Generalized Predictive Control Parameter Adaptation Using a Fuzzy Logic Approach

John William Lloyd

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Mehdi Ahmadian, Chair
Dan J. Inman
Adrian Sandu
Steve C. Southward
Saied Taheri

October 7, 2011
Blacksburg, Virginia

Keywords: Active Control, Adaptive Control, Generalized Predictive Control, Fuzzy Logic, Vibration Control

Generalized Predictive Control Parameter Adaptation Using a Fuzzy Logic Approach

John W. Lloyd

Abstract

A method to adapt the Generalized Predictive Control parameters to improve broadband disturbance rejection was developed and tested. The effect of the parameters on disturbance rejection has previously been poorly understood and a trial and error method was used to achieve adequate results. This dissertation provides insight on the effect of the parameters, as well as an adaptive tuning method to adjust them.

The study begins by showing the effect of the four GPC parameters, the control and prediction horizons, control weighting λ, and order, on the disturbance rejection and control effort of a vibrating plate. It is shown that the effect of increases in the control and prediction horizon becomes negligible after a certain point. This occurs at nearly the same point for a variety of λ's and orders, and hence they can be eliminated from the tuning space.

The control effort and closed-loop disturbance rejection are shown to be highly dependant on λ and order, thereby becoming the parameters that need to be tuned. The behavior is categorized into various groups and further investigated. The pole and zero locations of the closed-loop system are examined to reveal how GPC gains control and how it can fail for non-minimum phase plants.

A set of fuzzy logic modules is developed to adapt λ with order fixed, and conversely to adapt order with λ fixed. The effectiveness of the method is demonstrated in both numerical simulations and laboratory experiments.

Acknowledgements

I would like to thank the Structural Acoustics Branch of the NASA Langley Research Center for funding this research and taking me into their family. Special thanks to my mentors, Jer-Nan Juang, Gary Gibbs, and Ran Cabell, who guided me throughout this process. I truly appreciate their efforts. I would also like to thank Travis Turner for his contribution to this work, as well as for sharing his office with me. Thank you to Richard Silcox and Kevin Shepherd, who were my bosses at NASA, and who always kept me going forward.

I would like to thank Dr. Ahmadian, my adviser, for giving me this special opportunity to go to a NASA Research Center and earn my doctorate. The experience was invaluable. Thank you for your patience and helping me when I needed it the most.

Thank you to my parents, without whom none of this would have been possible. You were always there for me and I will always be grateful for that.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Generalized predictive control (GPC) was developed by Clarke et al. in the late 1980's [1, 2]. The algorithm sought to address problems that were present in other control laws and was intended as a fully adaptive control technique. GPC made it possible to control a plant that was both unstable and non-minimum-phase at the same time.[1] Furthermore, the control law could provide good closed-loop control even when the plant was overspecified or underspecified. Calculating the control law depends on four parameters: the control and prediction horizon, the control weighting λ, and the order of the controller. By judicious setting of these parameters, it was possible to generate vastly different control laws. One group of settings was intended for slow-moving process control, while other settings were intended for high bandwidth and high oscillation systems that are present in the control of vibrations.

This latter group, the control of vibrations, has recently been an active research area for the use of GPC, specifically, the vibration suppression in plates in which the motion is caused by an unknown broadband disturbance. Disturbances considered include acoustically-induced vibrations and excitations induced by a turbulent boundary layer present in a high-altitude, high-speed aircraft. Studies have shown that GPC is not only able to control this type of vibration, but can do so exceedingly well.

Previous studies also developed a new technique in system identification that automatically

[1] The system identification method we used in this work is not suitable for unstable plants.

includes the disturbance information into the plant parameters. GPC control laws based on this new technique have the ability to automatically reject disturbances in closed-loop mode, despite the fact that the disturbance is never directly measured. The amount of disturbance rejection proved to be heavily dependant on the GPC parameters settings. Although the previous work showed the efficaciousness of this method, there was no method for tuning the GPC parameters to induce to optimal disturbance rejection. To attain adequate disturbance rejection required a lengthly trial-and-error method, which simply implemented different GPC control laws and measured the result in closed-loop. The researchers had no prior knowledge of what was reasonably attainable, or of how to adjust the parameters in order to achieve it.

1.2 Objective

This research will address the need for a tuning method for the GPC parameters. Our focus is on the closed-loop disturbance rejection, called *the performance*, which is the reduction of vibration over the entire bandwidth with respect to the open-loop response.

Specifically, we wished to accomplish the following objectives:

- To study the effect of the four GPC parameters on the broadband disturbance rejection and the associated control cost;
- To categorize and explain the observed behavior. In particular, we show how variations in the parameters lead to possible performance improvements and changes in control effort. This includes an analysis of how GPC achieves control and an investigation into some of the most important effects of varying the parameters; and
- To use this knowledge to develop a fuzzy logic rules system to automatically adapt the GPC parameters to achieve the best possible disturbance rejection.

The use of fuzzy logic allows us to describe the adaptations we wish to perform at particular operating points in linguistic terms. This is particularly useful as the observed behavior can be fit into various groups, each described in linguistic terms, thereby making fuzzy logic a natural fit for the adaptation of the GPC parameters. The power of fuzzy logic is that these linguistic terms are

easily turned into mathematical operations, i.e. a decision surface, that can be highly non-linear and can contain complex behavior. Our final fuzzy logic rules and their corresponding decision surface demonstrate this complexity.

1.3 Approach

In order to accomplish our objectives, we use a dual testbed, the first being a laboratory experiment that consists of a thin plate clamped into a heavy and rigid frame. Disturbed by a shaker, the acceleration is measured at a specific point by a small accelerometer, and control actuation is achieved by a piezoelectric patch mounted in the center of the plate. The second testbed is a numerical simulation of the laboratory setup. The primary advantages of having a numerical simulation are the ability to rapidly run experiments, knowing the true plant, as well as the ability to simplify the plant if desired. Of course, our ultimate goal is to have the fuzzy logic adaptation successfully working on both testbeds.

Our first step is to generate closed-loop experiments that show the effect of the four parameters on the disturbance rejection performance. Since higher performances are usually achieved by higher control effort, we also characterize the effect of these parameters on the required control effort. This is also a fundamental goal of the fuzzy logic adaptation: to provide maximum performance while keeping the control effort below a maximum effort. Initially, we study the effect of the control and prediction horizons, while keeping λ and order fixed. After running a series of tests, we find that the horizons can be set to a specific number, which is related to the true system order, and that this setting will optimize the performance for a range of λs and order. Hence, the horizons can be eliminated from the tuning/adaptation space.

Next we study the effect of λ and order on the closed-loop performance and control effort. The effects prove to be very complex. For a fixed order, reductions in λ from an initial high value will provide better performance. In some cases, this effect has diminishing returns, while in other cases, the performance can actually get worse after reaching some local minimum. In all cases, it is possible to push λ too low, resulting in a closed-loop instability. Usually, the reductions in λ correspond to increases in control effort. In some cases, the effort actually levels off, while in other cases, the effort will increase exponentially. Keeping λ fixed and increasing the order results

in some of the same behavior, but with marked differences.

Using this knowledge base of behavior, we develop two sets of fuzzy logic adaptation rules. The first set keeps order fixed and adapts λ, while the second set fixes λ and adapts the order. Experiments show that these rules work effectively in both the numeric simulation and in the laboratory.

1.4 Outline

This dissertation is organized as follows. Chapter 2 gives a brief overview of the literature highlighting the need for and the development of GPC, as well as its place in the larger adaptive control field. We then summarize the system identification we have used throughout this work, as well as a derivation of the GPC control in terms of this system identification technique.

Chapter 3 discusses how the numerical simulations were performed. We have shown and defended our use of a specific modelling technique and highlighted a unique problem that occurs when converting the system from a continuous-time system to a discrete-time system. We compared our model to one derived by an independently-developed finite element model and to a tap test performed on the laboratory setup. Furthermore, we discuss the numerical simulation procedure and how it differs from the laboratory experiment. We have set up the simulations so that they would mimic the laboratory experiment as closely as possible.

Chapter 4 shows the complete setup for the laboratory experiments. The electrical connections as well as the physical setup are discussed. We describe the operation and control of the digital signal processing (DSP) chip and how it was used to provide closed-loop control. The complete laboratory procedure is also described.

Chapter 5 is a detailed look at the effect of the GPC parameters on the closed-loop performance and the corresponding control effort. We show experimental and numerical results for both the effect of the control and prediction horizon, and the effect of λ and order. We then categorize the behavior into various groups. One of these behaviors, the diminishing return on performance for reduction in λ, is investigated in detail in the frequency domain. In addition, we devote an entire section to the evolution of the control law and closed-loop system in the z-domain, giving us an insight into what GPC does as the control weighting λ is reduced to zero. The analysis will show

the cause of the closed-loop instabilities that always seem to occur.

Chapter 6 shows the complete derivation of the fuzzy logic rules used to adapt λ (with order fixed) or adapt order (with λ fixed). Three fuzzy logic modules are developed that examine the closed-loop operation with respect to one particular characteristic. The first module examines the current control effort and compares it to the maximum allowable. The second module looks for the point of diminishing returns. The third module only examines the closed-loop stability. A fourth module combines and prioritizes the first three modules and delivers a final adaptation parameter. Extensive results are given for both the numerical simulation and the laboratory experiment. Chapter 7 summarizes the work of the previous six chapters.

1.5 Contribution

We believe that this body of work is a significant contribution to the field of GPC tuning. We develop a technique that is based on the complexities and knowledge of a real-world plant. The use of fuzzy logic has been exploited to provide a complex and automatic method that replaces the former trial and error method. Although it was developed for a plate, we believe that the method will work for any modally dense and lightly damped structure.

Chapter 2

Background

2.1 Literature Review

Generalized Predictive Control (GPC) stems from an evolution of ideas that started with Astrom and Wittenmark's self-tuning regulators [3]. This method is for a single-input single-output (SISO) system with an unknown but constant plant and broadband, i.e. white, disturbance affecting the system. The plant is modeled via a finite difference equation with a built-in delay (k), i.e. the control input (u) at time t does not affect the output (y) until time $t + k$. The plant parameters are estimated with a least squares algorithm.

The finite difference equation is used to predict the values of $u(t + k)$ for a given $y(t + k)$, using a simple polynomial identity. The future value of the future output, $y(t+k)$, is then assigned a value of zero and the equation is solved for the control input u. The resulting equation is the control law. Put another way, the estimated system is evolved in a predictive manner to provide the first value of output y after the dead time and control input is used to set this value to zero. This type of control law is called the minimum variance.

Although the method has been used successfully for many practical applications, it has several problems. First, there is no way to limit the control input. Second, there is no way to force the output to follow non-zero set-points, as is often needed for practical systems. Lastly, plants with non-minimum-phase zeroes result in closed-loop instabilities.

To rectify these problems, Clarke and Gawthrop invented generalized minimum variance controllers, GMV [4, 5], which are a direct extension of Astrom and Wittenmark's self-tuning concept.

An auxiliary output Φ is created, consisting of a sum of filtered output, control input, and setpoint signals. It is this quantity that is predicted to the first time step after the dead time, and then minimized in order the calculate the control law.

This method has several advantages over the Astrom/Wittenmark controllers. First, the user-created filtering polynomials act as a control weighting, giving the user the ability to influence the control input levels. This is similar to the role of λ in GPC. Second, the filtering polynomials are present in the closed-loop characteristic equation, meaning that one can control the closed-loop dynamics. Additionally, the non-minimum-phase problems of the Astrom/Wittenmark regulator are resolved. Third, the algorithm has the ability to set the output to a non-zero setpoint.

Generalized predictive control law was developed by Clarke, Mohtadi and Tuffs to address the remaining shortcomings of generalized minimum variance, pole placement, and other control laws [1, 2]. The algorithm was developed to control plants that were non-minimum-phase and/or open-loop unstable, plants where the dead time was unknown or variable, and plants of unknown order. Based on the concepts of generalized minimum variance, the output of the system was predicted at a future time based on the knowledge of the current time step. However, instead of being predicted at one time step ($t + k$), the output was predicted over a range of time steps called the prediction horizon. The control law was calculated by minimized a cost function that includes a control weighting and the predicted output, as shown in Section 2.2.2.2.

This adaptive algorithm was shown to be superior to proportional-integral-derivative (PID) control, adaptive GMV, and adaptive pole placement in simulation studies in which the plant was drastically changed during closed-loop operation.

In the late 1990's, work was done by Juang and Eure at NASA that focused on using GPC and other predictive control laws in order to dampen acoustically-induced vibrations on a plate [6, 7]. The disturbance was left unknown, i.e. there was no measurable disturbance signal that could be feedforward. They developed a theory of how this disturbance information was incorporated in the system identification process and then used to enhance the disturbance rejection. This was done by leaving the disturbance on during the system identification data collection, e.g. the plate was excited by both the controller and disturbance at the same time.

For a single-tone disturbance at one frequency, it was shown that there exists a finite-difference model that directly maps the input to output, without knowing the exact nature of the disturbance.

This finite-difference model was slightly larger than that of the original plant. In the case of single-input single-output it would result in an increase order of two. Furthermore, the existence of such a finite-difference model was shown in the general case of excitation by a multiple-tone disturbance, with an corresponding increase in order for every disturbance frequency. In essence, the disturbance is implicitly embedded in the finite-difference model, and in such a way that the transfer function from the control input to the output is not altered in any way.

Hence, the system identification order must be chosen large enough to incorporate both the plant parameters as well as the disturbance information. In the case of a broadband excitation, which consists of a sum of infinite tones, there exists no theoretical finite-difference model that completely models both the plant and the disturbance information. However, by making the order sufficiently large, the effect can be approximated.

The system identification model is then used to calculate the predictive control law, and closing the loop results in disturbances being automatically rejected. Juang and Eure show that this works experimentally, and demonstrate that unknown disturbances consisting of multiple tones and/or a random broadband disturbance can be controlled.

This method is the basis of the work presented in this dissertation. Although previous research showed disturbance rejection, the GPC parameters were selected in a trial-and-error fashion and no effort was made to either optimize the disturbance rejection, or to keep track of the amount of control effort required. This dissertation will do just that.

Eure and Juang's work was further incorporated by Gibbs et al. in suppressing the sound radiation from an aircraft-style panel which is excited by a turbulent boundary layer [8]. The turbulent boundary layer is both spatially and temporally incoherent, making the induced broadband vibrations difficult to control. The technique incorporated the use of GPC and the earlier method of embedding the disturbance information into the plant. The sound emission was estimated with a technique called Radiation Modal Expansion (RME) and the use of an evenly-spaced array of accelerometers mounted to the panel. The overall use of the three techniques, GPC, RME, and embedded disturbance, proved very successful in controlling the radiated sound. The researchers again had to rely on a costly and lengthy method of trial and error to tune the GPC parameters, proving the need for the method developed in this dissertation.

Lin and Juang combined GPC with a fuzzy logic system in order to tune the control weighting

λ [9]. Their idea was the starting point for the work shown in this dissertation. However, their fuzzy logic system did not work in practice. The adaptation had no way of stopping, and λ would continually oscillate as the adaptation progressed. Absolute performance was used to drive the adaptation, which ignores the reality that performances are vastly different for both measurement location and controller order. The fuzzy logic system was not aware of the different types of performance and control-effort curves that will be described in detail in this dissertation. Their concept of using a performance and maximum control effort to drive the adaptation was used in our work, but we developed entirely different fuzzy logic rules, which were based on experimental and simulated behavior of the closed-loop plate.

The fuzzy logic system we have developed is one approach to the adaptation problem. Another possible approach is the gradient descent method. A complete description of the various gradient descent algorithms is given by Snyman [10]. The performance as a function of λ and system order would be the function to be minimized. The control effort could be used as a constraint on the optimization. Snyman describes several methods, such as spherical quadratic steepest descent method (SQSD), or the constrained optimization method Dynamic-Q, which may be useful for our adaptation problem. The main drawback with these methods is that the performance surface and its gradient are not known ahead of time. Each point on the performance surface requires collecting closed-loop data, which is not a quick calculation. Similarly, estimating the gradient would require the same performance calculation for a nearby operating point. The computational expense would certainly not be less than the fuzzy logic system we have developed.

Since GPC was introduced, there have been a few papers that have suggested how to tune the four parameters. These will now be summarized. The summary will use the following abbreviations: N_u is the control horizon, N_1 and N_2 are the minimum and maximum prediction horizons, respectively, and λ is the control weighting. Section 2.2.2.2 explains these terms in detail.

In 1991, McIntosh et al. [11] provide three tuning strategies for GPC. In each strategy, all parameters except one are fixed, while the remaining parameter was tuned.

The first tuning strategy varies N_2, the prediction horizon, to match the desired speed of response, i.e. the ability to force the output to respond to step changes in setpoint. The control horizon is set to one, making this method relevant only to industrial-type plants.

The second strategy tunes λ. The tuning starts with $\lambda = 0$, which results in a deadbeat con-

troller, and is then progressively detuned by increasing λ. Again, this approach is appropriate for the industrial setting.

The third approach is detuned model following. This is accomplished by filtering the output sequence. This in essence allows the user to place the poles at an arbitrary, yet predetermined position. McIntosh et al. show how to make this position equivalent to first- and second-order systems, whose closed-loop characteristics are well-defined. This method is also appropriate only for low-order industrial-type plants.

Using a state-space method, Elshafei et al. perform a stability analysis on simple plants under GPC control, with the control horizon set to one [12]. Analysis shows that small perturbation analysis can be used to determine the approximate closed-loop poles. This allows for a simple way to determine the effect of prediction horizon on the closed-loop stability. The usefulness of these properties is limited for our work, since our plant requires a control horizon much larger than one.

In a paper by Scattolini and Bittanti, the stability of three model-predictive control schemes, including GPC, are investigated as a function of prediction horizon [13]. The cost function of GPC is altered so that the output costing starts at the end of the control horizon, which is entirely different from the GPC algorithm used for our experiments. A stability analysis shows that by choosing the beginning and end of the cost horizon so that the sum of the step responses is positive, a stable closed system for λ larger than a critical value is guaranteed. This result is geared toward industrial-type plants that respond in a monotonic fashion.

In "A Tuning Strategy for Generalized Predictive Control" by Peng and Hanus, a tuning methodology is developed for N_u [14]. Specifically, the control and output sequences are filtered via a polynomial P, and the GPC control law is based on this augmented system. N_1, N_2, and λ are given values to provide deadbeat control. This filtering plus the deadbeat settings result in the closed-loop poles being placed at the zeros of the polynomial P. Hence this provides a pole-placement type of control, but uses the GPC control law instead of the usual approach. The control horizon N_u is the only tuned variable.

Converting the GPC law into a pole placement introduces the fundamental problem of pole-placement into GPC design. Specifically, an overparameterized system leads to equations that are unsolvable. Peng and Hanus sidestep this problem by tuning N_u to guarantee the inversion of $G^T G$. This is accomplished by developing a recursive formula by which the inversion is calculated by a

block matrix inversion formula. The formula is initiated for $N_u = 1$ and progressively increased. The recursion fails at some critical value of N_u and this value then becomes the final tuned N_u. Hence the tuning methodology provided by the authors is essentially a method to establish a bound on N_u that guarantees the existence of the inversion in the GPC control law calculation. Since our goal is to tune GPC to provide maximum disturbance rejection, and not pole placement, this study provides no insight on useful tuning rules for our case.

In a series of papers, Jun Zhang and Yugeng Xi examine the properties of the closed-loop under GPC control [15, 16, 17]. The primary assumption is that the model is exactly equal to the plant, i.e. no model-plant mismatch and no overparameterization. The authors convert the GPC control law into the IMC structure (Internal Model Control [18]) and derive the closed-loop transfer function. The resulting closed-loop equation gives three main results [15] . First, GPC law is based on zero-pole cancellation. The plant's poles are cancelled by the zeroes of the controller, and closed-loop poles are assigned by coefficient mappings. Second, the closed-loop characteristic equation is completely determined by the control and prediction horizons, as well as the control weighting λ. Third, the plant zeros cannot be altered in any way.

Since a characteristic equation is completely controlled by the GPC parameters, it is possible to reduce the order of the closed-loop system. Zhang and Xi derive tuning parameters under which this occurs, and the result depends on λ set to zero and the poles being driven to the deadbeat location [16]. However, this deadbeat setting is, in practice, very costly in terms of control effort, and is therefore not practical.

Again, these results are based on ideal conditions of no model-plant mismatch, no disturbance and no overparameterization. In Section 5.3, we will show how the the closed-loop pole locations evolve as λ is reduced in the case where there is a disturbance and the system identification is overspecified. It is true that GPC does try to cancel the poles as λ approaches zero, but the details will be explained in that section.

Clarke et al. provide numerous suggestions on how to set the GPC parameters in their original GPC paper [1, 2] and several other follow-up articles [19, 20, 21]. The advice is not always consistent. In [19], Clarke and Mohtadi provide several theorems on the stability and properties of GPC. The paper also provides a list of recommended settings for GPC:

1. $N_1 = n$

2. $N_2 \geq 2n - 1$

3. $N_u \leq n$

4. $\lambda = 0$ or very small number

As will be shown in later sections, these settings will not necessarily work for our system. Specifically, N_1 , the start of the costing, is always set to one in our case. This is somewhat irrelevant since this recommendation is largely based on avoiding the dead time, which is presumed to be less than n. For our system, we don't have any dead time, hence a setting of one is adequate.

The second recommendation, to set $N_2 \geq 2n - 1$, is loosely based on two theorems developed in the paper. Our experiments on the effect of the control and prediction horizon on the closed-loop disturbance rejection will result in a general rule of thumb that essentially matches this setting, but for different reasons.

The third setting, $N_u \leq n$ varies depending on the nature of the plant. For a simple process plant, setting N_u to one is adequate. For a plant with many unstable/underdamped poles, N_u should be set to the number of such poles. Our experiments will show that it is better for disturbance rejection to make this parameter large, approximately two to three times the plant order.

The fourth recommendation, to make $\lambda = 0$ or very small, cannot be closely followed due to the presence of non-minimum-phase zeros that result in closed-loop instability, as shown in Section 5.3. Furthermore, variations in λ are the primary tool for adjusting the balance between closed-loop performance and increases in control effort.

In "Generalized Predictive Control - Part II. Extensions and Interpretations" [2], Clarke et al. explore several issues, including several ways of "tuning" the control law. The GPC control is extended using techniques developed for use with the GMV (Generalized Minimum Variance) self-tuner, specifically, the concept of using a filtered output $\Psi(t)$ instead of the output $y(t)$ in the long-range prediction:

$$\begin{aligned} \Psi(t) &= P(q^{-1})y(t) \\ P(q^{-1}) &= \tfrac{P_n(q^{-1})}{P_d(q^{-1})} \end{aligned} \tag{2.1}$$

where P is the transfer function. Using this method allows GPC to act as a model-follower, i.e. the closed-dynamics are controlled by $M = 1/P$. Clarke envisions two uses for this P filtering. First,

for process application on a simple plant, P can be used to penalize the overshoot that occurs with step changes in the process setpoint. Second, the closed-loop can be made to act as M, which may be more desirable for high-performance applications. In order for this to work, N_u and N_2 are set equal to each other (and greater than the dead time of the plant), λ is set to zero, and N_1 is set to one.

The authors also derive conditions under which the closed-loop system becomes deadbeat, i.e. all poles are placed at the z-plane origin and therefore have no dynamics. The settings that provide this type of control are:

$$N_u = n \quad N_1 = n \quad \lambda = 0 \quad N_2 = 2n - 1$$

This type of control requires that the plant be completely controllable and observable, and results in a very active controller.

By combining the P filtering with the settings for deadbeat control, it is possible to turn GPC into a pole-placement law. Here, P is assigned a polynomial instead of a full transfer function, and the closed-loop poles are placed at the zeros of this polynomial. However, this approach suffers from the same problem as regular pole-placement, in that the equations become unsolvable for overparameterized plants. Hence, the identified model must have the correct plant order. This type of controller can be detuned, i.e. made less active, by making λ non-zero.

Furthermore, Clarke et al. investigate the similarity of GPC to the state-space LQ control law. The CARIMA equation can be converted to a state-space system:

$$\begin{array}{rcl} x(t+1) & = & Ax(t) + b\Delta u(t) + \tilde{w}(t) \\ y(t) & = & c^T x(t) + w(t) \end{array},$$

that is in observable canonical form. The GPC cost then becomes:

$$J = x(t+N_2)^T x(t+N_2) + \sum_{i=t}^{N_2+t-1} [x(i)^T Q x(i) + \lambda(i)\Delta u(i)^2]$$

Hence this is exactly the same form as a deterministic linear optimal regulator. The solution of this problem is a recursive formula calculated backward in time, i.e. starting with $t = N_2$, that results in a control law in the form of $\Delta u(t) = k^T \hat{x}(t|t)$. If N_2 is large enough, the solution process

Table 2.1: Clarke's GPC Tuning Overview

Nu	N_1	N_2	P	λ	Plant	Controller
1	1	10	1	0	s,d	"Default"
1	1	$\rightarrow \infty$	1	0	s,d	"Mean-level"
N_2	1	$\geq k$	P	0	mp	Exact model-following - P=1/M
$< N_2$	1	$\geq k$	P	0,λ		"Detuned" model-following
N_2	1	$\rightarrow \infty$	1	>0	s,d	LQ infinite stage
$N_2 - n + 1$	1	$\rightarrow \infty$	1	0	s,d	Cheap LQ
n	n	$\geq 2n - 1$	1	0	o,c	State Deadbeat
n	n	$\geq 2n - 1$	P	0	o,c	Pole Placement
n	n	$\geq 2n - 1$	P	λ	s,d	"Detuned" Pole Placement

s:stabilizable, d: detectable, o: observable, c: controllable, mp: minimum-phase

results in control gains k that stabilize as the index moves closer to the initial time. Hence the deterministic optimal regulator becomes the steady state LQR (linear quadratic regulator), which is solved by an algebraic Riccati equation.

However, the cost equation for a state-space system is identical to the GPC cost equation and the true minimum must therefore also be equivalent. Therefore, as the prediction horizon N_2 goes to infinity, and N_u is set to N_2 , the GPC control law approaches the steady state LQR control law.

Although Clarke makes this argument, it will be shown in this dissertation that GPC will intentionally create unstable closed-loop systems as λ goes to zero, but LQR always stabilizes the closed-loop system. Hence the similarity must end at some point. The various settings are all summarized in Table 2.1, which is reproduced from Reference [2].

As has been demonstrated, most of the previous work done on tuning GPC has no direct applicability to the type of tuning we require, i.e. tuning to reject an unknown broadband disturbance. The goal of this dissertation is to provide the base knowledge to allow for such tuning, and then to provide an adaptive fuzzy logic algorithm to accomplish this tuning automatically.

GPC is part of a larger adaptive control field that includes several subcategories. Isermann et al. [22] provide an excellent overview and classification of these fields. At the base level, the field is divided into feedforward and feedback controllers and includes the categories of gain scheduling, dual adaptive control, model reference adaptive control (MRAC), and model identification adaptive controllers (MIAC).

Gain scheduling is considered to be a feedforward controller by Isermann, in that the performance of the closed loop is not used to alter the controller gain. The gain, or more accurately, the controller parameters, are changed according to some external or indirect measurement of the plant operating condition. This measurement is called the scheduling variable. For a given operating condition, a suitable control law is calculated beforehand, and this calculation is repeated for the entire operating range. These control laws are then saved for later closed-loop implementation. While a plant is controlled in closed loop, the external measurement is used to characterize the operating state of the plant. As this state changes, the control law is automatically switched to the one that was pre-calculated for the given state. Since no further calculations need to be made for changes in the state, the adaptation is very rapid, making the gain scheduling method suitable for plants that undergo rapid and drastic changes.

Astrom and Wittenmark [23] give several examples of gain scheduling, including one of its first uses in the autopilots of military planes in the 1950's. These planes were designed for high altitude and high speeds, and the wide range of operating states induces large changes in the effectiveness of the control surfaces. Gain scheduling proved to be an effective tool for the task.

Rugh and Shamma [24] give a recent survey of research in gain scheduling as well as an historical overview. Although the idea of gain scheduling is very old, the literature before 1990 is relatively sparse, as most of the application areas were either proprietary or military in nature. The 1990's saw a revival in gain scheduling, as its use in nonlinear systems was considered.

Two particular methods were utilized in the linearization of the plant. The first method is the standard Jacobian linearization about a family of points. A second and newly-developed method is known as the quasi-LPV (linear parameter-varying) approach. This involves rewriting the plant dynamics so that the nonlinearity is hidden, and the new plant dynamics are strictly linear but time varying. Specifically, the nonlinear terms are replaced with a simple time-varying variable, and this new variable becomes one of the gain-scheduling variables.

An example of the quasi-LPV technique applied to integrated circuit (IC) manufacturing is given by Groot Wassink et al. [25]. The positioning of the wafer, which holds 80-200 IC's, must be precisely positioned in the wafer scanning machine during the photolithographic process, with tolerances in nanometers and microradians. The electro-mechanical systems that position the wafer in the machine have dynamics that vary depending on the position of the wafer. This

makes the problem a candidate for the gain-scheduling technique. The usual approach to the problem includes manually tuning a linear time-invariant controller based on the experimentally-identified frequency-response functions for each position of the wafer. The authors show that the LPV approach improves tracking errors, therefore increasing the throughput of the manufacturing process.

Model reference adaptive control (MRAC) uses a user-chosen reference model to specify the performance of the closed system. The desired setpoint signal is fed to both the controller acting on the plant, as well as to an input to the reference model. The difference of the output of the plant (in closed-loop) to the output of the reference model creates an error signal. The goal of MRAC is to drive this error to zero, so in essence, the closed-loop system acts as if it were the reference model. The error signal is fed into an adaptation algorithm that adjusts the control law parameters in order to induce a decrease in the error signal. Two main methods exist for this adaptation. The first method is a gradient method known as the MIT rule [26] and was developed for continuous-time systems. The parameter adjustment is by made by calculating the derivative of the error signal with respect to the parameter itself, i.e. the gradient, and then multiplying this number by a fixed number known as the adaptation constant. The second method uses a form of the Lyapunov stability theory in order to adapt the parameters, but does so in a manner that guarantees stability during the adaptation process [27, 28]. The method was later extended to discrete-time [29, 30, 31, 32]. The various techniques are further summarized by survey papers written by Landau [33] and Hang and Parks [34].

Sunwoo et al. use the MRAC technique for active suspension of a vehicle [35]. A sky-hook damping model is used as a reference model. The mass, stiffness, and damping rate of the sky-hook model can all be set by a user and embody preferred ride qualities. The authors envision a system in which a user will be able to choose from a predefined set of these characteristics by pushing a button on the vehicle's console. A simulation shows that active suspension improves the sprung mass acceleration by 30% to 50% over a passive system. Furthermore, the controller makes the closed-loop system invariant to changes in the sprung load or changes in the suspension characteristics.

Lin and Chen control a four-bar linkage with an MRAC controller [36]. Linkages are often used in the design of machinery in order to perform certain tasks such as path generation or rigid

body guidance. The kinematic movement and weight of the linkages make the system highly nonlinear, making it difficult to control the speed of the linkages with a motor. The study proposes a second-order reference system, used in conjunction with a nonlinear auxiliary feedback term based on the reference system, in order to provide closed-loop control. The effectiveness of the method was demonstrated on an experimental model. Results showed the ability to control the linkage at constant radial velocity as well as commanding the linkage to follow a variable velocity setpoint.

Dual control was developed by Feldbaum [37, 38] in 1960. The controller attempts to achieve two goals simultaneously. The first goal is to control the process, while the second goal is to inject a signal into the system that will cause the system identification parameters to converge to their true values as rapidly as possible. This is called active probing. Unlike MRIC, there exists no separate step for system identification – the control law essentially does both at once.

The optimal dual controller is calculated by minimizing a loss function, which is based on the expected values of future, i.e. unknown, system outputs. It is similar to a predictive loss function in that it contains the sum over a control horizon. Unlike a predictive control, the certainty of the parameters is included in the calculation, thereby giving the control law its dual properties. The solution is computed by dynamic programming and results in the so-called Bellman equation. This equation includes a nested minimization and a mathematical expectation, and it can only by solved numerically. However, the difficulty is so great that even this becomes impractical.

Only a few studies have attempted the optimal solution. Astrom and Helmersson [39] show the solution for a plant that consists only of an unknown integrator with an unknown gain. The complexity of the paper for such a simple plant demonstrates the underlying difficulty of the optimal dual control problem. The advantages of a dual controller are also shown. Specifically, it is shown that the dual control law automatically switches on its active probing mode when the parameter estimates are of poor quality. Sternby [40] gives an analytical solution to a simple dual control problem, in which the plant is represented by a Markov chain with four states.

Various methods have been investigated for approximating dual control or solving the equation suboptimally. Several survey papers summarize the work. Wittenmark's [41] short overview classifies the methods into six categories. The first category includes adding a perturbation, i.e. probing, signal to a cautious controller. In the second category, the loss function is limited to a

one-step-ahead prediction. The third method approximates the Bellman equation with its serial expansion. The loss function is slightly modified for the fourth method, making it easier to solve. In the fifth category, the mathematical expectation is replaced by a summation by considering only finite parameter sets. Finally, methods from robust design are introduced. Filatov and Unbehauen give a complete history of the suboptimal dual controllers [42]. They demonstrate that the innovations can be split into two main groups: implicit dual control and explicit dual control. A detailed mathematical analysis and numerical simulations of several approximation methods are given by Lindoff et al. [43].

Allison et al. [44] apply suboptimal dual active control to a wood chip refiner. The refiner uses spinning plates to turn wood chips into pulp. The distance between the plates, controlled by hydraulics, determines the motor load and quality of the product, but the relationship is both nonlinear and time-varying. In general, the motor load goes up as the distance between the plates decreases. However, a minimum distance is reached after which the motor load quickly decreases and pulp gad decreases to the point where the plates collide. These highly undesirable conditions, known as pulp pad collapse and plate clash, are to be avoided as much as possible. The critical gap wanders unpredictably during the use of the machine. The goal of the control law is to control the motor load at a given set point and to avoid the plate crash.

Wittenmark and Elevitch use an active suboptimal dual controller (ASOD) [45] in order to control the process. The cost function contains a probing weight called λ, which determines the extent at which the controller applies a probing signal to the plant. The control strategy successfully controlled the chip refiner, and it was shown that setting λ for higher levels of probing further improves the closed-loop results, i.e. less pulp pad collapse. Dual control proved especially helpful during the startup of the machine, during which the control law automatically uses high levels of probing in order to quickly estimate and control the plant.

Model identification adaptive controllers (MIAC) are the last main group of controllers. In general terms, the calculation of the control law is a two-step process that occurs during closed-loop operation. The first step, system identification, collects the output and input measurements and feeds them into the parameter identification algorithm. These algorithms are typically recursive, which reduces the computation time. Commonly-used techniques include recursive least squares (RLS), recursive extended least squares (RELS), and recursive maximum likelihood (RML). In

the second step, the updated system identification parameters are used to calculate the control law. Hence this calculation assumes that the identified parameters are equal to the true system parameters. This is known as the certainty equivalence principle. This assumption is not always correct; in fact, during system startup or during the changes in the plant, the estimated parameters will not be equal to the true parameters. In the best-case scenario, further closed-loop operations result in the estimated parameters converging to their true values. Typically, the two steps are carried out after each sampling instant.

This extensive group of controllers includes the previously-mentioned minimum variance and generalized minimum variance, adaptive pole placement, LQG state-space techniques, and model predictive control laws such as GPC. Although the idea for MIAC goes all the way back to Kalman in 1958, the 1973 minimum variance strategy of Astrom and Wittenmark is one of the first examples of this type of control. The controllers are also referred to as "self-tuning" or certainty equivalency controllers. Although they were first intended to control systems that were unknown but time-invariant, the system identification algorithms were modified to include a forgetting-factor that allowed the techniques to work on unknown and time-varying systems. Since the system identification is completely separate from the control law calculation, it is possible to combine different system identification approaches with the same control law, or vice versa.

A survey and analysis paper written by Astrom et al. [46] in 1977 discusses the RLS, ELS, and RML identification techniques and the minimum variance (MV) and linear quadratic (LQ) controllers. Some preliminary theoretical conditions under which the identified parameters converge are presented. It is shown that the ELS method in particular does not always converge to the true values when coupled with the self-tuning method. Numerical simulations show the various methods controlling a simple plant. Some examples of the use of self-tuning in industrial applications are given, specifically, in the control of a paper machine and an ore crusher. A much broader survey of the entire adaptive control field was published by Astrom in 1983 [47]. A discussion of the various issues and precautions that must taken to use self-tuning on a real-world plant was done by Wittenmark and Astrom [48].

Wellstead et al. developed the pole placement self-tuning regulator [49] in order to address some of the shortcomings of the MV/GMV method. In particular, the pole placement controller works well for non-minimum-phase zeros that give the other methods trouble. The controller

places the closed-loop poles at particular locations, which are pre-defined by the user. These poles are chosen to give a desirable closed-loop effect, for instance, a response with very little overshoot or control signals that do not oscillate excessively (process control). The method uses recursive least squares to estimate the system parameters. A simulation study shows that the technique has the ability to control non-minimum-phase systems with variable time delay. Astrom and Wittenmark [50] extend this idea to the servo problem, i.e. the output of the system is forced to follow a non-zero reference signal.

Many authors have investigated the combination of LQG and self tuning. The advantage of using this method is that the closed-loop system is stable for all control weighting values if the plant is known, even if the plant has non-minimum-phase zeros. This is in direct contrast to the minimum variance techniques. Grimble [51] derives a polynomial approach to the LQG calculation. His method uses the solution of two Diophantine equations and spectral factorization, and has computational advantages over previous methods. The system parameters are estimated by the extended least squares method.

Clarke et al. provide a detailed study of a LQG self-tuner based on the state-space approach [52, 53], which was published only two years priors to Clarke's GPC papers. Much like GPC, an N-stage cost function base is minimized in order to provide the control law. It also uses the auxiliary output technique that Clarke introduced in his earlier GMV method and recursive extended least squares (RELS) technique to estimate the system parameters. The control calculations are much more involved than those of GPC. For each time step, the plant parameters are estimated and then input into state-space form, the Riccati equation is iterated to calculate the Kalman gain vector, the states are reconstructed using the transmittance matrix, and the control value is then calculated as a matrix multiplication of the gain vector and the state matrix. Nevertheless, the algorithm provides very good closed-loop control. In the simulation study [53], it is shown that it performs better than the pole-placement technique when the plant is overspecified. It is also demonstrated that the LQG self-tuner can provide good control of a non-minimum-phase plant, remains stable when the plant is underparameterized, can handle underestimated time-delays, and provides outputs that do not have steady-state errors. This algorithm is a stepping stone between Clarke's earlier GMV work and his subsequent GPC algorithm.

A subset of MIAC is the field of model predictive controllers (MPC). This field also contains

GPC. All MPC's have the same basic algorithm. The plant is modeled explicitly, and the output of the plant is predicted into the future over a finite interval (prediction horizon) due to a yet-unknown control action. The difference between the plant output and a reference signal and the weighted control action are fed into a cost equation. The optimal solution of the cost equation is calculated, and the resulting equation gives the future control action over a finite time interval, i.e. the control horizon. Only the first control action, corresponding to the next time interval, is used to actuate the plant. The rest are discarded. At the following time step, the process of optimizing the cost equation is repeated. Therefore, the control and prediction horizons recede into the future, and hence MPC's are sometimes referred to as receding horizon controllers. The differences in the various model predictive controllers occur in what technique is used to model the plant, and the specific cost and various techniques for placing constraints on the outputs or inputs.

Model predictive controllers were developed mainly by the process control industry, with the primary innovations aimed at solving problems that occur in the particular applications of that industry. MPC's were first introduced by Richalet et al. in the late seventies [54]. Their algorithm was called model predictive heuristic control (MPHC) and was later referred to as IDCOM (identification and command) or MAC. The plant was modelled with an impulse response model, and a reference trajectory was used to describe the desired plant output. The method was successfully used to control a PVC plant, a distillation column in an oil refinery, and a steam generator in a power plant.

A second MPC algorithm was developed around the same time frame by Shell Oil engineers, Cutler and Ramaker, and was called dynamic matrix control (DMC) [55]. It models the plant via a step-response model and tries to drive the plant to a desired setpoint. The method proved superior to conventional PID controllers in the control of temperatures in a furnace.

Qin and Badgwell provide an excellent history of MPC and their evolution into current-day commercially available products that are commonly used in industrial settings [56]. DMC and IDCOM are considered the first generation MPC, with current products being the fourth generation. The second generation algorithm, QDMC, reformulated the DMC into a quadratic programming approach [57]. This made it possible to include hard constraints into the problem solution, which was only possible in an ad-hoc manner in the previous generation. Effective constraint handling is one of the main driving forces in the evolution of industrial MPC algorithms, since the most

economical plant operations occur when the inputs and outputs are driven as close as possible to constraints without violating them.

Third-generation products sought to solve two main concerns. First, QDMC does not always provide a mathematically feasible result, hence some method was needed to recover from that condition. Second, the previous algorithms required that constraint violations were given relative weight in the objective function in order to prioritize their importance. It was difficult to assign these weights, and their inclusion does not always provide the ideal performance. The new algorithms provided the ability to include hard and soft constraints, a method to rank constraints in order of importance, and multiple objective functions. Algorithms included in this generation are IDCOM-M [58] and SMOC [59] as well as others.

Fourth generations, i.e. current, commercial MPC products, include a wide range of techniques for handling and ranking constraints and numerous methods for modeling the plant. Qin and Badgwell provided a detailed list of which methods are used for each specific product. Linear plant models include finite impulse response (FIR), finite step response (FSR), state-space, auto-regressive with exogenous input (ARX), and transfer functions. Nonlinear plant models that are used include nonlinear neural net, nonlinear state-space, and static nonlinear polynomial, as well as plant equations derived from first principles.

Commercial MPC products at the time of the survey still had several problems. The methods that used FIR and FSR were not able to deal with unstable processes or processes with fast dynamics that would necessitate a high model order. Many methods have sub-optimal feedback and lack nominal stability. Tuning the MPC controllers still remains difficult and requires the use of lengthy closed-loop simulations. Model uncertainty is not widely used in the design of the control law. Ideas developed by the academic community that address some of these issues had not been incorporated into the available products. Most algorithms are not fully adaptive in the sense that they automatically adjust to variations in the plant parameters. Qin and Badgwell cite GPC as one of the proposed fully-adaptive algorithms, but it had not been used in any commercial MPC products.

Generalized predictive control has been used in a wide variety of applications. Anwar studied the use of GPC in controlling the yaw error in a brake-by-wire automobile [60]. GPC was used to selectively apply the brakes when an understeer or oversteer condition was detected during steering

maneuvers. The method was tested on a car driven on a snow-covered road and was shown to dramatically improve the steering performance.

The use of GPC in a solar energy collection field is investigated by Camacho et al. [61]. A series of parabolic mirrors reflect solar energy onto a pipe filled with oil. The oil heats up and is moved into a system that turns the heat into electricity. The goal of the controller is to speed up or slow down the flow oil rate in order to keep the outlet temperature constant. GPC was shown superior to adaptive PID and adaptive pole-placement control.

Park et al. [62] used GPC to control Henon- and Lozi-type chaotic systems. The plant is modeled by a nonlinear autoregressive moving average with exogenous input (NARMAX) equation. GPC was able to provide faster settling than typical one-step-ahead (OSA) controllers. Furthermore, GPC was able to control for a wider range of initial values of the chaotic system, therefore making it more robust.

Mahfouf et al. used GPC to automatically administer the anesthetic drug isoflurane to human patients while in surgery [63]. The controller successfully keeps the mean arterial pressure (MAP) at a certain low value in order to insure that the patient is sedated without being aware.

Geng and Geary used GPC in an air-handling plant, i.e. a machine that regulates the temperature inside a building [64]. The plant is nonlinear, but the parameters change slowly enough that it can be represented with a time-varying linear process. Their method uses a backpropagating multilayer neural network in order to improve the RLS system identification.

Low et al. evaluated the performance of GPC in controlling brushless DC (direct current) drives [65]. DC drives are used in a large variety of fields including robotics, aerospace, and military applications. It was shown that GPC successfully tracks setpoint changes and performs well when the load changes unexpectedly. Furthermore, the control law works effectively even when the main motor parameters, i.e. winding resistance, inductance, or viscous-friction coefficients, were inaccurate by large margins.

2.2 Experimental System Identification and Generalized Predictive Control

Generalized predictive control is based on the predicted output of a plant. In order to predict the future output of the plant, we need a model of the plant. There are two different ways of arriving at such a model. The first way is to derive the model analytically. The plant is described using fundamental principles that govern the behavior of the plant in general. Such a characterization often leads to a system of differential equations. This system of equations can be solved in various ways. The solution can then be recast in such a way that we can design a generalized predictive controller. The accuracy of such characterization is not always good enough to ensure the closed-loop stability of the controller. One can imagine the phase error induced by miscalculation of a natural frequency by only a few percent for lightly damped systems.

We are then left with the second way of arriving at a model of the plant. This is the technique of experimental system identification. The plant, actuator, and sensor are set up in the configuration in which we wish to control the plant. Instead of sending a control signal to the actuator, we send a random broadband signal. As we send this random signal to the actuator, we record both the response of the plant as well as the random signal. Using this data, we can calculate a model of the plant. This model is more useful than its analytical cousin because it represents not only the dynamics of the plant itself, but also the dynamics of the actuator, the sensor, and the characteristics of the various filters in the control path.

2.2.1 System Identification

Techniques presented in this section are a collection of ideas presented in [66].

The ultimate goal of system identification is to quantify the A, B, C and D matrices of the discrete state-space equations

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y &= Cx(k) + Du(k) \end{aligned} \tag{2.2}$$

using experimental data. To design a generalized predictive controller, we do not need to know the

full state-space equation but instead only need to have the ARX (Auto-Regressive moving average with eXogenous input) model of the plant:

$$\begin{aligned} & y(k) + \alpha_1 y(k-1) + \cdots + \alpha_p y(k-p) \\ & \qquad = \beta_0 u(k) + \beta_1 u(k-1) + \cdots + \beta_p u(k-p) \end{aligned} \tag{2.3}$$

The Observer/Kalman filter identification technique (OKID) described by Jer-Nan Juang [66] first calculates this ARX model as an intermediate step and then uses it to calculate the A, B, C and D matrices of the state-space system. The OKID technique is therefore ideally suited for the purposes of GPC design. We will now show how we can calculate the α's and β's of the ARX equation. We start with the state-space equation and add and subtract a $Gy(k)$ term:

$$\begin{aligned} x(k+1) &= A(k) + Bu(k) + Gy(k) - Gy(k) \\ &= (A+GC)x(k) + (B+GD)u(k) - Gy(k) \end{aligned}$$

This can be rewritten as:

$$x(k+1) = \overline{A}x(k) + \overline{B}v(k) \tag{2.4}$$

where

$$\begin{aligned} \overline{A} &= A + GC \\ \overline{B} &= \begin{bmatrix} B+GD & -G \end{bmatrix} \\ v(k) &= \begin{bmatrix} u(k) \\ y(k) \end{bmatrix} \end{aligned} \tag{2.5}$$

By rewriting the equation in this way, we have defined an observer for the system. This can be shown by considering the state of Equation (2.4) as the estimated state, and the state of Equation (2.2) as the true state. The error decays at a rate defined by the eigenvalues of $\overline{A}$. These eigenvalues can be arbitrarily set by the gain matrix G. We will see that the best choice for G is one that makes the observer deadbeat, meaning that $\overline{A}^p = 0$, where p is the dimension of the $\overline{A}$ matrix. We now

come up with a series of equations that describe the evolution of the states for any input $u(k)$:

$$\begin{aligned}
x(k+1) &= \overline{A}x(k) + \overline{B}v(k) \\
x(k+2) &= \overline{A}x(k+1) + \overline{B}v(k+1) \\
&= \overline{A}^2 x(k) + \overline{A}\,\overline{B}v(k) + \overline{B}v(k+1) \\
x(k+3) &= \overline{A}x(k+2) + \overline{B}v(k+2) \\
&= \overline{A}^3 x(k) + \overline{A}^2\overline{B}v(k) + \overline{A}\,\overline{B}v(k+1) + \overline{B}v(k+2) \\
&\vdots \\
x(k+p) &= \overline{A}x(k+p-1) + \overline{B}v(k+p-1) \\
&= \overline{A}^p x(k) + \overline{A}^{p-1}\overline{B}v(k) + \overline{A}^{p-2}\overline{B}v(k+1) + \ldots \\
&\quad + \overline{B}v(k+p-1)
\end{aligned}$$

Using the output equation to predict the output at this time step:

$$\begin{aligned}
y(k+p) &= Cx(k+p) + Du(k+p) \\
&= C\overline{A}^p x(k) + C\overline{A}^{p-1}\overline{B}v(k) + C\overline{A}^{p-2}\overline{B}v(k+1) + \ldots \\
&\quad + C\overline{B}v(k+p-1) + Du(k+p)
\end{aligned} \tag{2.6}$$

By substituting for v, we see that this equation has the general form of the ARX equation except for the addition of the $C\overline{A}^p x(k)$ term. This term describes the effect of the initial condition of the plant. We can gather even more information by looking at the next time step:

$$\begin{aligned}
x(k+p+1) &= \overline{A}x(k+p) + \overline{B}v(k+p) \\
&= \overline{A}^{p+1}x(k) + \overline{A}^p\overline{B}v(k) + \overline{A}^{p-1}\overline{B}v(k+1) + \overline{A}^{p-2}\overline{B}v(k+2) + \ldots \\
&\quad + \overline{A}\,\overline{B}v(k+p-1) + \overline{B}v(k+p) \\
y(k+p+1) &= Cx(k+p+1) + Du(k+p+1) \\
&= C\overline{A}^{p+1}x(k) + C\overline{A}^p\overline{B}v(k) + C\overline{A}^{p-1}\overline{B}v(k+1) + C\overline{A}^{p-2}\overline{B}v(k+2) + \\
&\quad + C\overline{A}\,\overline{B}v(k+p-1) + C\overline{B}v(k+p) + Du(k+p+1)
\end{aligned} \tag{2.7}$$

If we define the observer so that it is deadbeat, then $\overline{A}^p = 0$, and Equations (2.6) and (2.7) reduce to:

$$\begin{aligned} y(k+p) &= C\overline{A}^{p-1}\overline{B}v(k) + C\overline{A}^{p-2}\overline{B}v(k+1) + \dots \\ &\quad + C\overline{B}v(k+p-1) + Du(k+p) \\ y(k+p+1) &= C\overline{A}^{p-1}\overline{B}v(k+1) + C\overline{A}^{p-2}\overline{B}v(k+2) + \dots \\ &\quad + C\overline{A}\,\overline{B}v(k+p-1) + C\overline{B}v(k+p) + Du(k+p+1) \end{aligned}$$

Clearly these equations are identical except for the one-time step shift for the second equation. This development shows that after the p'th time step, the output can be predicted without knowledge of the initial condition $x(k)$ with an equation of the form:

$$y(k+p) = C\overline{A}^{p-1}\overline{B}v(k) + C\overline{A}^{p-2}\overline{B}v(k+1) + \cdots + C\overline{B}v(k+p-1) + Du(k+p). \quad (2.8)$$

Substituting for v and $\overline{B}$ from Equation (2.5), this equation can be written as:

$$\begin{aligned} &y(k) + CGy(k-1) + C\overline{A}Gy(k-2) + \cdots + C\overline{A}^{p-1}Gy(k-p) \\ &= Du(k) + C[B+GD]u(k-1) + C\overline{A}[B+GD]u(k-2) + \cdots + C\overline{A}^{p-1}[B+GD]u(k-p), \end{aligned} \quad (2.9)$$

which is the ARX equation. Again, it is important to note that this equation is valid for all time steps after and including the p'th time step. For a single-input single-output (SISO) system, we have one equation and $2p+1$ unknown coefficients. To solve for the coefficients, we take our experimental data and use Equation (2.8) to assemble a series of equations:

$$\overline{y} = \overline{Y}\,\overline{V}$$

where

$$
\begin{aligned}
\overline{y} &= \left[\begin{array}{cccc} y(p) & y(p+1) & \ldots & y(l-1) \end{array}\right] \\
\overline{Y} &= \left[\begin{array}{cccc} D & C\overline{B} & C\overline{A}\,\overline{B} & \ldots & C\overline{A}^{p-1}\overline{B} \end{array}\right] \\
\overline{V} &= \left[\begin{array}{cccc} u(p) & u(p+1) & \ldots & u(l-1) \\ v(p-1) & v(p) & \ldots & v(l-2) \\ v(p-2) & v(p-1) & \ldots & v(l-3) \\ \vdots & \vdots & \ddots & \vdots \\ v(0) & v(1) & \ldots & v(l-p-1) \end{array}\right],
\end{aligned}
\tag{2.10}
$$

and l are the number of data samples collected. For a SISO system, we have l equations and $2p+1$ unknowns. In general, we will collect enough data samples so that $l \gg 2p+1$. We will therefore use the least squares technique to solve for $\overline{Y}$:

$$
\overline{Y} = \overline{y}\overline{V}\left[\overline{V}\,\overline{V}^{T}\right]^{\dagger}, \tag{2.11}
$$

where the † symbol stands for the pseudo-inverse. Looking at Equations (2.8) and (2.9), we see that the $\overline{Y}$ matrix contains the ARX coefficients. Recognizing that $\overline{B}$ consists of two parts as shown in Equation (2.5), it is easy to see how me must partition the $\overline{Y}$ to get the ARX coefficients.

2.2.2 Generalized Predictive Control (GPC)

This section is adapted from [67].

2.2.2.1 Output Prediction

We start off with the ARX equation slightly rewritten:

$$
\begin{aligned}
y(k) &= -\alpha_1 y(k-1) - \cdots - \alpha_p y(k-p) \\
&\quad +\beta_0 u(k) + \beta_1 u(k-1) + \cdots + \beta_p u(k-p)
\end{aligned}
\tag{2.12}
$$

and then move the whole equation one time-step forward:

$$
y(k+1) = -\alpha_1 y(k) - \alpha_2 y(k-1) - \alpha_3 y(k-2) - \cdots - \alpha_p y(k-p)
$$

$$+\beta_0 u(k) + \beta_1 u(k-1) + \cdots + \beta_p u(k-p) \tag{2.13}$$

Plugging Equation (2.12) into (2.13), we get:

$$\begin{aligned} y(k+1) &= -(\alpha_2 - \alpha_1\alpha_1) y(k-1) - (\alpha_3 - \alpha_1\alpha_2) y(k-2) \\ &\quad -(\alpha_4 - \alpha_1\alpha_3) y(k-3) - \cdots - (\alpha_p - \alpha_1\alpha_{p-1}) y(k-p+1) \\ &\quad +\alpha_1\alpha_p y(k-p) \\ &\quad +\beta_0 u(k+1) + (\beta_1 - \alpha_1\beta_0) u(k) + (\beta_2 - \alpha_1\beta_1) u(k-1) + \ldots \\ &\quad +(\beta_p - \alpha_1\beta_{p-1}) u(k-p+1) - \alpha_1\beta_p u(k-p) \end{aligned}$$

defining the terms:

$$\begin{array}{cc} \alpha_1^{(1)} = \alpha_2 - \alpha_1\alpha_1 & \beta_0^{(1)} = \beta_1 - \alpha_1\beta_0 \\ \alpha_2^{(1)} = \alpha_3 - \alpha_1\alpha_2 & \beta_1^{(1)} = \beta_2 - \alpha_1\beta_1 \\ \vdots & \vdots \\ \alpha_{p-1}^{(1)} = \alpha_p - \alpha_1\alpha_{p-1} & \beta_{p-1}^{(1)} = \beta_p - \alpha_1\beta_{p-1} \\ \alpha_p^{(1)} = -\alpha_1\alpha_p & \beta_p^{(1)} = -\alpha_1\beta_p \end{array}$$

we can simplify this equation to:

$$\begin{aligned} y(k+1) &= -\alpha_1^{(1)} y(k-1) - \alpha_2^{(1)} y(k-2) - \cdots - \alpha_{p-1}^{(1)} y(k-p+1) - \alpha_p^{(1)} y(k-p) \\ &\quad +\beta_0 u(k+1) + \beta_0^{(1)} u(k) + \cdots + \beta_p^{(1)} u(k-p) \end{aligned}$$

We now advance the whole equation one time-step and substitute for $y(k)$:

$$\begin{aligned} y(k+2) &= -\left(\alpha_2^{(1)} - \alpha_1^{(1)}\alpha_1\right) y(k-1) - \left(\alpha_3^{(1)} - \alpha_1^{(1)}\alpha_2\right) y(k-2) - \ldots \\ &\quad -\left(\alpha_p^{(1)} - \alpha_1^{(1)}\alpha_{p-1}\right) y(k-p+1) + \alpha_1^{(1)}\alpha_p y(k-p) \\ &\quad +\beta_0 u(k+2) + \beta_0^{(1)} u(k+1) + \left(\beta_1^{(1)} - \alpha_1^{(1)}\beta_0\right) u(k) + \\ &\quad +\left(\beta_2^{(1)} - \alpha_1^{(1)}\beta_1\right) u(k-1) + \cdots + \left(\beta_p^{(1)} - \alpha_1^{(1)}\beta_{p-1}\right) u(k-p+1) - \alpha_1^{(1)}\beta_p u(k-p) \end{aligned}$$

Once again we define new terms:

$$\begin{array}{cc}
\alpha_1^{(2)} = \alpha_2^{(1)} - \alpha_1^{(1)}\alpha_1 & \beta_0^{(2)} = \beta_1^{(1)} - \alpha_1^{(1)}\beta_0 \\
\alpha_2^{(2)} = \alpha_3^{(1)} - \alpha_1^{(1)}\alpha_2 & \beta_1^{(2)} = \beta_2^{(1)} - \alpha_1^{(1)}\beta_1 \\
\vdots & \vdots \\
\alpha_{p-1}^{(2)} = \alpha_p^{(1)} - \alpha_1^{(1)}\alpha_{p-1} & \beta_{p-1}^{(2)} = \beta_p^{(1)} - \alpha_1^{(1)}\beta_{p-1} \\
\alpha_p^{(2)} = \alpha_1^{(1)}\alpha_p & \beta_p^{(2)} = -\alpha_1^{(1)}\beta_p
\end{array}$$

and the equation simplifies to:

$$\begin{aligned}
y(k+2) &= -\alpha_1^{(2)}y(k-1) - \alpha_2^{(2)}y(k-2) - \cdots - \alpha_p^{(2)}y(k-p) \\
&\quad +\beta_0 u(k+2) + \beta_0^{(1)}u(k+1) \\
&\quad +\beta_0^{(2)}u(k) + \beta_1^{(2)}u(k-1) + \cdots + \beta_p^{(2)}u(k-p)
\end{aligned}$$

This process can be generalized for any time-step j:

$$\begin{aligned}
y(k+j) &= -\alpha_1^{(j)}y(k-1) - \alpha_2^{(j)}y(k-2) - \cdots - \alpha_p^{(j)}y(k-p) \\
&\quad +\beta_0 u(k+j) + \beta_0^{(1)}u(k+j-1) + \cdots + \beta_0^{(j)}u(k) \\
&\quad +\beta_1^{(j)}u(k-1) + \beta_2^{(j)}u(k-2) + \cdots + \beta_p^{(j)}u(k-p)
\end{aligned} \tag{2.14}$$

and the parameters:

$$\begin{array}{cc}
\alpha_1^{(j)} = \alpha_2^{(j-1)} - \alpha_1^{(j-1)}\alpha_1 & \beta_1^{(j)} = \beta_2^{(j-1)} - \alpha_1^{(j-1)}\beta_1 \\
\alpha_2^{(j)} = \alpha_3^{(j-1)} - \alpha_1^{(j-1)}\alpha_2 & \beta_2^{(j)} = \beta_3^{(j-1)} - \alpha_1^{(j-1)}\beta_2 \\
\vdots & \vdots \\
\alpha_{p-1}^{(j)} = \alpha_p^{(j-1)} - \alpha_1^{(j-1)}\alpha_{p-1} & \beta_{p-1}^{(j)} = \beta_p^{(j-1)} - \alpha_1^{(j-1)}\beta_{p-1} \\
\alpha_p^{(j)} = -\alpha_1^{(j-1)}\alpha_p & \beta_p^{(j)} = -\alpha_1^{(j-1)}\beta_p \\
 & \beta_0^{(j)} = \beta_1^{(j-1)} - \alpha_1^{(j-1)}\beta_0
\end{array}$$

We now wish to set up a system of equations to predict the output for $j = 1, 2 \ldots, s-1$. Using Equation (2.14), we get:

$$y_s(k) = \mathcal{T} u_s(k) + \mathcal{B} u_p(k-p) - \mathcal{A} y_p(k-p) \tag{2.15}$$

where:

$$y_s(k) = \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+s-1) \end{bmatrix} \qquad u_s(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+s-1) \end{bmatrix}$$

$$y_p(k-p) = \begin{bmatrix} y(k-p) \\ y(k-p+1) \\ \vdots \\ y(k-1) \end{bmatrix} \qquad u_p(k-p) = \begin{bmatrix} u(k-p) \\ u(k-p+1) \\ \vdots \\ u(k-1) \end{bmatrix}$$

and

$$\mathcal{T} = \begin{bmatrix} \beta_0 & & & \\ \beta_0^{(1)} & \beta_0 & & \\ \vdots & \vdots & \ddots & \\ \beta_0^{(s-1)} & \beta_0^{(s-2)} & \dots & \beta_0 \end{bmatrix} \qquad \mathcal{B} = \begin{bmatrix} \beta_p & \beta_{p-1} & \dots & \beta_1 \\ \beta_p^{(1)} & \beta_{p-1}^{(1)} & \dots & \beta_1^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_p^{(s-1)} & \beta_{p-1}^{(s-1)} & \dots & \beta_1^{(s-1)} \end{bmatrix}$$

$$\mathcal{A} = \begin{bmatrix} \alpha_p & \alpha_{p-1} & \dots & \alpha_1 \\ \alpha_p^{(1)} & \alpha_{p-1}^{(1)} & \dots & \alpha_1^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_p^{(s-1)} & \alpha_{p-1}^{(s-1)} & \dots & \alpha_1^{(s-1)} \end{bmatrix}$$

2.2.2.2 Control Law

Now that we have Equation (2.15), the derivation of the generalized predictive control law is relatively straightforward. We start off with this cost equation:

$$J(k) = \frac{1}{2} \left\{ \left[y_s(k) - \tilde{y}_s(k) \right]^T \left[y_s(k) - \tilde{y}_s(k) \right] + u_s^T(k) \lambda u_s(k) \right\}$$

where $\tilde{y}_s(k)$is the desired output vector and λ is a scalar that is know as the control weighting or control penalty. Taking the derivative with respect to the control signal, we have:

$$\frac{\partial J}{\partial u_s} = \mathcal{T}^T \left(y_s(k) - \tilde{y}_s(k) \right) + \lambda u_s(k).$$

Setting this derivative equal to zero and using Equation (2.15) to substitute for $y_s(k)$, we arrive at:

$$0 = \left(\mathcal{T}^T \mathcal{T} + \lambda I \right) u_s(k) + \mathcal{T}^T \left(\mathcal{B} u_p(k-p) - \mathcal{A} y_p(k-p) - \tilde{y}_s(k) \right).$$

Solving for $u_s(k)$:

$$u_s(k) = -\left[\left(\mathcal{T}^T \mathcal{T} + \lambda I \right)^{-1} \mathcal{T}^T \right] \left[\mathcal{B} u_p(k-p) - \mathcal{A} y_p(k-p) - \tilde{y}_s(k) \right]$$

For plate vibration suppression, we can set the desired plant output vector $\tilde{y}_s(k)$ to zero. We can now separate the terms of this equation and use only the first value of the $u_s(k)$ vector to give us a finite difference form of the control law:

$$\begin{aligned} u(k) &= \alpha_1^c y(k-1) + \alpha_2^c y(k-2) + \cdots + \alpha_p^c y(k-p) \\ &\quad + \beta_1^c u(k-1) + \beta_2^c u(k-2) + \cdots + \beta_p^c u(k-p) \end{aligned} \tag{2.16}$$

where:

$$\left[\begin{array}{cccc} \alpha_p^c & \alpha_{p-1}^c & \cdots & \alpha_1^c \end{array} \right] = \text{the first r rows of } \left[\left(\mathcal{T}^T \mathcal{T} + \lambda I \right)^{-1} \mathcal{T}^T \right] \mathcal{A}$$

$$\left[\begin{array}{cccc} \beta_p^c & \beta_{p-1}^c & \cdots & \beta_1^c \end{array} \right] = \text{the first r rows of } -\left[\left(\mathcal{T}^T \mathcal{T} + \lambda I \right)^{-1} \mathcal{T}^T \right] \mathcal{B}$$

Inherent in this control design process was the length of the output prediction s. The prediction horizon h_p is defined as $h_p \equiv s - 1$ and is one of the four design parameters of the GPC controller. The last design parameter is the control horizon h_c, the length of time for which the the control signal can be non-zero. For the above derivation, the control horizon was equal to the prediction

horizon. To design a controller for which the h_c is less than h_p, Equation (2.16) is evaluated with this truncated $\mathcal{T}$:

$$\mathcal{T} = \begin{bmatrix} \beta_0 & & & \\ \beta_0^{(1)} & \beta_0 & & \\ \vdots & \vdots & \ddots & \\ \beta_0^{(h_c)} & \beta_0^{(h_c-1)} & \dots & \beta_0 \\ \beta_0^{(h_c+1)} & \beta_0^{(h_c)} & \dots & \beta_0^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_0^{(s-1)} & \beta_0^{(s-2)} & \dots & \beta_0^{(s-h_c-1)} \end{bmatrix}$$

2.3 Fuzzy Inference System

For this work, we used Sugeno-type fuzzy inference systems.

A Sugeno-type fuzzy inference system is an algorithm of the form:

$$\begin{array}{cc} \text{if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } z_1 = a_1 + b_1 x + c_1 y & \text{ELSE} \\ \text{if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } z_2 = a_2 + b_2 x + c_2 y & \text{ELSE} \\ \vdots & \vdots \\ \text{if } x \text{ is } A_n \text{ and } y \text{ is } B_n \text{ then } z_n = a_n + b_n x + c_n y & \end{array} \tag{2.17}$$

Here x and y are the inputs to the system and z is the output. Each *if* statement is a fuzzy logic rule. The a_i, b_i and c_i's are constant values. In the fuzzy logic rules developed in Chapter 6, we will always set the b_i's and c_i's to zero.

While the algorithm looks like a piece of code from a programming language, its evaluation is different. The variables A_i and B_i do not represent concrete numbers as they would in a computer program. Instead, the variables are meant to represent somewhat vague linguistic terms. These linguistic terms are modeled with fuzzy sets. The inputs x and y will be real numbers. In general, the system can have any number of inputs and outputs.

To illustrate and explain these concepts, we will use a simple example. We will consider designing a fuzzy logic system that adjusts a simple air conditioning window unit mounted in an apartment. We will assume that this fictitious air conditioner has a continuously adjustable

setting. We will designate the highest setting with a value of 2 and the lowest setting with a value of 1. The person using the air conditioner will adjust the setting based on the ambient room temperature. If the room temperature is hot, the user will set the unit to a high setting to bring the room temperature to a comfortably level quickly. For room temperatures that are only warm and not hot, this particular user sets the air conditioner to a setting of low, thereby minimizing the risk of making the apartment too cold. When the ambient temperature is between hot and cold, the user sets the air conditioner to a medium level. This type of selection process can be mimicked by the following Sugeno-type fuzzy logic system:

$$\begin{aligned} &\text{if } temperature \text{ is } hot \quad \text{then } AC = 2 \quad \text{ELSE} \\ &\text{if } temperature \text{ is } warm \quad \text{then } AC = 1 \end{aligned} \tag{2.18}$$

Here *temperature* and *AC* are the input and output, respectively. The air conditioning settings (*AC*) are constant values, i.e. the b's and c's of Equation (2.17) are set to zero. *Hot* and *warm* are linguistic variables that will be modelled with fuzzy sets. This will be explained shortly.

Fuzzy sets are extensions of crisp sets. For example, consider the crisp set $D = \{2, 3, 4\}$. We can say with certainty whether a number belongs to this set. The statement $x \in D$ has only two possible answers: it is either true (1) or false (0). Fuzzy sets are defined by allowing partial truth to this statement, i.e. the truth value can take on any value between and including 0 and 1. This necessitates that fuzzy sets are described by ordered pairs consisting of a value and its *degree of membership* (μ) in that set. The set of all ordered pairs is called a membership function. A μ close to zero indicates only a slight membership in that set, whereas a strong membership in a set would be indicated by a μ close to 1.

For instance, we wish to describe room temperatures that we consider *hot*. Most people would consider room temperatures of $70°$F normal, i.e. not hot. Therefore, a temperature of $70°$F would have a membership value of close to zero in the fuzzy set *hot.* A room temperature of $90°$F would be considered hot by most people and the *degree of membership* would be close to 1. This type of examination would be performed over a range of temperatures. The result will be a membership function such as the one shown in Figure 2.1. Similarly, we have defined the linguistic variable *warm* of Equation (2.18) by the membership function shown in Figure 2.2.

We will now show the analytic form of evaluating the fuzzy logic rules. The evaluation of

Sugeno-type fuzzy logic rules is much easier to understand graphically, and this will be demonstrated with the air conditioning example. The two-input system of Equation (2.17) will be evaluated with a generic input $x = x_o$ and $y = y_o$. The first step is to calculate the output z for each rule:

$$z_i = a_i + b_i x_o + c_i y_o \tag{2.19}$$

The next step is to calculate the *degree of fulfillment* (*DOF*) for each rule:

$$DOF_i = \mu_{A_i}(x_o) \wedge \mu_{B_i}(y_o) \tag{2.20}$$

where μ_{A_i} and μ_{B_i} are the membership functions of the fuzzy variables A_i and B_i. Here the minimum operator is denoted by the $\wedge$. The degree of fulfillment is also called the firing strength of the rule.

The final output of the fuzzy inference system is:

$$z = \frac{DOF_1 \times z_1 + DOF_2 \times z_2 + \cdots + DOF_n \times z_n}{DOF_1 + DOF_2 + \cdots + DOF_n} \tag{2.21}$$

We will now demonstrate the evaluation of the air-conditioning fuzzy logic system, Equation (2.18), with a temperature input of 77°F. The rules and their corresponding evaluation are shown graphically in Figure 2.3. The linguistic variables *hot* and *warm* are now replaced with their membership functions shown in Figures 2.1 and 2.2. These are the yellow lines in Figure 2.3. The output *AC* are the constant values 1 and 2 and are shown as light blue bars with a height of one and lateral position equal to their value.

The first step is to calculate outputs z_1 and z_2 according to Equation (2.19). The constants $b_1$and b_2 are zero, therefore the outputs are $z_1 = 2$ and $z_2 = 1$ regardless of the temperature input.

The second step is to calculate the *degree of fulfillment* (DOF*)* for each rule, Equation (2.20). Since we only have a one input system, the equation reduces to $DOF_i = \mu_{A_i}(temperature)$. This selection is denoted by the intersection of the red temperature line with the membership functions in Figure 2.3, resulting in the shaded yellow section. The height of this shaded region is the *DOF*. Also note that the output values z_i, the light blue bars, are cut by each respective *DOF* to yield the

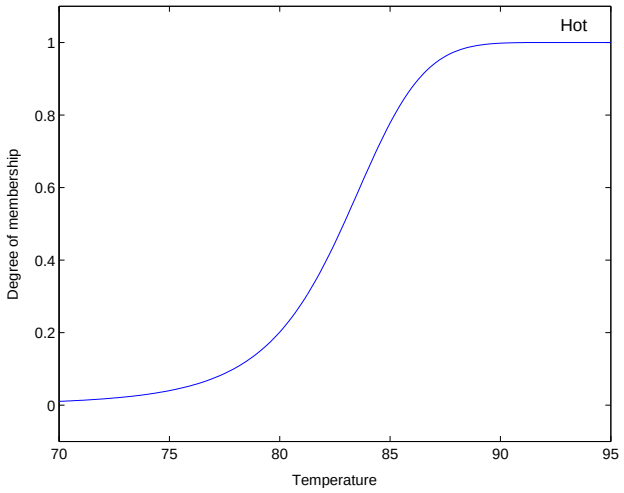


Figure 2.1: Hot Temperatures Membership Function

dark blue bars.

The final output of the fuzzy logic system is calculated by Equation (2.21) and shown by the red arrow in Figure 2.3. Note that the calculation is analogous to a center of gravity calculation with the z_i's representing the position of the object and the DOF_i's (dark blue bars) representing the mass of the object.

This rule evaluation is repeated for a range of temperatures. This results in the input-output mapping shown in Figure 2.4. We see that the two-rule fuzzy logic rule system does exactly what we intended. For low temperatures, those around $70°$F, the system will select an air conditioning setting around 1. High temperatures will receive an air conditioning setting around 2, while medium temperatures will result in a setting between 1 and 2.

The exact shape of the input-output map is heavily influenced by the shape of the membership functions. Hence, if the overall shape is undesirable, then the membership functions can be altered to achieve the required effect.

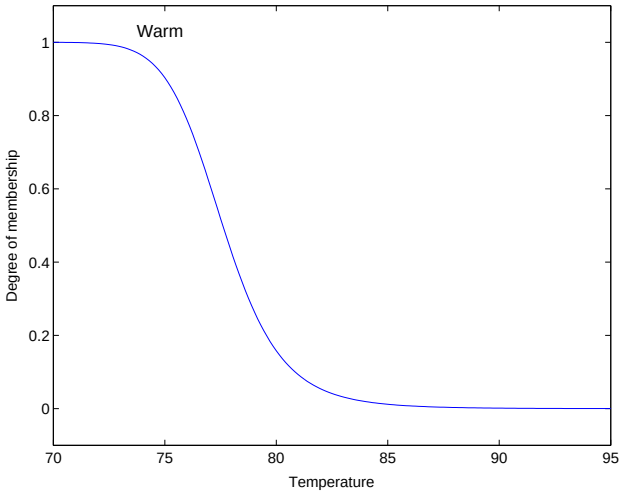


Figure 2.2: Warm Temperatures Membership Function

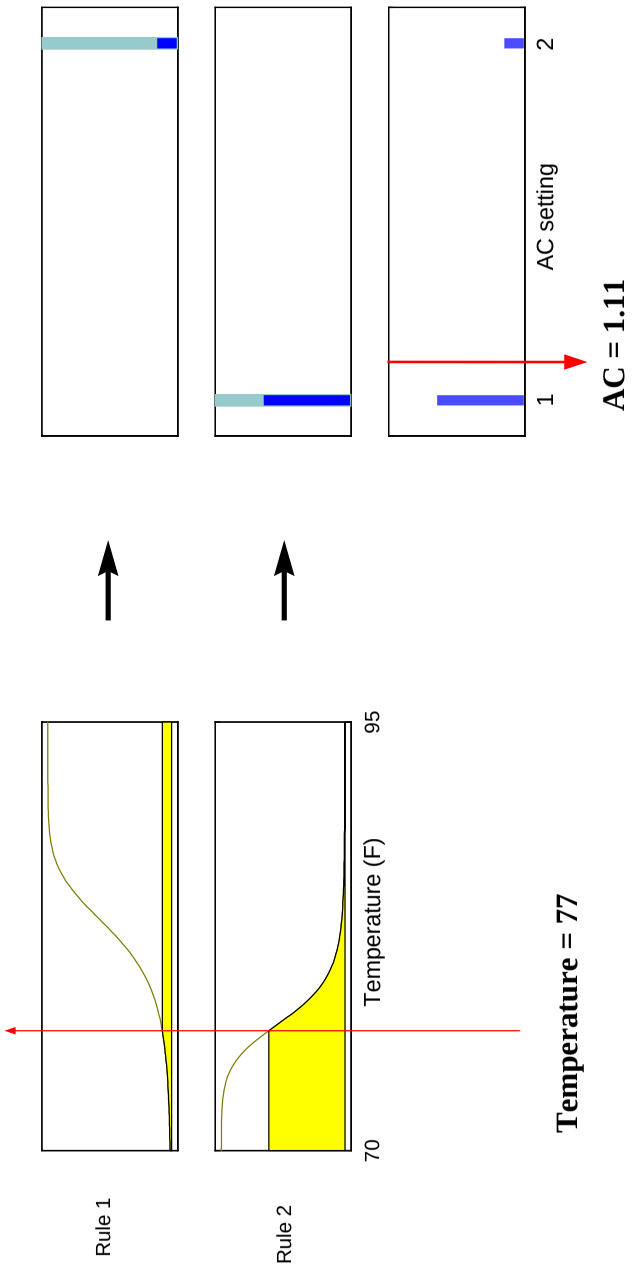


Figure 2.3: Fuzzy Air Conditioning Control - Rule Evaluation

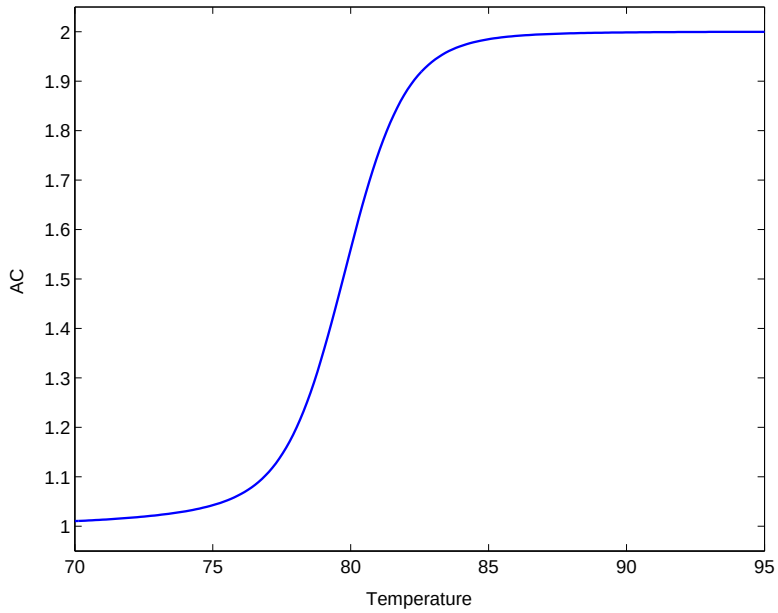


Figure 2.4: Fuzzy Air Conditioning - Input/Output Map

Chapter 3

Numerical Simulations

A numerical model, whose action was meant to emulate that of the laboratory setup, was developed. There were several reasons why such a model was necessary. First, the model allowed us to rapidly run closed-loop experiments. This was especially important during the development of the fuzzy logic rules used to adapt the controller weighting (λ) or controller order. The input-output maps of each fuzzy logic module were carefully tuned by a trial-and-error method so that they would achieve their intended effect. This required a large number of experiments, which could be done relatively easily in the numerical simulation environment. Undertaking the corresponding development in the laboratory would have taken a prohibitive amount of time. The final fuzzy logic rules are shown in Chapter 6.

A second reason for the numerical model development is the inherent ability to alter the number of plant modes. This allowed us to run experiments on various plant sizes. In Chapter 5, we make use of this feature to ensure that we have come to the correct conclusion about a rule of thumb for the selection of control and prediction horizon. In addition, it was also found that plants of different sizes had slightly different performance curve characteristics, something which we could not have demonstrated by just performing the laboratory experiments.

A third reason for the development of the numerical model is that every segment of the experiment can be completely known. In the numerical simulations, the plant is a pre-determined quantity and was made completely time invariant. In contrast, the plant is never completely known in the laboratory environment and can only be estimated by the system identification. In addition, the time invariance of the plate in the laboratory cannot be guaranteed. The fact that everything is

known in the numerical simulations allows us to compare the anticipated closed-loop poles with those of the actual closed-loop poles. This will be further discussed in Section 5.3, where we will show how generalized predictive control achieves control and disturbance rejection.

To model the plate, we wanted to pick a method that would allow us to model the actual physical experiment. The technique described by Hagood et al. [68] allows us to do this. This method allows us to model the plate and the mass and stiffness effect of adding one or more piezoelectric patches, as well incorporate the effect of any external electronic circuits connected to these patches. An overview of this approach is presented in Section 3.1.

The calculation necessary for this technique results in a pair of second-order continuous-time differential equations. These must be combined and converted to a single first-order differential equation, a so-called state-space system, in order to be used in Matlab. Hagood et al. show that there are two main methods for combining these two equations. In Section 3.2, we will present these two methods and justify our use of the simpler method. Also, the continuous-time state-space system was converted to a discrete-time state-space system. The pitfalls of this conversion, and the necessity of a continuous-time anti-aliasing filter, are shown in Section 3.2.5.

The resulting model is validated against a finite element model in Section 3.3. The Hagood model gives us very accurate results for the plate with ideal clamped-boundary conditions. We also show the results of tap test experiments performed on the laboratory plate. An analysis of the natural frequencies shows that the boundary conditions of the laboratory plate are not ideally clamped. For this reason, the model cannot be used to accurately predict the exact response of the laboratory plate.

In Section 3.4, we will present the numerical simulation procedure for the fuzzy logic adaptations. This procedure was designed to replicate the process that must be taken in the laboratory environment, which is described in Section 4.4. The ramifications of this fact are discussed in this section.

3.1 Hagood Model Overview

The procedure for modeling a piezoelectric system was described by Hagood et al. [68]. The only input to the calculations are the physical properties of the structural and piezoelectric materials.

The calculations also require a set of functions that describe the shape of the deflected plate. These shape functions will be discussed in more detail later. The procedure yields two equations that completely describe the behavior of the system:

$$(M_s + M_p)\,\ddot{r} + C\dot{r} + (K_s + K_p)\,r - \theta v = B_f f \tag{3.1}$$

$$\theta^T r + C_p v = B_q q \tag{3.2}$$

where

r is the vector of generalized mechanical coordinates

v is the vector of generalized electrical coordinates; these are voltages

q is the charge applied at the electrodes

C is an arbitrary damping matrix

M_s is the mass of the plate

M_p is the mass of the piezoelectric patch

K_s is the stiffness of the plate

K_p is the stiffness of the piezoelectric patch

θ is the electro-mechanical coupling matrix

B_f is the generalized coordinate conversion matrix for forces

f is the vector of forces

C_p is the piezoelectric capacitance matrix

B_q is the generalized coordinate conversion matrix for charges at electrodes

Equation (3.1) is known as the actuator equation. It describes how the physical motion of the structure is affected by various inputs. The voltage present on the piezoelectric patch affects the

motion of the system through the θ matrix. In an analogous manner, the point forces f change the motion of the system through the B_f matrix. The actuator equation is commonly referred to as the *equation of motion* in vibrations literature.

Equation (3.2) describes the opposite effect of the actuator equation and is known as the sensor equation. It describes how the motion of the structure generates a charge on the piezoelectric patches. To make these equations useful, they are converted to a state-space system.

3.2 State-space System

3.2.1 Voltage-driven Electrodes

There are several different ways of generating a state-space model. The first method is to assume that the physical system is driven by a voltage applied to the piezoelectric patch. The voltage is commanded so that the ensuing voltage generated by the motion of the plate does not alter the actual voltage present on the patch. This method uses only the actuator equation to derive the state space system. The resulting state-space system is:

$$\begin{aligned} \dot{x}_r &= A^v x_r + B^v u \\ y &= C^v x_r + D^v u \end{aligned} \tag{3.3}$$

where

$$x_r = \begin{bmatrix} r \\ \dot{r} \end{bmatrix}, \qquad u = \begin{bmatrix} f \\ v \end{bmatrix} = \begin{bmatrix} f \\ v_p \end{bmatrix}$$

$$A^v = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \quad B^v = \begin{bmatrix} 0 & 0 \\ M^{-1}B_f & M^{-1}\theta \end{bmatrix}$$

$$M = M_s + M_p \qquad K = K_s + K_p$$

Setting v, the generalized voltages, equal to the voltage on the piezoelectric electrodes v_p is equivalent to setting the B_q matrix to the identity matrix. The state-space output matrices C^v and D^v depend on what we wish to observe and will be described in Section 3.2.4.

The second method is to assume that the patch is connected to an electronic circuit and is

described in Section 3.2.3. The state-space system of the electronics is combined with the actuator and sensor equations in such a way that the resulting state-space system describes not only the motion of the plate, but also the voltages and current inside the circuit. Although we normally think of the electronics as driving the piezoelectric device, the reverse effect also happens. This second method captures both effects simultaneously. The reverse effect occurs as the motion of the plate generates a charge on the piezoelectric patch. This charge generates a current in the electronic circuit. This current will be dissipated by any resistive elements in that circuit. A circuit can be designed to take advantage of this effect to dampen plate modes. This is the concept behind shunt damping.

3.2.2 Charge-driven Electrodes

Specifying the system as charge-driven instead of voltage-driven is an intermediate step to describing a piezoelectric system that is driven by an electronic circuit. The sensor equation is solved for the voltages and is substituted into the actuator equation. The resulting system has force and charge as inputs:

$$\begin{aligned} \dot{x_r} &= A^q x_r + B^q u \\ y &= C^q x_r + D^q u \end{aligned} \tag{3.4}$$

where

$$x_r = \begin{bmatrix} r \\ \dot{r} \end{bmatrix}, \qquad u = \begin{bmatrix} f \\ q \end{bmatrix}$$

$$A^q = \begin{bmatrix} 0 & I \\ -M^{-1}K^{oc} & -M^{-1}C \end{bmatrix}, \; B^q = \begin{bmatrix} 0 & 0 \\ M^{-1}B_f & M^{-1}\theta^q \end{bmatrix}$$

$$K^{oc} = K + \theta C_p^{-1}\theta^T \qquad \theta^q = \theta C_p^{-1}$$

3.2.3 Piezoelectric Systems with Electronics

As was stated earlier, the formulation in this section is used to describe a system in which a piezoelectric structure is connected to an electronic circuit. It is the charge buildup on the piezoelectric

patch that ties these two systems together. As charge is added or removed from the patch by the motion of the plate, the current in the electric circuit is altered. This then also alters the charge on the patch, which in turn affects the vibration of the structure. This formulation captures all of these effects.

We start off by looking at the electric circuit. The circuit attached to the piezoelectric structure is described by its own state-space equations:

$$\dot{x}_{el} = A^{el}x_{el} + \begin{bmatrix} B^{el}_{I_o} & B^{el}_{V_o} & B^{el}_{v_p} \end{bmatrix} \begin{bmatrix} I_o \\ V_o \\ v_p \end{bmatrix}$$

$$I_p = C^{el}x_{el} + \begin{bmatrix} D^{el}_{I_o} & D^{el}_{V_o} & D^{el}_{v_p} \end{bmatrix} \begin{bmatrix} I_o \\ V_o \\ v_p \end{bmatrix}.$$

Here, I_o and V_o are the current and voltage sources, respectively, inside the electric circuit. The voltage on the piezoelectric patch (v_p) is considered as an input to the electric system. I_p must be defined to be the current going into the piezoelectric. The states x_{el} are the currents in the circuit. We can now combine these equations with those of the charge-driven system to get the complete system equation:

$$\begin{aligned} \dot{x_{sh}} &= A^{sh}x_{sh} + B^{sh}u_{sh} \\ y &= C^{sh}x_{sh} + D^{sh}u_{sh} \end{aligned} \tag{3.5}$$

where

$$x_{sh} = \begin{bmatrix} x_r \\ q \\ x_{el} \end{bmatrix}, \qquad u_{sh} = \begin{bmatrix} f \\ I_o \\ V_o \end{bmatrix}$$

$$A^{sh} = \begin{bmatrix} A^q & [B^q]^q & 0 \\ D'_{v_p} & D''_{v_p} & C^{el} \\ B'_{v_p} & B''_{v_p} & A^{el} \end{bmatrix}, \quad B^{sh} = \begin{bmatrix} [B^q]^f & 0 & 0 \\ 0 & D^{el}_{I_o} & D^{el}_{V_o} \\ 0 & B^{el}_{I_o} & B^{el}_{V_o} \end{bmatrix}$$

and

$$B'_{v_p} = -B^{el}_{V_p} C_p^{-1} \theta^T \begin{bmatrix} I & 0 \end{bmatrix} \qquad B''_{v_p} = B^{el}_{v_p} C_p^{-1} B_q$$
$$D'_{v_p} = -D^{el}_{V_p} C_p^{-1} \theta^T \begin{bmatrix} I & 0 \end{bmatrix} \qquad D''_{v_p} = D^{el}_{v_p} C_p^{-1} B_q.$$

The last four terms are slightly different than those presented by Hagood. We believe that he has made an error. The notation $[]^q$ means to take those columns associated with q. The state-space output matrices C^{sh} and D^{sh} depend on what we wish to observe and will be described in Section 3.2.4.

In the experimental setup, the plate is driven by a PCB Series 790 power amplifier. The output impedance of this device is 10 ohms, and the device is designed so that this impedance is purely real. The power amplifier can therefore be modeled as an ideal voltage source and a resistor as shown in Figure 3.1.

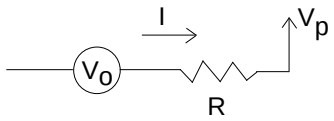


Figure 3.1: Amplifier Model

Since this system does not have any capacitors or inductors, and hence no dynamics, the electrical state-space equations reduce to:

$$\begin{aligned} A^{el} &= \text{empty} \\ B^{el} &= \text{empty} \\ C^{el} &= \text{empty} \\ D^{el}_{I_o} &= \text{empty} \\ I_p &= \begin{bmatrix} D^{el}_{V_o} & D^{el}_{v_p} \end{bmatrix} \begin{bmatrix} V_o \\ v_p \end{bmatrix} = \begin{bmatrix} \frac{-1}{R} & \frac{1}{R} \end{bmatrix} \begin{bmatrix} V_o \\ v_p \end{bmatrix} \end{aligned}$$

The complete system equation (from Equation (3.5)) is now:

$$\begin{aligned} \dot{x_{sh}} &= A^{sh}x_{sh} + B^{sh}u_{sh} \\ y &= C^{sh}x_{sh} + D^{sh}u_{sh} \end{aligned} \tag{3.6}$$

where

$$x_{sh} = \begin{bmatrix} x_r \\ q \end{bmatrix} \qquad u_{sh} = \begin{bmatrix} f \\ V_o \end{bmatrix}$$

$$A^{sh} = \begin{bmatrix} A^q & [B^q]^q \\ D'_{v_p} & D''_{v_p} \end{bmatrix} \qquad B^{sh} = \begin{bmatrix} [B^q]^f & 0 \\ 0 & D^{el}_{V_o} \end{bmatrix}.$$

At this point, we must decide which method to use. A comparison of Equation (3.6) and Equation (3.3) is now in order. Figure 3.2 shows the transfer functions of these two equations from the voltage input of each system to the displacement measured at location 3[1]. The two plots are nearly identical. This indicates that the combined effect of the output resistance of the amplifier and the capacitance of the piezoelectric patch is very small and can be ignored. We therefore use Equation (3.3) to convert the system to a state-space form.

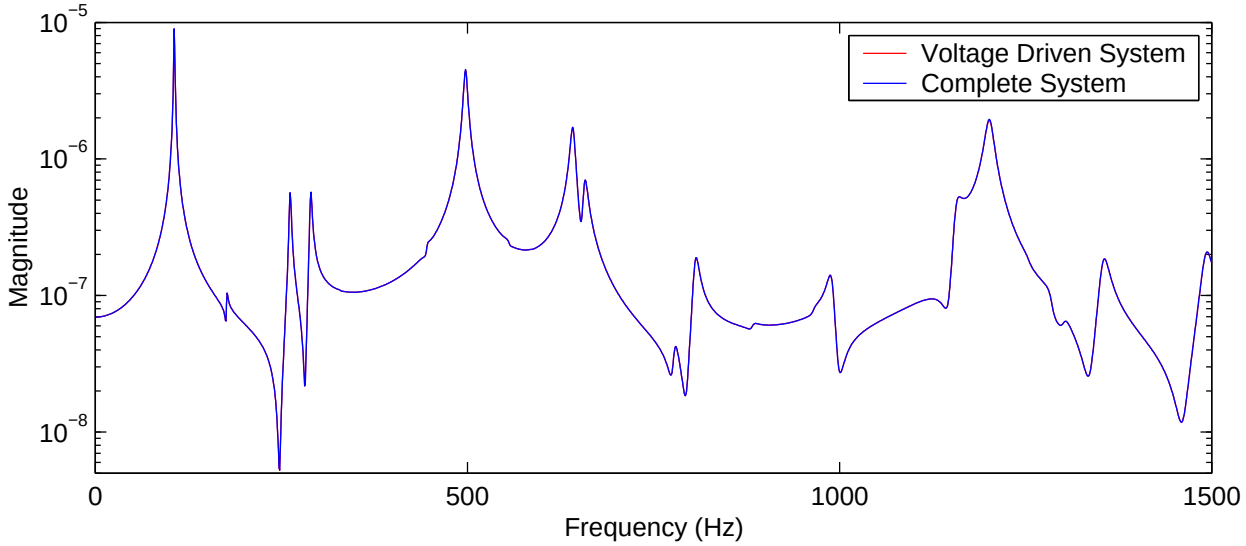


Figure 3.2: Voltage to Displacement Transfer Function Comparison (Point 3)

[1]Figure 3.6 shows the position of the measurement locations on the plate.

3.2.4 Displacement and Acceleration Output

For a state-space system:

$$y = Cx + Du$$

the output matrices C and D are arbitrary. However, these matrices can be chosen so that the output y has physical significance. This is particularly true for the Hagood formulation. We can calculate C and D so the output is either the displacement or the acceleration of a point on the plate.

The Hagood method is essentially an assumed modes method. This means that the displacement of the plate at a point is:

$$w(x,y,t) = \begin{bmatrix} \Psi_1(x,y) & \Psi_2(x,y) & \dots & \Psi_n(x,y) \end{bmatrix} \begin{bmatrix} r_1(t) \\ r_2(t) \\ \vdots \\ r_n(t) \end{bmatrix} = \Psi_r \cdot \begin{bmatrix} r_1(t) \\ r_2(t) \\ \vdots \\ r_n(t) \end{bmatrix} \tag{3.7}$$

where $\Psi_i(x,y)$ are the admissible functions evaluated at the point of interest. The output equation for displacement is therefore:

$$y = \begin{bmatrix} \Psi_r & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} + [0]\, u$$

To get acceleration output, we take two derivatives of Equation (3.7) and substitute the actuator equation for $\ddot{r}$. This gives us an output equation for acceleration:

$$y = \begin{bmatrix} -\Psi_r M^{-1} K & -\Psi_r M^{-1} C \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} \Psi_r M^{-1} B_f & \Psi_r M^{-1} \ \theta \end{bmatrix} \begin{bmatrix} f \\ v_p \end{bmatrix} \tag{3.8}$$

Both displacement and acceleration output systems are used throughout this paper.

3.2.5 Continuous-time to Discrete-time Conversion

All of the state-space models described up to this point have been continuous-time systems. These systems need to be converted to discrete-time models inside Matlab in order to be useful. This is typically done with the *c2d* command in the controls toolbox. This conversion process had unforeseen consequences for the system with acceleration outputs. The response of the system at frequencies above the Nyquist frequencies aliased back into the frequency span of interest, thereby corrupting the system response.

This aliasing problem is illustrated in Figure 3.3 for a simple five-mode system. The transfer function of the force input to the acceleration measured at location five for the continuous-time state-space system (Equation (3.3)) is shown as the red curve. The state-space system is then discretized with the *c2d* command, using zero-order hold as the discretization method.[2] The transfer function of the discrete system is shown as the blue curve in Figure 3.3. These two curves should be nearly identical, but they are not. The *c2d* conversion process has resulted in an aliasing-type effect in which the zeros of the system are not correctly transformed.

This problem was corrected by adding a continuous-time elliptical filter to the output of the plate system before beginning the *c2d* conversion process. This elliptical filter was chosen because the PCB signal conditioning box used in the laboratory also uses a seventh-order elliptical anti-aliasing filter. The transfer function of this larger system after the *c2d* discretization process is shown in Figure 3.3 as the green line. We see that the discrete system with the filter has a transfer function that is nearly identical to that of the original continuous-time system. The low-pass filter starts to roll off at 1400 Hz, resulting in some separation between the two curves.

The elliptical filter was designed with the Matlab *ellip* command. The designed filter was of seventh order, with 0.1 dB of ripple in the stopband and 100 dB of attenuation in the stopband, and a cutoff frequency of 1400 Hz . Although the attenuation is very high in the stopband, the stopband does not occur until 3000Hz. At the Nyquist frequency of 1500 Hz, the attenuation is only 4.2 dB. The corresponding transfer function of the filter is shown in Figure 3.4. Figure 3.5 shows the pole-zero map of the filter on the s-plane.

The inclusion of the filter is necessary not only for a proper representation of the transfer function, but also for closed-loop stability for large changes in λ. This will be further discussed in

[2] Zero-order hold is also used by the DSP hardware in the laboratory.

Section 5.3. For this reason, the elliptical filter is included in most numerical simulations.

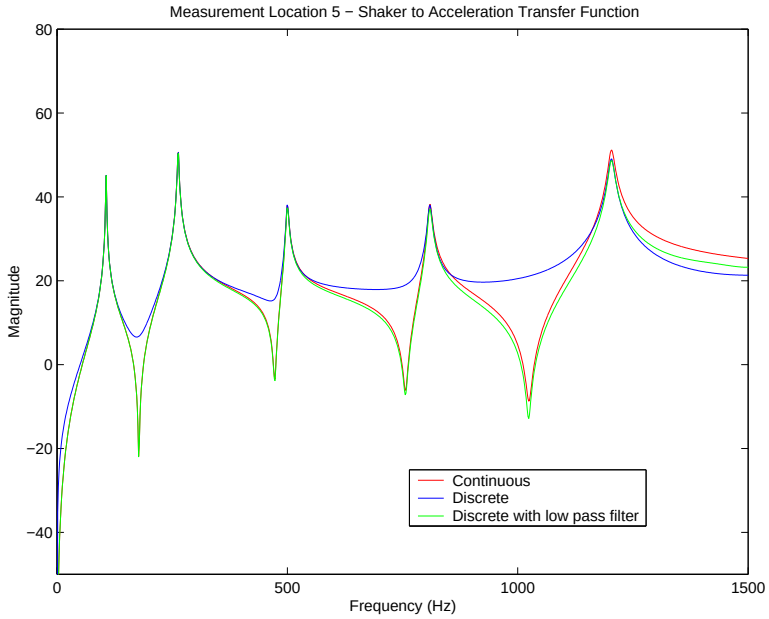


Figure 3.3: Continuous to Discrete Conversion Problems

3.3 Validity of the Hagood Model

This section describes various tests that were performed in order to validate the clamped-plate Hagood model. The physical constants used for the Hagood calculation were identical to those of the actual plate in the experimental setup. The first step was to check the accuracy of the calculation for a bare plate. Tap tests were then performed on the actual plate to characterize its response.

The natural frequencies of a bare plate were computed using the Hagood method. These frequencies were compared to those computed using a finite element analysis (Table 3.1). The table shows that the Hagood approach gives us accurate results. The finite element analysis was done with a code developed by Travis Turner of the Structural Acoustics Branch of NASA Langley. His code uses a C1 continuous-plate-element with sixteen out-of-plane degrees-of-freedom (DOF) and eight in-plane degrees-of-freedom. The formulation is bilinear for the in-plane DOF, and cubic for the out-of-plane DOF. This plate element is described on page 423 of Reference [69], while Dr. Turner's work is further documented in [70].

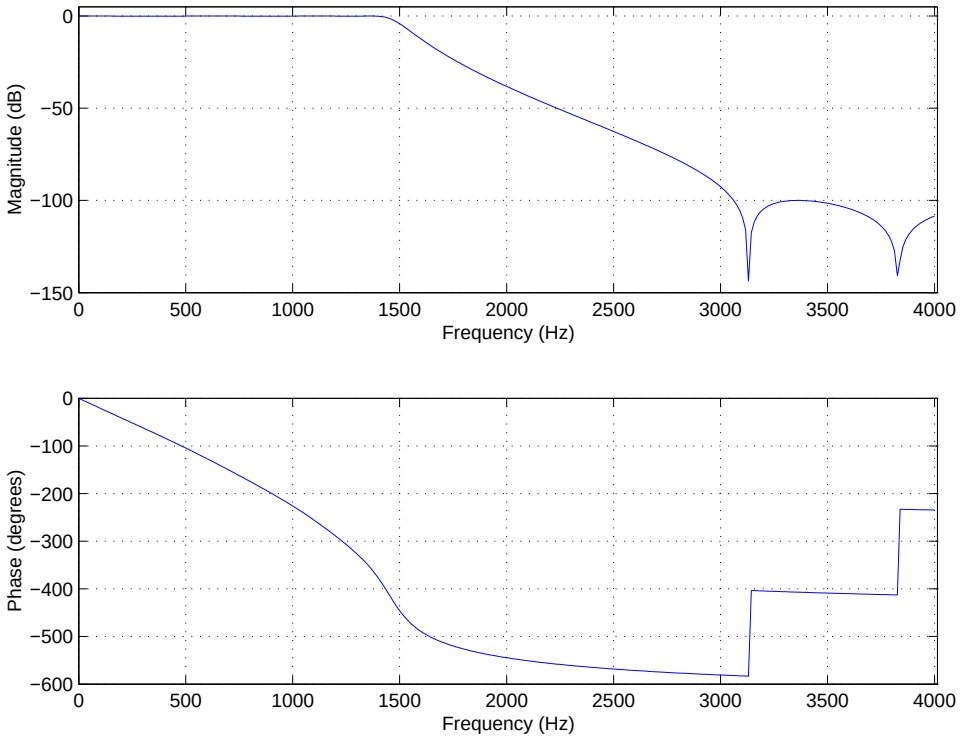


Figure 3.4: Anti-aliasing Filter - Frequency Response

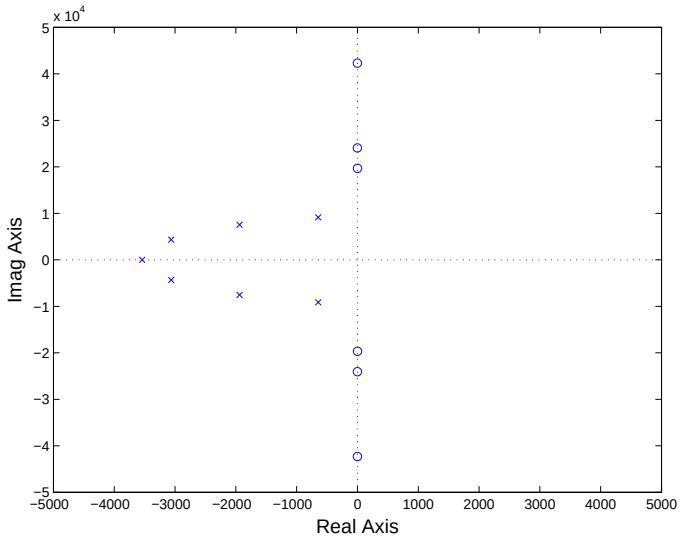
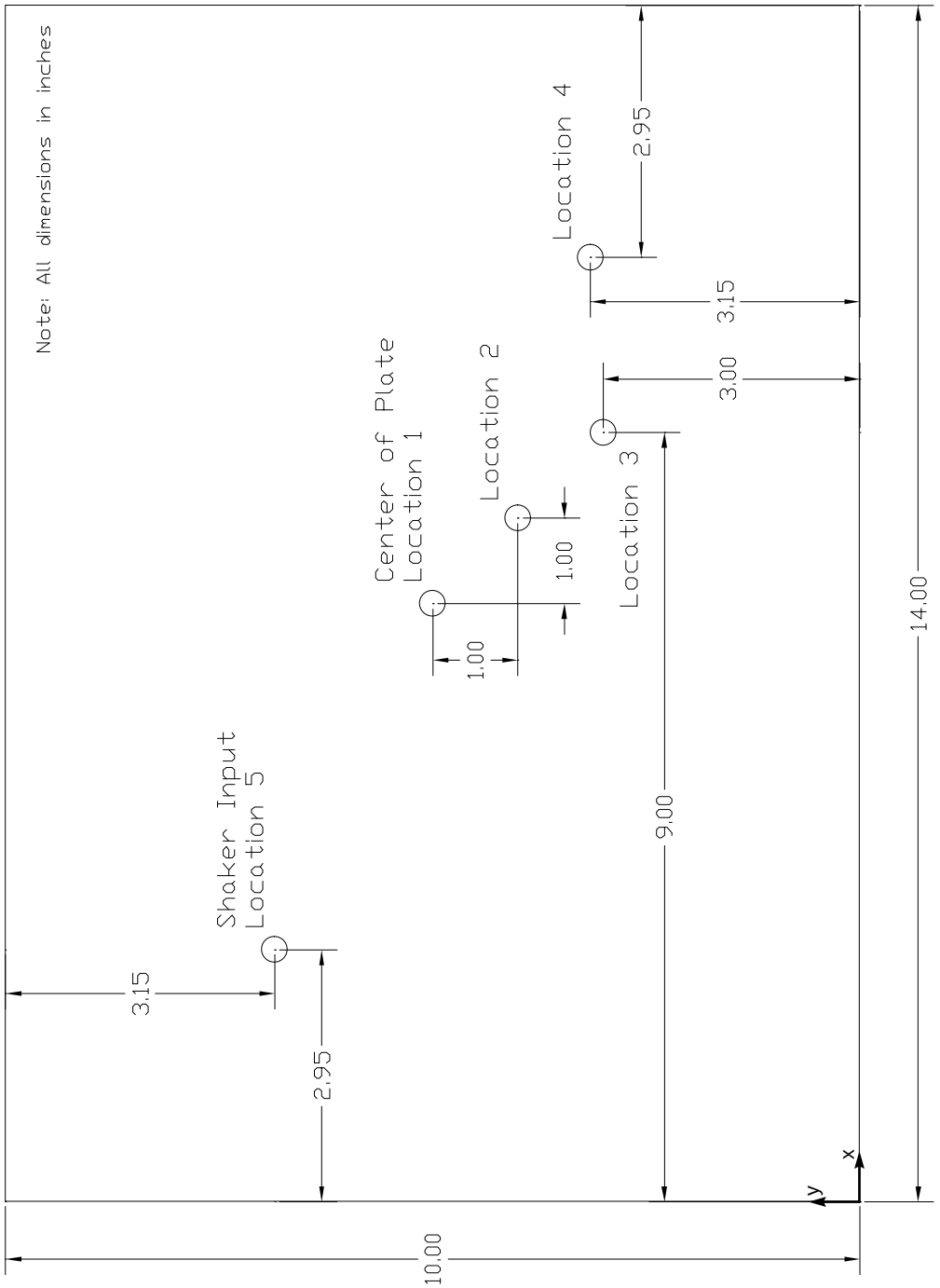


Figure 3.5: Anti-aliasing Filter - Poles and Zeros in S-Plane

Table 3.1: Clamped Plate Natural Frequencies for Hagood and Finite Element Models

Mode	Finite Element Analysis (Hz) (1.0 in. elements)	Finite Element Analysis (Hz) (0.5 in. elements)	Hagood Approach (Hz)	Percent Error (Hagood vs. 0.5 in.)
1	101.2	101.2	101.1	-0.11
2	164.0	164.0	164.1	0.04
3	242.4	242.4	242.0	-0.16
4	268.5	268.5	268.7	0.09
5	300.5	300.5	300.4	-0.02
6	399.2	399.0	399.2	0.06
7	412.1	411.8	412.4	0.14
8	457.0	456.6	455.8	-0.17
9	513.3	512.9	512.6	-0.06
10	537.9	537.6	538.3	0.13
11	593.6	592.8	593.5	0.12
12	608.7	608.2	608.3	0.01
13	715.9	715.0	716.2	0.16
14	743.5	742.6	741.3	-0.18
15	744.4	742.8	743.8	0.13
16	799.9	798.2	797.6	-0.07
17	813.0	810.9	812.0	0.13
18	893.5	891.8	891.7	-0.01
19	917.7	916.5	918.0	0.16
20	932.7	930.5	932.4	0.20
21	1026	1024	1026	0.16
22	1071	1066	1067	0.10
23	1106	1100	1098	-0.17
24	1131	1129	1132	0.24
25	1161	1156	1154	-0.15



Note - Measurement Location 6 is 2 mm above and to the left of Location 1 ($x_6 = x_1 + 2\,mm$, $y_6 = y_1 + 2\,mm$)

Figure 3.6: Measurement Locations

In order to characterize the actual plate, a series of tap tests were performed on the bare plate and on the plate with the piezoelectric patch mounted. The plate was hit at the shaker location (see Figure 3.6) and the response was measured at points 1, 2, 3, and 4 with an accelerometer. This was done ten times for each location, and the results were averaged. Since the clamping conditions are not ideal, we expect to see the natural frequencies fall between those of a simple supported plate and a clamped plate. Table 3.2 shows that this is the case.

Table 3.2: Natural Frequency Comparison to Ideal Clamping Conditions

Mode	Simply Supported Plate Modes (Hz)	Clamped Plate Modes (Hz)	Actual Modes (Hz)
1	59.2	109.4	88.0
2	119.4	177.3	145.0
3	176.7	261.9	237.5
4	219.7	290.2	250.0
5	236.9	324.9	294.5

A comparison of tap test and clamped-plate Hagood model transfer functions is shown in Figures 3.7 and 3.8 for two different measurement locations. These figures show that the model captures many of the dynamics of the actual plate. Again we see that the natural frequencies are over-predicted. The differences between the actual plate and the model are significant, and the plate model therefore cannot be used to predict the exact response of the actual physical plate.

We now wish to examine the effect of adding the piezoelectric patch to the structure. Figure 3.9 shows a transfer function of the actual plate before and after the patch was glued to the structure. Figure 3.10 shows this exact transfer function comparison for the model. We can see that the model correctly predicts many of the shifts in the natural frequencies for modes higher than 200 Hz. Below 200 Hz, the model does not appear to correctly capture the effect of adding the piezoelectric patch. We believe that this can be attributed to experimental uncertainty in the tap test data. The Hagood method therefore seems to correctly predict the effect of added mass and stiffness of the piezoelectric patch.

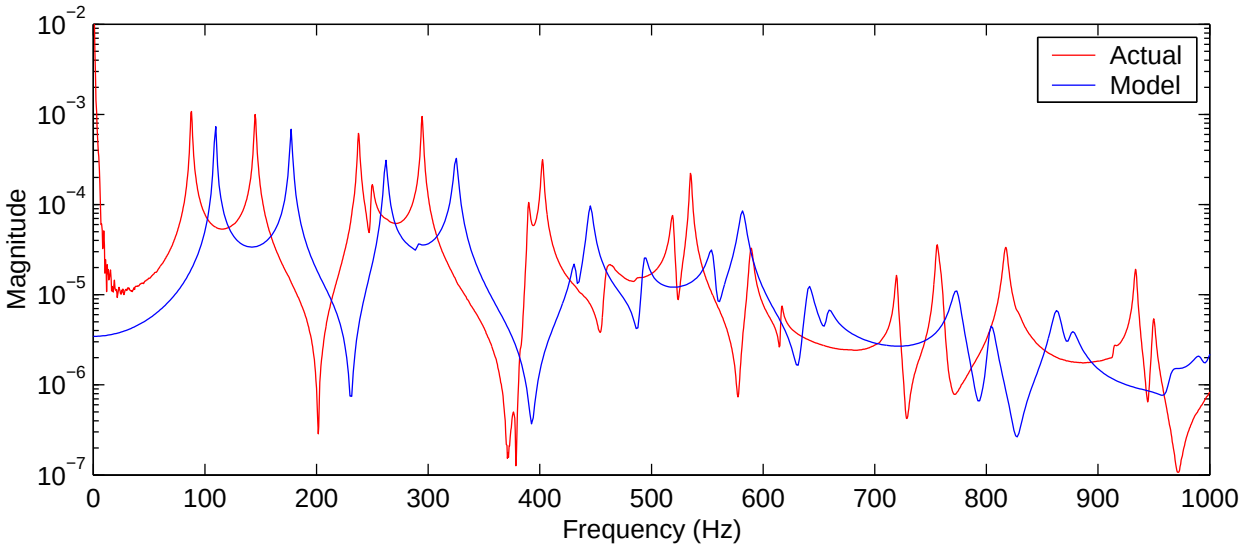


Figure 3.7: Point 3 Bare Plate Shaker to Displacement Transfer Function - Model vs. Actual

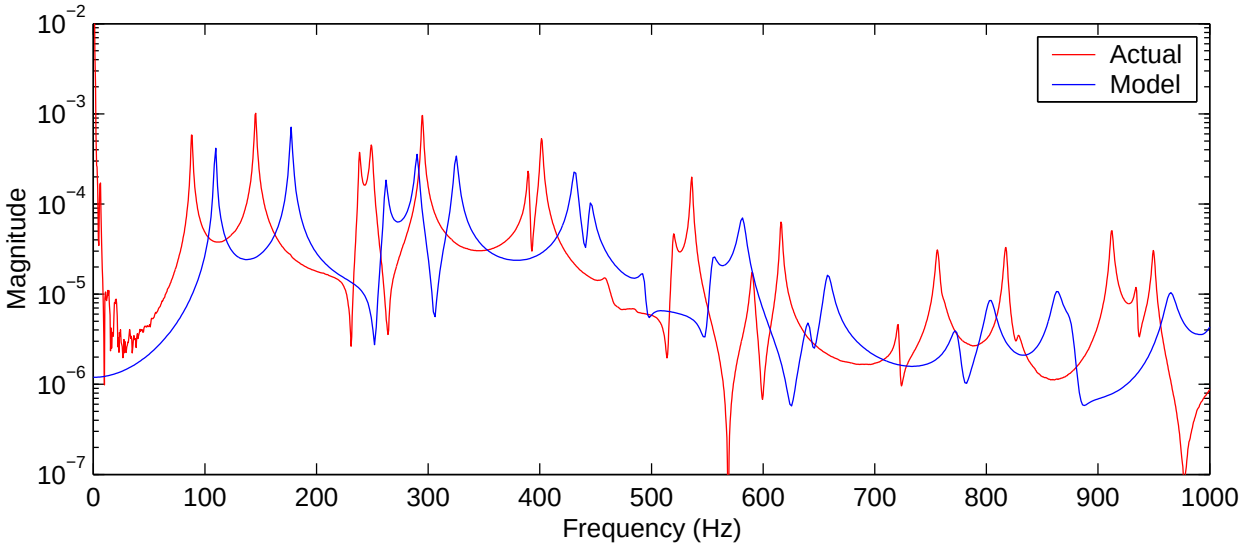


Figure 3.8: Point 4 Bare Plate Shaker to Displacement Transfer Function - Model vs. Actual

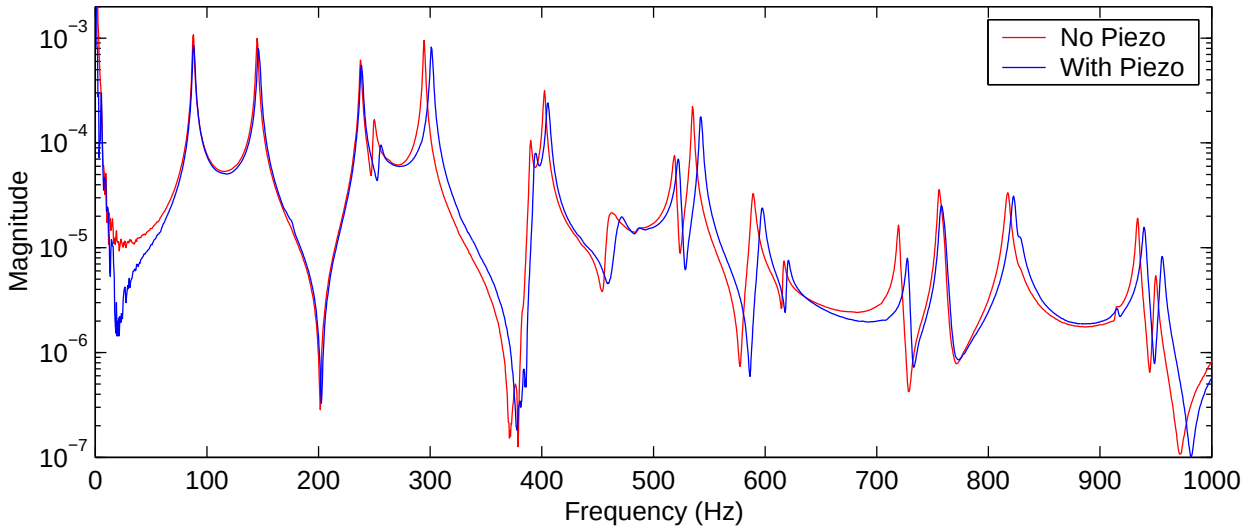


Figure 3.9: Point 3 Actual Shaker to Displacement Transfer Function - Plate with and without Piezoelectric Patch

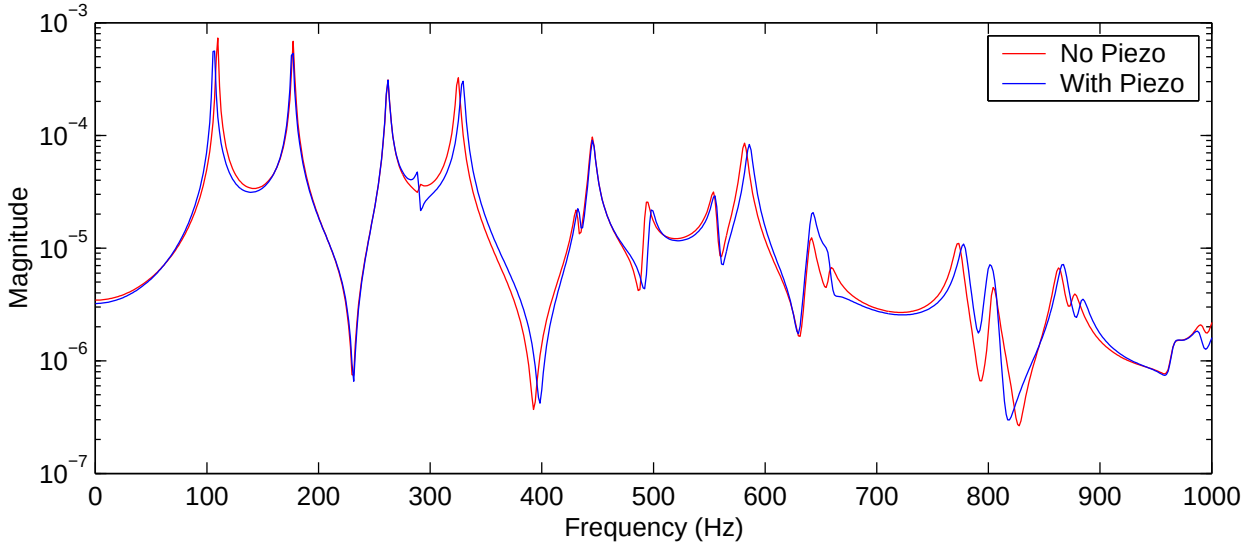


Figure 3.10: Point 3 Modeled Shaker to Displacement Transfer Function - Plate with and without Piezoelectric Patch

3.4 Numerical Simulation Procedure

We have shown in the previous section of this chapter the development of a discrete-time plant whose action emulates the behavior of the clamped-plate laboratory setup. Running a fuzzy logic adaptation of λ or order in the laboratory necessitates a certain logical sequence of steps, as included in Section 4.4. Our goal in designing the numerical simulation procedure was to replicate these steps as closely as possible.

Our intent was to capture the inexact nature of the laboratory environment in the numerical simulations. For example, the true plant is an unknown quantity in the laboratory. The plant is estimated by the system identification procedure and this information is then used in the design of the control law. Errors present in the system identification may therefore be magnified when the controller closes the loop on the system. This often results in unexpected closed-loop instabilities.

In the numerical simulation environment, the plant is a known quantity and could therefore be used in the controller design. We have chosen not to do this, but instead to simulate the process that occurs in this laboratory. Also, the loop is closed on the actual plant and not on the identified (estimated) plant, just as in the laboratory. The numerical simulation therefore allows us to examine the two sets of closed-loop poles: those that were intended, and those that actually occurred. This concept is explored in Section 5.3. The unexpected closed-loop instabilities also occur in the numerical simulations, thereby indicating some success in duplicating the laboratory experimental uncertainty in the numerical environment.

The task of replicating the laboratory procedure steps is made easier by the nature of the individual steps. In fact, the steps can be classified into two main categories: data collection, and calculations based on the collected data.

For example, the process of identifying the plant in the numerical simulations involves two distinct steps, the first of which is a data collection step, and the second of which is a calculation step. The first step is to excite the system with a known random noise through the control actuator (Step 2 on the list below). Note that during this time, an unknown disturbance is also acting on the structure.[3] The response of the plate to this dual input is calculated by a simple state-space evolution, the Matlab *lsim* command. Hence, this first step results in two distinct time vectors: the

[3]The random noise sent to the actuator is uncorrelated to the disturbance signal.

control input and the corresponding response of the plate.[4]

The second step in the system identification is to calculate the ARX equation (Step 3). The inputs to this calculation are the time vectors collected in the previous step. The source of the data, whether experimental or numerical, is irrelevant to the calculation. This calculation is therefore completely independent of the environment in which the experiment is run.

As this example shows, the two main categories of steps also define two levels of abstraction in the procedure. The execution of a data collection step such as that described above is always highly dependent on the experimental environment. Hence, these steps must be tailored to the particular experimental environment. In the numerical simulation environment, we use the Matlab *lsim* command to simulate the response of an open- and closed-loop plant. An entirely different mechanism is used to collect the necessary data in the laboratory, as documented in Section 4.4.

The calculations steps, such as the computation of the ARX equation described above, are more abstract than the data collection steps. Their only inputs are numbers or vectors of numbers. Therefore, the completion of a calculation step is independent of the particular experimental environment.

The task of making the numerical and laboratory procedures nearly identical is simplified when the abstraction levels of the steps are recognized. The abstract nature of the calculation steps implies that the exact same step can be used in both procedures, and this is exactly what was done. Note that this includes all of the major tasks that occur in the adaptation process: system identification, controller design, and fuzzy logic evaluation.

The data collection steps are also duplicated in both procedures. The fact that data must be collected at a specific time is replicated. The type of data collected is also the same in both procedures (such as the vectors of time data described above). Since the data collection is specific to the experimental environment, the particular details of the execution are different for each environment.

The similarity of the numerical and laboratory experimental procedures allowed us to write a single Matlab program that ran the adaptation experiments in both environments. A flag set at the start of the execution of the program determines which implementation is used throughout the

[4]Obviously in the numerical simulation, there is no such thing as an unknown quantity. For this step, the disturbance excitation force and the actuator signal are predetermined, and the response of the plate is calculated by the state-space evolution. However, we are interested in the rejection of unknown disturbances that occur in the laboratory. For the numerical simulations, the disturbance signal is not used in the subsequent ARX equation, thereby making it an unknown quantity just as it is in the laboratory.

experiments. This program is set up to follow the overall abstract procedure in a generic sense.

Specifically, this program is one large Matlab function that is called from within a Matlab script that sets specific operating parameters. Major steps such as system identification, closing the loop, and calculating the control law are subfunctions of this larger overall function. Subfunctions that are calculation steps are passed the relevant information needed to perform the calculation step.

Subfunctions that are data collection steps must first determine which of the two implementations must be used. The execution then branches off to the implementation specific instructions to collect or simulate the necessary data. In the case of the numerical simulations, this is done by calculating the time response of the various plants, closed and open, with the Matlab *lsim* command. In the laboratory, we collect the response from the sensors and actuators. This was done with digital signal processing (DSP) hardware fully described in Chapter 4. We have the ability to access and control this hardware from within Matlab using a library of specialized Matlab functions. Hence the calls to *lsim* in the numerical simulation are replaced with calls to one of these specialized functions in order to collect the required data and import it into the Matlab workspace.

Since both the numerical and laboratory experiments are run by the same Matlab program, the general flow of the experiments is the same in both cases. The only real difference between the two implementations is the method by which data is collected.

The list of steps that were used in the adaptation of λ or order in the numerical simulations are shown below:

1. Determine plant

   The user first selects the desired number of vibration modes. The continuous-time state-space system of Section 3.2.1 (Equation (3.3)) is now calculated. For most numerical simulations, we used an acceleration output; therefore, the matrices C^v and D^v are calculated according to Section 3.2.4. Unless noted otherwise, the anti-aliasing filter of Section 3.2.5 is added to the output of the state-space system. It is this larger system that is converted to a discrete-time system via the Matlab command *c2d*. This discrete system is the plant used in the remaining steps.

2. Perform system identification

   Two separate random noise series are generated by the Matlab *rand* command. The first

random series, *u*, will represent the voltage input into the piezoelectric patch. The other random series, u_{noise}, represents the disturbance input to the system. For each time sequence, the state of the random noise generator is set so that the resulting random sequences are uncorrelated. Both sequences were 50,000 samples long.

The response of the plant due to both simultaneous inputs *u* and u_{noise} is now calculated. This is done with the *lsim* command in the Matlab control toolbox. The resulting output y_{sys} is saved.

3. Calculate system identification

   The ARX equation is now calculated using Equation (2.11) for a given fixed order, as previously described in Section 2.2.1. Only the time series *u* and y_{sys} are used in this calculation, and the disturbance path therefore remains unidentified. The resulting ARX equation does, however, have the disturbance information embedded in it.

4. Collect open loop data

   The response of the plate just due to *u* or u_{noise} is now calculated. The response of the plate, y_{nocon}, due to disturbance input u_{noise} is calculated using the *lsim* command. During the simulation, the voltage input to the system is set to zero. The response y_{nocon} is used in the performance calculation. Also, the response of the plate to just the piezoelectric voltage u is simulated. The disturbance force is set to zero for this simulation. The response, y_{id}, is saved.

   The simulations in this step allow us to compare the relative vibrations due to the disturbance (y_{nocon}) and the system identification input (y_{id}). [5]

5. Collect preliminary performance curve data

   This optional step was performed to characterize the performance curve for combinations of changing λ/fixed order or changing order/fixed λ. This was done by executing Steps 6a through 6f for fixed intervals of the changing parameter (either order or λ).

[5] The relative levels of *u* and u_{noise} were scaled to ensure that the response y_{id} was larger than y_{nocon} in most frequency regions. In addition, the low frequency region of *u* is boosted to counteract the poor actuation of the piezoelectric patch in that region.

This preliminary performance was used to characterize the effectiveness of the fuzzy logic adaptation.

6. Fuzzy logic adaptation of λ or order

   (a) Design controller

   Given the set of design parameters (control horizon, prediction horizon, λ and order) and the ARX equation from Step 3, a GPC control law is now calculated. Specifically, Equation (2.16) of Section 2.2.2.2 is used for this calculation. The resulting IIR filter is implemented as a Matlab *lti* system using the control toolbox command *tf*.

   (b) Close loop

   Using the appropriate calls to *append* and *connect* in the Matlab control toolbox, the controller closes the loop on the piezoelectric voltage input. The loop is not closed on the first input, which corresponds to the disturbance force input. Figure 3.11 illustrates this process schematically. During the *connect* call, an extra system output is created. This extra output corresponds to the voltage output of the controller. The resulting closed-loop system is a control toolbox *lti* object with one input, the disturbance force, and two outputs, the acceleration measurement and the control voltage.

   (c) Evaluate stability

   The eigenvalues of the closed-loop system created in Step 6b are now calculated. The radial distance from the origin to the location of each eigenvalue is determined. If this distance is larger than 1.0, than the system is unstable, and iteration continues with Step 6g.

   (d) Collect closed-loop data

   The system created in Step 6b is now simulated with the *lsim* command. The disturbance force input is driven by the random sequence u_{noise} created in Step 2. The resulting acceleration output, y_{cl}, and the control voltage, u_{cl}, are both saved. The system response is calculated for a large number of time steps, for example 50,000 in Section 6.1.2.1. The adaptation is therefore a batch-type adaptation, and subsequent control laws will be redesigned based on the results of batches of closed-loop data.

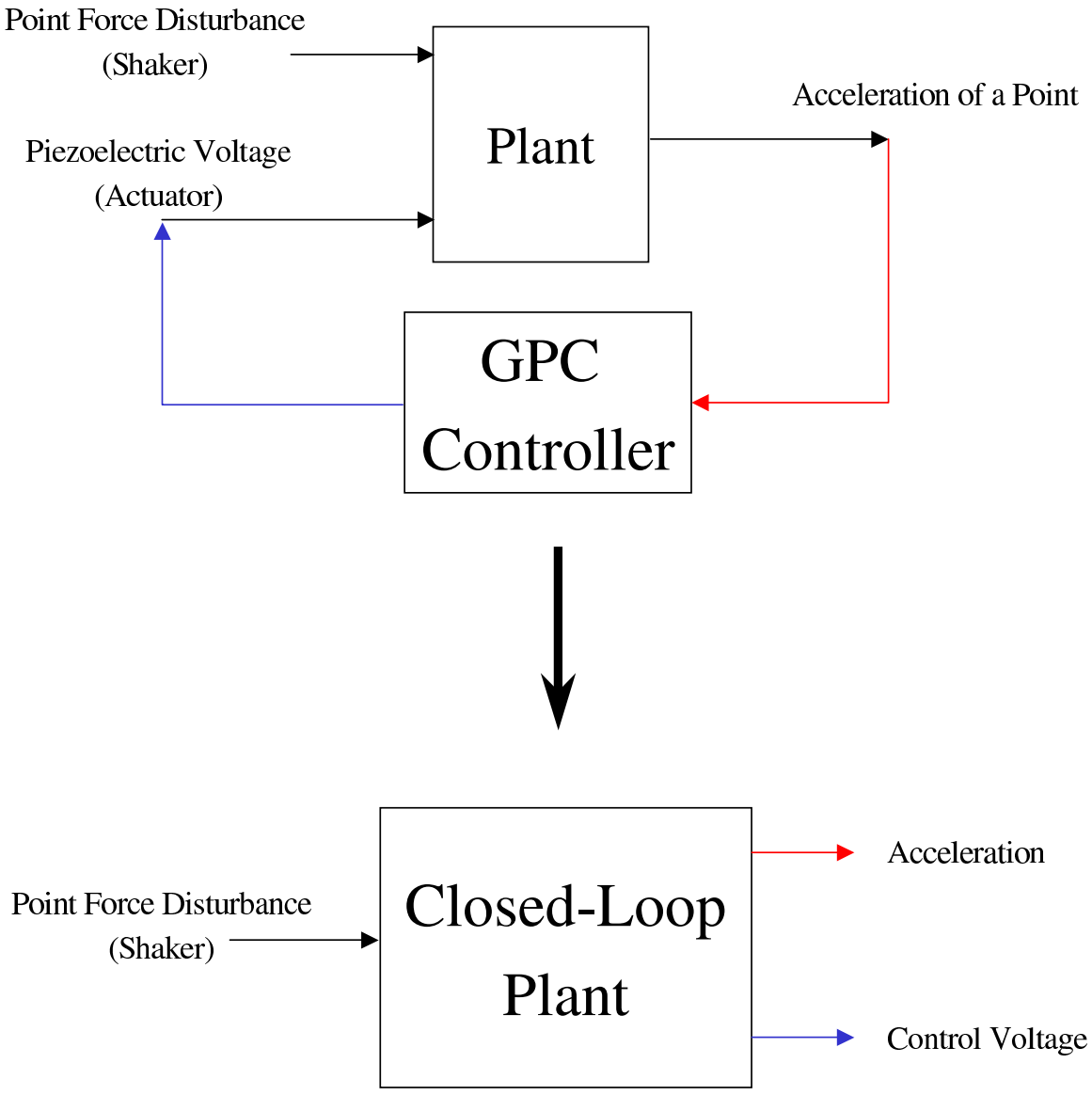


Figure 3.11: Closed-loop System Creation

(e) Calculate performance

We defined the performance as the reduction of vibration level of the closed-loop response (y_{cl}) relative to the open-loop response y_{nocon}. Specifically, the performance is calculated by:

$$performance \equiv 10 \times \log_{10} \frac{\sum P_{closed\,loop}}{\sum P_{no\,control}}$$

where $\sum P_{closed\,loop}$ and $\sum P_{no\,control}$ are the sums of the frequency bins of the autospectrums of y_{cl} and y_{nocon}, respectively.[6]

(f) Calculate maximum control effort

The control voltage, u_{cl}, exerted in the closed-loop simulation of Step 6d is analyzed. During the λ adaptations, described in Section 6.1, the maximum control voltage is calculated by:

$$U_{current} = \max\left(|u_{cl}|\right).$$

For the order adaptations, a more sophisticated statistical calculation was performed, as shown in Section 6.2.1.3.

(g) Calculate stability module output

If the closed-loop system was found to be unstable (Step 6c), this fuzzy logic module attempts to revert the adapted parameter (λ or order) back to a previously stable level. For further information, see Sections 6.1.1.1 and 6.2.1.1.

(h) Calculate U_{max} module output

This module evaluates how close $U_{current}$ is to the maximum allowable control effort. As the controller effort increases, the module will slow or stop the adaptation in order to keep the effort below the maximum allowable, as shown in Sections 6.1.1.3 and 6.2.1.3.

(i) Calculate derivative module output

The shape of the performance curve is analyzed by the derivative module. If this curve flattens out, i.e. there are no performance improvements for further adaptation, then this module will stop the adaptation. Sections 6.1.1.2 and 6.2.1.2 explain this module

[6] Alternatively the performance could be calculated with the ratio of the root mean square values of y_{cl} and y_{nocon}.

in detail.

(j) Evaluate metarules

This fuzzy logic system takes the outputs of the first three modules and comes to a final conclusion on how to adapt the parameter. For λ adaptations, this takes the form of

$$\lambda_{new} = Factor \times \lambda_{old}$$

where $Factor$ is the output of the metarules module. The variable λ_{old} is the control weighting used in Step 6a for the current iteration, and λ_{new} is the control weighting that will be used in the next iteration. The order adaptations use the formula

$$O_{new} = O_{old} + Factor \times step\,size$$

where $step\,size$ is a user-defined constant. For a detailed explanation, see Sections 6.1.1.4 and 6.2.1.4.

(k) Stop or go back to Step 6a

The default operation for this step is to return to Step 6a and therefore continue the iterations. Note that further iterations do not necessarily mean that the adaptation parameter (λ or order) is changing. The user is also given the option of stopping the iteration.

When we are adapting the order, an additional step must be taken before we can continue with Step 6a. The controller calculation of Step 6a was based on an ARX equation of order O_{old}. Since we wish to change the order, a new ARX equation of order O_{new} must be calculated. Therefore, Step 3 is repeated before the iteration continues with Step 6a.

Chapter 4

Laboratory Setup

The laboratory setup consists of a shaker, an accelerometer, a clamped plate with a piezoelectric patch, and an assortment of filters and signal conditioning boxes. A complete description of the setup is given in Sections 4.1 and 4.2. Just as in the numerical simulations, the shaker provides a random disturbance excitation to the clamped plate. An accelerometer is used to measure this vibration, and the piezoelectric patch is the control actuator.

A digital signal processing chip (DSP) board mounted inside a personal computer was used in conjunction with a Matlab process running on the same computer in order to actively control the plate vibrations. The DSP chip is used primarily to close the loop, collect time data, and sense any closed-loop instabilities. Further details of these functions are provided in Section 4.3.

We have the ability to communicate with the DSP chip from within the Matlab environment, which allowed us to do several things. First, we could write a DSP program that has several operating states, and the state in which the chip is currently running is controlled by Matlab. Second, data that was saved in the DSP's memory could be uploaded to the Matlab process throughout the experiment.

These abilities allowed us to write a Matlab program that executes an entire adaptation experiment from start to finish in an autonomous manner. As was stated in Section 3.4, this same program is also used in the numerical simulations. The specifics of when this Matlab program alters the operating condition of the DSP chip and the steps used in the laboratory experiments are shown in Section 4.4.

4.1 Physical Setup

The system is set up as shown in Figures 4.1, 4.2, and 4.3. The 0.04”-thick aluminum plate is rigidly bolted in a frame that consists of two 25”x20”x1” aluminum pieces. Both of these pieces have a 14”x10” rectangular cutout that exposes the thin aluminum plate. This exposed area is the vibrating surface. This surface is surrounded by twenty-six evenly-spaced bolts that are all tightened to the same degree. The result is a 14”x10” plate with approximately clamped-plate conditions[1]. The assembled system weighs in excess of 40 lbs and is extremely rigid. This frame is set upright in wooden blocks.

The shaker (B&K Type 4810) is suspended with string from a hanger that is rigidly attached to the workbench (see Figure 4.3). If the shaker were suspended from the frame, the vibration induced by the shaker might in turn cause a vibration in the shaker hanger. This would cause the dynamics of the plate system to be coupled with the dynamics of the shaker. This is an undesirable effect that can be minimized by hanging the shaker in the manner shown. The shaker applies force to the plate via a stinger. The stinger is screwed into a load cell (PCB 208A02). The shaker is used to apply the disturbance vibration to the plate.

The piezoelectric patch (PZT 5A) is glued to a position in the center of the plate (Figures 4.1 and 4.2). The piezoelectric patch is the actuator with which we achieve control. The dimensions of the patch are approximately 3”x1.5”.

The accelerometer (Endevco Model 2250A-10) is mounted to the center of the plate in these photographs. This type of accelerometer is extremely small and lightweight (0.4 gm). A photograph of the device next to a dime is shown in Figure 4.4. It was mounted to the plate with accelerometer wax. The signal generated by the accelerometer is the error signal for the control law.

4.2 Connections and Signal Conditioning

A complete wiring diagram of the laboratory configuration is shown in Figure 4.7. The accelerometer signal is sent to the PCB Model 481 signal conditioner. The unit includes an adjustable seventh-

[1] As was shown in Section 3.3 (Table 3.2), the boundary conditions of the laboratory plate are between the theoretical simple-supported and clamped-plate conditions.

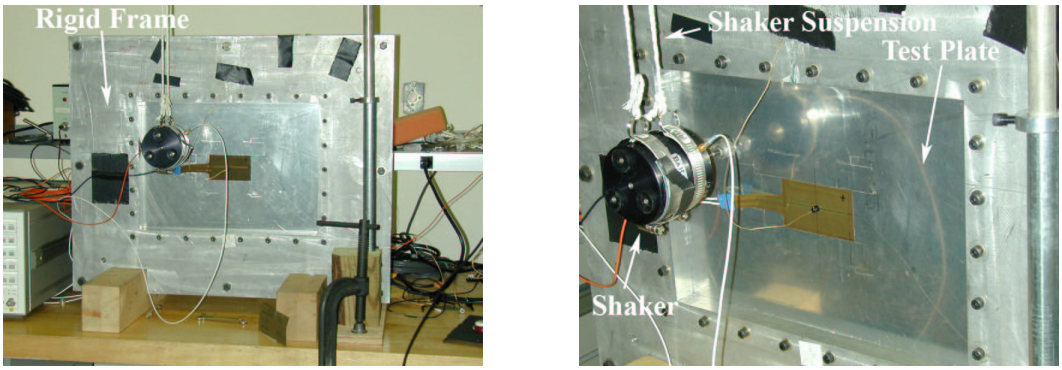


Figure 4.1: Laboratory Setup

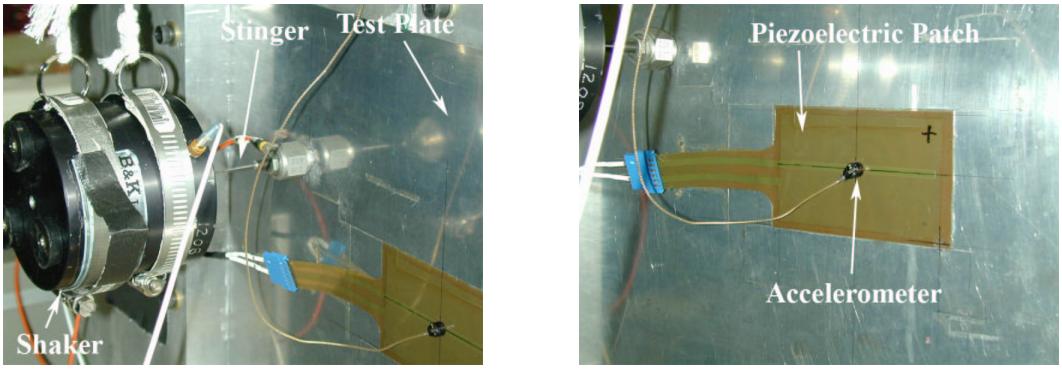


Figure 4.2: Close-up Views of Plate

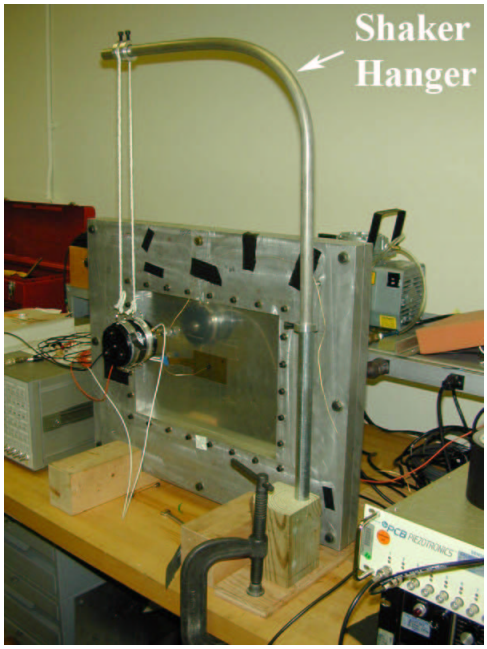


Figure 4.3: Shaker Suspension Overview

Figure 4.4: Accelerometer Size Comparison

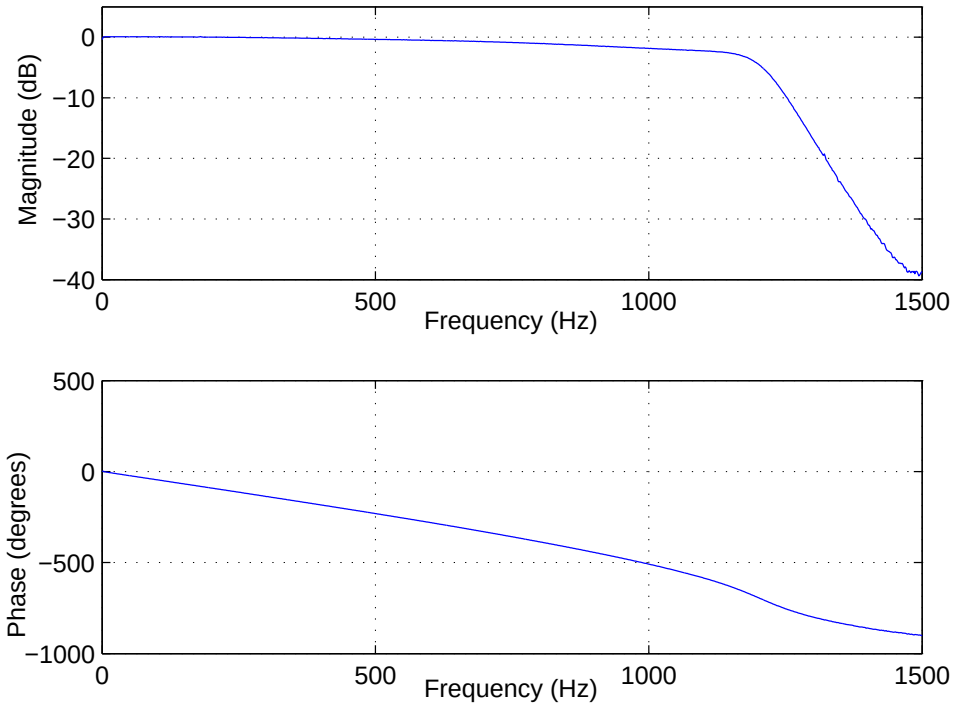


Figure 4.5: PCB 481 Anti-aliasing Filter - Transfer Function

order anti-aliasing elliptical digital filter. The cutoff frequency of this filter was set to 1350 Hz. The gain of the accelerometer signal is set to 50. The output of the signal conditioner is sent to the first input channel of the DSP board via a breakout box.

The transfer function of the anti-aliasing filter, with the PCB signal conditioner set to voltage mode, the gain set to one, and the cutoff frequency set to 1200 Hz, is shown in Figure 4.5. This figure was generated by sending a random noise from the DSP board to the input of the PCB unit, with the output of the unit sent back to the DSP board. The transfer function is then estimated with the Matlab *tfe* command. Setting the cutoff frequency to 1200 Hz proved to be undesirable in early experimentation. The large attenuation of the accelerometer signal near the Nyquist frequency caused the closed-loop system to become unstable at that frequency region for relatively high λs. Moving the cutoff frequency to 1350 Hz solved this problem and was used throughout this work.

The DSP generates two outputs. The first output is connected to the piezoelectric patch. The B&K shaker signal is the second output of the DSP board. The piezoelectric signal is amplified by a PCB Series 790 amplifier, and the shaker signal is amplified by a B&K Type 2706 amplifier.

Both signals are sent through an Ithaco Model 4113 filter array before they are amplified. This

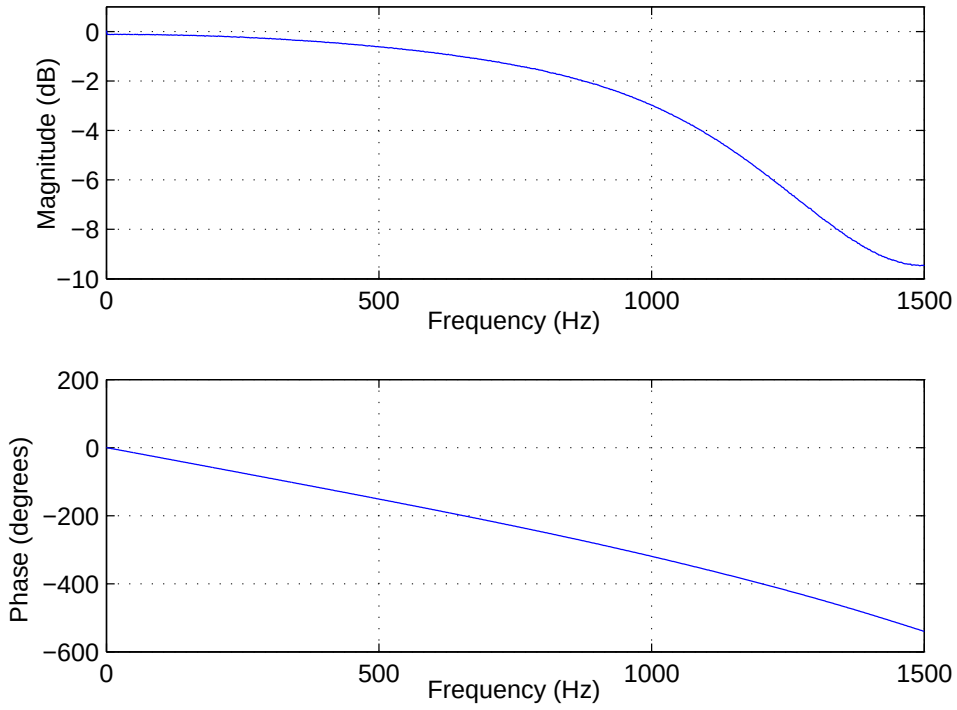


Figure 4.6: Ithaco Filter - Transfer Function

array includes both a high-pass filter and a low-pass filter. The high-pass filters were set to 4 Hz for the piezoelectric signal and 1 Hz for the shaker signal. The low-pass filters were set to 1250 Hz for both signals. The transfer function of this filter, with only the low-pass setting turned on, is shown in Figure 4.6. The roll-off of these filters at high frequencies is very gradual. The plate is therefore excited at frequencies well past the Nyquist frequency.

The purpose of this filter array is to remove any DC component of the signal and to remove very high frequency content. Since the output of the DSP board is a discrete-time signal, the voltage of each output is held constant between sampling intervals. The time history of the voltages therefore consists of a series of step changes in voltage, which generate unwanted high-frequency content in the analog signal.

4.3 DSP Operation

As was mentioned in the introduction, both a Matlab program and a digital signal processing chip (Texas Instruments TMS320C40) are used in the adaptation process. A Matlab program executes the experiments by controlling the various operating states of the DSP chip. This program is further

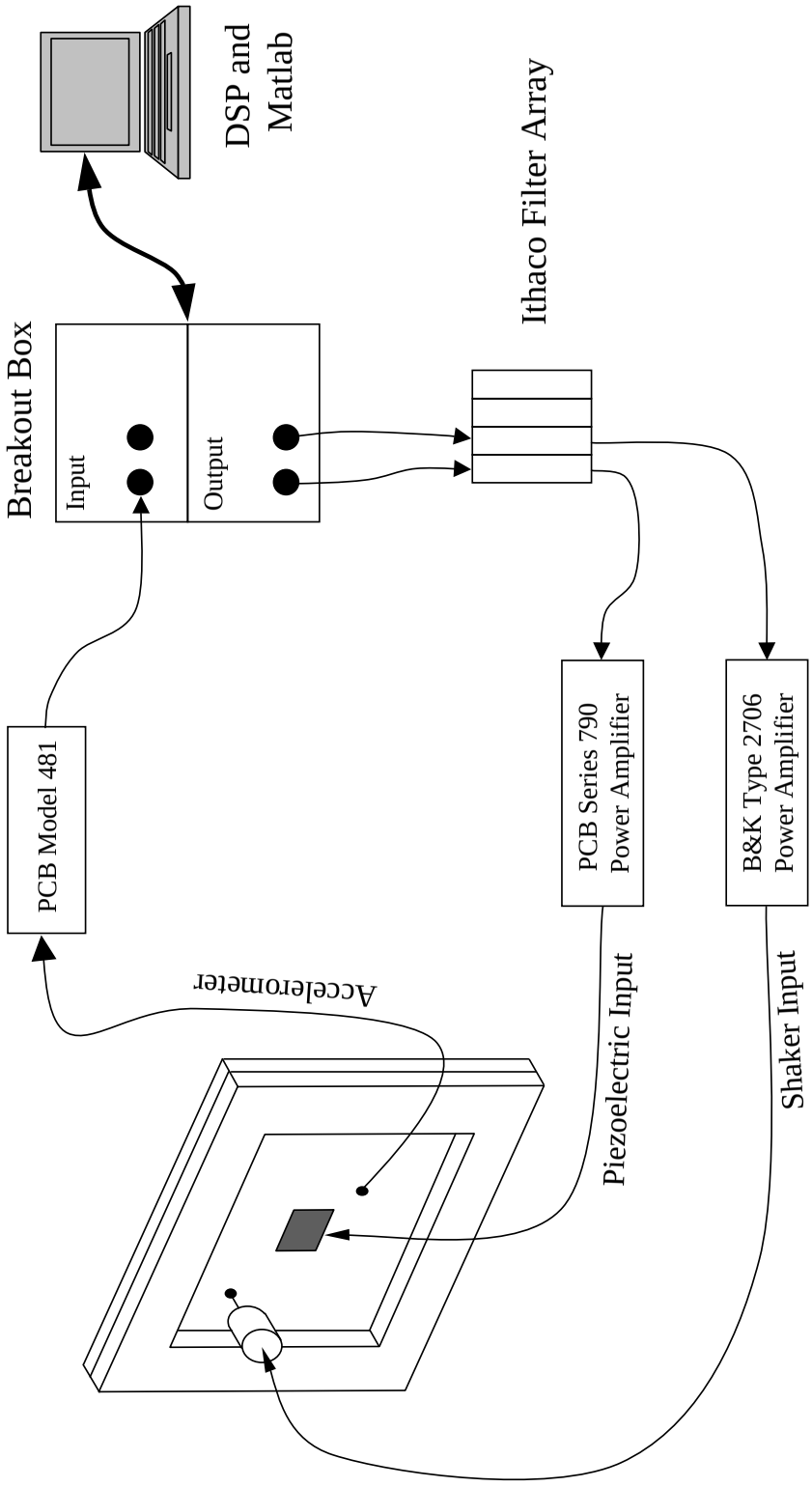


Figure 4.7: Wiring Diagram for Laboratory Setup

described in Section 4.4.

The DSP chip itself runs a program whose operation is detailed in this section. This program completes three main functions: calculation of the control voltage, collection of response data, and sensing any closed loop-instability. A compiled version of this program is downloaded to the chip by Matlab at the beginning of each experiment. The DSP board also has the ability to read and output data by a separate input-output board. It is this input-output board that is connected to the breakout box of Figure 4.7. The acceleration signal is sampled by the input-output board and sent to the DSP chip. The piezoelectric patch voltage (control signal) and the random disturbance signal sent to the shaker are both generated by the DSP chip. The input sampling and output voltage are always set at the sample rate of 3000 Hz, which is also the sampling rate of the controller. The communication between Matlab and the DSP chip does not interfere with this sampling process[2].

A large part of the DSP program is an operating system developed by Ran Cabell of the NASA Langley Structural Acoustics Branch. This operating system is used to execute a series of tasks at the beginning of each sampling interval. In our particular case, we have defined three tasks: the input/output task, a buffer task, and the GPC task[3]. The tasks are always completed before the next sampling interval occurs.

The input/output task collects information from the acceleration channel and sends out signals to the piezoelectric patch and shaker. The values of these signals are calculated by the GPC task. The buffer task is used to collect real-time information from the DSP chip. Once activated, specific variables on the DSP that correspond to the sampled acceleration, control voltage, shaker input, etc. can be stored in these buffers for a user-defined amount of time. The resulting time vectors of data are then uploaded to Matlab for analysis. The process of using these buffers is therefore analogous to using the *lsim* command in the numerical simulations.

The GPC task itself has three separate modes: system identification, open-loop, and closed-loop. The mode in which the task is operating is set by the Matlab program running the experiment. In the system identification mode, a random signal is generated by the DSP chip and is then sent to the piezoelectric patch[4]. In the open-loop mode, the voltage sent to the actuator is 0V.

[2]The communication between Matlab and the DSP chip usually takes several time steps and is not guaranteed to be lag free.

[3]The input/output task, the buffer task, the linux driver and the Matlab programs that enable the communication between Matlab and the DSP chip were all written by Ran Cabell.

[4]The low frequency content of the random signal is boosted by an IIR filter to counteract the weak actuation of the

The closed-loop mode of the GPC task uses the control law to actively control the plate vibrations. Just as in the numerical simulations, the controller uses the acceleration as its input and the piezoelectric voltage as its output. The control law is implemented as an IIR filter. The control voltage is therefore calculated by an inner product of two vectors. The first vector is made up of the filter coefficients, while the second vector consists of past values of the control voltage and the acceleration response. This second vector is updated with the current values at the beginning of each time step.

If the control voltage calculation results in a voltage whose absolute is larger than 10V (the hardware limit of the input-output board), the GPC task is programmed to limit the voltage to 10V. The task also counts the number of times this limit is breached. Once the count becomes larger than 500, the GPC task reverts back to its open-loop mode in order to protect the equipment from breaking.

This automatic switching to the open-loop mode also became our indicator for stability. It was noted in early experiments that a closed-loop instability resulted in two conditions. The first condition was that the autospectrum of the closed-loop acceleration was higher than the corresponding open-loop acceleration autospectrum at most frequencies[5]. This difference in vibration levels was also very noticeable from an audible standpoint, as the unstable system radiates an alarmingly loud roar. In contrast, a stable closed-loop system emits a relatively quiet white noise. The second symptom of instability is a control signal that grows to larger and larger absolute values. The signal eventually starts to clip and then rapidly alternates between values of 10V and -10V. A stable closed-loop condition, on the other hand, resulted in bounded control voltages whose absolute values were well below the 10V limit.

Hence, a large amount of clipping in the control signal could be used as an indicator of closed-loop instability. Since we had already programmed the GPC task to revert back to its open-loop mode after clipping an arbitrarily-chosen 500 times, the reversion itself could be used to indicate an instability of the closed-loop system. Although a count of 500 seems small, the corresponding time interval of the instability is at least 0.17 seconds long. This is long enough to hear the instability. The test for instability is therefore to check the operating mode of the GPC task. If the task is

piezoelectric patch.

[5]The instabilities usually start with a large amplitude increase in the vibration at a single frequency. As the instability grows, the vibration levels at all frequencies are increased.

running in its open-loop mode, when it should be operating in its closed-loop mode, then the system has become unstable.

There is one important exception in which this test for instability fails. It is of course possible that a stable closed-loop system would generate control voltages consistently above 10V. Such a signal would cause the GPC task to eventually revert to its open-loop mode, thereby indicating a closed-loop instability. To counteract this effect, we have carefully adjusted the disturbance excitation level and the gain of the control signal amplifier so that most stable controllers would not generate much more than two or three volts. In addition, we have designed the fuzzy logic rules to keep the control voltage well below the 10V limit.

A critical component of the GPC task is the alteration of the control law while operating in closed-loop mode. The GPC task is placed in its closed-loop mode at the beginning of the adaptation procedure, as described in Step 6b of Section 4.4. As the adaptation continues, new control laws are generated in Matlab and then downloaded to the DSP chip. The DSP program then begins to use these new laws in its control voltage calculation as they become available. Throughout this switching process, the GPC task continues to operate in its closed-loop mode.

To facilitate the control law switch, the DSP program reserves two memory locations on the DSP chip that correspond to two separate control laws. Only one of these filters is used to calculate the control voltage at any specific time. A pointer is used to mark the memory location of this active filter. The other memory location is used as the download location for a new control law, i.e Step 6a in Section 4.4. This secondary memory location is the inactive filter.

After a new control law is downloaded to the DSP chip, Matlab sets a flag indicating that the new law should be used. The GPC task checks the value of this flag at every time step. Once the flag is set, the pointer marking the active filter is moved to the memory location of the previously inactive filter. Since IIR filter calculation always uses the active filter in its inner product, the new control law is now used to calculate the current control voltage. The vector containing the past values of the control voltage and acceleration response are not modified during the switching process.

This switching process was investigated at an early stage of the research. Initially it was thought that switching the control laws in such a manner might cause a sudden spike in the control effort, or perhaps even a closed-loop instability. Experiments that collected data during the switching

process were run and analyzed. No undesirable effects were noticed, and this switching process was used throughout the work.

4.4 Laboratory Procedure

As was stated in Section 3.4, the procedure used for the fuzzy logic adaptation in the laboratory is nearly identical to that used in the numerical simulations. This was done intentionally so that we could capture some of the inexact nature of the laboratory environment in the numerical simulations.

In fact, the experiments in both environments are run by the same Matlab program, which was described in Section 3.4. Here we discussed how the numerical and laboratory procedures could be split into two categories: data collection steps and calculation steps. The calculation steps are independent of the experimental environment and are therefore identical in both procedures.

The data collection steps, however, are highly dependent on the experimental environment, and therefore their execution is much different in the laboratory environment. In general, the buffer routine on the DSP chip is used to collect the required data, such as vectors of acceleration response and control voltages, for a given length of time. This data is then uploaded to the Matlab program running the experiment. The type of data gathered using this technique is therefore identical to the type of data generated by the *lsim* command used in the numerical simulations.

There is one significant difference between the numerical and laboratory experiments: once the loop is closed in the laboratory environment, it remains closed unless the system becomes unstable. The particular nature of the DSP hardware is that it runs independently of Matlab. Once the loop is closed by a command issued in Matlab, it remains closed unless the system becomes unstable. Further computations that occur between adaptation cycles, such as a redesign of the control law, occur inside Matlab while the system is still operating in closed-loop mode. The control law is updated to the control law on the fly while the plate is still vibrating. This process was not replicated in the numerical simulation because of the difficulty of incorporating it.[6]

The following is a list of steps used in the laboratory experiments. The steps that are identical

[6]This process could be implemented by using the Matlab Simulink toolbox in addition to the Stateflow toolbox that is designed for these types of rapid changes.

in both environments are not fully described here; the reader is directed to Section 3.4.

1. DSP initialization

   The DSP operating code is downloaded to the DSP chip and initialized by Matlab. The shaker signal is turned on.

2. Perform system identification

   The GPC task is switched to its system identification mode. The plate is therefore driven by the shaker and the piezoelectric patch simultaneously. The buffer task is used to collect the control voltage *u* and the acceleration response y_{sys} for 200,000 samples.

3. Calculate system identification

   This step is identical to that of the numerical simulations procedure Step 3.

4. Collect open loop data

   The shaker is turned off while the GPC task is still in its system identification mode. The acceleration response (y_{id}) is now collected by the buffer task for 200,000 samples.

   The GPC task is set to its open-loop mode and the shaker signal is turned on. The response of plate (y_{nocon}) due just to the disturbance input is now collected by the buffer task.

5. Collect preliminary performance curve data

   This optional step was performed to characterize the performance curve for combinations of changing λ/fixed order or changing order/fixed λ. This was done by executing Steps 6a through 6e for fixed intervals of the changing parameter (either order or λ). This step was often skipped in the laboratory experiments in order to minimize the time between the system identification and the fuzzy logic adaptation.

6. Fuzzy logic adaptation of λ or order

   (a) Design controller

   The control law is calculated using the four design parameters and the ARX equation. This calculation is performed in Matlab, and the resulting filter coefficients are sent to

a memory location on the DSP chip that corresponds to the currently inactive control law.

(b) Switch control laws and close loop

Matlab sets a flag to indicate that a new control law has been downloaded. The Matlab also sets the DSP GPC task to operate in its closed-loop mode. From this point onward, the DSP will calculate a control voltage with the current control law.

(c) Collect closed-loop data

The buffer task is used to collect a sample of closed-loop data. Both the acceleration response (y_{cl}) and the control voltage (u_{cl}) are collected, and the resulting vectors are uploaded to Matlab. The number of samples collected varied from 30,000 to 80,000 and is noted in each case of Chapter 6.

(d) Evaluate stability

Matlab now polls the state of the DSP GPC task. If the task is operating in the open-loop mode, then an instability has occurred. Similarly, if the GPC task is in its closed-loop mode, then an instability has not occurred.

(e) Calculate performance and calculate maximum control effort

The performance and maximum control effort are calculated in the same manner as in the numerical simulations procedure Steps 6e and 6f.

(f) Calculate fuzzy logic module outputs

The relevant data are input into the fuzzy logic modules (see Steps 6g-6j of the numerical simulation procedure and Chapter 6)

(g) Stop or go back to Step 6a

This step is identical to Step 6k of the numerical simulations. The default operation for this step is to return to Step 6a and therefore continue the iterations.The user is also given the option of stopping the iteration.

It is important to note that the operation of the DSP chip is not altered in any way during this step. Therefore, the plate continues to be disturbed by the shaker input. Also, the controller continues to suppress plate vibrations unless a closed-loop instability occurs.

Chapter 5

GPC Parameter Selection and Closed-Loop Behavior

The generalized predictive control law is a function of four design parameters: the control and prediction horizons, the control weighting λ, and the desired controller order. We are interested in the effect of variations of these parameters on the broadband disturbance rejection and the maximum controller effort.

Since our ultimate goal was to design fuzzy logic systems to adjust these parameters automatically, we first developed an understanding of how their variations affect the closed-loop disturbance rejection. Hence we must establish some methodology for choosing these parameters.

In Section 5.1, we will show the effect of varying the control and prediction horizons for fixed order and control weighting. We have examined this effect on various systems, both numerical and experimental, and have developed a simple rule of thumb for their selection. For a given plant, the horizons can be set to a fixed number that will ensure a near-optimal performance for a wide range of controller orders and λ. Hence what we have done is to eliminate the need to design fuzzy logic rules for the adaptation of the horizons.

The effect of the remaining parameters, controller order and control weighting λ, is explored in Section 5.2. Here we will not only examine the effect of the parameters on the performance, but also on the control effort. Specifically we investigate the overall shape of the performance and control effort curves for the changes in λ (fixed order) and order (fixed λ) . A close examination of the data from numerical and laboratory experiments revealed that these curves could be categorized

into one of several different types. These trends then lead to a set of ideas around which we generated the fuzzy logic rules. The factors that lead to a particular type of performance curve are also examined.

Up to this point in the chapter, we have examined the closed-loop system in general terms of the attenuation of the disturbance or the effort generated by closing the loop. Although this information was necessary to design the fuzzy logic rules that will automatically tune the design parameters, it does not give us any understanding of how generalized predictive control achieves control.

A detailed investigation of the closed-loop poles and zeros under GPC control is shown in Section 5.3. A theory of what generalized predictive control is doing as λ is reduced to zero is developed using a simple two-mode system disturbed by single tone. It is then shown that the theory holds true for the general case of a broadband disturbance.

5.1 Control and Prediction Horizons

As stated in the introduction, the generalized predictive control law (Equation (2.16)) is a function of four parameters: the control horizon, the prediction horizon, the control weighting λ, and the order of the control law. We will show that these parameters can be split into two groups. The first group will consist of the control and prediction horizon, and the second group will be comprised of λ and order. The goal of this section is to develop a simple rule of thumb to select the control and prediction horizon. Furthermore, it will be shown that for a given system, the horizons can be set to static values independent of λ and order, and that this setting will optimize the closed-loop disturbance rejection. This static setting is approximately two to three times the estimated plant order.

With two of the four GPC parameters set to a static value, the control law essentially becomes a function of two parameters: the control order and the control weighting (λ). The effect of these parameters on the closed-loop disturbance rejection will be discussed in Section 5.2.

Our methodology for investigating the effect of the horizons was to fix λ and order, and calculate the closed-loop disturbance rejection (the performance) as a function of the control and prediction horizons. The result of these calculations is a three-dimensional surface with performance

as a function of control and prediction horizons. We will refer to this surface as the *horizon map*. Changes were then made to λ and order, and the calculations were repeated. These experiments were performed both in the laboratory and via numerical simulation.

In all of the experiments performed, the horizon maps showed the same general trend (see Figure 5.1 for an example): the performance improves dramatically as the control horizon is increased from a small value and then becomes nearly constant once the control horizon increases beyond some critical value. The surface is largely invariant to changes in prediction horizon. The results presented subsequently will show that this surface will retain its general shape for changes in λ and order.

The reason for this performance stabilization was further investigated. Our analysis showed that its cause was that the closed-loop transfer function becomes stationary once the control horizon is set larger than a critical value. This is turn was the result of the control law stabilization. This process was observed in both the numerical simulations and in the laboratory experiments.

5.1.1 Horizon Maps - Numerical Simulations

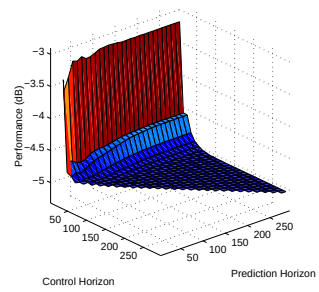
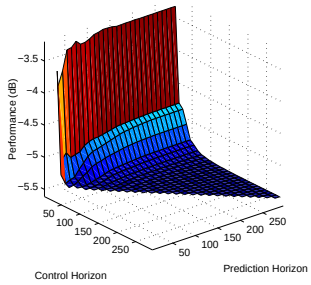
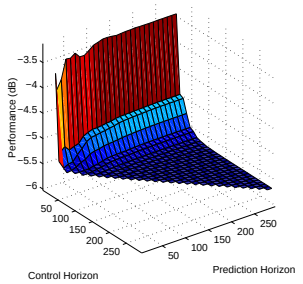
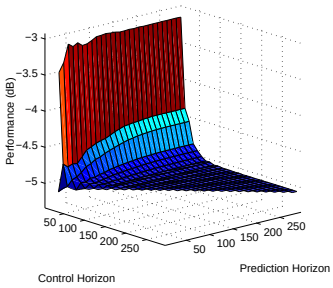
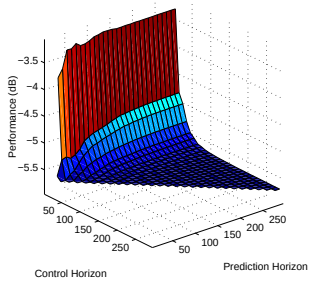
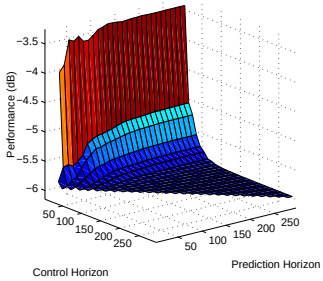
These numerical simulations are performed in the same manner as the fuzzy logic adaptations; specifically Steps 1 through 5 of Section 3.4 are performed. Step 5 is altered so that controllers are designed for a range of control and prediction horizons. Hence the performance surface, i.e. the horizon map, is calculated for a coarse grid in which the control and prediction horizons are varied, but λ and order remain fixed.

Horizon maps were calculated for multiple configurations including a five-mode system, a fifteen-mode system, and various 28-mode systems[1]. The results presented in this section are from a 28-mode system with the acceleration measured at location 6 (2 mm above and to the right of plate dead center; see Figure 3.6). All of the configurations showed essentially the same behavior, but we decided to present the results of the 28-mode systems because they most closely resemble the physical plate in the laboratory.

The horizon maps were generated at orders 45, 70, and 150 with λ set to 1 and 0.1. The overall

[1]Systems with less than 28 modes were created by using only a subset of the admissible functions, therefore excluding entire modes. Hence this process creates plants with less complexity, but their action is no longer completely representative of a physical plate.

Table 5.1: 28-mode Numerical Simulation - Acceleration Measurement Location 6

	Order		
	45	70	150
$\lambda = 1$			
$\lambda = 0.1$			

results are shown in Table 5.1 and explored indepth in Sections 5.1.1.1 to 5.1.1.3. Clearly the results from different pairs of λ and order display the same general trend: a rapid improvement in performance as the control horizon is increased from an initially small value. Once the control horizon reaches a critical value, the performance surface plateaus, and further increases in the control horizon do not improve the performance.

In the following sections, we will show very detailed results for horizon maps with order $70/\lambda = 1$, order $70/\lambda = 0.1$, and order $150/\lambda = 0.1$, which include the evolution of the closed-loop transfer function and control law for changes in the control horizons. The process of examining these plots is very repetitive, and the results are very nearly the same for each case. For this reason, we have chosen not to include this level of detail for horizon maps with order $45/\lambda = 1$, order $45/\lambda = 0.1$, and order $150/\lambda = 1$.

5.1.1.1 Order 70 $\lambda = 1$

The horizon map for order $70/\lambda = 1$ is shown in Figure 5.1. This plot shows that performance improves rapidly for increases in the control horizon from an initially small value. The surface

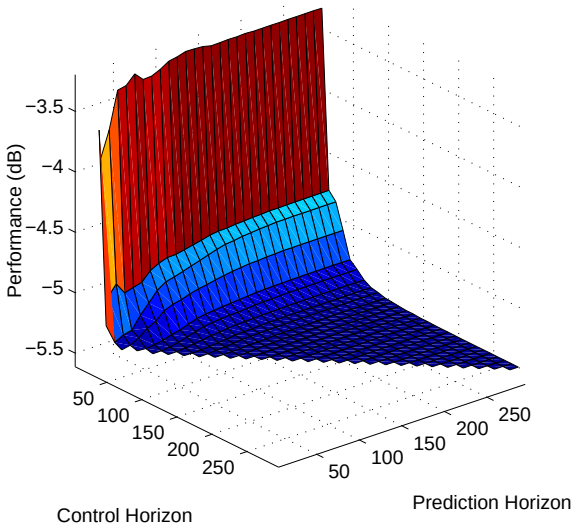


Figure 5.1: Order $70/\lambda = 1$ Horizon Map

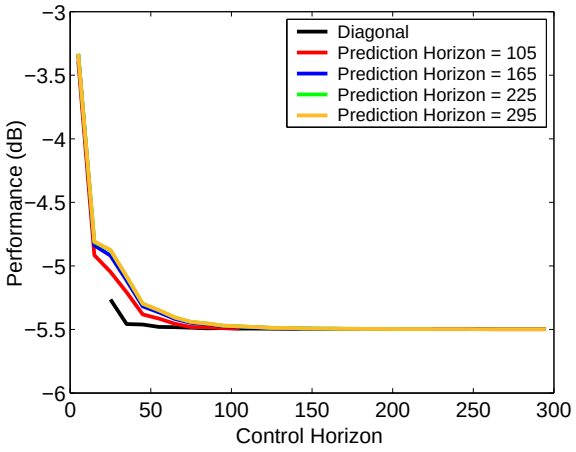


Figure 5.2: Order $70/\lambda = 1$ - Constant Prediction Horizon Performance Curves

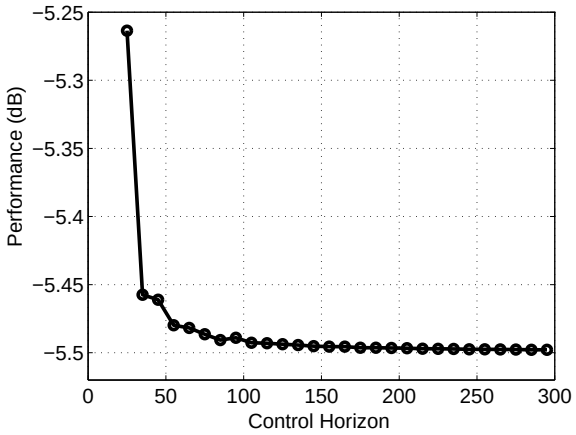


Figure 5.3: Order 70/$\lambda = 1$ - Performance along Diagonal (PH=CH)

then flattens as the control horizon reaches a critical value. Furthermore, the surface appears to be largely invariant to changes in prediction horizon.

The horizon map is cut along constant prediction horizon lines in Figure 5.2, in which we show the performance as a function of control horizon for constant prediction (PH) horizons of 105, 165, 225, and 295. The black line, labeled diagonal, is the line at which the prediction horizon is equal to the control horizon at all times[2], i.e. the performance of the diagonal line shown in Figure 5.1. These curves confirm the trend we saw in Figure 5.1 in that the performance stabilizes once the control horizon is made large enough. All curves stabilize at a value of performance of roughly -5.5 dB. From this figure we see that all curves except the diagonal appear to stabilize at a control horizon of 100.

In Figure 5.3, we have shown the performance for the line labeled diagonal in Figure 5.2, i.e. the line in which the control horizon is equal to the prediction horizon at all times. Here we see that the performance improves dramatically from a change of horizons from 25 to 35. Once the horizons reach 55, the gains in performance become much smaller. For control horizons of 105 or higher, the improvements in performance are even smaller. At this point, the changes in performance are so small that the performance can be considered stable.

We will now show that this stabilization in performance is the direct result of stabilizations of the closed-loop transfer function caused by stabilization of the control law. To consider a closed-

[2]The diagonal arises out of the fact that in the GPC control law design, the control horizon is always less than or equal to the prediction horizon.

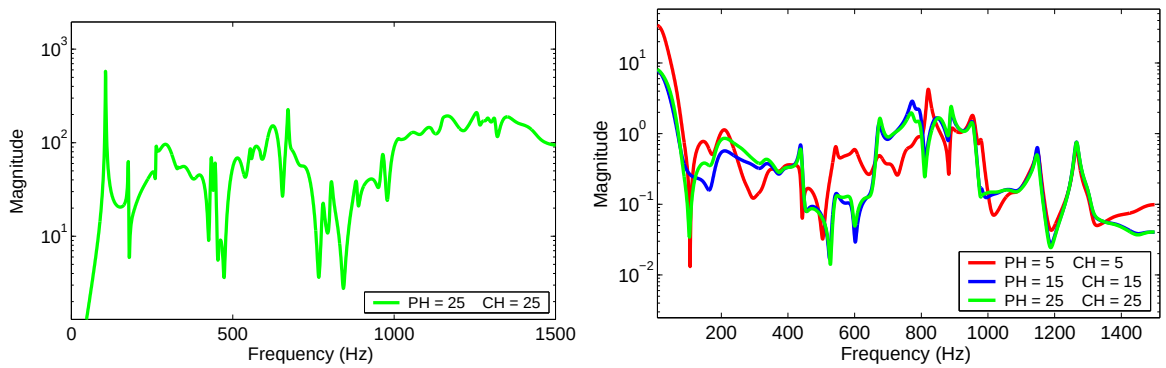


Figure 5.4: Order 70/$\lambda = 1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 1

loop transfer function or control law stabilized is somewhat different than a stabilization of a scalar quantity such as performance. A closed-loop transfer function (or control law) can only be described in mathematical terms as a set of absolute pole and zero positions. Hence, in order to consider a control law stabilized, we must look at either the change in absolute pole and zero positions by plotting their position on the complex plane or in a tabular format. Alternatively, a change in pole or zero position will be reflected by a change in the frequency response or the system. Since a visual comparison is easier to do in the frequency domain, we have chosen to view the changes in the system in that domain. We will consider a system stabilized once the changes in every frequency bin can be considered minimal relative to the absolute level.

In Figures 5.4 to 5.7, we have shown the evolution of the transfer function and the control law along the diagonal (control horizon = prediction horizon), i.e. the black line in Figure 5.2. Each figure includes two plots: the magnitude of the closed-loop transfer function is shown on the left, and the corresponding magnitude of the control law frequency response is shown on the right.

The first group of plots is shown in Figure 5.4 and includes the closed-loop transfer function and control law for control horizons 5, 15, and 25. The first two settings, PH=5/CH=5 and PH=15/CH=15, result in a closed-loop instability, and the corresponding closed-loop transfer functions are not shown. The control law shows significant changes in almost all frequency regions as the horizons are increased from PH=5/CH=5 to PH=15/CH=15. The change to the next set of horizons, PH=25/CH=25, is not as large. Here the control law is starting to stabilize for frequencies higher than 400 Hz.

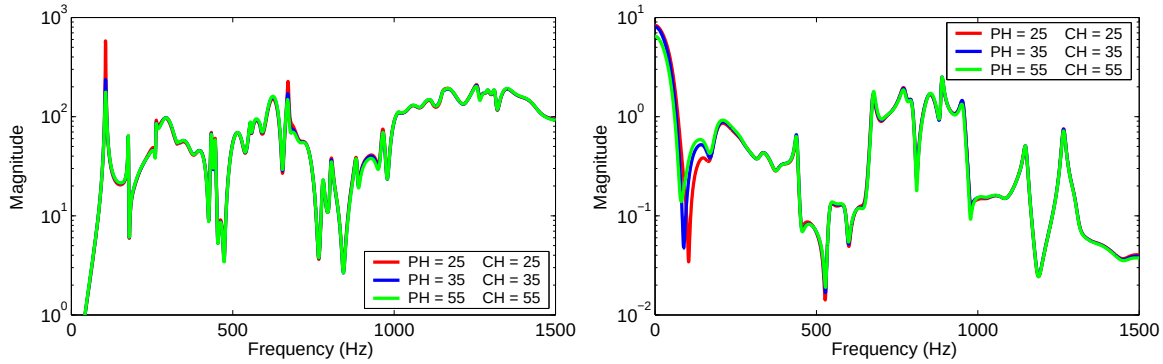


Figure 5.5: Order 70/$\lambda = 1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 2

In Figure 5.5, we have shown the next set of plots for PH=25/CH=25, PH=35/CH=35 and PH=55/CH=55. Clearly the closed-loop transfer function is stabilizing in most frequency regions. There are still significant changes in the modal magnitude at frequencies 106 Hz and 670 Hz. These large changes also explain the steep improvement in performance seen in Figure 5.3. The control law is also stabilizing in all frequencies above 250 Hz, with only minute changes in magnitude occurring in this region. Below 250 Hz there are still significant changes in the control law.

Figure 5.6 continues to show the evolution of the closed-loop transfer function and control law for increases in horizon from 55 to 145. For the transfer function, the only significant change is occurring for the mode at 106 Hz, in which the higher horizons continue to reduce the modal amplitude. There continue to be minute differences in region between 670 Hz and 1000 Hz. The reductions in the modal amplitudes that occur when horizons are increased from 55 to 105 are greater than the corresponding reductions that occur from horizons 105 to 145. This explains why the rate of performance improvement is decreasing (Figure 5.3) for this same interval.

In the previous groups of plots, Figure 5.5, we had seen a control law that was stabilizing above 200 Hz, but changing for frequencies below 200 Hz. In this set of comparisons, Figure 5.6, this continues to be the case. The control law is becoming more aggressive (higher gain) at around 106 Hz, indicating that the controller is adding more energy in this region in order to damp the mode at 106 Hz.

The final group of plots, Figure 5.7, shows the changes for PH=145/CH=145, PH=205/CH=205, and PH=295/CH=295. There continue to be minute changes in the magnitude of the mode at 106

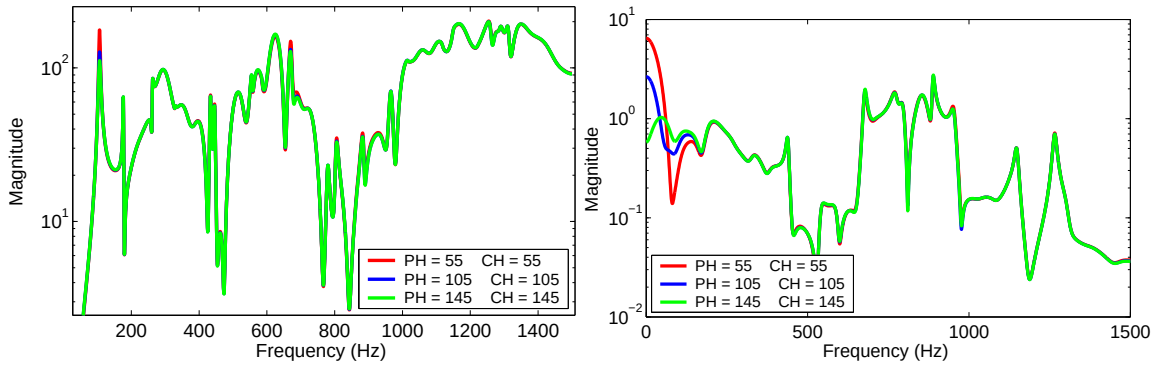


Figure 5.6: Order 70/$\lambda = 1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 3

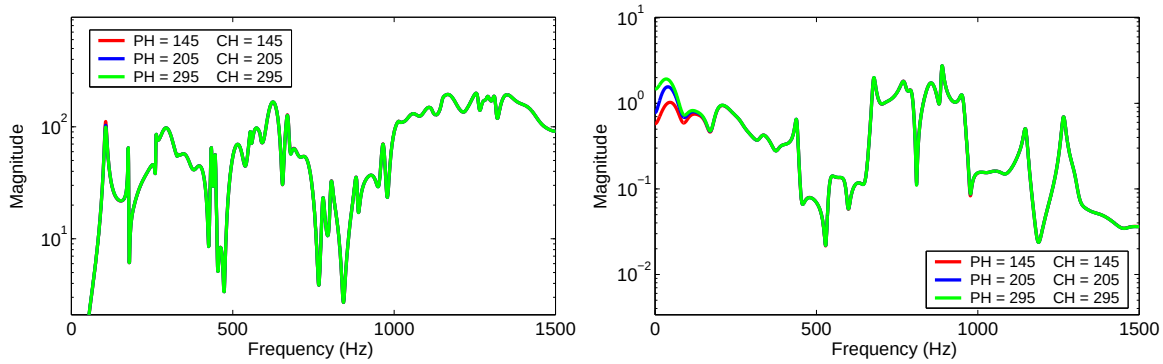


Figure 5.7: Order 70/$\lambda = 1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 4

Hz. The change is so small that the corresponding performance change is only -0.003 dB from horizons 145 to 295.

The control law has also stabilized in all frequency regions above 200 Hz. Below 200 Hz, the control law is still changing in magnitude, but the degree of change is small in comparison to that seen in the previous group (Figure 5.6). Furthermore, these changes in control law at low frequencies account for the increasing damping of the closed-loop mode at 106 Hz. Again the decrease in the magnitude of the changes in the control law accounts for the decreases in the change in the 106 Hz mode.

The preceeding figures allow us to come to a preliminary conclusion on how to set the control and prediction horizons. Clearly the horizon must be selected large enough to avoid any instability in the system. This then would indicate that horizons over 25 must be selected. An examina-

tion of the performance curve, Figure 5.3, indicated that increasing the horizon to 55 leads to a large improvement in performance. Further increasing the horizons to 105 shows a small gain in performance, and increasing the horizons beyond this yields only a minimal improvement in performance. Just based on the performance point of view, there seems to be very little reason to increase the horizons beyond 105.

From the point of view of the stabilization of the control law and the closed-loop transfer function, the situation is less clear. In Figures 5.6 and 5.7, we have shown that control law is well-stabilized for horizons larger than 55 for frequencies higher than 200Hz. This in turn resulted in very minor changes in the closed-loop transfer function for frequencies higher than 200 Hz, with a majority of the changes occurring between horizon 55 and 145 (in particular the mode at 670 Hz). For horizons larger than 145, there is almost no change in the control law or closed-loop transfer function for frequencies higher than 200 Hz.

For frequencies below 200 Hz, the control law continues to change all the way up to horizons of 295. The rate of the changes is decreasing so that by a horizon of 145, the change is very small. The effect of these changes is seen in the closed-loop mode at 106 Hz, whose magnitude continues to decrease with increase in the horizons. This change in modal amplitude becomes small once the horizons have been set to values of 145 or larger.

Hence, from the point of view of the stabilization of the control law or closed-loop transfer function, we must set the horizons to values of 145 or larger. The plant for this system is of order 63, hence this stabilization value is more than two times the plant order. In all of the numerical simulations performed, the stabilization value was found to be a function of the plant order. In every case, it was found that setting the horizons to a value of two to three times the estimated plant order stabilized the control law and closed-loop disturbance rejection. The following sections will show that this setting is independent of λ or order and that this is also true in the laboratory experiments.

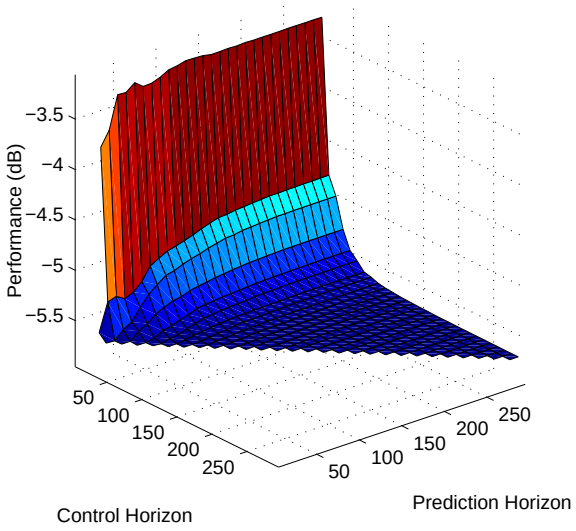


Figure 5.8: Order 70/$\lambda = 0.1$ - Horizon Map

5.1.1.2 Order 70 $\lambda = 0.1$

In this section, we will show the results in the case in which order is set to 70 and λ is set to 0.1. Therefore this is the same setup as in Section 5.1.1.1 with a change in λ from 1 to 0.1.

The horizon map is shown in Figure 5.8. The trend is identical to that seen in Section 5.1.1.1. Specifically, the performance stabilizes once the control horizon is made large enough, and the surface is largely invariant to changes in prediction horizon.

The surface is cut along a constant prediction horizon in Figure 5.9. The diagonal, in which control and prediction horizon are equal at all times, stabilizes between horizons 25 and 50. The remaining curves (PH=105, PH=165, PH=225, PH=295) all show similar behavior: a steep drop in the performance curve from a control horizon of 5 to 50, followed by gradual stabilization. As the control horizon is increased to values over 100, the performance stabilizes on all of these curves.

Similar to that shown in Section 5.1.1.1, we will now demonstrate that the cause of the performance stabilization is the direct result of the control law becoming stagnant once the control horizon is made large enough. We will investigate the behavior of the closed-loop transfer function and the control law along the diagonal. The corresponding performance curve is shown in Figure 5.10. The curve exhibits a large improvement in performance from its first stable point, CH=PH=15, to the next stable point, CH=PH=25. The performance continues to undergo very

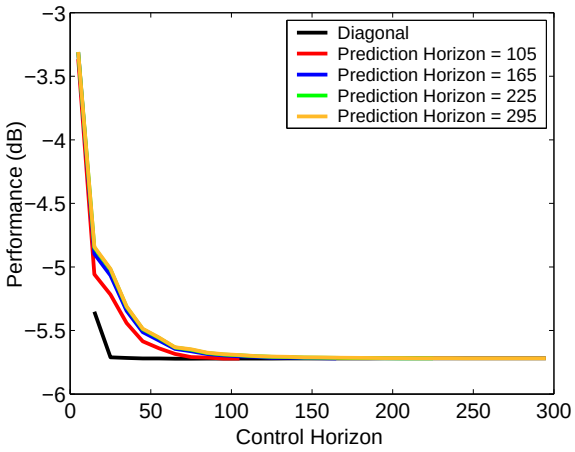


Figure 5.9: Order 70/$\lambda = 0.1$ - Constant Prediction Horizon Performance Curves

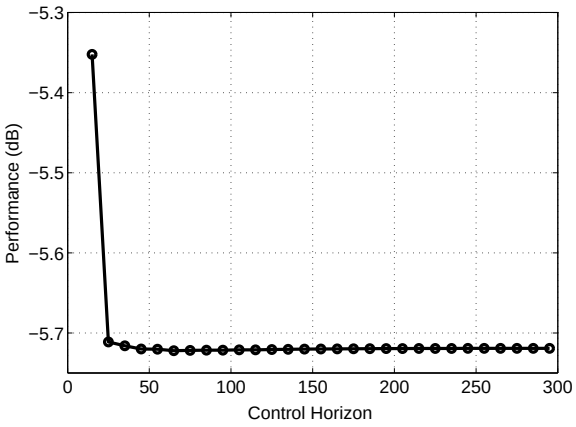


Figure 5.10: Order 70/$\lambda = 0.1$ - Performance along Diagonal (PH=CH)

minor changes as the control horizon is made larger, but is essentially stable after a value of 105.

The first group of closed-loop transfer functions and control laws is shown in Figure 5.11. As in Section 5.1.1.1, the closed-loop transfer magnitude is shown on the left, and the control law magnitude is shown on the right. The first setting, CH=PH=5, results in a closed loop instability, thereby making the closed-loop transfer function irrelevant. The control law is undergoing major changes from CH=5, the unstable control law, to the first stable point CH=15. The next change, from CH=15 to CH=25, results in changes in the control law primarily below 300 Hz and at the controller pole at 800 Hz, as well as minor changes in the rest of the bandwidth. However, this seemingly small change in the control law results in the large attenuation of the plate modes at 106 Hz and 650 Hz, and therefore the large improvement in performance seen in Figure 5.10.

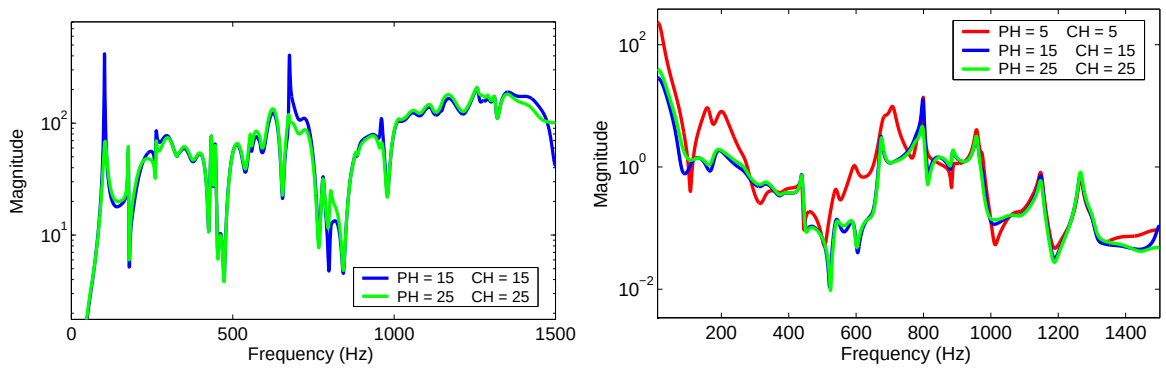


Figure 5.11: Order 70/$\lambda = 0.1$- Closed-loop Transfer Function and Control Law - Diagonal - Group 1

The next group of controllers and transfer functions is shown in Figure 5.12, where the control horizon is varied from 25 to 105. There continue to be small changes in the closed-loop transfer function, particularly at modes occurring at 106 Hz, 673 Hz, and in the frequency region between 880 Hz to 970 Hz. The magnitude of the changes is also decreasing as the horizons are increased.

Although the effect at the 106 Hz mode is relatively large as the horizons are increased from 25 to 65, the magnitude of the mode is almost a degree of magnitude below the dominant response between 1000 Hz and 1500 Hz. Hence the energy contributed by this mode is negligible, and the large modal reduction has very little effect on the total closed-loop disturbance rejection. The transfer function has also largely stabilized in the other frequency regions.

The control law has also largely stabilized, but there continue to be significant changes below 200 Hz. The change in horizon is also inducing minor changes in the control law magnitude, particularly in the frequency region between 650 Hz and 1000 Hz. This is difficult to see on the plot as shown, however.

The final group of magnitude responses is shown in Figure 5.13. Here we plot the control law and closed-loop transfer function for control horizon settings from 105 to 195. The three curves essentially lie on top of each other in both the control law and transfer function plots, indicating that both have fully stabilized. Hence, a control and prediction horizon value of 105 represents the point after which the control law has fully stabilized.

In conclusion, increasing the horizon leads to a stabilization in the control law. This in turn causes the closed-loop transfer function and the corresponding disturbance rejection to stabilize.

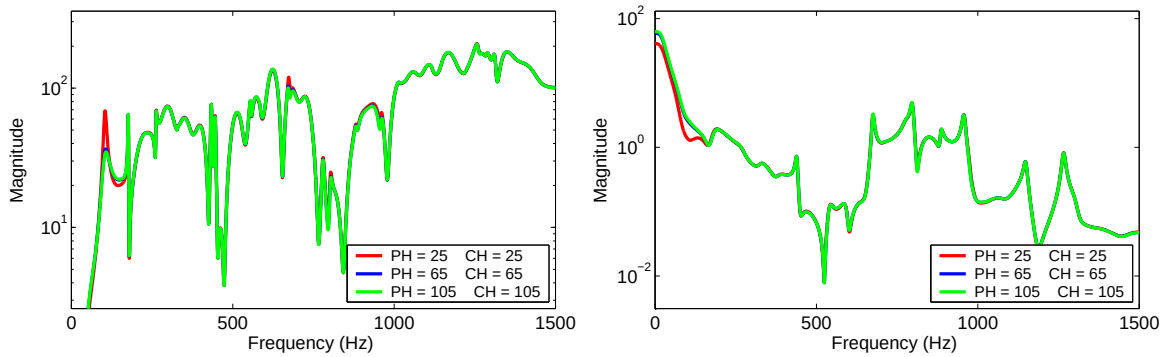


Figure 5.12: Order 70/$\lambda = 0.1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 2

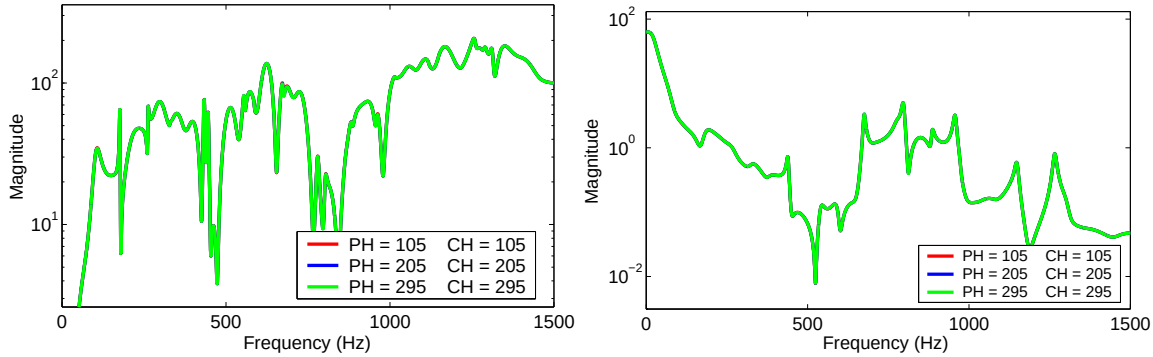


Figure 5.13: Order 70/$\lambda = 0.1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 3

In this particular case, this occurs as the horizons are set to values of 105 or larger. As discussed before, the plant in this particular case is of order 63[3], therefore the stabilization point is nearly 1.5 times the plant order.

In the first case, Section 5.1.1.1, the horizon value of 145, over two times the plant order, was needed to stabilize the control law. This larger value would therefore also ensure that the control law in the second case has stabilized and the performance has optimized. As was stated before, a general rule of thumb for setting the horizons is to assign them values of two or three times the plant order. These last two cases illustrate that this optimizes the performance and stabilizes the control law.

[3]There are 28 modes in addition to the seventh-order anti-aliasing filter.

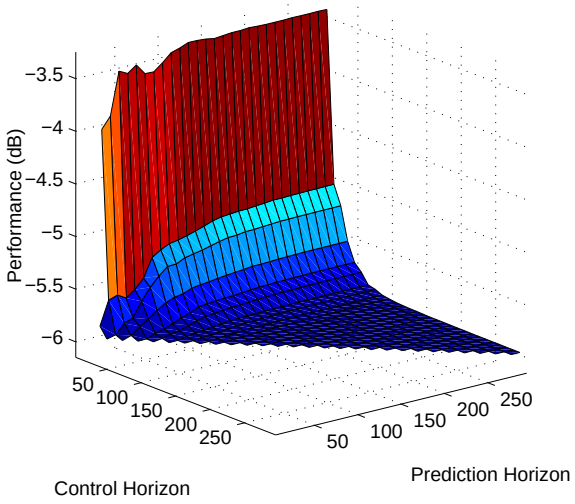


Figure 5.14: Order 150/$\lambda = 0.1$ - Horizon Map

5.1.1.3 Order 150 $\lambda = 0.1$

This section shows the results for the numerical simulation in which order is set to 150 and λ is set to 0.1. Hence this is the same configuration as in Section 5.1.1.2, with an increase in order from 70 to 150.

The horizon map for this system is shown in Figure 5.14. The surface displays the same general trend as the previous two cases, i.e. the performance improves dramatically as the control horizon is increased from an initially small value. Once the horizon reaches a critical value, the performance stabilizes. Also, the surface is largely invariant to changes in prediction horizons.

The surface is cut along constant prediction horizons curves in Figure 5.15 to provide additional details of the surface. All lines other than the diagonal confirm the trend we have previously discussed. Specifically, the large improvement in performance occurs between a control horizon of 5 and 50. By a control horizon of roughly 100, these curves have largely stabilized.

The diagonal is shown again in Figure 5.16. This curve undergoes a very steep drop from the first stable point (CH=15) to the second point (CH=25). For control horizons larger than 35, there continue to be some minor changes in the performance until the horizons reach 65. At that point, the performance curve stabilizes.

This performance stabilization is a direct result of the stabilization of the control law, just as it was in the previous sections. To prove this, the control horizon range was split into three groups,

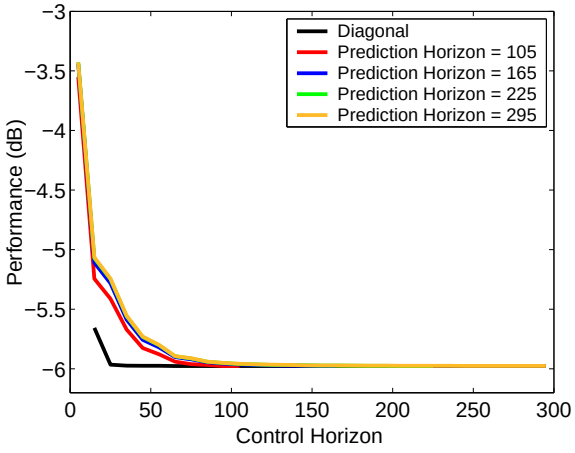


Figure 5.15: Order 150/$\lambda = 0.1$ - Constant Prediction Horizon Performance Curves

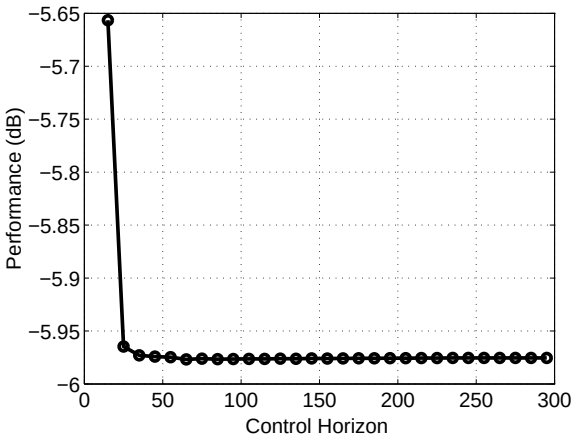


Figure 5.16: Order 150/$\lambda = 0.1$ - Performance along Diagonal (PH=CH)

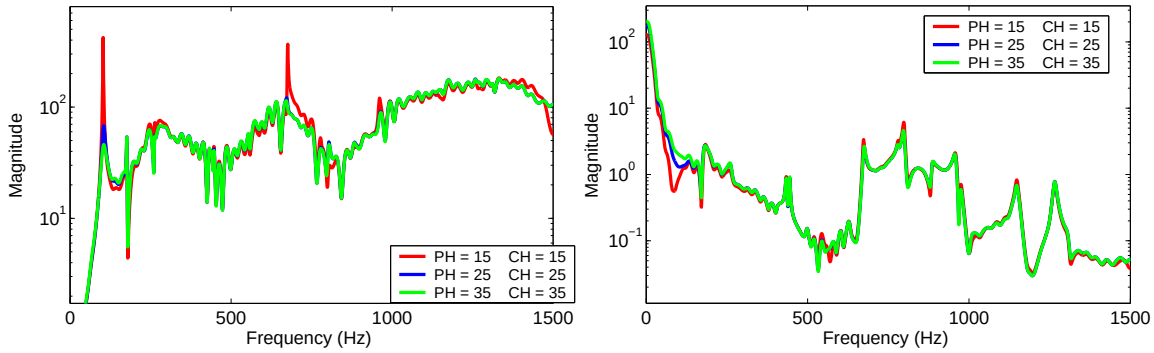


Figure 5.17: Order 150/$\lambda = 0.1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 1

and the changes in the closed-loop transfer functions and control laws within each group were tracked. The results of this analysis are shown in Figures 5.17 to 5.19.

The first group, Figure 5.17, shows the transition from horizons of 15 to 35. This region corresponds to the large drop in the performance curve shown in Figure 5.16. The cause of improvement is very clearly the increased damping added to the closed-loop modes at 104 Hz and 676 Hz, especially as the horizons are increased from 15 to 25. Further increasing the horizons to 35 incurs further modal magnitude reductions, particularly at the 104 Hz mode, but the overall change is not as great as in the previous change. In the other frequency regions, the closed-loop transfer function is starting to stabilize.

The control is also changing significantly in certain frequency regions as the control horizon is increased from 15 to 35. Specifically in the region below 200 Hz, in which the gain is increased by as much as 100 percent as the horizon is changed from 15 to 25. Also there are large reductions in the controller mode at 798 Hz, as well as minor changes throughout the remaining bandwidth. The control law continues to change, albeit at a smaller scale, as the horizons are increased to 35.

The second group of closed-loop transfer function and control laws is shown in Figure 5.18, demonstrating the effect of changes in horizons from 35 to 105. Clearly the transfer function is stabilizing, since the only significant change is at the closed-loop mode at 104 Hz. However, the modal magnitude is well below the dominant response between 1000 Hz and 1500 Hz, and therefore its reduction has very little impact on the overall disturbance rejection. There are also some minor changes in the region between 650 Hz and 1000Hz, so the transfer function has not fully stabilized.

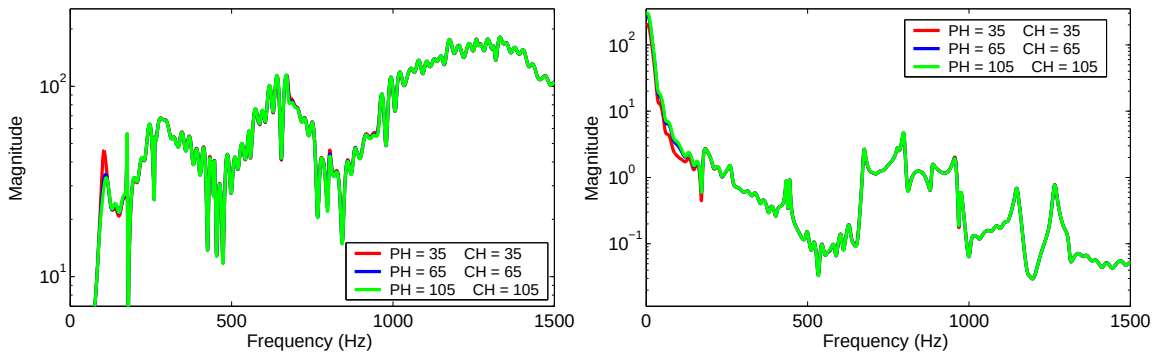


Figure 5.18: Order $150/\lambda = 0.1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 2

The control law is also stabilizing, with the only significant changes occurring below 200 Hz, and a small change occurring at the 955 Hz controller mode. The remaining bandwidth has stabilized.

The final group of comparisons, Figure 5.19, shows the transition from horizons of 105 to 295. The sets of control laws and closed-loop transfer functions lie on top of each other, indicating that both have fully stabilized. Therefore a horizon setting of 105 marks the point of control law stabilization.

This stabilization point is the identical result we obtained in Section 5.1.1.2. More important is that the rule of thumb of selecting horizons two or three times the plant order continues to give us a correct setting for the control and prediction horizons. Furthermore, this rule of thumb is valid for a range of λs and orders. Since this horizon setting is a static value, we have essentially eliminated the horizons as design parameters for the GPC control law.

In Section 5.1.2, we will show that the same dynamics occur in the laboratory system, i.e. that the control law stabilizes at a certain point and causes the corresponding disturbance rejection to stabilize. It will also be shown that the rule of thumb continues to correctly predict this stabilization point in the laboratory setting.

5.1.2 Horizon Maps - Laboratory Experiments

The laboratory experiments are performed in the same general manner as the numerical simulations. Six attempts were made to calculate the horizon map in the laboratory. Of these attempts,

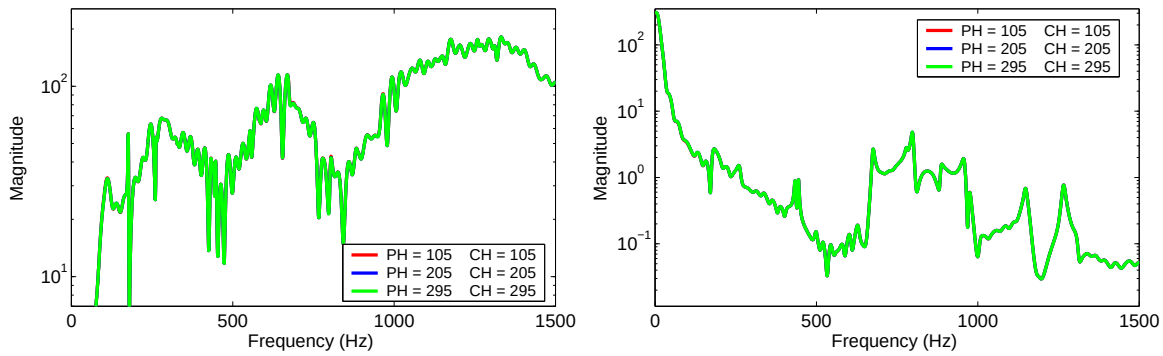


Figure 5.19: Order 150/$\lambda = 0.1$ - Closed-loop Transfer Function and Control Law - Diagonal - Group 3

four can be considered successful. In this section, we will show the results for three of these attempts.

Running this type of experiment in the laboratory takes a long time and this presents problems unique to the laboratory. The system identification is only taken once in the experiment, in the very beginning, and this information must remain accurate throughout the entire experiment. However, there seemed to be small temperature variations inside the building, which in turn caused the plate modes to drift in frequency and therefore make the system identification slightly inaccurate.

Horizon map results for order 150/$\lambda = 0.02$ are shown in Section 5.1.2.1, order 400/$\lambda = 0.02$ results are described in Section 5.1.2.2, and the order 150/$\lambda = 1$ horizon map is detailed in Section 5.1.2.3. For these experiments, the accelerometer is mounted to location 3.

In the laboratory, we do not have the true plant information, hence we cannot plot the closed-loop transfer function. In the numerical experiments, we established that performance stabilizes as a result of closed-loop transfer function stabilizing, which in turn was caused by the controller becoming stagnant. Hence for the laboratory experiments, we will directly infer that the performance stabilization is a direct result of the control law stabilizing.

5.1.2.1 First Case - Order 150/$\lambda = 0.02$

For this laboratory experiment, the accelerometer was fixed to location 3, order was set to 150, and λ was set to 0.02. Control horizon and prediction horizon were set to 15 values.

The resulting horizon map is shown in Figure 5.20. The surface demonstrates the same general

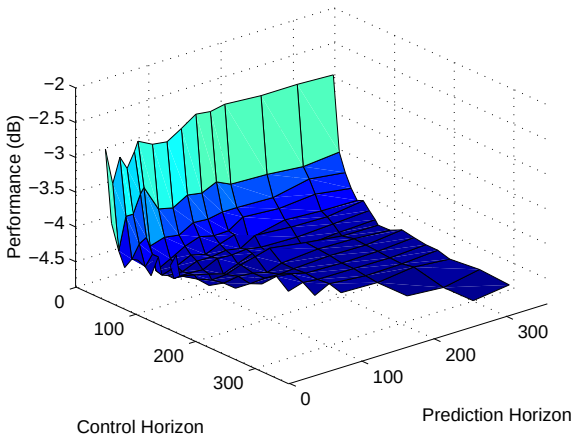


Figure 5.20: Order 150/$\lambda = 0.02$ - Experimental Horizon Map

trend that is shown in the numerical simulations. Specifically, we see that the performance improves rapidly as some control horizon is increased from an initially small value. Once the control horizon is made large enough, the performance stabilizes. Just as in the numerical simulation, the surface is invariant to changes in prediction horizons.

In Figure 5.21, we have shown the constant prediction horizon cuts of the horizon map. The performance curves for prediction horizons of 100, 200 and 350 all display the same general trend: a performance that stabilizes at a control horizon of around 100. These curves also start at a horizon of 15, which is the first value of the control horizon, meaning that the closed-loop system remained stable for even the small horizons for these curves. This is not the case for the diagonal (prediction horizon = control horizon) in which stability is not reached until much later. Nevertheless, the diagonal also becomes stable at around CH=100. The global value of performance is approximately 4.8 dB.

We will now investigate the behavior of the controller and the performance along the diagonal. The performance curve for just the diagonal is shown in Figure 5.22. The closed-loop system remains unstable until the control horizon reaches 65. As the control horizon is increased to 100, the performance improves rapidly. After CH=100, the performance curve is essentially flat.

In Figure 5.23, we have shown the magnitude of the control law frequency response for the first two groups. In the first group, we have shown three controllers between control horizons of 65 and 100. Hence this plot shows the control law as the performance curve (Figure 5.22) is undergoing its initial steep descent. The control law is undergoing large changes as the horizons are increased

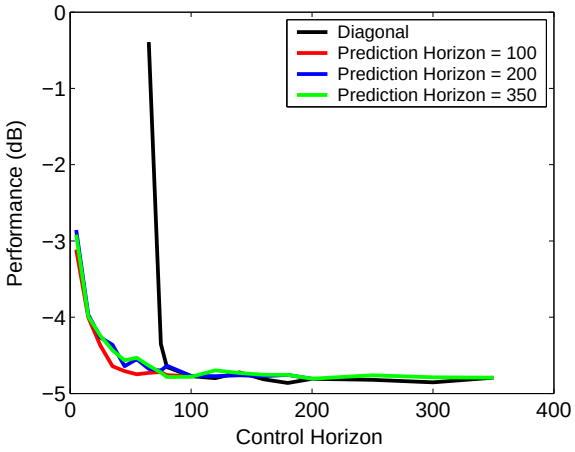


Figure 5.21: Order 150/$\lambda = 0.02$ - Constant Prediction Horizon Performance Curves

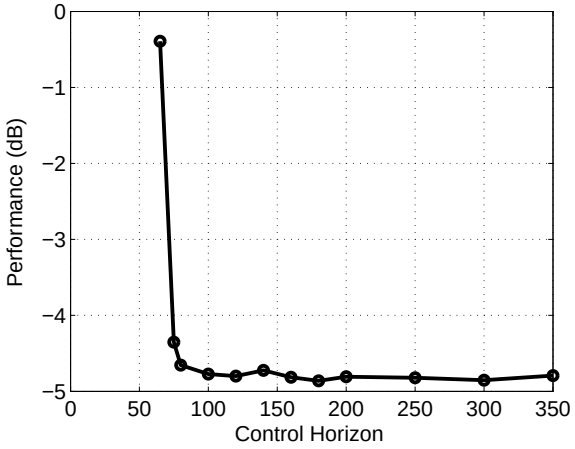
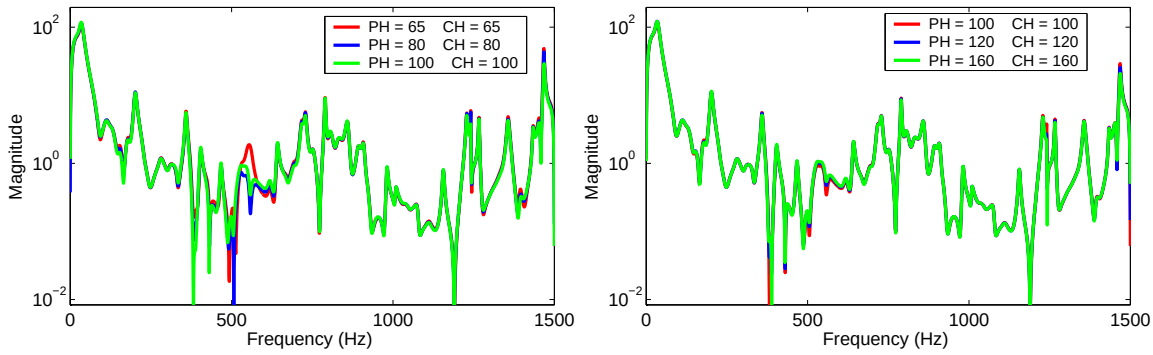


Figure 5.22: Order 150/$\lambda = 0.02$ - Performance along Diagonal (PH=CH)



(a) Group 1

(b) Group 2

Figure 5.23: Order 150/$\lambda = 0.02$ - Control Law - Diagonal - Groups 1 and 2

from 65 to 80, especially in the region between 500 Hz and 650 Hz, and also at the controller modes at 1240 Hz and 1469 Hz. This change induces a large improvement in performance, as seen in Figure 5.22. As the control horizon is further increased from 80 to 100, the magnitude of changes in the control law is decreasing, and the corresponding change in performance is also becoming smaller.

In the second controller group (Figure 5.23), the horizons have been increased from 100 to 160. The control law stabilized in most frequency regions, although there continue to be some small changes in the previously mentioned regions. As expected, the minor changes in the control law result in a performance value that has virtually stabilized.

The third and final group of controllers is shown in Figure 5.24, in which we show four control laws ranging from horizons 160 to 350. The four curves essentially lie on top of each other, indicating that there is almost no change in the control law once the horizons have reached a value of 160. Hence, a horizon value of 160 is the stabilization point for this particular system.

To summarize, we have seen that the diagonal performance curve (Figure 5.22) stabilizes at horizons of 100, and that this was the direct result of the stabilization of the control law in the frequency domain. The control law also starts to stabilize at a horizon value of 100, but does not fully stabilize until the horizons are set to 160 or higher.

During the analysis of the numerical simulations, we developed a simple rule of thumb to select the control and prediction horizons. We suggested that they should be given values of two or three

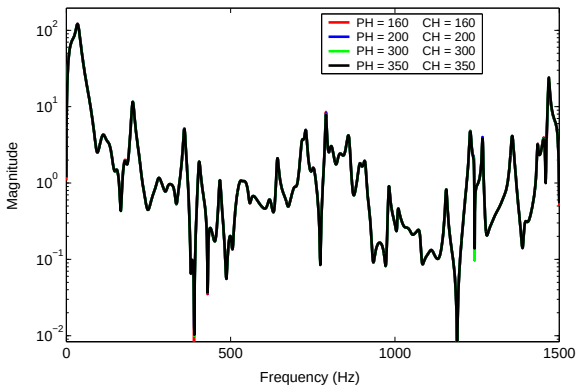


Figure 5.24: Order 150/$\lambda = 0.02$ - Control Law - Diagonal - Group 3

times the plant order, and that this would optimize the performance as well as stabilize the control law over a range of λ and orders.

We have estimated that the laboratory plate has between 30 to 35 modes in the 0 Hz to 1500 Hz band. In addition, the accelerometer signal conditioning equipment contains a low-pass filter that adds seven orders to the plant order. Therefore, the plant order in the laboratory is roughly 67 to 77. Hence the stability setting of 160 is roughly twice the overall plant order. Therefore, the rule of thumb developed in the numerical simulations also works in the laboratory setting.

Just as we have done with the numerical simulation, we also wish to examine the control law stabilization on an off-diagonal. The following figures will show the performance and control law for the line in which the prediction horizon is fixed to a value of PH=350.

The performance curve for this line is shown in Figure 5.25. In contrast to the diagonal performance curve (Figure 5.22), this curve is stable at even the lowest control horizon of 5. Just as with the diagonal performance curve, the performance improves rapidly as the control horizon is increased from its initial small value and starts to level off as it is increased to a value of 80 or larger. Beyond a control horizon of 80, the performance is relatively steady.

Once again we have split the control horizon into three groups. Groups 1 and 2 are shown in Figure 5.26, and Figure 5.27 shows the final group.

Group 1 (Figure 5.26) shows the control law magnitude as the control horizon is increased from 5 to 80. As the horizons are increased from 5 to 45, the control law undergoes major changes in magnitude throughout the entire bandwidth. This large change results in drastic improvement in performance as seen in Figure 5.25. The control law continues to undergo changes as the horizons

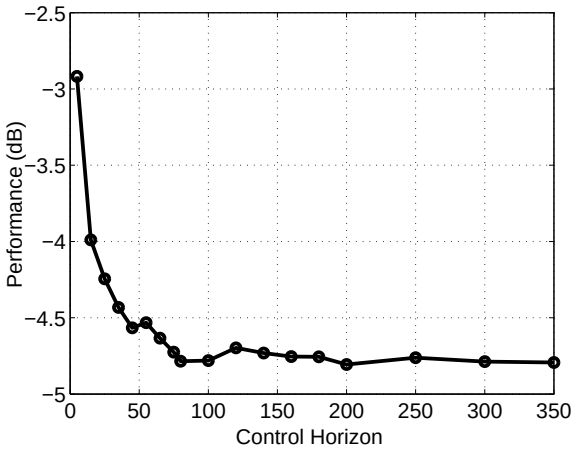


Figure 5.25: Order 150/$\lambda = 0.02$ - Performance along PH=350

are further increased to 80, but the size of the changes is decreasing and hence the corresponding improvement in performance is not as large.

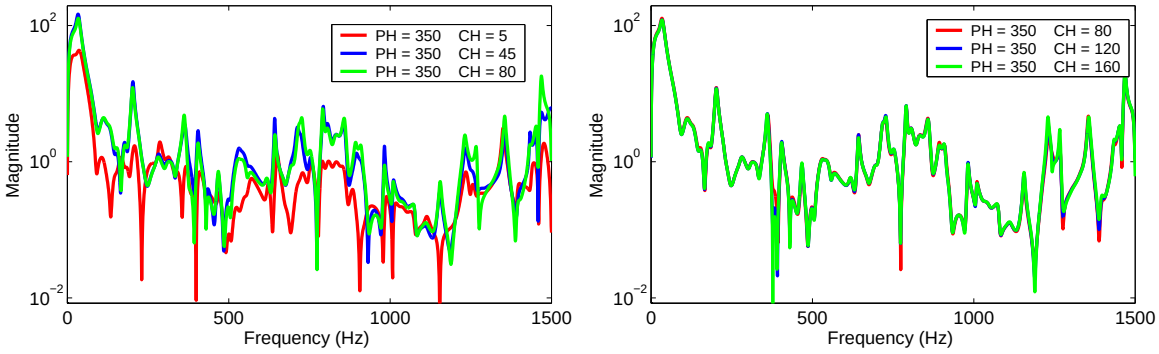
The transition of the control law from control horizons of 80 to 160 is shown in Figure 5.26 (Group 2). The control law has started to stabilize in all frequency regions, with some minor changes still occurring at the controller modes, e.g. 643 Hz, 858 Hz, 1231 Hz, 1267 Hz, 1356 Hz, and 1469 Hz. The corresponding performance has also largely stabilized.

In the final group, group 3 (Figure 5.27), the control law is shown for horizons of 160 and higher. The control law is now stable, with only very minor changes occurring at some controller modes (643 Hz, 1356 Hz). The stability point is the same as that seen on the diagonal, and from this point of view, there is no advantage in setting the prediction horizon to a constant value of 350. However, using a high constant value does improve the stability for very low values of control horizon. The disadvantage of the high value is that it increases the number of computations needed to calculate the control law.

5.1.2.2 Second Case - Order 400/$\lambda = 0.02$

For this experiment, the accelerometer was fixed to location 3, order was set to a fixed value of 400, and λ was set to a value of 0.02. Hence, this case is the same as the first experimental case, Section 5.1.2.1, except that order has been increased from 150 to 400.

The horizon map for this experiment is shown in Figure 5.28. This surface shows the exact



(a) Group 1

(b) Group 2

Figure 5.26: Order 150/$\lambda = 0.02$ - Control Law - PH =350 - Groups 1 and 2

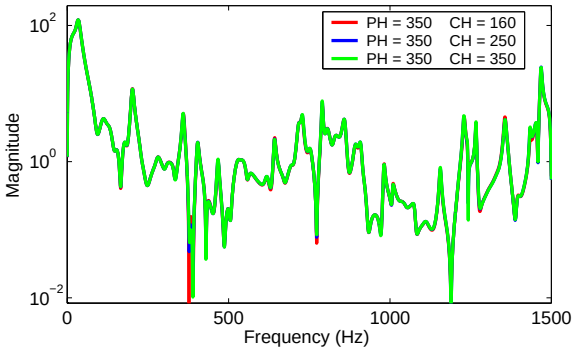


Figure 5.27: Order 150/$\lambda = 0.02$ - Control Law - Diagonal - Group 3

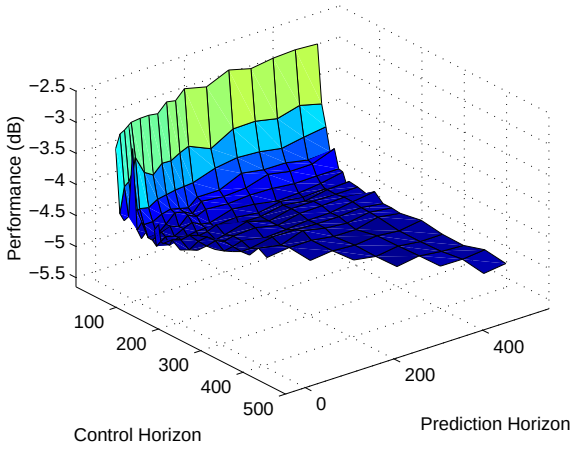


Figure 5.28: Order 400/$\lambda = 0.02$ - Experimental Horizon Map

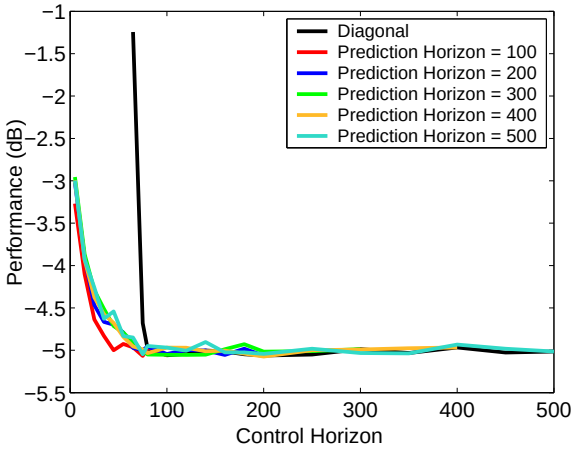


Figure 5.29: Order 400/$\lambda = 0.02$ - Constant Prediction Horizon Curves

same pattern as all the previous horizon maps, specifically that an increase in the control horizon from an initially small value leads to an improvement in performance. Furthermore, this effect disappears once the control horizon is increased beyond a critical value, and the performance surface becomes relatively flat.

The corresponding constant prediction horizon curves are shown in Figure 5.29, where this trend is confirmed. All curves except the diagonal are stable for the lowest control horizon, and stabilize at a control horizon of approximately 100 and a performance value of 5 dB. In comparison to the first case, described in Section 5.1.2.1, this global value of performance is a slight improvement over the 4.8 dB seen in that case.

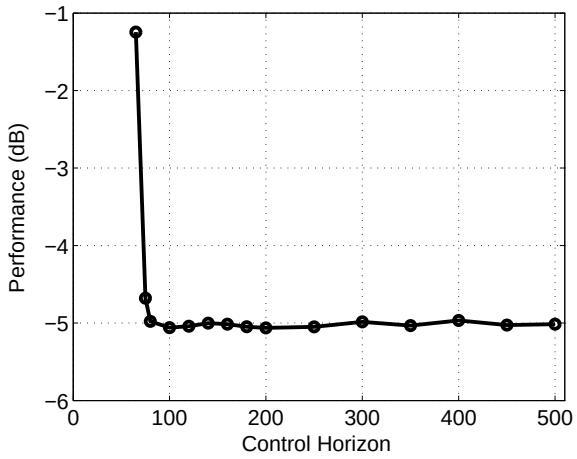


Figure 5.30: Order 400/$\lambda = 0.02$ - Performance along Diagonal

The performance along the diagonal is again shown in Figure 5.30. Just as for the first case, Section 5.1.2.1, the closed-loop system is unstable for all control horizon below 65. The performance quickly stabilizes as the control horizon is increased from 65 to 100.

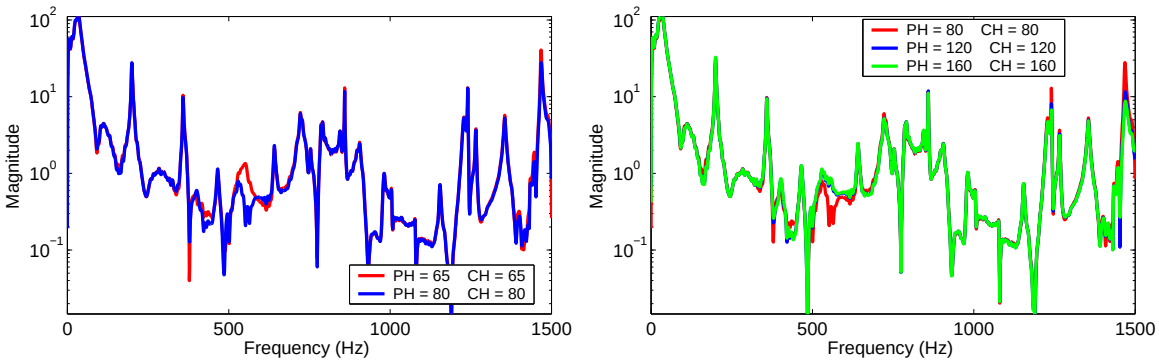
An examination of the control law in the frequency domain reveals the cause of the performance stabilizations. The control law along the diagonal is shown in Figures 5.31 and 5.32, and is split into three groups.

Group 1, Figure 5.31, shows the transition from horizons of 65 to 80, i.e. the change that corresponds to the large improvement in performance seen in Figure 5.30. Although the control law has stabilized in many frequency regions, there are significant changes taking place at some controller modes, including 35 Hz, 859Hz, and 1468 Hz, and in the frequency region between 500 Hz and 650 Hz .

From horizons 80 to 160, i.e. group 2 in Figure 5.31, the control law continues to undergo changes in some frequency regions, such as at modes at 35Hz, 1241 Hz, and 1468 Hz, and in the region between 500 Hz and 650 Hz. Clearly, the control law has not fully stabilized, however, these changes do little to change the closed-loop performance as seen in Figure 5.30.

Group 3, shown in Figure 5.32, shows the remaining controllers, i.e. those from horizons 160 to 500. The changes in the control are now minimal, and only occur at three controller modes: 859 Hz, 1241 Hz, and 1266 Hz.

Therefore, a horizon setting of 160 marks the point of stabilization in the frequency domain of



(a) Group 1

(b) Group 2

Figure 5.31: Order 400/$\lambda = 0.02$ - Control Law - Groups 1 and 2

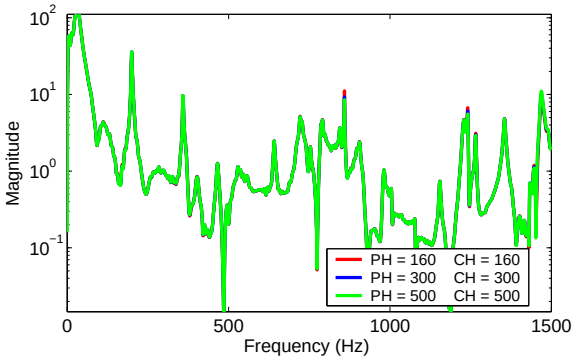


Figure 5.32: Order 400/ $\lambda = 0.02$ - Control Law - Group 3

the control law. This stabilization point is roughly twice the estimated plant order, i.e., the value predicted by the rule of thumb. This guideline worked in Section 5.1.2.1, and continues to work in this experiment in which order was increased from 150 to 400.

5.1.2.3 Third Case - Order 150/$\lambda = 1$

In this final case, we will show the results in which order is fixed to 150, λ is set to 1, and the acceleration is measured at point 3. These are the same settings as the first laboratory case, shown in Section 5.1.2.1, with an increase of λ from 0.02 to 1.

The resulting horizon map is shown in Figure 5.33, and the corresponding constant prediction horizon performance curves are shown in Figure 5.34. As expected, this surface shows the same

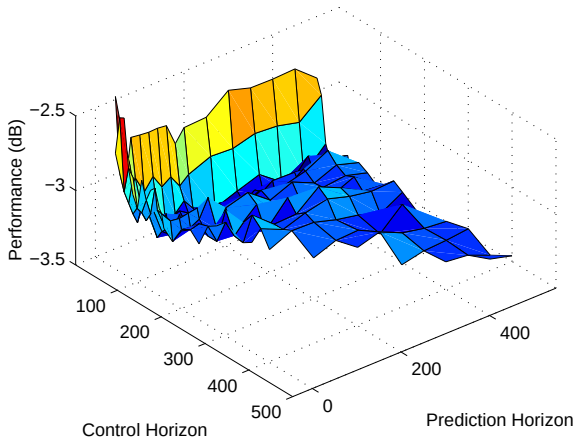


Figure 5.33: Order 150/ $\lambda = 1$- Experimental Horizon Map

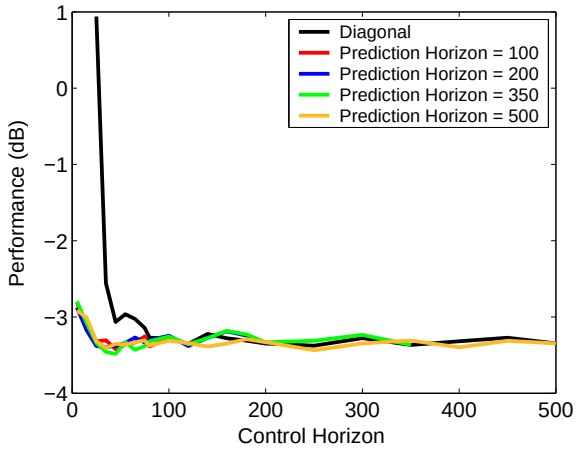


Figure 5.34: Order 150/ $\lambda = 1$- Constant Prediction Horizon Performance

trend as all of the other numerical and laboratory experiments: a performance surface that stabilizes once the control horizon is made large enough.

In Figure 5.34, all curves except the diagonal (PH=100, PH=200, PH=350, PH=500) are stable for the smallest control horizon, 5, and undergo a small improvement as the control horizon is increased. The performance stabilizes as the control horizon is increased to a value of 45. Larger values of the horizon result in some oscillation in the performance, which we believe is due to experimental variation. The overall level of reduction is between 3.2 and 3.4 dB. In comparison to the first laboratory case, described in Section 5.1.2.1, the $\lambda = 1$ setting provides approximately 1.4 dB less attenuation.

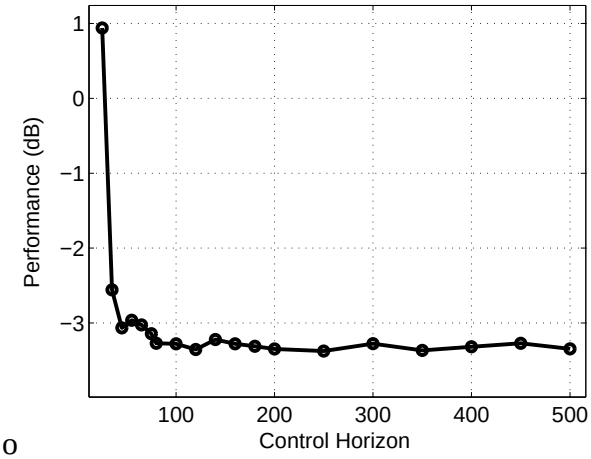


Figure 5.35: Order 150/ $\lambda = 1$ - Performance along Diagonal

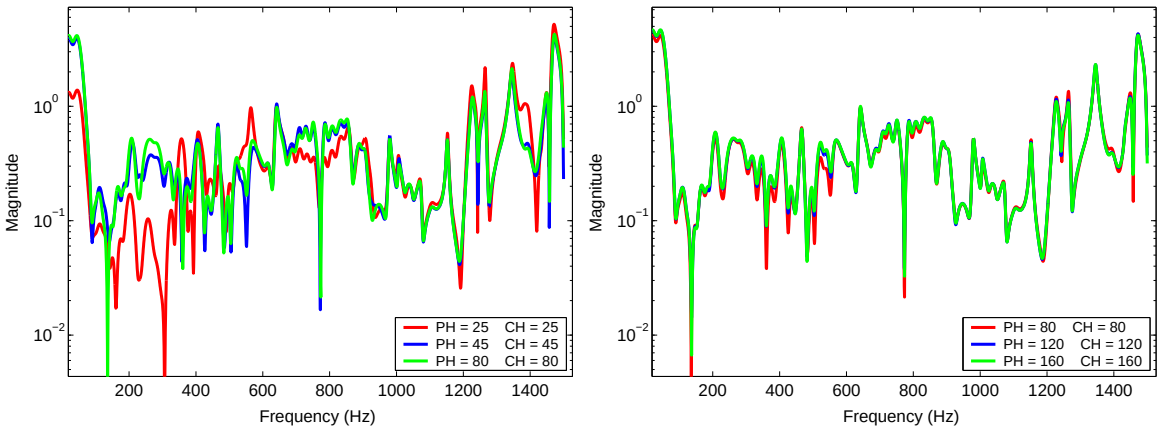
The performance curve for the diagonal is shown again in Figure 5.35. The first stable point (PH=CH=25) results in a 1 dB increase in the closed-loop system relative to the open-loop measurement. The next point (PH=CH=35) results in a 2.56 dB attenuation. As control is increased beyond a value of 100, the performance stabilizes.

As in previous examples, the performance curve stabilizes as a direct result of the control law stabilizing in the frequency domain. We have split the controllers into three groups, which are shown in Figures 5.36 and 5.37.

Group 1, shown in Figure 5.36, shows the transition for horizons 25 to 80, the region in which performance improves rapidly. The large-scale changes in the control law in the frequency, especially from horizons 25 to 45, result in the large improvement seen in the performance. As the horizons are further increased from 45 to 80, the pace of the changes in the control law decreases, but the control law has not stabilized.

In group 2, Figure 5.36, we show the evolution of the control law from horizons 80 to 160. There continue to be small changes throughout the entire frequency region, indicating that the control law has not yet stabilized. As seen in Figure 5.35, the changes are not large enough to significantly affect the performance, which remains relatively steady after horizons of 80 or 100.

The third group of controllers, Figure 5.37, shows the control law for horizons larger than 160. The three curves are nearly on top of each other, indicating that the control law has stabilized in all frequency regions. Therefore, setting the horizons to a value of 160 or larger guarantees a stable



(a) Group 1 (b) Group 2

Figure 5.36: Order 150/ $\lambda = 1$ - Control Law - Groups 1 and 2

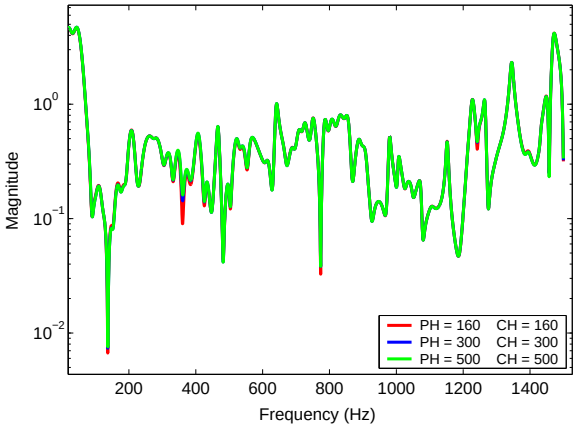


Figure 5.37: Order 150/ $\lambda = 1$ - Control Law - Group 3

control law for this particular system.

This stabilization point once again follows the simple rule of thumb developed in the numerical simulations. The numerical value of this point is the same as in Sections 5.1.2.1 and 5.1.2.2. Since each section featured a different combination of λ and order, the stability value is stationary for a range of λs and order. Hence, using the rule of thumb provides fixed values for horizons that in turn stabilize the control law and provide maximum disturbance rejection for any λ or order.

5.1.3 Setting the Horizons - Rule of Thumb

In conclusion, a rule of thumb was developed that provides a fixed value setting for both horizons. Specifically, it was shown that the horizons should be set to values of two or three times the estimated plant order. This setting was shown to stabilize the control law in the frequency domain, and to provide nearly optimal disturbance rejection. Furthermore, this setting is independent of λ and order.

We have therefore eliminated the horizons as a design variable for the control law, since fixed values will be used regardless of order or λ. The effect of these parameters on the closed-loop disturbance rejection is further developed in the next section.

5.2 λ and Order

In Section 5.1, we developed a simple rule of thumb for the selection of the control and prediction horizons. It was shown that the horizons can be set to static values that are independent of the controller order and control weighting λ. We have therefore eliminated two of the four design parameters of the generalized predictive control law.

The effect of the two remaining parameters, order and λ, are the focus of this section. Specifically, we will investigate how their variations impact not only the closed-loop performance, but also the maximum control effort generated by the controller.

In Chapter 6, we will develop rules for the fuzzy logic adaptation of these parameters. Two separate fuzzy logic systems are developed. The first system adjusts λ while the order remains fixed, and the second system adjusts order with λ fixed. Hence, what is needed is an understanding

of how variations in the adaptation parameters affect both the closed-loop performance and the control effort required to achieve this performance.

Our approach to this problem was similar to that of the previous section. For a given plant, an appropriate range of λs and controller orders is selected. Both ranges are subdivided into intervals, thereby creating a grid of desired points. Every grid-point represents a separate GPC control law. A closed-loop response for each control law is calculated or collected.[4] The resulting plate vibrations are used to calculate the performance of the controller. The closed-loop control signal is also saved and is used to compute the maximum control voltage.

After running closed-loop experiments for all of the controllers, we assembled two three-dimensional surfaces. The first is of the performance as a function of controller order and λ, and the second is of the corresponding maximum control voltages as a function of order and λ.

These two surfaces include everything that is needed in order to design the fuzzy logic systems. Specifically, constant order slices of the surfaces reveal the nature of the performance and control effort curves for a λ adaptation. Conversly, constant λ slices of the two surfaces show the performance and control effort curves for the order adaptations. These curves allowed us to answer basic questions such as:

- Does the performance always improve with decreases in λ? Is this improvement nearly constant or is there a point of diminishing returns?
- For decreases in λ, does the control effort always increase? Is the behavior linear, exponential or does it vary?

The process of examining these surfaces was repeated for numerous numerical simulations and laboratory experiments. Examination of the results indicated that the performance curves of all of these systems could be categorized into one of several behavior types. This knowledge of the various behavior types serves as the background for the design of the fuzzy logic system necessary to adapt λ and order. An overview of the observed behavior is shown in Section 5.2.3.

[4]For numerical simulations, the response of the plate is calculated using the *lsim* command in the same manner as for the adaptation experiments. Hence the process as described in Chapter 3 is also used here. In the laboratory, the response of the plate and control effort are collected using the DSP hardware. Again, this process is identical to that used in the adaptation experiments and is completely described in Chapter 4.

We have chosen to show the results of a pair of numerical and laboratory results, both of which have the accelerometer mounted to the third measurement location (see Figure 3.6). It will be shown that the trend for each result is similar, suggesting that the numerical model correctly captures the laboratory behavior.

5.2.1 Performance and Control Effort Simulations

The numerical simulation consists of a twenty-eight mode system whose action closely resembles the laboratory environment. We have calculated the performance and control effort surfaces as described in the introduction. Twenty thousand samples of closed-loop data were collected for each controller. The controller order was given values of 10 to 500, and λ was given values of 100 to 1×10^{-5}. Parameters that lead to closed-loop unstable systems are denoted by gaps in the surface. All controller orders less than 45 turned out to be unstable.

Figure 5.38 shows the resulting performance as a function of λ and the controller order. The figure shows that the performance is a strong function of λ. Reductions in λ lead to improvements in performance. The corresponding control effort is shown in Figure 5.39. Here we see that decreases in λ lead to increases in control effort.

We will now determine the trends in the data for constant order lines. In Figure 5.40, we show slices of the performance surface for various orders. In Figure 5.40, we clearly see that increases in order lead to improvements in performance. Note that the order (o) $o = 45$ line has a maximum performance of about 2.8 dB, but the $o = 300$ line has an additional 5 dB in performance.

Each performance curve shows a similar trend: each curve has a general downward trend between λs of 100 to 0.1 and then starts to flatten out for λs below 0.01. The slope change is more gradual for some of the curves, such as the $o = 60$ line, than for other lines ($o = 300,\ 120$). The $o = 60,\ 70$ and 90 lines stay stable until $\lambda = 1 \times 10^{-5}$, but the other curves become unstable near $\lambda = 1 \times 10^{-4}$. The reasons for these types of instabilities will be further explored in Section 5.3.

The corresponding slices of the maximum control effort surface are shown in Figure 5.41. All of the curves show an increase in control effort as λ is reduced. The control effort is roughly equal for all orders when λ is less than 0.01. Above this λ, the effort lines start to diverge. In one case,

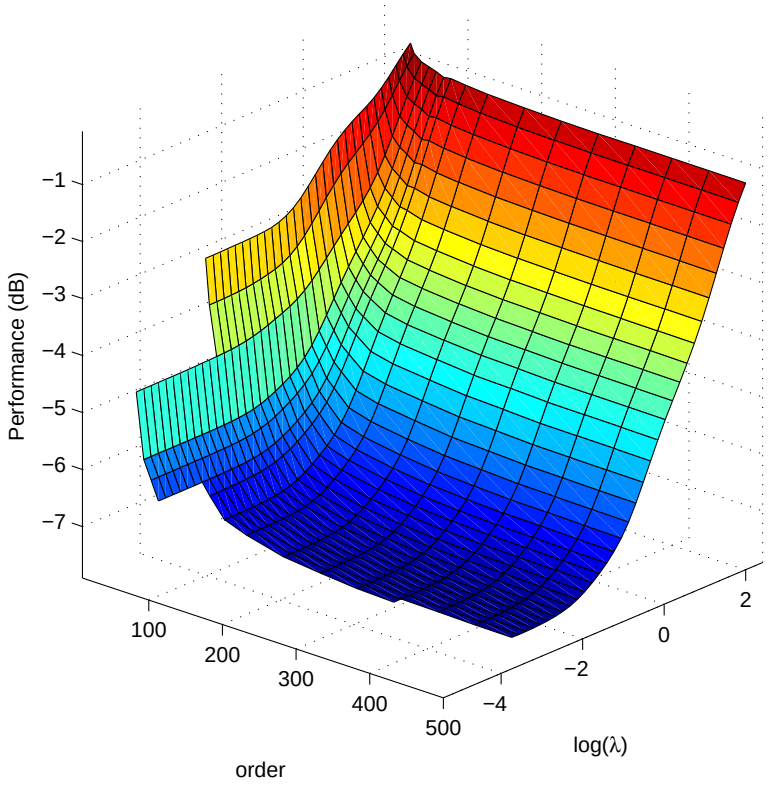


Figure 5.38: Performance Surface

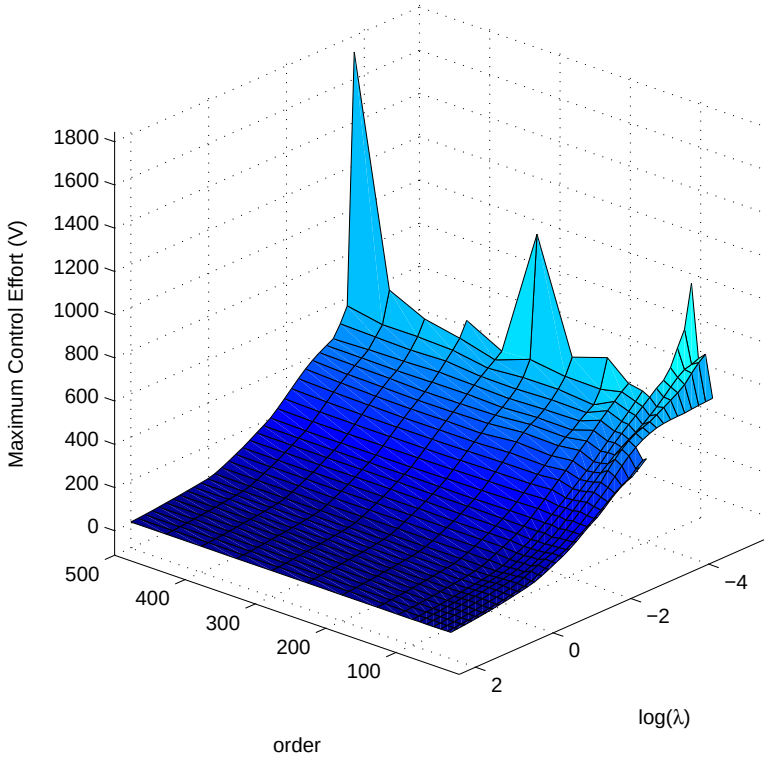


Figure 5.39: Control Effort Surface

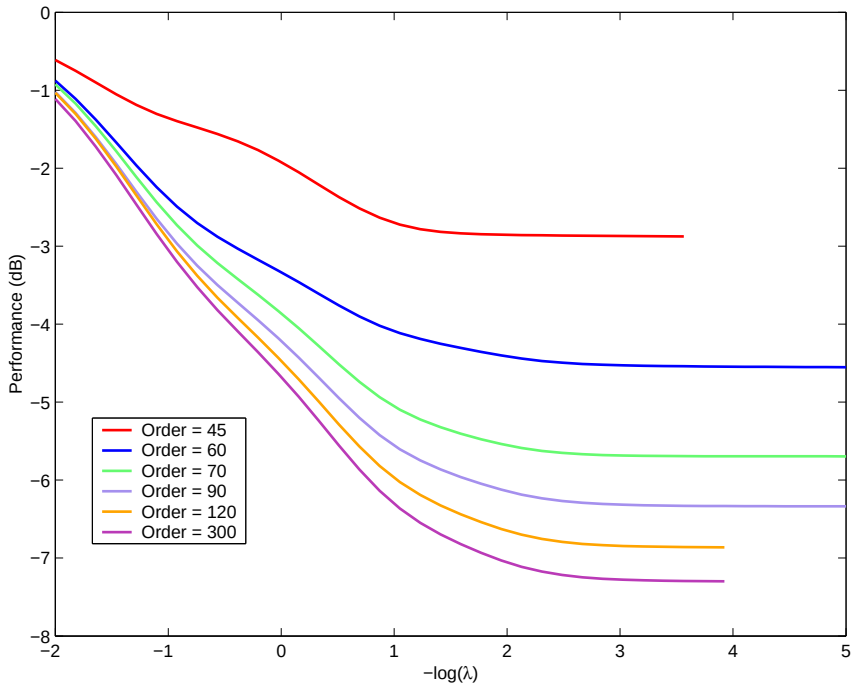


Figure 5.40: Constant Order Performances

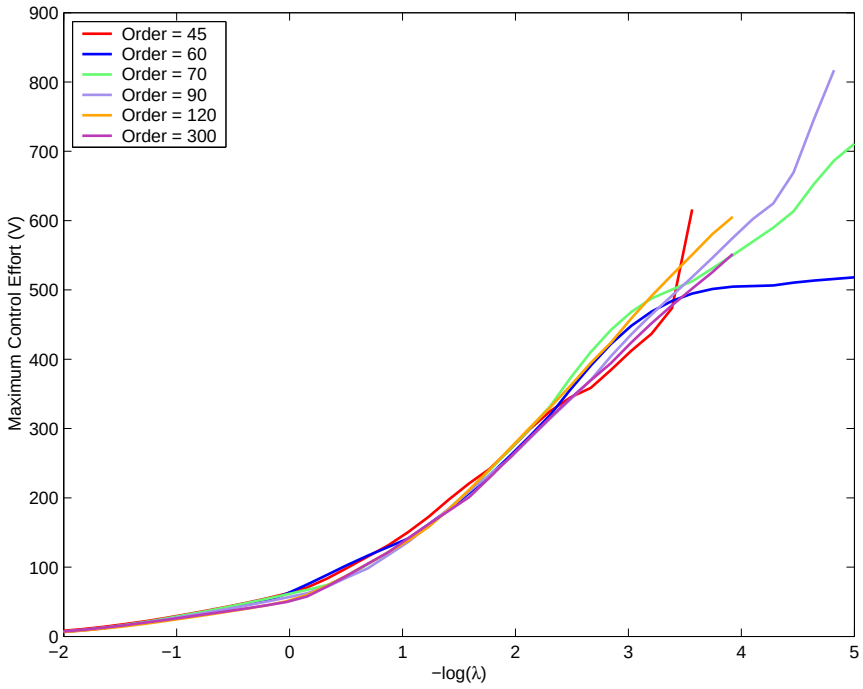


Figure 5.41: Constant Order Control Effort

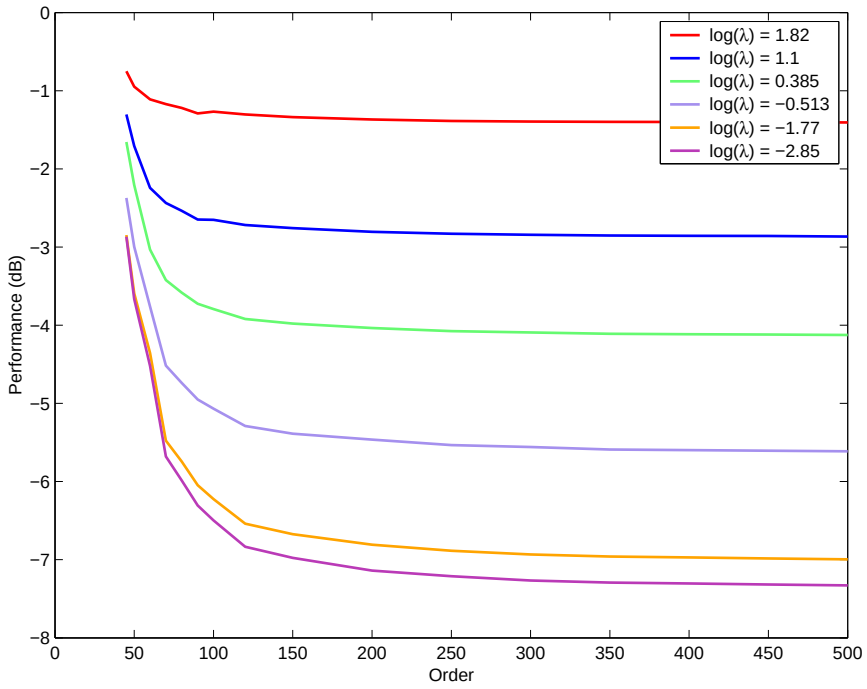


Figure 5.42: Constant λ Performances

$o = 60$, the control effort curve changes slope dramatically, but we still see an increase in control effort for decreases in λ. Note that the control effort continues to increase for λs below 0.01, even though the corresponding performance curves show little or no improvements for this decrease in λ. The increase in control effort therefore does little to improve performance.

We will now examine constant λ slices of these performance and control effort surfaces. The trends we will identify will be used to design the rules of the order adaptation. In Figure 5.42, we have shown the performance curves for several λs. Each curve has a significant improvement as order is increased from 45 to 100. As order is increased beyond 150, the curves flatten out and there is only a small improvement in performance. The initial improvement in performance is much larger for smaller λs, and the curves with smaller λs also show much better performance overall.

The corresponding maximum control effort curves are shown in Figure 5.43. All of the curves show a relatively constant level of control effort for changes in order. As expected, the lower λs use more control effort and also achieve better closed-loop performance (see Figure 5.42).

For the order adaptation, we will start off with a small order. For the sake of argument, let's

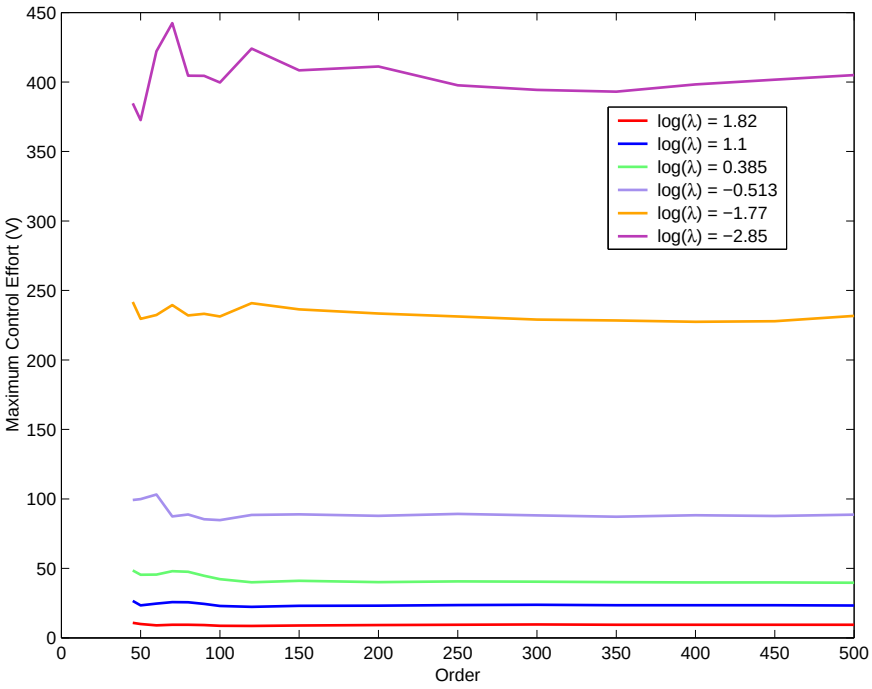


Figure 5.43: Constant λ Control Effort

assume that we start the adaptation with an initial order of 30 and a fixed λ of $\lambda = 1 \times 10^{-1.77}$. The initial iteration will be unstable. The adaptation algorithm needs to recognize that the small orders can be unstable and that order should subsequently be increased. If the order is then increased to 50, the closed-loop system will now be stable, but the performance is only 3dB. Further increases in order will lead to large improvements in performance, with very little change in control effort. Eventually, the order would become large enough so that further increases in order lead to little or no improvement in performance, e.g. once order reaches 150 or 200.

5.2.2 Performance and Control Effort Laboratory Experiments

This test was run with the accelerometer mounted to the third measurement location. The system is therefore comparable to the numerical simulation of Section 5.2.1. Running this type of experiment proved to be very difficult in the laboratory environment. The problem lies in the length of time it takes to complete the experiment, which in total was measured in hours. Over this length of time, temperature variations lead to small changes in the natural frequencies, which in turn cause the closed-loop to be unstable at points at which this shouldn't occur. Both the data collection process

and the control law calculation max out the cpu of the laboratory machine, and precious time was wasted waiting for a new control law to be calculated.

In order to reduce the length of the entire experiment, we devised a method in which we could use two computers to run the experiment. The computer in the laboratory controls the DSP hardware, collects closed-loop data, and calculates the performance and the maximum control effort. The second computer is only used to compute the system identification and each control law, with the necessary files sent back and forth via ftp. While the laboratory computer is collecting data, the second computer is calculating the next control law. Once the computation is complete, the second computer sends the results to the laboratory computer. After the laboratory computer collects a set of closed-loop data, it calculates the performance and control effort and then implements the next controller.

For this particular experiment, the system was identified in the beginning of the experiment. All of the controller designs were then based on this system identification. It took almost three hours to complete the experiment. Order was given 25 values between 30 and 500, and λ was assigned 20 values between 100 and 1×10^{-5}. This experiment therefore designed and implemented 500 different controllers.

The resulting performance surface is shown in Figure 5.44, while the corresponding maximum control surface is shown in Figure 5.45.

We will now determine the trends of the constant order slices of the performance and control effort surfaces. In Figure 5.46, we have shown the constant order slices of the performance surface. All of the performance curves have a near-constant downward slope for λs above 1. As λ drops below 1, the curves separate. The performance curves show one of three trends.

The first trend is demonstrated by the $o = 50$ curve. This curve shows a near-constant drop and then becomes unstable very early at a λ of 1. The second trend is shown by the $o = 70$ line. This curve has a local minimum at a λ of about $1 \times 10^{-2.5}$. The third trend is represented by the $o = 120$,180, and 300 curves. Here, the slope of the curves change gradually and the curves start to level off. These curves are very similar to those of the numerical simulation (see Figure 5.40). It is not possible to directly compare the λ values of the numerical simulation and laboratory experiment since the units of the inputs and outputs are different. In the laboratory experiments, the controller was based on voltage units because this is the information collected by the digital

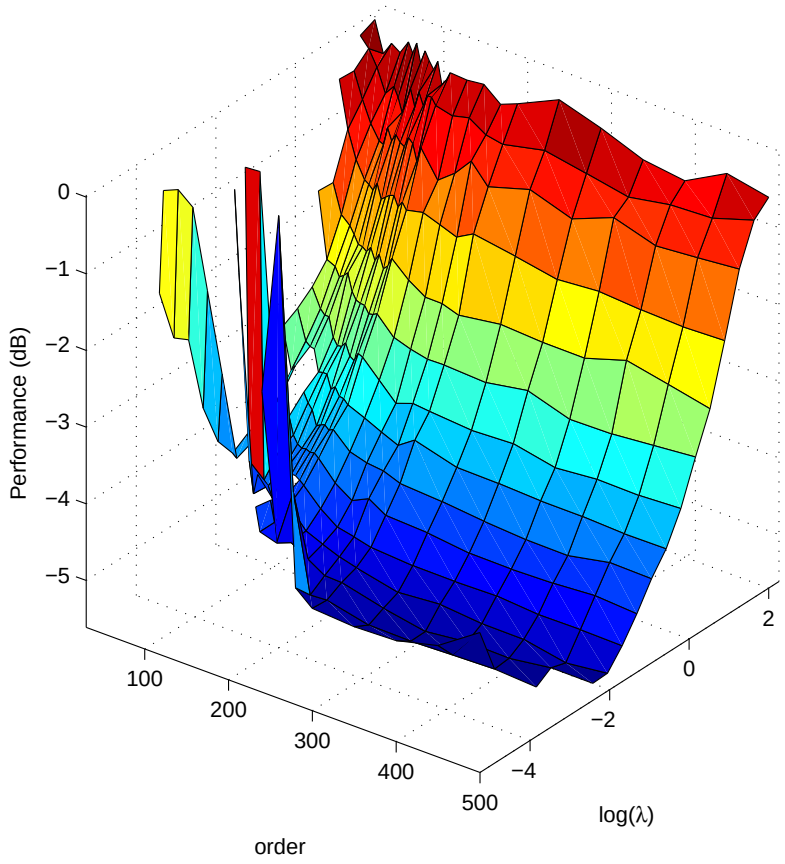


Figure 5.44: Performance Surface

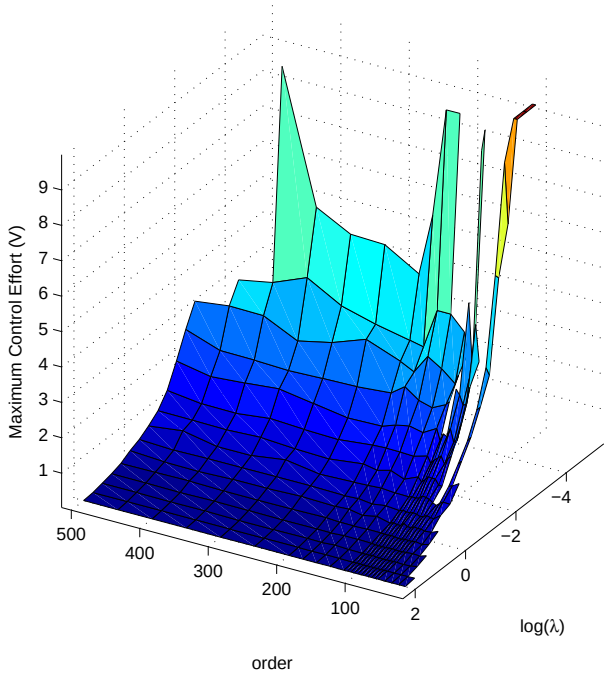


Figure 5.45: Maximum Control Effort Surface

signal processing board. The input data is acceleration in volts, and the output of the DSP is the voltage sent to the amplifier connected to the piezoelectric patch. In the numerical experiment, the input is the acceleration in m/s^2 and the output is the actual voltage applied to the piezoelectric patch.

The corresponding slices of control effort surface are shown in Figure 5.47. The effort is shown as voltage output of the DSP board. This signal is sent to the piezoelectric amplifier that multiplies the signal by about 6.5. Each curve shows an increase in control effort for decreases in λ. The control effort levels are nearly identical for λs less than 0.01, but start to slightly diverge for λs larger than this value. This behavior is identical to that seen in the numerical simulation (see Figure 5.41)

We will now determine the trends for the constant λ curves. The performances for constant λ are shown in Figure 5.48. Each curve starts with an instability for very low orders. Increasing the order stabilizes the system and gives us some closed-loop performance with the exception of the $\lambda = 1 \times 10^{-2.42}$ line. Further increases in order result in a performance gain. Eventually, each of the curves levels off, and there is no performance benefit for increasing order. Again, this trend is

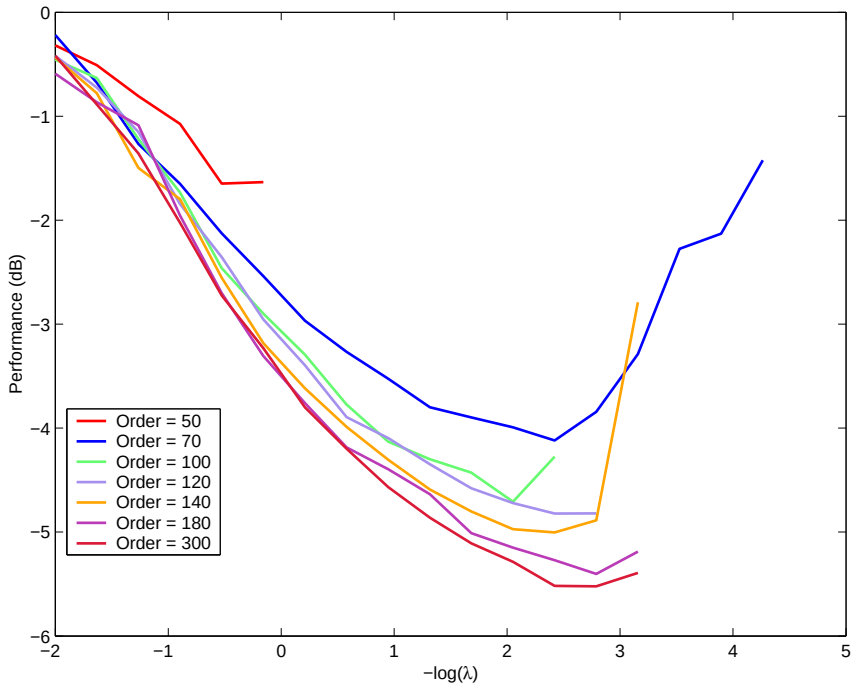


Figure 5.46: Constant Order Performance

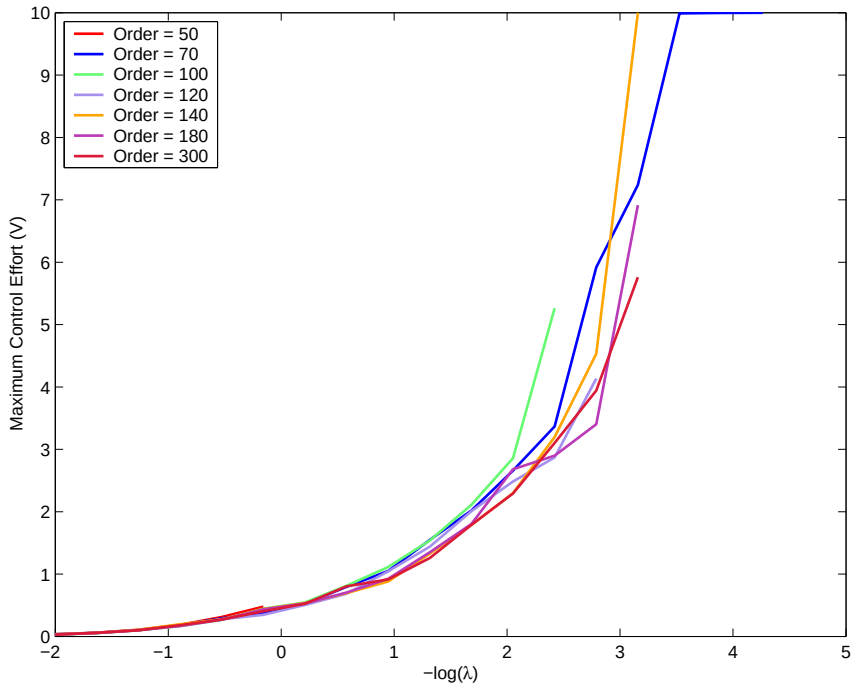


Figure 5.47: Constant Order Maximum Control Effort

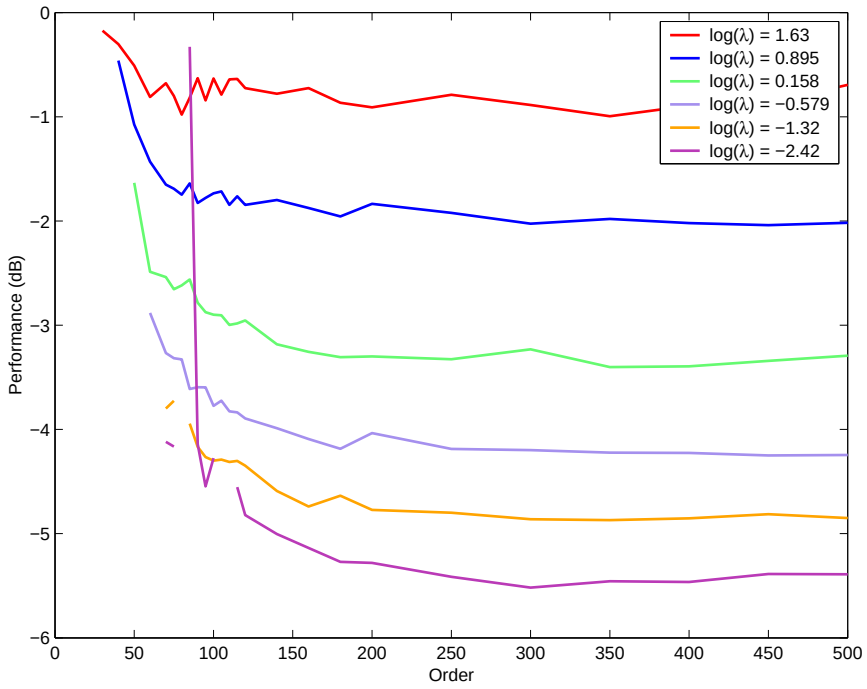


Figure 5.48: Constant λ Performance

exactly what we saw in the numerical simulation.

The control effort for these constant λ lines is shown in Figure 5.49. Each curve is relatively flat; therefore, increasing the order does not result in an increase in control effort.

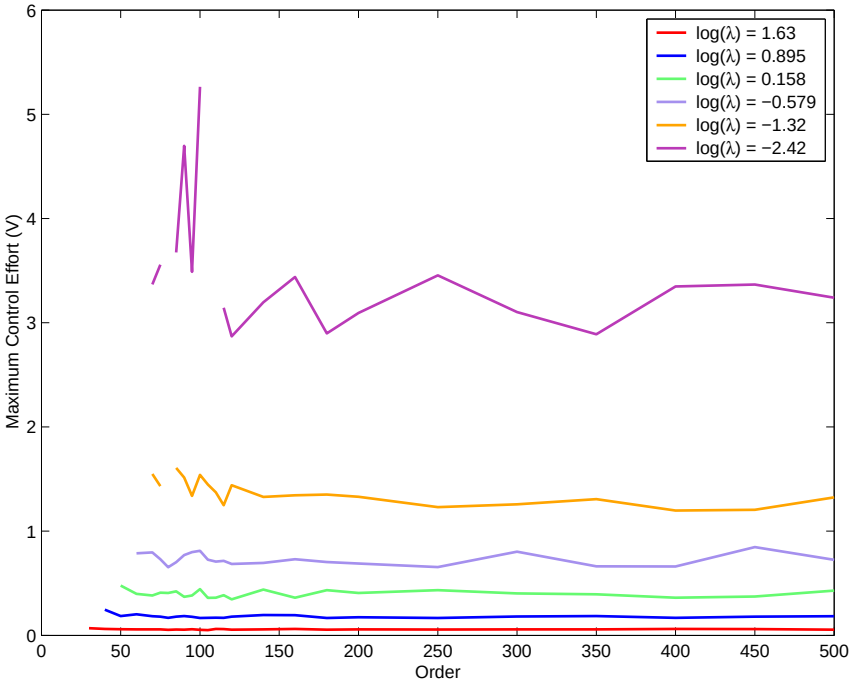


Figure 5.49: Constant λ Maximum Control Effort

5.2.3 Overview of Observed Behavior

In Sections 5.2.1 and 5.2.2, we have shown performance and control effort curves for both constant λ and constant order for a numerical and laboratory system. In these examples, we saw that the overall shape of these curves could be categorized into one of several types. However, the examples did not show all of the behavior types observed in the various numerical and laboratory experiments. In this section we will show a complete set of categories for both the constant order (λ adaptation) and constant λ (order adaptation) curves.

We will show that different types of curves will create certain needs that must be addressed by the fuzzy logic adaptations. Specifically, the curves will generate the need for monitoring the closed-loop stability, estimating the slope of the performance curve and its relationship to past values, and monitoring the control effort in relation to the maximum allowable level. These needs are present in both the λ and order adaptations.

For each adaptation type, we have separated the performance curve types from the control effort types. This was done in recognition of the fact that for any given system, a change in the fixed parameter might lead to the performance curve type that is nearly identical, but the control effort curve is dramatically different. An illustration of this phenomena follows the summary for constant order performance and control effort curves.

5.2.3.1 Constant Order - Performance and Control Effort

In this section, we will summarize the types of performance and control effort curves that have been observed when order is held constant and λ is varied. This knowledge therefore serves as the background of the fuzzy logic adaptations of λ (constant order) developed in Section 6.1.

We will describe the curves with respect to an initial λ that is large, and then discuss changes in the curve as λ is decreased. This direction was chosen because it virtually ensures that performance improves in the short term. Just as in Sections 5.2.1 and 5.2.2, the sample performance and control effort curves shown below are functions of $-log_{10}(\lambda)$. Therefore, a decrease in λ will result in moving the operation point farther to the right on the curve.

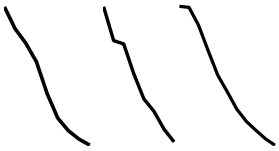
With respect to stability, all systems with sufficient order proved to be stable for very high λs. This is due to the fact that the plate system is a naturally stable system, and that controllers with

a high λ use very little control effort. Hence the controller is not moving the system poles very far, and the closed-loop system remains stable. Once λ becomes low enough, all systems in the laboratory and in the numerical simulation become unstable. The cause of these instabilities is discussed in Section 5.3.

The fuzzy logic adaptation process will start with a very high λ and with further iterations reducing λ. The operating point therefore moves in the same direction in which we described the performance curves, starting on the left side of the curves and then moving to the right. These reductions occur in a series of step changes in λ.

The step size is dynamically altered by a fuzzy logic system to account for conditions at the current operating point and in anticipation of future operating conditions. The adaptation is stopped, i.e. λ remains unchanged, for sets of circumstances further discussed below. The goal of the adaptation process is to optimize performance while keeping the control effort within the allowable bounds set by the user.

Performance Curve Types

1. 

   This type of curve features a steep drop that indicates that further iterations would improve the performance drastically. As λ is reduced, the slope of the curve remains relatively constant. Upon further decreases in λ, the closed-loop system reaches a sudden instability.

2. 

   This type of curve also features a steep drop that indicates large improvements in performance for reductions in λ. The curve then reaches a transition region in which the slope of the curve gradually changes so that the resulting curve is flatter. It is important to note that the seemingly flat region does not necessarily have a slope of zero. In some systems, there is still a small downward slope, i.e. performance improves marginally for decreases in λ, while other systems show no improvement in performance.

3. 

   The first half of these performance curves are similar to type 2 curves. As λ is decreased from its initial high value, the performance improves rapidly. Once a certain point is reached, the slope changes dramatically to become zero. Further reductions in λ lead to a positive slope in the performance curve, i.e. performance degrades for decreases in λ. The changes in slope establish a global minimum at a specific λ.

Not all systems feature the range of performance curve types. For example, the system in Section 5.2.1 does not have type 3 curves. Some configurations feature only one type of performance curve, while others have a variety.

In general, type 2 curves most often occur when the order is much larger than the true system order. Type 3 curves usually occur when order is below or equal to the true system order. Type 1 curves have been observed at all orders. It is not possible to say that any given system will have all of the type of performance curves mentioned, or that a particular order will result in a certain type of performance curve. Such statements can only be made after the fact.

The characteristics of these curves are important in the development of the fuzzy logic rules, which will need to successfully deal with each type of performance curve. To this end, it becomes important to characterize the various types of curves with respect to the desired adaptation process.

The first performance curve type,  , is different from the other two types in that it features no significant change in slope throughout the curve. If an adaptation is performed on this type of system, λ is reduced in a series of steps so that the operating point gradually moves from the left to the right. At some point, a new λ will be implemented that causes the closed-loop system to become suddenly unstable.

A fuzzy logic module was developed to recognize and correct this highly undesirable condition, as discussed in Section 6.1.1.1. When a system instability is detected [5], this module will increase λ to a previously stable level. If this new λ results in a stable system, then the adaptation is halted.

The second performance curve type () features a significant change in slope as λ is reduced. An example of this type of curve is shown in Figure 5.50. We have split the curve into three regions each of which represent a different slope condition.

[5] The methodology for detecting the closed-loop instabilities is further discussed in Section 6.1.1.1.

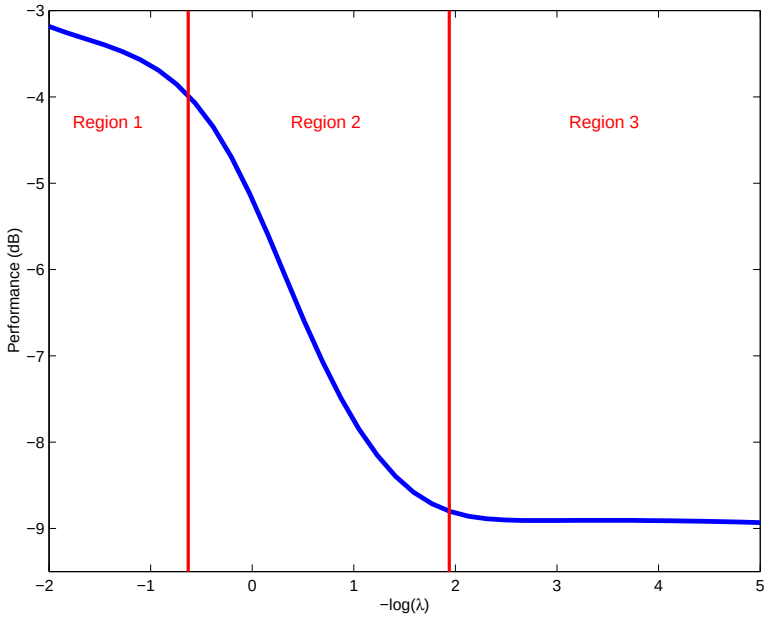


Figure 5.50: Example of a Type 2 Performance Curve

The adaptation will start in region one and then proceed to regions two and three. Once the current operating point reaches region two, we see the steepest drop in the curve and therefore the largest gains in performance for changes in λ. As the adaptation continues, we see the operating point moving into region three, in which the slope magnitude is relatively small. Hence, in this third region, we see, at best, marginal improvement in performance in comparison to that seen in region two. At this point, further reductions in λ do very little to improve the overall performance, which was very nearly optimized by traveling through region two.

In summary, the significant changes in slope result in an operating point in region three in which the performance has been very nearly optimized, and further reductions in λ seem unnecessary. At this point, it may be beneficial to stop the adaptation, especially once the rise in control effort is considered. Clearly, the fuzzy logic adaptation must be aware of this type of curve, and therefore the slope must be estimated in some way throughout the adaptation.

Not only must the fuzzy logic be aware of the slope at the operating point, but it must also recognize the relation of this slope to that seen in the performance curve up this point (left of the operating point). The transition from regions two to three is marked by a large decrease in the magnitude of the slope, and it is this decrease that becomes the indicator that will be used by the fuzzy logic rules to control the adaptation rate.

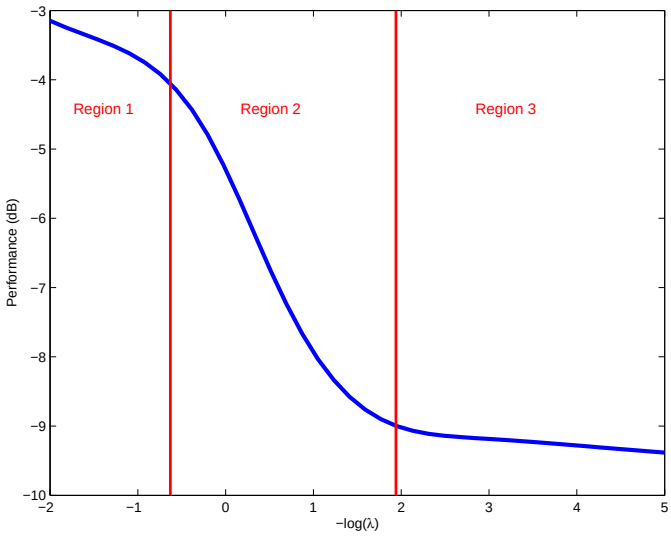


Figure 5.51: Type 2 Performance Curve - Some Improvements in Region 3

A fuzzy logic module was designed to look for curves of this type and is further explained in Section 6.1.1.2. The adaptation rate is adjusted according to which region of the curve the operating point is in. When the slope at the operating point is large (regions one and two), this module will set the adaptation rate at its maximum, thereby allowing large step changes in λ. Once the transition from region two to three is detected, the adaptation rate is gradually reduced. As the operating point enters the third region, the module will stop the adaptation.

As we have already noted, the exact nature of the slope in region three varies. In Figure 5.50 we showed a system in which the slope was almost zero. Others systems, such as that shown in Figure 5.51, have very small but non-zero slopes in region three, and hence an improvement in performance for further reductions in λ is possible. Hence, if the control effort is low enough, there is no reason not to allow a small reduction in λ. This exception was built into the fuzzy logic system and is further described in Section 6.1.1.4.

The third performance curve also features large changes in slope that establish a global minimum. The changes in slope for the region to the left of the minimum are identical to those seen for the second performance curve type. These curves also feature a region in which the slope is the steepest (similar to region two of type 2), and then a transition region in which the slope magnitude decreases to zero. This transition is identical to that seen for type two 2 from regions two to three. Hence, the fuzzy logic module that was designed for type 2 curves will also work for the third type,

and the adaptation will stop near the global minimum.

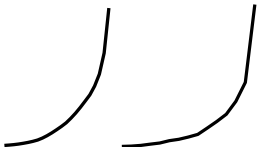
We have shown that the three types of performance curves give rise to two issues that must be identified by the fuzzy logic rules. The first is the occurrence of closed-loop instabilities. The fuzzy logic adaptation must recognize closed-loop instabilities and then increase λ to return the closed-loop system to a stable operating region.

The second issue is the dramatic change in slope seen in types 2 and 3. The changes in slope result in an operating condition in which the performance cannot be improved, or can only be improved marginally. This establishes the need for a fuzzy logic system that monitors and analyzes the slope at the current operating point and adjusts the adaptation rate accordingly.

Maximum Control Effort Curves

Since λ is the control effort weighting in the cost function (Equation (2.16)), we naturally expected the control effort to increase with decreases in λ. This was shown to be true in all systems observed, with variations in the exact shape of the curve. We have categorized these shapes into four arbitrary groups below. As previously mentioned, we are interested in the maximum control voltage exerted over a given interval of time, and the curves shown below reflect this fact.

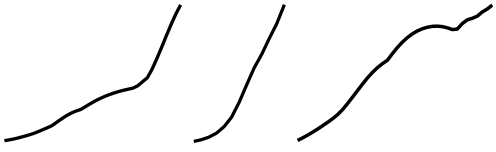
For reductions in λ from an initial high value, we see maximum control effort values that:

1. 

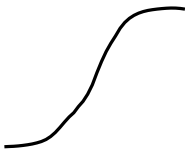
   Continually increase, with the control effort rising exponentially for decreases in λ

2. 

   Continually increase, with the control effort varying nearly linearly as a function of $log_{10}(\lambda)$

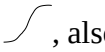
3. 

   Continually increase, but the curve does not fit into one of the first two categories. This is the most common type of curve.

4. 

   Increase until some critical λ, after which the control effort remains relatively constant. This type of behavior is only seen in rare cases.

The first three types all have one characteristic in common: the control effort increases as λ is reduced. Obviously, the only difference is the overall shape of the curves, or put more specifically, the changes in slope as λ is reduced. In the first type,  , the slope continually increases so that decreases in λ will result in large changes in control effort. For the second type,  , the slope is relatively constant, and for the third type,  , the slope changes in unpredictable ways.

The fourth type of curve,  , also shows an increase in control effort for decreases in λ so that any adaptation of λ will increase control effort relative to its initial value. After reaching a critical value of λ, the control effort levels off so that further adaptation would not incur the penalty of an increase in control effort. Up to the critical value of λ, this type of curve is indistinguishable from types 1,2, and 3.

The significance of these curves is that as λ is reduced in order to improve performance, the control effort will invariably rise. We wish, however, to keep the control effort below a level set by the user, which we refer to as the *maximum allowable control effort*. As the adaptation progresses, the control effort will rise toward this ceiling for all four types of curves, thereby creating a need for a fuzzy logic module that regulates the adaptation rate based purely on control effort. Depending on the specific value of maximum allowable control effort, it is conceivable that the current operating point requires much more control effort than the user would like. We would like to avoid this condition as much as possible.

Obviously, for each system there is a specific λ at which the control effort is nearly equal to the maximum allowable level. The problem is that this critical value of λ is not know in advance. It may be possible to predict this value, but the large difference in the categories of curves show that this would be very difficult. If this prediction were possible, then a fuzzy logic module could be designed that allows a full adaptation rate up to this point, and then abruptly stops the adaptation once it is reached.

However, the unpredictable nature of the curves dictates that we must take a more conservative

approach. Instead of stopping the adaptation abruptly, the fuzzy logic module will gradually slow the adaptation rate as λ reaches the critical value. The module will therefore act as a brake, where the ratio of the control effort relative to the maximum allowable level becomes the indicator for changes in adaptation rate. Hence, as the control effort reaches a level close to the maximum allowable, the module must restrict the adaptation rate so that only small steps in λ are allowed. Contrarily, the fuzzy logic module should allow for a fast adaptation rate (large step changes in λ) when the effort is low in relation to the maximum allowable. The primary goal of the fuzzy logic module is to stop the adaptation before the control effort exceeds the maximum allowable.

This fuzzy logic module, named the U_{max} system, uses these principles to regulate the control effort, and is described in Section 6.1.1.3. The ratio of the current maximum control to the maximum allowable is the only input. When the adaptation is started (operating point on the far left of the curve), the control effort is usually low relative to the maximum allowable level, resulting in a low input ratio. Since the current operating point is far away from the region in which the effort exceeds the maximum allowable, the module will allow the maximum step size. As λ is reduced, the control effort, and therefore the input ratio, will rise, indicating that the current controller is now closer to using the maximum available control effort. The module will start to decrease the step-size in anticipation of a future operating point that will use too much control effort.

Furthermore, we have built into the U_{max} module two critical regions. The first region stops the adaptation once the control effort reaches a value of 70% to 87% of the maximum allowable. The second region occurs when the control effort exceeds 87% of the maximum allowable. If this occurs, the direction of adaptation is reversed, i.e. λ is increased, so that a future operating point falls into the stopping regions.

This second region was needed due to the fact that even a small step change can result in a dramatic increase in control effort. This is especially true when the control effort curve fits into category one, $\smile$. Early experimentation showed that it was entirely possible that the operating point misses the stoppage condition completely, i.e. going from 65% to 95% in one iteration. Even worse, some systems overshot the maximum allowable control effort limit. This was also why the stopping region was set below the maximum allowable region instead of being right up against the limit (85%-100%, for example).

We have shown the reasons for the development of three fuzzy logic modules: the stability

module, the derivative module, and the U_{max} systems. As has been stated, the U_{max} system works independently from the other two modules. A fourth module, named the *metarules*, combines the output of these modules and determines on the final adaptation rate, and is further discussed in Section 6.1.1.4.

5.2.3.2 Constant λ - Performance and Control Effort

This section serves as a summary of the types of performance and control effort curves that are seen when λ is held constant and the controller order is allowed to vary. This information therefore serves as a background for the fuzzy logic adaptations of order described in Section 6.2.

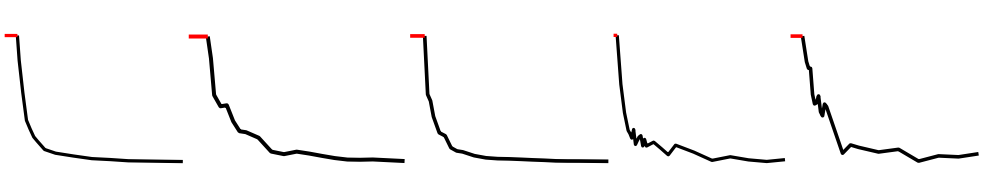
These curves are described with respect to increases in order from an initially low order. This is also the chosen adaptation direction, i.e. order is increased as the iterations go on, and the operating point moves from the left to the right on the curves.

The curves are shown without axes just as in the previous section, since the overall shape of the curve is more significant than the actual values of performance or control effort. One major difference to the curves in the previous section is with respect to stability. In all systems in which the initial order is very small (in relation to the plant order), the closed-loop systems are not stable, and the order must be increased in order to restore stability. This initial instability is shown as a thin red line on the curves below. Some systems will also become unstable as order is increased beyond a critical value[6].

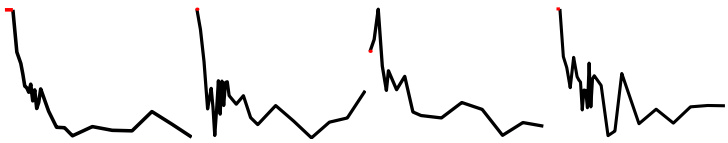
Just as was the case for the λ adaptations, the fuzzy logic modules determine the size of the step changes in order. The following discussion will show that the curves have similar characteristics to those shown for the constant λ curves. This makes it possible to adapt the fuzzy logic modules designed for those adaptations for use with the order adaptations.

Performance Curve Types

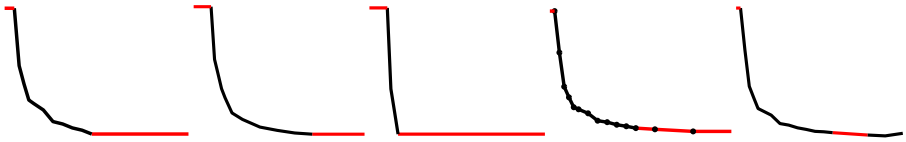
1.



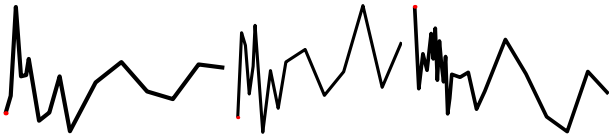
[6]Only points that are known to be unstable are marked in red. Since the minimum order is well above zero, there are still points below those tested that are most likely to be unstable as well.



This type of curve features a steep drop in the initial part of the curve. As order is increased, the curve flattens out, and performance gains become marginal. These types of curves do not become unstable once the order reaches a critical value. In the laboratory experiments, the curves can become somewhat erratic, as shown in the last three pictures.

2. 

These curves also feature a steep drop in the initial part of the curve. As order is increased beyond a critical value, the closed-loop system loses stability. Some systems may have a few stable points inside this second unstable region, but this rare.

3. 

These curves are very erratic and were only seen in experimental data at very high λs. Since these curves only occur at very high λs, their corresponding performances are very low. For this reason, adaptations were not pursued on these type of curves.

A vast majority of the curves seen in the numerical simulation and in the laboratory experiments are of type 1. In the laboratory, the curves are not always as clean as in the numerical simulation and sometimes have a ragged appearance  . Curves of type 2 occur when λ is near the stability boundary and were seen only in rare cases. Type 3 curves were only seen in the laboratory experiments at extremely high λs and have very poor performances overall (less than 2 dB). Once λ was reduced for these systems, the performance curve would once again follow type 1 behavior.

Clearly, type 1 (constant λ) curves have the same basic shape as the type 2 constant order curves described previously. Hence, all of the discussion attributed to the type 2 constant order curves is also relevant to the type 1 constant λ curves, with an increase in order being equivalent to a drop in λ.

Specifically, the curve undergoes a dramatic change in slope as order is increased so that later iterations (higher order) see very little performance gain in comparison to earlier iterations in which order was lower. Once the operating point moves into this region, it is no longer beneficial to further increase the order.

This dramatic change in slope creates the need, just as did it for the λ adaptations, to monitor the current slope as well as its relationship to past values. Since the mechanism is essentially the same as in the λ adaptations, the fuzzy logic module designed for the λ adaptations could be adapted for use with the order adaptations. The resulting fuzzy logic module is described in Section 6.2.1.2.

As was already stated, the expected stable region in the order adaptations is entirely different than in the λ adaptations, and therefore the stability module must be modified to work with the order adaptations. In many cases, the order adaptations will start with an initial value of order that is smaller to the plant order, and the resulting closed-loop system is often not stable. Since the order controls not only the controller order, but also the order of the ARX equation (Equation (2.11)), an order that is well below the plant order will attempt to identify an insufficient number of modes, thereby contributing to the loss of stability.

In order to stabilize the system, the order must be increased. Once the order becomes sufficiently large with respect to the plant order, the closed-loop system will stay stable for the entire range of orders if the system exhibits type 1 behavior ,  .

For type 2 systems,  , the closed-loop system will return to instability if order is increased beyond a critical value. In some very rare cases, the system may become haphazardly stable if order is increased further, but in most cases, it will remain unstable. In order the stabilize the closed-loop system, the order must be decreased to a previously stable value.

Hence, there are two regions in which we can expect instability: when order is very low (this is almost guaranteed), and when the order is very high (less likely). The stability module was redesigned around this observation and is further described in Section 6.2.1.1.

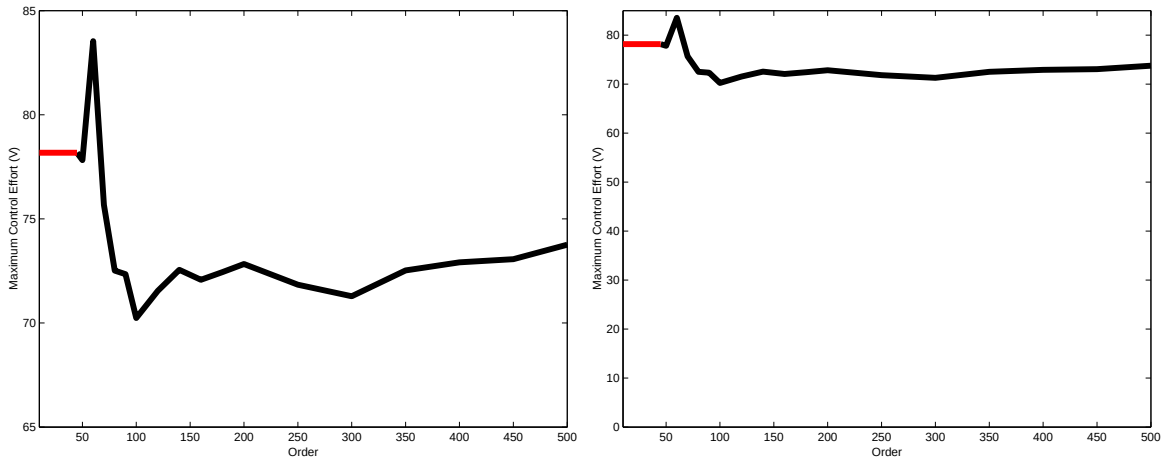


Figure 5.52: Effect of Y-Axis Scale on Perceived Shape

Control Effort Curve Types

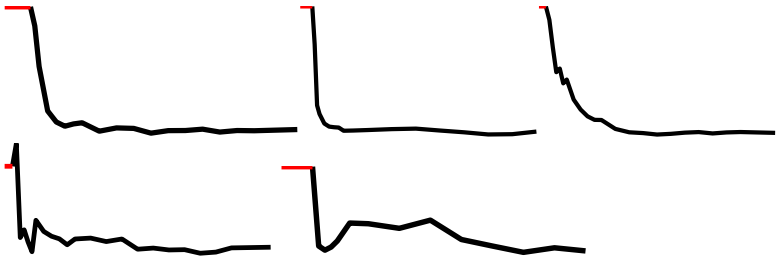
The three main control effort types are shown below. Unlike the λ adaptations, there is no expectation of an increase in control effort with an increase in order. One significant difference with the control effort types of the constant order (λ adaptations) is that the curves do not start off at or near 0V. The overall level of the effort is determined by the λ chosen, where lower λs create higher levels. Examples of this behavior are shown in Section 5.2.

Since the curves do not start off at 0V, the scale of y-axis of the plots dramatically changes how we might view the shape of the curve. An example of how the y-axis range affects the perceived shape of the curve is shown in Figure 5.52. In the left graphic of this figure, we see the y-axis zoomed in on the data, resulting in a y-axis range of 65V to 85V. Here, the control effort curves appears to undergo a dramatic drop between orders of 50 to 100, with a near-constant level with order over 100. When the range of the plot is adjusted to include 0V to 85V, the result is the rightmost graphic in Figure 5.52. At this scale, the seemingly large drop seen in the leftmost graphic now seems insignificant, and the control effort is clearly bounded within a small region.

Just as with the λ adaptations, we are still concerned with the overall control effort relative to the allowable maximum in the order adaptations. This therefore justifies looking for trends in plots with the y-axis lower limit of zero. This was undertaken for the trend lines shown below.

The control effort types for constant λ are:

1:



As order is increased, the control effort undergoes a dramatic reduction and then reaches a somewhat constant level.

2:

These types of systems have a control effort that is bounded to a small region relative to the overall level.

3:

For these systems, control effort increases as order is increased.

Curves of type 1 and 2 are very common in the numerical simulations and in the laboratory experiments. Type 3 curves typically occur when λ is very low, although a very low λ does not guarantee the behavior.

Just as with the λ adaptations, the order adaptations are geared toward improving the performance while keeping the control effort within bounds (and also keeping the order within the allowable limit). In the λ adaptations, improving the performance always occurred at the cost of increasing controller effort. With order adaptations, this is not necessarily the case.

For example, if the performance curve is of type 1,  , and the corresponding control effort is of type 1,  , then an increase in order will improve performance and reduce controller effort. If a system has a type 1 performance curve,  ,and the control effort curve type 2, , then improvements in performance occur without a significant change in control effort.

If the performance curve is of type 1,  , and the control effort is of type 3,  , then improvements in performance come at the expense of a rise in control effort. Out of three control

effort types, this represents the worst case scenario. However, this situation is completely analogous to that seen in the λ adaptations (constant order) discussed previously, where all the control effort curves have this type of behavior.

The fuzzy logic module designed for the control effort monitoring in the λ adaptations, the U_{max} module, was designed around this behavior. This module was altered for use with the order adaptations, as shown in Section 6.2.1.3, and will therefore control the adaption rate based on assumption of type 3,  , behavior. However, since this module was not designed with consideration for control effort types 1,  , and 2, , the module will behave in ways that may be considered non-ideal. This topic will be further explored in the examples shown in Sections 6.2.2 and 6.2.3, and in the discussion in Chapter 7.

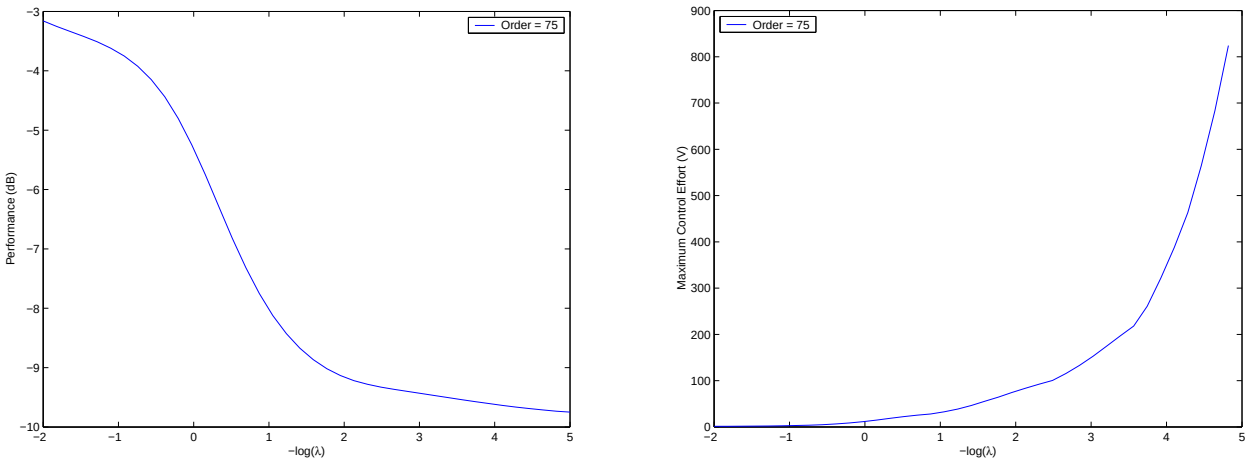
5.2.4 Example of a Type 1 Performance Curve

We will now take a closer look into a fixed-order performance curve of type 1. Figure 5.53 shows the performance and maximum effort for a 75th-order controller. The performance curve includes a steep drop for $1 < \lambda < 0.1$ and then flattens out. The performance improves even as λ is reduced beyond 0.01, and this further reduction comes at the expense of dramatic increases in controller effort. We will now investigate this behavior.

Figure 5.54 shows controller frequency response magnitude for each λ, while Figure 5.55 shows a different view of this same surface. Note that the lowest λs are in the back half of the plot. We can see that the controller design rapidly changes for the higher λs and seems to stabilize in a majority of the frequency bins for the lower λs. A two-dimensional view of this plot will be shown later.

In Figure 5.56, we see the closed-loop frequency response magnitude due to these same controllers. Note that the lowest λs are now displayed in the closest half of the plot to aid in the visualization. For the higher λs, we still see the peaks of the modal response. As λ is reduced, the modes are attenuated and the overall shape becomes flattened. It is also evident that the shape of the surface does not change much as λ is dropped below a certain point.

We now take an in-depth look at these controller and closed-loop response surfaces. Figure 5.57 shows several constant λ cuts of these surfaces. For convenience, the highest λ has been labeled



(a) Performance

(b) Maximum Effort

Figure 5.53: 75th-order Controller

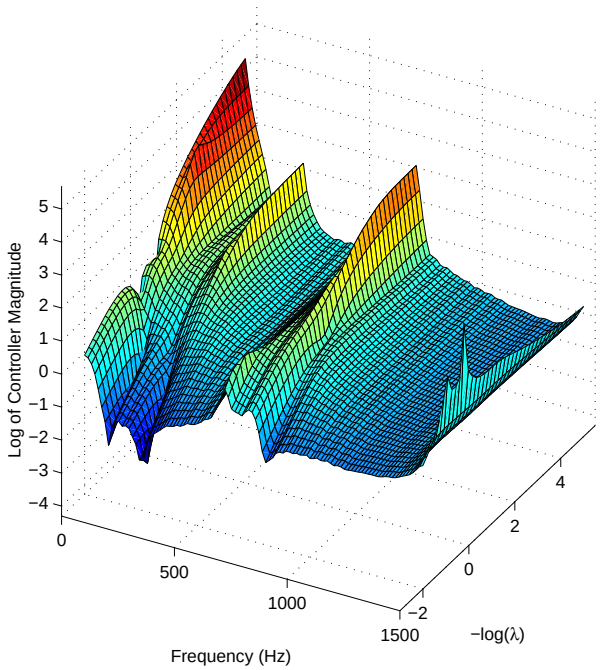


Figure 5.54: First View of the 75th-order Controllers

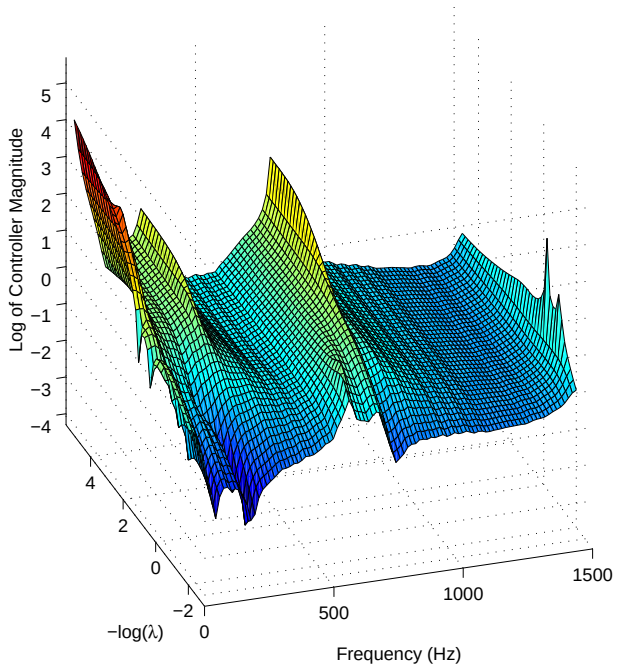


Figure 5.55: Second View of the 75th-order Controllers

as the first controller, and the lowest λ as the 40th controller. We can see once we reach the 30th controller that we don't note much change in controller frequency response for higher frequencies. There is, however, a significant boost in the 0 Hz region. The closed-loop frequency response also does not change much after the 30th controller. Figures 5.58 and 5.59 further illustrate this. Therefore, the reason that performance no longer improves dramatically for $\lambda < 0.01$ is that the controller design remains relatively constant.

We have established that the performance for a 75th-order controller gradually improves for $1 \times 10^{-5} < \lambda < 0.01$ (see Figure 5.53). Looking at the curve it would appear that the performance could be improved indefinitely with further reduction in λ. However, this is note the case. The closed-loop system becomes unstable before λ ever reaches 1×10^{-6}. Since we are dealing with a discrete system, this occurs when at least one of the closed-loop poles has a radial distance from the origin greater than one (the unit circle).

Figure 5.60 shows the farthest closed-loop pole range distance for a wider range of λs. The *actual* line represents the poles of a system in which the controller is combined with the actual plate model. If the system identification incorrectly identifies the system, the controller will be

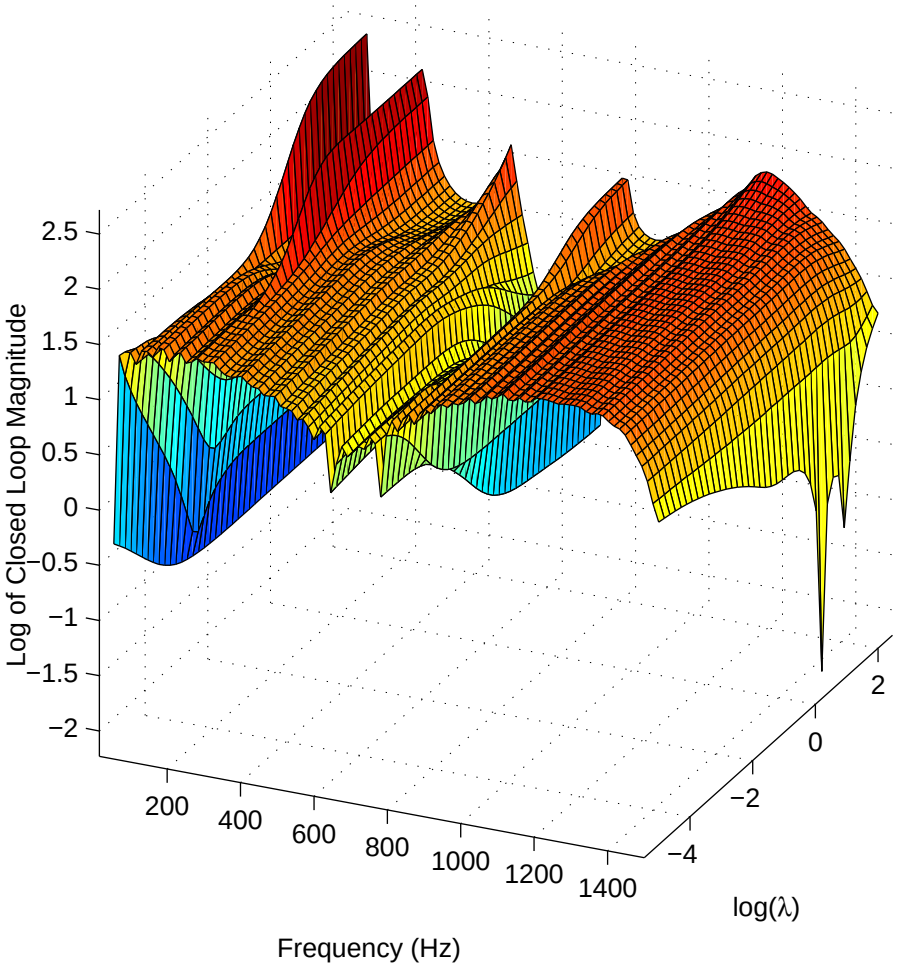


Figure 5.56: Closed-loop Frequency Response

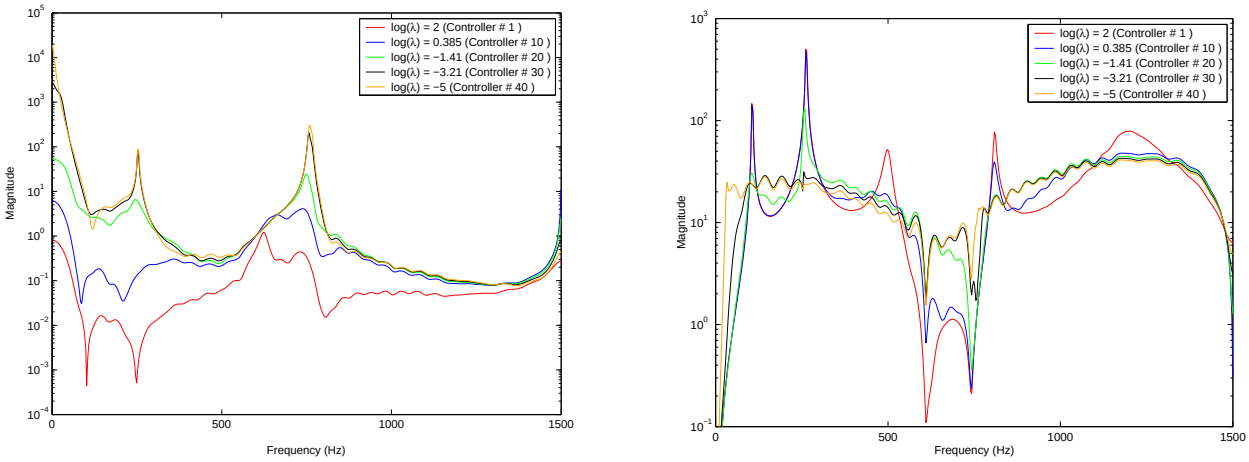


Figure 5.57: Control and Closed-loop Response Cuts

designed on incorrect information. Hence, the *actual* line will include the effect of the system identification error on the closed-loop pole locations. The *apparent* line represents the closed-loop pole locations in which the controller is combined with the identified plate model.

What this figure indicates is that the farthest actual closed-pole location moves outside the unit circle, and therefore becomes unstable, shortly after λ drops below 1×10^{-5}. The apparent pole locations do not move into the unstable region. This means that the resulting instability is not a design flaw of GPC, but instead could be a result of a system identification error.

5.3 Closed-loop Behavior of GPC

In this section, we will demonstrate how generalized predictive control achieves control. We saw in Section 5.2 that λ adaptations are the dominant effect on performance, and that order adaptations are the secondary effect[7]. We will therefore analyze the effect of reductions in λ on the position of the closed-loop poles. This analysis is best accomplished in the z-plane. This discussion will be limited to results obtained from numerical simulations. Furthermore, we will investigate the role of the plant zeros in the closed-loop pole positions.

In order to do this, we will look at two different systems with increasing complexity. We will

[7] λ is the control effort. Reducing λ gives us more aggressive controllers, i.e. controllers that will use more control effort to achieve control.

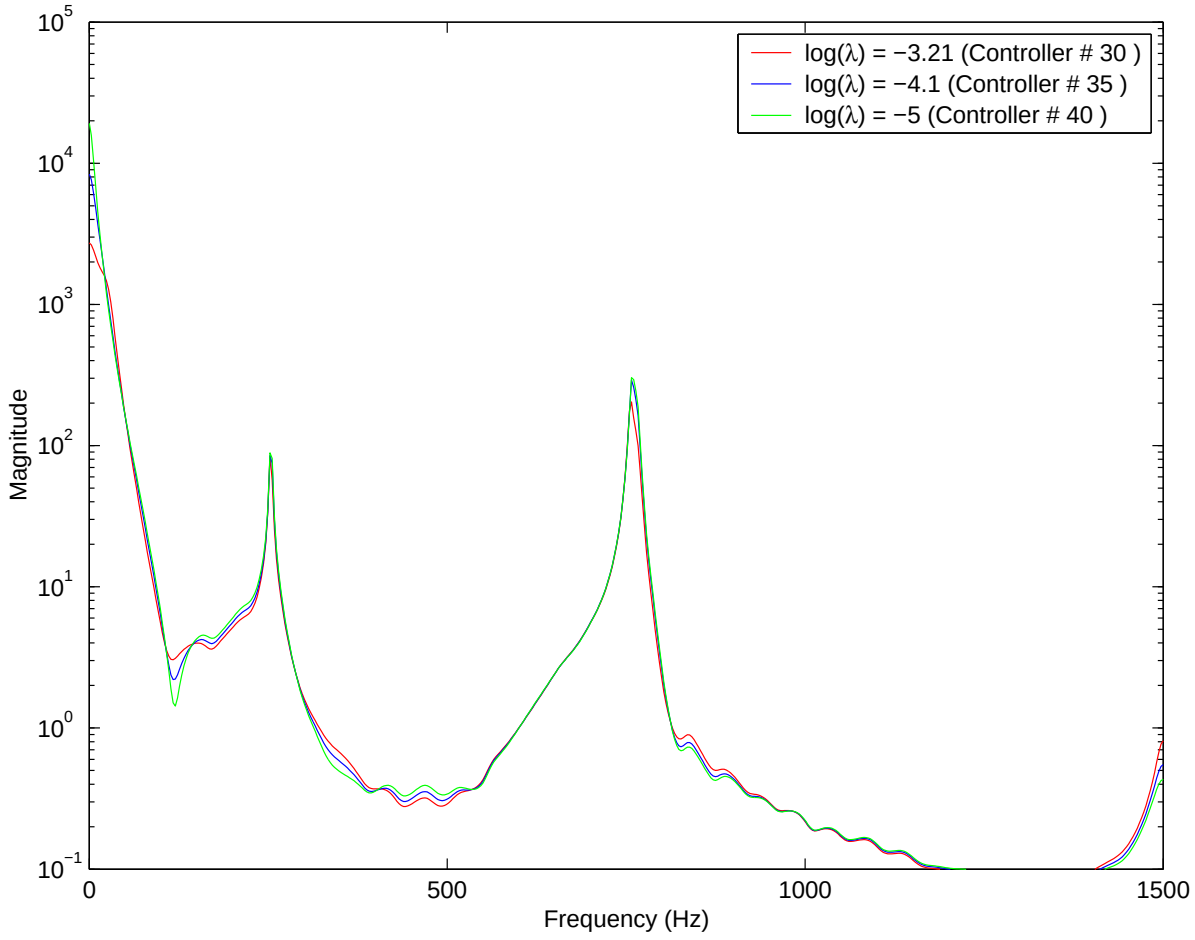


Figure 5.58: High Performance Controllers Frequency Responses

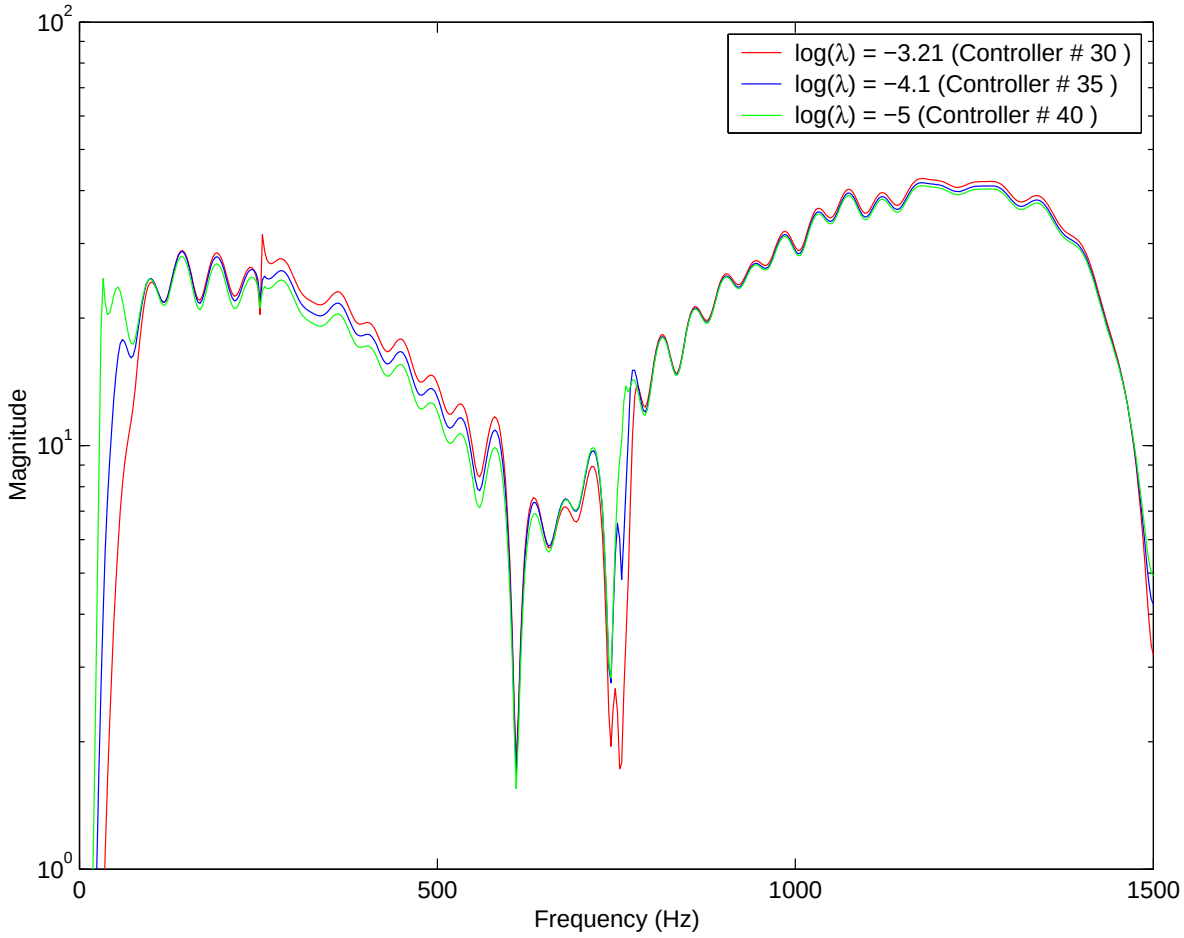


Figure 5.59: High Performance Closed-loop Frequency Responses

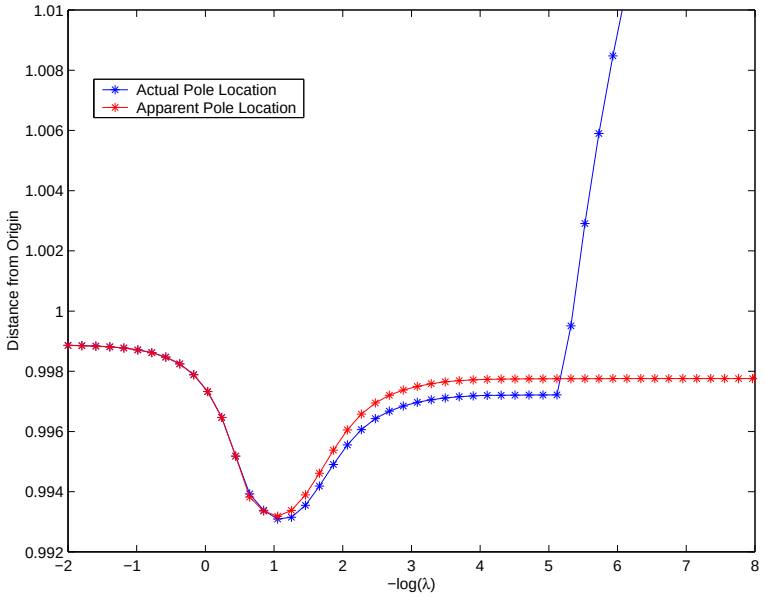


Figure 5.60: Farthest Closed-loop Location

start with a very simple case of a two-mode system without the anti-aliasing filter. The system will be disturbed by a single tone instead of the usual broadband disturbance. The closed-loop system goes becomes for relatively high λ's, and the reason for this will be explored.

Having achieved an understanding of what GPC is doing, we will move on to a five-mode system with a broadband disturbance and an anti-aliasing filter. The resulting closed-loop pole locations at first appear to be unintelligible, but a closer look reveals that the principles of the single-tone disturbance are still applicable to the broadband case. The analysis can be performed on larger mode systems, but the number of plots necessary to do this makes their inclusion in this paper prohibitive. The results derived in this section will, however, be relevant to systems of any size.

It is important to note that all the systems in this section are still subsets of the clamped plate model developed for this work. The plant zero locations of these types of plants makes them particularly interesting to investigate. The final conclusions of this section will not be limited to clamped plate systems but will instead be general results applicable to all systems.

We will also show that it is the apparent plant zeros that drive the iteration of the system. Non-minimum-phase zeros can be especially dangerous.

Table 5.2: Plant Poles and Zeros

Poles				
Number	Real	Imag	Abs	
1	0.849609	-0.5221769	0.9972483	first mode
2	0.849609	0.5221769	0.9972483	
3	0.4976451	-0.8613564	0.9947791	second mode
4	0.4976451	0.8613564	0.9947791	
Zeros				
Number	Real	Imag	Abs	
1	0.8737867	-0.5117801	1.012631	near first mode
2	0.8737867	0.5117801	1.012631	
3	1	0	1	
4	0.4706691	0	0.4706691	

5.3.1 Two-mode System without Filter - Single-Tone Disturbance

We will start this discussion with an extremely simple system consisting of a two-mode system disturbed by a single tone. In addition, we will not include the anti-aliasing filter in order to reduce the complexity of the system even further.

The plant poles and zeros are shown in the z-plane plot of Figure 5.61; the pole locations are marked by an 'x' and the zeros locations are marked by an ○. The large red circle is the stability boundary. The poles are very close to the stability boundary, demonstrating the light damping present in the system. The exact numerical positions of the poles and zeros are show in Table 5.2. Each position in the table not only shows the real and imaginary coordinates, but also the radial distance from the origin (the column labeled Abs). If the radial distance is larger than 1.0, than the pole or zero is in the unstable region.

Some explanation of the zeros is in order. The zeros shown in Figure 5.61 are associated with the transfer function between the voltage input of the piezoelectric patch to the acceleration output. We will refer to this as the control path. The zeros for the disturbance path, i.e. those from the disturbance force input to the acceleration output are not known in the laboratory environment. For this reason, they will not be shown in any of the figures in this section. We will sometimes refer to the plant as the *actual plant* in order to distinguish it from the *apparent plant* defined subsequently.

In Figure 5.62, we have shown an enlarged view of the plant in a region around the first mode. From this view, it is clear that one set of plant zeros is outside the stability boundary. This will play an important role in the closed-loop stability as will be demonstrated shortly.

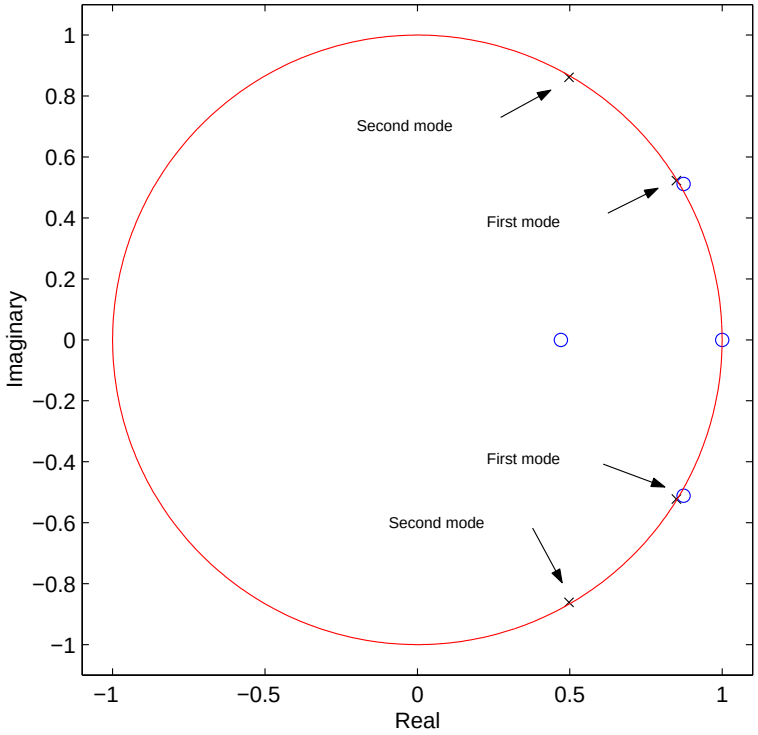


Figure 5.61: Plant Poles and Zeros

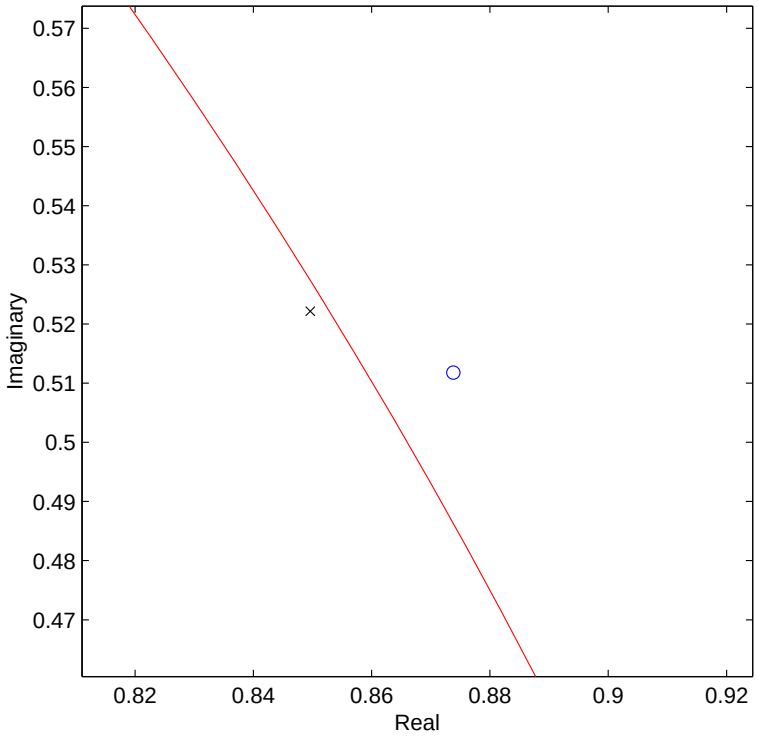


Figure 5.62: Plant Detail

Once the system identification is performed, we can plot its pole and zero locations in the same manner in which we plotted the plant. These poles and zeros will be referred to as the *apparent plant.* Recall that the GPC control law is based on the results of the system identification. Hence, it the poles and zeros of the apparent plant and not those of the actual plant that are used to calculate the control law. In Figures 5.63 and 5.64, we have shown this apparent plant. The numerical pole locations are shown in Table 5.3. A visual comparison of the apparent plant to the actual plant (Figures 5.61 and 5.62) shows that the two are almost identical. This is confirmed by looking at the numerical position of Tables 5.2 and 5.3. If we consider each pole and zero as a vector, we can calculate the vector difference between the apparent and actual locations. The magnitudes of these vectors are:

First Mode Poles	1.77E-05
Second Mode Poles	2.74E-07
Plant Zeros #1,2	3.03E-05
Plant Zero #3	8.60E-07
Plant Zero #4	5.90E-08

These magnitudes quantify the quality of the system identification. The highest errors occur on the first and second plant zeros with a value of 3×10^{-5}, and this system identification can be considered to be very accurate.

The most significant difference between the actual and apparent plants is the inclusion of an additional pole and zero in the apparent plant. This can be seen clearly in Figure 5.64 and in Table 5.3. The position of this pole and zero pair corresponds to a frequency of 250 Hz, which is also the frequency of the single-tone disturbance. Hence, this pole-zero pair represents the inclusion of the disturbance information in the system identification process. The GPC control law is calculated using the apparent plant. Since the disturbance information is embedded into this apparent plant, the control law is not only based on the system characteristics, but also on the disturbance information.

We now wish to show the effect of changing λ on the closed-loop poles. To show this effect, we have iterated λ from an initial value of 100 (non-aggressive controller) to a final value of zero (very aggressive controller). For each value of λ, we compute a controller and then close the loop on the actual plant. The resulting closed-loop poles and zeros will be referred to as the *actual*

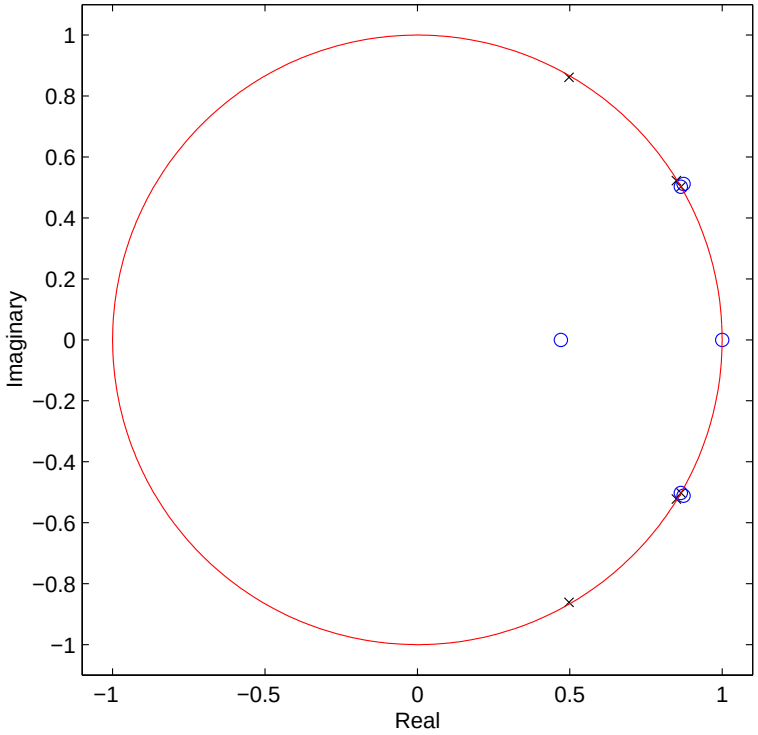


Figure 5.63: Apparent Plant

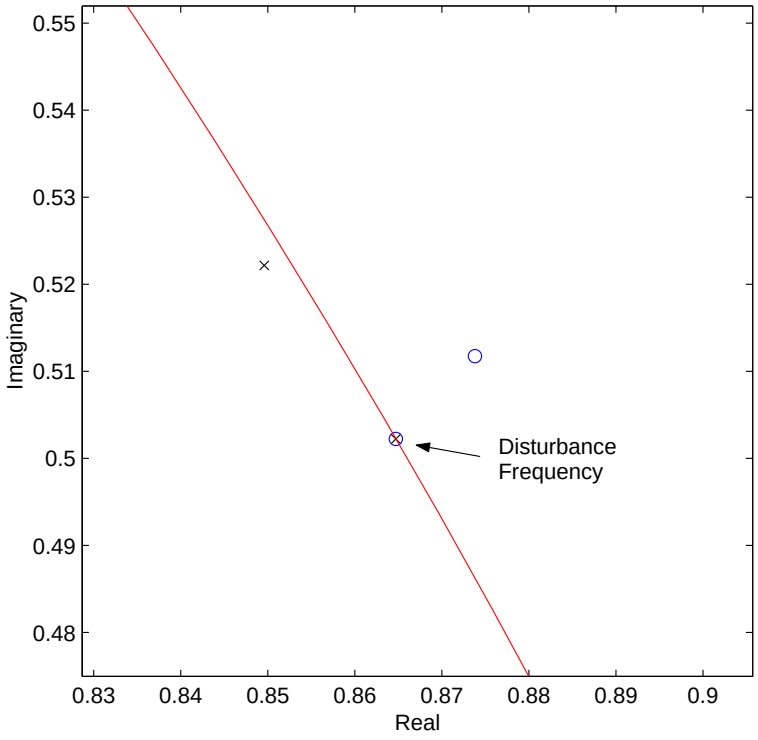


Figure 5.64: Apparent Plant Detail

Table 5.3: Apparent Plant Poles and Zeros

Poles Number	Real	Imag	Abs	
1	0.8647223	-0.5022245	0.999987	disturbance
2	0.8647223	0.5022245	0.999987	information
3	0.849608	-0.5221592	0.9972382	first mode
4	0.849608	0.5221592	0.9972382	
5	0.4976448	-0.8613565	0.9947791	second mode
6	0.4976448	0.8613565	0.9947791	

Zeros Number	Real	Imag	Abs	
1	0.8737896	-0.5117499	1.012619	near first mode
2	0.8737896	0.5117499	1.012619	
3	0.9999991	0	0.9999991	plant zero #3
4	0.8647186	-0.5022366	0.9999899	disturbance
5	0.8647186	0.5022366	0.9999899	information
6	0.4706691	0	0.4706691	plant zero #4

closed-loop poles. The locations of all of these actual closed-loop poles for the entire range of λs are shown in Figure 5.65. The pole and zero locations are color-coded to show the starting and final position. The poles start with a blue color and end in red. The zero locations start with a light blue color and end in dark green.

There are several features of this figure that should be noted. First, we see that the second-mode poles are moved a large distance from their original position to their final positions on the real line. Second, we also see that two pairs of zeros are pushed from an initial position well within the stability boundary to a final position near the boundary. The third feature is a cluster of poles that are pushed to the deadbeat position of $z = 0 + 0i$. The reasons for these features will become more clear as this discussion progresses.

The region around the first mode is extremely busy and warrants a closer look. A detail of this region is shown in Figure 5.66. There two important things occurring in this region. The first is the movement of the first mode pole. As λ is reduced, this pole is pushed into the unstable region. Compared to the second-mode pole, the first-mode pole moves only a very small distance during the iteration. Second, we note that a zero coming in from the top of the figure meets up with the first-mode pole in an attempt to cancel this pole. Finally, we see a zero settling in on the stability boundary.

The movement of the first-mode pole causes a system instability for a relatively high λ of 0.36.

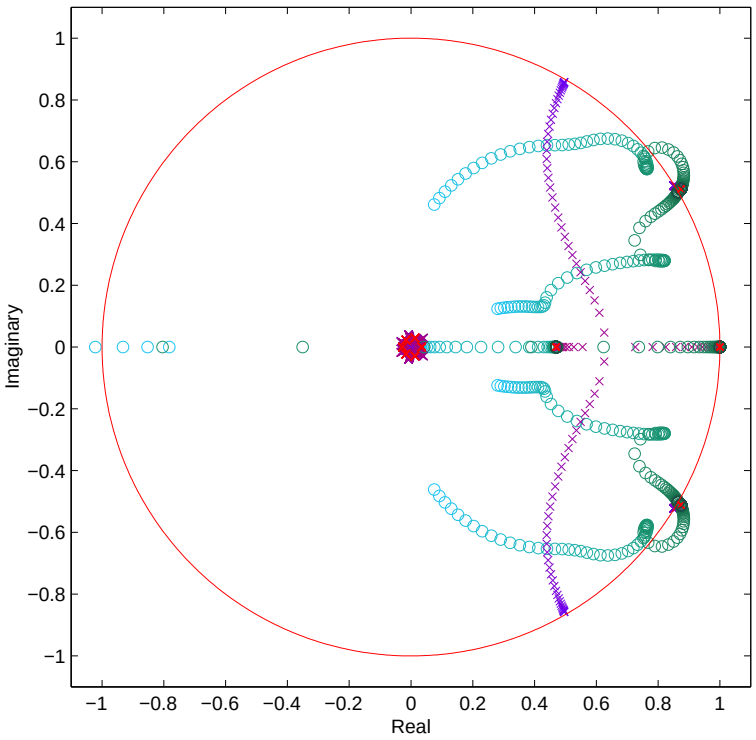


Figure 5.65: Actual Closed-loop Poles

Since the instability occurs at an early iteration, the performance is very poor -0.9 dB. The fact that the final positions of both the poles and zeros are related to the apparent plant will be demonstrated shortly.

We will now show the *apparent closed-loop poles and zeros*. These apparent poles are calculated by closing the loop on the apparent plant, i.e. combining the controller with the system identification. Figures 5.67 and 5.68 show these apparent poles. We are interested in how well we can predict the movement of the actual closed-loop poles by looking at the apparent closed-loop poles. A visual comparison of these figures to those of the actual closed-loop poles (Figures 5.65 and 5.66) shows that this prediction seems accurate for this system. Both the motion of the first- and the second-mode poles is predicted. The most obvious difference between the two systems is the cluster of poles near the origin.

A more in-depth analysis of the exact details will now be performed. We will limit our analysis to the final iteration ($\lambda = 0$) for reasons that will become clear. A visual comparison of the actual and apparent closed-loop locations will give us some idea of what is taking place. In another comparison, we will look at the numerical position to gain further understanding of the

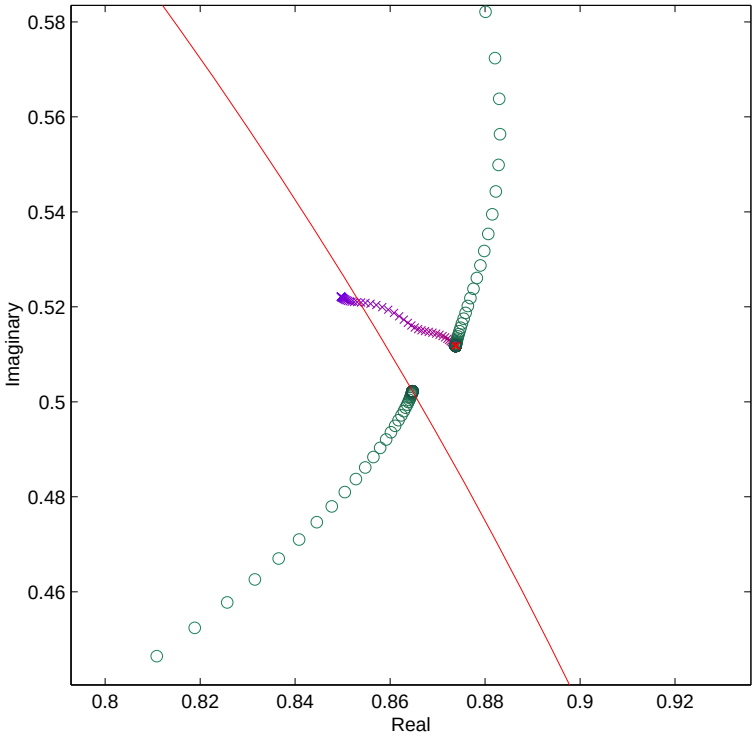


Figure 5.66: Actual Closed-loop Poles Detail

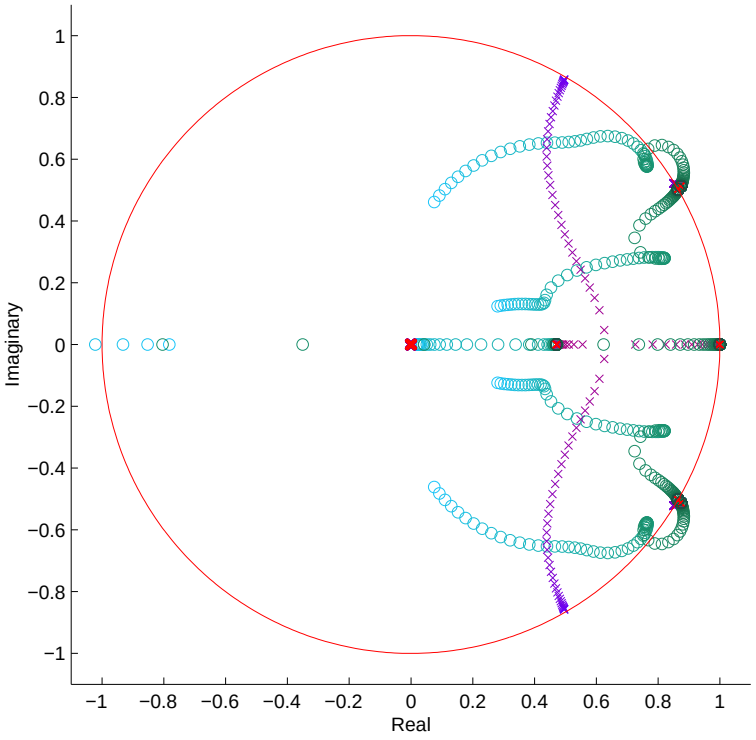


Figure 5.67: Apparent Closed-loop Poles

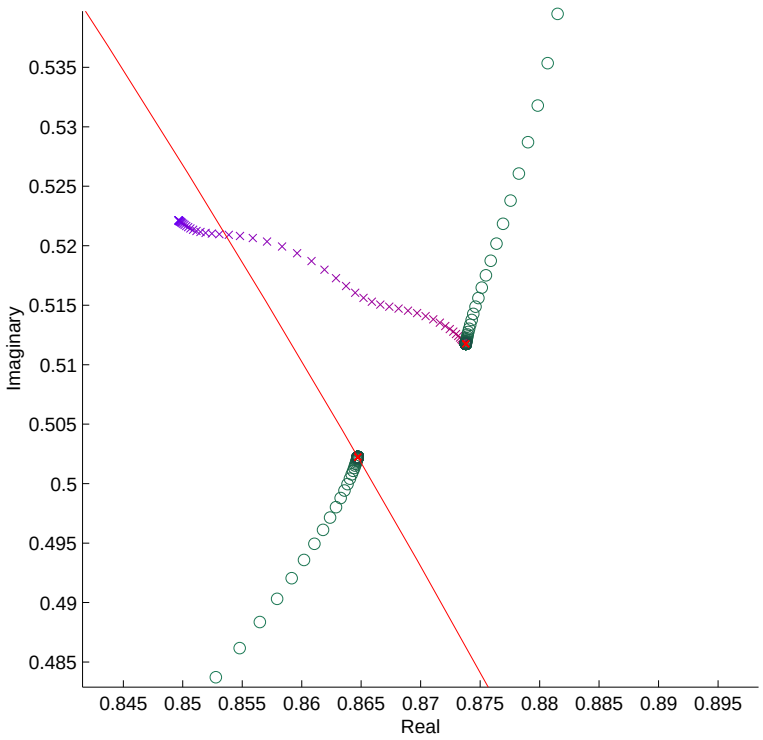


Figure 5.68: Apparent Closed-loop Poles Detail

phenomenon.

Since it is the apparent plant poles that determine the control law, it is reasonable to look at the apparent closed-loop pole locations first. These locations are shown in Figures 5.69 and 5.70, and numerically in Table 5.4. A visual comparison of these positions to those of the apparent plant (Figures 5.63 and 5.64) indicates that some of the poles and zeros are driven to the apparent plant zero locations.

This fact is confirmed by comparing the numerical positions of the closed-loop locations (Table 5.4) to those of the apparent plant (Table 5.3). The results of this comparison are the annotations shown in Table 5.4. Clearly, the closed-loop poles can be split into two groups. The first group (group A) consists of poles 1 through 6, which are placed exactly on the apparent plant zero locations. The remaining poles (7-12) form a second group (group B). The group B poles are driven to locations near the origin.

The final closed-loop zero locations are also influenced by the apparent plant zeros. The controller does not have the ability to move the plant zeros; therefore, these zeros will always be present in any closed-loop configuration. We have already seen in Figures 5.67 and 5.68 that the

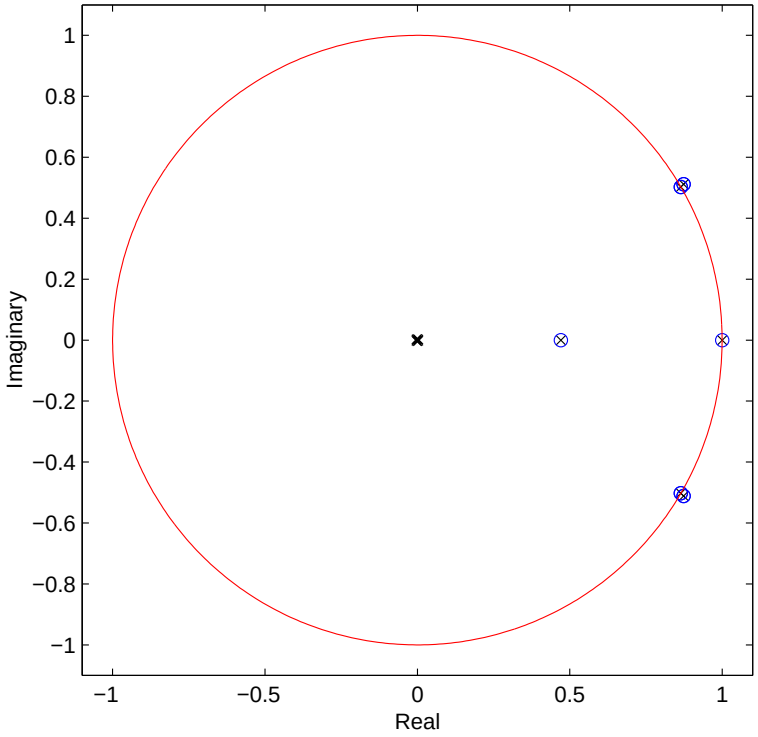


Figure 5.69: Apparent Closed-loop Poles $\lambda = 0$

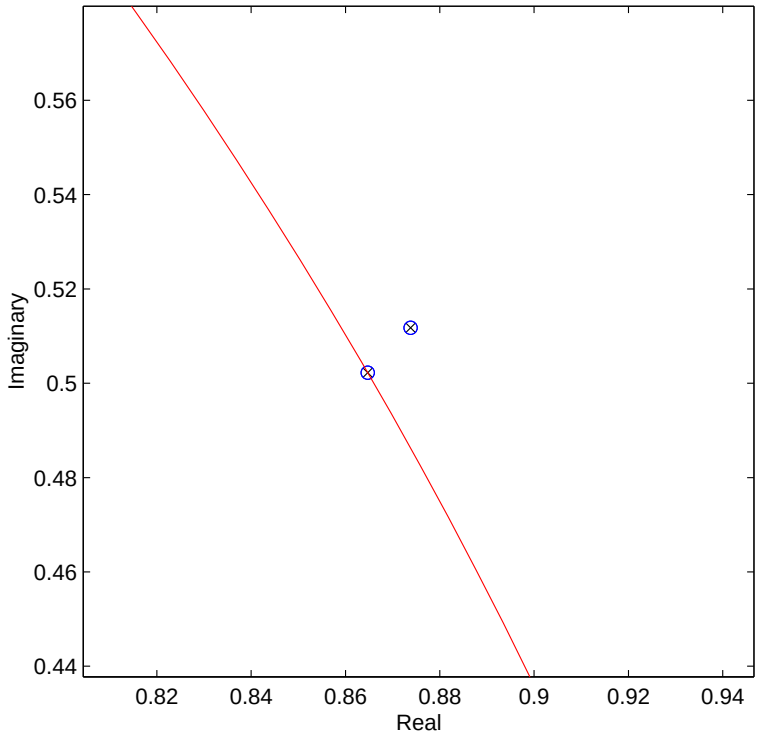


Figure 5.70: Apparent Closed-loop Poles - $\lambda = 0$ Detail

Table 5.4: Apparent Closed-loop Poles - $\lambda = 0$

Poles

Number	Real	Imag	Abs	
1	0.8737896	-0.5117499	1.012619	} poles are on app plant zeros #1,2
2	0.8737896	0.5117499	1.012619	
3	0.9999991	0	0.9999991	pole is on app zero #3
4	0.8647186	-0.5022366	0.9999899	} poles are disturbance zeros (app plant #4,5)
5	0.8647186	0.5022366	0.9999899	
6	0.4706691	0	0.4706691	pole is on app zero #6
7	0.003826745	0	0.003826745	} cluster of poles near origin
8	0.001894504	-0.003313975	0.003817274	
9	0.001894504	0.003313975	0.003817274	
10	-0.001913233	-0.003281541	0.003798549	
11	-0.001913233	0.003281541	0.003798549	
12	-0.003789288	0	0.003789288	

Zeros

Number	Real	Imag	Abs	
1	0.8737896	-0.5117499	1.012619	} app plant zeros #1,2
2	0.8737896	0.5117499	1.012619	
3	0.8737896	-0.5117499	1.012619	} additional zeros placed on plant zeros #1,2
4	0.8737896	0.5117499	1.012619	
5	0.9999991	0	0.9999991	app plant zero #3
6	0.9999991	0	0.9999991	zero driven to app plant zero #3
7	0.8647186	-0.5022366	0.9999899	} disturbance zeros (app plant zero #4,5)
8	0.8647186	0.5022366	0.9999899	
9	0.8647186	-0.5022366	0.9999899	} additional zeros driven to disturbance zeros
10	0.8647186	0.5022366	0.9999899	
11	0.4706691	0	0.4706691	app plant zero #6
12	0.4706691	0	0.4706691	zero driven to app plant zero #6

GPC control law creates zeros which are moved around the z-plane. In Table 5.4, we see that these additional zeros are driven to apparent plant zero locations.

It now becomes clear what the GPC control law is doing as λ is reduced to zero. The group A poles are driven to the apparent plant locations. Simultaneously, GPC drives zeros to these exact locations in an attempt to cancel out the pole dynamics. Hence, the apparent plant zeros act as attraction points for both the closed-loop pole and zeros. The extraneous poles (group B) are driven to locations near the origin. Also note that GPC makes no distinction between those poles and zeros belonging to the plant, and those representing the disturbance information.

Given this new insight into how GPC achieves control, we can reexamine the instability that occurred in this system. We have already shown that the apparent plant has a non-minimum zero near the first-mode poles. Since these zeros act as attraction points for the closed-loop poles, we know that the closed-loop system will become unstable if λ is made small enough. The same can be said for any system with non-minimum zeros in its system identification.

We can now compare what GPC is attempting to do (apparent closed-loop poles) to what is actually occurring (actual closed-loop poles). Given our high accuracy system identification, we would expect them to be almost exactly the same. In Figures 5.71 and 5.72 and Table 5.5, we have shown the actual closed-loop poles for $\lambda = 0$.

A visual comparison of Figures 5.71 and 5.72 to the apparent closed-loop poles of Figures 5.69 and 5.70 shows that this is indeed the case, however, there are two important differences between the two. The first difference is the cluster of poles near the origin. The actual pole cluster is farther away from the origin than the apparent pole cluster. This indicates that the actual poles are not as close to being deadbeat as the apparent poles. The second difference is the absence of the disturbance pole and zero in the actual closed-loop poles.

The annotations in Table 5.5 show the relevance of each pole and zero location. Here we see that the first four poles are driven to the plant zero locations, while the remaining poles are driven in a cluster around the origin. This is just as the apparent closed-loop poles predicted.

The actual closed-loop zero locations are a combination of the plant zeros, which never move, and the additional zeros created by closing the loop. Here we see that these additional zeros are moved to positions on top of the apparent plant zero locations. This behavior is identical to that seen for the apparent closed-loop pole movements.

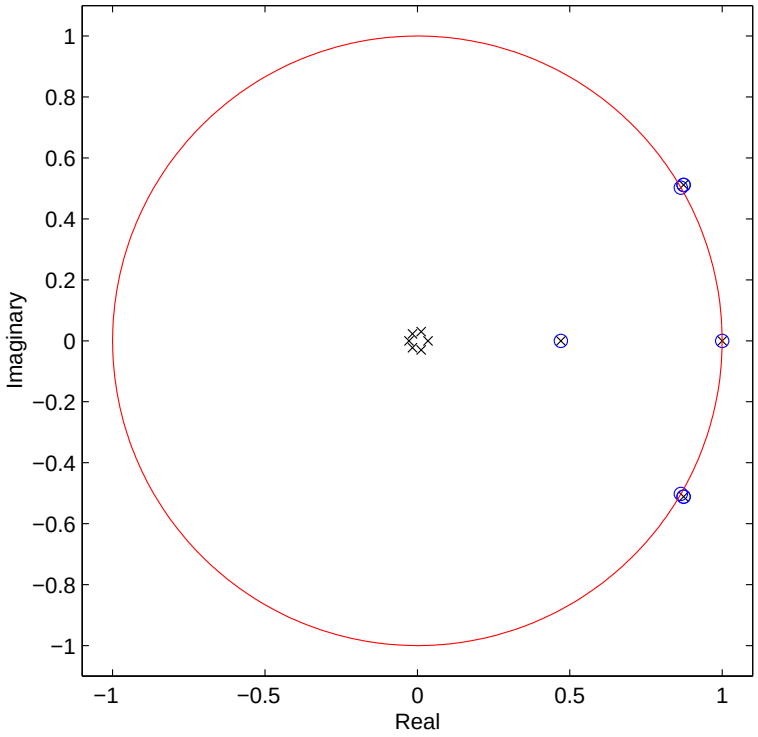


Figure 5.71: Actual Closed-loop Poles - $\lambda = 0$

Also note that a pair of zeros (numbers 7 and 8) are placed at the disturbance zero location. The placement of these zeros shows how GPC achieves disturbance rejection. By forcing these zeros to the disturbance location, GPC creates a notch in the closed-loop transfer function. This ensures that the effect of the disturbance is minimized.

As we have already noted, the system becomes unstable for a very high λ due to the non-minimum-phase zero. For this system, we have not included the anti-aliasing filter described in Section 3.1.3. Including this filter moves the non-minimum-phase zero back into the stable region, therefore allowing the closed-loop system to stay stable for much lower λs.

Table 5.5: Actual Closed-loop Poles - $\lambda = 0$

Poles Number	Real	Imag	Abs	
1	0.8737867	-0.5117801	1.012631	poles are on plant zeros #1,2
2	0.8737867	0.5117801	1.012631	
3	0.9999999	0	0.9999999	pole is on plant zero #3
4	0.4706684	0	0.4706684	pole is on plant zero #4
5	0.03487844	0	0.03487844	cluster of poles near origin
6	0.01269045	-0.02978617	0.0323769	
7	0.01269045	0.02978617	0.0323769	
8	-0.01624841	-0.02304249	0.02819516	
9	-0.01624841	0.02304249	0.02819516	
10	-0.02776178	0	0.02776178	

Zeros Number	Real	Imag	Abs	
1	0.8737867	-0.5117801	1.012631	app plant zeros #1,2
2	0.8737867	0.5117801	1.012631	
3	0.8737896	-0.5117499	1.012619	additional zeros placed on plant zeros #1,2
4	0.8737896	0.5117499	1.012619	
5	1	0	1	plant zero #3
6	0.9999991	0	0.9999991	zero driven to plant zero #3
7	0.8647186	-0.5022366	0.9999899	zeros driven to disturbance zeros
8	0.8647186	0.5022366	0.9999899	
9	0.4706691	0	0.4706691	plant zero #4
10	0.4706691	0	0.4706691	zero driven to plant zero #4

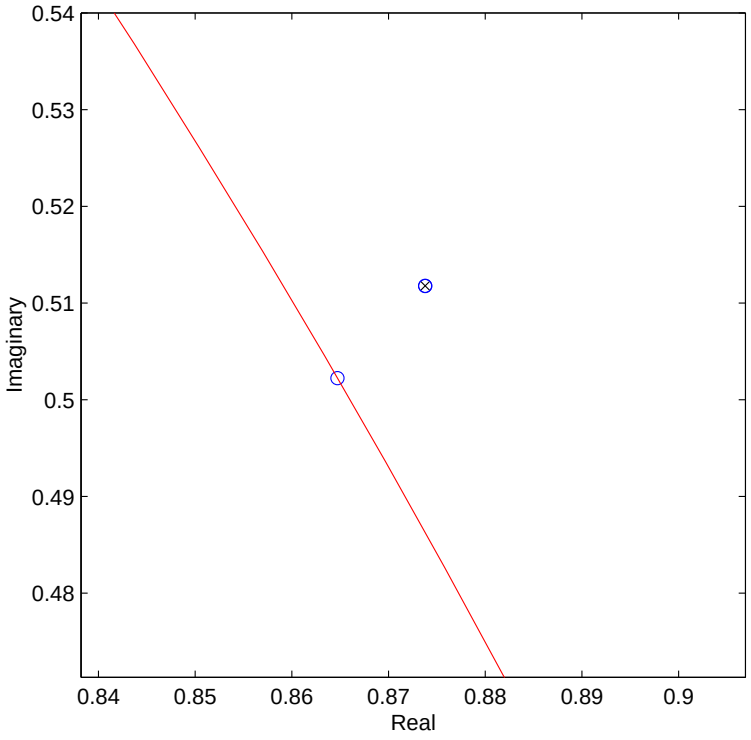


Figure 5.72: Actual Closed-loop Poles - $\lambda = 0$ Detail

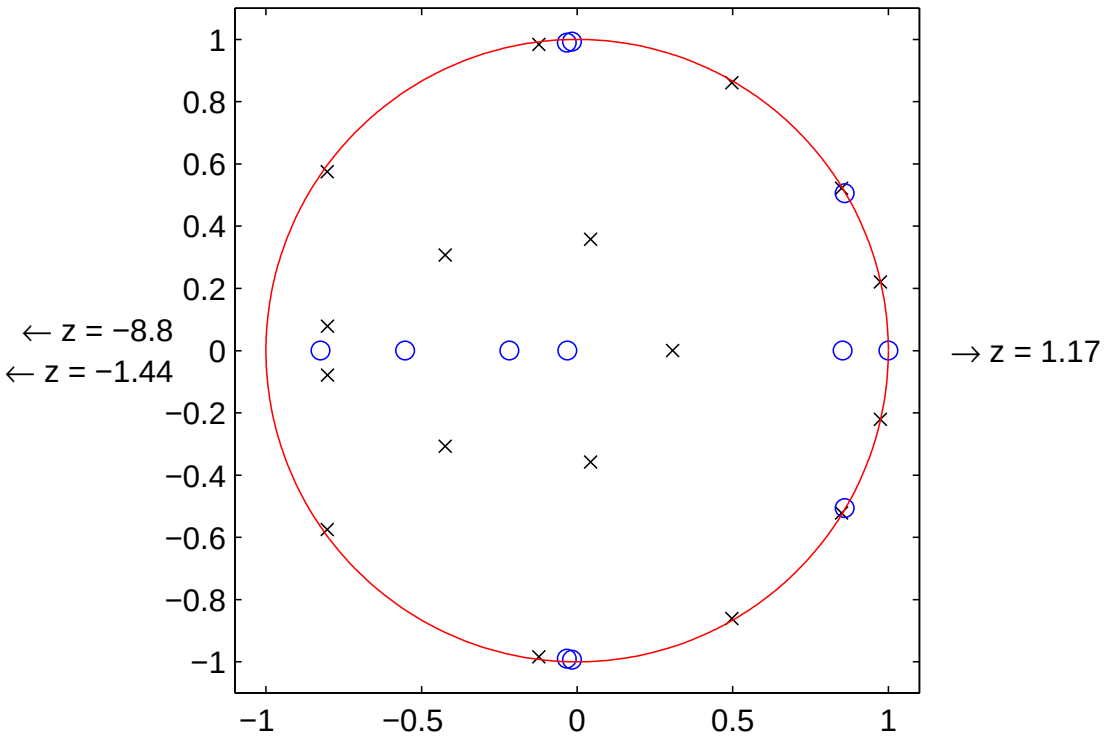


Figure 5.73: Plant Poles and Zeros

5.3.2 Five-mode System - Broadband Disturbance

In this section, we will analyze a five-mode system with a broadband disturbance. We will include the anti-aliasing filter, which will be used in all of the numerical simulations of Chapter 6. The acceleration is measured at location 3.

The plant is shown in Figure 5.73, and the numerical locations of the poles are shown in Table 5.6. The poles that create the modes are close to the stability boundary because of the low damping present in the system. The remaining poles belong to the anti-aliasing filter. Also note that there are three zeros outside of the stability boundary. Based on the conclusion of the last section we expect that these zeros will lead to a closed-loop instability.

We have chosen to close the loop with a 30th-order controller. A 30th-order system identification and controller is much larger than the true system order, which is 17. Choosing a much larger order helps to guarantee a high quality system identification. The results of the system identification are shown in Figure 5.74, and the numerical pole and zero locations are shown in Table 5.7.

A visual comparison of Figure 5.74 to Figure 5.73 gives us a preliminary understanding of

Table 5.6: Plant Poles and Zeros

Poles

Number	Real	Imag	Abs	
1	0.974107	-0.221069	0.998877	first mode
2	0.974107	0.221069	0.998877	
3	0.849785	-0.521885	0.997246	second mode
4	0.849785	0.521885	0.997246	
5	0.497602	-0.861318	0.994725	third mode
6	0.497602	0.861318	0.994725	
7	-0.12266	-0.983943	0.991559	fourth mode
8	-0.12266	0.983943	0.991559	
9	-0.80295	-0.574709	0.987429	fifth mode
10	-0.80295	0.574709	0.987429	
11	-0.80262	-0.07896	0.806499	filter poles
12	-0.80262	0.07896	0.806499	
13	-0.42424	-0.30749	0.523959	
14	-0.42424	0.30749	0.523959	
15	0.043054	-0.357532	0.360115	
16	0.043054	0.357532	0.360115	
17	0.307123	0	0.307123	

Zeros

Number	Real	Imag	Abs	
1	-8.80228	0	8.802277	
2	-1.441	0	1.441004	
3	1.170081	0	1.170081	
4	1.000003	0	1.000003	
5	1	0	1	
6	0.859389	-0.506055	0.997317	near second mode
7	0.859389	0.506055	0.997317	
8	-0.01632	-0.99353	0.993664	near fourth mode
9	-0.01632	0.99353	0.993664	
10	-0.03365	-0.989843	0.990415	near fourth mode
11	-0.03365	0.989843	0.990415	
12	0.853367	0	0.853367	
13	-0.82469	0	0.824689	
14	-0.55302	0	0.553022	
15	-0.21752	0	0.217521	
16	-0.03133	0	0.031325	

Table 5.7: Apparent Plant Poles and Zeros

Poles

Number	Real	Imag	Abs	
1	0.974103	-0.221072	0.998874	first mode
2	0.974103	0.221072	0.998874	
3	0.849694	-0.521769	0.997107	second mode
4	0.849694	0.521769	0.997107	
5	0.497602	-0.86132	0.994726	third mode
6	0.497602	0.86132	0.994726	
7	-0.12264	-0.983953	0.991566	fourth mode
8	-0.12264	0.983953	0.991566	
9	-0.80294	-0.574711	0.987423	fifth mode
10	-0.80294	0.574711	0.987423	
11	-0.2692	-0.903294	0.942553	
12	-0.2692	0.903294	0.942553	
13	-0.48922	-0.785787	0.925633	
14	-0.48922	0.785787	0.925633	
15	-0.67409	-0.6099	0.909054	
16	-0.67409	0.6099	0.909054	
17	0.793038	-0.444188	0.908962	
18	0.793038	0.444188	0.908962	
19	0.491501	-0.729695	0.879789	
20	0.491501	0.729695	0.879789	
21	-0.79346	-0.359904	0.871273	
22	-0.79346	0.359904	0.871273	
23	0.601151	-0.616472	0.861058	
24	0.601151	0.616472	0.861058	
25	-0.77522	-0.04584	0.776575	close to plant poles #11,12
26	-0.77522	0.04584	0.776575	
27	-0.61564	-0.246786	0.663258	close to plant poles #13, 14 (large error)
28	-0.61564	0.246786	0.663258	
29	0.067074	-0.225288	0.235061	close to plant poles #15,16
30	0.067074	0.225288	0.235061	

plant pole #17 not identified

Zeros

Number	Real	Imag	Abs	
1	-1095.18	0	1095.177	extraneous
2	-8.86485	0	8.864849	plant zero #1
3	-1.44094	0	1.440937	plant zero #2
4	1.169403	0	1.169403	plant zero #3
5	1.002578	-0.015951	1.002705	close to plant zeros #4,5
6	1.002578	0.015951	1.002705	
7	0.859625	-0.505635	0.997308	near second mode, plant zeros #6,7
8	0.859625	0.505635	0.997308	
9	-0.0162	-0.993693	0.993825	near fourth mode, plant zeros #8,9
10	-0.0162	0.993693	0.993825	
11	-0.03383	-0.989801	0.990379	near fourth mode,plant poles #10,11
12	-0.03383	0.989801	0.990379	
13	-0.26924	-0.903411	0.942676	
14	-0.26924	0.903411	0.942676	
15	-0.48909	-0.785774	0.925554	
16	-0.48909	0.785774	0.925554	
17	-0.67432	-0.609729	0.909106	
18	-0.67432	0.609729	0.909106	
19	0.791149	-0.445375	0.907896	
20	0.791149	0.445375	0.907896	
21	0.491208	-0.7298	0.879712	
22	0.491208	0.7298	0.879712	
23	-0.79227	-0.360391	0.870388	
24	-0.79227	0.360391	0.870388	
25	0.600454	-0.617216	0.861104	
26	0.600454	0.617216	0.861104	
27	-0.84417	0	0.844174	plant zero #13
28	0.840182	0	0.840182	plant zeros #12
29	-0.67899	-0.244964	0.721825	
30	-0.67899	0.244964	0.721825	

plant zeros #14,15 and 16 not identified

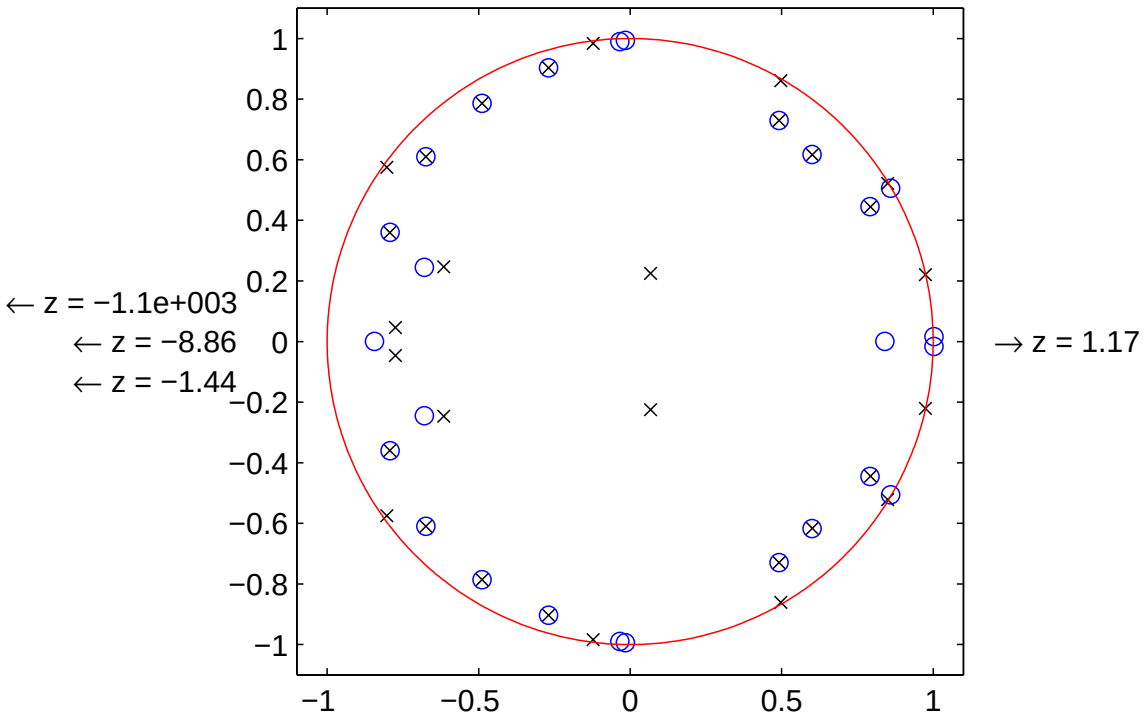


Figure 5.74: Apparent Plant Poles and Zeros

how well our system identification worked. It appears that we have correctly identified those poles that are close to the stability boundary, i.e. the poles that create the modes. We also appear to have correctly identified the non-minimum-phase zeros and the zeros close to mode poles 2 and 4. Three filter zeros and one filter pole are completely missing from the apparent plant.

The apparent plant also includes a number of pole and zero pairs that appear to be almost on top of each other (they are not completely on top of each other, which can be confirmed by examining Table 5.7). Since these pairs do not have the exact same location, they model the disturbance information.

A comparison of the numerical positions in Tables 5.6 and 5.7 confirms the preliminary conclusion of the visual comparison. The magnitude of the error vector for the pole locations is shown in the following table:

First-Mode Poles	5.09E-6
Second-Mode Poles	1.48E-4
Third-Mode Poles	1.82E-6
Fourth-Mode Poles	2.29E-5
Fifth-Mode Poles	9.52E-6
Plant Poles #11,12 (Filter)	4.30E-2
Plant Poles #13,14(Filter)	0.201
Plant Poles #15,16 (Filter)	0.134
Plant Pole #17	not identified

Here we see that the poles that create the modes, poles number 1 through 10 in both Table 5.6 and Table 5.7, are identified very accurately. The largest error for these poles is the 1.48×10^{-4}. The poles that belong to the filter are not identified very accurately, with the largest error of 0.2. The 17th plant pole is not identified at all.

The magnitude of the error vectors for the zero locations are:

Plant Zero #1	6.26E-2
Plant Zero #2	6.66E-5
Plant Zero #3	6.78E-4
Plant Zero #4	1.62E-2
Plant Zero #5	1.62E-2
Plant Zeros #6, 7	4.82E-4
Plant Zeros #8, 9	2.07E-4
Plant Zeros #10,11	1.87E-4
Plant Zeros #12, 13	1.32E-2
Plant Zeros #14, 15, 16	not identified

The largest errors occur for the first zero with an error magnitude of 6.26×10^{-2}. The 14th, 15th, and 16th plant zeros are not identified at all. Clearly the identification of the plant zeros is not as accurate as the identification of the plant mode poles. This combination of high accuracy and low accuracy identification ensures that some of the features of the actual closed response can be predicted, but that others cannot.

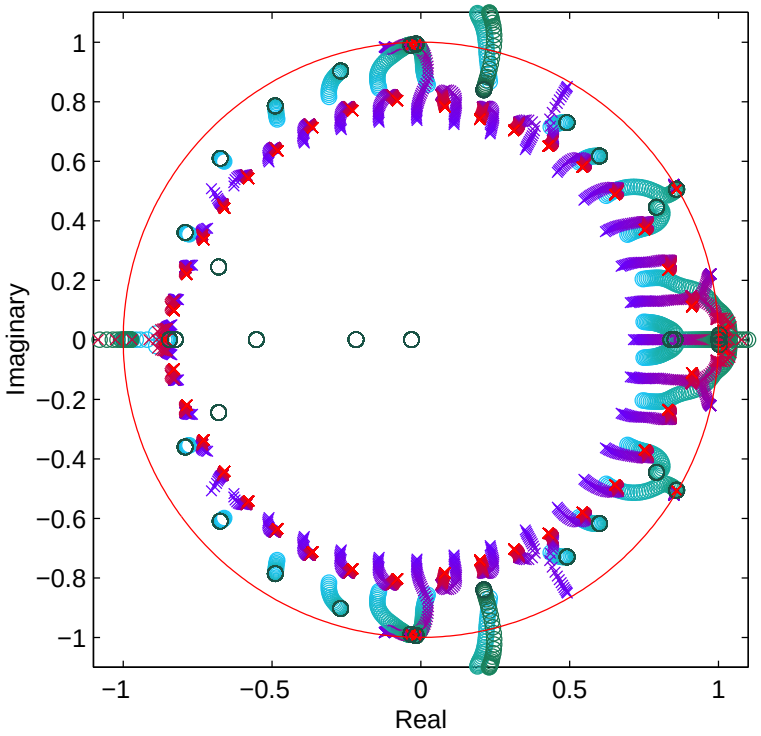


Figure 5.75: Actual Closed-loop Poles

The actual closed-loop pole locations for all iterations are shown in Figure 5.75. The corresponding apparent closed-loop pole locations are shown in Figure 5.76. There are several differences between the two figures. The most obvious difference is the cluster of poles near the origin in Figure 5.76. The cluster of poles is arranged in a circular pattern around the origin. In contrast, while the actual closed-loop poles also show a circle of poles, the diameter of the circle is much larger than in the apparent pole locations.

The second major difference is the movement of the fifth-mode poles. The apparent closed-loop poles show this pole being moved all the way over to a region near the fourth mode. This is not what happens in the actual closed-loop pole locations, however, where the fourth modes poles are moved a small distance toward the circular pattern of poles around the origin.

The third difference is the movement of the third-mode poles. The apparent poles show that these poles are moved a large distance toward $z = 0.85$. Again, this is not what actually happens. Figure 5.75 shows that these poles are moved into the circular pattern around the origin.

We will now examine the details of the final ($\lambda = 0$) closed-loop pole and zero positions. Due to the large numbers of closed-loop poles and zeros, we will not examine the individual numerical

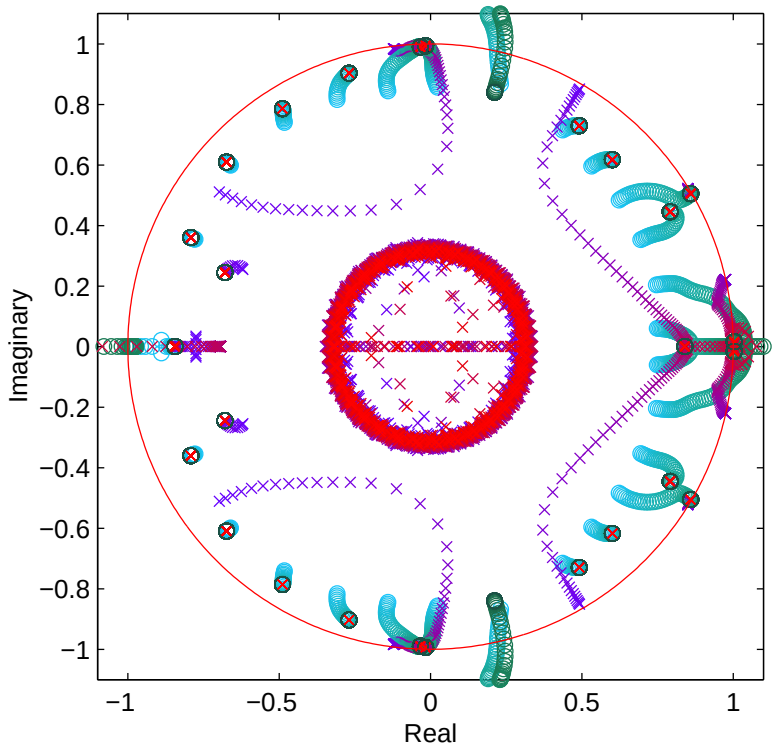


Figure 5.76: Apparent Closed-loop Poles

positions of the $\lambda = 0$ locations . Instead we will adopt a visual approach, showing several diagrams in one figure. In the previous section we saw that the poles and zeros are driven to the apparent plant zero locations for a system that is disturbed by a single tone. We will show that is is true for a system with a broadband disturbance.

In order to visually examine the final positions, we have split the viewing area into seven regions. These detail regions are shown in Figure 5.77, where they are superimposed on the apparent plant. Each detail region was chosen to include an apparent plant zero feature.

The first of the detail regions, labeled D1 in Figure 5.77, is shown in Figure 5.78. We will show five different views of the data in order to aid our understanding. In part (a) of the figure, we will show the poles and zeros of the apparent plant. Part (b) shows the actual closed-loop pole locations for $\lambda = 0$. The corresponding apparent closed-loop poles are shown in part (c). Parts (d) and (e) of the figure show the actual and apparent closed-loop pole and zero locations for all of the iterations.

Part (a) of Figure 5.78 shows that we have four apparent zeros in the viewing area, three of which are outside the stability boundary (see Table 5.7). Parts (c) and (e) show that a pole and zero are driven to each of the zero locations. This indicates that the GPC controller is attempting to

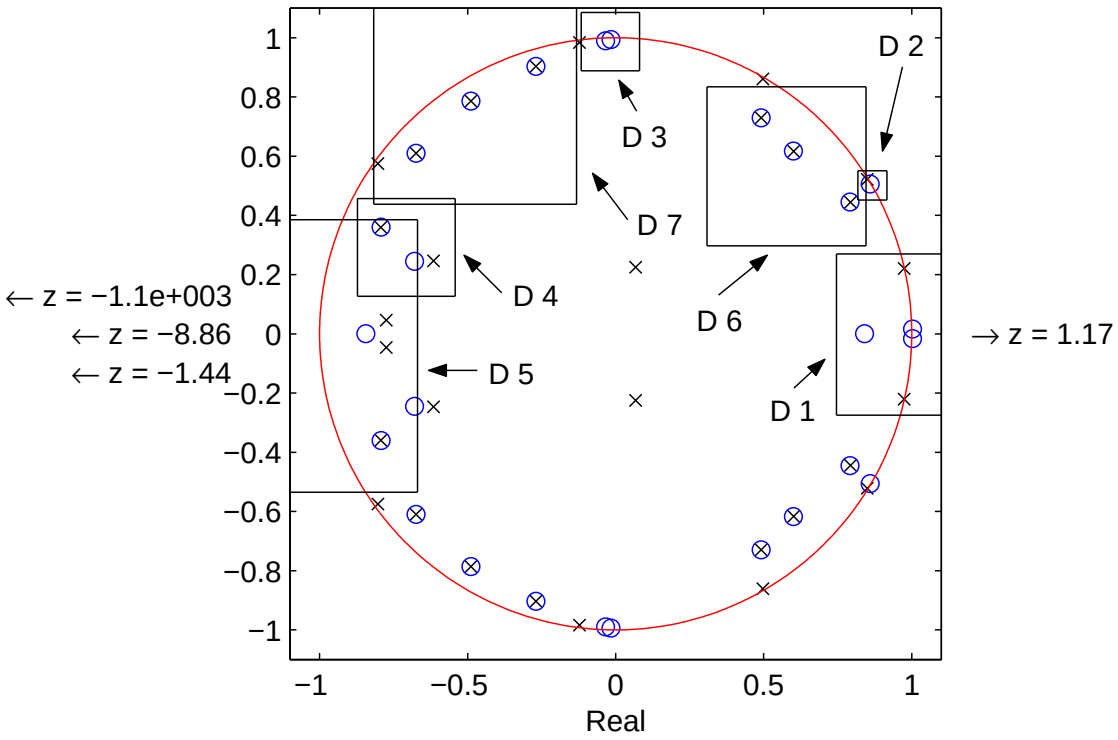
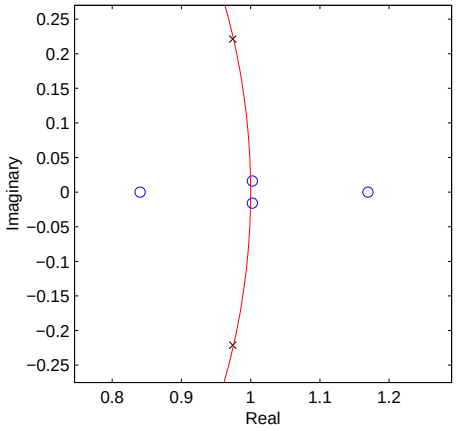


Figure 5.77: Apparent Plant - Detail Locations

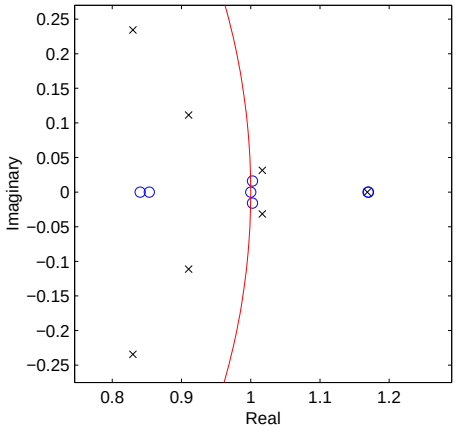
move three poles into unstable positions. Parts (b) and (d) show that the actual poles and zeros are driven close to apparent zero locations. The placement of these poles and zeros is not as accurate as in the apparent closed-loop location of part (c). There are, however, three unstable poles in this detail region.

The details of the second region, labeled D2 in Figure 5.77, are shown in Figure 5.79. This detail region shows the area around the second-mode poles. Part (a) shows that the second-mode pole is very close to a zero. Parts (c) and (e) show that this pole is apparently driven to the zero location. A zero is moved from outside the viewing region to the location of the apparent plant zero. Parts (b) and (d) show that the actual pole and zeros are nearly identical to the apparent pole locations. There is some positional error in the placement of the pole and zero (part (b)). For this detail region, we see that the apparent closed-loop pole and zero locations can be used to accurately predict the position of the actual closed-loop positions.

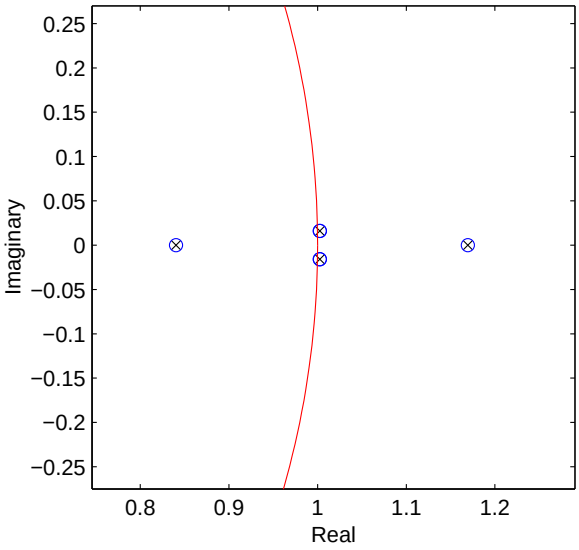
The details of the third region are shown in Figure 5.80. This detail focuses on the zero locations near the third-mode pole. In part (a) of the figure, we see the locations of these zeros. Parts (d) and (e) show that a pole and zero are moved to these zero locations. A comparison to parts (b)



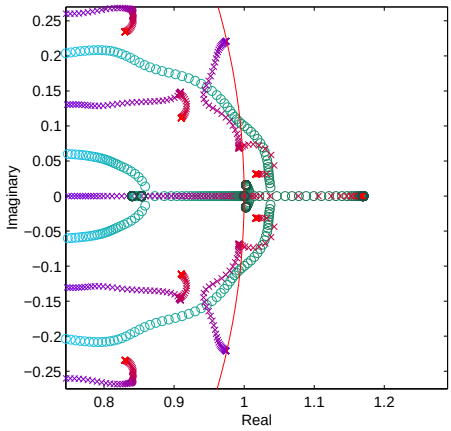
(a) Apparent Plant



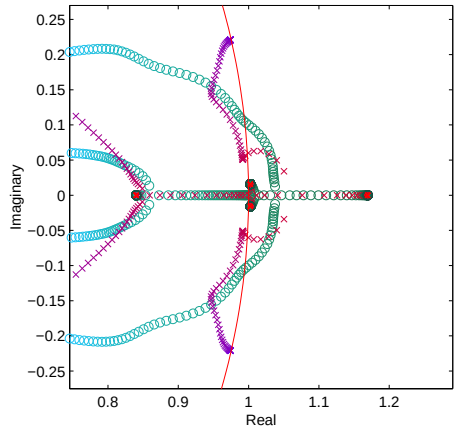
(b) Actual Closed Loop



(c) Apparent Closed Loop

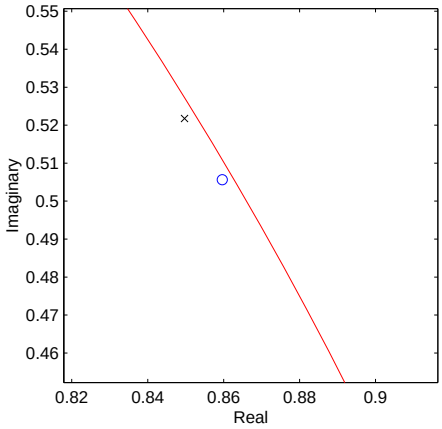


(d) Actual Closed Loop

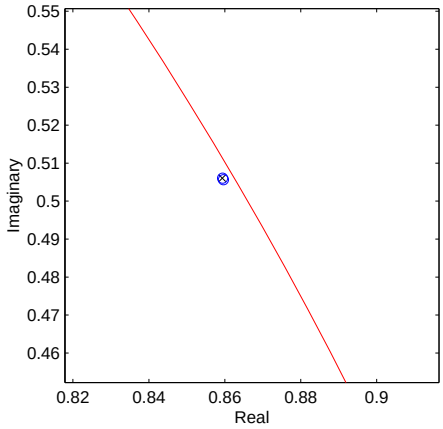


(e) Apparent Closed Loop

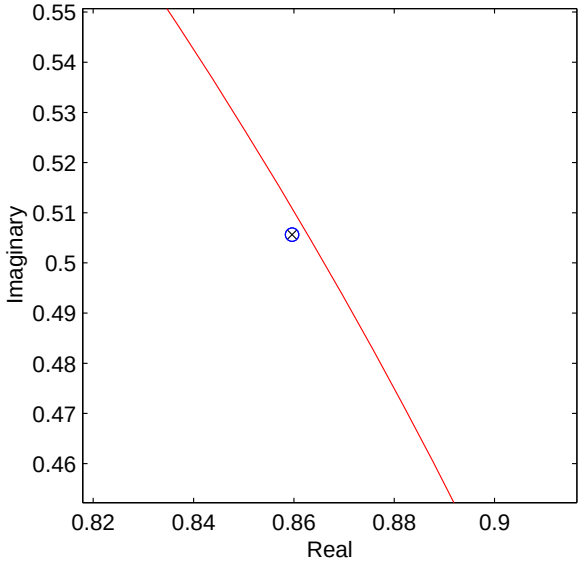
Figure 5.78: $\lambda = 0$ - Detail 1



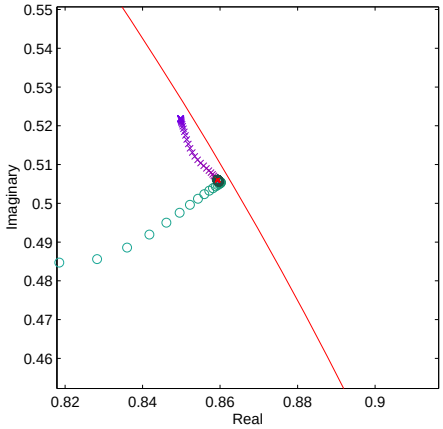
(a) Apparent Plant



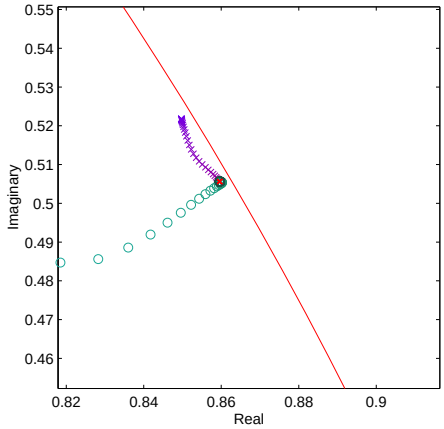
(b) Actual Closed Loop



(c) Apparent Closed Loop



(d) Actual Closed Loop



(e) Apparent Closed Loop

Figure 5.79: $\lambda = 0$ - Detail 2

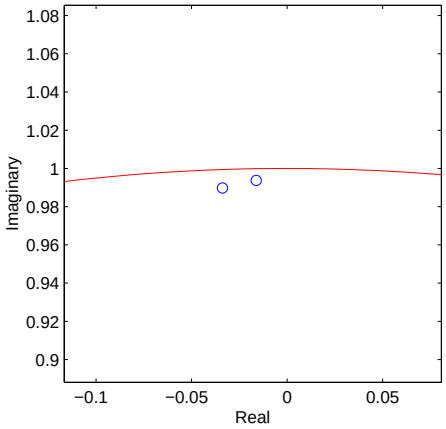
and (c) shows that the placement of the poles and zeros is accurate. The pole coming from the left in parts (d) and (e), is the third-mode pole. The pole coming from the bottom of parts (d) and (e) is different for each case. For the apparent case, the pole is the fifth-mode pole (see Figure 5.76). The pole coming from the bottom for the actual case originates from the circle of poles around the origin (see Figure 5.75).

The details of the fourth region are shown in Figure 5.81. Part (a) shows that there are two apparent zeros in this region. In parts (c) and (e) we see that the GPC controller tries to drive a pole and zero to each of the zero locations. Parts (b) and (d) show that this is not occurring in the actual closed-loop locations. The apparent closed-loop locations therefore do not accurately predict the actual closed-loop locations.

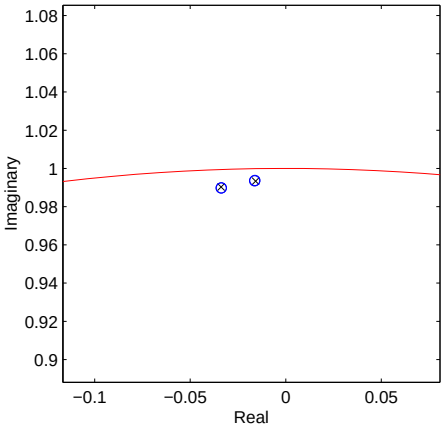
The fifth region is shown in Figure 5.82. The region shows the details of the zero locations close to the Nyquist frequency. The zero at $z = -1.44$ is obviously in the unstable region. Again, we see that GPC attempts to move a pole and zero to each of the apparent plant zero locations (parts (c) and (e)). In parts (b) and (d) we see that only the locations of the real line are somewhat accurately placed, while the remaining poles are not. Most important is that the controller does move a pole into the unstable region. Also, note that the zeros at $z = -1095$ and $z = -8.86$ do not attract a pole or zero. The reason for this is unknown.

In Figures 5.83 and 5.84, we have shown the details of regions six and seven. Both regions show apparent poles and zeros which model the disturbance information. Again, we see that the GPC controller attempts to move a pole and zero to the apparent zero location. This is equivalent to the behavior we saw in the previous section. The actual closed-loop pole and zero locations are much different than those of the apparent locations. We don't understand the importance of this fact, since GPC still achieves disturbance rejection.

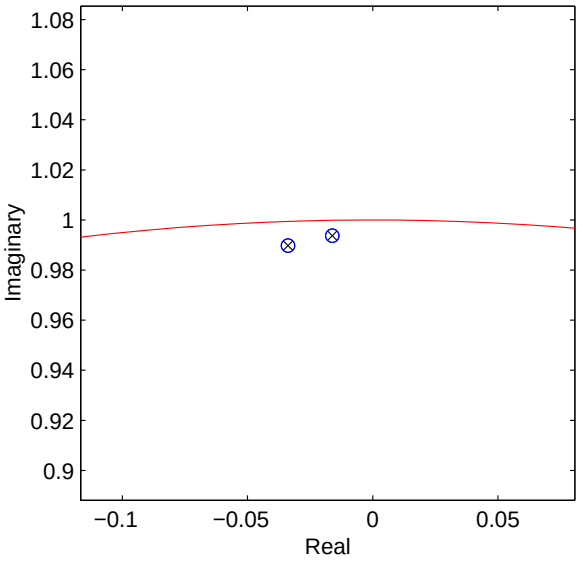
We have shown that the conclusion from the previous section is valid for both a single-tone disturbance and a system with a broadband disturbance. Specifically, we have shown that GPC moves both a pole and zero to the apparent zero locations.



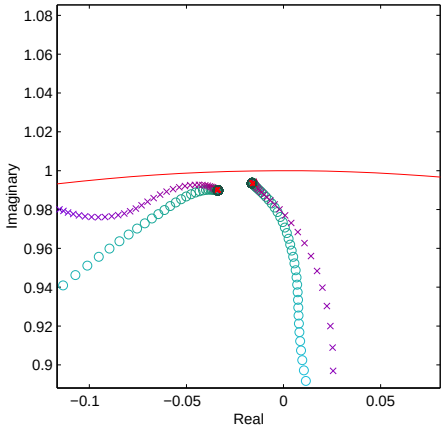
(a) Apparent Plant



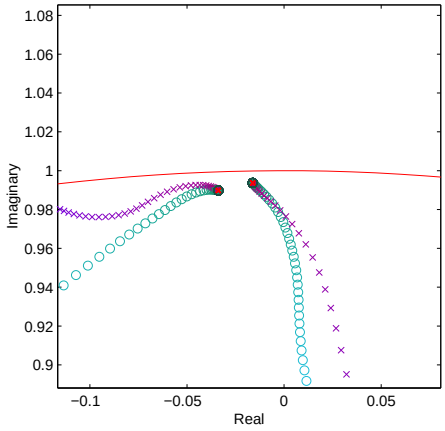
(b) Actual Closed Loop



(c) Apparent Closed Loop

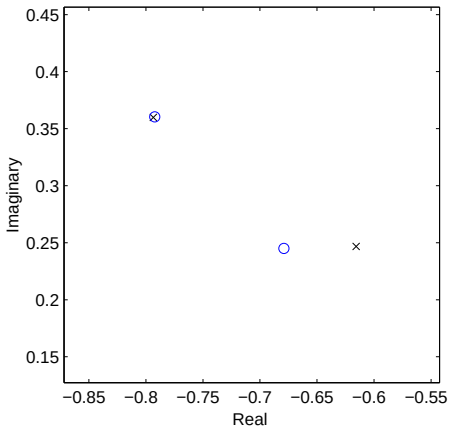


(d) Actual Closed Loop

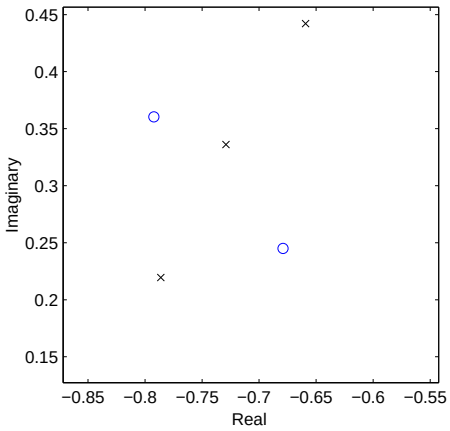


(e) Apparent Closed Loop

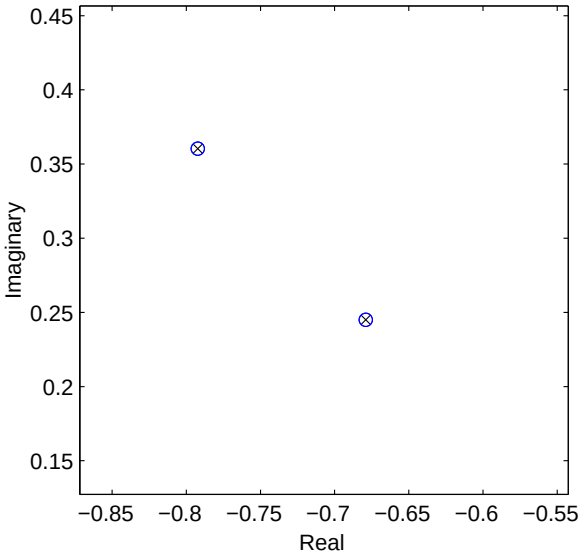
Figure 5.80: $\lambda = 0$ - Detail 3



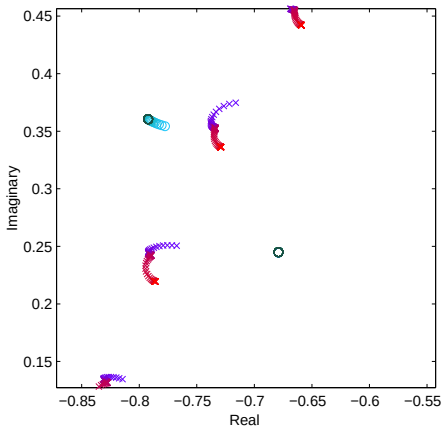
(a) Apparent Plant



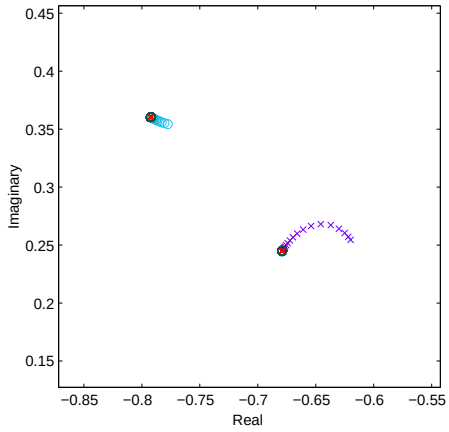
(b) Actual Closed Loop



(c) Apparent Closed Loop

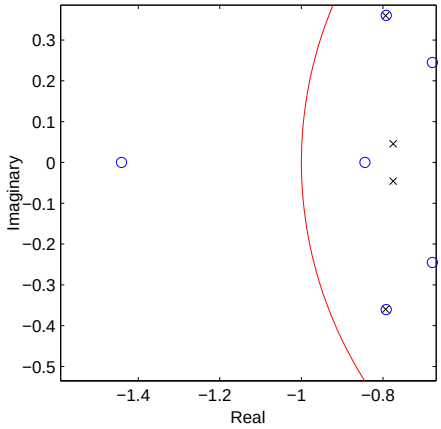


(d) Actual Closed Loop

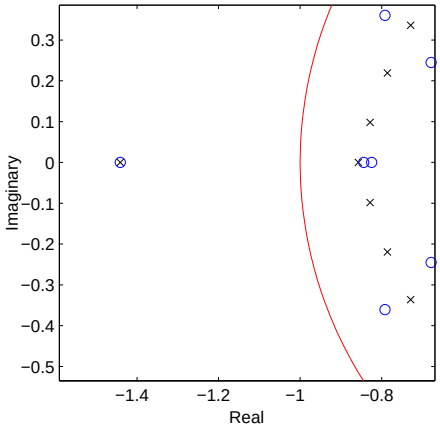


(e) Apparent Closed Loop

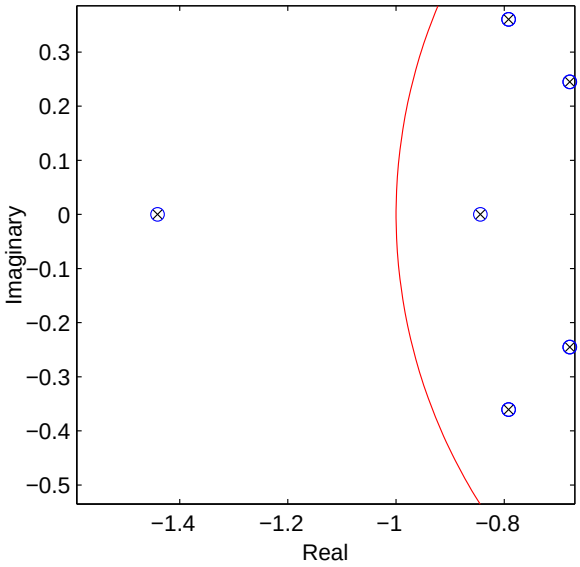
Figure 5.81: $\lambda = 0$ - Detail 4



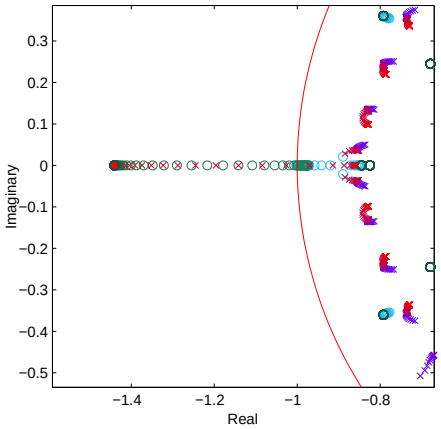
(a) Apparent Plant



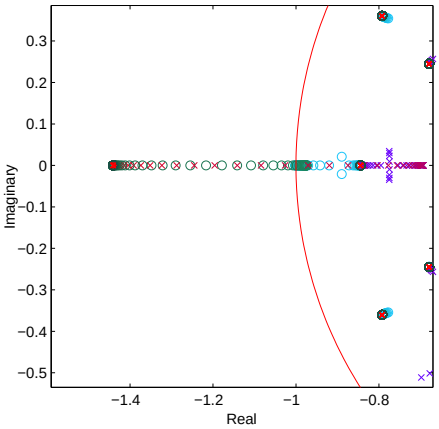
(b) Actual Closed Loop



(c) Apparent Closed Loop

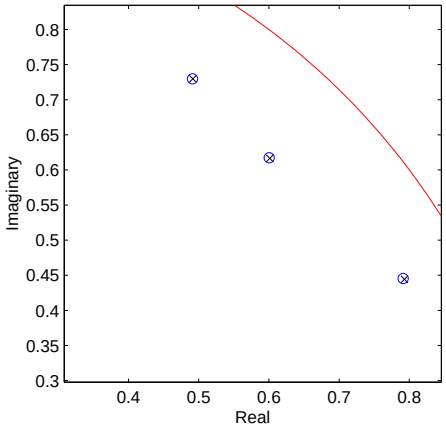


(d) Actual Closed Loop

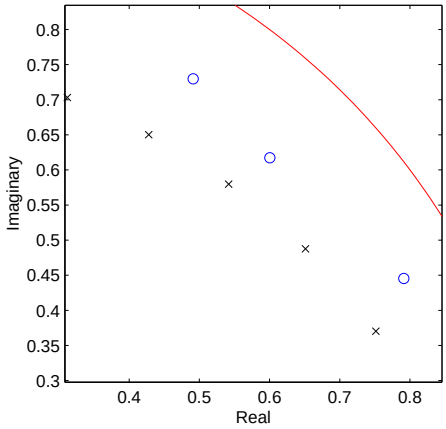


(e) Apparent Closed Loop

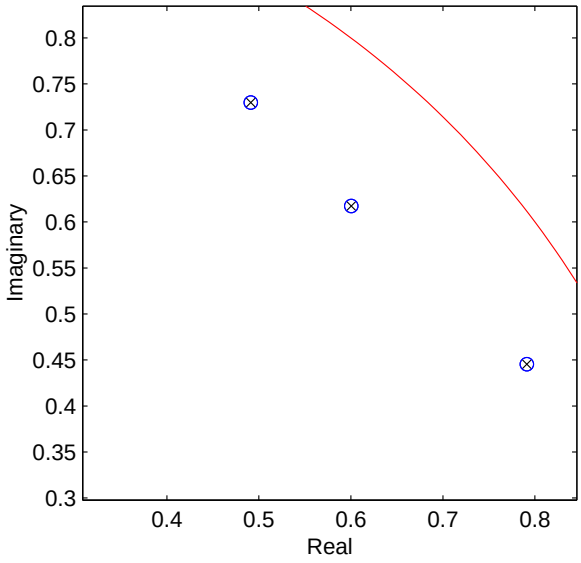
Figure 5.82: $\lambda = 0$ - Detail 5



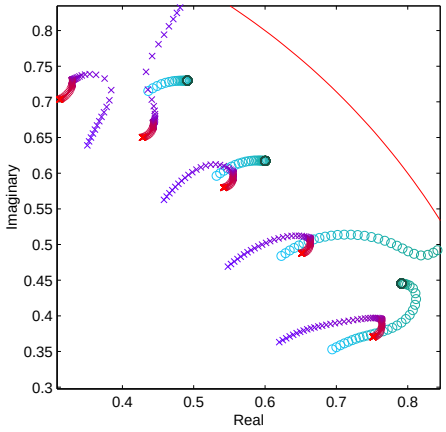
(a) Apparent Plant



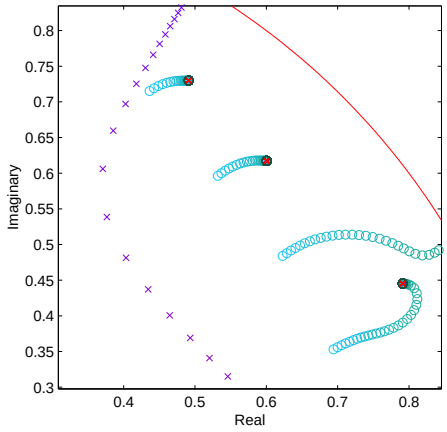
(b) Actual Closed Loop



(c) Apparent Closed Loop

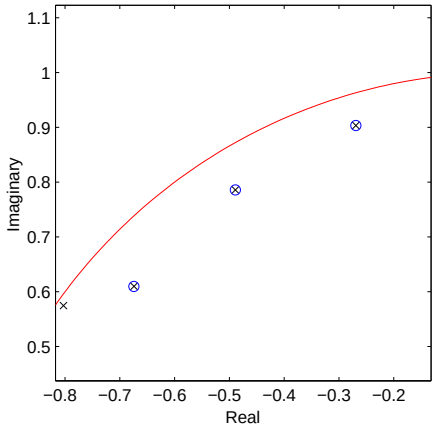


(d) Actual Closed Loop

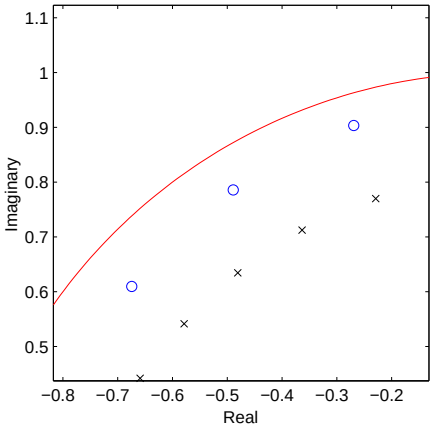


(e) Apparent Closed Loop

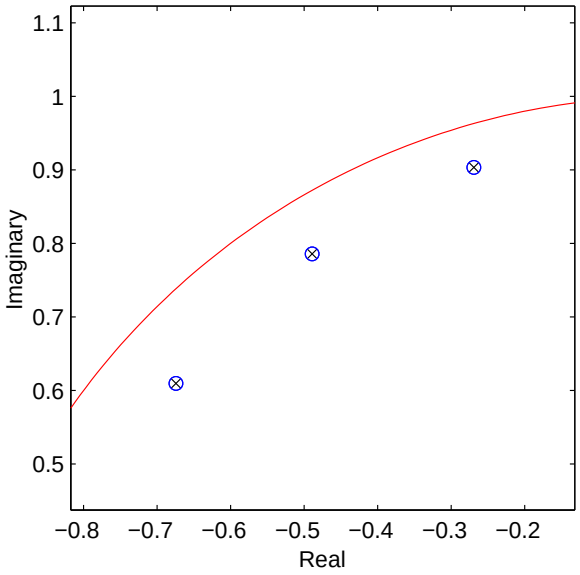
Figure 5.83: $\lambda = 0$ - Detail 6



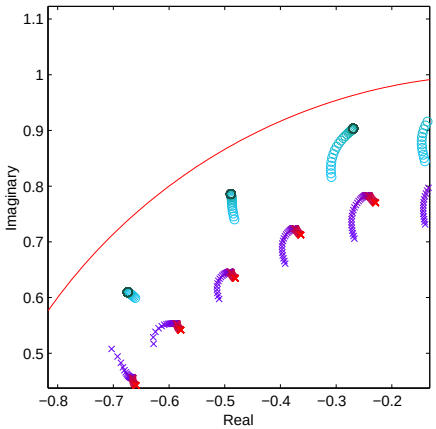
(a) Apparent Plant



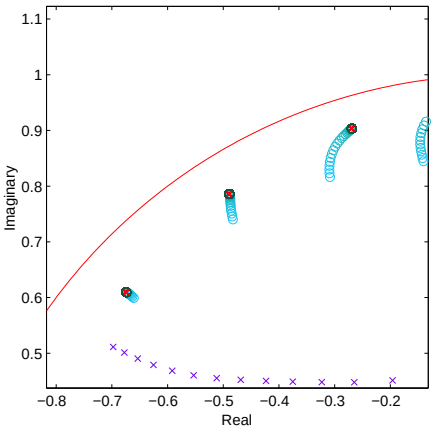
(b) Actual Closed Loop



(c) Apparent Closed Loop



(d) Actual Closed Loop



(e) Apparent Closed Loop

Figure 5.84: $\lambda = 0$ - Detail 7

Chapter 6

Fuzzy Logic Adaptation of λ and Order

In Section 5.2.3, we presented an overview of the observed behavior of the performance and effort curves. It was shown that the behavior could be fit into several distinct groups. We also discussed how the adaptation rules must take into account each of these groups.

Each group can be described in simple linguistic terms, as well as represented by a simple diagram. Since fuzzy logic is based on linguistic terms and creates a decision surface based on these terms, it becomes a natural fit for our problem. Although a fuzzy logic system has a simple description, the decision surface it creates can become quite complex and highly non-linear. Moreover, fuzzy logic rules that independently monitor different conditions can easily be combined into a system that decides on all factors simultaneously.

We have used these characteristics to our advantage in creating a rule system to adapt the GPC parameters. We have built several fuzzy logic modules, each of which monitor a specific condition, e.g. the control effort relative to allowable maximum. A fourth fuzzy logic module then combines the output of the other modules and computes a final decision. The overall decision surface is highly non-linear and incorporates many features that were deemed desirable. The fact that such a complex decision surface could be built simply, is one of the primary advantages of using fuzzy logic.

In Section 6.1, we develop the fuzzy logic modules that adapt λ while keeping order constant. Numerical simulation and laboratory experiments were performed and showed that the fuzzy logic system works well under a variety of conditions. Section 6.2 repeats the process for order adaptations.

6.1 λ Adaptation

6.1.1 Rules

In the previous chapter, we explored the relationship between λ and performance. We have seen that the performance will improve as λ is reduced and that this trend continues until one of three conditions occurs. The first of these trends was that the performance curve flattens out so that further reductions in λ do not result in a performance gain. The second significant trend was that the performance curve reached a local minimum and then started to worsen. The third trend was that the performance curve has an overall downward slope until a point of instability is reached.

In the previous chapter, we have also seen the various possible relationships between λ and control effort. We saw that, in general, the control effort rose as λ is reduced, and that the slope of this curve depended on the controller order. However, we saw that the control effort curve sometimes also flattens out.

We want to start our adaptations with a relatively high λ, one that has very little performance and uses a small control effort. As λ is decreased, we will see an improvement in the performance and also an increase in the control effort. Eventually the performance curve will start to follow one of the trends mentioned above.

The goal of this section is to show the system of fuzzy logic rules that was used to adapt λ given these various behaviors. The question of how to adapt λ becomes easier to answer if we instead ask how to stop the iteration. Hence, what we have done is to create a system of rules that decrease λ until one of the stopping conditions is met. We have classified the various stopping conditions into three categories:

1: A reduction in λ results in an unstable system. The next iteration needs to revert λ to a previously stable λ, and then the adaptation is stopped.

2: The performance curve reaches a plateau or minimum such that further reductions in λ do not improve the performance.

3: Reductions in λ lead to a controller design that uses a very large control effort. When this effort becomes too large, we will stop the adaptation.

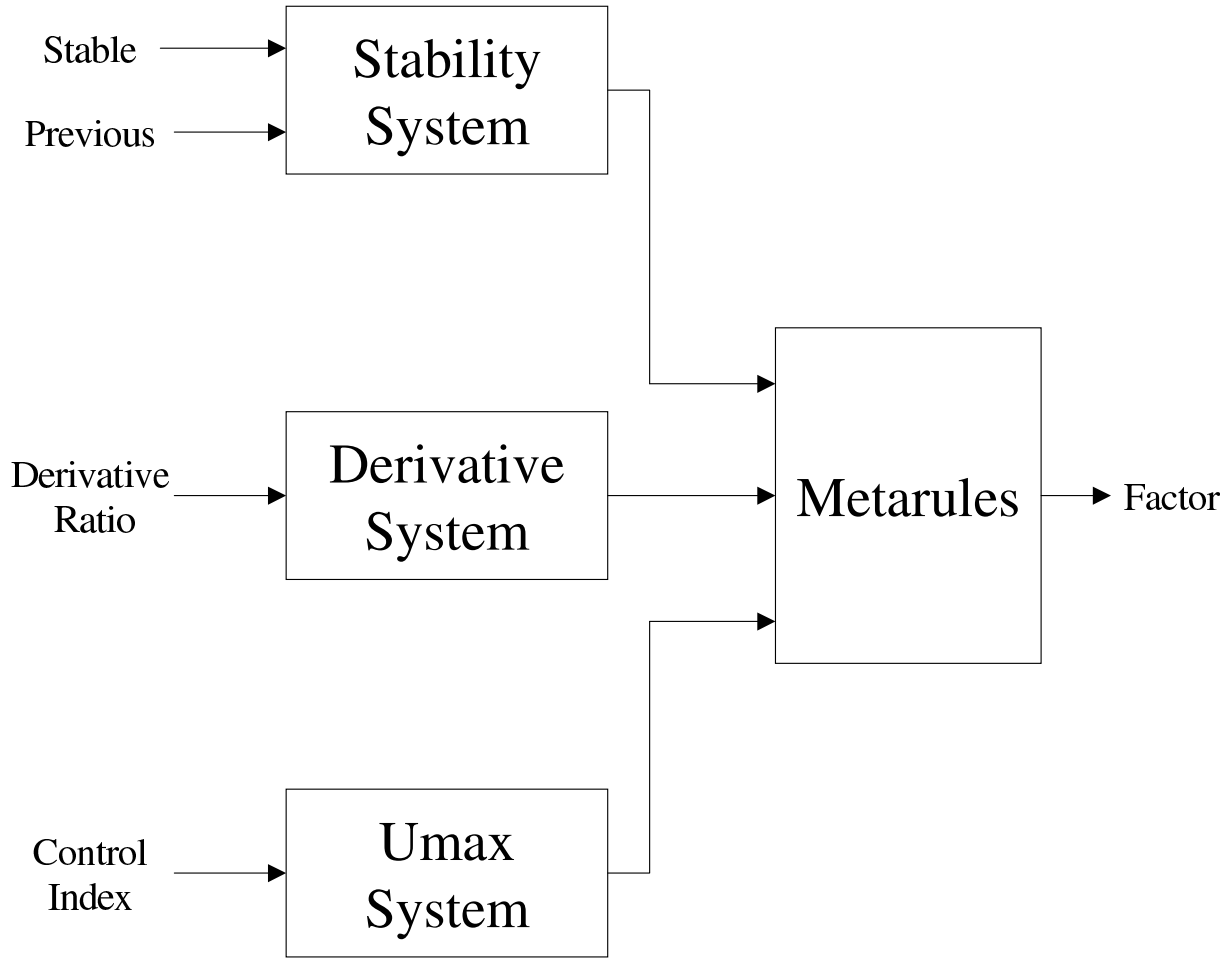


Figure 6.1: Rule Overview

For each of these stopping conditions, we have created a separate fuzzy logic system, each of which analyzes the results of the previous iteration. These systems are called the stability, derivative and U_{max} systems, and are shown in an overall system diagram in Figure 6.1. The output from each of these three modules becomes the input to the metarules module that calculates the final factor. The overall system is a four-input, one-output, four-module fuzzy logic system. The new λ is then calculated by:

$$\lambda_{new} = Factor \times \lambda_{old} \tag{6.1}$$

where *Factor* is the output of the metarules module.

Inherent in this iteration is the range of *Factor*. We have chosen to define this range from 0.25

to 1.1 for normal iterations, and assign a value of 4 for unstable systems. A value of 0.25 will result in the fastest change in λ, resulting in large changes in controller design and the resulting closed-loop system. We have seen that the performance curves can change dramatically for decade (a *Factor* of 10) changes in λ. Making the smallest factor equal to 0.25 gives us over a decade change in λ in two iterations.[1] It was thought that making this minimum factor any smaller would result in changes in the performance curve that were too dramatic, but that it could easily be adjusted.

As the closed-loop response nears the second or third stopping condition, the value of *Factor* is increased so that there are small changes in λ. If one of the stopping conditions is satisfied, then the value of *Factor* becomes 1.0 and the iteration is halted. A description of each module follows.

6.1.1.1 Stability System

It is the job of this system to find and correct instabilities in the closed-loop response. The function of this module is relatively simple: once the adaptation results in an unstable closed-loop system, this module will increase λ to a previously stable controller. If this new higher λ results in a stable system, which it should, then the value of λ is held constant. The iteration of λ is now complete. The fuzzy logic rules of this system are:

1: If (stable is yes) and (previous is iterate) then (action is iterate)

2: If (stable is yes) and (previous is revert) then (action is hold)

3: If (stable is yes) and (previous is hold) then (action is hold)

4: If (stable is no) and (previous is iterate) then (action is revert)

5: If (stable is no) and (previous is hold) then (action is revert)

6: If (stable is no) and (previous is revert) then (action is revert)

This system has two inputs, which are *stable* and *previous*. The system has one output that is called *action*. The inputs and output variables take on crisp values for this system; therefore, this

[1]Given an initial λ of 1.0 ($log_{10}(\lambda) = 0$), two iterations of 0.25 will give us as a $\lambda = 1 \times 0.25 \times 0.25 = 0.0625$. $log_{10}(0.0625) = -1.2$, therefore we have covered over a decade in λ in two iterations.

system could have been implemented with a traditional if-then-else statement. Here we have used the Sugeno-type fuzzy logic inference.

Once a controller is implemented and the loop is closed, we determine whether the closed-loop system stayed stable. This determination becomes the *stable* input and is allowed to be only one of two possible values: a 1 for *yes* (system stayed stable) and a 0 for *no* (system became unstable).

The determination of stability is different in the numerical simulation and the laboratory environment. As we have seen in Section 5.3, the stability of the numerical simulation is determined by examining the radius of the actual closed-loop pole locations. A pole outside the unit circle results is an unstable system. In the laboratory environment, an unstable system will result in a signal that repeatedly hits the maximum or minimum voltage limits of the DSP board. Once this occurs, the DSP board automatically opens the loop. After collecting the closed-loop data, we check to see whether the DSP board is still operating in closed-loop mode. If it is not in closed-loop mode, then we know that the system became unstable.

The output *action* takes one of three distinct linguistic values: *iterate*, *hold*, and *revert*. These linguistic values are also given numerical values of 0.25, 1 and 4. If the action is iterate, then we will continue to reduce λ. If the action is hold, then the metarules module will stop the iteration by assigning $Factor$ in Equation (6.1) a value of 1.0. If the action is revert, then the metarules will assign a value of 4 to *Factor*.[2]

The second input, *previous*, is the value of *action* from the last time the stability module was evaluated. This input is needed to keep track of what the stability system is doing. A short example will illustrate this point.

For a typical adaptation of λ, we will start off with $stable = yes$ and $previous = iterate$. This will cause the first rule to fire, and *action* will be assigned the linguistic value of *iterate*. For illustration purposes, assume that the iteration eventually becomes unstable. This would make $stable = no$ and $previous = iterate$. Rule 4 would therefore fire and *action* would be assigned to *revert*. The metarules would make $Factor = 4$. The next iteration will probably be stable, so rule 2 would be fired and *action* would be assigned a value of *hold*. The metarules will turn this action into a $Factor = 1$. If the system continues to be stable, rule 3 will fire and *action* will be assigned

[2]The value of 4 is somewhat arbitrary. We choose 4 because it is the inverse of 0.25, but we could just as easily made it any value larger than one. The advantage of making it 4 is that we will most likely reduce the time that the system spends in instability.

hold. At this point, the adaptation of λ will have essentially stopped, as rule 3 continues to fire for any ongoing iterations if the system continues to be stable.

In the laboratory environment, the closed-loop system doesn't stay stable indefinitely. We believe this is due to small changes in the plant that make the system identification slightly inaccurate. This inaccuracy, coupled with a high gain controller, results in an instability. This instability would cause rule 5 to fire and λ would be increased again. If this new λ proves stable, then the stability module would once again hold this value.

Rule 6 is included to cover all the possible combinations of *stable* and *previous*. This rule might fire if there is a drastic change in the plant. The rules shown above are those used for the numerical and laboratory experiment. In retrospect rules 4, 5 and 6 could be replaced by a single rule that reads:

4. If (stable is no) then (action is revert)

6.1.1.2 Derivative System

We have seen that some performance curves tend to flatten out as λ is reduced, i.e. further reductions in λ give us negligible or no improvement in performance. It is the task of the derivative system to look for this type of behavior and stop the adaptation of λ when this occurs.

An example of this type of performance curve is shown in Figure 6.2. The top part of the figure shows the performance curve, and the bottom part of the figure shows the derivative of this curve. For this particular situation, it makes no sense to adapt λ to levels below 0.001, as there is virtually no improvement in performance.

When we adapt λ, we get discrete points on the curve, represented by circles in Figure 6.2. To turn these points into an actual curve, we fit a smoothing spline to them.[3] This type of spline is implemented in the Matlab spline toolbox as the function *csaps.* In order to calculate the spline, we needed to choose a smoothing parameter. This parameter has a range from 0 to 1, where a value of 1 results in a spline that goes through every point, and a value of 0 results in a straight-line least-squares fit to the data. We used a value of 0.98 for both the numerical simulation and the laboratory experiments, resulting in a spline fit that nearly goes through every point. The solid line

[3]By applying a spline fit to our data, we are assuming that the performance curve should be smooth. Extensive experimentation showed that performance curves were relatively smooth, for example, the curves shown in Section 5.2.2.

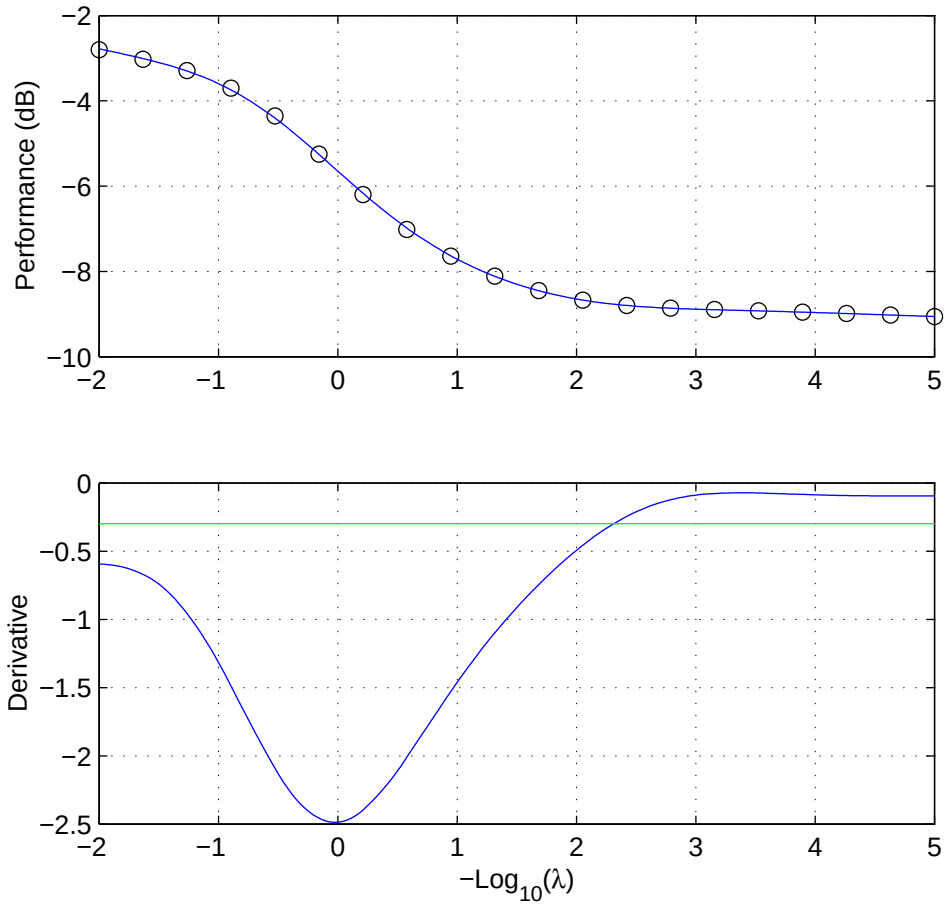


Figure 6.2: Example Performance Curve

shown at the top of Figure 6.2 is this exact spline fit.

In addition, the Matlab toolbox includes the *fnder* function that allows the user to compute the analytic first derivative of any spline. Once the spline and its derivative are calculated, we can evaluate them at an arbitrary number of points. We found that evaluating the spline and the derivatives at 200 points proved more than adequate. This derivative is shown in the bottom half of Figure 6.2. It is the values of this spline derivative that are used by this fuzzy logic system. The spline fit and its derivate are recalculated after each iteration, using all of the available data points.

The spline fit algorithm does not allow the user to input values with the same x coordinates but different y coordinates. This happens when the overall factor equals one, and successive iterations have the same λ but slightly different values of performance. We have chosen to eliminate the duplicate values for the purpose of fitting the spline; only the first point in a set of duplicate points is used. Alternatively, we could have averaged the performance values for points that have the same λ.

The rules for this module are:

1: If (ratio is unity) then (factor is fulldrop)

2: If (ratio is half) then (factor is fastdrop)

3: If (ratio is quarter) then (factor is slowdrop)

4: If (ratio is small) then (factor is nodrop)

Here, again, we used a Sugeno-type fuzzy logic system. The input to this system is defined as

$$ratio = \frac{F'(\lambda)}{F'_{min}},$$

where $F'(\lambda)$is the value of the derivative at the current λ, and F'_{min} is the absolute minimum of the derivate for the available range of λs. The input linguistic variables *unity*, *half*, *quarter*, and *small* are the fuzzy sets shown in Figure 6.3. The output linguistic variables *fulldrop*, *fastdrop*, *slowdrop*, and *nodrop* have been assigned the discrete values of 0.25, 0.3, 0.75, and 1. If this derivative system were acting on its own, then the output of this system would be the *Factor* of Equation (6.1). We will therefore refer to the output of the derivative system as the *derivative factor*.

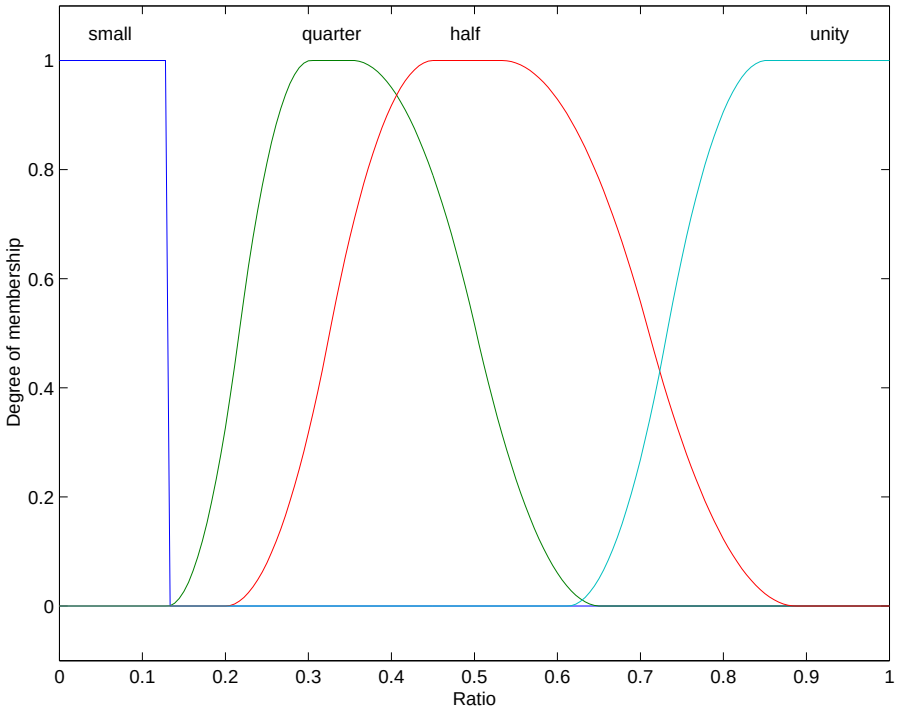


Figure 6.3: λ Derivative System Membership Functions

As was stated in Section 2.3 , the fuzzy logic process results in an input-output mapping, which for this system is shown in Figure 6.4. There are several features of this curve that are important, and they will be explained in the following adaptation example.

For this example, we will assume that the performance curve of our system is that of Figure 6.2. We will start with λ of 100. As the adaptation progresses, we spline fit the available data and calculate the input *ratio*. This ratio will be one until $\lambda = 1$ because the current derivative will be equal to the minimum derivative. Referring to Figure 6.4, this means that we will get a factor of 0.25. As λ is reduced below 1, the current derivative will no longer be the minimum derivative and our ratio will be slightly less than one. We therefore move up and to the left on the curve in Figure 6.4 and get a factor greater than 0.25. As we continue to adapt λ, we will continue to get smaller and smaller ratios and therefore get larger factors. In essence, the derivative system slows down the rate of adaptation once the current λ moves past the λ that gave us F'_{min}. Eventually, our ratio will fall below 0.12 and we will get a factor of one, stopping the adaptation of λ. This 0.12 level is shown as the green line in the bottom half of Figure 6.2.

There are three important features of the curve in Figure 6.4. The most obvious is that we have

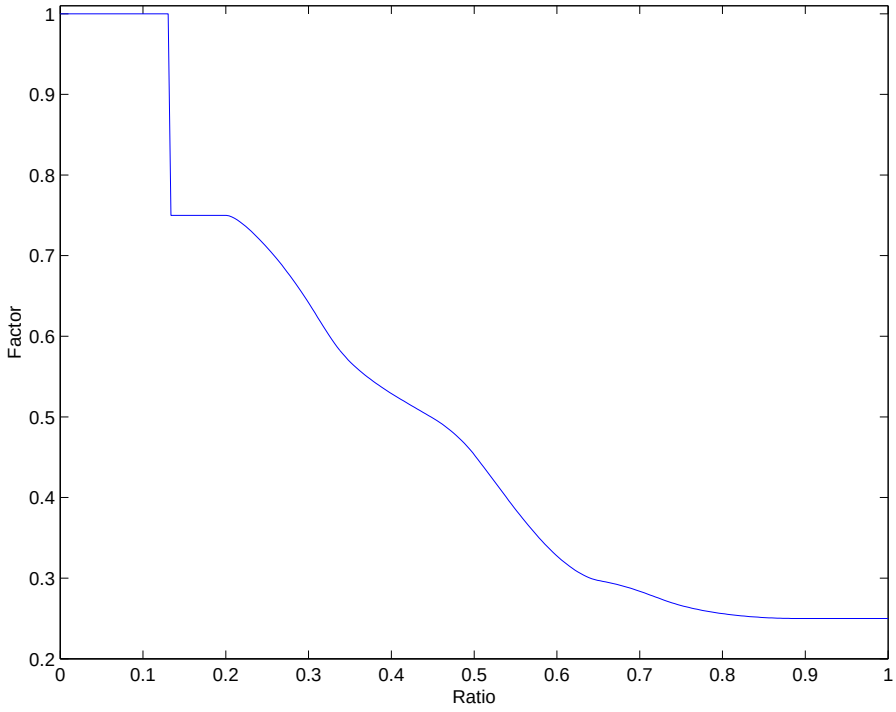


Figure 6.4: Derivative System

chosen to stop the iteration if the derivative ratio falls below 0.12. This value is somewhat arbitrary and could be easily adjusted if desired. In the example, the adaptation would have stopped at about $10^{-2.5}$, which in our opinion is about where it should stop.

The second important feature of the curve is the near step change from a factor of 0.75 to 1. It was found in early experiments that it was undesirable to allow the factor to achieve values between 0.9 and 1.0. These values would result in adaptation that would make smaller and smaller changes in λ so that it would approach the desired value of λ asymptotically without ever reaching the final value. By not allowing these very small changes in λ, it was found that adaptation would reach a final value, hence the inclusion of the step change. We found that the largest allowable factor could be any number between 0.75 and 0.85, and here we have used 0.75.

The third feature of the curve is the overall shape between the ratio values 0.12 and 1.0. Note that the curve stays near 0.25 for a small range of ratios near 1, and then gradually curves upward to a factor of 0.75. This creates a curve whose concavity is pointing upward. It was found that this general shape was more desirable than one whose concavity points downward, i.e. a drastic upturn from a factor of 0.25 to higher factors. This alternate shape would result in adaptations that would

start to slow down the rate of adaptation too soon and therefore use too many iterations to get to the final value. The wavy nature of the curve is not important and is a result of using only four rules to form this curve.

Since this module depends on a curve fit to work properly, it is not evaluated until we get to the third iteration. A default output value of 0.25 is used for the first two iterations. It is also important to note that the input is only defined to be between 0 and 1. If a performance curve starts to turn upward, giving us a small positive slope and a negative *ratio*, then *ratio* will be reassigned to 0 and the adaptation will stop.

6.1.1.3 U_{max} System

As λ is reduced, the control effort has a tendency to increase. We allow the user to pick a ceiling for this control effort that we will call U_{max}. It is the task of this module to keep the control effort below U_{max}, and to stop the adaptation once we get close to it. If the control effort goes above this value, then this module will increase λ until the effort reaches the acceptable level.

The rules of the U_{max} system are as follows:

1: If (ratio is zero) then (factor is fulldrop)

2: If (ratio is quarter) then (factor is halfdrop)

3: If (ratio is half) then (factor is smalldrop)

4: If (ratio is threequarters) then (factor is stop)

5: If (ratio is unity) then (factor is increase)

The input ratio is defined as:

$$ratio = \frac{U_{current}}{U_{max}},$$

where $U_{current}$ is the control effort for the last iteration. This ratio will also be referred to as the control index (CI). After a controller is implemented, we collect a sample of the closed-loop data. In the laboratory environment, we typically collect between 30,000 and 80,000 samples (10 to 26 seconds). $U_{current}$ is the maximum of the absolute value of the control effort time data. This is a more crude method than the one used in the adaptation of order, but it nevertheless seemed to work

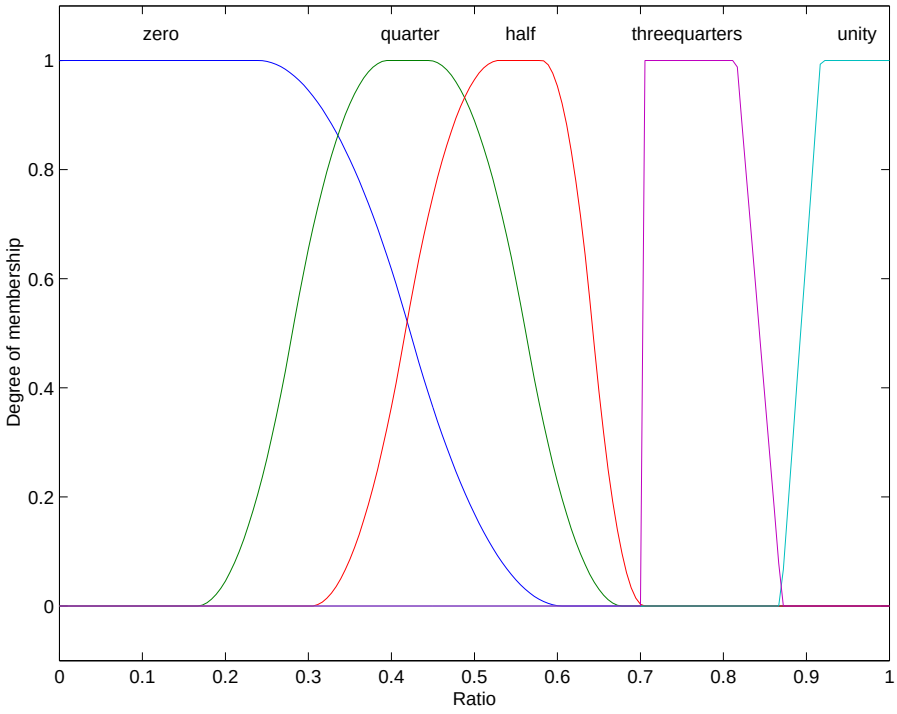


Figure 6.5: U_{max} System Membership Functions

reasonably well. The linguistic variables *zero*, *quarter*, *half*, *threequarters* and *unity* are fuzzy sets which roughly represent their names. Their membership functions are shown in Figure 6.5. The output linguistic variables *fulldrop*, *halfdrop*, *smalldrop*, *stop*, and *increase* are given the fixed values of 0.25, 0.55, 0.85, 1.0, and 1.1. If this U_{max} system were acting on its own, then the output of this system would be the *Factor* of Equation (6.1). We will therefore refer to the output of this system as the U_{max} factor.

The input-output mapping for this system is shown in Figure 6.6. We have again built several important features into this curve, which will be introduced using a short example.

In general, an adaptation of λ will begin with a high λ and will use very little control effort. This would make *ratio* very small, firing rule 1 and giving us a factor near 0.25. As λ is decreased, the controller would use more and more effort, thereby causing *ratio* to increase. This in turn will cause us to move to the right in Figure 6.6, and the output factor will start to increase. Hence, the adaptation rate slows down as the controller uses more and more of the allowable control effort U_{max}. If the control effort continues to rise as λ is decreased, the adaptation will eventually reach a *ratio* that would give an output factor of 1. At this time, the adaptation of λ will have halted.

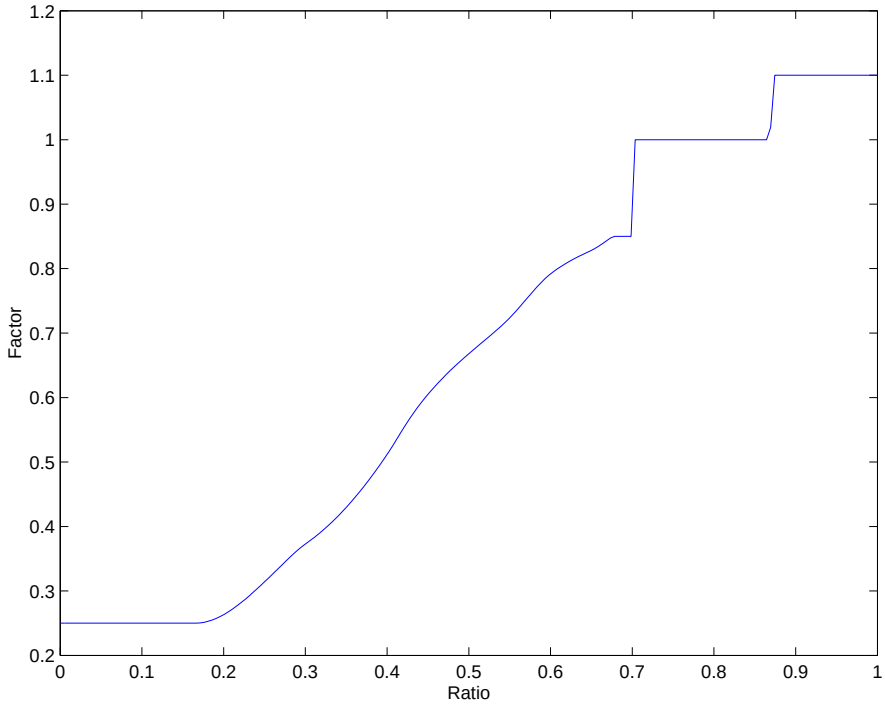


Figure 6.6: U_{max} System

The features that we have built into this input-output mapping are similar to those of the derivative system described in Section 6.1.1.2. The most important feature is that we allow the adaptation to stop, i.e. $factor = 1$, for a range of *ratios* of about 0.70 to 0.87. The goal, therefore, is not to use all of the allowable control effort. If the *ratio* is larger than 0.87, the U_{max} system will increase λ by 10 percent until the *ratio* is within the allowable range of 0.70 to 0.87. We have also built a step into this curve so that the output factor will not be between 0.85 and 1.0. This was done for the same reason as in the derivative system.[4] The last important feature is the gradual upturn of the curve from a factor of 0.25 to 0.85, again based on the same reasoning as in the derivative system: to limit the number of iterations until stoppage.

6.1.1.4 Metarules System

The metarules take the outputs of the first three systems and calculates the final *Factor* of Equation (6.1). The system has three important functions. The first function is that if the stability module indicates that the system has become unstable, the metarules system will allow the stability module

[4]Note that the undesirable range, 0.85 to 1.0, is slightly different than that used in the derivative system (0.75-1.0). This makes only a small difference, but it is nevertheless corrected by the metarules.

to take over the adaptation of λ. Once the system becomes unstable, it is no longer important what the U_{max} and derivative systems want; the first priority must be to stabilize the system. Of course, the adaptation will only reach an unstable point if it hasn't been stopped by the U_{max} and derivative systems prior to that point.

The second function is that if the output *action* from the stability system is *iterate*, then the output from the metarules should be an intelligent combination of the derivative and U_{max} factors. For example, if the U_{max} factor is 0.75, a small drop, and the derivative factor is 0.25, the largest drop possible, then the metarules should recognize that the U_{max} module is getting ready to stop the adaptation and make the final factor equal to 0.75. Also, if either the derivative module or the U_{max} module wants to stop the adaptation, then the metarules system should allow this stoppage.

The third function of the metarules will be to allow the derivative module to stop the iteration only if the control index[5] is larger than 0.3, i.e. we are using more than 30% of our allowable control effort. This exception is included for the following reason. We recognize that if the derivative module wants to stop the adaptation, we could still be getting some performance benefit for decreases in λ. In fact, we could be getting as much as 12% of the maximum performance rate F'_{min}. Since we are using less than 30% of our allowable control effort, the metarules will allow a small drop in λ to occur in order to gain a small performance benefit.

The rules of the metarules system therefore are:

1: If (action is hold) then (factor is nodrop)

2: If (action is revert) then (factor is revert)

[5]The control index (CI) is the input of the U_{max} system.

3: If (action is iterate) and

Derivative Factor \ Umax Factor	SFD	SLD	SMD	FD	LD	MD	SD	ND	I
ND	SM	SM	ND				ND	ND	I
SD				SD	SD	SD	SD	ND	I
MD				MD	MD	MD	SD	ND	I
LD				LD	LD	MD	SD	ND	I
FD				FD	LD	MD	SD	ND	I

ND = no drop, SD = small drop, MD = medium drop, LD = large drop, FD = full drop
SFD = sharp full drop, SLD = sharp large drop , SMD = sharp medium drop
SFD, SLD, and SMD are modified versions of FD, LD, and MD

The input *action* is the output of the stability module. Each table entry corresponds to a separate rule. For example:

if (action is iterate) and (derivative factor is mediumdrop) and (umax factor is smalldrop) then (factor is smalldrop)

The metarules system therefore has a total of 32 rules. Since this is a Sugeno fuzzy logic system, the input linguistic variables are fuzzy sets, and the output linguistic variable are crisp numbers. The membership functions are shown in Appendix B. The output variables *full drop*, *large drop*, *medium drop*, *small drop*, *nodrop*, *increase*, and *revert* are given the values of 0.25, 0.45, 0.65, 0.75, 1, 1.1 and 4. The rules that involve the U_{max} variables *sharp full drop*, *sharp large drop*, and *sharp medium drop* are used to create the previously described third function of the metarules.

If *action* is iterate, then the table of rules defines a three-dimensional surface that is shown in Figure 6.7. It is important to note that the inputs are factors, i.e. outputs of the U_{max} and derivative systems. The large L-shaped red surface is at the height of $Factor = 1$ and therefore represents combinations of U_{max} and derivative factors at which the adaptation of λ will stop. We have again built a sharp step into the surface so that the range of allowable *Factors* does not include $0.75 \leq Factor < 1.0$ for the same reason as described previously. We can also make a three dimensional surface that goes from the inputs of the U_{max} and derivative systems to the output of the metarules system. This is shown in Figure 6.8. This surface includes the input-output

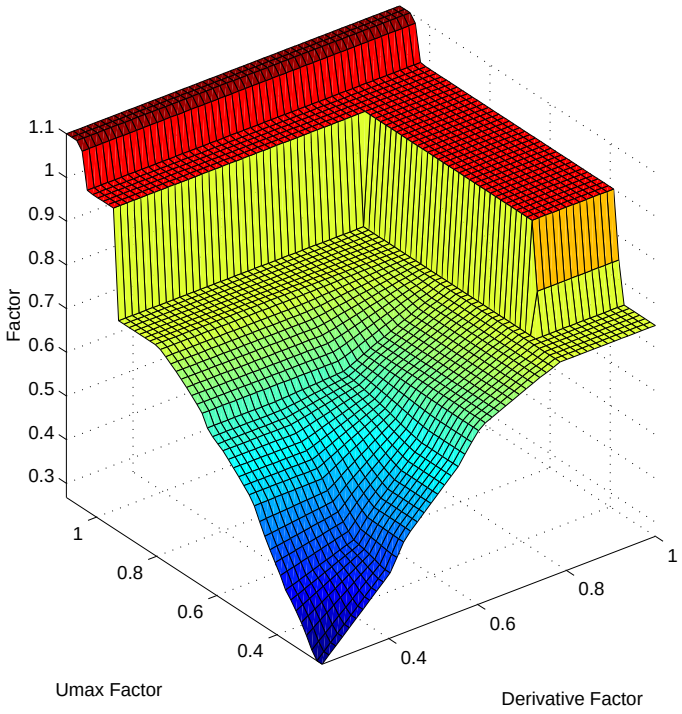


Figure 6.7: Decision Surface

mappings of U_{max} and the derivative modules, as well as the three-dimensional surface of the metarules mapping. A contour plot of this surface is shown for reference in Figure 6.9.

A normal adaptation of λ will start in the blue region and then go up the surface of Figure 6.7. As the adaptation continues, the operating point will, in general, move to the light blue and then the yellow regions. If the system stays stable, then the adaptation will eventually reach the stopping condition, which is the red L-shaped part of the surface. The exact path up the surface depends on the characteristics of the actual performance and control effort curves for the system in question. If the closed-loop system becomes unstable during the iteration, the stability module takes over any further adaptation of λ, and only the first two rules of the metarules system will fire.

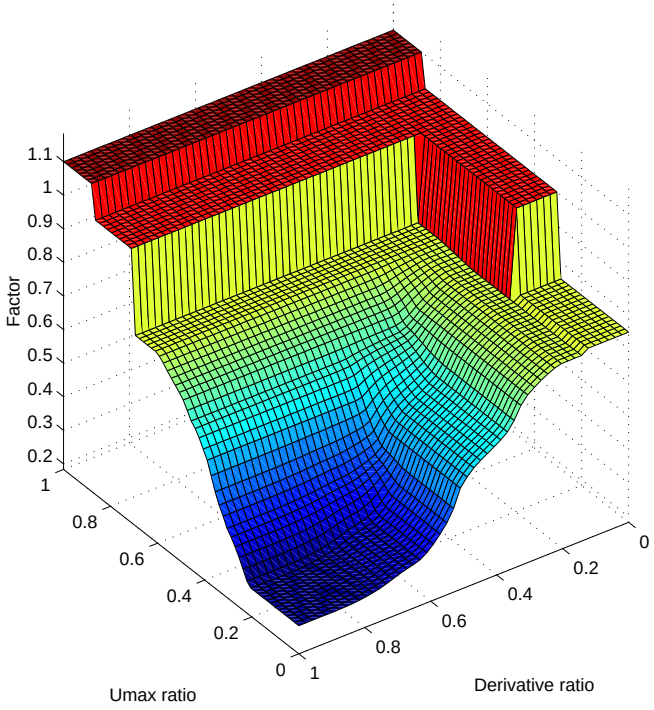


Figure 6.8: Decision Surface with Ratio Inputs

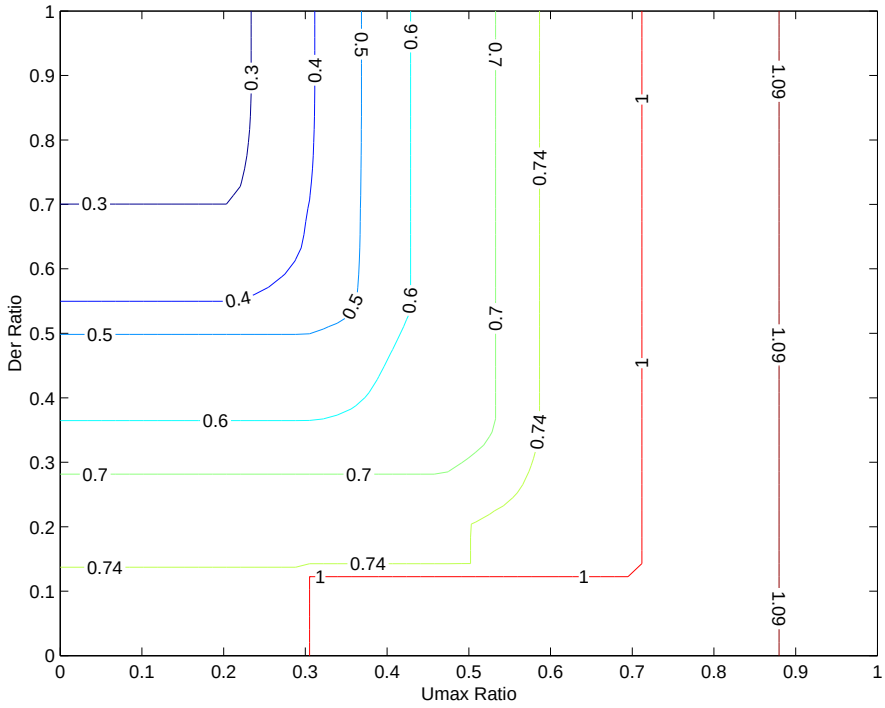


Figure 6.9: Decision Surface Contour Plot

6.1.2 Numerical Results

We will present three cases that are designed to show how the various fuzzy logic systems work. The first case will show how the derivative system stops the adaptation. The second case will restrict the U_{max} of the first case, thereby causing the U_{max} system to stop the adaptation. Therefore, the second case will be a subset of the first case. For the third case, we will show a closed-loop system that becomes unstable and the stability system takes over the iteration.

We will show the results in a variety of forms. First, we will show the results of the adaptation in tabular form. We included these tables for completeness, and recognize that examining a table of numbers is difficult. We will therefore show the various factors in two- and three-dimensional plots that will highlight which system controls the adaptation. Most importantly, of course, are the performance curves, the λ curves and control effort curves. We will show these curves both as a function of $log_{10}(\lambda)$ and the iteration number.

6.1.2.1 Case 1

For this case, we have run the fuzzy logic adaptation on the 28-mode system, with the accelerometer measurement at location 3. We have fixed the order at 80, and the control and prediction horizons at 300. We picked an overly large U_{max} of 1000 V so that the U_{max} system would not control the iteration. We have collected 50,000 samples for each iteration. The results of the adaptation are shown in Table 6.1.

Here we have used various abbreviations in the interest of space. CI (control index) is the input to the U_{max} system[6]. Umax-F is the output of the U_{max} system, and Der-F is the output of the derivative system. Stability F is the output of the stability system, and Factor is the output of the metarules system. The table shows that the derivative factor goes to 1 after iteration 14, and the adaptation is halted.

The performance curve and control effort curves are shown as functions of $log_{10}(\lambda)$ in Figure 6.10. λ and control effort are shown as functions of iteration in Figure 6.11. Each iteration is shown as a diamond. The left axis shows the scale of the blue performance curve, and the right axis shows the scale of the green control effort curve.

[6]control index = $\frac{U_{current}}{U_{max}}$

Table 6.1: Case 1 - Iteration Results

Iteration	CI	Umax-F	Der-F	Stability F	Factor	λ
1	0.00758	0.25	NaN	0.25	0.267	100
2	0.0173	0.25	NaN	0.25	0.267	26.75
3	0.0338	0.25	0.25	0.25	0.267	7.154
4	0.0576	0.25	0.255	0.25	0.272	1.914
5	0.0904	0.25	0.277	0.25	0.294	0.5203
6	0.127	0.25	0.275	0.25	0.291	0.1528
7	0.167	0.25	0.466	0.25	0.511	0.0445
8	0.21	0.271	0.566	0.25	0.614	0.02273
9	0.244	0.307	0.682	0.25	0.706	0.01396
10	0.27	0.339	0.733	0.25	0.729	0.009854
11	0.294	0.366	0.75	0.25	0.736	0.007183
12	0.316	0.388	0.75	0.25	0.736	0.005288
13	0.338	0.414	0.75	0.25	0.736	0.003892
14	0.359	0.442	1	0.25	1	0.002865
15	0.359	0.442	1	0.25	1	0.002865

In Figure 6.10, the overall factor, i.e. the output of the metarules system, is reflected in the horizontal distance between the points. For this adaptation, we see that there are large steps (Factor =0.267) for the initial iterations and a very small horizontal difference in the later iterations as Factor is increased to 0.736. This behavior is exactly what we set out to do when we designed the derivative rules.

In Figure 6.11, we are looking at the overall shape of the λ curve to find a relatively smooth transition from the initial slope to a slope of zero as the adaptation progresses to the stopping condition. In this figure, we see that this slope transition is indeed relatively smooth, but we do see a slight kink from the 6th to 7th iteration.

It is much easier to see the factors in a two- or three-dimensional form. In Figure 6.12, we show the four factors as a function of iteration. Here we see that the metarules factor closely follows the derivative factor. Therefore, it is the derivative system that controls this adaptation. The derivative factor rises smoothly from the 6th iteration to the 10th iteration, a reflection of the gradual slope change in the performance curve. This gradual change in the factor is also shown in the smooth change in the slope of the λ curve (Figure 6.11). The derivative system then selects a factor of 0.75 for the 11th through the 13th iterations, as the slope of the performance curve warrants taking

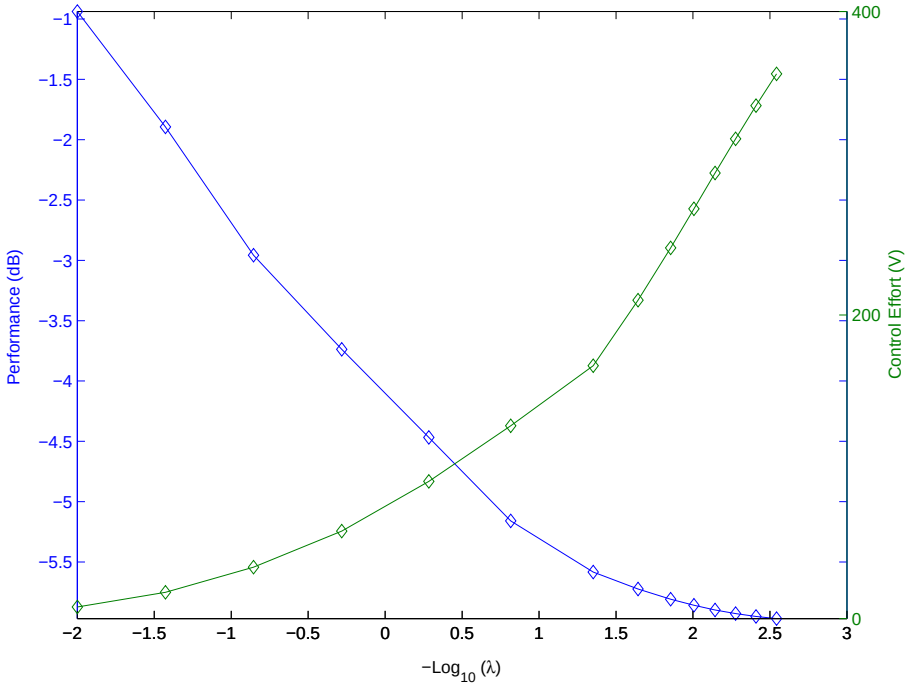


Figure 6.10: Performance during Iteration - Case 1

small steps in λ.

After the 14th iteration, the derivative to the spline fit shows that the current derivative is less than 12 percent of the minimum derivative. The spline fit and its derivative are both shown in Figure 6.13. The minimum derivative occurs at about $\lambda = 10^{-1.3}$, between the 2nd and 3rd iterations. The green line is the 12 percent level at which the adaptation should be stopped, and the current derivative is within this range. The derivative system therefore outputs a factor of 1. Since the control index is above 0.3, the metarules allow the derivative system to stop the adaptation.

In Figure 6.14, we have superimposed the derivative and U_{max} factors on the metarules decision surface. The adaptation starts in the blue region and slowly works its way up the curve. The path stays on the right side of the surface, indicating that the decisions are heavily based on the derivative factors. The adaptation stops when the path reaches the red L-shaped surface.

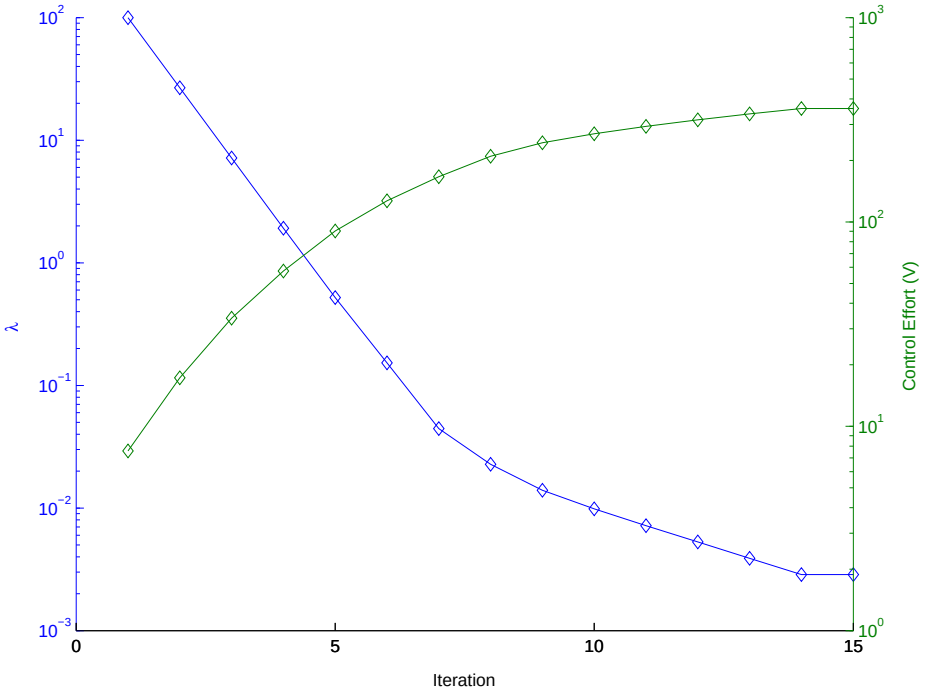


Figure 6.11: λ during Iteration - Case 1

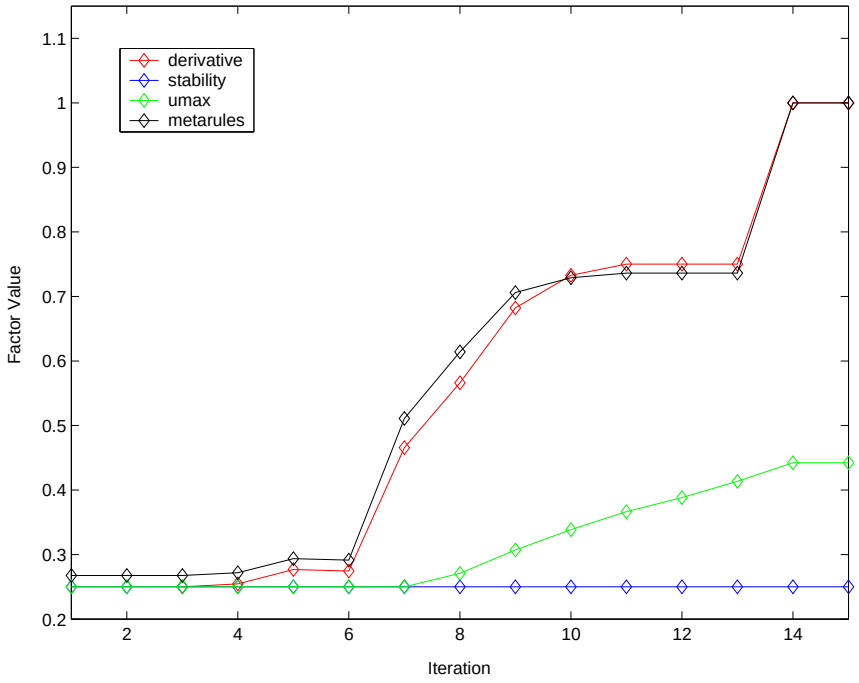


Figure 6.12: Factors - Case 1

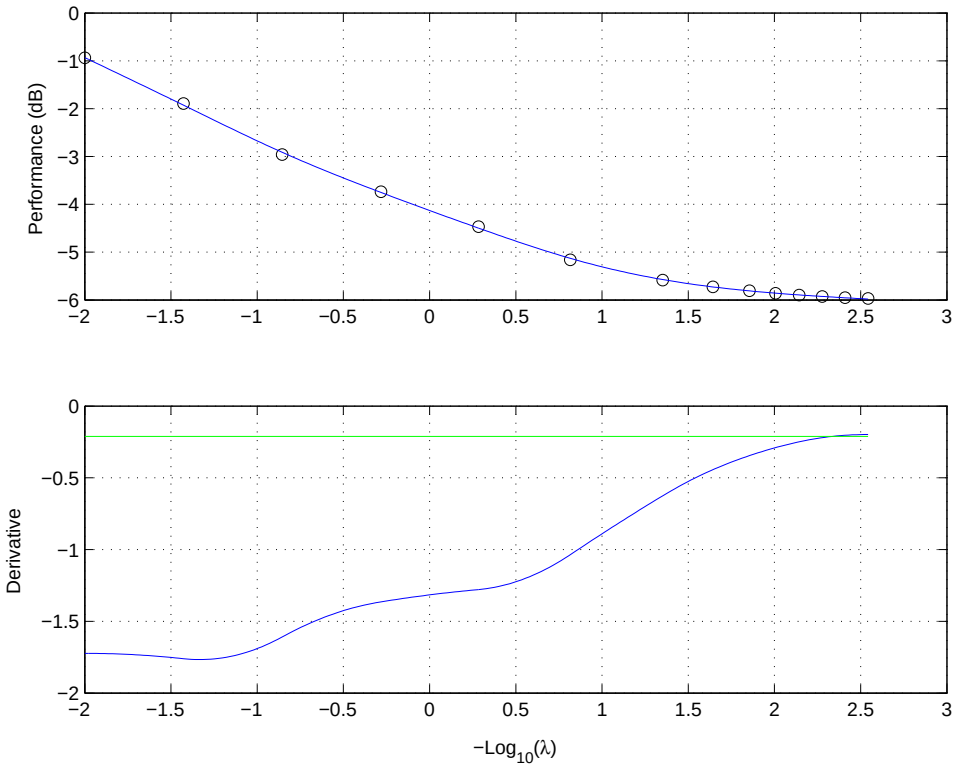


Figure 6.13: Spline Fit - Iteration 14 - Case 1

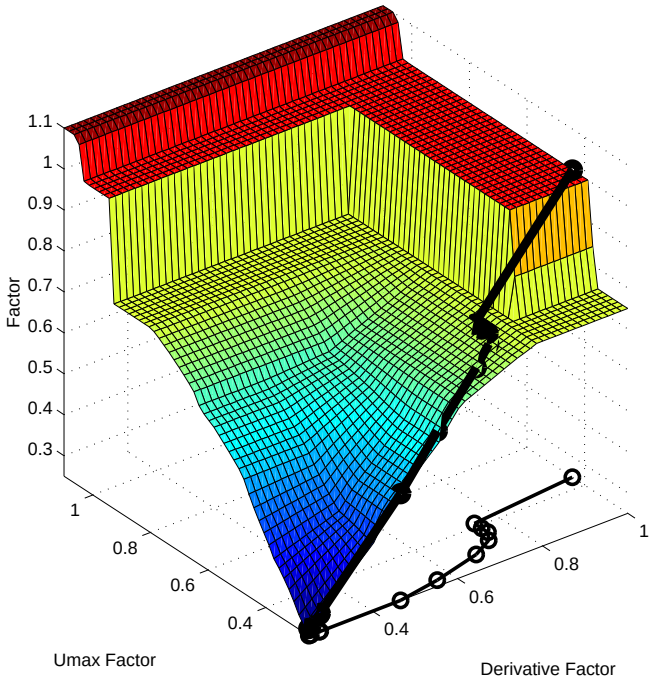


Figure 6.14: Decision Surface - Case 1

6.1.2.2 Case 2

For the second case, we will run the adaptation on the 28-mode system with the acceleration measured at the third location. The order was set to 80, and the horizons were set to 300. These settings are identical to those of the first simulation case. We have, however, chosen a U_{max} of 300 V instead of the 1000 V, as in the first case.

We saw in Case 1 that the derivative rules stopped the iteration at a λ of roughly $10^{-2.6}$ with a control effort of roughly 350 V (see Figure 6.10). Since we have selected a U_{max} smaller than 350 V, we should get an adaptation that is stopped by the U_{max} system. Here are the results of the adaptation:

Table 6.2: Case 2 - Iteration Results

Iteration	CI	Umax-F	Der-F	Stability F	Factor	λ
1	0.0253	0.25	NaN	0.25	0.267	100
2	0.0576	0.25	NaN	0.25	0.267	26.75
3	0.113	0.25	0.25	0.25	0.267	7.154
4	0.192	0.258	0.255	0.25	0.272	1.914
5	0.301	0.374	0.277	0.25	0.394	0.5203
6	0.394	0.501	0.27	0.25	0.552	0.205
7	0.453	0.609	0.289	0.25	0.622	0.1132
8	0.497	0.665	0.338	0.25	0.675	0.07039
9	0.543	0.715	0.464	0.25	0.707	0.04748
10	0.614	0.804	0.525	0.25	0.75	0.03357
11	0.676	0.849	0.575	0.25	0.75	0.02518
12	0.742	1	0.65	0.25	1	0.01888
13	0.742	1	0.65	0.25	1	0.01888

The corresponding performance curve is shown in Figure 6.15.

A quick look at the table indicates that the U_{max} factor goes to 1 at the 12th iteration, and the metarules therefore halt the adaptation.

A two-dimensional view of the factors is shown in Figure 6.16. Here we see that the metarules output closely follows the output of the U_{max} system. Note that in the 10th and 11th iterations, the U_{max} system output is higher than 0.75 (0.804 and 0.849), but the metarules have mapped these values to an output of 0.75. This is by design, as we wrote the metarules to not allow factors between 0.75 and 1. We also see that the derivative module outputs a factor higher than 0.4 after

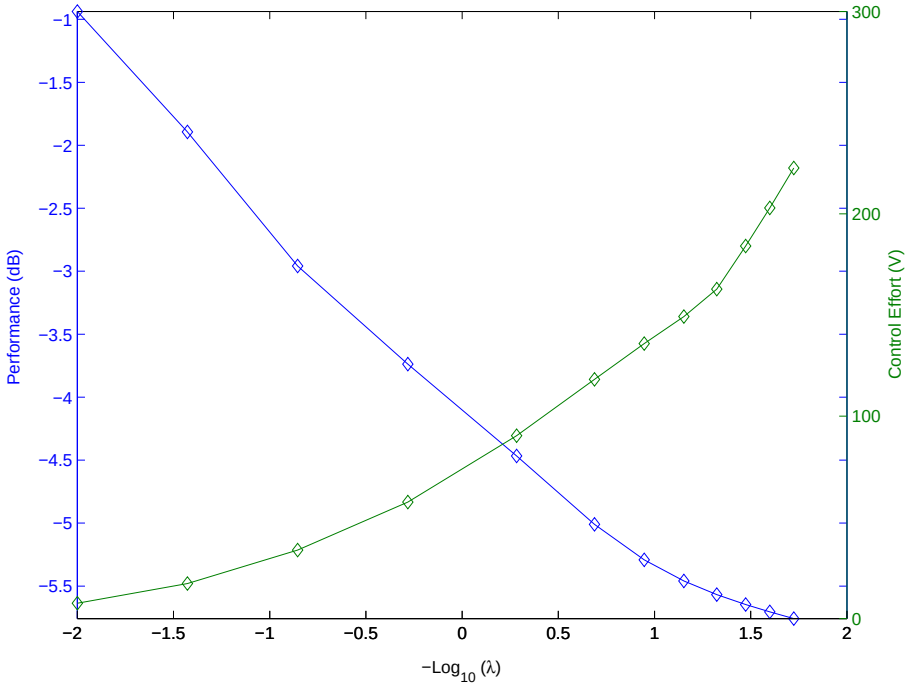


Figure 6.15: Performance during Iteration - Case 2

the 9th iteration, but this does not affect the output of the metarules. In general, we have tried to design the metarules to pick the more restrictive of the derivative and U_{max} factors, and we see that this method is working. The metarules curve gradually slopes upward from 0.25 to 0.75, indicating that the adaptation gradually comes to a stop, which is how we intended the fuzzy logic rules to work. This can also be seen in the λ curve in Figure 6.17, where we show λ as a function of iteration. The λ curve shows a gradual slope transition indicating that the adaptation comes to a smooth stop.

Figure 6.15 shows that the adaptation comes to stop at a λ of about $10^{-1.7}$, which is roughly a decade earlier than in case 1. By drastically reducing the U_{max} of Case 1 from 1000 to 300 for this second case, we have not only changed where the adaptation stops, but also which rules control and stop the adaptation.

This is further reinforced by looking at the factors superimposed on the decision surface (Figure 6.18). The path up the surface stays to the left of the surface, once again indicating that the U_{max} system is controlling the adaptation. A comparison to the same surface for Case 1 (Figure 6.14), shows that the path for Case 2 is much different than in the first case.

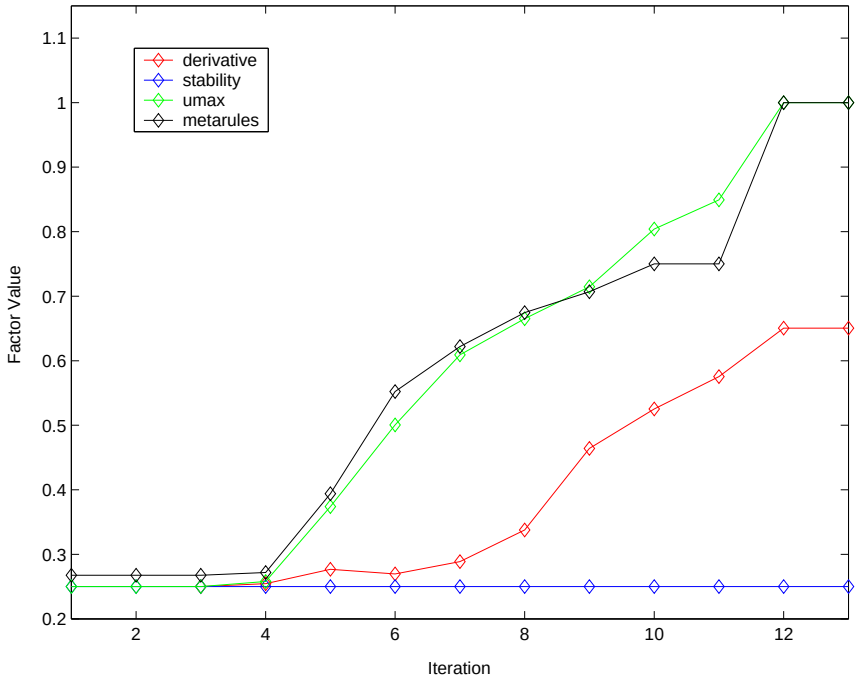


Figure 6.16: Factors - Case 2

This illustrates that the path up the surface depends not only on the shape of the performance and control effort curves, but also on the user-defined U_{max}. This is not necessarily true for all systems, however. Some performance curves do not flatten out before becoming unstable. This would result in a derivative system that does not output a factor larger than 0.25, and the path up the decision surface would stay near the left side of the surface. This will be illustrated in the experimental results.

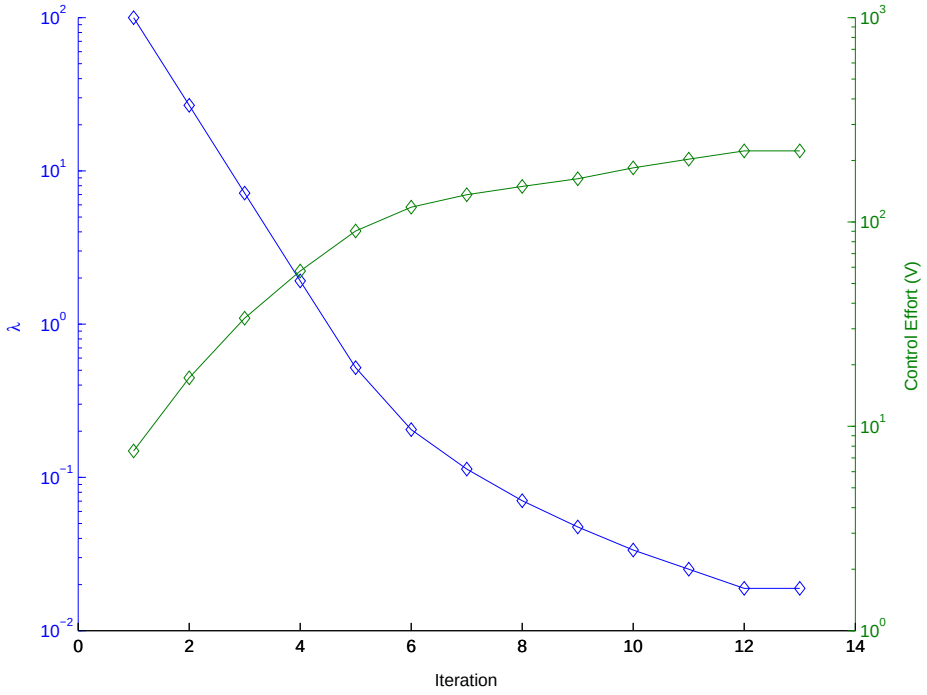


Figure 6.17: λ during Iteration - Case 2

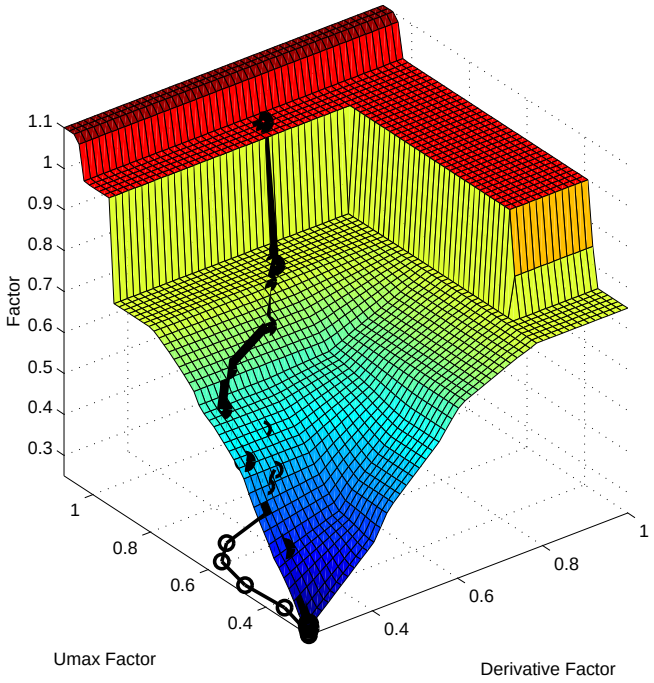


Figure 6.18: Decision Surface - Case 2

6.1.2.3 Case 3

For this case, we have chosen to run the adaptation on the 28-mode system with the acceleration measured at location 2. The order was set to 100, and U_{max} was set to 1200. The control and prediction horizons were set to 70. This value for the horizons is too low based on the recommendation in Chapter 5, but was used to ensure that the adaptation would become unstable relatively early.

The tabular results of the adaptation are:

Table 6.3: Case 3 - Iteration Results

Iteration	CI	Umax-F	Der-F	Stability F	Factor	λ
1	0.00216	0.25	NaN	0.25	0.267	100
2	0.00553	0.25	NaN	0.25	0.267	26.75
3	0.015	0.25	0.25	0.25	0.267	7.154
4	0.0362	0.25	0.25	0.25	0.267	1.914
5	0.0658	0.25	0.25	0.25	0.267	0.5118
6	0.0945	0.25	0.377	0.25	0.391	0.1369
7	NaN	0.25	0.377	4	4	0.05352
8	0.0834	0.25	0.348	1	1	0.2141
9	0.0834	0.25	0.348	1	1	0.2141

The performance curve is shown in Figure 6.19.

The 7th iteration becomes unstable and the stability module outputs a factor of 4, which is the value of $action = revert$. The second rule of the metarules system is fired and *Factor* is assigned the value of 4. The unstable point is not shown on the performance curve. The next value of λ is 0.2141 ($log_{10}(\lambda) = -0.67$) and results in a stable closed-loop system during the 8th iteration. This is shown as a point unattached from the rest of the curve.

Since the last iteration was unstable and the current iteration is stable, the stability system outputs the action *hold* (factor = 1). The first rule of the metarules system is fired and the final factor is also given a value of 1. The stability system will continue to output a value of *hold*, thereby halting the adaptation of λ. The final value of λ has a performance that is almost as good as the λ before the instability.

Analysis of the actual closed-loop poles shows that the 7th iteration becomes unstable because a pole is pushed outside the stability boundary at a frequency of 1285.77 Hz. Remarkably, the po-

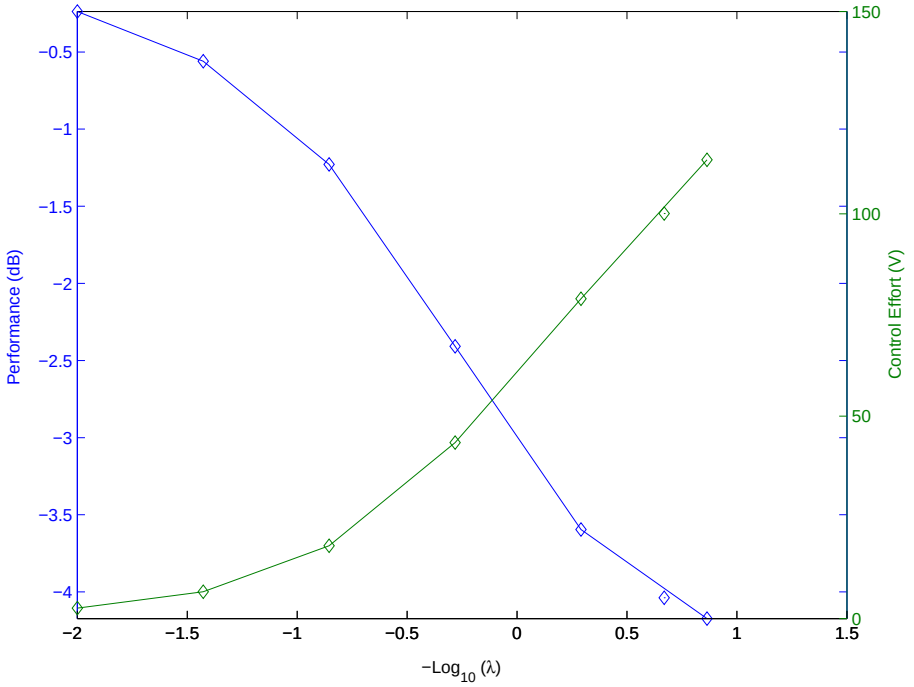


Figure 6.19: Performance during Iteration - Case 3

sition of the unstable pole is correctly predicted by the apparent closed-loop poles with a positional error of 1.08×10^{-4}. Since we could have predicted the instability ahead of time, there was no need to implement the unstable λ. We have not built these predicted instabilities into the current fuzzy logic rules, but this case shows that this should be included.

6.1.3 Experimental Results

We have shown the results of three laboratory experiments that serve to illustrate the utility of the fuzzy logic rules. The first case shows an interesting experiment in which the adaptation is halted by the U_{max} system, but further iterations cause it to become unstable. For the second case, we restricted the U_{max} of the first case, thereby avoiding the instability altogether. The third case demonstrated an experiment in which the derivative rules dominated and successfully stopped the adaptation.

6.1.3.1 Case 1

For this case, the accelerometer was mounted to location 1 and the order was fixed to 200. U_{max} was set to 3 V, and both horizons were given a value of 150. Each iteration collects 30,000 samples of closed-loop data. The adaptation results are:

Table 6.4: Case 1 - Iteration Results

Iteration	CI	Umax-F	Der-F	Stability F	Factor	λ
1	0.0127	0.25	NaN	0.25	0.267	100
2	0.021	0.25	NaN	0.25	0.267	26.75
3	0.0366	0.25	0.296	0.25	0.313	7.154
4	0.0639	0.25	0.25	0.25	0.267	2.236
5	0.131	0.25	0.25	0.25	0.267	0.5982
6	0.277	0.347	0.25	0.25	0.36	0.16
7	0.471	0.634	0.25	0.25	0.641	0.05768
8	0.469	0.631	0.25	0.25	0.637	0.03696
9	0.609	0.8	0.25	0.25	0.75	0.02352
10	0.598	0.79	0.251	0.25	0.747	0.01764
11	0.724	1	0.25	0.25	1	0.01318
12	0.648	0.827	0.25	0.25	0.75	0.01318
13	0.783	1	0.267	0.25	1	0.009885
14	0.706	1	0.267	0.25	1	0.009885
15	1	1.1	0.267	4	4	0.009885
16	0.501	0.669	0.25	1	1	0.03954
17	0.476	0.64	0.25	1	1	0.03954
18	0.439	0.588	0.25	1	1	0.03954
19	0.468	0.63	0.25	1	1	0.03954
20	0.538	0.709	0.25	1	1	0.03954

The performance curve is shown in Figure 6.20.

There are several interesting things occurring in this adaptation. The plot of the factors, Figure 6.22, shows that there is a steady increase in the U_{max} factor from the 5th to the 10th iteration. The metarules output closely follows the U_{max} factor. This gradual increase in Factor results in a gradual decrease in the slope of the λ curve (Figure 6.21).

At the 10th iteration, the control index (CI) is large enough to result in a U_{max} factor of 1, and the adaptation is stopped. The next iteration, even though it uses the same λ, has a control index that is slightly smaller. This results in a metarules factor of 0.75, and λ is reduced to 0.009885. The control index is again large enough for the U_{max} system to stop the adaptation. This λ stays stable for two iterations and then becomes unstable during the 15th iteration. The unstable iteration has a performance of a positive 7.3 dB and a control effort of 10 V.[7] The stability module takes over the adaptation and increases λ by a factor of 4 for the 15th iteration. This value of λ stays stable, and is therefore held constant by the stability system. This halts the adaptation.

Note that this final value of λ results in a performance of about 6.4 dB (see Figure 6.20). This is about 0.9 dB worse than the absolute best performance. It could be said that in this particular case, the reversion increased λ to a level that has an unacceptable performance. The reversion factor of 4 could be considered too large for this system. As we stated in Section 6.1.1.1, the reversion factor of 4 was chosen because it was the inverse of the largest drop factor of 0.25. For this case, a reversion of 2 or maybe even $\frac{1}{0.75} = 1.33$ would have been more appropriate.

This experimental case can be compared to the third numerical simulation case. In both cases, the closed-loop systems become unstable and the stability module stabilizes the system. A key difference is that a λ that results in a stable closed-loop system in the laboratory does not necessarily stay stable indefinitely. For this case, we had two stable iterations with a λ of 0.009885 before the system became unstable. Another difference is that in this case, we lost about 0.9 dB in performance by going into instability, but in the third simulation case, we did not lose such a significant amount of performance.

The apparent-closed loop poles do not show an instability for the 15th iteration, and therefore the instability could not have been predicted before closing the loop. The stability system

[7]The control index for the 15th iteration was initially 10/3=1.3, but this value was reset to 1 to keep it within the allowable input range (0-1) of the U_{max} system.

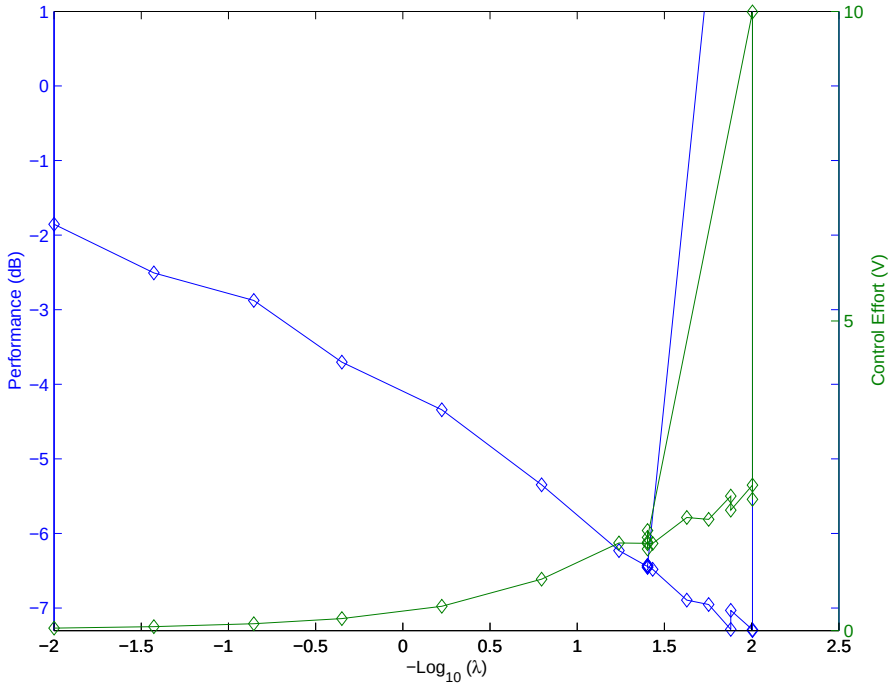


Figure 6.20: Performance during Adaptation - Case 1

nevertheless corrects the situation.

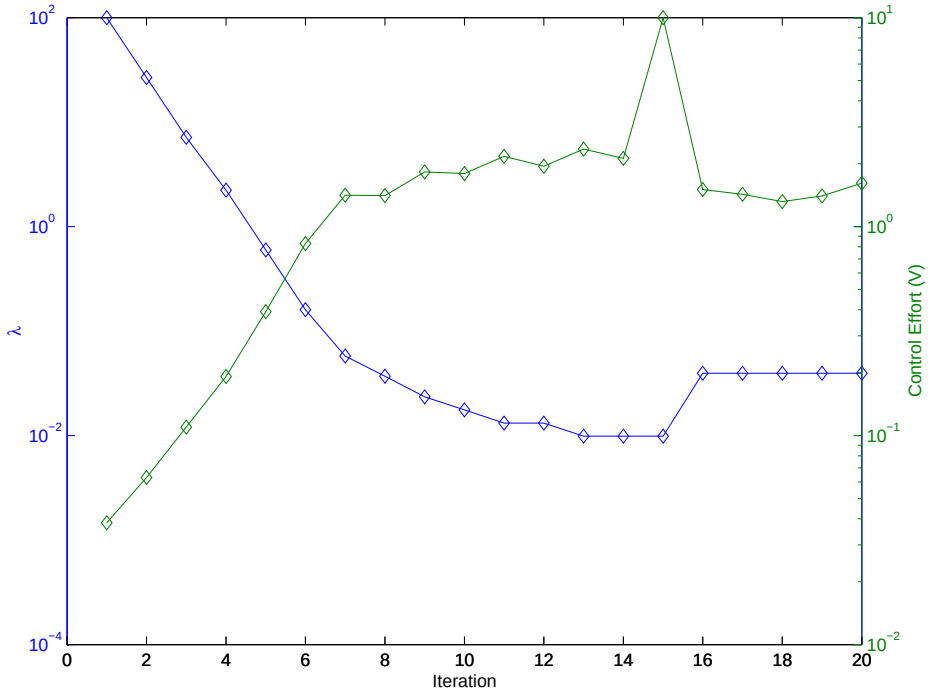


Figure 6.21: λ during Iteration - Case 1

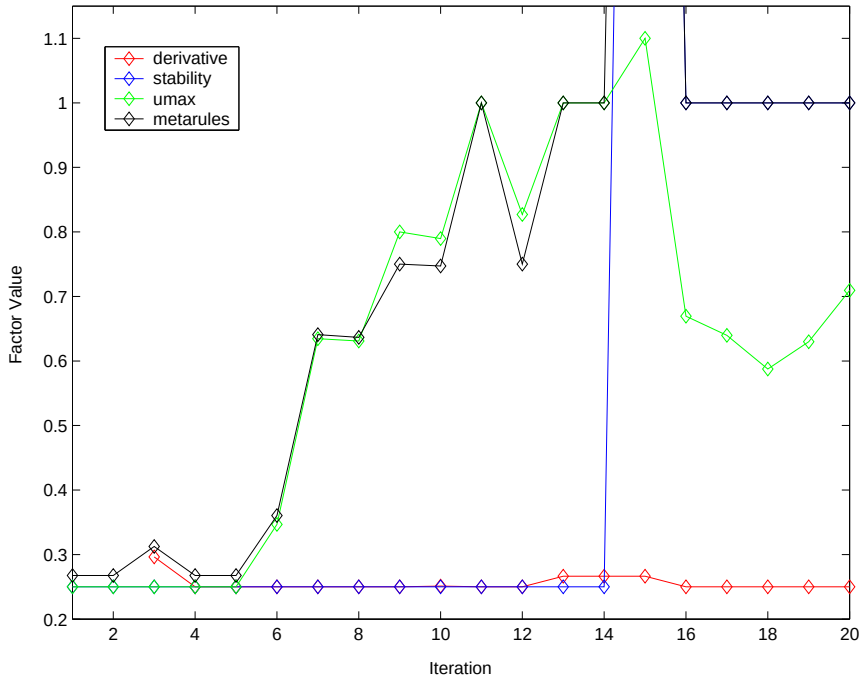


Figure 6.22: Factors - Case 1

6.1.3.2 Case 2

The settings for this case are identical to the first experimental case except that U_{max} is now set to 2 V. The horizons are set to 150, the order is fixed at 200, the accelerometer is fixed to location 1, and 30,000 closed-loop samples are collected for each iteration. The previous case became unstable, then settled with a final value of λ of 0.03954 ($log_{10}(\lambda) = -1.4$). The user might find this descent into instability very undesirable, and wish to avoid it altogether. Since the control effort gradually increased (see Figure 6.20) until the point of instability, the user could theoretically prevent the instability by decreasing U_{max}. This is exactly what we have done for this case.

The results of adaptation are:

Table 6.5: Case 2 - Iteration Results

Iteration	CI	Umax-F	Der-F	Stability F	Factor	λ
1	0.0187	0.25	NaN	0.25	0.267	100
2	0.0317	0.25	NaN	0.25	0.267	26.75
3	0.0504	0.25	0.25	0.25	0.267	7.154
4	0.105	0.25	0.25	0.25	0.267	1.914
5	0.213	0.274	0.25	0.25	0.268	0.5118
6	0.408	0.528	0.25	0.25	0.572	0.1371
7	0.635	0.819	0.25	0.25	0.75	0.07847
8	0.614	0.804	0.25	0.25	0.75	0.05886
9	0.625	0.813	0.25	0.25	0.75	0.04414
10	0.813	1	0.264	0.25	1	0.03311
11	0.831	1	0.264	0.25	1	0.03311
12	0.772	1	0.264	0.25	1	0.03311
13	0.785	1	0.264	0.25	1	0.03311
14	0.789	1	0.264	0.25	1	0.03311

The performance curve is shown in Figure 6.23. Examining the individual factors in Figure 6.24 shows that the metarules output follows the output of the U_{max} system. The U_{max} system therefore completely determines the adaptation of λ. Due to the rapid rise in the control effort curve (Figure 6.23), the U_{max} system rapidly increases its output after the 5th iteration. By the 7th iteration, the effort is large enough to result in a metarules output of 0.75, the smallest possible change in λ. By the 10th iteration, the control index reaches a level that warrants a halt in the adaptation. This rapid increase in the metarules factor is evident in the fast change in slope of the λ curve (Figure 6.25).

Further iterations result in small changes in the control index, but the output of the U_{max} system continues to be 1. This is exactly why we created the large flat region in the U_{max} curve (Figure 6.6): we did not want small changes in the control index to cause continuous changes in λ. This case validates this design intent.

The final λ results in performances between 6.43 dB and 6.6 dB. Comparing this case to the first experimental case we see that we have achieved a nearly identical performance without causing the system to become unstable. Also, the final λ is almost the same as in the first case. We therefore have achieved the goal we had established for this case: we have prevented the instability yet still achieved good performance.

This case is similar to the second numerical simulation case, since both cases are controlled by the U_{max} system. In the simulation case, we saw a gradual increase in the metarules factors, whereas this case showed a very rapid increase in this factor. The difference is the slope of the control effort curves. In the simulation case (Figure 6.15), the change of slope as λ is reduced is very small compared to the change in slope for this case (Figure 6.23). Nevertheless, the U_{max} system handles either type of situation. If the increase in slope was even more drastic than in this case, the adaptation would most likely overshoot the stopping condition of the U_{max} system and go directly to the rule that would increase λ by 10%.

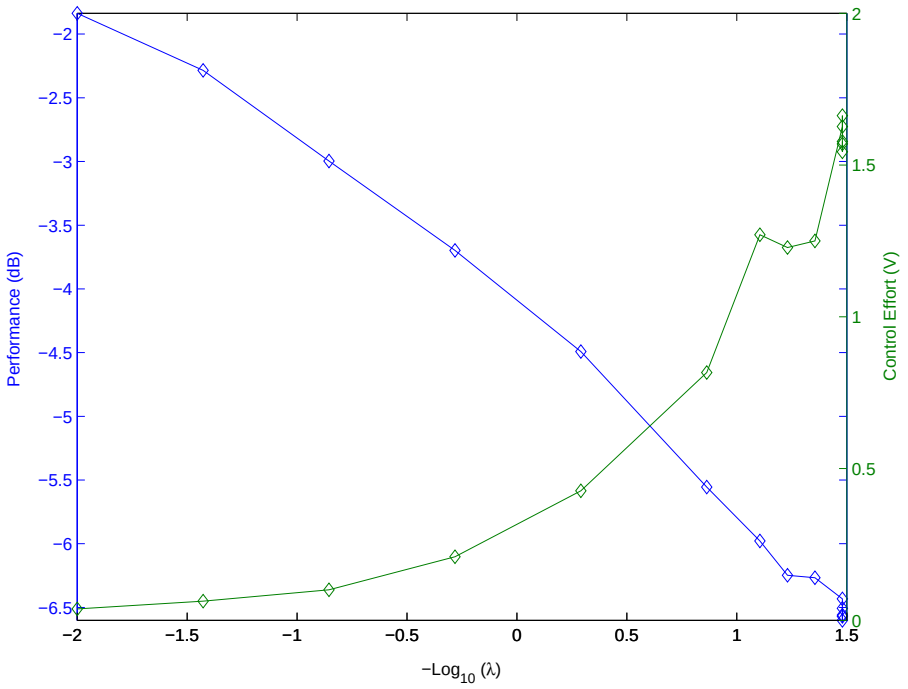


Figure 6.23: Performance during Iteration - Case 2

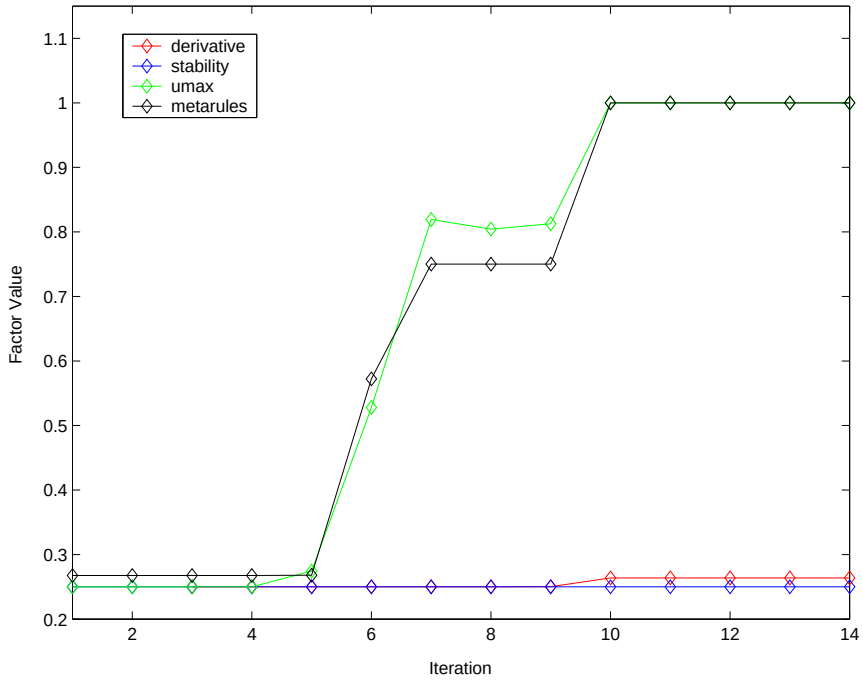


Figure 6.24: Factors - Case 2

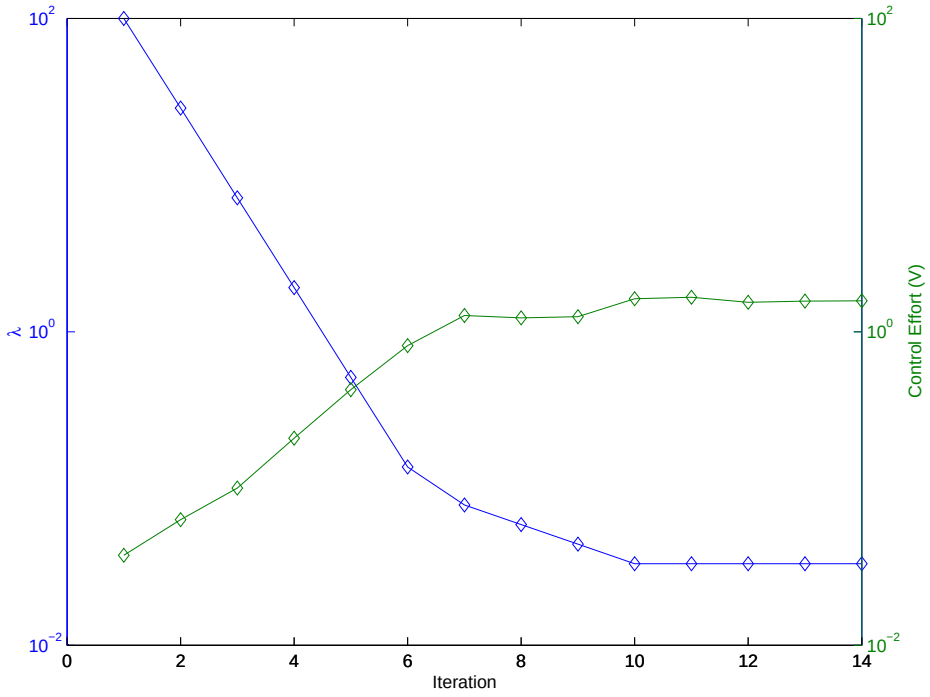


Figure 6.25: λ during Adaptation - Case 2

6.1.3.3 Case 3

The order was fixed to 200 for this case, and the accelerometer was moved to the fourth measurement location. The horizons were given a value of 300, and U_{max} was set to 4. The results of the adaptation are:

Table 6.6: Case 3 - Iteration Results

Iteration	CI	Umax-F	Der-F	Stability F	Factor	λ
1	0.0085	0.25	NaN	0.25	0.267	100
2	0.0238	0.25	NaN	0.25	0.267	26.75
3	0.0619	0.25	0.25	0.25	0.267	7.154
4	0.102	0.25	0.25	0.25	0.267	1.914
5	0.177	0.252	0.25	0.25	0.267	0.5118
6	0.28	0.352	0.294	0.25	0.385	0.1369
7	0.338	0.414	0.665	0.25	0.698	0.05268
8	0.436	0.583	0.524	0.25	0.634	0.03678
9	0.438	0.587	0.519	0.25	0.633	0.0233
10	0.615	0.805	0.592	0.25	0.75	0.01476
11	0.562	0.74	0.579	0.25	0.724	0.01107
12	0.679	0.85	0.75	0.25	0.75	0.008018
13	0.71	1	1	0.25	1	0.006014
14	0.66	0.836	1	0.25	1	0.006014
15	0.633	0.818	1	0.25	1	0.006014
16	0.668	0.843	1	0.25	1	0.006014
17	0.664	0.839	1	0.25	1	0.006014

The performance curve is shown in Figure 6.26. Figure 6.27 shows that both the derivative and U_{max} systems slow down the adaptation rate. The metarules module outputs the larger of the two factors, but different systems control the adaptation during the first twelve iterations. For example, in the 6th, 10th, 11th, and 12th iterations, the U_{max} system is the larger of the two and therefore controls the iteration. In the 7th iteration, the derivative system controls the iteration.

By the 12th iteration, both the derivative and the U_{max} systems request the smallest possible change in λ. After the 13th iteration, both systems wants to stop the adaptation. The metarules output a factor of 1 and the adaptation is halted. The spline fit and its derivative are shown in Figure 6.28. The current derivative is just within the green line, indicating that the adaptation should stop.

For the rest of the iterations, the control index is slightly smaller and the U_{max} factor is no

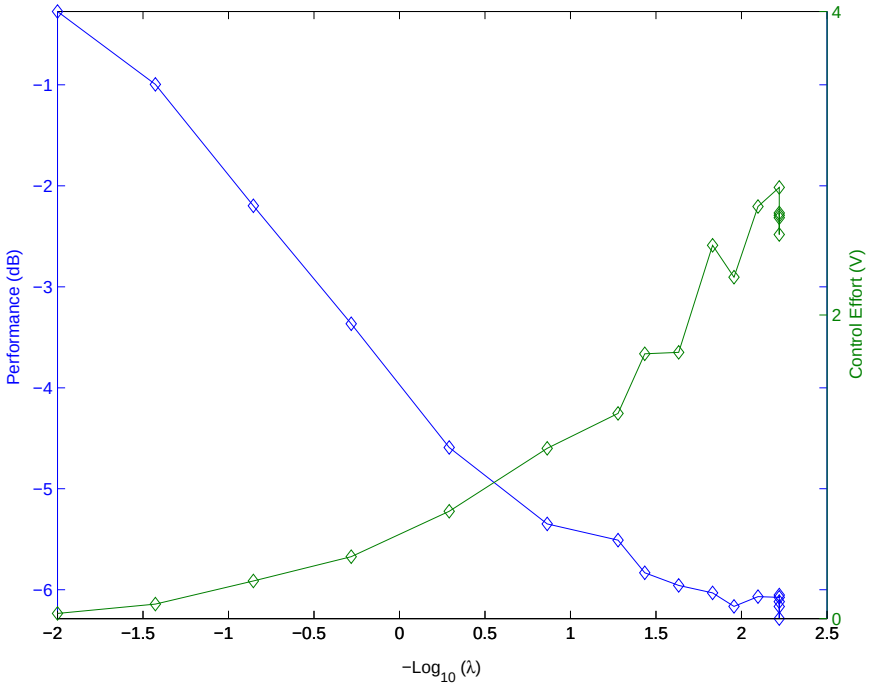


Figure 6.26: Performance during Iteration - Case 3

longer 1. The derivative system, however, continues to output a factor of 1. As was stated in Section 6.1.1.2, the list of data points must be scanned for duplicate values of λ. The algorithm we implemented scans the list of points and eliminates any duplicate points, keeping only the first instance. Therefore, the performance value of the 14th through the 17th iterations are seen as duplicates of the 13th iteration and are eliminated for the purpose of the spline fit. It is for this reason that the derivative system continues to output a value of 1 after the 13th iteration.

In Figure 6.26, we see that most of the iterations past the 13th have a slightly better performance than the 13th iteration. If we had chosen to average the performance values of duplicate λs instead of eliminating them, then the adaptation would have continued to decrease λ.

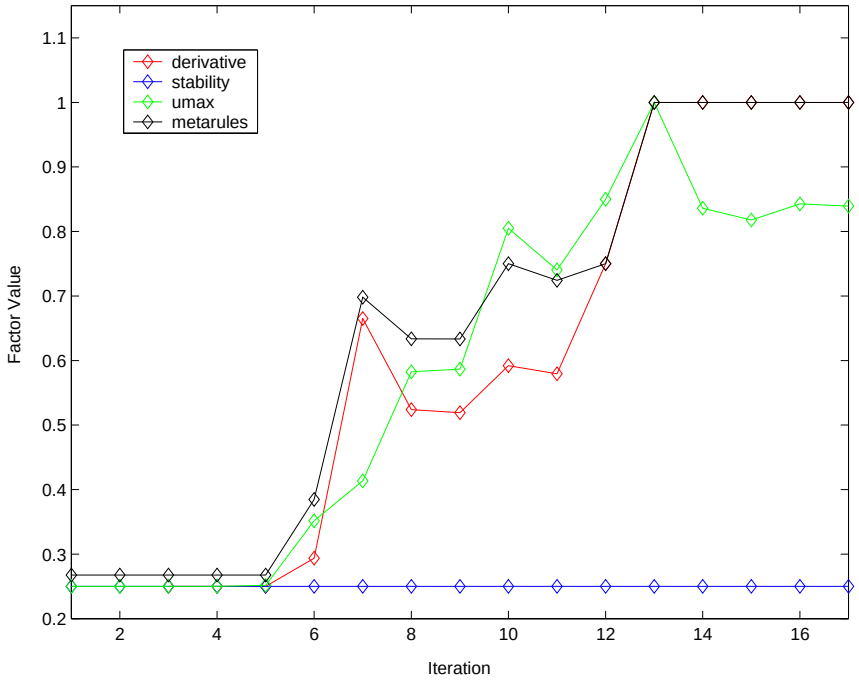


Figure 6.27: Factors - Case 3

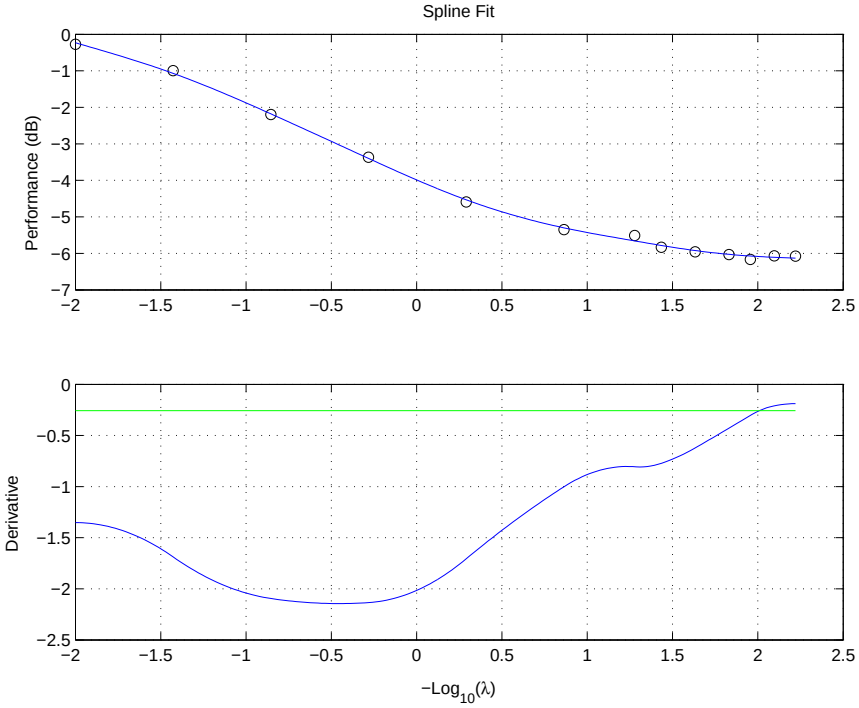


Figure 6.28: Spline Fit - Iteration 13 - Case 3

6.2 Order Adaptation

6.2.1 Rules

The adaptation of order is in most ways exactly the same as the adaptation of λ. We want to keep the three main stoppage conditions of stability, maximum control effort, and performance curve derivative, and then add a fourth condition. This additional condition is the idea of maximum order. Our signal processing hardware must compute the new control levels between sample periods, and this naturally places an upper limit on the number of computations that can be completed during that time. The number of computations depends heavily on the order of the controller, and we will therefore allow the user to choose an upper limit for the order.

The output of the metarules will still be called *Factor*, but we will use the following formula to adapt order (*O*):

$$O_{new} = O_{old} + Factor \times step\,size \tag{6.2}$$

Factor will take on values of 0, -1, and 1-4. The *step size* is a user-chosen number that should reflect the overall size of the plant being controlled. Since we allow non-integer values of *Factor*, and *order* must be an integer, the product of $Factor \times step\,size$ is rounded to the nearest integer before it is added to O_{old}. A *Factor* of 0 corresponds to no change in order, and therefore a halt in the adaptation. The adaptation of order will always start with a small order, and subsequent iterations will result in an increase in order until one of the four stopping conditions is met. A reversion occurs when $Factor = -1$.[8]

A *Factor* of 4, the maximum allowable rate of change, is equivalent to the *fulldrop* (0.25) factor of the λ adaptation. A *Factor* of 0 in the order adaptation (no change in order) is equivalent to a *Factor* of 1 in the λ adaptations. Once this equivalency was established, we used the rules that we developed for the λ adaptation and converted them into rules that adapt order. There are, however, some significant differences between the order and λ adaptations in the stability and metarules systems. This will be covered in the following sections.

We have chosen a value of *step size* of 5 for our experimental work, as this gives us a maximum

[8]The value of -1 is somewhat arbitrary, just as a reversion factor of 4 for the λ adaptation was arbitrary.

change in order of 20 and a minimum change of 5.

6.2.1.1 Stability System

Although the derivative and U_{max} systems will remain relatively unchanged from their λ counterparts, the stability system requires large changes for it to work in the order domain. We have seen in Chapter 5 that most closed-loop systems are unstable if the order is significantly smaller then the true order of the plant. Therefore, we must allow the system to be unstable for small orders and still continue to increase order. However, if the order is already large, then an instability should trigger a reversion, i.e. a decrease in order.

We have also included in this system the ability to compare the current order ($O_{current}$) to the user-defined maximum order (O_{max}). If the current order is larger than O_{max}, then the stability module will trigger the action revert until the order falls below the maximum allowable.

These are the new rules:

1: If (stable is yes) and (previous is iterate) and (ratio is not oversize) then (action is iterate)

2: If (stable is yes) and (previous is revert) then (action is hold)

3: If (stable is yes) and (previous is hold) then (action is hold)

4: If (stable is no) and (previous is iterate) and (ratio is small) then (action is iterate)

5: If (stable is no) and (previous is hold) then (action is revert)

6: If (stable is no) and (previous is revert) then (action is revert)

7: If (stable is no) and (previous is iterate) and (ratio is large) then (action is revert)

8: If (ratio is oversize) then (action is revert)

The inputs *stable* and *previous* are the same as in the λ stability system described in Section 6.1.1.1. The output *action* has the same linguistic variables, *iterate*, *hold*, and *revert*, but they have new values of 4, 0, and -1.

We have defined a new input *ratio* as:

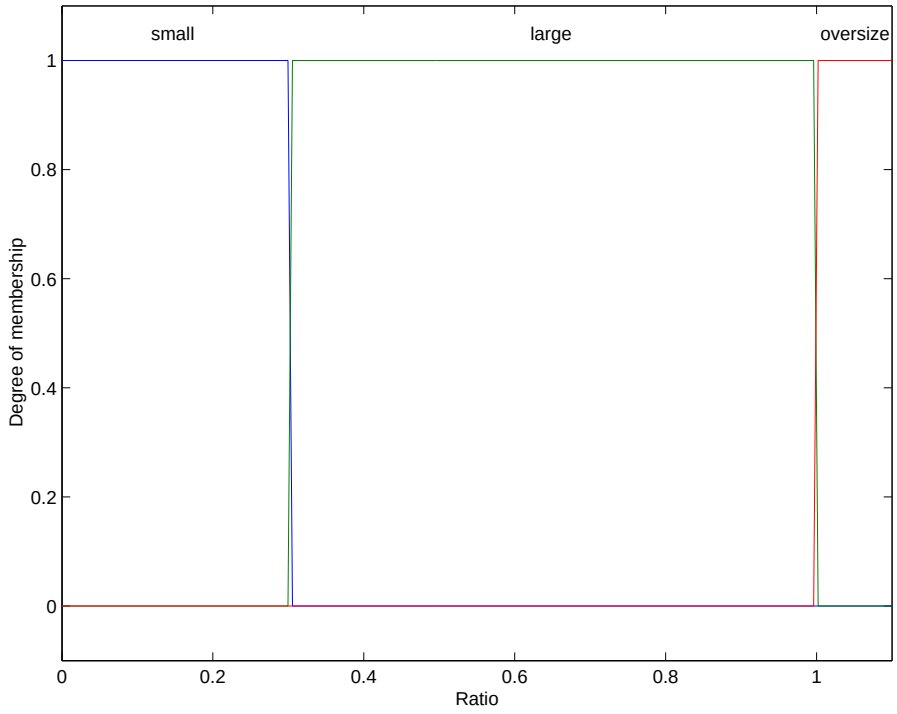


Figure 6.29: Ratio Membership Functions for Order Stability System

$$ratio = \frac{O_{current}}{O_{max}}.$$

Ratio can take on one of three possible linguistic variables: *small*, *large*, or *oversize*. The membership functions for these variables are shown in Figure 6.29. Ratios that are below 0.30 are *small*, ratios between 0.30 and 1 are *large*, and ratios larger than 1 are *oversize*.

Rules 1, 8, 4 and 7 are the new or modified rules. Rule 8 is very straightforward: if $O_{current}$ is larger than O_{max}, then we will revert, i.e. decrease the order by one step size.

Rules 4 and 7 work together to determine what is done when the system is unstable and we had previously iterated. If $O_{current}$ is less than 30% of O_{max}, then we allow the adaptation to proceed and order to increase. The assumption here is that we have not yet begun to see the stable region and that increasing the order will help us stabilize the system. If *ratio* is larger than 0.3, then rule 8 will fire and the adaptation will revert to a smaller order. This assumes that we have a stable region which we passed by making order too large.

6.2.1.2 Derivative System

The function of the derivative system remains unchanged from the λ adaptation of Section 6.1.1.2. We again form the derivative ratio:

$$ratio = \frac{F'(order)}{F'_{min}},$$

where $F'(order)$ is the value of the derivative at the current order, and F'_{min} is the absolute minimum of the derivate for the available range of orders. The spline fit also remains the same except that we have changed the smoothing parameter to 1×10^{-4} to better fit the data.

The rules of the order derivative system are:

1: If (ratio is unity) then (factor is 4steps)

2: If (ratio is half) then (factor is 3steps)

3: If (ratio is quarter) then (factor is 2steps)

4: If (ratio is small) then (factor is 1step)

5: If (ratio is verysmall) then (factor is nodrop)

The membership functions for the input linguistic variables are shown in Figure 6.30. The output linguistic variables *4steps*, *3steps*, *2steps*, *1step*, and *nodrop* are given the values 4, 3.5, 2.5, 1, and 0.

The input-output map of this module is shown in Figure 6.31. We have built two important features into this curve. First, we have built a step into the curve from a factor of 0 to a factor of 1.0 so that we do not get less than one step change in the size of order. The location of this step defines a region in which the derivative system halts the adaptation. We have changed the maximum ratio that will stop the adaptation to 8%, instead of the 12% for λ derivative system. This change will be explained in the metarules section.

The second feature is the overall concavity of the curve. Here the concavity points downward, so that the curve slowly turns downward to a factor of 1. This was done for the same reason as in the λ system – to limit the number of iterations until stoppage.

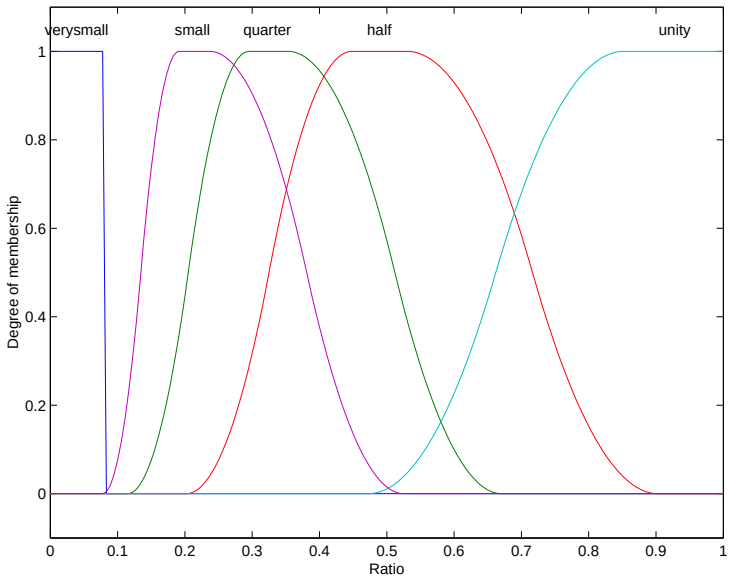


Figure 6.30: Order Derivative System Membership Functions

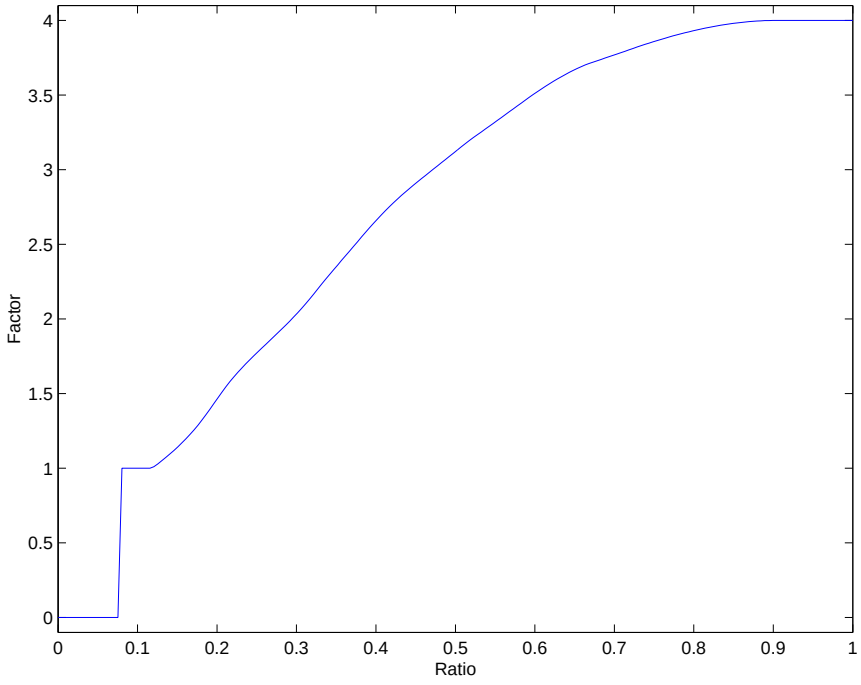


Figure 6.31: Derivative System

6.2.1.3 $\mathbf{U_{max}}$ System

The function of the order U_{max} system is identical to that of the λ U_{max} system of Section 6.1.1.3. The rules are:

1: If (ratio is zero) then (factor is 4steps)

2: If (ratio is quarter) then (factor is 3steps)

3: If (ratio is half) then (factor is 2steps)

4: If (ratio is 6tenths) then (factor is 1step)

5: If (ratio is threequarters) then (factor is stop)

6: If (ratio is unity) then (factor is stepback)

The output variables *4steps*, *3steps*, *2steps*, *1step*, *stop*, and *stepback* are assigned the values 4, 3, 2, 1, 0 and -1. The input membership functions are shown in Figure 6.32, and the input-output mapping is shown in Figure 6.33.

The input-output mapping has the same important features as in the λ system. First, we define a region of allowable control effort in which *Factor* equals 0. Second, we have a step change from a factor of 0 to a factor of 1 so that no step is smaller than the step size. Finally, we define a region in which the control effort is too large, and we must decrease the order to lower the effort.

For adaptation of order, we have used a more sophisticated control effort than in the adaptation of λ. For the λ adaptation, the current control effort was defined as the maximum of the absolute value of the controller output time data. It was noted in early experiments that the use of this metric resulted in very choppy control effort curves for order adaptations. Using the new metric, however, smooths out the curves considerably. This new method first calculated the standard deviation of the control time data. The points that lie outside three standard deviations were averaged and become the current control effort. Typically, about 0.25% of control effort time data was outside of three standard deviations.

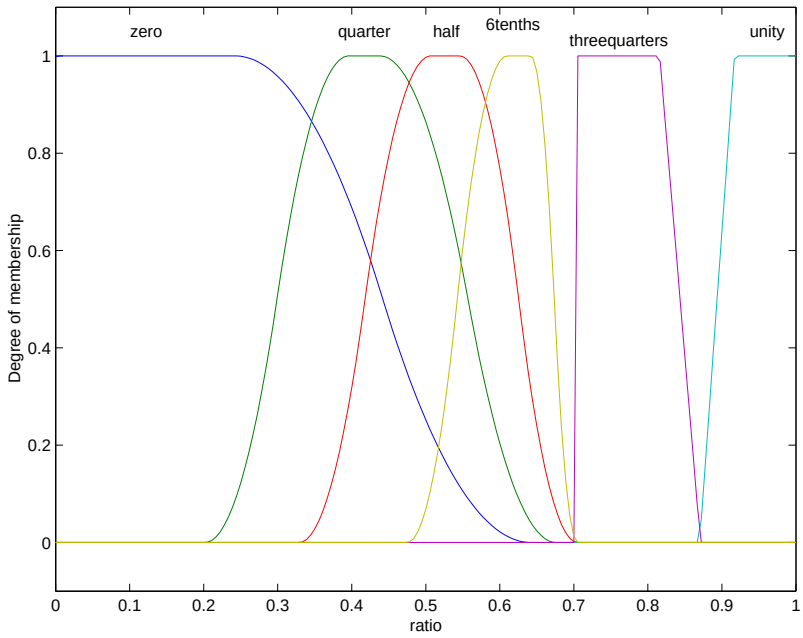


Figure 6.32: Order U_{max} System Membership Functions

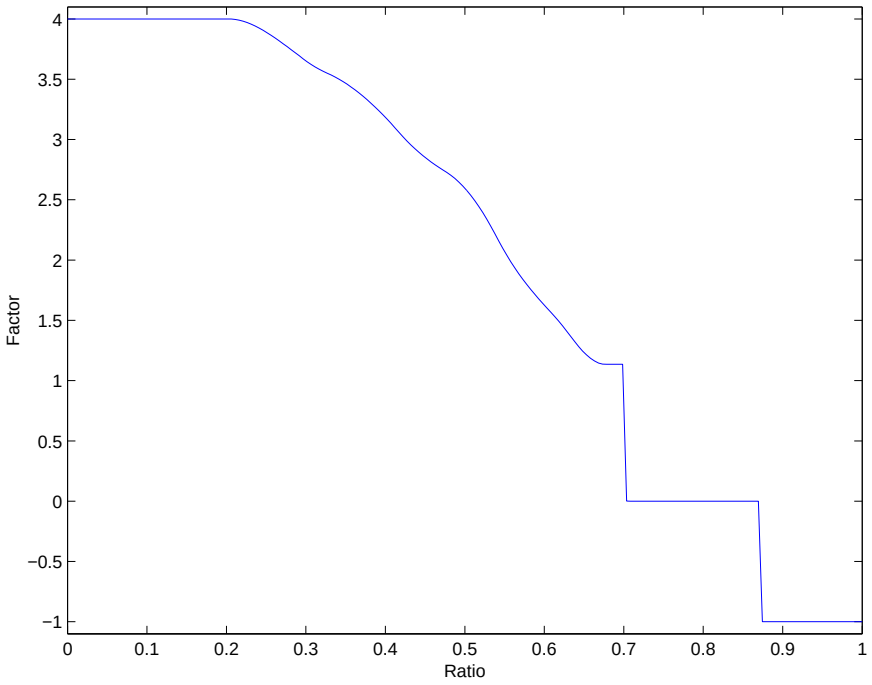


Figure 6.33: U_{max} System

6.2.1.4 Metarules

The function of the metarules system for the adaptation of order is identical to the metarules system for the adaptation of λ. The metarules system still allows the stability module to take over the adaptation if the closed-loop system is unstable. If the closed-loop system is stable, then the metarules will combine the derivative and U_{max} factors into the overall factor of Equation (6.2).

We have removed one feature of the λ metarules system. In the λ adaptation, we allowed the derivative system to stop the adaptation only if the control effort was above 30% of the allowable effort. This was done because we recognized that we could still get some performance gain by decreasing λ. This decrease in λ would lead to an increase in control effort, and the adaptation would eventually stop. In the order adaptation, it is often the case that the control effort drops or remains steady as order is increased. This means that the derivative rules might not ever stop the adaptation if we left the 30% restriction in place. For this reason, we removed the 30% restriction, and allowed the derivative rules to stop the adaptation at any control effort level. In order to make it more difficult for the derivative system to stop the adaptation, we changed the 12% stopping condition of the λ derivative system to 8% for the order system.

The rules of the metarules system are:

1: If (action is hold) then (factor is nochange)

2: If (action is revert) then (factor is revert)

3: If (action is iterate) and

Derivative Factor \ Umax Factor	S4	S3	S2	4	3	2	1	NC	I
NC	NC	NC	NC				NC	NC	B
1				1	1	1	1	NC	B
2				2	2	2	1	NC	B
3				3	3	2	1	NC	B
4				4	3	2	1	NC	B

NC = no change, 4 = four steps, 3 = three steps, 2 = two steps, 1 = 1 step, R = revert
S4 = sharp four steps, S3 = sharp three steps , S2 = sharp two steps
S4, S3, and S2 are modified versions of 4, 3, and 2

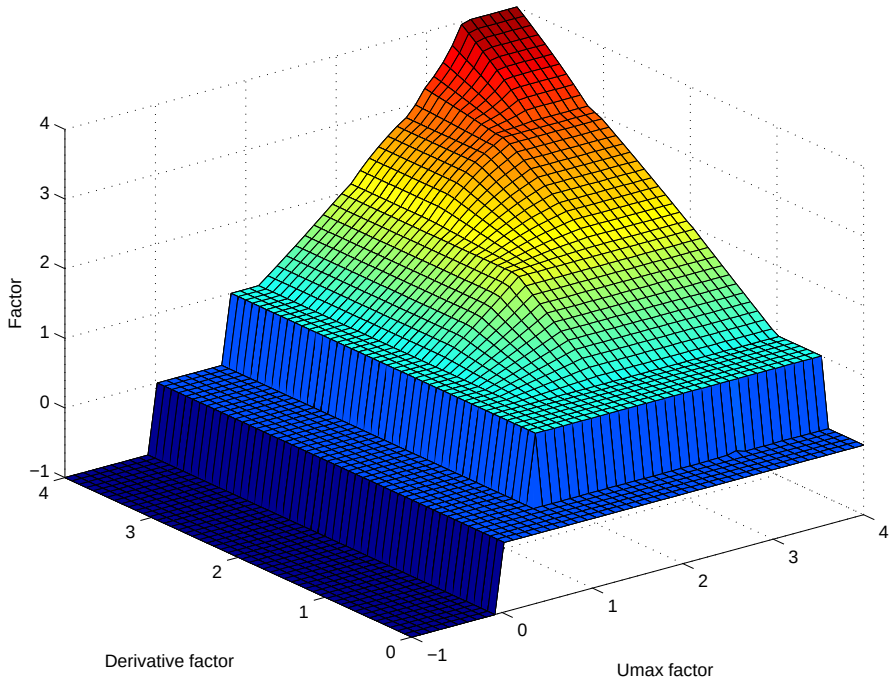


Figure 6.34: Decision Surface

A comparison to the λ metarules shows that the rules are nearly identical to those used previously. Here, we just used the equivalency rules established in the introduction to convert the rules. The output linguistics variables NC, 4, 3, 2 ,1, and R take on the crisp values of 0, 4, 3, 2, 1, and -1. The input membership functions are shown in Appendix B.

If *action* is *iterate*, then the table forms a three-dimensional surface. This surface is shown in Figure 6.34. Here, the outputs of the U_{max} and derivatives system are the inputs to the surface. We can also generate a surface from the inputs of the U_{max} and derivatives system to the output of the metarules. This is shown in Figure 6.35, with its corresponding contour plot shown in Figure 6.36.

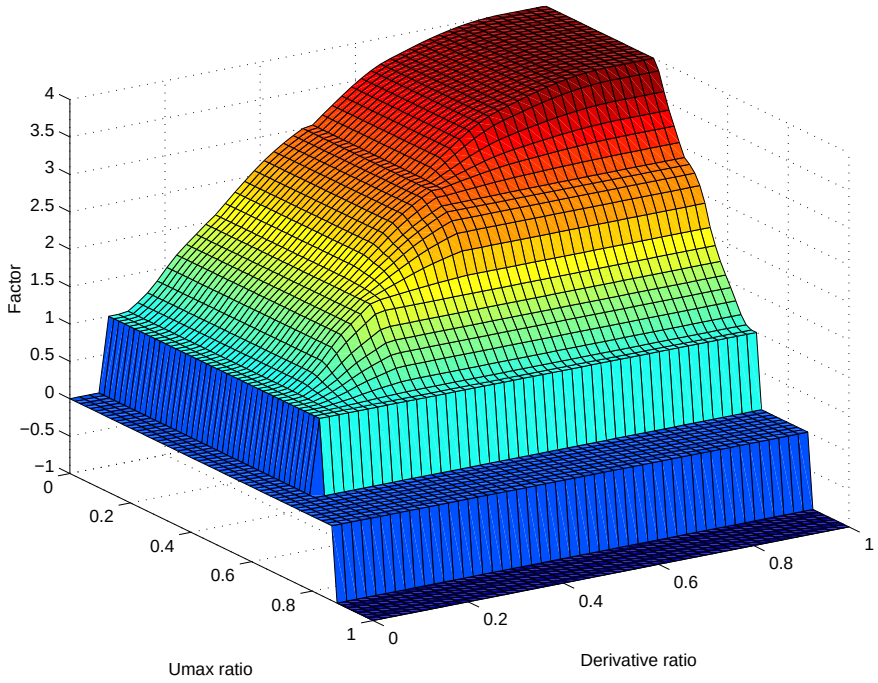


Figure 6.35: Decision Surface with Ratio Inputs

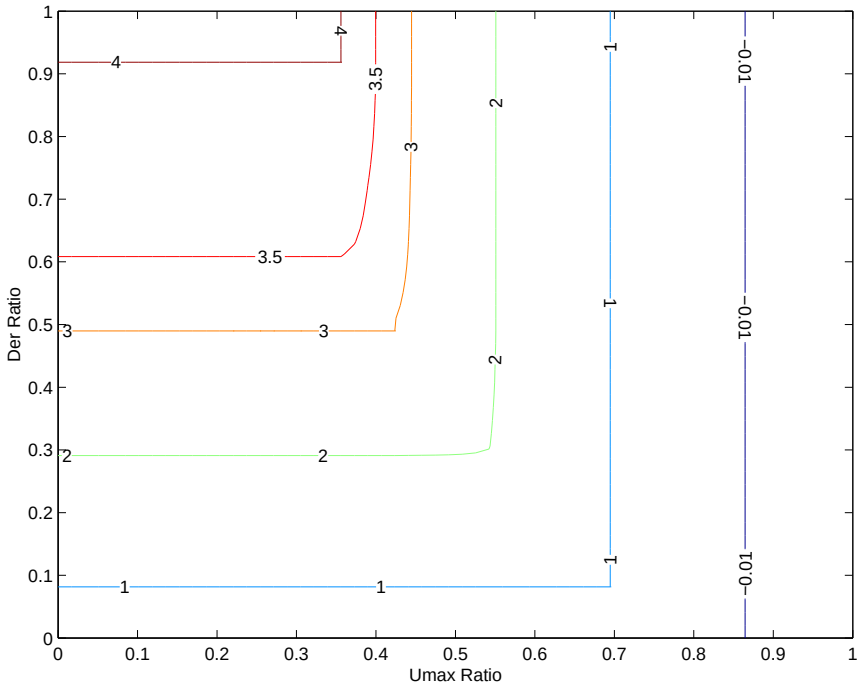


Figure 6.36: Decision Surface - Contour View

6.2.2 Numerical Results

We have selected four cases that demonstrate the utility of the fuzzy logic rules in a variety of ways. The first case shows an adaptation in which the derivative rules stop the iteration. For the second case, we restricted the maximum allowable order of the first case to test the new stability system. The U_{max} system controls the adaptation in the third case, while the fourth case shows a system in which the adaptation becomes unstable.

6.2.2.1 Case 1

For this case, we have used the 28-mode system with the acceleration measured at location 4. λ was set to 0.0389, and the horizons were given values of 300. O_{max} was set to 400, and U_{max} was set to 400V. The step size was set to 5, and 50,000 closed loop points were collected for each iteration. The adaptation was started with an order of 50. The results of the adaptation are:

Table 6.7: Case 1 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	1	249	0.622	-2.73	1.46	NaN	4	1.34	50
2	1	214	0.535	-3.41	2.25	NaN	4	2.24	57
3	1	170	0.425	-4.12	3	4	4	3.21	68
4	1	172	0.431	-4.59	2.96	4	4	3.17	84
5	1	172	0.429	-4.77	2.98	3.79	4	3.14	100
6	1	173	0.432	-4.91	2.95	2.86	4	2.82	116
7	1	172	0.43	-4.99	2.97	1.77	4	1.78	130
8	1	171	0.428	-5.03	2.99	1.2	4	1.2	139
9	1	172	0.431	-5.04	2.97	1	4	1	145
10	1	172	0.431	-5.05	2.96	1	4	1	150
11	1	172	0.431	-5.05	2.96	0	4	0	155
12	1	172	0.431	-5.05	2.96	0	4	0	155

The performance curve is shown in Figure 6.37. Here the abbreviations are similar to those used in the λ adaptations. Stable is the first input of the stability system. Con Max is the control effort as defined in Section 6.2.1.3, and Ci is the control index, i.e. the input to the U_{max} system. The Per column is the performance in dB. The next four columns are the outputs of the U_{max} , derivative, stability and metarules systems.

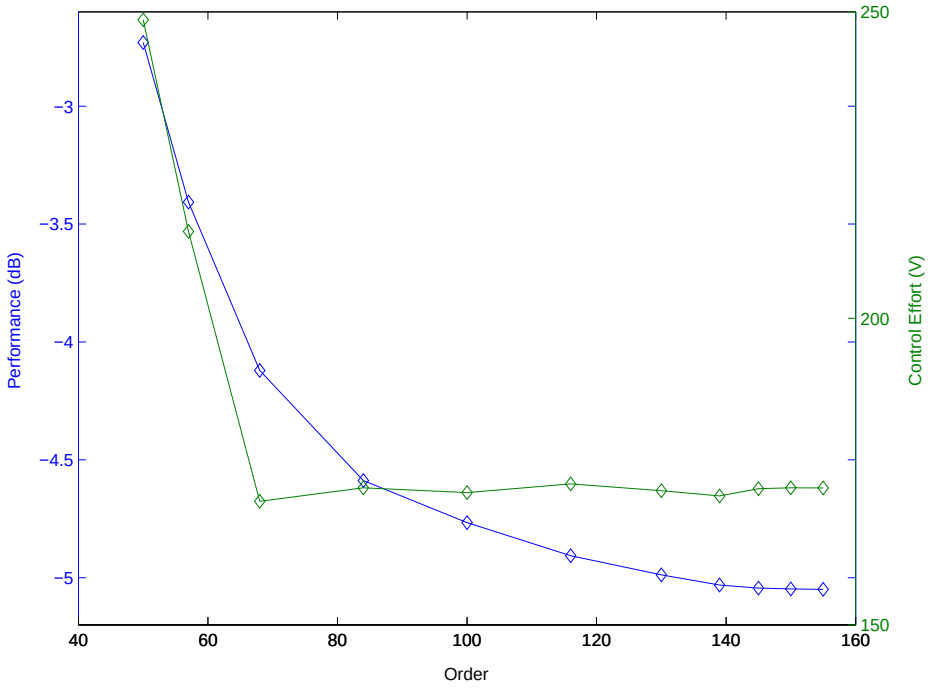


Figure 6.37: Performance during Iteration - Case 1

For this particular case, we see that the derivative factor goes to 0 after the 11th iteration. This causes the metarules output to also go to 0, thereby halting the iteration.

A two-dimensional plot of the factors (Figure 6.38) shows which system controls the adaptation at which iteration. During the first five iterations, the metarules output closely follows the U_{max} factor. Just as in the λ adaptations, the metarules choose the more conservative of the U_{max} and derivative factors. After the 6th iteration, the derivative system controls the adaptation.

By the 10th iteration, the derivative of the spline fit is almost within the stopping condition (see Figure 6.39). The derivative system therefore selects the smallest possible change in order, increasing the order to 155. Note that the spline fit has a large error for the early iterations, but a smaller error for later iterations. This is a result of choosing a very small smoothing parameter of 1×10^{-4}. This value works well for the laboratory experiments, and was used here for the sake of consistency. Using a larger smoothing parameter would have reduced the error in initial iterations and also steepened the slope.

After the 11th iteration, the current derivative is well within the stopping region (see Figure 6.40) and the adaptation of order is halted. A comparison of the 11th iteration spline to the 10th

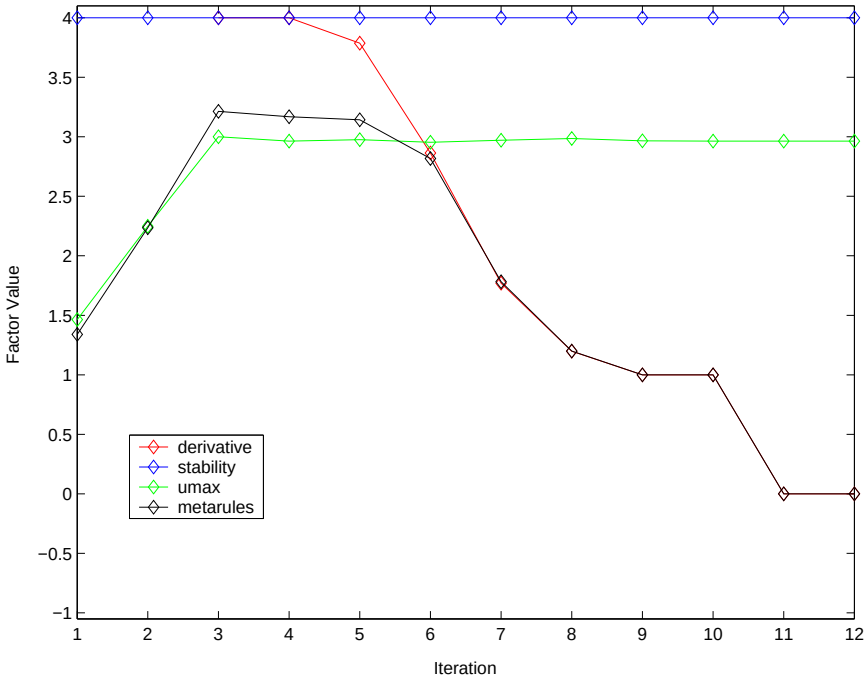


Figure 6.38: Factors - Case 1

iteration spline fit (Figure 6.39) show a dramatic shift in derivative for orders larger than 100 for the 11th iteration. The 11th iteration spline fit seems to indicate that the iteration should have stopped at an order of around 135, but the spline fit from the previous iteration indicates otherwise.

This case illustrates an important point in regard to the U_{max} rules. The control effort curve in Figure 6.41 shows a steep decline in the first two iterations. Since the initial effort levels are so high, the U_{max} system restricts the overall *Factor* to 1.34 and 2.24. The U_{max} system rules were built to stop the adaptation, with the assumption that an increase in order would lead to a further increase in the control effort. The control effort curve in this case shows the complete opposite behavior, and therefore the restrictions in factor are completely unnecessary.

We have shown the adaptation path on the decision surface in Figure 6.42. The near-constant control effort level after the 3rd iteration leads to a path that remains parallel to the derivative factor axis. This path is very similar to the path in the second experimental case (Figure 6.58).

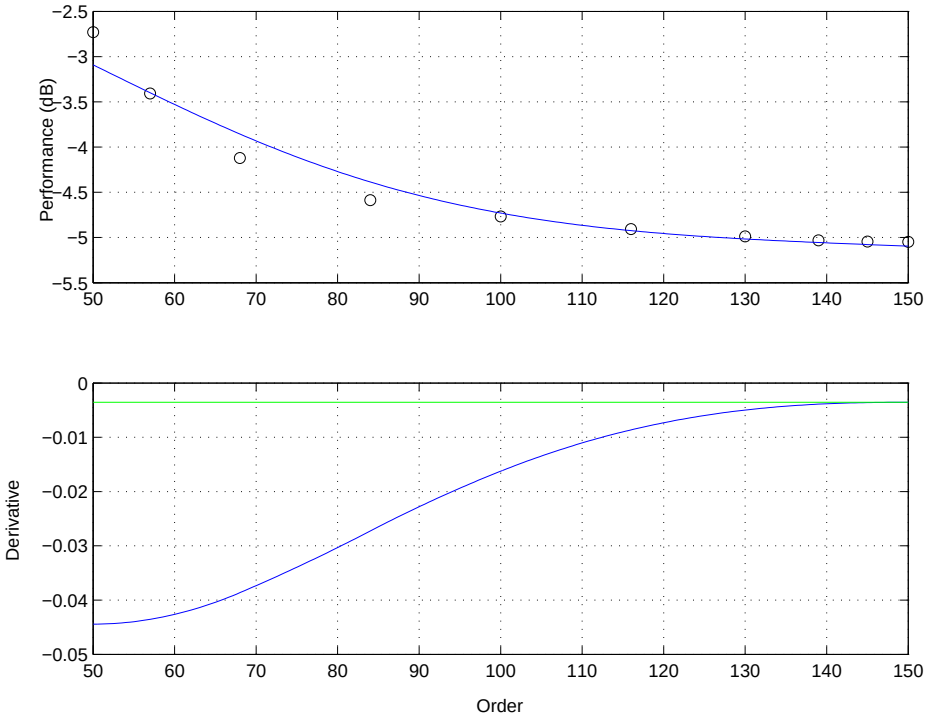


Figure 6.39: Spline Fit - Iteration 10 - Case 1

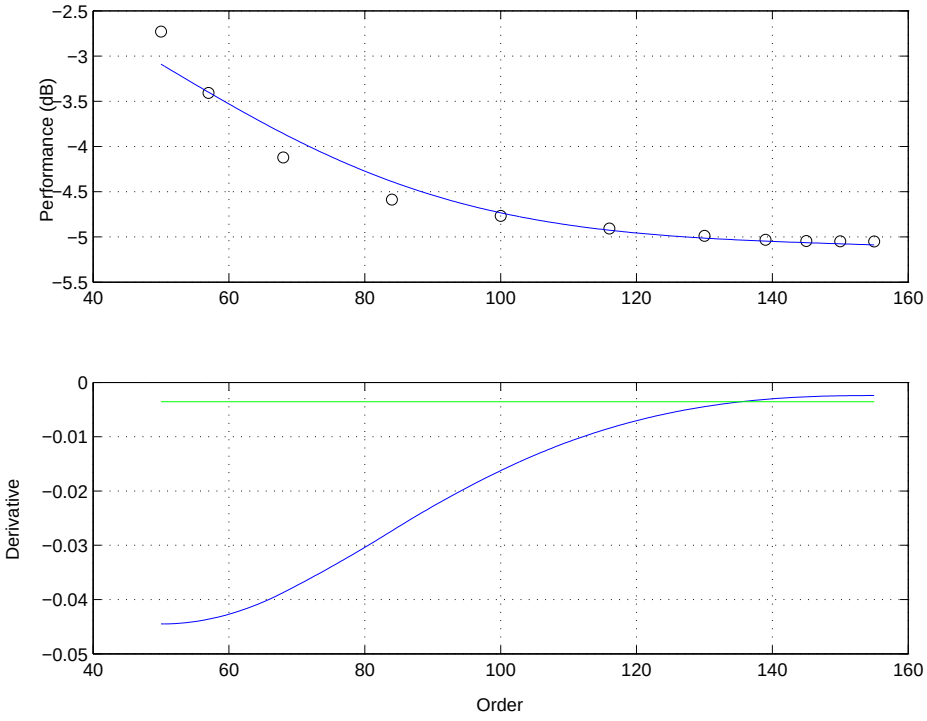


Figure 6.40: Spline Fit - Iteration 11 - Case 1

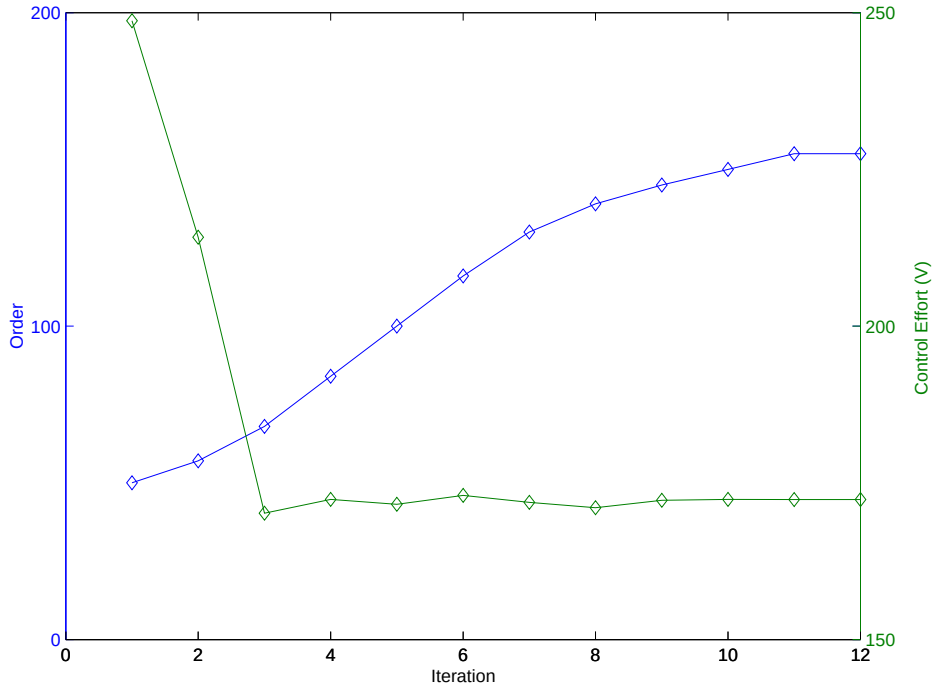


Figure 6.41: Order during Iteration - Case 1

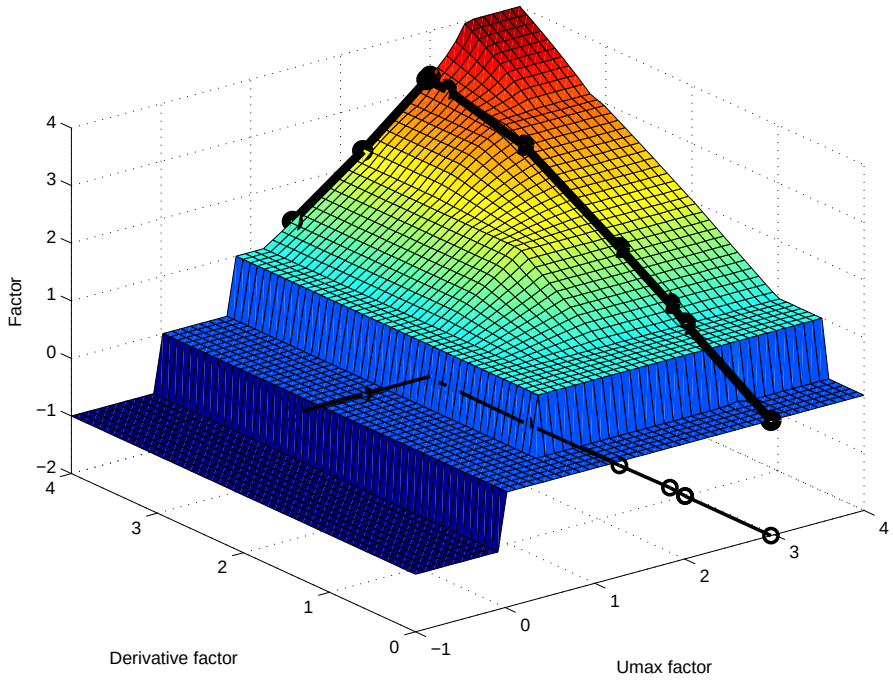


Figure 6.42: Decision Surface - Case 1

6.2.2.2 Case 2

The settings for this case are nearly identical to those used in case 1. Again, we used the 28-mode system with the acceleration measured at location 4. λ again was set to 0.0389, and the horizons were given values of 300. The step size was set to 5, and 50,000 closed-loop points were collected for each iteration. The adaptation was started with an order of 40. The significant changes were that O_{max} was set to 100, and U_{max} was set to 700V. The adaptations in the first case were stopped by the derivative rules with the final order of 155. Since our O_{max} is smaller than this order, this case should show us how well the new stability rules work. The results of the adaptation are:

Table 6.8: Case 2 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	1	447	0.638	0.0112	1.32	NaN	4	1.19	40
2	1	219	0.312	-1.32	3.6	NaN	4	4	46
3	1	173	0.247	-4.16	3.9	4	4	4	66
4	1	172	0.245	-4.6	3.91	3.91	4	3.91	86
5	1	174	0.248	-4.8	3.9	2.68	-1	-1	106
6	1	171	0.245	-4.77	3.91	2.57	-0.5	-1	101
7	1	173	0.247	-4.76	3.9	2.67	0	0	96
8	1	173	0.247	-4.76	3.9	2.67	0	0	96

The performance curve is shown in Figure 6.43. A plot of the factors is shown in Figure 6.44, clearly demonstrating what is occurring in this case.

The control effort is so high for the 1st iteration that the U_{max} system restricts the factor to 1.19. The effort dramatically drops during the 2nd iteration, and the metarules output becomes 4 or nearly 4 for the next three iterations. A controller with order 106 is designed and implemented for the 5th iteration. After the 5th iteration, the stability module recognizes that the order ratio $\left(\frac{O_{current}}{O_{max}}\right)$ is larger than 1, i.e. it is *oversize*. This causes the 8th rule of the stability module to fire, and *action* becomes *revert*. The metarules system then chooses to decrease the order by one step size.

After the 6th iteration, the order ratio is still *oversize* and rule 8 is fired. Since the system is still stable and *previous* is *revert*, the second rule is also fired, which attempts to make *action* equal to *hold*. The output of the stability system therefore becomes $\frac{-1+0}{2} = -0.5$, a half-step decrease

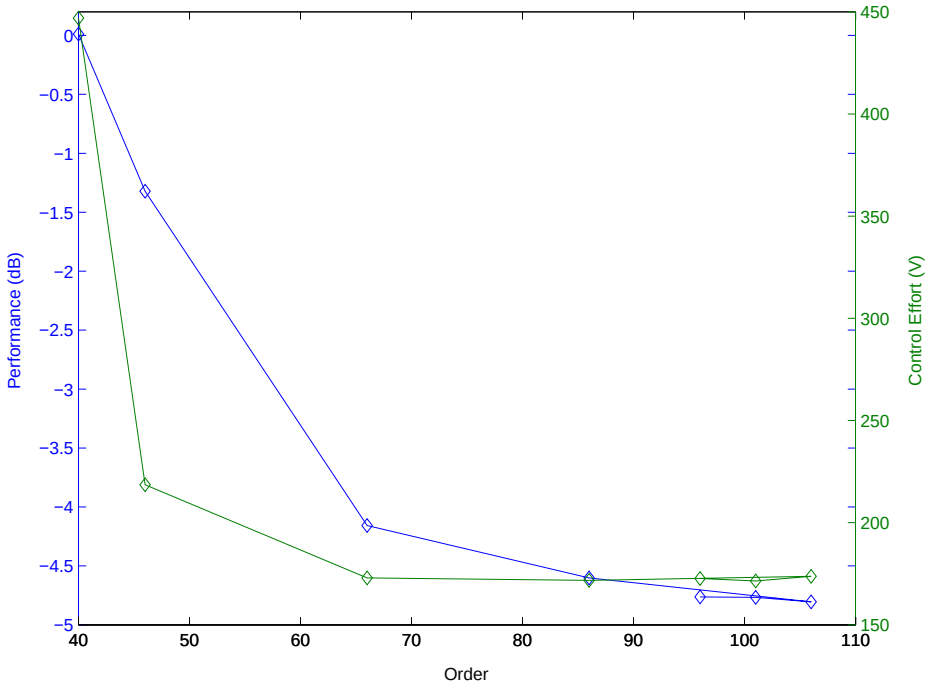


Figure 6.43: Performance during Iteration - Case 2

in order. We set out to create a set of rules that does not make a half-step change in order, so this output of the stability module represents an unforeseen mistake. The metarules module corrects this mistake and outputs the correct value of *revert*, which is -1.

The 7th iteration is then completed with an order of 96, and remains stable. This causes the output of the stability system to be *hold* ($Factor = 0$), and the adaptation is halted.

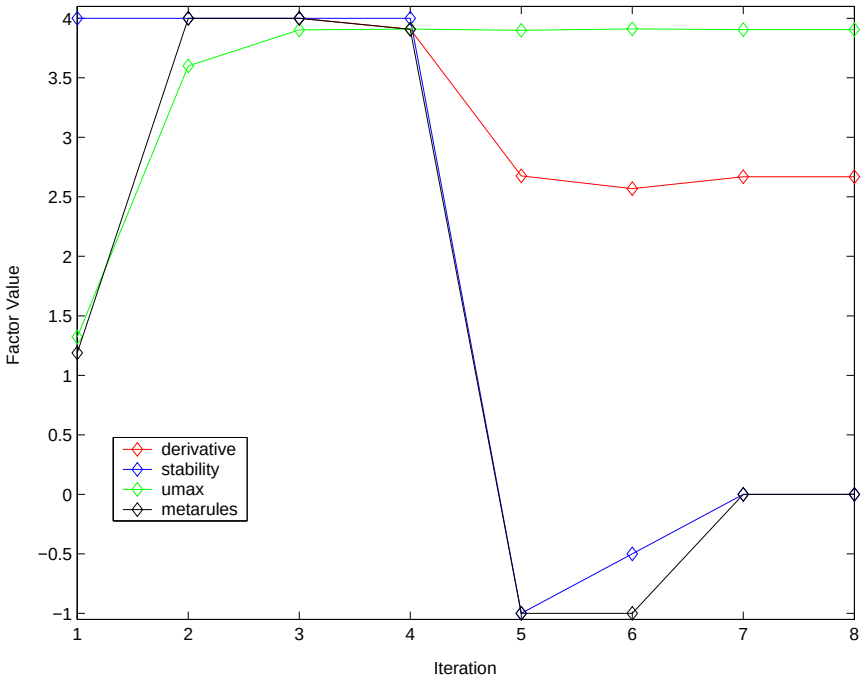


Figure 6.44: Factors - Case 2

6.2.2.3 Case 3

We will now demonstrate a case in which the system becomes unstable. We have chosen to simulate the five-mode system with λ fixed at 1×10^{-5}, $U_{max} = 1200$, a step size of 5, O_{max} set to 150, and horizons set to 80. The acceleration was measured at the third measurement location, and 50,000 samples were collected for each iteration. The results of the adaptation are:

Table 6.9: Case 3 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	0	NaN	0	NaN	4	NaN	4	4	10
2	1	152	0.127	-9.06	4	NaN	4	4	30
3	1	375	0.312	-9.33	3.6	NaN	4	4	50
4	1	627	0.523	-9.46	2.39	4	4	2.43	70
5	1	735	0.613	-9.51	1.53	3.92	4	1.41	82
6	1	799	0.666	-9.53	1.15	3.75	4	1.01	89
7	1	774	0.645	-9.54	1.27	3.53	4	1.13	94
8	1	774	0.645	-9.56	1.27	3.22	4	1.13	100
9	0	NaN	0	NaN	4	3.22	-1	-1	106
10	1	838	0.698	-9.56	1.14	3.03	0	0	101
11	1	838	0.698	-9.56	1.14	3.03	0	0	101

The performance curve is shown in Figure 6.45.

The adaptation starts with an order of 10. Closing the loop during the first iteration causes an instability. Since the order ratio is less than 30% ($ratio = small$), the fourth rule of the stability system is fired and *action* becomes iterate. Since the system has become unstable, we set the control index to 0, thereby making the U_{max} factor equal to 4. This then causes the metarules to output a factor of 4 and the order becomes 30.

In Figure 6.46, we see that the metarules output closely follows the U_{max} output for the 2nd through 8th iterations. The U_{max} system is slowing down the adaptation rate as the control effort continues to increase (see Figure 6.45). After the 8th iteration, the overall factor becomes 1.13 and order is increased to 106.

The implementation of this controller order becomes unstable during the 9th iteration. The order ratio is now $\frac{106}{150} \approx 0.7$, which will be evaluated as *large* in the stability system rules. The seventh rule of the stability system is fired and *action* becomes *revert*. This action causes the

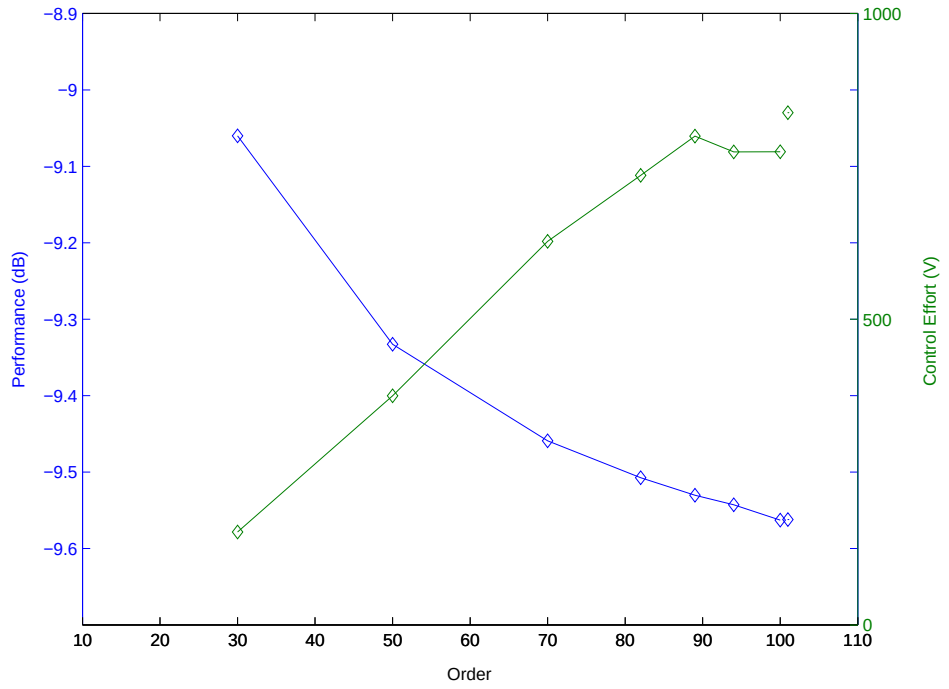


Figure 6.45: Performance during Iteration - Case 3

metarules to also output *revert* and *Factor* becomes -1.

For the 10th iteration, a 101-order controller is implemented and the resulting closed-loop system stays stable. This means that *stable* is *yes*, *previous* is *revert*, and the second rule of the stability system fires. This makes *action* equal to *hold* and the metarules output becomes 0. After the 11th iteration, *stable* is *yes*, *previous* is *hold*, and the third rule of the stability module is fired. This makes *action* equal to *hold* and the metarules output equal to 0. The adaptation of order is now complete.

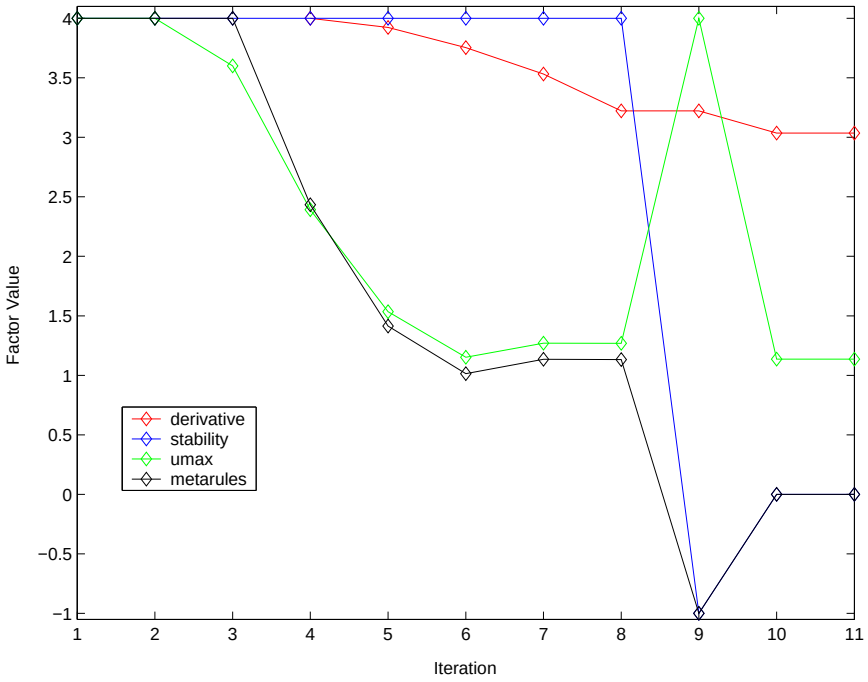


Figure 6.46: Factors - Case 3

6.2.2.4 Case 4

For this final case, we have chosen a system similar to that in the previous case. The goal of this case is to show an order adaptation in which the U_{max} system stops the adaptations. We therefore have chosen a system in which the control effort increases as the order is increased.

We have run the simulation on the five-mode system with $\lambda = 3.46 \times 10^{-5}$, the maximum order set to 150, $U_{max} = 650\,V$, the horizons set to 80, 50,000 samples per iteration, a step size of 5, and the acceleration measured at location three. The results of the adaptation are:

Table 6.10: Case 4 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	0	NaN	0	NaN	4	NaN	4	4	10
2	1	133	0.205	-9.01	4	NaN	4	4	30
3	1	283	0.436	-9.27	2.93	NaN	4	3.1	50
4	1	390	0.599	-9.38	1.63	4	4	1.52	66
5	1	425	0.654	-9.41	1.21	4	4	1.07	74
6	1	442	0.68	-9.44	1.14	3.96	4	1	79
7	1	447	0.688	-9.45	1.14	3.86	4	1	84
8	1	454	0.699	-9.46	1.14	3.68	4	1	89
9	1	465	0.716	-9.47	0	3.32	4	0	94
10	1	465	0.716	-9.47	0	3.32	4	0	94

The performance curve is shown in Figure 6.47. The table shows that the adaptation starts with one unstable iteration and is stopped by the U_{max} system after the 9th iteration.

In Figure 6.48, we have shown the various factors that control the iteration. The metarules output almost exactly follows the U_{max} system output. We see the U_{max} output sharply drop from a factor of 4 to a factor less than 1.5 in three steps. This is due to the steep slope of the control effort curve (Figure 6.47) for the first three stable iterations.

At this point, the slope of the control effort curve dramatically decreases, and the U_{max} system outputs a factor of near 1 for four iterations. By the 9th iteration, the control index (CI) becomes large enough for the U_{max} system to stop the adaptation.

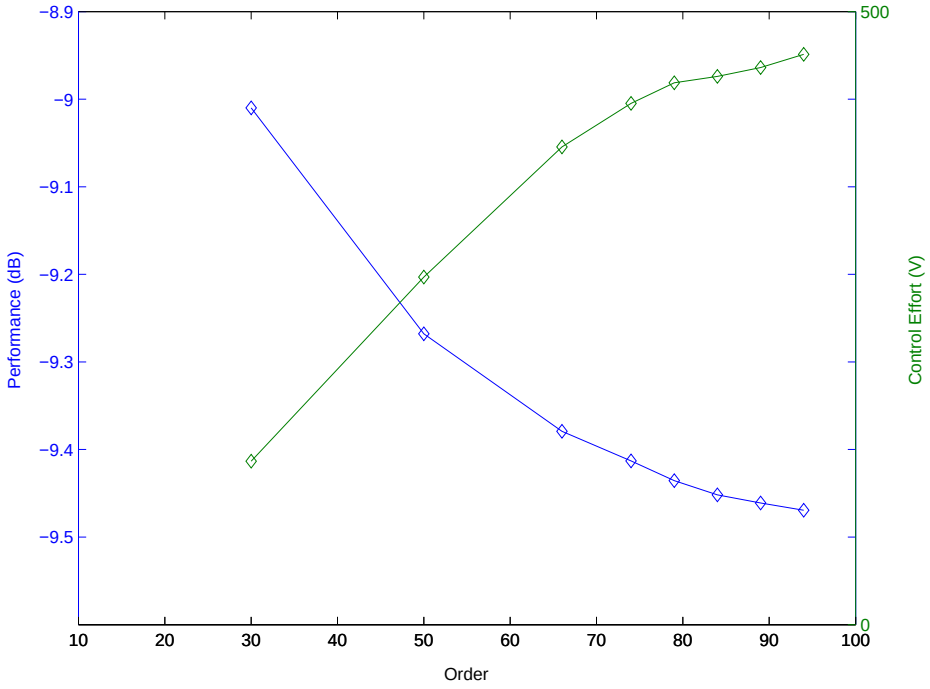


Figure 6.47: Performance during Iteration - Case 4

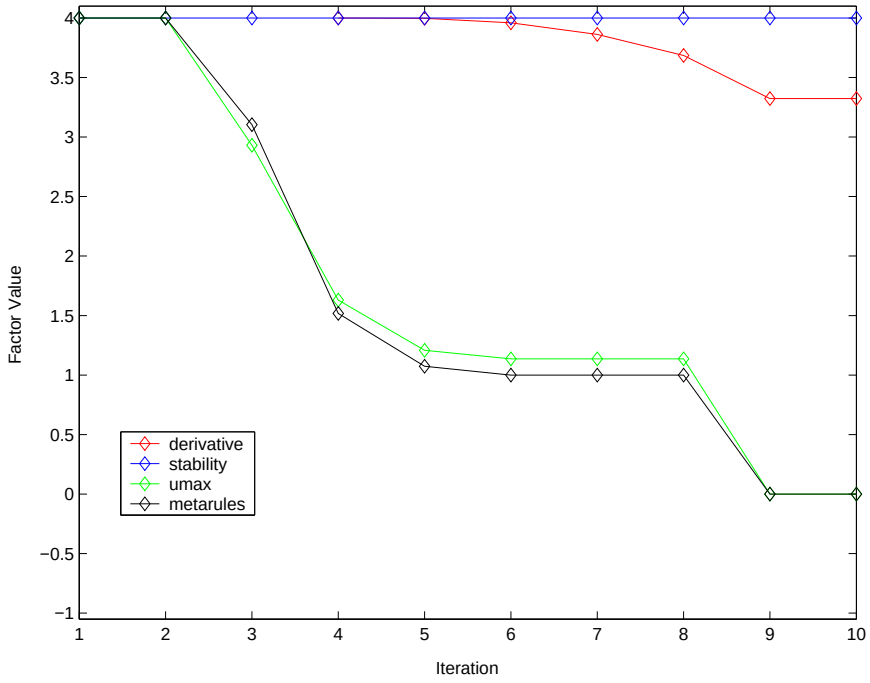


Figure 6.48: Factors - Case 4

6.2.3 Experimental Results

Most of the experimental order adaptations had the same outcome. The experiments were run with λs that were know to have at least 2 or 3 dB of performance. The performance curves all had a general downward trend that levels off. The control effort would decrease or remain at about the same level, and therefore the U_{max} system never stopped the adaptation. In most cases, therefore, the adaptations were stopped by the derivative system.

We have chosen three cases that are representative results of all of the experiments. The first case is an adaptation that is completely controlled by the derivative system. For the second case, the control effort is large enough that the U_{max} system controls the early iterations, but the adaptation is stopped by the derivative system. In the third case, we have made the maximum allowable order small, which dominates the overall adaptation.

6.2.3.1 Case 1

For this case, the accelerometer was attached to location 3, λ was fixed to 0.298, O_{max} was set to 300, U_{max} was set to 2, 70,000 closed-loop data points were collected for each iteration, and the horizons were set to 300. The adaptation was started with an order of 50, which was believed to be below the true order of the system. The results of the adaptation were:

Table 6.11: Case 1 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	0	NaN	0	NaN	4	NaN	4	4	50
2	1	0.608	0.304	-3.17	3.63	NaN	4	4	70
3	1	0.546	0.273	-3.63	3.79	NaN	4	4	90
4	1	0.526	0.263	-3.83	3.83	4	4	4	110
5	1	0.532	0.266	-4.03	3.82	3.87	4	3.86	130
6	1	0.551	0.276	-3.97	3.77	2.02	4	2.03	149
7	1	0.536	0.268	-4.03	3.81	1.43	4	1.43	159
8	1	0.538	0.269	-4.06	3.81	1.27	4	1.27	166
9	1	0.532	0.266	-4.11	3.82	1.49	4	1.49	172
10	1	0.536	0.268	-4.1	3.81	1.26	4	1.27	179
11	1	0.544	0.272	-4.05	3.79	0	4	0	185
12	1	0.54	0.27	-4.12	3.8	0	4	0	185
13	1	0.539	0.269	-4.14	3.8	0	4	0	185

The performance curve is shown in Figure 6.49. The table clearly shows that the adaptation is halted after the 11th iteration by the derivative system.

The adaptation starts with a controller of the 50th order that results in an unstable closed-loop system. Since the order ratio is well below 0.3, it is considered *small* when input into the stability system and *action* becomes *iterate*. The control index is assigned the value of 0 for an instability and metarules selects a factor of 4.

A plot of the four factors, shown in Figure 6.50, shows how the adaptation continues. The metarules output is almost identical to the derivative factor throughout the adaptation. After the 6th iteration, the derivative module senses that the performance curve is starting to flatten out and starts to slow down the adaptation rate. The rate at which the performance curve changes slops is dramatic and causes the derivative factor to drop from 4 to 1.43 in three iterations.

After the 11th iteration, the current derivative of the spline fit (Figure 6.51) is less than 8% of the minimum. This makes the output of the derivative system 0, and the metarules module halts the iteration. Although the adaptation was halted, we allowed the experiment to continue for another two iterations. The performances of the next two iterations were slightly better than the 11th iteration. Our spline fit algorithm still deletes the double values, and therefore these values had no impact on the adaptation. If we had instead used the mean performance for a point that used the same order, then the iteration would have continued.

The drastic drop in the metarules factor is also evident in the slope of the order curve of Figure 6.52 . The slope changes dramatically at the 6th iteration. In an ideal adaption, we would like to see this slope change smoothly, but the performance curve flattens out so drastically in this case that it was simply impossible for this to happen. Nevertheless, the fuzzy logic rules we have designed still performed their task.

U_{max} was set high enough to make the U_{max} factor above 3.6 for the entire adaptation, thereby never playing a role in this adaptation. This is evident in the decision surface of Figure 6.53, as the adaptation moves down the right hand edge of the surface.

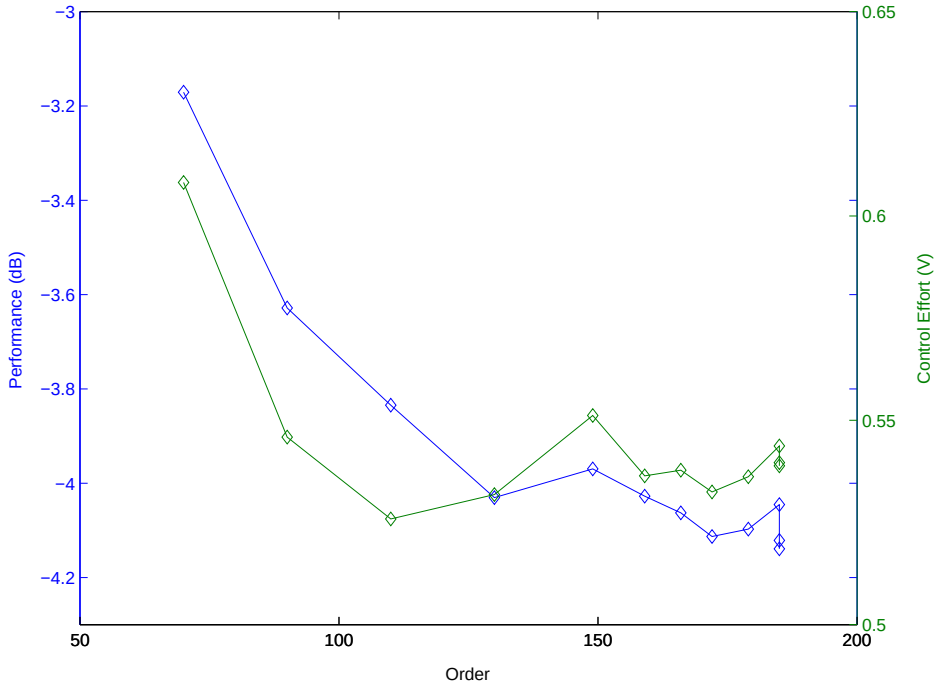


Figure 6.49: Performance during Iteration - Case 1

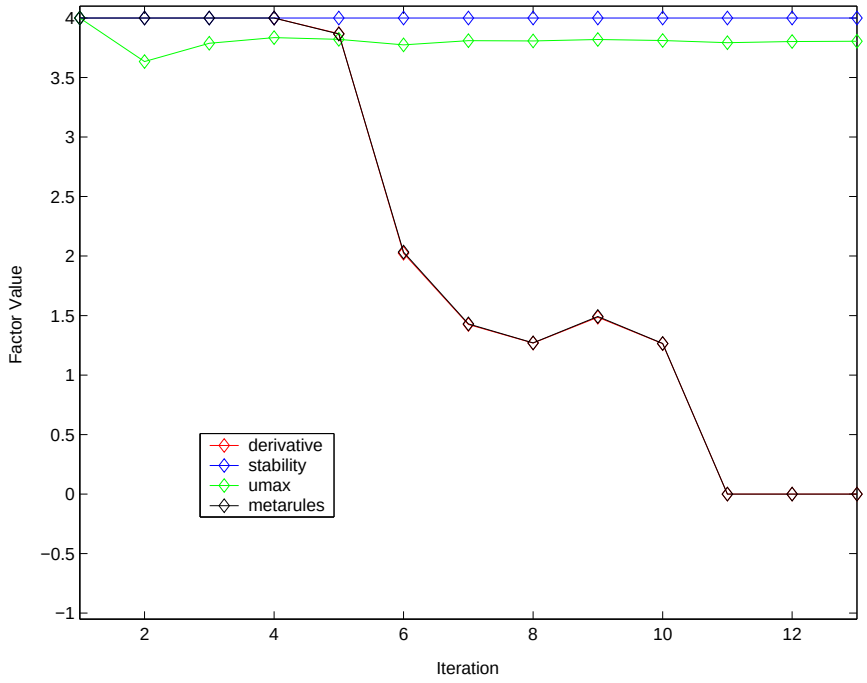


Figure 6.50: Factors during Iteration - Case 1

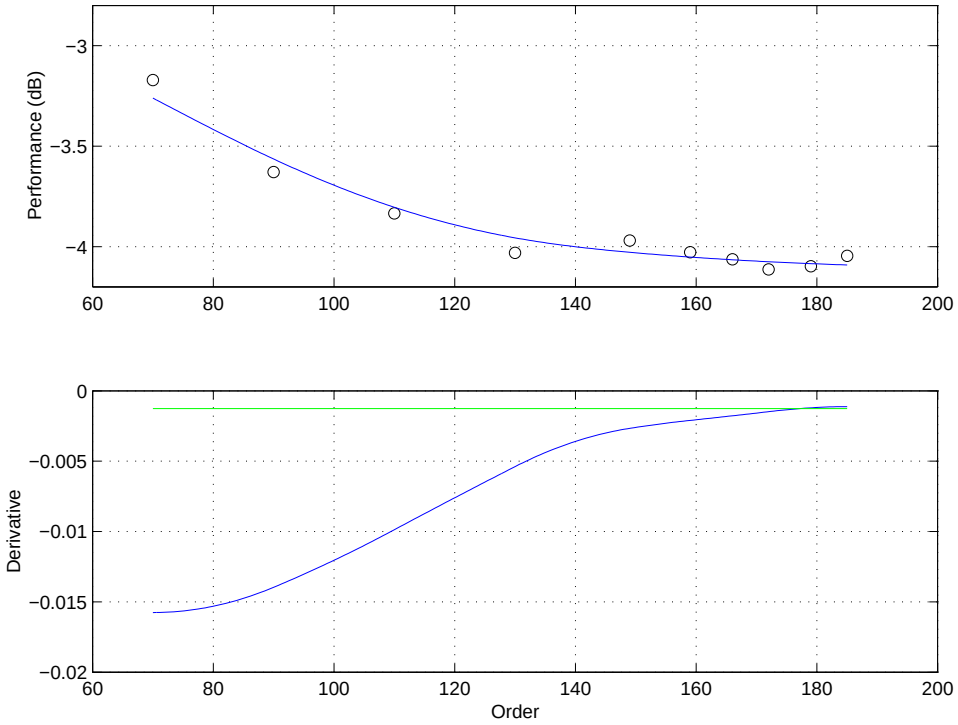


Figure 6.51: Spline Fit - Iteration 11 - Case 1

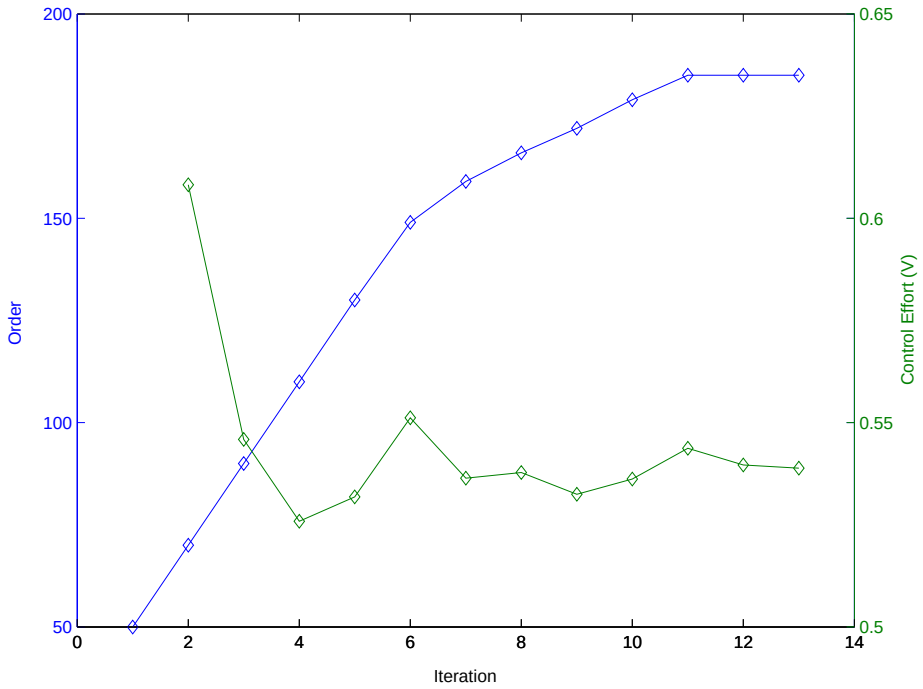


Figure 6.52: Order during Adaptation - Case 1

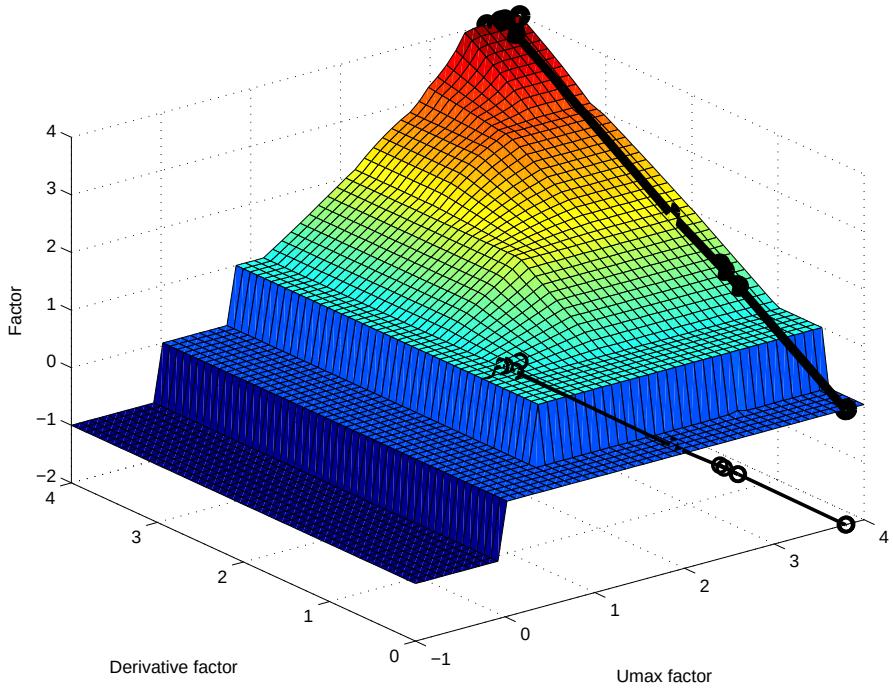


Figure 6.53: Decision Surface - Case 1

6.2.3.2 Case 2

For this case, we measured the acceleration at measurement location 2 and fixed λ at 0.5. The horizons were fixed to 300, U_{max} was assigned 0.8, O_{max} was given a value of 300, and 80,000 closed-loop points were collected for each iteration. The adaptation was started with an order of 60. The results of the adaptation are:

Table 6.12: Case 2 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	1	0.396	0.495	-2.59	2.63	NaN	4	2.71	60
2	1	0.398	0.497	-3.02	2.62	NaN	4	2.69	74
3	1	0.413	0.516	-3.52	2.46	4	4	2.52	87
4	1	0.406	0.507	-3.45	2.54	3.98	4	2.61	100
5	1	0.414	0.518	-3.47	2.44	3.34	4	2.5	113
6	1	0.416	0.521	-3.78	2.41	3.47	4	2.46	125
7	1	0.438	0.548	-3.8	2.1	2.98	4	2	137
8	1	0.439	0.549	-3.83	2.09	2.35	4	2	147
9	1	0.436	0.546	-3.91	2.12	2.05	4	2	157
10	1	0.431	0.539	-3.98	2.21	1.93	4	1.95	167
11	1	0.43	0.537	-3.89	2.23	1.01	4	1.01	177
12	1	0.421	0.526	-3.97	2.35	1.02	4	1.02	182
13	1	0.419	0.524	-3.91	2.38	0	4	0	187
14	1	0.428	0.535	-3.87	2.25	0	4	0	187

The corresponding performance curve is shown in Figure 6.54. The adaptation is halted by the derivative system after the thirteenth iteration.

The derivative system does not control the iteration, however. The plot of the individual factors, Figure 6.55, shows this particularly well. For the first 8 iterations, the metarules output closely follows the output of the U_{max} system. The control index does not significantly change for these iterations, rising from the control index of 0.495 for the 1st iteration, to 0.549 for the 8th iteration. Therefore the output of the U_{max} system stays relatively constant, varying from 2.63 to 2.1.

The decline of the derivative factor (see Figure 6.55) after the 4th iteration is relatively smooth, although the performance curve is not smooth. This is due to the small smoothing parameter of the spline fit, which removes the irregularities of the performance curve. After the 9th iteration, the derivative factor becomes more restrictive than the U_{max} system, and therefore controls the

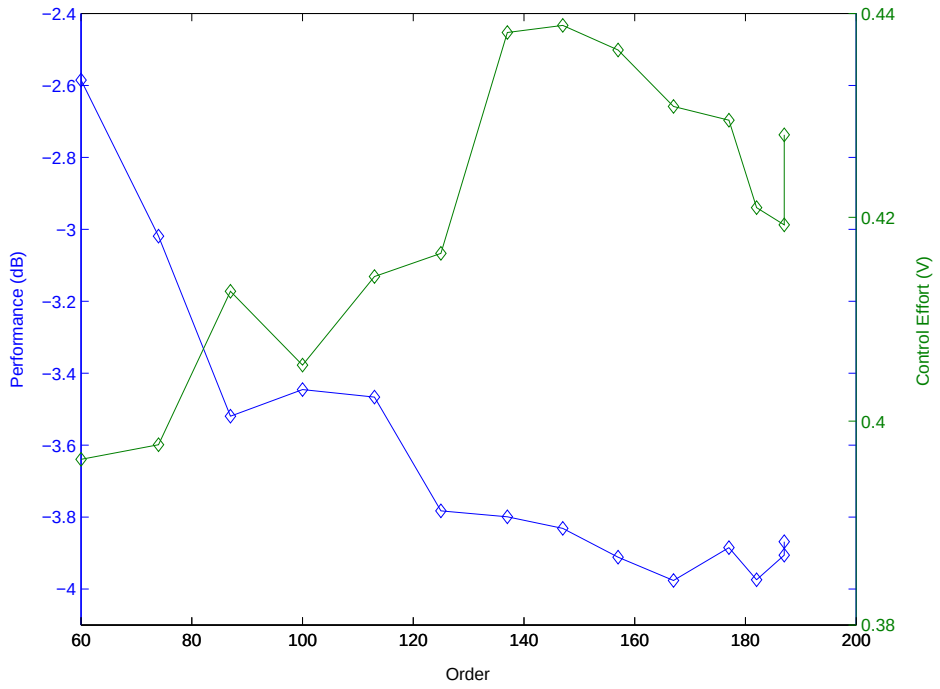


Figure 6.54: Performance during Iteration - case 2

remaining adaptations. The derivative of the spline fit becomes less than 8% of the maximum rate with the addition of the 13th iteration, and this is shown in Figure 6.56. This leads to a derivative and metarules factor of 0, thereby halting the adaptation.

The progression of the metarules output, from an initial value of 2.71 to its final value of 0, is gradual (see Figure 6.55). This leads to gradual changes in order (Figure 6.57), evident in the smooth change in slope of the order curve.

The factors are superimposed on the decision surface in Figure 6.58. The near constant-control effort leads to an adaptation path that stays nearly parallel to the derivative factor axis. This path is similar to that of the first simulation case (see Figure 6.42).

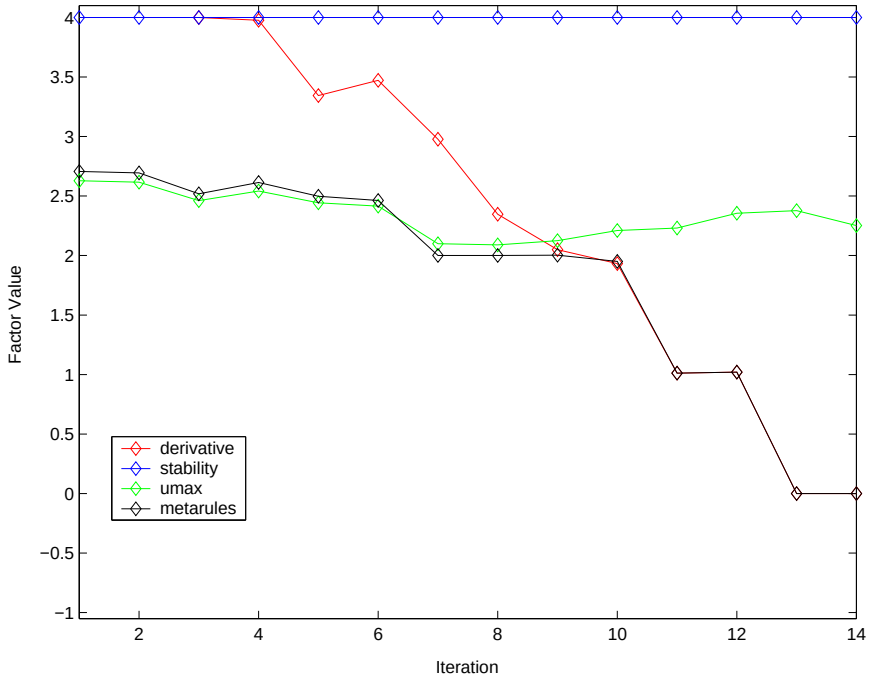


Figure 6.55: Factors - Case 2

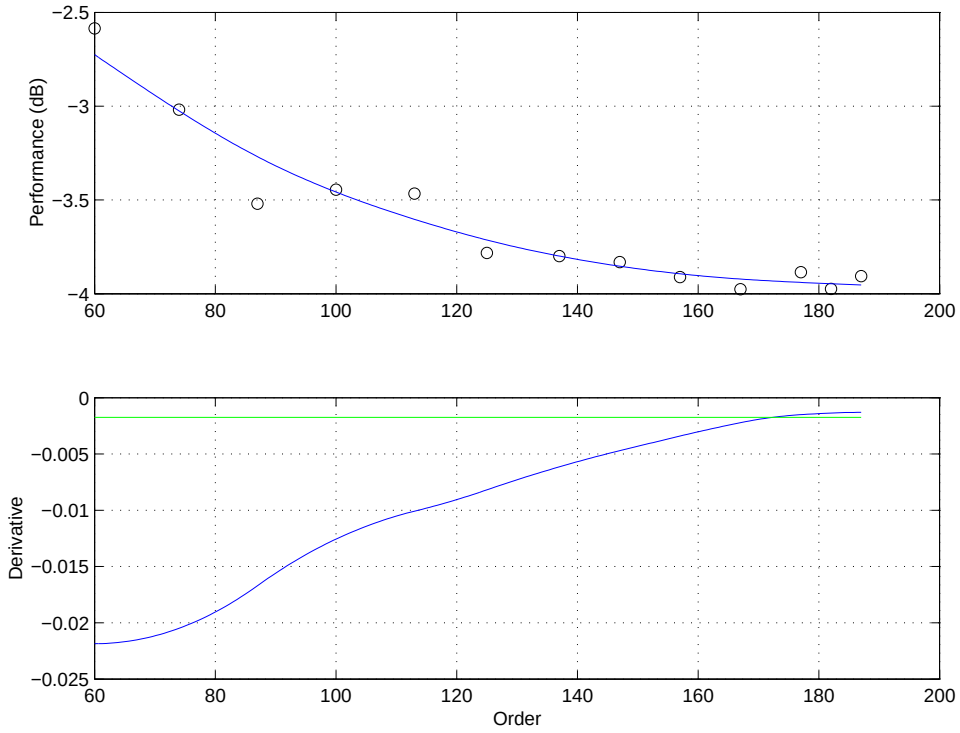


Figure 6.56: Spline Fit - Iteration 13

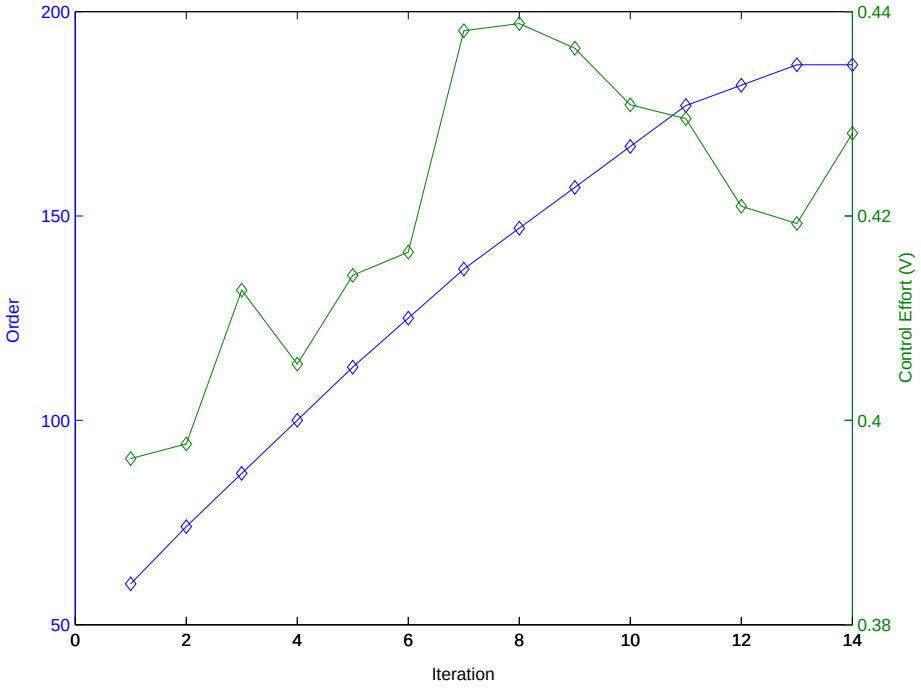


Figure 6.57: Order during Iteration - Case 2

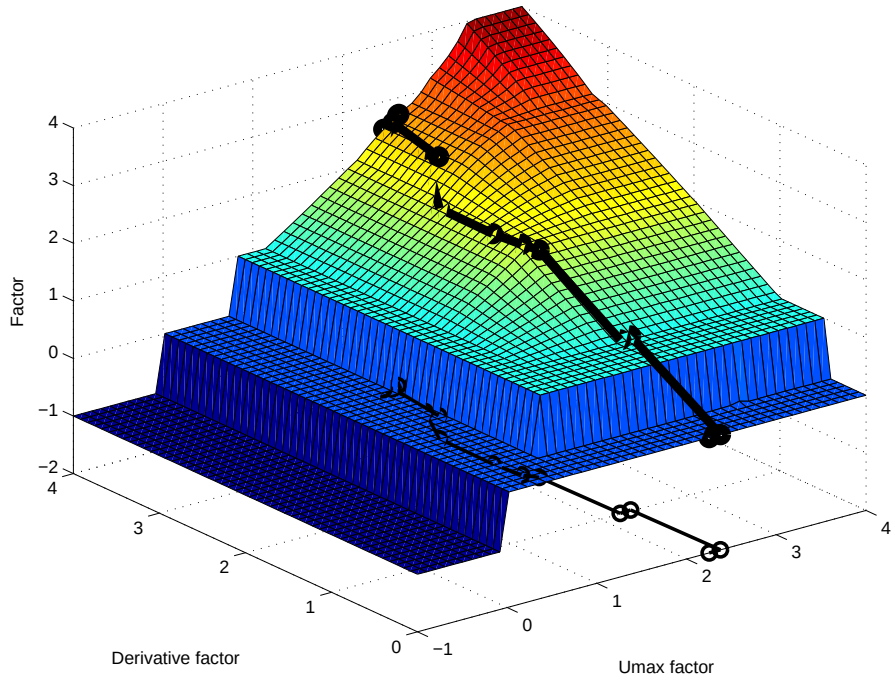


Figure 6.58: Decision Surface - Case 2

6.2.3.3 Case 3

In this case, we will demonstrate an adaptation in which the maximum allowable order stops the adaptation.

The accelerometer was mounted to measurement location 1 and λ was fixed to 0.01. U_{max} was set to 8, the horizons were given values of 300, and 40,000 closed-loop points were collected. The maximum allowable order was set to 160, and the adaptation was started with an order of 90.

The results of the adaptation are:

Table 6.13: Case 3 - Iteration Results

Iteration	Stable	Con Max (V)	Ci	Per. (dB)	Umax-F	Der-F	Stability F	Factor	Order
1	1	1.67	0.209	-6.18	4	NaN	4	4	90
2	1	1.62	0.202	-6.76	4	NaN	4	4	110
3	1	1.62	0.203	-6.9	4	3.99	4	3.99	130
4	1	1.63	0.204	-7.06	4	3.64	4	3.62	150
5	1	1.7	0.212	-7.15	3.99	2.78	-1	-1	168
6	1	1.64	0.205	-7.06	4	2.52	-0.5	-1	163
7	1	1.66	0.207	-7.1	4	2.57	0	0	158
8	1	1.68	0.21	-7.05	4	2.57	0	0	158
9	1	1.7	0.212	-7.1	3.99	2.57	0	0	158
10	1	1.69	0.212	-6.99	4	2.57	0	0	158
11	1	1.67	0.209	-7.03	4	2.57	0	0	158

The corresponding performance curve is shown in Figure 6.59.

The adaptation starts with 3 iterations with the metarules output of 3.99 or higher (Figure 6.60). With the 4th iteration, the derivative rules start to slow down the adaptation with a factor of 3.62 sent to the metarules.

For the 5th iteration, a 168th-order controller is implemented, and closed-loop data is collected. The stability module then determines that the order ratio ($\frac{O_{current}}{O_{max}}$) is *oversize*, i.e. the ratio is larger than 1. This causes the 8th rule of the stability module to fire and its output *action* becomes revert. This makes the output of the metarules -1, and a 163-order controller is implemented for the 6th iteration.

The 6th iteration stays stable, but the order is still *oversize*, i.e. order is larger than the maximum allowable. The second and eighth rules of the stability module are fired and the output of

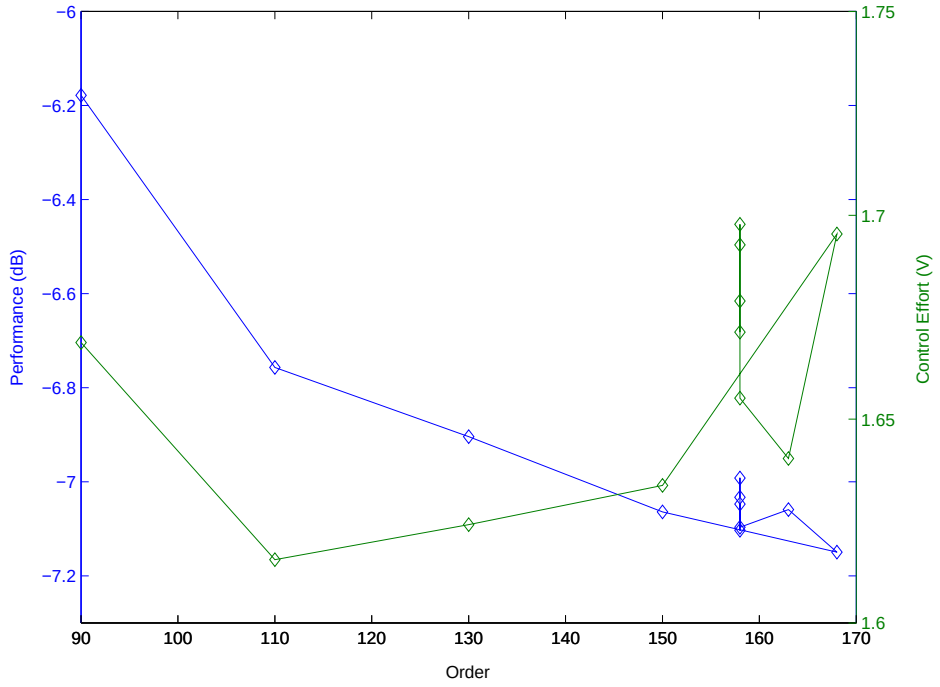


Figure 6.59: Performance during Iteration - Case 3

the stability module becomes -0.5. This situation is exactly what happened in the numerical Case 2. Just as in the numerical case, the metarules module corrects the mistake and correctly outputs a factor of -1.

A 158-order controller is implemented for iteration 7 and stays stable. The stability system then holds this value until the experiment is stopped.

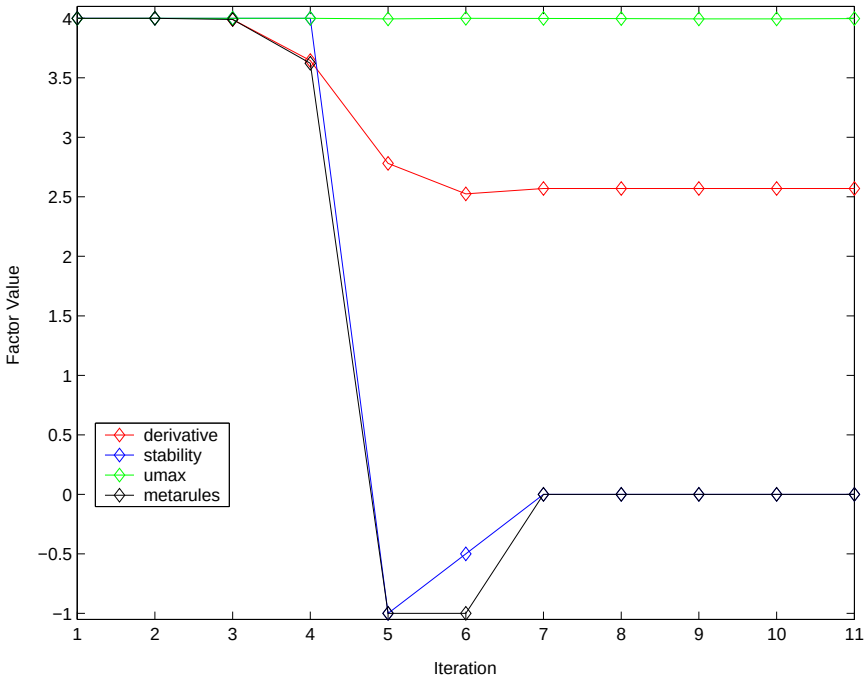


Figure 6.60: Factors - Case 3

Chapter 7

Conclusion and Further Work

7.1 Conclusions

This work provided an analysis of the effect of the four GPC design parameters on the closed-loop disturbance rejection and the corresponding control effort on a vibrating plate. Based on this knowledge, we developed an automatic method for tuning λ and order in order to optimize performance, without violating a maximum control effort constraint.

The investigation started by characterizing the effect of control and prediction horizons on the performance. It was shown that increasing the horizon beyond a certain point does not further improve performance. Furthermore, this point was identical when the experiments were repeated for different λ and orders. Hence the performance can be optimized by setting the horizons to a static value, which was shown to be about two to three times the true plant order. This in effect removes these two parameters from the tuning space, and they do not need to be adapted with the fuzzy logic rules.

Next, the effect of λ and order on the performance and control effort was calculated for both the numerical simulation and the laboratory setup. The effect proved to be complex, although it was possible to group the behavior into finite sets. For a λ adaptation, i.e. fixed order, it was shown that the performance curve could be fit into one of three categories, and control effort curves into four categories. In general, the performance initially improves for reduction in λ, and the control effort increases. For order adaptation, i.e. fixed order, it was shown that the performance curves fit into three broad categories, and the control effort into three categories. These were somewhat

different than the λ adaptations.

Based on the observed behavior of the performance and control effort curves, we were able to devise a set of fuzzy logic modules in order to provide an automatic adaptation. The adaptation would adjust the parameters in distinct iterations; at each iteration, the closed-loop data of the performance, control effort and stability were collected. This information was then analyzed and fed into the fuzzy logic module in order to provide the next parameter adjustment, which in turn would be used to calculate the next control law.

A set of four fuzzy logic modules was developed for each adaptation case. Three of the modules worked independently to provide an adaptation factor, and the metarules module combined this into a final factor. The first three modules were the stability module, the derivative module, and the control effort module, each of which would look at closed-loop results from a specific perspective. Each had the ability to stop adaptation once a specific condition was met. The modules were designed to rapidly adjust the parameters until it was determined that one of the stopping conditions was near, at which point the adaptation rate was throttled.

We believe that both the adaptation of λ and the adaptation of the order proved successful. The results from the λ adaptation, which was the simpler of the two, showed that the fuzzy logic modules performed as designed. The adaptation rate was successfully throttled as the stopping conditions were approached. Furthermore, the undesirable condition of approaching the settling point in an asymptotic manner with many unnecessary small iterations was avoided by the design features of the fuzzy logic systems. The system also recovered from an unstable closed-loop system.

The fact that such complex behavior could be built into the decision surface was one of the main advantages of choosing the fuzzy logic approach. The ability to create non-linear behavior in the control effort and derivative modules to obtain the desired adaptation rates was easily accomplished with fuzzy logic. This would have been difficult with any other method. Furthermore, the ability to create an exception in the rules to allow further adaptation in cases where the performance benefit is minimal, but the control effort is extremely low, was also simple to incorporate with fuzzy logic. Fuzzy logic also proved beneficial in its ability to monitor and combine the various stopping conditions simultaneously.

The order adaptation proved more complex because of the large variety of observed behaviors.

For instance, the system could be unstable for orders that were too high and others that were too low. The control effort could decrease, increase, or remain constant as order was increased. The experiments showed that this could unnecessarily limit the adaptation rate, although the rules were designed for the worst case scenario – increases in effort for increases in order. The adaptations were successful, although a bit conservative in some cases.

7.2 Future Work

Future work could be performed on three main areas. First, it was shown that in some cases, a closed-loop instability could be predicted. Whether these types of controllers should be implemented could be a point of discussion. The most conservative approach would be not to implement them. Obviously, this would necessitate another module that examines the control law before it is implemented, and then overrides the implementation if it is unstable.

A second area of future work is to combine the λ and order adaptations into one fuzzy logic system. It's not exactly clear how this should be done, but one possible solution would be to keep order fixed initially and then adapt λ. Then, as one of the stopping conditions is encountered, the λ adaptations would be turned off and the order adaptations would start. This could result in improvement performances, or control effort increases (or decreases) until the optimal point is reached. A detailed examination of the performance surface in Section 5.2 shows that this could just as easily not work, with the final stopping point not being optimal in any sense.

A third area of future investigation would be to use the fuzzy logic system as-is on another type of vibrating system, for instance, vibration on a cylinder or a structure with a free end. The fuzzy logic rules might need to be altered in order to work on such a system.

References

[1] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control part I - the basic algorithm. *Automatica*, 23(2):137–148, 1987.

[2] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control part II - extensions and interpretations. *Automatica*, 23(2):149–160, 1987.

[3] K. J. Astrom and B. Wittenmark. On self tuning regulators. *Automatica*, 9(2):185–199, 1973.

[4] D. W. Clarke and P. J. Gawthrop. Self-tuning controller. *Proceedings of the Institution of Electrical Engineers*, 122(9):929–934, 1975.

[5] D. W. Clarke and P. J. Gawthrop. Self-tuning control. *Proceedings of the Institution of Electrical Engineers*, 126(6):633–640, 1979.

[6] Jer-Nan Juang and Kenneth Eure. Predictive feedback and feedforward control for system with unknown disturbances. *NASA/TM-1998-208744*, December 1998.

[7] Kenneth W. Eure. *Adaptive Predictive Feedback Techniques for Vibration Control*. PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[8] G. P. Gibbs, K. W. Eure, and J. W. Lloyd. Active control of turbulent boundary layer induced sound radiation from aircraft style panels. In *Proceedings of Active 99, Ft. Lauderdale, FL, December 2 - 4*, pages 837–848, 1999.

[9] Chaung Lin and Jer-Nan Juang. Self-tuning of design variables for generalized predictive control. *NASA/TM-2000-210619*, December 2000.

[10] Jan A. Snyman. Practical mathematical optimization - an introduction to basic optimization theory and classical and new gradient-based algorithms, 2005.

[11] A. R. McIntosh, S. L. Shah, and D. Grant Fisher. Analysis and tuning of adaptive generalized predictive control. *Canadian Journal of Chemical Engineering*, 69(1):97–110, 1991.

[12] A. L. Elshafei, A. Elnaggar, and G. Dumont. Stability and convergence analysis of an adaptive gpc based on state space modeling. In *Proceedings of the 35th IEEE Conference on Decision and Control 1996*, volume 3, pages 3498–3503, 1996.

[13] Riccardo Scattolini and Sergio Bittanti. On the choice of the horizon in long-range predictive control–some simple criteria. *Automatica*, 26(5):915–917, 1990.

[14] Y. Peng and R. Hanus. A tuning strategy for generalized predictive control. *Control Theory and Advanced Technology*, 7(1):153–166, 1991.

[15] J. Zhang. Property analysis of gpc based on coefficient mapping. In *Proceedings of the 13th World Congress, International Federation of Automatic Control*, pages 457–462, 1996.

[16] Y. G. Xi and J. Zhang. Study on the closed-loop properties of gpc. *Science in China (Series E)*, 40(1):54–63, 1997.

[17] J. Zhang and Y. Xi. Brief communication some gpc stability results. *International Journal of Control*, 70(5):831–840, 1998.

[18] Carlos E. Garcia and Manfred Morari. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323, 1982.

[19] D. W. Clarke and C. Mohtadi. Properties of generalized predictive control. *Automatica*, 25(6):859–875, 1989.

[20] C. Mohtadi and D. W. Clarke. Generalized predictive control, lq, or pole-placement: A unified approach. In *Proceedings of the 25th IEEE Conference on Decision and Control 1986*, volume 25, pages 1536–1541, 1986.

[21] D. W. Clarke. Application of generalized predictive control to industrial processes. *Control Systems Magazine, IEEE*, 8(2):49–55, 1988.

[22] R. Isermann, D. Matko, and K. H. Lachmann. *Adaptive Control Systems*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1992.

[23] K. J. Astrom and B. Wittenmark. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1994.

[24] W. J. Rugh and J. S. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

[25] Matthijs Groot Wassink, Marc van de Wal, Carsten Scherer, and Okko Bosgra. Lpv control for a wafer stage: beyond the theoretical solution. *Control Engineering Practice*, 13(2):231–245, 2005.

[26] P. V. Osburn, H. Whitaker, and A. Kezer. New developments in the design of model reference adaptive control systems. In *Proceedings of the IAS 29th Annual Meeting*, New York, 1961.

[27] R. L. Butchart and B. Shackcloth. Synthesis of model reference adaptive control systems by lyapunov's second method. In *Proceedings of the 2nd IFAC Symposium on Theory of Self-Adaptive Control Systems*, pages 145–152, Teddington, England, 1966. Plenum Press.

[28] P. Parks. Liapunov redesign of model reference adaptive control systems. *IEEE Transactions on Automatic Control*, 11(3):362–367, 1966.

[29] G. Goodwin, P. Ramadge, and P. Caines. Discrete-time multivariable adaptive control. *IEEE Transactions on Automatic Control*, 25(3):449–456, 1980.

[30] I. Landau. Unbiased recursive identification using model reference adaptive techniques. *IEEE Transactions on Automatic Control*, 21(2):194–202, 1976.

[31] Tudor Ionescu and Richard Monopoli. Discrete model reference adaptive control with an augmented error signal. *Automatica*, 13(5):507–517, 1977.

[32] I. D. Landau and R. Lozano. Unification of discrete time explicit model reference adaptive control designs. *Automatica*, 17(4):593–611, 1981.

[33] I. D. Landau. A survey of model reference adaptive techniques - theory and applications. *Automatica*, 10(4):353–379, 1974.

[34] C. Hang and P. Parks. Comparative studies of model reference adaptive control systems. *IEEE Transactions on Automatic Control*, 18(5):419–428, 1973.

[35] M. Sunwoo, K. C. Cheok, and N. J. Huang. Model reference adaptive control for vehicle active suspension systems. *IEEE Transactions on Industrial Electronics*, 38(3):217–222, 1991.

[36] Ming-Chang Lin and Jian-Shiang Chen. Experiments toward mrac design for linkage system. *Mechatronics*, 6(8):933–953, 1996.

[37] A. A. Feldbaum. Dual control theory i-ii. *Automation and Remote Control*, 21:874–880, 1033–1039, 1960.

[38] A. A. Feldbaum. Dual control theory iii-iv. *Automation and Remote Control*, 22:1–12, 109–121, 1961.

[39] K. J. Astrom and A. Helmersson. Dual control of an integrator with unknown gain. *Computers & Mathematics with Applications*, 12(6, Part 1):653–662, 1986.

[40] J. Sternby. A simple dual control problem with an analytical solution. *IEEE Transactions on Automatic Control*, 21(6):840–844, 1976.

[41] B. Wittenmark. Adaptive dual control methods: An overview. In *5th IFAC Symposium on Adaptive Systems in Control and Signal Processing*, volume 72, Budapest, Hungary, 1995.

[42] N. M. Filatov and H. Unbehauen. Survey of adaptive dual control methods. *IEE Proceedings - Control Theory and Applications*, 147(1):118–128, 2000.

[43] B. Lindoff, J. Holst, and B. Wittenmark. Analysis of approximations of dual control. *International Journal of Adaptive Control and Signal Processing*, 13(7):593–620, 1999.

[44] Bruce J. Allison, Joe E. Ciarniello, Patrick J. C. Tessier, and Guy A. Dumont. Dual adaptive control of chip refiner motor load. *Automatica*, 31(8):1169–1184, 1995.

[45] B. Wittenmark and C. Elevitch. An adaptive control algorithm with dual features. In *7th IFAC/EFORS Symposium on Identification and System Parameter Estimation*, pages 587–592, York, UK, 1985.

[46] K. J. Astrom, U. Borisson, L. Ljung, and B. Wittenmark. Theory and applications of self-tuning regulators. *Automatica*, 13(5):457–476, 1977.

[47] K. J. Astrom. Theory and applications of adaptive control–a survey. *Automatica*, 19(5):471–486, 1983.

[48] Bjorn Wittenmark and Karl Johan Astrom. Practical issues in the implementation of self-tuning control. *Automatica*, 20(5):595–605, 1984.

[49] P. E. Wellstead, D. Prager, and P. Zanker. Pole assignment self-tuning regulator. *Proceedings of the Institution of Electrical Engineers*, 126(8):781–787, 1979.

[50] K. J. Astrom and B. Wittenmark. Self-tuning controllers based on pole-zero placement. *Control Theory and Applications, IEE Proceedings D*, 127(3):120–130, 1980.

[51] M. J. Grimble. Implicit and explicit lqg self-tuning controllers. *Automatica*, 20(5):661–669, 1984.

[52] D. W. Clarke, P. P. Kanjilal, and C. Mohtadi. A generalized lqg approach to self-tuning control part i. aspects of design. *International Journal of Control*, 41(6):1509–1523, 1985.

[53] D. W. Clarke, P. P. Kanjilal, and C. Mohtadi. A generalized lqg approach to self-tuning control part ii. implementation and simulation. *International Journal of Control*, 41(6):1525–1544, 1985.

[54] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[55] C. R. Cutler and B. L. Ramaker. Dynamic matrix control-a computer control algorithm. In *Proceedings of the Joint Automatic Control Conference*, San Francisco, 1980.

[56] S. Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[57] C. E. Garcia and A. M. Morshedi. Quadratic programming solution of dynamic matrix control (qdmc). *Chemical Engineering Communications*, 46(1):73–87, 1986.

[58] P. Grosdidier, B. Froisy, and M. Hammann. The idcom-m controller. In *Proceedings of the 1988 IFAC Workshop on Model Based Process*, pages 31–36. Pergamon Press, 1988.

[59] P. Marquis and J. P. Broustail. Smoc, a bridge between state space and model predictive controllers: application to the automation of a hydrotreating unit. In *Proceedings of the 1988 IFAC Workshop on Model Based Process*, pages 37–45. Pergamon Press, 1988.

[60] Sohel Anwar. Generalized predictive control of yaw dynamics of a hybrid brake-by-wire equipped vehicle. *Mechatronics*, 15(9):1089–1108, 2005.

[61] E. F. Camacho, M. Berenguel, and C. Bordons. Adaptive generalized predictive control of a distributed collector field. *IEEE Transactions on Control Systems Technology*, 2(4):462–467, 1994.

[62] Kwang-Sung Park, Jin-Man Joo, Jin-Bae Park, Yoon-Ho Choi, and Tae-Sung Yoon. Control of discrete-time chaotic systems using generalized predictive control. In *Proceedings of 1997 IEEE International Symposium on Circuits and Systems*, volume 2, pages 789–792, 1997.

[63] M. Mahfouf, A. J. Asbury, and D. A. Linkens. Unconstrained and constrained generalised predictive control of depth of anaesthesia during surgery. *Control Engineering Practice*, 11(12):1501–1515, 2003.

[64] G. Geng and G. M. Geary. Application of a neural-network-based rls algorithm in the generalized predictive control of a nonlinear air-handling plant. *IEEE Transactions on Control Systems Technology*, 5(4):439–445, 1997.

[65] Kay-Soon Low, Koon-Yong Chiun, and Keck-Voon Ling. Evaluating generalized predictive control for a brushless dc drive. *IEEE Transactions on Power Electronics*, 13(6):1191–1198, 1998.

[66] Jer-Nan Juang. *Applied System Identification*. Prentice-Hall, 1994.

[67] Jer-Nan Juang and Minh Q. Phan. *Identification and Control of Mechanical Systems*. Cambridge University Press, 2001.

[68] Nesbitt W. Hagood, Walter H. Chung, and Andreas von Flotow. Modelling of piezoelectric actuator dynamics for active structural control. *AIAA-90-1087-CP*, 1990.

[69] T. Y. Yang. *Finite Element Structural Analysis*. Prentice-Hall, 1986.

[70] T. L. Turner. Thermomechanical response of shape memory alloy hybrid composites. *NASA/TM-2001-210656*, January 2001.

Appendix A

Hagood Model Calculations

The hard part of using the procedure described by Hagood is in actually doing the calculations. The most intensive calculation are for the mass and stiffness matrix. The mass and stiffness matrices are computed by Equation (A.1).

$$\begin{aligned} M_s &= \int_{V_s} \Psi_r(x)^T \rho_s(x) \Psi_r(x) dV_s & M_p &= \int_{V_p} \Psi_r(x)^T \rho_p(x) \Psi_r(x) dV_p \\ K_s &= \int_{V_s} (L_u \Psi_r(x))^T c_s (L_u \Psi_r(x)) dV_s & K_p &= \int_{V_p} (L_u \Psi_r(x))^T R_s^T c^E R_s (L_u \Psi_r(x)) dV_p \end{aligned} \tag{A.1}$$

where

V_s, V_p	Volume of structure and piezoelectric
Ψ_r	Matrix of assumed shape function for structure
ρ_s	Density of structure
ρ_p	Density of piezoelectric
L_u	Elastic differential operator
c_s	Stiffness matrix for structure
c^E	Stiffness matrix for piezoelectric when leads are shorted
R_s	Engineering strain rotation matrix

There are several important things to notice about these variables. The elastic differential operator describes what type of problem we are dealing with, whether the structure is a beam, a plate or something else. The Ψ_r matrix is a vector where each entry is a different shape function. These shape function need only satisfy the geometric boundary conditions and are often referred to as admissible functions.

The difficulty in these calculations arise from the nature of the triple integrals. Calculating multiple integrals numerically is difficult for complicated functions. Doing these computations symbolically is somewhat feasible. The problem in doing these calculations symbolically is that the result is many terms long. We then must substitute numbers into all of the variables and then add all of the terms together. Adding the terms together can result in numerical instabilities. A special algorithm must be implemented to correct this. In addition we must have numerous cases for the repetition of arguments. The multiple integrals can be simplified to a product of one dimensions integrals. How this is done is explored in the next several sections.

If we orient a coordinate axis such that the x and y axis run along the width and length of the plate then the z axis will be along the thickness part of the plate. Since the plate is thin, isotropic and homogeneous and the three dimensional plate problem reduces to an equation in only x and y. The admissible functions are therefore a function of only x and y. The actual mode shapes will be a linear combination of weighted admissible functions. Although we could have used any set of functions that satisfy the geometric boundary conditions we choose to use a product of beam functions. Products of beam functions approximate the modes shape to a large degree. This is desirable because we need to use less functions to get accurate mode shapes.

For a simple supported plate the admissible function take the form:

$$W(x,y) = \sin(\frac{m\pi x}{a}) \sin(\frac{n\pi y}{b}) \tag{A.2}$$

where

a, b are the length in the x direction and the width in y direction respectively

m, n are positive integers. These are the mode indices.

For a simple supported plate, the products of the beam function are actually the modes shapes. The m and n modes indices are used to designate a certain mode. When one refers to the one-three

mode we are talking about $W(x,y)$ where m equals one and n equals three.

The beam functions for a clamped beam are much more complicated than those of a simple supported beam:

$$\left(\cos(\frac{m\,x}{a}) - \cosh(\frac{m\,x}{a})\right)\,(\sin(m) - \sinh(m)) + (-\cos(m) + \cosh(m))\,\left(\sin(\frac{m\,x}{a}) - \sinh(\frac{m\,x}{a})\right)$$

The admissible function are therefore also much more complicated than those of simple supported plate:

$$\begin{aligned} W(x,y) = \Bigg(\left(\cos(\frac{m\,x}{a}) - \cosh(\frac{m\,x}{a})\right)\,(\sin(m) - \sinh(m)) + (-\cos(m) + \cosh(m))\,\left(\sin(\frac{m\,x}{a}) - \sinh(\frac{m\,x}{a})\right)\Bigg) \\ \Bigg(\left(\cos(\frac{n\,y}{b}) - \cosh(\frac{n\,y}{b})\right)\,(\sin(n) - \sinh(n)) \\ + (-\cos(n) + \cosh(n))\,\left(\sin(\frac{n\,y}{b}) - \sinh(\frac{n\,y}{b})\right)\Bigg) \end{aligned} \quad \text{(A.3)}$$

Here, m and n are not positive integers. They are they eigenvalues of beam solution. These eigenvalues are 4.730, 7.853,10.996, 14.137 and so on.

Mass Matrix Overview

Let's take a look at the mass matrix in detail. We wish to reduce the required calculations to the simplest form symbolically. We can find a pattern in the required calculation and exploit this pattern to simplify the calculations. To do this, we use generic functions for the admissible functions into the calculation instead of using the actual admissible functions. Let's use a generic Ψ_r matrix with four generic admissible functions:

$$\Psi_r = \begin{bmatrix} f_1(x,y) & f_2(x,y) & f_3(x,y) & f_4(x,y) \end{bmatrix} \quad \text{(A.4)}$$

The mass matrix for a homogeneous plate will then be:

$$M_s = \int_{-tb}^{tb}\int_{0}^{b}\int_{0}^{a} M_{s\,temp}\,dx\,dy\,dz$$

$$M_{s\,temp} = \begin{pmatrix} \rho_s f_1(x,y)^2 & \rho_s f_1(x,y) f_2(x,y) & \rho_s f_1(x,y) f_3(x,y) & \rho_s f_1(x,y) f_4(x,y) \\ \rho_s f_1(x,y) f_2(x,y) & \rho_s f_2(x,y)^2 & \rho_s f_2(x,y) f_3(x,y) & \rho_s f_2(x,y) f_4(x,y) \\ \rho_s f_1(x,y) f_3(x,y) & \rho_s f_2(x,y) f_3(x,y) & \rho_s f_3(x,y)^2 & \rho_s f_3(x,y) f_4(x,y) \\ \rho_s f_1(x,y) f_4(x,y) & \rho_s f_2(x,y) f_4(x,y) & \rho_s f_3(x,y) f_4(x,y) & \rho_s f_4(x,y)^2 \end{pmatrix}$$

The mass matrix is symmetric . From this formulation we can see that the (i,j) entry of the mass matrix is computed by:

$$M_{s_{ij}} = \int_{-tb}^{tb} \int_0^b \int_0^a \rho_s f_i(x,y) f_j(x,y)\, dx\, dy\, dz$$

This triple integral can be simplified if we consider the nature of the integrand. Nothing in the integrand is a function of z. The integral therefore reduces to:

$$M_{s_{ij}} = 2\rho_s tb \int_0^b \int_0^a f_i(x,y) f_j(x,y)\, dx\, dy$$

Since the admissible functions f_i are products of beam functions we can further simplify the integral:

$$f_i(x,y) = f_{i,x}(x) f_{i,y}(y)$$
$$f_j(x,y) = f_{j,x}(x) f_{j,y}(y)$$
$$M_{s_{ij}} = 2\rho_s tb \left(\int_0^b f_{i,y}(y) f_{j,y}(y)\, dy \right) \left(\int_0^a f_{i,x}(x) f_{j,x}(x)\, dx \right)$$

The volume integral used to calculate the mass matrix has been reduced to a product of two one dimensional integrals. While the triple integral is very difficult to integrate the one dimensional integrals are not that difficult to compute.

Stiffness Matrix Overview

We will now use this same procedure to come up with simplified version of the (i,y) entry of the stiffness matrix K_s. Specifically, the generic Ψ_r described by described by Equation (A.4) is plugged into Equation (A.1) to yield:

$$K_{s_{ij}} = \int_{-tb}^{tb}\int_0^b\int_0^a \frac{1}{1-\nu_s^2}\Big(E_s z^2\Big(-2\left(\nu_s-1\right)f_i^{(1,1)}f_j^{(1,1)} + f_i^{(2,0)}\left(\nu_s f_j^{(0,2)} + f_j^{(2,0)}\right) + f_i^{(0,2)}\left(f_j^{(0,2)} + \nu_s f_j^{(2,0)}\right)\Big)\Big)\,dx\,dy\,dz$$

The $f_i^{(1,1)}$ notation refers to the partial derivative $\frac{\partial^2 f_i}{\partial x \partial y}$ and $f_i^{(0,2)}$ refers to $\frac{\partial^2 f_i}{\partial y^2}$. As with the mass matrix we make the substitutions $f_i(x,y) = f_{i,x}(x)f_{i,y}(y)$ and $f_j(x,y) = f_{j,x}(x)f_{j,y}(y)$ to give us a simplified form:

$$\begin{aligned}\frac{3K_{s_{ij}}\left(\nu_s^2-1\right)}{2tb^3} = &-2\left(\int_0^b f'_{i,y}(y)f'_{j,y}(y)\,dy\right)\left(\int_0^a f'_{i,x}(x)f'_{j,x}(x)\,dx\right)\\ &+2\nu_s\left(\int_0^b f'_{i,y}(y)f'_{j,y}(y)\,dy\right)\left(\int_0^a f'_{i,x}(x)f'_{j,x}(x)\,dx\right)\\ &-\left(\int_0^b f_{i,y}(y)f_{j,y}(y)\,dy\right)\left(\int_0^a f''_{i,x}(x)f''_{j,x}(x)\,dx\right)\\ &-\nu_s\left(\int_0^b f''_{i,y}(y)f_{j,y}(y)\,dy\right)\left(\int_0^a f_{i,x}(x)f''_{j,x}(x)\,dx\right)\\ &-\nu_s\left(\int_0^b f_{i,y}(y)f''_{j,y}(y)\,dy\right)\left(\int_0^a f''_{i,x}(x)f_{j,x}(x)\,dx\right)\\ &-\left(\int_0^b f''_{i,y}(y)f''_{j,y}(y)\,dy\right)\left(\int_0^a f_{i,x}(x)f_{j,x}(x)\,dx\right)\end{aligned}$$

The calculation for the stiffness matrix are much more complicated than those of the mass matrix.

We now have the generic forms for the mass and stiffness matrices. To make these calculations applicable to a certain physical system we only need to use the correct admissible functions.

Appendix B

Metarules Membership Functions

B.1 λ Adaptations

The input membership functions for the metarules of Section 6.1.1.4 are shown in Figures B.1 to B.3.

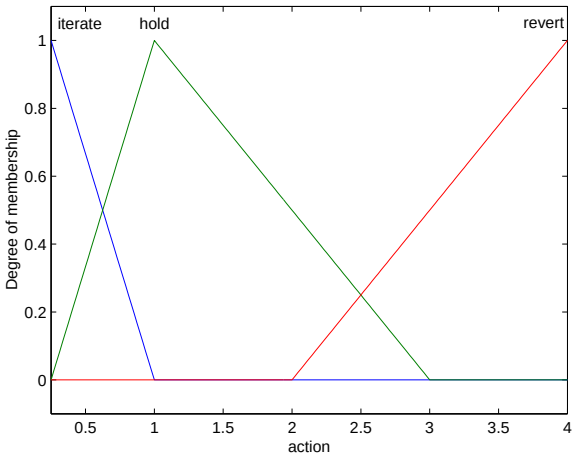


Figure B.1: λ Adaptations - Metarules Membership Function - Action Input

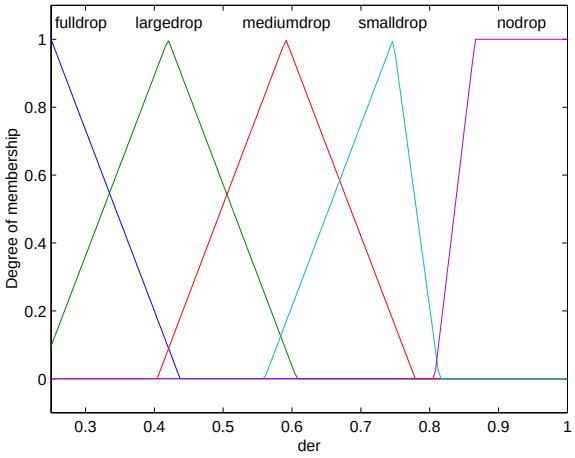


Figure B.2: λ Adaptations - Metarules Membership Function - Derivative Input

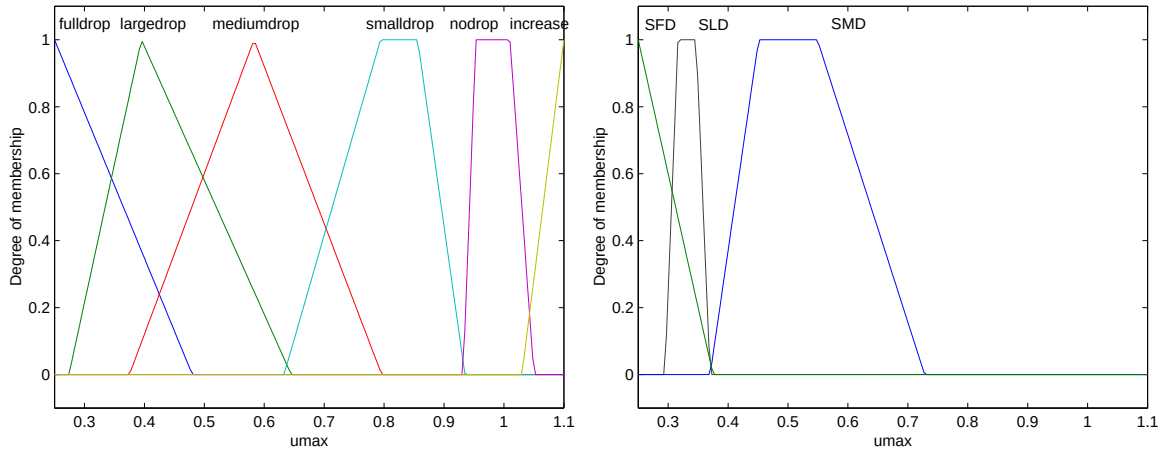


Figure B.3: λ Adaptations - Metarules Membership Function - Umax Input

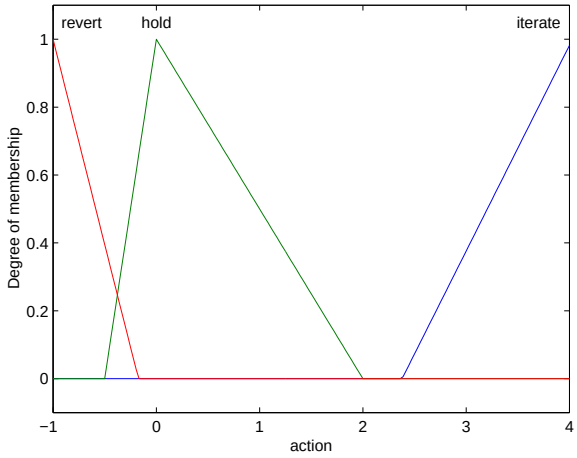


Figure B.4: Order Adaptations - Metarules Membership Function - Action Input

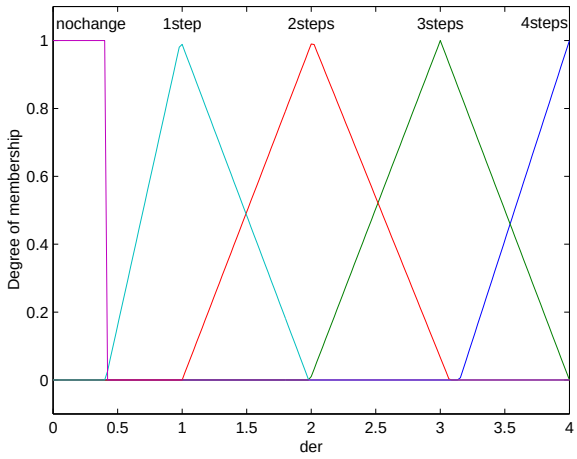


Figure B.5: Order Adaptations - Metarules Membership Function - Derivative Input

B.2 Order Adaptations

The input membership functions for the metarules of Section 6.2.1.4 are shown in Figures B.4 to B.6.

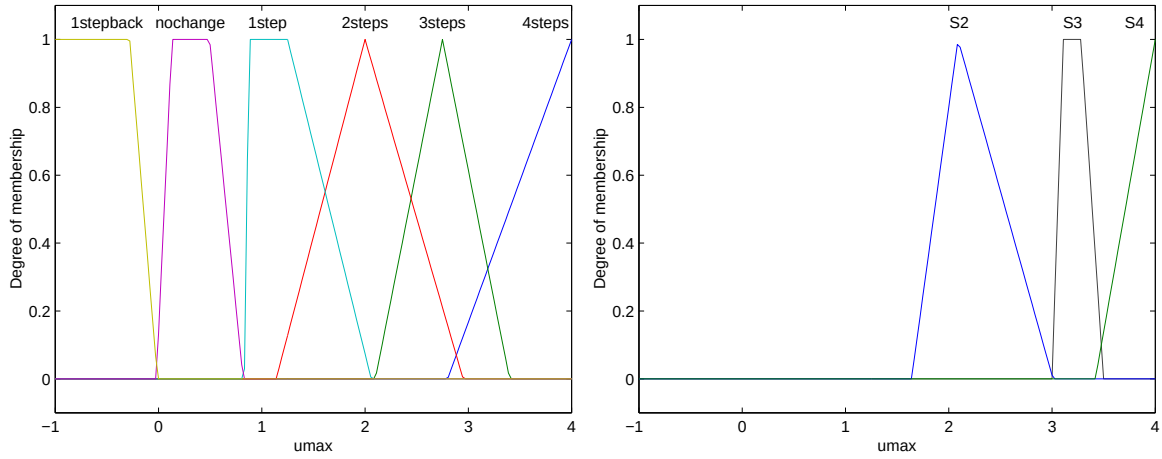


Figure B.6: Order Adaptations - Metarules Membership Function - Umax Input