Integrated Model-Based Run-to-Run Uniformity Control for Epitaxial Silicon Deposition

by

Aaron Elwood Gower-Hall

B.S., Worcester Polytechnic Institute, 1994
S.M., Massachusetts Institute of Technology, 1996

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2001

Author .....
Electrical Engineering and Computer Science
September, 2001

Certified by .....
Duane S. Boning
Associate Professor
Electrical Engineering and Computer Science

Accepted by .....
Professor Arthur C. Smith
Chairman, Department Committee on Graduate Students
Electrical Engineering and Computer Science

Integrated Model-Based Run-to-Run Uniformity Control for Epitaxial Silicon Deposition

by

Aaron Elwood Gower-Hall

Submitted to the Department of Electrical Engineering
and Computer Science on August 10, 2000, in partial fulfillment of
the requirements for the degree of Doctor of Philosophy in
Electrical Engineering and Computer Science

Abstract

Semiconductor fabrication facilities require an increasingly expensive and integrated set of processes. The bounds on efficiency and repeatability for each process step continue to tighten under the pressure of economic forces and product performance requirements. This thesis addresses these issues and describes the concept of an "Equipment Cell," which integrates sensors and data processing software around an individual piece of semiconductor equipment. Distributed object technology based on open standards is specified and utilized for software modules that analyze and improve semiconductor equipment processing capabilities.

A testbed system for integrated, model-based, run-to-run control of epitaxial silicon (epi) film deposition is developed, incorporating a cluster tool with a single-wafer epi deposition chamber, an in-line epi film thickness measurement tool, and off-line thickness and resistivity measurement systems. Automated single-input-single-output, run-to-run control of epi thickness is first demonstrated. An advanced, multi-objective controller is then developed (using distributed object technology) to provide simultaneous epi thickness control on a run-to-run basis using the in-line sensor, as well as combined thickness and resistivity uniformity control on a lot-to-lot basis using off-line thickness and resistivity sensors.

Control strategies are introduced for performing combined run-to-run and lot-to-lot control, based on the availability of measurements. Also discussed are issues involved with using multiple site measurements of multiple film characteristics, as well as the use of time-based inputs and rate-based models. Such techniques are widely applicable for many semiconductor processing steps.

Thesis Supervisor: Duane S. Boning
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

Wow, what a long and exciting ride it has been. At first, there's no clear direction and the end seems nebulous and far away. Then an end can be seen, but the path is unmarked and full of wrong turns. It seems like you're a year away, but it's really more like two or three. Then suddenly you wake up and realize it's over! Thinking back over all of the years and countless interactions, it becomes obvious that this brief section cannot do justice to the vast number of people that supported and sustained this work.

First, to Duane Boning, the driving force behind so much fine research here at the Microsystems Technology Labs (MTL). Thank you for providing the resources, direction, and spirit to enable this work. Your energy and open-mindedness produce an amazing research environment with a wonderful blend between goal-oriented guidance and the freedom to innovate and explore new ideas. You are an inspiration and a role model for me and many other fine individuals.

The caliber of the faculty and staff here at MIT never ceases to amaze me. To Mike McIlrath, a Research Scientist at MTL, thank you for helping me understand and express the concepts of distributed object-based computing. Professors Donald Troxel and Leslie Kolodziejski, thank you for taking part in my thesis committee and providing your constructive feedback.

Many thanks go out to the industrial partners that created the necessary funding and experimental opportunities that made this work possible. In particular, On-Line Technologies acted as a central hub for coordinating the interactions between Applied Materials, Wacker Siltronic, and MIT. Peter Rosenthal, thank you for supporting this work from within On-Line Technologies, and serving as a well-spoken champion of our technology. Oskar Bosch, thank you for all of your hard work at integrating your Cell Control software at On-Line Technologies with our controller implementation. To Ann Waldhauer at Applied Materials, thank you so much for providing your expertise in epi deposition, as well as sharing with me the ups and downs of running the first control experiments and our DOE on your development system. To John Lobo at Wacker Siltronic, thank you for taking an interest in our work and helping us organize for an industrial demonstration of the control system.

Of course the day-to-day life in our office is an integral part of the experience, filled with talented research group members who act as both sounding boards for new ideas, as well as much needed distractions from our sometimes daunting research challenges. Taber Smith, while you taught me just about everything I know about run-to-run process control, I feel that you've given me many more valuable lessons about character, professionalism, and the importance of "the big picture." Your energy and determination are an inspiration to me, and I look forward to working closely with you again as I finish this degree. David White and Han Chen, thank you both for many stimulating conversations, both technical and otherwise. Han, thanks for putting up with all of my ramblings about computer hardware and software. I have seen a few generations of group members pass through the office during my transition between first year Master's student and senior Ph.D. student. Many thanks go out to Ka Shun Wong, Chantal Wright, William Moyne, Minh Le, Sandeep Sadashivapa, Eric Stuckey, Angie Nishimoto, Brian Lee, Vikas Mehrotra, Tamba Gbondo-Tugbawa, Tae Park, Karen Gonzalez-Valentin, Joe Panganiban, and Allan Lum. Each of you has helped make this a more enjoyable work environment.

A special thank you also must go out to Debb Hodges-Pabon; you saved my bacon countless times with your amazing ability to find conference rooms and overhead projectors at the last minute. Also, thank you for keeping the life-giving hot chocolate machine stocked and operational!

Last but not least, it is important to acknowledge the support and love of my close family, who have given me a strong foundation and sense of purpose. First, to my wonderful wife, Kimberly, who has endured much to make this accomplishment possible. Our unquestioning love and devotion has sustained me through these last few years and given me reason to look beyond this degree toward a lifetime of shared experiences. To Kimberly's parents, Drs. Steve and Lee Whitfield, your hospitality, warmth, and encouragement have helped me remain focussed and confident. Many thanks to both you and Kimberly's sister, Ande, for attending my thesis defense and providing some much needed moral support. To my sister Rose, thank you for all of our wonderful childhood memories, and for giving me a home in which to live while I was working at On-Line Technologies in Connecticut! Finally and most importantly, to my parents Donald and Lucy Gower, I owe the two of you more than I can ever know or express. Thank you both for being a constant source of comfort and strength.

This research has been funded in part by the MIT Leaders for Manufacturing Remote Diagnostics Program, by a National Science Foundation Small Business Innovative Research grant, and by a National Institute of Science and Technology Advanced Technology Program grant.

Table of Contents

List of Figures

Chapter 6.

Chapter 7.

List of Tables

Chapter 1

Introduction

Semiconductor fabrication facilities (fabs) require an increasingly expensive and integrated set of processes. The bounds on efficiency and repeatability for each process step continue to tighten under the pressure of economic forces and product performance requirements. The semiconductor industry's desire for superior manufacturing capability can be addressed through the use of networked, integrated equipment and data processing software, which are supported by an automated, coordinated flow of product wafers and information throughout the fab. The integration is hierarchical in nature and can be approached at a number of levels. Many industrial fabs currently use high-level Computer Integrated Manufacturing (CIM) systems to coordinate the flow of wafers throughout the fab and maintain some form of database history on wafer lots or possibly individual wafers.

Today, one of the most promising avenues for increasing semiconductor manufacturing capability comes from a fairly low level of integration, that of the individual Equipment Cell (Cell). A Cell is composed of an individual piece of fabrication equipment and any sensors or software that evaluate or affect the performance of that equipment. This includes any sensors that measure the equipment state, the process state, or the wafer state for product which is processed by that equipment. A Cell also includes any data processing software that manipulates information from the equipment or sensors in the Cell. Clearly there is room for overlap between similar Cells which, for example, could time share a single sensor or a data processing resource. The equipment itself is the true center-

piece of the Cell, and for development purposes, one can usually consider an individual Cell as a unique entity that is managed by some form of "Cell Controller." This Cell Controller coordinates the activities within the Cell and can act as a primary contact point for a higher level CIM system. Thus the integrated software and hardware components constitute the Equipment Cell, while their combined actions represent Cell Control.

The research for this thesis consists primarily of the specification and implementation of modular Equipment Cell components to demonstrate control of a Centura epitaxial silicon (epi) deposition tool from Applied Materials, using epi film metrology tools from On-Line Technologies. Software engineers from On-Line Technologies have aided in specifying and implementing the equipment and sensor integration. The contributions of this thesis include implementing Cell infrastructure and data processing modules for the Cell, as well as specifying interfaces to those modules. It is worth noting that a Cell-based manufacturing system provides a sound basis for any highly integrated, information-rich manufacturing environment, and is not tied specifically to the semiconductor industry.

This thesis focusses primarily on the use of run-to-run control strategies for semiconductor processing. Run-to-run control is a form of feedback process control whereby measurements made during or after a wafer processing step are fed into a controller, which in turn assesses the product quality and adjusts process settings for the next wafer (or lot). These adjustments should drive product quality towards an optimum by reacting to changes in the process and filtering out noise in the system.

As the name implies, a key feature of run-to-run control is its ability to react once per execution of a process step, creating an inherently discrete-time control system. Depending upon the availability of feedback information, control can occur on a run-to-run basis

or more generally on a “lot-to-lot” basis, where a “lot” could be any number of wafers. This thesis will deal with a number of issues related to combined run-to-run and lot-to-lot feedback control, where multiple feedback loops are in place.

As is the case with many run-to-run control scenarios for semiconductor processing, the goal is to drive particular thin film characteristics uniformly to a target. That is, we want specific features of the resulting film to meet the “ideal” target specifications across the whole wafer (not just in their average values or at a single location). This thesis contains a detailed treatment of process modeling and control techniques for uniformity control of multiple film characteristics.

The following sections provide background information for this work. First, Section 1.1 considers the components that an integrated Equipment Cell should (or could) contain and discusses the economic and performance benefits of this approach. Section 1.2 takes a step back and describes the hierarchical structures involved in controlling semiconductor fabrication. Section 1.3 describes how Equipment Cell Control fits into the hierarchical framework for semiconductor fabrication. Section 1.4 briefly touches on some important concerns regarding network security. Finally, Section 1.5 provides an overview and lead-in for the work described in this thesis.

1.1 Integrated Equipment Cell Control

Today’s highly competitive semiconductor fabrication industry presses companies to become ever more efficient. Integrated Equipment Cell Control provides a great opportunity to reduce production cost and improve quality. This section first defines the basic building blocks for an Equipment Cell, then discusses how such an integrated system addresses economic and performance goals for semiconductor manufacturing.

1.1.1 Cell Design

As described earlier, a Cell includes an individual piece of fabrication equipment and any sensors or software that can evaluate or affect the performance of that equipment. The concepts of Equipment Cells and Cell Control are not new. Researchers at the University of Michigan have specified and implemented a Generic Cell Controller [MM92]. The Microelectronics Manufacturing Science and Technology (MMST) project at Texas Instruments designed a fully automated semiconductor fabrication facility, including a Cell-like architecture for process control [BCDH94][SBHW94]. Previous work at MIT demonstrated automated run-to-run control through an integrated cell for an Applied Materials plasma etcher [Gow96]. Much of the previous work provides good structural foundations, but they were implemented before the arrival of today's open Internet and World Wide Web ("web") based protocols and applications. A more recent collaboration between SEMATECH, Advanced Micro Devices (AMD), Honeywell, ObjectSpace, and Oakleaf Engineering produced the Advanced Process Control Framework Initiative (APCFI) specifications, which attempts to provide detailed specifications for Cell Control structure and software interfaces using Internet-based standards [Mul97][Gro99]. The APCFI specifications are used as guidelines wherever possible for this work. Figure 1-1 shows a basic Cell Control structure from the APCFI 1.0 Specifications. Figure 1-2 graphically depicts a generic Equipment Cell, as envisioned for this thesis.

Figure 1-1: APCFI Cell Control

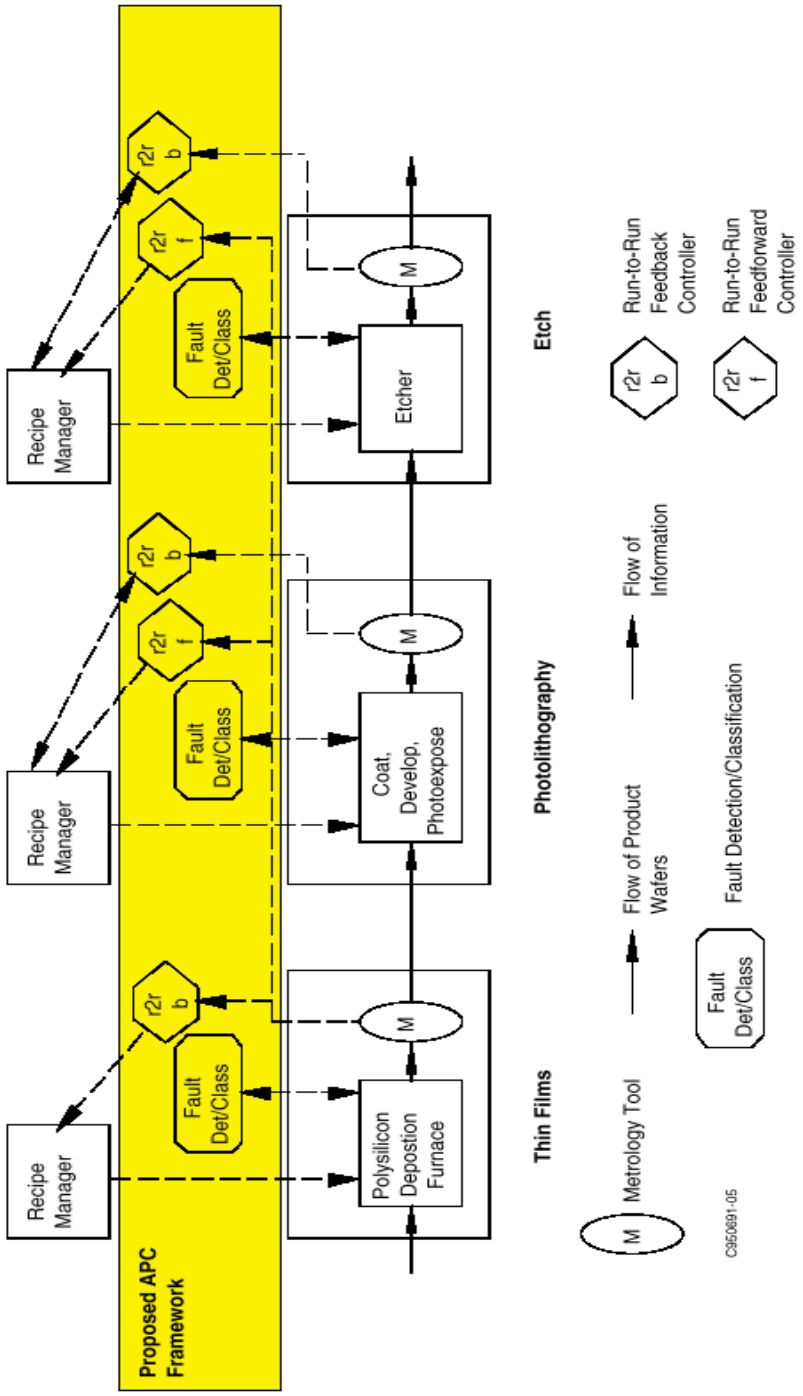
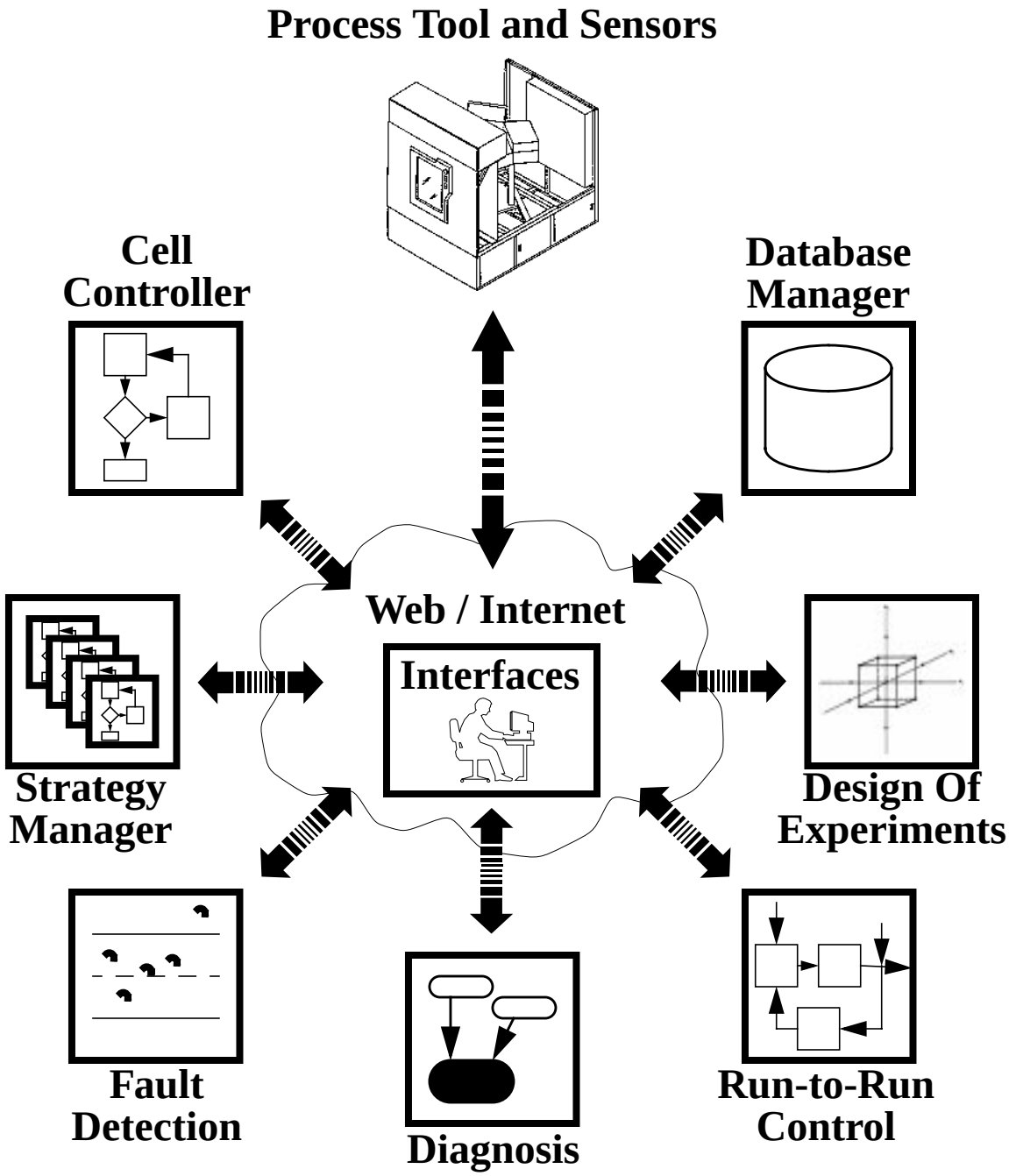


Figure 1-2: Generic Equipment Cell



The Cell structure may be broken down into three categories: Equipment, Data Processing Software, and Infrastructure. The following sections describe these categories and place each Cell component accordingly.

Equipment:

Equipment components include hardware that provides specific capabilities, in conjunction with software controllers to enable networked access and control of the hardware. Included are the processing tool and the sensors.

- **Processing Tool**: At the heart of the Equipment Cell is the actual tool that generates product. This is the most important part of the Cell and represents the only true revenue generating component. The sole purpose of the other Cell components is to improve the effectiveness of the processing tool.
- **Sensor(s)**: There are many metrics that can be used to judge the processing tool's performance and health. Measurements mainly fall into three categories: Equipment state, Process state, or Wafer state. Equipment state sensors are often built into the processing tool and provide information about the health of the equipment. In-situ Process state sensors measure the environment which transforms the incoming raw material (pre-processed wafers) into the output product. Wafer state sensors measure properties of the product itself before, during, or after processing by the equipment. Ideally these sensors can automatically and non-destructively measure product wafers as part of the production line.

Data Processing Software

Data processing components analyze data collected from the equipment to evaluate and/or enhance the process capability. Included are modules for run-to-run control, fault detection, fault diagnosis, design of experiments, cell control (coordination), and strategy management.

- **Run-to-Run Control**: Run-to-run control modules monitor the quality of the product coming out of the process tool (via one or more sensors) and adjust the equipment settings on a per-run basis to maintain process outputs at target specifications. Often functional models of the product quality in terms of the process settings are used to select optimal settings. These models are updated as sensor

information is passed to the run-to-run controller from the sensor(s).

- **Fault Detection**: Fault detection modules track Equipment state, Process state, and Wafer state (or any subset of those three) and set off alarms when a state deviates from its normal operating range. Detection algorithms can use simple Statistical Process Control (SPC) charting, or complicated process modeling techniques.
- **Fault Diagnosis**: After a Fault Detection unit raises an alarm, it is up to the Fault Diagnosis software to analyze the available data and suggest likely causes for the fault. Often Fault Detection and Diagnosis capabilities can be combined within one package.
- **Design of Experiments (DOE)**: Many run-to-run, fault detection, and diagnosis modules utilize process models. Building such models often requires the design and execution of experiments specifically for that purpose. A DOE module can construct and execute a sequence of experimental runs, which can be used for process modeling.
- **Cell Controller**: If the processing tool is the heart of the Equipment Cell, then the Cell Controller represents the brain. This software receives the various events, alarms, and data from all other Cell components and coordinates the actions between them. Graphical flow charts or scripting tools are often used to specify module interaction strategies. One important Cell design issue that must be addressed is determining how centralized the Cell Control will be. At one extreme, there is no Cell Controller and all actions and events travel directly between subcomponents. At the other extreme, all information passes through the Cell Controller for routing and no component is directly aware of any other. Somewhere in between lies a set of optimal Cell Control architectures, and a complete Cell Control system must address this issue.
- **Strategy Management**: The Cell Controller should provide flexible configuration capability for coordinating module interactions. Some form of database or an actual strategy manager should provide strategies for the Cell.

Infrastructure:

The Cell infrastructure should be designed to make the system flexible and effective.

While these concepts are not necessarily individual components of the system, they enable the components described above to find and interact with each other. The infrastructure is composed of network communication, data access, and web/Internet-based interfaces.

- **Network Communication**: An Equipment Cell relies on a large number of separate applications, all of which can exchange information with each other. It is unreasonable to require that the Cell components run on the same computer, or even on the same platform. Ideally all of them will speak the same language (across the Inter/intra-net where necessary), so it is important to pick a flexible, distributed communications standard through which the modules can interact.
- **Data Access**: Most Cell modules require data logging and/or data transfer capabilities. There are clearly many advantages to centralizing data storage for optimized data flow pathways. Selection of a convenient distributed database provides these capabilities where necessary. Ideally the interface to this database will be through a database manager, which exposes the same type of network communication interface used by the rest of the Cell components.
- **Web/Internet-Based Interfaces**: Various status or data views and configuration interfaces should be available via an easily distributable interface wherever possible. Ideally such interfaces are based on internet standards and can be easily brought up with a web browser. This feature provides flexible management and status viewing capabilities. (Specialized local interfaces may be necessary for some components as well, which is often the case for equipment controllers.)

Limitations

The applicability of Equipment Cells can be limited by a lack of widely available network infrastructures and/or integrated metrology tools. A Cell as described above requires data-rich environments with equipment and sensor data that are highly available for distributed data processing modules. The recent explosion of the Internet and intranets has made such integrated systems feasible. Also important is the development and integration

of in-line sensors. The effectiveness of Cell Control depends greatly on fast, accurate, and meaningful sensor data that convey the quality of product leaving the processing tool. Ideally non-destructive wafer state sensors can provide feedback on actual production wafers, preferably on every wafer.

1.1.2 Economic Drivers for Cell Control

Many cost reduction strategies for semiconductor manufacturers involve improving a fab's manufacturing efficiency. That is, one wants to create a given amount of product with the least amount of equipment and material cost, or alternatively, create the most product using fixed quantities of equipment and material resources. A common estimate of production efficiency is based on the Overall Equipment Effectiveness (OEE) for all of the manufacturing tools in the fab [Lea97]. OEE is based on the product of Availability, Efficiency, and Quality values for the equipment, all of which are ideally 100%. Availability refers to the percentage of time that the tool is scheduled for production. This metric penalizes the equipment for scheduled downtime and is mainly a function of the maintenance requirements for the equipment and the scheduling efficiency of the production line. Efficiency measures the amount of product that the equipment actually generates during its scheduled operational time as a percentage of the maximum theoretical product that could be produced in that amount of time. This value penalizes unscheduled downtime, periods where the equipment operates "more slowly" than it could, and any equipment idle time, such as upstream or downstream production line bottlenecks that limit product flow through the tool. Finally, quality measures the amount of product passing through the equipment that meets specification limits as a percentage of total product passing through the tool.

Today's CIM systems and high level integration technologies often seek to improve OEE through better scheduling and automation capabilities. Work at Texas Instruments, under their MMST (Microelectronics Manufacturing Science and Technology) program, demonstrated that a highly integrated, networked fabrication facility reduced wafer cycle time by almost an order of magnitude [BCDH94]. They used an object oriented approach to redesign the whole factory automation environment, resulting in a remarkably flexible system that is able to incorporate a wide variety of fabrication equipment. Here at MIT, the Computer-Aided Fabrication Environment (CAFE) provides a CIM framework for the Microsystems Technology Laboratories [McI92].

CIM scheduling systems try to optimize OEE availability by coordinating and optimizing the equipment maintenance schedule and the flow of product between processing tools. OEE efficiency can be affected by matching product flow rates through the fab to avoid bottlenecks. Use of scheduling strategies alone implies that the limitations of the equipment are accepted and are used as scheduling constraints. Clearly these optimization techniques are important and effective, but they are not part of the research for this thesis. Instead, this work will consider integrated Cell Control, which attempts to fundamentally increase the maximum theoretical productivity of a given piece of equipment. Thus an integrated Cell can simultaneously improve OEE through availability, efficiency, and quality.

Availability can be improved in the following ways:

- Shorten process "tune-up" time after scheduled process disturbances
  - Disturbances such as scheduled maintenance, system warm-up after a period of downtime, and switching of product type, can all require process engineers

to spend time running test wafers for tuning-up and verifying that the process is creating output that is within the specification limits and equipment control limits. Good sensors and equipment models can drastically shorten the time and effort it takes to tune-up a process.

- Flexible Manufacturing
  - Short (and reliable) tune-ups between processing different products can enable a single piece of equipment to become more flexible. For example, instead of lightly scheduling two similar tools to produce two different products, a single tool could utilize a higher availability and perform both tasks.
- Longer run lengths between scheduled maintenance
  - Run-to-Run control can help keep a process on target for longer periods of time, even in the face of equipment wear or material build-up. Scheduled tune-ups and maintenance can often be done less frequently when the process is using feedback control.

Efficiency can be improved in the following ways:

- Shorten unscheduled downtime
  - Disturbances such as faults can require maintenance and subsequent process tune-ups. Integrated Fault Detection and Diagnosis can help process engineers rapidly find and fix the problem, then a run-to-run control module can quickly get the process tuned and running again.
  - Distributed interfaces enable experts to diagnose problems from anywhere on the inter/intra-net, saving valuable time by not (necessarily) requiring them to bring up interfaces from fixed locations.

Quality can be improved in the following ways:

- Fewer misprocessed wafers
  - Integrated metrology and fault detection enable rapid triggering of alarms and halting of wafer processing, thus avoiding the misprocessing of many incoming wafers.
- Fewer (random) out-of-spec wafers

- Run-to-run feedback control will tighten statistical process control limits around the process targets. Statistically fewer wafers would fall out of the specification limits. Also, product within the specification limits will tend to be closer to target.

Other economic benefits of Cell Control:

- Fast production ramping
  - Integrated process modeling and tuning provides fast fab start-up and production ramping. Thus the equipment starts producing revenue earlier.
- Reduction in clean room space requirements
  - Space for equipment in a clean room is expensive. More efficient equipment can enable production with fewer tools. There is a possible trade-off here, depending upon how much clean room space the sensors and other Cell-related hardware require.
- Fewer test wafers
  - Effective process modeling enables faster tune-ups using fewer test wafers to qualify a process.
  - Integrated metrology that can non-destructively measure product wafers reduces or even eliminates the need to process and measure test wafers during normal production. The product wafers themselves are used to evaluate the process instead of monitor wafers.

1.1.3 Product Performance Drivers for Cell Control

Manufacturing efficiency benefits are not the only driving forces for integrated Cell Control. As semiconductor devices continue to become smaller, operate at higher speeds, and utilize a greater number of processing steps, the specification limits for the processes tighten. Product which met quality standards for previous generations suddenly stops passing inspection. There are only two choices at this point: buy new, expensive processing equipment that has better repeatability, or improve the older tools' capabilities. In

essence, these performance drivers are demanding improvement in OEE quality. Cell Control provides improved quality through the use of run-to-run control, which can shrink the statistical process control limits for the equipment, thus enabling the tool to reliably meet tighter specification limits.

Cell Control capability also becomes more important as wafer diameters continue to expand. The era of 300 millimeter wafers is quickly approaching, and with it come many concerns about controlling the uniformity of product features across the wafer. Processing tools are likely to start including more complex recipe settings to enable spatial uniformity control. Run-to-run feedback control could significantly simplify and improve the effectiveness of using these settings for tuning after process disturbances or to compensate for process drift.

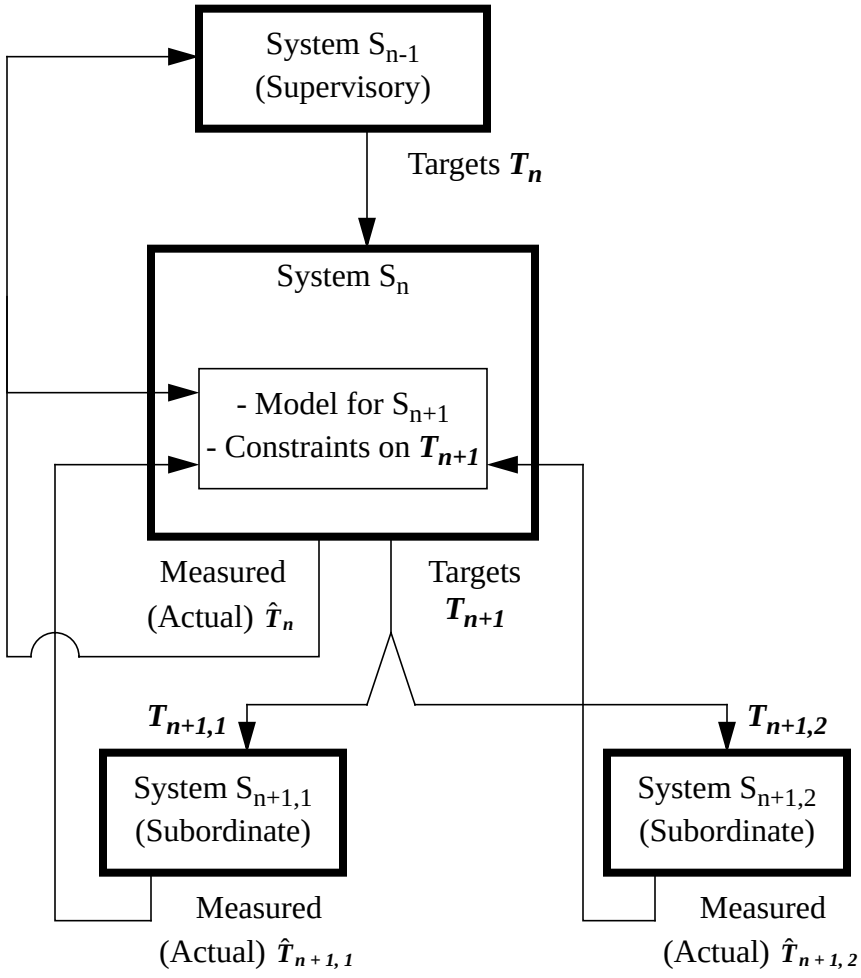
1.2 Hierarchical Control for Semiconductor Manufacturing

Semiconductor manufacturing requires a complex and inherently hierarchical set of decision-making processes. The structuring and implementation of these processes may be accomplished by a variety of techniques, containing a varying degree of improvisation and mathematical rigor. An elegant and useful approach to the scheduling and control of a semiconductor fabrication facility is presented in [SBG94], which is used as a basis for much of the discussion below.

At the top level, decisions are made to dictate how much of each product type should be produced, which provides target outputs for a whole facility, or possibly multiple facilities. These decisions are based upon models of consumer demand and facility fabrication capabilities. For example, linear programming techniques are used in [LH96] to solve the production planning problem. Target production levels are passed down to the next level

of the control hierarchy, which must turn its control knobs to satisfy those targets. The highly **self-similar** structure of this scenario begins to fall into place: the control knobs from one decision-making level are the short-term target set points (or trajectories) for the level immediately below. Figure 1-3 graphically depicts this structure, where each system (S_n) in the hierarchy interacts with a supervisory system (S_{n-1}) and one or more subordinate systems (S_{n+1}). Generally it is convenient and accurate to consider hierarchies where there is a single supervisory system and possibly multiple subordinate systems [MM92].

Figure 1-3: Hierarchical Decision Making (Control) for Manufacturing



As mentioned, the hierarchy begins with the highest level decisions, such as facil-

ity-wide production targets, which are based on external demands and capacity models. Terminating branches generally occur when real-time systems are reached. They are often found in low level equipment controllers, such as gas flow meters and temperature controllers. These systems are generally designed to track their target set points directly, ending the hierarchy. This is basically due to the lack of discrete time-steps between which a lower level system could operate.

This structure makes intuitive sense, but one must clarify a number of issues. The following section describes methods of breaking down a complex manufacturing system into these hierarchical systems.

1.2.1 Hierarchy Division

Implementing a hierarchical design requires a breakdown of decision-making tasks into an appropriate tree-like structure. A crucial question to ask is how one creates the boundaries and data exchanges between levels. It has been suggested that the division should be based upon a classification of the available events within the facility. Here, events are classified by controllability, frequency, and location [SBG94].

"Controllability" classifies events as either operations that can be controlled and manipulated (i.e. equipment setup changes or target settings), or events that must be reacted to (i.e. machine failures). Uncontrollable events result in actions from a decision-making algorithm, while controllable events are the outputs from the algorithm. Note that a low-level equipment failure event can easily propagate itself up the decision-making hierarchy and quickly affect many other systems. For this reason, all systems need to have the ability to deal with real-time events, even if their (normal) primary actions take place infrequently.

"Frequency" classifies events by the relative rates at which they occur. Generally uncontrollable events and decisions with similar frequencies should be considered together. This maps nicely into the structure shown in Figure 1-3. System S_n sets targets for its subordinate system, S_{n+1}. Clearly the internal control loops within S_{n+1} must cycle at a higher frequency than the process that sets its target set points, otherwise its dynamics need to be incorporated into S_n. This leads to an interesting observation that frequency boundaries can often be determined when a **steady-state** model of the subordinate system can be used. The dynamics required to achieve these targets should be handled by the subordinate system, which is operating at a higher frequency, but should not need to be modeled by the current system, S_n. This "shielding" of system dynamics is why simple, possibly static, models can often be used to represent subordinate systems. Such a structure helps to simplify and distribute the decision-making processes.

Alternatively, the frequency of decision-making opportunities or requirements can be based on a simple breakdown of a given system's operation. For example, consider manipulating the deposition time for an epi deposition process. It makes sense to set this value on a per-run basis at best, or perhaps even at a lower frequency. This set point does not directly correspond to a target for an underlying dynamic system. However, frequency-based grouping for this decision-making event still makes sense.

"Location" classifies events by the physical regions influenced by or influencing an event. Intuitively, events that physically overlap are likely to affect many of the same decision-making processes. Buffers between processing steps can help generate finer grained classification boundaries.

Theory involving the full hierarchical decomposition of complex systems is beyond

the scope of this thesis. However, this decomposition can be primarily based on the controllability, frequency, and location-based classification of events. Dynamic programming techniques are used in [SBG94] to demonstrate hierarchical control of a semiconductor fabrication system.

1.3 Cell Control and the Decision-making Hierarchy

Cell Control represents a fairly low level system within the hierarchical structure described in the previous section. Actually, a Cell's decision-making level is generally one step above the real-time controllers within the processing equipment. The Cell's supervisory controller, which will be referred to as a "multi-stage" controller, selects target outputs for the Cell. Generally these targets represent some desired wafer state features that should be achieved as a result of a wafer having passed through the Cell's processing equipment. The Cell Controller must then select appropriate recipes for processing the incoming wafers. Equipment recipes are generally comprised of such things as electrical power, gas pressures, and gas flows. Consider what these settings actually represent; they are target outputs for real-time controllers within the processing equipment.

An Equipment Cell can select process recipes using a number of different techniques. A commonly used approach is essentially open-loop control, where the controller simply selects the recipe corresponding to the desired targets, and relies on process engineers to keep this recipe optimized. This simplistic level of control does not technically require Cell Control at all; a multi-stage controller can handle recipe selection through a simple lookup table. An Equipment Cell might still exist, however, to perform fault detection (e.g. SPC), and possibly fault classification, to alert process control engineers and multi-stage controllers when the equipment is "out of control."

A more sophisticated Cell uses process sensor data for run-to-run control, as well as fault detection and/or classification. Once run-to-run control is incorporated, Cell Control is accurately represented by the hierarchical structure shown in Figure 1-3. The run-to-run controller maintains an internal model of the processing equipment, which is used to select recipes (targets for real-time controllers in the equipment).

The following sections discuss the recent trends and technologies related to hierarchical run-to-run control. First, the interactions with supervisory (multi-stage) controllers and subordinate (real-time) controllers are presented, then the “state of the art” in run-to-run control is described.

1.3.1 Multi-stage Control

There are many possible policies and constraints governing a run-to-run controller’s supervisory system. Broadly speaking, however, this system is responsible for providing targets to a number of Equipment Cells (run-to-run controllers). Presumably these Cells contain a sequential, or otherwise related, set of processes, where the set points for each step are inter-dependent in some way. Thus the term “multi-stage control” depicts the basic functionality for these supervisory systems. This controller could be a simple lookup table that maintains a pre-set sequence of processes, or a complicated decision-making system that dynamically models and reacts to the various capabilities of, and measurements from, its subordinate process steps.

Figure 1-4 shows the structure of a multi-stage controller that manages targets for three consecutive Cells in a process sequence. Targets for the complete multi-stage unit ($\mathbf{T_n}$) are passed down from above. These could represent a number of individual targets for each of the Cells or, more likely, they are simply target outputs for the final Cell in the

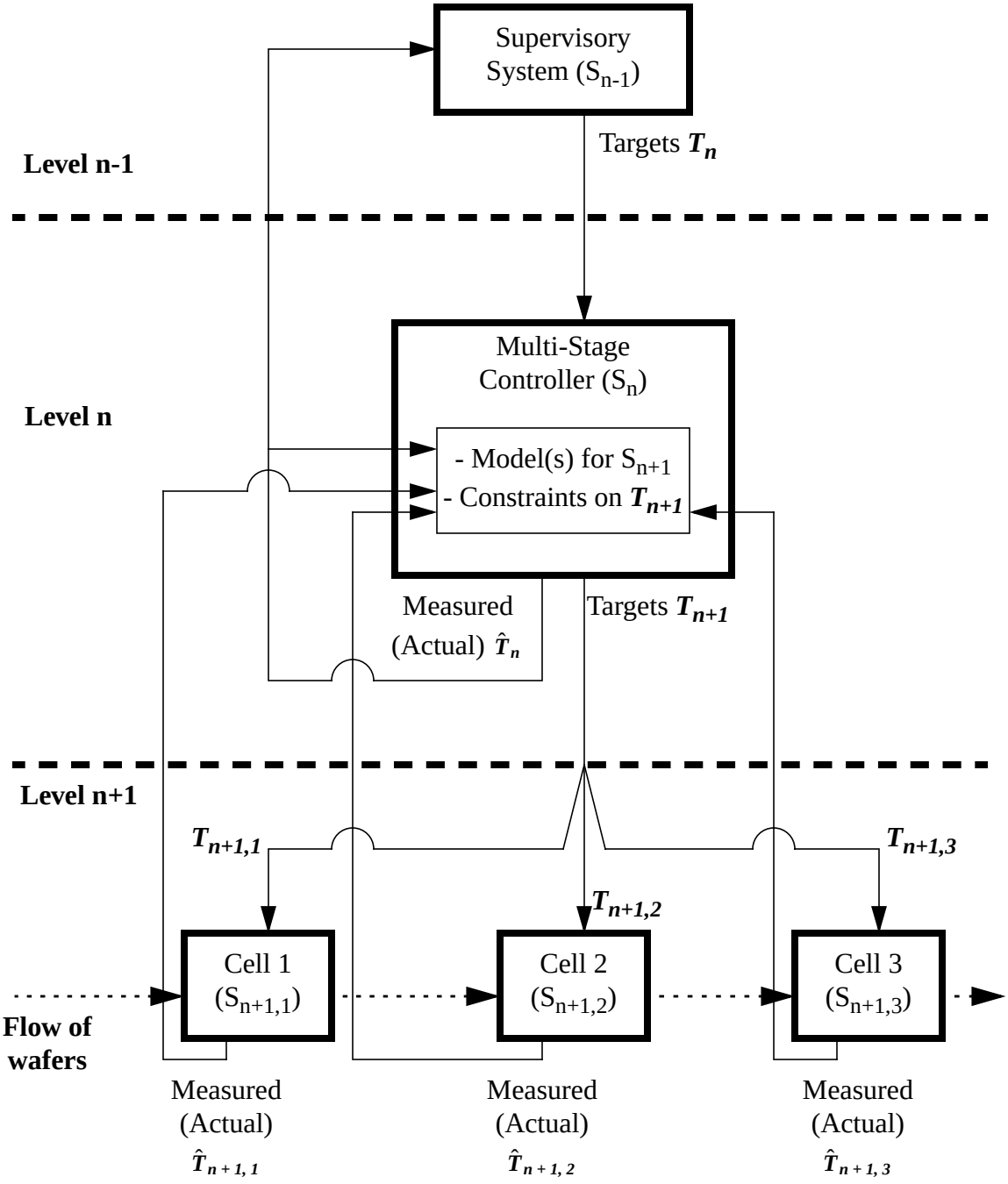
sequence. Usually engineers only care about the quality of the final output from the multi-stage system; how the product gets to that state is irrelevant to the supervisory system (S_{n-1}). It is the multi-stage controller's job to monitor the progress of the wafers as they pass through each Cell, and to optimize the remaining processing steps to hit the final stage's targets. The controller can compensate for processing errors that might occur at one stage by modifying targets for subsequent stages.

One of the most straightforward implementations of multi-stage control is feed-forward control, which has been analyzed and used in a number of semiconductor processing scenarios [SCKT94][Lea96][CMR98][RWFG99]. Generally, feed-forward control uses static models to select target outputs for a given Cell, based upon measurements from previous process steps (typically, but not necessarily, from other Cells). The following four step fabrication process flow is considered in [SCKT94]:

1. Silicon oxidation step
2. Aluminum metallization step
3. Lithography step
4. Aluminum etching step

This flow is used to generate capacitors, and the multi-stage controller's goal is to maintain the final capacitance of the devices. The system uses feed-forward information about oxide thickness to adjust targets for the lithography and aluminum etch steps, based upon a theoretical derivation of capacitance as a function of oxide thickness and the area of the aluminum plates. Other research has used feed-forward information from photolithography steps to maintain post-etch Critical Dimension (CD) [RWFG99].

Figure 1-4: Multistage Control Structure



To date, most multi-stage controller demonstrations utilize fixed models for generating each Cell's targets. That is, while individual Cells are operating under closed-loop feed-

back control, the multi-stage controller is not. (To be fair, if one assumes that key Equipment Cells can accurately hit their targets, then the multi-stage controller can often successfully operate in an open loop.) Basically the feed-forward controllers described above do not keep any state information. Given two wafers with the same processing history and measurements, these controllers will always provide the same targets for the next stage. A more versatile, dynamic multi-stage controller would maintain process capability information for each Cell and, at each step, plan complete target trajectories through all remaining stages. Theoretical work has been done with such controllers based upon Dynamic Programming (DP) algorithms [SBG94][Fen00].

1.3.2 Real-Time Control

While a Cell Controller receives directives from a multi-stage controller from above, it typically manipulates a number of real-time controllers below. From its structure, one can see that hierarchical control requires good performance at lower levels before creating more sophisticated supervisory controllers at higher levels. Precise real-time systems are a prerequisite for good Cell Control, much like precise Cell Control is a prerequisite for good multi-stage control. To put this in context, if a Cell controller asks for a gas flow rate of 20 sccm, then the real-time gas flow meter must do its job, or a model-based prediction of process results is likely to be considerably off. (Depending on the amount of true drift in such a controller, and the amount of random variation, Cell Control may or may not help the situation.)

Much of the real-time control found in today's fabrication facilities is localized within the actual processing tools. Equipment vendors invest significant resources to ensure that recipe set points are accurately maintained every time the process is executed. The com-

mon practice of running equipment under open-loop control demands this type of repeatability even more than a system using closed-loop run-to-run control. It is important to remember that these embedded real-time controllers may be very complex and inherently multi-input-multi-output. For example, the simultaneous control of gas pressure, flow, and temperature requires the control of strongly coupled outputs. Fortunately, most semiconductor processing tools in use today provide highly repeatable, highly accurate real-time recipe control.

With good embedded real-time control available, the obvious question is: why is there so much research involving more advanced real-time and run-to-run control? The reason is basically that, while these real-time systems are very good at controlling the outputs they **can** measure, there are many other important outputs that they **cannot** see or react to. Factors like equipment aging, variability of incoming materials, and changing ambient conditions create circumstances where those perfectly tuned recipe settings no longer produce the desired results.

Typically a process step can be abstracted in the following way:

1. Machine settings (recipes) induce an equipment state.
2. An equipment state induces a chamber state or wafer environment.
3. The chamber state or wafer environment induces a wafer state.

(A more detailed process modeling framework is found in [BMPS92].) The goal of each semiconductor processing step is to achieve the final target wafer state. Presuming that the machine settings uniquely map to an equipment state, the equipment state uniquely maps to a chamber state, and the chamber state uniquely maps to a wafer state, then perfect equipment state control should yield perfect wafer state control. However, various drifts and shifts in the system can break any or all of these links. Thus much of the ongoing

research involving real-time control uses advanced sensors to directly measure and maintain chamber state and/or wafer state variables [ZR95] [RP95] [Ras95] [Asp97] [HVIK97] [KGCK97] [WD98] [Joh98]. Others are focussing on temperature uniformity control as a means of controlling wafer state uniformity [ESK96] [TZA99] [DD99]. Directly controlling wafer state is clearly the ideal situation, but controlling the chamber state often yields better results than simply maintaining an equipment state.

Sometimes an approximation to real-time control is used, where a number of mid-course corrections are issued instead of true continuous-time feedback [Won96]. In truth, most of these “real-time” systems are actually discrete-time systems with a “relatively fast” sampling rate. In this sense, many controllers use a lot of small mid-course corrections rather than continuous control. Often this is due to some sampling period for the sensor and/or time needed to analyze the sensor data. As long as the sampling rate is high enough with respect to the frequency content of the system’s continuous-time outputs (e.g. Nyquist sampling criteria are met), continuous-time analyses and control strategies are still applicable.

However, an important question to consider is whether or not real-time control is required; typically this question is not fully addressed by researchers. For example, real-time control is provided for a reactive ion etching (RIE) system in [Ras95]. It is noted that the RIE process is not robust and requires frequent tune-ups. They use in-situ sensors to measure the dc bias voltage and the concentration of fluorine. These outputs are controlled through real-time adjustments to the (CF_4) flow rate and the power delivered to the plasma. Thus a Cell Controller can select chamber state set points instead of simply selecting gas flow and plasma power recipes. While the paper does a good job describing the

problem and presenting a promising solution, it does not consider the possibility that a run-to-run control strategy might achieve similar results. In general one should usually consider exploring this option first, since a run-to-run controller is often simpler and faster to build and integrate. Recipe updates between runs are usually available through the equipment front panel or through a communications port. Access to real-time controllers during a process is typically not a built-in option.

The real question is: what is the time scale over which equipment process settings need to be modified? Does the chamber state and/or the wafer state (trajectory) drift or change in a non-deterministic manner **during** the course of **single** processing step? If so, then real-time control is probably required. If the disturbances take place gradually over a number of runs, or if the disturbances are deterministic and can be modeled (e.g. a relatively constant drift), then an appropriate run-to-run controller can often achieve similar levels of control with lower integration costs. (The real-time sensor data might still be needed, but continuous feedback control might not be required.)

1.3.3 Run-to-Run Control

Much like real-time control, run-to-run (Cell) control has received considerable attention. As noted, this type of control is often the easiest and the most appropriate. Run-to-run control modules monitor the quality of the product coming out of the process tool (via one or more sensor) and adjust the equipment settings on a per-run basis to maintain process outputs at target specifications. Often functional models of the product quality in terms of the process settings are used to select optimal settings. Figure 1-5 presents a sample run-to-run control hierarchy, where “Level n” is centered on the Cell Controller. To demonstrate some of the possibilities, an external mid-course (or real-time) controller

is included, which monitors and maintains certain chamber state or wafer state parameters during each run. Both the Cell Controller and the mid-course controller can provide set points for real-time controllers within the actual processing equipment.

While there has been extensive work on real-time and run-to-run control for semiconductor processing, there has been comparatively limited focus on multi-stage control. This is at least partly due to the lack of highly accurate run-to-run control capability. Typically run-to-run controllers use process models that provide stable control, but they do not have exact models of the actual processing systems. This means that a number of runs might be required to achieve a significant target change, even if the system does not experience any disturbances. With active feed-forward control, the supervisory system is often asking the run-to-run controllers to readjust their targets on **every** wafer. Currently, such an aggressive strategy could be asking too much from an Equipment Cell. In fact, this type of control could violate our rule that the decision-making hierarchy is separated such that subordinate systems are treated as if they are in **steady-state**. A more reasonable strategy only adjusts multi-stage targets after the various Cell Controller's time constants have been given enough time to die out.

Figure 1-5: Run-to-Run Control Structure

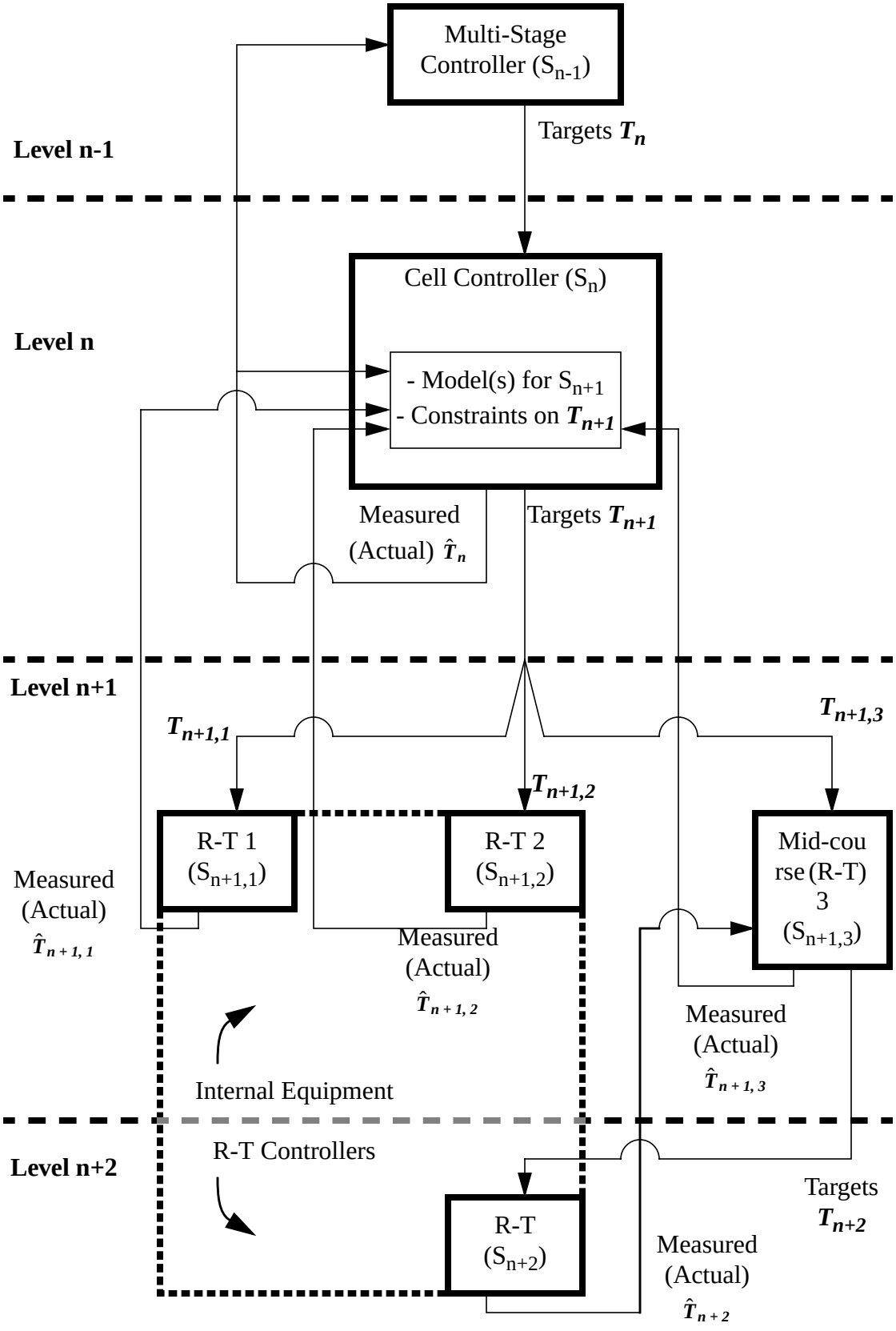
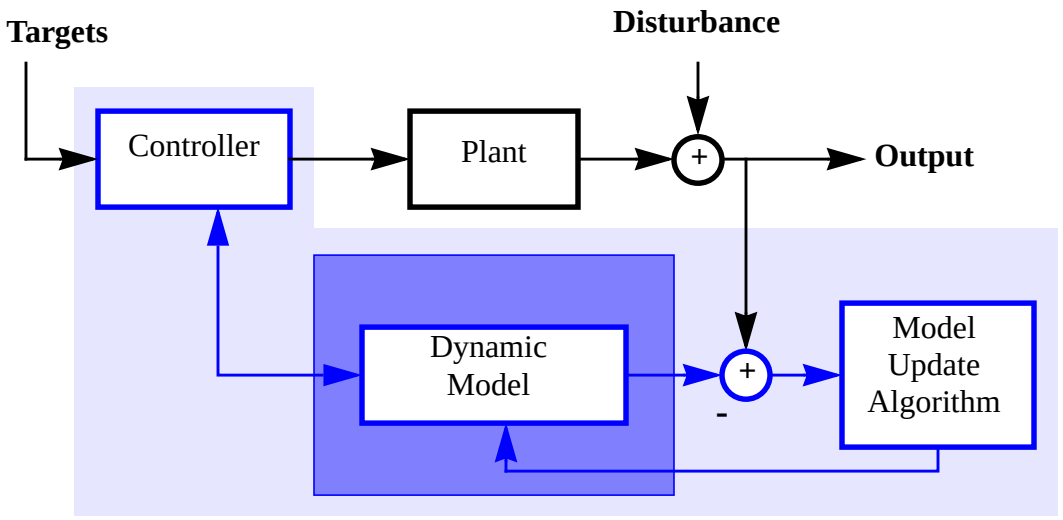


Figure 1-6 shows the generic structure for model-based run-to-run control, which has been used extensively. The controller utilizes a dynamic model of the processing system, which maps plant inputs to projected outputs. This model is used to select inputs that should achieve optimal outputs. When processing is complete and new measurements are available, the model is updated to account for any model error. In truth, the model update algorithm determines whether or not a model is truly dynamic. Various implementations of the controller, dynamic model, and model update algorithms have been explored, including an "internal model approach" [AZ97] [AZ98], "robust run by run control" [BP95] [BP97], "linear Exponentially Weighted Moving Average (EWMA) control" [Smi96], "neural network based EWMA" [SB97] [LSBH97], "probabilistic EWMA" [HKK98], and "Predictor Corrector Control (PCC)" [Smi96]. This thesis explores in detail the "linear EWMA" run-to-run controller and some of its variations.

Figure 1-6: Generic Model-Based Run-to-Run Control



Parallel Operation and Control

Generally a controller's model is maintained and used for a specific process on a spe-

cific processing chamber. Every chamber has its own model, which is only updated with measurements from wafers passing through that specific chamber. All of these models are needed to account for different possible drifts and disturbances between chambers, even identical chambers running the same processes on a single cluster tool. In fact, a single chamber might have a separate model for each recipe "class" that the chamber might run. This can occur when low-order models are used to approximate local system behavior in a number of different operating regimes. Each model is only valid within a certain set of processing conditions.

Recent work has explored various means of combining information from processes that share a common process chamber or recipe [Smi99] [Fen00]. This type of modeling and analysis broadly falls under the term "parallel operation and control." This term makes sense when considering the decision-making hierarchy that has been discussed above. Sharing information between two different recipes on the same chamber, or between two different chambers running the same process, requires the sharing information between two different Equipment Cells that are at the same level. Thus we are sharing "parallel" information, which breaks somewhat from the strict hierarchical structure described above.

Consider a Chemical Mechanical Polishing (CMP) process where two different types of product wafers are sent through the same polisher. Generally, each type of wafer requires a different polishing recipe to achieve the target output characteristics. Therefore, separate models are often made for each recipe. However, the CMP process is known to drift fairly strongly as the polishing pad ages. Clearly there should exist some kind of shared, device-independent model structure that can account for changes in polishing

capability for both types of product [Smi99]. For example, if alternating lots of two different product types are run on the same equipment, then the model for one process quickly becomes “out of date” as the pad is worn down by running the other process. The first wafer of each new lot would see a large disturbance because there was no feedback for the changes taking place while running the previous lot.

Parallel process model information can also be shared between two unique copies of the same type of equipment. In general, we can consider data from a number of similar sources. There must be a compromise between sharing global data and focussing on smaller, local data sets. Linear models using Bayesian estimation methods can be used by combining common (global) “hyperparameters” and local parameters [Fen00].

While these two types of parallel operation and control are interesting and significant extensions to basic run-to-run control scenarios, this thesis will not address them in detail.

1.4 Network and Information Security

With the widespread acceptance of the Internet and distributed computing have come many well-founded concerns about network security. In particular, malicious attackers often scan networks for weaknesses that will allow them to gain access to private information or enable them to anonymously execute applications on compromised computers. The classic technique of “security through obscurity” is becoming increasingly unreliable as computer crackers become more sophisticated. Distributed systems, such as the Cell Control architecture described above, provide services that are accessible through Internet protocols and therefore create added security risks.

The scope of this problem is well beyond that of this thesis, so this topic is superfi-

cially treated here. Clearly the Cell Control components should be secured within an Intranet behind a strong firewall, which provides the first line of defense. Additional access and information security may be provided through the use of secure communication protocols, such as Secure SHell (SSH) port forwarding [BS01], Secure Socket Layers (SSL) [Res00], and Virtual Private Network (VPN) [McD00] technologies. These types of encrypted communications are especially important when distributed system components interact over the global Internet, between two separate Intranets. Performance implications for using a distributed system must also be taken into consideration. Additionally, server side applications may be tested by third parties to discover vulnerabilities to attacks, such as buffer over-run techniques. The remainder of this thesis will not address these concerns.

1.5 Thesis Overview

Integrated Cell Control technology, including run-to-run control, provides a promising avenue for semiconductor fabrication facilities to make cheaper, higher performance devices. The integration of a recently developed in-line thickness sensor for epi deposition processes provides an excellent launching point for exploring aggressive run-to-run uniformity control. In-line sensor information can be augmented with detailed off-line sensor data, which are acquired on a lot-to-lot basis. A full development, analysis, and implementation of this scenario provides a number of promising extensions to current Cell Control and run-to-run control technologies.

Chapter 2 presents background material for a Cell Control testbed system. This includes a discussion of the epi deposition process and epi film sensors.

Chapter 3 presents some preliminary proof of concept work involving a relatively simple epi thickness control scenario. Background on time-based EWMA control is presented.

Chapter 4 extends the preliminary work from a Single-Input-Single-Output scenario to the simultaneous control of both epi thickness uniformity and resistivity uniformity. The issues of time-based control and mixed run-to-run and lot-to-lot feedback loops are dealt with in detail. (A reader who is interested in jumping directly to the experimental results should skip the next two chapters and move on to Chapter 7.)

Chapter 5 extends Chapter 4's brief treatment of solving techniques for time-based control. The iterative method used in this work is analyzed in detail, and solutions for both overdetermined and underdetermined systems are formulated.

Chapter 6 details the run-to-run controller interface and implementation used to support distributed Cell Control. Microsoft's Component Object Model (COM) Interface Definition Language (IDL) is used for specifying interactions between controller components.

Chapter 7 describes and analyzes experimental demonstrations of epi thickness and resistivity uniformity control. This includes an aggressive use of Multiple Response Surface (MRS) modeling based on a Design of Experiments (DOE), as well as an experimental design for testing the closed loop control system.

Finally, Chapter 8 concludes the thesis with an assessment of the technologies and implementations presented herein, as well as directions for future work.

Chapter 2

Cell Control Testbed: Epi Deposition

An appropriate testbed development system was chosen to explore and develop Cell Control architectures and technology. Based on the previous discussion of Equipment Cells, the following criteria are desired for an effective testbed:

- Processing Equipment
  - Utilizes a process step which is difficult to tune after a disturbance and/or drifts with equipment usage.
  - Can be manipulated and queried through a software interface via networked computers.
- Sensors
  - In-situ equipment and/or chamber state sensor(s) that can record activity of interest within the equipment during the actual processing.
  - In-situ and/or in-line wafer state sensor(s) which can directly assess the outgoing product quality.
  - Can be manipulated and queried through a software interface via networked computers.
- Infrastructure
  - Available high speed network for distributed infrastructure.

Most other Cell components are actually pieces of software, which are to be specified and implemented as part of the project. The list of prerequisites is basically comprised of the hardware components that should be available.

Such a system was constructed through a collaboration between a small sensor company (On-Line Technologies), an equipment vendor company (Applied Materials), and MIT. On-Line Technologies managed the project and provided a non-destructive, in-line

sensor that measures epitaxial silicon film thickness. (They are also hoping to develop a sensor that can measure both film resistivity and thickness at the same time, but that work is ongoing.) Also available are off-line measurements from On-Line Technologies' stand-alone thickness sensor and a resistivity sensor. Applied Materials provided technical support and processing time on a Centura semiconductor wafer fabrication system with an epi deposition chamber. Work for this thesis includes specification and implementation of integrated Equipment Cell capabilities for the system. The following sections provide a background for the equipment and sensor technologies used in the project.

2.1 Applied Materials Centura Epitaxial Deposition Tool

The Applied Materials Centura system with an epi deposition chamber provides an excellent testbed fabrication tool around which to build an Equipment Cell. The Centura can be controlled by a remote host using the Semiconductor Equipment Communications Standard (SECS) protocol through an RS-232 serial link [SEMI1][SEMI2]. This host computer can expose a network and/or other software interfaces for the other Cell components. Thus the Centura together with its host are considered to be the "Processing Tool" component of the Cell. Its interfaces should enable process recipe manipulation, equipment state data access, and posting of events and alarms as wafers pass through the system.

Effective implementation of cell control activities like run-to-run process control, fault detection, and fault diagnosis require a good working knowledge of the process involved. This includes a basic understanding of how various equipment settings affect the process conditions and how those changes influence the final product state. Complete understanding of the underlying physics is not required, but an intelligent methodology for selecting

equipment settings that are likely to significantly affect the equipment, process, or wafer states should be determined.

It is also important to consider what sensor information can be extracted from the system, which includes externally controlled sensors and sensors that are integrated into the equipment. This section focuses on equipment capabilities; external sensors are considered in later sections. Many tools, including the Centura, enable real-time status queries from sensors on the machine through the SECS communication link. There can be literally thousands of variables available for monitoring, so one must carefully consider which of these are likely to provide status information of interest.

The next sections discuss the epi process in general and provide details related to epi deposition using the Centura tool and epi film characterization.

2.1.1 The Epitaxial Deposition Process

Epitaxial deposition is a process whereby a thin single crystal film is grown on a single crystal substrate. A thorough treatment of this process is given by S. Wolf and R. Tauber [WT86]. This thesis utilizes lightly doped silicon epitaxial growth on heavily doped bare silicon wafers. This type of epitaxial deposition is used as the first processing step for many devices, and accurate control of the physical, chemical, and electrical properties of the film are important for creating functional, high performance integrated circuits. In particular, this type of epitaxy is used to improve the performance of bipolar and CMOS VLSI circuits. Bipolar devices that are fabricated using a properly optimized epi layer contain collector-substrate junctions with high breakdown voltages and collectors with low resistance. CMOS circuits fabricated with a lightly doped epi layer over a heavily doped substrate exhibit superior immunity to latch-up effects when power is applied.

Devices that utilize a silicon epi layer also benefit generally from increased control of the doping concentration in the epi, and an absence of unwanted oxygen and carbon.

The following sections discuss epi deposition fundamentals and the mechanisms for introducing dopants into epi films.

Epitaxial Film Growth

Epi film growth generally utilizes a Chemical Vapor Deposition (CVD) process. The epi deposition CVD process is comprised of the following steps:

1. Reactants are transported to the wafer surface.
   (Note: Some processes include reactions in the gas phase that generate film precursors before step 1.)
2. Reactants are adsorbed on the wafer surface.
3. A chemical reaction at the surface produces the film and reaction products.
4. Reaction products are desorbed from the surface.
5. Products are transported from the wafer surface.

Rigorous mathematical modeling of these steps has proven difficult, and a simplified model (the Grove model) involving only steps 1 and 3 is often used [WT86]. Despite its simplifications, this model describes many of the observed epi CVD characteristics. A steady state deposition rate of

$$V = \frac{k_s h_g}{k_s + h_g} \cdot \frac{C_T}{N_1} \cdot Y \qquad \textbf{(Eq 2-1)}$$

is predicted. V is the growth rate, k_s is the chemical surface reaction rate constant, h_g is the gas-phase mass transfer coefficient, C_T is the total number of molecules per unit volume in the gas, N_1 is the number of silicon atoms incorporated per unit volume in the film, and Y is the mole fraction of the reaction species in the gas. This model predicts a linear change

in growth rate with changes in *Y,* which matches observations for typical industrial growth conditions where *Y* is quite small ($Y < 0.1$).

Another important feature of the Grove model is the implication of a surface reaction limited regime and a mass transfer limited regime. When surface reaction limiting criteria are in effect ($k_s << h_g$) the growth rate operates independently of h_g and is given by

$$V = C_T \cdot k_s \cdot Y \quad . \qquad \textbf{(Eq 2-2)}$$

When mass transfer limiting criteria are in effect ($h_g << k_s$) the growth rate operates independently of k_s resulting in the following growth rate model:

$$V = C_T \cdot h_g \cdot Y \quad . \qquad \textbf{(Eq 2-3)}$$

For the purposes of process modeling and Cell Control one would like to discern what process conditions and factors contribute to the epi growth rate. Clearly C_T should be a function of pressure and temperature, while *Y* is a function of the various source gas concentrations and the relative flow rates of each. The chemical surface reaction rate and gas-phase mass transfer constants, however, require further analysis.

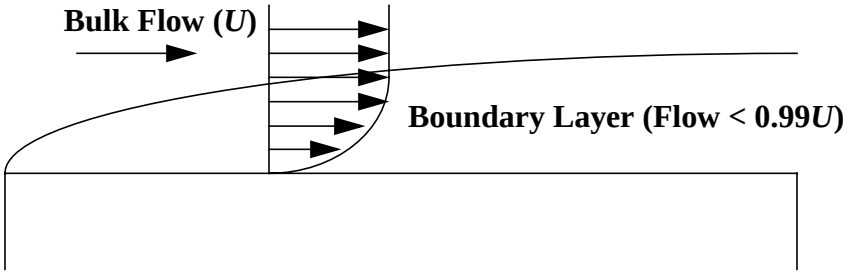
Assuming the reactions are of an Arrhenius type (meaning they are thermally activated), k_s is a function of a temperature independent frequency factor (k_0), the activation energy of the reaction (E_a), Boltzmann's constant (k), and the process temperature (*T*):

$$k_s = k_0 e^{-\left(\frac{E_a}{kT}\right)} \quad . \qquad \textbf{(Eq 2-4)}$$

Assuming the process is not in a mass transfer limited regime, the chemical surface reaction rate should mainly be affected by the process temperature. At high temperatures the reaction rate can increase until the mass flow of reactants limits the process. This simplified model does generally predict the experimental results for typical process conditions.

A number of models have been used for deriving gas-phase mass transfer coefficients. *Boundary layer theory* provides a reasonably accurate estimate of h_g for reactant gases flowing over a surface [WT86]. A boundary layer is defined to be the region above a surface where the drag due to that surface lowers the gas flow rate below 99% of the bulk gas flow rate (U). The theory enforces a zero velocity constraint at the surface and a gradual rise in flow rate until the bulk flow is reached. This results in a boundary layer that expands along the surface in the flow direction. Figure 2-1 graphically depicts the boundary layer structure.

Figure 2-1: Boundary Layer Diagram [WT86]



Boundary layer theory leads to a model of h_g which looks like

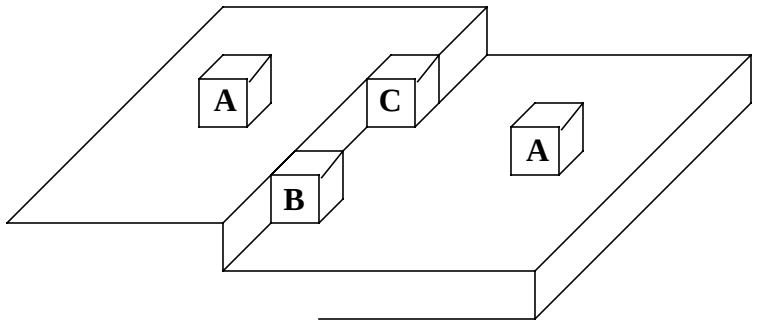
$$h_g = \frac{3D_g}{2L}\sqrt{Re_L} \quad , \quad Re_L = \frac{dUL}{\mu} \qquad \textbf{(Eq 2-5)}$$

where D_g is the diffusion coefficient for the active species, L is the length of the surface, and Re_L is the Reynolds number for the gas. The Reynolds number depends on the gas density (d), the bulk flow (U), the surface length (L), and the gas viscosity (μ). Assuming a fixed chemistry, h_g (and the deposition rate for a mass transfer limited process) is primarily driven by a square root dependence on U. Theory and experimental results display the growth rate's lack of dependence on temperature when in the mass flow limited regime.

Atomistic Model of Epitaxial Growth

Silicon epi films are believed to grow through adatom migration. Figure 2-2 graphically represents the growth process. Adatoms (A) attach to the silicon surface and migrate to kink positions at boundary steps between monolayers (B). A “corner” kink position (C) provides the most energetically favorable position for stable attachment (growth) because half of the silicon lattice bonds are linked to the crystal. Growth progresses via the lateral extension of the step layers.

Figure 2-2: Epi Film Growth Model [WT86]



A maximum single crystal growth rate can be found for a given temperature; deposition at higher rates results in polycrystaline films. It is believed that adatoms do not have sufficient time to migrate to kink positions when growth rates exceed a certain bound. Higher temperatures provide more energy for faster migration, and thus the maximum growth rate is found to increase with temperature.

Introducing Dopants Into Epitaxial Films

Epi growth provides the ability to precisely control film doping levels and thus the electrical characteristics (resistivity and conductivity type) of the deposited material. Dopants are typically introduced with the reaction gases through the use of their hydrides:

- Boron: Diborane (B_2H_6)
- Phosphorus: Phosphine (PH_3)
- Arsenic: Arsine (AsH_3)

The dopant delivery gases are usually heavily diluted with hydrogen to prevent dissociation of the dopant material [WT86].

There is currently no analytical model that accurately relates the ratio of the dopant concentration in the deposited film to the process conditions (such as dopant concentration in the reaction gases). Empirical solutions of the doping levels must be found for each set of deposition parameters. Fortunately the repeatability of the film doping concentration is very good for typical target doping concentrations and processing settings.

Most epi growth processes call for a lightly doped silicon layer to be added to a heavily doped wafer substrate. This scenario results in dopant flux into the deposited film via two mechanisms. First, there is direct solid state diffusion of dopant atoms from the substrate into the growing film, which tends to create a wide transition layer between the bulk substrate and the steady state deposition doping levels. This effect clearly varies with epi deposition time and temperature. A second path for substrate dopant to enter the epi film is through vapor phase autodoping. This is a mechanism whereby dopant from the backside and edges of the wafer evaporates into the gas, increasing the dopant concentration in the process chamber and subsequently in the growing film. Autodoping effects are typically noticeable as “tails” at the end of the diffused transition layer between substrate and final surface (intentional) doping levels. Solid state diffusion and autodoping thus impose restrictions on the minimum epi thickness for a given set of process conditions.

2.1.2 Applied Materials Centura Epitaxial Deposition Chamber

The Applied Materials Epi Centura Process Manual covers the basic design and operation of their deposition tool [AM95]. The Centura system is a cluster tool to which single wafer processing chambers are attached. A single Centura may have as many as three epi deposition chambers. The system can be configured for deposition at atmospheric pressure or reduced pressure, but switching between these two configurations requires some component changes within the processing chambers. The epi chamber is basically composed of two (upper and lower) quartz domes. Figures 2-3 and 2-4 contain cross-sectional side-view and top-down diagrams of the deposition chamber, respectively. Process control parameters primarily include set points for chamber pressure, gas flows, temperature, and processing time.

Figure 2-3: Cross-Sectional Side View of the Epi Deposition Chamber

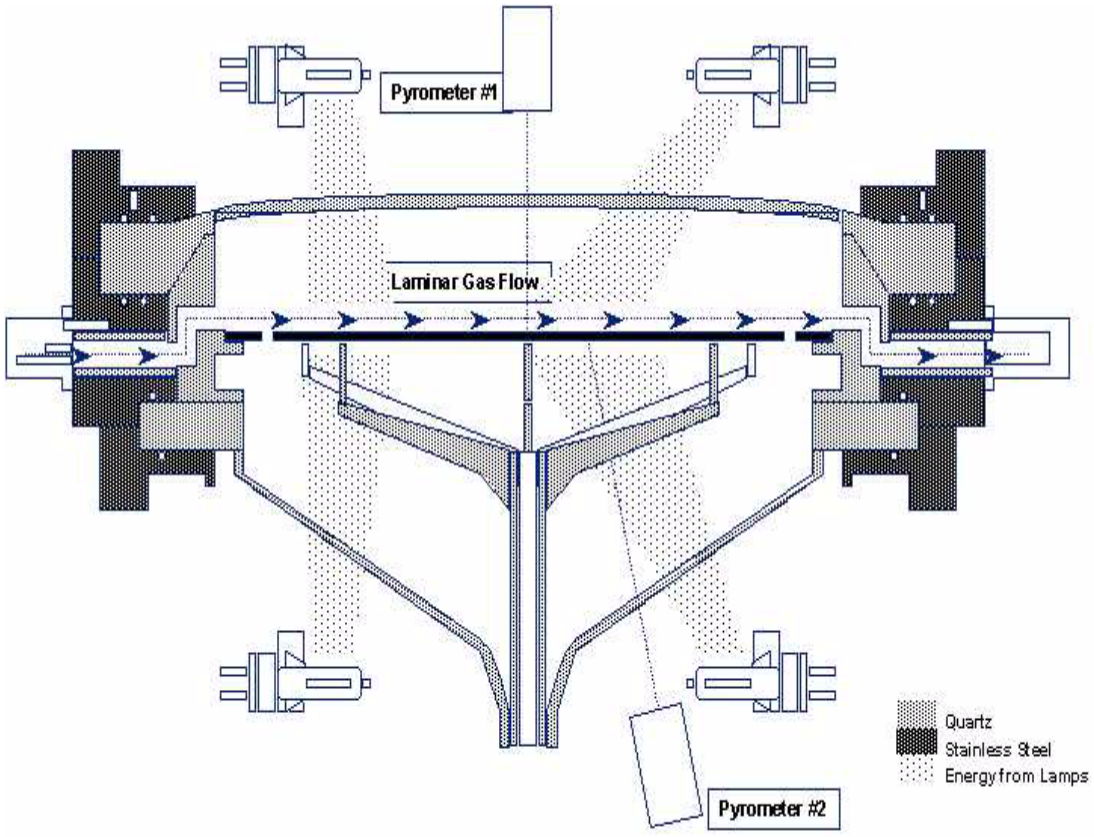
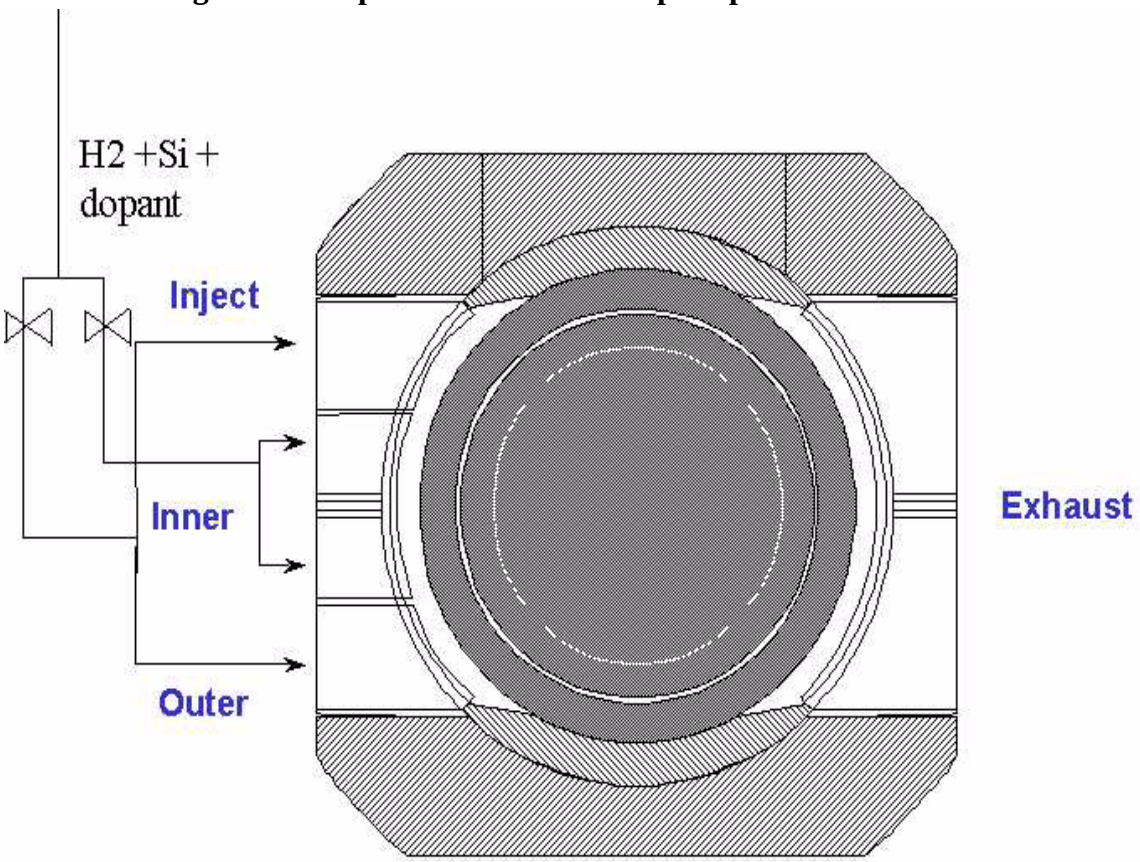


Figure 2-4: Top-Down View of the Epi Deposition Chamber



The following sections first describe the various process settings that are available, then present a standard process recipe and some of the most important factors for process control.

Process Settings

Chamber pressure and gas flows into the chamber are regulated by Mass Flow Controllers (MFC's), which are set via process recipe fields. Flows of silicon source gas, dopant gas, and carrier gas are all part of the recipe. As shown in Figure 2-4, gas injection lines are divided into two flows prior to entering the chamber, one line supplies the outer edges of the chamber while the other supplies the chamber's center. Control of the radial

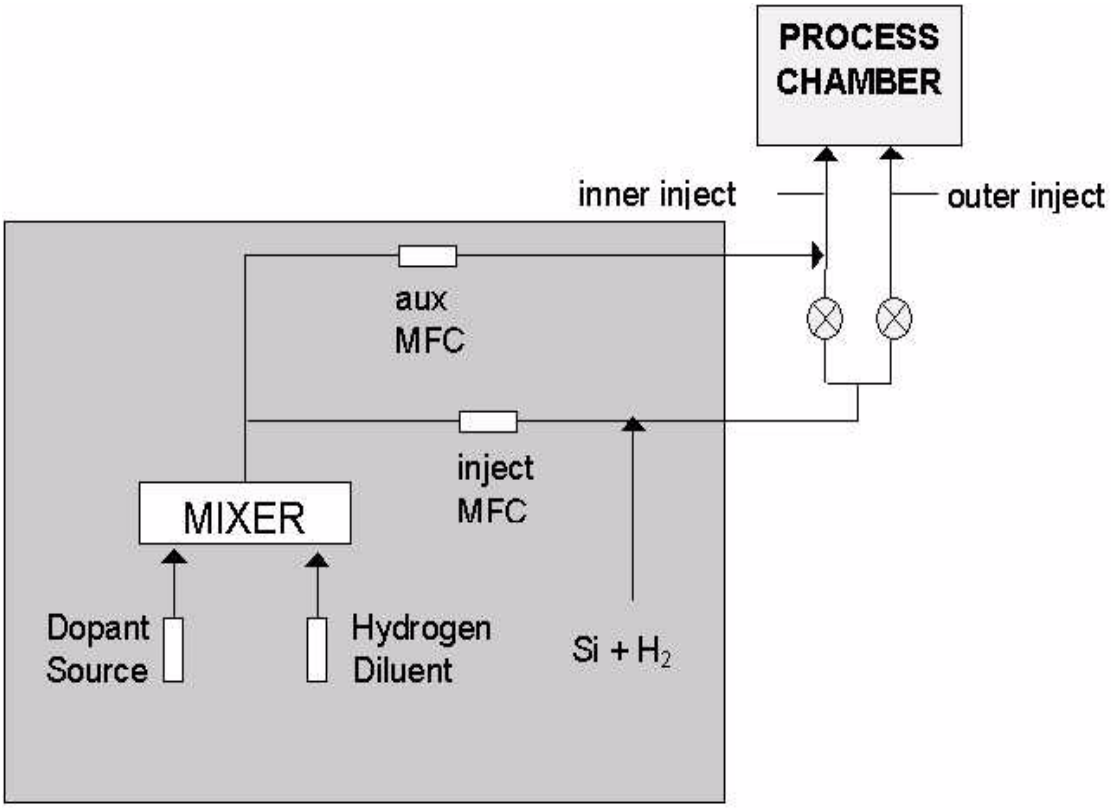
edge / center gas flow ratio is enabled through valves whose set points are configured externally. Originally these were Bellows Metering Valves (BMV's) that could be adjusted by hand [AM95]. The BMV's have since been replaced with an Accusett system that enables control of the two valves with a handheld unit that uses a serial port connection into the equipment. These valves cannot be controlled by either the equipment front panel interface or SECS communication protocols, but they are important for uniformity control of various film parameters. The lack of a programmable interface for these valves makes automated control of uniformity more difficult.

Introduction of dopant material is mainly provided by a system that uniformly injects the dopant gas across the whole chamber, as shown in Figure 2-5. The main dopant flow is mixed with the silicon source gases before the inner and outer gas flows are separated. Thus the dopants from this source are essentially uniformly mixed upon entry to the process chamber.

The system can operate in two different configurations, where the wafer is either "backsealed," or "non backsealed." The backsealed wafer process provides an airtight separation between the wafer's top surface and backside surface, thus shielding the wafer's backside from the process gases and preventing autodoping from the bulk silicon. This configuration typically yields fairly uniform resistivity profiles for the deposited film.

A non backsealed configuration allows autodoping from the wafer's backside, which typically yields a higher doping concentration near the edge of the wafer. To counteract this effect, radial dopant uniformity can be fine tuned with an auxiliary flow that is injected near the center of the wafer [AM95], as shown in Figure 2-5.

Figure 2-5: Gas Flows and Valves for the Epi Deposition Chamber



Chamber heating capability is provided by radiant lamp modules on the top and bottom of the chamber, as shown in Figure 2-3. Each module is divided into inner and outer rings of lamps. Chamber temperature is monitored via two optical pyrometers, one focussed near the bottom of the susceptor and one focussed on the center of the wafer's surface. Temperature setpoints are maintained by Proportional, Integral, Derivative (PID) controllers that use feedback from the optical pyrometers to vary the power delivered to the lamp modules. Alternatively, the lamp power can be set to a constant and the temperature allowed to vary accordingly.

The ratios of heating power directed to the lower and upper modules, upper inner and upper outer rings, and lower inner and lower outer rings are all part of the process recipe. These ratios are important for uniformity control because the internal temperature feed-

back loop can only maintain the temperature of at most two locations in the chamber, namely the locations where the pyrometers are focussed. While the temperature at the center of the wafer may be controlled to target, the radial temperature uniformity across the rest of the wafer is strongly affected by these ratio settings.

The Standard Epi Deposition Recipe

A basic epi process on a Centura tool generally includes the following steps:

1. The lamps uniformly raise the wafer temperature to the appropriate process temperature.
2. Native oxide on the wafer is reduced through a 30 second (or longer) bake in hydrogen gas. Silicon source and doping gases are purged at this time to stabilize their flows before deposition.
3. Process gases are flowed to the chamber through the deposit manifold.
4. Epi growth on the substrate surface and the susceptor continue during the deposition step. Products from the chemical reaction are exhausted from the chamber.
5. A short hydrogen gas purge clears the process gases from the chamber.
6. The wafer is cooled to a temperature at which it can be removed.
7. After removing the wafer an etch-back process is performed to clear material that was deposited on the chamber during the growth step. The frequency of the chamber clean varies, based on the deposition parameters. Generally the etch process is performed once every one to three epi wafers.

Process Control

Epi film resistivity, surface structures, and deposition rate are determined by many complex interactions between the physical and chemical characteristics of the process. Applied Materials specifically recommends tuning the following parameters:

- Substrate temperature
- Chamber pressure

- Silicon and dopant gas flows
- Carrier gas flow
- Gas flow patterns in the process chamber
- Susceptor temperature profile

Process variability and repeatability are often affected by the following:

- Temperature variation in the source gases
- Buildup due to inadequate etch-back (step 7 above) between deposition steps
- MFC variability when operating near the maximum or minimum flow limits

The goals for process control include rapid tune-up capability to optimize the process and minimize variability. Tuning parameters may then be utilized to continually optimize the process against drifting, wandering, or other deterministic disturbances.

2.1.3 Epitaxial Film Characterization

For developing any form of process monitoring or control one must consider the final product characteristics that are important for proper functionality of the product. In the case of epi films, there are a number of properties that must meet design specifications. In general these characteristics can be separated into physical and electrical properties. The following sections discuss these properties and common sensor techniques used to measure them.

Physical Properties

The primary physical properties of interest for epi layers are surface quality, crystallographic defects, and film thickness. General crystal and surface quality inspection are mainly performed with bright light illumination techniques. Examination of bright UV light that is reflected from the wafer surface (after epi growth) can detect the presence of pits, haze, scratches, particles, and spikes. The reflected light clearly indicates any depar-

tures from the desired smooth, highly reflective wafer surface. Microscopic examination of the film using 75-200X magnification can be used to detect a number of crystallographic defects. Laser scanners are also employed to find light scattering points on the surface, which indicate epi layer defects.

Epi film thickness is a critical feature to monitor and control for a number of reasons. As mentioned earlier, film thickness must achieve a certain minimum to ensure that autodoping and dopant diffusion effects are covered. Also, bipolar transistor device characteristics including breakdown voltage, junction capacitance, transistor gain, and AC performance all depend on the epi layer thickness. Thickness may be measured using a number of destructive and non-destructive techniques. Typically non-destructive methods are preferred and used because they enable measurements on actual product wafers. These sensors make use of the reflective characteristics of the epi film and underlying substrate. In particular, a technique called Fourier Transform Infrared (FTIR) Spectroscopy is used, which Section 2.2 will describe in detail.

Electrical Properties

Two of the most important epi layer electrical characteristics that influence the functionality of subsequently fabricated devices are minority carrier lifetimes and film resistivity. Minority carrier lifetime represents a measure of the heavy metal impurity concentration within the epi film. It is generally measured electrically through devices that are fabricated on the wafer. These measurements require further processing and are not made immediately after the epi deposition process, which introduces a large lag between deposition and feedback sensors.

Film resistivity (ohm-cm) is directly related to the concentration of dopant material in

the epi layer through the relation

$$\rho = \frac{1}{qn\mu} \quad , \qquad \textbf{(Eq 2-6)}$$

where q is the electronic charge (Coulombs), n is the doping concentration (atoms/cm^3), and μ is the carrier mobility (cm^2/V sec). Resistivity is controlled by changing the amount of dopant that is incorporated into the film. Various forms of four-point probe measurements may be used to measure the film resistivity. However, such techniques are not usually used on product wafers since a physical contact must be made with the wafer surface. Capacitance properties of the film may also be used to measure resistivity through a Schottky barrier or a p-n junction. Liquid mercury probes are available for this type of measurement; the mercury creates a Schottky diode with the substrate [WT86]. A mercury probe is available for use with our work.

2.2 On-Line Technologies Epi Film Sensor

Extracting information directly from an epi deposition tool itself can only reveal limited information, usually data that represent, or are correlated with, equipment state and possibly process state. However, the outgoing wafer state is usually the most important factor in judging product quality. In most cases taking such measurements requires the addition of third party sensors. Ideally the sensor can gather in-situ process state and wafer state measurements directly from the process chamber. However, this is often not possible due to physical or functional constraints on the system. A good alternative is the integration of an in-line wafer state sensor that can automatically receive and analyze wafers after they leave the process chamber. To enable automated Cell Control, this step becomes part of the normal wafer flow through the Equipment Cell.

To measure the characteristics of deposited epi layers, On-Line Technologies has developed an in-line sensor based on Fourier Transform Infrared (FTIR) technology [Cha98]. This sensor can be placed on the wafer cool down chamber of a Centura tool, enabling it to analyze epi film characteristics immediately after wafers leave the process chamber. It is also important to note that this setup does not require the addition of any new metrology stations or chambers; measurements are taken during the normal flow of wafers through the system. A computer is required to control the sensor and analyze its data, but the additional clean room requirements are minimal.

Epi film thickness and resistivity are among the most important physical and electrical characteristics one would like to measure. The On-Line Technologies' sensor has proven its ability to measure epi film thickness and can theoretically determine film resistivity. However, recent developments show that resistivities within the range of interest cannot be accurately found using the sensor. The following section provides a basic theoretical background for On-Line Technologies' FTIR thickness sensor.

2.2.1 FTIR Sensor

The On-Line Technologies' Fourier Transform Infrared sensor is essentially composed of an infrared (IR) source, a Michelson Interferometer, and an IR detector (see Figure 2-6). IR light (approximately 500-5000 wavenumbers) is used for epi thickness measurements because interfaces between silicon layers with different doping concentrations can reflect IR light, while other optical frequencies pass directly through the interfaces. Thus the air / lightly doped epi layer / heavily doped substrate form a stack of materials and interfaces that exhibit IR reflectance interference patterns. That is, light reflecting directly from the air / epi interface and light that passes through the interface and is subsequently

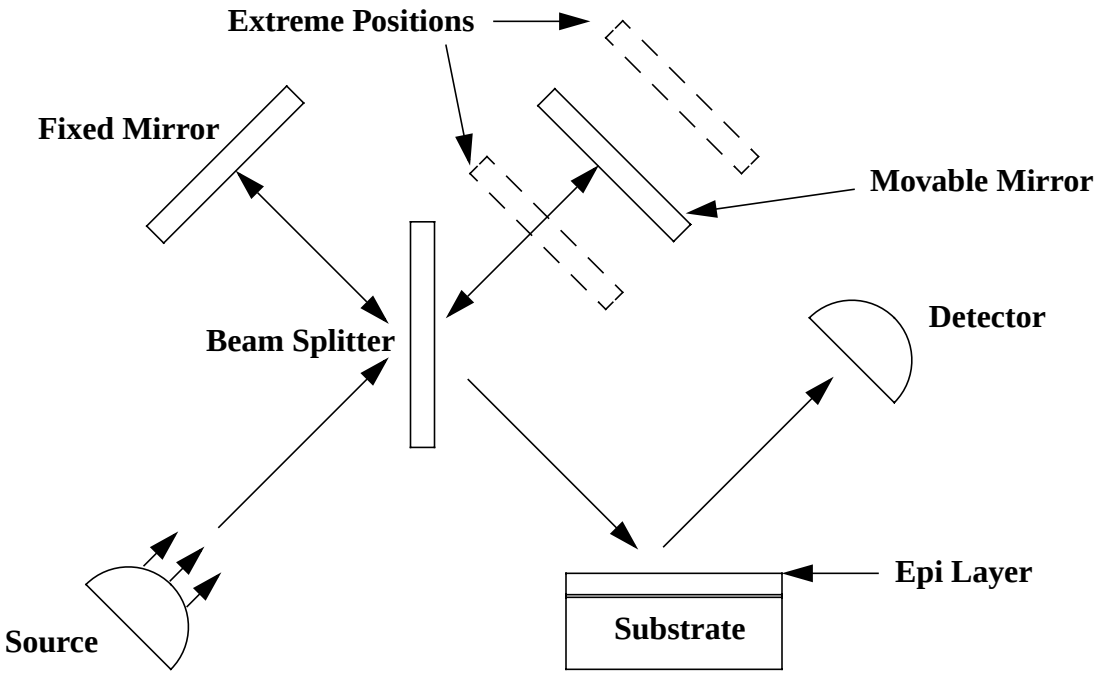
reflected from the epi / substrate interface interfere at the surface of the wafer. The type (constructive or destructive) and amount of interference depend on the wavelength of the light, the thickness of the epi layer, and the angle of incidence. Reflectance information from a range of wavelengths can uniquely determine the film thickness, as well as some other parameters.

Light from an IR source is directed through the interferometer, then reflected off the epi wafer surface into the detector. The detector provides a weighted sum of intensities over a continuous spectrum of light wavelengths. The Michelson interferometer's primary purpose is to modulate the IR light in such a way that a single detector can simultaneously provide a range of intensity measurements across the whole IR spectrum. The interferometer could be eliminated if a single detector could provide multiwavelength information directly. Since that is not the case, an interferometer is used to modulate the constant IR source intensities at different frequencies. The amplitude modulation frequencies are directly proportional to the frequencies of the original light source. Thus the weighted sum of intensities read by the detector becomes a sum of multi-frequency sinusoids instead of a sum of constants. A Fourier Transform of the detector's output as a function of time yields the linearly scaled frequency content of the incoming IR light.

The Michelson Interferometer essentially uses a beam-splitter, a fixed mirror, and a moving mirror to modulate the incoming IR light. Figure 2-6 displays a basic interferometer design in an epi sensor configuration. The source light hits the beam splitter at an angle of 45^{o}, sending half of the energy to the fixed mirror and half to the moving mirror. The light is reflected back and recombined at the beam-splitter, where half of the recombined light is directed out of the interferometer. The intensity of this light depends on the optical

path difference between the light which travelled to the fixed mirror and that which travelled to the movable mirror. Maximum additive interference is achieved when the movable mirror and the fixed mirror are the same optical distance from the beam splitter, and at integer multiples of wavelengths from this point. Assuming the mirror is moving at a constant velocity, one can imagine the amplitude of a single wavelength of light changing sinusoidally as the optical path varies between maximum constructive and destructive interference. The frequency of this amplitude modulation is linearly dependent on the frequency of the incoming light and the velocity of the moving mirror.

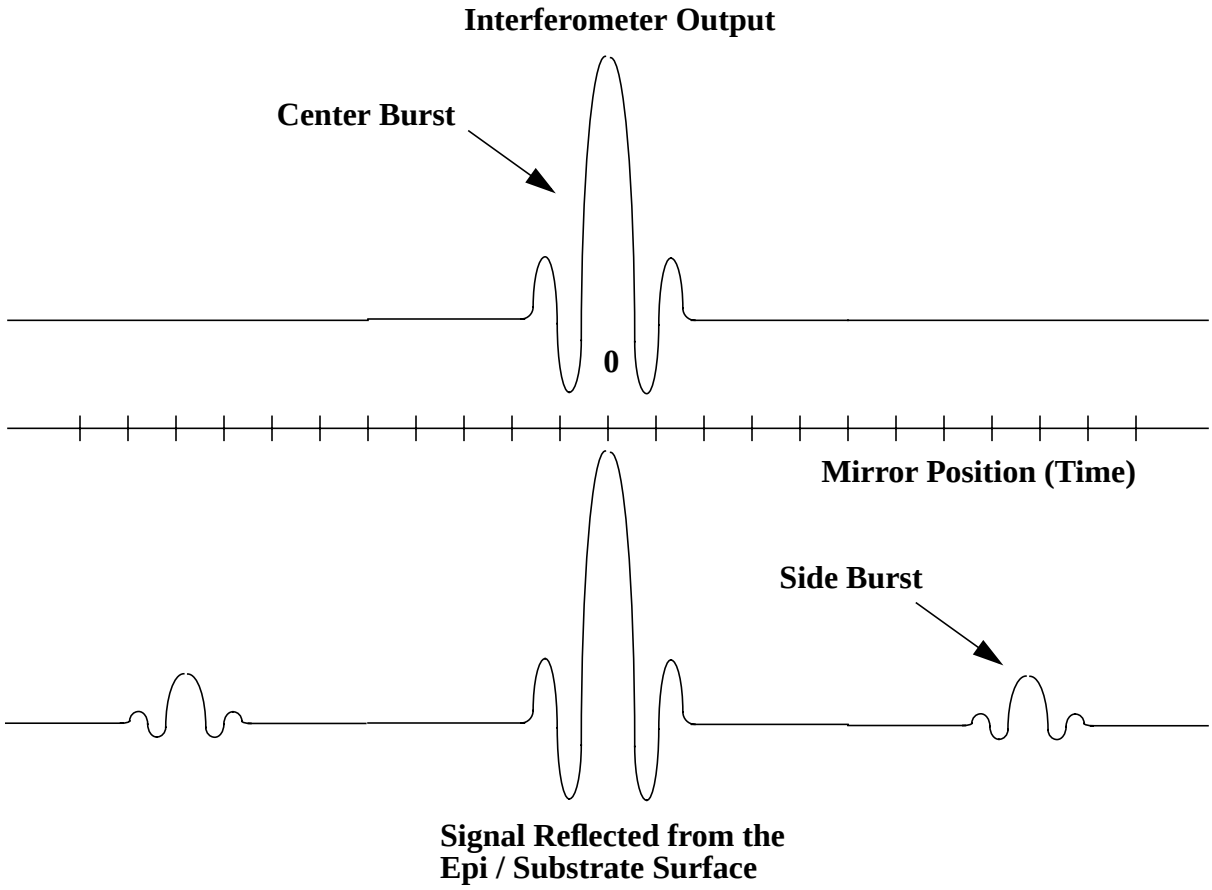
Figure 2-6: Michelson Interferometer in an Epi Sensor Configuration



The resulting detector intensity as a function of mirror position (or time, in the case of a constant mirror velocity) is called an interferogram (see Figure 2-7). The interferogram typically has a large peak where the optical path for the fixed and movable mirrors are equal, and damped, sinusoidal-like tails on either side. This large center burst results from

the additive interference of all frequencies in that single configuration, which can be defined as the origin of the mirror position (or time) axis. When the reflective epi layer and substrate are included in the optical path, as in the diagram, scaled copies of the center burst, called side bursts, are found at mirror positions corresponding to multiples of the optical path through the epi layer. This results from the added optical path lengths from the internal reflections within the epi layer. Typically only the first one or two of these sidebursts are recognizable due to the small amount of light that makes multiple journeys within the epi layer.

Figure 2-7: FTIR Interferograms



Until recently, most epi thickness measurements have been based on analysis of the

interferogram to determine the distance between the center burst and the side bursts, which directly translates into a film thickness estimate. This works well when the epi film layer is thick enough to adequately separate the center burst and the side bursts. As silicon processing technology has scaled down year after year, epi film thicknesses are often too thin for side burst measurement techniques. To overcome this limitation researchers have turned to model-based analyses of the reflected signal frequency content [Cha98][Che95]. The On-Line Technologies FTIR sensor makes use of this type of analysis, which enables accurate measurement of very thin epi layers. This sensor also provides some ability to measure doping profile information by modeling the transition layer as a multi-layer stack of graded doping levels.

2.3 Summary

This chapter provides essential background for understanding the epi deposition process and epi film characterization. The development of a successful Cell Control testbed requires a solid understanding of the underlying process and process assessment tools. The following chapters apply this knowledge and extend a number of run-to-run control techniques to demonstrate an effective application of Cell Control. Simple single-input-single-output tests first confirm the applicability of this technology, then a more advanced multi-input-multi-output system is developed.

Chapter 3

Basic Run-to-Run Control

Initial testing of integrated Cell technology focussed on a “proof of concept” strategy by developing the core equipment access and demonstrating simple integrated run-to-run control. Engineers at On-Line Technologies committed significant resources to integrating software control of their sensor and SECS communications with the Centura tool. The original in-line FTIR sensor system simply provided a single center point measurement of epi film thickness for wafers as they passed through the cool-down chamber. As an initial demonstration of Cell Control, it was decided to integrate the Applied Materials epi tool, On-Line Technologies in-line thickness sensor, and the MIT run-to-run control algorithm, for automated control of center point thickness by changing the epi deposition time [Ros98]. On-Line Technologies provided the sensor, the equipment communication software, and general software integration. Applied Materials supplied processing equipment, epi deposition expertise, and test wafers.

This chapter presents background and experimental results for the basic run-to-run control tests. Section 3.1 describes the control strategy used in this work, Section 3.2 provides the experimental design and results, and Section 3.3 summarizes the significance of the findings.

3.1 Control Strategy

There has been considerable research at MIT involving run-to-run control of semiconductor processes [SGHH91] [Ha93] [Yeb94] [SHI95] [Moy95] [Smi96]. The run-to-run

control software is based on an Exponentially Weighted Moving Average (EWMA) controller that was written as a C program by William Moyne as part of his Master's thesis at MIT [Moy95]. This multi-input-multi-output (MIMO) model-based controller uses a static linear input-output model of the following matrix form:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \qquad \textbf{(Eq 3-1)}$$

where $\boldsymbol{x}$ is the input vector and $\boldsymbol{y}$ is the output vector. Thus all that is needed to completely specify the model are the slope terms $\boldsymbol{A}$, and the offsets $\boldsymbol{b}$. Between each run the EWMA controller updates the offset terms ($\boldsymbol{b}$) based on the previous input-output values and some controller parameters. Based on the new model and a set of target outputs, the controller uses back-solving techniques to suggest a new set of inputs for the next run. Investigations concerning theoretical and practical issues of EWMA run-to-run control algorithms are found in a number of sources, including a detailed analysis in Taber Smith's MIT Master's thesis work involving run-to-run process control of Chemical Mechanical Polishing (CMP) [Smi96]. This controller has been used successfully for a number of semiconductor processes [Gow96] [SmiB96].

For deposition and etch/polish processes it often makes sense to model the deposition and etch rates as a function of process settings and calculate final thickness as a product of the rate and process time. This enables flexible, robust models that are valid over large variations in thickness targets. Such a modeling strategy also makes intuitive sense. Process tool settings induce a processing rate, which is expected to remain constant over a wide range of processing times.

For the initial epi deposition experiments, time was the only process setting to be updated; therefore the single-input-single-output (SISO) model looks like

$$r = b,\ y = r \cdot d; \qquad \textbf{(Eq 3-2)}$$

r is simply an EWMA updated estimate of the deposition rate (there is only one offset term, b, and no linear terms, $\boldsymbol{x}$), y is the final thickness, and d is the deposition time. We do not attempt to change the deposition rate; we merely model (track) the rate and try to achieve target by changing the deposition time. The solution for the optimal deposition time, given a target thickness (T) and the deposition rate estimate (r), is simply

$$d = \frac{T}{r}. \qquad \textbf{(Eq 3-3)}$$

3.2 Experimental Design and Results

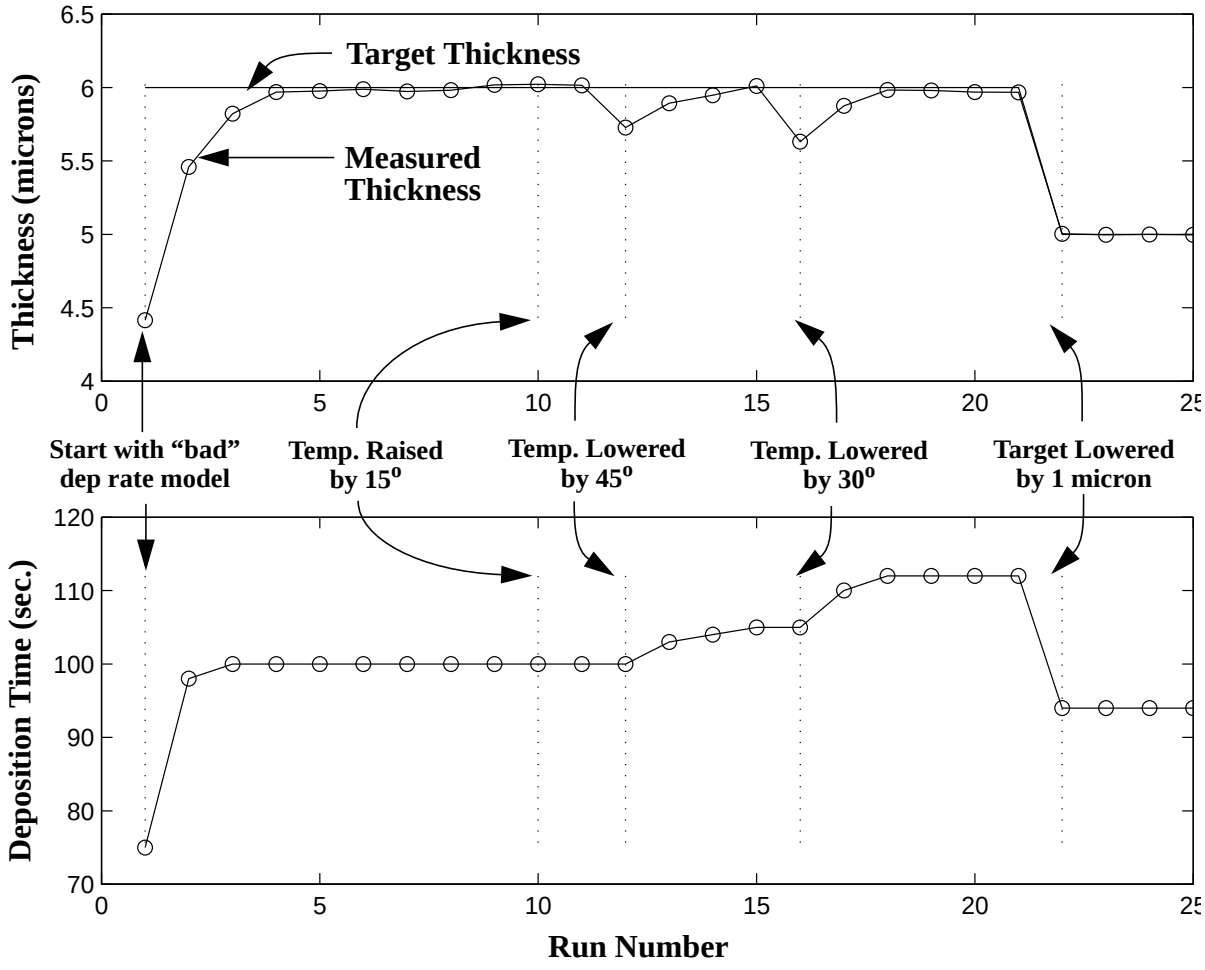
Since there are no slope terms, run-to-run control using only deposition time does not require response modeling of the epi deposition process. Automated feedback control is demonstrated experimentally as follows:

1. Start the run with an incorrect (or “bad”) estimate of the deposition rate.
   - The controller learns and reacts to the true deposition rate.
2. Modify the process in such a way that the deposition rate changes.
   - The controller learns and reacts to the new deposition rate.
3. Select a new thickness target.
   - If the time-based model is perfect, the controller will select a recipe to immediately hit the target. (The deposition rate has not changed, so the controller’s rate model and solution strategy for deposition time are still valid.)
   - If the time-based model is not perfect, then there will be an initial jump toward target, followed by an exponential approach to that target. (This will be the case if the deposition rate is not truly constant during the process. For example, there might be some transients at the start of the deposition process.)

The experiments were performed at Applied Materials, successfully demonstrating all of

these steps. Figure 3-1 shows the experimental results. At run 1 the controller's estimate of the deposition rate was considerably off. The subsequent drive to target is a classic demonstration of the EWMA controller's exponential learning rate. This learning rate is a function of the controller's EWMA weight, or forgetting factor. An EWMA weight of 0.7 was used for these experiments, meaning that the controller updated the deposition rate model after each run to account for 70% of the difference between the measured results and the previous model. This weight was chosen to avoid overreacting to system noise, while still providing quick responses to true disturbances.

Figure 3-1: Simple Automated Run-to-Run Control of Epi Thickness



The deposition temperature was modified to force a shift in the deposition rate. At run 10 the temperature was raised by 15^{o}C. Note that this caused a negligible increase in the deposition rate, which was not large enough change to require a control action. At runs 12 and 16 the temperature was lowered by 45^{o}C and 30^{o}C, respectively, each of which resulted in a significant drop in deposition rate. The EWMA controller increased the deposition time accordingly in both cases to drive the thickness back to target. Finally, at run 22 the target thickness was lowered by 1 micron, which was achieved in a single step. The correct solution to a new target indicates that the time-based model is appropriate for this system.

3.3 Summary

These experiments demonstrate the successful integration of a basic Equipment Cell, composed of a process tool, an in-line sensor, and a run-to-run controller. Even this simple scenario shows that the system can rapidly tune itself and keep the process on target despite process disturbances or drifts (which can be viewed as many small, systematic disturbances). This early success indicates that the time-based modeling strategy is effective, and paves the way to extend the system for full multi-objective run-to-run control.

Chapter 4

Multi-Objective Run-to-Run Control

Enhanced process tuning and target tracking capabilities are provided through the development and testing of a more complex run-to-run process control scenario. In particular, we consider sensor readings of film thickness and film resistivity at multiple sites across the wafer, making uniformity control possible. Detailed strategies for experimentation, data analysis, and control were developed early in the project to ensure success and maximize the utility of this research. A sequence of subtasks leading to good uniformity control is defined, where the structure of these tasks are highly coupled. This chapter documents the issues involved and proposes a detailed set of steps for initial uniformity control experiments.

There are three major stages to consider for model-based run-to-run control:

1. Selection of process parameters and Design of Experiments (DOE)
2. Process optimization
3. Run-to-run control setup and testing.

Parameter selection involves a careful analysis of the process to determine which outputs to monitor and control, as well as which inputs a controller will use to drive the outputs. A set of designed experiments is then selected and performed to provide input-output data. Next, the measurement data are analyzed to build models of the relationships between the inputs and outputs. These models are then used in conjunction with a cost or objective function to select inputs that minimize the cost. This leads to the next stage, where run-to-run control models and a model update strategy are created for the optimized pro-

cess. Finally, an integrated Cell is implemented and a series of runs are performed to test the response of the controller. The following three sections explore each of these stages, as well as their interactions, for the case of epitaxial deposition.

4.1 Selection of Process Parameters and Design of Experiments

The success of this work depends on gathering information about the epi deposition process and sensor capabilities. The following sections outline an ordered methodology for selecting process parameters of interest and gathering experimental data. Section 4.1.1 describes guidelines for determining which sensor outputs will be monitored. Section 4.1.2 covers the subsequent selection of equipment settings (inputs). Section 4.1.3 discusses the development of a DOE to capture relations between the inputs and outputs.

4.1.1 Selecting System Outputs

The primary goal of this work is to demonstrate run-to-run uniformity control of epi film thickness and resistivity via in-line and off-line sensor measurements. The in-line thickness measurements, which are taken while the wafer is in the cool-down chamber, represent the only outputs that are available for true run-to-run control (where control decisions are made after each and every wafer run). Off-line measurement stations can be used for additional lot-to-lot control.

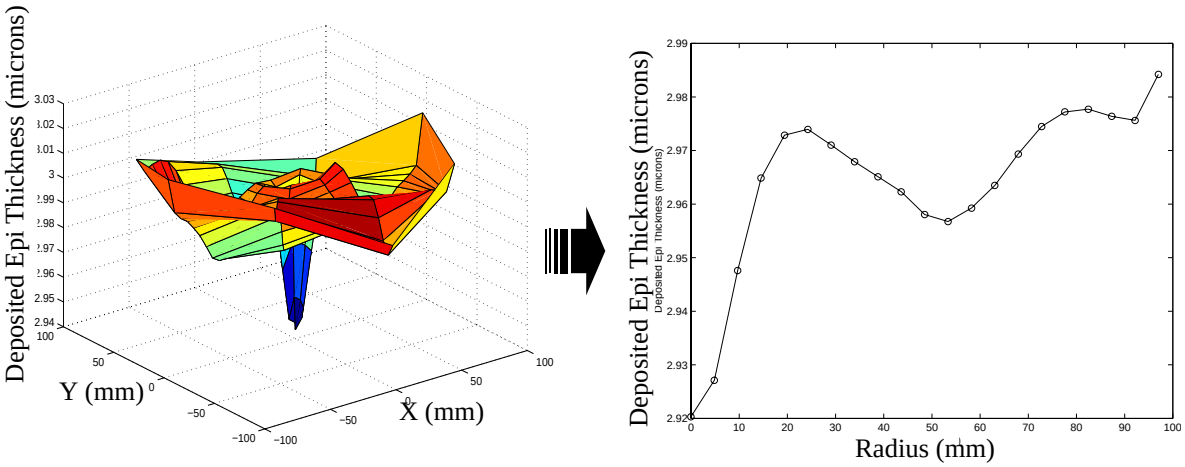
The system was enhanced after the SISO experiments, and multiple (N-site) thickness measurements are now taken along the wafer's diameter, instead of just a single center point. However, since the in-line sensor only provides a few measurements along a single diameter, it is also important to evaluate the ability of these N data points to approximate global thickness uniformity. To do this we also use an off-line thickness measurement sta-

tion that provides a densely sampled wafer map. While these measurements are not available for run-to-run control, they can be used in a lot-to-lot feedback loop to make run-to-run control more effective. They are also important for evaluating the system's ability to perform run-to-run control based on only the in-line measurements.

Resistivity readings are available from an off-line measurement station; there is currently no in-situ or in-line sensor. Thus resistivity must be controlled in a lot-to-lot fashion for this work.

Wafer rotation during epi deposition yields highly radially symmetric films, both in thickness and resistivity. Thus wafer map measurements may be compressed (averaged) into radial scans, as shown in Figure 4-1. Note that the wafer map is not perfectly radially symmetric. While wafer rotation increases radial symmetry, there are no equipment process settings that can control this symmetry. For this reason, we cannot directly control the full wafer map profiles. However, instead of simply controlling the grand averages of thickness and resistivity, a model-based controller can update process settings between runs to control their radial uniformities.

Figure 4-1: Compression: Wafer map to radial scan



Real-Time Sensor Traces

Another set of measurements is available for possible use in the Equipment Cell, namely the real-time traces of equipment sensor measurements. Referring back to Figure 1-5 (page 43), we can see that the measured (actual) outputs of the real-time and mid-course correction controllers should be available for the Cell Controller. There are optionally other equipment traces that may be useful, but certainly traces from the selected recipe set point controllers are among the most useful. (After all, they are important enough to be selected for control purposes.)

While these measurements are currently not utilized as part of the run-to-run control system, they should be available for possible extensions to the work presented here. These traces provide information about how well the various real-time controllers are performing. Assuming the real-time systems are adequately meeting their targets, we can safely assume that the recipe setpoints are constant throughout each wafer processing step. However, access to these data streams could be extremely useful for control, fault detection, and fault classification. The traces for system inputs (described in the next section) are also taken for the DOE runs. Sampled equipment status traces are extracted by periodically polling the Centura tool's SECS interface during the deposition process.

4.1.2 Selecting System Inputs

After determining the sensor readings (outputs) that will be monitored, the machine settings (inputs) that are most likely to affect those outputs must be selected. This does not imply that the exact relation between the inputs and outputs is known. There should be either empirical or theoretical evidence that changes in the inputs correlate with or induce changes in the outputs. Based on the experience of process engineers at Applied Materials,

process settings that are likely to affect the epi film thickness and resistivity were selected, along with “reasonable” ranges. These factors include those listed below in Table 4-1.

Table 4-1: Input factors and DOE ranges

Factor	Range
Deposit time	50 sec *
Deposit temperature	1090 - 1150 C
Dopant mixing ratio	20 - 80%
Dopant main flow	40 - 260 sccm
% Lower power	45 - 55%
% Inner power	40 - 60%
Dilutant (H_2) flow	30 - 60 slm
Trichlorosilane (TCS) flow	10 - 14 slm
Center gas flow valve (Accusett inner)	115 - 155 “unit”
Outer gas flow valve (Accusett outer)	60 - 100 “unit”

(* The use of time-based models means that a range of values does not need to be selected for deposit time. Deposited thickness is modeled as rate times time, which means that we must build models of deposition rate as a function of the other nine “non-time” inputs.) Additional inputs that affect thickness or resistivity could still be found. Also, more inputs could be considered if other outputs were selected for monitoring.

4.1.3 Developing a DOE

Once the system outputs and inputs are chosen, a set of designed experiments is selected and performed to evaluate the relations between the inputs and outputs. First, “reasonable” operating ranges for the inputs are selected. Specifically, upper and lower bounds are determined, which define a “hyper-box” within the input space. These bounds are based on a number of criteria, including the following:

- Machine limitations
- Safety limitations
- Undesirable effects on uncontrolled outputs
- Settings beyond which the resulting cost function is known to increase

This last criterion implies some knowledge from previous experimentation and about the process optimization methodology. Ideally a set of optimal outputs can be achieved with machine settings that are near the center of the bounded input space. Table 4-1 also lists the operating ranges for the DOE.

The input space is also selected to balance an important trade-off between model complexity and flexibility. For the epi deposition process, an input space that is too wide will probably contain highly non-linear relations between the inputs and outputs, which would require a dense sampling of the input space. An input space that is too small will probably not provide a large enough “operating space” to be useful, and is also likely to miss any true optimal operating points. The input space for these experiments was selected such that linear models would likely capture most of the system behavior, yet still provide enough room to perform control.

There are a number of ways to select such an appropriate input space. For this work, an expert process engineer was available to suggest ranges that made sense for the given equipment. Otherwise, a small number of screening experiments should be run first to aid in the selection process. For example, a few runs might be performed while varying a single input at a time. These experiments provide information about the first order effects from each input factor, and help determine ranges where linear approximations will be sufficient. There must be at least three runs per input (preferably more), which means the number of screening experiments increases linearly with the number of inputs.

Next, a set of experimental operating points was selected from within the constrained input space. The exact configuration of these points is at the heart of experimental design methodologies and depends upon a number of factors. Some knowledge of the relative non-linearities between the input-output mapping should be used to determine how many levels of input settings will be explored. The number of levels determines how finely the input space will be sliced; generally two or more levels are used with DOEs. The intersection of these slices form lattice points, from which a subset is usually selected for experimental runs. There are a number of designs to consider, such as Box-Behnken and Central Composite [Mon91].

A Central Composite design is selected for this work, as it is both commonly used and effective for response surface modeling. Noting that deposit time is used as a multiplicative factor on rate estimates, there are nine factors under consideration for modeling the deposition rate and resistivity. The Central Composite DOE calls for a two level full-factorial design for all of the input factors, plus axial and center points. This means that a nine factor DOE requires over 512 experiments. Ideally all of these experiments could be performed, but processing that many wafers is probably unnecessary. A one-quarter fractional factorial design [Mon91] is used in this work, which can be augmented later with more experiments, if needed.

4.2 Process Optimization

The process optimization methodology is based on the input and output parameters that have been selected. In the most generic case, a cost or objective function is created in terms of the outputs (and possibly the inputs), which defines the relative quality of a set of outputs (and inputs). Specifically, a lower cost implies a better set of process parameters,

meaning that a globally minimum cost is the desired operating point.

The following sections describe the strategies and issues involved in process optimization for the multi-input-multi-output (MIMO) epi control problem. There are three main steps involved: selection of an appropriate cost function (Section 4.2.1), empirically modeling the cost function in terms of the process inputs (Section 4.2.2), and performing a constrained optimization over the cost model to find the inputs that yield the lowest cost (Section 4.2.3).

4.2.1 Cost Functions

Often the cost is based primarily on the outputs and only slightly, if at all, on the inputs. In the epi control problem, a minimum cost solution should occur when all measured sites have achieved the target thickness and resistivity, and all of the inputs are within the valid operating ranges. A constrained optimization capability eliminates the need to worry about input ranges, so the cost function is often based solely on the output optimality criteria.

Cost functions take many forms, but a weighted sum of squared errors from the output targets is common and often analytically meaningful. A general form for this is

$$Cost = \sum_i [W_i(T_i - o_i)]^2 \quad . \qquad \textbf{(Eq 4-1)}$$

This function sums over all outputs (i), the product of a weighting term (W_i^2) and the squared difference between the output target (T_i) and the actual (measured or predicted) output (o_i). The weighting terms scale the importance of meeting various output targets. The weights could reflect trade-offs between comparing outputs of different types, or trade-offs between the relative importance of outputs that are the same type.

There are many possible cost functions for the epi control problem, but

$$Cost = \sum_i [W_{ti}(T_{ti} - t_i)]^2 + \sum_j [W_{\rho j}(T_{\rho j} - \rho_j)]^2 \qquad \textbf{(Eq 4-2)}$$

is a convenient and meaningful choice. T_{ti} and $T_{\rho j}$ are the target thicknesses and target resistivities, respectively for each of the thickness (i) and resistivity (j) measurement sites, while t_i and ρ_j are the measured thickness and resistivity values, respectively. W_{ti} and $W_{\rho j}$ are relative weighting constants between the individual thickness and resistivity site measurements. The weights encapsulate the trade-offs between achieving target thickness uniformity versus achieving target resistivity uniformity. Any differences between the number of thickness and resistivity sites will affect the weights, as well as the relative magnitudes of target thicknesses and target resistivities.

The weights are also likely to be affected by the spatial organization of the outputs. There is a weighting structure based on the relative area that each output site represents. For the epi control problem, run-to-run control makes use of data from points along a radius. Clearly, sites near the outer edge represent a larger area than those near the center. Therefore it makes sense to associate a higher cost for these sites when they deviate from target.

4.2.2 Cost Function Modeling

The optimization process makes use of a function (model) of the cost in terms of the inputs. This function is back-solved to find a minimum cost solution. The DOE data are used to build a model of the cost as a function of the inputs. There are many ways to do this, especially when the system outputs include multiple site measurements. One approach involves empirically modeling the cost or measurement statistics as a direct

function of the DOE input settings. This is known as a Single Response Surface (SRS) methodology, which could simply generate the cost function as

$$Cost = c(\boldsymbol{x}), \qquad \textbf{(Eq 4-3)}$$

where $\boldsymbol{x}$ is the vector of process settings. That is, the model does not directly receive any information from individual sites; the cost is calculated for each DOE run and modeled directly in terms of the inputs. In general this overly simplified SRS cost function model is not practical, as it does not provide any means for adjusting output targets.

A more reasonable SRS approach is to consider modeling the average thickness, $t(\boldsymbol{x})$, and average resistivity, $\rho(\boldsymbol{x})$, then calculating the cost function in terms of these two models, as in

$$Cost = [W_t(T_t - t(\boldsymbol{x}))]^2 + [W_\rho(T_\rho - \rho(\boldsymbol{x}))]^2. \qquad \textbf{(Eq 4-4)}$$

To include a penalty for (and control of) nonuniformity, this idea can be expanded slightly to include models of the standard deviations for both thickness, $\sigma_t(\boldsymbol{x})$, and resistivity, $\sigma_\rho(\boldsymbol{x})$. The resulting cost function has four process models that are combined as follows:

$$Cost = ([W_{t\mu}(T_t - t(\boldsymbol{x}))]^2 + [W_{t\sigma}(\sigma_t(\boldsymbol{x}))]^2) + ([W_{\rho\mu}(T_{\rho\mu} - \rho(\boldsymbol{x}))]^2 + [W_{\rho\sigma}(\sigma_\rho(\boldsymbol{x}))]^2). \qquad \textbf{(Eq 4-5)}$$

While this might not technically appear to be a “single response surface” cost function model, consider that the four underlying models are individual surfaces that represent distinct statistical metrics. SRS models are often used to monitor and control the statistics of the underlying process, rather than directly controlling the individual outputs of the process itself.

A Multiple Response Surface (MRS) methodology builds individual models of each site, then uses the cost function's form or expression to combine the outputs of the models.

The weighted MRS cost function for the epi process becomes

$$Cost = \sum_i [W_{ti}(T_{ti} - t_i(\boldsymbol{x}))]^2 + \sum_j [W_{\rho j}(T_{\rho j} - \rho_j(\boldsymbol{x}))]^2 . \quad \textbf{(Eq 4-6)}$$

There are a number of issues involved in choosing between SRS and an MRS strategies. The following sections consider the implications of model complexity, the transition from DOE based optimization to a run-to-run control scenario, the modeling concerns of time-based inputs, and the effects of site noise on the cost function models. Finally, this section addresses the problem of selecting an appropriate complexity level for the model.

Model Complexity

MRS models typically enable low dimension models of the sites while achieving a much higher order model of the cost [GS93]. Recent work analyzes the use of SRS vs. MRS techniques to determine the effects of noise on variance models that are based on data from multiple sites and experimental replicates [SGBS99]. Assuming statistically Identical and Independently Distributed (IID) site noise, SRS models give unbiased estimates of underlying spatial variation plus the site noise, while MRS models give biased estimates of the spatial variation alone [SGBS99]. However, the bias of the MRS model must be taken into account, and one must be careful when trying to compare the two types of models. Work to date has only considered this effect for replicates of single process settings, not for modeling variance as a function of multiple process settings.

Transition to Run-to-Run Control

One major concern for selecting a modeling strategy is the transition from the initial “tune-in” optimization step to a run-to-run control strategy. Ideally, the process models generated from DOE data are subsequently optimized and translated into models that fit

into the run-to-run control framework. Also, an initial optimization of the run-to-run controller should consequently arrive at the same optimal process settings as were found in the DOE optimization. These criteria pose a significant barrier to using SRS modeling techniques.

Consider a properly optimized process where inputs have been found such that the cost function yields a global minimum. Locally the cost function (and usually any models of variance, if applicable) have slopes that are zero in all directions, with respect to the input space. Assume this model is then implemented in a run-to-run controller, and is used with a process whose cost due to variance is subsequently found to be increasing with consecutive runs. The rising cost probably indicates that the process is drifting. However, because the model currently indicates that the process is operating at a minimum variance, the controller cannot determine which input trajectory will help compensate for the non-uniformity. Important spatial information has been lost by combining all of the data from the sites into variance measurements.

Another problem with the SRS modeling strategy is the difficulty in building a controller around a complex model. Note that using a model that contains a local minimum requires at least a second order function. The current MIT algorithm-based controller uses a linear model of the output as a function of the input. MRS models can use lower order models and thus enable a better fit for linear run-to-run control models. Also, a linearization at the minimum of a (cost or variance) function would result in a slope matrix of all zeros, which cannot be used for control.

This of course assumes that the global minimum is within the interior of the input space, and not beyond the boundaries of the DOE. Even if the latter situation occurs, the

high dimensionality of the SRS models could make linearization difficult. Also, it is not desirable to initiate run-to-run control when the optimal operating point is pinned against the bounds of the input space. If this is the case, then there is little control to be done. In fact, inputs which are strongly "pushed" against a bound might be eliminated from the control strategy. It is known that moving this setting away from the bound must increase the cost function, so it is virtually impossible for this input to go anywhere. The goal is to find optimal settings that are inside the operating ranges.

Additionally, since the run-to-run models are based on a linearization of the "tune-in" models, it is often advisable to perform a second DOE which has tighter bounds and is centered on the optimal settings. This enables a better linearization and helps verify the accuracy of the original models.

Time-Based Control

Another important modeling issue is the effect of time-based settings. For thickness measurements it is often more effective to model the etch or deposition rate, then multiply by time to arrive at final thicknesses. This is better than trying to include time as a "regular" input and create a (linear) model that includes time as an additive term rather than the multiplicative factor it actually is. This added twist results in a cost function of the form

$$Cost = \sum_i [W_{ti}(T_{ti} - (r_i(\boldsymbol{x}) \cdot d))]^2 + \sum_j [W_{\rho j}(T_{\rho j} - \rho_j(\boldsymbol{x}))]^2, \qquad \textbf{(Eq 4-7)}$$

where $r_i(\boldsymbol{x})$ is the model of the i^{th} site's deposition rate as a function of the inputs, and d is the deposition time. It is assumed that deposition time does not affect resistivity in the same manner. Within reasonable ranges, the final epi resistivity will be essentially independent of the film thickness, and thus independent of the deposition time. This cost func-

tion model should achieve a fairly accurate representation of the process and will reasonably map into a linear model-based run-to-run control strategy. The complications involved with using time-based inputs for run-to-run control will be introduced in Section 4.3.

Site Noise

Assuming an MRS modeling strategy is selected, the effects of site noise must be considered. Theoretical investigations of MRS modeling have only been performed for data involving replications of a single set of process parameters with IID site noise. Under these conditions the expected variance (cost) of the outputs from the MRS models is

$$E[Var_{MRS}] = Var(f_n) + \frac{1}{M} Var(w_{m,n}), \qquad \textbf{(Eq 4-8)}$$

where N is the number of sites and M is the number of replicates. $Var(f_n)$ is the variance of the underlying function and $Var(w_{m,n})$ is the variance of the site noise [SGBS98]. This has important implications when comparing the variance between two different process settings. If there are more replicates at one setting than another, the $1/M$ factor that multiplies the variance from the noise can lead to an incorrect assessment. Consider a typical DOE where a number of replicates are performed at the center of the input space and only few or single replicates are performed throughout the rest of the space. This strategy will bias the resulting variance measurements in favor of the center point, when directly comparing the results of the center with those of other design points that have few replicates.

Use of a well-constructed modeling strategy will help to avoid this issue. First, models of the site outputs as a function of the inputs are fit, then the variance (cost) of a given point in the input space can be found through the model outputs, rather than actual site

data, which are compromised with site noise. One can view this methodology as combining all of the results from the DOE as replicates of a single process (epi deposition). Building the site models helps to make use of all the data to filter noise and put all points of the input space on more equal ground with respect to the site noise.

In fact, if one assumes that the site noise is IID across the wafer and the design points, a more formalized analysis of MRS modeling could be considered. While this assumption is probably not exactly accurate, it is likely to be approximately true. A thorough theoretical treatment of noise effects on MRS modeling across multiple process settings is an open area for further development.

Model Complexity Selection

DOE data can be used to help select the model complexity for both SRS and MRS modeling strategies. Replicates at the DOE center, and perhaps a few other operating points, can be used to estimate the noise of the system. Various models can be fit to the full data set, and the models that yield noise values (variation between the model outputs and the experimental outputs at the design points) that are consistent with the replicate-based noise estimates should be selected as likely candidates. Possible models include polynomials of varying order and neural networks with varying numbers of nodes and/or hidden layers. Any of these models could then be linearized at an optimal point for subsequent use with a run-to-run control strategy.

4.2.3 Cost Function Optimization

Based on the arguments from the previous section, the MRS time-based cost function modeling structure is used (Eq 4-7). Using this model of the cost function in terms of the

process settings, a set of inputs that yield a minimum cost are found through a constrained optimization. While the run-to-run controller will use linear models of deposition rate and resistivity, the DOE-based process optimization could use higher order models. A general nonlinear optimization process such as this is provided by a number of software packages. In particular, we use MATLAB [Mat97, Version 5.3] for modeling and optimization. The details of constrained optimization are beyond the scope of this paper and are expected to be automated through the use of a software package. This applies to the initial process optimization step, but the integrated run-to-run controller contains its own constrained optimization routines for the linearized, time-based models.

4.3 Run-to-Run Control

Having selected process parameters, performed a DOE, and created and optimized a set of site models, the run-to-run controller is initialized to begin processing. As with previous run-to-run control work at MIT [Smi96], linear process models whose offsets are updated in an EWMA manner are used in the controller. However, the deposited thickness site models have been modified to take advantage of the time-based structure described earlier.

Because this controller is used for our run-to-run control experiments, it is worthwhile to note again that the run-to-run process models might require a DOE from a smaller, more linear region of the input space. This will be the case if the original DOE space requires highly non-linear models.

Even before the DOE is specified and analyzed, the run-to-run controller's cost function and model update procedure is considered. It is important to match the cost function and optimization strategies between initial DOE-based optimizations and the subsequent

run-to-run control optimizations. Section 4.3.1 looks at cost function optimization for the run-to-run controller, Section 4.3.2 discusses the controller's model update strategy, and Section 4.3.3 describes the issues involved in combining lot-to-lot and run-to-run feedback data.

4.3.1 Cost Function Optimization

The run-to-run controller's cost function modeling structure should be identical to that shown in Section 4.2.2 (Eq 4-7):

$$Cost = \sum_i [W_{ti}(T_{ti} - (r_i(\boldsymbol{x}) \cdot d))]^2 + \sum_j [W_{\rho j}(T_{\rho j} - \rho_j(\boldsymbol{x}))]^2 . \qquad \textbf{(Eq 4-9)}$$

It is important to carry this cost function into the run-to-run controller to ensure a proper "hand-off" from the DOE-based process optimization. However, the site models for the run-to-run controller, $r_i(\boldsymbol{x})$ and $\rho_j(\boldsymbol{x})$, are restricted to be (incrementally) linear, as in

$$r_i(\boldsymbol{x}) = \boldsymbol{A_i x} + b_i \text{ and } \rho_j(\boldsymbol{x}) = \boldsymbol{A_j x} + b_j . \qquad \textbf{(Eq 4-10)}$$

This restriction implies that the run-to-run site models are linearized versions of the DOE optimization models. Ideally the linear models capture (nearly) all of the important behavior from the DOE models. If they fit well, linear models may be generated directly from the DOE data. If this is not the case, the non-linear models can be linearized at an optimal operating point, and the allowable input space is truncated such that the linear models are a "good enough" fit for that region, as defined by the desired stability criteria. (This is the case where a follow-up DOE is recommended.)

For each run, the controller must be able to select inputs $\boldsymbol{x}$ and d that optimize the cost function, given the current site models. Software routines have existed in the MIT controller for quite some time that can efficiently minimize a weighted sum-of-squared-error cost

function based on linear site models. However, the inclusion of time-based models implies that these routines cannot be directly applied to the optimization problem.

The optimization is currently solved by holding one set of inputs (either $\boldsymbol{x}$ or d) constant, then solving for the other. When the deposition time, d, is held constant, the "normal" run-to-run control solver may be used to find the process settings, $\boldsymbol{x}$. The weighting and target terms are first adjusted to account for the time factor (d). Specifically,

$$\hat{W}_{ti} = d \cdot W_{ti}, \text{ and } \hat{T}_{ti} = T_{ti} / d. \qquad \textbf{(Eq 4-11)}$$

Once the vector $\boldsymbol{x}$ is chosen, the optimal deposition time may be found by minimizing the cost function with respect to d. In this second stage of the optimization, the inputs ($\boldsymbol{x}$), and thus the rates ($r_i(\boldsymbol{x})$) are known and can be treated as constants. This leads to a straightforward solution for d,

$$d = \frac{\sum_{i=1}^{N} W_{ti}^2 \cdot T_{ti} \cdot r_i}{\sum_{i=1}^{N} W_{ti}^2 \cdot r_i^2}. \qquad \textbf{(Eq 4-12)}$$

Note that the optimal deposition time is only a function of terms corresponding to deposition rate outputs. Intuitively, resistivity models should not affect the selection of a deposition time when all other inputs are held constant.

In fact, a complete non-linear optimization scheme is achieved by iterating between the solution of optimal non-time inputs $\boldsymbol{x}$ (while holding constant the deposition time), and the solution of an optimal deposition time d (while holding constant the non-time inputs). Essentially this strategy guarantees that each step decreases the cost function toward a minimum, which could be local or global. A minimum will be found unless process setting bounds are hit first.

A single-step solution to the time-based optimization problem is available for the special case where there is only one time factor, and **all** outputs are thicknesses that use linear rate models and are multiplied by that unique time factor. While the epi deposition scenario does contain a single time factor, the resistivity outputs are not rate-based and are therefore not multiplied by time. Thus the multi-objective epi deposition controller cannot make use of this single-step solution. A full development of solvers for time-based models is found in Chapter 5.

Input Constraints

The controller's cost function optimization routines must also deal with a number of practical constraints, including input bounds and input discretization. Input bounds must be enforced for empirically modeled systems to ensure that the selected inputs fall within "reasonable" ranges, essentially regions of the input space where the model is likely to be fairly accurate. Input discretization constraints exist because the settings on most equipment have a finite number of digits with which they can be entered. For example, the deposition time for the epi process can be entered in tenths of a second via the front panel. A proper back-solver needs to be aware of this limitation and restrict itself to suggesting inputs of the proper resolution. This limitation creates a discrete optimization problem whose full theoretical treatment has not been analyzed.

Previous work has been done to create tractable, fixed iteration back-solving methods to simultaneously address both discretization and bounded input constraints. [Moy95] [BMSM95] [Smi95]. Work presented here uses the "delta" discretization method for selecting inputs. Essentially the relative output change for a discrete change in input is calculated for every input. The optimization process iterates once per input, fixing one input

at a time until all inputs are fixed. The inputs with the largest “discretized output” effects are fixed first.

4.3.2 Model Updates

As described earlier, the run-to-run controller’s dynamic site models consist of linear functions of deposition rates and resistivities in terms of the equipment settings. The models contain a slope matrix, $\boldsymbol{A}$, and a vector of offset terms, $\boldsymbol{b}$, as in Equation 3-1. The slope matrix is determined from the DOE and is not modified by the EWMA controller.

The controller tracks output disturbances and drifts by updating the offset terms. After a set of measurements is presented to the controller, it corrects for model errors by changing the offset for each output model. The EWMA weights (α) determine the amount of model update after each run (n), based on the equations

$$b_i[n] = \alpha_i \cdot \hat{b}_i[n] + (1-\alpha_i) \cdot b_i[n-1], \; \hat{b}_i[n] = y_i[n] - \boldsymbol{A_i x}[\boldsymbol{n}], \qquad \textbf{(Eq 4-13)}$$

where $y_i[n]$ is the measured output after processing a wafer with equipment settings $\boldsymbol{x}[n]$. These update formulas are used for both the deposition rates (y_i) and the resistivities (y_j).

Optimal EWMA weights for a given process are selected as a function of the statistical noise in the output measurements and the amount of actual deterministic drift or disturbances the system exhibits; a thorough treatment is found in [Smi96]. Estimates of output noise and drift should be determined by the DOE runs and historical data from the equipment. However, for demonstration of the controller’s reaction to a staged disturbance event, the controller typically uses a “large” EWMA constant for each output (e.g. as in the SISO run-to-run control demonstration, where α was set to 0.7).

4.3.3 Combining Lot-to-Lot and Run-to-Run Feedback Data

An important issue for run-to-run control is the combined use of in-line (run-to-run) and off-line (lot-to-lot) sensors. Measurements from the DOE give a full set of data for each run, but many of these will not be available on a run-to-run basis in an actual production setting. Model update strategies and feedback loops must work together to ensure that sensor data are used effectively.

Consider the epi thickness measurements, which are available from two different sensors. An in-line thickness sensor provides a few measurements along a single diameter, yielding an even smaller number of radially averaged site measurements. An off-line thickness sensor provides a densely sampled wafer map, yielding many radial site thickness averages. In other words, one sensor samples more densely in time but less densely in space, while the other sensor does just the opposite. The question then arises: how should the modeling and control strategies make use of combined feedback from these two sensors?

More generally, this problem presents itself when multiple sensors provide **correlated** measurements at **different** sampling rates. If the sensors return correlated data at the same time intervals, then appropriate strategies can be applied to extract important features before the modeling steps are performed. If the sensors return uncorrelated data, then the modeling steps are performed independently and do not rely on the availability of both measurements at the same time.

Note that the resistivity measurements fall into the latter category; resistivity is not correlated with film thickness. An epi film of a given resistivity can exist for essentially any thickness, and vice versa. Therefore, given information about one characteristic, we

can make no (or very few) assumptions about the other. Thus the resistivity site models are updated only when resistivity data comes back, which is on a lot-to-lot basis.

The two different thickness measurement systems, however, provide the more difficult case. Measurements of the same sites on the same wafers are almost perfectly correlated. However, the in-line sensor provides thickness readings for wafers that the off-line sensor does not see, and the off-line sensor provides thickness readings for sites that the in-line sensor does not see. How does one effectively use the combined information provided by these two systems?

There are many possible ways to deal with this problem, most of which will fall into one of the following three strategies.

1. Model the output from one of the sensors, while ignoring the other.
2. Use individual models for both sensor outputs, ignoring any correlation.
3. Create a single set of models that can be updated by **either** sensor's output.

The following three sections consider these approaches with respect to the epi thickness control problem.

Single Sensor Monitoring

One way to deal with this problem is to simply model the outputs from the sensor that provides the most important information and ignore the other sensor(s). Perhaps the other sensor(s) could be used for fault detection, but modeling and control is based on the outputs from just one sensor. Of course one needs to define a method for selecting the "best" sensor for the feedback system. This will depend on the characteristics of the system to be controlled, the information provided by the sensors, and our ability to model that system.

The epi thickness monitoring problem involves a trade-off between the temporal and

spatial resolutions, as well as the noise characteristics, of the in-line and off-line sensors. The former provides better temporal resolution, while the latter provides better spatial resolution. The off-line sensor also provides better noise characteristics because it averages multiple measurements to generate each radial output "site." Deciding which measurement type is better for process control depends on the types of disturbances the system normally exhibits. Is the system relatively stable over the course of a lot, and are the process models accurate enough to run a complete lot between feedback measurements? If so, then the added spatial resolution and lower sensor noise might be more important than the faster sampling rate. If the process tends to drift during a lot, or if the process models are not accurate enough, then perhaps the temporal resolution of the in-line sensor is required.

Epi thickness models are built from the densely sampled radial profiles of the DOE wafers. These outputs directly apply to modeling and control based on the off-line sensor, which may be used if pure lot-to-lot control is satisfactory. However, assuming that run-to-run control using the in-line sensor data is preferred, then one might question the wisdom of gathering densely sampled profiles from the DOE runs. Why not just evenly sample the radial profiles at as many locations as the in-line sensor will allow? Models and controllers based on these data can be built and implemented without ever using the off-line sensor.

While such a simplification is certainly possible, gathering DOE data from the off-line sensor provides the opportunity for more effective modeling and control, even if the in-line sensor will be used exclusively after the DOE. These benefits can be achieved by questioning the assumption that a radial profile should be sampled at evenly spaced intervals. There is no guarantee that these are the "best" sites for measuring nonuniformity. It

might turn out that certain radial regions are prone to larger variations and require dense sampling, while other regions might be more stable and can be sampled lightly. In any case, densely sampling the whole radial profile provides the data necessary to make these analyses, and to decide exactly where the in-line sensor should measure thicknesses.

Given a constraint of selecting *N* radial sites for the in-line sensor, optimal measurement locations may be found by using the full radial profiles from the DOE. One approach is to choose the locations such that the cost function of the epi thickness measurements from the *N* sites along a radius maximally covaries with the cost function of the full radial scans, over the set of DOE runs. Models can be built for those *N* locations, which may be optimized and integrated into the run-to-run control algorithm. The in-line sensor can then be configured to return data from those sites.

Combined Single Sensor Monitoring

A second approach is to simply maintain separate models for the outputs from each sensor, and integrate the combined set of outputs into the cost function. Models based on the in-line sensor outputs would be updated more frequently than models based on the off-line sensor outputs, but there would be fewer of them. An appropriate cost function might then become

$$Cost = \sum_i [W_{ti}^{in}(T_{ti}^{in} - (r_i^{in}(\boldsymbol{x}) \cdot d))]^2 + \sum_j [W_{tj}^{ex}(T_{tj}^{ex} - (r_j^{ex}(\boldsymbol{x}) \cdot d))]^2 + \sum_k [W_{\rho k}^{ex}(T_{\rho k}^{ex} - \rho_k^{ex}(\boldsymbol{x}))]^2, \quad \textbf{(Eq 4-14)}$$

where *in* selects in-situ related terms and *ex* selects ex-situ related terms. Only the models of in-situ rates would be updated on every run.

The single sensor monitoring approach is actually a special case of this strategy. The weighting terms for some of the outputs may be set to zero, thus ignoring those models. The combined single sensor monitoring approach enables an engineer to trade-off the relative importance of temporal resolution versus spatial resolution, yet still use both components in the control strategy.

Of course this scenario leads to concerns about lags in off-line sensor data, and delays in response to the in-situ data. One approach to solving this could be through the use of an adaptive weighting scheme that penalizes (lowers the weight on) output terms based on an estimate of the model's "age." However, proper configuration and analysis of a control system's weighting scheme is already difficult, even with static weights. Further analysis of this method is not addressed in this work, as the following "joint sensor monitoring" techniques are more appealing.

Joint Sensor Monitoring

Finally, there is the promising concept of "joint sensor monitoring," where a single set of output models is maintained for all of the sensors with correlated measurements. Broadly speaking, the goal is to create models and model update strategies that do not require all of the measurements to be available at once, yet capture the important information from each sensor.

Implementing such a strategy is aided by first considering an ideal scenario, where all sensor data are available at the same time, and at the highest measurement frequency provided by any of the sensors. As mentioned earlier, this type of system presents a case where there is no longer an issue of combining lot-to-lot and run-to-run data. The simplified problem requires the engineer to decide what output models to build, and how to gen-

erate those outputs from the combined set of measurements. This is the type of analysis performed in Section 4.1.1, when it was decided to build models of radial sites. These virtual outputs are not directly provided by the sensor, but are formed by compressing the full set of measurements into a smaller set of outputs that are less noisy and more controllable. Generally, the same types of analyses can be used to merge correlated measurements into a set of virtual output signals. Ideally the virtual outputs exhibit better noise characteristics than the pure measurements and capture only the features that can actually be manipulated or controlled.

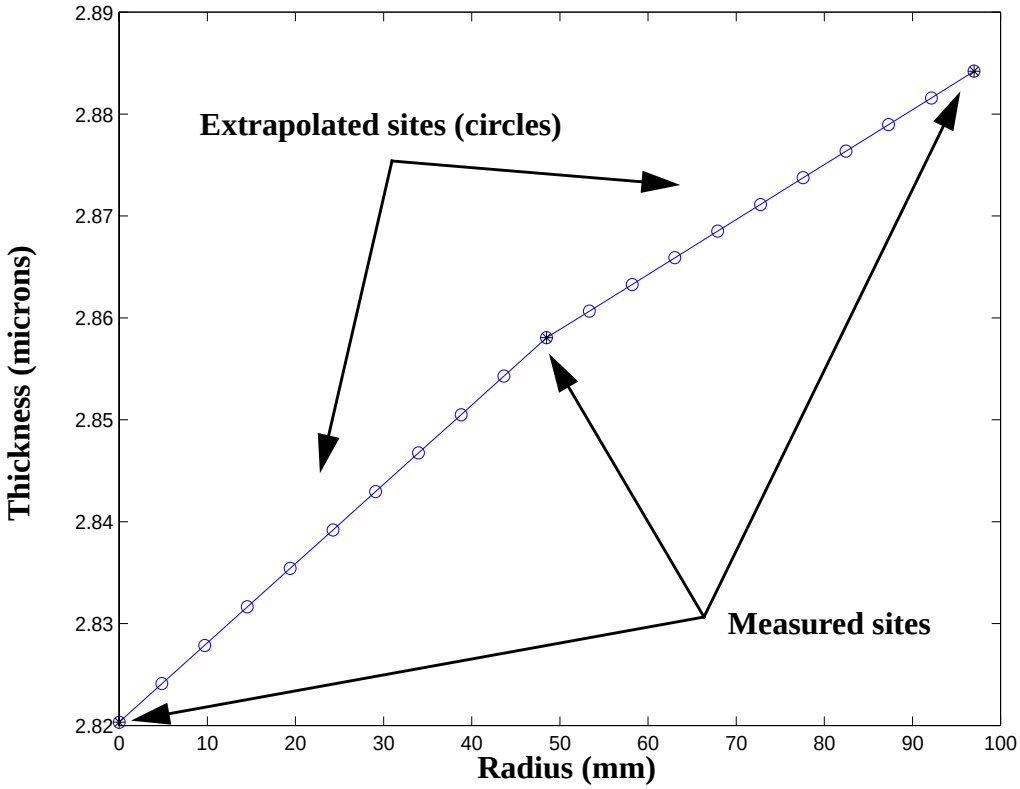
For the epi thickness modeling and control problem there are two different sets of outputs, one is a densely sampled radial thickness profile, while the other is a lightly sampled version of the same profile. Essentially the in-line sensor provides a subset of the spatial information from the off-line sensor. If one is to consider a situation where both of these sensors provide data after every run, then we would clearly keep the densely sampled profile and simply average in or ignore the lightly sampled profile. Thus the selection of virtual outputs is quite simple; we will keep the densely sampled radial profile.

Given a mapping from sensor measurements to virtual outputs, the next step is to define appropriate strategies for mapping individual sensor measurements to the full set of virtual outputs. Presumably this would be difficult to do for some or all of the sensors. Remember that the measurements must be correlated in some manner for them to be grouped together, so knowing the outputs from one sensor does provide information about what the other sensor(s) would measure. It would seem that we could use the correlation structures to create the most likely set of combined outputs, given the measurements from a single sensor. However, the correlation could be weak, so generating a set of virtual out-

puts based on only one sensor will likely introduce noise and errors. This cannot be avoided.

We will see that there is a way to improve upon this basic strategy, but for now consider formulating the epi thickness control problem using these concepts. First, it is easy to see what should happen when a full radial scan is produced by the off-line sensor. The virtual outputs are simply equal to the sampled radial profile, since that is what they were defined to be. The interesting problem is how to generate a full radial scan from the small number of sites sampled by the in-line sensor. A logical and quite defensible suggestion is to simply "connect the dots" and generate a piecewise linear extrapolation of the available sites, as demonstrated by Figure 4-2.

Figure 4-2: Piecewise linear extrapolation for a virtual radial profile



Jumping ahead slightly, we can justify this type of extrapolation based on an analysis

of our DOE data. The data will show that the radial site thickness correlation matrix has an approximately linear falloff with increasing distance between the radial samples. For example, the correlation coefficient matrix for the densely sampled radial sites might look like

$$\begin{bmatrix} 1.0 & 0.9 & 0.8 & 0.7 & 0.6 \\ 0.9 & 1.0 & 0.9 & 0.8 & 0.7 \\ 0.8 & 0.9 & 1.0 & 0.9 & 0.8 \\ 0.7 & 0.8 & 0.9 & 1.0 & 0.9 \\ 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \end{bmatrix}.$$

Given this type of correlation structure and a multivariate gaussian process with identical mean and noise characteristics at each site, a piecewise linear construction provides a minimum mean squared error (MMSE) estimation for all of the sites. This can be verified by computing the conditional mean of a multivariate gaussian as

$$\boldsymbol{m}_{x|y}(\boldsymbol{Y}) = \boldsymbol{m}_x + \Lambda_{xy}\Lambda_y^{-1}(\boldsymbol{Y} - \boldsymbol{m}_y). \qquad \textbf{(Eq 4-15)}$$

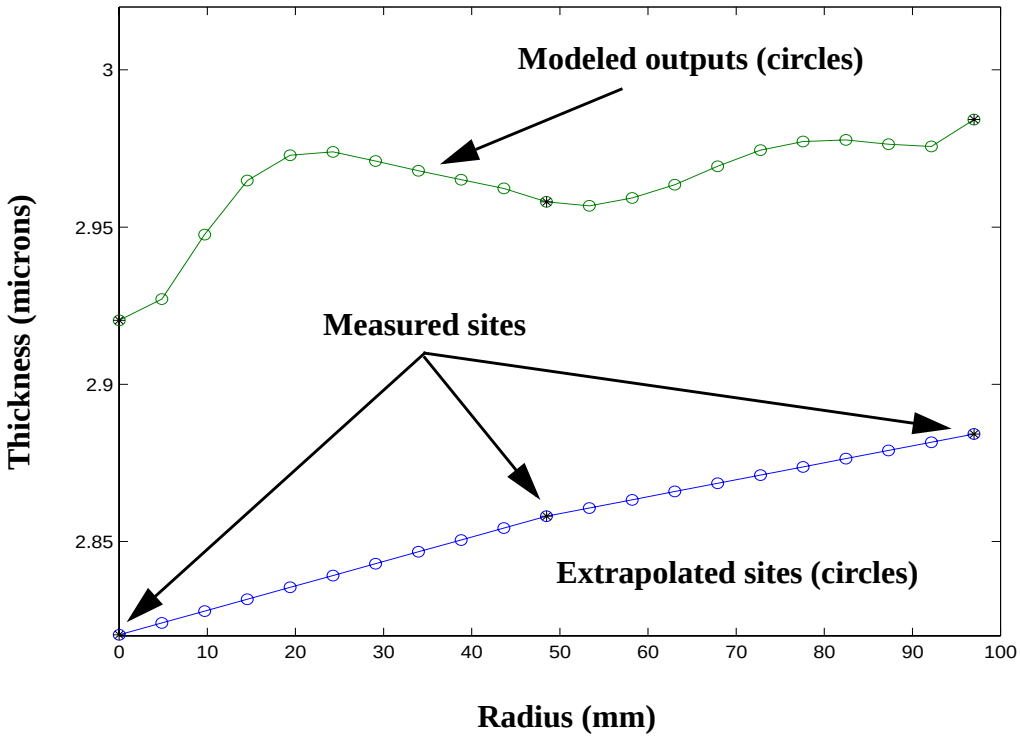
The conditional mean for the complete set of radial sites ($\boldsymbol{X}$), given the measured sites ($\boldsymbol{Y}$), is shown in terms of the mean of $\boldsymbol{X}$ ($\boldsymbol{m}_x$), the mean of $\boldsymbol{Y}$ ($\boldsymbol{m}_y$), the cross-covariance matrix between $\boldsymbol{X}$ and $\boldsymbol{Y}$ (Λ_{xy}), and the covariance matrix for $\boldsymbol{Y}$ (Λ_y).

After generating a new set of virtual outputs from the available sensor data, the corresponding output models are updated. The question is, does the methodology described above really utilize all of the available information for generating virtual outputs? There is in fact another source of information about the relative thicknesses of the measured and unmeasured radial sites. Weighted histories for all of the thickness outputs are contained in the current thickness models. While the above derivation for extrapolating thicknesses is reasonable, given that nothing else is known, it does not take into account information contained in the current models.

This insight leads to a more effective update strategy: given a set of input parameters, a set of output models, and a mapping from sensor measurements to virtual outputs, we must define appropriate methods for mapping individual sensor measurements to a full set of virtual outputs. Built into the current models are past measurements from **all** of the sensors, plus the expected relations between the various inputs and outputs of the system. Clearly an output estimator should take these models into account when extrapolating virtual outputs from a subset of measurements.

Algorithms for generating virtual outputs are highly dependent on the structure of the problem. Additionally, there are likely to be many reasonable and mathematically sound methods available for any given case. Consider the example in Figure 4-2, where Figure 4-3 now also includes the outputs from the current models; these are the expected outputs for the measured run. Notice that the extrapolation assumes a linear profile between the measured sites, ignoring information provided by the model. Over time, this type of strategy gradually eliminates any information that the models provide about the unmeasured sites. Between updates from off-line measurements, this system becomes one that simply attempts to maintain the outputs from the in-line sensor readings, much like a “single sensor” monitoring system.

Figure 4-3: Original virtual radial profile and modeled radial profile



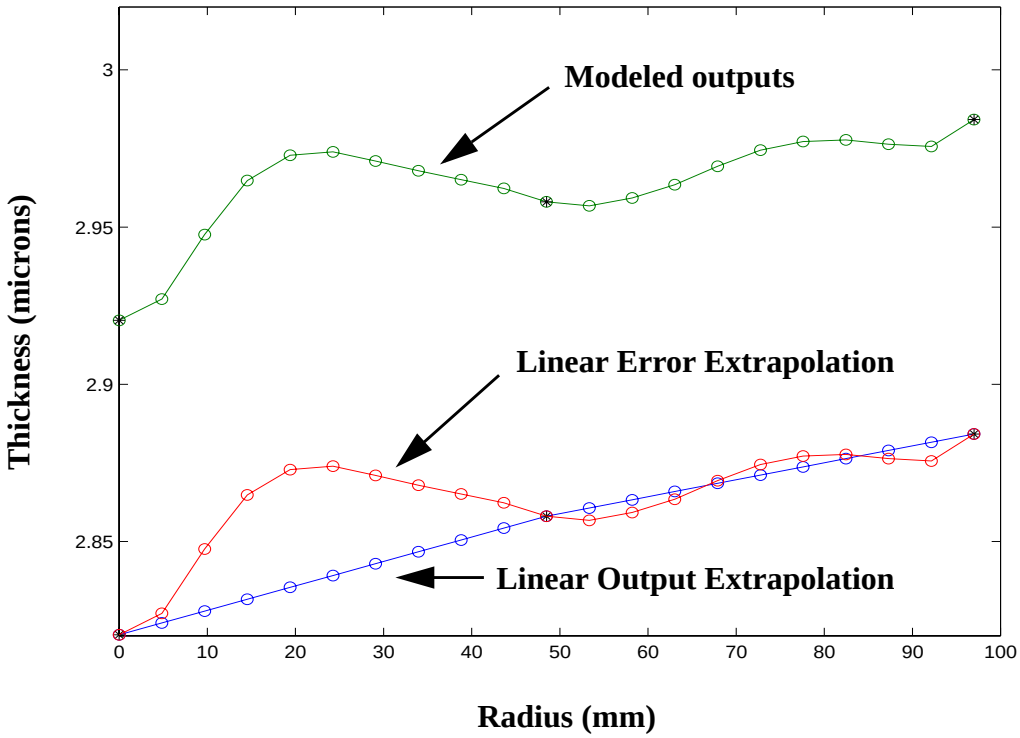
A better strategy for extrapolating the full set of outputs is to assume that the modeled outputs provide deterministic components (means), while the measured sites provide information about the deviation, or error. Instead of directly using the piecewise linear extrapolation of the measured sites, a piecewise linear extrapolation of the deviation (error) can be added to the modeled outputs. For the given example, Figure 4-4 shows what the new set of virtual outputs looks like. Instead of virtual outputs with the form

$$\boldsymbol{X_v} = f(\boldsymbol{X_m}) \qquad \textbf{(Eq 4-16)}$$

$$\text{we have } \boldsymbol{X_v} = \hat{\boldsymbol{X}}_v + f(\boldsymbol{X_m} - \hat{\boldsymbol{X}}_{v=m}), \qquad \textbf{(Eq 4-17)}$$

where $\boldsymbol{X_v}$ are the virtual outputs, $\boldsymbol{X_m}$ are the measured outputs, and $\hat{\boldsymbol{X}}_v$ are the modeled (expected) outputs.

Figure 4-4: Virtual radial profile using models and in-line measurements



This error extrapolation strategy has a number of useful features:

- The error surfaces (curves) often exhibit smaller (in magnitude) and/or lower frequency variations than the actual output surfaces. Thus fewer samples can be used to achieve a given level of confidence, and simpler extrapolation techniques (like piecewise linear methods) can be used.
- The exact in-line measurements are still part of the extrapolated virtual outputs. This is important because the thickness sensors (both in-line and off-line) are very accurate. The predicted (model) outputs should not influence these sites.
- There is less worry about picking the optimal measurement sites, as described in the "single sensor" monitoring scenario. The output models help account for variation in unmeasured locations. Thus evenly spaced samples can work well without skewing the results.
- This strategy uses (presumably) more accurate assumptions about the relative spatial behavior between the measured sites. It relies primarily on the model, and thus

the information garnered from the DOE and previous off-line sensor measurements, to help fill in the gaps where there are no measurements.

- Finally, as shown by Figure 4-4, the error extrapolation method responds appropriately when there is a simple mean shift across the entire profile. If all of the in-line measurements shift by a fixed amount, then this strategy simply shifts the profile so that the measured sites line up properly, but the shape is left unchanged. This can be seen by recognizing that a mean shift yields identical error readings for all measured sites. The piecewise linear extrapolation of this constant error is a straight line. Thus, the same error value is added to all of the modeled output sites. (As we will see later from the results of the DOE, this type of shift or drift is the largest type of disturbance found in the system.)

There are a number of synchronization details that make the joint sensor monitoring strategy difficult to implement. The described process works fine as long as the sensor information comes back immediately after the wafer is ready to be measured. Of course there are the “standard” issues of lag associated with updating a model used for process control. Problems can arise because it takes time to get wafers to off-line sensors, take the measurements, and get the data back to the controller. If this is the case, then the controller can be updating the recipe for the today’s wafers, based on yesterday’s processing. Proper filtering, modeling, data handling, and stability analyses can be incorporated to make sure this is not a problem. This issue is important regardless of whether or not combined lot-to-lot and run-to-run sensor data are used.

However, measurement lag presents a bigger problem with joint sensor monitoring. First, consider what happens to the system as a monitor wafer passes through. Here, a monitor is defined to be a wafer that is processed, then pulled out of the process flow and measured by off-line sensors. (Presumably these wafers are not returned to the process flow, but that is not really of concern here.) There is generally one monitor wafer per “lot.”

This wafer is first measured by the in-line sensor right after processing, then it is measured again when it reaches the off-line thickness sensor. The control system must be careful not to update the model twice with outputs from the same wafer. The proper update strategy is to first merge the two sets of outputs for that wafer, then update the model once. For our example, merging of the outputs would most likely mean averaging the in-situ measurements with the off-line measurements, for the overlapping sites. Since both sensors are accurate and based on the same technology, the averaging provides extra noise immunity for those sites. Otherwise the in-situ measurements could simply be ignored.

Of course the major problem with this solution is the lack of coordination between the sensors. Clearly the in-situ sensor will provide its data for the monitor wafer well before the off-line sensor. How can the information be merged when only one of the data sets is available? The proper action is to first use the in-line data as they become available and continue processing and updating the models accordingly. However, when the delayed off-line data become available, the model should be "retroactively" updated for the wafer that was measured, as well as all wafers measured after that. Basically the system should be able to restore the model that was originally used when the monitor wafer was processed, then update the model as described in the previous paragraph, by merging the in-situ and off-line sensors. The new model should then be brought up to date by re-running all of the updates, based on the new model and all of the subsequent in-situ measurements.

The ability to update the model retroactively requires the system to maintain an easily accessible history of all in-situ measurements since the last set of off-line measurements. This strategy adds considerable complexity to the controller, but it is required to properly

address the problem of combining in-line and off-line data. In an experimental setting the problem can be circumvented by ensuring that all measurements come back in order and before the next wafer is processed. A simplified experimental scenario like this is discussed in more detail later, but a full scale production system needs to gracefully handle uncoordinated feedback lag.

4.4 Summary

This chapter developed strategies and techniques for demonstrating run-to-run uniformity control of multiple characteristics. Specifically, deposited epi film thickness and resistivity uniformities are to be controlled for an Applied Materials Centura system. Effective multi-objective control requires experimental process data from a DOE, followed by complementary methods for process modeling, optimization and run-to-run control. Aggressive use of multiple response surface modeling is suggested, in conjunction with "joint sensor techniques" for merging run-to-run and lot-to-lot sensor data. One of the key components in this methodology is process optimization of the time-based models. Brief coverage of our optimization strategy is found in Section 4.3.1, but the following chapter will delve more deeply into this problem.

Chapter 5

Time-Based Back-Solvers

The formulation of time-based models is intuitive and straightforward; the use of rate-based outputs and time-based inputs provides considerable flexibility and modeling accuracy. However, this added capability comes at a price. Even with simple linear models of processing rate, the resulting time-based model is nonlinear, which means that more advanced solver techniques must be used for finding optimal inputs, given target outputs.

Generally two different types of solving techniques are required, depending on the details of the system model. Models are said to be "overdetermined," "underdetermined," or "exactly determined," which can be defined in terms of the system's "achievable" outputs. Here we define achievable outputs to be the subspace of all possible outputs that can be obtained by passing all possible inputs through the system.

Overdetermined systems have possible outputs (generally an infinite number) that cannot be achieved by any set of inputs. Further, in the case of an overdetermined linear system with full rank, there is a unique pairing between every achievable output and its corresponding input. For these two reasons, the solver for an overdetermined system often tries to select inputs which achieve an output that is of minimum (weighted) distance from the target output.

Underdetermined systems can achieve any output with multiple (usually an infinite number of) input vectors. In the case of an underdetermined linear system with full rank, there will be a linear subspace of inputs that can achieve a given output. An exactly deter-

mined system can be thought of as a special case of overdetermined system where the complete output space is achievable, but there is a unique mapping between each input and each output. In underdetermined cases, the solver routine generally tries to select an input that simultaneously achieves the target output and is of minimum (weighted) distance from a given starting input. Thus overdetermined systems find minimum distance solutions in the **output** space, while underdetermined systems find (constrained) minimum distance solutions in the **input** space.

The following sections extend the well-known linear system solver techniques for use with time-based models. Section 5.1 details the theoretical background for the iterative solver introduced in Chapter 4, Section 5.2 provides the single-step solution for a special case of overdetermined systems, and Section 5.3 considers solutions for exactly determined and underdetermined systems.

5.1 The General Time-Based Back-Solver

An iterative back-solver for the time-based weighted sum of squared error solution was described in Section 4.3.1. The cost function structure found in that section is:

$$Cost = \sum_{i=1}^{I} [W_{ti}(T_{ti} - (r_i(\boldsymbol{x}) \cdot d))]^2 + \sum_{j=1}^{J} [W_{\rho j}(T_{\rho j} - \rho_j(\boldsymbol{x}))]^2, \qquad \textbf{(Eq 5-1)}$$

where $r_i(\boldsymbol{x}) = \boldsymbol{A}_{ri}\boldsymbol{x} + b_i$ and $\rho_j(\boldsymbol{x}) = \boldsymbol{A}_{\rho j}\boldsymbol{x} + b_j$.

This can be put into matrix form as

$$Cost = (\boldsymbol{T}_t - (\boldsymbol{A}_t\boldsymbol{x} + \boldsymbol{b}_t) \cdot d)^t \boldsymbol{W}_t^2 (\boldsymbol{T}_t - (\boldsymbol{A}_t\boldsymbol{x} + \boldsymbol{b}_t) \cdot d) + (\boldsymbol{T}_\rho - (\boldsymbol{A}_\rho\boldsymbol{x} + \boldsymbol{b}_\rho))^t \boldsymbol{W}_\rho^2 (\boldsymbol{T}_\rho - (\boldsymbol{A}_\rho\boldsymbol{x} + \boldsymbol{b}_\rho)),$$

where $(\ )^t$ is the transpose operator, and

$$\boldsymbol{W_t} = diag\begin{bmatrix} W_{t1} \\ \dots \\ W_{tI} \end{bmatrix},\ \boldsymbol{T_t} = \begin{bmatrix} T_{t1} \\ \dots \\ T_{tI} \end{bmatrix},\ \boldsymbol{A_t} = \begin{bmatrix} \boldsymbol{A_{t1}} \\ \dots \\ \boldsymbol{A_{tI}} \end{bmatrix},\ \boldsymbol{b_t} = \begin{bmatrix} b_{t1} \\ \dots \\ b_{tI} \end{bmatrix},$$

$$\mathbf{W}_\rho = diag\begin{bmatrix} W_{\rho 1} \\ \dots \\ W_{\rho J} \end{bmatrix},\ \boldsymbol{T}_\rho = \begin{bmatrix} T_{\rho 1} \\ \dots \\ T_{\rho J} \end{bmatrix},\ \boldsymbol{A}_\rho = \begin{bmatrix} \boldsymbol{A}_{\rho \boldsymbol{1}} \\ \dots \\ \boldsymbol{A}_{\rho \boldsymbol{J}} \end{bmatrix},\ \boldsymbol{b}_\rho = \begin{bmatrix} b_{\rho 1} \\ \dots \\ b_{\rho J} \end{bmatrix}.$$

5.1.1 Solving for the Non-Time Inputs

Consider the case where the deposition time (d) is held constant. It can be factored away from the linear rate models and into the weighting and target terms; the cost function summation can then be written as

$$Cost = (\boldsymbol{T} - (\boldsymbol{Ax} + \boldsymbol{b}))^t \boldsymbol{W^2} (\boldsymbol{T} - (\boldsymbol{Ax} + \boldsymbol{b})), \qquad \textbf{(Eq 5-2)}$$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W_t} \cdot d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_\rho \end{bmatrix},\ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T_t} / d \\ \boldsymbol{T}_\rho \end{bmatrix},\ \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A_t} \\ \boldsymbol{A}_\rho \end{bmatrix},\ \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b_t} \\ \boldsymbol{b}_\rho \end{bmatrix},$$

which is simply in the form of the "classic" weighted least squares optimization problem with linear models [Moy95]. Given targets ($\boldsymbol{T}$), and a deposition time (d), the solution for the minimum cost input ($\boldsymbol{x}$) is found to be

$$\boldsymbol{x} = (\boldsymbol{A}^t \boldsymbol{W}^2 \boldsymbol{A})^{-1} \boldsymbol{A}^t \boldsymbol{W}^2 (\boldsymbol{T} - \boldsymbol{b}), \qquad \textbf{(Eq 5-3)}$$

when there are more outputs than inputs (the overdetermined case). Assuming the slope matrix ($\boldsymbol{A}$) has full row rank and that all weights ($\boldsymbol{W}$) are nonzero, then the structure of the time-based model guarantees that $(\boldsymbol{A}^t \boldsymbol{W}^2 \boldsymbol{A})^{-1}$ exists, and a unique solution will be found.

If there are more inputs than outputs (the underdetermined case), we are also provided with a starting point ($\mathbf{x_0}$) and an input weighting matrix ($\boldsymbol{V_x}$). Since we have fixed the deposition time, its weighting term is irrelevant. Therefore, the input weighting matrix used

here only has terms for the non-time inputs. The zero cost solution that is a minimum weighted distance from the starting point is

$$\boldsymbol{x} = \boldsymbol{x_0} - \boldsymbol{V_x}^{-2}\boldsymbol{A}^{\boldsymbol{t}}(\boldsymbol{A}\boldsymbol{V_x}^{-2}\boldsymbol{A}^{\boldsymbol{t}})^{-1}(\boldsymbol{A}\boldsymbol{x_0} - \boldsymbol{T} + \boldsymbol{b}), \qquad \textbf{(Eq 5-4)}$$

where the distance cost function is given by

$$Cost = (\boldsymbol{x} - \boldsymbol{x_0})^{\boldsymbol{t}}\boldsymbol{V_x}^{\boldsymbol{2}}(\boldsymbol{x} - \boldsymbol{x_0}).$$

It is unlikely but conceivable that there could be two or more different time inputs as part of a single time-based model. One plausible scenario is a multi-step recipe where two different processing times affect the wafer during a single run. Perhaps there are two separate deposition steps within a single recipe, where two layers with different characteristics are grown sequentially. In any case, it is clear that the time-based model and optimization structure should support this type of problem. This is actually not difficult at all; consider the case where there are two time inputs, $d1$ and $d2$ We can again generate the proper "classic" weighted least squares optimization using

$$W = \begin{bmatrix} W_{t1} \cdot d1 & 0 & 0 \\ 0 & W_{t2} \cdot d2 & 0 \\ 0 & 0 & W_{\rho} \end{bmatrix}, \; T = \begin{bmatrix} T_{t1} / d1 \\ T_{t2} / d2 \\ T_{\rho} \end{bmatrix}, \; A = \begin{bmatrix} A_{t1} \\ A_{t2} \\ A_{\rho} \end{bmatrix}, \; b = \begin{bmatrix} b_{t1} \\ b_{t2} \\ b_{\rho} \end{bmatrix}.$$

Clearly this can be extended to handle as many simultaneous time inputs as required. It is important to keep in mind the fact that these solutions are used when all of the time inputs are held constant.

5.1.2 Solving for the Time Inputs

Now consider the case where the non-time inputs are held constant, and we must solve for the time input(s). Looking at the structure of the time-based model, only the outputs that are multiplied by a particular time input need to be considered together. Thus, for the

epi deposition scenario, the cost function to optimize is

$$Cost = (\boldsymbol{T_t} - (\boldsymbol{A_t x} + \boldsymbol{b_t}) \cdot d)^t \boldsymbol{W_t^2} (\boldsymbol{T_t} - (\boldsymbol{A_t x} + \boldsymbol{b_t}) \cdot d). \quad \textbf{(Eq 5-5)}$$

since the non-time inputs are held constant, we can treat the outputs of the linear models as constants, simplifying the cost function to

$$Cost = (\boldsymbol{T_t} - \boldsymbol{r_t} \cdot d)^t \boldsymbol{W_t} (\boldsymbol{T_t} - \boldsymbol{r_t} \cdot d), \quad \textbf{(Eq 5-6)}$$

$$\text{where } \boldsymbol{r_t} = \boldsymbol{A_t x} + \boldsymbol{b_t}.$$

Given a set of targets and rates, the minimization of this cost function yields an optimal deposition time of

$$d = \frac{\boldsymbol{T_t^t W_t^2 r_t}}{\boldsymbol{r_t^t W_t^2 r_t}}. \quad \textbf{(Eq 5-7)}$$

If there are multiple time inputs, then each of them can be optimized individually, based on the output weights and output rates upon which they operate.

Note that there is no such thing as an underdetermined solution for these inputs, assuming that we fix the non-time inputs. At most there are the same number of inputs as there are outputs, which becomes a degenerate case where each deposition time d_i is found as simply

$$d_i = \frac{T_{ti}}{r_{ti}}. \quad \textbf{(Eq 5-8)}$$

There cannot be more inputs than outputs because each input is tied to at least one output and each output has at most one time factor associated with it.

5.1.3 An Iterative Back-Solver

The solutions provided above solve for one type of input, given that the other type is held constant, but we need a system that can simultaneously solve for all of the inputs. A

complete, single step solution does not neatly fall out of the structure. However, one can set up a simple iterative scheme, where the non-time inputs are solved while the time inputs are held constant, then the time inputs are solved while the non-time inputs are held constant, then this sequence is repeated. The given input values ($\mathbf{x_0}$) are used as a starting point for the iterations. (The selection of which step begins the sequence is discussed in more detail below.)

For an overdetermined system with more outputs than combined time and non-time inputs, the iterative process provides a convenient solution whereby the cost function is guaranteed to decrease on each iteration, descending toward a local minimum. As described above, the iteration solutions are

$$\boldsymbol{x} = (\boldsymbol{A}^t\boldsymbol{W}^2\boldsymbol{A})^{-1}\boldsymbol{A}^t\boldsymbol{W}^2(\boldsymbol{T}-\boldsymbol{b})$$

$$\text{and } d = \frac{\boldsymbol{T}_t^t\boldsymbol{W}_t^2\boldsymbol{r}_t}{\boldsymbol{r}_t^t\boldsymbol{W}_t^2\boldsymbol{r}_t}.$$

The iteration stopping rules include:

- The cost function stops decreasing with continued iterations, to within a preset threshold.
- The inputs do not change after one iteration of the two-step cycle, to within a preset threshold.
- The cost has been driven to zero, to within a preset threshold. *
- The maximum number of iterations has been reached.

* If the cost is driven to zero, then an exact solution must have been found, meaning that the system is actually either underdetermined or exactly determined (one unique solution).

Solutions for underdetermined systems will be discussed in more detail below. However, since the uniformity controller used in this work maintains a large number of individ-

ual site models (more output models than input process settings), solutions for the underdetermined case need not be implemented. Even if the system were underdetermined, the iterative solution described above will still yield a minimum cost solution; however a minimum distance (input change) solution is not guaranteed.

Consider a system where there are more non-time inputs ($\boldsymbol{x}$) than there are outputs. This implies that there are infinite solutions for $\boldsymbol{x}$, regardless of what the time inputs are. A single iteration will arrive at a zero cost operating point, since a solution for the $\boldsymbol{x}$ vector will match itself to any deposition time in one step. For any given time inputs, the non-time inputs will be solved using a minimum distance rule. However, there could be a smaller input change solution when including the distance moved in the time input dimension(s). The combined movement cost is

$$(\boldsymbol{x} - \boldsymbol{x_0})^t \boldsymbol{V_x}^2 (\boldsymbol{x} - \boldsymbol{x_0}) + (d - d_0)^t V_d^2 (d - d_0)\,, \qquad \textbf{(Eq 5-9)}$$

which is not guaranteed to be minimized by the iterative strategy.

It is also possible that the non-time input solution step is overdetermined, but the complete solution is underdetermined. This can happen if there are fewer non-time inputs than outputs, but the number of combined (non-time + time) inputs is greater than the number of outputs. In this case multiple iterations of the overdetermined solution may be required. The resulting solution will be locally minimum cost, but again, it will not necessarily be minimum distance.

Input Discretization

Input discretization constrains the solver by forcing inputs to be selected with a given resolution. Using the iterative back-solver simplifies this problem somewhat. Each step of

the iteration process is forced to return a discretized set of inputs, either non-time inputs or time inputs. As described in the previous chapter, the non-time input solver routine makes use of the “delta” discretization procedure to deal with this constraint [BMSM95] [Smi95]. Then, given a constant set of non-time inputs, discretizing the time inputs is simple. The time inputs do not interact since there is a one-to-one mapping between each rate-based output and a single time input. Thus each time input can be discretized independently of each other by simply rounding them. (Any optimization procedure that changes both time and non-time inputs simultaneously would have to deal with this issue in more detail.)

5.1.4 An Example Solution

Consider a simple system representing a scenario similar to that seen in the epi control problem. Specifically there are two inputs, a single non-time input and a single time input that multiplies some, but not all of the outputs. There are three outputs, defined as

$$\begin{aligned} y_{thick1} &= d(A_0 x + b_0) = d(1x + 0) = d \cdot x \\ y_{thick2} &= d(A_1 x + b_1) = d(0x + 1) = d \\ y_{rho1} &= (A_2 x + b_2) = (1x + 0) = x \end{aligned} .$$

Using the matrix notation defined at the beginning of this chapter, we have

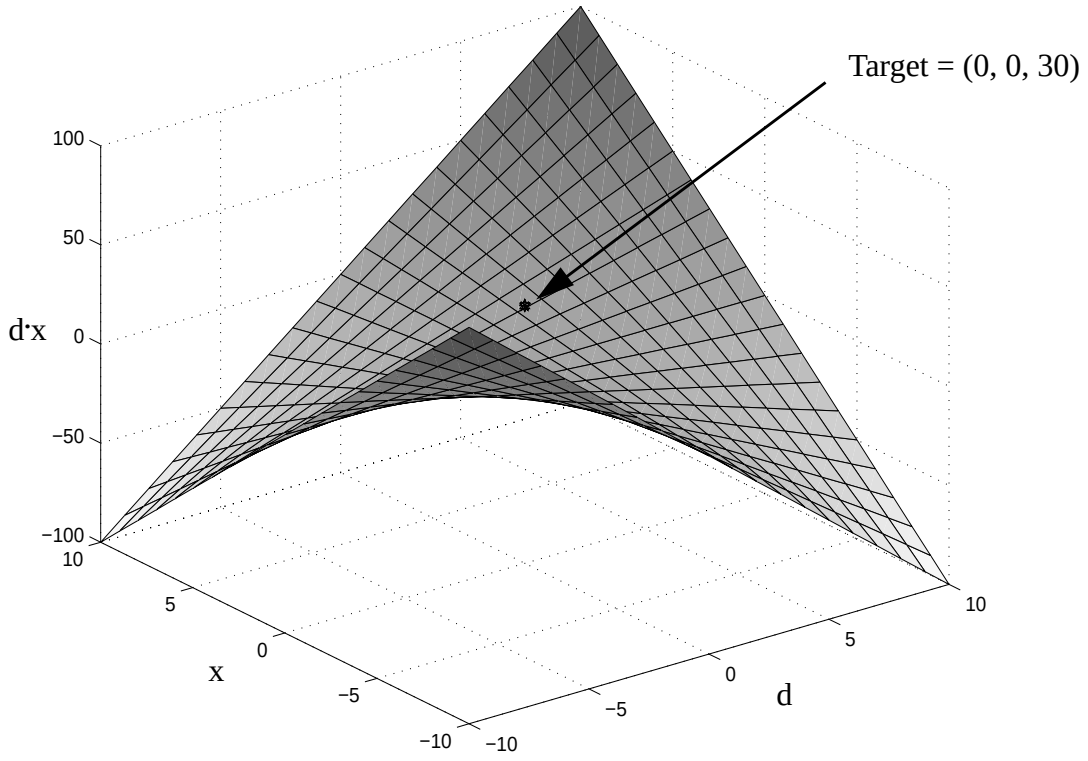
$$W_t = diag\begin{bmatrix}1\\1\end{bmatrix},\ A_t = \begin{bmatrix}1\\0\end{bmatrix},\ b_t = \begin{bmatrix}0\\1\end{bmatrix},$$

$$W_\rho = diag\begin{bmatrix}1\end{bmatrix},\ A_\rho = \begin{bmatrix}1\end{bmatrix},\ b_\rho = \begin{bmatrix}0\end{bmatrix}.$$

This structure creates a convenient model where one output is *x*, another is *d*, and the third is their product. Figure 5-1 plots the achievable output space for this system. Notice that cross sections taken along the *d* and *x* axes remain linear, while the full three dimensional

surface is nonlinear. This is because the time-based models produce second order terms through cross terms only, and not through squared terms.

Figure 5-1: Achievable outputs for the example system



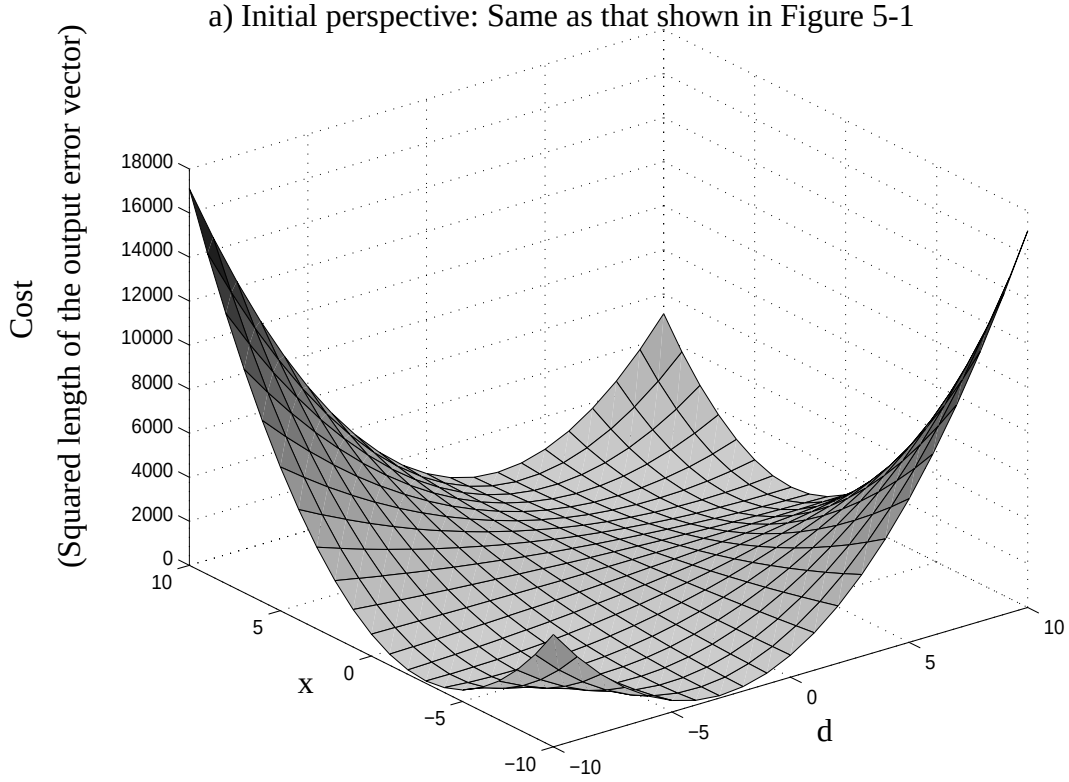
The iterative back-solver progresses by moving along these linear cross sections. There are certainly other gradient descent types of algorithms, some of which are likely to converge on solutions more quickly. However, this method makes excellent use of the existing linear solver software [Moy95] and is easily implemented.

To get a sense of how the solver works, consider a target output of (d = 0, x = 0, d·x = 30), as shown in Figure 5-1. Clearly this output is not achievable, since zero times zero is not thirty. To find minimum error solutions, we need to look for points on the achievable surface where the error vector (the vector between the target and the achievable point) is orthogonal to the surface gradient at that point. These will be the locations where the sum

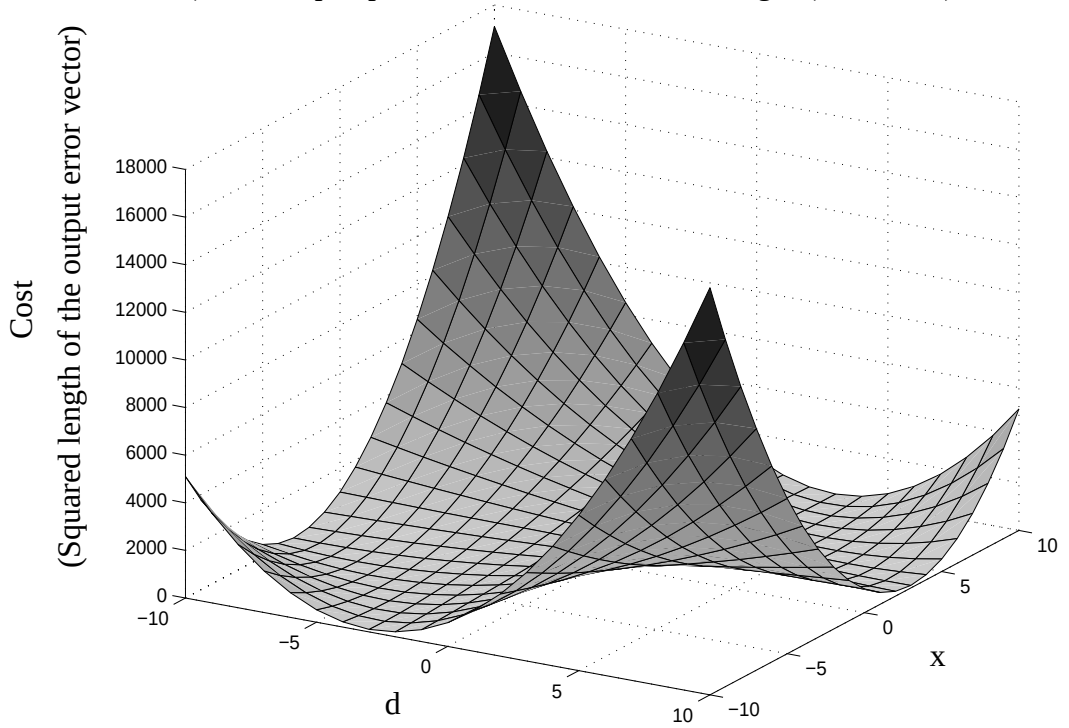
of squared error function is either a local minimum, local maximum, or both, in the case of a saddle point. Equivalently, the gradient of the cost function (sum of squared error) should be equal to zero at these points. Figures 5-2-a and 5-2-b show the cost as a function of the inputs. Note that, while solutions involving negative time and negative rates do not make any sense, the reader is asked to suspend his or her disbelief for the sake of mathematical rigor. These issues are taken care of through the use of constraints on the allowable ranges for d and $\boldsymbol{x}$.

Figure 5-2: Squared Error as a function of the inputs (d,x)

a) Initial perspective: Same as that shown in Figure 5-1



b) Second perspective: Rotated from left to right (clockwise)



By looking at these plots, we can picture three possible locations on the achievable surface where the cost function has a slope of zero. There are two local minima on the "upslopes" of the achievable surface, and a saddle point at the very center of the graph (d = 0, x = 0). It is interesting to note that the cost function surface is guaranteed to have a unique minimum along any cross section where *d* or *x* is held constant, yet the joint surface is **not** guaranteed to have a unique minimum. The uniqueness of the cross sectional minima is seen by noting that the achievable surface for any cross section is linear. Given a target vector, there is always a unique point on a linear surface that minimizes the distance between that point and the target. Thus the cross sections of the cost function along the defined axes are always parabolic with unique minima. Counter to intuition, the complete cost function surface need not have a unique minimum. Look carefully at the cost function plots above. The cross sections along both the *d* and the *x* axes are indeed parabolas, yet the cost function surface does not have a unique minimum. The construction does not necessarily yield a paraboloid due to the fact that saddle points are still possible. A local maximum is not possible however, since there must be orthogonal cross sections that contain only unique local minima. Thus it will always be saddle points that create the opportunity for multiple local minima.

The solver should return one of the local minima from the cost function, preferably the global minimum. The iterative solver is only guaranteed to find a local minimum, which will not necessarily be the global minimum. Starting with a (somewhat arbitrarily) selected initial input vector, $\boldsymbol{x_0}$ = (10, 10), Figures 5-3 and 5-4 plot the iterative solver's trajectory as it searches for an optimal input vector. Figures 5-3-a and 5-3-b show the trajectory in the output space, while Figures 5-4-a and 5-4-b display the trajectory along the

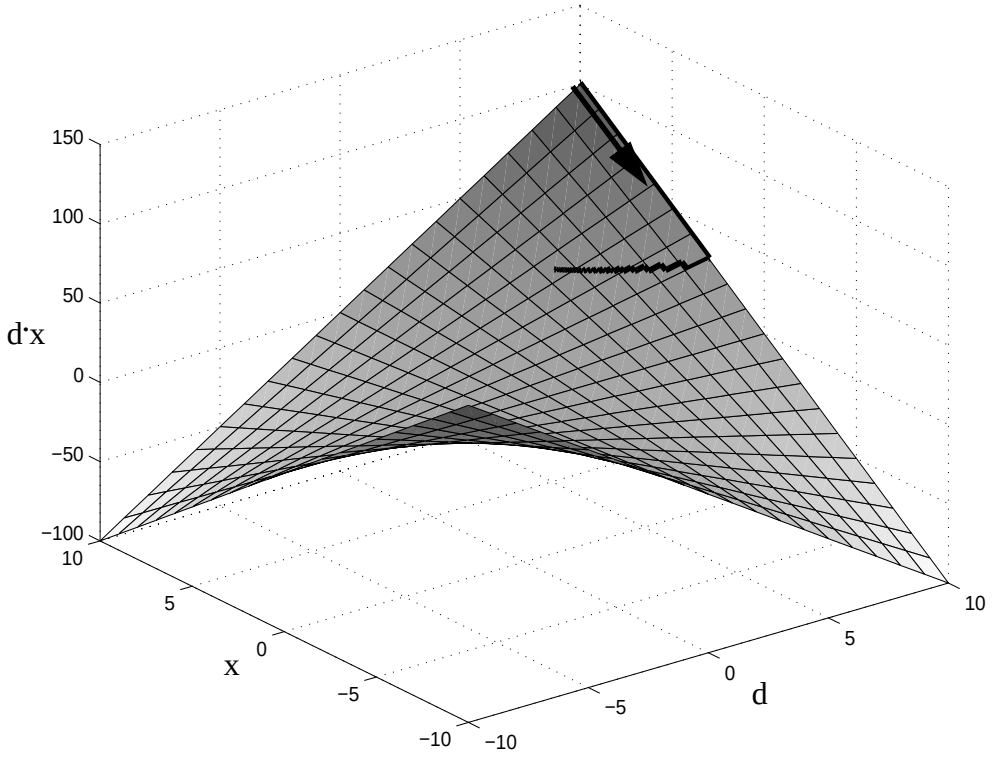
cost function. In a run-to-run control scenario the initial starting point is typically selected to be the inputs that were found in the previous optimization step. For this example, the locally optimal output vector is found to be approximately $\boldsymbol{y}$ = (5.385, 5.385, 29.00).

The figures clearly display the iteration steps along orthogonal axis vectors. Each line segment parallels either the time axis or the *x* axis. It is important to remember that the algorithm is not limited to stepping along a single axis. During one step, all non-time inputs are optimized simultaneously, then all time inputs are optimized simultaneously. Since graphs with more than three dimensions are difficult to plot, the example problem has a single time input and a single non-time input. In this case, the solver moves in only one dimension at a time. The epi control problem described in this work uses nine non-time inputs and one time input. Thus, when time is held constant, all nine non-time inputs are optimized simultaneously using the linear solver.

Note that the first step in the optimization does not follow the steepest descent vector. This is because the optimizer must make steps along the axis dimensions; “off angle” trajectories are not allowed. Note however that this only limits the first step in the optimization process. After an optimization step occurs along one dimension, that dimension cannot contribute to the subsequent gradient vector, and therefore will not show up as part of a gradient descent direction vector. Thus the iterative solver is a form of gradient descent algorithm after the first step is made.

Figure 5-3: Iterative solver's trajectory in the output space

a) Initial perspective: Same as that shown in Figure 5-1



b) Second perspective: Zoom in on the final trajectory

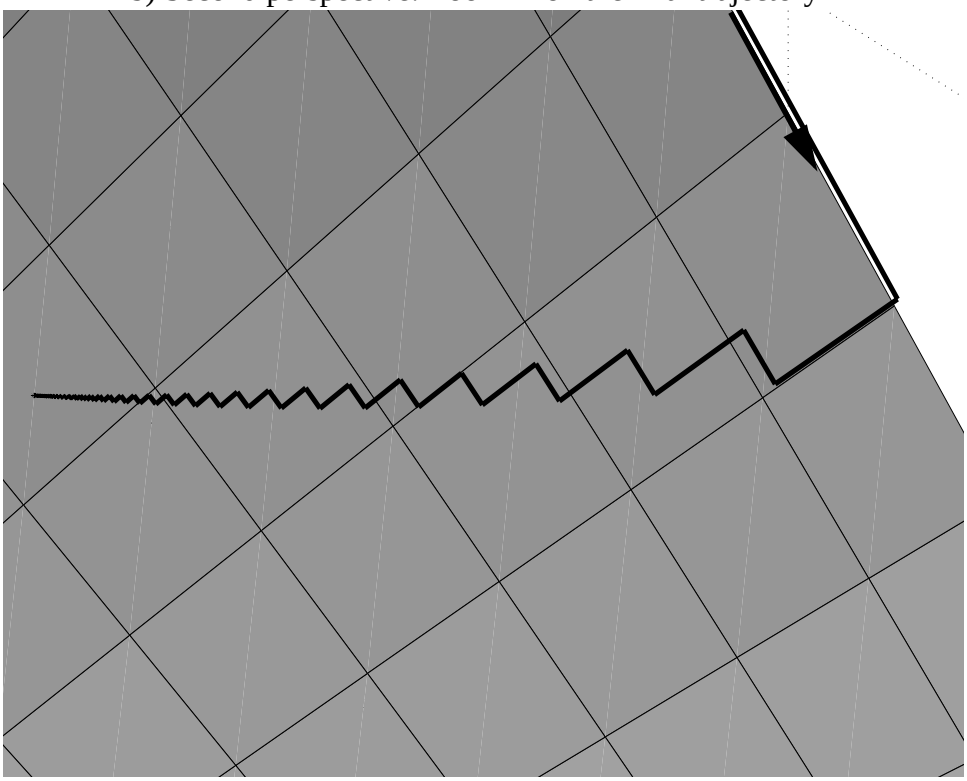
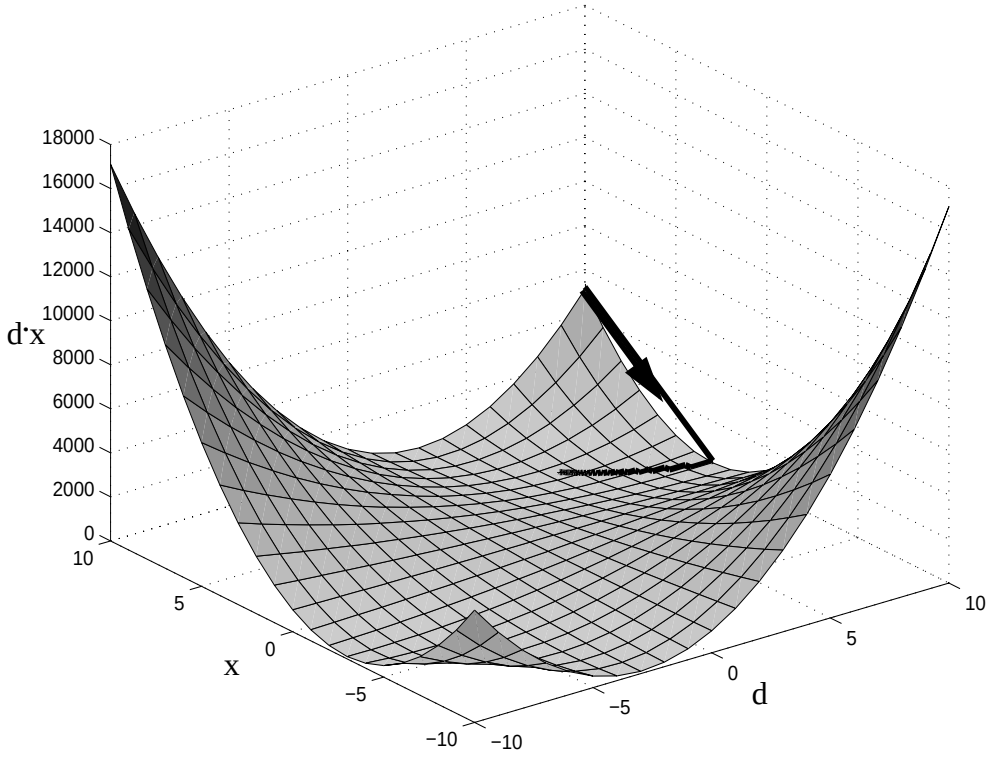
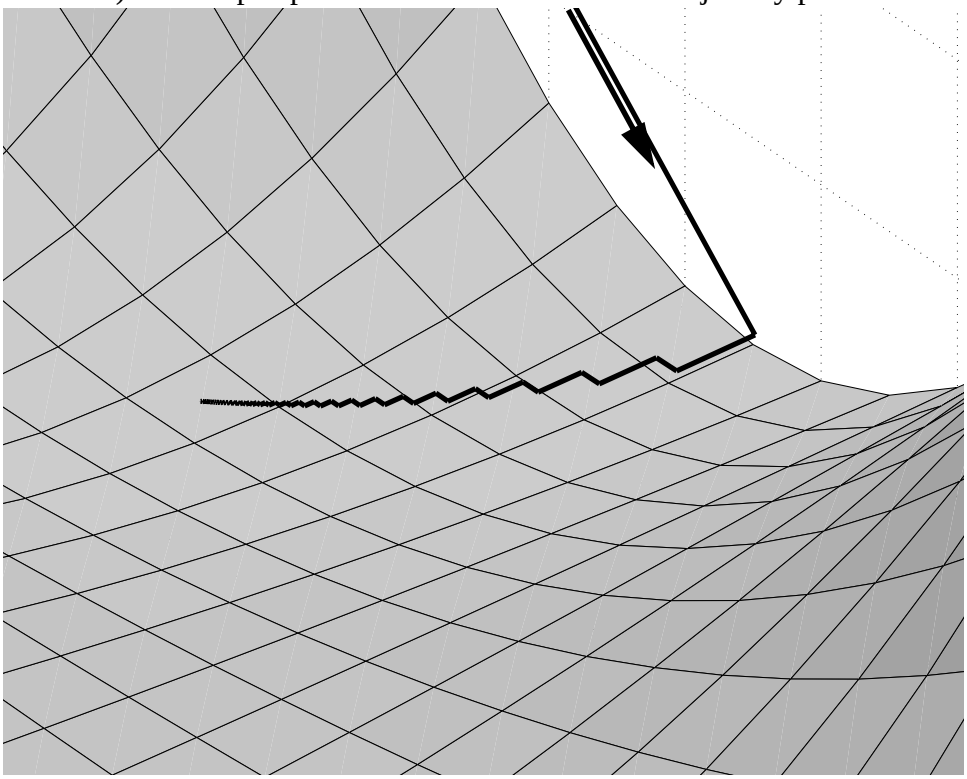


Figure 5-4: Iterative solver's trajectory in the Cost Function space

a) Initial perspective: Same as that shown in Figure 5-2



b) Second perspective: Zoom in on the final trajectory path



This brings up a question: in which direction should the first step proceed, the time dimension(s) or the non-time dimension(s)? Looking at the example, there is no clear way to decide. The displayed trajectory makes its first step in the non-time dimension, while holding the time value constant. Clearly the step could have been made along the time dimension first, which (for this example) would have resulted in a "mirror image" type of solution that looks much like the one shown in Figures 5-3 and 5-4. There are at least three different approaches that make sense, each with a varying degree of complexity and computational expense:

- Pick one and "go with it."

  The simplest approach is to consistently pick the same first step direction every time. This method is currently used by the controller, as it appears relatively unlikely that it will affect the final solution in most cases. (Currently the non-time inputs are optimized first, for no reason in particular.)
- Follow the steepest of the two gradients.

  Calculate the gradients along both the time and non-time axes, then follow the steepest of the two paths. This option is only slightly more computationally expensive than the first option, but adds more algorithm complexity.
- Try them both and pick the best results.

  Perform two different complete solving passes, one that first modifies time input(s) and a second that first modifies non-time input(s); follow each of them to completion. It is possible that each would lead to two different local minima, from which the smaller one should be selected. This option is the most computationally expensive, but can be implemented without adding much algorithmic complexity.

5.2 Solution for a Special Case of Overdetermined Systems

Optimal solutions for an overdetermined system must simultaneously solve the equations used in the iterative solver described above. Namely, these equations are

$$\boldsymbol{x} = (\boldsymbol{A}^t \boldsymbol{W}^2 \boldsymbol{A})^{-1} \boldsymbol{A}^t \boldsymbol{W}^2 (\boldsymbol{T} - \boldsymbol{b})$$

$$\text{and } d = \frac{\boldsymbol{T}_t^t \boldsymbol{W}_t^2 \boldsymbol{r}_t}{\boldsymbol{r}_t^t \boldsymbol{W}_t^2 \boldsymbol{r}_t},$$

where the target vector $\boldsymbol{T}$ and the weight matrix $\boldsymbol{W}$ incorporate the deposition time(s) d as described in Equation 5-2, and the deposition rates $\boldsymbol{r}$ incorporate the non-time inputs $\boldsymbol{x}$ as described in Equation 5-6. There does not appear to be a convenient closed form solution that can simultaneously solve these equations for optimal $\boldsymbol{x}$ and d vectors. Clearly there can be multiple solutions to these equations, as the simple example of Section 5.1.4 demonstrated.

There is however, a relatively common case where a simple single-step solution does exist. Consider a set of rate-based outputs that are **all** multiplied by the same time input d, which looks like

$$\boldsymbol{y}_t = d(\boldsymbol{A}_t \boldsymbol{x} + \boldsymbol{b}_t),$$

where d is a scalar and $\boldsymbol{x}$ is a vector. All outputs from the linear models must be multiplied by (the same) processing time. Such a system is quite common. Consider a simplified version of the epi control problem presented in this work, where the multiple thickness outputs are controlled, but the resistivities are not. This is exactly the type of special case considered below.

For this problem, it is useful to consider $\boldsymbol{x}$ as a set of coefficients that are supplying linear combinations of the column vectors in $\boldsymbol{A}_t$ in the output space, as in

$$\boldsymbol{y}_t = d\left(\begin{bmatrix}\boldsymbol{A}_1 & \cdots & \boldsymbol{A}_N\end{bmatrix}\begin{bmatrix}x_1 \\ \cdots \\ x_N\end{bmatrix} + \boldsymbol{b}_t\right) = \left(\begin{bmatrix}\boldsymbol{A}_1 & \cdots & \boldsymbol{A}_N\end{bmatrix}\begin{bmatrix}x_1 \\ \cdots \\ x_N\end{bmatrix} d + \boldsymbol{b}_t d\right) = \left(\begin{bmatrix}\boldsymbol{A}_1 & \cdots & \boldsymbol{A}_N & \boldsymbol{b}_t\end{bmatrix}\begin{bmatrix}d \cdot x_1 \\ \cdots \\ d \cdot t x_N \\ d\end{bmatrix}\right).$$

The slope matrix and input vector can be redefined to yield a new system:

$$\boldsymbol{y_t} = \boldsymbol{A_{tb}}\boldsymbol{x_d}, \text{ where } \boldsymbol{A_{tb}} = \left[\boldsymbol{A}_1 \ \ldots \ \boldsymbol{A}_N \ \boldsymbol{b_t}\right] \text{ and } \boldsymbol{x_d} = \begin{bmatrix} d \cdot x_1 \\ \cdots \\ d \cdot x_N \\ d \end{bmatrix} \qquad \textbf{(Eq 5-10)}$$

This is simply a linear system in the new input vector. Thus the use of a single time input multiplying all of the outputs results in what appears to be a **linear** system! It is true that the system is nonlinear in $\boldsymbol{x}$ and d, but the achievable outputs from the system form a linear subspace. Essentially the set of construction vectors $\boldsymbol{A_n}$ is augmented with the offset vector $\boldsymbol{b}$. As long as the offset vector is not included in the space spanned by $\boldsymbol{A_n}$, then the new achievable subspace will increase by one dimension.

In the redefined input space, the problem simply becomes the minimum distance solution between a target point $\boldsymbol{T}$ and the linear surface, which is

$$\boldsymbol{x_{bopt}} = (\boldsymbol{A_{tb}^t}\boldsymbol{A_{tb}})^{-1}\boldsymbol{A_{tb}^t}(\boldsymbol{T}). \qquad \textbf{(Eq 5-11)}$$

The optimal time setting d is found by simply stripping off from the last term from the $\boldsymbol{x_{bopt}}$ vector. The non-time inputs $\boldsymbol{x}$ are then found by dividing the remaining vector by d. Output weighting may be included by simply premultiplying the target vector $\boldsymbol{T}$ and the slope matrix $\boldsymbol{A_{tb}}$ with the diagonal weight matrix.

The catch is, of course, that d must not be equal to zero, which would result in a divide by zero condition. This problem is dealt with by placing proper constraints on d. While d equalling zero causes mathematical problems, any processing time less than or equal to zero does not make sense when dealing with a real system. Reasonable bounds on d must be provided, and bounds checking needs to occur before the non-time inputs are found.

5.3 Solutions for Exactly Determined and Underdetermined Systems

As described in Section 5.1.3, the iterative solver will select a minimum cost solution for underdetermined (and exactly determined) systems. For the underdetermined case, however, there are an infinite set of solutions that will minimize the output cost function (achieve the target outputs). We want to provide a framework for selecting a unique optimal solution to this type of problem, as has been done for systems of linear models. The optimization strategy selects the input vector that simultaneously exactly solves for the output target vector and is the minimum (weighted) distance from a preselected starting input vector. The well known result is shown in Equation 5-4 and is derived in a number of sources, including [Moy95].

Here we consider the optimization strategy for overdetermined systems of the time-based models used in this work. Section 5.3.1 formulates the problem with a convenient notation, while Sections 5.3.2 and 5.3.3 consider the exactly determined solution and the underdetermined case, respectively.

5.3.1 Problem Formulation

In formulating solutions for exactly determined and underdetermined systems, the notation defined below is particularly useful. First, we separate the output models into groups that are multiplied by the same processing time, along with one group for all outputs that are not multiplied by any of the time inputs. Since we are looking at underdetermined systems, it can be assumed that at least one solution exists that will achieve the target vector. The following equations are defined:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \dots \\ x_N \end{bmatrix}, \ \boldsymbol{d} = \begin{bmatrix} d_1 \\ \dots \\ d_P \end{bmatrix}, \ \begin{bmatrix} \boldsymbol{A_0 x} + \boldsymbol{b_0} = \boldsymbol{T_0} \\ d_1(\boldsymbol{A}_1 \boldsymbol{x} + \boldsymbol{b}_1) = \boldsymbol{T}_1 \\ \dots \\ d_P(\boldsymbol{A}_P \boldsymbol{x} + \boldsymbol{b}_P) = \boldsymbol{T}_P \end{bmatrix},$$

where there are *N* non-time inputs $\boldsymbol{x}$, *P* time inputs $\boldsymbol{d}$, the $\boldsymbol{A_p}$'s are the grouped slope matrices, the $\boldsymbol{b_p}$'s are grouped offset vectors, and the $\boldsymbol{T_p}$'s are grouped target vectors. There are *M* outputs, which must be less than or equal to the sum *N+P*. Now reformulate the problem in terms of "inverse-time" variables, where $r_p = 1/d_p$:

$$\begin{bmatrix} \boldsymbol{A_0 x} + \boldsymbol{b_0} = \boldsymbol{T_0} \\ \boldsymbol{A}_1 \boldsymbol{x} + \boldsymbol{b}_1 = (\boldsymbol{T}_1) r_1 \\ \dots \\ \boldsymbol{A}_P \boldsymbol{x} + \boldsymbol{b}_P = (\boldsymbol{T}_P) r_P \end{bmatrix} \Rightarrow \begin{bmatrix} \boldsymbol{A_0 x} + (\boldsymbol{0}) r_1 + \dots + (\boldsymbol{0}) r_P = \boldsymbol{T_0} - \boldsymbol{b_0} \\ \boldsymbol{A}_1 \boldsymbol{x} - (\boldsymbol{T}_1) r_1 + \dots + (\boldsymbol{0}) r_P = -\boldsymbol{b}_1 \\ \dots \\ \boldsymbol{A}_P \boldsymbol{x} + (\boldsymbol{0}) r_1 + \dots - (\boldsymbol{T_P}) r_P = -\boldsymbol{b}_P \end{bmatrix}.$$

This looks like a linear system in the $\boldsymbol{x}$ and $\boldsymbol{r}$ vectors, where the following matrices and vectors are defined:

$$\boldsymbol{A_x} = \begin{bmatrix} \boldsymbol{A_0} \\ \boldsymbol{A_1} \\ \dots \\ \boldsymbol{A_P} \end{bmatrix}, \ \boldsymbol{r} = \begin{bmatrix} r_1 \\ \dots \\ r_P \end{bmatrix}, \ \boldsymbol{A_r} = \begin{bmatrix} 0 & \dots & 0 \\ -\boldsymbol{T_1} & \dots & 0 \\ & \dots & \\ 0 & \dots & -\boldsymbol{T_P} \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} \boldsymbol{T} - \boldsymbol{b_0} \\ -\boldsymbol{b_1} \\ \dots \\ -\boldsymbol{b_P} \end{bmatrix}, \ \boldsymbol{A_x x} + \boldsymbol{A_r r} = \boldsymbol{c}. \qquad \textbf{(Eq 5-12)}$$

Finally, putting all of the variables together in one matrix yields

$$\boldsymbol{x_r} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{r} \end{bmatrix}, \ \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A_x} & \boldsymbol{A_r} \end{bmatrix}, \ \boldsymbol{A x}_r = \boldsymbol{c}. \qquad \textbf{(Eq 5-13)}$$

Solutions to these linear equations are the set of inputs that achieve the desired target vector.

5.3.2 The Exactly Determined Solution

First consider the exactly determined case, where the number of outputs *(M)* equals the number of inputs (*N+P*). Notice that the construction described above creates a new slope

matrix $\boldsymbol{A}$ whose dimensions are M by $N+P$. The exactly determined case generates a square slope matrix, and the solution for all inputs can be found as

$$\boldsymbol{x}_r = A^{-1}\boldsymbol{c}\,. \qquad \textbf{(Eq 5-14)}$$

The non-time inputs are directly available and the time inputs can be found by inverting the "inverse-time" inputs. We must be careful of solutions where the "inverse-time" inputs are equal to zero, which leads to solutions involving infinite time. Again, setting and checking for proper input bounds will avoid this problem.

It is worth noting that this solution has no constraints on how many time inputs (or non-time inputs) are in the system. There can be multiple time inputs acting on any number of different outputs. Outputs that are not multiplied by any time input are also permitted. The exactly determined system is currently the only scenario where an elegant single-step solution strategy holds for all configurations of the time-based model.

5.3.3 The Underdetermined Case

Turning to the fully underdetermined case, there are an infinite number of solutions that will achieve a given target output vector. As is common practice, input weighting vectors and initial input vectors will be used to form a cost function in the input space. A constrained optimization process selects a set of inputs that achieve the target outputs while minimizing the weighted distance from the initial input vector. The cost function we wish to minimize is given by

$$Cost = (\boldsymbol{x}-\boldsymbol{x}_0)^t\boldsymbol{V}_x^2(\boldsymbol{x}-\boldsymbol{x}_0) + (\boldsymbol{d}-\boldsymbol{d}_0)^t\boldsymbol{V}_d^2(\boldsymbol{d}-\boldsymbol{d}_0)\,, \qquad \textbf{(Eq 5-15)}$$

where $\boldsymbol{V_x}$ is the diagonal input weighting matrix for the non-time inputs, $\boldsymbol{x_0}$ is the initial non-time input vector, $\boldsymbol{V_d}$ is the diagonal input weighting matrix for the time inputs, and

$\boldsymbol{d_0}$ is the initial time input vector. This cost function should be minimized over the set of solutions

$$\boldsymbol{A_x x} + \boldsymbol{A_r}\left\{\frac{1}{\boldsymbol{d}}\right\} = \boldsymbol{c}\,, \qquad \textbf{(Eq 5-16)}$$

which is a slightly rewritten version of Equation 5-12. The curly brace notation, $\{f(\boldsymbol{matrix})\}$, is simply an element by element operation on the matrix (or vector).

The constrained optimization problem can be attacked with the use of Lagrange multipliers (λ) as follows:

$$L = \frac{1}{2}[(\boldsymbol{x}-\boldsymbol{x_0})^t \boldsymbol{V_x^2}(\boldsymbol{x}-\boldsymbol{x_0}) + (\boldsymbol{d}-\boldsymbol{d_0})^t \boldsymbol{V_d^2}(\boldsymbol{d}-\boldsymbol{d_0})] + \lambda^t\left(\boldsymbol{A_x x} + \boldsymbol{A_r}\left\{\frac{1}{\boldsymbol{d}}\right\} - \boldsymbol{c}\right) \qquad \textbf{(Eq 5-17)}$$

Taking the partial derivatives and setting them to zero, we have

$$\frac{\partial L}{\partial \boldsymbol{x}} = (\boldsymbol{x}-\boldsymbol{x_0})^t \boldsymbol{V_x^2} + \lambda^t \boldsymbol{A_x} = 0\,, \qquad \textbf{(Eq 5-18)}$$

$$\frac{\partial L}{\partial \boldsymbol{d}} = (\boldsymbol{d}-\boldsymbol{d_0})^t \boldsymbol{V_d^2} + \lambda^t \boldsymbol{A_r} diag\left\{\frac{-1}{\boldsymbol{d}^2}\right\} = 0\,, \text{ and} \qquad \textbf{(Eq 5-19)}$$

$$\frac{\partial L}{\partial \lambda} = \boldsymbol{A_x x} + \boldsymbol{A_r}\left\{\frac{1}{\boldsymbol{d}}\right\} - \boldsymbol{c} = 0\,. \qquad \textbf{(Eq 5-20)}$$

Beginning with Equation 5-18, we can take the transpose and perform the following manipulations:

$$\boldsymbol{A_x^t}\lambda = \boldsymbol{V_x^2}(\boldsymbol{x_0}-\boldsymbol{x})$$

$$\boldsymbol{V_x^{-2}}\boldsymbol{A_x^t}\lambda = (\boldsymbol{x_0}-\boldsymbol{x})\,.$$

Beginning with Equation 5-19, we can take the transpose and perform the following manipulations:

$$diag\left\{\frac{-1}{\boldsymbol{d}^2}\right\}\boldsymbol{A}_r^t\lambda = \boldsymbol{V}_d^2(\boldsymbol{d}_0 - \boldsymbol{d})$$

$$\boldsymbol{V}_d^{-2}diag\left\{\frac{-1}{\boldsymbol{d}^2}\right\}\boldsymbol{A}_r^t\lambda = (\boldsymbol{d}_0 - \boldsymbol{d})$$

$$diag\left\{\frac{-1}{\boldsymbol{d}^2}\right\}\boldsymbol{V}_d^{-2}\boldsymbol{A}_r^t\lambda = (\boldsymbol{d}_0 - \boldsymbol{d})$$

$$\boldsymbol{V}_d^{-2}\boldsymbol{A}_r^t\lambda = diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d}).$$

The order of multiplication for $\boldsymbol{V_d}^{-2}$ and $diag\{-1/\boldsymbol{d}^2\}$ can be swapped because both are diagonal matrices. The results of the last two equation blocks can be merged as follows:

$$\begin{bmatrix}\boldsymbol{V}_x^{-2}\boldsymbol{A}_x^t\\ \boldsymbol{V}_d^{-2}\boldsymbol{A}_r^t\end{bmatrix}\lambda = \begin{bmatrix}(\boldsymbol{x}_0 - \boldsymbol{x})\\ diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})\end{bmatrix}$$

$$\begin{bmatrix}\boldsymbol{A}_x & \boldsymbol{A}_r\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_x^{-2}\boldsymbol{A}_x^t\\ \boldsymbol{V}_d^{-2}\boldsymbol{A}_r^t\end{bmatrix}\lambda = \begin{bmatrix}\boldsymbol{A}_x & \boldsymbol{A}_r\end{bmatrix}\begin{bmatrix}(\boldsymbol{x}_0 - \boldsymbol{x})\\ diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})\end{bmatrix}.$$

Using the substitutions

$$\boldsymbol{A} = \begin{bmatrix}\boldsymbol{A}_x & \boldsymbol{A}_r\end{bmatrix}, \text{ and } \boldsymbol{V} = \begin{bmatrix}\boldsymbol{V}_x & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{V}_d\end{bmatrix},$$

where $(\boldsymbol{AA^t})$ is invertible because there are more inputs than outputs, we have

$$\boldsymbol{AV}^{-2}\boldsymbol{A}^t\lambda = \begin{bmatrix}\boldsymbol{A}_x & \boldsymbol{A}_r\end{bmatrix}\begin{bmatrix}(\boldsymbol{x}_0 - \boldsymbol{x})\\ diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})\end{bmatrix}$$

$$\boldsymbol{AV}^{-2}\boldsymbol{A}^t\lambda = \boldsymbol{A}_x(\boldsymbol{x}_0 - \boldsymbol{x}) + \boldsymbol{A}_r diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})$$

$$\lambda = (\boldsymbol{AV}^{-2}\boldsymbol{A}^t)^{-1}(\boldsymbol{A}_x\boldsymbol{x}_0 - \boldsymbol{A}_x\boldsymbol{x} + \boldsymbol{A}_r diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})).$$

Solving Equation 5-20 for $\boldsymbol{A_x x}$ and plugging into the last equation yields

$$\lambda = (\boldsymbol{AV}^{-2}\boldsymbol{A}^t)^{-1}\left(\boldsymbol{A}_x\boldsymbol{x}_0 - \boldsymbol{c} + \boldsymbol{A}_r\left\{\frac{1}{\boldsymbol{d}}\right\} + \boldsymbol{A}_r diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})\right).$$

Here λ is found in terms of $\boldsymbol{d}$ only, which can be substituted into Equation 5-19 as

$$\boldsymbol{A}_r^t\lambda = diag\{-\boldsymbol{d}^2\}\boldsymbol{V}_d^2(\boldsymbol{d}_0 - \boldsymbol{d})$$

$$\boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\left(\boldsymbol{A}_x\boldsymbol{x}_0 - \boldsymbol{c} + \boldsymbol{A}_r\left\{\frac{1}{\boldsymbol{d}}\right\} + \boldsymbol{A}_r diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})\right) = \boldsymbol{V}_d^2 diag\{-\boldsymbol{d}^2\}(\boldsymbol{d}_0 - \boldsymbol{d})$$

$$\boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\left(\boldsymbol{A}_x\boldsymbol{x}_0 - \boldsymbol{c} + \boldsymbol{A}_r\left\{\frac{1}{\boldsymbol{d}}\right\} + \boldsymbol{A}_r\{\boldsymbol{d}^3\} - \boldsymbol{A}_r\{\boldsymbol{d}_0\boldsymbol{d}^2\}\right) = \boldsymbol{V}_d^2\{\boldsymbol{d}^3\} - \boldsymbol{V}_d^2\{\boldsymbol{d}_0\boldsymbol{d}^2\}.$$

Combining the like terms in $\boldsymbol{d}$ we get

$$\begin{gathered}(\boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\boldsymbol{A}_r - \boldsymbol{V}_d^2)\{\boldsymbol{d}^3\} - (\boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\boldsymbol{A}_r - \boldsymbol{V}_d^2)\{\boldsymbol{d}_0\boldsymbol{d}^2\} + \ldots \\ \boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}(\boldsymbol{A}_x\boldsymbol{x}_0 - \boldsymbol{c}) + \boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\boldsymbol{A}_r\left\{\frac{1}{\boldsymbol{d}}\right\} = 0\end{gathered} \quad . \qquad \textbf{(Eq 5-21)}$$

These equations must be solved for the processing times $\boldsymbol{d}$. With P independent time inputs, Equation 5-21 provides P fourth order polynomial equations with P unknowns. In general, the equations cannot be solved independently. These nonlinear equations generate multiple solutions for the time inputs, the nature of which will be explored below. After finding solutions that solve these equations, the corresponding solutions for the non-time inputs are found, as was described in Section 5.1.1. Finally, the minimum cost (input change) solution can be selected from among the candidates.

While implementing a general purpose algorithm for solving these equations will prove difficult, there are some significant simplifications that follow if we can assume that there is a single time input d. This assumption means that both d and V_d are scalars, and that $\boldsymbol{A}_r$ is a vector, leading to the following simplifications for Equation 5-21:

$$\begin{gathered}(\boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\boldsymbol{A}_r - V_d^2)d^4 - (\boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\boldsymbol{A}_r - V_d^2)d_0d^3 + \ldots \\ \boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}(\boldsymbol{A}_x\boldsymbol{x}_0 - \boldsymbol{c})d + \boldsymbol{A}_r^t(\boldsymbol{A}\boldsymbol{V}^{-2}\boldsymbol{A}^t)^{-1}\boldsymbol{A}_r = 0\end{gathered} \qquad \textbf{(Eq 5-22)}$$

This is a fourth order polynomial in *d*, so there are four different solutions that will satisfy it. There are well known algorithms for solving polynomials in one variable, so implementation should not be an issue.

It is worth noting that this solution is **not** limited to systems that multiply **all** of their outputs by a single time input, as was the condition for the overdetermined special case solution. This single time input solution for the underdetermined case is valid for situations where all or some of the outputs are multiplied by the (single) time input. This type of solver could be implemented for the multi-objective epi control problem if there were more inputs than outputs. However, since this is not the case, a source code version of this solver was not implemented.

Some Example Solutions

While the above derivations mathematically find solutions to the underdetermined, time-based optimization problem, they give very little sense of what the solutions "look" like. A few simple examples will give the reader a better sense of what is really going on.

First consider a very simple underdetermined system:

$$y = d \cdot (x + 1),$$

where there is a single output *y*, a single time input *d*, and a single non-time input *x*. For any given target output *T* there will be an infinite set of inputs that can achieve the target. Assuming a target of one ($T = 1$), the solution space looks like

$$d \cdot (x + 1) = 1$$

$$x + 1 = \frac{1}{d}$$

$$x - \frac{1}{d} = -1,$$

which can be rewritten in terms of an “inverse time” input as

$$x - r = -1 .$$

Figures 5-5 and 5-6 show the input solution space using the “inverse time” and time dimensions, respectively. There is a discontinuity in the actual solution space where the “inverse time” input crosses zero. Other than that exact point, there is a one-to-one mapping between the solution spaces in the two figures.

Figure 5-5: Solution space using the “inverse time” dimension

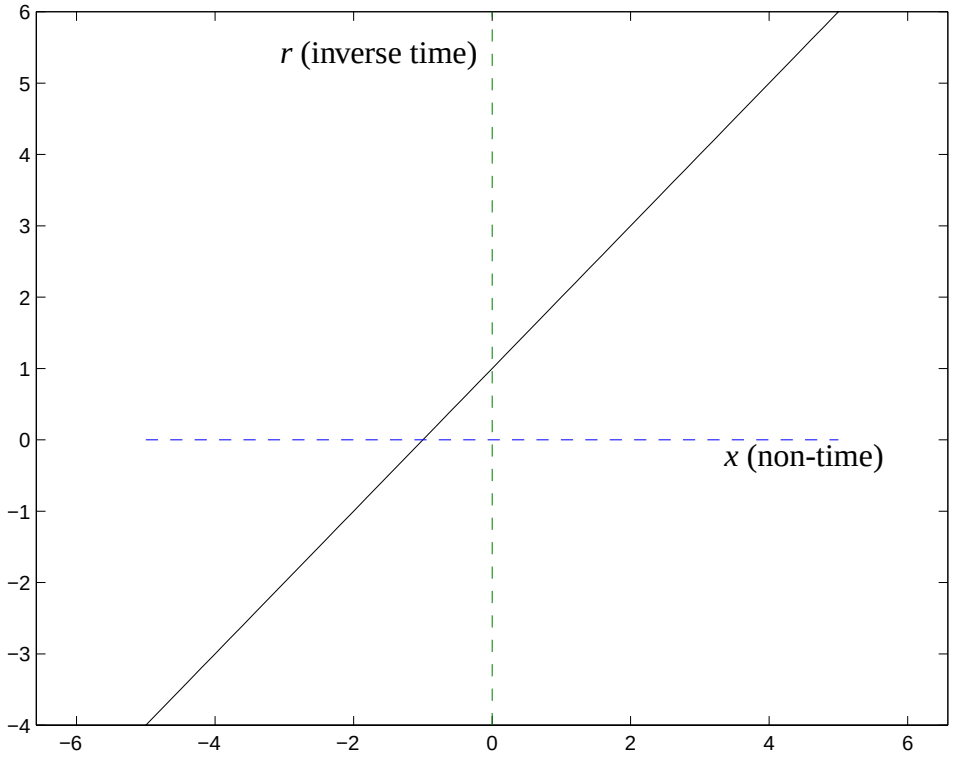
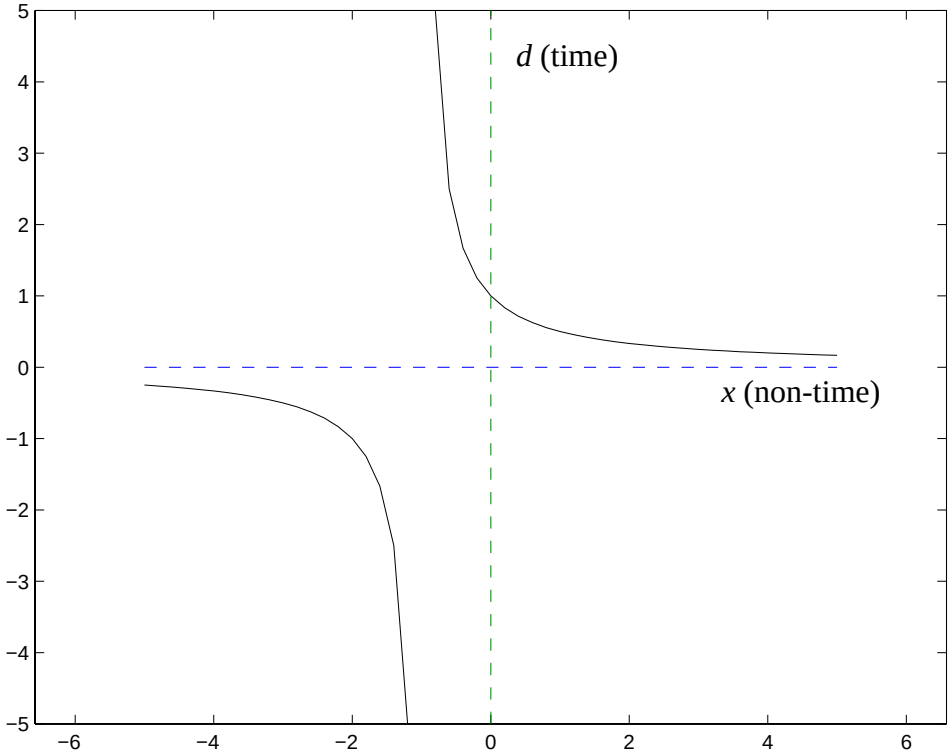


Figure 5-6: Solution space (actual) using the time dimension

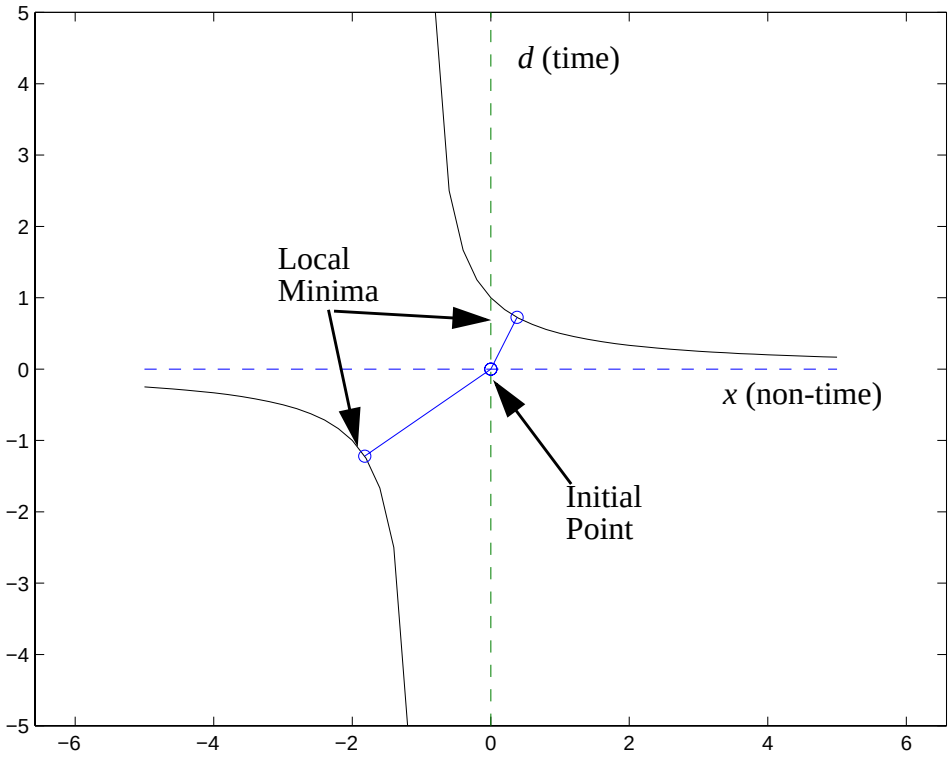


The underdetermined solver selects inputs that lie in the solution space, while minimizing the distance between those selected inputs and the given initial input vector. These points are achieved at locations where the error vector between the selected input and the initial input vector is normal to the solution space's gradient surface at that point. We must be somewhat careful with this description, as local maxima also meet the selection criterion.

Some sample solutions will help make this concept more clear. First consider selecting an initial vector that is positioned "between" the hyperbolic surfaces shown in Figure 5-6, such as the point ($x_0 = 0$, $d_0 = 0$). (By "between," we refer to points that are not within either of the two convex regions formed by the solution space.) Figure 5-7 plots this point and the solutions found by solving the fourth order polynomial from Equation 5-22. Note

that the error vectors between the initial point and the two optimal sites are orthogonal to the solution surface. The two solutions are both local minima, and the solution from the positive time half-space is the global minimum.

Figure 5-7: Possible solutions for the starting vector ($x_0 = 0$, $d_0 = 0$)

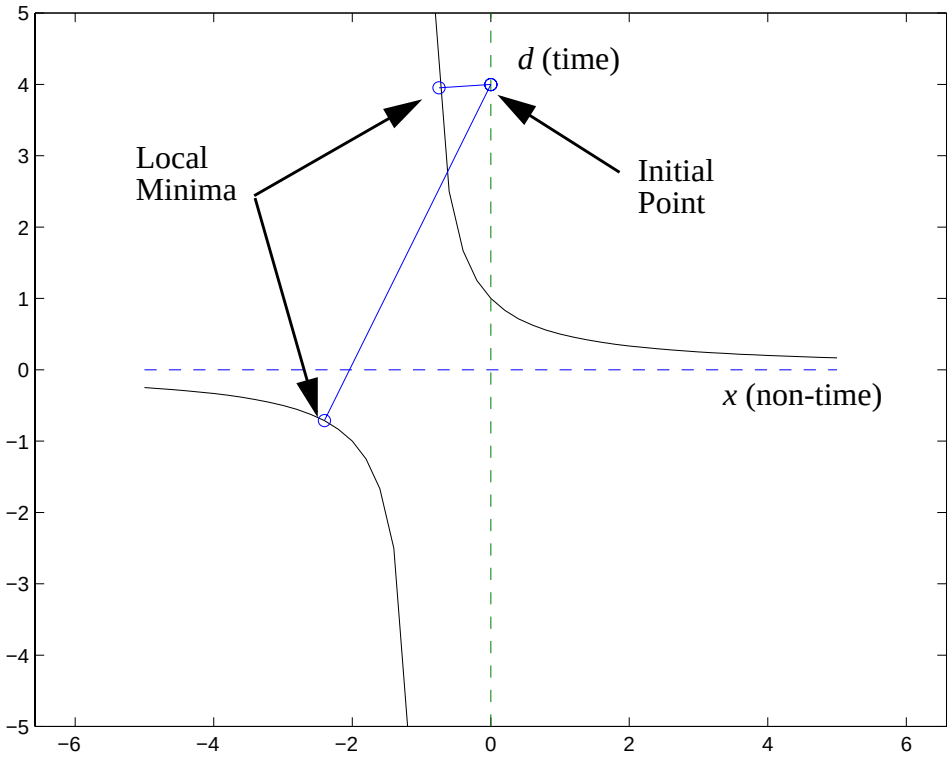


Clearly the geometry of this problem is such that any initial vector residing between the hyperbolic surfaces will yield two local minima, one from the positive time region and one from the negative time region. Remember however that a fourth order polynomial was solved to find these points, meaning that four solutions were found. Two of those solutions were found to be imaginary and need not be considered. At any location where only two solutions are possible, the other two solutions will be either imaginary, or duplicates of a valid solution.

Next, imagine selecting an initial point that lies “within” one of the hyperbolic regions.

Intuitively, there are two different scenarios. Figure 5-8 shows the solutions where the starting vector is set as ($x_0 = 0$, $d_0 = 4$). Again we see that there are two solutions found as local minima. Clearly the solution whose time input has the same sign as the initial point is the global minimum.

Figure 5-8: Possible solutions for the starting vector ($x_0 = 0$, $d_0 = 4$)

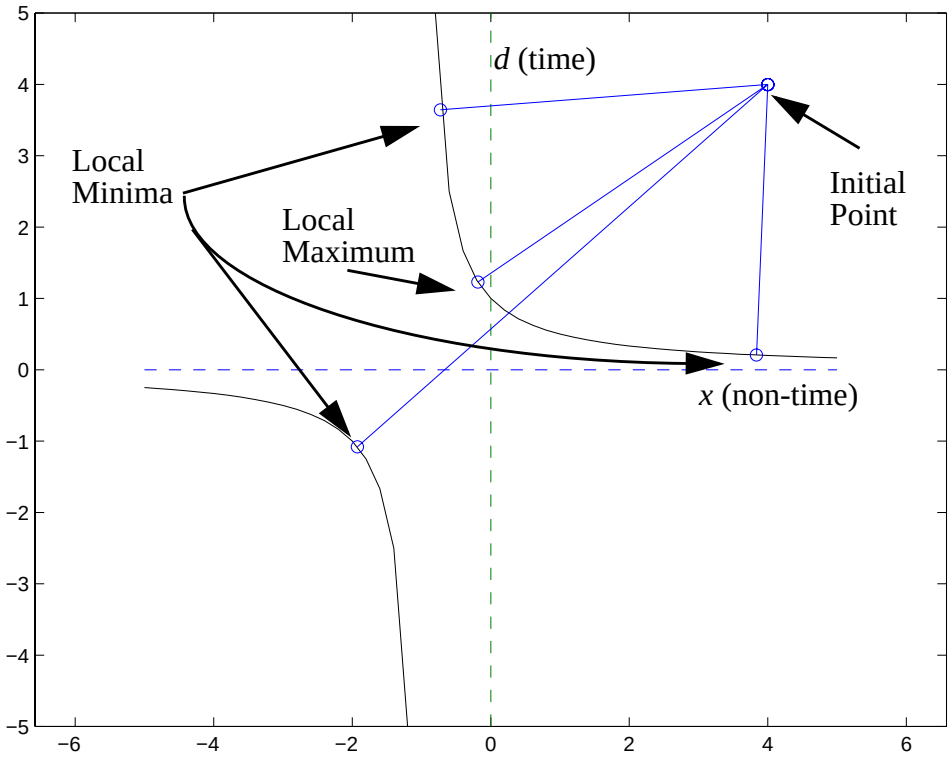


The solutions shown in Figure 5-8 are for an initial vector that is relatively "close" to the hyperbolic surface, where "closeness" is not well defined yet. Looking at the figure, it does appear that there are no other points on the solution space where the error vector would be orthogonal to the derivative of the solution curve.

As we think about moving the initial point further away from the hyperbolic surface and closer to the positively sloped line of symmetry, we can start to imagine how there might be two local minima in the same hyperbolic region. Figure 5-9 demonstrates this

case by presenting solution vectors for the initial point (x_0 = 4, d_0 = 4). Finally here is a situation where all four solutions are real. In this case, there are three local minima and one local maximum. The global minimum is selected from the two minima that are on the same side of the axis line, d = 0.

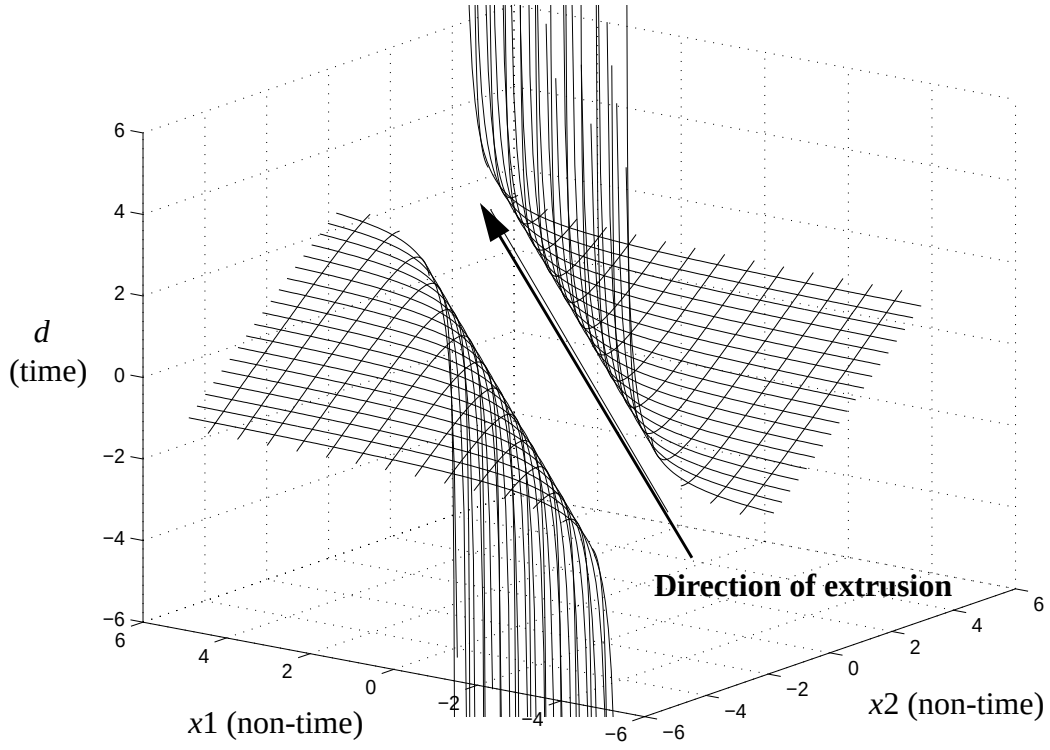
Figure 5-9: Possible solutions for the starting vector (x_0 = 4, d_0 = 4)



To extend these examples to higher dimensioned problems, Figure 5-10 shows the input solution space for a system with one output, one time input, and two non-time inputs. A three input, one output system such as this generates a two dimensional solution surface. Actually, the scenario forms two disconnected solution surfaces, which are always separated by the plane, d = 0. This is somewhat difficult to see from the perspective shown here, but the plot does demonstrate the three dimensional structure of these surfaces. The surfaces can be viewed as the extrusion of a 2-D hyperbolic (1/x) surface along a vector in

the *d* plane, as shown in the figure.

Figure 5-10: Solution space for a single output, three input system



A minimum distance solution between an initial point ($x1_0$, $x2_0$, d_0) and these surfaces must lie in the plane that passes through the initial point and is normal to the extrusion vector. This intersection forms a two dimensional solution surface much like those found in Figures 5-6 through 5-9. Again, the minimum distance solution will be found by solving a fourth order polynomial in *d*, and then selecting the smallest minimum from the two or three (real) local minima. This concept can be extended to higher dimensioned non-time input vectors (***x***) to explain why, when there is only one time input (dimension), the roots of a fourth order polynomial will provide all possible minimum cost solutions.

5.4 Summary

An automated run-to-run control solution for the epi control problem requires the integration of process optimization software. Specifically, back-solving routines for time-based models need to be developed and implemented. While general nonlinear optimization packages exist, they suffer from a number of problems, including:

- Expensive
- Difficult to integrate into a larger software system
- Performance might be improved with a solver that has special knowledge of the underlying optimization problem

An iterative solution to the time-based solver problem has been developed and analyzed for use with the epi control problem. The algorithm utilizes a simple extension to the "standard" back-solver for linear systems, and as such, it is relatively simple to create and integrate.

The aggressive use of multiple response surface modeling for the epi control problem makes it an inherently overdetermined system. Also, its combined use of time-based models (for thickness) and standard linear models (for resistivity) precludes the use of any simplified single-step solution. However, for completeness, this chapter analyzed a special case of overdetermined systems, as well as exactly determined and underdetermined systems. The application of time-based models for other semiconductor processes could take advantage of these results.

The last two chapters have fully developed the experimental strategies and theoretical background for multi-objective uniformity control. However, before moving on to actual experimental results, it is worthwhile to consider software integration strategies for automated control. Keeping in mind that our final goal is **integrated** model-based run-to-run

control, the next chapter considers the packaging, distribution, and integration of our software packages that provide run-to-run control services.

Chapter 6

Distributed Run-to-Run Control: Interfaces and Objects

One important goal of this work is the exploration of Cell Control software integration using distributed object technologies. Until now, the various MIT run-to-run controller implementations have only been available through source code or as simple socket-based servers, all of which used single agglomerated interfaces [Moy95] [Gow96]. The use of low-level socket streams limited our ability to explore object-oriented interface constructs. The introduction of Microsoft's Component Object Model (COM) and the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) as integration technologies has reopened the question of how to best present an interface to the MIT EWMA controller and the time-based EWMA controller.

The following sections provide a brief introduction to distributed object interface technologies and applies them to the EWMA-based control problem. Section 6.1 provides some high-level distinctions between COM and CORBA technologies and discusses why a COM interface was required for this work. Section 6.2 gives a brief introduction to object interface design strategies, and Section 6.3 applies these strategies to the time-based EWMA control problem. This chapter focusses on high-level concepts, while detailed IDL specifications and implementation issues are left for Appendices A and B, respectively.

6.1 Distributed Object Interface Technologies: COM and CORBA

At this point there are primarily two popular competing technologies for creating and

manipulating distributed objects, Microsoft's Component Object Model (COM)[Red97] and the Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA)[OH98]. Both are open standards that provide comparable services for building distributed client/server infrastructures. One of the most important features these technologies provide is interoperability between computing platforms and programming languages. In our context, interoperability refers to the ability of software components to properly cooperate and communicate with each other regardless of the underlying operating system(s) or original development package(s) with which the components were created.

CORBA and COM both rely on an "implementation language neutral" Interface Definition Language (IDL) to specify how clients interact with servers. Essentially IDL provides data type declarations and method call prototypes for services provided by server objects. In general, IDL must be "compiled" to header and/or source files for a particular target development language. Client and server implementations then include or extend these source files to take advantage of the object technology services. The goal of these services is to make method calls appear to the software developer as though they are local requests, both for the client and the server. CORBA and/or COM take care of the interprocess and network transactions to make this happen.

IDL for CORBA and COM differ in design, but both attempt to provide similar Object Oriented capabilities. "Interworking" specifications describe how to bridge between these two object technologies; however, some fundamental differences make this difficult. Specifically, CORBA provides for multiple inheritance of interfaces, while COM only allows single inheritance [OH98].

Although both CORBA and COM are open standards, there are limited offerings of COM. Microsoft is the only provider of COM on Microsoft Windows operating systems. There are extremely few implementations of COM on non-Microsoft platforms. CORBA, on the other hand, is widely available across many platforms, including Microsoft-based systems and most popular Unix environments. In fact, many implementations of CORBA are written completely in Java, which makes them portable to any platform supporting a Java runtime environment.

The selection of a distributed object integration strategy for this project was one of the most significant decisions to be made. A CORBA-based solution offers the greatest flexibility; also this is the standard utilized by the APCFI. It is beyond the scope of this paper to assess the comparative strengths and weakness of these technologies, but CORBA is generally thought to be based upon more modern distributed software foundations [OH98]. However, integration with software from On-Line Technologies required that a COM interface be available. For this reason, object specifications and implementations for this work were primarily done using COM, but the groundwork is laid for easy conversion to CORBA-based implementations. Actual object implementations were limited to the Microsoft environment, however.

An implementation of the MIT time-based run-to-run controller has been packaged into a convenient, modular form for use with any distributed object system. Core functionality is written in pure Java and assumes no knowledge of a COM or CORBA object system. COM interfaces have been written to support the functionality of these Java classes, and simple “wrapper” classes are used to exchange data between the actual class implementations, through the desired object system. These “wrappers” will be discussed in

more detail in Section 6.3.

6.2 Designing Object Interfaces

Our EWMA controller interface design problem provides a good example for exploring some important object-oriented interface design strategies. An object interface is defined by the object's attributes (internal state variables) and its methods (the actions or functions that it can be called upon to perform), which often use and/or modify the object's attributes. It is important to note here the distinction between **interface specifications** and actual **object implementations**. This chapter concentrates primarily on interface specifications, which are essentially "contracts" for how information is exchanged between server objects and their clients. While these contracts often imply certain underlying structure and functionality for the server objects, the true object implementation is hidden behind the interface that it exposes.

Object interfaces are often built in a hierarchical manner. That is, a given interface definition may rely on one or more other interfaces to be defined first. There are basically two ways to create a new interface that makes use of another object interface, inheritance and containment. Inheritance is a way of "extending" a previously defined object with new attributes and methods. The new object retains all of the state variables and methods of the original, plus the newly defined ones. Inheritance should be used when the new object is a special case of the super-object. On the other hand, containment is a way for one object to use (or contain) another object as an attribute. With distributed object interface technology, containment can be implemented by passing "interface pointers" as attributes.

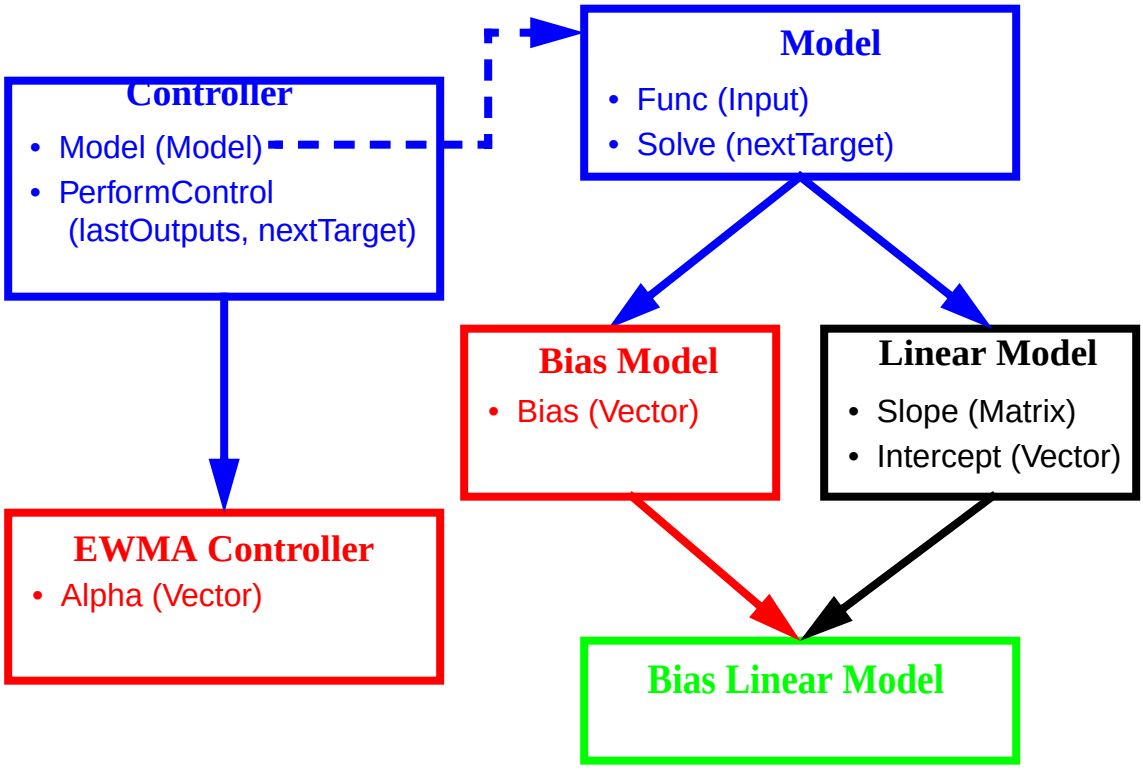
Often, an interface could use either inheritance or containment to accomplish the same goal. However, usually one of them is functionally more appropriate than the other. A

good rule of thumb for deciding when to use inheritance and when to use containment is to answer the following question: Which is more appropriate, “Object A **is a** type of Object B,” or “Object A **has a** component of type Object B?” This distinction will be made more clear as we investigate the object structures described below.

6.3 Interface Strategy for the MIT Time-Based Run-to-Run Controller

Figure 6-1 depicts a simplified version of some interfaces that have been specified and implemented for an EWMA controller using a basic linear model with an offset (bias) vector. (Here, bias and intercept are both equivalent to a vector of offsets.) The figure displays interface inheritance with solid arrows and containment with dashed arrows. Thus an EWMA Controller **inherits** from (extends) a generic Controller, while all Controllers **contain** a Model.

Figure 6-1: Linear EWMA Controller Interface Structure



For a model-based controller, the first important distinction is the separation of the controller from its model. Previously, the linear model and the EWMA controller interfaces and implementations were fused into a single entity. By splitting the controller and its model into distinct objects we greatly enhance the reusability of both objects. One controller can optionally use many different models, or one model could be used by many different controllers.

For generic Controller objects, there are two features in particular that are worth noting here. First, for model-based control, the controller must have an underlying model. This implies the "**has a**" relationship described in the previous section, meaning that interface containment is used here. The second feature is the "PerformControl()" method, which uses the process inputs and sensor outputs from the last run to update the Model, then asks the Model to select process inputs for the next run. (This is described in more detail below.)

Next consider the EWMA Controller, which is a special type of Controller. (Here the "**is a**" relationship implies that interface inheritance should be used.) The EWMA Controller retains all of the attributes and methods of the generic Controller, but it adds a new "Alpha" attribute. This is the vector of EWMA update weights, as described in Section 4.3.2; there is one weight for each of the model's outputs.

For Model objects the key methods are "Func()" and "Solve()." The "Func()" method simply asks the Model to return its outputs, given a set of inputs. A Model's "Solve()" method is provided with a target output vector, and is then expected to return a set of inputs whose outputs will achieve a minimum (weighted) distance from the targets. There are additional solver configurations, such as upper and lower bounds and discretization

levels for the inputs [Moy95] [Smi96]. The full set of attributes and methods that are required for a Model are provided in Appendices A and B. Thus the underlying model is expected to have the multi-input-multi-output functional form

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}),$$

and it should be able to pass values forward (find $\boldsymbol{y}$ given $\boldsymbol{x}$) or backward (find $\boldsymbol{x}$ given $\boldsymbol{y}$) through the system.

Notice that the generic Model does not make any assumptions about the form of $f(\boldsymbol{x})$. A client can make use of the Model interface to interact with a Model object, but it will not know anything about the true nature of the underlying model. This actually implies that the generic Model interface should not be implemented directly (without inheriting from it). Presumably the Model (function) has additional attributes that can be accessed and configured.

As shown in Figure 6-1 there are two types of Models that directly extend a generic Model, namely the Bias Model and the Linear Model. The Bias Model interface was created specifically for the purpose of enabling flexible EWMA-based control. A Bias Model is simply a multi-input-multi-output function that contains a bias (offset) vector, as in

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{b},$$

where the offsets, $\boldsymbol{b}$, can be accessed independently from the rest of the underlying model, $\boldsymbol{f}(\boldsymbol{x})$. Remember that the EWMA Controller's update strategy only requires modification of these offset terms. This type of control can actually use any underlying model, as long as the offset vector is available for model updates. Thus a Model that is contained by an EWMA Controller must support the Bias Model interface. Again, the undefined $\boldsymbol{f}(\boldsymbol{x})$ in this system implies that an actual object should not implement the Bias Model interface

directly.

A Linear Model extends the generic Model with the following underlying function:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b},$$

where $\boldsymbol{A}$ is the slope matrix and $\boldsymbol{b}$ is the vector of offsets. Note that the underlying function for this Model is completely defined, so this interface could be directly implemented as an object.

Finally, we need to provide the EWMA Controller with access to the Linear Model's offset vector. (The Controller does not need to know anything about the slope terms.) This is done by extending both the Bias Model and the Linear model, using multiple inheritance, if available. (Microsoft's COM technology does not offer multiple inheritance; Appendix B describes how this shortcoming can be addressed.) This affect could also have been achieved if the Linear Model had inherited from the Bias Model instead of the generic Model. In either case, the Bias Model interface provides convenient methods for the EWMA Controller to specifically access the bias (offset) vector of the underlying linear model. (Note that an implementation of the Bias Model interface could provide direct access to the Linear Model's offsets, or it could keep a separate set of bias terms that are distinct from the Linear Model's attributes. This decision is completely implementation dependent and both designs are correct.)

Using this strategy, different Bias Models, such as second order Polynomials, Artificial Neural Network (ANN), etc., could be swapped in without changing the EWMA Controller at all. For example, at runtime the Controller could be given a reference to the Bias ANN Model instead of the Bias Linear Model. As long as the Model supports a Bias Model interface, the EWMA Controller can use it.

6.3.1 The Time-Based Model Interface

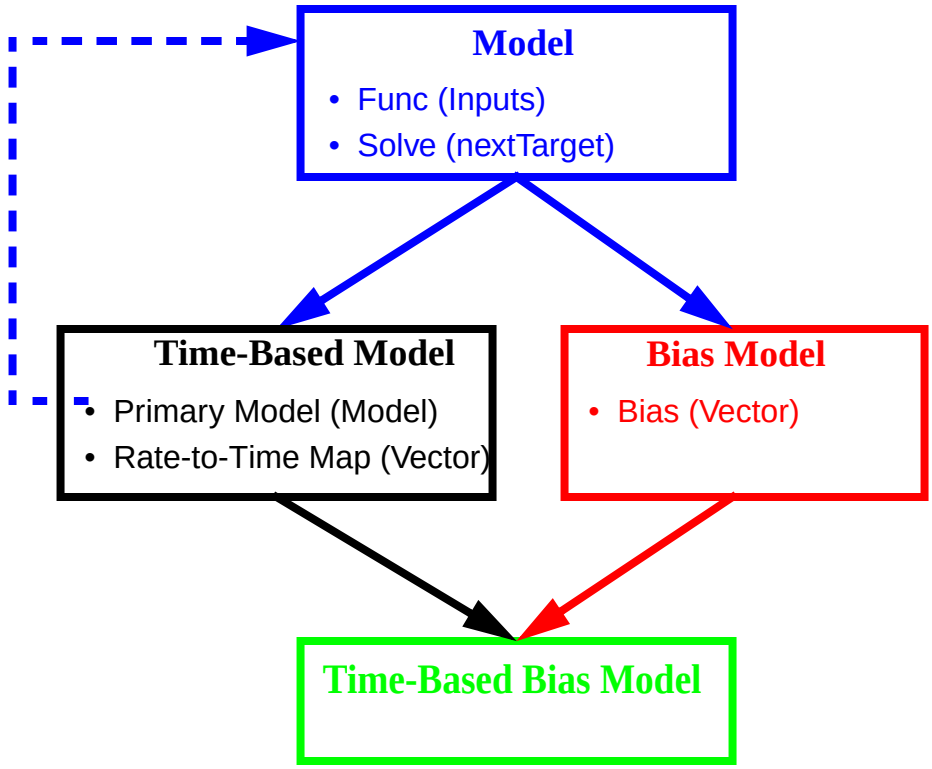
This work extends the generic Model to include a Time-based Model interface. For the system described in this thesis, thickness models use an added structural form to improve their predictive accuracy. As described earlier, deposition processes are often driven by a deposition rate; that is, the equipment settings induce a deposition rate, which can be multiplied by deposition time to estimate the final deposited thickness. Our thickness models include this nonlinear structure by modeling deposition rate as a function of equipment settings, rather than directly modeling the deposited thickness. Thus the complete Model developed for this project could be referred to as a Time-based Bias Linear Model.

A Time-based Model contains unique attributes to handle the use of special time-based inputs and rate-based outputs. These attributes are used to specify which inputs are values of time, which outputs are rates, and a mapping structure to link each output rate to the proper input time. Functionally, the Time-based Model looks like

$$\boldsymbol{y} = \boldsymbol{T}\boldsymbol{f}(\boldsymbol{x}),$$

where $\boldsymbol{T}$ is a diagonal matrix of processing times that is constructed to multiply the rate-based outputs from $\boldsymbol{f}(\boldsymbol{x})$. Figure 6-2 shows the structure for this interface. We can see that a Time-based Model **is a** Model that **has an** underlying Model of process rates.

Figure 6-2: Linear EWMA Controller Interface Structure



For added flexibility, this interface includes the option to "slave" another Model (referred to as the "Primary Model") to the Time-based Model. That is, a Time-based Model does not necessarily need to know what type of underlying Model is generating the rate-based outputs (and possibly non rate-based outputs). It can simply make use of the available method calls to interact with the underlying Model. The iterative back-solver method described in Section 5.1 can be used in this configuration. The Time-based Model implementation for this project makes use of this feature to take advantage of implementation reuse and modularity.

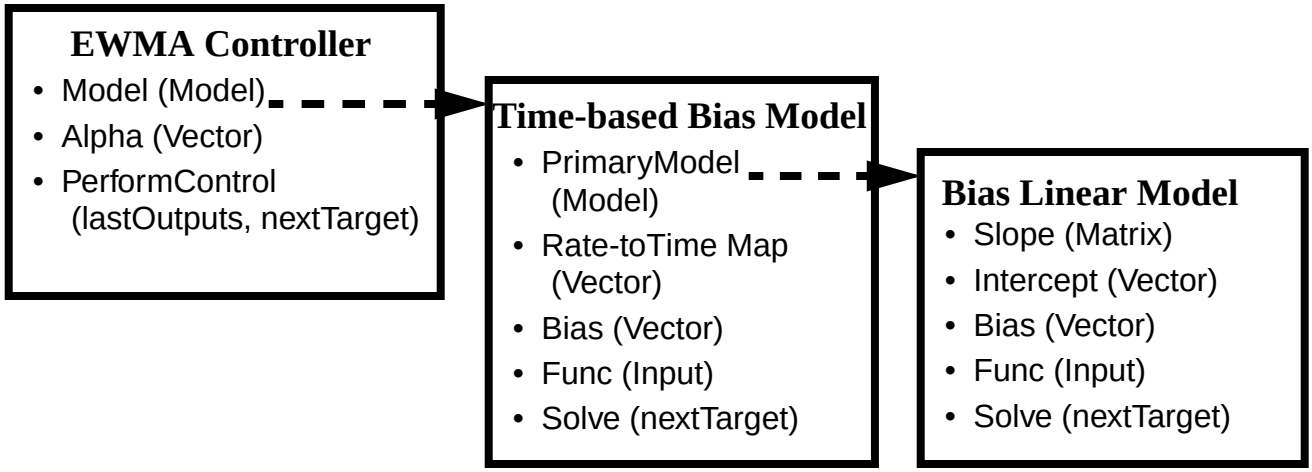
To make the Time-based Model usable by the EWMA Controller, we need to provide access to an underlying offset vector through the Bias Model interface. Our implementation assumes a new model of the form

$$y = T(f(x) + b).$$

To do this, the Time-based Model and the Bias Model are (multiply) extended into a Time-based Bias Model. In this case the multiple inheritance scheme is more appropriate, since the Time-Based Model does not imply or include an underlying offset vector. There are some implementation details required to make this work properly, but these are reserved for the discussion in Appendix B.

Figure 6-3 shows the fully configured EWMA Controller and Time-based Bias Linear Model objects that were developed for this project. The core Bias Linear Model and EWMA Controller implementations are reused completely from the original (non Time-based) EWMA Controller implementation. The Bias Linear Model is slaved to a Time-based Bias Model, which has knowledge of (and can manipulate) everything except the Linear Model attributes.

Figure 6-3: Object Structure: EWMA Controller and Model



6.3.2 Object Implementations

Because there are two competing distributed object technologies under consideration, and because we want to have a flexible system that can easily adapt to future communica-

tion infrastructures, one software design goal was to separate the core controller functionality from the underlying communication mechanisms. Ideally, the Controller objects can be easily and quickly fitted with new interface technologies. As mentioned earlier, this is accomplished through the use of “wrapper” classes that encapsulate and expose the functionality of the various objects.

This design constraint is simplified for objects at the end of the chain, such as the Bias Linear Model object. It does not call on any other objects, so none of its internal methods need to access external objects. The Bias Linear Model is simply extended or included with a “wrapper” object that can accept requests from other objects, call the proper pure java routines from the Bias Linear Model implementation, then return the results. Objects that need to call on other objects, such as the Time-based Bias Model and the EWMA Controller, must be able to access objects that are “contained” within them. Specifically, difficulties arise when the internal, pure java implementation requires access to another object’s data or methods. In the end, this access must make use of the selected distributed object technology.

The Java programming language provides the means for an elegant solution to this problem. First, each object is written as an **abstract** java class that contains all of the required internal variables and data processing methods, using pure java data types. To clarify, an abstract class is one that must be extended through inheritance before it can be instantiated. Generally this is because the object has (empty) abstract methods that must be overridden and given functionality. Abstract methods are included for every type of access that the current object (the client) requires of its “slave” (a “contained” server). Internally, these abstract methods are called, providing an indirect path to the server

object. Finally, these abstract java classes are extended through inheritance, and the abstract methods are filled in with code that utilizes the selected distributed object communication technology.

Thus, the “wrapper” classes for the Time-based Bias Model and the EWMA Controller provide two services. First, they expose methods for clients to control their functionality. Second, they maintain references to the appropriate “contained” objects and exchange data to and from those objects. In this way, only the “wrapper” classes need to be rewritten when switching to a new object communication infrastructure.

6.4 Summary

A flexible, modular interface design for the time-based EWMA run-to-run controller has been developed using distributed object technology. Further, a set of Microsoft COM-based interfaces and object implementations have been developed for integration into the epi deposition testbed system. This work completes the basic background and building blocks necessary for experimenting with automated run-to-run control. Experimental designs, execution strategies, and results are presented in the next chapter.

Chapter 7

Experimental Results

Experimental work for this thesis is composed of two sequential steps. First, as described in Chapter 4, a large set of designed experiments was performed to gather information about the system and build process models. Section 7.1 describes these experiments and analyzes the resulting data. Section 7.2 introduces detailed plans for utilizing DOE data in an industrial control run scenario.

7.1 Design of Experiments: Structure, Execution, and Analysis

This section provides a description and analysis of the designed experiments. Section 7.1.1 details the experimental design and execution, while Sections 7.1.2 through 7.1.4 analyze the resulting data in terms of process stability and noise, process model construction, and process optimization. Section 7.1.5 presents simulated run-to-run control scenarios using the resulting noise characteristics, time-based models for thickness, and linear models for resistivity.

7.1.1 Experimental Design and Execution

The DOE was composed of 128 two-level fractional factorial design points, 18 axial points, 8 center points, and 4 additional replicates of a randomly chosen fractional factorial design point, for a total of 158 runs. There were nine tunable parameters, excluding deposition time, which is held constant across all of the experiments. A full factorial design with these nine parameters at two different levels would require 512 experiments, but a one-quarter fractional factorial design was used to select 128 design points. Table 7-1

specifies the actual input levels that were used in the design. Input levels for the axial runs were set to at the $\pm\alpha$ levels.

Table 7-1: DOE input levels

Factor	Lower axial ($-\alpha$)	Lower (-1)	Center (0)	Upper (+1)	Upper axial ($+\alpha$)
Deposit time (sec)	N/A	N/A	50	N/A	N/A
Deposit temperature (^{o}C)	1090 (-1.500)	1100	1120	1140	1150 (1.500)
Dopant mixing ratio (%)	20 (-1.500)	30	50	70	80 (1.500)
Dopant main flow (sccm)	40 (-1.571)	80	150	220	260 (1.571)
% Lower power	45 (-1.667)	47	50	53	55 (1.667)
% Inner power	40 (-1.429)	43	50	57	60 (1.429)
Dilutant (H_2) flow (slm)	30 (-1.500)	35	45	55	60 (1.500)
Trichlorosilane (TCS) flow (slm)	10 (-1.667)	10.8	12	13.2	14 (1.667)
Center gas flow valve (Accusett inner setpoint)	115 (-1.667)	123	135	147	155 (1.667)
Outer gas flow valve (Accusett outer setpoint)	60 (-1.667)	68	80	92	100 (1.667)

C programs were written to prepare the set of DOE recipes for execution. Techniques based on those described in [Mon91] were used to break the design into a one-quarter fraction. However, limitations in the ability to manipulate recipe settings required the use of our own DOE software, rather than simply utilizing an off-the-shelf package. While most parameters are conveniently available through the system's front panel, binary recipe files, and through SECS-based communications with the equipment, there are two settings, center gas flow and outer gas flow, that had to be manipulated manually at the equip-

ment before processing a wafer.

The preferred method for running a DOE is to randomize the order in which the design points are executed; either the whole design is randomized, or blocks of design points are randomized together. Unfortunately, these randomizations would require someone to sit next to the equipment during all 158 DOE runs and manually modify the center and outer gas flow valves (Accusetts) before each run. This scenario simply leaves too much room for human error, and requires too much of the process engineer. The goal is to create a set of DOE runs, configure the software and the processing equipment, and let most of the experiments run automatically without intervention. To enable this, a program was written to extract a blocked, randomized fractional factorial design such that each block contains groupings of designs with the same (two-level) Accusett values. With two manual settings there are four different combinations that occur in the fractional factorial design. Thus the software generates four different internally randomized blocks, plus separate blocks of axial and center points. This DOE structure provides a reasonable trade-off between a truly randomized design and a convenient method for running the experiments.

After selecting the ordered set of DOE runs, a system for efficiently and accurately performing the experiments was implemented. Ideally a DOE module within a Cell Controller could generate the experimental design, then take over the processing equipment and step through the recipes one at a time. Eventually a complete Cell Controller implementation should provide this capability, but a simple and effective method was used to ensure smooth, error-free processing of the design. Binary recipe files were created and named sequentially for each of the DOE runs. Before processing a cassette, the proper recipe was assigned to each wafer slot. The equipment was then configured to automatically

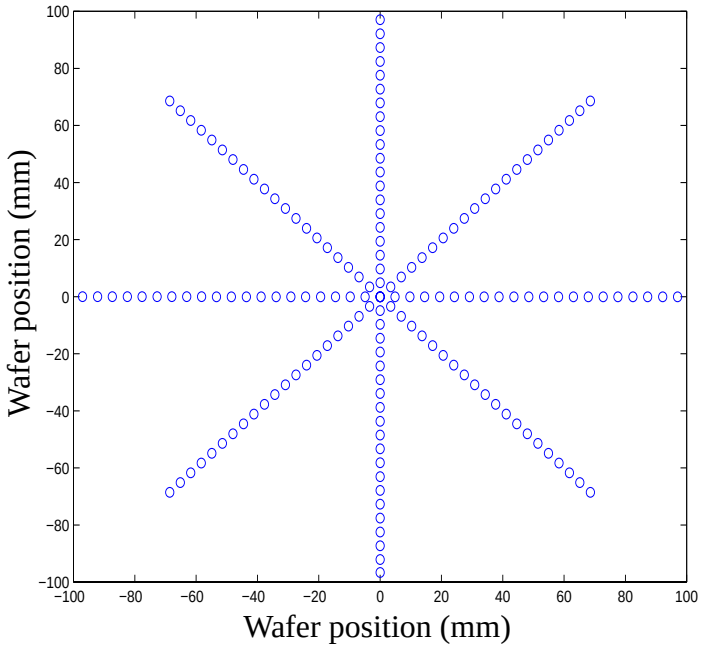
process the wafers in sequence, which took care of setting all recipe parameters except for the two Accusetts.

The runs were broken up into four lots, three with 40 wafers and one with 38. All recipes that required Accusett modifications between runs were loaded into the front end of each lot, so the process engineer only needed to physically be at the equipment for the first five to seven wafers. After that, a single combination of Accusett values applied for the rest of the lot.

The wafers in each lot were run sequentially on a single (epi) chamber tool at Applied Materials. Before the start of each lot, a monitor wafer was run at the same settings that were used for the last wafer in the lot. Unfortunately, wafers were broken during the processing of lots 1 and 4, which left wafer shards in the chamber. The chamber was opened and cleaned, then “redo” lots were run to replace the data for broken wafers and all wafers that were processed after a broken wafer.

After depositing epi on the wafers, each were measured by off-line thickness and resistivity sensors. 164 thickness measurements were taken using four diameter scans with 41 sites each, as shown in Figure 7-1. All measurements at each radius (four measurements for the center point and eight measurements for all other radii) were averaged to provide 21 radial site thicknesses. These measurements are highly repeatable and accurate.

Figure 7-1: DOE thickness measurement map

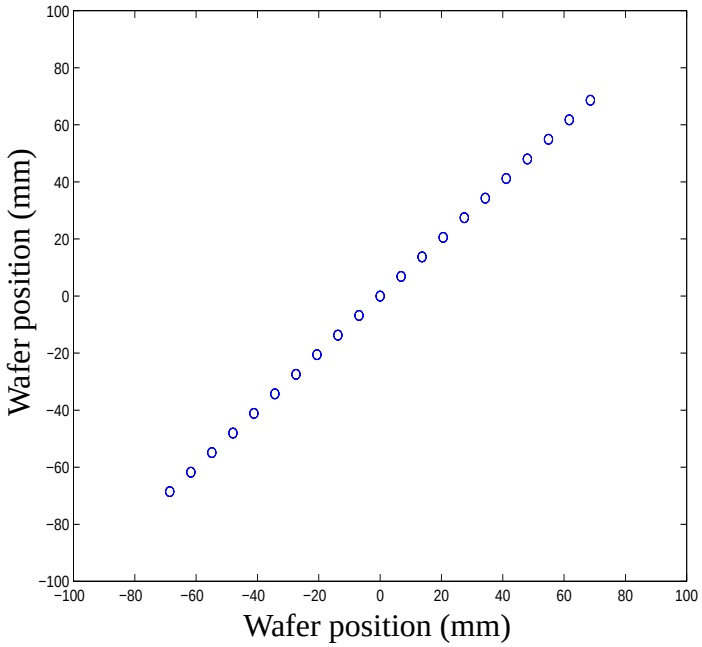


Due to repeatability and throughput issues for the mercury probe, four repetitions of a single diameter scan (with 21 sites) were taken for resistivity measurements, as shown in Figure 7-2. All four scans were reviewed and the “best” one was selected for each wafer. (Process engineers at Applied Materials were the primary source for determining which results were the best. Selection criteria included looking for measurement scans that provided reasonable values with minimal site-to-site variation and good radial symmetry.) This scan was similarly averaged, where there was a single center point and two measurements for all other radii, resulting in 11 radial site resistivities. Only 10 sites were retained for analysis since the outermost resistivity measurements demonstrated high variation that was not well modeled.

Unfortunately, the sensor was only able to provide reasonably accurate data for 113 out of the 158 experiments. Two separate rounds of measurements were performed by engineers at Applied Materials, but the lack of human resources and sensor availability

precluded attempts to re-measure more of the wafers. While not optimal, it is hoped that the available data provide enough information for good models of resistivity.

Figure 7-2: Resistivity measurement map



7.1.2 Process Stability and Noise

Process stability and noise are addressed by analyzing experimental replicates and monitor wafers. Pairs of center points were run at the beginning of each of the four DOE lots, yielding a total of eight center point replicates. There were also five replicates of a randomly selected fractional factorial design point, which were also run at the start of each lot. Monitor wafers provide information about process stability and drift during the course of each lot.

Examination of the measurements reveals that deposited thickness appears to be quite stable over time and after processing at a variety of equipment settings. Also, the data demonstrate considerable variation in thickness profiles across the different DOE process settings, indicating that effective process optimization and control are possible. The resis-

tivity data show much noisier results, and the lack of a full data set limits the ability to compensate for any process shifts or drifts. The next two subsections consider process noise and stability for thickness and resistivity.

Thickness

Center point replicate pairs provide both a means for estimating process shifts between separate lots, and a way to estimate the type(s) of noise in the system. Figure 7-3 shows the radial thickness measurements from these eight runs. The data do appear to be nicely clustered around the grand mean of all center point measurements. However, consider taking each pair of center points and shifting them such that the mean of those two radial scans lines up with the grand mean from all of the scans. Figure 7-4 shows the resulting data.

Figure 7-3: Thickness: Original Center point replicates

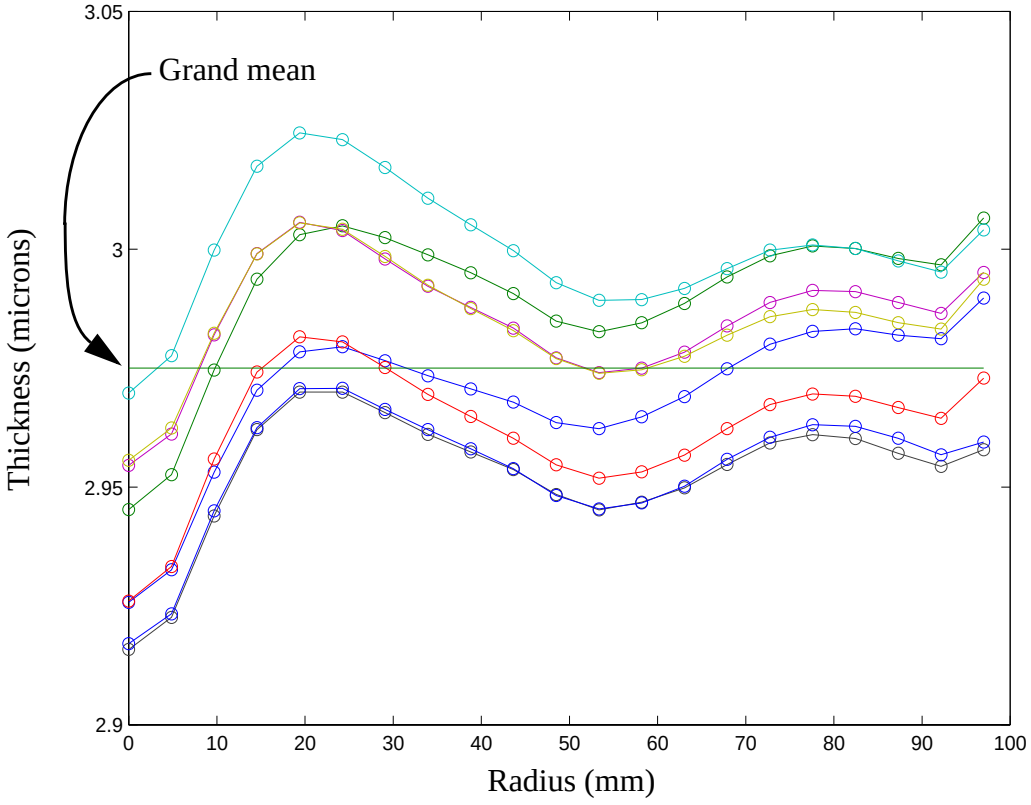
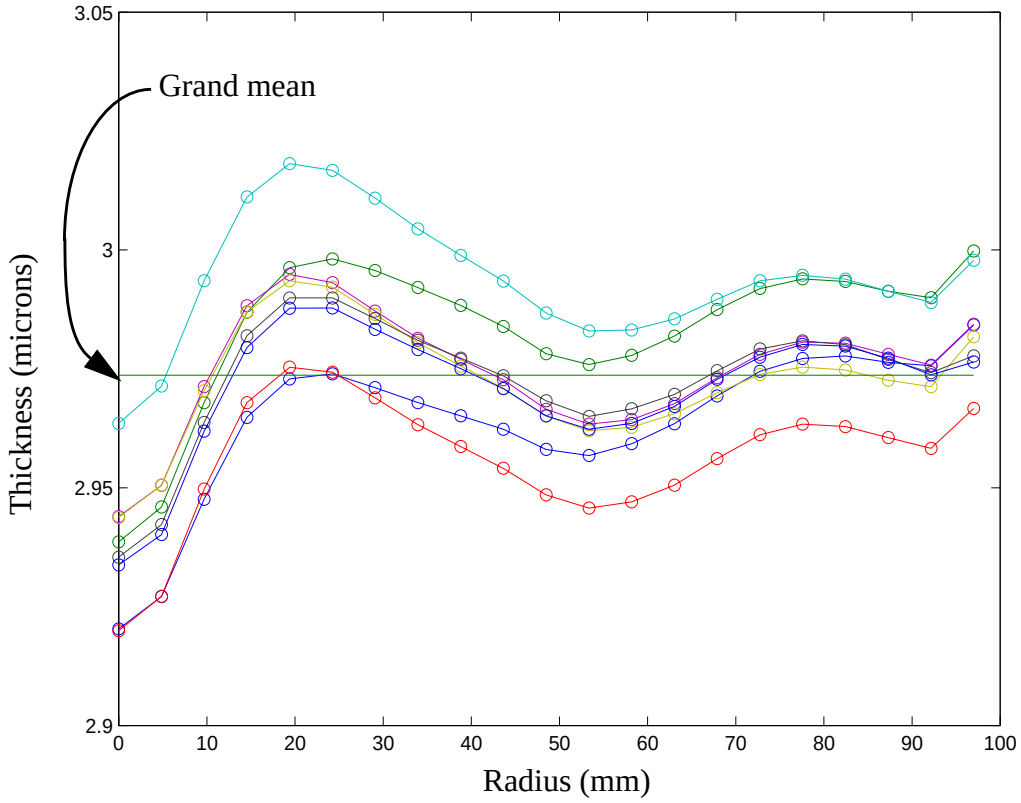


Figure 7-4: Thickness: Adjusted Center point replicates



The adjusted center point replicates cluster themselves considerably more tightly around the grand mean. All of the data in each lot can be adjusted using the same scale factors to account for process shifts between the lots. While there is no guarantee that these manipulations actually produce a "better" data set, we will see later that models built using the adjusted data provide a better fit. This strongly suggests that the adjustments are appropriate.

Monitor wafers provide information about process drift during the course of each lot. Unfortunately, the monitor wafer for lot 2 was incorrectly processed, so no information on process drift is available for that lot. However, monitors and their corresponding DOE runs are available for all other lots, including the two redo lots. For example, Figure 7-5 shows

the thickness profiles of the monitor and actual DOE wafers for lots 3 and 4. All monitor wafers indicate a slight trend toward increasing thickness (deposition rate) over the course of processing a lot. Table 7-2 presents the drift measurements based on monitor wafers. The table shows that these drifts are quite small, but they can be accounted for.

Figure 7-5: Thickness: Lots 3 and 4: monitor and final runs

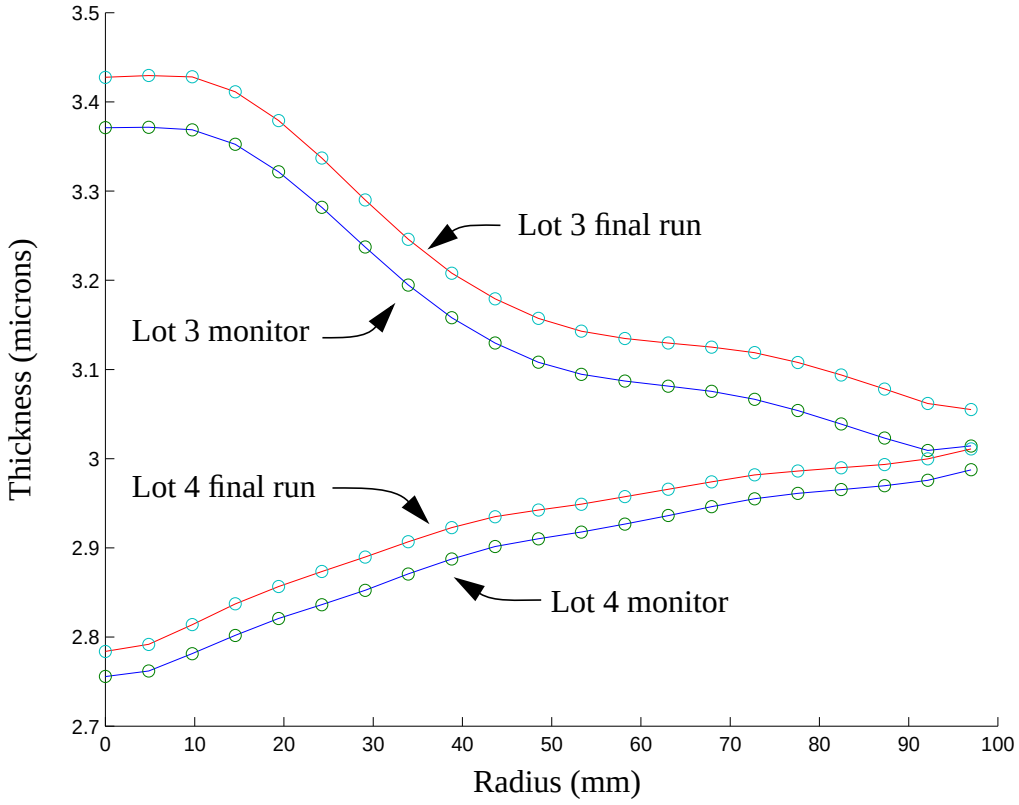


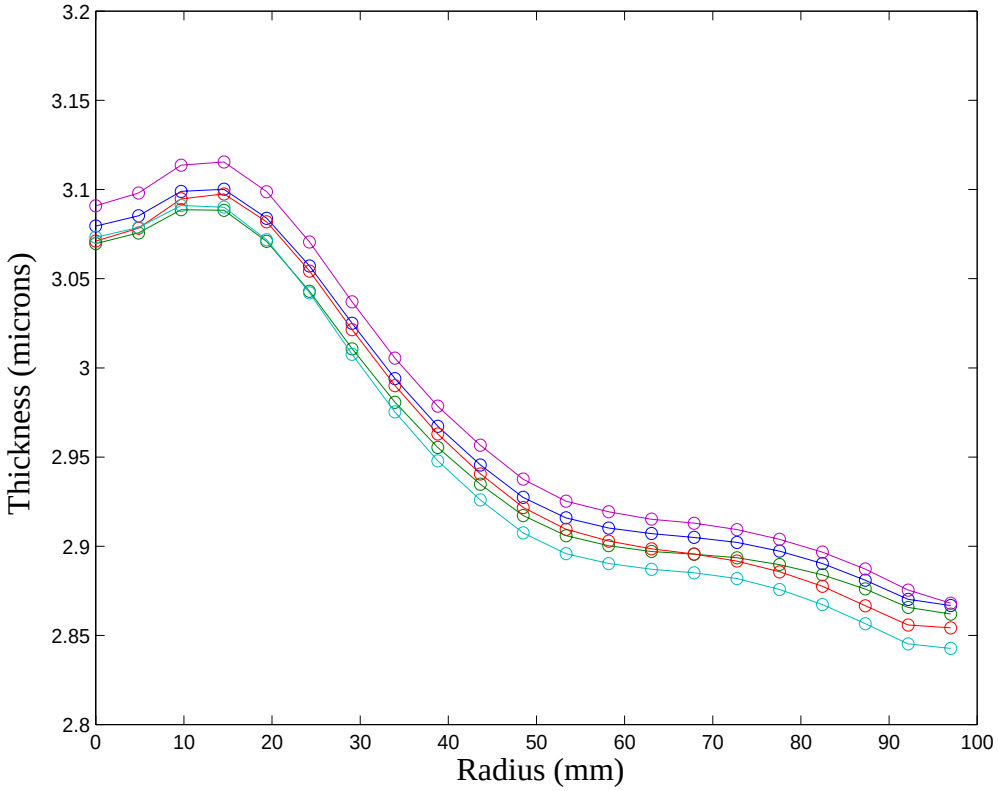
Table 7-2: Thickness: Monitor wafer drift measurements

Lot	Thickness drift (microns/run)	Deposition rate drift (microns/sec/run)
Lot 1	0.0011	0.0224×10^{-3}
Lot 1 Redo	0.0055	0.1091×10^{-3}
Lot 2	---	---
Lot 3	0.0013	0.0263×10^{-3}
Lot 4	0.0028	0.0566×10^{-3}
Lot 4 Redo	0.0019	0.0381×10^{-3}
Average	0.0025	0.0505×10^{-3}

A linear extrapolation of scale factors can be used to account for this process drift throughout each lot. Again, the argument for making these types of adjustments will be backed by superior modeling results when using the adjusted data. Also note the range of uniformity characteristics shown by the two DOE runs in Figure 7-5. Profiles vary from center thick to center thin, indicating that there is great opportunity for process modeling, optimization, and control. Also, it is worth noting that drifts in deposited thickness appear to primarily occur uniformly across all of the radial sites. The use of our “joint sensor” monitoring strategy described in section 4.3.3 is ideal for this scenario.

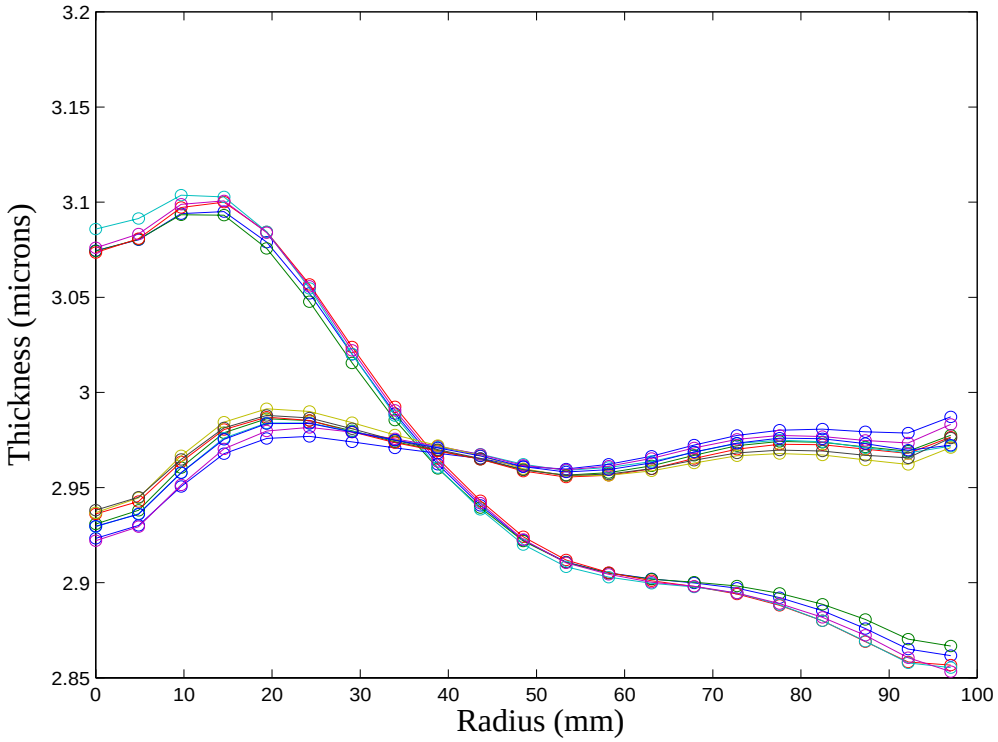
To further compare repeatability and variation properties of the system, five replicates of a single randomly selected factorial design point were run. The results may be compared with those from the center point replicates. Figure 7-6 displays the radial thickness plots for the five replicates of this experimental design point, after shift and drift adjustments.

Figure 7-6: Thickness: Factorial design point replicates (adjusted)



Most of the noise found in the replicate data of Figures 7-4 and 7-6 appears to come from a general mean shift across all of the radial sites. Figure 7-7 shows the center and factorial replicates together with each radial scan shifted such that the means of each replicate lie on their respective grand means. Clearly the individual site noise is much smaller than the mean shift noise. Estimates of the mean shift noise are extracted from the data seen in Figures 7-4 and 7-6, while individual site noise can be estimated using the data displayed in Figure 7-7.

Figure 7-7: Thickness: Mean centered replicates



An estimate of mean shift noise is determined from the center point replicates through a range-based method. Remember that the data are already mean shifted such that the grand average of each replicate pair lines up with the grand mean across all center point replicates. This implies that estimates of noise (standard deviation) should be based on the pooled variance from each replicate pair. For small sample sizes, such as groups of two, the estimate of standard deviation should use sampled ranges, as in

$$\mu_i = \frac{1}{21}\sum_{j=1}^{21} t_{ij},\ R_i = \left|\mu_i - \mu_{i-1}\right|_{i=(2,4,6,8)},\ \hat{\sigma}_\mu = \frac{\frac{1}{4}\sum_{i=1}^{4} R_{2i}}{d_2}. \qquad \textbf{(Eq 7-1)}$$

The *j* indices refer to the radial site location, while *i* indexes the eight replicates. There are four sample ranges for mean shift noise, one for the difference in mean within each replicate pair.

After removing the mean shifts between all center point replicates, as shown in Figure 7-7, there are 21 groupings of eight data points, by which the site noise may be estimated. A pooled variance strategy can be used to combine those 21 estimates of variance, as in

$$S_j^2 = \frac{1}{8-1}\sum_{i=1}^{8}(ct_{ij} - \overline{ct}_j)^2 \text{ and } \hat{\sigma}_{site}^2 = \frac{1}{21}\sum_{j=1}^{21} S_j^2 . \qquad \textbf{(Eq 7-2)}$$

The *ct* terms refer to thickness measurements after the scans are re-centered. There might appear to be some added structure to the site noise seen in Figure 7-7, but for the sake of simplicity, and due to the relatively small number of replicates, the site noise is estimated as a constant across all of the sites.

Estimates of mean shift noise for the factorial replicates are simply found using the "normal" sample variance, which is

$$\hat{\sigma}_{\mu}^2 = \frac{1}{5-1}\sum_{i=1}^{5}(\mu_i - \bar{\mu})^2 . \qquad \textbf{(Eq 7-3)}$$

This method may be used because, unlike the center point replicates, these radial scans have not been shifted in relation to each other. After mean centering the scans, an estimate of site noise uses the same form as that shown in Equation 7-2.

Table 7-3 shows the estimates for mean shift noise and site noise for the two different replicate types. Both types of noise are exceedingly small when compared to the grand mean of the respective replicates. As a ratio of the mean, the standard deviation is less than 0.5 percent in all cases. The mean shift noise appears to be roughly three times larger than the individual site noise. Also, the system exhibits roughly the same type and amount of noise at two different process settings, which lends support to some of the assumptions made in Section 4.2.2 on the analysis of multiple response surface techniques.

Table 7-3: Thickness: Noise estimates from the DOE replicates

Replicate Type	Type of Noise	Std. Dev. of Noise (microns)
Center	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0136 (0.46%)
Center	Site noise ($\hat{\sigma}_{site}$)	0.0041 (0.14%)
Fractional Factorial	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0104 (0.35%)
Fractional Factorial	Site noise ($\hat{\sigma}_{site}$)	0.0035 (0.12%)

Some of the added structure in the site noise is likely explained by the same feature that is used to justify the joint sensor monitoring strategy described in Section 4.3.3. The correlation structure of the DOE (measured-modeled) thickness errors is shown in Figure 7-8. Each individual line in the plot is the set of correlation coefficients for one of the thickness measurement sites with respect to all other thickness sites. For the linear error extrapolation strategy proposed earlier, the ideal output error correlation structure would look like the set of parallel lines that appear in Figure 7-9. While the measured correlation structure is not exactly ideal, its striking similarity provides hope that linear error extrapolation will work well. This structure is also likely to affect the correlations in site noise, which could further explain any “non-randomness” seen in Figure 7-7. However, as stated earlier, this possibility will be ignored for simulation and implementation purposes.

Figure 7-8: Thickness: Output error correlation structure

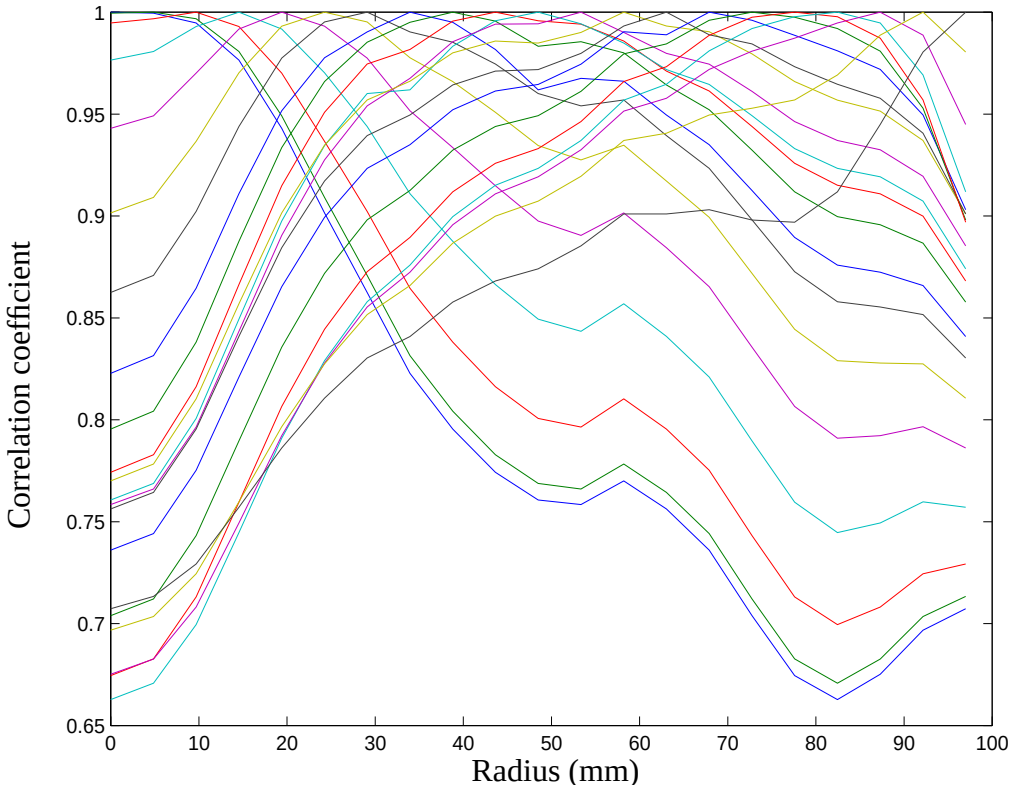
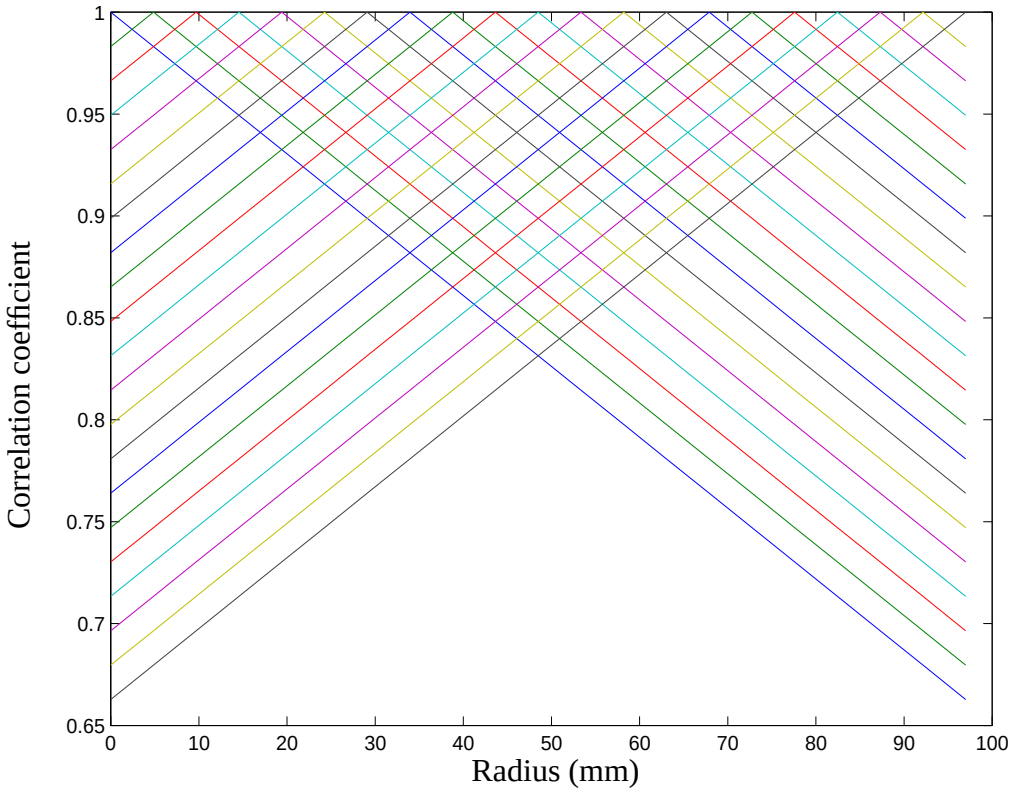


Figure 7-9: Thickness: Ideal output correlation structure



Resistivity

The lack of a complete data set for resistivity limits the ability to compensate for process drifts and shifts during the DOE. Only six of the eight center point replicates and four of the five factorial replicates are available. Also, there is only a single lot whose monitor and final wafer measurements are valid for estimating drift. For these reasons the resistivity data are used "as is," with no mean centering or drift adjustments.

Figure 7-10 shows the data for the center point and fractional factorial design point replicates. Considering the scale for these plots, it is clear that there is much more variation in these outputs, both in terms of mean shift noise and individual site noise. Since these data are not adjusted, sample estimates of variance use the forms found in Equations 7-3 and 7-2 for both replicate types. These estimates are shown in Table 7-4.

Figure 7-10: Resistivity: Center point and factorial design point replicates

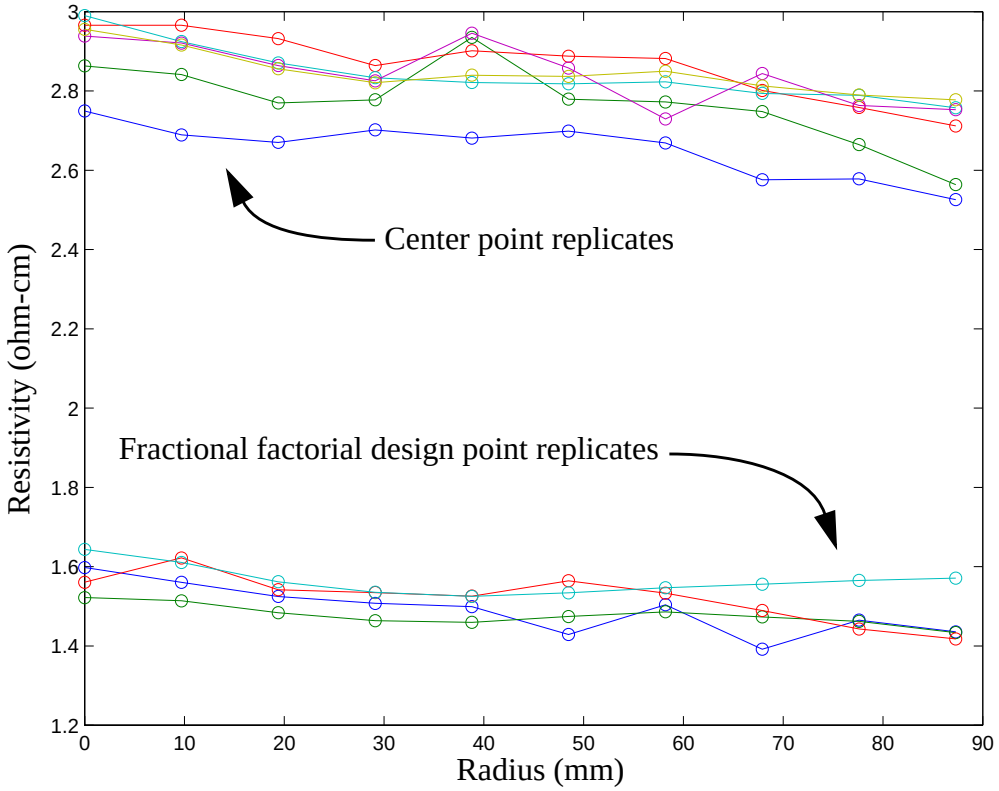


Table 7-4: Resistivity: Noise estimates from the DOE replicates

Replicate Type	Type of Noise	Std. Dev. of Noise (ohm-cm)
Center	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0805 (2.87%)
Center	Site noise ($\hat{\sigma}_{site}$)	0.0376 (1.34%)
Fractional Factorial	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0389 (2.57%)
Fractional Factorial	Site noise ($\hat{\sigma}_{site}$)	0.0322 (2.13%)

The sample standard deviation of the noise appears to be considerably higher for resistivity than that found in the thickness measurements. It is unclear how much of the noise is due to the system (actual process noise) and how much is due to the sensor (measurement noise). The difficulty in getting the sensor to provide repeatable measurements indicates that a substantial part of the noise is probably introduced by the sensor itself.

7.1.3 Process Model Construction

Models for each of the 21 radial deposition rate sites and the 10 radial resistivity sites are generated via a statistically driven polynomial response surface modeling methodology. (Remember that the 21 radial thickness sites use the time-based model, where deposition rate times deposition time yields final thickness.) A backward elimination, weighted least square residuals procedure was developed in Matlab, using [DS81] and the JMP software package from SAS [SAS95]. Model construction begins by fitting a polynomial function to the input-output data and generating Analysis of Variance (ANOVA) parameters. The polynomial terms accounting for the least amount of output variation are iteratively removed until all remaining terms contribute to the output with probability greater than or equal to a given cutoff value. First and second order models are considered for the DOE's deposition rate and resistivity data.

Table 7-5 provides a sample (final) ANOVA for a linear model of the center point thickness output. The “Enter” column indicates with a yes (Y) or no (N) whether or not the specified parameter remains in the model. The fitted “Estimate” is a constant (offset) for the intercept parameter, or the slope coefficient for a first order parameter. The number of “Degrees of Freedom” for each parameter in a linear model is one. The “Partial Sum of Squares” is the amount of variation in the model that is accounted for by that particular parameter, while the “F Ratio” is the ratio of the “Partial Sum of Squares” to the sum of squared error that is not accounted for by the model, each scaled by their respective number of degrees of freedom. Finally, the “Prob>F” column indicates the probability that the given parameter is **not** part of the model. For this table, any parameter with a “Prob>F” exceeding 0.1 was removed from the model.

Table 7-5: Thickness center point: Linear model ANOVA (Prob. to leave = 0.1)

SSE (Sum of Squared Error)	DFE (Degrees of Freedom in Error)	MSE (Mean Square Error)	RSquare	RSquare Adj.
0.6535×10^{-4}	151	0.4328×10^{-6}	0.9731	0.9721

Enter ?	Parameter	Estimate (slope)	Num Deg Fr	SS (Partial Sum of Squares)	“F Ratio”	“Prob>F”
Y	(intercept)	0.0583	1	-	-	-
Y	Deposit temperature	1.9341×10^{-3}	1	0.5084×10^{-3}	$1.175 \times 10^{+3}$	0
N	Dopant mixing ratio	-	1	0.0001×10^{-6}	0.0002	0.9876
N	Dopant main flow	-	1	0.2288×10^{-6}	0.5269	0.4690

Enter ?	Parameter	Estimate (slope)	Num Deg Fr	SS (Partial Sum of Squares)	“F Ratio”	“Prob>F”
N	% Lower power	-	1	0.2267×10^{-6}	0.5221	0.4711
Y	% Inner power	0.1788×10^{-3}	1	4.332×10^{-6}	10.01	0.0019
Y	Dilutant (H_2) flow	0.6176×10^{-3}	1	0.0518×10^{-3}	$0.1198 \times 10^{+3}$	0
Y	Trichlorosilane (TCS) flow	3.2041×10^{-3}	1	1.406×10^{-3}	$3.2487 \times 10^{+3}$	0
Y	Center gas flow valve	0.7994×10^{-3}	1	0.0875×10^{-3}	$0.2022 \times 10^{+3}$	0
Y	Outer gas flow valve	-1.5719×10^{-3}	1	0.3384×10^{-3}	$0.7819 \times 10^{+3}$	0

Deposition Rate

Each of the 21 deposition rate (site) models is a direct function of all nine (non-time) inputs. Based on the ANOVA information from the resulting linear deposition rate models, the inputs that most strongly affect the outputs are Center gas flow valve, Outer gas flow valve, Deposition Temperature, H_2 Flow, and TCS Flow. Figures 7-11 through 7-15 plot the modeled deposition rate for each radial site as a first order function of the aforementioned inputs. For each simulation, the selected input is varied throughout its DOE range, while the remaining eight process settings are held at their DOE center point values.

Figure 7-11: Deposition Rate: 1st order functions in terms of Center Gas Flow Valve

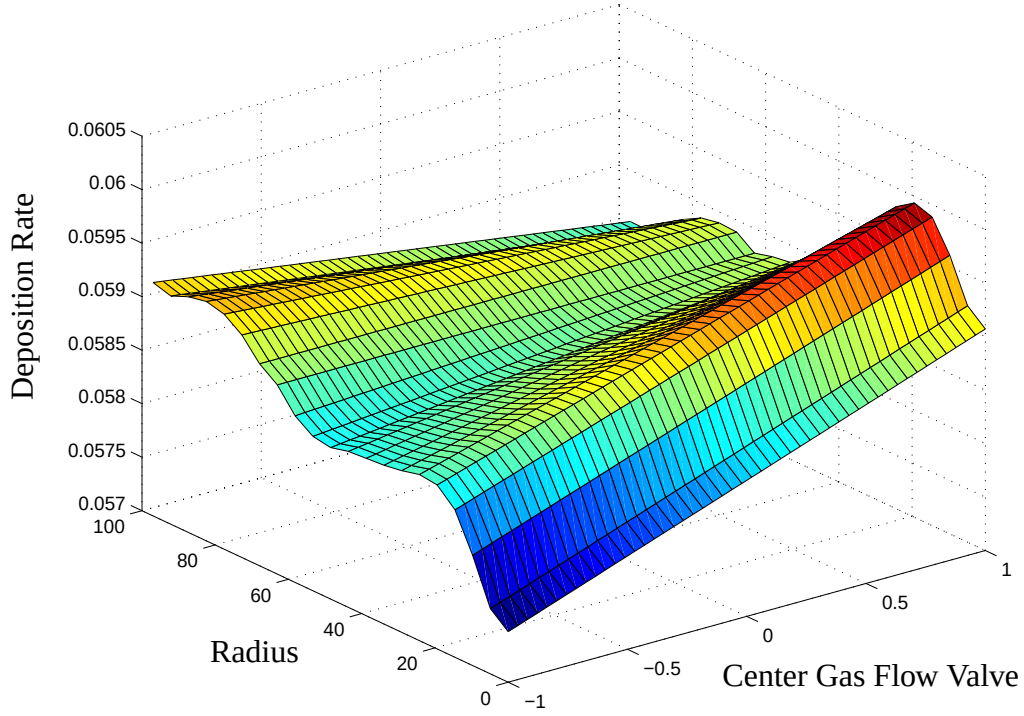


Figure 7-12: Deposition Rate: 1st order functions in terms of Outer Gas Flow Valve

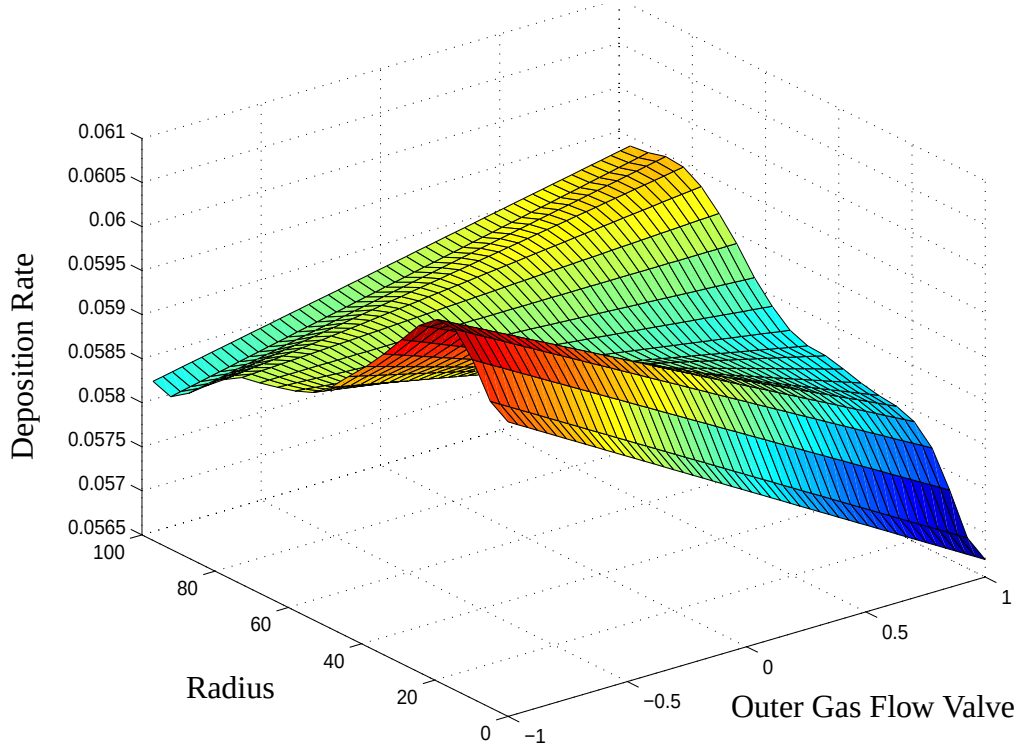


Figure 7-13: Deposition Rate: 1st order functions in terms of Temperature

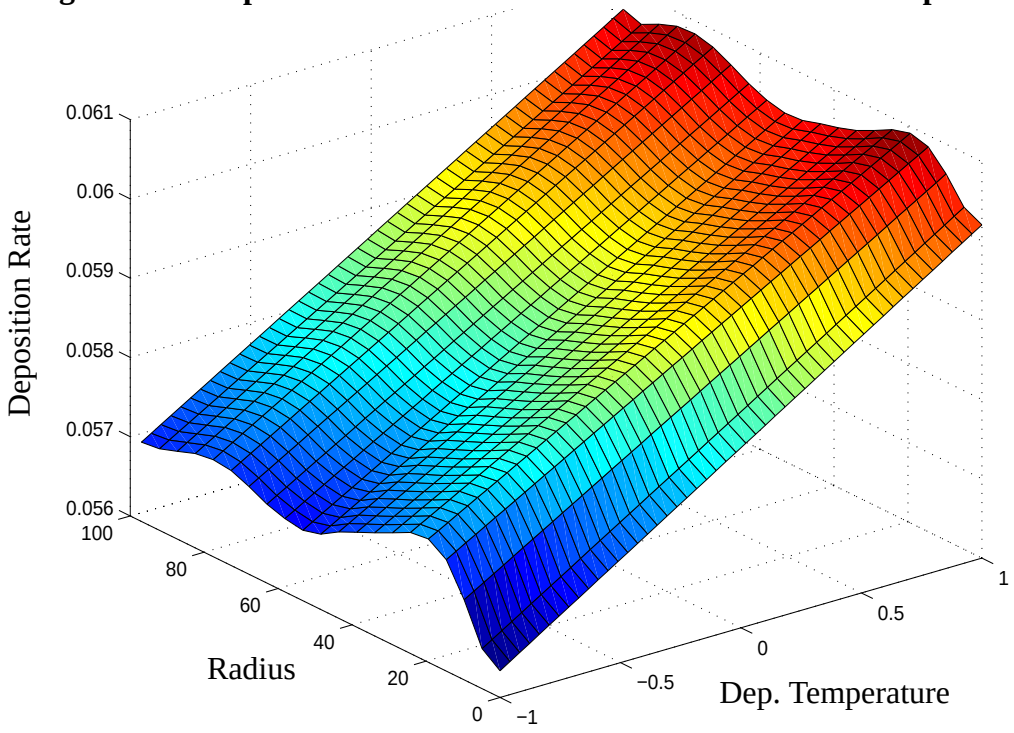


Figure 7-14: Deposition Rate: 1st order functions in terms of H_2 Flow

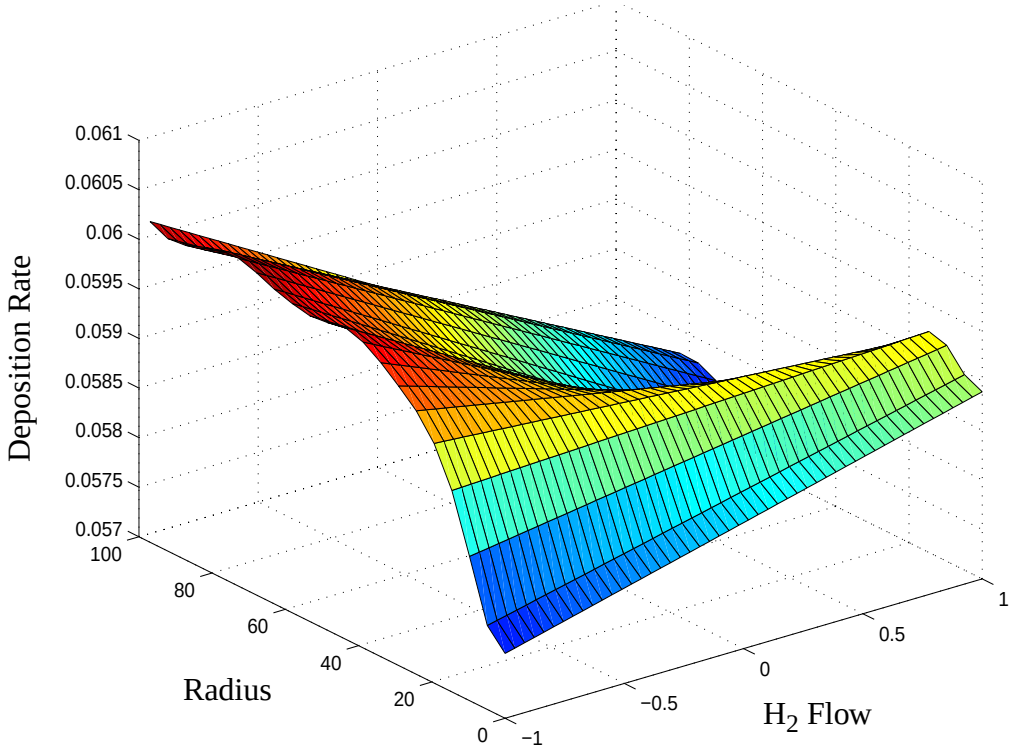
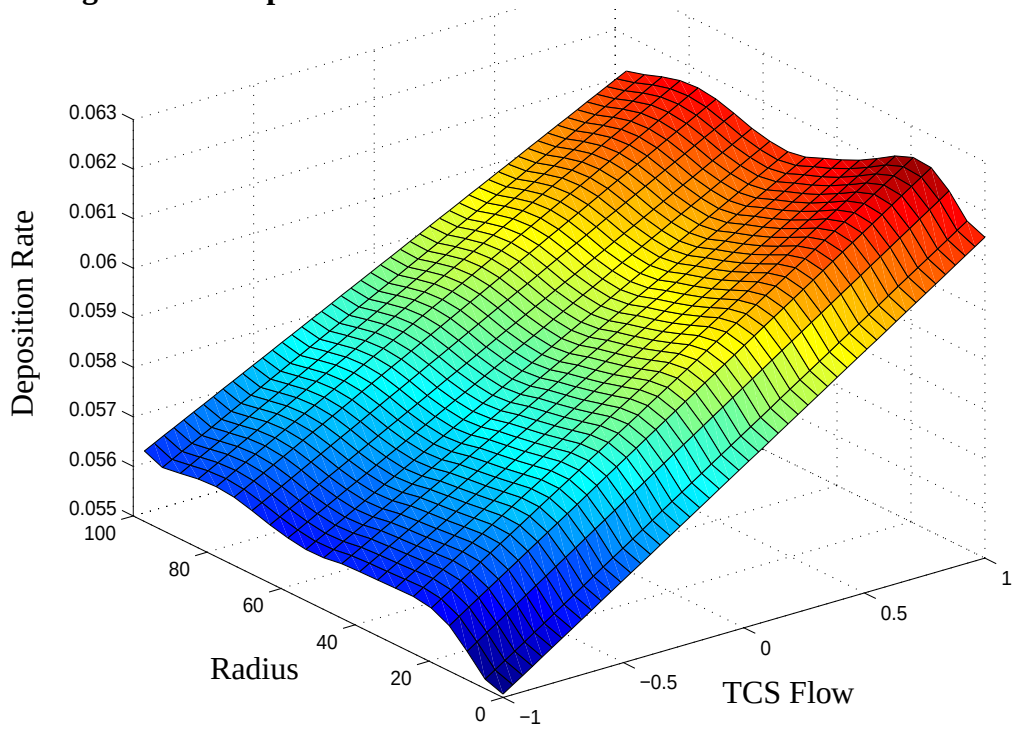


Figure 7-15: Deposition Rate: 1st order functions in terms of TCS Flow



Notice that the deposition rate as a function of radius is highly nonlinear for many of the plots. However, we must remember that the site models absorb this nonlinearity. The outputs do not directly model the rate as a function of radius; each radial site is a separate model. Thus, in these five figures, the models only contain terms from the process settings and not the radius.

It is interesting to see the various effects these inputs have on the deposition rate uniformity. The inner and outer gas flow valves operate as one would expect; increasing gas flow to the center of the wafer increases the deposition rate at the center of wafer and decreases the deposition rate near the edge of the wafer. Increasing the gas flow to the edge of the wafer has the opposite effect. Increasing the deposition temperature and source gas flow increase the deposition rates roughly uniformly across the wafer, which makes

sense for a reaction rate limited system. The intuition on effects from the hydrogen gas flow is not so clear. It seems to exhibit characteristics similar to those from the center gas flow valve. The inner and outer gas flow valves and the hydrogen gas flow appear to have the most influence over uniformity.

The quality of these deposition rate models can be considered using a number of different criteria. First, an (ANOVA-based) R-squared analysis provides estimates for the percentage of output variation that is accounted for by the models. Table 7-6 provides the R-squared information for the 21 deposition rate sites using both the adjusted and unadjusted DOE data. For the unadjusted DOE data, the first order (linear) models capture an average of 96.5% of the variation, while the second order polynomials capture approximately 98.7%. For the adjusted data, the linear models contain about 97.2% of the variation, and the second order models contain about 99.1%. Note that all models built with the adjusted data demonstrate better fit than those built with the unadjusted data. This is strong evidence that the adjustments described earlier for process shifts and drifts are providing a "cleaner" data set. These results also show that the second order models represent the DOE data extremely accurately. However, it is more important to note that the linear models also provide excellent modeling capability, suggesting that the proposed time-based controller with linear rate models will work well.

Table 7-6: Deposition rate: R-squared results for model fit

Radial site (microns)	Data Unadjust 1st order	Data Unadjust 2nd order	Data Adjust 1st order	Data Adjust 2nd order
0.00	0.9667	0.9893	0.9731	0.9918
4.85	0.9670	0.9894	0.9734	0.9916
9.70	0.9678	0.9887	0.9750	0.9918

Radial site (microns)	Data Unadjust 1st order	Data Unadjust 2nd order	Data Adjust 1st order	Data Adjust 2nd order
14.55	0.9694	0.9884	0.9763	0.9912
19.40	0.9696	0.9873	0.9768	0.9908
24.25	0.9688	0.9871	0.9765	0.9903
29.10	0.9684	0.9865	0.9758	0.9900
33.95	0.9676	0.9871	0.9748	0.9902
38.80	0.9678	0.9875	0.9743	0.9901
43.65	0.9677	0.9872	0.9737	0.9903
48.50	0.9670	0.9868	0.9728	0.9906
53.35	0.9657	0.9866	0.9726	0.9906
58.20	0.9649	0.9867	0.9729	0.9902
63.05	0.9634	0.9865	0.9721	0.9901
67.90	0.9617	0.9865	0.9716	0.9901
72.75	0.9602	0.9864	0.9705	0.9897
77.60	0.9597	0.9864	0.9693	0.9900
82.45	0.9595	0.9864	0.9685	0.9901
87.30	0.9613	0.9866	0.9683	0.9906
92.15	0.9628	0.9866	0.9677	0.9904
97.00	0.9591	0.9852	0.9626	0.9903
Average	0.9650	0.9871	0.9723	0.9905

These models also provide a means for estimating the amount of noise in the system. The model error for each DOE run can be broken down into mean shift noise and site noise, as was done with the noise estimates from the DOE replicates. Using first and second order models from the adjusted DOE data, Table 7-7 presents the noise due to model error. (The table data for the replicate-based noise estimates are also provided for convenience.)

Table 7-7: Thickness: Noise estimates from the process models

Estimate based on:	Type of Noise	Std. Dev. of Noise (microns)
1st Order process models	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0267 (0.90%)
1st Order process models	Site noise ($\hat{\sigma}_{site}$)	0.0096 (0.32%)
2nd Order process models	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0159 (0.53%)
2nd Order process models	Site noise ($\hat{\sigma}_{site}$)	0.0049 (0.16%)
Center Replicates	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0136 (0.46%)
Center Replicates	Site noise ($\hat{\sigma}_{site}$)	0.0041 (0.14%)
Fractional Factorial Repl.	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0104 (0.35%)
Fractional Factorial Repl.	Site noise ($\hat{\sigma}_{site}$)	0.0035 (0.12%)

The second order models demonstrate noise characteristics that are very close to those found in the replicate data. These models demonstrate a mean shift noise with a standard deviation of 0.0159 microns, while the replicate data indicate mean shift noise with a standard deviation of 0.0136 and 0.0104 microns. Similarly, an estimate of the standard deviation of the site noise from the second order models is 0.0049 microns, which compares favorably with replicate-based estimates of 0.0041 and 0.0035 microns. For this reason, the second order models provide an ideal level of model complexity. That is, higher order models could better fit the data, but they would probably be fitting to the noise in the system, and therefore providing worse process models.

The first order models demonstrate approximately twice as much noise as the replicate-based estimates. This indicates that some model-based noise will be generated while using them for control, but we expect the system to perform adequately.

Resistivity

Each of the 10 resistivity (site) models is a direct function of all nine (non-time) inputs. Based on the ANOVA information from the resulting linear resistivity models, the inputs that most strongly affect the outputs are Dopant Ratio, H_2 Flow, TCS Flow, and Deposit Temperature. Dopant Main Flow, while not providing large coefficients for the resistivity models, is forced to be included because it is a well known control knob for fine-tuning resistivity. It should also be noted that Dopant Ratio has by far the strongest effect on the outputs, and that all others have significantly weaker effects than those seen in the deposition rate models. Figures 7-16 through 7-20 plot the modeled resistivity for each radial site as a first order function of the aforementioned inputs. (For viewing purposes, the directions for increasing radius and input factor settings are reversed from those used in the deposition rate model plots.) For each simulation, the selected input is varied throughout its DOE range, while the remaining eight process settings are held at their DOE center point values.

Figure 7-16: Resistivity: 1st order functions in terms of Dopant Ratio

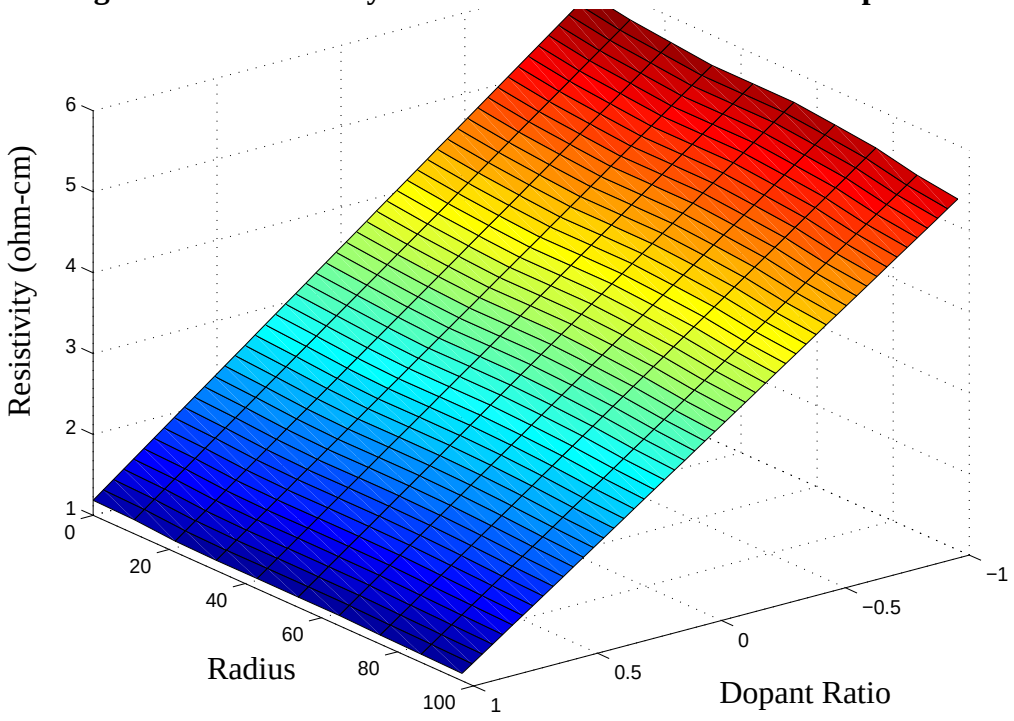


Figure 7-17: Resistivity: 1st order functions in terms of Main Dopant Flow

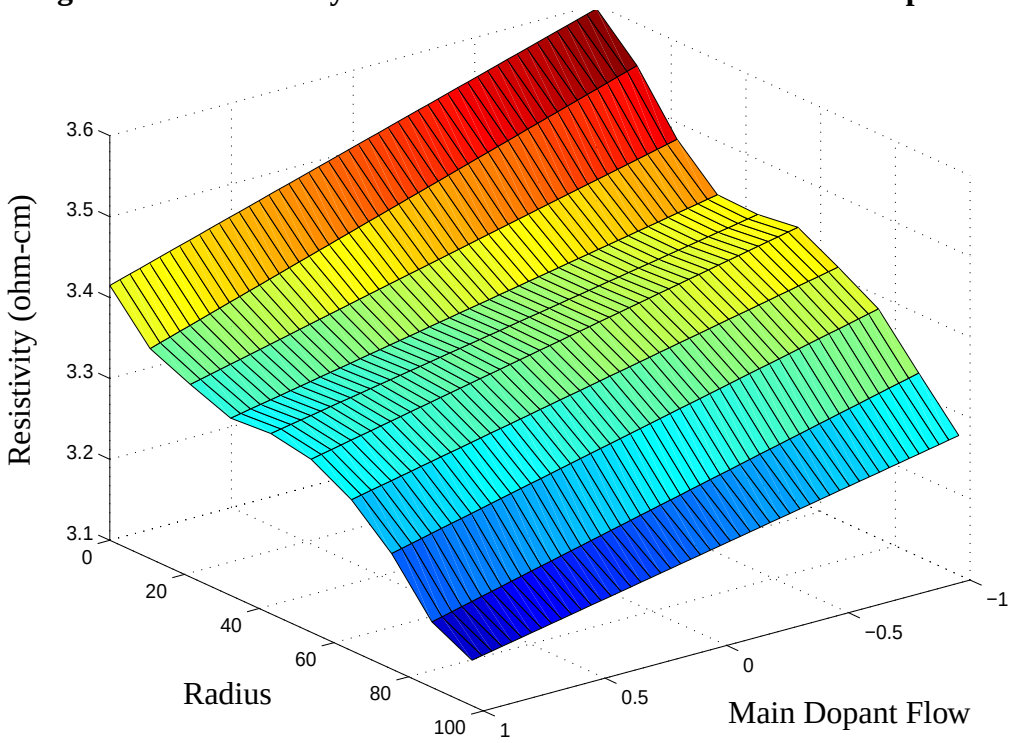


Figure 7-18: Resistivity: 1st order functions in terms of H_2 Flow

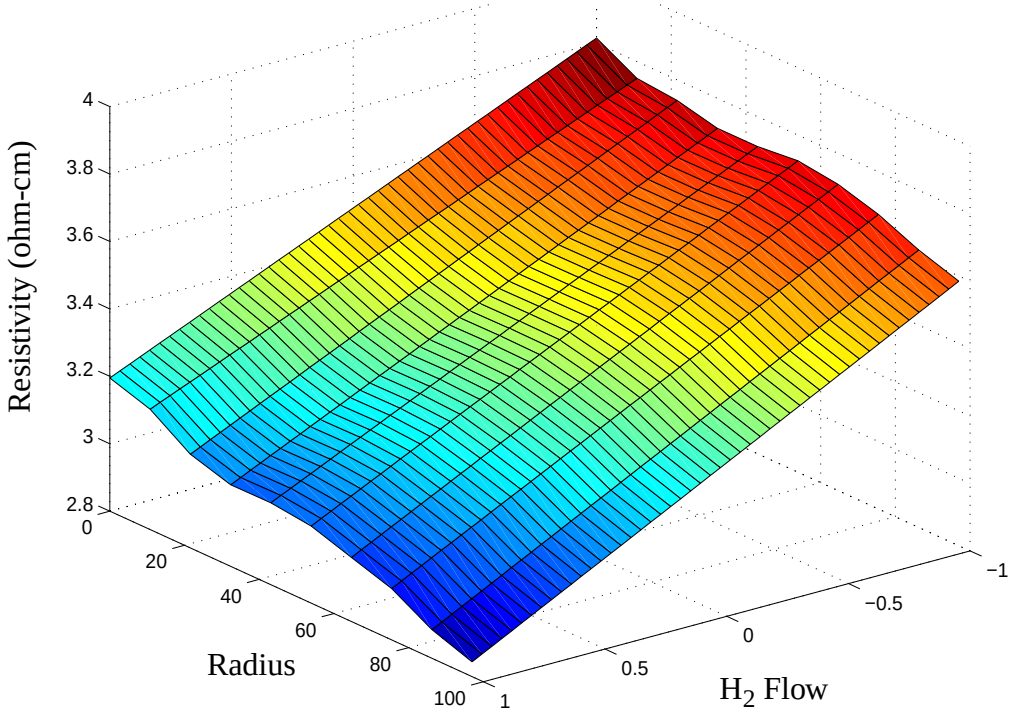


Figure 7-19: Resistivity: 1st order functions in terms of TCS Flow

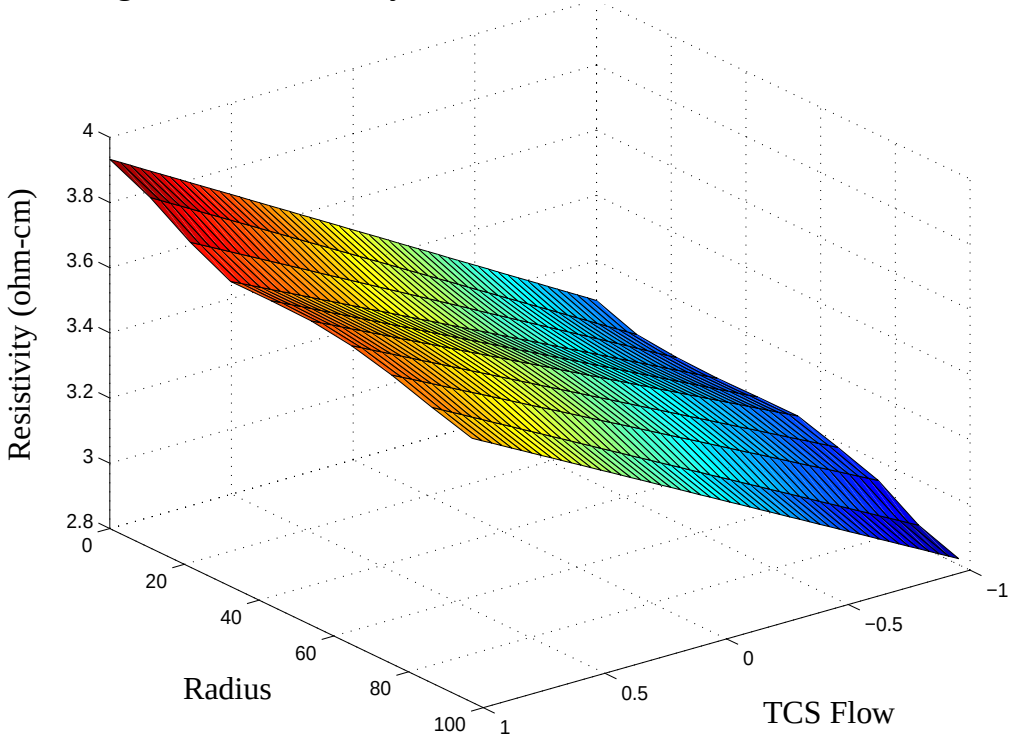
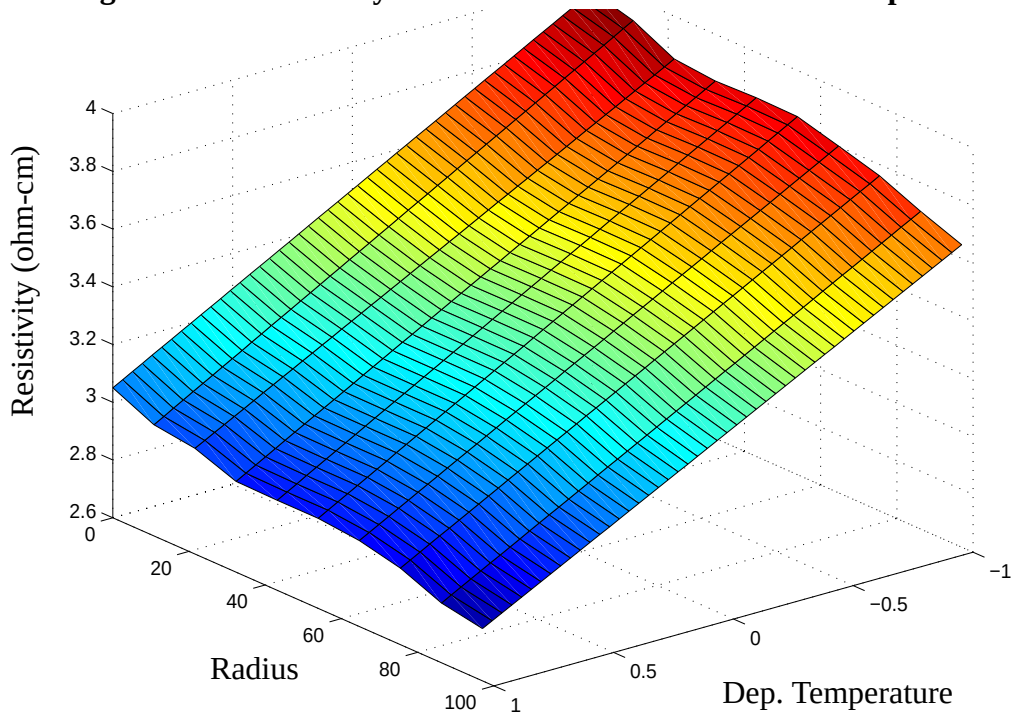


Figure 7-20: Resistivity: 1st order functions in terms of Temperature



The plots of uniformity as a function of Dopant Ratio, Dopant Main Flow, and TCS Flow make intuitive sense. Increasing the Dopant Ratio or the Dopant Main Flow provides more doping material to the deposition chamber, leading to higher doping concentrations in the epi film. (Keep in mind that increased doping concentration maps into decreased resistivity due to a $1/N$ type of relation.) Conversely, increasing the TCS source gas flow introduces more pure silicon into the system, yielding films with lower doping concentrations and higher resistivities.

Changing the dilutant H_2 Flows as shown in Figure 7-18 could affect the system through one or two mechanisms. First, the increasing H_2 gas flows could dilute the TCS source concentration more than dopant source gas, yielding an environment where the TCS source concentration decreases relative to that of the dopant source. Alternatively, the

altered H_2, TCS, and dopant concentration mix could yield a system with different relative chemical reaction or transport rates for incorporating silicon and doping material into the film.

There is not a clear explanation for the temperature induced effects seen in Figure 7-20. It is possible that temperature changes are altering the chemical and/or physical processes involved with incorporating silicon source and dopant material into the forming epi film. It appears as though increased temperatures result in an increased rate of dopant incorporation, relative to that of silicon.

Notice that the resistivity models do not demonstrate any strong controls for manipulating the resistivity uniformity. That is, the plots shown above demonstrate an ability to shift the resistivity profile up and down, but they do not indicate that the resistivity near the wafer's center can be adjusted relative to the resistivity near the wafer's edge. This makes sense, given the type of control knobs available in the system. The primary uniformity controls for thickness were through manipulating the relative gas flows to the inner and outer regions of the wafer. However, the relative composition of these two flows cannot be changed; the same source gases flow through each. Thus, while the resulting film may be nonuniform in thickness, we expect that the film composition, and thus resistivity, should be relatively uniform. Observations that this is indeed true also provide evidence for the assumption that thickness (and thus deposition time) does not affect resistivity.

As with the thickness models, the resistivity modeling quality can be considered using the (ANOVA-based) R-squared analysis. Estimates for the percentages of output variations that are accounted for by the models are found in Table 7-8. With the available unadjusted DOE data, the first order (linear) models capture an average of 91.9% of the

variation, while the second order polynomials capture approximately 97.6%. These results show that neither first order, nor second order models represent the resistivity data as well as the thickness data. There are any number of possible reasons, such as high measurement or process noise, or the response surface might simply require a high order model. However, with only 113 (or even 158) data points it is impossible to build a generic third order model for this system. A third order model with 9 inputs has 220 coefficients, which would require at least 220 data points for fitting purposes. While the second order models do provide better data fitting capability, it is hoped that the first order models will be good enough for closed loop control.

Table 7-8: Resistivity: R-squared results for model fit

Radial site (microns)	Data Unadjust 1st order	Data Unadjust 2nd order
0.00	0.9224	0.9807
9.70	0.9148	0.9710
19.40	0.9301	0.9835
29.10	0.9370	0.9892
38.80	0.9389	0.9886
48.50	0.9376	0.9889
58.20	0.9310	0.9852
67.90	0.9180	0.9757
77.60	0.8883	0.9544
87.30	0.8738	0.9415
Average	0.9192	0.9759

Once again, model error from the DOE runs is broken down into mean shift noise and site noise for comparison against the replicate-based noise estimates. These results are

shown in Table 7-9. (The table data for the replicate-based noise estimates are also provided for convenience.)

Table 7-9: Resistivity: Noise estimates from the process models

Estimate based on:	Type of Noise	Std. Dev. of Noise (microns)
1st Order process models	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.6117 (18.0%)
1st Order process models	Site noise ($\hat{\sigma}_{site}$)	0.1635 (4.8%)
2nd Order process models	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.3117 (9.16%)
2nd Order process models	Site noise ($\hat{\sigma}_{site}$)	0.1481 (4.35%)
Center Replicates	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0805 (2.87%)
Center Replicates	Site noise ($\hat{\sigma}_{site}$)	0.0376 (1.34%)
Fractional Factorial Replicates	Mean shift noise ($\hat{\sigma}_{\mu}$)	0.0389 (2.57%)
Fractional Factorial Replicates	Site noise ($\hat{\sigma}_{site}$)	0.0322 (2.13%)

As one might suspect from the R-squared analysis, both the first and second order models provide relatively noisy estimates of the DOE data. This noise is larger than that predicted by the replicates. It is expected that this will create less accurate resistivity control, especially when coupled with the fact that resistivity data are only available on a lot-to-lot basis. However, this information gives us the opportunity to run simulations with models of different fidelity and systems with different noise characteristics.

7.1.4 Process Optimization

The resulting site models for deposition rates and resistivities are combined with a cost function to model the cost in terms of the ten process settings. As presented in Section 4.2.2 (Equation 4-7), the cost function is

$$Cost = \sum_i [W_{ti}(T_{ti} - (r_i(\boldsymbol{x}) \cdot d))]^2 + \sum_j [W_{\rho j}(T_{\rho j} - \rho_j(\boldsymbol{x}))]^2 .$$

This function uses time-based models for thickness and may be combined with any type of underlying deposition rate and resistivity models, such as the first and second order polynomials described above. Constrained optimizations of the cost function provide optimal operating points for initiating run-to-run control experiments. The bounds for each input are set to the factorial design limits (± 1) shown in Table 7-1, which are presumed to contain the input space where the models are most accurate.

The following sections look at the details involved in selecting the exact cost function parameters and the subsequent optimization results.

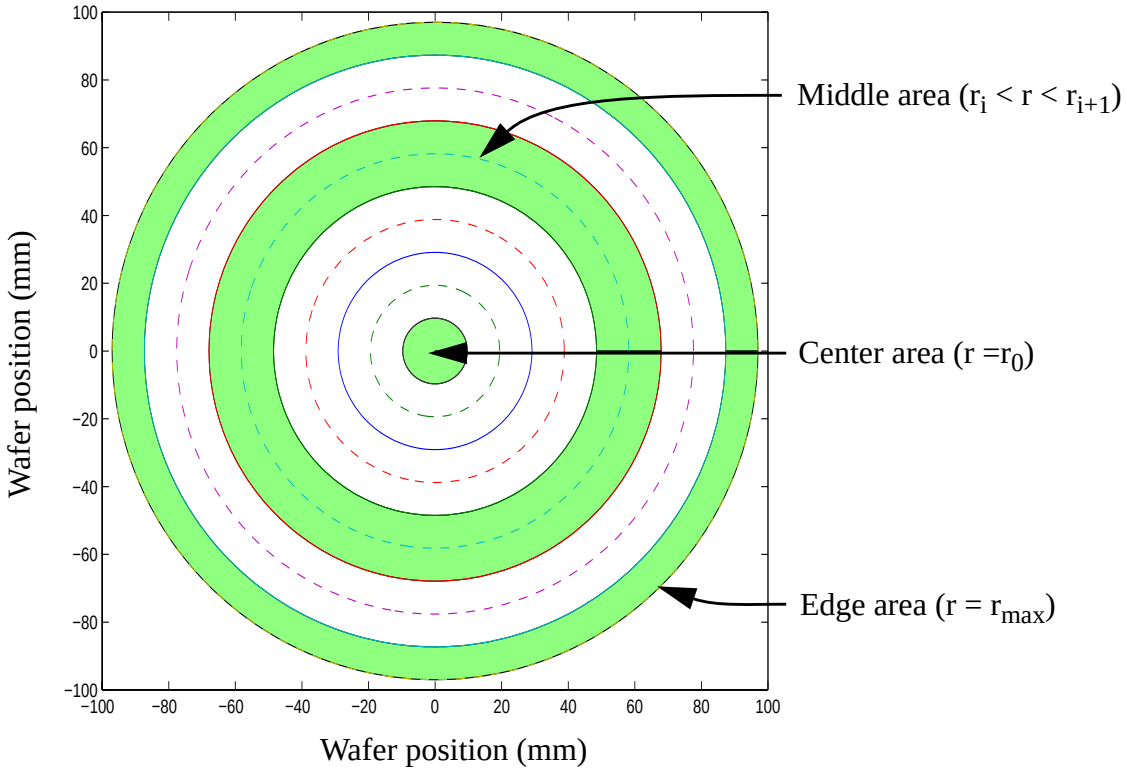
Selecting the Cost Function Parameters

Given the process models constructed in the previous section, the cost function need only be filled in with appropriate targets and weighting terms. Target values are (somewhat arbitrarily) selected to be near the measurements from the center point replicates. Target thicknesses are all set to be 3.00 microns and target resistivities are all set to be 3.36 ohm-cm.

The output weights are based on a number of factors, as described in Section 4.2.1. First the weighting terms for thickness and resistivity are initialized to the relative surface area that each radial site represents. These values essentially set the relative weights between outputs of the same type. Figure 7-21 shows a sample area weighting scheme, which is based on the radial position of the measurement sites. The dashed circles have radii at the measurement locations, while the circles with solid lines have radii that average the two adjacent measurement locations. The innermost circle (center point) and outermost circle are technically both dashed and solid. The representative regions for the endpoints are exceptions and must be handled differently. The “lower” radius for the cen-

ter area is forced to be zero, while the “upper” radius for the edge area is set equal to the maximum measurement radius. The algorithm makes no assumptions about any representative area beyond the last measurement radius.

Figure 7-21: Sample output weighting based on representative area



One concern arises from the form of the cost function, where the output weights are squared before acting on the error terms. For this reason the square root of the area-based weighting terms are kept. This concern shows itself repeatedly throughout the following manipulations.

This type of weighting is analogous to using a cylindrical shell approximation for error volume. Better approximations may be obtained using thin plate spline error volume estimation techniques, as presented in [DGLH96]. However, the use of a dense sampling pattern can eliminate the need for a more complicated weighting scheme.

Next, the weights need to be adjusted to properly reflect the trade-off between hitting target thickness versus hitting target resistivity, which is driven by a user supplied trade-off factor. For this work, the factor was chosen to be unity, implying that both thickness and resistivity are equally important. This factor could be selected by using a number of different criteria, including the following:

- Is it desirable for one output type to achieve target more closely? If so, use larger weights for those outputs.
- Are the process models more accurate for some of the outputs? If so, then the weights could be increased for the more accurate output models, since they are more likely to achieve the predicted results. Alternatively, the weights for the less accurate model outputs could be increased to force them closer to their targets, knowing that larger noise factors might make the predicted errors even worse.

This trade-off is controlled through two different adjustments. The relative weights within each output type are left unchanged during both of these adjustments. First, the area-based weights from above are scaled such that the ratio of their sum of squares (due to the fact that the weighting terms are squared) matches the given trade-off factor. Next, each output weight is divided by its respective target value. This ensures that errors are properly combined with respect to their scale. Considering that different output types might use vastly different scales, relative errors must be considered as a percentage of deviation from target.

Table 7-10 shows the initialization of and adjustments to the output weights based on this strategy. The thickness vs. rho trade-off factor is set to unity, the target thickness is 3.00 microns, and the target resistivity is 3.36 ohm-cm.

Table 7-10: Output weighting adjustments

Output	1)Area weights	2)Trade-off adjust	3)Target scaling
Thick (r = 00.00)	4.2982	3.8684	1.2895
Thick (r = 04.85)	12.1571	10.9414	3.6471
Thick (r = 09.70)	17.1928	15.4734	5.1578
Thick (r = 14.55)	21.0568	18.9510	6.3170
Thick (r = 19.40)	24.3143	21.8827	7.2942
Thick (r = 24.25)	27.1842	24.4656	8.1552
Thick (r = 29.10)	29.7788	26.8007	8.9336
Thick (r = 33.95)	32.1648	28.9481	9.6494
Thick (r = 38.80)	34.3856	30.9468	10.3156
Thick (r = 43.65)	36.4714	32.8241	10.9414
Thick (r = 48.50)	38.4443	34.5996	11.5332
Thick (r = 53.35)	40.3207	36.2884	12.0961
Thick (r = 58.20)	42.1136	37.9020	12.6340
Thick (r = 63.05)	43.8332	39.4496	13.1499
Thick (r = 67.90)	45.4879	40.9388	13.6463
Thick (r = 72.75)	47.0844	42.3757	14.1252
Thick (r = 77.60)	48.6286	43.7654	14.5885
Thick (r = 82.45)	50.1252	45.1124	15.0375
Thick (r = 87.30)	51.5784	46.4202	15.4734
Thick (r = 92.15)	52.9918	47.6923	15.8974
Thick (r = 97.00)	38.2032	34.3827	11.4609
Rho (r = 00.00)	8.5965	8.5965	2.5585
Rho (r = 09.70)	24.3172	24.3172	7.2373
Rho (r = 19.40)	34.3919	34.3919	10.2357
Rho (r = 29.10)	42.1201	42.1201	12.5358
Rho (r = 38.80)	48.6162	48.6162	14.4691
Rho (r = 48.50)	54.3511	54.3511	16.1759
Rho (r = 58.20)	59.5596	59.5596	17.7261
Rho (r = 67.90)	64.3329	64.3329	19.1467
Rho (r = 77.60)	68.7756	68.7756	20.4689
Rho (r = 87.30)	50.8606	50.8606	15.1371

Cost Function Optimization Results

Matlab's constrained optimization routine ("fmincon") is used with the cost function described above for both the first and second order models. As was shown in Section 5.1.4, even a simple overdetermined system with time-based models can result in multiple optimal solutions. The added constraints of finding solutions within a bounded space also provides the opportunity for local minima. Assuming an iterative optimization solver is used, different optimal solutions may be found by starting at different operating points. For this reason, a set of 1,024 different initial operating points is constructed and passed through the optimizer, one at a time. The points are selected through a two-level full factorial combination of all 10 inputs at their ± 0.5 levels. (Reasonable levels are selected for the deposition time, since it does not technically have ± 0.5 levels.)

Performing a complete optimization with a large number of iterations and a fine termination tolerance takes considerable computational time, even with the computing power of today's high end workstations. For this reason all of the starting points are first processed using a limited number of iterations and a course termination tolerance. The resulting set of optimal solutions are sorted to help find local minima. Figures 7-22 and 7-23 show the final optimized cost from all 1,024 optimizations, sorted from lowest cost to highest cost, for both first and second order models.

Figure 7-22: First order models: Sorted optimized cost

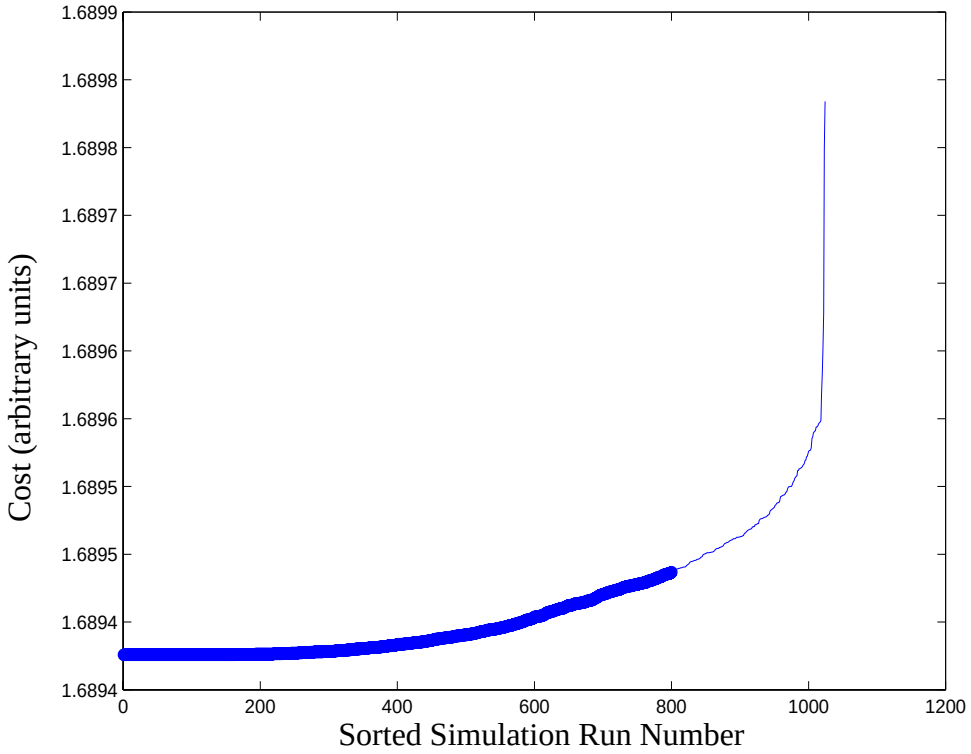
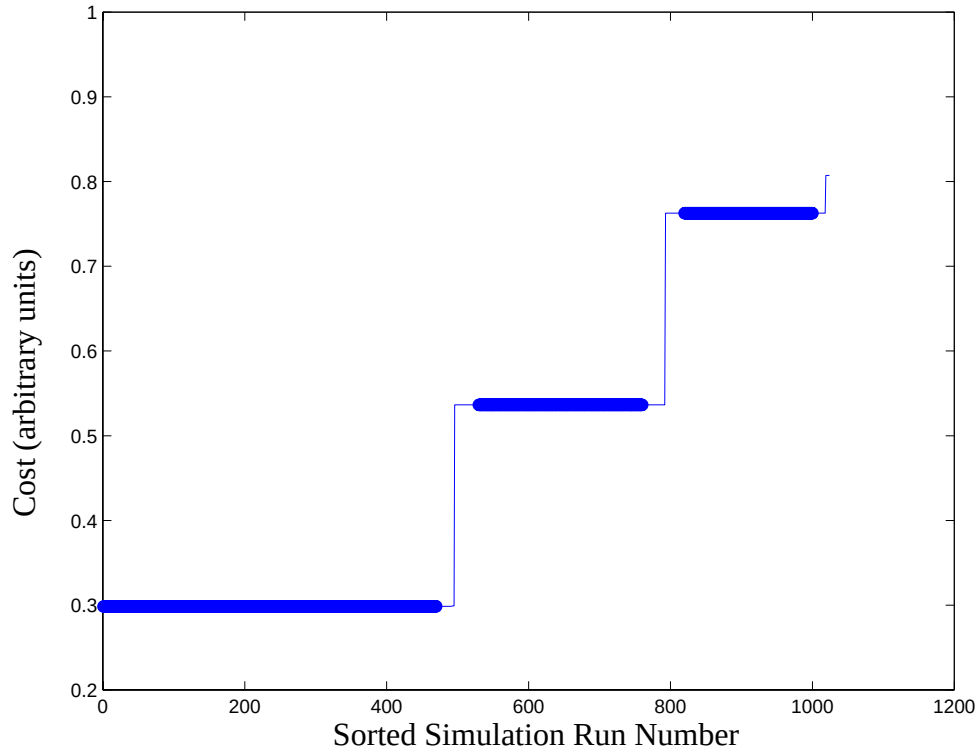


Figure 7-23: Second order models: Sorted optimized cost



The highlighted regions in each graph represent groups of optimizations where the first pass solutions provide nearly identical optimal input settings, and thus nearly identical costs. Exemplar solutions from each of these regions are passed through the optimization routine again, but with an extremely fine termination tolerance and an essentially unlimited number of available iterations. The results of the second optimization pass determine where actual local minima exist. (It is possible that some of these regions don't actually lead to minima, but simply "slowed down" the rough optimization process from the first pass on its way to true local minima.) We find that the first order models lead to a unique solution within the bounded region, while the second order models result in three different local minima.

Solutions from the first order time-based models are verified against that found by the iterative solver described in Chapter 5, and they are found to be approaching the same minimum. Clearly the simpler models lead to lower risk of multiple local minima lying within the bounded region, making these systems more suitable for use with an iterative solver. Note that the second order models are likely to provide better simulated results because they are able to find lower cost solutions, as seen in Figures 7-22 and 7-23. However, second order models make the optimization problem much more difficult due to increased complexity and computational demands, combined with the increased likelihood of obtaining multiple locally optimal solutions.

7.1.5 Run-to-Run Control Simulations

To demonstrate control, thickness and resistivity models are initialized to uniform radial profiles at target values. Process drift is added to the simulated system, while the controller attempts to maintain target output values. This scenario is considered first with a

system that presents no noise to the controller and allows the controller to apply input changes with very fine (nearly continuous) precision. Next, the controller acts on the same noiseless system, but is only allowed to use discretized inputs, where the discretization levels are selected to match those available on the actual processing equipment. The controller is then tested with a noisy system, where the process noise is similar to that found through the DOE replicate data. This system is again controlled using both continuous and discretized inputs.

The Noiseless Continuous System

The time-based controller is first tested with a drifting system that has no process or sensor noise. In addition, the controller is allowed to select inputs with a granularity (discretization level) of 0.00001, meaning that there are 20,000 discrete input levels available between -1 and +1. This enables the controller to select from a near-continuous set of input levels. The noiseless, continuous system provides a best case scenario, or benchmark, for how well the system could perform if the process models and sensors provide perfect information.

For convenience, the target thicknesses and resistivities are again uniformly set to 3.00 microns and 3.36 ohm-cm, respectively. These values are selected to be near the average outputs predicted by the process models when the center point inputs are applied to the system.

Figure 7-24 shows a drifting process where the uncontrolled thickness becomes gradually thicker with time (run number). Note also that sites near the center drift more quickly than those near the edge. Thus the films are gradually becoming thicker and less uniform. The full multi-objective controller is demonstrated by simultaneously engaging a drifting

resistivity model in the system. Figure 7-26 shows a drifting process where the mean resistivity across the whole wafer increases linearly with time, while the uniformity remains constant. It is important to note that the drifting effects shown in Figures 7-24 and 7-26 are taking place simultaneously, and thus the controller is compensating for both drifts simultaneously as well. Figures 7-25 and 7-27 plot the thickness and resistivity uniformities achieved by using the run-to-run controller, when applied to this drifting process. The controller uses EWMA weights (α) of 0.7 for all outputs.

Figure 7-24: Noiseless System: Uncontrolled Thickness

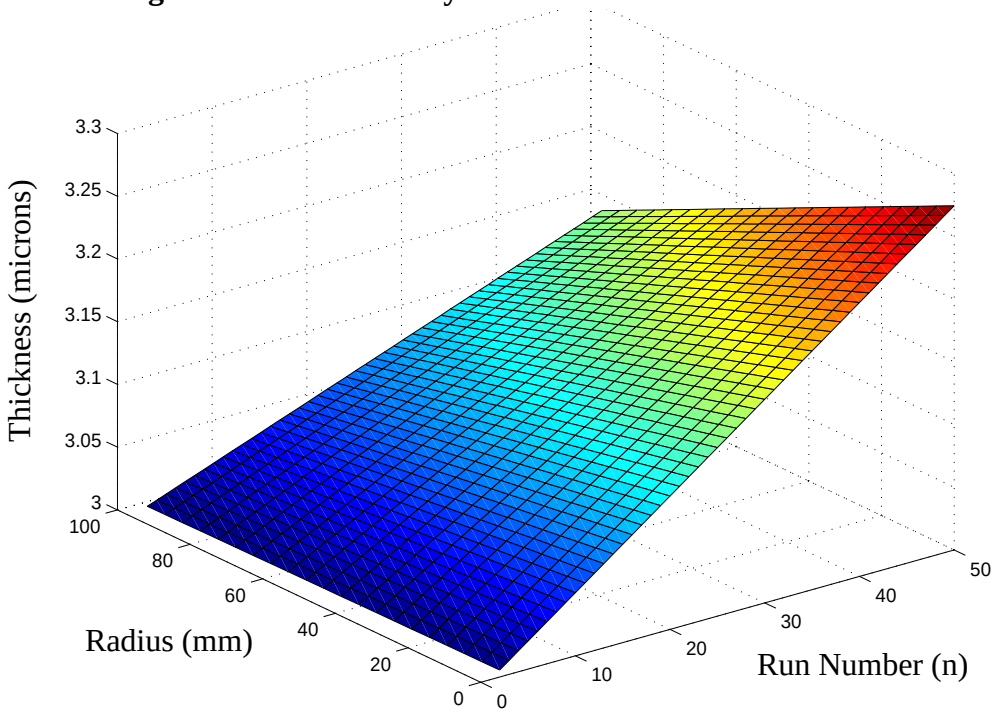


Figure 7-25: Noiseless Continuous System: Controlled Thickness

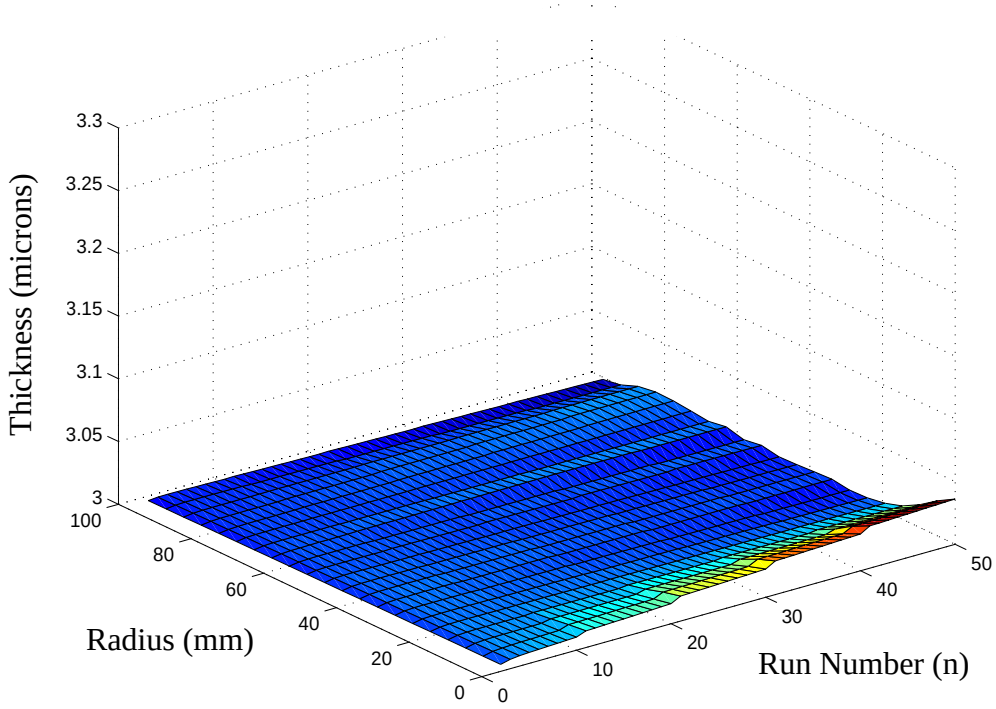


Figure 7-26: Noiseless System: Uncontrolled Resistivity

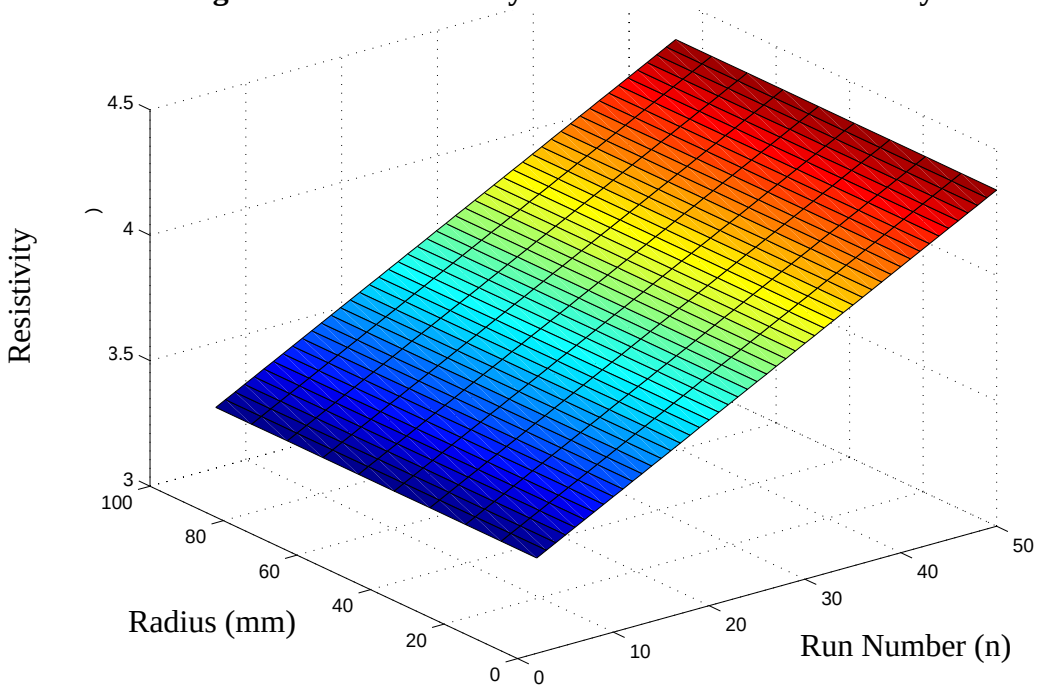
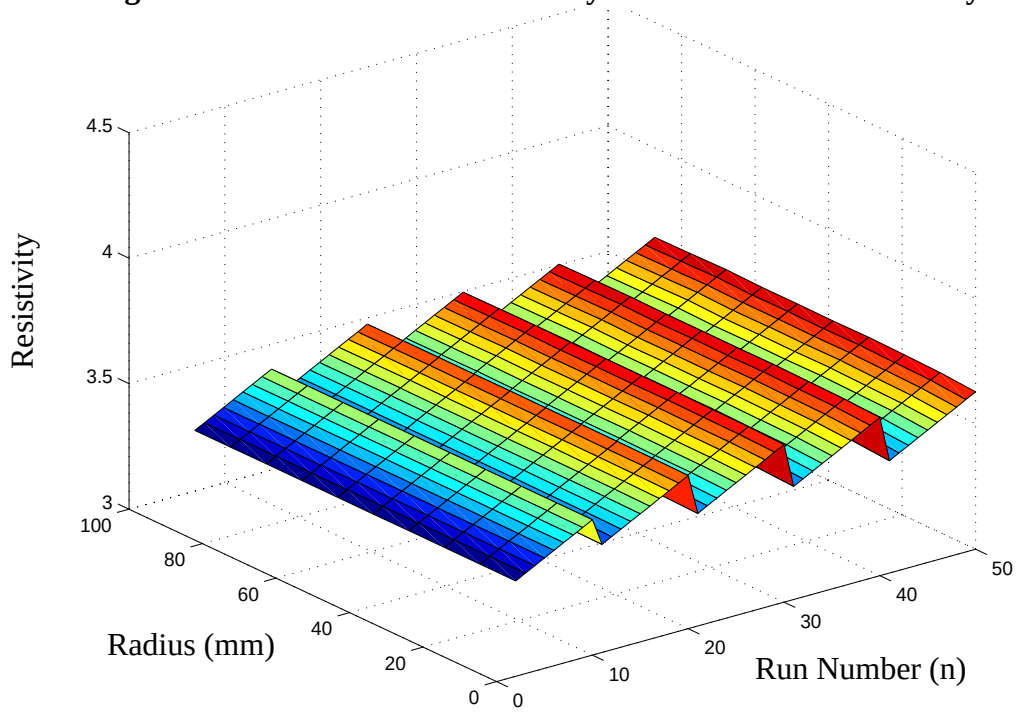


Figure 7-27: Noiseless Continuous System: Controlled Resistivity



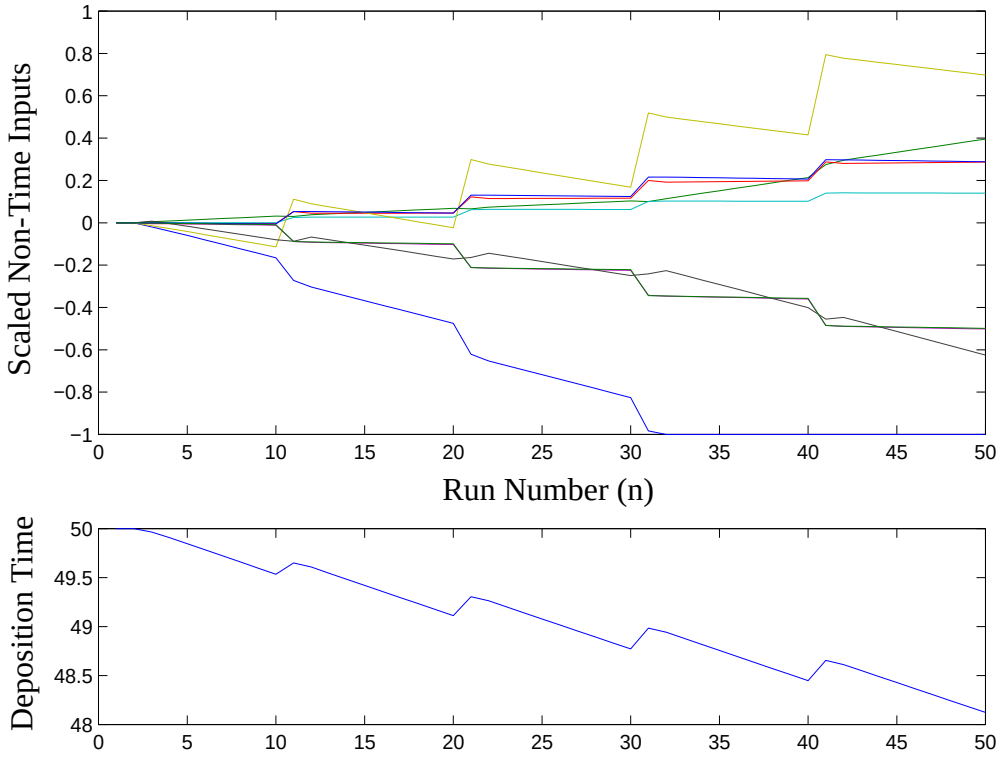
As shown by Figure 7-25, the controller is able to correct for the increasingly nonuniform thickness and maintain outputs very close to target. While it is difficult to see from these plots, the system accurately simulates the combined use of lot-to-lot and run-to-run thickness measurements. Once every ten runs (one “lot”), the controller is presented with the full set of thickness measurement sites. For the other nine runs in each lot, the error extrapolation joint sensor measurement technique described in Section 4.3.3 is used to fill in sites that are not sampled. Small disturbances in the controlled thickness plot are noticeable where full radial scan information is used. One could imagine using different sets of EWMA update weights, depending on whether the true off-line data are available, or if the system has virtual outputs from the in-line data. These simulations use the same weights for both cases. This technique works well for the noiseless simulations in part because the

uncontrolled thickness maintains a linear profile as the drift progresses.

Note the “bumpiness” of the controlled resistivity in Figure 7-27 versus the smooth surface of the controlled thickness. This part of the simulation demonstrates some effects of using lot-to-lot EWMA control for a rapidly drifting process. As described above, measured thicknesses are presented to the controller after every wafer, representing run-to-run control. Measured resistivities are only presented to the controller once every ten runs, representing lot-to-lot control. The controller cannot react to drifts between measured wafers, thus the process continues to drift during the processing of a lot. This type of drifting resistivity is not expected to occur in the actual equipment, so the results are primarily for demonstration purposes. If such persistent and rapid drifts are found to occur, a double EWMA (or PCC) controller can be implemented and applied to the problem [Smi96].

Figure 7-28 plots the input trajectories for the time and non-time inputs. The simulation requires changes in the non-time inputs that are comparable to their bounded ranges (+/- 1). Also, the lot-to-lot updates result in noticeable discontinuities (bumps) in the input trajectories, much like those seen in the controlled resistivity plot.

Figure 7-28: Noiseless Continuous System: Controlled Input Trajectories



The Noiseless Discrete System

The drifting processes shown in Figures 7-24 and 7-26 are employed once again, but this time the controller is required to select discretized process settings. Table 7-11 specifies the discretization levels available for the processing equipment. Figures 7-29 and 7-30 show the resulting controlled thickness and resistivity profiles. Figure 7-31 shows the input trajectory used to obtain these results.

Table 7-11: DOE input discretization levels

Factor	Discretization level
Deposit time	0.1 sec
Deposit temperature	1.0 ^{o}C
Dopant mixing ratio	0.1 %
Dopant main flow	0.1 sccm
% Lower power	1.0 %
% Inner power	1.0 %
Dilutant (H_2) flow	0.1 slm
Trichlorosilane (TCS) flow	0.1 slm
Center gas flow valve (Accusett inner setpoint)	1 “unit”
Outer gas flow valve (Accusett outer setpoint)	1 “unit”

Figure 7-29: Noiseless Discrete System: Controlled Thickness

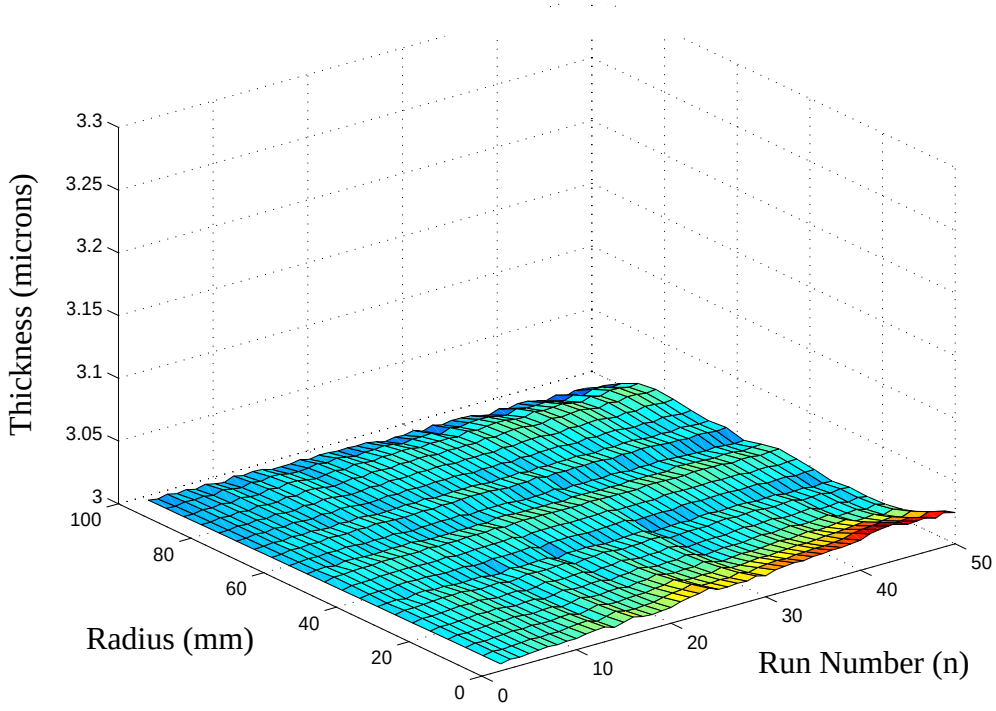


Figure 7-30: Noiseless Discrete System: Controlled Resistivity

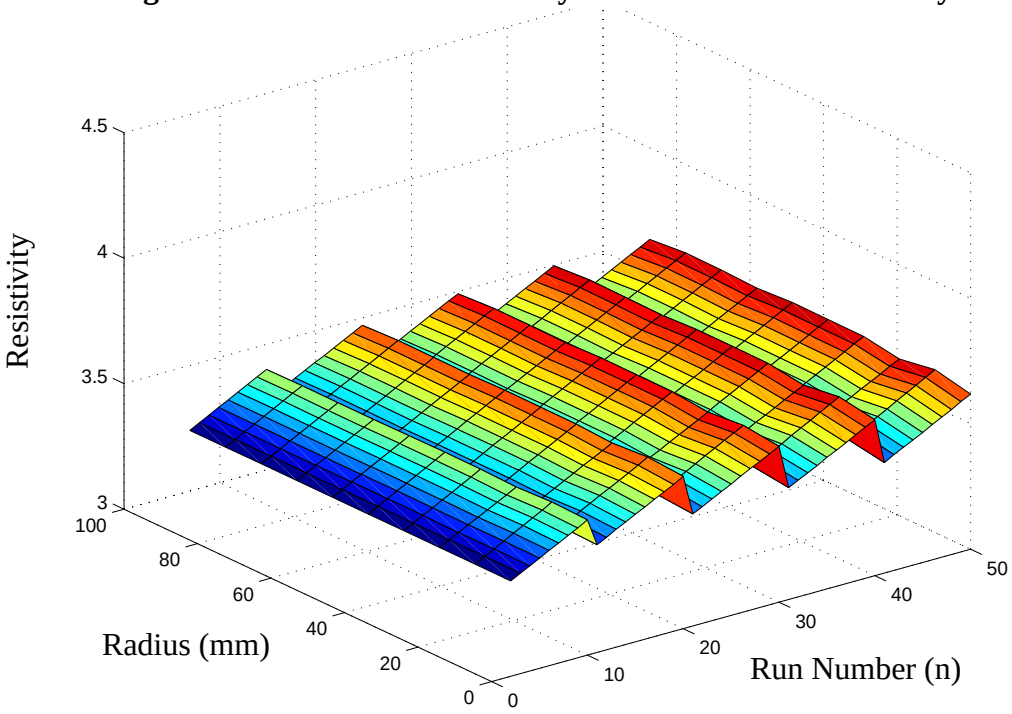
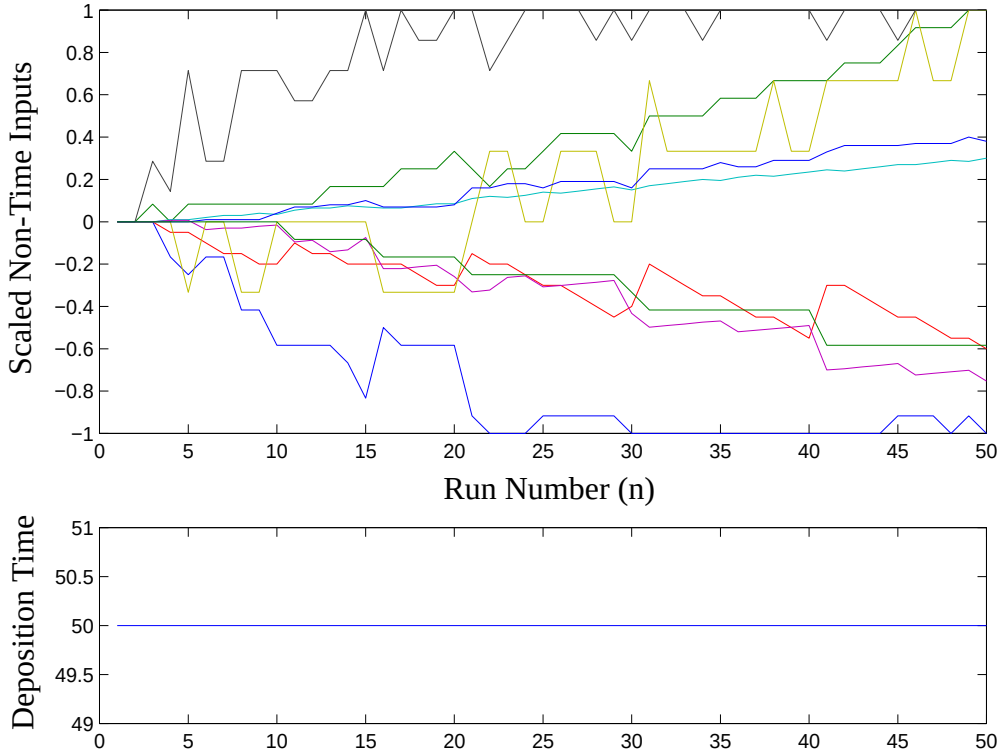


Figure 7-31: Noiseless Discrete System: Controlled Input Trajectories



The two controlled output plots (Figures 7-29 and 7-30) demonstrate a few noticeable artifacts, but the results appear to be relatively unaffected by limiting the controller in this way. We will see in the following sections that process noise provides considerably more variation than those induced by discretization effects. Figure 7-31 exhibits the same general trends seen with the continuous control system (Figure 7-28), but the actions taken by the controller are slightly more complicated (less smooth) to compensate for the use of discrete input values.

The Noisy Continuous System

While the scenarios described above provided some basic benchmarks for the epi control system, a more realistic simulation is demonstrated by including random process noise. Figures 7-32 through 7-35 show the uncontrolled and controlled thickness and resistivity profiles for a noisy, drifting process. Figure 7-36 shows the input trajectory used to obtain these results. The deterministic drifting components are identical to those found in Figures 7-24 and 7-26. The random noise components are generated by mean shift noise and site noise using the statistics from Tables 7-3 and 7-4.

Figure 7-32: Noisy System: Uncontrolled Thickness

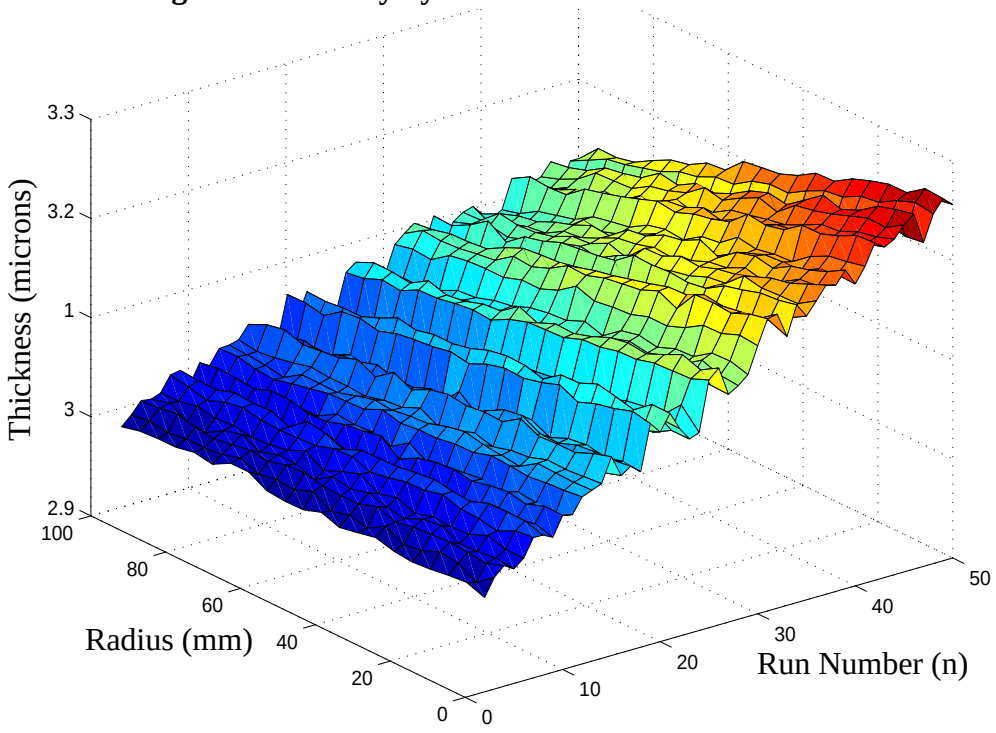


Figure 7-33: Noisy Continuous System: Controlled Thickness

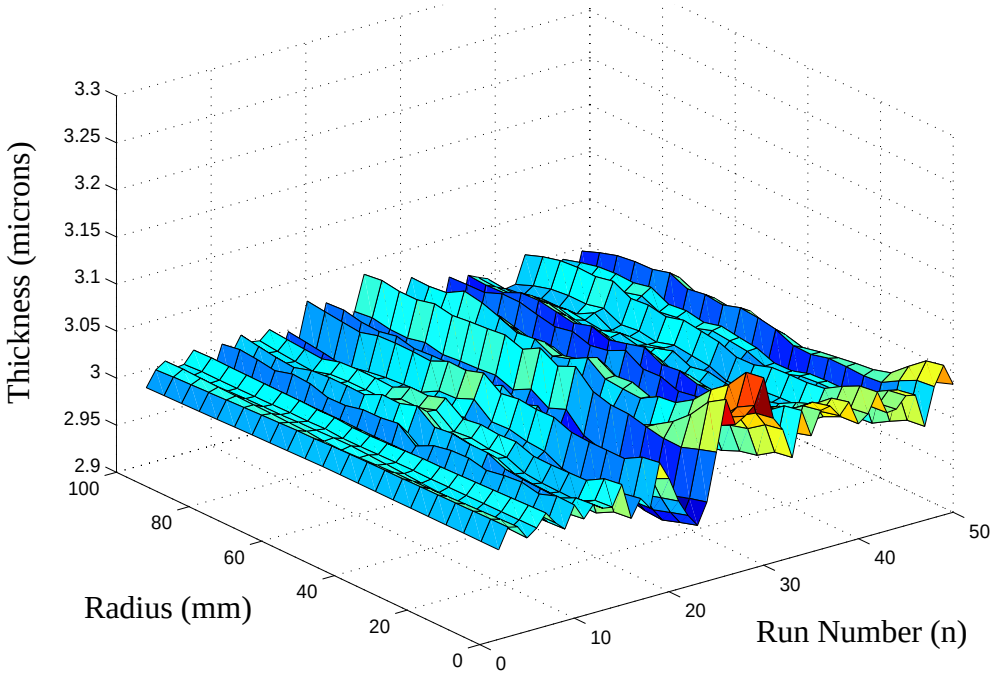


Figure 7-34: Noisy System: Uncontrolled Resistivity

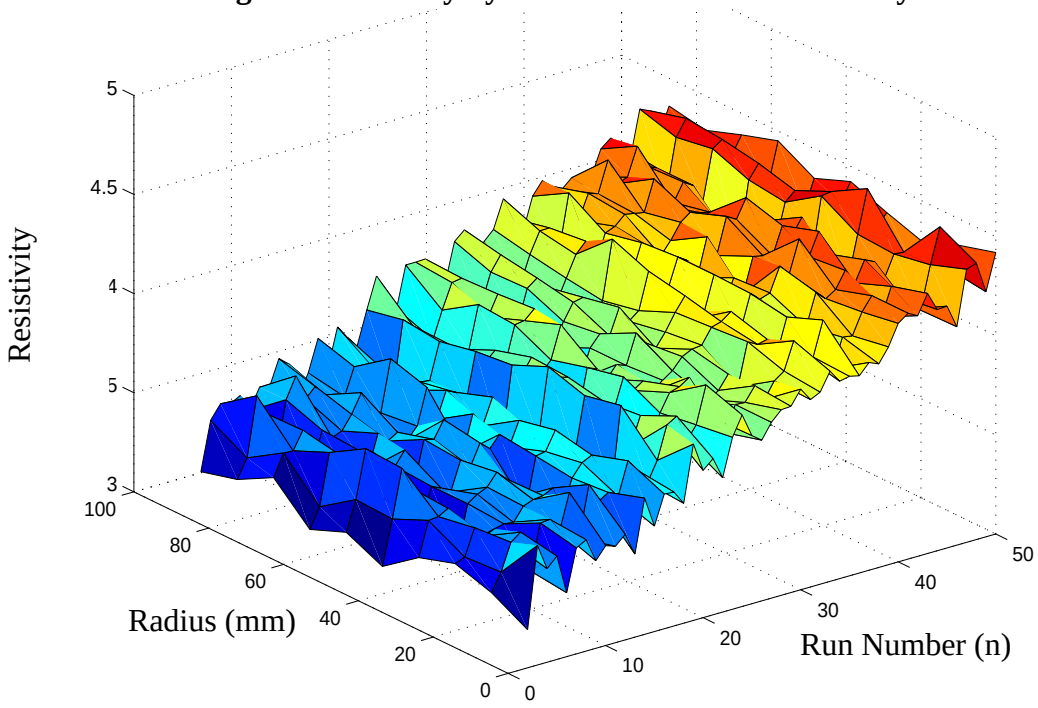


Figure 7-35: Noisy Continuous System: Controlled Resistivity

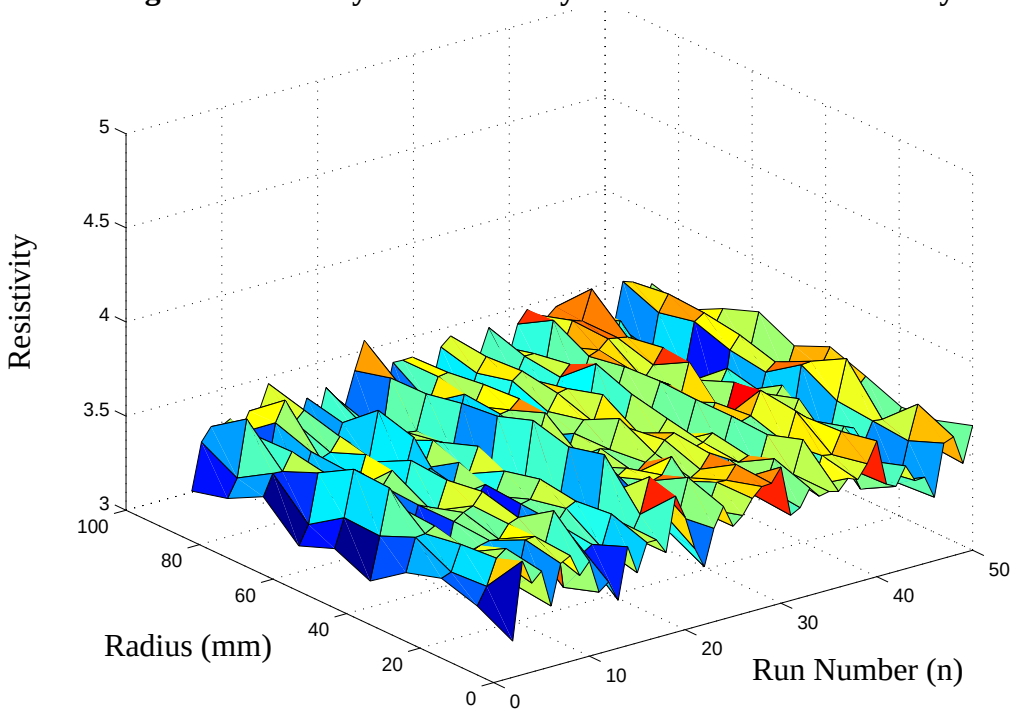
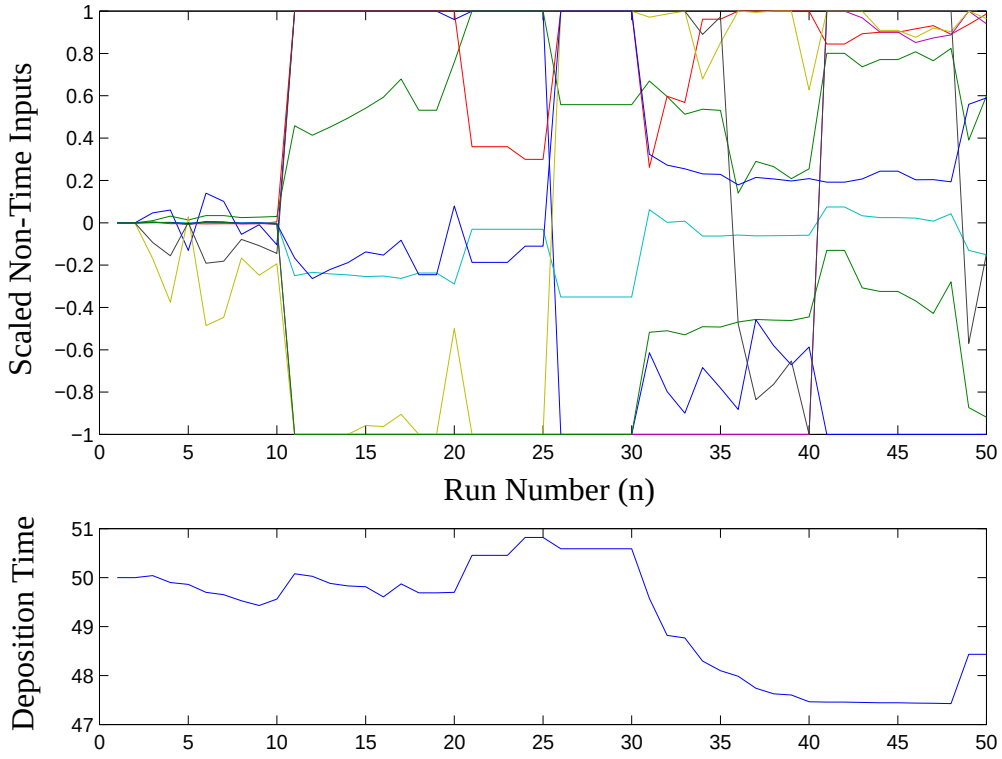


Figure 7-36: Noisy Continuous System: Controlled Input Trajectories



The controller is able to maintain fairly good thickness and resistivity control despite the addition of process noise. Variation seen in the controlled profiles appears to be similar to, but slightly larger than, the underlying process noise. For thickness control, additional noise comes from the drift components and the error extrapolation strategy. Additional noise in the controlled resistivity primarily comes from process drift combined with lot-to-lot model updates. Notice that the obvious “bumps” in controlled resistivity seen with control of a noiseless system are almost completely hidden by the addition of process noise. The resistivity process noise is considerably larger than that seen in the thickness measurements. The input trajectories for the noisy system swing much more wildly than either of the noiseless systems. If this appears to degrade the performance of a real system, it could be addressed by lowering the EWMA weights and/or penalizing input changes,

the use of which will be described and simulated in a subsequent section.

The Noisy Discrete System

For completeness, the noisy drifting system of Figures 7-32 and 7-34 is controlled using discretized inputs, as was done with the noiseless system. Figures 7-37 and 7-38 show the controlled thickness and resistivities, respectively, while Figure 7-39 shows the input trajectory used to obtain these results. The results do not change significantly by constraining the controller in this manner; additional noise seems to affect the system much more strongly than discretization.

Figure 7-37: Noisy Discrete System: Controlled Thickness

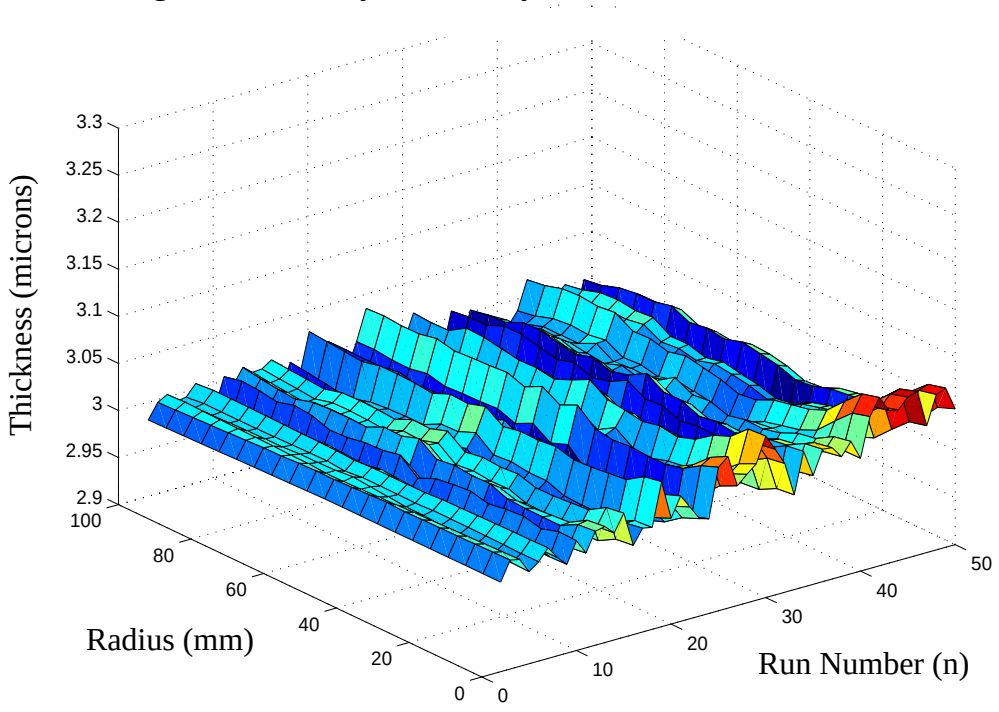


Figure 7-38: Noisy Discrete System: Controlled Resistivity

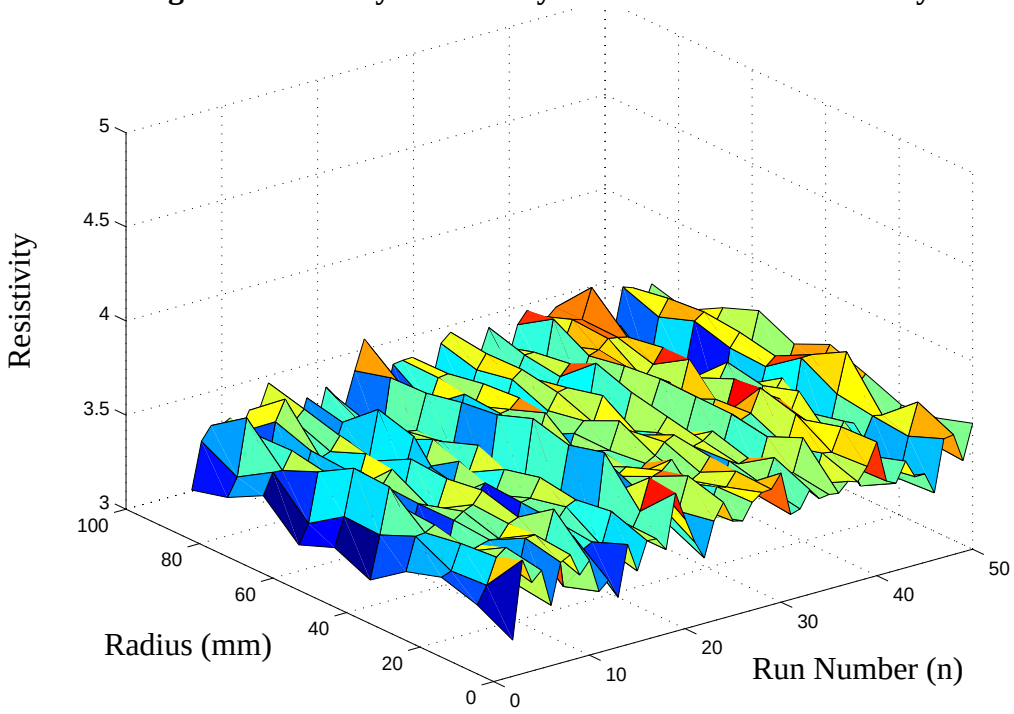
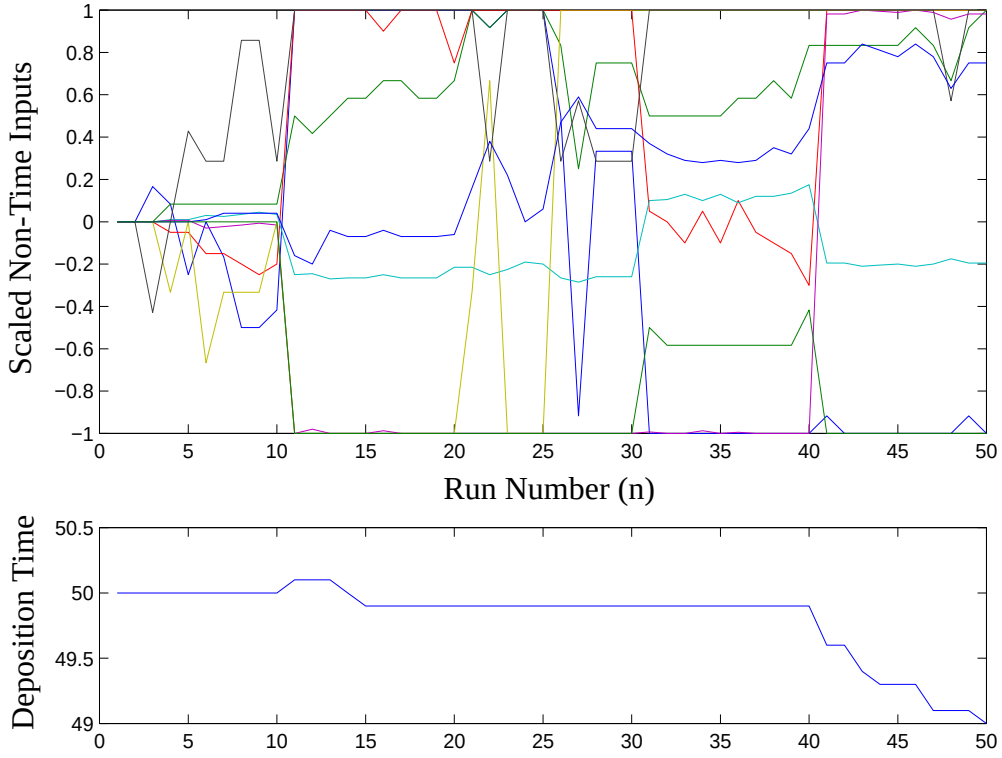


Figure 7-39: Noisy Discrete System: Controlled Input Trajectories



Reducing Input Trajectory "Jitter" (for Noisy Systems)

The extreme swings in input trajectories for the noisy systems are a significant concern, potentially restricting the likelihood that an industrial partner would test and adopt this system. As mentioned in the previous sections, there are two straightforward ways to deal with this problem, namely, lowering the EWMA weights to better filter out noise, or by creating additional cost to moving in the input space.

Lowering the EWMA weights is the most straightforward approach, as it requires a simple parameter change within the simulation system. The noisy drifting system of Figures 7-32 and 7-34 is controlled using discretized inputs and EWMA model updates with the alphas (weighting terms) set to 0.3 instead of 0.7. Figures 7-40 and 7-41 show the controlled thickness and resistivities, respectively, while Figure 7-42 shows the input trajectory used to obtain these results.

The controlled outputs exhibit better noise suppression due to the increased filtering, but they also show a greater offset between the controlled outputs and the target outputs. (This is particularly noticeable in the controlled resistivity plot.) This type of behavior is the expected result of using a low EWMA weight on a drifting system. Looking at the input trajectories, however, we still see fairly strong parameter swings, even with the low EWMA weights, meaning this strategy does not look promising.

Figure 7-40: Noisy Discrete System: Controlled Thickness (w/ Low Alpha)

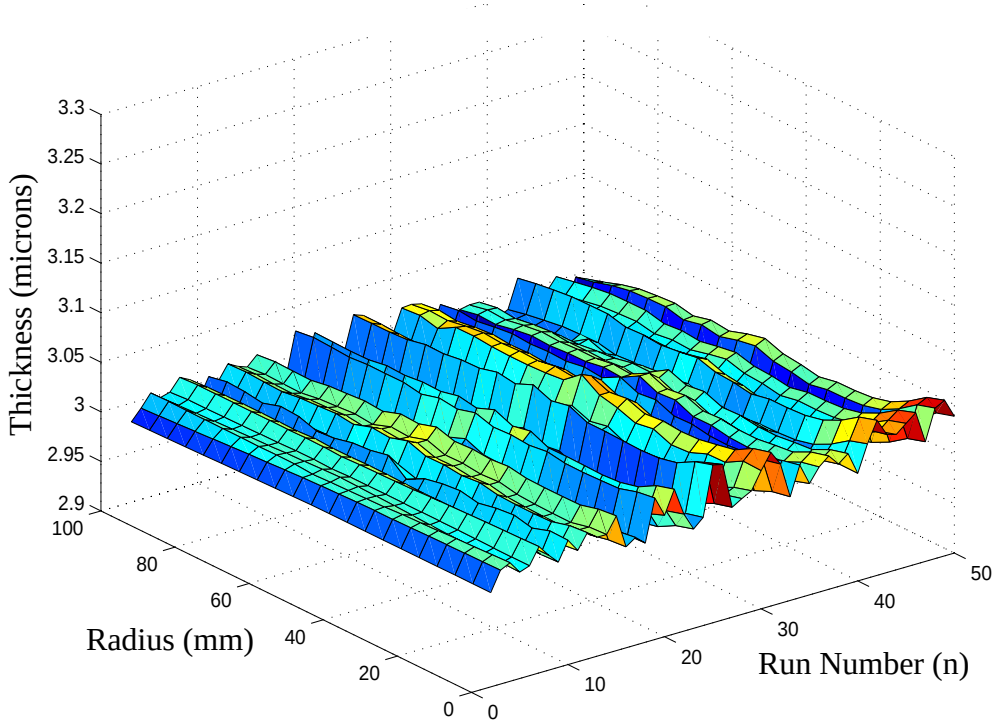


Figure 7-41: Noisy Discrete System: Controlled Resistivity (w/ Low Alpha)

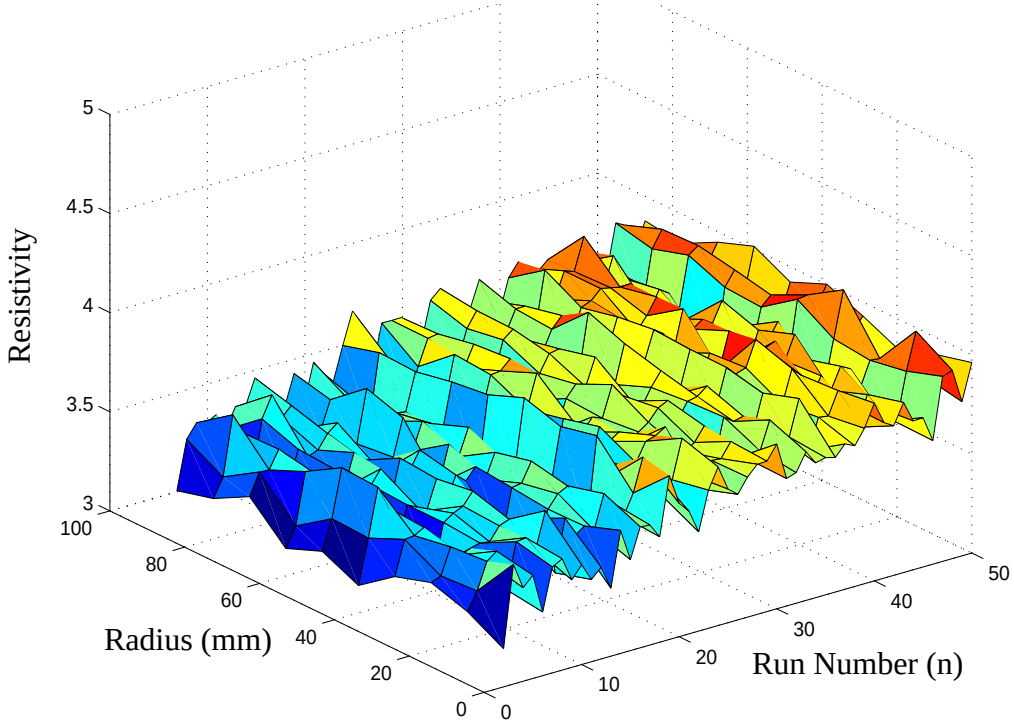
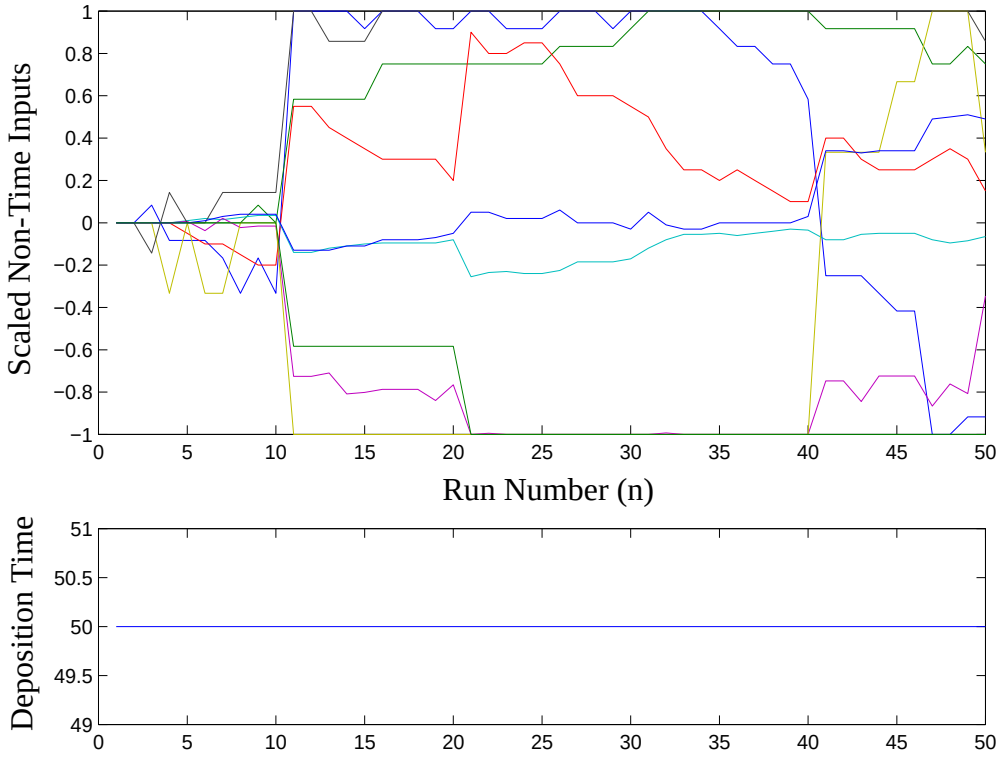


Figure 7-42: Discrete System: Controlled Input Trajectories (w/ Low Alpha)



Next consider adding a cost structure that penalizes movements in the input space between one run and the next. This can be implemented fairly easily by creating an additional set of outputs that track the inputs. Time-based inputs can be tracked using rate models whose slope coefficients are all zero ($\boldsymbol{A_t} = \boldsymbol{0}$) and offsets are set to unity ($\boldsymbol{b_t} = \boldsymbol{1}$). Non-time inputs are tracked using slope coefficients equal to one for each specific input and zero for all other inputs ($\boldsymbol{A_{nt}} = \boldsymbol{I}$), and all offsets set to zero ($\boldsymbol{b_{nt}} = \boldsymbol{0}$). The targets for these outputs must be (continuously) set equal to the current suggested inputs.

The issue of setting the appropriate weighting terms comes up again here. The relative weights between each input can be set with user defined criteria. However, it makes sense to at least scale them by the inverse of their allowable ranges (maxima minus minima) to produce equivalent cost when swinging across the full input ranges. Further, we must

select the overall importance of trading off errors in outputs vs. movement in the inputs. For the following simulations, errors in outputs were chosen to be one hundred times more important than moving the inputs.

The noisy drifting system of Figures 7-32 and 7-34 is controlled using discretized inputs and additional cost for moving in the input space. Figures 7-43 and 7-44 show the controlled thickness and resistivities, respectively, while Figure 7-45 shows the input trajectory used to obtain these results. We have traded off a small amount of output error for stabilizing the inputs, which demonstrate fairly smooth trajectories, especially when compared to the previous simulations with a noisy system.

Much of this added stability comes from the structure of the resulting weighting and slope matrices. The underlying linear solver must perform a matrix inversion that is based on these two matrices. The condition number (ratio of the largest singular value to the smallest singular value) for the matrix to be inverted decreases from approximately 1.38×10^{10} to 2.15×10^{3} after including the additional input weights, a reduction of nearly seven orders of magnitude. It is of little surprise that the solver achieves more stable solutions to the system when input weighting is included.

While it is difficult to see from these plots, it has been verified that most of the "wilder" inputs from the previous simulations remain at zero for the system with additional input weights. This indicates that these inputs have relatively little effect on the outputs; it must cost more to move them in the input space than can be gained from any resulting movements in the output space. It also follows from this logic that "wild" changes in these particular inputs will not result in "wild" swings in the outputs, assuming that we have fairly accurate process models.

Figure 7-43: Noisy Discrete System: Controlled Thickness (w/ Input Weight)

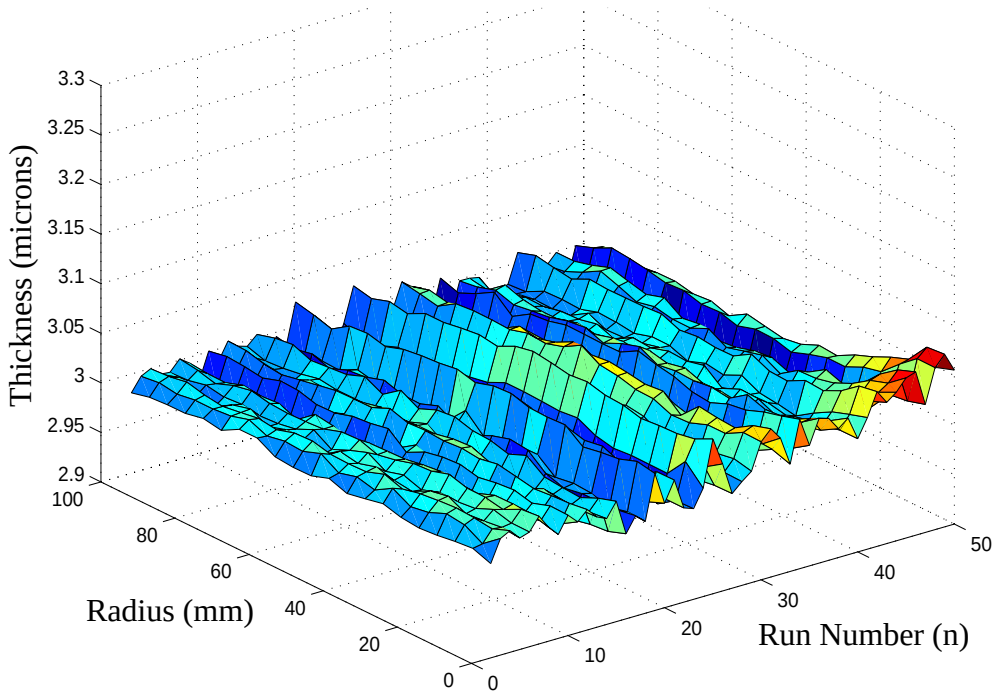


Figure 7-44: Noisy Discrete System: Controlled Resistivity (w/ Input Weight)

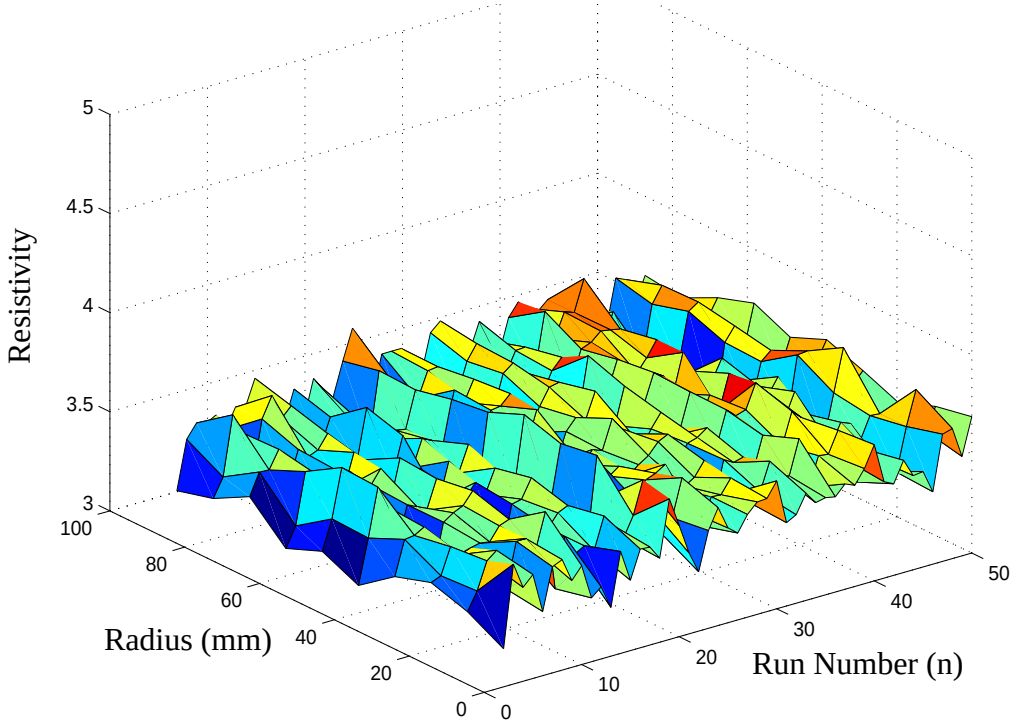
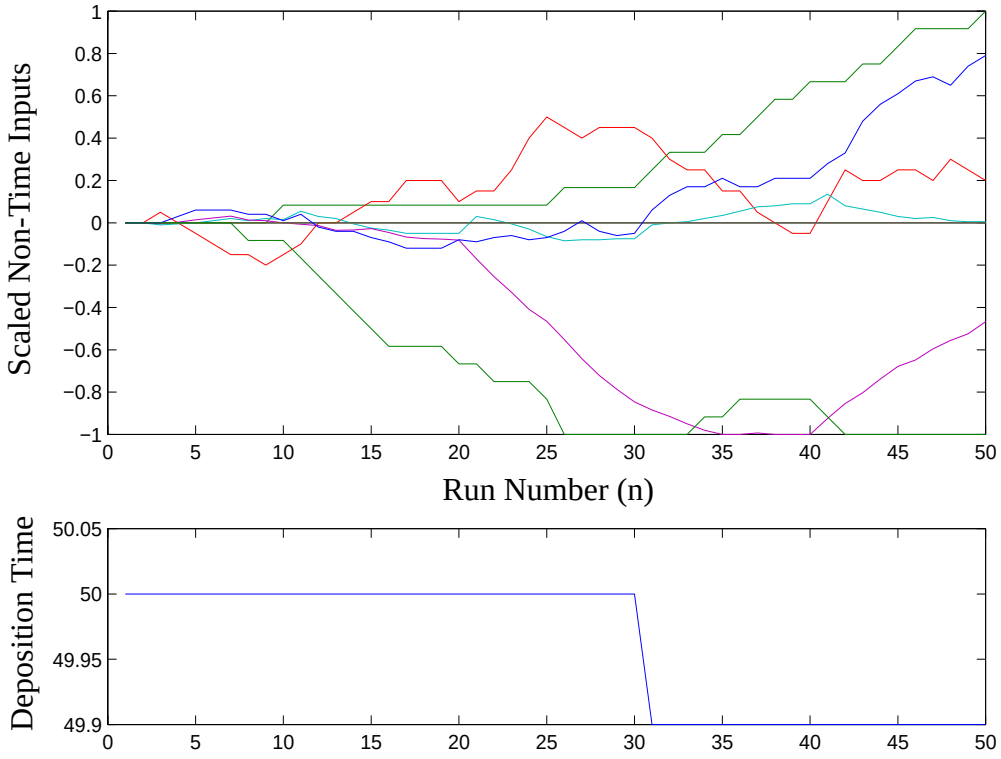


Figure 7-45: Noisy Discrete System: Controlled Input Trajectories (w/ Input Weight)



7.2 Automated Control Runs (Industrial Scenario)

The analyses and simulations described above indicate that this epi deposition system is a good candidate for run-to-run (and lot-to-lot) control. MIT, On-Line Technologies, Applied Materials, and Wacker Siltronic are jointly developing a test system for demonstrating multi-objective epi deposition process control at Wacker's facilities in Portland, Oregon. Wacker Siltronic is Wacker's semiconductor division, a producer of "blank" silicon wafers for chip-makers. They sell uniformly doped bulk silicon wafers, as well as wafers with a blanket layer of epi. For this work, Wacker is providing integration guidance and access to one of their production epi deposition tools.

The direct usefulness of the original DOE data is limited because the control experiments will be run on a different processing tool than that used in the DOE. The same type

of equipment will be used, but not the exact same tool. However, the demonstration will make use of a true production class system instead of an isolated development tool, like that used for the original DOE. The following sections describe the experimental design and execution details for the automated control experiments. These experiments are scheduled to take place, but could not be completed in time for inclusion in this thesis. We expect to report the results at a future time.

7.2.1 Experimental Design

Ideally a full DOE as described above is performed and analyzed to create models for process control. Run-to-run control is then demonstrated using the resulting models and the same equipment, shortly after finishing the DOE. However, as noted above, we will perform our experimental demonstration of automated run-to-run control at a different facility than the DOE. While the same type of processing equipment will be used, we are not able to use the exact same tool. Additionally, the experiments will be run more than a year and a half after the original DOE. For these reasons the original DOE data are of limited use.

Another complication is Wacker's desire to explore the control of epi deposition in both the backsealed and non backsealed configurations. As described in Chapter 2, processing in a backsealed configuration isolates the wafer's backside from the process chamber conditions, stopping deposition of epi on the backside and eliminating the redistribution of dopants from the backside to the outer edges of the wafer's surface (autodoping). Autodoping in a non backsealed configuration requires the use of an auxiliary dopant flow, which provides extra dopant at the center of the wafer, to achieve a uniform resistivity profile across the wafer.

Backsealed wafers were used in the original DOE, meaning that resistivity uniformity was not a concern, and that the auxiliary dopant flow was not used. However, control of non backsealed wafers requires us to model and use the auxiliary dopant flow input setting. Therefore some data must be collected on the effects of this input before process control can be attempted.

The use of a different piece of equipment, coupled with the desire to test a non backsealed process chamber configuration, requires us to perform a new DOE for the control experiments. Actually, two DOE's are created for execution, one for the backsealed configuration and one for the non backsealed. However, the available resources do not permit the execution of two large DOE's, like the 158 runs described earlier. Smaller "tuning" DOE's must be used instead, where approximately one cassette of 25 wafers can be allocated for each of the two chamber configurations. The tuning DOE's are specified by selecting the input ranges ($-\alpha$, -1, 0, +1, $+\alpha$) and the combinations of inputs, as described below.

Input Range Selection

An important underlying goal for specifying the tuning DOE's (and the control scenario) is to retain as much as possible from the original DOE structure. For selecting new input ranges, a process engineer will first tune the process, starting with the center point from the original DOE. The goal is to have a well optimized center point to start from, which will hopefully lead to a local optimum within the design space.

After adjusting the center point, the upper and lower ranges ($-\alpha$, -1, +1, $+\alpha$) are selected such that the size of the overall process space is the same as that of the original DOE. Essentially the new input space becomes a shifted version of the original; the new

space has the same “widths” along the input dimensions (^{o}C, sccm, slm, etc.). This helps ensure that modeling analyses and assumptions from the original DOE remain valid.

Selecting the Design Points

Assuming that the equipment used in the original extensive DOE has similar processing characteristics to the tool at Wacker, the data from that DOE can help in selecting good tuning DOE’s. Recall that linear models of deposition rate and resistivity capture more than 90 percent of the variation in the original DOE data, and that we expect these models to be used for control. When considering a central composite DOE, the axial runs provide a small set of experiments that should produce good models for systems with little or no interactions between the inputs (e.g. a linear system). Each axial run sets one of the inputs to either the plus or minus α level, and leaves all other inputs at their center point levels. Thus the axial runs provide information about the effects of each input, one at a time. Center point replicates can also be included to provide better estimates of the constant terms and to check for linearity.

This idea can be tested with the DOE data. First we extract the outputs from the axial and center point runs to build linear models from that subset of the data. Then we can compare the resulting slope terms with those created from the full data set. Because the EWMA control algorithm modifies and optimizes the constant terms, we don’t have to worry about comparing the DOE-based (fitted) constant terms. These take care of themselves during the control runs, but slope terms are fixed for the duration of the experiment. Figures 7-46 and 7-47 graphically depict the slope matrices of the linear deposition rate models that are fit using the full DOE data set and the subset of axial runs, respectively. The slope terms form a matrix with 9 coefficients for each of the 21 radial sites. In the

plots, each line represents the slope coefficient for a given input across the radius of the wafer. The collection of coefficients at a given radial site (along a vertical line) are the slope terms corresponding to that site.

First it is worth noting the striking resemblance of these two plots. The slope matrix of the first figure was found by fitting linear models to 158 data points, while the second was created with data from only 18 axial and 8 center points (19 different recipes). Using the largest coefficient from these matrices as a reference, the two slope matrices have a maximum error of 10.0% and an average error of 3.26%.

The coefficients from the first figure are found using the statistically driven fitting (backward elimination) method described in Section 7.1.3. The coefficients in the second figure are found with a single-step fitting process that keeps all coefficients. This is done because the backward elimination process tends to “zero out” many more input factors when fitting against the 26 axial and center points, than it does when fitting with the full 158 design points. Most of the small errors near the “radial site axis” (slope coefficient = 0) are due to this discrepancy in fitting algorithms.

Figure 7-46: Thickness Linear Model Coefficients: Full DOE

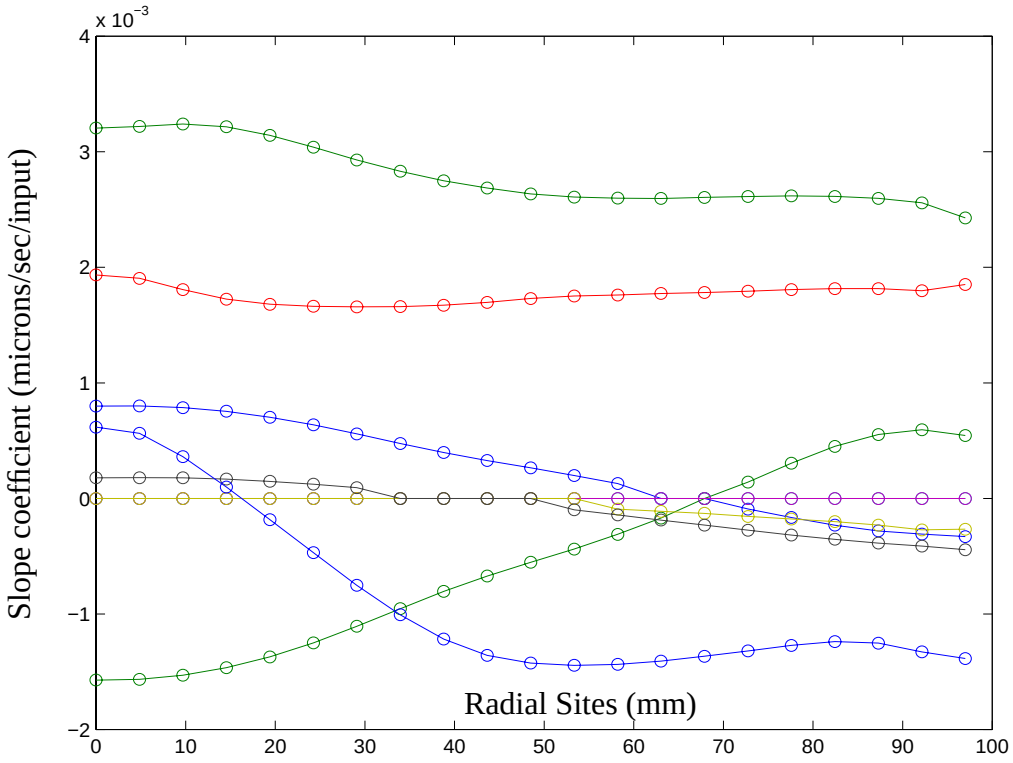
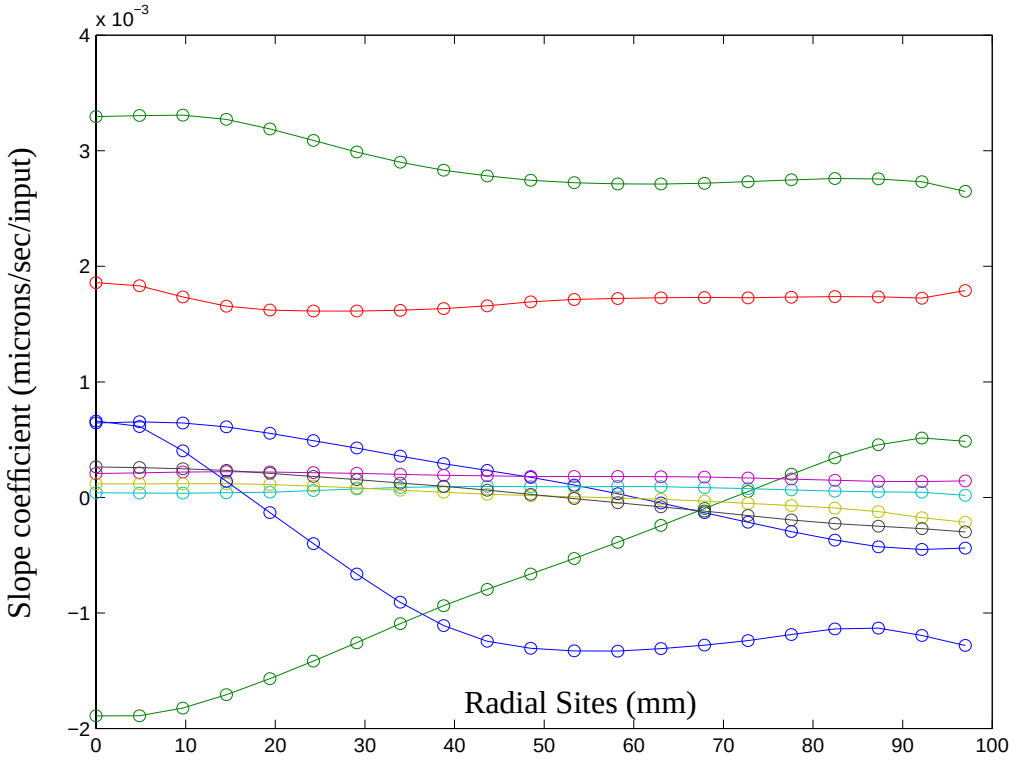


Figure 7-47: Thickness Linear Model Coefficients: Axial and Center Points



Figures 7-48 and 7-49 show the slope coefficients for the 10 resistivity sites using the full DOE and subset of axial runs, respectively. Unfortunately there were three axial runs included among the wafers for which we have no resistivity data. Both the positive and negative axial runs for the “Dopant Ratio” are missing, which means that there is no slope information for this input in the available axial/center runs. Therefore the slope terms for this input are not shown.

Based on the previous noise and modeling analyses, we expect these results to be worse than those found with the thickness data. The two plots show this to be true, as we find a maximum error of 63.2% and an average error of 16.2%. Still, we can clearly see that most of the slope information is retained when using only the axial and center point data. Also, with a small tuning DOE we can guarantee the accurate measurement of all resistivity sites, which should lead to better models. For robustness, one might want to pick a reasonable cutoff for the resistivity slope coefficients, such that a coefficient will be set to zero if its magnitude is below the cutoff.

Figure 7-48: Resistivity Linear Model Coefficients: Full DOE

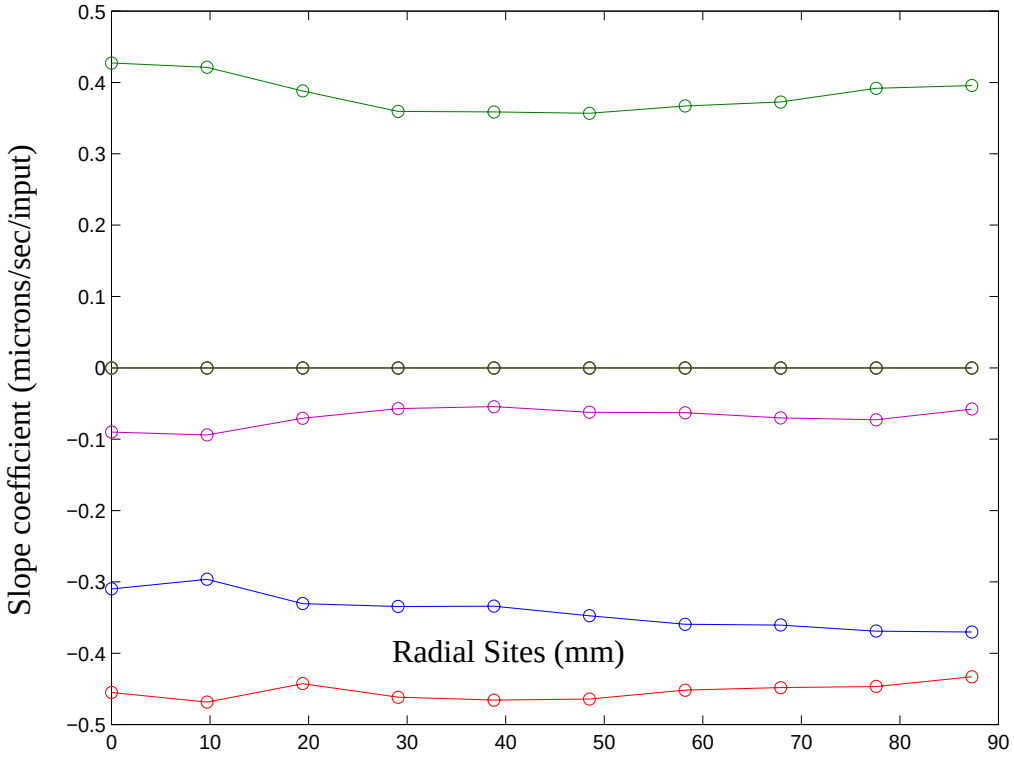
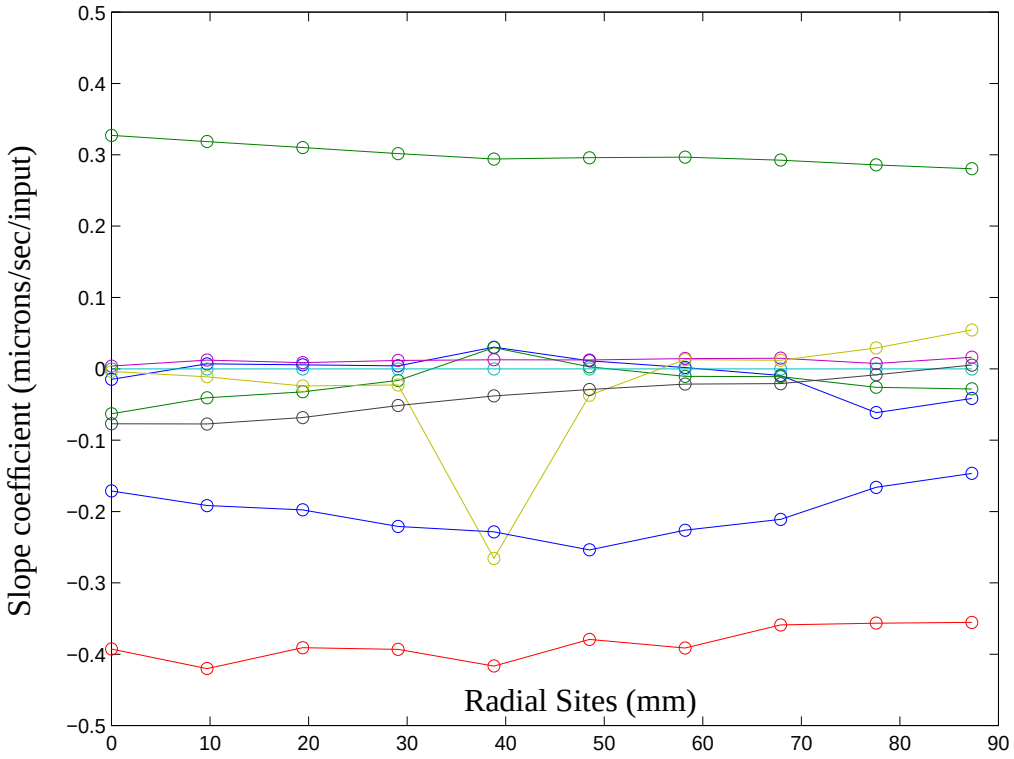


Figure 7-49: Resistivity Linear Model Coefficients: Axial and Center Points



The axial runs account for 18 recipes (2 x 9) in the backsealed configuration and 20 recipes (2 x 10) in the non backsealed configuration, with the use of the auxiliary dopant flow as an additional input. With 25 wafers available for the tuning DOE there are about 5 more runs available. Two or three center points are used, leaving just two or three “spare” runs. There is only a limited amount of additional information that can be provided by two runs. As an *ad-hoc* selection method, two factorial design points are chosen such that the models from the original DOE predict the minimum and maximum output levels. For the maximum output case, this implies that an input level of +1 is used for inputs with positive slopes and levels of -1 are used for inputs with negative slopes. The minimum output case uses just the opposite combinations of +/-1, meaning that the two design points are at a maximal distance from each other. These runs will provide some information about the linearity of the system. Table 7-12 shows the combination of inputs used for the minimum and maximum output design points.

Table 7-12: Factorial Inputs Yielding the Minimum and Maximum Outputs

Factor	Minimum Output	Maximum Output
Deposit temperature (^{o}C)	-1	1
Dopant mixing ratio (%)	1	-1
Dopant main flow (sccm)	1	-1
% Lower power	1	-1
% Inner power	1	-1
Dilutant (H_2) flow (slm)	1	-1
Trichlorosilane (TCS) flow (slm)	-1	1
Center gas flow valve	-1	1
Outer gas flow valve	1	-1
Auxiliary dopant flow (sccm) (for non backsealed wafers)	1	-1

While this simple idea makes intuitive sense, trade-offs must be considered for certain inputs. Specifically, there are inputs that apply positive changes (slopes) for some outputs and negative changes (slopes) for other outputs. One example is the inner gas flow valve, which has positive slopes for deposition rates near the center of the wafer and negative slopes for deposition rates near the edge. Simply adding the slope terms for outputs of the same type (grouping deposition rates and resistivities) is one way to eliminate this problem. However, the slope terms between two different types of outputs cannot be compared directly. If there is a case where a given input has an overall positive influence on deposition rate and a negative influence on resistivity (or vice-versa), then they must be compared to select an appropriate input level. There is no single "right" answer for handling these cases, although a sensible approach is to select the input (+/- 1) that maximizes the combined changes in outputs as percentages of the total available ranges for each type of output. Additionally, one could include a weighting factor to preferentially select inputs that maximally change outputs of a particular type. Depending upon the structure of the slope matrix, it might also be more appropriate to maximize some types of outputs while minimizing other types. The selection process certainly leaves room for heuristics and the intuition of experienced process engineers.

7.2.2 Experimental Execution

Execution of the run-to-run control experiments should make use of automated information and wafer handling wherever possible and demonstrate a clean integration with an existing production line environment. These goals are primarily addressed through integrated Cell Control software that communicates with the sensor(s) and the equipment, and by establishing the operational "mechanics" of the experiments.

Software Integration (Cell Control)

Ideally a flexible distributed modular Cell Controller is used to manage the processing equipment, metrology tools, and data processing software, as described in the Introduction. This type of system requires a well-organized software development strategy and some extra initial effort before any usable tools are developed. On-Line Technologies initially created an integrated software package to control their FTIR epi film thickness sensor. This software does not have any programmable interfaces to enable multi-process communication and control, so it was expanded into a monolithic application that performs all aspects of Cell Control (e.g. this single application manages the processing equipment, the sensor, and the control algorithms). As described in Chapter 6, the time-based run-to-run control algorithms have been packaged into Microsoft COM-based Dynamic Link Libraries (DLL's) for modular use in a wide variety of development environments. These DLL's are called upon by the monolithic Cell Control system.

These "big picture" software integration issues are mainly concerned with Cell Control architectures. However, regardless of the implementation infrastructure, a number of important details must be addressed, including the following:

- Providing sensor data to the controller.
- Defining how a controller's suggested inputs map into an actual equipment recipe.
- Coordinating recipe updates with automated wafer handling and processing.

Sensor data must be acquired and processed before they can be presented to the run-to-run controller. For off-line thickness and resistivity sensors, this means that a manual system is required for retrieving sensor data and injecting those measurements into the system. Ideally the off-line sensors are on a network and the data can be retrieved and

manipulated automatically, but this is currently not a feasible scenario. For an initial run-to-run control demonstration, off-line measurement data will be injected into the system using simple text files. While this process allows us to preprocess the off-line data appropriately before the system operates on them, the in-line thickness data have the opposite problem. These data are available automatically through software, but the Cell Controller is required to preprocess the data internally before presenting measurements to the run-to-run controller. For the epi deposition control scenario, in-line thickness measurements must be radially averaged first, then the full set of virtual outputs (see “Joint Sensor Monitoring” in Section 4.3.3) must be generated before the controller can use them. The Cell Control software must handle these data streams appropriately.

In addition to properly feeding data into a run-to-run controller, the recipe suggestions that it returns must be handled correctly. First, the Cell Controller must map the run-to-run controller’s suggested inputs into actual equipment recipe updates. The run-to-run controller’s internal model uses process parameters such as electrical power ratios and gas flows. To create a usable recipe however, these parameters often must be copied into multiple locations within a single recipe. In some cases, a recipe parameter is calculated in terms of a function of the run-to-run controller’s suggested input vector, rather than just using a copy of one particular value. The Cell Controller is required to extrapolate a control model’s input vector into a fully functional process recipe. For the epi deposition system, the 10 inputs used in the backsealed configuration expand to over 30 individual recipe modifications.

While simply generating valid recipes requires extra data handling, this process is further complicated by synchronization issues. Recipe updates cannot take place randomly,

whenever sensor data (in-line or off-line) become available. It is important not to modify a recipe while it is running on a wafer in the process chamber. A mechanism is required to ensure that wafers are not processed with “merged” recipes, where the first few process steps come from one valid recipe, and the remaining steps come from a different (new) recipe. We require that updates are synchronized with the flow of wafers through the process chamber. Ideally these updates occur in an atomic manner, immediately after one wafer finishes processing and before the next wafer enters the process chamber.

“Mechanics” of the Control Runs

Finally, we must specify the overall flow of wafers and information through the system during the control runs, essentially defining the “mechanics” of the experiments. The available system is a single Applied Materials Centura tool with multiple epi deposition chambers and a single cool down chamber. As described earlier, two separate control runs are planned, one with backsealed wafers and one with non backsealed wafers. There are at least three possible modes of operation for performing these distinct control runs, serial, parallel, and “interleaved.”

In serial mode, each of the two control runs are run in order, first all of the backsealed wafers are processed, followed by all of the non backsealed wafers. This operating procedure is fairly inefficient since there are multiple process chambers available, but only one of them is used. However, this is the easiest way to run the experiments. Only the merged run-to-run and lot-to-lot synchronization issues discussed at the end of the “joint sensor” monitoring discussion (Section 4.3.3) are of special concern. Feedback from the in-line sensor is available after every run, so process model and recipe updates occur regularly and in order. Synchronization of recipe updates with the wafer flow (as described above) is

likely to create a one wafer lag between sensor feedback and recipe updates. However, synchronization of model and recipe updates with off-line lot-to-lot feedback data is more difficult. These data are not available until a cassette of wafers is removed from the system and taken to the stand-alone thickness and resistivity measurement tools. This process takes a fair amount of time; if the flow of wafers through the process chamber continues, then there will be out of order measurement feedback to the run-to-run controller when off-line measurements are retrieved. A fairly complex Cell Control system is required to properly perform run-to-run control in this environment, where lot-to-lot data arrive after many subsequent run-to-run control updates have already taken place. To avoid this issue, a simplified testbed system may require the “in order” feedback of data. That is, the lot-to-lot data must be fed back into the system before “normal” run-to-run wafer processing and in-line control can continue. For the serial mode processing scenario, this means that the tool will remain idle while a cassette is removed and the off-line measurements are taken. This added limitation makes serial mode operation even more inefficient.

Parallel mode operation essentially runs multiple serial mode systems across the process chambers. While this sounds like a simple and effective extension, its actual implementation adds considerable complexity to the system. The parallel mode system is required to keep separate run-to-run controllers, each with a distinct model for a given process chamber. Also, the Centura sends all wafers through the same cool down chamber after the deposition process, which is where the in-line sensor readings are taken. The Cell Controller must track the movement of each wafer to ensure that measurements are fed back to the proper run-to-run controller. However, the most severe limitation is our use of the “dopant ratio” recipe setting. This parameter is set on a “per tool” basis for each

dopant source gas, meaning that changes in the recipe for one process chamber will affect all of the other chambers. Changes cannot be made while another process is running on any chamber, making parallel mode processing difficult to implement directly.

Finally, in an attempt to combine some of the best features from both serial and parallel processing, we introduce an “interleaved” mode. Ideally the simplicity of the serial mode can be merged with the added throughput of the parallel mode. To achieve this end, consider using two process chambers, one for each type of control run. However, instead of running them simultaneously, one chamber at a time is used to process a cassette of wafers. When one cassette is done, the system starts processing the next cassette in the other chamber. The strategy runs as follows:

1. Start one cassette of backsealed wafers (Use automated run-to-run control)

   When the cassette is done:
2. Start one cassette of non-backsealed wafers (Use automated run-to-run control)
3. Remove the completed backsealed wafers for off-line measurements of thickness and resistivity (measure the last wafer)

   When the cassette of non-backsealed wafers is done:
4. Update the model for the backsealed wafers with the off-line data
5. Start one cassette of backsealed wafers (Use automated run-to-run control)
6. Remove the completed non-backsealed wafers for off-line measurements of thickness and resistivity (measure the last wafer)

   When the cassette of backsealed wafers is done:
7. Update the model for the non-backsealed wafers with the off-line data
8. Loop back to step 2.

This strategy maximizes the (single chamber at a time) throughput of the system; the

equipment does not have to sit idle while off-line measurements are being taken. There is parallel operation of process equipment and off-line measurements, but the system is simplified because different control runs are not allowed to interact simultaneously on the equipment. Also, the strict ordering of both in-line and off-line measurements is maintained for each chamber.

However, the delay times between processing each cassette act as disturbances to the system, which would be a concern for a true manufacturing environment. However, for the control run experiments we expect the controller to demonstrate its capabilities by rapidly compensating for any such disturbances.

In addition to lot-to-lot delay-induced disturbances, we expect to see larger disturbances from “start-up” effects at the beginning of each day. Further, as with the SISO control experiments described in Chapter 3, run-to-run (and lot-to-lot) control should be demonstrated by purposefully injecting known disturbances into the system. Ideally the disturbances can be applied directly to the outputs. This was accomplished in the SISO experiments by modifying the deposition temperature. Since the controller did not use this input it was forced to compensate for the resulting output disturbance by changing the deposition time. However, for the MIMO control runs, the controller already uses all of the inputs that are known to significantly affect the deposited epi film thickness and resistivity. For this case we expect to add offsets to some of the input parameters. The input offsets cause disturbances in the outputs for which the controller must attempt to compensate.

7.3 Summary

Detailed DOE, modeling, and control strategies have been defined to prepare for dem-

onstrating multi-objective uniformity control. With the help of Applied Materials process engineers, an extensive DOE was specified and executed for characterizing the non back-sealed epi deposition process. The data demonstrate some noise that a controller must be robust against, and indicate that there are process disturbances and drifts, which a controller must be able to compensate for. Further analysis shows that the process is modeled well as a linear function of the inputs, for the region of input space that we considered.

Process optimization details have been presented, including selection of appropriate output weighting terms. This optimization strategy was combined with linear response surface models for multiple thickness and resistivity sites to simulate a set of control runs. The simulation results indicate that the "joint sensor" techniques for merging run-to-run and lot-to-lot data will work well, and that good uniformity control for both types of outputs is achievable.

Finally, practical issues for demonstrating run-to-run control in an industrial environment were presented. These include the use of small DOE's for verifying and tuning process models for a specific piece of equipment, as well as synchronization requirements for feeding data into a run-to-run controller and feeding back the suggested recipe modifications. The scenario has been completely defined and we will soon have the opportunity to test the system at a wafer manufacturing facility.

Chapter 8

Summary and Conclusions

Semiconductor fabrication requires a large number of complex interrelated processing steps. The continuous drive toward greater yield, higher throughput, smaller circuit dimensions, and larger wafers demands a distributed hierarchical decision-making system that can manipulate everything from manufacturing production targets to mass flow controller setpoints. These complicated systems are broken down into smaller units that essentially operate independently at many different levels. While advanced decision-making techniques are widely applicable throughout the hierarchy of a semiconductor manufacturing facility, the expanded use of Equipment Cell Control, and run-to-run control in particular, provides a great opportunity for improving product quality and equipment efficiency.

An Equipment Cell is composed of an individual piece of fabrication equipment and any sensors or software that evaluate or affect the performance of that equipment. The increased availability of in-situ (or in-line) sensors and high speed networking and data processing make Cell Control an effective technology for development and deployment in the semiconductor industry. In particular, the use of many spatially resolved measurements of multiple characteristics provides the opportunity for aggressive multi-objective run-to-run uniformity control. Measurements from multiple in-line and off-line sensors can be utilized with extensions to current Cell Control and run-to-run control technologies.

This work provides theory and practical implementations for testing the effectiveness of integrated Cell Control, specifically in the context of using run-to-run uniformity con-

trol. Section 8.1 summarizes the contributions of this thesis, while Section 8.2 looks forward to future work that can extend the results found here. Finally, Section 8.3 succinctly states the major conclusions that are drawn from this work.

8.1 Contributions

A Cell Control testbed involving epitaxial silicon (epi) deposition using an Applied Materials Centura reactor, an in-line epi film thickness sensor, and off-line film thickness and resistivity sensors, has been introduced, analyzed and experimentally tested. Our goal for this work was to maintain the deposition of epi films with uniform thicknesses and resistivities using multiple site measurements of both characteristics, in the face of process noise, drift and disturbances. While the detailed methodologies and analyses contained in this thesis reference the epi deposition testbed specifically, they are broadly applicable to many processes with similar characteristics.

Effective integrated run-to-run process control is preceded by good model-building strategies, model optimization and update methods, and software integration technologies. Creating and running an appropriate Design of Experiments (DOE) is a crucial first step, as good control depends on having good process models, which in turn require good data for fitting. Selecting process inputs and input levels is an art that requires practical and engineering background with the system in question. We were fortunate to have process engineers from Applied Materials and Wacker Siltronic working on this project.

Control strategies using linear models and Exponentially Weighted Moving Average (EWMA) model updates have been extensively tested and analyzed in previous work. However, this thesis extends the commonly used EWMA-based control scenario with aggressive use of site models, strategies for combining the asynchronous feedback of cor-

related in-line and off-line sensor measurements, and an extensive coverage of solving techniques for nonlinear time-based models, which are widely applicable for etch/polish and deposition processes.

Modular distributed object technologies have been used to specify interfaces for model-based control using distributed object technology. Interface inheritance and containment provide elegant mechanisms for software to programmatically interact with controllers and their underlying models. A Microsoft COM-based solution has been developed and integrated into the Cell Control software for this work.

Extensive experimental testing of a commercial epi deposition system was undertaken. A simple single-input-single-output (SISO) control system showed that a center point measurement of thickness could be effectively controlled by modeling deposition rate and updating deposition time. These experiments verify the basic integrated control system and support our use of time-based models. Next, a full-blown multi-input-multi-output (MIMO) DOE was specified and executed to capture the effects of all inputs that are normally used to tune the epi deposition process. A detailed analysis of the resulting measurements indicates that, within the bounds of the DOE input parameters, the epi deposition process can be effectively modeled and controlled with linear models of deposition rates and resistivities. Finally, these results have been used to structure an actual industrial test of the fully integrated MIMO controller with combined lot-to-lot and run-to-run feedback loops. Execution of these experiments should take place soon, and the results will be reported in future documents.

This work has provided the necessary context, motivation, and theory for utilizing run-to-run control in a semiconductor manufacturing environment. Run-to-run control has

been successfully demonstrated with a number of semiconductor processes, such as chemical mechanical polishing (CMP), plasma etch, and epitaxial silicon deposition. There are a number of issues that complicate the control of these types of processes.

- The amount of removed or deposited material is usually best modeled as the product of a processing rate and a processing time, which leads to nonlinear models.
- There may be characteristics other than thickness that should be controlled to a target value simultaneously.
- There may be multiple measurements of the same (or correlated) outputs with differing sampling rates in time and/or space. Further, some data are likely to arrive as out-of-order feedback.
- The uniformity of the measured outputs should be controlled, not just a single measurement or average of multiple measurements.

This thesis provides theory for overcoming these complications, as well as practical demonstrations for integrating this type of feedback control into a manufacturing environment.

8.2 Future Work

Scattered throughout this thesis are a number of references to ideas and experiments that should be further developed, many of which are extensions beyond the scope of this work. However, the next step is clearly the demonstration and analysis of the run-to-run uniformity control experiments. Unfortunately they could not be completed in time for inclusion in this thesis. In addition to a staged demonstration of multi-objective control, use of the controller for continuous operation in a true manufacturing environment should be pursued. Specifically, we would like to see results over very long run lengths (days, weeks, months, etc.) in a fabrication line that makes revenue generating product. After all, the ultimate goal of this work is to increase manufacturing efficiency through the use of an integrated Equipment Cell.

This basic Cell Control system could also be extended in many other ways. Application of this type of time-based uniformity control for other deposition, etch, or polish processes should be straightforward and effective. The testbed Cell Control system described here is also missing a number of important components. Further work is needed to specify interfaces and implement modules (objects) for integrating Design of Experiments (DOE), fault detection, and diagnosis capabilities. Many of these modules, including the run-to-run controller, might also want to make use of real-time trace data from the system, which were not utilized in this thesis.

Beyond that, a number of theoretical questions have been left open. In particular, a more detailed (and general) theoretical exploration of mixed feedback loops and "joint sensor" techniques could be undertaken. Also, "better" time-based solvers could be found. Perhaps a single step solution, or at least an iterative solver with faster convergence, is possible for solving the general overdetermined case.

8.3 Conclusions

Integrated Cell Control provides a great opportunity for increasing semiconductor manufacturing efficiency. In particular, model-based run-to-run uniformity control using proper process modeling and optimization strategies can generate significantly better product using fewer resources. Time-based modeling and control is appropriate for many deposition and etch/polish processes, including the growth of thin epitaxial silicon films. Uniformity control is effective on systems with sensors and control knobs that can detect and manipulate process uniformity, and these capabilities are becoming increasingly available as wafer diameters expand to 300 millimeters and beyond. Finally, the use of modular, distributed software integration techniques and tools are important for system

flexibility and reusability.

References

[AM95] Applied Materials, Inc., *Epi Centura Process Manual*, Sept. 1995.

[Asp97] D. Aspnes, "Real-Time Optical Analysis and Control of Semiconductor Epitaxy: Progress and Opportunity," *Solid State Communications*, Vol. 101, No. 2, pp. 85-92, 1997.

[AZ97] S. Adivikolanu and E. Zafiriou, "Internal Model Control Approach to Run-to-Run Control for Semiconductor Manufacturing," *Proceedings of the American Control Conference*, pp. 145-149, June 1997.

[AZ98] S. Adivikolanu and E. Zafiriou, "Robust Run-to-Run Control for Semiconductor Manufacturing: An Internal Model Control Approach," *Proceedings of the American Control Conference*, pp. 3687-3691, June 1998.

[BCDH94] R. Beaver, A. Coleman, D. Draheim, and A. Hoffman, "Architecture and Overview of MMST Machine Control," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 7, No. 2, pp. 127-133, May 1994.

[BMPS92] D. Boning, M. McIlrath, P. Penfield, and E. Sachs, "A General Semiconductor Process Modeling Framework," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 5, No. 4, pp. 266-280, Nov. 1992.

[BMSM95] D. Boning, W. Moyne, T. Smith, J. Moyne, A. Hurwitz, "Practical issues in run by run process control," Advanced Semiconductor Manufacturing Conference and Workshop, 1995. ASMC 95 Proceedings. IEEE/SEMI, pp. 201 -208, Nov. 1995.

[BP95] J. Baras and N. Patel, "Designing Response Surface Model Based Run-by-Run Controllers: A New Approach," in Proc. *IEEE/CPMT Int. Elec. Manuf. Tech. (IEMT) Symp.*, pp. 210-217, 1995.

[BP97] J. Baras and N. Patel, "A Framework for Robust Run by Run Control with Lot Delayed Measurements," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 10, No. 1, pp.75-83, Feb. 1997.

[BS01] D. Barrett and R. Silverman, *SSH, the Secure Shell: The Definitive Guide*, O'Reilly & Associates, 2001.

[Cha98] S. Charpenay, P. Rosenthal, G. Kneissl, C. Gondran, H. Huff, “Model-based analysis for precise and accurate epitaxial silicon measurements,” *Solid State Technology*, July 1998.

[Che95] A. Cherkassky, *Metrology of thin silicon expitaxial films: determination of epitaxial film thickness by Fourier-transform infra-red spectrometry*, M.S. Thesis, MIT EECS, 1995.

[CMR98] N. Chaudhry, J. Moyne, and E. Rundensteiner, “Active Controller: Utilizing Active Databases for Implementing Multistep Control of Semiconductor Manufacturing,” *IEEE Transactions on Components, Packaging, and Manufacturing Technology - Part C*, Vol. 21, No. 3, pp. 217-224, July 1998.

[DD99] R. DeKeyser and J. Donald III, “Model Based Predictive Control in RTP Semiconductor Manufacturing,” *Proceedings of the 1999 IEEE International Conference on Control Applications*, pp.1636-1641, 1999.

[DGLH96] J. Davis, R. Gyurcsik, J. Lu, and J. Hughes-Oliver, “A Robust Metric for Measuring Within-Wafer Uniformity,” *IEEE Trans. on Comp., Packaging, and Manuf. Technology*, Vol. 19, No. 4, pp. 283-289, Oct. 1996.

[DS81] N. Draper and H. Smith, *Applied Regression Analysis, Second Edition*, John Wiley & Sons, Inc, 1981.

[ESK96] K. El-Awady, C. Schaper, T. Kailath, “Improvemnts in C_{pk} Using Real-Time Feedback Control,” *IEEE Transactions on Semiconductor Manufacturing*, Vol. 9, No. 1, pp. 87-94, Feb. 1996.

[Fen00] J. Fenner, *Multi-Systems Operation and Control*, Ph.D. Thesis, NC State University Statistics, 2000.

[Gow96] A. Gower, *An Architecture for Flexible Distributed Experimentation and Control with an AME 5000 Plasma Etcher*, S.M. Thesis, MIT EECS, Aug. 1996.

[Gro99] J. Groce, “Advanced Process Control Framework Initiative (APCFI) Project: Detailed System Description,” SEMATECH Technology Transfer #90053736A-TR, June 1999.

[GS93] R. Guo and E. Sachs, “Modeling, Optimization and Control of Spatial Uni-

formity in Manufacturing Processes," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 6, No. 1, pp. 41-57, Feb. 1993.

[Ha93] S. Ha, *On-Line Control of Process Uniformity Using Categorized Variabilities*, Ph. D. Thesis, MIT Mechanical Engineering, June 1993.

[HKK98] E. Hamby, T. Kabamba, and P. Khargonekar, "A Probabilistic Approach to Run-to-Run Control," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 11, No. 4, pp. 654-669, Nov. 1998.

[HVIK97] M. Hankinson, T. Vincent, K. Irani, P. Khargonekar, "Integrated Real-Time and Run-to-Run Control of Etch Depth in Reactive Ion Etching," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 10, No. 1, pp. 121-130, Feb. 1997.

[Joh98] B. Johs, C. Herzinger, J. Dinan, A. Cornfeld, J. Benson, D. Doctor, G. Olson, I. Ferguson, M. Pelczynski, P. Chow, C. Kuo, S. Johnson, "Real-time monitoring and control of epitaxial semiconductor growth in a production environment by in situ spectroscopic ellipsometry," *Thin Solid Films*, 313-314, pp. 490-495, 1998.

[KGCK97] T. Knight, D. Greve, X. Cheng, B. Krogh, "Real-Time Multivariable Control of PECVD Silicon Nitride Film Properties," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 10, No. 1, pp. 137-146, Feb. 1997.

[Lea96] S. Leang, S. Ma, J. Thomson, B. Bombay, and C. Spanos, "A Control System for Photolithographic Sequences," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 9, No. 2, pp. 191-207, Feb. 1996.

[Lea97] R. Leachman, "Closed-Loop Measurement of Equipment Efficiency and Equipment Capacity," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 10, No. 1, pp. 84-97, Feb. 1997.

[Lee95] S. Lee, E. Boskin, H. Liu, E. Wen, and C. Spanos, "RTSPC: A Software Utility for Real-Time SPC and Tool Data Analysis," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 8, No. 1, pp. 17-25, Feb. 1995.

[LH96] Y. Hung and R. Leachman, "A Production Planning Methodology for Semiconductor Manufacturing Based on Iterative Simulation and Linear Programming calculations," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 9, No. 2, pp. 257-269, May 1996.

[LS95] S. Lee and C. Spanos, “Prediction of Wafer State After Plasma Processing Using Real-Time Tool Data,” *IEEE Transactions on Semiconductor Manufacturing*, Vol. 8, No. 3, pp. 252-261, Aug. 1995.

[LSBH97] M. Le, T. Smith, D. Boning, and H. Sawin, “Run to Run Model Based Process Control on a Dual Coil Transformer Coupled Plasma Etcher,” *191st Meeting of the Electrochemical Society*, pp. 332, May 1997.

[Mul97] T. Mullins, “Advanced Process Control Framework Initiative (APCFI) 1.0 Specifications,” SEMATECH Technology Transfer #97063300A-ENG, June 1997.

[Mat97] The Mathworks, Inc., *Matlab (The Language of Technical Computing), Using MATLAB Version 5*, 1997.

[McD00] D. McDysan, *VPN Applications Guide: Real Solutions for Enterprise Networks*, John Wiley & Sons, Inc, 2000.

[McI92] M. McIlrath, D. Troxel, M. Heytens, P. Penfield, D. Boning, and R. Jayavant, “CAFE - The MIT Computer-Aided Fabrication Environment,” *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, Vol. 3, No. 2, pp. 353-360, May 1992.

[Mon91] D. Montgomery, *Design and Analysis of Experiments (Third Edition)*, John Wiley & Sons, Inc, 1991.

[MM92] J. Moyne and L. McAfee, “A Generic Cell Controller for Automated VLSI Manufacturing Facility,” *IEEE Transactions on Semiconductor Manufacturing*, Vol. 5, No. 2, pp. 77-87, May 1992.

[Moy95] W. Moyne, *Run-by-Run Control: Interfaces, Implementation, and Integration*, S. M. Thesis, MIT EECS, Feb. 1995.

[OH98] R. Orfali and D. Harkey, *Client/Server Programming with JAVA and CORBA (Second Edition)*, John Wiley & Sons, Inc, 1998.

[Ras95] B. Rashap, M. Elta, H. Etemad, J. Fournier, J. Freudenberg, M. Giles, J. Grizzle, P. Kabamba, P. Khargonekar, S. Lafortune, J. Moyne, D. Teneketzis, and F. Terry, Jr., “Control of Semiconductor Manufacturing Equipment: Real-Time Feedback Control of a Reactive Ion Etcher,” *IEEE Transactions on Semiconductor Manufacturing*, Vol. 8, No. 3, pp. 286-297,

Aug. 1995.

[Red97] F. Redmond, *DCOM: Microsoft Distributed Component Object Model*, IDG Books Worldwide, Inc, 1997.

[Res00] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison Wesley Professional, 2000.

[Ros98] P. Rosenthal, P. R. Solomon, S. Charpenay, V. Yakovlev, W. Zhang, A. Bonanno, M. Spartz, and J. Xu, A. Gower, T. Smith, and D. Boning, A. Waldhauer, and W. Aarts, "Infrared Spectroscopy for Control and Fault Detection of Advanced Semiconductor Processes," *1998 International Conf. on Characterization and Metrology for ULSI Technology*, Gaithersburg, MD, March 1998.

[RP95] E. Rietman and S. Patel, A Production Demonstration of Wafer-to-Wafer Plasma Gate Etch Control by Adaptive Real-Time Computation of the Over-Etch Time from *in Situ* Process Signals," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 8, No. 3, pp. 304-308, Aug. 1995.

[RWFG99] S. Ruegsegger, A. Wagner, J. Freudenberg, and D. Grimard, "Feedforward Control for Reduced Run-to-Run Variation in Microelectronics Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 12, No. 4, pp. 493-502, Nov. 1999.

[SAS95] SAS Institute, Inc., *JMP Introductory Guide, Version 3*, 1995.

[SB97] T. Smith and D. Boning, "Artificial Neural Network Exponentially Weighted Moving Average Controller for Semiconductor Processes," *J. Vac. Sci. Technol. A*, Vol. 15, No. 3, pp. 1377-1384, May 1997.

[SBG94] N. Srivatsan, S. Bai, and S. Gershwin, "Hierarchical Real-Time Integrated Scheduling of a Semiconductor Fabrication Facility," *Control and Dynamic Systems: Advances in Theory and Applications*, Vol 61, pp. 197-241.

[SBHW94] M. Sullivan, S. Butler, J. Hirsch and C. Wang, "A Control-to-Target Architecture for Process Control," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 7, No. 2, pp. 134-148, May 1994.

[SCKT94] K. Stoddard, P. Crouch, M. Kozicki, and K. Tsakalis, "Application of Feed-Forward and Adaptive Feedback Control to Semiconductor Device

Manufacturing," *Proceedings of the American Control Conference*, pp. 892-896, Vol. 1, June 1994.

[SEMI1] Semiconductor Equipment and Materials International, *SEMI Equipment Communications Standard 1 Message Transfer (SECS-I)*, SEMI E4-0699.

[SEMI2] Semiconductor Equipment and Materials International, *SEMI Equipment Communications Standard 2 Message Content (SECS-II)*, SEMI E5-0701.

[SGBS99] T. Smith, B. Goodlin, D. Boning, and H. Sawin, "A Statistical Analysis of Single and Multiple Response Surface Modeling," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 12, No. 4, pp. 419-430, Nov. 1999.

[SGHH91 E. Sachs, R. Guo, S. Ha, and A. Hu, "Process Control System for VLSI Fabrication," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 4, No. 2, pp. 134-144, May 1991.

[SHI95] E. Sachs, A. Hu and A. Ingolfsson, "Run by Run Process Control: Combining SPC and Feedback Control," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 8, No. 1, pp. 26-43, Feb. 1995.

[Smi96] T. Smith, Novel Techniques for the Run by Run Process Control of Chemical-Mechanical Polishing, S.M. Thesis, MIT EECS, Feb. 1996.

[Smi99] T. Smith, Device Independent Process Control of Dielectric Chemical Mechanical Polishing, Ph.D. Thesis, MIT EECS, Sept. 1999.

[SmiB96] T. Smith, D. Boning, J. Moyne, A. Hurwitz, and J. Curry, "Compensating for CMP Pad Wear Using Run by Run Feedback Control," VMIC, pp. 437-440, Santa Clara, CA, June 1996.

[TZA99] A. Theodoropoulou, E. Zafiriou, and R. Adomaitis, "Inverse Model-Based Real-Time Control For Temperature Uniformity of RTCVD," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 12, No. 1, pp. 87-101, Feb. 1999.

[WD98] S. Warnick and M. Dahleh, "Feedback Control of MOCVD Growth of Submicron Compound Semiconductor Films," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 6, No. 1, pp. 62-71, Jan. 1998.

[Whi95] D. White, *In-Situ Wafer Uniformity Estimation using Principal Component Analysis Methods*, S.M. Thesis, MIT EECS, May 1995.

[Won96] K. Wong, *Real-Time Analysis and Control of Plasma Etching via Full Wafer Interferometry*, Ph.D. Thesis, MIT EECS, Sep. 1996.

[WT86] S. Wolf and R. Tauber, *Silicon Processing for the VLSI Era (Volume 1 - Process Technology*), Lattice Press, 1986.

[Yeb94] S. Yeboah-Amankwah, *Epitaxial Silicon Growth Rate Control*, M. Eng. Thesis, MIT EECS, May 1994.

[ZR95] Z. Zhou and R. Reif, “Epi-Film Thickness Measurements Using Emission Fourier Transform Infrared Spectroscopy - Part II: Real-Time *in Situ* Process Monitoring and Control,” *IEEE Transactions on Semiconductor Manufacturing*, Vol. 8, No. 3, pp. 340-345, Aug. 1995.

Appendix A

COM IDL For the Controller Objects

Interface Definition Language (IDL) files are written to provide Microsoft COM object interfaces for the model and controller objects introduced in Chapter 6. Please refer to Chapter 6 and Appendix B for design strategies, descriptions of interface parameters, and implementation details. The interfaces are hierarchical in nature and build upon each other through inheritance and aggregation. File names beginning with the letter “I” include pure interface definitions, while those beginning with the letter “C” are used to generate “libraries” with “coclasses” that can be compiled and implemented. A good familiarity with Microsoft COM IDL is required to fully understand these files and their use, but a familiarity with CORBA or other software interfacing technology should provide enough background to understand their structure.

Imit_Base.idl

```
#ifndef _L__Imit_Base_idl_INCLUDED
#define _L__Imit_Base_idl_INCLUDED

import “oaidl.idl”;
import “ocidl.idl”;

[
    object, version(1.0),
    uuid(0B715C63-226B-11d3-A942-00C06D13817E),
    dual,
    helpstring(“Imit_IObject Interface”),
    pointer_default(unique)
]
interface Imit_IObject : IDispatch
{
    [id(10), helpstring(“method getId”)] HRESULT getId(
        [out,retval] BSTR* objId);
    [id(11), helpstring(“method setId”)] HRESULT setId(
        [in] BSTR objId);

    [id(12), helpstring(“method connectedTo”)] HRESULT connectedTo(
        [in,out] SAFEARRAY(BSTR)* id,
        [in,out]SAFEARRAY(IUnknown*)* pInterface,
        [out] long* numConnections);
    [id(13), helpstring(“method callbacksTo”)] HRESULT callbacksTo(
        [in,out] SAFEARRAY(BSTR)* id,
        [in,out] SAFEARRAY(IUnknown*)* pInterface,
        [out] long* numConnections);

    [id(14), helpstring(“method getStorage”)] HRESULT getStorage(
        [out,retval] IUnknown** pStorage);
    [id(15), helpstring(“method setStorage”)] HRESULT setStorage(
        [in] IUnknown* pStorage);

    [id(16), helpstring(“method store”)]   HRESULT store();
    [id(17), helpstring(“method restore”)] HRESULT restore(
        [in] BSTR objId);
}

#endif /* _L__Imit_Base_idl_INCLUDED */
```

Imit_r2r_IModel.idl

```
#ifndef _L__Imit_r2r_IModel_idl_INCLUDED
#define _L__Imit_r2r_IModel_idl_INCLUDED


import “Imit_Base.idl”;

typedef enum mit_r2r_EDiscFlag
{
    NOT_INITIALIZED = 0,
    NO_DISC = 1,
    SIMPLE_ROUNDING = 2,
    DELTA_DISC = 3,
    STAT_DISC = 4,
    STAT_DISC_2 = 5
} mit_r2r_EDiscFlag;

[
    object, version(1.0),
    uuid(67c6f433-f7f5-2921-1d67-948cbb1c51e7),
    dual,
    helpstring(“Imit_r2r_IModel Interface”),
    pointer_default(unique)
]
interface Imit_r2r_IModel : Imit_IObject
{
    [id(100), helpstring(“method getModel”)] HRESULT getModel(
        [out,retval] Imit_r2r_IModel** model);

    /*
    [id(101)] getModelState();
    [id(103)] setModelState();
    */

    [id(110), helpstring(“method fit”)] HRESULT fit(
        [in] SAFEARRAY(double) runInputs,
        [in] SAFEARRAY(long) runNums,
        [in] SAFEARRAY(double) runEquipTimes,
        [in] SAFEARRAY(double) runOutputs);

    [id(111), helpstring(“method func”)] HRESULT func(
        [in] SAFEARRAY(double) runInput,
        [in] long runNum,
        [in] double runEquipTime,
        [in,out] SAFEARRAY(double)* runOutput,
        [out] long* runOutputLen);

    [id(112), helpstring(“method getConfigs”)] HRESULT getConfigs(
        [in,out] SAFEARRAY(double)* configs,
        [out] long* configsLen);

    [id(113), helpstring(“method getEquipTime”)] HRESULT getEquipTime(
        [out,retval] double* equipTime);

    [id(114), helpstring(“method getInput”)] HRESULT getInput(
        [in,out] SAFEARRAY(double)* input,
        [out] long* inputLen);
```

```
[id(115), helpstring(“method getInputNames”)] HRESULT getInputNames(
    [in,out] SAFEARRAY(BSTR)* inputNames,
    [out] long* inputNamesLen);

[id(116), helpstring(“method getModelName”)] HRESULT getModelName(
    [out,retval] BSTR* modelName);

[id(117), helpstring(“method getNumInputs”)] HRESULT getNumInputs(
    [out,retval] long* numInputs);

[id(118), helpstring(“method getNumOutputs”)] HRESULT getNumOutputs(
    [out,retval] long* numOutputs);

[id(119), helpstring(“method getOutputNames”)] HRESULT getOutputNames(
    [in,out] SAFEARRAY(BSTR)* outputNames,
    [out] long* outputNamesLen);

[id(120), helpstring(“method getRunNum”)] HRESULT getRunNum(
    [out,retval] long* runNum);



// Solution Constraints

[id(121), helpstring(“method getInputMax”)] HRESULT getInputMax(
    [in,out] SAFEARRAY(double)* inputMax,
    [out] long* inputMaxLen);
[id(122), helpstring(“method getInputMin”)] HRESULT getInputMin(
    [in,out] SAFEARRAY(double)* inputMin,
    [out] long* inputMinLen);
[id(123), helpstring(“method getResolution”)] HRESULT getResolution(
    [in,out] SAFEARRAY(double)* resolution,
    [out] long* resolutionLen);
[id(124), helpstring(“method getWeightInput”)] HRESULT getWeightInput(
    [in,out] SAFEARRAY(double)* weightInput,
    [out] long* weightInputLen);
[id(125), helpstring(“method getWeightOutput”)] HRESULT getWeightOutput(
    [in,out] SAFEARRAY(double)* weightOutput,
    [out] long* weightOutputLen);
[id(126), helpstring(“method getDiscFlag”)] HRESULT getDiscFlag(
    [out,retval] mit_r2r_EDiscFlag* discFlag);



[id(127), helpstring(“method setConfigs”)] HRESULT setConfigs(
    [in] SAFEARRAY(double) configs);

[id(128), helpstring(“method setEquipTime”)] HRESULT setEquipTime(
    [in] double equipTime);

[id(129), helpstring(“method setInput”)] HRESULT setInput(
    [in] SAFEARRAY(double) input);

[id(130), helpstring(“method setInputNames”)] HRESULT setInputNames(
    [in] SAFEARRAY(BSTR) inputNames);

[id(131), helpstring(“method setModelName”)] HRESULT setModelName(
    [in] BSTR modelName);
```

```
    [id(132), helpstring(“method setNumInputs”)] HRESULT setNumInputs(
        [in] long numInputs);

    [id(133), helpstring(“method setNumOutputs”)] HRESULT setNumOutputs(
        [in] long numOutputs);

    [id(134), helpstring(“method setOutputNames”)] HRESULT setOutputNames(
        [in] SAFEARRAY(BSTR) outputNames);

    [id(135), helpstring(“method setRunNum”)] HRESULT setRunNum(
        [in] long runNum);

    // Solution Constraints
    [id(136), helpstring(“method setInputMax”)] HRESULT setInputMax(
        [in] SAFEARRAY(double) inputMax);
    [id(137), helpstring(“method setInputMin”)] HRESULT setInputMin(
        [in] SAFEARRAY(double) inputMin);
    [id(138), helpstring(“method setResolution”)] HRESULT setResolution(
        [in] SAFEARRAY(double) resolution);
    [id(139), helpstring(“method setWeightInput”)] HRESULT setWeightInput(
        [in] SAFEARRAY(double) weightInput);
    [id(140), helpstring(“method setWeightOutput”)] HRESULT setWeightOutput(
        [in] SAFEARRAY(double) weightOutput);
    [id(141), helpstring(“method setDiscFlag”)] HRESULT setDiscFlag(
        [in] mit_r2r_EDiscFlag discFlag);



    [id(142), helpstring(“method solve”)] HRESULT solve(
        [in] SAFEARRAY(double) runInput,
        [in] long nextRunNum,
        [in] double nextEquipTime,
        [in] SAFEARRAY(double) nextTarget,
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);

}

#endif /*
_L__development_cpp_experSCppTest_bindings_midl_Imit_r2r_IModel_idl_INCLUDED */
```

Imit_r2r_ILinModel.idl

```
#ifndef _L__Imit_r2r_ILinModel_idl_INCLUDED
#define _L__Imit_r2r_ILinModel_idl_INCLUDED


import “Imit_r2r_IModel.idl”;


[
    object, version(1.0),
    uuid(45a1d671-6cfc-5306-1d78-019f6a19ee59),
    dual,
    helpstring(“Imit_r2r_ILinModel Interface”),
    pointer_default(unique)
]
interface Imit_r2r_ILinModel : Imit_r2r_IModel
{
    [id(200), helpstring(“method getLinModel”)] HRESULT getLinModel(
        [out,retval] Imit_r2r_ILinModel** linModel);
    /*
    [id(201)] getLinState();
    [id(202)] getLinModelState();
    [id(203)] setLinState();
    [id(204)] setLinModelState();
    */

    [id(210), helpstring(“method getIntercept”)] HRESULT getIntercept(
        [in,out] SAFEARRAY(double)* intercept,
        [out] long* interceptLen);
    [id(211), helpstring(“method getSlope”)] HRESULT getSlope(
        [in,out] SAFEARRAY(double)* slope,
        [out] long* slopeLen,
        [out] long* slopeWid);

    [id(212), helpstring(“method setIntercept”)] HRESULT setIntercept(
        [in] SAFEARRAY(double) intercept);
    [id(213), helpstring(“method setSlope”)] HRESULT setSlope(
        [in] SAFEARRAY(double) slope);
}


#endif /* _L__Imit_r2r_ILinModel_idl_INCLUDED */
```

Imit_r2r_IBiasModel.idl

```
#ifndef _L__Imit_r2r_IBiasModel_idl_INCLUDED
#define _L__Imit_r2r_IBiasModel_idl_INCLUDED

import “Imit_r2r_IModel.idl”;

[
    object, version(1.0),
    uuid(d26515fc-d7c6-a17d-1d52-95e18511d2c9),
    dual,
    helpstring(“Imit_r2r_IBiasModel Interface”),
    pointer_default(unique)
]
interface Imit_r2r_IBiasModel : Imit_r2r_IModel
{
    [id(200), helpstring(“method getBiasModel”)] HRESULT getBiasModel(
        [out,retval] Imit_r2r_IBiasModel** biasModel);
    /*
    [id(201)] getBiasState();
    [id(202)] getBiasModelState();
    [id(203)] setBiasState();
    [id(204)] setBiasModelState();
    */

    [id(210), helpstring(“method getBias”)] HRESULT getBias(
        [in] SAFEARRAY(double) runInput,
        [in] long runNum,
        [in] double runEquipTime,
        [in,out] SAFEARRAY(double)* bias,
        [out] long* biasLen);

    [id(211), helpstring(“method setBias”)] HRESULT setBias(
        [in] SAFEARRAY(double) runInput,
        [in] long runNum,
        [in] double runEquipTime,
        [in] SAFEARRAY(double) bias );
}

#endif /* _L__Imit_r2r_IBiasModel_idl_INCLUDED */
```

Imit_r2r_ITimeModel.idl

```
#ifndef _L__Imit_r2r_ITimeModel_idl_INCLUDED
#define _L__Imit_r2r_ITimeModel_idl_INCLUDED


import “Imit_r2r_IModel.idl”;


[
    object, version(1.0),
    uuid(7e5ec855-15b8-f8a5-1d7b-31490ed33609),
    dual,
    helpstring(“Imit_r2r_ITimeModel Interface”),
    pointer_default(unique)
]
interface Imit_r2r_ITimeModel : Imit_r2r_IModel
{
    [id(200), helpstring(“method getTimeModel”)] HRESULT getTimeModel(
        [out,retval] Imit_r2r_ITimeModel** timeModel);
    /*
    [id(201)] getTimeState();
    [id(202)] getTimeModelState();
    [id(203)] setTimeState();
    [id(204)] setTimeModelState();
    */

    [id(210), helpstring(“method getNumRateOutputs”)] HRESULT getNumRateOutputs(
        [out,retval] long* numRateOutputs);
    [id(211), helpstring(“method getNumTimeInputs”)] HRESULT getNumTimeInputs(
        [out,retval] long* numTimeInputs);
    [id(212), helpstring(“method getPrimaryModel”)] HRESULT getPrimaryModel(
        [out,retval] Imit_r2r_IModel** primaryModel);
    [id(213), helpstring(“method getRateToTimeMap”)] HRESULT getRateToTimeMap(
        [in,out] SAFEARRAY(int)* rateToTimeMap,
        [out] long* rateToTimeMapLen);

    [id(214), helpstring(“method setNumRateOutputs”)] HRESULT setNumRateOutputs(
        [in] long numRateOutputs);
    [id(215), helpstring(“method setNumTimeInputs”)] HRESULT setNumTimeInputs(
        [in] long numTimeInputs);
    [id(216), helpstring(“method setPrimaryModel”)] HRESULT setPrimaryModel(
        [in] Imit_r2r_IModel* primaryModel);
    [id(217), helpstring(“method setRateToTimeMap”)] HRESULT setRateToTimeMap(
        [in] SAFEARRAY(long) rateToTimeMap);

}


#endif /* _L__Imit_r2r_ITimeModel_idl_INCLUDED */
```

Imit_r2r_IBiasLinModel.idl

```
#ifndef _L__Imit_r2r_IBiasLinModel_idl_INCLUDED
#define _L__Imit_r2r_IBiasLinModel_idl_INCLUDED


import “Imit_r2r_ILinModel.idl”;
import “Imit_r2r_IBiasModel.idl”;


[
    object, version(1.0),
    uuid(5842d85f-72bb-3321-1d7e-6f071792b084),
    dual,
    helpstring(“Imit_r2r_IBiasLinModel Interface”),
    pointer_default(unique)
]
interface Imit_r2r_IBiasLinModel : Imit_r2r_ILinModel
{
    // Include IBiasModel methods

    [id(300), helpstring(“method getBiasModel”)] HRESULT getBiasModel(
        [out,retval] Imit_r2r_IBiasModel** biasModel);
    /*
    [id(301)] getBiasState();
    [id(302)] getBiasModelState();
    [id(303)] setBiasState();
    [id(304)] setBiasModelState();
    */

    [id(310), helpstring(“method getBias”)] HRESULT getBias(
        [in] SAFEARRAY(double) runInput,
        [in] long runNum,
        [in] double runEquipTime,
        [in,out] SAFEARRAY(double)* bias,
        [out] long* biasLen);

    [id(311), helpstring(“method setBias”)] HRESULT setBias(
        [in] SAFEARRAY(double) runInput,
        [in] long runNum,
        [in] double runEquipTime,
        [in] SAFEARRAY(double) bias );
}


#endif /* _L__Imit_r2r_IBiasLinModel_idl_INCLUDED */
```

Imit_r2r_ITimeBiasModel.idl

```
#ifndef _L__Imit_r2r_ITimeBiasModel_idl_INCLUDED
#define _L__Imit_r2r_ITimeBiasModel_idl_INCLUDED


import “Imit_r2r_IBiasModel.idl”;
import “Imit_r2r_ITimeModel.idl”;


[
    object, version(1.0),
    uuid(27f858d7-8b5c-9898-1d73-55c620b6a91d),
    dual,
    helpstring(“Imit_r2r_ITimeBiasModel Interface”),
    pointer_default(unique)
]
interface Imit_r2r_ITimeBiasModel : Imit_r2r_IBiasModel
{
    // Include TimeModel methods

    [id(300), helpstring(“method getTimeModel”)] HRESULT getTimeModel(
        [out,retval] Imit_r2r_ITimeModel** timeModel);
    /*
    [id(301)] getTimeState();
    [id(302)] getTimeModelState();
    [id(303)] setTimeState();
    [id(304)] setTimeModelState();
    */

    [id(310), helpstring(“method getNumRateOutputs”)] HRESULT getNumRateOutputs(
        [out,retval] long* numRateOutputs);
    [id(311), helpstring(“method getNumTimeInputs”)] HRESULT getNumTimeInputs(
        [out,retval] long* numTimeInputs);
    [id(312), helpstring(“method getPrimaryModel”)] HRESULT getPrimaryModel(
        [out,retval] Imit_r2r_IModel** primaryModel);
    [id(313), helpstring(“method getRateToTimeMap”)] HRESULT getRateToTimeMap(
        [in,out] SAFEARRAY(int)* rateToTimeMap,
        [out] long* rateToTimeMapLen);

    [id(314), helpstring(“method setNumRateOutputs”)] HRESULT setNumRateOutputs(
        [in] long numRateOutputs);
    [id(315), helpstring(“method setNumTimeInputs”)] HRESULT setNumTimeInputs(
        [in] long numTimeInputs);
    [id(316), helpstring(“method setPrimaryModel”)] HRESULT setPrimaryModel(
        [in] Imit_r2r_IModel* primaryModel);
    [id(317), helpstring(“method setRateToTimeMap”)] HRESULT setRateToTimeMap(
        [in] SAFEARRAY(long) rateToTimeMap);

}


#endif /* _L__Imit_r2r_ITimeBiasModel_idl_INCLUDED */
```

Imit_r2r_ITimeBiasLinModel.idl

```
#ifndef _L__mit_r2r_ITimeBiasLinModel_idl_INCLUDED
#define _L__mit_r2r_ITimeBiasLinModel_idl_INCLUDED



import “Imit_r2r_IBiasLinModel.idl”;
import “Imit_r2r_ITimeModel.idl”;


[
    object, version(0.0),
    uuid(1bb976e8-cf38-7966-1d6d-6507de5d2663),
    pointer_default(unique)
]
interface Imit_r2r_ITimeBiasLinModel : IBiasLinModel
{
    // Include TimeModel methods

    [id(400), helpstring(“method getTimeModel”)] HRESULT getTimeModel(
        [out,retval] Imit_r2r_ITimeModel** timeModel);
    /*
    [id(401)] getTimeState();
    [id(402)] getTimeModelState();
    [id(403)] setTimeState();
    [id(404)] setTimeModelState();
    */

    [id(410),helpstring(“method getNumRateOutputs”)] HRESULT getNumRateOutputs(
        [out,retval] long* numRateOutputs);
    [id(411),helpstring(“method getNumTimeInputs”)] HRESULT getNumTimeInputs(
        [out,retval] long* numTimeInputs);
    [id(412), helpstring(“method getPrimaryModel”)] HRESULT getPrimaryModel(
        [out,retval] Imit_r2r_IModel** primaryModel);
    [id(413), helpstring(“method getRateToTimeMap”)] HRESULT getRateToTimeMap(
        [in,out] SAFEARRAY(int)* rateToTimeMap,
        [out] long* rateToTimeMapLen);

    [id(414),helpstring(“method setNumRateOutputs”)] HRESULT setNumRateOutputs(
        [in] long numRateOutputs);
    [id(415),helpstring(“method setNumTimeInputs”)] HRESULT setNumTimeInputs(
        [in] long numTimeInputs);
    [id(416), helpstring(“method setPrimaryModel”)] HRESULT setPrimaryModel(
        [in] Imit_r2r_IModel* primaryModel);
    [id(417), helpstring(“method setRateToTimeMap”)] HRESULT setRateToTimeMap(
        [in] SAFEARRAY(long) rateToTimeMap);
}

#endif /* _L__Imit_r2r_ITimeBiasLinModel_idl_INCLUDED */
```

Imit_r2r_IController.idl

```
#ifndef _L__Imit_r2r_IController_idl_INCLUDED
#define _L__Imit_r2r_IController_idl_INCLUDED


import “Imit_r2r_IModel.idl”;

[
    object, version(1.0),
    uuid(7d42ff08-b39d-3d90-1d48-48176b1800d6),
    dual,
    helpstring(“Imit_r2r_IController Interface”),
    pointer_default(unique)
]
interface Imit_r2r_IController : Imit_IObject
{
    [id(100), helpstring(“method getController”)] HRESULT getController(
        [out,retval] Imit_r2r_IController** controller);

    /*
    [id(103)] getControllerState()
    [id(104)] setControllerState()
    */


    [id(110), helpstring(“method getConfigs”)] HRESULT getConfigs(
        [in,out] SAFEARRAY(double)* configs,
        [out] long* configsLen);


    [id(111), helpstring(“method getControllerName”)] HRESULT getControllerName(
        [out,retval] BSTR* controllerName);


    [id(112), helpstring(“method getModel”)] HRESULT getModel(
        [out,retval] Imit_r2r_IModel** model);


    [id(113), helpstring(“method getNextEquipTime”)] HRESULT getNextEquipTime(
        [out,retval] double* nextEquipTime);

    [id(114), helpstring(“method getNextInput”)] HRESULT getNextInput(
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);

    [id(115), helpstring(“method getNextOutput”)] HRESULT getNextOutput(
        [in,out] SAFEARRAY(double)* nextOutput,
        [out] long* nextOutputLen);

    [id(116), helpstring(“method getNextRunNum”)] HRESULT getNextRunNum(
        [out,retval] long* nextRunNum);

    [id(117), helpstring(“method getNextTarget”)] HRESULT getNextTarget(
        [in,out] SAFEARRAY(double)* nextTarget,
        [out] long* nextTargetLen);
```

```
[id(118), helpstring(“method getRunEquipTime”)] HRESULT getRunEquipTime(
    [out,retval] double* runEquipTime);

[id(119), helpstring(“method getRunInput”)] HRESULT getRunInput(
    [in,out] SAFEARRAY(double)* runInput,
    [out] long* runInputLen);

[id(120), helpstring(“method getRunNum”)] HRESULT getRunNum(
    [out,retval] long* runNum);

[id(121), helpstring(“method getRunOutput”)] HRESULT getRunOutput(
    [in,out] SAFEARRAY(double)* runOutput,
    [out] long* runOutputLen);




[id(122), helpstring(“method setConfigs”)] HRESULT setConfigs(
    [in] SAFEARRAY(double) configs);


[id(123), helpstring(“method setControllerName”)] HRESULT setControllerName(
    [in] BSTR controllerName);


[id(124), helpstring(“method setModel”)] HRESULT setModel(
    [in] Imit_r2r_IModel* model);


[id(125), helpstring(“method setNextEquipTime”)] HRESULT setNextEquipTime(
    [in] double nextEquipTime);

[id(126), helpstring(“method setNextInput”)] HRESULT setNextInput(
    [in] SAFEARRAY(double) nextInput);

[id(127), helpstring(“method setNextOutput”)] HRESULT setNextOutput(
    [in] SAFEARRAY(double) nextOutput);

[id(128), helpstring(“method setNextRunNum”)] HRESULT setNextRunNum(
    [in] long nextRunNum);

[id(129), helpstring(“method setNextTarget”)] HRESULT setNextTarget(
    [in] SAFEARRAY(double) nextTarget);


[id(130), helpstring(“method setRunEquipTime”)] HRESULT setRunEquipTime(
    [in] double runEquipTime);

[id(131), helpstring(“method setRunInput”)] HRESULT setRunInput(
    [in] SAFEARRAY(double) runInput);

[id(132), helpstring(“method setRunNum”)] HRESULT setRunNum(
    [in] long runNum);

[id(133), helpstring(“method setRunOutput”)] HRESULT setRunOutput(
    [in] SAFEARRAY(double) runOutput);
```

```
    [id(134), helpstring(“method solve”)] HRESULT solve(
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);

    [id(135), helpstring(“method solve_IT”)] HRESULT solve_IT(
        [in] SAFEARRAY(double) runInput,
        [in] long nextRunNum,
        [in] double nextEquipTime,
        [in] SAFEARRAY(double) nextTarget,
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);

    [id(136), helpstring(“method control”)] HRESULT control(
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);

    [id(137), helpstring(“method control_O”)] HRESULT control_O(
        [in] SAFEARRAY(double) runOutput,
        [in] long nextRunNum,
        [in] double nextEquipTime,
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);

    [id(138), helpstring(“method control_IOT”)] HRESULT control_IOT(
        [in] SAFEARRAY(double) runInput,
        [in] long runNum,
        [in] double runEquipTime,
        [in] SAFEARRAY(double) runOutput,
        [in] long nextRunNum,
        [in] double nextEquipTime,
        [in] SAFEARRAY(double) nextTarget,
        [in,out] SAFEARRAY(double)* nextInput,
        [out] long* nextInputLen);
}


#endif /* _L__Imit_r2r_IController_idl_INCLUDED */
```

Imit_r2r_IEwmaController.idl

```
#ifndef _L__Imit_r2r_IEwmaController_idl_INCLUDED
#define _L__Imit_r2r_IEwmaController_idl_INCLUDED


import “Imit_r2r_IController.idl”;

[
    object, version(1.0),
    uuid(06881417-79d3-3581-1d61-a7b8a7a5d0b6),
    dual,
    helpstring(“Imit_r2r_IEwmaController Interface”),
    pointer_default(unique)
]
interface Imit_r2r_IEwmaController : Imit_r2r_IController
{
    [id(200), helpstring(“method getEwmaController”)] HRESULT getEwmaController(
        [out,retval] Imit_r2r_IEwmaController** ewmaController);
    /*
    [id(201)] getEwmaState();
    [id(202)] getEwmaControllerState();
    [id(203)] setEwmaState();
    [id(204)] setEwmaControllerState();
    */

    [id(210), helpstring(“method getBiasAlpha”)] HRESULT getBiasAlpha(
        [in,out] SAFEARRAY(double)* biasAlpha,
        [out] long* biasAlphaLen);

    [id(211), helpstring(“method getBiasAlpha”)] HRESULT setBiasAlpha(
        [in] SAFEARRAY(double) biasAlpha);

}


#endif /* _L__Imit_r2r_IEwmaController_idl_INCLUDED */
```

CBiasLinModel.idl

```
// CBiasLinModel.idl : IDL source
//

// This file will be processed by the MIDL tool to
// produce the type library (CBiasLinModel.tlb) and marshalling code.

// import “oaidl.idl”;
// import “ocidl.idl”;


import “Imit_r2r_IBiasLinModel.idl”;
import “Imit_r2r_IBiasModel.idl”;

[
    uuid(14AFE341-50CF-11d3-A942-00C06D13817E),
    version(1.0),
    helpstring(“BiasLinModel 1.0 Type Library”)
]
library BiasLinMODELLib
{
    importlib(“stdole32.tlb”);
    importlib(“stdole2.tlb”);

    [
        uuid(14AFE342-50CF-11d3-A942-00C06D13817E),
        helpstring(“CBiasLinModel Class”)
    ]
    coclass CBiasLinModel
    {
        [default] interface Imit_r2r_IBiasLinModel;
        interface Imit_r2r_IBiasModel;
        // interface Imit_r2r_ILinModel;
    };
};
```

CTimeBiasModel.idl

```
// CTimeBiasModel.idl : IDL source
//

// This file will be processed by the MIDL tool to
// produce the type library (CTimeBiasLModel.tlb) and marshalling code.

// import “oaidl.idl”;
// import “ocidl.idl”;


import “Imit_r2r_ITimeBiasModel.idl”;
import “Imit_r2r_ITimeModel.idl”;

[
    uuid(14AFE343-50CF-11d3-A942-00C06D13817E),
    version(1.0),
    helpstring(“TimeBiasModel 1.0 Type Library”)
]
library TimeBiasMODELLib
{
    importlib(“stdole32.tlb”);
    importlib(“stdole2.tlb”);

    [
        uuid(14AFE344-50CF-11d3-A942-00C06D13817E),
        helpstring(“CTimeBiasModel Class”)
    ]
    coclass CTimeBiasModel
    {
        [default] interface Imit_r2r_ITimeBiasModel;
        interface Imit_r2r_ITimeModel;
    };
};
```

CTimeBiasLinModel.idl

```
// MyBiasLinModel.idl : IDL source for MyBiasLinModel.dll
//

// This file will be processed by the MIDL tool to
// produce the type library (MyBiasLinModel.tlb) and marshalling code.

// import “oaidl.idl”;
// import “ocidl.idl”;

import “Imit_r2r_ITimeBiasLinModel.idl”;
import “Imit_r2r_ITimeBiasModel.idl”;
import “Imit_r2r_ITimeModel.idl”;


[
    uuid(1A0B4411-346A-11D3-A942-00C06D13817E),
    version(1.0),
    helpstring(“Model 1.0 Type Library”)
]
library MODELLib
{
    importlib(“stdole32.tlb”);
    importlib(“stdole2.tlb”);

    [
        uuid(937E7DC1-50CB-11d3-A942-00C06D13817E),
        helpstring(“CTimeBiasLinModel Class”)
    ]
    coclass CTimeBiasLinModel
    {
        [default] interface Imit_r2r_ITimeBiasLinModel;
        interface Imit_r2r_ITimeBiasModel;
        interface Imit_r2r_IBiasModel;
        interface Imit_r2r_ITimeModel;
    };
};
```

CEwmaController.idl

```
// CEwmaController.idl : IDL source
//

// This file will be processed by the MIDL tool to
// produce the type library (CEwmaController.tlb) and marshalling code.

// import “oaidl.idl”;
// import “ocidl.idl”;


import “Imit_r2r_IEwmaController.idl”;

[
    uuid(A95892C1-5189-11d3-A942-00C06D13817E),
    version(1.0),
    helpstring(“EwmaController 1.0 Type Library”)
]
library EwmaCONTROLLib
{
    importlib(“stdole32.tlb”);
    importlib(“stdole2.tlb”);

    [
        uuid(A95892C2-5189-11d3-A942-00C06D13817E),
        helpstring(“CEwmaController Class”)
    ]
    coclass CEwmaController
    {
        [default] interface Imit_r2r_IEwmaController;
    };
};
```

Appendix B

User's Guide for COM-Based Controller Objects

COM provides a fairly rich language for developing data structures and interfaces that use those structures. However, many COM client environments support only Automation servers instead of generic COM servers. In particular, National Instruments' LabView falls under this category, which is currently one of the primary development environments at On-Line Technologies. Automation servers must extend the "IDispatch" interface and cannot support complex data structures built into the IDL. Clients can call object methods indirectly through the IDispatch interface rather than directly calling methods through interface pointers.

Many Automation clients are also limited to creating and accessing the "default" interface of Automation servers. Thus simple aggregation does not make all methods available to those clients. To circumvent this limitation, aggregation is supplemented by adding aggregated methods to the default (singly inherited) Automation interface. For example, the Bias Model and Linear Model interfaces both follow simple single inheritance constructs to support the Model interface. The Bias Linear Model, however, must include both the Bias Model and Linear Model interface methods, without multiply inheriting from both. This is accomplished by creating a Bias Linear Model interface that inherits from the Linear Model interface and copies the additional Bias Model methods into its IDL.

The COM IDL for this project is broken into a number of hierarchical files that build

component interfaces through single inheritance and aggregation / method copies (where multiple inheritance would be required). Automation compatible IDL is used to provide the widest possible application coverage. The following sections describe the State variables and Methods provided by the run-to-run control objects. Appendix A contains the full COM IDL files.

Imit_Base.idl

This IDL file contains the base object interface methods that all of the MIT objects should support. These methods provide convenient identification, connection, and state storage (persistence) capabilities. Only the getID() and setID() methods are completely implemented for the run-to-run controller objects used here. (This does not limit functionality, as the additional methods would primarily be used with high level organization and object management tools.)

State Variables:

BSTR Id .....String tag that identifies the object (should be unique across all objects)
This parameter is not used by the implementation, but is available for object management and display purposes

Imit_r2r_IModel.idl

This IDL file contains methods that all input / output process models must provide. The interface supports the ability to pass inputs forward through the model to get outputs, and methods to back-solve for optimal inputs based on a target output vector and constraints on the solution.

State Variables:

BSTR ModelName.....String tag that identifies the Model name (could be unique across different Model implementations) This parameter is not used by the implementation, but is available for object management and display purposes.

Long numInputs.....Number of system inputs
Long numOutputs .....Number of system outputs

BSTR[numIn] InputNames.....Array of strings that name the input parameters
BSTR[numOut] OutputNames.....Array of strings that name the output parameters
These parameters are not used by the implementation, but are available for object management and display purposes.

Long runNum.....Run number on which the current Model is based (from the last update)
Double equipTime.....Equipment Time on which the current Model is based
Double[numIn] input .....Input vector used for the last model update

Double[numIn] inputMax .....Upper bounds on input parameter selection
Double[numIn] inputMin.....Lower bounds on input parameter selection
Double[numIn] resolution.....Granularity constraint for input parameter selection
Double[numIn] weightInput .....Input weighting vector for a minimum change input solution (if multiple solutions exist)
Double[numOut] weightOutput.....Output weighting vector, for a minimum error solution (if no exact solution exists)
Int discFlag.....Enumerated input discretization flag
0 = Not initialized,
1 = No discretization,
2 = Simple rounding,
3 = Delta Discretization,
4 = Statistical Discretization,
5 = Statistical Discretization (method 2)

Double[] configs.....Configuration “catchall” parameter, a vector of doubles

Methods:

fit(.....Fit the model to data from a set of experimental runs (probably a DOE)

[in] Double[numRuns][numIn]
runInputs, ...The output vectors for each run
[in] Long[numRuns] runNums, ..The run numbers for each run
[in] Double[numRuns]
runEquipTimes, ...The equipment times for each run
[in] Double[numRuns][numOut]
runOutputs .....The output measurements for each run
)

func(.....Use the Model to find outputs based on an input vector
[in] Double[numIn] runInput, ...Input vector for the run
[in] Long runNum, ...Run number for the run
[in] Double runEquipTime, ...Equipment time for the run
[in,out] Double[numOut]
runOutput, ...The projected output from the system, given the input vector
[out] Long runOutputLen .....Length of the output vector
)

solve (.....Use the Model to "backsolve" for optimal inputs based on requirments for the next run
[in] Double[numIn] runInput, ...Input vector for the last run
(Only used if multiple exact solutions exist)

[in] Long nextRunNum, ...Run number for the solution
[in] Double nextEquipTime, ...Equipment time for the solution
[in] Double[numOut]
nextTarget, ...Target output vector for the solution

[in,out] Double[numIn]
nextInput, ...Resulting input vector solution
[out] Long nextInputLen.....Length of the solution vector
)

Imit_r2r_ILinModel.idl

This IDL file contains methods for accessing the Linear Model State variables.

State Variables:

Double[numOut] intercept.....Linear Model intercept terms (may or may not be distinct from the bias terms)
Double[numOut][numIn] slope .....Slope coefficient matrix (2 dimensional) relating the

outputs to the inputs (elements should be contiguous along the numOut dimension)

Imit_r2r_IBiasModel.idl

This IDL file contains methods for accessing the Bias Model State variables.

State Variables:

Double[numOut] bias.....Output bias vector (normally modified by a control scheme)

* Note, the methods used to get and set the bias (offset) vector also include the parameters “runNum,” “runEquipTime,” and “runInput.” These parameters are present for proper management of the bias vector in the event that the input settings (including the run number and/or equipment time) affect the bias in some manner. This is the case for the Time-based Bias Model, where the bias vector is actually stored inside of the time layer, and time factors must be accounted for before passing bias terms in or out of the rate Models. (The use of a zero length “runInput” vector is used to specify direct access to the underlying model’s bias vector.)

Imit_r2r_ITimeModel.idl

This IDL file contains methods for accessing the Time-based Model State variables.

State Variables:

Long numTimeInputs.....Number of time-based inputs
Long numRateOutputs.....Number of rate-based outputs
Imit_r2r_IModel* primaryModel .....Reference to the “slaved” underlying Model
Int[numRateOut] rateToTimeMap .....Index map linking each rate-based output to a time-based input

Imit_r2r_IController.idl

This IDL file contains methods that all Model-based run-to-run Controllers must provide. This interface supports requests for run-to-run control actions and “back-solving” for input vectors from the Model.

State Variables:

BSTR controllerName .....String tag that identifies the Controller type (could be unique across different Controller implementations) This parameter is not used by the implementation, but is available for object management and display purposes.

Imit_r2r_IModel* model .....Reference to the Controller’s Model

Long runNum.....Run number from the last run
Double runEquipTime.....Equipment time from the last run
Double[numIn] runInput.....Input settings for the last run
Double[numOut] runOutput.....Output measurements from the last run

Long nextRunNum.....Run number for the next run
Double nextEquipTime .....Equipment Time for the next run
Double[numOut] nextTarget .....Target outputs for the next run

Double[numIn] nextInput .....Suggested inputs for the next run

Double[numOut] nextOutput .....Expected outputs for the next run

Double[] configs.....Configuration “catchall” parameter, a vector of doubles

Methods

solve (.....Use the Model to “backsolve” for optimal inputs based on the Controller’s requirements for the next run
[in,out] Double[numIn]
nextInput, ...Suggested inputs for the next run (no control updates)
[out] Long nextInputLen.....Length of the solution vector
)

solve_IT (.....Use the Model to “backsolve” for optimal inputs based on the supplied requirements for the next run

[in] Double[numIn] runInput, ...Input settings for the last run

[in] long nextRunNum, ...Run number for the next run
[in] double nextEquipTime, ...Equipment Time for the next run
[in] Double[numOut]
nextTarget,.....Target outputs for the next run

[in,out] Double[numIn]
nextInput, ...Suggested inputs for the next run (no control updates)
[out] Long nextInputLen.....Length of the solution vector
)

control (.....Based on the Controller’s state, update the Model, and use the Model to “backsolve” for optimal inputs

[in,out] Double[numIn]
nextInput, ...Suggested inputs for the next run (after control updates)
[out] Long nextInputLen.....Length of the solution vector
)

control_O (.....Based on the output from the last run and the Controller’s state, update the Model, and use the Model to “backsolve” for optimal inputs

[in] Double[numOut] runOutput,.Outputs from the last run

[in] Long nextRunNum,.....Run number for the next run
[in] Double nextEquipTime, ...Equipment Time for the next run

[in,out] Double[numIn]
nextInput, ...Suggested inputs for the next run (after control updates)
[out] Long nextInputLen.....Length of the solution vector
)

control_IOT (.....Based on the inputs and output from the last run and a new target run, update the Model and use the Model to “backsolve” for optimal inputs

[in] Double[numIn] runInput, ...Inputs for the last run
[in] Long runNum, ...Run number for the last run
[in] Double runEquipTime,.....Equipment Time for the last run

[in] Double[numOut] runOutput,.Outputs from the last run

[in] Long nextRunNum,.....Run number for the next run
[in] Double nextEquipTime, ...Equipment Time for the next run
[in] Double[numOut]
nextTarget,.....Target outputs for the next run

[in,out] Double[numIn]
nextInput, ...Suggested inputs for the next run (after control updates)
[out] Long nextInputLen.....Length of the solution vector
)

Imit_r2r_IEwmaController.idl

This IDL file contains methods for accessing the EWMA Controller State variables.

State Variables:

Double[numOut] biasAlpha.....The “forgetting factors” for each of the bias terms

B.1 Implementation of the Run-to-Run Controller Components

The three objects shown in Figure 6-3 have been implemented as in-process Automation servers (Windows dll’s). The Bias Linear Model, Time-based Bias Model, and EWMA Controller were all written in Java and compiled with Microsoft’s Visual J++ using Microsoft’s extensions to support COM Automation objects. The following sections describe some of the relevant details about each of these implementations.

ComBiasLinModel.dll

The Bias Linear Model (BLModel) implementation is a straightforward adaptation of the original MIT run-to-run controller’s Linear Model function and back-solution algorithm. Please note that it is up to the client to set and use the “runNum,” “equipTime,” and “input” State variables. The BLModel implementation does not use these variables in any

way.

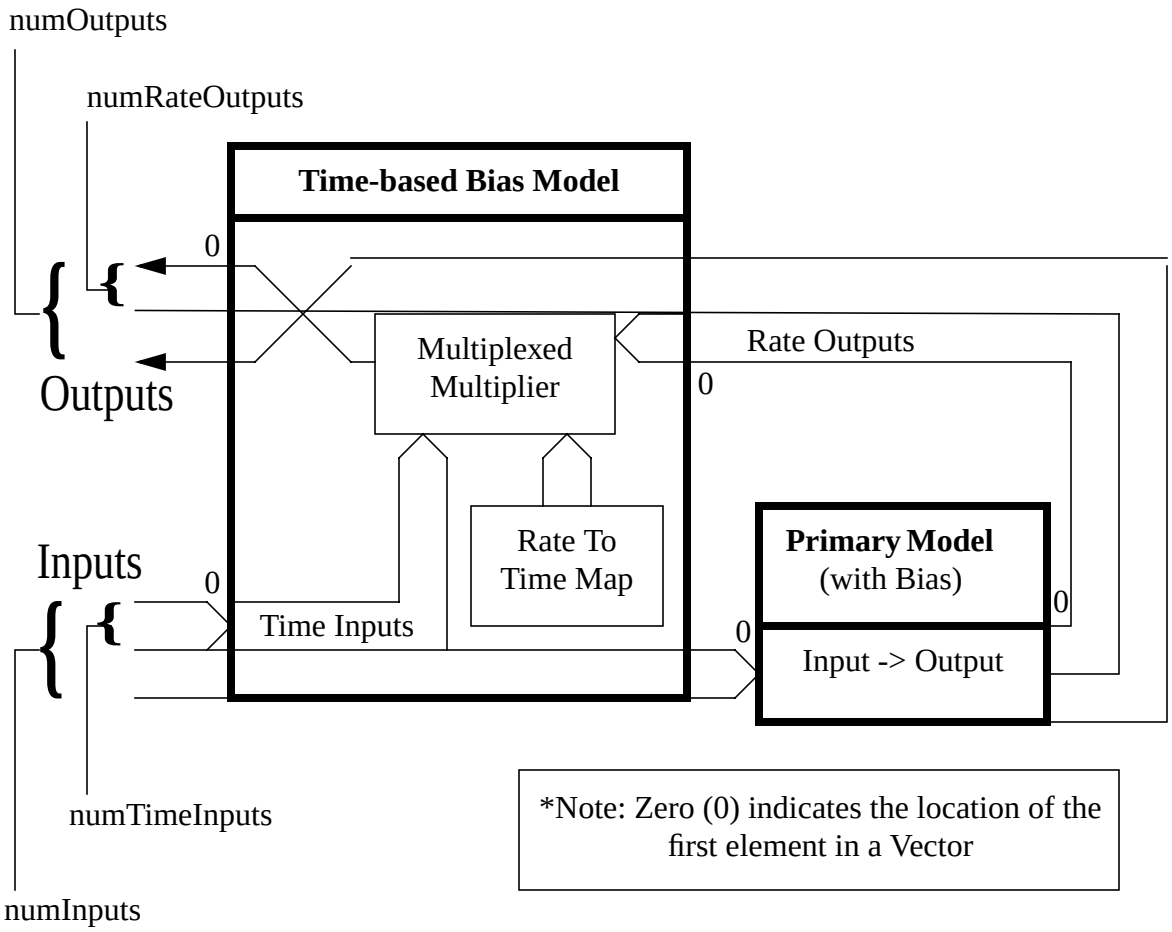
ComTimeBiasModel.dll

The Time-based Bias Model (TBModel) implementation requires a pointer to a Bias Model, such as the BLModel described above. It does not require the use of a Linear Model and will properly utilize any Bias Model. The “solve()” routine iteratively searches for an optimal solution by calling on the Bias Model’s “solve()” method. While this may not be the most efficient algorithm, it is arguably one of the most flexible. Chapter 5 discussed this solver in detail.

The “configs” parameter is used to control the optimization process. It is a two-element array, the first of which determines the cost improvement threshold for the optimization process. The second “configs” parameter establishes the maximum number of iterations that the Model will perform during optimization. That is, the optimization process will stop when the cost function improvement from one iteration to the next falls below the threshold, or the maximum number of iterations has been reached.

The TBModel uses the “numTimeInputs,” “numRateOutputs,” and “rateToTimeMap” variables to manage the structure of the Time-based Model. Figure B-1 demonstrates how this is accomplished. The first “numTimeInputs” elements of the input vector are separated as time-based inputs and the first “numRateOutputs” elements of the Primary Model’s outputs are treated as rate-based outputs. The “rateToTimeMap” is a vector of integer indices that map each rate-based output element to one of the time-based input elements. (This vector uses an index of zero to indicate the first time-based element.)

Figure B-1: Time-based Model Input / Output Data Paths



It is also important to note that the TBModel completely takes over the Primary Model. All Bias Model parameter writes and reads to and from the TBModel invoke the proper parameter writes and reads on the Primary Model. The TBModel does not store any parameter information that would be replicated in the Primary Model (with the exception of the “DiscFlag” setting), which helps to avoid any synchronization issues. It is therefore important to set the TBModel’s Primary Model interface before any other State variables are initialized. The sizing information (“numInputs,” “numOutputs,” “numTimeInputs,” “numRateOutputs”) should be set next to ensure proper breakdown and storage of the other attributes.

The details of configuring the Primary Model's internal structure must be performed directly on the Primary Model; the TBModel does not know anything about the underlying model type. For the BLModel, this means that the slope and intercept terms must be set through the BLModel's interface. All other parameters should be set through the TBModel's interface.

ComEwmaController.dll

The EWMA Controller (EController) implementation is also a direct conversion from the original MIT run-to-run controller. The EWMA control action is composed of a simple update to the bias vector of the EController's Model. Thus any Model that supports the Bias Model interface is compatible with the EController.

The EController does not use the "equipTime" setting, but the run number is used to ensure the proper sequencing of control updates and model solutions. In particular, control updates are not performed when the Model's run number exceeds the run number for the update data. Also, Model "back-solutions" are disallowed when the target run number precedes the Model's run number.

When calling methods that specify new "run" settings or "nextRun" settings ("Solve _IT()," "Control_O()," or "Control_IOT()"), the appropriate State variables in the Econtroller are set to match the input parameters. Also, upon successful completion of a Model "back-solution" (from "Solve()," "Solve_IT()," "Control()," "Control_O()," or "Control_IOT()"), the "nextRun" settings, including the "nextInput" vector, are copied into the "run" settings in preparation for receiving feedback from the next run. It is recommended that the "Control_O()" method be used for continuous operation of the Econtroller when "nextTarget" is held constant and all of the "nextInput" recommendations are

used.