Constrained Optimization for Hierarchical Control System Design

by

Michael Brian Jamoom, 2LT, USAF

B.S., Astronautical Engineering
United States Air Force Academy, 1997

SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Master of Science
at the
Massachusetts Institute of Technology

June, 1999

Signature of Author _____
Department of Aeronautics and Astronautics
7 May 1999

Certified by _____
Dr. Marc W. McConley
Charles Stark Draper Laboratory
Thesis Supervisor

Certified by _____
Professor Eric Feron
Assistant Professor of Aeronautics and Astronautics
Thesis Advisor

Accepted by _____
Professor Jaime Peraire
Chairman, Department Graduate Committee

Constrained Optimization for Hierarchical Control System Design

by

Michael Brian Jamoom

Submitted to the Department of Aeronautics and Astronautics on 7 May 1999 in partial fulfillment of the requirements for the degree of Master of Science

Abstract

This thesis presents a constrained convex optimization control design procedure that takes advantage of a decoupled hierarchical system structure. To achieve a hierarchical structure, a system is decoupled into primary channels that have its own dynamics, control actuator, and a resulting controller. These MIMO (multiple input/multiple output) channels are then decoupled into a hierarchical structure of SISO (single input/single output) loops. The constrained optimization procedure is then applied to each SISO loop to determine a solution controller.

The procedure minimizes the closed loop $\mathcal{H}_2$ norm as the primary objective, while presenting the Q-minimization objective as a secondary option. The procedure applies closed loop frequency and transient response constraints to the constrained optimization, which utilizes Q-parameterization of the objective function. The Q-parameter optimization decision variables are the coefficients of a stable orthogonal basis. This basis is constructed using a presented Basis Function Algorithm, designed as a consistent procedure for basis selection. The algorithm is designed to create the most efficient basis from three forms of basis functions: Finite Impulse Response (FIR) functions, Laguerre functions, and Legendre functions. User defined Fixed Pole Model basis functions are also addressed.

The design method is applied using the Draper Laboratory's Structural Control Toolbox (SCTB), a Computer Aided Design (CAD) tool. The design procedure is used on the Draper Small Autonomous Aerial Vehicle (DSAAV) helicopter as a design example. The solution procedure was able to make improvements in the design of the DSAAV forward motion channel. This design utilized the modularity of the hierarchical structure, using both constrained optimization and classical controller designs.

Thesis Supervisor: Dr. Marc W. McConley
Title: Senior Member of the Technical Staff

Thesis Advisor: Professor Eric Feron
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgments

There are many to whom I owe a debt of gratitude for the success and completion of this thesis. I first wish to recognize those who have had direct influence on this success:

It must be understood, I cannot address any success I have in this life without first recognizing my Mom. My father died 20 years ago and my mother's strength, courage, and love focused her life to raise my brothers and myself, completely on her own. Amidst tremendous adversity, my mother ensured we had every opportunity available to succeed in life. Any success my brothers and I enjoy is a testament to her eternal dedication and support. Thank you, Mom. I love you.

I thank the Air Force, Draper Laboratory, and MIT for giving me this educational opportunity that will forever enrich my life and career. Marc McConley, my Draper thesis supervisor, I can never thank enough for his exceptional guidance and expertise that made this thesis a reality. I consider myself very fortunate to be his first master's student. He helped me when I was stuck and gave me the freedom to learn and solve my own problems. I am very grateful for Eric Feron, my MIT thesis advisor, for his diligence to continually have meetings, which motivated my continual progress with this research. His remarkable enthusiasm for research is truly inspirational.

I thank my brothers, Joe and Eric, for their support. Joe: for all those games we watched together (over the phone). Eric: for his newfound interest in sports (we're so proud of you!). I am overjoyed to recognize my sister-in-law Pam, who with Joe, delivered Matthew Daniel on February 23rd, raising the total number of confirmed living Jamooms to six.

This thesis is the monument of my 2 years in Cambridge. Those who have enhanced my life in these years are contributors to this thesis, and must be recognized.

Vik Kanwar, among my oldest friends, deserves much appreciation for his influence on my development into the person I am today. Often the first to hear my *long* stories, Vik taught me guitar (and some music history) and even proofread one of the last drafts of this thesis. He should sleep easy, knowing of his contribution to the military industrial complex. I also thank Marsha Pollard, who I've known just as long, for her almost daily affirmations. I'll always value our friendship, no matter how much it confuses me.

Geoff Billingsley, a fellow Air Force Draper Fellow, and I truly shared the MIT-Draper experience. Geoff, being a great friend, was a motivational workout partner, helpful classmate, and my primary reminder of our exciting future in flight. I also recognize my Draper Fellow friends. The current Draper Fellows are Jim Smith (who will finally get rid of me), Bob Bibeau, Larry McGovern (who was always generous with his knowledge and time), Pat Brown, Chisholm Tracy, HF Vuong, and Nate Shenck, among others. Draper Fellows past include Andy Coop, Nick Nuzzo, Carla Haroz (Coach of the "Draper Monkeys"), Chris "The Man" Dever, Rudy Boehmer, Beau Lintereur, Tony Giustino, and others. I also thank Joe, the fifth floor B-core janitor, for his motivational words every evening of this semester.

I must thank my roommates for enduring me on a daily basis. My first roommate, Andy, is the most stable person I know. He's a great guy that will always command my respect. Then there is Josh Kurland, my old friend and second roommate, who I thank for adding much spice (and spices) to my daily life and showing me how to get a nightlife. Josh was not only gracious enough to introduce me to his closest friends (including the wonderful Sandy and Amanda), but incorporated me into his wacky world of Harvard Law, a very amusing and rewarding experience.

I must recognize the crazy people (besides Vik and Marsha) that are my friends from home, for they remind me of my roots. Jeff Few: for road trips, Hippo in the City, and making every night a Denny's night. Kate Ogden: for the best in air mail and intense intercontinental conversations, I still wonder. Mike Brown: for bowling bets and piñata parties. Lacey Torge: for Plymouth Rock and Sweet Kee (my private dancer). Vicki Mertz: for Atlanta layover visits (Moo!). Also thanks to Hugh Brown and Ish Beloso. I need to make a special note to Rena and the Brugmans, who are essentially an extension of the Jamoom family.

A Ta-Dow goes out to the Fellas: especially Jake Bice, who was the driving force behind all those schemes (DC, MG, the West Point body paint incident, ...) and always brought it strong, and Todd "Treats" Smith. These Fellas constantly reminded me of the ways. Treats and AJ need to be recognized for their role in Rena's birthday surprise for me (Thanks Rena!).

I must thank the Funny Farm for providing me with a sense of community. An enormous amount of admiration and credit goes out to Caroline Crescenzi, who had the vision to appoint me a board member. I also want to thank Funny Farmers Mary, Jody, and Chris. I also need to thank SPAM™ (for inspiration) and my Hormel Contact, Meri, for SPAM™-Man at SPAM™-Jam. I also need to thank Matt Fetzer and Joanna Poggione for their role in the madness.

Some extra thanks for Mona Ortman, Joy Kanwar, Doug Creviston, Cora Stubbs-Dame, Nina Erlich, Tara Harris, Mimi Martin, Aunt Frieda, The Hedins, Kristine Arrieta, Brad Head, Heather Guess, Gail Mader, Sunny Bunny, Fuzzy Eddie, Frankie Coker, the MoD, Josh's buddies: Jason, Jeremy, and Dan; Josh's Dad, Josh's friendly HLS classmates, and the T.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Individual Research and Development project 1012.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Permission is hereby granted by the author to the Massachusetts Institute of Technology to reproduce any or all of this thesis.

Michael B. Jamoom 7 May 1999

Contents

List of Figures

Chapter 1

Introduction

1.1 Historical Overview

Control engineering is rooted in two main methods of solving problems: classical methods and model-based methods. Classical control methods, such as use of root-locus techniques, Bode plots, and Proportional Integral Derivative (PID) controllers infer closed-loop performance and are very effective for single input/single output (SISO) systems. However, applying classical control to multiple input/multiple output (MIMO) systems is of limited value because of the coupling of the inputs and outputs within the dynamics. Also, classical methods deal with design specifications indirectly, requiring the control engineer to iterate by trial and error until a solution is found.

Model-based methods, such as the $\mathcal{H}_\infty$ and Linear Quadratic Gaussian (LQG or $\mathcal{H}_2$) methods, are based on state space representations of the system. These methods map the closed loop system in terms of a metric or norm. Often frequency weights are added to the input and output signals to shape the closed loop response. These weights become the design variables and are adjusted until the desired closed loop performance is achieved. Although model-based methods are effective in controlling MIMO systems, the adjustment of these design parameters is still a difficult and indirect way to design controllers.

With computer technology becoming faster and cheaper, convex optimization has

been researched as a more direct method of controller design. Convex optimization is distinct from non-convex optimization because convex optimization does not have local minima. Therefore, any minimum in a convex problem is a global minimum. Important convex optimization programs are the linear program and the quadratic program. A linear program has a linear objective, such as $c^T x$, and linear constraints. A popular technique for solving linear programs is the Simplex Method [17]. A quadratic program has a quadratic objective function, such as $x^T C x$. If the constant matrix C is positive definite and the constraints are linear, then the quadratic program is convex. In the 1960s, Fegley and colleagues [2], investigated the application of linear and quadratic programming to optimal control problems.

In 1976, Youla first recognized Q (or Youla) parameterization [18]. In the 1980s, Q-parameterization methods of controller design were developed, applying the Q parameter to the closed loop. Gustafson and Desoer generalized the parameterization of Q, representing Q in terms of zero and pole locations [3, 4]. Boyd [1] and Polak [15] used Q-parameterization in a convex optimization control design. This was done through a finite approximation of Q via basis functions. Optimization over all closed loops was reduced to an optimization over all stable Q. This is a foundation of constrained convex optimization.

Most recently, constrained optimization was successfully applied by McGovern, using $\mathcal{H}_2$ and ℓ_1 objectives on an actual hardware system, the structural control of a telescope [9, 10]. Also, Lintereur has successfully applied constrained optimization, using the $\mathcal{H}_2$ objective, on a spacecraft attitude control system [6, 7].

1.2 Contributions

There are two main goals of this thesis. One is to uncover some benefits of decoupling a system into a hierarchical structure and applying the constrained optimization one element at a time. These benefits could range from improving optimization efficiency to taking advantage of modular control design, where controllers for separate modes (within the same channel) could be designed with completely different techniques.

The other goal is to apply constrained optimization to an actual control problem. The actual control problem presented in this thesis is the Draper Small Autonomous Aerial Vehicle (DSAAV), a helicopter.

In addition to the above goals, this thesis makes several further contributions. One such contribution is a procedure, based on constrained optimization, that solves for controllers within the hierarchical structure. Another contribution is the application of the Structural Control Toolbox (SCTB) to the solution procedure. The SCTB, developed at the Draper Laboratory, is a Computer Aided Design (CAD) toolbox which runs under MATLAB and can be very useful for many forms of controller design. This thesis explains how to apply the powerful SCTB, detailed in Chapter 5, and the benefits of this relatively new toolbox. Another useful aspect is an algorithm for choosing the most efficient basis, within the solution procedure, based on the forms of basis functions available in the SCTB. This algorithm aims at reducing the number of basis functions necessary to estimate the objective function in order to improve the speed of the optimization and produce lower order controller solutions.

1.3 Organization

The organization of this thesis falls into seven chapters. The second chapter provides developmental information for constrained optimization. This starts with a detailed explanation of how to formulate Q-parameterization into a constrained optimization problem. It then explains the different options for the portions of the constrained optimization problem: choosing an objective function, applying constraints, and selecting a basis function representation of Q. The chapter ends detailing the model reduction techniques necessary for applying constrained optimization to higher order systems.

Chapter 3 details the decoupled hierarchical system architecture and how to break a coupled system into the hierarchical structure. Chapter 4 details the solution procedure, complete with algorithms for choosing objective functions and setting constraints. There is also a detailed Basis Function Algorithm in this chapter, designed

to find the most efficient basis that will provide a feasible solution. Chapter 5 explains the SCTB and how to apply it to the solution procedure. Chapter 6 applies the solution procedure to the DSAAV helicopter in a control design example. Finally, the conclusions and suggestions for future work are presented in Chapter 7.

Chapter 2

Constrained Optimization

This chapter presents a brief overview of some concepts necessary for the development of the Constrained Optimization solution procedure in this thesis. It will be useful for understanding concepts and notation used in later chapters. The reader may skip this chapter if these are familiar concepts and should refer back as necessary.

A general control optimization problem is presented based on the generic system in Figure 2.1. The closed loop system H is represented by $n_u \times n_y$ compensator K and $n_z \times n_w$ system $H = \mathcal{F}_\ell(P, K)$. Here, $\mathcal{F}_\ell(P, K)$ denotes the linear fractional transformation of the plant P with the compensator K. The objective function that determines how the closed loop performance is measured is $\Phi(H)$. The closed loop system H is constrained to the closed loop constraint set $\mathcal{K}_H$. Enough information is provided to define the following general control optimization problem.

Problem 1 *Given the closed loop constraint set $\mathcal{K}_H$, choose the stabilizing compensator K that solves the following optimization problem, minimizing the closed loop system H:*

$$\begin{aligned} \min_{K\,stabilizing} \quad & \Phi(H) \\ subject\ to \quad & H = \mathcal{F}_\ell(P, K) \in \mathcal{K}_H \end{aligned}$$

Problem 1 is a non-convex optimization problem, which is difficult to solve. Q-parameterization (or Youla Parameterization) allows for parameterization of all sta-

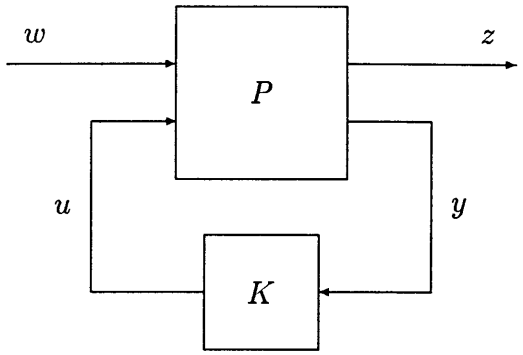

Figure 2.1: General Control Problem

bilizing controllers by the parameter Q. Q lies in a convex set where H is linear in Q, making for a convex optimization problem that is easier to solve.

This thesis will define constrained optimization as the important application of the powerful Q-parameterization allowing for convex optimization formulation. This formulation allows $\mathcal{K}_H$ to be mapped to convex constraints on Q. Every stabilizing compensator is related to Q through a bilinear map. The mathematical formulation is the following optimization problem [1, 7].

Problem 2 *Given the closed loop constraint set $\mathcal{K}_H$ and the set of Q that closes loops in $\mathcal{K}_H$, $\mathcal{K}_Q$, choose the infinite dimensional design variable Q to solve the following optimization problem that minimizes the closed loop system H:*

$$\begin{aligned} \min_{Q \in \mathcal{K}_Q} \quad & \Phi(H) \\ \textit{subject to} \quad & \mathcal{K}_Q = \{Q \textit{ stable} \mid T_1 + T_2 Q T_3 \in \mathcal{K}_H\} \\ & H = T_1 + T_2 Q T_3 \end{aligned}$$

The equation that defines the closed loop system H is the Q-parameterization of H, later defined as Equation 2.1.

This chapter starts, in Section 2.1, with a discussion of Q-parameterization, the parameterization technique that makes constrained optimization possible. The following section, Section 2.2, defines the possible objective functions used to determine the performance of the optimization. Section 2.3 details basis functions and how they represent Q. This section explains each of the four basis functions available to create

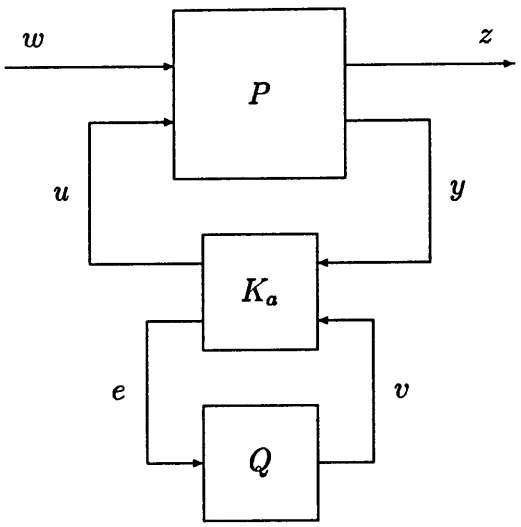

Figure 2.2: Q-Parameterization.

the orthogonal basis for the decision variables. Section 2.4 describes how constraints on the frequency and time response are formulated into constrained optimization constraints. The final section in the chapter explains the model reduction techniques necessary for applying constrained optimization to higher order systems.

2.1 Q-Parameterization

The first step in solving the constrained optimization problem is to find the Q-Parameterization, or the Youla Parameterization [8, 18], of the closed loop system. The closed loop system is represented by $n_u \times n_y$ compensator $K = \mathcal{F}_\ell(K_a, Q)$ and $n_z \times n_w$ system $H = \mathcal{F}_\ell(P, K)$ in Figure 2.2.

2.1.1 Q Representation of the Closed Loop System

Q-parameterization is achieved by developing the stabilizing augmented observer-based controller K_a, with augmented control input v (added to the actuator command signal) and augmented observer error output e. The new output e becomes the input and the new input v becomes the output for the unknown stable and realizable Q ($v = Qe$). For any stable Q, this does not change the stability of the closed loop

system. Furthermore, it is well known that the closed loop map from v to e is zero. This allows Q to appear linearly in the closed loop system H:

$$H = T_1 + T_2 Q T_3 \tag{2.1}$$

where T_1 is the open loop map from w to z, T_2 is the map from w to e, and T_3 is the map from v to z. H can now be represented by Equation 2.1 as opposed to the linear fractional representation $\mathcal{F}_\ell(P, K)$. The parameterization in Equation 2.1, affine in Q, represents all stabilizing controllers, allowing a search over all realizable closed loop systems to be replaced by a search over all stable Q. Similar detailed discussion of Q-parameterization can be found in [1, 5, 6, 7, 8, 9, 18].

2.1.2 Q-Parameterization in State Space

The plant P maps the exogenous input w and control u to the regulated output z and measurement y as follows:

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} = P \begin{bmatrix} w \\ u \end{bmatrix}. \tag{2.2}$$

For the compensator K, the closed loop system H is:

$$H = P_{11} + P_{12} K (I - P_{22} K)^{-1} P_{21} = \mathcal{F}_\ell(P, K) \tag{2.3}$$

For K to be internally stabilizing, H can be replaced by the Q-parameterization of Equation 2.1.

The Q parameter can be derived from any nominally stabilizing controller. The state space description of the discrete plant P is:

$$\begin{bmatrix} x[k+1] \\ z[k] \\ y[k] \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x[k] \\ w[k] \\ u[k] \end{bmatrix}. \tag{2.4}$$

A controller gain matrix F and observer gain matrix G are chosen to place the eigenvalues of $(A - B_2 F)$ and $(A - G C_2)$ inside the unit circle to create a stabilizing model-based controller. To augment the controller with Q as in Figure 2.2, an input

v is added to the actuator command signal. The output measurement residual e and v are described in the following equations:

$$\begin{aligned} v[k] &= Qe[k] \\ u[k] &= -F\hat{x}[k] + v[k] \\ e[k] &= y[k] - C_2\hat{x}[k] - D_{22}u[k] \end{aligned} \tag{2.5}$$

where $\hat{x}$ represents the state estimate. A stable Q will not affect the stability of the closed loop system. After including the contribution of v to the state estimate equation, the state space representation of the augmented controller K_a is:

$$\begin{bmatrix} \hat{x}[k+1] \\ u[k] \\ e[k] \end{bmatrix} = \begin{bmatrix} A - B_2F - GC_2 + GD_{22}F & G & B_2 - GD_{22} \\ -F & 0 & I \\ -C_2 - D_{22}F & I & -D_{22} \end{bmatrix} \begin{bmatrix} \hat{x}[k] \\ y[k] \\ v[k] \end{bmatrix} \tag{2.6}$$

where the space of all stabilizing controllers is spanned by $K = \mathcal{F}_\ell(K_a, Q)$ if Q is stable.

Algebraically eliminating u and y in a lower linear fractional transformation of P and K_a results in the following state space representation of T:

$$\begin{bmatrix} x[k+1] \\ \tilde{x}[k+1] \\ z[k] \\ e[k] \end{bmatrix} = \begin{bmatrix} A + B_2F & -B_2F & B_1 & B_2 \\ 0 & A + GC_2 & B_1 + GD_{21} & 0 \\ C_1 + D_{12}F & -D_{12}F & D_{11} & D_{12} \\ 0 & C_2 & D_{21} & 0 \end{bmatrix} \begin{bmatrix} x[k] \\ \tilde{x}[k] \\ w[k] \\ v[k] \end{bmatrix} \tag{2.7}$$

where the state estimate error $\tilde{x} = x - \hat{x}$. The state estimate error is uncontrollable from v and the state x is unobservable from e. Therefore, the transfer function from v to e is zero. Figure 2.3 shows the relationship between T and Q. The closed loop from w to z is:

$$H(z) = T_1 + T_2Q(I - T_4Q)^{-1}T_3 = \mathcal{F}_\ell(T, Q) \tag{2.8}$$

where $T = \begin{bmatrix} T_1 & T_2 \\ T_3 & T_4 \end{bmatrix}$. Since T_4, the transfer function from v to e, is zero, Equation 2.8 reduces to Equation 2.1, where the elements of T are:

$$T_1 = \left(\begin{bmatrix} A + B_2F & -B_2F \\ 0 & A + GC_2 \end{bmatrix}, \begin{bmatrix} B_1 \\ B_1 + GD_{21} \end{bmatrix}, \begin{bmatrix} C_1 + D_{12}F & -D_{12}F \end{bmatrix}, D_{11} \right) \tag{2.9}$$

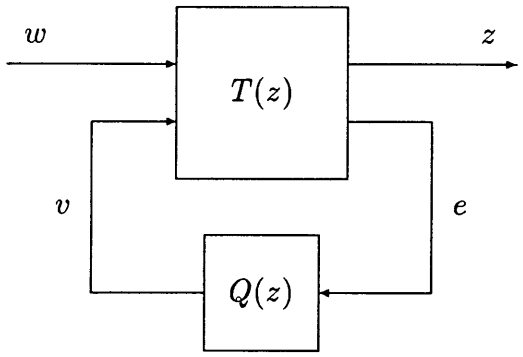

Figure 2.3: T and Q Depiction of System

$$T_2 = (\ A + B_2F,\ \ B_2,\ \ C_1 + D_{12}F,\ \ D_{12}\ ) \tag{2.10}$$

$$T_3 = (\ A + GC_2,\ \ B_1 + GD_{21},\ \ C_2,\ \ D_{21}\ ). \tag{2.11}$$

If the open loop plant is stable, the controller gain matrix F and observer gain matrix G can go to zero. This results in $T_1 = P_{11}$, $T_2 = P_{12}$, and $T_3 = P_{21}$.

2.1.3 Infinite Dimensional Representation of $Q(z)$

Q can be represented by an infinite number of basis functions and coefficients. This infinite representation of Q is as follows:

$$Q(z) = \sum_{n=0}^{\infty} F_n(z)\phi_n \tag{2.12}$$

where $F_n(z)$ is a complete set of basis functions and ϕ_n represents the free basis function coefficients. To be complete, $F_n(z)$ must span the space of all stable realizable transfer functions. The relationship between this representation of Q, basis functions, and optimization, including the finite representation of $Q(z)$, is discussed in Section 2.3.

2.2 Objective Functions

The objective function defines how the constrained optimization measures system performance. The choice of objective function can radically alter how the problem

is formulated and the success of the resulting controller. There are various types of objective functions, and the SCTB looks at three: $\mathcal{H}_2$ minimization, Q-minimization, and ℓ_1 minimization. This thesis will investigate the first two: $\mathcal{H}_2$ minimization and Q-minimization.

2.2.1 $\mathcal{H}_2$ Minimization

The most effective way to initially design a constrained optimization controller is through use of an $\mathcal{H}_2$ objective function [5]. This objective function will minimize the $\mathcal{H}_2$ norm of the transfer matrix from disturbance and noise inputs to error and control outputs. Looking back to Figure 2.2, there are two kinds of plant inputs: the exogenous input w, containing disturbances and noises, and the controller output u. Likewise there are two kinds of plant outputs: the regulated output z, containing errors and control signals, and the controller input y.

Q-Parameterization of $\mathcal{H}_2$ Norm

One of the most popular standards of measurement of a model-based system is the system $\mathcal{H}_2$ norm. This norm is the root mean squared (RMS) value of the system output to a given unit variance (Gaussian) white noise input. The $\mathcal{H}_2$ norm of a MIMO system is:

$$\|H\|_2 = \sqrt{\mathrm{Tr}\frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{i\omega})H^*(e^{i\omega})d\omega} = \sqrt{\sum_{i=1}^{n_z}\sum_{j=1}^{n_w}\sum_{k=0}^{\infty} h_{ij}[k]^2} \tag{2.13}$$

where $h[k]$ represents the closed loop impulse response.

The Q-parameterization representation of the $\mathcal{H}_2$ norm is based on the following Q-parameterization representation of the closed loop impulse response:

$$h_{ij}[k] = t_{1,ij}[k] + (\sum_{p=1}^{n_u}\sum_{q=1}^{n_y} t_{2,ip} * \sum_{n=0}^{N-1} f_n\phi_{pqn} * t_{3,qj})[k] \tag{2.14}$$

where $t_{1,ij}$, $t_{2,ij}$, and $t_{3,ij}$ are the impulse responses of the (i,j) elements of T_1, T_2, and T_3 respectfully and the $f_n\phi_{pqn}$ term provides the impulse response of Q for N finite number of basis functions f_n and free coefficients ϕ_{pqn}.

$\mathcal{H}_2$ Objective Function

The $\mathcal{H}_2$ suboptimal objective function is the following quadratic function of ϕ_{pq}, the vector representation of ϕ_{pqn}:

$$\min_{\phi} \sum_{p=1}^{n_u} \sum_{q=1}^{n_y} \phi_{pq}^T M_{pq} \phi_{pq} + g_{pq}^T \phi_{pq} \tag{2.15}$$

where the terms m_{ijpq}, g_{pq}, and M_{pq} are defined as follows:

$$m_{ijpq} = t_{2,ip} * t_{3,qj} * [f_0 \ f_1 \ \cdots \ f_{N_1}] \tag{2.16}$$

$$g_{pq} = 2 \sum_{i=1}^{n_z} \sum_{j=1}^{n_w} t_{1,ij}^T m_{ijpq} \tag{2.17}$$

$$M_{pq} = \sum_{i=1}^{n_z} \sum_{j=1}^{n_w} m_{ijpq}^T m_{ijpq}. \tag{2.18}$$

Once constraints are added, the $\mathcal{H}_2$ minimization constrained optimization problem is represented by the convex quadratic program [5]:

$$\begin{aligned} \min_{\phi} \quad & \phi^T M \phi + g^T \phi \\ \text{subject to} \quad & A\phi \leq b \end{aligned}$$

where A and b are derived from user defined constraints. Constraints are formulated in Section 2.4. Similar explanations of the $\mathcal{H}_2$ objective function are found in [5, 7, 9].

2.2.2 Q-Minimization

The Q-minimization technique attempts to improve upon an existing stabilizing nominal controller K_{nom}. This form of constrained optimization designs an augmentation to the nominal controller. This is illustrated in Figure 2.4, where the nominal plant P and K_{nom} are closed to form the nominal closed loop H_{nom}. The constrained optimization design is the augmentation controller K_{aug}.

With the stability of the nominal closed loop system H_{nom} as a given, an observer based Q-parameterization can occur without the observer and state feedback gains previously required to stabilize the system. This stabilizing controller K_s is then combined with Q to complete the augmentation controller.

$$K_{aug} = \mathcal{F}_\ell(K_s, Q) \tag{2.19}$$

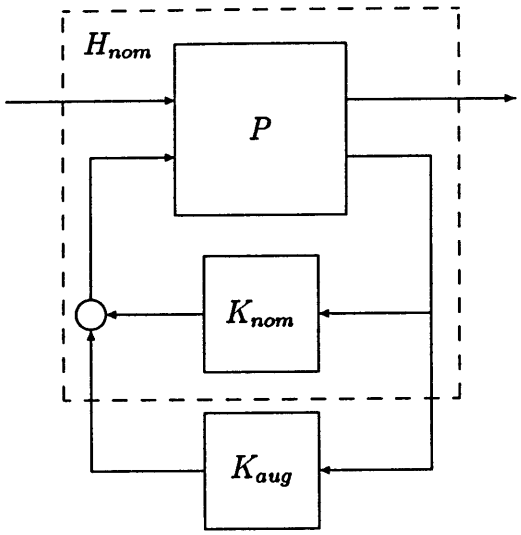

Figure 2.4: Augmented Controller Design

The nominal controller is then added to complete the final augmented controller K.

$$K = K_{nom} + K_{aug} \tag{2.20}$$

It cannot be stressed enough that a good K_{nom} is necessary. As shown in Equation 2.20, any undesirable characteristics of K_{nom} will very likely be retained in the final design.

Q-Minimization Objective Function

A good K_{nom} is required because Q-Minimization tries to minimize the $\mathcal{H}_2$ norm of Q, as opposed to $\mathcal{H}_2$ minimization, which attempts to minimize the $\mathcal{H}_2$ norm of the closed loop system. This way, the modifications to K_{nom} are penalized, with the worst case scenario of $Q = 0$ resulting in an unaugmented K_{nom} as the final controller. For orthonormal basis functions, the $\mathcal{H}_2$ norm of Q is defined as follows:

$$\|Q(z)\|_2 = \sqrt{\sum_{i=1}^{n_u}\sum_{j=1}^{n_y}\sum_{n=0}^{N-1}\phi_{ijn}^2} \tag{2.21}$$

Now, the suboptimal Q-minimization objective function is:

$$\min_{\phi} \phi^T\phi \tag{2.22}$$

with ϕ being the vector representation of ϕ_{ijn}. Similar explanations of the Q-minimization objective function are found in [5, 7, 9].

2.3 Basis Functions

In the Q-parameterization-based optimal control formulation of Problem 2, Q represented the infinite dimension design variable. As depicted in Equation 2.12, Q can be defined as a function of coefficients ϕ and basis functions $F_n(z)$. This can be easily approximated to a finite representation of Q with a finite number of basis functions.

There are several different forms of basis functions. This thesis will focus on the four kinds of basis functions that are available in the SCTB: Finite Impulse Response (FIR), Laguerre functions, Legendre functions, and the user defined Fixed Pole Model (FPM) [5]. Further discussions on FIR and Lagaurre basis functions are found in [5, 7, 9]

2.3.1 Finite Representation of $Q(z)$

To make the optimization problem tractable, it is necessary to make a finite approximation of Q. A finite approximation of Equation 2.12 is defined below:

$$Q(z) \approx \sum_{n=0}^{N-1} F_n(z)\phi_n \tag{2.23}$$

where $F_n(z)$ is the sequence of N stable basis functions. Now, ϕ_n, the corresponding set of free coefficients, becomes the decision variables for the objective functions described in Section 2.2. It should be noted that if $F_n(z)$ is orthonormal and N goes to infinity, Equation 2.23 can represent any stable transfer function.

2.3.2 Finite Impulse Response

Simplicity makes the FIR the most popular basis. The FIR formulation assumes the impulse response of Q is zero after N time steps. The FIR representation of $Q(z)$ is:

$$Q(z) = \sum_{n=0}^{N-1} \frac{\phi_n}{z^n}. \tag{2.24}$$

This basis is orthonormal, satisfying the following relationship:

$$\sum_{k=0}^{\infty} f_i[k] f_j[k] = \begin{cases} 1 \text{ for } i = j \\ 0 \text{ for } i \neq j \end{cases} \tag{2.25}$$

where the time domain form of the functions are $f_i[k] = \delta(i-k)$.

The FIR basis is best used for modeling fast, highly damped systems. However, if the optimal Q contains lightly damped modes or very slow dynamics, the FIR basis is less effective. Slow systems can require an inefficiently high number of basis functions N, limiting the effectiveness of the FIR basis. It is suggested that FIR basis functions are most effective in combination with other forms of basis functions [5].

2.3.3 Laguerre Functions

Laguerre functions are a generalization of the FIR basis. The difference between the two basis forms is Laguerre functions have their poles placed at a specified time scale a, given that $|a| < 1$. The FIR has all poles placed at the origin.

The discrete Laguerre basis representation of $Q(z)$ is:

$$Q(z) = \sum_{n=0}^{N-1} \frac{\sqrt{1-a^2}}{z-a} \left(\frac{1-az}{z-a}\right)^{n-1} \phi_n. \tag{2.26}$$

Notice that for $a = 0$, the Laguerre basis is equivalent to the FIR basis of Equation 2.24. Every transfer function has an associated optimal time scale a. Similar to FIR, the Laguerre basis functions are orthonormal to each other and restrict all of the poles to one location, but the user specifies the location through the choice of a.

Selection of the Time Scale a

The number of basis functions N necessary to adequately approximate the optimal Q is directly related to the choice of a. A poorly chosen a will result in a very large basis, where a well chosen a will keep the basis size small. A well chosen a will place it close to the dominant dynamics of the optimal Q. However, selection of the best time scale and corresponding number of basis functions is hardly intuitive and often requires several design iterations. The SCTB suggests choosing a time scale with a frequency in a region where the problem is highly constrained in the frequency domain [5].

2.3.4 Legendre Functions

Legendre functions are an orthonormal set of basis functions with poles spread over a range of frequencies. This differs from the FIR and Laguerre basis, which have their functions located at the same pole. The discrete Legendre basis representation of $Q(z)$ is:

$$Q(z) = \sum_{n=0}^{N-1} \frac{\prod_{i=0}^{n-1}(1 - a_i z)}{\prod_{j=0}^{n}(z - a_j)} \phi_n \tag{2.27}$$

where $a_k = ae^{-(k+0.5)}$ and $|a| < 1$.

Selection of the Time Scale a

As with Laguerre functions, the time scales a_i determine the location of the basis poles. Poles of Legendre functions are distributed across the frequency band at 0.5, 1.5, 2.5, 3.5, ..., $(N - 0.5)$ times the frequency associated with the chosen time scale a. For example, if the chosen a has a frequency of 10 Hz, the corresponding Legendre basis would have poles located at 5 Hz, 15 Hz, 25 Hz, 35 Hz, ..., $(10N - 5)$ Hz. The SCTB User's Guide suggests choosing a time scale frequency no higher than the frequency of the lowest expected (non-integrator) controller pole or zero [5]. With their dynamics distributed across a range of frequencies, Legendre functions provide for a very flexible basis. This flexibility makes the Legendre functions a great initial guess for a basis.

2.3.5 Fixed Pole Model

The fixed pole model (FPM) basis allows the user to determine the pole locations of the basis. The SCTB gives the user the opportunity to define FPM poles based on the plant poles and the Q poles, from the most recent design of Q [5]. More information on how the SCTB allows the user to use the plant poles and Q poles can be found in Chapter 5 or [5].

2.4 Closed Loop Constraints

After defining the objective function and basis, the remaining task to complete the formulation of the constrained optimization problem is to define constraints. For the purposes of this thesis, constraints are either applied to the closed loop frequency response or the closed loop time response.

2.4.1 Frequency Response Constraints

Frequency response constraints allow the user to improve such system characteristics as command following and stability. The frequency constraints are applied as an upper bound $\gamma_{ij}(\omega)$ to the magnitude of a single input single output (SISO) transfer function H_{ij} in the multiple input multiple output (MIMO) system H:

$$|H_{ij}(\omega)| \leq \gamma_{ij}(\omega) \tag{2.28}$$

where H_{ij} is parameterized and discretized as follows:

$$H_{ij}(z) = T_{1,ij} + \sum_{p=1}^{n_u} \sum_{q=1}^{n_y} \sum_{n=0}^{N-1} T_{2,ip} F_n \phi_{pqn} T_{3,qj}. \tag{2.29}$$

Breaking $H_{ij}(\omega)$ down into real and imaginary components, the constraint in Equation 2.28 is equivalent to:

$$\mathrm{Re}[H_{ij}(\omega)] \cos\theta + \mathrm{Im}[H_{ij}(\omega)] \sin\theta \leq \gamma_{ij}(\omega) \; \forall \theta \in [0, 2\pi). \tag{2.30}$$

In [5, 7], this form of magnitude constraint is approximated by a finite number of linear constraints. This is done by choosing a discrete set of evenly spaced θ between 0 and 2π:

$$\mathrm{Re}[H_{ij}(\omega)] \cos\theta_{n,N} + \mathrm{Im}[H_{ij}(\omega)] \sin\theta_{n,N} \leq \gamma_{ij}(\omega) \cos\frac{\pi}{N} \tag{2.31}$$

where $\theta_{n,N} = \frac{2n\pi}{N}$ for $n = 1, \ldots, N$.

After defining $S_{ijpq}(\omega)$ to be:

$$S_{ijpq}(\omega) = T_{2,ip}(\omega)[F_0(\omega) \cdots F_{N-1}(\omega)] T_{3,qj}(\omega) \tag{2.32}$$

and defining $L_{ij}(\omega)$ to be:

$$L_{ij}(\omega) = \gamma_{ij} \cos \frac{\pi}{N} - \mathrm{Re}[T_{1,ij}(\omega)] \cos \theta_{n,N} - \mathrm{Im}[T_{1,ij}(\omega)] \sin \theta_{n,N} \tag{2.33}$$

the approximate constraint of Equation 2.31 corresponds to a finite number of linear scalar constraints on ϕ:

$$\sum_{p=1}^{n_u} \sum_{q=1}^{n_y} (\mathrm{Re}[S_{ijpq}(\omega)] \cos \theta_{n,N} + \mathrm{Im}[S_{ijpq}(\omega)] \sin \theta_{n,N}) \phi_{pq} \leq L_{ij}(\omega). \tag{2.34}$$

With ϕ being the vector of ϕ_{pq}'s, Equation 2.34 is now written as $A_{freq}\phi \leq b_{freq}$ and can be applied to the constrained optimization problem.

2.4.2 Time Response Constraints

Constraints on time responses allow the user to try to improve output characteristics to a given input disturbance. For a closed loop MIMO system $H(z)$, time constraints, in the form of upper bounds γ_{up} and lower bounds γ_{low}, are applied to the transient response of the SISO transfer function $H_{ij}(z)$:

$$\gamma_{low,ij}[k] \leq z_{ij}[k] \leq \gamma_{up,ij}[k]. \tag{2.35}$$

The output z_{ij} from a given disturbance w_j is found by the following convolution:

$$z_{ij}[k] = (h_{ij} * w_j)[k] \tag{2.36}$$

where $h_{ij}[k]$ is the impulse response of $H_{ij}(z)$. The transient response constraint of Equation 2.35 can be depicted as a constraint on ϕ:

$$(\gamma_{low,ij} - w_j * t_{1,ij})[k] \leq \sum_{p=1}^{n_u} \sum_{q=1}^{n_y} (w_j * m_{ijpq})[k] \phi_{pq} \leq (\gamma_{up,ij} - w_j * t_{1,ij})[k] \tag{2.37}$$

where m_{ijpq} is defined in Equation 2.16. With ϕ being the vector of ϕ_{pq}'s, Equation 2.37 is now written as $A_{time}\phi \leq b_{time}$ and can be applied to the constrained optimization problem.

2.5 Model Reduction

As discussed in [1], a post optimization controller K will generally have the order of the plant plus the entire order of Q, which is based on the number of basis functions used to estimate Q in Section 2.3. Often, the resulting K has too high an order for implementation. If the user does not wish to implement the high order K, a form of model reduction is necessary for obtaining a reduced compensator K_{red} of acceptable order.

There are many forms of model reduction. For the purposes of this thesis, model reduction will be reduced to two options: Balance and Truncate and Fractional Balanced Reduction (FBR). The Balance and Truncate method is chosen for its widely accepted use. FBR, an extension of the Balance and Truncate method, is chosen because of its applicability to models with unstable modes.

2.5.1 Balance and Truncate

Balance and Truncate is the most widely used method for model reduction. Based on Moore's algorithm [13], Balance and Truncate relies on a Cholesky decomposition to factor the controllability and observability grammians, which then balance the system.

Balanced Realization

The first step of Balance and Truncate is to arrange the plant into an internally balanced form. A given nth order minimal stable model $G(s)$ is defined as follows:

$$G(s) = C(sI - A)^{-1}B \tag{2.38}$$

where (A, B, C) is the state space representation of $G(s)$ and I is the nth order identity matrix. The first step of balancing the plant is to solve for the observability grammian W_O and the controllability grammian W_C in the following Lyapunov functions:

$$AW_O + W_O A^T + BB^T = 0 \tag{2.39}$$

$$A^T W_C + W_C A + C^T C = 0 \tag{2.40}$$

where W_O and W_C are unique, symmetric, and positive definite.

The next step is to decompose W_O with the Cholesky decomposition:

$$W_O = R^T R. \tag{2.41}$$

This is used to form the matrix $RW_C R^T$, which is subsequently diagonalized:

$$RW_C R^T = U\Sigma^2 U^T \tag{2.42}$$

where $U^T U = I$ and Σ is defined as a diagonal matrix of Hankel singular values, ordered greatest to least ($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$):

$$\Sigma = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_n\}. \tag{2.43}$$

A Balancing Transformation Matrix Φ_{bal} is defined as:

$$\Phi_{bal} = \Sigma^{-\frac{1}{2}} U^T R. \tag{2.44}$$

Now $G(s)$ can be transformed into the minimal, internally balanced model $G_{bal}(s)$:

$$G_{bal}(s) = C_{bal}(sI - A_{bal})^{-1} B_{bal} \tag{2.45}$$

where $A_{bal} = \Phi_{bal} A \Phi_{bal}^{-1}$, $B_{bal} = \Phi_{bal} B$, and $C_{bal} = C\Phi_{bal}^{-1}$. The condition for any internally balanced plant, such as $G_{bal}(s)$, is that the corresponding Σ is the solution to both the controllability and observability Lyapunov functions:

$$A_{bal}\Sigma + \Sigma A_{bal}^T + B_{bal} B_{bal}^T = A_{bal}^T \Sigma + \Sigma A_{bal} + C_{bal}^T C_{bal} = 0 \tag{2.46}$$

Truncation of Balanced Model

The next step is to determine the size of the reduced model r, where $r < n$. Now the internally balanced $G_{bal}(s)$ can be represented by:

$$A_{bal} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad B_{bal} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \quad C_{bal} = [C_1 \;\; C_2] \tag{2.47}$$

where A_{11} is size $r \times r$, B_1 is size $r \times u$ (u inputs), and C_1 is size $y \times r$ (y outputs). Likewise, Σ is also partitioned:

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \tag{2.48}$$

where $\Sigma_1 = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ and $\Sigma_2 = \text{diag}\{\sigma_{r+1}, \sigma_{r+2}, \ldots, \sigma_n\}$.

Taking the components of Equation 2.47, the truncated reduced rth order model G_{red} is defined:

$$G_{red}(s) = C_1(sI - A_{11})^{-1}B_1 \tag{2.49}$$

where I is the rth order identity matrix. This truncated rth order model satisfies Equation 2.46 in the following manner:

$$A_{11}\Sigma_1 + \Sigma_1 A_{11}^T + B_1 B_1^T = A_{11}^T\Sigma_1 + \Sigma_1 A_{11} + C_1^T C_1 = 0 \tag{2.50}$$

and is, therefore, internally balanced.

Balance and Truncate only reduces stable modes of the model. Unstable modes are removed before reduction and returned to the model after reduction. Balance and Truncate can also take advantage of frequency weighting [5]. This allows the modes to be emphasized based on frequency. If a lowpass filter is used as a frequency weight, the model reduction emphasizes, and likely keeps, the low frequency states and truncates the high frequency states. Further commentary on the Balance and Truncate method can be found in [5, 11, 12, 13].

2.5.2 Fractional Balanced Realization

The Fractional Balanced Reduction method of model reduction was developed by Meyer [11, 12], extending the Balance and Truncate method. FBR allows for reduction of all modes of a system, where Balance and Truncate fails to reduce unstable modes. FBR takes advantage of coprime fractional representations of the plant, generalizing Balance and Truncate for all models.

FBR starts with the minimal state space representation of model $G(s)$, as described in Equation 2.38. The first departure from Balance and Truncate is the need

to solve the following algebraic Riccati equation (ARE):

$$PA + A^T P - PBB^T P + C^T C = 0 \tag{2.51}$$

where P is the positive definite solution. This step is based on a coprime fractionalization, with the derivation found in [11, 12]. A parameter C_K is defined as follows:

$$C_K = -B^T P. \tag{2.52}$$

A new dynamics matrix $\bar{A}$ is defined as:

$$\bar{A} = A + BC_K. \tag{2.53}$$

An entirely new state space representation is then defined as:

$$\bar{G}(s) = \bar{C}(sI - \bar{A})^{-1}B \tag{2.54}$$

where $\bar{C} = \begin{bmatrix} C \\ C_K \end{bmatrix}$. It should be noted that Equation 2.51 is now:

$$P\bar{A} + \bar{A}^T P + C_K^T C_K + C^T C = 0 \tag{2.55}$$

making P the observability grammian of this system.

FBR now returns to Balance and Truncate procedures. Using the procedure in Section 2.5.1, balance the system $\bar{G}(s)$ to get the balanced model $\bar{G}_{bal}(s)$. In the balanced model, $\bar{C}_{bal} = \begin{bmatrix} C_{bal} \\ C_{K,bal} \end{bmatrix}$. Then partition $\bar{G}_{bal}(s)$ as in Equation 2.47, noting that $\bar{C}_1 = \begin{bmatrix} C_1 \\ C_{K,1} \end{bmatrix}$.

The second departure from Balance and Truncate is the partition manipulation procedure, based on coprime fractions and Graph Hankel singular vales, derived in [11, 12]. The manipulation is simplified to defining A_{11} as follows:

$$A_{11} = \bar{A}_{11} - B_1 C_{K,1} \tag{2.56}$$

where $B_1 = \bar{B}_1$. Similar to Balance and Truncate, the rth order FBR reduced model G_{red} is now:

$$G_{red}(s) = C_1(sI - A_{11})^{-1}B_1 \tag{2.57}$$

where I is the rth order identity matrix.

Chapter 3

Decoupled Hierarchical System Architecture

This chapter presents the decoupled hierarchical system architecture, the foundation of the solution procedure found in the next chapter. Consider the example system depicted in Figure 3.1, there are four channels, one for each Cartesian position (x,y,z) and the heading ψ. Each channel has an associated primary control actuator. For example, the control actuator u_1 is associated with the x channel. Since each channel has an associated control actuator, it is assumed each channel can have its own associated controller. For each channel, the dynamics are entirely coupled. The first decoupling separates the dynamics of the entire system into individual channels. Within each channel, more coupled dynamics exist. For the purposes of this thesis, a coupled system will be the coupled channel (which is decoupled from the other channels) as depicted in Figure 3.2. A decoupled hierarchical system will be decoupled within the channel. Here, the pertinent states are decoupled from one another as in Figure 3.3.

There are several reasons for decoupling the system into the hierarchical structure. There might be bandwidth separation between the modes of the dynamics. Decoupling would give the control engineer the advantage of control design, one mode at a time, with the opportunity to design for each bandwidth separately. This also means that each mode can be controlled individually using multiple control techniques. The

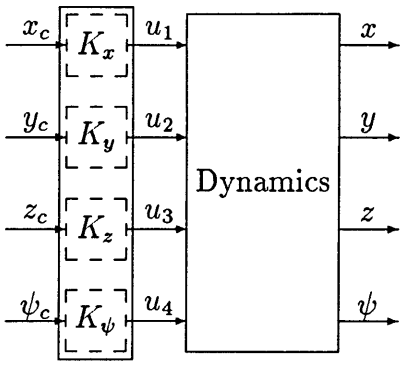

Figure 3.1: Entire System of Coupled Channels

hierarchical structure also simplifies the problem. Solving several decoupled SISO (single input/single output) systems is often a simpler and more computationally efficient process than solving a coupled MIMO system.

3.1 System Architecture

A coupled MIMO system should have a structure similar to Figure 3.2. In this structure, the input is the commanded state x_{1c}. The commanded state is fed into the MIMO plant $P(s)$, where the coupled states (x_1, x_2, x_3) are the outputs. The outputs are fed back into the controller $K(s)$, which then feeds the control input u into the plant. Coupling may exist between internal plant states.

For the purposes of this thesis, a decoupled system will have the hierarchical structure depicted in Figure 3.3. Here, the MIMO system is broken down into SISO inner loops. Each pertinent coupled state is allocated its own loop. The order of the loops is user defined. However, it is suggested that the state loops are ordered by having the faster and higher order derivative states in the inner loops. For example, when dealing with aircraft, tracking position and velocity profiles may be primary performance objectives. In this case, the faster attitude states (roll, pitch, yaw, and associated rates) are allocated to the more internal loops and the position states (position, velocity, and acceleration) are allocated to the outermost loops. In addition, attitude rate state loops are inside the attitude state loops and the acceleration loop

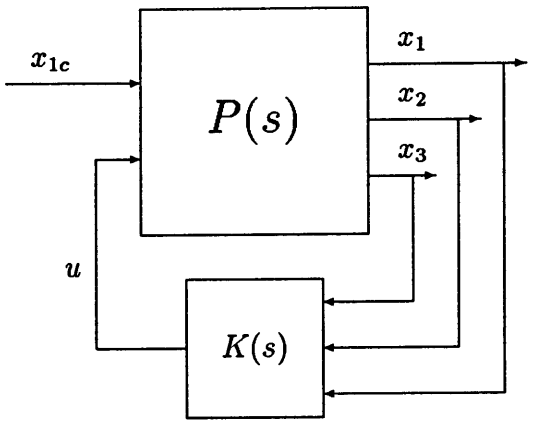

Figure 3.2: Coupled System Architecture for a Decoupled Channel

comes before the velocity loop, which comes before the position loop.

The plant model is partitioned into a series of transfer functions from the control u to the pertinent states. The error of the innermost state x_{3err} is the input for the innermost loop controller $K_3(s)$, that determines the control input u. The control input is then sent to the innermost plant $P_3(s)$, which sends out the innermost state as an input to the next loop's plant $P_2(s)$ and as negative feedback in the innermost loop. In the subsequent outer loops, the commanded states become the outputs of the outer controllers, with the commanded state of the outer loop x_{1c} acting as the original input to the system. Likewise, the outermost state x_1 is the last output of the system.

3.2 Example of Coupled Dynamics

The mathematical decoupling of the system starts with the state space description of the coupled dynamics $P(s)$. For the purposes of the rest of this chapter, the example system will be the system depicted in Figure 3.2 and Figure 3.3 with three pertinent states (x_1, x_2, x_3) and two internal states (x_i and x_{ii}). The state space representation of the coupled system $P(s)$ is:

$$\dot{\underline{x}} = A\underline{x} + Bu \tag{3.1}$$

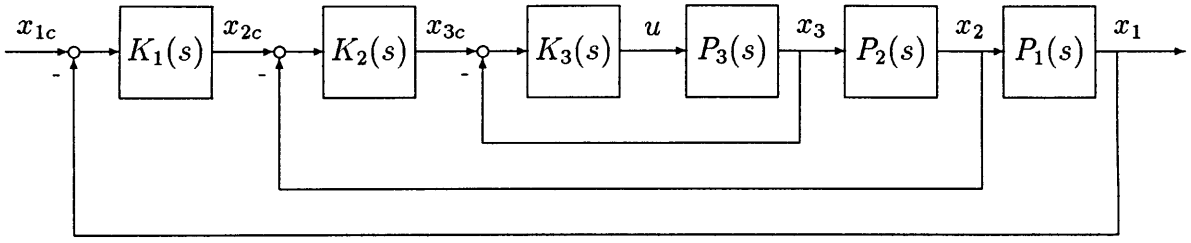

Figure 3.3: Decoupled System Architecture

where the dynamics matrix A and control matrix B are:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{1i} & a_{1ii} \\ a_{21} & a_{22} & a_{23} & a_{2i} & a_{2ii} \\ a_{31} & a_{32} & a_{33} & a_{3i} & a_{3ii} \\ a_{i1} & a_{i2} & a_{i3} & a_{i,i} & a_{i,ii} \\ a_{ii1} & a_{ii2} & a_{ii3} & a_{ii,i} & a_{ii,ii} \end{bmatrix} \quad B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_i \\ b_{ii} \end{bmatrix} \tag{3.2}$$

with the control as a sole control input u and the states are $\underline{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_i \;\; x_{ii}]^T$.

3.3 The Innermost Loop

To construct the plants of the inner loops, the dynamics that couple the pertinent states are removed and are then returned with each outer plant. For the innermost loop, the system is truncated into the innermost pertinent state and the internal states. The input to the system is the commanded innermost state.

The generic innermost loop is depicted in Figure 3.4 for R pertinent states. $R = 3$ for the current example, where the commanded state input is x_{3c}. The state space representation of the innermost system $G_3(s)$, as described in Figure 3.5, has the truncated states $\underline{x}_3 = [x_3 \;\; x_i \;\; x_{ii}]^T$. This state space system is as follows:

$$\dot{\underline{x}}_3 = A_3\underline{x}_3 + B_3\underline{u}_3 \tag{3.3}$$

$$\underline{y}_3 = C_3\underline{x}_3 + D_3\underline{u}_3 \tag{3.4}$$

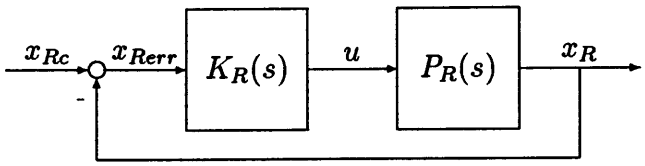

Figure 3.4: Generic Innermost Loop

where

$$A_3 = \begin{bmatrix} a_{33} & a_{3i} & a_{3ii} \\ a_{i3} & a_{i,i} & a_{i,ii} \\ a_{ii3} & a_{ii,i} & a_{ii,ii} \end{bmatrix} \quad B_3 = \begin{bmatrix} b_3 & 0 \\ b_i & 0 \\ b_{ii} & 0 \end{bmatrix}$$
$$C_3 = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad D_3 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.5}$$

with the inputs $\underline{u}_3 = [u \;\; x_{3c}]^T$ being the control input u and the commanded state x_{3c}. The outputs $\underline{y}_3 = [x_{3err} \;\; x_3 \;\; u]^T$ are the state error x_{3err} (which is fed into the controller), the pertinent state x_3, and the control u. These are necessary outputs for applying the solution procedure and analyzing the controller design using the SCTB in Chapter 5.

3.3.1 Design of Innermost Controller and Closing the Loop

Once the system $G_3(s)$ is defined, the innermost controller K_3 is ready for design. Follow the design procedure in Chapter 4 to solve for K_3. The state space representation of K_3 is as follows:

$$\dot{\underline{x}}_{k3} = A_{k3}\underline{x}_{k3} + B_{k3}x_{3err} = -B_{k3}x_3 + B_{k3}x_{3c} + A_{k3}\underline{x}_{k3} \tag{3.6}$$

$$u = C_{k3}\underline{x}_{k3} + D_{k3}x_{3err} = -D_{k3}x_3 + D_{k3}x_{3c} + C_{k3}\underline{x}_{k3} \tag{3.7}$$

where $\underline{x}_{k3}$ are the controller states of K_3 and A_{k3} is a $n_{k3} \times n_{k3}$ matrix.

After K_3 is designed, close the loop to obtain the closed loop system H_3:

$$\dot{\underline{x}}_{III} = A_{III}\underline{x}_{III} + B_{III}x_{3c} \tag{3.8}$$

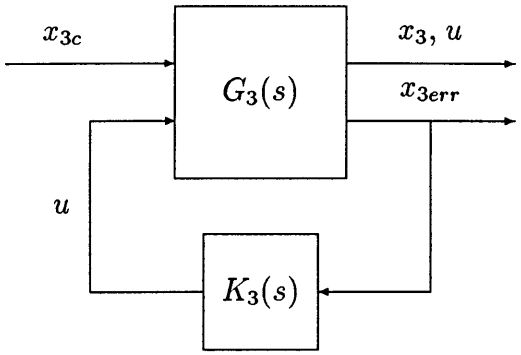

Figure 3.5: Innermost Loop: Model Based Representation

$$\underline{y}_{III} = C_{III}\underline{x}_{III} + D_{III}x_{3c} \tag{3.9}$$

where

$$A_{III} = \begin{bmatrix} A_3(:,1) - B_3(:,1)D_{k3} & A_3(:,2:3) & B_3(:,1)C_{k3} \\ -B_{k3} & \underline{0}_{1\times 2} & A_{k3} \end{bmatrix} \quad B_{III} = \begin{bmatrix} B_3(:,1)D_{k3} \\ B_{k3} \end{bmatrix}$$
$$C_{III} = \begin{bmatrix} 1 & 0 & 0 & \underline{0} \\ -D_{k3} & 0 & 0 & C_{k3} \end{bmatrix} \quad D_{III} = \begin{bmatrix} 0 \\ D_{k3} \end{bmatrix} \tag{3.10}$$

where the states are $\underline{x}_{III} = [\underline{x}_3^T \quad \underline{x}_{k3}^T]^T$ and the outputs are $\underline{y}_{III} = [x_3 \quad u]^T$. The notation $A_3(:,1)$ corresponds to the entire first column of matrix A_3 and $A_2(:,2:3)$ corresponds to a matrix consisting of the second *through* third column of matrix A_3. The input is the commanded state x_{3c}. Once the closed loop formulation of the innermost loop is completed, the next loop $G_{R-1}(s)$ (or $G_2(s)$ for the example) can be formulated.

3.4 The Intermediate Loops

For each intermediate loop, one pertinent state is added. This requires that state's dynamics to be returned in the formulation of the respective intermediate plant. For the intermediate loop, the system consists of the previous inner pertinent states and the internal states. The input to the system is the commanded intermediate state.

The generic intermediate loop is depicted in Figure 3.6 for pertinent state number

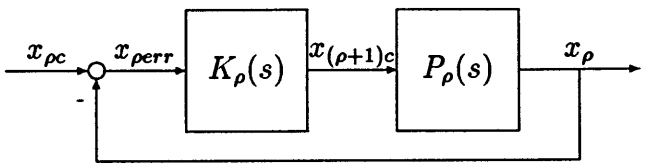

Figure 3.6: Generic Intermediate Loop

ρ. The current example has $\rho = 2$ and x_{2c} as the commanded intermediate state. The state space representation of the innermost system $G_2(s)$, as described in Figure 3.7, has the states $\underline{x}_2 = [x_2 \;\; x_3 \;\; x_i \;\; x_{ii} \;\; \underline{x}_{k3}^T]^T$. This state space system is as follows:

$$\dot{\underline{x}}_2 = A_2\underline{x}_2 + B_2\underline{u}_2 \tag{3.11}$$

$$\underline{y}_2 = C_2\underline{x}_2 + D_2\underline{u}_2 \tag{3.12}$$

where

$$A_2 = \begin{bmatrix} a_{22} & a_{23} - b_2 D_{k3} & a_{2i} & a_{2ii} & b_2 C_{k3} \\ a_{32} & & & & \\ a_{i2} & & A_{III} & & \\ a_{ii2} & & & & \\ \underline{0}_{n_{k3}\times 1} & & & & \end{bmatrix}_{n_2 \times n_2}$$

$$B_2 = \begin{bmatrix} b_2 D_{k3} & 0 \\ & 0 \\ B_{III} & 0 \\ & 0 \\ & \underline{0}_{n_{k3}\times 1} \end{bmatrix} \quad C_2 = \begin{bmatrix} -1 & 0 & 0 & 0 & \underline{0}_{1\times n_{k3}} \\ 1 & 0 & 0 & 0 & \underline{0}_{1\times n_{k3}} \\ 0 & -D_{k3} & 0 & 0 & C_{k3} \end{bmatrix} \quad D_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ D_{k3} & 0 \end{bmatrix} \tag{3.13}$$

with the inputs $\underline{u}_2 = [x_{3c} \;\; x_{2c}]^T$ being the previous commanded state x_{3c} and the current commanded state x_{2c}. The outputs $\underline{y}_2 = [x_{2err} \;\; x_2 \;\; u]^T$ are the state error x_{2err} (which is fed into the controller), the current pertinent state x_2, and the control u. As explained earlier, these outputs are chosen for application of the solution procedure and analysis of the controller design using the SCTB in Chapter 5.

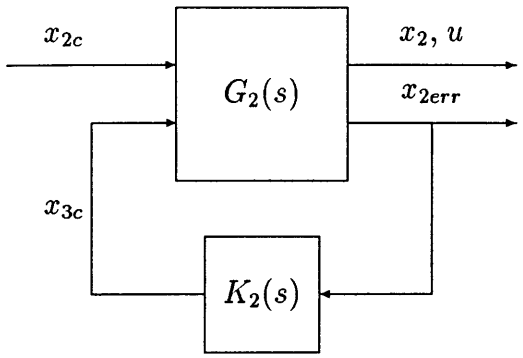

Figure 3.7: Intermediate Loop: Model Based Representation

3.4.1 Design of Intermediate Controller and Closing the Loop

Once the intermediate system $G_2(s)$ is defined, the intermediate controller $K_2(s)$ is ready for design. Follow the design procedure in Chapter 4 to solve for K_2. The state space representation of K_2 is as follows:

$$\dot{\underline{x}}_{k2} = A_{k2}\underline{x}_{k2} + B_{k2}x_{2err} = -B_{k2}x_2 + B_{k2}x_{2c} + A_{k2}\underline{x}_{k2} \tag{3.14}$$

$$x_{3c} = C_{k2}\underline{x}_{k2} + D_{k2}x_{2err} = -D_{k2}x_2 + D_{k2}x_{2c} + C_{k2}\underline{x}_{k2} \tag{3.15}$$

where $\underline{x}_{k2}$ are the controller states of K_2.

After $K_2(s)$ is designed, close the loop to obtain the closed loop system H_2:

$$\dot{\underline{x}}_{II} = A_{II}\underline{x}_{II} + B_{II}x_{2c} \tag{3.16}$$

$$\underline{y}_{II} = C_{II}\underline{x}_{II} + D_{II}x_{2c} \tag{3.17}$$

where

$$A_{II} = \begin{bmatrix} A_2(:,1) - B_2(:,1)D_{k2} & A_2(:,2:n_2) & B_2(:,1)C_{k2} \\ -B_{k2} & \underline{0}_{1\times(n_2-1)} & A_{k2} \end{bmatrix}$$

$$B_{II} = \begin{bmatrix} B_2(:,1)D_{k2} \\ B_{k2} \end{bmatrix} \tag{3.18}$$

$$C_{II} = \begin{bmatrix} 1 & 0 & 0 & 0 & \underline{0}_{1\times n_{k3}} & \underline{0}_{1\times n_{k2}} \\ -D_{k3}D_{k2} & -D_{k3} & 0 & 0 & C_{k3} & D_{k3}C_{k2} \end{bmatrix} \quad D_{II} = \begin{bmatrix} 0 \\ D_{k3}D_{k2} \end{bmatrix}$$

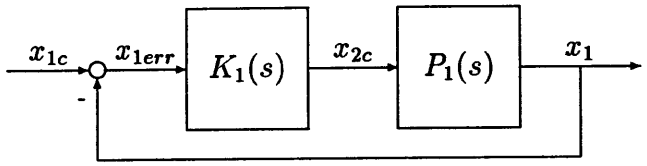

Figure 3.8: Generic Outer Loop

where the states are $\underline{x}_{II} = [\underline{x}_2^T \ \underline{x}_{k2}^T]^T$ and the outputs are $\underline{y}_{II} = [x_2 \ u]^T$. The input is the commanded state x_{2c}. Once the closed loop formulation of the last intermediate loop is completed, the outer loop $G_1(s)$ can be formulated.

3.5 The Outer Loop

For the outer loop, the last pertinent state x_1 is added. The remaining dynamics are returned in the formulation of the outer plant. The input to the system is the commanded outer state x_{1c}.

The generic outer loop is depicted in Figure 3.6. The notation is the same for the example. The state space representation of the innermost system $G_1(s)$ as described in Figure 3.9 has the states $\underline{x}_1 = [\underline{x}^T \ \underline{x}_{k3}^T \ \underline{x}_{k2}^T]^T$. This state space system is as follows:

$$\dot{\underline{x}}_1 = A_1\underline{x}_1 + B_1\underline{u}_1 \tag{3.19}$$

$$\underline{y}_1 = C_1\underline{x}_1 + D_1\underline{u}_1 \tag{3.20}$$

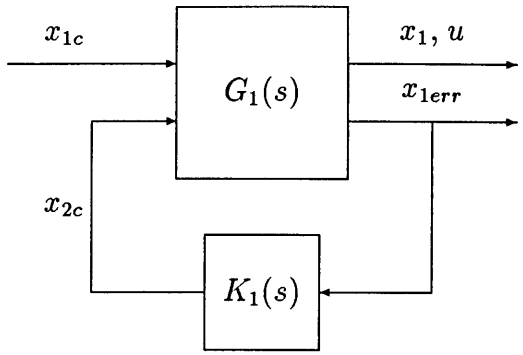

Figure 3.9: Outer Loop: Model Based Representation

where

$$A_1 = \begin{bmatrix} a_{11} & a_{12} - b_1 D_{k3} D_{k2} & a_{13} - b_1 D_{k3} & a_{1i} & a_{1ii} & b_1 C_{k3} & b_1 D_{k3} C_{k2} \\ a_{21} & & & & & & \\ a_{31} & & & & & & \\ a_{i1} & & & A_{II} & & & \\ a_{ii1} & & & & & & \\ \underline{0}_{(n_{k3}+n_{k2})\times 1} & & & & & & \end{bmatrix}$$

$$B_1 = \begin{bmatrix} b_1 D_{k3} D_{k2} & 0 \\ B_{II} & \underline{0}_{(n_1-1)\times 1} \end{bmatrix}$$

$$C_1 = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & \underline{0}_{1\times n_{k3}} & \underline{0}_{1\times n_{k2}} \\ 1 & 0 & 0 & 0 & 0 & \underline{0}_{1\times n_{k3}} & \underline{0}_{1\times n_{k2}} \\ 0 & -D_{k3} D_{k2} & -D_{k3} & 0 & 0 & C_{k3} & D_{k3} C_{k2} \end{bmatrix} \quad D_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ D_{k3} D_{k2} & 0 \end{bmatrix} \tag{3.21}$$

with the inputs $\underline{u}_1 = [x_{2c} \quad x_{1c}]^T$ being the previous commanded state x_{2c} and the outer commanded state x_{1c}. The outputs $\underline{y}_1 = [x_{1err} \quad x_1 \quad u]^T$ are the state error x_{1err} (which is fed into the controller), the outer pertinent state x_1, and the control u. Once again, these outputs will be applied to the solution procedure and used to analyze the controller design in the SCTB, detailed in Chapter 5.

3.5.1 Design of Outer Controller

Once the outer system $G_1(s)$ is defined, the final controller $K_1(s)$ is ready for design. Follow the design procedure in Chapter 4 to solve for K_2. This should be all the system manipulation necessary for the decoupled design. Other system manipulation may be necessary pending model reduction implementation. This will be described in Chapter 4.

Chapter 4

Controller Design Solution Procedure

A flowchart of the controller design solution procedure is depicted in Figure 4.1. The first step is to decouple the system into SISO loops, as described in Chapter 3. Once the innermost decoupled SISO loop is defined, as in Section 3.3, then that loop is ready for the controller design solution procedure to design the innermost controller. Likewise, for subsequent loops, once the loop is defined (in Section 3.4 for an intermediate loop or Section 3.5 for the outer loop), the controller for that loop is ready for the design solution procedure.

With the dynamics set up, the optimization problem is ready to be defined. The first step in defining the optimization problem is to choose an objective function. Choosing an objective function is described in detail in Section 4.1. Once the objective function is determined, the constraints are applied. The application of constraints is described in Section 4.2. With an objective function and constraints, the problem is ready for the Basis Function Algorithm, which solves the optimization problem numerous times (with a different basis each time) to find the controller (the design solution) with the lowest order. This algorithm is detailed in Section 4.3.

After the Basis Function Algorithm is completed for a set of constraints, the iterative portion of the solution procedure begins. If the Basis Function Algorithm determines that the problem is infeasible for the given constraints, then the con-

straints need to be relaxed before returning to the Basis Function Algorithm. If the Basis Function Algorithm has a feasible solution, then the solution needs to be analyzed for its performance characteristics. The analysis phase is coupled with model reduction. Model reduction is necessary if the solution is not of implementable order. Therefore, the original solution and all reduced order solutions are analyzed. If this analysis determines that the solution (and reduced order solutions) has unsatisfactory performance, constraints must be added to correct this performance issue. If the analysis determines that the solution (or reduced order solution) is acceptable but could be improved, then either constraints can be added or the Q-minimization objective function may be attempted in conjunction with modest constraint additions.

If the solution controller passes the scrutiny of the analysis step, then the implementability of this solution needs to be tested. This could be accomplished on the actual hardware the model used for the design is based on, but failures on the hardware can prove costly. Instead, the implementability of the solution should be determined on a simulator, based on the hardware. If the simulator test proves the solution to be a failure, it must be noted why and how the solution failed. Then that information is used to alter the constraints for another iteration of the solution procedure. However, if the solution passes the simulator test, it becomes the controller for that SISO loop. Now, the SISO loop is closed around this controller as described in Chapter 3. If this is not the outermost loop, the dynamics of the next SISO loop are added and prepared for the impending optimization problem. If the loop is closed around the solution controller and this is the outermost loop, the procedure is completed.

4.1 Choosing an Objective Function

Two options for the objective function are the $\mathcal{H}_2$ objective function and the Q minimization objective function. If starting from scratch, use the $\mathcal{H}_2$ objective function. If modest design improvements are imposed on an existing controller design, the Q-minimization objective function may be appropriate. However, aggressive design

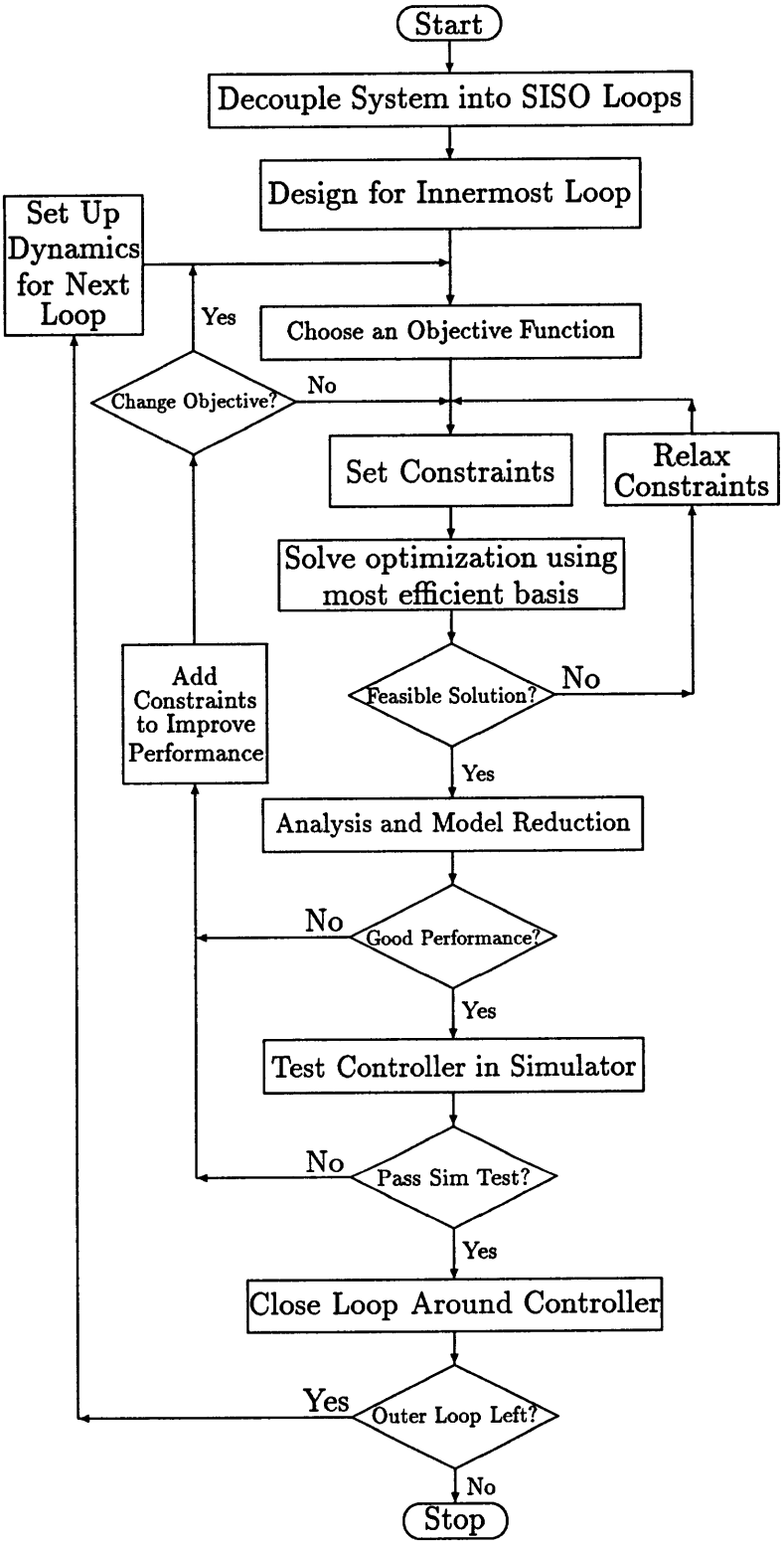

Figure 4.1: Flowchart of Main Solution Procedure

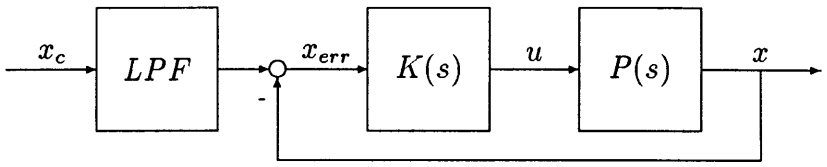

Figure 4.2: SISO Loop with Low Pass Filter

improvements on previous designs require the $\mathcal{H}_2$ objective function.

4.1.1 $\mathcal{H}_2$ Objective Function

Once the $\mathcal{H}_2$ objective function is chosen, a frequency weight should be applied to the SISO loop. This solution procedure will consistently apply the frequency weight as a first order lowpass filter (LPF) on the commanded input x_c as in Figure 4.2. This is to counter a design flaw in the SCTB. The goal of the optimization is to minimize the $\mathcal{H}_2$ norm of the transfer function from the commanded state to the error: $\frac{x_{err}}{x_c}$. The SCTB searches for a characteristic time scale within its solution procedure. In this case, there is unity high frequency gain. This will make the SCTB act as if there is infinite bandwidth unless there is a LPF on the input. It must be noted that the lowpass filter must have a cutoff frequency of less than half of the sample rate of the discrete time system. For example, if the system has a sample rate of 25 Hz, the lowpass filter cutoff frequency must be less than 12.5 Hz. Implementation of the $\mathcal{H}_2$ objective is described in Section 2.2.1.

4.1.2 Q-Minimization Objective Function

The Q-minimization objective may be used only if there is a very good nominal controller design. The plant, in this case, must be the closed loop system H_{nom} of the original plant P and the nominal controller K_{nom} as in Figure 2.4. In conjunction with this performance objective, constraints need to reflect slight improvements on the nominal design. Constraints that reflect more than modest improvements will likely make the optimization infeasible. Implementation of the Q-minimization objective is in Section 2.2.2.

4.2 Setting Constraints

There are four situations that lead the solution procedure to the constraint setting phase. The initial situation immediately follows the choice of objective function. Here, constraints are applied as the user sees fit, usually based on requirements. The second situation will follow the Basis Function Algorithm if it determines that the optimization problem is infeasible. In this case, it is assumed the problem is infeasible as a result of overly aggressive constraints. Naturally, constraints must be relaxed for the next iteration to try to get a feasible solution. The third situation follows solution controller analysis. If the analysis determines the solution is unsatisfactory or has room for improvement, constraints need to be altered accordingly for the next iteration. The final situation where the constraints need to be set follows the simulator test. If the solution is determined to be either unimplementable or lacking robustness to unmodeled dynamics and sensor noise, constraints must be altered for the next iteration. In the implementation phase, controllers that fail usually reveal why they failed, whether it is a stability failure or a command following failure.

The types of constraints available for this solution procedure are in the frequency domain and the time domain. Frequency domain constraints are upper bounds on the Bode magnitude for a given input/output pair. Time constraints are upper and lower bounds on the response curves of outputs in relation to a given input, such as a unit step response.

4.2.1 Frequency Domain Constraints

Frequency constraints can be set directly to the desired transfer function. The desired transfer function could be the closed loop transfer function or the sensitivity transfer function, which will be defined in terms of the generic SISO loop structure of Figure 4.3. For the purposes of this procedure, all constraints are upper bounds for the Bode magnitude of the given transfer function. This is consistent with the SCTB. Implementation of the frequency constraints to the optimization problem is in Section 2.4.1.

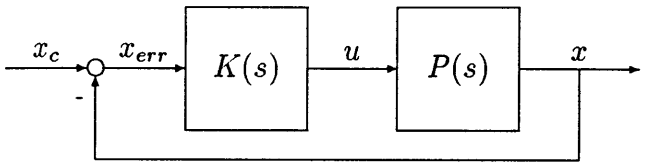

Figure 4.3: Generic SISO Loop

The closed loop transfer function $C(s)$ is defined as:

$$C(s) = \frac{x(s)}{x_c(s)} \tag{4.1}$$

Constraining the high frequency end of the closed loop transfer function should improve the stability robustness of the system. This can be accomplished with constraints of low Bode magnitude on the high frequency region. A cluster of these constraints can create a roll off constraint.

The sensitivity transfer function $S(s)$ is defined as:

$$S(s) = \frac{x_{err}(s)}{x_c(s)} \tag{4.2}$$

Constraining the low frequency end of the sensitivity transfer function should improve the command following characteristics of the system. This can be accomplished with low Bode magnitude constraints on the low frequency region.

4.2.2 Time Domain Constraints

The time constraints are applied to the responses of the desired output to a given input disturbance. Commonly, the given input disturbance (or commanded state input) will be in the form of an impulse, step, or ramp response. However, it is possible to have a user defined input.

The constraints are applied as upper or lower bounds to the specified outputs, specifically in relation to the chosen input. Typical time domain constrains are on elements such as maximum response, rise time, settling time, overshoot, and steady state value. Implementation of time constraints is found in Section 2.4.2.

In this thesis, the time domain responses to be constrained will often be the state error curves, with a desired steady state value of zero. The input response of choice

will be the step disturbance response. In this case, the commanded state input is set at one unit value.

4.3 Basis Function Algorithm with Optimization Solution

The Basis Function Algorithm is applied for each set of desired constraints. This is clearly the most tedious portion of the solution procedure. The main goal of the Basis Function Algorithm is to provide a feasible solution to the optimization that results in the lowest order controller. To achieve this goal it is necessary to achieve some subgoals.

The first of these subgoals is to reduce the number of basis functions used to approximate the optimal Q. This requires a good basis choice. A poor basis choice can make the problem highly complex computationally, requiring an inefficient amount of time for a design. In addition, the problem may result in a design of such a high order that the controller cannot be implemented, rendering the optimization useless. On the other hand, an efficient basis will result in lower order controllers that were designed in less computational time.

Another subgoal is based on the notion of reducing the number of basis functions, to find the most efficient time scale for those forms of basis that depend on time scales. As described in Section 2.3, there are many forms of basis functions. The Legendre and Laguerre functions require a given time scale to define the basis. The better the time scale, the fewer functions necessary to approximate the optimal Q. A time scale is better than another if it requires fewer functions to achieve a feasible solution or it uses the same number of functions with a better objective value.

The last subgoal is to reduce the amount of computational time necessary for achieving the optimal solution. Finding the most efficient basis requires solving the optimization over each iteration of the problem. An algorithm is necessary to make the procedure consistent and quicker than pure brute force.

4.3.1 Step 1: Best Homogeneous Basis

The first step of the basis function algorithm is to find the best basis using only one type of basis, a homogeneous basis. It is in this phase of the algorithm where the best time steps for the Legendre and Laguerre basis will be determined for future steps.

According to [5], the the Legendre basis provides the best initial guess. Using that logic, the Basis Function Algorithm will use the Legendre as the first homogeneous basis. Based on what is known about the dynamics of the loop to be controlled, the designer must make an educated guess at a time scale frequency a_{t0}. In addition, the user must have an idea how large of a solution will be tolerated. An upper bound on the number of basis functions N_{tol} is implemented for this reason. Once the time scale and upper bound are determined, the Algorithm is ready to enter the Basis Reduction Procedure (BRP).

The Basis Reduction Procedure

The BRP applies to all types of basis functions, not just the Legendre. The homogeneous BRP is detailed in a flowchart in Figure 4.4. After entering the BRP, values for the current number of basis functions N, the lower limit (the largest known infeasible number of basis functions) N_{inf}, and the smallest known number of basis functions that provide an optimal solution (an upper limit) N_{best} are all initialized. N_{inf} is initialized to zero, N_{best} is initialized to infinity, and N is user defined. Beyond the initial iteration, there may be a best previous solution, say to a previous time scale, that used N_{Best} functions. For every subsequent iteration entering the BRP, N_{best} will be initialized to be N_{Best} instead of infinity. Since N_{Best} is assumed to be within N_{tol}, N_{best} would then replace N_{tol} as the upper limit on the number of functions.

After initialization, the optimization problem is solved for N functions. If the feasible solution is found, then N_{best} will be set to be N, replacing N_{tol} as the upper limit on N. Then N is decreased, but within the lower limit N_{inf}. If the problem is infeasible the lower bound N_{inf} is raised to be N. Then N is increased, but within the upper limit N_{tol} (or N_{best}). The relationship between N_{tol}, N_{best}, and N_{inf} is shown

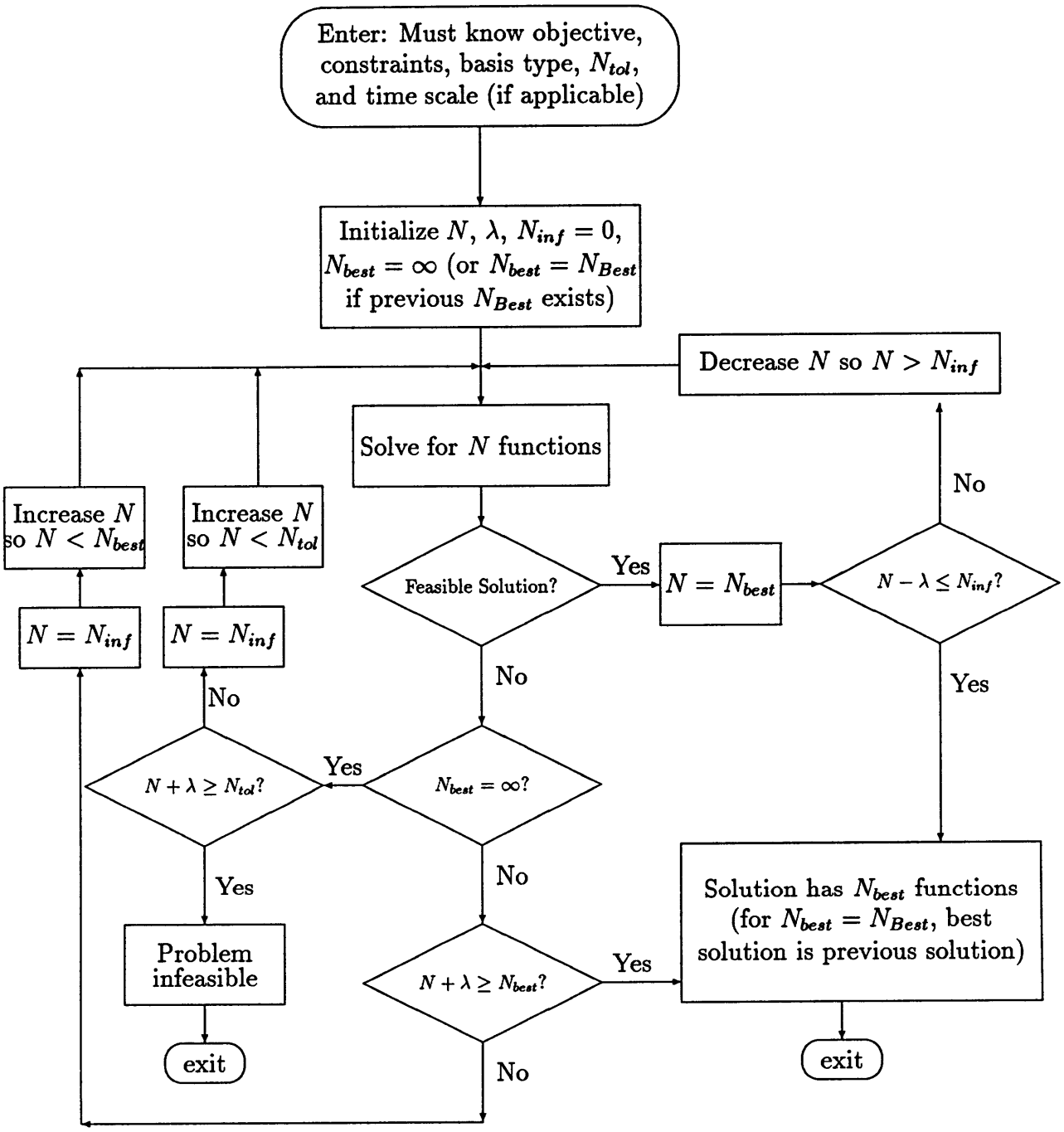

Figure 4.4: Flowchart of Homogeneous Basis Reduction Procedure

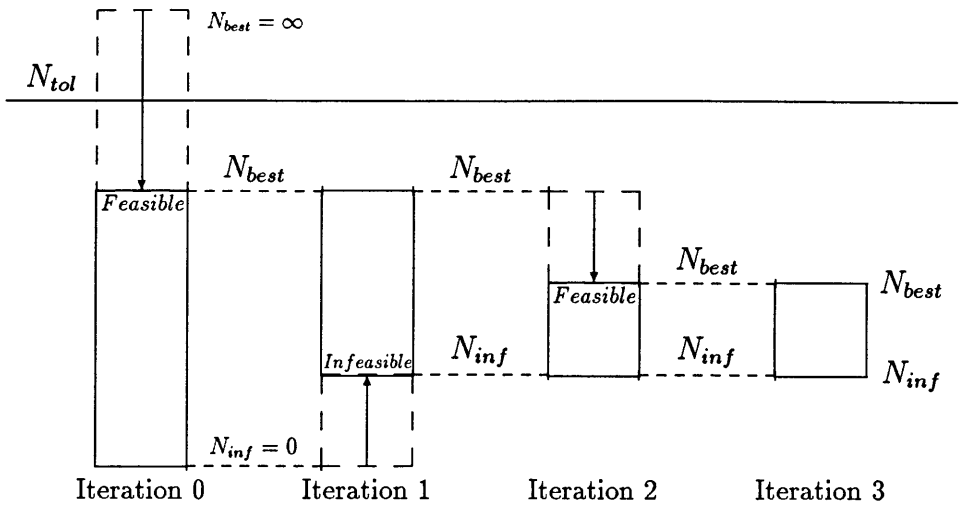

Figure 4.5: Basis Reduction Procedure Diagram

in Figure 4.5.

This recursion continues, closing in the upper and lower bounds until the solution is within a value λ of both the upper and lower bounds. The more precise the basis, the lower the value of λ. However, a higher λ can reduce the computational time. If no feasible solutions are found and $(N + \lambda)$ breaks the toleration limit N_{tol}, then the problem is deemed infeasible and the constraints need to be relaxed. If a feasible solution is found, $N_{best} = N_{Best}$ and the iteration for the best time scale can begin.

The Time Scale Iteration

The iteration for the frequency of the best Legendre time scale is similar to the BRP. After the initial solution is found, find the smallest pole in Q, p_Q. The upper and lower bounds on the time scale frequency a_t are then:

$$p_Q \leq a_t \leq 2p_Q \tag{4.3}$$

where the upper bound is just twice the lower bound. Each new value of a_t that results in a successful solution will then create a new upper or lower bound. A successful solution can be determined in one of two ways. One is where a feasible solution is found with less than N_{Best} functions (and then $N_{best} = N_{Best}$). The other is where a

successful solution is also found with N_{Best} functions, but the objective value for the optimization is lower than the previous N_{Best} solution. Like the BRP, this recursion continues, closing in the upper and lower bounds until the solution is within a value λ_a of both the upper and lower bounds. The time scale frequency will then be coronated as a_{Leg}, to be used for Legendre functions for the remainder of the Basis Function Algorithm. The more precise the basis, the lower the value of λ_a. However, a higher λ_a can reduce the computational time.

The time scale iteration for the Laguerre basis is slightly different than the Legendre basis. There are no set upper and lower bounds on the time scale frequency, as in the Legendre. The original time scale frequency a_{t0} is treated as the upper bound. Then the time scale frequency is systematically decreased until no successful solutions are found. The lowest time scale frequency to have a successful solution will be titled a_{Lag}. If decreasing the time scale frequency below a_{t0} did not provide a successful solution, then a_{t0} is treated as the lower bound. Then the time scale frequency is systematically increased until no successful solutions are found. The highest time scale frequency to have a successful solution will be titled a_{Lag}. If increasing the time scale frequency above a_{t0} did not provide a successful solution, then a_{t0} is coronated as a_{Lag}, to be used as the frequency of the Laguerre time scale for the remainder of the Basis Function Algorithm.

Ordering the Basis Types

After finding the best homogeneous Legendre basis solution, the best homogeneous Laguerre basis is determined. The Laguerre basis still uses the BRP, but it uses the alterations noted in the time scale iteration. The order of the solution for the Legendre and Laguerre functions are compared to each other. Then the FIR basis is solved for one more than the amount of functions used in the higher order solution between the Legendre and Laguerre basis. For example, if the Legendre basis used 11 functions for a 14th order solution and the Legendre basis used 16 functions for a 19th order solution, then the FIR basis is solved for 17 functions (one more than the number needed for the higher order Laguerre solution). If the FIR basis produces a

feasible solution, which it rarely does in this case, use the BRP to get the lowest order FIR solution. If the FIR basis produces an infeasible solution, then it is ordered last out of the three types of functions.

If there exists a good original controller K_{nom} from a previous iteration, poles from that controller may be added or form the basis using the fixed pole model (FPM) basis. The poles are ordered from smallest to greatest (or slowest to fastest). The FPM basis is a special case and is not involved in the basis ordering process unless it is the most efficient basis.

To define some terms, the best homogeneous basis form will be called Basis Function A creating a solution controller from N_A functions of n_Ath order. For Legendre and Laguerre basis, the solution controller order is the sum of the plant order n_P, the number of functions, and one:

$$n_A = N_A + 1 + n_P. \tag{4.4}$$

For the FIR basis, the solution controller order is the sum of the plant order and the number of functions:

$$n_A = N_A + n_P. \tag{4.5}$$

The next best homogeneous basis form will be called Basis Function B and the worst homogeneous basis (most likely the FIR) will be called Basis Function C.

4.3.2 Step 2: Best Pair Combination Basis

Once the homogeneous solutions are determined, the time scales are finalized, and the basis types are ordered by their efficiency, the Basis Function Algorithm is ready to find the best combination for a pair of basis types. The goal of the Pair Combination Basis Step is to find a solution lower than n_Ath order, the order of the best homogeneous solution. A successful solution will find a feasible solution that is either lower than n_Ath order or is n_Ath order with a lower objective function value.

Before starting this step, the relationship between the three different pair combinations must be understood. For *Leg* Legendre functions, *Lag* Laguerre functions, *F*

FIR functions, and a n_Pth order plant, the controller solution order n_s can always be determined by the following "2-combo" rules.

$$n_s = Lag + Leg + 1 + n_P \tag{4.6}$$

$$n_s = Lag + F + n_P \tag{4.7}$$

$$n_s = Leg + F + n_P \tag{4.8}$$

The A/B Combination

The step starts with the combination of basis types A and B. Following the 2-combo rules, one B-function is added to as many A-functions necessary A_0 to comprise the basis that keeps n_s equal to n_A. If this basis makes the optimization infeasible, then a basis of two B-functions and $(A_0 - 1)$ A-functions is tried. If the optimization is still infeasible, continue this pattern with a basis of three B-functions and $(A_0 - 2)$ A-functions. If all attempts up to this point are infeasible, experience has indicated that it is unlikely to find a feasible solution continuing on this pattern. Instead, go straight to the Alternate Pair Combinations portion of this section.

If the basis of one B-function and A_0 A-functions provides a successful solution, then try the basis of one B-function and $(A_0 - 1)$ A-functions. Keep reducing the amount of A-functions by one until the optimization is infeasible. The best solution from this iteration is n_{A0}th order. Following the 2-combo rules, the two B-function basis is attempted where A_1 A-functions are used such that n_s equals n_{A0}. The same procedure applies where the amount of A-functions is reduced by one until the optimization is infeasible. This general procedure is applied until the addition of two B-functions provides no successful solutions. At this point, try the Alternate Pair Combinations.

Alternate Pair Combinations

After trying the conventional combinations for pairs of basis types, these two alternate attempts are worth trying. First, try flipping the amount of the superior and inferior functions. In the A/B case, start with one A-function and A_0 B-functions (or however

many B-functions necessary to match the order of the current best solution). Then follow the procedure as in the A/B Combination portion of this section. If this provides no successful solution, only two iterations were wasted.

The other alternate attempt is to try an even number of A-functions and B-functions. If very small numbers of basis functions are being attempted, this might have already been tried. However, for large numbers of basis functions, this gives the design engineer the ability to check an A-heavy basis, a B-heavy basis, and now a A/B balanced basis. If this produces a successful solution, then follow the procedure as described in the A/B Combination portion of this section. However, if this does not produce a successful solution, end the procedure for this particular basis pair combination.

A/C and B/C Combinations

The A/C and B/C combinations are found in the same manner as the A/B combination, with the C-function assuming the inferior function role (B was the inferior function in the A/B Combination). The goal for these combinations is finding successful solutions that improve on the best current solution. At this point, the best current solution is either the n_Ath order homogeneous A-basis solution or the A/B solution that improved upon the n_Ath order solution. The Alternate Pair Combinations also apply to these combinations.

Adding Fixed Pole Model Functions

Using FPM functions from a good controller from a previous iteration is a special Pair Combinations case. Unless the FPM functions were the best homogeneous basis, these functions will always be treated as the inferior function. First, the A/FPM combination is tried in the same manner as the A/B combination. If no successful solution comes out of an A/FPM combination, then the Pair Combination procedure ends there. If a successful A/FPM solution is found, then the B/FPM combination is tried. The same conditions apply for the C/FPM combination to be attempted.

4.3.3 Step 3: Best Three-Combination Basis

After the Pair Combination procedure is complete, the current best solution will either be the A-function homogeneous case or the best Pair Combination solution. The best Three-Combination procedure tries to determine if a basis combination of Legendre, Laguerre, and FIR functions can improve on the best current solution. If the Pair Combination is the best current solution, the procedure is simple. Add one Third Function and remove one function from the most represented of the Pair Combination. If this produces a successful solution, a function of the most represented Pair Combination is removed one at a time. This continues until the problem is infeasible. Then another Third Function is added to the basis and the most numerous Pair Combination member loses a function. This procedure continues until adding 2 Third Functions does not produce a successful solution.

If the homogeneous case is the current best solution, then start with one B-function, one C-function, and as many A-functions necessary to get an n_Ath order solution. When there is a successful solution, remove one A-function at a time until the problem is infeasible. When there is no successful solution, remove one A-function and add one B-function. If this does not provide a successful solution, remove the new B-function and add one C-function instead. The procedure ends when two B-functions or C-functions are added without producing a successful solution.

Adding FPM

For a known given good controller, if no Three-Combination produces a solution, try the Three-Combination with the FPM functions replacing the C-functions. If there is a Three-Combination solution, FPM functions can be added as a fourth form of basis. This is done by removing the one function of the most represented function from the Three-Combination basis for every FPM function added to the basis. When solutions are found, the most represented function reduces one at a time until the problem is infeasible. When adding two FPM functions fails to produce a successful solution, the procedure is over.

4.4 Analysis and Model Reduction

After the best feasible solution is found, it must be determined whether this design truly meets all the constraints. There also may be characteristics of the design that were unaccounted for by the optimization. In addition, there might be an implementation requirement on the order of the controller. Very often, the optimization will produce, at best, a design that is still of very high order. This is where model reduction is applied. Model reduction is also useful for designs of moderate order, because it is always better to have a lower order design.

The analysis portion of the procedure determines if the solution's performance is acceptable. If the solution has satisfactory performance, then model reduction is applied to the solution. If the controller has all stable modes, then the Balance and Truncate method of Section 2.5.1 will reduce the system. If the controller has unstable states or modes, then the reduction method will be the Fractional Balanced Reduction (FBR) method of Section 2.5.2. The FBR method is an extension of the Balance and Truncate method for unstable systems. Once the controller is reduced to a desired order, this reduced order design is analyzed. If there is an implementation requirement having controllers less than n_ith order, then all controllers under n_ith order, that have acceptable performance, advance to the simulator test phase. If there are no designs that meet this requirement, constraints need to be altered for a new iteration of controller design.

Chapter 5

Structural Control Toolbox

The Charles Stark Draper Laboratory developed a MATLAB based Computer-Aided Design (CAD) tool called the Structural Control Toolbox (SCTB) [5]. The SCTB is driven in a Graphical User Interface (GUI) environment that greatly reduces the interaction between the user and the MATLAB command line. Although the SCTB has numerous capabilities, this chapter will only discuss those capabilities necessary for implementing the solution procedure presented in the previous chapter. Additional details and alternate explanations of the SCTB are found in [5, 7].

5.1 Main Panel

Once the SCTB is launched from the MATLAB command line, an empty Main Panel is displayed, as in Figure 5.1. The Main Panel has several menus. On the top menu bar, there are four pull-down menus: *File*, *Windows*, *Options*, and *Help*. Within the Main Panel there are pull-down menu bars: **Modify** (for plants and controllers), **Pre-Processor** (for plants and controllers), **Design Methods**, and **Analysis**. In addition to menu bars, there are function buttons on the Main Panel. The function buttons of interest include: **Import P**, **Close Loop**, and **Close**.

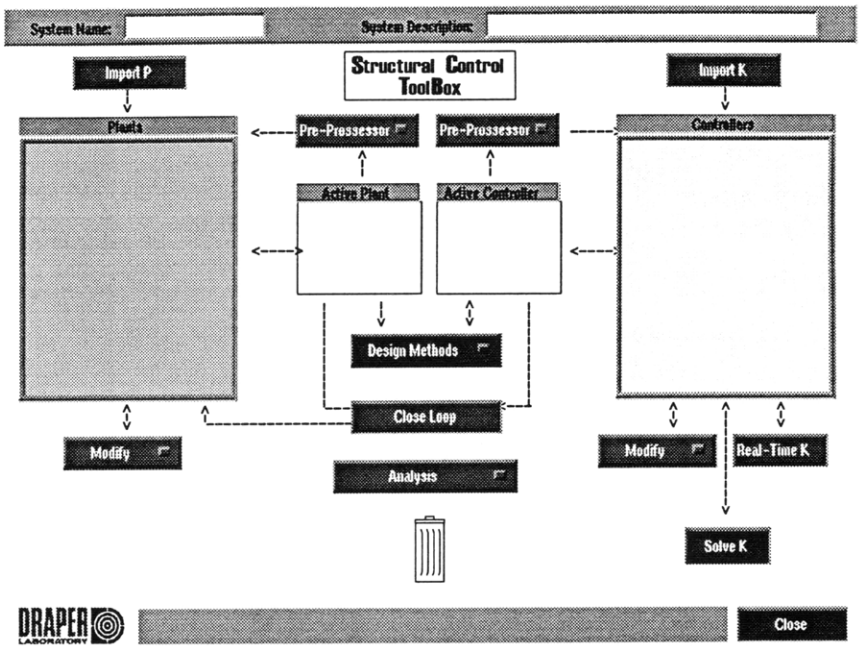

Figure 5.1: SCTB Main Panel

5.1.1 Top Pull-Down Menus

The *File* pull-down menu has six items: **New**, **Open DB**, **Close**, **Print**, **Save**, and **Save As**. The **New** option will clear the current database from the SCTB, giving the user the option of saving or deleting the current database. The **Open DB** option will load a previously saved database into the SCTB. If there is already an active database, the **Open DB** command will replace the current database, giving the user the option of saving or deleting that database. The **Close** option will quit the SCTB, giving the user the option of saving or deleting the current database. The **Print** option will print the panel to a printer or file. The **Save** and **Save As** options will save the current database to a MATLAB MAT file. Under the **Save** option, the user may save the database under the current filename or save the database to a new filename. The **Save As** option saves the database to a new filename.

The *Windows* pull-down menu lists the currently opened SCTB figure windows. Selecting a window will activate that particular figure window. The *Options* pull-down menu has two items: **Empty Trash Can** and **Refresh**. The **Empty Trash Can** option deletes all data in the trash can, located below the **Analysis** menu bar

in the Main Panel. The **Refresh** option updates the Main Panel based on data in the current database. The *Help* pull-down menu lists different help windows that will display help topics on the SCTB, MATLAB, and the Main Panel itself.

5.1.2 Panel Menu Bars

The **Modify** menu bars have three items: **Group, Edit**, and **Modify via Simulink.** The **Edit** option is the only option of interest to this thesis. The **Edit** option allows the user to modify properties of the selected plant or controller model in the current database using portions of the Data Input Module, described in Section 5.2. The **Pre-Processor** menu bars have two items: **Reduction** and **Unity Feedback.** The **Reduction** option is the only option of interest to this thesis. The **Reduction** option allows the user to reduce the order of the selected plant or controller model in the current database using the model reduction tool, described in Section 5.5.

The **Design Methods** menu bar has five items: **Classical Design, Constrained Optim, H2 Design, Hinf Design,** and **Mu Tools.** The only option of interest to this thesis is **Constrained Optim.** This option, detailed in Section 5.3 allows the user to design a controller using constrained convex optimization techniques. The other options are all various other forms of controller design. The **Analysis** menu bar has two items: **Time/Freq Analysis** and **Signal Transmission Analysis.** The **Time/Freq Analysis** item is the only option of interest to this thesis and is detailed in Section 5.4. This allows the user to analyze the current database plants and controllers in the time and frequency domains.

5.1.3 Main Panel Buttons

There are six Main Panel buttons: **Import P, Import K, Close Loop, Real-Time K, Solve K,** and **Close.** The buttons of interest are **Import P, Close Loop,** and **Close.** The **Import P** button allows the user to load a plant model from a MATLAB MAT file into the SCTB database using the Data Input Module. This loading process is presented in Section 5.2. The **Close Loop** button will generate the closed loop

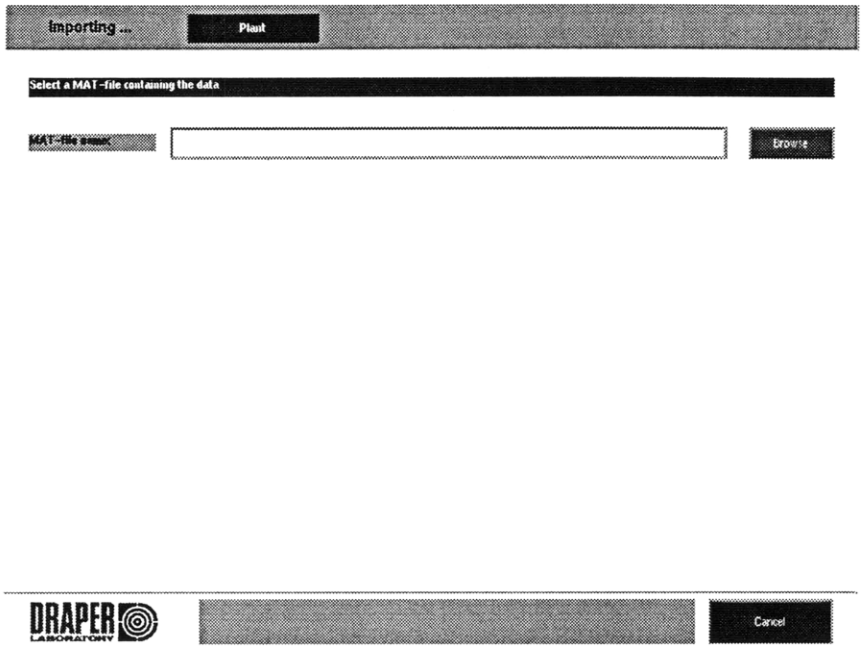

Figure 5.2: Data Input Module: MAT File Selection

plant model for the active plant and the active controller using the Closed Loop Tool, presented in Section 5.6. The **Close** button has the same effect as selecting the **Close** item of the *File* pull-down menu: quitting the SCTB and giving the user the option of saving or deleting the current database.

5.2 Loading a Plant Model

A state space representation of a plant model can be loaded from a MATLAB MAT file and placed into the SCTB. When the **Import P** button is selected in the Main Panel the first Data Input Module GUI is displayed as in Figure 5.2. Here, the GUI calls for the MAT file of the plant model. Upon receiving this file, the GUI adds the *Select the type of data* portion to the window as in Figure 5.3. This portion of the GUI has three buttons: **State-Space model**, **FRF data**, and **ZPK model**. For this thesis, only state space models are used and the **State-Space model** is the only button of interest and should be selected. However, if the model was in frequency response function (FRF) or zero/pole/gain (ZPK) form, the other two options should

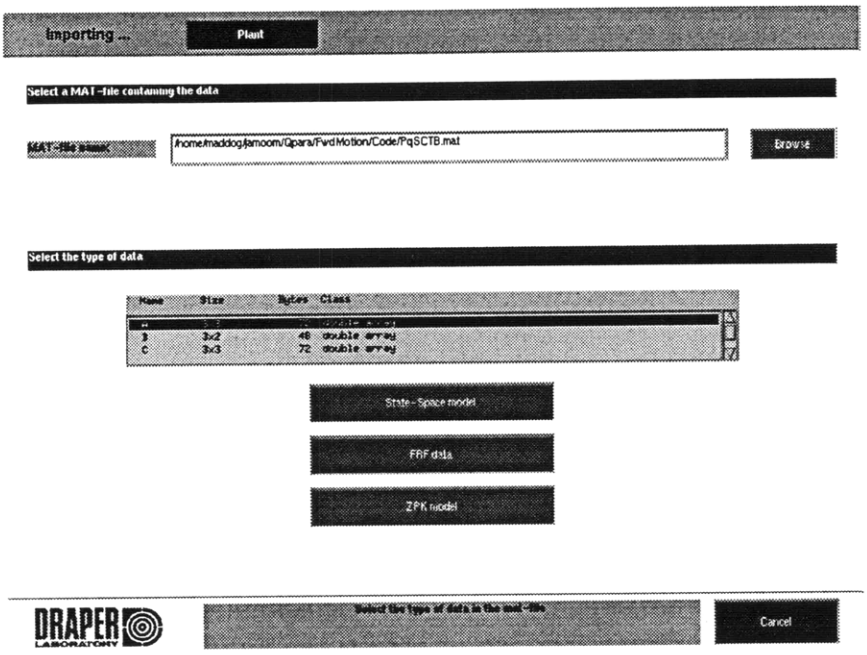

Figure 5.3: Data Input Module: Data Type Selection

be selected.

Once the **State-Space model** button is selected, the State-Space Model Input Window is opened as in Figure 5.4. The variables in the MAT file are assigned to the variables A, B, C, D, and T_s in Figure 5.4 to build the plant in the SCTB database. The A, B, C, and D variables correspond to state space matrices and T_s is the time step for discrete systems. For continuous models, a 0 is entered in the T_s field. If the state space matrices in the MAT file are not named A or a, B or b, C or c, and D or d, respectfully, the SCTB will not place the matrices in the appropriate field and the user will need to type the proper variable names into their corresponding matrix fields. Based on the matrices in the A, B, C, and D fields, the corresponding inputs and outputs are listed in the *Input/Output Descriptions* portion of the GUI. There the user may edit the name of any input or output. Once the variables are set and the inputs and outputs are named, the **Build P** button should be selected to continue assembling the plant model.

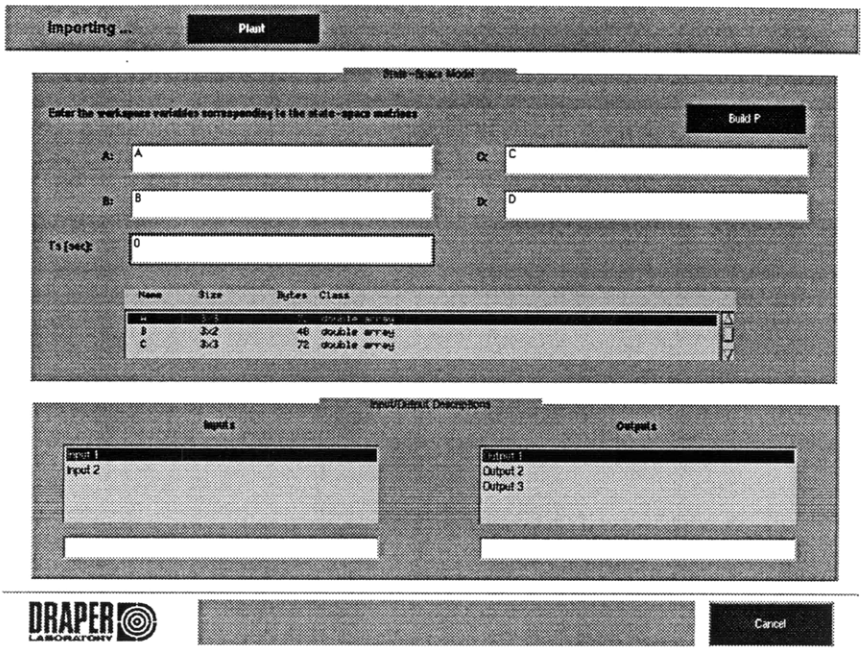

Figure 5.4: Data Input Module: State-Space Model Input Window

5.2.1 Defining Plant Characteristics

The **Build P** button will invoke the Plant Definition Window as in Figure 5.5. This window allows the user to select input and output channels, add plant uncertainty and input/output augmentation, choose between discrete and continuous time, and edit the plant name. The input channel selection portion of this panel assigns each input to be a control actuator (u), a disturbance (d), or both. At least one input must be designated a control actuator. The output channel selection portion of this panel assigns each output to be a sensor (y), a measurement (e), or both. At least one output must be designated a sensor. This thesis will not select any plant uncertainty, but will always select the **Add Noise Inputs** and **Add Control Outputs** buttons. Constrained optimization problems involve discrete time systems. For consistency, the Tustin conversion method is always selected when converting continuous systems to discrete systems. Upon completion, selecting the **Done** button will create the plant model, add it to the SCTB database and Main Panel, and close the Data Input Module GUI.

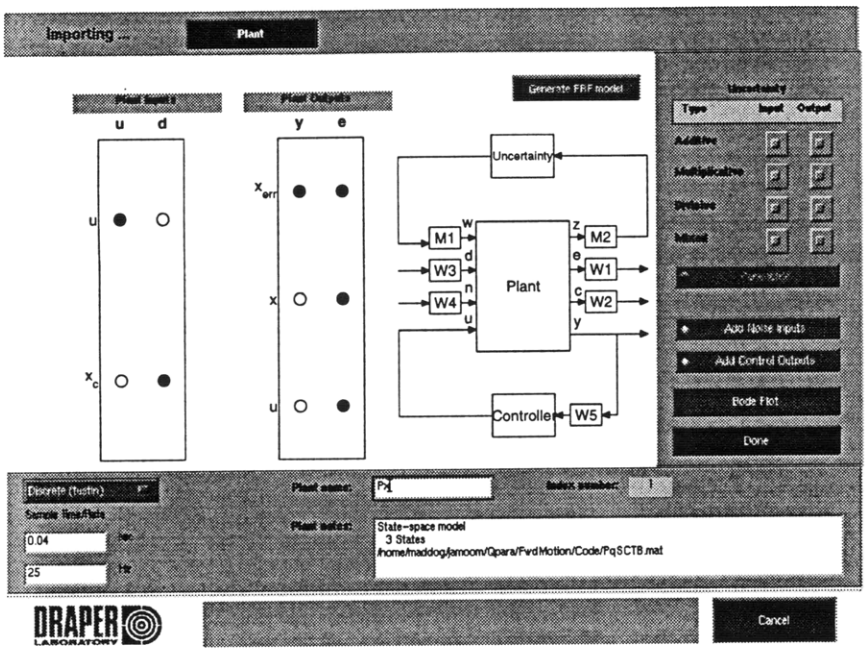

Figure 5.5: Data Input Module: Plant Definition Window

5.3 Constrained Optimization Controller Design

After the plant is defined and added to the SCTB database, it is time for controller design. Select the **Constrained Optim** item of the **Design Methods** menu bar in the main panel.

This invokes the Constrained Optimization Controller Design panel in Figure 5.6. This panel has four main modes: **Set Objective**, **Edit Basis**, **Edit Freq. Const.**, and **Edit Time Const.** There are also four buttons (excluding the **Close** button): **Feedback Sign:+**, **View Controller**, **Save**, and **Solve**. The feedback sign toggle switch should always remain on positive. The **View Controller** button gives you a graphical representation of a solved controller. The **Save** button saves the current solved controller. The **Solve** button solves the optimization for the given objective, basis, and constraints.

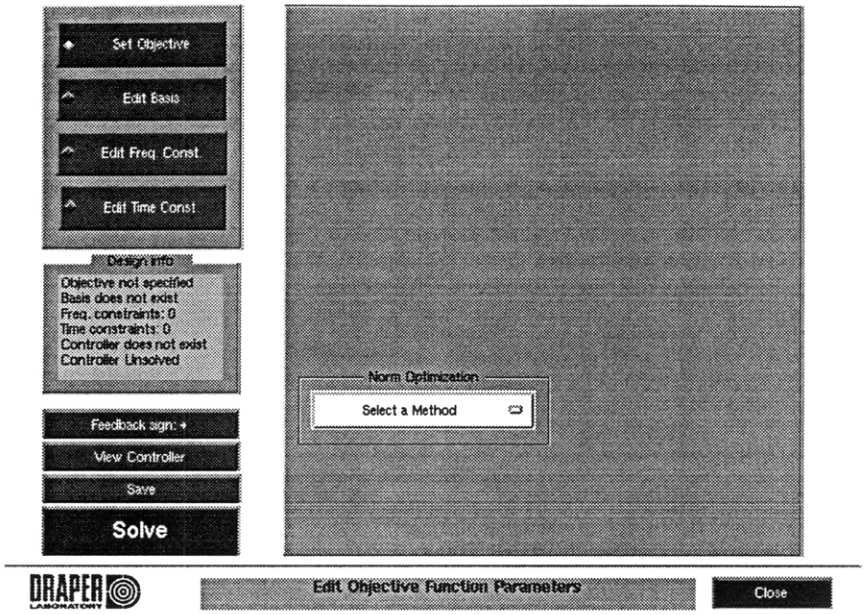

Figure 5.6: Constrained Optimization Controller Design

5.3.1 Setting the Objective Function

Upon its initial opening, the Constrained Optimization Controller Design is in **Set Objective** mode. There is a *Norm Optimization* menu bar requesting the user to *Select a Method.* The menu bar has three items: **H2 Minimization**, **Q Minimization**, and **L1 Minimization**. The $\mathcal{H}_2$ and Q-Minimization Objective Functions are the only objectives of interest to this thesis.

$\mathcal{H}_2$ Minimization

After selecting the **H2 Minimization** item in the *Norm Optimization* menu bar, the window looks like Figure 5.7. Here, the inputs and outputs are based on the generic SISO loop of Figure 4.3. There are two parts to preparing the $\mathcal{H}_2$ Objective Function: selecting the input/output pairs and adding the frequency weights. Selecting the input/output pairs are relatively easy. First, the user should select the input/output pair that needs to be minimized. In the generic SISO case, x_{err} is to be minimized, so the (x_c,x_{err}) pair should be selected. Also, the plant is assumed to be four block, meaning that the number of regulated outputs exceeds the number of measurements

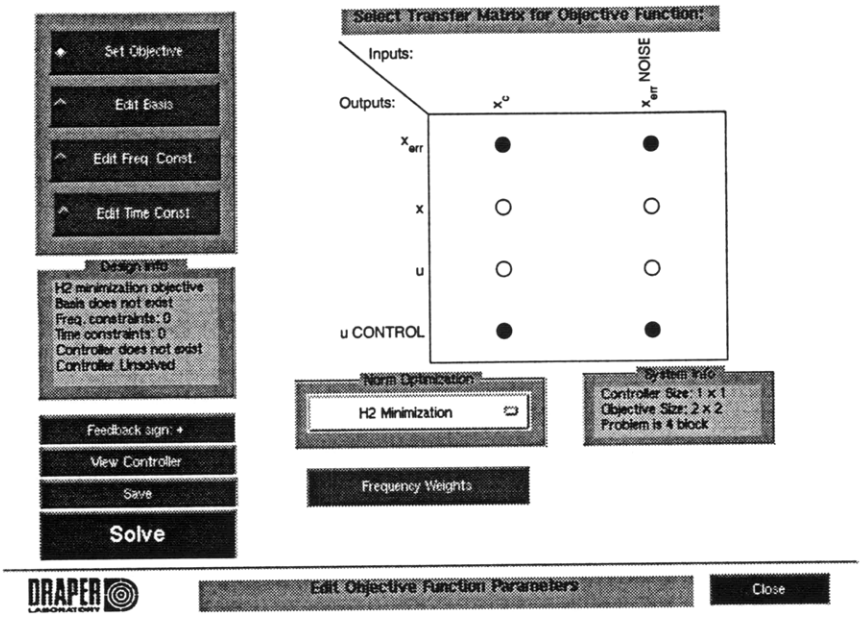

Figure 5.7: $\mathcal{H}_2$ Objective Function Control Panel

and the number of exogenous inputs exceeds the number of controls. Therefore, the pair of the regulated output $x_{err}NOISE$ and the exogenous input $uCONTROL$ is also selected. The intersections of the two selected pairs are automatically selected. This is reflected in Figure 5.7.

Frequency Weights

To add frequency weights, the **Frequency Weight** button must be selected. This will invoke the Defining Weighting Matrices GUI. To get a LPF on the commanded input, as in Figure 4.2, select the **W3 (Disturbance Input-*d*)** button. Then the **Edit** button will invoke the Function Editing window. Select **Lowpass** in the *Filter* menu and **Order (1 or 2)** in the *Filter Parameters* menu. To get a first order LPF, select **1st Order** in the menu bar. To add the desired break frequency, choose **Frequency** in the *Filter Parameters* menu. Select the **Update** button to enter the LPF into the *General Description* window. Now, select the **Save** button to enter the LPF as the W3 function and return to the Defining Weighting Matrices panel. The Bode plot of the LPF should be displayed. Selecting the **Save** button will add the frequency weighting to the objective function and return to the $\mathcal{H}_2$ Objective

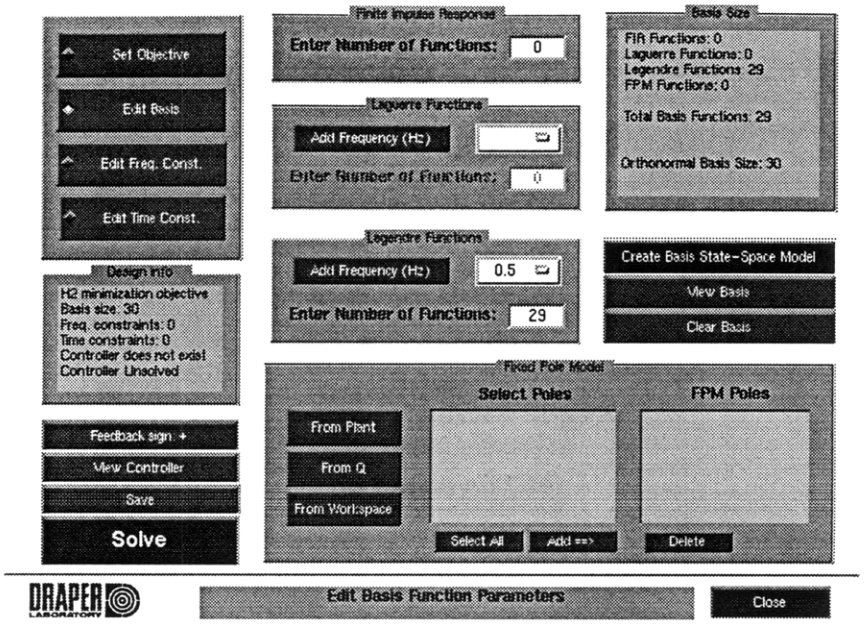

Figure 5.8: Basis Function Control Panel

Function Control Panel.

Q-Minimization

If Q-minimization is the desired objective function, the plant must be a product of the Close Loop Tool in the Main Panel. The Close Loop Tool, presented in Section 5.6, will produce a closed loop plant consisting of an open loop plant and a nominal controller. After the closed loop plant is selected and the Constrained Optimization Controller Design Window is opened, the Q-Minimization objective is chosen by choosing the **Q Minimization** item in the *Norm Optimization* menu bar of the **Set Objective** panel.

5.3.2 Editing the Basis

When the objective function is determined, the basis may be edited by selecting the **Edit Basis** button. This will invoke the Basis Function Control Panel in Figure 5.8. To create a basis, basis functions must be chosen. If Laguerre or Legendre functions are desired, a time scale frequency must be defined. To define a time scale, select the

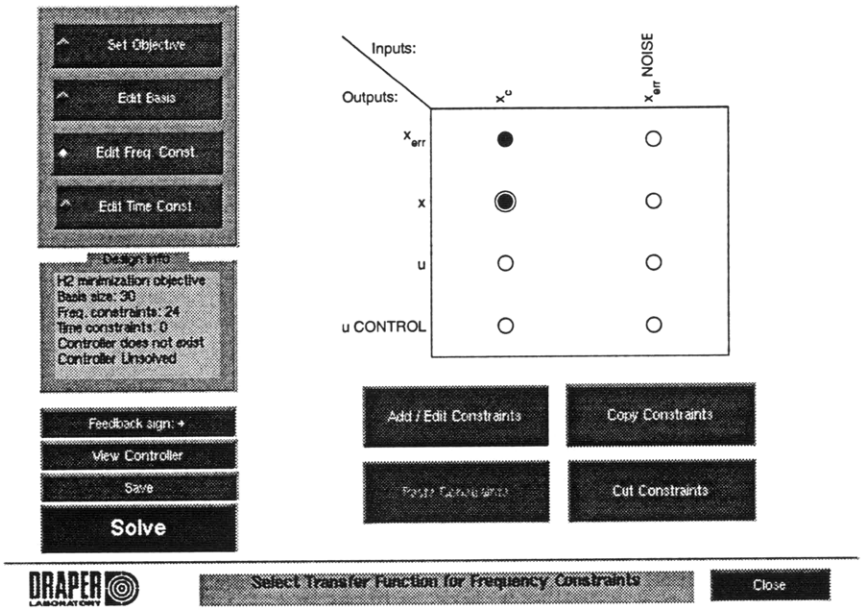

Figure 5.9: Frequency Constraint Transfer Function Selection Panel

respective **Add Frequency (Hz)** button. The appropriate frequency can now be entered. Then to add Laguerre, Legendre, or FIR functions, enter the desired number of functions in the respective portions of the panel. To enter the user defined FPM functions, they can be taken from the plant (select the **From Plant** button), from the Q of the last solution (select the **From Q** button), or from the workspace. There will be a list of pole choices in the *Select Poles* menu. Select the desired poles and add them to the basis by selecting the **Add ==>** button. Once all of the FIR, Laguerre, Legendre and FPM functions are determined, the basis is created by selecting the **Create Basis State-Space Model** button.

5.3.3 Editing Frequency Constraints

To set frequency constraints, select the **Edit Freq. Const.** button. This will invoke the Frequency Constraint Transfer Function Selection panel in Figure 5.9. To select the sensitivity transfer function $\frac{x_{err}(s)}{x_c(s)}$, select the ($x_c$,$x_{err}$) pair. To select the closed loop transfer function $\frac{x(s)}{x_c(s)}$, select the (x_c,x) pair. Once a transfer function is chosen, selecting the **Add/Edit Constraints** button will invoke the Frequency Constraint

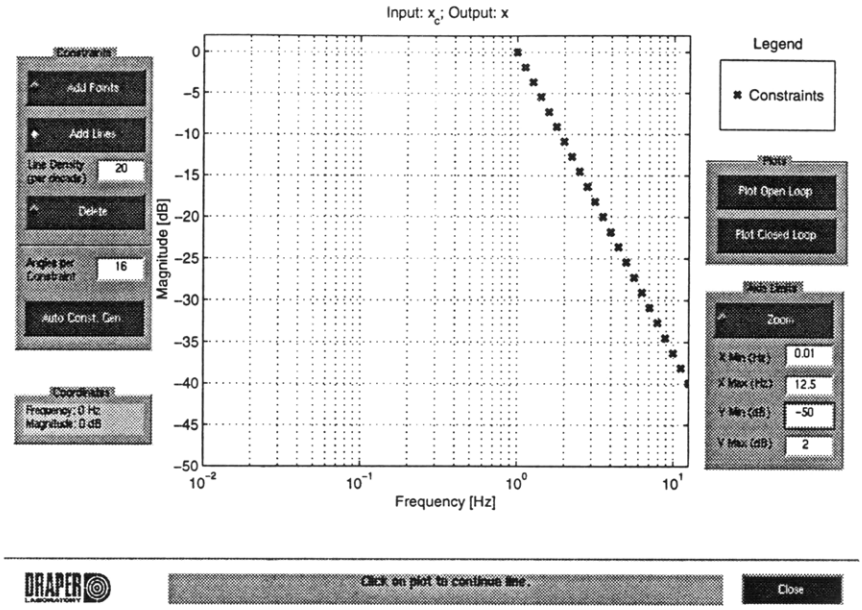

Figure 5.10: Frequency Constraint Tool

Tool in Figure 5.10. The Frequency Constraint Tool allows the user to place upper bound constraints on the Bode magnitude of the chosen transfer function. These constraints can be a point or a line of points. The point density of the constraint line is offered in terms of points per decade and is chosen in the constraint portion of the tool, just under the **Add Points** and **Add Lines** buttons. After the constraint type (point or line) is determined, the constraints are set by placing them on the magnitude grid itself. The Tool also allows the user to plot the open and closed loop versions of the selected transfer function to compare with the constraints. In Figure 5.10, a high frequency roll off constraint is placed on the closed loop transfer function $\frac{x(s)}{x_c(s)}$.

5.3.4 Editing Time Constraints

To set constraints on the time response, select the **Edit Time Const.** button. This will invoke the Time Constraint Transfer Function Selection panel in Figure 5.11. To select the error response to the commanded input disturbance, select the (x_c,x_{err}) pair. Once a transfer function is chosen, selecting the **Add/Edit Constraints** button will invoke the Time Constraint Tool in Figure 5.12. The Time Constraint Tool allows

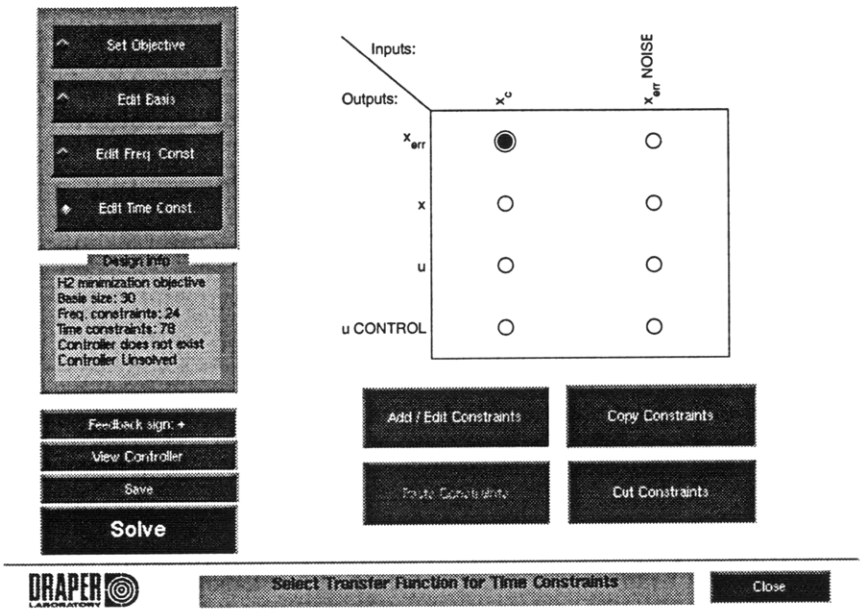

Figure 5.11: Time Constraint Transfer Function Selection Panel

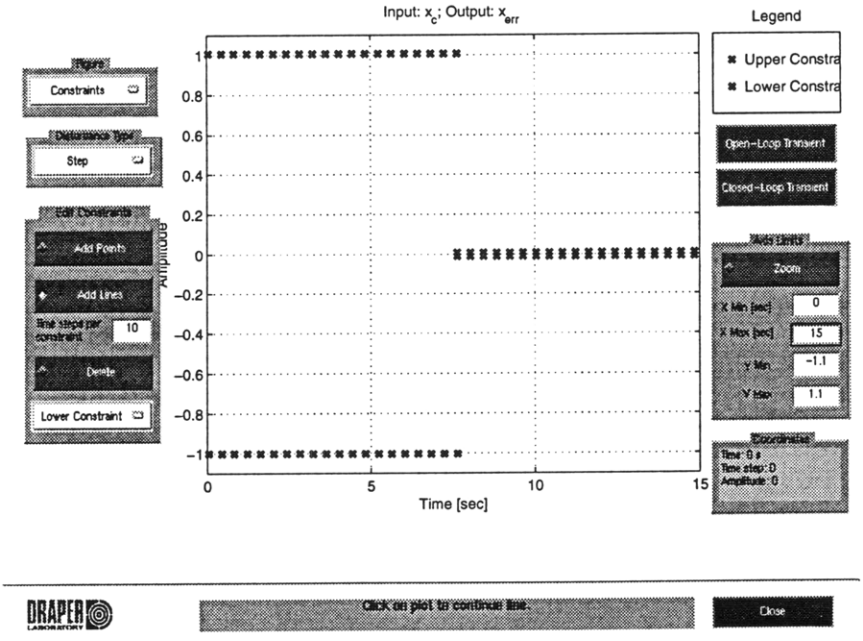

Figure 5.12: Time Constraint Tool

the user to place upper and lower bounds on the time response of an output to a given input disturbance. The Time Constraint Tool has three menu bars. The *Figure* menu bar toggles between the constraint plot and the disturbance plot. The *Disturbance Type* menu bar allows the user to choose the type of input disturbance. The choices are **Impulse**, **Step**, **Ramp**, and **Other**. The **Other** choice is a user defined input disturbance. The third menu bar toggles between Upper and Lower Constraints.

These constraints can be a point or a line of points. The point density of the constraint line is in terms of time steps per constraint (for discrete systems) and is chosen in the *Edit Constraints* portion of the tool. After a disturbance type is chosen and a constraint type (point or line) is determined, the constraints are set by placing them on the magnitude grid itself. The Tool also allows the user to plot the open and closed loop transients to compare with the constraints. In Figure 5.12, time constraints are placed on the error response $x_{err}(t)$ to a unit step commanded input $x_c(t)$. The constraints are a ± 1 amplitude constraint and a settling constraint of ± 0.01 from 8 seconds to 15 seconds.

5.3.5 Solving the Optimization Problem

After the objective, basis, and constraints have been set, the optimization problem is ready to be solved. Selecting the **Solve** button will start the execution of the optimization code. The SCTB uses MINOS 5.4 to solve the optimization problem [14]. Depending on the number and type of the constraints and the number and type of the basis, the optimization can be solved very quickly or very slowly. Through experience with solving SISO problems for systems of 8th order or less, it is determined that if the basis size is under 100, the constraints are feasible for a basis within 100 functions, and the SCTB is run on a Sparc 20 system or better, the user can expect typical MINOS runs to take less than ten minutes.

When MINOS starts solving the problem, it starts producing output to the MATLAB workspace. There are two steps to a the MINOS solution process. At first, MINOS tries to find a feasible solution. Here the columns of interest in the MINOS

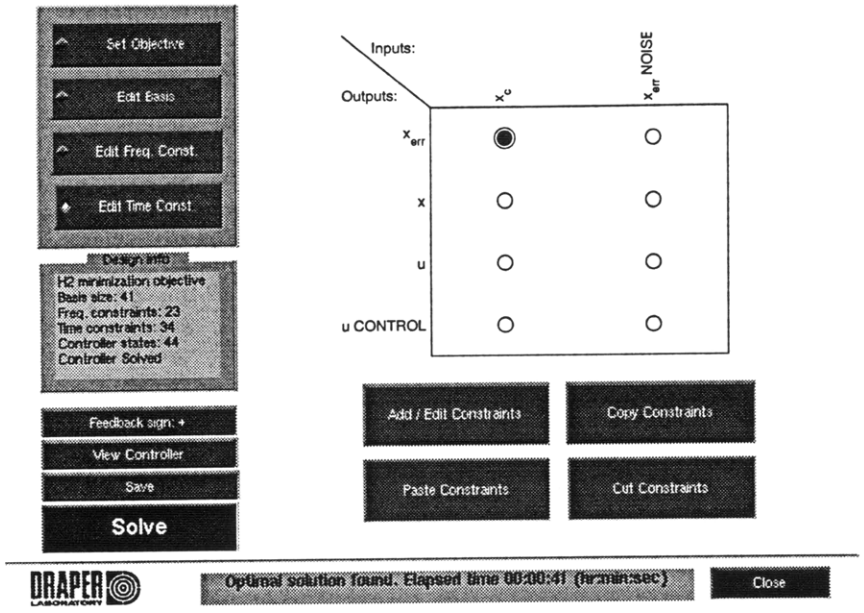

Figure 5.13: Constrained Optimization Control Design Panel: Post-Solution

output are the `Itn`, `ninf`, and `sinf` columns. The `Itn` column is the iteration number. For a particular iteration, the `ninf` column shows the number of infeasibilities and the `sinf` column shows the sum of those infeasibilities. As MINOS reaches a feasible solution, the `ninf` column and the `sinf` should decrease to zero, although the `sinf` column may have slight increases.

Once the feasible solution is reached, the second portion of the solution process aims to optimize the objective function. Here, the `objective` column becomes a column of interest. This is the objective value, which MINOS is working to minimize, for a particular iteration.

It is possible that there is no feasible solution. When this occurs, the number and sum of the infeasibilities may show how close the problem is to feasibility. If the sum of infeasibilities is very small, an addition of more basis functions or a slight relaxation of constraints may make the problem feasible.

If the problem is feasible, then a controller will be sent to the SCTB and the Constrained Optimization Controller Design panel will look like Figure 5.13. There, the order of the controller design and the length of time the SCTB used to solve the optimization problem are displayed. The final objective function value is found in the

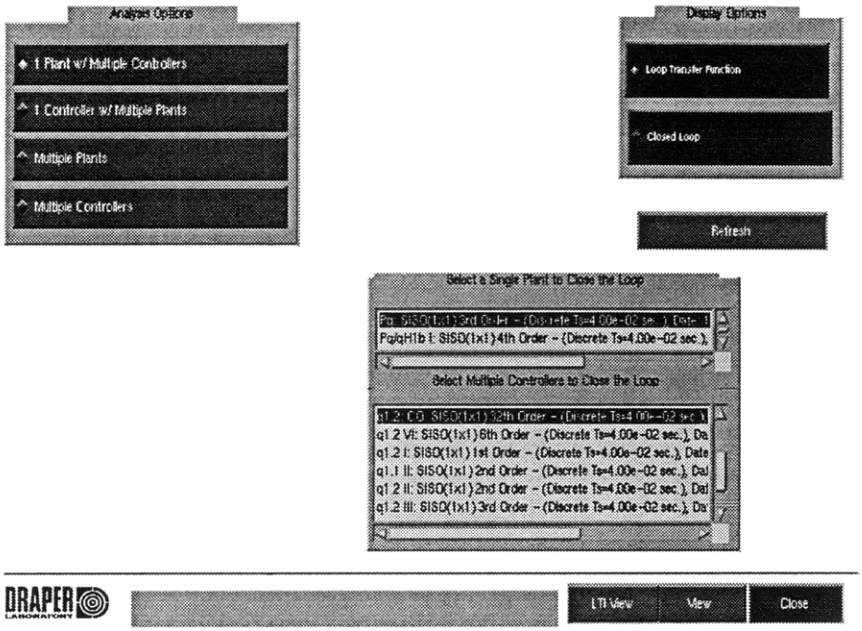

Figure 5.14: Analysis Tool: Initial Display

MINOS workspace output.

After the feasible solution is found, the controller needs to be analyzed to determine if it is a satisfactory solution. A controller that meets all constraints may not be a desirable controller. There may be a spiking problem, where the solution leaks through holes in constraints. Spiking is corrected by increasing the density of the constraints. Settling problems might also occur in the time response, settling constraints are not extended to infinity. Depending on when and where the settling constraints end, the solution may not have actually settled. In this case, the solution may be unstable, or may stabilize at a time much later that desired. The remedy is to extend the settling constraints to a further point in time.

5.4 Analysis Tool

To invoke the Analysis Tool, select the **Time/Freq Analysis** item of the **Analysis** menu bar. The initial display of the Analysis Tool is shown in Figure 5.14. In the Analysis Tool, there are four buttons in the *Analysis Options* section: **1 Plant w/Multiple Controllers**, **1 Controller w/Multiple Plants**, **Multiple Plants**,

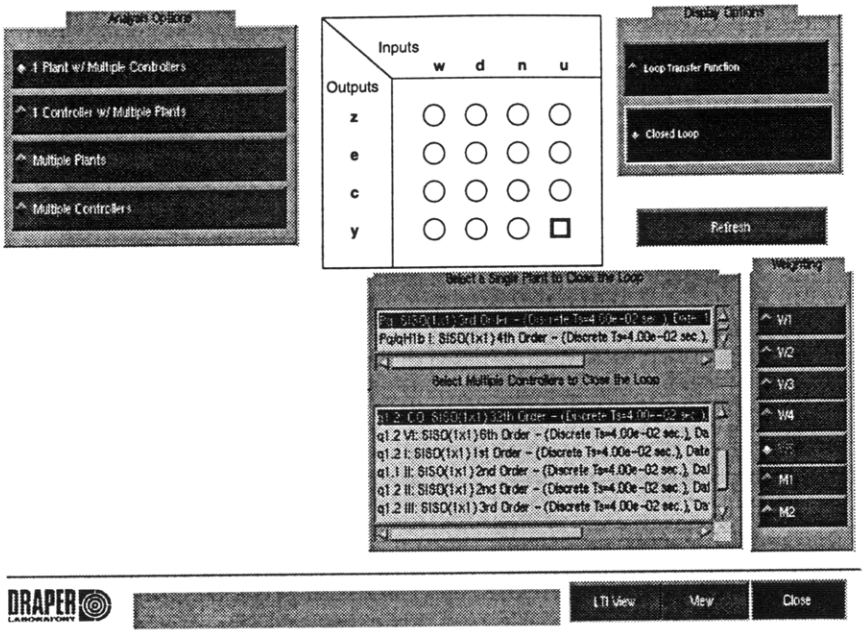

Figure 5.15: Analysis Tool: Closed Loop Display Option

and **Multiple Controllers**. The **1 Plant w/Multiple Controllers** button is depressed in the initial display with the corresponding menus for selecting a single plant and multiple controllers. There are two buttons in the *Display Options* section: **Loop Transfer Function** and **Closed Loop**. The **Loop Transfer Function** button is selected in the initial display. There are also four other buttons in the Analysis Tool GUI: **Refresh**, **LTI View**, **View**, and **Close**.

5.4.1 Analysis Options

The first of the *Analysis Options* is the **1 Plant w/Multiple Controllers** option. If the **Loop Transfer Function** button is selected, the Analysis Tool will prepare to plot open loop transfer functions of the plant/controller combinations of the selected plant and controllers. The selected controllers must have the same number of inputs and outputs. If the **Closed Loop** button is selected, the input/output selection grid and list of weighting functions are displayed as in Figure 5.15. The input/output pair usually selected is the d/e pair. For the SISO loops in this thesis, the (d,e) pair will contain all the outputs (defined as e outputs) in relation to the commanded input

(defined as a *d* input). The weighting options are not used in this thesis.

The second of the *Analysis Options* is the **1 Controller w/Multiple Plants** option. This option works just like the **1 Plant w/Multiple Controllers** option. The only difference is that there are multiple plants selected for one controller. The selected plants all need to have the same inputs and outputs. The third of the *Analysis Options* is the **Multiple Plants** option, for comparing plants to other plants. When selected by itself, the **Multiple Plants** option will automatically have the input/output selection grid and list of weighting functions displayed while the *Display Options* section disappears. The **Multiple Plants** option may be used in conjunction with the 1 **Plant w/Multiple Controllers** option or the 1 **Controller w/Multiple Plants** option. In either case, the *Display Options* section reappears. The final option is the **Multiple Controllers** option, which compares controllers to other controllers. In this case, there are no *Display Options*, no input/output grid, and no weighting function options.

5.4.2 Analysis Plots

Since all the plants and controllers will have the same sample rate (for discrete systems) and are not FRF objects, the **LTI View** option is the only option necessary (making **View** an unnecessary option). **LTI View** will combine the selected systems and call the MATLAB `ltiview` function. This will invoke the LTI Viewer. An important feature of the LTI Viewer is the *Plot Type* menu bar. There are many plot options, but the two plot options used will be the **Bode** and **Step**, which will plot the Bode Plots and Step Responses respectfully. The *Zoom* bar allows for horizontal, vertical, or combination zooming. There is also a **Full View** button that returns the zoomed plot to its original dimensions.

There are four pull-down menus: *File*, *Tools*, *Plots*, and *Help*. The *File* menu has three options: **New Viewer**, **Print Response**, and **Close Viewer**. **New Viewer** creates a new viewer window and **Close Viewer** closes the viewer. **Print Response** allows the user to print the plot on the screen to either a printer or to a file. Figure 5.16 is an example of printed LTI Viewer output. The *Tools* menu has three

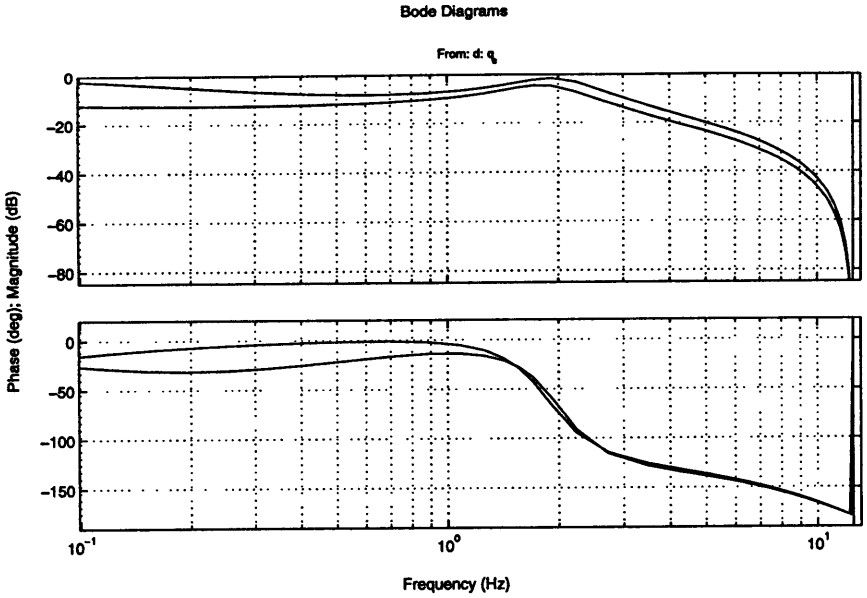

Figure 5.16: LTI Viewer Output

options: **Viewer Controls**, **Response Preferences**, and **Linestyle Preferences**. **Viewer Controls** toggles between having viewer controls available or displaying the plot by itself. **Response Preferences** controls the axes and units of the plots. **Linestyle Preferences** controls the lines of the plots. Different linestyles include different colored lines, dashed and dotted lines, and lines composed of repeated symbols. The *Plots* menu has a solitary **Grid On** option that toggles between displaying gridlines on the plots. *Help* is standard and self explanatory.

5.5 Model Reduction

To reduce a model, select the controller (or plant) to be reduced, and select the **Reduction** item of the respective **Pre-Processor** Main Panel menu bar. This will invoke the Model Reduction Tool as in Figure 5.17. To reduce the model, a method must first be selected out of the *Method* menu bar. There are many different methods available, but this thesis concentrated on the **Balance and Truncate** and the **Fractional Balanced Reduction** methods, both of which are offered in the *Method* menu bar.

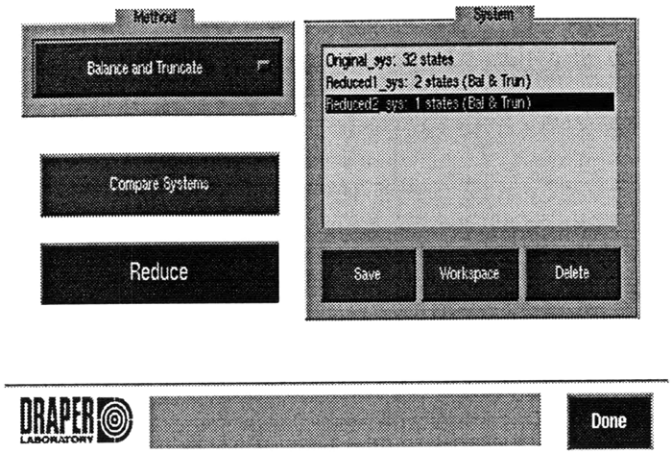

Figure 5.17: Model Reduction Tool

Once the method is determined, selecting the **Reduce** button will invoke a frequency weighting option. When this option is accepted, the Frequency Weighting Tool is invoked. There are options for a lowpass, highpass, or notch filter. This filter can be placed at the input, output, or both. After the filter is done (or if there was no weighting), the Reduction Tool is invoked as in Figure 5.18. This tool allows the user to select how many modes the reduced order model will have. The plot on the Reduction Tool provides a criterion for selecting modes. Note that in Figure 5.18, the minimum normalized Hankel singular value over the fifth most significant mode is similar to that for the 25th most significant mode. This means that the five mode reduced system is almost as good the 25 mode reduced system.

Once the model is reduced, the reduced order model is placed in the *System* menu of the Model Reduction Tool. This system can be saved as a controller in the Main Panel by hitting the **Save** button in the Model Reduction Tool.

5.6 The Close Loop Tool

The final SCTB application to be discussed is the Close Loop Tool. Before the Close Loop Tool can be used, the active plant and active controller must be selected. To invoke the Close Loop Tool, shown in Figure 5.19, select the **Close Loop** button in the Main Panel. Select the **Close Loop** button in the Close Loop Tool GUI to

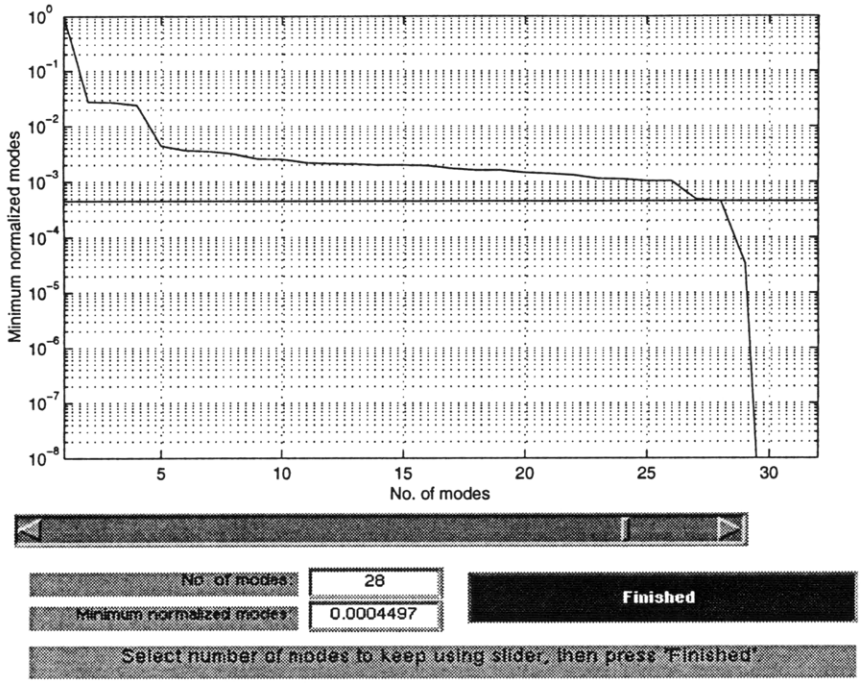

Figure 5.18: Reduction Tool

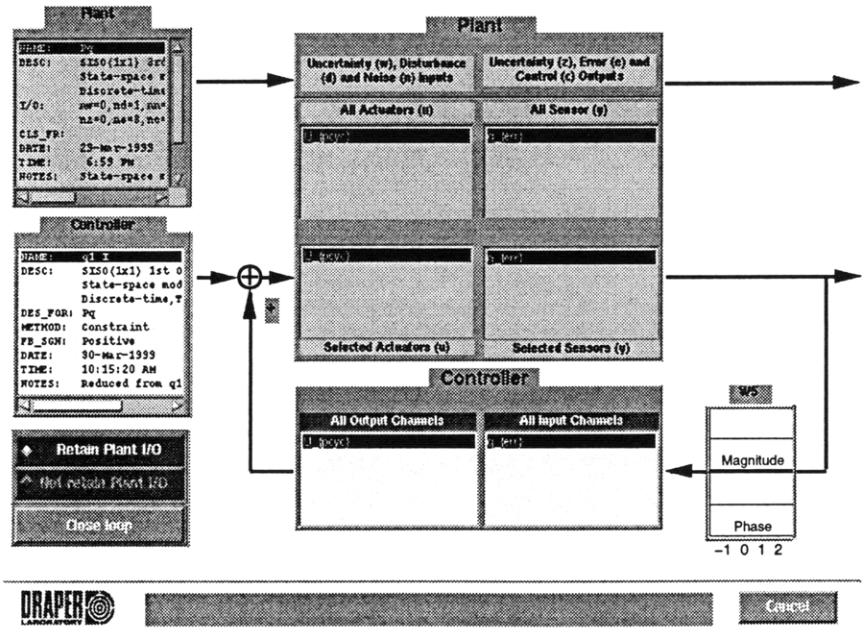

Figure 5.19: Closed Loop Tool

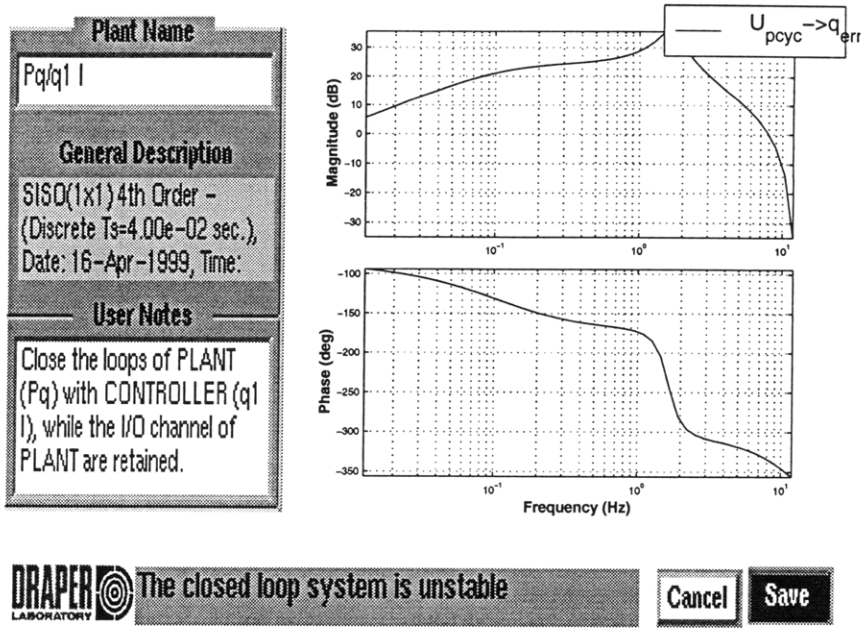

Figure 5.20: Closed Loop Tool: Save Panel

close the loop around the active controller. This will invoke the Save Panel. This panel, shown in Figure 5.20 can show the Bode plots, if desired. This window will also always state that the closed loop system is unstable. Closed loop stability should be determined before closing the loop. To save the closed loop plant, select the **Save** button in the Save Panel. This will save the closed loop system as a plant in the Main Panel and close all associated Close Loop Tool windows.

Chapter 6

Draper Small Autonomous Aerial Vehicle Control Example

The Draper Small Autonomous Aerial Vehicle (DSAAV) is a small model helicopter equipped with an avionics system designed by a team from MIT, Boston University, and the Draper Laboratory for the 1996 International Aerial Robotics Competition. Using classical control techniques, the DSAAV successfully performed completely autonomous missions and won the competition. Since the contest, the DSAAV has become a research platform for advanced flight control, among other things. An example of this is described in [16].

6.1 Design Objectives

The primary design objective is to improve the performance of the design through reduction of the error responses of the inner loops. The classical controllers previously designed for the DSAAV are the standards to which this design will be held. Another major objective is to determine the utility of decoupling the channels into the hierarchical structure, as opposed to leaving each channel coupled as a MIMO system. A further design objective is to determine the effect of the constraints on the solutions.

There are also objectives within the procedure. One is to determine the tradeoff

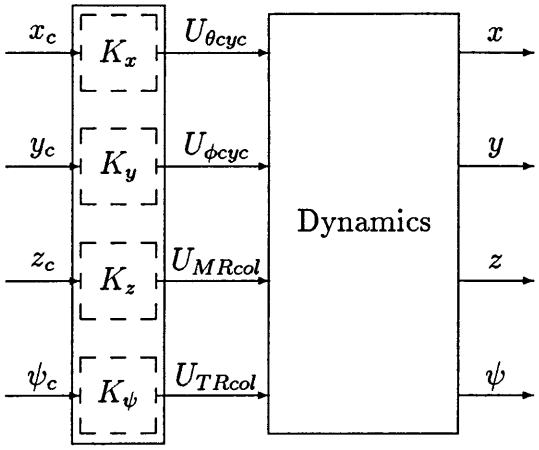

Figure 6.1: Entire DSAAV System of Coupled Channels

between using the Basis Function Algorithm and finding a high order solution through a large, inefficient basis. It is also of interest to determine how well the time scale iteration of the Basis Function Algorithm works. A final objective is to determine the utility of the Q-minimization objective function.

6.2 Decoupled Hierarchical System Architecture

The main DSAAV system is depicted in Figure 6.1. There are four main channels for each direction of motion: forward position x, lateral position y, vertical position z, and heading ψ. Each channel has an associated primary control actuator. Forward motion has the main rotor pitch cyclic $U_{\theta cyc}$, lateral motion has the main rotor roll cyclic $U_{\phi cyc}$, vertical motion has the main rotor collective U_{MRcol}, and heading has the tail rotor collective U_{TRcol}. Since each channel has an associated control actuator, it is assumed that each channel can have its own associated controller.

The dynamics of the four channels are coupled. The first decoupling separates the dynamics of the entire system into the four channels. More coupled dynamics exist within each channel. The coupled forward motion MIMO channel is depicted in Figure 6.2. The pertinent states for the forward motion channel are forward position x, forward velocity u, pitch θ, and pitch rate q. Each channel is decoupled into

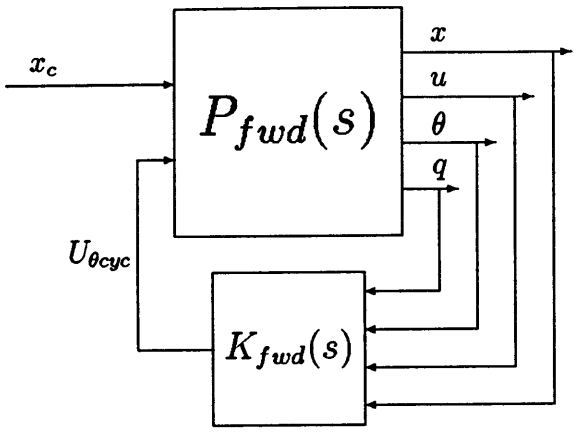

Figure 6.2: Coupled MIMO Structure for DSAAV Forward Motion

the hierarchical architecture of SISO loops, as in Figure 6.3 for the forward motion channel. Here each pertinent state is given its own SISO loop, complete with its own commanded state, dynamics, and controller. Each SISO loop has its own commanded state as the primary loop input. This commanded state is the output of the previous loop's controller. For example, the commanded pitch rate q_c is the output of the pitch controller K_θ.

6.2.1 Forward Motion Dynamics

The mathematical decoupling of the forward motion starts with the state space description of the coupled dynamics $P_{fwd}(s)$. There are two internal states for the forward motion loop: the main rotor plane pitch angle a_1 and the flybar plane pitch angle a_{1fb}. The state space representation of the coupled system $P_{fwd}(s)$ is:

$$\dot{\underline{x}} = A\underline{x} + BU_{\theta cyc} \tag{6.1}$$

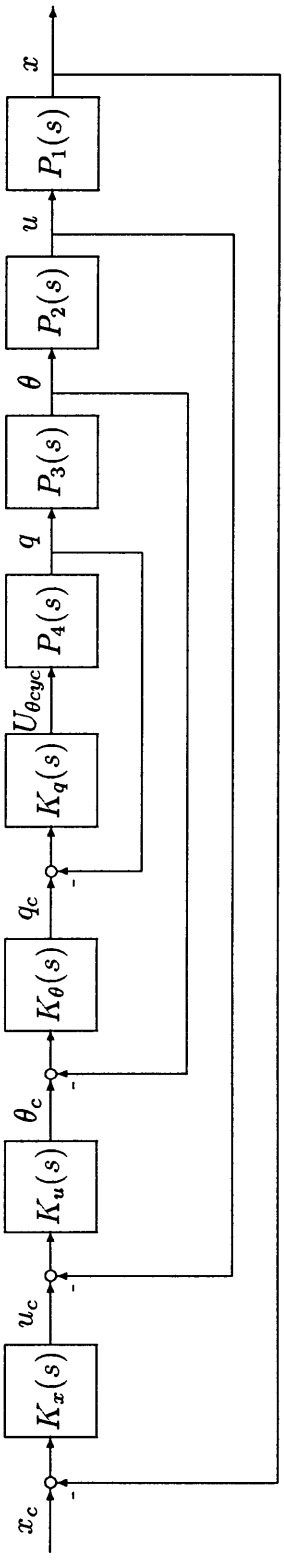

Figure 6.3: Hierarchical SISO Structure for DSAAV Forward Motion

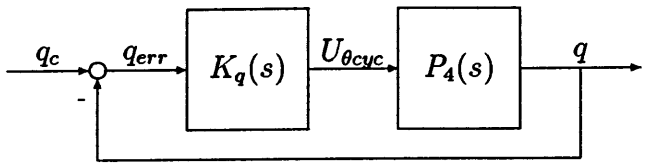

Figure 6.4: Pitch Rate Loop

where the dynamics matrix A and control matrix B are:

$$A = \begin{bmatrix} a_{xx} & a_{xu} & a_{x\theta} & \cdots & a_{x,a_{1fb}} \\ a_{ux} & a_{uu} & a_{u\theta} & & \\ a_{\theta x} & a_{\theta u} & a_{\theta\theta} & & \vdots \\ \vdots & & & \ddots & \\ a_{a_{1fb},x} & & \cdots & & a_{a_{1fb},a_{1fb}} \end{bmatrix}, \quad B = \begin{bmatrix} b_x \\ b_u \\ b_\theta \\ b_q \\ b_{a_1} \\ b_{a_{1fb}} \end{bmatrix} \tag{6.2}$$

with the states being $\underline{x} = [x \;\; u \;\; \theta \;\; q \;\; a_1 \;\; a_{1fb}]^T$.

6.2.2 The Pitch Rate Loop

The innermost loop of the forward motion channel is the pitch rate loop. To construct the pitch rate loop, the pitch rate plant $P_4(s)$ is created. $P_4(s)$ is a truncated version of P_{fwd}, with the dynamics associated with x, u, and θ completely removed. The pitch rate loop is depicted in Figure 6.4. The input is the commanded pitch rate q_c. The resulting pitch rate plant $P_4(s)$ is the third order system:

$$P_4(s) = \frac{7154s + 91271}{s^3 + 53.903s^2 + 323s + 5285} = \frac{7154(s + 12.82)}{(s + 2.1834 \pm 10.096j)(s + 49.536)}. \tag{6.3}$$

The state space representation of the model based pitch rate plant $P_q(s)$, as described in Figure 6.5, has the truncated states $\underline{x}_q = [q \;\; a_1 \;\; a_{1fb}]^T$. This state space system is as follows:

$$\dot{\underline{x}}_q = A_q\underline{x}_q + B_q\underline{u}_q \tag{6.4}$$

$$\underline{y}_q = C_q\underline{x}_q + D_q\underline{u}_q \tag{6.5}$$

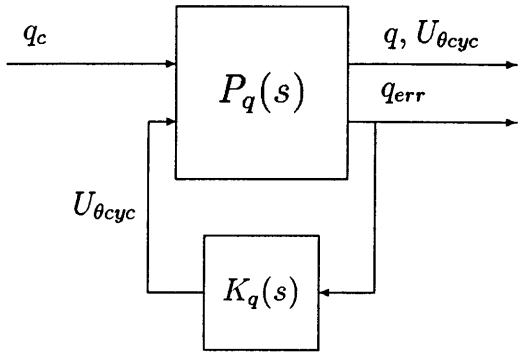

Figure 6.5: Pitch Rate Loop: Model Based Representation

with the state space matrices defined as:

$$A_q = \begin{bmatrix} a_{qq} & a_{q,a_1} & a_{q,a_{1fb}} \\ a_{a_1,q} & a_{a_1,a_1} & a_{a_1,a_{1fb}} \\ a_{a_{1fb},q} & a_{a_{1fb},a_1} & a_{a_{1fb},a_{1fb}} \end{bmatrix} \quad B_q = \begin{bmatrix} b_q & 0 \\ b_{a_1} & 0 \\ b_{a_{1fb}} & 0 \end{bmatrix}$$
$$C_q = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad D_q = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{6.6}$$

with the inputs $\underline{u}_q = [U_{\theta cyc} \;\; q_c]^T$ being the control actuator $U_{\theta cyc}$ and the commanded pitch rate q_c. The outputs $\underline{y}_q = [q_{err} \;\; q \;\; U_{\theta cyc}]^T$ are the pitch rate error q_{err} (which is fed into controller K_q), the pitch rate itself, and the control. These outputs were necessary for analysis in the SCTB.

Closing the Loop Following Pitch Rate Controller Design

Once $P_q(s)$ is defined, the pitch rate controller K_q is ready for design using the solution procedure. This design is described later in this chapter. The state space representation of K_q is as follows:

$$\dot{\underline{x}}_{kq} = A_{kq}\underline{x}_{kq} + B_{kq}q_{err} = -B_{kq}q + B_{kq}q_c + A_{kq}\underline{x}_{kq} \tag{6.7}$$

$$u = C_{kq}\underline{x}_{kq} + D_{kq}q_{err} = -D_{kq}q + D_{kq}q_c + C_{kq}\underline{x}_{kq} \tag{6.8}$$

where $\underline{x}_{kq}$ are the controller states of K_q and A_{kq} is a $n_{kq} \times n_{kq}$ matrix.

After K_q is designed, close the loop to obtain the closed loop system H_q:

$$\dot{\underline{x}}_Q = A_Q \underline{x}_Q + B_Q q_c \tag{6.9}$$

$$\underline{y}_Q = C_Q \underline{x}_Q + D_Q q_c \tag{6.10}$$

with the state space matrices defined as:

$$A_Q = \begin{bmatrix} A_q(:,1) - B_q(:,1)D_{kq} & A_q(:,2:3) & B_q(:,1)C_{kq} \\ -B_{kq} & \underline{0}_{1\times 2} & A_{kq} \end{bmatrix} \quad B_Q = \begin{bmatrix} B_q(:,1)D_{kq} \\ B_{kq} \end{bmatrix}$$
$$C_Q = \begin{bmatrix} 1 & 0 & 0 & \underline{0} \\ -D_{kq} & 0 & 0 & C_{kq} \end{bmatrix} \qquad D_Q = \begin{bmatrix} 0 \\ D_{kq} \end{bmatrix} \tag{6.11}$$

where the states are $\underline{x}_Q = [\underline{x}_q^T \ \ \underline{x}_{kq}^T]^T$ and the outputs are $\underline{y}_Q = [q \ U_{\theta cyc}]^T$. The input is the commanded pitch rate.

6.2.3 The Outer Three Loops

Construction of the pitch, forward velocity, and forward position loops is very similar to the pitch rate loop construction. The construction of these loops used the procedures described in Chapter 3 for intermediate and outer loops. Therefore, this section will not detail how the outer three loops were constructed.

6.3 Controller Design using Solution Procedure

The standard to which the controller designs will be measured against is the classical controller designs presently used on the DSAAV. The DSAAV is a discrete time system with a sample rate of 25 Hertz (or 0.04 seconds). The current actual classical designs for pitch rate K_{qA} (the 'A' in K_{qA} stands for actual) forward velocity K_{uA} are first order controllers. The current actual classical designs for pitch $K_{\theta A}$ and forward position K_{xA} are proportional controllers.

The performance standard to which the controller designs will be held in this section are the closed loop step responses (based on a unit step of the commanded

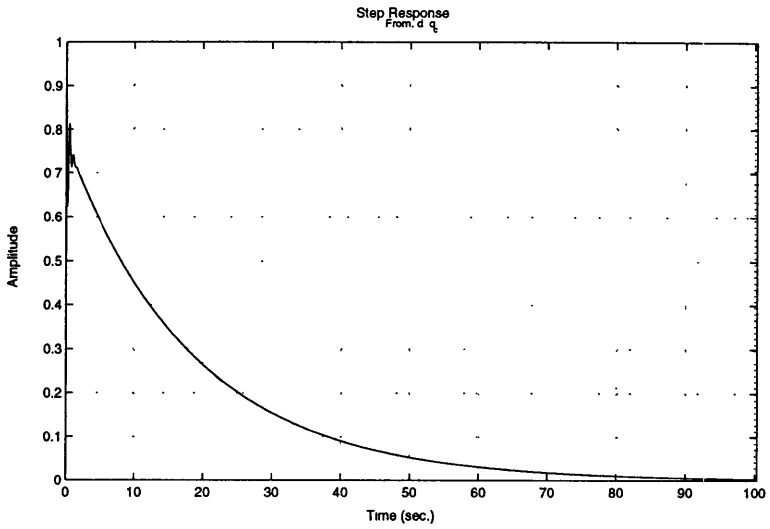

Figure 6.6: Pitch Rate Error Step Response: Based on Controller K_{qA}

state) of the state errors. For the pitch rate loop, this would be the pitch rate error transient $q_{err}(t)$. The step responses for controllers K_{qA}, $K_{\theta A}$, K_{uA}, and K_{xA} are in Figures 6.6 through 6.9 respectively.

The rest of this section will detail the application of the solution procedure. The pitch rate loop controller (K_q) design will be used as the detailed example for the solution procedure. Only the resulting solutions to the other three loops will be presented.

6.3.1 Pitch Rate Loop: Constrained Optimization Preparation

This subsection sets up the constrained optimization problem for the pitch rate loop controller. This preparation describes the final objective function and constraints used to find the best solution.

The initial objective function is always the $\mathcal{H}_2$ minimization objective. This objective function was also used to find the pitch rate loop controller. For this entire example, a first order LPF with a 10 Hertz cutoff frequency was always added to the input. Notice that 10 Hertz is less than half of the system sample rate.

Throughout the solution procedure, the constraints selected were based on the

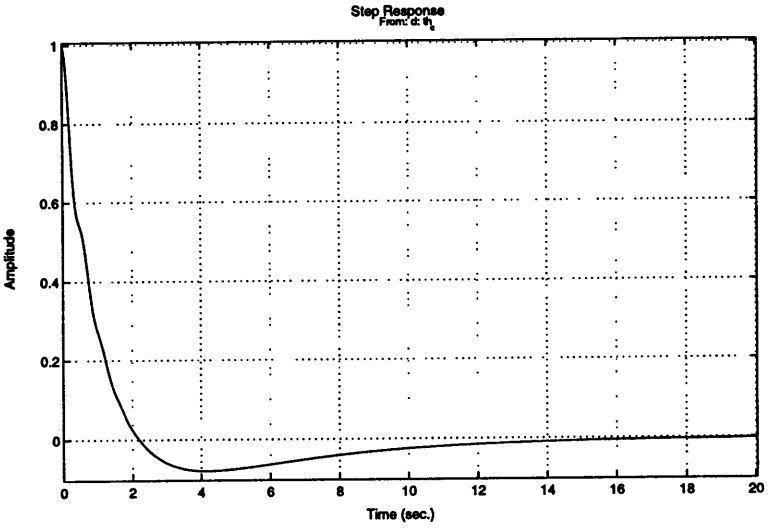

Figure 6.7: Pitch Error Step Response: Based on Controller $K_{\theta A}$

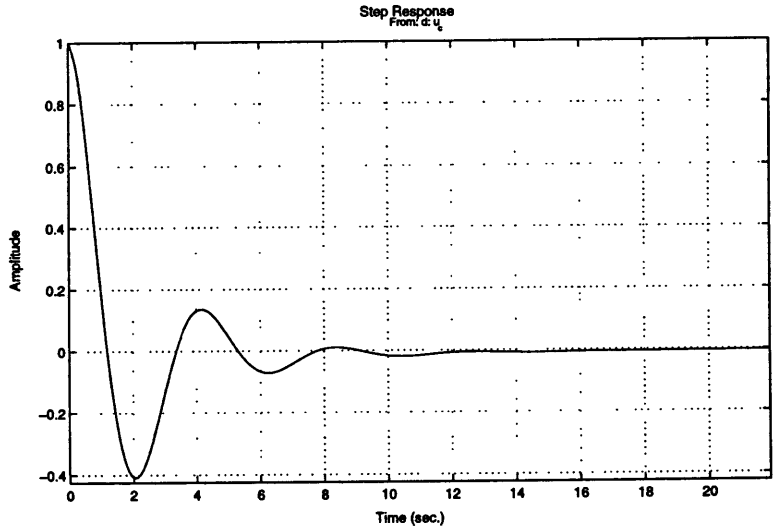

Figure 6.8: Forward Velocity Error Step Response: Based on Controller K_{uA}

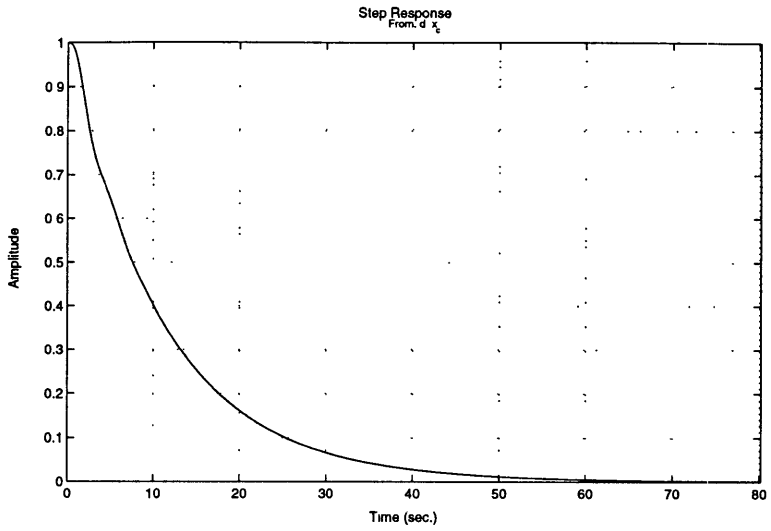

Figure 6.9: Forward Position Error Step Response: Based on Controller K_{xA}

closed loop characteristics of K_{qA}. The constraints presented were those used to find the design solution and were altered from the initial constraints as part of the iterative nature of the solution procedure. The frequency constraint for the sensitivity transfer function $S(s)$ (for command following) are the points (0.00001 Hz, -50 dB), (0.001 Hz, -40 dB), (0.01 Hz, -20 dB), (0.1 Hz, -5 dB), and (2 Hz, 1.25 dB). The first four point constraints were chosen to push the low to middle frequency gain further than K_{qA}, for command following. The last constraint was chosen to control the high frequency peak. The sensitivity transfer function constraints with the corresponding closed loop curve for controller K_{qA} are shown in Figure 6.10.

The frequency constraint for the closed loop transfer function $C(s)$ (for stability robustness) consists of two line constraints. One is an upper bound line at 0 dB from 0.01 Hertz to 2 Hertz. The other is the roll-off line from (1 Hz, 0 dB) to (10 Hz, -40 dB). These line constraints for the closed loop transfer function and the corresponding closed loop curve for controller K_{qA} are shown in Figure 6.11.

After much experience with the SCTB Constrained Optimization Tool, it is determined that a settling constraint is often sufficient for a time constraint. If the settling constraint is not too aggressive, the amplitude of the time response will remain within limits. Aggressive settling constraints will result in solutions that compensate with

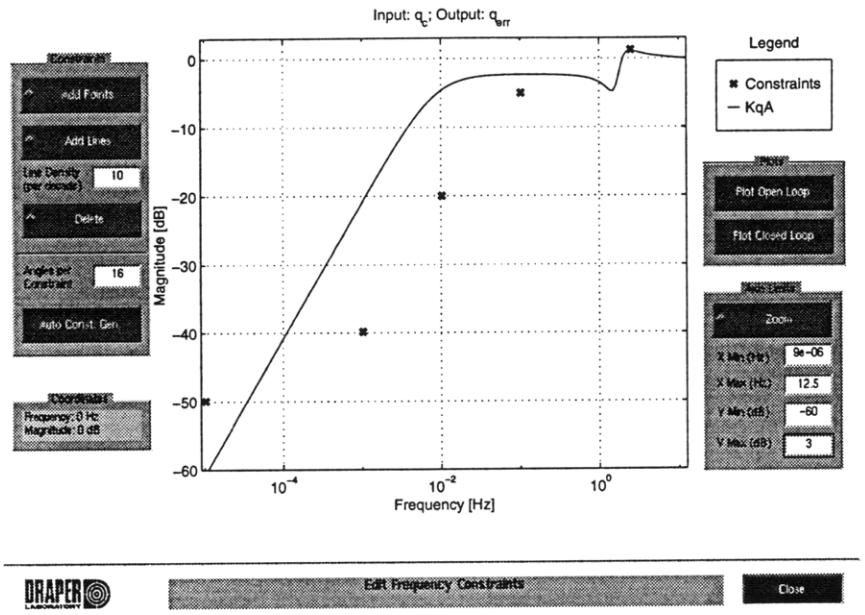

Figure 6.10: $S(s)$ Frequency Constraint Based on K_{qA}

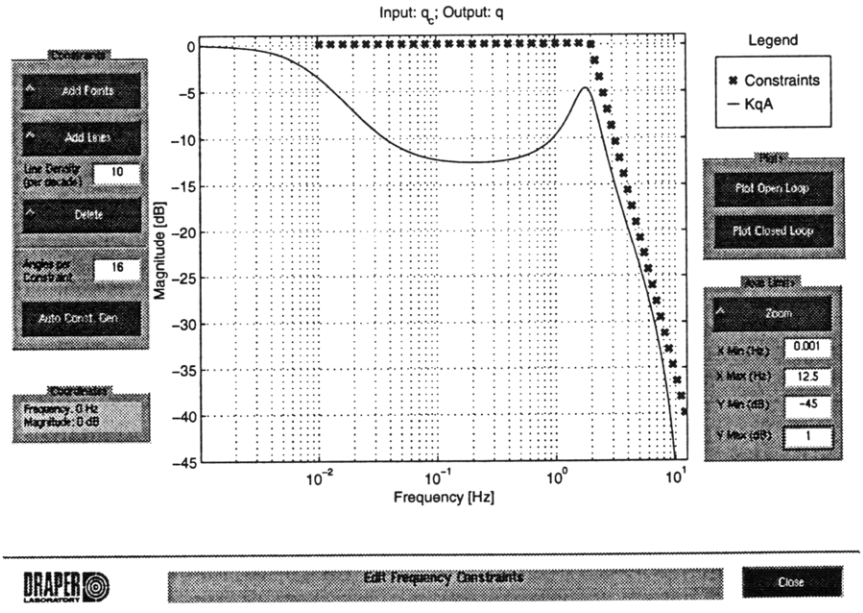

Figure 6.11: $C(s)$ Frequency Constraint Based on K_{qA}

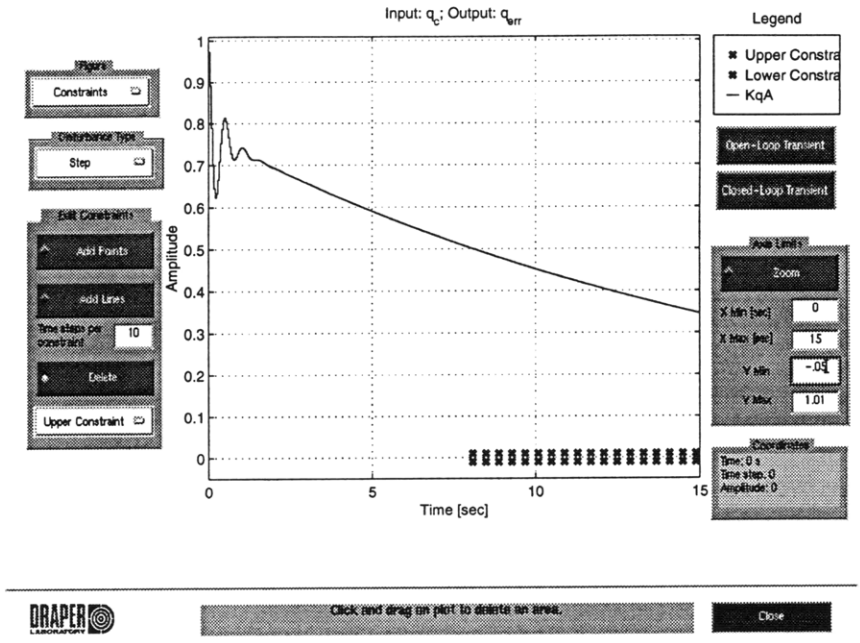

Figure 6.12: $q_{err}(t)$ Step Response Constraint

very high error amplitude peaks before meeting the settling constraint. The time constraint for the step response $q_{err}(t)$ is to settle within $\pm 1\%$ error from 8 to 15 seconds. This settling constraint and the corresponding step response for controller K_{qA} is shown in Figure 6.12.

6.3.2 Pitch Rate Loop: Basis Function Algorithm

Once the objective and constraints are in place, the Basis Function Algorithm can be applied to find an efficient solution. Based on experience and the size of the problem, the toleration limit on the number of basis functions N_{tol} is set to 100. Also, based on the size of the problem, the acceptable bound separation range λ is set to 1.

Two cases of the Basis Function Algorithm were applied. The first case did not use the time scale iteration for Legendre and Laguerre functions. Instead, the time scale frequencies of 1.25 Hertz for Legendre functions and 6.25 Hertz for Laguerre functions were applied. These were chosen because they are the nominal frequencies that the SCTB will suggest if a specific time scale is not entered. This Laguerre frequency is one fourth of the sampling rate and this Legendre frequency is one twentieth of the

sample rate. This is the only reasoning, other than convenience, behind specifically choosing these frequencies over any other. The other case used the time scale iteration to try to come up with an efficient time scale for Legendre and Laguerre functions.

As a result of numerous iterations (involving altering constraints) of the Basis Function Algorithm, an acceptable solution was found after the final set of constraints was determined to be those in the previous subsection. This best designed controller K_{qD32} (the 'D' stands for design), was considered acceptable when its first order reduction passed implementation on the DSAAV simulator. K_{qD32} was found using the fixed time scale case. The corresponding basis was comprised of 25 Laguerre functions at 6.25 Hertz, two Legendre functions at 1.25 Hertz, and two FIR functions. This produced a solution controller of 32nd order. To reach this solution, the Basis Function Algorithm solved 40 optimization problems, each between 45 seconds and 75 seconds.

When the Basis Function Algorithm with time scale iterations was applied to the same constraints, it produced an efficient 8th order solution controller. The basis was comprised of four Laguerre functions at 0.5 Hertz and one FIR function. To reach this solution, the Basis Function Algorithm solved 71 optimization problems, each requiring between 17 seconds and 64 seconds. The first order reduction appeared to be acceptable in the SCTB analysis phase. However, it had large pitch oscillations in the DSAAV simulator test. Therefore, this was an unacceptable design.

6.3.3 Pitch Rate Loop: Model Reduction and Analysis

After DSAAV simulator test failures for all controllers of third order and higher, it was determined that controllers that are second order or less are implementable. This led to the model reduction of K_{qD32}. The FBR model reduction method reduced K_{qD32} into the first order controller K_{qD}.

$$K_{qD}(s) = \frac{0.02481(s+2.3729)}{s+0.002497} \quad K_{qD}(z) = \frac{0.02598z-0.02363}{z-0.9999} \tag{6.12}$$

It is noted that this design is very similar to a proportional integral (PI) controller.

The sensitivity transfer function curves for K_{qA} and K_{qD} are shown in Figure 6.13.

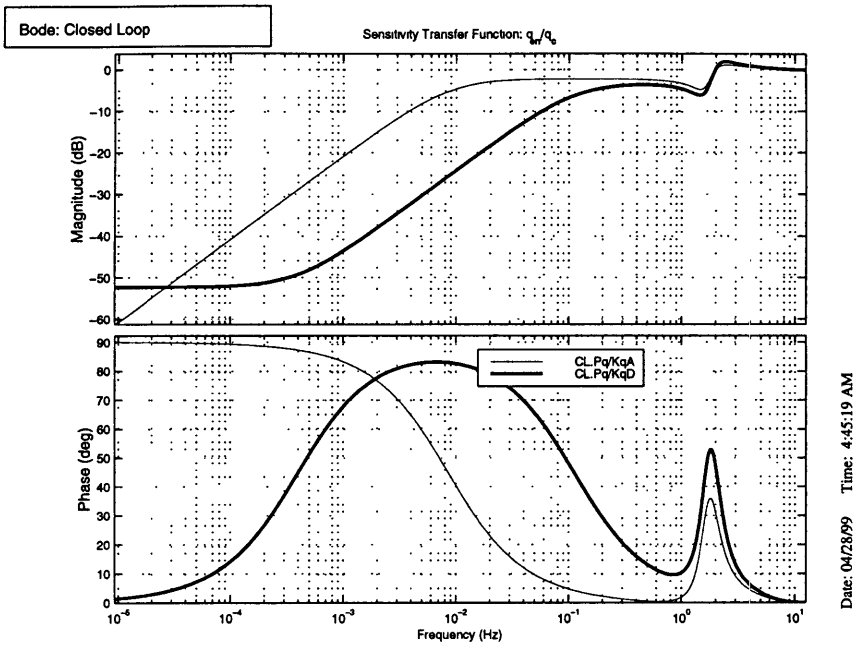

Figure 6.13: $S(s)$ for Pitch Rate Controller Designs

Here, K_{qD} (the darker curve) starts at -50 dB for the low frequency range, suggesting good command following. At the high frequency end, the two designs share similar dynamics.

The closed loop transfer function curves for the controller designs are shown in Figure 6.14. Here, both designs roll off well at the high frequency end. Although the classical controller (darker curve) is at a lower gain than the constrained optimization controller, both curves roll off well enough to suggest sufficient stability robustness.

The step responses for the controller designs are shown in Figure 6.15. The step response for the constrained optimization controller (solid line) performed well, settling to $\pm 1\%$ error in 6.4 seconds. This represents a significant improvement over the classical design, which settles to $\pm 1\%$ error in 81.1 seconds.

DSAAV Simulator

To determine the implementability of the designs, they were tested on a DSAAV simulator. This simulator shows a picture of the DSAAV helicopter in a field as it flies the predetermined flight script on the screen. There are four main possible

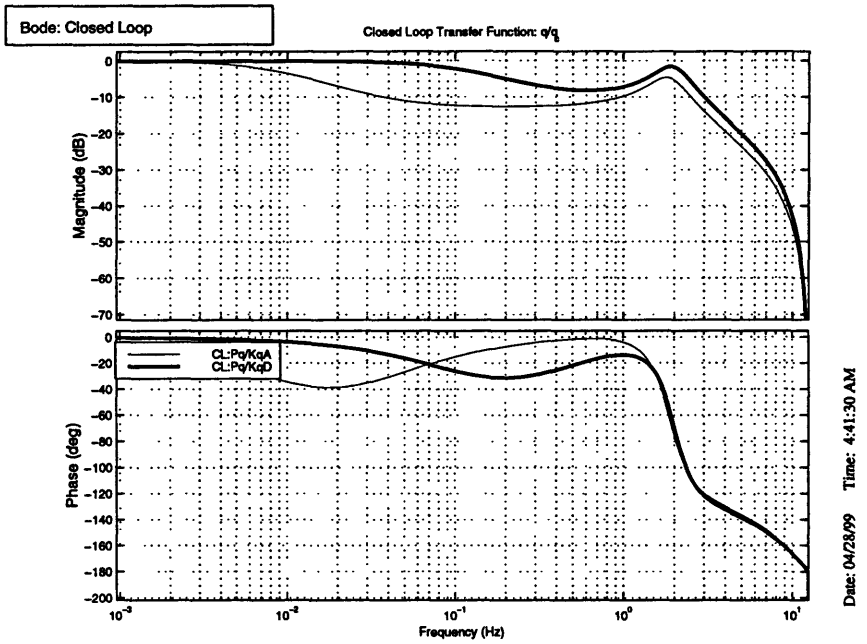

Figure 6.14: $C(s)$ for Pitch Rate Controller Designs

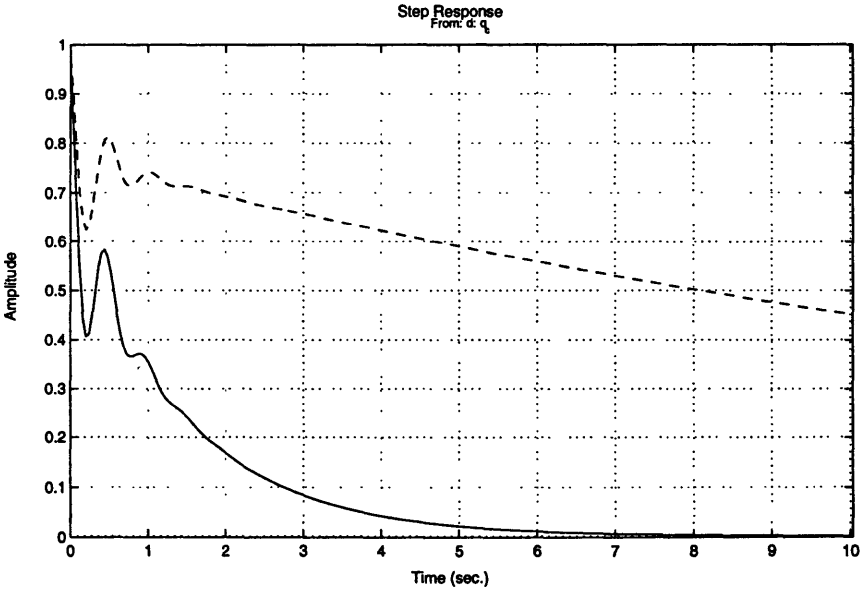

Figure 6.15: Step Responses for Pitch Rate Controller Designs

results from a simulator flight test. One is a smooth flight where the DSAAV flies its script well without oscillations or diversions from its flight path. The second result is that the DSAAV will oscillate throughout its entire flight. For the forward motion loop, these will be pitch oscillations. If the oscillations are moderate enough, the DSAAV will fly and complete its flight script, oscillating most of the way. Oscillations represent a stability problem. The third result is the DSAAV will deviate from its scripted flight path. In the moderate case, the control does not follow commands well and passes the scripted landing point. A higher level guidance loop will try to compensate for this and turn the DSAAV around in an attempt to return to the scripted flight path. The extreme case has the DSAAV immediately deviating from the entire flight path. This is usually coupled with oscillations and represents a stability and command following problem. The final case is where the DSAAV crashes. In the simulator, when the DSAAV crashes, it will often bounce off the ground and flip around like a football until the simulator decides to end the simulation with an error message. The crashing DSAAV indicates a design that is poor and can have problems that may not be apparent. At the very least, it is obvious that there is a stability problem.

6.3.4 Solution Controllers of Outer Loops

After the design of K_{qD}, the solution procedure was applied to the pitch loop. The constrained optimization tool was able to design higher order controllers that were improvements upon $K_{\theta A}$. Unfortunately, when these controllers were reduced to an implementable order, the performance loss made the reduced order designs worse than $K_{\theta A}$. Since the classical controller was just a proportional controller, it was decided to keep $K_{\theta A}$ as the pitch controller. There is still a difference between the classical design and the constrained optimization design from the difference in pitch rate controllers. This is depicted in the closed loop step response of the pitch error in Figure 6.16. The constrained optimization design (solid line) has more overshoot than the classical design, but reduces the settling time from 13.7 seconds to 4.2 seconds.

The forward velocity design was very similar to the pitch rate design. The reduced

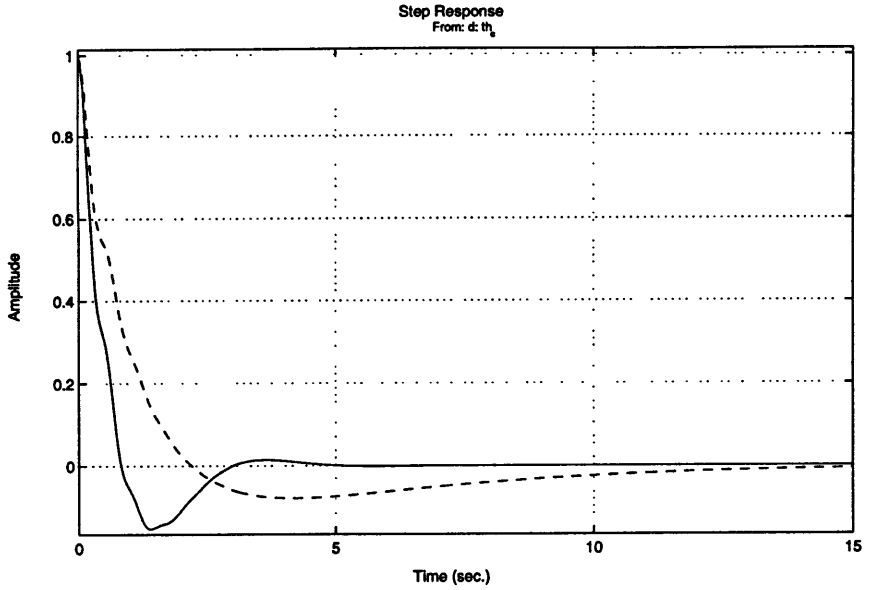

Figure 6.16: $\theta_{err}(t)$ for Pitch Controller Designs

first order design K_{uD} is as follows:

$$K_{uD}(s) = \frac{-0.06531(s+0.078956)}{s+0.001837} \quad K_{uD}(z) = \frac{-0.06541z+0.06521}{z-0.9999}. \tag{6.13}$$

This was reduced from a 37th order design. It is also noted that K_{uD} is similar to a PI controller. The comparison between the classical design and the constrained optimization design is shown in the closed loop step response of the forward velocity error in Figure 6.17. The constrained optimization design (solid line) smoothes the error curve reducing the classical design overshoot from 41% to 30% while maintaining a similar settling time (11.6 seconds to 13.8 seconds). Most importantly, the forward velocity design passed the DSAAV simulation test.

After the design of K_{uD}, it was time to design the outermost forward position loop controller K_{xD}. Since the classical controller K_{xA} was a proportional controller, like $K_{\theta A}$, it was decided to keep K_{xA} as the forward position controller. The result was a minimal difference between classical design and the constrained optimization design. This is shown in the closed loop step response of the forward position in Figure 6.18. The only difference between the designs is not very remarkable. This difference is that the constrained optimization design (solid line) reduced the settling time from 51.9 seconds to 51.8 seconds.

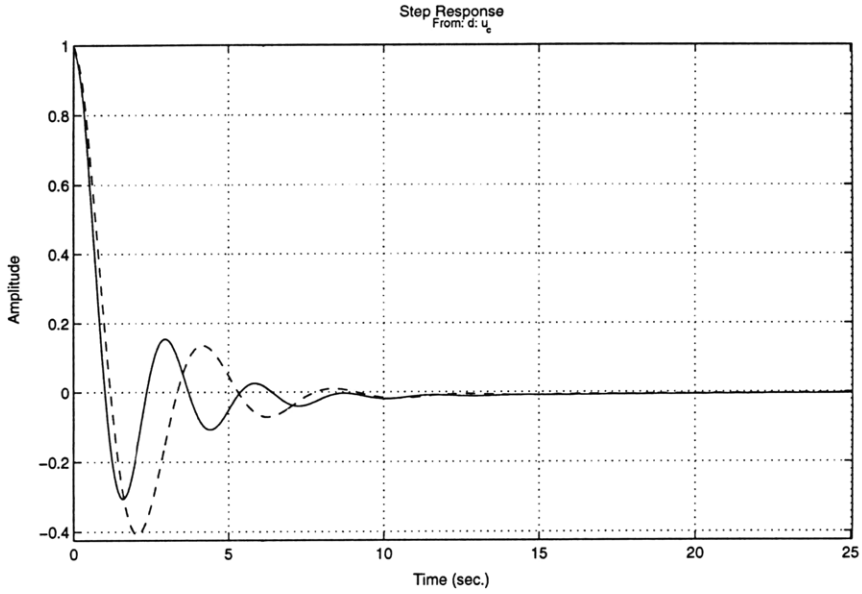

Figure 6.17: $u_{err}(t)$ for Forward Velocity Controller Designs

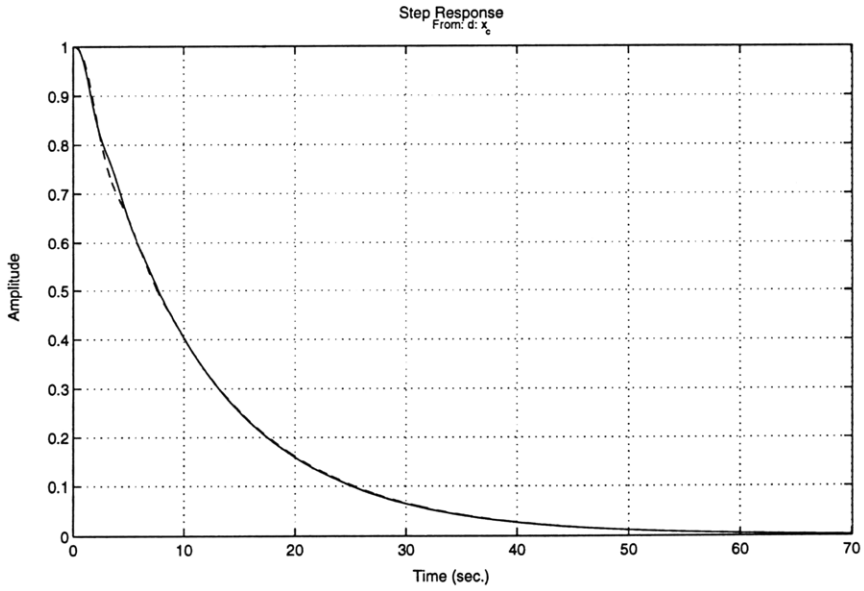

Figure 6.18: $x_{err}(t)$ for Forward Position Controller Designs

6.3.5 Controller Design for Coupled Forward Motion

To compare the controller design process of the SISO hierarchical structure to the entire coupled MIMO system, the same constraints that were applied to the SISO solutions in the previous subsections were applied to the coupled MIMO forward motion channel. Not only was the SCTB unable to find a feasible solution, it was taking well over two hours to determine that the problem was infeasible. In addition, the SCTB was returning unsolved controllers of over 600th order. If there was a feasible solution to the coupled problem, it would take way too long and be way too complex to be beneficial. With the hierarchical structure, the solutions were always found in less than ten minutes and were rarely solved in more than two and a half minutes. This allowed the Basis Function Algorithm to solve the optimization problem 40 to 80 times for one design.

6.3.6 Q-Minimization Controller Design

The constrained optimization controllers designed for the SISO loops of the DSAAV forward motion channel were all based on the $\mathcal{H}_2$ objective function. Although the $\mathcal{H}_2$ objective was sufficient in designing these controllers, the Q-minimization objective required investigation. Using the designed pitch rate controller K_{qD} as the nominal controller, the Q-minimization objective was used to design a pitch rate controller to modestly improve upon the K_{qD} design. The same frequency constraints used to design K_{qD} were applied to the Q-minimization design. The difference was a slight tightening of the step response constraint: to settle within $\pm 1\%$ error from 6 to 15 seconds (instead of 8 to 15 seconds).

The Q-minimization pitch rate controller was very similar to K_{qD}. The reduced first order design K_{qQ} is as follows:

$$K_{qQ}(s) = \frac{0.022458(s+2.618)}{s+0.0024945} \quad K_{qQ}(z) = \frac{0.02363z-0.02128}{z-0.9999}. \tag{6.14}$$

This was reduced from a 6th order design. K_{qQ} is also very similar to a PI controller design. The improvement was very modest: reducing the settling time from

6.4 seconds to 6.2 seconds. Most importantly, K_{qQ} passed the DSAAV simulation test.

6.3.7 Observations on Constraints

A major element of the design procedure is adjusting constraints to achieve feasible solutions and desired performance. Knowing where to place constraints is not an exact science and is dependent on the designer's experience and intuition. From this DSAAV example, there were noted trends regarding constraints.

For time constraints, it was usually sufficient to apply a settling constraint. This is more desirable than adding an amplitude constraint because more constraints adds complexity to the optimization, resulting in infeasible problems or higher order solutions. When applying only settling constraints, do not place them too early. Constraints that force the error to decay too rapidly will result in error curves with high amplitude peaks before settling. Settling constraints also need to be applied for seven to twenty seconds, at the very least. Settling constraints that do not endure can lead to error curves that either do not settle, settle at a non-zero value, or regain a significant amount of amplitude before settling.

Often, designs that performed well in the design phase did not perform well in the DSAAV simulator. In the design phase, the decoupled dynamics are linearized, where the simulator includes the non-linearized dynamics. Once a design was successful in the simulator, the frequency domain characteristics of that design can be used for determining the implementability of future designs. For a given loop, implementable designs (designs that pass the simulator) often would have similar characteristics in the mid to high frequency range.

6.4 Conclusions

Replacing the pitch rate and forward velocity classical controllers with the constrained optimization designs of K_{qD} and K_{uD} showed considerable improvement in the pitch rate performance, noticeable improvement in the pitch and forward velocity perfor-

mance, and negligible improvement in the forward position performance. This indicates that the constrained optimization method has the potential to find controllers that are improvements over classical designs. In addition, the Q-minimization objective may be used successfully, in the procedure, to modestly improve upon well designed controllers.

The solution procedure was able to find successful controllers. However, the time scale iteration for Legendre and Laguerre functions is suspect. As shown in the pitch rate loop, the nominal Legendre and Laguerre time scales proved to be more successful than the detailed time scale iteration. This is despite the time scale iteration finding more efficient (lower order) controllers. Two possible explanations come to mind. One explanation is that the model used for design is not as accurate as the DSAAV simulator model. The other explanation is that choosing the correct time scale for these functions is still an evolving art, requiring further investigation.

Another conclusion is that the hierarchical structure is required for constrained optimization of highly coupled systems. The highly coupled MIMO system is too complex for all the states of interest to be constrained at the same time. Although the set of constraints should remain feasible, the coupled problem is too computationally complex for the constrained optimization procedure to solve. Also, the probable solution will be of an unmanageable order. The hierarchical structure allows the design to focus on each pertinent state and make adjustments swifter and easier.

Chapter 7

Conclusions

A constrained convex optimization control design procedure that takes advantage of a decoupled hierarchical system structure has been presented in this thesis. The solution procedure was applied to an autonomous helicopter, to show a practical application of the procedure. The experimental results did not show significant improvement in the outer loop (position) performance. However, within the coupled channel, there was improvement in the performance of the inner loop dynamics (attitude, body rates). Constrained optimization allowed specifications to be chosen directly as constraints and the hierarchical structure allowed for controller specialization, focusing on improvement of one pertinent state at a time. The hierarchical structure also allowed for a modular design, where two of the controllers were found using the constrained optimization solution procedure and the other two were the previously designed classical controllers. Modular design is attractive because different approaches might be better for different levels of the hierarchy, depending on the dynamics for each level's control problem.

The decoupled hierarchical structure also emphasized a significant limitation on constrained optimization: available computational power. The highly coupled MIMO control system is almost impossible to solve with constrained optimization. However, the hierarchical structure makes the problem tractable by breaking the problem down into simpler SISO loops. Computer weakness is still an issue for the SISO loops, hindering constrained optimization if there are too many constraints or too many

basis functions.

The constrained optimization solution procedure proved to be effective for DSAAV controller design. In addition, the procedure demonstrated the advantages of the SCTB MATLAB linear controller design tool. However, the procedure is far from perfect. The procedure was able to estimate an appropriate time scale for the Legendre and Laguerre basis functions that made for a more efficient (lower order) solution. However, these efficient solutions were proven unimplementable, failing in the simulator. This is in contrast to higher order solutions made implementable through model reduction. The procedure was unable to distinguish between unimplementable, efficient solutions and higher order solutions made implementable after model reduction.

7.1 Recommendations for Future Work

The most difficult portion of solving the constrained optimization control problem is finding the most appropriate basis for estimating Q. This thesis showed an example where the most efficient basis (lowest order) for a set of constraints may not be the most appropriate basis for a given control problem. Therefore, a higher order solution based on a higher order basis might provide the best solution. There are numerous design variables that come with choosing a basis. An obvious area of uncertainty is deciding how many of each type of basis function to include. Then there is the issue of the proper time scale for the Legendre and Laguerre basis, if they are selected. If there is a previous design, poles from that Q may be added to the basis as well. If any of these decisions are errant, the basis may be inefficient or make the optimization problem infeasible. Therefore, if constrained optimization is to be used in future control problems, as it should, more work needs to be done to demystify the art of finding the best basis. At the very least, reducing the uncertainty of one of the design variables of basis choice, such as time scale, can greatly simplify the problem.

In addition to the uncertainty in choosing design variables for the basis functions, there are many design variables when it comes to selecting objective functions and constraints. The SCTB gives three options for objective functions. This thesis only

fully investigated the $\mathcal{H}_2$ minimization objective and lightly touched upon the Q-minimization objective. The ℓ_1 objective was not even addressed. In addition, the impact of the frequency weighting on the objective was not fully addressed. As for constraints, gain margin and phase margin constraints may be beneficial to a constrained optimization design.

Another avenue for future investigation is a more systematic procedure for mapping constraints within the hierarchical structure. This involves determining constraints on the innermost loops to a given constraint for the outermost loop. The objectives of the outer loops would then depend on the capabilities of the lower levels. The hierarchical structure should provide insight into determining appropriate objectives and constraints for all the loops. This thesis provided some intuitive observations about the relationships between frequency domain constraints and closed loop performance, but a more systematic approach is needed to facilitate automation of the constrained optimization procedure.

As mentioned earlier, computational power is an obstacle to constrained optimization control. To take advantage of the increasing computing power that comes with the advances of technology, it is important that the software available for solving constrained optimization problems is always adapting and improving upon the latest advances in operations research. For example, MINOS 5.4, the software used to solve the constrained optimization problems in the SCTB is dated December 1992 and its user's guide is dated February 1995 [14]. Software that incorporates the model reduction as part of the optimization would be extremely useful for systems with implementation limits.

Bibliography

[1] S.P. Boyd, V. Balakrishnan, C.H. Barrat, N.M. Khraishi, X. Li, D.G. Meyer, and S.A. Norman, "A New CAD Method and Associated Architectures for Linear Controllers," *IEEE Transactions on Automatic Control.* vol. 33, pp. 268–283, March 1988.

[2] K.A. Fegley, S. Blum, J.O. Bergholm, A.J. Calise, J.E. Marowitz, G. Porcelli, and L.P. Sinha, "Stochastic and deterministic design and control via linear and quadratic programming," *IEEE Transactions on Automatic Control,* vol. AC-16, no. 6, pp. 759–765, December 1971.

[3] C.L. Gustafson and C.A. Desoer, "Controller design for linear multivariable feedback systems with stable plants, using optimization with inequality constraints," *International Journal of Control,* vol. 37, no. 5, pp. 881–907, 1983.

[4] C.L. Gustafson and C.A. Desoer, "A CAD methodology for linear multivariable feedback systems based on algebraic theory," *International Journal of Control,* vol. 41, no. 3, pp. 653–675, 1985.

[5] T.C. Henderson, M.D. Piedmonte, L.K. McGovern, J.W. Jang, B.V. Lintereur, *Structural Control Toolbox.* User's Guide, Charles Stark Draper Laboratory, November 1997.

[6] B.V. Lintereur and L.K. McGovern, "Constrained $\mathcal{H}_2$ Design via Convex Optimization Applied to Precision Pointing Attitude Control," *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, December 1997.

[7] B.V. Lintereur, *Constrained $\mathcal{H}_2$ Design via Convex Optimization with Applications.* Masters Thesis, MIT, June 1998.

[8] J.M. Maciejowski, *Multivariable Feedback Design*. New York: Addison Wesley, 1989.

[9] L.K. McGovern, *A Constrained Optimization Approach to Control with Application to Flexible Structures*. Masters Thesis, MIT, June 1996.

[10] L.K. McGovern and T.C. Henderson, "Design of an Active Vibration Isolation Control System Using Constrained Optimization," *Proceedings of the American Control Conference*, Albuquerque, NM, pp. 3115–3120, 1997.

[11] D.G. Meyer, "Fractional Balanced Reduction: Model Reduction via Fractional Representation," *IEEE Transactions on Automatic Control*. vol. 35, no. 12, pp. 1341–1345, December 1990.

[12] D.G. Meyer, *Model Reduction via Fractional Representation*. Ph.D. Thesis, Stanford University, April 1987.

[13] B.C. Moore, "Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction," *IEEE Transactions on Automatic Control*. AC-26, pp. 17–31, February 1981.

[14] B.A. Murtagh and M.A. Saunders, *MINOS* 5.4 *User's Guide*. Systems Optimization Laboritory, Stanford University, February 1995.

[15] E. Polak and S.E. Salcudean, "On the Design of Linear Multivariable Feedback Systems via Constrained Nondifferentiable Optimization in $\mathcal{H}_\infty$ Spaces," *IEEE Transactions on Automatic Control*, vol. 34, no. 3, pp. 268–276, March 1989.

[16] C.P. Sanders, P.A. DeBitetto, E. Feron, H.F. Vuong, and N. Leveson, "Hierarchical Control of Small Autonomous Helicopters," *Proceedings of the IEEE Conference on Decision and Control*, Tampa, Florida, December 1998.

[17] W.L. Winston, *Operations Research: Applications and Algorithms*. Belmont, CA: Duxbury Press, 1994.

[18] D.C. Youla, H.A. Jabr, and J.J. Bongiorno, "Modern Wiener-Hopf design of optimal controllers—Part II: The multivariable case," *IEEE Transactions on Automatic Control*. vol. AC-21, no. 3, pp. 319–338, June 1976.