**The Dissertation Committee for Brent Constant Bregenzer Certifies that this is the approved version of the following dissertation:**

# Control-Friendly Scheduling Algorithms For Multi-tool, Multi-product Manufacturing Systems

**Committee:**

S. Joe Qin, Supervisor

John J. Hasenbein, Co-Supervisor

Thomas F. Edgar

Erhan Kutanoglu

Gyeong S. Hwang

Roger T. Bonnecaze

# Control-Friendly Scheduling Algorithms For Multi-tool, Multi-product Manufacturing Systems

**by**

**Brent Constant Bregenzer, B.S.CH.E.; M.S.E.**

**Dissertation**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**DOCTOR OF PHILOSOPHY**

**The University of Texas at Austin**

**December, 2011**

# Dedication

To My Parents,

And To Tracy

## Acknowledgements

First I would like to thank Dr. Joe Qin for many years of support through the long process of receiving the Ph.D. which included coursework, two internships, a complete change in research focus, and his move to another school. His patience, understanding, and final approval are greatly appreciated. I would also like to thank my co-advisor for the dissertation research, Dr. John Hasenbein, who was very helpful with the scheduling portion of the research and whose timely responses to questions and requests for revisions were extremely helpful and I am grateful for them. Additionally, I would like to thank Dr. Tom Edgar for agreeing to recategorize from committee member to advisor when Dr. Qin moved to USC and also Dr. Roger Bonnecaze who at the same time agreed to join my committee. Finally, I would like to thank Dr. Gyeong Hwang and Dr. Erhan Kutanoglu for their help as committee members as well. The input from the committee was invaluable in helping to shape the final form of this dissertation and it is much improved because of them.

Several lab mates deserve recognition for their company and help along the way including Scott, Elaine, Peter, Jin, Greg, Rick, Quan, Carlos, and Clare. I would especially like to thank Chris Harrison for many useful discussions that helped shape this research in its early days. Finally, I owe a great debt of gratitude to Dan Barad and Kevin Chamness who patiently listened to my ideas and were instrumental in helping me to wrap things up.

I would be remiss if I didn't thank the many wonderful friends who made grad school so rewarding outside of the office and classroom. This includes the students listed above and also April, Brian, Bryan, Devina, Jessie, Landry, Phuong, Reken, Shravi, Stephanie, Tom, and many others. A special thanks to Brooks for introducing me to

coffee, going to lunch on an almost daily basis, and participating in many entertaining conversations. Also, I'd like to give a special thanks to Michael for many memorable moments and the same to my roommates of three years and friends for many more, Scott and Christian. The numerous good times we had will never be forgotten and your encouragement as I finished up was always welcome and refreshing.

Finally, I wish to thank my family for all of their love and encouragement, especially my Aunt Joan and Uncle Dave. I am also grateful for my parents; their unwavering support throughout the years that I was in grad school (and my entire life for that matter). Their unconditional love and faith in me were and continue to be a source of inspiration. Last but certainly not least, I cannot express enough how grateful I am to Tracy for standing by me for the duration of this dissertation process. Her patience, understanding, and optimism were superhuman, and I hope I will be able to repay her for all of her kindness and love as we make our way through life together.

A huge thanks again to everyone! I am humbled by your caring and generosity.

# Control-Friendly Scheduling Algorithms For Multi-tool, Multi-product Manufacturing Systems

Publication No._____

Brent Constant Bregenzer, Ph.D.

The University of Texas at Austin, 2011

Supervisors: Joe Qin, John Hasenbein, Thomas Edgar

The fabrication of semiconductor devices is a highly competitive and capital intensive industry. Due to the high costs of building wafer fabrication facilities (fabs), it is expected that products should be made efficiently with respect to both time and material, and that expensive unit operations (tools) should be utilized as much as possible. The process flow is characterized by frequent machine failures, drifting tool states, parallel processing, and reentrant flows. In addition, the competitive nature of the industry requires products to be made quickly and within tight tolerances. All of these factors conspire to make both the scheduling of product flow through the system and the control of product quality metrics extremely difficult. Up to now, much research has been done on the two problems separately, but until recently, interactions between the two systems, which can sometimes be detrimental to one another, have mostly been ignored. The research contained here seeks to tackle the scheduling problem by utilizing objectives based on control system parameters in order that the two systems might behave in a more beneficial manner.

A non-threaded control system is used that models the multi-tool, multi-product process in a state space form, and estimates the states using a Kalman filter. Additionally, the process flow is modeled by a discrete event simulation. The two systems are then merged to give a representation of the overall system. Two control system matrices, the estimate error covariance matrix from the Kalman filter and a square form of the system observability matrix called the information matrix, are used to generate several control-based scheduling algorithms. These methods are then tested against more tradition approaches from the scheduling literature to determine their effectiveness on both the basis of how well they maintain the outputs near their targets and how well they minimize the cycle time of the products in the system. The two metrics are viewed simultaneously through use of Pareto plots and merits of the various scheduling methods are judged on the basis of Pareto optimality for several test cases.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Brief Overview of Semiconductor Manufacturing

The fabrication of semiconductor devices is viewed as the most complicated and technologically demanding volume manufacturing process in the world [5, 51]. The industry is driven by competition on the international level and the desire to continuously fulfill Moore's law. Moore's law states that the number of transistors on an integrated circuit (IC) doubles every 18 to 24 months [61]. This path of development has led from small-scale integration (SSI) in the early 1960's with fewer than 100 transistors on a chip to today's ultra-large-scale integration (ULSI) with more than a billion transistors on a chip (see Table 1.1 for the progression of integration levels) [63]. Maintaining such a rapid pace of advancement requires innovations in product design and process technology and in management and control of the process environment.

| Integration Level | Time Period | Number of Devices per Chip |
|---|---|---|
| Small-Scale Integration (SSI) | 1960–65 | 2–100 |
| Medium-Scale Integration (MSI) | 1965–75 | 100–10,000 |
| Large-Scale Integration (LSI) | 1975–85 | 10,000–500,000 |
| Very-Large-Scale Integration (VLSI) | 1985–95 | 500,000–5,000,000 |
| Ultra-Large-Scale Integration (ULSI) | 1995–Present | > 5,000,000 |

Table 1.1: Progression of integration levels from 1960 to Present

The fabrication of semiconductor devices begins with blank, polished silicon wafers, which have increased in diameter from 150 mm in the 1980's to 200 mm in the 1990's to the current standard of 300 mm. Periodic increases in diameter are necessary to

enable the economies of scale required for the maintenance of Moore's law [62]. The fabrication process ends with wafers that have many completed integrated circuits (IC'S), also known as die, on their surfaces. Forming the circuitry of the finished IC's requires many hundreds of complex chemical and physical processing steps in which layers of metal, semiconductor, and insulator are deposited and patterned. The unit operations necessary to produce these layers include implantation of dopant molecules to alter the electrical properties of the of the underlying silicon, deposition of thin films, patterning of films using lithography, rapid thermal processing (RTP), etching, and chemical-mechanical planarization (CMP).

The aforementioned processes constitute what is known as wafer fabrication. Wafer fabrication and testing of the IC's on the completed wafer (i.e., wafer probe) comprise the so called front end of the line (FEOL). The back end of the line (BEOL) processes involve cutting the wafers into individual dies and packaging them to make the finished products (e.g., microprocessors or memory chips) and then testing these products for performance. Figure 1.1 gives a diagram of the overall manufacturing process [60]. The interested reader is referred to [1, 2, 5, 9, 12, 52] for more detailed descriptions of processing steps and unit operations.

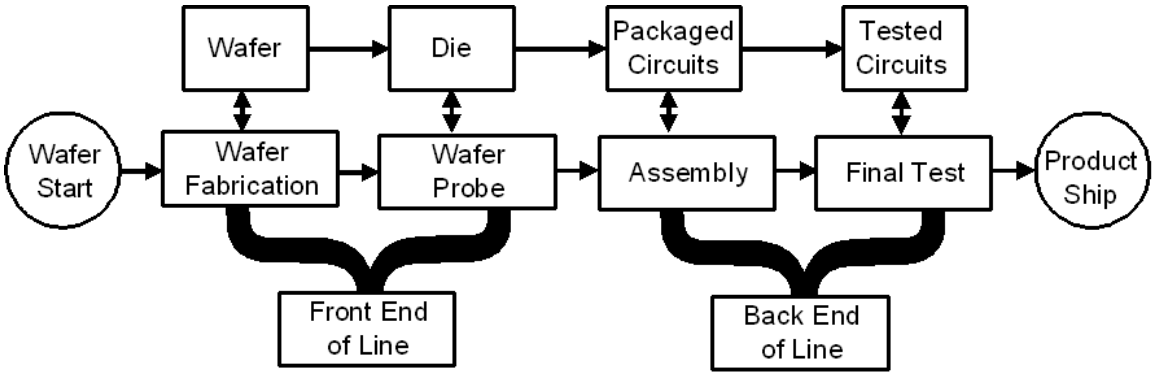


Figure 1.1: A simplified diagram of semiconductor manufacturing. The upper row of boxes represents the product flow, while the middle row represents the process flow.

Due to the extremely small size of the features on the surface of the wafers, IC devices are particularly sensitive to any particles (e.g., dust) that may settle on their surfaces while they are being manufactured. Therefore, wafers are processed in a "clean room" environment where highly efficient filters tightly control air quality and employees must wear specialized suits to prevent the introduction of foreign matter from their persons. Building a clean room large enough to house all of the machinery necessary for the production of IC's is expensive.

Additionally, the tools themselves are quite costly. For example, a single photolithography stepper tool costs around $20 million and with the trend towards sub-50 nm devices, the price is expected to reach more than $50 million [61]. Building a modern wafer fabrication facility, or fab, currently costs billions of dollars [49]. Therefore, it is expected that the expensive tools should be utilized as much as possible in order to recoup their initial cost rapidly. Additionally, products should be made as quickly as possible and with a very high yield in order to minimize wasted material and keep the facility competitive with other manufacturers. Generating a high percentage of devices

with the required functionality (i.e., a high yield) is also difficult due to the miniscule dimensions of modern IC's and the complex processes necessary for their production.

The economic pressures of the industry combined with the inherent difficulty of manufacturing such complex parts to extremely tight tolerances, confronts both process control and scheduling researchers with a unique and an exceptionally demanding set of problems. (Here, the term scheduling is used in the most general sense to represent any type of control used to manage the flow of products within the manufacturing environment. A more detailed description of individual scheduling terms will be given in the next chapter.)

### 1.2 Interactions Between Process Control and Scheduling Systems

Up to this point in time, large amounts of research have been conducted in the area of advanced process control (APC) of semiconductor processes. See, for example, [1, 9, 10, 50, 51, 52] for surveys of the field. The same can also be said for research in the field of scheduling (see [3, 5, 6, 7, 8, 31]). The responses to a survey of industrial practitioners in both areas [8] indicates that the interplay between scheduling and APC systems exists, as decisions made by one system can sometimes affect the performance of the other, but studies of these interactions and methods for their management have been largely ignored.

However, a few exceptions do exist. Ruiz et al. [37] use a fault detection system (FDS) to supply supplementary information to a rescheduling system for a batch chemical production facility. It is shown that the integration of the FDS with the rescheduling system allows for better overall performance of the plant than with the scheduling system alone.

More recently, the interest in extending semiconductor control methods to multi-product and multi-process systems [38] has led to several studies where control and

scheduling are considered together. Chen et al. examine the effect of wafer sequencing on the effectiveness of run-to-run (R2R or RtR) control for a chemical mechanical polishing (CMP) tool [39]. Under the assumption that R2R controllers, as negative feedback controllers, are effective at rejecting low-frequency disturbances, they define a method for arranging incoming wafers in such a way that they mimic a low-frequency disturbance. Rather than letting the wafers enter the system at random, they are ordered from thinnest to thickest or vice versa. Patel [40] uses a dynamic game methodology to design a dispatch system which takes into account the effect of dispatching decisions on future controller performance for threaded EWMA controllers.

Anderson and Hanish [133] give a brief review of some dispatch rules used in industry and then use data from a CMP process to test two dispatch rules for their ability to minimize cycle time (CT). The first rule is a simple random selection using FIFO; the second is a two step rule that uses historical data from the measurement and control systems. The first half of the new rule forms a priority queue by selecting products based on their incoming thickness (based on previous measurement information), placing wafers with thinner layers to be removed first. The second half of the dispatch rule selects tools based on an SPT (shortest processing time) rule that uses historical data from the control system to determine the tool (from those available) with the highest removal rate. It is found at low system volumes, that queues are mostly empty and only the second part of the new dispatch is in affect and results in CT reduction of 2–5%. Conversely, high volume systems rarely have idle tools so only the first portion of the rule is active; here the priority queue yields improvements of up to 50% over the random selection. In medium volume systems, the rule showed little effect. The paper finishes with some general observations on possible interactions between dispatching and control including benefits and conflicts.

In the thesis by Pasadyn [41] and related papers with Edgar et al. [53, 54, 55], a Kalman filter based run-to-run controller for multi-product and multi-machine systems is studied. In the multiple product/process environment, it is found by way of some small scale example problems (only 20 samples and one replicate) that the way in which jobs and measurements are scheduled can greatly affect the controller performance. Using the trace of the state estimate error covariance matrix as a measure of controller performance, several simple, scheduling-related objective functions for the minimization of controller error are proposed and studied.

The approach of Pasadyn is used as a starting point for this research. The system model is developed in a state space form and a Kalman filter is used for state estimation. Some adjustments are made to the model and several updating schemes are tested for control performance and schedule performance using simulations. The results from these tests are compared to some of the basic methods that are sometimes employed in industry and in other academic works. The objective of the algorithms employed is to alter the schedule (hopefully not too far from the optimal schedule) in order to improve the controller performance and keep the outputs close to their targets.

# Chapter 2

# Background and Literature Review

## 2.1 Scheduling for Semiconductor Manufacturing

"Scheduling" is often used as a general term in industry to refer to any number of different activities that involve the control of material movement throughout the manufacturing process. This type of control actually takes place on several different levels, each with their own time scales and scopes. Figure 2.1 diagrams the various levels of "scheduling" activity as they are defined in a pair of industrial surveys [7, 8]. See Appendix A for concise definitions of the terms used in Figure 2.1.

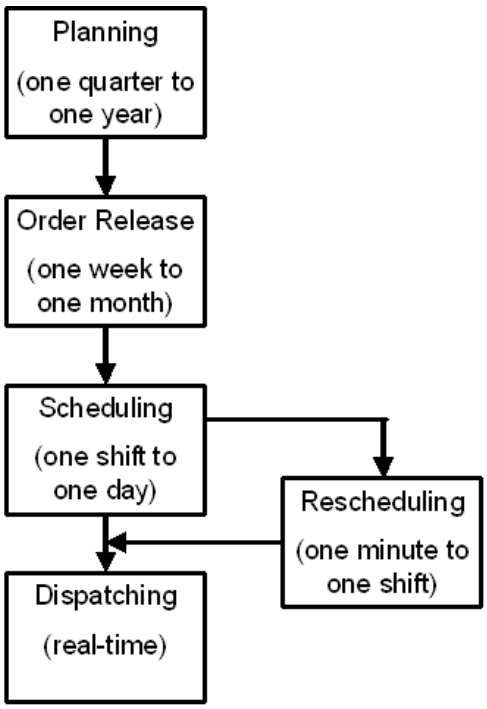


Figure 2.1: A flow diagram demonstrating the various levels of product flow control.

The highest level activity is production planning, which uses plant capacity models and market demand forecasts to assess the fab's ability of meet customer orders. Here, decisions are made that affect such things as the mix of products, staffing levels, and number of tools needed. Immediately below planning is the order release system, which acts as the link between the planning system and the actual shop floor. The order release system is often called the lot release system because it does exactly what this name implies: it decides what type of lots to send to manufacturing, how many, and at what rate they will be introduced.

Once the lots reach the shop floor, there are several options for control of their movement. Ideally, a detailed schedule describing the exact movement of all lots in the system would be made using mathematical programming methods. Unfortunately, this is usually impossible due to the high complexity of the fabrication setting. Therefore, scheduling is usually done with a time horizon of one shift, one day, or one week and serves as a guideline for production over this period. Scheduling is considered to be more global in nature than other lower level methods and takes into account such information as planned tool maintenance, product mix changes, tool utilization, queue sizes, etc. The scheduling system may also contain a rescheduling or reactive scheduling portion that takes action in response to sudden changes in the system. When a disturbance occurs, a rescheduling algorithm generates an entirely new plan for the fab [36] while a reactive scheduler makes necessary alterations to the existing plan [34, 35, 37].

Finally, at the lowest level, the dispatching system handles the movement of wafers in real-time. Typically, a system of rules is established that determines which lot in a given tool's queue will be run when that tool becomes available. These systems will

be discussed in more detail later after a description of the manufacturing process from a scheduling point of view.

From the perspective of a single wafer, the production process can be viewed as a flow shop [57] where the wafer moves through an ordered series of machines until it reaches completion, but this is an oversimplification. Previously, it was stated that the number of layers needed to make a finished device is so large that the total processing steps necessary for its completion can reach well into the hundreds. Also, the tools involved in the manufacturing of IC's are incredibly expensive, and floor space within the fab is valuable and limited. Consequently, it is impossible to dedicate a single tool to every step of the process for a given product.

Fortunately, most layers of the devices are similar and can be made through the same series of operations. This results in recirculation of each wafer through the line where it visits the same (or same type of) machine multiple times before it is finished. A cyclic process of this type is referred to as reentrant [3, 4]. Additionally, many fabs are high mix, i.e., they make many different products, each with their own unique set of processing steps and/or tool settings. Also, fabs have multiple identical tools at each step of the process (i.e., parallel processors) to allow for sufficient capacity and to offer a degree of redundancy in the event of breakdowns. Because the semiconductor fab is characterized by multiple types of jobs visiting a given number of stations with several identical machines in a reentrant fashion, it is sometimes viewed as a flexible job shop with recirculation [30, 58]. Alternatively, other researchers view the products as being roughly equivalent and consider the fab to be a flexible flow shop with reentrancy [6, 59].

Large quantities of process steps and reentrancy are just a few characteristics of the semiconductor line that greatly complicate the scheduling task [5, 6]. Others include:

1. Multiple equipment types: Some equipment processes one wafer at a time, while others process entire lots (usually 25 wafers), and still others can process several lots together in a batch. Additionally, many batch tools do not require the batch to be the same size for every run so that finding the "best" batch size is an open question. Some pieces of equipment also have large sequence-dependent setup times when changing from processing one type of product to another.
2. Random machine downtime: Many of the tools used for semiconductor production rely on cutting edge technology that is not completely (or even well) understood, and the error tolerances for the tiny chip features are very tight. This combination often causes tools to be taken out of service for producing products that fall outside of quality specifications. The time to repair varies widely depending on how long it takes plant personnel to troubleshoot the problem (e.g. the problem might be fixed by recalibration or it might require replacement of worn parts). Preventive maintenance (PM) is also performed at regular intervals based on how long the tool has been in production or how many wafers it has processed. These rules are somewhat arbitrary and another area of research lies in determining the best time to perform PM.
3. Shared facilities: It is not uncommon for a production line to be used for other purposes such as development or pilot runs of new products or test and qualification lots which are used to by control engineers to gage the current states of the tools. These lots are not accounted for in the planning stage and therefore, serve as disruptions to the scheduler [8].
4. Demand based due dates: Increasingly, the semiconductor industry has transitioned from a make-to-stock environment to a make-to-order environment [5, 30]. The desire to reduce inventories and the advent of application-specific

integrated circuits (ASIC) has increased the importance of meeting due dates established by the production planning entity. The need to meet due dates greatly complicates the scheduling task.

As mentioned earlier in this section, the complexity of the fab environment and the large amounts of uncertainty involved in running the production line make it impractical to attempt a detailed optimization of the system for scheduling purposes. Also, owing to the uniqueness of the system, scheduling approaches from other industries are not easily transferred to semiconductor fabrication [6].

In the semiconductor industry, it is more common for manufacturers to focus shop floor control efforts on the dispatching system than on higher level, supervisory scheduling systems. As lots move through the fab, they will invariably end up waiting in queue for a tool that can perform their next processing step. When a tool of the correct type becomes available, the dispatcher sends the lot with the highest priority to that tool for processing. Industrial surveys [7, 8] indicate that simple, localized rules such as first-in-first-out (FIFO), as well as more complex, global rules that incorporate due dates, starvation avoidance for bottleneck machines, and combinations of two or more rules [32], are employed. Very few of the approaches studied in academia, such as those based on rescheduling, robust scheduling [34, 35, 36], and control theory [28, 29], have been implemented in industry. One finding that seems consistent in both industry and academia is that the policies implemented by the order release system to the shop floor can have a high impact on dispatch system performance [30]. For this reason, order release and dispatching are sometimes lumped together into a single procedure referred to as workload control [6]. For further reading on fab related scheduling, several review papers have been written which explore this area of research [5, 6, 31, and 33].

## 2.2 RUN-TO-RUN CONTROL FOR SEMICONDUCTOR MANUFACTURING

While process control practices have been well established for many decades in other, more mature businesses such as the chemical, refinery, and aerospace industries, they have only come into widespread use in the semiconductor industry over the last fifteen to twenty years [52]. Several factors inherent to the unit operations used in the semiconductor fabrication have prevented the implementation of many of the established control technologies that are used in other industries. These factors include [9, 10, 11, 52]:

1. It is the nature of many of the processes to induce disturbances. The manner in which some tools age can cause a drift disturbance or cause process data to be autocorrelated. The rate of drift may also vary due to different materials being deposited or different products being run on the same tool. Eventually, the tool may drift too far from the acceptable operating region and maintenance will be necessary; the sudden change in the tool state caused by the maintenance activity appears as a step disturbance to the control system. Such problems occur in deposition tools where the material deposited on the wafers is also deposited on the walls of the chamber and builds up over time causing a drift disturbance. Eventually, the accumulated material must be cleaned off causing a step disturbance [11].
2. The unit operations employ highly complex technologies. Often, the chemical and/or physical phenomena on which process tools are based are not well understood from a theoretical standpoint. Because of the rapid pace of development in the industry, technologies that have been proven to work empirically are deployed to production even though comprehensive theoretical models of their underlying chemistry and physics are unavailable. In other cases

where detailed models of a process are available, they are so highly nonlinear that their solution time is much larger than the sampling time of the process. Accordingly, it is often impractical to use modeling from first principals for control purposes. This lack of process knowledge coupled with the fact that process variables are often highly correlated can make the selection of variables for any control algorithm difficult [1, 9, 52].

3. The availability of measurements is limited. Frequently, important process and quality variables that would allow for the use of feedback control during a given processing step are not available because appropriate measurement technology does not exist or cannot be implemented in situ due to the harshness of the processing environment. Typically, the tool parameters (e.g., gas flow rates or chamber pressures) are kept at constant setpoints by individual PID controllers. The collection of PID setpoints is known as the process recipe. In general, the recipe is given before a run begins and may change based on some predetermined timing for a process but not in response to any real time feedback of the on-wafer state. Additionally, even post-process measurements of wafers made ex situ can be infrequent because it is too costly or time consuming to obtain them on a more regular basis [1, 9, 12, 52].

To contend with the difficulties listed above, practitioners and researchers developed a paradigm known as run-to-run (R2R or RtR) control. Also called run-by-run (RbR) control or supervisory control, R2R control combines the ideas of engineering process control (EPC) with those from the area of statistical process control (SPC).

SPC is used throughout the semiconductor industry as a means of monitoring various processes. SPC uses a set of monitoring charts based on the mean and variance of process variables and outputs. In the early years of semiconductor process control,

SPC was used to make infrequent, manual corrections in an ad hoc manner to the process recipe in order to compensate for disturbances that moved the process far away from the statistically measured "normal" operating region. When applied to a drifting process in which the output steadily moves away from the target and outside of the control limits, two corrective actions are possible. The first choice is to adjust the recipe to bring the process back into the acceptable region. Alternatively, if it is not possible to correct the process through recipe adjustment because of physical constraints on the inputs, then a maintenance operation is necessary to bring the tool back to its original physical state (e.g., cleaning a deposition chamber to remove build up).

Unfortunately, several assumptions that underlie the basic use of SPC made it a poor fit as the primary source of control for many semiconductor processes. Firstly, SPC is designed for stationary processes with uncorrelated data, but the propensity of many semiconductor processes to drift as a function of run number or experience sudden shifts makes this assumption incorrect [14]. Secondly, it is assumed that sources of error in the process can be isolated and removed and that input adjustments are infrequent due to their high cost. For semiconductor tools, the disturbances mentioned previously are inherent to the process and cannot be removed, and as machines have become more automated, input adjustments are quite easy. For early practitioners who were trying to apply control in the ad hoc manner mentioned above, the number of SPC alarms and manual adjustments became difficult to manage, and it was apparent that new approaches to solving the control problem were necessary. Thus the principle of automatic feedback control was introduced from the area of EPC, and a supervisory controller was created for the correction of output errors from one run to the next.

In the late 1980's and early 1990's, several papers from outside of the semiconductor industry studied the relationship between SPC and automated process

control (APC), for example, see [64] and subsequent responses in the same issue. This research led to the development of so called algorithmic process control (ASPC), which integrates a feedback controller with a statistical monitoring system [65, 66]. In short order, similar approaches were applied to semiconductor processes; the concept of feedback was incorporated into existing SPC monitoring schemes and the first run-by-run controllers were developed.

Early R2R controllers used the strengths of feedback control to offset the effects of slow process drifts and other smaller disturbances in what was called the "gradual mode." SPC was applied to detect the point at which the process had drifted too far from the controllable region and at this point a "rapid mode" was used to make larger corrections [14, 67]. Eventually, the SPC portion of advanced process control (APC) became more strongly associated with the realms of fault detection and classification (FDC) and tool maintenance scheduling. At the same time, most research in the area of run-to-run control has focused on compensating for drifting disturbances and smaller shifts not associated with maintenance events by taking post-processing measurements and using them to adjust the recipe on the tool before the next run.

### 2.2.1 EWMA and its Extensions

#### *2.2.1.1 EWMA Control*

The basic structure of the R2R controller is divided into three parts: a process model, an observer, and a control law. The early R2R controllers of Sachs et al. [14, 64] employed a linear process model, with an exponentially weighted moving average (EWMA) filter for the observer and a dead beat control law. Because this set up has proven to be both simple to use and quite robust, it is used quite widely in the industry.

In general, the process is considered to be pure gain system which is described well by a linear model of the form

$$y_k = \beta u_k + x_k + \varepsilon_k \tag{2.1}$$

where $y$, $u$, $\beta$, $x$, and $\varepsilon$ represent the output, input, process gain, intercept and noise, respectively. Methods such as response surface modeling (RSM) or linear regression are performed on data taken from designed experiments during qualification of the tool in order to find an estimate of the gain; this estimate is denoted as $b$. Often, drift disturbances are considered to be slow enough that the gain is left as a constant between qualification events and the disturbances and noise are lumped into the estimate of the intercept term. Thus, the resulting estimate equation is

$$\hat{y}_k = bu_k + \hat{x}_k. \tag{2.2}$$

The recursive calculation employed to generate EWMA charts in the realm of SPC is used as a filter to give updated estimates of the disturbance, $b$. The EWMA equation is of the form

$$\hat{x}_{k+1} = \lambda(y_k - bu_k) + (1-\lambda)\hat{x}_k \tag{2.3}$$

where $k$ represents the run index, $y_k$ is the measured value of the output from the current run, $u_k$ is the recipe used for the current run, $\hat{x}_k$ is the current disturbance estimate that was calculated from the previous run, $\hat{x}_{k+1}$ is the one-step-ahead disturbance estimate made from the current run's data and will be used to calculate the next run's recipe, and $\lambda$ is the EWMA tuning constant where $\lambda \in [0,1]$. Note that a value of $\lambda$ equal to zero will give a disturbance estimate equal to the last value, so smaller values make for a less sensitive filter (i.e., more robust to noise). Conversely, values of $\lambda$ closer to one place more emphasis on the recent measurement and give a faster acting filter.

Rearranging equation (2.3) and substituting equation (2.2) allows the update for the intercept to be written in terms of the output estimate error (also called the output residual) and yields the following equation (see Appendix B-I for the derivation):

$$\hat{x}_{k+1} = \hat{x}_k + \lambda(y_k - \hat{y}_k). \tag{2.4}$$

From this form, it has been shown that the EWMA R2R controller is equivalent to an integral controller with a measurement delay of one [14].

Once the updated estimate for the disturbance is known, it can be applied to the control law in order to get the new values for the process inputs that will drive the next run to target. The simplest and most aggressive control law is the dead beat controller which is just an inversion of the process model. Here, the quality target for the output, $T$, is substituted for the output in (2.3) and the equation is solved for $u_k$, the new recipe to be used for the current run as shown in equation (2.5).

$$u_{k+1} = \frac{T - \hat{x}_{k+1}}{b}. \tag{2.5}$$

A summarized version of the EWMA run-to-run control algorithm can be found in Appendix B-II.

Because of its ease of implementation and wide usage in the semiconductor industry, the EWMA controller has been extensively studied. Ingolfsson and Sachs [68] give stability conditions and formulas for mean output error as functions of EWMA weighting, plant and model gains, and disturbance parameters for both single input/single output (SISO) and multiple input/single output (MISO) systems. These results are extended to include more types of disturbances by Del Castillo [69] and multiple input/multiple output (MIMO) systems by Tseng et al. [71]. Adivikolanu and Zafiriou [11] use the equivalence of EWMA control to internal model control (IMC) in order to discuss the robustness of EWMA controllers in the presence of measurement delays and

model errors; both SISO and MISO systems are explored. Finally, Good and Qin [70] explore the stability criteria for MIMO EWMA controllers operating in systems where measurements are delayed by more than one run.

#### *2.2.1.2 Extensions to EWMA with Static Models*

One shortcoming of EWMA controllers is that they will cause an offset of the output from the target in the presence of a large drift or ramp disturbance. This problem was first addressed by Butler and Stefani [15] in their development of the predictor corrector controller (PCC). The PCC splits the intercept term into two terms: one for the usual disturbances and one specifically for the drift. Each of these disturbance terms has its own EWMA estimator; because the controller now employs two filters, it is often referred to as a double EWMA (DEWMA) controller. The authors apply the PCC to an etch process and demonstrate improved control over previous approaches for the drifting process. In the EWMA robustness paper mentioned previously [11], the PCC is also studied and an IMC formulation is given.

Chen and Guo [16] reformulate the PCC into what they call the age-based double EWMA controller, which gives a larger control region and allows for more straight forward usage. Del Castillo [75] studies the stability characteristics of the DEWMA controller and gives a theoretical method for optimal selection of the tuning parameters for both SISO and MISO systems. In general, the author finds that smaller values of the weights for the two EWMA filters give better asymptotic stability (i.e., long run output performance) but can lead to sluggish transient performance (i.e., larger errors at the beginning of a run). Tseng et al. [76] find similar results to Del Castillo [75] but for a more general class of disturbance models (i.e., the results in [75] represent a specific case of the general results given in [76]). As a follow up to [75], Del Castillo and Rajagopal [72] extend the usage of DEWMA controllers to MIMO systems and again give stability

conditions and recommendations on how to choose tuning parameters. Good and Qin [77] study a DEWMA controller where the weights for the two filters are identical, and they calculate the stability regions for the controller in the presence of model mismatch and measurement delay for both SISO and MIMO systems.

Building upon the logic behind DEWMA controllers which add a drift term to the process model and estimate it with an additional EWMA filter, Fan et al. [80] add an output autocorrelation term in their process model and a third EWMA filter. The resulting triple EWMA (TriEWMA) controller is developed only for SISO systems and is shown through extensive simulations to outperform both DEWMA and ST controllers. The largest improvements are shown for processes with colored noise and high positive autocorrelation of the output values. The authors also provide guidelines for selection of the three EWMA parameters.

Rather than adding more disturbance terms to the process model and additional EWMA filters to generate estimates of these new terms, Smith and Boning [81] instead alter the form of the input/output model. Rather than relying on the linear gain model of (2.1), the authors employ an artificial neural net (ANN) model to map the nonlinear relationship between inputs and outputs. Essentially, the $bu_k$ term of the controller model (2.2) is replaced by the ANN model, and the rest of the controller functions as before. The resulting controller is found to be robust in the presence of model errors and noise and can maintain stability for second order systems where as an EWMA controller with a linear model cannot.

Wang et al. [82] construct R2R controllers based on the recursive least squares (RLS) algorithm with a single forgetting factor to discount older data. The disturbance is represented by a polynomial whose order is determined from process data, and the parameters of the polynomial are found by minimizing a cost function that is the

weighted summation of the process innovations. The authors show that both EWMA and DEWMA can be derived from the RLS framework and that in these cases, the RLS formulations have the same asymptotic behavior. Simulation examples are then used to compare the performance of the RLS controllers against EWMA, DEWMA, and IMC controllers for three different disturbance types and in systems with measurement delays. Finally, an application to the shallow trench isolation (STI) etch process is shown.

#### *2.2.1.3 Adaptive Methods*

As opposed to the approaches above that employ a static process model and constant filter parameters, other researchers have developed alternative methods that reject disturbances by adapting the gain of the linear model as the process is run. Qin et al. [79] begin with a linear model for the preheat recipe of a rapid thermal annealing (RTA) process and develop an adaptive algorithm for the gain of the process model. The gain is updated at each run by using a linear least squares regression of available process data. A control chart is used to monitor for drifts in the process; using a set of Western Electric rules, the decision on whether to alter the recipe for the next step is made.

Castillo and Hurwitz [10] suggest several self-tuning (ST) controllers in which a recursive least squares algorithm is utilized to estimate model parameters associated with process gain, autocorrelation of outputs, and deterministic drifts. The resulting ST controllers are found to be more robust to a variety of disturbances than the typical EWMA controllers in the case of a SISO system. This approach is later extended to the MIMO case in [73] and is the predecessor to the Optimizing Adaptive Quality Controller (OAQC) developed by Del Castillo and Yeh [78].

Hankinson et al. [74] study the control of reactive ion etching (RIE) and develop a methodology called knowledge-based interactive run-to-run control (KIRC). The KIRC algorithm takes experimental data from designed experiments and models them using

neural networks (NN) so that missing operating points can be filled in by interpolation. A decision tree algorithm is then used to classify all response surface data (both experimental and model generated) so that incoming process data can be put into a particular group and decisions on how to adjust the next run can be made.

As opposed to adapting the input/output model as the process is run, some researchers propose to adapt the weights of the EWMA filter in order to make the resulting controllers more responsive to disturbances. In the paper by Smith and Boning [17], the authors choose to adapt the weights of the MIMO EWMA controller rather than the gain (or other parameters) of the process model using an ANN model. Through Monte Carlo simulation, the authors develop a mapping from the disturbance values to the optimal EWMA tuning parameters. The ANN portion of the controller is then able to take disturbance estimates from the EWMA filter and determine the optimal EWMA weighting that should be used given the current state of the process.

Patel and Jenkins [18] create an adaptive optimization scheme which seeks to adjust the EWMA weight over the series of all runs in order to minimize the limit of the mean squared error of the output as number of runs approaches infinity. With this objective in mind, they design a recursive formulation based on an approximation of the signal to noise ratio that will adjust the EWMA parameter at each run. The system is applied to an industrial lithography process and shown to perform well while reducing the amount of engineering time spent retuning the controllers and monitoring their performance.

#### *2.2.1.4 Model Predictive Control (MPC) Approaches*

Developed in the last 30 years, MPC is a methodology that has found wide acceptance in the chemical and petroleum processing industries. In their MPC review paper, Qin and Badgwell [23] list over 4500 instances of MPC controllers used in

industrial applications as of 1999, but that number is known to have increased greatly since then. Mullins et al. [83] first suggested applying MPC to run-to-run control because MPC can easily handle MIMO systems with input and output constraints. Also, with MPC, constant time delays and dynamics such as a drift can be incorporated into the process model itself. Mullins' paper, which gives a brief example of a CMP process, and subsequent work by Campbell [25] (also on CMP) and Bode [26] (photolithography overlay) use a linear model predictive control (LMPC) formulation originally proposed by Muske and Rawlings [20]. The general form of LMPC can use a process model like the one in (2.1) but presents it in state space form.

$$x_{k+1} = Ax_k + Bu_k \tag{2.6}$$

$$y_k = Cx_k. \tag{2.7}$$

In equations (2.6) and (2.7), $x_k$, $u_k$, and $y_k$ are vectors representing the states, inputs, and outputs of the system, respectively. $A$, $B$, and $C$ are matrices of parameters that define the system; and for the R2R application, $k$ represents the run index rather than a time index.

Often, a Kalman filter is used to give updated estimates of the states which cannot be measured directly. In order to use the Kalman filter theory, the state space model is reformulated as follows:

$$x_{k+1} = Ax_k + Bu_k + G_w w_k \tag{2.8}$$

$$y_k = Cx_k + v_k \tag{2.9}$$

where $w_k$ and $v_k$ represent process and measurement noise respectively and are assumed to be zero-mean, white noise signals with covariances of $Q = \mathrm{E}\left(w_k w_k^T\right)$ and $R = \mathrm{E}\left(v_k v_k^T\right)$, respectively. Additionally, the noise terms should be uncorrelated with one and other and also with the initial condition for the state, $x_0$, where $x_0 \sim (\bar{x}_0, P_0)$.

The equivalence of the state space system to the process model of the EWMA example can easily be shown. If $A = C = 1$ and $B = b$, (2.8) and (2.9) can be

combined to give (2.2) with $\varepsilon_k = w_k + v_k$. As Mullins et al. [83] indicate, the fact that the noise term can be split into contributions from the process and the measurements gives MPC another advantage over other forms of run-to-run control.

The estimates for states, $\hat{x}$, at the next step, *k+1*, given information up to the current step, *k*, are given by the equation

$$\hat{x}_{k+1|k} = Ax_{k|k-1} + Bu_k + L(y_k + C\hat{x}_{k|k-1}) \tag{2.10}$$

where $L$ is the Kalman filter gain. If the covariances of the noise terms and the initial condition of the state can be assumed zero mean Gaussian, then the tuning parameter, $L$, has an optimal value that is calculated by a set of discrete time Riccati equations (see Appendix B); otherwise, it is found empirically. Estimation by Kalman filtering is discussed further in Chapter 3.

The final part of LMPC is the control law which consists of an objective function that is minimized by finding an input trajectory, $u^N$, over a control horizon, $N$. The objective function is normally a quadratic equation of the form

$$J = \min_{u^N} \sum_{j=0}^{\infty} (y_{k+j}^T Q y_{k+j} + u_{k+j}^T R u_{k+j} + \Delta u_{k+j}^T S \Delta u_{k+j}) \tag{2.11}$$

where $Q$, $R$, and $S$ are weighting matrices and $Q$ and $S$ are symmetric, positive semidefinite while $R$ is symmetric, positive definite. These matrices can be used to weight the importance of the various terms so their magnitudes relative to each other are more important than their absolute values. The first term of $J$ minimizes output deviation, the second term minimizes input deviation, and the third term minimizes the rate of change of the input.

Equations (2.8) and (2.9) are combined with the following inequalities to define the quadratic program:

$$
\begin{aligned}
u_{\min} \le u_{j+k} \le u_{\max} \qquad & j = 0, 1, \ldots, N-1 \\
y_{\min} \le y_{j+k} \le y_{\max} \qquad & j = j_1, j_1+1, \ldots, j_2 \\
\Delta u_{\min} \le \Delta u_{j+k} \le \Delta u_{\max} \qquad & j = 0, 1, \ldots, N.
\end{aligned}
\tag{2.12}
$$

As can be seen from (2.12), the MPC formulation can be used to restrict the inputs, outputs, and the size of the input moves to user or system defined ranges. Implementing these types of min/max constraints is much more difficult to accomplish with the typical EWMA controller.

Muske [84] gives solution methods for the above optimization problem for a number of conditions including stable and unstable systems. Before each new run, the quadratic program is solved to find $u^N = \begin{bmatrix} u_k & u_{k+1} & \ldots & u_{k+N-1} \end{bmatrix}^T$ but only the first control move, $u_k$, is implemented. At the next time step, the entire process is repeated to find the new optimal trajectory, and again, only the first move is implemented. For the interested reader, several reviews and tutorials on the subject of MPC are available in the literature [21, 22, 23, 24].

### 2.2.2 Multi-tool, Multi-product Process Control Methods

Due to the fact that processing tools used in fabs are very expensive, it is expected that their utilization be maximized. It is also true that many fabs are of a high mix nature and therefore, are not able to dedicate tools to the processing of only one type of product or processing layer. An example of this is "mix and match" processing [45, 85] in the photolithography area. Lithography tools (or steppers) are the most expensive machines employed in the fab and also have one of the longest processing times. Therefore, it is common to run many different types of products on the same stepper in order to reduce bottleneck effects at these machines. Because all tools of a similar type do not behave in exactly the same manner and because making different products on a tool can give

different results, the process control in a high mix environment is more complex than the simple single product/single tool scenario discussed in most R2R control papers.

The problem of running multiple products and processes on tools was first addressed by Miller [38]. He begins by identifying different types of disturbances that occur in semiconductor processing and classifying them into three categories.

1. Tool based disturbances such as drift disturbances caused by buildup on etch chamber walls. These will affect all products processed on the tool.
2. Disturbances caused by inconsistent quality of raw materials. The example given is of photoresist used in the photolithography step; this type of disturbance affects all wafers that use the particular raw material.
3. Product based disturbances such as the incoming layer thickness and uniformity of wafers entering a CMP step. These disturbances are particular to a given type of product and should be rejected in such a manner that they do not affect other product types. Feedforward measurements can be used to compensate for incoming product specific inconsistencies, but this requires additional modeling effort.

Miller also discusses the complications associated with design of R2R control schemes for multi-product and -process systems and suggests several different approaches along with advantages and disadvantages of each. He first discusses three methods that can be classified as multi-product/tool extensions of the methods reviewed in Section 2.2.1 while the final control methodology discussed is termed cooperative control. Section 2.2.2.1 will discuss the first three approaches and Section 2.2.2.2 will discuss cooperative controllers.

#### *2.2.2.1 Threaded and Composite Controllers*

The first two control methods discussed in [38] fall into a category known as threaded control. The first approach uses independent controllers which model the individual relationships between each combination of product and tool in the fab—these combinations (along with some other information such as processing layer or previous step for example) are often called processing threads [86] or contexts [53, 54]. In a large system with many tools and products, the independent controller approach requires the management of a substantial number of different control models.

One advantage of independent controllers is that the disturbance estimate for one product on a machine is unaffected by disturbances that are attributable to other products. Unfortunately, the same is not true for tool based disturbances because they will affect all products. Because the disturbance estimate of each model is actually a combination of effects from the tool and the product in the context, information on the tool related portion is lost when the context is changed (i.e., when different products are run consecutively on the same machine).

A simple example of two products running on a tool with a gradual drift disturbance is used in [38] to demonstrate the shortcomings of independent controllers. A plot of the system with a product switch can be seen in Figure 2.2 (please note that no noise is injected into the system so that the trends are easier to see). If Product 1 is run for an extended period time, its controller is able to compensate for the tool drift disturbance by incorporating it into the updated disturbance estimate after each run. On the other hand, the model for Product 2 does not receive updated information on the growth of the tool disturbance while Product 1 is being run. As can be seen in Figure 2.2, this results in the first run of Product 2 (run number 26 in the figure) being far from target and the drifting tool disturbance appears as a step disturbance to the controller for

Product 2. Because the controllers for the products are tuned to compensate for the drifting disturbance, there are several off target runs of Product 2 before the controller is able to recover the process and completely reject the large step disturbance.

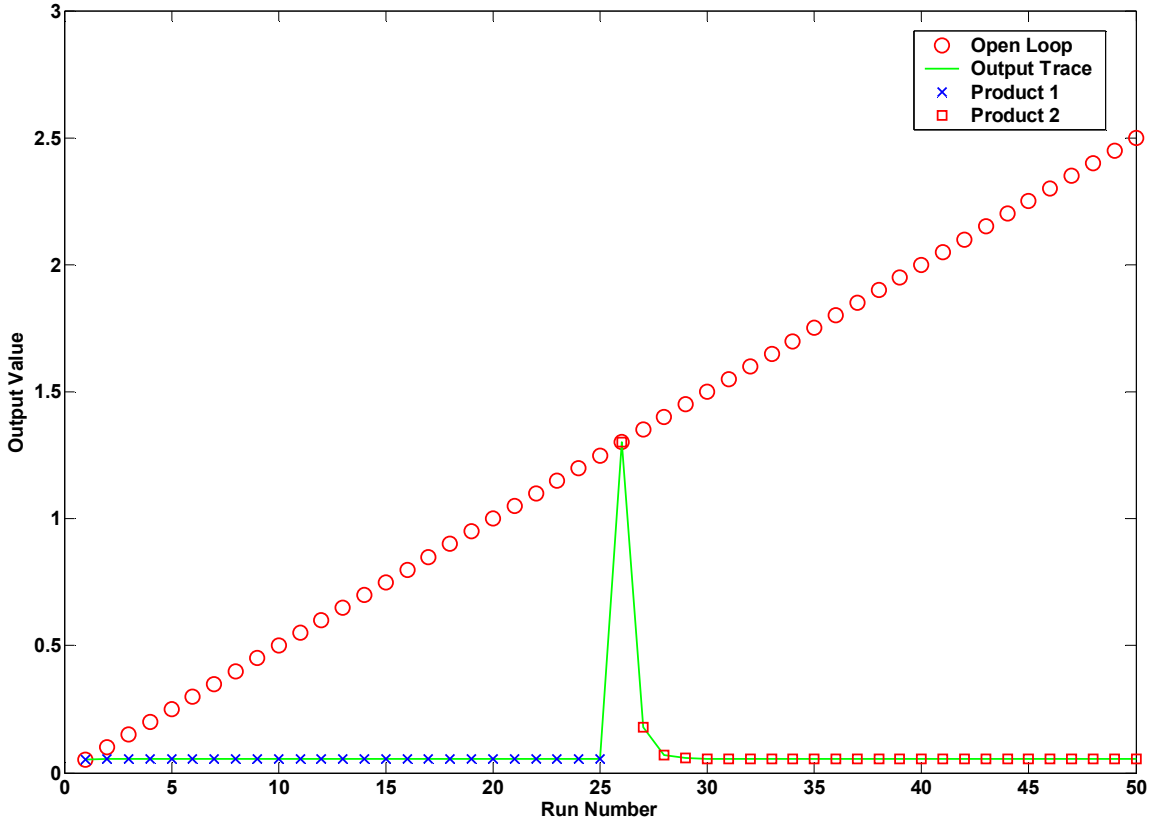


Figure 2.2: Example of a controller switch in the presence of a gradual disturbance [38].

In order to reduce the number of switches between products, Miller suggest the idea of grouping controllers. This method is similar to independent controllers except that different contexts are grouped if they demonstrate similar performance to one another. Grouping offers several advantages. Fewer controllers are necessary because one model can be used for several different contexts rather than every context needing its own model. Additionally, it will be easier to update the models with fresh information because measurements for all contexts in the group can be used. The main disadvantage

to grouping is choosing which products to gather together to form the various groups (i.e., how close is close enough when comparing the similarity of two or more contexts).

Finally, there is the limiting case in which the number of groups goes to one (i.e., all contexts will be controlled using the same model), which is known as composite control. Having a single controller eliminates problems associated with context switching, but has other drawbacks. It is highly unlikely that a single global model can be developed that will be able to sufficiently account for the all of the interactions between various tools and products. The other problem results from various processes having different setpoints. A controller that can handle setpoint changes may not be as adroit at rejecting disturbances and will suffer from reduced performance.

Multi-product systems with switching like the one illustrated above were studied in depth by Zheng et al. [87]. Using a two product, one tool system like the one illustrated for independent controllers above, two different EWMA based controllers are compared and their stability properties are examined. Analytical results for the output and asymptotic mean squared error (AMSE) are also given. For a more detailed summary and discussion of the results in [87], see Appendix C, which also includes equations and figures.

Zheng et al. first discuss what they call "tool-based" control, which is a blend of independent and composite control. Separate input/output models like equation (2.2) and control models like equation (2.5) are defined for each product, but the noise estimate, $b$, is shared across all products. In other words, there are as many gain estimates, $a$, process models, (2.2), and control laws, (2.5), as there are products, but there is only one EWMA estimator and only one disturbance estimate, $b$. Except for the trivial cases where either the mismatch between $\beta$ and $b$ is zero or the disturbance is nothing but stationary, white noise, the tool-based controller is found to be unstable.

As it relates to the discussion in [38], tool-based control with different plant/model mismatches can be thought of as the case where two controllers are grouped but are not actually similar. In this case, the same process, controller, and estimation models are used for both products, but their underlying behaviors (plant models from a simulation perspective) are actually different. Therefore, according to the results in [87], grouping products whose behaviors are not similar can lead to long run instability. From a practical standpoint, the effect of this instability may be insignificant if the products are sufficiently similar and the number of runs is low enough.

The second type of control studied in [87] is labeled "product-based" control which is the same as the independent control approach discussed in [38]. Product-based control is shown to be stable for a wide range of operating conditions, but as shown by Miller, long break lengths between runs of given product in the presence of a non-stationary process can lead to large offsets in the output after a switch. As would be expected, it is found that products that make up a larger portion of the product mix will suffer less from disturbances created by switching because they have a shorter break length (i.e., time between runs) than other products that comprise a smaller percentage of the product mix. When dealing with a drifting process, Zheng et al. suggest tuning the controllers for low running products more aggressively (i.e., setting $\lambda$ closer to one than to zero) than the controllers for high running products.

#### *2.2.2.2 Cooperative Controllers*

The final type of controller suggested by Miller is called a cooperative controller. The idea behind such a control scheme is to share information across tools and products so that the disturbance estimate for each context will reflect the most up to date information. Additionally, many fabs are now run in a high-mix fashion which greatly increases the number of context items in one or more categories (e.g., tools, products,

layers, etc.). In a threaded control system, the number of states to estimate increases as the product of the number of context items in each category, but in a non-threaded control system, the number of states grows only as the sum of the number of context items in each category. For example in a system with three context groups labeled tools, products, and layers with each group having three, four, and two items, respectively, a threaded control system must estimate 24 different states, while the non-threaded system estimates only nine. Therefore, as the size of the system increases, the estimation effort required to control the system is greatly reduced in a non-threaded (i.e., cooperative) control system.

Campbell [25] applies the MPC formulation described previously to the CMP process. Because wafer surfaces of different product types have different topographies, the polishing rate varies from one product to another even though the tool being employed remains constant. To account for the difference between polish rates, a unique topography factor is assigned to each product being run on the tool. Campbell then uses an observer scheduling technique in which the topography factor used in the observer is changed according to the product being run. For the single tool case, the use of blank oxide wafers to identify the polish rate independent of the product being run is drastically reduced because this information is now shared across controllers for different products.

Patel et al. [90] develop a R2R control scheme for multiple CMP tools and multiple products. First, qualification (also called qual) wafers are defined as unpatterned wafers with a blanket (i.e., uniform) oxide film on their surface. A sheet film equivalent (SFE) for each product is then defined as the amount of oxide that would be removed from a qual wafer in the same amount of time it would take to polish the product wafer from its incoming (i.e., pre-polish) target thickness to its outgoing (i.e., post-polish) target thickness. Additionally, an effective blanket polish rate is defined for each tool.

Through a series of control loops, the SFE for each product and blanket polish rate for each tool are estimated so that the polish time for any incoming wafer on any tool can be calculated before the run is started. The system can also account for drift disturbances, metrology delays and feedforward measurements of the incoming oxide thickness. Simulations and data from an industrial application show that the control scheme has distinct advantages over the previously used EWMA controller with static product dependent factors (i.e., independent controllers).

Similar to the work by Patel et al. [90], other cooperative controller approaches begin by assuming that the disturbance for a context can be divided into contributions from the various parts of the context. One of the earliest examples of this approach is the just-in-time adaptive disturbance estimation (JADE) run-to-run controller proposed by Firth [88]. JADE uses a controller process model similar to the one in equation (2.2) except the intercept term, $\hat{x}_k$, is a sum of contributions from several different sources. The equation is now

$$\hat{y}_k = bu_k + \hat{x}_{tot,k} \tag{2.13}$$

where,

$$\hat{x}_{tot,k} = \sum_i \hat{x}_{i,k}. \tag{2.14}$$

The example for a photolithography overlay process is given with the resulting equation for the disturbance being

$$\hat{x}_{tot,k} = \hat{x}_{tool(k),k} + \hat{x}_{reticle(k),k} + \hat{x}_{ref.tool(k),k} + \hat{x}_{ref.reticle(k),k}. \tag{2.15}$$

In this case, the disturbance is a function of the tool, reticle, reference tool, and reference reticle used to process the product. Once all of the context items are identified, the full set of equations like (14) can be summarized in matrix form:

$$x_{tot} = A_0 x \tag{2.16}$$

where $x_{tot}$ is a vector of total offsets calculated from the measured inputs and outputs to the process for each context, that is,

$$x_{tot,i} = y_i - bu_i \tag{2.17}$$

for the $i^{th}$ context. The vector $x$ has entries that are states which represent the disturbance contribution of each individual context item (e.g., tool, reticle, product, etc.). The matrix $A_0$ consists of ones and zeros and ensures the proper summation of individual disturbance states to form the total disturbance state for a thread. The example of a two tool, two product system is given as an example. In this case, there are $2 \times 2 = 4$ possible contexts and the full matrix equation looks like

$$\begin{bmatrix} x_{tot_1} \\ x_{tot_2} \\ x_{tot_3} \\ x_{tot_4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{tool_1} \\ x_{tool_2} \\ x_{prod_1} \\ x_{prod_2} \end{bmatrix}. \tag{2.18}$$

The JADE observer uses a recursive least squares algorithm [89] to calculate the state updates. The least squares objective function seeks to minimize the square of the residuals:

$$J = \tfrac{1}{2}\left(c - A_0 c_{tot}\right)^T \left(c - A_0 c_{tot}\right). \tag{2.19}$$

A normal equation gives the optimal solution and has the form:

$$c = \left(A_0^T A_0\right)^{-1} A_0^T c_{tot}. \tag{2.20}$$

In order to solve the normal equation, the term $\left(A_0^T A_0\right)^{-1}$ must be invertible which implies that $A_0$ must be full rank. Unfortunately, $A_0$ is always rank deficient as can be seen in the example equation (17). To overcome this problem, Firth augments $A_0$ with an identity matrix and the recursive equation for the observer is formulated as follows:

$$\begin{bmatrix} A_0 \\ I \end{bmatrix} c_{k+1} = \begin{bmatrix} c_{tot} \\ c_k \end{bmatrix} \text{ or } \quad Ac_{k+1} = \tilde{c} \tag{2.21}$$

for brevity.

After the problem of rank deficiency of $A$ is addressed, Firth then reformulates the least squares objective function and adds a weighting matrix, $Q$, which contains tuning weights similar to those in EWMA control for the discounting of older data. The matrix $Q$ also contains parameters that allow the user to weigh the relative importance of different context items against one another. The new weighted recursive least squares objective function is

$$J(c_{k+1}) = \tfrac{1}{2}\left(\tilde{c} - Ac_{k+1}\right)^T Q\left(\tilde{c} - Ac_{k+1}\right), \tag{2.22}$$

which is solved with the following normal equation

$$c_{k+1} = \left(A^T Q A\right)^{-1} A^T Q \tilde{c}. \tag{2.23}$$

Through simulation the JADE algorithm is shown to outperform a threaded EWMA system for both photolithography overlay and CMP control. The improved control is due to the fact that the JADE observer is able to share information across tools and products. One other advantage occurs when a tool state needs to be reset after a maintenance event or when a new product is added to the system. During these events, the new states can be calculated without affecting other existing states. On the other hand, a threaded EWMA system must recalculate all disturbance states that contain a contribution from the reset tool or the new product.

Applying a process model similar to the one used by Firth [88] in which the disturbance is split into contributions from a context's tool and product, Pasadyn [41] develops an estimation format based on state space modeling and Kalman filtering. He

also discusses ways in which this type of estimation might be useful in conjunction with scheduling systems. This system is the basis for the work discussed in Chapters 3 and 4.

In order to insure observability of the state space system, Pasadyn utilizes qualification runs which give an independent measurement of the tool state. As discussed by other authors [25, 90], qual runs should be minimized because they do not produce any sellable products but have an associated cost due to the purchase of additional wafers and time spent by tools running non-product wafers. In order to address the problem of needing qual runs to ensure system observability, Jang and his coworkers [91–93] present approaches based on analysis of variance (ANOVA) and analysis of covariance (ANCOVA) models. By using ANOVA and ANCOVA model forms, observability of the system is assured because of the equality constraints placed on the states by these models. The authors find that these systems provide not only good control but also estimates of the individual tool and products states which mimic the behavior of the actual states. Finally, they demonstrate a method for reinitializing products states that have experienced a period on nonproduction.

In [127], Patel uses a set of equality constraints similar to the ANOVA models above accept that the tool states are left unconstrained. Thus tool states are assumed to contain all random variations while states for other context items are assumed constant and centered around an arbitrary constant. Using a Kalman Filter, the new model is tested against threaded EWMA for a simulated system and real but unspecified processing system. The non-threaded model is found to be superior.

Finally, Wang et al. [123], use term non-threaded control (coined in [86]) when discussing state space based, cooperative controllers and show that a system without quals can be transformed to a system which has states that are linear combinations of the original states. While the original system is not fully observable, the transformed system

is, and unbiased estimates of the new states can be found. This is sufficient for control because the controller model requires only a good estimate of the current context (which is a linear combination of the original states) and not necessarily exact estimates of the original states themselves. The Kalman filter, recursive least squares (RLS), and JADE are all shown to be specific examples of the general scheme to find the Best Linear Unbiased Estimate (BLUE) and are tested and compared with threaded EWMA on simulation and process data. Additionally, Bayesian enhanced versions of the Kalman filter and RLS are proposed to account for shift disturbances.

One particular advantage of non-threaded controllers compared to their threaded counterparts, which is alluded to in the discussion of Figure 2.2 and examined in more detail in [123] and [86], is their improved performance for initialization of threads. In non-threaded control, each wafer is modeled as the sum of linear contributions from the various context items, and each item is estimated. Alternately, threaded control estimates a single lumped parameter for each thread that represents the combined effect of the context items for that thread [86].

To demonstrate the differences, a threaded control system which employs EWMA filters to estimate the lumped context states is run on the test systems of Section 4.6 for comparison. In the EWMA implementation, the input-output gain is left constant and the EWMA parameter, lambda, is used to tune filter performance. In practice, a lambda of 0.1 to 0.3 is commonly used [13]. Here 0.3 is chosen from this range as the best value. If the state estimates of the threads are initialized in the same manner as in non-threaded control (i.e., 50% error in product states), the outputs slowly converge to the target due to the low setting for lambda (see dashed blue line in Figure 1). In simulation work, this is normally compensated for by initializing threads when they first appear with the true state values; the advantage of this approach is clear in Figure 1 where the solid red line

represents the output errors when the system is run with thread initialization. Not only is the system output better centered about the target over the length of the simulation, but larger errors are avoided when a product arrives to a tool on which it has not been previously processed.

In practice, initialization is accomplished by separating several wafers from the new product lot to make a child or send-ahead lot. The child lot is sent to the new tool while the rest of the lots of this product are put on hold. Additionally, depending on operating requirements, the tool itself may be put on hold [130]. The hold time is a function of the processing, measurement, and analysis times required to characterize the new product on the tool. This obviously has a major detrimental impact on the tool utilization and throughput of the system and is therefore discouraged [123, 130, 131]. Additionally, there is a strong possibility that child lots will have measurements that are out of tolerance specifications [132], leading to rework or scrap. Therefore, send-ahead lots can have a strong negative effect on both scheduling and process control.

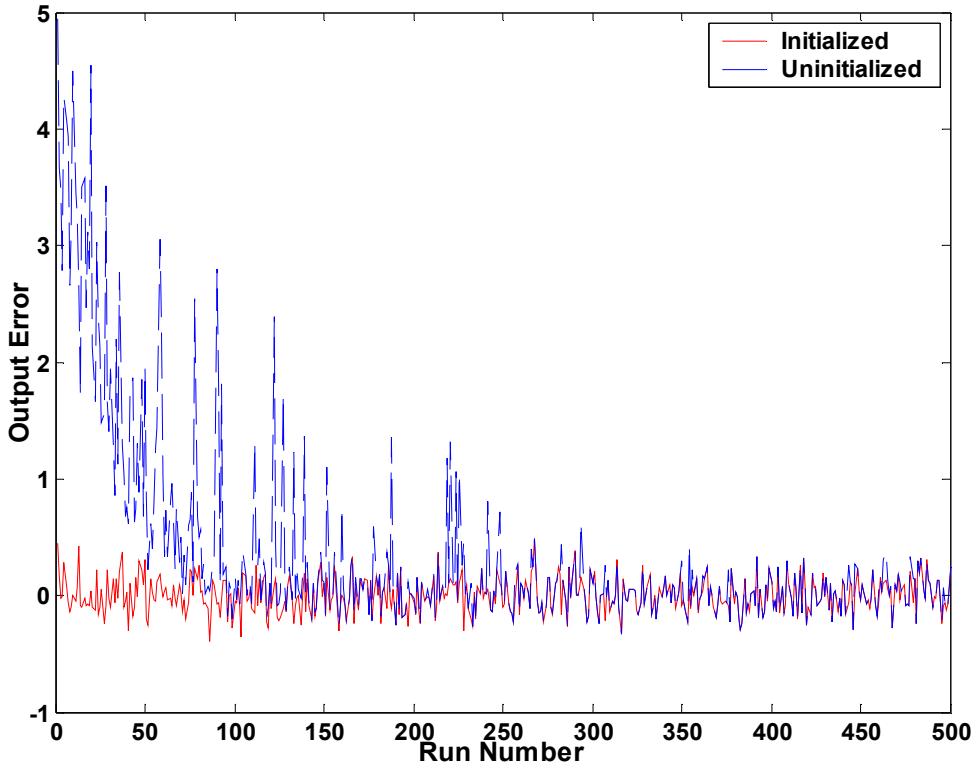


Figure 2.3: Threaded control with and without thread state initialization

By comparison, the non-threaded control simulations in Section 4.6 initially have output errors that are much higher than those of an initialized threaded EWMA system, but they do not have the same advantage of optimized tuning. When the value of $Q$ used by the Kalman filter is decreased to give better noise rejection, the control is greatly improved (this includes the elimination of unstable results seen for some systems with delays). With better tuning, the output error results for the non-threaded system are comparable to those of the threaded system for purely white noise disturbances but without the need for send-ahead lots.

The major difference in performance between the two control systems is observed when non-stationary disturbances such as shifts, drifts, and IMA(1,1) noise are present.

Table 2.1 shows a comparison of the threaded and non-threaded control systems based on root mean squared error (RMSE) of the outputs for the eight test systems with non-stationary disturbances and a FIFO schedule. It is clear from Table 1 that the non-threaded control is superior to the threaded control, especially in cases where the system is larger (Tests5, 6, and 10 have more products and/or tools than the other systems). It is also noted that the threaded control system does not experience any estimation delay because each thread is controlled individually, and it is not possible for the same thread (i.e., tool/product combination) to run more than once at any given time. Despite this fact, the threaded system with estimation delays still outperforms the threaded system.

| Test | Threaded | Non-threaded | |
|---|---|---|---|
| Number | EWMA | Ideal | Delayed |
| **1** | 0.270 | 0.206 | 0.211 |
| **2** | 0.272 | 0.209 | 0.213 |
| **3** | 0.271 | 0.206 | 0.209 |
| **4** | 0.270 | 0.204 | 0.213 |
| **5** | 0.382 | 0.239 | 0.257 |
| **6** | 0.363 | 0.229 | 0.237 |
| **7** | 0.284 | 0.219 | 0.224 |
| **8** | 0.804 | 0.439 | 0.574 |

Table 2.1: RMSE of outputs for threaded vs. non-threaded control with non-stationary disturbances, which demonstrate the superiority of non-threaded control. The scheduling policy used is FIFO.

The only way to improve the performance of the threaded control system is to allow more send-ahead lots as the processing proceeds. Typically, the use of these additional child lots is determined by counting the number of runs since the last appearance of the next thread to be processed. If this number exceeds a predetermined threshold, then the thread is reinitialized through the use of a send-ahead lot. Obviously,

the more quickly the underlying states change in a non-stationary manner, the more child lots are required to maintain good control. This is a major drawback to threaded control.

# Chapter 3

# Kalman Filter Based Estimation for Multi-tool, Multi-product Systems with Qualification Runs

## 3.1 INTRODUCTION

In the previous chapter, the research of Pasadyn and coworkers [41, 53–55] is given as an example of a cooperative control scheme for systems with multiple products and tools. Also discussed in these works are some possibilities for using the ordering of products to enhance control performance. Because Pasadyn's system can be used in this way, it offers a convenient framework for the study of process control and scheduling interactions and will be used as the basis for much of the work in this thesis. The current chapter reviews the work in [41] and related papers [53–55] to provide a background for models used in the current work and also gives further analysis of system properties (especially observability). Additionally, adjustments to the model are made for multi-input systems, a new form of the model, which eliminates the "adjustment" state, is presented, and the use of qualification runs is analyzed.

## 3.2 SYSTEM MODEL

In the dissertation by Pasadyn [41], a CMP example is used to show how the process model can be written for multiple tools and products. The basic model of the CMP process is given as $\overline{x} = \overline{r} \cdot \overline{t} \cdot \overline{f}$ where $\overline{x}$ is the average amount of material removed from the surface, $\overline{r}$ is the average rate of removal for the tool, $\overline{t}$ is the average time for removal, and $\overline{f}$ is a product dependency factor that accounts for the difference in removal rate due to product characteristics such as topography. This equation is then

linearized about the nominal operating point $(r_0, t_0, f_0)$ to give an expression for the deviation from the nominal removal:

$$y = r_0 t f_0 + r t_0 f_0 + r_0 t_0 f \tag{3.1}$$

where $r$, $t$, and $f$ represent deviations from the nominal values.

Equation (3.1) can be put into a state space form if $t$ is considered to be the input to the system, $y$ is considered to be the output, and $r$ and $f$ are considered to be the tool and product related states, respectively. The resulting state space equations can be written as

$$\begin{bmatrix} x_{adj} \\ r \\ f \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{adj} \\ r \\ f \end{bmatrix}_k + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} t_k \tag{3.2}$$

$$y_k = \begin{bmatrix} r_0 f_0 & t_0 f_0 & r_0 t_0 \end{bmatrix} \begin{bmatrix} x_{adj} \\ r \\ f \end{bmatrix}_k \tag{3.3}$$

where the adjustment state, $x_{adj}$, is used to represent the input to the process. In the general form, the state space equations are written as

$$x_{k+1} = A x_k + B u_k \tag{3.4}$$

$$y_k = C x_k, \tag{3.5}$$

where (3.4) and (3.5) are the same as (3.2) and (3.3), respectively.

In the multi-tool/product system, equation (3.1) can be written for all contexts that contain a tool from the set, $i$, and a product from the set, $j$:

$$y_{ij} = r_0 f_0 t + t_0 f_0 r_i + r_0 t_0 f_j. \tag{3.6}$$

For the case of two tools $(i = 1, 2)$ and two products $(j = 1, 2)$, the set of output equations is

$$
\begin{aligned}
y_{11} &= r_0 f_0 t + t_0 f_0 r_1 + r_0 t_0 f_1 \\
y_{12} &= r_0 f_0 t + t_0 f_0 r_1 + r_0 t_0 f_2 \\
y_{21} &= r_0 f_0 t + t_0 f_0 r_2 + r_0 t_0 f_1 \\
y_{22} &= r_0 f_0 t + t_0 f_0 r_2 + r_0 t_0 f_2,
\end{aligned}
\tag{3.7}
$$

which can be expressed in state space form as

$$
\begin{bmatrix} x_{adj} \\ r_1 \\ r_2 \\ f_1 \\ f_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{adj} \\ r_1 \\ r_2 \\ f_1 \\ f_2 \end{bmatrix}_k + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} t_k \tag{3.8}
$$

$$
\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{bmatrix}_k = \begin{bmatrix} r_0 f_0 & t_0 f_0 & 0 & r_0 t_0 & 0 \\ r_0 f_0 & t_0 f_0 & 0 & 0 & r_0 t_0 \\ r_0 f_0 & 0 & t_0 f_0 & r_0 t_0 & 0 \\ r_0 f_0 & 0 & t_0 f_0 & 0 & r_0 t_0 \end{bmatrix} \begin{bmatrix} x_{adj} \\ r_1 \\ r_2 \\ f_1 \\ f_2 \end{bmatrix}_k \tag{3.9}
$$

To simplify the presentation, all parameters ($r_0$, $t_0$, $f_0$) are assumed to equal one. Also, the rates are represented by the tool states, $x_{t_i}$, the product factors are represented by the product states, $x_{p_j}$, and the time, $t_k$, is represented by the input, $u_k$. The resulting state space equations are

$$
\begin{bmatrix} x_{adj} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{adj} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_k \tag{3.10}
$$

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{bmatrix}_k = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{adj} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k . \tag{3.11}$$

### 3.3 SYSTEM OBSERVABILITY

In order to successfully estimate the states, the system must be observable (i.e., the system states can be reconstructed from the available measurements) [94]. To test the observability of a system with *n* states, the following matrix is calculated:

$$\Gamma_O[A,C] = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}. \tag{3.12}$$

If the matrix $\Gamma_O$ (called the observability Gramian) is full rank, then the system is considered fully observable.

For the two tool, two product system above, the observability matrix (after eliminating duplicate rows due to the fact that $A = A^2 = A^3 = A^4$) is

$$\Gamma_O[A,C] = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.13}$$

which has a rank of four. Therefore, the system is not observable and the model needs to be adjusted.

Pasadyn [41] suggests several methods for ensuring the observability of the system. One method is to choose a tool or product that has a known nominal bias as a reference state. The problem with this approach is that the reference value may not remain accurate for very long in a manufacturing setting with high variability. A second method is to add constraints to the system which effectively reduces the number of states to be identified, as in [91-93]. Other researchers have used alternative methods of model reduction [122, 123] or model regularization [127]. The final approach is to add more measurements; specifically measurements of qualification runs. A qualification run uniquely identifies the bias of the tool on which it is performed (i.e., it gives a direct measurement of the tool state).

For the two tool case demonstrated earlier, the two new outputs are

$$y_1 = r_1 \tag{3.14}$$

and

$$y_2 = r_2. \tag{3.15}$$

After adding the measurements in equations (3.14) and (3.15) to the system, the output equation for the state space model becomes

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_1 \\ y_2 \end{bmatrix}_k = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{adj} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k, \tag{3.16}$$

and the new observability matrix is

$$\Gamma_O[A,C]=\begin{bmatrix}1&1&0&1&0\\1&1&0&0&1\\1&0&1&1&0\\1&0&1&0&1\\0&1&0&1&0\\0&1&0&0&1\\0&0&1&1&0\\0&0&1&0&1\\0&1&0&0&0\\0&0&1&0&0\end{bmatrix} \tag{3.17}$$

which has a rank of five. Thus, the new system is fully observable. In general, the observable system with quals has $t(p+1)$ outputs (where $t=$ number of tools and $p=$ number of products) as opposed to the original system which had *tp* outputs.

One fact not pointed out in [41] is that adding the qualifications on both tools in order to make the system observable is not actually necessary; the system is fully observable if only one of the tool states is directly measurable. However, adding qual runs for both tools to the system is advantageous because it allows for more flexibility. From the standpoint of the estimator, information on the states can be shared more efficiently when quals are not always run on the same tool. As Pasadyn [41] illustrates, if the state of one tool is identified with a qual (thus reducing its estimate variance greatly), then subsequently running the same product on the qualified tool and then on another tool will help to reduce the variance in the state estimate of the second tool. In larger systems, this "trickle down" effect would take longer if quals were only allowed on one tool, whereas variance reduction in the tool states is accomplished more quickly by allowing quals on all tools. From a scheduling standpoint, allowing the number of qualification wafers to be distributed over all tools prevents a build up of non-production wafers on one tool, and thus, exceedingly low tool utilization for this tool.

Because physical constraints require that only one of the products can be processed on a given tool at a given time, the real manufacturing system can not run all outputs of the process model at the same time. Therefore, only a subset of the rows in the output equation (3.16) can be used for modeling and estimation during a given process run. As an example, consider the case where product 2 is processed on tool 2. In this case, only the fourth row of (3.16) is needed and the resulting equation is

$$y_{22_k} = C_r x_k = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{adj} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k . \tag{3.18}$$

The observability matrix for this reduced system is

$$\Gamma_O \left[ A, C_r \right] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.19}$$

which has a rank of two so that only two of the system states are observable from the measurement of $y_{22}$. Likewise, many other reduced systems created by possible run combinations allowed by the physical system are found to have rank deficient observability Gramians. Table 3.1 shows the full range of physically possible subsets and their observability.

At any given run time, it is apparent from Table 3.1 that only Cases 5, 6, and 15 are fully observable. These cases involve the processing of nothing but qualification wafers, so that any run that includes production wafers will be only partially observable. As Pasadyn [41] notes, such a system is similar to one seen in continuous processes with

multiple sampling rates where only a subset of the outputs are available for measurement at each time step. This leads to a system in which the observability is time varying [95], but Pasadyn states that the major requirement is that the full system is fully observable. In the next subsection, the subject of observability in systems where only a subset of the full system is active at any given run is more thoroughly addressed.

| Case Number | Row(s) of Output Equation Used | Product Run on Tool 1 | Product Run on Tool 2 | Number of States Used in Subsytem | Rank $\Gamma_o$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | - | 3 | 2 |
| 2 | 2 | 2 | - | 3 | 2 |
| 3 | 3 | - | 1 | 3 | 2 |
| 4 | 4 | - | 2 | 3 | 2 |
| 5 | 5 | Qual | - | 1 | 1 |
| 6 | 6 | - | Qual | 1 | 1 |
| 7 | 1 and 3 | 1 | 1 | 4 | 3 |
| 8 | 1 and 4 | 1 | 2 | 5 | 3 |
| 9 | 1 and 6 | 1 | Qual | 4 | 3 |
| 10 | 2 and 3 | 2 | 1 | 5 | 3 |
| 11 | 2 and 4 | 2 | 2 | 4 | 3 |
| 12 | 2 and 6 | 2 | Qual | 4 | 3 |
| 13 | 5 and 3 | Qual | 1 | 4 | 3 |
| 14 | 5 and 4 | Qual | 2 | 4 | 3 |
| 15 | 5 and 6 | Qual | Qual | 2 | 2 |

Table 3.1: Analysis of observability for model subsystems; fully observable systems shaded.

### 3.3.1 Partial Observability and Switched Systems Theory

While the multi-tool/product system can be thought of as a multi-rate sampled system, this may not be the best way of approaching the observability problem. Most literature on multi-rate sampled systems assumes some periodic form of the

measurements with the frequency of at least one measurement being a common denominator (c.f., [107] and references therein). Such constraints could be imposed on the system studied in this work but requiring the various outputs to be periodic is far from realistic in practice.

Another way of approaching the problem is to consider the system to be a switched linear system. Switched linear systems are a subclass of hybrid systems which use a combination of continuous and discrete models to control a process [104, 106]. In a switched linear system, the system matrices are restricted to linear models whose system matrices are allowed to change during the course of the process run. The possible values for the system matrices form a finite set and are called the modes of the system, while the order in which the system changes from one mode to the next is called the switching signal, switching rule, switching path, or mode signal [105].

Some research in the field of switched systems focuses on the case where all system matrices are allowed to change (c.f., [109, 112]). Other papers focus on the case where only the state transition matrix, $A$, changes [108], only the output matrix, $C$, changes [111], or both of these are allowed to change [110]. Obviously, the scenario where only $C$ changes is most similar to the multi-tool/product system in this work, but results from analysis of any system where $C$ changes can be applied.

Another factor that also differentiates the papers on switched linear systems is whether the switching rule is unknown and needs to be identified [109, 110] or is known, in which case it can be arbitrary [111] or used as an input [112]. In the multi-tool/product system, the mode is known because the composition of the measurement matrix is determined by the tool being used and the product being run. In the case where product ordering is determined by a dispatch rule like FIFO, the switching signal is considered

arbitrary, but in the case where product ordering is changed to enhance system performance it is considered a control variable.

As the interest in switched linear systems has increased over the last ten to fifteen years, many researchers have focused their efforts on analysis of the fundamental properties of observability [109, 111, 112, 114], controllability [112, 114], and stabilization [112, 113, 115]. Several papers focus on the fact that systems with time variant $A$ and/or $C$ matrices also have time variant observability [111, 114, 116, 117]. Further more, the extent to which the states are observable indicates the quality of their estimates [116, 117]. In [111, 114], Babaali and Egerstedt discuss the concept of pathwise observability for switched linear systems. In [114], the authors study a linear system in which $A$, $B$, and $C$ all change values depending upon the switching signal, which is labeled $\theta$. The switching signal has a path length, $N$, and can take on values in set $\{1, 2,\ldots, s\}$. The path-dependent observability Gramian for such a system is defined thusly:

Given $A(\theta_k)$, $C(\theta_k)$, where $\theta_k \in \{1,2,\ldots,s\}$ and $k = 1,2,\ldots,N$,

$$O(\theta) \triangleq \begin{bmatrix} C(\theta_1) \\ C(\theta_2)A(\theta_1) \\ \vdots \\ C(\theta_N)A(\theta_{N-1})\cdots A(\theta_1) \end{bmatrix} \tag{3.20}$$

and for systems in which $A$ is not dependent upon the switching signal [111]

$$O(\theta) \triangleq \begin{bmatrix} C(\theta_1) \\ C(\theta_2)A \\ \vdots \\ C(\theta_N)A^{N-1} \end{bmatrix}. \tag{3.21}$$

The second definition of the Gramian above is more appropriate for the multi-tool/product system and can be simplified further because $A$ is the identity matrix. Thus, the pathwise observability Gramian becomes

$$O(\theta) = \begin{bmatrix} C(\theta_1) \\ C(\theta_2) \\ \vdots \\ C(\theta_N) \end{bmatrix}. \tag{3.22}$$

The implication of the definition of pathwise observability for the multi-tool/product system is that the observability matrix in (3.22) is always full rank for a given path when all tools are utilized, at least one qual is run and every product is run at least once. While this is interesting from a theoretical standpoint, it gives no incite into path selection and observability at individual runs. Much like the observability Gramian test for a non-switched, LTI system in Section 3.2.1.1, the rank of (3.22) only gives a yes or no answer for the observability of a particular path.

To gain more insight into the instantaneous behavior of a changing system's observability, a third branch of the literature that involves the study of sensor fusion for multi-sensor, time-varying systems is investigated. A major application of this research is flight guidance systems that integrate information from various measurement devices such as global positioning systems (GPS), inertial navigation systems (INS), radar, etc. Because the models for such systems can be placed in a state space form similar to those of switched linear systems, results from their analysis are also useful in the study of the multi-tool/product system.

In [116], Chen studies the concept of local observability for a discrete time, linear system with $n$ states and a single output whose measurement matrix is time varying. At a given run number, $k$, the path length needed for local observability is shown to be $n$ (the

total number of states), and the Gramian for the rank test is similar to (3.20) but is defined for steps $k$ through $k+n-1$:

$$O(k) \triangleq \begin{bmatrix} C(k) \\ C(k+1)A(k) \\ C(k+2)A(k+1)A(k) \\ \vdots \\ C(k+n-1)A(k+n-2)\cdots A(k) \end{bmatrix} \tag{3.23}$$

which reduces to

$$O(k) = \begin{bmatrix} C_r(k) \\ C_r(k+1) \\ C_r(k+2) \\ \vdots \\ C_r(k+n-1) \end{bmatrix} \tag{3.24}$$

for the multi-tool/product system.

If $\text{rank}\big(O(k)\big) = n$ for all $k$, then the system is completely observable. If $\text{rank}\big(O(k)\big) < n$ for some $k$, then the system is only locally observable at runs where $\text{rank}\big(O(k)\big) = n$. For the multi-tool/product system studied here, the only way to guarantee complete observability of the system (i.e., $\text{rank}\big(O(k)\big) = n \ \forall \ k$) is to abide by the following four conditions for every block of $n$ consecutive samples:

1. Include exactly one qual run.
2. Include at least one run of each product.
3. Include at least one production run on each tool.
4. Each tool should process at least one product which has also been processed by at least one other tool.

As they relate to the system as a whole, the processing requirements for full observability outlined above lead to the following interesting results:

1. As the number of states increases, complete observability can be maintained with a lower qual run frequency because as n gets larger, the proportion of quals within each block, 1/n, gets smaller. This works in favor of real manufacturing settings where quals need to be minimized in order to reduce costs and there are many more tools and products than in the smaller examples studied thus far in this work.
2. Additionally, if the above constraints are enforced when selecting the contexts for processing over a block of length n, then the contexts in the selected set will be unique. While this does not necessarily preclude the same product from being run multiple times, it does preclude any product from being run more than once on the same tool in the given block of n runs. In other words, the above processing rules for full observability discourage tool dedication.

Unfortunately, for cases in which the local observability Gramian is not full rank, the previous problem of the rank test only giving a yes/no type answer still exists. Therefore, Chen suggests for single output systems, that the condition number of $O(k)$ can be used to determine the degree of local observability. For this method, the multi-tool/product system can be framed as a single output system if concurrent starts are not allowed on multiple machines (i.e., only one row of the output equation is used at each run). In testing of small multi-tool/product systems like those studied so far, this method does not work well because most Gramians, $O(k)$, which are rank deficient will have a condition number of infinity. In larger scale systems with Gramians that are rank deficient and also of the same rank, small differences in the condition numbers of the Gramians are observed. Thus, in larger systems, it may be possible to use $\mathrm{cond}\left(O\left(k\right)\right)$ as the objective to be minimized when choosing a schedule for the next $n$ runs or choosing the next run based on the past $n$-1 runs.

As an improvement upon Chen's work, Hong et al. [117], study observability measures for time-varying systems with multiple outputs and propose a new measure based on the information matrix, $L_{0,k}$, where

$$L_{0,k} = \sum_{i=0}^{k} A_{i,0}^T C_i^T R_i^{-1} C_i A_{i,0}, \tag{3.25}$$

with $A_{i,0}$ being the product of all state transition matrices from the initial value to step $i$:

$$A_{i,0} = A_i A_{i-1} \ldots A_0. \tag{3.26}$$

For the multi-tool/product system where $A$ is the identity matrix, the information matrix is can be simplified to

$$L_{0,k} = \sum_{i=0}^{k} C_i^T R_i^{-1} C_i. \tag{3.27}$$

Note that the information matrix is a square form of the observability Gramian and is scaled by the output noise variance, $R$. Like the error covariance matrix, the information matrix is non-negative and symmetric with size $n \times n$. The authors show that using the information matrix to determine the degree of observability is preferred to using the error covariance matrix because methods based on the latter are much more sensitive to errors in the process and noise models [117]. Additionally, the value of $P_k$ is sensitive to its initial value, $P_0$, and this can give erroneous results when testing for the degree of observability. The authors also reiterate the fact presented above that while methods based on the rank of the observability Gramian can determine whether or not the system is observable, they do not give a good indication as to the degree of observability.

In order to measure the degree to which the system is observable, Hong et al. [117] perform a singular value decomposition (SVD) on the information matrix [118], i.e.,

$$L_{0,k} = U_k \Sigma_k U_k^T, \tag{3.28}$$

where $\Sigma_k$ is a diagonal matrix with the singular values of $L_{0,k}$ ordered from largest to smallest along the diagonal, and $U_k$ contains the singular vectors associated with each singular value. The singular vectors represent subspaces of the overall state space, and their associated singular values give an indication of the degree of observability for each of the individual subspaces. In this case, the smallest singular value of $L_{0,k}$ is associated with the least observable subspace, and because this subspace is a part of the overall state space, its singular value is considered to be the measure of the degree observability for the entire state space at run $k$.

The methods described in this section are useful for monitoring the observability of switched or time varying systems. Because the multi-tool/product system discussed in this work can be categorized as either type of system, the above methods are applicable and are used later in this chapter in the development of control friendly scheduling algorithms.

## 3.4 CONTROLLER

### 3.4.1 Feedback Control Law

Similar to many of the run-to-run algorithms reviewed in Chapter 2, Pasadyn's [41] work uses a deadbeat control law. To construct the control law, the state equation (3.4) is substituted into the output equation (3.5). Because the actual values of the states are unknown, estimates of the states, $\hat{x}_{k|k-1}$, are used and because the process targets are the desired values of the outputs, a vector of process targets, $T$, is substituted for the output vector, $y_k$. The resulting equation is

$$T = C\left(A\hat{x}_{k|k-1} + Bu_k\right). \tag{3.29}$$

Equation (3.29) is then solved for $u_k$, the input needed to return the estimated output to the target at the next run;

$$u_k = \left(CB\right)^{-1}\left(T - CA\hat{x}_{k|k-1}\right). \tag{3.30}$$

Equation (3.30) is given for a case where all outputs are run at the same time (i.e., the full *C* matrix is used). As mentioned earlier, running all contexts simultaneously is not possible and a reduced version of the *C* matrix, $C_r$, should be employed. When multiple tools start at the same time (e.g., Cases 7–15 of Table 3.1), Cr has more than one row, but the matrix *B* is constructed for a single input system and therefore, has only one column. The resulting controller gain $(C_rB)^{-1}$ is then the inverse of a non-square matrix. In such a case, the Moore-Penrose pseudoinverse can be used to obtain a least squares solution for the single input that minimizes the sum of the predicted output errors for the two contexts being run. The non-square gain matrix discussed above results from representing a system with multiple machines that can start simultaneously with a model that has only a single input. This problem can be addressed by adding individual inputs for each tool to the system model, an approach which is demonstrated in the next subsection.

### 3.4.2 Model Adjustments for Multiple Simultaneous Starts

As was mentioned in the previous subsection on the controller model, the current form of the combined model for the multi-tool/product system does not allow for multiple inputs. This can be problematic if there are occasions where multiple tools are started simultaneously. This shortcoming can be corrected easily by adding additional inputs along with their corresponding adjustment states to the state equation. Thus, the new state equation for an example system with two tools and two products is

$$\begin{bmatrix} x_{adj,t_1} \\ x_{adj,t_2} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{adj,t_1} \\ x_{adj,t_2} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{t_1} \\ u_{t_2} \end{bmatrix}_k, \tag{3.31}$$

and the new output equation is

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_1 \\ y_2 \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{adj,t_1} \\ x_{adj,t_2} \\ x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k. \tag{3.32}$$

The problem with this form of the model is that for every new tool added to the system, not only are a new tool state and input added to the equations but also a new adjustment state. Because the inputs are well known, this represents a waste of computational effort to calculate estimates for the many adjustment states. Fortunately, it is possible to alter the model form in order to do away with the adjustment states.

### 3.4.3 Controllability and Elimination of Adjustment States

Upon inspection of the system equations, it becomes clear that the adjustment states may not be necessary. First, the form of the state equation (3.31) is examined. It is obvious from the form of the input matrix, *B*, that at the update time *k+1*, the adjustment states are the only states that are functions of the input. It is also apparent from the form of the state transition matrix, *A*, that none of the updated tool and product states are functions of the current adjustment states. This is easily demonstrated by looking at the usual test for controllability of the states. Controllability is the ability to use the system

input(s) to steer the states from an initial value at step $k_0$ to a zero value at step $k_f < \infty$ [99]. A related concept is reachability which implies that the states at a nonzero initial condition can be steered by the input to the origin in a finite number of steps (i.e., $k_f < \infty$). If all states in the system are controllable (reachable), then the system is said to be completely controllable (reachable).

Similar to the measure for observability, a controllability Gramian is calculated as follows:

$$\Gamma_C[A,B] = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}. \tag{3.33}$$

If the controllability Gramian is full rank, then the system is considered to be completely controllable. For the system with two tools and two products in equation (3.10), the controllability Gramian (after eliminating duplicate rows due to the fact that $A = A^2 = A^3 = A^4$) is

$$\Gamma_C[A,B] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.34}$$

which has a rank of 1. For the system in (3.31), the controllability Gramian (after eliminating duplicate rows due to the fact that $A = A^2 = \ldots = A^5$) is

$$\Gamma_C[A,B] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.35}$$

which has rank of two. It is clear that the only controllable states are the adjustment states (i.e., the adjustment states make up the controllable subspace). Finally, from the

form of the output matrix, *C*, one can see that the adjustment states serve simply to transfer the input from the state equation to the output equation.

Fortunately, it is quite easy to alter the form of the model to eliminate the need for adjustment states. By adding a feed-forward matrix, *D*, the input can be added directly to the output equation without the use of adjustment states, thus creating a more parsimonious model. The system equations in the new feedforward model now take on the general form

$$x_{k+1} = Ax_k \tag{3.36}$$

$$y_k = Cx_k + Du_k. \tag{3.37}$$

For the two tool, two product system, (3.36) and (3.37) are

$$\begin{bmatrix} x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k \tag{3.38}$$

$$\begin{bmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \\ y_1 \\ y_2 \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{t_1} \\ x_{t_2} \\ x_{p_1} \\ x_{p_2} \end{bmatrix}_k + \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_k. \tag{3.39}$$

As another alternative, the input and its corresponding gain can simply be subtracted from both sides of Equation (3.39) and the open-loop output of the form

$$\tilde{y}_k = y_k - u_k \tag{3.40}$$

can be used.

A final note on the elimination of the adjustment state concerns the stability of the estimation scheme. Condition number can be thought of as an indication of the nearness

to singularity of a matrix and a smaller condition number is favorable. Because both the matrices A and C are used in the update equations for the estimate error covariance matrix, which is in turn used to update the Kalman filter gain for estimate updates, it is interesting to study the effect that a change from the adjustment state form of the model to the feedforward form of the model has on the condition number of P. It has also been suggested that in a non-threaded control system, monitoring of the condition number of P is useful to determine when estimates are becoming biased [93]. Table 3.2 shows the condition number of P for several different systems under the two different model forms. The first row gives values for the full system, assuming it is possible to obtain the entire measurement matrix at every step. For this ideal case, the stability of the estimation is far superior with the new model matrices. The last three rows of Table 3.2 give values for a system operating under the partial measurement policy (i.e., only one row of C is used at a time). The three systems employ different proportions of qualification runs, and again it is seen that the values of the condition number of P are far superior with the feedforward model. Additionally, the values for the feedforward system trend downward as the number of qualification runs is increased, which is to be expected.

| | **Mean condition number of P** | |
|---|---|---|
| | **With $x_{adj}$** | **Without $x_{adj}$** |
| **Steady State** | 2.65E+03 | 1.40E+01 |
| **Zero Quals** | 3.08E+06 | 2.05E+06 |
| **5% Quals** | 3.12E+06 | 1.65E+04 |
| **10% Quals** | 3.16E+06 | 2.72E+03 |

Table 3.2: Values of the condition number of P for several test systems.

## 3.5 STATE ESTIMATOR

### 3.5.1 Kalman Filtering

In order to obtain estimates of the process states, Pasadyn employs a discrete time Kalman filter [96]. The Kalman filter was introduced earlier in Section 2.2.1.4 which covered MPC methods. Pasadyn's model can be formulated as a discrete-time linear system like the one in equations (2.8) and (2.9) with process noise, $w_k$, and measurement noise, $v_k$. If the noise terms conform to the assumptions given in Section 2.2.1.4 and are also Gaussian (i.e., $w_k \sim N(0,Q)$ and $v_k \sim N(0,R)$), then the discrete Kalman filter is known to give optimal estimates of the states. Otherwise, it is the best linear estimate of the states.

Because the system is driven by white noise processes (i.e., the terms $w_k$ and $v_k$), the states and outputs are considered to be random variables. The goal of the Kalman filter is to find the best estimate of the states conditioned on the measurements up to the present time [97, 98]. To do this, the filter must propagate the conditional mean of the state (i.e., the state estimate) and the covariance of the state estimate error from one time step to the next. In other words, the measurement information is used to give an unbiased estimate of the state and by way of the covariance, an indication of how confident one can be in this estimate. The state estimate and the estimate error covariance can be described in terms of expected values as follows:

$$\hat{x}_k = \mathrm{E}\left(x_k\right) \tag{3.41}$$

$$P_k = \mathrm{E}\left(\tilde{x}_k \tilde{x}_k^T\right) \tag{3.42}$$

where the estimate error is defined as $\tilde{x}_k = \left(x_k - \hat{x}_k\right)$.

Unlike the Kalman filter equation (2.10) given in the previous chapter, a predictor-corrector form is used by Pasadyn. As implied by its name, the predictor-corrector form of the Kalman filter gives a prediction of the states at the beginning of a

step (e.g., a wafer start on a tool), and then corrects the estimates when measurements are received. In the first step, also called the time update, the state equation is used to give the a priori estimate of the states using (2.8):

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{3.43}$$

where $\hat{x}_{k+1|k}$ represents the one-step-ahead prediction of the state given measurements up to step $k$. Similarly, a time update for the estimate error covariance matrix, $P$, is found from the following Lyapunov equation [96]:

$$P_{k+1|k} = AP_{k|k}A^T + GQG^T. \tag{3.44}$$

Note that a time update will increase the value of the elements of $P$, that is, a prediction step will decrease the confidence in the state estimates.

Once a measurement is received, a second update is done so the estimator can incorporate the new information. First, the state estimate is updated using the following equation:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + P_{k+1|k}C^T\left(CP_{k+1|k}C^T + R\right)^{-1}(y_{k+1} - C\hat{x}_{k+1|k}) \tag{3.45}$$

where the last term in parentheses,

$$\tilde{y}_{k+1} = (y_{k+1} - C\hat{x}_{k+1|k}), \tag{3.46}$$

is called the residual or innovation. Since the residual is the difference between the measured output and the expected (estimated) output, it gives an indication of the error in the state estimates. To determine how much of the residual should be incorporated into the new state estimate (i.e., the level of confidence in the measurement as opposed to the previous state estimate), the residual is multiplied by the term

$$L_{k+1|k} = P_{k+1|k}C^T\left(CP_{k+1|k}C^T + R\right)^{-1} \tag{3.47}$$

which is referred to as the Kalman gain. With the noise covariance matrices, $Q$ and $R$, specified by the user, the equations of the Kalman filter are designed to produce a gain that offers the optimal balance between information contained in the previous estimate

and information contained in the measurement. Notice the similarity of (3.45) to the EWMA update equation (2.4), which uses the constant gain, $\lambda$, instead of the Kalman gain.

To acquire a better understanding of the effect of the Kalman gain on the contributions to the a posteriori state estimate, $\hat{x}_{k+1|k+1}$, a simple analysis of some limiting cases can be insightful. Looking at the two cases where $P$ and $R$ become small, the effect of the Kalman gain in (3.45) can be seen. First, if the a priori estimate error covariance is small compared to the measurement noise covariance, then there is higher confidence in the a priori estimate than the measurement and the contribution by the innovation to the a posteriori state estimate will be small. On the other hand, if the measurement noise covariance is small compared to the estimate error covariance, then there is higher confidence in the measurements than the estimates and the contribution of the residual to the a posteriori estimate is larger. Table 3.3 gives a summary of this analysis.

| High confidence in: | Measurement | Estimate |
|---|---|---|
| Change in covariance | $R \to 0$ | $P_{k+1\|k} \to 0$ |
| Effect on Kalman Gain | $\lim_{R\to 0} L_{k+1\|k} = C^{-1}$ | $\lim_{P_{k+1\|k}\to 0} L_{k+1\|k} = 0$ |
| Residual contribution to a posteriori estimate | HIGH | LOW |

Table 3.3: Analysis of Kalman filter gain for limiting cases and effects on state estimates.

The error covariance is also updated during the correction step using the following equation which is written in terms of the Kalman gain:

$$P_{k+1|k+1} = (I - L_{k+1|k}C)P_{k+1|k}. \tag{3.48}$$

Alternatively, the a posteriori error covariance can first be written in terms of the a priori error covariance as

$$P_{k+1|k+1} = (P_{k+1|k}^{-1} + C^T R^{-1} C)^{-1}, \tag{3.49}$$

and the Kalman gain can then be written in terms of the a posteriori error covariance as

$$L_{k+1|k} = P_{k+1|k+1} C^T R^{-1}. \tag{3.50}$$

Note that the measurement update step decreases the error covariance, that is, it increases the confidence in the state estimates.

### 3.5.2 Run-varying Kalman Filter

Similar to the situations encountered with the process model and control law, the estimator requires some adjustments when used with the multi-tool/product system. Again, a reduced version of the $C$ matrix, called $C_r$, must be used, where $C_r$ contains only the rows that correspond to the contexts being made during the present run. Also, a properly sized version of the measurement noise covariance matrix, $R$, must be chosen. Here only the entries of $R$ that are at the intersections of rows and columns corresponding to the measured outputs are used in the reduced version of the measurement covariance matrix, called $R_r$. Appendix D gives an example of pseudo-code for calculating $R_r$. Generally, the measurements are considered independent of one another and $R$ is a diagonal matrix. In this case, $R_r$ is an appropriately sized diagonal matrix with nonzero elements corresponding to the noise variances of the outputs which are measured.

It is important to note that the estimate error covariance matrix, and therefore the Kalman gain, are dependent on the run index, $k$. Under normal circumstances where the system matrices ($A$, $B$, $C$) are unchanged from one run to the next, the a priori and a posteriori error covariance matrices and the Kalman gain reach steady state values after a short transition period. In fact, for time invariant processes, the optimal, steady state value of the Kalman gain can be calculated off-line before starting the process, and this gain can then be used for every run from start to finish [94, 96]. In the case of the multi-

tool/product system, the $C$ matrix varies amongst the different values of $C_r$ depending on which product-tool combination is being run. In this case, the matrix $P$ is not able to reach a steady state value, and as a result, neither is the Kalman gain.

### 3.6 Effect of Qualification Runs on System Performance

It was previously stated that the usage of qual runs causes the incursion of additional costs; therefore, it is important to gage the importance of the qual runs to the control system. It is stated in Wang et. al [123] that the outputs of a multi-tool/product system can be controlled well by a non-threaded control system without qualification runs because the controller does not need exact estimates of the individual states, but rather an exact estimate of the predicted output for the context(s) being run. The input move calculated by the controller model (Equation (3.30)) is a function of the difference between the output target, $T$, and the predicted open loop output based on the state estimates. It is therefore unimportant from a control stand point that the individual states be exact estimates of the true states as long as the sums of the individual states for each context are good estimates of the outputs for the contexts. It is also shown in [123] that a system without quals can give unbiased estimates of the outputs, but unbiased estimates of the individual tool and product states cannot be found. Therefore, qual runs are not strictly necessary for good control of the system outputs, but their use does offer the advantage of accurate estimates of the individual tool and product states. As stated in [93] and [127], this can be very useful in terms of process monitoring; therefore, including qual runs could be a good strategy in a system where improved monitoring provides a cost savings that offsets the added costs of running qualifications.

To demonstrate the effects discussed above, a series of simulation studies are performed in which the number of qualification runs, as a percentage of the total number of runs, is varied from zero to ten percent in increments of one percent. The basic system

(called Test 1) under consideration has three tools and three products with an equal percentage of each product in the product mix. Additionally, each product has its own distinct processing time which also differs from the processing time of a qual run. In order to demonstrate the effects of different process variables, five other system setups are tested in addition to the basic system (Test 1). In the Test 2 system, the processing times for the qual runs and products are reversed. In the Test 3 system, the traffic intensity (a measure of the product volume in the system) is decreased from a high volume setting to a low volume setting, and in the Test 4 system, the product mix is unevenly divided among the three products. Finally, the Test 5 system has six tools instead of three, and the Test 6 system has six products instead of three. Table 3.4 summarizes the variable settings for the five test cases.

| Variable \ Test | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Num of Tools** | 3 | 3 | 3 | 3 | 6 | 3 |
| **Num of Prods** | 3 | 3 | 3 | 3 | 3 | 6 |
| **Product 1 Processing Time** | 1 | 4 | 1 | 1 | 1 | 1 |
| **Product 2 Processing Time** | 2 | 3 | 2 | 2 | 2 | 2 |
| **Product 3 Processing Time** | 3 | 2 | 3 | 3 | 3 | 3 |
| **Product 4 Processing Time** | - | - | - | - | - | 4 |
| **Product 5 Processing Time** | - | - | - | - | - | 5 |
| **Product 6 Processing Time** | - | - | - | - | - | 6 |
| **Qual Processing Time** | 4 | 1 | 4 | 4 | 4 | 7 |
| **Product Mix*** | All 1/*p* | All 1/*p* | All 1/*p* | (0.6, 0.3, 0.1) | All 1/*p* | All 1/*p* |
| **Traffic Volume** | High | High | Low | High | High | High |

Table 3.4: Summary of variable settings for different test cases; differences from the base case are highlighted. *Indicates the fraction of each product in the mix where *p* is equal to the number of products.

To test the performance of the estimator and controller with different disturbance models, four different types of test signals are used with each having a different set of disturbances added to the states. Test signal one (Sig1) has only stationary white noise added to the states. In the other three tests (Sig2-3), non-stationary disturbances are added to the states. For Sig2, IMA(1,1) disturbances (see Appendix C for more information on IMA(1,1) models) are added to all states. For Sig3, shift and drift disturbances are added to the tool states with only white noise added to product states, while Sig4 employs a combination of disturbances with shifts and drifts added to tool states and IMA(1,1) added to the product states. In all cases, small white noise disturbances (compared to the size of the state disturbances) are added to the outputs. A summary of the disturbance signals for the states can be found in Table 3.5.

| Signal | Description |
|---|---|
| **1** | White noise added to all states |
| **2** | IMA(1,1) disturbance added to all states |
| **3** | White noise added to all states, shift and drift added to tools |
| **4** | IMA(1,1) added to product states; white noise, shift, and drift added to tools |

Table 3.5: Summary of disturbance signals applied to test systems.

Each simulation is a combination of three factors: test system, disturbance signal, and qualification percentage, and can be labeled for easy identification as (Test, Signal, Qual Percentage) = ($i$,$j$,$k$) with $i \in [1,5], j \in [1,4], k \in [0,10]$. Each simulation is run over a campaign length of 500 runs with qual runs spaced as evenly as possible while also trying to insure an equal distribution of qual runs across the tools. The root mean squared errors for outputs (RMSE$y$) and state estimates (RMSE$x_{est}$) are recorded over the last 450 steps. Note that the RMSE$x_{est}$ is the sum of errors over all states. Each of the experiments is replicated 50 times with the results being averaged to get smoothed values

of RMSE$y$ and RMSE$x_{est}$. The estimates for tool states are initialized at the true values while the initial values for product states have a 50% error. The matrix $R$ is diagonal and has entries equal to the covariances of the white noise disturbances added to each output. The only exception is that the elements of $R$ corresponding to qual runs are set to a value one order of magnitude smaller than those of other outputs so the estimator has greater sensitivity to these measurements. Initially, the matrix $Q$ is also diagonal with entries equal to the covariances of white noise signals used to generate the disturbances for each state; call this matrix, $Q_{plant}$. However, this tuning is found to be too aggressive, especially in systems with delays (see Section 4.5.2.2), and is decreased for better noise rejection; the final value is equal to $Q_{plant}$ /5000.

Results for the base case (Test 1) with all four disturbance signals are shown in Figure 3.1. Note all output error points in Figure 3.1 are scaled by the RMSE$y$ value for the Test1, Sig1, zero qual (1,1,0) case for easier comparison and plotting; the same is done for estimate error points using RMSE$x_{est}$ (1,1,0). As the qual percentage is increased from zero to ten, it is clear that the impact on the output error (solid lines in Figure 3.1) is minor except for systems with shift and drift disturbances (Sig3 and Sig4); for these cases, most of the improvement is seen in the transition from zero quals to one percent quals. For the state estimate error (dashed lines in Figure 3.1), there is large drop in error from zero quals to one percent quals which is due primarily to the initial offset between the product state estimates and the actual states. Without a way to get unique measurement of any of the states, the zero qual system is never able to overcome this initial error. Once quals are introduced to the system, however, the rate of decrease in estimate error is much slower after the one percent qual level; this is evident from the plots of the state estimate errors which are relatively level in the section that spans one percent quals to ten percent quals. As with the output errors, decreases after the one

percent qual level are more noticeable in the estimate errors for the cases with disturbance signals containing shift and drift errors. This is due primarily to larger errors immediately after shifts, and the addition of extra quals aids in the mitigation of such errors.

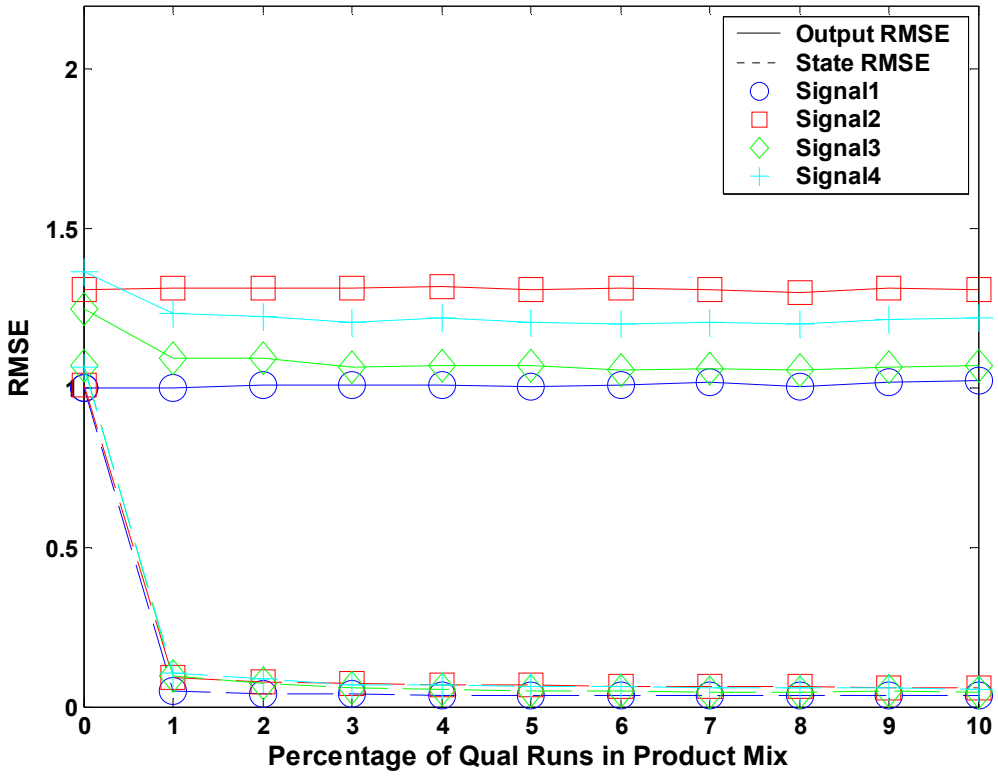


Figure 3.1: Output error from target (solid lines) and state estimate error (dashed lines) for Test1 system with four different disturbance models.

Simulation results for Test 2 (product processing times reversed from the base case), Test 3 (plant run in lower volume regime), and Test 4 (unbalanced product mix) are similar to those of Test 1 and are therefore, not shown here. The effects of the variable changes for these cases are more interesting once the scheduling portion of the system is added to the simulations (see Chapter 4) because the idealized system used here

treats the outputs as if they are run in series rather than representing the outputs true behavior in the plant where several products can run in parallel. There is one interesting observation contained in the Test4 results; the unbalanced product mix of this system causes the output and estimate errors for the three products to be quite different from one another. Despite the fact that the sums of errors over all products or all states in Test 4 are similar to those of the base case (where individual product errors are roughly the same for all products), the individual errors for each product in Test 4 are inversely proportional to their percentage in the product mix (i.e., product 1, which is 60% of the mix, has the lowest error while product 3, which is 10% of the mix, has the highest). This effect is expected since more frequently run products have more samples available for feedback to the estimator.

The results for Test5 (six tools) and Test6 (six products) are plotted in Figures 3.2 and 3.3, respectively. Note that like Figure 3.1, all data points in Figures 3.2 and 3.3 are scaled by the case (1,1,0) values which allows for comparison across tests. In comparison to the base case (Test 1), the output errors for a system with purely white noise disturbances (Sig1) perform about the same when additional states are added. However, an increase in output error is more apparent in the larger systems (Tests 5 and 6) when non-stationary disturbances are injected (Sig2–4). The addition of extra states has little effect on the estimation errors except in the zero qual case; notice that the RMSE$x_{est}$ values for (4,*j*,0) are smaller than those in (1,*j*,0) while the same values in (5,*j*,0) are larger. As was discussed earlier, the zero qual system has difficulty overcoming the initial 50% errors of the product states, so in Test 5, where the number of products is larger, the estimate errors for the zero qual case are larger and in Test 6, where the number of tools is larger, the estimate errors are smaller.

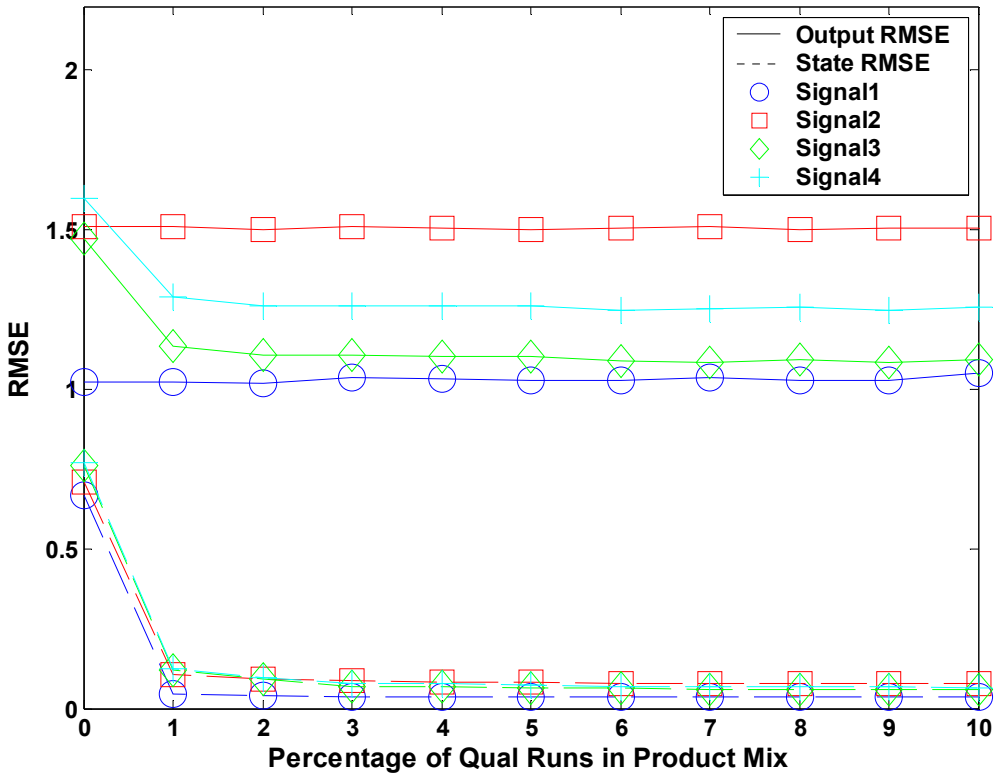


Figure 3.2: Output error from target (solid lines) and state estimate error (dashed lines) for Test 5 (six tool) system with four different disturbance models.

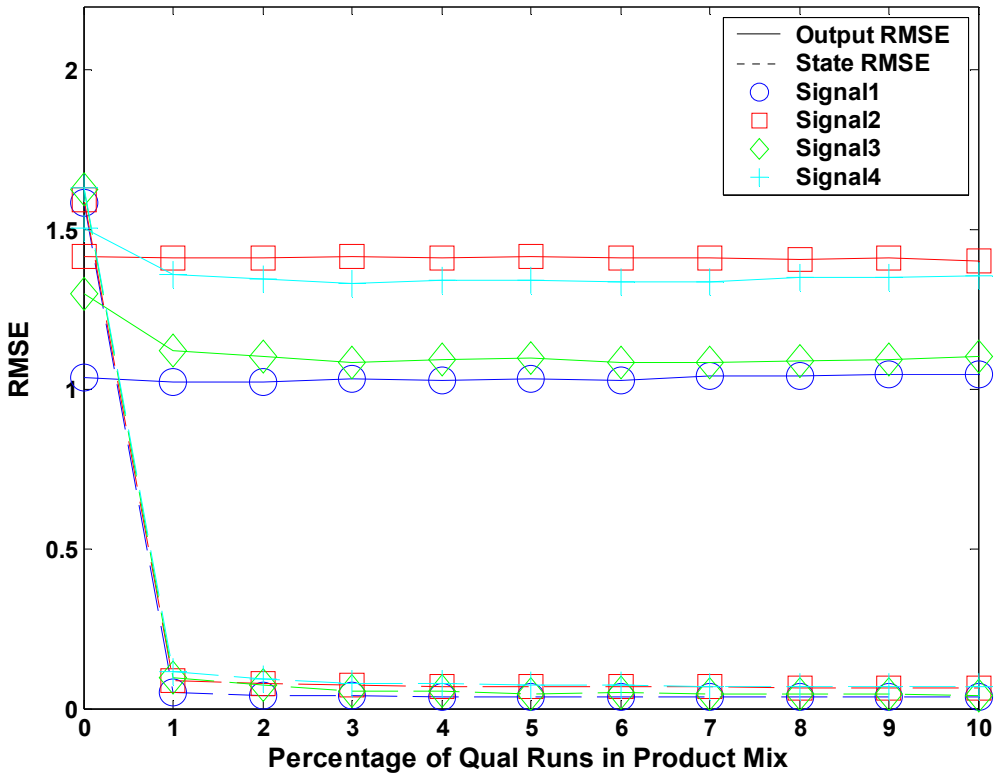


Figure 3.3: Output error from target (solid lines) and state estimate error (dashed lines) for Test 6 (six product) system with four different disturbance models.

## 3.7 CONCLUSIONS

In this chapter, the methods described by Pasadyn [41, 53] for the non-threaded control of a multi-tool, multi-product system are reviewed and studied. Included are a state space model of the system, a run-varying Kalman filter for estimation, and a controller based on model inversion. In addition to some review of previous observability analysis of this system based on the full system matrices, analyses based on the partial and pathwise observability of the run-varying system are also performed. These analyses are important because they form the basis for scheduling algorithms presented in the succeeding chapter. Also, the system model is adjusted to eliminate

additional states that track the inputs to the system and to allow for multiple inputs. This has the added benefit of increasing the stability of the observer as well.

Finally, the effects of varying percentages of qual runs in the system are studied under a wide range of conditions. It is found that the percentage of qual runs used has little effect on the estimation error after the three to five percent level is reached. Furthermore, the output errors show very little improvement in performance with additional quals; only the systems with shift and drift disturbances show any improvement. The models and methods described in this chapter are important as they are used in the next chapter to create and test a combined scheduling and control system.

# Chapter 4

# Integration of Process Control and Scheduling with Control-friendly Scheduling Algorithms

## 4.1 INTRODUCTION

After establishing the necessary modeling approach for the multi-tool, multi-product system in the previous chapter, the current chapter identifies methods for using the control system model to make scheduling decisions with the goal of improving estimation and control performance. Control-friendly scheduling algorithms are developed based on the partial observability analysis performed in the previous chapter and are compared with other methods from previous work on both the basis of control and scheduling performance. In order to do these comparative studies, simulations for both systems are constructed and integrated.

## 4.2 USE OF PROCESSING DECISIONS TO IMPROVE ESTIMATION

### 4.2.1 Objective Function Based on Estimate Error Covariance

After observing that the error covariance matrix continuously varies over the course of the runs in the multi-tool/product system, Pasadyn [41, 54] proposes the idea that by manipulating the order of the runs, the performance of the estimator can be improved. A simple one tool, one product system with two states and two outputs (i.e., a production run and a qualification) is used to demonstrate the effect that processing order can have on the state estimate variance. Using simulation experiments, he shows that a series of runs that alternates between the two outputs gives better estimation results than two alternative sequences which use long runs of each product consecutively (i.e., tool dedication). When saying the estimation is "better," it is in the sense that, of the three

orderings, the alternating schedule gives the minimum variance for the state estimation error (i.e., the diagonal elements of the *P* matrix are kept the smallest). A re-creation of the above experiment can be found in Appendix C and can be used for comparison to other methods presented later this chapter.

Building upon the simple example above, Pasadyn [41, 54] discusses ways in which the quality of the estimation can be improved by changing the manner in which processing decisions are handled. Starting with a reference set of twenty runs in which the product-tool pairings and processing order are already determined, three methods for minimizing the estimate error over the sample set are suggested:

1. While maintaining the product-tool pairings and order of processing in the reference set, and assuming that measurements can only be made on a limited number of runs, determine which runs to measure (i.e., create a sampling plan).
2. Given a sampling plan and maintaining the assignment of products to tools in the reference set, choose the order in which the runs should occur.
3. Given a sampling plan and maintaining the order of products in the reference set, choose the tool on which each product should be run.

In all cases, the objective function to minimize utilizes the trace (i.e. sum of the diagonal elements) of the error covariance matrix at each step. The purpose of this design is to reduce the total estimate error over all states in the system over the course of the twenty runs. In mathematical terms, the objective function, *J*, can be written as:

$$J = \sum_{i=1}^{20} trace(P_i). \tag{4.1}$$

To obtain the lowest total estimation error in each case above, an optimization problem is solved by changing the decision variables in order to minimize *J* in Equation (4.1). The decision variables for cases 1–3 are, respectively, the sampling plan, the

processing ordering, and the assignment of products to specific tools. Of the three problems listed, only the first does not have a solution space that grows exponentially as the number of states increases. For Case 1, the global optimum can be found by complete enumeration for smaller systems, but the solution space for the other two systems is too large for such an approach. Instead, the author uses a suboptimal approach where the context (Case 2) or tool (Case 3) selected for the next run is the one in the available set that gives the maximum reduction in trace($P$) for the current step [41, 54]. This is equivalent to sequentially minimizing $J$ at each step and can be categorized as a greedy heuristic.

While trace($P$) is a sensible choice for the control-friendly scheduling algorithm it is not the only one of several objectives based upon the estimate error covariance matrix that is tested here. One such alternative objective is the trace of $CPC^T$, which is a transform of the estimate error covariance matrix and represents the output error covariance matrix; the trace of this matrix is equal to the total variance of the output estimate error. The next $P$-based objective, which is similar to the previous one but is calculated only on the basis a single output, is $\text{trace}\left(C_r P C_r^T\right)$. Here only the variance of the output (or outputs in the case of simultaneous starts) that is used at each step is considered. The final objective using $P$ to be tested is the condition number of $P$. Because condition number is measure of numerical stability, the condition number of the error covariance matrix gives an indication of the accuracy of the estimates.

To gain a better understanding of the behavior of the objectives that use the trace of $P$ or one of its transformations, it is useful to expand the matrix in symbolic form and then perform the trace operation to see which entries of the error covariance contribute to the final objective function. Additionally, the effect that particular measurement actions have on the entries of the error covariance matrix is important. Viewing the effects that

different measurements have on $P$ through the lens of the objectives in equation form gives a good insight into the algorithmic scheduling process.

First the response of $P$ to different measurements is discussed. In general, a qualification run decreases variance for all state errors (represented by the diagonal elements of $P$) and also the absolute values of the covariances between state errors (the off-diagonal elements). The largest reductions are incurred by the variance and associated covariances of the tool state being qualified. For measurements of production runs, the variance terms for the product and tool states that compose the output being measured decrease while the variance terms of the other states increase. Also, the absolute value of the covariance between the measured product/tool combination increases while other covariance terms are affected very little. The factors that affect the size of these changes are the size of the terms in the noise covariance matrices $Q$ and $R$ and the frequency with which states are measured. Because the time update step of the Kalman filter adds $Q$ in the update of $P$, states with larger noise and those that are measured less frequently tend to have larger variance terms in $P$, and the measurement of such states leads to larger drops in variance at the measurement update step. Finally, it is noted that, normally, covariance within tools and within products is positive, while covariance between tool/product combinations is negative.

The simplest of the three objectives uses trace($P$) which is the sum of the variances of the state estimate errors. In equation form, the objective is

$$\text{trace}\left(P\right) = \sum_{i=1}^{t+p} P_{ii} = \sum_{i=1}^{t+p} \text{var}\left(\tilde{x}_i\right) \tag{4.2}$$

where $\tilde{x}_i$ is the estimate error for the ith state, $t$ is the number of tools in the system, and $p$ is the number of products. Accordingly, a qual run always reduces this objective, while a production run will have varying effects depending on the size of increases and

decreases in the variance terms. Typically, the longer a product goes without measurement, the larger its variance term becomes so that its selection is preferred when it does become available.

The next objective uses the trace of the matrix $CPC^T$. Recall that $C$ is a matrix consisting of ones and zeros that is used by the output equation to combine tool and product states to make the various outputs. The transformation, $CPC^T$, represents the error covariance matrix of the outputs, and its trace is equal to the sum of variances of the output errors. The objective can be expressed as

$$\begin{aligned}\operatorname{trace}\left(CPC^T\right) &= (p+1)\left(\sum_{i=1}^{t}\operatorname{var}\left(\tilde{x}_i\right)\right)+t\left(\sum_{j=t+1}^{t+p}\operatorname{var}\left(\tilde{x}_j\right)\right)+2\left(\sum_{i=1}^{t}\sum_{j=t+1}^{t+p}\operatorname{cov}\left(\tilde{x}_i\tilde{x}_j\right)\right)\\ &= (p+1)\left(\sum_{i=1}^{t}P_{ii}\right)+t\left(\sum_{j=t+1}^{t+p}P_{jj}\right)+2\left(\sum_{i=1}^{t}\sum_{j=t+1}^{t+p}P_{ij}\right).\end{aligned} \tag{4.3}$$

It is apparent from Equation (4.3), that the objective is a weighted combination of variance and covariance terms from $P$. Interestingly, the product variance terms are weighted by the number of tools while the tool variances are weighted by the number of products plus one; the plus one is due to the quals. Also, the only covariance terms included are those between tool/product pairs. Because tool/product covariance terms are negative, quals tend to increase the third term of the objective while product runs tend to decrease this term. As with the trace($P$) objective, a qual run reduces the first two terms, but now at least some of these decreases are offset by the increase in the third term due to inclusion of covariance, and the likelihood of selecting a qual is decreased. Conversely, product runs now have an additional decreasing term due to the covariance, and are more likely to be selected with this objective than with trace($P$).

The final objective function studied in this section is $\text{trace}\left(C_r P C_r^T\right)$, where *Cr* represents a reduced form of the *C* matrix and is composed of the row of *C* corresponding to the output being run. The equation form of the objective for a product run is

$$\begin{aligned}\text{trace}\left(C_r P C_r^T\right) &= \text{var}\left(\tilde{x}_i\right) + \text{var}\left(\tilde{x}_j\right) + 2\,\text{cov}\left(\tilde{x}_i \tilde{x}_j\right) \\ &= P_{ii} + P_{jj} + 2P_{ij}.\end{aligned} \tag{4.4}$$

which shows that it is made up of the variance and covariance terms for the tool, *i*, and product, *j*, that compose the output. For a qual run, it is simply the variance of the tool being measured. Because a product run causes all three terms of Equation (4.4) to decrease, once a particular tool/product combination has been measured, this objective favors running the same output over and over. It is common with this objective to see long stretches of the same product being run on a particular tool.

### 4.2.2 Objective Functions Based on Partial Observability Analysis

In an attempt to develop other algorithms for comparison to the *P*-based heuristics above, the methods of Section 3.3.1 can be employed. A dispatching method based on the research of Chen [116] was already discussed in Section 3.2.1.1 and may work well for larger systems with a single output. Here the focus is on the work of Hong et al. [117] since it is developed for a broader range of systems. As was explained in the earlier section on partial observability, the degree of observability for a time varying system can be quantified by performing a SVD on the information matrix. Because the smallest singular value, $\sigma_n$, is associated with the least observable subspace (LOSS, for short), one can deduce that this subspace is most in need of measurement information in order to become more observable. Therefore, the state(s) associated with the LOSS should be measured as soon as possible.

Because the rows of the output matrix, $C$, dictate which subspaces of the state space are measured at each output, it is logical to compare the rows of $C$ to the least observable singular vector of the information matrix. By doing such a comparison, the measurement which needs to be taken in order to increase the observability of the system can be determined. To quantify the similarity between the singular vector in question and the rows of $C$, a measure of colinearity can be used (this is possible because all vectors in question have length $n$). Finding the degree of colinearity between the measurable subspaces and the LOSS forms the basis for a new algorithm to select the next output to be run and measured when a tool becomes available. First, the angles between each row of $C$ corresponding to the available tool and the least observable singular vector are calculated:

$$\angle_{t,p} = \cos^{-1}\left(\frac{C_{t,p}^{T}}{U_{\min}}\right), \tag{4.5}$$

where $C_{t,p}$ represents the row of $C$ which corresponds to the available outputs at the current step, $t$ is the tool selected from the set of available tools at the current step, and $p$ is the product selected from the set consisting of all products and the qual(s) for the available tool(s). $U_{min}$ is the singular vector corresponding to the smallest singular value of $\Sigma_k$.

After all angles have been calculated, the row of $C$ with the smallest angle between itself and the singular vector is selected because it represents the measurable subspace that is closest to being collinear with the least observable subspace. Finally, the product(s) and tool(s) associated with the selected row of $C$ are chosen for processing the next run. This procedure is the same as running the following optimization problem at each step:

$$\underset{t\in\bar{T},p\in P,\bar{Q}}{\arg\min}\left(\angle_{t,p}\right) \tag{4.6}$$

where $\bar{T}$ is the set of available tools at the current step and $\bar{Q}$ is the set of quals corresponding to the tools in $\bar{T}$.

The authors of [117] also show that the smallest singular value is the measure of the overall degree of observability for the system. The smaller the value of the minimum singular value, the more unobservable the system is. Therefore, a second objective is developed that seeks to choose the measurement from the outputs available that causes the largest increase in $\sigma_n$.

### 4.3 COMPARISON OF CONTROL-FRIENDLY SCHEDULING OBJECTIVES ON SINGLE TOOL/SINGLE PRODUCT SYSTEM

To demonstrate the performance of scheduling systems that use performance indices based on the estimate error covariance and partial observability, several small scale simulations are performed. For the sake of comparison, the single tool, single product system from Section 4.2.1 and Appendix E is used, but rather than look only at the effect of product ordering on the estimate error covariance of the states, the values of several different objective functions are observed and compared.

The simulated system from Appendix E has two states (one tool, one product) and two outputs (one production run and one tool qualification run). The same set of three simulations is repeated and the values of the following different possible objective functions are observed:

1. Trace of the estimate error covariance matrix, $P$. Proposed by Pasadyn, and discussed in Section 4.2.1, this objective is the sum of the theoretical error variance for all state estimates (tools and products) in the system. The goal is to choose the outputs which, when measured, will give the largest reduction in the trace of $P$.

2. Degree of colinearity. As discussed in the previous section, this objective is calculated by finding the angles between the subspaces of the available contexts and the least observable subspace of the system state space. The subspaces for the available contexts are represented by the rows of the $C$ matrix that correspond to available tools and products, and the least observable subspace is represented by the singular vector corresponding to the minimum singular value of the information matrix, $L_{0,k}$. When using this objective, the context with the smallest angle is chosen for the next run because it will provide the best improvement in system observability.
3. Degree of observability. Also discussed in the previous section is the fact that the smallest singular value of $L_{0,k}$ gives a measure of the relative degree of system observability. With this objective, the context that will give the largest increase (or smallest decrease) in the degree of observability is chosen to be run at the next step.

For each of the three sample schedules used in the simulations of Section 4.2.1 and Appendix C, Figures 4.1–4.3 show (a) the values for the estimate error covariance matrix, (b) the trace of P, (c) the degree of colinearity, and (d) the degree of observability.

Figure 4.1 and 4.2 show the simulation results when a tool dedication schedule is used. In Figure 4.1, the schedule used consists of 25 consecutive runs of the product followed by an equally long run of tool qualifications, while the opposite schedule (qual runs followed by product runs) is used to generate Figure 4.2. Plot (a) in each figure shows the behavior of the individual error variances for each state while plot (b) shows the trace of $P$. Because the trace of $P$ is simply a sum of the individual estimate error variances for all of the states in the system, plot (b) shows similar behavior to plot (a) in both figures. In Figure 4.1, the large decrease in individual variances at the switching

point in plot (a) is also seen for the total variance in plot (b), and in Figure 4.2, the large decrease in product state error variance and small increase in tool state error variance apparent in plot (a) are seen as a large decrease in the total variance value seen in plot (b). Interestingly, the behavior of the plot of total variance, plot (b), is almost identical for both Figures 4.1 and 4.2, even though the two schedules are the reverse of one another. This occurs because, in this simulation example, the noise covariance matrix for the state equation, $Q$, is defined to be the identity matrix (i.e., both state noise variances have a value of one). Having different noise variance values for each state (as would be more common in practice) would lead to less similar patterns in plots of the trace of $P$ for different schedules.

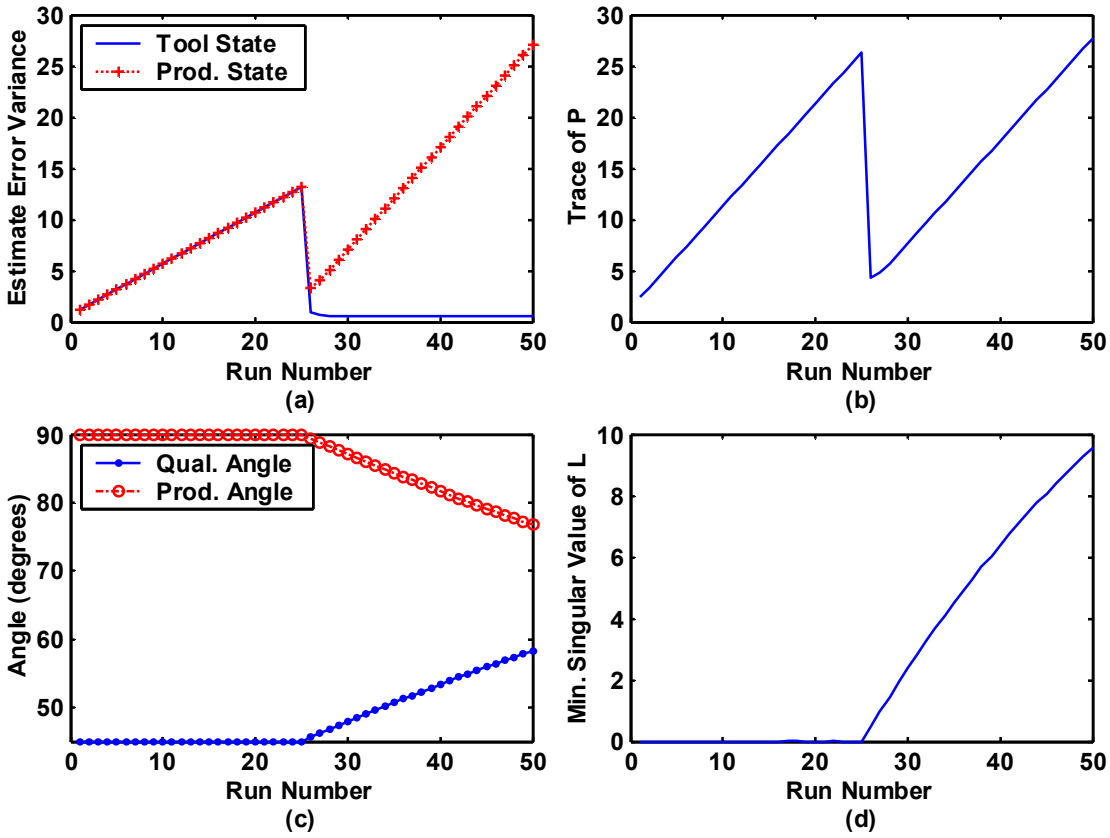


Figure 4.1: Single tool, single product system with tool dedication schedule; the first 25 runs are of the product and the last 25 runs are qualifications. (a) Estimate error variance for each state. (b) Total error variance for the system (trace of P). (c) Degree of colinearity of each output to the least observable subspace. (d) Minimum singular value of the information matrix, L0.

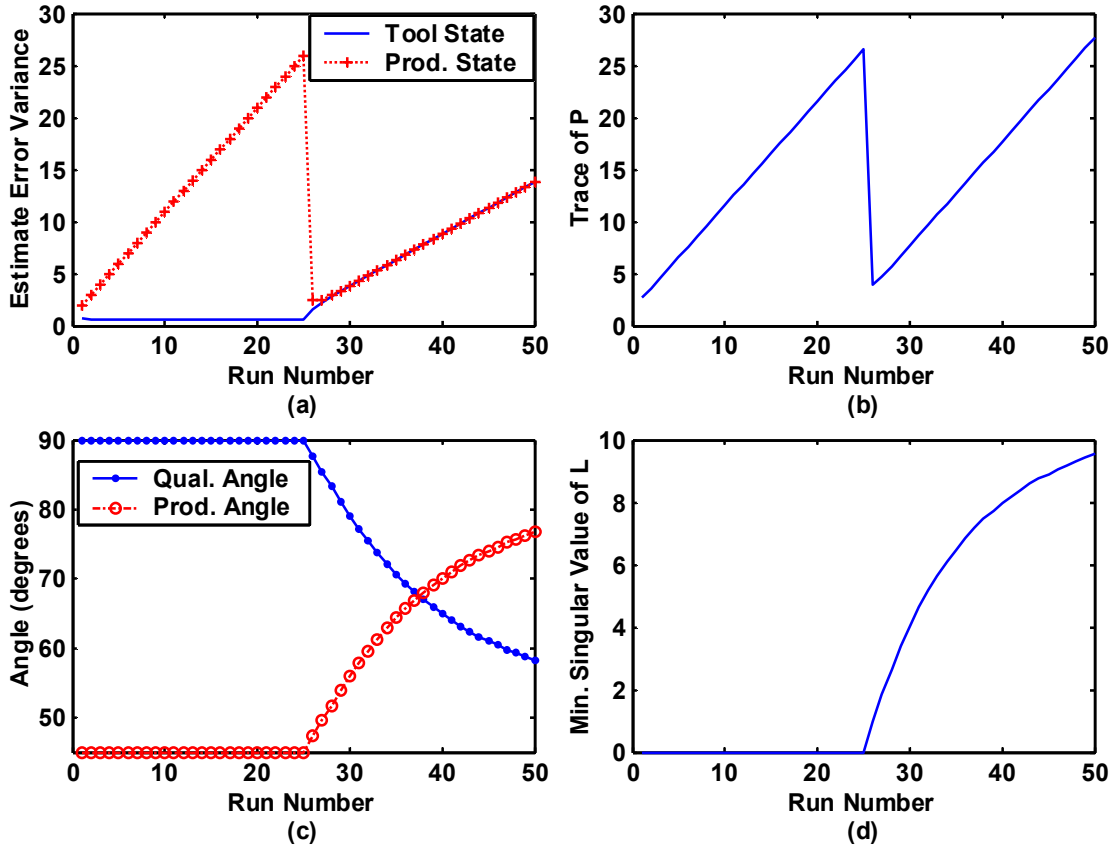


Figure 4.2: Single tool, single product system with tool dedication schedule; the first 25 runs are qualifications and the last 25 runs are of the product. (a) Estimate error variance for each state. (b) Total error variance for the system (trace of P). (c) Degree of colinearity of each output to the least observable subspace. (d) Minimum singular value of the information matrix, L0.

For the third schedule in which single product runs and qual runs are alternated, the individual estimate error variances and total estimate error variance remain low (Figure 4.3, plots (a) and (b), respectively) with the value for the trace of *P* reaching a steady state rather quickly. After the system passes through a brief transition period, the increase in variance of one state estimate becomes equal to the decrease in variance of the other state estimate from one run to the next (see Figure 4.3, plot (a)), which results in no net change in the total estimate error variance for the system. The steady state value for

the trace of $P$ is also a result of the simulation set up (i.e., being the identity matrix). Given non-identical values for the state noise variances, a stable pattern that looks like a triangle wave develops in the plot of the total estimate error variance (i.e., the trace of $P$) because changes in each individual estimate error variance, while still opposite in sign, are no longer equal.

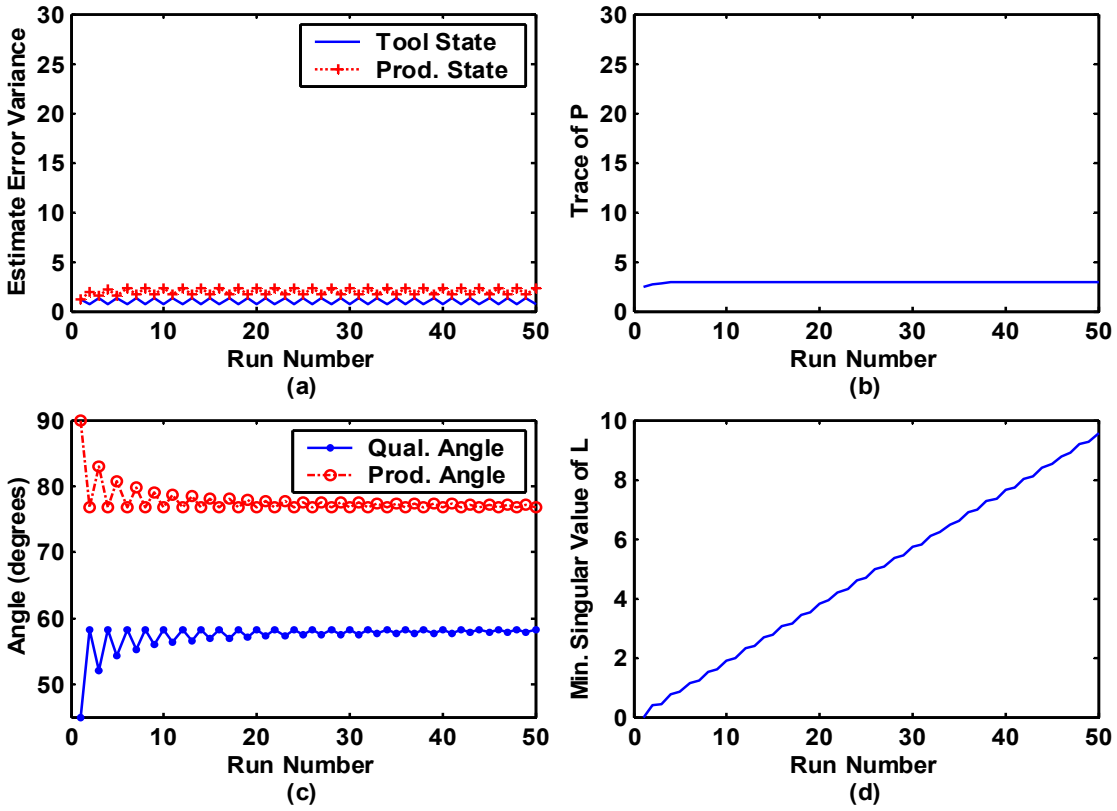


Figure 4.3: Single tool, single product system with an alternating schedule; the 25 product runs are alternated with the 25 qualifications. (a) Estimate error variance for each state. (b) Total error variance for the system (trace of P). (c) Degree of colinearity of each output to the least observable subspace. (d) Minimum singular value of the information matrix, L0.

Also shown in Figures 4.1–4.3 are the values of the other two objective functions discussed in the sections on partial observability. Plot (c) in each figure shows the degree of colinearity (represented by an angle in degrees) of the two output subspaces in relation

to the least observable subspace. For the two tool dedication simulations, the plots of the degree of the colinearity show very different behavior. In Figure 4.1, the initial long campaign of product runs causes the subspace associated with the quals to be closer to the least observable subspace as expected. After the switch, the two angle values for the output subspaces move toward each other as the product subspace becomes less observable and the qual subspace becomes more observable during the long string of qual runs.

When the order of the two campaigns is switched in the next simulation, the behavior of the two angle values is also reversed. In Figure 4.2 (c), the rate of change of the angle values after the switching point is much more rapid than in the first plot. In the first case, the two angles move slowly toward each other, but in the second case, their plots actually cross over one another with the qual subspace becoming closer to the least observable subspace after the $12^{th}$ product run ($37^{th}$ run overall). Clearly, the system favors qual runs as the means to provide more information about the least observable subspace. This is also evident in the third scenario (Figure 4.3, plot (c)) where the two angles move toward their own steady state values but the qual angle remains smaller than the product angle.

For the tool dedication schedules, the behavior of the colinearity measures ((c) and (d)) is unlike that of the state estimate error variances ((a)) when looking within the two individual campaigns of 25 runs. For the variances, the order in which the two campaigns are run makes no difference. During the long series of product runs (runs 1–25 in Figure 4.1 and runs 26–50 in Figure 4.2), the two variances show the same trends; this is also true during long series of qual runs (runs 26–50 in Figure 3.1 and runs 1–25 in Figure 4.2). On the contrary, the colinearity values show a dependence on the order in which the two campaigns are processed.

One last difference between the colinearity angles and the state variances concerns the sensitivity of the values to changes in the schedule. It is apparent from the figures that the state variances change drastically at the switching points of the two tool dedication schedules, but the angles of colinearity change much more slowly. Therefore, it is suggested that a moving window approach may be used with the colinearity angles in order to speed up their response to changes in the schedule. Recalling the theory on local observability in Section 3.3.1 in which a path length of $n$ (where $n$ = total number of states) is suggested, a window length of two is used in this case to gage the effect of using the moving window approach with the angle based objective. Figure 4.4 shows the results for the same set of simulations run in the previous examples, but here, the angles are calculated from an information matrix based on only the last two steps. For the two tool dedication schedules, the angle values change much more drastically after the switching point than they did in the previous example. Rather than slowly move toward each other with an eventual crossover, the two angles completely exchange values within two steps. Similarly, with the third schedule, the angle values quickly reach their steady state values in two steps. It is clear a moving window approach allows the angle based objective to respond much more quickly to changes in the schedule.

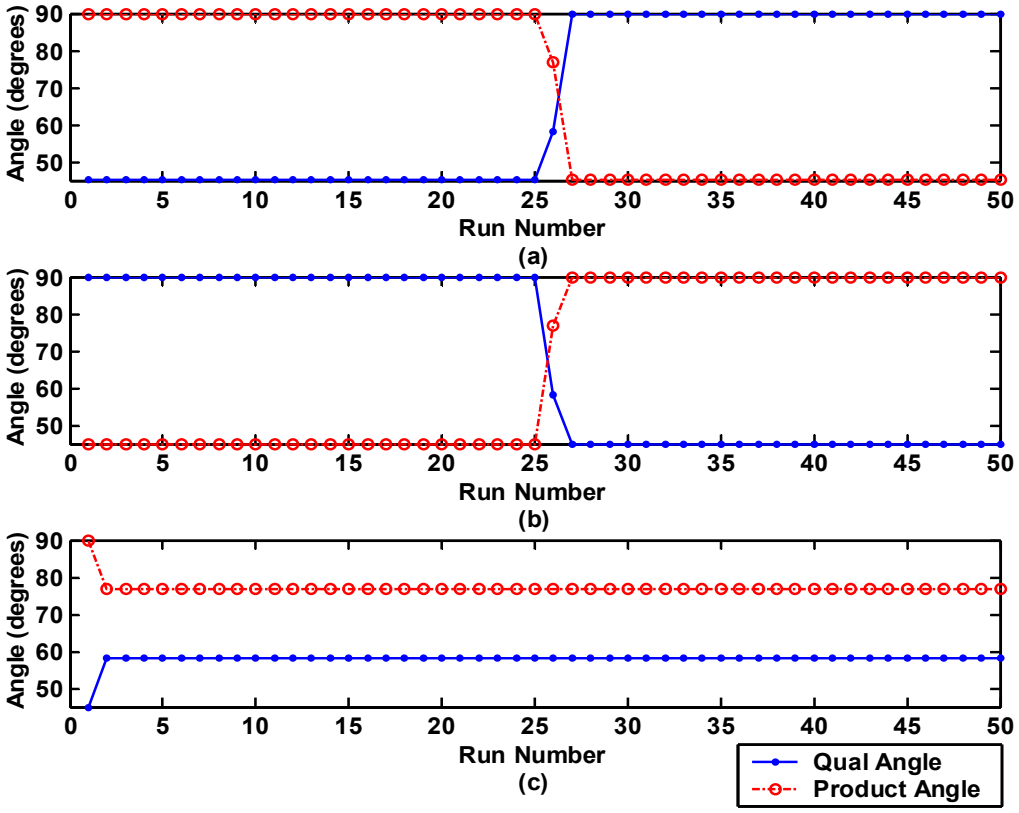


Figure 4.4: Degree of colinearity objectives using a moving window calculation (window size of two). (a) Schedule 1: 25 product runs followed by 25 qual runs. (b) Schedule 2: 25 qual runs followed by 25 product runs. (c) Schedule 3: One product run alternated with one qual run for 50 runs.

The final objective function presented is the smallest singular value of the information matrix (Figures 4.1–4.3, plot (d)). This is the singular value associated with the least observable subspace and its value increases as the system becomes more observable. While the order in which outputs are run does not cause the value to decrease, it does affect the rate of change of the value. Comparing the plots for the two tool dedication schedules shows that both have values of zero for the first 25 runs, but after the switching point, the second schedule (quals followed by products) causes the value of the smallest singular value to increase at a faster rate initially and then keeps it at

a higher value until the end of the simulation. This differs from the plots of the trace of $P$, which showed almost no difference between the two tool dedication schedules.

For the final schedule, which alternates the two outputs, the singular value of the least observable subspace rises steadily throughout the course of the simulation. In this case, a switch from a product run to a qual run causes a larger increase in the objective function than a switch from a qual run to a product run. Therefore, the singular value based objective seems to favor qual runs and shares this trait with the angle based objective function.

Interestingly, all three schedules result in the same final value for the smallest singular value, but the paths of the intermediate values are quite different. This is plainly seen in Figure 4.5 where the singular value trends for all three schedules are shown on the same plot. Judging by the mean value of the objective over the course of the runs, the alternating schedule (Schedule 3) maintains the highest degree of observability while the dedication schedule with production runs followed by qual runs (Schedule 1) maintains the lowest. The mean values for Schedules 1-3 are 2.72, 3.35, and 4.82, respectively.

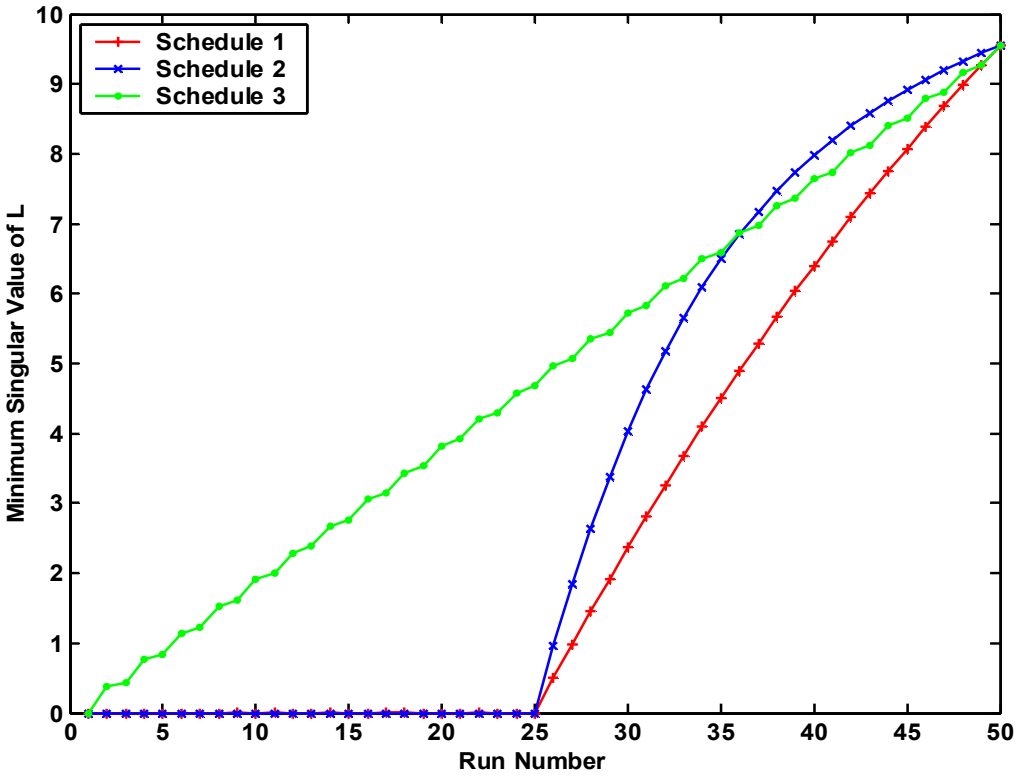


Figure 4.5: Values of the smallest singular value of the information matrix for the three different schedules on the single tool, single product system.

Finally, recall that the information matrix is a square version of the observability matrix and is scaled by $R$ (see Equation (3.27)). Therefore, in much the same way that estimate error variances (plots (a) and (b)) are affected by the state noise variance values in $Q$, the two objectives based on the information matrix (plots (c) and (d)) are affected by the values of output noise variances in $R$. When comparing scenarios where $R(1,1) \neq R(2,2)$ to the case where the variances are equal, the angle based objective indicates that the output with the higher noise variance tends to have a lower angle value and is therefore more likely to be selected for measurement. The observed trend in the angle data seems reasonable given that a higher output noise variance indicates a less

dependable measurement and more observations are needed for good estimation. For the singular value based objective, lowering the noise level of one of the outputs tends to increase the value of the objective and vise versa. Also, changing the noise level of the qualification output has a stronger effect on the objective than changing the noise level of the production run.

After observing the behavior of the various objective functions for the single tool/single product system, testing on systems with more tools and products is necessary to observe the effects on these added complexities. Additionally, because this work aims to gauge the effect of the control system on the scheduling system, the various schedules produced by the above control based objectives need to be evaluated against standard scheduling methodologies. Therefore, the larger scale simulations need to have a component that allows for assignment of processing times to the various products in the system and a way of tracking the movement of the products. The following section discusses the development of the scheduling portion of the simulations.

## 4.4 MULTI-TOOL/PRODUCT SYSTEM FROM AN OPERATIONS RESEARCH PERSPECTIVE

### 4.4.1 Introduction

Thus far, the main focus of this chapter has been on the modeling, control, and estimation of the multi-tool/product system and the development of algorithms for improving process control performance by manipulating the processing order and tool allocation of products. While good control performance is a key aspect of the manufacturing facility, it was stated in the earlier chapters that scheduling performance is also another important metric. Therefore, it is important that the schedules obtained from the control-friendly heuristics need to be compared to schedules which are considered to be optimal in relation to some performance indices which are more typically used in the

operations research community (i.e., from a traditional scheduling point of view). These types of criteria are usually based on the amount of time a product spends in the system (cycle time) and/or its completion date relative to its due date (tardiness). Another important measure of schedule performance is the rate at which a system is able to make products (throughput).

### 4.4.2 Discrete Event System Simulation

To evaluate the scheduling performance of the multi-tool/product system, a simulation which keeps track of the relevant time based metrics must be built. A discrete event simulation can be used to accomplish the aforementioned task. In a discrete event simulation, the state of the system is described by a set of variables whose values completely express the condition of the system at a given point in time. Further more, the state of the system is considered to change only at discrete points in time that correspond to some system related event [100].

For the simplified example of a single tool with multiple products, the events of interest are the arrivals of products to the buffer in front of the tool, the product starts on the tool, and the releases of products from the tool once processing is completed. The times at which these events occur are important variables. Other variables of interest are the status of the tool (i.e., busy or idle), the number of products in the buffer, the number of products in service, and the number of products in the system as a whole. Note that one of the events listed and several of the variables listed are redundant. The wafer start time for a given product is equal to either its arrival time or the completion time of the product ahead of it depending upon the value of the tool status. If we assume that the tool is only idle when there are no products available for processing (i.e., an empty buffer) and there are no tool breakdowns, the tool status itself is a function of the number of products in the buffer. It is also assumed that if the tool is not idle or broken down,

then the number of products in service is one. Finally, the number of products in the system is simply equal to the sum of products in the buffer and in service, so that only two of the three values need to be tracked. Table 4.1 shows a list of useful system variables. Given these variables, the following relations hold:

$$T_q = t_s - t_a, \tag{4.7}$$

$$T_q = t_s - t_a, \tag{4.8}$$

$$T_c = t_c - t_a = T_a + T_s. \tag{4.9}$$

The use of Equations (4.7)–(4.9) is discussed further in section 4.4.3.1 on the setup and analysis of the discrete event simulation used in this work, but first the use of queueing theory to analyze the system is discussed.

| Variable | Symbol |
|---|---|
| Arrival Time | $t_a$ |
| Start Time | $t_s$ |
| Completion Time | $t_c$ |
| Time in Queue (wait time) | $T_q$ |
| Service Time | $T_s$ |
| Cycle Time | $T_c$ |

Table 4.1: Time related variables for discrete event simulation

### 4.4.3 Queueing Theory Analysis

Because manufacturing systems like those found in the semiconductor industry consist of a tool or set of tools with buffers for holding in process material, they are often analyzed using queueing theory. A queueing system consists of a server (or servers) that will process the customers that arrive into the system. In the manufacturing case, the servers are the tools and the products are the customers. In the event that all servers are busy, there is a buffer (queue) where they can wait until a server becomes available. A

brief review of notation and terminology for queueing systems can be found in Appendix E (see also [100], Chapter 22 and references therein for a more detailed discussion).

Before proceeding further, some important system parameters and variables are defined. The two major parameters which are used to define the queueing system are the arrival rate, $\lambda$, which has units of products (arriving) per unit time, and the service rate, $\mu$, which has units of products (processed) per unit time. The inverses of these two variables are called, respectively, the mean interarrival time ($1/\lambda$) and the mean service time ($1/\mu$). The term $\rho = \lambda/\mu$ is referred to as the traffic intensity. Because it is normally desirable to keep the number of customers in the system bounded, $\rho$ is usually required to be less than one (i.e., the service rate is higher than the arrival rate). If $\rho > 1$, then the number of customers in the system will grow without bound because the service process can not keep pace with the arrival process. From a manufacturing perspective, $\rho$ is akin to the overall tool utilization for the system. The term $\pi_0$ represents the steady state probability that there are zero customers in the system, and for a system with no unforced idle time or breakdowns, it represents the percentage of time the system is idle. Depending upon the nature of the system, steady state probabilities for non-zero states may or may not be easily found.

While the arrival rate and service time describe how the system operates, it is also important to gage the performance of the queueing system. Typically one would like to know how much time (on average) customers spend in the queue waiting, in service, or in the system as a whole (i.e., the cycle time). It may also be interesting to know how many customers are typically in line (i.e., the mean queue length). Table 4.2 gives a list of the variables of interest with their symbols.

| Variable | Symbol |
|---|---|
| Mean number of customers in the system | $L$ |
| Mean number of customers in the queue | $L_q$ |
| Mean number of customers in service | $L_s$ |
| Mean time customer remains in the system | $W$ |
| Mean time customer remains in the queue | $W_q$ |
| Mean time customer remains in service | $W_s$ |

Table 4.2: Important steady state variables for queueing systems

A useful property of queueing systems is given by Little's queueing formulae, which state for any queueing system with a steady state distribution, the follow are true:

$$\begin{aligned} L &= \lambda W, \\ L_q &= \lambda W_q, \\ L_s &= \lambda W_s. \end{aligned} \tag{4.10}$$

These equations are quite useful in the analysis of queueing systems because if the average number of customers in the system, queue, or service area is known then the corresponding average time they spend there can be calculated.

There is great value in modeling a process in terms of a queueing system. If the process can be assumed to fit one of the standard queueing models it becomes possible (in many cases) to determine important system metrics such as average queue length ($L_q$), average cycle time of the products ($W$), etc., simply by specifying the parameters of the distributions for the arrival times and service times. Little's law can then be applied to find other variables of interest [100].

As an example, one of the most basic systems considered is the M/M/1 queue where the arrivals and service times are determined by Poisson processes (see Appendix E for a description of the relation between the Poisson distribution and queueing networks). Through the use of Markov chains, one can easily derive expressions for $L$,

$L_q$, and $L_s$, as well as the steady state probabilities for particular system states [100]. Once $L$, $L_q$, and $L_s$ are found, Little's Law can be used to find $W$, $W_q$, and $W_s$. Unfortunately, not all queueing systems allow for such easy and exact calculation of their mean, steady state values. In such cases, many researchers have found approximations for these values or failing at that, upper and/or lower bounds.

#### *4.4.3.1 Single Tool Discrete Event Simulation with Queueing Theory Verification*

As mentioned in the previous section on queueing, there is much work on finding theoretical calculations for the steady state values of various queueing systems. Therefore, it makes sense to frame the process model of the system being studied in terms of a queueing system for which theoretical expressions of the steady state values exist. The steady state results of the discrete event simulation of the same system can then be checked for accuracy against the theoretical results from the queueing theory.

As a starting point for the system simulation, a single tool, two product system is first modeled. The arrivals of products to the tools are modeled by a Poisson process so that the interarrival times are exponential. For the semiconductor system, each product is assumed to have its own constant processing time so that the service process is assumed to have a general distribution, and a first in first out (FIFO) scheduling discipline is used. In Kendall-Lee notation, this is considered an M/G/1 queueing system, and its steady state values can be calculated by using the Pollaczek-Khinchin (P-K) mean value formulae [100]. The equation for the mean number of products in queue is

$$L_q = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)}, \tag{4.11}$$

where $\sigma^2$ is the variance of the service time. Using Equation (4.11), Little's law, and the following two relations:

$$W_s = 1/\mu \tag{4.12}$$

$$L = L_q + L_s, \tag{4.13}$$

all of the mean values in Table 4.2 are easily obtained.

When setting up the discrete event simulation for an industrial system, the actual arrival rate of products and product service times can be derived from process data, and the traffic intensity can then be calculated. In this work, the service times for different products are specified in order to study their effects on scheduling and the traffic intensity is then specified to study different plant conditions (i.e., light traffic versus heavy traffic). Given the service times and traffic intensity, the appropriate value for arrival rate is then calculated. Additionally, the product mix must also be specified. The percentage of each product in the mix determines which product randomly arrives at each arrival time and is also used with the deterministic service time for each product to calculate the mean and variance of the overall service time (i.e., the first and second moments of the service time distribution).

With the values mentioned in the previous paragraph in hand, the discrete event simulation is first run for a single tool, two product system in which the products both have the same processing time. This is equivalent to a single product system and is used first since the Pollaczek-Khinchin equations are intended for use with such a system. In the second test of the simulation setup, the two products are given different processing times so that the first two moments of the service rate must be calculated using the percentages from the product mix. The values used for variables which must be specified at the beginning of the simulation are given in Table 4.3. Results for testing of the single product equivalent and two product systems with moderate traffic intensity ($\pi_0 = 0.75$) are shown in Figure 4.6 and Figure 4.7, respectively. Results for a high traffic intensity system are similar and can be found in Appendix G. In the figures, the green dashed

lines indicate theoretical values from P-K equations while red dots indicate the mean values from simulations and blue bars show the 95% confidence intervals. Steady state (i.e., long run) values of the system variables demonstrate good matching between the simulations and the theoretical results as the number of runs (indicated on the x-axis) increases. This is evident in the figures where the mean values for the variables of interest approach the theoretical values given by Pollaczek-Khinchin and also have smaller 95% confidence intervals. Assured that the setup of the discrete event simulation is correct for the smaller system, it is safe to expand it to larger systems for further study. For more detail on simulation setup and testing, see Appendix G.

| **Variable** | **Test 1 Value** | **Test 2 Value** |
|---|---|---|
| Product 1 service time | 2 | 2 |
| Product 2 service time | 2 | 0.5 |
| Product 1 percentage of product mix | 0.33 | 0.33 |
| Product 2 percentage of product mix | 0.67 | 0.67 |
| Traffic intensity | 0.75 | 0.75 |

Table 4.3: Values of system variables which must be specified before running simulation for both tests.

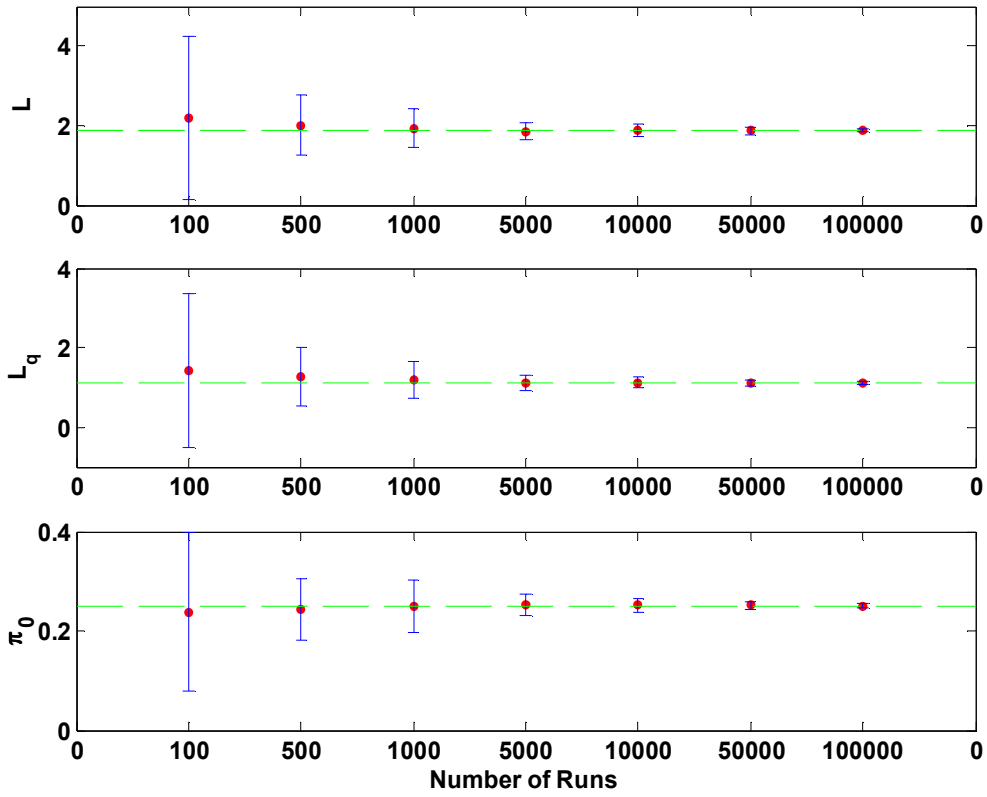


Figure 4.6: Comparison of mean time in system (L), in queue (Lq), and idle time for discrete event simulations and Pollaczek-Khinchin theory for a low volume system with one tool and two products that have equal processing times.

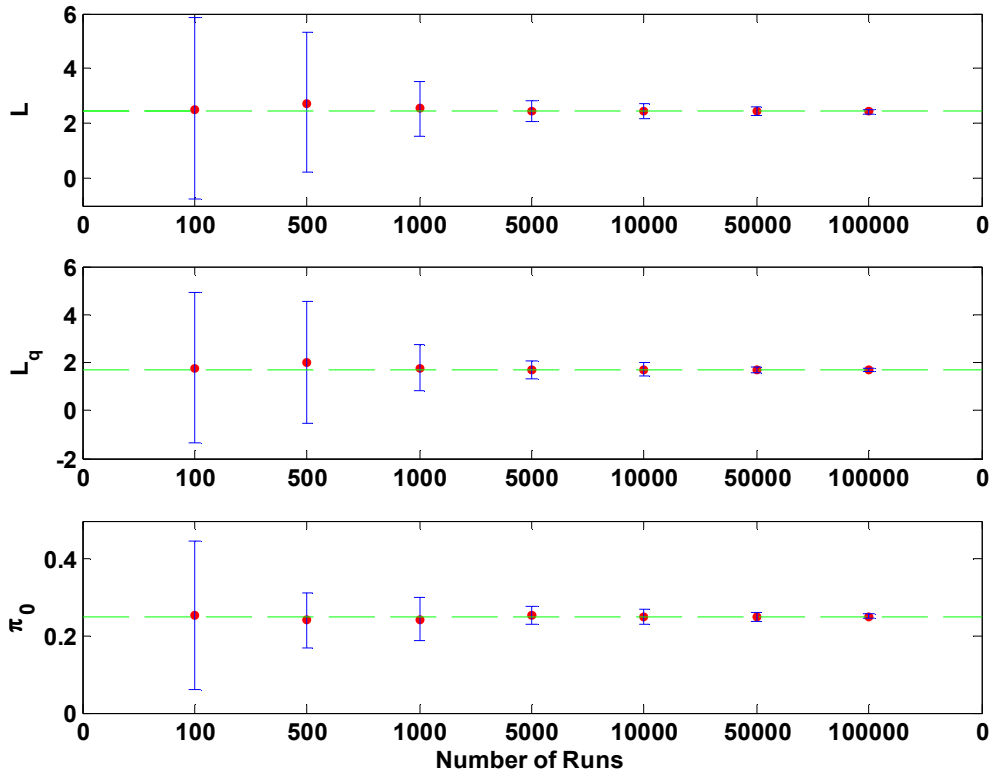


Figure 4.7: Comparison of mean time in system (L), in queue (Lq), and idle time for discrete event simulations and Pollaczek-Khinchin theory for a low volume system with one tool and two products that have different processing times.

### 4.4.4 Scheduling Algorithms for Comparison: FIFO and SPT

In order to evaluate the scheduling performance of the control-based objectives outlined in Section 4.2, they are compared with two traditional scheduling algorithms on the basis of mean product cycle time (both the mean taken over all products and the means of individual products). The two scheduling policies used for comparison are the first in, first out (FIFO) policy and the shortest processing time (SPT) policy. As its name indicates, the FIFO policy dictates that products are processed in the order in which they arrive to the tool. While FIFO is not necessarily optimal under common scheduling

related objective for the M/G/1 system on which the simulations are based, it is a simple rule to implement and is often used as a baseline case for comparative studies. On the other hand, under the SPT rule jobs with shorter expected processing times are processed first. SPT is known to minimize the average cycle time of all products in a multi-product M/G/1 type queue. It is important to note that SPT minimizes the average cycle time over *all* products but not necessarily the cycle time of each individual product. In fact, for a system like the one studied here, where processing times are deterministic, magnitudes of the mean cycle times for individual products correspond to the magnitudes of their processing times (i.e., smallest processing time equals smallest cycle time; largest processing time equals largest cycle time). This can lead to large disparities in cycle times for the individual products, whereas the cycle times tend to be more balanced in a FIFO schedule.

## 4.5 Simulation Testing of Multi-tool/Product System with Integration of Control and Scheduling Systems

### 4.5.1 Effect of Real-time Processing: Measurement Delay

The integration of the discrete event simulation with the process control simulation leads to a more complex system. In other studies of non-threaded control systems, it is assumed that outputs are processed in series with no overlap, but an actual system run in this fashion would require large idle times on the individual tool in order to allow each output to be completed and measured before the start of the next one. The resulting Gantt chart for a system run in this manner looks like the one in Figure 4.8, plot (a). In a true fab setting, however, multiple tools run in parallel, and the processing times of many outputs end up overlapping; therefore, it is unlikely the current process run is able to wait for measurements from previously started runs on other tools before starting.

Using the same processing order and tool selection as in Figure 4.8, plot (a), a Gantt chart for the realistic, parallel processing scenario is shown in plot (b).

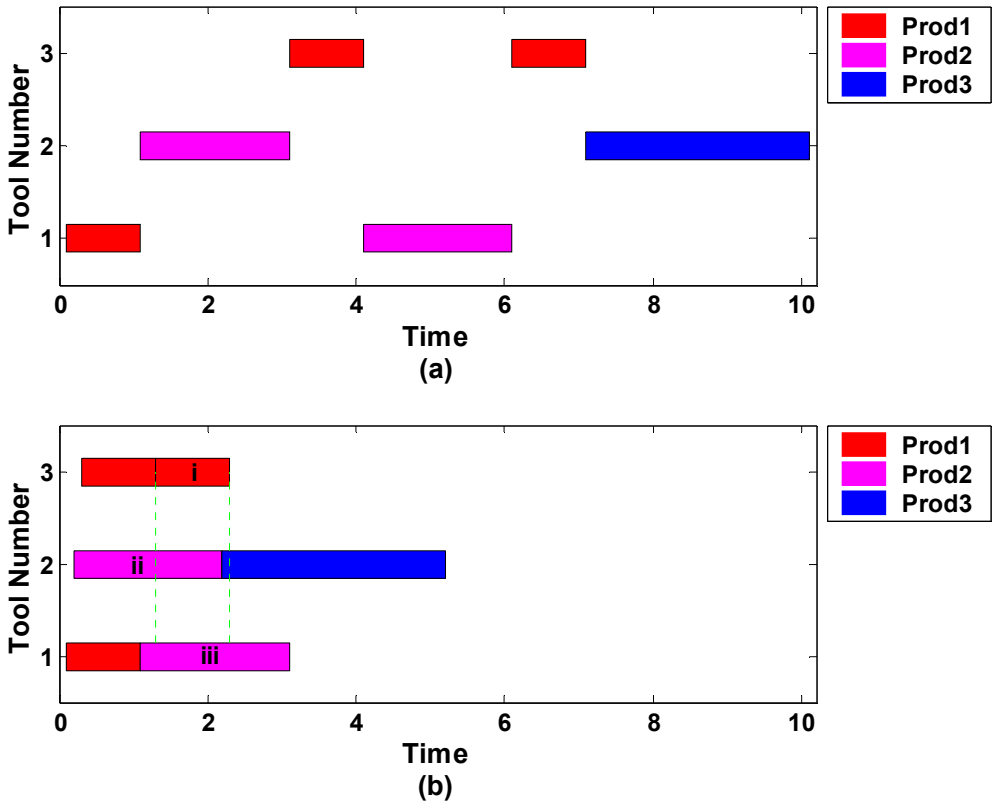


Figure 4.8: Sample Gantt charts for a three product, three tool system in which two product runs are performed on each tool. (a) Idealized (from control perspective) schedule with large idle times. (b) Practical (from processing/scheduling perspective) schedule with no forced idle times.

The discrete event simulation runs as shown in Figure 4.8, plot (b), accurately portraying the processing of products on multiple tools running in parallel, but the process control simulation must be adjusted in order to capture this behavior. In this work it is assumed that the scheduler only has knowledge of the products in queue and has no advance knowledge about future arrivals. A scheduling decision is made when a

tool becomes available and is based solely upon information related to the products currently in queue and the currently available tool. Once the product is chosen, the schedule is complete only up to the current product being started but not beyond (i.e., no scheduling of future starts is done). Consequently, the control system has knowledge of the order of starts, but not necessarily the order of completions, and should order the estimation updates accordingly.

As stated previously, problems with the estimation system arise in the parallel processing environment because products can begin (and sometimes finish) processing on one tool before the completion of a previously started product on another tool. The dotted, vertical lines in plot (b) of Figure 4.8 are used to highlight occurrences of delayed and out of sequence measurements. The second run on Tool 3 (labeled **i**) starts during the first run on Tool 2 (labeled **ii**) and completes after it. Because the measurement of **ii** is unavailable at the start of **i**, the measurement of **ii** is considered to be delayed. By contrast, **i** starts during the second run on Tool 1 (labeled **iii**), but completes before it. The measurement of **iii** occurs after the measurement of **i** and the measurements are considered out of order; this problem is commonly known as the "out of sequence measurement" (OOSM) problem [125].

For the systems studied in this work, it is assumed that measurements occur immediately at the completion of processing and before the start of the next product on a tool. To look at it another way, one can assume that the measurement time is built into the processing time for the product. Therefore, delays are only created by the differences in processing times for the various products. To calculate the maximum possible delay for a measurement, we need to calculate the maximum number of starts which are possible on other tools during the time span between a product's start and completion (i.e., its processing time). Theoretically, the maximum delay would occur when a large

number of starts of the product with the smallest processing time take place during a run of the product with the largest processing time. Using this fact, the following calculation gives the maximum delay (in steps) that a product could experience in the systems under consideration:

$$d_{Max} = \left\lceil \frac{Ts_{Max}}{Ts_{Min}} \right\rceil (t-1) \tag{4.14}$$

where $d_{Max}$ is the maximum number of runs that a measurement can be delayed, $Ts_{Max}$ and $Ts_{Min}$ are the maximum and minimum product service times, respectively, and $t$ is the total number of tools in the system. It is clear from Equation (4.14) that maximum delay grows as the ratio between the maximum and minimum processing times increases, and also, as the total number of tools in the system increases. This is an important factor since the possible size of delays tends to increase as the number of tools in the system increases.

The most direct solution to the measurement delay problem is to save the state estimate and covariance at the time of the last, in-order measurement. When a measurement is delayed, the associated filter update is skipped and succeeding updates proceed as they would normally so that the controller has a working estimate. When the delayed measurement arrives, the saved values and the processing order are used to recalculate the filter up to the time of the most recent, in-order measurement. These new state and covariance estimates are then saved and the working estimate is calculated up to the present time. This method, called "recalculating the filter through the delay period" [126], is described in greater detail by Pasadyn [41] as a way of implementing the Kalman filter in the fab, but it is not incorporated into later simulation testing by the author.

### 4.5.2 Simulation Testing of Combined Control/Scheduling System

#### *4.5.2.1 Effect of Changing Qualification Percentage in Systems with Delays*

In succeeding simulations in this work, the estimation updates occur in accordance with the start and completion times as they are defined by the discrete event simulator, and recalculation of the filter is done to compensate for delayed and out of sequence measurements. The idealized system is also run for comparison. To gage the effects of delayed measurements on system performance, the simulation tests of Section 3.6 are repeated using the new integrated system. To review, the simulations in Section 3.6 consist of 6 different system setups (labeled Tests), four different disturbance signal types (Signals), and 11 different qual run percentages. As in Figures 3.1-3, the plots of the new simulations are scaled by the corresponding output and estimate error values of the Test1 (base) case with white noise disturbances (Sig1), no quals, and no measurement delays; this case is labeled as (1,1,0,0) with the new fourth index now indicating no delays (0) or delays (1) in the system.

When the simulations include measurement delays, the plots for the test systems with three tools look quite similar to those of their ideal counterparts but with higher levels of error (see Figures 4.9–14, which show side by side results for Tests1–6, respectively). In terms of estimate error, the delayed measurements make little difference except at the one, two, and three percent qual levels where the error is noticeably higher for the delayed measurement case and a downward trend as the qual percentage increases is more apparent. Also of note is that while the output errors increase in the delayed measurement systems for all disturbance signal types, the signals that include non-stationary errors (Sig2, Sig3, and Sig4) experience greater increases than the purely white noise case (Sig1), indicating that measurement delays pose more of a problem in systems with disturbances more indicative of those found in practice.

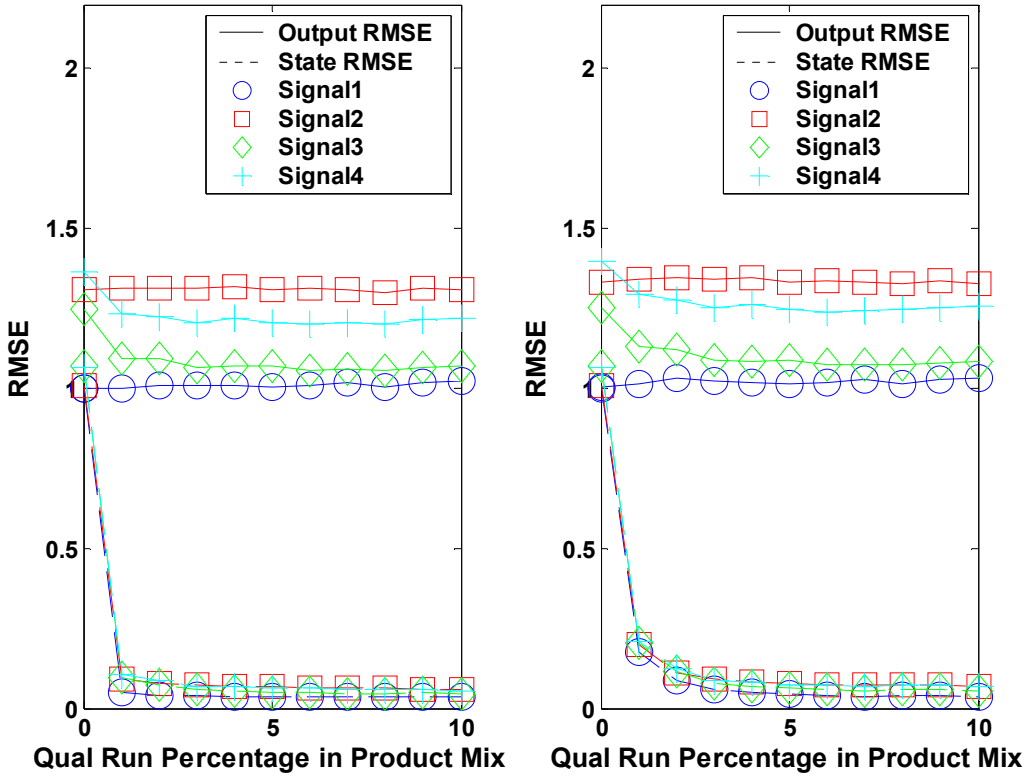


Figure 4.9: Comparison of output and state estimate errors for the Test1 system without delays (left) and with delays (right).

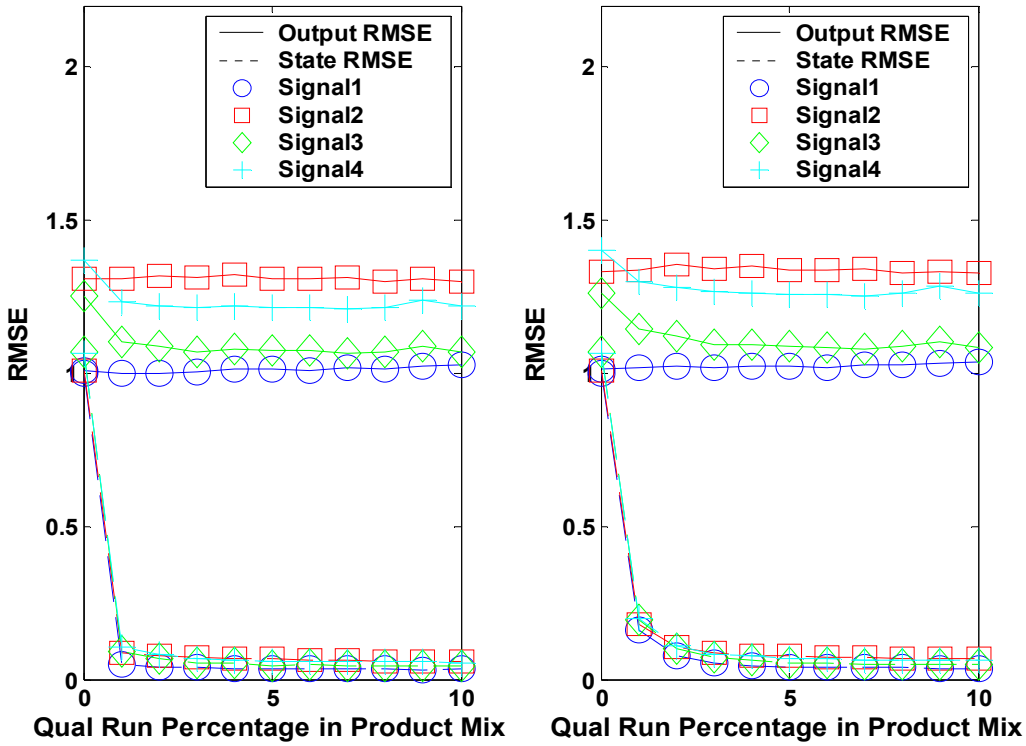


Figure 4.10: Comparison of output and state estimate errors for the Test2 system without delays (left) and with delays (right).

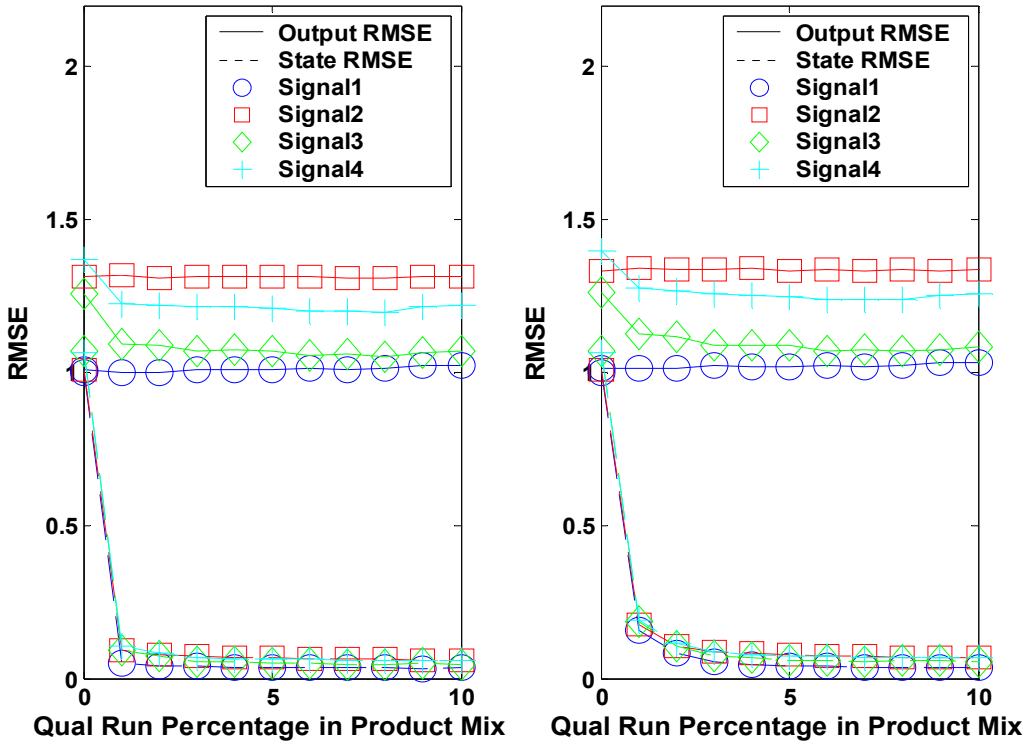


Figure 4.11: Comparison of output and state estimate errors for the Test3 system without delays (left) and with delays (right).

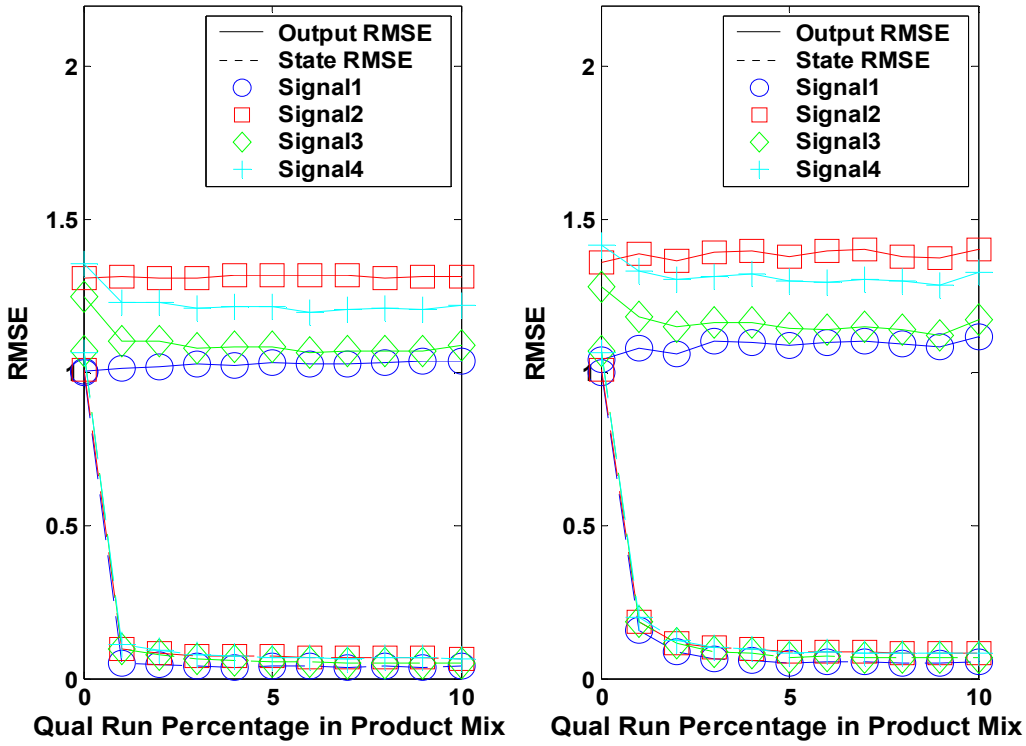


Figure 4.12: Comparison of output and state estimate errors for the Test4 system without delays (left) and with delays (right).

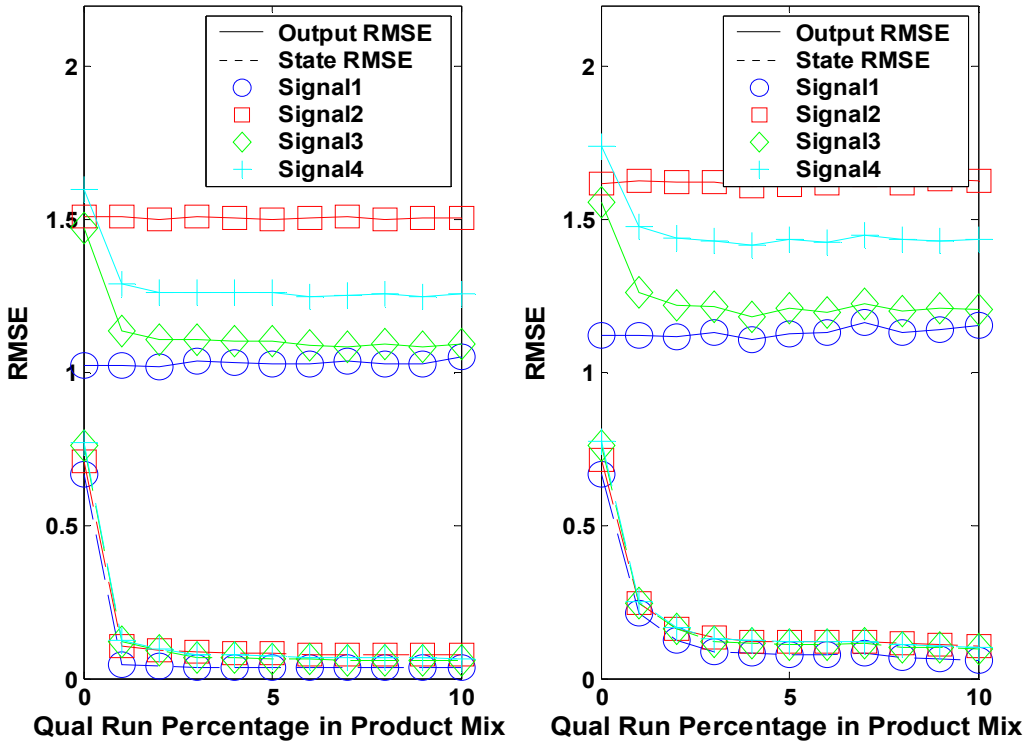


Figure 4.13: Comparison of output and state estimate errors for the Test5 system without delays (left) and with delays (right).

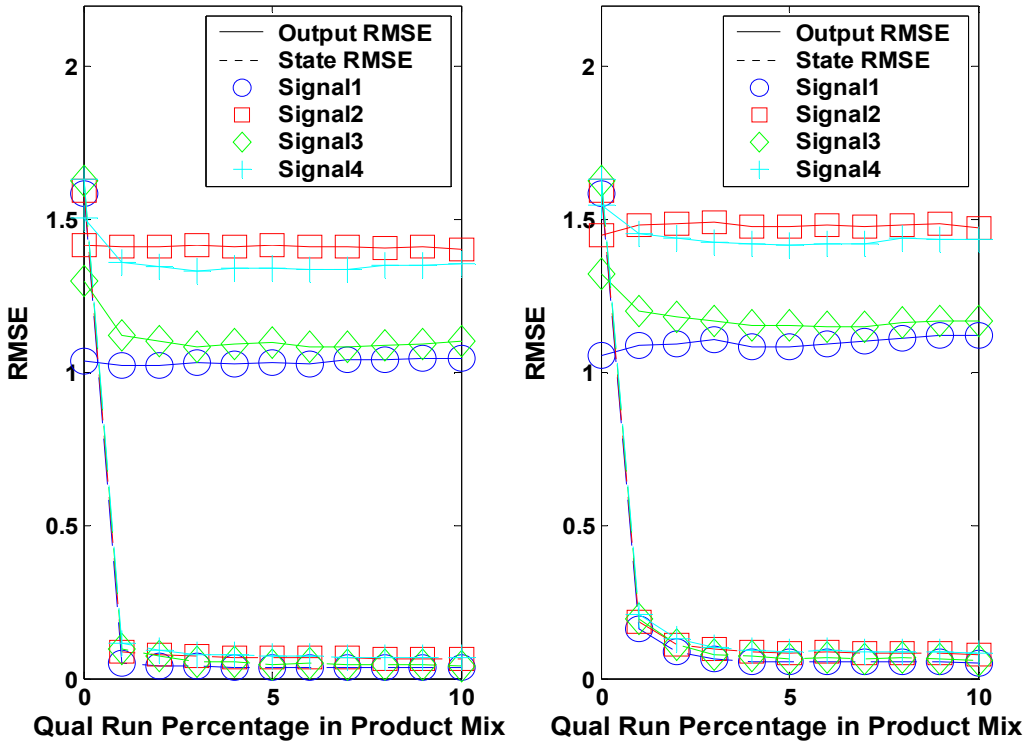


Figure 4.14: Comparison of output and state estimate errors for the Test6 system without delays (left) and with delays (right).

Table 4.4 shows the percentage change in output and state estimate errors for all of the three tool systems after the delayed measurement behavior is added to the simulations. The calculation of the data in Table 4.4 does not include the zero qual cases (as was previously explained, without quals, the estimator is unable to overcome the offset in the product state estimates created by their 50% initial error from the true states). The large magnitudes of the errors in the zero qual case tend to mask the changes in the subsequent cases when the errors are summed. It is clear from Table 4.4 that measurement delays have an adverse effect on system performance as errors in both outputs and state estimates increase.

| Test | RMSEy | RMSE$x_e$ |
|---|---|---|
| **1** | 2% | 42% |
| **2** | 2% | 35% |
| **3** | 2% | 37% |
| **4** | 7% | 46% |
| **5** | 10% | 84% |
| **6** | 6% | 47% |

Table 4.4: Percent change in root mean square error of outputs, y, and state estimates, $x_e$, for the six test systems after measurement delays are added to the simulations. Errors are summed over all disturbance types (Sig1-4) for each test, and all qual percentage levels except for the zero qual cases.

It is obvious from Table 4.4 that while the addition of measurement delays to the simulations causes an increase in error for outputs and states for all test cases, the effect is much greater in the case of Test5 which has twice as many tools (six) as the other cases. Despite the large increase in error, the plot of estimate errors for Test5 (see Figure 4.13) is similar in shape to the others. There is, however, a difference in the estimate error results; in the case of Test5, the percentage of qualification runs has a stronger effect on the reduction in error as quals are added. Figure 4.13 shows that systems with larger numbers of tools benefit more from an increase in the number of qual runs, at least in the range below 5%.

Looking at the size of delays in the various systems, it is apparent they are related to the settings for the system parameters. For Tests1–4, the systems have only three tools and three products and the ratio of largest processing time to smallest is four (three for the first test with no quals), so according to Equation (4.8), the maximum possible delay is eight (six). For Test5 the max-to-min processing time ratio is still 4:1 (3:1 for zero qual case), but with six tools, the maximum possible delay is now 20 (15). In the case of Test6, there are three tools but the processing time ratio increases to 7:1 (6:1), and the

maximum possible delay is 14 (12). In this case, adding the extra tools to the system creates larger delays than adding products with processing times that increase in increments of one. To check the results from Equation (4.14), delay statistics are gathered from the simulations. In Table 4.5 the average maximum delay for each qual percentage and test system is given. Note that the values in Table 4.5 are much smaller than those predicted by Equation 4.8 because the predicted values represent an upper bound. A very specific (but rare) set of circumstances must occur in the real system for the maximum delay to reach the upper bound.

From the data in Table 4.5, it is clear that Test5 has much larger maximum delays than the other systems; this is because the addition of extra tools makes it more probable that during the processing of the product at hand, additional product starts will occur on the other tools. For all systems, the maximum delay trends upward as the number of quals in the system increases due to the fact that the qual runs, with the exception of Test2, have the largest processing times. Therefore, as the occurrence of qual runs increases, so does the likelihood of longer delays.

In the case of Test2, qual runs have the shortest processing time ($PT_Q$=1) in the system. Therefore, the insertion of a qual run on one tool increases the likelihood of longer delays on the products running simultaneously on the other tools. Conversely, because the quals represent a small fraction of the total runs, Test2 has many fewer runs with a processing time of one when compared to the other three tool/three product systems, and so for a given qual percentage, it has a lower average maximum delay. Finally, Test4, owing to its unbalanced product mix, has greater average maximum delays than the same sized systems of Tests1–3. Test4 has a larger proportion (60%) of products in the system with a processing time of one (the minimum processing time in the system). Again this leads to a higher probability of an increase in the number starts

that occur during the processing of products that have longer service times. In general, it appears that for a given system, adding more runs at the extremes of the processing time distribution leads to the occurrence of larger maximum delays.

| Max Delay / Qual % | Test1 | Test2 | Test3 | Test4 | Test5 | Test6 |
|---|---|---|---|---|---|---|
| **0** | 5.29 | 4.00 | 5.11 | 5.87 | 11.14 | 6.31 |
| **1** | 5.46 | 4.13 | 5.28 | 6.48 | 11.81 | 6.44 |
| **2** | 5.61 | 4.28 | 5.34 | 6.87 | 12.13 | 6.47 |
| **3** | 5.70 | 4.31 | 5.40 | 6.93 | 12.58 | 6.49 |
| **4** | 5.78 | 4.49 | 5.53 | 7.15 | 12.67 | 6.56 |
| **5** | 5.90 | 4.54 | 5.62 | 7.28 | 12.80 | 6.65 |
| **6** | 6.10 | 4.59 | 5.82 | 7.36 | 13.12 | 6.62 |
| **7** | 6.02 | 4.59 | 5.73 | 7.45 | 13.10 | 6.71 |
| **8** | 6.11 | 4.65 | 5.79 | 7.46 | 13.03 | 6.74 |
| **9** | 6.15 | 4.80 | 5.83 | 7.56 | 13.15 | 6.80 |
| **10** | 6.20 | 4.72 | 5.94 | 7.56 | 13.07 | 6.82 |

Table 4.5: Average maximum delays calculated from 50 replicate studies for each test system and qualification percentage. Delays are measured not in time units but by the number of product starts that occur on other tools between the start and completion of the product under consideration.

Similar to Table 4.5, the mean delays over all runs for each test (in Table 4.6) show similar behavior. Test5 has noticeably longer delays than the other systems, and with the exception of Test2, the values increase as the number of qual runs in increased. In the case of Test2, the addition of quals with their brief processing time of one leads to extra runs with fewer delays, and apparently these more than counteract the additional delays created in longer running products. Also, note that Test3 has the shortest average delays due to its low volume schedule. When the system is run at a lower capacity, it allows for more tool idle time and therefore, fewer overlapping product runs.

| Mean Delay / Qual % | Test1 | Test2 | Test3 | Test4 | Test5 | Test6 |
|---|---|---|---|---|---|---|
| **0** | 1.92 | 1.92 | 1.68 | 1.92 | 4.75 | 1.92 |
| **1** | 1.93 | 1.92 | 1.69 | 1.93 | 4.77 | 1.92 |
| **2** | 1.93 | 1.91 | 1.70 | 1.94 | 4.79 | 1.93 |
| **3** | 1.94 | 1.90 | 1.71 | 1.95 | 4.81 | 1.94 |
| **4** | 1.95 | 1.90 | 1.73 | 1.96 | 4.83 | 1.95 |
| **5** | 1.95 | 1.89 | 1.73 | 1.97 | 4.85 | 1.95 |
| **6** | 1.96 | 1.89 | 1.74 | 1.97 | 4.87 | 1.96 |
| **7** | 1.96 | 1.88 | 1.75 | 1.98 | 4.88 | 1.96 |
| **8** | 1.97 | 1.87 | 1.76 | 1.98 | 4.89 | 1.97 |
| **9** | 1.97 | 1.87 | 1.77 | 1.98 | 4.90 | 1.97 |
| **10** | 1.97 | 1.86 | 1.78 | 1.98 | 4.91 | 1.97 |

Table 4.6: Average delays for all runs in each test system by qualification percentage. Delays are measured not in time units but by the number of product starts that occur on other tools between the start and completion of the product under consideration.

Comparing Tables 4.5 and 4.6 shows that adding qualification runs has a weaker effect on the mean delay than the maximum delay. From the zero qual case to the 10% qual case, the average maximum delay changes anywhere from 8–29% depending on the test system observed, but changes in the mean delay over the same course of experimentation range only from 3–6%. In the case of the mean delay, the qual runs are a relatively minor portion of the total runs and when the delay is averaged over all runs and replicates, the longer delays created by the quals have a small effect. Conversely, in the case of the maximum delay, having more quals increases the likelihood of a single longer delay occurring, and within each replicate, a long delay only has to occur once to be included in the final average (based on 50 samples).

In Section 3.6 and the current section, different settings for system parameters are explored and analyzed The control system performs quite similarly for a variety of

disturbances, but is more strongly affected by increases in the number of tools in the system and by measurement delays. In general, adding more long running or more short running products leads to an increase in the maximum possible delays that can be experienced but there is little effect on the average delay over all runs. Also, an increase in the number of tools leads to larger delays in the system and thus degraded control results. With these observations in mind, the control-friendly scheduling algorithms discussed in Section 4.2 are next integrated into the simulations so that their usefulness can be judged.

#### *4.5.2.2 Filter Tuning in Systems with Delays*

When settings for the simulation system are given in Section 3.6, the state noise covariance matrix, $Q$, is set to $Q_{plant}/5000$. This smaller value favors noise rejection over state tracking and leads to better control than is initially seen with $Q = Q_{plant}$. This is particularly true when estimation delays due to parallel processing are included; for the case with delays, unstable results such as the one shown in Figure 4.15 can occur when $Q$ is too large. The poor control is made worse as the delays increase such as in Test5; Figure 4.16 shows the results of running the same system as in Figure 4.13 but with $Q = Q_{plant}$ instead of $Q = \frac{Q_{plant}}{5000}$. Unlike in Figure 4.13, Figure 4.16 shows a large difference in performance for the delayed system when compared to the non-delay system and the effects of unstable results are seen in the spikes in output error for Sig3 and Sig4 at the two and four percent qual levels. Interestingly, the effects of the delays are mitigated by an increase in quals but overall, the control is not as good as the system with smaller $Q$.

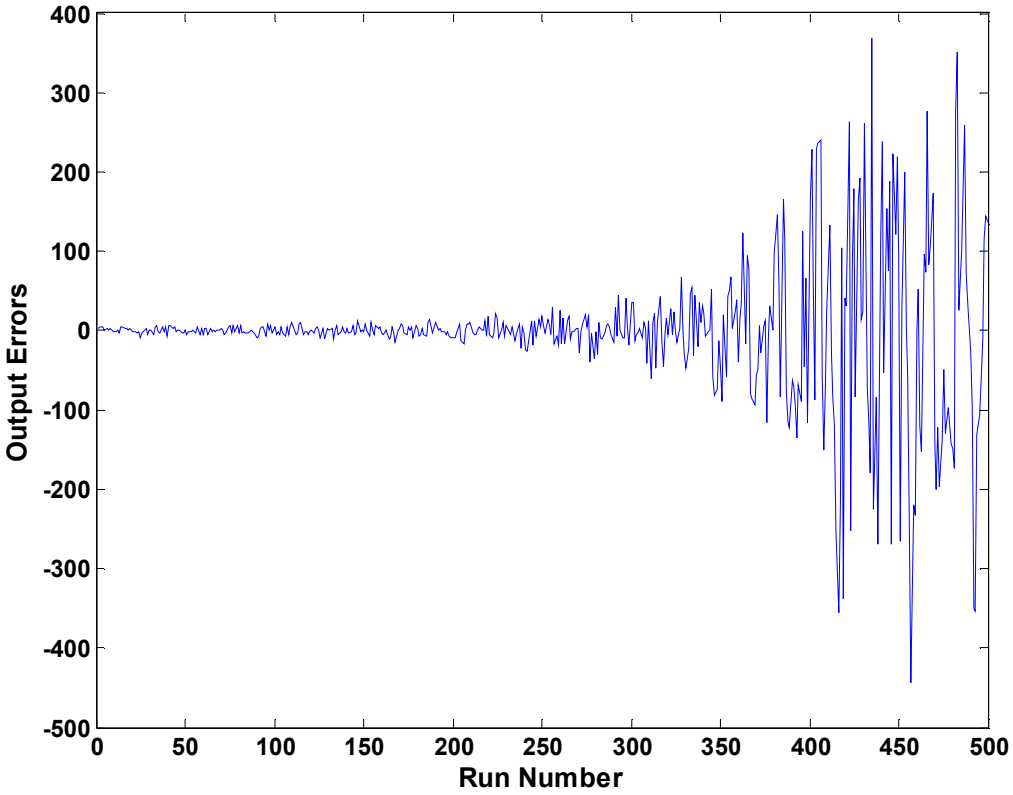


Figure 4.15: Plot of outputs for unstable system: Test 5, zero quals, measurement delays, scheduling performed using Method 3 (trace(P)).

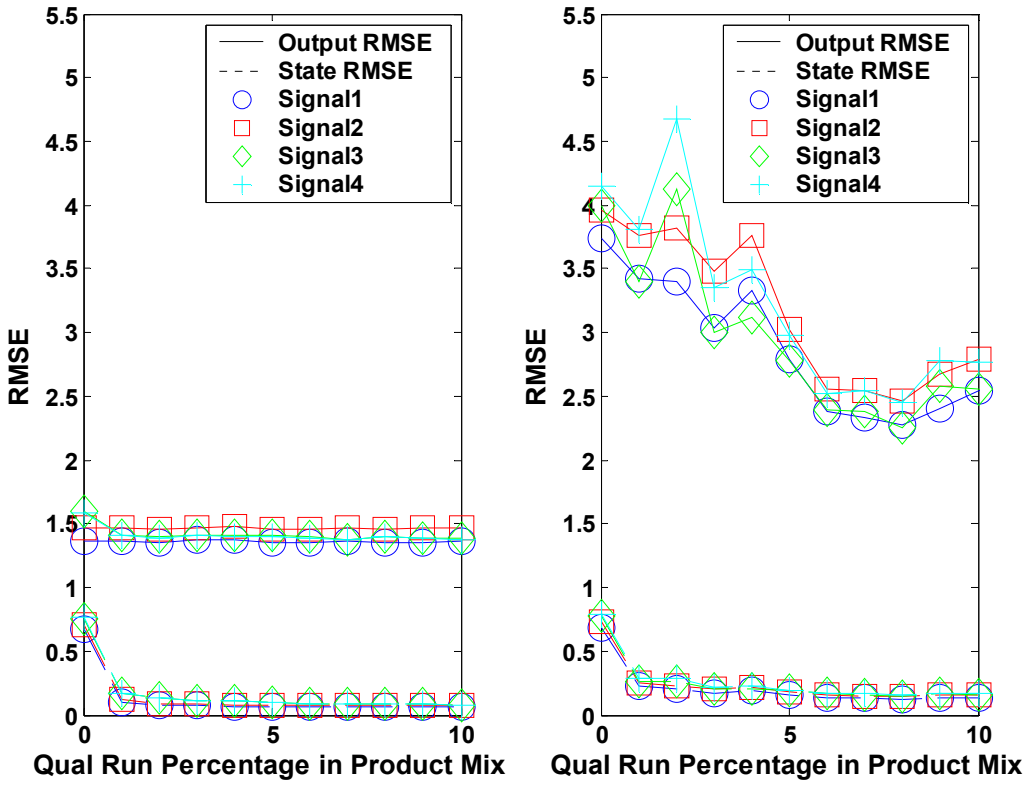


Figure 4.16: Comparison of output and state estimate errors for the Test5 system without delays (left) and with delays (right). Here $Q = Q_{plant}$.

## 4.6 Simulation Testing of Combined System with Control-Friendly Scheduling Algorithms

### 4.6.1 Simulation Setup

A series of simulations are run to test the performance of the various scheduling approaches discussed earlier in this chapter. This analysis includes both traditional scheduling policies and those explained in Section 4.2, which incorporate elements of the control model in order to enhance control system performance. Table 4.7 gives a list of the tested methods along with a brief description of each and an assigned reference number.

The first two items in Table 4.7 are common scheduling policies and are used for comparison to the control-based methods. The next method is the approach suggested by Pasadyn in which the trace of $P$ is minimized (Method 3), but there also several other methods based on the estimate error covariance matrix which are tested. Methods 4 and 5 are based on the theoretical output covariance matrix which is a transform of the estimate error covariance matrix using the state transition matrix, $C$. By using the objective of Method 4, a context is selected which minimizes the total output error variance of the system, whereas in Method 5, only the variance of the selected context is minimized. The final approach that uses the error covariance matrix, Method 6, is designed to choose the context which minimizes the condition number of $P$ at each step, this approach directly aids the stability and accuracy of the Kalman filter.

The last six scheduling algorithms listed in Table 4.7 are based upon the partial observability measures discussed in Section 4.2.2. All of these approaches make use of the information matrix, $L_{0,k}$. Method 7 seeks to maximize the smallest singular value of $L_{0,k}$ thereby increasing the degree of observability of the system. Method 8 uses a similar approach but instead chooses the context (i.e., measurement) which is closest to the subspace associated with smallest singular value of $L_{0,k}$. The final four methods are the same as Methods 7 and 8 but use a moving window approach to the recursive calculation of $L_{0,k}$ rather than the full process history. While Methods 9 and 11 use a window length, $w$, of $n$ (the number of states in the system), Methods 10 and 12 use window length of $2n$.

| Number | Name | Description | Objective to be optimized |
|---|---|---|---|
| 1 | FIFO | First in, first out | none |
| 2 | SPT | Shortest processing time | Minimize average cycle time |
| 3 | trace($P$) | Trace of estimate error covariance matrix | Minimize total estimate error variance |
| 4 | trace($CPC^T$) | Trace of output error covariance matrix | Minimize total output error variance |
| 5 | trace($C_rPC_r^T$) | Trace of context error covariance matrix | Minimize variance of selected output |
| 6 | cond($P$) | Condition number of $P$ | Minimize condition number of $P$ |
| 7 | Deg. Obs. | Smallest singular value of $L_0$ | Maximize degree of system observability |
| 8 | Ang. | Angle between output & LOSS | Minimize angle between output & LOSS |
| 9 | MW(Deg. Obs.), $w = n$ | Deg. obs. with moving window of size $n$ | Maximize degree of system observability |
| 10 | MW(Deg. Obs.), $w = 2n$ | Deg. obs. with moving window of size $2n$ | Maximize degree of system observability |
| 11 | MW(Ang), $w = n$ | Ang. with moving window of size $n$ | Minimize angle between output & LOSS |
| 12 | MW(Ang), $w = 2n$ | Ang. with moving window of size $2n$ | Minimize angle between output & LOSS |

Table 4.7: Summary of various scheduling methods tested via simulation. LOSS = least observable subspace.

In the following simulation studies, the control and scheduling systems each measure performance using their own distinct metrics; the level of output error or estimation error in the case of the control system and the average cycle time per product or level of tool utilization (i.e., the percentage of time tools are making products rather than processing quals or sitting idle) in the case of the scheduling system. Because alterations to processing order, the placement of products on tools, or the use of qual runs result in changes to the performance of both systems, it is natural to treat the combined system as a bi-criteriate problem and generate Pareto plots for its analysis [57, 100].

For the simulations in this section, the same test systems introduced in Section 3.6 (see Table 3.3) are used, but only Signal 1 (white noise) and Signal 4 (non-stationary disturbances) are used (see Table 3.4) to compare the most basic disturbance case to the most complex. An additional test system is also introduced to give further insights into the performance of the various scheduling approaches. Test 7 is the same as the Test 1 except the noise levels for the tool states and product states are no longer uniform. Table 4.8 is similar to Table 3.4 and shows the difference between the new test and Test 1.

| Variable \ Test | 1 | 7 |
|---|---|---|
| **Num. of Tools** | 3 | 3 |
| **Num. of Prods** | 3 | 3 |
| **Prod1 Proc. Tm.** | 1 | 1 |
| **Prod2 Proc. Tm.** | 2 | 2 |
| **Prod3 Proc. Tm.** | 3 | 3 |
| **Prod4 Proc. Tm.** | - | - |
| **Prod5 Proc. Tm.** | - | - |
| **Prod6 Proc. Tm.** | - | - |
| **Qual Proc. Tm.** | 4 | 4 |
| **Product Mix** | All 1/p | All 1/p |
| **Tool Noise Std. Dev.** | All 0.1 | (0.25, 0.1, 0.05) |
| **Prod. Noise Std. Dev.** | All 0.1 | (0.25, 0.1, 0.05) |
| **Traffic Volume** | High | High |

Table 4.8: Summary of settings for new test systems compared to Test 1.

As before, each simulation is 500 process runs long and is repeated 20 times with different seed values for the random number generator. In each case, the first 50 runs are taken from the FIFO schedule with each scheduling policy taking over at the 51$^{st}$ run and dictating the processing order until the 500$^{th}$ run is reached. Statistics are then calculated using the final 450 runs so that the different methods can be compared.

For a given set of process conditions, a series of simulations is run with varying levels of quals and with the different schedules produced by the methods discussed previously. Each combination of test conditions, disturbance type, qual percentage, and schedule is run for multiple replicates with the results being averaged to give output RMSE, state estimate RMSE, product cycle times, and tool utilizations. The appropriate

combinations of control and scheduling metrics are then plotted against each other to produce the final Pareto plots. From these plots, differences in both scheduling and control performance for the algorithms can be viewed simultaneously and those with desired characteristics can be chosen. As with the comparisons between Sections 3.6 and 4.5.2, the ideal systems, in which measurement delays do not occur, are run in parallel with the systems that include delays in order to judge differences in performance results.

To help simplify the analysis and keep the plots from being overcrowded with symbols, only a few of the qual percentages are shown. From the earlier studies on qual percentage it is seen that there is little to no improvement in control performance after the five percent level of quals is passed. Therefore, no qual percentages above five are shown and initially, only every other percentage from one to five is shown along with zero percent.

### 4.6.2 Pareto Analysis of Output RMSE versus Product Cycle Time

The set of Pareto plots explored here compare the output RMSE and the average cycle time of products. It was pointed out earlier that it is desirable to minimize both of these metrics so the methods which appear closer to the bottom left corner of the plot are of the greatest interest. The particular methods in this region strike the best balance between the objectives of the two systems. Initial results when looking at all methods over all tests do not indicate one clear winner. Several general observations include:

- SPT does give the minimum cycle time over products while FIFO tends to do the best job of balancing the cycle time amongst the products.
- All of the control-based objectives give priority to quals except for Method 6.
- The moving window versions of partial observability objectives give inconsistent results for these test systems due to a lack of information. There are often ties when choosing among multiple products at a step; because the algorithm chooses

the product with the lowest product number in the event of a tie, the m.v. approaches are more weighted toward these products. More research should be done to find an optimal window length and these methods are dropped from the remainder of the analysis.

To try to narrow the field of potential scheduling algorithms, two specific cases which are common in practice are chosen to highlight the differences between the various scheduling approaches. The first test chosen is Test 4 which uses an unbalanced mixture of products as is common in fabs today. The second test chose is Test 7 in which the products and tools are differentiated from each other by their inherent levels state variance. The results of the analysis of these to tests follows.

#### *4.6.2.1 Unbalanced Mix of Products (Test 4)*

One key test for the combined system is the case of low running products. In most fabs, the mix of products is unbalanced with a few high running products comprising the majority of the production runs. In many cases these high running products are considered low priced commodities while the lower running products are specialty orders that have higher value [92, 93]. It is therefore important for the low running products to be controlled well and to move through the fab quickly. In simulations for Test 4, the mix is split amongst three products with products 1 and 2 comprising 90% of the mix so that Product 3 only appears in the system, on average, 10% of the time. Because the performance of the low running product is of high importance, Pareto plots of its cycle time and output error are shown in Figures 2 and 4 for qual percentages of zero, one, three, and five percent.

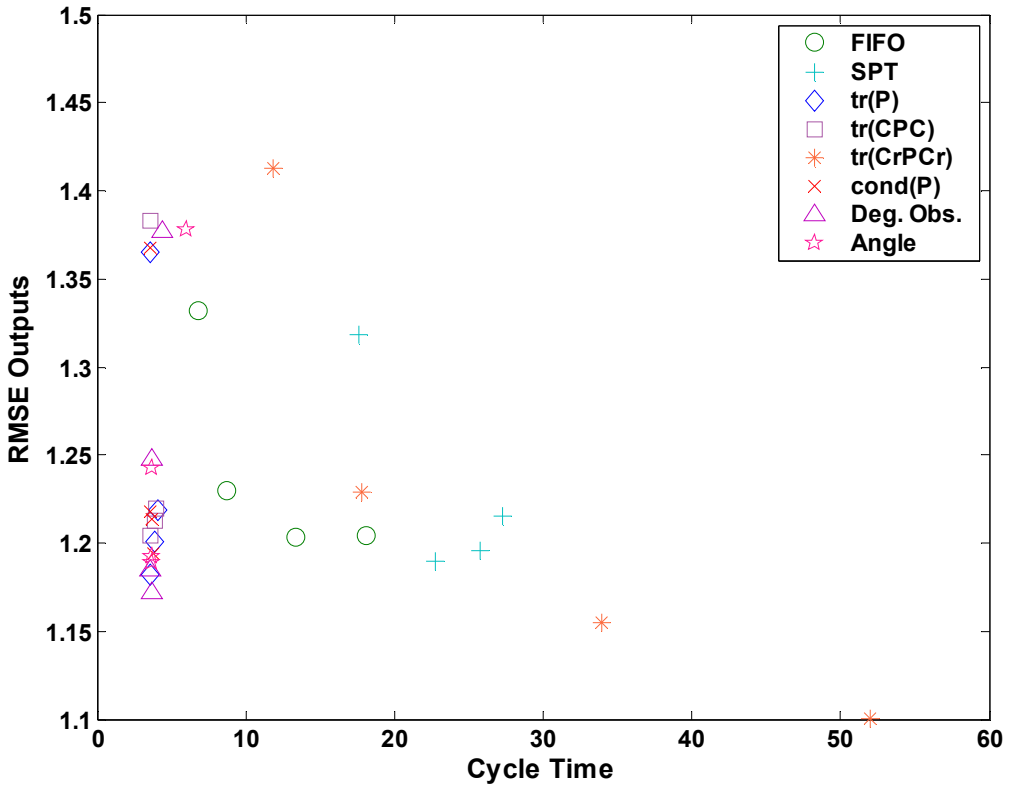


Figure 4.17: Pareto plot of output error v. cycle time for Product 3, the low running product in Test 4, for an ideal system with no Kalman update delays.

Figure 4.17 presents the Product 3 Pareto plot in the ideal case where all measurements are received in order and without delays. Here, the output error and cycle time for Product 3 are plotted for each schedule at four different qual percentages (zero, one, three, and five). Note the while SPT leads to minimum average cycle time over all products, it gives very high cycle times for Product 3 because it has the longest processing time of the products. The two schedules generated using the partial observability measures and all but one of those generated using the estimate error covariance matrix give very low cycle times for Product 3 across all qual levels. The

exception is the objective that uses $\text{trace}\left(C_r P C_r^T\right)$ which incurs a heavy cycle time penalty negating its excellent control performance.

The $\text{trace}\left(C_r P C_r^T\right)$ algorithm selects the thread from those available that has the minimum variance after measurement which can cause the scheduler to "stick" on certain threads for long stretches when they reach a state of low variance. In an extreme case, the low running products can all get pushed to the end of the schedule; such a case occurs in one of the replicate simulations and is shown in the form of a Gantt chart in Figure 4.18. From this figure, it is also easy to see the long runs of single products on tools created by the scheduling algorithm.

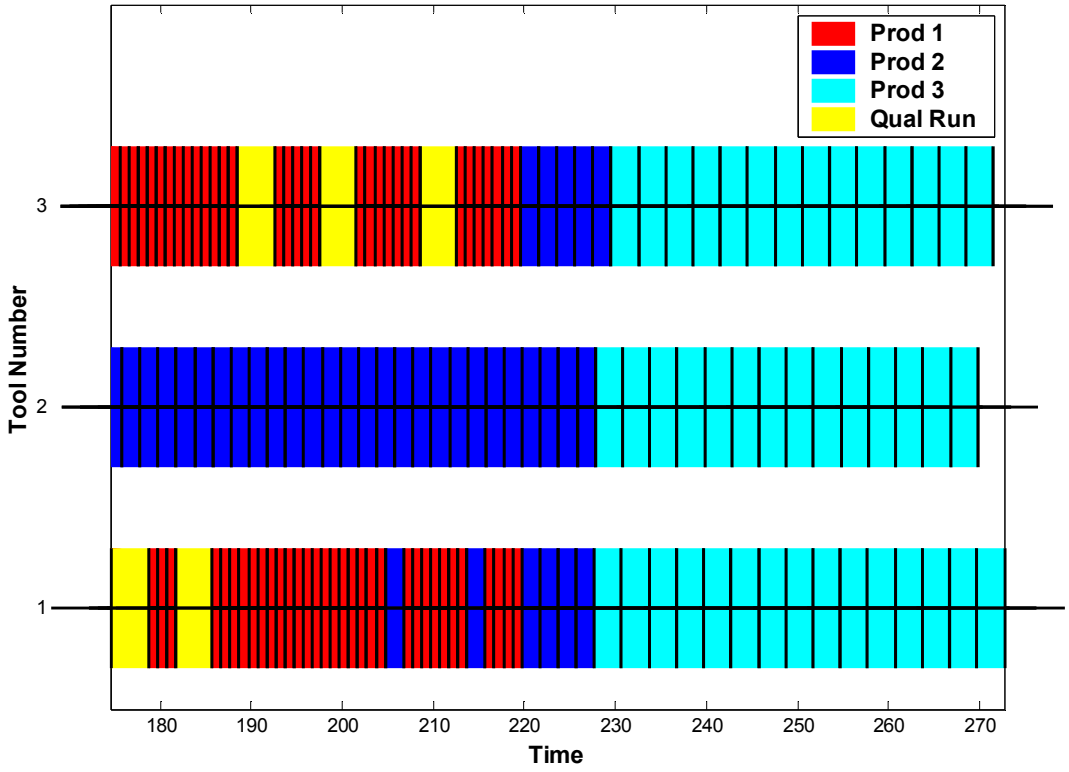


Figure 4.18: Gantt chart showing order of runs on the three tools using the scheduling Method 5, minimizing $\text{trace}\left(C_r P C_r^T\right)$. Settings: Test 4, Sig4, 5% quals. Only the last 250 runs are shown for ease of viewing.

The same system seen in Figure 4.17 is presented in Figure 4.19 for the case with delays. The simulations now include delays in the updates to Kalman filter estimates caused by the parallel processing environment, and the results are similar except the error levels are higher in general and are particularly so for the scheduling methods that had poor cycle time performance. There is an especially drastic change in the output error values for the $\text{trace}\left(C_r P C_r^T\right)$ method. This behavior points out the importance of using the discrete event simulator to more accurately model the behavior of the real parallel processing environment. It is clear that in this more realistic case, the methods based on

partial observability methods are superior choices with several of the *P*-based metrics also doing well in terms of cycle time, but slightly worse in terms of control performance.

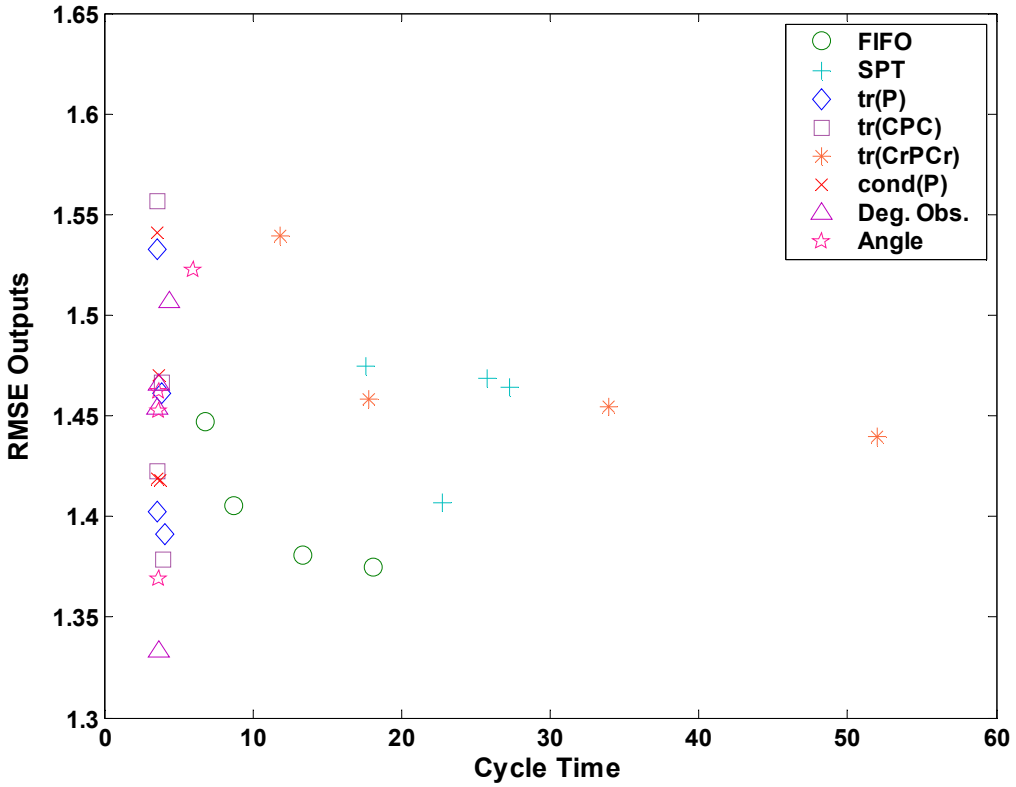


Figure 4.19: Pareto plot of output error v. cycle time for Product 3, the low running product in Test 4, for a system with Kalman update delays.

Looking at other products in the system, Product 2, which accounts for 30% of the product mix, has similar patterns of performance to Product 3 as seen in Figure 4.20. In the case of Product 2, the partial observability measures and those based on condition number of *P* and trace of *P* strike the best balance between cycle time and control performance. The only major change from the Product 3 case is that SPT is now competitive in terms of cycle time but is still inferior in terms of output error.

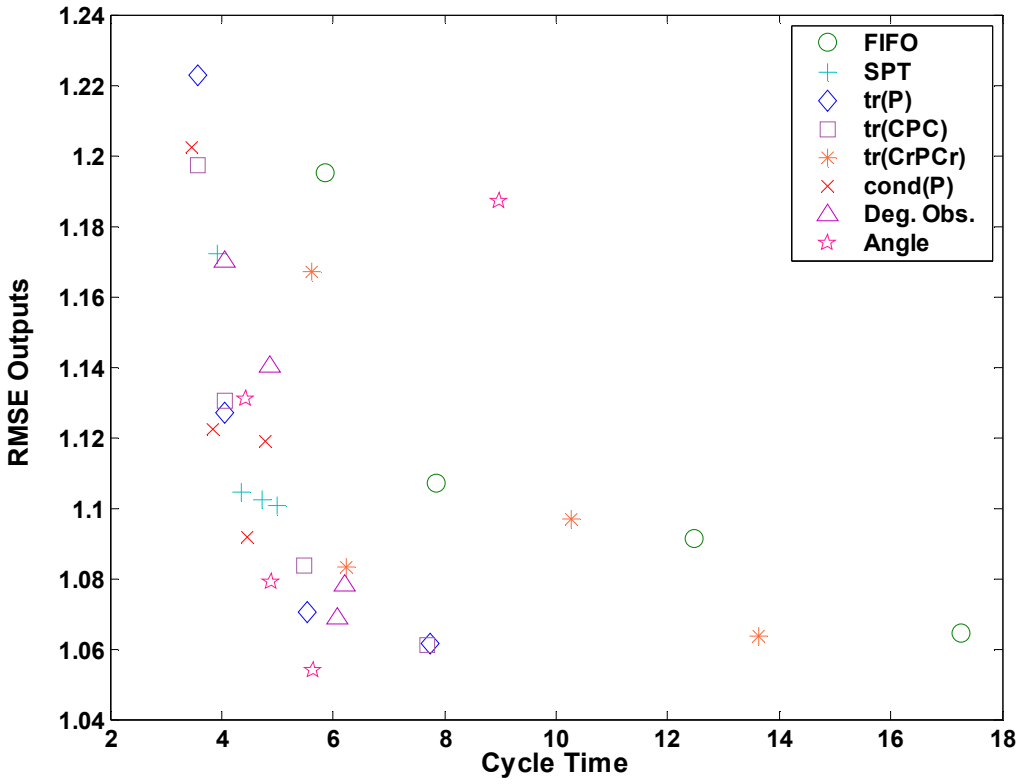


Figure 4.20: Pareto plot of output error v. cycle time for Product 2, the medium running product in Test 4, for a system with Kalman update delays.

The final Pareto plot (Figure 4.21) for Test 4 is for Product 1 which is the highest running product and represents 60% of the mix. For this product, all methods are fairly equal in terms of control performance at qual percentages beyond zero (the collection of symbols in the upper left corner of the plot represent results for zero quals simulations). As would be expected, the various scheduling methods have reversed themselves in terms of cycle time from the Product 3 case, but due to the dominance of Product 1 in the mix, this is not problematic. Even in the extreme case discussed previously, if a partial observability method such as min(Angle) is used, all runs of Product 1 cannot be completely pushed back to the end of the production period. In fact less than half of the

Product 1 runs are held off while the other half are distributed fairly evenly throughout schedule.

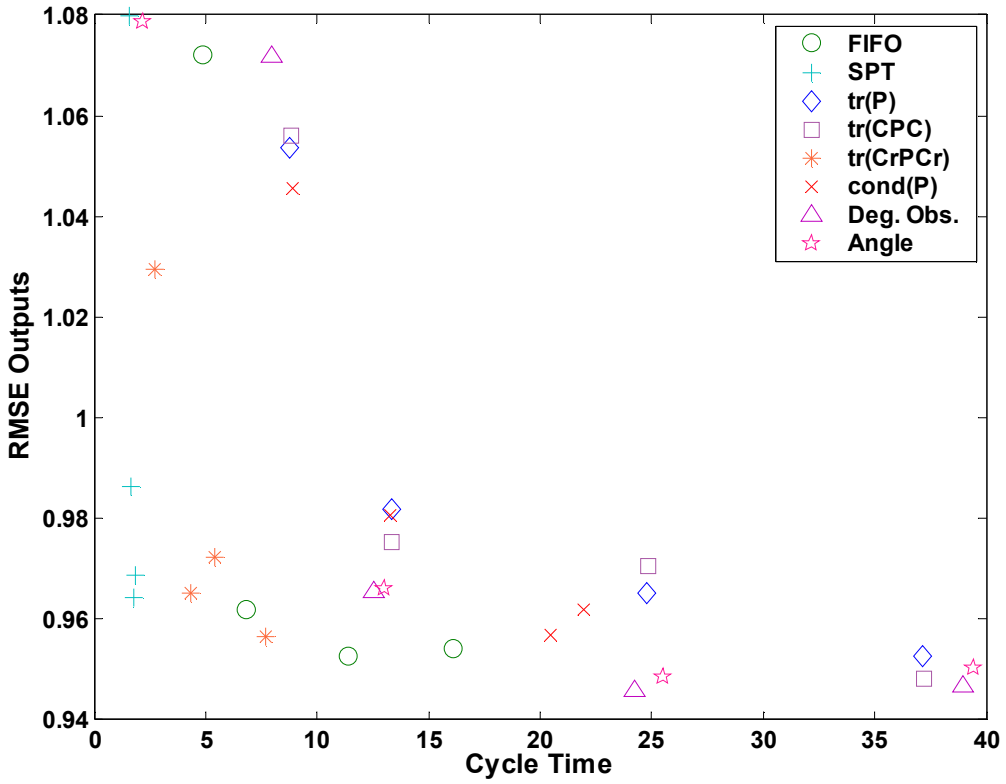


Figure 4.21: Pareto plot of output error v. cycle time for Product 1, the low running product in Test 4, for a system with Kalman update delays.

Finally, to address concerns that the results presented for the system with an unbalanced product mix are restricted only to this particular case, a variant of the experiment is performed in which the product processing times are reversed. Thus, the low running product is given the lowest processing time rather than the highest and vice versa for the high running product. While the values of the cycle times for each product change in accordance with the change in processing times, the relative performance of the scheduling methods does not; the P-based methods (except Method 5) and the partial

observability methods still yield superior performance by moving the low running products quickly through the system and with lower levels of error. The only real change is that SPT now has similar cycle time performance to the desirable P-based and partial observability methods, but still has poorer control performance.

Because the large majority of fabs now function in a fashion in which there are one or more low running products mixed in with a small number of high running products that dominate the product mix, it is an important test for any control or scheduling system to be able to handle the low running products effectively. The results of Test 4 demonstrate that SPT is too dependent on product processing times and therefore does not deliver consistent results. Additionally, FIFO is dependent upon the arrival sequence and therefore, is also limited in its effectiveness. Of the six control-based algorithms, all but Method 5 perform well, and one of the partial observability algorithms, Method 7, does a marginally better job of minimizing Product 3 error than the others.

#### *4.6.2.2 Differences in Performance within Tool and Product Groups (Test 7)*

An observed behavior of the fabrication process is that not all tools in a group perform alike and similarly for products. To mimic this behavior, Test 7 uses different noise parameters for the states; the three tools are given high, medium, and low levels of noise for generating their disturbance trajectories and the same is done for the products. Given the nine possible tool/product combinations that generate the outputs, the outputs can range from those with very low variability to those with very high variability. Ideally, a successful scheduling algorithm would sacrifice some outputs at the low end of the spectrum in order to reduce the number at the high end of the spectrum, and thus decreasing the likelihood of excursions from the process limits. Recall that products that fall outside of control limits lead to scrapping or reworking of wafers, both of which

reduce yield. Here it is assumed that the low variance product is easily controlled (i.e., measurements fall well within control limits) while the high variance product operates very close to its control limits.

Given the results of the previous test in the last section, and the fact that neither method performs particularly well for the current example, FIFO and SPT are eliminated from consideration. To make interpretation of the Pareto plots easier, FIFO and SPT are not shown. Despite the fact that the $\text{trace}\left(C_r P C_r^T\right)$ algorithm performed poorly in the last test, it is included because it shows an interesting behavior that contrasts with the behavior of the other *P*-based methods. Finally, since the reduction in output error when moving from three percent quals to five percent rarely justifies the associated increase in cycle time, the 5% samples are also eliminated.

In the particular test system used here, there is a uniform distribution of products and their processing times one, two, and three, respectively. The products and tools both have noise variances that proceed from high to low as seen in Table 4.9. Accordingly, it should be advantageous to process Product 1 more often on Tool 3 in order to reduce its level of error. To minimize the effects of this shift, Product 3 should be processed on Tool 1 more often. This is, in fact, the type of behavior demonstrated by all of the *P*-based metrics except for Method 5. Method 5 actually produces the opposite type of schedule with Tool 1 favoring Product 1 and Tool 3 favoring Product 3. Figure 4.22 shows the distribution of products across tools for the objective based on the condition number of *P* (Method 6). Note that the number of Product 1 runs is heavily skewed toward Tools 2 and 3, while Product 3 shows the opposite behavior. Additionally, the number of quals run on a tool is proportional to its variance (i.e., Tool 1 receives the most quals and Tool 3 the fewest).

| State | Variance |
|---|---|
| Product 1 | High |
| Product 2 | Medium |
| Product 3 | Low |
| Tool 1 | High |
| Tool 2 | Medium |
| Tool 3 | Low |

Table 4.9: Relative noise levels for various states in Test 7.

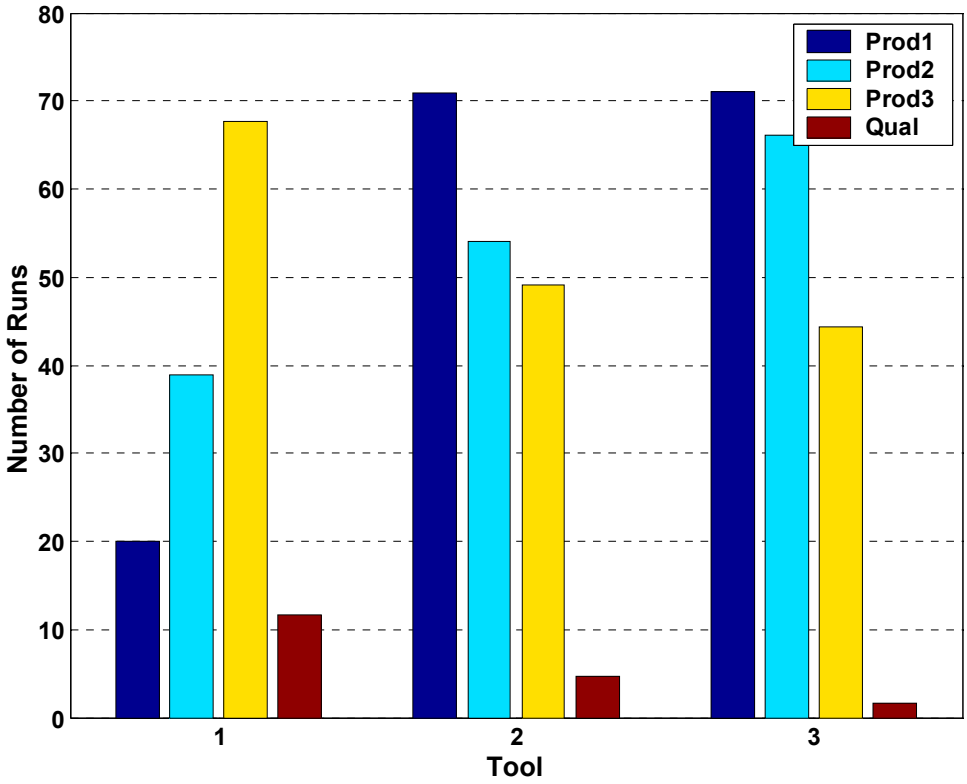


Figure 4.22: Distribution of product and qual runs across tools for Method 6, based on cond($P$), and three percent quals.

Turning to the other $P$-based algorithms, it is found that Method 4 (trace($CPC^T$)) performs similarly to Method 6 but that the distributions are not as strongly skewed. Method 4 results in about twice as many Product 1 runs on Tool 1 and a balanced number

of quals on Tools 1 and 2. The algorithm which uses trace($P$) (Method 3) has similar results to Method 4 except that it balances the quals across the tools; this difference is due to the fact that it does not consider any of the covariance terms in the $P$ matrix. Finally, Method 5, which seeks the single available output with minimum variance, prefers to run Product 3 on Tool 3, thus leaving the majority of Product 1 runs to Tool 1. The distribution for this method is shown in Figure 4.23.

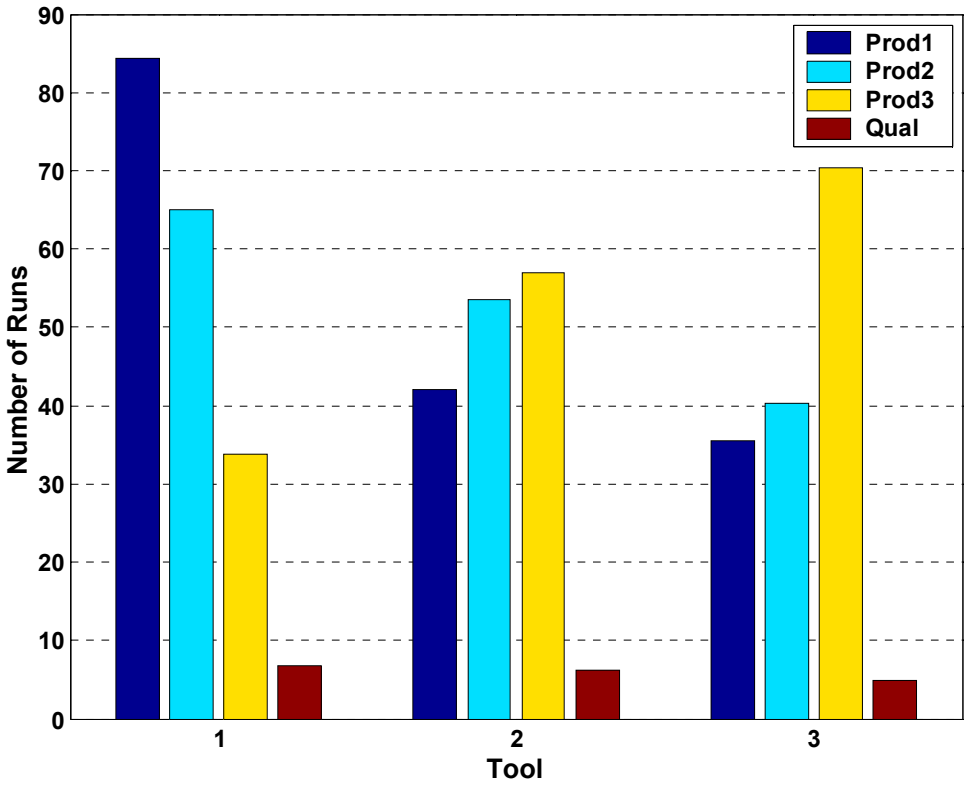


Figure 4.23: Distribution of product and qual runs across tools for Method 6, based on trace$\left(C_r P C_r^T\right)$ (Method 5), and three percent quals.

The remaining two methods are those based on partial observability measures which use the information matrix $L_{0,k}$. Because the calculation of $L_{0,k}$ only uses $C_r$ and the output noise covariance matrix, $R$, Methods 7 and 8 (unlike $P$-based methods) do not

have the additional information about differences between the states that is included in the state noise covariance matrix, $Q$. Thus, the plot for Method 7 (Figure 4.24) shows that it maintains a balanced distribution of products and quals across the tools. Method 8 performs similarly.

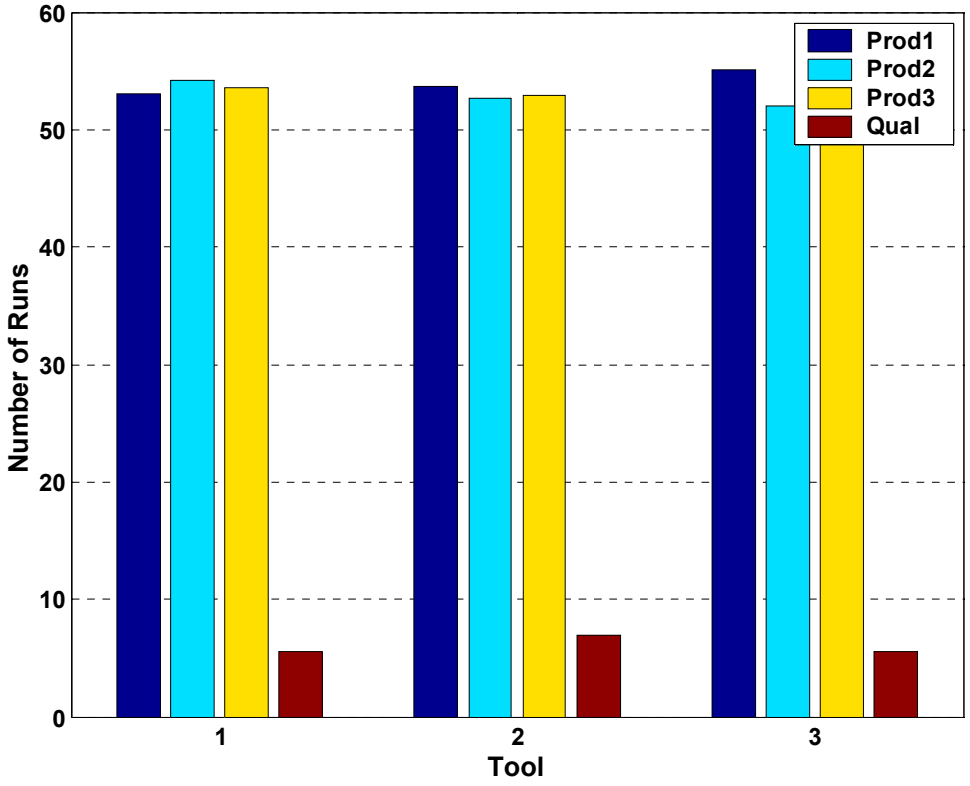


Figure 4.24: Distribution of product and qual runs across tools for Method 6, based on Method 7 (maximize degree of observability), and three percent quals.

To gage the performance of the schedules produced, the Pareto plots from the system are analyzed. Looking first at the Product 1 Pareto plot (Figure 4.25), it is immediately obvious how the various schedules rank. Methods 3, 4, and 6, which use $P$, are clearly dominant in terms on cycle time and Method 6, based on the condition number of $P$, is especially good in terms of output error when quals are used. On the other hand,

Method 5, which produces the opposite type of schedule, does poorly for both performance metrics, and the partial observability algorithms fall somewhere in between.

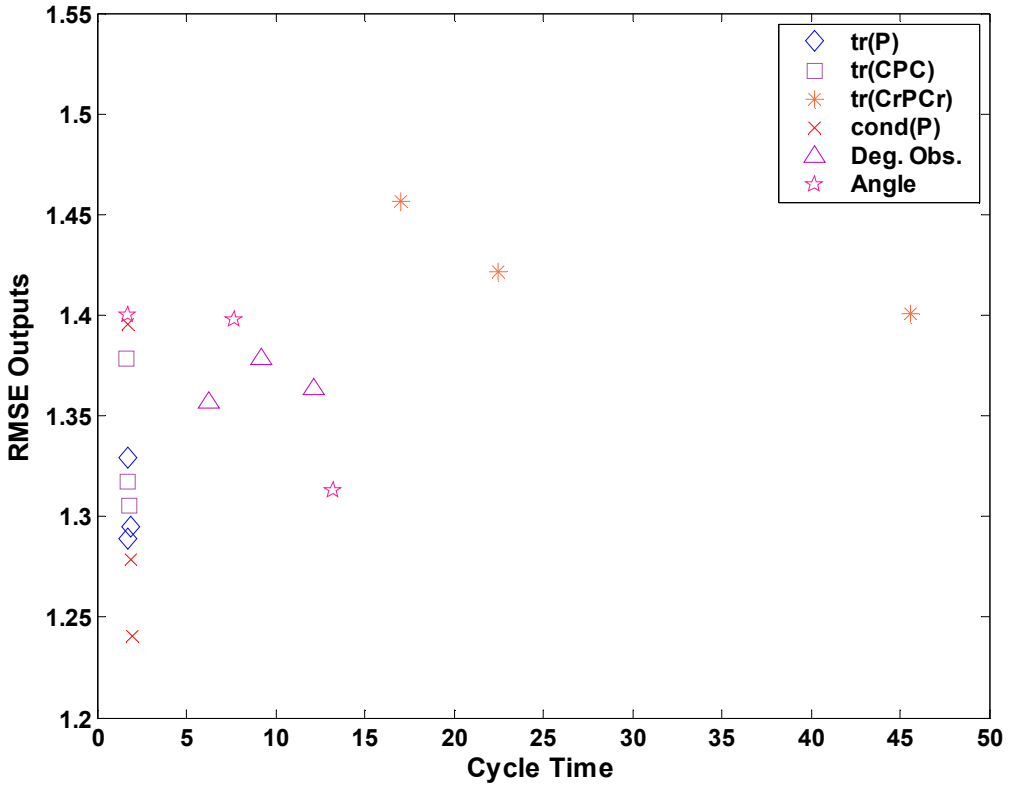


Figure 4.25: Pareto plot of output error v. cycle time for Product 1, the high variance product in Test 7, for a system with Kalman update delays.

The Pareto plot for Product 2 is similar to Figure 4.25 with lower output error and cycle time for Methods 3, 4, and 6 and Method 6 giving the greatest reduction in output error but also incurring the largest cycle times of the three. Again, the other three methods perform poorly by comparison, but Method 5 is now lumped in with Methods 7 and 8 instead of being far to the right and higher in the plot. Conversely, the plot for Product 3 is the opposite of Figure 6; in this case, Method 5 is now the best, the other *P*-based methods are the worst, and the partial observability measures fall in between.

These results are all expected as Method 5 favors reduction of error in Product 3 at the expense the other products, while the other *P*-based metrics sacrifice the performance of Product 3 in favor of improvements in the other two. The more balanced schedules generated by Methods 7 and 8 lie between these two extremes.

Finally, the Pareto plot over all products (Figure 4.26) shows that the schedules created by Methods 3, 4, and 6 which prefer reductions in output error for Products 1 and 2 and increases in Product 3 produce better overall results than the opposite types of schedules generated by Method 5. These schedules are also superior to the more balanced schedules generated by Methods 7 and 8. Again, this is due to the fact that the *P*-based methods can incorporate information about the tools and products through use of the $Q$ matrix which allows them to differentiate the states based on more than just the frequency and types of their combinations.

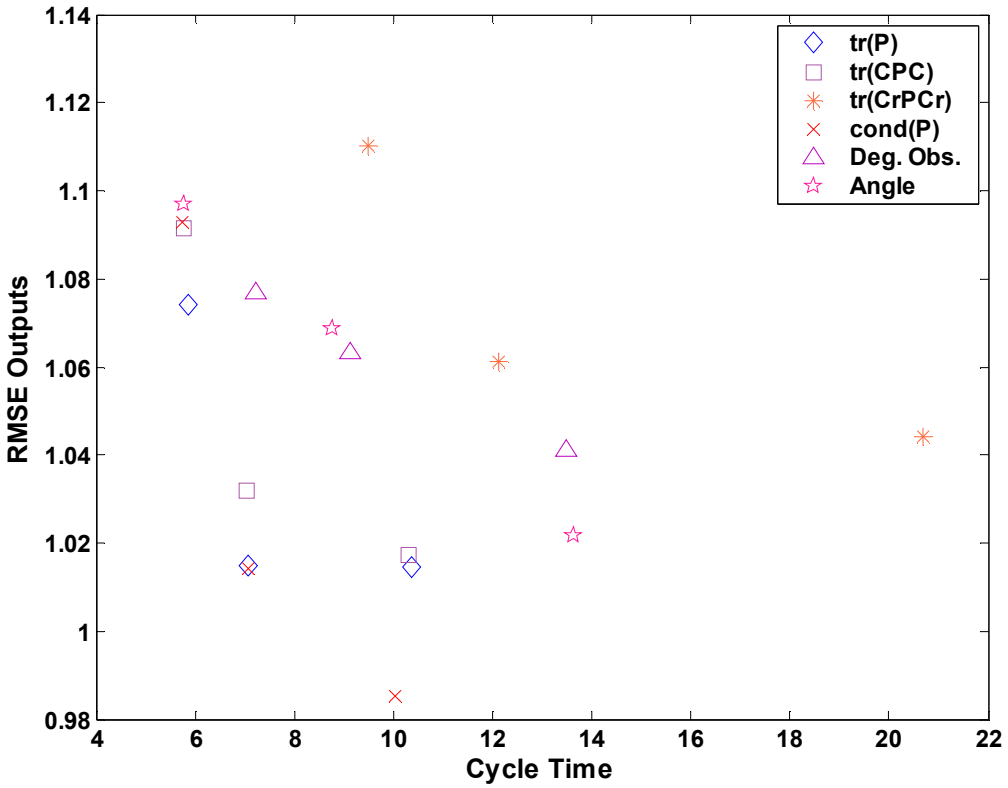


Figure 4.26: Pareto plot of output error v. cycle time for all products in Test 7 for a system with Kalman update delays.

### 4.6.3 Conclusions from Pareto Analysis

Because the system under study has two interacting portions with separate performance metrics, it is appropriate to use Pareto plots for its analysis. The primary measures used for the two systems are the average product cycle time for the scheduling system and the RMSE of the measured outputs. While results taken over all products and tools for the various test systems yield no clear cut winners, it is possible to create a process of elimination by using some particular test cases which are important in the real processing environment.

The two cases highlighted here are the system with an unbalanced mix and a low running product (Test 4) and the system with different levels of variation in tool and product performance (Test 7). In the first case, the cycle time and output error performance are especially important for the low running product, and FIFO, SPT, and the algorithm based on $\text{trace}\left(C_r P C_r^T\right)$ are eliminated based on poor performance for the low running product. In the second test case, the major separation in the remaining methods is between those based on estimate error covariance, $P$, and those based on partial observability measures. Due to the fact that the matrix, $Q$, which gives an estimate of the noise variance in the tool and product states, is a component of the calculation of $P$, Methods 3, 4, and 6 have a distinct advantage over Methods 7 and 8 in this scenario. Method 6 in particular is able to use the knowledge about relative levels of variance in products and tools to match high variance products to low variance tools and also to increase the frequency of quals on the high variance tool. Therefore, it should be preferred in cases like Test 7.

## 4.7 CONCLUSIONS

In this chapter, the idea of scheduling based on objectives related to the control system is explored. Multiple methods based on either the error covariance matrix of the Kalman filter, $P$, or the system information matrix, $L_{0,k}$, are proposed and described in detail. Next, a discrete event simulation is setup and tested to act as the basis for a multi-tool, multi-product processing system. The use of the discrete event simulation introduces the complication of overlapping runs and measurement delays, and the estimation scheme is adjusted accordingly to operate under these conditions. The new combined system is then simulated using the same set of tests used in Section 3.6, and it is found that the control system operates less effectively in the newer environment with delays. Additionally, qualification runs play a larger role in maintaining good control as

the error levels show a more distinct downward trend as the number of quals in the system is increased, this is especially true for the larger system with added tools.

Finally, the control based scheduling methods are tested against the more traditional methods, FIFO and SPT. It is found that while SPT always leads to minimal mean cycle times for all products in the system, is not ever the best from a control point of view. Also, because SPT is sensitive only to product processing times, it can create problems in systems where low running products exist; if the low runner has one of the higher processing times in the distribution, then it can experience long delays. FIFO does a good job of balancing CT amongst the products and is often competitive from a control standpoint; however, much of this has to do with the fact that the underlying arrival process is already randomized so that good mixing of products across tools is inherent. If the underlying arrival process were more like the schedules generated by Method 5, in which long strings of the same product are run while other products are left in queue, it would have the same weaknesses that Method 5 displays in Test 7 (i.e., trouble dealing with unequal noise levels in the process state).

Through the use of two special cases (Tests 4 and 7), the better scheduling methods can be determined through a process of elimination. For the reasons listed above, SPT and FIFO are eliminated as their weaknesses are particularly apparent when looking at results for individual products in Test 4. In the case where an unequal distribution of products is the major distinguishing factor between products, all of the control based methods except Method 5 perform well. If this method is also eliminated, and the remaining methods are run on Test 7 (unequal variance for tools and products), it is found that the remaining *P*-based methods outperform those based on $L_{0,k}$.

# Chapter 5:

# Conclusions, Contributions, and Future Work

## 5.1 Conclusions and Contributions

### 5.1.1 Introduction and Comparison to Existing Work

Two of the major systems involved in the automation of modern semiconductor manufacturing facilities are the advanced process control system and the scheduling system. Practitioners note that conflicts can arise between the systems and that it maybe possible to regulate their interactions to minimize conflicts or generate mutually beneficial solutions [8]. With this in mind, a framework for studying the problem is built around a non-threaded control system which uses a Kalman filter as its estimator. Such a controller is suggested in [54] and is shown to have potential for integration with scheduling for a series of small scale examples (only one replicate with 20 samples per experiment).

Additionally, as the semiconductor industry has shifted toward high-mix environments with many tools and products, the use of threaded control has become more difficult to implement. As a result, non-threaded control systems which share information from measurements in order to identify contributing factors from various sources such as tools and products have been suggested and studied [53, 86, 93, 121, 122, 123, 127, 131]. By defining each output as the sum of individual contributions from each context group (tools, products, etc.), the system can be modeled in a state space form with the set of states equal to the full set of individual context items. The obvious problem with this approach is a lack of system observability. One solution is to allow for

qualification runs in the system to uniquely identify tool states which is the approach used in [53] and also in this work.

The primary focus of previous non-threaded control studies has involved improving performance through changes to the model form or the estimator used. The different forms of the system have been compared to each other and/or threaded EWMA systems on the basis of estimation performance. In these tests, different outputs are generated by randomly selecting one item from each context group (products, tools, etc.) based on predetermined probabilities of occurrence, but the actual manner in which the runs occur in parallel processing systems is not considered. Additionally, other than the small examples in [54] mentioned above, no thought is given to how the thread selection might effect the control performance.

### 5.1.2 Unique System Setup

The goal of this research is to expand upon the ideas in [54] by using a similar control system but with a more realistic process flow and scheduling system to test various scheduling methods on the basis of both control and scheduling performance. First, some improvements are made to the control system models from [53, 54]. Starting with the model equations for a multi-tool, multi-product, non-threaded control system (Equations 3.10–3.11) from [53], it is noted they are set up only for a single measurement system with inputs for all tools lumped together in $u$. Therefore, the model is reformulated to allow for easier recording of inputs for each tool and to allow for the possibility of multiple measurements at a given step (Equations 3.31–3.32).

Additionally, it is noted that the tool and products states are not functions of the input or the adjustment state which represents the input in the state vector. Therefore, it is determined that this adjustment state can be removed and replaced by a feedforward input term in the output equation. Alternatively, the output could be replaced by the

difference between the measurement and the input multiplied by the appropriate tool gain. In either case, elimination of the adjustment state has the effect of simplifying the model and making the estimation more stable (see Table 3.2).

To create a test environment that more realistically mimics the flow of products through tool groups in a fab (i.e., similar tools running in parallel and fed by a queue), a discrete event simulation (DES) system is built from scratch and tested against existing queueing theory for accuracy. To represent the combined plant, the DES is then integrated with the state space estimation and control system to allow for interactions between the two. Also, all system metrics can be tracked simultaneously including cycle times, tool utilizations, product distributions, output errors, and estimation errors.

One major enhancement of the new combined system over previous studies is the ability to have overlapping runs in a parallel environment and track the runs from start to finish. This feature creates scenarios such as delayed and out of sequence measurements (OOSM) as would happen when controlling a real parallel process. The measurement delays are defined in terms of the number of product starts that occur on other tools between the start and measurement of the run under study. These delays are quantified through simulations and an equation is given for their theoretical maximum in terms on the number of tools and products in the system.

The updating scheme for the estimator is adjusted to run under these less than ideal conditions, but a separate idealized system in which all measurement information is received on time and in order also runs in parallel for comparison. It is found that the delays in updates can cause large differentiation from the ideal system including instability if the Kalman filter is tuned too aggressively. It is also seen in at least one instance during the Pareto analysis that these delays can cause a scheduling method to go

from one of the best in terms of output error for the ideal case to one of the worst in the realistic case.

### 5.1.3 Development of New Scheduling Algorithms

After integration of the DES with the control system, various scheduling algorithms were developed which actively use information from the control system to select the best tool/product combination for the next run based on a given control-based objective. As suggested in [54], the trace of P (i.e., the total state error variance) can be used as one of these objectives. Additionally, several other objectives based on $P$ were suggested by the author:

- The trace of $CPC^T$. This objective represents the total error variance of the outputs rather than the states. It incorporates not only the state variance but also information from the covariance terms between product and tool states.
- The trace of $C_r P C_r^T$. This objective is used to select the single output with the lowest error variance. Used to see if minimizing a single output at each step yields better results than trying to minimize the total variance at each step.
- The condition number of $P$. Condition number indicates the sensitivity of a solution to perturbations or how well a problem is conditioned. In this case it gives an indication of how accurate the estimates are. By scheduling to minimize this number, better estimates should be found.

In addition to the objectives that use the error covariance, alternative methods based on the theory of local or partial observability of systems with incomplete measurements were developed. Two objectives that have their roots in the aerospace literature [116, 117] were adapted to the non-threaded control system and used for scheduling. The local observability matrix, $L_{0,k}$, can be updated at each step using the reduced $C$ matrix, $Cr$, and the output noise covariance matrix, $R$, from the Kalman filter.

This matrix is then used to calculate the objectives through use of a singular value decomposition (SVD). The two objectives are:

- The degree of observability. This is measured by the smallest singular value of $L_{0,k}$, and gives an indication of how close the system is to full observability. The scheduler seeks to maximize this value.
- Angle to the least observable subspace. The last singular vector of $L_{0,k}$ gives the direction of the least observable subspace; the available output that is most collinear to this vector is chosen. Measurement of this output provides maximum information on the least observable subspace, thus increasing the observability of the system.

Finally, after initial testing with a small two state system, it seems that the two objectives above are somewhat sluggish in their response to new measurements, so moving window approaches are proposed for both. A window length of $n$ is decided upon as this is the minimum path length for testing pathwise observability. Additionally, a window length of $2n$ is also tested as a compromise between the shorter window and the full matrix information versions. A summary of all of the methods is found in Table 4.7.

### 5.1.4 Conclusions from System Analysis

Because information on tool availability, products in queue, and the current state of the controller are available to the algorithmic scheduler at every step, it is possible to do scheduling in conjunction with control on a run-by-run basis in a parallel processing environment. Because all statistics are tracked for both systems simultaneously, it is also possible to evaluate the various scheduling methods on the basis of both control and scheduling performance. Both of these capabilities are unique to this research.

To better understand the behavior of the systems, a series of tests is setup and simulation experiments are performed. The first tests seek to gage the effects of qual runs on control system performance. Quals are included in the model to ensure that the full system is observable. The largest reductions in both output and estimation error are seen when going from zero quals to 1% quals and additional improvements are usually seen up to 3% quals before there is a leveling off. This is truer for estimate error than for output error and for systems with step and drift disturbances than for other types of disturbances. In practice, the 1-3% range for quals is acceptable for areas of the fab that use quals to identify the tool state so it is good that most of the process improvement is found in this range. Obviously, fewer quals are better since they represent non-production runs and the cycle time penalties incurred by the system make their use dubious beyond the 3% level considering the marginal improvements seen in control performance.

Finally, the system performance is analyzed in terms of Pareto plots which can show control and scheduling metrics simultaneously. Pareto plots for the all combinations of the seven test systems and two disturbance types (Sig1 and Sig4) are generated and studied. Across the full range of tests there are only a few generalizations that can be made, but no scheduling method is clearly the best for all products in all situations. Therefore, to eliminate some methods from consideration, two special scenarios seen in practice are selected and studied in more detail. Some major conclusions from the Pareto analysis are as follows:

- For the white noise only case (Sig1), there is little difference in control performance across the different scheduling methods. The results that follow are valid for systems with disturbances that are more difficult to control; in particular

Sig4 which uses IMA(1,1) for the products and shift and drift disturbances for the tools.

- An analysis of the Pareto plots for output RMSE v. mean cycle over all products for the various tests does not reveal a single best scheduling method. The only consistent result is that, as expected, SPT yields the minimum mean cycle time over all products. Unfortunately, depending on the system, SPT is a very poor choice when looking at results for individual products.
- FIFO does a good job of balancing CT amongst the products and is often (but not always) competitive from a control standpoint; however, much of this has to do with the fact that the underlying arrival process is already randomized. If the underlying arrival process is a poor fit from a control standpoint, FIFO does nothing to correct this.
- The first special case is Test 4, which has an unbalanced mix of products. Here the low running product is considered the most valuable. The proposed control-based methods all perform well accept for the one based on trace$\left(C_r P C_r^T\right)$ (Method 5) which tends to put off processing the low runner for long periods of time. SPT is only viable when the low runner also has the lowest processing time, but even in this case, its control performance is not as good as the control-based methods. FIFO is found to be inconsistent due to its dependence on the incoming order of products. The remaining control based methods all give priority to the low running product automatically and reduce both cycle time and output error compared to the other methods.
- The second special case is Test 7, which has unequal noise variance applied to the states. For both tools and products there is a high, medium, and low variance option. In order to increase control performance, a desired scheduler should

assign the low variance (i.e., easier to control) product to the high variance tool and vice versa for the high variance product. Due to results of Test 4, FIFO and SPT are eliminated outright. The results for the other methods show that Method 5 does the opposite of the desired scheduler while the rest of the *P*-based methods demonstrate the desired behavior. In fact, the algorithm based on the condition number of *P* (Method 6) does the best by skewing the distributions of the high and medium variance products toward the low variance tool the most and does the opposite for the low variance product. Additionally, Method 6 distributes quals according to the error level of the tool. Methods 7 and 8 fall short of the performance demonstrated by the better *P*-based methods because the do not contain a mechanism for capturing and processing the information on the differences between the states (i.e., their objectives are not functions of *Q*).

- Because the perform well and very well in Test 4 and Test 7 respectively, Methods 3 (trace(*P*)), 4 (trace($CPC^T$)), and 6 (cond(*P*)) are preferred and especially Method 6 in a scenario like Test 7.

## 5.2 FUTURE WORK

### 5.2.1 Use of Multi-Objective Optimization Approaches

While control-based scheduling methods are studied extensively in this research, there are definitely areas the can be expanded upon and improved. As mentioned at the end of the previous section, there is no clear cut best method for reducing both process errors and cycle times. Each method has its own strengths and weaknesses. With two seemingly conflicting objectives, this is the type of problem that is often addressed with multi-objective optimization [128]. Because a large number of solutions are possible to balance the two objectives, genetic algorithms (GA) could be used to generate a set of

Pareto optimal solutions, allowing the user to select the point on the tradeoff curve that best suits his or her needs [100].

Because all of the control-based objectives proposed for scheduling can be calculated without measurement information, it is possible to calculate the objectives for any schedule where the process order is specified. Therefore, a suboptimal but real time solution could be a moving horizon-type approach where a schedule is generated over the horizon and the control-based scheduling objective is calculated for the schedule. This is again a bi-criteria optimization problem, where the two objectives need to be weighted appropriately so that a suitable solution is found. Once the full schedule is found, the first scheduled product in the new schedule is run. As long as no new products arrive in the queue, the same schedule can continue to be used. To keep from having to make a new schedule from scratch after each new product arrival, a reactive scheduling algorithm that makes appropriate adjustments to the old schedule could be employed [36] with a threshold on the value of the control-based objective determining whether to reactively adjust the schedule or reschedule completely.

While many semiconductor processes use linearized models for control, there are examples of non-linear controllers being used for run-to-run control [129]. In the cases like [129], where an extended Kalman filter (EKF) is used, it is still possible to use approaches like those employed here to schedule according to a control-based objective. Many approaches have been developed in the realm of sensor selection in which observability measures for non-linear systems can be calculated. In [130], the authors solve the sensor placement problems by performing PCA analysis on empirical observability Gramians which is similar to the approach used here of performing SVD on the information matrix. They then define a penalty function based on sensor cost to constrain a GA and solve for the optimal sensor design. A similar approach could be

implemented to generate schedules for the approaches mentioned above by having a penalty function based on cycle time, makespan, or some other scheduling based objective.

### 5.2.2 Improvements to Control Based Objectives

While many control based objectives are studied in this work there are other possible objective functions based on $P$ and $L_0$. One other objective is created by making a slight change to the methods based on $L_0$ that have already been suggested. While it is true that the singular vector associated with the smallest singular value of $L_0$, $\sigma_n$, represents the LOSS, it is possible that this is only a portion of the entire unobservable subspace. As stated in [117], it is possible to set up a threshold that can be used to find all singular values associated with unobservable subspaces, and any singular value that falls below this threshold shall be identified as such. The threshold for this test of observable subspaces is equal to $\sigma_1 * \varepsilon$, where $\sigma_1$ is the largest singular value and $\varepsilon$ is the machine roundoff error. By using this threshold, the angle based scheduling method can be adjusted to include the full unobservable subspace rather than just the LOSS. This should yield more accurate choices for the best measurement to choose when using the angle based metric.

Finally, it should be noted that the methods outlined in [117] are similar to those of Ham and Brown presented in [119]. In [119], the authors scale the error covariance matrix with respect to its initial condition and then perform the SVD on the scaled matrix to find its singular values and singular vectors. In this case the largest singular value represents the variance of the state or linear combination of states that is least observable. As the authors point out, this method greatly simplifies the analysis of the error covariance matrix for the following reasons. Without performing the SVD step first,

there are $\frac{n(n+1)}{2}$ unique elements (due to symmetry) in the error covariance matrix to examine, or if one attempts to simplify the analysis by looking only at the diagonal entries of $P$ (i.e., the trace), $n$ unique entries. In the first case, the SVD step simplifies the task to looking only at the $n$ singular values. For the second case, the SVD based method has the same number of values to study but has a distinct advantage. The simplified method of observing only the diagonal elements of $P$ spots large variance for individual states but not for combinations of states. In the SVD based method, singular vectors associated with large singular values can point out combinations of states which have higher variance and are least observable; therefore, these vectors can be used in an angle based scheduling method similar to the one above. Additionally, the scaling of the error covariance matrix by its initial value should eliminate the problem of sensitivity of $P$ to $P_0$ mentioned in [117].

### 5.2.3 Suggestions for Further Testing

While many simulations are run to test the performance of the various scheduling methods, the different systems change only one parameter at a time (e.g., number of tools or product mix). It would be interesting to change multiple factors and see how the scheduling algorithms perform. Additionally, it is seen in the results for Test 7, that by setting the entries of $Q$ equal to the noise levels of the states, one can give preference to given products on given tools. This could be tried in systems where such preferences are desired for reasons other than the inherent variance of the given products and tools.

Also, while Method 5 is found to be a poor choice for the examples studied, one situation in which it would be useful is on a set of tools where setup avoidance at a change-over is important. I such a scenario, there is large cycle time penalty associated with setting up the tool for a different product when a change-over occurs. The algorithm

based on $\text{trace}\left(C_r P C_r^T\right)$ shows a propensity for scheduling long campaigns of the same product on the same tool automatically so it could be useful in the setup avoidance case.

Along these lines, it would be interesting to expand the scheduling simulation to include this scenario or others such as those studied in [134] where the metrology delay is considered separately from the processing time. Additionally, [134] contributes to the literature on sampling plans in semiconductor processing. All of the tests run in this work use only systems with full measurement data (i.e., every run is measured and fed back to the controller). This is often not a realistic case in practice and the use of the control based schedulers studied here in the creation of sampling plans or in the presence of limited measurement information would be of great interest. This has been done in a limited manner for the method based on trace($P$) in [54] and [135].

Finally, it is noted that the scheduling algorithms proposed here are useful with other non-threaded control systems as well. The P-based scheduling algorithms can be applied in any system that uses a Kalman filter as its estimator, and the partial observability methods can be used with any system that uses a state space model. If the model doesn't use a Kalman filter, then $L_{0,k}$ is calculated without using the matrix, $R$, for scaling. Of particular interest are the systems that us equality constraints in place of qual runs to insure observability of the overall system such as [91–93] and [127].

# Appendices

## APPENDIX A: DEFINITIONS OF SCHEDULING TERMS

The following definitions (with a slight modification to the term, scheduling) were used in the survey conducted for the FORCe I project [7] and were used for a new survey as part of the FORCe II [8] project as well.

- Planning: The development of detailed capacity and material plans that assess the fab's capability to meet market demands. Decisions here include determining product mix, new equipment purchases, staffing levels, etc.
- Order Release: The determination of when to release lots to the manufacturing floor.
- Scheduling: The creation of an overall plan of how lots will move through the fab. Scheduling generally encompasses a more holistic view of the fab than dispatching and may incorporate such knowledge as planned tool maintenance, product mix changes, demand forecasts, etc. In general, scheduling is performed on a shiftly, daily or weekly basis to provide a nominal schedule for the entire interval. This step is optional.
- Rescheduling: The re-evaluation of a scheduling rule decision within the original scheduling time period. This is typically done either at fixed intervals or when a schedule deviates from its original plan.
- Dispatching: The immediate assignment of a specific resource to one of several possible lots. It answers the question: which lot should be processed on this machine now? If scheduling has been performed, the goal of dispatching is to choose the lots that best meet the schedule. If scheduling has not been performed, dispatching rules (such as FIFO, critical ratio) are chosen that have been shown to work well for a given factory measure(s).

The roles of scheduling, rescheduling, and dispatching are often collectively referred to as scheduling. In this work, the combination is referred to as scheduling/dispatching or where specified, just scheduling for easier reading. Figure A.1 gives a flow chart representation of the hierarchy of the various systems.

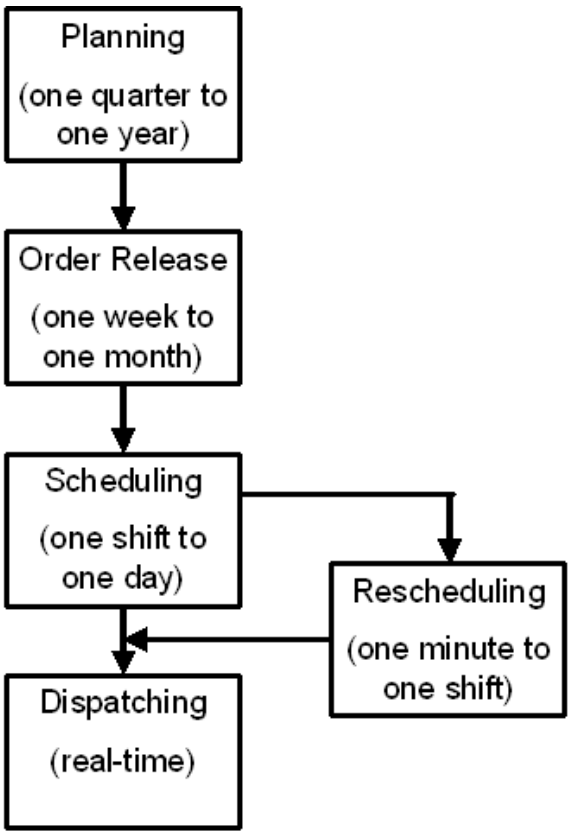


Figure A.1: A flow diagram demonstrating the various levels of product flow control.

## APPENDIX B: EWMA CONTROL

### B.1 Derivation of EWMA Controller in Integral Form

Starting from the standard form of the EWMA controller (Equation (2.3)), the integral form (Equation (2.4)) is derived:

$$\hat{x}_{k+1} = \lambda(y_k - bu_k) + (1-\lambda)\hat{x}_k \tag{2.3}$$

$$\hat{x}_{k+1} = \lambda y_k - \lambda b u_k + \hat{x}_k - \lambda \hat{x}_k \tag{B.1}$$

$$\hat{x}_{k+1} = \hat{x}_k + \lambda(y_k - bu_k - \hat{x}_k) \tag{B.2}$$

$$\hat{x}_{k+1} = \hat{x}_k + \lambda(y_k - (bu_k + \hat{x}_k)) \tag{B.3}$$

$$\hat{x}_{k+1} = \hat{x}_k + \lambda(y_k - \hat{y}_k). \tag{2.4}$$

### B.2 The EWMA R2R Controller Algorithm

A general form of the EWMA run-to-run controller for a single input, single output process is shown below:

1. Given the gain, *b*, from experiments, a starting value for the state estimate, $\hat{x}_0$, and the target, *T*.
2. Calculate the inputs, $u_k$, from Equation (2.5).
3. Using the inputs from Step 2 and the current value of $\hat{x}_k$ with Equation (2.2), calculate the predicted value for the outputs, $\hat{y}_k$.
4. After the process step finishes, take a measurement of the output, $y_k$.
5. Using the output residual, $y_k - \hat{y}_k$, with the observer equation, (2.4), get an updated estimate for the disturbance, $\hat{x}_{k+1}$.
6. Go to Step 2 and repeat for the next run.

## APPENDIX C: SUMMARY AND ANALYSIS OF RESULTS FROM ZHENG ET AL [87]

### C.1 Simulation Model

In Zheng et al [87], the effect on an EWMA controller of switching between two different products on a single tool is evaluated. In order to obtain systematic results, the two products are constrained to running in a cyclic fashion. For example, a cycle length of four and a 50-50 product split would result in the following sequence of runs: [1 1 2 2 1 1 2 2 … 1 1 2 2 ].

The output from the two product, one tool system is defined as a linear function:

$$Y_{it+n} = \begin{cases} \alpha_1 + \beta_1 X_{it+n} + \eta_{it+n} & 1 \le n \le j \\ \alpha_2 + \beta_2 X_{it+n} + \eta_{it+n} & j+1 \le n \le i \end{cases} \tag{C.1}$$

where $\alpha_1$ and $\beta_1$ are the product one intercept and slope, respectively, and $\alpha_2$ and $\beta_2$ are the product two intercept and slope, respectively. The variable $i$ is the number of runs in a cycle (i.e., the campaign length), and $j$ is the number of consecutive runs of product 1 in a cycle so that $i - j$ is equal to the number of consecutive runs of product 2 in a cycle; this is also called the break length for product 1. The index $t$ is the cycle index and $n$ is the intercycle index. Therefore, $Y_{it+n}\left(n = 1, 2, \cdots, j\right)$, and $Y_{it+n}\left(n = j+1, j+2, \cdots, i\right)$ are the outputs of products 1 and 2, respectively, and $X_{it+n}\left(n = 1, 2, \cdots, i\right)$ are the control actions (inputs) at a given run number, $it+n$. There is a common disturbance $\eta$ which represents the change in tool condition and is modeled as an IMA(1,1) disturbance with drift:

$$\eta_k - \eta_{k-1} = \delta + \varepsilon_k - \theta\varepsilon_{k-1}, \tag{C.2}$$

where $\delta$ is the deterministic drift term, $\theta$ is the moving average parameter ($\theta = 1$ is a random walk and $\theta = 0$ is white noise), and the noise term, $\varepsilon_k$ is $N\left(0, \sigma^2\right)$.

## C.2 Tool Based EWMA Control

A single, observed tool disturbance is found from input-output data and the linear models for the two products:

$$\hat{\eta}_{it+n} = \begin{cases} Y_{it+n} - \left(a_1 + b_1 X_{it+n}\right) & 1 \le n \le j \\ Y_{it+n} - \left(a_2 + b_2 X_{it+n}\right) & j+1 \le n \le i \end{cases} \tag{C.3}$$

where $a_1$ and $b_1$ are fixed model parameters found from experiments that approximate $\alpha_1$, $\beta_1$, and $a_2$ and $b_2$ are similar parameters that approximate $\alpha_2$, $\beta_2$.

The observed "tool noise" is filtered using an exponentially weighted moving average (EWMA) to get an estimate of the disturbance:

$$\tilde{\eta}_{it+n} = \lambda \hat{\eta}_{it+n} + \left(1-\lambda\right)\tilde{\eta}_{it+n-1}, \tag{C.4}$$

where $0 \le \lambda \le 1$ is the EWMA weighting factor.

The disturbance estimate is then fed into deadbeat control laws with output targets of zero for the two products:

$$X_{it+n} = \begin{cases} \dfrac{0 - a_1 - \tilde{\eta}_{it+n-1}}{b_1} & 1 \le n \le j \\ \dfrac{0 - a_2 - \tilde{\eta}_{it+n-1}}{b_2} & j+1 \le n \le i \end{cases}. \tag{C.5}$$

By substituting the expressions for $\hat{\eta}_{it+n}$ and $X_{it+n}$ into (C.4), the expression simplifies to the familiar integral form of the EWMA controller:

$$\tilde{\eta}_{it+n} = \lambda Y_{it+n} + \tilde{\eta}_{it+n-1}. \tag{C.6}$$

Finally, an important parameter in discussion of the controller performance is the plant-model mismatch for the gain, $\xi$:

$$\xi = \frac{\beta}{b}, \; where \begin{cases} \xi = 1 & no\ mismatch \\ \xi < 1 & gain\ overestimated \\ \xi > 1 & gain\ underestimated \end{cases}. \tag{C.7}$$

The relationship of the plant-model mismatch to other system parameters can be seen by substituting equation (C.5) into equation (C.1) to derive the closed-loop output equation:

$$Y_{it+n} = \begin{cases} \alpha_1 + \eta_{it+n} - \xi_1 \left( a_1 + \tilde{\eta}_{it+n-1} \right) & 1 \le n \le j \\ \alpha_2 + \eta_{it+n} - \xi_2 \left( a_2 + \tilde{\eta}_{it+n-1} \right) & j+1 \le n \le i \end{cases}. \tag{C.8}$$

### *C.2.1 System Stability and Simulation Results for Tool-based Control*

Figures C.1-3 display the outputs of simulations that demonstrate several of the previously mentioned scenarios. In Figure C.1, the system is simulated with tool-based EWMA control with the mismatch parameters having different values and a stationary disturbance (i.e., $\delta = \theta = 0$). As mentioned before, this is a trivial case and the system is stable. In Figure C.2, the opposite approach is taken; the mismatch parameters are identical, and the drift and integrating parts are added to the disturbance term. As expected, the system is stable; in fact, the system is equivalent to a single product system. Because of the drift term, the controlled output has a steady state offset from the target value of zero. The red line shows the mean value of the simulated output, which is equal to 0.118. According to Del Castillo [120], the steady state offset of a single product system under EWMA control will have an offset equal to $\delta / \lambda \xi$, which in this case is equal to 0.119. It is well known that a double EWMA (DEWMA) controller can be used to compensate for the drift in this sort of disturbance and eliminates the offset. Results of a DEWMA filter applied to the same simulation can be seen in Figure C.3. Unfortunately, the DEWMA controller still demonstrates the same instability problems as the EWMA. Several unstable systems are demonstrated in Figures C.4-6 where it can be seen that the drift disturbances cause the system instability to grow quickly (e.g., Figure C.4) while the integrating portion of the disturbance term causes an unbounded oscillation (e.g., Figure C.5). For these simulations, the drift is the more dominant instability when both terms are included (e.g., Figure C.6).

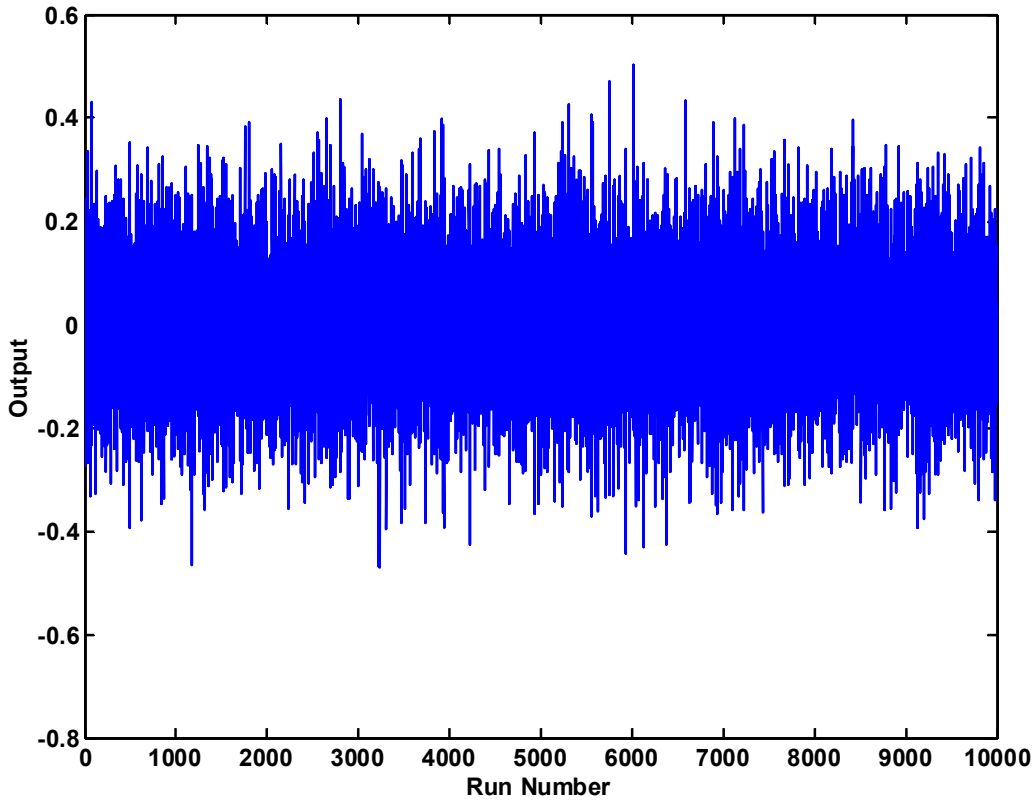


Figure C.1: Tool-based EWMA control with different plant-model mismatch for each product and a stationary disturbance. The system is stable. Parameter values: $\lambda = 0.7$, $\delta = 0$, $\theta = 1$, $\xi_1 = 1$, $\xi_2 = 1.2$, $i = 10$, $j = 5$.

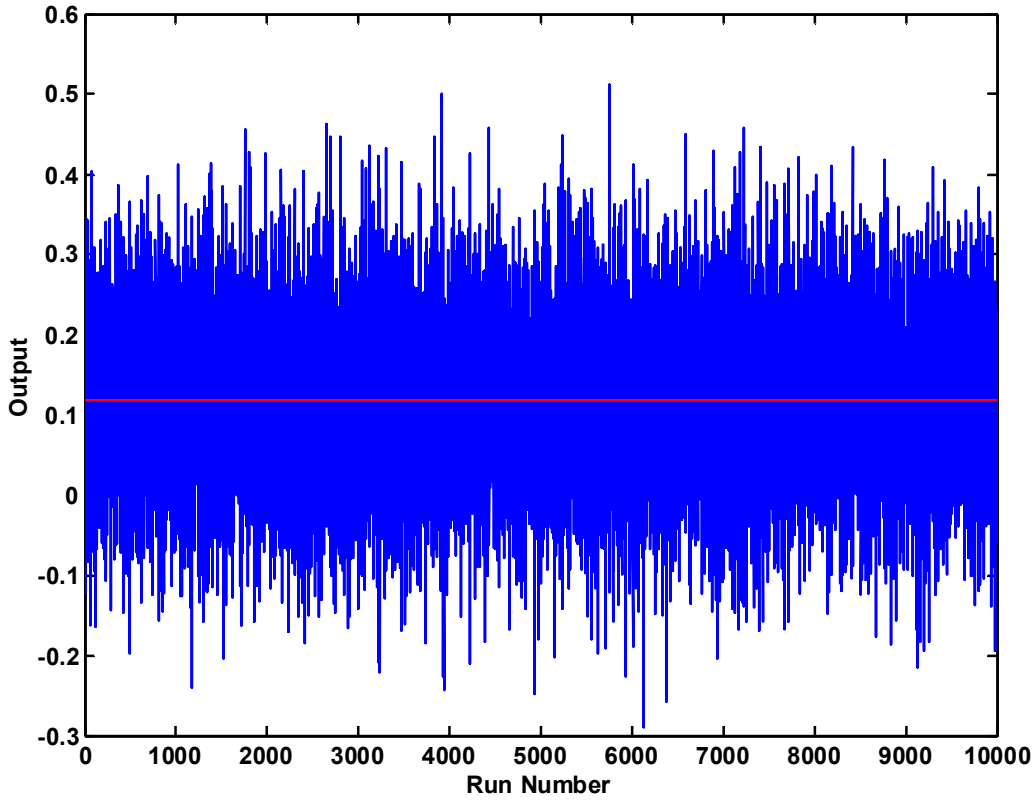


Figure C.2: Tool-based EWMA control with the same plant-model mismatch for both products and a non-stationary disturbance. The system is stable, but the output mean is offset from the target value of zero. Parameter values: $\lambda = 0.7$, $\delta = 0.1$, $\theta = 0.5$, $\xi_1 = 1$, $\xi_2 = 1.2$, $i = 10$, $j = 5$.

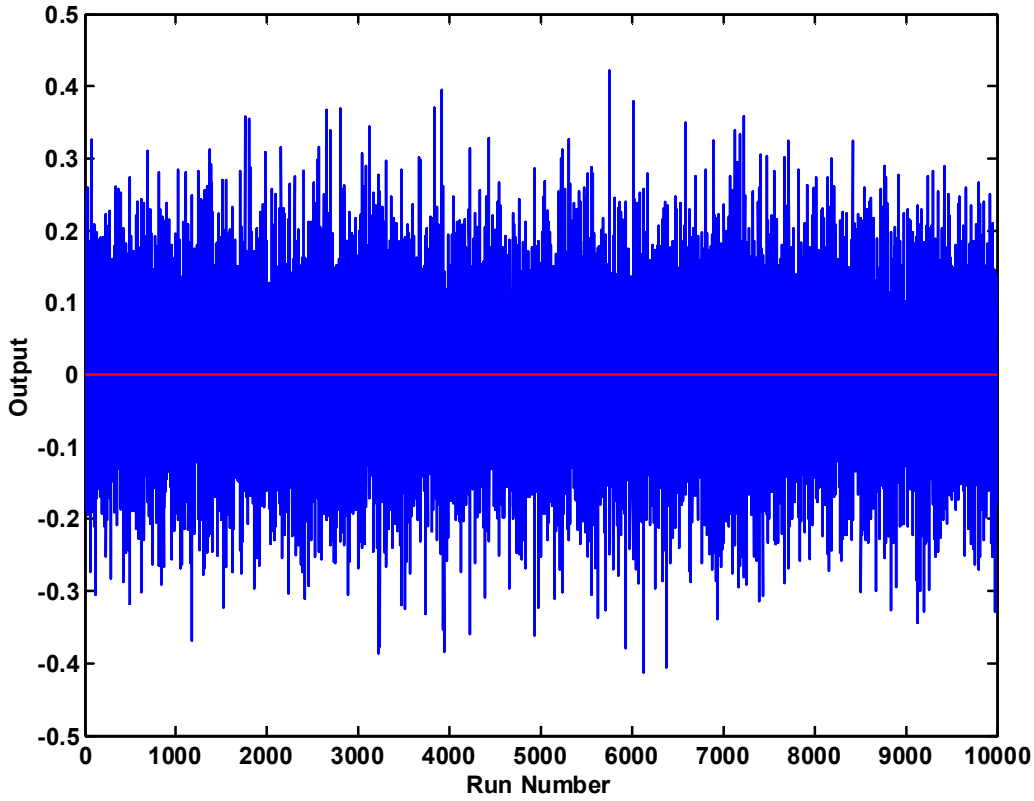


Figure C.3: Tool-based DEWMA control with the same plant-model mismatch for each product and a non-stationary disturbance. The system is stable and the offset seen with EWMA control is eliminated. Parameter values: $\lambda_1 = 0.7$, $\lambda_2 = 0.1$, $\delta = 0.1$, $\theta = 0.5$, $\xi_1 = \xi_2 = 1.2$, $i = 10$, $j = 5$.

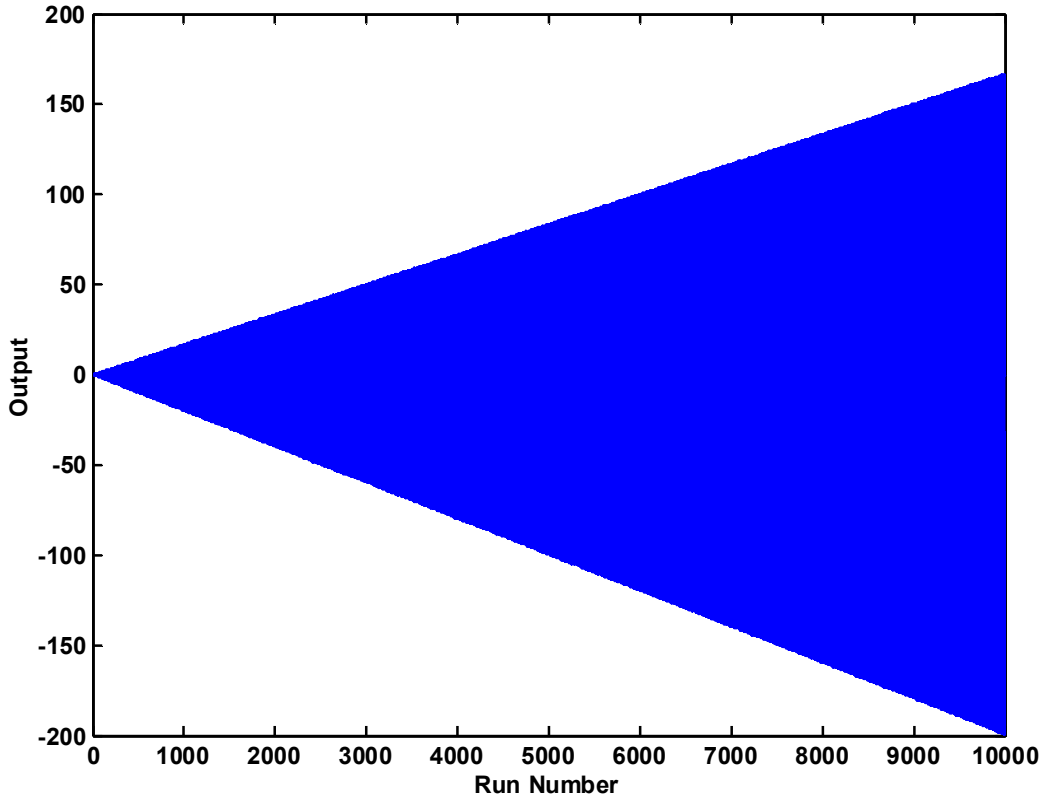


Figure C.4: Tool-based EWMA control with different plant-model mismatch for each product and drift added. The system is unstable. Parameter values: $\lambda = 0.7$, $\delta = 0.1$, $\theta = 1$, $\xi_1 = 1$, $\xi_2 = 1.2$, $i = 10$, $j = 5$.

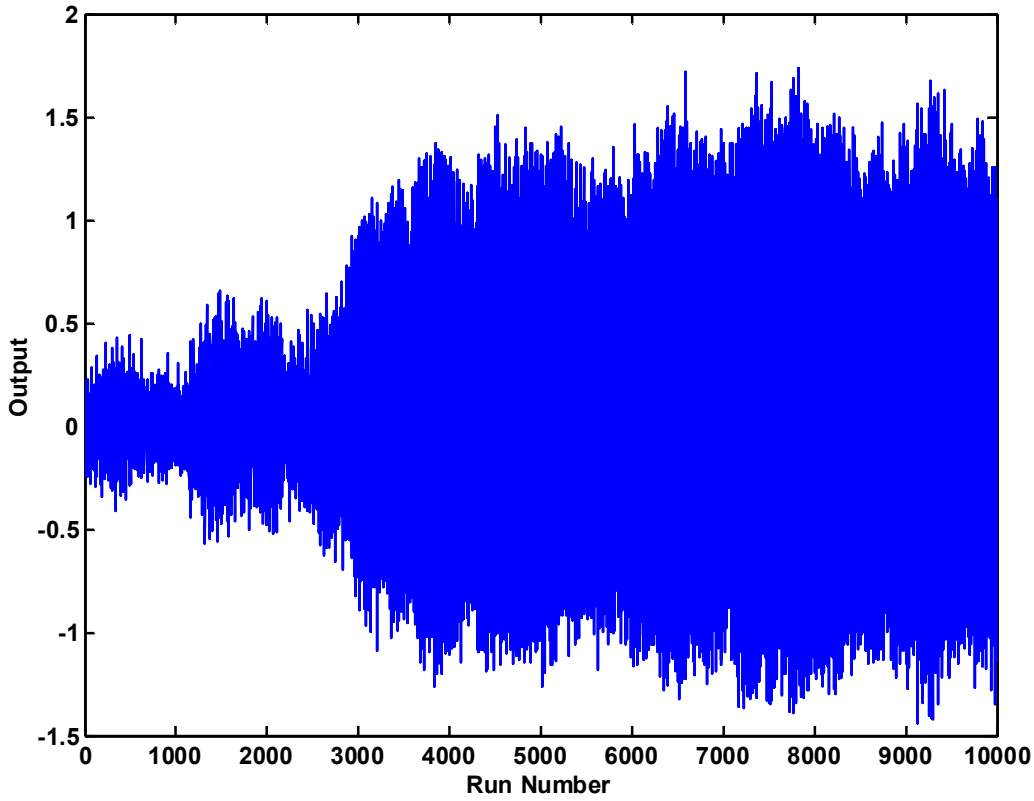


Figure C.5: Tool-based EWMA control with different plant-model mismatch for each product and integrating noise term added. The system is unstable. Parameter values: $\lambda = 0.7$, $\delta = 0$, $\theta = 0.5$, $\xi_1 = 1$, $\xi_2 = 1.2$, $i = 10$, $j = 5$.

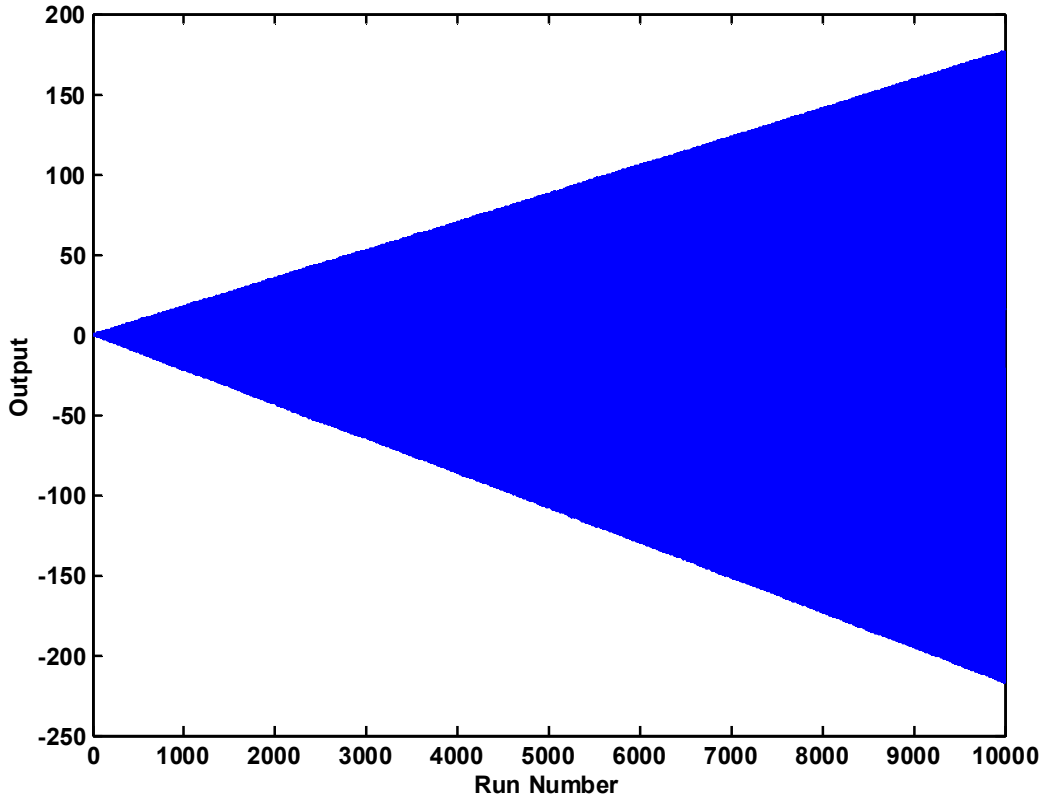


Figure C.6: Tool-based DEWMA control with different plant-model mismatch for each product and a non-stationary disturbance. The system is unstable. Parameter values: $\lambda_1 = 0.7$, $\lambda_2 = 0.1$, $\delta = 0.1$, $\theta = 0.5$, $\xi_1 = 1$, $\xi_2 = 1.2$, $i = 10$, $j = 5$.

By looking at one of the unstable examples more closely, it is interesting to note that the unbounded growth of the output occurs at the switching points while points in between stay in a stable region near the target value. Figure C.7 shows the simulation results when a longer cycle and campaign length are used in a system with different plant-model mismatches and a drifting disturbance. The circular and triangular markers highlight the first outputs after the system switches from product to another. It is easy to see that the system is quite stable between switching points but the output error grows at

each switching point as a function of the cycle number (a mathematical proof is provided in [87] where an equation for the product 1 output at the switching point is given).

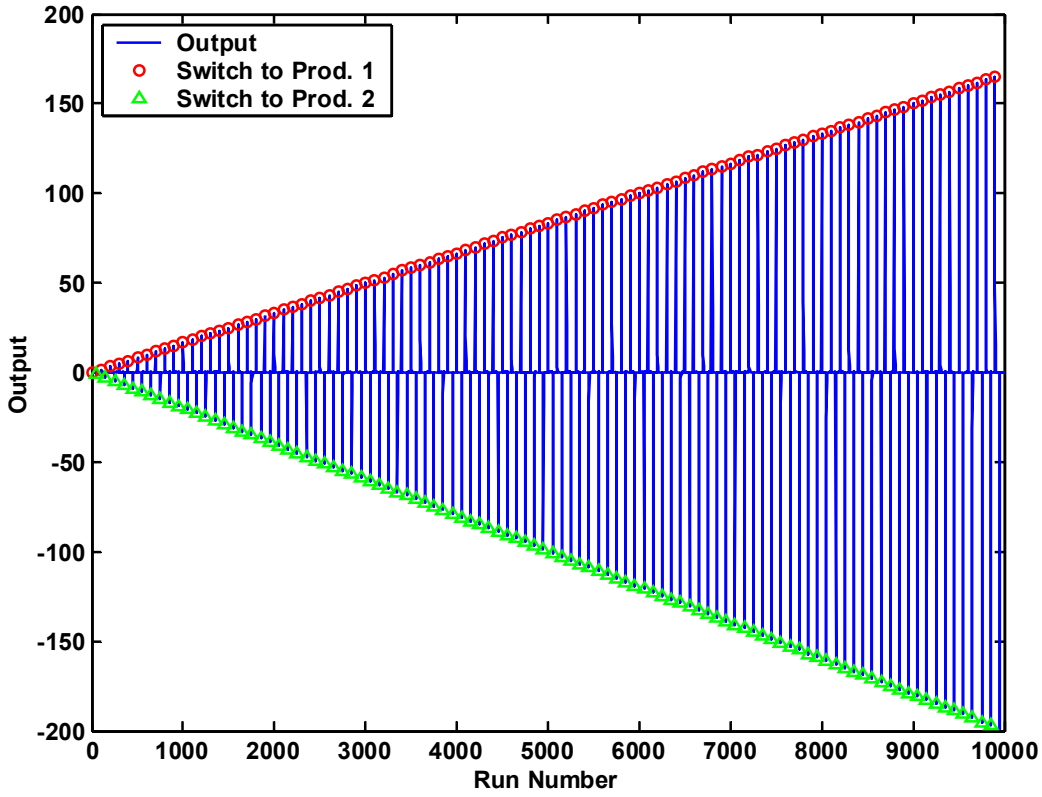


Figure C.7: Tool-based EWMA control with different plant-model mismatch for each product and drift added. The system becomes unstable as the output error at the switching points increases over time. Parameter values: $\lambda = 0.7, \delta = 0.1, \theta = 1, \xi_1 = 1, \xi_2 = 1.2, i = 100, j = 50.$

### C.3 Product-based Control

The second control approach investigated in [87] is called "product-based" control. This is essentially the same approach as threaded control in which each context (i.e., unique product-tool combination) is tracked individually. In the example system, which has only one tool, EWMA filtering is performed with respect to the last run of the

same product, rather than the previous run in which a different product may have been processed.

The noise estimate is the same as before, but the equation used to calculate it is product dependent. The filter equations for product 1 are

$$\tilde{\eta}_{it+n} = \begin{cases} \lambda\hat{\eta}_{it+n} + (1-\lambda)\tilde{\eta}_{i(t-1)+j} & n = 1 \\ \lambda\hat{\eta}_{it+n} + (1-\lambda)\tilde{\eta}_{it+n-1} & n = 2,\cdots,j \end{cases}, \tag{C.9}$$

and the deadbeat control law for product 1 is

$$X_{it+n} = \begin{cases} \dfrac{0 - a_1 - \tilde{\eta}_{(i-1)t+j}}{b_1} & n = 1 \\ \dfrac{0 - a_1 - \tilde{\eta}_{it+n-1}}{b_1} & n = 2,3,\cdots,j \end{cases}. \tag{C.10}$$

For the two product system, the filter equations for product 2 are

$$\tilde{\eta}_{it+n} = \begin{cases} \lambda\hat{\eta}_{it+n} + (1-\lambda)\tilde{\eta}_{it} & n = j+1 \\ \lambda\hat{\eta}_{it+n} + (1-\lambda)\tilde{\eta}_{it+n-1} & n = j+2,\cdots,i \end{cases}, \tag{C.11}$$

and the deadbeat control law for product 2 is

$$X_{it+n} = \begin{cases} \dfrac{0 - a_2 - \tilde{\eta}_{it}}{b_2} & n = j+1 \\ \dfrac{0 - a_2 - \tilde{\eta}_{it+n-1}}{b_2} & n = j+2,\cdots,i \end{cases}. \tag{C.12}$$

Because the estimates used in equations (C.9) and (C.10) are based only on results of product 1 runs, the quality of product 1 is no longer dependent on the measurements of the other product (or products) made during the break time between campaigns of product 1 (i.e., runs $it+n$ where $(j+1) \le n \le i$). Therefore, from the viewpoint of product 1, it is unimportant whether a single product or a variety of products is produced in the runs during the break [87].

The product based control is stable for systems with equal or unequal plant-model mismatches; this is true regardless of whether the disturbance includes either drift or integrating parts or both [87]. Figures C.8 and C.9 display the output of a system with unequal plant-model mismatches and a disturbance with both the drifting and integrating parts added. In the case of Figure C.8, the drifting portion of the disturbance is more dominant, but in Figure C.9, the integrating portion of the disturbance is more dominant. As with the tool-based system, the major deviations in the output for both cases occur at switching points, but in this case, the error from target is bounded. Also, the error at switching seems to be proportional to the break length for each product. Notice that the errors at switching points are larger for product 1 than for product 2 and that product 1 has the longer break length because it represents a smaller portion of the product mix. This is explored further in following section.

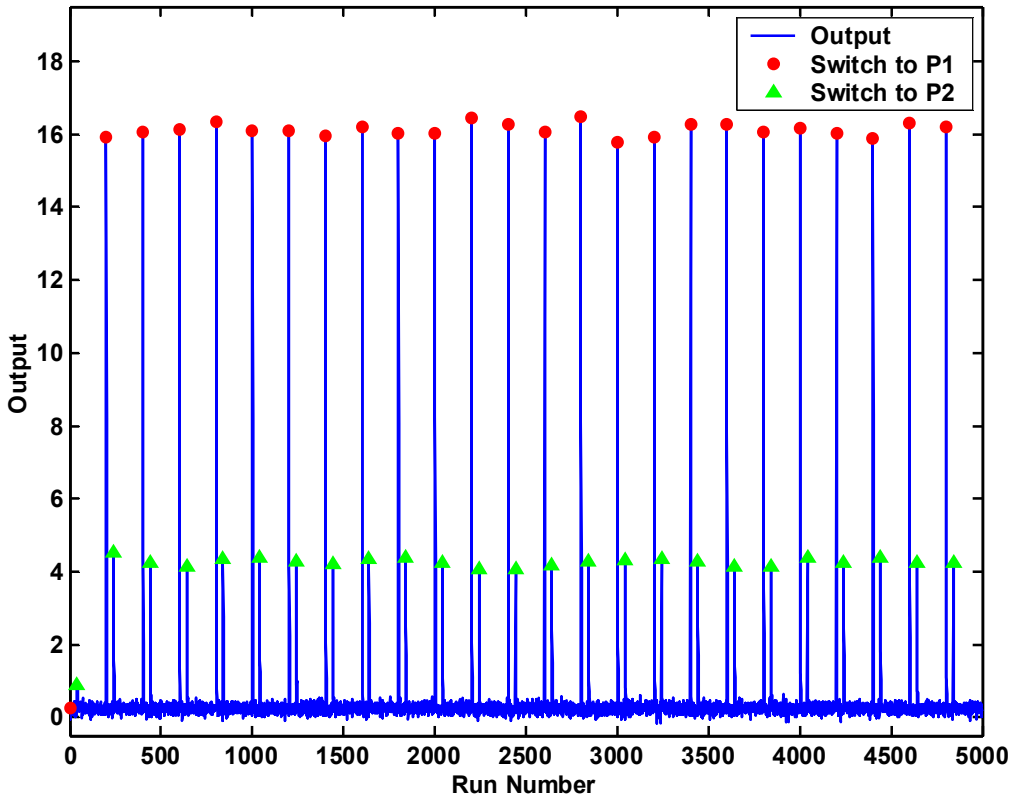


Figure C.8: Product-based EWMA with a drift-dominant disturbance. The process is stable and errors from the target are bounded. Parameter values: $\lambda = 0.5, \delta = 0.1, \theta = 0.9, \xi_1 = 1, \xi_2 = 0.8, i = 200, j = 40.$

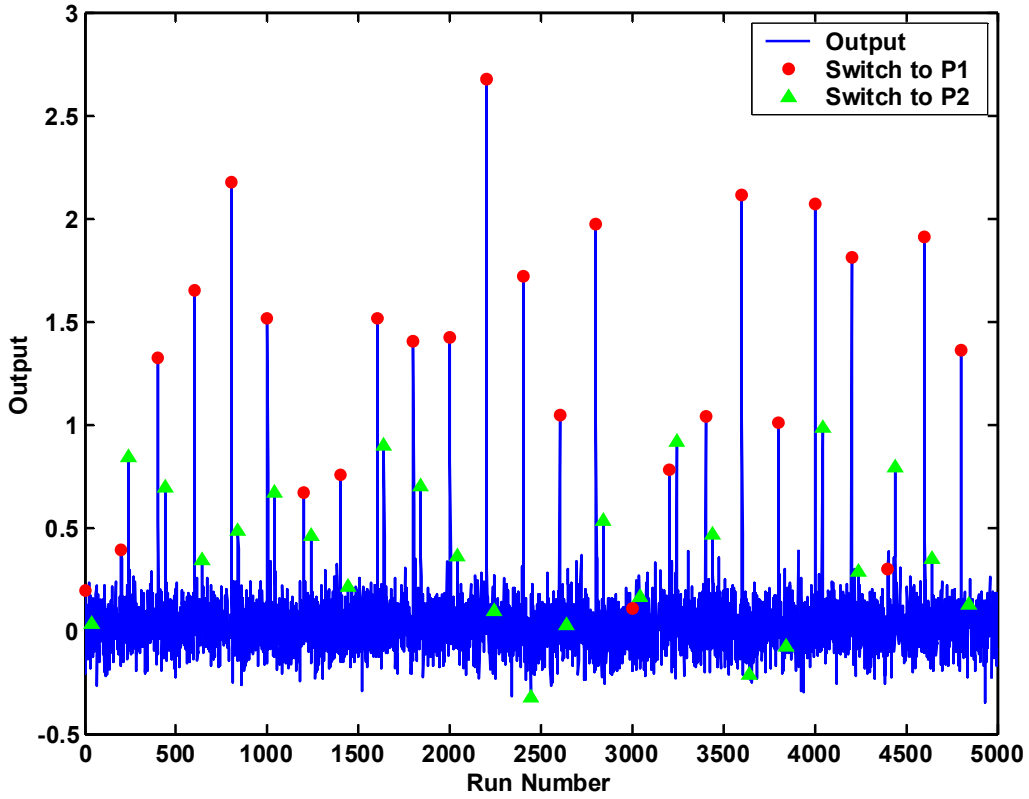


Figure C.9: Product-based EWMA with a disturbance in which the integrating portion is dominant. The process is stable and errors from the target are bounded. Parameter values: $\lambda = 0.5, \delta = 0.01, \theta = 0.5, \xi_1 = 1, \xi_2 = 0.8, i = 200, j = 40$.

In addition to stability under a broader range of conditions, product-based control has one other major advantage because the estimators for the two products can be tuned separately. In other words, the disturbance estimates for the two products are independent of each other (i.e., Equations (C.9) and (C.11) are independent of each other). This is especially important for products that are run less frequently (i.e., those which represent a smaller percentage of the product mix). For a less frequent product, lambda can be made closer to one so that the estimator can adjust more quickly to large changes in the disturbance that occur due to the longer break length between product

runs; this effect is especially apparent in systems with drift. Note the difference in results for product 1 between (a) and (b) of Figure C.10. In both cases, product 1 is a low volume product (5% of the product mix), but in case (b), lambda is closer to one than in case (a). The error in the first run of product 1 after a switch is high for both cases but the error for subsequent runs (i.e., runs 2–*j*) is much lower for case (b) because older information about the disturbance is discarded more quickly. Even though the estimator is more sensitive to noise, the output error due to noise sensitivity is small compared to the output error when the system is slow to compensate for large changes in the disturbance signal. This is made clear by comparing the mean output error of product 1 for the two cases when they are run for 50 cycles; scenario (a) has a mean squared error of 26.6, while scenario (b) has a mean squared error of 18.1. Using a larger value of lambda after a switch over is akin to the "rapid mode" approach employed by Sachs et al in [14] to compensate more quickly to shift disturbances.

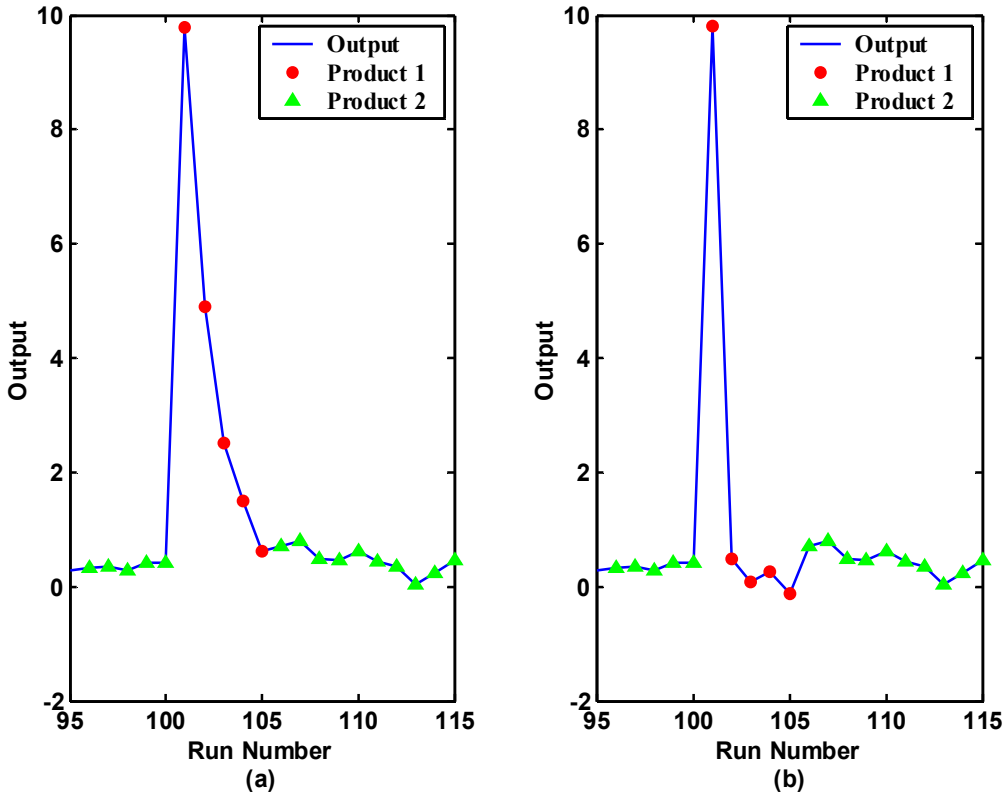


Figure C.10: Product-based EWMA control for an infrequent product (product 1 is only 5% of the product mix). In (a), $\lambda = 0.5$ and the system adjusts slowly after the change from product 2 to product 1 resulting in multiple large deviations from the target of zero. In (b), $\lambda = 0.95$ and the system adjusts rapidly to the large disturbance at the switch. Parameter values: $\lambda_{1(a)} = 0.5, \lambda_{1(b)} = 0.95, \lambda_2 = 0.3, \delta = 0.1, \theta = 1, \xi_1 = 1, \xi_2 = 1.2, i = 100, j = 5.$

## C.4 AMSE for Product Based Control During Long Runs Between Switches

In [87], the following formula for asymptotic mean squared error (AMSE) is derived:

$$AMSE(Y_n) = S_{pd}(n)\delta^2 + \frac{S_{ps}(n) - 2q_1^{2j-1}\theta}{1 - q_1^{2j}}\sigma^2 \qquad 1 \le n \le j, \qquad \text{(C.13)}$$

where the variables Spd and Sps are defined as

$$S_{pd}(n) \triangleq \left[ \frac{{q_1}^{n-1}(i-j)}{1-{q_1}^{j}} + \frac{1}{1-q_1} \right]^2, \tag{C.14}$$

and

$$S_{ps}(n) \triangleq 1 + \frac{(q_1-\theta)^2\left(1-{q_1}^{2j-2}\right)}{1-{q_1}^{2}} + {q_1}^{2j-2}\theta^2 + {q_1}^{2n-2}(1-\theta)^2(i-j), \tag{C.15}$$

respectively.

From plots of the system (e.g., Figure C.8), it is observed that the individual product outputs reach what appears to be a steady state between switches as long as the campaign length is long enough (i.e. large enough $j$ for the case of product 1). So if a point, $n$, is selected sufficiently far away from one but still less than or equal to $j$, then it lies in, what is effectively, a steady state region and behaves similarly to a run in a single product system that has reached a steady state.

For reasonable values for $\lambda$ and $\xi$, the term, $q_1 = 1-\lambda\xi_1$, goes rapidly toward zero when raised to higher powers (see, for example, Figures C.11 and C.12 for $q_1$ raised to the 10th and 20th powers, respectively). So for $n$ and $j$, sufficiently large (i.e., points in the steady state region), assume that terms of the form ${q_1}^{p}$, where $p = f(n,j)$, go to zero, and the following simplifications to the above equations are made:

$$\begin{aligned} S_{pd}(n) &= \left[ \frac{0(i-j)}{1-0} + \frac{1}{1-q_1} \right]^2, \\ S_{pd}(n) &= \left[ \frac{1}{1-q_1} \right]^2, \\ S_{pd}(n) &= \left( \frac{1}{\lambda\xi_1} \right)^2; \end{aligned} \tag{C.16}$$

$$S_{ps}(n) = 1 + \frac{(q_1 - \theta)^2 (1-0)}{1 - q_1^2} + 0\theta^2 + 0(1-\theta)^2 (i-j),$$

$$S_{ps}(n) = 1 + \frac{(q_1 - \theta)^2}{1 - q_1^2}, \tag{C.17}$$

$$S_{ps}(n) = \frac{1 + \theta^2 - 2(1 - \lambda\xi_1)\theta}{\lambda\xi(2 - \lambda\xi_1)};$$

$$AMSE(Y_n) = S_{pd}(n)\delta^2 + \frac{S_{ps}(n) - 0\theta}{1-0}\sigma^2,$$

$$AMSE(Y_n) = \left(\frac{1}{\lambda\xi_1}\right)^2 \delta^2 + \frac{\frac{1 + \theta^2 - 2(1 - \lambda\xi_1)\theta}{\lambda\xi(2 - \lambda\xi_1)} - 0\theta}{1-0}\sigma^2, \tag{C.18}$$

$$AMSE(Y_n) = \left(\frac{\delta}{\lambda\xi_1}\right)^2 + \frac{1 + \theta^2 - 2(1 - \lambda\xi_1)\theta}{\lambda\xi_1(2 - \lambda\xi_1)}\sigma^2.$$

Del Castillo [120] derives the following equation for $AMSE(Y_n)$ for a single product system with an IMA(1,1) disturbance with drift as:

$$AMSE(Y_n) = \left(\frac{\delta}{\lambda\xi}\right)^2 + \frac{1 + \theta^2 - 2(1 - \lambda\xi)\theta}{\lambda\xi(2 - \lambda\xi)}\sigma^2, \tag{C.19}$$

which is identical to the final equation derived above in Equation (C.18). Thus it is shown that products in a two product system which are run sufficiently long enough after a switching point take on the characteristics of those in a single product system.

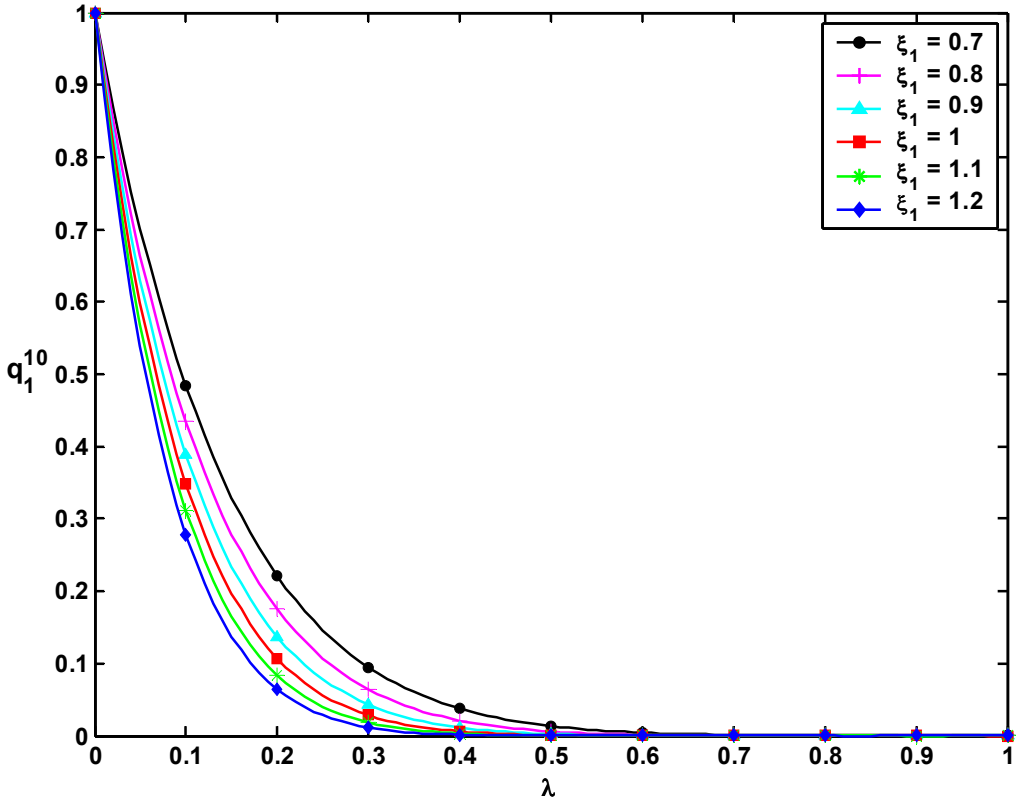


Figure C.11: Values of $q_1$ to the 10$^{th}$ power for various values of $\lambda$ and $\xi_1$.

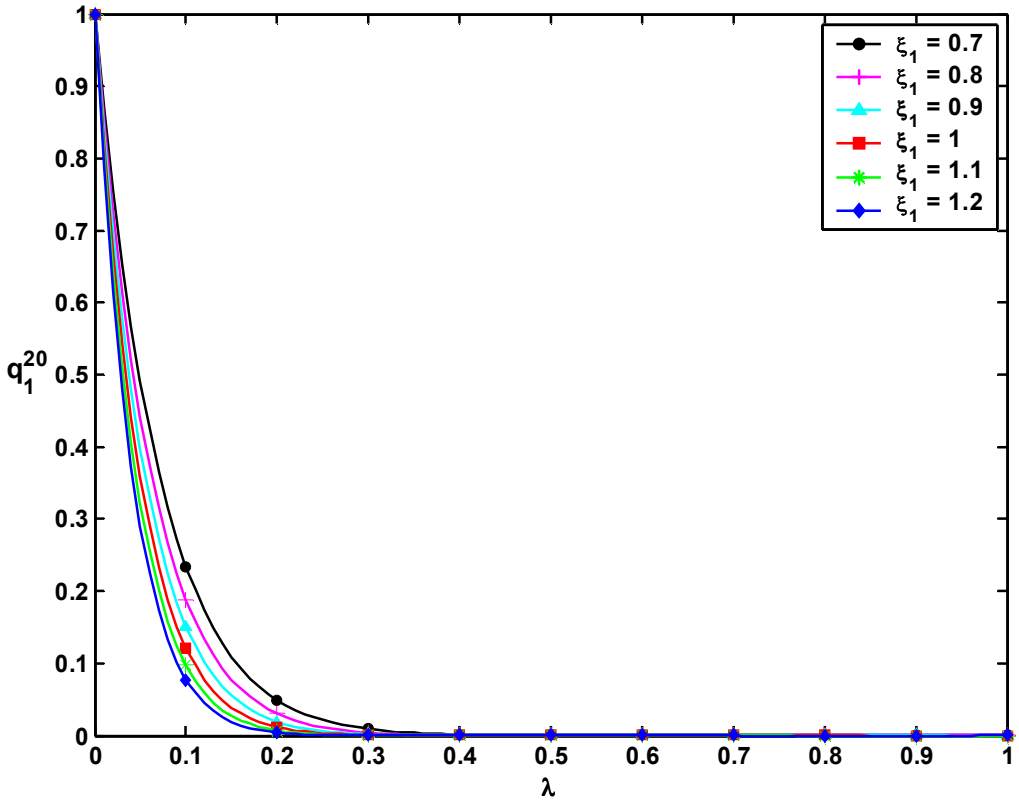


Figure C.12: Values of $q_1$ to the 20th power for various values of $\lambda$ and $\xi_1$.

## C.5 AMSE for Product Based Control at Switching Points in Drift Dominant Systems

At the point when a drift dominant process switches from product 2 back to product 1 at the beginning of a new cycle, the run index is equal to one (i.e., $n=1$) and the equations for the calculation of AMSE can be simplified as follows (for the drift dominant process, theta is assumed equal to one):

$$S_{pd}(1)=\left[\frac{(i-j)}{1-q_1^{\,j}}+\frac{1}{1-q_1}\right]^2, \tag{C.20}$$

$$S_{ps}(1)=1+\frac{(q_1-1)^2\left(1-q_1^{\,2j-2}\right)}{1-q_1^{\,2}}+q_1^{\,2j-2}, \tag{C.21}$$

$$AMSE(Y_1) = S_{pd}(1)\delta^2 + \frac{S_{ps}(1) - 2q_1^{2j-1}}{1 - q_1^{2j}}\sigma^2. \quad \text{(C.22)}$$

For processes with large enough $j$, many $q_1$ terms go to zero (as demonstrated in the previous section) and Equation (C.20) can be simplified further, as follows:

$$S_{pd}(1) = \left[ (i - j) + \frac{1}{1 - q_1} \right]^2, \quad \text{(C.23)}$$

$$S_{ps}(1) = 1 + \frac{(q_1 - 1)^2}{1 - q_1^2} = \frac{2}{1 + q_1}, \quad \text{(C.24)}$$

$$AMSE(Y_1) = \left[ (i - j) + \frac{1}{1 - q_1} \right]^2 \delta^2 + \left[ \frac{2}{1 + q_1} \right] \sigma^2. \quad \text{(C.25)}$$

For the above case, the AMSE at switching for product 1 becomes a strong function of $(i - j)^2$ (i.e., the square of the break length between campaigns of product 1). Using a pair of simulations for a drift dominant system where the break length of product 1 is doubled from the first simulation to the second, the above effect is demonstrated. In both cases, the product mix is 20 percent product one, but in the second simulation, $i$ and $j$ are both doubled from their values in the first simulation. It is clear from a visual inspection of Figure C.13 that the error from target for product 1 at switching points approximately doubles from plot (a) to plot (b). Calculations of the AMSE at switching points of product 1 from both the simulation data and the derived equation show that it approximately quadruples (i.e., it is a strong function of $(i - j)^2$).

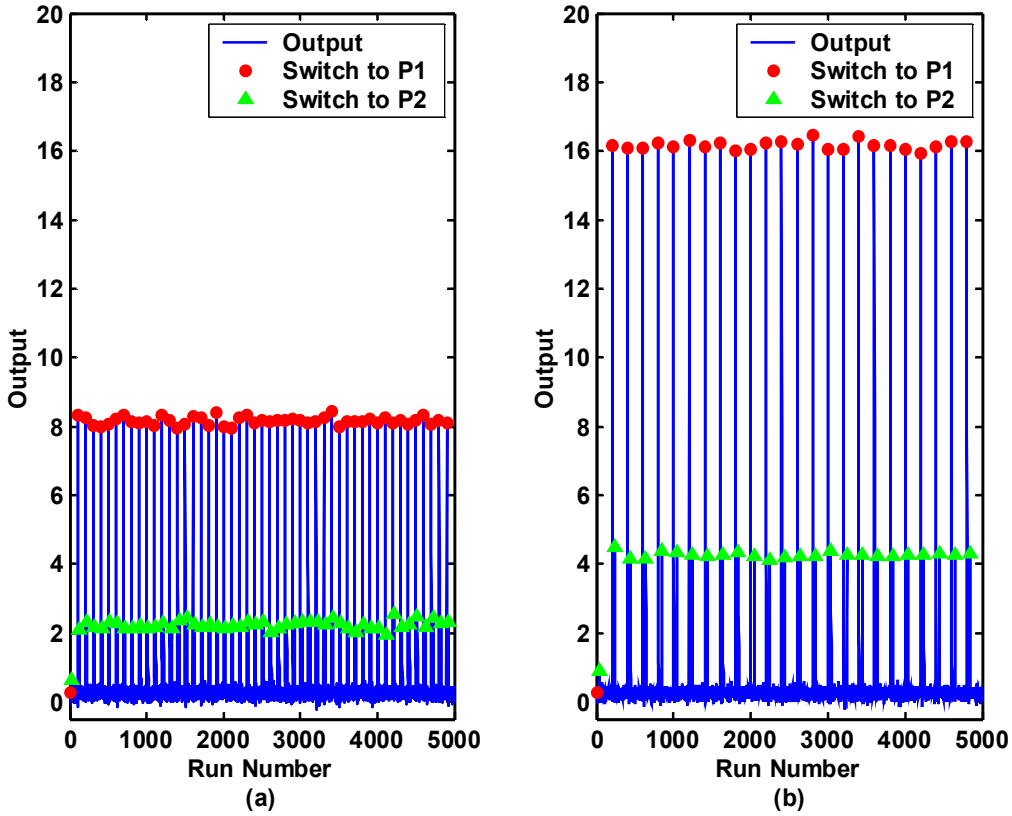


Figure C13: Output response for product based EWMA. (a) $i = 100$ and $j = 20$; the switch points for product 1 have a mean error of 8.16 and an AMSE of 66.6. (b) $i = 200$ and $j = 40$; the switch points for product 1 have a mean error of 16.2 and an AMSE of 261.

## APPENDIX D: CREATING A REDUCED MEASUREMENT NOISE COVARIANCE MATIX

### D.1 Pseudo-code

The measurement noise covariance matrix, R, is square by definition and size $p \times p$, where p is the length of the output vector. In cases where not all outputs are measured, it is necessary to create a "reduced" version of the output equations which includes only the rows corresponding to the measured outputs. Therefore, it is necessary to create a reduced version of the covariance matrix which is still square but contains only the information related to the measured outputs.

1. (Define reduction vector, $r$) The reduction vector, $r$, has size $p_m \times 1$, where $p_m$ is the number of measured outputs, and contains the row numbers corresponding to the measured outputs. For example, if outputs 1 and 3 are measured, then $r = \begin{bmatrix} 1 & 3 \end{bmatrix}^T$.

2. (Given $R$ and $r$, create the reduced version of $R$, $R_r$)

$$\begin{aligned} &\text{For } i = 1 : \text{length}(r) \\ &\text{For j} = 1 : \text{length}(r) \\ &R_r(i,j) = R(r(i), r(j)); \\ &\text{End} \\ &\text{End} \end{aligned} \tag{D.1}$$

In MatLab, pseudo-code (D.1) can be implemented simply as

$$R_r = R(r, r) \tag{D.2}$$

### D.2 A Three Tool, Two Product Example

For this example, there are six contexts and three qualification runs and therefore, nine outputs. The 9x9 measurement noise covariance matrix is

$$R = \begin{bmatrix} \sigma_{11}^2 & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} & \theta_{16} & \theta_{17} & \theta_{18} & \theta_{19} \\ \theta_{21} & \sigma_{22}^2 & \theta_{23} & \theta_{24} & \theta_{25} & \theta_{26} & \theta_{27} & \theta_{28} & \theta_{29} \\ \theta_{31} & \theta_{32} & \sigma_{33}^2 & \theta_{34} & \theta_{35} & \theta_{36} & \theta_{37} & \theta_{38} & \theta_{39} \\ \theta_{41} & \theta_{42} & \theta_{43} & \sigma_{44}^2 & \theta_{45} & \theta_{46} & \theta_{47} & \theta_{48} & \theta_{49} \\ \theta_{51} & \theta_{52} & \theta_{53} & \theta_{54} & \sigma_{55}^2 & \theta_{56} & \theta_{57} & \theta_{58} & \theta_{59} \\ \theta_{61} & \theta_{62} & \theta_{63} & \theta_{64} & \theta_{65} & \sigma_{66}^2 & \theta_{67} & \theta_{68} & \theta_{69} \\ \theta_{71} & \theta_{72} & \theta_{73} & \theta_{74} & \theta_{75} & \theta_{76} & \sigma_{77}^2 & \theta_{78} & \theta_{79} \\ \theta_{81} & \theta_{82} & \theta_{83} & \theta_{84} & \theta_{85} & \theta_{86} & \theta_{87} & \sigma_{88}^2 & \theta_{89} \\ \theta_{91} & \theta_{92} & \theta_{93} & \theta_{94} & \theta_{95} & \theta_{96} & \theta_{97} & \theta_{98} & \sigma_{99}^2 \end{bmatrix}. \quad \text{(D.3)}$$

In this example, assume all three tools run product one at the same time; therefore,

$$r = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}^T \quad \text{(D.4)}$$

and

$$R_r = \begin{bmatrix} \sigma_{11}^2 & \theta_{13} & \theta_{15} \\ \theta_{31} & \sigma_{33}^2 & \theta_{35} \\ \theta_{51} & \theta_{53} & \sigma_{55}^2 \end{bmatrix}. \quad \text{(D.5)}$$

A final note: Usually, R is diagonal so that all cross covariance terms, θij, are zero; in this case, the pseudo-code given above can be greatly simplified and therefore, made much more efficient.

## APPENDIX E: ONE-TOOL, ONE-PRODUCT EXPERIMENT FROM PASADYN [41, 54]

In [41] and [54], the author uses a system with a single tool and single product to demonstrate the effect of run ordering on the estimate error variance of the various states in the system. The state and output equations for the system in open-loop form (i.e., no inputs) are

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{k+1} = \begin{bmatrix} x_{t_1} \\ x_{p_1} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t_1} \\ x_{p_1} \end{bmatrix}_k + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_k \tag{E.1}$$

and

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_k = \begin{bmatrix} y_{t_1} \\ y_{11} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_{t_1} \\ x_{p_1} \end{bmatrix}_k + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_k, \tag{E.2}$$

respectively.

The first output, $y_{t_1}$, is a qual run for the tool and the second output, $y_{11}$, is production run of product 1 on tool 1. Three simulations, each with a different processing order, are run on the same set of 50 outputs (25 runs each of $y_{t_1}$ and $y_{11}$). In the experiments, a set of runs in which the system alternates between the two outputs is compared to a sequence of runs in which a long string of one output is followed by an equally long string of the other output. In other words, an output sequence of the form [2, 1, 2, 1,…, 2, 1] is compared to two output sequences that have the form [1, 1,…, 1, 2, 2,…, 2] and [2, 2,…, 2, 1, 1,…, 1]. Plots of the estimate error variance of the two system states for each of the three simulations are shown in the following figures.

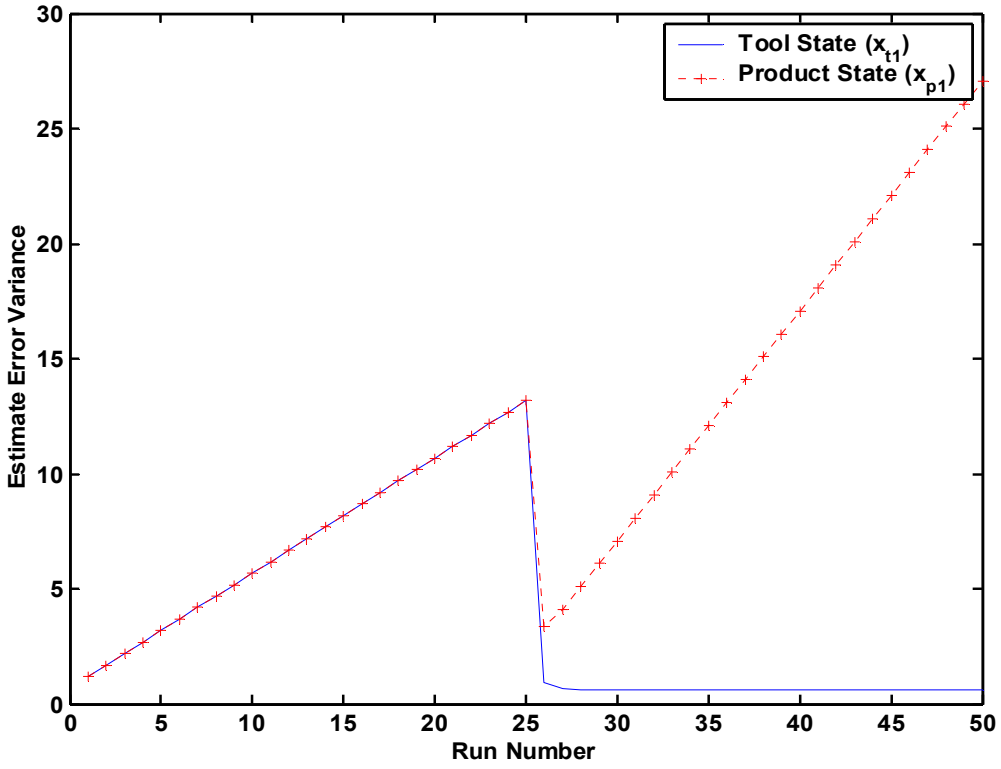


Figure E.1: All 25 production outputs, $y_2$, are run first and are then followed by the 25 qualification runs, $y_1$.

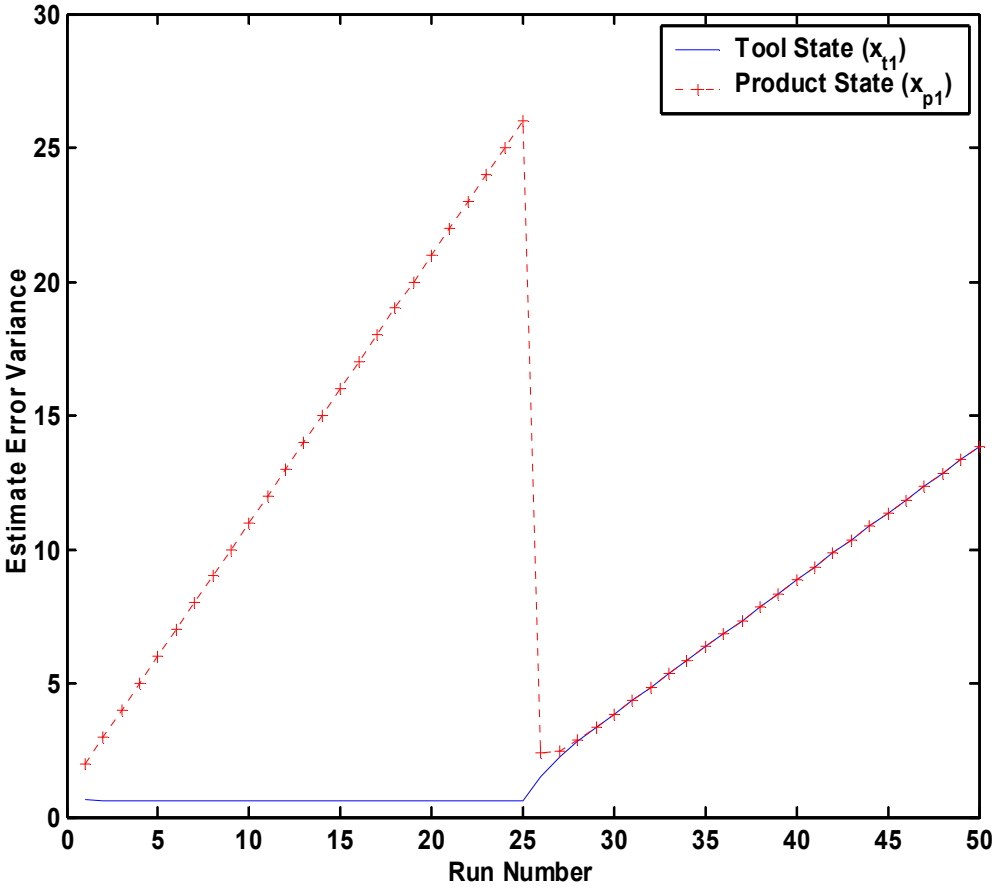


Figure E.2: All 25 qualification runs, $y_1$, are run first and are then followed by the 25 production outputs, $y_2$.

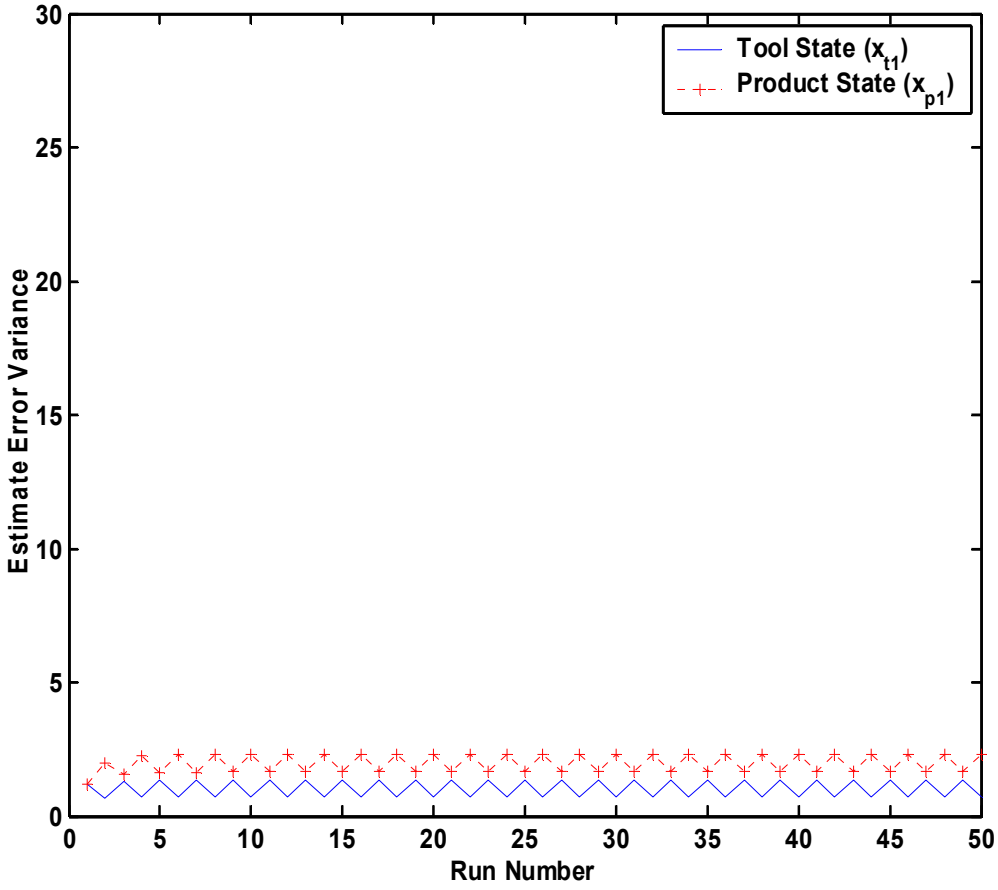


Figure E.3: The system switches back and forth between single runs of each output.

It is clear from the first two figures that when $y_1$ is run for a long stretch, there is no information available for the product state and its variance rises quickly. On the other hand, when $y_2$ is run for continuously for many runs, the measurement is a combination of the two states and there is no way to determine the unique contributions from each state, and variances for both states rise together. Finally, Figure E.3 demonstrates that alternating the two measurements allows the variance of both states to remain low.

## APPENDIX F: BRIEF REVIEW OF QUEUING TERMINOLOGY AND NOTATION

### F.1 Kendall-Lee Notation [100]

The Kendall-Lee notation for queuing systems is the standard manner in which queuing systems are described. There are six attributes which each have their own set of abbreviations depending upon the type of system being studied. The applicable abbreviations are expressed in a standard order separated by forward slashes as follows: 1/2/3/4/5/6.

Characteristic 1 indicates the type of arrival process. That is, it gives the distribution used to describe the manner in which products arrive at the queue in terms of their interarrival times (i.e., the amount of time that passes between product arrivals). Four abbreviations are used for this characteristic:

- M: interarrival times are independent and identically distributed (iid) random variables with an exponential distribution.
- D: iid and deterministic; that is, the variance is of the interarrival times is zero so they are constant.
- Ek: iid with an Erlang distribution having shape parameter k. The Erlang distribution has two parameters that can be changed to give many different shapes. For certain values of the parameters, the Erlang distribution can be made to take on the shape of the exponential, normal, or deterministic distributions.
- GI: iid with some general distribution (e.g., normal, uniform, etc.)

Characteristic 2 indicates the service process. That is, it gives the distribution used to describe the service times (i.e., the amount of time it takes a tool to process a product). This characteristic uses the same set of abbreviations as the previous one.

Characteristic 3 denotes the number of parallel servers in the system. In general, if there is more than one server in the system, they are all assumed to be equivalent (i.e., homogenous).

Characteristic 4 tells what type queue discipline is used in the system. The queue discipline specifies how the next customer is selected from the queue when server becomes available. The four most common abbreviations used are:

- FCFS: first come, fist served, also called first in, first out (FIFO). The earliest arriving customer (the customer in queue the longest period of time) is selected.
- LCFS: last come, first served, also called last in, first out (LIFO). The latest arriving customer (the customer in queue the shortest period of time) is selected.
- SIRO: service in random order. The next customer is randomly selected from the queue.
- GD: some general discipline is used.

Characteristic 5 gives the maximum number of customers allowed in the system at any given time. This maximum includes all customers in queue and those in service. Often the queue is considered to be sufficiently large enough to handle all unserviced customers who might arrive to the system; in this case, it is considered infinite.

Characteristic 6 dictates the size of the population outside of the system from which customers are obtained. This pool of potential customers is usually considered infinite if it is not very close in value to the number of servers.

It is important to note that many of the models studied in the literature consider the last three characteristics to be GD/∞/∞. In this case, the model will be described using only the first three characteristics and the last three are not shown but assumed to be GD/∞/∞.

## F.2 The Exponential and Poisson Distributions

The exponential distribution is often used to model interarrival times in queuing systems because it is characterized by the no memory property. The no memory property implies the current interarrival time is independent of the past interarrival times; that is, the time until the next arrival is not dependent on how long it has been since the last arrival. The probability density function for the exponential distribution of a continuous random variable is

$$f(x) = \begin{cases} \frac{1}{\alpha} e^{-x/\alpha} & \text{for } x > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{F.1}$$

where the parameter α is the both the mean and standard deviation of the distribution [101].

The exponential distribution is closely related to the Poisson distribution. Where as the exponential distribution is used to describe the waiting time between random events, the Poisson distribution can be used to model the rate at which the same events occur. The Poisson distribution for a discrete random variable has the following probability function:

$$f(y) = \begin{cases} \frac{e^{-\lambda}\lambda^{y}}{y!} & \text{for } x \in \mathbb{Z}^{+} \\ 0 & \text{otherwise} \end{cases}, \tag{F.2}$$

where $\lambda$ is both the mean and variance of the distribution [101]. Practically, $\lambda$ represents the rate of occurrence in a given period of time and the arrivals are said to be Poisson with rate $\lambda$ (denoted Poi(λ)).

In terms of queuing systems, $\lambda$ is considered to be the arrival rate. It is proven that if the number of arrivals in an interarrival time of length $x$ follows a Poisson distribution with parameter $\lambda x$, then the interarrival times follow an exponential

distribution with parameter $\alpha = 1/\lambda$ [100]. By using the Poisson distribution with occurrence rate multiplied by the interarrival time produces the probability that the number of arrivals in *x* is equal to *y*, that is,

$$P(n_a = y) = \frac{e^{-x\lambda}(x\lambda)^y}{y!},$$

where $n_a$ is the number of arrivals in the time period *x*. This leads to the exponential distribution for the interarrival times with parameter $\alpha = 1/\lambda$ as follows:

$$f(x) = \lambda e^{-x\lambda}. \tag{F.3}$$

In this case, the mean interarrival time will be $1/\lambda$ [100].

## APPENDIX G: DISCRETE EVENT SIMULATION CODE AND VERIFICATION TESTING

### G.1 Simulation Algorithm Pseudo-code

For the discrete event simulations, only the time stamps at which events occur are important. Therefore, a list of events with their corresponding time stamps are first generated. To check the accuracy of the simulations, they are constructed so that their long-run mean results are easily comparable to equations derived from queuing theory.

1. (Create random sequence of Poisson arrivals) In order to apply queuing theory, interarrival times for products need to be exponentially distributed, which leads to Poisson distributed arrival times. Given a mean arrival rate, $\lambda$, of products to the system, a sequence of exponentially distributed product interarrival times with mean, $m = 1/\lambda$, is constructed. These interarrival times can then be used to construct a series of Poisson distributed arrivals with mean $\lambda$.

   a. Generate a sequence of random numbers, $R$, from a uniform distribution on the interval (0,1) on length $n$:

   $$R = rand(n), \tag{G.1}$$

   b. Convert $R$ to a random sequence from an exponential distribution to get the desired sequence of interarrival times, $I$:

   $$I = -m\left(\ln\left(R\right)\right). \tag{G.2}$$

   c. (Calculate the sequence of arrival times) The arrival time of the $k^{th}$ product is generated by summing the 1st through $k^{th}$ interarrival times; call the sequence of arrival times $A$:

   $$A_k = \sum_{i=1}^{k} I_i. \tag{G.3}$$

2. (Calculate the start and completion times of the products) Use the set of $n$ products and their associated arrival times and processing times, $s$, to calculate

their start times, *S*, and completion times, *C*. Here, a one tool system and FIFO queue discipline are assumed.

a. (Initialize start times) Initial start time is set equal to the first arrival time:

$$S_1 = A_1. \tag{G.4}$$

b. (Initilize completion times) Initial completion time equals the sum of the first start and processing times:

$$C_1 = S_1 + s_1. \tag{G.5}$$

c. (Calculate remaining $S_k$ and $C_k$) Future start times are determined by comparing the arrival times of the products to the completion times of the previous products; if a product arrives after the completion of the product preceding it in the queue then it starts immediately and its start equals its arrival time; otherwise, start time equals the completion time of the preceding product. Completion times are calculated by checking the type of product being run and adding the appropriate processing time to the start time (i.e., using Equation (G.5) with *k* substituted for 1).

$$\begin{aligned} &\text{If } A_k > C_{k-1}; \\ &\quad \text{Then } S_k = A_k; \\ &\text{Else } A_k \le C_{k-1}; \\ &\quad \text{Then } S_k = C_{k-1}; \\ &\text{End.} \end{aligned} \tag{G.6}$$

3. (Calculate the long run statistics for simulations) The mean number of products present and mean residence time of products for both the queue and the system as a whole are calculated as well as the idle time for the system.

a. (Construct event timeline matrix) Arrival, start, and completion times are labeled with an identifier, or *EID*, where, *EID = 1* for arrivals, *EID = 2* for starts, and *EID = 3* for completions.

$$timeline = \begin{bmatrix} 1 & A_1 \\ 1 & A_2 \\ \vdots & \vdots \\ 1 & A_n \\ \hline 2 & S_1 \\ 2 & S_2 \\ \vdots & \vdots \\ 2 & S_n \\ \hline 3 & C_1 \\ 3 & C_2 \\ \vdots & \vdots \\ 3 & C_n \end{bmatrix}. \tag{G.7}$$

b. (Sort *timeline*) Sort according to the times in the second column. All event times are now in chronological order and each has an *EID*.

$$timeline = \text{sort}\left(timeline\right). \tag{G.8}$$

c. (Calculate times between events) Let $T$ equal the second column timeline and use it to make a new vector of time increments between events called $T_b$.

$$T_{b_i} = T_{i+1} - T_i \tag{G.9}$$

d. (Track the number of wafers in queue and in system) A running count of wafers in queue (*InQ*) and in system (*InS*) can then be kept for each time between events depending on the event occurring at the beginning of the time interval, $Tb_i$.

If $timeline_{1,i} = 1$;

Then $InQ_i = InQ_{i-1} + 1$ & $InS_i = InS_{i-1} + 1$;

ElseIf $timeline_{1,i} = 2$;

Then $InQ_i = InQ_{i-1} - 1$ & $InS_i = InS_{i-1}$; (G.10)

Else $timeline_{1,i} = 3$;

Then $InQ_i = InQ_{i-1}$ & $InS_i = InS_{i-1} - 1$;

End.

e. (Calculate average number of products in the system, $L$) The number of products in the system during each interval between events is multiplied by the length of the interval; all of these values are summed and divided by the total length of simulation, which is equal to the final completion time, $C_n$.

$$L = \frac{\sum_{i=1}^{n-1}\left(InS_i * T_{b_i}\right)}{C_n}. \quad \text{(G.11)}$$

f. (Calculate average number of products in the queue, $L_q$) The number of products in the queue during each interval between events is multiplied by the length of the interval; all of these values are summed and divided by the total length of simulation, which is equal to the final completion time, $C_n$.

$$L_q = \frac{\sum_{i=1}^{n-1}\left(InQ_i * T_{b_i}\right)}{C_n}. \quad \text{(G.12)}$$

g. (Calculate the mean time in the system for the products)

$$W = \text{mean}(C - A). \quad \text{(G.13)}$$

h. (Calculate the mean time in the queue for the products)

$$W_q = \text{mean}(S - A). \quad \text{(G.14)}$$

4. (Calculate the percentage of time the system is idle, $\pi_0$) Because unforced idle time is not allowed in the system (i.e., if the system is empty, any product

entering the queue immediately begins processing), then the system can only be idle when it is empty.

a. (Find index of 0 entries in set InS) Identify inter-event intervals where the system is empty.

$$index_{empty} = \text{find}\left(InS == 0\right). \tag{G.15}$$

b. (Calculate total idle time, *IT*) *IT* equals the sum of the initial idle time (i.e., before the first arrival) and all idle times when the system is empty.

$$IT = \sum_{i=index_{empty}} T_{b_i} + A_1. \tag{G.16}$$

c. (Calculate percent idle time, $\pi_0$)

$$\pi_0 = IT/C_n. \tag{G.17}$$

## G.2 Testing Simulation Against Theoretical Results

For a M/G/1 system with FIFO queuing discipline, the Pollaczek-Khinchin mean value formulae can be used to get the long-run mean statistics for the system [100]. These theoretical results are compared to the simulations of the same system to determine the accuracy of the simulations. The following two examples use a one-tool, two-product system to demonstrate this process. First the product mix, the service time for each product, and the system traffic intensity are specified, then the remaining values are calculated. In both examples the simulation results compare favorably with their theoretical counterparts.

### *G.2.1 Example with Identical Service Times for Each Product*

Set the percentage of product 1, $P_{P1}$, in the mix to be $\frac{1}{3}$: $P_{P1} = \frac{1}{3}$.
Therefore, the percentage of product 2, $P_{P2}$, in the mix is $\frac{2}{3}$: $P_{P2} = \frac{2}{3}$.
Set the service times for both products to be 2.
Product 1 service time: $s_1 = 2 = 1/\mu_1$.
Product 2 service time: $s_2 = 2 = 1/\mu_2$.
Total mean service time: $s_T = P_{P1}\left(s_1\right) + P_{P2}\left(s_2\right) = \frac{1}{3}\left(2\right) + \frac{2}{3}\left(2\right) = 2$

Product 1 deterministic service rate: $\mu_1 = 1/s_1 = \frac{1}{2}$.

Product 2 deterministic service rate: $\mu_2 = 1/s_2 = \frac{1}{2}$.

Total mean service rate: $\mu_T = 1/s_T = \frac{1}{2}$.

Set a total traffic intensity of 0.75: $\rho_T = \frac{3}{4}$.

Then the total mean interarrival rate is: $\lambda_T = \mu_T \rho_T = 0.375$

And the mean arrival rates for the two products are:

$\lambda_1 = P_{P1}(\lambda_T) = \frac{1}{3}(\frac{3}{8}) = \frac{1}{8}$ ;

$\lambda_2 = P_{P2}(\lambda_T) = \frac{2}{3}(\frac{3}{8}) = \frac{1}{4}$.

The mean interarrival times are:

$1/\lambda_T = 2.67$ ; $1/\lambda_1 = 8$ ; $1/\lambda_2 = 4$.

The variance of the service time is:

$\sigma^2 = P_{P1}(s_1 - s_T)^2 + P_{P2}(s_2 - s_T)^2 = \frac{1}{3}(2-2)^2 + \frac{2}{3}(2-2)^2 = 0$

Mean number of products in queue and the system according to Pollaczek/Khinchin:

$$L_q = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)} = \frac{(\frac{3}{4})^2 0^2 + (\frac{3}{4})^2}{2(1-\frac{3}{4})} = 1.125$$

$L = L_q + \rho = \frac{9}{8} + \frac{3}{4} = 1.875$

The steady state results for the simulation and P-K theorem match well.

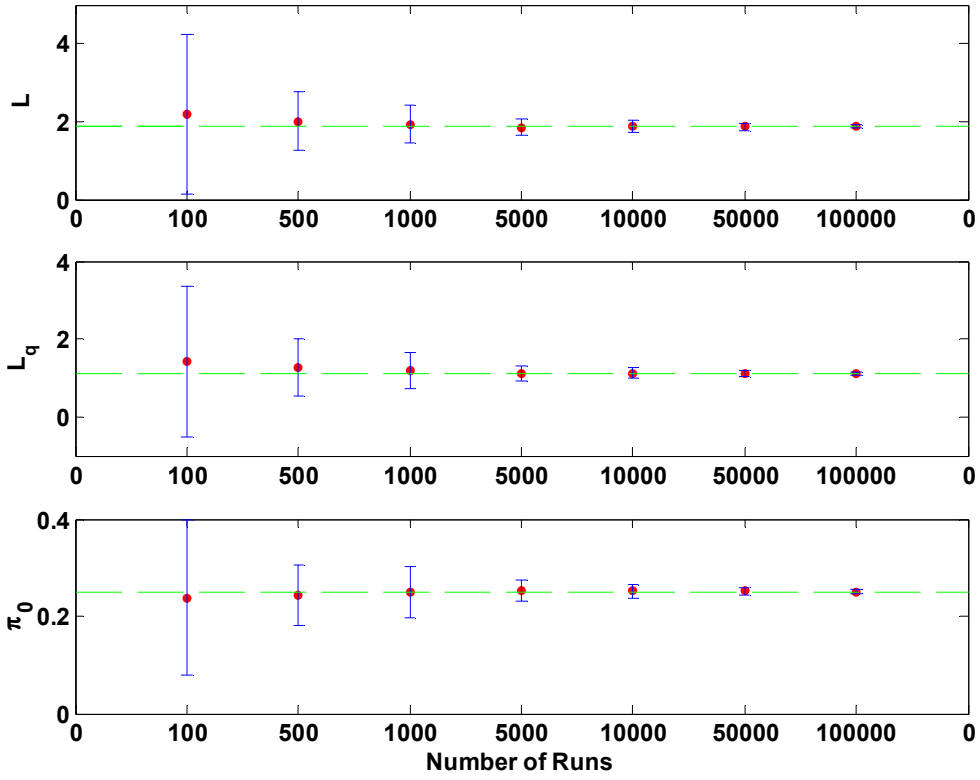


Figure G.1: Comparison of mean time in system (L), in queue (Lq), and idle time for discrete event simulations (21 simulation runs) and Pollaczek-Khinchin theory for a low volume system ( $\pi_0 = 0.75$ ) with one tool and two products that have equal processing times. Mean values (red dots), P-K values (green dashed line), 95% confidence interval (solid blue bars). Means approach predicted steady state values and confidence intervals shrink as the number of product runs increases.

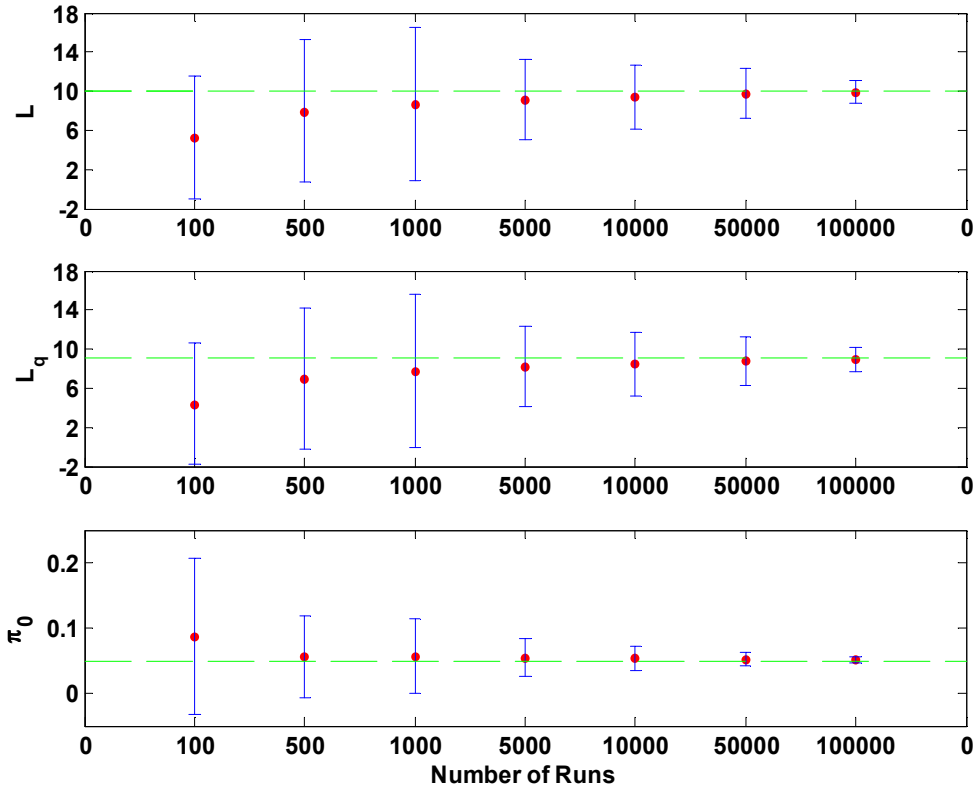


Figure G.2: Comparison of mean time in system (L), in queue (Lq), and idle time for discrete event simulations (21 simulation runs) and Pollaczek-Khinchin theory for a high volume system ( $\pi_0 = 0.95$ ) with one tool and two products that have equal processing times. Mean values (red dots), P-K values (green dashed line), 95% confidence interval (solid blue bars). Means approach predicted steady state values and, after initial transition where number of product runs is low, confidence intervals shrink as the number of product runs increases.

### *G.2.2 Example with Different Processing Times for Each Product*

Set the percentage of product 1, $P_{P1}$, in the mix to be $\frac{1}{3}$ : $P_{P1} = \frac{1}{3}$.

Therefore, the percentage of product 2, $P_{P2}$, in the mix is $\frac{2}{3}$ : $P_{P2} = \frac{2}{3}$.

Set the service times for both products.

Product 1 service time: $s_1 = 2 = 1/\mu_1$.

Product 2 service time: $s_2 = \frac{1}{2} = 1/\mu_2$.

Total mean service rate $= s_T = P_{P1}(s_1) + P_{P2}(s_2) = \frac{1}{3}(2) + \frac{2}{3}(\frac{1}{2}) = 1$

Product 1 deterministic service rate: $\mu_1 = 1/s_1 = \frac{1}{2}$.

Product 2 deterministic service rate: $\mu_2 = 1/s_2 = 2$.

Total mean service rate: $\mu_T = 1/s_T = 1$.

Set a total traffic intensity of 0.75: $\rho_T = \frac{3}{4}$.

Then the total mean interarrival rate is: $\lambda_T = \mu_T \rho_T = 0.75$

And the mean interarrival rates for the two products are:

$\lambda_1 = P_{P1}(\lambda_T) = \frac{1}{3}(\frac{3}{4}) = \frac{1}{4}$ ;

$\lambda_2 = P_{P2}(\lambda_T) = \frac{2}{3}(\frac{3}{4}) = \frac{1}{2}$.

The mean interarrival times are:

$1/\lambda_T = 1.33$ ; $1/\lambda_1 = 4$ ; $1/\lambda_2 = 2$.

The variance of the service time is:

$\sigma^2 = P_{P1}(s_1 - s_T)^2 + P_{P2}(s_2 - s_T)^2 = \frac{1}{3}(2-1)^2 + \frac{2}{3}(\frac{1}{2}-1)^2 = \frac{1}{3} + \frac{1}{6} = \frac{1}{2}$

Mean number of products in queue and the system according to Pollaczek/Khinchin:

$$L_q = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)} = \frac{(\frac{3}{4})^2 \frac{1}{2} + (\frac{3}{4})^2}{2(1-\frac{3}{4})} = 1.6875$$

$L = L_q + \rho = \frac{27}{16} + \frac{3}{4} = 2.4375$

As seen below, the results of the simulations match well.

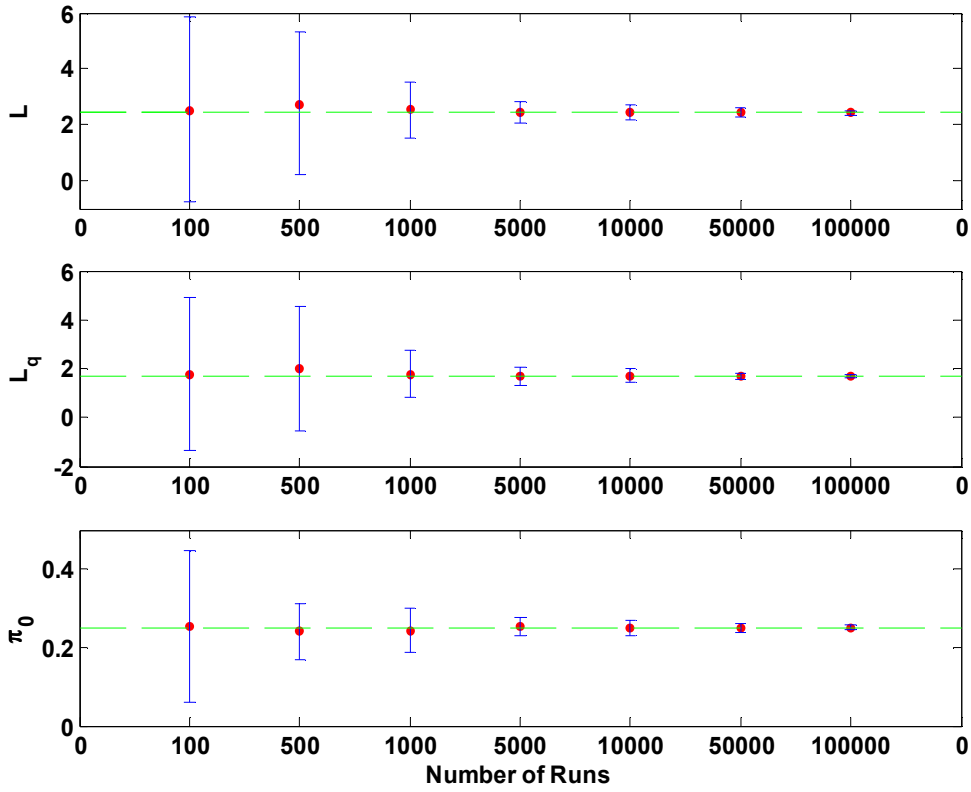


Figure G.3: Comparison of mean time in system (L), in queue (Lq), and idle time for discrete event simulations (21 simulation runs) and Pollaczek-Khinchin theory for a low volume system ( $\pi_0 = 0.75$ ) with one tool and two products that have different processing times. Mean values (red dots), P-K values (green dashed line), 95% confidence interval (solid blue bars). Means approach predicted steady state values and confidence intervals shrink as the number of product runs increases.

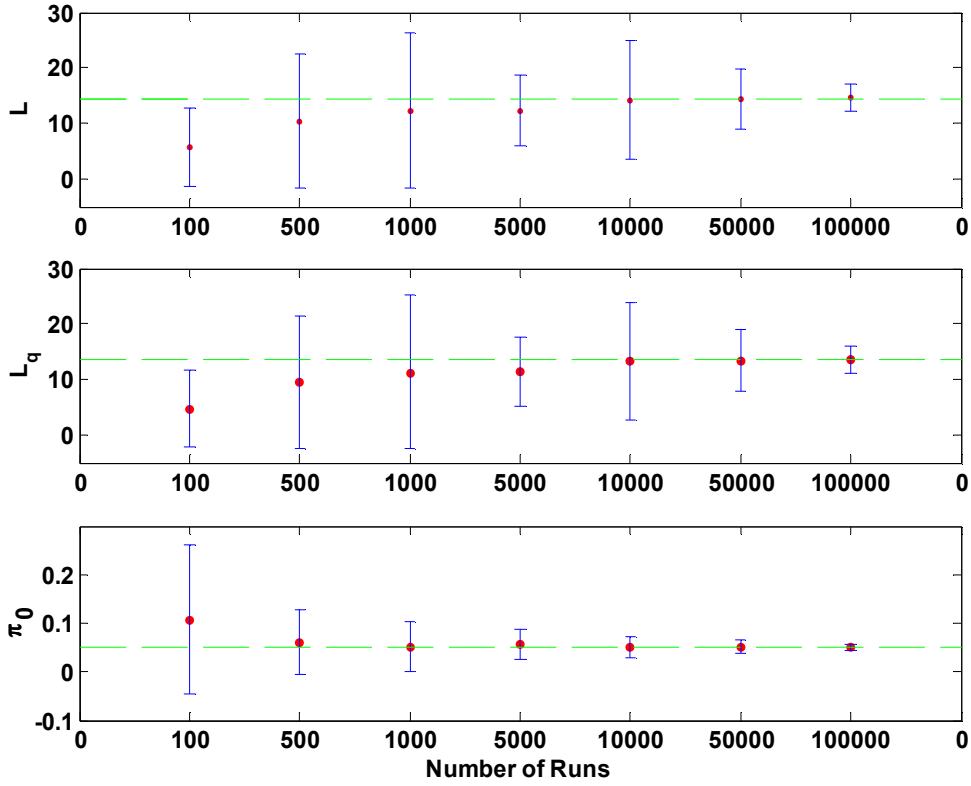


Figure G.4: Comparison of mean time in system (L), in queue (Lq), and idle time for discrete event simulations (21 simulation runs) and Pollaczek-Khinchin theory for a high volume system ( $\pi_0 = 0.95$ ) with one tool and two products that have different processing times. Mean values (red dots), P-K values (green dashed line), 95% confidence interval (solid blue bars). Means approach predicted steady state values and, after initial transition where number of product runs is low, confidence intervals shrink as the number of product runs increases.

# Bibliography


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

2. Plummer, J. D., Deal, M. D., and Griffin, P. B. *Silicon VLSI Technology*. Prentice Hall, New Jersey, 2000.

3. Mittler, M., and Schoemig, A. K. Comparison of dispatching rules for semiconductor manufacturing using large facility models. In P. A Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds. *Procceedings of the 1999 Winter Simulation Conference*, Phoenix, AZ, December, 1999, pp. 709–713.

4. Kumar, P. R. Re-entrant lines. *Queueing Systems*, 13: 87–110, 1993.

5. Uzsoy, R., Lee, C., and Martin-Vega, L. A. A review of production planning and scheduling models in the semiconductor industry part I: System characteristics, performance evaluations, and production planning. *IIE Transactions on Scheduling and Logistics*, 24: 47–60, 1992.

6. Fowler, J. W., Hogg, G. L., and Mason, S. J. Workload control in the semiconductor industry. *Production Planning and Control*, 13(7): 568–578, 2002.

7. Pfund, M. E., and Fowler, J. W. State of the Art Scheduling Survey Results, SRC Publication P003240, 25pp, 2001.

8. Bregenzer, B. C., Qin, S. J., Kutanoglu, E., and Hasenbein, J. J. Survey of scheduling practices in semiconductor manufacturing, SRC Publication P018781, 16pp, 2005.

9. Badgwell, T. A., Breedijk, T., Bushman, S. G., Butler, S. W., Chatterjee, S., Edgar, T. F., Toprac, A. J., and Trachtenberg, I. Modeling and control of microelectronics materials processing. *Computers and Chemical Engineering*, 19(1): 1–41, 1995.

10. Del Castillo, E. and Hurwitz, A. M. Run-to-run process control: Literature review and extensions. *Journal of Quality Technology*, 29(2): 184–196, 1997.

11. Adivikolanu, S. and Zafiriou, E. Extensions and performance/robustness tradeoffs of the EWMA run-to-run controller by using the internal model control structure. *IEEE Transactions on Electrical Packaging Manufacturing*, 23(1): 56–68, 2000.

12. Nishi, Y. and Doering, R., eds. *Handbook of Semiconductor Manufacturing Technology*. Marcel Decker, New York, 2000.

13. Campbell, W. J., Firth, S. K., Toprac, A. J., and Edgar, T. F. A comparison of run-to-run control algorithms. *Proceedings of the American Control Conference*, Anchorage, AK, May, 2002, pp. 2150–2155.

14. Sachs, E., Hu, A., and Ingolfsson, A. Run by run control: Combining SPC and feedback control. *IEEE Transactions on Semiconductor Manufacturing*, 4(1): 26–43, 1995.

15. Butler, S. W. and Stefani, J. A. Supervisory run-to-run control of polysilicon gate etch using *in situ* ellipsometry. *IEEE Transactions on Semiconductor Manufacturing*, 7(2): 193–201, 1994.

16. Chen, A. and Guo, R. Age-based double EWMA controller and its application to CMP processes. *IEEE Transactions on Semiconductor Manufacturing*, 14(1), 11–19, 2001.

17. Smith, T. H. and Boning, D. A self-tuning EWMA controller utilizing artificial neural network function approximation techniques. *IEEE Transactions on Components, Packaging, and Manufacturing Technology—Part C*, 20(2): 121–132, 1997.

18. Patel, N. S. and Jenkins, S. T. Adaptive optimization of run-to-run controllers: The EWMA example. *IEEE Transactions on Semiconductor Engineering*, 13(1): 97–107, 2000.

19. Braun, M. W., Jenkins, S. T., and Patel, N. S. A comparison of supervisory control algorithms for tool/process disturbance tracking. *Proceedings of the American Control Conference*, Denver, CO, June, 2003, pp. 2626–2631.

20. Muske, K. R. and Rawlings, J. B. Model predictive control with linear models. *A.I.Ch.E. Journal*, 39(2): 262–287, 1993.

21. Garcia, C. E., Prett, D. M., and Morari, M. Model predictive control: Theory and practice – A Survey. *Automatica*, 25(3): 335–348, 1989.

22. Morari, M. and Lee, J. H. Model predictive control: Past, present, and future. *Computers and Chemical Engineering*, 23: 667–682, 1999.

23. Qin, S. J. and Badgwell, T. A. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11: 733–764, 2003.

24. Rawlings J. B. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20(3): 38–52, 2000.

25. Campbell, W. J. *Model predictive run-to-run control of chemical mechanical planarization*. PhD thesis, The University of Texas at Austin, August, 1999.

26. Bode, C. A. *Run-to-run control of overlay and linewidth in semiconductor manufacturing*. PhD thesis, The University of Texas at Austin, May, 2001.

27. El Chemali, C. E. *Run-to-run control for wafer patterning in semiconductor manufacturing*. PhD thesis, The University of Michigan, 2003.

28. Vargas-Villamil, F. D. and Rivera, D. E. Multilayer optimization and scheduling using model predictive control: application to reentrant semiconductor manufacturing lines. *Computers in Chemical Engineering*, 24: 2009–2021, 2000.

29. Vargas-Villamil, F. D., Rivera, D. E., and Kempf, K. G. A hierarchical approach to production control of reentrant semiconductor manufacturing lines. *IEEE Transactions on Control Systems Technology*, 11(4): 578–587, 2003.

30. Wein, L. M. Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 1(3): 115–130, 1988.

31. Uzsoy, R., Lee, C., and Martin-Vega, L. A. Shop-floor control. *IIE Transactions*, 26(5): 44–55, 1994.

32. Dabbas, R. M. and Fowler, J. W. A new scheduling approach using combined dispatching criteria in wafer fabs. *IEEE Transactions on Semiconductor Manufacturing*, 16(3): 501–510, 2003.

33. Aytug, H., Lawley, M. A., McKay, K., Mohan, S., and Uzsoy, R. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operations Research*, 161: 86–110, 2005.

34. Herroelen, W. and Leus, R. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42(8): 1599–1620, 2004.

35. Raheja, A. S. and Subramanian, V. Reactive recovery of job shop schedules – A review. *International Journal of Advanced Manufacturing Technology*, 19: 756–763, 2002.

36. Vieira, G. E., Herrmann, J. W., and Lin, E. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, 6(1): 39–62, 2003.

37. Ruiz, D., Canton, J., Nougues, J. M., Espuna, A., and Puigjaner, L. On-line fault diagnosis system support for reactive scheduling in multipurpose batch chemical plants. *Computers in Chemical Engineering*, 25: 829–837, 2001.

38. Miller, M. L. Impact of multi-product and -process manufacturing on run-to-run control. In A. Ghanbari and A. Toprac, eds. *Process, Equipment, and Materials Control in Integrated Circuit Manufacturing III*, *Proceedings of SPIE*, 3213: 138–146, 1997.

39. Chen, Y., Su, A., Shiu, S., Yu, C., and Shen, S. Batch sequencing for run-to-run control: Application to chemical mechanical polishing. *Industrial and Engineering Chemistry Research*, 44: 4676–4686, 2005.

40. Patel, N. S. Lot Allocation and Process control and process control in semiconductor manufacturing – A dynamic game approach. *Proceedings of the 43rd IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, December $14^{th}$ –$17^{th}$, 2004, pp. 4243–4248.

41. Pasadyn, A. J. *Simultaneous control and identification for multiple product and process environments in semiconductor manufacturing*. PhD thesis, The University of Texas at Austin, December, 2001.

42. El Adl, M. K., Rodriguez, A. A., and Tsakalis, K. S. Hierarchical modeling and control of re-entrant semiconductor manufacturing facilities. *Proceedings of the 35th IEEE Conference on Decision and Control*, Kobe, Japan, December, 1996, pp. 1736–1742.

43. Tsakalis, K. S., Flores, J. J., and Rodriguez, A. A. Hierarchical modeling and control of re-entrant semiconductor fabrication lines: a mini-fab benchmark. *Proceedings of ETFA '97*, Los Angeles, CA, 1997, pp. 508–513.

44. Vargas-Villamil, F. D., Rivera, D. E., and Kempf, K. G. A hierarchical approach to production control of reentrant semiconductor manufacturing facilities. *IEEE Transactions on Control Systems Technology*, 11(4): 578–587, 2003.

45. Bode, C.A., Ko, B. S., and Edgar, T. F. Run-to-run control and performance monitoring of overlay in semiconductor manufacturing. *Control Engineering Practice*, 12(7), 893–900, 2004.

46. Maravelias, C. T., and Grossman, I. E. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers and Chemical Engineering*, 28, 1921–1949, 2004.

47. Sarin, S. C. and Prakish, D. Equal processing time bicriteria scheduling on parallel machines. *Journal of Combinatorial Optimization*, 8: 227–240, 2004.

48. Prakish, D. *Bi-criteria scheduling problems on Parallel Machines*. PhD thesis, Virginia Polytechnic and State University, May, 1997.

49. Joseph, A. K. Intel starts building USD 2.5 billion chip plant in China. Press Trust of India. September 8th, 2007. (http://www.reedexpo.com.cn/en/news/ShowInfo.aspx?ID=557)

50. Emami-Naeini, A., Ebert, J. L., Kosut, R. L., de Roover, D., and Ghosal, S. Model-based control for semiconductor and advanced material processing: An overview. *Proceedings of the American Control Conference*, Boston, MA, June 30th–July 2nd, 2004, pp. 3902–3909.

51. May, G. S., and Spanos, C. J. *Fundamentals of Semiconductor Manufacturing and Process Control*. John Wiley and Sons, Hoboken, New Jersey, 2006.

51. May, G. S., and Sze, S. *Fundamentals of Semiconductor Fabrication*. Wiley, New York, 2002.

52. Moyne, J., del Castillo, E., and Hurwitz, A. M., eds. *Run-to-run Control in Semiconductor Manufacturing*. CRC Press, New York, 2001.

53. Pasadyn, A. J., and Edgar, T. F. Observability and state estimation for multiple product control in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 18(4): 592–604, 2005.

54. Pasadyn, A. J., Lee, H., and Edgar, T. F. Scheduling semiconductor manufacturing processes to enhance system identification. *Journal of Process Control*. 18(10): 946–953, 2008.

55. Pasadyn, A. J., Toprac, A. J., and Edgar, T. F. Adaptive control of multiple product processes. In M. L. Miller, and K. A. Ashtiani, eds. *Process Control and Diagnostics*, *Proceedings of SPIE*, 4182: 22–30, 2007.

56. Edgar, T. F., Campbell, W. J., and Bode, C. Model-based control in microelectronics manufacturing. *Proceedings of the 38th IEEE Conference on Decision and Control*, Phoenix, Arizona, USA, December, 1999, pp. 4185–4191.

57. Pinedo, M. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, Upper Saddle River, New Jersey, 2002.

58. Kubiak, W., Lou, S. X. C., and Wang, Y.-M. Mean flow time minimization in re-entrant job shops with hub. Faculty of Management, University of Toronto, 1990.

59. Graves, S. C., Meal, H. C., Stefek, D., and Zeghmi, A. H. Scheduling of Re-Entrant Flow Shops. *Journal of Operations Management*, 3(4): 197–207, 1983.

60. Chao, Q. and Sivakumar, A. I. Simulation based cause and effect analysis in semiconductor wafer fabrication. *Journal of the Institution of Engineers, Singapore*, 44(4): 53–65, 2004.

61. Sreenivasan, S. V., Willson, C. G., and Resnick, D. J. Small print. *Mechanical Engineering Magazine*, April, 2004.

62. *Cost Effective IC Manufacturing 1998–1999*. Integrated Circuit Engineering Corporation, 1997. (http://smithsonianchips.si.edu/ice/cd/CEICM/TITLE.pdf)

63. Wolf, S. *Microchip Manufacturing*. Lattice Press, Sunset Beach, California, 2004.

64. Box, G. and Kramer, T. Statistical process monitoring and feedback adjustment—a discussion. *Technometrics*, 34(3): 251–267, 1992. (See also responses to discussion, pp. 268–285.)

65. Vander Wiel, S. A., Tucker, W. T., Faltin, F. W., and Doganaksoy, N. Algorithmic statistical process control: concepts and application. *Technometrics*, 34(3), 286–297, 1992.

66. Tucker, W. T., Faltin, F. W., and Vander Wiel, S. A. Algorithmic statistical process control: an elaboration. *Technometrics*, 35(4): 286–297, 1993.

67. Sachs, E., Guo, R., Ha, S., and Hu, A. Process control system for VLSI fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 4(2): 134–144, 1991.

68. Ingolfsson, A., and Sachs, E. Stability and sensitivity of an EWMA controller. *Journal of Quality Technology*, 25(4): 271–287, 1993.

69. Del Castillo, E. Some properties of EWMA feedback quality adjustment schemes for drifting processes. *Journal of Quality Technology*, 33(2): 153-166 2001.

70. Good, R. P. and Qin, S. J. On the stability of MIMO EWMA run-to-run controllers with metrology delay. *IEEE Transactions on Semiconductor Manufacturing*. 19(1): 78–86, 2006.

71. Tseng, S. T., Chou, R. J., and Lee, S. P. A study on a mulivaritate EWMA controller. *IIE Transactions*, 34: 541–549, 2002.

72. Del Castillo, E. and Rajagopal, R. A multivariate double EWMA process adjustment scheme for drifting processes. *IIE Transactions*, 34: 1055–1068, 2002.

73. Del Castillo, E. A multivariate self-tuning controller for run-to-run process control under shift and trend disturbances. *IIE Transactions*, 28(12): 1011–1021, 1996.

74. Hankinson, M., Vincent, T., Irani, K. B., and Khargonekar, P. P. Integrated real-time and run-to-run control of etch depth in reactive ion etching. *IEEE Transactions on Semiconductor Manufacturing*. 10(1): 121–130, 1997.

75. Del Castillo, E. Long run and transient analysis of a double EWMA feedback controller. *IIE Transactions*, 31(12): 1157-1169, 1999.

76. Tseng, S. T., Chou, R. J., and Lee, S. P. Statistical design of double EWMA controller. *Applied Statistical Models in Business and Industry*, 18: 313–322, 2002.

77. Good, R. P. and Qin, S. J. Stability of double EWMA run-to-run control with metrology delay. *Proceedings of the American Control Conference*, Anchorage, AK, May, 2002, pp. 2156–2161.

78. Del Castillo, E. and Yeh, J. An adaptive run-to-run optimizing controller for linear and nonlinear semiconductor processes. *IEEE Transactions on Semiconductor Manufacturing*. 11(2): 285–295, 1998.

79. Qin, S. J., Scheid, G. W., and Riley, T. J. Adaptive run to run control for intermittent batch operations. *Proceedings of the American Control Conference*, Anchorage, AK, May, 2002, pp. 2168–2173.

80. Fan, S. S., Jiang, B. C., Jen, C., and Wang, C. SISO run-to-run feedback controller using triple EWMA smoothing for semiconductor manufacturing processes. *Internation Journal of Production Research*. 40(13): 3093–3120, 2002.

81. Smith, T. H. and Boning, D. S. Artificial neural network exponentially weighted moving average controller for semiconductor processes. *Journal of Vacuum Science and Technology A*. 15(3): 1377–1384, 1997.

82. Wang, J., He, Q. P., Qin, S. J., Bode, C. A., and Purdy, M. A. Recursive least squares estimation for run-to-run control with metrology delay and its application to STI etch process. *IEEE Transactions on Semiconductor Manufacturing*. 18(2): 309–319, 2005.

83. Mullins, J. A., Campbell, W. J., and Bode, C. A. An evaluation of model predictive control in run to run processing in semiconductor manufacturing. In *Process, Equipment, and Materials Control in Integrated Circuit Manufacturing III*, A. Ghanbari and A. Toprac, eds., SPIE, 1997.

84. Muske, R. M. *Linear Model Predictive Control of Chemical Processes*. PhD thesis, The University of Texas at Austin, May, 1995.

85. Martinez, V. M. *Adaptive run-to-run control of overlay in semiconductor manufacturing*. PhD thesis, The University of Texas at Austin, May, 2002.

86. Bode, C. A., Wang, J., He, Q.P., and T.F. Edgar. Run-to-run control and state estimation in high-mix semiconductor manufacturing. *Annual Reviews in Control*. 31: 241–253, 2007.

87. Zheng, Y., Lin, Q., Wang, D. S., Jang, S., and Hui, Keung. Stability and performance analysis of mixed product run-to-run control. *Journal of Process Control*. 16(5): 431–443, 2006.

88. Firth, S. K. *Just-in-time adaptive disturbance estimation for run-to-run control in semiconductor processes*. PhD thesis, The University of Texas at Austin, December, 2002.

89. Åström, K. J. and Wittenmark, B. *Adaptive Control*. Addison-Wesley Publishing Company, Reading, MA, second edition, 1995.

90. Patel, N. S., Miller, G. A., Guinn, G., Sanchez, A. C., and Jenkins, S. T. Device dependent control of chemical-mechanical polishing of dielectric films. *IEEE Transactions on Semiconductor Manufacturing*. 13(3): 331–343, 2000.

91. Wu, M., Jang, S., and Wong, D. S. Mixed products run-to-run process control—an ANCOVA model based control approach. *Proceedings of IFAC Workshop on Advanced Process Control for Semiconductor Manufacturing*, Singapore, December 4th–5th, 2006, pp. 7–12.

92. Jang, S. A novel mixed product run-to-run control algorithm—dynamic ANCOVA approach. *AIChE Spring Meeting*, Houston, Texas, April 22nd–26th, 2007.

93. Ma, M., Chang, C., Wong, D.S., Jang, S. Identification of tool and product effects in a mixed product and parallel tool environment. *Journal of Process Control*, 19(4): 591–603, 2009.

94. Goodwin, G. C., Graebe, S. F., and Salgado, M. E. *Control System Design*. Prentice-Hall, Upper Saddle River, New Jersey, 2001.

95. Longhi, Sauro. Structural properties of multirate sampled-data systems. *IEEE Transactions on Automatic Control*. 39(3): 692–696, 1994.

96. Lewis, F. L. *Optimal Estimation with an Introduction to Stochastic Control Theory*. John Wiley and Sons, New York, 1986.

97. Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*. 82(Series D): 35–45, 1960.

98. Maybeck, P. S. *Optimal Models, Estimation, and Control, Vol. 1*. Academic Press, New York, 1979.

99. Gajic, Z. and Lelic, M. *Modern Control Systems Engineering*. Prentice Hall International, London, 1996.

100. Winston, W. L. *Operations Research: Applications and Algorithms*. Duxbury Press, Belmont, CA, 1994.

101. Vardeman, S. B. *Statistics for Engineering Problem Solving*. PWS Publishing Company, Boston, 1994.

102. Devroye, L. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

103. Martinez, W. L. and Martinez, A. R. *Computational Statistics Handbook with MATLAB*. CRC Press, New York, 2002.

104. Matveev, A. S. and Savkin, A. V. *Qualitative Theory of Hybrid Dynamical Systems*. Birkhäuser, Boston, 2000.

105. Sun, Z. and Ge, S. S. *Switched Linear Control Systems: Control and Design*. Springer-Verlag, London, 2005.

106. Liberzon, D. *Switching in Systems and Control*. Birkhäuser, Boston, 2003.

107. Salt, J. and Albertos, P. Model-based multirate controllers design. *IEEE Transactions on Control Systems Technology*. 13(6): 988–997, 2005.

108. Das, T. and Mukherjee, R. Optimally switched linear systems. *Automatica*. 44: 1437–1441, 2008.

109. Baglietto, M., Battistelli, G., and Scardovi, L. Active mode observability of switching linear systems. *Automatica*. 43: 1442–1449, 2007.

110. Alessandri, A., Baglietto, M., and Battistelli, G. Receding-horizon estimation for switching discrete-time linear systems. *IEEE Transactions on Automatic Control*. 50(11): 1736–1748, 2005.

111. Babaali, M. and Egerstedt, M. Nonpathological sampling of switched linear systems. *IEEE Transactions on Automatic Control*. 50(12): 2102–2105, 2005.

112. Sun, Z. and Ge, S. S. Analysis and synthesis of switched linear control systems. *Automatica*. 41: 181–195, 2005.

113. Lin, H. and Antsaklis, P. J. Stability and stabilizability of switched linear systems: a short survey of recent results. *Proceedings of the 2005 IEEE International Symposium on Intelligent Control*. Limassol, Cyprus, June 27th–29th, 2005, pp. 24–29.

114. Babaali, M. and Egerstedt, M. Pathwise observability and controllability are decidable. *Proceedings of the 42nd IEEE Conference on Decision and Control*. Maui, Hawaii, December, 2003, pp. 5771–5776.

115. Sun, Z. and Ge, S. S. Dynamic output feedback stabilization of a class of linear systems. *IEEE Transactions on Circuits and System—I: Fundamental Theory and Applications*. 50(8): 1111–1115, 2003.

116. Chen, Z. Local observability and its application to multiple measurement estimation. *IEEE Transactions on Industrial Electronics*. 38(6): 491–496, 1991.

117. Hong, S., Chun, H., Kwon, S., and Lee, M. H. Observability measures and their application to GPS/INS. *IEEE Transactions on Vehicular Technology*. 57(1): 97–105, 2008.

118. Golub, G. H. and Van Loan, C. F. *Matrix Computations*, 3rd ed. John Hopkins University Press, Baltimore, 1996.

119. Ham, F. M. and Brown, R. G. Observability, eigenvalues, and Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*. AES-19(2): 269–273, 1983.

120. Del Castillo, E. *Statistical Process Adjustment for Quality Control*. Wiley, New York, 2002.

121. Firth, S.K., Campbell, W.J., Toprac, A., Edgar, T.F. Just-in-time adaptive disturbance estimation for run-to-run control of semiconductor processes. *IEEE Transactions on Semiconductor Manufacturing*. 19(3): 298–315, 2006.

122. Hanish, C.K. Run-to-run state estimation in systems with unobservable states. *Proceedings of AEC/APC Symposium XVII*, Indian Wells, CA, 2005.

123. Wang, J., He Q.P., Edgar, T.F. State estimation in high-mix semiconductor manufacturing. *Journal of Process Control*. 19(3): 443–456, 2009.

124. Jobson, J.D. *Applied Multivariate Data Analysis, Vol. I: Regression and Experimental Design*. Springer-Verlag, New York, 1991.

125. Tasoulis, D. K., Adams N. M., and Hand, D. J. Selective fusion of out-of-sequence measurements. *Information Fusion***.** 11(2): pp. 183-191, 2010.

126. Larsen, T. D., Andersen, N. A., and Ravn, O. Incorporation of time delayed measurements in a discrete-time Kalman filter. In *Proceedings of the 37th IEEE Conference on Decision & Control*, Tampa, Florida, December, 1998, pp. 3972–3977.N. 127. Patel, S. Model regularization for high-mix control. *IEEE Transactions on Semiconductor Manufacturing*. 23(2): 151–158, 2010.

128. Marler, R. T., Arora, J. S. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*. 26: 369–395, 2004.

129. Chen, J-H, Kuo, T-W, Lee, A-C. Run-by-run process control of metal sputter deposition: combining time series and extended Kalman filter. *IEEE Transactions on Semiconductor Manufacturing*. 20(3): 278–285, 2007.

130. Sheats, J. R., Smith, B. W. *Microlithography: science and technologyg*, 2nd ed. CRC Press, Boca Raton, FL, 2007.

131. Prabhu, A. V., Edgar, T. F. A new state estimation method for high-mix semiconductor manufacturing processes. *Journal of Process Control*. 19(7): 1149–1161, 2009.

132. Good, R. P. and Schulze, U. An Algorithm for the Initialization of Thread States: Thread Reconstruction. *Proceedings of AEC/APC Symposium XVIII*, Westminster, CO, 2006.

133. Anderson, M., Hanish, C. K. An evaluation of the benefits of integrating run-to-run control with scheduling and dispatching systems. *IEEE Transactions on Semiconductor Manufacturing*. 20(4): 386–392, 2007.

134. An-Jhih Su, A., Yu, C., Ogunnaike, B. A. On the interaction between measurement strategy and control performance in semiconductor manufacturing. *Journal of Process Control*. 18(3): 266–276, 2008.

135. Lee, H. J. *Advanced process control and optimal sampling in semiconductor manufacturing*. PhD thesis, The University of Texas at Austin, August, 2008.

# Vita

Brent Bregenzer was born in Memphis, TN, on September 11th, 1976, to Dale and Carol Bregenzer. He attended Christian Brothers High School in Memphis, graduating in 1995, before moving on to the University of Clemson where he received a Bachelors of Science in Chemical Engineering in 2001. He then moved to Austin for graduate school at the University of Texas where he received a Masters of Science in Chemical Engineering in 2005 and a Ph.D. in same field in 2011. He has extensive work experience having completed six semesters of cooperative education in the chemical manufacturing industry as an undergrad and three semesters as an intern while in graduate school. The internships were in the fields of polymer processing and semiconductor manufacturing. He also completed two semesters as a teaching assistant while at UT and has participated in numerous conferences related to research in the fields of process control and semiconductor manufacturing.

Permanent address: 1510 W. North Loop Blvd, Apt 913, Austin, TX, 78756

This dissertation was typed by Brent Bregenzer.