OPTIMIZATION-BASED TUNING OF A DISCRETE LOOP TRANSFER RECOVERY CONTROLLER WITH HARDWARE IN-THE-LOOP

By

BAN FU CHEE

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1999

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2002

OPTIMIZATION-BASED TUNING OF A DISCRETE
LOOP TRANSFER RECOVERY CONTROLLER
WITH HARDWARE IN-THE-LOOP

Thesis Approved:

[illegible]
Thesis Adviser

prabhakar pagilla

[illegible]

[illegible]
Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincerest appreciation to my major advisor, Dr. Eduardo A. Misawa for his thoughts, supervision and advise throughout this project. Without his guidance and expertise in over seeing the project, I would not have completed this work. I would like to acknowledge and thank my committee members: Dr. Prabhakar R. Pagilla and Dr. Frank W. Chambers for their help and guidance throughout my master studies.

I wish to thank my present and former colleagues at Oklahoma State Advanced Control Laboratory, Dr. Brian O'Dell, Dr. Hanz Richter, Todd Lyle, Thomas Rendon, Chad Stoecker, Li Zhou, and Neusa Maria. They among the finest people that i know and worked with. I would like to thank Dr. Brian O'Dell for proof reading my thesis.

I would like to express my appreciation to my parents for giving me the encouragment to pursue an advanced degree. To my brother and sisters, thank you for all your support throughout my undergraduate and graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Notations used in this thesis include:

A^T := the transpose of matrix A,

$\mathbb{C}^-$:= open left half complex plane, $(z \in \Im | Re[z] < 0)$,

$\mathbb{C}^{\odot}$:= unit disk of the complex plane,

I := identity matrix of size $n \times n$,

$Im[z]$:= complex component of complex number z,

Ω := symmetric positive definite solution of the filter algebraic Ricatti equation,

$\Phi(s) := (sI - A)^{-1}$,

$\Re^{n \times m}$:= set of real matrices with n rows, m columns,

$Re[z]$:= real component of complex number z,

s := Laplacian variable,

Σ_c := quadruple $\Sigma(A, B, C, D)$ constrained by $(\dot{x} = Ax + Bu, y = Cx + Du)$,

Σ_d := quadruple $\Sigma_d(A, B, C, D)$ for $(x(k+1) = Ax(k) + Bu(k), y = Cx(k) + Du(k))$,

T_s := sampling time,

u := vector of control inputs $(u | u \in \Re^{m \times q})$,

x := vector of states $(x | x \in \Re^{n \times p})$,

z := complex number, $(z = a + jb$ for $a, b \in \Re)$.

CHAPTER 1

Introduction

Control systems generally operate in environments where unknown disturbances are present. Such unknown disturbances often take the form of varying forces, torques, voltages, etc., which can be attributed to natural phenomena such as fluctuations in load or random air flow variations. The hard disk drive (HDD) is an engineering application which is subject to a wide spectrum of of disturbances. In recent years, the track density (TPI) and the spindle angular velocities have drastically increased in response to the demand for higher storage capacities and lower average seek times. These changes have caused the head positioning control on the hard disk drives to be more sensitive to a variety of external disturbances.

Loop transfer recovery (LTR) theory has provided a powerful modern compensator design technique. The LTR technique allows for a control system to counteract disturbances at the plant input or output for either single-input/single-output (SISO) or multi-input/multi-output (MIMO) systems, yielding designs that retain the properties associated with optimal control theory.

The LTR theory is based on three main steps as stated [1]. Step 1, formulate the design specifications (i.e., robustness requirement and performance criteria) as restrictions on singular values of the open loop at either the input or the output of the loop transfer function matrix obtained by breaking the control loop at either the input or output of the plant. Step 2, design a target loop using optimal control theory to meet the design specifications of Step 1. Step 3, solve a linear quadratic regulator (LQR) problem for small control weighting to recover the loop shape of the target loop design in Step 2. The feature of the LTR theory is that a single design process combines the observer and controller design. Following the three step design procedure, a compensator (i.e., controller/observer pair) results which meets specific design requirements across a desired range of frequencies.

The LTR theory has been widely applied in designing controllers for the HDD. Hansleman $et\,al.$ [2], successfully used the continuous-time LQG/LTR technique to design a tracking controller for a highly resonant disc drive actuator. Lin $et\,al.$ [3], used the Linear Quadratic Gaussian/Loop Transfer Recovery (LQG/LTR) technique to design a family of controllers that minimizes the position error variance due to disturbance and measurement noise during track following in the hard disc drive. Beghi $et\,al.$ [4], analyzed the performances of discrete-time controllers obtained by means of Linear Quadratic Gaussian (LQG) optimal control theory and the effectiveness of the discrete-time Loop Transfer Recovery (LTR) technique in achieving a satisfactory recovery. The key control design issue was to achieve sufficiently high closed loop bandwidth while granting adequate disturbance rejection in the loop gain crossover frequency region. Chang and Ho [5], used discrete-time Linear Quadratic Gaussian/Loop Transfer Recovery (LQG/LTR) design technique, and some knowledge on disturbances to develop a systematic method for designing a track following controller for hard disc. The controller was implemented on a commercial HDD, and an improvement of 10% in track misregistration (TMR) budget was observed. The discrete-time LQG/LTR was used by Suh [6] to design a dual stage controller for disc drives. A structured approach for tuning an Observer Based Discrete Variable Structure Control (OBDVSC) compensator scheme that combines the OBDVSC and the LTR mechanism was proposed by Lyle, [1]. Lyle [1], showed the OBDVSC with LTR hyperplane design technique on a hard disc drive example.

1.1 Objectives and Motivations

The motivation of this research is to develop a structured method to tune the discrete Linear Quadratic Gaussian/Loop Transfer Recovery controller. Incorporating an optimization technique into the discrete LQG/LTR control yields a structured tuning method. With this technique, the initial discrete LQG/LTR controller can be initially designed with a low bandwidth while optimization technique then searches for gains that produce a controller with higher bandwidth. When the controller gains obtained from simulation were implemented into the hardware, the controller gains needed to be manually tuned to improve the performance. A solution to this problem is to tune the controller through an optimization technique while the hardware is running. This technique saves time and reduces user interaction given that most of the computations are done by the computer, hence the result is a hardware in-the-loop tuning technique.

1.2 Contributions

The main contributions of this research are as follow:

1. The need for a structured tuning method inspired the idea of integrating a discrete-time LQG/LTR controller with an optimization method.

2. An automated tuning method with hardware in-the-loop using a robust controller with optimization reduces the control problem to an unconstrained optimization problem.

3. A comparison of different performance indices used in the optimization-based tuning of a discrete LQG/LTR controller with hardware in-the-loop.

Contributions that are significant to the Advanced Controls Laboratory are as follows:

1. System identification of the hard disc drive manufactured by Conner.

2. Simulation of the discrete-time LQG/LTR controller with optimization-based tuning on the 4^{th} order hard disc drive model.

3. Implementation of the optimization-based tuning of a discrete LQG/LTR controller with hardware in-the-loop using the dSPACE controller board.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 reviews the loop transfer recovery methodology, including the model based compensation (MBC) technique known as LQG/LTR. Chapter 3 gives an overview of the optimization techniques. Chapter 4 introduces the controller design proposed for this thesis. Chapter 5 discusses the experimental setup, on which the proposed controller is implemented. Chapter 6 presents the MATLAB-dSPACE interface libraries which were used to interface with the real-time processor from the MATLAB workspace. Chapter 7 discusses the results obtained from experiments. Chapter 8 provides the conclusion and future research ideas. Appendix A provides the mathematical model of the hard disc drive actuator. Procedure to use MLIB/MTRACE is given in Appendix B. MATLAB m-files used for simulation and experiments are provided in Appendix C and Appendix D.

CHAPTER 2

Loop Transfer Recovery Overview

The theory of loop transfer recovery (LTR) evolved from the search for a dynamic compensator which would provide command following and stability robustness properties. Its development can be traced to the most basic state feedback control techniques, the linear quadratic regulator (LQR), which involves the selection of an "optimal" control gain matrix, K. This control method provides certain inherent benefits, including nominal stability and stability robustness. However, while LQR ensures a stable system, it is not designed for good command following roles.

When state estimation was added to a system, the combined dynamics of this filter loop and the control loop were found to have an interesting structure, called the model based compensator (MBC). This control structure was promising because it allowed for non-zero reference input to the system. Furthermore, the Separation Theorem permitted the gain matrices to be selected using pole placement (which, if the poles had negative real components, ensures nominal stability), Linear Quadratic Gaussian (LQG) techniques (ensuring both nominal and stability robustness), or any other method.

One drawback to the MBC is that, even if stability robustness is guaranteed for the separate controller and filter loops, the same may not be true for the combined MBC/plant structure. However, it was found if LQG techniques were used to select both the gain matrices, the singular values of the overall transfer function would approach those of a simpler target loop.

2.1 LQG/LTR

In recent years, a design procedure called LQG/LTR, originally proposed by Doyle and Stein [7], has gained some prominence. Essentially, LQG/LTR is a two step design procedure. In the first step of design, "loop shaping" is done utilizing a standard state feedback controller, and a target loop transfer function is prescribed in terms of the state feedback gain. One could utilize LQG theory or any other theory to do this. The next step of the design, called the LTR, is to recover the target loop transfer function attainable by a state feedback controller by utilizing only the measurement feedback controller.

The fundamental idea for recovery in the LTR mechanism is accredited to Doyle and Stein [7], although a prior work also by Doyle and Stein [8] alludes to the LTR concept. As discussed by Stein and Athans [9], the objective of the LTR is to shape the target filter loop, $C(sI - A)^{-1}H$, where A, B, and C are the standard state-space form, Equation (2.1) and Equation (2.2),

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.1}$$

$$y(t) = Cx(t) \tag{2.2}$$

and then attempt to recover its singular value loop shapes by properly selecting the recovery gain matrix, F. Therefore, if the "target filter loop" is designed as a Kalman filter and the control gain matrix chosen so as to recover this loop, then the complete control system will exhibit the estimator's stability and robustness properties.

Suppose a continuous linear time invariant (LTI) system, exists such that the $\sum_c(A, B, C)$ is a state space representation of the design plant to be controlled given by Equation (2.1) and Equation (2.2). Figure 2.1 shows a block diagram of a standard feedback configuration

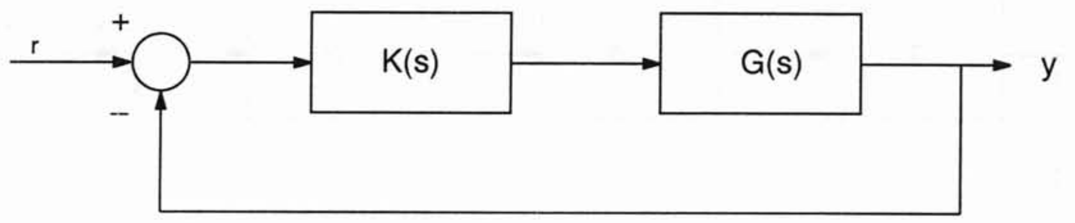

Figure 2.1: Standard Feedback Configuration

with blocks representing controller $K(s)$ and design plant $G(s)$. The design plant must be a minimum phase plant, where for the continuous plant, $G(s)$, for $\sum_c(A, B, C, D)$, all the zeros of $G(s)$ are in the $\mathbb{C}^-$. For a discrete plant, $G(z)$ is said to be minimum phase for $\sum_d(A, B, C, D)$, if the zeros of $G(z)$ are contained in $\mathbb{C}^{\odot}$. The transfer function of the design plant, $G(s)$, defined in Equation (2.3), represents $G(s)$ in Figure 2.1,

$$G(s) = C\Phi(s)B \tag{2.3}$$

such that the $\Phi(s)$ is shown in Equation (2.4).

$$\Phi(s) = (sI - A)^{-1} \tag{2.4}$$

In order to meet the desired stability, robustness, and performance of the compensator, $K(s)$, (i.e., controller/observer pair), the pair (A, B) is assumed to be stabilizable, and the pair (A, C) is assumed detectable [10].

There are three major steps in the LTR methodology, as discussed by Lyle [1].

1. Given a design plant, first characterize design requirements as restrictions on the singular values of an open loop transfer function matrix formed by breaking the control loop in Figure 2.1 at the input or output of the plant $G(s)$.

2. Design a *target loop* to meet specifications outlined by Step 1 with the intention of implementing a compensator composed of state feedback control and a state estimator. For example, breaking the control loop at the plant output of Figure 2.1, the target loop would then be given as Figure 2.2, where the matrix H of appropriate dimension would be called the *filter gain matrix*. By breaking the plant at the output,

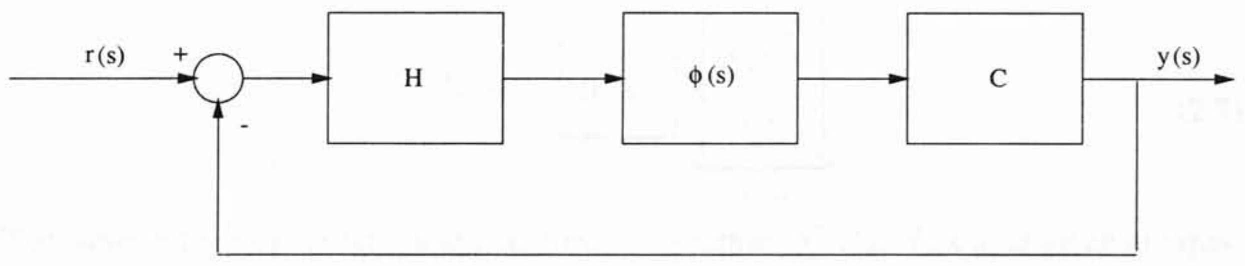

Figure 2.2: Target Loop for Recovery at Plant Output.

H represents an observer gain matrix. The target loop in this situation is referred to as the *target filter loop* because it consists of designing a state observer.

3. The last step is to hold H found in Step 2 constant and to *recover* the target filter loop characteristics in the control loop using compensator $K(s)$ of Figure 2.1 in a special way such that the performance of control loop approximates that of the target loop.

2.2 Target Filter Loop Design Methods

The design requirement for a system is to have zero steady state error. By putting constraint on the number of free integrators and fixing the system to minimum system type, zero tracking error can be achieved. Thus, it is a common practice to augment the state dynamics with a free integrators during the target loop shaping phase of the LQG/LTR design. Suppose the plant dynamics is given by $\sum(A, B, C, D)$, augmenting the system with a free integrator is achieved by the state vector as shown in Equation (2.5),

$$x_{aug} = [u_p \; x]^T \tag{2.5}$$

where $\dot{u}_p(t) = u(t)$ or $u_p(s) = \frac{1}{s}u(s)$. The augmented dynamics is written as shown in Equation (2.6)

$$\underbrace{\begin{bmatrix} \dot{u}_p(t) \\ \dot{x}(t) \end{bmatrix}}_{\dot{x}_{aug}(t)} = \underbrace{\begin{bmatrix} 0 & 0 \\ B & A \end{bmatrix}}_{A_d} \underbrace{\begin{bmatrix} u_p(t) \\ x(t) \end{bmatrix}}_{x_{aug}} + \underbrace{\begin{bmatrix} I \\ 0 \end{bmatrix}}_{B_d} u(t) \tag{2.6}$$

$$y(t) = \underbrace{\begin{bmatrix} 0 & C \end{bmatrix}}_{C_d} \underbrace{\begin{bmatrix} u_p(t) \\ x(t) \end{bmatrix}}_{x_{aug}} \tag{2.7}$$

The target filter loop design for the augmented dynamics $\sum(A_d, B_d, C_d)$ is given in Equation (2.8),

$$G_{kf}(s) = C_d(sI - A_d)^{-1}H \tag{2.8}$$

where solving a continuous time Kalman filter problem often leads to a suitable solution for observer gain matrix, H. Let $\zeta(t)$ be process noise characterized as white, zero mean with an identity (I) noise intensity matrix, and at the same time let $\theta(t)$ be white, zero mean with noise intensity matrix μI. Given the stochastic system using augmented dynamics from Equations (2.6, 2.7),

$$\dot{x}(t) = A_d x(t) + L\zeta(t) \tag{2.9}$$

$$y(t) = C_d x(t) + \theta(t) \tag{2.10}$$

it is known that the solution of the Kalman filtering problem is given by an observer matrix in Equation (2.11)

$$H = (\frac{1}{\mu})\Omega C_d^T \tag{2.11}$$

where Ω is the symmetric positive definite solution of the filter algebraic Ricatti equation (FARE) given in Equation (2.12)

$$0 = A_d\Omega + \Omega A_d^T + LL^T - (\frac{1}{\mu})\Omega C_d^T C_d \Omega \tag{2.12}$$

where $\mu > 0$ and L can be used as design parameters in $C_d(sI - A_d)^{-1}H$, which is nominally stable due to Kalman filtering theory assuming $[A_d, L]$ is stabilizable and $[A_d, C]$ is detectable.

L is chosen from the frequency domain equality (FDE), which can be derived using FARE and $C_d(sI - A_d)^{-1}H$. The gain matrix, applying the suggestion by Athans *et al.*, [11] to partition L into low and high frequency as shown in Equation (2.13),

$$L \triangleq [L_{low}\ L_{high}]^T \tag{2.13}$$

where L_{low} is $[m \times m]$ and L_{high} is $[n \times m]$. The frequency domain equality is given in Equation (2.14).

$$\sigma_i[I + G_{kf}] = \sqrt{1 + (\frac{1}{\mu})\sigma_i^2[C_d(sI - A_d)^{-1}L]} \tag{2.14}$$

The matrix $C_d(sI - A_d)^{-1}L$ in Equation (2.14) reveals conditions sufficient to match singular values at low and high frequencies. Athans [10], suggested that L_{low} and L_{high} are selected as in Equations (2.15, 2.16), given that A^{-1} exists and using μ to govern the crossover frequency (w_{cof}) as in Equation (2.17).

$$L_{low} = -[CA^{-1}B]^{-1} \tag{2.15}$$

$$L_{high} = C^T(CC^T)^{-1} \tag{2.16}$$

$$\mu = (\frac{1}{w_{cof}})^2 \tag{2.17}$$

2.3 Loop Tranfer Recovery

The target filter loop is possible to *recover* using the Linear Quadratic Regulator (LQR). The LQR problem seeks to minimize the performance index given in Equation (2.18),

$$J = \sum_{k=0}^{k=\infty} x^T(k)Qx(k) + \rho Ru^2 \tag{2.18}$$

where the weighting terms $Q = Q^T > 0$ and $R = R^T > 0$. When the control weight R approaches zero, the LQR problem is known as a "cheap control". The following condition

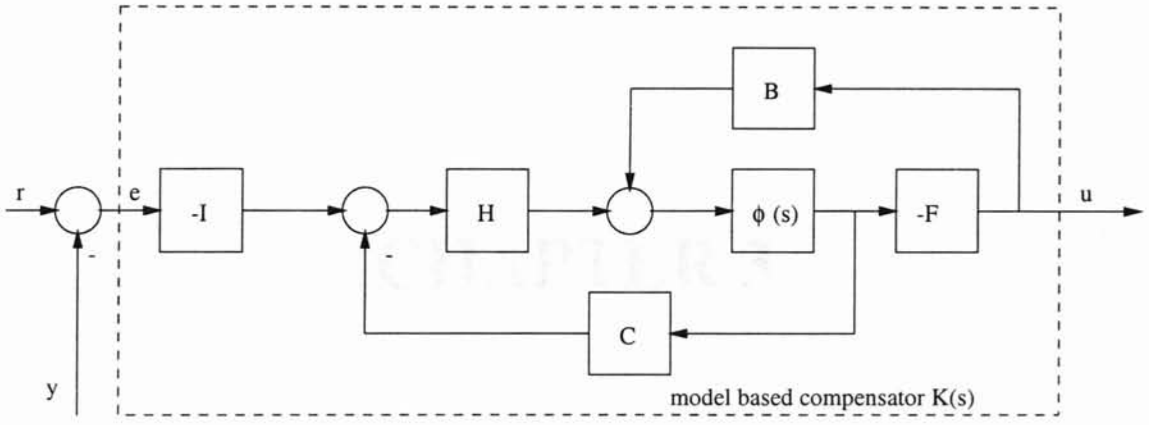

Figure 2.3: Model Based Compensator, $K(s)$.

in Equation (2.19) must hold, in order for the step (3) of the LTR methodology to be valid.

$$lim_{\rho \to 0} \sqrt{\rho} F = WC, \; W^T W = I \tag{2.19}$$

Lemma 2.1 *[7] For recovery at the plant output, given $\Sigma(A, B, C)$ and the continuous-time model based compensator of Figure 2.3 as $K(s) = F(sI - A + BF + HC)^{-1}H$ where F is the recovery gain matrix and H is the observer gain matrix. $G(s) = C(sI - A)^{-1}B$ is minimum phase, if conditions $1 \to 3$ are valid, where*

1. $Re[\lambda_i(A - BF)] < 0$
2. $Re[\lambda_i(A - HC)] < 0$ *and*
3. $lim_{\rho \to 0} \sqrt{\rho} F = WC, \; W^T W = I$

then

$$lim_{\rho \to 0} K(s) = [C(sI - A)^{-1}B]^{-1}[C(sI - A)^{-1}H] \tag{2.20}$$

For recovery at the plant output, **Lemma 2.1** suggests as $\rho \to 0$ for minimum phase systems, the LTR mechanism replaces the design plant dynamics with the dynamics of the target loop. The main LTR result for recovery at the output of the design plant is given by Doyle *et. al.* [7].

CHAPTER 3

Optimization-based Tuning of LQG/LTR Controller

Optimization is a powerful tool for design of controllers. The method is conceptually simple. A controller structure with a few parameters is specified. Specifications are expressed as inequalities of functions of the parameters. The specification that is most important is chosen as the function to optimize. There are several pitfalls when using optimization. Care must be exercised when formulating criteria and constraints. Otherwise, a criterion will indeed be optimal but the controller may still be unsuitable because of neglected constraints. Another difficulty is that the cost function may have many local minima. A third is that the computation required maybe excessive. Numerical problems may also arise.

Three popular optimization criteria are the integral of absolute error (IAE), the integral of time multiplied by the absolute error (ITAE), and the integral of the square of the error (ISE).

3.1 Unconstrained Nonlinear Optimization

Although a wide spectrum of methods exists for unconstrained optimization, the methods can be categorized in terms of the derivative information that is or is not used. Search methods that use only function evaluations are suitable for problems that are very nonlinear or have a number of discontinuities. Gradient methods are generally more efficient when the function to be minimized is continuous in its first derivative. Higher order methods, such as Newton's method, are suitable when second order information, using numerical differentiation is computationally expensive.

3.1.1 Quasi-Newton Methods

Among the methods that use gradient information, the most popular are the quasi-Newton methods. These methods build up curvature information at each iteration to formulate a quadratic model as shown in Equation (3.1),

$$\min_{x} \frac{1}{2} x^T H x + c^T x + b \tag{3.1}$$

where the Hessian matrix, H, is a positive definite symmetric matrix, c is a constant vector, and b is a constant. The optimal solution for this problem occurs when the partial derivatives of x go to zero, *i.e.*,

$$\nabla f(x^*) = Hx^* + c = 0 \tag{3.2}$$

The optimal solution point, x^*, can be written as

$$x^* = -H^{-1}c \tag{3.3}$$

Newton-type methods (as opposed to quasi-Newton methods) calculate H directly and proceed in a direction of steepest descent to locate the minimum after a number of iterations. Calculating H numerically involves a large amount of computation. Quasi-Newton methods avoid this by using the observed behavior of $f(x)$ and $\nabla f(x)$ to build up curvature information and form an approximation to H using an appropriate updating technique.

A large number of Hessian updating methods have been developed. MATLAB Optimization Toolbox [12], uses the formula of Broyden [13], Fletcher [14], Goldfarb [15], and Shanno [16] (BFGS) method. BFGS is thought to be the most effective for use in a general purpose method. The formula given by BFGS is shown in Equation (3.4),

$$H_{k+1} = H_k + \frac{q_k q_k^T}{q_k^T s_k} - \frac{H_k^T s_k^T s_k H_k}{s_k^T H_k s_k} \tag{3.4}$$

where

$s_k = x_k + 1^{-x_k}$

and

$q_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

As a starting point, H_0 can be set to any symmetric positive definite matrix, for example, the identity matrix I.

3.1.2 Unconstrained Optimization Method

MATLAB Optimization Toolbox's unconstrained nonlinear optimization method was used to perform the optimization routine. The optimization command is given by the MATLAB command

```
>> [X] =FMINUNC(FUN,X0)
```

This FMINUNC command uses a quasi-Newton algorithm. The algorithm consist of two phases:

1. Determination of a direction of search (Hessian update)
2. Line search procedures

Implementation details of the two phases is discussed in [12].

3.2 Performance Index

A performance index, as stated by Dorf [17], is a quantitative measure of the performance of a system and is chosen so that emphasis is given to the important system specifications. Modern control theory assumes that the system engineer can specify quantitatively the required system performance. Then the performance index can be calculated or measured and used to evaluate the system's performance. A quantitative measure of the performance of a system is necessary for the operation of modern adaptive control systems, for automatic parameter optimization of a control system, and for the design of optimum systems.

A system is considered an optimum control system when the system parameters are adjusted so that the index reaches an extremum value, often a minimum value. A performance index must be a number that is always positive or zero in order to be considered useful. The best system is defined as the system that minimizes this index.

Three performance indexes will be considered:

- Integral of time multiplied by absolute value of the error, ITAE
- Integral of the square of error, ISE
- Integral of absolute magnitude of error, IAE

1. The ITAE index, proposed by Graham and Lathrop [18], is shown in Equation (3.5). This performance index is designated the integral of time multiplied by absolute error, ITAE. The performance index reduces the contribution of large initial error to the value of the performance integral, while emphasizing errors occurring later in the response.

$$ITAE = \int_0^T t|e(t)|dt \tag{3.5}$$

   T is a finite time chosen arbitrarily so that the integral approaches a steady-state value.

2. The ISE index, which is the integral of the square of the error, ISE is defined in Equation (3.6). The squared error in this performance index is mathematically convenient for analytical and computational purposes [17]

$$ISE = \int_0^T e^2(t)dt \tag{3.6}$$

   T is a finite time chosen arbitrarily so that the integral approaches a steady-state value.

3. The IAE index, which is the integral of the absolute magnitude of error, IAE is defined in Equation (3.7). The difference between this performance index compared to the performance index of ITAE is that, time weighting is not included.

$$IAE = \int_0^T |e(t)|dt \tag{3.7}$$

   T is a finite time chosen arbitrarily so that the integral approaches a steady-state value.

CHAPTER 4

Application of LQG/LTR Control with Optimization-based Tuning on Disc Drive

This chapter investigates the optimization-based tuning of a discrete LQG/LTR controller for a disc drive. A symbolic 3^{rd} order disc drive model developed using MATLAB System Identification Toolbox is used as the design plant. Next, the compensator design technique encompassing both target filter loop and loop transfer recovery design are covered. Followed by, simulation of a discrete LQG/LTR controller with optimization-based tuning on a disc drive model.

4.1 Disk Drive Model

It is known that the mathematical model of the each disc drive differs from one another. In order to run experiments on a chosen disc drive, a mathematical model of the disk drive must be developed. With a known mathematical model of the disc drive, one can do simulations on the controller before implementing the controller on actual hardware.

A 3^{rd} order mathematical model which maps from current to velocity was developed.

The model is estimated using MATLAB System Identification Toolbox's Estimation of Parametric Models. Equations (4.1)-(4.4) are the state-space form $\sum_{3^{rd}}(A,B,C,D)$ of the 3^{rd} order disc drive obtained using MATLAB. Equation (4.5) is the discrete transfer function of the 3^{rd} order disk drive model with a sampling time of $T_s = 1 \cdot 10^{-5}$ seconds. The bode diagram in Figure 4.1 shows the 3^{rd} order disc drive model with the first resonance mode.

$$A_{3^{rd}} = \begin{bmatrix} 2.9860 & -1.4916 & 0.4986 \\ 2.0 & 0 & 0 \\ 0 & 1.0 & 0 \end{bmatrix} \tag{4.1}$$

$$B_{3^{rd}} = \begin{bmatrix} 0.5 & 0 & 0 \end{bmatrix}^T \tag{4.2}$$

$$C_{3^{rd}} = \begin{bmatrix} 0.2250 & -0.0820 & 0.0274 \end{bmatrix} \tag{4.3}$$

$$D_{3^{rd}} = [0.0275] \tag{4.4}$$

$$G_{3^{rd}} = \frac{0.0275z^3 + 0.03038z^2}{z^3 - 2.986z^2 + 2.983z - 0.9971} \tag{4.5}$$

The goal of this controller design is to control the position of the actuator. The 3^{rd} order model is cascaded with an integrator to give the output of the plant in position. The final design model is a 4^{th} order mathematical model that maps from current to position. The discrete-time state-space representation $\sum_d(A,B,C,D)$ is given in Equations (4.6)-(4.9) with sampling time $T_s = 1 \cdot 10^{-5}$ seconds. The discrete-time transfer function of the 4^{th} order model is given in Equation (4.10). The bode plot of the 4^{th} order disk drive model is in Figure 4.2.

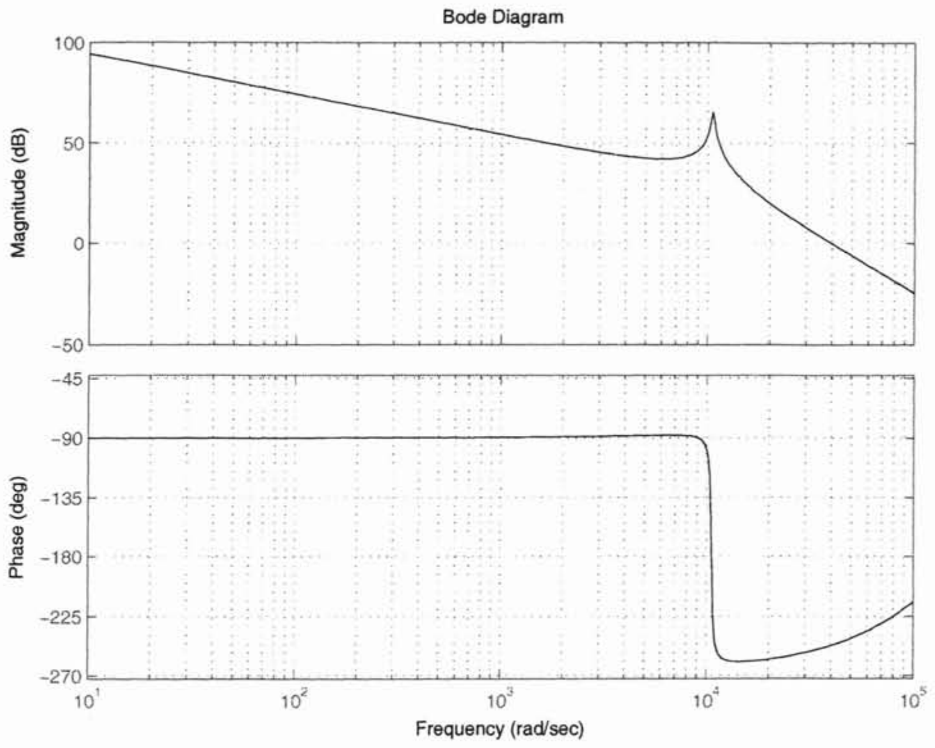

Figure 4.1: Bode Plot of the 3^{rd} Order Disk Drive Model

$$A = \begin{bmatrix} 3.9860 & -1.4923 & 9.9506 \cdot 10^{-1} & -4.9856 \cdot 10^{-1} \\ 4.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 5.0 \cdot 10^{-1} & 0 \end{bmatrix} \tag{4.6}$$

$$B = \begin{bmatrix} 2.5 \cdot 10^{-1} & 0 & 0 & 0 \end{bmatrix}^T \tag{4.7}$$

$$C = \begin{bmatrix} 1.1 \cdot 10^{-1} & 3.0382 \cdot 10^{-2} & 0 & 0 \end{bmatrix} \tag{4.8}$$

$$D = [0] \tag{4.9}$$

$$G = \frac{0.0275z^3 + 0.03038z^2}{z^4 - 3.986z^3 + 5.969z^2 - 3.98z + 0.9971} \tag{4.10}$$

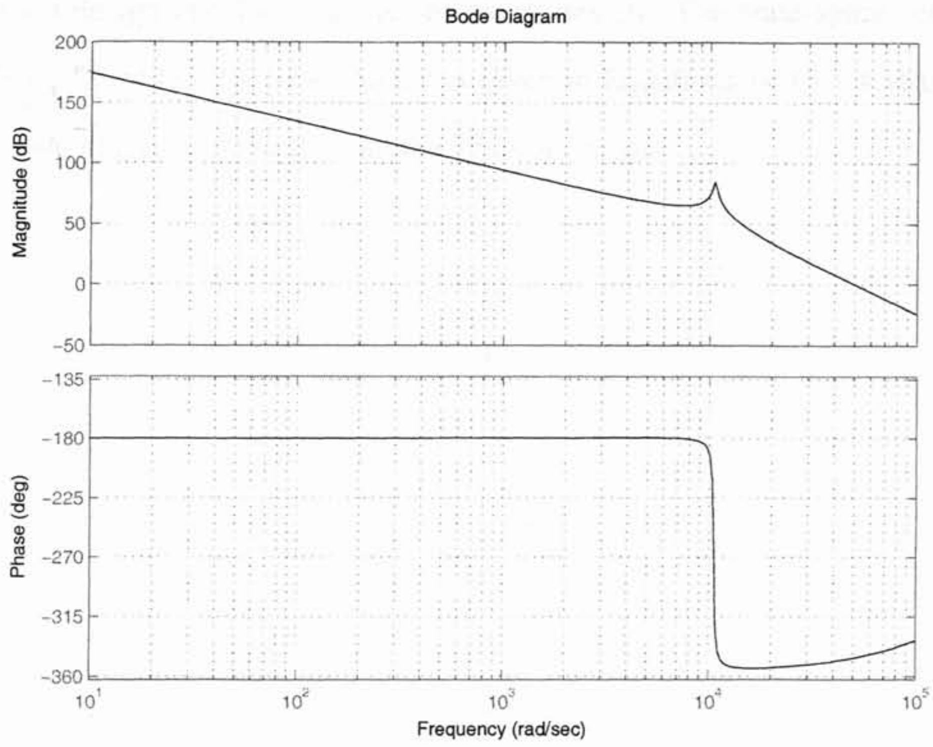

Figure 4.2: Bode Plot of the 4th Order Disk Drive Model

The continuous-time 4th order state-space representation $\sum_c(A,B,C,D)$ is given in Equations (4.11)-(4.14).

$$A_c = \begin{bmatrix} -2.8836\cdot 10^2 & -6.8354\cdot 10^3 & 0 & 0 \\ 1.6384\cdot 10^4 & 0 & 0 & 0 \\ 0 & 8.1920\cdot 10^3 & 0 & 0 \\ 0 & 0 & 8.1920\cdot 10^3 & 0 \end{bmatrix} \tag{4.11}$$

$$B_c = \begin{bmatrix} 2048 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.12}$$

$$C_c = \begin{bmatrix} -3.3714\cdot 10^{-1} & -4.5471 & 1.0047\cdot 10^2 & 2.5814\cdot 10^3 \end{bmatrix} \tag{4.13}$$

$$D = [0] \tag{4.14}$$

By using the 4th order disc drive model as the plant model, a simple double integrator model is designed by matching the gain on the bode diagram. This 2nd order system is used for the target loop design and loop transfer recovery design. The state-space representation $\sum_{2^{nd}}(A, B, C, D)$ of the 2^{nd} order model is given in Equations (4.15)-(4.18). Figure 4.3 shows the Bode diagram of the 2^{nd} and the 4^{th} order model.

$$A_{2^{nd}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{4.15}$$

$$B_{2^{nd}} = \begin{bmatrix} 0 & 0.75 \cdot 10^{11} \end{bmatrix}^T \tag{4.16}$$

$$C_{2^{nd}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{4.17}$$

$$D_{2^{nd}} = [0] \tag{4.18}$$

4.2 Target Filter Loop Design

A double integrator system with gain is used to design the target filter loop. This 2^{nd} order system has an A matrix which is singular. The target filter loop design suggested by [10], as shown in Equation (2.15) and Equation (2.16), is not applicable. This is because the A matrix is not invertable. The Kalman filter method is used to design the target filter loop. The target filter loop is designed for a cross-over frequency of 1000 rad/sec. The MATLAB command

```
>>[H,P,E] = LQE(A,G,C,Q,R)
```

is used to produce the desired target filter loop, where A and C are matrices for the 2^{nd} order designed model. G and Q are are identity matrices and $R = \mu$. By setting $R = 0.000001$, the observer gain matrix H is found using MATLAB LQE command. Equation (4.19)

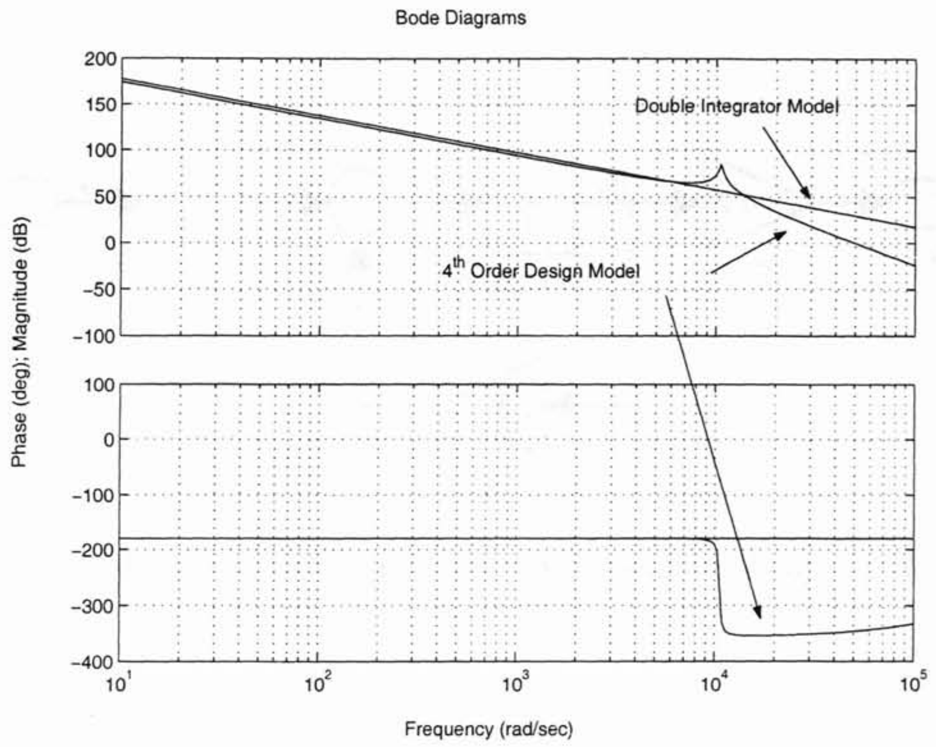

Figure 4.3: Model Matching of the 2^{nd} Order Model with 4^{th} Order Disk Drive Model

shows the observer gain matrix H. Using MATLAB's `linmod.m` and `sigma.m` commands in conjunction with the Simulink diagram shown in Figure 4.4, the frequency response of the target filter loop can be estimated.

```
>> [A,B,C,D]=LINMOD('SYS')
```

```
>> SIGMA(SYS,{WMIN,WMAX})
```

Figure 4.5 shows the frequency response of the design target filter loop. Figure 4.5 show that a cross-over frequency of 1000 rad/sec will result in a settling time of $T_s = 5$ milliseconds.

$$H = \begin{bmatrix} 1001 \\ 1000 \end{bmatrix} \tag{4.19}$$

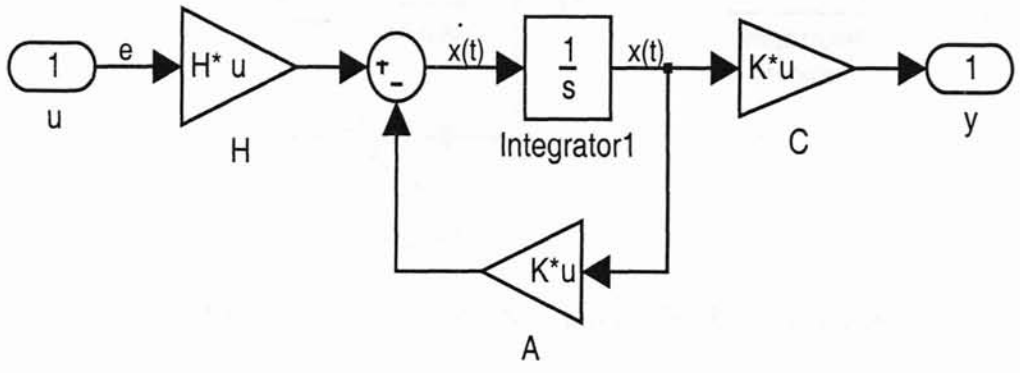

Figure 4.4: Simulink Block Diagram of Target Filter Loop, $C\Phi H$.

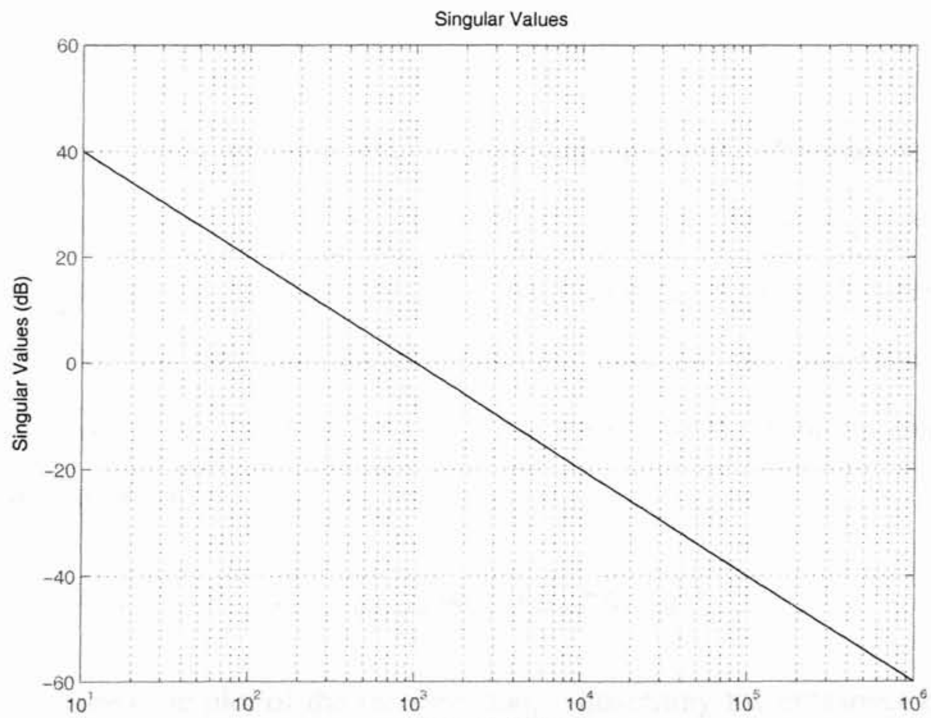

Figure 4.5: Frequency Response of the Target Filter Loop.

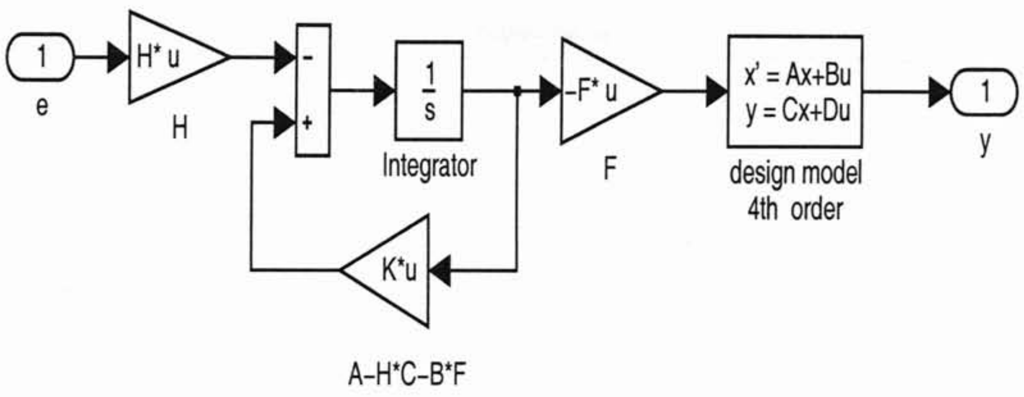

Figure 4.6: Simulink Block Diagram of the Recovery Loop.

4.3 Loop Transfer Recovery Design

The next part of the controller design is to design the recovery loop. The recovery loop is designed using the linear quadratic regulator method in Chapter 2. The MATLAB command

```
[F,S,E] = LQR(A,B,Q,R)
```

produces the recovery loop, where A and B are matrices of the 2^{nd} order design model. Let $Q = C^T C$ and $R = \rho I$, where I is an identity matrix. By setting $\rho = 0.001$, the recovery gain matrix F is found using MATLAB's LQR command. The result is given in Equation (4.20). The frequency of the recovery loop can be plot using the `linmod.m` and `sigma.m` command from MATLAB in conjunction with the Simulink diagram of the target loop in Figure 4.6,

$$F = \begin{bmatrix} 31.62086 & 2.90376 \cdot 10^{-5} \end{bmatrix} \tag{4.20}$$

Figure 4.7 shows the plot of the recovery loop, illustrating the crossover frequency at 1000 rad/sec. Figure 4.8 shows the recovery loop converge to the target filter loop.

The target filter loop and the recovery loop are both designed in the continuous-time domain. Then, the controller, either in state-space form or in transfer function form, is converted into discrete-time using MATLAB's `c2d.m` command as shown below.

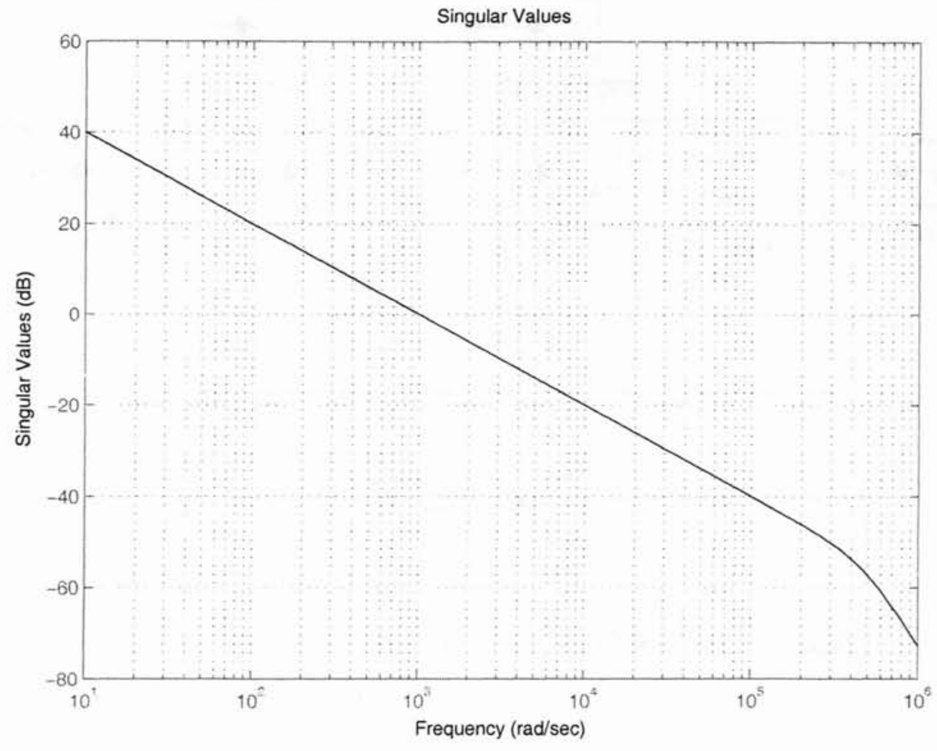

Figure 4.7: Frequency Response of the Recovery Loop.

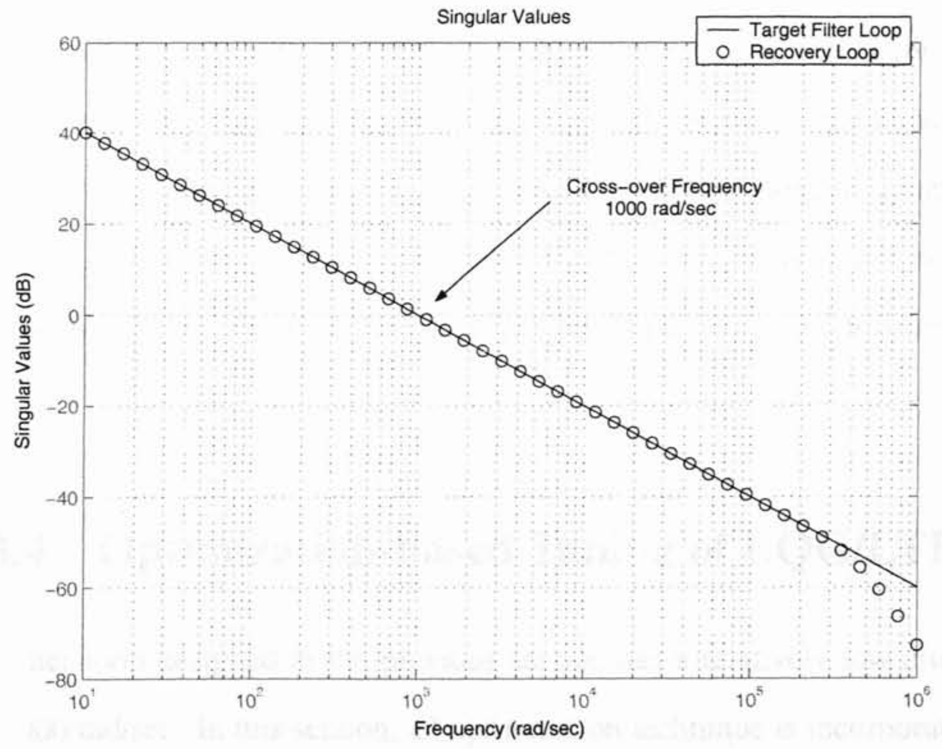

Figure 4.8: Frequency Response of Target Filter Loop and Recovery Loop.

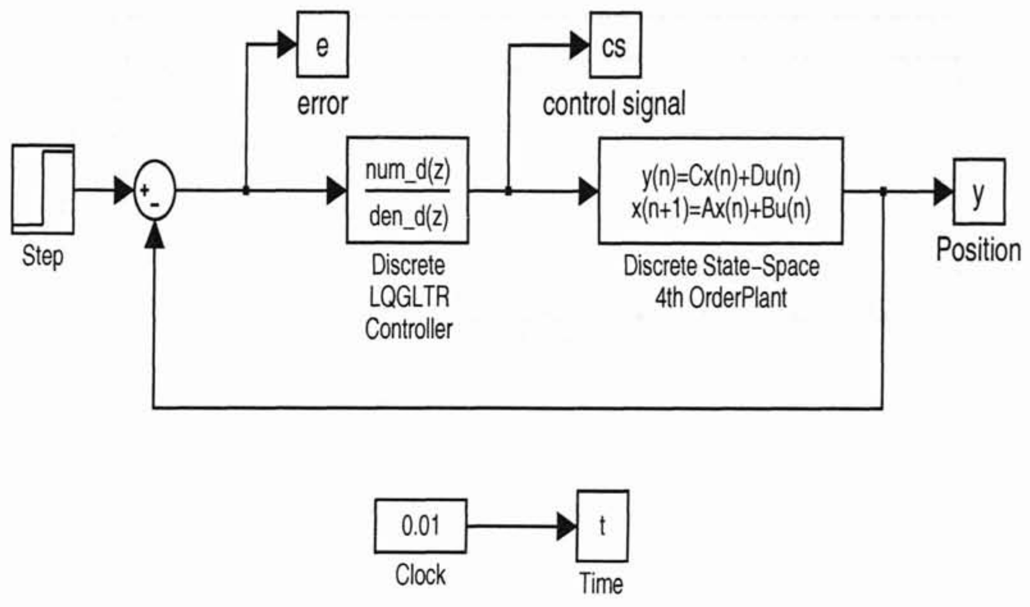

Figure 4.9: Block Diagram of LQG/LTR Compensator.

```
>> SYSD = C2D(SYSC,Ts,METHOD)
```

The discrete LQG/LTR controller found is simulated with a discrete 4^{th} order HDD plant as shown in Figure 4.9. Figure 4.10 shows the simulation with a reference step input of 1 μm. The settling time, $T_s = 0.005$ seconds, corresponds to the target loop crossover frequency of 1000 rad/sec. Since the controller is 2^{nd} order and the simulation plant is 4^{th} order, the oscillation in Figure 4.10 is caused by the unmodelled dynamics of the simulation plant. The frequency of the oscillation in Figure 4.10 is 10473 rad/sec. This frequency matches the resonant frequency of the simulation plant which is around 10200 rad/sec.

4.4 Optimization-based Tuning of LQG/LTR

The target filter loop designed in the previous section has a relatively low crossover frequency of 1000 rad/sec. In this section, an optimization technique is incorporated into the LQG/LTR design to allow the optimization tool to search for the optimum target filter loop and recovery loop.

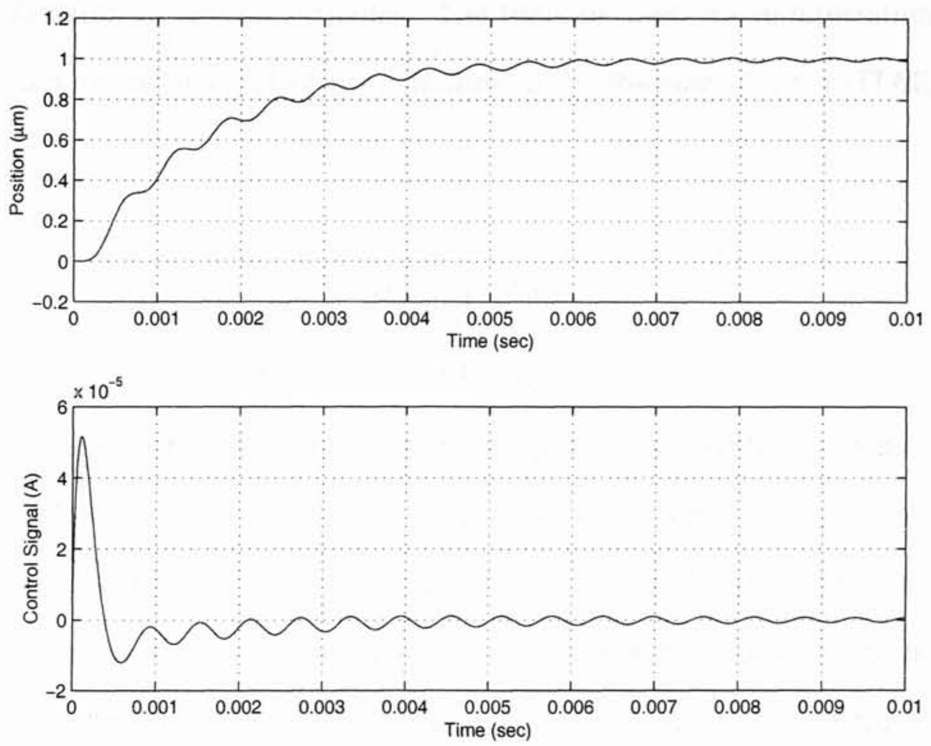

Figure 4.10: Simulation Result of $1\mu m$ Step Response.

MATLAB Optimization Toolbox's unconstrained nonlinear optimization technique is used to perform the search. This method incorporates the LQG/LTR controller design with the unconstrained nonlinear optimization method. The unconstrained nonlinear optimization command can be found in MATLAB `FMINUNC.m`. The `FMINUNC.m` command is shown below,

```
>> X=FMINUNC(FUN,X0,OPTIONS)
```

In this design, the optimization tool will search two variables: μ, which corresponds to the target filter loop, and ρ, which corresponds to the recovery loop. Q and R from the LQR and LQE command must be positive definite matrices, therefore the multipliers, μ and ρ have to be positive numbers. Artificial constraint is imposed by squaring the multipliers, μ and ρ. The search parameters are defined as μ^2 and ρ^2. Since there are no constraints to be dealt with, the unconstrained nonlinear optimization method may be used.

The main function of the unconstrained nonlinear optimization is to find the minimum of a scalar function of several variables. The function used for minimization is the performance index of the integral of time multiplied by absolute of error (ITAE), Dorf and Bishop, [17].

$$ITAE = \int_0^T t|e(t)|dt \tag{4.21}$$

Equation (4.21) is the performance index of ITAE.

Figure 4.11 shows the implementation of the optimization method for simulation. First, the input signal, the 2nd order controller, and the 4th order simulation plant are developed in Simulink blocks. The controller gains designed in Section 4.3 are used as the initial controller gains. The m-files used to compute new controller gains and optimization method is located in the "Optimization Script" in Figure 4.11. There are two m-files for the optimization simulation. The `lqg_ltr.m` file is a local file that contains the target filter loop, recovery loop and the performance index. The `run_lqgltr.m` is the file that runs the optimization technique. The m-files can be found in Appendix C. The initial $\mu^2 = 0.000001$ and $\rho^2 = 0.001$ are set at the value that produced cross-over frequency of 1000 rad/sec. During simulation, the error signal is used to compute the performance index. Optimization method searches for μ^2 and ρ^2 to compute new controller gains. The new controller gains are implemented into the "Controller Gain" block in Figure 4.11. The optimization method will continue to compute new controller gains until an optimum result is computed.

The observer gain or the target filter gain, H, shown in Equation (4.22) is the result for the unconstrained nonlinear optimization method. The unconstrained nonlinear optimization computed $\mu^2 = 0.000167^2$ to produce the target filter gain, H.

$$H = \begin{bmatrix} 5992.03766 \\ 5991.03774 \end{bmatrix} \tag{4.22}$$

The recovery gain, F, in Equation (4.23), is the resulting gain with unconstrained nonlinear optimization. The ρ^2 computed to produce the recovery gain, F, is $\rho^2 = 4316^2$.

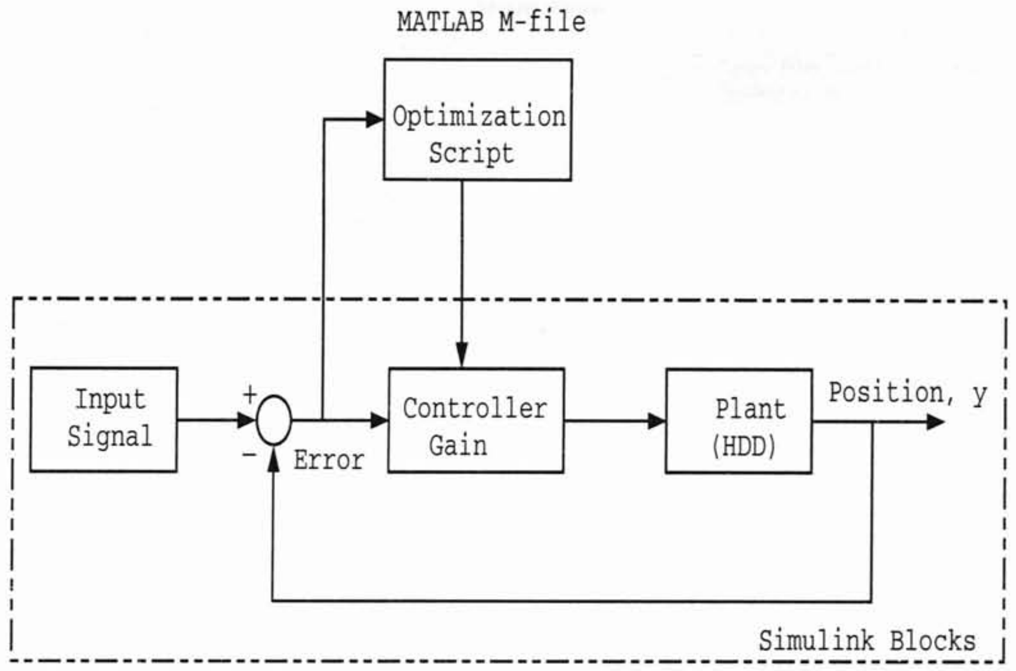

Figure 4.11: Block Diagram Shows The Implementation of Optimization Method.

$$F = \begin{bmatrix} 2.31678 \cdot 10^{-4} & 7.86008 \cdot 10^{-8} \end{bmatrix} \tag{4.23}$$

The performance index found by this optimization method is $f = 6.71001 \cdot 10^{-7}$. Figure 4.12 show the frequency response of target filter loop and loop transfer recovery. The algorithm, optimized the target filter loop to a cross-over frequency of 6000 rad/sec and the the recovery loop cross-over frequency to 2000 rad/sec.

Figure 4.13 shows the simulation result of the optimized discrete LQG/LTR. The response shows improvement in rise time and settling time. The settling time, defined as the point at which the trajectory stays within 10% of its final value, was measured as 2.5 ms. The oscillation in the response is caused by the unmodelled dynamics of the system. The frequency of the oscillation matches the resonant frequency of the simulation plant.

The optimization method converges to a local minimum as shown by the performance index, ITAE. The range of μ^2 and ρ^2 that were tested were from $\mu^2 = 0.00001$ to $\mu^2 = 1$ and $\rho^2 = 0.001$ to $\rho^2 = 4500^2$. This range of μ^2 and ρ^2 were used as the initial values

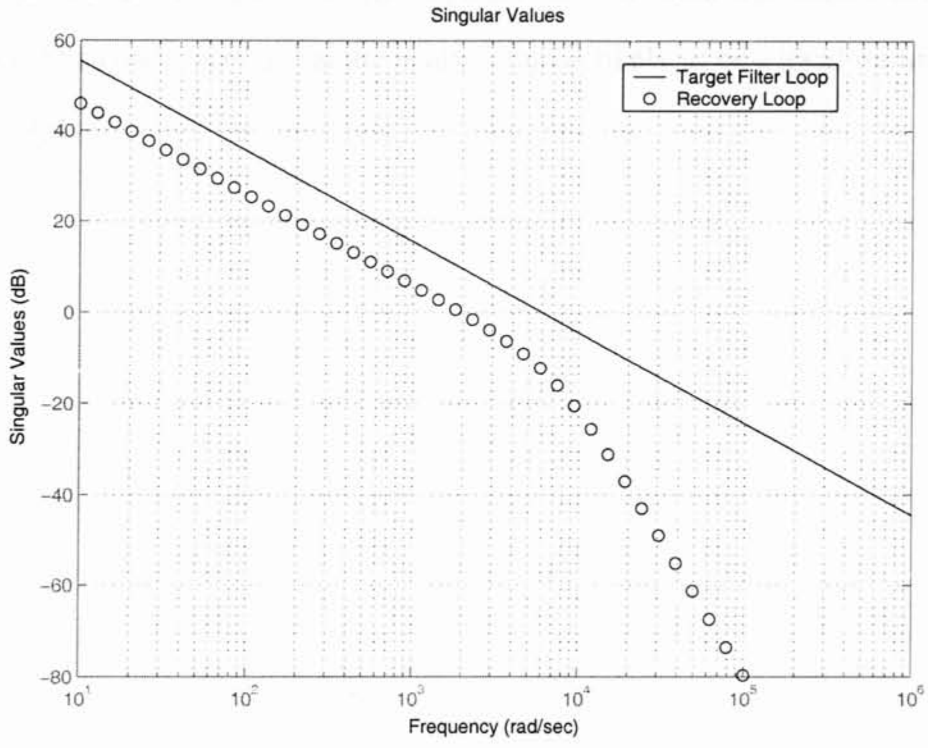

Figure 4.12: Frequency Response of Target Filter Loop and Recovery Loop.

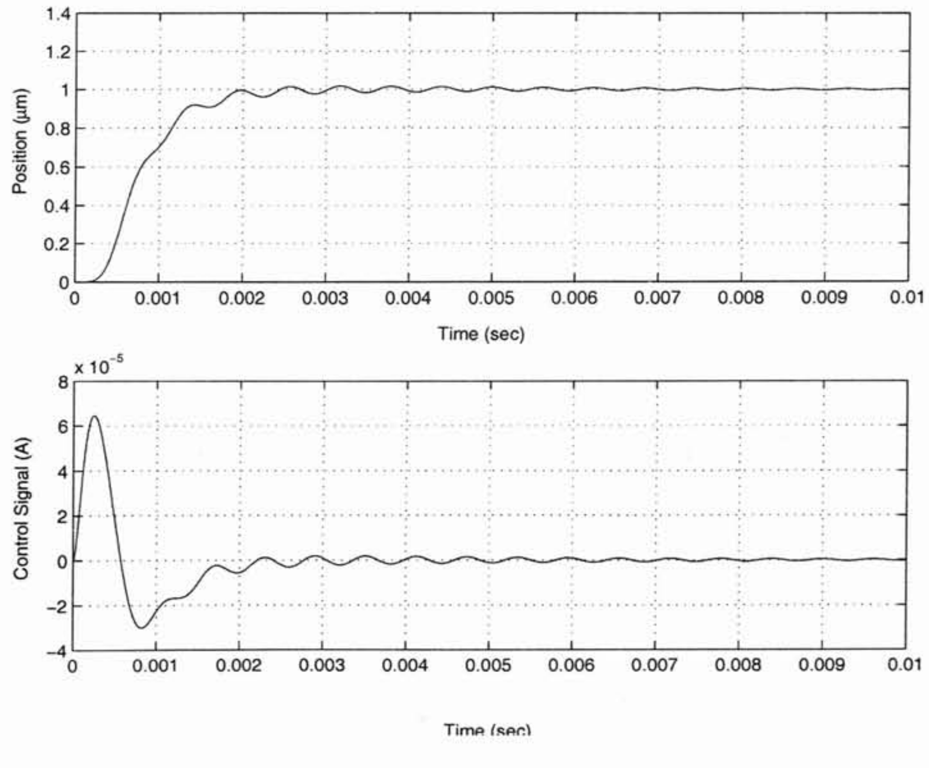

Figure 4.13: Simulation Result of $1\mu m$ Step Response.

for the simulation. By using MATLAB's optimization method, the result converges to the same region. The conclusion is that the result found is likely to be a local minimum for this range of μ^2 and ρ^2.

CHAPTER 5

Experimental Setup

This chapter gives a detailed description of the experimental setup on which the designed controller was tested. The equipment used to setup the experiment is shown in Table 5.1. The disc drive used for the setup was an old drive manufactured by Conner. The model is $CP3000$ and the series is $E59JKA$. Since the focus of this research was to control the actuator arm, the disc and the cover were removed for convenience. The Polytec laser doppler vibrometer (LDV) consists of controller OFV-3001 and sensor head OFV-303. The main purpose of the LDV was to feedback the position and velocity signal of the actuator. A Kepco power amplifier with a maximum output of 2 Amps was used to supply current to drive the disc drive. DS1104 PPC controller board from dSPACE was used to interface between the real system and computer. This controller board has a frequency range of 100 KHz. A Lecroy 1 GHz digital oscilloscope was used to take measurements. The software used for the real-time control was MATLAB real-time workshop. The tests were performed on a Newport vibration isolation table to minimize external disturbances. The computer used for this experiment has a Pentium II 450 MHz processor.

Equipments For Disk Drive Research
An open disk drive
Kepco Power Supply/ Amplifier
Polytec Laser Doppler Vibrometer (LDV)
DS1104 PPC Controller Board (dSPACE)
1GHz Lecroy Oscilloscope
MATLAB Real-Time Workshop
Vibration Isolation Table
Pentium II 450 MHz computer

Table 5.1: Equipment for the Disk Drive Experiment

5.1 Hardware

Figure 5.1 shows the block diagram of the hardware architecture of the experimental setup. In standard operation, the controller is designed in the host computer using MATLAB and SIMULINK Real-Time Workshop. The controller is built and downloaded on to the DS1104 PPC controller board. The DS1104 controller board sends a signal to the Kepco power amplifier, which supplies the current to the voice coil motor (VCM). The VCM controls the position of the actuator. The Polytec laser doppler vibrometer (LDV) measures the position and velocity of the actuator and provides feedback to the DS1104 PPC controller board. The Lecroy digital oscilloscope is used to measure the signals of interest.

5.1.1 Polytec Laser Doppler Vibrometer (LDV)

The Polytec vibrometer is an instrument for non-contact measurement of surface vibrations based on laser interferometry [19]. The vibrometer consists of the controller OFV-3001 and the sensor head OFV-512.

The beam of a helium neon laser is focused on the object under investigation, scattered

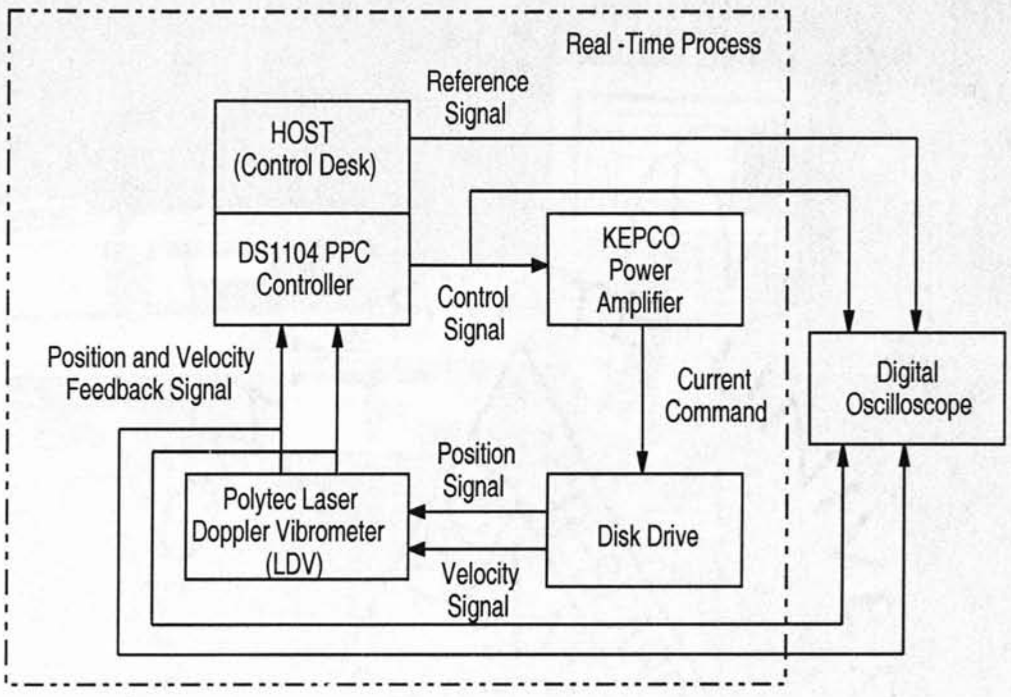

Figure 5.1: Hardware Architecture of the Experimental Setup.

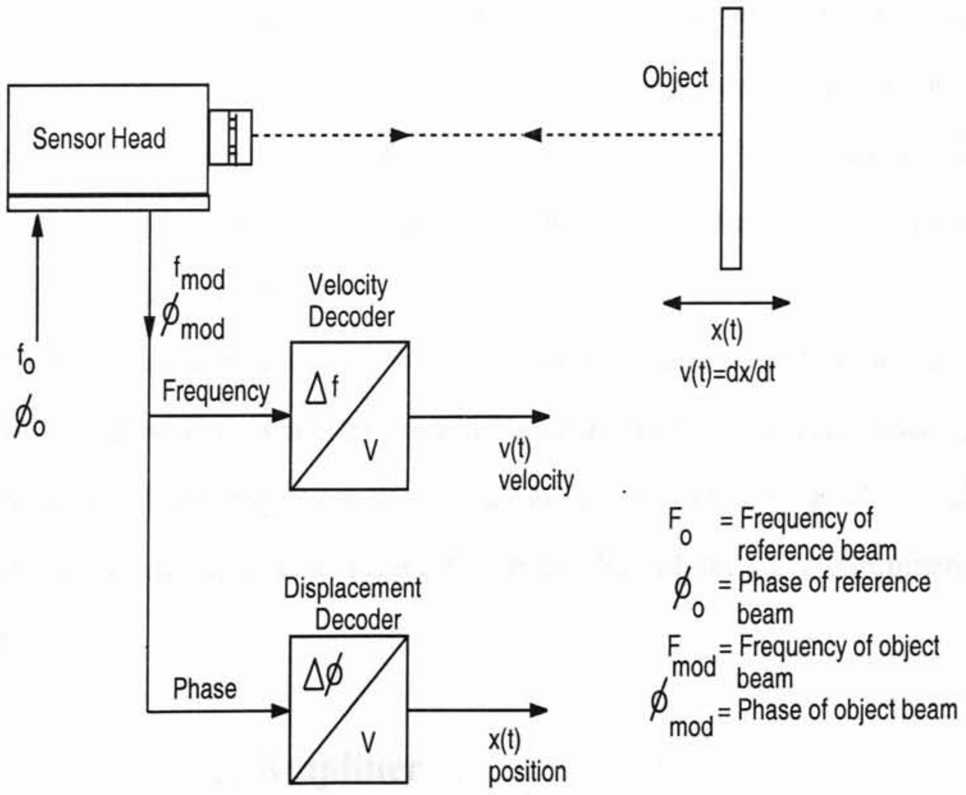

Figure 5.2: Vibrometer Signals.

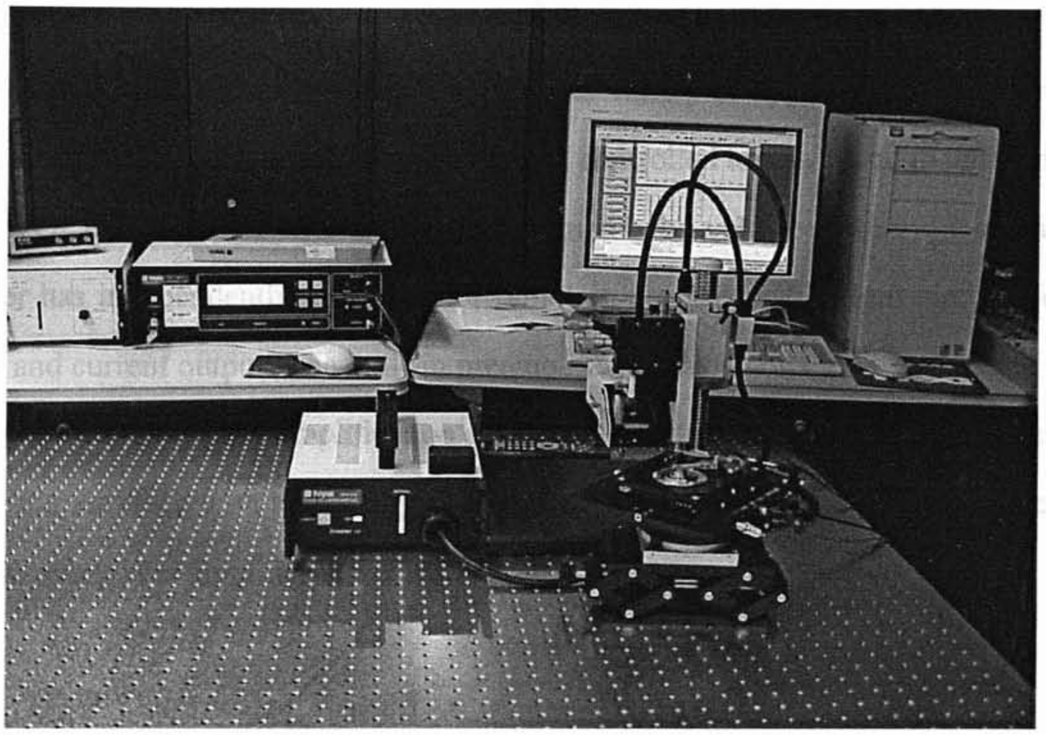

Figure 5.3: Polytec LDV in the Experimental Setup.

back from there and coupled into the interferometer in the sensor head. The interferometer in Figure 5.2 compares the phase, ϕ_{mod}, and the frequency, f_{mod}, of the object beam and with those of the internal reference beam, ϕ_o and f_o. The frequency difference is proportional to the instantaneous velocity and the phase difference is proportional to the instantaneous position of the object.

The signal is decoded using the velocity decoder and the position decoder. Two voltage signals are generated which are proportional to the instantaneous velocity and to the instantaneous position (displacement) of the object, respectively. Both signals are externally available for measurement. Figure 5.3 shows the Polytec LDV equipment used in the experiment.

5.1.2 Kepco Power Amplifier

The Kepco power amplifier model BOP $50-2M$ amplifies the controller output to a level which is capable of driving the voice coil motor. The amplifier has direct current (dc) range of $\pm 50V$ and $\pm 2A$. The Kepco power amplifier is a bipolar operational power (BOP) supply/amplifier, which can be used for in a great variety of applications. As a precision voltage or current source, the BOP output can be controlled locally through the front panel bipolar VOLTAGE or CURRENT controls or remotely by voltage and current signals. The amplifier has independently adjustable or remotely programmable limit circuits for both voltage and current output. The built-in preamplifiers for the voltage as well as the current channel of the BOP permit amplification of the control signal to the required amplitude and provide the interface with high and low impedance signal sources. A detailed description on the Kepco power amplifier is in [20].

5.1.3 DSP Controller Board

THE DSP board model DS1104 PPC controller board is from dSPACE. This type of board is specifically designed for development of high-speed multivariable digital controllers and real-time applications in various fields. It is a complete real-time system based on a 603 PowerPC processor running at 250 MHz. For advanced I/O purposes, the board includes a slave-DSP subsystem based on the Texas Instruments TMS320F240 DSP micro-controller. A detailed description about the board is available in [21].

Information on the DS1104 PPC controller board:

- 603 PowerPC at 250MHz
- 2 MB local SRAM
- 32 MB global DRAM
- 1 16-bit ADC with four multiplexed input signals
- 4 12-bit parallel ADC with one one input signal each
- 8 14-bit parallel DAC

- Incremental encoder interface (2 inputs)
- 1 bit I/O unit with 20-bit I/O
- Serial interface

5.1.4 Disc Drive

The disc drive used for the experiment is produced by Conner. The model of the disc drive is $CP3000$ and the series number is $E59JKA$. For the purpose of this experiment, an open disk drive with the disc and the cover removed is used. The object to be controlled is the actuator arm. The Polytec LDV shines the laser at the tip of the actuator arm where the read/write head is located. Figure 5.4 shows the open disc drive that is used for the experiment. The function of the voice coil motor is to control the position of the actuator arm. The read/write head which is located at the tip of the arm read and write information onto the magnetic disc. Figure 5.5 shows the flexible printed circuit. The flexible printed circuit creates a one directional force on the disc drive actuator. Figure 5.6 shows the whole setup of the experiment.

5.2 Software

5.2.1 MATLAB Real-Time Workshop

MATLAB real-time workshop provided by MathWorks is the final piece in the design process. MATLAB real-time workshop provides a real-time development environment. The real-time workshop is the direct path from system design to hardware implementation. The MATLAB real-time workshop supports the execution of dynamic system models on hardware by automatically converting models to code and providing model-based debugging support. It is well suited for accelerating the development of simulations and embedded real-time applications [23].

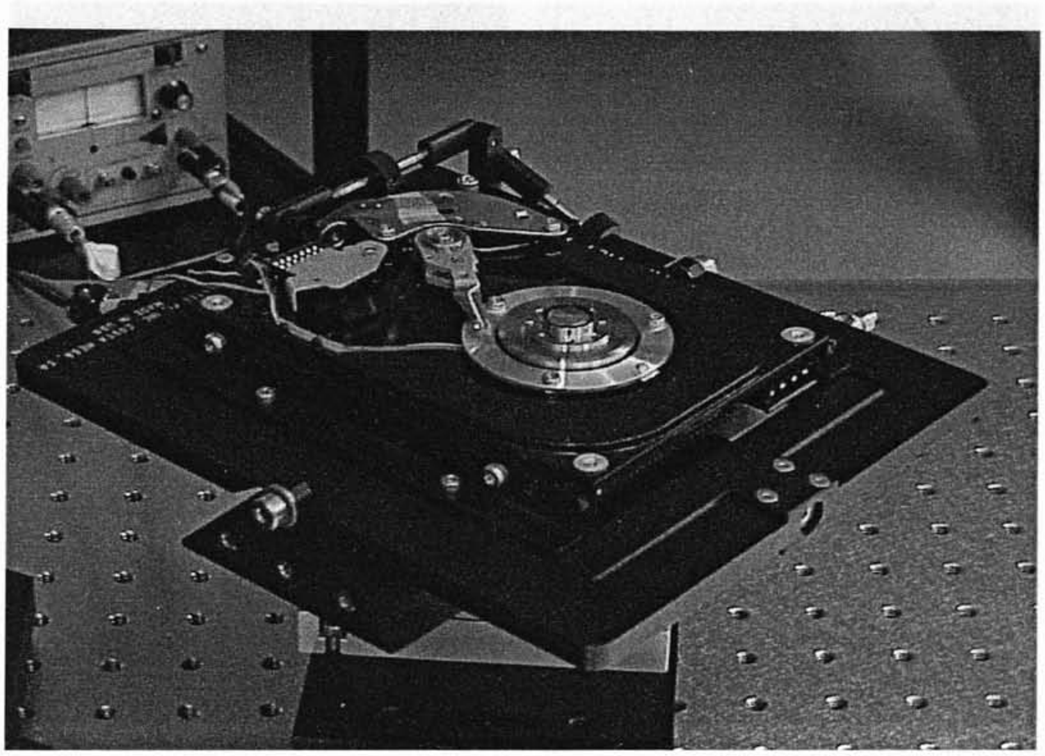

Figure 5.4: Open Disk Drive.

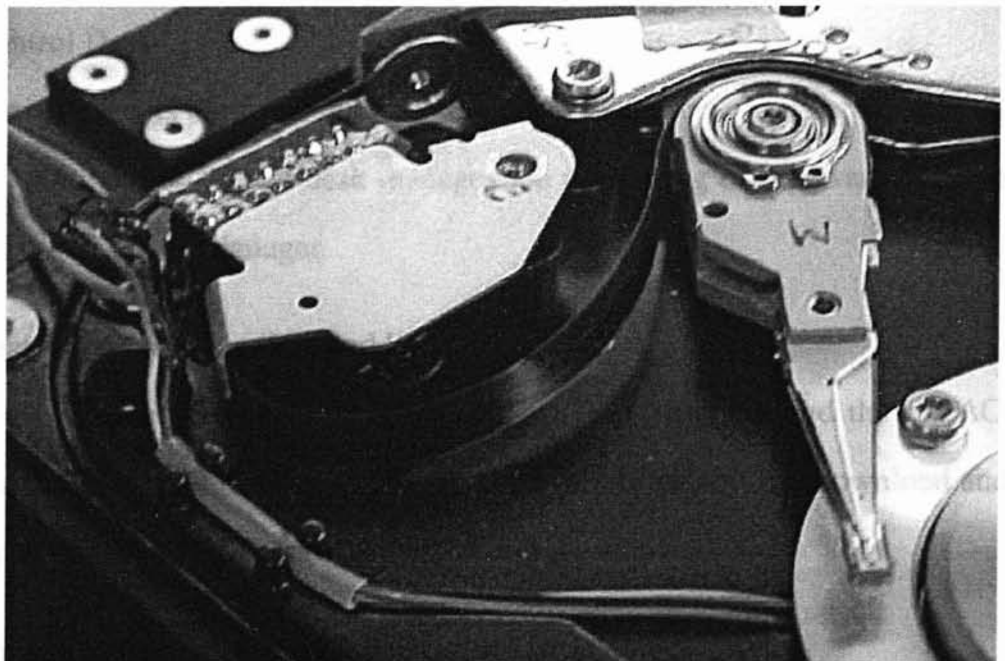

Figure 5.5: Flexible Printed Circuit.

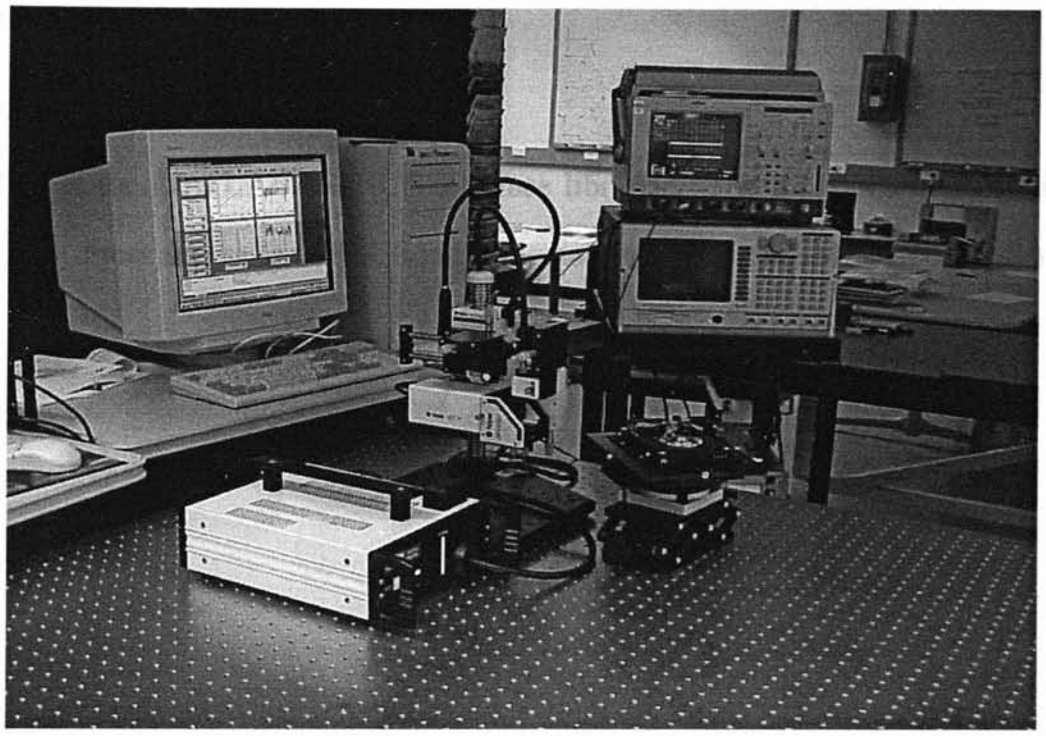

Figure 5.6: The Experimental Setup.

5.2.2 dSPACE Software

- Control Desk

  Control desk is a graphical user interface software for managing the dSPACE board. In addition, the control desk manages the registering of hardware and applications via the Platform Manager.

- Real-Time Interface (RTI and RTI-MP)

  The real-time Interface communicates between Simulink and the dSPACE board. The real-time interface, RTI, is used to build real-time code, download and execute this code on dSPACE real-time processor.

- Control Desk Standard

  Control desk standard offers a variety of virtual instruments to build and configure virtual instrument panels via instrumentation providing functions to perform parameter studies via the parameter editor and functions to automate control desk's via automation.

- MLIB/MTRACE

  This is the MATLAB-dSPACE interface libraries. The functions of these libraries allow direct access to dSPACE real-time hardware from the MATLAB workspace.

CHAPTER 6

MATLAB-dSPACE Interface Libraries

This section discusses on the use of the MLIB/MTRACE MATLAB-dSPACE interface libraries.

6.1 MLIB/MTRACE

The MATLAB-dSPACE interface libraries provide access to dSPACE real-time processor hardware from the MATLAB workspace. The MLIB/MTRACE functions can be called from the MATLAB command window or from m-files. Thus, powerful numerical tools running under MATLAB can be used in combination with the MLIB/MTRACE for:

- Analyzing real-time data
- Test automation
- Optimizing control algorithms

MLIB/MTRACE provides basic functions for reading and writing data to the dSPACE processor board and other functions like generating interrupts, setting the processor state and getting processor status information.

MLIB/MTRACE provides real-time data capture capabilities including the following features:

- Free-running or level-triggered mode
- Pre- and post-trigger
- Simultaneous start of multiple data captures
- Distinction between double, float and integer (signed and unsigned) variables
- Adjustable trace capture sampling rate (downsampling)
- Direct data transfer to the MATLAB workspace
- Data storage on the hard disk drive in continuous mode (optional)
- Specification of data capture parameters by property/values pair

MLIB/MTRACE functions are suited to modify parameters online and to send sequences of test data to real-time processor with MLIB/MTRACE. One does not need to know the hardware address because several functions are implemented that return the required addresses when the symbolic name of a variable is specified.

MLIB/MTRACE complements the software environment, which consist of MATLAB/ SIMULINK real-time workshop, and the dSPACE RTI and control desk. Information on MLIB/MTRACE functions and programming can be found in [22].

6.2 Experimental Setup Flow Chart with MLIB/MTRACE

Figure 6.1 shows the flow chart of the experimental setup and how MLIB/MTRACE interfaces with the real-time process. In standard operation, the controller that will be implemented is designed in the host computer using the MATLAB/SIMULINK real-time workshop. It is then downloaded onto the DS1104 PPC controller board from the control desk. The DS1104 controller board will send a control signal to the Kepco power amplifier, which amplifies the current that is supplied to the voice coil motor (VCM). The VCM controls the position of the actuator. The Polytec laser doppler vibrometer (LDV) measures the position and velocity of the actuator. It provides position and velocity feedback to the DS1104 PPC controller board. The Lecroy digital oscilloscope is used to monitor the reference, position, velocity, and control signal. In Figure 6.1, the dotted line shows the system running on real-time process. The MLIB/MTRACE allow direct access to the DS1104 PPC controller board from the MATLAB workspace window using the MLIB/MTRACE functions. Real-time data can be accessed from the MATLAB workspace with MLIB/MTRACE functions. The controller gains in the real-time processor can be modified in real-time through MATLAB workspace or from the m-files.

6.3 Procedure to Implement M-file with MLIB/MTRACE on Experiment

This section discusses the steps to use the MLIB/MTRACE functions to access the dSPACE real-time processor. The m-file scripts given below are attached in Appendix D. The Simulink model used is `dlqgltr.mdl`, and it is given in Figure 7.1.

6.3.1 M-file for Optimization-based Tuning of a Discrete Loop Transfer Recovery Controller with Hardware In-the-loop

This section gives a detailed explanation on the m-files used to run the optimization-based tuning with hardware in-the-loop experiment. The main file used is `dlqgltr_mlib.m` and the secondary file is `lqg_ltr.m`. The procedure and setup of `dlqgltr_mlib.m` and `lqg_ltr.m` are explained below. In `dlqgltr_mlib.m`, the first thing to do is to select the data-acquisition board used for the experiment. The following MLIB/MTRACE command selects the data-acquisition board:

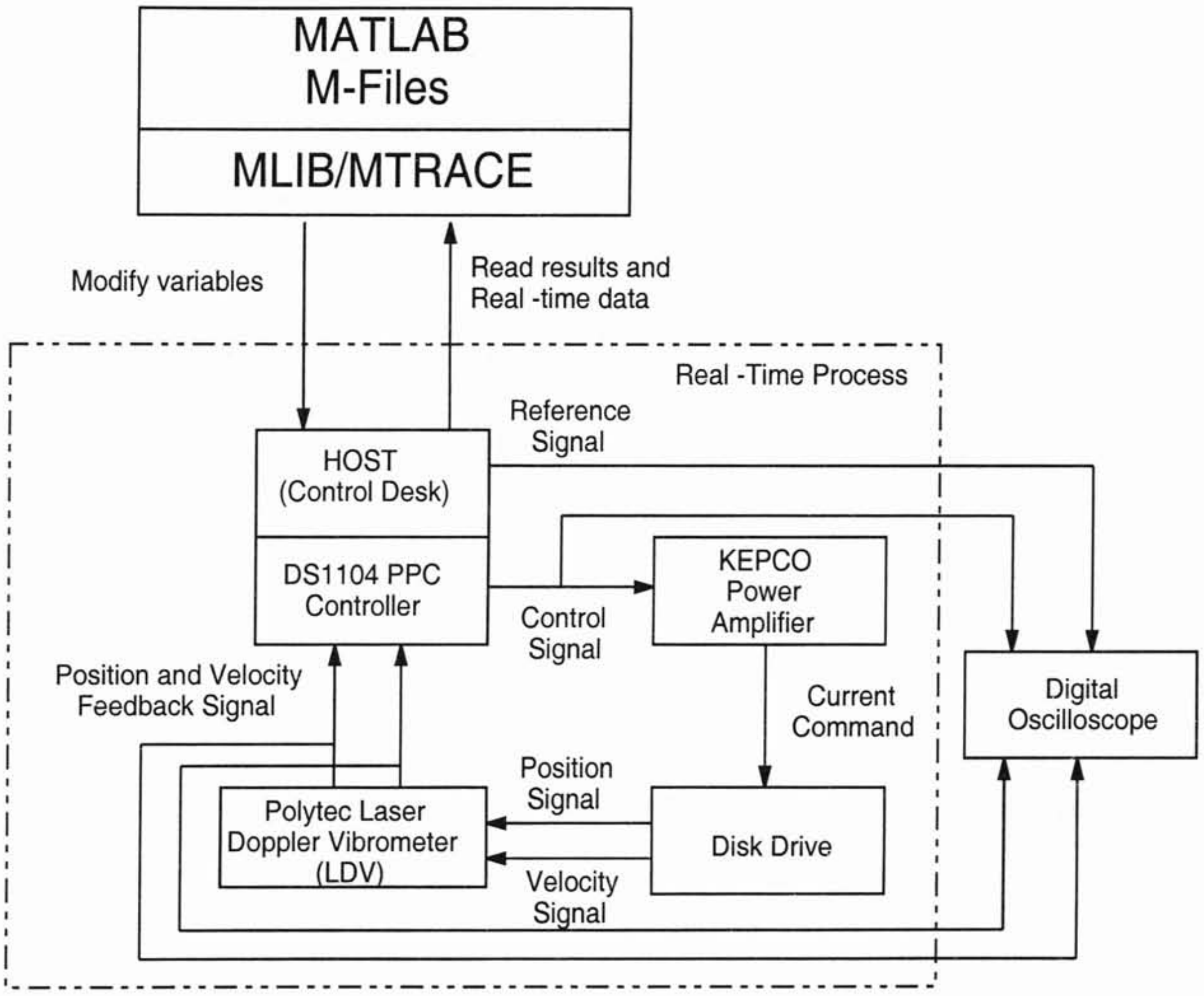

Figure 6.1: Experimental Setup with MLIB/MTRACE Interface.

```
% Initialize dSPACE MLIB
  mlib('SelectBoard','DS1104');
```

Then the following script checks if the real-time processor's application file is loaded. The real-time application file used is `dlqgltr.ppc`. This file is generated by MATLAB Real-time Workshop when the Simulink file, `dlqgltr.mdl`, is compiled.

```
% Check if the Application dlqgltr.ppc is Running
  ApplName = lower([pwd '\dlqgltr.ppc']);
  if mlib('IsApplRunning'),
   ApplInfo = mlib('GetApplInfo');
    if strcmp(ApplName,lower(ApplInfo.name)) ~= 1
   err_msg = sprintf('*** This MLIB file needs the real-time
     processor application\n*** ''%s'' running!',...
                              ApplName);
     error(err_msg);
    end;
  else
   err_msg = sprintf('*** This MLIB file needs the real-time
   processor application\n*** ''%s'' running!',...
                              ApplName);
      error(err_msg);
  end;
```

The following paragraphs detail the collection of datas for position signal, reference signal, and error signal.

- Position Signal

  The information on the position in `dlqgltr.mdl` Simulink diagram is accessed through the following MLIB/MTRACE function. `Model Root/Position/In1` is the address of the position block. The address of the signal is found using the dSPACE's TraceView Utility. Every time a real-time application is built, dSPACE's Real-Time Interface generates a TRC file that maps Simulink variable name to the corresponding global variables in the generated code. Thus, variables can be specified with the name of the Simulink blocks. Procedure to search for the address of the signals is given in Appendix B. The MLIB/MTRACE `'GetTrcVar'` function is used to obtain the discriptor for the position block. The MLIB/MTRACE `'Read'` function is used to read data from the real-time processor.

```
% Get descriptors for the position block
  var_names = {'Model Root/Position/In1'; ...
      };
  position_desc = mlib('GetTrcVar', var_names);
  position = mlib('Read', position_desc)
```

- Reference Signal

  The reference signal in the `dlqgltr.mdl` Simulink diagram is accessed through the following MLIB/MTRACE functions. The address of the reference signal block is `'Model Root/Signal_nGenerator/Amplitude'`. The functions used to obtain the discriptor and read data from the real-time processor are the same as those functions used to access the position block.

```
% Get descriptors for the Reference Signal block
  var_names = {'Model Root/Signal\nGenerator/Amplitude';
      };
  referencesignal_desc = mlib('GetTrcVar', var_names);
  referencesignal = mlib('Read', referencesignal_desc)
```

- Error Signal

  The information on the error in the `dlqgltr.mdl` Simulink block diagram is accessed through the following MLIB/MTRACE functions. The address for the error block is `'Model Root/error/In1'`. The rest of the MLIB/MTRACE functions used are the same as the ones used to access the position and the reference signal block.

  ```
  % Get descriptor for the error block
    var_names = {'Model Root/error/In1'; ...
        };
    error_desc = mlib('GetTrcVar', var_names);
    error = mlib('Read', error_desc)
  ```

The following part of the `dlqgltr.m` script uses the MATLAB Optimization Toolbox's nonlinear unconstrained optimization command to call MATLAB `lqg_ltr.m` file. Cost function computed in `lqg_ltr.m` is evaluated by the nonlinear unconstrained optimization toolbox in MATLAB command window.

```
% Nonlinear Unconstrained Optimization
  [X]=fminunc('lqg_ltr',[roh;mu]);
```

The secondary file used is `lqg_ltr.m`. The following part of the script declares the variables used in the script, initializes the sample time and function `'f'` that calculates the cost function. The final two lines displays the parameters that are used to design the controller in MATLAB workspace.

```
% Function Call
  function [f]=lqg_ltr(X,T,deltT)


% Declare Global Variables
```

```
  global H  F a b c d Ac Bc Cc Dc ac bc cc dc num_d den_d
  num_p den_p y t cs u error roh mu f Ad Bd Cd Dd

% Initialize Variables
  deltT=1e-5;   % Sample Time
  t=(0:1e-5:1999*1e-5);

  roh=X(1);
  mu=X(2);

% Display the parameters mu and roh
  disp(sprintf('====== roh=roh(1)=%6.8f ======',roh(1)));
  disp(sprintf('====== mu=mu(1)=%6.8f ======',mu(1)));
```

The following part shows how the controller is designed. The plant used to design the controller is based on a 2^{nd} order system with gains that match the 4^{th} order disc drive model. The 2^{nd} order plant is used to compute the observer gain matrix for target loop design and to compute the recovery gain matrix for the recovery loop. The continuous-time controller is converted in to a discrete-time controller using MATLAB's `c2d.m` command.

```
% 2nd Order Plant
  Ac=[0 1; 0 0];
  Bc=[0 0.75e11]';
  Cc=[1 0];
  Dc=[0];
  [num_p,den_p]=ss2tf(ac,bc,cc,dc);

% Target Filter Loop Design (C phi H)
```

```
  G=[1 0; 0 1];
  Q=diag([1 1]);
  [H,P,E]=lqe(Ac,G,Cc,Q,(mu)^2);
  H;   % Observer Gain Matrix H

% Recovery Loop Design
  Q2=Cc'*Cc;
  [F]=lqr(Ac,Bc,Q2,(roh)^2);
  F;   % Recovery Gain Matrix F


% Controller in Discrete and Continuous State-Space Form
  sys_c=ss(Ac-H*Cc-Bc*F,-H,-F,0);
  sys_d=c2d(sys_c,Ts,'tustin');
  [a,b,c,d]=dssdata(sys_d);
```

The error data must be collected for computation of the cost function. The following command gets the discriptor of the error block from the address and reads the error data. The trigger command set to trigger on the variable 'Model Root/error/In1'. The trace interval is 2000 samples with a post-trigger of 500 samples. Post-triggering will delay the trace interval by 500 samples before the samples are collected.

```
% Get descriptor for the error block
  var_names = {'Model Root/error/In1'; ...
    };
  var = mlib('GetTrcVar', var_names);
  Error_data = mlib('Read', error_desc)


  mlib('Set','Trigger','ON', ...
    'TriggerLevel',0, ... % Default, can be omitted
    'TriggerEdge','rising', ... % Default, can be omitted
    'TriggerVariable', var(1), ...
    'TraceVars', var, ...
    'NumSamples', 2000, ...
    'Delay',500);
```

The following MLIB/MTRACE `'StartCapture'` command starts the capture of error data on the real-time processor board. The error data that has been traced by the trigger command is captured on the real-time processor board. Then the MLIB/MTRACE `'FetchData'` command fetches the data from the real-time processor buffer and transfers the data to the MATLAB workspace.

```
% Start Capture on DS1104
  mlib ('StartCapture');
  while mlib('CaptureState')~=0, end

% Fetch After Capture is Complete
  error_data = mlib('FetchData');
```

Once the error data is transfered to the MATLAB workspace, the following functions are used to compute the cost function. Three types of performance indices are used to compute the cost function. The cost function computed is evaluated by the MATLAB Optimization Toolbox's unconstrained nonlinear optimization command.

```
% Cost Function (minimization)
  f=sum(t.*abs(error_data))*deltT;    %ITAE
  % f=sum(abs(error_data))*deltT;     %IAE
  % f=sum((error_data)^2)*deltT;      %ISE
```

The new discrete controller gains that are computed through the optimization method are transfered from the MATLAB workspace to the real-time processor board. Since the controller gains are in the form of state-space matrices, MLIB/MTRACE `'mlib_matrix'` commands are used. These commands are used to write the new controller gains into the state space block `State-Space` that resides in the Simulink subsystem controller which is downloaded into the real-time processor.

```
% Write New Data to the Following Area: row 1:2, columns 1:2
% for A Matrix
  myMatrix = mlib_matrix('Init',...
  'Model Root/Discrete State-Space\nLQGLTR Controller/A',...
  2, 2, 'Row-Wise');

  AnewData = a ;
  mlib_matrix('Write', myMatrix, [1:2], [1:2], AnewData);

% Write New Data to the Following Area: row 2,1, columns 1:1
% for B Matrix
  myMatrix = mlib_matrix('Init',...
  'Model Root/Discrete State-Space\nLQGLTR Controller/B',...
  2, 1, 'Row-Wise');

  BnewData = b;
  mlib_matrix('Write', myMatrix, [1:2], [1:1], BnewData);

% Write New Data to the Following Area: row 1,2, columns 1:2
% for C Matrix
  myMatrix = mlib_matrix('Init',...
  'Model Root/Discrete State-Space\nLQGLTR Controller/C',...
  1, 2, 'Row-Wise');

  CnewData = c;
  mlib_matrix('Write', myMatrix, [1:1] ,[1:2], CnewData);

% Write New Data to the Following Area: row 1,1, columns 1:1
% for D Matrix
  myMatrix = mlib_matrix('Init',...
  'Model Root/Discrete State-Space\nLQGLTR Controller/D',...
  1, 1, 'Row-Wise');

  DnewData = d;
  mlib_matrix('Write', myMatrix, [1:1] ,[1:1], DnewData);
```

Finally, the command `'siggen_dlqgltr'` resets the reference signal. The reference signal is a square wave with amplitude of 1 μm and the frequency of 50 Hz. By initializing the reference signal and delaying the data capture by 5 ms, transients from downloading new controller gains can be omitted from the cost function calculations.

```
% Reset reference input
  siggen_dlqgltr(1,50);  % Amplitude = 1 um
                         % Frequency = 50 Hz
```

The optimization process will continue to search for new controller gains and implement the gains into the real-time processor board until the cost function reaches the optimization termination tolerance. The MATLAB Optimization Toolbox `'fminunc.m'` command's default termination tolerance is $1 \cdot 10^{-4}$. When the optimization process terminates, the controller gains obtained are the optimum gains for that choice of performance index.

6.3.2 Program Block Diagram for Optimization-based Tuning of a Discrete Loop Transfer Recovery Controller with Hardware in-the-loop

The optimization-based tuning of a discrete loop transfer recovery controller with hardware in-the-loop discussed in the previous section is shown in Figure 6.2 as a program block diagram. The program block diagram is used to provide better understanding of the m-file.

First, the experiment which consists of dSPACE data-acquisition board, Polytec LDV, disc drive, Kepco power amplifier must be running before the `dlqgltr_mlib.m` program is initialized. Then, the `dlqgltr_mlib.m` program is initialized in MATLAB Workspace. The `'mlib('SelectBoard','DS1104')'` command is used by the program to select the DS1104 board. When the program selects the DS1104 board, the reference signal and the position signal can be accessed with MLIB/MTRACE command. The `'fminunc('lqg_ltr','[roh;mu]')` command is used to initialize the MATLAB Optimization method. If the DS1104 board is not found, an error message will appear on the MATLAB Workspace and the program will terminate.

The secondary program, `lqg_ltr.m` is initialized by the MATLAB Optimization method. Error signal from the board is accessed using the MLIB/MTRACE command. The error signal is captured on the board using `'mlib('CaptureState')'` command. The error signal is fetched from the board to the MATLAB Workspace using `'mlib('FetchData')'` command.

The data collected in MATLAB is used to compute the cost function. If the cost function does not reach the termination value, a new set of controller gains is computed. The new controller gains are written on the board using MLIB/MTRACE commands. The reference signal consists of a square wave with amplitude of 1μm and frequency of 50 Hz is initialized.

The program will go through all the steps of the secondary program in a loop as shown in Figure 6.2 until the cost function reaches the termination value. The program will terminate when the cost function reaches the value of $1 \cdot 10^{-4}$.

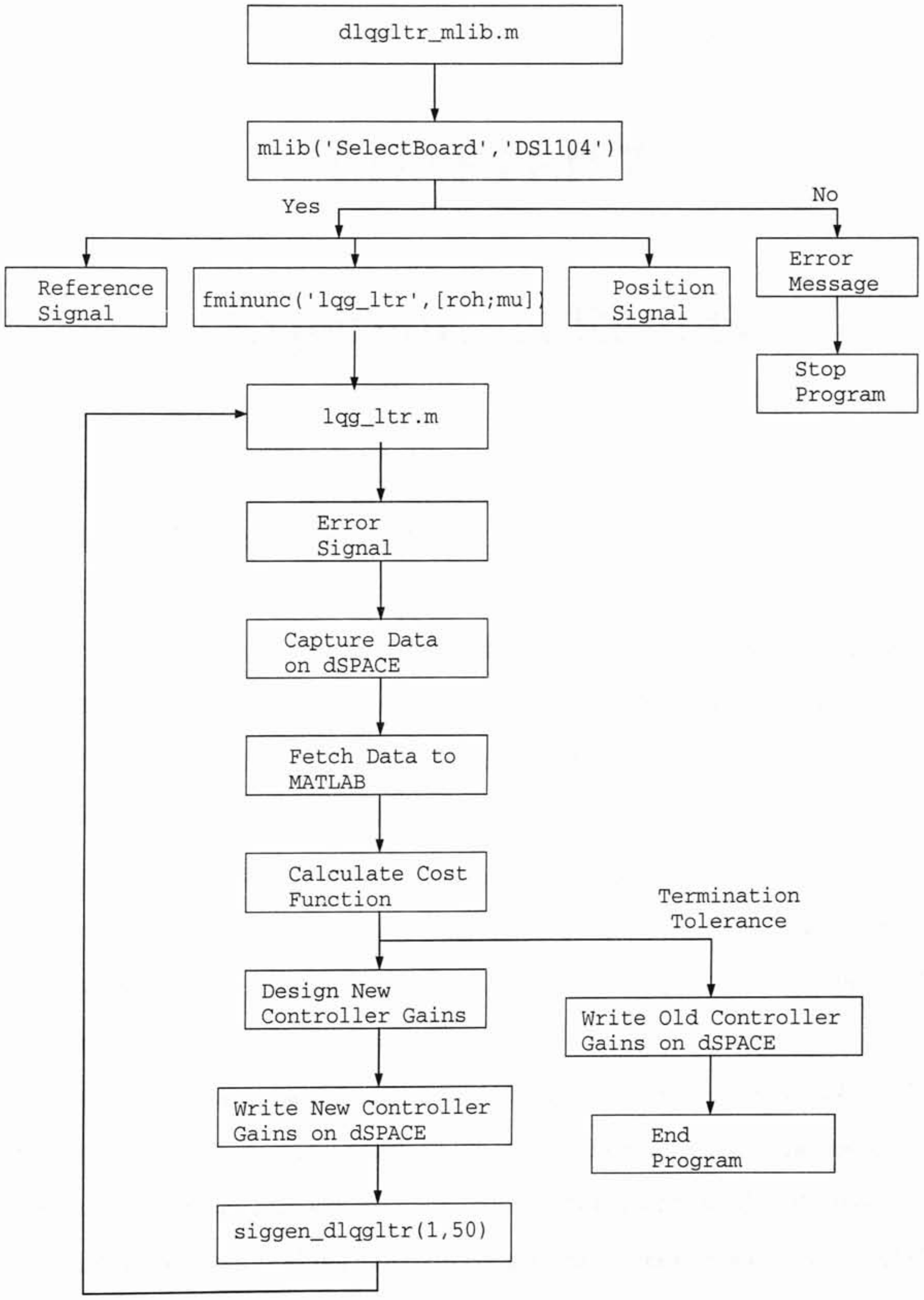

Figure 6.2: Program Block Diagram.

CHAPTER 7

Experimental Results

This chapter discusses the controller implementations and experimental results performed on actual hardware. The optimization-based LQG/LTR controller designed in Chapter 4, is implemented into actual hardware. Simulation results from Chapter 4 are compared with the experimental result. Experimental results for an optimization-based tuning of a discrete LQG/LTR with hardware in-the-loop with different cost functions are compared. ITAE, ISE and IAE are the performance indices used to compute the cost functions.

Figure 7.1 shows the Simulink model of the LQG/LTR controller used for the experiment. As shown in Figure 7.1, the reference signal is provided by the signal generator block or the step function block. The signal generator block provides sinusoidal, square, triangle and random signals. For this experiment, square wave signal is used. The amplitude corresponds to position measured in μm.

Two digital to analog channels of the dSPACE board were used. $DS1104DAC\_C1$ corresponds to the reference signal and $DS1104DAC\_C2$ corresponds to the control signal. Three analog to digital channels from the dSPACE board were used. $DS1104ADC\_C5$ corresponds to position, $DS1104ADC\_C6$ corresponds to velocity and $DS1103ADC\_C7$ corresponds to the control signal. The LDV parameter block and gain constant blocks provide gains for the unit conversion blocks.

The reference signal used in the experiment is a square wave with an amplitude of 1μm

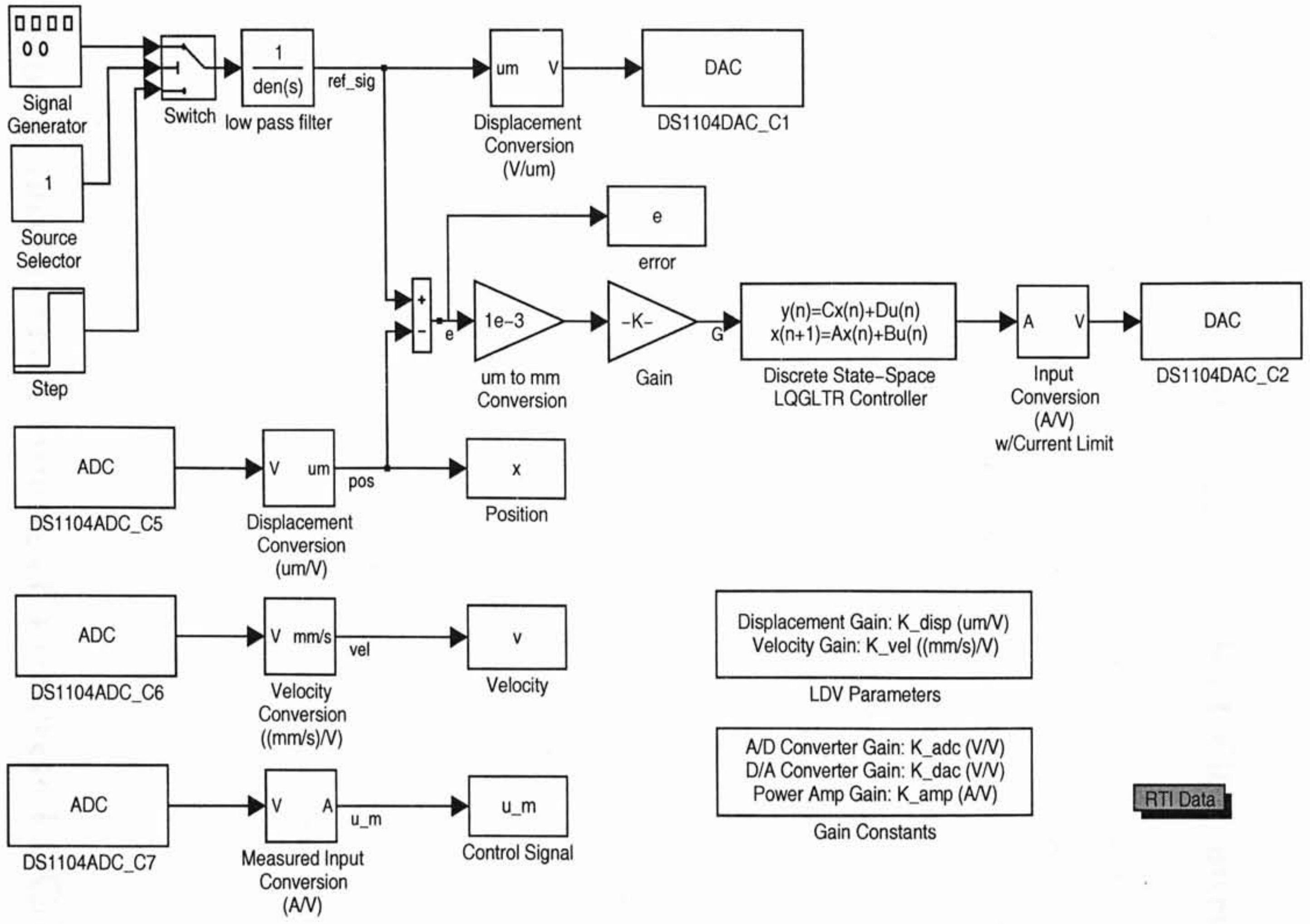
Signal Generator
Switch
1
den(s)
low pass filter
ref_sig
um V
Displacement Conversion (V/um)
DAC
DS1104DAC_C1
1
Source Selector
Step
e
error
+
-
e
1e-3
um to mm Conversion
-K-
Gain
G
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space LQGLTR Controller
A V
Input Conversion (A/V) w/Current Limit
DAC
DS1104DAC_C2
ADC
DS1104ADC_C5
V um
pos
Displacement Conversion (um/V)
x
Position
ADC
DS1104ADC_C6
V mm/s
vel
Velocity Conversion ((mm/s)/V)
v
Velocity
ADC
DS1104ADC_C7
V A
u_m
Measured Input Conversion (A/V)
u_m
Control Signal
Displacement Gain: K_disp (um/V)
Velocity Gain: K_vel ((mm/s)/V)
LDV Parameters
A/D Converter Gain: K_adc (V/V)
D/A Converter Gain: K_dac (V/V)
Power Amp Gain: K_amp (A/V)
Gain Constants
RTI Data

Figure 7.1: Simulink Diagram of the LQG/LTR Controller.

and frequency of 50 Hz. The LDV displacement gain is set at 20 μm/V and the LDV velocity gain is set at 25 mm/s/V.

7.1 Optimization-based Discrete LQG/LTR Controller

Controller gains from the disc drive simulation in Chapter 4 were implemented on hardware. The result from the experiment did not match the result from simulation. The experimental results have overshoot and a slow settling time. Thus, the controller gains needed to be manually tuned to achieve better performance. Figure 7.2 shows the result of optimization-based discrete LQG/LTR controller with manual tuning. The actual position tracks the reference signal and has a settling time, $T_s = 5$ ms. The simulation result in Figure 4.13 from Chapter 4 has a settling time, $T_s = 2.5$ ms. This result is 2.5 ms faster than the experimental result with manual tuning. This controller uses a substantial amount of control effort in order to track the reference signal, while the simulation result uses a small control signal to track the reference signal. The printed flexible cable provide a bias effect on the experimental result. The bias effect on the experimental result with manual tuning is compensated by the strong control signal.

Tuning the gains manually on the real-time processor through the control desk can improve response and settling time. However this process requires a lot of time and patience. A structured method like optimization-based tuning of the controller with hardware in-the-loop is used to speed up the tuning process.

7.2 Optimization-based Tuning of a Discrete LQG/LTR Controller with Hardware In-the-loop

The experimental results discussed below are performed using the optimization-based tuning of a discrete LQG/LTR controller with hardware in-the-loop using different perfor-

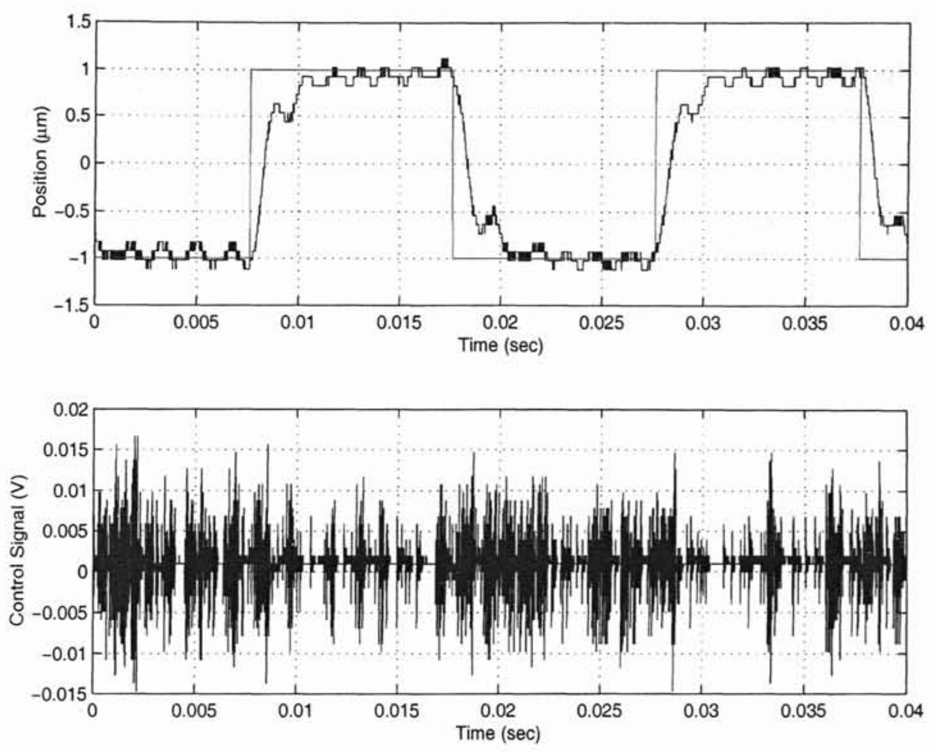

Figure 7.2: Experimental Result for $1\mu m$ Step with Manual Tuning.

mance indices. ITAE, IAE and ISE performance indices are used to compute the cost functions. To avoid transient data caused by implementation of new controller gains, error data used to compute the cost function is captured 5 ms after new controller gains are implemented. One cycle of data, which is equivalent to 20 ms or 2000 data points is collected for computation of the cost function. This process will continue to capture data for the optimization method until an optimum controller is found. To avoid insufficient memory on the computer for computation, 2000 data points for each iteration is captured.

Figure 7.3 shows the experimental result of optimization-based tuning of a discrete LQG/LTR controller with hardware in-the-loop using the ITAE performance index. The response of the controller is measured according to the settling time. Settling time is defined as the time required for a response to decrease and stay within a specific percentage of its final value [17]. This experiment is performed on a reference signal having a peak-to-peak amplitude of 2 μm. This is a representative of a single track in a 10000 tracks per inch disc drive. A relevant measure of settling time would be when the response stays within

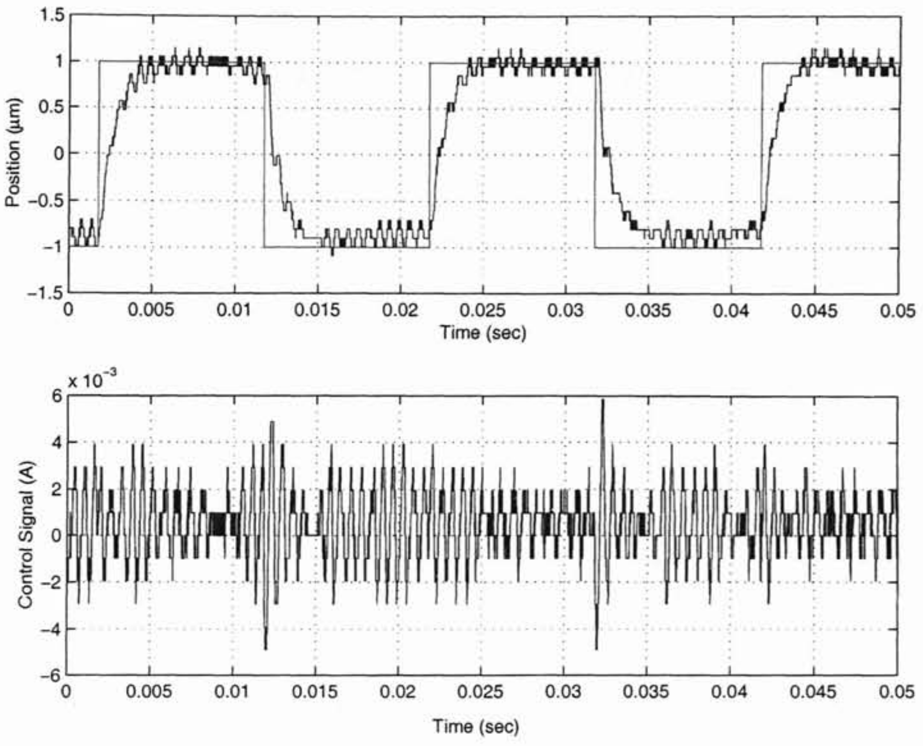

Figure 7.3: Experimental Result for $1\mu m$ Step with ITAE Performance Index.

10% of the final value. In disc drives, it is very common to use 5% to 10% of the track as the bound for the response. Using the ITAE performance index, the discrete LQG/LTR controller has a settling time of $T_s = 2.5$ ms. The response matches the simulation result in Chapter 4. As shown in Figure 7.3, the controller tracks the upper half of the reference signal. However, the bottom part of the reference signal does not track the reference well. The controller using the ITAE performance index uses less control effort for tracking compared to the controller with manual tuning. There is a one-directional force that pushes on the opposite direction when the controller tries to track the reference signal. The cause of the one-directional force is from the flexible printed circuit. The low control signal was not sufficient to compensate the bias force from the flexible printed circuit. When a response does not reach steady-state, the definition of settling time cannot be used. The upper part of the response that reached steady-state is used to measure the settling time.

Figure 7.4 shows the experimental result of optimization-based tuning of a discrete

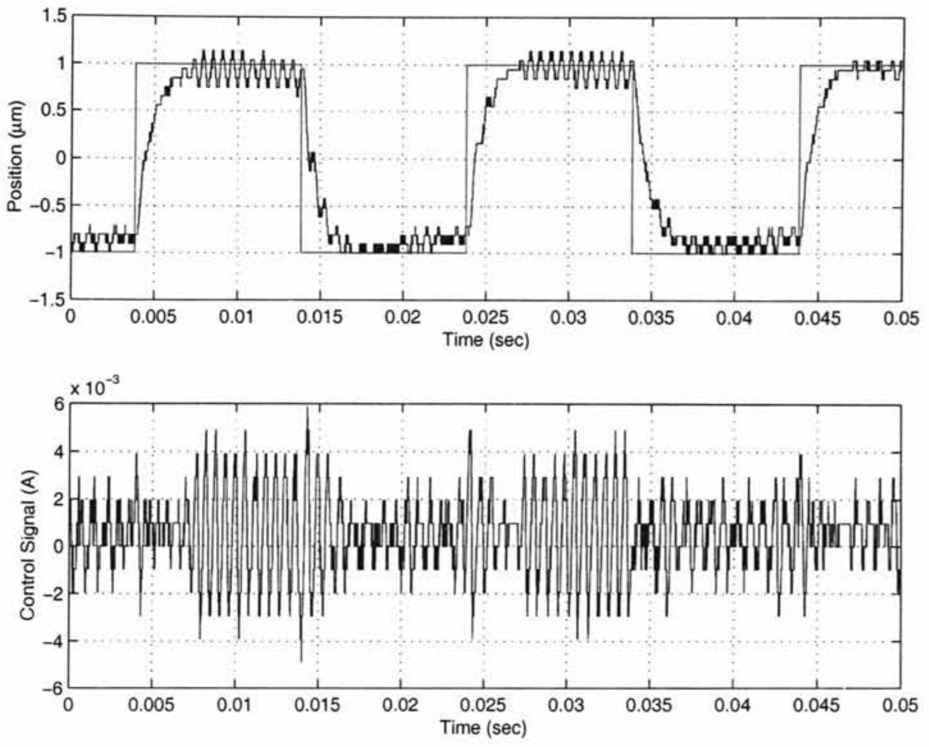

Figure 7.4: Experimental Result for $1\mu m$ Step with IAE Performance Index.

LQG/LTR controller with hardware in-the-loop using IAE performance index. The IAE performance index produces a response that rises smoother than the ITAE performance index. The settling time is, $T_s = 3.5$ ms. The controller tracks the upper half of the reference signal. The control signal in Figure 7.4 shows strong control signal at the upper part of the reference. However, the controller does not track the lower part of the reference signal well. The bias force from the flexible printed circuit caused the controller response to track poorly. The controller uses low control effort, like the controller that uses ITAE performance index. The low control effort is not sufficient to compensate the bias caused by the flexible printed circuit.

Figure 7.5 shows the experimental result of optimization-based tuning of a discrete LQG/LTR controller with hardware in-the-loop using ISE performance index. The settling time is, $T_s = 3$ milliseconds. ISE performance index produces response that rises smoother than ITAE performance index. The controller tracks the upper part of the reference signal. Strong control signal is used to track the response. The controller does not track the bottom

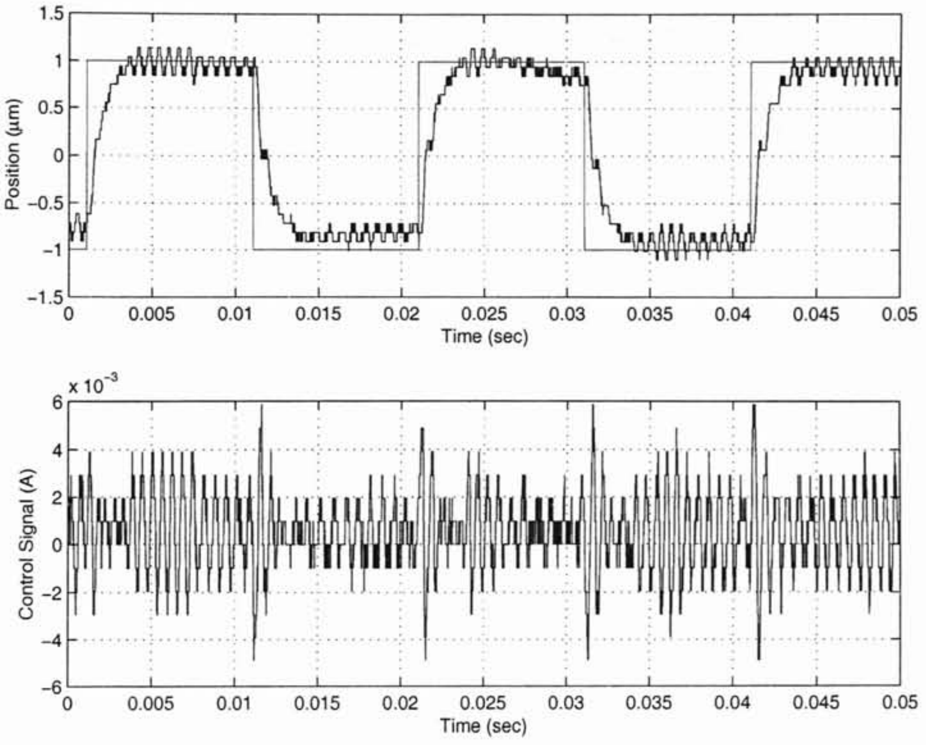

Figure 7.5: Experimental Result for $1\mu m$ Step with ISE Performance Index.

part of the reference signal well. A one directional force from the flexible printed circuit affects the tracking.

Comparing the results from Figure 7.3, Figure 7.4, and Figure 7.5, the controller using ITAE and IAE performance index is able to compensate the bias force more compared to controllers using ISE performance index. The bias force is not an equipment limitation. Bias force do affect manually tuned results but the strong control signal was able to compensate the bias force.

The mean square error (MSE) method is used to calculate the error and compare the results from 3 performance indices. A cycle of data is computed for each performance index. The mean square error method used is shown in Equation (7.1),

$$MSE = \frac{S_1^2 + S_2^2 + \cdots + S_I^2}{I} \tag{7.1}$$

where S^2 is the sample and I is the number of sample. The result is given in Table 7.1. The

Controller Design	Mean Square Error
LQG/LTR using ITAE	0.1970
LQG/LTR using IAE	0.1985
LQG/LTR using ISE	0.2022

Table 7.1: Mean Square Error for 3 Performance indices.

LQG/LTR controller using the ITAE performance index has the lowest mean square error. The LQG/LTR controller using the ISE performance index has the biggest error. The result proved that the controller using the ITAE performance index provides better result than the controller using ISE performance index.

CHAPTER 8

Conclusion & Future Work

When controller gains obtained from simulation are implemented on actual hardware, the controller gains needed to be manually tuned to improve performance. The solution to the problem is to tune the controller through an optimization technique while the hardware is running. An optimization-based method to tune the discrete Linear Quadratic Gaussian/Loop Transfer Recovery controller with hardware in-the-loop is presented.

First, a discrete Linear Quadratic Gaussian/Loop Transfer Recovery controller is designed with a low bandwidth. Then, optimization-based method is integrated with the LQG/LTR controller to search for gains that produce a controller with higher bandwidth. The controller gains obtained from simulation are implemented into the hardware. Finally, the controller is tuned using the optimization-based tuning with hardware in-the-loop.

The experimental results from manual tuning show that the controller tracks the reference signal with aggressive control effort and a slower settling time compared to the automated tuning method. Furthermore, tuning the controller manually requires a lot of time and patience. The experimental result from optimization-based tuning of a discrete LQG/LTR controller with hardware in-the-loop on the 3 different types of performance indices used show increased in performance. Table 8.1 show the summary of the performance of the controllers. The controller using ITAE has the fastest settling time compared to controller using IAE and ISE performance index. The automated tuning of the controller

Controller Design	Settling Time, T_s	Control Signal (A)	Bias Effect
LQG/LTR using manual tuning	5 ms	High	No
LQG/LTR using ITAE	2.5 ms	Low	Yes
LQG/LTR using IAE	3.5 ms	Low	Yes
LQG/LTR using ISE	3.0 ms	Low	Yes

Table 8.1: Result of the Disk Drive Experiment.

in real-time process produced controllers that used small control effort and achieved faster settling time.

Optimization-based tuning of the discrete Linear Quadratic Gaussian/Loop Transfer recovery controller with hardware in-the-loop shows better result compared to manual tuning method. The automated tuning method produces faster settling time and smaller control signal.

8.1 Future Work

Future work on this research are as follow:

1. Experiment show that the flexible printed circuit causes bias on the disc drive. The bias caused by the flexible printed circuit can be solved by including an estimate of the bias into the design.

2. A 2^{nd} order controller and a 4^{th} order simulation plant are used for simulation. The oscillations in Figure 4.10 and Figure 4.13 are caused by the unmodelled dynamics of the simulation plant. A higher order controller design that matches the characteristics of the disc drive will be used for simulation and experiment.

3. The idea of incorporating the optimization method to a controller will be applied to different types of controller.

BIBLIOGRAPHY

[1] Lyle, R. T. "Hyperplane Design for Discrete Variable Structure Control with Loop Transfer Recovery", *Masters Thesis*, Oklahoma State University, 2000.

[2] Hansleman, H. and Engelke, A. "Linear Quadratic Gaussian Control of a Highly Resonant Disk Drive Head Positioning Actuator", *IEEE Trans. Ind. Electron.*, vol. 35, pp. 100-104, 1988.

[3] Chen, L., Guo, G., Chen, B. M., and Ko, C. C. "Optimal Track Following Control For Hard Disk Drives", *AMC2000-Nagoya, Japan*, pp. 502-506, 2000.

[4] Beghi, A., Oboe, R., Capretta, P., and Soldavini, F. C. " Loop Shaping Issues in Hard Disk Drive Servo System Design", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings, Italy*, pp. 828-832, 2001.

[5] Chang, J-K., and Ho, H. T. "Linear Quadratic Gaussian/Loop Transfer Recovery Frequency Loop Shaping to Improve TMR Budget", *IEEE Transactions on Magnetics*, vol. 35, no. 5, pp. 2280-2283, 1999.

[6] Suh, S. M., Chung, C. C., and Lee, S. H. " Discrete-Time LQG/LTR Dual-Stage Controller Design in Magnetic Disk Drives", *IEEE Transactions on Magnetics*, vol. 37, no. 4, pp. 246-247, 2001.

[7] Doyle, J. C., and Stein, G. "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis", *IEEE Transactions on Automatic Control*, vol. 24, no.1, pp. 4-6, 1986.

[8] Doyle, J. C., and Stein, G. "Robustness of Observers", *IEEE Transactions on Automatic Control*, vol. 24, no.4, pp. 607-611, 1979.

[9] Stein, G. and Athans, M. "The LQG/LTR Procedure for Multivariable Feedback Control Design", *IEEE Transactions on Automatic Control*, vol. AC-32, no. 2, pp. 105-114, 1987.

[10] Athans, M. "A Tutorial on the Linear Quadratic Gaussian/Loop Transfer Recovery Method", *Proceedings of the American Control Conference, Seatle*, pp. 1289-1296, 1986.

[11] Athans, M., Valavani, L., and Martin, R. J. "Multivariable Control of a Submersible Using the LQG/LTR Design Methodology", *Proceedings of the American Control Conference*, vol. 2, pp. 1313-1324, 1986.

[12] Mathworks. "Optimization Toolbox for Use with MATLAB", *User Guide*, 2000.

[13] Broyden, C. G. "The Convergence of a Class of Double-rank Minimization Algotirhms", *J. Inst. Math. Applic.*, vol. 6, pp. 76-90, 1970.

[14] Fletcher, R. "A New Approach to a Variable Metric Algorithms", *Computer Journal*, vol. 13, pp. 317-322, 1970.

[15] Goldfard, D. "A Family of Variable Metric Updates Derived by Variation Means", *Mathematics of Computing*, vol. 24, pp. 23-26, 1970.

[16] Shanno, D. F. "Conditioning of Quasi-Newton Methods for Function Minimization", *Mathematics of Computing*, vol. 24, pp. 647-656, 1970.

[17] Dorf, R. C., and Bishop, R. H. " Modern Control Systems", *Addison Wesley*, 1998.

[18] Graham, D., and Lathrop, R. C. "The Synthesis of Optimum Response: Criteria and Standard Forms", *Transactions of the AIEE*, pp. 273-288, 1953.

[19] Polytec. "Laser Doppler Vibrometer", *User Manual*, 2000.

[20] Kepco Inc. "BOP 50-2M Operational Power Supply", *Instruction Manual*, 2000.

[21] dSPACE. "DS1104 Installation and Configuration Guide", *Instruction Manual*, 2001.

[22] dSPACE. "MATLAB-dSPACE Interface Libraries", *Instruction Manual*, 2001.

[23] Mathworks. "Real-Time for Use with Simulink", *User's Guide*, 2000.

APPENDIX A

Mathematical Model of the Disc Drive

A.1 Mathematical Modeling

This section discusses the mathematical modeling of the disc drive. The disc drive used for the experiment is manufactured by Conner. A Simulink model is generated to perform system identification of the disc drive actuator. The Simulink model is compiled and downloaded into the dSPACE board. The input signal used for the system identification experiment is a random signal.

The data collected is processed using MATLAB System Identification Toolbox. Figure A.1 is the frequency response of the disc drive actuator from current to velocity. The frequency response is estimated directly using the spectral analysis function in the System Identification Toolbox. The first peak resonance mode is at 1660 Hz, the second resonance mode is at 2900 Hz, and the third resonance is at 3350 Hz.

A.1.1 3^{rd} Order Design model

A 3^{rd} order mathematical model which maps from current to velocity is estimated using the System Identification Toolbox's Estimation of Parametric Models. **ARX Model** estimation method is used to produce the 3^{rd} order mathematical model. The transfer function of the

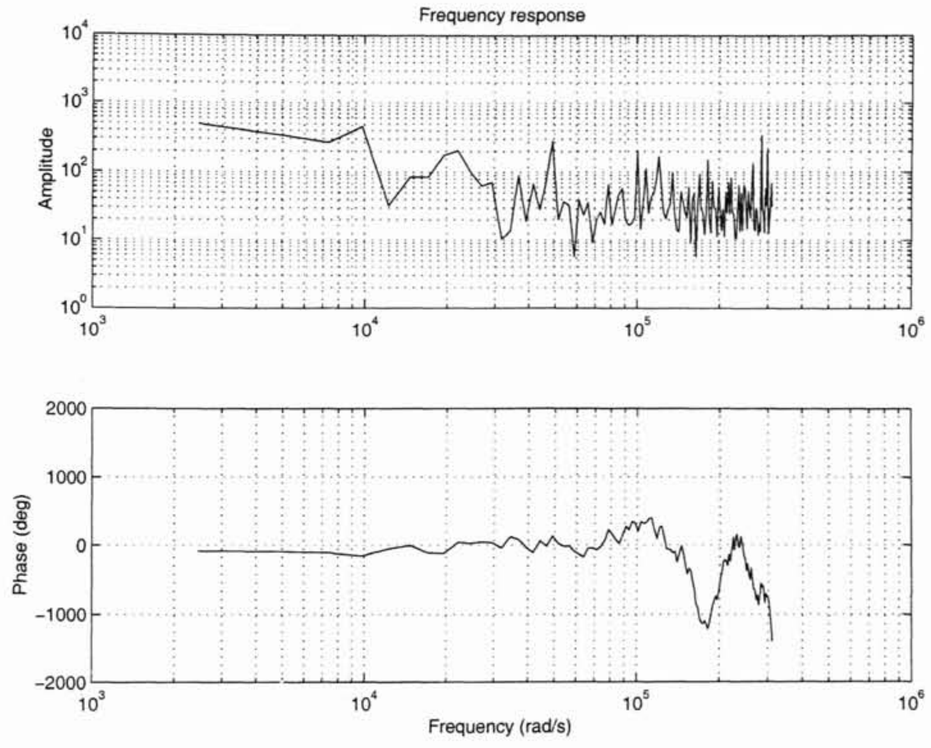

Figure A.1: Frequency response of the disk drive

3rd order mathematical model is given in Equation (A.1).

$$G_{ARX} = \frac{0.0275z^3 + 0.03038z^2}{z^3 - 2.986z^2 + 2.983z - 0.9971} \tag{A.1}$$

The sampling period used in this experiment is $T_s = 1 \cdot 10^{-5}$ seconds.

Figure A.2 shows the bode plot of the 3rd order disc drive model. The $*$ represents the actual data points of frequency, magnitude and phase measured from experiment.

A.1.2 7th Order Model

The 7th order mathematical model which maps from the current to velocity is estimated using the **ARMAX Model** estimation method. The transfer function of the 7th order model is given in Equation (A.2).

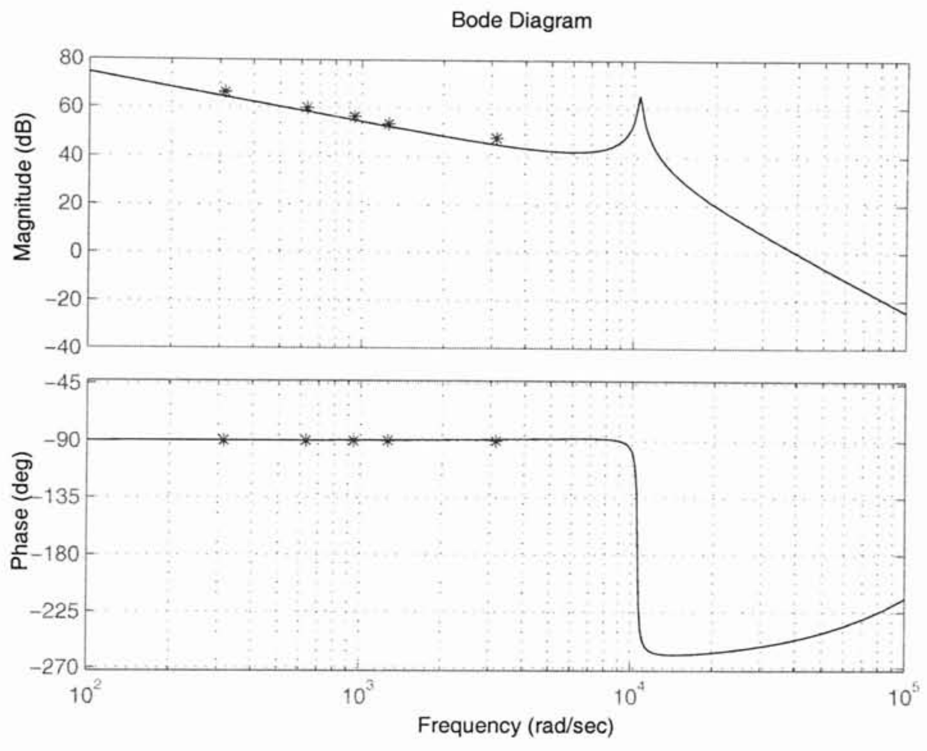

Figure A.2: Bode Plot of the 3^{rd} Order Mathematical Model

$$G = \frac{-1.565z^7 + 7.93z^6 - 16.11z^5 + 16.41z^4 - 8.38z^3 + 1.716z^2}{0.9686z^7 - 6.664z^6 + 19.72z^5 - 32.56z^4 + 32.37z^3 - 19.38z^2 + 6.473z - 0.9301} \tag{A.2}$$

The sampling period used in this experiment is $T_s = 1 \cdot 10^{-5}$ seconds.

Figure A.3 shows the bode plot of the 7^{th} order disc drive model. The $*$ represents the actual data points of frequency, magnitude and phase measured from experiment.

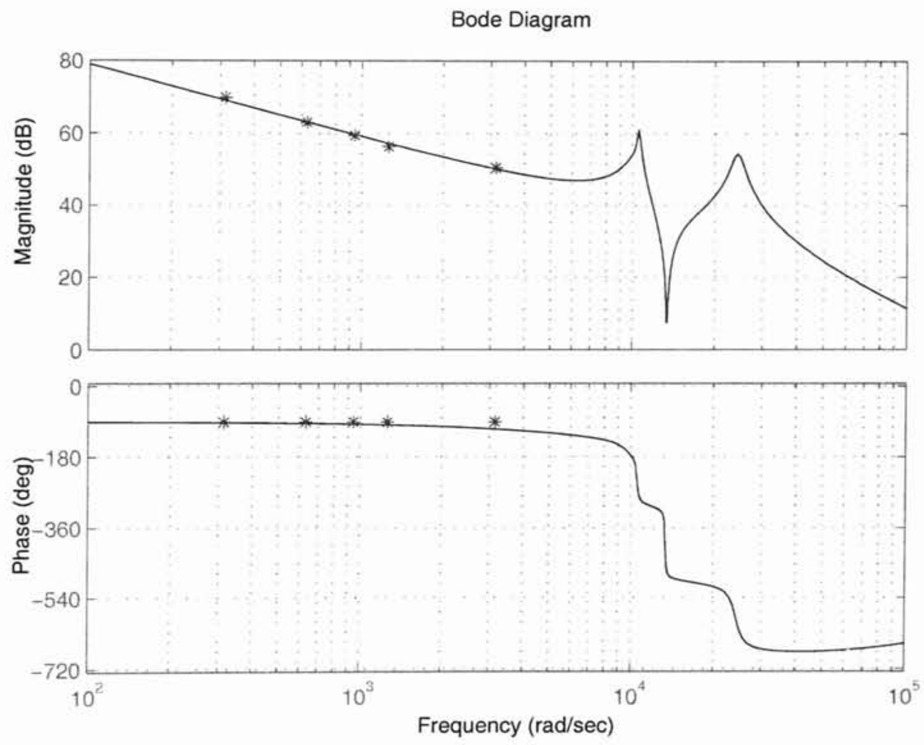

Figure A.3: Bode Plot of the 7th Order Mathematical Model

APPENDIX B

Procedure for using MLIB/MTRACE

This section discusses on the procedure to use the MATLAB-dSPACE Interface Libraries.

B.1 Getting Started

A real-time processor board for the use of MLIB/MTRACE must be selected before using the MLIB/MTRACE. The processor board must be registered in the Control Desk's Platform Manager. To obtain the board that is currently registered in the Control Desk's Platform Manager, type

```
board_info = mlib('GetBoardInfo')
```

in the MATLAB workspace window. The board_info will contain the basic information about registered board. If the board_info is empty, no board is registered.

MLIB/MTRACE is implemented as a MATLAB MEX DLL file, which is loaded into the PC memory when called for the first time during a MATLAB session. MLIB/MTRACE remains loaded until a **clear mex, clear functions, clear mlib,** or **clear all** commands is invoked, or until MATLAB is closed. If MLIB/MTRACE is cleared, **SelectBoard** must be called before proceeding with other MLIB/MTRACE functions.

B.2 MLIB/MTRACE Functions

This section discusses on the MLIB/MTRACE functions.

B.2.1 Select Board

In order to select the real-time processor board, type

```
mlib{'SelectBoard','DS1104');
```

DS1104 represents the board that is selected for real-time process. If a different board is used, replace 'DS1104' with the real-time processor board used.

B.2.2 CaptureState

The purpose of this function is to get capture status information.

```
state = mlib('CaptureState')
% wait for a data acquisition to finish
while mlib('CaptureState')~=0, end
disp('CaptureState');
```

B.2.3 FetchData

This MLIB/MTRACE function fetch the data from the real-time processor buffer and transfer it to the MATLAB workspace.

```
traced_data = mlib('FetchData')
traced_data = mlib('FetchData', [count ,] ...
   property_name, property_value, ...
```

Depending on the setting in the MLIB/MTRACE function Set, the FetchData function returns the traced data in one of the following formats:

- matrix
- matrix with time
- structure
- structure with time

B.2.4 GetTrcVar

This function obtain the descriptor for a variable specified within the TRC file.

```
var_vec              = mlib('GetTrcVar', Variable_names)
[var1, ..., varN] = mlib('GetTrcVar', variable_names)
```

If the real-time application is generated from a Simulink model using the MATLAB Real-Time Workshop and the dSPACE RTI, a TRC file is generated automatically. The TRC file maps Simulink variable names to the corresponding variable generated code (source code variable). Variables in real-time applications can be accessed within the names of the Simulink blocks, or with the label of the corresponding signal line.

B.2.5 Read

This function is used to read data from the real-time processor.

```
data                    = mlib('Read', var_vec)
[data1, ..., dataN] = mlib('Read', var_vec)
```

B.2.6 Write

This function is used to write data to the real-time processor.

```
mlib('Write', var_vec, 'Data', data)
```

The naming conventions and the MLIB/MTRACE function reference are shown in page 40 and page 41 of [22].

B.3 The TraceView Utility

Discriptors for global variables in the real-time application generated with dSPACE's RTI can be obtain with the GetTrcVar function. Every time a real-time application is built, dSPACE's Real-Time Interface generates a TRC file that maps Simulink variable name to the corresponding global variables in the generated code. Thus variables can be specified with the name of the Simulink blocks whose outputs, parameters, states, or state derivatives they represent, or with the label of the corresponding signal line.

Below is the procedure to use the TrcView Utility.

1. Enter the command trcview in the MATLAB command window. Then open the desired TRC file. A model browser and a variable list box let you search easily for the desired variable.

2. double click in the list box to paste the full variable name with the syntax required by the GetTrcVar into an edit field.

3. From the edit field copy and paste the variable name into the M-file editor or the MATLAB command window.

B.4 MLIB/MTRACE Error Messages and Troubleshooting

The appendix in [22] shows the MLIB/MTRACE's error messages in numerical order and offers solutions where applicable.

APPENDIX C

Matlab Setup Files

C.1 Setup File For Unconstrainted Optimization Method

```
% This M-file is use to search for the optimum roh and
% mu for Discrete LQG/LTR using MATLAB Toolbox's
% Optimization Method.
% run_lgqltr.m
%
% Design Criteria
% 1) Initial bandwidth of 1000 rad/sec
% 2) Target Filter Loop, mu=0.000001
% 3) Loop Transfer Recovery, roh=0.0001
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

% Initially clear the workspace
 clear all
 close all
 clc

% Initialize values
```

```
 global H  F a b c d Ac Bc Cc Dc ac bc cc dc num_d
 num_p den_p t y cs e u roh mu f Ad Bd Cd Dd den_d

 roh=1e2;  % Initial value used for unconstrainted
 mu= 1e-6; % Optimization method
 Ts=1e-5;
 deltT=Ts;  % sample time per second
 stoptime=0.01; % Simulation stop time in msec
 T=stoptime;

% Optimization Options
 options(2)=1e-5;  % terminal for X
 options(3)=1e-5;  % terminal for f
 options(14)=1000*4; % max iterations

% Using the Unconstrainted method
% FMINSEARCH Multidimensional unconstrained nonlinear
% minimization (Nelder-Mead).
% [X] = FMINSEARCH(FUN,X0,OPTIONS,P1,P2,...)

 [X]=fminsearch('lqg_ltr',[roh;mu]);
```

C.2 Setup File for Discrete-Time LQG/LTR Control with Unconstrained Optimization

```
% This M-file set up all the parameters to be called by
% run_lqgltr. This file is a local file.
% lqg_ltr.m
%
% Design Criteria
% 1) Initial bandwidth of 1000 rad/sec
% 2) Target Filter Loop, mu=0.000001
% 3) Loop Transfer Recovery, roh=0.0001
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

% Function Call
 function [f,g]=lqg_ltr(X,T,deltT)

% Initialize variables
 global H  F a b c d Ac Bc Cc Dc ac bc cc dc num_d
 num_p den_p y t cs u e roh mu f Ad Bd Cd Dd den_d

 roh=X(1);
 mu=X(2);

 Ts=1e-5; % Sample time per second
 deltT=Ts;
 stoptime=0.01; % Simulation stop time
 T=stoptime;

% Display of the parameters that is been controlled
 disp(sprintf('=====roh=roh(1)=%6.8f =====',roh(1)));
 disp(sprintf('=====mu=mu(1)=%6.8f =====',mu(1)));
 mu=(mu).^2; % Keep the values positive
 roh=(roh).^2;

% Load 3rd order model of the disk drive to workspace.
% Note the model is designed from current to velocity
% The 4th order model is the desgin plant
 load final3model.mat
 nnum;
 nnden=conv(nden,[1 -1]);     % 4th order discrete model
 dmodel=tf(nnum,nnden,Ts);
 [Ad,Bd,Cd,Dd,Ed,ts]=dssdata(dmodel);% Disc. State-Space
 dmodelc=d2c(dmodel,'tustin');
 [ac,bc,cc,dc]=ssdata(dmodelc);      % Cont. State-Space

% A 2nd order state space model with magnitudes that
% match the 4th order plant
 Ac=[0 1; 0 0];
 Bc=[0 0.75e11]';
 Cc=[1 0];
 Dc=[0];
 [num_p,den_p]=ss2tf(ac,bc,cc,dc);

% Target filter loop design (C phi H)
 G=[1 0; 0 1];
 Q=diag([1 1]);
%mu=0.000001; % Initial value for 1000 rad/sec
 [H,P,E]=lqe(Ac,G,Cc,Q,mu);
 H;    % Observer Gain Matrix H

% Loop Transfer Recovery
%roh=1e-4; % Initial value for 1000 rad/sec
 Q2=Cc'*Cc;
 [F]=lqr(Ac,Bc,Q2,roh);
 F;  % Loop Transfer Recovery Gain


% LQG/LTR Controller in State Space form
```

```
 sys_c=ss(Ac-H*Cc-Bc*F,-H,-F,0);
 sys_d=c2d(sys_c,Ts,'tustin');
 [a,b,c,d]=dssdata(sys_d); % Discrete Controller

% Controller in Transfer Function Form
 [num_c,den_c]=ss2tf(Ac-H*Cc-Bc*F,-H,-F,0);
 [num_d,den_d]=ss2tf(a,b,c,d);
 d_ssc=tf(num_d,den_d,Ts);

% Call the Simulink diagram for simulation
 sim('sim2model')

% Error between reference and actual signal
 e=r-y;

% Objective Function or Cost Function (minimization method)
 f=sum(t.*abs(e))*deltT
```

C.3 Setup File for Discrete-Time LQG/LTR Control on a 4^{th} Order Model

```
% This M-file is used to run simulation using Discrete
% LQG/LTR control with the new observer gain and recovery
% gain with the roh and mu found from the Optimization
% search method.
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

% Initially clear the workspace
 clear all
 close all
 clc

% Physical parameters
 Ts=1e-5; % samples per second
 stoptime=0.01; % simulation stop time in miliseconds
 T=stoptime;

% Load 3rd order model of the disk drive to workspace.
% Note the model is designed from current to velocity
% The 4th order model is the desgin plant
 load final3model.mat
 nnum;
 nnden=conv(nden,[1 -1]); % 4th order discrete model
 dmodel=tf(nnum,nnden,Ts);
 [Ad,Bd,Cd,Dd,Ed,ts]=dssdata(dmodel);% Disc. State Space
 dmodelc=d2c(dmodel,'tustin');
 [ac,bc,cc,dc]=ssdata(dmodelc);      % Cont. State Space

% A 2nd order state space model with magnitudes that match
% the 4th order plant
```

```
 Ac=[0 1; 0 0];
 Bc=[0 0.75e11]';
 Cc=[1 0];
 Dc=[0];
 [num_p,den_p]=ss2tf(Ac,Bc,Cc,Dc);

% Target Filter Loop Design (C phi H)
 G=[1 0; 0 1];
 Q=diag([1 1]);
 mu= (0.000001);
 [H,P,E]=lqe(Ac,G,Cc,Q,mu);
 H;    % Observer Gain Matrix H

% Plot the Target filter Loop
% Calculate  and plot the singular values for the
% filter loop
 [At,Bt,Ct,Dt]=linmod('target');
 [num_t,den_t]=ss2tf(At,Bt,Ct,Dt);
 sys=tf(num_t,den_t);
 sigma(sys,{10,1000000}), grid
 hold

% Loop Transfer Recovery
 roh=(4315.71447617).^2);  % Roh from Optimization
 Q2=Cc'*Cc;
 [F]=lqr(Ac,Bc,Q2,roh);  % Loop Recovery Loop Gain

% Plot the Loop Transfer Recovery
 [Ar,Br,Cr,Dr]=linmod('recovery');
 [num_r,den_r]=ss2tf(Ar,Br,Cr,Dr);
 rsys=tf(num_r,den_r);
 sigma(rsys,'r',{10,100000}), grid


% LQG/LTR Controller in State Space form
 sys_c=ss(Ac-H*Cc-Bc*F,-H,-F,0);
 sys_d=c2d(sys_c,Ts,'tustin');
 [a,b,c,d]=dssdata(sys_d); % Discrete Controller

% Controller in Transfer function form
 [num_c,den_c]=ss2tf(Ac-H*Cc-Bc*F,-H,-F,0);
 [num_d,den_d]=ss2tf(a,b,c,d)
 d_ssc=tf(num_d,den_d,Ts)

% Call the Simulink diagram for simulation
 sim('sim2model')
```

C.4 UNPCK to Extract Experimental Data

```
% UNPCK Extracts experimental data from structures
% generated by dSpace Control Desk.
%
% To use, load the data file into the workspace, assign
% the name of the data set to the variable 'data', then
% execute the command 'unpck'.  The data arrays are
% assigned to variable names matching their label name
% in Control Desk (i.e., in the Simulink system loaded by
% Control Desk.)  This information is displayed by
% 'unpck' as it executes.
%
%  Example:
%  >> load experiment_001.mat
%  >> data = experiment_001;
%  >> unpck
%
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

 disp(['Extracting: t'])
 t=getfield(getfield(data,'X'),'Data');

 temp_data=getfield(data,'Y');
 for i=1:size(temp_data,2)
 temp_name=getfield(temp_data,{i},'Name');
 disp(['Extracting: ',temp_name(1,findstr(temp_name,'/')
 +2:size(temp_name,2)-1)])
 assignin('base',temp_name(1,findstr(temp_name,'/')
 +2:size(temp_name,2)-1),getfield(temp_data,{i},'Data'));
 end

 disp(['Extracting: T_sample'])
 T_sample=getfield(getfield(data,'Capture'),
'SamplingPeriod');

clear i
clear temp_data
clear temp_name
clear temp_label
```

APPENDIX D

MLIB/MTRACE Matlab-dSPACE Interface Libraries

D.1 Setup File For Signal Generator Block

```
% This M-file is used to access and change the amplitude and
% frequency of the signal generator via MATLAB workspace
% when the simulink model is downloaded into the dSPACE
% acquisition board
%
% siggen_dlgqltr.m
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

function siggen_dlqgltr(amplitude,frequency);

% -------------------------------------------------------
% function siggen_dlqgltr(amplitude,frequency);
%
% SIGGEN_DLQGLTR() sets the parameters (amplitude and
% frequency) of a signal generator using the dSPACE MATLAB
% Interface Library. The signal generator is part of the
% application 'dlqgltr'. It is assumed that the unit of the
% parameter 'frequency' of the signal generator is 'Hertz'.
%
% Amplitude peak value of the signal measured in micrometer
%
% Frequency frequency of the signal generator in Hz
%
% When called without input parameters, SIGGEN_DLQGLTR
% displays the current settings of the signal generator.
%
% Before invoking this M-file the real-time processor
% application dlqgltr.ppc must be loaded
% ----------------------------------------------------------


% Initialize dSPACE MLIB
 mlib('SelectBoard','DS1104');

% check if the application dlqgltr.ppc is running
 DemoApplName = lower([pwd '\dlqgltr.ppc']);
 if mlib('IsApplRunning'),
 ApplInfo = mlib('GetApplInfo');
   if strcmp(DemoApplName,lower(ApplInfo.name)) ~= 1
   err_msg = sprintf('*** This MLIB file needs the real-
 time processor application\n*** ''%s'' running!',...
                            DemoApplName);
     error(err_msg);
   end;
 else
   err_msg = sprintf('*** This MLIB file needs the real-
 time processor application\n*** ''%s'' running!',...
                            DemoApplName);
   error(err_msg);
 end;

% Get descriptors for signal generator parameters
 variables = {'Model Root/Signal\nGenerator/Amplitude';...
             'Model Root/Signal\nGenerator/Frequency'};
 var_desc   = mlib('GetTrcVar',variables);

 if nargin == 0,
% Print out command syntax
 fprintf('\n Usage: siggen_dlqgltr(amplitude,frequency)\n
 \n');
% Read and print out signal generator parameters
 [amp, freq] = mlib('Read',var_desc);
 fprintf (' Signal generator parameters: Amplitude=%g um,
 Frequency=%g Hz\n',amp,freq);
 return
 end

% Set period and peak value
 mlib('Write',var_desc,'Data',{amplitude;frequency});
```

D.2 Setup File For Step Function Block

```
% This M-file is used to access and change the amplitude and
% frequency of the Simulink Step block via MATLAB workspace
% when the simulink model is downloaded into the dSPACE
% acquisition board
%
% step_dlgqltr.m
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

function step_dlqgltr(time,before,after);

% This M-file 'Read' information from the position,
% reference signal and error block.
%
% Get descriptors for the Step input block
%
% var_names = {'Model Root/Step/After'; ...
%     };
%
% step_desc = mlib('GetTrcVar', var_names);
%
% step = mlib('Read', step_desc)
%
% % Set  peak value
% mlib('Write',var_names,'Data',{After});

% Initialize dSPACE MLIB
 mlib('SelectBoard','DS1104');

% check if the application dlqgltr.ppc is running
 DemoApplName = lower([pwd '\dlqgltr.ppc']);
 if mlib('IsApplRunning'),
  ApplInfo = mlib('GetApplInfo');
  if strcmp(DemoApplName,lower(ApplInfo.name)) ~= 1
   err_msg = sprintf('*** This MLIB file needs the real-
 time processor application\n*** ''%s'' running!',...
                                DemoApplName);
      error(err_msg);
  end;
 else
  err_msg = sprintf('*** This MLIB file needs the real-time
 processor application\n*** ''%s'' running!',...
                                DemoApplName);
  error(err_msg);
 end;
 variables   = {'Model Root/Step/Time';...
                'Model Root/Step/Before'; ...
                'Model Root/Step/After'};

 var_desc    = mlib('GetTrcVar',variables);

 if nargin == 0,
% Print out command syntax
 fprintf ('\n Usage: step_dlqgltr(time,before,after)\n\n');
% Read and print out signal generator parameters
 [time,before,after] = mlib('Read',var_desc);
 fprintf ('Step Input parameters: Time=%g sec,Initial=%g um,
 Final=%g um\n',time,before,after);
 return
 end

% Set period and peak value
 mlib('Write',var_desc,'Data',{time;before;after});
```

D.3 Setup File for Optimization-Based Tuning of a Discrete- Time LQG/LTR Control with Hardware In-the-loop

```
% This M-file is used to run the Optimization-Based Tuning
% of a Discrete-Time LQG/LTR Controller with Hardware
% In-the-loop.
% dlgqltr_mlib.m
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%


% Initialize dSPACE MLIB
 mlib('SelectBoard','DS1104');

% Check if the application dlqgltr.ppc is running
 DemoApplName = lower([pwd '\dlqgltr.ppc']);
 if mlib('IsApplRunning'),
  ApplInfo = mlib('GetApplInfo');
 if strcmp(DemoApplName,lower(ApplInfo.name)) ~= 1
  err_msg = sprintf('*** This MLIB file needs the real-time
  processor application\n*** ''%s'' running!',...
                             DemoApplName);
    error(err_msg);
  end;
 else
  err_msg = sprintf('*** This MLIB file needs the real-time
  processor application\n*** ''%s'' running!',...
                             DemoApplName);
  error(err_msg);
 end;
```

```
% Get descriptors for the position block
var_names = {'Model Root/Position/In1'; ...
    };

 position_desc = mlib('GetTrcVar', var_names);

 position = mlib('Read', position_desc)

% Get descriptors for the Reference Signal block
 var_names = {'Model Root/Signal\nGenerator/Amplitude'; ...
    };

 referencesignal_desc = mlib('GetTrcVar', var_names);

 referencesignal = mlib('Read', referencesignal_desc)

% Get descriptor for the error block
 var_names = {'Model Root/error/In1'; ...
    };

 error_desc = mlib('GetTrcVar', var_names);

 error = mlib('Read', error_desc)


%%%%%%%%*************************************%%%%%%%%%%


 global H  F a b c d Ac Bc Cc Dc ac bc cc dc num_d den_d
 den_p t y cs error u roh mu f Ad Bd Cd Dd num_p

 roh= 4342;
 mu= 0.0000167;

 deltT=1e-5; % Sampling time

% Unconstrained Optimization
```

```
%%%**** [X] = FMINSEARCH(FUN,X0,OPTIONS,P1,P2,...) ***%%%

 if error > (0.001)
   [X]=fminunc('lqg_ltr',[roh;mu]);
 else
    a, b, c, d  % [X] = a,b,c,d;
 end             % Discrete state-space controller gain
```

D.4 Setup File for Discrete-Time LQG/LTR Control with Unconstrained Nonlinear Optimization

```
% This M-file setup all the parameters to be called by
% dlqgltr_mlib. This is a local file.
%
% lgq_ltr.m
%
% M-file written by Ban Fu Chee
% Advanced Controls Laboratory
% Oklahoma State University
% Stillwater OK 74075
%

% Function Call
 function [f,g]=lqg_ltr(X,T,deltT)

% Initialize variables
 global H  F a b c d Ac Bc Cc Dc ac bc cc dc cs u t y
 num_p den_p error roh mu f Ad Bd Cd Dd num_d den_d

 roh=X(1);
 mu=X(2);

 deltT=1e-5; % Sample time per second
 t=(0:1e-5:1999*1e-5); % Simulation time

% Display of the parameters that is been controlled
 disp(sprintf('====== roh=roh(1)=%6.8f ====',roh(1)));
 disp(sprintf('====== mu=mu(1)=%6.8f ====',mu(1)));

% A 2nd order state space model with magnitudes that
% match the 4th order model
 Ac=[0 1; 0 0];
 Bc=[0 0.75e11]';
 Cc=[1 0];
 Dc=[0];
 [num_p,den_p]=ss2tf(ac,bc,cc,dc);

% Target filter loop design (C phi H)
 G=[1 0; 0 1];
 Q=diag([1 1]);
 [H,P,E]=lqe(Ac,G,Cc,Q,(mu)^2);
 H;   % Observer Gain Matrix H

% Loop Transfer Recovery
 Q2=Cc'*Cc;
 [F]=lqr(Ac,Bc,Q2,(roh)^2);


% LQG/LTR Controller in State Space form
 sys_c=ss(Ac-H*Cc-Bc*F,-H,-F,0);
 sys_c=sys_c*10000;
 sys_d=c2d(sys_c,Ts,'tustin');
 [a,b,c,d]=dssdata(sys_d);  % Discrete Controller

% Get descriptor for the error block
 var_names = {'Model Root/error/In1'; ...
    };

 var = mlib('GetTrcVar', var_names);

 mlib('Set','Trigger','ON', ...
    'TriggerLevel',0, ... % Default, can be omitted
    'TriggerEdge','rising', ... % Default, can be omitted
    'TriggerVariable', var(1), ...
    'TraceVars', var, ...
    'NumSamples', 2000, ...
    'Delay',500);

% Start Capture on DS1104
 mlib ('StartCapture');
 while mlib('CaptureState')~=0, end
```

```
% Fetch data to workspace after capture is complete
 error_data = mlib('FetchData');

% Objective Function or Cost Function (minimization method)
f=sum(t.*abs(error_data))*deltT   % ITAE
% f=sum(abs(error_data))*deltT    % IAE
% F=sum((error_data)^2)*deltT       % ISE

%%% BEGIN WRITE DISCRETE STATE SPACE LQG/LTR CONTROLLER %%%
% Write new data to the following area: row 1:2, columns 1:2
% for A matrix
 myMatrix = mlib_matrix('Init',...
 'Model Root/Discrete State-Space\nLQGLTR Controller/A',...
 2, 2, 'Row-Wise');

 AnewData = a ;%[10 20 ; -0.5 1];
 mlib_matrix('Write', myMatrix, [1:2], [1:2], AnewData);

% Write new data to the following area: row 2,1, columns 1:1
% for B matrix
 myMatrix = mlib_matrix('Init',...
 'Model Root/Discrete State-Space\nLQGLTR Controller/B',...
 2, 1, 'Row-Wise');

 BnewData = b; %[ 20 ; -0.5 ];
 mlib_matrix('Write', myMatrix, [1:2], [1:1], BnewData);

% Write new data to the following area: row 1,2, columns 1:2
% for C matrix
 myMatrix = mlib_matrix('Init',...
 'Model Root/Discrete State-Space\nLQGLTR Controller/C',...
 1, 2, 'Row-Wise');

 CnewData = c; %[10 1];
 mlib_matrix('Write', myMatrix, [1:1] ,[1:2], CnewData);

% Write new data to the following area: row 1,1, columns 1:1
% for D matrix
 myMatrix = mlib_matrix('Init',...
 'Model Root/Discrete State-Space\nLQGLTR Controller/D',...
 1, 1, 'Row-Wise');

 DnewData = d; %[10];
 mlib_matrix('Write', myMatrix, [1:1] ,[1:1], DnewData);

% Reset Reference Input or Signal Generator
 siggen_dlqgltr(1,50);  % Amplitude = 1um
                        % Frequency = 50 Hz
```

VITA 2

Ban Fu Chee

Candidate for the Degree of

Master of Science

Thesis: OPTIMIZATION-BASED TUNING OF A DISCRETE LOOP TRANSFER RECOVERY CONTROLLER WITH HARDWARE IN-THE-LOOP

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Perak Darul Ridzuan, Malaysia, on January 5th, 1977, the son of Chee E En and Wong Poh Yook.

Education: Graduated from S.M. Seaport, Petaling Jaya, Selangor, Malaysia in December 1995; attended Metropolitan College, Selangor, Malaysia until December 1996; received a B.S. degree from Oklahoma State University, Stillwater, Oklahoma, in 1999, in Mechanical Engineering. Completed the requirements for the Master of Science degree with a major in Mechanical Engineering with specialization in Dynamic Systems and Control Engineering at Oklahoma State University in December, 2001.

Professional Experience: Worked as teaching and research assistant, Advanced Controls Laboratory, Oklahoma State University, August 1999 to date.

Professional Memberships: American Society of Mechanical Engineers.