Applications of Probabilistic Machine Learning Models to Semiconductor Fabrication

Christopher Ilic Lang

S.B., Massachusetts Institute of Technology, 2015
M.Eng., Massachusetts Institute of Technology, 2017

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Signature of Author: _____

Department of Electrical Engineering and Computer Science
October 22, 2021

Certified by: _____

Duane S. Boning
Clarence J. LeBel Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: _____

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Abstract

Semiconductor fabrication relies heavily on the precision and accuracy of its individual processes in order to meet device requirements. If left unchecked, variations in these processes can lead to decreased performance and yield of the final product. While analysis and control of these variations have been employed for decades, recent developments in machine learning have introduced a wide variety of new methods that can potentially be used to better model, monitor, and control these processes. These methods offer the possibility of being more powerful, scalable, and accurate than traditional process control methods.

While many machine learning methods are promising, unique aspects of semiconductor fabrication create challenges for many machine learning approaches. In particular, the high cost of semiconductor fabrication often results in data limited scenarios, as collecting large quantities of data can be infeasibly expensive. Because of this limitation, we investigate the use of probabilistic methods in a variety of semiconductor fabrication settings. These methods are often less prone to overfitting compared to alternative machine learning methods, while still being flexible enough to model complex systems. Specifically, we investigate the application of probabilistic machine learning methods in four distinct case studies.

First, we study virtual metrology systems, with two goals in mind. Our first goal is to define a virtual metrology framework that allows us to better understand the sources of error commonly seen in these systems. This framework relates the recipe, chamber, sensor, and wafer variables, and incorporates two common sources of error: observability errors and concept drift. Our second goal is to then use this framework to develop our own modeling approach that is well suited to model systems where these errors are present. Our solution is a Bayesian approach that is similar to the traditional Kalman filter; however, it models the relationship between two variables, as opposed to an unknown system state.

We then investigate a probabilistic method for optimizing dose uniformity in ion implantation systems. A common approach for improving dose uniformity relies on adjusting the implantation time across the wafer in order to compensate for beam variations. Here, we learn these variations, then solve for a set of compensating times. Our approach is comprised of two components, a modeling component and an optimization component. The modeling component is similar to the probabilistic method we use for modeling virtual metrology systems, but also incorporates prior beliefs tailored to the ion

implantation setting. The optimization component then uses our forward model to improve dose uniformity given physical constraints of the tool and process. We compare this method to the prior existing industry tuning method, and see significant improvements in tuning time, process throughput, and tuning success.

Next, we investigate probabilistic anomaly detection methods, which we use to detect process faults as they occur. These methods use process sensor information to determine whether the current process is operating nominally or not. We use kernel density estimation methods to estimate probability distributions for the sensor signals under normal operating conditions; These distributions are then used to determine the likelihood that a process is operating nominally. The approach is shown to compare favourably to a number of traditional process control methods, including statistical process control, one-class support vector machines, as well as variational auto encoder based anomaly detection methods.

Finally, we investigate the use of Bayesian optimization and Gaussian process models to improve thickness uniformity in sputtering deposition processes. Here, we use Gaussian processes to model the thickness uniformity in sputtering deposition processes as a function of both chamber configuration and recipe parameters. This model is used in an iterative manner to find parameters that meet the desired uniformity requirements. Our modeling technique compares favourably to a number of standard regression approaches, including polynomial models, multivariate splines, gradient boosted regression trees, and a number of different deep learning architectures.

While these four case studies each consider a unique application of probabilistic methods to semiconductor fabrication, two key themes run throughout. First, we find that these probabilistic methods are less prone to overfitting in data limited scenarios compared to many alternative methods. The inherent regularization provided by priors and observation noise estimates is key to the success of these methods. Second, the incorporation of process or domain specific knowledge is crucial to training with limited data. Understanding the underlying system, structuring the approach accordingly, and making minor approximations reduces the complex original problems to a simpler form, enabling effective application of probabilistic machine learning methods.

Thesis Supervisor: Duane S. Boning

Title: Clarence J. LeBel Professor of Electrical Engineering and Computer Science

Acknowledgements

Without the support of my colleagues, friends, and family, I would not have been able to complete this PhD and thesis. I am deeply grateful to have all of them in my life, and for their continued support.

First, I would like to thank Professor Duane Boning for being my advisor during this PhD program. His advice has been instrumental in producing the following work, and his patience and enthusiasm have made graduate school a pleasure. I could not have asked for a better advisor, and will always remember and benefit from what he has taught me. I am deeply grateful for the opportunity to work with him during graduate school and for all that I have learned while working together.

Additionally, I would like to thank other members of the MIT community for their support during graduate school. First, I would like to thank Luca Daniel and Tamara Broderick for being on my thesis committee. Their advice and assistance has been very helpful, and their contribution is greatly appreciated. Additionally, I would like to thank the other members of Professor Boning's group who have been a pleasure to work with, and have provided invaluable ideas that made this work possible. I have greatly benefitted from working with Hongge Chen, Sally El-Henawy, Zhengxing Zhang, Fan-Keng Sun, Dylan Grullon, David Amirault, Damien Martin, and many others. Finally, I would like to thank all the member of the MTL community for their assistance and for creating a community that I am grateful to be a part of. In particular, I'd like to thank Kurt Broderick and Vicky Diadiuk for their help, and for being particularly enjoyable to work with.

Next, I would like to thank all of the industry collaborators, Applied Materials, Lam Research, and Analog Devices, as well as the Machine Intelligence for Manufacturing and Operations group, who have provided funding and advice necessary to complete this work. Without them, this work would not have been possible. In particular, I'd like to thank Rick Sprenkle, Eric Wilson, Ana Samolov, Alexander Jansen, Sima Didari, Prashanth Kothnur, John Yamartino, Ramana Veerasingam, Bruce Lawler, Jack Dillon, Ash Al Dujaili, John Ruth, Peter Cardillo, Perry Alfred, Alan Bowers, and Adrian McKiernan for their help and advice.

Finally, I would like to thank my friends and family for supporting me outside of academia. My friends Joe Bergeron, Carlos Cruz, Spencer Wilson, Matthew Stringfellow, Gretchen Eggers, Gianna Reza-Ortega, Charlie Drucker, Theia Henderson, Peter Downs, Meghan Cum, Mateo Williams, Alex List and Matthew Migliozzi, among many others, as

well as my girlfriend, Cordelia Avery, have made life outside of academia incredibly valuable, and I am grateful to have all of them in my life.

Most importantly, I would like to thank my parents, Jeffrey Lang and Marija Ilić, for supporting and encouraging me throughout my life. I owe them everything I have, and without their support, patience, encouragement and love I would not be where I am today. Their advice and guidance have been invaluable, both inside and outside of academia, and I am deeply grateful to have them as parents. I love them both deeply and dedicate this thesis to them.

Bibliographic Notes

The main contents of this thesis have been submitted to peer reviewed journals. These include:

- **Modeling and Optimizing the Impact of Process and Equipment Parameters in Sputtering Deposition Systems Using a Gaussian Process Machine Learning Framework** – IEEE Transactions on Semiconductor Manufacturing (TSM).
- **One Class Process Anomaly Detection Using Kernel Density Estimation Methods** – IEEE Transactions on Semiconductor Manufacturing (TSM). Process-Level Machine Learning Applications in Semiconductor Manufacturing Special Section.
- **Understanding and Improving Virtual Metrology Systems Using Bayesian Methods** – IEEE Transactions on Semiconductor Manufacturing (TSM).
- **Intelligent Optimization of Dosing Uniformity in Ion Implantation Systems** – IEEE Transactions on Semiconductor Manufacturing (TSM). Process-Level Machine Learning Applications in Semiconductor Manufacturing Special Section.

Table of Contents

List of Figures

1 Introduction

In this chapter, we review key background information, as well as present the structure of the thesis. First, we discuss semiconductor fabrication in general, and highlight key challenges related to process variation present in this field. We then discuss our approaches for overcoming these challenges, and highlight key themes seen throughout this thesis. Next, we present the structure of the thesis itself. Finally, we conclude by surveying common applications of machine learning to semiconductor fabrication, and review the probabilistic machine learning methods which we later employ.

1.1 Semiconductor Fabrication

Semiconductor fabrication is the process of manufacturing integrated circuits (ICs). These mass-produced devices are used in nearly all electronics, and have increased our communication and computing power enormously. A key factor in this advancement is the gradual reduction in device sizes, often expressed as Moore's Law: every two years, the average number of transistors per unit area doubles [1]. This reduction in device size enables faster, smaller and more energy efficient devices and ultimately leads to the improved computing power we see today [2].

These ICs are typically fabricated in a series of individual steps, or processes, common examples of which are deposition, lithography, etching, oxidation, and ion implantation. A critical problem in the continued improvement of IC performance are variations in these fabrication processes. If left unchecked, these variations can lead to decreased device performance, or even non-functional devices [3]–[7]. Common process variations include alignments, geometries, roughness, and material properties that differ from the intended designs.

As devices continue to shrink, these variations must be controlled to an increasing degree. For example, physical variations, such as alignments, or device geometries, must be reduced as devices shrink in order to keep them proportional to the device itself. Therefore, there is an ever growing need for more tightly controlled processes.

While these variations have been studied for decades, a new class of algorithms, those based on machine learning, offer a powerful alternative to model and compensate for these variations. Machine learning algorithms have had an enormous impact on a variety

of fields, such as computer vision [8], [9], economics [10]–[12], and natural language processing [13], [14].

While these methods are promising, unique aspects of semiconductor fabrication hinder the adoption of these methods. A key major difference between this application and others is the availability of data. In many applications, data is ample; stock market records are gathered and freely available and the internet provides a nearly limitless supply of text for natural language processing. However, because semiconductor fabrication and metrology is extremely costly, data may be scarce in these scenarios [15], [16]. In the case studies that we will focus on, the available data ranges from only a handful of training examples, up to a hundred examples, which is often not sufficient for many machine learning algorithms. For this reason, machine learning algorithms that perform well in these data limited scenarios are particularly valuable tools for modeling and compensating variations in semiconductor fabrication.

In this thesis, we show that a subclass of machine learning methods, those based on probabilistic methods, are particularly well suited for this task. These methods offer two main advantages over comparable methods. First, their probabilistic nature provides inherent regularization that helps prevent overfitting in data limited scenarios. For example, we will see that a variation of Kalman filters outperforms traditional fitting methods for linear systems. A second advantage of probabilistic machine learning methods is their flexibility and ability to model complex functions. While probabilistic algorithms include a wide variety of methods, many of these are capable of approximating a function to an arbitrary degree. Additionally, these functions can be non-parametric, and their accuracy increases as more training data becomes available. This is desirable as it allows the model accuracy to scale based on the available data, unlike many deep learning methods which have a fixed structure. For example, we will demonstrate the use of Gaussian processes to model highly complex and non-linear sputtering deposition processes, and use kernel density estimators to model complex likelihood distributions.

Throughout this thesis, we will see two common themes. The first is the advantage that probabilistic machine learning methods offer over alternative approaches. We will see that these probabilistic methods benefit from the inherent regularization that these methods provide, both in the form of prior belief, and through the explicit incorporation of estimated noise. The second major theme is the incorporation of domain specific knowledge into our modeling approaches. Some machine learning approaches can be

structured in order to take advantage of our existing knowledge, again allowing for more accurate training when data is limited. This may be accomplished through belief priors, intelligent feature selection, or assumptions about underlying relationships. Both of these themes are critical to modeling complex processes with little training data.

1.2 Thesis Structure

This thesis is structured in six chapters. In the remainder of Chapter 1, we give an overview of relevant background information. We start with an introduction to our problem, and outline the structure of the thesis. We then highlight common problems in semiconductor manufacturing that machine learning methods are well suited to solve, and review the most common solutions to these problems. The following four chapters then each cover a unique case study. Each of these chapters looks at an important challenge seen in semiconductor manufacturing, and we present a probabilistic machine learning solution to each of these problems.

In Chapter 2, we explore virtual metrology applications. These applications estimate key device or wafer properties using sensor information gathered during processing. We first create a probabilistic framework that incorporates many of the errors commonly seen in these applications, then apply a Bayesian modeling method similar to Kalman filters that is well suited to the presence of these errors.

In Chapter 3, we investigate a process optimization task. Here, we use an iterative Bayesian tuning algorithm to rapidly improve dose uniformity for ion implantation tasks. This algorithm is divided into a modeling component that resembles our Bayesian modeling method from Chapter 2, and an optimization component that is framed as a constrained optimization problem. This method is shown to outperform an existing industry method in a variety of metrics.

In Chapter 4, we explore the use of kernel density estimation methods for fault detection. These methods estimate probability distributions using historical examples, and we use them to model the probability distributions of sensor outputs during normal operating conditions. If incoming data is seen as unlikely under these distributions, we flag them as anomalous, allowing us to detect potential faults during processing. This method is compared to traditional statistical process control methods, as well as deep learning and one-class support vector machine based anomaly detection methods.

In Chapter 5, we then consider a second process optimization task. Here, we use Gaussian processes and Bayesian optimization to model and compensate for thickness non-uniformities in sputtering deposition processes. This method is compared against other modeling approaches, including polynomial regression, multivariate splines, gradient boosted regression, and a variety of deep learning architectures.

Finally, in Chapter 6, we review the contributions from this thesis and suggest areas for future work. In particular, we emphasise the common themes seen within the case studies, and discuss how these findings may generalize to other applications.

1.3 Machine Learning for Semiconductor Manufacturing

In this section, we highlight typical problems seen in semiconductor manufacturing that machine learning algorithms have the potential to address. As both machine learning and semiconductor manufacturing are broad and deep fields, we focus here several important problems and their corresponding solutions, and present a general taxonomy to help organize these problems and solutions. While there are many applications of machine learning in semiconductor manufacturing, most of these can be placed in one of two broad categories: fabrication monitoring, and fabrication optimization.

The first broad category, monitoring, tracks the conditions of a fabrication environment in order to confirm that they remain within a desired range, and that a desired percentage, or yield, of the final product meets its quality specifications. Three common subclasses of the monitoring category are fault detection, virtual metrology, and predictive maintenance [17]–[20].

The first of these, fault detection, uses production data, typically measurements recorded during processing, such as the temperature, pressure, and gas flow of tools, to identify errors when they occur in fabrication processes. These fault detection methods can again be divided in many ways, but an important distinction is whether they are one-class or multi-class methods. One-class methods only have access to known good data, and determine whether or not new incoming data comes from the same underlying distribution as the example known good data. Multi-class methods have access to both nominal and faulty data, and create classification boundaries between these known sets. As the available data will dictate which of these two sets of methods can applied, the distinction between one-class or multi-class fault detection methodologies is particularly important.

A traditional set of one-class fault detection methods used in the semiconductor industry consists of Statistical Process Control (SPC) methods [21]–[25]. These approaches create control limits for measurements taken during fabrication, and raise alarms when these limits are violated. Recently, other non-parametric methods have been studied, such as One-Class Support Vector Machines (OC-SVMs) [26]–[28], and One-Class Gaussian Processes [29], [30] (OC-GPs) which are one-class extensions of the traditional multi-class classification algorithms. Oftentimes, these data driven methods are used in combination with feature reduction techniques such as Principle Component Analysis (PCA) [31], [32] or Auto-Encoding (AE) neural networks [33], [34] in order to reduce the dimensionality of multi-sensor time series data. Finally, a neural network based approach which uses the discriminator created in Generative Adversarial Networks (GANs) has also been proposed as a pure deep learning approach to one-class fault detection [35], [36].

Multi-class fault detection tasks are simpler than one-class approaches, as they are direct applications of traditional machine learning classification methods. These methods include applications of K-Nearest Neighbour [37][38], SVMs [39], [40], various tree based models [24], [41], as well as artificial Neural Networks (NNs) [42]–[44]. While these rely on more traditional methods, they are less desirable as they require examples of faults to train, which may be unique in nature, and difficult to gather ample examples of before they occur.

Virtual metrology is another application of machine learning used in semiconductor industries. While similar to fault detection, as they both monitor the health of processes or product, virtual metrology estimates key characteristics of a fabricated device based on secondary measurements such as recorded sensor process data. These estimates are less expensive and less invasive than direct measurements and are used for run-to-run control, in order to adjust for changing processes [45], [46]. These methods have been applied to create virtual representations of a variety of processes, such as deposition [47], plasma etch [48], chemical-mechanical polishing [49], spin-coating [50], and electro-chemical plating [51], or can be used to determine properties for fabricated devices [52], [53]. When implementing these models, almost any machine learning based regression model can be used; however, the simplest and most commonly used are polynomial based compact models [48], [49], [54], while neural network based approaches are also used in data-rich scenarios [47], [55], [56].

A final sub-category of machine learning based monitoring applications are predictive maintenance tasks. These tasks predict when a fault will happen, then raise alerts before a fault occurs. While similar to both fault detection and virtual metrology tasks, they are often more difficult to implement as they must predict future conditions, as opposed to analyze current ones [57]. Predictive maintenance applications can take the form of either a regression or classification problem [58], [59]. In the classification case, a machine learning method is used to determine whether a process or tool will fail within some predetermined time window [60], [61], while in the regression case, the method tries to model how much time is left until failure [62], [63]. While this task is of substantial interest, as preventing down-time can significantly improve final cost-efficiency of a fabrication process, it is still relatively unexplored and rarely deployed in practical settings at this point in time [59].

In addition to fabrication monitoring, fabrication optimization is another primary application of machine learning in semiconductor fabrication. These tasks focus on actively improving a fabrication process by modeling the process, then using this model to choose new equipment or process settings that improve the quality of that process. These models may focus on the reduction of spatial and temporal non-uniformities, as these non-uniformities lead to variations in the final device properties, or may focus on improving average process metrics, instead of reducing variations.

One optimization approach is the Response Surface Methodology (RSM) [64]–[68]. These methods begin by creating and conducting a Design of Experiments (DoE), where process settings are explicitly chosen and the output variables of interest are then recorded under each set of conditions. The resulting data is then modelled, typically with a second order polynomial, although any regression model can be used, and this model is used to determine the optimal set of process conditions given some desired specifications. It is worth noting that data is often already collected, and explicit DoEs do not have to be performed [69]. Recently, process optimization using reinforcement learning techniques have been considered alongside RSM based methodologies. One large class of these methods are deep learning based reinforcement learning (RL) methods, which are used in particular for scheduling tasks [70]–[72], as the flexibility in these tasks leave them well suited for the exploration component of RL. Bayesian Optimization (BO) methods are another class of reinforcement learning methods which have been used to optimize aspects of semiconductor fabrication and design [73]–[75]. These methods typically use

Gaussian Processes (GPs) to model, then iteratively optimize black-box functions with the help of an acquisition function, a secondary function that selects future process inputs and balances exploration and exploitation, based on the current GP model.

1.4 Probabilistic Machine Learning Methods

In this section, we give brief overviews of the main probabilistic machine learning methods that we explore in the thesis. These probabilistic methods offer distinct advantages over other machine learning methods such as neural networks, and in particular excel in data limited settings [76]. While neural networks have played a critical role in countless applications, their applicability is not universal. In particular, for data-limited problems, neural networks struggle with overfitting and in these cases, probabilistic models offer an attractive alternative. Because collecting fabrication data is often extremely costly, it is crucial that an accurate model can be created with as little data as possible. Collecting data in many of these fabrication settings often requires manual tuning of a tool followed by fabrication and measurement. Not only does this require costly materials for the fabrication itself, but also requires highly trained engineers to perform the data collection, in addition to the opportunity cost of dedicating a tool towards data collection instead of product fabrication. In total, this makes the cost of data collection larger than is found in many other machine learning settings, such as product recommendations, financial analysis, or natural language processing where data is cheap and is produced regardless of whether or not it is used in a machine learning context. For this reason, probabilistic models provide substantial benefits in many semiconductor fabrication related applications.

We first review Kalman filters, a method that estimates unknown variables using related observations. These unknown variables are modeled as normal random variables, and we update our belief of them over time, and as new observations are taken. Then, we review Gaussian Processes. These are non-parametric kernel based methods that are well suited for small data regression problems. Gaussian Processes represent the outputs of interest as a multivariate normal distribution, whose underlying correlations are a function of their inputs. Finally, we review kernel based density estimation methods. These give likelihood distributions for a random variable based on past historical data.

1.4.1 Kalman Filters

Kalman filter methods are algorithms that estimate latent variables based on directly observed variables and have been used successfully for decades in a wide variety of fields, such as navigation and robotics [77]–[81]. Kalman filters track the parameters of a probability distribution for the latent variable (x), such as the position of an aircraft [82], using secondary measurements, such as the distance to nearby landmarks. Typically, this distribution is assumed to be a normal distribution with mean μ and covariance Σ:

$$x \sim N(\mu, \Sigma). \tag{1}$$

At each subsequent timestep, these random variables are assumed to evolve based on a linear combination of the previous timestep (x_{t-1}), an input to the system (u_t) and some amount of random Gaussian noise (w_t). Here, the transition model A_t determines how the previous state affects the subsequent one, and the control-input model B determines how any system input will affect the state:

$$x_t = A_t x_{t-1} + B_t u_t + w_t. \tag{2}$$

Additionally, observations (z) are made, which are a function of the hidden state. Traditionally, the model assumes a linear relationship between the hidden state and the observation. This relationship is known as the Kalman filter observation model (H), and these measurements are corrupted by some amount of noise, v_t:

$$z_t = H_t x_t + v_t. \tag{3}$$

Under these assumptions, the current belief can be updated after new observations are made, and between timesteps using a Bayesian update. This allows us to model how we expect the variables to evolve over time, and we can update this belief as new information becomes available.

The applications that we will explore focus on inferring an unknown input-output relationship between u_t and z_t. The modeling of an unknown input-output relationship, instead of a traditional state variable, such as the position of an aircraft, requires some adjustment to the traditional Kalman filter. In particular, the observation model (H) will be a function of the system input, as it is the system input in combination with the modeled input-output relationship that determines the value that is observed (z). This results in the observation model $H = kron(u_t, I)$, where $kron(x, y)$ is the Kronecker product of two variables [83] and will be discussed further in the following sections.

In Chapter 2, we will explore the application of Kalman filters to virtual metrology tasks. We will examine a simple but powerful framework for semiconductor fabrication systems, and will show that Kalman filters are an effective method for estimating an unknown input-output relationship under common sources of error. Then in Chapter 3, we will explore applications of Kalman filters to modeling and optimizing implantation dose uniformity in ion implantation systems, and will compare them to conventional tuning approaches.

1.4.2 Gaussian Processes

A Gaussian Process is a probabilistic method that models a collection of outputs, Y, as a multivariate normal distribution, with mean μ, and correlation structure Σ [29]:

$$\begin{bmatrix} Y_1 \\ \dots \\ Y_n \end{bmatrix} \sim N\left(\mu, \begin{bmatrix} \sigma_{11} & \dots & \sigma_{1n} \\ \dots & \dots & \dots \\ \sigma_{n1} & \dots & \sigma_{nn} \end{bmatrix}\right) = N(\mu, \Sigma). \quad (4)$$

A critical part of this distribution is its correlation structure, Σ, which represents how similar these outputs are expected to be to one another. This correlation structure is determined by the similarity of their inputs, X, using a kernel function, $\sigma_{i,j} = K(x_i, x_j)$. Here, the kernel function, K, maps a pair of function inputs, x_i and x_j, to their covariance, $\sigma_{i,j,}$ which determines how correlated these points will be in the normal distribution. Outputs with similar inputs will be highly correlated in the distribution, while dissimilar inputs result in little correlation between their outputs.

Gaussian processes include unobserved outputs, Y_*, in the collection of modeled outputs, along with observed outputs, Y. The combined distribution can then be used to predict the unobserved outputs, Y_*, by calculating the marginal distribution of Y_*:

$$Y_* | \, Y, X, X_* \sim N(\mu^*, \Sigma^*) \quad (5)$$

where

$$\mu^* = K(X_*, X)K(X, X)^{-1}Y \quad (6)$$

$$\Sigma^* = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*). \quad (7)$$

An example predictive distribution generated with a Gaussian Process can be seen in Fig. 1. Here, predictions with inputs similar to past examples have outputs that are highly correlated to those observed outputs. Both their mean values are similar, and the uncertainty of the prediction distribution is low. Conversely, as the similarity to past

example inputs decreases, these distributions become less certain, and the outputs become less correlated.

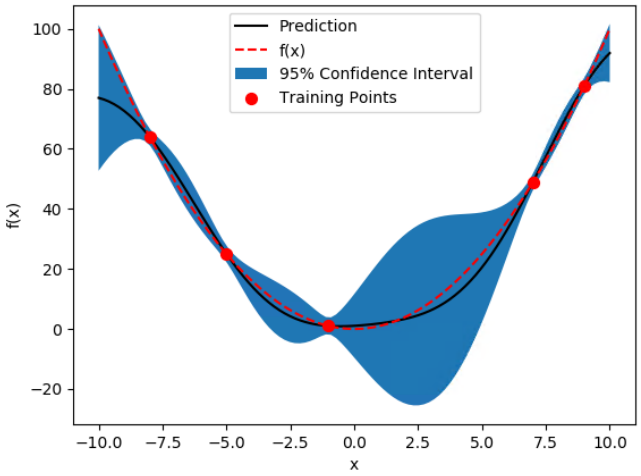


Fig. 1: Example GP model for $\mathbf{f}(\mathbf{x}) = \mathbf{x}^2$ with four training data points.

While the assumption that outputs can be modelled as a Gaussian random variable may seem to imply that they can only model relatively simple functions, Gaussian Processes are indeed capable of modeling complex functions, as the non-parametric nature of the model allows its complexity to grow with the amount of training data. This also allows them to succeed in data limited scenarios, as automatically limiting model complexity when data is limited prevents overfitting, making them well suited for applications where data collection is costly [84], [85]. In Chapter 5, we will explore the application of GPs to modeling and optimizing thickness uniformity in deposition processes and demonstrate these ideas.

1.4.3 Kernel Density Estimation

Kernel density estimation (KDE) is a non-parametric method to estimate the underlying probability density for a random variable based on a set of data samples. These methods use a kernel function $K(x_1, x_2)$ to relate discrete past observations (x) into a continuous density function estimate, $\hat{d}(x)$. Here, each historical example point adds some density to surrounding points in the resulting distribution:

$$\hat{d}(x) = \frac{1}{n}\sum_{i=1}^{n} K(x, x_i). \tag{8}$$

The choice of the kernel function determines exactly how past examples affect the estimated density of nearby points. The most common choice of kernel function is the Gaussian kernel:

$$K(x_1, x_2) = e^{-\frac{1(x_1 - x_2)^2}{2 \quad l^2}}. \tag{9}$$

Alternatively, this method can be viewed as a convolution between the kernel function and past historical examples. Conceptually, the discrete historical points are filtered by the kernel function, producing a continuous estimate of their underlying distribution. An example of an underlying distribution, example data points, and the estimated distribution using KDE can be seen below (Fig.2). Here, each piece of historical data is drawn from the true underlying distribution, and these are then used in combination with KDE to estimate this underlying distribution.

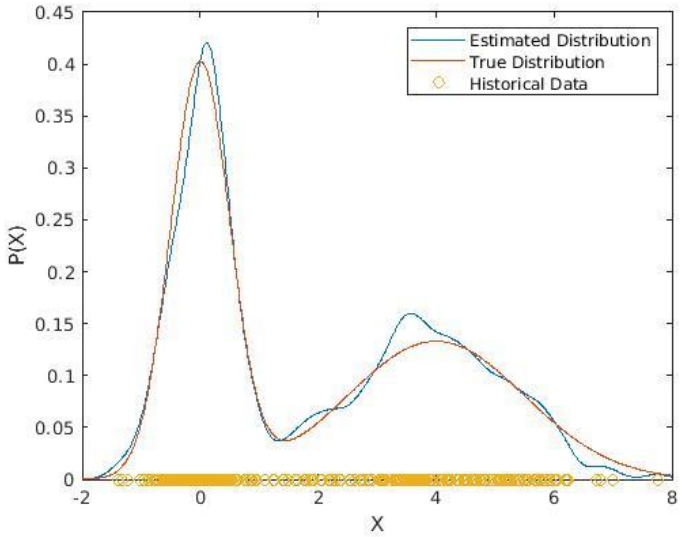


Fig. 2: Example showing KDE estimating underlying distribution of random variable X.

When using the Gaussian kernel, as well as most other kernel choices, a length scale parameter (l) determines the smoothness of the resulting density function. The choice of this parameter is often critical, as too small a choice will result in a rough density function, heavily influenced by the specific historical examples, and choosing too large a length scale may smooth out any true features of the underlying distribution.

In Chapter 4, we will explore the use of these kernel density estimation techniques for fault detection applications in both plasma etch, and ion implantation processes. Here, we will use these methods to generate a likelihood distribution for sensors monitoring the conditions of the tool. Then, when a new process is run, we use new incoming sensor data

to determine the likelihood that it came from the same conditions as the known good process.

2 Understanding and Improving Virtual Metrology Systems with Bayesian Methods

Virtual Metrology (VM) is a well-known approach for monitoring many semiconductor processes. These methods estimate key device and wafer properties without the need for costly direct measurements. Instead, sensors placed inside the tool monitor related properties of the process. Ultimately, these sensor signals are used to infer the wafer properties of interest, allowing for fault detection [86] and run to run control [46], [87], [88]. These methods are used in a wide variety of processes such as plasma etch [85], [89], [90], deposition [47], [91], [92], lithography [93], and chemical-mechanical planarization [94], [95].

An illustrative example of this technique is to monitor emissions from a plasma etch process in order to estimate device geometries [85] (Fig. 3). As material is etched, by-products from the etch itself are released and can be monitored by sensors inside of the tool. A sufficient model can then infer the rate at which this etch takes place based on the sensed concentration of by-products inside of the chamber. Finally, these etch rates can be translated to critical device geometries that determine the performance of the devices. This virtual metrology technique can then be used in place of costly and invasive direct device measurements.

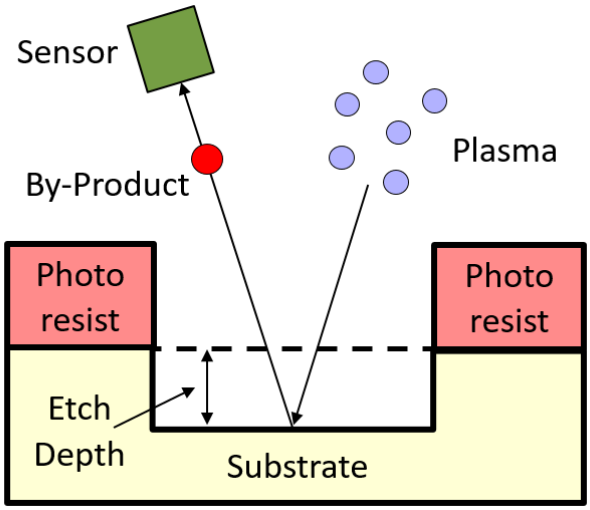


Fig. 3: Illustration of monitored plasma etch process. A sensor monitors etch by-products which can be used to infer etch depth and device geometries.

While many methods are used for virtual metrology, including neural networks [55], [96] and Gaussian Processes [84], [85], many rely on polynomial models [97]–[100]. These relate the collected sensor signals to the wafer properties of interest through a polynomial function. The simplest approach uses a first order polynomial; however, higher orders are frequently used as well.

Nearly all virtual metrology methods, including polynomial models, suffer from two sources of error beyond traditional modeling errors. The first is concept drift, which occurs when the underlying system changes over time [45], [101], [102]. This change in the system decays model performance, as an initially well-fit or optimal model soon becomes out of date. A second, and subtler, source of error comes from limited chamber observability. Here, the available sensors do not give sufficient information in order to infer the wafer properties of interest. An extreme example of this is when available sensor signals are entirely uncorrelated to the wafer properties of interest and provide no relevant information. While these issues are significant obstacles and have been investigated in other contexts such as run to run control [103], little existing work explicitly considers these errors in the virtual metrology context.

Due to these challenges, many virtual metrology modeling approaches perform sub-optimally in practice. Static approaches that do not adjust to changing conditions degrade over time, and methods prone to overfitting are made worse by low chamber observability. Ideally, virtual metrology methods should be created with these challenges in mind.

Problematically, developing these methods may be difficult, as collecting ample semiconductor data is often prohibitively expensive [16]. For this reason, there is a need to generate synthetic data that mimics sensor and wafer data seen in real world settings. This allows for the development and evaluation of model approaches better suited to these common problems.

In this section, we present a framework for better understanding virtual metrology systems. This framework incorporates both drift and observability errors, and we use this framework to develop a virtual metrology method that is well suited for these problems. While the model is linear, our contribution is a Bayesian fitting method that adapts to concept drift, and is less likely to overfit than conventional ordinary least squares (OLS) linear methods. This method is widely applicable, and can be adapted to fit models beyond simple linear models.

In Section 2.1, we present our system framework and the approximations made when analyzing these systems. In Section 2.2, we present our modeling approach developed with the sources of error found in our system framework in mind. This modeling approach is similar to Kalman filters [104]; however, it models a relationship between two variables as opposed to an unknown state variable. In Section 2.3, we present the results of applying our Bayesian modeling approach to our synthetic system, as well as to real-world data, and compare them to traditional approaches. Finally, in Section 2.4, we present conclusions and related future work.

2.1 System Framework

In this section, we present our proposed framework for virtual metrology systems. First, we present the framework itself, then discuss approximations used to simulate these systems. Afterwards, we use these approximations to study the effects of the underlying system, such as the number of system variables, and the rate at which the system drifts.

Our proposed system framework resembles the general process modeling framework originally developed for process flow representations [105], but is developed here for the specific study of virtual metrology systems. The framework contains four sets of system variables: recipe variables, R, which represent inputs to the process, such as desired set-points, chamber variables, C, which capture the operational state of the tool, sensor variables, S, the signals recorded during fabrication, and wafer variables, W, which represent the key wafer properties to be estimated (Fig. 4).

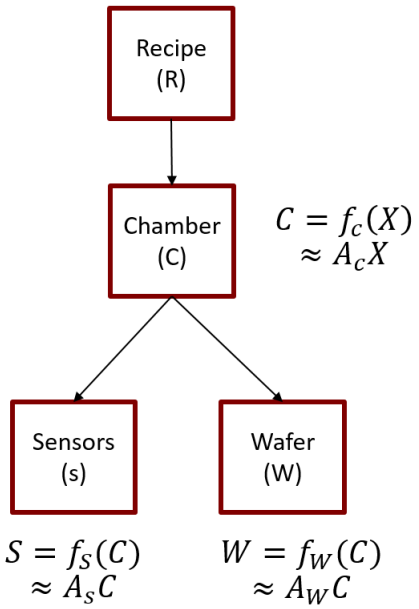


Fig. 4: Proposed virtual metrology model framework showing relationship between recipe, chamber, sensor, and wafer variables.

These four sets of variables are each multivariate and are represented as vectors. Here, the chamber variables are a function of the recipe as the conditions in the chamber are in part determined by set points defined in the recipe:

$$C = f_c(R). \tag{10}$$

Both the sensor and the wafer variables are a function of the chamber variables, since the state of the chamber influences both the sensor signals as well as the wafer variables:

$$S = f_s(C), \tag{11}$$

$$W = f_w(C). \tag{12}$$

Under this framework, the virtual metrology relationship that we estimate, f_m, models W as a function S. This function infers C through S, then produces an estimate of W, $\widehat{W}$. For an invertible f_s this is written as:

$$f_m(S) = \widehat{W}(S) = f_w\left(f_S^{-1}(S)\right). \tag{13}$$

This setup implies one source of error commonly seen in virtual metrology applications, observability errors. If f_s is not invertible, then C cannot be perfectly inferred from the sensor values. If variables in W depend on variables in C that are not observed by S, then those W variables cannot be perfectly estimated from S.

In order to better study these systems, we approximate the underlying functions that define C, S and W, f_C, f_S, and f_W, as linear functions, A_C, A_S and A_W, plus added Gaussian noise, with covariances $\Sigma_{n,c}, \Sigma_{n,s}$, and $\Sigma_{n,w}$, respectively:

$$C = f_c(R) = A_C R + N(O, \Sigma_{n,c}), \tag{14}$$

$$S = f_s(C) = A_s C + N(O, \Sigma_{n,s}), \tag{15}$$

$$W = f_w(C) = A_W C + N(O, \Sigma_{n,w}) \tag{16}$$

While this linear approximation may appear restrictive, process conditions often reside in a relatively small range of values, making a linear model valid in many cases. Additionally, with the presence of simulated system drift, which we will discuss shortly, these linear systems can well-approximate non-linear systems; a drift in these systems represents a change in the operating point of such non-linear systems, and when the model adapts to this moving operating point, the validity of the linear approximation can be maintained. Most importantly, the linear approximation is sufficient for the analysis and understanding of the fundamental issues of concept drift and observability in virtual metrology that we seek to study.

With this approximation, we define the optimal, minimum mean squared error, modeled system, $f_m \approx A_M$ as:

$$f_m(S) = \widehat{W}(S) = A_W A_S^{-1} S = A_M S. \tag{17}$$

When A_S is not invertible, we use the Moore-Penrose pseudoinverse of A_s, A_s^+, to approximate this system. Later, we use this to define the error for the system when an optimal model is used:

$$\widehat{W}(S) = A_w A_S^+ S = A_M S, \tag{18}$$

where

$$A^+ = A^T (A A^T)^{-1} \tag{19}$$

2.1.1 System Drift

In order to incorporate concept and chamber drift into our framework, we assume that the coefficients of the system change over time. Specifically, we assume that each system coefficient in our framework models, A_C, A_S, and A_W, follows a bounded random Gaussian walk [106]. Here, we index the system by time, t, and at each timestep add a

Gaussian drift, with standard deviations $\sigma_{d,C}, \sigma_{d,S}$ and $\sigma_{d,W}$, to each element, i, of A_C, A_S, and A_W, then scale the result:

$$A_{C,i,t+1} = \frac{\sigma_{O,C,i}}{\sqrt{\sigma_{O,C,i}^2 + \sigma_{d,C,i}^2}}\left(A_{C,i,t} + N(0, \sigma_{d,c,i})\right) \forall\, i \in 1{:}\,|A_C|. \tag{20}$$

The scaling constant, $\frac{\sigma_{O,C,i}}{\sqrt{\sigma_{O,C,i}^2+\sigma_{d,C,i}^2}}$, is chosen such that the overall variances of the system coefficients, $\sigma_{O,C}, \sigma_{O,S}$, and $\sigma_{O,W}$ are constant before and after each step in the walk.

Importantly, drifts in these systems are not equivalent. As we will discuss in the following section, drift in A_C results in a drift in the system chamber, and these variables can be inferred through S. Conversely, changes in A_S and A_W represent changes in the relationship between S and W, i.e., concept drift, which decays model performance.

It is important to note that both the drift coefficients, σ_d, as well as the overall variance coefficients, σ_O, have unique values for each of the system coefficients in A. Specifically, $\sigma_{O,C,i}$ refers to the overall system coefficient for the i^{th} coefficient in A_C.

Here, the drift coefficients, σ_d, represent how quickly the coefficients in the modeled systems change, while the overall variance coefficient, σ_O, represents the variance of the variables over all time. A choice of $\sigma_d = 0$ implies a constant system, and a choice of $\sigma_d = \infty$ results in no correlation between timesteps. In Fig. 5, we show examples of a single chamber variable with two different drift rates, $\sigma_{d,c,i}$, as well as two different overall variances, $\sigma_{o,c,i}$.

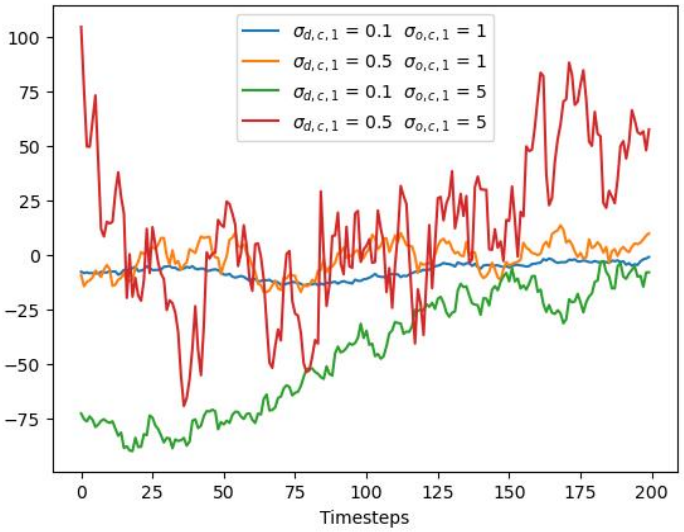


Fig. 5: Example system coefficients following a bounded random walk. Each walk uses either high or low values for the overall system variance, $\sigma_{O,C,1}$, and drift $\sigma_{d,c,1}$,

In addition to modeling system drift, the inclusion of time varying system coefficients also helps replicate non-linear underlying systems. Assuming that the system drift at each time step is relatively small, and the underlying systems are continuous, time varying linear systems can approximate a changing operating point in non-linear systems. For example, if A_C, and thus the chamber variables, C, are slowly changing, we can approximate f_S as the linear system:

$$f_s(C) \approx f_s(C_0) + \nabla_C f_s(C_0)(C - C_0). \tag{21}$$

Then, as the underlying chamber values change, the operating point, C_0, changes, thus changing the gradient $\nabla_C f_s(C_0)$. This change in the linear approximation can be modeled by our proposed framework, making it still applicable to non-linear systems in addition to linear systems.

2.1.2 Effects of System Size

While the number of elements in each set of system variables, R, C, S, and W, can vary, the number of linearly independent elements of S and C are the most important in virtual metrology performance. Intuitively, this makes sense, as our underlying goal is to sense and infer the state of the chamber, and based on that infer parameters of the wafer. The size of R is relatively inconsequential, as regardless of the number of recipe parameters, the same chamber state variables must be inferred. The number of wafer variables, W, is also inconsequential, as we can either model them simultaneously, or independently.

In contrast to R and W, the number of linearly independent elements of C and S have a substantial impact on the performance of a virtual metrology approach. To see this, let us consider the optimal system model. Using the approximations from the previous section, this model is:

$$f_m(S) = \widehat{W}(S) = A_W A_s^+ S. \tag{22}$$

We define the error of our prediction as the difference between our prediction and the true values of W:

$$Error = \ W - \widehat{W} = A_W C - A_W A_s^+ S = A_W (I - A_s^+ A_s) C \tag{23}$$

This expression is similar to the residual in traditional least squares fitting. Here, the term $A_s^+ A_s$ is the projection matrix of A_S, that informs how well we can infer C from S [107]. When A_S has a full column rank, i.e., it has linearly independent columns, $A_S^+ A_S = I$, and there is zero observability error. Practically, this translates to the case where there

are at least as many linearly independent sensors as there are linearly independent chamber variables, and those sensors can be used to fully infer the state of the chamber due to their A_S relationship.

Far more common is the case where there are more linearly independent chamber variables than sensor variables. In most cases, as the number of linearly independent chamber variables increases, or the number of linearly independent sensors decreases, the performance of a virtual metrology method deteriorates, because at some point the state of the chamber cannot be adequately inferred.

To explore this, we first look at the second half of Eq. 23, $(I - A_S^+ A_s)C$. The projection matrix, $A_s^+ A_S$, projects the true values of C onto the null space of $I - A_S^+ A_S$. The difference between the original value of C and the projection, $\hat{C}$, is the error in our inference of C. Adding additional linearly independent terms to C increases the dimensionality of the C space, and decreasing the rank of A_s decreases the dimensionality of this null space. Therefore, both of these changes increases the average projection residual, making predictions more difficult, even if the true underlying systems are known (Fig. 6). Again, this translates to larger observability errors due to insufficient sensors.

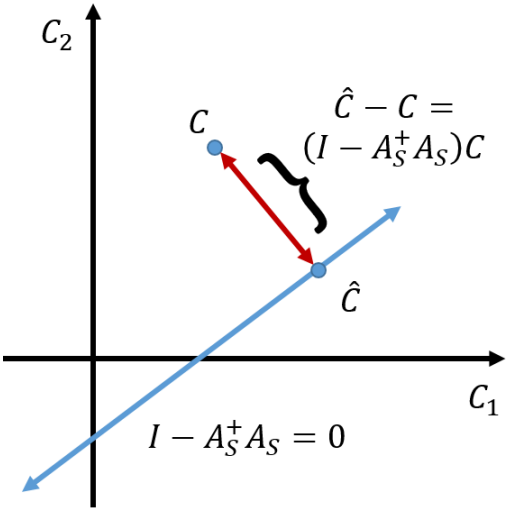


Fig. 6: Illustration showing inference of chamber variables ($\hat{C}$). Addition of more linearly independent chamber variables hinders inference as residual distance between the original and projection point increases.

Crucially, our system setup deviates from traditional least squares fitting in an important way. Inferring C from S is not the goal of a virtual metrology system; instead, virtual metrology seeks to infer W from S. In Eq. 23, $(I - A_S^+ A_s)C$ is the error in C;

however, this is mapped to errors in W through a multiplication with A_W. Therefore, errors in our inference of C are scaled based on how much they impact W. Prediction errors of W are unaffected by an inability to infer irrelevant chamber variables, while inference error in highly relevant chamber variables are highly influential. Practically, this makes sense, as there are a possibly infinite number of variables that describe the state of the chamber, including both useful information such as the average pressure, flow rates, temperature, etc., as well as irrelevant information, such as the exact position of a single atom inside of it, or the name of the tool. The performance of a virtual metrology method is thus influenced both by its ability to infer information about the state of the chamber, and how relevant those states are to the final wafer quantities of interest.

We empirically explore observability implications for virtual metrology by modeling data generated by our proposed system framework. We generate synthetic values of W and S and evaluate our ability to predict W using an optimal ordinary least squares (OLS) model, while adjusting the number of linearly independent elements of C. As we are only interested in determining observability errors in this first analysis, no drift in the modeled systems, A_S and A_W, is present; however, we do drift A_C.

In these experiments, we fix the number of linearly independent sensor variables to 10, then sweep the number of linearly independent chamber variables from 1 to 50. For each set of chamber variables we generate 5000 timesteps of time series wafer and sensor data. Here, the values of A_C, A_S and A_W are drawn from Gaussian distributions, and the drift and variance coefficients are drawn from Poisson distributions. We use the first half of each time series sequence to fit an OLS model for W as a function of S, and then predict W using S for the second half. Fig. 7 shows the average mean squared errors (MSEs), normalized by the system variance and averaged across all timesteps.

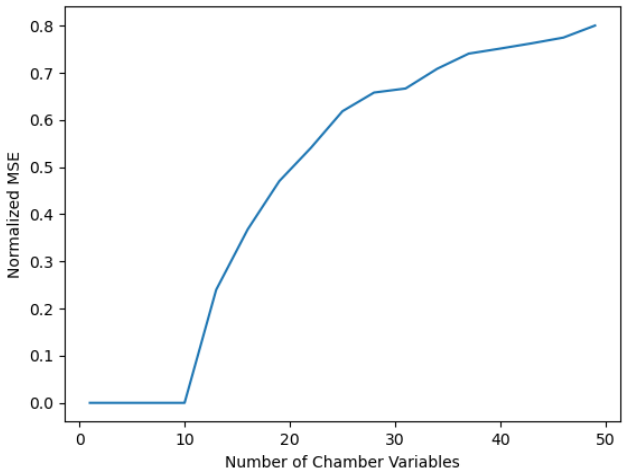


Fig. 7: MSE (normalized) of the optimal model $\widehat{W}(S)$ as a function of the number of linearly independent chamber variables. In each case, the model has access to 10 linearly independent sensor variables.

These results match with our intuition and theory. When there are fewer linearly independent chamber variables than linearly independent sensor variables as linked by A_S, the MSE is effectively 0, as we can perfectly infer the chamber variables. Conversely, as the number of linearly independent chamber variables increases, predicting W becomes more difficult, and the MSE asymptotically approaches the underlying system variance.

2.1.3 Effects of Drift Rates

The rate at which the optimal system model, A_M, drifts influences the rate at which the model performance degrades. Higher drift coefficients, σ_d, represent quicker changes in the underlying systems that lead to more rapid model degradation. However, drift in A_C is distinct from drift in A_S and A_W.

In most cases, drift in A_S and A_W decrease modeling performance; however, drift in A_C does not impact modeling performance. Because the system that we estimate is $A_M = A_W A_S^+$, a change in A_S and A_W changes this optimal model; however, a change in A_C does not affect it.

Intuitively, this makes sense, as a change in A_C results in a change in the system chamber, C, which is possible to infer via S as long as A_s is constant; this C can then be used to predict the effects on W as long as A_W is constant. For non-invertible A_s, and for

non-linear systems where changes in C influence the operating point of f_W and f_S, this may not be the case; however, this effect is second order, and is not considered here.

In practice, we believe that changes in the chamber state are more likely than changes in the *relationship* between $C, S,$ and W, that is, changes in A_S and A_W. For example, changes in S may indicate that the temperature of the chamber (a C variable) has changed, while changes in A_S may represent a degradation of the temperature sensor itself, which is less likely. Critically, changes in W and S do not imply changes in A_W and A_S, respectively, as drifts in the chamber via A_C also impact W and S. In Fig. 8, we plot a synthesized value of W, when A_W is held constant, but A_C drifts. The plot demonstrates the distinction between the wafer variables, W, and the sensor function, A_W.

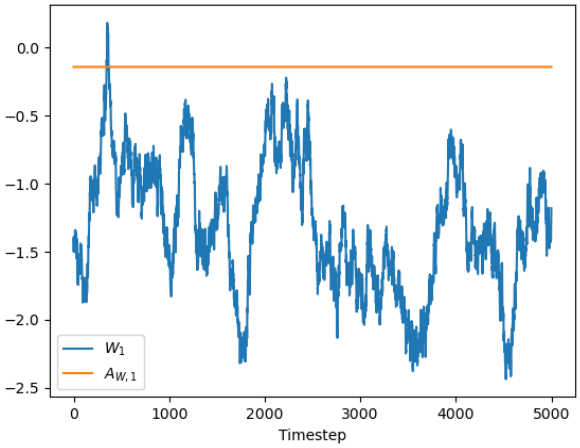


Fig. 8: Wafer variable, W (blue) and system coefficient, A_W, (orange) when the chamber drifts, but the wafer system coefficient is held constant.

To demonstrate the impact of draft rates, σ_d, we simulate data with varying drift rates and observe how a fit OLS model degrades over time. We simulate a system similar to the one described in the previous section. This system has 10 linearly independent chamber and 10 linearly independent sensor variables, 1 wafer variable, and the starting system coefficients are drawn from a Gaussian distribution. The drift coefficients, σ_d, are drawn from a Poisson distribution, but this distribution is scaled in order to observe the effects of varying the drift rates.

For each drift rate, we fit the optimal model for the starting system, according to Eq. 22. We then synthesize S and W while drifting the system, predict W using S and the optimal model, and record the prediction mean squared errors (MSEs). We average these MSEs over 1000 ensembles to produce an average error vs. number of timesteps since

model fit, and fit these results to an exponential function $A - Be^{-\frac{t}{\tau}}$ (Fig. 9). Importantly, the fit exponential decay constant, τ, informs us how quickly the model decays, and we use this as a metric for model decay. Finally, we separately repeat this process while individually varying $\sigma_{d,C}$, $\sigma_{d,S}$, and $\sigma_{d,W}$, and plot the results (Fig. 10).

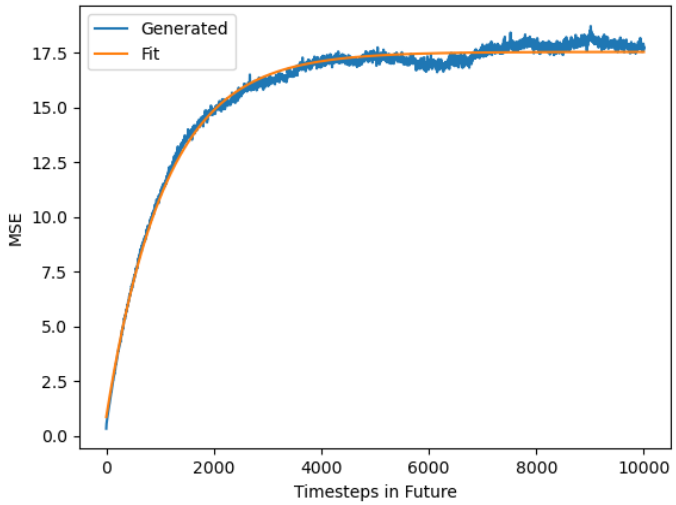


Fig. 9: Experimentally generated MSE vs. number of time steps since model fit, and exponential fit.

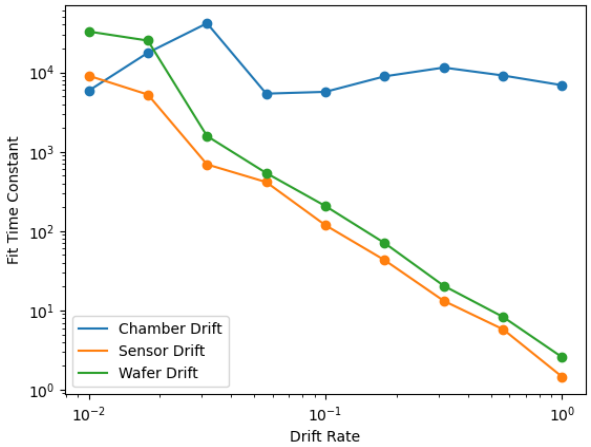


Fig. 10: Mean drift rate in A_C, A_S and A_W vs. fit model decay coefficient. Smaller time constants imply faster model decay.

Increasing the drift rates of A_W and A_S hasten model decay, while adjusting the drift of A_C does not significantly impact the results. As previously described, only A_S and A_W impact the combined system that we aim to model ($A_W A_S^+$), so only drift in these

functions influence the model decay. Interestingly, drift in A_S is slightly more impactful than drift in A_W. We believe this is due to the inversion of A_S in the optimal model, which may amplify the effects of small changes to the system.

2.2 Modeling Framework

In this section, we propose a predictive Bayesian modeling approach that is well suited to the system framework from the previous section. We first present the form of the predictive model, then discuss how to update it as new information is available, and finally, how to decay our belief over time, in order to account for a drifting underlying system.

2.2.1 Model Form

Our proposed model is linear, $\widehat{W}(S) = A_M S$. While the model itself is traditional, our contribution focuses on how the model coefficients are fit. As we will show, a Bayesian fitting method that adjusts to concept drift and is less prone to overfitting can provide significant improvements over traditional fitting methods. Other models can be used in place of the linear model; however, the model updates have to be adjusted accordingly. This modeling approach is similar to a Kalman filter; however, it models a relationship between two variables as opposed to an unknown state variable.

We model our belief of the optimal model, A_M, as a multivariate normal distribution with mean μ_A and covariance Σ_A:

$$A_M \sim N(\mu_A, \Sigma_A). \tag{24}$$

The mean is the most likely estimate for the modeled system, while the covariance captures how likely this belief is, and how the model coefficients are related to one another.

2.2.2 Model Updates

A key part of our modeling approach is updating this belief as new information is available. We use a Bayesian approach that is both less prone to overfitting in the case of observability errors, and is updated over time to deal with concept drift. When new observations of S and W are made, we update our belief of A_M using a Bayesian update:

$$P(A_M|S,W) = \frac{P(W|A_M,S) \cdot P(A_M)}{P(W|S)} \tag{25}$$

This updated likelihood can be written explicitly; however, it must first be simplified. As $P(W|S)$ does not depend on A_M, it is ignored during this calculation. Later we will find that the posterior, $P(A_M|S,W)$ is also a normal random variable, and this constant is determined retroactively:

$$P(A_M|S,W) \propto P(W|A_M,S) \cdot P(A_M) \tag{26}$$

The prior belief of A, $P(A_M)$, provides inherent regularization and prevents overfitting, as our posterior is weighted by how likely the coefficients of A_M are under this prior. Extreme values of A_M that commonly arise when overfitting, are unlikely under both the prior and posterior. This helps combat overfitting due to observability errors, ensuring a better fit model.

As A_M is a matrix, we must vectorize it to explicitly write this prior, $P(A_M)$, as a normal random variable:

$$P\left(A_{M,vec}\right) \propto e^{-\frac{1}{2}\left(A_{M,vec}-\mu_A\right)^T \Sigma_A^{-1}\left(A_{M,vec}-\mu_A\right)} \tag{27}$$

To model the probability of an observation given a model, $P(W|A_M,S)$, we assume normal random noise on the observation of wafer quantities, W, having standard deviation σ_n. We then write the likelihood of these observations given A_M as the likelihood of the residual, $A_M S - W$, given the assumed noise:

$$P(W|S,A_M) \propto e^{-\frac{1}{2}(A_M S-W)^T \Sigma_n^{-1}(A_M S-W)} \tag{28}$$

where Σ_n is a diagonal matrix with all non-zero elements equal to the wafer observation variances:

$$\Sigma_n = \begin{bmatrix} \sigma_n^2 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \sigma_n^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \sigma_n^2 \end{bmatrix} \tag{29}$$

The value of σ_n is a crucial regularization parameter, as it estimates how much of the output is unpredictable due to observability error. For processes with low observability, high values of σ_n act as effective regularization parameters, as the model attributes large portions of the output variance to uninferable "noise," making it less prone to overfitting.

While this update is similar to the traditional Kalman filter update [104], we must make slight adjustments as A_M is a matrix, and not a vector. Multivariate normal

distributions conventionally treat the random variable as a vector, so we use the vectorized version of A_M, $A_{M,vec}$, in Eq. 28. As we use $A_{M,vec}$ in Eq. 28, we must then also use it, as well as the vectorized wafer values, W_{vec}, in Eq. 28. In order to do so, we must rewrite the matrix multiplication using the Kronecker product:

$$(A_M S)_{vec} = S_{kron} A_{M,vec} \tag{30}$$

where S_{kron} is the Kronecker product, as indicated by $\otimes$, of S^T and the identity matrix of size $|S|$, $I_{|S|}$:

$$S_{kron} = S^T \otimes I_{|S|} \tag{31}$$

This allows us to combine the two terms of $P(B|I,T)$:

$$\begin{aligned} P(A_M|S,W) \propto e^{e^{-\frac{1}{2}(A_{M,vec}-\mu_A)^T \Sigma_A^{-1}(A_{M,vec}-\mu_A)}} \\ \times e^{-\frac{1}{2}(S_{kron}A_{M,vec}-W_{vec})^T \Sigma_n^{-1}(S_{kron}A_{M,vec}-W_{vec})} \end{aligned} \tag{32}$$

Finally, these terms can be rearranged into a normal distribution with posterior mean, μ_B^*, and covariance Σ_B^*:

$$P\left(A_{M,vec}|S,W\right) = N(\mu_A^*, \Sigma_A^*) \tag{33}$$

where

$$\Sigma_A^* = \left(S_{kron}^T \Sigma_N^{-1} S_{kron} + \Sigma_A^{-1}\right)^{-1} \tag{34}$$

and

$$\mu_A^* = \Sigma_A^* \left(S_{kron}^T \Sigma_n^{-1} W_{vec} + \Sigma_A^{-1} \mu_A\right). \tag{35}$$

This allows us to update our belief of A_M in a straightforward and efficient manner, as our entire belief is captured in μ_A and Σ_A. As new observations are made, we use this closed form update to incorporate them into our belief in constant time.

2.2.3 Model Decay

In addition to updating our model as new sensor and wafer observations are made, we also decay our belief to account for concept drift. Between model updates, the underlying system is likely to change, and our belief should account for this. When viewing our model as a Kalman filter, this decay is equivalent to the state transition function that describes how our belief at one time, $\mu_{A,t}$ and $\Sigma_{A,t}$, transition to later beliefs, $\mu_{A,t+1}$, $\Sigma_{A,t+1}$.

There are two qualities that are desirable for this function. First, it should leave the mean of the belief, μ_A, unchanged, as we generally do not know how the system will change. Secondly, we want our variance matrix, Σ_A, to increase over time. With this in mind, we use the following state transition function to update our belief from one timestep to the next:

$$\mu_{A,t+1} = \mu_{A,t} \tag{36}$$

$$\Sigma_{A,t+1} = \left(\Sigma_{A,t} + \Sigma_{D,A}\right) \odot \Sigma_{S,A}\,. \tag{37}$$

Here, $\Sigma_{S,A}$ represents a pointwise multiplication, indicated by $\odot$, of our uncertainty, while $\Sigma_{D,A}$ represents a Gaussian drift of the coefficients of the modeled system, A_M.

For a system where the optimal model, A_M, follows a Gaussian random walk, the Gaussian drift decay and model update is the optimal system estimate. However, it is important to note that while the underlying synthetic systems, A_W and A_S, follow a Gaussian walk, the combined system, $A_W A_S^+$, does not necessarily follow a Gaussian walk as well. This is primarily due to the inversion of A_S that may lead to the optimal system heavily deviating from a Gaussian random walk. For this reason, we also include the term $\Sigma_{S,A}$, which supplements the Gaussian drift of the modeled system.

2.2.4 Model Parameter Selection

To apply this model, we must first select its parameters. These model parameters are distinct from the model coefficients A_M, and represent the model's belief of the system drift and observability. These include the model observation noise, σ_n, and the sets of decay drift, $\Sigma_{D,A}$, and scaling, $\Sigma_{S,A}$, coefficients. Here, we select parameters that maximize the validation accuracy of our model on historical data and use these for all future predictions; however, these could also be continuously refit as new data becomes available.

Ideally, parameters could be selected using an Expectation Maximization algorithm (EM), a common approach for optimizing large numbers of parameters [108]. Unfortunately, given our model formulation, the Maximization step is difficult to calculate, and we rely on alternative methods.

Instead, we use a grid search to optimize these parameters. We first choose a discrete set of values for each parameter, then perform a full factorial grid search over these to determine the set that gives the best predictive accuracy when evaluating validation data.

To evaluate the accuracy for a given set of parameters, we iterate through historical data, predict W given S at each timepoint, and average the mean squared errors (MSEs) across all iterations. We decay the model belief before each prediction, and update the model once every U predictions. This infrequent update is important; in practice, observations are not taken with every prediction, as this would defeat the purpose of the predictive virtual metrology model.

As the full factorial search quickly becomes infeasible with large numbers of parameters, we make some approximations. Primarily, we assume that the model drift decay values, $\Sigma_{D,A}$, are equivalent for each coefficient in A_M. A similar assumption is also made for the scaling decay values, $\Sigma_{S,A}$. While these values may not be equal for all coefficients in A_M, this approximation is required to feasibly perform a full factorial search on these coefficients. Additionally, assuming uniform drift rates is often desirable, as all coefficients must change at approximately the same rate in order to appropriately update a linearization of a non-linear function. To implement this, we restrict $\Sigma_{D,A}$ to be a diagonal matrix with a constant value on the diagonal, and $\Sigma_{S,A}$ to be a single constant, and optimize these two constants.

While this parameter selection method provides good empirical results, as will be shown in Section 2.3, improving this fitting method is one opportunity for future work. Specifically, we believe it is possible to select unique parameters for all terms in A_M via an EM algorithm; however, this approach is not yet known. This will likely improve real world performance, as some coefficients are more likely to change over time, while other are likely to remain stable.

2.3 Results

In this section, we present results of our proposed Bayesian modeling methodology. First, we present the results of our predictive model using data synthesized with our proposed system framework under a variety of system conditions. Then, we use our proposed modeling approach to predict key device characteristics taken from a real world industrial setting.

In these scenarios, we compare our proposed modeling approach to a number of different standard linear modeling approaches. These methods include fixed ordinary least squares (FOLS), weighted ordinary least squares (WOLS), and previous observation (PO) methods.

The simplest of these is the previous observation method. Here, the prediction for new wafer values are the last wafer values observed. While simple, this method serves as a baseline approach; a predictive model must perform better than this method in order to be useful.

A second baseline approach is a FOLS model. Here, we fit an OLS model on a set of training observations, then use this fixed model to predict all future wafer values. While rudimentary, models such as these are some of the most frequently used approaches for predicting key device characteristics, and serve as a second baseline comparison. Additionally, the lack of adjustment for concept drift provides further evidence for its necessity, both in the simulated and real world cases.

Finally, we also compare our method to WOLS. This method is a common approach for adjusting linear models to concept drift [109], [110]. Here, linear models are refit using all available data as new observations are made. Critically, recently made observations are given more weight, while older observations less. This reflects our belief that the modeled system is changing over time, and allows for our model to adapt to these changes. In our comparisons, we use an exponential weighting, where the weighting of each sample is $e^{-\frac{t}{\alpha}}$. Here, t represents how much time has elapsed since this sample, and α is a decay constant that represents how quickly we expect the underlying system to drift. Large decay constants represent relatively steady systems, while small decay constants represent quickly changing systems. In practice, we select this parameter in a method similar to the selection for our proposed method. We perform a grid search on a set of likely values, and choose the value that performs best to predict a validation set.

We present the predictive results of our proposed and comparison methods using our synthetic system framework under varying system conditions to show the effects of these conditions, as well as the superior performance of our proposed method in a variety of settings. We adjust the number of linearly independent chamber variables, the drift rates of A_C, A_S and A_W, and the update frequency, U, of our model. By default, we use 10 linearly independent chamber variables, 10 linearly independent sensor variables, 1 wafer variable, and equal drifts in A_C, A_S and A_W and adjust each of these system parameters individually.

In Fig. 11 through Fig. 15, we present the results when sweeping the number of chamber variables from 1 to 30, when sweeping the individual drift rates from a factor of

10^{-3} to 10^{2} of their default values, and when sweeping the update frequency from every prediction to every 10^{th} prediction.

In all experiments, we first generate training and testing data using the specified system. We then select the proposed model parameters, α for the WOLS method, and train the FOLS model using the training data, and determine the model accuracies using the testing data. We iterate through all testing data, predicting W and calculating the MSE at each iteration, then updating the models after every U observations. Finally, we average the MSE error over each iteration and present the results.

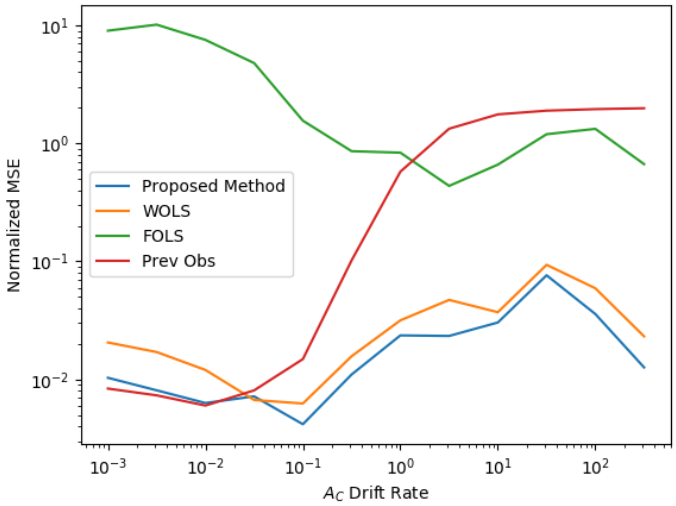


Fig. 11: Empirical normalized MSE as a function of drift rate of A_C.

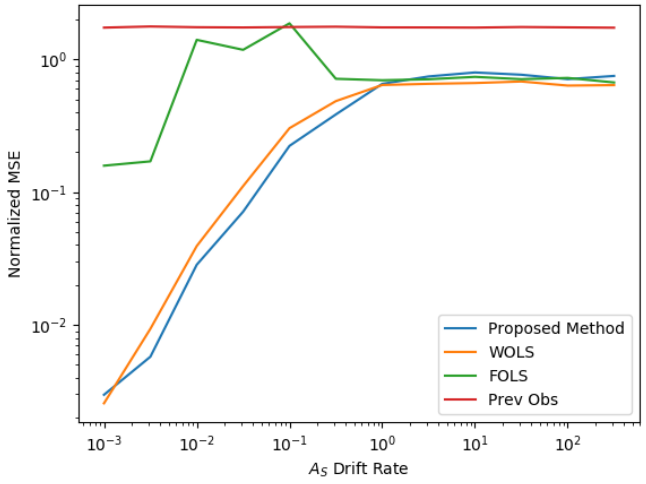


Fig. 12: Empirical normalized MSE as a function of drift rate of A_s.

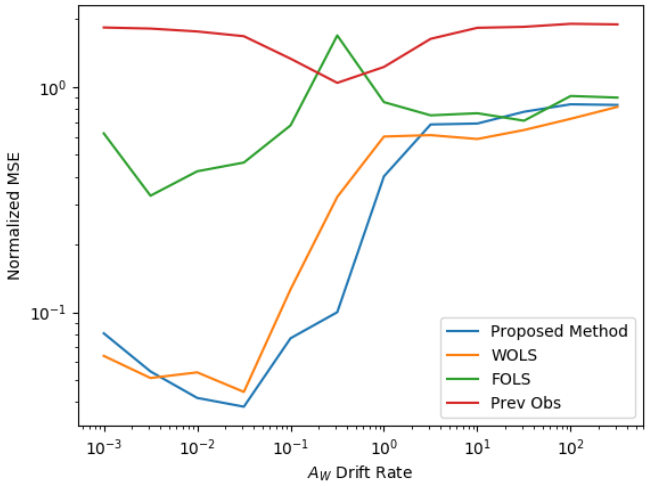


Fig. 13: Empirical normalized MSE as a function of drift rate of A_w.

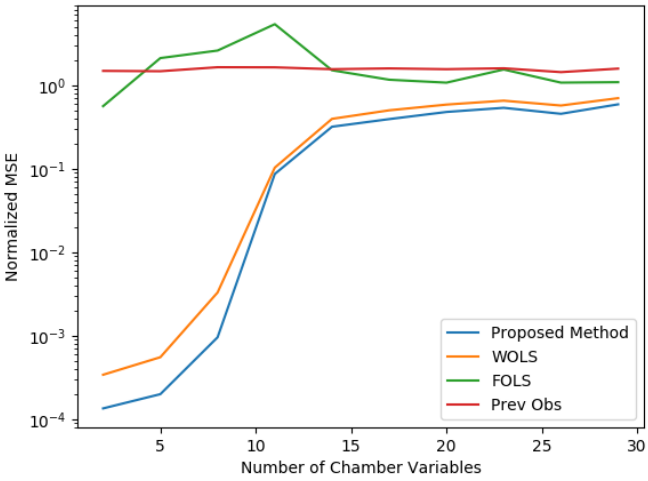


Fig. 14: Empirical normalized MSE as a function of the number of linearly independent chamber variables.

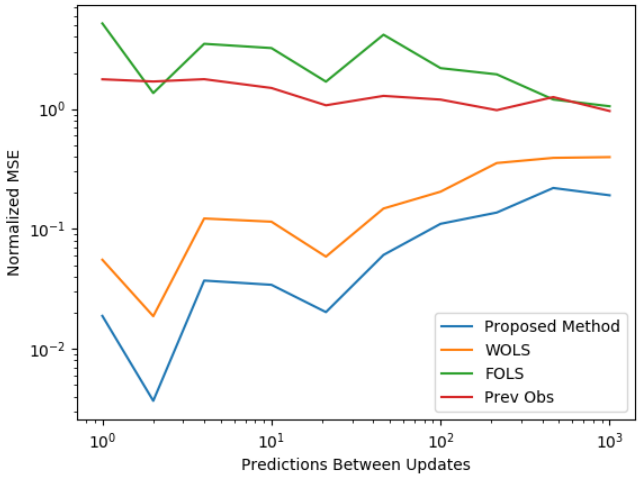


Fig. 15: Empirical normalized MSE as a function of observation frequency.

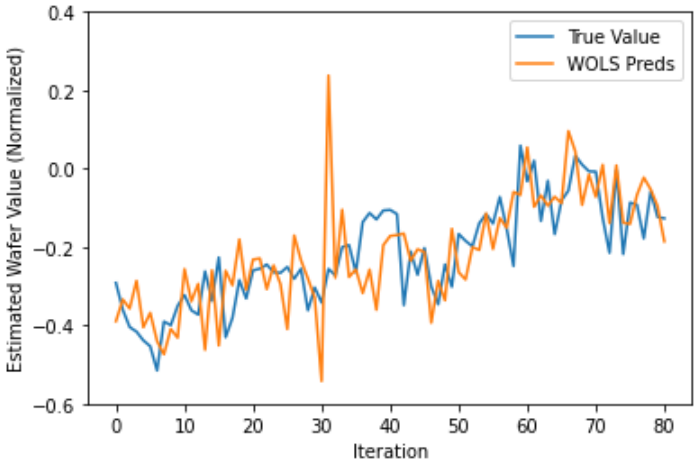
In all of these results, the proposed Bayesian modeling technique performs better than the baseline methods. The fixed OLS model performs worst, as it cannot adapt to changing process conditions. In cases where drift A_S or A_W in the modeled systems is more rapid, this effect is exacerbated as the fixed model becomes out of date more quickly.

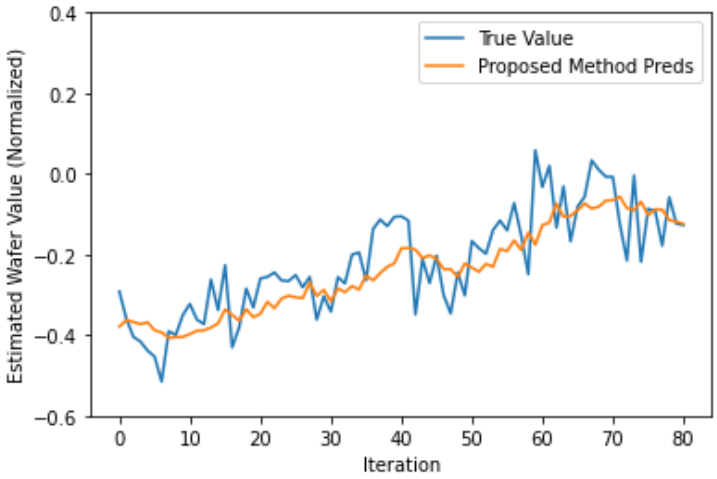
Additionally, the proposed Bayesian method outperforms the traditional WOLS method. We believe this is for two reasons. First, the more expressive model decay function allows the model to adapt to changing process conditions more accurately. Secondly, the Bayesian update and prior help prevent overfitting in the case of limited observability. Specifically, the proposed model noise, σ_n, determines how much of the underlying system cannot be predicted, and helps to prevent overfitting due to low observability.

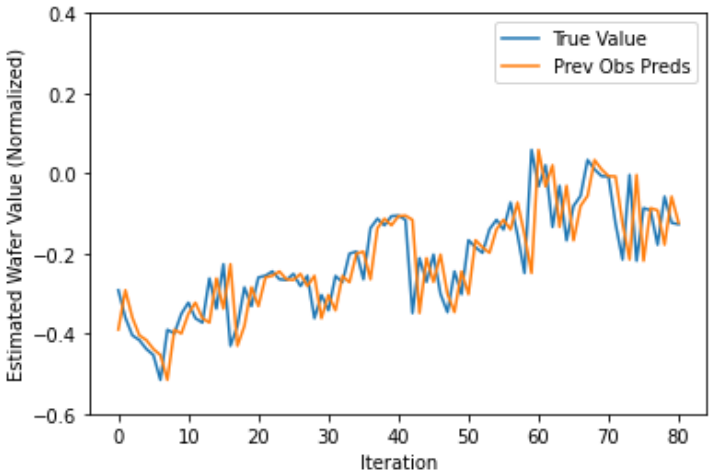
In these results, changing the update frequency U and changing the drift rates of the modeled systems, A_S and A_W produce similar results. Intuitively, this makes sense, as drifting the modeled systems at a faster pace is equivalent to updating a slower moving system more infrequently. Importantly, the reverse of this is also true: systems that drift rapidly can be modeled more effectively with more frequent modeling updates. Therefore, the update frequency of a model can often be chosen such that it leads to the desired model accuracy.

In addition to modeling synthetic data, we also use our proposed and comparison methods to model a key wafer property for real world industrial fabrication data. Here, we model this key value using 50 available sensors and the same approach as described

when modeling synthetic data. We show the predictive accuracies in four different tools in Table 1, and example predictions for all methods for one of these in Fig. 16.







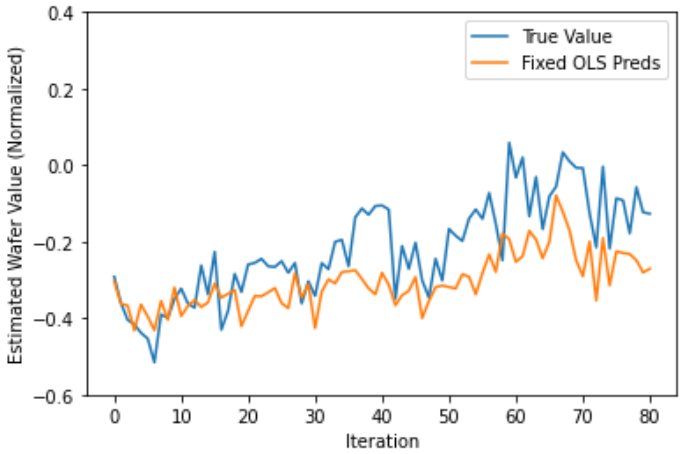


Fig. 16: Predicted and experimental wafer values for proposed and comparison methods, using industrial process sensor and wafer data.

TABLE I
RMSE OF PROPOSED AND COMPARISON METHODS WHEN PREDICTING KEY WAFER VALUE IN FOUR DIFFERENT TOOLS.

Tool Number	Data STD.	Previous Observation	WOLS	Fixed OLS	Proposed Method
1	0.096	0.058	0.076	0.388	0.041
2	0.085	0.065	0.065	0.203	0.061
3	0.087	0.063	0.087	0.227	0.044
4	0.129	0.069	0.083	0.101	0.064

Here, we see similar results to those of our synthetic system. The fixed OLS model performs by far the worst, as it cannot adapt to changing process conditions. Additionally, this error tends to grow over time, indicating that drift is likely present in the underlying real system. The WOLS model also performs poorly, likely to due to overfitting of the sensor data. As there are 50 sensors, substantial amounts of data are required to accurately fit a model. Because the system is also drifting rapidly, sufficient relevant data is never available, and regularization is needed beyond what WOLS provides.

When fitting parameters of our proposed Bayesian model, values are chosen to prevent overfitting in this difficult modeling scenario. While both WOLS and the proposed method each have parameters that estimate the model drift, only the proposed method is tuned for observability errors. The prediction noise, σ_n, informs the model how much of the output is predictable, and how much cannot be inferred due to observability errors. In this real system, it is difficult to fully infer the necessary information to relate all 50

sensors to the output, and the Bayesian model estimates a large observation noise value, leading to less overfitting. This is seen in the less varied, but more accurate predictions of the proposed method compared to the WOLS results.

The improved performance of our proposed method highlights the two key themes of this thesis. First, based on our prior knowledge of virtual metrology systems, we expect two key sources of error, observability errors and concept drift. We then compensate for these sources of error by incorporating probabilistic model decay and expected observability errors into our Bayesian fitting method. These help prevent overfitting while still allowing our model to update to changing system conditions, highlighting the benefit of incorporated process knowledge, and probabilistic modeling approaches.

2.4 Conclusions and future work

In this chapter, we explore virtual metrology for semiconductor fabrication and make two key contributions to this area. First, we propose a framework for simulating these systems that incorporates concept drift as well as observability errors, both of which are expected to be present in real world virtual metrology applications. While this framework does not directly model key wafer properties, it can be used to better understand how these systems function, and can be used to develop virtual metrology modeling techniques. Wafer and sensor data can be synthesized, and improved modeling techniques can be developed using such synthetic data. Finally, we use this framework to draw conclusions about modeling these systems, and in particular study the effects of system drift and chamber observability.

Our second major contribution is a virtual metrology technique to model the key wafer values of interest as a function of sensor values. This Bayesian approach models an unknown linear system as a normal random variable, allowing for Bayesian updates as new observations are available, and is designed with concept drift and observability errors in mind. The proposed method is well suited to model drifting systems, as we decay our belief between updates to reflect an unpredictable change in the underlying system. Finally, we apply this modeling technique to both synthetic and industrial data, and achieve superior results compared to traditional OLS and WOLS methods.

This case study highlights both key themes of this thesis. First, prior process knowledge, such as the expectation of concept drift and observability errors, are incorporated into the system framework. This allows us to design our modeling approach

with the presence of these errors in mind by incorporating model decay and expected observability errors. Secondly, the Bayesian nature of our method helps prevent overfitting caused by observability errors, as it has a prior that limits the model to a feasible range and a noise parameter, σ_n, that assigns some portion of the output variance to unpredictable sources. Both the incorporation of system knowledge, as well as the Bayesian modeling approach are critical to the improved performance compared to traditional modeling methods.

While the proposed system and modeling frameworks are both promising, we believe that improvements to both exist. For the system framework, it is important to investigate how our results change for non-linear systems. While our linear models provide good approximations for continuously and slowly drifting non-linear systems, real world scenarios are often non-linear, and it would be of interest to explicitly investigate the impacts of non-linear systems and non-linear models. Additionally, extensions to this framework that incorporate tool-tool variations may also be useful for examining multiple tools simultaneously. For the modeling framework, an improved drift and noise parameter fitting method could also be considered. Currently, we limit our selection by assuming equal drift rates for all coefficients relating sensor and wafer variables; however, this is likely not the case. Optimizing the parameters for each model coefficient uniquely could increase performance. Finally, we believe that there is a further, unexplored, benefit from the combined modeling and system frameworks. In our analysis, we did not explore the effects of the recipe; however, this is a variable that can be controlled in practice. We believe it may also be possible to incorporate these variables into the modeling approach for combined virtual metrology and run to run control.

3 Optimization of Dosing Uniformity in Ion Implantation Systems

In this chapter, we explore the use of the Bayesian modeling approach presented in the previous chapter to model and optimize dose uniformity in ion optimization processes.

3.1 Introduction

Ion implantation is a fundamental process in semiconductor fabrication [111]–[113]. In this process, ions are accelerated towards a wafer in order to implant dopants in the substrate. This gives the substrate a desired dopant concentration, allowing for either p-type or n-type doping and modifying the conductivity of the substrate.

Commonly, the ion beam is swept across the wafer in order to provide approximately equal dose to all areas of the wafer as illustrated in Fig. 17 [114]. Problematically, methods for directing the ion beam often distortion the shape and intensity of the beam as it is swept across the wafer, as shown in Fig. 18. In these cases, assigning uniform dose time across the wafer leads to non-uniform implantation dose profiles as seen in Fig. 19 [115], [116]. To compensate for this, the implantation time is often spatially varied in order to achieve a uniform implantation dose [117]–[120]. Areas of the wafer with low uncompensated doses are given additional implantation time and those with greater unadjusted dose less.

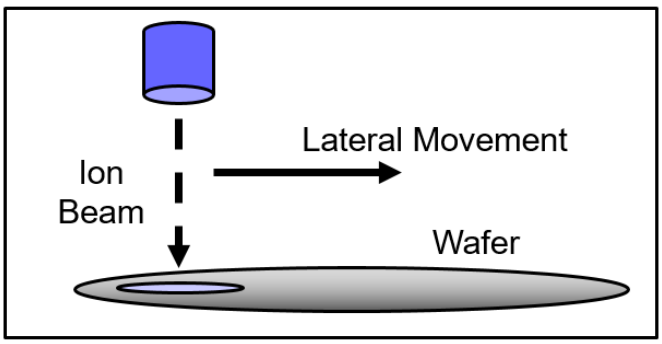


Fig. 17: Illustration of ion beam sweeping across wafer.

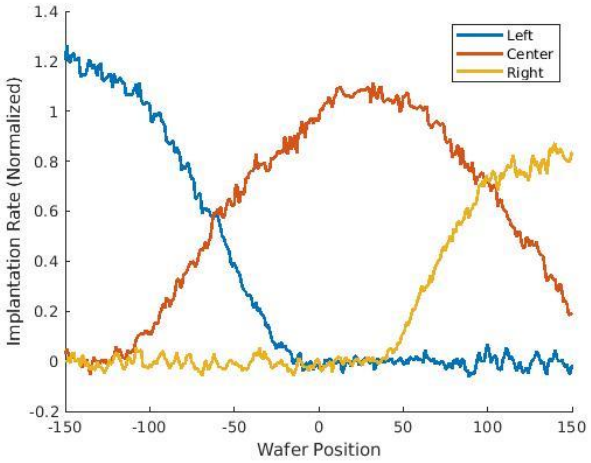


Fig. 18: Experimentally measured implantation rate cross sections when the beam is placed at three different wafer locations. Note the change in intensity as the beam location changes.

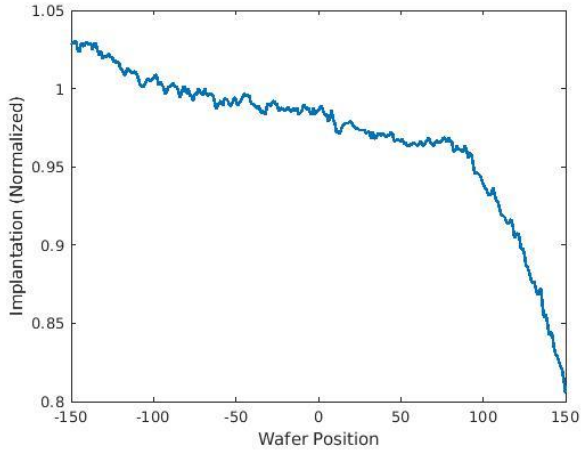


Fig. 19: Cross section of implantation dose profile when sweeping the beam at a constant speed.

A key modeling challenge faced is drift of the system dynamics over the course of successive implantations, preventing a single set of compensating times from being sufficient. When the system dynamics change significantly, previous implantation times no longer meet the uniformity criteria, and a new set of compensating times must be found. Because this drift may occur relatively quickly, minimizing the re-tuning time is critical to reducing loss from tool downtime, and for retaining high yield. Additionally, the presence of a drifting system may also lead to spurious correlations between the model inputs and outputs resulting in model overfitting. We can see the presence of this drift when viewing subsequent implantations using the same tuning times. Below we plot four dose profiles taken immediately after one another, when the same implantation times are used (Fig. 20), confirming that the systems drift between runs.

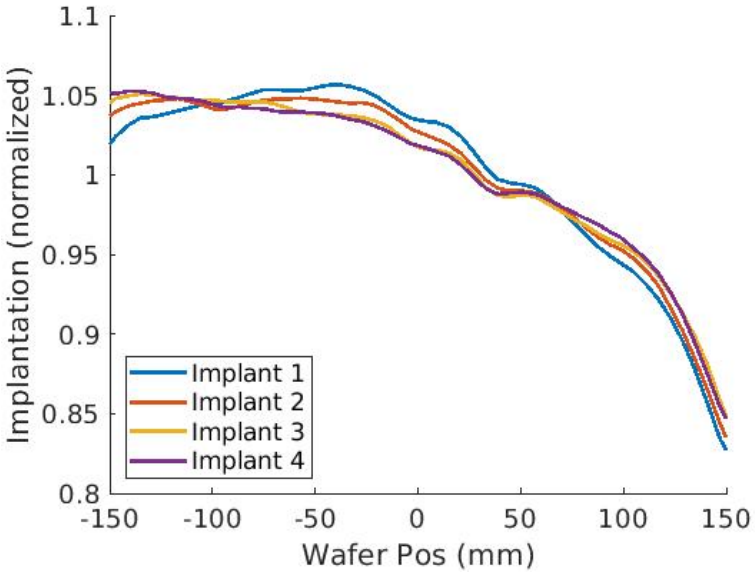


Fig. 20: Sequential implantation profiles using the same set of process times, demonstrating system drift.

This rate of drift is often influenced by the process recipe, which includes parameters such as the ion element, the implantation energy, and mean implantation current. For example, low energy implantations, which are often desirable, are more heavily impacted by these problems [121]–[123]. Additionally, the recipe also influences the amount of required compensation, as some are naturally less uniform and require more extreme compensation. Therefore, improved tuning techniques expand the range of recipes that can feasibly be run, as high variation recipes can still be sufficiently controlled with effective tuning methods.

In this section, we present a machine learning approach to model and rapidly tune implantation dose uniformity through spatial adjustments to the implantation time. Our approach is comprised of a forward model, as well as an optimization component, and is similar to Bayesian Optimization (BO) methods [124]. The forward model is a Bayesian estimate of the *relationship* between the implantation times and the resulting implantation dose profile. This method was presented in the previous section, and is similar to Kalman filters, which are used in a variety of tasks such as position tracking [77], [78], [80]; however, we again track an input-output relationship that deviates from the traditional Kalman Filter method in that the latter only tracks an unknown state variable. Our optimization method uses this inferred relationship to select new implantation times believed to compensate for dose non-uniformities. Here, we define a

constrained optimization problem that minimizes the difference between our desired and predicted profiles.

The combination of these two components results in an effective iterative tuning method. After initializing a starting belief, we solve for compensating implantation times. The resulting implantation dose profile is then measured, and the model updated. If the desired uniformity is not met, the process repeats, updating the model and retuning until a sufficient solution is found. We find that this method converges in fewer iterations than the prior deployed industry method of record, while also finding solutions that give equivalent uniformities with less overall implantation time, increasing the throughput of the system.

In Section 3.2, we discuss the specifics of the modeling approach. We first formalize our model, then discuss additional features, including the prior belief initialization, incorporation of additional measurement types, and adjustments for beam intensity changes in the system. In Section 3.3, we discuss the optimization approach in further detail. In Section 3.4, we present a tuning example that illustrates our modeling and optimization approach. We present the results of our method in Section 3.5, and compare them to the existing industry method of record. Finally, in Section 3.6, we summarize our conclusions and discuss future work.

3.2 Modeling Approach

Here, we present our machine learning model which predicts the implantation dose as a function of the implantation times. First, we present the model itself, and explain how model updates are made using new observations. We discuss how to adjust the model to account for a drifting system. We then discuss our prior belief, which is critical for rapid tuning. Finally, we discuss how to update the model with fast spot measurements, a second implantation measurement that is faster to perform than traditional implantation.

3.2.1 Model

Here, we present the Bayesian predictive model used in our tuning approach. This model predicts the implantation dose across the wafer, I, as a function of the time spent above each point on the wafer, T, and is an application of the probabilistic modeling technique presented in Section 2.

During modeling, we discretize both T and I into discrete elements. Each element of I represents the dose given to a small section of the wafer, while each element of T represents the implantation time spent above a small section of the wafer. During implantation, T is translated into sweep speeds used for each section of the wafer.

Although the wafer dose profile is two-dimensional, we approximate it as one-dimensional. When the beam is swept in one direction, the wafer is quickly moved back and forth in the perpendicular direction giving nearly uniform implantation in that direction. Thus, we only model and compensate for non-uniformities in the beam sweep direction, making T and I one-dimensional.

Both T and I have their own corresponding position vectors, x_T and x_I, that denote the physical positions of their elements relative to the wafer center. For 300mm wafers, the wafer position vector, x_I, takes values between -150mm and 150mm, and each value of x_I represents the wafer positions where dose measurements are made. Similarly, the beam position vector, x_T, represent discretized locations where the beam can be placed. This range is typically larger, as the beam width is non-zero, and off-wafer beams are frequently used to adjust the dose on the wafer edges.

A key assumption we make is that the relationship between T and I is linear, i.e., the sum of two time vectors results in the sum of their dose profiles. Intuitively this make sense, as the implantation dose at each location should be the accumulation of impacting ions. We define a linear relationship between the dose profile, I, the time profile, T, and the implantation rates as a function of beam and wafer location, B:

$$I = BT. \tag{38}$$

The beam matrix, B, describes the relationship between the implantation dose profile and the implantation times. B is a matrix of size $|I| \times |T|$ that gives the implantation rate at each wafer location, x_I, for each potential beam location, x_T. Specifically, each element, $B_{i,j}$, gives the implantation rate at point x_{I_i} when the beam is placed at x_{T_j} as pictured in Fig. 21. Vertical slices of B represent the beam shape at one beam location, while horizontal slices represent the implantation at one wafer location as the beam is swept across the wafer.

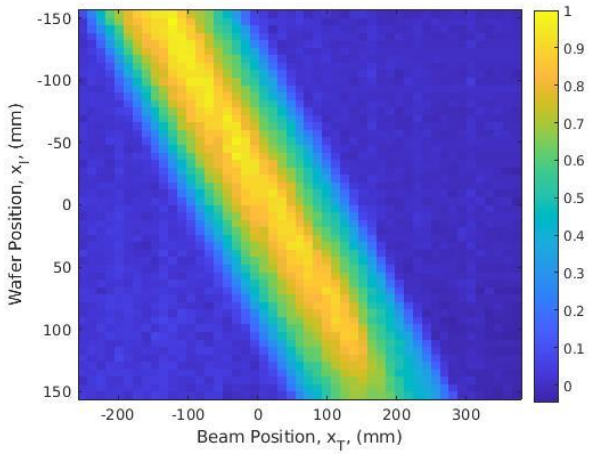


Fig. 21: Experimentally measured B matrix (normalized), that shows the implantation rates as a function of the beam, x_T, and wafer position, x_I.

If B is known, we can predict I for any value of T. Therefore, building a predictive model is equivalent to estimating B. Our modeling approach estimates B using a method similar to Kalman filters [125], and is nearly identical to the virtual metrology approach discussed in Section 2.3. Kalman filters infer unobservable variables through a set of observable variables and are commonly used for position estimation tasks. Here, we use a modified approach, where the inferred variable is the *relationship*, B, between two observed variables, I and T. Specifically, we model B as a multivariate normal random variable with mean μ_B and covariance matrix Σ_B:

$$B \sim N(\mu_B, \Sigma_B). \tag{39}$$

This representation captures our most likely belief of B as well as its uncertainty. The mean of our belief, μ_B, is the most likely value we expect B to take, and the covariance, Σ_B, describes both the confidence in our belief, as well as how the uncertainties are related to one another.

Under this model, we can update our belief as new implantations are observed. When a new implantation observation pair, I, T, is made, we can calculate the posterior belief of B:

$$P(B \mid I, T) = \frac{P(I|T, B) \cdot P(B)}{P(I, T)}. \tag{40}$$

As $P(I, T)$ is a constant that does not depend on B, we ignore it. Later we show that the posterior is again a normal random variable, and the constant can be determined retroactively:

$$P(B|I,T) \propto P(B) \cdot P(I|T,B). \tag{41}$$

In order to explicitly write these probability functions as normal random variables, we vectorize these variables. We write the first half of Eq. 41, our prior belief of B, using the vectorized version of the matrix B, B_{vec}:

$$P(B_{vec}) \propto e^{-\frac{1}{2}(B_{vec}-\mu_B)^T \Sigma_B^{-1}(B_{vec}-\mu_B)}. \tag{42}$$

To define the second half of Eq. 41, $P(I|T,B)$, we assume normal random noise on the observed dose profile, with a standard deviation σ_n. This allows us to write the likelihood of the observation given B as the likelihood of the prediction error under this assumed noise:

$$P(I|T,B) \propto e^{-\frac{1}{2}((BT-I)_{vec})^T \Sigma_n^{-1}((BT-I)_{vec})}. \tag{43}$$

Here, Σ_n is a diagonal matrix with all non-zero elements equal to the implantation observation variance:

$$\Sigma_n = \begin{bmatrix} \sigma_n^2 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \sigma_n^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \sigma_n^2 \end{bmatrix}. \tag{44}$$

While this update is similar to traditional Kalman filter updates, we must make adjustments because the unknown variable is a matrix, instead of a vector. In particular, we rewrite the matrix multiplication in Eq. 43 using the Kronecker product [126]:

$$(BT)_{vec} = T_{kron}\, B_{vec}. \tag{45}$$

Here, T_{kron} is the Kronecker product of T^T and the identity matrix of size $|T|$, $I_{|T|}$:

$$T_{kron} = T^T \otimes I_{|T|}\,. \tag{46}$$

This allows us to combine the two terms of $P(B|I,T)$:

$$\begin{aligned} P(B|I,T) \propto e^{e^{-\frac{1}{2}(B_{vec}-\mu_B)^T \Sigma_B^{-1}(B_{vec}-\mu_B)}} \\ \cdot e^{-\frac{1}{2}(T_{kron}B_{vec}-I)^T \Sigma_n (T_{kron}B_{vec}-I_{vec})}. \end{aligned} \tag{47}$$

Finally, we rewrite this expression as a new normal distribution with posterior mean, μ_B^*, and covariance Σ_B^*:

$$P(B|I,T) = N(\mu_B^*, \Sigma_B^*). \tag{48}$$

where

$$\mu_B^* = {\Sigma_B^*}^{-1}\left(T_{kron}^T \Sigma_n^{-1} T_{kron} \cdot I_{vec} + \Sigma_B^{-1}\mu_B\right) \tag{49}$$

and

$$\Sigma_B^* = \left(T_{kron}^T \ \Sigma_B^{-1} \ T_{kron} + \Sigma_B^{-1}\right)^{-1}. \tag{50}$$

This belief update of B is straightforward and efficient, as we update our belief coefficients, μ_B and Σ_B, using closed form calculations in constant time. In our testing, this update takes approximately 0.1 seconds; however, this may change for different sizes of B, in addition to differences in computational power.

3.2.2 Observation Scaling

For some implantation recipes, the total dose may change between implantations even when T is unchanged, as the source current may fluctuate from one implantation to the next, resulting in scaled dose profiles. In order to avoid overfitting to these expected run to run variations, we scale incoming observations to account for these current and dose differences. Prior to model updates, we scale the observed dose profile such that the total dose of the scaled profile is equal to the total dose of the profile predicted under our current belief:

$$I_{scaled} = I\frac{\sum(BT)}{\sum I}. \tag{51}$$

3.2.3 Decay

Because the beam shapes may vary unintentionally between implantations, we decay the certainty of our belief before each update. Without this decay, the precision of our belief, Σ_B^{-1}, will grow indefinitely, lessening the impact of new observations. This would prevent our model from adjusting to changing process conditions, and also lead to numerical issues. This decay is equivalent to the state-transition function in Kalman filters that describes the evolution of our belief between observations, and is also similar to the model decay discussed in Section 2.3.3.

The transition function should have two qualities. First, it should preserve the mean belief, μ_B, as we do not know how the system will drift, and should only increase the uncertainty of our belief. Secondly, we do not want the uncertainty to increase infinitely. As we will discuss in the next section, we have a prior belief of B that represents our belief with no observations. When many iterations are performed and no observations are

made, our uncertainty should approach that of the prior. With these two characteristics in mind, we choose the decay function to be an exponential decay of our belief covariance into the prior covariance, $\Sigma_{B,prior}$, with no change to the mean belief:

$$B_{t+1} \sim N\left(\mu_{B,t}, (1-\alpha)\Sigma_{B,t} + \alpha\Sigma_{B,prior}\right). \tag{52}$$

Here, α is a constant that determines how much change we expect between observations. Larger values of α more drastically increase the system uncertainty between implantations, while lower values suggest that the underlying system is relatively unchanged. In our testing, we chose a single value for α; however, more sophisticated methods that consider the time between implantation, or the likelihood of observations, may improve performance.

3.2.4 Prior

Tuning a process begins by choosing a prior belief of B. This choice is critical, as an accurate prior greatly reduces the number of iterations required to converge to a desired uniformity. Here, we present our parameterized prior that incorporates our existing belief in the form of B.

The prior belief has two components, the mean, μ_B, and the covariance matrix, Σ_B. Our choice of μ_B makes two approximations. While beliefs after future observations may not adhere to these approximations, they allow us to initialize our belief with a relatively accurate prior. The first approximation is that the beam does not change as it moves across the wafer, as perturbations of the beam shape are relatively minor, and these are learned during tuning. Additionally, as the direction of these perturbations is unknown, the most likely belief is an unchanged beam across the entire wafer. The second approximation is that the beam shape is Gaussian. While this is not perfectly accurate, the implantation rates are most intense at the center of the beam and asymptotically decay to zero as the distance from the center increases. Therefore, a Gaussian prior is a relatively accurate, parameterizable, approximation that can be refined with additional observations. Under this assumption, we parameterize the beam with intensity A_μ, beam width W, and beam center C as pictured in Fig. 22, and according to:

$$\mu_{B,i} = A_\mu \; e^{-\frac{1}{2}\left(\frac{(X_{I,vec})_i - (X_{T,vec})_i - C}{W}\right)^2}. \tag{53}$$

In this, and future, definitions, we use full position matrices, $X_T \in R^{|x_I|\cdot|x_T|}$ and $X_I \in R^{|x_I|\cdot|x_T|}$ that give the values of the time, x_T, and implant, x_I, positions respectively for any

pair of time and implant position indices, $i \in 1{:}\,|x_I|$ and $j \in 1{:}\,|x_T|$. These matrices are useful as they are the same size as B and allow us to index the two position vectors as well as B with the same set of indices:

$$X_{I,i,j} = x_{I,j} \tag{54}$$

$$X_{T,i,j} = x_{T,j}. \tag{55}$$

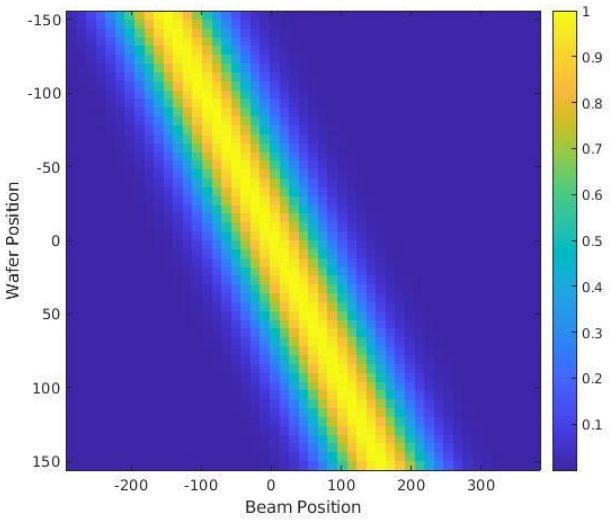


Fig. 22: Example prior mean, μ_B.

The second component, Σ_B, requires a more sophisticated design. Each term in the covariance matrix contains the uncertainties, as well as the expected relationship, of two points in B. Therefore, we decompose each covariance term, $\Sigma_{B,i,j}$, into the pointwise uncertainties of the points, $\sigma_{B,i}$ and $\sigma_{B,j}$, and the correlation between them, $\rho_{i,j}$. The pointwise uncertainties, $\sigma_{B,i}$ and $\sigma_{B,j}$, represent the uncertainty of each term in B, while the correlation coefficients, $\rho_{i,j}$, represent how we expect the terms of B to be related. Finally, we include a third pointwise noise term, σ_p, in the diagonal to ensure that the matrix is invertible:

$$\Sigma_{B,i,j} = \sigma_{B,i}\sigma_{B,j}\rho_{i,j} + \begin{cases} 0 & if\ i \neq j \\ \sigma_p & if\ i = j \end{cases}. \tag{56}$$

When defining σ_B, we make the same assumptions as when defining the mean belief. Areas with greater intensity, such as the center of the beam, have high uncertainty, while areas with little intensity, such as those far away from the beam center, have low uncertainty. Therefore, we again parameterize the pointwise uncertainties as a Gaussian with the same beam width W, and the same beam center C, but with a max uncertainty

A_σ. Additionally, we add a constant variation, σ_m, to ensure that there is a minimum uncertainty at each point in B:

$$\sigma_{B,i} = A_\sigma \, e^{-\frac{1}{2}\left(\frac{(X_{I,vec})_i - (X_{T,vec})_i - C}{W}\right)^2} + \sigma_m. \tag{57}$$

When constructing ρ, we make two assumptions to improve our prior belief. First, we assume that as the beam moves across the wafer, its shape remains relatively constant, and thus points in B with the same distance from the beam center are heavily correlated with a trans-wafer correlation coefficient, β. Secondly, we assume that minor perturbations within the beam shape are correlated to nearby beam locations, with a length scale, l_b, and that these perturbations are also correlated within the beam with a length scale, l_o. With these two assumptions in mind, we define our correlations coefficients as:

$$\rho_{i,j} = \left(\beta + (1-\beta) e^{-\frac{1}{2}\left(\frac{(X_{T,vec,i} + X_{I,vec,i}) - (X_{T,vec,j} + X_{I,vec,j})}{l_b}\right)^2}\right) \cdot e^{-\frac{1}{2}\left(\frac{(X_{T,vec,i} - X_{I,vec,i}) - (X_{T,vec,j} - X_{I,vec,j})}{l_o}\right)^2}. \tag{58}$$

In Fig. 23, we show the correlation coefficients for one point in B. Here, points with similar beam locations are highly correlated, and this correlation decreases as the distance between the beam locations increases. Additionally, points with similar distances from the beam center are also heavily correlated and this too decreases as the distance between the beam locations increases; however, these asymptotically approach a correlation coefficient of β, instead of 0, as the beam shape is mostly constant across the wafer.

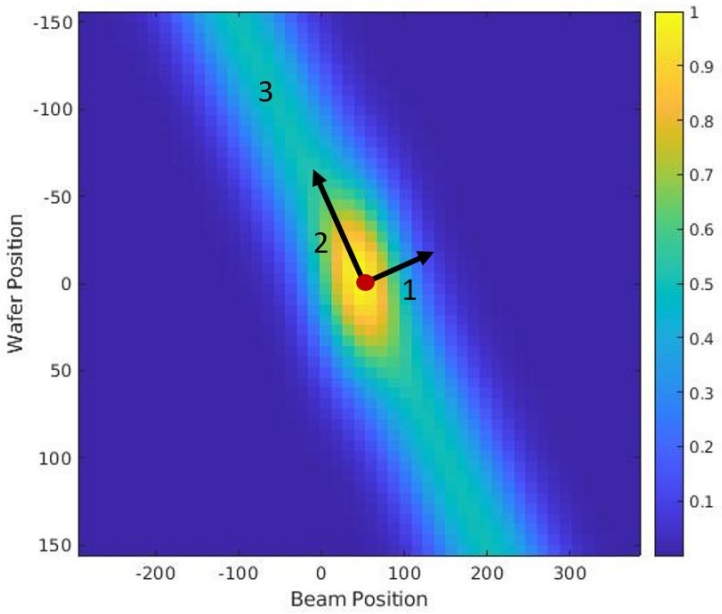


Fig. 23: Example correlation coefficient of marked position (red) and all other locations. Note the decay as the distance from the beam center, $X_{I,vec} - X_{T,vec}$, (1) as well as the beam location, $X_{I,vec} + X_{T,vec}$, (2) changes. Additionally, note the minimum trans-wafer correlation (3).

3.2.5 Fast Spots

In addition to wafer dose measurements, fast spots are a useful supplemental measurement. These are cheaper and quicker to perform, as they do not require a direct measurement of the wafer dose. Instead, a faraday, a sensor that measures ion current, is placed on the wafer at a specific location, $x_{I,S}$. The beam is then swept across the wafer with times T_f, and the dose at the faraday location, I_f, is measured as the beam moves across the wafer. This provides the implantation rate at one point on the wafer for every beam location; however, these measurements must be incorporated into our modeling framework in order to utilize them.

To do so, we express fast spots as a series of sequential observations, where each observation in the series represents one location of the beam during its sweep, and the corresponding faraday measurement. Under this description, fast spot measurements can be incorporated into our Bayesian modeling framework.

Since the ion current is continuously measured during the beam sweep, each measurement represents the implantation from only a single beam location. Therefore, the time vector for the i^{th} measurement, T_i, is zero except at the i^{th} beam location, where $T_{i,i} = T_{f,i}$.

Additionally, as we only measure the dose at the faraday location, each dose profile is zero for all non-sensor locations, and the dose at the faraday location for the i^{th} observation, $I_{i,s}$, is the i^{th} faraday measurement, i.e., $I_{i,s} = I_{f,i}$. Because the implantation dose is likely non-zero at all non-faraday locations, we set the implantation noise to ∞ at all non- faraday locations. This represents our lack of observation at non-faraday locations, and is implemented by making Σ_n from Eq. 47 and 49 a diagonal matrix with the sensor noise equal to the default sensor noise, and all others infinity, i.e., $\Sigma_{n,s,s} = \sigma_n$ and $\Sigma_{n,i,i} = \infty$ if $i \neq s$.

$$\Sigma_n = \begin{bmatrix} \infty & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \sigma_n^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \infty \end{bmatrix}. \tag{59}$$

3.3 Optimization

After creating a predictive model, we exploit our belief to optimize the implantation dose profile. Specifically, we use our belief of B to choose a set of times, T, that lead to our desired dose profile, I_d. However, we must also consider the constraints of our solution. These include minimum and maximum sweep speeds, translated to minimum, T_{min}, and maximum, T_{max}, times for each element of T, as well as a constraint on the total implantation time. Formally, we frame this as a constrained optimization problem, where we minimize the mean square error (MSE) between the predicted and desired profiles subject to our linear constraints.

While this can be sufficient, we also add a regularization term to our cost function that punishes accelerations in T by a factor of λ. This has many benefits: it prevents overfitting to our current belief, it leads to smooth time profiles that give more repeatable results, and finally, it ensures a well-conditioned optimization problem. Our final formulation is:

$$\begin{gathered} \min_T \sum (BT - I_d)^2 + \lambda \sum (T_{i-1} - 2T_i + T_{i+1})^2 \\ s.t.\ T_{min} \leq T_i \leq T_{max} \forall i \in 1{:}\,|x_T|, \\ \sum T = T_{total}\,. \end{gathered} \tag{60}$$

This is a convex quadratic optimization problem with linear constraints, and numerous tools exist to solve these problems.

In order to fully define this optimization problem, we must choose a value for the regularization parameter λ. As λ decreases, so does the penalty for variations in T,

allowing for a finer tuning of I. However, larger values of λ produce smoother T vectors that help prevent overfitting. Therefore, we should choose the value of λ in order to achieve the uniformity that we desire. Here, we select a value λ just small enough such that the predicted profile meets a desired uniformity requirement. Practically, we use a binary search on λ to find the value that meets this requirement. In our testing, the binary search problem and original optimization problem can be solved quickly. Typically, the combined problem takes 0.1 seconds or less to solve.

3.4 Illustrative Example

In this section, we walk through an example tune using synthetic data in order to demonstrate key aspects of the approach. Here, we synthesize data using a beam with a fixed beam width, with value $W = 75$ mm, but whose intensity oscillates as it is swept across the wafer. The synthetic beam matrix can be seen below in Fig. 24:

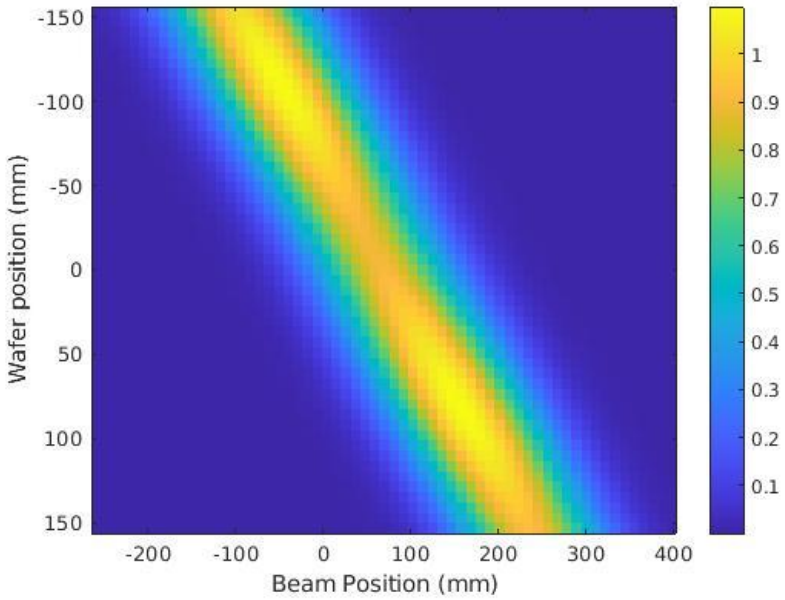


Fig. 24: True B matrix used to synthesize illustrative data.

When initializing our belief, we choose to use a beam width 50% wider than the true width, i.e., $W = 112.5$ mm, in order to demonstrate how it learns the true underlying system (Fig. 25).

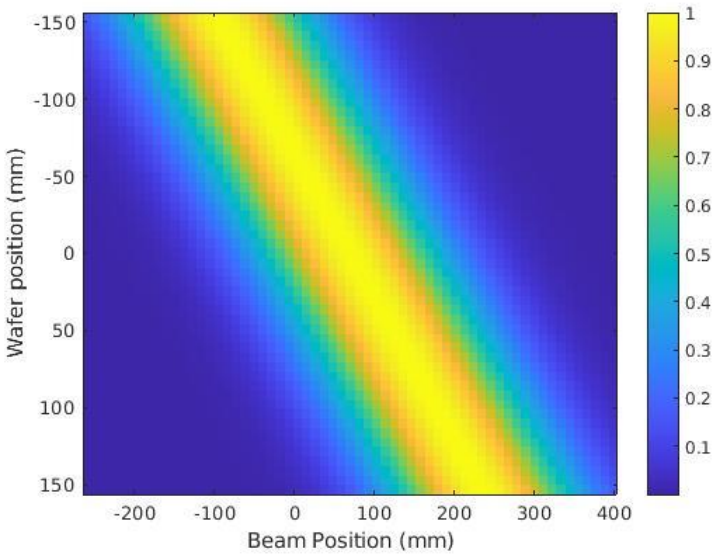


Fig. 25: Starting mean belief of the B matrix (μ_B).

Under the starting system belief, the first profile to be selected by the optimizer will have constant speeds, i.e., one with constant beam times, T, across the wafer. This is because it will minimize both the non-uniformity aspect of the cost function, since the predicted profile is flat, as well as the smoothness penalty, since constant times have no acceleration in the profile. Using our synthetic "true" system, and constant beam speeds, the resulting implantation dose is calculated (Fig. 26). The model is then updated using this profile, and the results can be seen (Fig. 27). The updated belief reflects the underlying change in intensity; however, its belief of the beam width is still wider than the true value

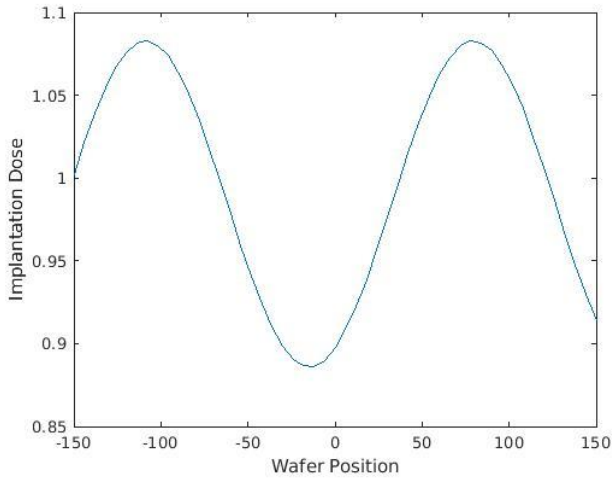


Fig. 26: Dose profile resulting from synthetic beam matrix and constant beam times.

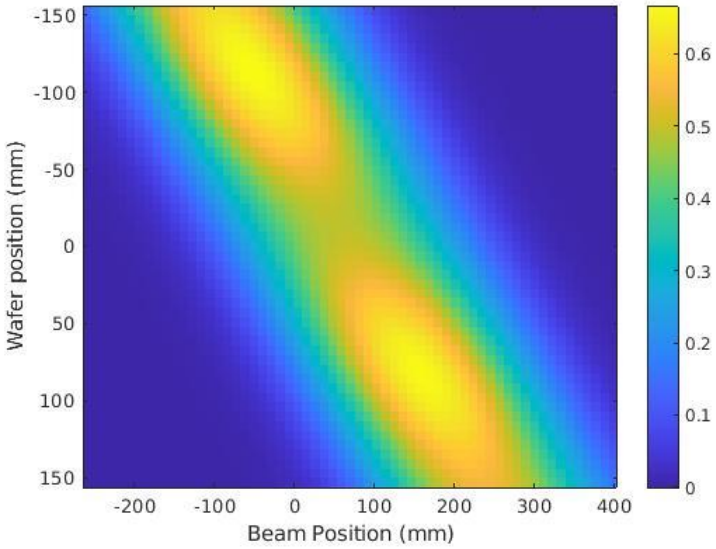


Fig. 27: Mean belief of B, μ_B, after observation of linear profile (in Fig. 26) using incorrect starting beam width.

Another valuable type of measurement, referred to as a dip profile, observes the effect of either slowing, or speeding up the beam at a specific location, in order to explicitly measure the shape of the beam at that location. The dip profile (Fig. 28) can be subtracted from the linear profile in order to determine the exact effect of the beam at that location. We do not need to explicitly perform this subtraction in our modeling approach, because sequentially updating the model with linear and dip observations automatically incorporates this knowledge into our belief (Fig. 29). Considering a dip profile provides a good example of how information is learned, and demonstrates key aspects of our model. Fig. 28 (left) shows an example beam times profile corresponding to our example dip profile, whose simulation is shown in Fig. 28 (right). Using these as a second set of observations, the model is again updated, resulting in the improved belief of B (Fig. 29).

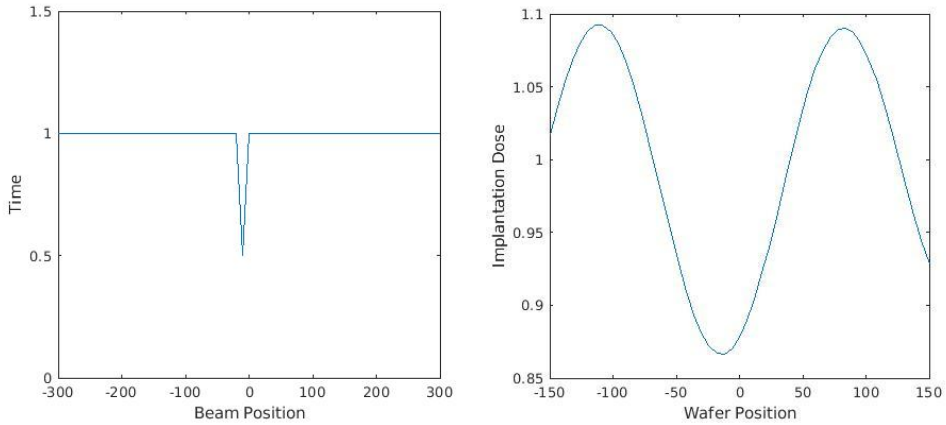


Fig. 28: Times profile (left) and resulting implantation dose (right) for a dip observation.

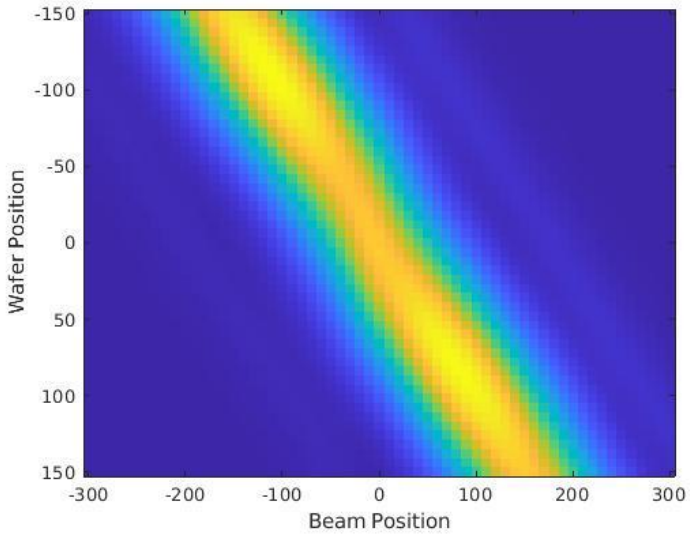


Fig. 29: Mean beam shape beliefs after linear and dip observations.

After observing the sequential linear and dip profiles, the resulting mean belief, μ_B, now provides a good approximation for the true underlying system, which can be seen when comparing Fig. 29 to Fig. 24. First, the linear profile has informed the model how the intensity of the beam changes across the wafer (Fig. 27), giving the belief the same oscillating pattern seen in the true system. Then, updating the model using the dip observation has improved the estimate of the beam width, as this was originally incorrectly set (Fig. 29). Even though we directly observed the beam and its width at one location, this information is critically transferred across the entire B matrix, giving the correct width at all locations. This is due to the trans-wafer correlation built into the prior, which assumes that the beam is relatively correlated as it is swept across the wafer. While

there is not perfect correlation, as we still see the intensity oscillations, our Bayesian prior and modeling approach do transfer critical knowledge about the width of the beam across the belief.

A key theme of this thesis is the incorporation of existing process knowledge into our approaches. Here, we see that our assumption of the beam prior allows us to accurately estimate the beam shapes with only two observations. Without this prior, it may take as many observations as there are beam locations (60 in this case) in order to approximate the true system.

In addition to looking at the posterior mean belief, we can also observe key changes to the pointwise uncertainties of our belief (the diagonal of Σ_B). These values express how certain we are of the current belief, on a pointwise basis. After updating the model using the sequential linear and dip profiles, we see a low variance at the location where the dip occurred (Fig. 30). As previously described, the combination of the dip and linear profile give us near perfect information of the beam shape at the dip location. Thus, our uncertainty of the beam at the dip location is lower than other locations, as we have exact measurements of these values.

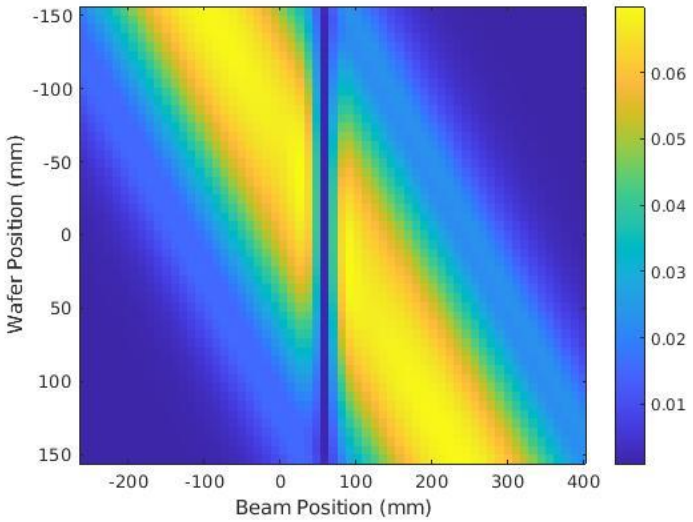


Fig. 30: Pointwise uncertainties after linear and dip observations.

After these initial model updates, we use our optimizer to select new times that are predicted to lead to our desired profile (i.e., a flat implantation dose). To perform this optimization, we require our desired profile, our mean belief of B, μ_B, as well as a regularization constant, λ. The value of λ determines how aggressively to tune the implantation profile. Larger values of λ impose larger penalties on variation in the times

profile, limiting how accurately a profile can be tuned, but also preventing overfitting. Below, we plot resulting times and dose profiles for three separate values of λ (Fig. 31). Here we see that lower values of λ allow for more variation and less smoothness in T. This allows for tighter tuning of the predicted implantation dose; however, in reality this may also lead to overfitting. Conversely, choosing too large a value of λ overly restricts the solution, and does not allow for T to vary enough to achieve uniformity. Finally, selecting a medium value of λ allows for both sufficient control over time implantation profile, while still retaining smooth values of T.

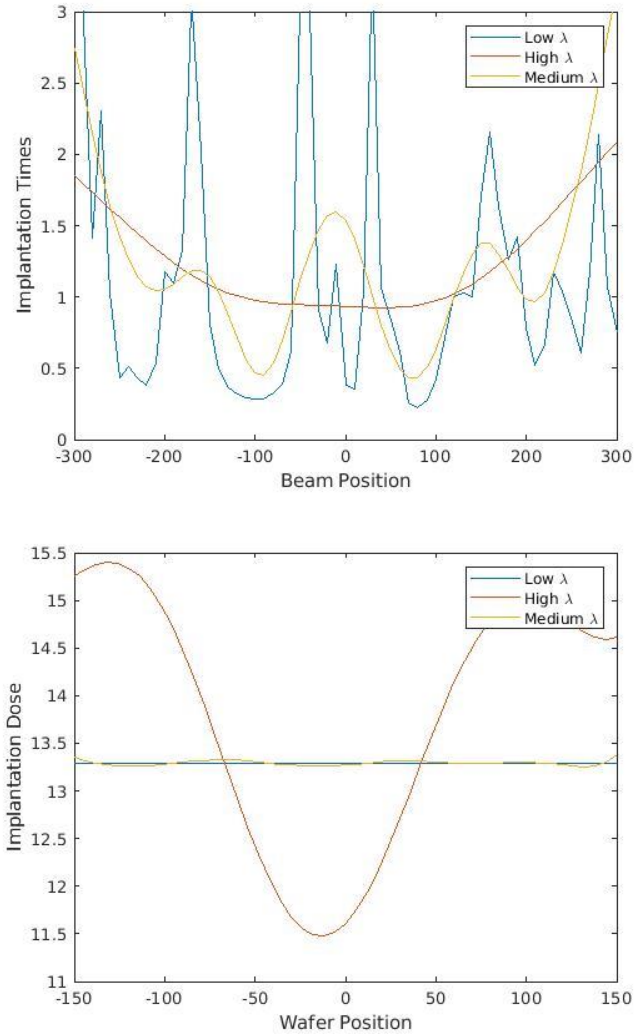


Fig. 31: Implantation time solutions (top) and predicted dose (bottom) for three values of λ.

In practice, we must determine which value of λ to use before selecting new implantation times. Here, we can choose a value of λ that is *predicted* to give us the uniformity we desire. We can solve the optimization problems for many values of λ, and

choose the value which we believe will lead to our desired non-uniformity. Below, we see the predicted non-uniformity as a function of the value of lambda and also mark the value of λ required in order to achieve a 0.5% non-uniformity (Fig. 32). When performing a tune with this desired uniformity, we use the marked value of λ when selecting new implantation times. Selecting significantly smaller values may lead to overfitting, while selecting larger values will not allow for sufficient compensation. As this is a monotonic function, we choose our value using a binary search in practice.

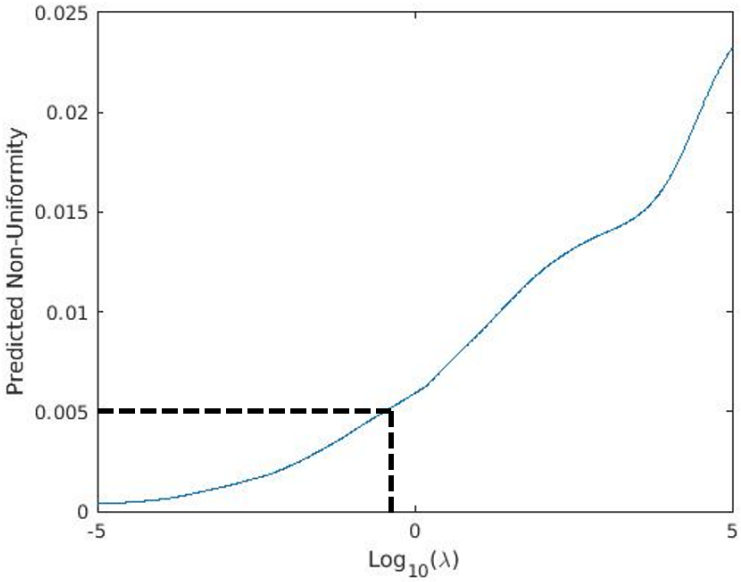


Fig. 32: Predicted non-uniformity as a function of λ. This is used to select a value of lambda before performing a new implantation. Value of λ needed to achieve 0.5% non-uniformity is marked.

Finally, using this value of λ (just below 10^0 in our case) we select a new set of times. A new implantation could then be run with these times for the next round of learning. For this illustrative example, the resulting implantation times are shown, as well as the simulated implantation dose resulting from these times (Fig. 33) The model could then be updated and the process could repeat until convergence if necessary. Here, we see large compensating times at regions with low doses in the linear profile, and lower time in high dose regions, resulting in a more uniform dose profile.

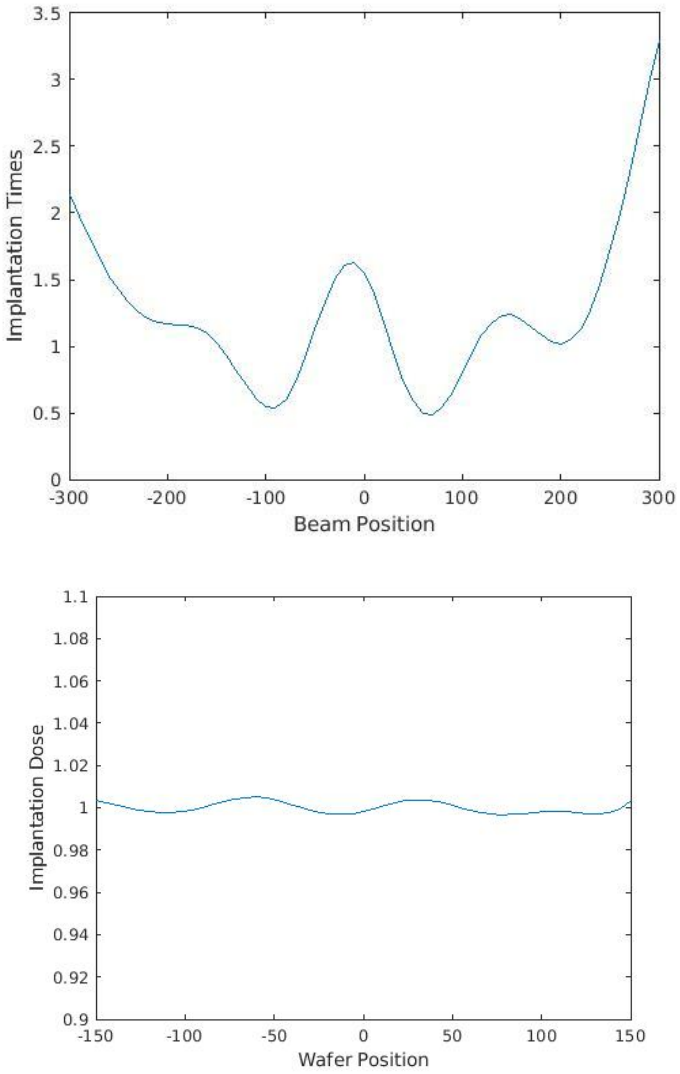


Fig. 33: Synthetic implantation times (top) and resulting dose profile (bottom) in first tune of illustrative example.

3.5 Results

We next present the results of our tuning algorithm and compare to an existing industry tuning method used by Applied Materials, Inc., when used in practice on a real implantation tool. Here, we evaluate performance using a low energy Argon implantation recipe. This particularly low energy recipe exceeds the limits for what is normally recommended, as the resulting beam shape varies drastically as it is swept across the wafer. Using this for our benchmarking recipe represents the worse-case tuning scenario, and also allows us to consider the feasibility of an expanded recipe set.

We perform 27 separate (repeated) tunes using each method. We begin each tune by initializing to our prior belief for the proposed method. Then, we alternate between tuning the process using our proposed method, and using the existing industry method, and record the results. At the beginning of each tune, a linear implantation, i.e., one using constant beam times, is performed and the results are used to update each model, a procedure required for the prior industry method. Additionally, we update the proposed method using cost and time efficient fast spots, taken at faraday locations of x_f = -50, 0, and 50 mm. The time devoted to these is negligible compared to a standard tuning implantation, as the sweep time is significantly lower, and the implantation is directly measured.

After 27 repeated tunings using each method, we determine the number of iterations required to achieve a desired non-uniformity (NU), defined as:

$$NU(I) = \frac{std(I)}{mean(I)}. \tag{61}$$

This is the most important performance metric, as our primary goal is to tune to a desired non-uniformity in as few iterations as possible. We choose a desired non-uniformity of 0.5%, as inherent random run to run variations limit further tuning.

In Fig. 34, we present the distribution of the number of iterations required to meet our desired non-uniformity. We see an enormous improvement, as on average, the proposed method tunes in 2.2 iterations, while the existing industry method tunes in 4.3 iterations. For poorly behaved beams that require frequent re-tuning, this represents a significant decrease in tool downtime, as well as in wasted material.

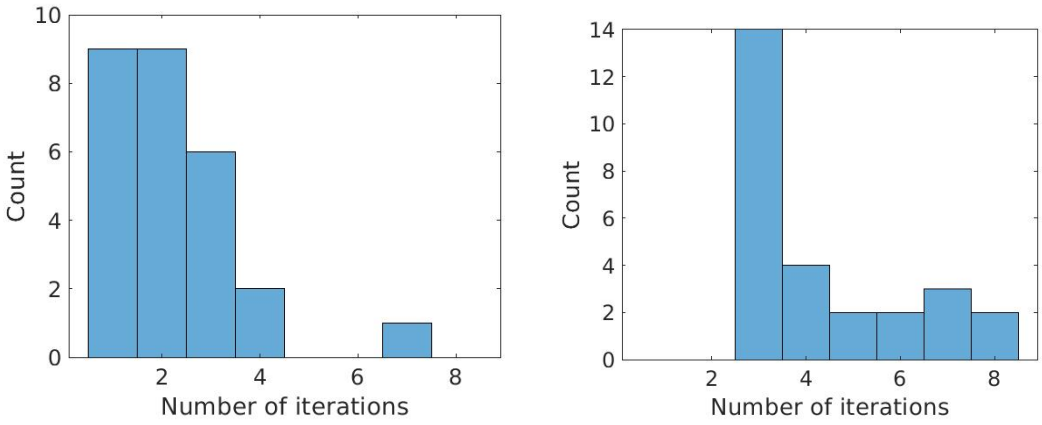


Fig. 34: Histogram of number of iterations required to achieve 0.5% uniformity in proposed (left) and existing industry (right) methods.

Additionally, we present the distribution of non-uniformities for the two methods as the tunes progress in Fig. 35. This shows how both methods improve uniformity before plateauing; however, the majority of the improvement for the proposed method is achieved with the first tune, signifying faster convergence, while still to having a superior steady-state non-uniformity, and lower variance in the results.

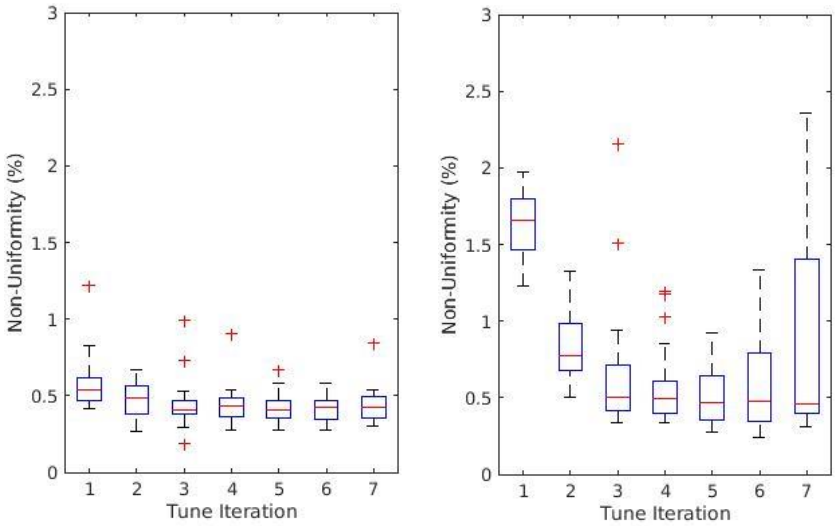


Fig. 35: Distributions of non-uniformity vs. number of tunes for proposed (left) and existing industry method (right).

Additionally, we plot an example tune for the proposed method in Fig. 36, showing the series of implantation times as well as resulting profiles for the three iterations required to converge. Here, we see that while the first iteration comes close to a desired solution, it is not quite sufficient. On the second tune, the solution overcompensates for the remaining variation, and finally this is corrected on the third iteration.

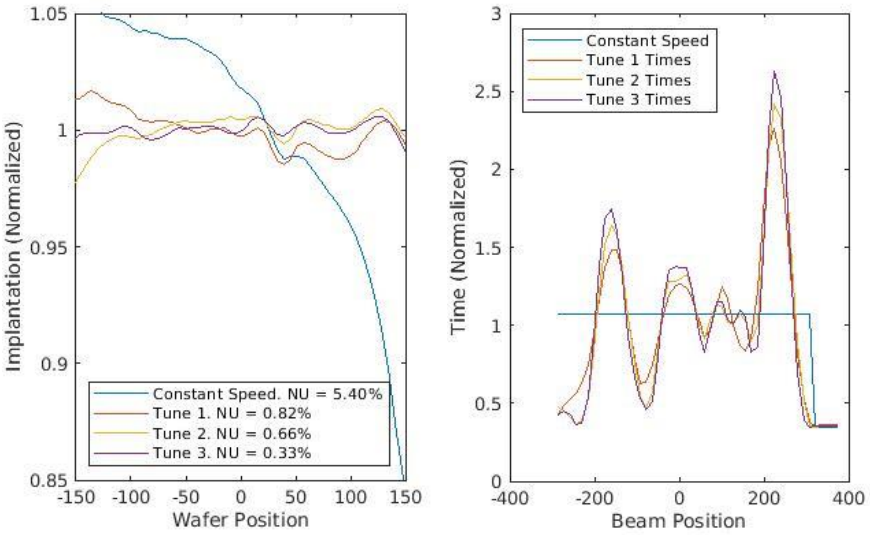


Fig. 36: Implantation dose (left) and corresponding times (right) for an example tuning run.

Additionally, we plot the resulting maximum likelihood belief for B, μ_B, after three iterations, and at the end of the tuning session in Fig. 37 and Fig. 38, respectively. It is important to note that the maximum likelihood belief, μ_B, is physically plausible after a small number of learns, as is seen in Fig. 37. Because the system has many inputs (the times at each beam location), one would expect a linear model to over-fit to this under-defined system after only three model updates. However, the presence of the well-chosen prior helps prevent this, as we have already imparted our general belief of the beam shapes onto the model. This highlights the advantages of a prior chosen using domain specific knowledge.

Additionally, the belief at the end of the example tuning session (Fig. 38) suggests that our approach to overcome concept drift is successful. Because the model tunes are still accurate even after many repeated model updates, the model is able to successfully adapt to a changing system. Additionally, because the structure of μ_B remains physically plausible after multiple model updates, this also suggests that we retain key elements of our prior belief, which is crucial to remaining accurate.

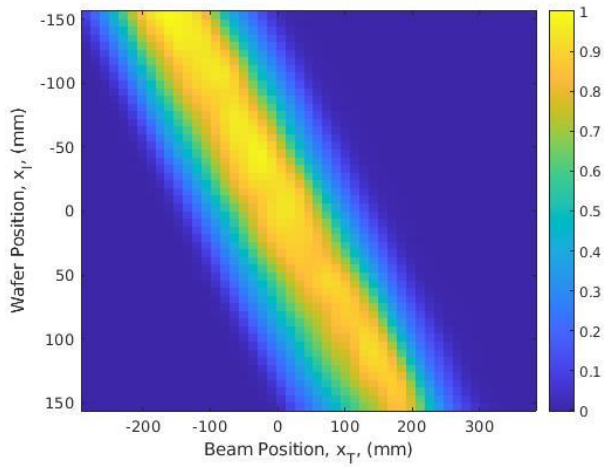


Fig. 37: Example normalized μ_B after three tunes.

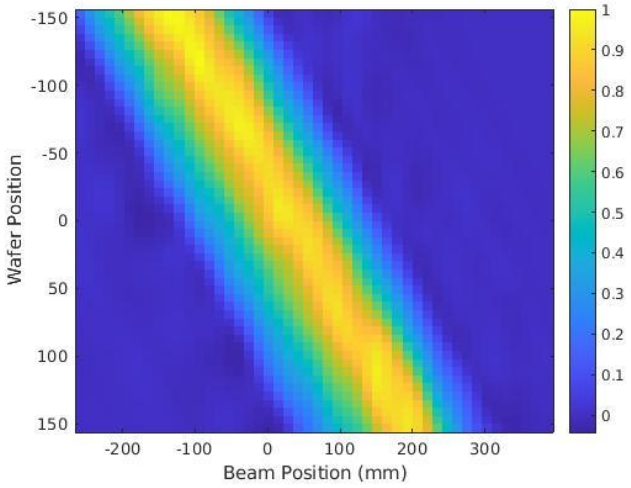


Fig. 38: Normalized μ_B at the end of tuning session.

In addition to the reduced number of required tuning iterations, the proposed approach also achieves a 100% tuning success rate, while the existing industry approach failed to converge in two out of 27 cases. When operating outside of the recommended limits, such as in our testing, the existing approach may fail to converge to a desired non-uniformity for particularly high variance recipes. When this occurs, the tool must be reset then retuned, which requires a large tool downtime. The ability to reliably converge in a wider range of recipes is therefore a significant additional advantage of the proposed method over the existing tuning approach.

Finally, solutions found by our proposed method also provide greater total implantation dose using the same total implantation time and uniformity requirements. In Fig. 39, we present distributions of the mean implantation dose for the proposed method, as well as those of the traditional approach. On average, the solution from the proposed method provides a significantly greater dose compared to the traditional method. This is highly valuable, as the solution from our proposed method can be scaled to give a similar dose as the existing method but in less process run time. This represents a significant increase in tool throughput.

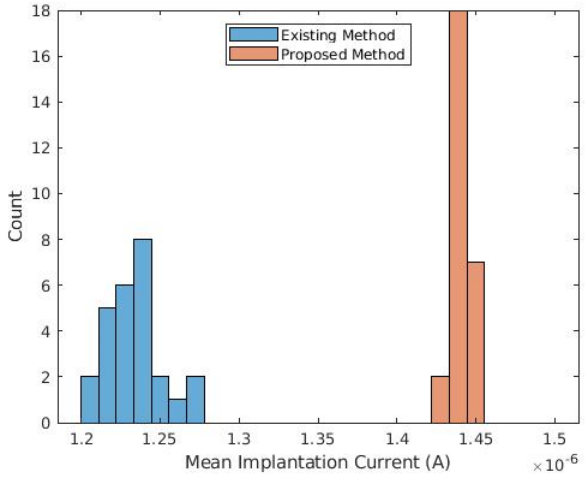


Fig. 39: Histogram of mean implantation rate for proposed and existing industry solutions.

The reason for this increased dose is that solutions from the proposed approach spend more time on wafer, and less time off wafer. This can be seen in Fig. 40 showing example solutions from both methods. This difference is likely due to a difference in the two optimizers. The optimizer in our proposed approach reduces the MSE between the predicted and desired profiles. As the desired profile is held constant, this prevents the total dose from decreasing over the course of a tune. In contrast, the existing industry approach iteratively reduces the time spent at high implantation areas in order to flatten the implantation profile, resulting in a lower final implantation dose.

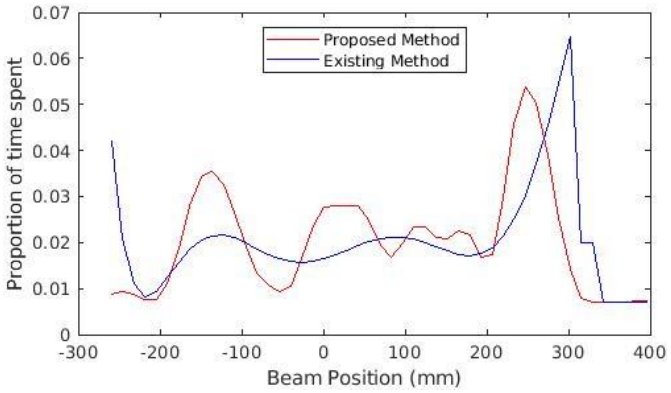


Fig. 40: Example implant times given by proposed and existing industry methods. Note the difference in time spent far outside wafer ($\pm$ 250mm).

3.6 Conclusions and Future Work

In this chapter, we present a machine learning method for modeling and optimizing dose uniformity in ion implantation processes. We first propose a linear relationship between implantation times and the resulting implantation dose profile. We use this relationship to develop the two halves of our proposed tuning approach. The first half is a Bayesian forward model that we iteratively update using new implantation measurements. The second half, the optimizer, uses the forward model to select new implantation times that meet a desired non-uniformity. This approach is enhanced with model decay, a physically motivated prior, and fast spot measurements. Using this proposed approach, we tune a worst-case implantation recipe in nearly half as many iterations on average compared to an existing industry approach. Additionally, our proposed method has a 100% convergence rate in testing, and achieves a significantly higher total dose, increasing the tool total throughput by a similar amount.

This case study highlights both key themes of the thesis. First, the approach effectively incorporates prior process knowledge in multiple ways. The underlying linear relationship between the implantation times and resulting implantation dose provides a necessary assumption that greatly simplifies the problem. Additionally, the selection of a physically motivated prior allows effective model updates with little training data. An example of this is demonstrated in Section 3.4, where a single dip measurement is able to correct for an incorrectly chosen starting beam width. The second key theme seen here is the advantage of Bayesian methods. As discussed in Section 3.5, the incorporation of a well-chosen prior prevents overfitting when only small numbers of examples are

available, because we have an accurate and physically plausible belief after only three model updates. The Bayesian framework also allows us to accurately adapt to a drifting system. Our tunes remain accurate, even after many repetitions in the presence of concept drift, confirming that we stay up to date with the changing system.

While we have demonstrated excellent results with the proposed tuning methodology, there are still additional improvements to explore in future work. In one direction, the limits of the method can be pushed to determine if further uniformity and total dose improvements can be made. We believe there is a fundamental tradeoff between these two metrics, and that greater uniformity can be achieved at the cost of lower dose and vice-versa; however, we must still experimentally confirm this. Additionally, this method can be tested on additional implantation recipes. While the chosen recipe represents a worst-case scenario, we expect to see performance gains in other recipes as well.

Additionally, there are still improvements that can be made to the proposed approach itself. We believe using a probabilistic cost function during optimization may perform better than the current deterministic cost function. The predictive model provides a likelihood distribution for B, and an optimization cost function that maximizes a likelihood under this distribution may perform better than the current deterministic approach which uses only the mean value of B. Finally, we also believe a dynamic choice of the model decay constant, α, may improve the performance of our forward model. Larger decay constants are appropriate when the system is rapidly changing, and lower decay constants help retain older knowledge when the system is stable. We believe it is possible to dynamically chose this coefficient based on the accuracy of the current model. When predictions are highly inaccurate, the decay coefficient can be increased and when the model remains accurate, the decay constant can be lowered.

4 One-Class Process Anomaly Detection Using Kernel Density Estimation Methods

Faults in semiconductor fabrication processes are extraordinarily costly, in particular when left undetected and unaddressed for extended periods of time. Without rapid real time or run to run detection methods, process faults may remain undetected until the final device electrical test (E-Test). This is often weeks after the initial fault occurred, potentially degrading or scrapping all wafers processed on the tool during this period. The ability to detect faults and anomalies when they arise is thus highly desirable, as they can quickly be investigated and addressed to prevent additional unnecessary losses; thus anomaly and fault detection has been a topic of great interest in the semiconductor community [127]–[133].

Anomaly detection is typically enabled by process sensors placed within a tool that monitor and record process information, such as the tool temperature, pressure, and flow rates. When a fault occurs, this information often differs significantly from past data recorded during normal operating conditions. An appropriate model can detect this difference and classify the incoming data as anomalous.

While many modeling options exist for classification tasks such as these, two aspects of semiconductor fabrication restrict these options. First, faults are often unique and infrequent, thus it is unlikely that there will be sufficient faulty data to train a traditional supervised classification algorithm. For many processes and recipes, particular faults may never have been seen at all. Therefore, we are interested in an anomaly detection or classification algorithm that is one-class, i.e., it must be trained using only examples of non-faulty (nominal) tool data, ruling out the use of supervised classification algorithms.

Second, in a typical fab, there exists a wide variety of processes, tools, and recipes, each with their own unique set of nominal sensor signals. What may seem nominal for one tool or recipe, may instead be anomalous for another. In order to be effective, an anomaly detection method must be applicable to a wide variety of settings. Additionally, as some recipes are used infrequently, even nominal training data may be limited. For this reason, we are interested in an anomaly detection algorithm that is able to be trained on a relatively small amount of data, and ideally that can employ transfer learning where data from one process, tool, or recipe is used to train detectors for others.

With these constraints in mind, we explore kernel density estimation (KDE) as the basis for an anomaly detection methodology. KDEs use example data to estimate underlying probability distributions. In the past, these methods have been used in a wide variety of contexts [134]–[137] including in outlier detection tasks [138]–[142]. In the context of anomaly detection for semiconductor fabrication, KDEs estimate the probability distribution of sensor data under nominal processing conditions. This distribution is then used to evaluate the likelihood that new incoming data is also nominal, and alerts are raised when the incoming data is unlikely to be nominal.

In Section 4.1, we overview the data used to develop and evaluate our proposed approach. In Section 4.2, we present our proposed anomaly detection methodology. We first review KDEs, then discuss the structure of the process data, and finally formalize our proposed approach. In Section 4.3, we review benchmark comparison approaches for one-class anomaly detection, including those utilizing statistical process control (SPC) methods, one-class support vector machines (OC-SVMs), and variational auto-encoders (VAEs). We present the results of our proposed method in Section 4.4, and compare to results using these benchmark methods. Finally, in Section 4.5, we conclude by summarizing key findings and proposing future extensions to the present work.

4.1 Datasets

Here we overview the data used to evaluate our proposed method. Historical manufacturing run data at Analog Devices Inc. is used that includes both nominal process data, and data recorded during process faults that were undetected for an extended period of time. The data from faulty runs is not used to train the anomaly detectors, but crucially, the faulty run data provides us the opportunity to test or evaluate anomaly detectors built using only the nominal data. This data spans a number of plasma etch and ion implant process recipes.

Data from the plasma etch processes contain 31 different sensor signals, while data from the ion implantation processes contain only two. In the plasma etch datasets, anomalies are seen relatively easily in one of the many sensors (Fig. 41), while in the ion implantation case, anomalies are often subtler (Fig. 42). The plasma etch datasets challenge the ability of an anomaly detector to monitor many signals, while the ion implantation dataset challenges the ability to detect subtle anomalies with more limited sensor data.

The plasma etch dataset contains data from two process recipes. One recipe contains 1764 nominal runs and 479 faulty runs, while the second has 1079 nominal runs and 557 faulty runs. The ion implantation dataset contains data from 12 process recipes; however, only one of these datasets contains faulty run data. This faulty ion implantation recipe contains 11,374 nominal runs and 50 faulty runs. It is important to note that only the known good data will be used to build and train our anomaly detectors; however, both nominal and faulty data will be used to evaluate detector performance.

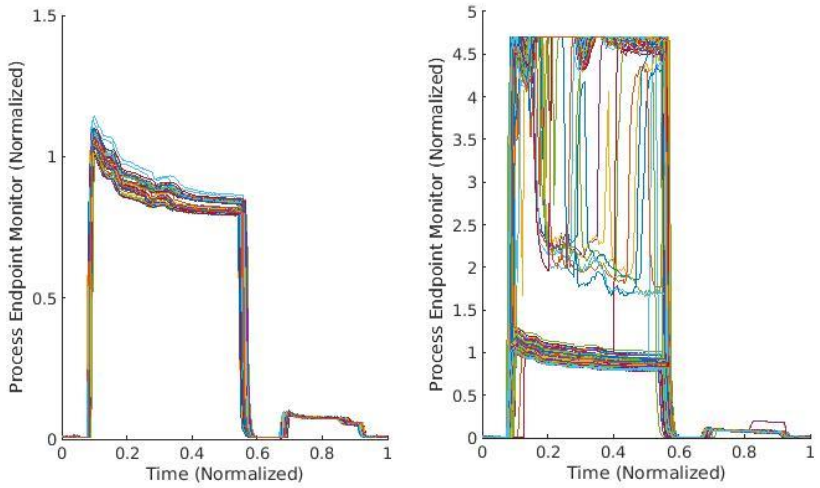


Fig. 41: Example of nominal (left) and fault (right) data for a critical sensor in a plasma etch case.

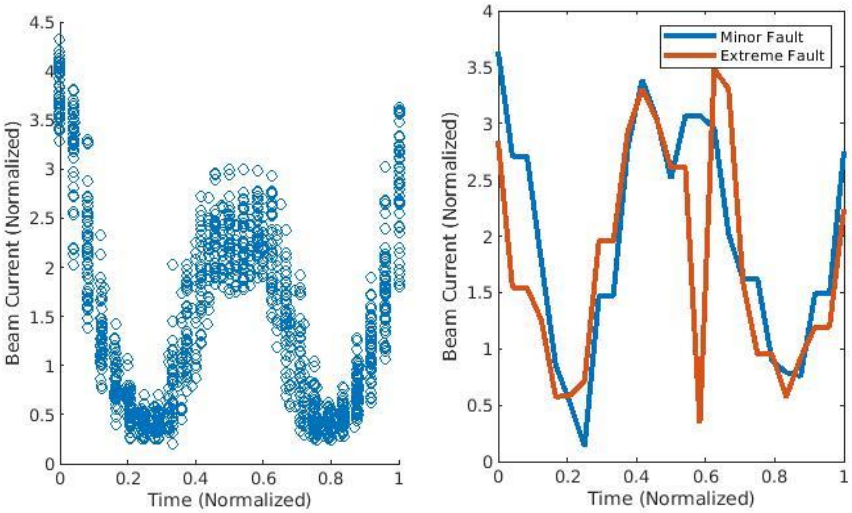


Fig. 42: Nominal beam currents vs. cycle times (left). An example minor fault (right) at $t = 0.25$ (blue) and extreme fault at $t = 0.6$ (red).

4.2 Anomaly Detection Methodology

In this section, we discuss the proposed anomaly detection methodology based on kernel density estimation (KDE). First, we give an overview of KDE methods. Then, we discuss how we structure the fabrication data for use in anomaly detection. Next, we discuss our specific approach for detecting anomalies in fabrication processes using KDE, and for choosing model hyper-parameters. Finally, we present extensions to the method for adjusting to concept drift and to employ transfer learning.

4.2.1 Kernel Density Estimation

Kernel density estimation is a non-parametric method that estimates the underlying probability density for a random variable using a set of sample data [143], [144]. These methods employ a kernel function, $K(x, x_{h,i})$, that creates a local probability distribution for an incoming data point, x, based on similarity or distance to a historical data point, $x_{h,i}$. The contributions from a set of historical examples, x_h, are summed to estimate a continuous probability distribution, $\hat{f}(x)$, for any new data point x:

$$\hat{f}(x) = \frac{1}{|x_h|} \sum_{i=1}^{|x_h|} K(x, x_{h,i}). \tag{62}$$

Alternatively, KDEs can be viewed as a convolution between the historical data and the kernel function. Conceptually, the discrete historical points are filtered by the kernel function, producing a continuous estimate of their underlying distribution.

The choice of the kernel function determines how historical examples are weighted when estimating the probability density of new points. The most common choice of kernel function is the Gaussian kernel:

$$K(x_1, x_2) = e^{-\frac{1}{2}\Sigma\frac{(x_1 - x_2)^2}{l^2}}. \tag{63}$$

While the kernel itself is Gaussian, the resulting distribution rarely is. KDEs produce a smoothed estimate of the historical data, so non-Gaussian data leads to a non-Gaussian estimate. The ability to estimate complex distributions is a notable advantage over traditional methods that assume an underlying Gaussian distribution, as we will see in Section 4.5.

In most kernels, a length scale parameter, or bandwidth (l), determines the smoothness of the resulting density function. This parameter can be more critical than the functional form of the kernel itself [145]. Too small a value will result in a rough

density function, heavily influenced by the specific historical examples, and too large a value will smooth out any true features of the underlying distribution [146], [147].

4.2.2 Data Structure

In order to develop and apply our anomaly detection methodology, we specify the structure of the process data. Because fabrication processes are typically separated into runs, our task is to assign a single label, either nominal (0), or anomalous (1), to an entire run using the sensor data, x, for that run. The sensor data is the key data that we analyze, and in this work is assumed to be multi-sensor, time series data, that contains one measurement, $x_{s,t}$, for each sensor s at each sampled point in time t. For each run, we normalize t such that it ranges from 0 to 1. This allows us to compare data between runs, even if there are small variations in the total runtime or number of samples taken. Extensions for non-constant scaling in time are possible, e.g., using dynamic time-warping [148], but are not considered here.

4.2.3 Application to Anomaly Detection

Here, we present our methodology for applying KDE to the original anomaly detection problem. The proposed approach creates univariate distributions for each sensor and for each point in time, then combines these before applying a low pass filter in order to reduce the effects of noise. Finally, a single aggregate likelihood is calculated for an incoming run, and this is compared to a threshold to assign a binary label.

When evaluating new incoming data, x, we first evaluate the probability of the individual sensor signals at each point in time. For each sensor, s, and point in time, t, we use KDE and relevant historical data, $x_{h,s,t}$, to estimate the nominal probability distribution:

$$f_s(x_{s,t}) = \frac{1}{|x_{h,s,t}|} \sum_{i=1}^{|x_{h,s,t}|} K(x_{s,t}, x_{h,s,t,i}). \tag{64}$$

Here, we use the standard Gaussian kernel, with different bandwidths l_s for each sensor:

$$K_s(x_1, x_2) = e^{-\frac{1}{2}\sum\frac{(x_1 - x_2)^2}{l_s^2}}. \tag{65}$$

In Fig. 43, we show example kernel density estimated probability distributions for a critical plasma etch and ion implantation sensor; these can be compared to Fig. 41 and Fig. 42.

It is important to note that when creating these probability distributions, we only use historical data, $x_{h,s,t}$, that occurs at similar points in the run time as the evaluated point, $x_{s,t}$. Specifically, we only consider the set of points whose normalized run times fall within a window of half-width Δt of our evaluated point:

$$x_{h,s,t} = \{x_{h,s,t'} \quad \forall t' \quad s.t. \quad |t - t'| \leq \Delta t\} \tag{66}$$

This hyper-parameter Δt is one of two parameters that must be selected prior to model training. Practically, we have found that a choice of Δt corresponding to the sampling rate performs well when there is a relatively constant sampling rate, as is typical in most data acquisition systems. However, a larger window size may be appropriate for slowly varying signals with substantial amounts of noise.

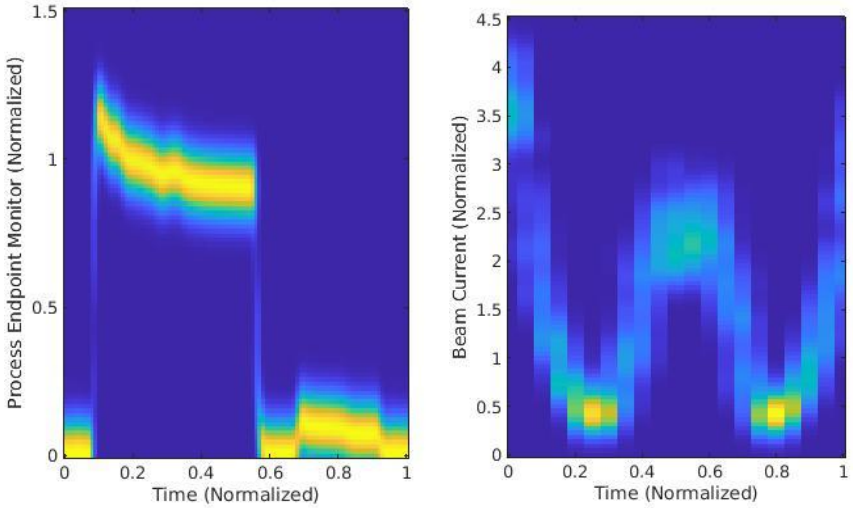


Fig. 43: Resulting probability distribution using kernel density estimation for critical plasma etch (left) and ion implantation sensors (right). Yellow corresponds to high probability, and blue to low probability.

Next, we combine the individual sensor probabilities to create a combined probability for each point in time, $f_t(x_t)$. Here, we take the product of the sensor probabilities to create the aggregate probability:

$$f_t(x_t) = \prod_{s=1}^{|s|} f_s(x_{s,t}) \tag{67}$$

These probabilities are then normalized as a function of time:

$$f_n(x_t) = \frac{f_t(x_t)}{n(t)} \tag{68}$$

Here $n(t)$ is a normalization coefficient that ensures that all times in a process cycle are equally likely to trigger a false alarm under nominal operating conditions, and is further discussed in Section 4.3.5. Without this normalization, times with higher inherent sensor variance are more likely to trigger false alarms compared to lower variance times. With this normalization, alerts are equally likely to be raised when minor variations occur at times with low sensor variance, as when larger variations occur at more inherently varied points in a run.

We then apply a Gaussian low pass filter, implemented as a convolution (denoted by $*$) with a filter, $g(x)$, with bandwidth Δt, in order to reduce false alarms due to noisy sensors:

$$f_f(x) = f_n(x) * g(x) \tag{69}$$

where

$$g(x) = e^{-\frac{1}{2}\left(\frac{x}{\Delta t}\right)^2}. \tag{70}$$

While this filtering step is not required, it often reduces the number of false alarms, and increases the separability of anomalous and nominal testing data. Finally, we assign the likelihood of the entire run being nominal, $f_r(x)$, as the minimum likelihood within that run:

$$f_r(x) = min\left(f_f(x)\right), \tag{71}$$

and decide the run is anomalous if the value is larger than a classification threshold, T.

The proposed approach is notable in specifically electing to *not* estimate full multivariate correlated distributions. We believe that the estimation of multiple univariate non-Gaussian distributions is better suited for use in anomaly detection with the time series multi-sensor data available in typical semiconductor processes. Specifically, efficient estimation of multiple univariate non-Gaussian distributions becomes possible with very limited data using multiple univariate KDEs. A complete run has $|s| \cdot |t|$ features by default; by evaluating the probabilities of each point in time separately, we reduce the number of kernel features to $|s|$, enabling density estimation with small amounts of run data. In contexts where detection of subtle anomalies in correlated data is important and larger amounts of run data is available, extension to multivariate KDE is possible. While evaluation of such contexts is beyond the scope of the present work, the next section summarizes how such extension can be carried out.

4.2.4 Multivariate KDE

In cases where either a small number of sensors are present, or where large amounts of sample training data is available, estimating the full multivariate distribution may be useful. Instead of combining univariate distributions for each sensor, as we do in Eq. 64 and Eq. 67, one can directly estimate the multivariate sensor distribution using a multivariate kernel with covariance matrix Σ_l:

$$K(x_1, x_2) = e^{-\frac{1}{2}(x_1 - x_2)^T \Sigma_l (x_1 - x_2)}. \tag{72}$$

The joint distribution can be estimated with fitting of Σ_l, and a KDE based on the estimated kernel is capable of detecting correlated errors. Synthetic data as shown in Fig. 44 illustrates a case where a full multivariate correlated distribution is required. Here, the values of the testing point (red) falls within the nominal univariate ranges for sensor 1 and sensor 2; however, the simultaneous values of both as represented by the red point is unlikely. Here, our proposed univariate method would miss this anomaly, while the multivariate approach would detect it.

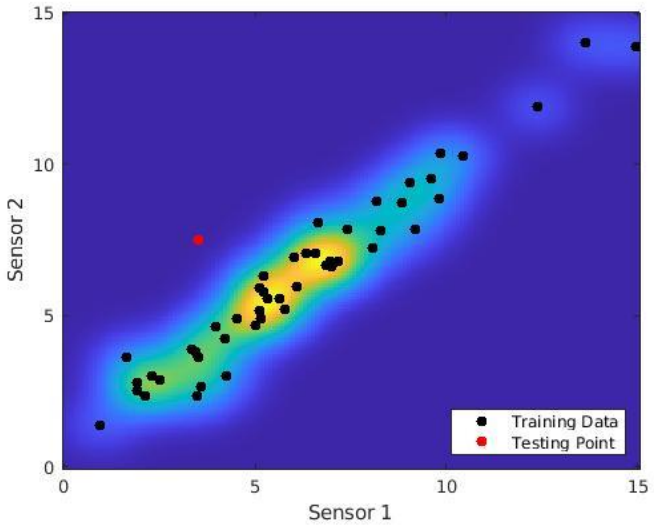


Fig. 44: Synthetic training data (black) and resulting multivariate KDE estimate showing a case where the combined multiple univariate KDE approach would not detect the faulty testing point (red).

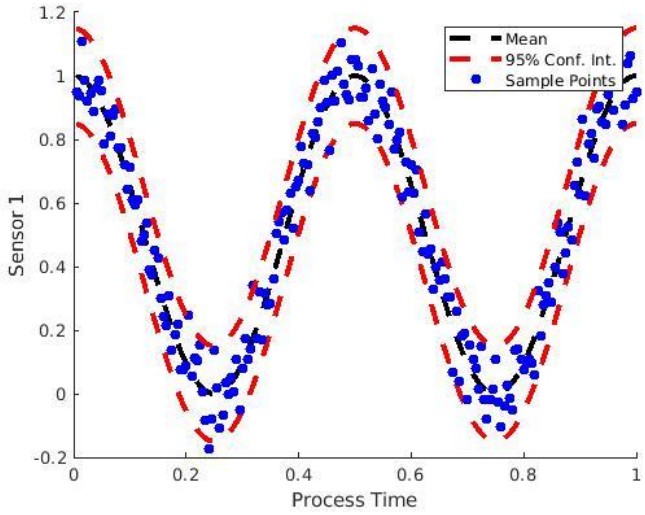
While the multivariate approach is capable of detecting a wider range of anomalies, the drawback of requiring exponentially more training data in order to estimate Σ_l is often too limiting. In testing with our process run data, we did not find any anomalies that were only detectable with joint distributions. Because these correlated-only errors are relatively uncommon, we believe that the less data intensive multiple univariate version

is more practically useful in limited data contexts, and we only consider multiple univariate detectors in the rest of this section.

4.2.5 Illustrative Example

In this section, we present a simple example that illustrates our fault detection methodology. We create a synthetic distribution for nominal process sensors, synthesize data using this distribution, and estimate this distribution using KDE. We then explore the effects of changing key hyper-parameters including the time window half width, Δt, and kernel bandwidth, l_s. Finally, we generate new testing data from the nominal distribution, synthesize example process faults, and apply our fault detection methodology to these two cases.

Our synthetic process includes two sensors. Both of these sensors have mean values that are a function of the process time, and at each point in time the underlying distributions are modelled as Gaussian distributions with standard deviations of 0.075 for both sensors. Below, we show the distributions for each sensor, as well as example points drawn from these distributions (Fig. 45).



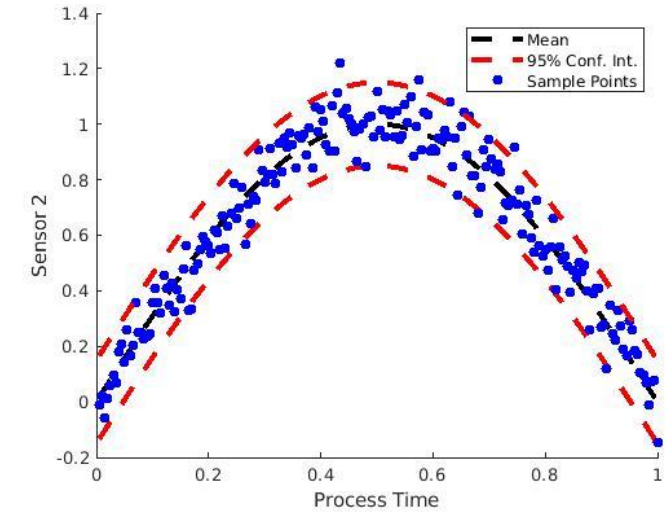


Fig. 45: Example distributions and example points as a function of process time for sensor 1 (top) and sensor 2 (bottom).

For each of these sensors, we use the example historical datapoints and kernel density estimation to approximate the underlying distributions for each point in time. Below, we plot the estimated distributions for sensor 1 (Fig. 46). We see that the estimated distribution gives a reasonable approximation of the underlying distribution as a function of time. At each point in time, sensor values with many historical examples are given high probabilities, while those with little or no nearby historical examples are given low probabilities.

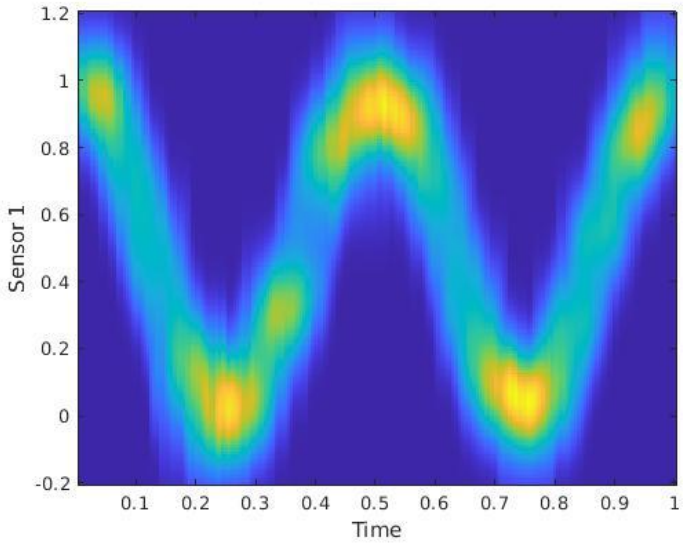


Fig. 46: Estimated probability distributions for sensor 1 as a function of time before time dependent normalization.

Importantly, sensor values with low signal derivatives have higher probabilities than those with high signal derivatives. This is because the underlying distributions are created using a time window of non-zero width, and areas with rapid changes in the signal mean include a wider range of historical examples. Areas with little change in the mean signal over time include a narrow range of historical values, as adjacent timepoints have similar sensor values. Conversely, areas with rapid changes to the signal, such as those between the corners, include the values of adjacent timepoints, whose means are different from the true mean value. This vertically stretches the estimated distribution, and assigns lower probabilities to the signal mean when the signal derivative is high.

This is particularly problematic when there are abrupt transitions within a process cycle, as these transitions have extreme rates of change in the sensor signals. This greatly stretches the probability distribution, thus making these points in time more likely to trigger false alarms. To compensate for this, we compute and apply normalization coefficients that are chosen such that the probability of raising a false alarm is equal for each point in the process cycle. The process of choosing these normalization coefficients will be described in the next section. Below, we plot the normalized probability distribution for sensor 1 when these normalization coefficients are applied (Fig. 47). After these are applied, the likelihood for the mean value of the signal is relatively constant, regardless of the time, ensuring that each point in time is equally likely to trigger a false alarm.

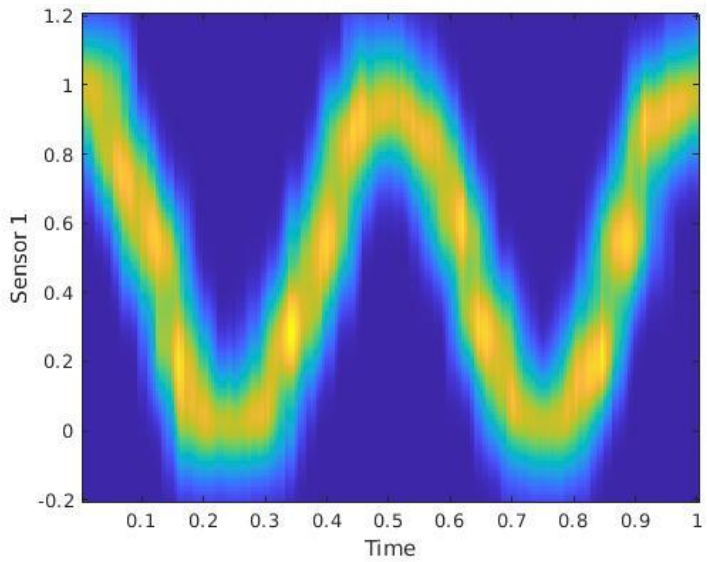


Fig. 47: Probability distribution for sensor 1 as a function of time with applied normalization coefficients.

In addition to these normalization coefficients, two other hyper-parameters are critical to the method. These include the historical window width, Δt, and the sensor length scales, l_s. The first of these determines which points are included when computing the underlying distributions. When determining the probability for incoming datapoints, we utilize all historical examples with time values within Δt from the new incoming point in our KDE estimate. Narrow time windows (low Δt) may result in more accurate estimates, as only samples with very similar process times are considered, but are also more prone to overfitting, and may lead to artifacts in time, as fewer historical examples can be used. Conversely, larger values of Δt result in a wider range of adjacent times, thus preventing overfitting but may also smooth important temporal features. Below, we plot distributions when Δt is chosen either too small or too large (Fig. 48), and these can be compared to Fig. 47 where an appropriate Δt is used.

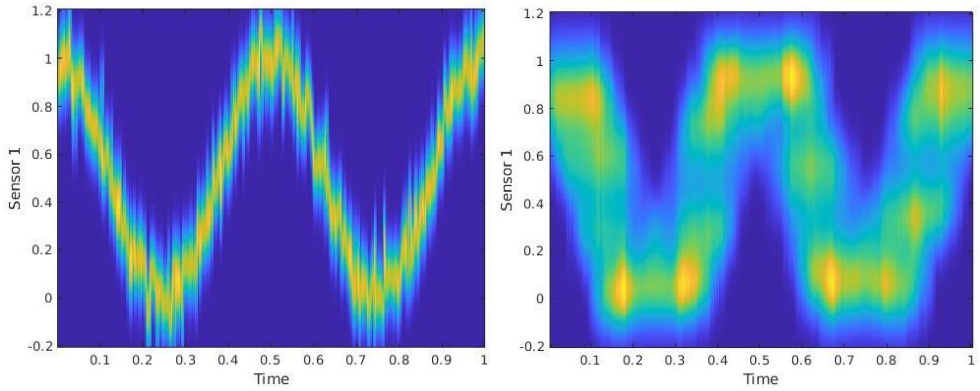


Fig. 48: Resulting sensor probabilities when the time window, Δt, is chosen too small (left), or too large (right).

The second hyper-parameter, l_s, is the kernel bandwidth used when determining the distributions at each point in time, and specifies how much smoothing to apply when computing these distributions. Smaller values lead to less smoothing, allowing for more accurate estimates, but are also prone to overfitting and lead to artifacts between historical sensor values. Conversely, larger kernel bandwidths prevent overfitting, but may smooth out important sensor features, and can result in overly wide estimated distributions. Below, we plot distributions when l_s is chosen either too small or too large (Fig. 49) and these can again be compared to Fig. 47 where l_s has been estimated using the procedure to be detailed in Section 4.2.6.

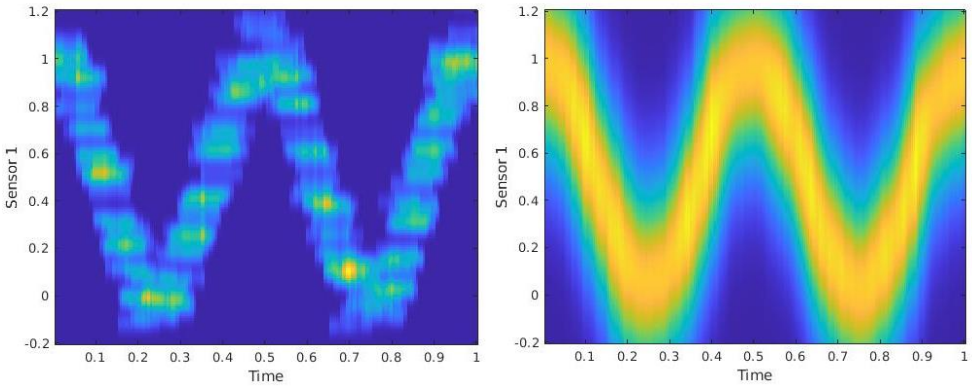


Fig. 49: Resulting sensor probabilities when the sensor length scale (l_s), are chosen too small (left) or too large (right).

Using these distributions, we now illustrate how an example fault would be detected. We consider both example nominal and anomalous signals, as well as their likelihoods of being nominal over the course of the cycle. The nominal data is drawn from the original synthetic historical distribution, while the anomalous data is also drawn from the same historical data, but remains fixed after $t = 0.5$, indicating a fault at that time. We plot the data itself (Fig. 50), as well as the probabilities for the example data for the individual sensors (Fig. 51), and the combined likelihood that the data came from a nominal process (Fig. 52).

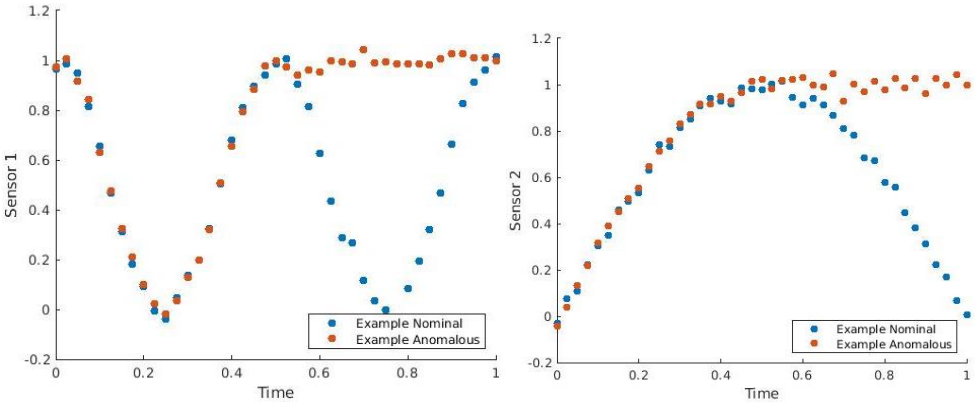


Fig. 50: Example data for sensor 1 (left) and sensor 2 (right), for nominal case, and anomalous case.

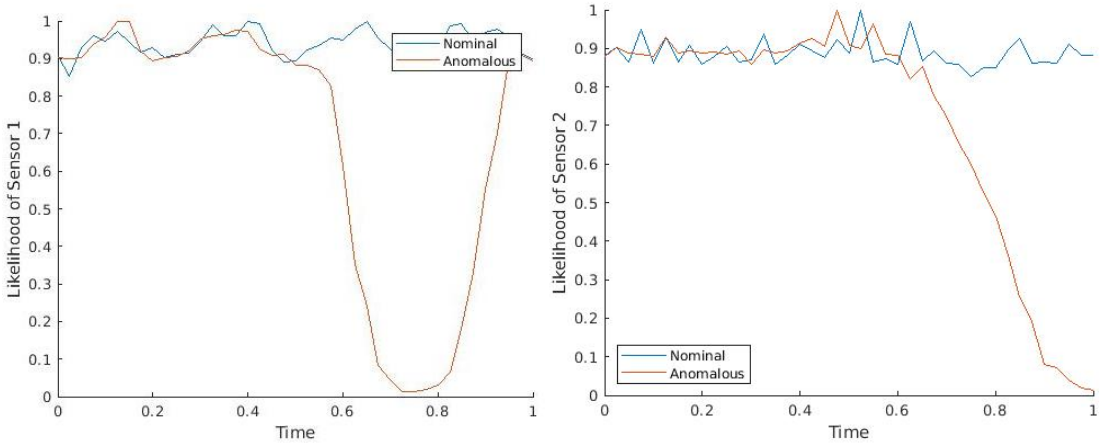


Fig. 51: Probabilities for sensor 1 (left) and sensor 2 (right) data for the nominal and anomalous cases.

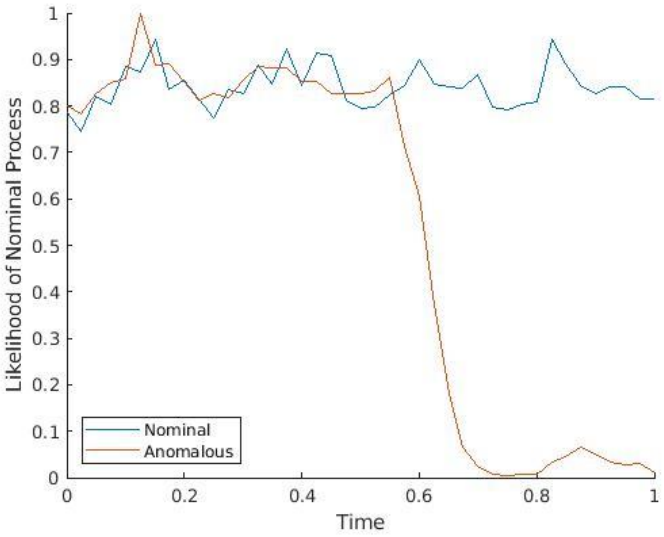


Fig. 52: Combined likelihoods as a function of time for the anomalous and nominal cases.

Several key points should be noted in these plots. First, because the example nominal sensor signals fall within the range of the historical examples, these are seen as highly probable throughout the cycle. Additionally, as both sets of sensor signals are always seen as highly probable, the likelihood that the combined nominal data is from a nominal distribution is also highly likely, demonstrating the ability to properly identify nominal signals.

Second, in the anomalous case, the first half of the cycle is seen as likely, as this data is drawn from the same distribution as the nominal data. However, after $t = 0.5$ these values diverge from the estimated distribution, and the probability of these signals, as

well as the combined likelihood of them being nominal, drops. Additionally, at the end of the run, when the value of sensor 1 once again is near its mean value, its probability increases; however, the overall likelihood is still low, as sensor 2 remains improbable.

These probabilities allow us to not only determine which points in time are seen as anomalous, but also the degree to which each sensor contributes to the overall score. At each point in time, we can look at the probabilities of the individual sensors, and determine which sensor data seem most anomalous, potentially giving insight into the source of a fault. In practice, after the most anomalous timepoint is first determined, the lowest probability sensor at that time can be identified, and its signal can be presented to a process engineer for further consideration.

4.2.6 Hyper-parameter selection

In this section, we describe the model hyper-parameters in greater detail, and present our approach for fitting them using only nominal data. These approaches rely on an acceptable false positive rate (AFPR), i.e., the false positive or false alarm rate that the model is designed to achieve. Contrary to two-class classification tasks, where both the false positive and true positive rates can be estimated for a validation dataset, we must choose our hyper-parameters using only the false positive rate because we only have access to nominal data in constructing any given anomaly detector.

The first and most critical set of hyper-parameters are the sensor bandwidths, l_s, used in the kernel function, $K_s(x_i, x_j)$. Each sensor bandwidth influences the degree that a new data point can deviate from past historical data while still being considered nominal. Noisy sensors and sensors with large variations in their historical data should have large bandwidths, since deviations in these signals must be large in order to trigger a fault. Conversely, sensors with low variance historical distributions should have small bandwidths in order to raise anomalies when incoming data deviates even a small amount from our narrow expectations.

While significant past work has investigated the selection of bandwidths for KDE applications [143], [146], [149], [150], these works are not directly applicable to our situation. The most basic techniques focus on cases where the underlying distribution is assumed to be Gaussian [143], a false assumption in many manufacturing settings (including our data, as we will show in Section 4.5). Additionally, recorded sensor data often takes discrete values which traditional methods may struggle with. Another major

problem with these past approaches is that they do not offer a method for selecting a single bandwidth that can be applied to each distribution created for each point in the cycle time, as is required in our time series methodology.

With this in mind, we propose a bandwidth selection method designed for our problem. Because our distributions are univariate, we can consider each sensor individually when determining the bandwidths. We first separate out E extremal values from the historical sensor data, where $|E| = AFPR \cdot |x_{h,s}|$, i.e., we remove extremal values corresponding to our acceptable false positive rate. To do this, we iteratively remove the point most distant from all remaining datapoints that are within a time window of half-width Δt of the data point under consideration, until we have removed a sufficient number of points.

We then iterate through the remaining points of the sensor data and find the distance to the nearest point above and below each point in its time window. Finally, we use the maximum of these distances across all timepoints as l_s. This equates to selecting a bandwidth equivalent to the largest "gap" in the historical data (Fig. 53), and ensures that that any new data falling between existing historical data will be within one bandwidth of past historical examples. This approach is particularly useful for sensors that take discrete values, as the difference between these discrete values is a desirable lower bound for l_s. While this method provides good empirical results as will be shown in Section 4.5, alternative robust methods might further reduce false positives, and could be considered for future work.

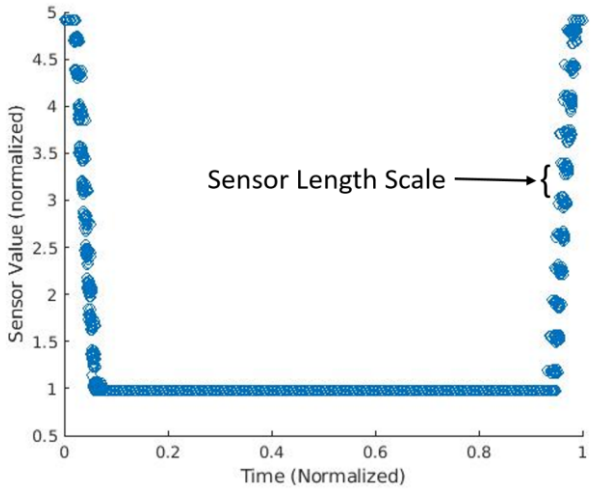


Fig. 53: Example sensor signal for nominal plasma etch data, showing largest "gap" in historical data which is used for sensor length scale.

The second set of hyper-parameters for selection are the normalization coefficients for each slice of time, $n(t)$. Determining these begins with removing the same E extremal values that were removed when determining the length scales, l_s. Then we divide the historical data into consecutive temporal bins of width Δt, and assign the normalization coefficient for each of these time windows as the minimum likelihood of all non-extremal values within the bins. This is done through cross validation: for each time window, we iterate through all points within it, and determine their likelihoods using the remaining data. The minimum likelihoods are then used for the normalization coefficients, making it such that each point in time is approximately equally likely to trigger an anomaly.

The final hyper-parameter to select is the likelihood threshold, T, used to classify incoming data as anomalous or nominal. Here, traditional methods are used. Given a historical dataset, we perform cross validation to determine a distribution of the training likelihoods. Then, given our desired AFPR, we choose the threshold such that an AFPR of those training points are classified as anomalous, and the remaining are classified as nominal.

4.2.7 Concept Drift

A common obstacle in many machine learning tasks is concept drift [151], [152], a tendency for the modelled system to change over time. While many modeling techniques require sophisticated adjustments in order to compensate for this [153]–[155], the non-parametric nature of the proposed KDE-based anomaly detection method enables a simple model update as new data becomes available.

In order to adjust to changing process conditions, we adjust the historical data, x_h, used to create the nominal distribution. Practically, we implement this set as a first in, first out, queue. When the model classifies new data as nominal, the oldest run is removed from the dataset, and the new run data is added. In contrast, if an anomaly is raised, no update to the historical known-non-faulty data and model is taken. This ensures that the historical set, and thus the nominal distribution, uses the most up to date nominal data available.

In Fig. 54, we show the effects of using this floating dataset. Without dataset updates, we see large temporal trends in the likelihood of known nominal data, while these temporal trends are eliminated in the floating model case. Using a threshold that targets an 80% true positive rate, we see a 15.1% false positive rate without a floating model, and

a 0.66% false positive rate with a floating model. This suggests that substantial concept drift is present in the underlying system, and the floating model is capable of adjusting to these changing conditions.

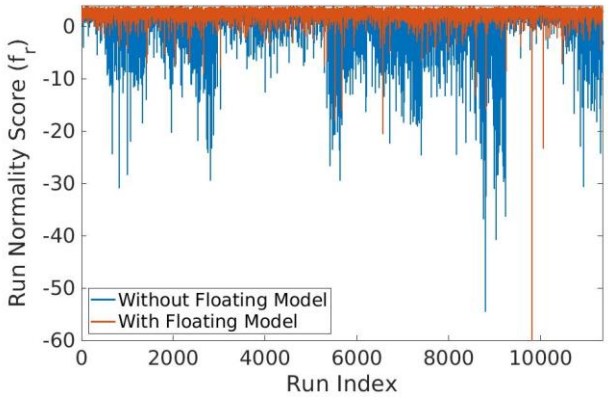


Fig. 54: Run normality score with and without floating model. Large temporal trends are seen if the model is not adjusted to changing conditions.

In terms of traditional SPC terminology, the underlying system is not in "statistical control" (because it is drifting); however, in the semiconductor fabrication context the process continues to be nominal (non-faulty). We seek a detector that distinguishes faulty from nominal historical data rather than a detector that signals "out of control." The ability to adapt to slow distribution changes is an important requirement and capability of the proposed approach.

In Fig. 55, we show the distribution of a critical sensor (wafer chuck temperature in plasma etch) at one point in the process cycle for the start of processing, and after 1500 runs. Here, we see a shift in this distribution over time, again implying changing process conditions. Importantly, under the starting distribution, values greater than 2.0 are seen to be highly unlikely, while later on, these conditions are seen to be relatively likely. Without adjusting to concept drift, these values trigger false alarms, as can be seen in Fig. 54; however, with this adjustment they are seen as nominal. In addition, Fig. 55 demonstrates that sensor signals as collected during each run for our fabrication processes can be non-Gaussian distributed.

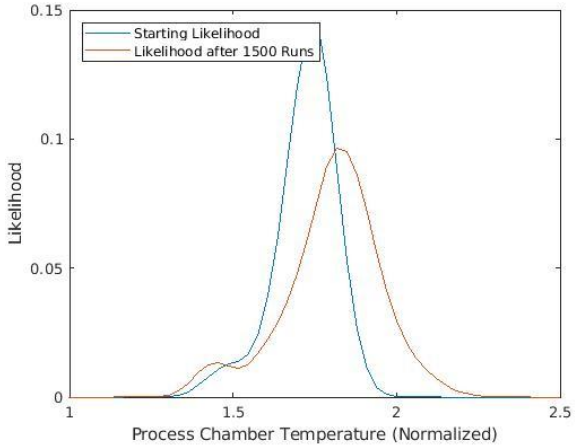


Fig. 55: Probability distribution for critical sensor at one point in process cycle. Notice high density of values greater than 2 after 1500 runs, which would be flagged as anomalous under the starting distribution.

Finally, we note that it may be beneficial to build both a floating and a stationary model, where the historical dataset, x_h, is unchanged over time. While the floating model adjusts to changing process conditions, we may want to detect these gradual changes, either for predictive maintenance, or other purposes. However, a full consideration of this extension is beyond the scope of this thesis, and could be a focus of future work.

4.2.8 Transfer Learning

In many fabrication settings, a process may encompass many different tools and recipes, each with their own unique set of nominal process characteristics, and thus each requiring their own anomaly detector. Problematically, infrequently used tools and recipes have little historical data, making them prone to overfitting. Therefore, it is desirable to transfer knowledge gained from a more frequent use case to these new or less frequent cases, a technique known as transfer learning. Historically, transfer learning has been explored in deep learning settings [156]–[159]; however, it can be applied to our method as well. Here, we present a simple but effective transfer learning technique that can be used with the proposed KDE-based anomaly detection method. This method creates a new model using limited training data in combination with an existing model from a different but related tool or recipe.

To define an anomaly detection model for a new recipe, four pieces of information are needed: example historical data, x_h, sensor bandwidths, l_s, the classification threshold, T, and the normalization coefficients, n. Several of these can be shared between recipes (e.g.,

similar recipes on the same tool) and tools (e.g., across multiple instances of the same equipment), and others need to be chosen based on the limited available data, as described below.

The first set of information, the examples of historical data, x_h, should come from the new recipe and tool of interest. Because each recipe and tool is unique and its sensor data may differ from other recipes and tools in important ways, some amount of example data is needed to model a new recipe or tool and estimate its sensor distributions; this must be provided in order to initialize a new model using our transfer learning method. However, by transferring other parameters of the model from a related tool or recipe, we require fewer historical examples in order to build an effective detector.

The second piece of information required is the set of sensor parameter bandwidths, l_s. These parameters estimate the run to run variation seen for each sensor. Critically, even if sensed process conditions such as temperature or pressure differ between recipes and tools, the variation of these conditions is often similar. Therefore, we directly transfer these parameters from an existing related model to a new tool or recipe when initializing the new model. Directly transferring these from one recipe to another is crucial, because selecting these length scales is often the most data intensive part of the training process. Therefore, transferring these key model parameters from one scenario to another allows us to build a new fault detection model with substantially less data.

The third piece of information needed to define a new model is the set of normalization coefficients, n. Unfortunately, we cannot easily transfer the normalization coefficients from one recipe to another. Because different recipes may have different numbers of recipe steps, each with different conditions and lengths, the underlying distributions vary greatly between recipes. For this reason, it is best to explicitly re-determine the normalization coefficients for a new recipe, as described in Section 4.2.6. While these are determined using a significantly smaller training set, in practice, we do not find a substantial negative impact on the classification accuracy, as will be shown in Section 4.4.

The fourth and final piece of information required to define our model is the classification threshold, T. Recipes or tools with ample historical data should have approximately the same classification threshold, as scaling the number of example points should leave the KDE distribution relatively unchanged when there is sufficient data. However, in cases where there is limited data, which are often common in transfer

learning scenarios, the KDE distribution may be inaccurate, and it can be desirable to choose a lower threshold to avoid increasing the false alarm rate.

In these scenarios, we use data from the frequently used process, $x_{h,1}$, and the infrequently used process, $x_{h,2}$, to choose a threshold for the data-limited model. We chose this threshold using a cross validation scheme evaluated on ensembles of $x_{h,1}$ with training size $|x_{h,2}|$, ensuring that a threshold has been chosen using a historical set of the correct size. We create ensembles of training data using a sliding window of size $|x_{h,2}|$, then determine the likelihoods of the sample of $x_{h,1}$ immediately following the sliding window. This matches the temporal relationship between the historical and incoming data, allowing for an accurate estimate of the distribution of testing likelihoods. For each training ensemble of $x_{h,1}$ created using a sliding window of data, we estimate the sensor signal distributions using KDE and the original model parameters, then calculate the run anomaly likelihood score of the testing sample as described in Section 4.2.3. Finally, we choose a threshold that results in our desired $AFPR$ for these cross validation likelihood scores, and use this as the threshold for the new model.

After determining these four pieces of information (example historical data, x_h, sensor bandwidths, l_s, the normalization coefficients, n, and the classification threshold, T), we can then apply our proposed fault detection method to data from the new recipe or tool. Evaluating new likelihood scores is performed just as was described in Section 4.2.3; however, the new model parameters are used when implementing this new model.

4.3 Benchmark Methods

In this section, we present several benchmark methods to compare with our proposed approach. We discuss traditional statistical process control (SPC) methods, one-class SVMs (OC-SVMs), and anomaly detection methods based on variational auto-encoders (VAEs).

Statistical process control methods are traditional approaches for monitoring semiconductor and other manufacturing processes [160]–[162]. These methods produce control limits for sensor signals, and raise alerts when incoming signals fall outside of their acceptable range. Typically, the distributions of the underlying signals are assumed to be Gaussian, and the acceptable limits are determined using this assumption. For extremely small amounts of training data, the student-t distribution is also used in place

of the Gaussian approximation [163]. As the difference between these two distributions is negligible for our training sizes, commonly near 250 data points, we choose to use the more common Gaussian approximation in our SPC benchmark anomaly detectors.

In cases where multiple sensor signals exist, such as ours, multivariate probability distributions can be used to determine the process control limits. In our comparison method, we use an uncorrelated multivariate Gaussian distribution to approximate the normal operating distribution of a process. This combines the univariate probabilities from each sensor into a single probability, which is then compared to an anomaly threshold in a similar fashion as our proposed KDE method. Multivariate distributions that create correlated distributions are also commonly used in SPC, and are capable of detecting a wider range of anomalies; however, accurately estimating the correlation structures requires more data. Because we seek anomaly detectors in cases where we have a limited training dataset, the uncorrelated distributions are also used in our SPC benchmark method.

When creating these distributions, we must determine their mean and standard deviations. Here, we implement two methods to compare our proposed method to. First, we use a time-invariant estimate of the Gaussian coefficients. In this approach, the mean and standard deviation of each sensor signal are determined using all data, regardless of when they occur in the process run. This is the simplest SPC approach and serves as a naïve baseline method, recognizing that it will miss many anomalies by expanding the control limits to cover the extent of normal signals across all time points in a run.

The second approach is a time-variant SPC method. Here, the distribution mean and standard deviation change within the process run time. This is implemented by dividing the historical data into temporal bins of size Δt, then creating a unique mean and standard deviation for each bin. This allows the Gaussian distribution to change within the course of the process run, giving a more refined estimate of the nominal process distribution. In both the time-invariant and time variant SPC methods, we use a floating dataset to adjust to changing process conditions. As the historical data changes, we also update the nominal distributions, which adjusts the model to changing process conditions. We note that this time-variant SPC method is similar in many ways to the KDE-based approach, with a key difference being the rigidity or flexibility in representing the underlying probability density functions.

In addition to these benchmark SPC methods, we also compare our proposed method to an SVM-based anomaly detector. Traditional SVMs are a common classification technique, used in countless applications [164]. These methods create a separating hyperplane that maximizes the distance between two classes. One-class SVMs are an extension of traditional SVMs, and create classification boundaries when data from only a single class is available. These one-class SVM methods are also widely used in various industrial settings [32], [165]–[167].

One-class SVMs have two common implementations. The first implementation, named OC-SVM [27], creates a classification boundary that separates data in the known class from the origin. The second implementation, Support Vector Data Descriptions (SVDDs) [28], instead create a hyper-sphere that contains data from the example class. Interestingly, when a radial basis function kernel function is used, both methods produce identical classification boundaries. Since this is the most common kernel and the one that we use, we treat these two methods as equivalent.

Because time series data can have many potential channels or sensors, feature reduction techniques are often required when kernel based methods such as SVMs are applied [34], [168]–[170]. The most common method is principle component analysis (PCA), which reduces all feature information of size $|s| \cdot |t|$ down to a smaller number of principle components. We thus use this approach as a preprocessing technique in conjunction with OC-SVMs, in order to reduce the problem size to what can feasibly be fit with OC-SVMs. In all results presented in the next section, the number of principle components used is equal to the number of training examples. This is the maximum number possible with limited training examples, and in all cases using fewer components resulted in worse performance.

Finally, we compare our approach to anomaly detection methods based on variational auto-encoders (VAEs). Auto-encoders are a deep-learning based feature reduction technique comprised of an encoder and a decoder. The encoder is commonly implemented as series of convolutional and fully-connected neural network layers that reduce an original high-dimensional data point, such as a picture, or set of time series sensor signals, down to a small number of latent features. These features are then expanded by a decoder, again implemented as a series of convolutional and fully connected layers, in order to reconstruct the original image. VAEs are an extension of traditional auto-encoders that create a probability distribution for the latent features

instead of single set of values [171]. This acts as an effective regularization technique, so that small perturbations in the latent space produce appropriately perturbed reconstructed signals.

VAEs are widely used for dimensionality reduction; however, they have also shown promise for one-class anomaly detection problems. Historically, there are two common approaches for utilizing VAEs for this task. The first uses the reconstruction error as a classification metric [172]–[174]. For incoming signals similar to the nominal set of training signals, the reconstruction error should be low, as the network is designed to compress then reconstruct nominal data. However, because the network was not trained on faulty data, it should do a poor job reconstructing signal sets from faulty runs, allowing the mean squared error (MSE) of the reconstructed signal to be used as a classification metric.

The second common approach creates a probability distribution for the latent features of the nominal training data, then uses the probability of incoming data under this distribution as a metric for classification [175], [176]. Incoming nominal run samples should have similar latent space representations to the training set, and thus high probabilities, while faulty run samples should have significantly different latent space representations, and thus low probabilities. Because the reconstruction error approach is more common, we use this approach for our VAE-based anomaly detector benchmark.

4.4 Results

In this section, we present results for our proposed KDE-based method, and compare with the benchmark methods from Section 4.4. We first show results of all methods in the standard one-class classification setting, where models are trained and evaluated on data from the same recipe. Afterwards, we present results for the proposed transfer learning method from Section 4.3 as applied to both the plasma etch and ion implant datasets, and show that similar performance can be achieved as in the single recipe case.

4.4.1 Single Recipe

Here, we compare the results of the proposed KDE method with the time-invariant and time-variant SPC methods, OC-SVM with PCA, and a VAE-based classifier. We test these five methods on data from one ion implantation recipe, and two plasma etch recipes. In all cases the models are trained with 250 training examples from the time before any

known anomalies reside in the data. We then apply the trained anomaly detector on the remaining data (11,174 examples in the ion implant case, and 1,993 and 1,386 examples in the two plasma etch cases).

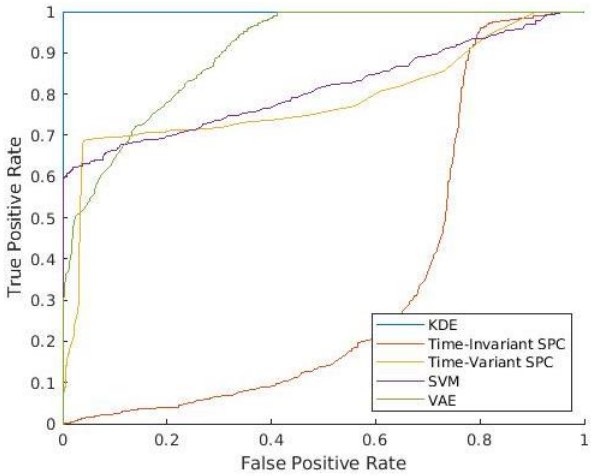


Fig. 56: ROC curve of proposed and comparison methods when applied to plasma etch recipe #1.

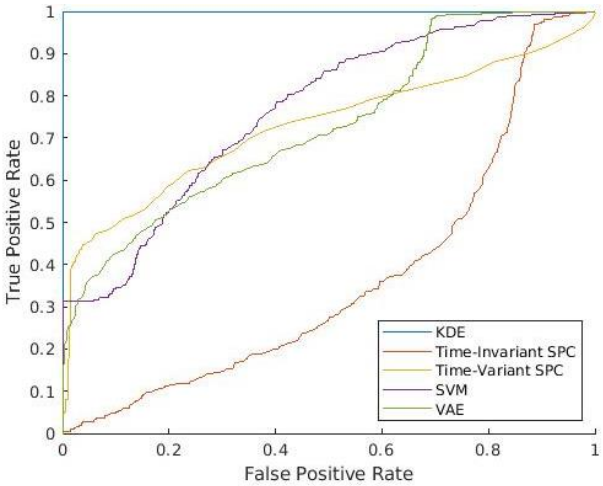


Fig. 57: ROC curve of proposed and comparison methods when applied to plasma etch recipe #2.

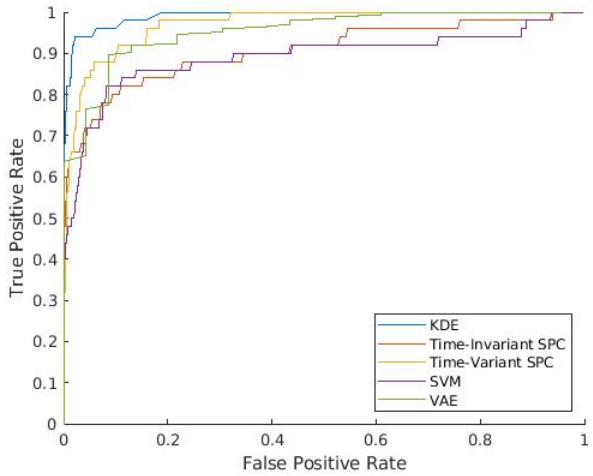


Fig. 58: ROC curve of proposed and comparison methods when applied to ion implant data.

TABLE II
Area Under ROC Curve for all methods and datasets.

Method	Plasma Etch Recipe #1	PLASMA ETCH RECIPE #2	Ion Implantation
KDE	1.00	1.00	0.99
Fixed SPC	0.33	0.36	0.91
Time variant SPC	0.79	0.74	0.97
PCA + OC-SVM	0.81	0.77	0.89
VAE	0.90	0.72	0.91

TABLE III
Single recipe TPR results for a 1% FPR.

Method	Plasma Etch Recipe #1	PLASMA ETCH RECIPE #2	Ion Implantation
KDE	100%	100%	82.0%
Fixed SPC	0.21%	0.54%	62.0%
Time variant SPC	16.1%	8.80%	60.0%
PCA + OC-SVM	60.1%	31.2%	48%
VAE	37.0%	23.7%	63.58%

We show the receiver operator characteristic (ROC) curves, for all methods, and all datasets in Fig. 56 through Fig. 58, and Table II shows the area under the curve (AUC) for these results. Additionally, Table III shows the true positive rate (TPR) that corresponds

to a 1% false positive rate (FPR) for all models, allowing for a straightforward comparison of real world performance.

In all of these results, we observe that the proposed KDE method outperforms the comparison methods by substantial margins. When applying our proposed method to the plasma etch datasets, we achieve 100% classification accuracy, while all of the comparison methods perform substantially worse. Similarly, applying our method to the implantation data again results in better classification results, as we achieve higher true positive rates for all false positive rates when compared to the additional methods.

It is important to note that many of the comparison methods produce similar classifications for relatively "easy" anomaly cases. These cases correspond to anomalies that are clearly visible in Fig. 41 and Fig. 42, and are detected by all anomaly detection methods we investigate. The remaining anomalies, however, prove challenging for the benchmark methods to detect, and the improved performance in these difficult cases highlights many of the advantages of our proposed method. The improvement in performance of the proposed KDE-based method compared to the benchmark methods is due to a number of factors which each highlight different advantages of the proposed method.

The worst performing model is the time-invariant SPC method. This can be seen in the AUC of this model, which is almost always the lowest when applied to the different datasets. This poor performance is primarily due to the lack a time-aware model, which is unique among the comparison methods. Since this method assumes constant sensor signal distributions for all points in time during a run, it must create excessively wide nominal distributions in order to cover all sensor data throughout the process runtime. This leads to more false negatives, as the model believes that these signals can take an unrealistically wide range of values, thus missing true anomalies.

Interestingly, this time-invariant SPC performs worse than a random guess for the plasma etch datasets, as seen in the ROC curve. This is a result of using the floating dataset, which is meant to account for changing process conditions; however, this technique can be susceptible to faulty data entering the historical dataset. This is particularly problematic when used in combination with low accuracy models (such as the time-invariant SPC method). When these low accuracy models incorrectly classify incoming faults as nominal, they are added back to the historical dataset. This leads to a feedback loop, where new incoming faults are seen as increasingly likely, while nominal data

increasingly differs from the saved historical dataset. The net effect is that the detector becomes worse at distinguishing faults from nominal data, as both are in the historical dataset, leading to even worse detection rates. Importantly, this effect is not seen in the time-variant SPC and KDE approaches, as both detectors are relatively accurate to begin with, so it is unlikely that substantial amounts of faulty data are added back into the training set.

The results for the time-variant SPC method are substantially better than those of the time invariant method. This is primarily because it (correctly) assumes that the sensor signals change over the course of a run. The resulting time-variant sensor distributions are often tighter at each point in time, since they only reflect the signals at a single time point, and are thus more likely to detect faults. Additionally, this improved initial classification performance also diminishes the negative effects of floating historical dataset, which is particularly problematic for the time-invariant SPC model as previously noted.

While the time-variant SPC method is time-aware, it still suffers from the assumed underlying Gaussian distribution. We find that processes with abrupt transitions, such as that in Fig. 43, commonly break this assumption. In Fig. 59, we see that the KDE estimate during this abrupt transition has a clear non-Gaussian distribution. Under the Gaussian assumption in the time-variant SPC anomaly detector, many of the tail points at this transition are seen as highly unlikely. In order to retain a 1% FPR, a low likelihood threshold must be used, which decreases the TPR. Because the proposed KDE method does not suffer from this assumption, its performance is often better in these key cases.

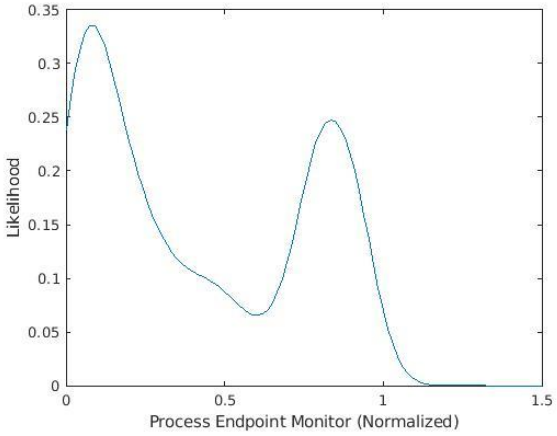


Fig. 59: Critical sensor signal KDE distribution slice at normalized cycle time t = 0.6, showing non-Gaussian underlying distribution. Estimation with Gaussian distribution leads to substantial number of false alarms.

Both the OC-SVM and VAE methods share two common problems. First, neither account for concept drift, which greatly decreases the likelihood of many nominal cases, and the total model accuracy, as was seen in Section 4.2.7. Secondly, both sub-optimally normalize the sensor data. The VAE and OC-SVM normalize data to ensure constant variance over all time-series data, i.e., the set of time-series data for each sensor are normalized to have standard deviation of one in the preprocessing stages of both of these approaches. In contrast, the KDE bandwidths (which effectively normalize the signals) are chosen based on the variation of sensor data between runs. Critically, a sensor signal may have large variance over the course of a run, but may have little variation between runs. In these cases, any small sensor deviations may seem unimportant under the VAE and OC-SVM methods, as these variations are small compared to the overall signal variance; however, these are interpreted as significant under our KDE approach, as there is historically little run to run variation. When classifying new data, the OC-SVM and VAE approaches are therefore more likely to miss minor deviations in sensors with high within-run variations compared to the proposed KDE approach. Conversely, the normalization approach of the KDE method helps ensure high sensitivity to historically stable sensors, and also helps prevent false alarms due to noisy signals.

Additionally, we believe that the poor performance of the VAE is in part due to overfitting of the model. In these results, we observe substantially higher reconstruction errors for the nominal testing set when compared to the nominal training set. This suggests that the network is overfit to the training set, and may not be able to properly reconstruct unseen nominal data, making it difficult to classify anomalies using the reconstruction error.

Finally, we believe that the poor OC-SVM performance also suffers due to critical information lost in the PCA stage. While PCA is a common feature reduction technique, we believe it is poorly suited for one-class anomaly detection tasks. PCA chooses a feature set that explains the maximum variation in its training set. When applying it to multi-class data, the chosen features often represent differences between those classes. However, in the one-class case, PCA instead chooses features that represent differences *within the nominal set*, and are not necessarily effective for representing, or detecting, differences between nominal and faulty data. This leads to poor performance when these features are then used for classification. This can be observed when reconstructing critical faulty signals using the poorly chosen PCA basis (Fig. 60). Here, we see that when performing

PCA, the reconstructed signal significantly differs from the original signal before PCA is applied, and is more similar to the average known good signal (the mean signal value in Fig. 60). This indicates that important information has been lost, and the resulting representation will often be insufficient for fault detection purposes as it fails to capture the distinguishing features of the faulty signals. Unfortunately, this feature reduction is needed with OC-SVM as using raw features is computationally infeasible.

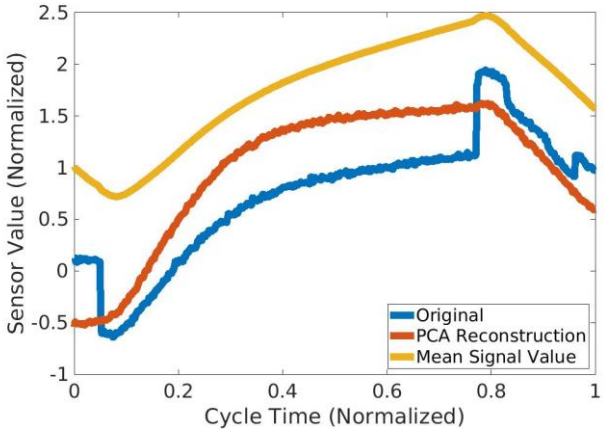


Fig. 60: PCA features chosen using only nominal data do not accurately describe faulty sensor data leading to poor classification results.

Finally, in Table IV, we show the TPR and FPR of our proposed model for all datasets when an a priori threshold is chosen with a AFPR of 1%. When the threshold is chosen a priori using our method proposed in Section 4.3.6, we see similar results to an a posteriori threshold resulting in an exact 1% FPR. This demonstrates that we can select a good classification threshold without faulty examples, as would be required in a real world application of our method.

TABLE IV
Single recipe KDE results with a priori threshold.

Method	Plasma Etch Recipe #1	Plasma Etch Recipe #2	Ion Implantation
KDE FPR	1.25%	1.09%	0.39%
KDE TPR	100%	100%	72.0%

4.4.2 Transfer Learning

Here, we present results for our transfer learning methodology with KDE-based anomaly detection. Since our ion implantation and plasma etch datasets each contain different numbers or proportions of available faulty and nominal recipes, we present

slightly different results for the two cases. The available ion implantation dataset has 11 recipes with only nominal data, and only one recipe with nominal and faulty data. We therefor choose the model hyper-parameters using the 11 partial recipes, then test the performance on the single complete dataset. The plasma etch data includes two recipes, but both contain faulty and nominal data. Thus, we select hyper-parameters from each etch recipe dataset, then test on the other recipe. In both the plasma etch and ion implantation data, we select hyper-parameters using 250 example runs, then create a model with 25 example runs from a new recipe. We set an AFPR of 1%, and present the resulting FPR and TPR for the ion implantation (Table V) and plasma etch cases (Table VI).

TABLE V
Transfer learning results with ion implantation data.

Training Recipe	Transfer Learning FPR	Transfer learning TPR
Recipe 1	1.46%	82.0%
Recipe 2	1.49%	72.0%
Recipe 3	2.92%	90.0%
Recipe 4	1.58%	80.0%
Recipe 5	0.42%	68.0%
Recipe 6	0.31%	60.0%
Recipe 7	1.75%	76.0%
Recipe 8	0.35%	48.0%
Recipe 9	1.42%	72.0%
Recipe 10	1.53%	80.0%
Recipe 11	0.52%	64.0%

TABLE VI
Transfer learning results with plasma etch data.

Training Recipe	Transfer Learning FPR	Transfer learning TPR
Recipe 1	2.94%	100%
Recipe 2	4.31%	100%

In all cases, transfer learning produces similar results to training with a full dataset; however, the performances are not identical. In the plasma etch cases, the FPRs are higher than in the single recipe cases; this is expected considering the smaller training size. In the ion implantation cases, the FPRs and TPRs vary greatly when different training sets are used. We believe this is primarily due to differences in the choice of threshold selected in the training stage. The FPRs and TPRs are highly varied, but positively correlated, indicating that the threshold is likely sub-optimally chosen. Selecting too low a threshold lowers both the FPR and TPR, and the opposite effect results when selecting too high a threshold. These results are promising even with only 25 training runs used for each case,

enabling relatively accurate detection for infrequently used recipes. This also enables the rapid startup of the anomaly detector. As additional runs are obtained, the accuracy of the detector improves due to the growth in the size of the example set, e.g., to a stable 250 samples after which the first-in, first-out update to the basis data can be performed.

4.5 Conclusions

We propose a one-class KDE anomaly detection method for semiconductor processes. Extensions enable adaptation to concept drift, as well as transfer learning between recipes. The approach is demonstrated with good success across two plasma etch recipes, and ion implant, in both single recipe and transfer learning scenarios. These results compare favorably to benchmark one-class anomaly detection methodologies, including naïve and time-varying SPC methods, OC-SVM, and VAE fault-detectors. Transfer learning between recipes is also demonstrated, enabling rapid model creation for related recipes and tools.

We believe the KDE-based anomaly detection method is attractive for a number of reasons: it is trained using only nominal data, it can be trained with relatively small amounts of data, and finally it is process indifferent, as it makes few assumptions about the modeled process. These qualities should allow it to be applied to a large number of processes and recipes, making it practically useful for semiconductor anomaly detection applications.

This case study again highlights the two key themes seen throughout this thesis. First, the incorporation of process knowledge allows us to train our model with less data. Primarily, this is seen in our data pre-processing. As we believe the majority of faults in our semiconductor context can be detected on a pointwise and univariate basis, we are able to create individual sensor distributions for each point in time, drastically reducing the data required to estimate these distributions. This allows us to train with limited data, outperforming generic feature compression methods such as PCA. Secondly, we see that the use of probabilistic methods outperforms deterministic comparison methods such as VAEs and OC-SVMs. Even in extremely data limited transfer learning scenarios, our method is capable of good performance, confirming its applicability to these data limited scenarios.

Future work could explore further extensions. One is the incorporation of anomalous data into the model. While this data may not always be available, there may be times when

it is, and faulty examples may improve the training and prediction capabilities of the model. Additionally, robust bandwidth selection techniques available in some KDE applications could be explored, as described in Section 4.3.6. Additionally, a variant of our approach is to use a fixed model to detect long-term system drift, as discussed in Section 4.3.7. These long-term drifts may be useful for preventative maintenance, or for detecting gradual degradation of a tool. Finally, extensions could be considered that explicitly consider the time-series nature of the data. The KDE-based detector method presented here looks at sensor data on a point by point basis, avoiding the complexity of time-series modeling; methods combining time-series and KDE are an interesting avenue for future research.

5 Modeling and Optimizing Sputtering Deposition Systems Using Gaussian Processes

In this chapter, we explore the use of Gaussian Process (GP) models in order optimize thickness uniformity in sputtering deposition processes. Thin film deposition is a fundamental process used in the fabrication of nearly all semiconductor products [177]. While there are many variants of this process, one is the sputtering deposition process. In these processes, ions are accelerated towards a target, striking the target, and sputtering material onto a nearby (rotating) wafer (Fig. 61). This creates a thin film of material on top of the wafer that forms the basis for many semiconductor structures.

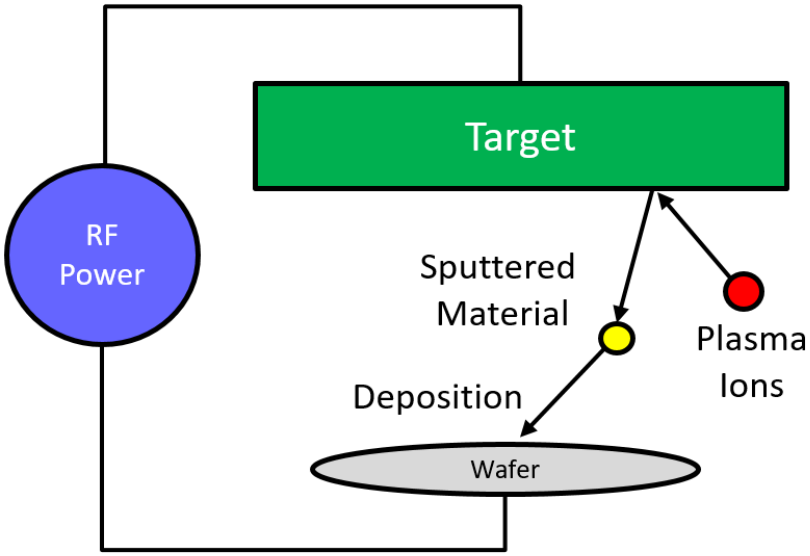


Fig. 61: Illustration of sputtering deposition process. Plasma ions (red) are accelerated by an RF power source (blue) towards a target (green). Material (yellow) is then sputtered from the target and deposited onto the wafer (grey).

Unfortunately, this deposition may not be evenly distributed across the wafer. Some areas may receive more material, resulting in thicker films, while other may receive less (Fig. 62). As these thicknesses later translate to device geometries, these spatial variations in the deposition thickness can lead to significant differences in device performance. To prevent these variations, equipment configuration or recipe settings can be adjusted in order to more evenly distribute material across the wafer. The effects of

such parameters on the thickness profile can first be modeled, then this model can be used to select configurations or recipe settings that meet a desired uniformity specification.

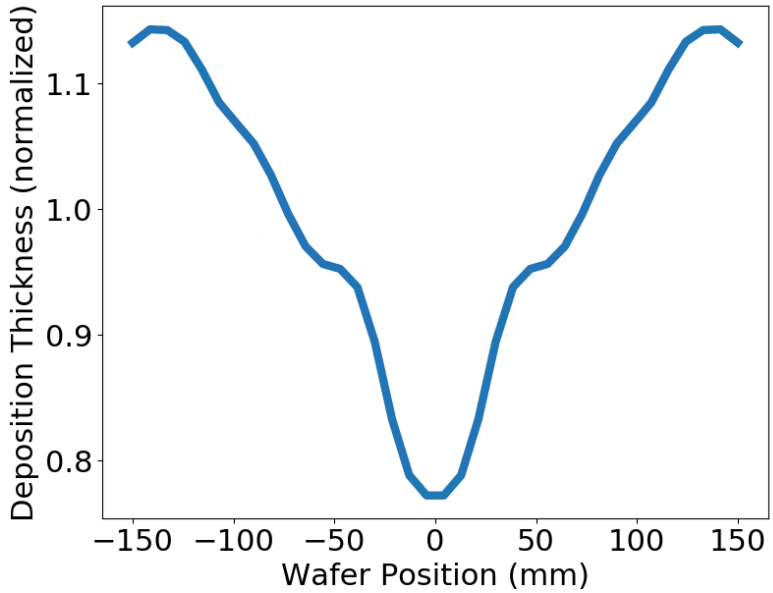


Fig. 62: Example 73 point sputtering deposition thickness profile demonstrating non-uniform deposition.

However, modeling the effects of these model inputs is not a trivial task. Because the system is particularly complex, models based on first principles are infeasible to apply, and empirical approaches must be used. Additionally, this process has a large number of equipment configuration or recipe parameters, each with potentially complex relationships to the output of interest, and the collection of new data is extremely costly, thus limiting the feasible amount of training data.

Because of these constraints, traditional modeling approaches such as polynomial models and deep learning methods can be insufficient for many process optimization tasks. Simple polynomial methods struggle to represent complex relationships, and neural networks are held back by small amounts of available training data. Therefore, semiconductor process optimizations tasks are in need of a highly expressive machine learning method that can accurately predict key process variations with limited training data.

Prior work has demonstrated the ability of GPs to model and control variations of other processes [84], [95], [178], [179], and here we show that they are well suited for our proposed application for two reasons. First, a key benefit of GP models is their ability

to learn complex functions with relatively small amounts of training data [85], [180]. This is critical, as a key constraint in tuning a new chamber or recipe is the cost of running new deposition experiments. Adjusting chamber parameters, running the process, and measuring the resulting profile is an extremely costly process, and minimizing the number of required adjustments is the primary metric of tuning procedure success. For this reason, we not only consider the predictive power of a model, but are primarily concerned with how effectively it can be used for process optimization. Secondly, GPs also provide predictive distributions, as opposed to deterministic outputs. As we will see, this is a critical advantage over deterministic methods, as the confidence of these GP models can be used to select inputs which are not only predicted to meet the desired specifications, but are also likely to be accurate, reducing the effects of overfitting to limited data.

In this chapter, we investigate two models that predict deposition thicknesses across the wafer: one considers the effects of the scalar process recipe parameters (pressure, target to wafer spacing, bias voltage, and radio frequency power) and a second model considers the effect of complex discrete parameters related to deposition chamber equipment configuration. This second case is the main focus of the chapter, as the greater number of inputs makes it the higher complexity modeling task. In this case, we also incorporate prior process knowledge in the form of a physics-based pre-processor. This reduces the large number of process inputs to a more feasible size, and which are believed to be more directly related to the deposition thickness.

In Section 5.1, we give an overview of Gaussian Process (GP) models. In Section 5.2, we employ the GP model framework to model the scalar process recipe parameters such as pressure, and power supply bias. In Section 5.3, we then explore how the same GP model framework can be combined with physics-based solvers to efficiently model the effects of large numbers of highly non-linear input parameters, including geometric equipment configuration parameters of the deposition chamber. In Section 5.4, we discuss how this model can be used to iteratively tune a process. In Section 5.5, we present polynomial, neural network, multivariate spline, and gradient boosted regression tree comparison methods which we benchmark our model against. In Section 5.6, we present the results of the proposed model, along with benchmark comparisons. Finally, in Section 5.7, we offer conclusions and suggestions for future work.

5.1 Gaussian Process Models

In this section we give a brief overview of Gaussian Processes, which are the basis of our proposed model. GPs are a probabilistic regression model that create likelihood distributions for new inputs based on previously observed outputs, and the similarity of their inputs and the new inputs.

Specifically, a GP models a collection of outputs, in our case the individual deposition thicknesses, T_i, as a multivariate normal distribution [181]:

$$\begin{bmatrix} T_1 \\ \dots \\ T_n \end{bmatrix} \sim N\left(0, \begin{bmatrix} \sigma_{11} & \dots & \sigma_{1n} \\ \dots & \dots & \dots \\ \sigma_{n1} & \dots & \sigma_{nn} \end{bmatrix}\right) = N(0, \Sigma). \tag{73}$$

The correlation structure of these outputs, Σ, determines how similar each output is to one another. This correlation structure is determined by the similarity of the corresponding inputs, X_i, in our case the process conditions and radial position. The more similar the process conditions and radius are, the more correlated their outputs are assumed to be. The function that determines the covariance used in the multivariate distribution is referred to as the kernel function, K:

$$\sigma_{ij} = K(X_i, X_j). \tag{74}$$

For our purposes we use a common kernel function, the radial basis function (RBF), also known as a Gaussian or squared exponential kernel (Eq. 75), with the addition of pointwise white noise. With this kernel, two inputs are correlated up to the pointwise noise when they are the same, and as the distance between them grows, the correlation between them smoothly decays with a length scale l. In particular, we use an anisotropic version, where the length scales of the kernel, l_p, can differ for each process parameter, $X_{i,p}$, since the scales of each parameter can be different:

$$\sigma_{ij} = \sigma \cdot \prod_p e^{-\left(\frac{X_{i,p} - X_{j,p}}{l_p}\right)^2}. \tag{75}$$

When predicting an unobserved deposition profile, $\overrightarrow{T_*}$, for a new set of process or equipment conditions, X_*, we calculate the distribution for these variables by creating a distribution for all variables, and marginalize out the observed outputs, $\vec{T}$ and inputs X. This becomes a new multivariate Gaussian distribution with mean μ^* and covariance Σ^*:

$$\overrightarrow{T_*} \mid \vec{T}, X, X_* \sim N(\mu^*, \Sigma^*) \tag{76}$$

where

$$\mu^* = K(X_*, X)K(X, X)^{-1}\vec{T} \tag{77}$$

$$\Sigma^* = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*). \tag{78}$$

An example predictive distribution generated with a GP model for a single variable function with limited data points can be seen in Fig. 63.

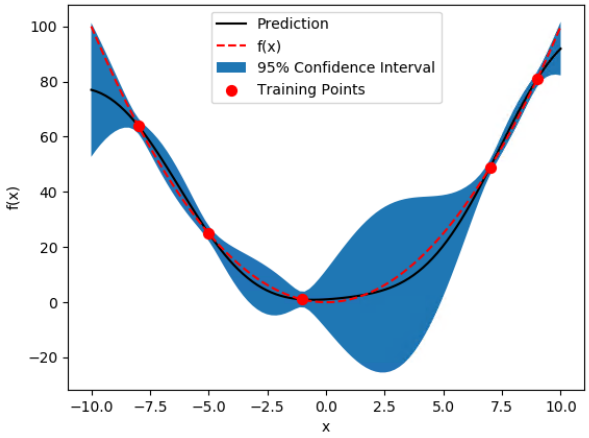


Fig. 63: Example GP model for $f(x) = x^2$ with four training data points.

A key benefit of GP models are the reliable confidence intervals provided with a prediction. These confidence intervals are useful in a process optimization setting as it not only allows us to choose new process or equipment parameters whose most likely output meets specifications, but also allows us to consider how likely this outcome will be.

5.2 Process Recipe Parameter Modeling

In this section, we demonstrate how the Gaussian Process framework can be applied to modeling the deposition profile as a function of the process recipe parameters, such as chamber pressure and target-to-wafer spacing. While a model based on first-principles physics may seem desirable, the underlying physics of the process are highly complex, and may depend on many other factors such as the chamber geometry and configuration, the properties of the deposition material, and the plasma composition [182]. As we will see, modeling the impact of the recipe parameters with an empirical model allows us to bypass these complex physics while still producing an accurate model.

In order to collect data to build the model, we first choose a set of recipe parameters to model the impact of, then create a design of experiments (DoE) to plan our data collection. In this section, we model the impact of four scalar recipe parameters: chamber pressure, RF power, bias, and target-to-wafer spacing. We design a 17-point fractional factorial central composite DoE. After choosing our DoE, we sputter our target material onto a wafer under each of these conditions and measure the resulting profiles.

In addition to our process recipe parameters, we must also consider the radial position on the wafer as an additional model input, as the deposition thickness changes across the wafer. Note that during sputtering, the wafer is rotated in order to improve uniformity. For this reason, deposition thickness does not significantly depend on the angular position on the wafer, but only on the radial position, i.e., all points on the same radius have approximately equal thicknesses. Therefore, when measuring thickness variations across the wafer we only require a scan across the center of the wafer, instead of collecting data from the entire wafer. For the same reason, we only use the radial position as the wafer position input for our GP model, and only plot this central scan when plotting thickness profiles.

After collecting the experimental data, we then build and validate our GP model. Here, our GP model takes in the scalar process recipe parameters, p, along with the radial position, r, as the model inputs, and maps these to the deposition thickness, $T(r, p)$.

Because all hyper-parameters required for our process recipe GP model are chosen in training, most importantly the kernel length scales, we use cross-validation to evaluate our model rather than training, validation, and testing splits where the effects of hyper-parameters are also investigated. Here, we train our GP model using all spatial observations except those from a single omitted set of process conditions. We then predict the deposition profile of the omitted wafer, and calculate the root mean squared error (RMSE), R-squared (R^2), and mean absolute error (MAE) of our prediction using data normalized to their mean. Finally, we repeat this procedure for all wafers, and average these metrics over all testing wafers to calculate the cross validation metrics, CVRMSE, CVR^2, and CVMAE, respectively. These values are 0.061, 0.98, and 0.054 for the three metrics, respectively, demonstrating the good ability of the model to predict deposition profiles for new process conditions. Example predictions compared to their experimental profiles are shown in Fig. 64.

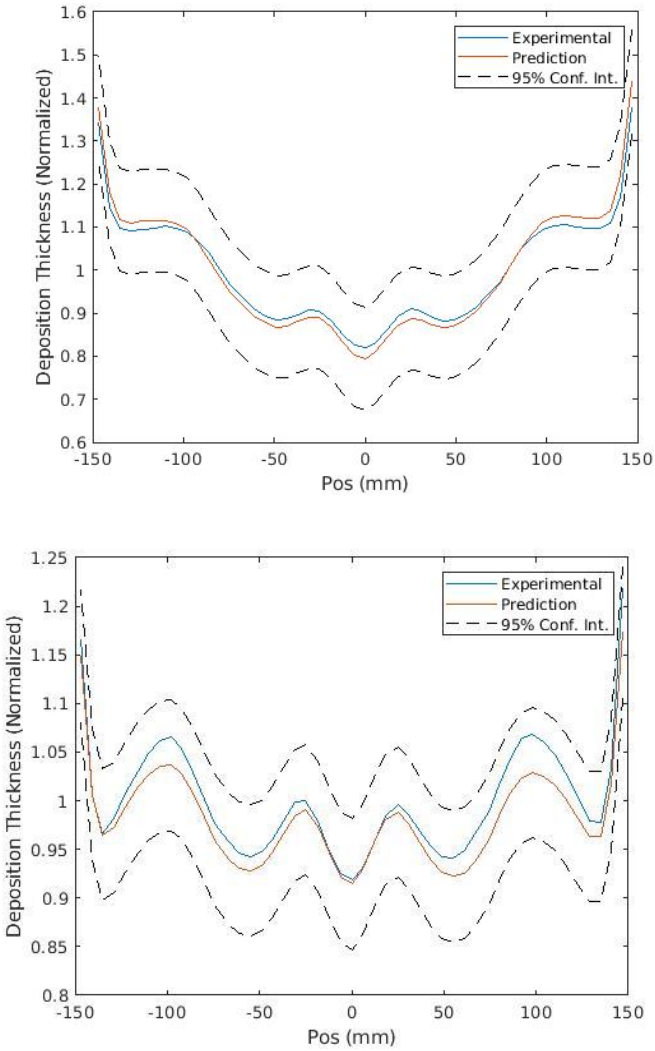


Fig. 64: Two examples of predicted vs. experimental deposition profiles.

In each training, the GP model nearly perfectly fits the training data, and thus results in a training R^2 value of approximately 1. The cross validated R^2 value estimates the model accuracy when we predict new conditions, and is 0.98 for the process recipe parameter models using our 17-point DoE.

5.3 Chamber Configuration Modeling

In this section, we demonstrate how the same GP framework can be used to model the deposition profile as a function of complex, geometric, chamber configuration variables. These equipment configuration variables allow for much finer tuning of the deposition profile, making them well suited for reducing the high frequency spatial variations seen in many deposition profiles, beyond what can typically be achieved with only process

recipe optimization. Here, we incorporate physics-based solvers into our modeling framework and add prior knowledge to the model in the form of additional structure, to create the overall model pipeline (Fig. 66). A large number of equipment configuration variables are our overall model input; these number about 100, compared to only four in the process recipe case. For any given equipment configuration, intermediate spatial variables are then computed using the physics-based solver; these provide spatial information about chamber or plasma conditions at a particular horizontal cut (parallel to the wafer) through the chamber. Feature selection is then performed on these intermediate spatial variables, resulting in distributions of these variables at each radius that are used as inputs to the GP model.

This approach highlights a key theme of our thesis, incorporating process knowledge into the modeling approach. Without this additional model structure, the original process inputs would be too numerous, and would have highly non-linear relations to the deposition thickness, making it infeasible to empirically model. In this section, we first describe the variables of interest, then the model structure, and finally the model fitting procedure, corresponding to the model components in Fig. 66.

5.3.1 Variable Overview

The equipment variables of interest, G_i, affect the geometry of the chamber, are high dimensional (there are approximately 100 independent variables), and are binary (they must take one of two states). The deposition profile is highly sensitive to these variables; this can be seen in Fig. 65 by observing the effect of changing only a single equipment configuration variable. Explicitly modeling the impact of these binary variables would be possible in theory; however, this would neglect all prior system knowledge, and would require significantly more training data to learn the complex, non-linear relationship between G and T. Instead, we incorporate an existing physics-based solver to solve for a set of intermediate, spatially changing variables, S, which are a function of G. This provides a set of basis variables that have a simpler relationship to the deposition profile, and allows us to train our model with less data. It is important to note that while these intermediate variables, $S(x, y)$, may have a simpler relationship to the deposition thickness, they are multi-dimensional and spatially varying, and thus the inclusion of the physics-based solver leads to an increase in the total number of features. While the

dimensions of these variables depend on the choice of discretization size, in practice there are often over 10,000 elements in S.

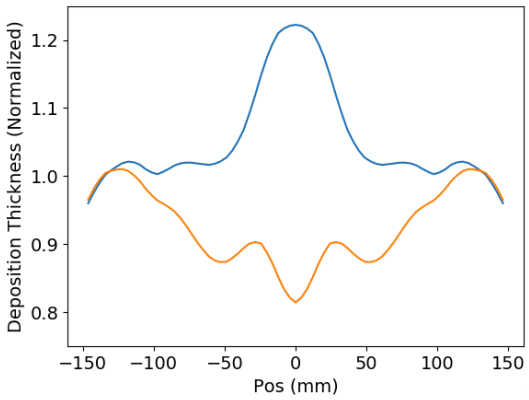


Fig. 65: Experimental change in deposition resulting from a single changed equipment geometry variable.

This problem of excessive features is compounded by the fact that we not only consider the direct outputs of the physics-based solver, but we also consider statistics and variants of these variables as additional potential features of the GP model. These statistics include the variable gradients, products, wafer means, wafer standard deviations, normalizations, and subsequent combinations of these operations. While these statistics contain no more information than unmodified spatially changing variables, it may be possible to choose the model inputs such that there is either a simpler input-output relation for the GP model to learn, or to reduce the larger number of variables to a few key statistics. The incorporation of the physics-based solver and these statistics both seek to provide the GP model with the simplest choice of input variables so that less training data is needed to model the underlying process. We will later discuss how the most effective subset of these statistics can be chosen.

5.3.2 Radial Binning

A key challenge in using the GP framework for this task is how to structure the GP model inputs. Representing the parameters of our system with as few variables as possible simplifies the function our GP model must learn, and thus reduces the amount of training data required. As previously described, the spatially changing (x, y) intermediate variables and their statistics, S, are solved for using a physics-based solver. While this allows us to incorporate some prior knowledge, it still does not simplify our model, as we

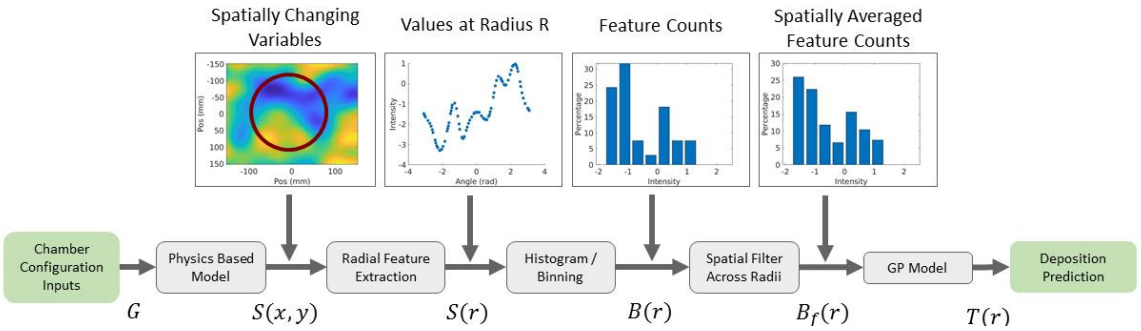


Fig. 66: Complete model flow. Chamber equipment configuration variables G are fed into a physics-based model which calculates the spatially changing variables, S. These are binned into histograms, and these bins are filtered across nearby radii to estimate the effects of sputtering. Finally, these filtered bins are used as inputs to the GP model.

have expanded our 100 model inputs to a spatially varying multivariable input, which can easy have tens of thousands of terms.

To simplify our model inputs, we make some assumptions. First, we assume that only values of S near radius r will impact the deposition thickness at that radius, i.e., $T(r) \approx f\big(S(r)\big)$ [183]. Second, we assume that instead of knowing the exact value of S at each radius, in reducing from $S(x,y)$ to radial $S(r)$ we only need to know an approximate distribution of these variables across the different (x,y) locations corresponding to each radius r.

With these assumptions in mind, we then simplify these spatially changing variables into a more compact form for our GP model input. For each modeled thickness, $T(r)$, we bin the values of $S(r)$ into a histogram, and use the normalized bin counts as the GP model input, $B(r)$, i.e., instead of using all values of S as our model inputs for $T(r)$, we use an approximate, discrete distribution of the values of $S(r)$. The intuition behind this feature reduction approach is that the intensity of the outputs of the physics-based model, $S(r)$, is believed to be correlated to how many ions strike the target at each location, and thus informs how much sputtering occurs at each location. Therefore, by creating a distribution of these variables at each radius, we can estimate how much sputtering occurs at each radius, and finally the deposition thickness at each radius.

This feature reduction technique has many properties which we desire. Primarily, it reduces the enormous number of total features to a feasible size. This is achieved both by reducing a large number of variables to a distribution, but also by only using values of S at radius r for predicting $T(r)$. In our testing, this reduces tens of thousands of features to approximately twenty key features in most cases. Secondly, this feature reduction

technique is rotation invariant: spatial rotation of the (x, y) spatial variables should result in a rotated output. However, because the deposition is radially symmetric, a rotation of S should result in the same predicted thickness. This property is achieved because the feature inputs, the distribution of S at each radius, are unaffected by a rotation, and thus predict the same output when a rotation is made.

5.3.3 Feature Filtering

Next, we adjust our model to account for spatial interaction effects. As previously described, our feature reduction technique does not take into account values of S at radii other than that being predicted. This is unrealistic, as sputtering from a point source will deposit material onto nearby points and thus the values of S at one radius can affect the deposition thickness at another. Therefore, rather than using only the distribution of $S(r)$, $B(r)$, to predict $T(r)$, we instead use a distribution of $S(r')$ for all values of r' close to r, i.e., for $r - \delta < r' < r + \delta$. This distribution of S across nearby values of radii, $B_f(r)$ is achieved using a spatial filter. Applying a spatial filter to the features creates a distribution of S not only at radius r, but also for nearby radii, and weights each feature in this distribution based on its spatial distance from the radius of interest.

To create this distribution, we must make two decisions: the shape of the filter, and where in the model to apply it. The spatial filter can occur either before or after the binning of S and each will result in a different set of features passed into the GP model. Applying the filter after binning results in a one dimensional convolution, as the binning process compresses $S(x, y)$ into $S(r)$, and the filter is then applied across the radius, while filtering before binning applies a two dimensional convolution in both x and y. We considered both before and after the radial binning with various filters, including Gaussian, exponential, line-of-sight cosine, raised cosine, and square. In practice, we found that the best performing filter occurs after radial binning, and takes the form of a raised cosine distribution with a length scale l_f.

5.3.4 Illustrative Example

In this section, we present a simple example that illustrates the binning and feature filtering steps in our feature engineering pipeline (Fig. 66). For simplicity, we consider the case where our physics-based pre-processor outputs a single spatially changing variable, $S(x, y)$, and this variable is only a function of the radial position (Fig. 67).

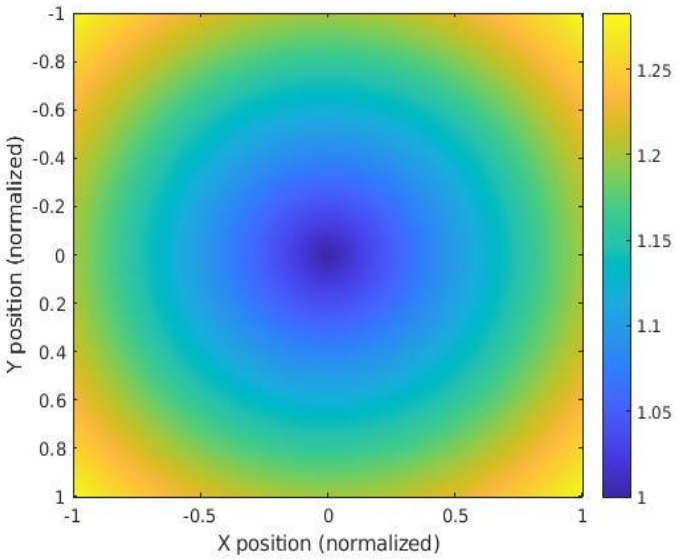


Fig. 67: Example spatially changing variable as a function of wafer position, $S(x, y)$.

As each set of features represents the spatial variables seen at one radius, the first step is to collect the values of all spatially varying variables at that radius. Below, we plot the spatial locations used when creating features for $r = 0.5$ (Fig. 68).

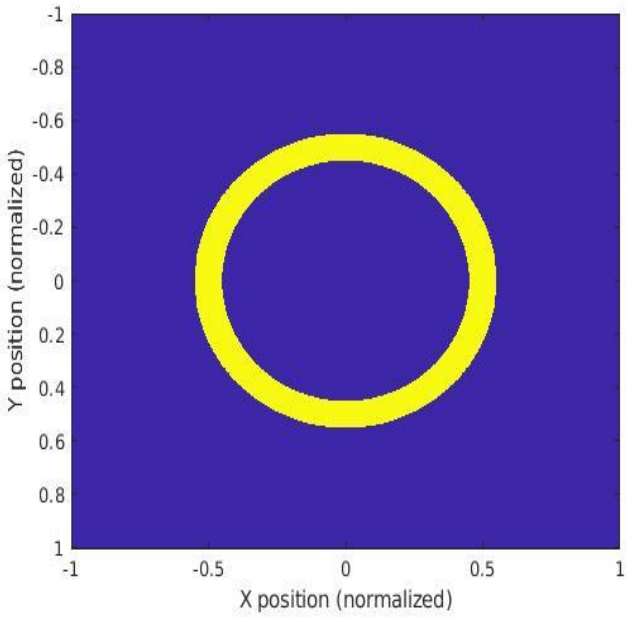


Fig. 68: Points used when determining features for $r = 0.5$. Yellow indicates selected points, and blue indicates omitted.

The values of S at these locations are collected and converted into a histogram, which determines how these spatial variables are distributed at the radius of interest.

Importantly, the edges of these bins are determined by the full range of training data, thus these bins are fixed both between chamber configurations, as well as between radii. As the example data we synthesize is only a function of chamber radius, one bin in the histogram contains all values of S at that radius, while the others are empty (Fig. 69).

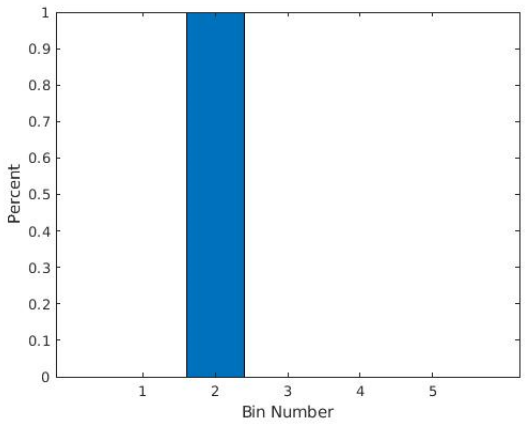


Fig. 69: Histogram showing distributions of data at r = 0.5.

Additionally, we can view these histograms as a function of wafer position (Fig. 70). Since each radius contains only one value of S, we see one bin containing all data for each radius, while the remainder are empty.

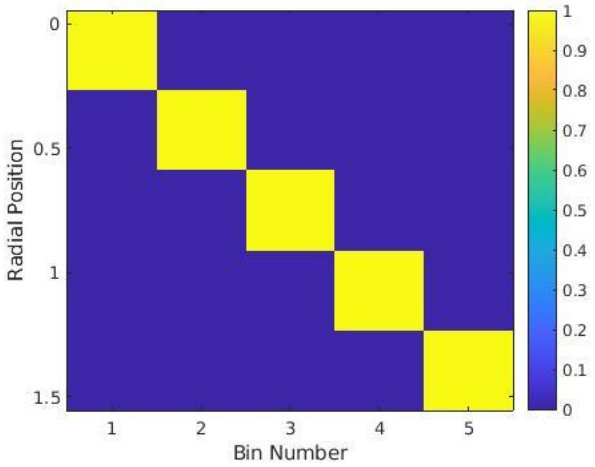


Fig. 70: Bin percentages as a function of radial position.

The pre-processing step before passing these features into the GP model consists of applying a spatial filter. Because sputtering from a point source deposits material into a nearby area, the values of S at one radius will affect nearby deposition thicknesses. Therefore, the bin percentages are filtered across radii before being passed to the GP

model. These filtered bin percentages can again be viewed as a function of wafer position (Fig. 71) or for a specific radius (Fig. 72).

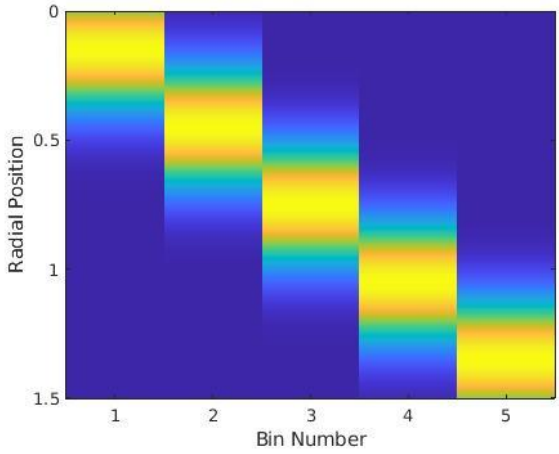


Fig. 71: Filtered bin percentages as a function of wafer position.

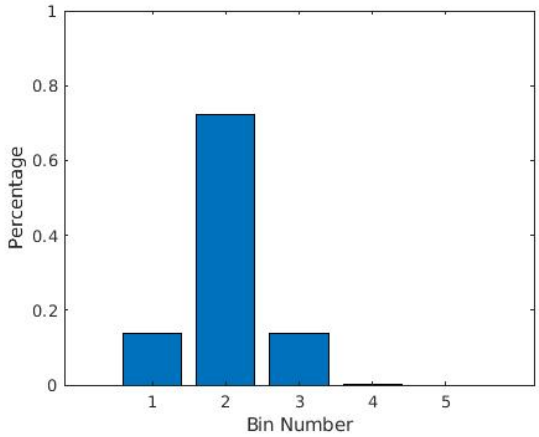


Fig. 72: Bin percentages after filtering at $r = 0.5$.

After preprocessing, these bin percentages can then be concatenated for each component of S. For example, if there are two spatial variables, features 1 through 5 correspond to the bins for the first spatial variable, and features 6 through 10 correspond to the bins for the second spatial variable. For each radius and observed deposition thickness, these concatenated features are then used for the inputs to the GP model, compressing a large number of spatial inputs down to a representative distribution.

5.3.5 Model Fitting

Fitting and evaluating the model is a non-trivial task, and in this section, we describe the procedure. In this case, there are a significant number of model hyper-parameters; thus we use a traditional training, validation, testing split. We train the GP model on the training dataset, then select the best performing hyper-parameters based on the ability to predict the validation dataset, and finally estimate the real-world predictive power using the held-out testing dataset.

The model hyper-parameters are crucial in properly fitting the model and include bounds on the GP model length scales, l_p, GP model noise, σ_n, filter length scale, l_f, and the choice of which potential features to include. These hyper parameters are selected by repeatedly optimizing each sequentially until no improvement on the validation set can be found. The objective function used to select the best performing hyper-parameters is the leave-one-out cross validation mean squared error (CVMSE), and is implemented the same way as was described in Section 5.3.

When optimizing each hyper-parameter, the scalar hyper-parameters, i.e., all but the choice of included features, are each sequentially optimized using a one dimensional grid search, while the choice of included features are fit using a stepwise method. In the latter, the potential features, i.e., the histograms of S and its resulting statistics, are iterated through, and for each potential feature, we determine if adding it to the list of included features (if not already included), or removing the potential feature (if already included) increases or decreases the cross validation MSE. Changes which decrease the cross validation MSE are kept, while those which increase it are reverted. In practice, we often select the distributions of two to three key features; however, this varies based on the training and validation sets.

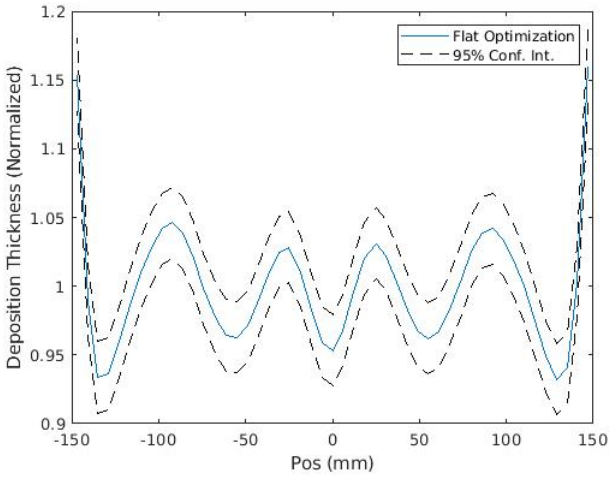
5.4 System Optimization

Once a model for the system (either recipe parameters model or an equipment configuration parameters model) has been fit, it can be used to optimize the parameters in order to meet some criteria. Advantageously, the output of the GP model is a probability distribution, enabling us to maximize the likelihood of a desired profile under the predicted distribution. This choice of objective function takes into account both how close our maximum likelihood prediction is to the desired profile, and how confident we are in this prediction.

Here, we consider two cases separately: the first is to optimize the recipe parameters, p, to achieve a desired sputtered film thickness profile; and the second is to achieve desired thickness profile uniformity by optimizing the equipment configuration chamber parameters, G. The geometric chamber parameters are highly influential in changing or improving higher-frequency variations in film thickness, while the scalar recipe parameters are less so. The first case involving process recipe parameters is simpler and, within the limitations on uniformity achievable by recipe tuning, generally does not require iterative tuning. In contrast, the second case involving equipment configuration parameters is more challenging, and places a high demand on a small number of iterative tunes in order to achieve acceptable uniformity.

5.4.1 Process Recipe Parameter Optimization

Optimizing the process recipe parameters is a relatively easy task, as the inputs are continuous, are limited in number (there are five in our case), and their effect on the objective function is smooth and contains few, if any, local minima. For this reason, we use a simple gradient descent algorithm, combined with a finite difference method to estimate the gradients, in order to optimize the scalar process parameters. We test this method using a curved profile for our desired deposition thickness. As can be seen in Fig. 73, the low frequency component of the predicted profile is nearly identical to the desired profile; however, the higher frequency ripples are still present. This is expected because the process recipe parameters can only control the low frequency components, and have little effect on the spatial ripples of the thickness profile. The resulting non-uniformity between the predicted and desired profiles is 4.36% in this example.



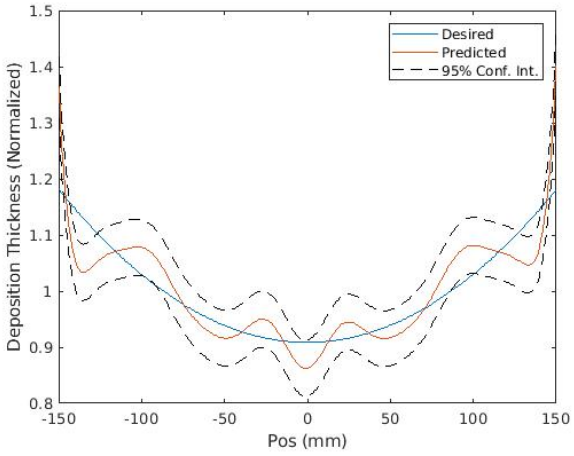


Fig. 73: Resulting profiles for the flat (top) and curved (bottom) desired responses predicted by tuning the simple process recipe parameters.

5.4.2 Chamber Geometry Optimization

Unlike the process recipe parameters which can effectively be modeled with a small amount of training data, the large number of chamber geometry parameters, combined with the much more non-linear and non-convex input-output relation, makes it impossible to accurately predict the profiles for all combinations of these inputs. This makes finding the true global optimum infeasible. However, due to the large number of chamber variables, there are many different configurations that will meet our improved uniformity requirements, and thus we can afford to stay in a local minimum if it satisfies our specifications.

Because modeling the geometric chamber parameters requires a substantial amount of training data, the processes of collecting training data, building a model, and finding an optimal set of conditions is not an isolated, one-time procedure, but rather is a continuous, iterative tuning cycle. This iterative procedure begins with a small amount of training data (in our case, five different chamber configurations), then the steps of modeling, optimizing, and collecting more data repeats until an acceptable configuration is found.

As the chamber configuration variables are discrete, binary, and have a highly non-linear relationship to the output, we use an incremental local search method to optimize these variables at each iteration. After collecting data and updating the model following a previous optimization, we select the best known configuration (in our case the configuration which results in an experimental profile with the minimum mean squared error between it and the desired profile), and use this as our start point for the next

optimization. Because each spatial variable is binary, we then create a list of all configurations that are at most two equipment parameter changes away from our current best known configuration, i.e., they have a Hamming distance of two from our current best configuration. This limits the number of configurations that must be considered by the model during the optimization process to a feasible number, and avoids too rapid movement into highly uncertain areas of the input space. After creating this list of (in our case 10,000) potential chamber configurations, we then use our model to predict the resulting profiles. We select the configuration that has the lowest cost, i.e., the highest likelihood of our desired profile. This chamber configuration is then set, the deposition is performed and the thickness profile measured. The new data is used to update the model and the process is repeated until an acceptable configuration has been found.

While we have found this method sufficient for converging to a desired profile, this approach differs from traditional Bayesian Optimization (BO), in an important way. Here, our acquisition function, the likelihood that the input configuration leads to a desired profile, only incorporates exploitation and does not incorporate exploration, i.e., it only attempts to select optimal configurations given its current belief, and does not attempt to gather additional information. There are two reasons for this. First, we primarily evaluate the performance of our method using historical data. For this reason, collecting new data with the goal of exploration is not possible, and we must rely solely on evaluating the purely predictive power of our model. Secondly, as the input space to our model is high dimensional, it is infeasible to model or explore the entire space; however, many sufficient solutions exist, and it is likely to find one of these solutions in a small region of this input space. Therefore, we suspect acquiring new data points in one small subsection of this space via exploitation is often more beneficial that exploring new regions. However, for alternative applications that utilize similar approaches, incorporating exploration into the acquisition function may improve overall model performance, and can reduce the number of iterations required to converge to a desired solution.

5.5 Comparison Methods

In this section, we briefly present methods that we compare our proposed Gaussian Process approach to. These two methods include polynomial models, gradient boosted regression trees (GBRTs), multivariate adaptive regression splines (MARS), convolutional neural networks (CNNs), dense neural networks (DNNs), and radial basis

function neural networks (RBFNNs). These methods represent common empirical modeling techniques widely used in semiconductor applications and in machine learning as a whole. These methods also represent different extremes of the bias-variance tradeoff. For example, polynomial models are simple to fit with little data, but have limited modeling power, while deep learning methods require significantly more data, but can model a much wider range of systems.

Because many of the comparison models also benefit from proper feature selection and engineering, we utilize the same feature engineering pipeline previously described in order to create inputs to the non-deep learning methods, i.e., the polynomial model, GBR, and MARS models. This compressed feature representation reduces the number of inputs to a manageable quantity, helping to prevent overfitting, while also providing terms which are more directly related to the modelled outputs. This also allows us to determine how effective the models are when the feature pre-processing stages are the same in both cases.

5.5.1 Polynomial Modeling

Modeling the effects of complex equipment configuration chamber parameters using a polynomial model presents the same challenges as does using a GP model for the same task. In particular, feature selection and engineering is important in both cases. As discussed in Section 5.4, there are many different ways to select inputs which can be used for either the GP or polynomial model. Using the raw chamber parameters alone results in over 100 inputs to the polynomial model. This is often significantly higher than the number of training sets available, making overfitting highly likely. Additionally, as there is a highly non-linear relationship between the original chamber parameter inputs and the deposition thickness, this would require many terms in the polynomial model, making overfitting even more likely.

In addition to sharing the feature selection process, the polynomial implementation also utilizes the same hyper-parameter selection technique. While the approaches are the same, the set of hyper-parameters are different. Both select which features are relevant to the model using the CVMSE; however, the polynomial approach will select the polynomial order instead of the GP specific hyper-parameters. As we will describe in Section 5.6.1, many training-testing ensembles are used to evaluate the performance of these models. Because hyper-parameters are chosen specifically for each ensemble, there

is not one set of parameters used; however, in most cases second order polynomials with interactions are created through hyper-parameter selection.

5.5.2 Gradient Boosted Regression Tree Modeling

A second comparison method we evaluate is gradient boosted regression trees (GBRTs) [184], [185]. This is an ensemble method that uses a collection of weak learners, in this case decision tree regressors, to create a more robust and expressive model.

Here, we apply gradient boosted regression tree models in a similar method to our application of polynomial and Gaussian Process models. Again, we use the same feature pre-processing pipeline and hyper-parameter tuning method as in both the GP and polynomial cases. While the hyper-parameter tuning methods are the same, the tuned parameters are not identical. Just as before, we select which features are used as inputs to the model; however, we optimize the depth of each tree and number of trees instead of the polynomial and GP specific hyper-parameters. Again, these hyper-parameters change for each ensemble of training data; however, a representative well-fit model uses 60 trees of maximum depth 3. Additionally, we use default values for the number of samples needed to split a tree (2), as well as the learning rate during gradient descent (. 1).

5.5.3 Multivariate Adaptive Regression Spline

A third comparison method we evaluate is multivariate adaptive regression spline (MARS) models [186]. These regression models create a piecewise approximation for an underlying function. Each piece of the approximation is a linear function; however, higher order terms are commonly added to the feature basis to improve the model performance.

We test this method using a similar approach to the GP, polynomial, and GBRT methods. We use the same feature pre-processing pipeline, and tune the hyper-parameters using the same selection method. In addition to selecting which features are passed into the MARS method, we also tune the maximum number of terms in each piecewise approximation, the maximum polynomial order of these approximations, as well as a penalty parameter that controls the smoothness of these approximations. For the resulting MARS fits, a representative model uses third order polynomials, a maximum of 10 terms in these approximations, as well as a regularization penalty of 3.09.

5.5.4 Neural Network Modeling

The final set of comparison methods we consider use neural networks. In total, we implement three unique architectures as comparison methods: dense neural networks, radial basis function networks, and convolutional neural networks. Because neural networks are prone to overfit to small amounts of training data, we impart some prior knowledge into the architecture in order to help alleviate this.

All three deep learning approaches again use the physics-based solver to perform feature pre-processing. Here, the spatially changing variables directly modeled by the physics-based solver are used as inputs to the neural networks. The values of these features are stored in a 3D tensor where the first two dimensions correspond to the spatial (x, y) position on the wafer plane, and the third corresponds to the variable type and is the channel dimension for the networks.

While each network uses the same preprocessing, the networks themselves vary. The most traditional of the three is a dense fully connected multilayer perceptron neural network. Here, the preprocessed features are first flattened and then are fed into three consecutive dense layers that each use rectifying linear unit (ReLU) activation functions. The number of nodes in these layers decrease logarithmically from the starting feature size to the output size. For our experiments, these layers have 109375, 1403, and 18 elements; these are the only elements in the baseline dense neural network. The network is trained in order to minimize the mean squared error of the training predictions using the ADAM optimizer with default parameters, except that a learning rate of $2 \cdot 10^{-2}$ is used instead of the default value. Dropout is not used.

A second deep learning variation we consider are radial basis function networks [187], [188]. These are similar to traditional dense neural networks; however, their activation function is a radial basis function, instead of a traditional rectifying linear unit (ReLU). A commonly seen benefit of these networks is their increased robustness [189], which is particularly advantageous for our data limited scenario. Our architecture is similar to the dense network, except the activation functions of the first two layers are replaced with radial basis functions, instead of the traditional ReLU. The final layer uses a fully connected linear layer with no activation function, in order to properly scale the output. The same loss function and training parameters as for the DNN are used.

The final deep learning variation we consider is a convolutional neural network, that also incorporates our physical assumptions about the process. As we previously

described, CNNs are commonly used in tasks with spatially related variables, such as our those produced from our physics-based pre-processor. Again, these spatial variables are the inputs to our network, and pass through six 2D filter layers, which each have 7, 14, 28, 14, 7, and finally 1 feature channels. The number of elements per channel is held constant, as these elements represent the spatial locations inside of the chamber, and the final layer represents the pointwise deposition at each (x, y) point on the (non-rotating) wafer. Following this, a custom layer applies a radial averaging that maps the 2D pointwise deposition map to a 1D vector. The outputs in the 1D vector are the average of the values on the 2D map that have approximately the same radial position. This structure allows for the convolutional layers to learn only the mapping between the chamber variables and deposition rate, while the radial averaging accounts for the rotation of the wafer during deposition. This substantially constraints this specific CNN structure, helping to fit with the limited available data.

For all of these networks, we also impart some knowledge into the training process itself. Here, we not only train with the direct outputs of the physics-based simulator, but also train it with rotated versions of these inputs. Because the wafer rotates during deposition, the absolute chamber orientation is irrelevant to the final deposition profile. This allows us to add additional examples to the training data set which helps reduce overfitting.

5.6 Results

In this section we present the performance of the proposed GP model and the associated optimization method, as well as comparisons to the polynomial, gradient boosted regression tree, multivariate spline, and deep learning methods, applied to the equipment configuration optimization problem. As discussed at the start of this chapter, the key criteria of tuning procedure success is the number of tunes required to find a profile that meets the desired specifications. While this will ultimately be the key performance metric, we will first look at the predictive capabilities of each modeling approach. Then we will evaluate the performance of the proposed optimizer, and finally evaluate the combined tuning performance.

5.6.1 Model Performance

Here, we present the predictive performance of the proposed model, in addition to the comparison methods. We use a limited dataset (40 chamber configurations) to evaluate the model performance. In each case, we train, validate, and test the model using a cross-testing procedure. Here, we divide the total data into training and testing sets, use all training data to select the best hyper-parameters via cross-validation, and then build the model on the entire training set. Finally, the model is used to predict the testing dataset, its performance is evaluated, and the process is repeated with a different training-testing split. This is done for training datasets of size 5, 10, 15, 20, 25, and 30.

Using data normalized to their means, the average predictive MAE (Table VII), RMSE (Table VIII), and R^2 (Table IX) values are shown for the proposed and comparisons methods as a function of training set size. For each training set size, the best predictive performance is bolded. Additionally, we also include these metrics when only considering the 25% most confident GP predictions. Here, the metrics are computed using the prediction, T_{pred}, the experimental value T_{exp}, and the set of all experimental thickness data, T_{total}, and these are later averaged over all training-testing splits:

$$R^2 = 1 - \frac{\sum\left(T_{exp} - T_{pred}\right)^2}{var(T_{total})} \tag{79}$$

$$RMSE = \sqrt{Mean\left(\left(\frac{T_{exp} - T_{pred}}{mean(T_{total})}\right)^2\right)} \tag{80}$$

$$MAE = Mean\left(\left|\frac{T_{exp} - T_{pred}}{mean(T_{total})}\right|\right). \tag{81}$$

We observe that all accuracies increase as more data is collected, confirming that the lack of training data can be a substantial impediment. Interestingly, many R^2 values are negative, corresponding to cases where the prediction is worse than a horizontal line through the mean, arising due to overfitting in many cases. We also note particularly poor performance for the MARS and polynomial models. This is likely due to the fact that both models utilize polynomial functions, which are particularly poor at extrapolating data. Because our input data is high dimensional and the training sizes are small, such extrapolation is likely necessary.

TABLE VII
NORMALIZED MEAN ABSOLUTE ERRORS AS A FUNCTION OF NUMBER OF TRAINING WAFERS.

Method	5	10	15	20	25	30
GP	0.093	0.098	0.075	0.064	0.055	0.052
GP- High Confidence	**0.066**	**0.079**	**0.052**	**0.028**	**0.045**	**0.025**
Polynomial	0.122	0.210	0.094	0.095	0.068	0.063
GBRT	0.105	0.085	0.073	0.074	0.073	0.061
MARS	0.128	0.179	0.244	0.084	0.111	0.074
CNN	0.091	0.090	0.078	0.058	0.050	0.045
DNN	0.101	0.083	0.080	0.069	0.066	0.064
RBFNN	0.098	0.093	0.086	0.085	0.084	0.086

TABLE VIII
NORMALIZED ROOT MEAN SQUARED ERRORS AS A FUNCTION OF NUMBER OF TRAINING WAFERS.

Method	5	10	15	20	25	30
GP	0.117	0.126	0.111	0.093	0.084	0.081
GP- High Confidence	**0.084**	0.104	**0.068**	**0.044**	0.064	**0.037**
Polynomial	0.153	0.360	0.132	0.128	0.085	0.090
GBRT	0.128	0.111	0.096	0.096	0.096	0.082
MARS	0.171	0.323	1.850	0.120	0.429	0.126
CNN	0.098	0.100	0.905	0.066	**0.056**	0.050
DNN	0.113	**0.092**	0.089	0.080	0.074	0.072
RBFNN	0.110	0.104	0.096	0.094	0.093	0.096

TABLE IX
NORMALIZED R^2 AS A FUNCTION OF NUMBER OF TRAINING WAFERS.

Method	5	10	15	20	25	30
GP	-0.73	-1.02	-0.57	-0.12	0.09	0.18
GP- High Confidence	**0.11**	-0.36	**0.41**	**0.75**	0.48	**0.83**
Polynomial	-1.97	-15.4	-1.22	-1.08	0.09	-0.03
GBRT	-1.09	-0.56	-0.17	-0.18	-0.18	0.14
MARS	-2.74	-12.3	-434	-0.82	-22.4	-1.00
CNN	-0.17	-0.20	0.016	0.471	**0.618**	0.704
DNN	-0.53	**-0.03**	0.042	0.235	0.342	0.385
RBFNN	-0.44	-0.30	-0.11	-0.07	-0.03	-0.12

In almost all cases, the high-confidence GP model performs the best. Interestingly, the overall GP model does not generally have the best average predictive performance across the other comparisons; however, we will later show that it still performs better in an optimization setting than the other methods. This is because in these predictive results,

the average predictions for the GP model are skewed by a relatively small number of highly inaccurate predictions. This can be seen when comparing the RMSE of the models to their MAE, the latter of which weights all error equally, and here the GP outperforms all comparisons except the CNN. While the GP results are skewed by small numbers of inaccurate predictions, the GP model also tends to have a larger number of highly accurate predictions (Fig. 74). These highly accurate predictions are key to achieving high uniformity in tuning, because the final uniformity will generally be limited by the most accurate prediction (i.e., a prediction that results in a tuning run that satisfies the goal or specification). Therefore, the *average* accuracy is not as important as the ability of the model to be more often highly accurate. Even though the mean prediction accuracy of the GP model is lower than the CNN, the ability to predict extremely accurately in more cases makes it more likely to find a single configuration that meets the desired specifications.

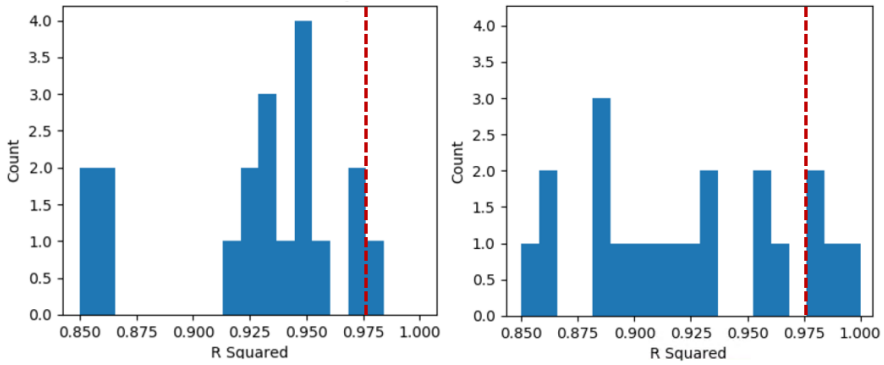


Fig. 74: Distribution of the most accurate predictive R^2 values for CNN (left) and GP (right) models using 20 training wafers. Note the difference in R^2 values greater than 0.975 (division marked in red).

Additionally, it is important to note that the GP model not only produces its maximum likelihood prediction for the deposition profile, but also provides its confidence in the form of the predicted distribution standard deviation. This is critical for the performance of our tuning approach, as a key piece of the overall approach is to identify which predictions (of the 10,000 evaluated at each iteration) are both similar to the desired profile, and we have high confidence in. For this reason, we should also consider the GP model confidences when evaluating its accuracy (Fig. 75). The confidence provided by the GP model is a key advantage over the deterministic comparison methods, as it greatly increases the effective prediction accuracy, and thus the likelihood of meeting the desired

specifications on a new tune. Therefore, we also provide the high confidence prediction accuracies in Table VII through Table IX in addition to the naïve estimate.

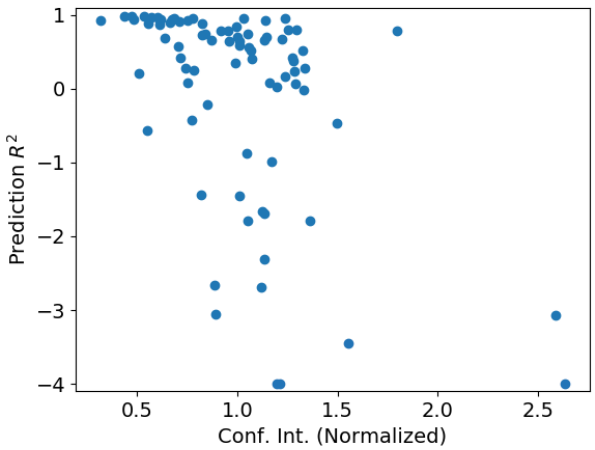


Fig. 75: Predictive accuracy vs. model 95% confidence interval width when trained with 20 wafers, for the GP model.

These results highlight a key theme of this thesis. Probabilistic methods are often flexible enough to model highly complex processes, while still providing protection from overfitting in data limited cases. The greatest concentration of highly accurate predictions demonstrates that GP models are capable of modeling with high accuracy; however, it is infeasible to do this in all cases, as data is limited and the underlying process is highly complex. However, the GP model is able to recognize this, and only performs poorly when it has low confidence in the predictions. This demonstrates a secondary benefit of probabilistic methods, that their predictive distributions can often be used to prevent overfitting by avoiding low confidence areas. As we will see in the next section, both of these aspects enable GP models to outperform alternative modeling methods.

5.6.2 Optimizer Performance

Here, we evaluate the performance of the optimizer alone, by estimating how close the optimized profile is to the desired profile. It is important to note that this evaluation does not depend on the accuracy of the model, but rather on the performance of the optimizer, i.e., we are determining how close we can get to the desired profile in one iteration, assuming the model is ideal. Here, we use a curved profile for our desired profile, and use the non-uniformity between the predicted and desired thicknesses as our metric, per Eq. 82. In this example, we achieve a non-uniformity of 1.03% as seen in Fig.

76, and note that this is substantially better than the previous non-uniformity using only process recipe parameter optimization.

$$NU = \frac{std(T_{des} - T_{pred})}{mean(T_{des})}. \tag{82}$$

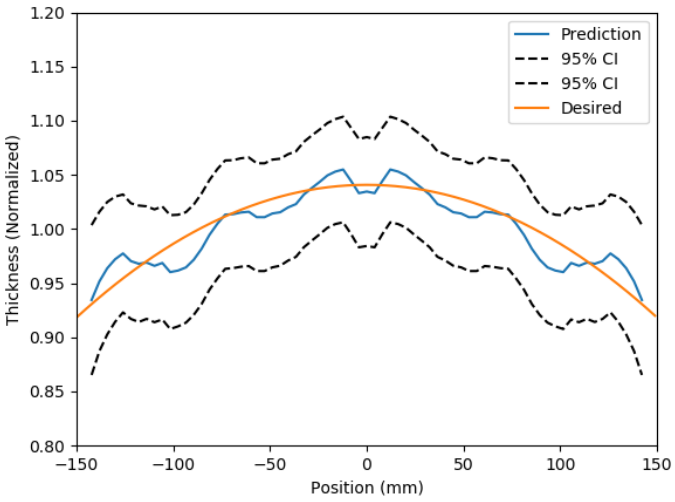


Fig. 76: Optimization results for curved desired profile.

5.6.3 Combined Model and Optimizer Performance

Finally, we estimate the performance of the overall iterative tuning approach combining the equipment configuration model and optimizer using historical data. For our application, we desire a total non-uniformity below a specific threshold, NU_{des}. We evaluate the performance of the iterative tuning approach by how quickly we expect to reach this specific goal. Overall, the probability of reaching the desired profile by iteration I, $P(NU_I < NU_{des})$, is the complement of not reaching the desired profile in all previous iterations.

$$P(NU_I < NU_{des}) = 1 - \prod_{i=1}^{I} P(NU_i > NU_{des}). \tag{83}$$

Now, we estimate the probability of achieving the uniformity goal in each individual iteration. Here, we break down the uniformity requirement into the optimizer and model components. The total squared error between the experimental film thickness, T_{exp}, and desired thicknesses, T_{des}, comes from both the predictive model and the optimizer; thus the uniformity requirement can be decomposed into two mean square errors, $MSE_{optimizer}$ and $MSE_{model,i}$:

$$NU_i^2 = \frac{var\left(T_{exp} - T_{des}\right)}{mean(T_{des})^2} \geq \frac{MSE_{optimizer} + MSE_{model,i}}{mean(T_{des})^2}. \quad (84)$$

As we have already estimated the MSE for the optimizer, we now determine the model accuracy at each iteration using past experimental data. In the previous section, we determined a distribution of the model accuracy for 5, 10, 15, 20, and 25 training wafers. Now, we estimate the probability of meeting the uniformity goal at each individual iteration as the percentage of testing wafers that we predict accurately enough to meet the combined non-uniformity metric from Eq. 84. Here, we assume that the first five wafers are strictly for training, i.e., we will not achieve the desired profile in the first five wafers, and use the accuracy of iteration $5N$, where N is an integer, to estimate the accuracy of the following four wafers. Combining this with Eq. 83, we estimate the probability of achieving the desired uniformity by iteration I for a variety of uniformity requirements.

The results of this analysis can be seen in Fig. 77 through Fig. 83 using the proposed GP methodology, as well as for the comparison methods. We see that even for extremely tight uniformity requirements, we are likely to converge to the desired profile in a feasible number of iterations. In almost all tuning cases, the GP model outperforms the comparison methods on this critical metric, and these improvements are greater when targeting tighter uniformity requirements, confirming that the GP method is well suited for optimizing these sputtering deposition processes. Even though its average predictive R^2 is not always superior, the greater number of highly accurate predictions enable optimization with the GP model to find a single configuration that meets uniformity specifications before the other methods.

It is important to note that in practice we expect significantly fewer wafers to be required in order to converge to a desired profile using the GP model compared to the estimations presented below. This is because we use all chamber configurations to estimate the model accuracy at each iteration, not only those with tight confidence intervals. As previously explained, we select new chamber configurations based on their likelihood of achieving the desired profiles, so configurations with tight confidence intervals are significantly more likely to be chosen than those with low confidence intervals. In our calculations, our model error estimates use all past data. Because the errors are much higher when the confidence interval is loose, as can be seen in Fig. 75, our model accuracy is likely underestimated, and we expect the true convergence rate to

be even quicker for the GP model. Therefore, we also present convergence results for the GP model using only the 25% most confident predictions. This more accurately estimates the true convergence rate, as the model errors take into account the prediction confidence, as is done in the selection of new configurations. As expected, this more realistic estimate outperforms the naïve convergence estimate of the GP model, which already outperforms the comparison methods. This suggests that the proposed GP approach is well suited to tuning new processes rapidly, as it has high accuracy predictions, and also considers the prediction confidence.

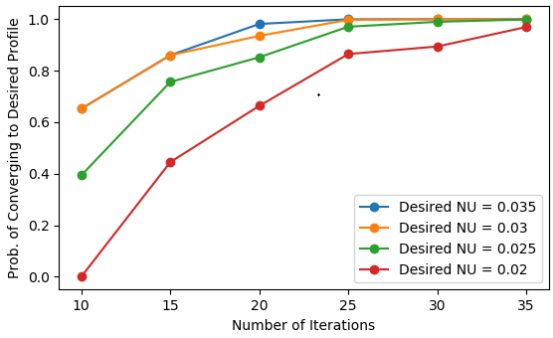


Fig. 77: Probability of converging to desired non-uniformity vs. number of iterations for GP model.

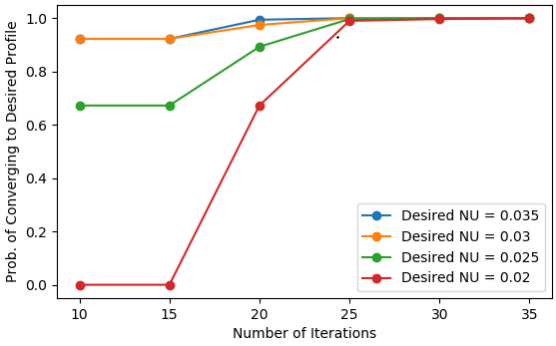


Fig. 78: Probability of converging to desired non-uniformity vs. number of iterations for GP model using high confidence predictions.

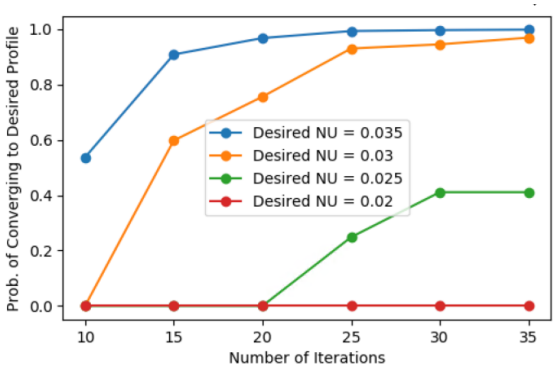


Fig. 79: Probability of converging to desired non-uniformity vs. number of iterations for polynomial model.

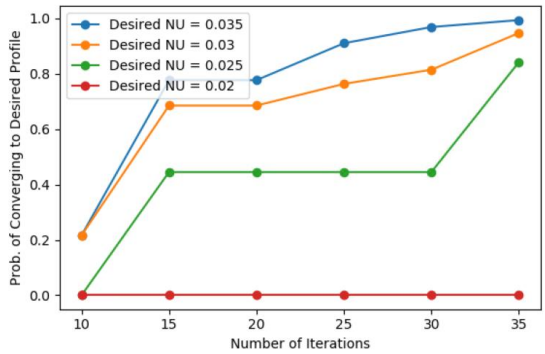


Fig. 80: Probability of converging to desired non-uniformity vs. number of iterations for MARS model.

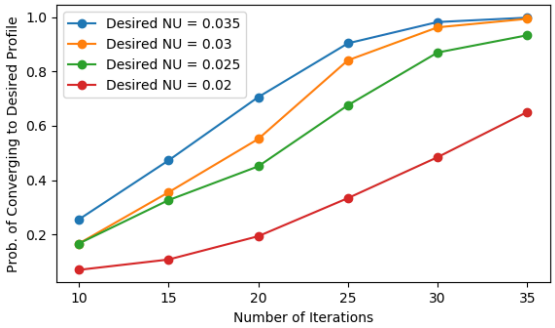


Fig. 81: Probability of converging to desired non-uniformity vs. number of iterations for CNN model.

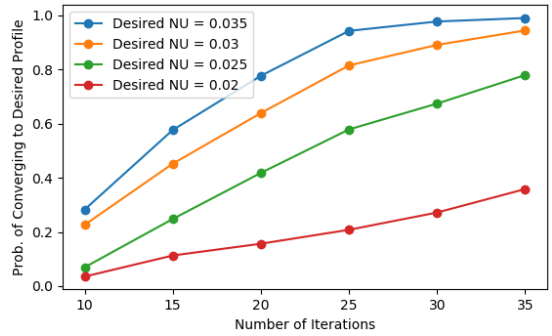


Fig. 82: Probability of converging to desired non-uniformity vs. number of iterations for DNN model.

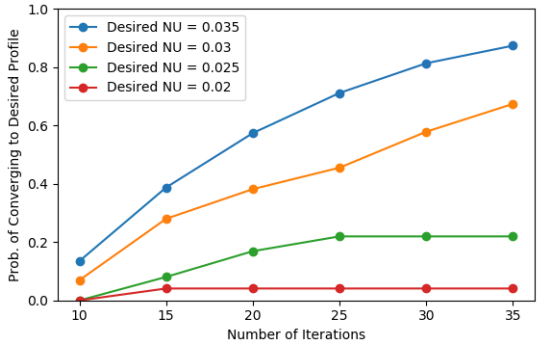


Fig. 83: Probability of converging to desired non-uniformity vs. number of iterations for RBFNN model.

5.7 Conclusions and Future Work

In this chapter, we discuss an empirical method to explore, model, and optimize sputtering deposition processes using a Gaussian Process framework with the goal of meeting a desired thickness profile. We first apply the GP framework to the modeling of film thickness uniformity based on simple scalar process recipe parameters such as pressure and rf power. While the recipe parameters are easier to model and optimize, their impact on the final profile is limited to long range or low frequency spatial uniformity effects. The GP framework is further extended and applied to the more challenging problem of modeling and optimizing complex equipment configuration parameters, in order to achieve greater uniformity. For modeling and optimization complex functions with many inputs such as this, our iterative tuning approach is able to converge in a small number of iterations. The GP model with iterative tuning is shown to achieve a desired profile in fewer iterations compared to using conventional methods, including polynomial gradient boosted regression, multivariate spline, and deep learning

modeling methods. Additionally, we believe that in practice the required number of iterations will actually be lower than this estimate.

This case study highlights both of the key themes of the thesis. First, the incorporation of process knowledge is critical to performance, as optimizing the process in as few runs as possible is the key metric of success. By incorporating a physics-based solver and process specific feature pre-processing, we reduce a large number of inputs to a smaller number of features with less complex relationships to the output of interest. This allows us to model these simpler input-output relationships with less data, and converge to a sufficient solution more rapidly. Secondly, the benefit of probabilistic methods is again highlighted in this study. We demonstrate that the probabilistic GP approach is more likely to be highly accurate compared to alternative methods, again enabling faster convergence to a desired profile. Additionally, we also demonstrate that predictive distributions can be used to overcome insufficient data. As probabilistic methods often output predictive distributions, inputs with high variance outputs can be avoided, and we can select only high confidence, and high accuracy, inputs during optimization. This enables greater predictive accuracy and a faster convergence rate, demonstrating another key benefit of probabilistic methods.

While we have laid the foundation for an effective Gaussian Process machine learning framework for semiconductor fabrication, there is still more that can be done. Future work should seek to use the proposed method to rapidly tune a real deposition chamber, to confirm that the estimates from Section 5.6 are conservative. Secondly, the approach could be applied to other outputs besides thickness, such as film stress or resistivity, and to explore multi-objective optimization. While we have demonstrated that it is possible to tune a single output such as the thickness profile, the same could be done for multiple variables of interest simultaneously. Finally, future work could seek to combine the process recipe and equipment configuration models. Both use a GP-based model framework; combining the two would be limited primarily by the amount of required data. We believe that our Gaussian Process framework can be applied to modeling spatial process variations in a wide variety of cases beyond the sputtering deposition process presented here.

6 Conclusion and Future Work

In this chapter, we highlight the major contributions of this thesis, and discuss future work that may build on the work presented here.

6.1 Thesis Contribution

In this thesis, we study the use of probabilistic machine learning methods for semiconductor fabrication applications, in particular in data-limited scenarios. Semiconductor fabrication offers enormous opportunities for machine learning, as the continuous drive to reduce device sizes provides nearly endless opportunities to model, optimize, and monitor fabrication processes. However, available data is often limited, and machine learning methods must be chosen with this constraint in mind. We believe that probabilistic machine learning methods are particularly well suited for these data limited cases, and explore their uses. While exploring all fabrication processes, all goals, and all probabilistic methods is infeasible, we present four case studies which highlight common processes, applications, and probabilistic methods.

Throughout each of these case studies, two common themes are seen. The first is the natural regularization of these methods which helps prevent overfitting in data limited scenarios. Many methods in this thesis contain a prior belief that acts as inherent regularization, and estimated observation noise also prevents overfitting in many of the methods. Secondly, we find that the incorporation of process, or domain specific, knowledge is critical to these data limited scenarios. In all cases, we incorporate some element of process knowledge, without which we would be unable to successfully apply our methods. These often come in the form of prior beliefs, data pre-processing, or specific model assumptions.

Our first case study explores the use of probabilistic methods to study and improve modeling performance in virtual metrology systems. Here, we make two primary contributions. First, we present a probabilistic framework that can be used to study virtual metrology systems and generate synthetic data. This framework contains two common sources of error, concept drift and observability errors, and data synthesized using it can be used to develop more robust virtual metrology methods. Our second

contribution are robust Bayesian models that adapt to concept drift, and provide inherent regularization to prevent overfitting to observability errors. This method performs better than traditional linear approaches, and highlights the main themes seen in this thesis. The presence of both a Bayesian prior and observation noise estimates help prevent overfitting, and the incorporation of assumed concept drift and observation errors improves the modeling method in real world scenarios.

Our second case study builds on the use of these Bayesian fitting methods to rapidly model and optimize dose uniformity in the ion implantation process. Here, our main contribution is the tuning algorithm itself, which is comprised of two components. The first component is a forward model that predicts the implantation dose across a wafer as a function of the implantation time spent at each point on the wafer, and the second is an optimization component that uses this forward model to find a sufficient set of tuning times. We compare this method to existing industry methods and see superior results in a number of metrics. This case study again highlights the key themes of this thesis. First, we assume a physically motivated relationship between the implantation times and doses, and second, we utilize a well-chosen, physically motivate prior. Both of these prevent overfitting with limited data and in many cases are able to accurately predict and tune ion implantation processes, often using only a single piece of training data. For traditional deep learning approaches this is virtually impossible, and even compared to a problem specific approach, such as the industry method, we still see less overfitting as both the tuning time and success rates of our method are superior.

The third case study focuses on using kernel density estimation for fault detection. We use this method to create probability distributions for sensor information under nominal processing conditions, then use these to determine the likelihood that new sensor information comes from a normally operating process. We compare this method to a number of existing methods, including traditional statistical process control, one-class support vector machines with principle component analysis, and variational auto-encoders. Again, this case study highlights the two key themes of this thesis. First, our incorporation of process specific knowledge, including the time-series specific feature selection and univariate sensor distributions, helps reduce a large number of potential features to a feasible amount, and this process specific method outperforms more generic methods such as PCA. Second, the proposed method is able to detect faults with high

accuracy even with small amounts of training data. Deep learning methods such as variational auto-encoders were unable to achieve this, as they suffered from overfitting.

In our final case study, we use Gaussian processes and Bayesian optimization to rapidly tune thickness uniformities in sputtering deposition processes. Here, we model the effects of both chamber geometries and process parameters using Gaussian processes and incorporate a physics-based pre-processor in order to simplify these inputs. We simulate the convergence rate with this model and other historical models, and find that it outperforms a wide variety of other models, including polynomial, spline, gradient boosted regression, and deep learning methods. Again, we see the two common themes present throughout this thesis. First, the inherent regularization of the Gaussian process models provides better tuning results with little training data, and secondly, the domain specific pre-processing makes not only the Gaussian process models feasible, but also the other tested methods.

6.2 Future Work

While this thesis presents results demonstrating the applicability of probabilistic methods to semiconductor fabrication, there is still additional work that can be investigated. Such additional investigation includes both extensions to the approaches presented here, as well as other applications.

As we already discussed extensions to the presented work in the previous chapters, we will not discuss them in depth here; however, we will highlight a few key extensions that are particularly relevant. For our proposed virtual metrology framework, we believe further consideration of recipe parameters,could be particularly valuable, in particular for run to run control applications. Additionally, for the corresponding modeling framework, we believe a more effective model parameter estimation method that does not assume equal values between all system coefficients would significantly increase its performance. For our proposed ion implantation tuning method, a key area of future exploration is the trade-off between mean dose and wafer uniformity. Preliminary observations suggest this trade-off may exist; however, it is not yet confirmed, and may help inform future recipe choices. For our kernel density estimation fault detection methodology, a primary future focus could be developing and incorporating existing robust length scale selection methods into our methodology. These length scales

significantly impact the method, and increasing the robustness of their selection will likely improve future performance. Finally, we believe our use of Gaussian processes to improve thickness uniformity in sputtering deposition processes can also be used to optimize other metrics of interest, such as the deposition resistivity uniformity.

In addition to these extensions, we believe there are numerous unexplored applications of probabilistic machine learning methods to semiconductor fabrication. Some examples of these are applications of our proposed methods, such as Bayesian optimization or kernel density estimation fault detection, to other processes which have not yet been considered, such as lithography or chemical mechanical planarization. While the benefit of this may seem limited, applying known techniques to related scenarios is useful. Most of these methods have not yet been adopted by industries, and by further demonstrating their generalizability, we present a stronger argument for their wide spread adoption.

One key area that we did not explore is predictive maintenance, the goal of which is to predict faults before they occur. While this can be thought of as an extension to fault detection, they underlying methods often differ significantly, and must be more carefully considered. Traditionally, these methods have been difficult to implement successfully; however, this also presents an opportunity for future successes.

A second key area for future work is the consideration of additional probabilistic methods. Additional methods like hidden Markov models, or probabilistic deep learning methods, both present new areas to explore. While we have explored many of the key methods and applications in this thesis, there are still numerous future works in the application of probabilistic machine learning techniques to semiconductor fabrication, making it a promising field for future research.

References

[1] G. Moore, "Moore's law," *Electron. Mag.*, vol. 38, no. 8, p. 114, 1965.

[2] K. F. Brennan and A. S. Brown, *Theory of modern electronic semiconductor devices*. John Wiley, 2002.

[3] G. S. May and C. J. Spanos, *Fundamentals of semiconductor manufacturing and process control*. John Wiley & Sons, 2006.

[4] K. Kuhn, C. Kenyon, A. Kornfeld, M. Liu, A. Maheshwari, S. Wei-kai, S. Sivakumar, G. Taylor, P. VanDerVoorn, and K. Zawadzki, "Managing process variation in Intel's 45 nm CMOS technology," *Intel Technol. J.*, vol. 12, no. 2, pp. 93–94, 2008.

[5] D. S. Boning and J. E. Chung, "Statistical metrology: understanding spatial variation in semiconductor manufacturing," in *Proceedings of SPIE - The International Society for Optical Engineering*, 1996, vol. 2874, pp. 16–26.

[6] D. Boning, J. Chung, D. Ouma, and R. Divecha, "Spatial Variation in Semiconductor Processes: Modeling for Control," 1997.

[7] M. Choi and L. Milor, "Impact on circuit performance of deterministic within-die variation in nanoscale semiconductor manufacturing," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 25, no. 7, pp. 1350–1367, 2006.

[8] N. Sebe, I. Cohen, A. Garg, and T. S. Huang, *Machine learning in computer vision*, vol. 29. Springer Science & Business Media, 2005.

[9] A. I. Khan and S. Al-Habsi, "Machine learning in computer vision," *Procedia Comput. Sci.*, vol. 167, pp. 1444–1451, 2020.

[10] S. Nosratabadi, A. Mosavi, P. Duan, P. Ghamisi, F. Filip, S. S. Band, U. Reuter, J. Gama, and A. H. Gandomi, "Data science in economics: comprehensive review of advanced machine learning and deep learning methods," *Mathematics*, vol. 8, no. 10, p. 1799, 2020.

[11] W. Huang, K. K. Lai, Y. Nakamori, S. Wang, and L. Yu, "Neural networks in finance and economics forecasting," *Int. J. Inf. Technol. Decis. Mak.*, vol. 6, no. 1, pp. 113–140, 2007.

[12] H. Ghoddusi, G. G. Creamer, and N. Rafizadeh, "Machine learning in energy economics and finance: A review," *Energy Econ.*, vol. 81, pp. 709–727, 2019.

[13] F. Olsson, "A literature survey of active machine learning in the context of natural language processing," 2009.

[14] T. H. Sandhu, "Machine Learning and Natural Language Processing – a Review," *Int. J. Adv. Res. Comput. Sci.*, vol. 9, no. 2, pp. 582–584, 2018.

[15] K. Wang, S. Tong, B. Eynard, L. Roucoules, and N. Matta, "Review on Application of Data Mining in Product Design and Manufacturing," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, 2007, vol. 4, pp. 613–618.

[16] M. Sendelbach, N. Sarig, K. Wakamoto, H. K. (Helen) Kim, P. Isbester, M. Asano, K. Matsuki, A. Vaid, C. Osorio, and C. Archie, "Impact of shrinking measurement error budgets on qualification metrology sampling and cost," in *Metrology, Inspection, and Process Control for Microlithography XXVIII*, 2014, vol. 9050, p. 90501M.

[17] G. A. Susto, S. Pampuri, A. Schirru, G. De Nicolao, S. F. McLoone, and A. Beghi, "Automatic Control and Machine Learning for Semiconductor Manufacturing: Review and Challenges," 2012.

[18] J. Moyne and J. Iskandar, "Big data analytics for smart manufacturing: Case studies in

semiconductor manufacturing," *Processes*, vol. 5, no. 3, 2017.

[19] D. S. Boning, I. M. Elfadel, and X. Li, "A Preliminary Taxonomy for Machine Learning in VLSI CAD," in *Machine Learning in VLSI Computer-Aided Design*, I. (Abe) M. Elfadel, D. S. Boning, and X. Li, Eds. Springer International Publishing, 2019, pp. 1–16.

[20] A. Diez-Olivan, J. Del Ser, D. Galar, and B. Sierra, "Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0," *Inf. Fusion*, vol. 50, pp. 92–111, Oct. 2019.

[21] C. J. Spanos, H. F. Guo, A. Miller, and J. Levine-Parrill, "Real-Time Statistical Process Control Using Tool Data," *IEEE Trans. Semicond. Manuf.*, vol. 5, no. 4, pp. 308–318, Nov. 1992.

[22] F. Tsung, Y. Li, and M. Jin, "Statistical process control for multistage manufacturing and service operations: A review and some extensions," *Int. J. Serv. Oper. Informatics*, vol. 3, no. 2, pp. 191–204, 2008.

[23] W. Tian, H. You, C. Zhang, S. Kang, X. Jia, and W. T. K. Chien, "Statistical Process Control for Monitoring the Particles with Excess Zero Counts in Semiconductor Manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 1, pp. 93–103, Feb. 2019.

[24] H. Chen and D. Boning, "Online and incremental machine learning approaches for IC yield improvement," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, Nov. 2017, pp. 786–793.

[25] N. B. Gallagher, B. M. Wise, S. W. Butler, D. D. White, and G. G. Barna, "Development and Benchmarking of Multivariate Statistical Process Control Tools for a Semiconductor Etch Process: Improving Robustness through Model Updating," *IFAC Proc. Vol.*, vol. 30, no. 9, pp. 79–84, 1997.

[26] V. A. Sotiris, P. W. Tse, and M. G. Pecht, "Anomaly detection through a Bayesian support vector machine," *IEEE Trans. Reliab.*, vol. 59, no. 2, pp. 277–286, Jun. 2010.

[27] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Piatt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems*, 2000, pp. 582–588.

[28] D. M. J. Tax and R. P. W. Duin, "Support Vector Data Description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, 2004.

[29] C. E. Rasmussen, "Gaussian Processes in machine learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3176, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Springer Berlin Heidelberg, 2004, pp. 63–71.

[30] M. Kemmler, E. Rodner, E. S. Wacker, and J. Denzler, "One-class classification with Gaussian processes," *Pattern Recognit.*, vol. 46, no. 12, pp. 3507–3518, 2013.

[31] H. Lian, "On feature selection with principal component analysis for one-class SVM," *Pattern Recognit. Lett.*, vol. 33, no. 9, pp. 1027–1031, Jul. 2012.

[32] S. Mahadevan and S. L. Shah, "Fault detection and diagnosis in process data using one-class support vector machines," *J. Process Control*, vol. 19, no. 10, pp. 1627–1639, Dec. 2009.

[33] V. L. Cao, M. Nicolau, and J. McDermott, "A hybrid autoencoder and density estimation model for anomaly detection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9921 LNCS, pp. 717–726.

[34] M. Gutoski, N. M. R. Aquino, M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes, "Detection of Video Anomalies Using Convolutional Autoencoders and One-Class Support Vector Machines," in *XIII Brazilian Congress on Computational Intelligence*, 2019, vol. 2017, pp. 1–12.

[35] W. Jiang, Y. Hong, B. Zhou, X. He, and C. Cheng, "A GAN-Based Anomaly Detection Approach for Imbalanced Industrial Time Series," *IEEE Access*, vol. 7, pp. 143608–143619, 2019.

[36] S. Plakias and Y. S. Boutalis, "Exploiting the generative adversarial framework for one-class multi-dimensional fault detection," *Neurocomputing*, vol. 332, pp. 396–405, Mar. 2019.

[37] J. Yang, Z. Sun, and Y. Chen, "Fault detection using the clustering-kNN rule for gas sensor arrays," *Sensors (Switzerland)*, vol. 16, no. 12, Dec. 2016.

[38] C. Zhang and Y. Li, "The application and research of fault detection based on PC-KNN in semiconductor batch process," in *2013 25th Chinese Control and Decision Conference, CCDC 2013*, May 2013, pp. 4209–4214.

[39] T. S. Li and C. L. Huang, "Defect spatial pattern recognition using a hybrid SOM-SVM approach in semiconductor manufacturing," *Expert Syst. Appl.*, vol. 36, no. 1, pp. 374–385, Jan. 2009.

[40] J. Park, I. H. Kwon, S. S. Kim, and J. G. Baek, "Spline regression based feature extraction for semiconductor process fault detection using support vector machine," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5711–5718, May 2011.

[41] K. Kerdprasop and N. Kerdprasop, "A data mining approach to automate fault detection model development in the semiconductor manufacturing process," *Int. J. Mech.*, vol. 5, no. 4, pp. 336–344, 2011.

[42] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 30, no. 2, pp. 135–142, May 2017.

[43] S. J. Hong, W. Y. Lim, T. Cheong, and G. S. May, "Fault detection and classification in plasma etch equipment for semiconductor manufacturing e-diagnostics," *IEEE Trans. Semicond. Manuf.*, vol. 25, no. 1, pp. 83–93, Feb. 2012.

[44] A. Zhakov, H. Zhu, A. Siegel, S. Rank, T. Schmidt, L. Fienhold, and S. Hummel, "Automatic fault detection in rails of overhead transport systems for semiconductor fabs," May 2019.

[45] Y. C. Su, T. H. Lin, F. T. Cheng, and W. M. Wu, "Accuracy and real-time considerations for implementing various virtual metrology algorithms," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 3, pp. 426–434, Aug. 2008.

[46] P. Kang, D. Kim, H. J. Lee, S. Doh, and S. Cho, "Virtual metrology for run-to-run control in semiconductor manufacturing," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2508–2522, Mar. 2011.

[47] M. H. Hung, T. H. Lin, F. T. Cheng, and R. C. Lin, "A novel virtual metrology scheme for predicting CVD thickness in semiconductor manufacturing," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 3, pp. 308–316, Jun. 2007.

[48] T. Hirai and M. Kano, "Adaptive virtual metrology design for semiconductor dry etching process through locally weighted partial least squares," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 2, pp. 137–144, May 2015.

[49] B. E. Stine, D. S. Boning, J. E. Chung, L. Camilletti, F. Kruppa, E. R. Equi, W. Loh, S. Prasad, M. Muthukrishnan, D. Towery, M. Berman, and A. Kapoor, "The physical and electrical effects of metal-fill patterning practices for oxide chemical-mechanical polishing processes," *IEEE Trans. Electron Devices*, vol. 45, no. 3, pp. 665–679, Mar. 1998.

[50] C. I. Lang and D. S. Boning, "Modeling Spin Coating over Topography and Uniformity Improvements Through Fill Patterns for Advanced Packaging Technologies," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 1, pp. 62–69, Feb. 2019.

[51] C. I. Lang and D. S. Boning, "Modeling and controlling layout dependent variations in semi-additive copper electrochemical plating," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 4, pp.

366–373, Nov. 2019.

[52] T. H. Pan, B. Q. Sheng, D. S. H. Wong, and S. S. Jang, "A virtual metrology system for predicting end-of-line electrical properties using a MANCOVA model with tools clustering," *IEEE Trans. Ind. Informatics*, vol. 7, no. 2, pp. 187–195, May 2011.

[53] C. Park, Y. Kim, Y. Park, and S. B. Kim, "Multitask learning for virtual metrology in semiconductor manufacturing systems," *Comput. Ind. Eng.*, vol. 123, pp. 209–219, Sep. 2018.

[54] G. A. Susto, S. Pampuri, A. Schirru, A. Beghi, and G. De Nicolao, "Multi-step virtual metrology for semiconductor manufacturing: A multilevel and regularization methods-based approach," *Comput. Oper. Res.*, vol. 53, pp. 328–337, 2015.

[55] K. B. Lee and C. O. Kim, "Recurrent feature-incorporated convolutional neural network for virtual metrology of the chemical mechanical planarization process," *J. Intell. Manuf.*, vol. 31, no. 1, pp. 73–86, 2020.

[56] S. Kang, "On Effectiveness of Transfer Learning Approach for Neural Network-Based Virtual Metrology Modeling," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 149–155, Feb. 2018.

[57] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches," *arXiv*, 2019. http://arxiv.org/abs/1912.07383.

[58] S. Kang, "Joint modeling of classification and regression for improving faulty wafer detection in semiconductor manufacturing," *J. Intell. Manuf.*, vol. 31, no. 2, pp. 319–326, 2020.

[59] W. Zhang, D. Yang, and H. Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2213–2227, 2019.

[60] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine Learning approach for Predictive Maintenance in Industry 4.0," 2018.

[61] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transp. Res. Part C Emerg. Technol.*, vol. 45, pp. 17–26, Aug. 2014.

[62] T. Le, M. Luo, J. Zhou, and H. L. Chan, "Predictive maintenance decision using statistical linear regression and kernel methods," 2014.

[63] C. Y. Lee, T. S. Huang, M. K. Liu, and C. Y. Lan, "Data science for vibration heteroscedasticity and predictive maintenance of rotary bearings," *Energies*, vol. 12, no. 5, p. 801, 2019.

[64] K. B. Irani, J. Cheng, U. M. Fayyad, and Z. Qian, "Applying Machine Learning to Semiconductor Manufacturing," *IEEE Expert. Syst. their Appl.*, vol. 8, no. 1, pp. 41–47, Feb. 1993.

[65] H. Zhang, Z. Jiang, and C. Guo, "Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology," *Int. J. Adv. Manuf. Technol.*, vol. 41, no. 1–2, pp. 110–121, 2009.

[66] J. Seo, J. H. Kim, M. Lee, K. You, J. Moon, D. H. Lee, and U. Paik, "Multi-objective optimization of tungsten CMP slurry for advanced semiconductor manufacturing using a response surface methodology," *Mater. Des.*, vol. 117, pp. 131–138, 2017.

[67] J. P. C. Kleijnen, "Response surface methodology," *Int. Ser. Oper. Res. Manag. Sci.*, vol. 216, no. 2, pp. 81–104, 2015.

[68] P. K. Mozumder and L. M. Loewenstein, "Method for Semiconductor Process Optimization Using Functional Representations of Spatial Variations and Selectivity," *IEEE Trans. Components, Hybrids, Manuf. Technol.*, vol. 15, no. 3, pp. 311–316, Jun. 1992.

[69] S. Lv, H. Kim, B. Zheng, and H. Jin, "A review of data mining with Big Data towards its applications in the electronics industry," *Appl. Sci.*, vol. 8, no. 4, p. 582, 2018.

[70] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmuller, T. Bauernhansl, A. Knapp, and A. Kyek, "Deep reinforcement learning for semiconductor production scheduling," in *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference, ASMC 2018*, 2018, pp. 301–306.

[71] I. B. Park, J. Huh, J. Kim, and J. Park, "A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1420–1431, 2020.

[72] A. Kuhnle, N. Röhrig, and G. Lanza, "Autonomous order dispatching in the semiconductor industry using reinforcement learning," *Procedia CIRP*, vol. 79, pp. 391–396, 2019.

[73] H. M. Torun, M. Swaminathan, A. Kavungal Davis, and M. L. F. Bellaredj, "A Global Bayesian Optimization Algorithm and Its Application to Integrated System Design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 26, no. 4, pp. 792–802, Apr. 2018.

[74] A. Sakurai, K. Yada, T. Simomura, S. Ju, M. Kashiwagi, H. Okada, T. Nagao, K. Tsuda, and J. Shiomi, "Ultranarrow-Band Wavelength-Selective Thermal Emission with Aperiodic Multilayered Metamaterials Designed by Bayesian Optimization," *ACS Cent. Sci.*, vol. 5, no. 2, pp. 319–326, Feb. 2019.

[75] H. C. Herbol, W. Hu, P. Frazier, P. Clancy, and M. Poloczek, "Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization," *npj Comput. Mater.*, vol. 4, no. 1, pp. 1–7, 2018.

[76] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[77] F. Auger, M. Hilairet, J. M. Guerrero, E. Monmasson, T. Orlowska-Kowalska, and S. Katsura, "Industrial applications of the Kalman filter: A review," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5458–5471, Dec. 2013.

[78] S. F. Schmidt, "The Kalman filter : Its recognition and development for aerospace applications," *J. Guid. Control. Dyn.*, vol. 4, no. 1, pp. 4–7, 1981.

[79] P. C. Young, "Comments on 'on-line identification of linear dynamic systems with applications to kalman filtering,'" *IEEE Trans. Automat. Contr.*, vol. 17, no. 2, pp. 269–270, 1972.

[80] S. Y. Chen, "Kalman filter for robot vision: A survey," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4409–4420, 2012.

[81] J. Z. Sasiadek and Q. Wang, "Sensor fusion based on fuzzy Kalman filtering for autonomous robot vehicle," in *Proceedings - IEEE International Conference on Robotics and Automation*, 1999, vol. 4, pp. 2970–2975.

[82] W. Yang *et al.*, "Prevalence of Diabetes among Men and Women in China," *N. Engl. J. Med.*, vol. 362, no. 12, pp. 1090–1101, 2010.

[83] R. H. Koning, H. Neudecker, and T. Wansbeek, "Block Kronecker products and the vecb operator," *Linear Algebra Appl.*, vol. 149, no. C, pp. 165–184, 1991.

[84] L. L. T. Chan, X. Wu, J. Chen, L. Xie, and C. I. Chen, "Just-In-Time Modeling with Variable Shrinkage Based on Gaussian Processes for Semiconductor Manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 3, pp. 335–342, Aug. 2018.

[85] S. A. Lynn, J. Ringwood, and N. MacGearailt, "Global and Local Virtual Metrology Models for a Plasma Etch Process," *IEEE Trans. Semicond. Manuf.*, vol. 25, no. 1, pp. 94–103, 2012.

[86] P. Kang, H. joo Lee, S. Cho, D. Kim, J. Park, C. K. Park, and S. Doh, "A virtual metrology system

for semiconductor manufacturing," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12554–12561, Dec. 2009.

[87] E. Sachs, A. Hu, and A. Ingolfsson, "Run by run process control: combining SPC and feedback control," *IEEE Trans. Semicond. Manuf.*, vol. 8, no. 1, pp. 26–43, 1995.

[88] A. Chen and R.-S. Guo, "Age-based double EWMA controller and its application to CMP processes," *IEEE Trans. Semicond. Manuf.*, vol. 14, no. 1, pp. 11–19, 2001.

[89] D. Zeng and C. J. Spanos, "Virtual Metrology Modeling for Plasma Etch Operations," *IEEE Trans. Semicond. Manuf.*, vol. 22, no. 4, pp. 419–431, 2009.

[90] S. Lynn, J. Ringwood, E. Ragnoli, S. McLoone, and N. MacGearailty, "Virtual metrology for plasma etch using tool variables," in *2009 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 2009, pp. 143–148.

[91] H. Purwins, B. Barak, A. Nagi, R. Engel, U. Höckele, A. Kyek, S. Cherla, B. Lenz, G. Pfeifer, and K. Weinzierl, "Regression methods for virtual metrology of layer thickness in chemical vapor deposition," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 1, pp. 1–8, 2013.

[92] J. Besnard, D. Gleispach, H. Gris, A. Ferreira, A. Roussy, C. Kernaflen, and G. Hayderer, "Virtual metrology modeling for cvd film thickness," *Int. J. Control Sci. Eng.*, vol. 2, no. 3, pp. 26–33, 2012.

[93] H. Tsuda, H. Shirai, and E. Kawamura, "A Precise Photolithography Process Control Method Using Virtual Metrology," *Electron. Commun. Japan*, vol. 97, no. 10, pp. 48–55, 2014.

[94] W.-T. Yang, J. Blue, A. Roussy, J. Pinaton, and M. S. Reis, "A Structure Data-Driven Framework for Virtual Metrology Modeling," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1297–1306, 2020.

[95] H. Cai, J. Feng, Q. Yang, W. Li, X. Li, and J. Lee, "A virtual metrology method with prediction uncertainty based on Gaussian process for chemical mechanical planarization," *Comput. Ind.*, vol. 119, p. 103228, 2020.

[96] Y.-J. Chang, Y. Kang, C.-L. Hsu, C.-T. Chang, and T. Y. Chan, "Virtual Metrology Technique for Semiconductor Manufacturing," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006, pp. 5289–5293.

[97] A. A. Khan, J. R. Moyne, and D. M. Tilbury, "Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares," *J. Process Control*, vol. 18, no. 10, pp. 961–974, 2008.

[98] D. Kurz, J. Pilz, A. Schirru, S. Pampuri, and C. De Luca, "A sampling decision system for semiconductor manufacturing - relying on virtual metrology and actual measurements," in *Proceedings of the Winter Simulation Conference 2014*, 2014, pp. 2649–2660.

[99] S. Pampuri, A. Schirru, G. Fazio, and G. De Nicolao, "Multilevel Lasso applied to Virtual Metrology in semiconductor manufacturing," in *2011 IEEE International Conference on Automation Science and Engineering*, 2011, pp. 244–249.

[100] G. A. Susto, A. B. Johnston, P. G. O'Hara, and S. McLoone, "Virtual metrology enabled early stage prediction for enhanced control of multi-stage fabrication processes," in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, 2013, pp. 201–206.

[101] N. G. Orji, Y. S. Obeng, C. Beitia, S. Mashiro, J. Moyne, and others, "Virtual Metrology White Paper - International Roadmap for Devices and Systems (IRDS)," 2018.

[102] J. Feng, X. Jia, F. Zhu, J. Moyne, J. Iskandar, and J. Lee, "An Online Virtual Metrology Model With Sample Selection for the Tracking of Dynamic Manufacturing Processes With Slow Drift," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 4, pp. 574–582, 2019.

[103] T. H. Smith and D. S. Boning, “Self-tuning EWMA controller utilizing artificial neural network function approximation techniques,” *IEEE Trans. components, Packag. Manuf. Technol. Part C. Manuf.*, vol. 20, no. 2, pp. 121–132, Apr. 1997.

[104] G. Welch, G. Bishop, and others, “An introduction to the Kalman filter,” 1995. https://perso.crans.org/club-krobot/doc/kalman.pdf.

[105] D. S. Boning, M. B. McIlrath, P. Penfield, and E. M. Sachs, “A general semiconductor press modeling framework,” *IEEE Trans. Semicond. Manuf.*, vol. 5, no. 4, pp. 266–280, 1992.

[106] J. Nicolau, “Stationary processes that look like random walks—the bounded random walk process in discrete and continuous time,” *Econom. Theory*, vol. 18, no. 1, pp. 99–118, 2002.

[107] C. Cardinali, “Observation influence diagnostic of a data assimilation system,” in *Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications (Vol. II)*, Springer, 2013, pp. 89–110.

[108] T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, 1996.

[109] Y. Wang, X. Hao, and C. Wu, “Forecasting stock returns: A time-dependent weighted least squares approach,” *J. Financ. Mark.*, vol. 53, p. 100568, 2021.

[110] S. Amasaki and C. Lokan, “On the effectiveness of weighted moving windows: Experiment on linear regression based software effort estimation,” *J. Softw. Evol. Process*, vol. 27, no. 7, pp. 488–507, 2015.

[111] N. W. Cheung, “Plasma immersion ion implantation for semiconductor processing,” *Mater. Chem. Phys.*, vol. 46, no. 2–3, pp. 132–139, 1996.

[112] J. S. Williams, “Ion implantation of semiconductors,” *Mater. Sci. Eng. A*, vol. 253, no. 1–2, pp. 8–15, 1998.

[113] L. A. Larson, J. M. Williams, and M. I. Current, “Ion implantation for semiconductor doping and materials modification,” *Rev. Accel. Sci. Technol. Accel. Appl. Ind. Environ.*, pp. 11–40, 2012.

[114] N. Tokoro, D. Holbrook, and D. Hacker, “Introduction of the Varian VIISta 3000 single wafer high-energy ion implanter,” in *2000 International Conference on Ion Implantation Technology Proceedings. Ion Implantation Technology - 2000 (Cat. No.00EX432)*, 2003, pp. 368–371.

[115] C. B. Yarling, W. A. Keenan, and L. A. Larson, “The history of uniformity mapping in ion implantation,” *Nucl. Inst. Methods Phys. Res. B*, vol. 55, no. 1–4, pp. 235–242, 1991.

[116] R. J. Matyi, D. L. Chapek, D. P. Brunco, S. B. Felch, and B. S. Lee, “Boron doping of silicon by plasma source ion implantation,” *Surf. Coatings Technol.*, vol. 93, no. 2–3, pp. 247–253, 1997.

[117] K. Howard, “Implant uniformity evaluation using a Varian/Extrion scan compensator module on an electrostatic scanning ion implanter,” *Nucl. Inst. Methods Phys. Res. B*, vol. 55, no. 1–4, pp. 202–206, 1991.

[118] P. Fisher, “Scan compensation for varian serial implanters,” *Nucl. Inst. Methods Phys. Res. B*, vol. 37–38, no. C, pp. 525–527, 1989.

[119] N. Turner, “Improved uniformity of implanted dose by a compensated scan pattern generator,” *Nucl. Instruments Methods*, vol. 189, no. 1, pp. 311–318, 1981.

[120] J. C. Olson, A. Renau, and J. Buff, “Scanned beam uniformity control in the VIISta 810 ion implanter,” in *Proceedings of the International Conference on Ion Implantation Technology*, 1999, vol. 1, pp. 169–172.

[121] B. J. Miga, “Low Energy Ion Implant Capabilities at RIT,” in *Journal of the Microelectronic*

Engineering Conference, 2002, vol. 12, no. 1, p. 14.

[122] A. Renau and J. T. Scheuer, "Comparison of plasma doping and beamline technologies for low energy ion implantation," in *Proceedings of the International Conference on Ion Implantation Technology*, 2002, vol. 22-27-Sept, pp. 151–156.

[123] K. Rudenko, S. Averkin, V. Lukichev, A. Orlikovsky, P. Alexander, and A. Vvatkin, "Ultra shallow p+-n junctions in si produced by plasma immersion ion implantation," *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 6260, Jan. 2006.

[124] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv Prepr. arXiv1807.02811*, 2018.

[125] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.6578&rep=rep1&type=pdf.

[126] C. F. Van Loan, "The ubiquitous Kronecker product," *J. Comput. Appl. Math.*, vol. 123, no. 1–2, pp. 85–100, 2000.

[127] Q. P. He and J. Wang, "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 20, no. 4, pp. 345–354, 2007.

[128] B. E. Goodlin, D. S. Boning, H. H. Sawin, and B. M. Wise, "Simultaneous Fault Detection and Classification for Semiconductor Manufacturing Tools," *J. Electrochem. Soc.*, vol. 150, no. 12, p. G778, 2003.

[129] C. F. Chien, C. Y. Hsu, and P. N. Chen, "Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence," *Flex. Serv. Manuf. J.*, vol. 25, no. 3, pp. 367–388, 2013.

[130] J. Yu, "Fault detection using principal components-based gaussian mixture model for semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 24, no. 3, pp. 432–444, Aug. 2011.

[131] Q. P. He and J. Wang, "Principal component based k-nearest-neighbor rule for semiconductor process fault detection," in *Proceedings of the American Control Conference*, Jun. 2008, pp. 1606–1611.

[132] S. K. S. Fan, C. Y. Hsu, D. M. Tsai, F. He, and C. C. Cheng, "Data-Driven Approach for Fault Detection and Diagnostic in Semiconductor Manufacturing," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1925–1936, Oct. 2020.

[133] E. Kim, S. Cho, B. Lee, and M. Cho, "Fault Detection and Diagnosis Using Self-Attentive Convolutional Neural Networks for Variable-Length Sensor Data in Semiconductor Manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 3, pp. 302–309, Aug. 2019.

[134] J. F. Bithell, "An application of density estimation to geographical epidemiology," *Stat. Med.*, vol. 9, no. 6, pp. 691–701, 1990.

[135] N. Donthu and R. T. Rust, "Note—Estimating Geographic Customer Densities Using Kernel Density Estimation," *Mark. Sci.*, vol. 8, no. 2, pp. 191–203, 1989.

[136] J. Jeon and J. W. Taylor, "Using conditional kernel density estimation for wind power density forecasting," *J. Am. Stat. Assoc.*, vol. 107, no. 497, pp. 66–79, Mar. 2012.

[137] Z. Xie and J. Yan, "Kernel Density Estimation of traffic accidents in a network space," *Comput. Environ. Urban Syst.*, vol. 32, no. 5, pp. 396–406, Sep. 2008.

[138] E. Schubert, A. Zimek, and H. P. Kriegel, "Generalized outlier detection with flexible kernel density estimates," in *SIAM International Conference on Data Mining 2014, SDM 2014*, vol. 2, 2014, pp. 542–550.

[139] R. T. Samuel and Y. Cao, "Fault detection in a multivariate process based on kernel PCA and kernel density estimation," in *ICAC 2014 - Proceedings of the 20th International*

Conference on Automation and Computing: Future Automation, Computing and Manufacturing, 2014, pp. 146–151.

[140] W. J. Lee, G. P. Mendis, M. J. Triebe, and J. W. Sutherland, "Monitoring of a machining process using kernel principal component analysis and kernel density estimation," *J. Intell. Manuf.*, vol. 31, no. 5, pp. 1175–1189, 2020.

[141] R. T. Samuel and Y. Cao, "Nonlinear process fault detection and identification using kernel PCA and kernel density estimation," *Syst. Sci. Control Eng.*, vol. 4, no. 1, pp. 165–174, 2016.

[142] W. Hu, J. Gao, B. Li, O. Wu, J. Du, and S. Maybank, "Anomaly Detection Using Local Kernel Density Estimation and Context-Based Regression," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 2, pp. 218–233, 2020.

[143] B. W. Silverman, *Density estimation: For statistics and data analysis*, vol. 26. CRC Press, 2018.

[144] Y. C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostat. Epidemiol.*, vol. 1, no. 1, pp. 161–187, 2017.

[145] B. A. Turlach, "Bandwidth selection in kernel density estimation: A review," in *CORE and Institut de Statistique*, 1993, pp. 23–493.

[146] M. C. Jones, J. S. Marron, and S. J. Sheather, "A brief survey of bandwidth selection for density estimation," *J. Am. Stat. Assoc.*, vol. 91, no. 433, pp. 401–407, 1996.

[147] S. J. Sheather and M. C. Jones, "A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation," *J. R. Stat. Soc. Ser. B*, vol. 53, no. 3, pp. 683–690, 1991.

[148] M. Müller, "Dynamic time warping," *Inf. Retr. Music Motion*, pp. 69–84, 2007.

[149] S.-T. Chiu, "Bandwidth Selection for Kernel Density Estimation," *Ann. Stat.*, vol. 19, no. 4, pp. 1883–1905, 2007.

[150] S. R. Sain, K. A. Baggerly, and D. W. Scott, "Cross-validation of multivariate densities," *J. Am. Stat. Assoc.*, vol. 89, no. 427, pp. 807–817, 1994.

[151] I. Žliobaitė, "Learning under Concept Drift: an Overview," *arXiv Prepr. arXiv1010.4784*, 2010.

[152] A. Tsymbal, "The problem of concept drift: definitions and related work," *Comput. Sci. Dep. Trinity Coll. Dublin*, vol. 106, no. 2, p. 58, 2004.

[153] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014.

[154] T. R. Hoens, R. Polikar, and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: An overview," *Prog. Artif. Intell.*, vol. 1, no. 1, pp. 89–101, 2012.

[155] A. S. Iwashita and J. P. Papa, "An Overview on Concept Drift Learning," *IEEE Access*, vol. 7, pp. 1532–1547, 2019.

[156] K. Imoto, T. Nakai, T. Ike, K. Haruki, and Y. Sato, "A CNN-based transfer learning method for defect classification in semiconductor manufacturing," in *IEEE Trans. Semicond. Manuf.*, 2019, vol. 32, no. 4, pp. 455–459.

[157] S. Kang, "On Effectiveness of Transfer Learning Approach for Neural Network-Based Virtual Metrology Modeling," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 149–155, 2018.

[158] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11141 LNCS, pp. 270–279.

[159] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[160] C. J. Spanos, "Statistical process control in semiconductor manufacturing," *Proc. IEEE*, vol. 80, no. 6, pp. 819–830, 1992.
[161] A. Chen, R.-S. Guo, and P.-J. Yeh, "An effective SPC approach to monitoring semiconductor manufacturing processes with multiple variation sources," in *Proceedings of ISSM2000. Ninth International Symposium on Semiconductor Manufacturing (IEEE Cat. No. 00CH37130)*, 2000, pp. 446–449.
[162] D. C. Montgomery, *Introduction to statistical quality control*. John Wiley & Sons, 2020.
[163] K. L. Lange, R. J. A. Little, and J. M. G. Taylor, "Robust statistical modeling using the t distribution," *J. Am. Stat. Assoc.*, vol. 84, no. 408, pp. 881–896, 1989.
[164] H. Yu and S. Kim, "SVM Tutorial-Classification, Regression and Ranking.," *Handb. Nat. Comput.*, vol. 1, pp. 479–506, 2012.
[165] H. J. Shin, D. H. Eom, and S. S. Kim, "One-class support vector machines - An application in machine fault detection and classification," *Comput. Ind. Eng.*, vol. 48, no. 2, pp. 395–408, 2005.
[166] A. Beghi, L. Cecchinato, C. Corazzol, M. Rampazzo, F. Simmini, and G. A. Susto, "A one-class SVM based tool for machine learning novelty detection in HVAC chiller systems," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, Jan. 2014, vol. 19, no. 3, pp. 1953–1958.
[167] F. Harrou, A. Dairi, B. Taghezouit, and Y. Sun, "An unsupervised monitoring procedure for detecting anomalies in photovoltaic systems using a one-class Support Vector Machine," *Sol. Energy*, vol. 179, pp. 48–58, Feb. 2019.
[168] L. Shuang and L. Meng, "Bearing fault diagnosis based on PCA and SVM," in *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, ICMA 2007*, Aug. 2007, pp. 3503–3507.
[169] Y. Xie and T. Zhang, "A fault diagnosis approach using SVM with data dimension reduction by PCA and LDA method," in *Proceedings - 2015 Chinese Automation Congress, CAC 2015*, Nov. 2016, pp. 869–874.
[170] P. Ming-qing, Z. Xiao-jun, W. Rui-ming, and L. Liang-yu, "Research of machine fault diagnosis based on PCA and SVDD," *Chinese J. Sensors Actuators*, vol. 19, no. 1, pp. 128–131, 2006.
[171] C. Doersch, "Tutorial on Variational Autoencoders," *arXiv Prepr. arXiv1606.05908*, 2016.
[172] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *ACM International Conference Proceeding Series*, 2014, vol. 02-Decembe, pp. 4–11.
[173] P. Park, P. Di Marco, H. Shin, and J. Bang, "Fault detection and diagnosis using combined autoencoder and long short-term memory network," *Sensors (Switzerland)*, vol. 19, no. 21, p. 4612, 2019.
[174] D. Y. Oh and I. D. Yun, "Residual error based anomaly detection using auto-encoder in SMD machine sound," *Sensors (Switzerland)*, vol. 18, no. 5, 2018.
[175] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," 2018.
[176] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, "Latent space autoregression for novelty detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, pp. 481–490.
[177] A. H. Simon, "Chapter 7 - Sputter Processing," in *Handbook of Thin Film Deposition (Fourth Edition)*, Fourth Edi., K. Seshan and D. Schepis, Eds. William Andrew Publishing, 2018, pp. 195–230.

[178] N. Kupp, K. Huang, J. Carulli, and Y. Makris, "Spatial estimation of wafer measurement parameters using Gaussian process models," in *Proceedings - International Test Conference*, 2012, pp. 1–8.

[179] J. Wan and S. McLoone, "Gaussian Process Regression for Virtual Metrology-Enabled Run-to-Run Control in Semiconductor Manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 12–21, 2018.

[180] M. Frean and P. Boyle, "Using Gaussian processes to optimize expensive functions," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5360 LNAI, pp. 258–267.

[181] C. E. Rasmussen, "Gaussian Processes in machine learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2004, vol. 3176, pp. 63–71.

[182] W. D. Westwood, "Sputter Deposition Processes," in *MRS Bulletin*, vol. 13, no. 12, Elsevier, 1988, pp. 46–51.

[183] S. Swann, "Film thickness distribution in magnetron sputtering," *Vacuum*, vol. 38, no. 8–10, pp. 791–794, 1988.

[184] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Front. Neurorobot.*, vol. 7, p. 21, 2013.

[185] P. Prettenhofer and G. Louppe, "Gradient Boosted Regression Trees," 2014. http://hdl.handle.net/2268/163521%5Cnhttp://orbi.ulg.ac.be/handle/2268/163521.

[186] S. Sekulic and B. R. Kowalski, "MARS: A tutorial," *J. Chemom.*, vol. 6, no. 4, pp. 199–216.

[187] M. J. L. Orr and others, "Introduction to radial basis function networks." Technical Report, center for cognitive science, University of Edinburgh, 1996.

[188] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Comput.*, vol. 5, no. 2, pp. 305–316, 1993.

[189] J. Chenou, G. Hsieh, and T. Fields, "Radial Basis Function Network: Its Robustness and Ability to Mitigate Adversarial Examples," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 102–106.