Survival of the Fittest Controller

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Edward Whitworth Samson

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Masters Of Science in Electrical and Computer Engineering

Peter J. Seiler, Advisor

May, 2020

Acknowledgements

Raye Sosseh and Seagate inspired me to try this approach, Peter Seiler gave guidance on the methods and Bin Huang helped in handling the available data. My thanks to the University and state of Minnesota where I greatly enjoyed doing this research.

Abstract

This thesis presents computational approaches to controller tuning. It addresses the hypothesis that standard hard disk drive track following control designs do not take full advantage of actuator capability. In the first experiment a controller is synthesized with Matlab's *hinfstruct* for H_∞ robust optimal performance on a training set modeled on real drive actuator response data. Compared to standard robust synthesis the sensitivity bandwidth of the closed loop system increases, at the expense of robustness to hardware uncertainty. In the second experiment the scalar parameters of a classical control law are tuned with Matlab's *fmincon* to increase the simulated closed loop bandwidth of the drives in a training set. The results of both experiments support the hypothesis while providing avenues along which these modification can be developed and applied to other applications.

Contents

List of Figures

Chapter 1

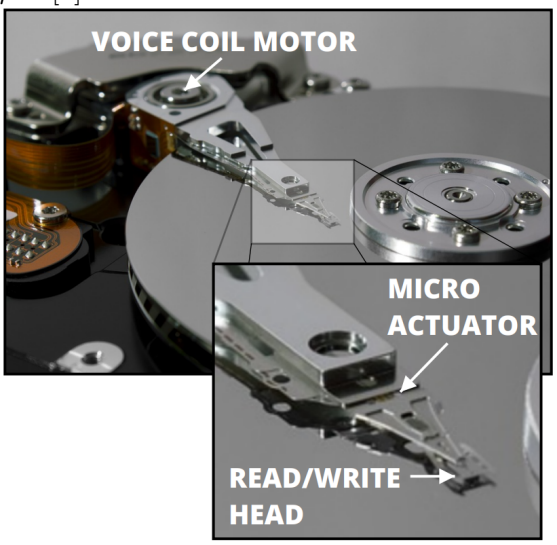
Introduction

1.1 Physical Paradigm

As we know, the world is now housing an incredible amount of data. It is less dizzying to think of all the files we store when one learns about hard disk drives (HDD's). A hard disk drive stores bits in the form of magnetic grains tens of nanometers in size. With a trillion bits on a drive, hundreds of thousands of drives in a larger data center, and half a million data centers worldwide, there is enormous storage capacity. HDD's contain metallic disks, and the bits lie in concentric circles (or 'tracks') on those disks. The disks rotate over 6000 revolutions per minute, so just by staying still the sensor reads hundreds of millions of bits per second.

During the read/write process, influences such as fans rattling the server rack cause the sensor and tracks to move relative to one another. If the misalignment is too large, bits will be misread or missed entirely. The disturbance rejection paradigm is the problem of keeping the "read/write head" over the bits as the disk rotates. There is a special section of each track that produces a voltage (the position error signal, or PES) in the reader proportional to the head's distance from the center of the track. The "head" is adjusted by the combined effect of two specialized motors called the voice coil motor (VCM) and micro actuator (MA). Figure 1.1 shows the inside of a drive with the two actuators labeled.

Figure 1.1: A hard disk drive has two actuators or motors, the VCM and the MA. The VCM has a large range of motion, allowing the head to sweep along the disk radius. The MA was added later to increase head movement speed for track following. It has a range of motion of 8 μm [1].



The engineering challenge is to design a controller that translates the PES into voltages in the VCM and MA so that the head stays centered over the bits as the whole assembly vibrates. This is called track following. Increasing the range of disturbances that the system can reject is the ultimate track following goal. This is complicated by the limitations of the actuators and standard design methods.

1.2 Motivation

Drives tend to be bought in multi-million-dollar orders to fill a data center. A better drive is so valuable that only three companies are advanced enough to compete, and these manufacture the vast majority of drives. Track following is an advanced control theoretic application, but the fact that one sensor is used to control two actuators means there is no unique "best" solution. All control strategies use heuristics, and their main justification is that they work [1].

Classical and robust optimal control designs are effective for rejecting low frequency disturbances, but may not use the full range of actuator capability. The high frequency

actuator dynamics are complex and uncertain, so design techniques that rely on human intuition and approximation may be sub-optimal. Adding automated tuning may be a way to design controllers that achieve better performance in track following. The aim of this research has been to improve the state of the art track following controller, but these methods might also be applicable to other projects where performance is at a premium and the standard design methods are thought to produce conservative results.

1.3 Overview

This is a practical as opposed to theoretical thesis. The designs are performed and evaluated as realistically as possible. The controllers abide by space limitations on a drive microprocessor. Both approaches take into account the need for robustness to hardware variability. All data used is taken from real production HDD's.

Chapter 2 explains the control system and describes how performance is evaluated. Chapters 3-4 address two data-driven design questions:

- Chapter 3: Can we make better use of the actuators by tuning an H_∞ control law to be robust *only* to drive variability that is present in a data set?
- Chapter 4: The engineer has completed a hand design, and believes some parameters can be changed to improve the performance. How do we tune these automatically?

The final chapter summarizes the conclusions from the above experiments and describes necessary or promising avenues that can be followed to improve these results and methods.

Note: In order to avoid disseminating proprietary information about the drives in this thesis, the frequency and magnitudes of all plots have been normalized. Likewise, the requirements, baseline, and improved performance presented are 'shifted' from real world values but consistent in that shifted framework.

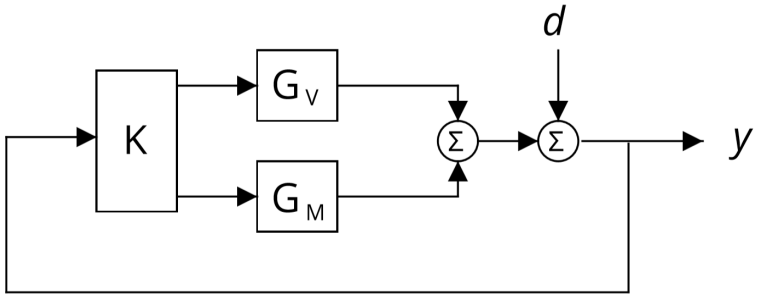
Chapter 2

Analysis

2.1 Control System

To quantify engineering objectives, the control system is described in detail. Figure 2.1 shows the closed loop block diagram for a dual stage HDD. The PES, y, is transformed by the controller $K(s)$ into two voltage signals. These control the VCM and MA, whose dynamics are represented by the functions $G_V(s)$ and $G_M(s)$, respectively. Actuators produce displacements which are added together with the disturbance displacement signal d. The control system drives the PES to zero since this is a disturbance rejection problem with no reference.

Figure 2.1: A system block diagram. Controller K, two actuators G, disturbance signal d, and error y.



Below, capital letters for signals indicate Laplace transforms, e.g. $Y(s) = \mathcal{L}\{y(t)\}$. All control design in this thesis is perfrormed in the frequency domain. Those unfamiliar

are referred to [2]. In the frequency domain, changes are measured according to fast or slow, large or small. There is no notion of the order events occur. Dynamical systems (functions of frequency) are characterized by how much they amplify or attenuate signals (their magnitude, or 'gain') as well as the lag between outputs and inputs (their 'phase').

The sensitivity function $S(s)$ is used to analyze how the closed loop system responds to disturbance signals. It is defined by Equation 2.1:

$$Y(s) = S(s)D(s)$$

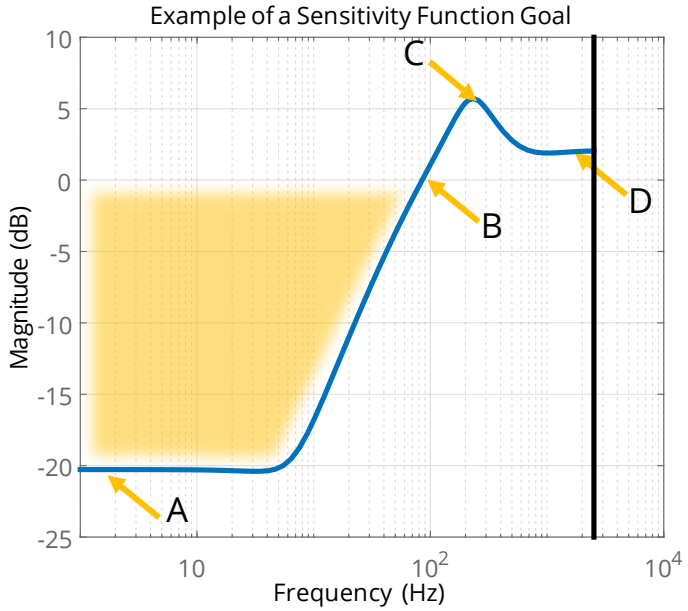
From Figure 2.1 it is clear that

$$S(s) = \frac{1}{1 + K(s)G(s)}, \tag{2.1}$$

where $K(s)$ is single input dual output (SIDO), i.e. $K(s) = [K_1(s)\ K_2(s)]$, and $G(s)$ is the vector of actuator open loop functions $[G_V(s)\ G_M(s)]'$. Simulations are used for evaluating performance and feasibility of each design. To simulate track following in a drive or set of drives, the sensitivity function $S(s)$ is computed from Equation 2.1.

The magnitude of the sensitivity function $|S(j\omega)|$ is an indication of which disturbances result in track alignment error. The PES magnitude will be low relative to disturbances at frequencies where $|S(j\omega)| \ll 0dB = 1$. Disturbances at frequencies where $|S(j\omega)| \approx 1$ will act directly on the drive without any help from the actuators, and $|S(j\omega)| > 1$ indicates the drive will actually amplify disturbances at some frequencies, a natural cost of pushing the sensitivity down elsewhere. Figure 2.2 is a schematic highlighting the notable features of the sensitivity function.

Figure 2.2: Various features of the simulated sensitivity magnitude help deduce expected track following error or PES: the magnitude at low frequencies **A** will be $\ll 1$ if low frequency disturbances produce low error. The bandwidth **B** indicates the frequency at which disturbances begin to cause larger PES; the peak sensitivity at **C** is the worst case amplification of disturbance signals. **D** is a region in which the actuators are assumed to be nonlinear and the controller should "do nothing." Designs are meant to make the low gain region (shaded yellow) as large as possible. Increasing the sensitivity bandwidth is complicated by the non-ideal actuator responses at higher frequencies.



The size of the shaded region in Figure 2.2 is limited by the Bode integral theorem, which states that systems with stable loop transfer functions and two more poles than zeros have equal area to the right and left of the 0 dB crossover frequency in the sensitivity magnitude plot. This results in the so-called 'waterbed effect,' meaning it is impossible to eliminate disturbance amplification without adding it somewhere else. As a result, engineers try to design controllers for good attenuation at lower frequencies, where the disturbances are assumed to have most of their power, and avoid large gains to avoid amplifying sporadic higher frequency disturbances.

2.2 Requirements

There are two requirements on the sensitivity magnitude $|S(j\omega)|$. A single controller is evaluated against them for every drive in the set, so they must be met even on the drive that performs worst. The sensitivity requirements are:

- Disturbance amplification does not exceed 5 dB at any frequency; $|S(j\omega)| < 5$ dB. And
- Good tracking of low frequency disturbances; $S(0) \leq -20$ dB.

The first ensures the maximum amplification of disturbances in all drives: no disturbance will be amplified by a factor of more than 1.8 = 5 dB. In Figure 2.2 this corresponds to a peak at **C** below 5 dB. The second requirement guarantees good tracking (reduction of the disturbance signal by a factor of at worst 0.1 = -20 dB) of slower vibrations, which are presumed to be the most common. This refers to the magnitude at **A** in Figure 2.2. In addition, high frequency disturbances should be allowed to act directly on the drive, i.e. the actuators should "do nothing." This refers to a low value for the magnitude at and beyond **D** in Figure 2.2.

The third requirement is on $K(s)$, the controller itself. This requirement must be present due to the hardware limitations, particularly on the micro actuator (the MA). It is only designed to move 8 μm in either direction, and if the input voltage signal is large, the actuator will saturate. This will damage the hardware, not to mention the models will be inaccurate and the control will not work in practice. Since Figure 2.1 shows that the input to $G_M(s)$ is $K_2(s)Y(s)$, the function $K_2(s)$ is always compared to a known reference to validate that it will not amplify the PES into higher voltages than the MA can handle.

2.3 Track Following Goal

The design goal by which the quality of a result is evaluated is increased sensitivity bandwidth. Higher sensitivity bandwidth means good track following for a wider range of possible disturbances. The larger the bandwidth, the wider the shaded region in Figure 2.2. The benchmark used in this study is a bandwidth of 125 Hz. This (scaled

number) is a high level of performance as achieved by standard design methods, but does not fully utilize the actuators, which are measured as having predictable albeit functionally complex linear response above 300 Hz. The following chapters describes techniques for outperforming that benchmark.

Chapter 3

Robust Synthesis and Tuning

Summary: A standard optimal control method is used to obtain an initial controller. Then the Matlab function *hinfstruct* tunes the controller to minimize the worst case closed loop H_∞ norm on the drives in a data set.

3.1 H_∞ Optimal Synthesis

Overview: Actuator dynamic data is collected and processed. The dynamics of the set of drives are captured by an uncertain model. The model, its uncertainty, and the design goals form a combined plant from which the optimal controller is determined using μ-synthesis. [3],[4], and [5] describe the process.

3.1.1 Data Processing

Open loop non-parametric models (frequency response data) of the drive actuators are obtained by applying a sweep of sinusoidal input voltages to each actuator and observing the gain and phase of the resulting displacement. The open loop responses of a set of ten VCM's and ten MA's are shown in Figure 3.1 (VCM) and 3.2 (MA). The variation in the responses at low frequency does not reflect actual variability in drive dynamics, it is an artifact of less precise testing at these frequencies during the sweep experiment. The actuator responses are nearly ideal (double integrator VCM and constant gain MA) below 100 Hz.

Figure 3.1: Open loop transfer functions of the two actuators. Each line represents an actuator. They vary slightly due to manufacturing imperfections and different operating environments, especially for high frequency input voltage signals, which can excite resonant modes i.e. cause the structure to become non-rigid. The low frequency variability is due only to imprecise data collection, the drives have identical responses below 20 Hz (VCM) and 200 Hz (MA). The VCM is approximately a double integrator, the MA is approximately a constant gain.

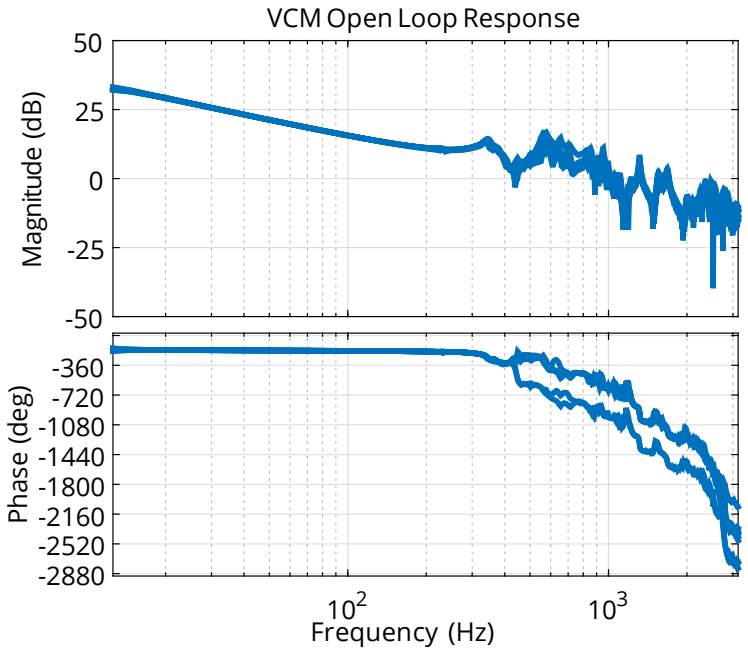
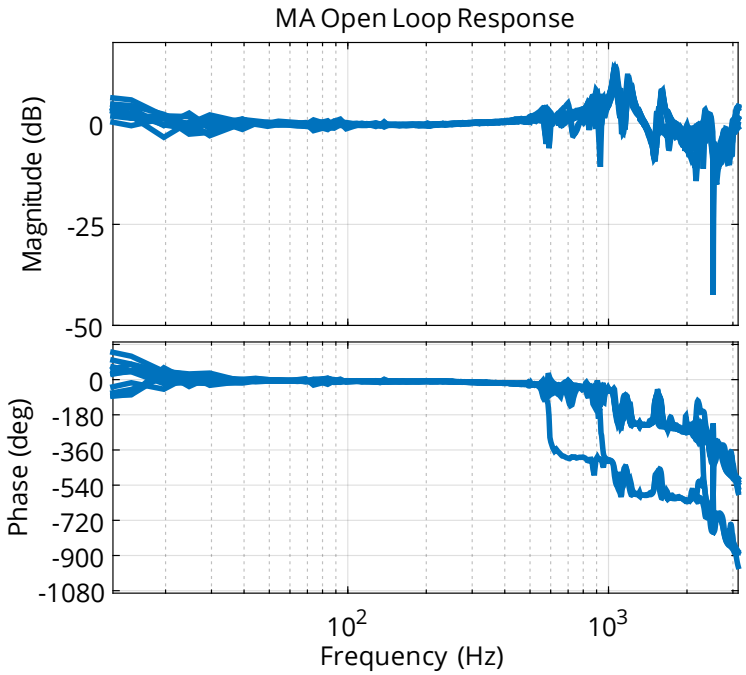


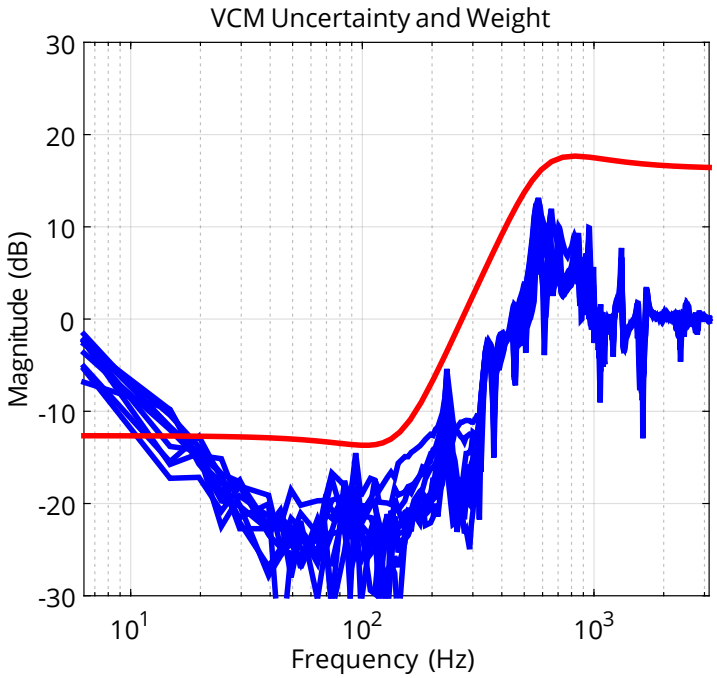
Figure 3.2: MA open loop transfer functions.



3.1.2 Modeling

For robust control design, the set of actuators is modeled as a nominal function and a frequency dependent weight that indicates the level of uncertainty. The uncertainty magnitude is chosen to be high where there is large variability in the set of all actuator responses. The nominal and uncertain models are designed based on heuristics, each having its own advantage [6]. The uncertainty used is often conservative, i.e. overstates drive variability, as demonstrated in Figure 3.3. The weights are hand tuned until the engineer decides the "best" performance has been achieved.

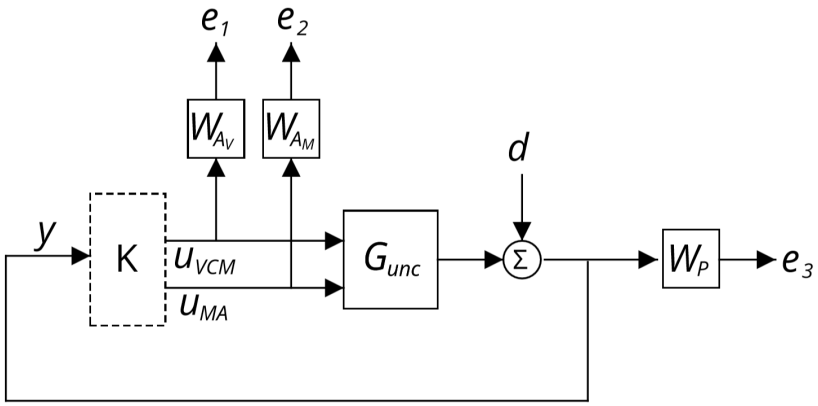
Figure 3.3: The magnitude of this weight is a measure of how far the drive dynamics are from the nominal model selected by the engineer. The red line indicates the uncertainty according to the model, which is a conservative estimate (e.g. 10 dB or over three times higher than the actual uncertainty at 300 Hz). Note that at the lowest frequencies the error is in reality below the model. As mentioned in Section 3.1.1 responses are less variable than they appear in these regions.



3.1.3 Standard robust synthesis

The full process by which variables are set up in Matlab for the μ-synthesis algorthm will not be detailed but can be found in [7],[5]. Included are the actuator models and signal weights as shown in Figure 3.4.

Figure 3.4: The modified block diagram for H_∞ control design. It includes the uncertain plant G_{unc} as well as design goals specified using weighting functions W_{A_V}, W_{A_M}, and W_P.

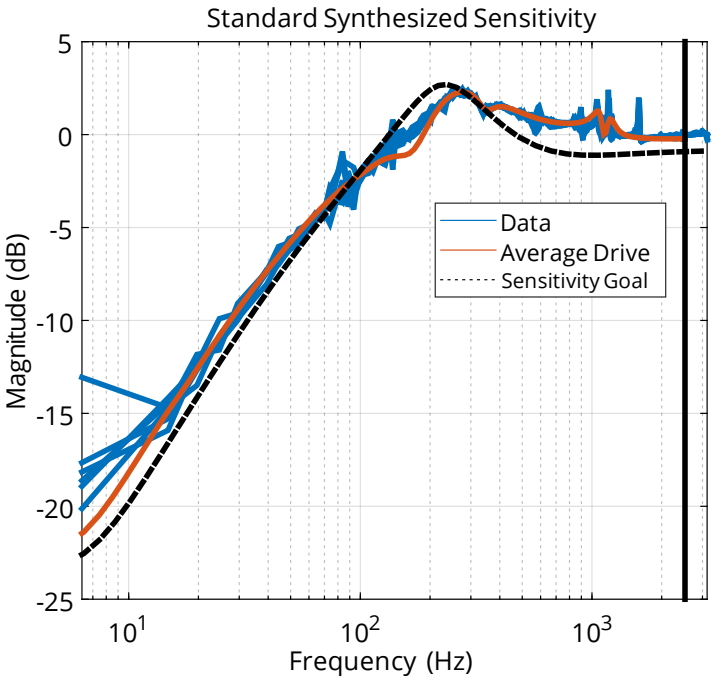


W_{A_V} and W_{A_M} normalize the actuator input voltages u_{VCM} and u_{MA} so that the outputs e_1 and e_2 are near or below 1 as long as the actuator voltages are below their rated levels. The sensitivity weight W_P normalizes the PES so that the sensitivity function lies below or around 1 when the requirements of Chapter 2 are met. This weight is also used to enforce the design goal. In practice, a constant is used for W_{A_V}, a third order filter for W_{A_M}, and a fourth order filter for W_P. These functions have significant impact on the result of the H_∞ synthesis because they directly determine what signal is prioritized at each frequency. The engineer typically chooses the weighting functions and tunes them by hand.

3.1.4 Baseline Performance

A μ-synthesis approach can produce a controller with the simulated performance of Figure 3.5. The low frequency sensitivity is -20 dB, the bandwidth is 125 kHz, and the peak is 2.5 dB. Clearly this meets the requirements of Chapter 2. The object of Section 3.2 is to test whether an H_∞ optimization performed only on a drive training set will be less conservative and better performing than a controller designed for a set of drives characterized by a parametric uncertainty model.

Figure 3.5: Closed loop sensitivity to disturbances in N=8 drives with a μ-synthesis controller. The low frequency sensitivity is -20 dB, the bandwidth is 125 Hz, and the sensitivity peak is 2.5 dB. Also shown is $1/W_P$ where W_P is the sensitivity weighting function. *Musyn* attempts to drive $|S(j\omega)|$ below that goal. As can be seen the engineer has reduced W_p to put additional priority on the sensitivity during design. The line marked "Average Drive" is the performance of the controller on the nominal models created for the VCM and MA.



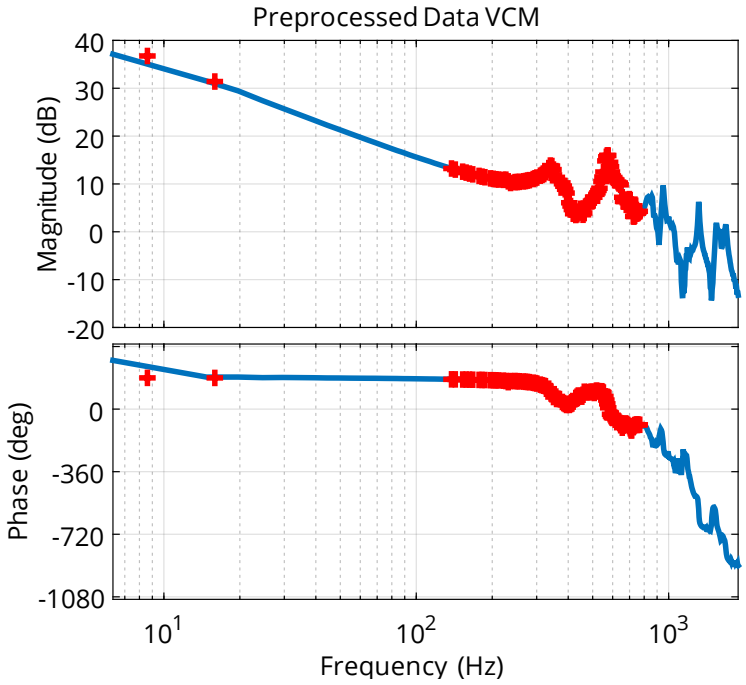
3.2 Data-driven H_∞ Synthesis

Overview: The set of drives from Figure 3.1 is combined into a large array of dynamical system models. The *hinfstruct* algorithm described in [8], initialized with the optimal controller from Section 3.1, tunes the controller only to minimize the worst H_∞ norm on the drive models in the array.

3.2.1 Modeling

Fitting the actuator response data with models is necessary for the Matlab dynamical system tuning algorithm *hinfstruct*. Modeling attempts to capture the real hardware dynamics so the controller will perform well in experiments, but choosing the best models for a generalizeable controller requires trial and error. A subset of data points are selected and fictitious data is added at low frequencies to correct low frequency data collection errors mentioned in Section 3.1. In the case of the VCM the ideal response is that of a double integrator; in the case of the MA, a constant gain with a delay. The resulting set of points is shown in Figure 3.6.

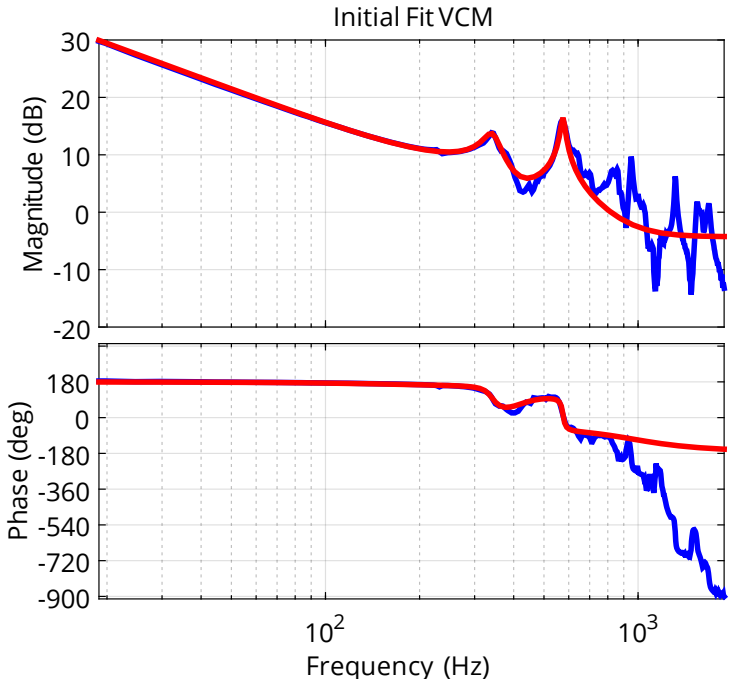
Figure 3.6: Pre-processed data for *hinfstruct* fits. The leftmost data is fictitious, but correct. Some data has been removed to ease fitting. Only the VCM is shown, but MA pre-processing is similar.



Next, high fidelity models are fit to this adjusted data set using the Matlab function *fitfrd*. The model characteristics including order are design choices that impact the performance and generalizability of the resulting controller. A sample fit is shown in

Figure 3.7. A second round of fitting is performed with the Matlab function *tfest*, which creates stable functions to approximate each actuator response. The fitting of models to data presents subtle difficulties discussed at the end of the chapter in Section 3.2.4.

Figure 3.7: Initial high fidelity model of a VCM fit using *fitfrd*. This is done for each VCM, then these accurate fits are re-sampled and re-fit with stable models.



3.2.2 Structured H_∞ Synthesis

The stable models serve as the "training" data. In principle a training data set could contain as many individual drives as desired. Here we consider six different pairs of actuators. *Hinfstruct* searches for the parameters of a controller K^* (a Matlab *tunableSS* system object with 51 states and 306 tunable parameters) that minimizes the worst (highest) H_∞ norm in the data set. Namely

$$K^* = \arg\min_K \sup_{i=1:N} ||F_l(P_i, K)||_\infty$$

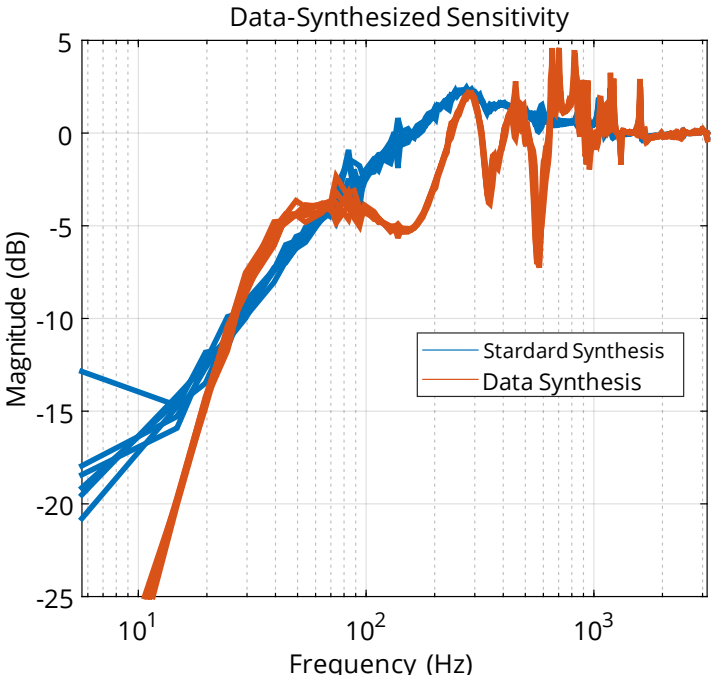
where $F_l(\cdot)$ is the lower fractional transformation, P_i is the i^{th} combined plant, which includes all models and weights, and N is the number of training models (N=6). The

controller designed in Section 3.1 with μ-synthesis is used to initialize the parameters of K. This is necessary because the *hinfstruct* optimization does not converge to a global or local minimum. In some sense *hinfstruct* is tuning the μ-synthesis controller for performance only on the dataset.

3.2.3 Results

The method above produces different results depending on a number of factors, in particular the actuators used in synthesis. The best performance obtained on a set of six drives is indicated by Figure 3.8. The bandwidth has been increased to 240 Hz, or nearly doubled, indicating a higher range of disturbance rejection. However, disturbance frequencies above 6.3 kHz are amplified by a factor of 5 dB and may cause large PES. Also, $|S(j\omega)|$ has a wide range of values for frequencies above 200 Hz. This means a sinusoidal disturbance of frequency e.g. 550 Hz would be attenuated by half, but one of frequency 650 Hz would be amplified by a factor of around 3.

Figure 3.8: Evaluation of the *hinfstruct* controller. The worst sensitivity bandwidth in the dataset is 240 Hz compared to the baseline performance of 125 Hz, while the low frequency attenuation is well below the baseline -20 dB. However, the peak disturbance amplification is close to 5 dB for disturbance frequencies between 600 Hz and 900 Hz and varies wildly in the high frequency region.



The controller gain $K_2(s)$ is shown in Figure 3.9 compared to the baseline function in order to justify that tuning does not cause the MA to saturate. Since the maximum magnitude decreases or does not change the tuned controller does not introduce any extra amplification of the MA voltage signal.

Figure 3.9: Controller gain $K_2(s)$ in *hinfstruct* and μ synthesis controllers. The second stage actuator input voltage signal is $U_2(s) = K_2(s)Y(s)$. Because the controller labeled "Standard Synthesis" does not saturate the actuator the *hinfstruct* controller, labeled 'Data Synthesis,' will not either.

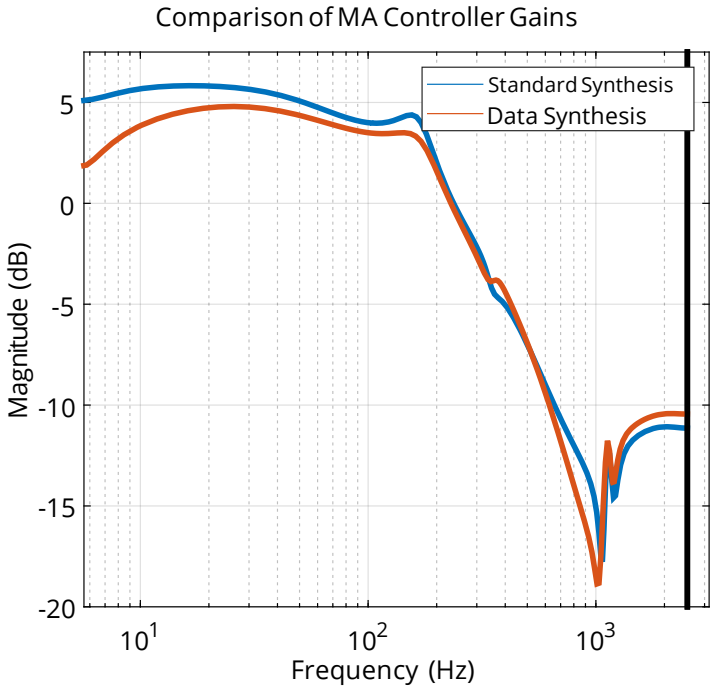
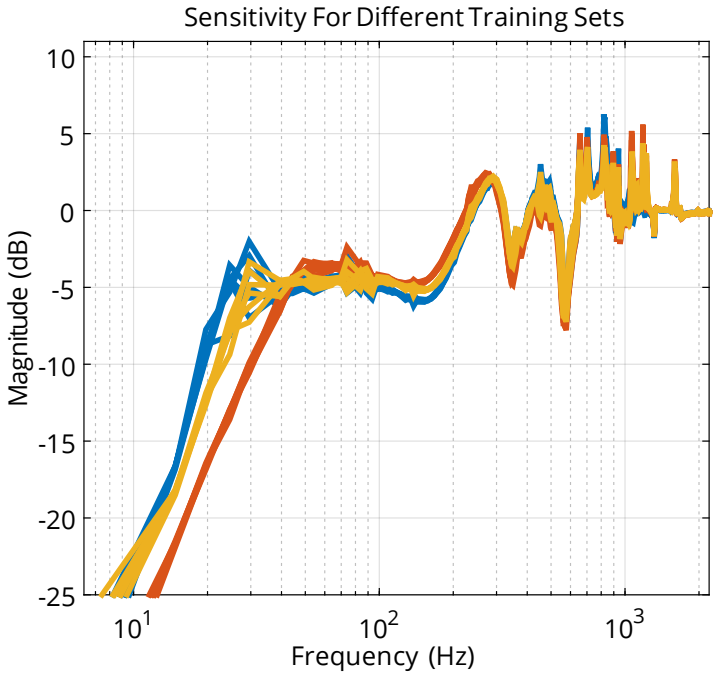


Figure 3.10 shows how *hinfstruct* is impacted by the training set of models. Each color represents a controller tuned to a different set of five pairs of actuator models and simulated on all six drives. *Hinfstruct* will return a different controller for each training set. Selecting the best set is an open problem that currently involves trial and error.

Figure 3.10: Results when *hinfstruct* is performed on different training data. Each color is a simulation on the same set of six drives using a controller designed with a different set of five actuator models. The dark controller may not be appropriate because of the gain spike at 20 Hz.



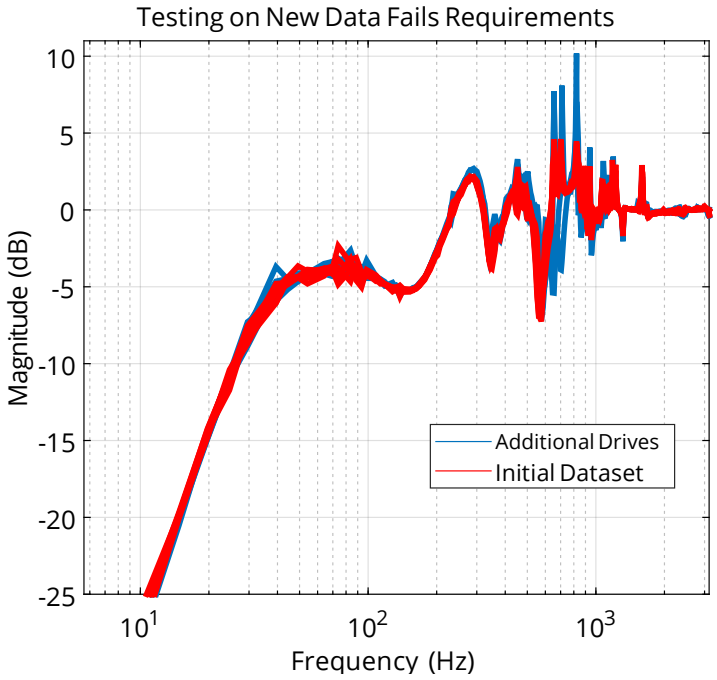
3.2.4 Discussion

Both the μ-synthesis and *hinfstruct* controllers in this chapter are reduced from order 51 to order 24 (the drive microprocessor only admits 24 states for track following control). Balanced truncation is used to achieve the necessary reduction in states, with no performance degradation. The 24^{th} order controller is the one whose sensitivity magnitude is shown in Figure 3.8. There was no change in the simulated sensitivities after model reduction to 24 states.

To analyze the generalizability of the *hinfstruct* controller several additional drives were simulated. Figure 3.11 shows that when the synthesized controller is used in three additional drives, their sensitivity gain exceeds the requirements ($|S(j\omega)| > 5$ dB between 700 and 900 Hz). This result indicates that more robustness is needed. This

might be achieved by better modeling of the actuators, additional models being included in the training set, and or a proper selection of models for training.

Figure 3.11: Robustness issues with the *hinfstruct* controller are demonstrated by testing it on three additional drives. The sensitivity indicates these drives will have unmanageably large error for disturbances at higher frequencies.



The algorithm runs within one minute on a standard desktop computer. However, if needed *hinfstruct* can be easily parallelized in the options argument. The computational resources of the Minnesota Supercomputing Institute (MSI) were often used during development of the tuning program. Each core can run the algorithm independently from a different random initial condition. Synthesizing the controllers presented here did not require the additional processing capability of MSI.

Hinfstruct automatically minimizes the H_∞ norm of the combined plant. However, similar tuning algorithms exist that will train a dynamical system object (such as a controller) based on alternative costs. These were not explored here since H_∞ was the most appropriate design cost, but the Matlab function *systune* could be used to tune a

controller if another type of cost (such as one based on the L_2 norm) is being used in design [8].

The fitting process can be a bottleneck to tuning with *hinfstruct*. Two models that both fit the data can have different tail behavior, as seen in Figure 3.12, and that impacts the H_∞ cost. To obtain a desired fit it may be necessary to repeat the fitting process many times, selecting different orders and dropping difficult data points. This was done by creating a graphical user interface in Matlab. Figure 3.13 is a snapshot of that interface.

Figure 3.12: Two fits accurately model the data between 20 Hz and 1 kHz but the 'tail' behavior of the clean fit is preferable. The controller is tuned to minimize the H_∞ norm which means all frequencies are considered. The troublesome fit is simply the discrete time version of the clean fit.

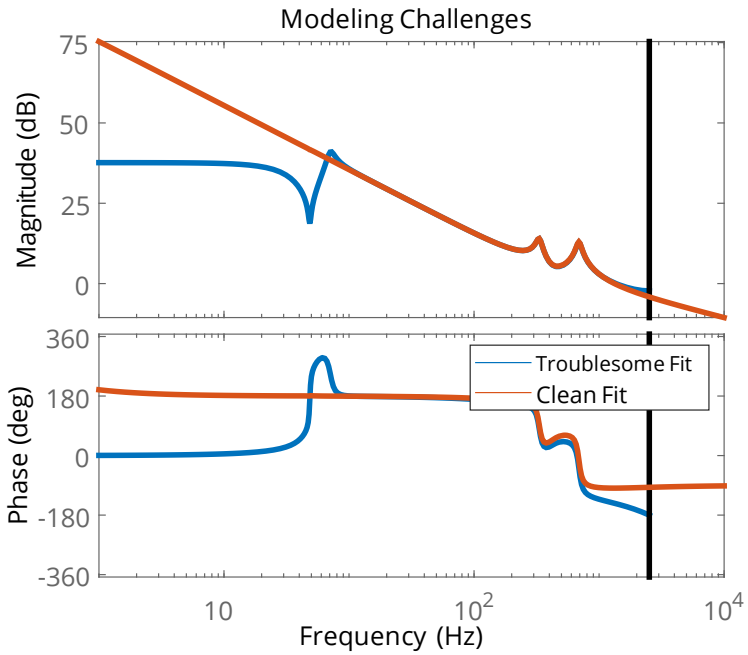
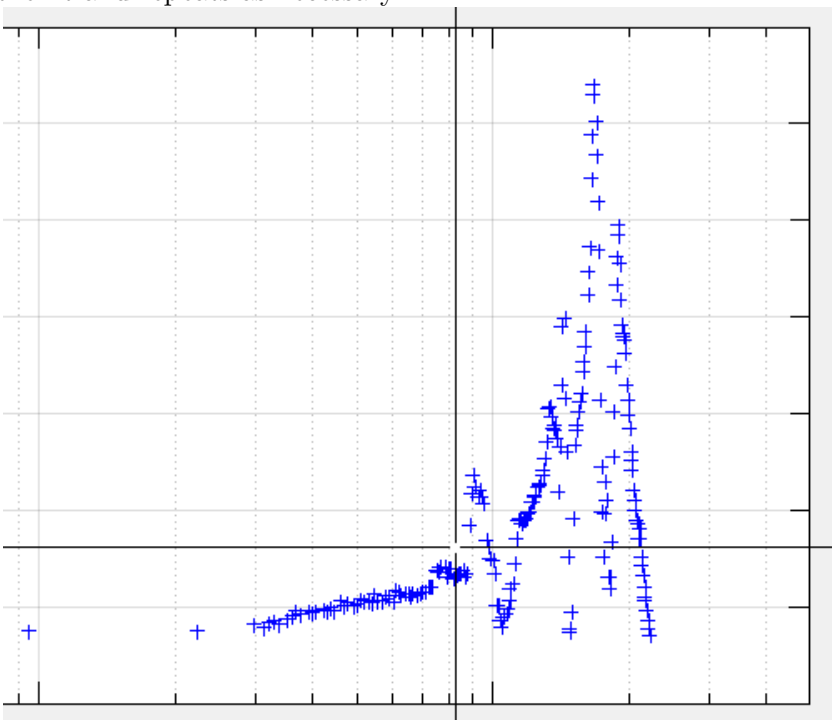


Figure 3.13: A simple graphical user interface was developed to facilitate fitting models for *hinfstruct* tuning. The crosses represent individual data points. The cross-hairs indicate mouse position. The user selects data points to drop them before fitting, then evaluates the fit and repeats as necessary.



In this experiment two drives were dropped from the training set because their fits caused issues with the algorithm: it did not converge to a feasible controller when those fits were in the training set. The resulting controller still performed robustly on six drives. As this section demonstrates, additional work on this method is needed to develop controllers that are robust to the full range of actuator dynamics encountered in production HDD's.

Chapter 4

Parametric Tuning Experiment

Summary: Classical design using loop shaping is effective, but leaves optimism that parameters can be tuned further to produce substantial or minor performance improvements. The parameters are used as an argument to a *fmincon* optimization with sensitivity bandwidth as cost and tuned automatically to their 'best' values.

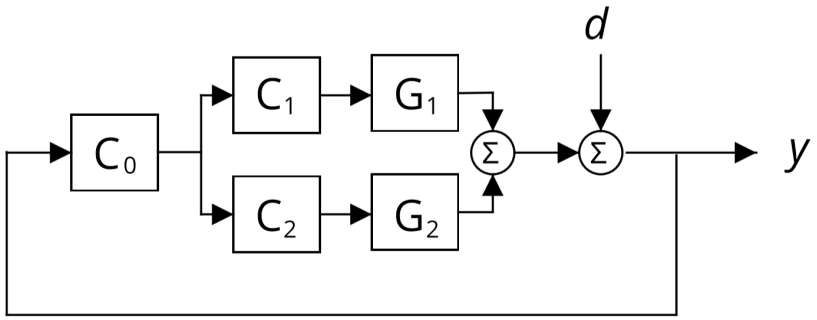
4.1 Initial Classical Approach

Overview: The PQ method reduces SIDO control design into two loop shaping steps and results in a low order controller with robust performance.

4.1.1 Introduction

SIDO systems present unique challenges because of the difficulty of determining the best relative contribution of the two actuators. The VCM is best whenever the disturbance has all its power at low frequencies. The MA has higher bandwidth, but saturates when it is displaced by 8 μm. Heuristic methods are used that generally try to give more responsibility to the VCM at low frequencies and the MA at high frequencies where disturbances are presumed to have less power. One classical design method for SIDO systems is called PQ [9]. It decomposes the SIDO problem into two single input single output (SISO) ones. The block diagram for the first step in PQ is shown in Figure 4.1.

Figure 4.1: The PQ block diagram. G_1 can be thought of as the VCM, and G_2 the MA. Their output displacements add to produce an overall head displacement. The overall loop gain is $L = C_0C_1G_1 + C_0C_2G_2$.



The two actuators have the symbols G_1 and G_2. It can be seen in Figure 4.1 that the combined output displacement is the sum of the two actuators' contributions. The VCM is approximated by a double integrator, and the MA is approximated by a constant gain with a delay. Tuning the free parameters during loop shaping is done under these assumptions, but the controller is evaluated in simulation on a set of N=8 real drives.

4.1.2 Loop Shaping

As mentioned in Section 4.1.1, the relative contribution from the two actuators must be specified in some way for SIDO design to proceed. In PQ, their ratio is designed to have desired frequency domain characteristics. This is done in the following way: Figure 4.1 shows the loop gain is $L = C_0C_1G_1 + C_0C_2G_2$. The ratio of the actuators' two output gains is

$$R = \frac{C_0C_1G_1}{C_0C_2G_2} = \frac{C_1G_1}{C_2G_2} \tag{4.1}$$

Letting G_1 be the VCM and G_2 the micro actuator, $Q = C_1/C_2$ is designed to make R higher (larger VCM contribution) at low frequencies, and decrease as much as possible (eliminate the use of the VCM) at high frequencies. To obtain the desired ratio, the function Q is designed as a combination of controller components. The desired magnitude is achieved by a low frequency boost f_b and high frequency roll off f_{ro} specified

by

$$f_b(s) = \frac{s + x_1}{s + x_1/x_2}$$

$$f_{ro}(s) = \frac{x_3}{s + x_3},$$

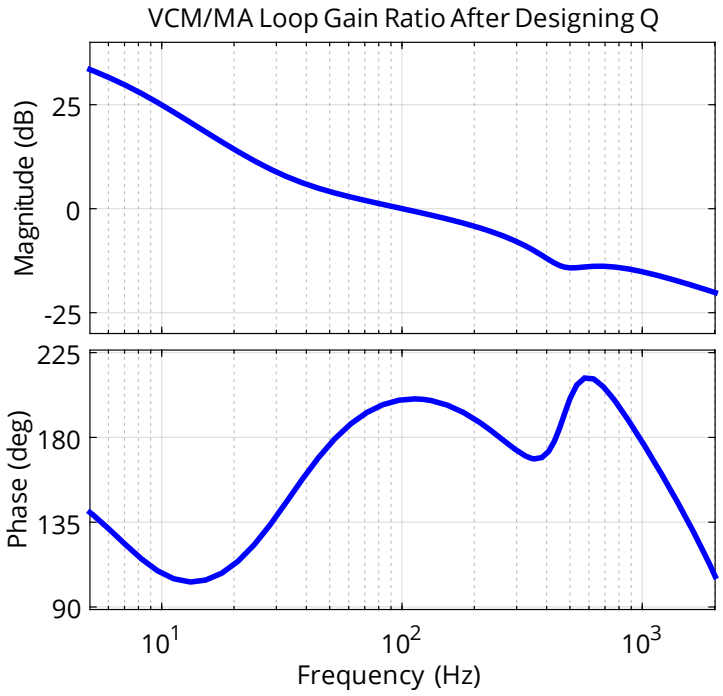
where x_i are free parameters tuned during design.

The phase of R determines the amount of destructive interference between the two actuators. 180 degrees of phase corresponds to the actuators moving in opposite directions. Therefore a large phase margin is preferable, especially where R is close to 1 (when $R = 1$ the actuators contribute equally to the overall loop gain). The phase margin is increased using a lead filter f_{l_1} of the form

$$f_{l_1}(s) = \frac{x_4 s + x_5}{s + x_4 x_5}$$

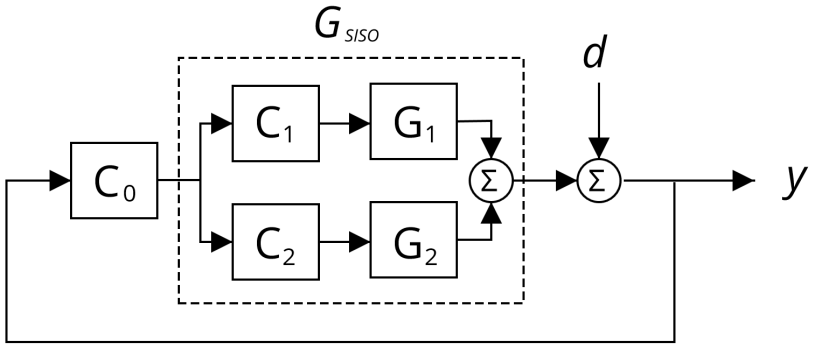
and the 0 dB crossover is set by a constant gain k_1. The final function $Q = C_1/C_2 = f_b f_{ro} k_1 f_{l_1}$. The ratio R is shown in Figure 4.2.

Figure 4.2: Bode plot of the actuator contribution ratio R. Q is designed so that the VCM contribution is higher below the 0 dB crossover frequency of 1 kHz (i.e. $R > 1$). The phase margin around 100 Hz is made as large as possible (20 degrees of phase margin) to avoid the two actuators moving in opposite directions, and thereby having to work much harder to achieve the same displacement. As mentioned design is performed to achieve the ratio below on ideal actuator models (double integrator VCM and constant gain MA).



Now the system can be redrawn as single input single output (SISO) as it is in Figure 4.3. The combined plant $G_{SISO} := C_1G_1 + C_2G_2$ is completely determined, although C_1 and C_2 are not unique. C_2 is usually chosen to be 1, so that $C_2 = Q$.

Figure 4.3: In this modified block diagram the SIDO problem has become a SISO problem where the plant is G_{SISO}. All that remains is to design C_0 for the desired overall loop shape.



The final step is to choose controller components in C_0 to achieve the desired loop transfer function $L = G_0C_0$. This is done by a combination of low frequency boost f_{lf} of the form

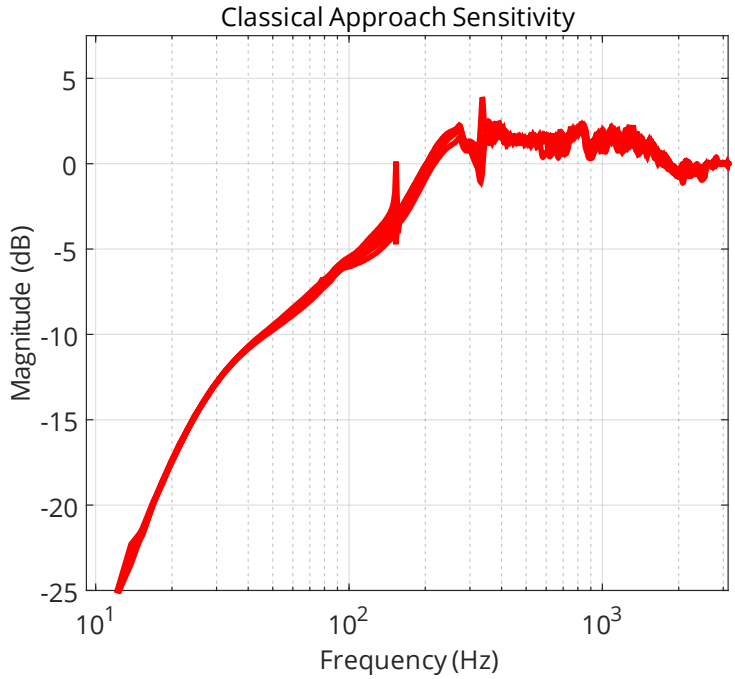
$$f_{lf}(s) = \frac{s + x_6}{s + x_6x_7},$$

another lead filter f_{l_2} with three free parameters x_8, x_9 and x_{10}, and a constant gain x_{11}. The final function $C_0 = f_{lf}f_{l_2}x_{11}$. Now the controller $C(s) = [C_0C_1 \ C_0C_2] = [f_{lf}k_2f_bf_{ro}k_1f_{l_1} \ f_{lf}f_{l_2}x_{11}]$ is determined.

4.1.3 Baseline Performance

The hand-tuned controller achieves the performance shown in Figure 4.4 on a set of N=8 drives. The bandwidth is 200 Hz and the peak sensitivity is 4 dB. (Note: this is an entirely different data set than that used in Chapter 3. In fact it is not even the same model of drive. Therefore this tuning example should be thought of as a separate experiment.)

Figure 4.4: The results of a classical approach to designing a SIDO controller, with the parameters tuned by hand. The bandwidth is 200 Hz, the peak sensitivity is 4 dB and the low frequency sensitivity is much less than -20 dB. Each line is a separate drive (N=8).



The performance is outstanding considering the controller is only of order 10, and that the design was done on simple actuator models. Even so, more improvement might be possible because $C(s)$ has eleven scalar parameters $\overline{x} = [x_1, x_2, \ldots x_{11}]$ that were tuned by hand. The large multi-dimensional parameter space is difficult to relate to overall performance, and may have been impossible to completely search by hand tuning. Section 4.2 explores one approach to automated tuning.

4.2 Parameter Tuning

Overview: Parameter tuning consists of re-computing the PQ controller as the scalar parameters are varied. The Matlab function *fmincon* checks the sensitivity and searches for parameters to minimize its bandwidth subject to applicable constraints.

4.2.1 Cost and Constraint Functions

For each set of parameter values $\overline{x}$, the controller $C(s)$ is recomputed. The closed loop sensitivity is simulated on the full data set according to the methods of Chapter 2. For a cost function $J(\overline{x})$, $\overline{x}^*$ is chosen so that

$$\overline{x}^* = argmin_{\overline{x}} J(\bar{x}).$$

In the hard disk drive paradigm, the goal is to maximize the sensitivity bandwidth ω_0, the frequency of 'B' in Figure 2.2. The corresponding cost function is

$$J(\bar{x}) = -\omega_0. \tag{4.2}$$

The bandwidth is computed by finding the first frequency at which any of the simulated sensitivities exceeds 0 dB.

If left unconstrained, *fmincon* may find parameters that produce high bandwidths, but the closed loop system may be unrealistic or fail under the criteria of Chapter 2. Therefore we impose constraints such as a bound $B(s)$ on the sensitivity function:

$$|S_{\bar{x}}(s)| < B(s). \tag{4.3}$$

This transfer function can enforce sensitivity maximum constraints at all frequencies, namely a 5 dB peak and -20 dB low frequency gain. A sensitivity bound is shown in Figure 4.5.

Another constraint was used to check the Bode integral in Equation 4.4.

$$\int_{-\infty}^{\infty} \log |S_{\bar{x}}(j\omega)| d\omega \tag{4.4}$$

In practice neither the classical method nor the tuned result have Bode integral exactly zero. It is forced to be below 5 for tuning, which is the maximum simulated Bode integral with the classically designed controller.

4.2.2 Automated Tuning with *fmincon*

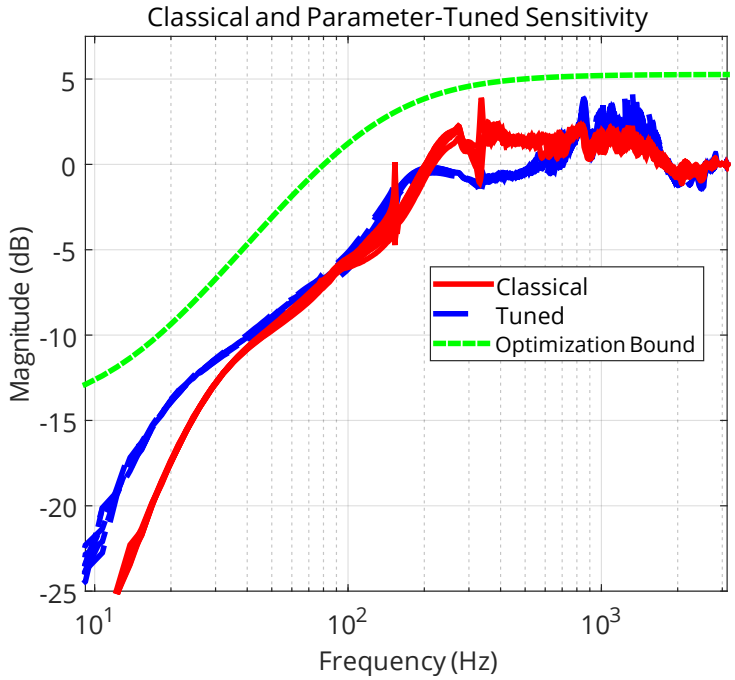
Tuning is performed with *fmincon* in Matlab. The initial values, cost, and constraint functions are its arguments. A range of values can also be specified if the tuned controller

is thought to lie in some specific parameter space. The initial values in $\overline{x}$ were chosen from their final values in the classical design of Section 4.1.2. *Fmincon* attempts to find parameters that maximize the sensivitity bandwidth as long as the Equation 4.4 and 4.3 constraints are satisfied.

4.2.3 Results

Tuning gave a controller with performance shown in Figure 4.5. Also shown is the optimization bound $B(s)$ referenced in Equation 4.3. The technique was able to increase the bandwidth from 200 Hz to 500 Hz, while keeping the peak sensitivity below 5 dB and low frequency gain below -20 dB as required.

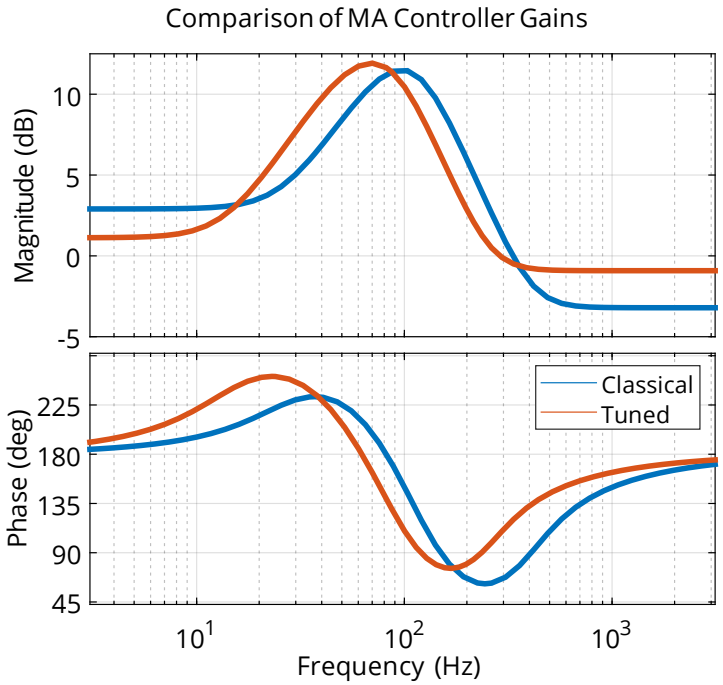
Figure 4.5: Performance before and after tuning the parameters of a classically designed controller. The optimization increased the bandwidth from 200 Hz to 500 Hz while keeping the peak sensitivity the same and the low frequency gain below the required -20 dB. Performance of the two controllers on the same set of N=8 drives. Also shown is the bound $B(s)$ referred to in Equation 4.3.



Again, the controller gain to the MA is also compared to the pre-tuned version in

Figure 4.6. The tuning did not substantially change $K_2(s)$, so there is no reason to believe the tuned controller will saturate the MA.

Figure 4.6: Controller gain $K_2(s)$ for the classical and tuned designs validate the assumption that tuning did not produce an unfeasible controller that would saturate the MA. The peak gain of the tuned controller is 8% higher.



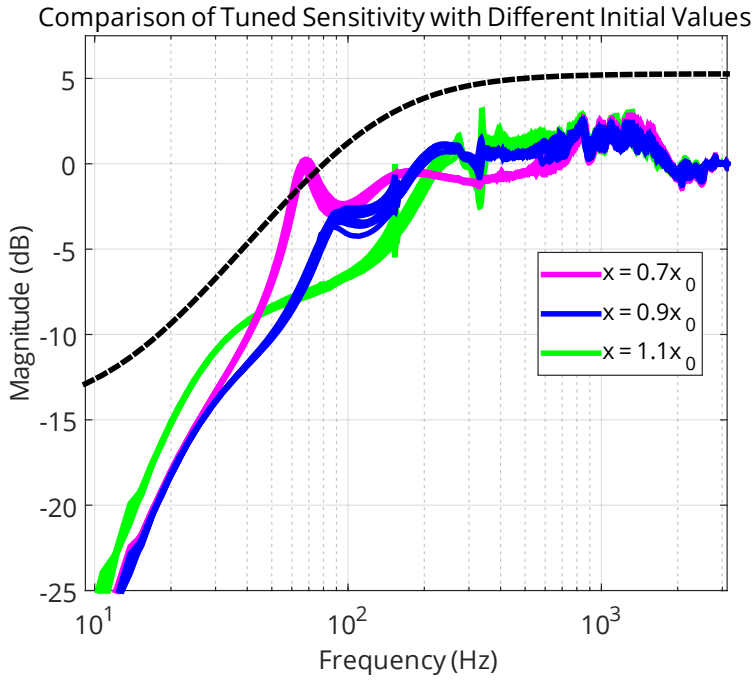
4.2.4 Discussion

The simulations in Figure 4.5 show that although the bandwidth increased considerably, disturbance attenuation is minimal from 200 Hz to 500 Hz. In this region $|S(j\omega)| \approx 1$, which means the controller "does nothing," so disturbances act directly on the drive and may still cause misread errors if they are large enough. It is clear the cost may need to be specified more precisely. This could be done by making $B(s)$ lower around 200 Hz. In any case, the results support the hypothesis that *fmincon* can be used to improve performance by automating parameter tuning after a classical design.

During this experiment it was found that the initial parameters from the classical design, $\overline{x}_0$ were not necessarily the best starting point for *fmincon*. The results in Figure

4.5 came from initial parameters values of $0.75 * \overline{x}_0$. Figure 4.7 shows several other starting parameter values, which indicate that one should try various initial conditions nearby $\overline{x}_0$.

Figure 4.7: Tuning parameters from various initial values. x_0 is the value of $\overline{x}$ when classical tuning is complete. The best performance *fmincon* can find depends on how parameters are initialized. Each color represents a separate controller tested on the same set of 8 drives.



The algorithm runs in under three minutes on a standard desktop computer. Depending on the size of the data set and the number and complexity of costs and constraints, this might be longer.

Chapter 5

Conclusion and Future Work

Two questions were motivated at the beginning of this thesis *1)* whether it is possible to synthesize a robust controller form a set of hardware data, and *2)* how the parameters of a classically designed controller can be automatically tuned to increase performance. In addressing question *1)* it was shown in Chapter 3 that for H_∞ design the Matlab function *hinfstruct* can be used to synthesize a controller that performs better on a training set compared to standard robust synthesis. As for question *2)*, Chapter 4 demonstrates that *fmincon* is an appropriate tool to automate tuning, and can improve performance while meeting the design constraints.

Actual testing in a HDD was not performed for this thesis, so there may be further complications that are not noticed in simulation. These may arise from nonlinearities in the hardware or other features not captured by the frequency response data. The data sets used for evaluation were also small (N=6 and N=8). In reality a controller must be placed on potentially millions of drives in vastly different environments (16^oC to 64^oC) and still achieve the minimum required disturbance rejection. Repeating the testing of the tuned classical controller on a larger data set is easy, but resolving the robustness issues mentioned in Section 3.2.4 is complicated. In some cases it is difficult to include additional data with *hinfstruct*, not so much because of the computational difficulty, but because fitting accurate and consistent stable models to each drive currently requires much human oversight. This is in some sense trading the work of hand tuning for the work of fitting, but is justified by the better final performance. The difficulty involved in fitting is a hurdle that can be resolved with additional study of the fitting process

and development of fitting tools, however some engineers might choose to use standard techniques that come closer to guaranteeing robustness.

As mentioned in Chapter 1, this thesis may provide introduction from which readers can use these method for other control design projects. The likely applications are those where performance is in high demand. Simpler and smaller data sets may be more amenable to these techniques at first, with more complex requirements and larger data sets coming later as the techniques are honed and improved. For now there are likely to be some applications where a small percentage improvement is desired at the end of design, and the emphasis is not on repeatable or predictable changes, which these methods do not provide. In some systems like the SIDO dual stage HDD that rely on heuristics for control this may be particularly useful.

References

[1] Abdullah Al Mamun, GuoXiao Guo, and Chao Bi. *Hard disk drive: mechatronics and control*. CRC press, 2017.

[2] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

[3] K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice-Hall, 1996.

[4] Michael Green and David JN Limebeer. *Linear robust control*. Courier Corporation, 2012.

[5] Masanori Honda. *Temperature Dependent Robust Control of Hard Disk Drives Using Parameter Varying Techniques*. PhD thesis, University of Minnesota, 2016.

[6] Gary J Balas, Andrew K Packard, and Peter J Seiler. Uncertain model set calculation from frequency domain data. In *Model-Based Control:*, pages 89–105. Springer, 2009.

[7] Gary J Balas, John C Doyle, Keith Glover, Andy Packard, and Roy Smith. μ-analysis and synthesis toolbox. *MUSYN Inc. and The MathWorks, Natick MA*, 1993.

[8] Pascal Gahinet and Pierre Apkarian. Structured h-infinity synthesis in matlab. *IFAC Proceedings Volumes*, 44(1):1435–1440, 2011.

[9] Steven J Schroeck and William C Messner. On controller design for linear time-invariant dual-input single-output systems. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 6, pages 4122–4126. IEEE, 1999.