Robust Tuning of Fixed Structure Controllers for Hard Disk Drives

by

Bo Zhu

B.S.(Tsinghua University, PRC) 1996

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Andrew Packard
Professor David Tse

Fall 2001

The dissertation of Bo Zhu is approved:

Professor Masayoshi Tomizuka, Chair — Date

Professor Andrew Packard — Date

Professor David Tse — Date

University of California at Berkeley

Fall 2001

Robust Tuning of Fixed Structure Controllers for Hard Disk Drives

by

Bo Zhu

Abstract

Robust Tuning of Fixed Structure Controllers for Hard Disk Drives

by

Bo Zhu

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California at Berkeley

Professor Masayoshi Tomizuka, Chair

This dissertation is concerned with the tuning of fixed-structure controllers and its application in the design of track-following controller for hard disk drives.

Instead of building the controller on an over-simplified nominal plant, a comprehensive method of statistically modeling a large number of drives is considered. The model is built based on the decomposition of PESs (position error signals) collected from multiple drives. It can be used to predict the time-domain performance of a population of drives with a given controller in terms of the mean value of variance of PES and the variance of variance of PES, without tedious time-domain simulations.

The parameter optimization of fixed-structure controller is of great interest in control practice. Due to the structure and order limitations, the problem cannot be parameterized as a convex optimization problem. A large number of approaches focus on making the optimization a convex one through appropriate parameterization and approx-

imation. The motivation is that there are effective and powerful algorithms to solve the convex optimization problem. This dissertation develops a MOGA (multi-objective genetic algorithm) to directly solve the multi-objective non-convex optimization problem. The population-based nature of the MOGA enables the evolution of a set of Pareto-optimal solutions without requiring weights before optimization. Furthermore, due to the stochastic nature of search mechanism, the MOGA is more likely to find the global optimum than conventional optimization methods in a non-convex search space. As shown by simulations and experiments, the proposed method is capable of optimizing the controller in a *large range* in which gradient-based methods generally fail.

While the gradient-based techniques lack robustness over global non-convex optimization problems and are sensitive to initial starting points, they are more efficient than the MOGA in local fine-tuning search. Therefore this dissertation proposes a two-phase algorithm combining the advantage of the MOGA and the gradient-based techniques to further improve the solution quality and computational efficiency.

This dissertation also provides a systematic analysis of the state truncation errors associated with the digital implementation of the track following controller.

Although the methods presented in this dissertation are devised to be applied in the design of HDD track following controllers, the mathematical treatment employed is general and applicable to other engineering applications.

Professor Masayoshi Tomizuka
Dissertation Committee Chair

To my parents

Qinggang and Shifang

and my wife Baochun

Contents

List of Figures **vii**

List of Tables **x**

1 Introduction **1**
1.1 Background and Motivation 1
1.2 Problem Description 5
1.3 Previous Research 7
1.3.1 Previous Research on Modeling HDDs 7
1.3.2 Previous Research on Optimization of Fixed Structure Controllers . 8
1.4 Contribution of Dissertation 10
1.5 Outline of Dissertation 13
1.6 Notation 14

2 Hard Disk Drive Servomechanisms **16**
2.1 Overview of Hard Disk Drives 16
2.2 Position Sensing of Read/Write Heads 18
2.3 Position Error Signal and Track Mis-Registration 19
2.4 Design Challenges for HDD Servo 25
2.4.1 Performance Specifications 25
2.4.2 Design Trade-offs 27
2.5 Design Techniques for Track-following Controller 30
2.5.1 PID 32
2.5.2 LQG/LTR 33
2.5.3 H_∞ Loop-shaping 35
2.6 Summary 37

3 One Model for A Population of Drives **38**
3.1 Introduction 38
3.2 Preliminaries of Statistical Signal Processing 40
3.3 Transfer Function Estimation 44
3.3.1 Swept-sine Estimation 45

3.3.2 Broadband FFT . . . 47
3.4 Nominal Linear Model . . . 49
3.4.1 Plant . . . 50
3.4.2 Fixed-Structure Controller . . . 54
3.5 Decomposition of Position Error Signals . . . 55
3.6 TMR synthesis . . . 64
3.7 Experimental Verification . . . 67
3.8 Summary . . . 73

4 Multi-objective Tuning of Fixed Structure Controller 74
4.1 Introduction . . . 74
4.2 Preliminaries . . . 77
4.3 Overview of Optimization Techniques . . . 78
4.3.1 Gradient-based Algorithms . . . 79
4.3.2 Non-gradient-based Deterministic Schemes . . . 80
4.3.3 Non-gradient-based Stochastic Algorithms . . . 81
4.4 Multi-objective Optimization . . . 82
4.4.1 Weighted Sum Strategy . . . 85
4.4.2 $\varepsilon-$Constraint Strategy . . . 87
4.4.3 Goal Attainment Programming (GAP) Strategy . . . 88
4.4.4 Linear Matrix Inequalities (LMIs) . . . 91
4.5 Parameterization of Fixed Structure Controller . . . 92
4.5.1 Convex Parameterization . . . 92
4.5.2 Direct Optimization by Non-gradient Methods. . . . 96
4.6 Summary . . . 98

5 Parameter Optimization by Multi-Objective Genetic Algorithm 100
5.1 Introduction . . . 100
5.2 General Genetic Algorithms . . . 101
5.2.1 Coding: From Binary to Real . . . 103
5.2.2 Initial Population . . . 104
5.2.3 Cost Function and Constraint Handling . . . 105
5.2.4 Fitness . . . 105
5.2.5 Genetic Operators: Selection, Crossover, and Mutation . . . 107
5.2.6 Solving a Benchmark Problem by GA . . . 109
5.3 Multi-Objective Genetic Algorithm . . . 115
5.3.1 Cost Functions of Multi-objective GAP . . . 116
5.3.2 Pareto Ranking . . . 117
5.3.3 Rank-based Fitness Assignment . . . 118
5.3.4 Intra-Rank Fitness Sharing . . . 119
5.3.5 Elitism and Random Migrants . . . 120
5.4 Example: Optimization of HDD Controller Parameters . . . 121
5.5 Summary . . . 124

6 A Two-Phase Optimization Algorithm **126**
6.1 Introduction 126
6.2 Nonlinear Goal Attainment Programming 127
6.3 Two-Phase Optimization Algorithm 129
6.4 Example: Optimization of HDD Controller Parameters 131
6.5 Summary 134

7 Minimize the State Truncation Error **135**
7.1 Introduction 135
7.2 Truncation Noise Propagation 136
7.3 Implementation Structure Optimization 142
7.4 Summary 146

8 Conclusions and Suggestions for Future Research **148**
8.1 Conclusions 148
8.2 Suggestions for Future Research 150

Bibliography **153**

A Codes of Multi-objective Genetic Algorithm **172**

List of Figures

1.1 Typical flowchart of control system design. 6
1.2 A simple discrete-time SISO system. 11
1.3 Linear fractional tranformation. 14

2.1 A typical rotary servomechanism of modern hard disk drives. 17
2.2 IBM highly compact 1-in Microdrive^{TM} (courtesy IBM, San Jose) 17
2.3 Embedded servo scheme: head position is measured by reading back the servo information embedded between data sectors. Two head positions are shown, one on track and one off track. 20
2.4 Genetic disk drive system (Courtesy Texas Instruments). 21
2.5 PES is a zero mean Gaussian. 21
2.6 The relationship between various track positions and position errors. 23
2.7 A typical blockdiagram of disk drive servo system. 31
2.8 Mixed-sensitivity H_∞ control for track following control. 36

3.1 The time-consuming time domain simulation is the bottleneck. Also, it does not directly give direction to optimizer. 39
3.2 A new design scheme using direct TMR projection in the optimization loop. 40
3.3 Identify the HDD plant dynamics by a two-step swept-sine approach. 46
3.4 A intact system ID based on the broadband FFT technique. 48
3.5 Identification of plant transfer function by the swept-sine technique and the broadband FFT method. 49
3.6 Discrete-time HDD servo model with lumped disturbances. 50
3.7 Simplified plant model for track-following. 51
3.8 Plant model validation through applying broad-band FFT technique to a number of drives. 54
3.9 The four lowest modes, (0,0) at 608 Hz, (0,1) at 619 Hz, (0,2) at 729 Hz, and (0,3) at 1168 Hz are the most significant TMR contributors. 55
3.10 Averaging PES traces over multiple revolutions seperates the repeatable components from the non-repeatable parts. 58
3.11 The variance of PES is not a constant but a random variable. 58
3.12 The PSD of RRO based on 48 PES traces. 59

3.13 The PSD of repeatable position disturbance is recovered from PSD(RRO). The magnitude of bin has a Weibull disturbance. 59
3.14 The random variable used to denote the magnitude of each RRO bin has a Weibull distribution. 60
3.15 Both measurement noise n_m and torque disturbanace d_t are assumed to be white noises. The fittings to the baseline of NRRO are colored versions of N_m and D_t. 62
3.16 The setup for model validation. 70
3.17 Nine models are built by repeating the building process on the same eight drives with nine different loop gains $k^0_{loop}\cdot$[0.92 0.94 0.96, 0.98, 1.00, 1.02, 1.04, 1.06, 1.08]. The trend predicted by models matches the experimental results. 71
3.18 The predicted trend of $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ by varying k_{loop}. 71
3.19 The predicted trend of $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ by varying k_a. 72
3.20 The predicted trend of $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ by varying k_b. 72

4.1 Different methods of dealing with multiobjective optimization problems. The upper part is the conventional weighting-based method and the lower part is the Pareto-based approach. 83
4.2 An example of Pareto optimality. 83
4.3 The convex trade-off boundary is accessible by varying weightings. 86
4.4 Some trade-off solutions on the nonconvex boundary are not accessible by varying w. 87
4.5 Reduction procedure (a) Dynamic output feedback control; (b) Decomposition; (c) Augmentation; (d) Static output feedback control. 93

5.1 The flowchart of a simple genetic algorithm for parameter optimization. It is easier to exemplify genetic operations by using the binary coding. 101
5.2 An example of fitness assignment for a minimizing problem with positive semidefinite objective function. 106
5.3 The probability density of $\phi(x, z)$ with respect to evolving generation t when b=2. 109
5.4 The probability density of $\phi(x, z)$ with respect to evolving generation t when b=5. 110
5.5 The illustration of the non-convex optimization example in Eq. (5.5) when n=1. 111
5.6 The right choice of the populaton size in one generation and the number of generations is very important in GAs. All trials have a total of 3000 samples. 112
5.7 Upon applying the GA to solve the given example, the relationship between the time to convergence ($|x_{i\_opt}| < 10^{-3}$) and the deminsion of search space is almost linear. 113
5.8 The flow chart of MOGA with elitism and random imigrants. 117

6.1 Goal attainment method with two objectives. 128

6.2 When GAP is solved by SQP, starting from the sub-optimal set of MOGA enhances the possibility of reaching Pareto-optimal surface. 130

7.1 Cascade form implementation of track following controller. 136
7.2 Identification of state truncation noise source. 138
7.3 Cascade form: linear model plus quantization noise sources. 139
7.4 Noise transfer functions. 140
7.5 Power spectral density of shaped noise sources. 142
7.6 Linear model of direct IIR form with noise sources. 144
7.7 Use a scaling couple to attenuate the truncation noises. 144
7.8 FFT components of PES for different implementations (Experiment) . . . 146

List of Tables

3.1 Plant parameters 52
3.2 HSA modes 53
3.3 Accuracy evaluation of Modeling 70
3.4 Evaluation of PES prediction capability ($k_{loop} = 1.06k^0_{loop}$) 70

4.1 A summary of evolutionary algorithms. 82
4.2 MOP and multi-objective GAP 90

5.1 The computational cost and solution quality v.s. the number of generations and population size 110
5.2 Robustness comparison with other algorithms 111
5.3 Relation between goals of GAP and cost functions of MOGA 116
5.4 MOGA configurations 121
5.5 Performance comparison before and after MOGA optimization 124

6.1 Two-phase algorithm: MOGA configurations in phase one 131
6.2 The result of solving GAP by SQP highly depends on starting point 132
6.3 Performance comparison among different controllers. 133

7.1 Variance percentage of state truncation noises in total PES: Cascade form (Simulation) 141
7.2 Variance percentage of state truncation noises in total PES with different optimization schemes (Simulation) 144

Acknowledgements

My foremost thanks go to my research advisor, Professor Masayoshi Tomizuka, for his consistent support and guidance throughout my graduate career. I would like to extend my sincere appreciation to Professor Andrew Packard who is always available to help me. I also thank Professor David Tse for reading my dissertation and for being in my qualifying examination committee.

My graduate study was supported by Computer Mechanics Laboratory of the University of California at Berkeley and Maxtor Corporation. Funding for this dissertation research was provided by Maxtor. I would like to give my appreciation to my colleagues at Maxtor's Advanced Technology Group, especially Dr. Lin Guo for his great leadership and Dr. Ho Seong Lee, Yu Sun, Xiaoping Hu, Wei Guo, and David Tung for their generous help.

I would like to offer my gratitude to all the fellow labmates, Erwin Satrya Budiman, Dr. Carlo Cloet, Jiagen Ding, Dr. Kai-Ten Feng, Dr. Yuping Gu, Jihua Huang, Dr. Soichi Ibaraki, Dong-Jun Lee, Lu Li, Guang Lu, Dr. Craig Smith, Shashikanth Suryanarayanan, Ye Sheng, Dr. Meihua Tai, Dr. Jeng-Yu Wang, Ryan Wayne White, Dr. Li Yi, and Kun Zhou with whom I had the greatest time.

I deeply appreciate my wife, Baochun Liu, for her love and support. I would like to thank all my family members, my mother Shifang, father Qinggang, and brother Dong for all their selfless love, patience and encouragement.

Chapter 1

Introduction

1.1 Background and Motivation

Hard disk drives (HDDs) have a short but fascinating history. Since IBM invented the first computer disk storage system, the 305 RAMAC (Random Access Method of Accounting and Control) in 1956, HDDs have evolved from a monstrosity with fifty, two-feet diameter disks holding only five Mbytes of data to today's drives measuring 3.5 inches wide and less than an inch high, holding more than 160 Gbytes. Despite the boom of solid-state non-volatile storage devices, HDDs are still leading the race in terms of capacity and cost per Mbyte, and this trend is expected to continue for a long time.

This evolution of HDDs is a consequence of both market needs and progression of various technologies. In today's highly competitive HDD market, manufacturers have to cut overall cost yet maintain high performance and meet an increasing demand for large capacity. The recording densities in current commercial products are around 50K tracks per inch (TPI) in the radial direction and 400K bits per inch (BPI) in the circular direction,

corresponding to an areal density of 20 Gbits per square inch and a track width of $0.5\mu m$. While the circular density is related to the disk rotation speed and read/write (R/W) channel throughput, the radial density is directly determined by the tracking performance of R/W head positioning systems. Since it is desirable that the track error be less than one eighth of the track width, the servo system needs to keep the R/W head within $0.06\mu m$ of the desired track. The modern magnetic materials have the capability of storing hundreds of Gbits per square inch. Thus, developing an accurate R/W head positioning system is critical to explore such a potential.

HDD servo design is an iterative process involving not only control engineers, but electrical engineers, mechanical engineers, and material scientists. The major difficulties in the design of HDD servo systems are associated with the presence of repetitive and non-repetitive disturbances, non-linear bias forces and frictions, model uncertainties, limited actuator bandwidth, and other system limitations. There are two major ways to further increase the recording density: 1) improve the mechanical structure of actuators, and 2) redesign and optimize the servo controller.

Improving the mechanical structure of actuators typically increases the supportable bandwidth and reduces the disturbances *at source*. Such improvements, however, take a significantly longer time and higher cost than redesigning the controller. Thus actuator redesign is considered to be a major change and seldom happens unless controller performance has already been pushed to the extreme. Neal in [1] made a complete survey of the present and expected future technologies of HDD mechanical design. It is clear that the conventional single-stage actuator can no longer support annual 60% areal density growth

in the coming decade. One reason for this is that the hysteresis effect due to the bearing friction of single-stage actuators limits the servo loop gain at low frequencies, while the closed-loop bandwidth is limited by its low resonance mode. As the track width is reduced to sub-micrometers, the influence of high frequency disturbances on track positioning accuracy becomes more serious. Since the early 1990s, dual-stage servomechanism, in which the secondary actuator is a high bandwidth micro-actuator, has been emerging as a promising solution [2] [3] [4] [5] to further improve the recording density.

On the other hand, redesigning and optimizing the embedded controller for better *disturbance rejection* is a more cost-effective way to reduce the tracking error. The HDD servo system consists of two modes of operation namely, 1) the track-following mode, in which the read-write (R/W) head is positioned on a particular track for read/write actions, and 2) the track-seeking mode, in which the R/W head is moved from one track to another. Some early works on the HDD servo design can be found in [6] [7] [8] [9] [10]. Because of technological advances in the design and implementation of servo controllers, there has been an increased interest since the late 1980s in applying modern control techniques to the design of HDD controllers. For example, the linear quadratic Gaussian control with loop transfer recovery technique (LQG/LTR) has been applied to the design of controllers for single-stage actuators [11] [12] [13] [14] and for dual-stage actuators [2] [15]. Applications of H_∞ loop-shaping are in [16] [17] [18] [19] [20].

As in most of industrial applications, however, the track-following controllers used in production HDDs still adopt simple structures like a PID with notch filters. There are a couple of reasons for slow adoption of modern control techniques by industrial control

engineers. The first reason is associated with the pressure of market and cost constraints. Generally speaking, the more complex the control law, the higher the cost, and the harder the maintenance. During design cycles, servo engineers usually have little control over the selection of the signal digital processor (DSP). The DSP that one can afford to put into a HDD has only few hundreds clock cycles between sample points. Moreover, 95% of clock cycles are consumed by computations that are not related to the update of control output [21]. This prohibits the use of high-order controllers unless a significant performance improvement by such a controller can justify a more expensive DSP. In order to get a manageable order for implementation, it is common practice to apply order reduction techniques on plant or controller. However, this restricts the use of comprehensive models during design. Furthermore, there is no guarantee that the structure of reduced-order controllers is consistent from design to design. If the controller structure is changed in every design cycle, the problems associated with implementation cost, reliability, and maintainability usually override the benefits of using modern control techniques. In fact, *it is more practical and efficient to start with a controller structure that has good nominal properties, and optimize its parameters directly in the discrete-time domain.* Fixing the structure of the controller minimizes the difficulties involved in design iterations. Optimizing the controller directly in discrete-time domain minimizes the performance degradation due to discretization, which is particularly critical in multi-objective designs.

As Steinbuch and Norg [20] have pointed out, some important issues in the design of control systems are largely underemphasized in academic research. Control design is a complete process of arriving at a formal control solution. This process includes, but is not

limited to, defining design objectives, modeling of plant and disturbances, making choices for controller structures and operation points, and implementing control laws. Modern control methods put much effort into the creation of elegant algorithms for control design, while industrial control practices are dominated by efforts to understand a particular physical process or problem. As an example, building effective control systems in the data storage industry is 90% process understanding and 10% control design [21]. This is in contrast to the majority of publications that present a wide range of solutions to neatly solvable mathematical problems. This *gap* deserves more attention in the control community. A design method which requires substantially more time to understand and implement stands little chance of being used in the market-driven design cycles. Moreover, the improvements that can be made by *fundamentally understanding the problem and then applying simple solutions* typically swamp out the improvements achieved by a complex control design.

1.2 Problem Description

This dissertation is focused on the problems associated with the tuning of fixed-structure track-following controllers for HDDs. The design objectives of the track-following controller can be summarized as following.

- In order to achieve higher TPI, minimizing the tracking position error is the major objective. The final performance assessment of a tracking controller therefore is in the time-domain, in terms of variance of the tracking error.
- Robustness is required because the same control algorithm has to be performed in millions of mass produced drives, where each drive has slightly different characteristics

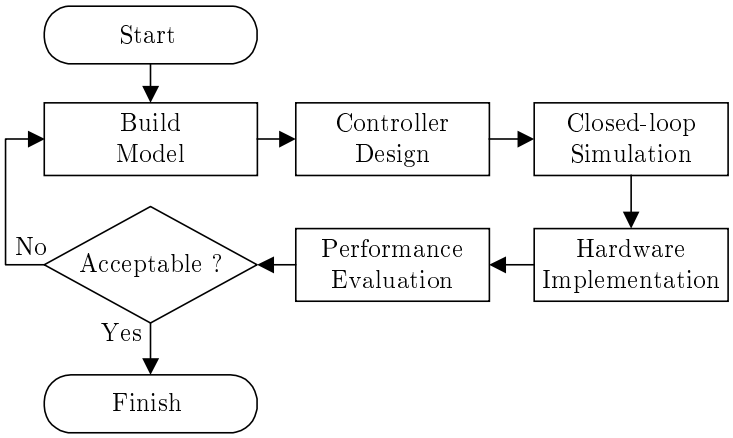

Figure 1.1: Typical flowchart of control system design.

due to manufacturing tolerances, temperature variations, and aging effects.

This is a typical multi-objective control design problem. As illustrated in Figure 1.1, the control system design is an iterative process. Building the model is the first step to design a controller for a target system to achieve multiple performance and stability goals. However, the HDD industry is lacking a non-conservative plant model that 1) covers the characteristics, especially of the time-domain, of a large number of drives, and 2) yet is computationally simple and dynamically upgradeable. Therefore, such a multi-objective design is usually performed in frequency-domain by loop-shaping based on the power spectrum of measured position errors. Time-domain performance is then evaluated by prototype drives and time-domain simulation tools. This is an *indirect* method to design the track-following controller in order to minimize tracking errors.

Tuning the parameters of the fixed-structure controller to achieve multiple objectives and under multiple constraints is of interest in numerous practical applications. Fixing

the controller structure inevitably destroys the convexity of the search space [22]. Gradient-based methods, e.g. [23], have no guarantee of global convergence for such a non-convex multi-objective optimization problem (NCMOP). Even where a convex approximation of the NCMOP is possible, its parameterization would require numerous auxiliary conditions which could make the final formulation too large to solve [24].

For multi-objective optimization problems (MOPs), it is desirable to provide a set of trade-off solutions to the decision maker. Due to the lack of efficient multi-objective optimization algorithms, most methods either combine all objectives into a single objective by using weighted-sum strategy, or prioritize the most important objective and treat others as constraints. Such strategies involve many trial-and-errors due to the need for heuristic tuning of a, usually poorly understood, set of weightings, especially when there are competing objectives. Moreover, some solutions on non-convex trade-off surface may not be accessible (see Section 4.4 for details).

1.3 Previous Research

This section presents a review of previous works on two distinct subjects which are the main themes of this dissertation: modeling HDDs and the optimization of fixed-structure controllers.

1.3.1 Previous Research on Modeling HDDs

In 1997, Abramovitch *et al.* [25] [26] [27] published a series of papers addressing the component-level breakdown of track mis-registration (TMR, See Section 2.3 for details)

sources. The results were based on intrusive experiments where the HDD cover had to be opened, thus this method is not practical for a large quantity of drives in production processes. Guo *et al.* ([28], 1999) proposed a comprehensive time-domain simulation tool for TMR prediction. It serves as a "virtual drive", which allows the prediction of time-domain performance and thus achievable recording density given a particular mechanical platform and controller. Since this tool was designed for a single drive and the time-domain simulation is inherently time-consuming, it was tedious to perform the prediction for a population of drives. Yi ([29], 2000) presented an intact method requiring only PES measurements to build the TMR model for a single drive. Yi's method makes it possible to dynamically update the TMR information in a production line and without damaging the drives. It is the issue of how PES measurements from multiple drives should be transferred into a mathematical model that dictates the design of a servo controller.

1.3.2 Previous Research on Optimization of Fixed Structure Controllers

Parameter optimization under a fixed controller structure has attracted a considerable number of researchers. There are two dominant trends as seen in the literature: 1) approximating the problem into a convex form through suitable parameterization, and 2) direct optimization by non-gradient-based search techniques.

Convex Approximation Nett *et al.* ([30], 1989) showed a systematic method of reducing optimal constrained-structure dynamic output feedback problems as optimal static output feedback problems. This reduction procedure was illustrated in nine special cases which cover most practical applications of interest. The motivation behind Nett's work was the

possible convex parameterization of static output feedback control and widely available convex optimization algorithms. In an accompanying paper by Bernstein *et al.* [31], a general theory of optimal static output feedback compensations was presented. This theory addresses both H_2 and H_∞ performance objectives, and in each case provides a Riccati equation characterization of optimal static output feedback control. Based on the cone complementarity linearization proposed by El Ghaoui ([22], 1997), Ibaraki and Tomizuka ([32], 2000) developed a method to overcome the non-convex rank constraint so that the fixed-order design is solvable by semidefinite programming (SDP). The proposed method was verified in redesigning a track-following controller of HDD. Rotunno ([33], 2000) utilized the H_∞ loop-shaping method to design a fixed-order track-following controller for a dual-stage HDD. It was based on an idea by Cahinet *et al.* ([34], 1994): the non-convex rank constraint is enforced by minimizing the k-th eigenvalue of generalized plant matrix P such that the k smallest eigenvalues of P are forced to be zero, where k is the order difference between P and controller.

Direct Optimization Goal attainment programming (GAP) is a technique used to find the best compromise solutions in MOPs. Initiated by Charnes and Cooper in 1955, GAPs are extensively used in innumerable engineering activities [35] [36] [37] [38] [39] [40]. Instead of finding only the feasible solutions, GAP requires the designer to assign a set of design goals, and enables the designer to minimize the deviation between performance indexes and design goals, or, if possible, attain the design goals with zero deviation. A comprehensive review of GAP methodology and applications can be found in [41].

Genetic algorithms (GAs) are global, parallel search techniques which emulate

natural genetic operations. Due to their potential for global optimization, GAs have been received great attention in many areas [42] [43]. In the field of automatic control, for example, various successful applications have been reported [44] [45] [46] [47]. In recent years, genetic algorithms have been applied to various MOPs. Schaffer ([48], 1985) proposed the vector evaluation genetic algorithm (VEGA) to solve MOPs. Fonseca and Flemming ([49], 1993) explored Pareto-based multi-objective genetic algorithms (MOGAs) which are likely to find multiple trade-off solutions simultaneously. Horn *et al.* ([50], 1994) investigated multi-objective problems via test functions. A good review of MOGAs can be found in Tamaki *et al.* [51].

Patton and Liu ([45], 1994) solved the problem of eigenstructure assignment for robust control design via a combination of genetic algorithm and gradient-based optimization. First, a GA was used to effectively find a sub-optimal solution for the problem above. Then a gradient-based Davidson-Fletcher-Powell (DFP) algorithm took this sub-optimal solution as the initial point from which to search for the optimum locally.

1.4 Contribution of Dissertation

One Model for A Family of HDDs towards Tracking Error Minimization

This dissertation solves an open question raised by Steinbuch and Norg in [20]; i.e., characterizing multimodal identification for mass produced products by developing a statistical model. Such a model should cover the characteristics for a population of drives and yet require little computational effort to predict the time-domain performance for a given controller. The basic idea is pretty straightforward. As shown in Figure 1.2,

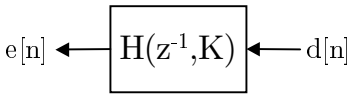

Figure 1.2: A simple discrete-time SISO system.

$H(z^{-1}, K)$ is a fixed-structure discrete-time system where K is the tunable parameter set. Considering a regulator problem, $d[n]$ is the disturbance and $e[n]$ is the error signal. For any given K, $e[n]$ is measurable but $d[n]$ is not. The primary design objective is to optimize $H(z^{-1}, K)$ over K such that the $\|e[n]\|_2$ or $\sigma^2_{e[n]}$ is minimized. Such an optimization problem becomes much more difficult for mass-produced systems like HDDs in track-following mode, where there can be a vast variation on $d[n]$ from one drive to another. This dissertation will try to identify $d[n]$ in frequency-domain based on a group of $e[n]$ measured from a population of drives having the same K. Then a statistical model is built aimed at synthesizing $\|e[n]\|_2$ or $\sigma^2_{e[n]}$ in a computationally effective way for a given K. The proposed technique can serve as an effective evaluation tool during the optimization of K towards $\|e[n]\|_2$ minimization, which could possibly automate the TMR-based controller design process. Experimental validations are performed and results are presented.

Parameterizing Fixed-Structure Control through Goal Attainment Programming

The fixed-structure controller is posed in the form of multi-objective GAP. The proposed formulation is free from dependence on relative weightings and has no convexity limitations of any kind. Thus it is suitable for optimization of *any* fixed-structure controller as long as the user-defined goal functions are calculable. Since some controllers cannot

be transformed into static feedback controller synthesis, LMI-based H_∞ methods are not applicable. To minimize the performance degradation due to discretization, this dissertation uses direct discrete-time design and optimization.

Solving Multi-objective GAP by Multi-objective Genetic Algorithm

This dissertation first demonstrates the superior robustness of GAs with respect to interior point and simplex methods by solving a non-convex benchmark problem. GAs are extended to handle multi-objective optimization by using the Pareto-based fitness assignment. The proposed multi-objective GA (MOGA) needs no weighting prior to optimization and uses embedded parallelism to search for the Pareto-optimal solutions, none of which are better than others in *all* objectives. The goal functions of GAP are used to dictate the fitness assignment of candidate solutions in MOGA. Since the interactions between conflicting, non-commensurable objectives are clearly revealed after optimization, the decision maker can easily pick up one solution from the Pareto-optimal set based on his preference. This greatly reduces the trial-and-errors in design cycles. Another significant advantage of MOGA is that it imposes no requirements like differentiability on the user-defined goal functions of GAP.

Two-Phase Genetic-Gradient Optimization

This dissertation proposes a two-step non-convex optimization method to take advantage of both 1) the robustness of genetic algorithms in identifying high performance regions in a non-convex search space, and 2) the efficiency of gradient-based methods in a local search. The performance goals can be of any type as long as they can be evaluated

in the optimization loop. The effectiveness of the proposed method is demonstrated by simulations and experiments, which also serve to investigate the computational tractability and efficiency of the proposed method.

Truncation Error Analysis in Digital Servo Implementation

This dissertation provides a systematic analysis of the state truncation errors associated with the digital implementation of the track-following controller. Simulation and experiment showed the effectiveness of two optimization schemes, i.e. changing the implementation structure and applying a scaling couple, in minimizing the PES due to state truncation errors.

1.5 Outline of Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 describes the HDD servomechanism and gives a brief review of design techniques for track-following control. The statistical model for a large number of HDDs is presented in Chapter 3. Chapter 4 discusses different parameterizations of optimal fixed-structure control and multi-objective optimization techniques. Chapter 5 proposes a MOGA to solve the GAP parameterization of the fixed-structure track-following controller. In Chapter 6, a two-phase optimization algorithm is presented that takes advantage of the efficiency of gradient-based techniques in local search to further improve the performance of the MOGA. In Chapter 7, a systematic approach is presented from a servo point of view to minimize the effect of state truncation errors on the PES.

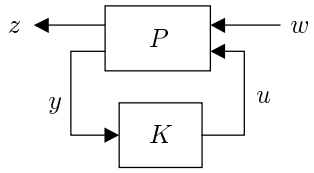

Figure 1.3: Linear fractional tranformation.

1.6 Notation

This section provides background materials, which are essential to understand this dissertation. Advanced readers may skip this section. One can refer to [52] and [53] for more details.

For a square matrix $X \in \Re^{n\times n}$, the following notations are used

$$\begin{aligned}
\mathrm{X} &\succ 0 \Leftrightarrow \mathrm{X} \text{ is symmetric and positivedefinite;} \\
\mathrm{X} &\succeq 0 \Leftrightarrow \mathrm{X} \text{ is symmetric and positivesemidefinite;} \\
\mathrm{X} &\prec 0 \Leftrightarrow \mathrm{X} \text{ is symmetric and negativedefinite;} \\
\mathrm{X} &\preceq 0 \Leftrightarrow \mathrm{X} \text{ is symmetric and negativesemidefinite.}
\end{aligned}$$

Let $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] := C(zI - A)^{-1}B + D$ denote a minimal realization of a discrete-time dynamic system which has following state space realization

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases} \tag{1.1}$$

where $u(k)$ and $y(k)$ are the input and output signal vectors respectively, and $x(k)$ is the state vector.

Figure 1.3 shows a general feedback configuration for the plant $P = \left[\begin{array}{c|c} P_{11} & P_{12} \\ \hline P_{21} & P_{22} \end{array}\right]$

with the controller K as a separate block. To make our notation simple during the analysis of closed-loop performance, K may be absorbed into the interconnection structure

$$F_l(P,K) := P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} = N \tag{1.2}$$

where $F_l(P,K)$ denotes a lower *linear fractional transformation* (LFT) of P with K as the parameter.

The p-norm of a sequence $e[k]$ is defined as

$$\|e[k]\|_p = \left(\sum_k |e[k]|^p \right)^{\frac{1}{p}}. \tag{1.3}$$

The most commonly used signal norms $\|e\|_1$, $\|e\|_2$, and $\|e\|_\infty$ denote the integral absolute value, energy, and peak value of the signal respectively. Given a proper system $e[k] = G(z^{-1})d[k]$ where $d[k]$ is the input signal and $e[k]$ is the output signal, the induced matrix norm describing the amplification or gain of the system matrix G is defined by

$$\|G\|_p = \max_{d \neq 0} \frac{\|e\|_p}{\|d\|_p} \tag{1.4}$$

where $e = Gd$.

Chapter 2

Hard Disk Drive Servomechanisms

2.1 Overview of Hard Disk Drives

Hard disk drives (HDDs) consist of a single disk or stack of disks, which have a thin magnetic coating and rotate at a high speed. Information is recorded on invisible concentric tracks on disk surfaces, by using a read/write (R/W) head integrated into a slider which is mounted at one end of the suspension. The other end of the suspension is connected to the arm which is moved across the disk surface by a high-speed rotary actuator. The slider flies above the rotating disk surface at a distance measured in micro-inches, sustained by an air bearing between them.

HDDs are developed to standard form factors which are stated in terms of the diameter of the disks: 5.25-in (130mm), 3.5-in(95mm), 2.5-in(65mm) and even smaller 1-in(25mm). 3.5-in drives currently dominate the desktop market while almost every mobile computer uses 2.5-in drives. Most current HDDs spin at 5400, 7200 or 10,000 RPM. 15,000 RPM drives are emerging. As a marvelous example of recent developments in HDD tech-

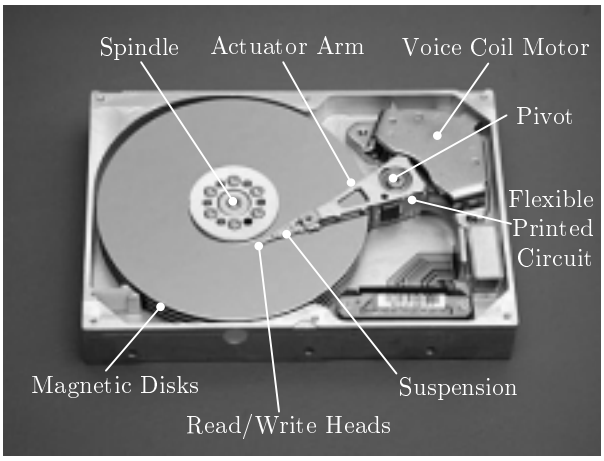

Figure 2.1: A typical rotary servomechanism of modern hard disk drives.

Figure 2.2: IBM highly compact 1-in Microdrive^{TM} (courtesy IBM, San Jose)

nology, in June 2000, IBM announced the 1-in Microdrive^{TM} (Figure 2.2) with a capacity up to 1Gbits paving the way for a new generation of "go-anywhere" pervasive-computing devices and applications.

2.2 Position Sensing of Read/Write Heads

Disk drives using a voice coil motor (VCM) type actuator require a closed-loop position control system to center the R/W head over the target track. The control system needs to know the real-time actual position of R/W head and uses this to compare with the desired position so that a compensating VCM error current can be generated. In the early 1960s, the position of R/W heads was measured by a linear or angular encoder attached to the actuator. However, mechanical resonances, temperature variation, and encoder resolution limit the track density of this indirect method [54]. It was desirable to have a direct method to measure the actual position of R/W heads. In the late 1960s, the encoder method was replaced by *dedicated servo* technology in which an entire disk surface, called servo surface, was reserved to record position information on reference track. One R/W head was dedicated to read the reference track. Other R/W heads in the head stack assembly were assumed to be in the same vertical position with respect to the reference R/W head. The dedicated servo approach gave adequate control performance as long as the offsets caused by manufacturing tolerance, mechanical resonances, and thermal effects were small relative to track width. This is no longer true with the current drives with high track densities. In modern HDD systems it is rare to find the dedicated servo in actual use.

In the late 1970s, the need to increase TPI to gain more capacity motivated HDD manufacturers to move from the dedicated servo to the embedded servo technology [55]. Figure 2.3 illustrates the embedded servo scheme and Figure 2.4 shows the schematic implementation of an embedded HDD servo system. Embedded servo sectors are interleaved with data sectors in a constant periodic manner on the same surface. At the time of man-

ufacturing, servo sectors are written on the magnetic surface of disk as "housekeeping" information by an expensive machine called servowriter, while data sectors are left blank for user to achieve the main purpose of HDD, data storage. In the reading order of R/W head, a servo sector consists of *sector address mark* (SAM), *gray code*, *analog fine position bursts*, and *optional RRO field* [56] [29]. The SAM identifies the beginning of a servo sector and also indicates the sector number. The gray code provides the track number. A gray code takes a binary sequence in which successive numbers differ by only a single bit for its efficiency in computation. Analog fine position bursts are a group of magnetic written-in patterns with certain radial offsets with respect to the track center. Each burst group has a width of a track. The position of R/W heads can be determined by the difference of the signal strength read back from these burst groups. Figure 2.3 shows the four-burst (A B C D) scheme which is being widely used in current commercial products. The servo drives the R/W head such that the difference of detected signal strength between A and B is minimum, and the difference between C and D is maximum. Because of the nonlinear nature of head position sensing mechanism and other nonlinear sources like pivot friction and printed cable force, the actuator dynamics is nonlinear. Although increasing the number of bursts will theoretically improve the dynamic range and the linearity of PES, it suffers from cost consideration, manufacturing difficulties and computational delay.

2.3 Position Error Signal and Track Mis-Registration

Figure 2.6 shows the relationship between various track positions and position errors. Ideal track position is circular centered at spindle pivot. The track position, X_{servo},

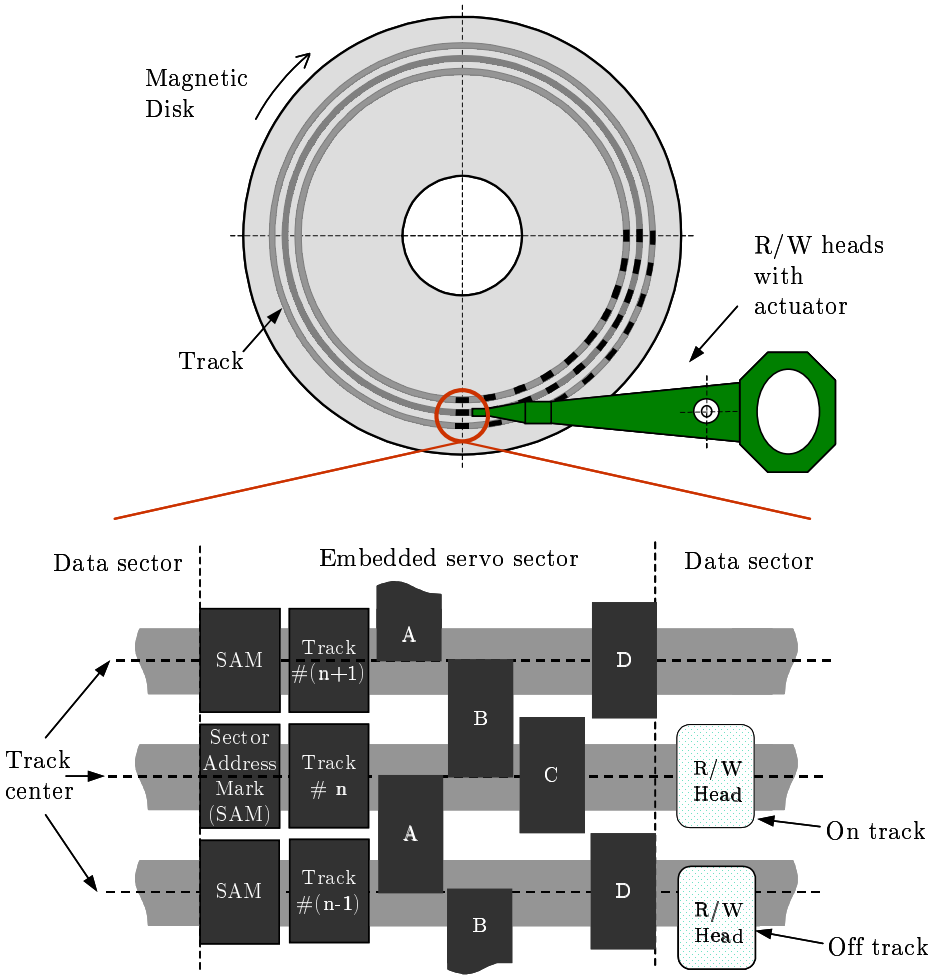

Figure 2.3: Embedded servo scheme: head position is measured by reading back the servo information embedded between data sectors. Two head positions are shown, one on track and one off track.

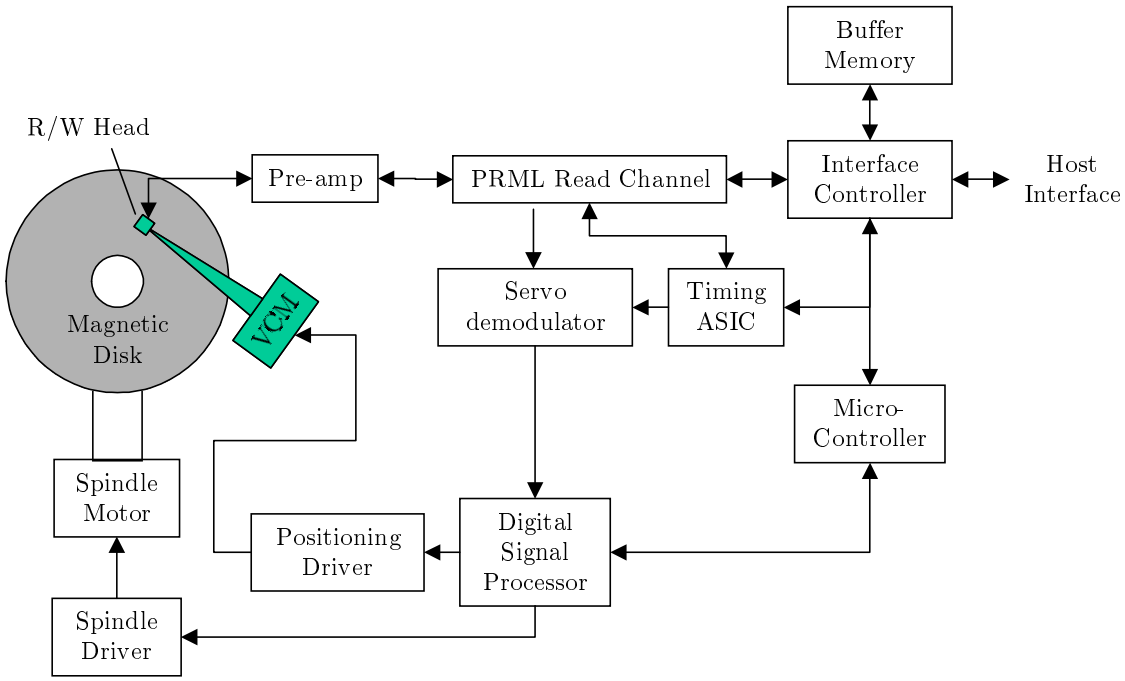

Figure 2.4: Genetic disk drive system (Courtesy Texas Instruments).

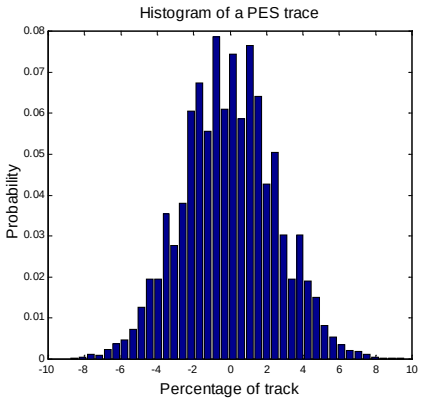

Figure 2.5: PES is a zero mean Gaussian.

written in servo track writing (STW) process is non-ideal due to head and disk motion during STW, and disk motion during track-following. Actually the written track position during STW is a summation of asynchronous pure tones and random noises for only one revolution. The mean track position after STW thus is not an ideal circular. The actuator servo follows the servo written track with error. At some frequencies the error is small, at other frequencies the error is large. This results in the head position, X_{head}, being apparently unrelated to the servo written track position, at times being closer to the ideal track, at other times being further from the ideal track. The head position will be the track position after STW operated upon by the servo system. The mean track-following head position is the head repeatable runout (RRO), $X_{headRRO}$, measured by averaging synchronously with servo index and typically at orders of the spindle rotational frequency. As contrast to RRO, non-repeatable runout (NRRO) denotes all leftover parts of position error that are not synchronized with the spindle rotational frequency. Spindle NRRO vibration combines with the servo system random noise, all operated upon by the servo system, to create an overall head NRRO, which is denoted as $X_{headNRRO}$.

The true position error, tpe, is defined as the difference between the servo written track X_{servo} and the R/W head position X_{head}.

$$tpe = X_{servo} - X_{head} \tag{2.1a}$$

$$tpe = tpe_{RRO} + tpe_{NRRO} \tag{2.1b}$$

$$tpe_{RRO} = X_{servo} - X_{headRRO} \tag{2.1c}$$

$$tpe_{NRRO} = X_{servo} - X_{headNRRO} \tag{2.1d}$$

The measured position error is called position error signal (PES), pes, which is tpe sam-

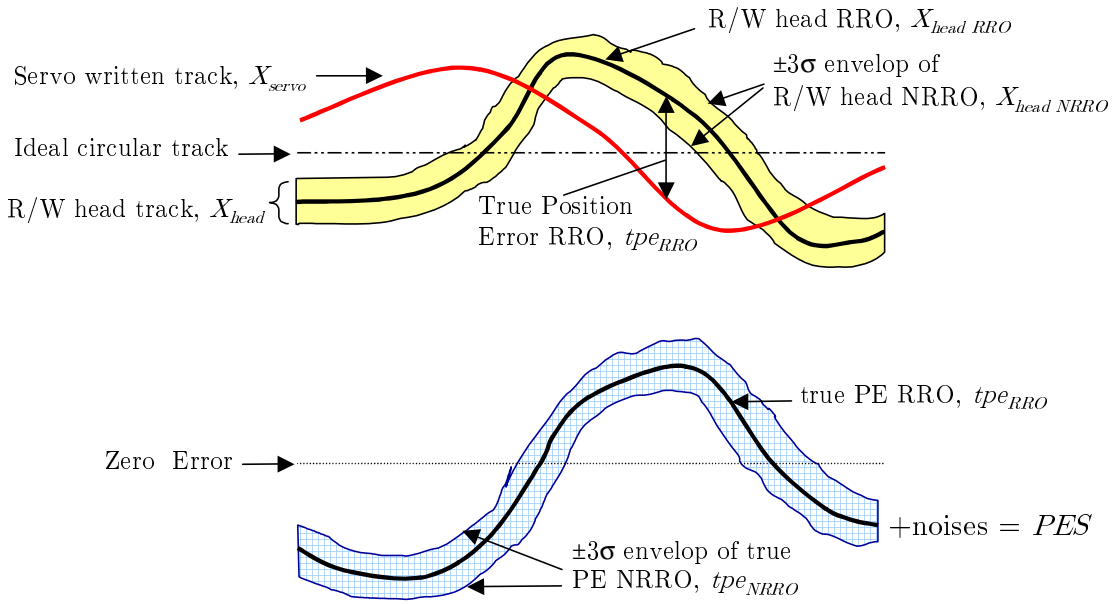

Figure 2.6: The relationship between various track positions and position errors.

pled at servo sectors, contaminated by the read channel noise passing through the PES demodulator. *pes* itself is a zero mean Gaussian process as shown in Figure 2.5.

$$pes = tpe + noise \tag{2.2a}$$

$$pes = pes_{RRO} + pes_{NRRO} \tag{2.2b}$$

$$pes_{RRO} = tpe_{RRO} \tag{2.2c}$$

$$pes_{NRRO} = tpe_{NRRO} + noise \tag{2.2d}$$

Track mis-registration (TMR) is the most important quantity in the HDD design process to evaluate the overall performance of servo systems. TMR is not a number but a statistical distribution, furthermore, a central design method for servo systems that

- provides compatibility between servo, mechanical, read/write channel, servo writer,

head/media, and control logic;

- provides the means to visualize design trade-offs: for a HDD with corresponding off track capability (OTC), TMR budget is a constraint such that within this limit, system can achieve an acceptable soft error rate (SER);
- allows continuous product evaluations for development cycle, manufacturing and planning enhancements.

TMR has two related aspects, write-to-read track mis-registration (W/R TMR) and write-to-write track mis-registration (W/W TMR). The W/R TMR is a measurement of the distribution of the relative misalignment of the head trajectory during data write and data read. This error lowers read signal to noise ratio and thus causes bit errors. The W/R TMR is usually quantified by pes_{NRRO}. The W/R TMR is Gaussian by definition and it is required that the 3σ(W/R TMR) is not greater than 12% of the track width. The W/W TMR is a measurement of the distribution of the encroachment upon the current track by writing operation on adjacent tracks. This part of TMR also causes bit errors. The W/W TMR is quantified by $X_{headRRO}$. It is theoretically possible to measure the track position RRO, $X_{headRRO}$, using a very low bandwidth track follow loop. With such a system the resulting PES RRO, $pes_{low\_BW\_RRO}$, would correspond to the servo track position RRO, X_{servo}. Following relation thus exists.

$$X_{headRRO} = pes_{low\_BW\_RRO} - pes_{RRO} \tag{2.3}$$

3σ(W/W TMR) is acceptable whin 10% ~15% of the track width and with a typical value of 1.3 times the 3σ(W/R TMR). A full discussion of TMR is beyond the scope of this

dissertation; interested readers can refer to [10] [57].

2.4 Design Challenges for HDD Servo

The HDD head positioning system has a great impact on almost every aspect of the overall performance of HDD, which is defined in terms of capacity, speed, reliability, power consumption, acoustic, and so on. The need to simultaneously improve and balance all specifications presents a tremendous challenge to the design of HDD servo systems. Two major functions of HDD servo are track-following and track-seeking. In the track-following mode, the controller functions as a regulator to minimize the on-track position error (*tpe*, not *pes*). In the track-seeking mode, the objective is to minimize the transition time from one track center to another as well as the acoustic noise and power consumption. The interactions between the overall and the servo performances are examined in details in the following subsections.

2.4.1 Performance Specifications

The recording density of HDD is usually described by *areal density*, which measures how many bits of data can be packed into each square inch. Higher areal density is achieved by writing more bits per inch (BPI) in circular direction along the data track and increasing the number of tracks per inch (TPI) in radial direction. The highest achievable areal density in current commercial HDDs is 22 Gbits per square inch (50k TPI and 440k BPI). Achievable BPI is determined by the spindle speed and R/W channel capability while achievable TPI is more related to the servo performance. Improving the regulation performance of track-

following controllers reduces tracking errors. Thus it not only allows more tracks being packed in radial direction, i.e. increases achievable TPI, but decreases soft error rate [10] and improves long term reliability.

The access speed of HDD can be approximated by summing the time contributors of data read/write and transfer. The inverse of this summation gives an approximate throughput rate for random data access in a non-queued operation environment. These time contributors are explained in the following.

- *Seek time* is the time required for the R/W heads to move from one track to another. The number quoted in commercial products is the average seek time between random tracks, normally from 5 ms to 10 ms. In normal operation, switching between adjacent tracks occur much more frequently than random seeks. Thus another specification called *track-to-track seek time* is used to denote the amount of time that the R/W head takes to jump between adjacent tracks. It is usually less than 1 ms. *Full stroke seek time* is the amount of time to move the R/W head from the innermost diameter/track (ID) to the outermost diameter/track (OD).

- *Latency* is another important performance index which is the amount of time that the R/W head must wait to reach the target sector. The latency depends on the spindle rotation speed and the angular distance between the current sector and the target sector. The worst case happens when the R/W head has to wait a full revolution to reach the target sector. The average latency is half the time of a full disk rotation, i.e., $\frac{30}{RPM}$ second, which is 5.6 ms for 5400 revolution per minute (RPM) drives, 4.2 ms for 7200 RPM drives, and 2.8 ms for 10,000 RPM drives.

- *Command overhead time* includes all computational delay imposed by implementing commands in embedded microprocessors. For digital control systems, the effect of delay is significant and thus must be taken into account in the design process.

- *Data transfer time* refers to the time interval between reading (writing) data bits from (to) disk/head interface and sending (getting) them to (from) host interface, as shown in Figure 2.4.

2.4.2 Design Trade-offs

The rapid development of control and data channel circuitries leads to a great improvement in the command overhead time and data transfer time. Actuator seek time and latency are still the largest contributors of HDD throughput performance. Seek time is determined by the performance of track-seeking controller. A well-designed track-seeking controller not only has a short seek time, but also reduces the power consumption and leads to quiet operation. Increasing the spindle speed, RPM, apparently will reduce the latency and enhance the bit transfer rate for a given BPI. However, it also adversely affects power consumption, acoustics, aerodynamic excited vibration of disks and actuator, and spindle dynamics, and may necessitate reduced BPI for a given channel speed.

Another important trade-off is between the sampling rate f_s, the spindle speed RPM, and the number of sectors per track N_{sector}. The sampling frequency is determined by RPM and N_{sector} as

$$f_s = \frac{RPM \cdot N_{sector}}{60}. \tag{2.4}$$

Because of the interleaving nature of embedded servo sectors, the embedded servo system is

a sampled system which leads itself to a digital implementation of controller. It is desirable to have a high sampling frequency for the digital control. Due to the aforementioned reasons, the choice of the spindle speed RPM is limited and it cannot be arbitrarily selected. The number of sectors on each track also should not be too large so as to use as much disk surface as possible for data storage. All are limiting the sampling frequency in HDD servo system. For example, for a 5400 RPM drive with 120 sectors, the sampling frequency is 10.8KHz. This is considered to be relatively low as the allowable track error is only one tenth of a micrometer. If the digital controller is converted from a continuous-time design under such a low sampling rate using discretization transformations, e.g. bilinear transformation, some performance degradation is expected. Therefore, *it is always desirable to be able to design directly in discrete-time.*

One way to reduce *tpe* is through improving the mechanical structure which reduces the disturbances and noises *at source*. Another method is to loop-shape the controller such that better disturbance rejection is attained. By loop-shaping we mean a design procedure that involves explicitly shaping the magnitude of different loop transfer functions. It starts with a nominal controller or a controller from the last design cycle. Then PES is collected and the spectrum of PES is calculated. The desired loop characteristics are then determined by comparing the PES spectrum with the bode plots of various loops.

However, there are always trade-offs in loop-shaping. The sensitivity function S governs how the position disturbance d_p and measurement noise n_m go through the system and show up at the track error *tpe*. If we make it less sensitive to disturbances at some frequencies, we then make the system more sensitive at other frequencies. This is called

waterbed effect and can be explained by the well-known Bode integral theorem by Bode [58]. Freudenberg [59] extended this theorem to unstable plants. Here is its discrete-time version by Mohtadi [60].

Theorem 1 (Bode Sensitivity Integral for Discrete-time Systems) *For all closed-loop stable discrete-time feedback systems, the sensitivity function has to satisfy the following integral constraint:*

$$\frac{1}{\pi}\int_0^{\pi} \ln\left|S(e^{j\omega})\right| d\omega = \sum_{i=1}^{m} \ln|\beta_i| \tag{2.5}$$

where β_i are the open-loop unstable poles of the system and m is the total number of these poles.

There are couple of important suggestions by this theorem. First, the theorem indicates that some of our objectives are competing during loop-shaping. For example, if PES spectrum tells us there is a mode below the crossover frequency which needs to be attenuated, we push down $|S(j\omega)|$ around the frequency of that mode. This may inadvertently amplify some noises at high frequencies because the gains after the crossover frequency are pumped up due to the waterbed effect.

Since most energy of PES is concentrated at low frequencies, one way of improving serve error rejection is by increasing the servo bandwidth. However, the Bode theorem suggests that for a given sampling rate (or say Nyquist frequency), we cannot push up the crossover frequency without degrading the high frequency performance. In the other words, high bandwidth implies high power consumption, amplification of audible noise, and last but not least poor robustness against variations of resonance peaks, time delay, and unmodeled

high frequency dynamics. One may suggest increasing the sampling rate such that there is more room to spread out the amplification. Unfortunately, increasing the sampling rate may not be able to improve the performance because the maximum attainable bandwidth is actually limited by the computational time delay or the lowest actuator resonance. For the basic rotary actuator used in today's HDDs, several design parameters limit the servo bandwidth, e.g., the in-plane out-of-phase bending mode of the arms and the electrical rise time of the VCM [1]. As a rule of thumb, the servo bandwidth frequency is limited by 25% of the frequency of the lowest actuator resonance, often either the first suspension torsional (T1) mode or the first actuator lateral (butterfly) mode. Therefore a faster DSP chip and a secondary high-bandwidth actuator are more helpful in improving the HDD performance.

2.5 Design Techniques for Track-following Controller

One of the main purposes of this dissertation is to develop a tuning method for fixed-structure controllers. For the simplicity of track-following controllers, it serves as the study-case in this dissertation. Figure 2.7 shows a typical blockdiagram of HDD track-following servo along with various disturbances and noises [10] [61]. C represents the digital controller and P represents the plant dynamics, including power amplifier (PA), voice coil motor (VCM), and head-disk assembly (HDA). There are two categories of disturbance sources: mechanical disturbances and electrical noises [62].

- On mechanical side, d_f denotes the force disturbances such as external shock, pivot friction, flex cable bias, and windage. d_p denotes the position disturbances including spindle motor vibration, disk flutter, and slider vibration.

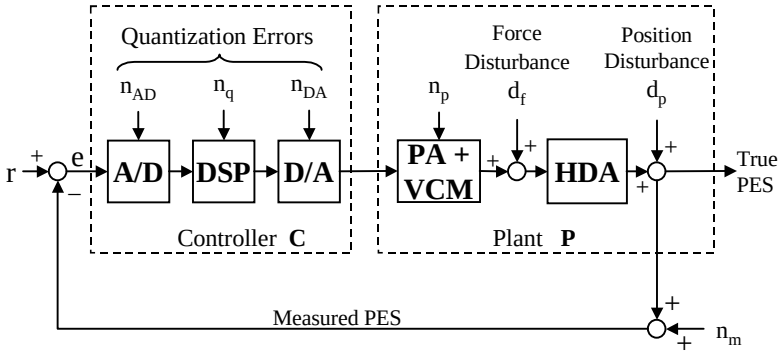

Figure 2.7: A typical blockdiagram of disk drive servo system.

- On the other hand, electrical noises consist of measurement noise n_m, power amplifier noise n_p, and quantization errors n_{AD}, n_{DA}, and n_q from A/D, D/A, and fixed-point DSP respectively. A detailed analysis of the effect of n_q on servo performance is presented in Chapter 7.

Track-following performance of HDDs determines the recording density and overall system robustness. Track-following is a regulation problem, therefore the major design objective of track-following control is to minimize the *true* tracking error *tpe*, which is, however, immeasurable. The position error signal *pes* being measured is the true position error *tpe* contaminated by the measurement noise n_m. Typical time-domain specifications are $|tpe[n]| \leqslant 12\%$ trackwidth and $3\sigma(tpe[n]) \leqslant 5\%$ trackwidth. Fortunately, n_m is shown to be white [63] thus minimizing the power of *pes* is equivalent to minimizing the power of *tpe*. The design also needs to have enough robustness due to the same servo algorithm having to perform in millions of disk drives that are mass produced, each with slightly different characteristics due to manufacturing tolerances, temperature variations, and aging effects.

In this dissertation, all constraints are related to the frequency-domain requirements, such as the minimum phase margin which is a direct safeguard against time delay uncertainty, the minimum gain margin which is a direct safeguard against steady-state gain uncertainty, the minimum crossover frequency, and the maximum peak of sensitivity function.

The PID-like controllers based on classical techniques still dominate the cost-sensitive industrial applications like HDDs. Due to the technological advance in the design and implementation of sophisticated controllers, there has been an increased interest in applying modern control techniques, such as LQG/LTR and H_∞ loop-shaping, to the design of HDD controllers since the late 1980s. The remainder of this section gives a review of different approaches on the design of HDD track-following controller.

2.5.1 PID

Analog PID control has been used in track-following control for years. Typically, an analog PID tracking controller has a crossover frequency of 400Hz and a phase margin of 35 degrees. One of the weaknesses of PID control was its large integrator time constant needed to maximize loop phase margin and low frequency gain for systems with low bandwidth. Analog design for HDD servo began to come to the end in the late eighties when it could no longer provide solutions for the increasingly complex control problem associated with increasing data track density, faster access requirements, and smaller board footprint. The adoption of DSP chips in HDD servo systems since the early nineties has not only solved the limited footprint problem, but significantly improved the performance and been proved to be cost effective [64]. The most straightforward application of digital track-following control is to use digital PID control. The crossover frequency of a typical digital PID track-

following controller is around 600 Hz and phase margin is slightly over 40 degrees. It must be pointed out that part of the improvement in bandwidth and phase margin was due to improvement in the R/W head and actuators.

Tuning of PID controllers belongs to a class of fixed-structure controller optimization problems, which are the main focus of this dissertation. It will be discussed in details in Chapter 4. There are numbers of methods to heuristically tune the PID controller for HDDs, such as Ziegler-Nichols method [65], pole placement [66], loop-shaping [67], H_∞ optimization [32], and mixed H_2/H_∞.

Usually, notch filters are cascaded with PID controller to tackle narrow mechanical resonances modes. The notch filter is designed separately and before the tuning of PID controller.

2.5.2 LQG/LTR

In HDD servo system, the R/W head position is usually the only measurable state. The observer based output feedback control is a right fit [68]. Formulating the problem in the state space setting allows to use some popular techniques such as LQG/LTR and H_∞ loop-shaping. Basic LQG control is the combination of linear quadratic regulator which provides the optimal state feedback control, and Kalman filter which is an optimal stochastic state estimator under fictitiously introduced process noises [69] [70]. The LQG approach proved particularly suited to meet performance specifications while guaranteeing closed-loop stability. The performance index is usually a balance of minimizing track error and minimizing control effort, defined in the weighted-sum quadratic form. Given the weighting of performance index and the power of fictitious noises, the feedback gain and Kalman

filter gain are designed separately based on the separation theorem. However, the choice of weights and power is not directly connected with the desired closed-loop characteristics.

In order to simplify the trial-and-error procedure needed for weight selection, the LQG control combined with loop transfer recovery, named LQG/LTR, is extremely useful from the practical viewpoint of HDD servo design. The basic steps of LQG/LTR are as follows.

- Based on measured PES spectrum and various frequency-domain requirements, design an cascaded weighting compensator that gives a satisfactory open-loop dynamics. Now the extended plant model includes the dynamics of this shaping compensator.
- Generate a target feedback loop with a selection of an estimator based on the extended plant model. Kalman filter is favored because of assured stability and excellent robustness properties.
- Recover the TFL through a cheap LQ feedback control.

Applying LQG/LTR control to the design of HDDs servo is presented in vast literatures for both single-stage actuator [11] [12] [13] [15] [14] and dual-stage actuator [2] [15]. Hanselmann *et al.* ([11], 1998) applied the continuous-time LQG/LTR methodology to design a track-following controller for a highly resonant disk drive. Yen *et al.* ([2], 1990) used a discrete LQG/LTR technique to design a tracking following controller for a dual-stage HDD actuator. Chiu *et al.* ([12], 1993) used frequency shaped LQG/LTR design to improve the runout performance of a disk file system. Weerasooriya *et al.* ([13], 1995) presented a discrete-time LQG/LTR design of a HDD track-following servo system. One important

point made by Weerasooriya [13] is that the excellent robustness and performance characteristics of a continuous-time LQG/LTR design are theoretically unachievable due to the relative low sampling rate and non-minimum phase plant characteristics in HDDs. Therefore the most efficient path to design track-following controllers is to use direct discrete-time design and optimization. Al-Mamun ([18], 1995) presents a two-step H_∞/LTR method. The loop transfer recovery (LTR) is achieved by minimizing the H_∞norm of a transfer function which represents the difference between the loop gain of the static full state feedback and feedback with observer.

The knowledge of disturbances and noises is often taken into account in loop-shaping. Chang ([14], 1999) designed a track-following controller using discrete-time LQG/ LTR. A noise model is augmented into actuator nominal plant to shape sensitivity and co-sensitivity such that TMR budget could be reduced. Lin ([71], 2000) augmented the disturbance and noise covariances into discrete-time LQG/LTR formulation. Optimization is performed by varying two tuning factors to achieves a minimum $3\sigma_{tpe}$.

2.5.3 H_∞ Loop-shaping

H_∞ loop-shaping is to explore the loop-shaping design procedure using H_∞ optimization theory. First, appropriate weighting functions are defined to reflect disturbance and desired closed-loop characteristics. Second, corresponding H_∞-norm optimization problem is solved to give a full-order controller $C(z^{-1})$. Figure 2.8 shows a typical S/T mixed-sensitivity optimization setup for HDD servo design where S is the sensitivity function and T is the closed-loop complementary sensitivity function. Unlike the loop-shaping in LQG/LTR which loop-shapes the open-loop transfer function, mixed-sensitivity design loop-shapes the

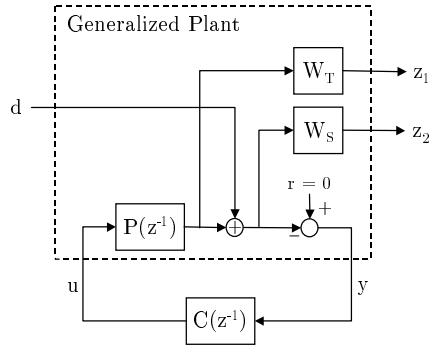

Figure 2.8: Mixed-sensitivity H_∞ control for track following control.

closed-loop transfer functions such that the noises and disturbances will be better attenuated. Actually Skogestad *et al.* [53] showed the open-loop properties, e.g. gain margin and phase margin, are closely related to $\|S\|_\infty$ and $\|T\|_\infty$ which are the peak values of $\left|S(e^{j\omega})\right|$ and $\left|T(e^{j\omega})\right|$ respectively. Furthermore, $\left|S(e^{j\omega})\right|$ and $\left|T(e^{j\omega})\right|$ directly govern the propagation of disturbances and noises to *tpe*. The performance dynamic filters W_S and W_T specify the desired shape of $\left|S(e^{j\omega})\right|$ and $\left|T(e^{j\omega})\right|$ in the way that $\left|S_{desired}(e^{j\omega})\right| < \frac{1}{|W_S(e^{j\omega})|}$ and $\left|T_{desired}(e^{j\omega})\right| < \frac{1}{|W_T(e^{j\omega})|}$ for all ω up to the Nyquist frequency. Therefore, loop-shaping of $\left|S(e^{j\omega})\right|$ and $\left|T(e^{j\omega})\right|$ is equivalent to solving the following H_∞ optimization problem

$$\min_C \left\| \begin{array}{c} W_S(e^{j\omega})S(e^{j\omega}, C) \\ \\ W_T(e^{j\omega})T(e^{j\omega}, C) \end{array} \right\|_\infty < 1. \tag{2.6}$$

Additional closed-loop transfer function (e.g. $F = \frac{P}{1+PC}$ for torque disturbance attenuation) can be added to the formulation above as an additional loop-shaping requirement. Because we are trying to push the performance to achievable limits, dictated by the Bode integral theorem, the iterative tuning of weighting filters based on the disturbance and performance specifications is the most difficult part of this technique [20].

The applications of H_∞ loop-shaping in HDD can be found in [16] [17] [18] [19] [20]. Hirata ([17], 1999) took into account the intersampling behaviors of the HDD digital controller by applying the sampled-data H_∞ control synthesis. H_∞ loop-shaping also attracts lots of interests in the control design of dual-stage HDD due to its significant advantages in handling multi-input multi-output (MIMO) systems [33].

2.6 Summary

In this chapter, an overview of HDD servomechanism was presented. The measurable position error signal (PES) is the true position error contaminated by measurement noise. A detailed breakdown of PES was given in this chapter and will be used as the basis for the statistical modeling of a large number of HDDs in Chapter 3. The design of HDD servo systems is always subject to numerous constraints and trade-offs.

Since this dissertation uses the design of track-following controllers as the illustration example, the last section of this chapter had a review on the design techniques for track-following controllers. The basic method is an signal-based loop-shaping process with experiments within each iterative cycle. Although there are great efforts in applying modern control techniques, such as LQG/LTR and H_∞ control, to the design of HDD track-following controller, most modern HDDs still use PID-like controllers plus notch filters. One of the down sides of modern control methods is that they usually yield a high order dynamic controller, which either is not implementable in on-drive DSP or causes too much implementation hassle.

Chapter 3

One Model for A Population of Drives

3.1 Introduction

For a typical design cycle in HDD industry in Figure 3.1, track-following servos are designed by loop-shaping in frequency-domain before a drive is built. To evaluate the servo performance, PESs are measured from prototype drives and TMR analyses are applied. In [28], PES was broken down into contributors. A comprehensive time-domain simulation tool, called virtual drive (Figure 3.1), was built for controller evaluation and TPI predication. Designers then adjust frequency-domain objectives according to time-domain results. Apparently, the evaluation of time-domain performance is not an objective directly associated with optimizer. It is an indirect and inefficient way to tune controller because the final performance assessment of track-following controller is in time-domain, i.e. in

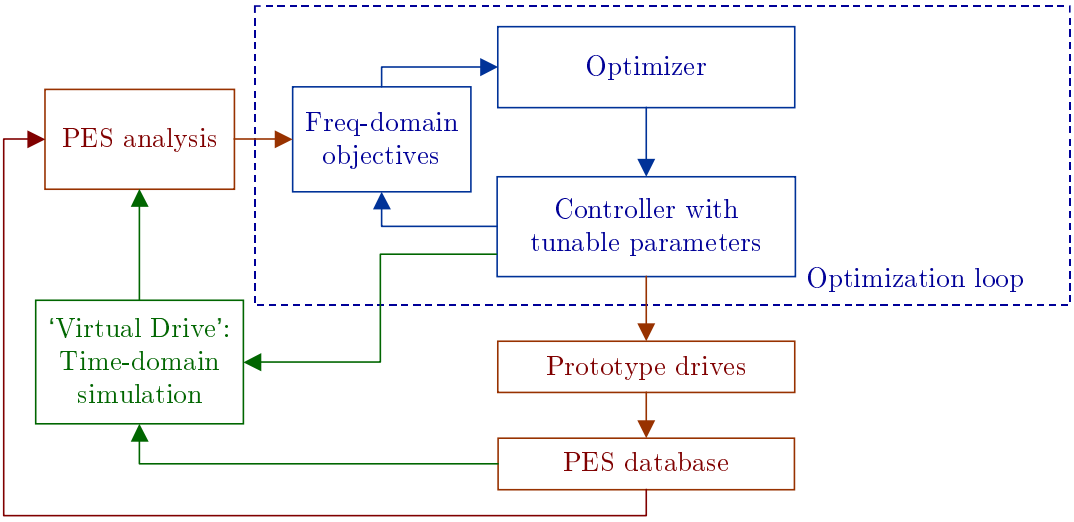

Figure 3.1: The time-consuming time domain simulation is the bottleneck. Also, it does not directly give direction to optimizer.

terms of variance of PES. On the other hand, current PES models are single-drive based, which only allow the candidate controller to be evaluated on one drive per simulation. However, we are designing controllers for mass produced HDDs. In order to get a precise statistical prediction of time-domain performance, this time-consuming simulation needs to be repeated on a large number of drives. It is impractical to dynamically include this tedious process in the parameter optimization loop.

To overcome the above dilemma, this chapter proposes a novel method of building a drive model which not only describes the characteristics of large population of drives but requires minimum calculation efforts for time-domain performance projection. The model will be finally used to synthesize two major time-domain performance indexes, average performance $mean(\sigma_{pes})$ and performance robustness $std(\sigma_{pes})$, for a given controller. This makes it possible to directly optimize the controller towards the minimization of PES, without tedious time-domain simulations (Figure 3.2). In [72], a multi-objective genetic

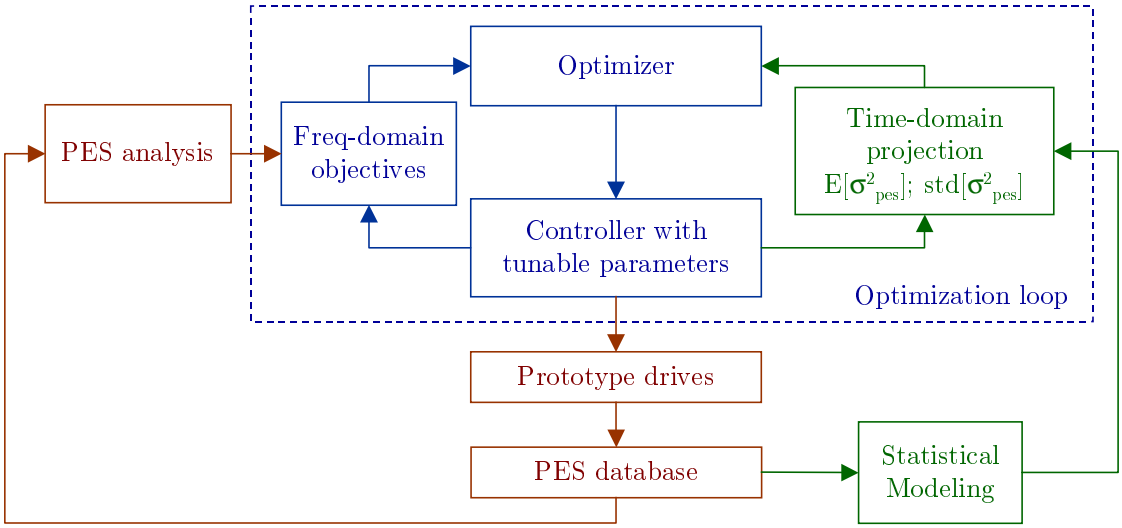

Figure 3.2: A new design scheme using direct TMR projection in the optimization loop.

algorithm (MOGA) was used to optimize the controller parameters based on this TMR projection and other design constraints and objectives.

The model building is a two-step process. First, a nominal linear plant model is constructed based on multi-model identification performed on a large amount of drives. Secondly, repeating PES decomposition on the same drives leads to a statistical model of disturbances. Experimental verification of the model is given at end of this chapter.

3.2 Preliminaries of Statistical Signal Processing

Following are some fundamental yet important definitions and theorems that will be used in the development of the aforementioned statistical model. For a systematic discussion on the statistical signal processing, readers may consult Oppenheim and Schafer [73], [74], and the references therein.

Definition 2 (Fourier Transform for Sequence) *The* Fourier transform (FT) *of a sequence* $x[n]$ *is given by*

$$X(e^{j\omega}) := \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \tag{3.1}$$

and the inverse Fourier transform *is*

$$x[n] := \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega \tag{3.2}$$

Definition 3 (Discrete Fourier Transform) *Given a N-point sequence* $x[n]$*, the* discrete Fourier transform (DFT) *is give by*

$$X[m] := \sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)mn} \tag{3.3}$$

and the inverse discrete Fourier transform *is*

$$x[n] := \frac{1}{N}\sum_{m=0}^{N-1} X[m]e^{j(2\pi/N)mn}. \tag{3.4}$$

Note that $FT(x[n]) = X(e^{j\omega})$ is a continuous periodic function, while $DFT(x[n])$ $= X[m]$ is a finite sequence. The relationship between them is

$$X[m] = \begin{cases} X(e^{j\omega})\big|_{\omega=\frac{2\pi}{N}m} & 0 \leqslant m \leqslant N-1 \\ 0 & otherwise \end{cases} \tag{3.5}$$

The fast Fourier transform (FFT) algorithms implemented in all commercial software, e.g. MATLAB and MathCAD, only provide $DFT(x[n]) = X[m]$.

Definition 4 (Parseval's Theorem for FT) *With* $x[n] \in C^{N\times 1}$ *as a square integrable sequence and* $X(e^{j\omega})$ *as its FT*

$$\sum_{n=-\infty}^{\infty} \|x[n]\|^2 := \frac{1}{2\pi} \int_{-\pi}^{\pi} \left\|X(e^{j\omega})\right\|^2 d\omega \tag{3.6}$$

where $\left\|X(e^{j\omega})\right\|^2$ is called the energy density spectrum.

Definition 5 (Parseval's Theorem for DFT) *For N-point sequence $x[n] \in C^{N\times 1}$ and its N-point DFT $X[m]$,*

$$\sum_{n=0}^{N-1} \|x[n]\|^2 := \frac{1}{N} \sum_{m=0}^{N-1} \|X[m]\|^2 . \tag{3.7}$$

Remark 6 *Parseval's Theorem for DFT is a more practical relationship as we always deal with DFT of signals in data processing.*

While stochastic signals are not absolutely summable or square summable and consequently do not directly have Fourier transforms, many of the properties of such signals can be summarized in terms of the autocorrelation or autocovariance sequence, for which the Fourier transform often exists.

Definition 7 (Autocorrelation Sequence) *Let $x[n]$ be a wide-sense stationary sequence. Its* autocorrelation *sequence is given by $R_{xx}(k) := E(x[n]x[n+k]) = R_{xx}(-k)$*

Definition 8 (Power Spectral Density) *The* power spectrum *of a stationary random process $x[n]$ is given by the Fourier transform of its autocorrelation,*

$$S_{xx}(e^{j\omega}) := \sum_{k=-\infty}^{\infty} R_{xx}[k]e^{-j\omega k} \tag{3.8}$$

where ω is the normalized frequency. In terms of physical frequency f (e.g., in hertz), it becomes

$$S_{xx}(f) := \sum_{k=-\infty}^{\infty} R_{xx}[k]e^{-j2\pi\frac{f}{f_s}k} \tag{3.9}$$

where f_s is the sampling frequency. The power spectral density (PSD) *of $x[n]$ is defined as*

$$P_{xx}(e^{j\omega}) := \frac{S_{xx}(e^{j\omega})}{2\pi} \tag{3.10}$$

and

$$P_{xx}(f) := \frac{S_{xx}(f)}{f_s}. \tag{3.11}$$

On the other side, the correlation sequence can be derived from the PSD by use of the inverse Fourier transform.

$$R_{xx}[k] = \int_{-\pi}^{\pi} P_{xx}(e^{j\omega})e^{j\omega k}d\omega = \int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} P_{xx}(f)e^{j2\pi\frac{f}{f_s}k}df \tag{3.12}$$

Remark 9 *Let $y[n] = H(z^{-1})x[n]$, we have $P_{yy}(e^{j\omega}) = \left\|H(e^{j\omega})\right\|^2 P_{xx}(e^{j\omega})$.*

For a *real* signals, $P_{xx}(e^{j\omega})$ is a even function, i.e. $P_{xx}(e^{j\omega}) = P_{xx}(e^{-j\omega})$, thus $R_{xx}[k] = 2\int_0^{\frac{f_s}{2}} P_{xx}(f)e^{j2\pi\frac{f}{f_s}k}df$. This is a very useful relation because all signals are assumed to be real and stationary in the dissertation.

In this dissertation, MATLAB Signal Processing Toolbox [74] is used to estimate the PSD of a real finite-length signal. It adopts the *one-sided PSD* defined as

$$\Phi_{xx}(e^{j\omega}) = \begin{cases} 2P_{xx}(e^{j\omega}) & 0 \leqslant \omega \leqslant \pi \\ 0 & -\pi \leqslant \omega < 0 \end{cases}. \tag{3.13}$$

For convenience, from this point, PSD refers to the one-sided PSD.

Definition 10 (Average power over a frequency band) *The* average power *(mean square value) of a signal over the frequency band $[f_1, f_2]$ is defined as*

$$\overline{P}_{[f_1,f_2]} = \int_{f_1}^{f_2} \Phi_{xx}(f)df, \quad 0 \leqslant f_1 < f_2 \leqslant f_s \tag{3.14}$$

Since $R_{xx}[k] = \int_0^{\frac{f_s}{2}} \Phi_{xx}(f) e^{j2\pi \frac{f}{f_s} k} df$, the average power of the real sequence $x[n]$ over the entire Nyquist interval $[0\ \frac{f_s}{2}]$ is

$$R_{xx}[0] = \int_0^{\frac{f_s}{2}} \Phi_{xx}(f) df. \tag{3.15}$$

On the other hand, $R_{xx}[0] = E(x^2[n]) = \sigma_x^2 + (E(x[n]))^2$. Therefore, we can estimate the variance of a sequence

$$\sigma_x^2 = \int_0^{\frac{f_s}{2}} \Phi_{xx}(f) df - (E(x[n]))^2 \tag{3.16}$$

3.3 Transfer Function Estimation

Transfer functions are sinusoidal steady-state output to input ratios, which usually are complex numbers. It is assumed that the system is linear or system operates in its linear range when a transfer function is being measured. As an unique example of electro-mechanical control systems, the head positioning servo of HDDs has higher noise level and less linearity than typical electrical or purely mechanical systems. The mechanical imperfections of the servo track and head-disk assembly introduce both periodic and random signals into the servo system. The non-linear behavior is primarily due to the position error signal which has a limited linear region. As the head is driven further off its target track, the position error signal becomes increasingly non-linear. The end result of this combined non-linearity is a measurement challenge.

There are two type of techniques in identifying a dynamic system, the classical swept-sine technique and the broadband FFT technique. The *broadband FFT* technique computes the transfer function over a band of frequencies simultaneously. In order to ac-

complish the broadband measurement, the excitation to the system must contain frequency components that cover the selected frequency range. The excitation is usually a periodic chirp or white noise. Based on the definition of transfer function, the *swept-sine* technique excites the system by using one single frequency sine signal a time, over a user defined frequency range. Compared with the broadband technique, the swept-sine approach is least effected by noise and non-linear system behavior, however, at the expense of measurement time. When time is not an issue, the swept-sine approach provides a more accurate estimate of transfer functions for dynamic systems like HDDs, which are noisy and not perfectly linear. On the other hand, it is not practical to measure every drive by using the swept-sine technique, thus the "quick-and-dirty" estimate of transfer functions is usually broadband FFT based which provides reasonable accuracy within seconds and keeps drives intact.

To balance accuracy and efficiency, a linear model is first built based on the swept-sine estimate from a couple of drives and then verified by the broadband measurements over a large population of drives.

3.3.1 Swept-sine Estimation

A multi-channel dynamic signal analyzer, DSPT SigLab 20-22, is used to implement the swept-sine measurement on our experimental HDDs with parameters listed in Table 3.1. SigLab is connected to a PC through a SCSI card, as shown in Figure 3.3. Setup and control of all SigLab measurements are through a graphical user interface (GUI) coded in MATLAB, running on a PC. There are two input channels and two output channels on SigLab 20-22. Making transfer function estimate on the noisy non-linear HDD system is not as straightforward as in noise-free linear systems. Our goal is to identify the open-loop

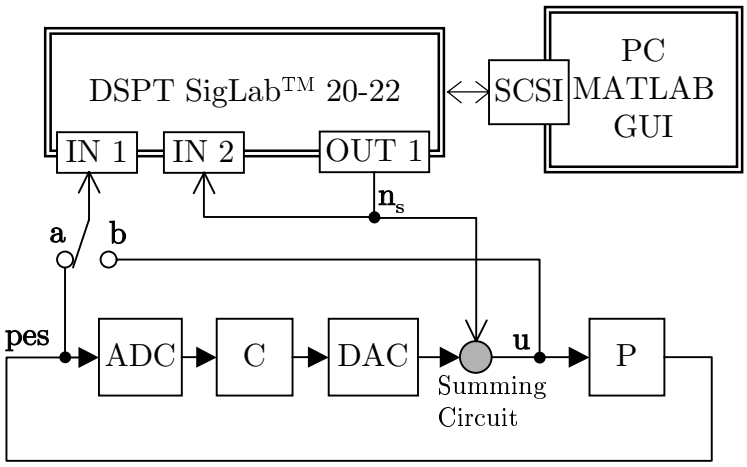

Figure 3.3: Identify the HDD plant dynamics by a two-step swept-sine approach.

and plant dynamics. However, the track-following loop must be closed because the plant is measurable only when R/W heads are on track. A practical solution is to add a summing circuit in the feedback loop. This allows the measurement instrumentation to inject a signal into the system . In this dissertation the summing circuit is between the controller DAC output and the power amplifier input, as shown in Figure 3.3. Note that the sampling of the position error signal $pes[n]$ is performed on the input side of the analog-to-digital converter (ADC) in servo demodulator so as to fully utilize the high-resolution ADC coming with the SigLab. However, deriving $P(\omega) = \frac{FFT(pes)}{FFT(u)}$ from the measurements of the position error signal $pes[n]$ and sampled control input $u[n]$ is risky because it may suffer from significant electrical disturbances. In order to minimize the effect of electrical noises and get unbiased estimation, it is recommended that the excitation source is directly measured by an input channel of SigLab [75]. Since there are only two input channels on SigLab 20-22, a two-step approach is adopted. As illustrated in Figure 3.3, when switch is in position 'a', the force

disturbance transfer function $F(\omega)$ is computed as

$$\widehat{F}(\omega) = \frac{P}{1+PC} = \frac{\widehat{\Phi}_{n_s pes}(\omega)}{\widehat{\Phi}_{n_s n_s}(\omega)} \tag{3.17}$$

where $\widehat{\Phi}_{n_s n_s}(\omega)$ is the auto power spectrum of the excitation signal $n_s[n]$ and $\widehat{\Phi}_{n_s pes}(\omega)$ is the cross power spectrum between n_s and the position error signal $pes[n]$. Similarly, when switch is in position 'b', the sensitivity transfer function $S(\omega)$ is computed as

$$\widehat{S}(\omega) = \frac{1}{1+PC} = \frac{\widehat{\Phi}_{n_s u}(\omega)}{\widehat{\Phi}_{n_s n_s}(\omega)} \tag{3.18}$$

where $\widehat{\Phi}_{n_s u}(\omega)$ is the cross power spectrum between the excitation signal $n_s[n]$ and the power amplifier input $u[n]$. Combining these two unbiased estimates, an unbiased estimate of the plant transfer function can be calculated as

$$\widehat{P}(\omega) = \frac{F(\omega)}{S(\omega)} = \frac{\widehat{\Phi}_{n_s pes}(\omega)}{\widehat{\Phi}_{n_s u}(\omega)} \tag{3.19}$$

Since this estimate is unbiased, it will converge to the actual transfer function with sufficient averaging. It takes approximately six minutes for the setup to give one estimate over full frequency range (10Hz~7000Hz) with 200 averages. The excitation amplitude increases with frequency to enhance the signal to noise ratio at high frequencies. Data are collected over inner diameter (ID), middle diameter (MD), and outer diameter (OD) respectively.

3.3.2 Broadband FFT

Since the swept-sine technique needs the added-on summing circuit and is time-consuming, it is impractical to apply this technique to every drive. In contrast, the broadband FFT method is more suitable for expeditious intact testing during production.

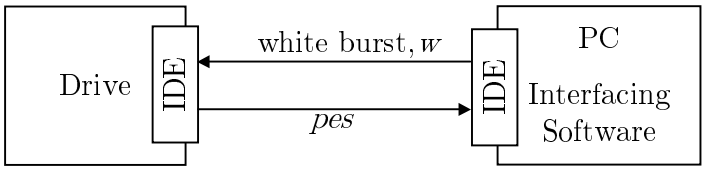

Figure 3.4: A intact system ID based on the broadband FFT technique.

Through the host interface (e.g. standard IDE port), a repeating pseudo-random burst is loaded into power amplifier by interfacing software and the responding position error signal *pes* is collected. The collected PES data is averaged along the position index to reduce the undesired measurement noise component, assuming such noise term is uncorrelated with the averaging process.

$$\widehat{P}(\omega) = \frac{FFT(pes[n])}{FFT(w[n])} \tag{3.20}$$

Since the white excitation stimulates all modes at the same time, the whole identification process only takes couple seconds and virtually requires no setup time. However, it is well known that the broadband method performs poorly in non-linear region [73], thus it is expected a low accuracy in low frequencies where the friction effect is dominating. Furthermore, the sampling rate of this method equals to the servo sampling rate and there is no anti-aliasing filter before sampling. Since the frequency components of some mechanical modes exceed the servo Nyquist frequency $f_N = \frac{f_s}{2}$, the aliasing effect in high frequencies also makes such method less accurate than the swept-sine technique. Figure 3.5 presents the result of identification experiments that are performed on the OD of the same drive by using two different methods. It clearly shows that the broadband FFT method is less accurate at both high (>4000Hz) and low (<200Hz) frequencies.

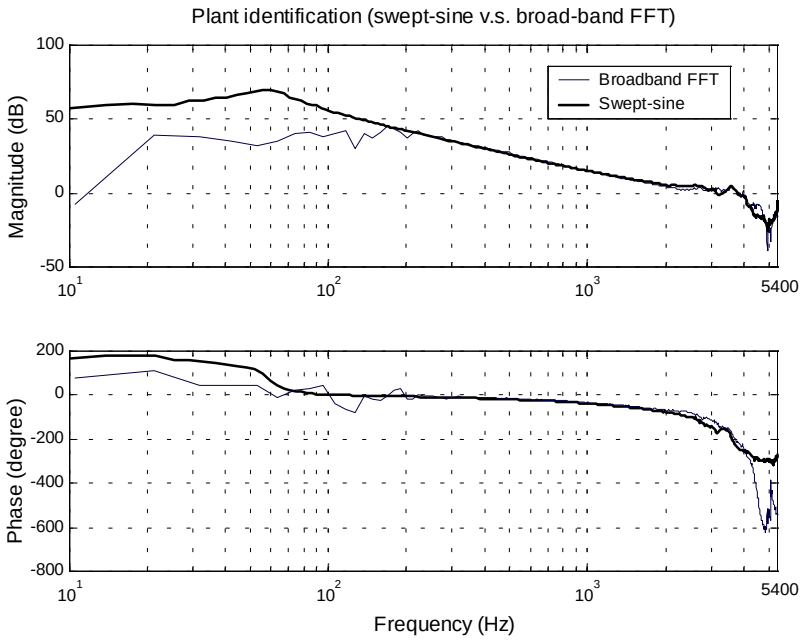

Figure 3.5: Identification of plant transfer function by the swept-sine technique and the broadband FFT method.

3.4 Nominal Linear Model

Figure 3.6 shows a simplified discrete-time model of HDD track-following servo system [10]. $C\left(z^{-1}\right)$ represents the digital controller. $P\left(z^{-1}\right)$ is the ZOH equivalence of continuous-time plant dynamics [66]. The sampling frequency is $f_s = \frac{RPM \cdot N_{sector}}{60}$. The sensitivity, co-sensitivity, and torque disturbance functions that are used throughout this dissertation are respectively defined as

$$S(z^{-1}) = \frac{1}{1 + P(z^{-1})C(z^{-1})}, \tag{3.21}$$

$$T(z^{-1}) = \frac{P(z^{-1})C(z^{-1})}{1 + P(z^{-1})C(z^{-1})}, \tag{3.22}$$

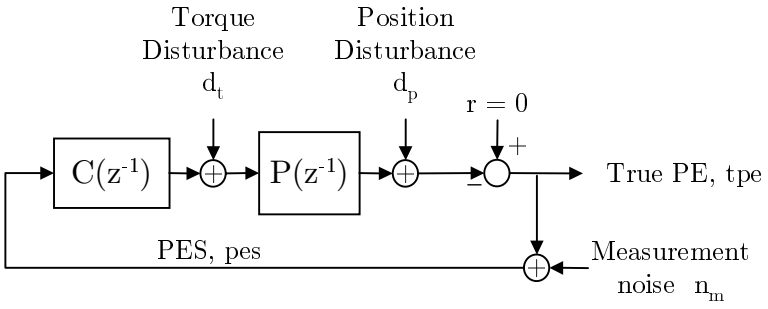

Figure 3.6: Discrete-time HDD servo model with lumped disturbances.

and

$$F(z^{-1}) = \frac{P(z^{-1})}{1 + P(z^{-1})C(z^{-1})}. \tag{3.23}$$

For convenience, disturbances and noises are lumped into three sources depending on their injecting points in the loop: torque disturbance d_t, position disturbance d_p and measurement noise n_m. Details regarding these PES contributors will be discussed in Section 3.5. This section synthesizes a linear model for the plant $P\left(z^{-1}\right)$ based on the transfer function measurements by the swept-sine technique and the prior knowledge of plant structure. The measurements from the broadband FFT method are used to validate the model.

3.4.1 Plant

A HDD servo plant consists of power amplifier (PA), voice coil motor (VCM) and head stack assembly (HSA). The desired plant model needs to achieve the best trade-off between the conflicting requirements of accuracy in representing the real system and low complexity to make a sensible control design possible.

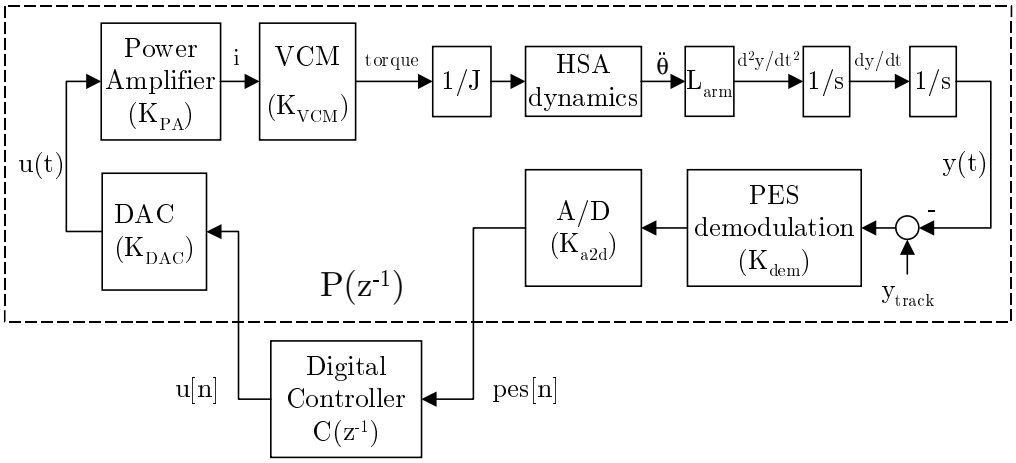

Figure 3.7: Simplified plant model for track-following.

PA is usually modeled as a low pass filter with a single pole at f_{PA},

$$P_{PA}(s) = K_{PA}\frac{f_{PA}\cdot 2\pi}{s + f_{PA}\cdot 2\pi}. \tag{3.24}$$

This dissertation is focused on the track-following control of HDD, in which PA operates in its linear range without saturation. Furthermore, the bandwidth of PA is much higher (e.g., $f_{PA} = 11$KHz in the experimental Maxtor drives) than the servo bandwidth which is usually less than 1KHz. The PA dynamics therefore has little effect on the stability margins. Thus, the PA dynamics is often simplified as a constant gain K_{PA}.

VCM dynamics is modeled as

$$P_{VCM}(s) = K_t\frac{\frac{R}{L}}{s + \frac{R}{L}} = K_t P_{VCM_o}(s) \tag{3.25}$$

where L is the coil inductance, R is the VCM loop resistance including the coil resistance and the current sensor resistance providing the current information.

Table 3.1: Plant parameters

Parameter description	**Symbol**	**Value**
Spindle speed	RPM	$5,400\ rev/minute$
Number of sectors	N_{sector}	120
Sampling rate	F_s	$10,800\ Hz$
Number of disks	N_{disk}	4
Number of heads	N_{head}	8
Track density	TPI	$20,000\ track/inch$
Arm length from pivot to the head gap	L_{arm}	$2.1343\ inch$
VCM torque constant	K_t	$0.096\ N \cdot m/Amp$
VCM coil inductance	L	$2.0 \times 10^{-6}\ H$
VCM loop resistance	R	$18.35\ \Omega$
Actuator inertia	J	$5.25 \times 10^{-6}\ Kg \cdot m^2$
Actuator imbalance	I_{imb}	$0.2 \times 10^{-5}\ kgm^2$
PA gain for track-following	K_{PA}	$0.15\ Amp/Volt$
PES demodulation gain	K_{dem}	$2\ Volt/Track$
ADC resolution	K_{A2D}	$1024\ bit/Volt$
DAC Voltage resolution	K_{DAC}	$4/1024\ Volt/bit$
Computational time delay	T_d	$24\ \mu s$

The overall open-loop gain is

$$K_o = K_{DAC} \cdot K_{PA} \cdot \frac{K_t}{J} \cdot L_{arm} \cdot TPI \cdot K_{dem} \cdot K_{A2D} \tag{3.26}$$

where values of the experimental drives used in this dissertation are shown in Table 3.1 and Table 3.2.

The ideal actuator would behave like double integrator due to the rigid bode mode of the arm, while the actual HSA dynamics exhibits friction at low frequencies and resonances at high frequencies.

Friction is a complicated nonlinear phenomenon. To make a linear design possible, it is approximated by a second order resonance mode

$$P_F(s) = \frac{1}{s^2 + 2\zeta_F(2\pi f_F) \cdot s + (2\pi f_F)^2} \tag{3.27}$$

Table 3.2: HSA modes

Parameter description	**Symbol**	**Value**
Pivot friction approximation frequency	f_F	$40Hz$
Pivot friction approximation damping ratio	ζ_F	0.15
Suspension T1 mode frequency	f_T	2700 Hz
Suspension T1 mode half-width	$f_{w/2}$	$4Hz$
Suspension T1 mode damping ratio	ζ_T	0.007
Actuator butterfly mode frequency	f_L	3950 Hz
Actuator butterfly mode damping ratio	ζ_L	0.06

Major mechanical resonances include suspension torsional modes and actuator lateral modes. Both can be modeled as second-order systems. A suspension torsional mode is usually described as

$$P_T(s) = \frac{s^2 + 2\zeta_T[2\pi(f_T + f_{w/2})] \cdot s + [2\pi(f_T + f_{w/2})]^2}{s^2 + 2\zeta_T[2\pi(f_T - f_{w/2})] \cdot s + [2\pi(f_T - f_{w/2})]^2} \cdot \frac{[2\pi(f_T - f_{w/2})]^2}{[2\pi(f_T - f_{w/2})]^2} \tag{3.28}$$

where f_T is the center frequency of torsional mode and $f_{w/2}$ represents half the bandwidth of mode. A actuator lateral mode can be described as

$$P_L(s) = \frac{(2\pi f_L)^2}{s^2 + 2\zeta_L(2\pi f_L) \cdot s + (2\pi f_L)^2} \tag{3.29}$$

where ζ_L is the damping ratio and f_L is the center frequency of lateral mode.

It is important to include the time delay in HDD modeling because time delay always reduces the phase stability of a system. The time delay in HDD servo is mainly from the computational overhead of digital signal processor. The nonrational time delay term $e^{-T_d s}$ can be approximated by the first order Pade approximation [66] as

$$P_d(s) = e^{-T_d s} = \frac{-0.5T_d \cdot s + 1}{0.5T_d \cdot s + 1}. \tag{3.30}$$

The overall plant transfer function in continuous-time domain is

$$P(s) = K_o \cdot P_{VCM_o}(s) \cdot P_F(s) \cdot P_T(s) \cdot P_L(s) \cdot P_d(s) \tag{3.31}$$

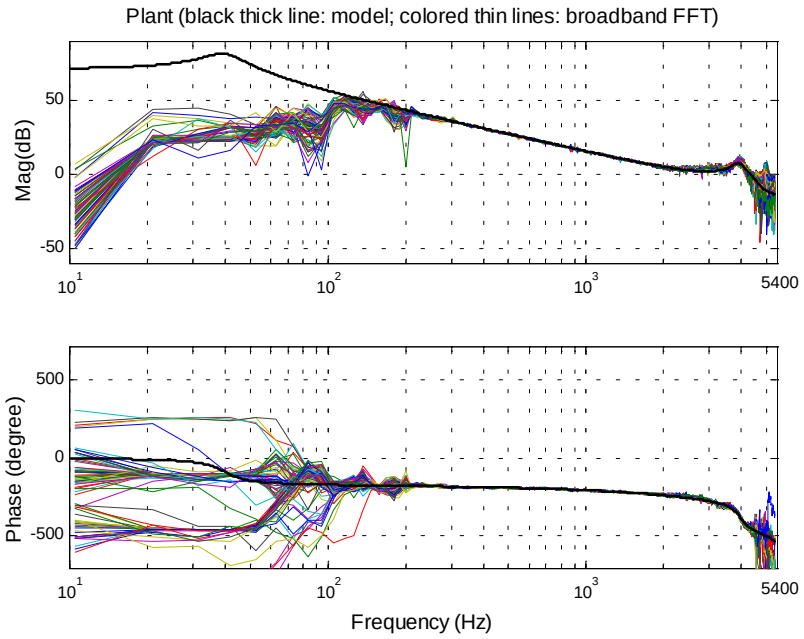

Figure 3.8: Plant model validation through applying broad-band FFT technique to a number of drives.

The ZOH equivalent form is calculated as

$$P(z^{-1}) = (1 - z^{-1})\mathbb{Z}[\mathbb{L}^{-1}[\frac{P(s)}{s}]] \tag{3.32}$$

where $\mathbb{Z}[.]$ denotes the z-transform action and $\mathbb{L}^{-1}[.]$ is the inverse Laplace transform action. The model is validated by comparing it with the broadband FFT measurement from a large number of drives, as shown in Figure 3.8.

3.4.2 Fixed-Structure Controller

The track-following controller $C(z^{-1})$ used in the experimental Maxtor drives is a fixed-structure compensator cascaded with a notch filter,

$$C(z^{-1}) = C_{pid}(z^{-1})C_{notch}(z^{-1}). \tag{3.33}$$

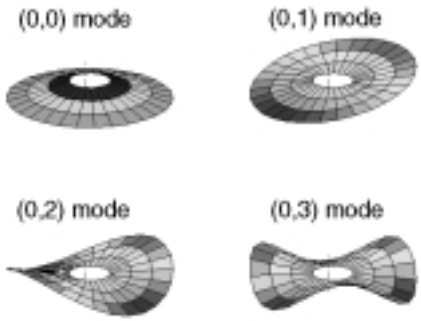

Figure 3.9: The four lowest modes, (0,0) at 608 Hz, (0,1) at 619 Hz, (0,2) at 729 Hz, and (0,3) at 1168 Hz are the most significant TMR contributors.

Compensator $C_{pid}(z^{-1})$ takes a PID-like form as

$$C_{pid}(z^{-1}) = k_{loop}[1 - (k_a + k_b)z^{-1} + k_a k_b z^{-2} + k_i \frac{1}{1 - z^{-1}}] \tag{3.34}$$

where $[k_{loop}, k_a, k_b, k_i]$ are tunable parameters. This structure is proven to have a good nominal performance in the last design cycle. The digital notch filter $C_{notch}(z^{-1})$ is added to attenuate the adverse effect due to the suspension resonance at 2700Hz.

3.5 Decomposition of Position Error Signals

The next step of building the statistical model is to identify the PES contributors from measured PES for every drive. For convenience, disturbances and noises are lumped into three sources depending on their injecting points in the loop: torque disturbance d_t, position disturbance d_p and measurement noise n_m, as shown in Figure 3.6.

- *Torque disturbance* d_t includes windage, pivot friction, flexible cable bias. Power amplifier noise and quantization errors from D/A and DSP [61] are also lumped in d_t

as they enter the loop as voltage disturbances. It has been shown that windage, power amplifier noise and quantization errors can be modeled as white noises [25] [26]. The closed-loop transfer function between d_t and pes is $F(z^{-1})$.

- *Position disturbance* d_p consists of servo written-in errors, disk modes, slider vibration and spindle motor vibration. Although disk modes consist of all possible (m,n) modes, the lowest four modes shown in Figure 3.9 are the most significant contributors to d_p. The sensitivity function $S(z^{-1})$ maps d_p to pes.

- *Measurement noise* n_m includes position sensing noise and quantization noise of A/D. Both can be modeled as white noises [25] [26]. n_m and pes are related by the sensitivity function $S(z^{-1})$.

In time-domain, the total relationship between disturbances and pes is

$$pes[n] = F(z^{-1})d_t[n] + S(z^{-1})(d_p[n] + n_m[n]) \tag{3.35}$$

In frequency-domain it is

$$\Phi_{pes}(f) = \|F(f)\|^2 \Phi_{d_t}(f) + \|S(f)\|^2 [\Phi_{d_p}(f) + \Phi_{n_m}(f)] \tag{3.36}$$

where Φ_{pes}, Φ_{d_t}, Φ_{n_m}, and Φ_{d_p} denote the power spectral density (PSD) of $pes(n)$, $d_t(n)$, $n_m(n)$ and $d_p(n)$ respectively. Note that the true position error tpe cannot be measured directly and position error signal (PES) is always contaminated by measurement noise n_m.

$$tpe[n] = F(z^{-1})d_t[n] + S(z^{-1})d_p[n] + T(z^{-1})n_m[n] \tag{3.37}$$

$$\Phi_{tpe}(f) = \|F(f)\|^2 \Phi_{d_t}(f) + \|S(f)\|^2 \Phi_{d_p}(f) + \|T(f)\|^2 \Phi_{n_m}(f) \tag{3.38}$$

The PES spectral decomposition and analysis method used in this dissertation is similar to those described in [25] [26] [27]. PES components can be categorized as repeatable run-out (RRO) which is synchronous with the spindle rotation and non-repeatable run-out (NRRO) which is not. Following procedure is used to separate RRO components and NRRO components from measured PES traces.

Algorithm 11 (Decomposition of RRO and NRRO)

1. Collect PES at a fixed cylinder for M revolutions. Assume each revolution covers N sectors: $pes[n]$, $n = 1, 2, ..., MN$

2. Average the PES over M revolutions to get the repeatable part:

$$\widehat{pes}_{RRO}[i] = (\frac{\sum_{j=1}^{M} pes[i + (j-1)N]}{M}), i = 1, ..., N \tag{3.39}$$

3. Extend $\widehat{pes}_{RRO}$ to the full length:

$$pes_{RRO}[n] = \widehat{pes}_{RRO}[\text{mod}(\frac{n}{N})], n = 1, ..., MN \tag{3.40}$$

4. Get NRRO by removing RRO from PES:

$$pes_{NRRO}[n] = pes[n] - pes_{RRO}[n], n = 1, ..., MN \tag{3.41}$$

5. Calculate the PSD of $pes_{RRO}[n]$ and $pes_{NRRO}[n]$, denoted as Φ_{pesRRO} and $\Phi_{pesNRRO}$ respectively.

It is well known that the pes is a white Gaussian random variable and the best way of quantifying it is to use $3\sigma_{pes}$. We are designing the controller for millions of drives,

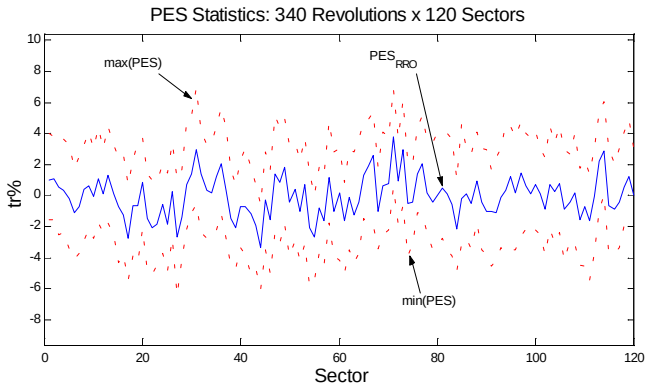

Figure 3.10: Averaging PES traces over multiple revolutions seperates the repeatable components from the non-repeatable parts.

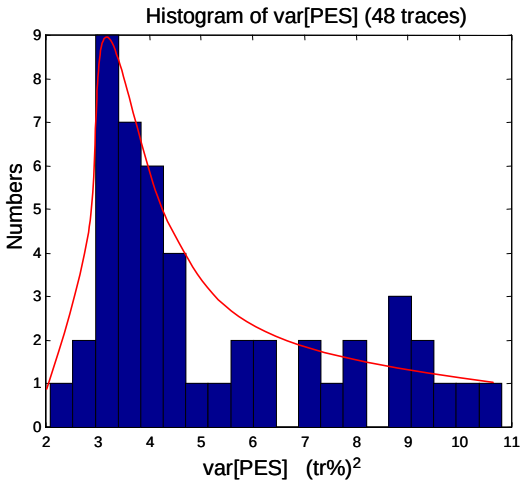

Figure 3.11: The variance of PES is not a constant but a random variable.

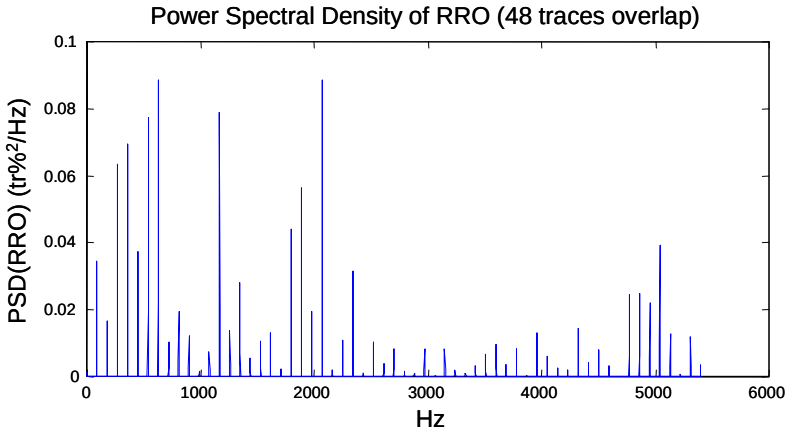

Figure 3.12: The PSD of RRO based on 48 PES traces.

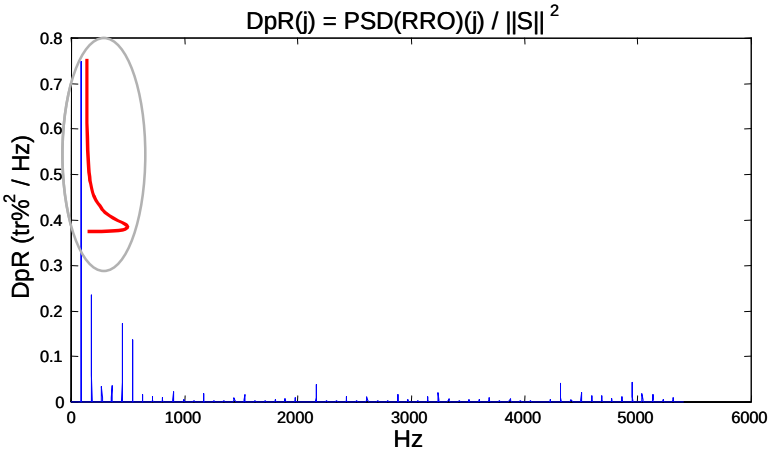

Figure 3.13: The PSD of repeatable position disturbance is recovered from PSD(RRO). The magnitude of bin has a Weibull disturbance.

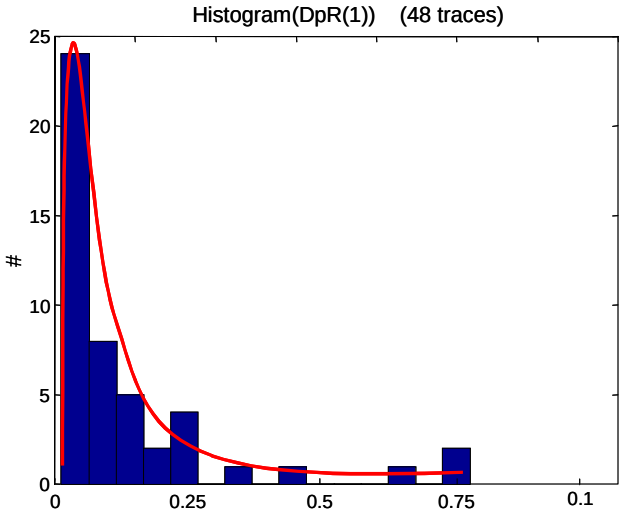

Figure 3.14: The random variable used to denote the magnitude of each RRO bin has a Weibull distribution.

therefore *the variance of pes* σ^2_{pes} *is not a constant but a random variable.* Figure 3.11 shows the histogram of 48 traces of PES collected from four identical drives. Since σ^2_{pes} has the unit of square of track percentage, it is expected that it has a Weibull distribution which is common for random variables with power units.

Decomposition of RRO

It is important to realize that the repeatable part of PES, pes_{RRO}, only comes from the position disturbance d_p [28]. For each drive, the PSD of repeatable components of d_p, denoted as $\Phi_{d_p RRO}$, is calculated by

$$\Phi_{d_p RRO}(f) = \Phi_{pesRRO}(f) \frac{1}{\|S(f)\|^2} \tag{3.42}$$

where the sensitivity function $S(f) = \frac{1}{1+P(f)C(f)}$ is not the measured one but the linear model, and f_n is the frequency of n-th RRO mode. Totally, there are $N_R = \frac{N_{sector}}{2}$ modes

at the multiplicities of the spindle frequency, where N_{sector} is the number of sectors. For example, Figure 3.12 shows that the experimental drives have 60 RRO modes. 48 traces are collected from four drives and measurements are repeated over the ID, MD, and OD of all disk surfaces. Figure 3.13 presents the overlapping Φ_{d_pRRO} recovered by using Eq. (3.42). Let a random variable $\mathbf{D}_{pR}(n)$ denote the power of n-th bin of Φ_{d_pRRO},

$$\mathbf{D}_{pR}[n] = \int_{BW_R[n]} \Phi_{d_pRRO}(f)df, \quad n = 1, ..., N_R \tag{3.43}$$

where $BW_R[n]$ is the bandwidth of n-th mode.

Repeat the procedure above for every drive. As shown in Figure 3.14 which is the histogram of $\mathbf{D}_{pR}(1)$, $\mathbf{D}_{pR}(n)$ has a Weibull distribution which is typical for a power measurement of a Gaussian random variable.

Decomposition of NRRO

$\Phi_{pesNRRO}$ can be divided into narrow-band and broadband components. Narrow-band components are mainly contributed by NRRO components of the position disturbance d_p, like disk modes and spindle modes. Broadband (baseline) noises are mainly contributed by the measurement noise n_m, the torque disturbance d_t, and some broadband noises from d_p. Broadband noises can be separated from $\Phi_{pesNRRO}$ by using root mean square (RMS) average. This smoothed version of $\Phi_{pesNRRO}$ is denoted as $RMS(\Phi_{pesNRRO})$. Figure 3.15 shows a typical relationship between $RMS(\Phi_{pesNRRO})$ and its contributors.

Modeling n_m: As shown in [27], the measurement noise n_m is white. It affects PES through sensitivity function $S(f)$. The power level of n_m can be estimated from $RMS(\Phi_{pesNRRO})$ as follows. At high frequencies (above 4 kHz), $S(f)$ has a unit gain and $F(f)$ rolls off

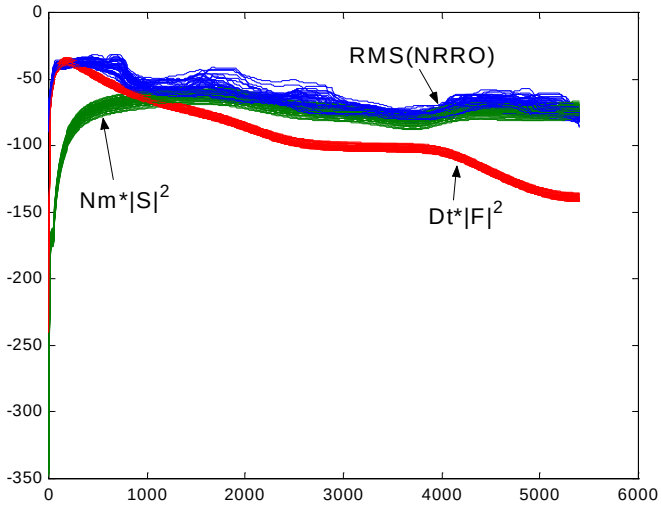

Figure 3.15: Both measurement noise n_m and torque disturbanace d_t are assumed to be white noises. The fittings to the baseline of NRRO are colored versions of N_m and D_t.

quickly. Based on Eq. (3.35) and (3.36), it is expected that the high frequency contents of $RMS(\Phi_{pesNRRO})$ are mainly contributed by the measurement noise n_m and the broadband part of d_p. A further component level measurement proves that the measurement noise n_m dominates $RMS(\Phi_{pesNRRO})$ at high frequency. Since $\|S(f)\|^2 = 1$ near the Nyquist frequency, the power level of n_m is simply the power spectral magnitude of $RMS(\Phi_{pesNRRO})$ at high frequency. Let a random variable $\mathbf{N}_m$ denote this power level. After identifying $\mathbf{N}_m$ for every drive, the mean value and variance of $\mathbf{N}_m$ are calculated. It is expected that $\mathbf{N}_m$ also has a Weibull distribution.

Modeling d_t: To identify the torque disturbance d_t, $\|S\|^2 \cdot \mathbf{N}_m$ is subtracted from RMS ($\Phi_{pesNRRO}$), i.e.,

$$\begin{aligned} & RMS(\Phi_{pesNRRO\_T}) \\ & = RMS(\Phi_{pesNRRO}) - \|S\|^2 \cdot \mathbf{N}_m \end{aligned} \tag{3.44}$$

where $RMS(\Phi_{pesNRRO\_T})$ is solely contributed by the torque disturbance d_t and some unaccounted TMR sources. Among the components of d_t, on one side, windage, power amplifier noise, and quantization errors can be modeled as white noises [25] [26]. On the other hand, due to the nonlinear nature of pivot friction and flexible print cable force, it is impossible to use a linear model to completely describe the torque disturbance d_t. However, since our overall objective is to do the variance projection of PES, it is reasonable to model d_t as white noise with $\text{PSD}(d_t) = \mathbf{D}_t$. For each drive, $\mathbf{D}_t$ is picked up such that the model has an *equivalent* TMR contribution as the measured data, more precisely,

$$\mathbf{D}_t = \frac{\int_0^{\frac{f_s}{2}} RMS(\Phi_{pesNRRO\_T}) df}{\int_0^{\frac{f_s}{2}} \|F(f_k)\|^2 \, df}. \tag{3.45}$$

Another benefit of using this model is that the unaccounted TMR sources are also automatically included. Totally, the power level $\mathbf{D}_t$ is a random variable with a Weibull distribution.

Modeling narrow-band NRRO components: Narrow-band components are identified to be contributed by NRRO components of the position disturbance d_p, like disk modes and spindle modes. Since those modes are not periodic, a 'bin' method is used to model the narrow-band components. Bins are defined according to the center frequencies and bandwidth of disk modes and spindle modes, which are normally known. Assuming there are total N_{NR} of such significant modes, $\mathbf{D}_{pNR}[n]$ is used to denote the power level for n-th

mode

$$\mathbf{D}_{pNR}[n] = \int_{BW_{NR}[n]} \frac{\Phi_{pesNRRO}(f) - RMS(\Phi_{pesNRRO})(f)}{\|S(f)\|^2} df, \quad n = 1, ..., N_{NR} \tag{3.46}$$

where $BW_{NR}[n]$ is the bandwidth of n-th mode.

3.6 TMR synthesis

The major design objective of servo system is to reduce the true PE, *tpe*. However, *tpe* is not directly measurable and we have to predict it based on measured *pes*. Actually, *pes* itself is a zero mean Gaussian process. It is impossible to synthesize *pes* and *tpe* completely in time-domain. However, TMR projection needs only the variance of *tpe*. Based on Eq. (2.2a-d) and Eq. (2.1a-d) and the fact that pes_{RRO} and pes_{NRRO} are independent [10], we have

$$\sigma^2_{pes} = \sigma^2_{pes_{RRO}} + \sigma^2_{pes_{NRRO}}. \tag{3.47}$$

and thus

$$\begin{aligned} \sigma^2_{tpe} &= \sigma^2_{pes} - \sigma^2_{n_m} \\ &= \sigma^2_{pes_{RRO}} + \sigma^2_{pes_{NRRO}} - \sigma^2_{n_m}. \end{aligned} \tag{3.48}$$

The variance of pes_{RRO}, pes_{NRRO}, and n_m are

$$\sigma^2_{pes_{RRO}} = \sum_{i=1}^{N_R} \mathbf{D}_{pR}[i] \left\|S(f_{RROi})\right\|^2, \tag{3.49}$$

$$\begin{aligned} \sigma^2_{pes_{NRRO}} = \mathbf{N}_m \int_0^{\frac{f_s}{2}} \|S(f)\|^2 df + \mathbf{D}_t \int_0^{\frac{f_s}{2}} \|F(f)\|^2 df \\ + \sum_{j=1}^{N_{NR}} \mathbf{D}_{pNR}[j] \left\|S(f_{NRROj})\right\|^2, \end{aligned} \tag{3.50}$$

$$\sigma^2_{n_m} = \int_0^{\frac{f_s}{2}} \mathbf{N}_m df. \tag{3.51}$$

where f_{RROi} is the center frequency of i-th RRO mode and f_{NRROj} is the center frequency of j-th NRRO narrow band mode. The random variable set $\{\mathbf{D}_{pR}(1)$, ..., $\mathbf{D}_{pR}(N_R)$, $\mathbf{N}_m$, $\mathbf{D}_t$, $\mathbf{D}_{pNR}(1)$, ..., $\mathbf{D}_{pNR}(N_{NR})\}$ fully describe the characteristics of the disturbances in track-following mode for a family of drives. Note that both $S = \frac{1}{1+PC}$ and $F = \frac{P}{1+PC}$ are functions of controller $C(z^{-1})$. This implies a direct method to tune the parameters of $C(z^{-1})$ to minimize σ^2_{tpe}. Since we are designing the controller for mass produced HDDs with plant variations and disturbance uncertainties, it is important to realize that σ^2_{tpe} is not a constant but rather a random variable. It is thus more appropriate to assess the average performance which is quantified by the expected value of σ^2_{tpe}, and the performance robustness which is quantified by the standard deviation of σ^2_{tpe}. According to (3.49), (3.50) and (3.51), the mean values of σ^2_{pes} and σ^2_{tpe} respectively are

$$\begin{aligned} E(\sigma^2_{pes}) = &\sum_{i=1}^{N_R} E(\mathbf{D}_{pR}(i)) \left\|S(f_i)\right\|^2 + E(\mathbf{N}_m) \int_f \left\|S(f)\right\|^2 \\ &+ E(\mathbf{D}_t) \int_f \left\|F(f)\right\|^2 + \sum_{j=1}^{N_{NR}} E(\mathbf{D}_{pNR}(j)) \left\|F(f_j)\right\|^2 \end{aligned} \tag{3.52}$$

and

$$\begin{aligned} E(\sigma^2_{tpe}) = &\sum_{i=1}^{N_R} E(\mathbf{D}_{pR}(i)) \left\|S(f_i)\right\|^2 + E(\mathbf{N}_m) \int_f \left\|S(f)\right\|^2 \\ &+ E(\mathbf{D}_t) \int_f \left\|F(f)\right\|^2 + \sum_{j=1}^{N_{NR}} E(\mathbf{D}_{pNR}(j)) \left\|F(f_j)\right\|^2 + E(\mathbf{N}_m)\frac{f_s}{2}. \end{aligned} \tag{3.53}$$

Before deriving the variance of σ^2_{pes} and σ^2_{tpe}, let $\mathbf{cov}(\mathbf{v}_1, \mathbf{v}_2) \in \Re^{n_1 \times n_2}$ denotes the covariance matrix for two random variable vectors $\mathbf{v}_1 \in \Re^{n_1 \times 1}$ and $\mathbf{v}_2 \in \Re^{n_2 \times 1}$ such that

$\mathbf{cov}(\mathbf{v}_1, \mathbf{v}_2)_{ij} = cov[\mathbf{v}_1(i), \mathbf{v}_2(j)]$. Then the variances of σ^2_{pes} and σ^2_{tpe} are given as

$$var(\sigma^2_{pes}) = var(\sigma^2_{pes_{RRO}}) + var(\sigma^2_{pes_{NRRO}}) + 2 \cdot cov(\sigma^2_{pes_{RRO}}, \sigma^2_{pes_{NRRO}}) \tag{3.54}$$

and

$$\begin{aligned} var(\sigma^2_{tpe}) &= var(\sigma^2_{pes_{RRO}}) + var(\sigma^2_{pes_{NRRO}}) - var(\sigma^2_{n_m}) \\ &+ 2 \cdot cov(\sigma^2_{pes_{RRO}}, \sigma^2_{pes_{NRRO}}) - 2 \cdot cov(\sigma^2_{pes_{RRO}}, \sigma^2_{n_m}) - 2 \cdot cov(\sigma^2_{pes_{NRRO}}, \sigma^2_{n_m}) \end{aligned} \tag{3.55}$$

where

$$\begin{aligned} var(\sigma^2_{pes_{RRO}}) &= E\left[\left(\sigma^2_{pes_{RRO}} - E(\sigma^2_{pes_{RRO}})\right)^2\right] \\ &= E\left[\left(\sum_{i=1}^{N_R} \mathbf{D}_{pR}(i) \left\|S(f_{NRROj})\right\|^2 - \sum_{j=1}^{N_R} E[\mathbf{D}_{pR}(i)] \left\|S(f_{NRROj})\right\|^2\right)^2\right] \\ &= \left[\left\|S(f_1)\right\|^2 ... \left\|S(f_{N_R})\right\|^2\right] \mathbf{cov}(\mathbf{D}_{pR}, \mathbf{D}_{pR}) \begin{bmatrix} \left\|S(f_1)\right\|^2 \\ ... \\ \left\|S(f_{N_R})\right\|^2 \end{bmatrix}, \end{aligned} \tag{3.56}$$

$$\begin{aligned} var(\sigma^2_{pes_{NRRO}}) &= E\left[\left(\sigma^2_{NRRO} - E(\sigma^2_{NRRO})\right)^2\right] \\ &= var(\mathbf{N}_m)\left[\int_0^{\frac{f_s}{2}} \left\|S(f)\right\|^2\right]^2 + var(\mathbf{D}_t)\left[\int_0^{\frac{f_s}{2}} \left\|F(f)\right\|^2\right]^2 + \\ &\left[\left\|S(f_1)\right\|^2 ... \left\|S(f_{N_{NR}})\right\|^2\right] \mathbf{cov}(\mathbf{D}_{pNR}, \mathbf{D}_{pNR}) \begin{bmatrix} \left\|S(f_1)\right\|^2 \\ ... \\ \left\|S(f_{N_{NR}})\right\|^2 \end{bmatrix}, \end{aligned} \tag{3.57}$$

and

$$\begin{aligned} &cov(\sigma^2_{pes_{RRO}}, \sigma^2_{pes_{NRRO}}) = \\ &\left[\left\|S(f_1)\right\|^2 ... \left\|S(f_{N_R})\right\|^2\right] \mathbf{cov}(\mathbf{D}_{pR}, \mathbf{D}_{pNR}) \begin{bmatrix} \left\|S(f_1)\right\|^2 \\ ... \\ \left\|S(f_{N_{NR}})\right\|^2 \end{bmatrix} \\ &+ [\mathbf{cov}(\mathbf{D}_{pR}, \mathbf{N}_m) + \mathbf{cov}(\mathbf{D}_{pNR}, \mathbf{D}_t)] \begin{bmatrix} \left\|S(f_1)\right\|^2 \\ ... \\ \left\|S(f_{N_R})\right\|^2 \end{bmatrix} \int_0^{\frac{f_s}{2}} \left\|S(f)\right\|^2, \end{aligned} \tag{3.58}$$

$$cov(\sigma^2_{pes_{RRO}}, \sigma^2_{n_m}) = \mathbf{cov}(\mathbf{D}_{pR}, \mathbf{N}_m) \begin{bmatrix} \|S(f_1)\|^2 \\ ... \\ \|S(f_{N_R})\|^2 \end{bmatrix} \frac{f_s}{2}, \tag{3.59}$$

$$\begin{aligned} cov(\sigma^2_{pes_{NRRO}}, \sigma^2_{n_m}) &= var(\mathbf{N}_m)\frac{f_s}{2}\int_0^{\frac{f_s}{2}} \|S(f)\|^2 \\ &+ cov(\mathbf{D}_t, \mathbf{N}_m)\frac{f_s}{2}\int_0^{\frac{f_s}{2}} \|F(f)\|^2 \\ &+ \mathbf{cov}(\mathbf{D}_{pNR}, \mathbf{N}_m) \begin{bmatrix} \|S(f_1)\|^2 \\ ... \\ \|S(f_{N_{NR}})\|^2 \end{bmatrix} \frac{f_s}{2}. \end{aligned} \tag{3.60}$$

Although the expressions above seem to be messy, note that those mean values $E(\cdot)$, variances $var(\cdot)$ and covariance matrices $\mathbf{cov}(\cdot,\cdot)$ can be calculated based on the PES data measured from hundreds of drives and saved as *constant* matrices. For a given controller parameter set K, $S(f,K)$ and $F(f,K)$ can also be easily derived based on the aforementioned linear plant model. Thus one can predict the time-domain performance of any given controller in terms of $E(\sigma^2_{tpe})$ and $var(\sigma^2_{tpe})$ based on Eq.(3.53) and (3.55) respectively. The overall statistical model includes the linear plant model and the formulations for $E(\sigma^2_{tpe})$ and $var(\sigma^2_{tpe})$. It can be easily built into optimization loops as an evaluation function. Thus it is especially useful for the non-gradient based optimization algorithms, such as genetic algorithms which will be discussed in later chapters.

3.7 Experimental Verification

Before putting the model into work, we need to verify its accuracy in prediction and its computational efficiency. The experimental setup is shown in Figure 3.16. The accuracy

verification is a two-step process. First, the modeling accuracy is evaluated by comparing the measured and calculated results based on original drives without changing controller parameters. Secondly, the capability of the PES prediction is evaluated by applying a different set of controller parameters.

Recovery Accuracy Evaluation 96 traces of PES are measured from eight drives and statistical analysis is applied to get $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$. On the other hand, a statistical PES model is built by using the above procedure. The synthesized $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ are compared with the results from measured PES, as shown in Table 3.3. The difference between the synthesized and measured results of $std(\sigma^2_{pes})$ is bigger than that of $mean(\sigma^2_{pes})$ because it involves more complex recovery process and thus more accumulated errors. Even so, the difference is less than 3% which is a clear indication of accuracy for the recovery process detailed in Section 3.6.

PES Prediction Evaluation In this stage, the model is used to predict the performance of a group of drives when some parameters in the controller are changed. As an example, a new firmware with the track-following loop gain ($k^0_{loop} = 0.27$) increased by 6%, i.e. $k_{loop} = 1.06k^0_{loop} = 0.2808$, is loaded into these drives and a new set of PESs are collected. The new statistical results from measured PES and synthesized results by the model are listed in Table 3.4.

As a natural extension, we repeat the above process for nine different loop gains. Every time PESs are collected for a new controller, a new statistical model is built and the prediction for both $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ is performed when k_{loop} is in a range of

$[0.92k^0_{loop}, 1.08k^0_{loop}]$ and k_a, k_b, and k_i are fixed at nominal values. Looking at Figure 3.17, the results are very interesting. Nine pairs of continuous lines are the predictions by nine different statistical models. Being nicely agreeing with each other suggests that the model is insensitive to its building point. In other words, whether it is built from PESs associated with k^0_{loop} or $1.06k^0_{loop}$, it gives a repeatable prediction. Furthermore, the measured data matching the prediction indicates the accuracy over the range. Another interesting point is the trend they suggested. Both the measured data and model suggest that the k_{loop} which achieves the smallest $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ is actually around k^0_{loop}. This implies that the original controller is well optimized.

To further illustrate the usefulness of the statistical model, we apply it to predict the time-domain performance by varying the tunable parameters of the track-following controller, k_{loop}, k_a, and k_b in large ranges respectively. In Figure 3.18, k_{loop} is varied from 0 to 1 as k_a, and k_b are kept at nominal values. It can be seen that as functions of k_{loop}, $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ have two singular points around 0 and 0.42 respectively. Similarly, they have two singular points around 0.5 and 0.98 in terms of k_a (see Figure 3.19), and two at -0.3 and 0.75 in terms of k_b (see Figure 3.20). Actually, the tuning of fixed-structure controllers is a non-convex optimization problem, and often with discontinuities like the track-following controller shown here. Thus it is not surprising that gradient-based methods are easily trapped in local optimum or just unable to converge. Some non-gradient based methods are therefore preferred. This topic will be discussed in details in Chapter 4.

Computational Efficiency All simulations are performed in MATLAB on a AMD 1GHz PC with 256 MB memory. For a given controller, the total CPU-time for computing $E(\sigma^2_{tpe})$

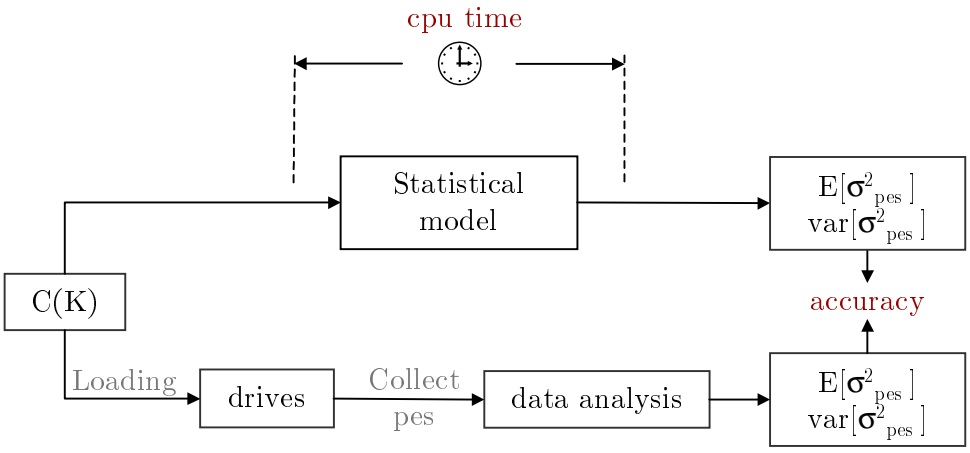

Figure 3.16: The setup for model validation.

Table 3.3: Accuracy evaluation of Modeling

	$mean(\sigma^2_{pes})$	$std(\sigma^2_{pes})$
Measured	$5.219(tr\%)^2$	$2.311(tr\%)^2$
Modeled (average)	$5.240(tr\%)^2$	$2.253(tr\%)^2$
Percentage difference	0.4%	-2.5%

and $var(\sigma^2_{tpe})$ based on Eq.(3.53) and (3.55) is always less than 0.06 second. This is in contrast to the 2-minute CPU-time achieved by a time-domain simulation tool [28], which even requires repetition if used to predict performance for multiple drives. Thus it is highly efficient to use the statistical model as an evaluation function inside the optimization loop.

Table 3.4: Evaluation of PES prediction capability ($k_{loop} = 1.06k^0_{loop}$)

	$mean(\sigma^2_{pes})$	$std(\sigma^2_{pes})$
Measured	$5.324\ (tr\%)^2$	$2.355\ (tr\%)^2$
Modeled (average)	$5.271\ (tr\%)^2$	$2.360\ (tr\%)^2$
Percentage difference	-1.0%	0.2%

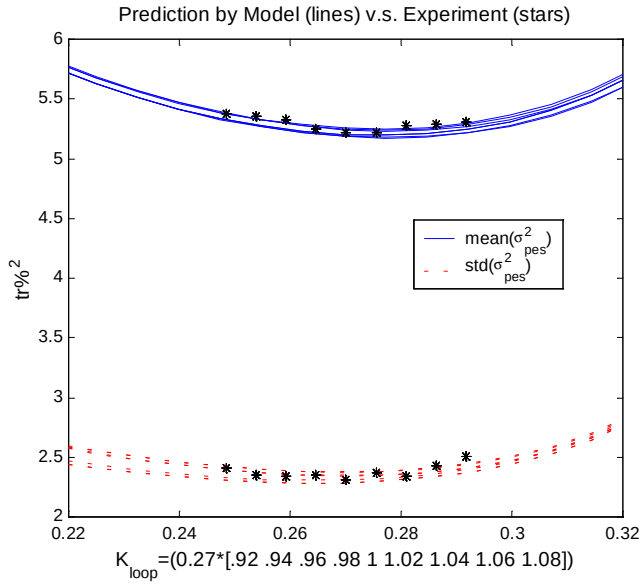

Figure 3.17: Nine models are built by repeating the building process on the same eight drives with nine different loop gains $k^0_{loop}\cdot$[0.92 0.94 0.96, 0.98, 1.00, 1.02, 1.04, 1.06, 1.08]. The trend predicted by models matches the experimental results.

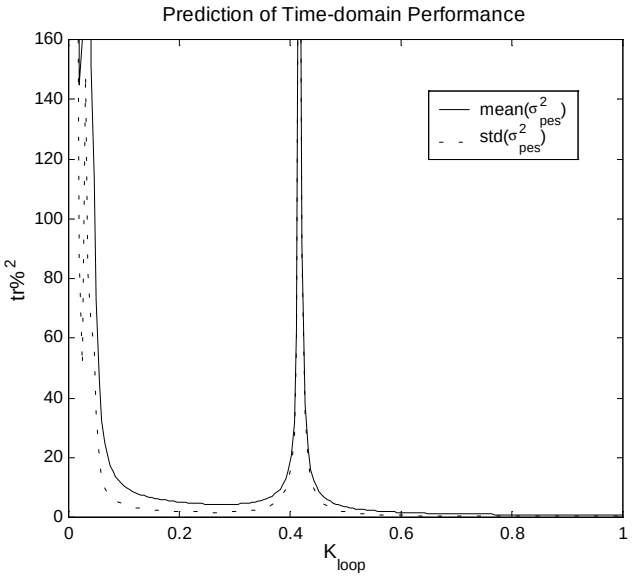

Figure 3.18: The predicted trend of $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ by varying k_{loop}.

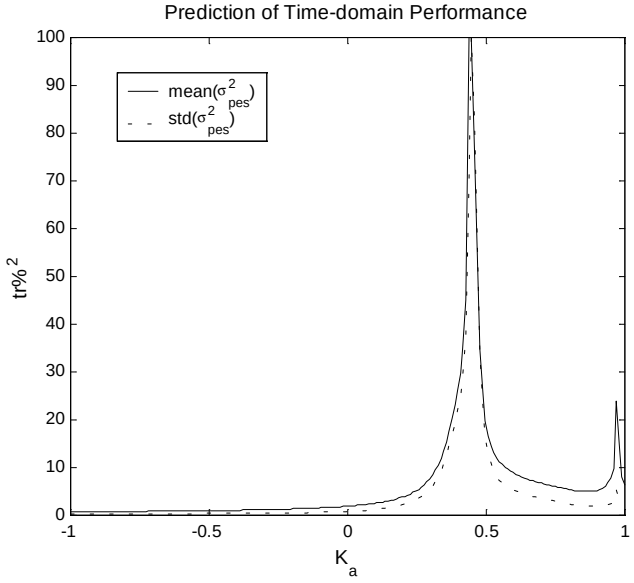

Figure 3.19: The predicted trend of $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ by varying k_a.

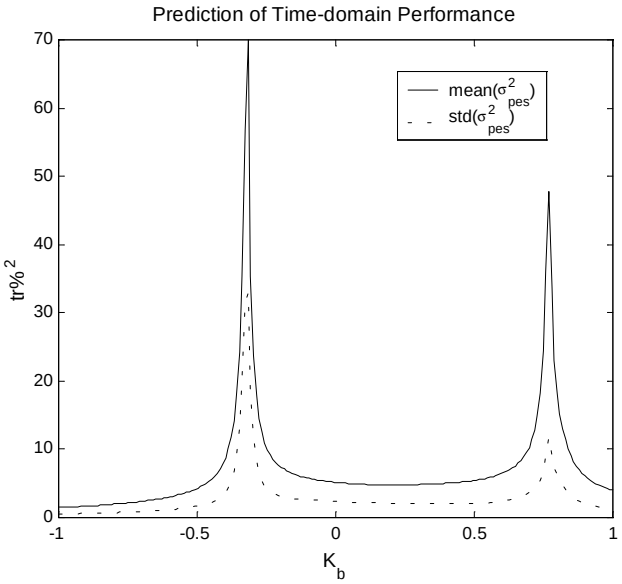

Figure 3.20: The predicted trend of $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ by varying k_b.

3.8 Summary

This chapter developed a statistical model for a population of HDDs. The overall formulation includes the linear plant model which was built by a multiple-phase identification process, and a set of random variables describing the PES contributors identified from a large number of drives. This model was proved to be efficient and accurate in predicting the time-domain performance in terms of $E(\sigma^2_{tpe})$ and $var(\sigma^2_{tpe})$ for a population of drives with a given controller. The model will be adopted by the non-gradient-based optimization tool in later chapters to tune the parameters of the track-following controller towards PES minimization. The methodology described here is also applicable to the tuning of controller parameters for other mass-produced electro-mechanical systems.

Chapter 4

Multi-objective Tuning of Fixed Structure Controller

4.1 Introduction

Modern control techniques like LQG/LTR, H_∞, H_2, and μ synthesis offer significant advantages over classical control methods in handling multiple control inputs and system uncertainties. On the down side, modern control methods usually lead to controllers with an order equal to or greater than the plant model used in design. Most literatures on the application of modern control techniques to HDDs showed a performance improvement by implementing the resulted high order controller in an add-on DSP [17] [13] [18] [76], while little emphasis was given to the comprehensive understanding of the plant to be controlled. This design philosophy has formed a gap between control theory and control practice. On one side, nice theory and laboratory results are there, but they are hard to be transferred

into control engineers who are working on industrial applications.

We use the servo design for HDDs as an example. The high-volume product nature of HDDs strongly constrains the complexity of the controller. The common approach to tackle this constraint is to first obtain a high-order controller which is then reduced to a manageable order for implementation by applying order reduction techniques on plant or controller. This not only limits the use of comprehensive models during design, but has no guarantee that the structure of the reduced-order controller is consistent from design to design. Due to the iterative nature of servo design, if changing the controller structure every design cycle, one is usually overwhelmed by difficulties associated with coding and version management. This makes the free-structure design prohibitive in the time-critical design cycle. Therefore, choosing a right controller structure is important not only in the design stage but also in the implementation stage. Readers may refer to the survey by Hanselmann [77] for a thorough discussion on this subject .

An alternative method is to put more efforts in building non-conservative plant model based on the fundamental understanding of plant and disturbances and then applying simply control laws. In fact, it is more efficient to start with a controller structure that has good nominal properties, and tune the parameters within that structure.

As discussed in the previous chapters, the design of the track-following controller involves numerous trade-offs and limitations. Although the frequency-domain properties provide strong analysis results, the real specifications and disturbances are in the time-domain. In fact, the combination of the variance of PES, hard bound on PES amplitude, and frequency-domain specifications is what is really needed to push the limits of performance.

A single performance objective is seldom adequate to capture multiple and often conflicting design objectives.

To solve such a multi-objective optimization problem (MOP), most optimization algorithms explicitly or implicitly require designer to assign weight to each objective before optimization. The problem is that the best combination of weights that leads to the desired performance is usually not known in advance. The designer needs to heuristically tune the weights until satisfactory performance is achieved.

In this chapter, the multi-objective tuning of fixed-structure controller is formulated as a goal attainment programming (GAP) problem [40]. Instead of finding feasible solutions which minimize or maximize objective functions, the GAP is intended to find solutions that, if possible, achieve a set of goals; otherwise, violate the goals minimally. GAP is conventionally solved by non-linear programming which still needs weights. This dissertation proposes a multi-objective genetic algorithm that is capable of parallelly searching for a set of Pareto optimal solutions without using weights before optimization.

This chapter will resolve the following two basic questions in order to tune the fixed-structure controller efficiently and reliably.

- How should the fixed structure control problem be formulated with multiple objectives and constraints?
- How should the formulated problem be solved or optimized?

From a point of view of decision making, however, for a given control engineering problem, we usually first want to know what optimization algorithms are available, and then we can formulate the problem in a solvable form accordingly. For example, a large

number of approaches focus on making the optimization a convex one through suitable parameterization. The motivation is that there are numbers of powerful algorithms for solving convex optimization problems. Therefore, the reminder of this chapter will answer the questions above in a reversed order. Section 4.2 gives some necessary preliminaries on optimization. Section 4.3 briefly reviews general optimization techniques. Section 4.4 considers the multi-objective optimization and introduces the concept of Pareto optimality. Section 4.5 discusses the parameterization of the fixed-structure control problem based on different optimization techniques.

4.2 Preliminaries

Definition 12 (Convex Set) *A set C is* convex *if the line segment between any two points in C lies in C, i.e., if for any x_1, $x_2 \in C$ and any θ that $0 \leqslant \theta \leqslant 1$, the following holds:*

$$\theta x_1 + (1-\theta)x_2 \in C. \tag{4.1}$$

Definition 13 (Convex Function and Concave Function) *A function f: $\Re^n \to \Re$ is* convex *if $\mathrm{dom} f$ is a convex set and if for any x, $y \in \mathrm{dom} f$ and any θ that $0 \leqslant \theta \leqslant 1$, the following inequality holds:*

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta)f(y). \tag{4.2}$$

A function f is called concave *if $-f$ is convex.*

Definition 14 (Local and Global Optimum) *For the following general optimization prob-*

lem,

$$\begin{aligned} \underset{x\in\Re^n}{\text{minimize}} \quad & f_0(x) \\ \textit{subject to} \quad & f_i(x) \leqslant 0 \quad (i = 1, ..., m) \\ & g_i(x) = 0 \quad (i = 1, ..., p) \end{aligned} \tag{4.3}$$

x is feasible if x satisfies all inequality and equality constraints. If x is feasible and

$$f_0(x) = \inf_{x\in\Re^n} \{ f_0(z) |\ \|x - z\| \leqslant R \} \tag{4.4}$$

holds for some feasible z and $\forall R > 0$, then x is locally optimal. *If (4.4) holds for any feasible z, then x is* globally optimal.

Definition 15 (Convex Optimization Problem) *The minimization problem of a convex function over the optimization variable $x \in \Re^n$ subject to inequality constraints on convex function of x and equality constraints on affine functions of x is a* convex optimization problem, *i.e.,*

$$\begin{aligned} \underset{x\in\Re^n}{\text{minimize}} \quad & f_0(x) \\ \textit{subject to} \quad & f_i(x) \leqslant 0 \quad (i = 1, ..., m) \\ & a_i^T x = b_i \quad (i = 1, ..., p) \end{aligned} \tag{4.5}$$

where $f_0, ..., f_m$ are convex functions. Let x be locally optimal *for the convex optimization problem (4.5). Then, x is also* global optimal *for (4.5).*

4.3 Overview of Optimization Techniques

No matter which parameterization framework is used, there is one fundamental issue that needs to be addressed: how should the global optimum be located efficiently

and reliably. The numerous search and optimization techniques can be classified into three categories: gradient-based algorithms, non-gradient-based deterministic schemes, and non-gradient-based stochastic algorithms.

4.3.1 Gradient-based Algorithms

Gradient-based techniques use numerical or analytic gradients information of target function to dictate a direction of search in which the optimum is thought to be. They can be classified into two groups: indirect and direct [42]. Indirect methods search local extreme by solving a set of usually nonlinear equations which result from setting the gradient of the objective functions equal to zero. On the other hand, direct methods seek local optima by jumping on the hyper-surface of target function and moving in a direction related to the local gradient. Both classes are local methods for they seek the optima only in a neighborhood of the current point.

General speaking, gradient-based methods are more efficient than other methods when the first derivative of the target function is continuous. Newton-type methods are only really suitable when the second derivative information can be easily calculated because calculating second derivatives, usually the Hessian matrix, by using numerical differentiation, is computationally expensive. Quasi-Newton algorithms developed by Dennis and Schnabel [78] avoid calculating Hessian directly by using the first derivative and function itself to build up curvature information to make an approximation to Hessian. A class of interior point methods have shown to be more efficient [79] [80]. They are based on Newton's method, to solve or approximately solve a sequence of smooth unconstrained or equality-constrained problems. Provided that the sequence of smooth problems is chosen appropriately, the

resulting algorithms are very efficient in both theory and practice.

However, gradient-based methods require the continuity and the existence of derivative of the target function and thus cannot provide sufficient *robustness* to find the global optimum for a non-convex optimization problem.

4.3.2 Non-gradient-based Deterministic Schemes

Non-gradient based methods use only evaluations of the objective function, not requiring its gradient or Hessian. Therefore they are most suitable for problems that are very nonlinear or have a number of discontinuities. The *enumerative search* is the simplest scheme in this class. Its idea is pretty simple; within a finite search space, or a discretized infinite search space, the search algorithm evaluates objective function values at every point in the space, one at a time. This scheme can arrive at reasonably good solutions for search spaces of small size. But when confronted with search spaces of enormous size and wide variation from point to point in their precinct, such schemes must ultimately be discounted in the robustness race for one simple reason: lack of efficiency. Even the *dynamic programming*, a highly touted enumerative scheme, may become exhausted on problems of moderate size and complexity, suffering from "the curse of dimensionality" [81]. Simplex search by Nelder and Mead [82] [83] is the most widely used non-gradient-based deterministic scheme. It implements extreme-point search by moving from one basic feasible solution to another until either the problem is shown to be unbounded or the current solution is proven to be optimal.

4.3.3 Non-gradient-based Stochastic Algorithms

Stochastic algorithms have been recognized to be able to overcome the shortcomings of gradient-based and deterministic schemes. The common feature of these methods is that they use stochastic rules to dictate the search. Random walk is the simplest version of stochastic optimization algorithms. Although it is capable of searching for the global optimum, it still lacks efficiency and does no better than the enumerative search. During the last three decades there has been a growing interest in algorithms that rely on analogies to the natural processes. The best-known examples are evolutionary algorithms (EAs), simulated annealing (SA), and neural networks (NN).

Particularly, the EAs refer to a class of such algorithms inspired by the process of natural evolution and heredity. The three mainstream versions of EAs are *genetic algorithms* (GAs), developed by Holland [84] [42], *evolutionary strategies* (ESs), developed by Rechenberg [85] [86] and Schwefel [87] [88], *evolutionary programing* (EP), developed by Fogel [89] [90]. EAs maintain a population of potential solutions, and have some selection processes based on fitness of individuals, and some genetic operators [43]. The difference and similarity of various EAs are summarized in Table 4.1. A comparative study of EAs can be found in [91] [92].

EAs were originally defined as stochastic search methods for scalar optimization problems of the following form:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \text{ with } f : \Omega \subseteq \Re^n \rightarrow \Re \tag{4.6}$$

where f is the fitness function, Ω is the feasible region, and n is the dimension of variables $\mathbf{x} = (x_1, ..., x_n)$ (and thus search space). The fitness function might be just the objective

Table 4.1: A summary of evolutionary algorithms.

	ES	EP	GA
Representation	Real coding	Real coding	Binary coding or real coding
Fitness	= Objective function	Scaled value of objective function	Scaled value of objective function
Mutation	Main operator	The only operator	Minor operator
Recombination	Different variants	None	Major operator Different variants
Selection	Deterministic extinctive	Probabilistic extinctive	Probabilistic Preservative
Self-adaptation	Standard deviations and covariances	Variances	None

function of optimization problem. Without loss of generality, assume $f(\mathbf{x}) \geqslant 0$ for any $\mathbf{x} \in \Omega$.

4.4 Multi-objective Optimization

Practical control design problems are often characterized by several non-commensurable and often competing measures of performance, or so-called objectives. A multi-objective optimization problem (MOP) can be posed as the minimization of a vector of objectives $\mathbf{f}(K) = \{f_1(K), ..., f_m(K)\}$ that may be the subject of a number of constraints or bounds.

$$\underset{\mathbf{K} \in \Re^n}{\text{minimize}} \quad \mathbf{f}(K) \tag{4.7}$$

$$\begin{aligned} \text{subject to} \quad & g_i(K) = 0, \ i = 1, ..., n_e \\ & g_i(K) \leqslant 0, \ i = n_e + 1, ..., n \\ & K_{LB} \preccurlyeq K \preccurlyeq K_{UB} \end{aligned} \tag{4.8}$$

If any of objectives are competing, there is no unique solution.

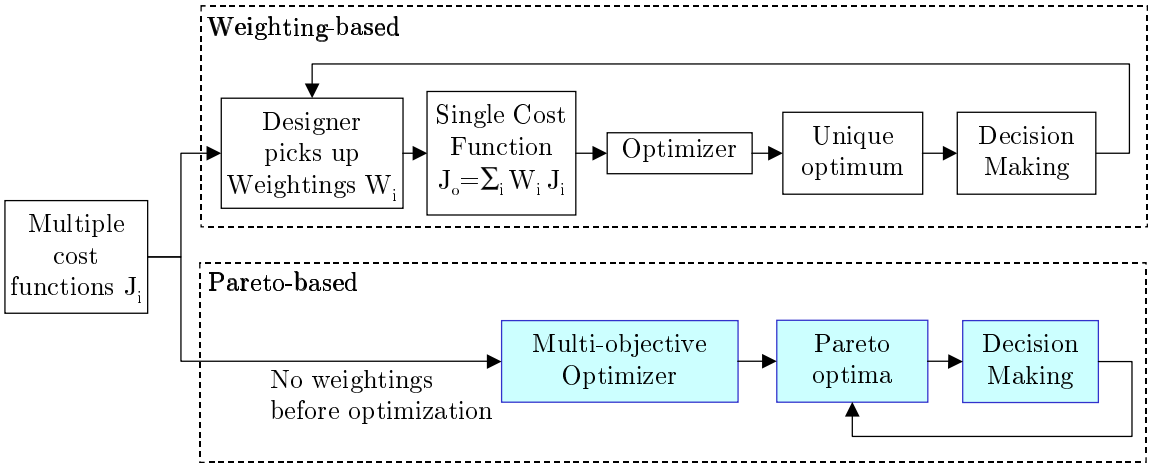

Figure 4.1: Different methods of dealing with multiobjective optimization problems. The upper part is the conventional weighting-based method and the lower part is the Pareto-based approach.

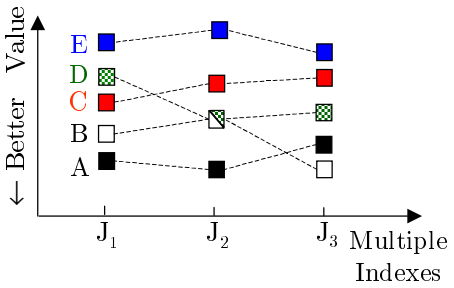

Figure 4.2: An example of Pareto optimality.

Pareto optimality, developed by Vilfredo Pareto (1848-1923, an Italian sociologist), is the most widely accepted criterion of economic efficiency. A state of a given system is Pareto optimal, and thus efficient, if and only if there is no feasible alternative state of that system in which at least one is better off and no one is worse off. And, for purposes of this criterion, a person is 'better off' with some alternative A rather than B if and only if this person *prefers* A to B [93]. The concept of Pareto optimality requires no weights *before* optimization and makes it possible to provide multiple optimal solutions to the decision maker.

Definition 16 (Domination) *The vector* $\mathbf{J}(K^1)$=$\{J_1(K^1), ..., J_n(K^1)\}$ *is said to* dominate *vector* $\mathbf{J}(K^2) = \{J_1(K^2), ..., J_n(K^2)\}$ *if and only if* $\mathbf{J}(K^1)$ *is partially less than* $\mathbf{J}(K^2)$, *denoted as* $\mathbf{J}(K^1) <_p \mathbf{J}(K^2)$, *more precisely*

$$(\forall i) J_i(K^1) \leqslant J_i(K^2) \wedge (\exists i) J_i(K^1) < J_i(K^2). \tag{4.9}$$

Definition 17 (Pareto Optimality) *A solution* K^1 *is* Pareto optimal *if and only if there is no* $K^2 \in \Omega_K$ *such that* $\mathbf{J}(K^2) <_p \mathbf{J}(K^1)$. *Pareto optimal solutions* $\mathbf{K}_p$ *are also called* non-dominated set *or* non-inferior set, *which are a set of* K^j *such that*

$$\begin{aligned} (\mathbf{J}(K^i) \quad &\not< \quad {}_p\mathbf{J}(K^j)) \wedge (\mathbf{J}(K^j) \not<_p \mathbf{J}(K^i)), \\ K^i \quad &\in \quad \mathbf{K}_p, K^j \in \mathbf{K}_p, \forall i \neq j. \end{aligned} \tag{4.10}$$

In other words, the Pareto set is optimal in the sense that no improvement can be achieved in any objective without degradation in others. An illustration example is shown in Figure 4.2, the solutions are evaluated based on three performance indexes J_1, J_2, and J_3 respectively. For each index, the smaller the value, the better the solution. It is easy to

see that solution A and B are better than all the other solutions in all aspects. However, A performs better than B in terms of J_1 and J_2 but worse than B in term of J_3. According to the definition of Pareto optimality, we say A and B are the equally best solutions and in the same Pareto set. Similarly, C and D are equally good solutions in the second best set. Both dominate solution E but are dominated by A and B. The overall Pareto ranking with respect to $\mathbf{J} = \{J_1, J_2, J_3\}$ can be expressed as

$$\mathbf{J}\{A, B\} <_p \mathbf{J}\{C, D\} <_p \mathbf{J}\{E\}. \tag{4.11}$$

The goal of MOPs is to search and identify noninferior solution points. Some of the major techniques to tackle MOPs are described below. Although there are different ways to classify these techniques, we divide them in two classes: weighting-based techniques which use weighting explicitly or implicitly, and Pareto-based methods which solve the MOP in a parallel way.

4.4.1 Weighted Sum Strategy

One way of dealing with multiple objectives is to combine them into a single objective by using weighting vector $\mathbf{w}=\{w_1, ..., w_m\}$ The MOP objective Eq. (4.7) becomes

$$\underset{K \in \Re^n}{\text{minimize}} \quad f_w = \sum_{i=1}^{m} w_i f_i(K) \ . \tag{4.12}$$

In most cases, however, the best combination of weightings is not known *in advance* due to the lack of knowledge of target problem. As shown in the upper part of Figure 4.1, the whole iterative process involves many try-and-errors. One has to play with weights and repeat optimization for many times before getting insight into the interaction among objectives. This situation will get worse with increasing number of competing objectives. Some

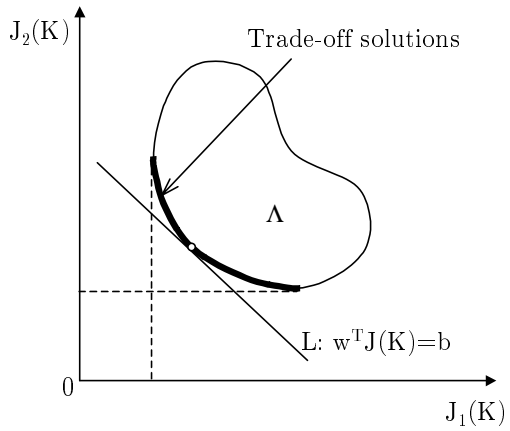

Figure 4.3: The convex trade-off boundary is accessible by varying weightings.

techniques such as loop-shaping use weightings implicitly. Therefore no matter whether we use LQG/LTR or H_∞ loop-shaping, the most difficult part is still the selection of the weighting functions or filters.

Furthermore, the trade-off solution boundary may be nonconcurrent so that certain solutions are not accessible [94]. This can be illustrated geometrically by the following example with two objectives. In the objective function space a line $L : wJ(K) = b$ is drawn. The minimization of b in domain Λ can be interpreted as finding the value of b with which L just touches the boundary of Λ as it proceeds outwards from the origin. Selection of weights $w = [w_1, w_2]$, therefore, defines the slope of L, which in turn leads to the solution point where L touches the boundary of Λ. For a Λ with convex lower boundary as shown in Figure 4.3, all optimal solutions are reachable by varying w. However, accessibility cannot be guaranteed for Λ with a non-convex lower boundary. For example, in Figure 4.4, all solutions between the point α and β are not accessible.

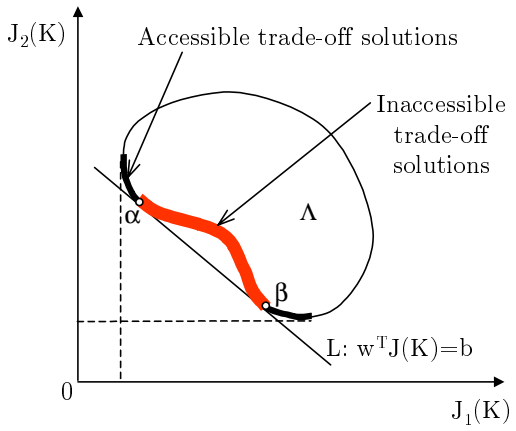

Figure 4.4: Some trade-off solutions on the nonconvex boundary are not accessible by varying w.

4.4.2 $\varepsilon-$Constraint Strategy

In the setup of $\varepsilon-$Constraint method, one primary objective J_p is minimized while the remaining objectives are converted into inequality constraints

$$\begin{aligned} \underset{\mathbf{K}\in\Re^n}{\text{minimize}} \quad & f_p(K) \\ \text{subject to} \quad & f_i(K) < \varepsilon_i,\ i = 1, ..., m,\ i \neq p \end{aligned} \tag{4.13}$$

along with the other constraints in MOP Eq. (4.8).

The ε-constraint method overcomes the inaccessible problems that the weighted sum strategy suffers from in non-convex optimization. However, it still has the same shortcoming as the weighted sum strategy that the designer must choose a suitable set of ε_i such that there is a feasible solution to the problem. In order to put the designer's true preferences into a mathematical description, one has to express a full table of their preferences and satisfaction levels for a range of objective value combinations [94]. Therefore, difficulties arise when the designer tries to prioritize the objectives that are equally important, especially at the early stage of optimization cycles.

4.4.3 Goal Attainment Programming (GAP) Strategy

The goal attainment programming (GAP) is a technique to find a compromised solution which will simultaneously satisfy a number of design objectives. GAP first appeared in the fifties to obtain "constrained regression" estimates for an executive compensation problem by Charnes and Cooper [40] [95]. In contrast to linear programming (LP), the GAP does not require that all constraints must be satisfied in order to have a feasible solution. Instead, the GAP requires the designer to assign a set of design goals, $\mathbf{f}^0=\{f_1^0, ..., f_m^0\}$, to the objectives $\mathbf{f}(x) =\{f_1(x), ..., f_m(x)\}$ and minimizes the deviation between the objectives and design goals and, if possible, attains the goals with zero deviation. The minimization process can be accomplished with different optimization techniques. Each one leads to a different GAP variant.

The classical GAPs as presented in [38] [39] [36] convert the multiple objectives (Eq.(4.7)) into a single objective optimization problem with additional constraints

$$\begin{aligned} &\underset{\gamma\in\Re,\mathbf{K}\in\Re^n}{\text{minimize}} \quad \gamma \\ &\text{such that} \quad f_i(K) - f_i^0 \leqslant w_i\gamma,\ i = 1, ..., m \end{aligned} \tag{4.14}$$

by introducing a slack variable γ and weighting vector $\mathbf{w}=\{w_1, ..., w_m\}$. This GAP formulation can be directly posed as a nonlinear programming problem such as sequential quadratic programming (SQP). A more complete description will be presented in Chapter 6. The GAP can also be posed as a minimax problem by using the following setup [96].

$$\begin{aligned} &\underset{\gamma\in\Re,\mathbf{K}\in\Re^n}{\text{minimize}} \quad \max_i\{\lambda_i\} \\ &\text{where} \qquad \lambda_i = \frac{f_i(\mathbf{K})-J_i^0}{w_i},\ i = 1, ..., m \end{aligned} \tag{4.15}$$

Other setups are also possible, such as lexicographic goal programming [96] [97].

All setups can take advantage of the existing single objective optimization algorithms, such as those in MATLAB Optimization Toolbox. A major drawback with above setups is that they still require the user to specify a set of weighting factors $\mathbf{w}$, signifying the relative importance of each criterion. This makes these approaches subjective to the designer. Moreover, these approaches have difficulty in finding solutions in problems having non-convex feasible solution space; they are still local methods.

Although using multi-objective optimization techniques to solve GAP has been previously presented in [98] [96], the works primarily focused on solving linear GAP due to the inefficiency of gradient-based nonlinear multi-objective optimization methods. The non-gradient-based GA was first introduced to solve general GAP by Kalyanmoy [97]. This dissertation parameterizes the parameter tuning problem for fixed structure controllers as a multi-objective GAP which is then solved by a Pareto-based multi-objective genetic algorithms (MOGA). The population-based nature of MOGA enables the evolution of a Pareto-optimal set of solutions without requiring weightings [99] [100] [49].

In order to convert the objective into a "goal" form, an user-defined target f_i^0 is needed. For example, $\min_K f_i(K)$ (Eq. (4.7)) can be converted to

$$\operatorname*{goal}_{K\in\Omega}(f_i(K) \leqslant f_i^0). \tag{4.16}$$

Note that f_i^0 is not necessarily achievable. If the minimum possible cost $f_i(K^*)$ is bigger than the target f_i^0, there exists no feasible solution which will achieve the target in Ω. In this case the objective of Eq. (4.16) is to find a solution that will minimize the deviation $d = f_i(K) - f_i^0$.

Handling constraints in GAP framework is very straightforward. Constraints are

Table 4.2: MOP and multi-objective GAP

Objectives and constraints of MOP	Objective functions of GAP
$\min\limits_{K} f_i(K)$	$\underset{K}{\text{goal}}(f_i(K) \leqslant f_i^0)$
$\max\limits_{K} f_i(K)$	$\underset{K}{\text{goal}}(f_i(K) \geqslant f_i^0)$
$g_i(K) \leqslant g_i^0$	$\underset{K}{\text{goal}}(g_i(K) \leqslant g_i^0)$
$g_i(K) \geqslant g_i^0$	$\underset{K}{\text{goal}}(g_i(K) \geqslant g_i^0)$
$g_i(K) = g_i^0$	$\underset{K}{\text{goal}}(g_i(K) = g_i^0)$
$g_i^{LB} \leqslant g_i(K) \leqslant g_i^{UB}$	$\underset{K}{\text{goal}}(g_i^{LB} \leqslant g_i(K) \leqslant g_i^{UB})$
$K_{LB} \preccurlyeq K \preccurlyeq K_{UB}$	$\Omega := \{K : K_{LB} \preccurlyeq K \preccurlyeq K_{UB}\}$

often seen as hard objectives, which need to be satisfied before the optimization of the remaining objectives takes place [101]. Without losing generality, constraints of Eq. (4.8) can expressed in the following type of inequality

$$g(K) \leqslant g^0 \tag{4.17}$$

where $g(K)$ is a real-valued function of the variable set K, and g^0 is a constant value. The inequality may also be strict ($<$ instead of $\leqslant$). Equality constraints of the type $g(K) = g^0$ can be formulated as a particular case of inequality constraints. According to the basic idea of GAP, constraint Eq. (4.17) is satisfied if

$$\underset{K \in \Omega}{\text{goal}}(g(K) \leqslant g^0)$$

is achieved, where Ω is user-defined search space.

Finally, the constrained MOP in Eq. (4.7) (4.8) is converted to an unconstrained MOP according to Table 4.2. *Note that goals are not subject to convex limitation of any kind.*

4.4.4 Linear Matrix Inequalities (LMIs)

Linear matrix equalities (LMIs) have received a great deal of attention since it provides a unifying framework for control design with multiple design specifications. If the state space description of a problem is available, many of the design objectives and constraints can be parameterized in convex LMI forms and then solved exactly and efficiently as semidefinite programmings (SDPs). See [102] [103] [104] for a collection of such problems.

To tackle the multi-objective control problem, the closed-loop system is affinely parameterized with the Youla parameterization (also called Q-parameterization). The desired closed-loop specifications are guaranteed by introducing auxiliary Lyapunov matrices into the problem. Because of the product terms between these Lyapunov variables and the state space matrices of Youla parameterization, the resulting MOP is a non-convex bilinear matrix inequality problem. It has been shown by Scherer [104] and Masubuchi [105] how this non-convex problem can be transformed into a convex feasibility problem by imposing the extra constraint that all the Lyapunov variables are equal, at the expense of conservatism for the overall design. Sub-optimal solutions are then computed by confining the search of the Youla parameterization to finite-dimensional subspaces. However, it still suffers from a substantial inflation of size of the resulting optimization problem if improving the approximation accuracy to the optimal solution. To avoid such a problem, [106] revealed a novel parameter transformation method to arrive at an efficient algorithm. The final MOP formulation as shown in [106] still adopted the weighted-sum strategy. Therefore the Pareto optimal controller was attained by varying the weights and repeating the SDP.

4.5 Parameterization of Fixed Structure Controller

Fixed structure control is a design method where the order and the structure (or parameterization) of controllers are pre-determined by the user and only some parameters are tunable.

Besides the classical methods for PID tuning, there are two dominant trends in dealing with the tuning of more general fixed-structure controllers. On one side, a large number of approaches focus on making the optimization problem a convex one, mostly by using some suitable parameterization. The motivation is that there are effective and powerful algorithms for solving the convex optimization problems. Examples of such algorithms are ellipsoid algorithms [107] and more efficient interior point methods for solving LMI-based problems [79]. The problem is that the parameterization itself usually introduces many auxiliary variables. For a multi-objective design problem, the final LMIs might be too large to solve [24]. On the other hand, non-gradient-based optimization techniques, such as genetic algorithms and simulated annealing, have drawn a great attention in directly solving the MOP in the original engineering setup, no matter it is convex form or not. This section discusses both techniques in great details.

4.5.1 Convex Parameterization

For a fixed structure controller with tunable parameters, the optimization problem cannot be directly parameterized as a convex optimization problem. It is usually recast into a static output feedback control problem which then has a number of well-developed optimization theories [108] [30] [31] [109] [110]. For example, Nett *et al.* [30] showed that

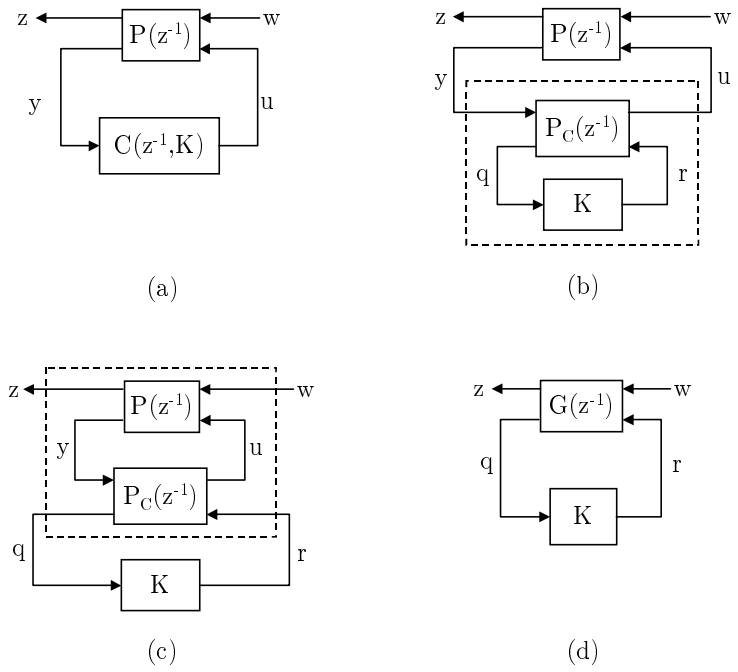

Figure 4.5: Reduction procedure (a) Dynamic output feedback control; (b) Decomposition; (c) Augmentation; (d) Static output feedback control.

nine cases of constrained-structure controllers, which cover most practical applications, can be recast as static output feedback controllers. The procedure is outlined as follows.

Decomposition of Controller Dynamics and Tunable Parameters

As depicted in Figure 4.5 (a) and (b), the first step of transformation is to separate the tunable parameters of controller with the controller dynamics such that $C(z^{-1}, K) = F_L(P_C(z^{-1}), K)$. The necessary and sufficient condition under which the decomposition exists is given in the following theorem [30].

Theorem 18 (Existence of Decomposition) *Suppose that the controller* $C(z^{-1}, K)$ *has*

a state space realization $\left[\begin{array}{c|c} A_C & B_C \\ \hline C_C & D_C \end{array}\right]$ *and is affine in* K*, then there exists* P_C *which satisfies*

$$C(z^{-1}, K) = F_L(P_C(z^{-1}), K) \tag{4.18}$$

if and only if there exists a solution to the following real matrix equation:

$$\begin{bmatrix} A_C & B_C \\ C_C & D_C \end{bmatrix} = \begin{bmatrix} P_{C_{11}} & P_{C_{12}} \\ P_{C_{21}} & P_{C_{22}} \end{bmatrix} + \begin{bmatrix} P_{C_{13}} \\ P_{C_{23}} \end{bmatrix} K \begin{bmatrix} P_{C_{31}} & P_{C_{32}} \end{bmatrix} \tag{4.19}$$

where $\mathbf{K}$ *is a full block of tunable parameters and the state space realization of* P_K *is*

$$\left[\begin{array}{c|cc} P_{C_{11}} & P_{C_{12}} & P_{C_{13}} \\ \hline P_{C_{21}} & P_{C_{22}} & P_{C_{23}} \\ P_{C_{31}} & P_{C_{32}} & 0 \end{array}\right]$$

Proof. See Nett *et al.*[30] or Ibaraki [19]. ■

Note that the assumption that $C(z^{-1}, K)$ is affine in K is satisfied in most applications of interest.

Augmentation of Controller Dynamics into Plant

The extended plant is constructed by absorbing the controller dynamics into the plant

$$G(z^{-1}) = F_L(P(z^{-1}), P_C(z^{-1})). \tag{4.20}$$

The the stability property of the transformation is stated by the following theorem.

Theorem 19 (Internal Stability of the Interconnection) *Suppose (4.18), (4.19), and (4.20) hold, then* $C(z^{-1}, K)$ *is internally stabilizing for* $P(z^{-1})$ *if and only if* K *is internally stabilizing for* $G(z^{-1})$*.*

Proof. See Nett *et al.* [30]. ■

Optimization of Static Output Feedback Controller

There are a large number of literatures on optimal static output feedback control [111] [22] [19]. The H_∞ optimization method is reviewed below.

H_∞ Optimal Static Output Feedback Control In contrast to the full-order controller synthesis case, the H_∞ synthesis problem of static output feedback or more generally reduced-order controller synthesis cannot be reparameterized as a convex optimization problem. As suggested by the following theorem [112], the order constraint destroys the convexity.

Theorem 20 (Solvability of Discrete-time H_∞ Controller Synthesis Problems)) *There exists a dynamic controller $C(s)$ of order k such that $\|F_L(P_K(s),C(s))\|_\infty < \gamma$ if and only if there exist two symmetric matrices $X \in \Re^{n\times n}$ and $Y \in \Re^{n\times n}$ such that*

$$N_1^T \begin{bmatrix} AXA^T - X & AXC_1^T & B_1 \\ C_1XA^T & -\gamma I + C_1XC_1^T & D_{11} \\ B_1^T & D_{11}^T & -\gamma I \end{bmatrix} N_1 \prec 0 \tag{4.21}$$

$$N_2^T \begin{bmatrix} A^TYA - Y & A^TYB_1 & C_1^T \\ B_1^TYA & -\gamma I + B_1^TYB_1 & D_{11}^T \\ C_1 & D_{11} & -\gamma I \end{bmatrix} N_2 \prec 0 \tag{4.22}$$

$$\begin{bmatrix} X & I \\ I & Y \end{bmatrix} \succeq 0 \tag{4.23}$$

$$rank(XY - I) \leqslant k \tag{4.24}$$

where $N_1 = diag\{N_{12}, I\}$, $N_2 = diag\{N_{21}, I\}$, N_{12} and N_{21} are bases of the null space of $\begin{bmatrix} B_2^T & D_{12}^T \end{bmatrix}$ and $\begin{bmatrix} C_2 & D_{21} \end{bmatrix}$, respectively.

In the full-order controller synthesis case where $k \geq n$, the condition (4.24) is automatically satisfied and the global solution that satisfies the LMI conditions (4.21)~(4.23) can be computed by convex optimization. For the reduced-order problem, the rank constraint (4.24) is non-convex. The overall problem becomes non-convex, and thus it cannot be directly solved by convex solvers. Fortunately, for static output feedback control, the rank constraint becomes $rank(XY-I) = k = 0$ and can be replaced by the following lemma [113] [19].

Lemma 21 *Suppose* $X \in \Re^{n\times n}$ *and* $Y \in \Re^{n\times n}$ *are symmetric and satisfy (4.23), then* $rank(XY - I) = 0$ *if and only if* $tr(XY) = n$.

Based on the lemma above and the cone complementarity linearization algorithm proposed by El Ghaoui *et al.* [22], Ibaraki [19] presented a practical H_∞ method of tuning the fixed-structure controller by using convex optimization algorithms such as SDP, with an application to the H_∞ loop-shaping of a HDD track-following (PID) controller.

H_2/H_∞ Optimal Static Output Feedback Control It is beyond the scope of this dissertation to review the vast literatures of H_2/H_∞ optimal static output feedback control. Interested readers may refer to Berstein [31] for a comprehensive theory of optimal H_2/H_∞ static output feedback control.

4.5.2 Direct Optimization by Non-gradient Methods.

The reason of transforming dynamic controller into static controller is that the overall fixed-structure control synthesis is a non-convex optimization problem. One can get

around this limitation by directly applying non-gradient-based optimization methods. Section 4.3 gives a review of non-gradient-based methods, which are classified as deterministic rule based (e.g. dynamic programming) and stochastic rule based (e.g. genetic algorithms). Such algorithms are not as efficient as gradient-based methods in solving convex optimization problems. But they usually work more closely on the original engineering problems and have no need for auxiliary information. Genetic algorithms (GAs) are a class of stochastic rule based search techniques that work with a population of potential solutions and simulate the natural evolution process. Details about GAs will be discussed in Chapter 5. The most significant advantage of GAs is that they are ready to tackle multi-objective problems in a parallel way. Chapter 5 develops a multi-objective genetic algorithm (MOGA) to solve the fixed-structure controller synthesis problem by the following procedure:

1. Define the design objectives and constraints.
2. Parameterize the problem as multi-objective GAP according to Table 4.2 with user-defined goals.
3. Define the performance evaluation functions in terms of tunable parameters.
4. Solve the GAP by MOGA.
5. Select a preferred solution from the Pareto-optimal set.

To further improve the computational efficiency and solution accuracy, there have been some attempts to combine non-gradient based and gradient based techniques in the hope of generating a more efficient hybridization [114] [45] [115] [116] [117]. The resulting hybrid algorithms usually take advantage of the robustness of the non-gradient techniques

in identifying high performance regions within a non-convex search space and utilize the gradient-based algorithms to rapidly search for the exact optimum. In light of this, Chapter 6 proposes a two-phase algorithm combining the MOGA and the nonlinear programming (SQP) to optimize the parameters of a fixed-structure HDD track-following controller.

The proposed methods allow consideration of arbitrary controller structures (like order, internal structure, and decentralization) and arbitrary parameterizations (like LMI, ARE, and GAP etc.). Actually, some practical design is very difficult or even impossible to be parameterized in standard frameworks. Practicing engineers prefer a flexible optimization technique that can be easily dropped into their practical applications. The examples in Chapter 5 and Chapter 6 adopt the classical frequency-domain loop-shaping techniques that have been successfully used by servo engineers for decades in the analysis and design of HDD servo systems.

4.6 Summary

This chapter first reviewed multi-objective optimization techniques. A number of approaches are currently employed including the ε-constraint, weighted-sum and GAP methods. However, such approaches usually require precise expression of a set of weightings and goals, which is usually not well understood. If the trade-off surface between the design objectives is to be better understood, repeated application of such methods is necessary. The parameterization of fixed-structure controllers is discussed in this chapter. The structure and order constraints make its optimization a non-convex problem. One class of methods are to approximate and parameterize it in a convex form, which is solved by efficient gradient-

based solvers. On the other hand, the population-based Pareto solvers, like GAs, enable the Pareto GAP parameterization, which requires no weighting before optimization and allows consideration of arbitrary controller structures. The next chapter will be devoted to the development of such method.

Chapter 5

Parameter Optimization by Multi-Objective Genetic Algorithm

5.1 Introduction

Fixing the controller structure destroys the convexity of overall optimization problem. The trade-off between reliably finding the global optima and efficiently searching the whole non-convex space makes it challenging to design an optimal fixed-structure controller. Considerable research efforts have been devoted in recent years to address such problem, e.g. [30] [**?**] [118] [110]. The servo design of HDDs needs to achieve multiple objectives and involves many design constraints. It is desirable to provide the decision maker with insight into the trade-offs of the optimization problem before a final balanced solution is selected.

This chapter explores the opportunity of tackling the design of fixed-structure control by using a multi-objective genetic algorithm (MOGA). Section 5.2 addresses the

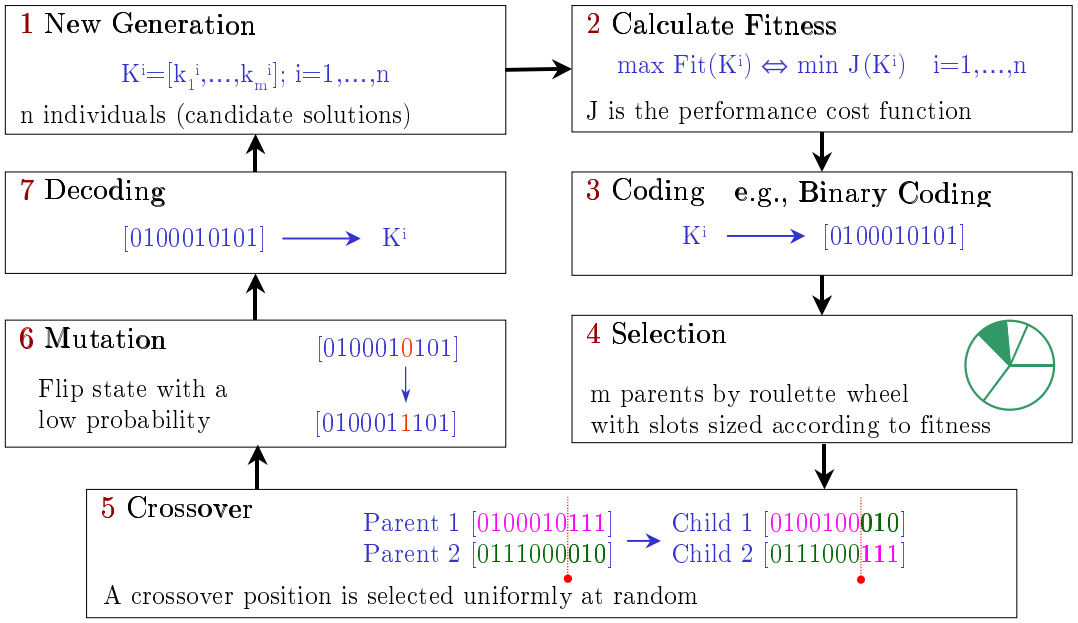

Figure 5.1: The flowchart of a simple genetic algorithm for parameter optimization. It is easier to exemplify genetic operations by using the binary coding.

general idea of genetic algorithms (GAs) in great details. Section 5.3 extends the GAs to address multi-objective optimization problems by adopting the concept of Pareto optimality. In Section 5.4, the proposed MOGA is applied to optimize a fixed-structure HDD track-following controller.

5.2 General Genetic Algorithms

GAs are biologically inspired global searching methods, described in one sentence as "an individual with greater vitality has a better chance to survive in this highly competitive world." Since the initial idea is brought out by Holland [84] in 1975, GAs have attracted a great interest and quickly become a flagship among machine learning and function optimization. It is beyond the scope of this dissertation to review the vast literature

associated with GAs, instead, this section addresses the basic ideas of GAs. The interested reader might consult Goldberg [42] and Michalewicz [43].

GAs have been widely applied to the design of automatic control laws, for example, tuning of PID controller [47] [119] [120] [121] [122] [123], design of H_2/H_∞ optimal fixed-structure control [44] [101], multi-objective control [124] [45] [46], robust control [125], and solving LMI [126].

GAs overcome the limitations of conventional searching algorithms described in Section 4.3 through the following aspects [43] [127].

- GAs use probabilistic transition rules to guide their search, but not deterministic rules.
- GAs use objective function information, but not derivative or other auxiliary data.
- GAs maintain a population of potential solutions to search in a parallel manner, while all other methods process a single point of the search space. This is referred to as "Implicit Parallelism".

The basic building blocks of GAs are shown in Figure 5.1. GAs start with a population of randomly generated solutions. Each solution in this very first generation will be evaluated through cost functions that are defined according to design objectives. During the evaluation, we define a fitness function and evaluate the fitness for each candidate solution. The fitness is assigned in a way that maximizing the fitness is equivalent to minimizing the cost function. Then the population is processed and evaluated through various operators to generate a new population. The basic operators of GAs are *selection*,

crossover, and *mutation,* as shown in steps 4, 5, and 6 of Figure 5.1 respectively. This process is repeated until a global optimal point is reached. The following subsections examine this process in details.

5.2.1 Coding: From Binary to Real

GAs work with a coding of parameters, but not parameters themselves. The main purpose of coding is to make it possible to imitate the natural evolution process in parameter space. The traditional way to illustrate GAs is to use *binary coding* (BC) . As shown in Figure 5.1-(3), each parameter is encoded into a string of bits. The individual bit is called a *gene*. The content of each gene is called an *allele*. The whole string of such genes for all parameters in a written sequence is called a *chromosome*. The choice of chromosome length depends on the accuracy requirement of the targeting optimization problem. GAs maintain a population of chromosomes or individuals in every generation, quite often these individuals are called *candidate solutions*. The number of chromosomes in a population is called the population size, denoted as n_{pop}. These chromosomes will evolve from generation to generation through some genetic operators.

However, there are some drawbacks when applying BC to multidimensional, high-precision numerical problems. For example, for 80 variables with domains in the range [-100 100] where a precision of six digits after the decimal point is required, the minimum length of the binary chromosome is $27 \times 80 = 2160$. For such a problem GAs perform poorly because the BC generates a search space of about 2^{2160}.

For many optimization problems, the elements in the space to be searched are naturally represented by data structure such as arrays, trees, digraphs, etc. It is more

natural to keep the original data structure of target problems and make the genetic operators work directly on such structure. Instead of using BC, this dissertation uses real-coded (RC) encoding scheme, i.e. each candidate solution is represented as a float-point vector $K^j = [k_1^j, ..., k_m^j]$. At $(t)th$ generation, MOGA maintains a population of individuals, $\mathbf{K}(t) = [K^1(t), ..., K^j(t), ..., K^{n_{pop}}(t)]$, with a fixed population size n_{pop}. This real-coded (RC) representation has many advantages over the classical binary-coded (BC) representation [43].

- The RC is faster, more consistent from run to run and provides a higher precision than BC. The precision of RC depends on the underlying machine, hence is generally much better than that of the BC. Although we can always enhance the precision of the RC by introducing more bits, this considerably slows down the algorithm.

- The RC is capable of representing quite large domains. On the other hand, the BC must sacrifice precision with an increase in domain size for a given fixed binary length. Thus the RC avoids Hamming Cliffs effect from which the BC suffers.

- The RC is conceptually closer to the problem space, so virtually no decoding is required. The RC also allows for an easier implementation of genetic operators.

5.2.2 Initial Population

A simple GA is an iterative procedure starting with a randomly generated population of candidate solutions. Since one has little geometric knowledge of the search space of target problem before solving it, the uniform distribution is arguably the best initialization scheme. Keep in mind that the population size n_{pop} is the only factor in this method

to balance the trade-off between initial diversity and computational complexity. If n_{pop} is too small, the chance that chromosomes in the population cover the entire search space is low. This makes it difficult to obtain the global optimal solution and may lead to a local optimum as a result of premature convergence. On the other hand, a population size that is too large decreases the rate of convergence. In the worst case scenario, it may lead to divergence. Hence the population size needs to be carefully selected based on the size of search space. An example at the end of this section will clearly illustrate this relationship.

Individuals in the first generation are evaluated and assigned fitness values based on their relative performance. Through applying genetic operators, the initial population will evolve into a new population which contains a group of better solutions.

5.2.3 Cost Function and Constraint Handling

Cost functions are directly related to our design objectives and constraints. If we are to solve a minimizing problem, the objective function itself can be used as cost function because GAs are aimed at minimizing cost functions. On the other hand, to define the cost function for maximization problems, one usually just takes the reciprocal or flips the sign of objective functions. Since GA is a population-based probabilistic optimization method and does not require the convexity of search space, the cost function $J(K)$ can be freely chosen for the convenience of designer.

5.2.4 Fitness

After the function evaluation, a real positive number called fitness is assigned to each candidate solution in the way that maximizing the fitness is equivalent to minimizing

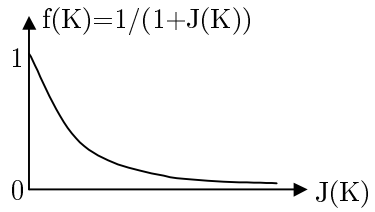

Figure 5.2: An example of fitness assignment for a minimizing problem with positive semidefinite objective function.

the cost function, i.e.,

$$max[Fit(K)] \Longleftrightarrow min[J(K)] \Longleftrightarrow \text{achieve objective and/or satisfy constraint.} \qquad (5.1)$$

The fitness serves as the surviving probability of each individual during genetic operations as described in the next section. An individual with a bigger fitness value implies higher quality with respect to design objectives, and a better chance to survive in genetic operations. The programmer is allowed to use any fitness function that adheres to the relationship in (5.1). This advantage over other optimization methods makes GAs more attractive when dealing with practical engineering optimization problems. Eq. (5.2) and Figure 5.2 show an example of defining fitness for a minimizing problem with positive semidefinite objective function. $Fit(K)$ gets the biggest value of 1 (for a possibility may not be greater than 1) when minimum $J(K) = 0$, and converges to zero as $J(K)$ increases.

$$Fit(K) = \frac{1}{1 + J(K)} \qquad (5.2)$$

5.2.5 Genetic Operators: Selection, Crossover, and Mutation

A new generation is generated from the current one by examining the fitness values of all candidate solutions and applying the genetic operators, which include *selection*, *crossover*, and *mutation*. Through the genetic operations, the survival of the fittest means transferring the highly fit chromosomes to the next generation and combining different chromosomes to explore new search points.

Selection The roulette-wheel-like *selection* is more likely to pick up candidates with higher fitness values into a mating pool. Each candidate solution corresponds to a slice in the weighted roulette wheel (Figure 5.1-(4)). The relative size of the slice equals to the fitness percentage of corresponding solution in the total fitness. The total fitness is obtained by summing the fitness over all individuals in the current generation. Every spin of the weighted roulette wheel yields one reproduction candidate for parent pool. Since the time to stop is a uniform random variable, the possibility of any individual being picked up is proportional to its fitness. This spinning process needs to be repeated until enough parents are generated.

Crossover The selected solutions in parent pool are then processed by applying *crossover* which pairs up the parents and exchanges portion of their segments pairwisely. The step 5 of Figure 5.1 shows a single point crossover for BC chromosomes. A *crossover point* is selected randomly as an integer between 1 and the chromosome length. Then children Child 1 and Child 2 are generated by exchanging the segments after the crossover point (bit 8-10 in Figure 5.1) between Parent 1 and Parent 2.

Besides the crossover point, another parameter called *crossover probability* P_c, is used to control the number of crossovers by acting as a decision variable before performing the crossover. A real number is uniformly generated in the range [0 1]. If this number is less than P_c, a crossover is performed, otherwise, Child 1 and Child 2 are simply direct copies of Parent 1 and Parent 2.

Mutation Nature uses large population sizes to keep her diversity. However, it is expensive to keep a very large population in GAs. Instead, *Mutation* performs a slight perturbation to the resulting solutions with a very low probability. As illustrated in the step 6 of Figure 5.1, the 7th gene is the bit being mutated. This is an effective way to preserve the diversity given a limited population size.

In this dissertation, a special dynamic mutation operator is used to improve single-element tuning and reduce the disadvantage of random mutation in the RC implementation. It is called *non-uniform mutation* [43]. If $K^j(t)= [k_1^j(t), ., k_i^j(t).., k_m^j(t)]$ is a chromosome in (t)th generation and the element k_i^j is selected for applying non-uniform mutation, the result is a vector $K^{j'}(t) = [k_1^j(t), ., k_i^{j'}(t).., k_m^j(t)]$, where

$$k_i^{j'} = \begin{cases} k_i^j + \Phi(t, UB - k_i^j) & \text{if a random digit is 0} \\ k_i^j - \Phi(t, k_i^j - LB) & \text{if a random digit is 1} \end{cases} \tag{5.3}$$

and LB and UB are lower and upper domain bounds of the variable k_i^j. The function $\Phi(t, z)$ is chosen such that it returns a value in the range $[0, z]$ and the probability of $\Phi(t, z)$ gets close to 0 as t increases. This property causes the non-uniform mutation to search the space uniformly initially and very locally at latter stages; thus increases the probability of generating a new point closer to the original one rather than a random choice. According

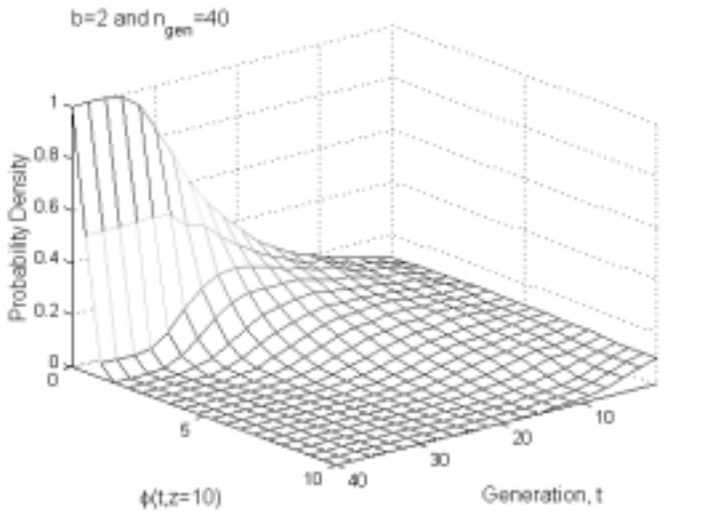

Figure 5.3: The probability density of $\phi(x, z)$ with respect to evolving generation t when b=2.

to this requirement, the following function is used in this dissertation:

$$\Phi(t, z) = z \cdot \left(1 - r^{(1-\frac{t}{n_{gen}})^b}\right) \tag{5.4}$$

where $r \in [0, 1]$ is a random number, n_{gen} is the maximal generation number, and b is a system parameter determining the degree of dependency on iteration number. As shown in Figure 5.3 and Figure 5.4, the bigger b is, the faster the probability density of $\Phi(t, z)$ becomes concentrated on zero with the increase of t.

5.2.6 Solving a Benchmark Problem by GA

Solving the following example (5.5) shows the effectiveness of GA in dealing with non-convex optimization problems. The global minimum of this example, $J(x_i) = 0$, is apparently achieved at origin $x_i = 0$, as shown in Figure 5.5 ($n = 1$). The gradient-based

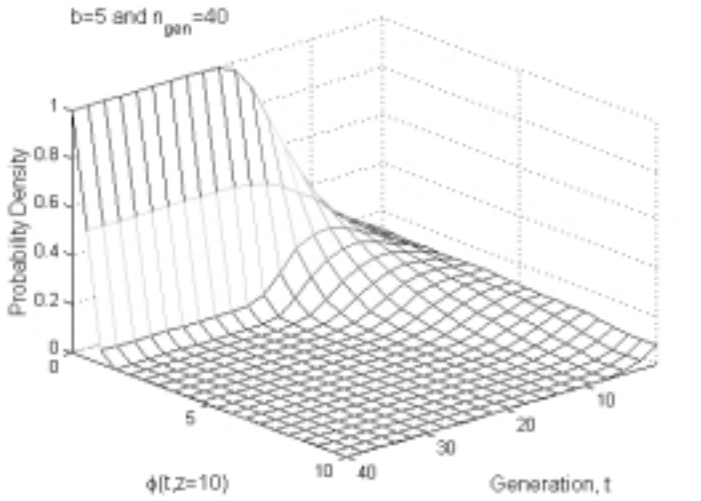

Figure 5.4: The probability density of $\phi(x, z)$ with respect to evolving generation t when b=5.

Table 5.1: The computational cost and solution quality v.s. the number of generations and population size

n_{gen}	5	10	15	30	50	100
n_{pop}	600	300	200	100	60	30
CPU time (s)	50.86	25.16	18.46	12.53	10.77	10.03
$\max_i x_i^{opt}$	0.15	0.0015	0.001	0.0016	0.00027	0.0047

methods are likely to be trapped in numerous local minima at $x_i = 2k\pi, k = 0, 1, \ldots$.

$$\begin{aligned} \min_{x_i} [J(x_i)] &= \min_{x_i} \sum_{i=1}^{n} \left(\left| \frac{x_i}{2} \right| + 4 \left| \sin(\frac{x_i}{2}) \right| \right) \\ x_i &\in [-40, 40];\ n = 1, 2, \ldots \end{aligned} \tag{5.5}$$

In De Jong's [128] study of genetic algorithms in function optimization, a series of parametric studies across a five function suite of problems suggested that good GA performance requires the choice of a moderate population size n_{pop}, a high crossover probability P_c, and a low mutation probability P_m, which is inversely proportional to the population

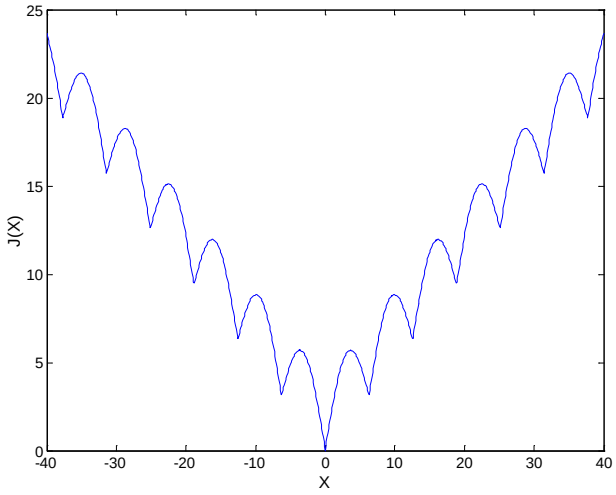

Figure 5.5: The illustration of the non-convex optimization example in Eq. (5.5) when n=1.

Table 5.2: Robustness comparison with other algorithms

		$\max_i x_i^{opt}$ (The global optimum has been found:$\surd$; otherwise:$\times$)					
Algorithms		Interior Point		Simplex		GA	
MATLAB function		fminunc		fminsearch		(n_{gen}=30; n_{pop}=100)	
Initial point x_i^0 $i = 1, ..., 10$	$\frac{\pi}{100}$	0.7072×10^{-10}	$\surd$	0.0458	$\surd$	0.016 (Random initial points)	$\surd$
	$\frac{\pi}{2}$	-0.1702×10^{-8}	$\surd$	4.4284	$\times$		
	$\frac{3\pi}{2}$	6.2823	$\times$	6.2823	$\times$		
	$\frac{9\pi}{2}$	6.2823	$\times$	18.9125	$\times$		

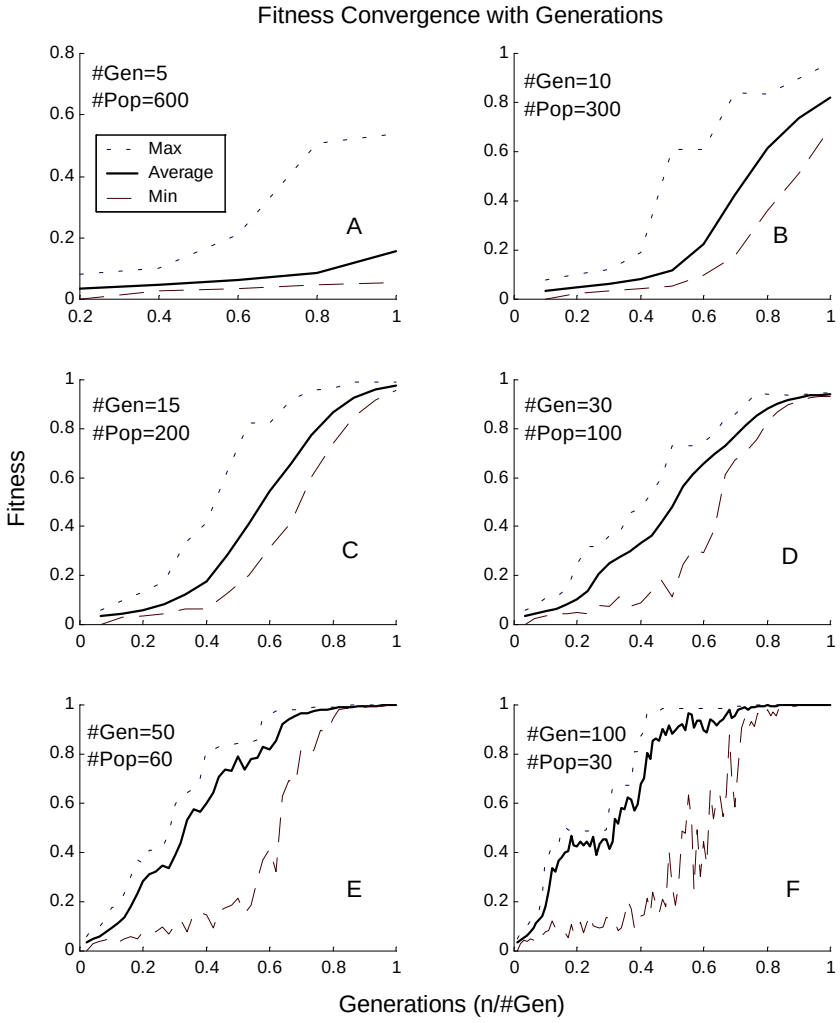

Figure 5.6: The right choice of the populaton size in one generation and the number of generations is very important in GAs. All trials have a total of 3000 samples.

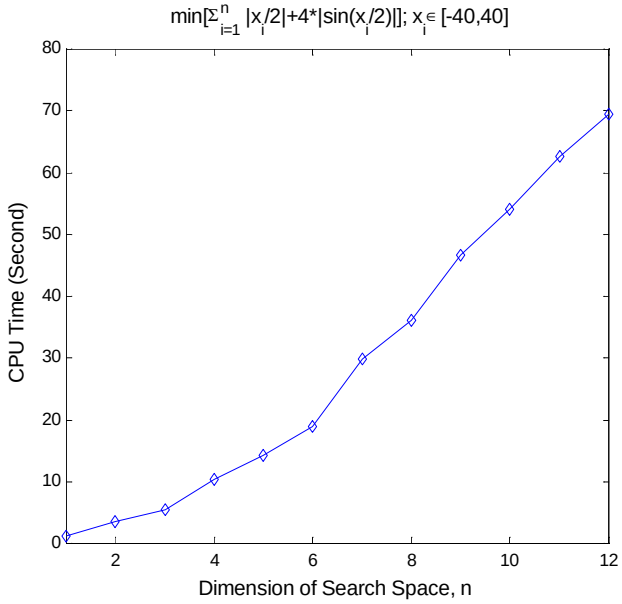

Figure 5.7: Upon applying the GA to solve the given example, the relationship between the time to convergence ($|x_{i\_opt}| < 10^{-3}$) and the demension of search space is almost linear.

size n_{pop}.

To deliver the best performance of GA with a given computational power, the right choice of the number of generations and the population size is crucial. Figure 5.6 illustrates the interaction between these two quantities on the fitness convergence. Table 5.1 shows their impact on the computational cost and solution quality. For a fair comparison, all six cases have the same total number of 3000 samples. The time to convergence for all cases is defined as when all variables of the best solution converge to a hyper-ball centered at the origin with a radius of 10^{-3}. We can see from Figure 5.6 that if the population size is too small, it may lead to the premature convergence problem because of the lack of diversity. Graphically it appears as a plateau in the average fitness curve before it converges to 1. A fitness of 1 generally corresponds to the global optimality (Figure 5.6-F). Other

graphical evidences of the lack of diversity include the existing of very rough curves, and that the average fitness curve is much closer to the maximum fitness curve than to the minimum fitness curve. As the population size is increasing, however, the convergence speed is lowering and the computational cost is increasing. As shown in Table 5.1 and Figure 5.6-A, using too large population size not only wastes computational resource but also lowers the quality of solution. All simulations are performed on a AMD 1GHz PC with 256MB memory.

Figure 5.7 shows the relationship between the computational cost (the time for all Pareto solutions converging to a small hyper-ball around the origin) and the dimension of search space. Unlike the enumerative or LMI+approximation+SDP methods, the CPU time representing the overall computational complexity does not increase exponentially, but rather grows in a linear manner with respect to the problem size. For non-convex MOPs with large number of tunable parameters, this is truly a big advantage of GAs over other optimization techniques.

In Table 5.2, we compared the robustness of GA with two existing algorithms, gradient-based interior point method and non-gradient-based deterministic simplex method, which correspond to functions fminunc and fminsearch respectively in the MATLAB Optimization Toolbox. fminunc is a large-scale subspace trust region method which is based on the interior-reflective Newton method [94]. fminsearch uses the simplex search method that does not use numerical or analytic gradients and is claimed to be able to handle discontinuity. The performance of both algorithms depends on the initial estimate. It is surprising that the simplex method fminsearch fails in most cases except the one in which the initial

point is very close to the global optimum. The interior point method fminunc performs well when initial points are within the monotone region around the global optimum but fails otherwise. Apparently both of them cannot handle the non-derivative local minimal points $x_i = 2k\pi$. This experiment shows the excellent robustness of GA in non-convex optimization.

We would like to point out that there are no hard and bound restrictions on what operators and strategies a programmer has to use. So one has the freedom to choose the operators and strategies in any combination according to the characteristics of target problem.

5.3 Multi-Objective Genetic Algorithm

This section proposes a multi-objective genetic algorithm (MOGA), which extends the general GAs to handle multi-objective optimization problems, especially the ones with non-commensurable and competing objectives. Ishibuchi and Murata [124] presented the single-objective genetic algorithm (SOGA) that translates multiple objective functions into a single-objective function by using weightings. Several multi-objective genetic algorithms (MOGAs), which are GAs combined with the concept of Pareto optimality, have been proposed in [42] [48]. A good review of MOGAs is in [51]. Many researchers [47] [125] [126] have successfully applied MOGA to solve MOPs.

The flow chart of the MOGA used in this dissertation is shown in Figure 5.8. The details are described below. A summary of this MOGA is given at the end of this section.

Table 5.3: Relation between goals of GAP and cost functions of MOGA

Goals of GAP	Cost functions of MOGA
$\underset{K}{\text{goal}}(f_i(K) \leqslant f_i^0)$ (MOP objective)	$\min_K \{J_i(K) = f_i(K)\}$
$\underset{K}{\text{goal}}(f_i(K) \geqslant f_i^0)$ (MOP objective)	$\min_K \{J_i(K) = -f_i(K)\}$
$\underset{K}{\text{goal}}(g_i(K) \leqslant g_i^0)$	$\min_K \{J_i(K) = Q[g_i(K) - g_i^0]\}$
$\underset{K}{\text{goal}}(g_i(K) \geqslant g_i^0)$	$\min_K \{J_i(K) = Q[g_i^0 - g_i(K)]\}$
$\underset{K}{\text{goal}}(g_i(K) = g_i^0)$	$\min_K \{J_i(K) = \lvert g_i^0 - g_i(K)\rvert\}$
$\underset{K}{\text{goal}}(g_i^{LB} \leqslant g_i(K) \leqslant g_i^{UB})$	$\min_K \{J_i(K) = \max(Q[g_i(K) - g_i^{UB}], Q[g_i^{LB} - g_i(K)])\}$

5.3.1 Cost Functions of Multi-objective GAP

Although the optimization problem of fixed-structure controller can be formulated as solving LMIs, limiting the objective functions in quadratic form does not fully explore the potential of GAs. Some practical design problem is very difficult or even impossible to be parametrized in standard frameworks. On the other hand, GAP formulation has no convexity limitation of any kind thus it can be fitted into most practical multi-objective control problems. As shown in Table 5.3, we follow the defining of GAP goals to setup cost functions of MOGA. One minor difference is that for *min* (or *max*) objectives, we set the $f_i^0 = -\infty$ (or ∞) because MOGA is naturally constructed for minimizing. $Q[.]$ is the explicit goal function

$$Q[a] = \begin{cases} a & a > 0 \\ 0 & a \leqslant 0 \end{cases}. \tag{5.6}$$

Certain types of goals, such as bounds on the search space, can be handled by mapping the search space so as to minimize the number of infeasible solutions it contains and designing the genetic operators carefully in order to minimize the production of infeasible

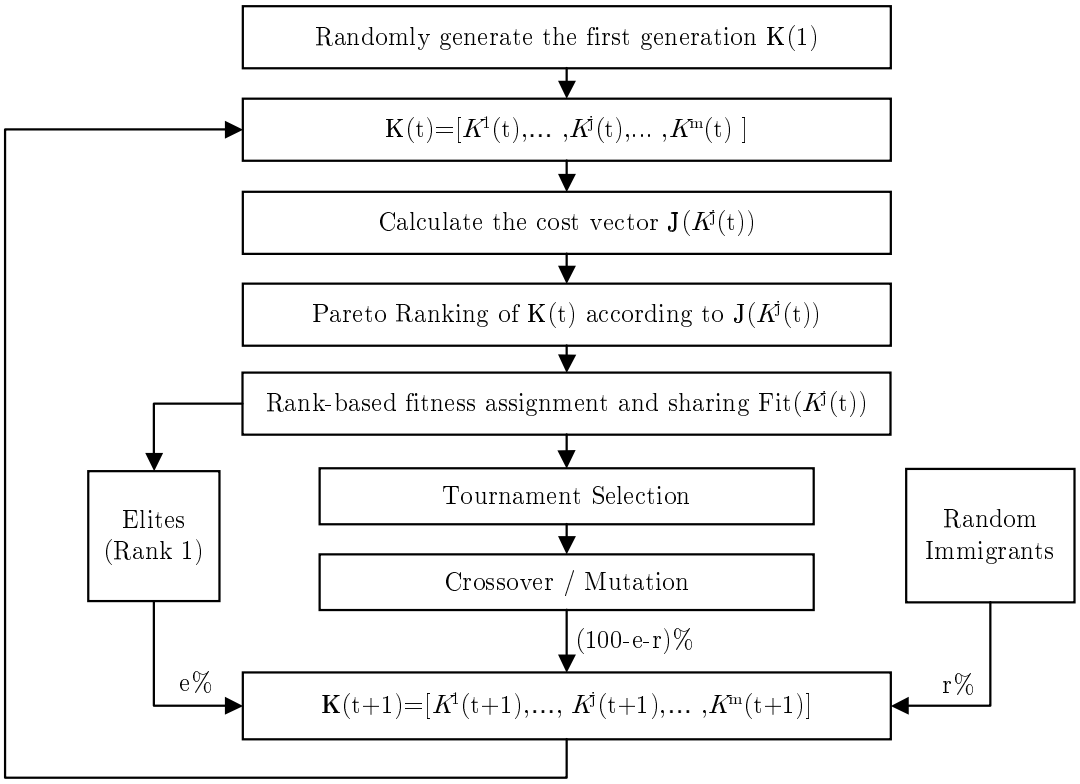

Figure 5.8: The flow chart of MOGA with elitism and random imigrants.

offspring from feasible parents [129].

5.3.2 Pareto Ranking

All K^j in the Pareto optimal set have the similar vectorial performance and thus are so-called "the equally best solutions" among the current generation $\mathbf{K} = [K^1, ..., K^{n_{pop}}]$. Therefore they are assigned the same rank of 1. The final solution of a MOP depends only on the vectorial performance and on the preferences of the decision maker, and not on any subsequent optimization [100]. Based on (4.9) and (4.10), the following Pareto ranking scheme derived from [119] is proposed for the MOGA,

Algorithm 22 (Pareto Ranking)

1. Sort $\mathbf{K} = [K^1, ..., K^j, ..., K^{n_{pop}}]$ from the least to the largest according to $||\mathbf{J}(K^j)||_1 = \sum_{i=1}^{n} J_i(K^j)$. The sorted vector is denoted as $\mathbf{K}_s$. Let $Rank$=1.

2. Use the first entry of $\mathbf{K}_s$, K_s^1, as criterion. Take out any K_s^j from $\mathbf{K}_s$ and put it into a dominated vector Φ_d if $\mathbf{J}(K_s^1) <_p \mathbf{J}(K_s^j)$, $j = 2, ..., length(\mathbf{K}_s)$.

3. Move out K_s^1 from $\mathbf{K}_s$ and put it into a Pareto optimal set $\mathbf{K}_p$.

4. Let the remaining candidate solutions in $\mathbf{K}_s$ form a new $\mathbf{K}_s$, then repeat 2 to 4 until all dominated solutions are removed.

5. Assign all entries in $\mathbf{K}_p$ with the same rank, $Rank$. Empty $\mathbf{K}_p$.

6. Replace $\mathbf{K}_s$ with Φ_d, i.e. $\mathbf{K}_s = \Phi_d$.

7. $Rank = Rank + 1$ and repeat 2 to 7 until the entire population is ranked.

5.3.3 Rank-based Fitness Assignment

Every candidate solution K is assigned a fitness value $Fit(K)$ which is the measurement of solution quality. For a candidate solution in MOGA, the smaller the rank number, the better the vectorial performance. By selection, the MOGA is biased to the solution with higher fitness value. Therefore the fitness assignment should be a mapping that maximizing the fitness $Fit(K)$ is equivalent to minimizing the cost vector $J(K)$, i.e. maximizing the vectorial perform of K. In this dissertation, a simple exponential mapping

is used

$$Fit^{'}(K) = \frac{1}{rank(K(t))}. \tag{5.7}$$

5.3.4 Intra-Rank Fitness Sharing

Although all "equally good" solutions are assigned the same fitness, their actual choice to be selected as parents may differ due to the random nature of selection. This imbalance can be accumulated with evolutions such that the population drifts towards an arbitrary region of the trade-off surface, a phenomenon known as *genetic drift* [130]. Various population diversity mechanisms have been proposed to make a GA to maintain a diverse population of individuals through its search [100]. These mechanisms allow GAs to identify multiple optima in a multimodal objective domain. *Intra-Rank Fitness sharing* is one of such mechanisms to counteract the genetic drift by re-distributing the fitness among the candidate solutions with the same rank [100]. The sharing function penalizes the fitness of individuals in popular neighborhoods and is in favor of more remote individuals. In this dissertation, the sharing function is defined as

$$Fit(K^j) = \frac{D(K^j)}{\sum_j D(K^j)} \sum_j Fit^{'}(K^j) \tag{5.8}$$

where $D(K^j(t))$ is the *mutual similarity distance* which is the summation of similarity distance between $K^j(t)$ and any solution $K^i(t)$, $i \neq j$, with the same rank in the current generation. $Fit^{'}(K^j)$ is the original fitness value for K^j and $Fit(K^j)$ is the fitness after sharing. The similarity distance can be defined either in the objective space (phenotype) [50] or in the solution space (genotype) [99]. A genotype similarity distance is usually objective independent metric, such as the Hamming distance between two candidate solutions. It

is used if there is little knowledge about the objective space. A phenotype metric is more meaningful distance measurement quantifying the similarity of two solutions in the objective space.

5.3.5 Elitism and Random Migrants

To increase the converging rate of the MOGA, the elites (individuals with the Rank 1) of current generation are directly copied into the new generation, taking up to $e\%$ of total population. The upper limit $e\%$ is enforced to prevent the premature convergence. As a complementary mechanism of mutation, $r\%$ of total population are generated randomly to preserve the population diversity. The remaining $(100 - e - r)\%$ individuals are generated from the process of selection/crossover/mutation.

The MOGA used in this dissertation is summarized as follows.

Algorithm 23 (MOGA)

1. Determine the search space of K, $\Omega_K \subset R^m$, which is the range of K that stabilizes the nominal plant $P_0(z^{-1})$ by Jury Criterion [66].

2. The MOGA randomly and uniformly generates the first generation with n_{pop} individuals (candidate solutions), $[K^1(1), ..., K^j(1), ..., K^{n_{pop}}(1)] \in \Omega_K$. Each individual is represented by RC.

3. For each individual $K^j(t)$ in the current $(t)th$ generation, calculate the cost function vector $\mathbf{J}(K^j(t)) = \{J_1(K^j(t)), ..., J_i(K^j(t)), ..., J_n(K^j(t))\}$.

4. Pareto ranking of $[K^1(t), ..., K^j(t), ..., K^{n_{pop}}(t)]$ according to $\mathbf{J}(K^j(t))$.

Table 5.4: MOGA configurations

Description	Symbol	Value
Number of generations	n_{gen}	60
Number of individuals per generation	n_{pop}	200
Probability of crossover	P_c	1
Probability of mutation	P_m	0.02
Maxium percentage of elitism	$e\%$	30%
Percentage of random immigrants	$r\%$	6%

5. Assign fitness to $K^j(t)$ based on its ranking, $Fit'(K^j(t)) = \frac{1}{rank(K^j(t))}$, and apply fitness sharing, $Fit(K^j(t)) = \frac{D(K^j)}{\sum_j D(K^j)} \sum_j Fit'(K^j(t))$.

6. Directly migrate the elite, individuals with rank 1, to the $(t+1)th$ generation. This makes up $e\%$ of total population n_{pop}, where $e\%$ is up to a certain percentage.

7. Randomly generate $r\%$ of total n_{pop} individuals in the search space Ω_K for the $(t+1)th$ generation.

8. Apply the *tournament selection* [42] to generate $n_{pop} \cdot (100 - e - r)\%$ parents from the $(t)th$ generation. A linear crossover is used to produce $n_{pop} \cdot (100 - e - r)\%$ new individuals from these parents. Apply *mutation* to these new individuals.

9. Set $[K^1(t+1), ..., K^j(t+1), ..., K^{n_{pop}}(t+1)]$ as the current generation. Should this new generation achieve the optimization goal, stop the MOGA; otherwise go to step 3.

5.4 Example: Optimization of HDD Controller Parameters

The specifications of the disk drives used in our experiments are listed in 3.1. The track-following controller for these drives is an extended PID controller in the following

form

$$C_{pid}(z^{-1}) = k_{loop}[1 - (k_a + k_b)z^{-1} + k_a k_b z^{-2} + k_i \frac{1}{1 - z^{-1}}] \tag{5.9}$$

where k_{loop}, k_a, k_b, and k_I are design parameters. A digital notch filter is added to attenuate the adverse effect due to the suspension resonance at 2700Hz. The integrator gain, K_I, is predetermined and hence not a target of optimization. So the tunable parameter set is $K = [k_{loop}, k_a, k_b]$. The original controller used by these drives was $K_{orig} = [k_{loop}, k_a, k_b] = [0.270, 0.850, 0.400]$, which was hand-tuned by experienced engineers in the previous design cycle. The search space $\Omega_K = [(0.01 : 0.98), (-0.8 : 0.99), (-0.8 : 0.99)]$ is the range of K that stabilizes the nominal plant. The gradient method used in [23] had always failed to converge in this range. To make our method more practical for most servo engineers in HDD industry, we follow the classical control framework. Since we are designing the controller for the mass produced HDDs with plant variations and disturbance uncertainties, σ^2_{pes} is not a constant but rather a random variable. It is more appropriate to assess the average performance $mean(\sigma^2_{pes})$ and the performance robustness $std(\sigma^2_{pes})$. The major objectives of HDD track-following control therefore are

$$\begin{aligned} J_1(K) &= mean(\sigma^2_{pes}) \\ J_2(K) &= std(\sigma^2_{pes}) \end{aligned} \tag{5.10}$$

The statistical model developed in Chapter 3 is used to synthesize these two indexes.

Design goals are $GM_{goal} = 6.5dB$, $PM_{goal} = 42^o$, $\omega_{ogoal} = 650Hz$, and $S_{\infty goal} = 5dB$. Note that the goals are slightly higher than the nominal performance in order to ensure a converging pressure over the whole evolution process. We also impose some search

constraints, $GM_{MAX} = 10dB$, $PM_{MAX} = 60^o$, and $\omega_{oMAX} = 800Hz$ to prevent generating some senseless solutions. By following aforementioned guideline (Table 5.3), the corresponding cost functions are defined as

$$\begin{aligned} J_3(K) &= \max\{Q[GM_{goal} - GM(K)], Q[GM(K) - GM_{MAX}]\} \\ J_4(K) &= \max\{Q[PM_{goal} - PM(K)], Q[PM(K) - PM_{MAX}]\} \\ J_5(K) &= \max\{Q[\omega_{ogoal} - \omega_o(K)], Q[\omega_o(K) - \omega_{oMAX}]\} \\ J_6(K) &= Q[S_\infty(K) - S_{\infty goal}] \end{aligned} \tag{5.11}$$

where the phase margin $PM(K)$, the gain margin $GM(K)$, the crossover frequency $\omega_o(K)$, and the peak of sensitivity function $S_\infty(K)=\|S(K,j\omega)\|_\infty$ are functions of the tunable parameter set $K \in \Omega_K$. $J_3(K) \sim J_6(K)$ will be zero if the candidate solution K satisfies all constraints. Thus, the original constrained MOP is converted to an unconstrained MOP whose target is to minimize the cost vector $\mathbf{J}(K) = \{J_1(K), ..., J_i(K), ..., J_6(K)\}$ over K. $\mathbf{J}(K)$ is therefore called the *vectorial performance* index.

The MOGA configurations are listed in Table 5.4. It took the MOGA 7.3 minutes in a AMD 1GHz computer to give a Pareto optimal set (Rank 1) with 23 solutions in the $60th$ generation. One solution, $K_{MOGA} = [0.246, 0.870, 0.275]$ which has the minimal $std(\sigma^2_{pes})$ among the Pareto optimal set, was picked up and loaded into twelve drives. The performance comparison between this optimized controller and the original one is shown in Table 5.5. The 3% improvement of $mean(\sigma^2_{pes})$ and 8.7% of $std(\sigma^2_{pes})$ in experiments may be considered moderate because the original controller had been well optimized. It can also be seen that $mean(\sigma^2_{pes})$ and $std(\sigma^2_{pes})$ predicted by the model match up reasonably well with the experimental results. The disturbance model was then updated based on new

Table 5.5: Performance comparison before and after MOGA optimization

		Before Optimization (K_{orig})		After Optimization (K_{MOGA})	
Objective	Unit	Simu	Exp	Simu	Exp (relative to orig)
$mean(\sigma^2_{pes})$	$tr\%^2$	5.24	5.22	5.05	5.09(-2.9%)
$std(\sigma^2_{pes})$	$tr\%^2$	2.25	2.31	2.02	2.11(-8.7%)
GM	dB	6.1	6.3	6.5	6.3
PM	$degree$	42.1	40.4	41.5	40.4
ω_o	Hz	540	544	600	595
S_∞	dB	5.7	5.5	5.9	5.8

PES measurements, and ready for the next run of optimization. This iterative optimization process can be repeated until satisfactory results are obtained.

5.5 Summary

This chapter developed a MOGA (multi-objective genetic algorithm) to directly solve the multi-objective non-convex optimization problem. As shown by simulations and experiments, the proposed method was capable of optimizing the controller in a *large range* in which gradient-based methods generally failed. The proposed method has the following advantages over existing tuning methods:

1. The population-based nature of the MOGA enables the evolution of a set of Pareto-optimal solutions without requiring weightings before optimization.

2. Compared to gradient based methods, the MOGA greatly increases the possibility of finding the global optima for an extremely broad class of optimization problems, especially non-convex optimization problems. The MOGA gives designer multiple Pareto solutions without tedious trial-and-errors for picking up weightings.

3. The GA offers almost linear computational complexity relative to the number of tunable variables. This is a very attractive feature for non-convex MOPs with a large number of variables. For example, the track-following controllers for dual-stage actuators usually have over twelve tunable variables for which the LMI + convex approximation parameterization is practically unsolvable.

4. Users can use any optimization setup as long as the cost functions are evaluatable in closed-form. For example, the statistical model developed in Chapter 3 was used to effectively predict the time-domain performance of candidate solutions for a large population of drives.

However, like any optimization algorithm, there is "no free lunch" [131]. In other words, no optimization algorithm is universally better than other algorithms. There is a trade-off between efficiency and applicable generality. Although the ideas of GAs are applicable to a broad range of problems, different GAs can be built, which displays a varying degree of problem dependence. GAs incorporating some problem specific knowledge generally outperform the one in canonical form, however, with the sacrifice of generality. The major drawback of the approach proposed in this chapter is that the MOGA suffers from poor numerical tractability. It should be noted that because of the stochastic nature of GAs, one cannot guarantee the convergence of GAs to the global optima in a deterministic way. Our experience has shown that the Pareto sets are different from run to run, although they are all close to the global optima in the objective space. In next chapter, a two-phase optimization method will be proposed to enhance the local fine-tuning capability of the MOGA.

Chapter 6

A Two-Phase Optimization Algorithm

6.1 Introduction

In the previous chapter, the MOGA was used to search for the Pareto-optimal parameter set of fixed-structure controller, through solution of an appropriately formulated GAP problem. The MOGA is capable of identifying the high performance region in non-convex search space, yet slow in fine-tuning local search. The major drawback of the MOGA is its poor computational tractability. Because of the stochastic nature of the MOGA, the Pareto solution set is slightly different from run to run. It has been found that the improved accuracy has to be sacrificed for efficiency [43]. On the other hand, as shown by the benchmark problem in Section 5.2.6, in a non-convex search space the results of gradient-based methods heavily depend on the initial starting points. If a good starting

point is encountered, the gradient-based search algorithms can efficiently attain the optimal solution with good numerical tractability. Yet, if a poor initial point is utilized, the result is likely to be trapped in a local minimum.

In this chapter, a numerical algorithm is presented by combining the advantage of the MOGA and the gradient-based techniques to provide a practical technique for tuning the fixed-structure controller. First, the MOGA performs a global search and seeks a near-optimal initial point for the second stage. In the second stage, a gradient-based nonlinear programming method is used to perform local search to further enhance the quality of solutions.

6.2 Nonlinear Goal Attainment Programming

Recall the nonlinear GAP setup presented in Section 4.4. A constrained MOP can be reformulated as an unconstrained GAP

$$\begin{array}{ll} \underset{\gamma\in\Re,\mathbf{K}\in\Omega}{\text{minimize}} & \gamma \\ \text{subject to} & J_i(K) \leqslant J_i^0 + w_i\gamma,\ i = 1, ..., m \end{array} . \tag{6.1}$$

The mechanism of Eq. (6.1) can be illustrated by a two-objective problem (m=2) as shown in Figure 6.1.

$$\begin{array}{lll} \underset{\gamma\in\Re,\mathbf{K}\in\Re^n}{\text{minimize}} & \gamma & \\ \text{subject to} & J_1(K) \leqslant J_1^0 + w_1\gamma & \\ & J_2(K) \leqslant J_2^0 + w_2\gamma & \\ & K \in \Omega & \end{array} \tag{6.2}$$

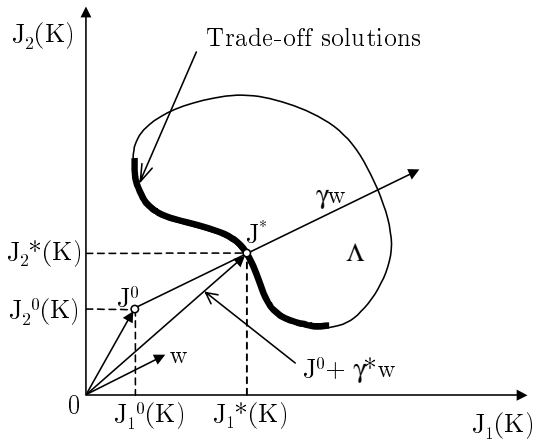

Figure 6.1: Goal attainment method with two objectives.

The weighting vector $\mathbf{w} = \{w_1, w_2\}$ defines the direction of search. $\mathbf{J}^0 = \{J_1^0, J_2^0\}$ are the design goals that are not necessary in the feasible region Ω. Given $\mathbf{w}$ and $\mathbf{J}^0$, the *direction* of vector $\mathbf{J}^0 + \gamma\mathbf{w}$ is determined. During the optimization γ is varied, which changes the length of $\mathbf{J}^0 + \gamma\mathbf{w}$**.** The Eq. (6.1) is equivalent to finding a feasible point along vector $\mathbf{J}^0 + \gamma\mathbf{w}$ in the objective space. When the weight w_i is positive, the GAP attempts to make the objective J_i less than the goal value J_i^0. To make the objective J_i greater than the goal value, set the weighting w_i to negative. The optimal value of the slack variable, denoted as γ^*, indicates whether the goals are attainable or not. A negative value of γ^* implies that the goal is attainable, and an improved solution will be obtained. Otherwise, if γ^* is positive, then the goal is unattainable. In the example of Figure 6.1, $w_1 > 0$ and $w_2 > 0$ mean that we try to find $\mathbf{J}(\mathbf{K}) \prec \mathbf{J}^0$. However, $\gamma^* > 0$ indicates that this goal cannot be achieved but the GAP does give a solution $\mathbf{J}^* = \mathbf{J}^0 + \gamma^*\mathbf{w}$ that has the least degree of under-achievement.

The motivation of parameterizing MOP in this GAP form is that Eq. (6.1) can be solved by nonlinear programming techniques, like sequential quadratic programming (SQP). We utilize the function fgoalattain in MATLAB Optimization Toolbox as the SQP solver. fgoalattain implements SQP by following the argument of Brayton *et al.* [132]. It first updates the Hessian of the Lagrangian function using a quasi-Newton approximation method. This information is then used to solve a quadratic programming subproblem whose solution is used to form a line search direction. An overview of SQP can be found in [133] [134] [94].

6.3 Two-Phase Optimization Algorithm

SQP requires that the objectives must be continuous in the interested search space. As shown in Chapter 3, both $J_1(K) = mean(\sigma^2_{pes})$ and $J_2(K) = std(\sigma^2_{pes})$ have some singular points in search space. Thus it is expected that randomly generated starting points may lead SQP to the local optima of GAP or even nonconvergence. As illustrated in Figure 6.2, starting from a randomly picked point J^r may generate a non-optimal solution J^{*r}. Since the weightings are selected heuristically, starting from sub-optimal Pareto solutions will make $\mathbf{J}^0 + \gamma\mathbf{w}$ more likely to be in the Pareto surface. Therefore, a two-phase optimization algorithm for tuning the fixed-structure control is proposed as follows:

Algorithm 24 (Two-Phase Optimization Algorithm)

1. (Phase-One) Run Algorithm 23 (MOGA) with a moderate n_{gen}.
2. (Phase-Two) Select relative weightings based on the performance of Pareto-optimal solutions of the MOGA.

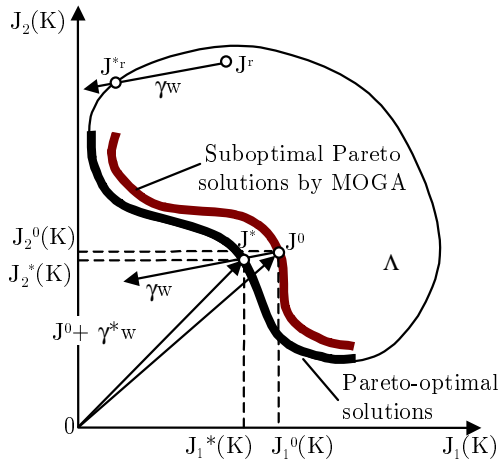

Figure 6.2: When GAP is solved by SQP, starting from the sub-optimal set of MOGA enhances the possibility of reaching Pareto-optimal surface.

3. Reparameterize the original MOP as a nonlinear GAP according to Eq. (6.1).

4. Use Pareto-optimal solutions of the MOGA as initial points.

5. Solve the GAP by using SQP.

6. If all goals are achieved, stop; otherwise adjust weightings based on the degree of under-attainment or over-attainment, and then go to step 4.

Like all weighted-sum strategies, selecting adequate weightings is still the most difficult step. But by using the best evolved solutions of the first stage (MOGA), we not only greatly increase the possibility of finding the true Pareto-optimal solutions, but gain valuable insights into the trade-offs among different objectives. Even so we cannot guarantee that SQP will achieve all goals under the weightings we select. At least, however, we know to what degree the solutions under-achieve or over-achieve our goals, and we can adjust the weights accordingly or pick up the solution that has the best trade-off.

Table 6.1: Two-phase algorithm: MOGA configurations in phase one

Description	Symbol	Value
Number of generations	n_{gen}	30
Number of individuals per generation	n_{pop}	200
Probability of crossover	P_c	1
Probability of mutation	P_m	0.02
Maxium percentage of elitism	$e\%$	40%
Percentage of random immigrants	$r\%$	5%

6.4 Example: Optimization of HDD Controller Parameters

The configuration of the MOGA used in the first phase is presented in Table 6.1. The definition of cost functions are same as Eq. (5.10) and Eq. (5.11). After 3.8 minutes in a AMD 1GHz PC, the MOGA generates a Pareto set with 19 solutions, five of which with the smallest $m_{\sigma^2_{pes}}$ are selected as starting points for phase two, as listed in Table 6.2.

As shown in Figure 6.2, the result of fgoalattain depends on not only the selection of weightings which dictate the search direction, but the initial starting point. The advantage of using the Pareto set of MOGA as starting points for SQP is shown in Table 6.2. The first six starting points are randomly picked and five of six cases fail to deliver a solution that satisfies all goals. The last five are picked from the Pareto-optimal (actually sub-optimal) solutions of the MOGA and all converge to slightly better optimal solutions. The MOGA sub-optimal solutions are very close (evaluated in objective space) to the final optimal solutions. It takes less than 10 seconds for fgoalattain to converge in all cases.

By fine-tuning the relative weightings, we can further improve the quality of solutions. In Table 6.2, the fourth solution by SQP $[0.261, 0.864, 0.335]$ with relatively small $m_{\sigma^2_{pes}}$ and $\sigma_{\sigma^2_{pes}}$, is evolved from the MOGA solution $[0.211, 0.065, 0.916]$ by using relative weights $[1, 1, -1, -1, -1, 1]$. According to Eq. (6.1), the bigger the weight, the larger

Table 6.2: The result of solving GAP by SQP highly depends on starting point

Objectives			$m_{\sigma^2_{pes}}$	$\sigma_{\sigma^2_{pes}}$	GM	PM	ω_o	S_∞	
Unit			$tr\%^2$	$tr\%^2$	dB	deg	Hz	dB	
Goals			<5.2	<2.2	>5.9 <10	>36 <60	>600 <1000	<6.5	
Weights			1	1	-1	-1	-1	1	
	$[k_0,k_1,k_2]$		Simulation results						
	StartingPoint	Solution							
R	[0.2,0.3,0.7]	[0.409,0.636,1.0]	4.84	1.48	2.2	87.3	591	13.1	×
a	[0.5,0.5,0.5]	[0.376,1.0,1.0]	3.50	1.20	1.8	72.0	4084	14.7	×
n	[0.1,0.5,0.5]	[0.368,1.0,0.562]	5.10	1.58	3.2	80.3	595	10.14	×
d	[0.5,-0.5,-0.5]	[0.268,0.876,0.334]	4.86	1.87	6.1	44.7	604	6.1	√
o	[0.9,-0.7,0.3]	[0.384,0.649,1.0]	4.71	1.45	2.7	93.1	474	11.38	×
m	[0.5,0.5,-0.5]	[0.3677,1.0,0.562]	5.10	1.56	3.2	80.3	595	10.14	×
M	[0.361,0.340,0.853]		4.38	1.77	4.6	38.6	768	8.4	
O		[0.229,0.204,0.909]	5.19	1.95	6.5	45.6	600	5.8	√
G	[0.351,0.813,0.415]		4.50	1.88	4.7	37.6	755	7.3	
A		[0.246,0.870 0.274]	5.05	2.02	6.5	41.5	600	5.8	√
	[0.251,0.325,0.735]		5.18	2.10	6.4	37.2	570	5.3	
		[0.272,0.332,0.877]	4.79	1.84	5.9	44.5	617	6.2	√
	[0.211,0.065,0.916]		5.22	2.04	6.1	41.8	643	6.5	
		[0.261,0.864,0.335]	5.05	1.96	6.3	42.7	592	5.8	√
	[0.221,0.192,0.921]		5..48	2.04	6.7	47.8	583	5.6	
		[0.245,0.274,0.870]	5.05	2.02	6.5	41.5	600	5.8	√

Table 6.3: Performance comparison among different controllers.

		Original		MOGA:		Two-Phase:	
		K_{orig}		K_{MOGA}		K_{two}	
				CPU time: 7.3 mins		CPU time: 3.9 mins	
Objective	Unit	Simu.	Exp.	Simu.	Exp.(relative to orig)	Simu.	Exp.(relative to orig)
$m(\sigma^2_{pes})$	$tr\%^2$	5.24	5.22	5.05	5.09(-2.9%)	4.92	4.93(-5.5%)
$\sigma(\sigma^2_{pes})$	$tr\%^2$	2.25	2.31	2.02	2.11(-8.7%)	1.90	1.97(-14.7%)
GM	dB	6.1	6.3	6.5	6.3	6.2	6.1
PM	$degree$	42.1	40.4	41.5	40.4	44.0	43.5
ω_o	Hz	540	544	600	595	602	598
S_∞	dB	5.7	5.5	5.9	5.8	6.0	5.9

the over- or under-attainment of corresponding objective. Therefore, a weighting vector $[10, 10, -1, -1, -1, 1]$ is used to further push down $m_{\sigma^2_{pes}}$ and $\sigma_{\sigma^2_{pes}}$. The resulting solution by fgoalattain is $K_{two} = [0.259, 0.877, 0.313]$. By loading K_{two} into our experimental drives, we compare its performance with that of $K_{MOGA} = [0.246, 0.870, 0.275]$ in the last chapter, as list in Table 6.3. It can be seen that besides the improvement on average performance in terms of $m(\sigma^2_{pes})$, there is a big improvement on the performance robustness over experimental drives in terms of $\sigma(\sigma^2_{pes})$. The reason is that the original controller K_{orig} was not systematically tuned toward minimal performance variation, due to the lack of a good multi-drive model. This is another proof for the effectiveness of the statistical model developed in Chapter 3. Compared with the pure MOGA method, the two-phase algorithm almost cut the total CPU time by half because the gradient-based SQP is much more efficient in the local search.

6.5 Summary

While the gradient-based techniques lack robustness over global non-convex optimization problems and are sensitive to initial points, the MOGA is less efficient than gradient-based methods in local fine-tuning. In light of the above, this chapter developed a two-phase numerical algorithm to optimize the parameter set of fixed-structure controller based on multiple design criteria. In phase one, the MOGA effectively searches the non-convex space and identifies a set of sub-optimal Pareto solutions, which are used as starting points in phase two. In phase two, the SQP rapidly generate a precise optimal solution around every starting point. The trade-offs among design objectives revealed by the MOGA in phase one dictate the selection of weightings in phase two. The simulation and experiment have shown that, compared with the pure MOGA method, the two-phase algorithm is more computationally efficient yet preserves the searching robustness of MOGA.

Chapter 7

Minimize the State Truncation Error

7.1 Introduction

Almost all servo systems of modern HDDs use the digital control structure consisting of analog-to-digital (A/D) converter, fixed-point digital signal processor (DSP) and digital-to-analog (D/A) converter [135]. Compared with floating-point DSPs, fixed-point DSPs are more economical. However, the unavoidable quantization errors of fixed-point DSPs are obstacle to the further performance enhancement of the HDD servo system.

Besides the quantization errors caused by A/D and D/A [136] [25], the quantization errors occurring inside fixed-point DSPs are the *coefficient quantization error* (CQE) which is due to the quantization of coefficient parameters, and the *state truncation error* (STE) which is due to the truncation of state variables [73]. In HDD servo systems, the

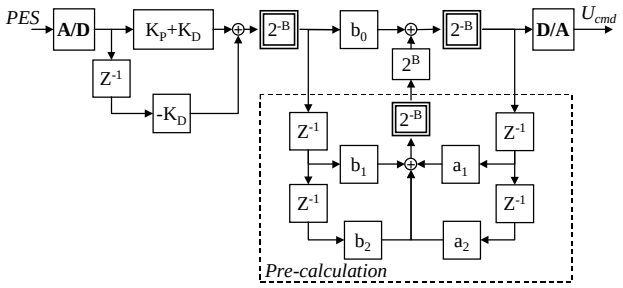

Figure 7.1: Cascade form implementation of track following controller.

CQE is negligible because in order to preserve the precision of coefficient parameters, the corresponding integers stored in fixed-point DSP are usually close to the full wordlength. On the other hand, the state variables may take very small values because of input uncertainty and arithmetic operation. Therefore the state truncation could lead to much larger *relative* error than the coefficient quantization. For this reason, the STE should not be overlooked. In fact, our experiments showed that even for a very good initial design, STEs from a careless DSP implementation could contribute up to 7% of the total energy of the position error signal (PES). In this dissertation, a systematic approach is presented from a servo point of view to minimize the effect of STEs in PES.

7.2 Truncation Noise Propagation

Contrary to the quantization errors of A/D and D/A, the STE depends not only on the hardware resource, e.g. wordlength, but also on the controller implementation structure. As a simple illustration, consider a track-following controller which consists of a PD

regulator

$$G_{pd}(z^{-1}) = K_P + K_D(1 - z^{-1}) \tag{7.1}$$

and a notch filter [28]

$$G_{notch}(z^{-1}) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{7.2}$$

which is added to attenuate the potential suspension resonance mode.

Figure 7.1 shows a *cascade form* implementation of this track-following controller $G_{notch}(z^{-1}) \cdot G_{pd}(z^{-1})$ in fixed-point DSP. The fixed-point implementation requires that all coefficients are scaled by 2^B where B is the standard wordlength. Results are then truncated or rounded into integers and stored in the data memory of DSP. For convenience, these integers are called in the format of Q_B. For example, Q_{16} means that a floating-point number is quantized as a 16-bit integer. Assuming that a double-word accumulator is available, the sum-then-truncate scheme is used in all summation points. Also, all blocks of the notch filter within the dashed line (Figure 7.1) include only prior states which can be calculated in the last clock cycle. This is called *pre-calculation* [137]. It decreases the computational delay between the measurement input PES and the updated control output U_{cmd}, and thus improves the system stability margin. Every double-line box indicates a quantizer, i.e., a scale-back factor 2^{-B} with a truncation or round-off operator $q[\cdot]$. State truncation errors occur at these quantizers.

It is well known that the fixed-point implementation with quantizers is nonlinear. However, under certain condition, it can be approximately treated as a linear floating-point system plus some quantization noise sources [66]. This will make the system much

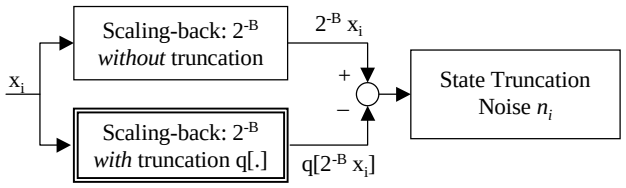

Figure 7.2: Identification of state truncation noise source.

easier to analyze. A cascade controller model as shown in Figure 7.1 is built by using MATLAB Simulink Fixed-point Blockset. To simulate the real scenario, STEs are identified by injecting some segments of measured PES traces into controller input and applying the method shown in Figure 7.2 to every quantizer. The decoupled quantization noise $n_i(k) = K_i x_i(k) - q[K_i x_i(k)]$ is the difference between two outputs, i.e., with and without truncation. A simple statistical data analysis shows that every STE is a stationary white process with amplitude uniformly distributed over [0 1] for truncation or [-0.5 0.5] for round-off, both in the format of Q_B. All STEs are mutually uncorrelated and independent of the input at the large. More precisely, the mean, auto-covariance, and cross-correlation of STEs are respectively given in the format of Q_B as

$$
\begin{aligned}
m_{n_i} = E[n_i(k)] &= \begin{cases} 0; & \text{roundoff} \\ 0.5; & \text{truncation} \end{cases}, \\
E[(n_i(k) - m_{n_i})(n_i(l) - m_{n_i})] &= \begin{cases} \frac{1}{12}; & k = l \\ 0; & k \neq l \end{cases}, \\
E[n_i(k) n_j(l)] &= 0; \quad i \neq j \ .
\end{aligned}
\tag{7.3}
$$

Based on these observations, one can convert every corresponding quantizer in the controller into a floating-point scaling factor 2^{-B}, plus a noise source n_i which is equal to the identified truncation error. The modified block diagram is shown in Figure 7.3. However,

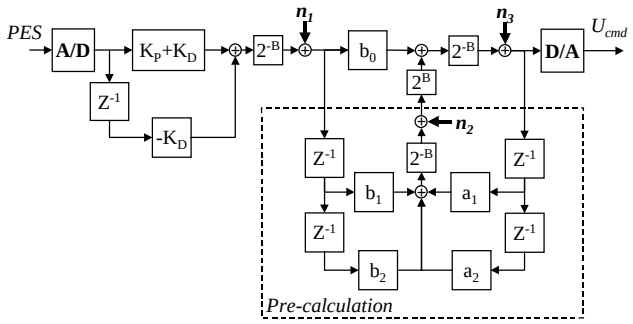

Figure 7.3: Cascade form: linear model plus quantization noise sources.

note that this approximation is true only if the controller input is a complicated wide-band signal [73]. Fortunately, in HDD servo systems, controller input PES is a wide-band random process that makes the above transformation valid.

The analysis above shows that the STEs can be treated as exogenous input noises. For convenience, based on the linear model shown in Figure 7.3, all truncation noise sources are lumped into one noise source n_q as shown in Figure 2.7, which enters the closed-loop system right before D/A and

$$n_q(k) = G_{notch}(z^{-1})n_1(k) + G_{den}(z^{-1})[n_2(k) + n_3(k)] \tag{7.4}$$

where G_{notch} is the transfer function of notch filter and $G_{den} = \frac{1}{1-a_1z^{-1}-a_2z^{-2}}$ which has the same denominator as the notch filter. Finally, referring back to Figure 2.7, the lumped noise n_q relates to PES by

$$PES_{n_q}(k) = G_q(z^{-1})n_q(k) \tag{7.5}$$

where $G_q(z^{-1}) = \frac{P(z^{-1})}{1+P(z^{-1})C(z^{-1})}$ and $P(z^{-1})$ is the discrete version of plant dynamics identified by measurements [138]. Given that the exogenous noise inputs are white noises with

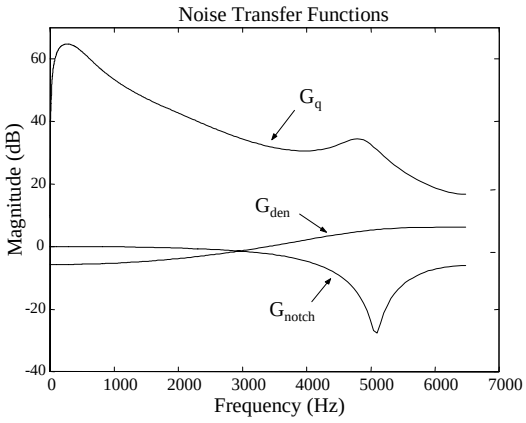

Figure 7.4: Noise transfer functions.

mean m_{n_i} (0.5 for truncation or 0 for round-off) and power spectral density $\Phi_{n_i}(e^{j\omega}) = \sigma^2_{n_i} = \frac{1}{12}$, the mean of PES_{n_q} is

$$m_{PES_{n_q}} = G_q(1)[G_{notch}(1) + 2G_{den}(1)] \cdot m_{n_i} \tag{7.6}$$

and the power spectral density of PES_{n_q} is

$$P_{PES_{n_q}}(\omega) = \left|G_q(e^{j\omega})\right|^2 \left[\begin{array}{c} \left|G_{notch}(e^{j\omega})\right|^2 \\ +2\left|G_{den}(e^{j\omega})\right|^2 \end{array} \right] \sigma^2_{n_1} \\ +2\pi\delta(\omega) \cdot m^2_{PES_{n_q}}. \tag{7.7}$$

From (7.6) and (7.7), the variance of PES_{n_q} can be calculated as

$$\sigma_{PES_{n_q}} = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{PES_{n_q}}(\omega)d\omega - \left|m_{PES_{n_q}}\right|^2. \tag{7.8}$$

As stated above, statistically, all noise sources share the same characteristic: they have the same energy spreading over the whole frequency range up to the Nyquist frequency. So the magnitude shapes of G_{den}, G_{notch} and G_q determine the energy distribution of noise output in the frequency-domain. Figure 7.4 shows one set of typical magnitude plots of

Table 7.1: Variance percentage of state truncation noises in total PES: Cascade form (Simulation)

	$CascadeForm$
$\frac{var[G_q G_{notch} n_1]}{var[PES_{total}]}$	4.5%
$\frac{var[G_q G_{den}(n_2+n_3)]}{var[PES_{total}]}$	2.1%
$\frac{var[PES_{n_q}]}{var[PES_{total}]}$	6.6%

G_{den}, G_{notch} and G_q [138] [28]. In this dissertation, the system has a sampling rate of 12,960 Hz. G_{den} attenuates the low frequency noises by -6 dB up to 1,500 Hz and amplifies noises starting from 3,500 Hz. On the other hand, notch filter G_{notch} has an almost opposite effect: it has a unit gain at the low frequency and gives the high frequency input a great attenuation. Furthermore, since G_q places great emphasis on the low frequency, noise n_1 associating with G_{notch} will map more energy into PES. This can be shown in Figure 7.5 where the power spectral density of two components of n_q, $G_{notch}(z^{-1})n_1(k)$ and $G_{den}(z^{-1})[n_2(k) + n_3(k)]$, are plotted. This plot combined with Figure 7.4 clearly explains why n_1 alone contributes more to PES than n_2 and n_3 together. Furthermore, based on (7.4), (7.5), and previously identified truncation noises ($m_{n_i} = 0$), components of PES_{n_q} can be quantitatively determined through numerical simulation. Then the variance percentage of PES_{n_q} and its components in PES_{total} are calculated based on (7.7) and (7.8), as shown in Table 7.1. In this case, PES_{total} is the same PES used to identify those STEs (Section 3). Surprisingly, as normal as this cascade implementation appears to be, simulation shows that the STEs contribute to 6.6% of the total PES variance.

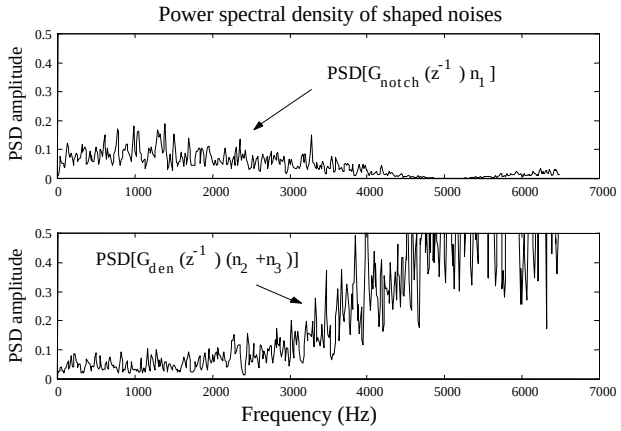

Figure 7.5: Power spectral density of shaped noise sources.

7.3 Implementation Structure Optimization

The above example reveals that STEs are statistically independent of the input and their propagation relates to the controller structure. Here an optimization problem arises: how should the implementation structure be designed to minimize the effect of STEs in PES? From the analysis above, several important guidelines can be drawn as discussed below.

First, try to minimize the number of noise sources, i.e. quantizers, especially the ones going through the loop that has relatively large gains at the low frequency. For example, as shown in Figure 7.6, the *direct IIR form* implementation of $G_{pd}(z^{-1})$ and $G_{notch}(z^{-1})$ will reduce the total number of truncation noise sources by combining the first-order controller and the second-order notch filter as

$$\begin{aligned} G_{all}(z^{-1}) &= G_{pd}(z^{-1})G_{notch}(z^{-1}) \\ &= \frac{N_0 + N_1 z^{-1} + N_2 z^{-2} + N_3 z^{-3}}{1 - D_1 z^{-1} - D_2 z^{-2} - D_3 z^{-3}} \end{aligned} \tag{7.9}$$

where

$$\begin{aligned} N_0 &= (K_P + K_D)b_0, \\ N_1 &= (K_P + K_D)b_1 - K_D b_0, \\ N_2 &= (K_P + K_D)b_2 - K_D b_1, \\ N_3 &= -K_D b_2, \\ D_1 &= a_1, \qquad D_2 = a_2, \qquad D_3 = 0. \end{aligned} \tag{7.10}$$

So the PES due to truncation noises is

$$PES_{n_q^{'}} = G_q(z^{-1}) \frac{1}{1 - D_1 z^{-1} - D_2 z^{-2}} (n_1^{'} + n_2^{'}). \tag{7.11}$$

Apparently, $\frac{1}{1-D_1 z^{-1} - D_2 z^{-2}}$ is equal to G_{den}. So, unlike the cascade form, all state truncation noises in direct IIR form pass through the loop which has attenuation effect at the low frequency. Simulation shows that this makes a big difference on the propagation of these noises: the variance percentage of $PES_{n_q^{'}}$ in PES_{total} is only 2.1% compared to 6.6% in the previous cascade form implementation. However, considering the potential problem of overflow, a direct IIR form with an order higher than four is not recommended.

Another method to attenuate truncation noises is to use a scaling couple, i.e., put the quantizer between a scale-up factor 2^S and a scale-down factor 2^{-S}. For comparison, this method is used to attenuate the most significant noise source n_1 in the previous cascade form example. As shown in Figure 7.7, n_1 is attenuated by 2^{-S} but the overall loop gain is unchanged because the multiplication of scaling couple is one. So the final output due to the STEs is

$$PES_{n_q^{''}}(k) = G_q(z^{-1}) \left\{ \begin{array}{c} \frac{1}{2^S} G_{notch}(z^{-1}) n_1(k) \\ + G_{den}(z^{-1})[n_2(k) + n_3(k)] \end{array} \right\}. \tag{7.12}$$

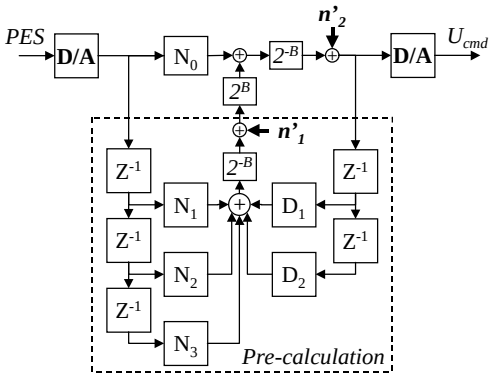

Figure 7.6: Linear model of direct IIR form with noise sources.

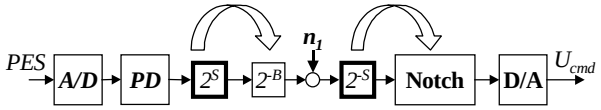

Figure 7.7: Use a scaling couple to attenuate the truncation noises.

Table 7.2: Variance percentage of state truncation noises in total PES with different optimization schemes (Simulation)

	Direct Form	Cascade Form with Scaling Couple			
		S=0	S=1	S=2	S=3
$\frac{var[PES_{n_q}]}{var[PES_{total}]}$	2.1%	6.6%	3.7%	2.5%	2.2%

Implementing the scale-up factor is equivalent to changing the scale-back factor 2^{-B} to 2^{S-B}. Note that directly implementing the scale-down factor (bits right-shift in DSP) will introduce another state truncation noise. The best way to avoid that is to absorb the scale-down factor into the coefficients of the following notch filter. One may argue that this will enlarge the CQEs and trigger the degradation of system performance due to coefficient shift. In fact, a small S is enough to dramatically reduce the effect of STEs while barely affecting the precision of coefficients. For example, when $S = 1$, the noise amplitude is cut by half and the energy is reduced by 75%. On the other hand, even for a very small 6-bit coefficient in the format of Q_{12}, the change of CQE by absorbing this 1-bit scale-down factor is tiny, say from $\frac{0.5}{2^6} = 0.78\%$ to $\frac{0.5}{2^5} = 1.56\%$.

Table 7.2 shows simulation results for different scaling S. Obviously, the biggest improvement is made from without scaling ($S = 0$) to with 1-bit scaling ($S = 1$). The variance percentage of the truncation noise output is almost reduced by half. However, with the increase of S, the profit exponentially decreases. When $S = 3$, the noise output level is already nearly equal to direct IIR form. It suggests that 1-bit or 2-bit scaling should be the most efficient.

Figure 7.8 shows experimental results of loading three different implementation structures into a disk drive embedded DSP respectively. The drive has a recording density of 17,000 TPI and a sampling rate of 12,960 Hz. The FFT components of measured PES are summed every 100 Hz for a better visualization. Only the low frequency components are shown because most energy of the PES is in this range. The experimental results agree with the trend predicted by numerical simulation: the cascade form without scaling couple

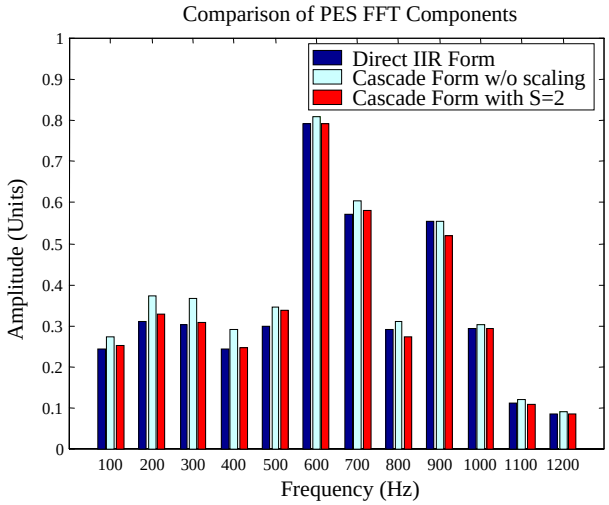

Figure 7.8: FFT components of PES for different implementations (Experiment)

is the worst case ($3\delta_{pes} = 5.02\%$ trackwidth) and the direct form has the best performance ($3\delta_{pes} = 4.79\%$ trackwidth) which is 5% smaller the worst case. Scaling couple with $S = 2$ ($3\delta_{pes} = 4.81\%$ trackwidth) greatly improves the performance of cascade form and almost matches the performance of direct form. Moreover, from the no-scaling cascade case, it can be seen that the influence of STEs peaks around $200 \sim 400$ Hz and decreases as the frequency goes up. This fits the magnitude profile of G_p as shown in Figure 7.4. Finally, note that because the STEs are relatively small PES sources, the resulted improvement on PES could vary depending on the overall PES level.

7.4 Summary

This chapter analyzed the characteristics and propagation mechanism of state truncation errors (STEs) of fixed-point DSP in disk drive servos. For a specified structure,

the effect of STEs can be evaluated even before it is implemented in a real drive. Simulation and experiment in track-following mode showed the effectiveness of two optimization schemes, i.e. changing the implementation structure and applying a scaling couple, in reducing the PES due to STEs. The rules of thumb are to reduce the number of state truncation noise sources first, then apply the scaling couple to unavoidable quantizers.

Chapter 8

Conclusions and Suggestions for Future Research

8.1 Conclusions

This dissertation investigated two major problems associated with the design of the track-following controller for HDDs: 1) modeling of a population of HDDs, and 2) parameter optimization of fixed structure controllers.

An overview of HDD servomechanism was given in Chapter 2, with an emphasis on the detailed breakdown of PES contributors in the track-following mode. Chapter 2 also showed that due to the numerous design constraints and trade-offs, the design of the HDD track-following controller is a multi-objective problem in which the major objective is to minimize the variance of the PES, σ^2_{pes}. Chapter 3 was devoted to building a statistical plant model, which was aimed to provide an accurate and efficient way to predict the time-domain

performance for a population of drives with a given controller. The overall formulation consists of a linear plant model built by a multiple-phase identification process, and a set of random variables describing the PES contributors identified from a large number of drives. The model synthesizes the average performance index $E(\sigma_{pes}^2)$ and performance robustness index $var(\sigma_{pes}^2)$ with a great accuracy (<3%) and a small computational overhead (<0.1s). It was utilized by the non-gradient-based optimization algorithms in later chapters to tune the controller parameters for minimization of PES.

Chapter 4 discussed the parameterization of fixed structure controllers and the optimization techniques for multi-objective design problems. Fixing the structure of controllers is favored in engineering practice, but it makes the tuning of controller parameters a non-convex optimization problem. In this dissertation, the tuning of fixed structure controllers was formulated as a non-convex multi-objective GAP. It was directly solved by a non-gradient-based Pareto solver, MOGA, presented in Chapter 5. The population-based nature of the MOGA enables the evolution of a set of Pareto optimal (or sub-optimal) solutions without requiring weights before optimization. Due to the stochastic nature of the MOGA, it is more robust than conventional optimization methods in a non-convex search space, but it lacks fine-tuning capability in local search. Our experiments showed that the Pareto sets are different from run to run, although they are all close to the global optima in the objective space. In Chapter 6, a two-phase numerical algorithm was developed by taking advantage of the robustness of the MOGA and the efficiency of the gradient-based techniques. The MOGA first performed a global search and found sub-optimal initial points for the second stage, in which a gradient-based nonlinear programming method was used

to perform local search to further enhance the quality of solutions. Comparing with the experimental results of the original controller, the average performance index $E(\sigma_{pes}^2)$ and performance robustness index $var(\sigma_{pes}^2)$ were reduced by 2.9% and 8.7% respectively for the controller tuned by the MOGA, and 5.5% and 14.7% for the controller tuned by the two-phase algorithm. In terms of the computational efficiency, the two-phase algorithm cut the total CPU time of the MOGA almost by half because the gradient-based algorithm is much more efficient than the MOGA in local search. On the other hand, if starting from a randomly generated point in the non-convex search space, the gradient-based algorithm was likely to be trapped in a local optimum or unable to converge.

Chapter 7 provided a systematic analysis of the state truncation errors (STEs) associated with fixed-point DSP implementation. In order to achieve desired performance, the implementation structure of controllers also needs to be taken into account. Simulations and experiments showed that a good implementation can reduce $3\delta_{pes}$ up to 5% solely by minimizing the effect of STEs.

8.2 Suggestions for Future Research

Having a good model or a set of models based on experimental data is very important for the design and evaluation of HDD servo systems. The statistical modeling method proposed in this dissertation can be extended to HDDs with dual-stage actuation, although it is more difficult to identify various PES contributors due to the added complexity of the servo loop. A good estimation algorithm or a new sensing mechanism is required to de-couple the effects of two actuators. An intact method is desirable for it can be easily

scaled to handle a large number of drives, even in production processes.

The MOGA offers almost linear computational overhead relative to the number of tunable variables. This is a very attractive feature for non-convex MOPs with a large number of variables. For example, the controller of dual-stage servo usually involves over twelve tunable variables for which the convex approximation with LMI parameterization is practically unsolvable.

Despite of advances in microelectronics, implementing a digital control algorithm is much more difficult than what it appears to be. The gap between control design and engineering implementation deserves more academic efforts. Much can be learned from the related field of digital signal processing in which the implementation issues have been tackled persistently and systematically from the early years of its development. Chapter 7 of this dissertation discussed the effect of limited wordlength in the implementation stage; a more efficient way, however, is to take into account the implementation imperfection in the *design* stage. For instance modern robust and optimal control designs are robust with respect to plant uncertainty. However, they (especially high-order controllers) can be very sensitive or fragile with respect to the variations of the controller coefficients due to finite wordlength in digital systems or imprecision inherent in analog systems. Studies on the fragility of control design can be found in [139], [140], [141], [142], [143], [144], and the references therein. GAs can be used to solve the problem of non-fragile control design as a robust optimization tool. One can encode the controller variables with the binary coding scheme according to hardware resolution and redefine the performance indexes to penalize fragile solutions. Such an integer programming problem is highly non-convex and extremely

difficult to solve by conventional optimization techniques, while GAs are stochastic rule based and thus are capable of directly searching for the best (or sub-optimal) solution that guarantees a compromise between optimality and fragility in the discretized search space.

Another potential application of GAs is to automatically synthesize control laws. The problem of synthesizing a controller involves the definition of basic functional blocks, the determination of topological interconnections between these blocks, and the optimization of tunable variables of each block. The first two steps which define the control law are usually performed manually in the traditional controller design. On the other hand, GAs have been successfully applied to automatic topology synthesis in other fields for years. Defining a set of basic functional blocks seems to be the most difficult part. A pioneering work of applying GAs to synthesize the control law can be found in Koza *et al.* [145].

In light of the two-phase algorithm discussed in this dissertation, it is meaningful to combine the merits of other optimization techniques with GAs in order to compensate for the deficiency of GAs in terms of local fine-tuning capability. This includes not only using the results of GAs as starting points for the next stage of optimization but incorporating the gradient information into genetic operators. The resulting hybrid GAs such as conjugate GA have been shown to be very effective in solving non-convex optimization problems with regard to the search robustness and solution quality. Readers can find more details on this topic from the references mentioned in Chapter 5 and 6.

Bibliography

[1] N. Schirle, "Key mechanical technologies for hard disk drives," *International Conference on Micromechatronics for Information and Precision Equipment*, July 1997.

[2] J.-Y. Yen, K. Hallamasek, and R. Horowitz, "Track-following controller design for a compound disk drive actuator," *Transactions of the ASME Journal of Dynamic Systems*, vol. 112, pp. 391–402, September 1990.

[3] K. Mori, T. Munemoto, H. Otsuki, Y. Yamaguchi, and K. Akagi, "A dual-stage magnetic disk dirve actuator using a piezoelectric device for a high track density," *IEEE Transactions on Magnetics*, vol. 27, no. 6, pp. 5298–5300, November 1991.

[4] T. Imamura, S. Hasegawa, K. Takaishi, and Y. Mizoshita, "Piezoelectric microactuator compensating for off-track errors in magnetic disk drives," *Advances in Information Storage Systems*, vol. 5, pp. 119–126, 1993.

[5] K. Takaishi, T. Imamura, Y. Mizoshita, S. Hasegawa, T. Ueno, and T. Yamada, "Microactuator control for disk drive," *IEEE Transaction on Magnetics*, vol. 32, no. 3, pp. 1863–1866, 1996.

[6] J. Frandsen, "Disk drive motion control," *Proceedings of the First Annual International MOTORCON '81 Conference*, pp. 312–322, 1981.

[7] R. K. Oswald, "High performance motion control for disk drive computer peripherals," *Drives and Controls International*, vol. 2, no. 5, pp. 33–34, June 1982.

[8] M. James, "Drive control: Disk drives control and interfacing," *Electronics and Computing Monthly*, vol. 5, no. 9, pp. 18–21, September 1985.

[9] H. Hanselmann and W. Moritz, "High-bandwidth control of the head-positioning mechanism in a Winchester disk drive," *IEEE Control Systems Magazine*, vol. 7, no. 5, pp. 15–19, October 1987.

[10] C. D. Mee and E. D. Daniel, *Magnetic Storage Handbook*. McGraw-Hill, 2nd edition ed., 1996.

[11] H. Hanselmann and A. Engelke, "LQG-control of a highly resonant disk drive head positioning actuator," *IEEE Transactions on Industrial Electronics*, vol. 35, no. 1, pp. 100–104, February 1988.

[12] T. C. Chiu, C. Kempf, W. H. Yao, and M. Tomizuka, "Compensation for repeatable and nonrepeatable tracking errors in disk file systems," *Presented at the JSME Conference on Advanced Mechatronics*, 1993.

[13] S. Weerasooriya and D. T. Phan, "Discrete-time LQG/LTR design and modeling of a disk drive actuator tracking servo system," *IEEE Transactions on Industrial Electronics*, vol. 42, pp. 240–247, June 1995.

[14] J. K. Chang and H. T. Ho, "LQG/LTR frequency loop shaping to improve TME budget," *IEEE Transactions on Magnetics*, vol. 35, no. 5, pp. 2280–2282, September 1999.

[15] W. Guo, L. Yuan, L. Wang, G. Guo, T. Huang, B. M. Chen, and T. H. Lee, "Linear quadratic optimal dual-stage servo control systems for hard disk drives," *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, vol. 3, pp. 1405–1410, Sept. 1998.

[16] M. Hirata, K. Z. Liu, T. Mita, and T. Yamaguchi, "Head positioning control of a hard disk drive using H_∞ control theory," *Proceedings of the 31st IEEE Conference on Decision and Control*, vol. 1, pp. 2460–61, December 1992.

[17] M. Hirata, T. Atsumi, A. Murase, and K. Nonami, "Following control of a hard disk drive by using sampled-data H_∞ control," *Proceedings of the 1999 IEEE International Conference on Control Applications*, pp. 182–186, August 1999.

[18] A. Al-Mamum, T. H. Lee, and T. S. Low, "Disk dirve servo controller design using H_∞-LTR," *Proceedings of the 1995 IEEE IECON*, vol. 2, pp. 925–928, 1995.

[19] S. Ibaraki, *Nonconvex Optimization Problems in Optimization and Their Applications*. PhD thesis, University of California at Berkeley, 2000.

[20] M. Steinbuch and M. L. Norg, "Advanced motion control: An industrial perspective," *European Journal of Control*, vol. 4, pp. 278–293, 1998.

[21] D. Y. Abramovitch and T. Yamaguchi, "Discussion on: "advanced motion control: An

industrial perspective" by m. steinbuch and m. l. norg," *European Journal of Control*, vol. 4, pp. 294–297, 1998.

[22] L. ElGhaoui, F. Oustry, and M. Aitrami, "A cone complementarity linearization algorithm for static output-feedback and related problem," *IEEE Transactions on Automatic Control*, vol. 42, pp. 1171–1176, 1997.

[23] H. S. Lee, "Controller optimization for minimum position error signals of hard disk drives," *Proceedings of the 2000 American Control Conference*, pp. 3081–3085, June 2000.

[24] M. Dettori, V. Prodanovic, and C. W. Scherer, "Mixed objectives MIMO control design for a compact disc player," *Proceedings of the American Control Conference*, pp. 1284–1288, June, 1998.

[25] D. Abramovitch, T. Hurst, and D. Henze, "Decomposition of baseline noise sources in hard disk position error signals in disk drives," *Proceedings of the American Control Conference*, vol. 5, pp. 2901–2905, 1997.

[26] D. Abramovitch, T. Hurst, and D. Henze, "The PES pareto method: Uncovering the strata of position error signals in disk drives," *Proceedings of 16th American Control Conference*, vol. 5, pp. 2888–2895, June 1997.

[27] D. Abramovitch, T. Hurst, and D. Henze, "An overview of the pes pareto method for decomposing baseline noise sources in hard disk position error signals," *IEEE Transactions on Magnetics*, vol. 34, pp. 17–23, January 1998.

[28] L. Guo, H. S. Lee, A. Hudson, and S. H. Chen, "A comprehensive time domain simulation tool for hard disk drive TPI prediction and mechanical / servo enhancement," *IEEE Transactions on Magnetics*, vol. 35, pp. 879–884, March 1999.

[29] L. Yi, *Two Degree of Freedom Control for Disk Drive Servo Systems*. PhD thesis, University of California at Berkeley, 2000.

[30] C. N. Nett, D. S. Bernstein, and W. M. Haddad, "Minimal complexity control law synthesis, part 1: Problem formulation and reduction to optimal static output feedback," *Proceeding of American Control Conference*, pp. 2056–2064, 1989.

[31] D. S. Bernstein, W. M. Haddad, and C. N. Nett, "Minimal complexity control law synthesis, part 2: Problem solution via H_2/H_∞ optimal static output feedback," *Proceedings of American Control Conference*, pp. 2506–2511, 1989.

[32] S. Ibaraki and M. Tomizuka, "H_∞ optimization of fixed structure controllers," *Proc. of the International Mechanical Engineering Congress and Exposition*, November 2000.

[33] M. Rotunno and R. A. de Callafon, "Fixed order H_∞ control design for dual-stage hard disk drives," *Proceedings of the 39th IEEE Conference on Decision and Control*, pp. 3118–19, December 2000.

[34] P. Gahinet and P. Apkarian, "A linear matrix inequality approach to H_∞ control," *International Journal of Robust and Nonlinear Control*, vol. 4, no. 4, pp. 421–448, July-August 1994.

[35] P. Fleming, A. Grace, and S. Hancock, "Control system design via multiobjective optimization," *IEE Colloquium on Advances in Optimization*, vol. 45, pp. 1–4, 1989.

[36] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making: Theory and Methodology*. North-Holland, New York, 1983.

[37] Y. L. Chen and C. C. Liu, "Multiobjective VAR planning using the goal-attainment method," *IEE Proceedings of Gener. Transm. Distrib.*, vol. 141, no. 3, pp. 227–232, May 1994.

[38] F. W. Gembicki, *Vector Optimization for Control with Performance and Parameter Sensitivity Indices*. PhD thesis, Case Western Reserve University, Cleveland, Ohio, 1974.

[39] F. W. Gembicki and Y. Haimes, "Approach for performance and sensitivity multiobjective optimization: The goal attainment method," *IEEE Transactions on Automatic Control*, vol. 20, no. 6, pp. 769–771, Decemeber 1975.

[40] A. Charnes and W. W. Cooper, *Management Models and the Industrial Applications of Linear Programming*. New York: John Wiley, 1961.

[41] M. J. Schniederjans, *Goal Programming: Methodology and Applications*. Boston: Kluwer Academic Publishers, 1995.

[42] D. E. Goldberg, *Genetic Algorithms in Search Optimization, and Machine Learning*. Reading, MA: Addison Wesley, 1989.

[43] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs, Second Extended Edition*. New York: Springer-Verlag Berlin Heidelberg, 1992.

[44] K. J. Hunt, "Polynomial LQG and H_∞ controller synthesis: A genetic solution," *Proceedings of IEEE 31st Conference on Decision and Control*, pp. 3604–09, 1992.

[45] R. J. Patton and G. P. Liu, "Robust control design via eigenstructure assignment, genetic algorithms and gradient-based optimization," *IEE Proc. of Control Theory Appl.*, vol. 141, pp. 202–207, May 1994.

[46] R. A. Krohling, "Genetic algorithms for synthesis of mixed H_2/H_∞ fixed-structure controllers," *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, pp. 30–35, September 1998.

[47] B. Chen, Y. Cheng, and C. H. Lee, "A genetic approach to mixed H_2/H_∞ optimal PID control," *IEEE Control Systems*, pp. 51–60, Octobor 1995.

[48] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," *Proceeding of 1st International Conference on Genetic Algorithms*, pp. 93–100, 1985.

[49] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416–423, 1993.

[50] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," *Proceeding of the First ICEC*, pp. 82–87, 1994.

[51] H. Tamaki, H. Kita, and S. Kobayashi, "Multi-objective optimization by genetic algorithm: A review," *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pp. 517–522, May 1996.

[52] Z. Keming, J. Doyle, and K. Glover, *Robust and Optimal Control.* Upper Saddle River, NJ: Prentice Hall, 1996.

[53] S. Skogestad, *Multivariable Feedback Control.* Chichester, West Sussex, England: John Wiley and Sons Ltd., 1996.

[54] F. Jorgensen, *The Complete Handbook of Magnetic Recording.* New York: McGraw-Hill, 1996.

[55] S. Finch and D. Maczarny, "Headerless disk formatting: Making room for more data," *Data Storage Magazine*, pp. 52–54, April 1997.

[56] Texas Instruments, *Servo/MSC Product Line Users's Guide*, 1999.

[57] T. Yamaguchi, K. Usui, and H. Hirai, "Modeling and TMR (track misregistraion) budget design for head positioning of high track-density disk drives," *IEEE Transactions on Magnetics*, vol. 35, no. 2, pp. 892–897, March 1999.

[58] H. W. Bode, *Network Analysis and Feedback Amplifier Design.* New York: Van Nostrand, 1945.

[59] J. S. Freudenberg and D. P. Looze, "Right half planes poles and zeros and design tradeoffs in feedback systems," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 6, pp. 555–565, June 1985.

[60] C. Mohtadi, "Bode's integral theorem for discrete-time systems," *Proceedings of the IEE*, vol. 137, pp. 57–66, March, 1990.

[61] B. Zhu, Y. Sun, L. Guo, and M. Tomizuka, "State truncation error in disk drive servo systems," *Proceedings of American Control Conference*, pp. 3071–3075, June 1999.

[62] L. Guo, "A new disturbance rejection scheme for hard disk drive control," *Proceeding of the American Control Conference*, pp. 1553–1557, June 1998.

[63] R. Ehrlich and D. Curran, "Major HDD TMR sources and projected scaling with TPI," *IEEE Transactions on Magnetics*, vol. 35, pp. 885–91, March 1999.

[64] M. Kiadeh and I. Ahmed, "Servo control of a disk drive using a digital signal processor," *Proceedings of the Sixteenth International Intelligent Motion Conference*, pp. 308–322, 1989.

[65] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems.* Addison-Wesley, 3rd ed., 1994.

[66] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems.* Addison-Wesley, 2nd ed., 1990.

[67] A. Beghi, R. Oboe, P. Capretta, and F. C. Soldavini, "Loop shaping issues in hard disk drive servo system design," *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings*, pp. 828–832, July 2001.

[68] O. L. Funches, "Application of modern control to disk drive actuator positioning," *Proceedings of the 13th IFAC*, vol. 2, pp. 381–386, 1996.

[69] M. Tomizuka, *Class Notes for ME233: Advanced Control System II.* University of California at Berkeley, Spring 1999.

[70] M. Athans, "A tutorial on the LQG/LTR method," *Proceedings of the 1986 American Control Conference*, pp. 1289–1296, 1986.

[71] C. Lin, G. Guo, B. M. Chen, and C. Ko, "Optimal track following control for hard disk drives," *Proceedings of the 6th International Workshop on Advanced Motion Control*, pp. 502–506, April 2000.

[72] B. Zhu, L. Guo, and M. Tomizuka, "A statistical PES model for direct controller optimization towards minimum PES in hard disk drives.." Maxtor Techanical Report, July 2000.

[73] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Prentice-Hall, 1989.

[74] MathWorks Inc., *Signal Processing Toolbox, User's Guide*, 5 ed., 2000.

[75] DSP Technology Inc., *Estimating Transfer Functions with Siglab*, 1995.

[76] M. N. Lee, J. H. Moon, K. B. Jin, and M. J. Chung, "Robust H_∞ control with multiple constraints for the track-following system of an optical disk drive," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 4, pp. 638–645, 1998.

[77] H. Hanselmann, "Implementation of digital controllers-a survey," *Automatica*, vol. 23, no. 1, pp. 7–32, 1987.

[78] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, 1983.

[79] Y. Nesterov and A. Nemirovsky, "Interior point polynomial methods in convex programming: Theory and applications," *Studies in Applied Mathematics*, 1993.

[80] L. E. Ghaoui, *Convex Optimization: Reading Materials for EE227A*. University of California at Berkeley, 2000.

[81] R. Bellman, *Adaptive Control Process: A Guided Tour*. Princeton, NJ: Princeton University Press, 1961.

[82] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[83] J. C. Lagarias, J. Reeds, M. H. Wright, and P. Wright, "Convergence properties of the nelder-mead simplex method in low dimensions," *SIAM Journal of Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[84] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery*, vol. 3, pp. 297–314, 1962.

[85] I. Rechenberg, "Cybernetic solution path of an experimental problem," *Royal Aircraft Establishment, Library Translation No. 1122*, August 1995.

[86] I. Rechenberg, *Evolutionsstrategie: Optimierung Techischer Systeme Nach Prinzipien der Biologischen Evolution*. Stuttgart: Frommann-Holzboog, 1973.

[87] H. P. Schwefel, *Kybernetische Evolution Als Strategie der Experimentellen Forschung in der Strömungstechnik*. PhD thesis, Technische Universität Berlin, 1965.

[88] H. P. Schwefel, *Evolution and Optimum Seeking*. New York: Sixth-Generation Computer Technology Series. Wiley, 1995.

[89] L. J. Fogel, "Toward inductive inference automata," *Proceedings of the International Federation for Information Process Congress*, pp. 395–399, 1962.

[90] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artifical Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[91] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computing*, vol. 1, pp. 1–23, 1993.

[92] T. Bäck and H. P. Schwefel, "Evolutionary computation: An overview," *Proceedings of the Third IEEE Conference on Evolutionary Computation*, pp. 20–29, 1996.

[93] A. Buchanan, *Ethics, Efficiency, and the Market*. Totowa, NJ: Rowman and Allanheld Texts in Philosophy, 1985.

[94] *Optimization Toolbox (Version 2) User's Guide*. The MathWorks Inc., 2000.

[95] A. Charnes, W. W. Cooper, and R. Ferguson, "Optimal estimation of executive compensation by linear programming," *Management Science*, vol. 1, pp. 138–151, 1955.

[96] C. Romero, *Handbook of Critical Issues in Goal Programming*. Oxford: Pergamon Press, 1991.

[97] K. Deb, "Solving goal programming problems using multi-objective genetic algorithms," *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1, pp. 77–84, July 1999.

[98] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application.* New York: Wiley, 1986.

[99] N. Srinivas and K. Deb, "Multiobjective optimization using non-dominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2(3), pp. 221–248, 1994.

[100] C. M. Fonseca and P. J. Fleming, "Multiobjective genetic algorithm made easy: Selection, sharing and mating restriction," *Genetic Algorithm in Engineering Systems: Innovations and Applications*, pp. 45–52, September 1995. Conference Publication No 414.

[101] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handing with evolutionary algorithms - part i: A unified formulation," *IEEE Transactions on System, Man, and Cybernetics- Part A: Systems And Humans*, vol. 28, January 1998.

[102] A. Packard, K. Zhou, P. Pandey, and G. Becker, "A collection of robust control problems leading to LMIs," *Proceedings of Conference on Decision and Control*, 1991.

[103] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, "Linear matrix inequalities in system and control theory," *SIAM*, 1994.

[104] C. Scherer, P. Gahinet, and M. Chilali, "Multiobjective output-feedback control via LMI optimization," *IEEE Transactions on Automatic Control*, vol. 42, pp. 896–911, 1997.

[105] I. Masubuchi, A. Ohara, and N. Suda, "LMI-based controller synthesis: A unified

formulation and solution," *International Journal of Robust and Nonlinear Control*, vol. 8, pp. 669–686, 1998.

[106] C. W. Scherer, "An efficient solution to multi-objective control problems with LMI objectives," *System and Control Letters*, vol. 40, pp. 43–57, 2000.

[107] R. G. Bland, D. Goldfarb, and M. J. Todd, "The ellipsoid method: A survey," *Operations Research*, vol. 29, no. 6, pp. 1039–1091, 1981.

[108] C. J. Wenk and C. H. Knapp, "Parameter optimization in linear systems with arbitrarily constrained controller structure," *IEEE Transactions on Automatic Control*, vol. 25, pp. 495–500, 1980.

[109] M. Mercadal, "Homotopy approach to optimal, linear quadratic, fixed-architechture compensation," *AIAA Journal of Guid. Contr. Dyn.*, vol. 14, pp. 1224–1233, 1991.

[110] R. S. Erwin, A. G. Sparks, and D. S. Bernstein, "Fixed-structure robust controller synthesis via decentralized static output feedback," *Int. Journal of Robust and Nonlinear Control*, vol. 8, pp. 499–522, 1998.

[111] D. S. Bernstein and W. Haddad, "Robust decentralized output feedback: The static controller case," *Proceedings of the Conference on Decision and Control*, pp. 1009–1013, December 1988.

[112] P. Gahinet and P. Apkarian, "An LMI-based parametrization of all H_∞ controllers with applications," *Proceedings of the 32nd Conference on Decision and Control*, pp. 656–661, 1993.

[113] P. Apkarian and H. D. Tuan, "LMI-constrained concave programs in robust control," *Proceedings of the American Control Conference*, pp. 1395–1399, June 1999.

[114] D. Waagen, P. Dercks, and J. McDonnell, "The stochastic direction set algorithm: A hybrid technique for finding function extrema," *Proceedings of the First Annual Conference on Evoluationary Programming*, vol. 1, pp. 35–42, 1992.

[115] H. Myung and J.-H. Kim, "Two-phase evolutionary programming for constrained numerical optimization," *Proceedings of the 35th Conference on Decision and Control*, vol. 5, pp. 4387–88, 1996.

[116] L. L. Show and J. C. Juang, "A two-phase optimization algorithm in controller synthesis," *Proceedings of the American Control Conference*, vol. 2, pp. 914–918, June 2000.

[117] F. Li and R. K. Aggarwal, "Fast and accurate power dispatch using a relaxed genetic algorithm and a local gradient technique," *Expert Systems with Applications*, vol. 19, pp. 159–165, 2000.

[118] J. D. Wolfe and D. F. Chichka, "An efficient design algorithm for optimal fixed structure control," *Proceedings of the 36th Conference on Decision and Control*, pp. 2625–27, December 1997.

[119] T. K. Liu, T. Ishihara, and H. Inooka, "Multiobjective control systems design by genetic algorithms," *Proceedings of the 34th SICE Annual Conference*, pp. 1521–26, July 1995.

[120] R. Krohling, "Design of PID controller for disturbance rejection: A genetic optimization approach," *Proceedings of Genetic Algorithms in Engineering Systems*, pp. 498–503, Sept. 1997.

[121] Y. Mitsukura, T. Yamamoto, and M. Kaneda, "A genetic tuning algorithm of PID parameters," *Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, pp. 923–928, 1997.

[122] E. G. Collins and Y. Zhao, "Design of H_∞-optimal, finite word length, PID controllers," *Proceedings of the American Control Conference*, vol. 2, pp. 902–903, June 2000.

[123] R. A. Krohling and J. P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 78–82, February 2001.

[124] H. Ishibuchi and T. Murata, "Multiobjective genetic local search algorithm," *Proceedings of 3rd ICEC*, pp. 119–124, 1996.

[125] A. J. Chipperfield, N. V. Dakev, P. J. Fleming, and J. F. Whidborne, "Multiobjective robust control using evolutionary algorithms," *IEEE Proc. of the International Conference on Industrial Tech.*, pp. 269–273, 1996.

[126] T. Kawabe and T. Tagami, "A real coded genetic algorithm for matrix inequality design approach of robust PID controller with two degree of freedom," *Proceedings of the 12th IEEE International Symposium on Intelligent Control*, pp. 119–124, July 1997.

[127] H. B. Kamepalli, "The optimal basics for GAs," *IEEE Potentials*, pp. 25–27, April, 2001.

[128] K. D. Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[129] Z. Michalewicz and C. Z. Janikow, "Handling constraints in genetic algorithms," in *Proceeding of 4th International Conference in Genetic Algorithms* (R. K. Belew and L. B. Booker, eds.), (San Mateo, CA), pp. 151–157, Morgan Kaufmann, 1991.

[130] D. E. Goldberg and P. Segrest, "Finite markov chain analysis of genetic algorithm," *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 1–8, 1987.

[131] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.

[132] R. K. Brayton, G. D. Hachtel, S. W. Director, and L. M. Vidigal, "A new algorithm for statistical circuit design based on quasi-newton methods and function splitting (minimax technique)," *Proceedings of the 1979 International Symposium on circuits and systems*, pp. 280–283, 1979.

[133] W. Hock and K. Schittowski, "A comparative performance evalution of 27 nonlinear programming codes," *Computing*, vol. 3, p. 335, 1983.

[134] R. Fletcher, *Practical Methods of Optimization.* John Wiley and Sons, 1980.

[135] D. Williamson, *Digital Control and Implementation.* Prentice-Hall, 1991.

[136] K. Ohkubo, Y. Yagishita, and T. Katayama, "An analysis of quantization errors in digital control systems and its application to a hard disk drive," *Proceedings of The IEEE International Conference on Industrial Technology*, 1996.

[137] Texas Instruments Inc., *TMS320C2XX DSP Design WorkShop*, 1996.

[138] H. S. Lee and L. Guo, "Servo performance prediction for high capacity disk drives," *Proceedings of the American Control Conference*, pp. 3050–3054, June 1998.

[139] L. H. Keel and S. B. Bhattacharyya, "Robust, fragile, or optimal?," *IEEE Transaction Automatic Control*, vol. 42, pp. 1098–1105, 1997.

[140] P. M. Mäkilä, "Comments on "robust, fragile, or optimal?"," *IEEE Transaction on Automatic Control*, vol. 43, pp. 1265–1267, 1998.

[141] A. Jadbabaie, C. T. Abdallah, D. Famularo, and P. Dorato, "Robust, non-fragile and optimal controller design via linear matrix inequalities," *Proceedings of the American Control Conference*, pp. 2842–2846, 1998.

[142] D. Famularo, C. T. Abdallah, A. Jadbabaie, P. Dorato, and W. M. Haddad, "Robust, non-fragile LQ controllers: The static state feedback case," *Proceedings of the American Control Conference*, pp. 1109–1113, 1998.

[143] J. R. Corrado and W. M. Haddad, "Static output feedback controllers for systems with parametric uncertainty and controller gain variation," *Proceedings of the American Control Conference*, pp. 915–919, 1999.

[144] G. H. Yang and J. L. Wang, "Robust, non-fragile $h\infty$ control for linear systems with a

class of controller gain variations," *Proceedings of the American Control Conference*, pp. 897–901, 2000.

[145] J. R. Koza, M. A. Keane, and J. Yu, "Automatic synthesis of both the topology and parameters for a robust controller by means of genetic programming," *Proceedings of the 1999 IEEE International Symposium on Intelligent Control, Intelligent Systems, and Semiotics*, pp. 344–352, 1999.

Appendix A

Codes of Multi-objective Genetic Algorithm

This appendix lists the MATLAB codes of multi-objective genetic algorithms

MOGA.m

```
function[bestK,bestObj,bestPerf,Offspring,Stat] = MOGA(g);
format long;
close all;
numGen  = g.para.numGen;  %total # of generations
numPop  = g.para.numPop;  %total # of populations in one generation
pCross  = g.para.pCross;  %probability of crossover
pMutate = g.para.pMutate; %probability of mutation
ub = g.para.ub';
lb = g.para.lb';
perImmig     = g.para.perImmig;
perEliteUB   = g.para.perEliteUB;
COSTfilename = g.para.COSTfilename;
MOGAflag     = g.para.MOGAflag;
bestK   = [];
bestObj = [];

%---Set up the search range and resolution of parameters
RangeK =[lb ub];
if length(lb)== length(ub)
```

```
    NumK = length(lb);
else
    disp('degrees of upper and lower bound do not agree.');
end
%%=======================================================
%% Randomly generate the first Generation
%%=======================================================
begin_time = cputime;
firstK = diag(ub-lb)*rand(NumK,numPop) + lb*ones(1,numPop);
for iPop = 1:numPop
    firstGen(iPop).K = firstK(:,iPop);% K in column
    firstGen(iPop).parents = [0 0];
    [firstGen(iPop).cost, firstGen(iPop).perf]
        = feval(COSTfilename,firstGen(iPop).K);
end
if (size(firstGen(iPop).cost,2)>1)&(MOGAflag==0)
    disp('MOGA warning:  You have multiple objectives.
            Please use MOGA by setting MOGAflag=1')
    break;
end

[Offspring(1,:), origParetoSet]
        = PARETORANKING(MOGAflag,firstGen,numPop,NumK,RangeK);

paretoSet = DEDUPLICATE(origParetoSet);
SumFit    = sum([Offspring(1,:).fitness]);
Stat.AveFit(1) = SumFit/numPop;
Stat.MaxFit(1) = max([Offspring(1,:).fitness]);

%%=================================================================
%% Main iteration
%% Generate the next generation by three ways:
%% [1] Elitism (e%)
%% [2] Normal selection/crossover/mutation
%% [3] Randomly generate small portation (10%)
%%=================================================================
NumImmig=floor(ceil(numPop*perImmig/100)/2)*2;
NumElite(1)=0;
for jGen=2:numGen
%=============
%-[1] Elitism
%- Directly move the Elite to the next generation
%- ,up to 1/2 population
```

```
NumElite(jGen)=floor(size(paretoSet,2)/2)*2;
if NumElite(jGen)>numPop*perEliteUB/100
    NumElite(jGen)=numPop*perEliteUB/100;
end
for jElite=1:NumElite(jGen)
    Offspring(jGen,jElite)=paretoSet(jElite);
end

%==============
%-[2] generate 90% children by selection/crossover/mutation
for jPop=NumElite(jGen)+1:2:numPop-NumImmig


mate1 = SELECT(SumFit,Offspring(jGen-1,:),4);
distanceK=0.7;
while (distanceK >=0.4)
    mate2 = SELECT(SumFit,Offspring(jGen-1,:),4);
    distanceK=DISTOFK(mate1.K,mate2.K,NumK,RangeK);
end
[child1, child2,crossedFlag]
        = CROSSOVER(mate1,mate2, pCross,'R',NumK );
[child1,mutatedFlag1]
        = MUTATION('R',pMutate, child1, RangeK, NumK,jGen,numGen);
[child2,mutatedFlag2]
        = MUTATION('R',pMutate, child2, RangeK, NumK,jGen,numGen);


if (crossedFlag==1)|(mutatedFlag1==1)
    [child1.cost, child1.perf]
        = feval(COSTfilename,child1.K); %evaluate ## COST ##
else
    child1.cost=mate1.cost;
    child1.perf=mate1.perf;
end

if (crossedFlag==1)|(mutatedFlag2==1)
    [child2.cost, child2.perf]
        = feval(COSTfilename,child2.K); %evaluate ## COST ##
else
    child2.cost=mate2.cost;
    child2.perf=mate2.perf;
end
```

```
child1.rank=[];
child1.fitness=[];
Offspring(jGen,jPop) = child1;
child2.rank=[];
child2.fitness=[];
Offspring(jGen,jPop+1)= child2;
end % end of jPop by selection
%===============
%-[3] Generate 10% children randomly
firstK = diag(ub-lb)*rand(NumK,numPop) + lb*ones(1,numPop);
randK = diag(ub-lb)*rand(NumK,numPop) + lb*ones(1,numPop);

for iPop=numPop-NumImmig+1:numPop
    Offspring(jGen,iPop).K=randK(:,iPop-numPop+NumImmig);
    Offspring(jGen,iPop).parents=[0 0];
    %%--Calculate fitness for the random generated first gene
    [Offspring(jGen,iPop).cost, Offspring(jGen,iPop).perf]
        = feval(COSTfilename,Offspring(jGen,iPop).K); %## COST ##
end %end of jPop by random

%===============
[Offspring(jGen,:), origParetoSet]
        = PARETORANKING(MOGAflag,Offspring(jGen,:),numPop,NumK,RangeK);

paretoSet=DEDUPLICATE(origParetoSet);

SumFit = sum([Offspring(jGen,:).fitness]);

Stat.AveFit(jGen)=SumFit/numPop;
Stat.MaxFit(jGen)=max([Offspring(jGen,:).fitness]);
Stat.MinFit(jGen)=min([Offspring(jGen,:).fitness]);

set(g.run_cputime_t,'String',num2str(cputime-begin_time));drawnow;
set(g.run_avgfit_t,'String',num2str(Stat.AveFit(jGen)));drawnow;

end % end of jGen

%%=====================================
%%     Return the optima
%%=====================================
if MOGAflag==1
%====== MOGA
```

```
    NumOpt=size(paretoSet,2);
    for jout=1:NumOpt
        bestVector(jout)=paretoSet(jout).cost(1);
    end
    [sortBest,sortBestIndex]= sort(bestVector,2);

    for kout=1:NumOpt
        size(paretoSet(sortBestIndex(kout)).K);
        bestK(kout,:)=(paretoSet(sortBestIndex(kout)).K)';
        bestObj(kout,:)=[paretoSet(sortBestIndex(kout)).cost]';
        bestPerf(kout,:)=[paretoSet(sortBestIndex(kout)).perf]';
        save Offspring Offspring bestK bestObj bestPerf
    end

else
%====== GA
    Stat.maxFitAll = 0;
    for iGen = 1:numGen
        for iPop = 1:numPop
            if (Offspring(iGen,iPop).fitness>=Stat.maxFitAll)
            Stat.maxKAll= Offspring(iGen,iPop).K;
            Stat.maxGenAll = iGen;
            Stat.maxPopAll = iPop;
            Stat.maxFitAll = Offspring(iGen,iPop).fitness;
            Stat.bestObj = Offspring(iGen,iPop).cost;
            Stat.bestPerf = Offspring(iGen,iPop).perf;
            end
        end
    end
    best_Fitness =[Stat.maxGenAll Stat.maxPopAll Stat.maxFitAll];
    bestK    = Stat.maxKAll;
    bestObj  = Stat.bestObj;
    bestPerf = Stat.bestPerf;
    save Offspring Offspring bestK bestObj bestPerf
end
```

SELECT.m

```
function Selected = SELECT(SumFit, OldGen, NumTour)
rnPoint = rand(NumTour,1)*SumFit;
cumSumFit=cumsum([OldGen.fitness],2);
for i=1:NumTour
    ind=find(cumSumFit>rnPoint(i));
    iPopS(i) = ind(1);
end
```

```
[maxF,indS]=max([OldGen(iPopS).fitness]);
Selected = OldGen(iPopS(indS));
Selected.ID=iPopS(indS);
```

CROSSOVER.m

```
function [child1,child2,crossedFlag]
                  = CROSSOVER(mate1,mate2,pCross,codingFlag,Kpara)
crossedFlag=FLIP(pCross);
child1.K = mate1.K;
child2.K = mate2.K;
if (crossedFlag==1)
    if codingFlag=='B';
        cLocation = ceil(rand(1)*(Kpara-1))
        child1.K(cLocation+1:Kpara) = mate2.K(cLocation+1:Kpara);
        child2.K(cLocation+1:Kpara) = mate1.K(cLocation+1:Kpara);
    end % end of 'B'
    if codingFlag=='R'
        cLocation = ceil(rand(1)*Kpara);
        a=0.3;
        child1.K(cLocation:Kpara)
        = a*mate1.K(cLocation:Kpara) + (1-a)*mate2.K(cLocation:Kpara);
        child2.K(cLocation:Kpara)
        = (1-a)*mate1.K(cLocation:Kpara) + a*mate2.K(cLocation:Kpara);
    end % end of 'R'
end % end of 'crossover'
child1.parents = [mate1.ID mate2.ID];
child2.parents = [mate1.ID mate2.ID];
```

MUTATION.m

```
function [mutated,mutatedFlag]
        = MUTATION(codingFlag,pMutate, orig, RangeK, Kpara,jGen,NumGen)
indM = 0;
T = 0;
mutated = orig;
mutatedFlag=0;
indM=find(rand(Kpara,1)<pMutate);
if (~isempty(indM))
    mutatedFlag=1;
    if codingFlag == 'B'
        mutated.K(:,indM)=bitcmp(orig.K(:,indM),1);
    end % end of if 'B'

% Mutation for floating point coding, Kpara= NumK
    if codingFlag == 'R'
```

```
        if (FLIP(0.5)== 0)
        y=RangeK(indM,2)-mutated.K(indM);
        mutated.K(indM)=mutated.K(indM)+y*(1-rand(1)^((1-jGen/NumGen)^5));
        else
        y=mutated.K(indM)-RangeK(indM,1);
        mutated.K(indM)=mutated.K(indM)-y*(1-rand(1)^((1-jGen/NumGen)^5));
        end
    end % end of if 'R'
end %end of if 'isempty'
```

PARETORANKING.m

```
function [after, paretoOptima]
           = PARETORANKING(MOGAflag,before,NumPop,NumK,RangeK)
%%=====================================================================
%%sort cost from the least to the largest according to sum(cost)
costMatrix=[before.cost];
costSumVector=sum(costMatrix,1);
if size(costMatrix)~=NumPop;
    disp('MOGA ERROR: cost must be in Mx1 form');
    break;
end
%%=====================================================================
%%use MOGAflag to switch between GA and MOGA
after=before;
if size(costMatrix,1)==1
    MOGAflag=0;
end
if MOGAflag==0;
%%=====================================================================
%%GA : if you have only ONE cost/objective or
%%      combine all cost into one objective.
    after=before(sortedPointer);
    for k=1:NumPop
    after(k).rank = k;
    after(k).fitness = 1/(after(k).cost+1);
    end
    paretoOptima = before(sortedPointer(1));
else
%%=====================================================================
%% beginning of MOGA
    after = before;
    domPointer= sortedPointer;
    RNK=1; % rank
```

```
while (~isempty(domPointer)) % search for next pareto set
undetPointer = domPointer;
domPointer = [];
paretoPointer = [];
ni=0;
%%search for the next element in this set
while (~isempty(undetPointer))
    %%let's look at the undeterminate set again
    pointer = undetPointer;
    undetPointer = [];
    first = pointer(1);
    pointerSize = size(pointer,2);
    if (pointerSize>1)
    dif = costMatrix(:,first)*ones(1,pointerSize)
           - costMatrix(:,pointer(1:pointerSize));
    domByFirstVector = ((max(dif,[],1)<=0)& (min(dif,[],1)<-1e-10));
    domByFirstInd = find(domByFirstVector);
    domPointer = [domPointer pointer(domByFirstInd)];
    undetByFirstInd = find(bitcmp(domByFirstVector,1));
    undetPointer = pointer(undetByFirstInd(2:length(undetByFirstInd)));
    end %end of if
    ni=ni+1; % ni is the size of current Pareto Set
    %%put the first element to non-dominated set
    paretoPointer(ni)=first;
    %%since sum(sorted(first)) is the minimum
end %end of inner while
%%=========================================
%%assign fitness
shareFlag=0;
%%============================
%% w/sharing
if shareFlag==1
    dij = 0;
    h = 1;
    sumS =[];

    %%calculate the mutual distance within the same ranking
    for ii=1:ni
    tempSum = 0;
    for jj = 1:ni
    dij = DISTOFK('B',after(paretoPointer(ii)).K,
    after(paretoPointer(jj)).K,NumK,RangeK);
    tempSum = tempSum + GSTEP(1 - dij/h);
```

```
        end
        sumS(ii)=tempSum;
        end
        sumSumS=sum(sumS);
        %totalFitness=ni*2*(1.5-1)*(NumPop-RNK)/(NumPop-1); % linear
        totalFitness=ni/RNK; % disp('Exponential') % exponetial

        %%fitness sharing
        for asi=1:ni
        after(paretoPointer(asi)).rank=RNK;
        %%shared among same rank
        %after(paretoPointer(asi)).fitness
                        = 2*(1.5-1)*(NumPop-RNK)/(NumPop-1);
        after(paretoPointer(asi)).fitness
                        = totalFitness*sumS(asi)/sumSumS;
        if RNK==1
        paretoOptima(asi)=after(paretoPointer(asi)); % pareto optima
        end
        end
    else
    %%==================
    %% w/o sharing

        for asi=1:ni
        after(paretoPointer(asi)).rank=RNK;
        %after(paretoPointer(asi)).fitness
                            = 2*(1.5-1)*(NumPop-RNK)/(NumPop-1);
        after(paretoPointer(asi)).fitness
                            = 1/(RNK^2); %shared among same rank
        if RNK==1
        paretoOptima(asi)=after(paretoPointer(asi)); % pareto optima
        end
        end % end of 'for asi'
    end % end of shareFlag
    RNK=RNK+1;
    end % end of outer while
end
```

FLIP.m

```
function BS=FLIP(p)
rn=rand(1);
if rn<=p;
    BS=1;
```

```
else
    BS=0;
end
```

GSTEP.m

```
function out=GSTEP(in)
if in<0;
    out=0;
else
    out=in;
end
```

COST.m

```
function y=cost(x)
%This is a user-defined evaluation function
y(1)=x(1)^2 % dummy code
y(2)=x(2)^2 % dummy code
y(3)=x(1)+x(2) % dummy code
```