# Novel Techniques for the Run by Run Process Control of Chemical-Mechanical Polishing

by

Taber H. Smith

B.S. in Electrical Engineering, Rochester Institute of Technology (1994)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

Author ..................................................

Department of Electrical Engineering and Computer Science

May 24th, 1996

Certified by..........................................................................

Duane S. Boning

Assistant Professor of Electrical Engineering

Thesis Supervisor

Accepted by ........................................

Chairman, Department Committee on Graduate Students

# Novel Techniques for the Run by Run Process Control of Chemical-Mechanical Polishing

by

Taber H. Smith

## Abstract

The purpose of this thesis is to explore control schemes for application to the Chemical-Mechanical Polishing (CMP) semiconductor fabrication process. Particular emphasis is placed on an Exponentially Weighted Moving Average (EWMA) controller, a Predictor-Corrector Controller (PCC), and Artificial Neural Network (ANN) controllers. These methods are explored with respect to their stability, responsiveness (optimal or otherwise), ability to incorporate practical issues and their applicability to the CMP process. These characteristics are evaluated through simulation and experiments performed on various CMP tools which highlight and demonstrate these principles.

## Acknowledgments

The work compiled in this thesis was accomplished with the help and support of many people who deserve more than my sincere thank you.

I would like to thank my advisor, Professor Duane Boning, for providing me with the opportunity to perform this research. Professor Boning has a rare ability to effectively communicate constructive criticism and provide keen insight and guidance with an amazingly positive attitude. This has enabled me to exceed even my own expectations while making my work exciting and enjoyable. In addition, his hard work put forth to read every report and discuss every idea in his limited time is tremendously appreciated.

I would like to thank my parents and brothers for their support from the day I was born; they have been an immeasurable influence on my life and career. I would also like to thank my wife, Tina, for being in every memory I have of the last 9 years and for being so supportive and forgiving of my work.

I also want to thank all my fellow research assistants who have helped me with this research. In particular, I wish to thank Aaron Gower for his tremendous insight into seemingly every problem I had to pose. His contributions helped close many holes in this research. David White has provided me with a much greater understanding of many control and neural network issues through numerous in-depth conversations regarding these issues. Minh Lee has greatly improved my limited knowledge of semiconductor physics and processing. William Moyne sparked much of my research into the EWMA controller and has continued to support me in my study of his past work. Ka Shun Wong has aided me immensely in this work through his unparalleled enthusiasm to help me (and everyone else) with any problem I (they) had. I would also like to thank Chantal Wright for being a great office-mate and friend over the past year and a half.

Arnon Hurwitz, our project leader has offered his ability to move this project forward while insuring that all the gaps be filled along the way. James Moyne has provided a great deal of support through his work in conjunction with ours as well as many valuable suggestions for my work (mostly on the fly in teleconferences). Dr. Armann Ingolfsson provided valuable suggestions and insight on the EWMA stability analysis. Strasbaugh Inc. provided the opportunity to perform experiments which were critical to the success of this thesis and John Curry's efforts in making these a reality are greatly appreciated. Finally, I would like to thank Dr. Stephanie Butler for her interest, suggestions, and constructive criticism of much of the EWMA and PCC research.

Finally, I would like to thank SEMATECH and ARPA (under contract #N00174-93-C-0035) for their generous financial support of this research, without which this would not have been possible.

# Table of Contents

**Part I - Background** 7

Chapter 1. Introduction ........ 8

Chapter 2. Background of CMP Process Control ........ 10

2.1 The CMP Process ........ 11
2.2 Background Issues for CMP Control ........ 13

Chapter 3. Potential Control Approaches ........ 14

3.1 Statistically Based Control Methodologies ........ 14
3.1.1 The EWMA and PCC Controllers ........ 15
3.1.2 Advantages and Disadvantages of the EWMA and PCC Controllers ........ 17
3.2 Classical Linear Feedback Control ........ 17
3.3 Stochastic Dynamic Programming ........ 18
3.3.1 A Brief Outline of Dynamic Programming ........ 18
3.3.2 An Example Dynamic Programming Architecture ........ 19
3.3.3 Advantages and Disadvantages of the DP Framework ........ 20
3.4 Artificial Neural Networks ........ 21
3.4.1 A Brief Review of Neural Networks ........ 21
3.4.2 Examples of Neural Network Controllers ........ 22
3.4.3 Advantages and Disadvantages of ANN Controllers ........ 25
3.5 Summary of Prospective Control Approaches ........ 26

**Part II - EWMA Control Strategies** 28

Chapter 4. The EWMA Run by Run Process Control Framework ........ 29

4.1 Specifics of the CMP Process and the EWMA Controller Related to this Chapter ........ 30
4.2 Input Constraints ........ 32
4.3 Input Weights ........ 33
4.3.1 Range Normalization ........ 34
4.3.2 Input Weighting ........ 34
4.4 Discretization ........ 35
4.5 EWMA and PCC Weighting Parameters ........ 37
4.6 Controller Experiments ........ 41
4.6.1 The First CMP Tool ........ 42
4.6.2 The Second CMP Tool (Strasbaugh 6DS-SP) ........ 43
4.7 Summary: The EWMA Run by Run Control Framework ........ 45

Chapter 5. Stability of the MIMO EWMA and PCC Algorithms ........ 46

5.1 The SISO EWMA Controller (An alternate approach) . . . . . . . . . . . . . . . . . . . . . . . . . 46
5.2 The MIMO EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
5.2.1 MIMO EWMA Example 1: Stability vs. Varying Model Error . . . . . . . . . . . . . . . 51
5.2.2 MIMO EWMA Example 2: Region of Stability Based vs. EWMA Weights. . . . . 56
5.3 The SISO PCC Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
5.3.1 SISO PCC Example 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.3.2 SISO PCC Example 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
5.4 The MIMO PCC Algorithm. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
5.4.1 MIMO PCC Example 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
5.5 Summary: Stability Analysis of the MIMO EWMA and PCC Controllers. . . . . . . . . . . 75

Chapter 6. Analytical Derivation of the Optimal SISO EWMA Controller Weights for a Linear Drifting Process Buried in White Noise .......................................................................76

6.1 Derivation of the Optimal EWMA Weight . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.1.1 Determination of the Process Output at Discrete-Time n. . . . . . . . . . . . . . . . . . . . 77
6.1.2 Minimize Steady-State Mean-Squared Error. . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
6.2 Simulation Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
6.3 Summary: Analytical Derivation of the Optimal EWMA Weights . . . . . . . . . . . . . . . . . 85

Chapter 7. Summary of EWMA Based Control Methods ...........................................................86

**Part III - Neural Network Controllers 88**

Chapter 8. A Direct Adaptive EWMA Controller Utilizing Artificial Neural Network Function Approximation Techniques.........................................................................................................89

8.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
8.2 The Optimal EWMA Weight Vector as a Function of the Disturbance State . . . . . . . . . 90
8.2.1 Model Error Impact on Optimal EWMA Weights . . . . . . . . . . . . . . . . . . . . . . . . . 90
8.2.2 Target Effect on Optimal EWMA Weights. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
8.2.3 Summary of Disturbance Effects on Optimal EWMA Weights . . . . . . . . . . . . . . . 94
8.2.4 Optimal EWMA Weight Map. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
8.3 The EWMA Controller with an ANN Optimal Weight Estimator . . . . . . . . . . . . . . . . . 96
8.3.1 Training the Neural Network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
8.3.2 Estimating the Disturbance State . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 98
8.4 Simulations of the EWMA Controller with ANN Weight Estimator . . . . . . . . . . . . . . . 99
8.5 Summary: A Direct Adaptive EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102

Chapter 9. An Indirect Adaptive Linear ANN Controller .........................................................103

Chapter 10. The ANN-EWMA Controller...................................................................................107

Chapter 11. An Adaptive HME ANN Controller ........................................................................110

Chapter 12. Comparative Analysis ..............................................................................................119

12.1 An Affine Process with ARMA Noise of the First Order . . . . . . . . . . . . . . . . . . . . . . 120
12.1.1 The EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
12.1.2 The PCC Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121
12.1.3 The Adaptive Linear Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122
12.1.4 The ANN-EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122
12.1.5 The HME Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
12.1.6 Summary of Controller Performances for the ARMA1 Case . . . . . . . . . . . . . . . 124
12.2 An Affine Process with a Linear Drift Buried in White Noise . . . . . . . . . . . . . . . . . . 124
12.2.1 The EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
12.2.2 The PCC Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
12.2.3 The Adaptive Linear Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
12.2.4 The ANN-EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
12.2.5 The HME Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127
12.2.6 Summary of Controller Performances for the Affine Process with a Linear Drift Buried in White Noise . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127
12.3 A Second Order Process with Small Model Error and a Linear Drift Buried in White Noise . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128
12.3.1 The EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128
12.3.2 The PCC Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
12.3.3 The Adaptive Linear Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
12.3.4 The ANN-EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 130
12.3.5 The HME Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
12.3.6 Summary of Controller Performances for a Second Order Process with Small Model Error and a Linear Drift Buried in White Noise . . . . . . . . . . . . . . . . . . . 131
12.4 A Second Order Process With Large Model Error and a Linear Drift Buried in White Noise . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
12.4.1 The EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
12.4.2 The PCC Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
12.4.3 The Adaptive Linear Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
12.4.4 The ANN-EWMA Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
12.4.5 The HME Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
12.4.6 Summary of Controller Performances for a Second Order Process With Large Model Error and a Linear Drift Buried in White Noise . . . . . . . . . . . . . . . . . . . 135
12.5 Overall Summary of Controller Performances . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135

Chapter 13. Conclusions and Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137

# Chapter 1

# Introduction

The Chemical-Mechanical Planarization (CMP) process is of critical importance to current and future generation interconnect for integrated circuit technologies. CMP sales and usage are increasing exponentially [Martinez, Jairath]. However, CMP is not easy and it is not cheap [Martinez]. These issues are at the heart of this thesis, whose purpose is to demonstrate that through run by run process control, the CMP process can be made less expensive and less difficult.

This thesis is broken up into three main parts. Part I consists of this introduction as well as the necessary CMP and run by run control background. Specifically, Chapter 2 summarizes the CMP process and CMP process control. Potential control approaches for the run by run control of the CMP process are described and qualitatively compared in Chapter 3.

Part II (Chapters 4-7) examines exponentially weighted moving average (EWMA) based controllers. Chapter 4 discusses some practical issues involved with properly implementing the EWMA controller and provides experimental results of the successful implementation of these methods. Stability analyses for multiple-input multiple-output (MIMO) versions of the EWMA and (Predictor Corrector Control) PCC controllers are provided in Chapter 5. An analytic derivation of the optimal EWMA weights to be used with a linearly drifting process buried in white noise is contained in Chapter 6. Chapter 7 summarizes Part II and also highlights areas where future work is needed.

Part III considers several artificial neural network (ANN) based controllers. Chapter 8 demonstrates an ANN technique for dynamic estimation of the control parameters in the EWMA and PCC controllers. Chapter 9 extends the idea of constant term adaptation to full model adaptation by outlining an indirect adaptive linear ANN controller. An indirect adaptive ANN controller consisting of a nonlinear ANN model whose bias terms in the output layer are adapted using an EWMA is demonstrated in Chapter 10. Chapter 11 outlines a fully adaptive nonlinear Hierarchi-

# Part I - Background

Part I describes the chemical-mechanical polishing (CMP) process and provides a survey of potential control methods which may be applicable to the CMP process.

cal Mixtures of Experts (HME) ANN controller. Chapter 12 compares many of these controllers on several different process plants and Chapter 13 concludes the thesis and discusses barriers and opportunities for CMP process control.

Readers who are interested primarily in the CMP aspects of the thesis and the results gained by the application of these control methods may start with Chapters 1 and 2, as well as Sections 1 and 4 of Chapter 3, then focus on Chapters 4 and 7, and finish with Chapters 8, 12, and 13. Those readers interested in the more mathematical issues of the run by run control techniques should start with Chapters 1, 3, and Sections 5 and 6 of Chapter 4, then focus on Chapters 5-8, and conclude with Chapters 10-13.

# Chapter 2

# Background of CMP Process Control

The purpose of a control system is to obtain a desired response from all or part of its environment. The environment we are attempting to control will be that of the chemical-mechanical polishing (CMP) process. Control systems are of two general types; open-loop or closed-loop [Dorf]. Open-loop control strategies, like Statistical Process Control (SPC), are widely used in the semiconductor industry, while closed-loop or feedback control has yet to be widely adapted. In SPC methods, a system such as a chemical-mechanical polisher is initially optimized with a fixed recipe and the process is monitored using statistical methods. A typical statistical method is to use an Exponentially Weighted Moving Average (EWMA) to track the process and make a recipe change only when a significant deviation from the process setpoint occurs. This is often the preferred type of process control because it is believed to reduce variation in the process [Deming, Box]. This is important in semiconductor processing, where locating problems in a long and extremely complex fabrication process is difficult. In the automatic control field the term control typically implies closed-loop feedback control [Dorf]. With feedback control, continuous (or discrete) measurements of the process outputs cause real-time (run by run) updates to the process recipes. The motivation behind this strategy is to make frequent small changes to the recipe in order to maintain a process target over a longer period of time in order to reduce equipment downtime, improve the process dynamics over many runs, extend the life of the machine and/or its parts, and decrease the usage of consumable materials. These characteristics motivate the use of feedback control in the semiconductor industry. In order to do so, it is necessary to develop control strategies that address the issues of increased process variability and the progression of variation over several process steps. This work will focus almost entirely on feedback control because it is believed that analysis methods exist to properly monitor process variation while effectively controlling the process with an appropriate control strategy. These issues must be central to any successful implementation of a closed-loop control strategy for the CMP process.

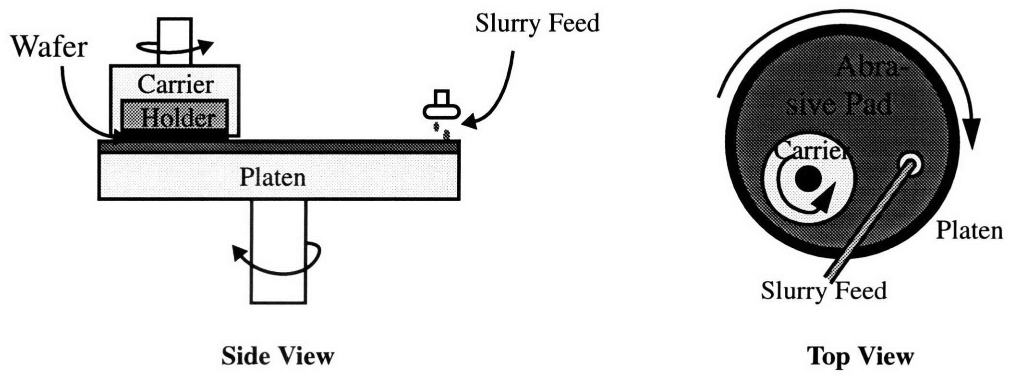
## 2.1 The CMP Process

Before deciding upon directions in which the control of CMP should move, it is important that we understand what the characteristics of the CMP process are. Only then can we intelligently reject possibilities that do not fit the demands of the CMP process and pursue methods which offer great promise.

In addition to CMP process and equipment development [Fury], the modeling of CMP processes is an active area of research, including work on wafer scale dependencies [Runnels1], feature scale models [Runnels2, Chang1], as well as behavior of the equipment over many runs [Hu1, Hu2]. The challenges posed by CMP for both sensor and control research are also becoming better known [Jairath, Hu3]. While a good deal of research into run by run control methods has been reported (see [DelCastillo] for a review), relatively little practical experience with CMP control exists [Jairath, Altman, Moyne1].

In the CMP process, the wafer is affixed to a wafer carrier, and pressed face-down on a rotating platen holding a polishing pad, as illustrated in Fig. 2-1. A slurry with abrasive material (e.g. silica particles of size 1-200$\mu$m) held in suspension is dripped onto the rotating platen during polish. The carrier and platen rotate at variable speeds, typically on the order of 30 rpm. Tools differ in the number of wafers that may be simultaneously polished; single-wafer, dual-wafer, and five-headed tools exist.

**Figure 2-1. Chemical-Mechanical Polish Tool Configuration.**

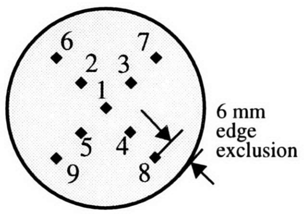


The process removes material at the surface of the wafer through a combination of mechanical and chemical action. A typical process goal is to achieve "global" planarization (across tens of mm) by preferential removal of "high" material on the wafer. The planarization of dielectric (silicon dioxide) layers between multilevel metallization steps is one common application. Metal planarization is also often performed.

The control of CMP is chronically poor, arising from poor understanding of the process, degradation (wear-out) of polishing pads, inconsistency of the slurry, and the lack of in-situ sensors. Because the process includes mechanical abrasion of the surface, the polishing pad wears rapidly. Concurrent or sequential "conditioning" is usually employed whereby the abrasive surface of the pad is restored (either by mechanical damage to the surface or removal of a thin surface layer). In addition to difficulties achieving a reliable film thickness (because of changing removal rates over time), the within-wafer uniformity of the polish is difficult to achieve and maintain. Differences between polish rates at the center and edge of the wafer (i.e. "bulls-eye patterns") may arise due to wafer asymmetry (e.g. wafer flat), non-constant relative pad velocity from the edge to the center, non-uniform slurry and by-product transport under the wafer, wafer bowing due to pressure, or machine drift in time of any of these parameters. As a result of these problems, it is conventional practice to use a number of send-ahead or dummy wafers to condition and/or calibrate the tool before or after each lot of wafers. Constant supervision and maintenance of the process is required in order to maintain a tight wafer-to-wafer uniformity (3-5%) and within-wafer uniformity (5-10%). The lifetime of a pad (for a tightly monitored wafer-to-wafer uniformity) is typically on the order of 100-200 wafers. This causes expensive machine downtime as well as wasted product and equipment consumables.

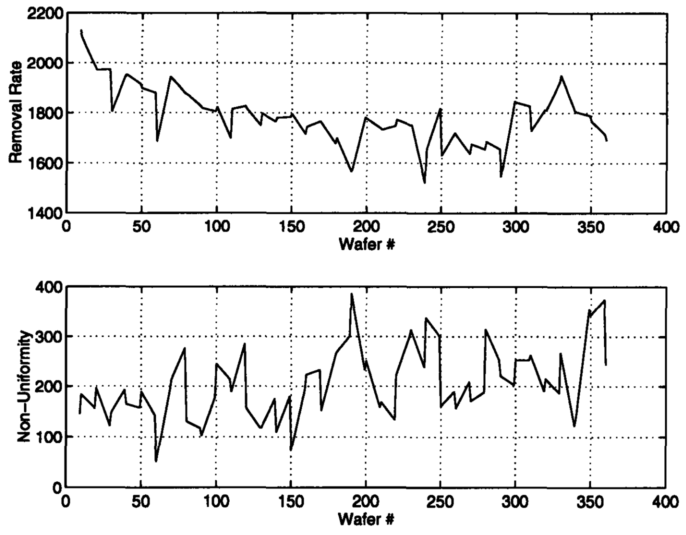
In this work, the product characteristics of concern are the removal rate (corresponding to a controlled amount of oxide polished during the step) and the within-wafer uniformity of that removal rate across the wafer. The removal rate is determined by the difference of the measured oxide film thickness before and after polish at each of several sites on the wafer (as shown in Fig. 2-2 for example), divided by the (fixed) polish time. The "removal rate" output is the average of the several sites on a wafer. The "nonuniformity" output parameter is computed for each wafer as the standard deviation of the amount removed over the nine sites on the wafer, divided by the average amount removed over the several sites, times 100.

**Figure 2-2. Measurement Sites**



The change in removal rate and nonuniformity for a typical uncontrolled or baseline oxide polish process (with a fixed recipe) is shown in Fig. 2-3. This run was conducted on a typical industrial single wafer polishing tool, using 8" silicon wafers with a thick blanket oxide deposition.

Figure 2-3. Baseline CMP Experiment



## 2.2 Background Issues for CMP Control

The CMP process is currently plagued by several problems. First, steady, mostly linear, decreases in removal rate not only increase processing time but also decrease the life of pads that are used [Boning1, Boning2]. Second, within-wafer uniformity is not only poor but inconsistent, thus reducing the effectiveness of the CMP process [Boning1, Boning2, Chang1, Davis]. Third, the effect of dishing causes pattern dependent changes in thickness [Stell]. Wafer scratching caused by CMP can create severe failures in manufactured circuits [Stell]. Finally, the lack of in-situ sensors make process control or even process monitoring difficult. An effective control methodology must address all of these issues. Specifically, the process control technique used for CMP must be capable of tracking and correcting linear drifts in removal rate as well as step changes in removal rate due to pad changes. The ability to maintain a steady target removal rate for long periods of time is essential in eliminating wasteful reprocessing and recalibration time, decreasing machine downtime, and increasing throughput. Any effective control method must also provide the ability to monitor and improve the uniformity of the CMP process. This becomes increasingly important as the number of circuit layers increase and line-width decreases [Martinez]. Finally, in addition to process noise (both in removal rate and uniformity,) many disturbances of the CMP process are poorly understood, e.g. dishing and scratching. Therefore, the ideal controller would also have some, albeit minor, ability to learn and compensate for these unmodeled effects.

A control algorithm for use on existing tools must be able to perform these tasks with a limited amount of information (due to poor metrology) which is inherently provided in a discrete manner (post process measurement data) due to the lack of in-situ sensors. In contrast, a control algorithm for use with new tools must be able to utilize a large amount of information provided by recent advances in metrology [Chang1, Davis]. The ability to simplify and/or utilize a large amount of information in an intelligent way may be a very difficult task for many control methods.

# Chapter 3

# Potential Control Approaches

In this chapter we consider a spectrum of control approaches to CMP process control. These include SPC, linear feedback control, stochastic dynamic programming, and neural network architectures. For each, we briefly review the method and consider the advantages and disadvantages for potential application to CMP control.

## 3.1 Statistically Based Control Methodologies

Open-loop Statistical Process Control (SPC) methods such as the Shewart control chart, the CUSUM chart, and the moving average have been around a long time [Hunter]. The Exponentially Weighted Moving Average (EWMA) is a more recent addition to this type of control chart.

An EWMA control chart monitors a process using an exponentially weighted moving average of the process output. The moving average at discrete-time $n$ has the form:

$$y_{EWMA}[n] = w\,y[n] - (1-w)y_{EWMA}[n-1], \qquad (3\text{-}1)$$

where $0 < w < 1$. We see that higher values of $w$ imply that recent measurements more strongly affect the average. For a given random process, one may determine an optimal $w$ for monitoring the process when a drift or shift has caused the process to exceed its performance limits. Once an alarm is signaled, the process is shut down to perform maintenance and to re-optimize the process recipe.

This type of open-loop operation is the most widely used method for process control in the semiconductor manufacturing industry. Numerous works including [Lucas, Crowder1, Crowder2] give methods for the design of EWMAs to monitor a process as well as charts for determining the optimal weighting schemes. Industry response to open-loop statistical process control has been overwhelming and the use of this type of control has become heavily ingrained in a wide variety of industries. Statistical process control has several limitations. The fundamental goal of SPC to minimize the variation and/or uncertainty caused by making changes on the fly (particularly when there are several inputs to a process and human intuition is not sufficient to guide modifications). The process is generally optimized off-line, and a large number of product or monitor wafers as well as a large amount of machine time may be lost. These features are particularly damaging in the semiconductor industry where wasted wafers and machine downtime are extremely expensive.

### 3.1.1 The EWMA and PCC Controllers

An alternative is to use closed-loop feedback control methodologies which make small incremental changes to the process recipes in order to avoid bringing the machine down to re-optimize the process. Due to the familiarity of the semiconductor industry with SPC methods, a closed-loop feedback control method based on the EWMA has been developed [Sachs]. The EWMA Controller shown in Fig. 3-1 uses an affine model of the form:

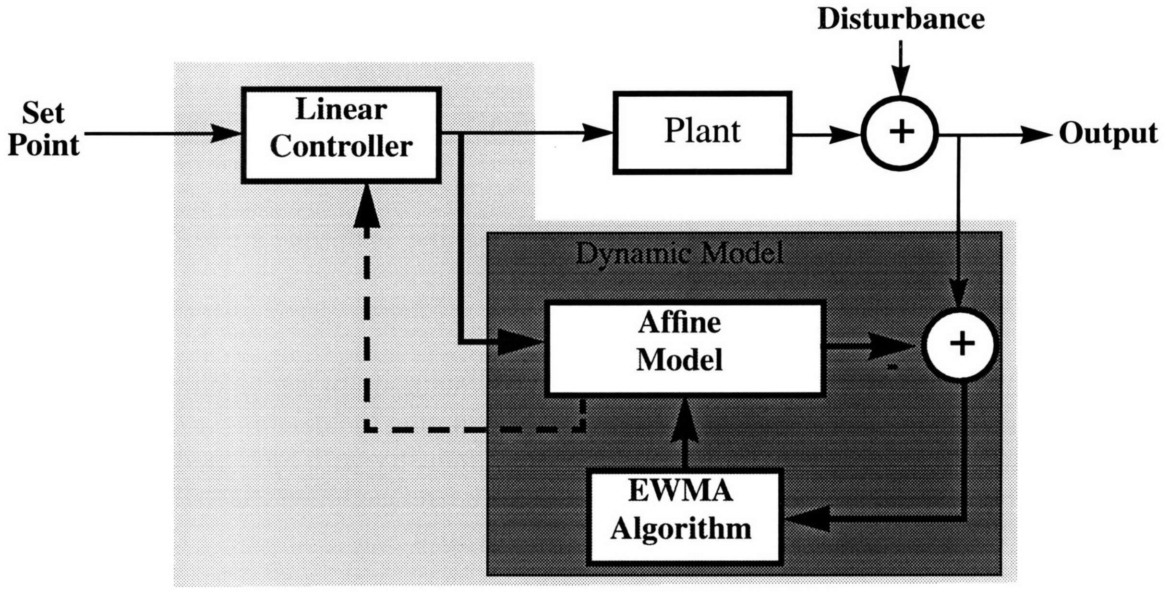
$$\boldsymbol{y}_m[n] = \boldsymbol{B}\boldsymbol{u}[n] + \boldsymbol{a}[n] \tag{3-2}$$

where $\boldsymbol{y}_m[n]$ is the model output vector, $\boldsymbol{u}[n]$ is the input recipe vector, $\boldsymbol{B}$ is a matrix of model coefficients, and $\boldsymbol{a}[n]$ is a vector of offset terms at discrete-time $n$. In most versions of this controller, only the offset term is adapted, while the gain coefficients $\boldsymbol{B}$ remain fixed. Recursive adaptation occurs by an EWMA update of the offset term, based on the error between model prediction and measurement:

$$\boldsymbol{a}[n] = \boldsymbol{W}(\boldsymbol{y}_p[n] - \boldsymbol{B}\boldsymbol{u}[n]) + (\boldsymbol{I} - \boldsymbol{W})\boldsymbol{a}[n-1]. \tag{3-3}$$

Here $\boldsymbol{a}[n-1]$ is the offset term used on the previous run. The selection of the diagonal weight matrix $\boldsymbol{W}$ is based on consideration of both noise and sampling, similar to selection of the weights in the open-loop control methods. After the EWMA makes an update to the process model, a linear controller is used to generate the next process recipe, which is a minimum distance solution for underdetermined systems and a linear least squares solution for overdetermined systems.

Figure 3-1. The EWMA Controller



An improvement can be made to the EWMA controller which uses a double exponentially weighted moving average as a measure of the offset term [Butler]. This type of controller, termed Predictor Corrector Control (PCC), uses an affine model similar to that of the EWMA controller. Recursive adaptation of the offset term is based on a double EWMA. The basic idea of the adaptation is to make a prediction of the offset term at discrete-time *n* by adding an EWMA of the *trend* (or slope) in the offset to an EWMA of the *value* of the offset. This prediction is used with the model to generate the next recipe as in the EWMA controller. The update of the prediction of the offset term takes place as follows.

Let:

$\boldsymbol{v}[n]$ be the EWMA of the *value* of the offset term,

$\boldsymbol{c}[n]$ be the measure of the current *trend,*

$\boldsymbol{s}[n]$ be the EWMA of the *trend* of the offset term, and

$\boldsymbol{a}[n]$ be the *prediction* of the offset term.

The equations for the update to the prediction term are:

$$v[n] = \boldsymbol{W}_1(\boldsymbol{y}_p[n] - \boldsymbol{B}\boldsymbol{u}[n]) + (\boldsymbol{I} - \boldsymbol{W}_1)\boldsymbol{v}[n-1] \tag{3-4}$$

$$\boldsymbol{c}[n] = (\boldsymbol{y}_p[n] - \boldsymbol{B}\boldsymbol{u}[n]) - \boldsymbol{v}[n] \tag{3-5}$$

$$\boldsymbol{s}[n] = \boldsymbol{W}_2(\boldsymbol{c}[n]) + (\boldsymbol{I} - \boldsymbol{W}_2)\boldsymbol{s}[n-1] \tag{3-6}$$

$$\boldsymbol{a}[n] = \boldsymbol{s}[n] + \boldsymbol{v}[n]. \tag{3-7}$$

The selection of the diagonal weight matrices, $W_1$ and $W_2$, is based on consideration of process noise, shifts and drifts. This controller is an improvement over the EWMA controller in that preferential choice of the $W_2$ matrix can either more closer track changes in the process drift (and shifts) with large diagonal entries or use an adaptation strategy more like the EWMA controller by using small diagonal entries. For a process that is drifting linearly with time (at a constant rate) the EWMA estimate of the slope will converge to that value and always be added to the prediction. This will cause any error due to a drift to be removed in the steady state.

### 3.1.2 Advantages and Disadvantages of the EWMA and PCC Controllers

The EWMA Controller provides good control for processes which have small variations over time. Stability for the Single Input Single Output (SISO) case is well understood and has been shown to be possible over a large range of model mismatch [Ingolfsson]. In addition, the EWMA Controller is designed around a statistically based filter which can be tuned to a given process (filter noise in the best possible way while minimizing errors due to real changes in the process). In particular, the performance of the EWMA controller (response to real process shifts and drifts while minimizing the response due to noise) is quite good. This is especially true for systems which have slow dynamics buried in large amounts of noise. The PCC controller has similar advantages but with improved responsiveness to sudden changes, and can produce zero steady state error due to drifts at the expense of overshoot and increased sensitivity to noise. Other advantages to these types of controllers, suggested in [Moyne1], are the ability to allow user preferences as to which controls in the process recipe should be more readily adjusted when the process is underdetermined and which errors in the outputs should be more readily compensated for when the system is overdetermined. The simplicity of the linear controller allows effective methods to be used to discretize the process inputs [Moyne1].

An apparent disadvantage of these controllers is that generating an effective affine model is often difficult or impossible. New sensors (e.g. the NOVA Sensor) are expected to be able to generate a large amount of intra-die as well as within-wafer measurements on every run. Two problems can be foreseen in using this information with the EWMA controller. First, measurements of within-wafer and intra-die variation are often nonlinear. However, it may be possible to overcome this problem using an EWMA controller which utilizes a polynomial model. The second problem is that the large number of measurements contain redundancies. EWMA controllers which utilize affine or polynomial models generally use some "measure" (such as the average removal rate over several sites on the wafer and the standard deviation of these removal rates) in order to create a statistically accurate model. These models generally do not take advantage of "features" in the data (e.g. pattern dependent changes in removal rate) unless a well formulated model of these effects is known. Therefore, it would seem that high dimensional systems which are difficult to model are not easily incorporated into the EWMA Controller. As will be discussed in Section 3.4, it is hoped that an EWMA framework can be combined with neural network methods to improve this type of controller.

## 3.2 Classical Linear Feedback Control

Another approach to providing closed-loop feedback control is robust linear feedback control.

Before the introduction of robust control, linear feedback control on the process level or plant level proved inferior to SPC methods in tightening the bounds on manufacturing processes [Koenig]. This led to the favor of SPC methods in manufacturing usage. An advantage of linear feedback control is that a vast amount of research has been done to demonstrate stability, ensure performance and more recently guarantee stability and adequate performance in the face of uncertainty (robust control). Research is currently being done to demonstrate the effectiveness of robust control in semiconductor processes [Baras]. These techniques tend to have difficulty, similar to the EWMA Controller, in their portability to high dimensional systems. More research into these control methodologies needs to be done for semiconductor process control in general.

## 3.3 Stochastic Dynamic Programming

The third method we consider is that of Dynamic Programming (DP). This framework provides several advantages. First, it is stochastically based, which means that the implicit randomness in the CMP process can be incorporated. This allows principles from estimation theory to be utilized in estimating the process state in the presence of noise. Recent advances made in statistical metrology for the CMP process [Chang1] which allow one to quantify the amount of variation wafer-to-wafer, within-wafer, and within-die can be used to obtain the statistically optimal control action to minimize the overall error. Second, DP allows for multi-step trajectory control. This property provides several advantages. For example the controller can be designed to return the process to target within a given number of steps while minimizing input volatility. In addition, rate prediction based on pad age can be incorporated to obtain optimal performance. Third, DP is formulated around a problem in which decisions are made in stages. In CMP run by run control, the measurements and control actions are often taken either on a lot to lot basis (e.g. once every 10 wafers) or a wafer to wafer basis. This is necessary for tools which have slow measurement times and a lack of in-situ sensors. DP is well suited for dealing with the discrete nature of CMP run by run control.

Although DP provides many advantages, several issues must be resolved with its implementation. First, DP requires a stochastic model of the process to be controlled. This could be very simple for some problems but very complicated for others, depending on the nature of the process noise and the length of the finite horizon used in the DP algorithm (to be described). In light of these issues, we will now highlight the major features of DP and controllers which utilize this methodology.

### 3.3.1 A Brief Outline of Dynamic Programming

As outlined in [Bertsekas], Dynamic Programming problems are formulated in the **Basic Problem** format.

Given a discrete-time dynamic system:

$$x_{k+1} = f_k(x_k, u_k, w_k) \tag{3-8}$$

where $x_k$ is the state vector, $u_k$ is the control vector, and $w_k$ is a random disturbance, the problem is to find an admissible control policy, $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, where $\mu_i$ is the set of control vectors which are dependent on the state $x_k$, that minimizes some cost function:

$$J_\pi = E\left\{g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, u_k, w_k]\right\}, \tag{3-9}$$

where $g_k$ is the state transistion cost from state $x_k$ to state $x_{k+1}$.

Once the problem is formulated in this fashion, the DP algorithm essentially works backward to find an optimal solution. Starting at the N-1th interval, the optimal control inputs are determined to take the system from $x_{N-1}$ to $x_N$, with the minimum cost $J_{N-1}$:

$$J_{N-1}(x_{N-1}) = E\{g_N(x_N) + g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\} \tag{3-10}$$

and subject to the constraint

$$x_N = f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}). \tag{3-11}$$

This is repeated for each step back until the optimal solution, $\pi$, is determined. The minimization problem at each step depends on the nature of the problem. For example, if the cost function is a mean squared error for a linear process, then the minimization is the solution of a linear problem. This type of problem is extremely fast, but does not exploit the power of the DP algorithm.

### 3.3.2 An Example Dynamic Programming Architecture

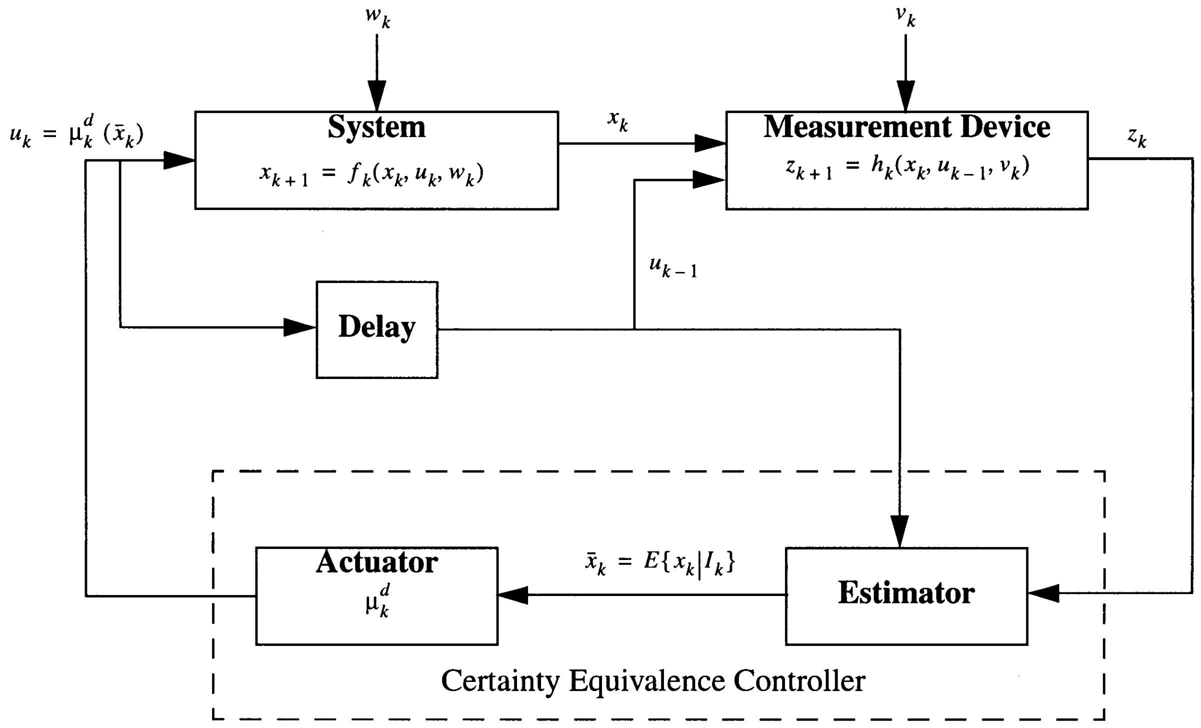
Naturally, the next question becomes what type of architecture is necessary to implement a DP control algorithm. Several architectures exist, including Certainty Equivalence Control (described below), Open-Loop Feedback Control, and Adaptive Control. As an example, we outline the Certainty Equivalence Control (CEC) architecture, shown in Fig. 3-2. This controller generates the optimal control vector $\mu_k^d$ based on an information state, $I_k$, which is the estimate of the state based on the last input and the expected value of the state through the measurement device. At each iteration the actuator recomputes the optimal control law based on the system cost function using the estimate of the state from the estimator.

Notice that each iteration requires the solution of an optimization problem in order to minimize the cost function and generate the optimal controls. These may be solved ahead of time and a look-up table generated to speed the operation of the controller (a neural network provides an excellent look-up table). A second issue is the loss of information in the state estimate when using the expected value.

Open-loop Feedback Control improves on this architecture by forming the distribution of the state given the information state and carries this through to the generation of the optimal control policy. The difficulty with this is the calculation of the conditional distribution of the state given the information state.

We defer our discussion of the Adaptive Control form of Dynamic Programming until Section 3.4 in which we focus on Artificial Neural Networks.

**Figure 3-2. The Certainty Equivalence Controller**



$w_k$
$v_k$
$u_k = \mu_k^d(\bar{x}_k)$
System
$x_{k+1} = f_k(x_k, u_k, w_k)$
$x_k$
Measurement Device
$z_{k+1} = h_k(x_k, u_{k-1}, v_k)$
$z_k$
$u_{k-1}$
Delay
Actuator
$\mu_k^d$
$\bar{x}_k = E\{x_k|I_k\}$
Estimator
Certainty Equivalence Controller

### 3.3.3 Advantages and Disadvantages of the DP Framework

The DP formulation has several advantages. First, nearly any cost function can be used. For the CMP process, this cost could simply be a weighted sum of nonuniformty and squared error of removal rate from target. Alternately, trade-offs between wafer-to-wafer, within-wafer, and within-die variation could be formulated, or additional goals such as slurry use or total time could be accommodated. In addition, input deviations from the last or even previous runs could be added to the cost structure. Second, the cost function is a probabilistic function. Thus, the inherent randomness with the CMP process for both nonuniformty and removal rate may be added to the controller and the expectation of this cost would be minimized. Third, the multistep nature of the problem formulation can easily allow recipe management on a lot to lot or wafer to wafer schedule which is a requirement for CMP control.

Posing the problem using the dynamic programming framework also causes problems. First, as alluded to earlier, it is not always possible to characterize the probabilistic distribution for the cost function or the state transition function [Bertsekas]. In these cases, theoretical minimization of the cost function, and hence the controls, may be impossible. Even for those cases in which it is possible to fully describe the probability distribution for the cost function and/or the transition function, it may not be a manageable task to determine the expectation for the cost function. For many multistep problems with a complex probability distribution, this task quickly becomes unmanageable. In response to this problem, techniques exist which use approximations of the moments and a finite number of steps (finite horizon) over which the optimal trajectory is determined [Maitelli, Ahmed]. Second, for many non-linear cost functions and/or state transition matrices, convergence of the cost function to a minimum is not always guaranteed. A solution to this has been proposed which uses an iterative technique reminiscent of that in many optimization routines whereby the finite horizon is scanned over iteratively with increasingly finer grid until a minimum which matches all grid points is obtained [Bojkav].

A final issue is that for systems with large state spaces, the number of state-control pairs that must be explored with the DP algorithm becomes unmanageable. The combination of Dynamic Programming with neural networks provides a large degree of flexibility in resolving this problem; this will be discussed in the following section.

## 3.4 Artificial Neural Networks

The second class of controllers we consider are those which use Artificial Neural Networks (ANNs). Neural networks for control often fall into a subclass of the so-called intelligent controllers. Several works attempt to differentiate intelligent control from adaptive control and deterministic feedback control [Meystel, Passino, Åström2]. We will not dwell on this distinction, but will consider controllers which incorporate "intelligent" neural network ideas. We will also illustrate how neural network controllers may be combined with the DP algorithm to utilize the features of both control types.

We begin by outlining the most popular neural network structure in Section 3.4.1. Several examples of neural network controllers are given in Section 3.4.2. Section 3.4.3 discusses the respective advantages and disadvantages to each type of neural network controller for possible use in the CMP process.

### 3.4.1 A Brief Review of Neural Networks

The use of neural networks has exploded since algorithms for training emerged in the 1980's [Åström1, DARPA]. Several different types of neural networks exist and have been applied to control problems. However, we will focus on one type of neural network, the multi-layer perceptron (MLP). This neural network consists of a collection of neurons, each of which forms a linear combination of its inputs and maps this through a nonlinear compression function. For example, the output of the jth neuron of the kth layer is:

$$y_j^k = f\left(\sum_i w_{ij}^k y_i^{k-1}\right), \tag{3-12}$$

where $y_i^{k-1}$ is the ith output from the previous layer ($y_i^0$ is the ith input to the neural network) and $w_{ij}^k$ is a scalar multiplication factor, or "weight", from the ith input to the jth output of the kth layer. The nonlinear "squashing" function is generally of the form:

$$f(x) = \frac{1}{1+e^{-ax}}. \tag{3-13}$$

It has been shown that by using this form a large class of continuous functions can be approximated arbitrarily closely with a proper selection of the weights. The appropriate number of weights for a particular problem varies and the values of these weights are typically determined using numerical methods, such as gradient descent, conjugate gradient descent, and gauss-newton methods [Rumelhart, Gill]. These methods provide a way to train a neural network to "learn" a generalized nonlinear mapping from a set of input data to a set of target data.

### 3.4.2 Examples of Neural Network Controllers

Several architectures exist for implementing neural networks into control systems. According to [Werbos1], neural network controllers are generally of one or more of the following five forms:

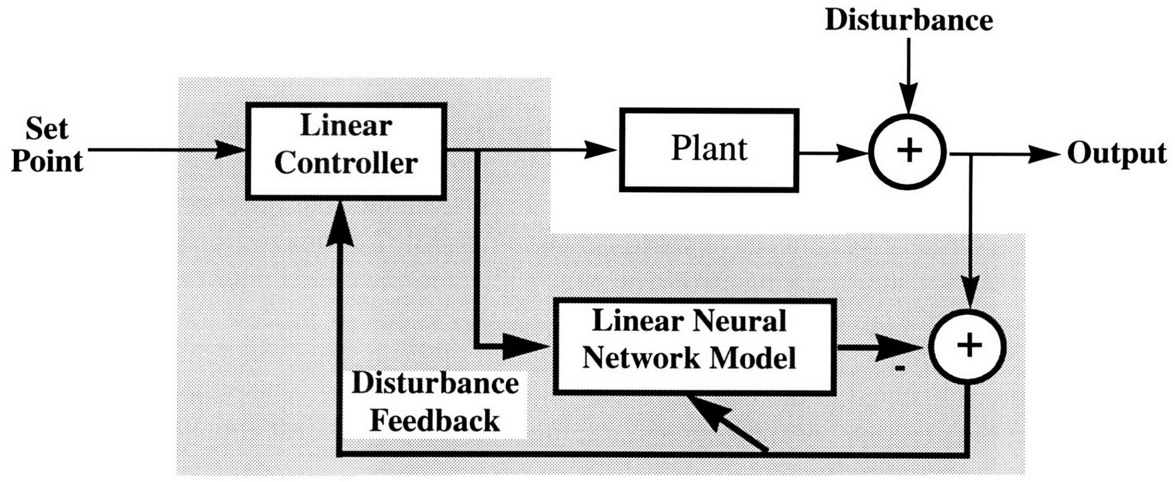
- Supervised control, where neural nets are trained on a database that contains the "correct" controls to use in sample situations.
- Direct inverse control, where neural nets directly learn the mapping from desired trajectories to the control signals which yield these trajectories.
- Neural adaptive control, where neural nets are used instead of linear mappings in standard adaptive control.
- The backpropagation of utility, which maximizes some measure of utility or performance over time, but cannot efficiently account for noise and cannot provide real-time learning for large problems.
- Adaptive critic methods, which may be defined as methods that approximate dynamic programming.

Supervised learning appears in almost all neural networks since nets are generally trained on a proper data set initially or at some point. Therefore, we will not explicitly consider controllers of this type, but the interested reader is referred to [Hofer]. Direct inverse control has been shown to provide adequate performance in several systems but generally there is a problem with systems with a non-unique mapping (more inputs than outputs) [Jordan]. Therefore, we will not consider controllers of this type either. Neural adaptive controllers attempt not only to provide adequate control of a plant while in its normal operating condition, but also to compensate for unmodeled changes. Two examples of this type of controller are given in Sections 3.4.2.1 and 3.4.2.2. An example of the backpropagation of utility is given in Section 3.4.2.3. Finally, an example of an adaptive critic controller is given in Section 3.4.2.4.

#### *3.4.2.1 Model Reference Control - Linear Neural Network*

Probably the most basic configuration of neural network controllers uses a linear neural network (this is really nothing more than a simple linear model) as the process model. A linear controller (solver) of the type described in [Boning2] is then used to calculate the controls to the system. The linear model is updated by training the neural network as new input-output pairs are generated as the process is run. This system, shown in Fig. 3-3, is derived from the pioneering work (see [Åström1, Narendra1]) in linear adaptive control methods. Other linear neural network controllers are given in [Widrow, Nguyen] and a critical review of these systems is given in [Narendra2].

**Figure 3-3. Adaptive Linear Network Controller**



These systems have the ability to account for unmodeled system dynamics as well as model error. Linear network models are useful in that the updates to the process model and control actions can be made extremely fast, thus making real-time as well as run-by-run control possible applications. These systems can be shown to be asymptotically stable, which is extremely important for many processes [Narendra1]. The disadvantage to a controller of this type is that it is unable to fully adapt to unmodeled nonlinear system dynamics.

### *3.4.2.2 Model Reference Control - Neural Adaptive Controller*

The neural network model reference control replaces both the controller and the process model with neural networks. An in-depth description of this system given in [Narendra3, Chang2] demonstrates the relationship between this hierarchy and that of stable adaptive control systems. Systems of this type can provide adaptive model based control with excellent performance for time-dependent systems. An interesting result is that the asymptotic stability of this nonlinear architecture can be shown when the neural networks squashing functions are radial basis functions. It is hoped that these properties can be extended to MLP ANNs as well.

Several advantageous features can be identified in this type of architecture. For example, the neural network models can be set up to represent nonlinear moving averages (MA), nonlinear auto-regressive averages (AR), or nonlinear ARMA models. The user can discover the time series dynamics of the system by observing what the neural network learns. This provides a built-in system identification utility as well as an adaptive nonlinear controller.

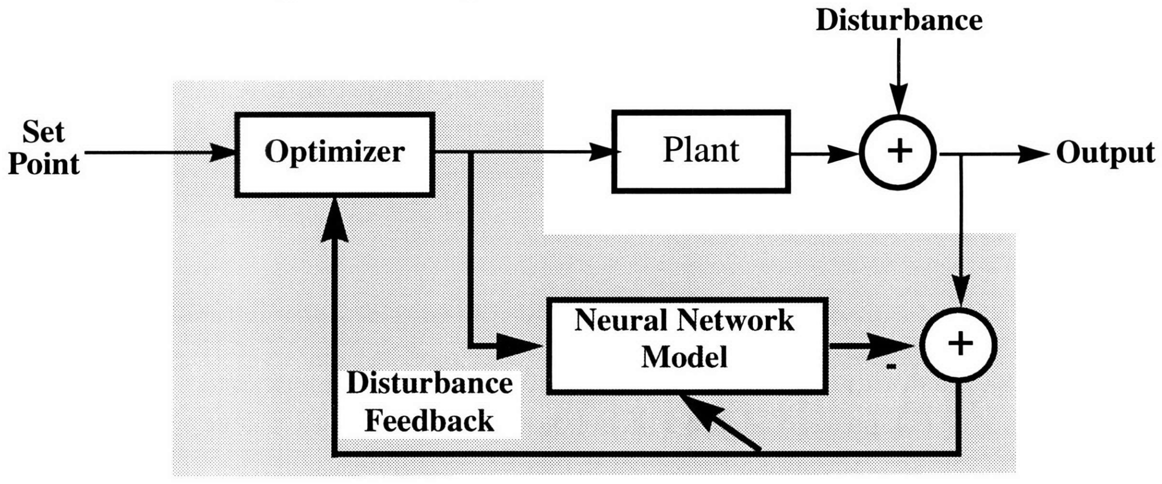
### *3.4.2.3 Backpropagation of Utility - Model Based Predictive Control*

In general, the backpropagation of utility may update a controller with a fixed architecture, such as in [Jordan,Morari,Werbos2], or may be used to directly generate the control action. An example of the latter type of backpropagation of utility is termed Model Based Predictive Control

(MPC) and is the type we consider here. This structure is shown in Fig. 3-4. The optimization routine in this method uses a cost function subject to constraints to determine the next control action. The cost function is often mean-squared-error based on the neural network process model. It should be noted that the process model is not required to be a neural network, but often is because of the ANN's ability to model arbitrary nonlinear systems. This architecture has been used in the chemical process industries to improve the dynamic response of complex systems [Werbos2].

The ANN process model is initially trained using a Design of Experiments (DOE) or past process data. Updates to the ANN process model can be made using input-output patterns generated on-line. This adaptive technique is thought to improve performance but questions regarding the stability of these types of systems have yet to be answered. MPC optimizes the process inputs such that the mean squared tracking error over time subject to the constraint of bounded inputs is minimized. At each iteration, the process is optimized for the next N process steps using the neural network process model. The controls are then applied to the system, the output measured, and the optimization is repeated in the next run. Many efficient and effective routines exist for properly optimizing the process. For many processes, this optimization is fast enough to be implemented at every process cycle.

**Figure 3-4. Adaptive Neural Network Controller**



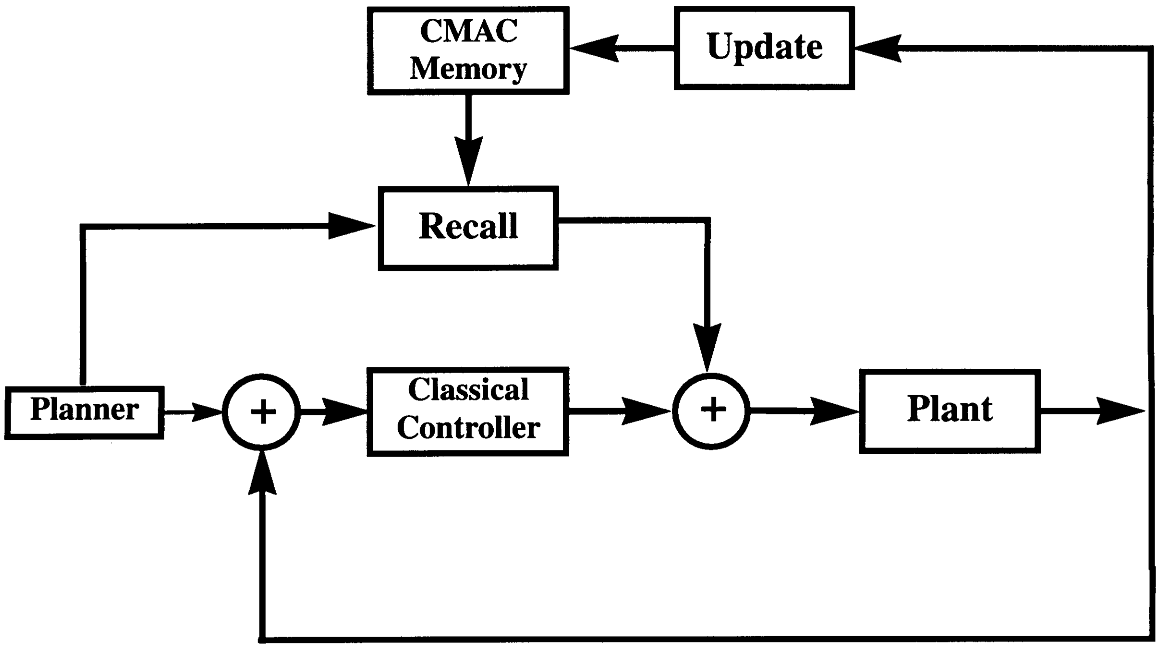
*3.4.2.4 An Adaptive Critic - The Cerebellar Model Articulation Controller*

Adaptive critic controllers are among the most complex neural network controllers. These controllers contain methodologies for approximating dynamic programming, which is the only exact method for finding the optimal solution of a nonlinear process in noise [Werbos2]. The problems of DP, as outlined in the previous section, are mainly computational expense and probabilistic modeling. Therefore approximation techniques are needed.

The basic idea of the architecture is to use a neural network as a critic, which provides an approximation to the true cost function of the DP algorithm, and an action network, which is adapted to minimize the approximate cost function provided by the critic network.

A popular adaptive critic neural network controller is the Cerebellar Model Articulation Controller (CMAC) [Franklin, Kraft, Sofge]. The critic network is an associative memory ANN which is different from an MLP network in that the typically large number of weights used in the MLP network is reduced by using a pseudo-random mapping for the hidden layer of the neural network. An advantage to using this type of structure is that it can be updated quickly, due to the small number of weights, thus making it possible to learn in real-time. The action network for this type of adaptive critic is a classical feedback controller.

**Figure 3-5. The CMAC Controller**



The disadvantage of this type of controller is that although it is adaptive, it does not utilize the full power of DP. For an example of networks which take advantage of this, see [Sofge].

### 3.4.3 Advantages and Disadvantages of ANN Controllers

Having explored several types of neural network controllers, we now consider which of these architectures holds promise for the CMP process. To begin this task we reconsider what each of these has to offer in terms of CMP process control.

The linear adaptive controller provides many advantages. These are well known in adaptive linear control theory [Åström1]. First, stability can be guaranteed. In addition, initial findings, as given in [Boning1], indicate that the CMP process is approximately linear over a large operating range, so this type of system might provide excellent control. However, increasing demands including patterned wafer control, sampling of measurement and control actions, and additional constraints and control goals motivate the need for nonlinear and more flexible control approaches.

The backpropagation of utility seems to be an excellent candidate for the control of CMP. This mainly comes from the versatility of the structure. The optimization routine provides a large amount of flexibility as to what is defined as the utility function. Such practical issues as input bounds, input movement trade-offs, output error trade-offs, and input discretization could fit into this architecture through the optimization routine. The increased computation time is not an issue with current technology CMP tools, as these have a relatively long wait time in-between wafer cycles. Even if throughput increases, the wafer to wafer time dynamics in the CMP process appear to be slow and thus periodic updating could be incorporated. This method can also encompass the introduction of improved sensor information such as optical thickness measurements provided by the NOVA sensor. The disadvantage of this type of control architecture is that if real-time sensors become available, real-time control could be difficult given the computation involved in the optimization step.

These issues make it appropriate to consider advanced methods in neural control. The adaptive critic method seems to offer the greatest flexibility and promise at the expense of complexity. This derives from the fact that the storage methods which serve as the memory for the critic network have complex ways in which the number of weights are reduced in order to make real-time control possible. In addition, the process of defining what the critic and action networks do and what their relation will be, combined with the complex methods (derived from DP) for utilizing the cost structure to update the action network, makes using this architecture more difficult. There also does not appear to be any guarantee of stability with this type of control. The potential benefits to using this type of control are great. The ability to optimize a cost function to update the action network using DP algorithms makes this method ideal for incorporating the inherent noise which is troublesome to the CMP process. In addition, unmodeled nonlinear dynamics can be "learned" on-line by the critic network and used to update the action network.

## 3.5 Summary of Prospective Control Approaches

From the background research outlined above, we have suggested that the EWMA and PCC Controllers provide a simple, yet effective, method to add feedback control to semiconductor processes. In addition, this framework allows for a smooth transition from open-loop to closed-loop control for industries that are already familiar with SPC methods. For these reasons, work on the EWMA Controller (and PCC controller) began some time ago and has been the central focus of much research. These controllers are also the central focus of this thesis. The multivariate EWMA Controller will serve as a baseline against which other alternatives can be judged. We will consider several issues involving the EWMA and PCC controllers. We will demonstrate stability criteria for these controllers and discuss practical issues relating to their successful implementation. An analytical derivation of the optimal weights to be used with these controllers will be shown for a process with a linear drift buried in white noise. Finally, we will discuss a self-tuning EWMA controller.

It appears that the DP algorithm is the best way to provide control in the presence of noise. This is a very important property for the CMP process which is plagued by noise and poor equipment reliability. The DP algorithm falls short when we arrive at high dimensional problems over a long horizon, due to the increasing computation time and the difficulty in providing a probabilistic model for such systems. Neural networks, on the other hand, offer many forms of controllers, some of which utilize paradigms from many well known control architectures. Therefore, ANN controllers will be the second focus of this thesis. Due to the stability problems with Direct Inverse Control and the apparent difficulty in addressing many practical issues with Neural Adaptive Control, we will avoid these controllers in this thesis. We will however provide a simple linear adaptive ANN controller and compare its performance to that of the EWMA controller. The backpropagation of utility seems to provide a great amount of flexibility through the use of a utility function. This allows many practical issues to be implemented easily in the structure. Therefore, we will consider one controller of this form which utilizes an ANN process model which is dynamically updated using an EWMA of the bias terms. A second form of this controller will also be considered which utilizes a fully adaptive nonlinear ANN model. The lack of speed and neglect of the advantages of DP based controllers lead us to the neurodynamic programming controllers. These controllers are left as future research along with the hope that these more complex controllers can significantly improve on current controllers by approximating dynamic programming with neural networks.

# Part II - EWMA Control Strategies

Part II explores EWMA based controllers in depth. It addresses several important aspects of EWMA controllers, including practical implementation issues, experimental results, stability, and optimal EWMA weight determination.

# Chapter 4

# The EWMA Run by Run Process Control Framework

We begin our study of run by run control with an in-depth look at the implementation of the EWMA controller on two CMP processes. We will first discuss several practical issues which may be addressed in the EWMA run by run control framework. Since semiconductor process control is complicated by practical implementation issues above and beyond the underlying control theory, these issues are critical to the successful implementation of control methods in existing semiconductor processes. Second, we will provide experimental results which verify the ability of the EWMA controller to greatly improve current CMP processes.

In Section 4.1, we briefly outline particulars of the CMP process, equipment model, and the EWMA control scheme used for the simulations and experiments throughout this chapter. Section 4.2 discusses the importance of input constraints and compares a fast heuristic routine for implementing input bounds to a more general optimization routine. A method for relative weighting of the movements of specific inputs is described in Section 4.3. In Section 4.4, we discuss the importance of properly handling discretization in the inputs of a Multi-Input/Multi-Output (MIMO) system and propose a heuristic which provides better results when compared to rounding all the inputs simultaneously. Section 4.5 describes the effects of control parameters and explores experimentally-based methods for determining optimal values of these parameters. Section 4.6 presents results from the experimental implementation of these methodologies on two CMP processes. Finally, in Section 4.7 we summarize this chapter and highlight areas where additional research and demonstration are needed.

## 4.1 Specifics of the CMP Process and the EWMA Controller Related to this Chapter

Recall from Chapter 2 that the product characteristics of concern are the removal rate and the within-wafer uniformity of that removal rate on a run by run basis. The control goal used here is to maintain a target removal rate and within-wafer nonuniformity in the face of equipment drifts and disturbances. Although the implementation of the EWMA controller used here uses a target for nonuniformty, the framework is not limited to this and more advanced linear programming techniques may be used to achieve targets for certain outputs (e.g. removal rate) and minimize or constrain others (such as nonuniformty). In this work, we will consider two CMP processes. The first CMP tool corresponds to an older tool with large amounts of process noise. The change in removal rate and nonuniformity for a typical uncontrolled or baseline oxide polish process (with a fixed recipe) was shown in Fig. 2-3. This run, and all those described in this chapter, were conducted using 8" silicon wafers with a thick blanket oxide deposition. For the first process, a central composite experimental design was conducted with Speed (20-40 rpm), Pressure (0-7 psi), Force (8-10 lb), and Profile (-0.9 to 0.9) as inputs. Second order polynomial regression models were constructed for removal rate and nonuniformty with adjusted R$^2$ of 89.7% and 76.9%, respectively. The response surfaces, shown in Fig. 4-1 as a function of two of the major variables, are nearly linear over the input space, justifying a linearization of the models for control. Therefore, each polynomial regression model was linearized around the operating point to generate the following multivariate model for the gradual mode run by run controller:

$$y_m[n] = Bu[n] + a[n], \tag{4-1}$$

where $y_m = \begin{bmatrix} RemovalRate \\ Non-Uniformity \end{bmatrix}$, $u = \begin{bmatrix} Speed \\ Pressure \\ Force \\ Profile \end{bmatrix}$, $B = K_1\begin{bmatrix} 25.27 & -3.33 & 89 & 3.92 \\ 7.25 & 9.31 & 15.08 & 2.45 \end{bmatrix}$, $a = K_2\begin{bmatrix} -134.85 \\ -80.96 \end{bmatrix}$, and constant scaling factors $K_1$ and $K_2$.

**Figure 4-1. Response Surfaces for the First CMP Process**

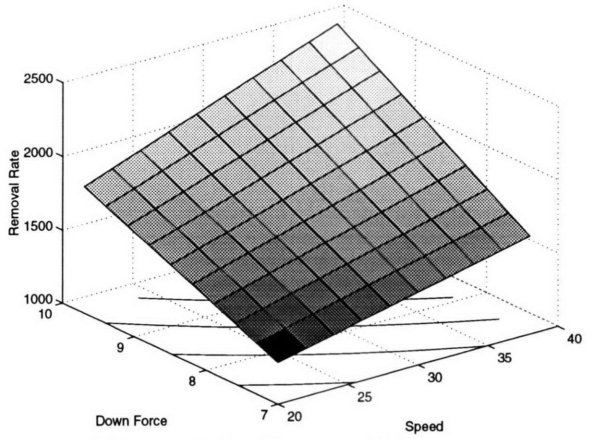


Figure 4-1a. Removal Rate vs. Force/Speed

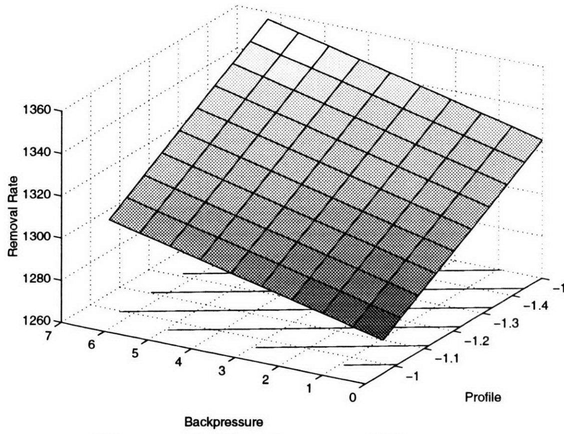


Figure 4-1b. Removal Rate vs. Pressure/Profile

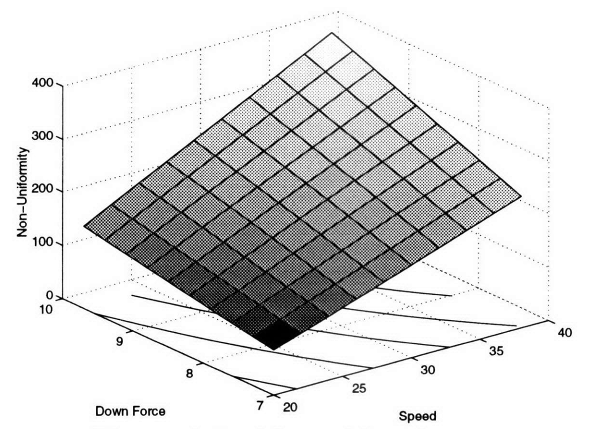


Figure 4-1c. Nonuniformity vs. Force/Speed

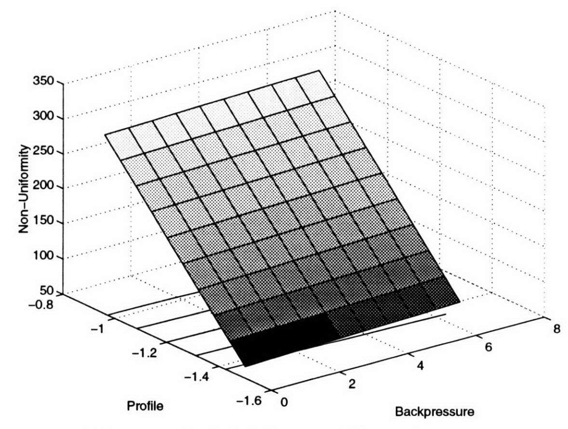


Figure 4-1d. Nonuniformity vs. Pressure/Profile

The second process is a Strasbaugh 6D S-SP CMP machine. A central composite design was conducted for this machine with Polish Pressure (7-9 psi), Backpressure (0-2 psi), Table Speed (20-30 psi), and Pad Profile (-1 to 1) as inputs. Second order polynomial regression models were constructed for removal rate and nonuniformity with adjusted $R^2$ of 96% and 82%, respectively. As seen in Fig. 4-2, the response surfaces for this process are nearly linear for removal rate, but contain nonlinearities for nonuniformity in the Backpressure and Pad Profile variables. Each polynomial regression model was linearized around the operating point to generate another multivariate model for the gradual mode run by run controller, where

$$\boldsymbol{y}_m = \begin{bmatrix} RemovalRate \\ NonUniformity \end{bmatrix}, \ \boldsymbol{u} = \begin{bmatrix} PolishPressure \\ BackPressure \\ TableSpeed \\ PadProfile \end{bmatrix}, \ \boldsymbol{B} = K_1 \begin{bmatrix} 43 & -3.2 & 7.2 & 2.4 \\ 0.92 & 14 & 0.59 & -22.1 \end{bmatrix}, \ \boldsymbol{a} = K_2 \begin{bmatrix} -179.42 \\ 2.43 \end{bmatrix}. \qquad (4\text{-}2)$$

The linear response surface for removal rate was very accurate (0.96 $R^2$) while the nonuniformity model was, as expected, relatively poor (0.56 $R^2$) due to the nonlinearities. The uncontrolled response of this machine is more typical of newer industrial tools. As described in Chapter 2, specifications for tight control of wafer-to-wafer uniformity are generally in the range of 3-5%, while within-wafer uniformity is typically maintained from 4-6%. Lifetime of a pad maintaining these specifications is generally on the order of 100-200 wafers.

**Figure 4-2. Response Surfaces of the Second CMP Process**

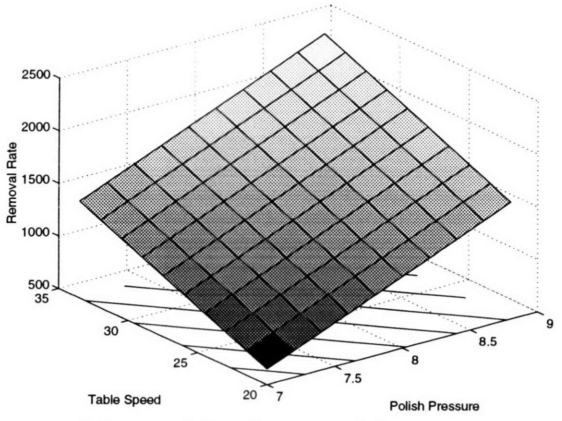


Figure 4-2a. Removal Rate vs. Speed / Polish Pressure

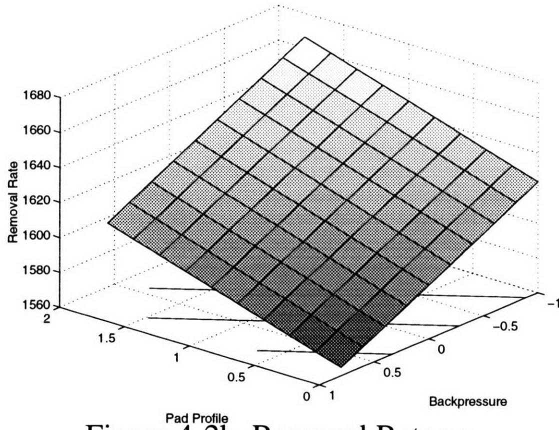


Figure 4-2b. Removal Rate vs. Backressure / Pad Profile

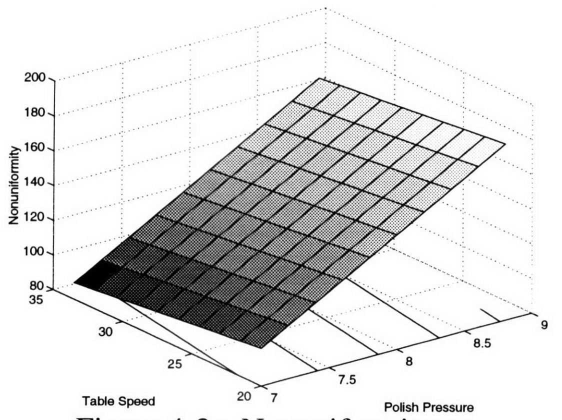


Figure 4-2c. Nonuniformity vs. Speed / Polish Pressure

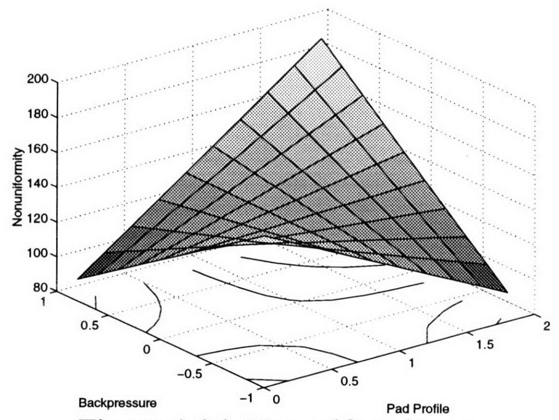


Figure 4-2d. Nonuniformity vs. Backpressure / Pad Profile

We now turn to describing specific aspects of the implementation of the EWMA controller described in Section 3.1.1. The first three practical issues revolve around recipe generation while the last issue is concerned with choosing an optimal EWMA weighting scheme in order to maximize the performance of the EWMA controller.

## 4.2 Input Constraints

The EWMA controller is prone to unstable behavior if there is significant mismatch between the dynamic model and the actual process [Ingolfsson]. In addition, linear models alone do not incorporate the limited input range inherent in actual equipment. Input constraints are therefore extremely important to ensure stability and localize equipment operation to the region where the controller model is known to be valid.

A general approach to implementing bounds on an inverse model controller is to solve a constrained linear or nonlinear optimization problem [Gill]:

$$\min_{\boldsymbol{u}[n]} \ \|\boldsymbol{u}[n] - \boldsymbol{u}[n-1]\| \ \text{, s.t. } \boldsymbol{u}_{max} \geq \boldsymbol{u}[n] \geq \boldsymbol{u}_{min} \text{ and } \boldsymbol{y}_d[n] = \boldsymbol{B}\boldsymbol{u}[n] + \boldsymbol{a}[n], \qquad (4\text{-}3)$$

where $\boldsymbol{u}[n]$ is the input vector at discrete-time $n$, $\boldsymbol{u}_{min}$ is the vector of input minima, $\boldsymbol{u}_{max}$ is the vector of input maxima, and $\boldsymbol{y}_d[n]$ is the vector of desired outputs. This problem statement ensures that the change in input space is minimal when multiple recipes are found (that is, multiple feasible recipes which satisfy $\boldsymbol{y}_d[n] = \boldsymbol{B}\boldsymbol{u}[n] + \boldsymbol{a}[n]$). If a solution cannot be found the optimization routine reverts to minimizing the the mean-squared error from target:

$$\min_{\boldsymbol{u}[n]} \ \|\boldsymbol{y}_d[n] - (\boldsymbol{B}\boldsymbol{u}[n] + \boldsymbol{a}[n])\| \ \text{ such that } \boldsymbol{u}_{max} \geq \boldsymbol{u}[n] \geq \boldsymbol{u}_{min}. \qquad (4\text{-}4)$$

Although this method is extremely general (the cost function to be minimized need not be linear), it is computationally expensive. A heuristic variation on linear programming, typical of ad hoc methods used to implement bounds, is used here as an alternative to the full optimization described above. In Recursive Bound Pinning (RBP), a recipe is first determined and then checked for any inputs which exceed their boundaries. The exceeding variable with the greatest effect on the output (largest coefficient) is fixed at the exceeded bound and removed from the system of equations to be solved. A new recipe is generated (as an unconstrained least squares problem) and the process is repeated until a solution entirely within the bounds is found. This method is faster than the general optimization routine since the number of steps to determine a "good" solution is limited to the number of inputs which can be fixed. Simulations have been performed which demonstrate that this is a simple effective method for generating bounded recipes [Boning2, Moyne1].

## 4.3 Input Weights

When solving linear equations in the underdetermined case, the minimum distance solution implies that variables which affect the output the most will be modified most, such that minimal input variation will occur. However, the relative size of each input also affects whether or not that input is moved more or less. Variables of larger sizes tend to move through their allowable ranges much less than inputs with small magnitudes. Given a set of inputs with corresponding maxima and minima, the movement of each variable is not necessarily determined by how much a change (as a percent of its allowable range) in that input affects the output. Rather, the absolute size of a variable interferes with the notion that the "strength" or "gain" of an input increases proportional to the total amount that the output can be affected by an input.

A second issue regarding input movement occurs when engineers wish to "weight" the movement of certain inputs over others. This is important when controller models are developed using a DOE. Certain inputs may contribute more to model error than do others. Therefore it is desirable to move those inputs which contribute the most to model error the least. Another situation which calls for input weighting arises when different inputs are allowed to move only in discrete steps; this will be discussed in Section 4.4.

### 4.3.1 Range Normalization

In order to minimize the change in input space, relative to the allowable range of each variable, each input is shifted by its midpoint and scaled to range between plus and minus one. It is important when doing this that the underlying linear equation (that is, the model) is not disturbed; the coefficient matrix must be scaled and the constant term adjusted:

$$y = Bu + a = B(u - u_{mid}) + Bu_{mid} + a \tag{4-5}$$

$$y = (BS)(S^{-1}(u - u_{mid})) + (Bu_{mid} + a) = \tilde{B}\tilde{u} + \tilde{a} \tag{4-6}$$

$$\text{where } S = \begin{bmatrix} scale_1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & scale_n \end{bmatrix},\ \tilde{B} = BS,\ \tilde{u} = S^{-1}(u - u_{mid}),\ \text{and } \tilde{a} = (Bu_{mid} + a). \tag{4-7}$$

In this space all the variables have equal ranges and average sizes, thereby allowing the gain of each variable to determine its movement, not its absolute size

### 4.3.2 Input Weighting

Once the input space is normalized we can directly control the relative importance of each input with the same procedure used to normalize the input space. The relative amount each variable moves in its range may be adjusted by changing weights on the inputs. The relative size of the variable is changed by multiplying the input vector, *u*, by a scaling matrix, *V*, as follows:

$$y = \tilde{B}\tilde{u} + \tilde{a} \tag{4-8}$$

$$y = (\tilde{B}V^{-1})(V\tilde{u}) + \tilde{a} = \hat{B}\hat{u} + \tilde{a} \tag{4-9}$$

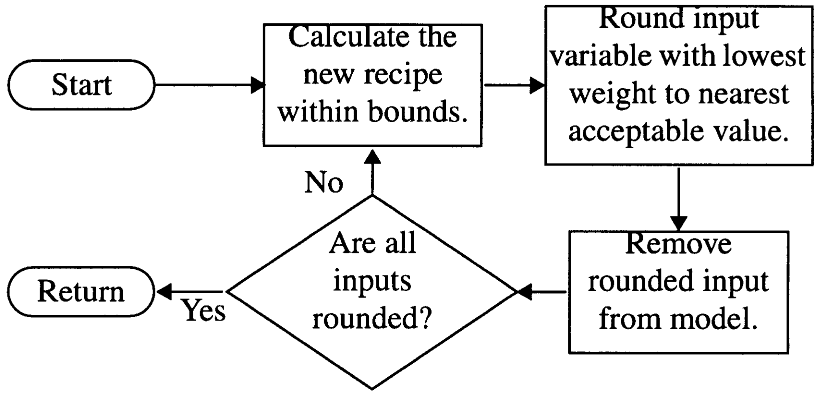
$$\text{where } V = \begin{bmatrix} v_1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & v_n \end{bmatrix},\ \hat{B} = \tilde{B}V^{-1},\ \text{and } \hat{u} = V\tilde{u}. \tag{4-10}$$

The solution which minimizes the change in *u* is altered by the newly scaled inputs. Inputs which have high weights have their distances shrunk and therefore the minimum change solution will move these inputs the most in their ranges.

## 4.4 Discretization

When implementing control on systems with less precision than the system on which the recipes are generated, discretization becomes an important issue. Although simply rounding the inputs seems like a reasonable method, it is not necessarily the best. This is especially true when there is low resolution in one or more inputs. Once again, it would be optimal to resort to a general optimization routine that intelligently searches the discrete points around an optimal solution. As with our method for implementing bounds, we consider a fast heuristic-based method typical of that used to implement discretization. The heuristic rounds the inputs one at a time as shown in Fig. 4-3. On each iteration, one input is locked to a discrete value and the system of linear equations is re-solved for the remaining free variables. We will consider two variations of this heuristic.

**Figure 4-3. Recursive Discretization**


Start
Calculate the new recipe within bounds.
Round input variable with lowest weight to nearest acceptable value.
Remove rounded input from model.
Are all inputs rounded?
No
Yes
Return

The first method, termed Statistical Discretization, is statistically based. In this method the order in which the inputs are removed from the system of equations is based on their statistical variance. Variables which contribute most to linearized response surface model error are locked last since locking these variables first would cause errors to propagate through the resolving process and thus have a greater effect on the output.

A second approach, termed Delta Discretization, considers how much a discrete change in each input would change the output of the system. Discrete changes in the outputs, caused by discrete changes in one input, $u_i$, of the system of equations $y = B_i u_i$ are given by $\Delta y = B_i \Delta u_i$. One would then lock the input, $u_i$, which, when changed by $\Delta u_i$, would cause the largest $\Delta y$. This would then allow other inputs which have a smaller $\Delta y$ to compensate for the error incurred by fixing the first input.

The following simulations compare the effectiveness of the different methods. It is important to note that the level of noise in the system may actually be greater than the errors caused by rounding the inputs. In these cases it is reasonable to simply round the inputs because the recursive methods increase the computational complexity by a factor slightly less than the number of inputs. Several simulations were performed and for the four input case the relative computation times for no rounding, simultaneous rounding, Statistical Discretization, and Delta Discretization are shown in Table 4-1 as multiples of the time required with no discretization.

**Table 4-1. Simulation Time Comparison (Ratio of Method Time to Time With No Discretization)**

| Method | Relative Time |
|---|---|
| No Discretization | 1 |
| Simple Rounding | 1.16 |
| Statistical Discretization | 3.61 |
| Delta Discretization | 3.76 |

The mean-squared-error between the output of the system with undiscretized inputs and the output of the system with discretized inputs for each method is listed in Table 4-2.

**Table 4-2. Mean Squared Error (Discrete vs. Non-Discrete) for 5% Constant Model Errors**

| Method | RR | NU |
|---|---|---|
| Rounding | 4.99 | 0.25 |
| Statistical Discretization | 4.26 | 0.14 |
| Delta Discretization | 2.59 | 0.23 |

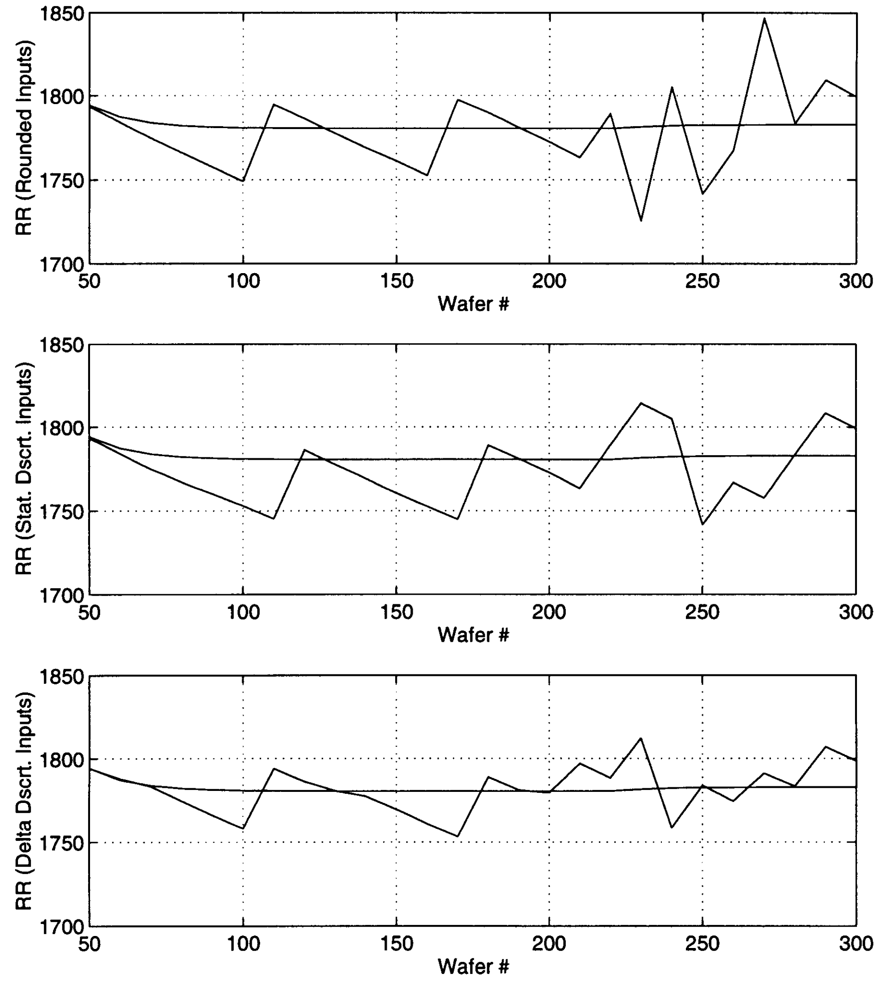
Further simulations were performed with an equipment model which had different amounts of model mismatch in different coefficients. The amount of mismatch used in the simulations was inversely proportional to the input weights. This was done so that the statistical discretization method would provide its best response, since it was designed with this assumption. We see from Table 4-3 that the Delta Discretization method still had the best MSE in removal rate, though both methods provide a better result than rounding.

**Table 4-3. Mean Squared Error (Discrete vs. Non-Discrete) for Proportional Model Errors**

| Method | RR | NU |
|---|---|---|
| Rounding | 5.73 | 0.25 |
| Statistical Discretization | 4.26 | 0.13 |
| Delta Discretization | 1.98 | 0.24 |

While difficult to see in the input trajectories, these results can be more readily observed in the outputs. Fig. 4-4 shows the removal rate output of each case along with the undiscretized output. From these we can see that the purely rounded inputs cause the largest deviation from the undiscretized case. The deviation caused by the Statistical Discretization is comparable to that of the Delta Discretization, which provides the best match to the undiscretized output.

**Figure 4-4. Output from Simple Rounding of Inputs vs. the Output from Undiscretized Inputs**



## 4.5 EWMA and PCC Weighting Parameters

As described in Section 3.1.1, the EWMA and PCC controllers are statistically based controllers. The EWMA controller utilizes an affine model of the process whose offset term is updated with an EWMA. The PCC controller uses a prediction of the offset term based on an EWMA of the value of the offset term, as well as an EWMA of the slope of the offset term. Both of these controllers filter process noise from the measure of the offset term while allowing the controller to respond to real changes in the process. In this section we demonstrate that there are optimal values for the smoothing parameters in these types of controllers by considering effects of the filter weight parameters of the EWMA controller on the controlled output (for both a shifting process as well as a drifting process). These effects are central to understanding the characteristics of these controllers, and provide insight into the determination of stability criteria and optimal weighting schemes for each of these controllers. We will also show that it is possible to empirically determine the optimal weighting scheme for the EWMA controller by considering the situation where we have a fixed amount of model error, additive output noise and linear drifts in the outputs.

For these simulations we will assume a SISO plant appropriately modeled by an affine model with a linear drift, $\delta$, a random shift, $\tau[n]$, and independent and identically distributed (i.i.d.) gaussian noise, $r[n]$:

$$y_p[n] = \alpha + \beta u[n] + \delta n + \tau[n] + r[n]\,. \tag{4-11}$$

The EWMA controller determines the inputs for the next run of a process by solving the linear control model,

$$y_m[n] = Bu[n] + a[n] \tag{4-12}$$

for $u[n]$ such that the output, $y_m[n]$, equals the desired output $T$. Recall that this is a dynamic model where $a[n]$ is updated according to the EWMA equation

$$a[n] = w(y_p[n] - Bu[n]) + (1 - w)a[n-1]\ . \tag{4-13}$$

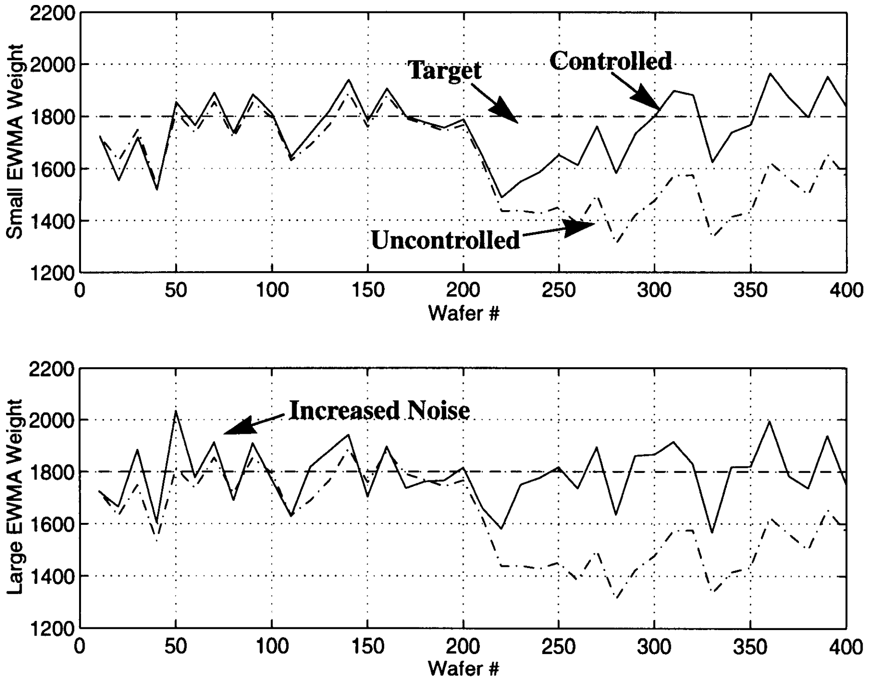
The performance is strongly dependent on the proper choice of the EWMA weight, $w$. In fact, if one is not careful, the controller may become unstable. This is determined by the degree of model mismatch and the size of the EWMA weight as described in [Ingolfsson]. The EWMA weight controls how much the constant term of the controller model is changed in response to the most recent output, $y_p[n]$. Several works explore methods for properly choosing $w$ such that an EWMA tracks an uncontrolled (open loop) output in an optimal sense [Lucas, Crowder1, Crowder2]. In those works, no control action is taken while the process is being monitored with the EWMA. Control actions are only taken when the EWMA of the process moves outside a given range. The optimal $w$ for the open loop situation does not necessarily correspond to the optimal value of $w$ when control actions are performed on a run by run basis. This is because, in general, using the EWMA estimate to generate control actions adds noise and changes the underlying behavior of the process. It will be shown that the optimal values for tracking a process and controlling a process on a run by run basis are different.

In order to understand how one might choose a value for $w$, it is important to consider first how $w$ affects the controlled output. A high value of $w$ increases the impact of the current model error (difference between the model and the actual output, $y_p[n] - Bu[n]$ ) on the control action. Therefore we expect a high value of $w$ to cause a faster update of the control model and thus provide a better response to true disturbances. On the other hand, if there is noise in the process a high value of $w$ would also cause control actions to increase the variance in the controlled output. Small values of $w$, on the other hand, would increase the smoothing of the previous model errors, and the control actions would be less swayed by individual changes in the output. This results in less rapid control over true disturbances but also less sensitivity to noise. Therefore it would seem that given a fixed amount of noise and expectations on the disturbances, one could choose an optimal value for this parameter.

This idea may be used to analyze the response of the EWMA controller to shifts and drifts.

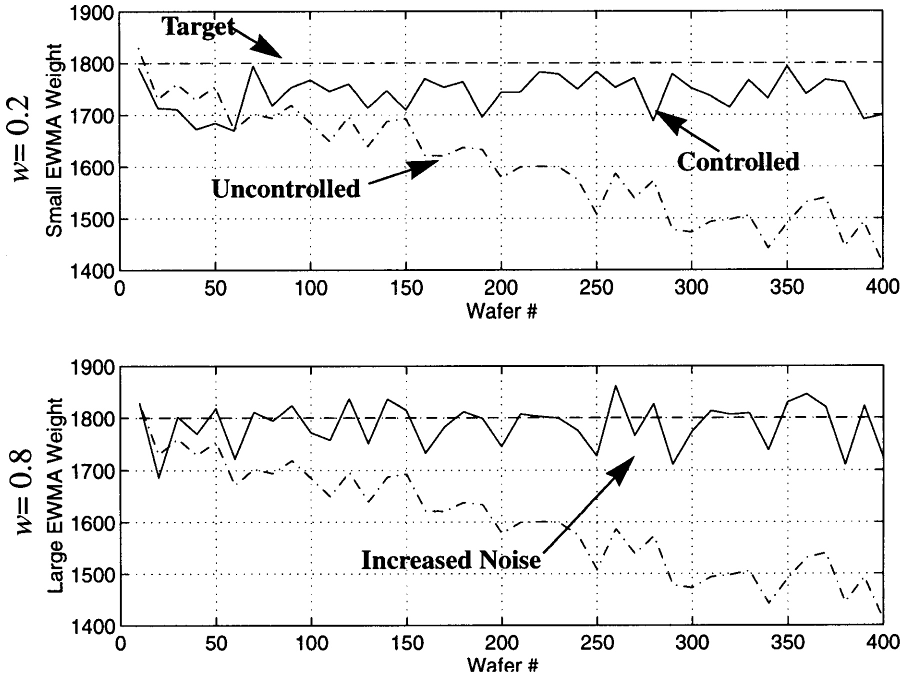
Several works have addressed the issue of detecting shifts with an EWMA [Crowder1, Crowder2, Hembree]. Others have explained the response of the EWMA controller to shifts [Ingolfsson, Sachs]. Here, we provide simulation results to illustrate the effect of different values of $w$. Fig. 4-5 shows the baseline and two controlled outputs, one with a large value of $w$, $w = 0.8$, and the other with a small value, $w = 0.2$. Notice that, after the shift, the $w = 0.8$ response returns to target much faster than the $w = 0.2$ response. On the other hand, higher values of $w$ increase the noise in the controlled output, as can be seen by comparing the two responses. The MSE for the response with $w = 0.2$ is 163 and the MSE for the response with $w = 0.8$ is 113. Further investigation shows that as $w$ continues to increase, performance eventually decreases and the MSE increases to 230 for $w = 1$. This leads us to believe that an optimal value for the EWMA weight exists.

**Figure 4-5. The Effect of Small and Large EWMA Weights on the Control of a Process Shift**



Many processes, including CMP, are known to have residual drifts in the baseline process [Martinez, Hu1, Hu2]. Thus it is important to understand the response of the EWMA controller to a linear drift. From the previous discussion we expect that a process with a large linear drift will be controlled better with a higher EWMA weight. Further, processes with a drift which is small compared to the amount of noise would be controlled better with a medium to small weight, since high weights would only increase the noise. Fig. 4-6 shows the effect of small and large EWMA weights for a process with a large drift and noise having standard deviation equal to 2% of the target. The MSE for the response with $w = 0.2$ is 41 and the MSE for the response with $w = 0.8$, is 17. Again we see that the $w = 0.2$ case has less noise but fails to compensate for the drift as well as the $w = 0.8$ case. We see that with both values of $w$ there is a steady-state constant error (both are on average below the target). This is caused by the fact that each time the EWMA updates the model to compensate for the amount the process has drifted, the process drifts again in the following run. Similar weighting considerations and simulations can be applied to the PCC controller, which can be effective in eliminating this offset term.

**Figure 4-6. The Effect of Small and Large EWMA Weights on the Control of a Process Drift**



These simulations show that for both a process shift and a process drift, there is an optimum value for the EWMA weight parameter, $w$. Theoretical determination of the optimal value of the EWMA parameter, $w$, is demonstrated in Chapter 6 for the SISO EWMA controller with a linear drift buried in white noise. However, such a derivation for the general MIMO controller with linear drifts or shifts has not been realized as of yet. In response to this we will discuss a *practical* two step method for obtaining the optimal weights for these more general systems. First, experimental data is used to characterize a process in terms of the types of disturbances which are expected. This information is then used to simulate the process for many possible combinations of EWMA weights to provide an empirically determined estimate of the optimal EWMA weight matrix. For the CMP process, the process is characterized by an estimate of the expected size and number of shifts in the process, the amount of noise, and the size of the drift. Using this estimate, simulations are performed and the optimum value is extracted. We do this by considering the MSE between the simulated process output and the target. This has been done for the CMP process with an equipment model containing a linear drift extracted from the baseline run. The linear gain coefficients were chosen with percent differences from the controller model based on the amounts the relative inputs contributed to model error. Finally, gaussian noise with standard deviation equivalent to that in the baseline run was added to the process. The plots for MSE in the controlled outputs are shown in Figs. 4-7 and 4-8 as a function of the EWMA weight. As can be seen, the optimal value of $w$ for removal rate lies at approximately 0.4 and the optimal value of $w$ for nonuniformty lies at approximately 0.25. Note that these are not the same as the optimal values for tracking the uncontrolled, or open loop, process. The optimal EWMA weights (in the sense of MSE) may also be increased or decreased to facilitate more aggressive tracking of drifts or reduction of process noise, respectively. In this case the effect of shifts were not taken into account since there are no perceivable shifts in the baseline process.

**Figure 4-7. MSE of the Controlled Removal Rate Versus** *w*

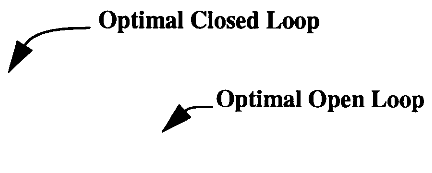
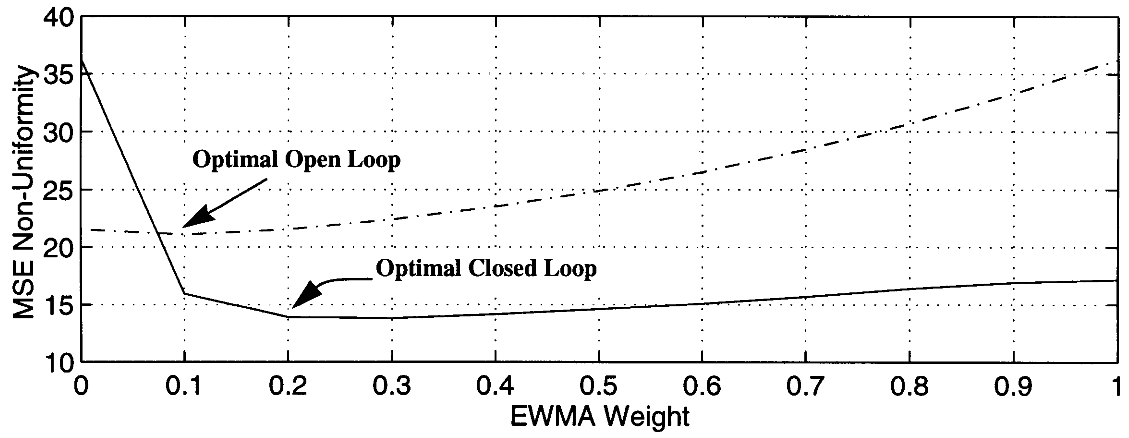


**Figure 4-8. MSE of the Controlled Nonuniformty Versus** *w*



## 4.6 Controller Experiments

Two experiments incorporating the aforementioned methods were performed on two separate CMP tools. For these experiments, the Recursive Bound Pinning method was implemented for determining constrained recipes. The input weighting method was used to compensate for output variance induced by different inputs. The Statistical Discretization method was used in these experiments to generate inputs with a fixed resolution. Due to the given resolution of the machine and the amount of the noise in the process, the difference of the Statistical Discretization and the Delta Discretization method is negligible. Finally, the EWMA weights were chosen as $w(1) = 0.5$ for removal rate and $w(2) = 0.3$ for nonuniformty. These are slightly higher than those predicted in the previous section to facilitate more aggressive control of the process drift.

### 4.6.1 The First CMP Tool

We will compare the performance of the first control run to the baseline process of the first CMP tool shown in Fig. 2-3. The results of the first experiment are shown in Fig. 4-9. The drift in the baseline process (shown in Fig. 2-3) is well compensated for, demonstrating the effectiveness of the algorithm. For comparison, an estimate of the results without control for the same run is shown, based on a constant ***a[0]*** in the model. Examining the input trajectory in Fig. 4-10, we can see several important things. First, the algorithm responds well, successively applying more aggressive control to compensate for accumulating drift. At the same time Speed is increased to maintain removal rate, we also see that Force is decreased and Pressure is increased to improve uniformity. The trade-off between these two goals is also apparent in the resulting control action.

Second, we can see the effectiveness of the normalization to allow variation in the larger inputs. Similarly, we see that Profile, which was found to have only a small correlation to changes in the output, has been effectively stifled by the assignment of a small input weight. Third, we notice from the input trajectory that the Statistical Discretization method has provided appropriate discrete inputs such that the output is maintained near target. Finally, we can compare the baseline and controlled experiments in terms of mean square error (mean deviation from target squared) in the removal rate. We find that the baseline gives an MSE of 3.2x$10^4$ and the controlled run has an MSE of 3.6x$10^3$. As mentioned earlier, in these experiments the controll goal was to maintain both removal rate and nonuniformity to a target, thereby trading off a maximum removal rate or minimal nonuniformity to maintaining an adequate level for each of these throughout the run.

**Figure 4-9. EWMA Controlled Outputs**

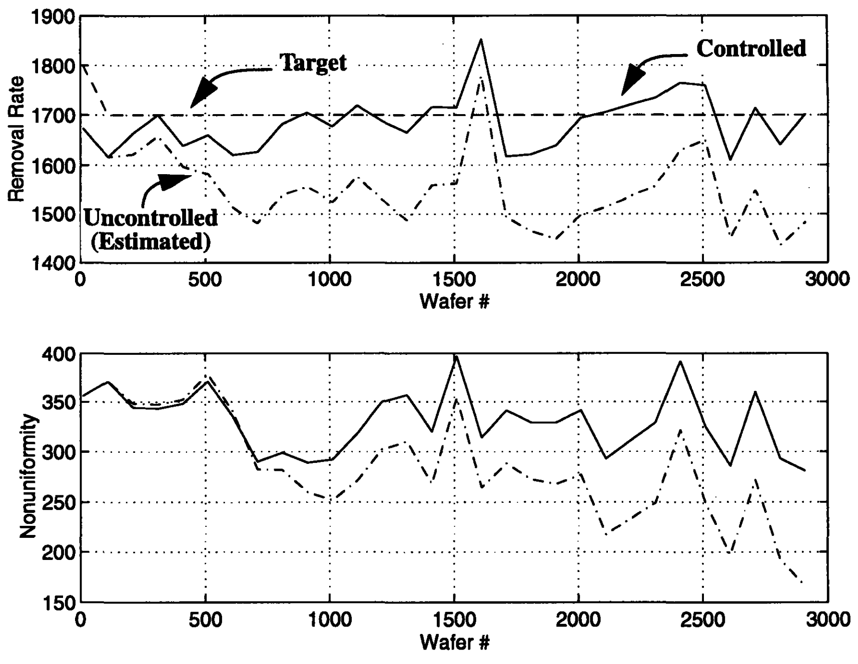
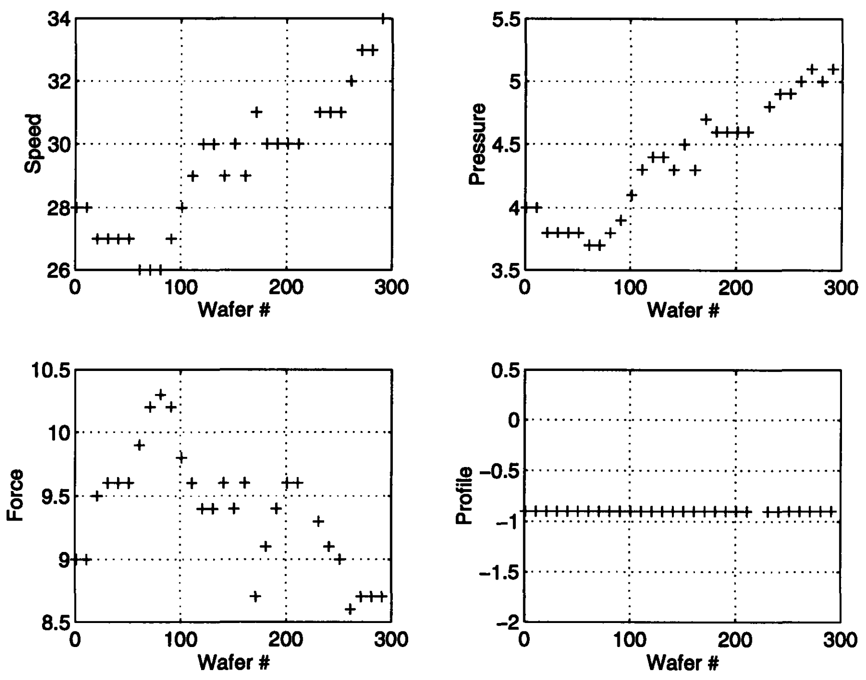


**Figure 4-10. Control Input Trajectory**

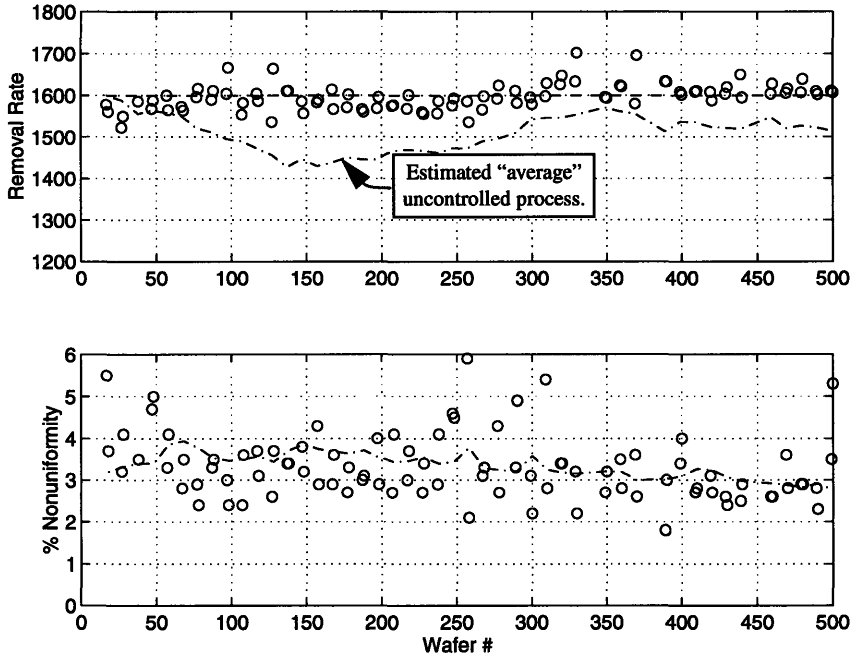


### 4.6.2 The Second CMP Tool (Strasbaugh 6DS-SP)

The second experiment was performed on a Strasbaugh 6DS-SP CMP tool and the control results from this experiment will be compared to industrial specifications described in Section 4.1. These results demonstrate the power of this simple control technique.

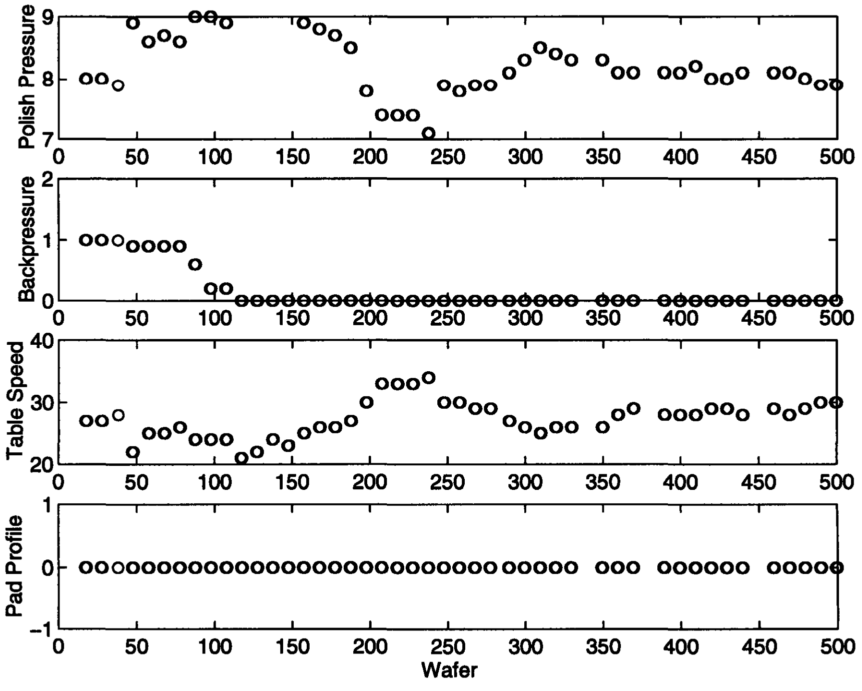
As can be seen in Fig. 4-11, 500 wafers were polished using the EWMA controller with no indication of pad wear or equipment disturbances in the removal rate and no increase in nonuniformity over the entire run. The within-wafer uniformity was 3.3% on average and the removal rate was maintained at 1597 Angstroms/minute with a wafer-to-wafer uniformity of 2.0%. An estimated "average" baseline was determined from the input output data and the model adaptation algorithm given above. As can be seen in Fig. 4-12, the controlled removal rate provides much improvement over the estimated uncontrolled removal rate. Notice also that there is no indication of performance degradation even at the 500 wafer mark. These results demonstrate that the control framework can maintain tight industrial specifications for wafer-to-wafer uniformity while greatly extending the pad life for this process. This suggests the possibility for this methodology to extend CMP pad life to 1000 wafers and beyond, with very little process monitoring, scrap wafers, or machine downtime, while maintaining very tight control of material removal.

Figure 4-11. 500 Wafer Run - Outputs



In addition to successfully controlling this process for more than 500 wafers, several other goals have been met. As can be seen by the input trajectories shown in Fig. 4-12, all the inputs are bounded (they are plotted in their allowable ranges) and discretized (shown by the discrete step changes). The user preference to vary Polish Pressure and Table Speed over Backpressure and Pad Profile is clearly seen in Fig. 4-12.

Figure 4-12. 500 Wafer Run - Inputs



## 4.7 Summary: The EWMA Run by Run Control Framework

We have demonstrated several methods for improving the implementation of run by run control for application in chemical mechanical polishing. These techniques have shown to provide constrained control of the CMP process in the presence of persistent drift. In addition, use of input weighting allows full multivariate control of the process while minimizing unnecessary changes in the input space and decreasing the movement of error-producing inputs. Proper discretization of the inputs was implemented to eliminate errors produced by rounding. Finally, we have demonstrated the successful application of run by run control to compensate for pad wear in chemical-mechanical polishing, extending the pad life beyond 500 wafers with tight control of wafer-to-wafer uniformity.

# Chapter 5

# Stability of the MIMO EWMA and PCC Algorithms

The Exponentially Weighted Moving Average (EWMA) Controller and the Predictor Corrector Controller (PCC) provide stable control for many processes which can be approximated by linear models. Stability criteria have been determined for the Exponentially Weighted Moving Average (EWMA) Controller in the single-input single-output (SISO) and two-input one-output case in [Ingolfsson]. The EWMA and PCC algorithms have been expanded to encompass multiple-input multiple-output (MIMO) systems. This paper explores methods for examining the stability of these MIMO algorithms.

Section 5.1 will begin by briefly discussing the SISO EWMA Controller and an alternate stability analysis to that given in [Ingolfsson]. This analysis will be expanded to the MIMO EWMA Controller in Section 5.2. The stability analysis of the SISO and MIMO PCC algorithms will be derived in Sections 5.3 and 5.4 respectively. Section 5.5 summarizes the chapter.

## 5.1 The SISO EWMA Controller (An alternate approach)

The first step toward a general method for testing the stability of the EWMA algorithm for the MIMO case is to fully understand the SISO case. Although [Ingolfsson] considers the SISO case, we rework the SISO problem here using a state space approach.

Assuming the SISO output of the plant system can be described as

$$y_p[n] = \alpha + \beta u[n], \tag{5-1}$$

the EWMA controller assumes a model system

$$y_m[n] = a[n] + bu[n], \tag{5-2}$$

where a[n] is an EWMA constant term which is updated as

$$a[n] = w(y_p[n] - bu[n]) + (1-w)a[n-1]. \tag{5-3}$$

Finally the equipment setting to the plant is generated as follows:

$$u[n+1] = \frac{1}{b}(T - a[n]). \tag{5-4}$$

In order to examine the stability of such a system, we will rewrite the system of equations (5-1) through (5-4) in state space notation. Specifically, we will let the state vector be as follows:

$$\begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} = \begin{bmatrix} a[n-1] \\ u[n] \end{bmatrix}. \tag{5-5}$$

It should be noted that the system is completely characterized (in the SISO case) by one state variable, $x_1 = a[n-1]$. We choose not to use such as characterization here because it leads to an analysis very similar to that given in [Ingolfsson]. Instead, we will use the state vector given above, as it lends itself to the MIMO case better. Substituting in equations (5-1) through (5-4) into (5-5) we obtain:

$$\begin{bmatrix} x_1[n+1] \\ x_2[n+1] \end{bmatrix} = \begin{bmatrix} w(y_p[n] - bu[n]) + (1-w)a[n-1] \\ \frac{1}{b}(T - a[n]) \end{bmatrix}. \tag{5-6}$$

$$\begin{bmatrix} x_1[n+1] \\ x_2[n+1] \end{bmatrix} = \begin{bmatrix} (w\alpha + w(\beta - b)x_2[n]) + (1-w)x_1[n] \\ \frac{T}{b} - \frac{x_1[n+1]}{b} \end{bmatrix}, \tag{5-7}$$

which becomes:

$$\begin{bmatrix} x_1[n+1] \\ x_2[n+1] \end{bmatrix} = \begin{bmatrix} (1-w) & w(\beta - b) \\ \frac{-1}{b}(1-w) & w - \frac{w\beta}{b} \end{bmatrix} \begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} + \begin{bmatrix} w\alpha & 0 \\ \frac{-w\alpha}{b} & \frac{1}{b} \end{bmatrix} \begin{bmatrix} 1 \\ T \end{bmatrix}. \tag{5-8}$$

Now that we have the state space representation of the system, we can check to see under what conditions the eigenvalues of the system have magnitude less than one. The eigenvalues of the matrix

$$A = \begin{bmatrix} (1-w) & w(\beta - b) \\ \frac{-1}{b}(1-w) & w - \frac{w\beta}{b} \end{bmatrix} \text{, are } \lambda_1 = 0 \text{ and } \lambda_1 = 1 - \frac{w\beta}{b}. \tag{5-9}$$

In order for both eigenvalues to have magnitude less than one,

$$0 < \frac{w\beta}{b} < 2 \ . \tag{5-10}$$

This condition for the stability of the SISO linear system is exactly that provided by [Ingolfsson]. For examples verifying these conditions consult [Ingolfsson].

## 5.2 The MIMO EWMA Controller

The development of the MIMO system is very similar to that above, with a few additions. The system of equations described by equations (5-1) through (5-3) are repeated here in their equivalent matrix forms.

Assuming the MIMO output of the plant system can be described as

$$\underline{y}_p[n] = \begin{bmatrix} \alpha_1 \\ \dots \\ \alpha_m \end{bmatrix} + \begin{bmatrix} \beta_{11} & \dots & \beta_{1q} \\ \dots & \dots & \dots \\ \beta_{r1} & \dots & \beta_{rq} \end{bmatrix} \begin{bmatrix} u_1[n] \\ \dots \\ u_q[n] \end{bmatrix} = \underline{\alpha} + \Im\, \underline{u}[n] \ , \tag{5-11}$$

the EWMA controller assumes a model system

$$\underline{y}_m[n] = \begin{bmatrix} a_1[n] \\ \dots \\ a_m[n] \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \dots & \dots & \dots \\ b_{m1} & \dots & b_{mn} \end{bmatrix} \begin{bmatrix} u_1[n] \\ \dots \\ u_n[n] \end{bmatrix} = \underline{a}[n] + B\underline{u}[n], \tag{5-12}$$

where $\underline{a}$[n] is an EWMA constant term which is updated as

$$\underline{a}[n] = diag(\underline{w})(\underline{y}_p[n] - B\underline{u}[n]) + diag(1 - \underline{w})\underline{a}[n-1]. \tag{5-13}$$

The most significant change to the SISO case is the generation of the equipment setting vector. This is because the matrix $B$ is not necessarily invertible. In almost all cases of MIMO systems $B$

represents an underdetermined set of equations. Therefore, in order to solve for the next equipment setting we use a minimum change solution:

$$\underline{u}[n+1] = M(\underline{T} - \underline{a}[n]) + (I_q - MB)\underline{u}[n]\text{, where } M = B^T(BB^T)^{-1}. \tag{5-14}$$

The determination of stability for this type of system using the method given in [Ingolfsson] appears nearly impossible due to the large number of equations. Although the state space representation is more complicated to develop, as we will see it becomes relatively simple to determine the stability for any given MIMO system and a corresponding EWMA controller.

As before, we let the state space representation be:

$$\begin{bmatrix} \underline{x}_1[n] \\ \underline{x}_2[n] \end{bmatrix} = \begin{bmatrix} \underline{a}[n-1] \\ \underline{u}[n] \end{bmatrix} \tag{5-15}$$

which is now a state vector of length *m*+*n*. Using (5-13) and (5-14) we find that the state vector at, step n+1 is:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \end{bmatrix} = \begin{bmatrix} W(\underline{y}_p[n] - B\underline{u}[n]) + (I_r - W)\underline{a}[n-1] \\ M(\underline{T} - \underline{a}[n]) + (I_q - MB)\underline{u}[n] \end{bmatrix}, \tag{5-16}$$

where $W = diag(\underline{w})$. Now using (5-11) and (5-15) we obtain:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \end{bmatrix} = \begin{bmatrix} W\underline{\alpha} + W(\Im - B)\underline{x}_2[n] + (I_r - W)\underline{x}_1[n] \\ M\underline{T} - M\underline{x}_1[n+1] + (I_q - MB)\underline{x}_2[n] \end{bmatrix} \tag{5-17}$$

and then substituting the first line in for the $\underline{x}_1[n+1]$ in the second line, we get:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \end{bmatrix} = \begin{bmatrix} W\underline{\alpha} + W(\Im - B)\underline{x}_2[n] + (I_r - W)\underline{x}_1[n] \\ M\underline{T} - M(W\underline{\alpha} + W(\Im - B)\underline{x}_2[n] + (I_r - W)\underline{x}_1[n]) + (I_q - MB)\underline{x}_2[n] \end{bmatrix}. \tag{5-18}$$

This can be simplified to:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \end{bmatrix} = \begin{bmatrix} (I_r - W) & W(\Im - B) \\ -M(I_r - W) & (I_q - MB) - MW(\Im - B) \end{bmatrix} \begin{bmatrix} \underline{x}_1[n] \\ \underline{x}_2[n] \end{bmatrix} + \begin{bmatrix} W\underline{\alpha} & 0 \\ -MW\underline{\alpha} & M \end{bmatrix} \begin{bmatrix} I \\ \underline{T} \end{bmatrix}. \tag{5-19}$$

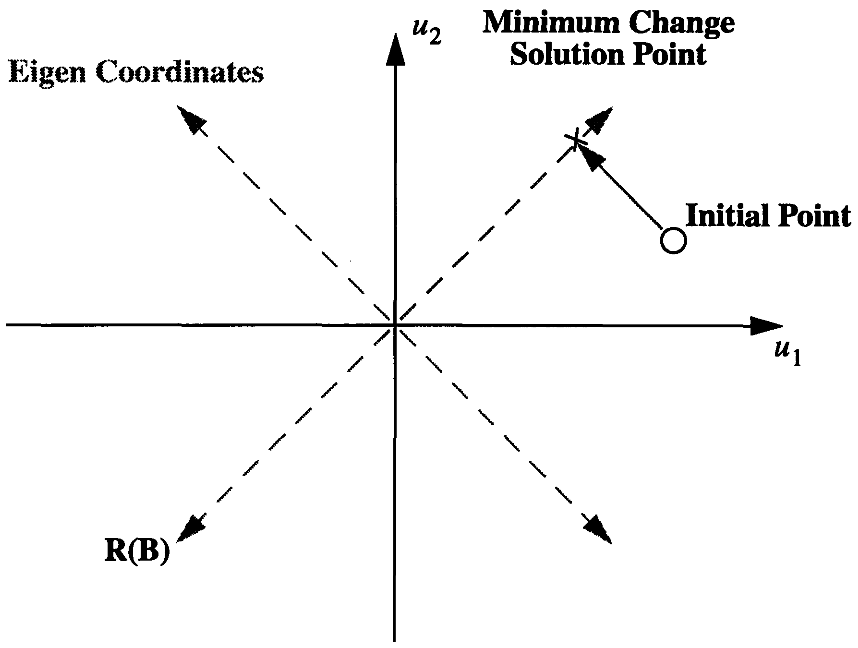
So finally we have

$$A = \begin{bmatrix} diag(1-\underline{w}) & diag(\underline{w})(\mathfrak{I}-B) \\ B^T(BB^T)^{-1}diag(\underline{w}-1) & B^T(BB^T)^{-1}diag(\underline{w})(B-\mathfrak{I})+(I_q-B^T(BB^T)^{-1}B) \end{bmatrix}, \qquad (5\text{-}20)$$

where $A$ is the one-step transition matrix for the state space representation of the EWMA controller. Using this matrix we can test the stability of the EWMA controller by determining the size of the eigenvalues.

At this point it is important to digress and discuss some issues regarding the stability of this system. In most state space representations, stability implies asymptotic stability, as one would normally expect the system to converge to zero given no input. The state space representation of the EWMA controller is slightly different. We have included the equipment settings (machine inputs) as states and we have set the system up to determine the minimum change solution, which will not always return the equipment settings to zero given a zero set point (target). This is because the equipment settings will move perpendicular to the r-dimensional range space of the gain matrix $B$ (inorder to change these inputs as "little" as possible) and stop as soon as the outputs equal the target set point. Since the movement of the equipment settings will always be perpendicular to the $r$-dimensional range space which (by our problem definition) is of dimension less than the $q$-dimensional space of the equipment settings, we note that the equipment settings will also move only in an $r$-dimensional space. Thus, there is no way that all the inputs of the $q$-dimensional space will converge to zero unless the starting point of the system lies on the perpendicular to the range space. This is illustrated in Fig. 5-1 with $r$=1 and $q$=2. We see that the minimum change solution causes only one state (equipment setting) in its eigenspace to return to zero. This has been verified through simulation by determining (for several initial conditions) that the change in inputs span only an $r$-dimensional space.

**Figure 5-1. Movement of the EWMA Controller Inputs in the Underdetermined Case**



This digression is important because it shows that this state space representation for the EWMA controller will not be asymptotically stable. It will however be stable in the sense of Lyapunov. Therefore, we should expect that the eigenvalues of the transition matrix have magnitude less than or equal to one. In fact, we expect from the analysis above that there will be *q-r* states that do not return to zero nor grow and thus we expect to have *q-r* eigenvalues with magnitude one. This also has been verified through simulation. In addition, the movement of the equipment settings in the *r*-dimensional subspace orthogonal to the range space will be completely determined by the matrix *B* (which specifies the r directions that the equipment settings move) and by the state vector $x_1[n]$ consisting of the EWMA offset terms for each output (which specify how much to move in that direction). This means that *r* of the states in $x_2[n]$ are linearly dependent on the states in $x_1[n]$. Therefore, we will also have *r* eigenvalues of the transition matrix with zero magnitude.

In summary, we know that *r* of the eigenvalues must be zero and *q-r* of the eigenvalues must be one. If all the remaining eigenvalues have magnitude less than one, the entire system will be stable in the sense of Lyapunov and the outputs will be asymptotically stable. If this is the case, then the output will be guaranteed to converge to the set point. We will demonstrate this technique for two cases.

### 5.2.1 MIMO EWMA Example 1: Stability vs. Varying Model Error

Under normal operating conditions the EWMA controller provides a stable response over a large range of weights. This is demonstrated in the following example where we consider a 4 input 2 output system. The weight vector, *w*, for this entire example will be $\{0.5, 0.3\}$ and the target vector is $T = \begin{bmatrix} 2000 \\ 350 \end{bmatrix}$.

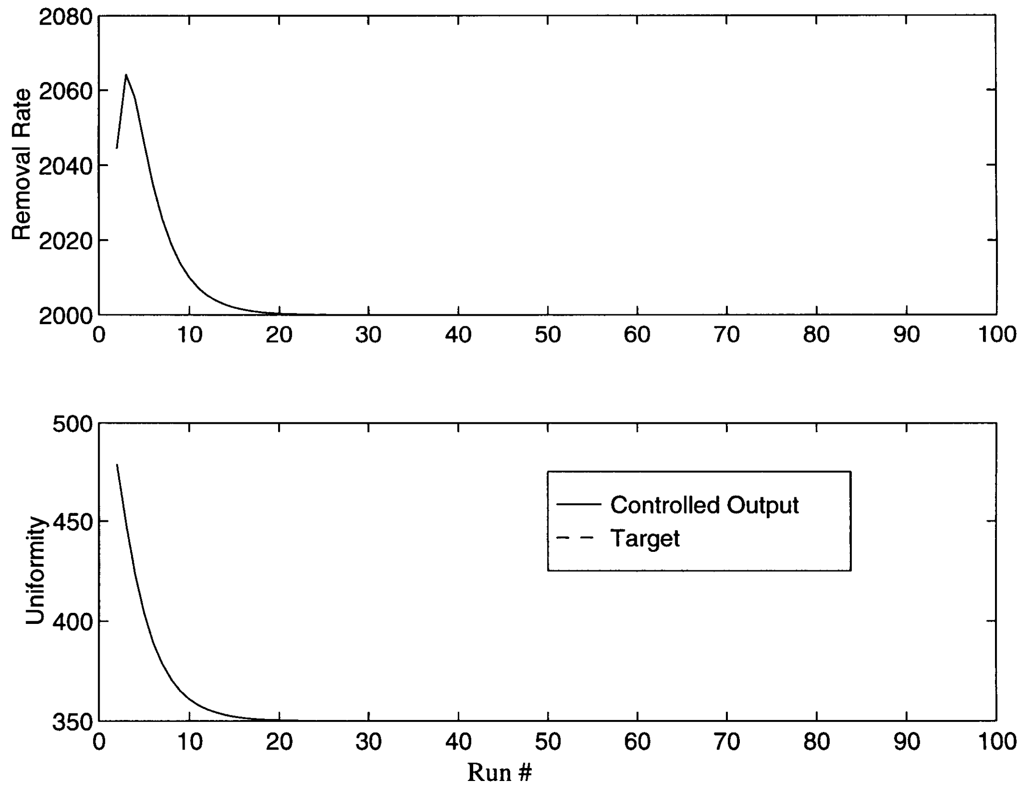
*5.2.1.1 A Stable System*

Consider the arbitrary example with the following coefficient matrices for the model and plant systems:

$$B = \begin{bmatrix} 50.35 & 120 & 163.4 & 8 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} \quad \text{and} \quad \Im = \begin{bmatrix} 52.09 & 183.56 & 164.66 & 12.76 \\ 14.41 & 33.34 & 38.11 & 9.61 \end{bmatrix}.$$

The corresponding eigenvalues of $A$ have magnitudes of $\{0.3522, 0.7266, 0, 0, 1, 1\}$. Since the dimension of the output space is $r$=2 and the dimension of the input space is $q$=4, we should have $q$-$r$=2 eigenvalues with magnitude one and $r$=2 eigenvalues with magnitude zero; this is exactly what we see. The remaining states have magnitude less than one and thus we expect to have a stable system, as shown in Fig. 5-2.

**Figure 5-2. A Normal EWMA Response**



*5.2.1.2 A Stable System Approaching Instability*

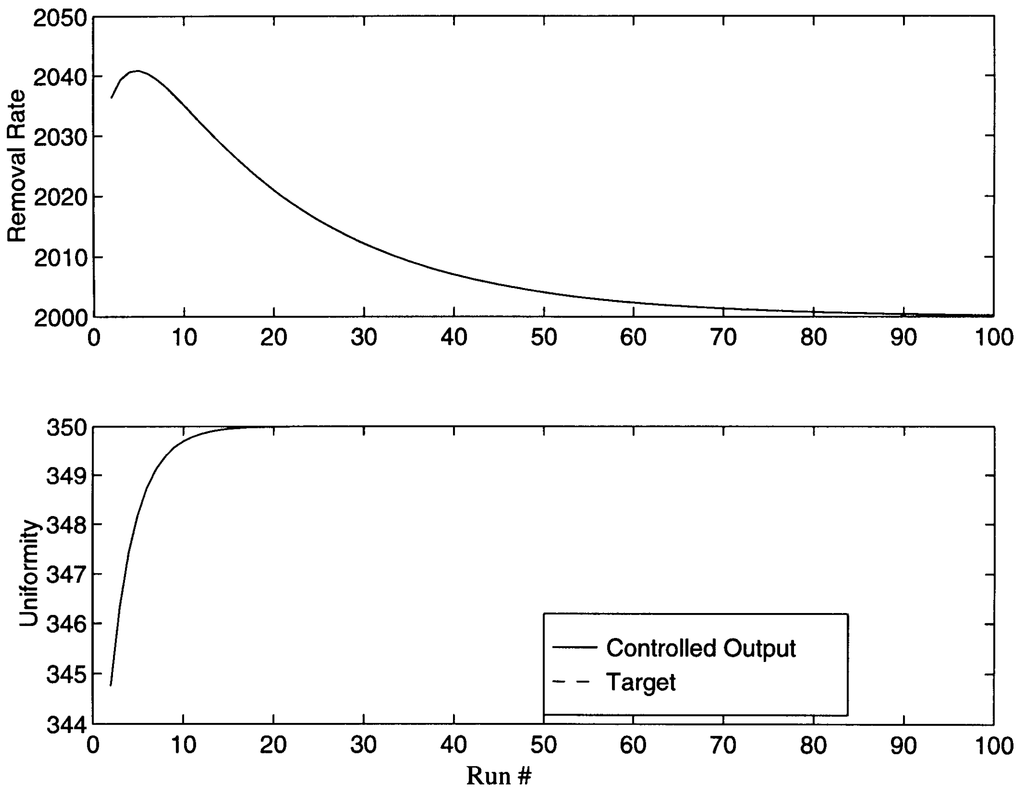
The previous case can be made to approach an unstable boundary by decreasing the (1,2) coefficient of the plant matrix from that of the controller. The remaining coefficients are kept relatively equal to those of the controller model. The associated input matrices for the model and plant linear systems are given below:

$$B = \begin{bmatrix} 50.35 & 120 & 163.4 & 8 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} \quad \text{and} \quad \mathfrak{I} = \begin{bmatrix} 50.35 & -6.65 & 163.4 & 8.4 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix}.$$

These give the eigenvalues of $A$ to have magnitudes of $\{0.9467, 0, 0, 1, 1, 0.7\}$. Again, we should have $q$-$r$=2 eigenvalues with magnitude one and $r$=2 eigenvalues with magnitude zero. The remaining state variables have magnitude less than one and thus we expect to have a stable system, as shown in Fig. 5-2. However, the response is beginning to be slow.

Recall that in the SISO case if the controller coefficient had a different sign than the plant, the system would be unstable. As we can see by comparing the two matrices above, a sign change in one coefficient does not necessarily imply unstable control for the MIMO EWMA Controller. This is due to the fact that other settings can be adjusted to compensate for the output error. As can be seen in Fig. 5-3, the first output with the poorly matched coefficients approaches the target very slowly.

**Figure 5-3. The Response of a System Approaching Instability**



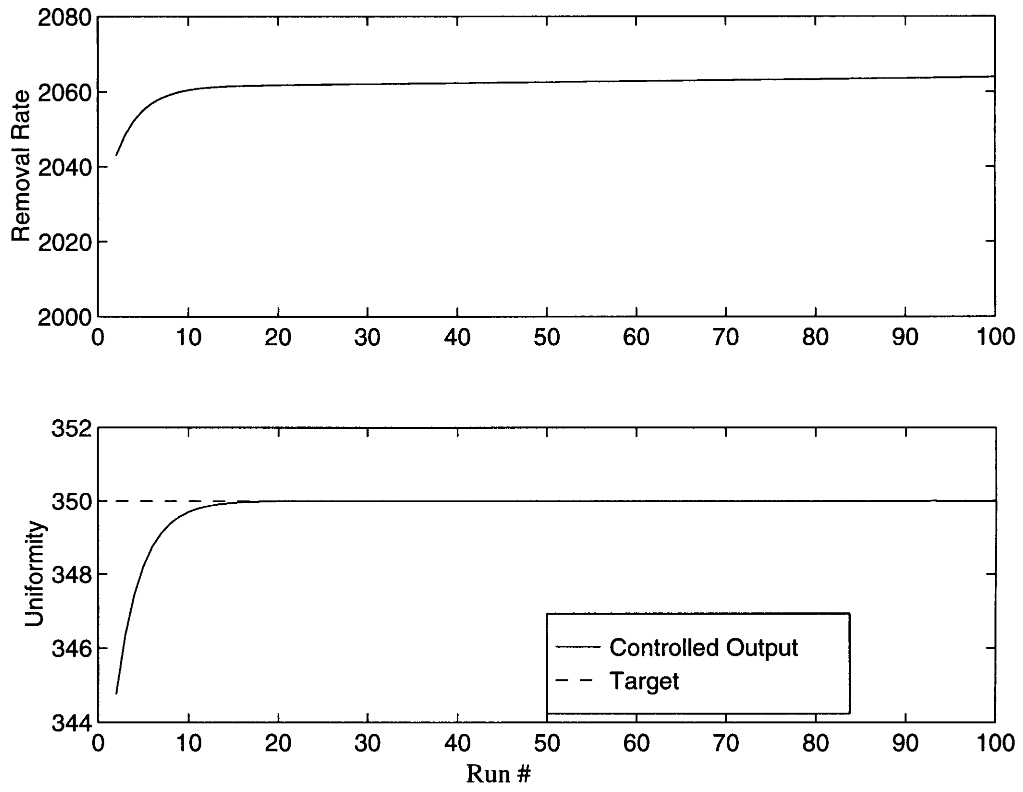
*5.2.1.3 An Unstable System*

The previous case can be made unstable by further decreasing the (1,2) coefficient of the plant. As the coefficient is decreased further, the system response slows until it eventually moves in the opposite direction and goes unstable. The corresponding matrices are given below.

$$B = \begin{bmatrix} 50.35 & 120 & 163.4 & 8 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} \quad \text{and} \quad \mathfrak{I} = \begin{bmatrix} 50.35 & -22 & 163.4 & 8.4 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix}.$$

These give the eigenvalues of $A$ to have magnitudes of $\{1.0004, 0, 0, 1, 1, 0.7\}$. Again, we have $q$-$r$=2 eigenvalues with magnitude one and $r$=2 eigenvalues with magnitude zero. However, one of the remaining states has magnitude greater than one and hence the system is just beginning to diverge away from the target in an unstable manner, as shown in Fig. 5-4.

**Figure 5-4. The Unstable System**



*5.2.1.4 A Stable Response Approaching Instability (Oscillating)*

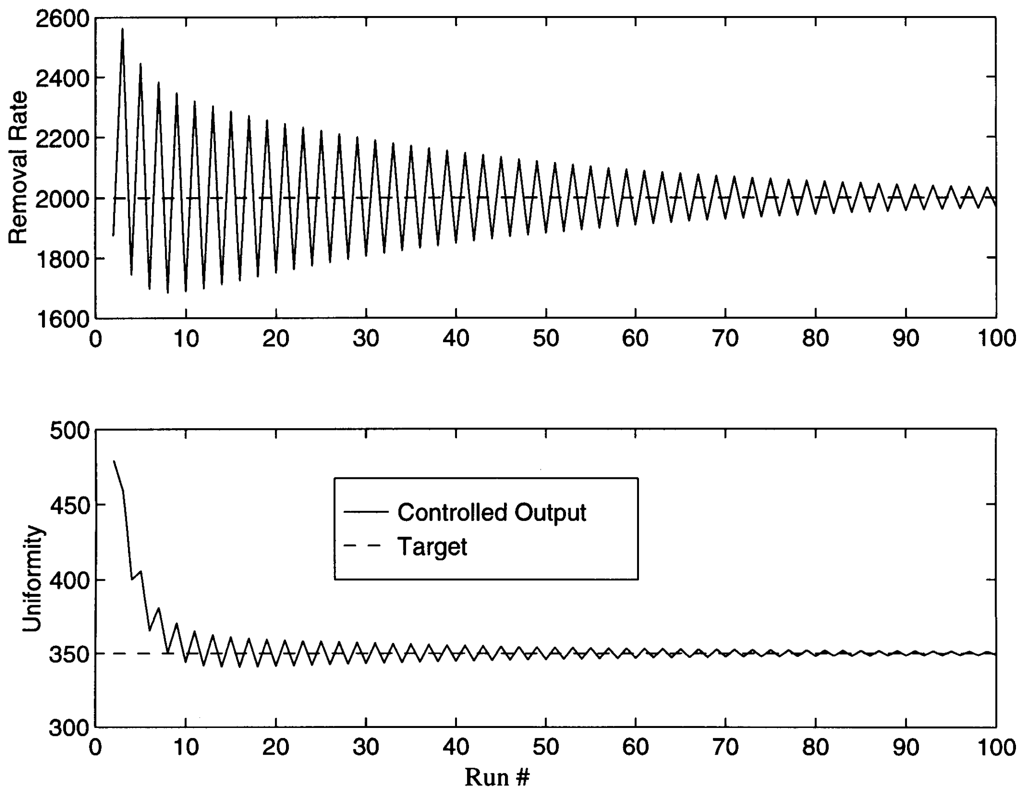
The two previous examples considered the effect when a coefficient of the plant is decreased relative to its corresponding model coefficient. In this example we consider the effect of increasing the plant coefficient relative to that of the controller model. We increase the (1,2) coefficient such that the system is near instability. The corresponding matrices are:

$$B = \begin{bmatrix} 50.35 & 120 & 163.4 & 8 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} \qquad \text{and} \qquad \Im = \begin{bmatrix} 50.35 & -22 & 163.4 & 8.4 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix}.$$

These give the eigenvalues of $A$ to have magnitudes of $\{0.9751, 0.6659, 0, 0, 1, 1\}$. We see that the eigenvalues remaining after ignoring the expected ones and zeros are all less than 1. The first remaining eigenvalue is very near one and hence we expect the system response to be poor even though it is stable. This is in fact the case and can be seen in Fig. 5-5. Similar to the case when the plant coefficient was severely decreased, we see that a single coefficient ratio times the weight can be greater than 2, unlike in the SISO case, and still allow for stable control:

$$w\frac{\Im(1,2)}{B(1,2)} = (0.5)\frac{580}{120} = 2.42. \qquad (5\text{-}21)$$

**Figure 5-5. A System Approaching Instability (Oscillating)**



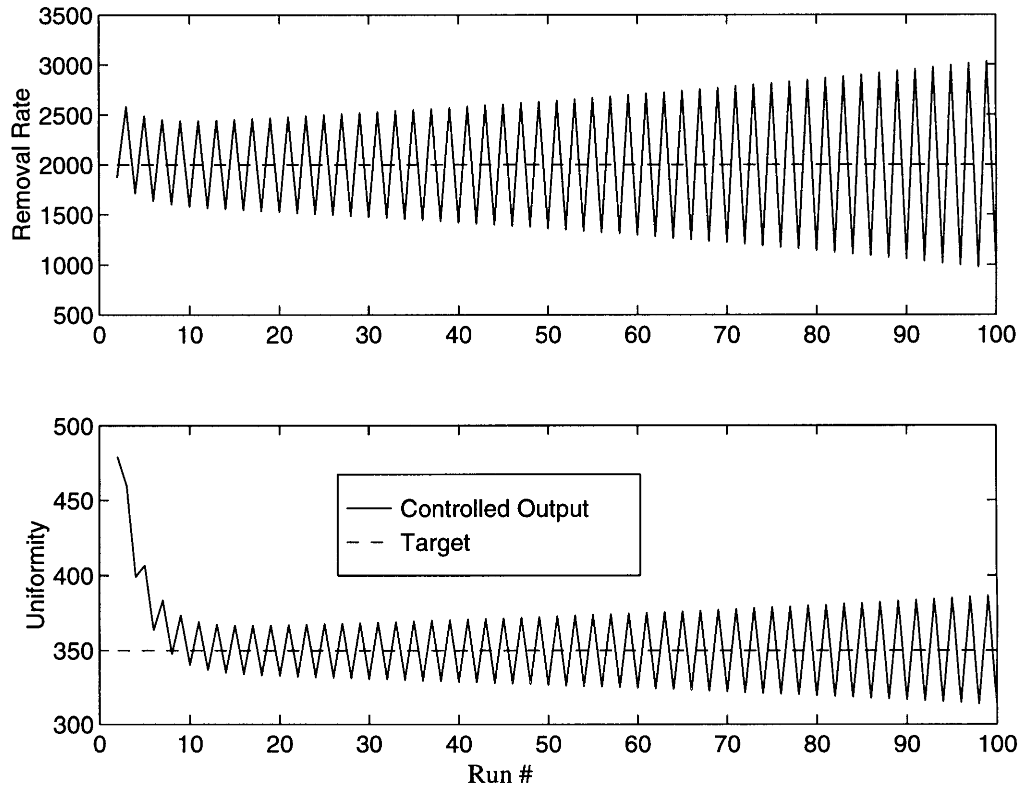
*5.2.1.5 An Oscillating Unstable System*

The previous case can be made unstable by further increasing the (1,2) coefficient of the plant. As the coefficient is increased further, the oscillations become worse until they begin to grow. The corresponding matrices are given below:

$$B = \begin{bmatrix} 50.35 & 120 & 163.4 & 8 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} \quad \text{and} \quad \Im = \begin{bmatrix} 50.35 & 590 & 163.4 & 8.4 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} .$$

These give the eigenvalues of $A$ to have magnitudes of $\{1.0099, 0.6659, 0, 0, 1, 1\}$. Again, we have $q$-$r$=2 eigenvalues with magnitude one and $r$=2 eigenvalues with magnitude zero. However, one of the remaining states has magnitude greater than one and hence we obtain the unstable system shown in Fig. 5-6.

**Figure 5-6. An Oscillating Unstable System**



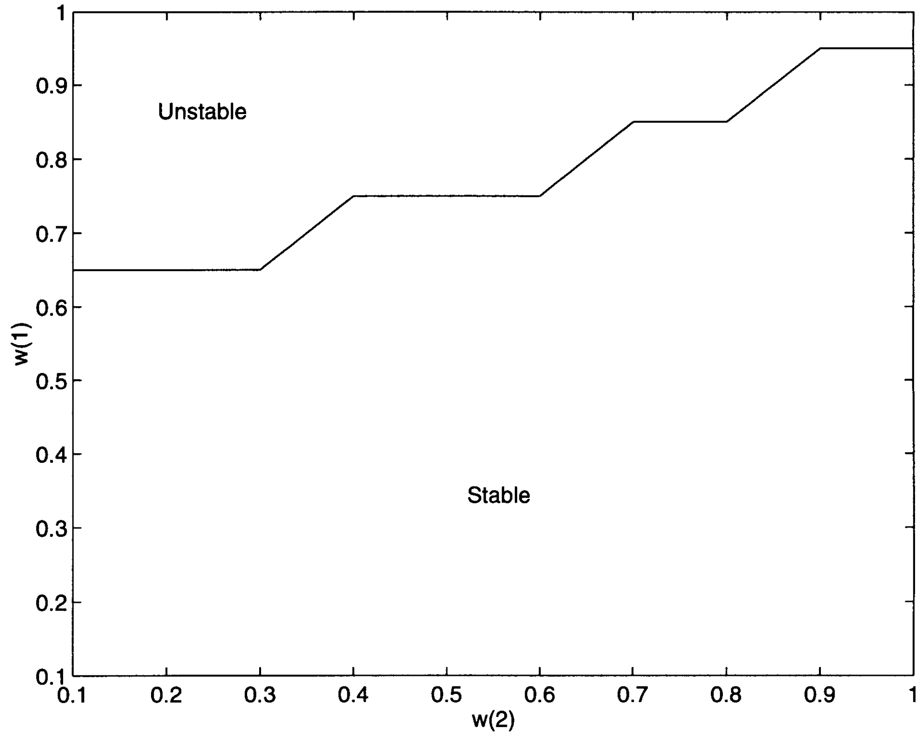
### 5.2.2 MIMO EWMA Example 2: Region of Stability Based vs. EWMA Weights

In addition to the example above where the stable regions were determined for different coefficients, we can also determine the region of stability for a given plant and controller model as a function of the EWMA weight. For the case we consider, the i,j$^{th}$ coefficient has normally distributed, $\aleph(0, \Im(i, j))$, noise. It also contains two extremely bad coefficients $\{(1,2), (2,3)\}$. The corresponding matrices are:

$$B = \begin{bmatrix} 50.35 & 120 & 163.4 & 8 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix} \quad \text{and} \quad \Im = \begin{bmatrix} 52.09 & 550 & 164.66 & 12.76 \\ 14.41 & 33.34 & 60 & 9.61 \end{bmatrix} .$$

Using these while varying the EWMA weights for each output over their possible ranges (0,1], we can efficiently compute which weights will provide stable control by simply doing finding the eigenvalues of a matrix, rather than doing a large number of computationally expensive simulations. The resulting regions for stable weights are shown in Fig. 5-7. As can be seen from the large difference in the plant and model matrices above, a surprisingly large model error and large weights are required for the system to be unstable.

**Figure 5-7. Stable and Unstable Regions for a Given Plant and Controller Model**



## 5.3 The SISO PCC Algorithm

Analogous the EWMA algorithm, the first step toward a general method for testing the stability of the PCC algorithm [Butler] for the MIMO case is to fully understand the SISO case.

Assume the "true" SISO model of the plant system can be described as:

$$y_p[n] = \alpha + \beta u[n], \tag{5-22}$$

where $\alpha$ is a constant and $u[n]$ is the equipment setting for the plant. The controller assumes a model system

$$y_m[n] = a[n] + bu[n], \tag{5-23}$$

where $a[n]$ is the model's prediction of the true offset term, $\alpha$. The dynamic prediction of the offset term is updated by adding an EWMA of the offset term and an EWMA of the slope, or change, of the offset term. This can be expressed as:

$$a[n] = v[n] + s[n]\text{, where} \tag{5-24}$$

$$v[n] = w_1(y_p[n] - bu[n]) + (1 - w_1)v[n-1]\text{, and} \tag{5-25}$$

$$s[n] = w_2c[n] + (1 - w_2)s[n-1]\text{, with} \tag{5-26}$$

$$c[n] = (y_p[n] - bu[n]) - v[n]\text{.} \tag{5-27}$$

The term *c[n]* is just the present value of the change in the offset term. The equipment setting at step *n* is generated by setting $y_m[n] = T$ and solving for *u[n]*:

$$u[n] = \frac{1}{b}(T - a[n-1]) = \frac{T}{b} - \frac{v[n-1]}{b} - \frac{s[n-1]}{b}\text{ .} \tag{5-28}$$

The stability of the PCC algorithm is best determined by recognizing that the equipment settings to the system at any step are characterized by the values of *v[n]* and *s[n]*. Using these as the states of the process, we will attempt to write the PCC in a state space representation. In order to do this, we first simplify the equations for *v[n]* and *s[n]* and eliminate their dependence on other variables. We begin the stability analysis by substituting (5-22) and (5-28) into (5-25) to obtain:

$$v[n] = w_1\left(\alpha + \left(\frac{\beta}{b} - 1\right)(T - v[n-1] - s[n-1])\right) + (1 - w_1)v[n-1]\text{,} \tag{5-29}$$

which is then simplified to

$$v[n] = w_1\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right) + \left(1 - \frac{w_1\beta}{b}\right)v[n-1] + w_1\left(1 - \frac{\beta}{b}\right)s[n-1]\text{.} \tag{5-30}$$

This is our simplified expression for *v[n]*. Now we seek a simplified expression for *s[n]*. In order to do this we first simplify the expression for *c[n]*. Substituting (5-22) and (5-30) into (5-27) we obtain:

$$c[n] = \alpha + (\beta - b)u[n] - \left\{w_1\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right) + \left(1 - \frac{w_1\beta}{b}\right)v[n-1] + w_1\left(1 - \frac{\beta}{b}\right)s[n-1]\right\}\text{,} \tag{5-31}$$

which we also simplify to,

$$c[n] = u[n](\beta - b) + \left(\frac{\beta w_1}{b} - 1\right)v[n-1] + w_1\left(\frac{\beta}{b} - 1\right)s[n-1] + \alpha - w_1\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right)\text{.} \tag{5-32}$$

Substituting (5-28) into (5-32) we obtain:

$$c[n] = \frac{\beta}{b}(w_1 - 1)v[n-1] + (w_1 - 1)\left(\frac{\beta}{b} - 1\right)s[n-1] + (1 - w_1)\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right), \tag{5-33}$$

which when substituted into (5-26) gives the desired simplified expression for *s[n]*:

$$s[n] = w_2(w_1 - 1)\frac{\beta}{b}v[n-1] + \left((1 - w_2) + w_2(w_1 - 1)\left(\frac{\beta}{b} - 1\right)\right)s[n-1] + w_2(1 - w_1)\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right). \tag{5-34}$$

In order to examine the stability of such a system, we will rewrite the system of equations (5-22) through (5-28) in state space notation. Specifically, we will let the state vector be as follows:

$$\begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} = \begin{bmatrix} v[n-1] \\ s[n-1] \end{bmatrix}, \tag{5-35}$$

and thus

$$\begin{bmatrix} x_1[n+1] \\ x_2[n+1] \end{bmatrix} = \begin{bmatrix} w_1\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right) + \left(1 - \frac{w_1\beta}{b}\right)v[n-1] + w_1\left(1 - \frac{\beta}{b}\right)s[n-1] \\ w_2(w_1 - 1)\frac{\beta}{b}v[n-1] + \left((1 - w_2) + w_2(w_1 - 1)\left(\frac{\beta}{b} - 1\right)\right)s[n-1] + w_2(1 - w_1)\left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right) \end{bmatrix}. \tag{5-36}$$

This can then be written as:

$$\begin{bmatrix} x_1[n+1] \\ x_2[n+1] \end{bmatrix} = \begin{bmatrix} \left(1 - \frac{w_1\beta}{b}\right) & w_1\left(1 - \frac{\beta}{b}\right) \\ w_2(w_1 - 1)\frac{\beta}{b} & \left((1 - w_2) + w_2(w_1 - 1)\left(\frac{\beta}{b} - 1\right)\right) \end{bmatrix} \begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2(1 - w_1) \end{bmatrix} \left(\alpha + \left(\frac{\beta}{b} - 1\right)T\right). \tag{5-37}$$

Note that this is now in the standard state space form:

$$\boldsymbol{x}[n+1] = A\boldsymbol{x}[n] + \tilde{B}\boldsymbol{u}[n], \tag{5-38}$$

where $A$ is the one-step transition matrix and $\tilde{B}$ is the state space input matrix (notice that the input is a constant). This linear discrete-time state space system is asymptotically stable if and only if all the eigenvalues of the transition matrix have magnitude less than one. Now that we have the state space representation of the system, we can readily check this condition. The eigenvalues of the transition matrix

$$\mathrm{A} = \begin{bmatrix} \left(1 - \frac{w_1\beta}{b}\right) & w_1\left(1 - \frac{\beta}{b}\right) \\ w_2(w_1 - 1)\frac{\beta}{b} & \left((1 - w_2) + w_2(w_1 - 1)\left(\frac{\beta}{b} - 1\right)\right) \end{bmatrix}, \tag{5-39}$$

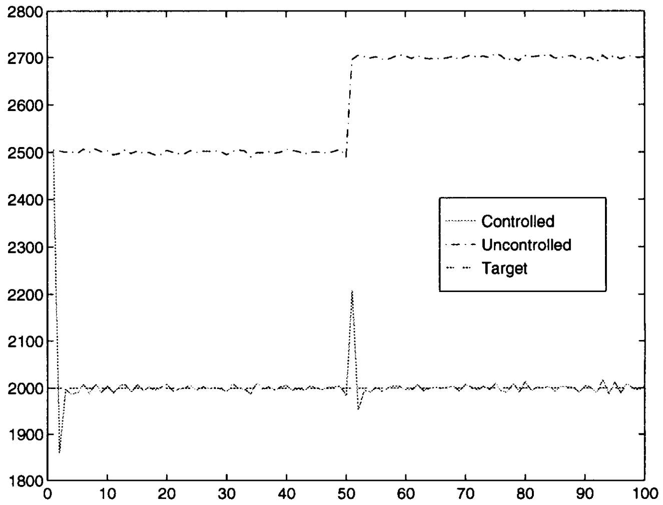
can be computed given any set of PCC weights, {w1,w2}, and the ratio of the plant and model coefficient, $\frac{\beta}{b}$.

### 5.3.1 SISO PCC Example 1

The following example demonstrates the technique outlined in the previous section. In this example, a PCC controller of the form given by equations (5-22) through (5-28) is used with PCC weights $w_1$ =0.5 and $w_2$=0.7. The model coefficient is fixed at $b$=200, as is the constant term in the plant, $\alpha$=1000. The initial values of *v[n]* and *s[n]* are set at zero.

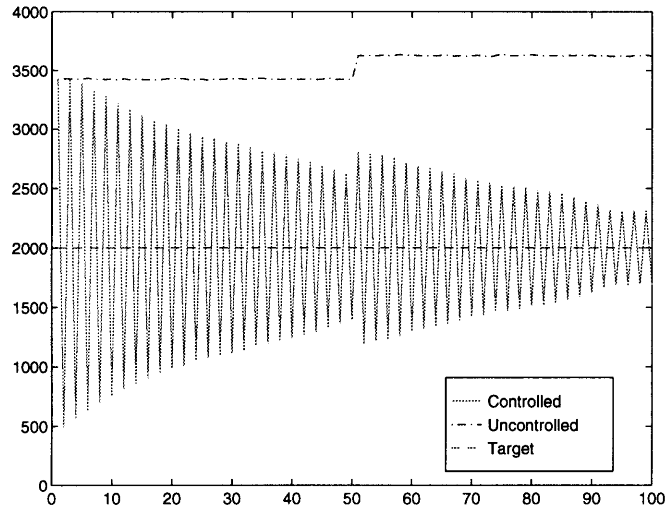
We begin with a value of $\beta$ = 300. By computing the eigenvalues of the transition matrix, we see that they have magnitudes of $|\lambda_1| = 0.5551$ and $|\lambda_2| = 0.1801$, which are both less than one and therefore we expect stable performance from the system. As seen from the simulation plot shown in Fig. 5-8, this is exactly what is observed.

**Figure 5-8. Stable SISO PCC**



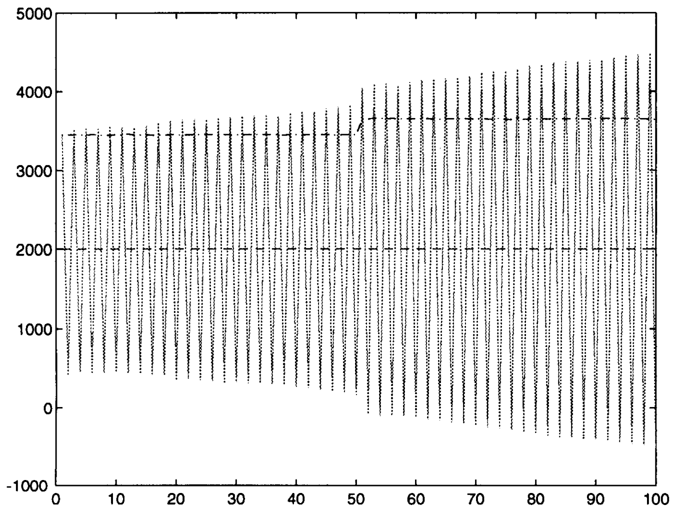
We can see by increasing $\beta$ to $\beta$=485, that we approach the limits of stability. This is demonstrated by the closeness of the eigenvalue magnitudes, $|\lambda_1| = 0.9833$ and $|\lambda_2| = 0.5721$, to one. The response of the system is shown in Fig. 5-9.

**Figure 5-9. Nearly Unstable SISO PCC**



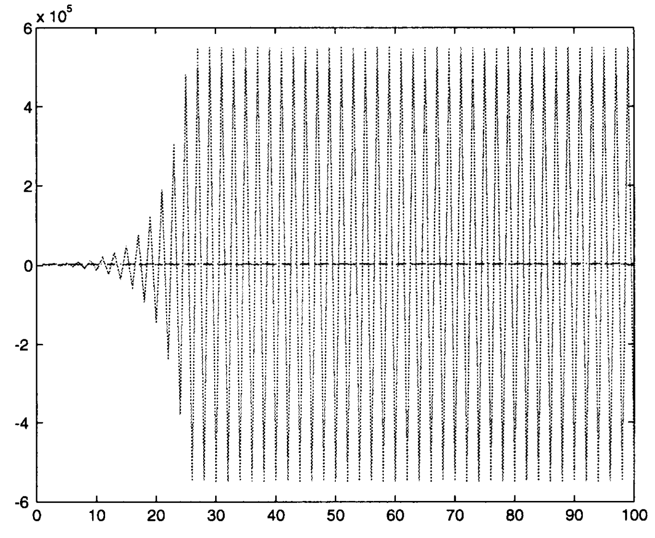
A slightly larger increase in $\beta$ to $\beta = 490$, results in eigenvalues of magnitude $|\lambda_1| = 1.0048$ and $|\lambda_2| = 0.5723$, which now exceed 1 and therefore, as seen in Fig. 5-10, the system is unstable.

**Figure 5-10. Unstable SISO PCC**



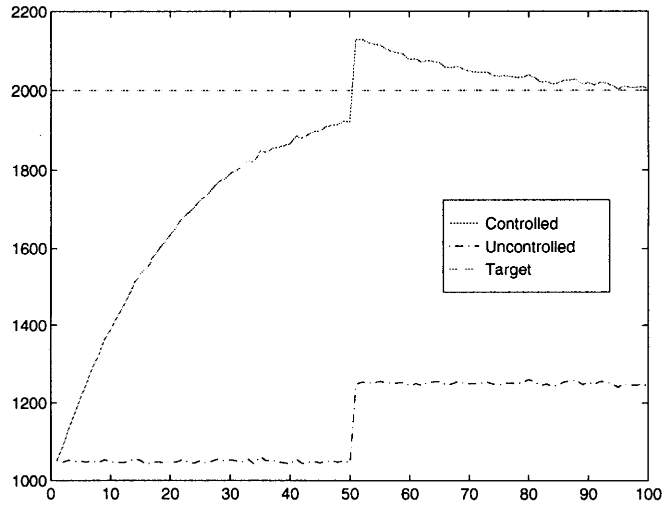
Finally, we see that further increases in the plant coefficient, $\beta$, to $\beta = 550$ rapidly increases the system's unstable behavior. This is seen in Fig. 5-11, where $|\lambda_1| = 1.2620$ and $|\lambda_2| = 0.5745$.

**Figure 5-11. Severely Unstable SISO PCC**



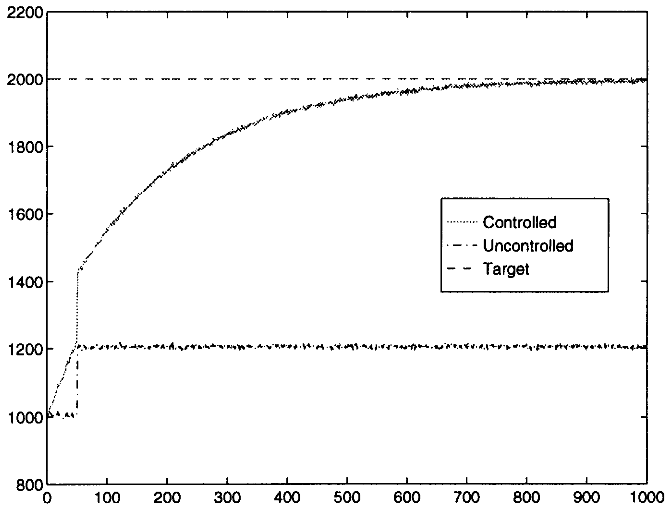
Next we will consider decreases in the plant coefficient, relative to that of the model. Specifically, we see that for $\beta = 10$, the system is still stable, although it has a slow response. Again, this can be seen by computing the magnitude of the eigenvalues, $|\lambda_1| = 0.9487$ and $|\lambda_2| = 0.6588$, which are both less than one. The response of this system in shown in Fig. 5-12.

**Figure 5-12. Small Plant to Model Coefficient Ratio Stable SISO PCC**



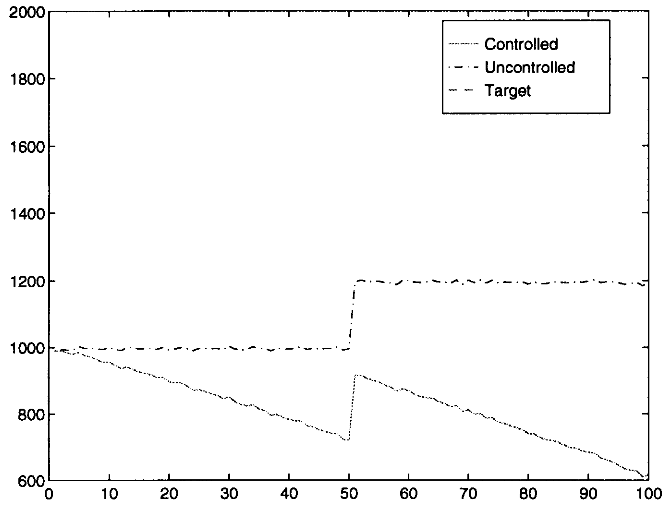
By further decreasing $\beta$ to $\beta = 1$, we see that the response time is extremely poor, but as pointed out by the magnitude of the eigenvalues, $|\lambda_1| = 0.9950$ and $|\lambda_2| = 0.6508$, the system is still stable. Notice that the time axis in Fig. 5-13 is much longer, which highlights the extremely poor response time due to the large model error.

Figure 5-13. Very Small Plant to Model Coefficient Ratio Stable SISO PCC



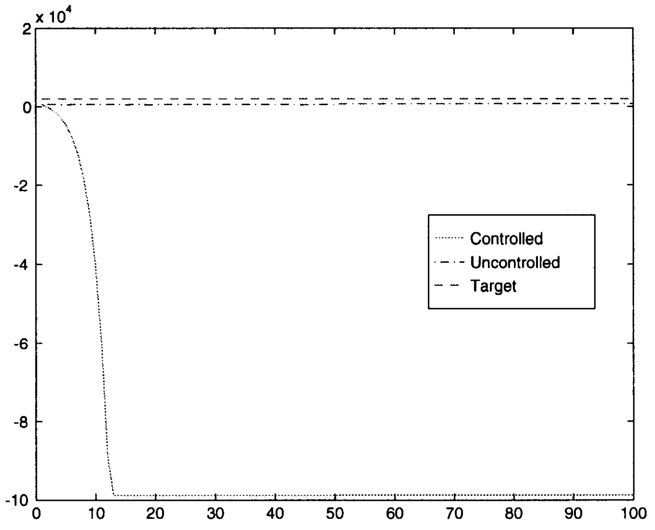
As with the stability criteria for the EWMA Controller, we see that once the plant coefficient changes sign from that of the model coefficient, the system becomes unstable. The value $\beta = -1$, corresponds to a transition matrix with eigenvalues which have magnitudes $|\lambda_1| = 1.005$ and $|\lambda_2| = 0.6493$. These indicate that we should expect the unstable behavior shown in Fig. 5-14. Here we see that the controlled response is actually moving opposite the direction needed to return the system to target.

Figure 5-14. Negative Plant to Model Coefficient Ratio (Unstable SISO PCC)



As a final example, we see that if $\beta$ has a large negative value, and the coefficient ratio becomes large, then the eigenvalues will also be much larger than one, $|\lambda_1| = 1.4575$ and $|\lambda_2| = 0.6175$. This indicates the severely unstable case shown in Fig. 5-15.

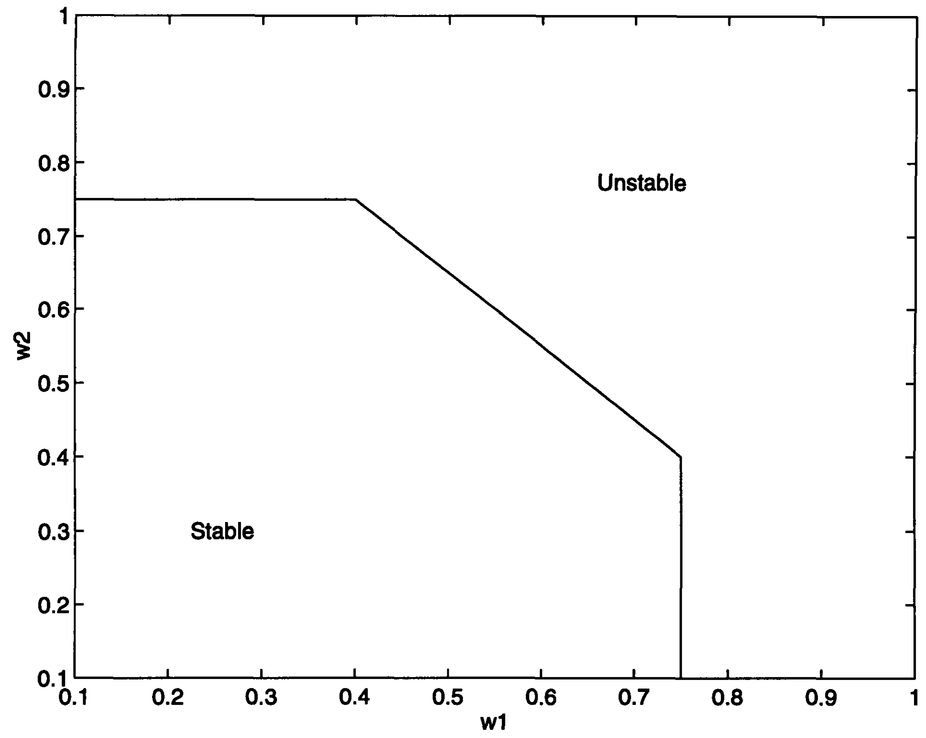
**Figure 5-15. Large Negative Coefficient Ratio (Severely Unstable SISO PCC)**



## 5.3.2 SISO PCC Example 2

Similar to the EWMA MIMO controller with two outputs (Section 3.2) we can generate the regions of stable operation given a fixed plant and controller model. For example, with $\beta$ =500, $b$ =200, and $\alpha$ =1000, the stable region was efficiently computed using the eigenvalues of the transition matrix. The resulting regions are shown in Fig. 5-16. As before, in order to generate a case such as this, the coefficient error had to be significant. In this case the plant coefficient was more than twice that of the controller model. This demonstrates the wide range over which stable control is maintained using this controller. As will be demonstrated in the next section, the MIMO version of this controller requires large errors on most of the coefficients or extremely large errors on a few coefficients in order for the controller to have unstable regions.

Figure 5-16. Regions of Stable PCC Control with a Given Model Error



## 5.4 The MIMO PCC Algorithm

The development of the stability analysis for the MIMO PCC system is similar to that given in Section 5.2 for the MIMO EWMA Controller, with an additional state for the average trend or slope. This minor addition greatly complicates the algebra, but the ideas remain the same. The system of equations described by equations (5-22) through (5-28) are repeated here in their equivalent matrix forms. Assuming the MIMO output of the plant system can be described as

$$\underline{y}_p[n] = \begin{bmatrix} \alpha_1 \\ \dots \\ \alpha_m \end{bmatrix} + \begin{bmatrix} \beta_{11} & \dots & \beta_{1n} \\ \dots & \dots & \dots \\ \beta_{m1} & \dots & \beta_{mn} \end{bmatrix} \begin{bmatrix} u_1[n] \\ \dots \\ u_n[n] \end{bmatrix} = \underline{\alpha} + \mathfrak{B}\,\underline{u}[n], \qquad (5\text{-}40)$$

the PCC controller assumes a model system

$$\underline{y}_m[n] = \begin{bmatrix} a_1[n] \\ \dots \\ a_m[n] \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \dots & \dots & \dots \\ b_{m1} & \dots & b_{mn} \end{bmatrix} \begin{bmatrix} u_1[n] \\ \dots \\ u_n[n] \end{bmatrix} = \underline{a}[n] + B\underline{u}[n], \qquad (5\text{-}41)$$

where *a[n]* is the PCC prediction of the constant term which is updated as

$$\underline{a}[n] = \underline{v}[n] + \underline{s}[n], \text{ where} \tag{5-42}$$

$$\underline{v}[n] = diag(\underline{w}_1)(\underline{y}_p[n] - B\underline{u}[n]) + diag(1 - \underline{w}_1)\underline{a}[n-1], \tag{5-43}$$

$$\underline{s}[n] = diag(\underline{w}_2)\underline{c}[n] + diag(1 - \underline{w}_1)\underline{s}[n-1], \text{ and} \tag{5-44}$$

$$\underline{c}[n] = (\underline{y}_p[n] - B\underline{u}[n]) - \underline{v}[n]. \tag{5-45}$$

The most significant change to the SISO case is the determination of the new equipment settings (similar to the EWMA controller). Again, the matrix B is not necessarily invertible and since the matrix B typically represents an underdetermined set of equations we use a minimum change solution:

$$\underline{u}[n+1] = B^T(BB^T)^{-1}(\underline{T} - \underline{a}[n]) + (I_n - B^T(BB^T)^{-1}B)\underline{u}[n]. \tag{5-46}$$

Theoretical determination of stability for MIMO PCC control using methods similar to [Ingolfsson] becomes an unmanageable task even for the SISO PCC Controller. Again we will see that with the state space representation one can indeed generate a matrix whose eigenvalues provide exact information regarding the stability of the system. Theoretically determining the limits of coefficient mismatch for a stable system is still difficult, but again it is relatively simple to determine the stability for any given system being controlled with the MIMO PCC algorithm.

The state space representation of the MIMO PCC algorithm we will explore is similar to that given in the development of the stability for the MIMO EWMA Controller. It is important to understand that this characterization, like all state space representations, is not unique. We feel that this particular representation minimizes the algebra and the resulting matrix.

We begin as before and let the state space representation be:

$$\begin{bmatrix} \underline{x}_1[n] \\ \underline{x}_2[n] \\ \underline{x}_3[n] \end{bmatrix} = \begin{bmatrix} \underline{v}[n-1] \\ \underline{s}[n-1] \\ \underline{u}[n] \end{bmatrix}. \tag{5-47}$$

Now using (5-40) through (5-46) we find that:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \\ \underline{x}_3[n+1] \end{bmatrix} = \begin{bmatrix} W_1(\underline{y}_p[n] - B\underline{u}[n]) + (I_m - W_1)\underline{v}[n-1] \\ W_2((\underline{y}_p[n] - B\underline{u}[n]) - \underline{v}[n]) + (I_m - W_2)\underline{s}[n-1] \\ M(\underline{T} - \underline{a}[n]) + (I_n - MB)\underline{u}[n] \end{bmatrix}, \tag{5-48}$$

where $W_1 = diag(\underline{w}_1)$, $W_2 = diag(\underline{w}_2)$ and $M = B^T(BB^T)^{-1}$.

Then using (5-40) to substitute in for the plant and using (5-47) to replace equation variables with state variables, we obtain:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \\ \underline{x}_3[n+1] \end{bmatrix} = \begin{bmatrix} W_1\underline{\alpha} + W_1(\mathfrak{I} - B)\underline{x}_3[n] + (I_m - W_1)\underline{x}_1[n] \\ W_2\underline{\alpha} + W_2(\mathfrak{I} - B)\underline{x}_3[n] - W_2\underline{x}_1[n+1] + (I_m - W_2)\underline{x}_2[n] \\ M\underline{T} - M(\underline{x}_1[n+1] + \underline{x}_2[n+1]) + (I_n - MB)\underline{x}_3[n] \end{bmatrix}. \tag{5-49}$$

The $\underline{x}_2[n+1]$ term can be simplified by substituting in $\underline{x}_1[n+1]$ as follows:

$$\underline{x}_2[n+1] = W_2\underline{\alpha} - W_2W_1\underline{\alpha} + W_2(\mathfrak{I} - B)\underline{x}_3[n] - W_2W_1(\mathfrak{I} - B)\underline{x}_3[n] - W_2(I_m - W_1)\underline{x}_1[n] + (I_m - W_2)\underline{x}_2[n] \tag{5-50}$$

$$\underline{x}_2[n+1] = W_2(I_m - W_1)\underline{\alpha} - W_2(I_m - W_1)\underline{x}_1[n] + (I_m - W_2)\underline{x}_2[n] + W_2(I_m - W_1)(\mathfrak{I} - B)\underline{x}_3[n]. \tag{5-51}$$

Note that,

$$\underline{x}_1[n+1] + \underline{x}_2[n+1] = \underline{x}_1[n+1] + W_2\underline{\alpha} + W_2(\mathfrak{I} - B)\underline{x}_3[n] - W_2\underline{x}_1[n+1] + (I_m - W_2)\underline{x}_2[n], \tag{5-52}$$

which can be simplified as follows:

$$\underline{x}_1[n+1] + \underline{x}_2[n+1] = W_2\underline{\alpha} + W_2(\mathfrak{I} - B)\underline{x}_3[n] + (I_m - W_2)\underline{x}_1[n+1] + (I_m - W_2)\underline{x}_2[n] \tag{5-53}$$

$$\underline{x}_1[n+1] + \underline{x}_2[n+1] = K_1 + (\mathfrak{I} - B)(W_2 + (I_m - W_2)W_1)\underline{x}_3[n] + (I_m - W_2)(I_m - W_1)\underline{x}_1[n] + (I_m - W_2)\underline{x}_2[n], \tag{5-54}$$

where $K_1 = (I_m - W_2)W_1\underline{\alpha} + W_2\underline{\alpha}$.

Using this, $\underline{x}_3[n+1]$ can be simplified by substituting (5-54) in the expression for $\underline{x}_1[n+1] + \underline{x}_2[n+1]$, and letting $\Psi_1 = I_m - W_1$ and $\Psi_2 = I_m - W_2$ :

$$\underline{x}_3[n+1] = M\underline{T} - M(K_1 + (\mathfrak{I} - B)(W_2 + \Psi_1 W_1)\underline{x}_3[n] + \Psi_2\Psi_1\underline{x}_1[n] + \Psi_2\underline{x}_2[n]) + (I_n - MB)\underline{x}_3[n] \tag{5-55}$$

$$\underline{x}_3[n+1] = M(\underline{T} - K_1) - M\Psi_2\Psi_1\underline{x}_1[n] - M\Psi_2\underline{x}_2[n] + ((I_n - MB) - M(\mathfrak{I} - B)(W_2 + \Psi_2 W_1))\underline{x}_3[n]. \tag{5-56}$$

Combining (5-49), (5-51), and (5-56) we see that we can write (5-49) in the state space notation:

$$\begin{bmatrix} \underline{x}_1[n+1] \\ \underline{x}_2[n+1] \\ \underline{x}_3[n+1] \end{bmatrix} = A\begin{bmatrix} \underline{x}_1[n] \\ \underline{x}_2[n] \\ \underline{x}_3[n] \end{bmatrix} + \tilde{B}\begin{bmatrix} \tilde{\underline{u}}_1[n] \\ \tilde{\underline{u}}_2[n] \end{bmatrix}, \text{ where} \tag{5-57}$$

$$A = \begin{bmatrix} (I_m - W_1) & 0 & W_1(\Im - B) \\ -W_2(I_m - W_1) & (I_m - W_2) & W_2(\Im - B)(I_m - W_1) \\ -M(I_m - W_2)(I_m - W_1) & -M(I_m - W_2) & (I_n - MB) - M(\Im - B)(W_2 + (I_m - W_2)W_1) \end{bmatrix}, \qquad (5\text{-}58)$$

$$\tilde{B} = \begin{bmatrix} W_1\underline{\alpha} & 0 \\ W_2(I_m - W_1)\underline{\alpha} & 0 \\ -MK_1 & M \end{bmatrix}, \text{ and} \qquad (5\text{-}59)$$

$$\begin{bmatrix} \tilde{\underline{u}}_1[n] \\ \tilde{\underline{u}}_2[n] \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \underline{T} \end{bmatrix}. \qquad (5\text{-}60)$$

From this, we can see that the inputs to the state space system (not the equipment settings) consist of a constant vector of *r* ones, where *r* is the number of outputs, augmented with the target vector.

Using the one-step transition matrix, *A*, we can test the stability of the PCC controller by simply checking that the magnitude of all the eigenvalues of the transition matrix are not greater than 1. Again, here we have added the equipment settings as states, thus we expect *q*-*r* ones and *r* zeros as eigenvectors (refer to Section 5.2). The remaining eigenvectors will determine the asymptotic stability of the outputs.

## 5.4.1 MIMO PCC Example 1

The first example demonstrates the manner in which one can determine, after having chosen a controller model, how much coefficient error is tolerable (whether due to modeling error or changes in the plant during operation.) This will demonstrate, just as in the EWMA Controller, that a major change in the most significant coefficient(s) of the system is necessary to cause instability. Unfortunately, there does not appear to be any rule of thumb in determining when the system will go unstable other than simply that the model error must be large.

The example we consider is a 4 input, 2 output system. We utilize the controller model: $B = \begin{bmatrix} 28 & -20 & 172 & 8 \\ 14.4 & 30 & 29 & -12 \end{bmatrix}$ and the PCC weight vectors $w_1 = \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}$ and $w_2 = \begin{bmatrix} 0.7 \\ 0.6 \end{bmatrix}$ for the controller. We will vary the coefficients of the plant in order to examine stability for different plant and model discrepancies.

### *5.4.1.1 A Stable Response*

The first case we consider is one that is stable. The associated input matrix for the plant is:

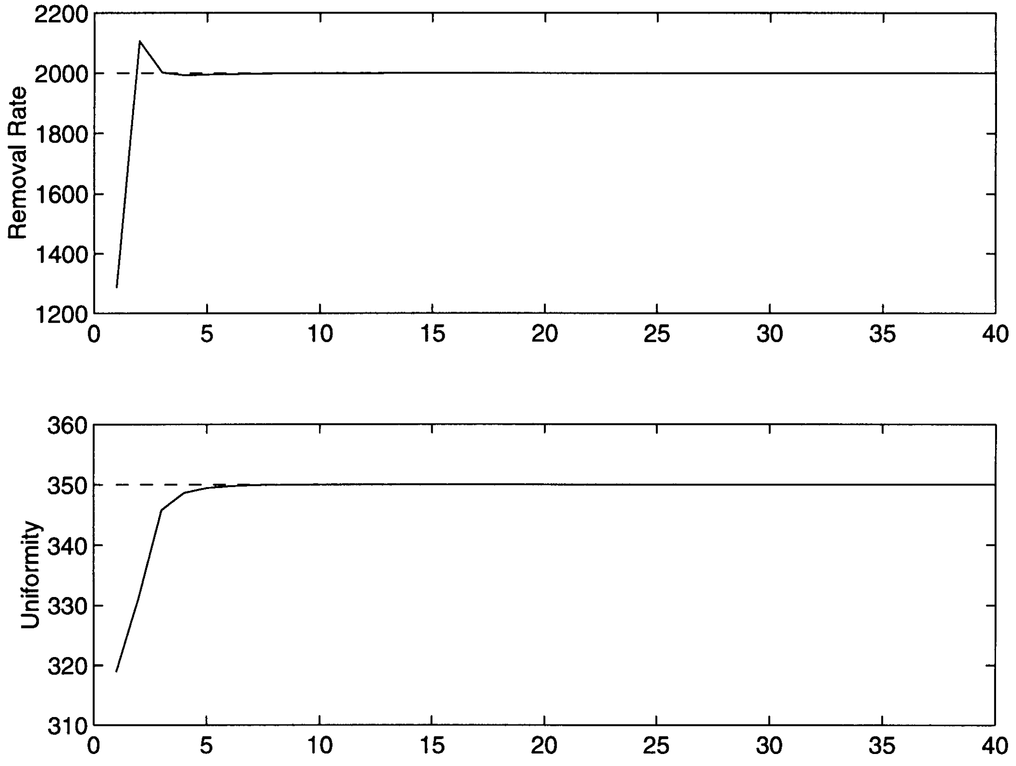
$$\Im = \begin{bmatrix} 22 & -18 & 199 & 16 \\ 19 & 26 & 20 & -14 \end{bmatrix}.$$

The eigenvalues of the corresponding transition matrix, A, have magnitude:

$$|\Lambda| = diag(\begin{bmatrix}0.5683 & 0.5259 & 0.4853 & 0.1407 & 0 & 0 & 1 & 1\end{bmatrix}).$$

We see that the four eigenvalues remaining after discounting the $q$-$r$=2 ones and $r$=2 zeros are all less than one. Therefore, the outputs will be asymptotically stable. This is shown in Fig. 5-17 below, where our two outputs are labeled "removal rate" and "nonuniformty."

**Figure 5-17. The Response of the Stable System**



*5.4.1.2 A Nearly Unstable Response*

As one of the coefficients of the plant is increased well beyond twice that assumed by the model, the system approaches instability. The corresponding matrix is shown below:

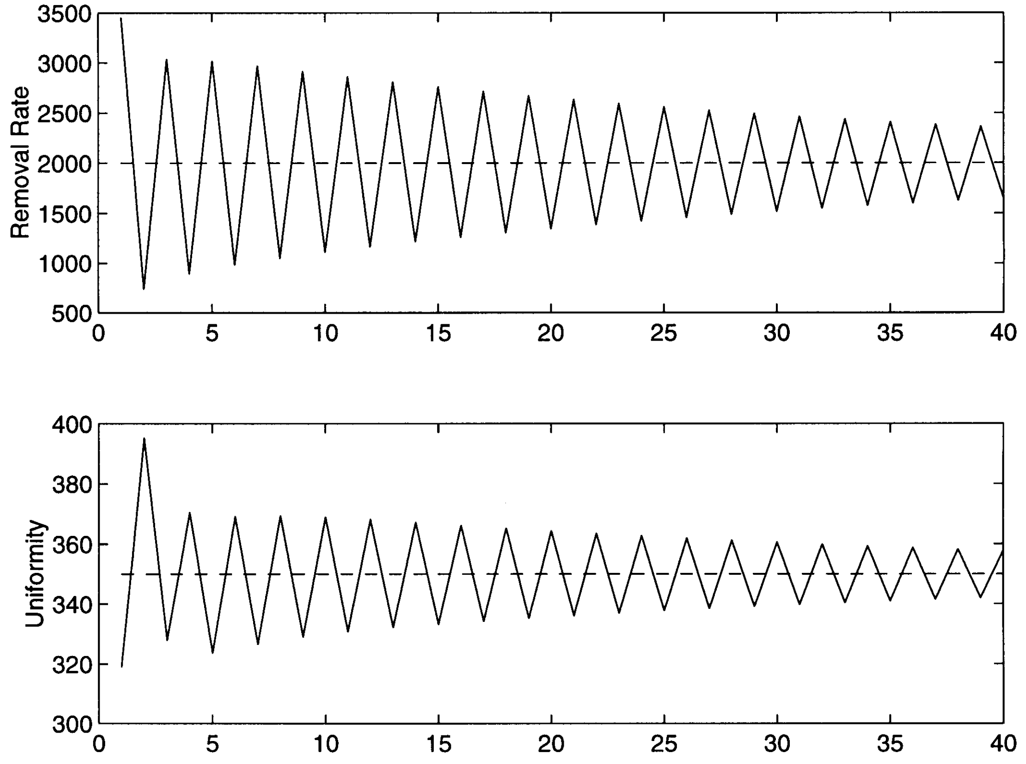
$$\Im = \begin{bmatrix}22 & -18 & 440 & 16\\ 19 & 26 & 20 & -14\end{bmatrix}.$$

The eigenvalues of the corresponding transition matrix, A, have magnitude:

$$|\Lambda| = diag(\begin{bmatrix}0.9696 & 0.5748 & 0.5748 & 0.4613 & 0 & 1 & 0 & 1\end{bmatrix}).$$

Notice that all the remaining eigenvalues corresponding to the average slope and average value state vectors have magnitude less than one, but that one of them is very close to 1. Therefore we expect the system to be near instability, as shown in Fig. 5-18.

**Figure 5-18. A Nearly Unstable Response Caused by a Large Increase in a Large Plant Coefficient**



*5.4.1.3 An Unstable Response Caused by a Large Increase in a Large Plant Coefficient*

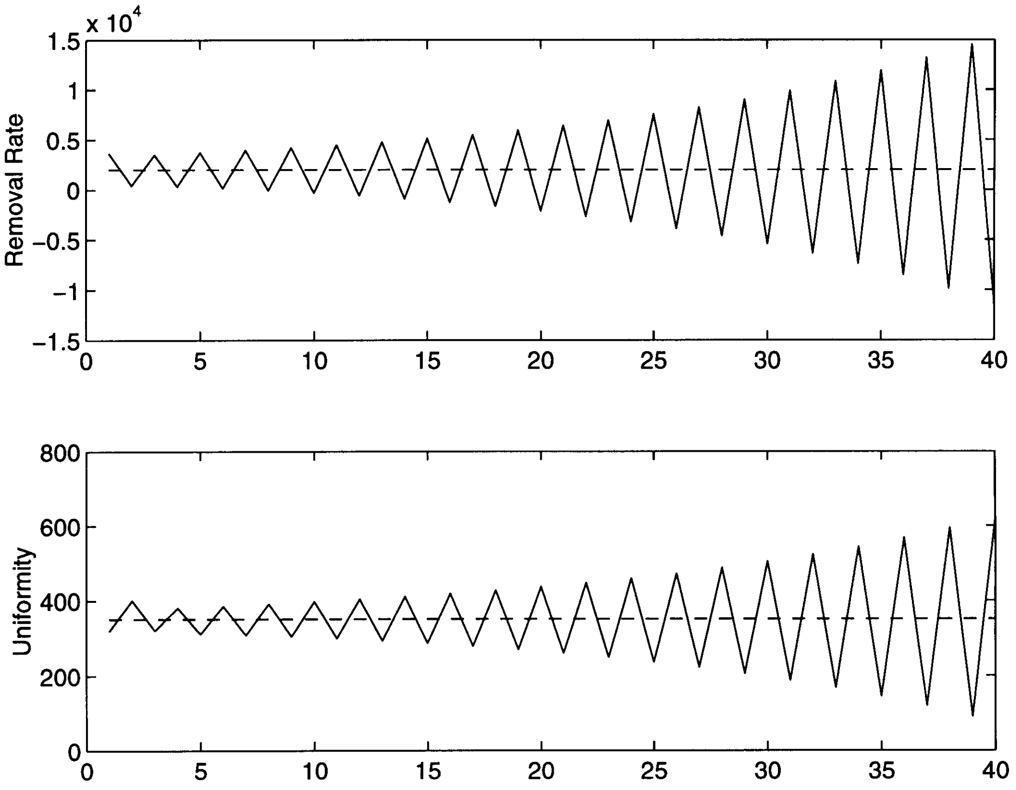
Further increasing the (1,3) coefficient finally causes the system to go unstable. Note, however that almost a 200% increase in the size of a large coefficient was necessary in order to cause the system to go unstable. Although one might propose to compare this with the $w\frac{\beta}{b}$ condition for the SISO EWMA controller, it seems a bit awkward since we have a prediction of the offset term, not an average. Also the added complexity of the MIMO system causes this rule of thumb to not apply so well due to the fact that stability is not guaranteed nor satisfied based on the errors of one single coefficient alone. However we can say that a large over-all amount of error seems necessary for unstable control, as demonstrated by another example. The coefficient matrix for this case is:

$\mathfrak{J} = \begin{bmatrix} 22 & -18 & 460 & 16 \\ 19 & 26 & 20 & -14 \end{bmatrix}$, which has eigenvalues with magnitude

$|\Lambda| = diag(\begin{bmatrix} 1.0599 & 0.5752 & 0.5752 & 0.4610 & 0 & 0 & 1 & 1 \end{bmatrix})$.

Since the magnitude of the first eigenvalue exceeds one, we conclude that the resulting response will be unstable as shown in Fig. 5-19.

**Figure 5-19. An Unstable Response Caused by a Large Increase in a Large Coefficient**



*5.4.1.4 A Stable Decrease in a Small Coefficient*

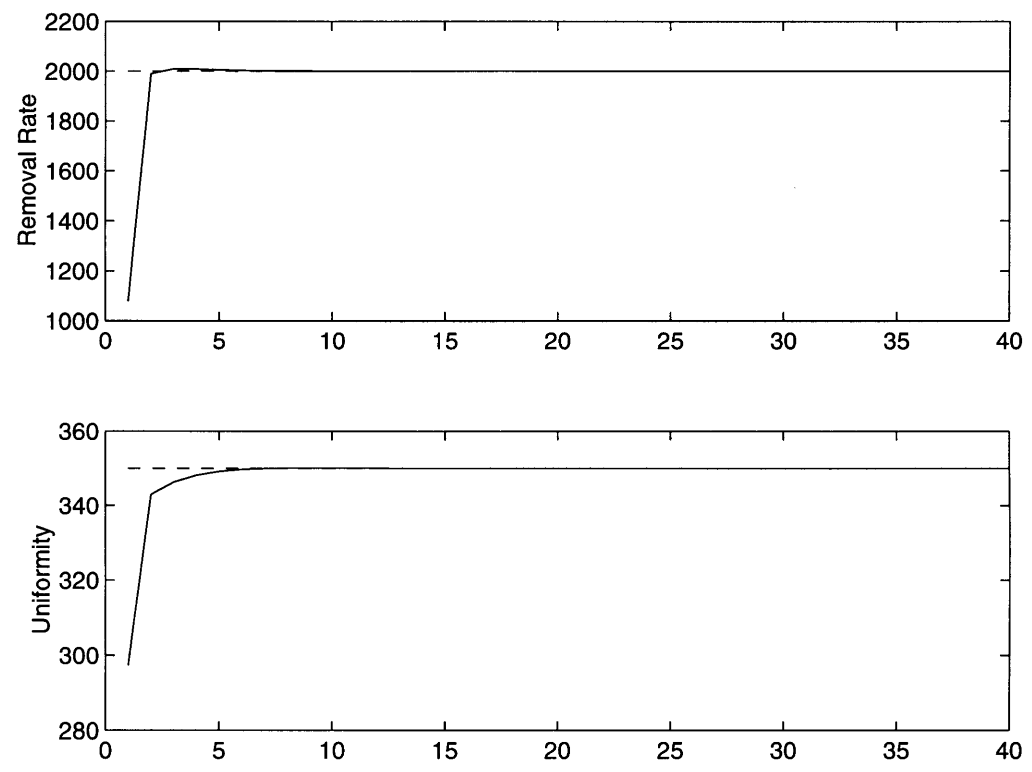
In the previous two cases one of the plant coefficients was increased, relative to its corresponding model coefficient. We will now consider decreases in the plant coefficients. In the first case, we force a coefficient small in magnitude, namely the (2,4) coefficient to be opposite in sign. As we will see this does not immediately imply that the system will be unstable as it did in the SISO system.

The corresponding input matrix is: $\mathfrak{J} = \begin{bmatrix} 22 & -18 & 176 & 16 \\ 19 & 26 & 20 & 10 \end{bmatrix}$ , which has eigenvalues

$|\Lambda| = diag(\begin{bmatrix} 0.2689 & 0.5083 & 0.6255 & 0.6255 & 0 & 0 & 1 & 1 \end{bmatrix})$.

Since the first four eigenvalues corresponding to the slope and value states are much less than one, we expect the resulting system to be stable. This can be seen in Fig. 5-20.

**Figure 5-20. A Stable Large Decrease in a Small Plant Coefficient**



*5.4.1.5 A Stable Decrease in a Large Coefficient*

Although the previous case considered a large relative decrease in a small coefficient, large decreases in a large coefficient (even a change in sign) does not guarantee that the resulting system will be unstable as it did in the SISO EWMA Controller. This can be seen by considering the following case.

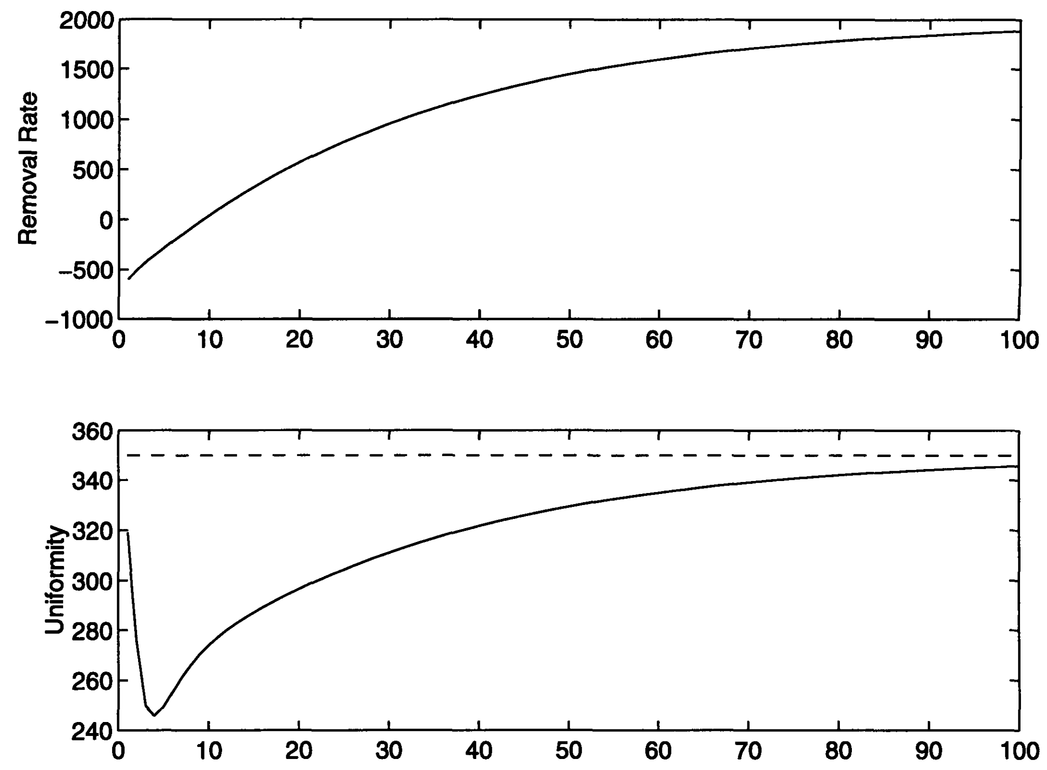
The large (1,3) coefficient of the plant is decreased past zero to -10:

$$\mathfrak{I} = \begin{bmatrix} 22 & -18 & -10 & 16 \\ 19 & 26 & 20 & -14 \end{bmatrix}$$, which has eigenvalues

$$|\Lambda| = diag(\begin{bmatrix} 0.4525 & 0.6616 & 0.5731 & 0.9688 & 0 & 0 & 1 & 1 \end{bmatrix}).$$

Although the fourth eigenvalue is nearing one, the system is still stable. When the plant coefficient is decreased relative to the controller model, the system response is slowed (note the time axis.) This can be seen in Fig. 5-21.

**Figure 5-21. A Slowed Response Caused by a Decrease in a Large Coefficient**



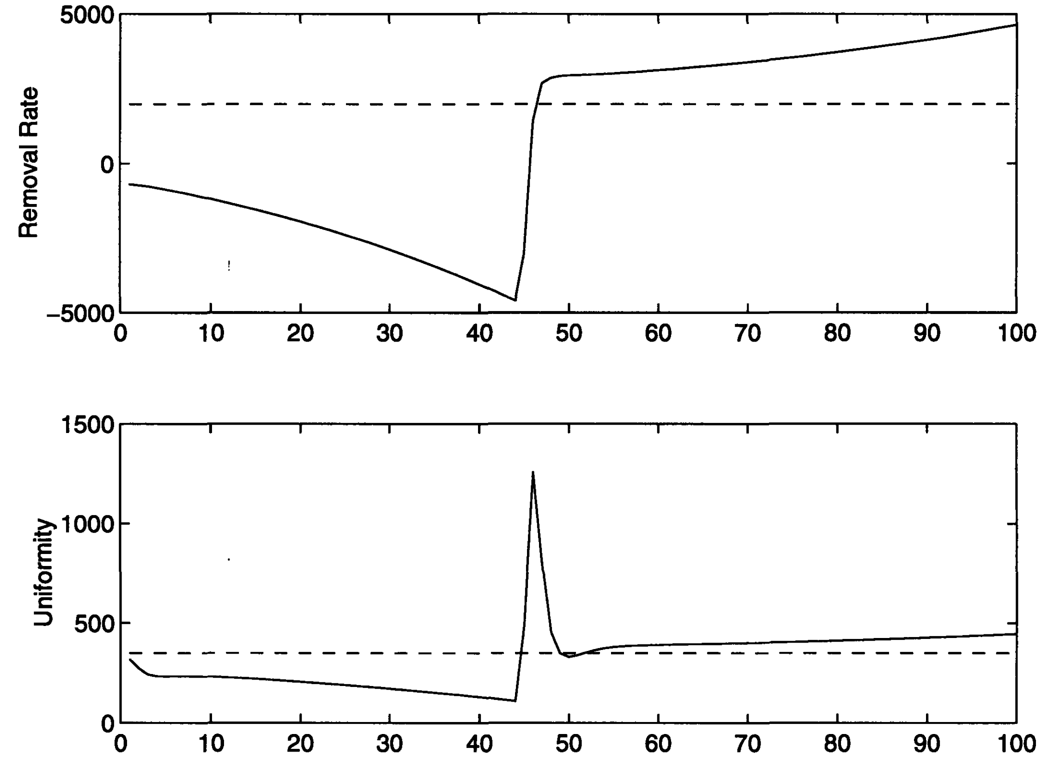
### *5.4.1.6 An Unstable Decrease in a Large Coefficient System*

Further decreases in a large coefficient eventually leads to instability. This can be seen by further decreasing the (1,3) coefficient in the previous example. The resulting plant coefficient matrix is: $\Im = \begin{bmatrix} 22 & -18 & -20 & 16 \\ 19 & 26 & 20 & -14 \end{bmatrix}$, which has eigenvalues

$|\Lambda| = diag(\begin{bmatrix} 0.6532 & 0.5732 & 0.4528 & 1.0216 & 0 & 0 & 1 & 1 \end{bmatrix})$.

As can be seen, the last eigenvalue of the average slope and value states exceeds one. Thus, we expect the unstable system shown in Fig. 5-22.

**Figure 5-22. An Unstable Decrease in a Large Coefficient**



*5.4.1.7 An Extremely Unstable Decrease in a Large Coefficient*

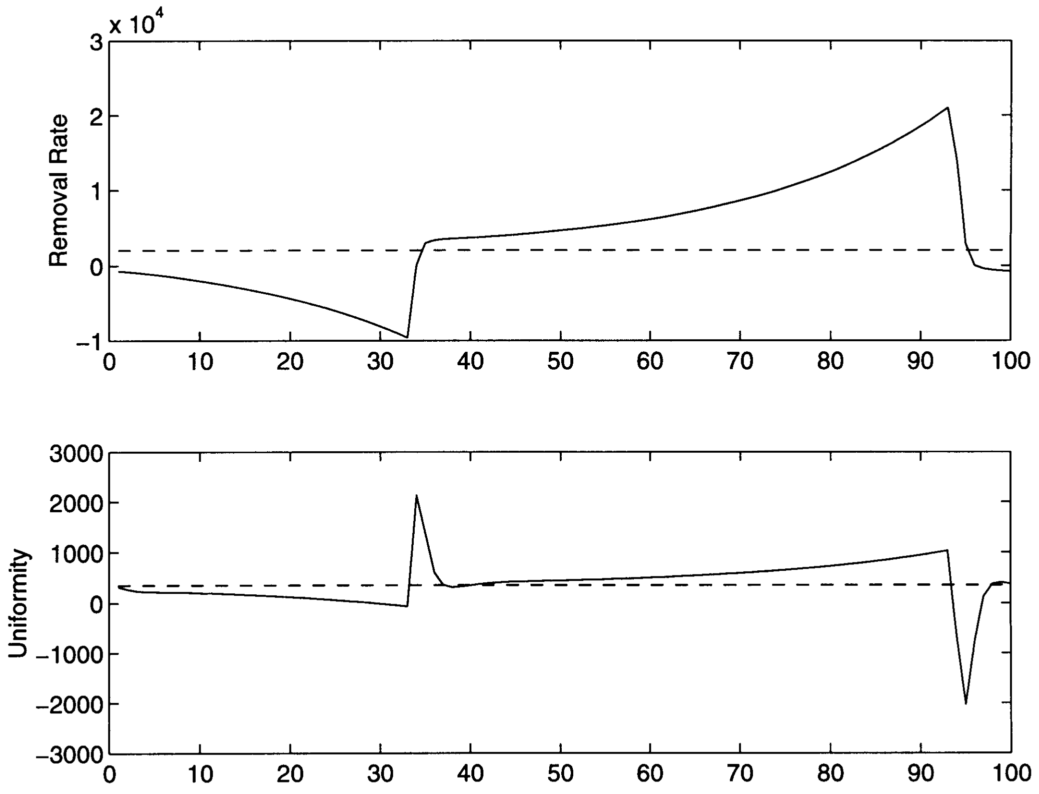
As a final case, we demonstrate that the severity of a system's instability can be extracted from the relative amount by which the magnitude of the eigenvalues exceed one. In the previous example, the magnitude of the exceeding eigenvalue was just over one. As can be seen in Fig. 5-22, the divergence of the system away from the targets was slow. In this example we demonstrate how an even further decrease in the (1,3) coefficient causes the magnitude of the exceeding eigenvalue to increase only slightly more.

$$\mathfrak{I} = \begin{bmatrix} 22 & -18 & -25 & 16 \\ 19 & 26 & 20 & -14 \end{bmatrix} \text{, has eigenvalues}$$

$$|\Lambda| = diag(\begin{bmatrix} 0.4530 & 0.5733 & 0.6499 & 1.0472 & 0 & 1 & 0 & 1 \end{bmatrix}).$$

Although there is only a slight increase in the eigenvalue, there is a dramatic increase in the instability. This demonstrates that, although large flexibility in the coefficients is tolerable for the controller to provide stable control, even small perturbations in the coefficients past the stability limits have a tremendous effect on the degree of instability. This can be seen in Fig. 5-23.

Figure 5-23. An Extremely Unstable Decrease in a Large Coefficient



## 5.5 Summary: Stability Analysis of the MIMO EWMA and PCC Controllers

Using the state space representation of the PCC controller algorithm, we can extract stability information for both the SISO and MIMO EWMA and PCC Controllers. Several examples were provided which demonstrated that, although exact formulas cannot be provided for the stability criteria of these controllers in terms of relative plant and controller model mismatch, stability can be determined for any given scenario. It was also shown that a great deal of insight into the stability issues of these controllers can be gained from simulation. Limits on coefficients for any system can be also be estimated through simulation. These estimations can be compared with the controller model uncertainty to provide estimations on the probability for unstable control. This work demonstrated this method for the underdetermined problem, i.e. the minimum distance recipe generation method. Although this is more common in many process, the framework provided here can be extended to include the exact and overdetermined situations as well. Finally, this method provides a reasonable way to develop bounds on the inputs of a given process by sampling the input space. Such bounds would otherwise be extremely expensive to compute by means of simulation, where the entire process must be simulated for several combinations of EWMA or PCC weights. In addition, a process which does not go unstable for a few hundred runs may not show up in simulation. Overall, these methods provide a fast and efficient method for determining the stability of the SISO or MIMO PCC algorithm.

# Chapter 6

# Analytical Derivation of the Optimal SISO EWMA Controller Weights for a Linear Drifting Process Buried in White Noise

Run by run control methods are receiving a great deal of attention as a means to improve and maintain the performance of modern semiconductor manufacturing processes, particularly in the areas of chemical-mechanical polishing and plasma etching [Boning1, Boning2, Jairath, Hu1, Hu2, Altman, Moyne3, Moyne4, Ingolfsson, Sachs, Butler, Hembree, White, DelCastillo, Chang]. The Exponentially Weighted Moving Average (EWMA) Controller provides the ability to control many processes with approximately affine models which are subject to approximately linear drifts. Several works have demonstrated methods for determining an optimal value for the EWMA weight when no control actions are being taken (open-loop) [Hunter, Crowder1, Lucas]. A method for determining the optimal value for the EWMA weight when control actions are being taken (closed-loop) using simulated data was shown in Section 4.5 and reported in [Boning2]. This chapter addresses the need for an analytical solution to this problem. The optimal EWMA weight for an EWMA Controller when used on a single-input single-output (SISO) affine process with a linear drift buried in white noise is derived in this chapter. This is an important step toward an analytical solution to the optimal weight problem and implementation of a self-tuning EWMA controller.

The analytical solution of the optimal EWMA weight for an affine process with a linear drift buried in white noise is presented in Section 6.1. Simulated verification of this is shown in Section 6.2 and Section 6.3 provides a brief conclusion.

## 6.1 Derivation of the Optimal EWMA Weight

The control architecture shown in Fig. 3-1 assumes the true process output, $y_p[n]$, is of the form:

$$y_p[n] = \alpha + \beta u[n] + \delta n + r[n], \tag{6-1}$$

where $\alpha$ is a constant offset, $\beta$ is a constant linear coefficient, $\delta$ is a constant drift rate, and *r[n]* is independent and identically distributed (i.i.d) noise with mean zero and variance $\sigma^2$. The EWMA controller assumes a model system:

$$y_c[n] = a[n] + bu[n], \tag{6-2}$$

where *b* is a constant linear coefficient. The *a[n]* term is an EWMA constant term which is updated as:

$$a[n] = w(y_p[n] - bu[n]) + (1-w)a[n-1], \tag{6-3}$$

where *w* is the EWMA weight, *u[n],* is the process control at discrete-time, *n*. The input *u[n]* is determined by solving the affine controller model for the input which causes the controller model output to equal the process target, *T*:

$$u[n+1] = \frac{1}{b}(T - a[n]). \tag{6-4}$$

The following derivation is for the SISO EMWA Controller being used on a process with a linear drift buried in white noise as formulated in (6-1). This derivation is broken into two parts. The first part is to find the solution of the controlled output at discrete-time, *n*. The second part is to determine the optimal EWMA weight based on some criteria. The criteria we will use is the mean squared error in the steady state output.

### 6.1.1 Determination of the Process Output at Discrete-Time n.

The first step in determining an optimal EWMA weight for controlling a linearly drifting process buried in white noise is to unwrap the recursion in the EWMA controller, leaving the output as a function of only discrete-time *n*. We begin by substituting the solution of the input at discrete-time *n* into the process output to obtain:

$$y_p[n] = \alpha + \frac{\beta}{b}T - \frac{\beta}{b}a[n-1] + \delta n + r[n]. \tag{6-5}$$

Similarly, by substituting the plant output equation into the EWMA of the offset term and cancelling the linear coefficients we obtain:

$$a[n] = wT\frac{\beta}{b} + w\alpha + w\delta n + wr[n] + w\frac{\beta}{b}(1-w)a[n-1]. \tag{6-6}$$

In order to obtain the solution of the process output at discrete-time $n$ we will rewrite the system of equations in state space notation, using the state vector:

$$\begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} = \begin{bmatrix} a[n-1] \\ n \end{bmatrix}. \tag{6-7}$$

The inclusion of discrete-time $n$ as a state is valid for two reasons. First, the EWMA controller is able to remove this effect from the output and this term will fall out of our equations. Second, the state only approaches infinity as discrete-time $n$ approaches infinity. It is clear that we will run our system only in finite time, and therefore the state will remain finite. Continuing on, we can write the state equations as follows:

$$\begin{bmatrix} x_1[n+1] \\ x_2[n+1] \end{bmatrix} = \begin{bmatrix} 1 - w\frac{\beta}{b} & w\delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} + \begin{bmatrix} w\alpha + wT\left(\frac{\beta}{b} - 1\right) + wr[n] \\ 1 \end{bmatrix}, \tag{6-8}$$

and the output as:

$$y_p[n] = \begin{bmatrix} -\frac{\beta}{b} & \delta \end{bmatrix} \begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} + \frac{\beta}{b}T + \alpha + r[n]. \tag{6-9}$$

The system is now in standard state space notation, i.e.:

$$\begin{aligned} \underline{x}[n+1] &= A\underline{x}[n] + B\underline{u}[n] \\ y[n] &= C\underline{x}[n] + D\underline{u}[n] \end{aligned}, \tag{6-10}$$

and thus from state space theory we know that:

$$\underline{x}[n] = A^n\underline{x}[0] + \sum_{l=0}^{n-1} A^{n-l-1}B\underline{u}[l]. \tag{6-11}$$

In our problem $B\underline{u}[n]$ is fixed for all discrete-time and $A$ may be diagonalized as:

$$A = P\Lambda P^{-1} = \begin{bmatrix} 1 & \frac{b}{\beta}\delta \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 - w\frac{\beta}{b} & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & \left(-\frac{b}{\beta}\delta\right) \\ 0 & 1 \end{bmatrix}, \tag{6-12}$$

and thus

$$A^n = P\Lambda^n P^{-1} = \begin{bmatrix} 1 & \frac{b}{\beta}\delta \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \left(1 - w\frac{\beta}{b}\right)^n & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & \left(-\frac{b}{\beta}\delta\right) \\ 0 & 1 \end{bmatrix}, \tag{6-13}$$

which can be combined to form:

$$A^n = \begin{bmatrix} \left(1 - w\frac{\beta}{b}\right)^n & \frac{b}{\beta}\delta\left(1 - \left(1 - w\frac{\beta}{b}\right)^n\right) \\ 0 & 1 \end{bmatrix}. \tag{6-14}$$

Using equations (6-11) and (6-14), and assuming that the initial value of the EWMA of the offset term is equal to the actual process offset $\alpha$,

$$\underline{x}[0] = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}, \tag{6-15}$$

we see that:

$$\underline{x}[n] = \begin{bmatrix} \alpha\left(1 - w\frac{\beta}{b}\right)^n \\ 0 \end{bmatrix} + \sum_{l=0}^{n-1}\left(\left(wT\left(\frac{\beta}{b} - 1\right) + w\alpha + wr[l] - \frac{b}{\beta}\delta\right)\left(1 - w\frac{\beta}{b}\right)^n + \frac{b}{\beta}\delta\right). \tag{6-16}$$

Now, using equation (6-9) we see that the output of the system at discrete-time $n$ is:

$$y_p[n] = \alpha + \frac{\beta}{b}T + r[n] - \alpha\frac{\beta}{b}\left(1 - w\frac{\beta}{b}\right)^n - \sum_{l=0}^{n-1}\frac{\beta}{b}\left(wT\left(\frac{\beta}{b} - 1\right) + w\alpha + wr[l] - \frac{b}{\beta}\delta\right)\left(1 - w\frac{\beta}{b}\right)^{n-l-1}. \tag{6-17}$$

Notice that as $n \to \infty$, the third term in (6-17), as well as the sum, will grow unbounded unless:

$$\left|1 - w\frac{\beta}{b}\right| < 1\text{, or } 0 < w\frac{\beta}{b} < 2. \tag{6-18}$$

This is the well-known stability criteria for the EWMA controller [Ingolfsson]. It seems reasonable to assume that the optimal EWMA weight will be one that is in the stable range, and thus we continue with this assumption.

### 6.1.2 Minimize Steady-State Mean-Squared Error

At this point we must decide on a criteria to use in determining the optimal EWMA weight. We choose to minimize the mean-squared error as $n \to \infty$. Therefore, the cost function is of the form:

$$J = \min_{w} E\{(y_p[n] - T)^2\}, \tag{6-19}$$

where the E{ } represents the statistical expectation operator. By considering (6-17) as $n \to \infty$ we see that $\alpha\frac{\beta}{b}\left(1 - w\frac{\beta}{b}\right)^n$ dies away. The sum may be broken up into a random portion and a non-random portion. The non-random portion is a geometric sum which converges given that the system is stable. Thus:

$$y_p[n] = T + r[n] - \frac{b\delta}{\beta w} + \sum_{l=0}^{n-1} w\frac{\beta}{b}\left(1 - w\frac{\beta}{b}\right)^{n-l-1} r[l], \text{ or} \tag{6-20}$$

$$y_p[n] - T = r[n] - \frac{b\delta}{\beta w} + \sum_{l=0}^{n-1} w\frac{\beta}{b}\left(1 - w\frac{\beta}{b}\right)^{n-l-1} r[l]. \tag{6-21}$$

Now squaring this and taking the expectation while noting that $r[n]$ is a zero-mean i.i.d. random process (cross terms fall out with the expectation), we see that:

$$E\{(y_p[n] - T)^2\} = \sigma^2 + \frac{b^2\delta^2}{\beta^2 w^2} + E\left\{\left(\sum_{l=0}^{n-1} w\frac{\beta}{b}\left(1 - w\frac{\beta}{b}\right)^{n-l-1} r[l]\right)^2\right\}.$$

Since $r[n]$ is a zero-mean i.i.d. random process, the contents of the expectation is the variance of a sum of i.i.d. random variables. Therefore:

$$E\{(y_p[n] - T)^2\} = \sigma^2 + \frac{b^2\delta^2}{\beta^2 w^2} + \sigma^2 w^2 \frac{\beta^2}{b^2} \sum_{l=0}^{n-1} \left(\left(1 - w\frac{\beta}{b}\right)^2\right)^{n-l-1}. \tag{6-22}$$

The sum is another convergent geometric series, and thus:

$$E\{(y_p[n] - T)^2\} = \sigma^2 + \frac{b^2\delta^2}{\beta^2 w^2} + \frac{w\beta\sigma^2}{2b - w\beta}. \tag{6-23}$$

Finally, we can determine the optimal weight as:

$$\hat{w}_{opt} = \underset{w}{\arg min}\left[\sigma^2 + \frac{b^2\delta^2}{\beta^2 w^2} + \frac{w\beta\sigma^2}{2b - w\beta}\right], \tag{6-24}$$

We see that if the model error is approximately equal to zero, i.e. $b = \beta$, this expression can be simplified to:

$$\tilde{w}_{opt} = \underset{w}{\arg min}\left[\sigma^2 + \frac{\delta^2}{w^2} + \frac{w\sigma^2}{2 - w}\right] \tag{6-25}$$

which by taking the derivative with respect to $w$ and setting this equal to zero, we find that the optimal EWMA weight is a root to the third order polynomial in $w$ (restricted to the range (0,1]):

$$0 = 2w^3\sigma^2 - 2\delta^2 w^2 - 8\delta^2 w - 8\delta^2. \tag{6-26}$$

The general solution takes on a similar form as can be seen by rewriting (6-24) as:

$$\hat{w}_{opt} = \underset{w}{\arg min}\left[\sigma^2 + \frac{\delta^2}{(w\xi)^2} + \frac{(w\xi)\sigma^2}{2 - (w\xi)}\right], \text{ where } \xi = \frac{\beta}{b}. \tag{6-27}$$

The the optimal EWMA weight in the general case is a root to another third order polynomial in $w$ (restricted to the range (0,1]):

$$0 = 2\sigma^2\xi^3 w^3 - 2\delta^2\xi^2 w^2 + 8\delta^2\xi w - 8\delta^2, \tag{6-28}$$

An important observation, more importantly, seen by comparing (6-25) and (6-27) is that the optimal EWMA for the model error case is nothing more than the optimal weight for the zero model error case scaled by the amount of model error:

$$\hat{w}_{opt} = \frac{\tilde{w}_{opt}}{\xi}. \tag{6-29}$$

This is an amazingly simple result that is not at all obvious at the start of the derivation. We can qualitatively verify this result by considering two cases. First, if the linear coefficient of the actual process is larger than the model coefficient ($\xi > 1$), then the system would tend to respond more to changes in the recipes. This would tend to increase the small amount of overshoot and thus increase the process variability. This would be similar to increased noise and thus the optimal EWMA weight would decrease, which is exactly what (6-29) states. On the other hand, if the actual coefficient were smaller than that of the model coefficient ($\xi < 1$), then the system would not tend to be as responsive as the controller would expect. Thus the offset due to the controller's inability to adequately keep up with the drift rate would increase. Therefore, in order to compensate, the optimal EWMA weight would have to increase. Again, this is exactly what is prescribed by (6-29). The exception to this is when the zero model error optimum is at one, because the optimum must always be bounded in the range (0,1]. Due to this bounding problem the roots should always be generated using the general equation, but ordinarily this rule of thumb holds.

Using the above approach, we can determine the optimal EWMA weight given any amount of drift $\delta$, noise with standard deviation $\sigma$, and the proportion of model mismatch $\xi$. Solutions of this form are easily computed. The surface plot for the optimal EWMA weights as a function of drift and standard deviation of the noise (with a small model error $\xi = 1.05$) is shown in Fig 6-1 over a fairly large range of drift and noise. We see that the resulting weight corresponds well to our expectation for an EMWA controller. For small amounts of drift and any amount of noise, the optimum weight lies near zero (maximum filtering). As the amount of drift increases relative to the amount of noise, the optimum EWMA weight increases until it reaches a maximum at one.

On the other hand, if the model error $\xi$ is increased to 1.5, we see in Fig. 6-2 that the optimal weight decreases over the entire surface (except along the zero noise line). Finally, we see in Fig. 6-3 that when $\xi$ is decreased to 0.5, the entire surface is raised (except along the zero noise line).

**Figure 6-1. Optimal EWMA Weight ($\xi = 1.05$)**

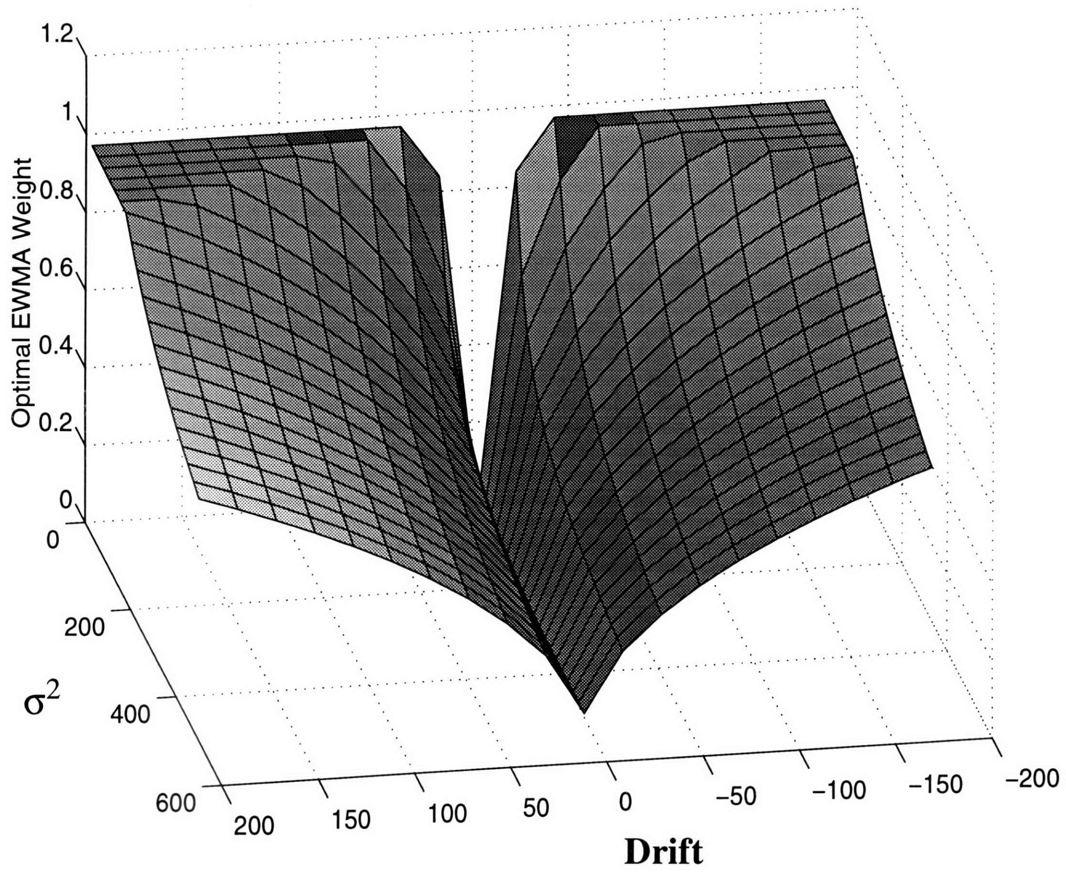


Figure 6-2. Optimal EWMA Weight ($\xi = 1.5$)

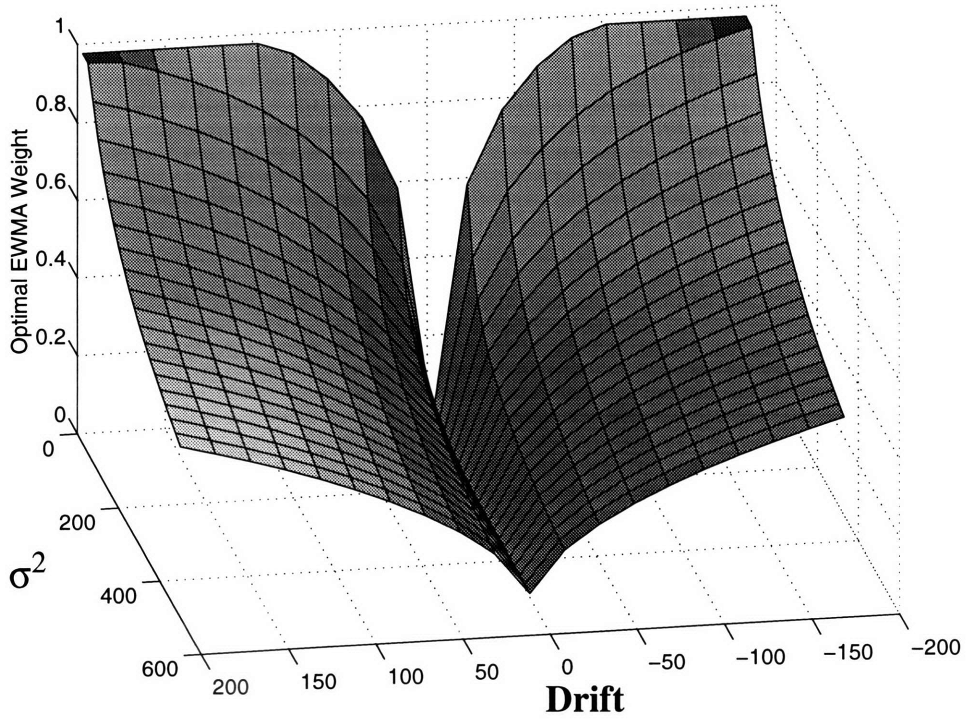
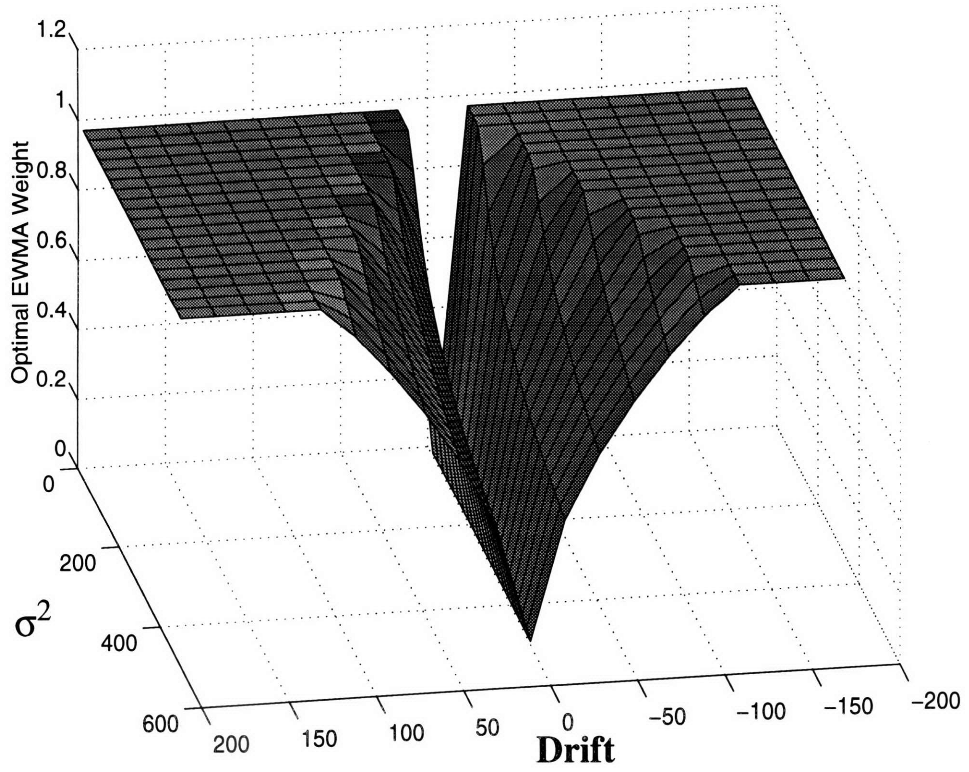


Figure 6-3. Optimal EWMA Weight ($\xi = 0.5$)



## 6.2 Simulation Results

In order to verify the previously derived result, simulations were performed on an affine process with a linear drift buried in white noise. The EWMA controller was used and the MSE for the simulation run was measured. This was performed for 20 values of $w$ for each of 20 drift values and 20 standard deviations. The result given above was also determined at these points and the resulting contour plots (for $\xi = 1$) are shown together in Fig. 6-4. We see that almost all the contour lines are very close. Irregularities in the simulations are due to the specific noise used in the simulations and the discrete number of points on which simulations were performed. The theoretical solid lines represent the "expected" optimal EWMA weights; these were also evaluated at discrete points which explains the slight kinks in the theoretical curves. Simulations were also performed to ensure that the simulated optimums do indeed fluctuate around the theoretical expected optimum EWMA weight. These simulations were repeated with 50% model error ($\xi = 1.5$), and the resulting contour plots are shown in Fig. 6-5. Again, we see almost perfect correlation between the theoretical and simulated drift and noise responses for teh system.

**Figure 6-4. Contour Plots for the Optimum EWMA Weight ($\xi = 1$)**
**Solid-Theoretical DashDot-Simulated**

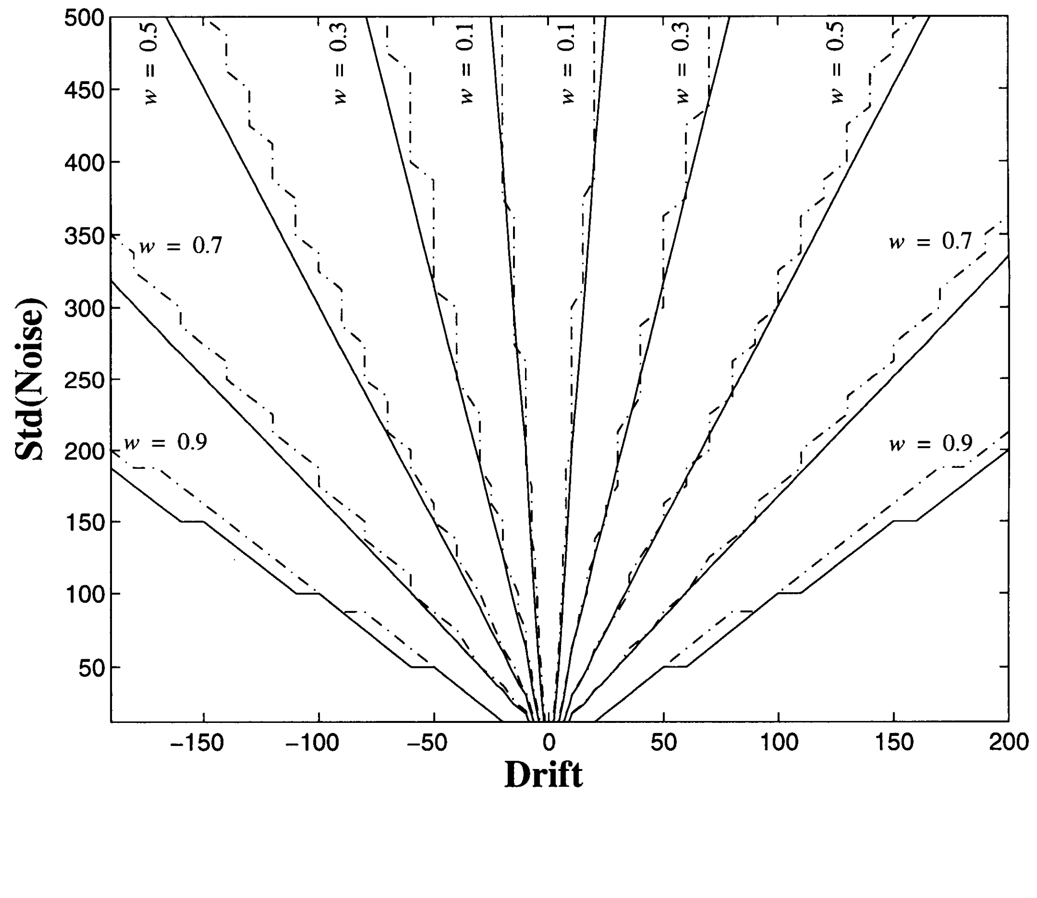
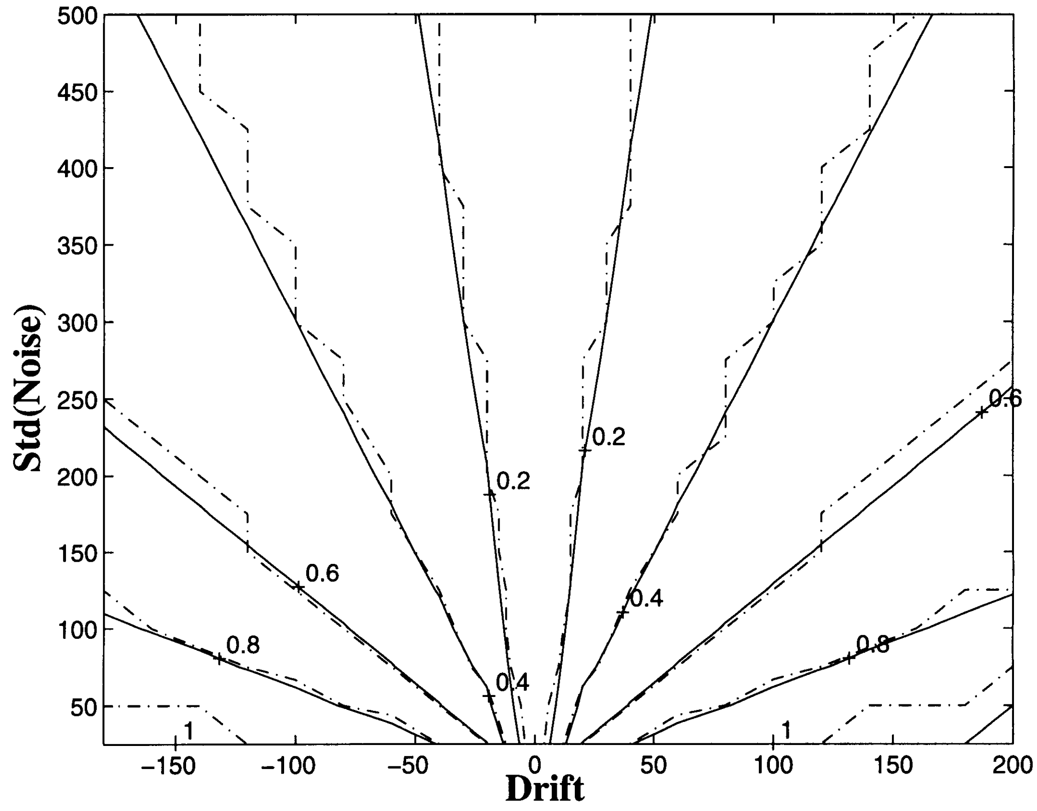


**Figure 6-5. Contour Plots for the Optimal EWMA Weight ($\xi$ = 1.5)**
**Solid-Theoretical DashDot-Simulated**



## 6.3 Summary: Analytical Derivation of the Optimal EWMA Weights

This work demonstrates that it is possible to analytically determine the proper EWMA weight for steady state optimal control with an EWMA controller. Similar derivations for the multiple-input multiple-output EWMA controller and the Double EWMA Controller (Predictor Corrector Control) are needed for both drifting processes and those susceptible to shifts. As a final application of this research, future work will consider methods for dynamic adaptation of the EWMA weight in the EWMA Controller using the theoretical optimal values. This would provide a self-tuning EWMA Controller, thereby removing the necessity for an experienced user in order to obtain optimal performance from an EWMA Controller. A self-tuning EWMA Controller is considered in Chapter 8 using simulated optimal weights for the MIMO case (since the theoretical optimal weights for MIMO case have yet to be determined). This controller uses simulated data to construct a neural network map from the output noise and drift to the optimal weights.

# Chapter 7

# Summary of EWMA Based Control Methods

The work presented above demonstrates that EWMA based control methods have the ability to effectively control many processes which may be approximately modeled by a set of affine equations. In addition, we have seen that these methods have several valuable properties. Their response to unmodeled disturbances, such as random shifts and linear drifts in the presence of noise, is quite good. This is particularly important for processes like CMP which have disturbances that are hard to model with simple dynamics. Another important aspect of EWMA based controllers is that the stability of these algorithms may be easily determined by finding the eigenvalues of a matrix (as outlined above). By considering several examples, it was shown that these controllers are stable over a large region of the parameter space even with a large degree of model mismatch. In addition to these features, we have seen several methods to resolve practical issues involved with the implementation of these controllers into an industrial quality architecture. It was shown that *ad hoc* methods for implementing bounds and discretization of the process inputs can be easily incorporated while minimizing errors. User preferences for input variation and output error trade-off are easily implemented as well. Since the performance of these controllers is highly dependent on the choice of their parameters, perhaps the most important practical issue is that of parameter selection. It was shown that we can find the optimal parameters through simulation and thus tune these controllers. It was also demonstrated that an analytical solution of the optimal weights exists for the SISO EWMA control of a linearly drifting process buried in white noise. This provides a first step to understanding how to analytically determine the optimal parameters for the general MIMO case in which shifts as well as drifts occur. More work needs to be done to provide this general solution. This is extremely important for the development of an on-line self-tuning EWMA based controller. Such a controller would have a tremendous impact on the semiconductor industry, where equipment and disturbance behaviors change in a random fashion and equipment operators are not trained to tune feedback controllers.

Although these many advantages of EWMA based controllers warrant the use of these controllers in many situations, we cannot forget that there are limitations in the underlying assumptions which these controllers are based on. One issue is the assumption that the process may be effectively modeled with an affine set of equations. If small nonlineararities are present, the EWMA based controllers can still effectively control the process. However, large nonlineararities generally appear as a large amount of model error and thus the EWMA controllers are prone to instability. A second case where these controllers often do not provide an optimum response is when there are significant time dynamics to the process which can be appropriately modeled. The EWMA controller provides an effective way to compensate for time dynamics which are poorly modeled, but when significant or well modeled time dynamics exist the EWMA based controllers do not capture relevant information which may be used to improve control. A third case where these controllers fall short is in the face of large amounts of data. This case has not been an issue with CMP process control until recent advances in sensor technology have indicated that large amounts of information pertaining to the wafer state may be captured on every run. With this case, extracting the optimal states from the large amount of data may be complex. Failure to capture such information may reduce the effectiveness of using an EWMA based controller, even if an affine model can be fit. In light of these limitations, we next turn our attention to controllers of a different type.

# Part III - Neural Network Controllers

Part III examines controllers which address two important issues. First, adaptive or self-tuning controllers are needed which include the ability to compensate for model error or unmodeled process dynamics. Second, nonlinear controllers are needed for many processes.

In addition, Part III provides a comparitive analysis of the performance of each controller described in this thesis. The final chapter concludes with a discussion of the results of this thesis and suggestions for future work.

# Chapter 8

# A Direct Adaptive EWMA Controller Utilizing Artificial Neural Network Function Approximation Techniques

This first chapter of Part III begins a transition from EWMA based feedback controllers to artificial neural network (ANN) model based adaptive approaches. It does so by keeping an EWMA controller as a core and utilizing an ANN approximation of the optimal weights to update the controller.

## 8.1 Introduction

Section 4.5 emphasized the necessity of properly choosing the EWMA weighting parameters. If we could determine an analytical solution for the optimal weights, we could have the controller choose its own weights on-line, without the need for an experienced engineer to tune the controller. In Section 6, it was demonstrated that there is an analytical solution for determining the optimal weight for controlling a linearly drifting SISO process buried in white noise. A similar derivation for the MIMO case becomes extremely complicated. As such, a self-tuning controller using an analytical function is not currently feasible. In response to the need for a self-tuning EWMA controller, we have developed an ANN system to provide on-line adaptation of the weighting parameter. The control system consists of a neural network (trained with empirical data) which serves as a mapping between the process disturbance state (drift and noise) and the corresponding optimal weighting parameter. The ANN is used to generate estimates of the optimal EWMA weighting parameters, thereby serving as an approximation to the optimal weights. This allows the on-line tuning of the EWMA controller for cases where the drift rates and noise levels change throughout the process or when the parameters were poorly chosen at the start of the process. We will see that this system can reduce noise added by the controller when a small drift is present (by reducing the EWMA weight), and decrease offset errors caused by a lack of control when a large drift is present (by increasing the EWMA weight).

Section 8.2 outlines how the method for determining the optimal weights for a given process (demonstrated in Section 4.5) can be expanded to form a function mapping from the disturbance state to the optimal EWMA weighting scheme. The EWMA controller with on-line weight adaptation using an ANN approximation of the mapping from the disturbance state to optimal weight vector is presented in Section 8.3 and simulations to demonstrate its effectiveness are presented in Section 8.4. Conclusions are drawn and suggestions for future work on this type of adaptive controller are given in Section 8.5.

## 8.2 The Optimal EWMA Weight Vector as a Function of the Disturbance State

Section 4.5 demonstrated that there is an optimal EWMA weight for a closed-loop EWMA controller. As we just mentioned, it is not currently known how to theoretically calculate the optimal value of the MIMO EWMA controller weight vector, $w$, under closed-loop control. This is primarily because the minimum distance solution in the underdetermined case requires that the input vector be incorporated into the state vector, thereby removing simplifications that could be made in the SISO case (Chapter 6). Without these simplifications, the effects of different amounts of model mismatch, noise, targets, drifts, and shifts (expected number and size) cannot be removed from the final analytical form for the optimal weight vector. The combination of these different quantities will be referred to as the disturbance state. As shown in Section 4.5, given any known disturbance state, the optimal EWMA weight can be determined by simulating the process for several values of $w$ and extracting the optimal weight vector. We would like to generalize this procedure to generate the optimal EWMA weight vector as a function of the disturbance state. The first step in doing this is to determine how each state in the disturbance state vector affects the optimal EWMA weight vector. This analysis is basically a verification that the results we saw in the theoretical derivation for the SISO case (Chapter 6) carry over to the MIMO case.

In Section 4.5, we found that a drift disturbance causes the optimal value of the EWMA weight to increase and additive process noise causes the optimal EWMA weight to decrease. In the following simulations we focus on the CMP process. Since CMP is not prone to shifts (except at pad changes), the effects of shifts are not considered in our relevant disturbance state. This is primarily because a rapid mode controller, like that discussed in [Hu1], can be applied to monitor and compensate for shifts. As previously mentioned, model mismatch affects the optimal point. In light of this we now turn our attention to the condition of model mismatch.

### 8.2.1 Model Error Impact on Optimal EWMA Weights

The actual model mismatch may be different than what is originally estimated (more or less extreme in reality) and may affect the optimal value for the EWMA weighting parameter. Also, in real processes mismatch errors may tend to increase as time progresses. The effect of model error is two-fold; on one hand it causes an initial offset which must be compensated for by the EWMA controller and on the other hand it increases the noise in the controlled output.

We will begin our study of model error by considering the initial transient which it causes. This is best done by considering two simulations. The simulations were performed with a fixed drift (-20 Ang./wafer for RR and 1 Ang$^2$/wafer$^3$ for NU) and a fixed amount of additive noise (std. dev=30% target for RR and 50% of target for NU) for 40 lots (lots equal 10 wafers). The EWMA weight vector was also fixed at $\alpha=[0.1\ 0.1]^T$. The first simulation (Fig. 8-1) shows the controlled run with no model error and the second (Fig. 8-2) shows the controlled run for model error with a coefficient error of +/-70% for each coefficient. The run with zero model error is right on target from the outset of the run whereas the run with +/-70% model error has a huge offset at the beginning of the run. The EWMA controller must compensate for this offset in the following runs. The rate at which this is compensated for has a tremendous impact on the MSE, and thus on the optimal EWMA weight. Therefore, the size of the optimal EWMA weight must be increased to compensate for the initial offset caused by model error. To further validate this point, this experiment was repeated for several amounts of model error and the corresponding optimal EWMA weights are shown in Table 8-1. We see that the optimal weight strictly increases as the model error is increased in order to compensate for the initial offset.

**Figure 8-1. Controlled Run with No Model Error**

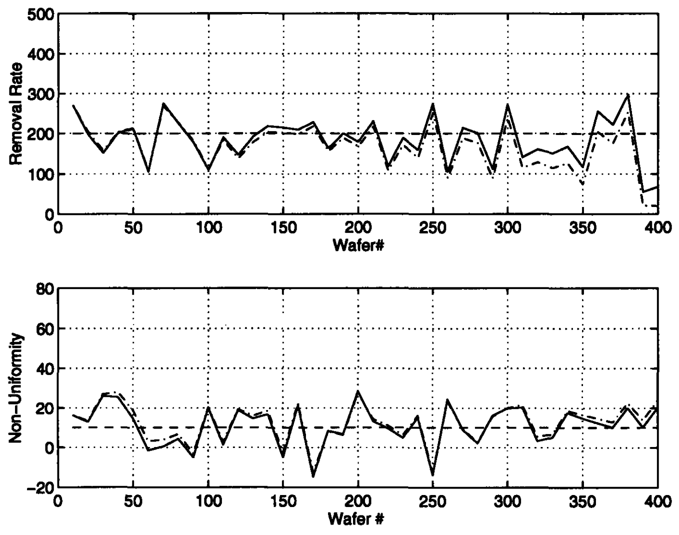


**Figure 8-2. Controlled Run with +/-70% Model Error**

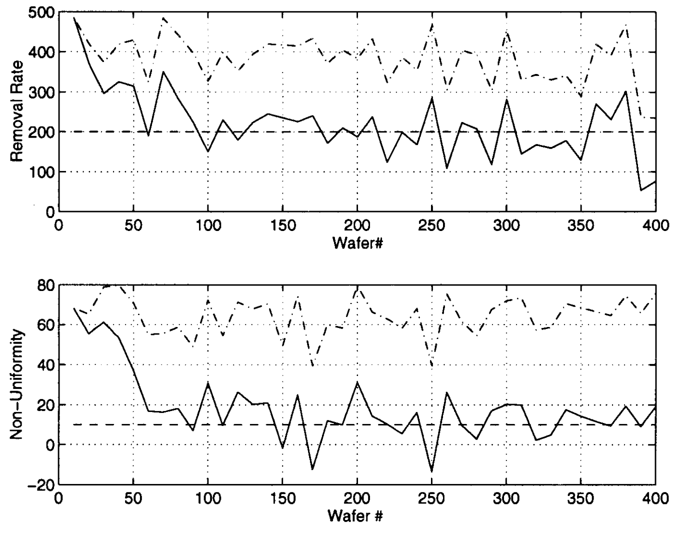


**Table 8-1. Optimal EWMA Weight vs. Different Model Error (Transient Effects)**

| | | Model Error | | | | |
|---|---|---|---|---|---|---|
| | | 0% | 30% | 50% | 70% | 90% |
| Optimal $\alpha$ | RR | 0.05 | 0.25 | 0.30 | 0.35 | 0.35 |
| | NU | 0.05 | 0.40 | 0.40 | 0.40 | 0.40 |

Although this is an interesting result it seems odd to place a large emphasis on initial model error. In an industrial setting, any significant offset would clearly be flagged as out of control and the model would be re-optimized before any long-term operation of the machine were to take place. Thus it seems as though we are more concerned about the situation where the process model error has increased over time, as the tool continues to operate without re-optimization and the EWMA controller has already gradually corrected for significant previous model errors. Therefore, it seems reasonable to rule out the effects of initial transients and determine any other effects of model error which occur after initial transients have died away.

The above simulations were repeated using 80 lot runs while measuring the MSE and output noise only in the second half of each simulated run. The drift rates for these simulations were increased (-200 Ang./wafer for RR and 10 Ang$^2$/wafer$^3$ for NU). From these simulations we see the second effect of model error. Specifically, model mismatch shows up in the output as increased or decreased noise (depending on whether the ratio of the coefficients is greater than or less than one) and drives down or up the optimal values of the EWMA weights. Table 8-2 shows the effect of model error on the optimal EWMA weight for each output. Again we see that the optimal weights are monotonically decreasing as the model coefficient error is increased (positively). Thus it appears as though model error has the same effect on the optimal EWMA weight as does additive noise. Therefore it would seem reasonable to combine the effects of model error and noise into one disturbance measure. In order to do so, we must first verify that increases in model error cause relative increases in the standard deviation of the noise measured in the controlled outputs. The standard deviation of the noise was measured in the controlled outputs as model error was increased (for a fixed $\alpha = [0.5\ 0.3]^T$). The results are shown in Table 8-3. Not surprisingly, we see that increases in model error correspondingly cause increases in the standard deviation of the noise measured at the output (additive noise was identical in all cases) just as do increases in additive noise. Therefore, when addressing the issue of the optimal EWMA weight we may simply consider model error and additive noise grouped together into one disturbance (noise measured at the output). These conclusions merely verify what was stated at the end of Chapter 6 for the SISO optimal weight determination.

**Table 8-2. Optimal EWMA Weight vs. Different Model Error (Steady State Effects)**

| | | Model Error | | | | |
|---|---|---|---|---|---|---|
| | | **0%** | **30%** | **50%** | **70%** | **90%** |
| Optimal $\alpha$ | RR | 0.55 | 0.45 | 0.40 | 0.33 | 0.30 |
| | NU | 0.30 | 0.25 | 0.20 | 0.18 | 0.15 |

**Table 8-3. Standard Deviation of the Noise vs. Model Error**

| | | Model Error | | | | |
|---|---|---|---|---|---|---|
| | | **0%** | **30%** | **50%** | **70%** | **90%** |
| Optimal $\alpha$ | RR | 784 | 828 | 862 | 903 | 953 |
| | NU | 101 | 106 | 109 | 111 | 114 |

## 8.2.2 Target Effect on Optimal EWMA Weights

The next disturbance we address is that of target variation. In order to understand the effect of target variation on the optimal EWMA weight vector, let us consider the example above and determine the optimal EWMA weights based on the entire run (including the initial transients). We will begin by considering the case of zero model error. Simulations were performed for several sets of targets and it was found that the optimal EWMA weights are unaffected by changes in target when there is no model error. When model error is present, however, the changes in target translate into the size of the initial offset through the controller's solution of the next recipe (using an incorrect model). Therefore, changes in target affect the size of the offset due to model error. The error caused by target changes is generally small but whether it increases or decreases the optimal weight depends on whether or not the change in target is further or closer to the point in *r*-dimensional space where the range of the process and control model intersect. The further away the target is from this intersection the larger the offset and hence the smaller the optimal weight will be. For the previous example, the intersection of these two models is the point $[0\ 0]^T$. Simulations were performed with a target of $[2000\ 100]^T$ and a target of $[8000\ 400]^T$. The first target has an optimal EWMA weight vector of $[\ 0.45\ \ 0.4\ ]^T$ and the second has the EWMA weight vector $[\ 0.55\ \ 0.5\ ]^T$. This is because the first target has a much smaller offset (since it is closer to the intersection point) and thus does not need as large a weight to obtain the minimum MSE. It can be shown that this relation holds independently for each output in the target vectors.

Therefore, the targets as well as the amount of model error determine the size of the offset at the beginning of a process run. As before we argue that this transient situation is not relevant to the determination of the optimal EWMA weight since any significant offset would cause process engineers to stop the process before a long run and re-optimize the process model to reduce error. A second situation which will cause these types of shifts will occur when the process target changes during a sequence of runs. Although this situation is certainly possible it is not likely for the CMP process because although different wafers are intended to be polished to different thicknessed, the polishing rate (which we are controlling here) is generally held constant.

Aside from the transient effect of target variation on the optimal weight vector, a question remains regarding long term effects caused by changes in targets. These questions can be answered by determining the optimal EWMA weights based on the MSE of the second half of long simulation runs (when the target and model error transients have died out). By performing several simulations for different targets we find the very interesting result that different targets have no long-term effect on the optimal EWMA weight vector. That is to say, different targets do not change the optimal EWMA weight. Once again, these results are an extension of the SISO case to multidimensional space.

### 8.2.3 Summary of Disturbance Effects on Optimal EWMA Weights

We now conclude our analysis of the effects of the different disturbances. First, we note that increases in the drift rate increase the optimal EWMA weight and increases in the additive noise decrease the optimal EWMA weight. These two observations are fundamental to the determination of the optimal EWMA weight vector. Model mismatch causes changes in the optimal EWMA weight in two ways. The first way is by an initial transient and the second is by changing the amount of noise in the output. Since model error tends to start small and increase gradually over time, we will neglect transient effects in our determination of the optimal EWMA weighting parameter. The second effect of model error, increased output noise, was shown to be similar to the effect of additive noise and thus model error and additive noise are combined to form one disturbance state (noise in the output). Finally, it was shown that changes in targets change the optimal EWMA weight only as a result of the transient created by model error when a change in target occurs. Since changes in target are expected only at the beginning of a run, when there is little error and there are no long term effects of target changes on the optimal EWMA weight, we can rule out specific targets as part of the disturbance state. Thus the affect of target shifts can be neglected in this study. In summary, the disturbance state for the CMP process can be reduced to a vector of two disturbances which affect the optimal EWMA weight vector; one containing noise terms (output noise) and the other containing drift information.

### 8.2.4 Optimal EWMA Weight Map

In order to create a general map from this reduced disturbance state to the optimal EWMA weight vector, we will simulate the process over a grid of possible disturbance state combinations and extract the optimal EWMA weight vector. Thus the simulations performed for a given process (as in Section 4.5) are merely repeated at every point on a grid in the disturbance state. The optimal value for $w$ was calculated here as a function of the output noise level and drift rate (the reduced disturbance state). In order to provide a fairly complete characterization of the optimal values as function of the disturbance state, the optimal values were determined over a large range of drift and output noise (significantly more than those that are realistically possible). This is a large range of input space and the simulation of this space is limited by computational expense. Therefore, a course grid of four hundred equally spaced points was simulated. In addition, for each of these points simulations for twenty EWMA weights were performed. The response surfaces of the optimal EWMA weight values for the removal rate and nonuniformty outputs, as a function of the process drift and output noise, are shown in Figs. 8-3 and 8-4, respectively. Notice that the main valley (low EWMA weights) is centered around the zero drift axis. Increasing drift magnitude (positive and negative) causes an increase in the optimal EWMA weight. In addition, increasing noise causes a corresponding decrease in the optimal EWMA weight. These results are similar to those obtained from the SISO case.

**Figure 8-3. Optimal EWMA Weight for Removal Rate as a Function of Output Noise and Drift**

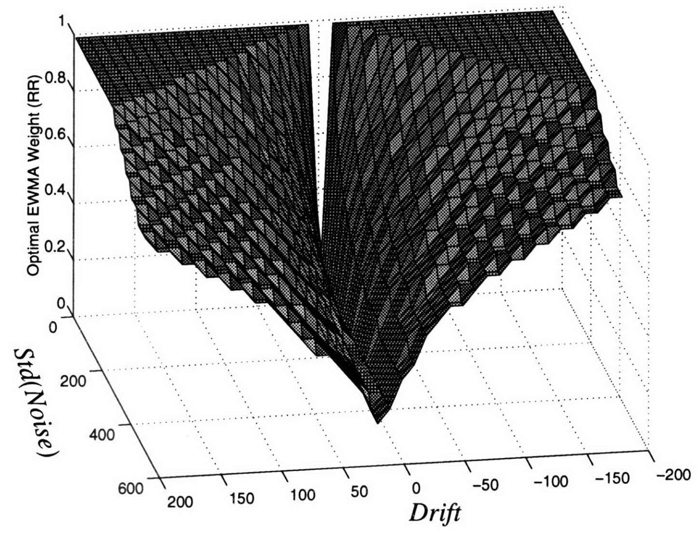
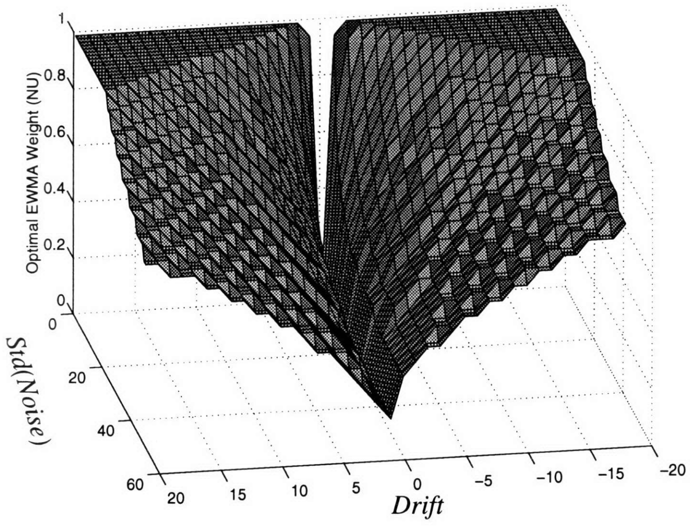


**Figure 8-4. Optimal EWMA Weight for Nonuniformity as a Function of Output Noise and Drift**

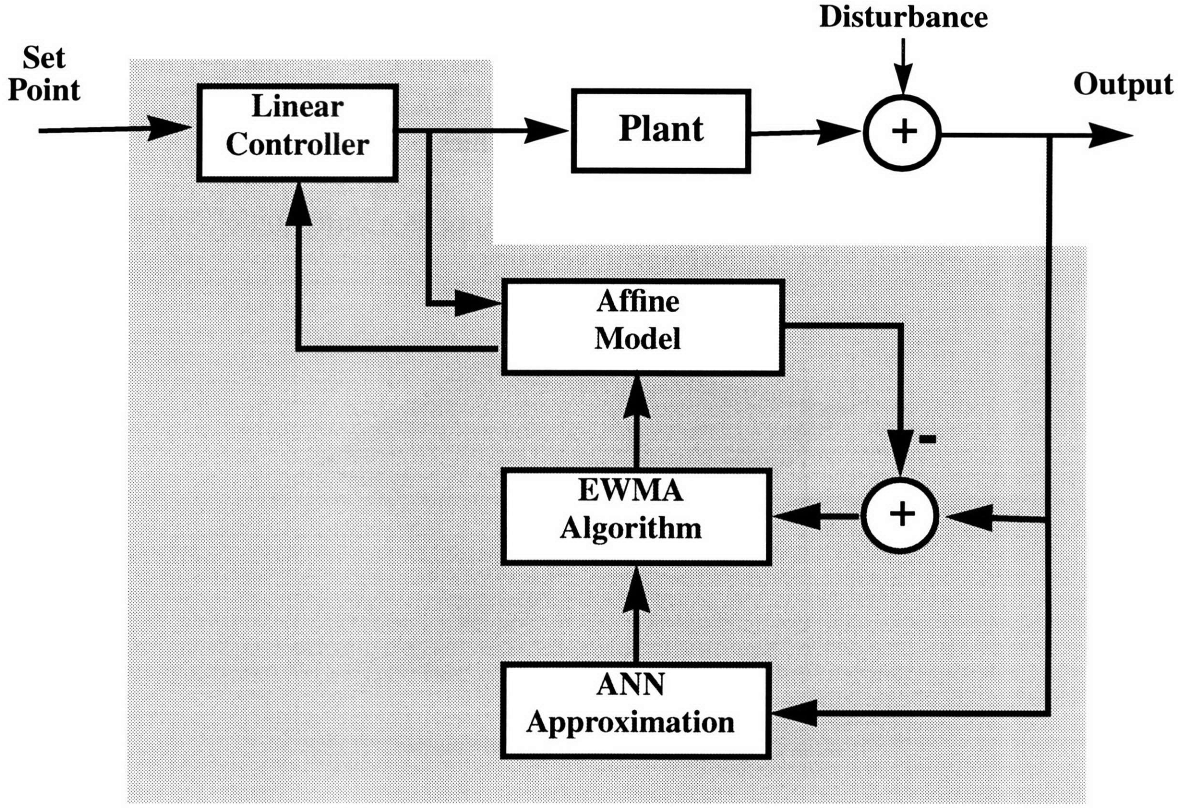


## 8.3 The EWMA Controller with an ANN Optimal Weight Estimator

Recently, adaptive updating of the EWMA weight for use in open-loop tracking of a process using adaptive Kalman filtering has been shown in [Hembree]. Our goal with the following framework is to provide a similar ability for closed-loop control. The strategy for the artificial neural network (ANN) EWMA weight estimator is to utilize an ANN approximation of the mapping from the disturbance state to the optimal EWMA weight vector to dynamically update the EWMA controller during the control run. This idea is outlined in Fig. 8-5.

A neural network was trained to learn the mapping from the disturbance state to the corresponding optimal value of $w$ using the course grid of empirically determined optimal EWMA weights generated in Section 8.2. The disturbance state is estimated on-line and fed into the neural network. The ANN provides an excellent approximation tool in this case. As the number of outputs increases, the number of points in this disturbance state increases at an incredible rate; the ANN can often learn the structure with only a small sample of the points over the grid. Once trained, the ANN provides the optimal value of $w$ given the estimated disturbance state. Updates to the EWMA weighting parameters can be made continuously or periodically.

Figure 8-5. The EWMA Controller with the ANN EWMA Weight Estimator



## 8.3.1 Training the Neural Network

An multi-layer perceptron (MLP) ANN was trained using the Levenburg-Marquardt training algorithm to learn the mapping from the disturbance state to the optimal EWMA weight. The neural net was trained with an MSE < 0.02 for each optimal weight in the training set. Several issues arose in the training of the neural net. The first was the fact that ANN's have the tendency to overtrain. In the case of EWMA weight estimation, this is a serious problem because poor estimates due to overtraining could cause fluctuations in $w$, unnecessary additive noise, improper tracking of process drift or unstable control action. Therefore, the response surface (including the points in between training points) of the ANN was plotted while the net was training. This allowed the ANN to be supervised while it learned the mapping so as to avoid overtraining. It also allowed us to insure that the surface was smooth so that noise was not added to the dynamic estimation of $w$ by the ANN. The response of the network outside the trained region could also be observed. This is very important in properly determining the value of $w$, since it is required to be within zero and one. Originally, the neural network was providing extremely high values of $w$ outside its trained region. This problem was solved by training the ANN using data over a significant region of noise and drift values where these limits were reached. This allowed the ANN to learn the leveling off at the top and bottom. This is crucial for proper performance of the system if the drift or noise fall outside the trained region. For actual implementation in the architecture shown in Fig. 8-5, bounds were added to the output of the ANN to ensure the zero and one limits (although these were found to be unnecessary). These may be lowered if it is felt that the system is prone to instability.

The response surfaces for the network outputs are shown in Figs. 8-6 and 8-7 for removal rate and nonuniformty values, respectively. The neural network has effectively smoothed out the noisy sections caused by running the simulations at discrete values of drift, noise, and $w$.

**Figure 8-6. Optimal EWMA Weight for Removal Rate as a Function of Noise and Drift (Neural Network)**

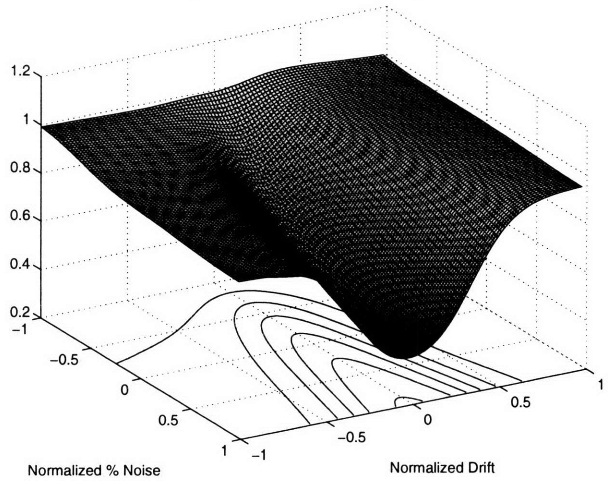
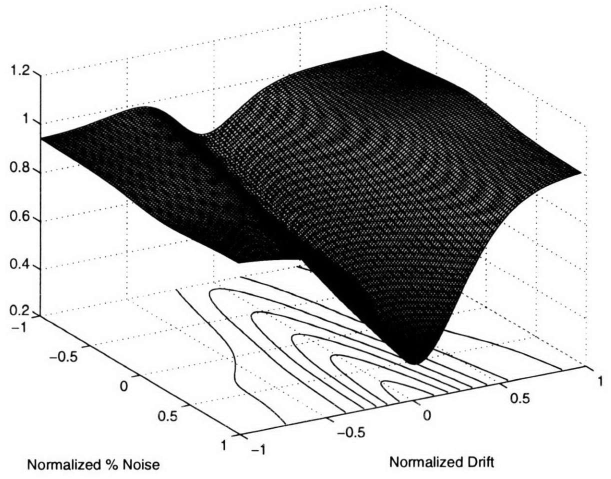


**Figure 8-7. Optimal EWMA Weight for Nonuniformity as a Function of Noise and Drift (Neural Network)**



### 8.3.2 Estimating the Disturbance State

The ability to properly estimate the process noise and the underlying drift size is imperative to the successful implementation of the ANN EWMA Weight Estimator. Measuring the amount of process noise is fairly simple as long as the process remains under control. Assuming no shifts or drifting in the output, the process noise is measured using a sample standard deviation. Assuming no shifts is reasonable since any shifts in the process are intended to be corrected by a rapid mode controller, not the EWMA controller. In addition, the CMP process is not prone to shifts during the use of a single pad, which is where the EWMA controller is intended to be used.

There are several possibilities for estimating drift. One could simply determine the change in output from one run to the next. Instead, one could calculate the change in the current output by averaging the change over several runs. Unfortunately, estimating the drift is not this easy because noise in the current output corrupts the estimate of drift calculated at that time. Noise in the estimate of drift is a problem because if the size of the drift is small (relative to the noise) and is mistaken to be large (due to noise), the neural net would provide a large value of the EWMA weight to be used. But when there is a large amount of noise, a low value of $w$ is desired since it decreases the noise due to overcontrol. Another problem with a large amount of noise in the estimate of drift is that this could cause a tremendous fluctuation in the dynamic value of $w$ and possibly cause the system to go unstable.

To solve this problem we use a separate EWMA of the change in the process with a very small weight. Once again, these weights should be determined based on the amount of noise in the process. In this case we simply choose them based on the amount of noise measured in the baseline process. These are chosen to be very small since we are looking for small changes in the disturbance state and because we desire a measure of drift which is insensitive to noise. With this we make the assumption that the drift will change relatively slowly during the process, since an EWMA with a small $w$ would be unable to track a rapidly changing drift.

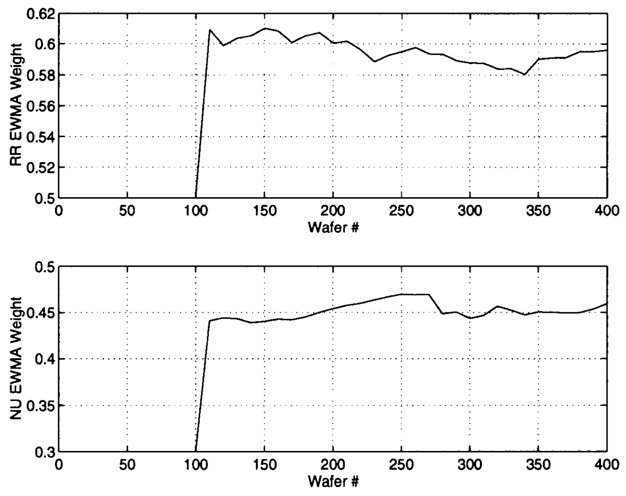
## 8.4 Simulations of the EWMA Controller with ANN Weight Estimator

The following simulations were conducted with two assumptions. The first assumption made is that no shifts will occur during the process and the second is that the drift will not change rapidly. The initial values for the EWMA weights were optimized for the baseline process shown in Fig. 2-3. The strategy given here is to show that the EWMA weights optimized for the baseline process will adapt with a varying amount of noise and underlying drift.

The simulations are compared based on their MSE. These values are summarized in Table 8-4 (at the end of the section) for all the simulations performed.

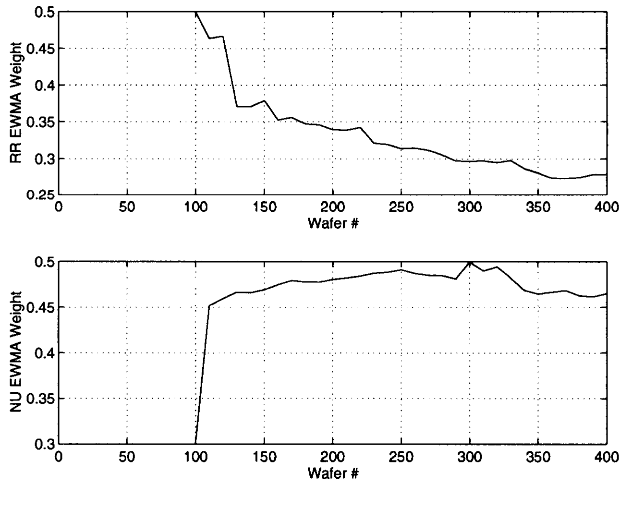
The control of the baseline process with and without adaptation of the EWMA weights result in nearly identical performance. There is a minimal difference in the MSE. This is expected since both have optimized EWMA weights for this process. Therefore, the dynamic system performs little adaptation on every run as shown in Fig. 8-8.

**Figure 8-8. Dynamic *w* Values for the Baseline Process**



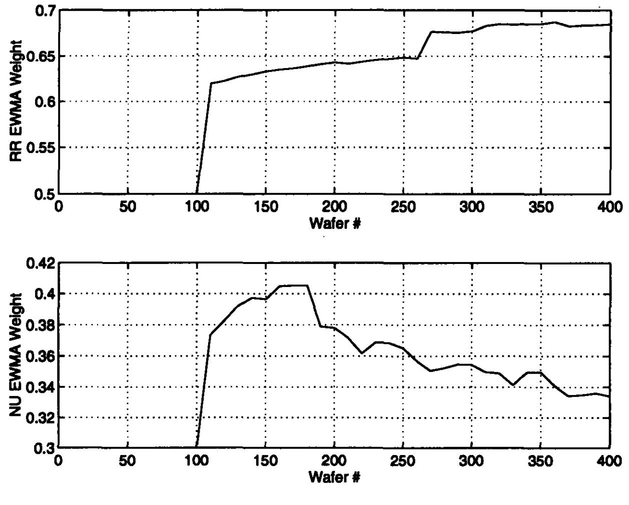
The improved performance of the ANN EWMA Weight Estimator system is demonstrated by the 9% decrease in MSE in the next example, where the process has been altered to consist of a much smaller drift and increased noise in the removal rate output. The dynamic values of the EWMA Weights are shown in Fig. 8-9. Notice how the removal rate weight decreases from its predetermined value as soon as it is determined that there is no underlying drift. The decrease in the EWMA weight for removal rate decreases the noise added to the controlled response by the controller.

**Figure 8-9. Dynamic *w* Values for the Small Drift/High Noise Removal Rate Process.**



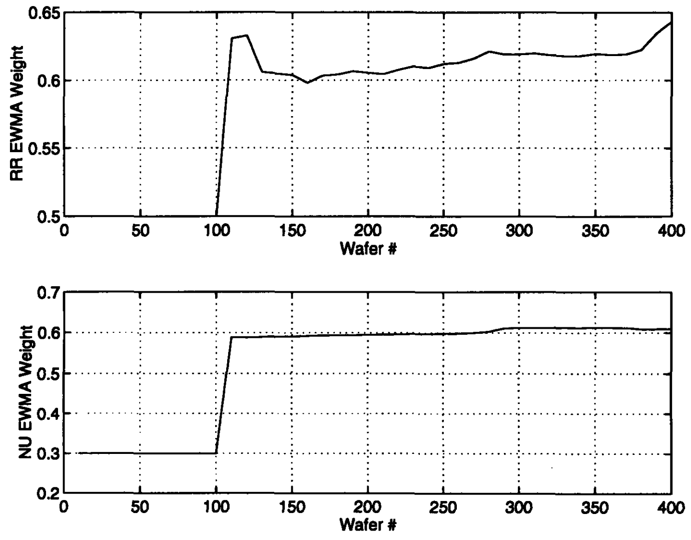
The next simulation shows the control of a process with a large drift and low noise in removal rate. The dynamic weights are provided in Fig. 8-10. In this case, the EWMA weight for removal rate increases to compensate for the large drift. The MSE for the adaptive controller is 30.7% better than that of the fixed EWMA weight case.

**Figure 8-10. Dynamic *w* Values for the High Drift/Low Noise Removal Rate Process.**



As a final demonstration we show an equivalent experiment with the nonuniformty (which has remained relatively steady for the previous experiments) by decreasing the noise from its usual high value and increasing the drift. The dynamic weights, provided in Fig. 8-11, show how the EWMA weight for nonuniformty nearly doubles. In addition, the MSE is improved by 38.7%.

**Figure 8-11. Dynamic *w* Values for the High Drift/Low Noise Nonuniformity Process.**

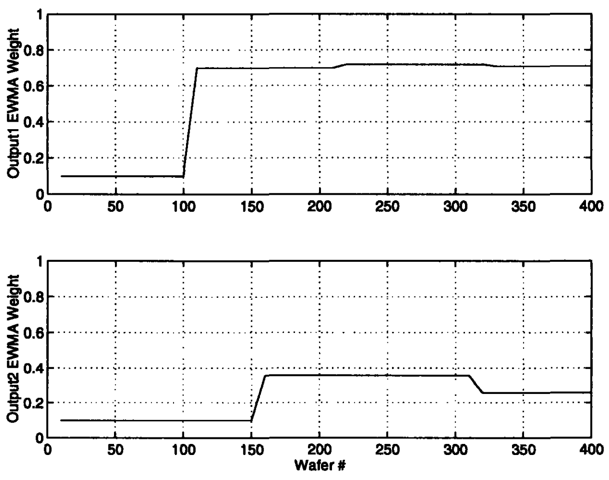


A variation to the method outlined above allows for periodic updating of the weights. This method has the advantage of reducing the amount of noise in the estimates of the process drift and standard deviation. The disadvantage, however, is that the update to the controller does not occur as often. Therefore, when a change in the disturbance state occurs just after an update has been made the controllers performance is weakened in that it must respond to the disturbance with a possibly lower EWMA weight. One should therefore estimate the likelihood of such changes and pick an optimal update rate. It is unlikely that this will make a large difference unless the update rate is very infrequent. An example of the periodic case is shown below. Here, both weights were started at low values (not their optimum). The removal rate output has a large drift relative to the amount of noise and the nonuniformty output has a small amount of drift relative to the noise. As expected, the removal rate weight moves to a high value and the nonuniformty weight moves to a low value.

**Table 8-4. Mean Squared Error of Fixed versus Dynamic EWMA Weights (Lower Numbers Are Better)**

| | Removal Rate | | Nonuniformity | |
|---|---|---|---|---|
| | Fixed α | Dynamic α | Fixed α | Dynamic α |
| Baseline | 135.5 | 132.8 | 43.9 | 49.9 |
| Low Drift High Noise in RR | 3,142 | 2,856 | 37.0 | 39.4 |
| High Drift Low Noise in RR | 266.6 | 184.7 | 99.4 | 103.3 |
| High Drift Low Noise in NU | 106.4 | 98.49 | 2.56 | 1.57 |

**Figure 8-12. High Drift RR Low Drift NU (Periodic Update)**



## 8.5 Summary: A Direct Adaptive EWMA Controller

We have demonstrated that a self-tuning EWMA controller using a neural network approximation of the emperically determined optimal EWMA weights provides the ability to dynamically update the EWMA weights. This removes the need for operators to tune EWMA weights, simplifies the implementation process, and improves performance.

# Chapter 9

# An Indirect Adaptive Linear ANN Controller

Adaptive linear control systems have been extensively developed [Narendra1, Åström1]. We discuss these systems here for several reasons. First, these systems are applicable to CMP process control because, much like EWMA based controllers, their adaptive nature allows them to compensate well for random changes in the process characteristics. These systems make an extension to the EWMA type controllers in that all the coefficients (not just the offset) are adapted as the process runs. This is important when we have a large amount of model mismatch in the linear coefficients. We will use the terminology "linear neural network" to refer to these systems in the view that they are "linear approximations" much as multi-layer perceptron neural networks are full nonlinear approximations. In light of this, these linear adaptive systems provide a natural transition to fully adaptive nonlinear systems (the next step up from these controllers). Linear neural networks of this type have shown great promise in solving complex problems [Nguyen, Widrow].

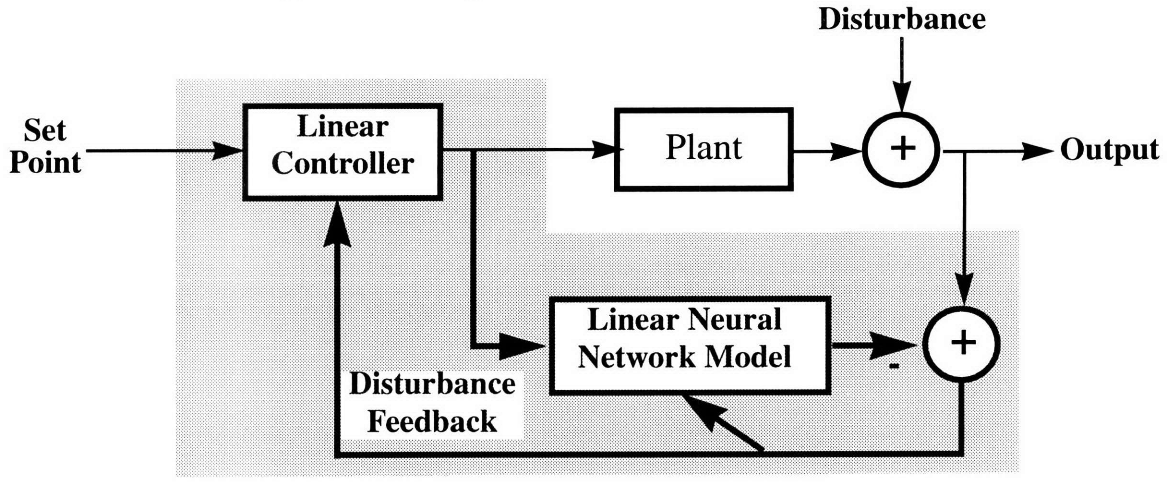
The adaptive linear controller utilizes the architecture shown in Fig. 9-1. The linear neural network process model is shown Fig. 9-2. As can be seen, the network consists of a single layer of linear neurons. Therefore, the process model output at discrete-time *n* is determined as follows:

$$y_n = W_n u_n , \qquad (9\text{-}1)$$

where $W_n$ is an *M* by *R* matrix of linear coefficients, $y_n$ is an *M* by 1 vector of outputs, and $u_n$ is an *R* by 1 vector of inputs (including a constant 1, as one of the elements).

At each step, the process recipe is determined by using this linear model to generate a minimum distance solution which drives the outputs to their target values. The process is run and the actual process output is measured and the output errors are calculated. This error is then used to update the network weights (linear coefficients) and the procedure repeats.

**Figure 9-1. Adaptive Linear Network Controller**



The weights are adapted using the Widrow-Hoff rule (gradient descent for a linear network) [Widrow]. The idea is to take a small step in the direction which reduces the sum squared error (SSE). Therefore, the direction should be a descent direction, and thus the direction of the negative gradient (of the SSE) gives a direction which always reduces the SSE. Therefore, we have the gradient:

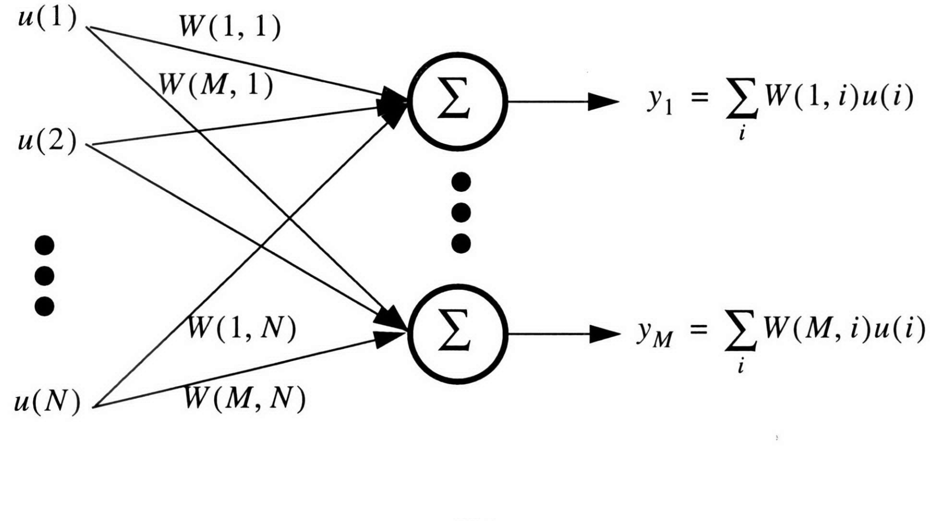
$$\frac{\partial}{\partial \boldsymbol{W}(i,j)} SSE = \frac{\partial}{\partial \boldsymbol{W}(i,j)} \frac{1}{2}\left(\boldsymbol{t}(i) - \sum_{j=1}^{R} \boldsymbol{W}(i,j)\boldsymbol{u}(j)\right)^2 = -\boldsymbol{e}(i)\boldsymbol{u}(j), \tag{9-2}$$

and the learning rule:

$$\Delta \boldsymbol{W}(i,j) = \boldsymbol{W}(i,j) + lr\,\boldsymbol{e}(i)\boldsymbol{u}(j) . \tag{9-3}$$

where *lr* is the step size or learning rate. The size of the step determines the degree of filtering performed at each measurement. This step size is analogous to the EWMA weight, where in both cases the smaller the step size or EWMA weight, the more the smoothing.

**Figure 9-2. The Linear Neural Network Model**



Simulations have been performed which demonstrate that, with the proper selection of the learning rate, the linear network can provide control nearly identical to that obtained with the EWMA controller when the model errors are small. In contrast to the EWMA controller, the adaptive linear network provides excellent stable control of approximately linear processes when initial model errors are large enough to make the EWMA controller go unstable. This effect is shown in Figs. 9-3 and 9-4, where the target values for the two outputs were 1800 and 200 respectively. Normally distributed noise and drifts of -20 and 4 were added to the outputs.

As can be seen by the controller's ability to account for drift in the previous examples, these systems have the ability to compensate for unmodeled system dynamics of low order as well as model error. Linear network models are useful in that the updates to the process model and control actions can be made extremely fast. The disadvantage to a controller of this type is that it is unable to control highly nonlinear systems or to adequately adapt to unmodeled high order system dynamics. We will not offer a complete analysis of these controllers since a large amount literature exists which offers a full treatment of this material [Narendra1, Åström1].

**Figure 9-3. EWMA Controller with Unstable Response]**

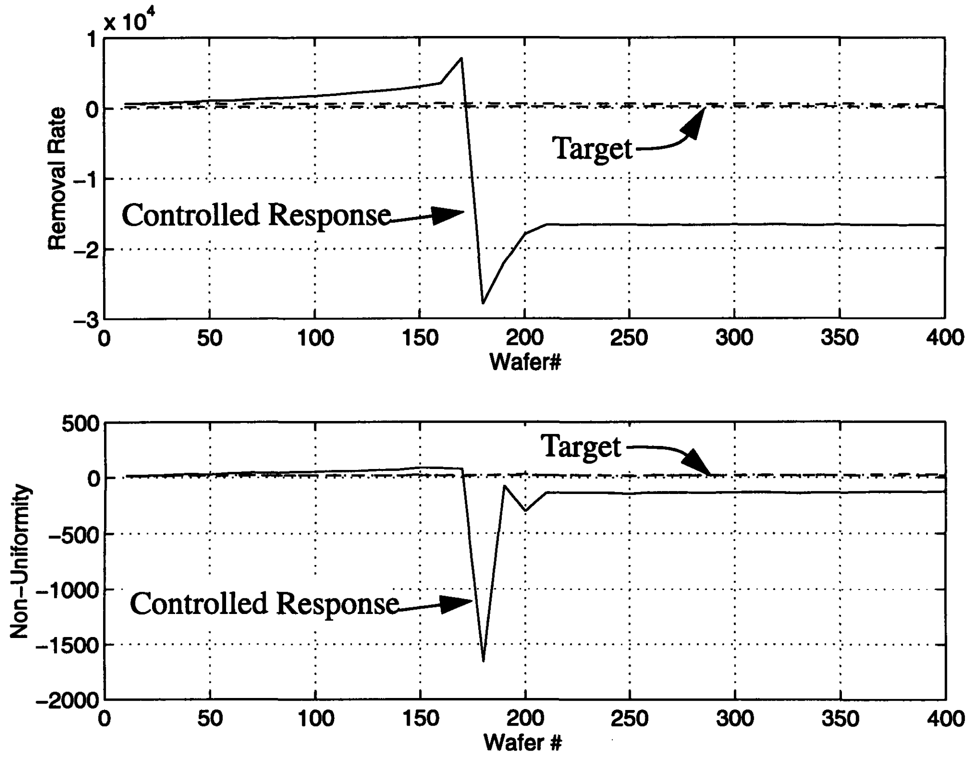
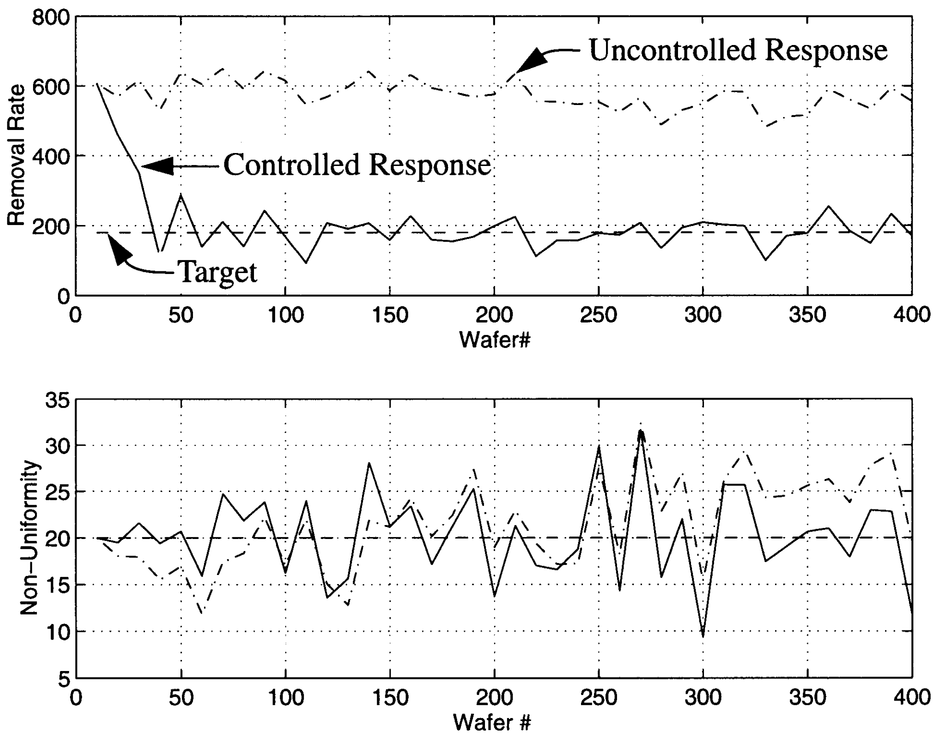


**Figure 9-4. Adaptive Linear ANN with Stable Response**

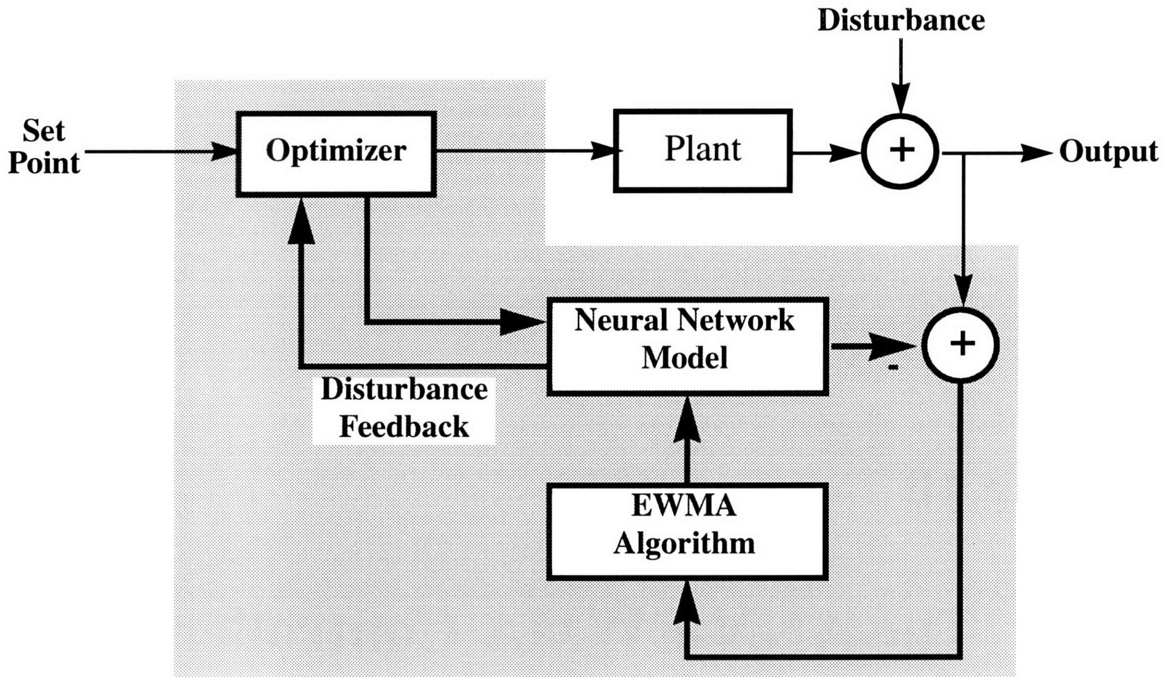


# Chapter 10

# The ANN-EWMA Controller

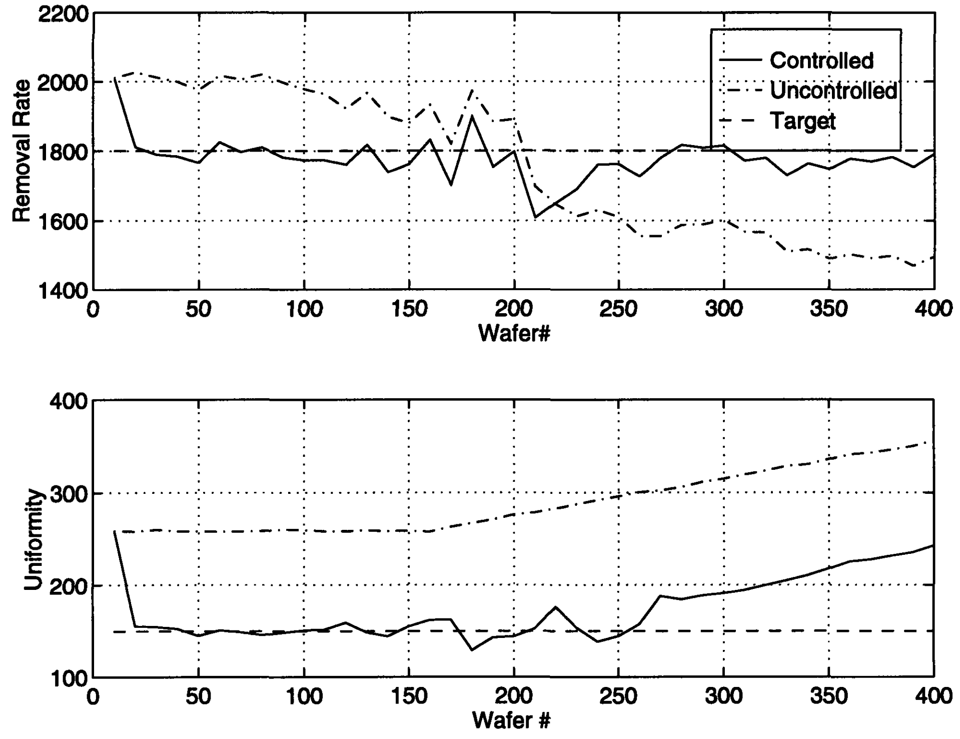
We have seen throughout Part II that the EWMA controller provides the ability to adequately control approximately linear systems with small to large amounts of noise [Boning1, Boning2]. Although many processes are in fact linear, this is not always the case and we would like to use a similar architecture for nonlinear systems. A first attempt at doing so is to use a neural network model, thus incorporating the necessary nonlinear mapping, and update the biases of the output layer to allow filtered changes to be made to the nonlinear mapping. In moving to a nonlinear process model, we lose the luxury of utilizing the efficient linear solver. Therefore, this is replaced with an optimization routine which determines the inputs to the plant using a utility function. In this case the utility is the minimization of control variation such that the inputs are bounded and the outputs are equal to target. If no solution can be found, variation in the inputs is sacrificed for minimizing the output error, while still requiring the inputs to be bounded. This method is an example of backpropagation of utility and is demonstrated in Fig. 10-1 [Werbos1, Werbos2].

**Figure 10-1. The ANN-EWMA Controller**



In the example provided below, the ANN process model was initially trained using random sample points from a second order polynomial. Updates to the ANN process model are made using an EWMA on the output layer bias terms. The optimization routine used was a Gauss-Newton gradient descent algorithm. The plant was a second-order polynomial model with 5% gaussian noise added to each coefficient. Although not shown here, the EWMA controller provides an unstable response due to the nonlinearity. The response of the ANN-EWMA controller to a process with a shift and drift in each output is shown in Fig. 10-2, with appropriate control in both cases, until the recipe limits are reached. At this point the nonuniformity begins to increase at a rate equal to the drift in the baseline process. These results demonstrate the controller's ability to provide stable control of the nonlinear system with process dirfts and shifts.

Figure 10-2. The ANN-EWMA Controller with Shift and Drift



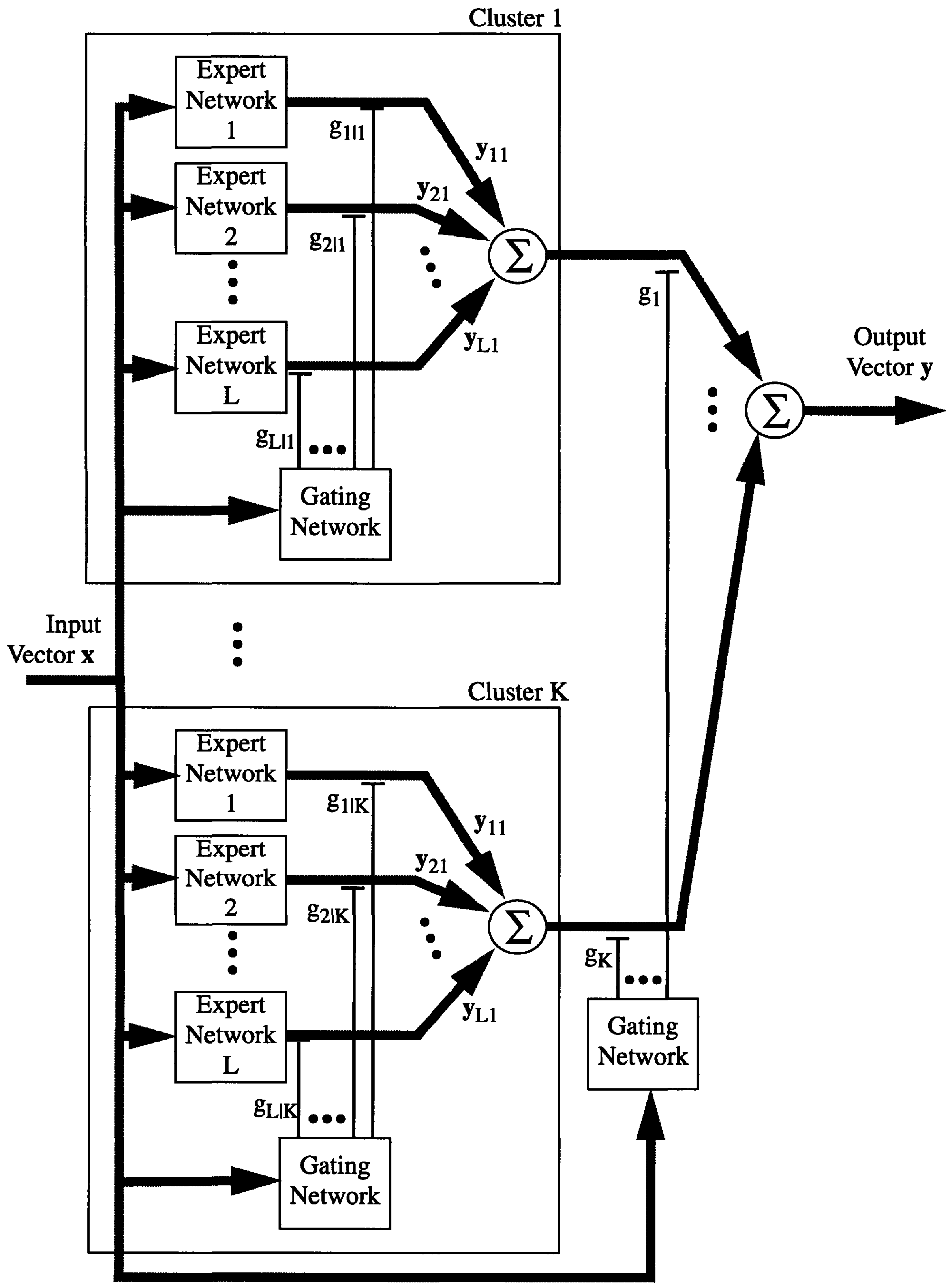
# Chapter 11

# An Adaptive HME ANN Controller

The next extension to this type of architecture is to update the entire neural network as opposed to simply adapting the bias layer of the neural network. This is an important difference because updating the bias terms corresponds to merely shifting the nonlinear map in hyperspace whereas full adaptation of all the weights corresponds to changing the shape of the nonlinear surface in hyperspace. A natural approach to this problem is to use an MLP ANN and update all the weights using a gradient descent type algorithm. However, this causes the global mapping to be changed in regions where we are not training. This is a problem in adaptive control because we typically wish to operate our process at one well-defined location in output space, and a corresponding limited region of operation in input space. In adapting an MLP ANN we would thus train (adapt) the network with only local data. However, when using a gradient descent type method on an MLP ANN, we adapt all the weights and thereby affect the global functional approximation. As we train in one locale, we improve the approximation in that region but at the same time degrade the functional map in the region outside the training region. Several solutions may be proposed to alleviate this problem, such as forcing the original data to be added to the adaptation set. These solutions often improve the results but adapt slowly (due to the increased data set) and hence defeat the purpose of adaptation. In light of this, we will not pursue such an architecture here. Instead, we suggest the use of an architecture proposed by [Jacobs].

The Hierarchical Mixture of Experts (HME) ANN architecture (shown in Fig 11-1) is based on the principle of modularity. Unlike the MLP ANN, the HME ANN is broken up into a hierarchical tree. This architecture uses a competitive scheme whereby branches of the hierarchical tree compete to respond to the input. This allows the network to break up the regions of input space and assign expert(s) to that region. The output of each expert is then weighted by the likelihood that that expert is correct. If the function which is being approximated is naturally decomposable into simpler functions, then the architecture of the HME network has the built-in ability to discover this decomposition [Haykin]. This aids the network in properly adjusting a fewer number of connections to learn patterns localized to specific regions in input space. This is the characteristic we will exploit here. In adaptation strategies, the ability of the HME network to separate the input space allows adaptation to specific parts of the network to take place without destroying the global functional mapping. The HME network has limitations; if a high learning rate is used then other experts will tend to be reclaimed for use in the current region. In addition, the local adaptation of this network keeps it from learning global trends, since only certain experts are updated in certain regions. We now turn to highlighting the structure and training procedure for this type of network.

**Figure 11-1. The Hierarchical Mixture of Experts Network**



The output $y_{ij}$ of the i^th^ expert in the j^th^ cluster is:

$$y_{ij}(m) = \sum_{n=1}^{p} W_{ij}(m,n)x(n), \tag{11-1}$$

where $W_{ij}$ is the weight matrix for that expert, $x$ is the input vector, and $p$ is the length of $x$. This output is assumed to be the conditional mean of a multivariate gaussian distribution (given that this expert is chosen to respond to the input, $x$). These outputs then compete probabilistically to respond to the output. The output of the j^th^ cluster is the sum of the outputs of the experts weighted by the likelihood that they are the correct output, $d$, given that the j^th^ cluster is chosen to respond to the input:

$$y_j = f(d|x,j) = \sum_{i=1}^{L} g_{i|j}y_{i|j}. \tag{11-2}$$

The final output is then a probabilistic combination of the cluster outputs:

$$y = f(d|x) = \sum_{j=1}^{K} g_j y_j. \tag{11-3}$$

The next issue is the determination of the gating weights. Let $u_{i|j}$ be a linear combination of the inputs for the j^th^ cluster:

$$u_{i|j} = a_{i|j}^{T}x, \tag{11-4}$$

where $a_{i|j}$ is a weighting vector for the i^th^ output of the gating network for the j^th^ cluster. The gating weight for the i^th^ expert in the j^th^ cluster are determined using the Softmax function:

$$g_{i|j} = \frac{\exp(u_{i|j})}{\sum_{k=1}^{L} \exp(u_{k|j})}, \tag{11-5}$$

and the gating weight for the j^th^ cluster is determined using another Softmax function:

$$g_j = \frac{\exp(u_i)}{\sum_{m=1}^{K} \exp(u_m)} \quad \text{, where } u_i = \boldsymbol{a}_i^T \boldsymbol{x}, \tag{11-6}$$

with $u_i$ and $\boldsymbol{a}_i$ similarly defined for the second layer. The Softmax function provides two important features. First, it provides a smooth transition between branches of the tree. Specifically, at any point in input space, the output is a weighted combination of the different branches of the tree, where in certain regions certain experts dominate. However, when near transition regions, more than one expert might have significant effect on the output. The second feature is that the Softmax function provides a way to maintain the gating weights as probabilities. Specifically, all the gating weights sum to one and therefore can be treated as a probability mass function.

In order to train this network to learn a mapping of input output-pairs, we need to answer the question "to what distributions do the conditional means belong?" We do this by considering the likelihood that the output of the i$^{th}$ expert in the j$^{th}$ cluster is correct, given that this expert is chosen to respond to the input as a multi dimensional gaussian:

$$f(\boldsymbol{d}|\boldsymbol{x}, i, j) = \frac{1}{(2\pi)^{q/2}} \exp\left(-\frac{1}{2}\|\boldsymbol{d} - \boldsymbol{y}_{ij}\|^2\right), \tag{11-7}$$

where $q$ is the length of the desired output vector $\boldsymbol{d}$.

Using this and the expressions above we see that the likelihood that the output is correct is:

$$f(\boldsymbol{d}|\boldsymbol{x}) = \frac{1}{(2\pi)^{q/2}} \sum_{j=1}^{K} g_j \sum_{i=1}^{L} g_{i|j} \exp\left(-\frac{1}{2}\|\boldsymbol{d} - \boldsymbol{y}_{ij}\|^2\right). \tag{11-8}$$

In order to simplify the calculations, we will use the log-likelihood:

$$l = ln\left(\frac{1}{(2\pi)^{q/2}} \sum_{j=1}^{K} g_j \sum_{i=1}^{L} g_{i|j} \exp\left(-\frac{1}{2}\|\boldsymbol{d} - \boldsymbol{y}_{ij}\|^2\right)\right). \tag{11-9}$$

In order to train the network, we now simply maximize this likelihood function with respect to all the weights. We begin by defining the *a posterior* probabilities:

$$h_j = \frac{g_j \sum_{i=1}^{L} g_{i|j} \exp\left(-\frac{1}{2}\|\boldsymbol{d} - \boldsymbol{y}_{ij}\|^2\right)}{\sum_{j=1}^{K} g_j \sum_{i=1}^{L} g_{i|j} \exp\left(-\frac{1}{2}\|\boldsymbol{d} - \boldsymbol{y}_{ij}\|^2\right)} \text{, and} \tag{11-10}$$

$$h_{i|j} = \frac{g_{i|j}\exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)}{\sum_{i=1}^{L} g_{i|j}\exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)}, \tag{11-11}$$

where $h_j$ is the *a posteriori* probability that the j$^{th}$ cluster generates the desired response and $h_{i|j}$ is the *a posteriori* probability that the i$^{th}$ expert of the j$^{th}$ cluster generates the desired response given that the j$^{th}$ cluster contains the expert which generates the desired response.

We begin by rewriting the log-likelihood as:

$$l = -(q/2)ln(2\pi) + ln\left(\sum_{j=1}^{K} g_j \sum_{i=1}^{L} g_{i|j}\exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)\right), \tag{11-12}$$

and realize that in maximizing this with respect to $g_j$ the first term does not affect the result. Because the denominator in the expression for $g_j$ is the same for every j, we can pull it out of the summation. Therefore, we can maximize the following log-likelihood:

$$l = ln\left(\sum_{j=1}^{K} \exp(u_j) \sum_{i=1}^{L} g_{i|j}\exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)\right) - \ln \sum_{j=1}^{K} \exp(u_j). \tag{11-13}$$

Taking the derivative of this with respect to the gating weights $u_j$:

$$\frac{\partial l}{\partial u_j} = \frac{\exp(u_j)\sum_{i=1}^{L} g_{i|j}\exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)}{\sum_{j=1}^{K} \exp(u_j)\sum_{i=1}^{L} g_{i|j}\exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)} - \frac{\exp(u_j)}{\sum_{j=1}^{K} \exp(u_j)}, \tag{11-14}$$

which when the top and bottom of the first term are divided by $\sum_{j=1}^{K} \exp(u_j)$, gives:

$$\frac{\partial l}{\partial u_j} = h_j - g_j. \tag{11-15}$$

Similarly, we use the modified log-likelihood:

$$l = ln\left(\sum_{j=1}^{K} g_j \sum_{i=1}^{L} g_{i|j} \exp\left(-\frac{1}{2}\|d - y_{ij}\|^2\right)\right), \tag{11-16}$$

to take the derivative with respect to the gating weights $u_{i|j}$ to obtain:

$$\frac{\partial l}{\partial u_{i|j}} = h_j(h_{i|j} - g_{i|j}). \tag{11-17}$$

We then take the derivative of the same log-likelihood with respect to the output $y_{ij}$, to obtain:

$$\frac{\partial l}{\partial y_{ij}} = h_j h_{i|j}(d - y_{ij}). \tag{11-18}$$

Now we note that

$$\frac{\partial l}{\partial a_j} = \frac{\partial l}{\partial u_j}\frac{\partial u_j}{\partial a_j}, \frac{\partial l}{\partial a_j} = \frac{\partial l}{\partial u_{i|j}}\frac{\partial u_{i|j}}{\partial a_{i|j}}, \text{ and } \frac{\partial l}{\partial w_{ij}} = \frac{\partial l}{\partial y_{ij}}\frac{\partial y_{ij}}{\partial w_{ij}}, \tag{11-19}$$

in order to determine that:

$$\frac{\partial l}{\partial a_j} = (h_j - g_j)x, \frac{\partial l}{\partial a_{i|j}} = h_j(h_{i|j} - g_{i|j})x, \text{ and } \frac{\partial l}{\partial w_{ij}} = h_j h_{i|j}(d - y_{ij})x. \tag{11-20}$$

Finally, we perform gradient *ascent* on these weights in order to maximize the probability that the network provides the desired output:

$$a_j(n) = a_j(n-1) + \gamma\frac{\partial l}{\partial a_j}(n), \tag{11-21}$$

$$a_{i|j}(n) = a_{i|j}(n-1) + \gamma\frac{\partial l}{\partial a_{i|j}}(n), \tag{11-22}$$

$$w_{ij}(n) = w_{ij}(n-1) + \gamma\frac{\partial l}{\partial w_{ij}}(n) \tag{11-23}$$

The learning rate $\gamma$ may be the same or different in each case. An interesting implication is that if we wish to adapt the network on-line, we can independently choose to update the experts in the network and not the structure of which experts respond to which inputs. This is an important property, because adaptation of the experts is proportional to the probability that that expert will respond to the output. Therefore, in certain regions training on certain experts will not occur because they are not likely to provide the correct output, and we do not update the probabilities of the experts being correct. Thus we will not corrupt the mapping of the response in other regions by repetitively training in one region.

We now turn to an example of full nonlinear adaptive control using the structure given in Fig. 11-2. In this case, the HME ANN was trained with a second order polynomial. The plant is a similar second order polynomial with added model error (*N*(0,10%) on each coefficient), a drift of -20 on removal rate and +1 on nonuniformty, added white noise (*N*(0,2% of target) on removal rate and *N*(0,10% of target) on nonuniformty) and shifts of -300 on removal rate and +80 on nonuniformty. On each run, the controller model was updated using gradient descent on a moving window of the past data. The inputs were generated by using a gauss-newton gradient descent optimization on a cost function. The cost function used was the squared error from target in removal rate while keeping the nonuniformty below its upper bound as long as possible. The results are shown in Fig. 11-3, where we can see the tremendous improvement in the control of the removal rate output while barely changing the nonuniformty from the baseline run.

**Figure 11-2. HME Controller**

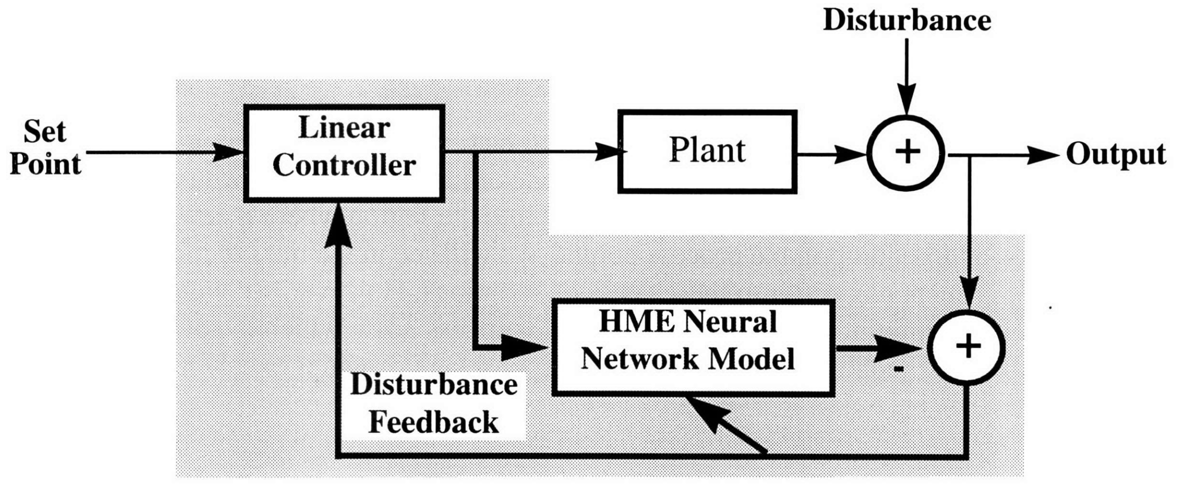
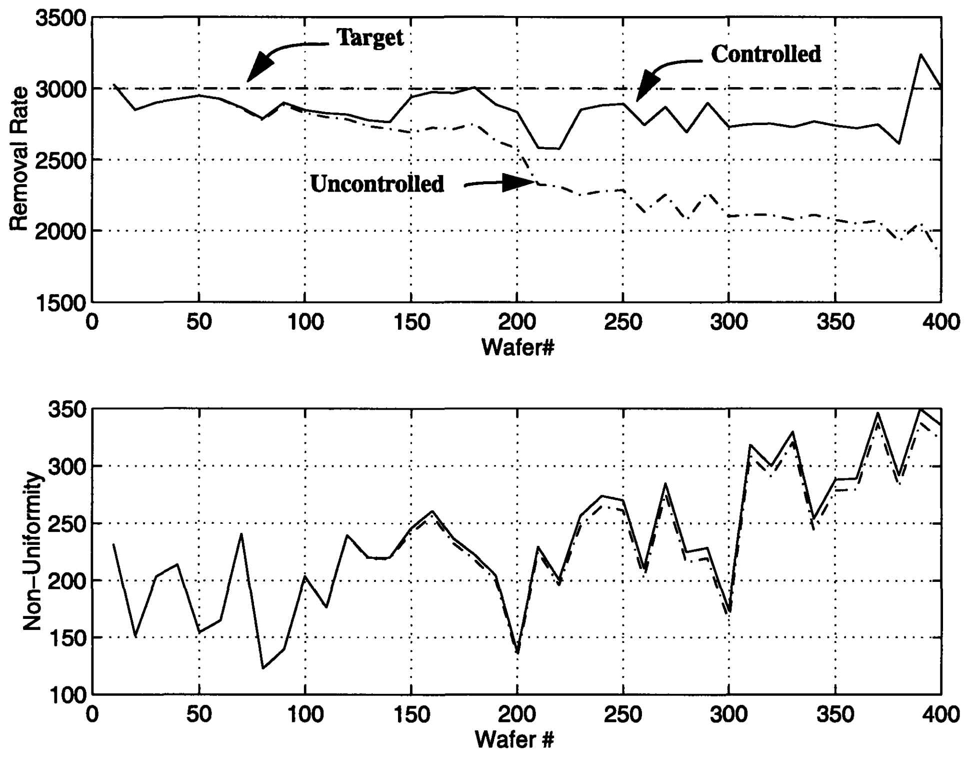


**Figure 11-3. HME Control of a Noisy Process**



This type of controller is not without problems. In fact, in situations where global changes in the process are occurring, the control model adapts one or more experts to account for this, but all experts are not updated and when the process moves into the region where a new expert takes over, the result is a large jump in the process output. We will see in the next section that the ANN-EWMA Controller responds better to these types of situations. However, we will show that in a problem with a large amount of model error and little process noise which is not due to any controlled inputs, the HME controller learns the true response and provides excellent control.

# Chapter 12

# Comparative Analysis

In this chapter we will explore families of processes which have different control needs. This will demonstrate that no single controller provides the best performance for all situations. We will see that for mostly linear processes with large amounts of noise, EWMA type controllers provide an effective simple solution. In fact, some of the more advanced controllers will not perform nearly as well. However, for many processes which have nonlinear behavior, we will see that the linear EWMA and PCC controllers are inherently unstable. When a small amount of error is present in the nonlinear control model and there is a large amount of process noise, the ANN-EWMA controller provides excellent control. Finally, we will see that when there is a large amount of model error and a small amount of process noise, the HME controller provides the optimal response because it is able to learn the true nonlinear response as the process runs. In light of this, we will now outline the processes we will investigate in this comparative analysis. For comparison, we will use a weighted mean-squared error (WMSE). This quantity is computed as the squared error of each output from target summed over the run, divided by the respective process targets, and summed together. This allows errors in more than one output to be weighted equally in the final measure.

The test cases we will study in this chapter are:

- An affine process with Auto-Regressive Moveing Average (ARMA) noise of the first order.
- An affine process with a linear drift buried in white noise.
- A quadratic process with small model error and a linear drift buried in white noise.
- A quadratic process with large model error and a linear drift buried in white noise.

These four processes were chosen because they represent a large portion of the variation seen in semiconductor processes. Specifically, the first two cases are very typical of CMP processes [Boning1, Boning2]. These CMP processes generally tend to have a large amount of ARMA noise or a persistent drift. The third and fourth processes are studied because more recent work done on newer CMP processes indicate that they may have second order characteristics.

In these comparisons, we leave out the direct adaptive EWMA controller in light of the fact that our disturbance state does not change in each example and the EWMA controller is pre-tuned to have optimal weights for each of the given processes. In addition, we have already outlined the direct adaptive EWMA controller's ability to provide on-line tuning, thereby allowing improvement over the EWMA controller when the process has a poor initial EWMA weight or the process changes and a new weighting scheme is required.

We will also not provide a completely optimized adaptive linear ANN network controller. Nor will we provide a simulation of a linear process in which the EWMA controller is unstable and the adaptive linear controller can stabilize the response because we have already done this as well. We will simply provide simulations which suggest a general idea of the performance obtainable with the linear adaptive controller, but will not guarantee that these controllers have optimal tuning parameters. For a detailed consideration of these topics, see [Åström, Narendra1].

## 12.1 An Affine Process with ARMA Noise of the First Order

The plant for this controller test case is of the form:

$$\underline{y}_p[n] = \underline{\alpha} + \Im\,\underline{u}[n] + r[n], \tag{12-1}$$

where

$$r[n] = r[n-1] + w[n] \tag{12-2}$$

and $w[n]$ is normally distributed white noise with zero mean and covariance matrix $\Lambda$. We will use:

$$\Im = \begin{bmatrix} 50.35 & -6.65 & 163.4 & 8.4 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix},\ \Lambda = \begin{bmatrix} 100 & 0 \\ 0 & 10 \end{bmatrix},\ \text{and } \underline{\alpha} = \begin{bmatrix} -1079 \\ -640.7 \end{bmatrix}. \tag{12-3}$$

The simulations throughout this section all use an identical realization of this type of noise, so that the different controllers can be effectively compared.

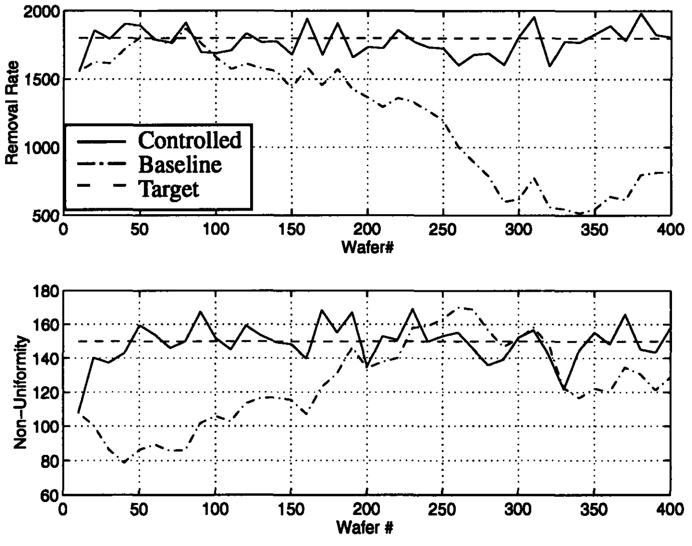
### 12.1.1 The EWMA Controller

The first step in using an EWMA controller is to develop an affine model. We assume this can be done with reasonable accuracy, and therefore use the following model:

$$\boldsymbol{B} = \begin{bmatrix} 53 & -7 & 172 & 8 \\ 14.4 & 19 & 29 & 5 \end{bmatrix},\ \text{with } \underline{\boldsymbol{a}} = \begin{bmatrix} -1079 \\ -640.7 \end{bmatrix}. \tag{12-4}$$

The second step is to determine the optimal EWMA weighting parameters. This was done by using the method described in Section 4.5 and the resulting weight vector is [ 0.9 ; 0.9 ]. The EWMA controller was run 100 times on this process and the weighted mean-squared error (WMSE), for each output was 7.7 x $10^{-3}$. A typical response from this controller is shown with the corresponding baseline process in Fig. 12-1.

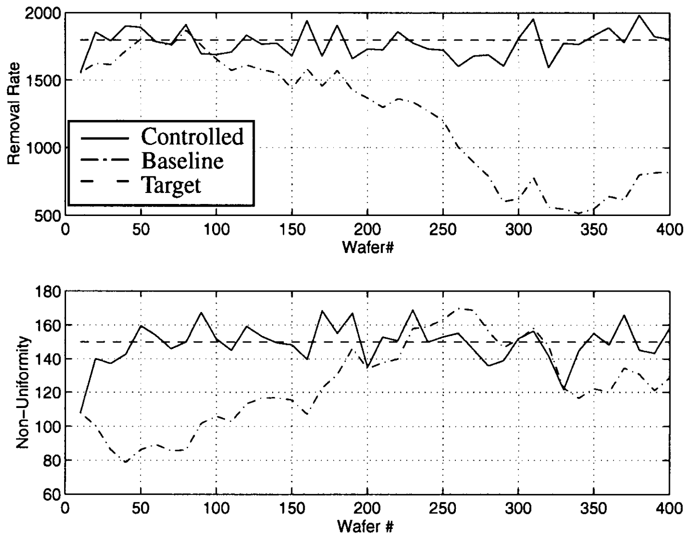
**Figure 12-1. EWMA Control of an ARMA1 Process**



### 12.1.2 The PCC Controller

The PCC controller uses the same affine model as the EWMA controller and the optimal PCC weighting parameters were determined in a similar manner. The resulting weight vectors are w1= [ 0.1 ; 0.6 ] and w2= [ 1 ; 1 ]. The PCC controller was also run 100 times on this process and the WMSE was 7.5 x $10^{-3}$. A typical response from this controller is shown with the corresponding baseline process in Fig. 12-2. Here we see that the PCC controller has slightly better mean deviation from the target due to the prediction in the PCC controller. However, the performance of the PCC controller is very close to that of the EWMA controller.

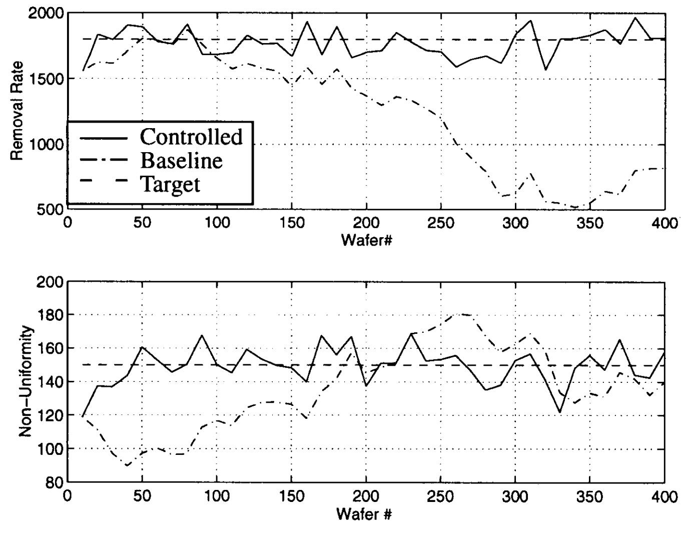
**Figure 12-2. EWMA Control of an ARMA1 Process**



## 12.1.3 The Adaptive Linear Controller

The adaptive linear controller also utilizes the same affine model as the starting process model. This is adapted using a learning rate of 0.002. This controller also performs slightly worse than the EWMA based controllers. Errors can be attributed to the misinterpretation of the noise in the system as coefficient error or non-optimal tuning of the learning rate. This error is balanced by the controller's ability to stabilize linear processes which are susceptible to large amounts of coefficient error (as described in Chapter 9). The typical response of this controller for the ARMA process is shown in Fig. 12-3. The resulting MSE for 100 runs was 1.3 x $10^{-2}$.

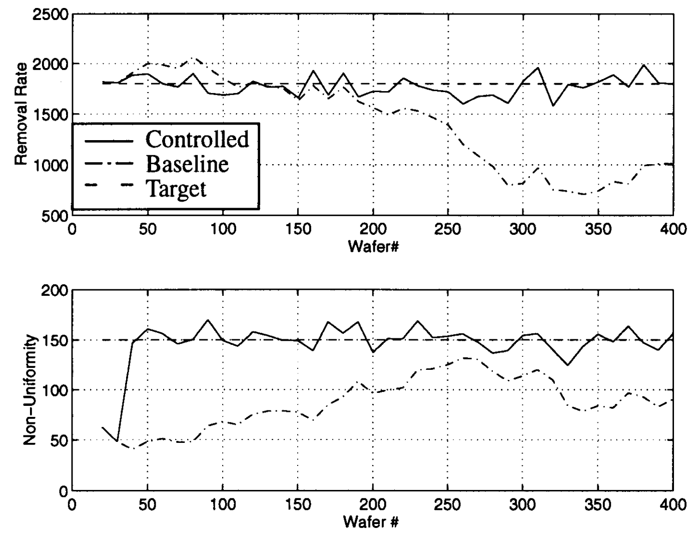
**Figure 12-3. Adaptive Linear Control of ARMA1 Process**



## 12.1.4 The ANN-EWMA Controller

The ANN-EWMA controller was also tested on this process. The neural network model was trained using a random sample of input-output pairs of the linear model outlined above. The EWMA weights for this controller were similar to that of the EWMA controller. Because the neural network is trained as a linear mapping and the adaptation of the bias layer in the output corresponds to the same update as the EWMA controller (in this case), we expect similar performance. This can be seen from the typical response shown in Fig. 12-4, and from the WMSE for 100 runs of 9.1 x $10^{-3}$. The small increase in WMSE is most likely due to the slight overfitting in training the neural network using a full nonlinear MLP network.

**Figure 12-4. ANN-EWMA Control of an ARMA1 Process**

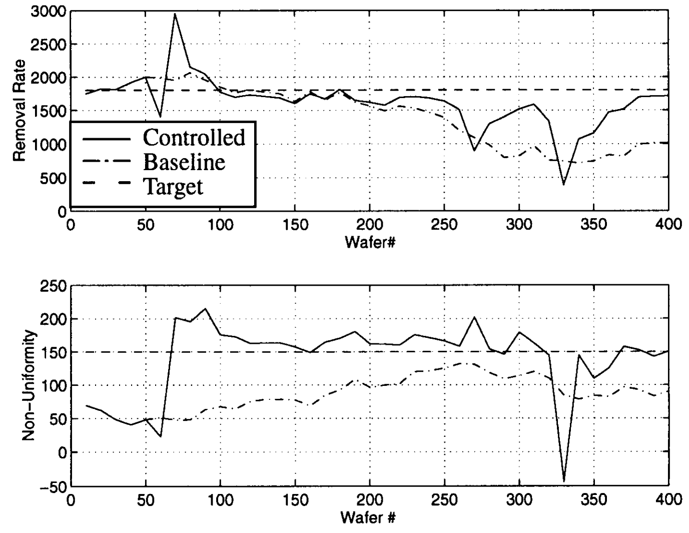


### 12.1.5 The HME Controller

The HME controller works quite poorly for this process. This is largely due to two effects. First, the HME controller has a credit assignment problem. In other words, the controller has the difficult task of assigning portions of the measurement error to specific weights of the network. This causes difficulties in the ARMA1 case where the amount of noise is large. This is because the noise we are adding is not attributable to any of the inputs we have control over and therefore assigning credit to an input for causing the error is an error itself. Situations like this occur in the CMP process when the pad wears independently of the equipment settings. The second problem with this controller is that the network itself is designed as a localized network. Specifically, each expert is assigned responsibility for the output of the network in a specific region of the input space. Therefore, if we are in one region and the process is changing in an ARMA1 fashion causing us to move into a region in the input space where another expert takes over, the current state of the ARMA noise has not been learned by the next expert. This causes a large dip or jump in the controlled output until the new expert has time to learn the ARMA noise. A controller of this type would be effective in learning characteristics of different processes which occur within the same equipment. It is, however, quite poor at learning global trends in the process.

We will provide the response of this controller here regardless of its poor performance. We do so because, as we will see, this controller provides good control over nonlinear processes which have medium to small amounts of process noise. The HME neural network was trained using another random sample of input-output pairs from the linear model outlined above. This model is adapted using a learning rate of 0.02. As we can see from the typical response of the controller shown in Fig. 12-5, the performance is poor. The spikes in the response occur due to the controller changing experts (which have not learned the global trends). The WMSE for 100 runs was 5.5 x $10^{-2}$.

**Figure 12-5. HME Control of ARMA1 Process**



### 12.1.6 Summary of Controller Performances for the ARMA1 Case

Provided below in Table 12-1, is a summary of the MSE for each of the controllers with the ARMA1 process. As we can see, the relatively simple EWMA controller provides excellent control of this process. The PCC controller provided slightly better results than the EWMA controller and the indirect adaptive linear controller and ANN-EWMA controllers both provided good control as well. The performance of the HME controller was quite poor.

**Table 12-1. Weighted Mean Squared Error for the ARMA1 Case**

| Method | MSE |
|---|---|
| EWMA | 7.7 x $10^{-3}$ |
| PCC | 7.5 x $10^{-3}$ |
| Adaptive Linear | 1.3 x $10^{-2}$ |
| ANN-EWMA | 9.1 x $10^{-3}$ |
| HME | 5.5 x $10^{-2}$ |

## 12.2 An Affine Process with a Linear Drift Buried in White Noise

The plant for this controller test case is of the form:

$$\underline{y}_p[n] = \underline{\alpha} + \mathfrak{B}\,\underline{u}[n] + w[n] + \delta n, \qquad (12\text{-}5)$$

where $w[n]$ is normally distributed white noise with zero mean and covariance matrix $\Lambda$. We will use:

$$\mathfrak{I} = \begin{bmatrix} 50.35 & -6.65 & 163.4 & 8.4 \\ 13.68 & 19.95 & 27.55 & 5.25 \end{bmatrix}, \Lambda = \begin{bmatrix} 100 & 0 \\ 0 & 10 \end{bmatrix}, \underline{\delta} = \begin{bmatrix} -40 \\ 4 \end{bmatrix}, \text{ and } \underline{\alpha} = \begin{bmatrix} -1079 \\ -640.7 \end{bmatrix}. \quad (12\text{-}6)$$

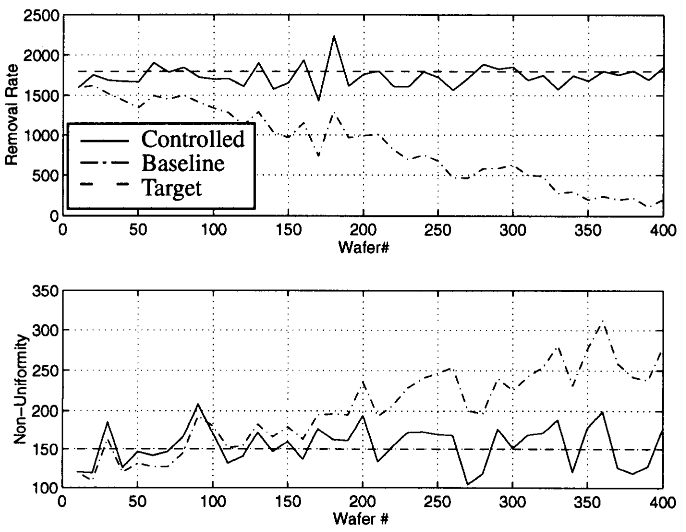
### 12.2.1 The EWMA Controller

As in the ARMA1 case, we use the following affine model:

$$\boldsymbol{B} = \begin{bmatrix} 53 & -7 & 172 & 8 \\ 14.4 & 19 & 29 & 5 \end{bmatrix}, \text{with } \underline{\boldsymbol{a}} = \begin{bmatrix} -1079 \\ -640.7 \end{bmatrix}. \quad (12\text{-}7)$$

The optimal EWMA weighting parameters were determined using the method described in Section 3.4 and the resulting weight vector is [ 0.7 ; 0.4 ]. The EWMA controller was run 100 times on this process and the WMSE was 3.11 x $10^{-2}$. A typical response from this controller is shown with the corresponding baseline process in Fig. 12-6.

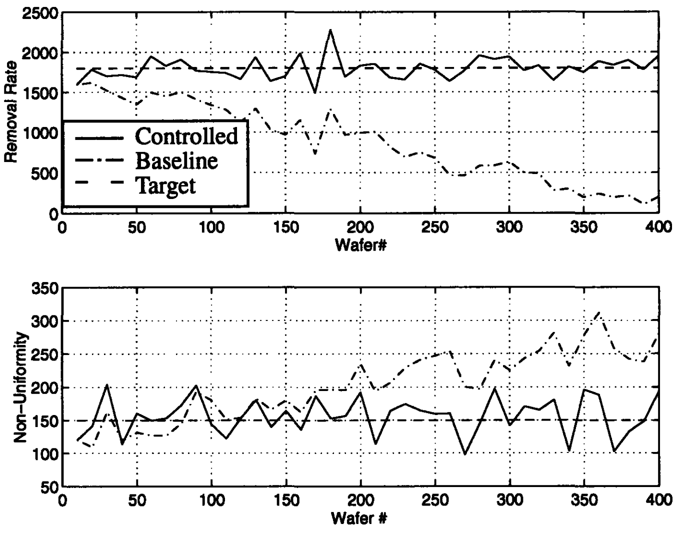
**Figure 12-6. EWMA Control of a Drifting Process**



### 12.2.2 The PCC Controller

The PCC controller uses the same affine model as the EWMA controller and the optimal PCC weighting parameters were determined in a similar manner. The resulting weight vectors are w1= [ 0.1 ; 0.6 ] and w2= [ 1 ; 1 ]. The PCC controller was also run 100 times on this process and the WMSE was 4.1 x $10^{-2}$. A typical response from this controller is shown with the corresponding baseline process in Fig. 12-7. We see that the PCC controller overcompensates for the baseline noise (due to the prediction term) and thus the EWMA controller achieves slightly better control.

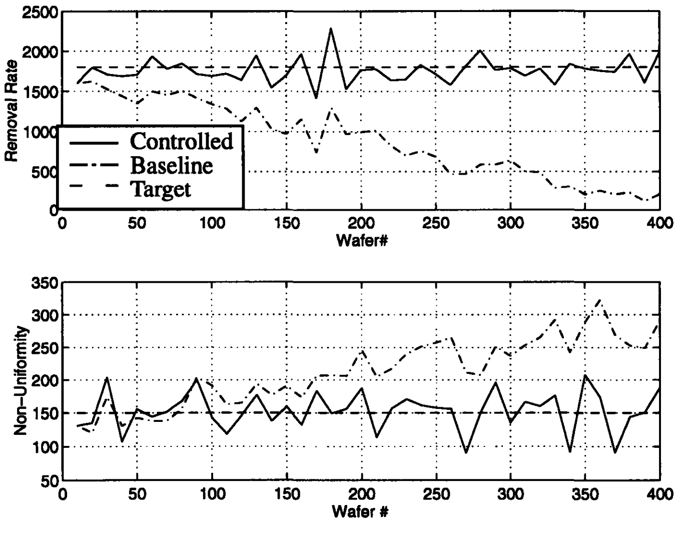
**Figure 12-7. PCC Control of a Drifting Process**



### 12.2.3 The Adaptive Linear Controller

The adaptive linear controller adapts the process model above using a learning rate of 0.002. This controller compensates for the process drifts, but not nearly as well as the EWMA or PCC controller. A typical response of this controller for the drifting process is shown in Fig. 12-8 and the resulting WMSE for 100 runs was 8.9 x $10^{-2}$.

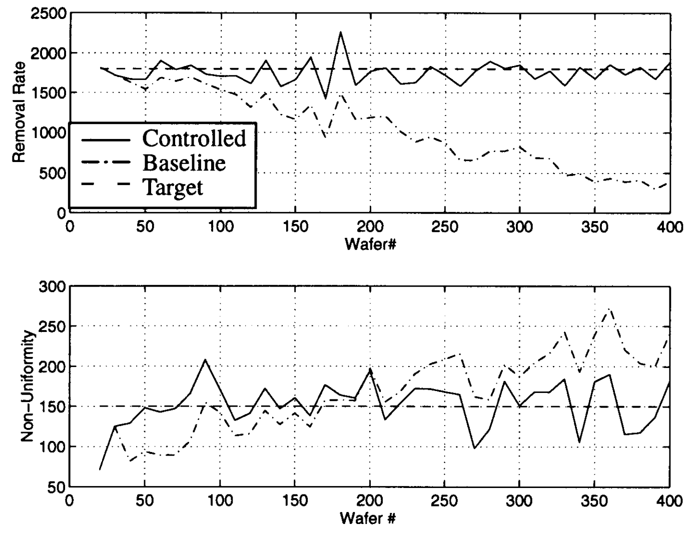
**Figure 12-8. Adaptive Linear Control of a Drifting Process**



### 12.2.4 The ANN-EWMA Controller

The ANN-EWMA controller was again trained using a random sample of input-output pairs of the linear model outlined above. The EWMA weights for this controller were the same as that of the EWMA controller. Again, we obtain a performance similar to the EWMA controller. A typical response is shown in Fig. 12-9 and the WMSE for 100 runs was 3.3 x $10^{-2}$.

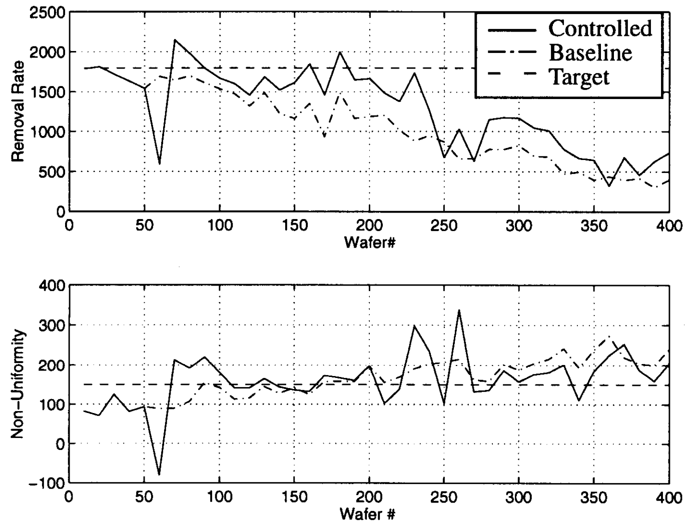
**Figure 12-9. ANN-EWMA Control of a Drifting Process**



## 12.2.5 The HME Controller

The HME controller again works quite poorly for this process, for similar reasons. A typical response of the controller is shown in Fig. 12-10 and the WMSE for 100 runs was 2.8 x $10^{-1}$.

**Figure 12-10. HME Control of a Drifting Process**



## 12.2.6 Summary of Controller Performances for the Affine Process with a Linear Drift Buried in White Noise

Provided below in Table 12-2 is a summary of the WMSE for each of the controllers with the drifting process. As we can see, the relatively simple EWMA controller provides excellent control of this process along with its nonlinear ANN-EWMA counterpart. The PCC controller provided slightly worse results than the EWMA due to errors caused by the prediction term and the indirect adaptive linear controller provided good control as well. Again, the performance of the HME controller was quite poor.

**Table 12-2. Weighted Mean Squared Error for an Affine Process with a Linear Drift Buried in White Noise**

| Method | WMSE |
|---|---|
| EWMA | $3.1 \times 10^{-2}$ |
| PCC | $4.1 \times 10^{-2}$ |
| Adaptive Linear | $8.9 \times 10^{-2}$ |
| ANN-EWMA | $3.3 \times 10^{-2}$ |
| HME | $2.8 \times 10^{-1}$ |

## 12.3 A Second Order Process with Small Model Error and a Linear Drift Buried in White Noise

The plant for this controller test case is a full second order polynomial function of the inputs,

$$f(u[n])(i) = \sum_{j=0}^{n} \beta(i,j)u(i)u(j), \tag{12-8}$$

plus a linear drift and noise:

$$\underline{y}_p[n] = \underline{\alpha} + f(u[n]) + w[n] + \underline{\delta}n, \tag{12-9}$$

where $w[n]$ is normally distributed white noise with zero mean and covariance matrix $\Lambda$ and $\underline{\delta}$ is a vector of drift rates. We will use:

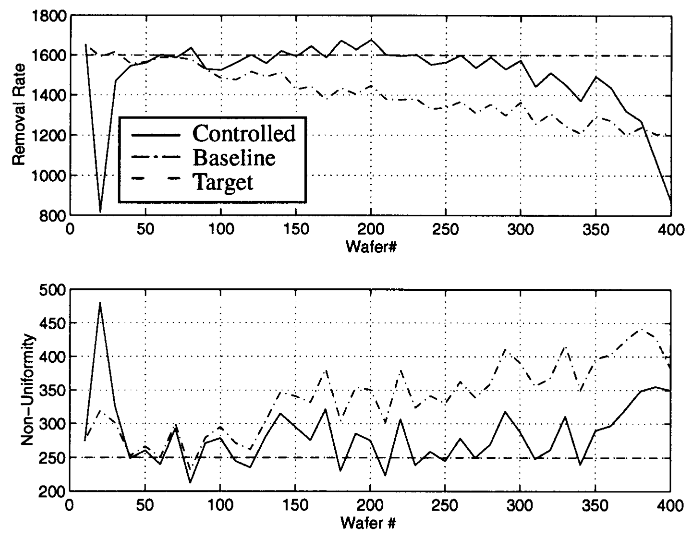
$$\Lambda = \begin{bmatrix} 100 & 0 \\ 0 & 10 \end{bmatrix}, \underline{\delta} = \begin{bmatrix} -10 \\ 4 \end{bmatrix}, \underline{\alpha} = \begin{bmatrix} -1079 \\ -640.7 \end{bmatrix}, \text{and} \tag{12-10}$$

$$\beta(i,j) = \begin{bmatrix} 1622.2 & 215.2 & -16.4 & 268.3 & 2.3 & 12 & 6.5 & 3 & -30.4 & 1.4 & 6 & 26.5 & 34.3 & -12.4 & 0.9 & -5.2 & -5.1 & -11 & 18.55 & 4.6 & 9.7 \\ 152.7 & -0.4 & 3.1 & 14.4 & 0.9 & -3 & 14.3 & -12.2 & -13.3 & 9 & 15.5 & -6.9 & 6 & -3.3 & 9.7 & -2 & -1.9 & 2.7 & 1.3 & 5.7 & 0.3 \end{bmatrix} \tag{12-11}$$

### 12.3.1 The EWMA Controller

As in the linear process cases above, we use a linearized version of a second order model with a coefficient matrix where each coefficient was normally distributed around the true value given above and with each coefficient having a standard deviation of 10% of the mean value. What we find is that the EWMA controller is often unstable for even the most conservative weights. Therefore, its WMSE is $\infty$. This controller can remain stable for an extended period if the noise is not too large. This case is shown in Fig. 12-11.

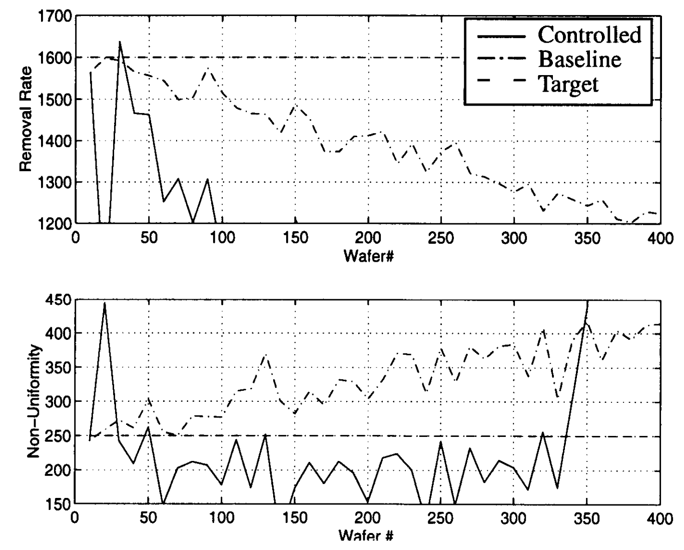
**Figure 12-11. EWMA Control of a 2nd Order Process with Small Model Error**



## 12.3.2 The PCC Controller

Similar to the EWMA controller, the PCC controller is also unstable for this process and its WMSE is ∞ as well. A typical response from this controller is shown in Fig. 12-12.

**Figure 12-12. PCC Control of a 2nd Order Process with Small Model Error**

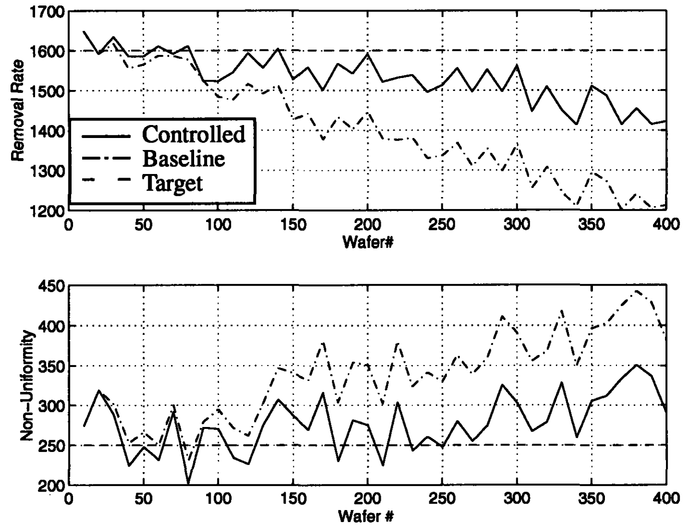


## 12.3.3 The Adaptive Linear Controller

The adaptive linear controller allows one to stabilize the response of a linear controller by adapting the process model above using a learning rate of 0.002. This controller compensates for the process drifts as well, providing an acceptable improvement over the baseline process. A typical response of this controller for the drifting process is shown in Fig. 12-13 and the resulting

WMSE for 100 runs was 9.9 x $10^{-2}$.

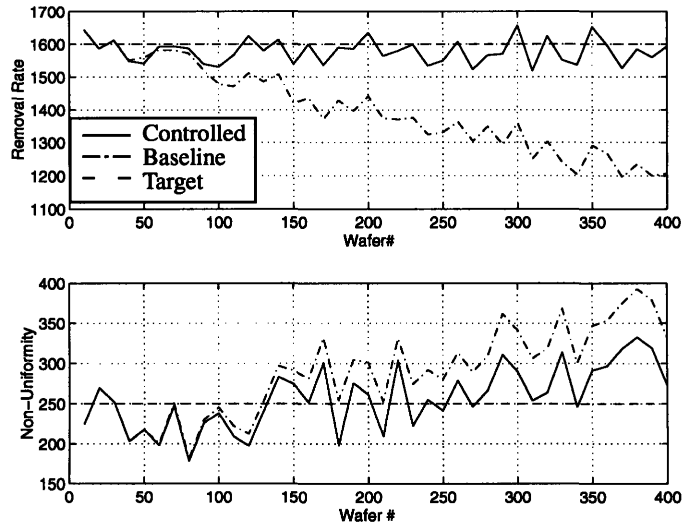
**Figure 12-13. Adaptive Linear Control of a 2nd Order Process with Small Model Error**



## 12.3.4 The ANN-EWMA Controller

The ANN-EWMA controller (a nonlinear map with an EWMA update of the bias terms) was trained by generating input-output pairs using a model with a coefficient matrix where each coefficient was normally distributed around the true value given above and with each coefficient having a standard deviation of 10% of the mean value. The EWMA weights for this controller were chosen so as to provide moderate performance in order to maintain a high level of stability. Here we see that this controller gives excellent control of the nonlinear process with elimination of the drift. A typical response is shown in Fig. 12-14 and the WMSE for 100 runs was 2.1 x $10^{-2}$.

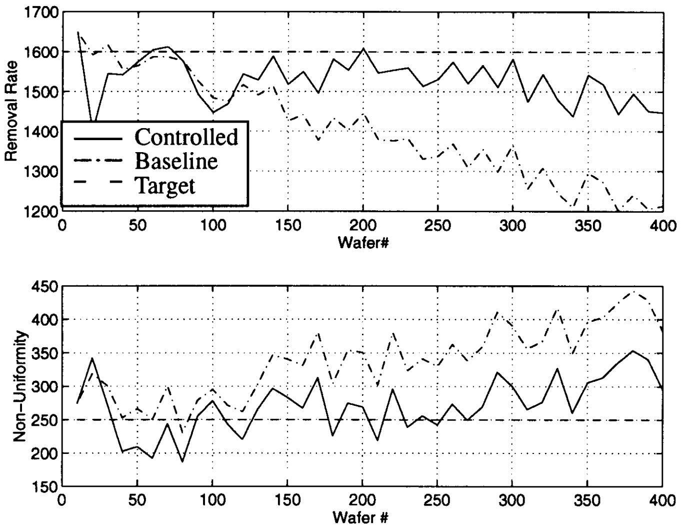
**Figure 12-14. ANN-EWMA Control of a 2nd Order Process with Small Model Error**



### 12.3.5 The HME Controller

The HME controller model was trained using the same model as that used for the ANN-EWMA controller. Here we see that although the HME controller is able to maintain sufficient control (since there is not a tremendous amount of noise) its performance is moderate in comparison to the ANN-EWMA controller because it is not designed to capture global trends in the process. A typical response of the controller is shown in Fig. 12-15 and the WMSE for 200 runs was 3.6 x $10^{-2}$.

**Figure 12-15. HME Control of a Drifting Process**



### 12.3.6 Summary of Controller Performances for a Second Order Process with Small Model Error and a Linear Drift Buried in White Noise

Provided below in Table 12-3 is a summary of the WMSE for each of the controllers with the second order process with small model error and a linear drift buried in white noise. As we can see, the relatively simple EWMA and PCC controllers are unstable, while the linear adaptive controller is stable and provides adequate control. The ANN-EWMA controller provides the best response due to it's ability to monitor global process drift. The HME controller also provides good control since the noise is small and the controller constructs a nonlinear model.

**Table 12-3. Weighted Mean Squared Error for a Second Order Process with Small Model Error and Linear Drift Buried in White Noise**

| Method | WMSE |
|---|---|
| EWMA | $\infty$ |
| PCC | $\infty$ |
| Adaptive Linear | 9.9 x $10^{-2}$ |
| ANN-EWMA | 2.1 x $10^{-2}$ |
| HME | 3.6 x $10^{-2}$ |

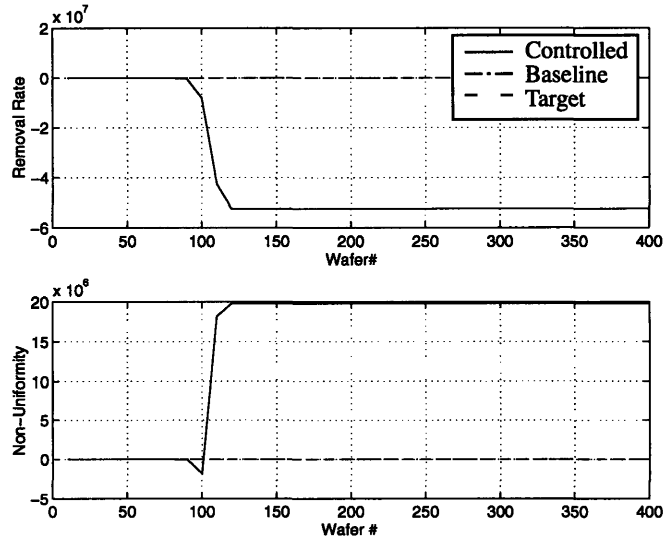
## 12.4 A Second Order Process With Large Model Error and a Linear Drift Buried in White Noise

The plant for this controller test case is a full second order polynomial function of the inputs, plus a linear drift and noise (identical the that in the previous section). In these cases, however, there will be a much larger model error than in the previous cases.

### 12.4.1 The EWMA Controller

As in the previous case, we use a linearized version of the second order model with a coefficient matrix where each coefficient was normally distributed around the true value and with each coefficient having a standard deviation of 100% of the mean value. What we find is that the EWMA controller is unstable for even the most conservative weights. Therefore, its WMSE is $\infty$. This controller can remain stable for a short period if the noise is not too large. This case is shown in Fig. 12-11.

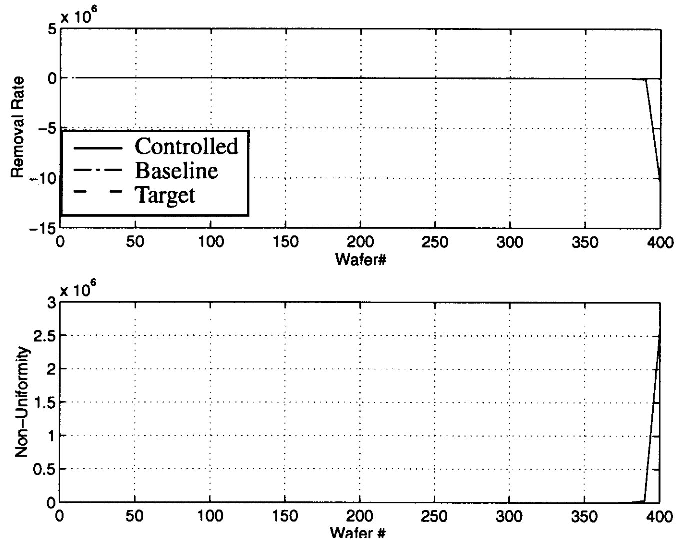
**Figure 12-16. EWMA Control of a 2nd Order Process with Small Model Error**



### 12.4.2 The PCC Controller

Similar to the EWMA controller, the PCC controller is also unstable for this process and its WMSE is $\infty$ as well. A typical response from this controller is shown in Fig. 12-12.

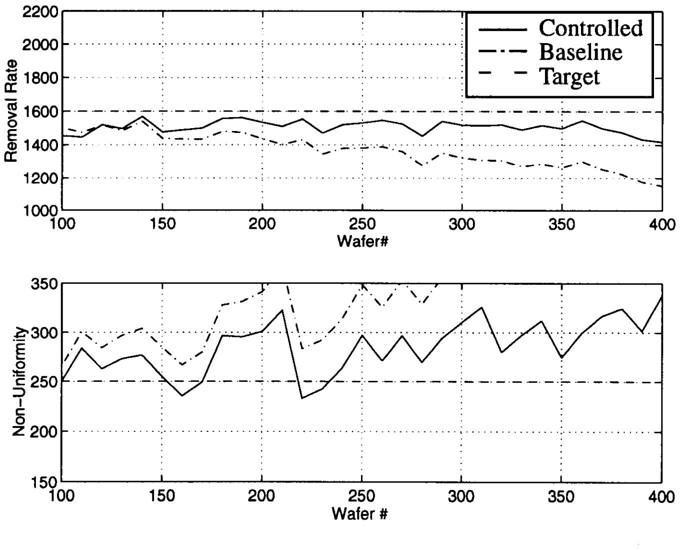
Figure 12-17. PCC Control of a 2nd Order Process with Small Model Error



## 12.4.3 The Adaptive Linear Controller

The adaptive linear controller allows one to stabilize the response of a linear controller by adapting the process model above using a learning rate of 0.002. This controller compensates for the process drifts as well, providing an acceptable improvement over the baseline process. A typical response of this controller for the drifting process is shown in Fig. 12-13 and the resulting WMSE for 100 runs was 4.5 x $10^{-2}$.

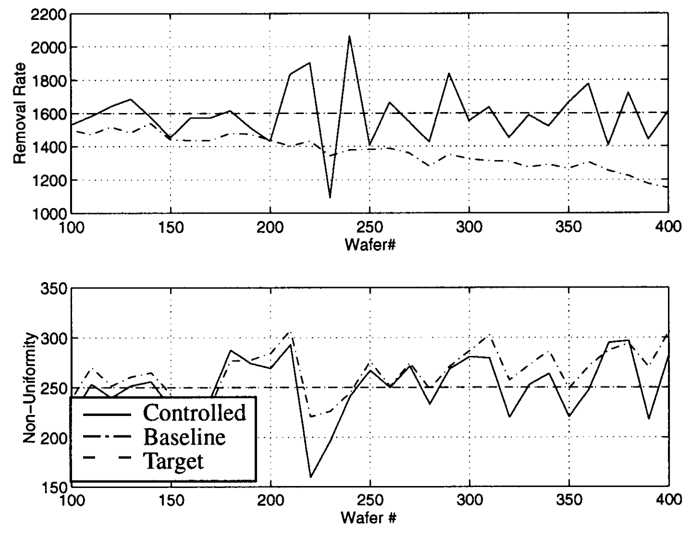
Figure 12-18. Adaptive Linear Control of a 2nd Order Process with Small Model Error



## 12.4.4 The ANN-EWMA Controller

The ANN-EWMA controller was trained by generating input-output pairs using a model with a coefficient matrix where each coefficient was normally distributed around the true value given above and with each coefficient having a standard deviation of 200% of the mean value. The EWMA weights for this controller were chosen small so as minimize the chances of instability. Even so, with the large amount of model error this controller was found to be unstable when significant process noise was added. This situation generally occurred for about 20% of the runs. For the remaining cases control was actually quite good, and a typical response is shown in Fig. 12-14. The WMSE for 100 runs was 1.2 x $10^{-2}$, but this does not consider the few unstable cases.

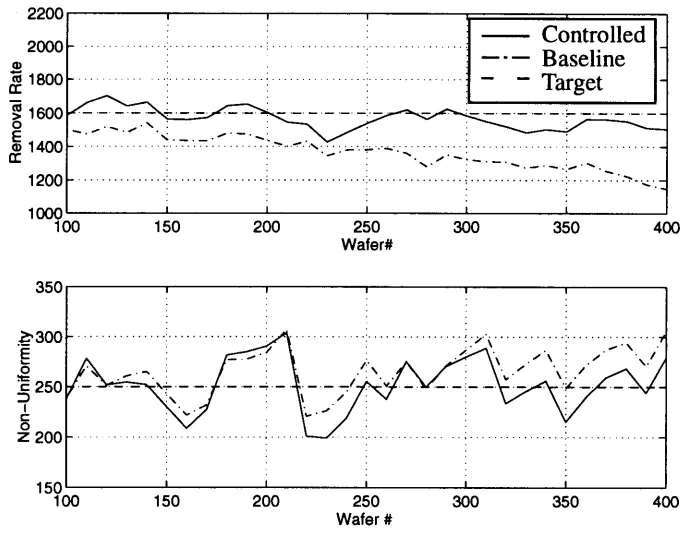
**Figure 12-19. ANN-EWMA Control of a 2nd Order Process with Small Model Error**



## 12.4.5 The HME Controller

The HME controller gave excellent performance for this case as well as providing the ability to recover from the large model. In 100 runs at this model error, the controller did not go unstable. However, when the model error was extremely large (std. dev. = 5 times the mean on each coefficient) the HME controller also exhibited unstable characteristics. A typical response of the controller is shown in Fig. 12-15 and the WMSE for 100 runs was 6.8 x $10^{-3}$.

Figure 12-20. HME Control of a Drifting Process



### 12.4.6 Summary of Controller Performances for a Second Order Process With Large Model Error and a Linear Drift Buried in White Noise

Provided below in Table 12-4 is a summary of the WMSE for each of the controllers with the second order process with small model error and a linear drift buried in white noise. As we can see, the relatively simple EWMA and PCC controllers are unstable, while the linear adaptive controller is stable a provides adequate control. The ANN-EWMA controller provides good control but is sensitive to process noise when large amounts of error are present and occassionally goes unstable. The HME controller provides the best response and is quite reliable even with large amounts of model error.

Table 12-4. Weighted Mean Squared Error for a Second Order Process with Large Model Error and Linear Drift Buried in White Noise

| Method | WMSE |
|---|---|
| EWMA | $\infty$ |
| PCC | $\infty$ |
| Adaptive Linear | $4.5 \times 10^{-2}$ |
| ANN-EWMA | $1.2 \times 10^{-2}$ |
| HME | $6.8 \times 10^{-3}$ |

## 12.5 Overall Summary of Controller Performances

These simulations demonstrate one important concept, that different processes require different controllers. The EWMA and PCC controllers provide the best control of approximately affine systems which are subject to unmodeled disturbances such as ARMA noise, drifts, and shifts. The PCC controller slightly outperforms the EWMA controller when low frequency process variations are large relative to high frequency process noise (as in the ARMA1 case shown above). The EWMA controller provides the best response to processes where the noise is large relative the low frequency variations (as in the linear drift buried in white noise case). These controllers begin to fall short with nonlinear processes. In cases such as the second order processes described in Sections 12.3 and 12.4, the nonlinear neural network controllers prevail. In particular, when the model errors are small, the ANN-EWMA controller provides excellent control due to its ability to capture the nonlinear effects of the process model while monitoring global trends with an EWMA update of the biases in the output layer. When the process model is inaccurate or changing, the simple update of the biases is not enough, and the more fully adaptive HME controller provides improved control with its ability to adapt to model errors.

# Chapter 13

# Conclusions and Future Work

We have shown the successful application of the EWMA controller to the run by run control of chemical-mechanical polishing. We have demonstrated the ability of this controller to greatly extend pad life while maintaining tight control of the wafer-to-wafer uniformity. In addition to greatly improving the control of removal rate, user preferences for weighting input variability and output error, input constraints and discretization, and optimal EWMA weight determination have been implemented. The stability condition for this MIMO controller have been derived and shown to be easily checked for a given process. Further, an analytical solution to the optimal EWMA weight has been presented for the SISO EWMA control of an affine process with a linear drift buried in white noise. This is an important step towards a full analytical solution of the optimal EWMA weights for a MIMO process. This would greatly improve the applicability of such a controller in manufacturing lines, where equipment operators are not trained to tune feedback controllers. The direct adaptive EWMA controller is a first step in developing such a self-tuning EWMA controller. We have shown that this controller can improve performance when the initial weighting parameters are not optimal. More work needs to be done to include process shifts and other disturbances and verify methods for properly estimating the disturbance state in order to create a fully self-tuning EWMA controller.

An initial exploration into neural network controllers has been presented here as a first step towards developing controllers which take full advantage of emerging empirical and physical models of the CMP process (particularly those incorporating process dynamics) [Runnels1, Chang1]. Other potential control techniques such as classical LQG, stochastic dynamic programming, and neurodynamic programming approaches should also be considered. We have shown the ability of a linear adaptive controller to provide stable control of affine processes which have poor initial process models as well as some nonlinear processes. We have also demonstrated an ANN-EWMA controller which utilizes a fully nonlinear model, adapted by an update of the bias terms using an EWMA, to effectively control second order processes with linear drifts and white noise.

Finally, we have outlined a controller with local adaptation of the process model which provides fully adaptive nonlinear control. While the neural network controllers presented in this work have focused on static input-output models, future work will explore the use of controllers which utilize current research being done on time-dependent dynamics of the CMP process [Runnels1]. In addition, the ANN controllers presented in this work have focused on the control of a small number of outputs, but are not limited to this. Future work will also consider systems with a large number of outputs. This is becoming an important topic as full-wafer sensors emerge in CMP. These sensors will allow pattern dependencies to be monitored, and it is hoped that these more advanced control techniques can effectively model and control pattern dependent polishing. While this thesis has focused on the CMP process, the algorithms and control methods presented in this thesis are general and their application to other semiconductor processes such as plasma etch, sputter deposition, and other processes remains an open area for future work.

# References

[Ahmed] M. Ahmed, "A New Algorithm for State Estimation of Stochastic Linear Discrete System," *IEEE Trans. on Auto. Control*, vol. 39, no. 8, Aug., 1994.

[Aloneftis] A. Aloneftis, *Stochastic Adaptive Control Results and Simulations (Lecture Notes in Control and Information Sciences, 98)*, Springer-Verlag, Berlin, Heidelberg, Germany, 1987.

[Altman] A. Altman, "Applying Run by Run Process Control to Chemical-Mechanical Polishing of Sub-Micron VLSI: A Technological and Economic Case Study," S.M. Thesis, MIT EECS, May 1995.

[Åström1] K. Åström and B. Wittenmark, *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.

[Åström2] K. Åström and T. McAvoy, "Intelligent Control: An Overview and Introduction," *Handbook of Intelligent Control*, pp. 3-34, D. White and D. Sofge (Eds.), Van Nostrand Reinhold, New York, NY, 1992.

[Baras] J.S. Baras and N.S. Patel, "Robust control of discrete time generalized dynamical systems: Finite and infinite time results," *Proc. of the American Control Conference*, pp. 1990-1994, 1995.

[Bernard] P. Bernhard, "Minimax versus stochastic partial information control," *Proceedings of the 33rd Conference on Decision and Control*, Dec., 1994.

[Bertsekas] D. Bertsekas, *Dynamic Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Bojkav] B. Bojkav and R. Luus, "Tine Optimal Control of High Dimensional Systems by Iterative Dynamic Programming," *The Canadian Journal of Chem. Eng.*, vol. 73, pp. 380-90, June, 1995.

[Boning1] D. Boning, W. Moyne, T. Smith, J. Moyne, R. Trelfeyan, A. Hurwitz, S. Shellman, and J. Taylor, "Run By Run Control of Chemical Mechanical Polishing," *IEEE Proc. of the 17th Int. Elect. Manuf. Tech. Symp.*, Oct. 1995.

[Boning2] D. Boning, W. Moyne, T. Smith, J. Moyne, and A. Hurwitz, "Practical Issues in Run by Run Control," *Sixth Annual SEMI/IEEE ASMC*, Boston, Nov. 1995.

[Box] G.E.P. Box and T. Kramer, "Statistical Process Control and Automated Process Control -- A Discussion," Center for Quality and Productivity Improvement, University of Wisconsin, 1990.

[Butler] S. Butler and J. Stefani, "Application of Predictor Corrector Control to Polysilicon Gate Etching," *Proc. Amer. Control. Conf.*, June 1993.

[Chang1] E. Chang, B. Stine, T. Maung, R. Divecha, D. Boning, J. Chung, K. Chang, G. Ray, D. Bradbury, S. Oh, D. Bartelink, "Using a Statistical Metrology Framework to Identify Random and Systematic Sources of Intra-Die ILD Thickness Variation for CMP Processes," *IEDM*, Dec. 1995.

[Chang2] F. Chang, "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control," *IEEE Control Systems Magazine*, April, 1990.

[Crowder1] S. Crowder, "Design of Exponentially Weighted Moving Average Schemes," *Journal of Quality Tech.*, Vol. 21, No. 3, July 1989.

[Corwder2] S. Crowder and M. Hamilton, "An EWMA for Monitoring a Process Standard Deviation," *Journal of Quality Tech.*, Vol. 24, No. 1, Jan. 1992.

[Davis] J. Davis et. al., "Improved Within-Wafer Uniformity Modeling Through The Use of Maximum-Likelihood Estimation of the Mean and Covariance Surfaces," *Proc. of the 187th Meeting of the Electrochem. Soc.*, May, 1995.

[DARPA] DARPA, *DARPA Neural Network Study October 1987-February 1988*, AFCEA International Press, Fairfax, VA 1988.

[DelCastillo] E. Del Castillo and A. Hurwitz, "Run to Run Process Control: a Review and Some Extensions," *submitted to J. Qual. Tech.*, Feb. 1994.

[Deming] W.E. Deming, "Out of the Crisis," MIT Center for Advanced Engineering Study, Cambridge, MA, 1986.

[Dorf] R. Dorf, *Modern Control Systems, Second Edition,* Addison-Wessley, Reading, Mass., 1992.

[Duncan] B. Pasek-Duncan, "Stochastic Adaptive Control-An Historical Perspective," *Proceedings of the 33rd Conference on Decision and Control*, Dec., 1994.

[Fury] M. Fury, "Emerging Developments in CMP for Semiconductor Planarization," *Solid State Tech.*, pp. 47-54, April 1995.

[Franklin] J. Franklin and D. White, "Artificial Neural Networks in Manufacturing and Process Control," *Handbook of Intelligent Control*, pp. 233-258, D. White and D. Sofge (Eds.), Van Nostrand Reinhold, New York, NY, 1992.

[Gill] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, London, 1981.

[Grace] A. Grace, "Optimization Toolbox for use with MATLAB," The Math Works Inc., Nov. 1990.

[Haykin] S. Haykin, "Neural Networks, A Comprehensive Foundation," Macmillann Englewoods, New Jersey, 1994.

[Hembree] G. B. Hembree, "Recursive Estimation of the Weighting Factor for EWMA Control Charts from Autocorrelated Data," personal communication, 1995.

[Hofer] D. Sbarbaro-Hofer, D. Neumerkel, and K. Hunt, "Neural Control of a Steel Rolling Mill," *IEEE Control Systems Magazine*, June, 1993.

[Hu1] A. Hu, X. Zhang, E. Sachs, and P. Renteln, "Application of Run by Run Controller to the Chemical-Mechanical Planarization Process, Part I," *IEEE Proc. of the 15th Int. Elect. Manuf. Tech. Symp.*, Oct. 1993.

[Hu2] A. Hu, H. Du, S. Wong, P. Renteln, and E. Sachs, "Application of Run by Run Controller to the Chemical-Mechanical Planarization Process, Part II," *IEEE Proc. of the 16th Int. Elect. Manuf. Tech. Symp.*, Oct. 1994.

[Hu3] A. Hu, H.-P. Dun, P. Renteln, and E. Sachs, "Sensor Development and Process Control for Chemical-Mechanical Planarization of Multilevel Interconnect Devices," *Electrochem. Soc. Meeting*, June 1995.

[Hunter] J. S. Hunter, "The Exponentially Weighted Moving Average," *Journal of Quality Tech.*, Vol. 18, No. 4, October 1986.

[Ingolfsson] A. Ingolfsson and E. Sachs, "Stability and Sensitivity of an EWMA Controller," *J. Qual. Tech.*, vol. 25, no. 4, pp. 271-287, Oct. 1993.

[Jacobs] R. Jacobs and M. Jordan, "A Competitive Modular Connectionist Architecture," *Advances in Neural Information Processing Systems 3* (R Lippman J. Moody, and D. Touretzky), pp. 767-773, San Mateo, CA: Morgan Kaufmann.

[Jairath] R. Jairath and L. Markert, "Metrology and Process Control Issues in Chemical Mechanical Polishing," *NIST 1995 Semiconductor Characterization Workshop*, Jan. 1995.

[Jordan] M. Jordan, "Generic constraints on underspecified target trajectories," *Proceedings of IJCNN*, IEEE, New York, NY, June 1989.

[Koenig] D. M. Koenig, *Control and Analysis of Noisy Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[Kraft] L. Kraft and D. Campagna, "A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems," *IEEE Control Systems Magazine*, April, 1990.

[Lucas] J. Lucas and M. Saccucci, "Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements," *Technometrics*, Vol. 32, No. 1, Feb. 1990.

[Maitelli] A. Maitelli and T. Yoneyama, "A Two-stage Dual Suboptimal Controller for Stochastic Systems using Approximate Moments," *International Control*, 1994.

[Martinez] M. Martinez, "Chemical-mechanical polishing: Route to global planarization," *Solid State Tech.*, p. 26, May 1994.

[Meystel] A. Meystel, "Intelligent Control: Issues and Perspectives," *Proceedings of the IEEE Workshop on Intelligent Control 1985*, Aug., 1985

[Morari] M. Morari and E. Zafiriou, *Robust Process Control*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Moyne1] W. Moyne, "Run by Run Control: Interfaces, Implementation, and Integration," S. M. Thesis, MIT EECS, May 1995.

[Moyne2] J. Moyne, R. Telfeyan, A. Hurwitz and J. Taylor, "A Process-Independent Run-to-Run Controller and Its Application to Chemical-Mechanical Planarization," to be presented, *Sixth Annual SEMI/IEEE ASMC*, Boston, Nov. 1995.

[Moyne3] J. Moyne and L. McAfee, Jr., "A Generic Cell Controller for the Automated VLSI Manufacturing Facility," *IEEE Trans. Semi. Manuf.*, vol. 5, no. 2, pp. 77-87, May 1992.

[Moyne4] J. Moyne, H. Etemad, and M. Elta, "A Run-to-Run Control Framework for VLSI Manufacturing," *Proc. Microelec. Processes, Sensors, and Controls*, SPIE, vol. 2091, pp. 379-390, 1994.

[Mozumder] P. Mozumder, S. Saxena, and D. J. Collins, "A Monitor Wafer Based Controller for Semiconductor Processes," *IEEE Trans. Semi. Manuf.*, vol. 7, no. 3, pp. 400-411, Aug. 1994.

[Narendra1] K. Narendra, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Narendra2] K. Narendra, "Parameter Adaptive Control-The End... ...or The Beginning?," *Proceedings of the 33rd Conference on Decision and Control*, Dec., 1994.

[Narendra3] K. Narendra, "Adaptive Control of Dynamical Systems Using Neural Networks," *Handbook of Intelligent Control*, pp. 141-84, D. White and D. Sofge (Eds.), Van Nostrand Reinhold, New York, NY, 1992.

[Nguyen] D. Nguyen and B. Widrow, "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, April, 1990.

[Passino] K. Passino, "Bridging the Gap between Conventional and Intelligent Control," *IEEE Control Systems Magazine*, June, 1993.

[Ren] W. Ren and P. Kumar, "Stochastic Adaptive Prediction and Model Reference Control," *IEEE Trans. on Auto. Control*, vol. 39, no. 10, Oct., 1994.

[Rumelhart] D. Rumelhart, G. Hinton, R. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. I, pp. 318-362, D. Rumelhart and J. McClelland (Eds.), MIT Press, Cambridge, MA, 1986.

[Runnels1] S. Runnels and L. Eyman, "Tribology Analysis of Chemical-Mechanical Polishing," *J. Electrochem. Soc.*, vol. 141, no. 6, pp. 1698-1701, June, 1994.

[Runnels2] S. Runnels, "Feature-Scale Fluid-Based Erosion Modeling for Chemical-Mechanical Polishing," *J. Electrochem. Soc.*, vol. 141, no. 7, pp. 1900-1904, July, 1994.

[Sachs] E. Sachs, A. Hu, and A. Ingolfsson, "Run by Run Process Control: Combining SPC and Feedback Control," *IEEE Trans. Semi. Manuf.*, vol. 8, no. 1, pp. 26-43, Feb. 1995.

[Sofge] D. Sofge and D. White, "Applied Learning: Optimal Control for Manufacturing," *Handbook of Intelligent Control*, pp. 259-281, D. White and D. Sofge (Eds.), Van Nostrand Reinhold, New York, NY, 1992.

[Stell] M. Stell, "Aluminum CMP Process Development Issues", *Proc. of the 1995 SRC Topical Research Conference on CMP*, July 1995.

[Werbos1] P. Werbos, “Neurocontrol and Supervised Learning: An Overview and Evaluation,” *Handbook of Intelligent Control*, pp. 65-90, D. White and D. Sofge (Eds.), Van Nostrand Reinhold, New York, NY, 1992.

[Werbos2] P. Werbos, T. McAvoy, and T. Su, “Neural Networks, System Identification, and the Control of the Chemical Process Industries,” *Handbook of Intelligent Control*, pp. 283-356, D. White and D. Sofge (Eds.), Van Nostrand Reinhold, New York, NY, 1992.

[White] D. White, “In-Situ Wafer Uniformity Estimation using Principal Component Analysis Methods,” S. M. Thesis, MIT EECS, May 1995.

[Widrow] B. Widrow and M. Hoff, “Adaptive Switching Circuits,” *1960 IRE WESCON Convention Record*, pp.96-104, IRE, New York, NY, 1960.