OPTIMIZATION OF A SELF-TUNING OPEN-LOOP CONTROLLER

by

Ross Bennett Levinsky

S.B., Aeronautics and Astronautics
Massachusetts Institute of Technology
(1989)

S.M., Mechanical Engineering
Massachusetts Institute of Technology
(1992)

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Mechanical Engineering

at the
Massachusetts Institute of Technology
February 1995

Signature of Author [illegible]
Department of Mechanical Engineering
January 17, 1995

Certified by [illegible]
Harry West
Thesis Supervisor

Accepted by [illegible]
Ain A. Sonin
Chairman, Departmental Graduate Committee

Optimization of a Self-Tuning Open-Loop Controller

by

Ross Bennett Levinsky

Submitted to the Department of Mechanical Engineering
on January 17, 1995 in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering

Abstract

A class of control problems has been identified in which the control objective is to transition from a measured initial state to a desired final state in a single control action. For systems that have a monotonic relationship between control action and steady-state system output, a self-tuning open-loop controller that divides the input space into non-overlapping "bins," each with an associated control action, can be applied.

This type of control scheme has been implemented on a prototype device, for use in automated teller machines, that straightens banknotes that have become rotated with respect to their normal direction of travel. The length of time that a solenoid fires is the control variable; by partitioning the state space (consisting of note skew), a mapping is created between measured incoming skew and solenoid firing time. As the deskewer corrects more notes and gains experience, the mapping is adjusted and the machine's performance improves.

For certain special cases, an analytical expression is presented from which the number of bins that minimizes average squared output can be found. This number is determined before any tuning occurs, and takes into account both the number of training iterations available and the "learning rate" of the algorithm used to adjust control action. Equations that allow numerical determination of the optimal divisions are given under more general assumptions. Simulation is used to validate the predictions and test the controller on more-general system models.

A technique is next presented that uses training data in progressively-finer regions of the control mapping. In this "bin splitting" technique, a wide region of constant control action is trained for an analytically-determined number of iterations. The region is then divided in two, and new training data that falls into one of the subdivisions no longer alters the control action of the other subdivision. The split regions learn independently, and may themselves split again at a later time. In certain cases, analysis of bin splitting yields a series of partially-coupled, structurally-identical nonlinear equations that are solved sequentially in order to find the number of iterations between splits. These splitting times are found to be independent of the number of training iterations available. Simulation is again used to validate the predictions and apply the controller to more-general systems. For the example system under consideration, bin splitting yields better performance than any static bin structure.

Thesis Supervisor: Dr. Harry West
Title: Senior Lecturer in Mechanical Engineering

To my parents, for gently pushing me this far,

and,

To the memory of Peter Robeck, the man who showed me the self-indulgent nature of boredom.

Acknowledgments

I have spent eight and one-half years at MIT; over one-third of my life as I write this. Along the way, I have met an enormous number of people who have touched my life in ways both large and small. The number of people and the variety of ways in which they've affected me is so large that I can't even list them to myself. Any section of acknowledgments is doomed to incompleteness.

I'd like to be able to write that I'm a profoundly different person now, after my time at MIT, but I'm not sure such an assertion has any meaning. MIT was my formative experience, and the small nugget of pride that motivates me to boast about my "profound change" is nothing but a byproduct of the irrational conviction that I'm somehow better for having been here instead of elsewhere. Who knows? But I can't deny the feeling; at least, I can't deny that my experiences here would have been very, very difficult to replicate elsewhere. My perspectives and views of the possible have broadened in ways that may be unique to MIT. For that, I owe my deepest gratitude to the incredible people I've run across here. Friend or foe, I am what I am because of my encounters with them.

But such a nebulous sentiment will never satisfy those who would like to see a short list of the main players in this particular drama. To wit:

I am once again indebted to Harry West for his continual support and counsel. I would not have done it without him. Thanks are also due to the people of Omron, for beginning the project that led to this thesis, and for their friendship.

Nathan Delson again provided insight at a critical juncture, this time into the solution of various probability problems. I literally could not have finished this thesis without Robert Flory's understanding and friendship.

Thanks are due to the many friends who provided relief from the difficulties of the Ph.D. experience. Whenever I said that I couldn't finish it, someone was there to laugh at me and point out that I probably wasn't the least-capable person ever to get a doctorate. I never believed them, but over time their comments carried enough weight to sustain my efforts.

As always, my deepest appreciation goes to my parents. Although they don't believe it, the only reason I've come this far is to make them proud.

Table of Contents

Abstract 3

Acknowledgments 7

Table of Contents 9

List of Figures 11

Notation 13

1 Introduction 15
- 1.1 Introduction 15
- 1.2 The Deskewer 18
  - 1.2.1 Historical Background 18
  - 1.2.2 Basic Technology of Bill Counting 18
  - 1.2.3 Skew 19
  - 1.2.4 The Deskewer 20
  - 1.2.5 Application of the Controller to the Deskewer 21
- 1.3 Problem Statement 24
  - 1.3.1 Limitations of the First Approach 24
  - 1.3.2 Performance Criterion 25
- 1.4 Example Solution for Fixed Bin Structure 26
- 1.5 Relation to Existing Research 28
- 1.6 Thesis Overview 30

2 The Static-Structure Case 31
- 2.1 Introduction 31
- 2.2 Analysis of the Learning Algorithm 31
  - 2.2.1 Effect of Learning Efficiency 31
  - 2.2.2 Expected Outputs for a Fixed Number of Trials in One Bin 32
  - 2.2.3 Expected Outputs for Multiple Bins 34
- 2.3 Number and Location of Bins 36
  - 2.3.1 Output Variation in a Single Bin 36
  - 2.3.2 Optimal Number of Bins for Arbitrary Input Distribution 38
  - 2.3.3 Optimal Bin Locations for Arbitrary Input Distribution 39

2.4 Solution of the Uniform-Input-Distribution Case 39
2.4.1 Optimal Bin Locations for Uniform Input Distribution 39
2.4.2 Optimal Number of Bins for Uniform Input Distribution 41
2.5 Simulation Results 45

3 Bin Splitting 53
3.1 Introduction 53
3.2 Necessary Results from Chapter 2 55
3.3 Notation 55
3.4 An Equivalent Representation of Output Error 57
3.5 Optimization of the First Split 57
3.6 The Second Split 59
3.7 The General Case 61
3.8 Theoretical Results 62
3.9 Simulation Results 66

4 Conclusions and Future Directions 73
4.1 Conclusions 73
4.2 Extensions of the Static Structure Analysis 74
4.3 Extensions of Bin Splitting 75

Appendix A Example Systems 77

Appendix B The Deskewer 79

Appendix C Analysis of the Deskewer 85
C.1 General Case of Coulomb-Frictional Contact 85
C.2 Deskewer-Specific Equations 86
C.3 Example Solution of Deskewer Equations 89

Appendix D Adjustment of Bin Boundaries 91

Appendix E Tracking Outputs of Individual Bins 95

Appendix F Estimation of Model Parameters 101
F.1 Parameters to be Estimated 101
F.2 Estimation of σ 101
F.3 Estimation of β_1 102

Appendix G Simulation Code 105
G.1 Introduction 105
G.2 Output Error vs. Number of Bins 107
G.3 Output Error vs. Number of Iterations 110

References 117

List of Figures

1.1 General Form of the Control Mapping 16
1.2 Piecewise-Constant Controller 17
1.3 Simplified Sketch of Feed Roller 19
1.4 Skewed Note 20
1.5 Main Elements of the Deskewer 21
1.6 Deskewing Action 22
1.7 Adjustment of Solenoid Firing Time 23

2.1 Output Variation Across Width of Bin 37
2.2 Plot of Upper Bound of $E(S^2)$ vs. b and γ 44
2.3 Contour Plot of Upper Bound of $E(S^2)$ 44
2.4 Contour Plot of Minimum of Upper Bound of $E(S^2)$ 44
2.5 Contour Plot of True Minimum of $E(S^2)$ 45
2.6 S^2 vs. Bin Number - First Model, Uniform Distribution 47
2.7 S^2 vs. Bin Number - Second Model, Uniform Distribution 47
2.8 S^2 vs. Bin Number - Third Model, Uniform Distribution 48
2.9 S^2 vs. Bin Number - Fourth Model, Uniform Distribution 48
2.10 S^2 vs. Bin Number - First Model, Normal Distribution 49
2.11 S^2 vs. Bin Number - Second Model, Normal Distribution 50
2.12 S^2 vs. Bin Number - Third Model, Normal Distribution 50
2.13 S^2 vs. Bin Number - Fourth Model, Normal Distribution 51

3.1 Bin Splitting 53
3.2 Evolution of Error for One and Two-Bin Systems 54
3.3 Changing the Number of Bins in Mid-Training 54
3.4 Error vs. Iterations for 1, 2, 4, 8, 16-Bin Static Systems 63
3.5 Error After 50 Iterations vs. Splitting Iteration (1 to 2 Bins) 63
3.6 Error After 500 Iterations vs. Splitting Iteration (1 to 2 Bins) 64
3.7 Error After 100 Iterations vs. First Two Splitting Iterations 64
3.8 Error vs. Iterations - Optimal Splitting and Static Structures 65
3.9 Ratio of Splitting Error to Static Error 66
3.10 S^2 vs. Iteration Number - First Model, Uniform Distribution 67
3.11 S^2 vs. Iteration Number - Second Model, Uniform Distribution 67

3.12 S^2 vs. Iteration Number - Third Model, Uniform Distribution 68
3.13 S^2 vs. Iteration Number - Fourth Model, Uniform Distribution 68
3.14 S^2 vs. Iteration Number - First Model, Normal Dist., Equal-Width Bins 69
3.15 S^2 vs. Iteration Number - Second Model, Normal Dist., Equal-Width Bins 69
3.16 S^2 vs. Iteration Number - Third Model, Normal Dist., Equal-Width Bins 70
3.17 S^2 vs. Iteration Number - Fourth Model, Normal Dist., Equal-Width Bins 70
3.18 S^2 vs. Iteration Number - First Model, Normal Dist., Equiprobable Bins........ 71
3.19 S^2 vs. Iteration Number - Second Model, Normal Dist., Equiprobable Bins..... 71
3.20 S^2 vs. Iteration Number - Third Model, Normal Dist., Equiprobable Bins....... 72
3.21 S^2 vs. Iteration Number - Fourth Model, Normal Dist., Equiprobable Bins 72

B.1 Test Fixture and Deskewer Closeup 80
B.2 Main Structure of Deskewer 81
B.3 Sensors and Solenoids 82
B.4 Side Plate of Deskewer 83

C.1 Small Plate Element Translating on Moving Surface 86
C.2 Direction of Frictional-Force Vector 87
C.3 Plan View of Deskewing System 88
C.4 Simulation Results for Varying Pin Position *P* 89
C.5 Simulation Results for Varying Solenoid Position *S* 90

D.1 Adjusting Bin Boundaries when $\mu \neq 0$ 93

E.1 Initial Bin - To Be Split at *b* 95

Notation

$\overline{x}$	Overbar denotes the average output of a stochastic process
a_i	Location of bin boundary i
β_1	Slope of linear function relating input and output (at a fixed τ); equals $\frac{\partial \overline{\theta_{out}}}{\partial \theta_{in}}$
$E(.)$	Expectation operator; expected value of term in parentheses
$E_{a,b,c}$	Special notation for $E(\mu^2)$. See Section 3.3.
ϕ	Defined as $\left(1 - \frac{\lambda}{\gamma}\right)^2$
ϕ_0	Defined as ϕ. See Section 3.3.
ϕ_i	Defined as $\phi_i = \left(1 - P_i + P_i\phi_0\right)$. See Section 3.3.
γ	Learning rate, used in updating control action τ. Larger values give smaller update.
λ	Lower bound on $\left\lvert \frac{\partial \mu_i}{\partial \tau} \right\rvert$
μ	Average output; when used with a subscript, average output for the region of input space in bin i
N	Total number of training notes available to the whole system (all bins)
n	General iteration counter
n_i	Iteration at which the i-th split occurs. See Section 3.3.
v	Upper bound on $\left\lvert \frac{\partial \mu_i}{\partial \tau} \right\rvert$
P_i	Probability of an input falling into a bin in the i-th generation. See Section 3.3.
$p(.)$	Probability density function of term in parentheses
p_i	Probability of i notes falling into a bin. See Section 2.2.3.
S^2	Average squared output of system; when used with a subscript, average squared output for the region of input space in bin i
σ^2	Variance of the inherent noise in the system, measured at a fixed input and fixed control action
σ^2_{bin}	Variance of the output for the region of input space covered by a particular bin
$\sigma^2_{bin,\,i}$	i tracks σ^2_{bin} in bin-splitting analysis. See Section 3.3.
τ	Control action associated with a single bin; when used with a subscript, refers to control action in a particular bin (except in Sections 1.4 and 2.2)
θ_{in}	System input
θ_{out}	System output
w	Width of a single bin
W	Width of the entire range of input space

Chapter 1
Introduction

1.1 Introduction

This thesis deals with a class of control problems in which the control objective is to use a single application of constant control action (an open loop controller) to drive a system from a given initial state to a desired constant final state. The systems considered have two fundamental characteristics: the relationship between control action and steady-state system output is monotonic, in a sense described below, and the system output is stable in the absence of an applied control action. While the systems are generally difficult to model precisely, their monotonicity serves as a simple heuristic relating control action to final state, and allows the application of a self-tuning open-loop controller that operates without a traditional model.

A simple illustration involves a variant of the childhood game of "marbles." Suppose we wish to control a device that, given the distance of a marble from some target location, strikes the marble with sufficient impulse for it to stop exactly on the target. The desired controller is a mapping from input space (initial marble position) to control output (applied impulse), with a qualitative form shown in Figure 1.1. Once the marble has been struck there is no further control of its trajectory; the controller is open-loop.

When the device applies a nominally-constant impulse to different marbles, all starting at that same distance from the target, we assume that there is some random variation in the marbles' final resting positions. Although the marbles are outwardly identical, some overshoot the target, while others undershoot. This variable error may be due to small differences in their weight, size, or smoothness, or to fluctuations in the amount of impulse the device applies. It does not occur because the marbles hit each other; we can remove each one from the playing field after it has stopped rolling, and before its successor has been struck by the device.

The random variation in final position suggests that without detailed characterization of each marble, there is no possible control mapping that will cause every marble to stop exactly on target. To fairly evaluate the success of our controller, which we will assume does not have detailed information about each marble (other than initial distance from the target), some type of average measure of its performance is required. In this example, it is reasonable to choose the success criterion to be the expected value of the square of the distance to the target after the marble has ceased rolling. The use of squared distance penalizes the controller equally for over- and undershooting.

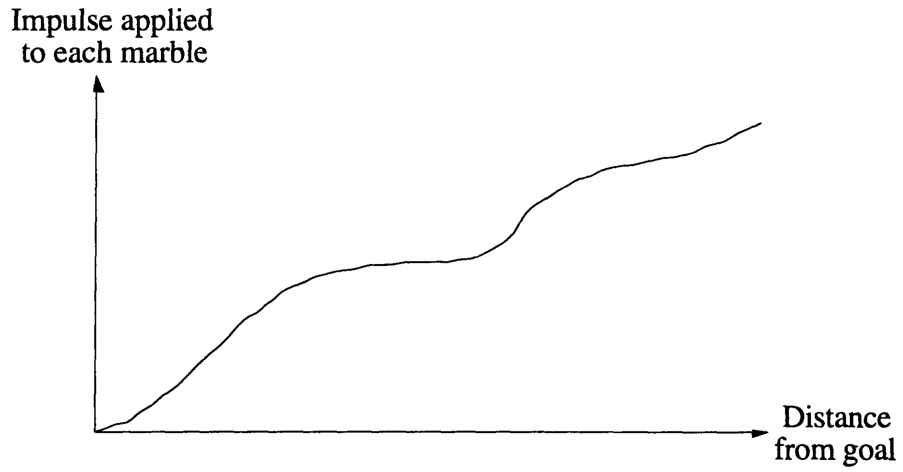


Figure 1.1: General Form of the Control Mapping

A proposed method of implementing an open-loop controller for the device is as follows. The input space (here, the distance from the marble to the target) is divided into a series of ranges, or "bins," as depicted in Figure 1.2. Within each bin is an associated single control action (the impulse). Before a marble is hit, its initial distance is measured, and the impulse to be applied by the device is taken from the bin into which the distance falls.

The control action within a bin is adjusted, typically during a separate training period, with an algorithm that uses observations of the results of previous applications of trial control values associated with that bin. These control adjustments - the "self-tuning" - serve to reduce output error within each bin.

For systems of the type considered in this work (see the next section and Appendix A), we can use an adjustment algorithm that requires no system model beyond knowledge of the sign of, and bounds on the magnitude of, the derivative of system output with respect to control action. The sign information (assumed to be always positive or negative) allows adjustment, in the proper direction, of control action, while the derivative bounds are used to prevent divergence of the chosen algorithm. Other learning methods, not considered here, may not require the derivative bounds.

An advantage of the proposed controller is that as the bins' prescribed actions are tuned, there is maximal "locality" to the change in the overall mapping. Only states that fall into a given bin are affected by the control action update. This is in contrast to many neural network schemes, in which the changes to the connection weights from a single training pair affect the entire input-output mapping.

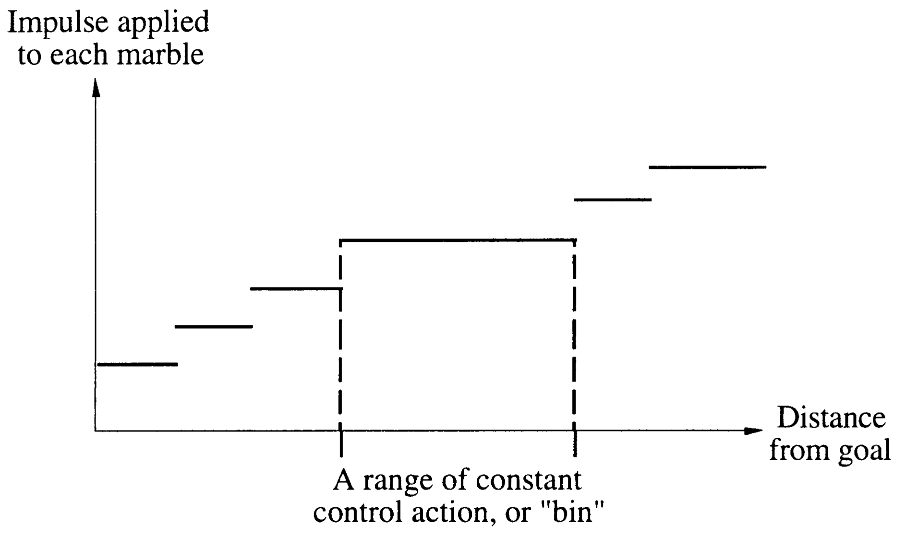

Impulse applied to each marble
Distance from goal
A range of constant control action, or "bin"

Figure 1.2: Piecewise-Constant Controller

A second advantage is that the control laws can be updated throughout operation, which allows the controller to compensate for slowly-varying changes in plant performance. This is not considered in Chapters 2 and 3; instead, training from a state of no knowledge (initial control action of zero) is used as a convenient benchmark of system performance. Sections 4.2 and 4.3 suggest methods by which the analyses presented in this work can be adapted to find the best controller structure when the main concern is updating an already-trained controller.

Finally, the proposed controller is conceptually simple, and easy to implement in hardware.

One disadvantage of the controller is that the number of bins increases geometrically with the number of input variables and arithmetically with the division of each input variable's range. A second is that the control-action-adjustment method is not clearly defined in situations where there is more than one control action; an additional heuristic is needed to decide which of the multiple actions should be adjusted, given a particular system output. Only SISO systems are considered here.

This thesis addresses the three design variables of this controller: the number of bins, the placement of bin boundaries, and the method of adjusting control action within each bin. In the basic formulation given in Chapter 2, before any tuning occurs, the number of input ranges is determined by the amount of desired or available training information, and other system parameters. This approach, with its fixed number of bins, is referred to in this thesis as a "static structure." After initial training is complete, the bins can be "split," as discussed in Chapter 3, or the boundaries of each bin can be adjusted so as to minimize error caused by suboptimal division of the input space (Appendix D).

1.2 The Deskewer

A practical example of the type of system described above, and the motivation for much of the work presented here, is a bill-deskewing device for automated teller machines (ATM). This is a machine, designed and constructed at MIT by the author and several others ([Kotovsky 90] and [Levinsky 1992]), that straightens banknotes that have become rotated with respect to their normal direction of travel in an ATM. The deskewer project is briefly recounted below.

1.2.1 Historical Background

Since 1969, the Omron Corporation of Japan has designed and produced various types of cash-handling equipment, from change dispensers to ATM. The machines have been experimentally tuned to a very high level of sophistication over time; by the early 1990's, Omron's most advanced model, the HX-ATM, reliably operated at a transfer rate of over ten notes per second.

During the late 1980's engineers at Omron became convinced that they needed to expand their analytical knowledge of the note-handling process in order to increase transfer speeds and improve reliability. To reduce the need for expensive human service, they also wanted to incorporate self-adjusting mechanisms and mechanical error-correction systems into their designs.[1] With those goals in mind they entered into a research project with the Mechatronics Design Lab of MIT.

1.2.2 Basic Technology of Bill Counting

The bill-handling technology used by Omron is similar to that employed by other major Japanese manufacturers of such products (Hitachi, Fujitsu, Oki, and Toshiba). In contrast to ATM in America, where cash to be deposited is first placed in an envelope, ATM in Japan accept direct input of bills from the customer. The notes are slid from the top of the deposit stack by a rotating feed roller (Figure 1.3) and carried through the machine by rubber transport belts that "sandwich" the notes. Through a series of belts and sensors the bills are validated, counted, and stacked in cartridges during a deposit cycle, and fed, again using rotating-feed-roller technology, out of the cartridges during cash withdrawal. While the machines examined during this project were built for the Japanese market (and were thus sized to handle yen), Omron uses near-identical technology to produce cash-handling equipment for other Asian countries.

[1]Information about Japan's ATM market, Omron's machines, and Omron's need for analytical models came from a series of discussions with Omron personnel over a two-year period. These people included, but were not limited to, Hiroyuki Nishimura, Ryuichi Onomoto, Ichiro Kubo, and Yoshimasa Sugitate.

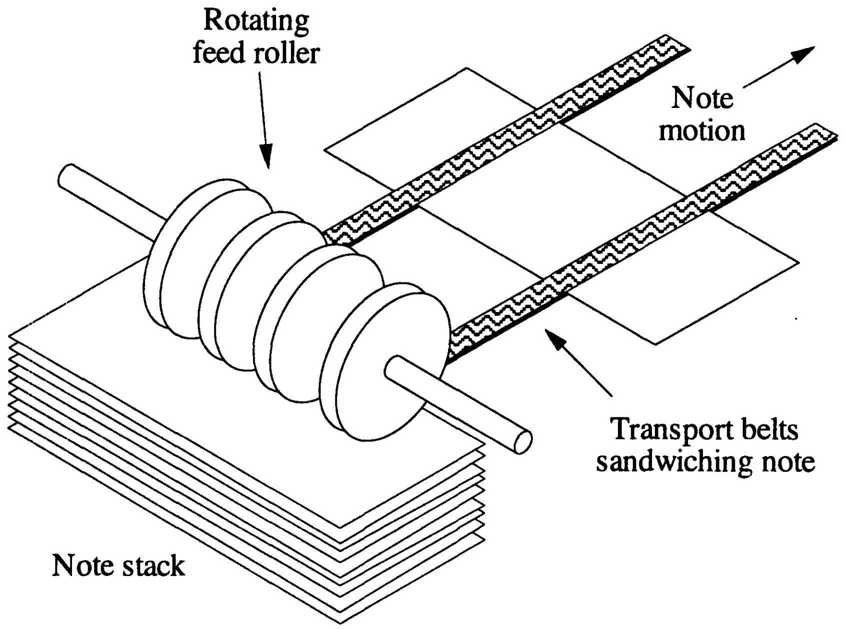


Figure 1.3: Simplified Sketch of Feed Roller

Jamming is the principal failure mode in Omron's machines. Typically, a note becomes stuck at a fork in the travel path while the notes behind it continue to move, causing one or more bills to crumple before the machine's optical sensors detect the problem and stop the transport mechanism. The exact causes of failure are difficult to pinpoint because the failure rate is so low - the only statistics Omron has are taken from in-service operation. It is has not been possible to observe enough failures in a controlled factory environment to adequately determine all causes of jamming.

1.2.3 Skew

Omron does possess enough statistical information about jamming, however, to indicate that a phenomenon known as "skew" is the most common cause of jamming incidents. It was therefore decided that the initial approach to improved machine reliability would be to correct skewed notes in transit. Skew (Figure 1.4) is defined as a note's angular deviation from the correct travel position in which its short side is parallel to the direction of belt travel. The spacing between a skewed bill and its two surrounding notes (leading and trailing) is incorrect; this can cause jamming at forks in the travel path.

Early research in the project found that skew occurs at the very beginning of the feeding process, when the note is accelerated by the slipping frictional contact of the feed roller. If a note enters the roller with one corner leading (slightly skewed), the frictional forces are initially applied at that corner, causing a torque that rotates the bill and increases its skew angle. This effect has been predicted by numerical simulation (see [Levinsky 1992]) and confirmed with high-speed video. In addition, if the feed roller does not exert even pressure at all points along its axis, the note skews. This occurs because the frictional feeding force on a given area of the note's surface is proportional to the pressure that the roller applies on that area; when the pressure is not uniformly distributed along the roller's axis, a net moment is exerted on the note. This phenomena is exploited during machine maintenance when technicians perform feed-roller pressure adjustments based on a machine-accumulated record of each deposited note's skew as it exited the feeder. An asymmetrical distribution of skew indicates that the roller pressure is uneven.

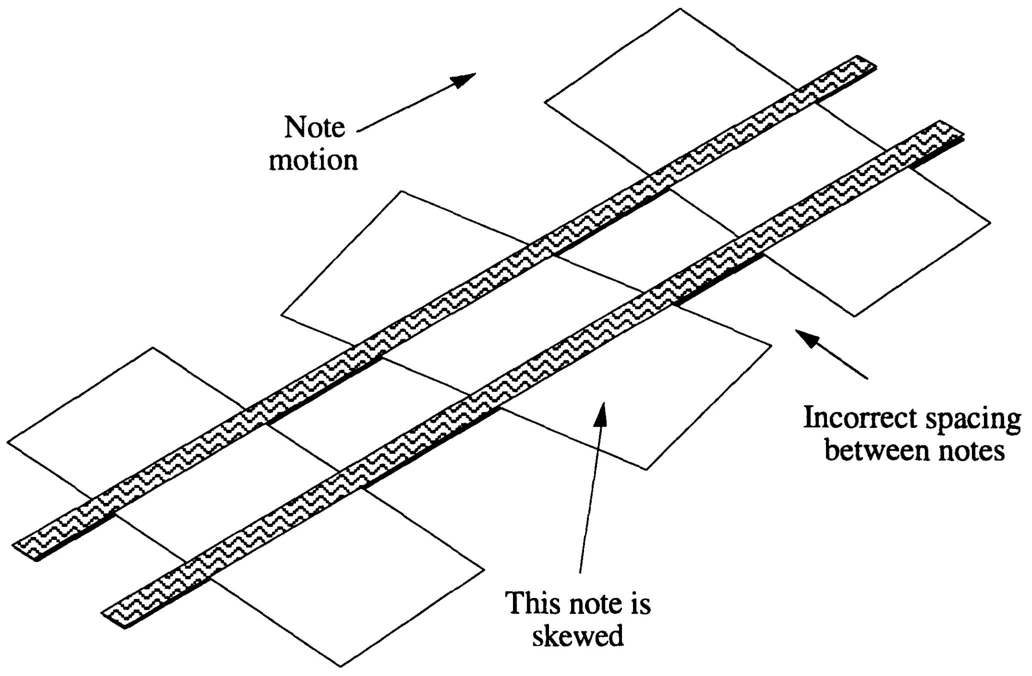
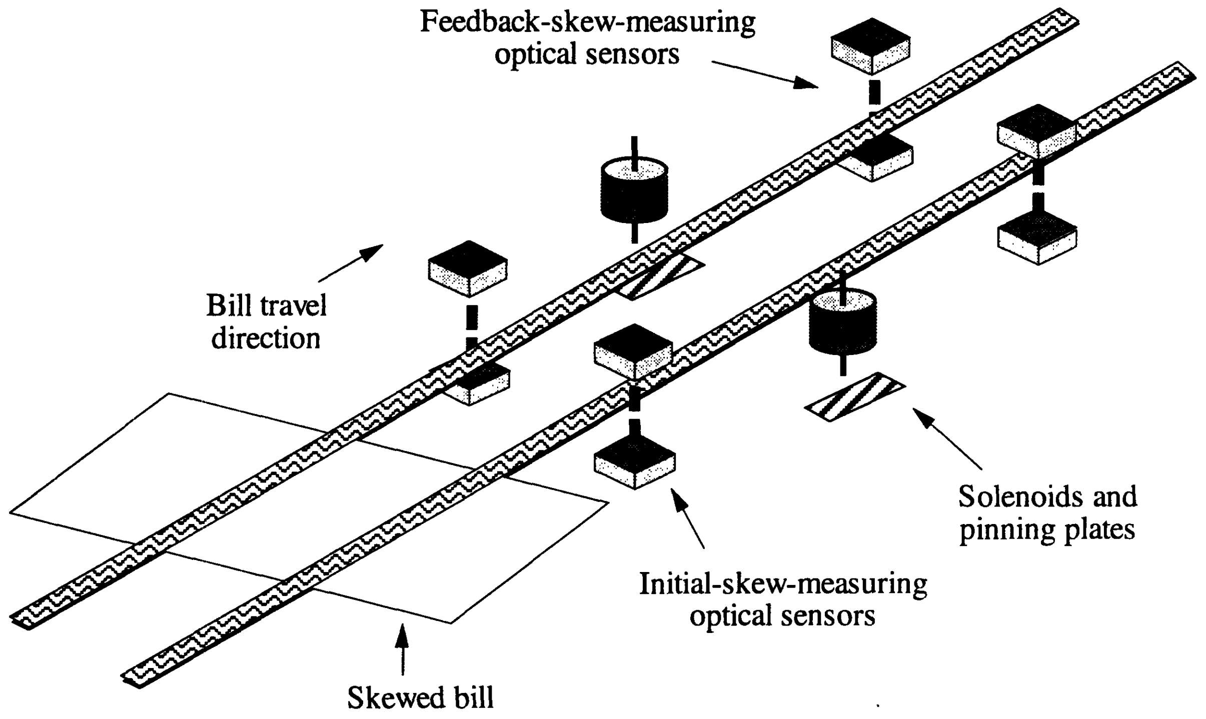


Figure 1.4: Skewed Note

1.2.4 The Deskewer

A conceptual sketch of the deskewer prototype is shown in Figure 1.5. Detailed drawings are presented in Appendix B, while additional information on design alternatives and construction has been documented elsewhere [Kotovsky 1989].

The deskewer uses optical sensors mounted perpendicularly to, and out of the plane of, the note travel path to detect the angle of skew. The note's skew is determined by measuring the time difference between obscuration of the left and right initial-skew optical sensor pairs. This angle is then tested in the antecedents of a bank of if-then rules, and a control action is selected. This is τ, the firing time to be applied by one of the two solenoids mounted perpendicular to the notes' path. Once the skewed bill has passed under the actuators, the solenoid on the leading side fires downward, pinning a small area of the bill against aluminum plates located underneath the travel path (see Figure 1.6). The pinned area serves as a pivot point while the motion of the belts rotates the bill in a direction opposite that of the initial skew. If the control action has been properly selected, the solenoid retracts just as the bill has been completely straightened; if not, the bill leaves with a residual skew. This error is then measured by a second set of feedback optical sensors, and the output skew value is used to immediately adjust the control action of the rule which was initially selected. Depending on note velocity and skew, solenoid firing time ranges from approximately 10 to 50 milliseconds.



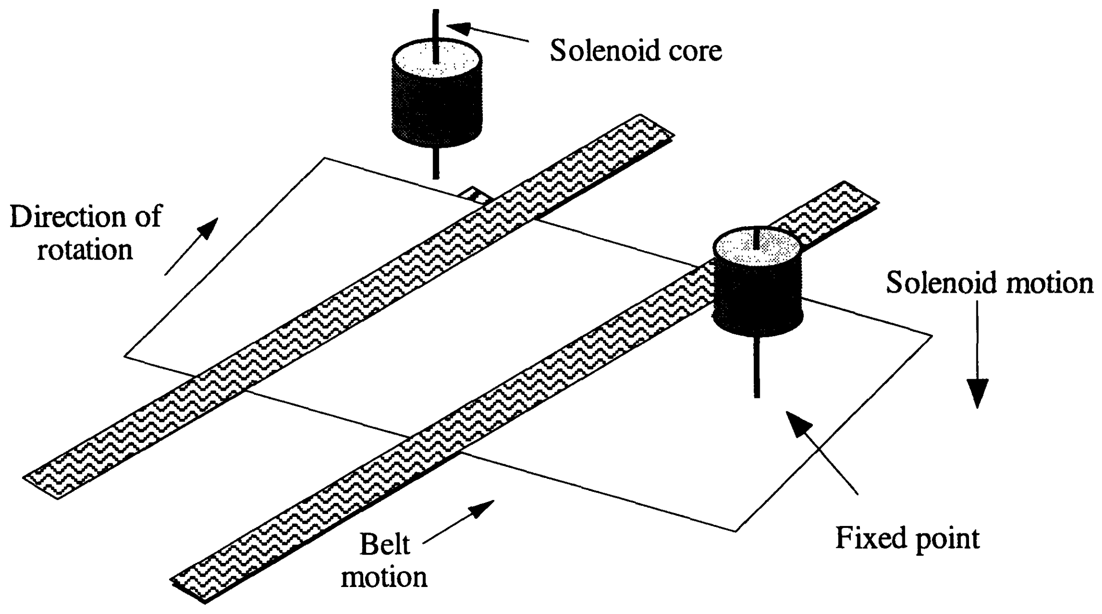
Feedback-skew-measuring optical sensors
Bill travel direction
Solenoids and pinning plates
Initial-skew-measuring optical sensors
Skewed bill

Figure 1.5: Main Elements of the Deskewer

1.2.5 Application of the Controller to the Deskewer

The detailed modeling of frictional-contact note-handling machines presents four fundamental difficulties. First, the Coulomb friction model (frictional force is proportional to normal force) is empirically derived and does not model the stick-slip behavior that can occur when a portion of the bill touches a surface such as the feed roller or the transport belts.

The second problem is the extreme variability in note quality encountered during actual operation. Notes are nonisotropic because they have usually been folded; at a crease they bend more easily in some directions than in others. They also show uneven surface wear, with accompanying variability in coefficient of friction.



Right-side solenoid fires and holds bill to pinning plate.
Belts on both sides continue to move, dragging bill about axis of right-side solenoid.

Figure 1.6: Deskewing Action

Third, machine performance can vary according to such factors as humidity, which alters the note's mechanical properties, and belt wear, which affects the coefficient of friction between belt and note.

Fourth, even with a considerably simplified system model, the resulting note-motion equations involve complex multidimensional integrals with time-varying limits of integration, as shown in Appendix C. These equations are usually solved numerically, and are thus principally used to examine the qualitative effects of the variation of design parameters on machine performance.

While a controller that effects closed-loop control on each note can compensate for note wear, machine wear, and humidity variation, there is currently no inexpensive sensor[2] for continuously measuring skew during the deskewing process. In contrast, control hardware for the open-loop controller is straightforward, consisting of timers and low-cost optical sensor pairs that measure the incoming and outgoing note skew, and a timer that holds the solenoid firing time. The drawback of a traditional open-loop controller is that its fixed mapping between incoming-note skew and solenoid firing time does not account for the changes in deskewing times due to environmental shifts, note variation, and machine wear.

[2]The margins on ATM are very slim. According to Omron, a sufficiently-affordable deskewing sensor must cost less than approximately $5 (1990 dollars).

In an attempt to combine the simplicity of open-loop control with the robustness of closed-loop, a controller identical in form to the one introduced in Section 1.1 was applied to the deskewer. Feedback is not effected on any single note while it is deskewing, but the measured outgoing skew is used to alter the control mapping and thus change the solenoid firing timing for all future notes. The system is "locally open, globally closed." The mapping evolves over time, and corrects for variations in deskewer performance.

The clear relationship between firing time τ and output θ_{out} allows the adjustment algorithm to proceed with minimal information. If the note is undercorrected, we have reason to believe that the firing time for the bin needs to be increased, while overcorrection suggests a decrease. In short, $\frac{d\theta_{out}}{d\tau}$ is negative.

Figure 1.7 shows the process. A note enters the machine; a skew time is selected; the note is corrected; the note's output skew is measured; based on the observed over-correction, the solenoid time associated with the input bin is reduced.

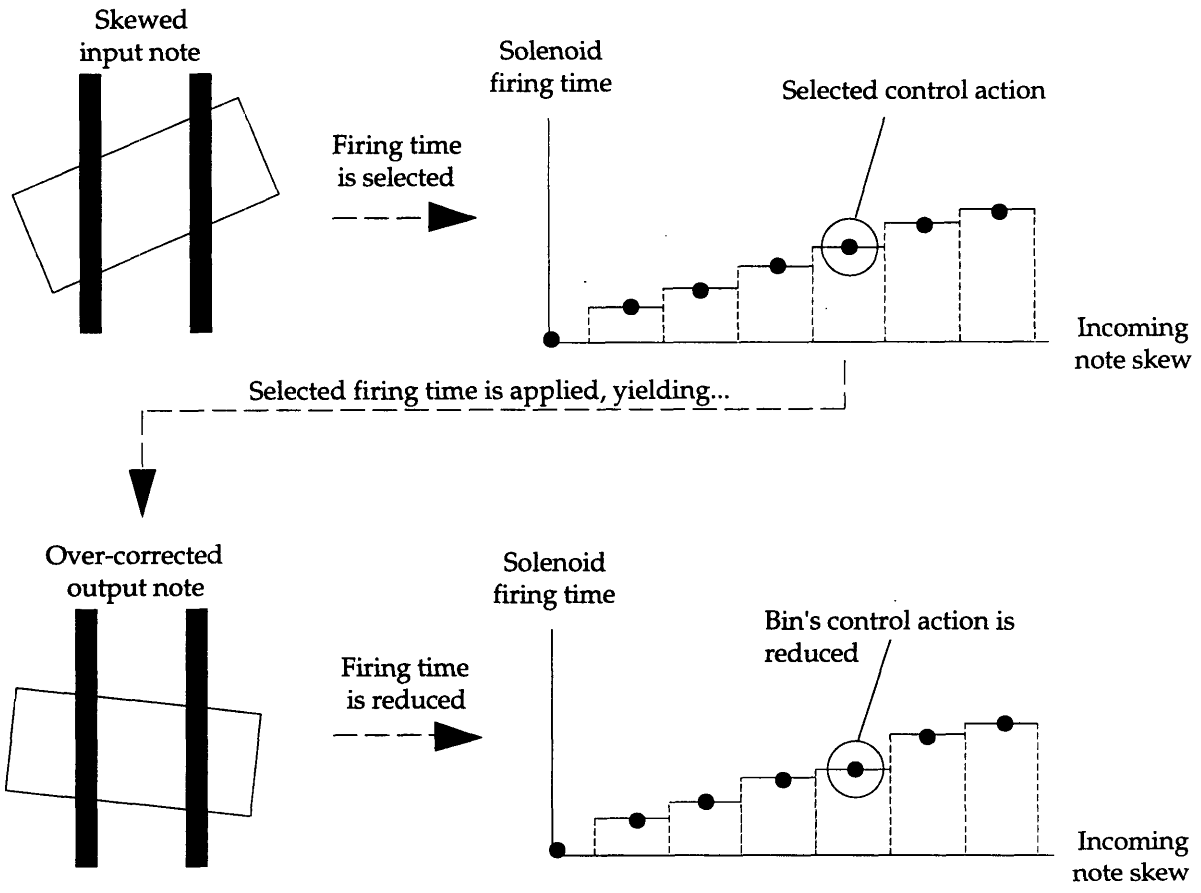


Figure 1.7: Adjustment of Solenoid Firing Time

1.3 Problem Statement

1.3.1 Limitations of the First Approach

A binned controller was first applied to the deskewer, analyzed, and experimentally found to give adequate performance [Levinsky 92]. The analysis assumed perfect behavior of the deskewer; a one-to-one relationship $\theta_{out} = F(\theta_{in}, \tau)$ in which there is no output variation for notes of identical input skew and applied firing time. Given this idealization, the control action within each bin was shown to converge to within a known range.

However, the work did not address two key aspects of the system's behavior. First, the finite range of each bin dictates that there can be no single firing time that yields a uniform output of 0° for all notes affected by that bin. The control-action-tuning algorithm can at best adjust the control value of each bin to yield an expected output skew of 0°.

Second, even for incoming notes of a single skew, and at a fixed firing time, the notes' outgoing skew will have some random error; the strict one-to-one relationship $\theta_{out} = F(\theta_{in}, \tau)$ does not hold. The output of the deskewer is best described as a joint probability distribution of the form $p_{\tau}(\theta_{in}, \theta_{out})$, where the subscript τ indicates that the distribution is a function of the solenoid firing time. The presence of such a distribution is due to the varying mechanical qualities of the notes, such as stiffness and surface friction, as well as the finite bin width. Since the deskewer's belts wear and the quality of incoming notes can vary, the distribution may also be nonstationary. For simplicity, we assume stationary distributions in this thesis.

Because of the nonzero bin width and the inherent variability of output skew (even with a fixed θ_{in} and τ), the deskewer will always yield nonzero output skews. The goal must be to optimize its behavior with respect to some broader measure of error.

If training time is unimportant and all probability distributions are stationary, the optimum piecewise-constant controller will tend towards an infinite number of bins. The number of bins must be limited, however, if the number of training samples is not infinite, as is the case in most practical problems. In a real ATM, the deskewer might only have several hundred notes available for self-tuning. If there are thousands of bins, most will receive no adjustment.

There is a tradeoff between bin width and training time. With many narrow bins, the *potential* performance of the machine improves. But, there will be fewer training samples per bin, which can increase the average error (measured with the control action fixed for all time post-training). With a few wide bins, there are more training samples per bin, but the bin width itself degrades performance.

The goal of this thesis is to find the optimal controller for a fixed number of training notes. The adjustable variables are the number of bins, the boundaries of each bin, and the control action associated with each bin. While the general solution of this problem is presently unknown, an idealized model that captures many important features of real system performance is created in Chapter 2. A rigorous analysis of the optimal formulation of the piecewise-constant controller can be applied to this model; controllers designed with this analysis are then evaluated, via simulation, on more-complex systems that do not obey the model's simplified assumptions.

1.3.2 Performance Criterion

Generally, we will minimize the expected value of average square of system deviation from a desired setpoint, with the evaluation measurements taken after all training is complete. No weight is given to error incurred during the training period. Because the motivating problem is the deskewer, in which we are trying to minimize skew, the cost function is here the average squared output skew of notes exiting the deskewer:

$$E(\theta_{out}^2) = \int_{(minimum\ \theta_{in})}^{(maximum\ \theta_{in})} \int_{-\infty}^{\infty} \theta_{out}^2\, p_\tau(\theta_{in}, \theta_{out}) d\theta_{out}\, d\theta_{in} \qquad (1.3\text{-}1)$$

This expression can be rewritten using the product of the conditional probability distribution $p_\tau(\theta_{out}|\theta_{in})$ and the distribution of θ_{in}, $p(\theta_{in})$.

$$E(\theta_{out}^2) = \int_{(minimum\ \theta_{in})}^{(maximum\ \theta_{in})} \int_{-\infty}^{\infty} \theta_{out}^2\, p_\tau(\theta_{out}|\theta_{in})\, p(\theta_{in})\, d\theta_{out}\, d\theta_{in} \qquad (1.3\text{-}2)$$

The subscript τ attached to $p_\tau(\theta_{out}, \theta_{in})$ and $p_\tau(\theta_{out}|\theta_{in})$ indicates that the distribution is altered by adjusting the firing time associated with each bin. We can divide (1.3-2) into a sum of integrals, each of which represents a single bin. If a_i represents the bin boundaries and b is the total number of bins:

$$E(\theta_{out}^2) = \sum_{i=1}^{b} \left(\int_{a_{i-1}}^{a_i} \int_{-\infty}^{\infty} \theta_{out}^2\, p_{\tau_i}(\theta_{out}|\theta_{in})\, p(\theta_{in})\, d\theta_{out}\, d\theta_{in} \right) \qquad (1.3\text{-}3)$$

This can be rewritten by integrating $\left(\theta_{out}^2\, p_{\tau_i}(\theta_{out}|\theta_{in})\right)$ across the range of θ_{out}, which gives us $\overline{\theta_{out}^2}(\theta_{in}, \tau_i)$, the average θ_{out}^2 at a given θ_{in} and τ_i. Replacing the inner integral in (1.3-3):

$$E(\theta_{out}^2) = \sum_{i=1}^{b} \left(\int_{a_{i-1}}^{a_i} \overline{\theta_{out}^2}(\theta_{in}, \tau_i)\, p(\theta_{in})\, d\theta_{in} \right) \qquad (1.3\text{-}4)$$

The minimization of (1.3-4) involves the selection of the total number of bins, b; placement of the b - 1 bin boundaries, a_i; and selection of the b firing times τ_i. Once b has been chosen, we are left with an optimization problem in $b + (b - 1) = 2b - 1$ variables. Generally, none of the probability distributions are known *a priori*; they are estimated through experimental observation.

1.4 Example Solution for Fixed Bin Structure

Without detailed knowledge of the behavior of the system to be controlled, the general optimization of (1.3-4) is impossible in closed-form (for arbitrary $\overline{\theta^2_{out}}(\theta_{in}, \tau_i)$ and $p(\theta_{in})$). The approach of this thesis is to formulate simplified models based on important elements of the general behavior of real-world systems, and to then optimize the controller structure based on these models. This "optimum" is then applied to the real system with the expectation that its performance will prove better than that of a structure designed on an *ad hoc* basis. This expectation is realized in Chapters 2 and 3. When the "optimal" controller is applied to simulated systems in which the simplifying assumptions are relaxed, good qualitative agreement is observed between the theoretical predictions and the simulated results.

As a simple example of the solution techniques used, let us assume a completely linear model of deskewer performance, with an added stochastic term. This is an analytic realization of three plausible heuristic models. First, for a fixed control action, notes that enter the deskewer with relatively greater skew will tend to leave with greater skew. Second, for a fixed incoming skew, longer solenoid firing times will tend to reduce outgoing skew. Third, there is some inherent variation in the note quality which leads to a concomitant variation in output.

Input skew is here assumed to be uniformly distributed. θ_{in} is the incoming skew of a note, θ_{out} is the outgoing skew after exiting the deskewer, τ is the solenoid firing time, β_1 and C are constants, and r is an independent (uncorrelated) noise term of zero mean and constant variance σ^2:

$$\theta_{out}(\theta_{in}, \tau) = \beta_1 \theta_{in} + C\tau + r \qquad (1.4\text{-}1)$$

Given that μ^2 is the squared average output value of the bin, it can be shown from equation (2.3-7) that $E(S^2)$, the expected average squared output (measured in degrees^2, in the case of the deskewer) of a bin of width w is:

$$E(S^2) = \frac{\beta_1^2 w^2}{12} + E(\mu^2_{bin\ width\ w}) + \sigma^2 \qquad (1.4\text{-}2)$$

If we divide the single bin into two sub-bins of width $w/2$, the output is then:

$$\begin{aligned} E(S^2) &= \frac{1}{2}\left(\frac{\beta_1^2}{12}\left(\frac{w}{2}\right)^2 + E\left(\mu^2_{\substack{\text{bin width } w/2,\\ \text{left bin}}}\right) + \sigma^2\right) + \frac{1}{2}\left(\frac{\beta_1^2}{12}\left(\frac{w}{2}\right)^2 + E\left(\mu^2_{\substack{\text{bin width } w/2,\\ \text{left bin}}}\right) + \sigma^2\right) \\ &= \frac{\beta_1^2 w^2}{48} + \sigma^2 + \frac{1}{2}\left(E\left(\mu^2_{\substack{\text{bin width } w/2,\\ \text{left bin}}}\right) + E\left(\mu_{\substack{\text{bin width } w/2,\\ \text{left bin}}}\right)\right) \qquad (1.4\text{-}3) \end{aligned}$$

If (1.4-3) is smaller than (1.4-2), we should use two bins. This decision hinges on the sizes of $E\left(\mu^2_{\text{bin width } w}\right)$ and $\frac{1}{2}\left(E\left(\mu^2_{\substack{\text{bin width } w/2,\\ \text{left bin}}}\right) + E\left(\mu^2_{\substack{\text{bin width } w/2,\\ \text{right bin}}}\right)\right)$ at the end of the training period. The value of $E(\mu^2)$ in a bin is a function of both the learning technique used and the number of training notes which fall into that bin. In Chapter 2, the analysis of the static bin structure allows the use of any learning method in which the evolution of $E(\mu^2)$, as a function of training iterations, can be expressed analytically.

The adjustment method examined in this thesis is a standard gradient descent technique that was applied to the deskewer in [Levinsky 1992]. In the remainder of this section, the subscript n refers to the values of the subscripted variables at specific training iterations. Each time an output θ_{out_n} is observed, the following update rule is applied:

$$\tau_{n+1} = \tau_n + \frac{\theta_{out_n}}{\gamma} \tag{1.4-4}$$

where γ is the learning constant. Defining $\phi = \left(1 - \frac{C}{\gamma}\right)^2$ for clarity, it can be shown that:

$$E(\mu_n^2) = \phi^n E(\mu_0^2) + \frac{(1-\phi^n)}{(1-\phi)}\left(\frac{C}{\gamma}\right)^2\left(\frac{\beta_1^2 w^2}{12} + \sigma^2\right) \tag{1.4-5}$$

For convergence of (1.4-5), we require $\gamma > \frac{C}{2}$. A more general analysis proceeds from the development of Gauss-Markov processes presented in [Gardner 1986], and is given in Section 2.2.

For a given bin trained from a state of no control knowledge (zero control action), $E(\mu_0^2)$, the value of $E(\mu^2)$ before any training occurs, is completely deterministic and can be shown to equal the square of the average value of θ_{out} arising from a θ_{in} that is halfway between the bin boundaries. For a single bin, total input space width of W, and N training notes, substituting (1.4-5) into (1.4-2) gives:

$$E(S^2) = \phi^N \frac{\beta_1^2 W^2}{4} + \left(\frac{(1-\phi^N)}{(1-\phi)}\left(\frac{C}{\gamma}\right)^2 + 1\right)\left(\frac{\beta_1^2 W^2}{12} + \sigma^2\right) \tag{1.4-6}$$

This is the expected average squared output as a function of training time for one bin. When we move to multiple bins, the determination of $E(\mu^2)$ is complicated by the fact that the number of notes falling into a single bin obeys a binomial distribution. This is addressed in the next chapter, but for simplicity we here assume that an even number of notes is guaranteed to fall into each bin. If the single bin is split into two equally-sized bins, the individual bins' error components are now weighted by bin width and summed:

$$\begin{aligned} E\left(S^2\right) &= \frac{1}{2}\left(\phi^{\frac{N}{2}}\beta_1^2\left(\frac{W}{4}\right)^2 + \left(\frac{\left(1-\phi^{\frac{N}{2}}\right)}{(1-\phi)}\left(\frac{C}{\gamma}\right)^2 + 1\right)\left(\frac{\beta_1^2}{12}\left(\frac{W}{2}\right)^2 + \sigma^2\right)\right) + \frac{1}{2}\left(\phi^{\frac{N}{2}}\beta_1^2\left(\frac{3W}{4}\right)^2 + \left(\frac{\left(1-\phi^{\frac{N}{2}}\right)}{(1-\phi)}\left(\frac{C}{\gamma}\right)^2 + 1\right)\left(\frac{\beta_1^2}{12}\left(\frac{W}{2}\right)^2 + \sigma^2\right)\right) \\ &= \phi^{\frac{N}{2}}\frac{5}{16}\beta_1^2 W^2 + \left(\frac{\left(1-\phi^{\frac{N}{2}}\right)}{(1-\phi)}\left(\frac{C}{\gamma}\right)^2 + 1\right)\left(\frac{\beta_1^2 W^2}{48} + \sigma^2\right) \qquad (1.4\text{-}7) \end{aligned}$$

If (1.4-7) yields a smaller $E\left(S^2\right)$ than (1.4-6), two bins should be used. Similar arguments are used to extend this result to an arbitrary number of bins. Relaxation of the above simplifications adds algebraic complexity, but the analytical technique is identical.

1.5 Relation to Existing Research

The piecewise-constant controller can be broadly classified as a nonparametric, unsupervised, reinforcement-trained direct controller. The principal value of this work, however, is not found in the controller structure; rather, for what appears to be the first time, it presents an analytical procedure for optimizing the division of state space for a simple model of a real control problem. While much of the literature reviewed presents techniques for classifying or dividing regions, none has presented a method based on the same type of analysis set forth in this work.

The problem of state-space division is common to much of modern "intelligent" control, pattern recognition/classification, and function approximation. In these three areas, neural and fuzzy techniques appear in many of the reviewed papers, while genetic algorithms, simulated annealing/stochastic reinforcement, and gradient descent are frequently used to optimize controller or classifier structure.

Control - One of the first applications of binning is found in [Michie 1968], in which a cart/pole system is controlled by dividing up the position and velocity inputs into "boxes," each containing a bivalued control action. A critic algorithm then adjusts each boxes' control and stabilizes the system. The overall algorithm came to be known as BOXES. Extensions are found in [Bain 1990], in which a two-pole system is controlled, and [Gullapalli 1994], in which the box boundaries are arbitrarily chosen, and then trained with a reinforcement algorithm. Genetic algorithms are used twice in [Varsek 1993] in the control of a cart/pole system. The first application finds control rules and boundaries of input-space bins. Following a sharp compression of the rule set, a second-stage genetic algorithm is used to retune the reduced set.

The automated selection of membership functions for fuzzy-logic-based controllers can be considered a division of input space. Early work includes [Procyk 1977], in which an invertible incremental system model is used to adjust the membership functions, while the breadth of recent work is represented in [Berenji 1992], [Isaka 1992], [Jang 1992], [Wang 1992], [Nie 1993], and [Yager 1993].

A field generally known as "memory-based" control is represented by the work of Atkeson ([Atkeson 1989], [Schall 1994]) and Kawamura ([Kawamura 1990]). Typically, values of control action and system-state vectors are stored together in a large computer memory; to plan a desired trajectory through state space, the nearest available state-space data points are found and their stored control actions are applied, sometimes in parallel with a PID controller. The main connection to this work lies in the nonparametric nature of the controller.

Other work with similarities to that presented here includes [Mikami 1992], in which shop scheduling is learned by stochastic automata trained with a reinforcement algorithm. The notable aspect of this work is that a genetic algorithm is applied to a system simulation to find a good compression function for the system states (which are used by the automata to schedule the shop). For the open-loop controller, that is equivalent to joining narrow bins so that a single control action is applied to a wider range of input space.

[Heiss 1994] presents a different approach to a similar problem. In estimating a one-dimensional input-output mapping, spline interpolation is used to ensure that non-updated areas of the mapping have a smooth, continuous representation. The learning algorithm only requires information about the direction of approximation error, rather than the usual function-approximation requirement of error magnitude. Detailed analysis is given for a variety of spline smoothing techniques. This approach is fundamentally different from bin splitting, and more akin to nearest-neighbor interpolation schemes.

[Gray 1982] contains a collection of papers dealing with quantization and binning issues in pulse-code-modulation (PCM) theory.

Pattern recognition - Pattern recognition can be viewed as division of a "feature vector" space into separate areas, each denoting a separate pattern class. This can be done with an explicit recognition of the boundary areas, as in the linearly separating classifiers in [Duda 1973] or the nearest-neighbor selection rules that are used to choose control action for a grinding problem in [Asada 1989]. Other techniques include forms of competitive unsupervised networks in which pattern cluster centroids are learned iteratively. Well-known work includes Kohonen's Learning Vector Quantization (LVQ) [Kohonen 1989]; recent extensions are found in [Pal 1993].

No application of pattern classification techniques that is directly relevant to the open-loop controller's binning problem has yet been found in the literature. However, one recent paper has elements in common with the boundary adjustment work described here. In [Sun 1992], a neural network is used to implement a fuzzy classifier, which then has its membership functions adjusted via gradient descent. The classifier has a fixed number of bins, though, and requires labeled training data.

Function approximation - Relevant papers in this large field tend to focus on the "universal approximation" properties of schemes that combine simple nonlinear elements. In [Poggio 1989] and [Girosi 1989], a detailed theory of radial-basis-function neural networks is presented, largely in the context of function approximation. These papers also contain extensive reviews of function approximation literature. [Whitehead 1994] describes the use of a genetic algorithm to optimize the locations of radial basis functions in a time-series-prediction problem. [Ying 1994] presents a method for calculating the number of fuzzy membership functions necessary to approximate a polynomial to a given degree of accuracy.

1.6 Thesis Overview

This thesis examines various applications and extensions of the piecewise-constant controller to simple analytical models of deskewer-like systems. Chapter 2 analyzes the static structure, in which the number of bins and their locations are fixed during the training period. Closed-form results are given for special cases, but the analysis is sufficiently broad to allow numerical solution of more-general cases. Simulation is then used to validate the analytical results, and to evaluate the model-optimal controller when it is applied to systems that do not obey the simplifying assumptions used to derive the optimal structure.

Using results from Chapter 2, Chapter 3 sets forth a bin splitting method that dynamically changes the number of bins during training; this allows more-effective utilization of training samples. The optimal iterations at which to split bins are found for certain cases. Simulation results are again presented.

Chapter 4 summarizes the results of this work and points out directions for further research.

Appendix A presents two additional systems to which the self-tuning open-loop controller can be applied. Appendix B has more-detailed views of the deskewer, along with lists of parts. Appendix C presents an analytical model of the deskewer, demonstrating the lack of a traditional transfer function. A method for adjusting bin boundary locations based on measured output is given in Appendix D. Appendix E contains a more-general derivation of some of the key results from Chapter 3. Two methods for estimating model parameters used in the analyses of Chapters 2 and 3 are examined and simulated in Appendix F. Appendix G contains the commented *Mathematica* programs used to simulate the deskewer, validate the predictions of the optimization expressions, and evaluate the controller when applied to more-general systems.

Chapter 2
The Static-Structure Case

2.1 Introduction

This chapter presents the analysis of the optimal structure for a piecewise-constant controller that has a fixed number of bins, in fixed locations, for the duration of the training period. The optimization criteria is the minimization of expected average squared output after training has ceased.

The analysis begins with an examination of an algorithm for updating the control action associated with each bin. Next, the effect of the presence of multiple bins on learning rate is considered. In order to find the optimal controller structure, the results from the analysis of learning are then combined with assumptions on the variation of output across the width of single bins.

At each step, the analysis proceeds with maximal generality. When an assumption is needed to continue it is labeled as such. The final, optimal controller is thus derived for a model built upon these assumptions. Most can be replaced with more-general suppositions at the price of either increased analytical complexity, or the need to resort to numerical solutions. Extensions to the analysis are discussed in Section 4.2.

2.2 Analysis of the Learning Algorithm

2.2.1 Effect of Learning Efficiency

If only a fixed number of training samples is available, selection of the proper number of input space divisions presents a clear tradeoff. While a large number of bins permits smaller output error by giving smaller sections of the input range their own control actions, and thus more closely approximating each θ_{in}'s optimal τ, the expected number of notes falling into each bin during the training process is lower than that of a controller with fewer bins. The increase in error due to insufficient training may offset the effect of narrow bins. Predicting the potential difficulties caused by the amount of training clearly depends on the efficiency of the algorithm used to adjust τ; once we have specified the algorithm, we can determine the number of bins that optimizes the balance between bin width and ability to reduce the average squared output in each bin through training.

2.2.2 Expected Outputs for a Fixed Number of Trials in One Bin

The adjustment method examined here is a standard gradient descent technique that was applied to the deskewer in [Levinsky 1992]. This method takes the error from a single training cycle, divides it by a "learning constant," and adds the result to the previous control action. This strategy was employed in early model-reference adaptive control systems to adjust system gains; in [Astrom 1989] it is referred to as the "MIT rule." It is a special case of the stochastic approximation methods discussed in [Gelb 1974], in which the learning constant is fixed. It is also identical to Newton's method of finding roots, but with a fixed Jacobian. Many other learning methods can be constructed, including those which save past data and use more than a single output error measurement to update control action, but in this thesis we examine the gradient descent method because of its analytical simplicity, ease of implementation, and continuing popularity in learning control. More information can be found in [Gardner 1986].

In this section we examine a single bin. The subscript n refers to the values of these variables at specific training iterations. Each time an output θ_{out_n} is observed, the following update rule is applied:

$$\tau_{n+1} = \tau_n + \frac{\theta_{out_n}}{\gamma} \qquad (2.2\text{-}1)$$

where γ is the learning constant. Although we are here trying to drive θ_{out} to zero, the following derivation does not change substantially if we select a different desired output value (reference input) for θ_{out}. We make the following assumptions:

Assumption 2.2.1: The mean output of a bin, composed of outputs from all inputs that fall in the bin, is given by:

$$\mu \text{ of bin } i = \mu_i = \frac{1}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} \int_{a_{i-1}}^{a_i} \overline{\theta_{out}}(\theta_{in},\tau)\, p(\theta_{in})\, d\theta_{in} \qquad (2.2\text{-}2)$$

and is assumed to be a monotonic function of τ with bounded derivatives. This is the sign and boundedness restriction referred to in Section 1.1:

$$0 < \lambda \le \left|\frac{d\mu_i}{d\tau}\right| \le \nu, \; with \; \frac{d\mu_i}{d\tau} < 0 \qquad (2.2\text{-}3)$$

The expression for μ contains a factor of $\frac{1}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}}$ to normalize the value of $\int_{a_{i-1}}^{a_i} \overline{\theta_{out}}(\theta_{in},\tau)\, p(\theta_{in})\, d\theta_{in}$, which is the contribution to the average output of the entire machine from bin i, appropriately weighted by $p(\theta_{in})$. When data from each bin is measured separately and a bin mean is computed, the bin is treated as a single "universe," and the data must be normalized by the percentage of notes that fall into that bin during machine operation.

Assumption 2.2.2: The learning constant is larger than one-half the upper bound on the derivative postulated in equation (2.2-3):

$$\gamma > \frac{\nu}{2} \tag{2.2-4}$$

This restriction is used below in (2.2-9) to guarantee the convergence of the learning algorithm.

Assumption 2.2.3: θ_{out_n} equals the average output of the bin, μ, plus an independent (uncorrelated) noise term r of zero mean and constant variance σ_{bin}^2:

$$\theta_{out_n} = \mu_n + r_n \tag{2.2-5}$$

This ends the assumptions. We define a (possibly) varying scaling factor c that relates the change in μ to the change in τ:

$$\mu_{n+1} - \mu_n = -c_n\left(\tau_{n+1} - \tau_n\right) \tag{2.2-6}$$

By (2.2-3) and the mean value theorem of calculus, c is bounded:

$$\lambda \le c_n \le \nu \tag{2.2-7}$$

Combining (2.2-1), (2.2-5), and (2.2-6):

$$\mu_{n+1} - \mu_n = -c_n\left(\tau_{n+1} - \tau_n\right) = -c_n\left(\frac{\mu_n + r_n}{\gamma}\right) \tag{2.2-8}$$

which shows that μ can be modeled as the output of a first-order time-varying Gauss-Markov process:

$$\mu_{n+1} = \left(1 - \frac{c_n}{\gamma}\right)\mu_n - \frac{c_n}{\gamma} r_n \tag{2.2-9}$$

Now we determine the expected values of τ for an arbitrary number of training cycles. As r has a mean of zero, the expected value of μ after one training cycle is:

$$E(\mu_{n+1}) = \left(1 - \frac{c_n}{\gamma}\right)E(\mu_n) \tag{2.2-10}$$

Assumption 2.2.2 insures $\left|1 - \frac{c_n}{\gamma}\right| < 1$, which guarantees convergence of $E(\mu)$ to zero as n tends towards infinity.

$E(\mu^2)$ is used later in this chapter. Squaring (2.2-9) and taking expected values:

$$E(\mu_{n+1}^2) = \left(1 - \frac{c_n}{\gamma}\right)^2 E(\mu_n^2) + \frac{c_n^2}{\gamma^2}\sigma_{bin}^2 \tag{2.2-11}$$

Conservative bounds are found by noting:

$$\left(1 - \frac{\nu}{\gamma}\right)^2 E(\mu_n^2) + \frac{\lambda^2}{\gamma^2}\sigma_{bin}^2 < E(\mu_{n+1}^2) < \left(1 - \frac{\lambda}{\gamma}\right)^2 E(\mu_n^2) + \frac{\nu^2}{\gamma^2}\sigma_{bin}^2 \tag{2.2-12}$$

Defining $\phi = \left(1 - \frac{\lambda}{\gamma}\right)^2$ for clarity and taking examples of the first three trials in a bin:

$$E(\mu_1^2) < \phi E(\mu_0^2) + \frac{\nu^2}{\gamma^2}\sigma_{bin}^2$$

$$E(\mu_2^2) < \phi^2 E(\mu_0^2) + \phi\frac{\nu^2}{\gamma^2}\sigma_{bin}^2 + \frac{\nu^2}{\gamma^2}\sigma_{bin}^2 \tag{2.2-13}$$

$$E(\mu_3^2) < \phi^3 E(\mu_0^2) + \phi^2\frac{\nu^2}{\gamma^2}\sigma_{bin}^2 + \phi\frac{\nu^2}{\gamma^2}\sigma_{bin}^2 + \frac{\nu^2}{\gamma^2}\sigma_{bin}^2$$

we find:

$$E(\mu_n^2) < \phi^n E(\mu_0^2) + \frac{\nu^2}{\gamma^2}\sigma_{bin}^2 \sum_{j=0}^{n-1} \phi^j \tag{2.2-14}$$

Using the algebraic identity:

$$\frac{1 - x^n}{1 - x} = x^{n-1} + x^{n-2} + x^{n-3} + \ldots + x + 1 = \sum_{j=0}^{n-1} x^j \tag{2.2-15}$$

(2.2-14) is simplified to:

$$E(\mu_n^2) < \phi^n E(\mu_0^2) + \frac{(1 - \phi^n)}{(1 - \phi)}\frac{\nu^2}{\gamma^2}\sigma_{bin}^2 \tag{2.2-16}$$

A lower bound is found by exchanging ν for λ in the expression for ϕ. It is the presence of bounds on $E(\mu_n^2)$, rather than exact values, which necessitates the optimization of upper bounds on error in both this chapter and the next.

2.2.3 Expected Outputs for Multiple Bins

If we are given N training samples and have b bins, it is incorrect to assume that N/b samples will fall into each bin; we must explicitly calculate the expected value of output using the probabilities of n equalling particular values. If p_i is the probability that the number of samples n that fall in a particular bin is i, and $E(\mu_{n=j}^2)$ is the expected squared average output of that bin when j samples are used in training, $E(\mu^2)$ in that bin is given by:

$$E(\mu^2) = p_0 E(\mu_{n=0}^2) + p_1 E(\mu_{n=1}^2) + \ldots + p_N E(\mu_{n=N}^2) \tag{2.2-17}$$

Using the upper bound given in (2.2-16):

$$\begin{aligned} E(\mu^2) < {} & p_0 E(\mu_{n=0}^2) + p_1 \phi E(\mu_{n=0}^2) + p_1 \frac{\nu^2}{(1 - \phi)\gamma^2}\sigma_{bin}^2 - p_1 \phi \frac{\nu^2}{(1 - \phi)\gamma^2}\sigma_{bin}^2 \\ & + p_2 \phi^2 E(\mu_{n=0}^2) + p_2 \frac{\nu^2}{(1 - \phi)\gamma^2}\sigma_{bin}^2 - p_2 \phi^2 \frac{\nu^2}{(1 - \phi)\gamma^2}\sigma_{bin}^2 + \ldots \\ & + p_N \phi^N E(\mu_{n=0}^2) + p_N \frac{\nu^2}{(1 - \phi)\gamma^2}\sigma_{bin}^2 - p_N \phi^N \frac{\nu^2}{(1 - \phi)\gamma^2}\sigma_{bin}^2 \end{aligned} \tag{2.2-18}$$

Rearranging:

$$
\begin{aligned}
E(\mu^2) < E(\mu_{n=0}^2)&\left(p_0 + p_1\phi + p_2\phi^2 + \ldots + p_N\phi^N\right) \\
&+ \frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2\left(p_1 + p_2 + \ldots + p_N\right) \\
&- \frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2\left(p_1\phi + p_2\phi^2 + \ldots + p_N\phi^N\right)
\end{aligned}
\tag{2.2-19}
$$

The number of inputs falling into a bin is properly modeled as a Bernoulli process. There is a probability P (for any bin i, $P = \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}$) that on any single trial the input falls into a particular bin, but over the entire series of N trials, the probability distribution of n, the number of notes falling into the bin, is given by the binomial theorem:

$$
p_n = \frac{N!}{n!(N-n)!}P^n(1-P)^{N-n} \tag{2.2-20}
$$

Direct expansion of powers of the binomial $\left((1-P)+P\right)$ gives:

$$
\begin{aligned}
\left((1-P)+P\right)^N &= (1-P)^N + N(1-P)^{N-1}P + \frac{N(N-1)}{2!}(1-P)^{N-2}P^2 + \ldots + P^N \\
&= p_0 + p_1 + p_2 + \ldots + p_N
\end{aligned}
\tag{2.2-21}
$$

Similarly:

$$
\begin{aligned}
\left((1-P)+Px\right)^N &= (1-P)^N + N(1-P)^{N-1}Px + \frac{N(N-1)}{2!}(1-P)^{N-2}P^2x^2 + \ldots + P^Nx^N \\
&= p_0 + p_1x + p_2x^2 + \ldots + p_Nx^N
\end{aligned}
\tag{2.2-22}
$$

Substituting equations (2.2-21) and (2.2-22) into (2.2-19):

$$
E(\mu^2) < E(\mu_{n=0}^2)\left((1-P)+P\phi\right)^N + \frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2(1-p_0) - \frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2\left(\left((1-P)+P\phi\right)^N - p_0\right) \tag{2.2-23}
$$

Simplifying:

$$
\begin{aligned}
E(\mu^2) &< E(\mu_{n=0}^2)\left((1-P)+P\phi\right)^N + \frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2 - \frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2\left((1-P)+P\phi\right)^N \\
&= \left((1-P)+P\phi\right)^N E(\mu_{n=0}^2) + \left(1 - \left((1-P)+P\phi\right)^N\right)\frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2
\end{aligned}
\tag{2.2-24}
$$

Note that (2.2-24) has the same structure as (2.2-16), the expression for a single bin. The only effect of multiple bins is to increase the effective ϕ, and slow the training. We know the effective ϕ is larger because:

$$\left((1-P)+P\phi\right)-\phi=(1-P)(1-\phi)>0 \tag{2.2-25}$$

The first term of (2.2-24) disappears as N increases to infinity, but the second term remains:

$$\lim_{N\to\infty} E\left(\mu^2\right)<\frac{v^2}{(1-\phi)\gamma^2}\sigma_{bin}^2=\frac{v^2}{\left(2\gamma\lambda-\lambda^2\right)}\sigma_{bin}^2 \tag{2.2-26}$$

As γ increases, $E\left(\mu^2\right)$ goes to zero; the slower learning reduces the effect of noise (as is also the case in stochastic approximation).

2.3 Number and Location of Bins

2.3.1 Output Variation in a Single Bin

In order to continue the analysis we must now add assumptions about the variation of output across a single bin. From this section to the end of the chapter, subscripts on μ and S refer to bin numbers instead of training iterations.

Assumption 2.3.1: The distribution $p_\tau\left(\theta_{out}|\theta_{in}\right)$ is white-noise (succeeding samples of noise in the system output are completely uncorrelated with each other) and Gaussian with a mean given by the function $\overline{\theta_{out}}\left(\theta_{in},\tau\right)$ and variance σ^2. σ^2 is constant for all values of θ_{in} and τ.

Assumption 2.3.2: Within each bin, and for each fixed τ, $\overline{\theta_{out}}\left(\theta_{in},\tau\right)$ is approximated by a linear function of θ_{in}. This is:

$$\overline{\theta_{out}}\left(\theta_{in},\tau\right)=\beta_0+\beta_1\theta_{in} \tag{2.3-1}$$

where β_0 and β_1 are constants which may vary according to bin location and τ. Figure 2.1 graphically summarizes Assumptions 2.3.1 and 2.3.2.

As in Section 2.2.1, μ_i is defined as:

$$\mu_i=\frac{1}{\int_{a_{i-1}}^{a_i} p\left(\theta_{in}\right)d\theta_{in}}\int_{a_{i-1}}^{a_i}\overline{\theta_{out}}\left(\theta_{in},\tau_i\right)p\left(\theta_{in}\right)d\theta_{in} \tag{2.3-2}$$

Substituting (2.3-1) into (2.3-2)

$$\mu_i=\frac{1}{\int_{a_{i-1}}^{a_i} p\left(\theta_{in}\right)d\theta_{in}}\int_{a_{i-1}}^{a_i}\left(\beta_0+\beta_1\theta_{in}\right)p\left(\theta_{in}\right)d\theta_{in} \tag{2.3-3}$$

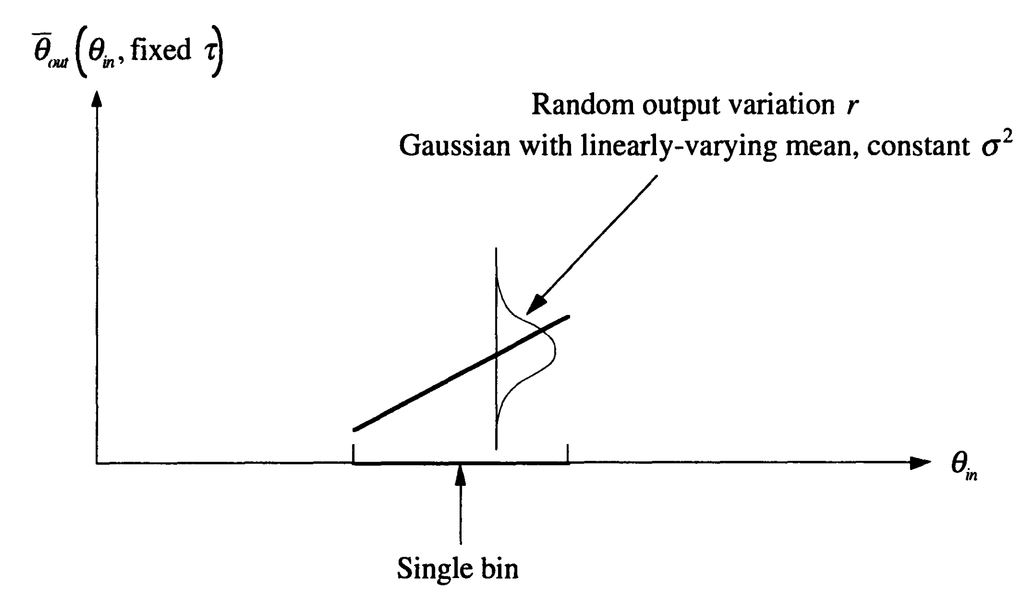

$\overline{\theta}_{out}\left(\theta_{in}, \text{fixed } \tau\right)$
Random output variation r
Gaussian with linearly-varying mean, constant σ^2
θ_{in}
Single bin

Figure 2.1: Output Variation Across Width of Bin

Similarly, the average squared output, S_i^2, is given by:

$$S_i^2 = \frac{1}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} \int_{a_{i-1}}^{a_i} \overline{\theta_{out}^2}\, p(\theta_{in})\, d\theta_{in} \tag{2.3-4}$$

From Assumptions 2.3.1 and 2.3.2, $\overline{\theta_{out}^2}(\theta_{in}, \tau_i)$ is given by:

$$\overline{\theta_{out}^2}(\theta_{in}, \tau_i) = \left(\overline{\theta_{out}}(\theta_{in}, \tau_i)\right)^2 + \sigma^2 = \beta_1^2\theta_{in}^2 + 2\beta_1\beta_0\theta_{in} + \beta_0^2 + \sigma^2 \tag{2.3-5}$$

Substituting:

$$\begin{aligned} S_i^2 &= \frac{1}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} \int_{a_{i-1}}^{a_i} \left(\beta_1^2\theta_{in}^2 + 2\beta_1\beta_0\theta_{in} + \beta_0^2 + \sigma^2\right) p(\theta_{in})\, d\theta_{in} \\ &= \frac{1}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} \int_{a_{i-1}}^{a_i} \left(\beta_1^2\theta_{in}^2 + 2\beta_1\beta_0\theta_{in}\right) p(\theta_{in})\, d\theta_{in} + \beta_0^2 + \sigma^2 \end{aligned} \tag{2.3-6}$$

Squaring (2.3-3) and substituting into (2.3-6):

$$S_i^2 = \frac{\int_{a_{i-1}}^{a_i} \beta_1^2\theta_{in}^2 p(\theta_{in})\, d\theta_{in}}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} - \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1\theta_{in} p(\theta_{in})\, d\theta_{in}\right)^2}{\left(\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}\right)^2} + \mu_i^2 + \sigma^2 \tag{2.3-7}$$

If we do not know the effect of changing τ on β_1, it is possible that at a non-zero μ_i, β_1 may decrease enough so that S_i^2 will be less than that found when $\mu_i = 0$. Because this generally cannot be predicted *a priori*, it is reasonable to adjust τ so that $\mu_i = 0$ in each bin.

The quantity σ_{bin}^2, first used in Section 2.2.2, can be immediately identified in (2.3-7):

$$\sigma_{bin}^2 = \frac{\int_{a_{i-1}}^{a_i} \beta_1^2 \theta_{in}^2 p(\theta_{in})\, d\theta_{in}}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} - \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} p(\theta_{in})\, d\theta_{in}\right)^2}{\left(\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}\right)^2} + \sigma^2 \qquad (2.3\text{-}8)$$

2.3.2 Optimal Number of Bins for Arbitrary Input Distribution

For tractability, we add the following assumption:

Assumption 2.3.3: In all bins, regardless of boundaries or τ, β_1 is constant.

We now use (2.3-7) to determine the average squared output for the system. This is:

$$\begin{aligned} S^2 &= S_1^2 \int_{a_0}^{a_1} p(\theta_{in})\, d\theta_{in} + S_2^2 \int_{a_1}^{a_2} p(\theta_{in})\, d\theta_{in} + \ldots + S_b^2 \int_{a_{b-1}}^{a_b} p(\theta_{in})\, d\theta_{in} \\ &= \int_{a_0}^{a_b} \beta_1^2 \theta_{in}^2 p(\theta_{in})\, d\theta_{in} - \sum_{i=1}^{b} \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} p(\theta_{in})\, d\theta_{in}\right)^2}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} + \sum_{i=1}^{b} \mu_i^2 \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in} + \sigma^2 \end{aligned} \qquad (2.3\text{-}9)$$

By substituting the bound on expected squared average output given in (2.2-24), we find:

$$\begin{aligned} E(S^2) < &\int_{a_0}^{a_b} \beta_1^2 \theta_{in}^2 p(\theta_{in})\, d\theta_{in} - \sum_{i=1}^{b} \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} p(\theta_{in})\, d\theta_{in}\right)^2}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} + \sigma^2 \\ &+ \sum_{i=1}^{b} \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in} \left(\left(1 - \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}\right) + \phi \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in} \right)^N E(\mu_{n=0,\,\mathrm{bin}\, i}^2) \\ &+ \sum_{i=1}^{b} \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in} \left(1 - \left(\left(1 - \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}\right) + \phi \int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in} \right)^N \right) \frac{v^2}{(1-\phi)\gamma^2} \sigma_{bin}^2 \end{aligned} \qquad (2.3\text{-}10)$$

Equation (2.3-10), the upper bound on mean squared output, must be optimized with respect to the number of bins b and the bin locations a_i. Generally, this is done numerically for arbitrary input distributions.

2.3.3 Optimal Bin Locations for Arbitrary Input Distribution

Because the general solution to (2.3-10) is currently unknown, we simplify the problem by finding "reasonable" bin locations and then minimizing $E(S^2)$ with respect to number of bins alone. Two special cases are immediately suggested. First, if we have a very large number of training samples available, it is possible to drive $\mu_i \approx 0$ in (2.3-9). Setting all μ_i to 0:

$$S^2 = \int_{a_0}^{a_b} \beta_1^2 \theta_{in}^2 p(\theta_{in})\, d\theta_{in} - \sum_{i=1}^{b} \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} p(\theta_{in})\, d\theta_{in} \right)^2}{\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}} + \sigma^2 \qquad (2.3\text{-}11)$$

The first and third terms of (2.3-11) are fixed and independent of bin location, so by maximizing $\sum_{i=1}^{b} \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} p(\theta_{in}) d\theta_{in} \right)^2}{\int_{a_{i-1}}^{a_i} p(\theta_{in}) d\theta_{in}}$, we minimize (2.3-11). Such a structure is the best possible when all control actions are set so that the mean output of each bin is zero. This is a reasonable approach, because if additional training date should become available after initial training is complete, the control actions might be made "nearly" perfect.

The second special case is that of equiprobable bins. Here $\int_{a_{i-1}}^{a_i} p(\theta_{in})\, d\theta_{in}$ is identical in each bin; and equals $1/b$. (2.3-10) reduces to:

$$\begin{aligned} E(S^2) < & \int_{a_0}^{a_b} \beta_1^2 \theta_{in}^2 p(\theta_{in})\, d\theta_{in} - b \sum_{i=1}^{b} \left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} p(\theta_{in})\, d\theta_{in} \right)^2 + \sigma^2 \\ & + \frac{1}{b} \sum_{i=1}^{b} \left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b} \right)^N E(\mu^2_{n=0,\,\text{bin}\,i}) + \frac{1}{b} \sum_{i=1}^{b} \left(1 - \left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b} \right)^N \right) \frac{\nu^2}{(1-\phi)\gamma^2} \sigma^2_{bin} \end{aligned} \qquad (2.3\text{-}12)$$

Even with the simplifying assumption of equiprobable bins, (2.3-12) must be solved numerically for arbitrary input distributions.

2.4 Solution of the Uniform-Input-Distribution Case

2.4.1 Optimal Bin Locations for Uniform Input Distribution

We now minimize (2.3-10) for the special case of a uniform distribution of the input variable:

$$p(\theta_{in}) = \frac{1}{W} \qquad (2.4\text{-}1)$$

where W is the range of the input variable. The first method of bin location, in which all μ_i are assumed to equal 0, requires the maximization of the second term of (2.3-11), which we label V. That is:

$$V = \sum_{i=1}^{b} \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} \frac{1}{W}\, d\theta_{in}\right)^2}{\int_{a_{i-1}}^{a_i} \frac{1}{W}\, d\theta_{in}} \tag{2.4-2}$$

To maximize V, we differentiate it with respect to each bin boundary and set the resulting system of equations to zero. Evaluating the terms in (2.4-2) that relate to an arbitrary bin i and its neighbor, bin $(i + 1)$:

$$\frac{\beta_1^2}{4W}\frac{\left(a_i^2 - a_{i-1}^2\right)^2}{\left(a_i - a_{i-1}\right)} + \frac{\beta_1^2}{4W}\frac{\left(a_{i+1}^2 - a_i^2\right)^2}{\left(a_{i+1} - a_i\right)} = \frac{\beta_1^2}{4W}\left(a_i - a_{i-1}\right)\left(a_i + a_{i-1}\right)^2 + \frac{\beta_1^2}{4W}\left(a_{i+1} - a_i\right)\left(a_{i+1} + a_i\right)^2 \tag{2.4-3}$$

Differentiating (2.4-3) with respect to bin boundary a_i:

$$\begin{aligned}\frac{\partial V}{\partial a_i} &= \frac{\beta_1^2}{4W}\left(\left(a_i + a_{i-1}\right)^2 + 2\left(a_i - a_{i-1}\right)\left(a_i + a_{i-1}\right) - \left(a_{i+1} + a_i\right)^2 + 2\left(a_{i+1} - a_i\right)\left(a_{i+1} + a_i\right)\right) \\ &= \frac{\beta_1^2}{4W}\left(a_i^2 + 2a_i a_{i-1} + a_{i-1}^2 + 2a_i^2 - 2a_{i-1}^2 - a_{i+1}^2 - 2a_{i+1}a_i - a_i^2 + 2a_{i+1}^2 - 2a_i^2\right) \\ &= \frac{\beta_1^2}{4W}\left(2a_i a_{i-1} - a_{i-1}^2 + a_{i+1}^2 - 2a_{i+1}a_i\right) = \frac{\beta_1^2}{4W}\left(a_{i+1} - a_{i-1}\right)\left(-2a_i + \left(a_{i+1} + a_{i-1}\right)\right)\end{aligned} \tag{2.4-4}$$

Setting (2.4-4) to zero:

$$a_i = \frac{1}{2}\left(a_{i+1} + a_{i-1}\right) \tag{2.4-5}$$

Equation (2.4-5) must hold for all bins in order to minimize V. It is clear that (2.4-5) locates a minimum, as differentiation of (2.4-4) shows:

$$\frac{\partial^2 V}{\partial^2 a_i} = -2 \tag{2.4-6}$$

Rearranging (2.4-5) gives the following difference equation:

$$a_{i+1} = 2a_i - a_{i-1} \tag{2.4-7}$$

If $a_i = a_{i-1} + w$, where w is a constant that we can select, we obtain the inductive step:

$$a_{i+1} = 2a_i - a_{i-1} = 2a_i - a_i + w = a_i + w \tag{2.4-8}$$

Since a_0 is fixed, if we choose the inductive basis $a_1 = a_0 + \frac{W}{b}$, (2.4-8) gives:

$$a_i = a_0 + \frac{Wi}{b} \tag{2.4-9}$$

and the boundary condition $a_b = a_0 + W$ is satisfied. Thus, for the special case of uniform input distributions, an equiprobable bin distribution satisfies (2.4-5), which maximizes V and thus minimizes (2.3-11). The two special cases of bin location discussed in Section 2.3.3 yield identical results, and will be treated together below.

2.4.2 Optimal Number of Bins for Uniform Input Distribution

The special case of equiprobable bins was shown in (2.3-12). Inserting the uniform input distribution and simplifying:

$$\begin{aligned} E(S^2) &< \left(1 + \frac{v^2}{(1-\phi)\gamma^2}\right)\left(\frac{\beta_1^2(a_b^3 - a_0^3)}{3W} - \frac{b\beta_1^2}{W^2}\sum_{i=1}^{b}\left(\int_{a_{i-1}}^{a_i} \theta_{in}\, d\theta_{in}\right)^2 + \sigma^2\right) \\ &+ \frac{1}{b}\left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N \sum_{i=1}^{b} E\left(\mu_{n=0,\,\text{bin}\, i}^2\right) - \frac{\beta_1^2(a_b^3 - a_0^3)v^2}{3W(1-\phi)\gamma^2}\left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N \\ &+ \frac{bv^2}{(1-\phi)\gamma^2}\left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N \frac{\beta_1^2}{W^2}\sum_{i=1}^{b}\left(\int_{a_{i-1}}^{a_i} \theta_{in}\, d\theta_{in}\right)^2 - \frac{v^2\sigma^2}{(1-\phi)\gamma^2}\left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N \end{aligned} \tag{2.4-10}$$

The term $\sum_{i=1}^{b}\left(\int_{a_{i-1}}^{a_i} \theta_{in}\, d\theta_{in}\right)^2$ must now be evaluated. From (2.4-9) we can write:

$$\left(\int_{a_{i-1}}^{a_i} \theta_{in}\, d\theta_{in}\right)^2 = \left(\int_{a_{i-1}}^{a_{i-1}+\frac{W}{b}} \theta_{in}\, d\theta_{in}\right)^2 = \left(\frac{a_{i-1}W}{b} + \frac{W^2}{2b^2}\right) = \frac{a_{i-1}^2W^2}{b^2} + \frac{a_{i-1}W^3}{b^3} + \frac{W^4}{4b^4} \tag{2.4-11}$$

Substituting (2.4-9) into (2.4-11) and simplifying:

$$\sum_{i=1}^{b}\left(\frac{a_{i-1}^2W^2}{b^2} + \frac{a_{i-1}W^3}{b^3} + \frac{W^4}{4b^4}\right) = \frac{W^4}{b^4}\sum_{i=1}^{b} i^2 + \left(\frac{2a_0W^3}{b^3} - \frac{W^4}{b^4}\right)\sum_{i=1}^{b} i + \left(\frac{a_0^2W^2}{b} + \frac{a_0W^3}{b^2} + \frac{W^4}{4b^3}\right) \tag{2.4-12}$$

The well-known identities:

$$\begin{aligned} \sum_{i=1}^{b} i &= \frac{b(b+1)}{2} \\ \sum_{i=1}^{b} i^2 &= \frac{b(b+1)(2b+1)}{6} \end{aligned} \tag{2.4-13}$$

are substituted into (2.4-12), and we find after simplification:

$$\sum_{i=1}^{b}\left(\int_{a_{i-1}}^{a_i} \theta_{in} d\theta_{in}\right)^2 = -\frac{W^4}{12b^3} + \frac{W^4}{3b} + \frac{a_0 W^3 + a_0^2 W^2}{b} \tag{2.4-14}$$

We can also simplify (2.4-10) by noting:

$$\frac{a_b^3 - a_0^3}{W} = \frac{a_b^3 - a_0^3}{a_b - a_0} = a_b^2 + a_b a_0 + a_0^2 \tag{2.4-15}$$

Inserting (2.4-14), (2.4-15), and $W = a_b - a_0$ into (2.4-10):

$$\begin{aligned} E(S^2) &< \frac{1}{12}\frac{\beta_1^2}{b^2}\left(a_b^2 - 2a_b a_0 + a_0^2\right) + \frac{1}{b}\left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N \sum_{i=1}^{b} E\left(\mu_{n=0,\,\text{bin}\,i}^2\right) \\ &+ \frac{v^2}{(1-\phi)\gamma^2}\left(1 - \left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N\right)\left(\frac{\beta_1^2}{12b^2}\left(a_b^2 - 2a_b a_0 + a_0^2\right) + \sigma^2\right) + \sigma^2 \end{aligned} \tag{2.4-16}$$

But $W^2 = a_b^2 - 2a_b a_0 + a_0^2$; inserting this into (2.4-16), we see that there is no explicit dependence on either the starting point of the input range, a_0, or the ending point, a_b. The only essential information is their difference, which is the input range W:

$$\begin{aligned} E(S^2) &< \frac{1}{12}\frac{\beta_1^2 W^2}{b^2} + \frac{1}{b}\left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N \sum_{i=1}^{b} E\left(\mu_{n=0,\,\text{bin}\,i}^2\right) \\ &+ \frac{v^2}{(1-\phi)\gamma^2}\left(1 - \left(\left(1 - \frac{1}{b}\right) + \frac{\phi}{b}\right)^N\right)\left(\frac{\beta_1^2 W^2}{12b^2} + \sigma^2\right) + \sigma^2 \end{aligned} \tag{2.4-17}$$

A value for $\sum_{i=1}^{b} E\left(\mu_{n=0,\,bin\,i}^2\right)$ must now be assumed. We here choose to examine the initial training of the controller from a state of no knowledge, as a baseline for comparing various bin structures. In Section 4.2, a procedure is suggested for using (2.4-17) to choose a structure for a controller that will be updated from an already-trained state.

Assumption 2.4.1: All bins have a control action of zero at the beginning of the training process. Furthermore, we assume that at zero control action, the output is identical to the input, with no additional variance. This is equivalent to assuming that the error in the system accounted for in σ^2 arises strictly from the application of control action, and has no component due to sensing errors.

Since the output mirrors the uniform input, there is no need to use $E\left(\mu_{\text{bin}\,i}^2\right)$. The term μ^2 is well-defined:

$$\mu_{n=0,\,\text{bin}\,i}^2 = \left(a_0 + \frac{(2i-1)}{2}\frac{W}{b}\right)^2 \tag{2.4-18}$$

Thus:

$$\sum_{i=1}^{b} \mu^2_{n=0,\,\mathrm{bin}\,i} = \frac{W^2}{b^2}\sum_{i=1}^{b} i^2 + \left(\frac{2a_0W}{b} - \frac{W^2}{b^2}\right)\sum_{i=1}^{b} i + \left(\frac{W^2}{4b} + ba_0^2 - a_0W\right) \quad (2.4\text{-}19)$$

Using the identities of (2.4-13):

$$\sum_{i=1}^{b} \mu^2_{n=0,\,\mathrm{bin}\,i} = \frac{bW^2}{3} - \frac{W^2}{12b} + ba_0^2 + a_0bW \quad (2.4\text{-}20)$$

Substituting (2.4-20) into (2.4-17):

$$\begin{aligned} E\left(S^2\right) < \frac{1}{12}\frac{\beta_1^2W^2}{b^2} &+ \left(\left(1-\frac{1}{b}\right)+\frac{\phi}{b}\right)^N\left(\frac{W^2}{3} - \frac{W^2}{12b^2} + a_0^2 + a_0W\right) \\ &+ \frac{v^2}{(1-\phi)\gamma^2}\left(1-\left(\left(1-\frac{1}{b}\right)+\frac{\phi}{b}\right)^N\right)\left(\frac{1}{12}\frac{\beta_1^2W^2}{b^2} + \sigma^2\right) + \sigma^2 \end{aligned} \quad (2.4\text{-}21)$$

Substituting $\phi = \left(1 - \frac{\lambda}{\gamma}\right)^2$ gives the final expression for an upper bound on the expected squared output of the controller after all training has ceased. Again, the key assumptions are uniform input distribution, uniform bin widths, and identical β_1 and σ^2 in each bin:

$$\begin{aligned} E\left(S^2\right) < \frac{1}{12}\frac{\beta_1^2W^2}{b^2} &+ \left(\left(1-\frac{1}{b}\right)+\frac{\left(1-\frac{\lambda}{\gamma}\right)^2}{b}\right)^N\left(\frac{W^2}{3} - \frac{W^2}{12b^2} + a_0^2 + a_0W\right) \\ &+ \frac{v^2}{\left(1-\left(1-\frac{\lambda}{\gamma}\right)^2\right)\gamma^2}\left(1-\left(\left(1-\frac{1}{b}\right)+\frac{\left(1-\frac{\lambda}{\gamma}\right)^2}{b}\right)^N\right)\left(\frac{1}{12}\frac{\beta_1^2W^2}{b^2} + \sigma^2\right) + \sigma^2 \end{aligned} \quad (2.4\text{-}22)$$

This expression can be differentiated with respect to b to find the optimum number of bins, but when left in terms of all the variable parameters, the result is a lengthy equation whose analytical solution is not known at this time. A less general but easier-to-grasp approach is to insert known values of parameters and make numerical calculations.

The two parameters over which we have direct control are b, the number of bins, and γ, the learning rate. Figure 2.2 is a graph of (2.4-22), while Figures 2.3 and 2.4 are contour plots of Figure 2.2. Figure 2.3 gives an overview of the meaningful design space, and Figure 2.4 examines the minimum area more closely. Values of parameters inserted into (2.4-22) represent appropriate values for the deskewer (see [Levinsky 1992]): $\lambda = 0.1$, $v = 0.5$, $\beta_1 = 1$, $\sigma = 1.75$, $W = 30$, and $a_0 = 0$. We choose $N = 1000$, which is a training set that can easily be completed in a few hours.

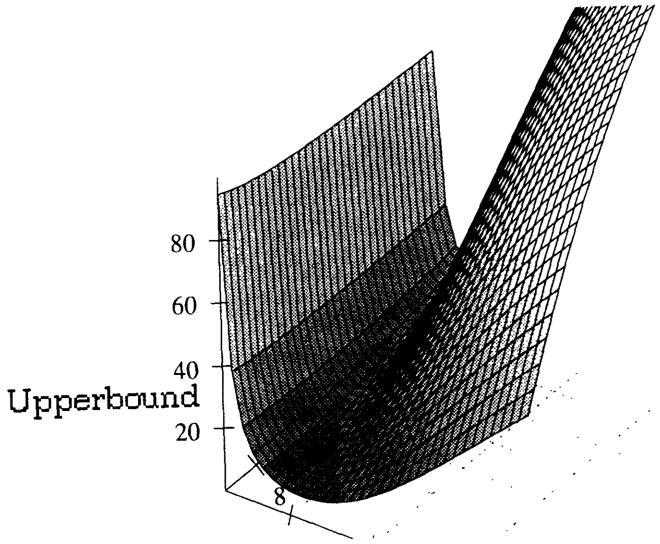


Figure 2.2: Plot of Upper Bound of $E(S^2)$ vs. b and γ

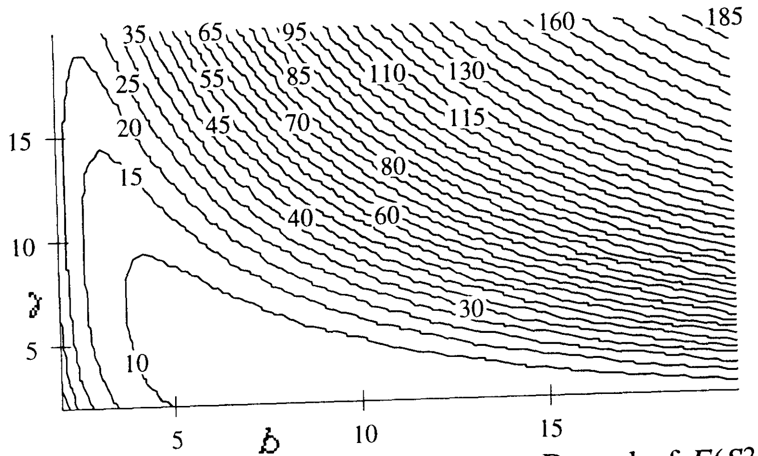


Figure 2.3: Contour Plot of Upper Bound of $E(S^2)$

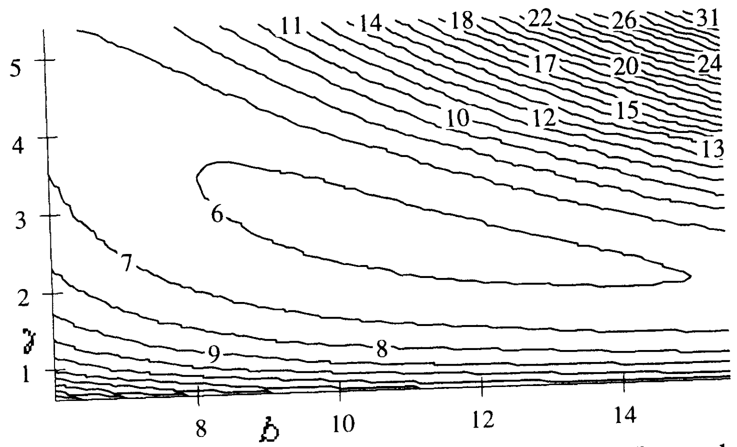


Figure 2.4: Contour Plot of Minimum of Upper Bound of $E(S^2)$

Note that the minimum of the upper bound does not necessarily coincide with the minimum that would be found if the system were known perfectly. For example, if the system model were completely linear, and $\lambda = \nu = 0.3$, the area containing the minimum contours of error differs slightly from that depicted in Figure 2.4. The "perfect knowledge" area is shown in Figure 2.5. In b-γ space, its minimum is located close to the upper bound's minimum, which validates our reliance on the upper bound.

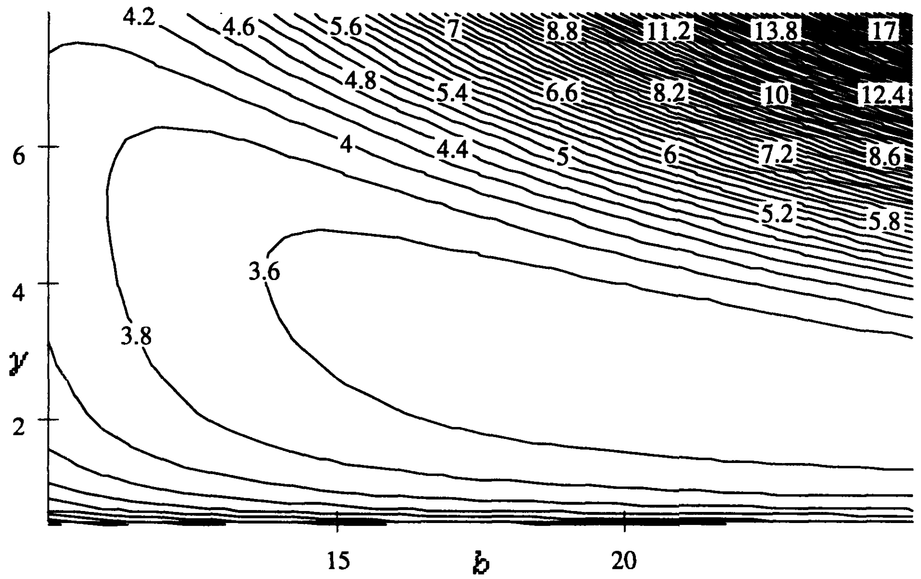


Figure 2.5: Contour Plot of True Minimum of $E(S^2)$

2.5 Simulation Results

To validate the preceding analytical predictions and to test the controller on systems that do not obey the assumptions used in the controller's derivation, a simulation has been written in *Mathematica*. The code is presented in Section F.2. For the static-structure case, the desired result is a graph showing output error as a function of number of bins.

The simulations in this section, as well as those in Chapter 3, begin from a state of no knowledge (uniformly zero control action). The deskewer is modeled with a given analytical expression, which allows faster simulation than an explicitly physics-based model such as the one given in Appendix C. The parameters used are identical to those of the preceding section: $\lambda = 0.1$, $\nu = 0.5$, $\beta_1 = 1$, $\sigma = 1.75$, $W = 30$, and $a_0 = 0$. γ is chosen to be 5, because Figures 2.4 and 2.5 indicate that such a value gives good, if not optimal, performance.

In these simulations, mean squared output is evaluated by fixing control actions and simulating a series of test iterations. The simulation trains the deskewer for 1200 cycles (chosen because this number allows six bin splits, as seen in the next chapter), then freezes control-action updating to evaluate output for 200 cycles. This is repeated for static structures ranging from one to 51 bins; at each bin structure, the entire process of training and evaluating is repeated ten times. These ten repeats are averaged to find each plotted data point.

The performance of the controller is evaluated on four system models. The first is linear:

$$\theta_{out}(\theta_{in},\tau) = \theta_{in} - 0.3\tau + r \qquad (2.5\text{-}1)$$

where r is a normally-distributed noise term of variance σ. Note that $\beta_1 = 1$, and $\lambda = \nu = 0.3$. This is the only model which obeys all of the assumptions used to derive the optimality equations.

The second model adds high-frequency variations, comparable in wavelength to the size of a bin (in the θ_{in} direction) and size of a single update in control action (in the τ direction). Here, $\lambda = 0.1$ and $\nu = 0.5$, but β_1 is no longer constant:

$$\theta_{out}(\theta_{in},\tau) = \theta_{in} + \sin(\theta_{in}) - 0.3\tau + \sin(0.2\tau) + r \qquad (2.5\text{-}2)$$

The third model adds the equivalent of a low-frequency variation of β_1 in θ_{in}, while retaining the sine terms. θ_{out} is modeled as an increasing parabolic function of θ_{in}, with the contribution to $\frac{\partial \theta_{out}}{\partial \theta_{in}}$ from the parabolic term, the analog of β_1, ranging from zero at $\theta_{in} = 0°$ to 2 at $\theta_{in} = 30°$. This is a system in which the output accelerates faster than linearly in the input variable, but the error in the 0° - 30° input range is less than that of the linear system in (2.5-1):

$$\theta_{out}(\theta_{in},\tau) = \frac{1}{30}\theta_{in}^2 + \sin(\theta_{in}) - 0.3\tau + \sin(0.2\tau) + r \qquad (2.5\text{-}3)$$

The fourth model has a parabolic term, but the curve is inverted and shifted to the right, relative to the model in (2.5-3). The parabolic term's contribution to the derivative here decreases from 2 at $\theta_{in} = 0°$ to zero at $\theta_{in} = 30°$. The error in the 0° - 30° range is greater in this model than in the preceding three.

$$\theta_{out}(\theta_{in},\tau) = -\frac{1}{30}(\theta_{in} - 30)^2 + 30 + \sin(\theta_{in}) - 0.3\tau + \sin(0.2\tau) + r \qquad (2.5\text{-}4)$$

Three solid curves are plotted in each of Figures 2.6 through 2.13. The central curve is the predicted average squared output when λ and ν are identical and known exactly, and the input distribution is uniform, while the upper and lower curves are the upper and lower bounds on average squared output predicted when $\lambda = 0.1$, $\nu = 0.5$ (and vice versa), and the input distribution is uniform. While only the simulation of Figure 2.6, using model (2.5-1) and a uniform input distribution, obeys the assumptions about system behavior used to find the optimal controller, the curves are drawn to clearly show how results obtained from systems of increased complexity differ from predictions of output based on the simplified models of this chapter.

Figures 2.6 through 2.9 present simulation results using models (2.5-1) through (2.5-4), respectively. The input distribution is uniformly distributed from 0° to 30°.

In Figure 2.6, results from the simulation of the first model, which obeys the optimization assumptions, show the validity of the theoretical expressions for expected error.

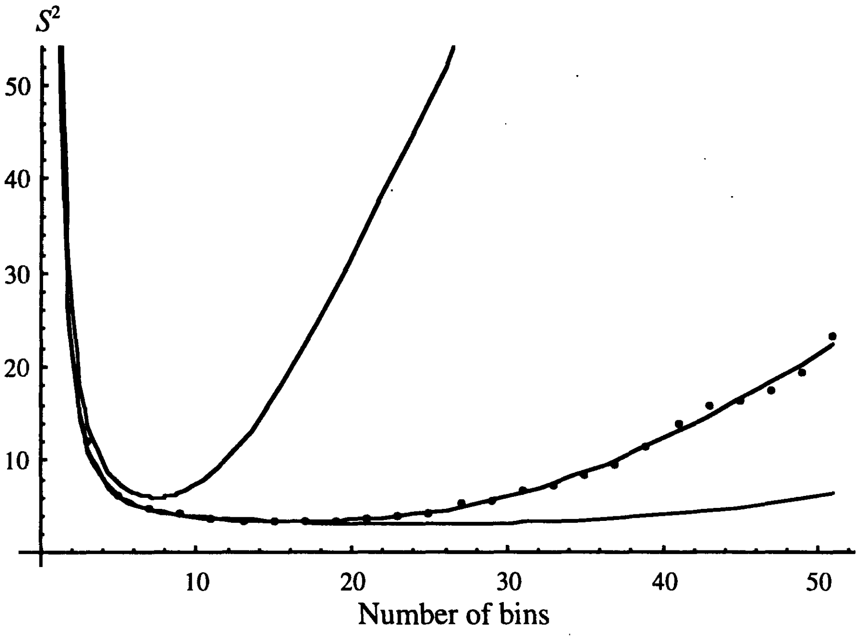


Figure 2.6: S^2 vs. Bin Number - First Model, Uniform Distribution

The addition of the sine terms does not substantially affect output error predictions, as seen in Figure 2.7.

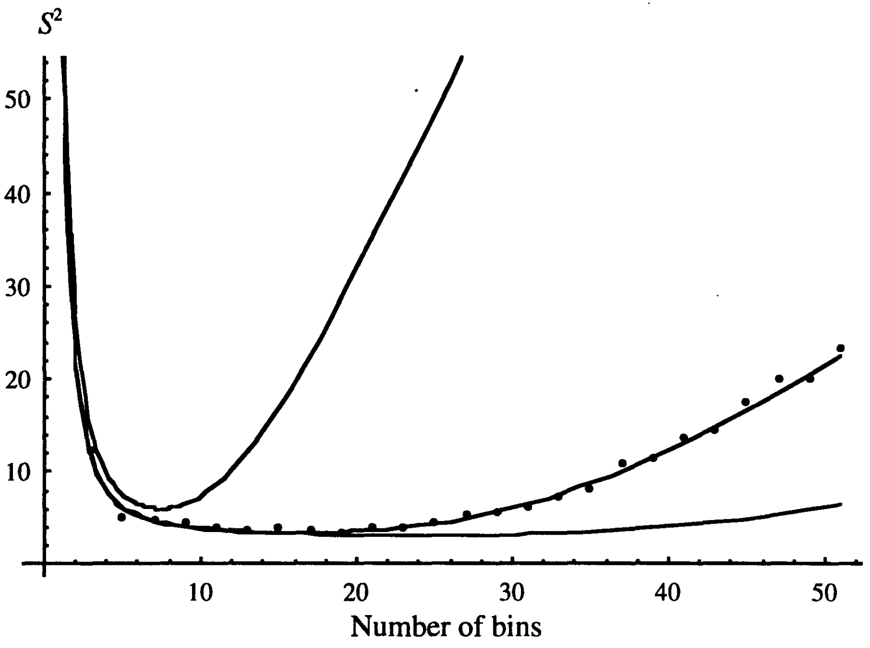


Figure 2.7: S^2 vs. Bin Number - Second Model, Uniform Distribution

The third model has less initial error in the 0° - 30° range than in the first two, which suggests that more bins might be desirable. This is seen in Figure 2.8, in which the output error at high bin numbers is less than that predicted for the linear-variation-across-bin-width models.

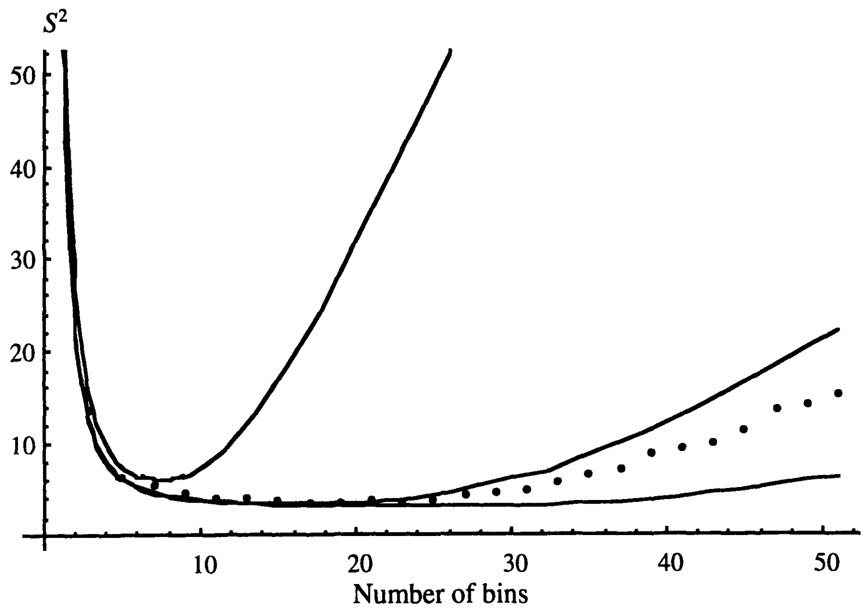


Figure 2.8: S^2 vs. Bin Number - Third Model, Uniform Distribution

Similarly, the greater initial error of the fourth model suggests the use of fewer bins than in the linear case, to allow for more training. Figure 2.9 shows that the minimum error point is nearly identical, but the output error for bin numbers greater than approximately 20 is higher than that predicted for the linear case.

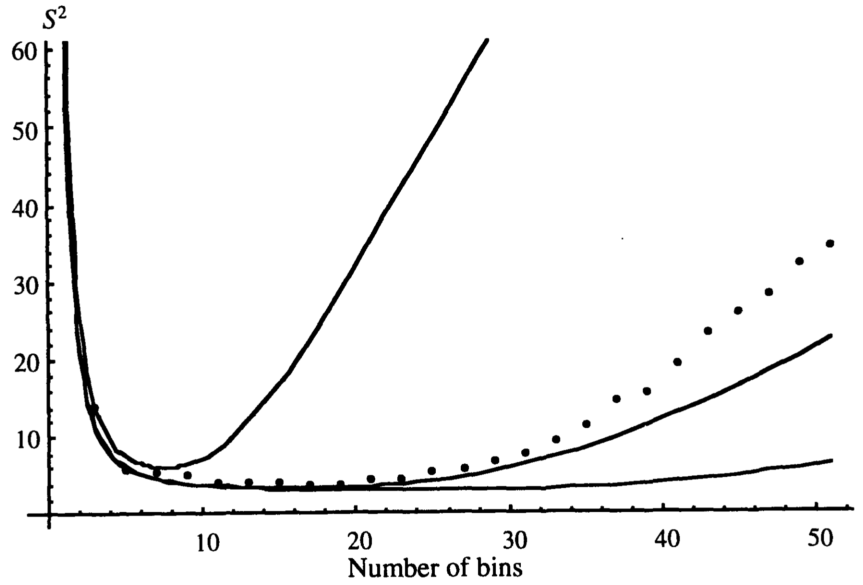


Figure 2.9: S^2 vs. Bin Number - Fourth Model, Uniform Distribution

The assumption of a uniform input distribution is perhaps the most unrealistic element of the analysis of this chapter. Real systems, including ATM, are much more likely to have modal distributions, with the majority of samples clustered around the nominal operating point. Taking the deskewer as an example, most notes are relatively unskewed.

While optimal controllers can be found numerically for arbitrary input distributions, we would like to know how a controller designed under the tractable uniform-distribution assumption fares when used with a different distribution. Figures 2.10 through 2.13 use the models of (2.5-1) through (2.5-4), respectively, but also employ a normal input distribution of 0° mean and 15° standard deviation. The tails of the distribution beyond -30° and 30° are cut off, as these represent unrecoverable situations; the skew is too excessive to allow proper note travel.

Two simulations have been performed for each of these test models. The first uses equal-width bins, as in the simulations represented in Figures 2.6 through 2.9, while the second uses bins that are equiprobable over the 0°-30° range. In all cases, the equiprobable bins outperform the equal-width bins. The higher-error data (at a given bin number) is from the equal-width simulations, while the lower-error data is from the equiprobable simulations.

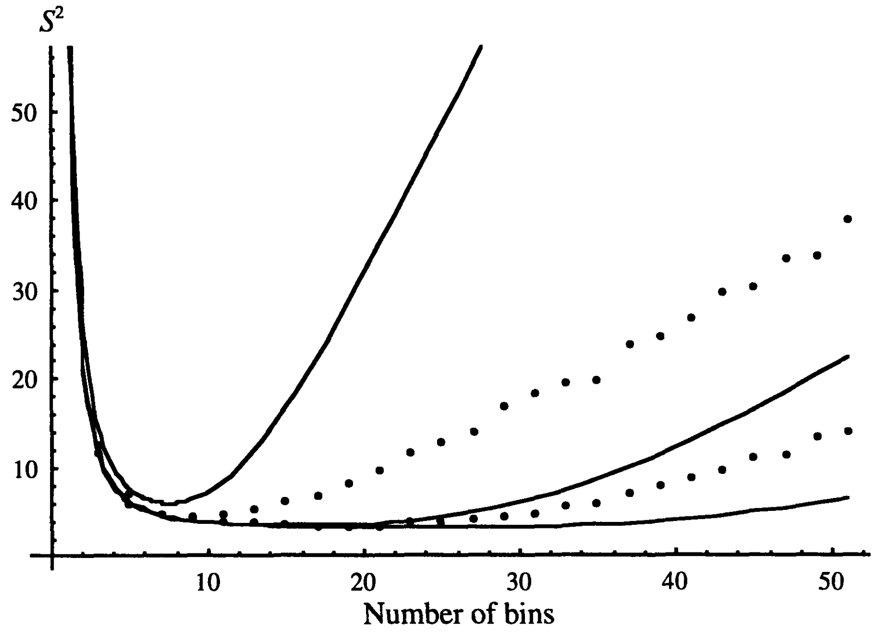


Figure 2.10: S^2 vs. Bin Number - First Model, Normal Distribution

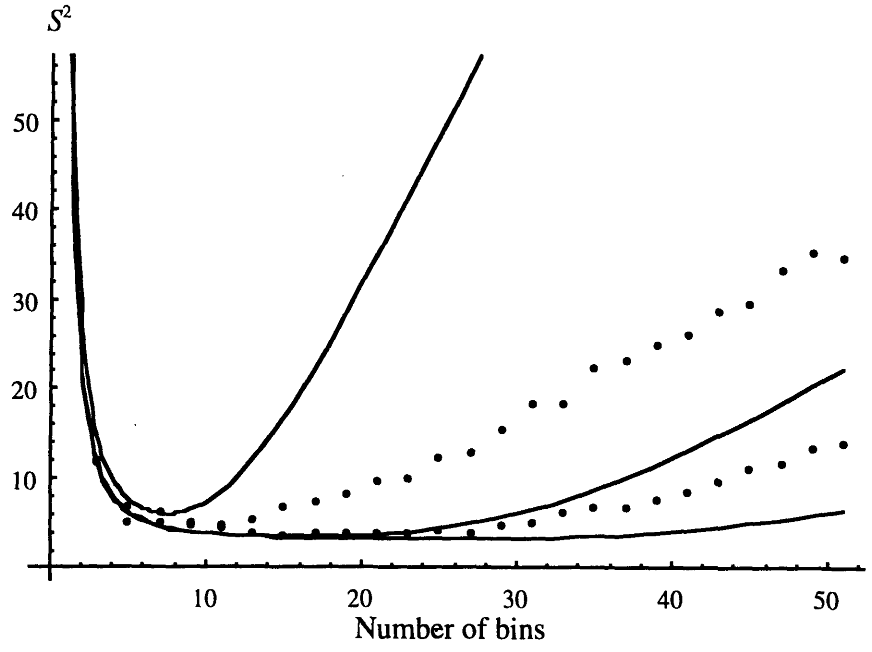


Figure 2.11: S^2 vs. Bin Number - Second Model, Normal Distribution

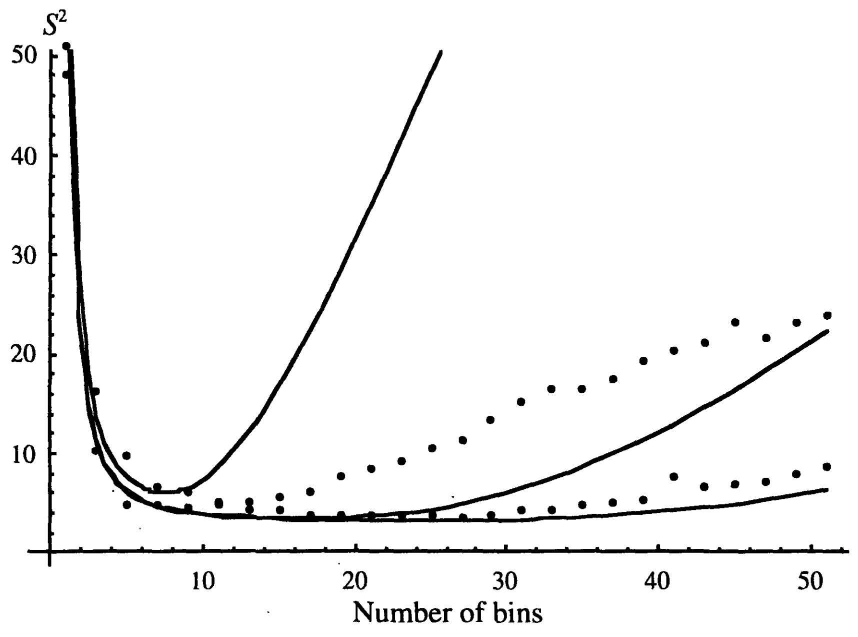


Figure 2.12: S^2 vs. Bin Number - Third Model, Normal Distribution

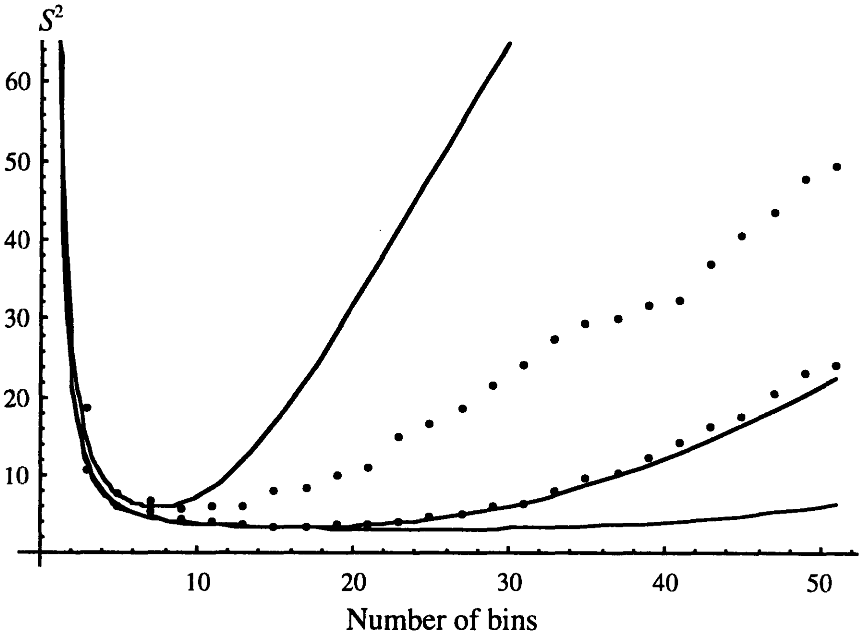


Figure 2.13: S^2 vs. Bin Number - Fourth Model, Normal Distribution

As expected, the observed error of the equal-width bins exceeds that predicted under the uniform distribution assumption. This is because too many bins are allocated to the higher input skew regions, in which comparatively few notes fall. The overall effect is like that of a static structure of fewer bins, applied to the lower-skew/higher-probability region of input space. This explains the similar shapes of the exact prediction curve and the observed data.

When the static structure of equal-probability bins is used, performance closely matches that predicted under the linear model/uniform-input-distribution case. By choosing the number of bins based on that model, but placing them equiprobably, good results are obtained.

Chapter 3
Bin Splitting

3.1 Introduction

The static-structure analysis of Chapter 2 assumed that adjustment of control action in a single range (a "bin") had no effect on the control action in neighboring bins. No interpolation, or related form of nearest-neighbor averaging, was considered.

Interpolation and averaging can be regarded as methods for spreading the lessons learned from local training data to larger regions of the control mapping. In this chapter, another approach to extending the realm of applicability of each training iteration is proposed. We call this alternative scheme "bin splitting." In this method, a single wide bin receives initial training. When a sufficient amount of learning has occurred, the wide bin splits into two narrower bins. Initially, the newly-split bins use the control action that was associated with the single large bin; this makes the analysis tractable. The process is shown in Figure 3.1.

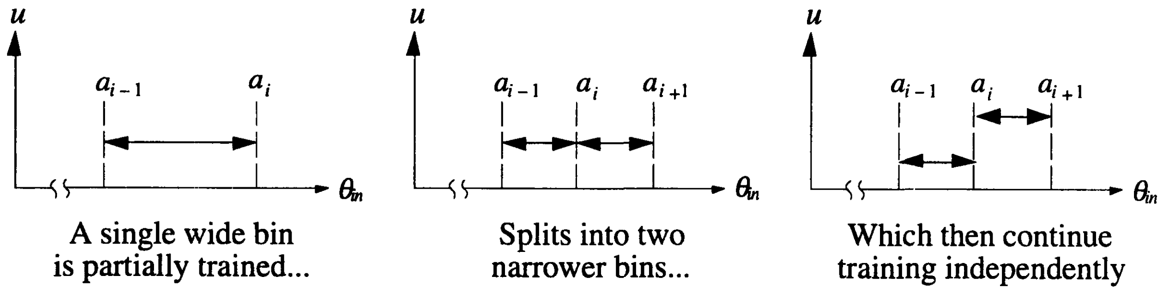

u
a_{i-1}
a_i
θ_{in}
A single wide bin is partially trained...
u
a_{i-1}
a_i
a_{i+1}
θ_{in}
Splits into two narrower bins...
u
a_{i-1}
a_i
a_{i+1}
θ_{in}
Which then continue training independently

Figure 3.1: Bin Splitting

This procedure uses early training samples to improve the control mapping over a large region of the input space. The splitting then allows refinement of the more narrowly-drawn regions. For an important special case, results have been obtained on the question of when to split from a single bin into two smaller bins, each half the width of the input space. In the examples tested, bin splitting yields better performance than a fixed structure, of either a single wide bin or two smaller bins, throughout the training process. The analysis is then extended to address the general question of when to split two bins into four, four into eight, eight into sixteen, *ad infinitum.* A surprising result is that all splitting times are independent of the total number of allowable training iterations.

Another way of viewing the premise of bin splitting is as follows. For a given number of notes entering the multi-bin system, we intuitively expect that wider bins' output error should decrease more quickly than the error of narrower bins. While the wider bins will have a larger error than narrower bins if both are perfectly trained, the wider bins should asymptotically approach their potential minimum error more quickly.

Consider a situation in which we train two systems from a state of no knowledge. The first system consists of a single wide bin covering the entire input range, and the second consists of two bins, dividing the input range evenly. Figure 3.2 shows graphs of (2.4-22), expected average squared output, as a function of N, with b set to one and two bins. Parameter values used in evaluating the equation are representative of those found in the deskewer.

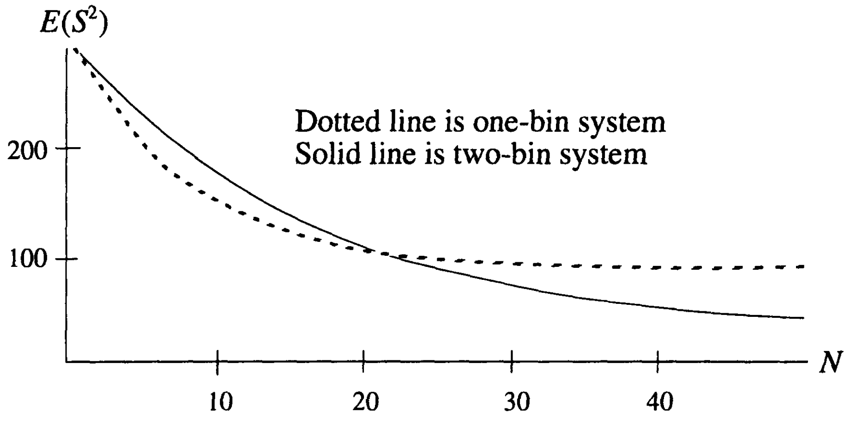


Figure 3.2: Evolution of Error for One and Two-Bin Systems

The initial rate of decrease in error is greater in the single-bin system, but the error quickly reaches an asymptotic value that is larger than that eventually achieved by the two-bin system. If the controller starts with a single bin to take advantage of the rapid decrease in error, and then moves to two bins to reach the lower-error asymptote, we expect to achieve better performance (for a given number of training iterations) than is possible with a static structure of either one or two bins. This is shown in Figure 3.3.

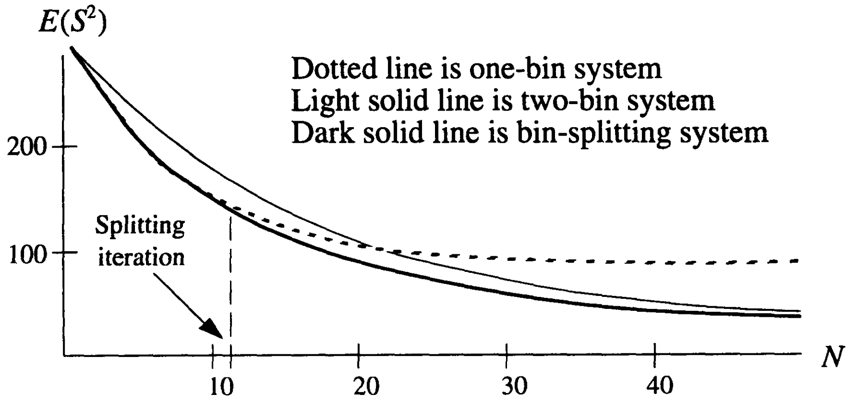


Figure 3.3: Changing the Number of Bins in Mid-Training

3.2 Necessary Results from Chapter 2

All results in this chapter again refer to the uniform input distribution model. By solving (2.3-7), we find that for a single bin of width w:

$$E(S^2) = \frac{1}{12}\beta_1^2 w^2 + \sigma^2 + E(\mu^2) \qquad (3.2\text{-}1)$$

where all parameters are as defined previously. $E(S^2)$ is treated as if it were the entire system under consideration, with 100% of all "interesting" notes falling into that single bin; that is why no normalization factors appear in (3.2-1).

From (2.3-8), we see that for this single bin:

$$\sigma_{bin}^2 = \frac{1}{12}\beta_1^2 w^2 + \sigma^2 \qquad (3.2\text{-}2)$$

For a bin that has probability P of a note falling into it, the following expression holds (equation (2.2-24) of the Bernoulli-process analysis):

$$E(\mu^2) \le \left(1 - P + P\phi\right)^N E(\mu_{n=0}^2) + \frac{\left(1 - \left(1 - P + P\phi\right)^N\right)}{\left(1 - \phi\right)} \frac{v^2}{\gamma^2} \sigma_{bin}^2 \qquad (3.2\text{-}3)$$

P is clearly equal to w/W in the uniform input distribution case (where W is the input space of the entire problem). N is the total number of training iterations to which the entire multi-bin system has been subjected; the fact that only a fraction of N has fallen into a particular bin is accounted for by the terms involving P.

Starting with the results stated above we can find a bin splitting algorithm. In the remainder of this work, the inequality in (3.2-3) will be written as an equality denoting the upper-bound on system error; the analysis in this chapter again optimizes the controller structure to minimize the upper bound. The source of this distinction is identical to that found in Chapter 2.

3.3 Notation

In this chapter, we will discuss the splitting of a single bin into a number of smaller sub-bins. For maximum generality, the single bin that we start with will again be treated as the entire system under consideration; the splitting process will serve to minimize the upper bound on error in that region. Within the initial region of interest, we assume a uniform input distribution, but this need not hold in adjacent regions. Other regions that start as a single bin can split concurrently, using an identical analysis, but with more or less-thinly-spread uniform distributions. In this way, a series of piecewise-constant, "stepped" regions can approximate an arbitrary input distribution, assuming each uniform-distribution region is covered initially by a single bin.

This algorithm given here assumes that all bins of a given "generation" split at the same time. That is, the system moves from two bins to four bins at a single value of n; there is no time when there are three bins. Clearly, the algorithm allows only powers of two as potential quantities of bins. This restriction on splitting freedom results in considerable analytical simplification (see Appendix E), and allows us to proceed without explicit estimation of errors in each individual bin (as will be seen below).

In the remainder of this chapter, the following four pieces of notation will be used:

1. For a specific bin, after a given number of iterations and bin splittings, $E(\mu^2)$ will be denoted by $E_{a,b,c}$. Here, a is the "generation number," b is the "bin number," and c is the "note number." The generation number is the base-2 logarithm of the total number of bins. Two bins are in the first generation, and four bins in the second generation. The generation number is clearly equal to the number of bin splits that have been performed. The bin number denotes specific bins within a particular generation, with 1 being the leftmost bin in the input space, and 2^a being the rightmost bin for generation number a. The note number is the cumulative number of training notes that have fallen into the initial wide region of interest, regardless of generation number. It starts at zero, when no training has been performed, and ends at N, when all notes have been used.

For example, $E_{3,5,19}$ is the expected squared average output of the fifth bin from the small end of the input range, after three bins splits have taken place (third generation), and nineteen training notes have been used in the entire region of initial interest.

2. As seen in (3.2-3), the term $(1 - P + P\phi)$ occurs twice in the expression for $E(\mu_n^2)$. This quantity changes according to the size of the bin, and occurs frequently in the analysis, so we define:

$$\phi_i = (1 - P_i + P_i\phi_0) \tag{3.3-1}$$

where $\phi_0 = \phi = (1 - \lambda/\gamma)^2$, as defined in our previous work, and P_i is the probability of a note falling into a particular bin in the i-th generation. If we start with a single bin that covers the entire input range of the system, $P_i = 2^{-i}$.

3. n_i refers to the training note after which the i-th split occurs. For example, if we move from one bin to two after the fourth training note, $n_1 = 4$.

4. The i in $\sigma^2_{bin,\,i}$ refers to the σ^2_{bin} associated with generation i. It is identical in all bins of a given generation, because the bins are all of the same width in this analysis.

3.4 An Equivalent Representation of Output Error

This section proceeds from the results derived in Appendix E. Immediately after we decide to split a bin, but before any additional training notes are used, we can write two alternative representations of that bin's output. The first gives the bin output as if there were only a single bin, while the second expresses the output as the sum of two sub-bins. No additional training has been done, so the expected average squared output should be identical. The different representations are only a notational distinction.

Using the notation introduced in the previous section, the output of a single bin is:

$$E(S^2) = \frac{1}{12}\beta_1^2 w^2 + \sigma^2 + E_{0,1,n_1} \tag{3.4-1}$$

Section 2.4.1 explained the rationale for choosing equal-width bins when working under the uniform input-distribution assumptions; Appendix E shows how this choice makes the splitting analysis tractable. For two bins of equal width:

$$\begin{aligned} E(S^2) &= \frac{1}{2}\left(\frac{1}{12}\beta_1^2\left(\frac{w}{2}\right)^2 + \sigma^2 + E_{1,1,n_1}\right) + \frac{1}{2}\left(\frac{1}{12}\beta_1^2\left(\frac{w}{2}\right)^2 + \sigma^2 + E_{1,2,n_1}\right) \\ &= \frac{1}{48}\beta_1^2 w^2 + \sigma^2 + \frac{1}{2}\left(E_{1,1,n_1} + E_{1,2,n_1}\right) \end{aligned} \tag{3.4-2}$$

Equations (3.4-1) and (3.4-2) are equal, as they are simply two different ways of writing the same system output. Equating them:

$$E_{0,1,n_1} + \frac{1}{16}\beta_1^2 w^2 = \frac{1}{2}\left(E_{1,1,n_1} + E_{1,2,n_1}\right) \tag{3.4-3}$$

This process of equating two system representations is used repeatedly in the remainder of this chapter.

3.5 Optimization of the First Split

The first split can now be analyzed. After this split, we have $N - n_1$ notes left with which to train the controller. The performance of the system at the end of all training can be written as:

$$E(S^2) = \frac{1}{48}\beta_1^2 w^2 + \sigma^2 + \frac{1}{2}\left(E_{1,1,N} + E_{1,2,N}\right) \tag{3.5-1}$$

However, this newly-split system has only $N - n_1$ notes for training, so $N - n_1$ appears in place of N in (3.2-3). This yields:

$$\frac{1}{2}\left(E_{1,1,N} + E_{1,2,N}\right) = \phi_1^{N-n_1}\frac{1}{2}\left(E_{1,1,n_1} + E_{1,2,n_1}\right) + \frac{\left(1 - \phi_1^{N-n_1}\right)}{\left(1 - \phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2 \tag{3.5-2}$$

But we can insert (3.4-3) into (3.5-2), and substitute the result back into (3.5-1):

$$E(S^2) = \frac{1}{48}\beta_1^2 w^2 + \sigma^2 + \phi_1^{N-n_1}\left(E_{0,1,n_1} + \frac{1}{16}\beta_1^2 w^2\right) + \frac{\left(1-\phi_1^{N-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2 \quad (3.5\text{-}3)$$

From (3.2-3), we can replace $E_{0,1,n_i}$ in (3.5-3):

$$E(S^2) = \frac{1}{48}\beta_1^2 w^2 + \sigma^2 + \phi_1^{N-n_1}\left(\phi_0^{n_1}E_{0,1,0} + \frac{(1-\phi_0^{n_1})}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2\right) + \frac{\left(1-\phi_1^{N-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2$$

(3.5-4)

This is identical to the final result in Appendix E. Differentiating (3.5-4) with respect to n_1 to find the optimal splitting time:

$$\begin{aligned}\frac{\partial E(S^2)}{\partial n_1} = &- \phi_1^{N-n_1}\ln\left(\phi_1\right)\left(\phi_0^{n_1}E_{0,1,0} + \frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2\right)\\ &+ \phi_1^{N-n_1}\ln\left(\phi_0\right)\left(\phi_0^{n_1}E_{0,1,0} - \frac{\phi_0^{n_1}}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2\right) + \ln\left(\phi_1\right)\frac{\phi_1^{N-n_1}}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2\end{aligned} \quad (3.5\text{-}5)$$

Factoring (3.5-5):

$$\begin{aligned}\frac{\partial E(S^2)}{\partial n_1} = \phi_1^{N-n_1}\Bigg(&-\ln\left(\phi_1\right)\left(\phi_0^{n_1}E_{0,1,0} + \frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2 - \frac{1}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2\right)\\ &+ \ln\left(\phi_0\right)\left(\phi_0^{n_1}E_{0,1,0} - \frac{\phi_0^{n_1}}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2\right)\Bigg)\end{aligned} \quad (3.5\text{-}6)$$

Since ϕ_1 is always greater than zero, (3.5-6) can only equal zero if the terms inside the outermost parentheses also equal zero. By finding the n_1 at which this occurs, we minimize the expected error. If $1 < n_1 < (N - 1)$, we know that by splitting, the error is less than if we had kept a fixed structure of either one or two bins for the entire training process. The question of the existence of such a zero is not addressed here; if one does not exist for $1 < n_1 < (N - 1)$, no splitting should be performed.

An interesting result in (3.5-6) is that the terms in the outermost parentheses (in which we seek the zero) do not depend on N, which only appears in the exponent of ϕ_1. While this may seem counterintuitive, a re-examination of Figures 3.2 and 3.3 confirms the plausibility of this result. The optimal splitting time yields the best results for all future iterations, regardless of how many are performed post-splitting.

3.6 The Second Split

When splitting a second time, we ask the following question: given a two-bin structure with initial conditions (starting $E(\mu^2)$) identical to those present at the time of the first split (from one bin to two), is there some iteration at which we can switch to four bins and reduce expected error for all operation after the end of the training period? We take the results of the single-bin training period as a starting point for this analysis, and hope to improve future performance by appropriately choosing the two-bin to four-bin splitting time.

First we write the equivalent representations for the two and four-bin structures, before any training has been performed with four bins:

$$\frac{1}{192}\beta_1^2 w^2 + \sigma^2 + \frac{1}{4}\left(E_{2,1,n_2} + E_{2,2,n_2} + E_{2,3,n_2} + E_{2,4,n_2}\right) = \frac{1}{48}\beta_1^2 w^2 + \sigma^2 + \frac{1}{2}\left(E_{1,1,n_2} + E_{1,2,n_2}\right) \tag{3.6-1}$$

Thus:

$$\frac{1}{4}\left(E_{2,1,n_2} + E_{2,2,n_2} + E_{2,3,n_2} + E_{2,4,n_2}\right) = \frac{1}{2}\left(E_{1,1,n_2} + E_{1,2,n_2}\right) + \frac{1}{64}\beta_1^2 w^2 \tag{3.6-2}$$

Using (3.2-3), we proceed as we did in (3.5-2) for the two bin case:

$$\frac{1}{4}\left(E_{2,1,N} + E_{2,2,N} + E_{2,3,N} + E_{2,4,N}\right) = \frac{\phi_2^{N-n_2}}{4}\left(E_{2,1,n_2} + E_{2,2,n_2} + E_{2,3,n_2} + E_{2,4,n_2}\right) + \frac{\left(1-\phi_2^{N-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,2}^2 \tag{3.6-3}$$

Inserting (3.6-2) into (3.6-3):

$$\frac{1}{4}\left(E_{2,1,N} + E_{2,2,N} + E_{2,3,N} + E_{2,4,N}\right) = \phi_2^{N-n_2}\left(\frac{1}{2}\left(E_{1,1,n_2} + E_{1,2,n_2}\right) + \frac{1}{64}\beta_1^2 w^2\right) + \frac{\left(1-\phi_2^{N-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,2}^2 \tag{3.6-4}$$

Now we back-substitute the equalities found in the two-bin case, which leaves the four-bin equations dependent on only the initial value $E_{0,1,0}$. In this way, the errors in the individual split bins do not need to be tracked or estimated. Apart from the analytical simplifications afforded by simultaneous splitting, the elimination of tracking is the main motivation for requiring all bins to split at the same iteration. From (3.2-3) and (3.5-2) we have:

$$\frac{1}{2}\left(E_{1,1,n_2} + E_{1,2,n_2}\right) = \phi_1^{n_2-n_1}\frac{1}{2}\left(E_{1,1,n_1} + E_{1,2,n_1}\right) + \frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2 \tag{3.6-5}$$

Inserting (3.6-5) into (3.6-4):

$$\frac{1}{4}\left(E_{2,1,N}+E_{2,2,N}+E_{2,3,N}+E_{2,4,N}\right)= \tag{3.6-6}$$

$$\frac{\left(1-\phi_2^{N-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,2}+\phi_2^{N-n_2}\left(\phi_1^{n_2-n_1}\frac{1}{2}\left(E_{1,1,n_1}+E_{1,2,n_1}\right)+\frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,1}+\frac{1}{64}\beta_1^2w^2\right)$$

Substituting (3.4-3) into (3.6-6):

$$\frac{1}{4}\left(E_{2,1,N}+E_{2,2,N}+E_{2,3,N}+E_{2,4,N}\right)= \tag{3.6-7}$$

$$\frac{\left(1-\phi_2^{N-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,2}+\phi_2^{N-n_2}\left(\phi_1^{n_2-n_1}\left(E_{0,1,n_1}+\frac{1}{16}\beta_1^2w^2\right)+\frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,1}+\frac{1}{64}\beta_1^2w^2\right)$$

Inserting (3.2-3) into (3.6-7) gives:

$$\frac{1}{4}\left(E_{2,1,N}+E_{2,2,N}+E_{2,3,N}+E_{2,4,N}\right)=\frac{\left(1-\phi_2^{N-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,2} \tag{3.6-8}$$

$$+\phi_2^{N-n_2}\left(\phi_1^{n_2-n_1}\left(\phi_0^{n_1}E_{0,1,0}+\frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,0}+\frac{1}{16}\beta_1^2w^2\right)+\frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,1}+\frac{1}{64}\beta_1^2w^2\right)$$

Inserting (3.6-8) into (3.6-1) gives the final expression for expected average squared output:

$$E(S^2)=\frac{1}{192}\beta_1^2w^2+\sigma^2+\frac{\left(1-\phi_2^{N-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,2} \tag{3.6-9}$$

$$+\phi_2^{N-n_2}\left(\phi_1^{n_2-n_1}\left(\phi_0^{n_1}E_{0,1,0}+\frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,0}+\frac{1}{16}\beta_1^2w^2\right)+\frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,1}+\frac{1}{64}\beta_1^2w^2\right)$$

We now differentiate (3.6-9) with respect to both splitting points, and factor:

$$\frac{\partial E(S^2)}{\partial n_1}=\phi_2^{N-n_2}\phi_1^{n_2-n_1}\left(-\ln(\phi_1)\left(\phi_0^{n_1}E_{0,1,0}+\frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,0}+\frac{1}{16}\beta_1^2w^2-\frac{1}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,1}\right)\right.$$

$$\left.+\ln(\phi_0)\left(\phi_0^{n_1}E_{0,1,0}-\frac{\phi_0^{n_1}}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma^2_{bin,0}\right)\right) \tag{3.6-10}$$

$$\frac{\partial E(S^2)}{\partial n_2} = \phi_2^{N-n_2}\left(-\ln(\phi_2)\left(\phi_1^{n_2-n_1}\left(\phi_0^{n_1}E_{0,1,0} + \frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2\right)\right.\right. \tag{3.6-11}$$

$$\left. + \frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2 + \frac{1}{64}\beta_1^2 w^2 - \frac{1}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,2}^2\right)$$

$$\left.\left. + \ln(\phi_1)\left(\phi_1^{n_2-n_1}\left(\phi_0^{n_1}E_{0,1,0} + \frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2\right) - \frac{\phi_1^{n_2-n_1}}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2\right)\right)$$

The earlier optimal splitting point, n_1, is independent of the later, n_2. Differentiation with respect to n_1 yields an equation identical to (3.5-5), with no dependence on N or n_2. Differentiation with respect to n_2 yields an equation identical in form to (3.5-5), but with a dependence on n_1. However, n_1 was already fixed in the differentiation for the first splitting point, so n_2 can be found immediately (if a root of $\frac{\partial E(S^2)}{\partial n_2}$ exists when $\frac{\partial E(S^2)}{\partial n_1}$ is zero). This pattern holds for later splits; the next splitting iteration can always be found from a single equation once all earlier splitting points are known.

3.7 The General Case

For the sake of brevity, a general expression will not be presented, but an iterative process for deriving the splitting-point equation is shown in the following derivation of the expression for the third split. First, we write the dual representation of output for the two generations at the moment of splitting, and then find the relationship between their E. This is:

$$\frac{1}{8}\sum_{\alpha=1}^{8} E_{3,\alpha,n_3} = \frac{1}{4}\sum_{\delta=1}^{4} E_{2,\delta,n_3} + \frac{1}{256}\beta_1^2 w^2 \tag{3.7-1}$$

Next, we look at the previous generation's expression for $\frac{1}{4}\sum_{\delta=1}^{4} E_{2,\delta,n_3}$, which can be found from (3.6-8). By substituting n_2 where N appears in (3.6-8):

$$\frac{1}{4}\sum_{\delta=1}^{4} E_{2,\delta,n_3} = \frac{\left(1-\phi_2^{n_3-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,2}^2 \tag{3.7-2}$$

$$+\phi_2^{n_3-n_2}\left(\phi_1^{n_2-n_1}\left(\phi_0^{n_1}E_{0,1,0} + \frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2\right) + \frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2 + \frac{1}{64}\beta_1^2 w^2\right)$$

By inserting (3.7-2) into (3.7-1), the expression for $\frac{1}{8}\sum_{\alpha=1}^{8} E_{3,\alpha,n_3}$ is found. We then use this to find the expected output, but we will also use it when we write the expected output for the next generation (just as we used (3.6-8) here). The complete expression is:

$$E(S^2) = \frac{1}{768}\beta_1^2 w^2 + \sigma^2 + \frac{\left(1-\phi_3^{N-n_3}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,3}^2 + \phi_3^{N-n_3}\left(\frac{\left(1-\phi_2^{n_3-n_2}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,2}^2 + \frac{1}{256}\beta_1^2 w^2\right.$$
$$\left. + \phi_2^{n_3-n_2}\left(\phi_1^{n_2-n_1}\left(\phi^{n_1}E_{0,1,0} + \frac{\left(1-\phi_0^{n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,0}^2 + \frac{1}{16}\beta_1^2 w^2\right) + \frac{\left(1-\phi_1^{n_2-n_1}\right)}{\left(1-\phi_0\right)}\frac{v^2}{\gamma^2}\sigma_{bin,1}^2 + \frac{1}{64}\beta_1^2 w^2\right)\right)$$

(3.7-3)

The process outlined here is iterative, and allows simple calculation of the output of the next generation's structure from expressions derived for the current generation. The general expression for an arbitrary generation can be obtained through this procedure. Optimization is again performed by taking derivatives of the n's; as before, earlier optimal splitting points are not affected by those that come later.

The derivation, being iterative, is amenable to numerical implementation; we need never write the general expression. All plots of theoretical predictions shown in the next section were implemented with a recursive function definition in *Mathematica*. The definition (but not the graphics code) is found in Section F.3, under the heading *Functions for optimization of splitting times*.

3.8 Theoretical Results

In all of the plots below, the system is trained from a state of uniformly-zero control action, and starts with a single wide bin that covers the entire input range. While this may not represent an actual, practical application of the splitting method, it does allow a standard basis of comparison for different techniques. For a system like the deskewer, we use the following parameters: $\lambda = 0.1$, $v = 0.5$, $\gamma = 5$, $\beta_1 = 1$, $\sigma = 1.75$, $W = 30$, and $N = 500$ (chosen because the first four splits of the deskewer-like system occur within 500 iterations). We first show a graph of expected output as a function of number of iterations for systems in which the bin structure is static (Figure 3.4). Clearly, more bins do not necessarily yield better performance.

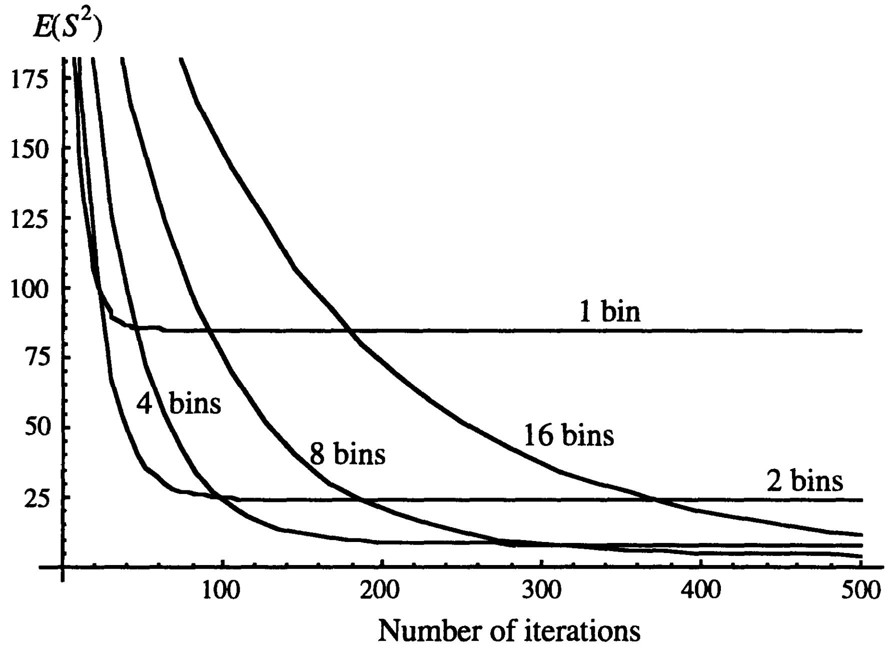


Figure 3.4: Error vs. Iterations for 1, 2, 4, 8, 16-Bin Static Systems

Figure 3.5 is a plot of the output error after the 50th iteration, as a function of the single splitting iteration, in a system which goes from one to two bins. The minimum is found at $n_1 = 10.46$, which must be rounded to an integer in practice; we have here reduced the total number of iterations to 50, to more clearly show the existence of a minimum:

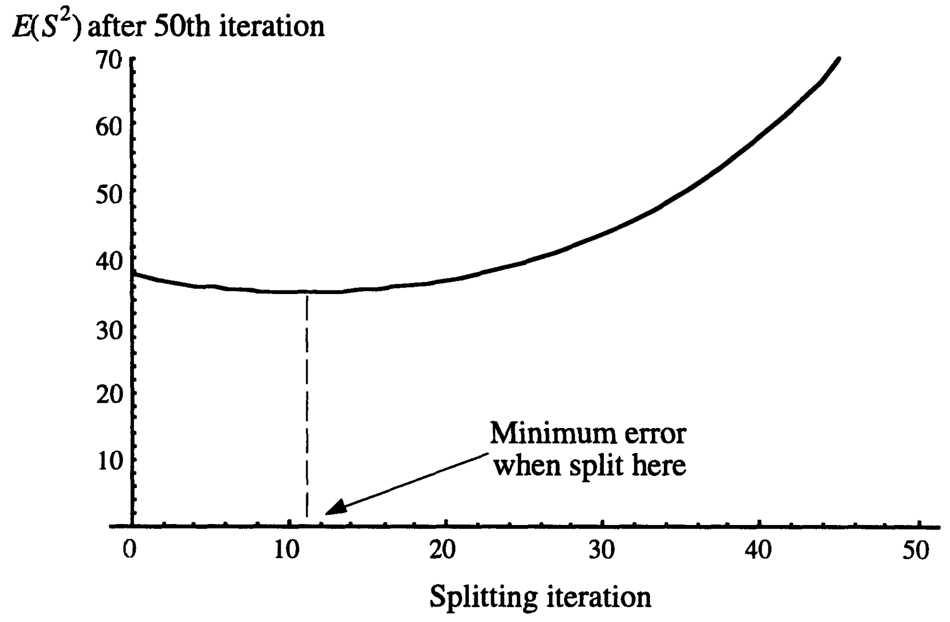


Figure 3.5: Error After 50 Iterations vs. Splitting Iteration (1 to 2 Bins)

Figure 3.6 duplicates the situation of Figure 3.5, but with N restored to its original test value of 500. Close examination shows that the minimum is still at the same place (between 10 and

11), but that practically, the splitting time is almost unimportant. The only significant effect comes from splitting too late, and thus not allowing sufficient training time for the two bins. Note also that the $N = 500$ case has much lower overall error than the $N = 50$ case, so even though the splitting point is the same, the additional training notes allow greatly improved system performance.

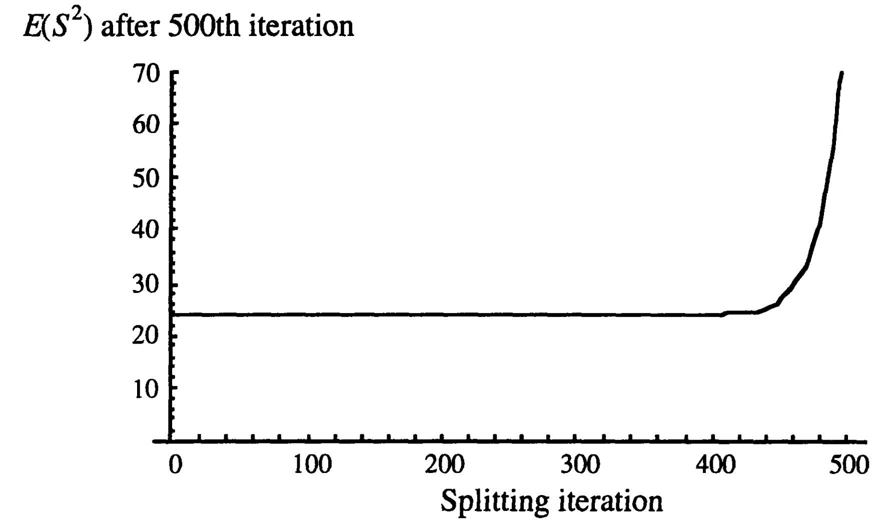


Figure 3.6: Error After 500 Iterations vs. Splitting Iteration (1 to 2 Bins)

Figure 3.7 is a two-dimensional plot of output error after the 100th iteration as a function of the first two splitting iterations; at the end of the process, there are a total of four bins. As expected, the optimal value for n_1 is the same as in the one-to-two bins case; n_2 is optimized at 45.5. While the plotted region focuses on the area of the minimum error, examination of larger regions shows there is only one minimum.

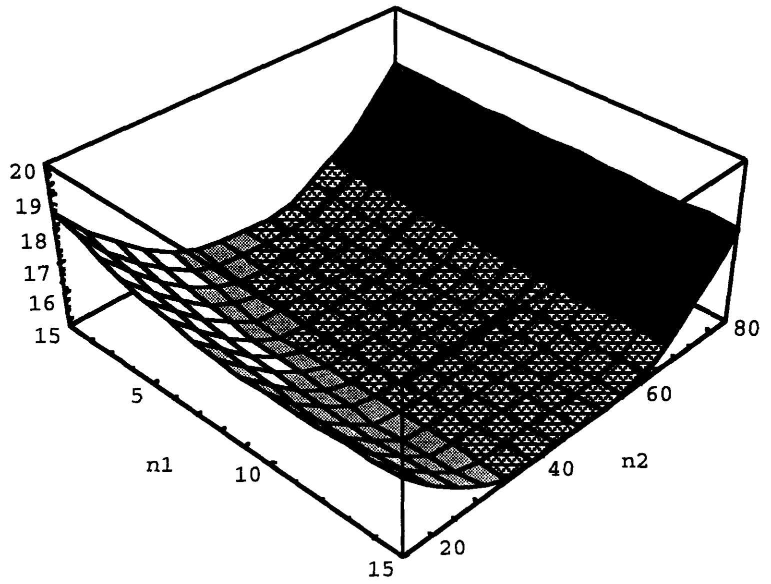


Figure 3.7: Error After 100 Iterations vs. First Two Splitting Iterations

Continuing the analysis through the first four splits, we find that the optimal splitting times for n_3 and n_4 are 115.8 and 257.8, respectively. Figure 3.8 shows a plot of expected average squared output as a function of number of training iterations, when all splitting occurs at the optimal points. For comparison, the plot of the output of the optimal static system (which consists of 9 bins when $N = 300$) is overlaid. The splitting structure is clearly superior.

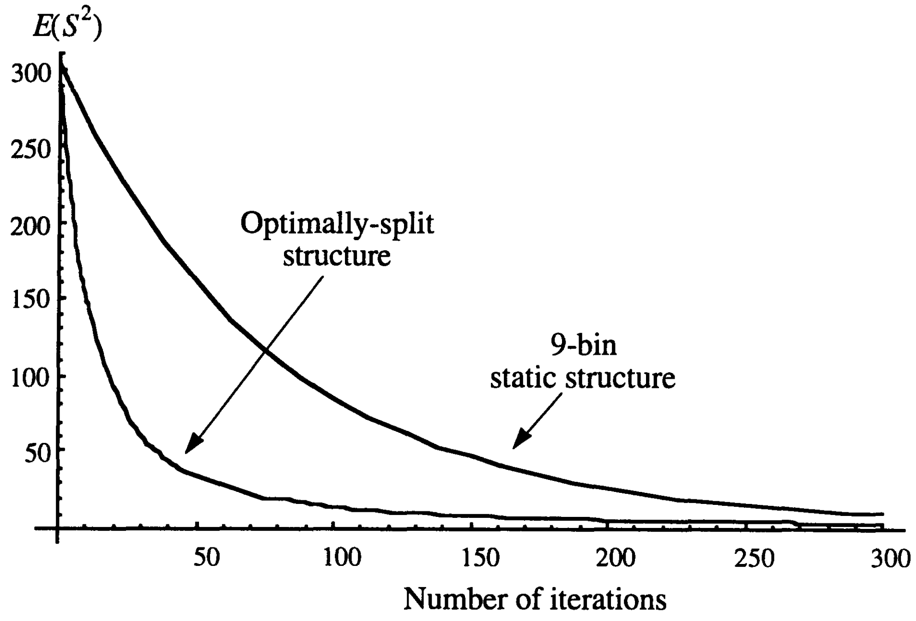


Figure 3.8: Error vs. Iterations - Optimal Splitting and Static Structures

Figure 3.9 presents the ratio of the expected average squared output of an optimally split controller to that of the optimal static structure at a given final iteration. The splits occur at 11, 47, 116, 258, 541, and 1117 iterations, while the number of static bins varies with the iteration (because the optimal number of bins depends on the final iteration). At each integer iteration value, the optimal number of bins in the static structure is found and, as it is generally fractional, rounded to the nearest integer. This integer number of bins is then used to re-evaluate the expected output error of the static structure. The rounding of bin numbers accounts for the sudden jumps in the plot.

As the final iteration increases, we expect the benefit of a split structure to tail off, and the ratio to go to unity. No matter how many bins are used, the performance of the controller is still limited by the inherent noise in the system. This trend is exhibited in Figure 3.9, but it is important to note that the error figure used is only a single-point value - the expected error for all time after the final training iteration. This is analogous to an L_∞ norm in real analysis. But, an advantage of splitting that is not quantified by Figure 3.9 can be seen by reinspecting Figure 3.8. The integrated error over the training period, like an L_2 norm, will at least tend towards a constant, and may increase unboundedly.

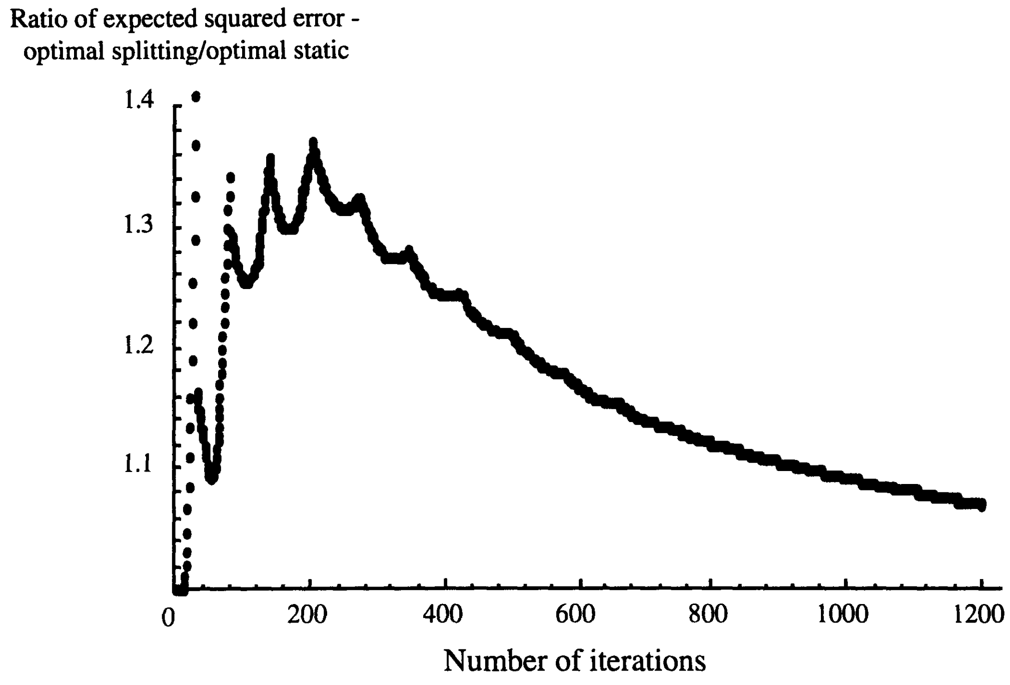


Figure 3.9: Ratio of Splitting Error to Static Error

3.9 Simulation Results

The bin splitting simulation code is presented in Section F.3. Unlike the static-structure simulations, in which N is fixed and the number of bins is varied, the desired results of the splitting simulations are graphs showing expected output error as a function of final iteration number, as in Figure 3.8.

Once again, the simulations begin from a state of no knowledge (uniformly zero control action); the deskewer is modeled with a given analytical expression; the parameters used are identical to those of the preceding section ($\lambda = 0.1$, $\nu = 0.5$, $\beta_1 = 1$, $\sigma = 1.75$, $W = 30$, and $a_0 = 0$); and γ is chosen to be 5.

For all simulations in this section, average squared output is evaluated after every 20 training notes by fixing control actions and feeding 200 test iterations through the simulation. The total number of training notes is 1200, and the entire process is repeated 20 times.

The progression of average squared output versus iteration number is also evaluated for a near-optimal static structure of 12 bins. The splitting schedule is identical to that used in Figure 3.8, while the static bin number was chosen through inspection of the results of the static-structure simulations of Chapter 2.

The two solid curves plotted in all figures below represent theoretical average squared output versus number of iterations for a completely linear system (obeying (2.5-1)), with a uniform input distribution of range 0° - 30°. The only simulation which matches these assumptions is that of Figure 3.10, but as in Chapter 2, we would like to see how the results from models that violate the analytical assumptions compare with theory. The upper curve, with greater error, is from the 12-bin static structure, while the lower curve comes from the splitting controller. In both curves, $\lambda = \nu = 0.3$.

In all simulations shown in this section, the splitting structure outperforms the static structure. The higher-error data (at a given iteration) is from the static simulations, while the lower-error data is from the splitting simulations.

Figures 3.10 through 3.13 show the results of simulations of the four system models of Section 2.5, respectively. The results in Figure 3.10 validate this chapter's analytical predictions.

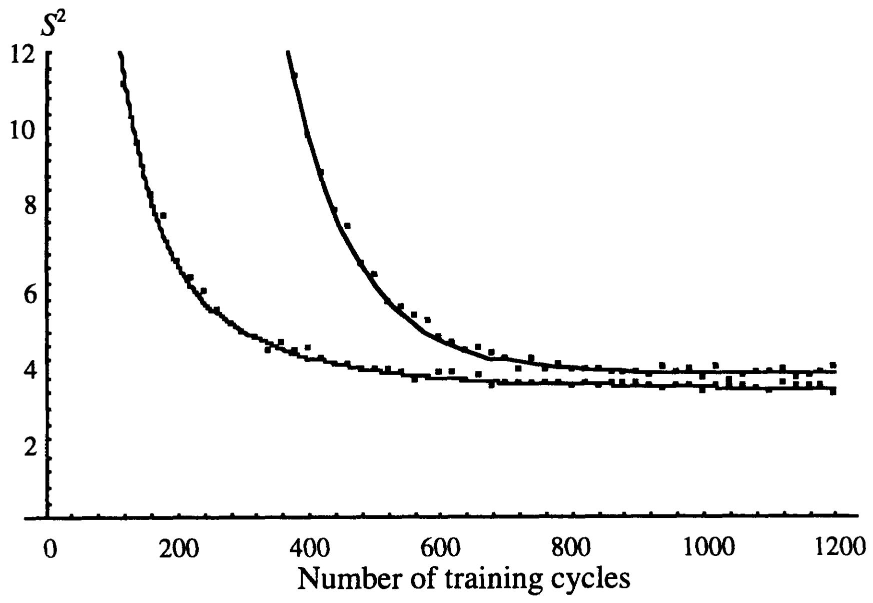


Figure 3.10: S^2 vs. Iteration Number - First Model, Uniform Distribution

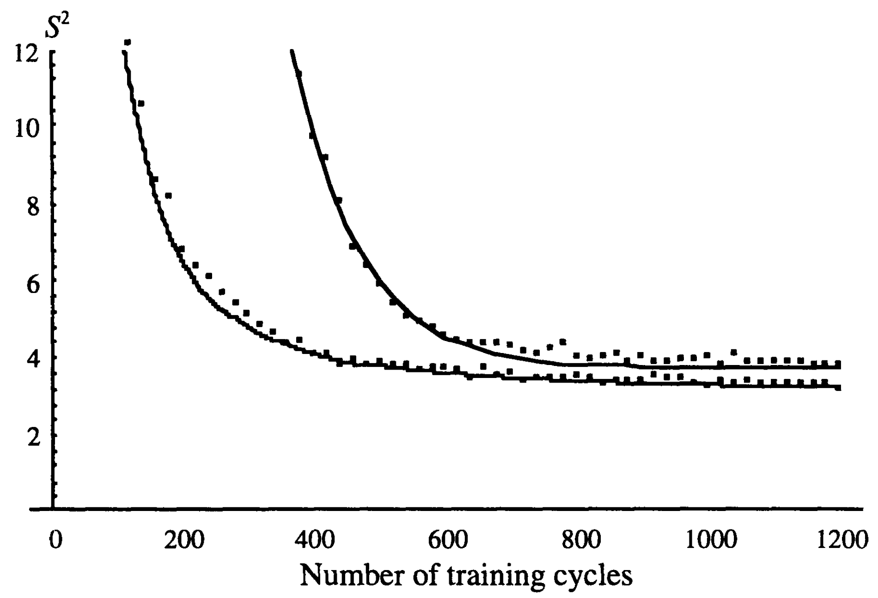


Figure 3.11: S^2 vs. Iteration Number - Second Model, Uniform Distribution

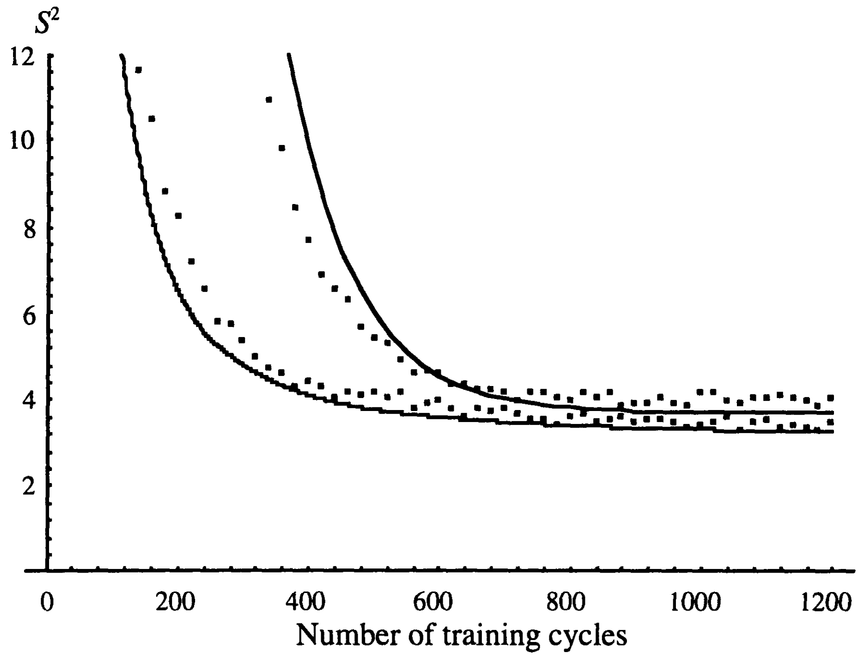


Figure 3.12: S^2 vs. Iteration Number - Third Model, Uniform Distribution

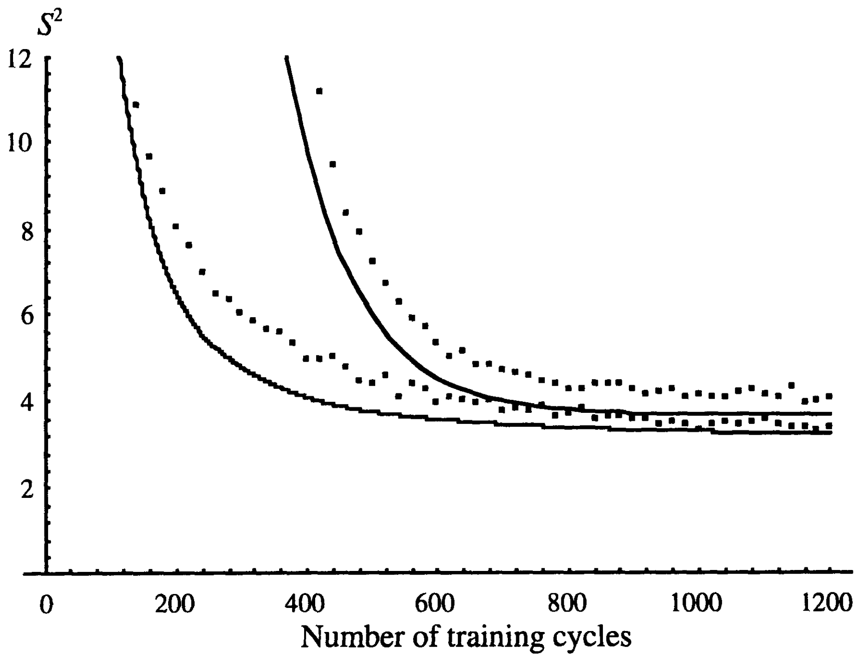


Figure 3.13: S^2 vs. Iteration Number - Fourth Model, Uniform Distribution

Figures 3.14 through 3.17 contain the results of simulations of the four models, but with a normal input distribution of 0° mean and 15° standard deviation. As in the simulations of Chapter 2, the tails of the distribution above 30° and below -30° are eliminated. All bins are of equal width. The observed output errors exceed those predicted, as seen in Chapter 2, because of the overly large number of bins in the area of high θ_{in}, but the splitting controller again outperforms the static controller.

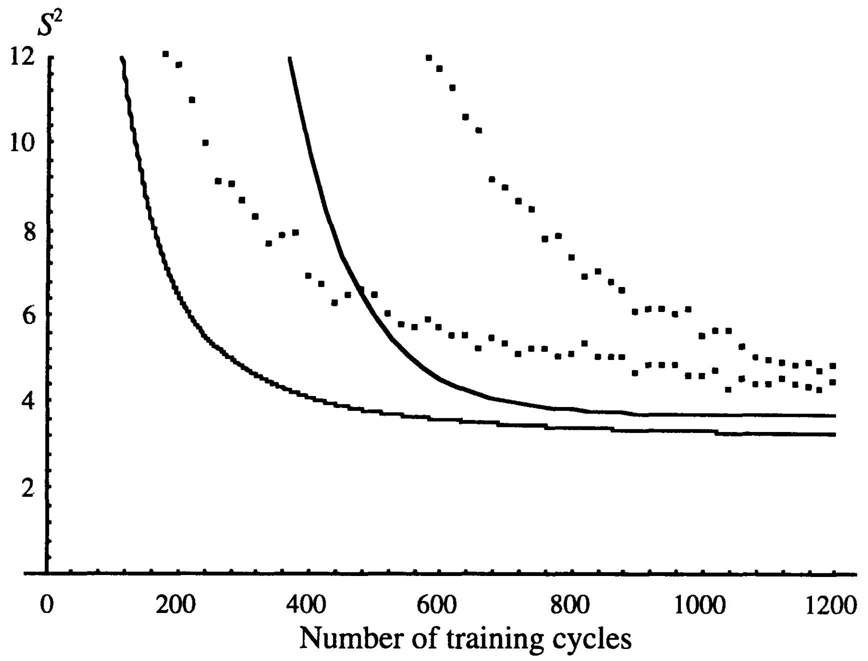


Figure 3.14: S^2 vs. Iteration Number - First Model, N.D., Equal-Width Bins

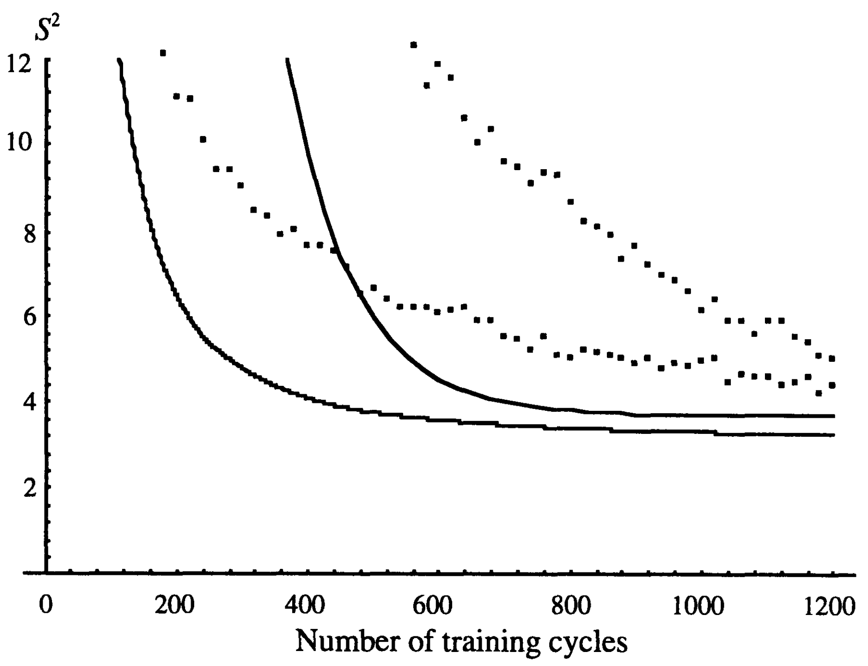


Figure 3.15: S^2 vs. Iteration Number - 2nd Model, N.D., Equal-Width Bins

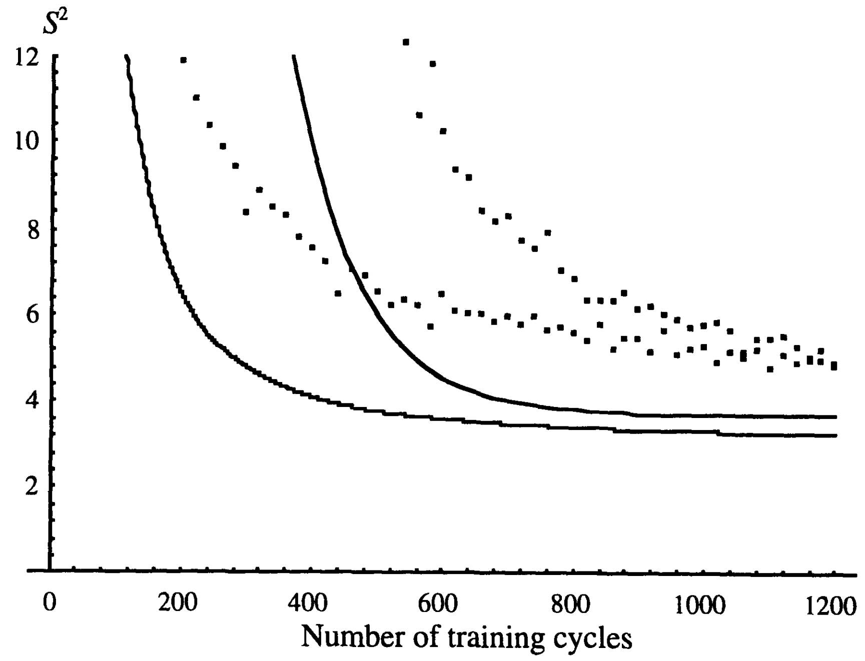


Figure 3.16: S^2 vs. Iteration Number - Third Model, N.D., Equal-Width Bins

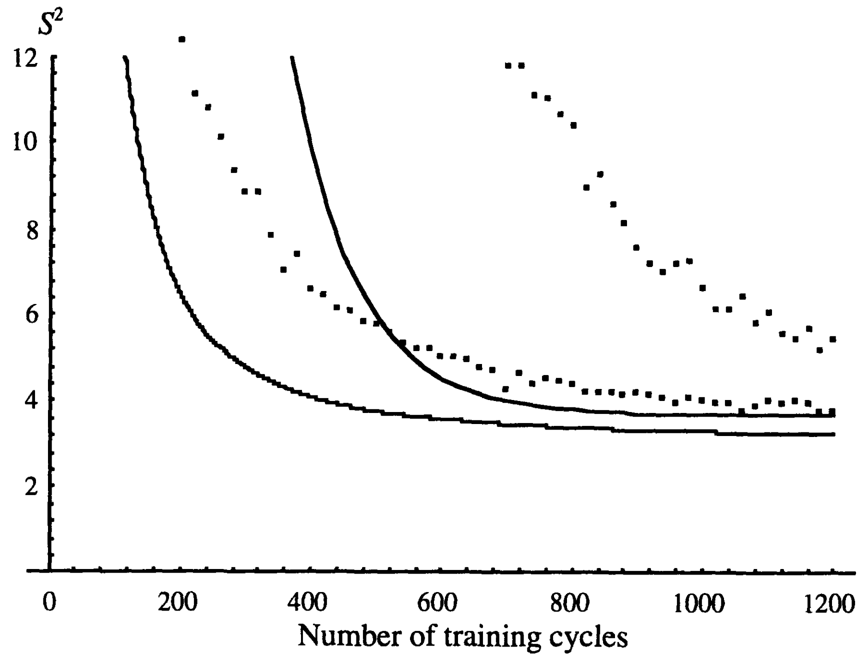


Figure 3.17: S^2 vs. Iteration Number - Fourth Model, N.D., Equal-Width Bins

Finally, the simulations are repeated with the same normal input distribution, but with an equiprobable bin structure. As seen in the previous chapter, the observed results have much lower error than the equivalent simulations with equal-width bins.

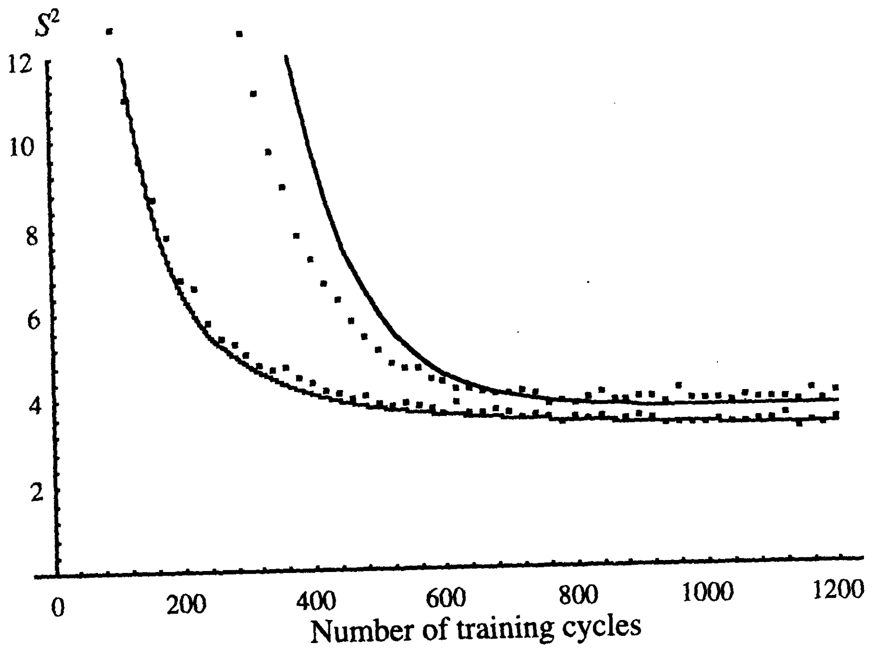


Figure 3.18: S^2 vs. Iteration Number - First Model, N.D., Equiprobable Bins

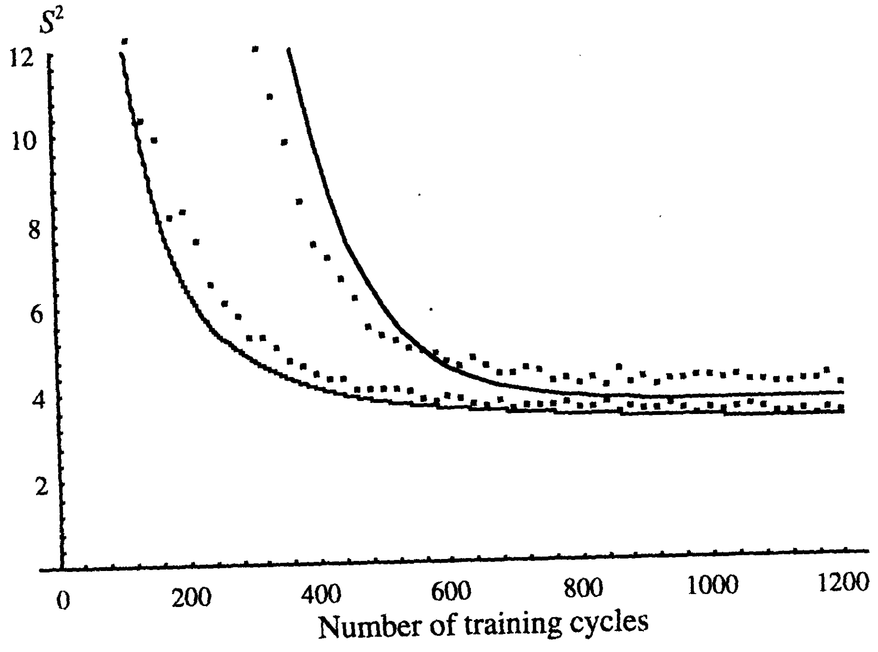


Figure 3.19: S^2 vs. Iteration Number - 2nd Model, N.D., Equiprobable Bins

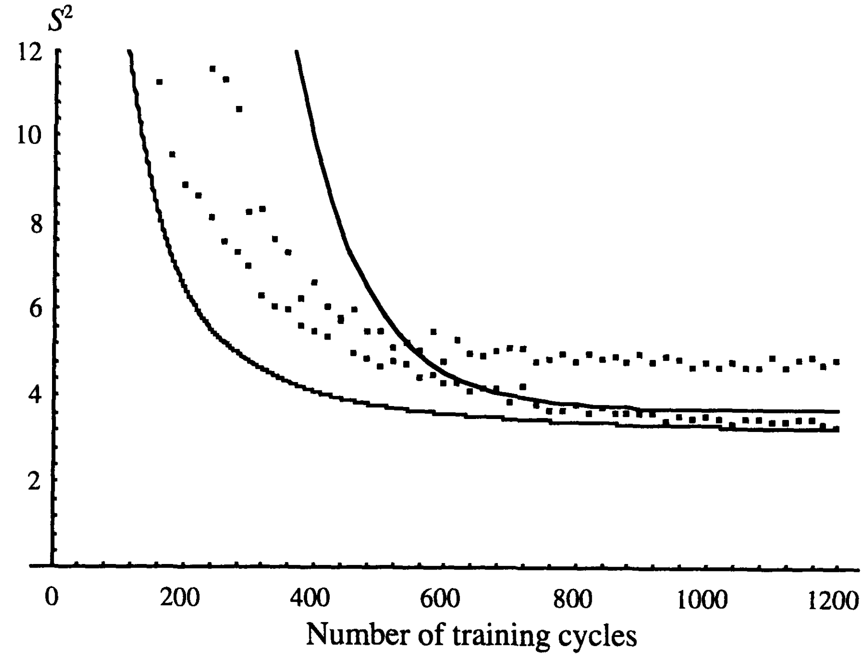


Figure 3.20: S^2 vs. Iteration Number - Third Model, N.D., Equiprobable Bins

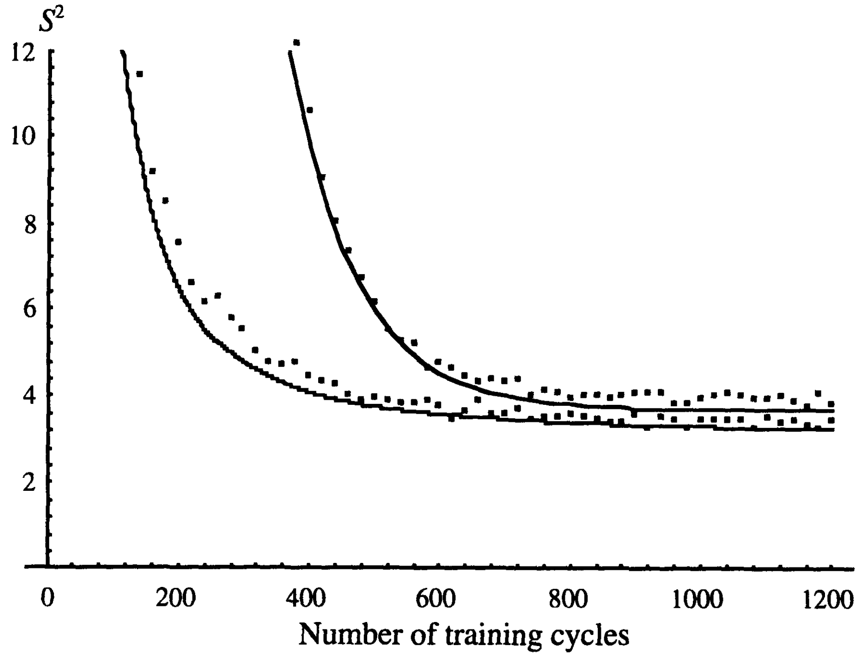


Figure 3.21: S^2 vs. Iteration Number - 4th Model, N.D., Equiprobable Bins

In the twelve simulations presented in this section, the bin splitting controller consistently yields better performance than an optimal static structure. While the exact amount of performance increase varies with the simulated model and the input distribution, this broadly-observed trend confirms the idea that splitting, with its application of early data to the entire input space, is more "sensible" than a static structure, with its strict segregation of training experiences.

Chapter 4
Conclusions and Future Directions

4.1 Conclusions

This thesis presents the analysis of the application of the self-tuning, open-loop, piecewise-constant controller to a class of control problems in which the control objective is to use a single application of constant control action to drive a system from a given initial state to a desired constant final state. The motivating example is the bill deskewer, but the analysis is applicable to other systems. The optimization goal is the minimization of expected average squared output, measured after the completion of training.

The first case examined is one in which the controller structure is fixed for the duration of training. No changes are allowed to the number or location of bins. For a simplified model of system performance, the optimal number and location of bins is found. The number of bins is determined both by the amount of available training data and by the method used to train the bins, while location is a function of the system model and the distribution of sample points in the input space. Simulation validates the analytical predictions, and shows that the qualitative predictions of the idealized analysis are correct even when the modeled system is generalized to include certain nonlinearities.

In order to broaden the range over which early training data is used, a bin splitting technique is next developed. A single wide bin is used at the beginning of training, followed by division of this bin into two halves. Each half continues training and is later split again, followed by further subdivisions. In this way, later training data is used in progressively-finer regions of the control mapping. For the same models examined in the static-structure analysis, a series of partially-coupled, structurally-identical nonlinear equations that determine the optimal splitting times is derived, and the splitting times are found to be independent of the number of training iterations available. For the system models examined, the splitting algorithm yields better results than any static configuration of equal-width bins, a finding that is confirmed with simulation. Adding plausible trial nonlinearities to the system model, in simulation, does not appear to reduce the advantage of the splitting technique.

While the example system used in this thesis is based on the deskewer, the analytical tools presented can be immediately applied to other systems by appropriate substitution of model parameters.

4.2 Extensions of the Static Structure Analysis

For the static structure of Chapter 2, there are five areas in which further research should be conducted.

First, the algorithm used to update control action is of central importance in the analyses of Chapters 2 and 3, because it determines the value of $E(\mu^2)$ as a function of training iterations. Alternate algorithms that have the merits of either converging more quickly, or requiring less information about the system (for example, not requiring knowledge of derivative bounds) should be considered.

Second, the next logical extension of the analytical results is to cases in which the input distributions are piecewise-linear. As some input distributions will yield non-integrable equations, a piecewise-linear approximation to the real distribution could be substituted, with the number of elements determined by the desired accuracy of approximation. An attempt at performing this work suggested that the resulting equations were likely to be tractable, but complicated. Numerical results could well be faster.

Third, a framework for numerically determining the optimal static structure for arbitrary input distributions has been presented, but the location of bins hinges on optimization of equation (2.3-11). This optimization problem needs additional investigation, and will likely need to be performed numerically.

Fourth, the linear model of variation of average output versus input (Assumption 2.3.2) could be enhanced. For the uniform-input-distribution case, an analytical solution may still be possible if the linear model is replaced by a higher-order polynomial. However, the effective choice of a polynomial model assumes a degree of detailed modeling information that may not be available for a given system.

Finally, the question remains of how to choose a controller structure when the main concern is not training from a state of no knowledge, but rather the updating of control action in an already-trained controller. If 1000 notes are available to retune the deskewer every morning, starting from existing control action is likely to yield better results than beginning anew with no knowledge. So, if the existing actions are the starting point, how many bins should be used?

For the uniform input distribution case, examination of equation (2.4-17) presents a possible answer. In that equation, the quantity $\sum_{i=1}^{b} E\left(\mu^2_{n=0,\,\text{bin}\,i}\right)$ needs to be determined as the last step in finding the expression for expected average squared output. When the controller is slightly mistuned, this quantity is nonzero. If, for an arbitrary number of bins, the "average" value of $\sum_{i=1}^{b} E\left(\mu_{n=0,\,\text{bin } i}^2\right)$ arising from "typical" misadjustment of the controller could be determined, this "average" value could be inserted into (2.4-17), and the optimal number of bins found. A similar technique could be easily used in the numerical solution of more general models.

Because the estimation of error arising from mistuning requires detailed knowledge about "typical" misadjustment, an analysis of this type was not carried out here.

4.3 Extensions of Bin Splitting

Because the bin splitting analysis is derived from that of the static structure, the comments of the preceding section generally apply here as well. However, there are three issues particular to the bin splitting case.

First, unlike the static-structure problem, no method is currently known for handling non-uniform input distributions, even numerically. These arbitrary distributions could, however, be approximated by a collection of uniform-distribution regions.

Second, while we expect that the bin splitting algorithm will always yield better performance than any static structure for a fixed number of iterations, there is no proof of this conjecture at this time. The analysis offers a straightforward way to check that this is the case for any particular system, however, so the lack of proof may be of mainly theoretical interest.

Finally, in order to eliminate the need for tracking of the bins' average outputs during the training process, the algorithm presented in Chapter 3 is restricted to powers-of-two numbers of bins, and identical splitting times for all bins in a given generation. Appendix E shows how the initial output error of sub-bins can be calculated, which allows complete freedom in splitting by treating each newly-created sub-bin as an isolated system. This possibility should be more fully explored.

An appropriate use of the splitting technique could be as a "second round" training procedure. For example, given a control system that has already been trained with some bin structure (either static, or one derived through bin splitting), a period of time could be allowed in which to collect data without altering control actions. This data could then be used to calculate actual (as opposed to expected) output errors, and based on these results, regions of high output error could be split.

Appendix A
Example Systems

There are other systems to which the self-tuning open-loop controller might be applied. This section briefly describes two of them, and notes the features they have in common with the deskewer.

The first example is a method for automated grinding of weld beads. We can construct a controller that examines a small section of unground weld, just as it is about to pass under the grinding wheel. This is analogous to a single note passing through the deskewer. The measured height of the oncoming section of bead serves as the input space, which is appropriately divided into bins; depending on our performance criterion, each bin holds one of two distinct types of control action. If the bead or wheel temperature is the limiting factor, and multiple passes are allowed, the controller sets grinder compliance. If the primary concern is one-pass grinding, the absolute position of the wheel over the surface is specified, and held as stiffly as possible with the available actuators. This binning process is repeated for successive lengths of bead.

The bead height of each section is then measured post-grind, and used to adjust the control action for all future grinds. In the position-specification case, any residual bead moves the wheel position associated with the input bin closer to the surface, while gouges in the unwelded area cause the adjustment algorithm to increase the gap between wheel and surface. Similarly, temperature rise or wheel kickback leads to increased compliance in the compliance-specification case. This does, of course, also mean that an increased number of passes is necessary to achieve sufficient material removal.

The second example concerns automatic volume control of a car radio in a situation with varying background noise. The measured performance is the listener's subjective sense of "loudness," where positive values are too loud, negative values are too quiet, and a value of zero is optimal.

The observation that "if the radio is too quiet, one should turn it up" serves as a heuristic with which a volume controller can be adjusted. The input space is the measured background noise; in this problem, the training time per bin may be quite short, but the recognition of a large number of different background noise levels (having many bins) is limited by the user's willingness to take the time required to train each bin.

The common themes in the deskewer and the two examples presented here are the stability of system output in the absence of an applied control action, which allows the use of an open-loop controller, and the monotonicity of steady-state output as a function of applied control action, which enables us to always tune the controller in the proper direction. We know intuitively that the car stereo will be heard more clearly if it is turned up in the presence of background noise; that the grinder will remove more material if it is pressed more forcefully against the weld bead; and that the deskewer will reduce note skew in proportion to the solenoid firing time.

Appendix B The Deskewer

The deskewer concept was created by a team of four people: Harry West, Ross Levinsky, Ryuichi Onomoto (a mechanical engineer from Omron), and Jack Kotovsky. Kotovsky, in close consultation with Onomoto, designed and built the actual prototype for his bachelor's thesis. All diagrams in this section use measurements that were either taken from his thesis [Kotovsky 1989] or directly from the deskewer.

The deskewer's optical sensor and solenoid-firing relay circuits were designed by Ichiro Kubo, an Omron electrical engineer, and wired by Kotovsky. Levinsky designed and built dedicated control hardware, handled interfacing of the deskewer to an IBM-PC/AT clone, and designed and programmed the control software.

Figure B.1 shows the functional elements of the Omron-supplied test fixture to which the deskewer is attached. Notes enter the transport belts from a feed cartridge identical to those used to store bills in Omron's GX-series ATM; however, the bills are manually inserted into the cartridge in a skewed position to provide rotated test notes for the deskewer. After the notes are deskewed they fall into a small bin at the end of the test fixture. Maximum transport speed of the fixture is approximately 1.5 m/s, equivalent to 10 notes/second throughput.

The deskewer is shown in greater detail at the bottom of Figure B.1. The upper and lower belts (between which the notes are transported) can be clearly seen, as can the two solenoids and four pairs of optical sensors. The whole section is driven by a toothed belt running on the drive gear partially visible on the left side of the drawing.

Figure B.2 lists the parts that make up the main frame of the deskewer. The shafts, belts, rollers, and drive gear are all supplied by Omron and are typical of their note-handling technology.

Figure B.3 lists the optical sensors (supplied by Omron), the solenoids, and miscellaneous hardware.

Figure B.4 gives dimensions of the deskewer's side plate. The 3 mm holes are through-holes for the fixed-shaft mounting screws, while the 10 mm hole holds the bearings for the drive shaft. The left and right side plates are identical.

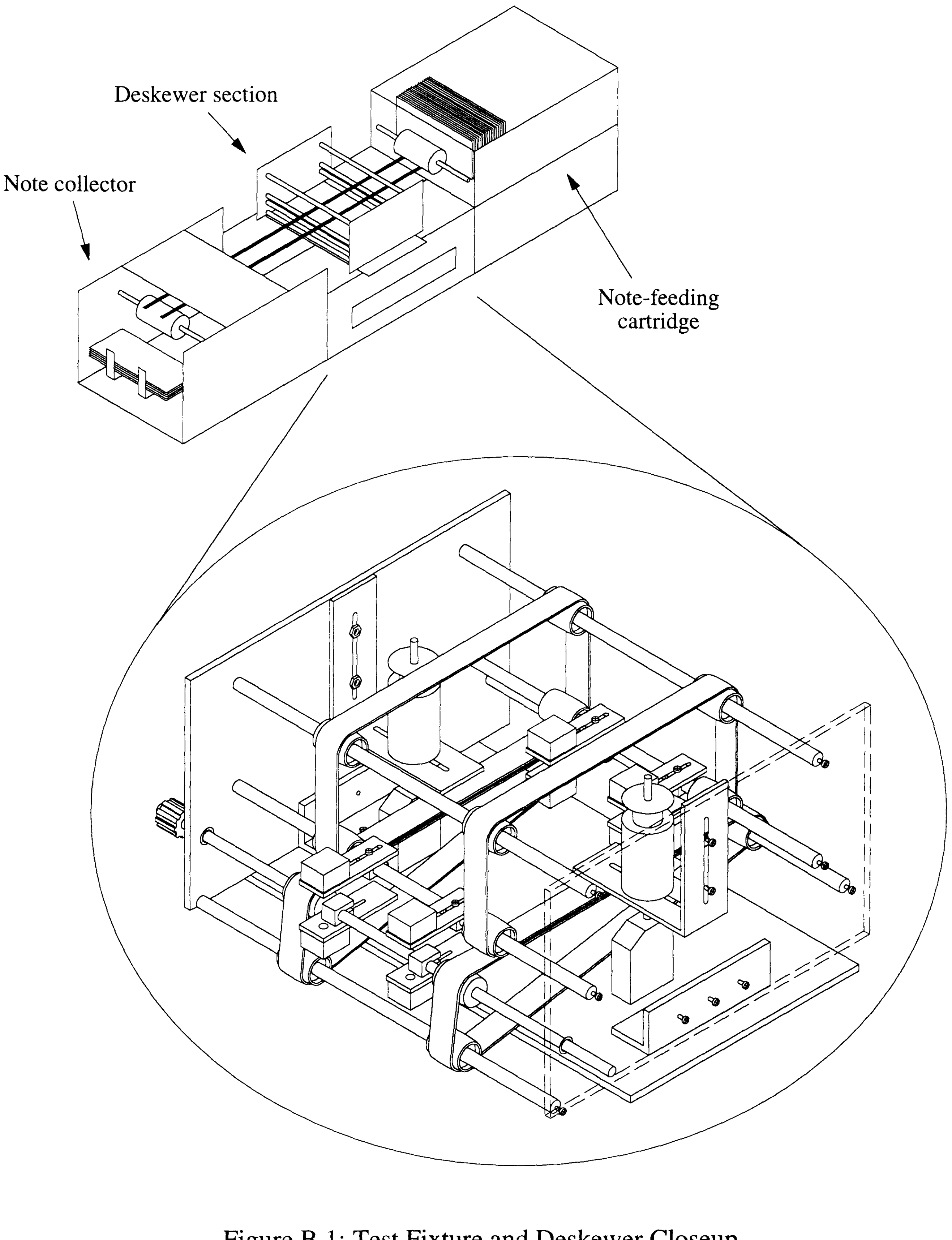
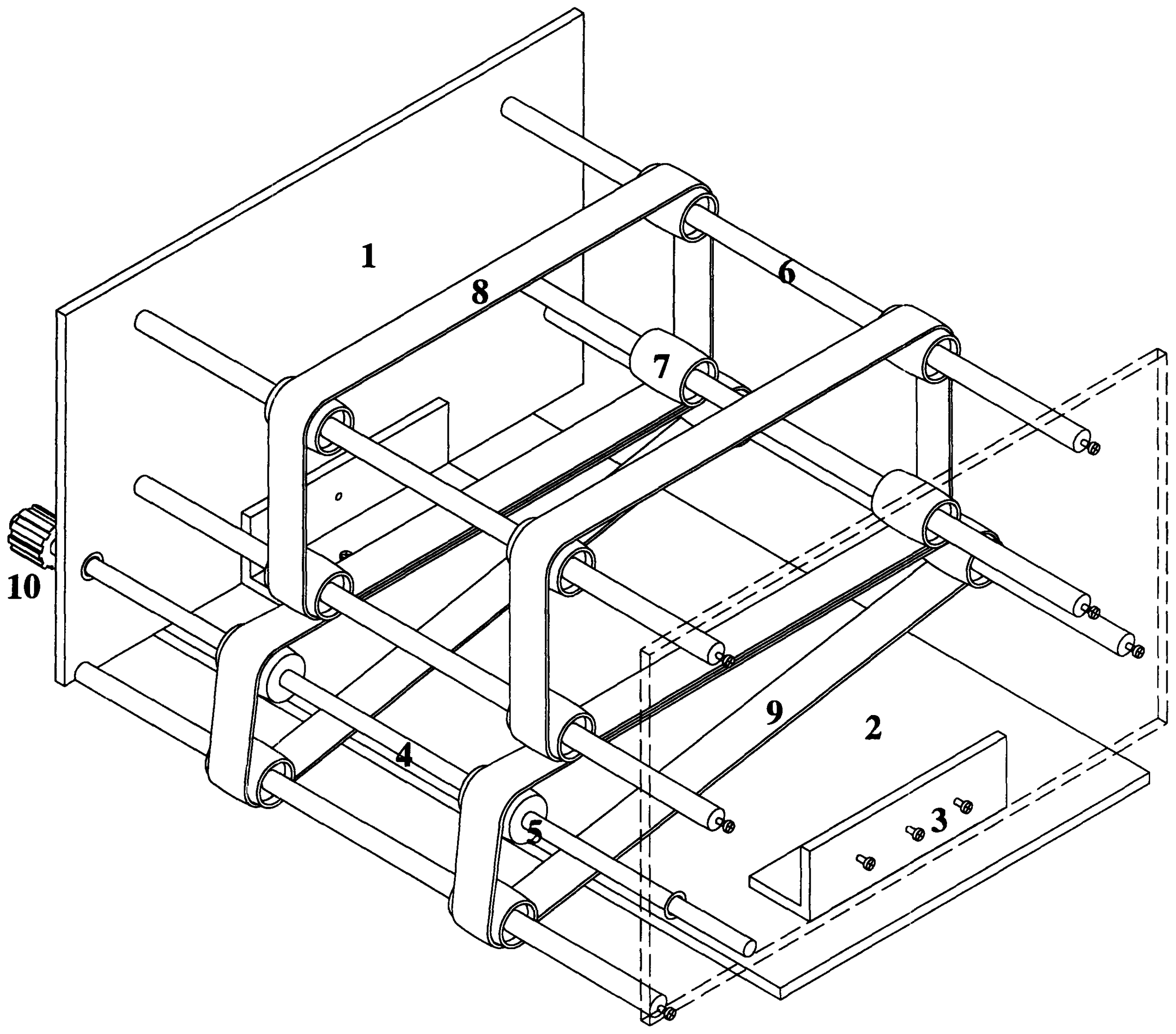
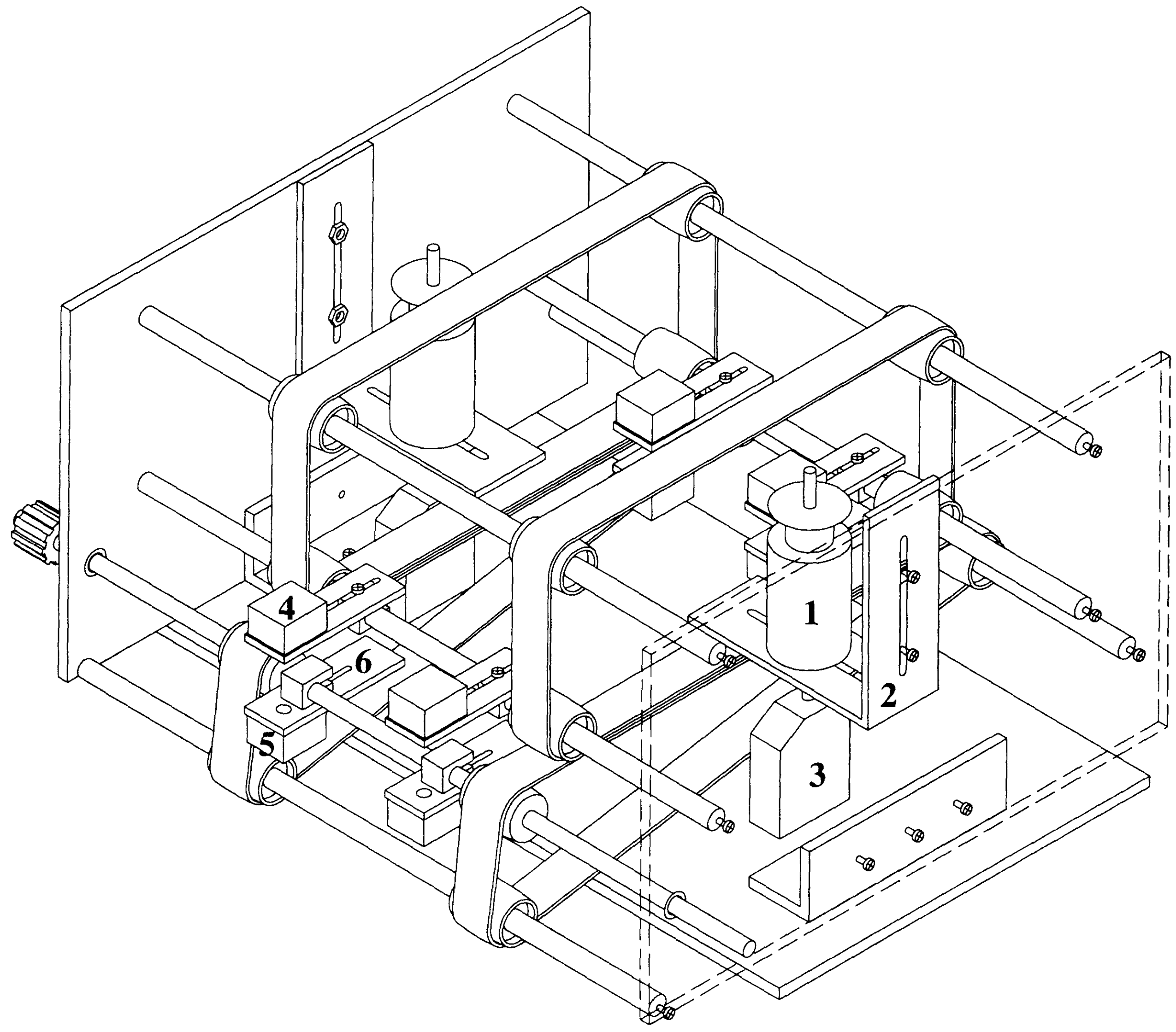


Figure B.1: Test Fixture and Deskewer Closeup



1. 171mm x 105mm x 3mm side plate (2)
2. Bottom plate (1)
3. Angle bracket for side plate/bottom plate attachment (2)
4. 6mm diameter x 250mm length drive shaft (1). Mounted in bearings.
5. 19mm peak-diameter x 18mm-wide crowned drive roller (2). Set-screwed to drive shaft.
6. 8mm diameter x 190mm length roller shaft (6). Fixed to side plates.
7. 19mm peak-diameter x 18mm-wide crowned transport-belt roller (12). Free-spinning on roller shafts.
8. 362mm x 9mm x 1mm (measured unstretched) rubber transport belt (2)
9. 366mm x 9mm x 1mm (measured unstretched) rubber transport belt (2)
10. 14.4mm diameter drive gear (1)

Figure B.2: Main Structure of Deskewer



1. Shindengen F224C-12V solenoid (2)
2. Solenoid mounting bracket (2)
3. Solenoid pinning block (2)
4. Omron/Sharp M601P phototransistor (4)
5. Omron infrared LED (4)
6. Optical sensor mounting assembly, consisting of shaft clamp and adjustable mounting plate (8)

Lateral spacing of the optical-sensor pairs is 5 cm from infrared-beam center to infrared-beam center

Figure B.3: Sensors and Solenoids

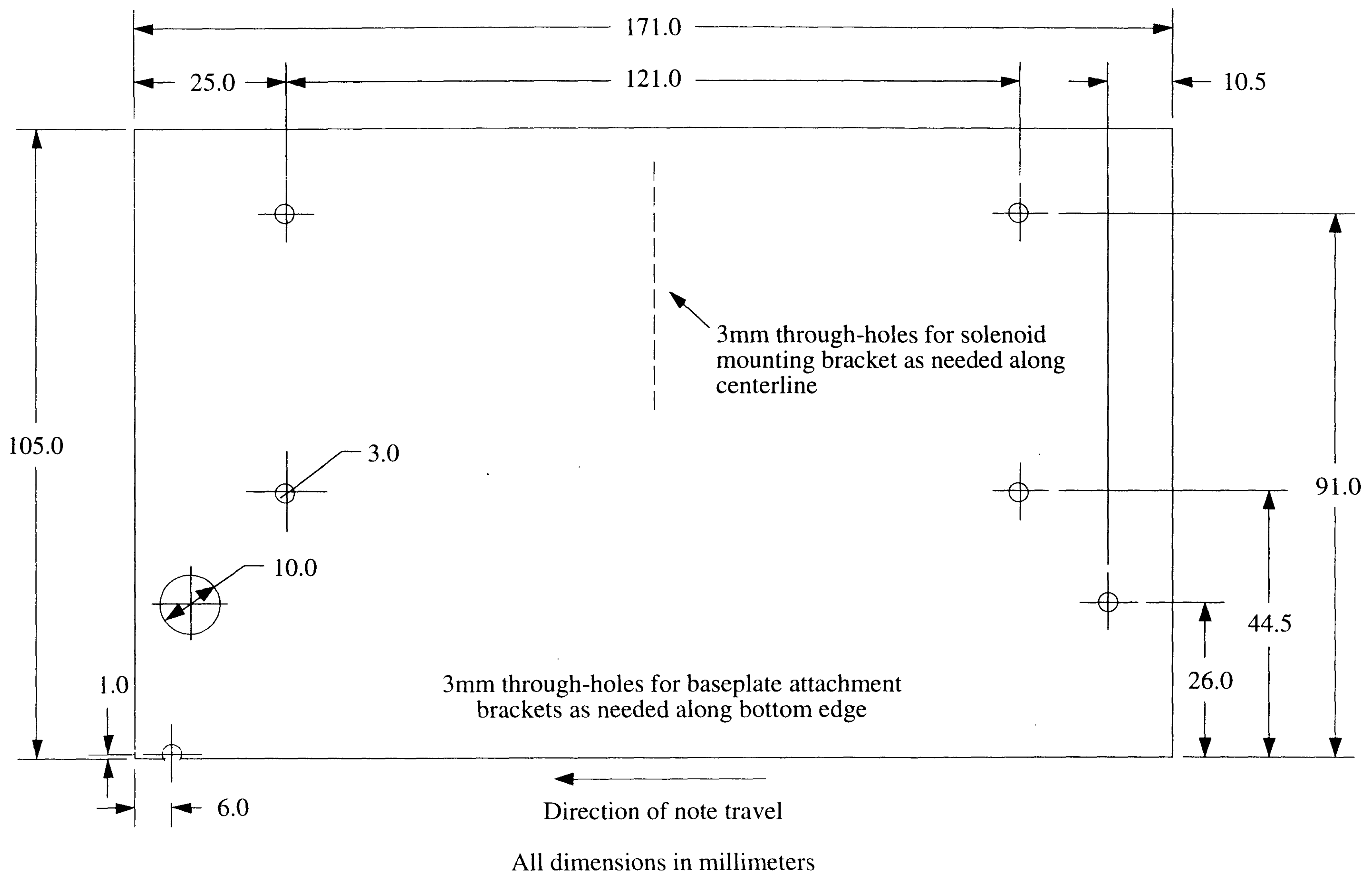

171.0
25.0
121.0
10.5
105.0
3mm through-holes for solenoid mounting bracket as needed along centerline
3.0
10.0
91.0
44.5
26.0
1.0
6.0
3mm through-holes for baseplate attachment brackets as needed along bottom edge
Direction of note travel
All dimensions in millimeters

Figure B.4: Side Plate of Deskewer

Appendix C
Analysis of the Deskewer

C.1 General Case of Coulomb-Frictional Contact

The fundamental model for many bill-handling tasks is that of a machine surface interacting with the note through frictional contact. This is the case in the feeding process, in which the feed roller slips with respect to the note's surface and accelerates the bill into the travel belts. It is also the reason that the deskewer is able to straighten bills; the belts slide over the note and exert a moment that rotates the note about the solenoid's pinning point. Because sliding frictional contact is so fundamental to the note-handling process, it is useful to consider the general case of Coulomb-frictional contact (in which the frictional force is linearly proportional to the contact force).

Consider a small translating plate in contact with a moving surface, as in Figure C.1. The plate can be viewed as a differential element of the note, while the moving surface can be considered a sliding belt or rotating feed roller. Coulomb's Law of Friction describes the magnitude of the frictional force as proportional to the normal force between plate and surface (with proportionality constant μ, the coefficient of friction), and the direction of the frictional force as that of the relative velocity between plate and surface (Figure C.2):

$$\text{Magnitude of frictional force} = |\boldsymbol{F_f}| = \mu |\boldsymbol{F_n}| \tag{C.1-1}$$

$$\text{Direction of frictional force} = \boldsymbol{V_s} - \boldsymbol{V_p} = (V_{sx} - V_{px})\,\hat{\mathbf{i}} + (V_{sy} - V_{py})\,\hat{\mathbf{j}} \tag{C.1-2}$$

To get the complete expression for frictional force on the plate, the force direction vector is normalized to a unit vector. This leads to:

$$\text{Normalized direction of } \boldsymbol{F_f} = \frac{\boldsymbol{V_s} - \boldsymbol{V_p}}{|\boldsymbol{V_s} - \boldsymbol{V_p}|} = \frac{(V_{sx} - V_{px})\,\hat{\mathbf{i}} + (V_{sy} - V_{py})\,\hat{\mathbf{j}}}{\sqrt{(V_{sx} - V_{px})^2 + (V_{sy} - V_{py})^2}} \tag{C.1 -3}$$

The frictional force vector is simply the frictional force's magnitude multiplied by its normalized direction vector:

$$\boldsymbol{F_f} = |\boldsymbol{F_f}| \cdot \frac{\boldsymbol{V_s} - \boldsymbol{V_p}}{|\boldsymbol{V_s} - \boldsymbol{V_p}|} = \mu |\boldsymbol{F_n}| \frac{(V_{sx} - V_{px})\,\hat{\mathbf{i}} + (V_{sy} - V_{py})\,\hat{\mathbf{j}}}{\sqrt{(V_{sx} - V_{px})^2 + (V_{sy} - V_{py})^2}} \tag{C.1-4}$$

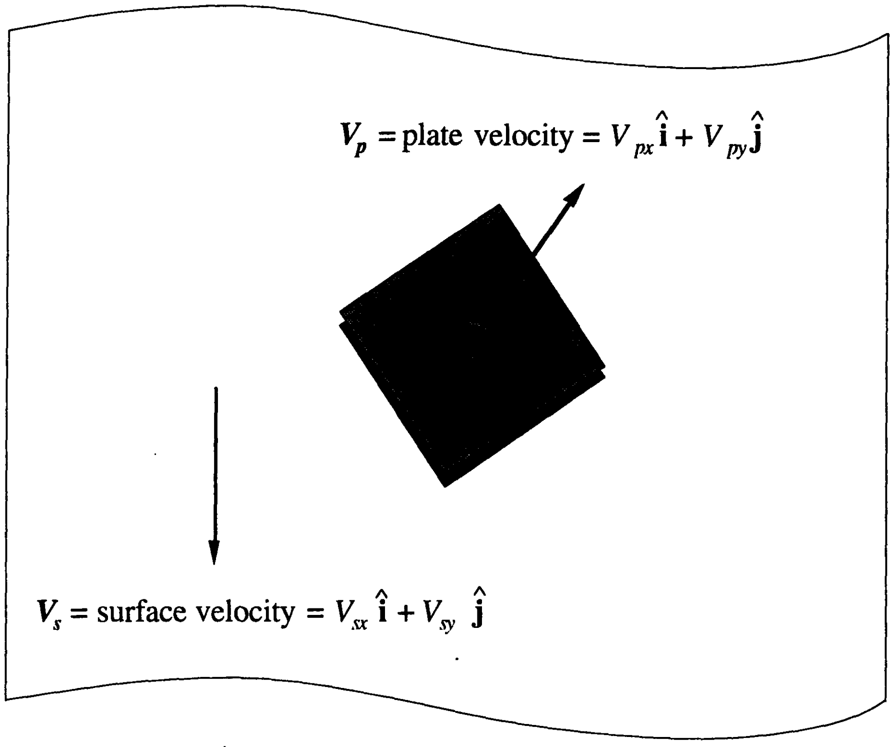

V_p = plate velocity = $V_{px}\hat{\mathbf{i}} + V_{py}\hat{\mathbf{j}}$
V_s = surface velocity = $V_{sx}\hat{\mathbf{i}} + V_{sy}\hat{\mathbf{j}}$

Figure C.1: Small Plate Element Translating on Moving Surface

Equation (C.1-4) is the general expression for sliding-friction force on a non-rotating element. A note can be thought of as many tiny elements connected together, each with a slightly different velocity vector (due to the entire note's rotation). If the entire region of contact between the note and the surface is integrated, the total force acting on the note can be found, and its motion can be predicted. The surface's definition changes according to the specific analysis; for the feeder it is the feed roller area in contact with the note, while for the deskewer it is the belt area in contact with the note (neglecting the area pinned by the solenoid). Because the contact area changes with time, a closed-form solution to the equations of motion may be impossible.

C.2 Deskewer-Specific Equations

Figure C.3 is a plan view of the deskewer prototype. Equation (C.1-4) can be applied by considering the moving belts to be the surface and the note to be a connected collection of small plates. With the coordinate system oriented as shown, V_{sx}, the x - direction component of the surface velocity, is zero, and V_{sy}, the y - direction component of the surface velocity, is set to V_b, the belt velocity:

$$\begin{aligned} V_{sx} &= 0 \\ V_{sy} &= V_b \end{aligned} \qquad \text{(C.2-1)}$$

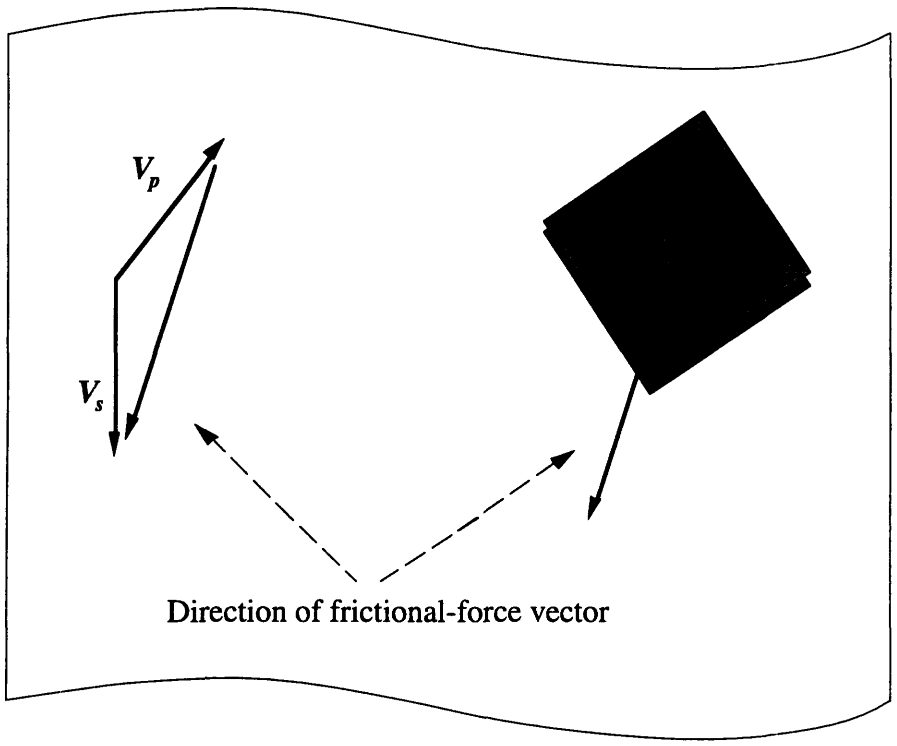


Figure C.2: Direction of Frictional-Force Vector

Because the note is pinned by the solenoid and rotates around the pinning point, it is convenient to express the note's surface velocity as a function of surface position and rotation rate about the pinning point. If ω (= $\dot{\theta}$) is the angular velocity of the note about the pinning point, (x, y) is the position of an arbitrary point on the note's surface, and S is the solenoid displacement from the centerline of the travel path (as in Figure C.3):

$$\text{Velocity of note point}\,(x, y) = -\,\omega y\,\hat{\mathbf{i}} - \omega(S - x)\,\hat{\mathbf{j}} \tag{C.2-2}$$

Thus:

$$\begin{aligned} V_{px} &= -\,\omega y \\ V_{py} &= -\,\omega(S - x) \end{aligned} \tag{C.2-3}$$

Substituting into (C.1-4):

$$\boldsymbol{F_f} = \mu\,|\boldsymbol{F_n}|\,\frac{\omega y\,\hat{\mathbf{i}} + \big(V_b + \omega(S - x)\big)\,\hat{\mathbf{j}}}{\sqrt{(\omega y)^2 + \big(V_b + \omega(S - x)\big)^2}} \tag{C.2-4}$$

The note is viewed as a connected group of small plates, so the normal force is more properly treated as a normal stress σ (assumed constant over a single plate due to its small size) multiplied by the area of a plate.

$$|\boldsymbol{F_n}| = \sigma(x, y)\,dx\,dy \tag{C.2-5}$$

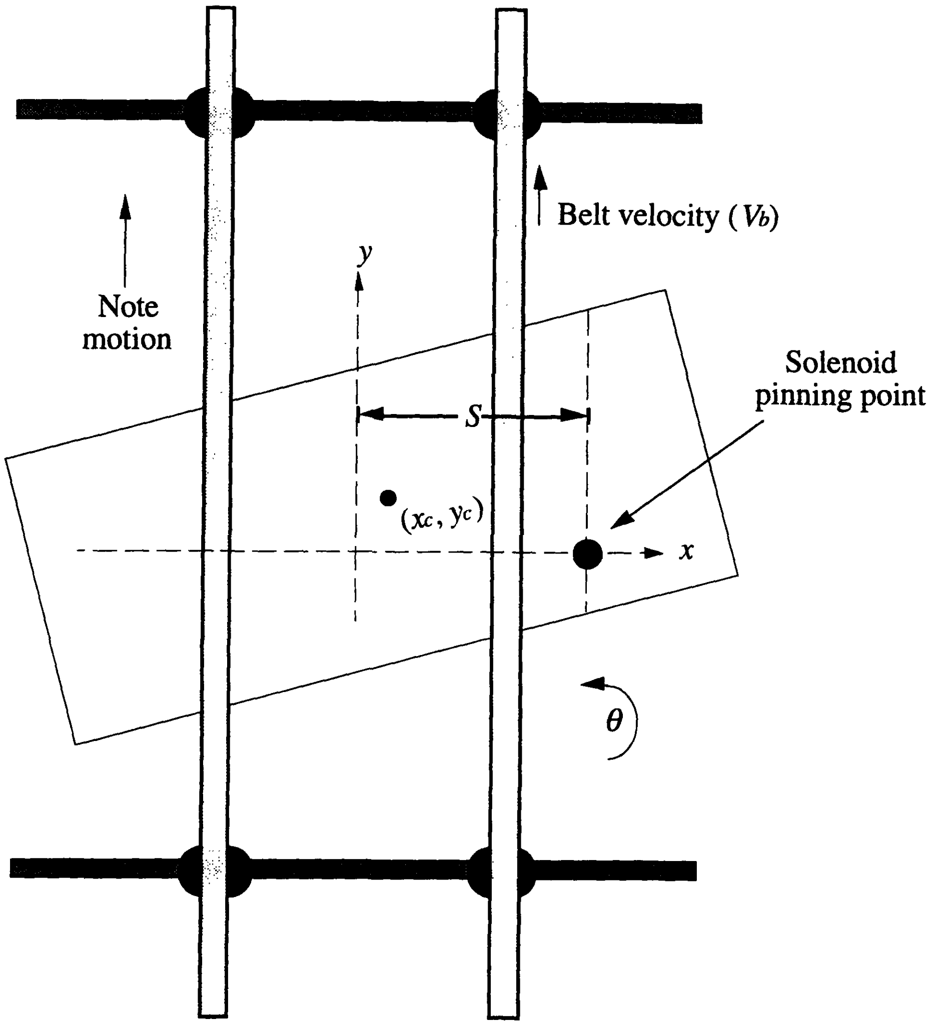


Figure C.3: Plan View of Deskewing System

The standard differential equation for a rotating rigid body is now applied. *I* is the moment of inertia of the note when rotating about a given solenoid pivot-point position, and *R*, the region of integration, is the contact area between the belts and the note. It varies with time.

$$I\ddot{\theta} = \iint_R \mu\,\sigma(x,y)\,\frac{\omega y\,\hat{\mathbf{i}} + \left(V_b + \omega(S-x)\right)\hat{\mathbf{j}}}{\sqrt{(\omega y)^2 + \left(V_b + \omega(S-x)\right)^2}}\,dx\,dy \qquad \text{(C.2-6)}$$

The remaining difficulty is the prediction of the normal forces between the note and the belts. This problem is complex because of the high flexibility of the belts and the note, so for numerical investigations the normal stresses are generally assumed to be constant over the contact area.

C.3 Example Solution of Deskewer Equations

It can be seen immediately that (C.2-6) is difficult to solve analytically even with a constant value for σ, so a computer simulation has been written to numerically solve the equation. For simplicity, the program ignores buckling of the bill by assuming the note to be a rigid plate. First a spacewise integration is performed over the note-belt contact area to determine the force on the bill, and then a time-domain integration is performed to find the motion of the bill over a small time interval. The fineness of the integration steps is then adjusted until consistent results are obtained. Predicted performance (in the form of necessary solenoid firing time versus input skew) has been obtained for a number of different machine configurations, and a sample result graph is shown below in Figure C.4.

Three graphs of the τ required to straighten the bill perfectly versus θ_{in} are displayed, each representing a different position, relative to the leading edge of the bill, of the solenoid pinning point. P refers to the normalized distance back from the leading edge; $P = 0$ means the pinning point is at the leading edge and $P = 1$ indicates that the pin occurs at the trailing edge. The lateral solenoid position, S, and belt speed, V_b, are fixed in all cases shown, with the solenoid position at 30 mm and the belt speed at 1 m/s.

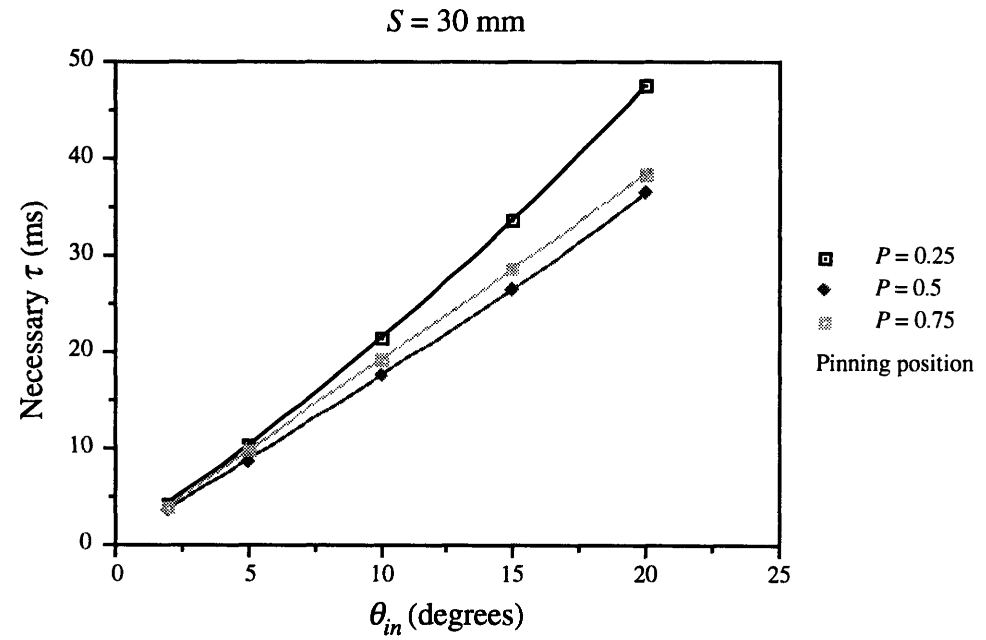


Figure C.4: Simulation Results for Varying Pin Position P

Experimental evidence indicates that this graph is qualitatively correct. Pinning positions of $P < 0.5$ tend to require longer firing times for adequate deskewing, while those towards the trailing edge need less time. As an additional advantage, the trailing-edge positions have been observed to cause less note-buckling than those towards the leading edge, because the deskewing force provided by the belts acts to tension the bill.

Figure C.5 shows a graph similar to that in Figure C.4, except that the pinning position is now fixed at $P = 0.75$, while the lateral solenoid position is varied from 30 mm to 65 mm. The simulation indicates that the required deskewing time is slightly affected by solenoid position, with $S = 65$ mm consistently requiring the most time. The biggest variation in timing occurs at an input skew of 20°, and is 4.4 milliseconds (between the 65 mm and the 50 mm positions).

This effect is notable because the feeding process normally causes slight (up to approximately 10 mm) variation in the lateral position of notes. Two bills with identical input skews but horizontally-offset positions in the transport belts are entering two slightly different deskewing systems, distinguished by a difference in S. The notes thus require different deskewing times.

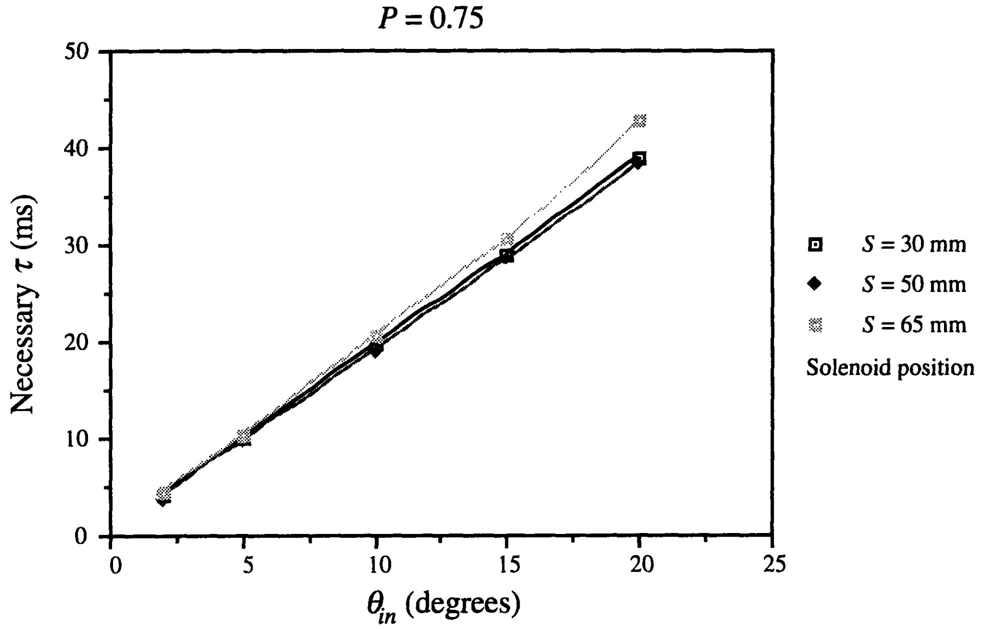


Figure C.5: Simulation Results for Varying Solenoid Position S

Appendix D
Adjustment of Bin Boundaries

This appendix discusses a gradient descent procedure that uses measured performance data to move the prechosen bin boundaries in directions that reduce output error. This technique is meant to be applied to systems that do not have a constant β_1.

We first examine the contribution to total system error from two adjacent bins. From (1.3-4):

$$E\left(\theta_{out}^2\right) \text{ for two adjacent rules} = \int_{a_{i-1}}^{a_i} \overline{\theta_{out}^2}\left(\theta_{in}, \tau_i\right) p\left(\theta_{in}\right) d\theta_{in} + \int_{a_i}^{a_{i+1}} \overline{\theta_{out}^2}\left(\theta_{in}, \tau_{i+1}\right) p\left(\theta_{in}\right) d\theta_{in} \quad \text{(D-1)}$$

We attempt to reduce the error caused by suboptimal placement of the boundary between these two terms with a gradient descent procedure. Taking the partial derivative of output error with respect to a_i:

$$\frac{\partial E\left(\theta_{out}^2\right)}{\partial a_i} = \left(\overline{\theta_{out}^2}\left(\theta_{in} = a_i, \tau_i\right) - \overline{\theta_{out}^2}\left(\theta_{in} = a_i, \tau_{i+1}\right)\right) p\left(\theta_{in} = a_i\right) \quad \text{(D-2)}$$

As is usual with gradient descent formulas, the boundary is moved in the direction of the negative of the gradient, to expand the range of the lower-error bin.

To evaluate (D-2) further, we again examine the uniform-input-distribution case with identical σ in all bins. By using measurements of average output and average squared output of adjacent bins, we can determine values for $\overline{\theta_{out}^2}\left(\theta_{in} = a_i, \tau_i\right)$ and $\overline{\theta_{out}^2}\left(\theta_{in} = a_i, \tau_{i+1}\right)$.

Evaluating (2.3-3) and (2.3-7) for a uniform input distribution:

$$\mu_i = \frac{1}{\int_{a_{i-1}}^{a_i} \frac{1}{W} d\theta_{in}} \int_{a_{i-1}}^{a_i} \left(\beta_0 + \beta_1 \theta_{in}\right) \frac{1}{W} d\theta_{in} = \frac{1}{2}\beta_1\left(a_{i-1} + a_i\right) + \beta_0 \quad \text{(D-3)}$$

$$S_i^2 = \frac{\int_{a_{i-1}}^{a_i} \beta_1^2 \theta_{in}^2 \frac{1}{W} d\theta_{in}}{\int_{a_{i-1}}^{a_i} \frac{1}{W} d\theta_{in}} - \frac{\left(\int_{a_{i-1}}^{a_i} \beta_1 \theta_{in} \frac{1}{W} d\theta_{in}\right)^2}{\left(\int_{a_{i-1}}^{a_i} \frac{1}{W} d\theta_{in}\right)^2} + \mu_i^2 + \sigma^2 = \frac{1}{12}\beta_1^2\left(a_i - a_{i-1}\right)^2 + \mu_i^2 + \sigma^2 \quad \text{(D-4)}$$

Solving (D-3) and (D-4) for β_1 and β_0:

$$\beta_0 = \frac{1}{2}\beta_1(a_{i-1} + a_i) - \mu_i \tag{D-5}$$

$$\beta_1 = \frac{2\sqrt{3(S_i^2 - \mu_i^2 - \sigma^2)}}{(a_i - a_{i-1})} \tag{D-6}$$

Substituting (D-5) and (D-6) into (2.3-5) and simplifying:

$$\begin{aligned}\overline{\theta_{out}^2}(\theta_{in},\tau_i) &= \beta_1^2\theta_{in}^2 + 2\beta_1\beta_0\theta_{in} + \beta_0^2 + \sigma^2 \\ &= 3S_i^2 - 2\mu_i^2 - 2\sigma^2 + (2\sqrt{3})\,\mu_i\sqrt{S_i^2 - \mu_i^2 - \sigma^2}\end{aligned} \tag{D-7}$$

An identical analysis performed on bin $(i + 1)$ yields:

$$\overline{\theta_{out}^2}(\theta_{in},\tau_{i+1}) = 3S_{i+1}^2 - 2\mu_{i+1}^2 - 2\sigma^2 - (2\sqrt{3})\,\mu_{i+1}\sqrt{S_{i+1}^2 - \mu_{i+1}^2 - \sigma^2} \tag{D-8}$$

Substituting (D-7) and (D-8) into (D-2)

$$\begin{aligned}\frac{\partial E(\theta_{out}^2)}{\partial a_i} &= p(\theta_{in}= a_i)\Big(3(S_i^2 - S_{i+1}^2) - 2(\mu_i^2 - \mu_{i+1}^2) \\ &\quad + 2\sqrt{3}\Big(\mu_i\sqrt{S_i^2 - \mu_i^2 - \sigma^2} + \mu_{i+1}\sqrt{S_{i+1}^2 - \mu_{i+1}^2 - \sigma^2}\Big)\Big)\end{aligned} \tag{D-9}$$

With this analysis we assume that perfect, noise-free measurements of stationary values of S^2 and μ are available. Thus, (D-9) cannot be used to move bin boundaries while adjustment of τ is ongoing, and we must have enough samples of system output to make good estimates of S^2 and μ in each bin. It is also questionable whether bin boundary adjustment should be applied while μ is not "sufficiently" close to zero and while further training may be possible. Figure D.1 depicts a situation in which the gradient descent method moves bin boundary a_i in a direction that is desirable if τ_i (and thus μ_i) is fixed for all time, but undesirable if possible future training moves μ_i and μ_{i+1} closer to zero.

If we restrict bin boundary motion to cases in which both μ_i and μ_{i+1} are approximately zero (the right side of Figure D.1), we find:

$$\frac{\partial E(\theta_{out}^2)}{\partial a_i} = 3(S_i^2 - S_{i+1}^2)p(\theta_{in}= a_i) \quad \text{for } \mu_i, \mu_{i+1} \approx 0 \tag{D-10}$$

The bin boundary is thus moved to reduce the size of the bin with larger average squared output. The premise of this process is clear: the value of β_1 may differ in adjacent bins, so we exploit this possibility to adjust bin boundary location in a way that reduces the contribution to overall system error from the bin with larger β_1.

From (D-10) we see that an added benefit of adjusting bin boundaries with $\mu \approx 0$ is that no knowledge of σ is necessary (although the assumption of constant σ still holds).

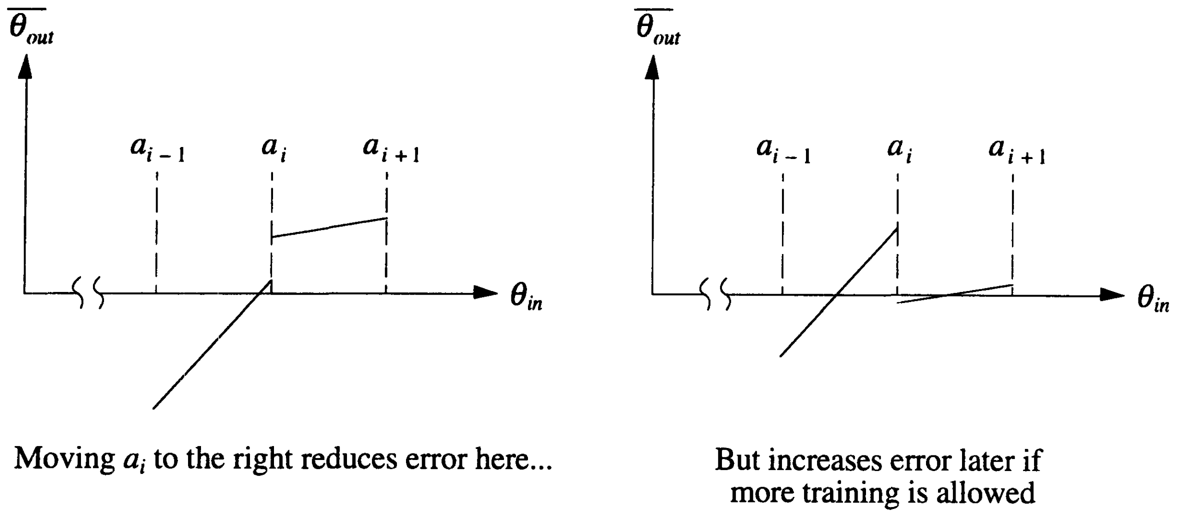

$\overline{\theta_{out}}$
a_{i-1}
a_i
a_{i+1}
θ_{in}
Moving a_i to the right reduces error here...
$\overline{\theta_{out}}$
a_{i-1}
a_i
a_{i+1}
θ_{in}
But increases error later if more training is allowed

Figure D.1: Adjusting Bin Boundaries when $\mu \neq 0$

Appendix E
Tracking Outputs of Individual Bins

In Chapter 3, the decision to split all bins into equal-width halves, simultaneously, yields an analytical expression for the optimal splitting iterations that only requires knowledge of the initial squared average output of the original parent bin. The more general case, in which each sub-bin is split independently, and not necessarily symmetrically, is addressed in this appendix. We find that if $E(\mu)$ and $E(\mu^2)$ are known in a parent bin, there is more flexibility in the splitting process; this information is sufficient to generate analogous values for the sub-bins. They can in turn be treated as independent parent-bins for the next generation, and are no longer forced to split at the same time.

To derive the relations between the $E(\mu)$ and $E(\mu^2)$ of the initial wide bin and the sub-bins, we consider the situation just after the splitting iteration, but before the sub-bins have been trained separately. The sub-bins still have the same control action as their parent; we simply wish to know what $E(\mu)$ and $E(\mu^2)$ would be in each sub-bin if they were to be identified at any given training iteration of the parent, but left with unaltered control actions.

Figure E.1 shows the boundaries of the initial bin at *a* and *c*, and the point at which it is later split, *b*. The initial bin (range *a*-*c*) will be referred to with a lower-case *w* subscript, while the *a*-*b* and *b*-*c* sub-ranges will be denoted with subscript *l* and *r*, respectively (the *a*-*b*-*c* notation is inconsistent with other sections' a_i notation, but here makes the equations easier to follow). The upper-case *W* refers to the entire range of the input space.

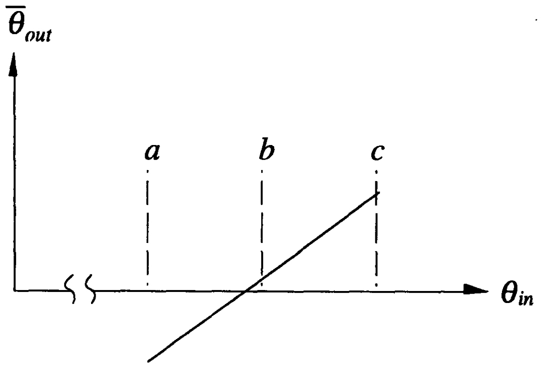


Figure E.1: Initial Bin - To Be Split at *b*

The average output for the initial range *a-c* is:

$$\mu_w = \frac{1}{\left(\int_a^c \frac{1}{W} d\theta_{in}\right)} \frac{1}{W}\left(\int_a^c (\beta_1 \theta_{in} + \beta_0) d\theta_{in}\right) = \frac{1}{2}\beta_1(c+a) + \beta_0 \qquad \text{(E-1)}$$

The integral in the denominator of (E-1) is the normalization factor that lets us treat the initial range as the entire system under consideration. Squaring (E-1):

$$\mu_w^2 = \frac{1}{4}\beta_1^2(c+a)^2 + \beta_0^2 + \beta_1\beta_0(c+a) \qquad \text{(E-2)}$$

The average output and its square for the left bin are given below. Note the presence of another normalizing factor; we treat this sub-bin as a self-contained system.

$$\mu_l = \left(\frac{1}{W}\left(\int_a^b (\beta_1 \theta_{in} + \beta_0) d\theta_{in}\right)\right) \frac{1}{\int_a^b \frac{1}{W} d\theta_{in}} = \frac{1}{2}\beta_1(b+a) + \beta_0 \qquad \text{(E-3)}$$

$$\mu_l^2 = \frac{1}{4}\beta_1^2(b+a)^2 + \beta_1\beta_0(b+a) + \beta_0^2 \qquad \text{(E-4)}$$

Doing the same for the right bin, μ_r:

$$\mu_r = \left(\frac{1}{W}\left(\int_b^c (\beta_1 \theta_{in} + \beta_0) d\theta_{in}\right)\right) \frac{1}{\int_b^c \frac{1}{W} d\theta_{in}} = \frac{1}{2}\beta_1(b+c) + \beta_0 \qquad \text{(E-5)}$$

$$\mu_r^2 = \frac{1}{4}\beta_1^2(b+c)^2 + \beta_1\beta_0(b+c) + \beta_0^2 \qquad \text{(E-6)}$$

The stochastic adjustment of control action means that μ_w has a probabilistic distribution. Since β_1 is fixed, we can write:

$$E(\mu_w) = \frac{1}{2}\beta_1(c+a) + E(\beta_0) \qquad \text{(E-7)}$$

or:

$$E(\beta_0) = E(\mu_w) - \frac{1}{2}\beta_1(c+a) \qquad \text{(E-8)}$$

Similarly, from (E-2):

$$E(\mu_w^2) = \frac{1}{4}\beta_1^2(c+a)^2 + E(\beta_0^2) + \beta_1 (c+a) E(\beta_0) \qquad \text{(E-9)}$$

Substituting (E-8) into (E-9):

$$E\left(\beta_0^2\right) = E\left(\mu_w^2\right) - \beta_1(c+a)\left(E(\mu_w) - \frac{1}{2}\beta_1(c+a)\right) - \frac{1}{4}\beta_1^2(c+a)^2 \qquad \text{(E-10)}$$

Simplifying:

$$E\left(\beta_0^2\right) = E\left(\mu_w^2\right) - \beta_1(c+a)E(\mu_w) + \frac{1}{4}\beta_1^2(c+a)^2 \qquad \text{(E-11)}$$

In equations (E-8) and (E-11) are expressions for $E(\beta_0)$ and $E\left(\beta_0^2\right)$. We can now put these expressions into those for $E(\mu_l)$ and $E(\mu_r)$. From (E-4):

$$E\left(\mu_l^2\right) = \frac{1}{4}\beta_1^2(b+a)^2 + \beta_1(b+a)E(\beta_0) + E\left(\beta_0^2\right) \qquad \text{(E-12)}$$

Substituting (E-8) and (E-11) into (E-12) and simplifying:

$$E\left(\mu_l^2\right) = E\left(\mu_w^2\right) - \beta_1(c-b)E(\mu_w) + \frac{1}{4}\beta_1^2(c-b)^2 \qquad \text{(E-13)}$$

From (E-6):

$$E\left(\mu_r^2\right) = \frac{1}{4}\beta_1^2(b+c)^2 + \beta_1(b+c)\,E(\beta_0) + E\left(\beta_0^2\right) \qquad \text{(E-14)}$$

Substituting (E-8) and (E-11) into (E-14) and simplifying:

$$E\left(\mu_r^2\right) = E\left(\mu_w^2\right) + \beta_1(b-a)E(\mu_w) + \frac{1}{4}\beta_1^2(b-a)^2 \qquad \text{(E-15)}$$

Following the previous definitions:

$$\begin{aligned}
\phi_l &= \left(1 - \frac{(b-a)}{W} + \frac{(b-a)}{W}\phi\right) \\
\phi_r &= \left(1 - \frac{(c-b)}{W} + \frac{(c-b)}{W}\phi\right) \\
\sigma_{bin,\,l}^2 &= \frac{1}{12}\beta_1^2(b-a)^2 + \sigma^2 \\
\sigma_{bin,\,r}^2 &= \frac{1}{12}\beta_1^2(c-b)^2 + \sigma^2
\end{aligned} \qquad \text{(E-16)}$$

Inserting (E-16) in the equation for iterative evolution of the squared average output, (2.2-24):

$$\begin{aligned}
E\left(\mu_{l,\,n}^2\right) &= \phi_l^n E\left(\mu_{l,\,n=0}^2\right) + \frac{\left(1-\phi_l^n\right)v^2}{(1-\phi)\gamma^2}\left(\sigma_{bin,\,l}^2\right) \\
E\left(\mu_{r,\,n}^2\right) &= \phi_r^n E\left(\mu_{r,\,n=0}^2\right) + \frac{\left(1-\phi_r^n\right)v^2}{(1-\phi)\gamma^2}\left(\sigma_{bin,\,r}^2\right)
\end{aligned} \qquad \text{(E-17)}$$

Writing the output of the two-bin system as in Section 3.4:

$$E\left(S^2\right)=\frac{(b-a)}{(c-a)}\left(\frac{\beta_1^2(b-a)^2}{12}+\sigma^2+\phi_l^{N-n}E\left(\mu^2_{l,\,n\,=\,\text{splitting iteration}}\right)+\frac{\left(1-\phi_l^{N-n}\right)v^2}{(1-\phi)\gamma^2}\left(\sigma^2_{bin,\,l}\right)\right)$$
$$+\frac{(c-b)}{(c-a)}\left(\frac{\beta_1^2(c-b)^2}{12}+\sigma^2+\phi_r^{N-n}E\left(\mu^2_{r,\,n\,=\,\text{splitting iteration}}\right)+\frac{\left(1-\phi_r^{N-n}\right)v^2}{(1-\phi)\gamma^2}\left(\sigma^2_{bin,\,r}\right)\right) \qquad \text{(E-18)}$$

By using the expressions for $E\left(\mu_w^2\right)$ and $E\left(\mu_w\right)$ as a function of n, and inserting them into (E-13) and (E-15), we obtain $E\left(\mu^2_{l,\,n\,=\,\text{splitting iteration}}\right)$ and $E\left(\mu^2_{r,\,n\,=\,\text{splitting iteration}}\right)$ in (E-18). We then substitute the notation of (E-16), and arrive at the equation for the expected average squared output of the two-bin system after the final iteration, N, as a function of splitting iteration n.

$$E\left(S^2\right)=\frac{(b-a)}{(c-a)}\left(\frac{\beta_1^2(b-a)^2}{12}+\sigma^2+\frac{\left(1-\left(1-\frac{(b-a)}{W}+\frac{(b-a)}{W}\phi\right)^{N-n}\right)v^2}{(1-\phi)\gamma^2}\left(\frac{1}{12}\beta_1^2(c-b)^2+\sigma^2\right)\right.$$
$$+\left(1-\frac{(b-a)}{W}+\frac{(b-a)}{W}\phi\right)^{N-n}\left(\phi_w^nE\left(\mu^2_{w,\,n=0}\right)+\frac{\left(1-\phi_w^n\right)v^2}{(1-\phi)\gamma^2}\left(\sigma^2_{bin,\,w}\right)\right.$$
$$\left.\left.-\beta_1(c-b)\phi_w^nE\left(\mu_{w,\,n=0}\right)+\frac{1}{4}\beta_1^2(c-b)^2\right)\right)$$
$$+\frac{(c-b)}{(c-a)}\left(\frac{\beta_1^2(c-b)^2}{12}+\sigma^2+\frac{\left(1-\left(1-\frac{(c-b)}{W}+\frac{(c-b)}{W}\phi\right)^{N-n}\right)v^2}{(1-\phi)\gamma^2}\left(\frac{1}{12}\beta_1^2(c-b)^2+\sigma^2\right)\right.$$
$$+\left(1-\frac{(c-b)}{W}+\frac{(c-b)}{W}\phi\right)^{N-n}\left(\phi_w^nE\left(\mu^2_{w,\,n=0}\right)+\frac{\left(1-\phi_w^n\right)v^2}{(1-\phi)\gamma^2}\left(\sigma^2_{bin,\,w}\right)\right.$$
$$\left.\left.+\beta_1(b-a)\phi_w^nE\left(\mu_{w,\,n=0}\right)+\frac{1}{4}\beta_1^2(b-a)^2\right)\right)$$

(E-19)

(E-19) may be optimized with respect to b and n to find the best boundary location and splitting time, but an analytical solution is not known at this time. A significant simplification occurs if we divide the initial range evenly by setting $b = {}^{(c+a)}\!/_2$; we will work from (E-18), to avoid the algebraic complexity of (E-19):

$$E(S^2) = \frac{\beta_1^2(b-a)^2}{12} + \sigma^2 + \frac{\left(1-\phi_l^{N-n}\right)v^2}{(1-\phi)\gamma^2}\left(\sigma_{bin,l}^2\right) + \frac{\phi_l^{N-n}}{2}\left(E\left(\mu_{l,\,n=\text{splitting iteration}}^2\right) + E\left(\mu_{r,\,n=\text{splitting iteration}}^2\right)\right)$$

(E-20)

The notation applicable to the left bin has been used in (E-20), but as $\sigma_{bin,\,l}^2 = \sigma_{bin,\,r}^2$ and $\phi_l = \phi_r$, the right-bin notation could have been used with no change in the form of the equation. Noting that $b - a = {}^{w}\!/_2$, and substituting (E-13) and (E-15) into (E-20) yields the following equation:

$$E(S^2) = \frac{1}{48}\beta_1^2 w^2 + \sigma^2 + \frac{\left(1-\phi_l^{N-n}\right)v^2}{(1-\phi)\gamma^2}\left(\sigma_{bin,\,l}^2\right) + \phi_l^{N-n}\left(E\left(\mu_{w,\,n=\text{splitting iteration}}^2\right) + \frac{1}{16}\beta_1^2 w^2\right) \quad \text{(E-21)}$$

By inserting a value of $E\left(\mu_{w,\,n=\text{splitting iteration}}^2\right)$ in terms of $E\left(\mu_{w,\,n=0}^2\right)$, (E-21) is transformed into (3.5-3).

Thus, while the general case of bin-splitting location may be examined via (E-19), a much simpler formulation is possible if the splitting point is located in the center of the bin, as in (E-21). It allows us to "work backwards" in the equations for output error (regardless of how many bins there are, as long as the number is a power of two) until all results depend on the single quantity $E\left(\mu_{w,\,n=0}^2\right)$.

As noted at the beginning of this appendix, by tracking both $E(\mu_w)$ and $E(\mu_w^2)$, we enable the sub-bins to become independent parents to the next generation of bins. $E(\mu_l)$ and $E(\mu_r)$ can be found trivially from (E-3), (E-5), and (E-8). Along with (E-13) and (E-15), we have all the necessary information. Taking the right sub-bin as an example, $E(\mu_r)$ and $E(\mu_r^2)$ are merely substituted for $E(\mu_w)$ and $E(\mu_w^2)$ in the above results.

Appendix F
Estimation of Model Parameters

F.1 Parameters to be Estimated

It can be shown from (2.3-7) that for the simplified model used in Chapters 2 and 3 (uniform input distribution, linear output variation), the expected average squared output of a bin of width w is:

$$E(S^2) = \frac{1}{12}\beta_1^2 w^2 + E(\mu^2) + \sigma^2 \qquad \text{(F.1-1)}$$

There are two fixed parameters in (F.1-1) that are not known *a priori*: σ and β_1. To optimize the controller, values of λ and ν are also needed, and Section F.3 suggests a method for their estimation.

The expectation operators are present in (F.1-1) because the future value of μ^2 (and by extension, S^2) has a stochastic component, as seen in the learning analysis of Section 2.2. Nonetheless, at any given time it is always true that:

$$S^2 = \frac{1}{12}\beta_1^2 w^2 + \mu^2 + \sigma^2 \qquad \text{(F.1-2)}$$

To find working values for S^2 and μ^2, it is well-known that the minimum-variance unbiased estimator is the sample statistic, taken from observed performance data. We denote sample statistics with a ^ over the estimated quantity.

F.2 Estimation of σ

In an extremely narrow bin ($w \approx 0$), the term of (F.1-2) in which w appears is nearly zero. We can then write:

$$S^2 \approx \mu^2 + \sigma^2 \qquad \text{(F.2-1)}$$

If we take enough samples to have "good" estimates of S^2 and μ^2, we find:

$$\sigma^2 \approx \widehat{S^2} - \widehat{\mu^2} \qquad \text{(F.2-2)}$$

This estimation can be repeated in narrow bins located anywhere in the input space, thus allowing local characterization of σ.

F.3 Estimation of β_1

In this section, we assume that σ is known, and examine two simple approaches to estimating β_1. The first is based on the well-known response-surface method, while the second is identical to that given for estimating σ in the preceding section.

In the response-surface method, a model of given structure is fitted to observed experimental data. The data is used to adjust model parameters, which are generally coefficients of terms in a polynomial model. A simple example is ordinary least-squares fitting of a line to noisy data.

To determine β_1, a model expressing θ_{out} in terms of θ_{in} and τ is needed, with β_1 as a linear parameter. For example, an appropriate model could be the one first given in equation (1.4-1):

$$\theta_{out}(\theta_{in}, \tau) = \beta_1 \theta_{in} + C\tau + r \qquad \text{(F.3-1)}$$

where β_1 and C are unknown constants and r is a noise term. By collecting triplets of data (θ_{out}, θ_{in}, τ), least-square-optimal estimates of β_1 and C can be easily determined. Because it is difficult to find an unbiased estimator for bounds on derivatives, the estimated value of C can be modified by "safety factors", and used as estimates of λ and ν.

The main drawback of this method is that an infinite number of models can be chosen, but without a clear idea of the correct model structure, the estimates obtained may be useless. As an example, consider the following case.

In *Mathematica*, a single bin of a system modeled by (F.3-1) was simulated, with $\beta_1 = 1$, $C = -0.3$, and $\sigma = 1.75$. The simulation starts from a control action of zero and executes 100 training cycles, all in the bin covering the input space from 15° to 18°. The 100 data points so obtained are then fit to two different models. The first is simply that of (F.3-1), while the second adds an unknown constant term. The entire process is repeated 30 times, to obtain an average measurement of the estimated β_1 and C.

For one particular run, the fit to the first model was:

$$\theta_{out}(\theta_{in}, \tau) = 1.01\,\theta_{in} - 0.31\,\tau \qquad \text{(F.3-2)}$$

The fit to the second model was:

$$\theta_{out}(\theta_{in}, \tau) = 0.95\,\theta_{in} - 0.31\,\tau + 0.95 \qquad \text{(F.3-3)}$$

In both cases, the estimates of β_1 and C are reasonable, but the constant term of (F.3-3) is not present in the system model of (F.3-1). This is the essence of the model-selection problem; data can be made to fit any model, but the resulting parameters may be meaningless.

The second method of estimating β_1 does not suffer from the difficulty of model selection. If an estimate of σ is known (perhaps through the process described in Section F.2), by fixing control action and taking sufficient data from a single bin, (F.1-1) yields:

$$\beta_1^2 = \frac{12}{w^2}\left(\hat{S}^2 - \hat{\mu}^2 - \hat{\sigma}^2\right) \tag{F.3-4}$$

Using the model of (F.3-1), this procedure was also simulated in *Mathematica*. First, 100 training cycles are used to train the single 15°-18° bin from an initial control action of zero. The control action is then fixed, and another 100 data points are taken to obtain an estimate of β_1. As before, the whole process is repeated 30 times. The average estimated value of β_1 was found to be 0.95. This value is close to the actual value of 1, and does not require selection of a response-surface model; however, this method also does not offer any estimates of λ and ν.

There is an additional problem: if a set of observed data is unusually noise-free (resulting in $S^2 \approx 0$), the right side of (F.3-4) can be negative, implying an imaginary value for β_1. Such estimation runs must be discarded.

Appendix G
Simulation Code

G.1 Introduction

All simulations used in this thesis were written in *Mathematica*, version 2.2. *Mathematica* was chosen because it allows easy symbolic manipulation of the analytical results of Chapters 2 and 3, and rapid prototyping and debugging of code.

Two programs are presented. The first, in Section G.2, simulates the deskewer (using a given analytic expression, rather than a physics-based model) and plots average values of average squared output versus number of bins, for use in the numerical optimization of the static structure. Functions that evaluate the theoretical predictions of Chapters 2 and 3 are also included, and are used in the plots that compare simulation results to theory.

The second program, in Section G.3, uses the same deskewer model to find average values of average squared output versus number of training iterations. It simulates both the static structure and the bin splitting controller, and plots the results for both cases. Again, symbolic functions are used to compare the theory of Chapter 3 to simulated experiment.

There are important differences between *Mathematica* programs and those of traditional languages such as *C*. Some of these differences are described below.

1. Normally, comments are marked with (* at the beginning and *) at the end. To facilitate reading the programs, and to allow room for more detailed explanations, statements in `Courier`, such as `Print`, are *Mathematica* code, while statements in *Times Italic, such as these words*, signify comments. Italicized comments must be removed or properly enclosed if the programs are entered as reproduced here.

2. *Mathematica* is an interpreted language, and unless explicitly marked for delayed evaluation (as explained in item 5, below), blocks of statements are executed immediately upon entry.

3. The code was written and executed on a Macintosh, via a "notebook" interface. This interface marks related blocks of code with brackets at the right-hand side of the page. Because these brackets are only displayed in the notebook, the blocks of code are denoted

by blank separator lines in the listings below. When typing them into a notebook, each block should receive its own bracket.

Blocks of code can also be denoted by separated pairs of parentheses, as in:

```
(
 Print["Hi!"];

 Print["Bye!"]
)
```

Note that parentheses-enclosed blocks in the programs may contain blank spaces, to improve readability.

4. Semicolons after a line suppress output in the notebook, if they are either at the end of a single line or after the last line of a block. They are also used to allow more than one statement in a block; in fact, within a block, lines must be ended with a semicolon to allow chaining of statements. Lines that do not end with a semicolon are either independent statements that do not suppress output, or the ending lines of blocks of chained statements.

5. Blocks that begin with a line of the form `functionName := xxx`, where `xxx` is other code, can be considered function definitions. The ":=" means that `functionName` is evaluated each time it is called, rather than only upon definition, as is the case with a "=" operator. Functions defined with ":=" are not initially evaluated upon typing.

6. In *Mathematica*, there is no "main" routine, as in a *C* program. Each block of code is an independent entity, and programs are written by placing calls to functions (both predefined and user-written) together in a block, separated by semicolons. This implies that all functions and variables are accessible to each other and are always present in the environment, unless explicitly defined as local or transient via such structures as `Module`.

   This lack of a "main" also implies that the order of supporting function entry can be somewhat haphazard, as long as a given function has been entered/defined before any later function attempts to execute it

7. In a real *Mathematica* session, all entered code is automatically marked with an input number, and all results return with an output number. The form of these tags is somewhat system-dependent, so they have been omitted from the programs.

8. Built-in functions have names that begin with a capital letter, and may have other capitals throughout the name (for example, "ToExpression"). In the programs below, user-written

functions and user-defined variables begin with a lowercase letter, but usually include a capital letter somewhere in the remainder of the name (such as "locateBin" and "binBoundaryInit").

9. *Mathematica* syntax is explained in [Wolfram 1991]. This reference book comes with every copy of the software, but is also available for purchase separately in bookstores.

G.2 Output Error vs. Number of Bins

This simulation assumes a nonlinear system model, as noted in the code. We vary the derivative $d\theta_{out}/dt$ by adding a sin function. The bounds of the derivative are now .1 and .5, with a mean of .3.

We also vary β_1 by adding a sinusoid, so it varies between 0 and 2. The period is a bit faster than that of the varying values of $d\theta_{out}/dt$; it's 1 rad/s.

We now have a normal input distribution. 2 $\sigma = 30°$

Functions used in the simulation

Reads in built-in functions that deal with statistics

```
  <<Statistics`ContinuousDistributions`
```

params={l->.1,nu->.5,g->5,B1->1,w->30,s->1.75} are assumed to be the "real" deskwer parameters

```
  params={l->.3,nu->.3,g->5,B1->1,w->30,s->1.75}
```

Delivers the skew of the incoming note. Other simulations use a uniform distribution. Here we've avoided using the built-in normal-distribution generator because for some reason, it's slow.

```
  getInput:=Sqrt[-2*Log[Random[]]] 15 Cos[Random[] 6.2831853]//Abs;
```

Initializes the locations of the boundaries between bins

```
  binBoundaryInit[n_]:= Range[0,n]*w/n/.params
```

Takes an input skew, and finds out which bin it falls into. The splitting simulation uses a more sophisticated but slower algorithm.

```
  locateBin[n_,numberOfBins_]:=Min[Floor[n*(numberOfBins/w)]+1,
      numberOfBins]/.params
```

Gets an input, finds the control action, applies both to the system model, then returns the input, output, and bin number of control action

```
  controlCycle[numberOfBins_]:=Module[{thetaIn,whichBin,whichControl,
      thetaOut},(
      thetaIn=getInput;
      whichBin=locateBin[thetaIn,numberOfBins];
      whichControl=control[[whichBin]];
      thetaOut=system[thetaIn,whichControl];
      {thetaIn,thetaOut,whichBin}
  )]
```

Given the input, output, bin number triple, this adjusts the control action
```
  controlAdjust[{thetaIn_,thetaOut_,whichBin_}]:=
      control[[whichBin]]+=thetaOut/g/.params;
```

System model. It's linear, plus the nonlinear terms mentioned above.
```
  system[input_,control_]:=Module[{systemOutput},(
      systemOutput=1*input+Sin[input]-.3*control+Sin[0.2*control];
      If[(control!=0),systemOutput+=
            Sqrt[-2*Log[Random[]]] s Cos[Random[] 6.2831853]/.params];
      systemOutput
      )]
```

Returns theoretical mean squared output for a given number of bins and a given final training iteration number
```
  staticBin[number_,finalIteration_]:=(
                      phi[0]=(1-1/g)^2;
                      1/12 B1^2 (w/number)^2 + s^2 +
                      (nu/g)^2/(1-phi[0]) *
                      (1-(1+(phi[0]-1)/number)^finalIteration) *
                      (1/12 B1^2 (w/number)^2 + s^2) +
                      (1+(phi[0]-1)/number)^finalIteration *
                      (w^2/3 - (w/number)^2/12))
```

Simulation

The number of iterations is the final iteration number of the training process. It's also the number of notes fed through the machine to evaluate its performance after training has ceased. The maximum number of bins sets the upper limit on how many bins we examine. The maximum number of repeats is how many times we do the whole process of training and testing for a given number of bins. We chose 5 here because the simulation takes so long.
```
  numberOfIterations=1000;
  maxNumberOfBins=51;
  maxNumberOfRepeats=5;
```

statistics is the record of measured output error in each bin. It has three indices. The first refers to the repeat number; the second refers to the number of bins in the control system during that trial; the third refers to each particular bin number. So, statistics[[2,9,6]] refers to the error measured in the 6th bin of the 2nd trial of a 9-bin control structure.
```
  statistics=Table[0,{maxNumberOfRepeats},{maxNumberOfBins},
      {maxNumberOfBins}];
```

  Same structure as the statistics array; holds the number of notes that fell into the bin
```
  howMany=Table[0,{maxNumberOfRepeats},{maxNumberOfBins},
      {maxNumberOfBins}];
```

  Begins the outermost loop, over the number of bins
```
  For[numberOfBins=1,numberOfBins<=maxNumberOfBins,numberOfBins+=1,
      (Print["Number of bins: ",numberOfBins];
```

    Begins the number of repeats loop
```
      For[numberOfRepeats=1,numberOfRepeats<=maxNumberOfRepeats,
            numberOfRepeats++,
```

      Clears the control action for this repeat
```
            (control=Table[0,{numberOfBins}];
```

Does the learning for numberOfIterations cycles
```
Do[(Module[{output},(
      output=controlCycle[numberOfBins];
      controlAdjust[output]
)]),{numberOfIterations}];
```

Does the evaluation of error for numberOfIterations cycles
```
Do[(Module[{output},(
      output=controlCycle[numberOfBins];
      statistics[[numberOfRepeats,numberOfBins,output[[3]]]]+=
            output[[2]]^2;
      howMany[[numberOfRepeats,numberOfBins,output[[3]]]] += 1;
)]) ,{numberOfIterations}];

Print["Number of repeats: ",numberOfRepeats]
```

`)];` *Closes the number of repeats loop*

Writes the simulation data to a file
```
Put[statistics,numberOfBins, UWS:User Folders:Ross:Temporary"];
```

`)]` *Closes the number of bins loop*

Sums and prints the statistics, showing measured average squared output for each bin. This summed data is then plotted against the theoretical predictions.
```
Print[ Apply[Plus,Apply[Plus,statistics],{1}]/
          (numberOfIterations maxNumberOfRepeats)];

data=Apply[Plus,Apply[Plus,statistics],{1}]/
          (numberOfIterations maxNumberOfRepeats);

Plot[staticBin[number,1000]/.params,{number,1,51}]

Show[%,ListPlot[data]]
```

G.3 Output Error vs. Number of Iterations

This simulation checks the predicted output error versus number of iterations, for both the bin splitting case and the static structure case.

This simulation assumes a nonlinear system model, as noted in the code. We vary the derivative $d\theta_{out}/dt$ by adding a sin function. The bounds of the derivative are now .1 and .5, with a mean of .3.

We also vary β_1 by adding a sinusoid, so it varies between 0 and 2. The period is a bit faster than that of the varying values of $d\theta_{out}/dt$; it's 1 rad/s.

We now have a normal input distribution. $2\ \sigma = 30°$

Parameters

Reads in built-in functions that deal with statistics
```
<<Statistics`ContinuousDistributions`
```

params={l->.1,nu->.5,g->5,B1->1,w->30,s->1.75,n0->0} are assumed to be the "real" deskwer parameters. n0 is the starting iteration of training.
```
params={l->.3,nu->.3,g->5,B1->1,w->30,s->1.75,n0->0};
```

Total number of training notes used in system
```
totalNumberOfNotes=1200;
```

Number of complete training runthroughs we do
```
numberOfRepeats=50;
```

How often we stop the training and do the evaluation of system output error
```
sampleSpacing=20;
```

Number of notes used to evaluate output error. No training is done on these notes.
```
numberOfEvalNotes=200;
```

Number of static bins in the static simulation
```
numberOfStaticBins=17;
```

Functions used in both bin splitting and static simulations

Delivers the skew of the incoming note. Other simulations use a uniform distribution. Here we've avoided using the built-in normal-distribution generator because for some reason, it's slow.
```
getInput:=Sqrt[-2*Log[Random[]]] 15 Cos[Random[] 6.2831853]//Abs;
```

Given the input, output, bin number triple, this adjusts the control action
```
controlAdjust[{thetaIn_,thetaOut_,whichBin_}]:=
     control[[whichBin]]+=thetaOut/g/.params;
```

System model. It's linear, plus the nonlinear terms mentioned above.
```
system[input_,control_]:=Module[{systemOutput},(
     systemOutput=1*input+Sin[input]-.3*control+Sin[0.2*control];
     If[(control!=0),systemOutput+=
          Sqrt[-2*Log[Random[]]] s Cos[Random[] 6.2831853]/.params];
     systemOutput
     )]
```

Takes an input skew, and finds out into which bin it falls. A more involved process than the first program, as the bin boundaries change location when splitting.

```
  locateBin[n_]:=If[n>=controlBoundary[[-1]],{Length[controlBoundary]-1},
       Position[Map[n >=# &, controlBoundary],False,{1}][[1]] -1]
```

Gets an input, finds the control action, applies both to the system model, then returns the input, output, and bin number of control action

```
  controlCycle:=Module[{thetaIn,whichBin,whichControl,thetaOut},
       thetaIn=getInput;
       whichBin=locateBin[thetaIn];
       whichControl=control[[ whichBin[[1]] ]];
       thetaOut=system[thetaIn,whichControl];
       {thetaIn,thetaOut,whichBin}//Flatten
  ]
```

Functions used only in the bin splitting problem

Takes the vector of control actions, doubles it, and puts the control action that was in one bin into two adjacent bins. Used when splitting.

```
  doubleControls:=control=Flatten[Map[{#1,#1}&,control]]
```

At splitting time, puts the new boundaries in. For example, the boundary vector {0,5,10} becomes {0,2.5,5,7.5,10}.

```
  doubleBinBoundaries:=Module[{i,temp},
       temp=Table[1,{2 Length[controlBoundary]-1}];
       For[i=1,i<Length[controlBoundary],i++,
              temp[[2i - 1]]=controlBoundary[[i]];
              temp[[2i]]=(controlBoundary[[i]]+controlBoundary[[i+1]])/2.0;
       ];
       temp[[2 Length[controlBoundary] -1]]=controlBoundary[[-1]];
       controlBoundary=temp
  ]
```

Functions for optimization of splitting times

This term appears often in the following functions, so we define it once here.

```
  errorFromWidth[generation_]:=1/12 B1^2 (w/2^generation)^2/.params
```

We want a general expression that will find the output error as a function of any number of splitting points. To do that, we'd like to use variables of the form n1, n2, n3, etc., and then feed the resulting expressions to the numerical optimizing function. There's no convenient way to make such a family of "subscripted" variables, so we use this kludge routine. We feed catVar the alphabetic and numeric parts of the variable, and it gives us back a concatenated expression that serves as a legitimate variable. There must be some better way to do this...

```
  catVar[n_,i_]:=ToExpression[ToString[n]<>ToString[i]]
```

ϕ_n is turned into a function

```
  phi[0]:= (1-1/g)^2/.params;
  phi[n_] := (1-(1/2^n) +(1/2^n) phi[0])/.params;
```

m2 is the theoretical expected squared mean and meanOutput2 is the expected squared output. With these functions, we can make recursive calls to find the general form for expected squared output error after any generation of splits.

```
  sbin2[n_]:=(errorFromWidth[n]+s^2)/.params;
```

```
startingError:=(B1 w/2)^2/.params;

m2Zero[generation_]:=meanOutput2[generation-1] -
    errorFromWidth[generation]-s^2;
m2Zero[0]:=startingError;

meanOutput2[generation_]:= errorFromWidth[generation]+s^2 +
    m2Zero[generation]phi[generation]^(catVar[n,generation+1]-
    catVar[n,generation])+ (1-phi[generation]^(catVar[n,generation+1]-
    catVar[n,generation]))/(1-phi[0]) (nu/g)^2 sbin2[generation]
```

Sets the initial search points and function to be optimized in the FindMinimum numerical optimization function

```
minPred[generation_]:=Prepend[
    Table[{catVar[n,i],10 2^i},{i,generation}],
    meanOutput2[generation]/.params/.
          {catVar[n,generation+1]->totalNumberOfNotes}];
```

Finds the maximum number of splits before the splitting iteration exceeds the available amount of training notes

```
howManySplits:=Module[{i},{
    i=1;
    While[(catVar[n,i]/.Apply[FindMinimum,minPred[i]][[2,i]])<
          totalNumberOfNotes,i++];
    i-1
}][[1]]
```

Plotting theoretical output of optimally-split system

```
maxNumberOfSplits=howManySplits;
```

Find the optimal splitting points

```
vectorOfSplits:=Module[{splitPoints,i},{
    splitPoints=Apply[FindMinimum,minPred[maxNumberOfSplits]][[2]];
    (Table[catVar[n,i],{i,1,Length[splitPoints]}]/.splitPoints)
    }][[1]];
```

This section of code plots the predicted squared output error

```
Module[{i,plotExpression,splitPoints},{
```

Add zero to the beginning and the last iteration to the end of vector of splitting points

```
    splitPoints=
          {-1}~Join~Round[vectorOfSplits]~Join~{totalNumberOfNotes};

    plotPoints={};
```

This loop goes through the different generations

```
    For[generation=0,generation<=maxNumberOfSplits,generation++,{
```

The theoretical expression for this generation, with the appropriate splitting points substituted in

```
          plotExpression=meanOutput2[generation]/.params/.
                Table[catVar[n,i]->splitPoints[[i+1]],{i,1,generation}];
```

This plugs specific iteration values into the plotExpression functions and concatenates the output with results from previous generations' results

```
                plotPoints=Append[plotPoints,
                      Table[plotExpression/.catVar[n,generation+1]->i,
                      {i,splitPoints[[generation+1]]+1,
                      splitPoints[[generation+2]]}]]//Flatten;
```

`}];` *Close generation loop*

Now we need to shift all the points over one to the left, as the first element of plotPoints is the output error at zero iterations. We also make each data point of plotPoints and an {x,y} pair so it can be graphed.

```
      plotPoints=Table[{i,plotPoints[[i+1]]},{i,0,totalNumberOfNotes}];

      ListPlot[plotPoints,PlotJoined->True]
```

`}]` *Close the plotting module*

Static bin boundary predicted error function

As before, returns theoretical mean squared output for a given number of bins and a given final training iteration number

```
   staticBin[number_,finalIteration_]:=(
                       phi[0]=(1-1/g)^2;
                       1/12 B1^2 (w/number)^2 + s^2 +
                       (nu/g)^2/(1-phi[0]) *
                       (1-(1+(phi[0]-1)/number)^finalIteration) *
                       (1/12 B1^2 (w/number)^2 + s^2) +
                       (1+(phi[0]-1)/number)^finalIteration *
                       (w^2/3 - (w/number)^2/12))
```

Optimization of static structure

We compile staticBin, for faster execution speed. We use the //N option because it stops Mathematica from using exact integer arithmetic later

```
   speedo=Evaluate[staticBin[bn,iterates]/.params]//N
```

This is the function that does the optimization of static structure for each iteration number

```
   optstatic = Table[FindMinimum[speedo/.iterates->iterator,
               {bn,1+.02 iterator}],{iterator,0,totalNumberOfNotes}];
```

This strips the variable name bn out of optstatic, and leaves a list of number pairs. The first element of the pair is the error, and the second is the optimal number of bins.

```
   optstatic = Map[{#[[1]],bn/.#[[2]]}&,optstatic];
```

Rounds the number of bins from optstatic to the nearest integer. Can't have fractional bin numbers.

```
   optstaticrounded=Map[{#[[1]],Round[#[[2]]]} &, optstatic];
```

Use the compiled version of staticBin (speedo) to find the output of the system given an integer number of bins, then plot

```
   optstaticroundedoutput =
      Table[{speedo/.{bn->optstaticrounded[[i,2]],iterates->i-1},
            optstaticrounded[[i,2]]}, {i,1,totalNumberOfNotes+1}];
   ListPlot[Table[{i,optstaticroundedoutput[[i,1]]},{i,1,1201}]];
```

Static simulation

Setup the size of the outputData table. Here, as in all variables below, the S suffix refers to the static simulation.

```
   outputDataS=Table[0,{numberOfRepeats}];
```

From here, all else gets enclosed for repeated evaluation. We're using a Do loop, so the number of repeats is at the bottom.

```
   Do[
```

Restart the control setup at the beginning of each iteration

```
        control=Table[0,{numberOfStaticBins}];
        controlBoundary=
              Table[i,{i,0.,w/.params,w/numberOfStaticBins/.params}];
```

Assess performance at the zeroth iteration

```
        Module[{output},
              output=0;
              Do[output+=((controlCycle[[2]])^2),{numberOfEvalNotes}];
              outputDataS[[repeatNumber]]={{0,output/numberOfEvalNotes}};
        ];
```

Start with the first note; after it's through, i still equals 1 (because it's a For loop), and it's fair to say that we've done "one" iteration

```
        For[i=1,i<=totalNumberOfNotes,i++,
```

Do a learning cycle

```
              Module[{output},
                    output=controlCycle;
                    controlAdjust[output];
              ];
```

If it's the right note, determined by sampleSpacing, evaluation output error

```
              If[Mod[i,sampleSpacing]===0,
                    Module[{output},
                          output=0;
                          Do[output+=
                                ((controlCycle[[2]])^2),{numberOfEvalNotes}];
```

Append the error measured at this data point

```
                          outputDataS[[repeatNumber]]=
                                Append[outputDataS[[repeatNumber]],
                                {i,output/numberOfEvalNotes}];
                    ] Closes the Module
              ,]; Closes the If
        ], Closes the For
   {repeatNumber,numberOfRepeats}]; Closes the Do loop
```

Average the various repeats

```
   averagedDataS=(Apply[Plus,outputDataS])/numberOfRepeats;
```

Save and plot he data

```
   Put[averagedDataS,"Control City:Applications:Mathematica 2.2.2:
        Ross's simulations:TemporaryS"];

   ListPlot[averagedDataS]
```

Splitting simulation

Find the splitting points
```
vectorOfSplits
splitPoints={0}~Join~Round[vectorOfSplits]~Join~{totalNumberOfNotes};
```

Setup the size of the outputData table
```
outputData=Table[0,{numberOfRepeats}];
```

From here, all else gets enclosed for repeated evaluation. We're using a Do loop, so the number of repeats is at the bottom.
```
Do[
```

Restart the control setup at the beginning of each iteration. It starts out as a single bin and is split at the appropriate times later.
```
    control={0};
    controlBoundary={0,w}/.params;
```

Assess performance at the zeroth iteration
```
    Module[{output},
          output=0;
          Do[output+=((controlCycle[[2]])^2),{numberOfEvalNotes}];
          outputData[[repeatNumber]]={{0,output/numberOfEvalNotes}};
    ];
```

In the splitting simulation, we loop over the number of generations. Within each generation, we loop over the notes that fall into that generation.
```
    For[generation=0,generation<=maxNumberOfSplits,generation++,
```

Loop over notes in generation
```
    For[i=splitPoints[[generation+1]]+1,
          i<splitPoints[[generation+2]]+1,i++,
```

Do a learning cycle
```
          Module[{output},
                output=controlCycle;
                controlAdjust[output];
          ];
```

If it's the right note, determined by sampleSpacing, evaluation output error
```
          If[Mod[i,sampleSpacing]===0,
                Module[{output},
                      output=0;
                      Do[output+=
                            ((controlCycle[[2]])^2),{numberOfEvalNotes}];
```

Append the error measured at this data point
```
                      outputData[[repeatNumber]] =
                            Append[outputData[[repeatNumber]],
                            {i,output/numberOfEvalNotes}];
                ]
```
Closes the Module

```
          ,];
```
Closes the If

```
    ];
```
Closes the For loop over notes of this generation

Now we split the bins, and the boundaries
```
        If[generation!=maxNumberOfSplits,
              doubleControls;doubleBinBoundaries;,]
        ],
```
Closes the For loop over number of generations

```
    {repeatNumber,numberOfRepeats}];
```
Closes the Do loop

Average the various repeats
```
  averagedData=(Apply[Plus,outputData])/numberOfRepeats;
```

Save and plot the data
```
  Put[averagedData,"Control City:Applications:Mathematica 2.2.2:
      Ross's simulations:Temporary"]

  ListPlot[averagedData]
```

References

Asada, H., and B.-H. Yang, "Skill Acquisition From Human Experts Through Pattern Processing of Teaching Data," *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 1302-1307 (1989).

Astrom, W., and B. Wittenmark, *Adaptive Control*, Addison-Wesley (1989).

Atkeson, C.G., and D. J. Reinkensmeyer, "Using Associative Content-Addressable Memories To Control Robots," *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 1859-1864 (1989).

Bain, M.E., "Machine-learned Rule-based Control," in J. McGhee *et al* eds.' *Knowledge-Based Systems for Industrial Control*, Peter Peregrinus (1990), pp. 222-243.

Berenji, H.R., and P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 724-740 (September 1992).

Duda, H., and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, NY (1973).

Gardner, W., *Introduction to Random Processes*, Macmillan, New York, NY (1986).

Gelb, A., *ed.*, *Applied Optimal Estimation*, MIT Press, Cambridge, MA (1974).

Girosi, F., and T. Poggio, "Networks and the Best Approximation Property," A.I. Memo 1164, MIT Artificial Intelligence Laboratory (October 1989).

Gray, R.M., *ed.*, "Special Issue on Quantization," *IEEE Transactions on Information Theory*, Vol. IT-28, No. 2 (March 1982).

Gullapalli, G., *et al*, "Acquiring Robot Skills via Reinforcement Learning," *IEEE Control Systems*, **Vol. 2**, pp. 13-24 (February 1994).

Heiss, M., "Inverse Passive Learning of an Input-Output-Map Through Update-Spline Smoothing," *IEEE Transactions on Automatic Control*, Vol. 39, No. 2, pp. 259-268 (February 1994).

Isaka, S., and A.V. Sebald, "An Optimization Approach for Fuzzy Controller Design," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 6, pp. 1469-1472 (November/December 1992).

Jang, J.-S., "Self-learning Fuzzy Controllers Based on Temporal Back Propagation," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 714-723 (September 1992).

Kawamura, S., and M. Nakagawa, "Memory-Based Control For Recognition of Motion Environment and Planning of Effective Locomotion," *IEEE International Workshop on Intelligent Robots and Systems (IROS)* (1990).

Kohonen, T., *Self-Organization and Associative Memory (3rd ed.)*, Springer-Verlag, Berlin, Germany (1989).

Kotovsky, J., *Design of a Bill-Deskewing Device for Automated Teller Machines*, Mechanical Engineering Bachelors' thesis, MIT, Cambridge, MA (May 1990).

Levinsky, R.B., *Design, Analysis, and Control of a Bill-Deskewing Device for Automated Teller Machines*, Mechanical Engineering Masters' thesis, MIT, Cambridge, MA (January 1992).

Levinsky, R.B., and H. West, "Design and Control of a Bill-Deskewing Device for Automated Teller Machines," *Proceedings of the 1993 JSME International Conference on Advanced Mechatronics - Tokyo, Japan*, pp. 402-407 (1993).

Michie, D., and R. Chambers, "BOXES: An Experiment in Adaptive Control," in E. Dale and D. Michie eds.' *Machine Intelligence 2*, Oliver and Boyd (1968).

Mikami, S., and Y. Kakazu, "Stochastic Rule Based Learning Controller For Manufacturing Environments," *Japan/USA Symposium on Flexible Automation 1992*, Vol. 2, pp. 1421-1426 (1992).

Nie, J., and D.A. Linkens, "Learning Control Using Fuzzified Self-Organizing Radial Basis Function Network," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 4, pp. 280-287 (November 1993).

Pal, N.R., *et al*, "Generalized Clustering Networks and Kohonen's Self-Organizing Scheme," *IEEE Transactions on Neural Networks*, Vol. 4, No. 4, pp. 549-557 (July 1993).

Poggio, T., and F. Girosi, "A Theory of Networks for Approximation and Learning," A.I. Memo 1140, MIT Artificial Intelligence Laboratory (July 1989).

Procyk, T.J., and E. H. Mamdani, "A Linguistic Self-organizing Process Controller," *Automatica*, Vol. 15, No. 1, pp. 15-30 (1977).

Schall, S., and C.G. Atkeson, "Robot Juggling: Implementation of Memory-Based Learning," *IEEE Control Systems*, pp. 57-70 (February 1994).

Sun, C.-T., and J.-S. Jang, "Adaptive Network Based Fuzzy Classification," *Japan/USA Symposium on Flexible Automation 1992*, Vol. 2, pp. 885-888 (1992).

Varsek, A., *et al*, "Genetic Algorithms in Controller Design and Tuning," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 5, pp. 1330-1339 (September/October 1993).

Wang, L.-X., and J.M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 6, pp. 1414-1427 (November/December 1992).

Whitehead, B.A., and T.D. Choate, "Evolving Space-Filling Curves to Distribute Radial Basis Functions Over an Input Space," *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, pp. 15-23 (January 1994).

Wolfram, S., *Mathematica: A System For Doing Mathematics by Computer (Second Edition)*, Addison-Wesley (1991).

Yager, R.R., and D.P. Filev, "Unified Structure and Parameter Identification of Fuzzy Models," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 4, pp. 1198-1205 (July/August 1993).

Ying, H., "Sufficient Conditions on General Fuzzy Systems as Function Approximators," *Automatica*, Vol. 30., No. 3, pp. 521-525 (1994).