments on three-way analyses used for batch process data. *Journal of Chemometrics*, 15:19–27, 2001.

[100] Gerhard Spitzlsperger, Carsten Schmidt, Guenther Ernst, Hans Strasser, and Michaela Speil. Fault detection for a via etch process using adaptive multivariate methods. *IEEE Transactions on Semiconductor Manufacturing*, 18(4):528–533, November 2005.

[101] L. R. Tucker. *In Contributions to Mathematical Psychology.* Holt, Rinehart and Winstor, 1964.

[102] S. Valle, W. Li, and S. J. Qin. Selection of the number of principal components: A new criterion with comparison to existing methods. *Industrial and Engineering Chemistry Research*, 38:4389–4401, 1999.

[103] Sergio Valle-Cervantes. *Plant-wide Monitoring of Processes Under Closed-loop Control.* Ph.D. Dissertation, University of Texas at Austin, 2001.

[104] V. Vapnic. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[105] J. Westerhuis, S. Gurden, and A. Smilde. Generalized contribution plots in multivariate statistical process monitoring. *Chemometrics and Intelligent Laboratory Systems*, 51:95–114, 2000.

[106] Western Electric Company. *Statistical Quality Control Handbook.* Delmar Printing Company, Charlotte, NC, 1956.

[107] Conny Wikstrom, Christer Albano, Lennart Eriksson, Hakan Friden, Erik Johansson, Ake Nordahl, Stefan Rannar, Maria Sandberg, Nouna Kettaneh-Wold, and Svante Wold. Multivariate process and quality monitoring applied to an electrolysis process Part I. Process supervision with multivariate control charts. *Chemometrics and Intelligent Laboratory Systems*, 42:221–231, 1998.

[108] B. E. Wilson, E. Sanchez, and B. R. Kowalski. An improved algorithm for the generalized rank annihilation method. *Journal of Chemometrics*, 3:493–498, 1989.

[109] David J. H. Wilson, George W. Irwin, and Gordon Lightbody. RBF principal manifolds for process monitoring. *IEEE Transactions on Neural Networks*, 10(6):1424–1434, 1999.

[110] B. M. Wise and N. B. Gallagher. The process chemometrics approach to process monitoring and fault detection. *J. Process Control*, 6(6):329–348, 1996.

[111] B. M. Wise and N. L. Ricker. Recent advances in multivariate statistical process control: Improving robustness and sensitivity. *Proc. IFAC ADCHEM Symp.*, pages 125–130, 1991.

[112] Barry M. Wise, Neal B. Gallagher, Stephanie Watts Butler, Daniel D. White Jr, and Gabriel G. Barna. A comparison of principal component analysis, multiway principal component analysis, trilinear decompo-

sition and parallel factor analysis for fault detection in a semiconductor etch process. *Journal of Chemometrics*, 13:379–396, 1999.

[113] Barry M. Wise, Neal B. Gallagher, and Elaine B. Martin. Application of PARAFAC2 to fault detection and diagnosis in semiconductor etch. *Journal of Chemometrics*, 15:285–298, 2001.

[114] H. Wold. *Nonlinear estimation by iterative least squares procedures. In Research Papers in Statistics, David F (ed.)*. Wiley, New York, 1966.

[115] S. Wold. Cross validatory estimation of the number of components in factor and principal component analysis. *Technometrics*, 20(4):397–406, November 1978.

[116] S. Wold. Exponentially weighted moving principal component analysis and projection to latent structures. *Chemometrics and Intelligent Laboratory Systems*, 23:149–161, 1994.

[117] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37–52, 1987.

[118] S. Wold, P. Geladi, K. Esbensen, and J. Ohman. Multi-way principal components and PLS analysis. *Journal of Chemometrics*, 1:41–56, 1987.

[119] Svante Wold, Nouna Kettaneh, Hakan Friden, and Andrea Holmberg. Modelling and diagnostics of batch processes and analogous kinetic experiments. *Chemometrics and Intelligent Laboratory Systems*, 44:331–340, 1998.

[120] Chang Kyoo Yoo, Peter A. Vanrolleghem, and In-Beum Lee. Nonlinear modeling and adaptive monitoring with fuzzy and multivariate statistical methods in biological wastewater treatment plants. *J. Biotechnology*, 105:135–163, 2003.

[121] H. Yue and S. J. Qin. Reconstruction-based fault identification using a combined index. *Ind. Eng. Chem. Res.*, 40:4403–4414, 2001.

[122] H. Henry Yue, S. Joe Qin, Richard J. Markle, Chris Nauert, and Michael Gatto. Fault detection of plasma etchers using optical emission spectra. *IEEE Transactions on Semiconductor Manufacturing*, 13:374–385, 2000.

[123] Yousheng Zeng and P. K. Hopke. Methodological study applying three-mode factor analysis to three-way chemical data sets. *Chemometrics and Intelligent Laboratory Systems*, 7:237–250, 1990.

Vita

Kevin Andrew Chamness was born in Fort Wayne, IN on September 16th, 1977, the son of Robert Chamness and Katherine Chamness. He spent his childhood growing up in Nashville, TN, Greenville, SC, and finally in Raleigh, NC. After completing high school at the North Carolina School of Science and Mathematics in Durham, North Carolina in 1995, he entered North Carolina State University in Raleigh. While in the chemical engineering department here, he worked as a co-op for Philip Morris, USA in expanded tobacco manufacturing and with Biogen in pharmaceutical fermentation production. He received a Bachelor of Science in Chemical Engineering as a university valedictorian in May of 1999. In August of 1999, he moved to Austin to begin a doctoral degree in chemical engineering at the University of Texas. During the summer of 2000, he began working with Tokyo Electron America in the advanced process control group for plasma etch. In December of 2001 he earned his Masters of Science and Engineering in Chemical Engineering from the University of Texas. In February of 2004 he began work in the advanced process control technology group at Advanced Micro Devices (AMD). In April of 2006 he was moved to AMD's flash memory spin off, Spansion, to be the initial member of a new advanced process control technology group. In July of 2006, Kevin became engaged to Sarah Hileman and plans to get married upon the completion of his degree on February 3rd, 2007.

Permanent address: 7121 Hart Lane #2117
Austin, Texas 78731

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.