AFIT/GAE/ENY/96J-1

Gain-Scheduled Aircraft Control
Using Linear Parameter-Varying Feedback

THESIS
Martin R. Breton
Captain, CAF

AFIT/GAE/ENY/96J-1

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government or the Government of Canada.

AFIT/GAE/ENY/96J-1

Gain-Scheduled Aircraft Control Using Linear Parameter-Varying Feedback

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Martin R. Breton, B.Eng., Electrical Engineering
Captain, CAF

June, 1996

Acknowledgements

I would first like to thank my advisor, Dr Brett Ridgely, whose enthusiasm and encouragement were a source of motivation throughout my time at AFIT. His support and guidance has enabled me to understand concepts and ideas which seemed at times beyond my grasp, and has given me the confidence to hopefully continue my studies in this field.

I would secondly like to thank the members of my committee, LtCol Stuart Kramer and Major Bob Canfield, and my fellow students for all their suggestions and helpful comments. I am also indebted to the Wright Lab Flight Dynamics Group, particularly Capt Mark Spillman, Lt Paul Blue and Lt Bill Reigelsperger who were always there to provide advice and exchange ideas. They were a tremendous resource and I am very grateful for the use of some of their algorithms which greatly helped me in my work.

Finally, I would also like to thank my family and friends, especially my parents Lily and Joe and my sister Lucy, who are always there for me. I am very lucky and honored to be a part of their lives.

Martin R. Breton

Table of Contents

List of Figures

List of Tables

Abstract

Systems which vary significantly over an operating envelope, such as fighter aircraft, generally cannot be controlled by a single linear time-invariant controller. As a result, gain-scheduling methods are employed to design control laws which can provide the desired performance. This thesis examines a relatively new approach to gain-scheduling, in which the varying controller is designed from the outset to guarantee robust performance, thereby avoiding the disadvantages of point designs. Specifically, the parameter-varying (LPV) aircraft model is linearized using linear fractional transformations (LFT's), and the resulting control problem is characterized as the solution to a set of four linear matrix inequalities (LMI's). The supporting theory is reviewed and two pitch-rate controllers are designed; one for the full longitudinal aircraft model, and another for the short period model. It is found that, even though the varying controllers are quite conservative, they can guarantee better robust performance over a large portion of an operating envelope when compared to time-invariant μ-synthesis controllers.

Gain-Scheduled Aircraft Control
Using Linear Parameter-Varying Feedback

I. Introduction

In the past decade, much of modern control theory has focused on determining the ideal controller to optimize the performance of some given system. This usually requires simplifying a complex real world system into a simpler workable mathematical model. This introduces many uncertainties due to, for example, neglected plant dynamics, nonlinearities, or inaccurate knowledge of the model parameters or how they may vary during operation. To compensate, the designer must then account for these uncertainties in his system model. The challenge becomes one of designing a controller which can obtain as much performance as possible in the presence of these modelled uncertainties. Inevitably, there is a tradeoff between performance and robustness and the designer must then use his skill to design a controller which can satisfy both.

Generally, if the system does not vary too significantly, it can be modelled fairly accurately as a linear time-invariant (LTI) plant with some small uncertainties; in such cases, standard H_∞ and μ-synthesis theory can be used to find a controller which satisfies both the performance and robustness objectives. Unfortunately, if the system varies considerably during operation, a satisfactory level of performance often cannot be achieved without sacrificing robustness, and vice versa. Under such demanding conditions, designers usually adopt a gain-scheduling approach.

Simply put, gain-scheduling is a method of changing the controller depending upon the operating conditions. There are two general schools of thought on how to generate these time-varying controllers. The first and more conventional approach

involves determining robustly performing controllers for several point designs, and then designing a method of scheduling the individual controllers such that robust performance is maintained at each design point. The resulting schedule can be as simple as a look-up table, or can involve designing a mathematical control law relating or approximating the individual controllers (see, for example, [Bla91], [AW89], [LR93], and [SC92]). While these types of methods have the advantage of designing optimal controllers for actual points in the operating envelope using conventional design methods, they also have significant disadvantages. Foremost, the designer needs to design several robust controllers to adequately cover the operating envelope; in doing so, he fixes the plant to specific operating points and neglects its time-varying nature. As a result, robust performance can no longer be guaranteed throughout the operating envelope. In addition, depending on the order of the controllers, the resulting scheduling problem is often quite complex. The individual controllers can be simplified, but this often comes at the expense of performance unless the parameter variations are moderate to begin with. In addition, robust stability throughout the operating envelope can only be guaranteed under certain specific conditions [SA91a],[SA91b]. In fact, much of the gain-scheduling research done in the past has focussed on addressing these issues.

A new school of thought has emerged in recent years that essentially reverses the classical gain-scheduling problem. Instead of designing several point controllers first and then trying to determine a relationship between the controllers and the changing operating parameters, these methods focus on designing the relationship between the gain-scheduled controller and the changing parameters from the outset. Once this relationship is known, appropriate controllers for all operating conditions can then be determined from values of the parameters. In short, these approaches directly design time-varying controllers. An obvious advantage is that the time-consuming process of designing point controllers and then a schedule is completely avoided; in addition, robust performance can be guaranteed over the entire design

envelope. However, these methods all depend on determining relationships between a linear time-varying plant and the varying parameters. While the relationship between the plant and the parameters can be expressed in several different ways, to date this invariably introduces conservatism at some point in the process which degrades the achievable level of robust performance.

Specifically, the method developed by Packard, Apkarian, and Gahinet [Pac94], [AG95] (on which this thesis is based) models linear parameter-varying (LPV) plants and controllers as, respectively, LTI plants and LTI controllers linearly dependent upon a separate varying parameter matrix. This parameter block is then treated as an uncertainty and the resulting H_∞ design problem is solved for the desired controller using Small Gain theory. Unfortunately, due to the Small Gain Theorem, substantial conservatism is introduced because the resulting controller is designed to address not only real parameter values, but also complex parameter values which will not exist for most physical systems.

Another method, developed by Becker [BP94], avoids breaking up the LPV plant; instead, the LPV state-space of the plant is kept within the constraints of the optimization problem. This avoids the conservativeness of the Small Gain Theorem, but requires solving an infinite number of constraints since the parameters are continuous. One way to resolve this is to discretize the parameters and formulate the constraints at each combination of parameter values. The resulting stack of finite constraints can then be solved for the desired controller. In particular, if the state-space matrices are affine functions of the parameters, then it has been shown that only the constraints corresponding to the vertices of the operating envelope need to be solved. This entire approach, however, assumes that all convex combinations of the LPV state-space matrices exist in the modelled system, which is not necessarily true and again introduces substantial conservatism.

Finally, a third approach introduced by Wu [WYPB94] extends the previous one to account for the rates of parameter variations. The first two approaches do

not place any bounds on the rates of the parameter variations, and the controllers are then designed to address any rate of change, no matter how unrealistic. In Wu's approach, bounds on the parameter rates of change are included and form part of the constraints to be solved. Unfortunately, this method also results in an infinite number of constraints; in addition, the set of matrix functions from which the controller is extracted is also infinite. Thus, while it does address parameter rates of change, it suffers from all the disadvantages of Becker's approach and its numerical tractability is even more difficult.

1.1 Objectives

The main objective of this thesis is to apply some of the latest techniques in gain-scheduling to a realistic problem. Specifically, an extension of the method of Packard, Apkarian, and Gahinet for uncertain LPV systems [AG95], [SLBB96] will be used to design a robustly performing pitch rate controller for an F-18 Supermaneuverable fighter aircraft. This extension incorporates the advantages of μ-synthesis to handle remaining uncertainties and add even more flexibility to the design. Using these techniques, controllers for both the full longitudinal state-space aircraft model and the short period aircraft model will be designed, simulated, and then compared.

In support of this thesis, most of the required theory has been included. A secondary objective was to become familiar with the underlying theory of linear matrix inequalities (LMI's), which are now often used to express the constraints of complex modern control problems. This work represents a very practical use of the newly released MATLAB LMI Control Toolbox [MAT],[GNLC92].

1.2 Outline

Following this introduction, the remainder of the thesis is divided into five chapters. Chapter II introduces some of the mathematical and control theory fundamentals essential to the thesis. While the reader is expected to be familiar with general H_∞ theory, this chapter reviews the concepts of linear fractional transfor-

mations (LFT's), LMI's, and μ-synthesis which will be used extensively in later chapters.

Chapter III describes the gain-scheduling theory for LPV systems introduced by Packard, Apkarian, and Gahinet [Pac94], [AG95]. It also describes the extension for uncertain LPV systems mentioned previously [SLBB96]. All of the constraints to solve the resulting problems are formulated as LMI's which must then be solved to obtain the LPV controller.

Chapter IV details the implementation of the theory to solve for the gain-scheduled pitch rate controller for the F-18 aircraft under study. The entire setup of the problem is described, beginning with how to obtain an LPV plant representation and how to then convert the LPV plant into an LTI plant linearly dependent upon a varying parameter block. The design model is then developed; for this particular design, a model-matching framework is adopted with a 2-degree-of-freedom controller. The selection of weights and design envelopes are then discussed, along with some practical implementation details which may not be readily apparent.

Chapter V presents the numerical results and simulations. The resulting controllers are described and then simulated to examine their performance throughout and beyond their design envelopes. Various types of simulations are performed and evaluated; as well, the time responses of LTI μ-synthesis controllers is included for comparison purposes.

Finally, Chapter VI summarizes the results, discusses some of the problems, and recommends some areas for further study.

Several appendices provide further information. Appendix A lists the aircraft flight conditions used as data in order to establish a relationship between the aircraft model and the varying parameters; Appendix B illustrates the resulting relationships. As well, most of the MATLAB programs for the implementation of the control

problem have been provided in Appendix C. Finally, the simulation programs are included in Appendix D.

II. Preliminaries: LFT's, LMI's, and μ

This chapter briefly reviews the concepts of Linear Fractional Transformations (LFT's), Linear Matrix Inequalities (LMI's), and the Structured Singular Value, a matrix function denoted by μ. It is intended only as an introduction to familiarize the reader with concepts used throughout this thesis, and is by no means a comprehensive review of the subjects. For a more thorough coverage, the reader should consult [BDG+91], [ZPD82], [PZPB91], [ZDG96], [BEFB94], [GNLC92], and the papers referenced therein.

2.1 Linear Fractional Transformations

LFT's are used extensively in modern control theory to describe the relationship between components of a linear model; most commonly, of a plant and its controller, or of a system and its set of possible uncertainties. Specifically, consider a matrix M such that

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{2.1}$$

Now suppose there is an appropriately dimensioned block structure Δ_l (such that $M_{22}\Delta_l$ is square) that is related to M as shown in Figure 2.1.

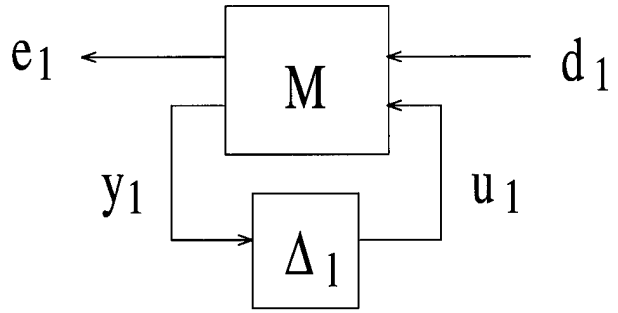


Figure 2.1 Lower LFT

The feedback equations for this system are then

$$\begin{aligned} e_1 &= M_{11}d_1 + M_{12}u_1 \\ y_1 &= M_{21}d_1 + M_{22}u_1 \\ u_1 &= \Delta_l y_1 \end{aligned} \tag{2.2}$$

Note that this is very general and could represent a state feedback or output feedback system, or the relationship between a nominal M and an uncertainty Δ_l, or any other similar relationship. Solving the above equations for the transfer function relating the input d_1 to the output e_1 gives the following

$$F_l(M, \Delta_l) = M_{11} + M_{12}\Delta_l(I - M_{22}\Delta_l)^{-1}M_{21} \tag{2.3}$$

$F_l(M, \Delta_l)$ is called the *lower LFT* on M by Δ_l because, as illustrated in Figure 2.1, the "lower" loop of M is the one closed by Δ_l. In addition, the system is considered well-posed (i.e. it has a unique solution) only if the inverse of $(I - M_{22}\Delta_l)$ exists; otherwise, we have a singular problem with either no solution or an infinite number of solutions.

Similarly, Figure 2.2 illustrates the relationship leading to the *upper LFT* formula

$$F_u(M, \Delta_u) = M_{22} + M_{21}\Delta_u(I - M_{11}\Delta_u)^{-1}M_{12} \tag{2.4}$$

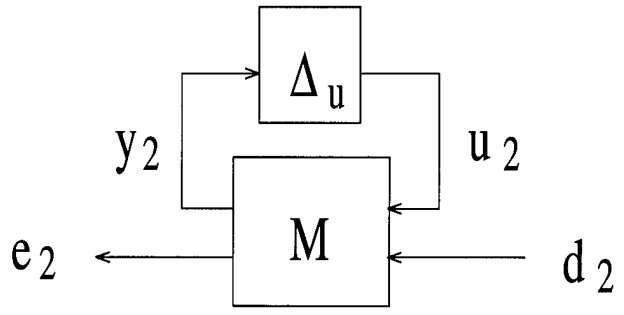


Figure 2.2 Upper LFT

The upper LFT and lower LFT expressions are each often referred to as simply LFT's, and the context or a diagram indicates the actual expression to be used.

Several interpretations of LFT's will be used extensively in this thesis. The most common interpretation and use of these expressions is that the LFT's are simply the closed-loop transfer functions from d_1 to e_1 in the case of the lower LFT, and from d_2 to e_2 in the case of the upper LFT. In other words,

$$T_{e_1 d_1} = F_l(M, \Delta_l) \tag{2.5}$$

$$T_{e_2 d_2} = F_u(M, \Delta_u) \tag{2.6}$$

where M might represent the controlled plant and Δ could be the model uncertainties or the controller.

Another useful interpretation of an LFT (for example, $F_l(M, \Delta_l)$) is that it includes a nominal M_{11} independent of Δ_l but perturbed by Δ_l, while M_{12}, M_{21}, M_{22} reflect *a priori* knowledge of how the perturbation will affect the nominal M_{11}. A similar interpretation can be made with $F_u(M, \Delta_u)$, with M_{22} then considered as the nominal mapping.

The LFT expressions can also be used to determine the transfer function for the state space realization of a system. A system given by

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.7}$$

has a transfer function $G(s) = D + C(sI - A)^{-1}B$. Not surprisingly, this can also be expressed as an LFT

$$G(s) = F_u\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}, \Delta\right) \tag{2.8}$$

where $\Delta = \frac{1}{s}I$ is now a matrix representing the frequency structure of the state space realization. In this case, Δ is more of a mathematical device than a "physical" disturbance or parameter matrix.

Such transfer functions $G(s)$ representing a system state space realization will often be denoted as

$$G = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \tag{2.9}$$

It is important to note that $\left[\begin{array}{cc} A & B \\ C & D \end{array}\right]$ is commonly referred to as a coefficient matrix, whereas $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ is referred to as a transfer matrix. Unlike the coefficient matrix, the transfer matrix implies that an upper LFT transformation with $\Delta = \frac{1}{s}I$ has been performed on the coefficient matrix, so that the only actual input/output channels remaining are those corresponding to the C, D rows. The A, B rows now represent internal states of the transfer matrix.

An important fundamental property of LFT's is that linear interconnections of LFT's can be always be grouped together into one combined LFT; that is, all of the uncertainties can be isolated into one block diagonal structure affecting a combined known fixed system. For example, consider the interconnection shown in Figure 2.3 consisting of three components, each of which is an LFT.

By collecting all of the known systems together and collecting all of the uncertainties together, the system can be redrawn as in Figure 2.4 which is now an LFT of a general known component perturbed by a block diagonal uncertainty structure. This uncertainty model is commonly referred to as *structured uncertainty* and is due to the fact that the uncertainty of each component is independent of the other component input/output channels.

There are many excellent examples of the uses of LFT's in control or mathematics given in [ZDG96],[ZPD82]. One particular use relevant to this thesis concerns expressing a nonlinear polynomial function of one or more indeterminate variables

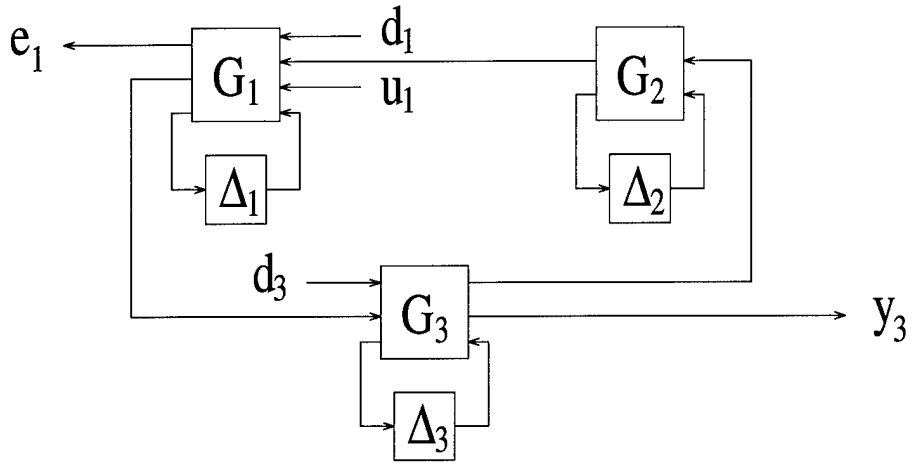


Figure 2.3 Linear interconnection of three LFT's

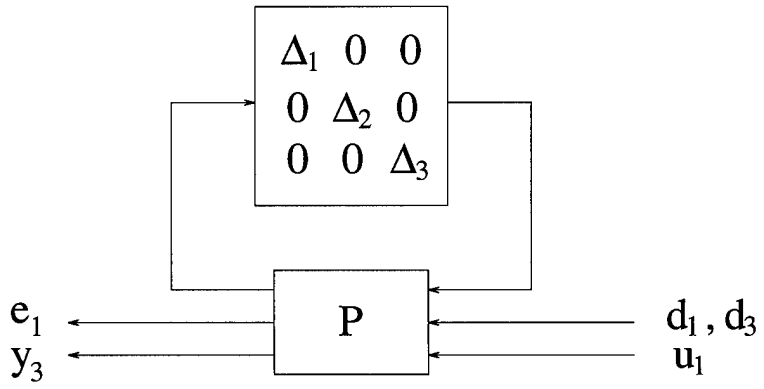


Figure 2.4 Combination of the LFT's with resulting structured uncertainty model

in an LFT form, thereby making it a linear model of the function. For example, consider the input/output relation

$$z(\theta_1, \theta_2) = (a + b\theta_2 + c\theta_1\theta_2^2)w := Xw \tag{2.10}$$

where a, b, and c are constants, and θ_1 and θ_2 are indeterminate. In order to express X as an LFT in terms of θ_1 and θ_2, a diagram showing the input/output relations with each θ should first be drawn, denoting the inputs and outputs of the θ's as y's and u's. For our example, this is shown in Figure 2.5.

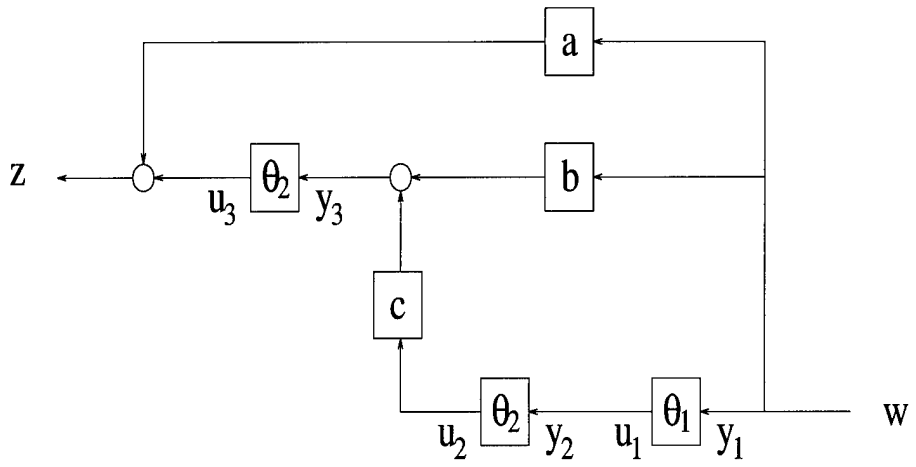


Figure 2.5 Block diagram of X showing input/output relations

The outputs y and z can now be written in terms of the inputs u and w, without referring to the indeterminate θ's. This essentially pulls out the θ's into one indeterminate block diagonal structure. Thus, for this example:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ z \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & c & 0 & b \\ 0 & 0 & 1 & a \end{bmatrix}}^{M} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ w \end{bmatrix} \tag{2.11}$$

Therefore,

$$z := Gw = F_u(M, \Theta)w, \text{ where } \Theta = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 I_2 \end{bmatrix}, \tag{2.12}$$

where I_n represents an $n \times n$ identity matrix.

This technique will be used in Chapter IV to rewrite several polynomial input/output expressions as LFT's. These individual LFT's are interconnected and will then be gathered into one combined LFT perturbed by a structured uncertainty block. As we will see, this enables us to model many time-varying systems, including the aircraft under study.

2.2 Linear Matrix Inequalities

A matrix inequality is any constraint of the form

$$A(x) < 0 \tag{2.13}$$

where

- $x = (x_1, \cdots, x_N)$ is a vector of free design variables, and
- $A(x)$ is a *symmetric* matrix.

Note that "< 0" indicates that A must be negative definite, and thus its eigenvalues are all negative. Similarly, an expression such as $X \geq 0$ would indicate that X is positive semidefinite so that all its eigenvalues are positive or equal to zero. This convention will be used throughout this thesis. Note also that the form of the inequality in (2.13) is really quite general, since equations can be manipulated and, if required, augmented with dummy variables to end up in this form.

Linear matrix inequalities (LMI's) are a special class of matrix inequalities where $A(x)$ depends affinely on the design variables x. That is

$$A(x) := A_0 + x_1 A_1 + \cdots + x_N A_N < 0 \tag{2.14}$$

where $A_0, A_1, \ldots, A_N$ are all given symmetric matrices.

A system of LMI's is then a set of M matrix constraints given by

$$A_i(x) < 0, \text{ where } i = 1, \ldots, M \tag{2.15}$$

This system can be treated as a single LMI by replacing it with

$$A(x) = \text{diag}(A_1(x), \ldots, A_M(x)) < 0 \tag{2.16}$$

LMI's are numerically appealing since finding a solution x to satisfy the above equation is a convex feasibility problem for which efficient interior-point algorithms are now available [NN94],[BEFB94],[GNLC92]. As explained in these references, such algorithms first define an objective function which is finite within the feasible set. In this way, the feasibility problem is transformed into a convex optimization problem. The solution chosen, if any exists, then corresponds to the minimum value of the objective function, defined as the analytic center of the LMI [BEFB94].

A basic example of an LMI is the Lyapunov inequality

$$A^T X + XA + Q < 0, \text{ where } X = X^T \in \Re^{n \times n} \tag{2.17}$$

In such cases, the variables x consist of the $\frac{n(n+1)}{2}$ free entries of the unknown symmetric matrix X.

Many theorems and expressions using LMI's are derived by making use of the *Schur complement* to transform equations into negative definite 2×2 block matrices which are then LMI's. Specifically, consider the block matrix

$$L = \begin{bmatrix} P & M \\ M^T & S \end{bmatrix}. \tag{2.18}$$

Then $L < 0$ if and only if

$$\begin{cases} S < 0 \\ P - MS^{-1}M^T < 0 \end{cases} \tag{2.19}$$

In this case, $P - MS^{-1}M^T$ is referred to as the Schur complement of S. Note that P, S and M can themselves be matrices or matrix expressions. For example, consider the Riccati inequality

$$A^TX + XA + XBB^TX + Q < 0, \text{ where } X = X^T \in \Re^{n \times n} \tag{2.20}$$

Using Schur complements, this can be rewritten as the LMI

$$\begin{bmatrix} A^TX + XA + Q & XB \\ B^TX & -I \end{bmatrix} < 0 \tag{2.21}$$

where X is again an unknown symmetric matrix consisting of the variables x.

In fact, since $S < 0$ in (2.19), it can also represent a matrix formed by use of the Schur complement. An important example of this is the Bounded Real Lemma (BRL) for continuous-time systems, which is used to turn H_∞ constraints into LMI's.

Continuous-Time BRL: Consider a transfer function $T(s) \equiv D + C(sI - A)^{-1}B$. The following statements are then equivalent:

1. A is stable and $\|T(s)\|_\infty < \gamma$.
2. There exists a solution $X > 0$ to the following inequalities

$$\begin{cases} \bar{\sigma}(D) \equiv \|D\|_\infty < \gamma \\ A^TX + XA + \frac{1}{\gamma}C^TC + \gamma(XB + \frac{1}{\gamma}C^TD)(\gamma^2 I - D^TD)^{-1}(B^TX + \frac{1}{\gamma}D^TC) < 0 \end{cases} \tag{2.22}$$

3. There exists a solution $X > 0$ to the LMI

$$\begin{bmatrix} A^TX + XA & XB & C^T \\ B^TX & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} < 0. \tag{2.23}$$

One can clearly appreciate the power of LMI's here, noting that the two conditions of item 2, including the rather complex Algebraic Riccati Inequality (ARI), can be expressed, after some manipulations, as the LMI of item 3; also, the LMI version can be solved efficiently since it is a convex programming problem.

This BRL forms the basis of the proofs for LMI-based H_∞ control, and thus for the theorems and results to be presented in Chapter III.

In general, there are three types of LMI problems:

1. determining a feasible solution x (or X) for some LMI constraints given by $L(x) < 0$.
2. minimization of a convex objective given some LMI constraints; that is,

$$\text{minimize } c^Tx \text{ subject to } L(x) < 0. \tag{2.24}$$

3. solving the generalized eigenvalue minimization problem (GEVP) given some LMI constraints; that is

$$\text{minimize } \lambda \text{ subject to } \begin{cases} A(x) < \lambda B(x) \\ B(x) > 0 \\ C(x) < 0. \end{cases} \tag{2.25}$$

The first two types of problems are convex, while the GEVP problem is quasi-convex. The feasibility and GEVP problems will be referred to in Chapter III and used to

obtain the gain-scheduled controller. All of these three types of problems can now be addressed using the new LMI Control Toolbox in MATLAB [GNLC92],[MAT].

As a final example of the power of LMI's, recall that in the standard Riccati-based [DGKF89] method of computing H_∞ controllers, the Algebraic Riccati Equation (ARE)

$$A^T X + XA + X(\gamma^{-2} B_1 B_1^T - B_2 B_2^T)X + C_1^T C_1 = 0 \tag{2.26}$$

is used to find a stabilizing solution X_∞, which is then used to determine the "central" H_∞ controller (note that $D = 0$ is assumed here for simplicity). All suitable controllers can then be parametrized via LFT's built around the central controller using a free parameter Q [DGKF89]. A problem with this approach is that the plant must be regular, and the resulting controllers are often of high order, especially when using Q-parametrization, since there is no clear connection between Q and the controller properties.

In the LMI-based approach, the ARE's are replaced by ARI's, and the solution set of all these inequalities is used to parametrize the H_∞ controllers. As a result, the plant need not be regular, and the resulting controller orders and properties can be shown to depend on two matrix parameters R and S [GA94]. In addition, the LMI constraints can be altered or other LMI constraints can be added to further define or modify the problem. Since a set of LMI's remains an LMI, this is an elegant way to ensure the problem remains convex and optimizable. This approach is taken in Chapter III to formulate the problem in order to be able to solve for the gain-scheduled controller.

2.3 The Structured Singular Value

Unstructured uncertainty is generally modelled as some norm-bounded $\Delta \in H_\infty$ perturbing a nominal plant or system, as shown in Figure 2.6.

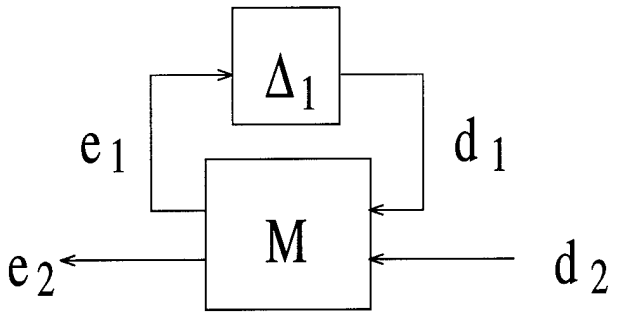


Figure 2.6 Modelled uncertainty Δ in a system

The problem, then, becomes one of either finding the largest Δ the system can handle or of finding the best controller K to maximize the allowable Δ and, in so doing, maximize the stability margins. This is the *robust stability* problem, and is addressed by using the Small Gain Theorem to derive tests for robust stability.

However, this does not directly address *robust performance*, in which we would like to maintain both robust stability and nominal performance throughout the envelope modelled by the uncertainty Δ. In other words, we want the plant not only to be stable to some uncertainties, but also to perform to a desired level even when perturbed. There are several possible ways to do this, and these all involve maximizing both a test for robust stability and another test for performance, and performing some tradeoffs for the optimal mix. Combining both subproblems as one test implies a structure; for strictly H_∞ problems, this has led to the development of the Structured Singular Value, called SSV or μ, in order to optimize this combined test.

Looking again at Figure 2.6, we see that the transfer function from d_2 to e_2 is given by

$$T_{e_2 d_2} \equiv F_u(M, \Delta_1) \tag{2.27}$$

where M is as defined in (2.1). This, of course, assumes that the LFT is well-posed and that Δ_1 has a block structure compatible with M_{11}. In this case, M_{22} is the nominal map and the rest of M reflects how the norm-bounded perturbation Δ_1

affects M_{22}. Now consider a structure

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \tag{2.28}$$

As mentioned in the last section, such an uncertainty block is considered a structured uncertainty since it results from combining all the uncertainties at different points of a system into one block. In this case, since we are interested in robust performance, Δ_2 could represent a fictitious uncertainty associated with the performance channel; that is, from e_2 to d_2. The resulting interconnection is shown in Figure 2.7. In fact, if we have multiple uncertainties in our system (or even multiple performance objectives), our uncertainty block structure could be composed of several more Δ's linked to the appropriate input-output channels.

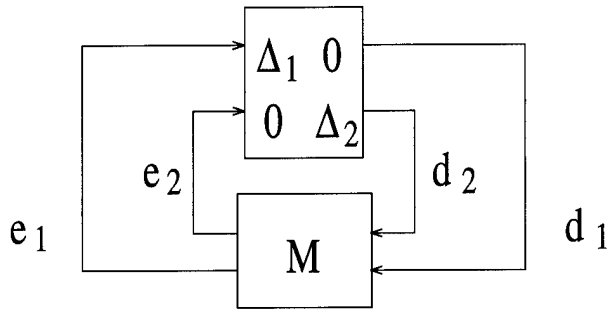


Figure 2.7 Robust Performance framework

In general, there are two types of uncertainty blocks: repeated scalar blocks, and full blocks. For S repeated scalar blocks δ_i, each of dimension $r_S \times r_S$, and F full blocks Δ_j, we can define a general structured uncertainty $\mathbf{\Delta} \in C^{n \times n}$ as

$$\mathbf{\Delta} = \{diag[\delta_1 I_{r_1}, \ldots, \delta_S I_{r_S}, \Delta_1, \ldots, \Delta_F] : \delta_i \in C, \Delta_j \in C^{m_j \times m_j}\} \tag{2.29}$$

For dimensional consistency, if $M \in C^{n \times n}$, then $\sum_{i=1}^{S} r_i + \sum_{j=1}^{F} m_j = n$. The norm-bounded subset of $\mathbf{\Delta}$ can be defined as

$$B_{\mathbf{\Delta}} = \{\Delta \in \mathbf{\Delta} : \bar{\sigma}(\Delta) \leq 1\} \tag{2.30}$$

where $\bar{\sigma}(\Delta)$ is defined as the maximum singular value of Δ.

The structured singular value μ is a function created to get a measure of the effect of the smallest perturbation $\Delta \in \mathbf{\Delta}$ for which a system becomes singular and therefore unstable. It is defined as

$$\mu_{\mathbf{\Delta}}(M) := \frac{1}{\min \{\bar{\sigma}(\Delta) : \Delta \in \mathbf{\Delta}, \det(I - M\Delta) = 0\}} \tag{2.31}$$

unless $I - M\Delta$ is always nonsingular, in which case $\mu_{\Delta} := 0$.

For example, in our earlier problem of (2.27)–(2.28) where Δ is a structured uncertainty comprised of an actual uncertainty and a fictitious performance "uncertainty", we can observe that minimizing μ will maximize the size of the allowable Δ, which is exactly what we would like to achieve! In fact, this leads us to the main theorem for the use of μ in linear system robustness analysis:

Main Loop Theorem: The following equations are equivalent:

$$\mu_{\mathbf{\Delta}}(M) < 1 \Longleftrightarrow \begin{cases} \mu_{\Delta_1}(M_{11}) < 1, \text{ and} \\ \max\limits_{\Delta_1 \in B_{\mathbf{\Delta}_2}} \mu_{\Delta_2}(F_l(M, \Delta_1)) < 1 \end{cases} \tag{2.32}$$

Proof: See [ZDG96].

The first equation on the right-hand side is μ of M_{11} with respect to the uncertainty Δ_1. This is basically equivalent to the Small Gain Theorem, and guarantees well-posedness and robust stability if less than 1. The second equation on the right-hand side is μ of $F_l(M, \Delta_1)$ with respect to the fictitious uncertainty Δ_2. This is

basically a measure of the inverse of the performance of the perturbed system. The point is that if μ_Δ is made less than 1, we can then guarantee robust stability and some level of performance; in other words, robust performance.

Expression (2.31) for μ is very hard to evaluate in practice; however, for the extreme case where $\Delta \in C^{n\times n}$, then

$$\mu_\Delta(M) = \bar{\sigma}(M) \tag{2.33}$$

This, then, is an overbound to μ for less extreme cases where Δ is a mixture of full blocks and repeated scalar blocks; i.e.,

$$\mu_\Delta(M) \leq \bar{\sigma}(M) \tag{2.34}$$

In order to close the gap in the overbound, positive definite similarity transformations on M can be found that affect the value of $\bar{\sigma}$ but do not affect the value of μ. This is given by the following theorem

Theorem 2.3.1: For all $D \in \mathbf{D}$ and $\Delta \in \mathbf{\Delta}$, where $D\Delta = \Delta D$ and

$$\boldsymbol{D} = \{diag[D_1, \ldots, D_S, d_1 I_{m_1}, \ldots, d_F I m_F] : D_i \in C^{r_i \times r_i}, D_i = D_i^* > 0, d_j \in \Re, d_j > 0\} \tag{2.35}$$

then

$$\mu_\Delta(M) = \mu_\Delta(DMD^{-1}) \tag{2.36}$$

Proof: See [ZDG96].

It is important to note that the transformations or "D-scales" are, like the corresponding uncertainties, of two types. For repeated scalar uncertainties the D-scales are full blocks; whereas for full block uncertainties the D-scales are scalars. Thus, if we would want to normalize the scaling for an uncertainty Δ_a by the scaling for an uncertainty Δ_b, it becomes necessary to treat the uncertainty Δ_b as a full

block in order for its corresponding D-scales to be scalar. This is done in Chapter III in the D-K-D scheme presented there, even though in that case the uncertainties were known to be repeated scalar blocks. This unfortunately introduces some conservativeness, since we are then allowing for uncertainties that are known not to exist.

Returning to our discussion, combining the results of Theorem 2.3.1 and (2.34), it follows that

$$\mu_{\mathbf{\Delta}}(M) \leq \bar{\mu}_{\mathbf{\Delta}}(M) \equiv \inf_{D \in \boldsymbol{D}} \bar{\sigma}(DMD^{-1}) \tag{2.37}$$

At this point, it should be pointed out that μ is not necessarily constant. In the previous development, we assumed that both M and Δ were constant, and for such cases, μ would be constant; however, as is often the case, M or Δ (or both) is usually a system matrix varying with frequency. For such cases, μ will be a frequency dependent function. The previous development remains valid, except that in order to determine the maximum value for μ, we must then evaluate it over all frequencies for its supremum. To handle these more general cases, (2.37) must then be modified slightly as follows:

$$\sup_{\omega} \mu_{\mathbf{\Delta}}(M) \leq \sup_{\omega} \bar{\mu}_{\mathbf{\Delta}}(M) \equiv \sup_{\omega} \inf_{D \in \boldsymbol{D}} \bar{\sigma}(DMD^{-1}) \tag{2.38}$$

Since we don't really know how to calculate μ, the upper bound $\bar{\mu}$ is what is used to approximate it. Determining the D-scales is a convex programming problem, so these can be solved for efficiently [PD93],[BDG+91]. The value of μ can then be approximated over all frequencies to determine its frequency response and its maximum. There is also a lower bound, but since it is nearly as difficult to evaluate as μ, it is not commonly used. In fact, for certain uncertainty block structures, $\bar{\mu}$ has been shown to be equal to μ [PD93]. Most importantly, if the uncertainty is composed of 1 to 3 full blocks (and no repeated scalars) then the upper bound is equal to μ. It has also been shown to remain relatively tight for 4 or more full

blocks. Thus, using a program to solve for the D-scales such as the μ-Toolbox in MATLAB [BDG+91],[MAT], we can approximate both the frequency response and the maximum value of μ for a system. This is called μ-analysis, and it can be used to guarantee that a general system exhibits robust performance by demonstrating that the peak value of $\bar{\mu} < 1$.

However, a more common problem is that of finding a stabilizing controller that would maximize robust performance and thus minimize μ. This is called μ-synthesis. For a general system like the one shown in Figure 2.8, we would therefore like to find a stabilizing K such that the peak value of $\mu < 1$. Noting that the transfer function from the inputs d to the outputs e is $T_{ed} = F_l(P, K)$, we can then express the μ-synthesis problem as

$$\begin{aligned} \inf_{K_{\text{stabilizing}}} \sup_{\omega} \mu_{\boldsymbol{\Delta}}[T_{ed}(s)] &\leq \inf_{K_{\text{stabilizing}}} \sup_{\omega} \inf_{D \in \boldsymbol{D}} \bar{\sigma}(DT_{ed}D^{-1}) \\ &= \inf_{K_{\text{stabilizing}}} \inf_{D \in \boldsymbol{D}} \sup_{\omega} \bar{\sigma}(DT_{ed}D^{-1}) \\ &= \inf_{K_{\text{stabilizing}}} \inf_{D \in \boldsymbol{D}} \|DT_{ed}D^{-1}\|_{\infty} \end{aligned} \tag{2.39}$$

Therefore, a tight approximation to our desired μ-synthesis robust performance problem is given by

$$\inf_{K_{\text{stabilizing}}} \inf_{D \in \boldsymbol{D}} \|DT_{ed}D^{-1}\|_{\infty} \tag{2.40}$$

Given rational functions for the D-scales, this expression is now a standard, albeit scaled, H_∞ problem which can be solved using Riccati-based or LMI-based methods for the optimal controller. This scaled problem is illustrated in Figure 2.9.

However, solving for both the D-scales and K is not a jointly convex problem. The most popular μ-synthesis approach is the so-called D-K iteration method, summarized in the following steps:

1. Set $D = I$ initially.

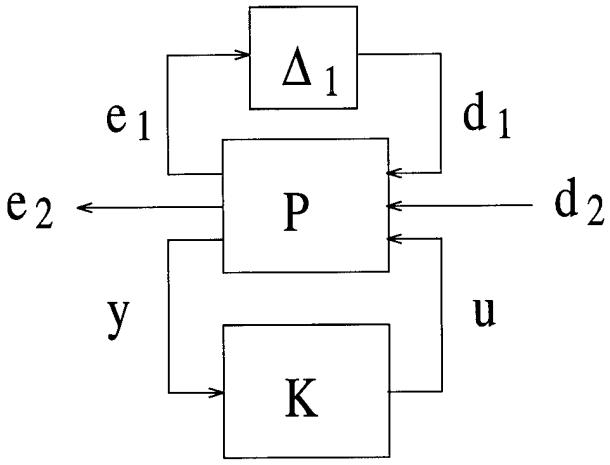


Figure 2.8 General system framework

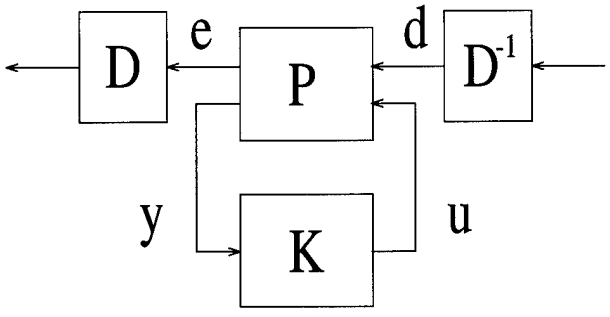


Figure 2.9 μ-synthesis scaled H_∞ problem

2. Solve the H_∞ optimization problem $\|DT_{ed}D^{-1}\|_\infty$ for the minimizing, stabilizing controller K. Note that $T_{ed} = F_l(P, K)$.

3. Use this K and determine the D-scales to minimize $\bar{\mu}$ given by

$$\inf_{D \in \boldsymbol{D}} \|DT_{ed}D^{-1}\|_\infty. \tag{2.41}$$

4. Fit the D-scales to rational transfer functions to closely approximate $\bar{\mu}$.

5. Repeat steps 2 to 4 until $\bar{\mu}$ no longer decreases.

Note that this procedure is not guaranteed to converge, but since it works well in practice it is implemented in the μ-Toolbox in MATLAB. Also, the discussion of μ dealt specifically with complex uncertainties. For real or mixed uncertainties, there are similar theoretical results [BDG+91]; unfortunately, while μ-analysis has been implemented for both complex and real uncertainties, μ-synthesis has not. This is unfortunate, since it forces practitioners to assume complex uncertainties for μ-synthesis even when some or all of these are known to be real, resulting in a much more conservative design than would otherwise be necessary. This, for example, is the case for the aircraft controller designed in Chapter IV.

III. Gain Scheduling Theory

This chapter presents the background theory of gain scheduling for linear parameter varying systems as developed principally by Packard [Pac94], and Apkarian and Gahinet [AG95]. Most of the following development is taken with only notational and other minor modifications from [AG95] and [Pac94]. While the following development is given only for continuous-time, the cited references also detail the discrete-time approach.

3.1 Linear Parameter-Varying Systems

Much of H_∞ theory to this point has involved the synthesis of controllers for linear time-invariant (LTI) plants. This applies well for a host of problems where the plants remain relatively constant over time. However, in many applications, the plant varies significantly over time. Such linear time-varying (LTV) plants can be expressed with state-space equations of the form

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{aligned} \tag{3.1}$$

where not only the states $x(t)$ and the inputs $u(t)$ vary with time, but also the state-space matrices $A(t)$, $B(t)$, $C(t)$, and $D(t)$. Further, a large class of these LTV plants can be expressed as systems of the form

$$\begin{aligned} \dot{x}(t) &= A(\theta(t))x(t) + B(\theta(t))u(t) \\ y(t) &= C(\theta(t))x(t) + D(\theta(t))u(t) \end{aligned} \tag{3.2}$$

where $\theta(t)$ is a vector of time-varying parameters and $A(\theta)$, $B(\theta)$, $C(\theta)$, $D(\theta)$ are now known fixed functions of $\theta(t)$. The difference lies in the fact that, while the state-space matrices of (3.1) are known functions of time, the state-space matrices of

(3.2) are known functions of $\theta(t)$. $\theta(t)$ must be measured real-time and is not known *a priori*, although its range of parameter values is often known.

Such plants are now commonly referred to as linear parameter-varying (LPV) systems, and can describe many types of systems such as aircraft, robotics, missiles, etc. In the case of an aircraft, its stability derivatives and thus its state-space model depends on parameters such as Mach number, altitude, angle of attack, or dynamic pressure; these parameters in turn change over time.

In many cases, $A(\theta)$, $B(\theta)$, $C(\theta)$, $D(\theta)$ can be expressed as a linear fractional function of $\theta(t)$. In other words, we can express $A(\theta)$, $B(\theta)$, $C(\theta)$, and $D(\theta)$ as an LFT relating $\theta(t)$ to an LTI plant P consisting of "nominal" values A, B, C, and D. Thus, using standard state-space notation,

$$P(s) = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \tag{3.3}$$

The time-varying parameter vector can be represented as

$$\theta = (\theta_1, \cdots, \theta_K) \in \Re^K \tag{3.4}$$

Referring to Figure 3.1, Θ is a block diagonal time-varying operator specifying how θ enters the plant dynamics. Specifically,

$$\Theta = \text{blockdiag}(\theta_1 I_{r_1}, \cdots, \theta_K I_{r_K}) \tag{3.5}$$

where $r_i > 1$ whenever the parameter θ_i is repeated [Doy85]. The set of all Θ is given by

$$\Delta := \{\Theta : \theta_i(t) \in \Re\} \tag{3.6}$$

Note that Δ is traditionally referred to as an uncertainty structure. In fact, if we also wanted to model some uncertainty in the plant, the structure of Δ could be

modified to account for this. This will be expanded upon later. Note also that $\theta \in \Re$ since the actual plant variations are real, not complex; however, since we will eventually be applying the Small Gain Theorem to solve for our controller, the solution will, in fact, allow for complex variations in the plant. This results in a much more conservative solution and less performance than could likely be achieved.

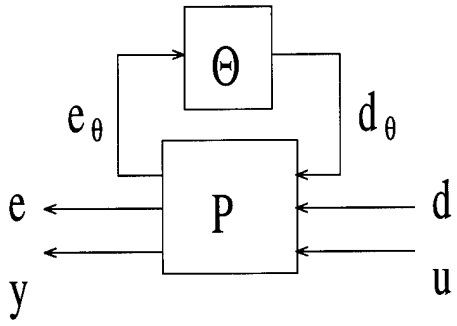


Figure 3.1 LPV Plant

The feedback equations for the plant LFT can now be written as

$$\begin{bmatrix} e_\theta \\ e \\ y \end{bmatrix} = \overbrace{\begin{bmatrix} P_{\theta\theta} & P_{\theta d} & P_{\theta u} \\ P_{e\theta} & P_{ed} & P_{eu} \\ P_{y\theta} & P_{yd} & P_{yu} \end{bmatrix}}^{P(s)} \begin{bmatrix} d_\theta \\ d \\ u \end{bmatrix} \tag{3.7}$$

$$d_\theta = \Theta e_\theta \tag{3.8}$$

Thus, for a given time t, the LPV plant defines an LTI plant whose transfer function is given by the upper LFT

$$\begin{bmatrix} e \\ y \end{bmatrix} = F_u(P, \Theta) \begin{bmatrix} d \\ u \end{bmatrix} \tag{3.9}$$

or, explicitly,

$$\begin{bmatrix} e \\ y \end{bmatrix} = \left\{ \begin{bmatrix} P_{ed} & P_{eu} \\ P_{yd} & P_{yu} \end{bmatrix} + \begin{bmatrix} P_{e\theta} \\ P_{y\theta} \end{bmatrix} \Theta[I - P_{\theta\theta}\Theta]^{-1}[P_{\theta d}\ P_{\theta u}] \right\} \begin{bmatrix} d \\ u \end{bmatrix} \tag{3.10}$$

Using the LPV model of the plant, traditional techniques based on the Small Gain Theorem could now be used to find a single robust LTI controller for the varying plant [Doy85]. Unfortunately, if the plant undergoes large deviations, this will generally be very conservative and exhibit poor performance. In fact, a stabilizing controller may not even be feasible. A better approach is to design a parameter-dependent controller, which will then allow it to vary as the plant varies. This is illustrated in Figure 3.2.

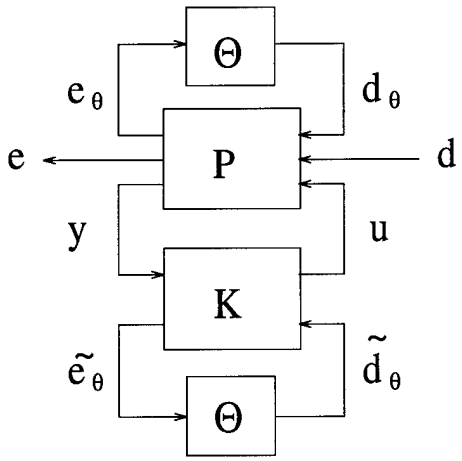


Figure 3.2 LPV control structure

The feedback equation for the gain-scheduled LPV controller is then

$$u = F_l(K, \Theta)y \tag{3.11}$$

where $F_l(K, \Theta)$ is the controller at a given time t, and K specifies the relationship between the changing parameters and the gain-scheduled controller. Since Θ is based on the known parameter variations of the plant, the only unknown is the LTI K given

by

$$K(s) = \begin{bmatrix} K_{uy} & K_{u\theta} \\ K_{\theta y} & K_{\theta\theta} \end{bmatrix} \tag{3.12}$$

$$\widetilde{d_\theta} = \Theta \widetilde{e_\theta} \tag{3.13}$$

Thus, once K has been found, the LPV gain-scheduled controller could then be physically implemented. The controller would then be continually updated with in-flight measurements of θ, and (3.11) gives the rule to produce the updated control input u.

Finally, note that the closed-loop transfer function from the disturbance d to the controlled output e is given by

$$\frac{e}{d} \equiv T_{ed}(P, K, \Theta) = F_l(F_u(P, \Theta), F_l(K, \Theta)). \tag{3.14}$$

Note also that, without any Θ, we recover the transfer function associated with LTI systems; namely $T_{ed} = F_l(P, K)$.

3.2 H_∞ Control of LPV Systems

Given an LTI plant $P(s)$, the usual H_∞ problem requires finding an internally stabilizing LTI controller $K(s)$ such that

$$\|F_l(P, K)\|_\infty < \gamma \tag{3.15}$$

where $\gamma > 0$ is some desired performance level. However, in the gain-scheduled version of this problem, both the plant and the controller are LPV vice LTI. The objective becomes one of robust performance; in other words, to guarantee some closed-loop performance γ for all admissible parameters θ. The H_∞ control problem is now required to find a controller $K(s)$ such that the LPV controller $F_l(K, \Theta)$ meets the following conditions

1. the closed-loop system must be internally stable for all admissible parameters θ such that

$$\gamma^2 \Theta^T \Theta \leq 1 \tag{3.16}$$

2. the closed-loop system satisfies

$$\max_{\|\Theta\|_\infty \leq 1/\gamma} \|T_{ed}(P, K, \Theta)\|_\infty < \gamma \tag{3.17}$$

Equation (3.16) comes from the Small Gain Theorem and restricts the parameter range of Θ to a ball of radius $1/\gamma$. This implies no loss of generality since the parameters θ can simply be scaled to comply with this requirement. For example, in the implementation described in Chapter IV, the parameter variations have been normalized such that $\gamma < 1$ would imply that the above two conditions have been met. Following the convention of [AG95], solutions to the aforementioned H_∞ problem for LPV systems will be called γ-suboptimal gain-scheduled controllers.

As shown in Figure 3.2, the H_∞ gain-scheduling problem as stated has a time-varying parameter block Θ entering both the plant and the controller. In order to make use of the Small Gain Theorem, the two parameter blocks can be combined into one single uncertainty block, as illustrated in Figure 3.3, resulting in a new plant P_a.

The feedback equations for P_a can be written as

$$\begin{bmatrix} \widetilde{e_\theta} \\ \begin{bmatrix} e_\theta \\ e \\ y \end{bmatrix} \\ \tilde{y} \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 0 & I_r \\ 0 & P(s) & 0 \\ I_r & 0 & 0 \end{bmatrix}}^{P_a(s)} \begin{bmatrix} \widetilde{d_\theta} \\ \begin{bmatrix} d_\theta \\ d \\ u \end{bmatrix} \\ \tilde{u} \end{bmatrix} \tag{3.18}$$

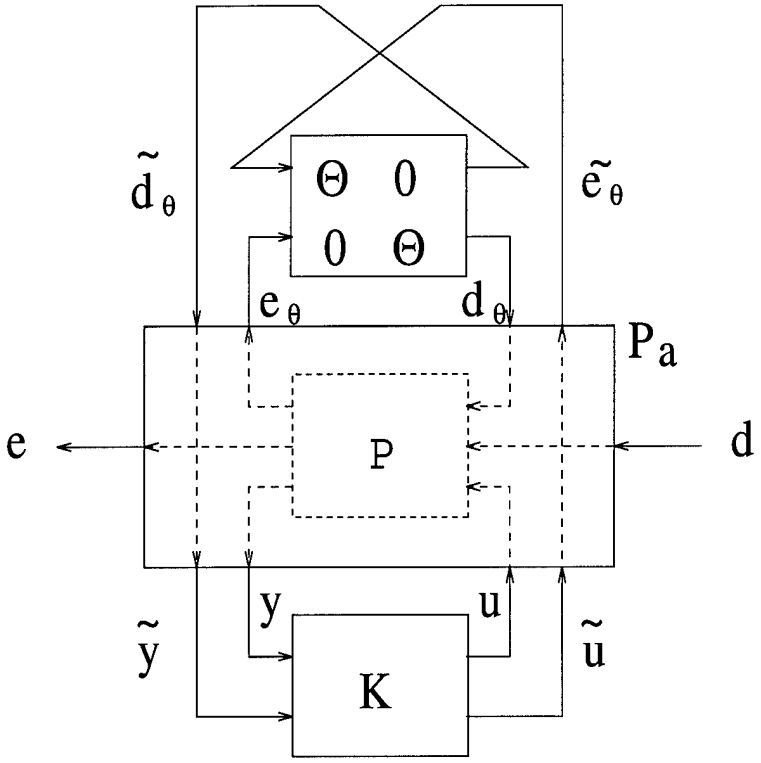


Figure 3.3 Modified LPV control structure

As a result, the closed-loop transfer function between the exogenous input d and the controlled output e is now

$$T_{ed}(P, K, \Theta) = F_u(F_l(P_a, K), \Delta \oplus \Delta), \text{where } \Delta \oplus \Delta = \begin{bmatrix} \Theta & 0 \\ 0 & \Theta \end{bmatrix} \tag{3.19}$$

In state-space, the LTI plant P can be written as

$$P(s) = \left[\begin{array}{c|ccc} A & B_\theta & B_d & B_u \\ \hline C_\theta & D_{\theta\theta} & D_{\theta d} & D_{\theta u} \\ C_e & D_{e\theta} & D_{ed} & D_{eu} \\ C_y & D_{y\theta} & D_{yd} & D_{yu} \end{array}\right] \tag{3.20}$$

Thus, the augmented LTI plant P_a is

$$P_a(s) = \left[\begin{array}{c|ccccc} A & 0 & B_\theta & B_d & B_u & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I_r \\ C_\theta & 0 & D_{\theta\theta} & D_{\theta d} & D_{\theta u} & 0 \\ C_e & 0 & D_{e\theta} & D_{ed} & D_{eu} & 0 \\ C_y & 0 & D_{y\theta} & D_{yd} & D_{yu} & 0 \\ 0 & I_r & 0 & 0 & 0 & 0 \end{array}\right] \tag{3.21}$$

Similarly,

$$K(s) = \left[\begin{array}{c|cc} A_K & B_{Ky} & B_{K\theta} \\ \hline C_{Ku} & D_{Kuy} & D_{Ku\theta} \\ C_{K\theta} & D_{K\theta y} & D_{K\theta\theta} \end{array}\right] \tag{3.22}$$

For a problem with n states, m_2 control inputs u, p_2 measured outputs y, p_1 exogenous inputs d and controlled outputs e, time-varying operator Θ of dimension r, and a controller K of order k, the problem dimensions are

$$A \in \Re^{n \times n}, D_{\theta\theta} \in \Re^{r \times r}, D_{ed} \in \Re^{p_1 \times p_1}, D_{yu} \in \Re^{p_2 \times m_2}, A_K \in \Re^{k \times k}. \tag{3.23}$$

Note that Θ and the transfer function from the disturbances d to the controlled outputs e have been considered square. This simplifies the notation and can be easily met by augmenting the problem with rows and/or columns of zeros.

There are also three assumptions required to solve for the gain-scheduled controller:

1. (A, B_u, C_y) must be stabilizable and detectable,
2. $D_{yu} = 0$, and
3. $D_{y\theta} = 0$ or $D_{\theta u} = 0$.

The first assumption is standard, and is necessary and sufficient to guarantee the existence of a stabilizing controller. The second assumption is not required, but greatly simplifies the calculations in the rest of this chapter. Appendix A of [All95] outlines the shifting technique that can be used to lift this restriction. The third assumption is sufficient but not necessary to guarantee that the LPV controller is causal and well-posed [AG95]. Many problems meet this condition since the y measurement equation is often independent of the parameters θ, or θ is independent of the control input u. This assumption and the previous ones, for example, all hold for the problem described in Chapter IV. Note that the last assumption can also be removed by imposing a well-posedness constraint on the problem [AG95].

We now have a more classical-looking robust performance problem, with a norm-bounded repeated uncertainty block $\Delta \oplus \Delta$, an LTI controller K, and a new LTI plant P_a consisting of the original plant P augmented with the interconnections $\tilde{u}$ and $\tilde{y}$ between K and Θ. It is precisely because of these extra interconnections that this problem differs from the more classical H_∞ control problem, since they ensure that the controller changes with exactly the same parameter variations as the plant. The only difference lies in *how much* the controller changes in response to the variations; this, as we will see shortly, will be optimized using LMI's to find the optimal scalings or weights which will dictate the magnitude of the response to the

parameter variations. Those optimized scales in turn will then be used to solve for the γ-suboptimal controller K.

Now that the problem has been reformulated to have a single uncertainty block, the Small Gain Theorem can be used to determine a sufficient condition for robust performance or, equivalently, for the existence of a gain-scheduled controller [Doy85]. Specifically, this leads to the following theorem from [AG95].

Theorem 3.2.1: Consider the set of all positive definite similarity scalings commuting with Δ

$$L_\Delta = \{L > 0 : L\Theta = \Theta L, \forall \Theta \in \Delta\} \subset \Re^{r \times r}, \text{where } r = \sum_{i=1}^{K} r_i. \tag{3.24}$$

Given L_Δ, the set of scalings commuting with $\Delta \oplus \Delta$ is then

$$L_{\Delta\oplus\Delta} \equiv \left\{ L = \begin{bmatrix} L_1 & L_2 \\ L_2^T & L_3 \end{bmatrix} > 0 : L_1, L_3 \in L_\Delta, L_2\Theta = \Theta L_2, \forall \Theta \in \Delta \right\}. \tag{3.25}$$

If there exists a scaling matrix $L \in L_{\Delta\oplus\Delta}$ and an LTI control structure K such that the nominal closed-loop system $F_l(K, \Theta)$ is internally stable and satisfies

$$\left\| \begin{bmatrix} L^{1/2} & 0 \\ 0 & I \end{bmatrix} F_l(P_a, K) \begin{bmatrix} L^{-1/2} & 0 \\ 0 & I \end{bmatrix} \right\|_\infty < \gamma \tag{3.26}$$

then $F_l(K, \Theta)$ is a γ-suboptimal gain-scheduled H_∞ controller.

Proof: See [AG95].

Note that, in (3.26), $L \in \Re^{2r \times 2r}$ since it is the scaling matrix for $\Delta \oplus \Delta$; and $I \in \Re^{p_1 \times p_1}$ because the controlled outputs e and the exogenous inputs d are not scaled. Their weights, if any, must be incorporated into P_a. The problem is now one of calculating L and K such that (3.26) holds.

3.3 Solving the General Scaled H_∞ Problem

The Bounded Real Lemma (BRL) can be used to transform scaled H_∞ optimization problems into a set of LMI constraints. For convenience, it is stated below.

Continuous-Time Scaled BRL: Consider an arbitrary uncertainty structure Δ, and the associated scaling set L_Δ as defined in (3.24). Also consider a square transfer function $T(s) \equiv C(sI - A)^{-1}B + D$. The following statements are then equivalent:

1. A is stable and there exists $L \in L_\Delta$ such that $\|L^{1/2}T(s)L^{-1/2}\|_\infty < \gamma$.
2. There exists solutions $X > 0$ and $L \in L_\Delta$ to the matrix inequality

$$\begin{bmatrix} A^TX + XA & XB & C^T \\ B^TX & -\gamma L & D^T \\ C & D & -\gamma L^{-1} \end{bmatrix} < 0. \tag{3.27}$$

Proof: See [GA94].

Now consider a proper LTI plant $G(s)$ where

$$\begin{bmatrix} e \\ y \end{bmatrix} = \begin{bmatrix} G_{ed} & G_{eu} \\ G_{yd} & G_{yu} \end{bmatrix} \begin{bmatrix} d \\ u \end{bmatrix} \tag{3.28}$$

Given a desired objective $\gamma > 0$, an arbitrary uncertainty structure Δ, and the associated scaling set L_Δ as defined in (3.24), the general scaled H_∞ problem is one of finding $L \in L_\Delta$ and an LTI controller K such that the closed-loop system is internally stable and

$$\|L^{1/2}F_l(G, K)L^{-1/2}\|_\infty < \gamma. \tag{3.29}$$

The Scaled BRL can then be used to derive the following theorem characterizing the solution of the general scaled H_∞ problem.

Theorem 3.3.1: Given

$$G(s) = \left[\begin{array}{c|cc} A & B_d & B_u \\ \hline C_e & D_{ed} & D_{eu} \\ C_y & D_{yd} & 0 \end{array}\right] \tag{3.30}$$

satisfying assumptions 1 and 2, the general scaled H_∞ problem is solvable if and only if there exist pairs of symmetric matrices $(R, S) \in \Re^{n\times n}$ and $(L, J) \in \Re^{p_1\times p_1}$ such that

$$N_R^T \begin{bmatrix} AR + RA^T & RC_e^T & B_d \\ C_eR & -\gamma J & D_{ed} \\ B_d^T & D_{ed}^T & -\gamma L \end{bmatrix} N_R < 0 \tag{3.31}$$

$$N_S^T \begin{bmatrix} A^TS + SA & SB_d & C_e^T \\ B_d^TS & -\gamma L & D_{ed}^T \\ C_e & D_{ed} & -\gamma J \end{bmatrix} N_S < 0 \tag{3.32}$$

$$\begin{bmatrix} R & I \\ I & S \end{bmatrix} \geq 0 \tag{3.33}$$

$$L \in L_\Delta, J \in L_\Delta, LJ = I \tag{3.34}$$

where N_R and N_S denote bases of the null spaces of $(B_u^T, D_{eu}^T, 0)$ and $(C_y, D_{yd}, 0)$, respectively.

Further, there exist suboptimal controllers of order k if and only if (3.31)–(3.34) hold for some (R, S, L, J) where R, S satisfy the rank constraint

$$\text{rank}(I - RS) \leq k. \tag{3.35}$$

Proof: See [AG95] and [GA94].

Matrix inequalities (3.31)–(3.33) are LMI's in R, S, L, J and constraints $L \in L_\Delta$ and $J \in L_\Delta$ are convex. As well, in the full order case, $k \geq n$ so that the rank

constraint (3.35) is trivially satisfied. However, the constraint $LJ = I$ is highly non-convex, so that solving the general scaled H_∞ problem remains a difficult problem.

It should also be pointed out that for the classical H_∞ control problem, L and J can be set to I without loss of generality. Equations (3.31)–(3.34) then reduce to a set of three convex LMI's which can be solved numerically. Compared to the standard Riccati-based method, this has the advantage of allowing imaginary axis invariant zeros and singular problems arising due to rank deficiencies in D_{eu} and D_{yd}. See [GA94] for more on this subject.

3.4 Solving the Gain-Scheduled H_∞ Problem

The gain-scheduled H_∞ problem can be viewed as a specific case of the general scaled H_∞ problem. This is evident, for example, from comparing (3.26) and (3.29). The main problem in solving the general scaled problem was the nonconvexity of the $LJ = I$ constraint. It turns out, however, that due to the particular structure of the LPV problem, this difficulty can be completely eliminated.

Theorem 3.4.1: Given the LPV system defined in Sections 3.1 and 3.2 and satisfying assumptions 1 and 2, the gain-scheduled H_∞ problem is solvable if and only if there exist pairs of symmetric matrices $(R, S) \in \Re^{n\times n}$ and $(L_3, J_3) \in \Re^{r\times r}$ such that

$$Z_R = N_R^T \begin{bmatrix} AR + RA^T & R\hat{C}_e^T & \hat{B}_d\hat{J}_3 \\ \hat{C}_e R & -\gamma\hat{J}_3 & \hat{D}_{ed}\hat{J}_3 \\ \hat{J}_3\hat{B}_d^T & \hat{J}_3\hat{D}_{ed}^T & -\gamma\hat{J}_3 \end{bmatrix} N_R < 0 \tag{3.36}$$

$$Z_S = N_S^T \begin{bmatrix} A^T S + SA & S\hat{B}_d & \hat{C}_e^T\hat{L}_3 \\ \hat{B}_d^T S & -\gamma\hat{L}_3 & \hat{D}_{ed}^T\hat{L}_3 \\ \hat{L}_3\hat{C}_e & \hat{L}_3\hat{D}_{ed} & -\gamma\hat{L}_3 \end{bmatrix} N_S < 0 \tag{3.37}$$

$$\begin{bmatrix} R & I \\ I & S \end{bmatrix} \geq 0 \tag{3.38}$$

$$L_3 \in L_\Delta, J_3 \in L_\Delta, \begin{bmatrix} L_3 & I \\ I & J_3 \end{bmatrix} \geq 0 \tag{3.39}$$

where

$$\hat{B}_d = [B_\theta \; B_d], \hat{C}_e = \begin{bmatrix} C_\theta \\ C_e \end{bmatrix}, \hat{D}_{ed} = \begin{bmatrix} D_{\theta\theta} & D_{\theta d} \\ D_{e\theta} & D_{ed} \end{bmatrix}, \hat{L}_3 = \begin{bmatrix} L_3 & 0 \\ 0 & I \end{bmatrix}, \hat{J}_3 = \begin{bmatrix} J_3 & 0 \\ 0 & I \end{bmatrix}. \tag{3.40}$$

and where N_R and N_S denote bases of the null spaces of $(B_u^T, D_{\theta u}^T, D_{eu}^T, 0)$ and $(C_y, D_{y\theta}, D_{yd}, 0)$, respectively.

Further, there exist suboptimal controllers of order k if and only if (3.36)–(3.39) hold for some (R, S, L_3, J_3) where R, S satisfy the rank constraint

$$\text{rank}(I - RS) \leq k. \tag{3.41}$$

Proof: See [AG95].

Equations (3.36)–(3.38) are LMI's in R, S, L_3, J_3 and are now all convex, such that testing these conditions for a feasible solution is a convex feasibility problem. As explained in Chapter II, software using interior-point LMI solvers [NN94] such as the LMI Control Toolbox in MATLAB [MAT],[GNLC92] can then be used to test the feasibility of the LMI conditions for some arbitrary γ values. Thus, we can iteratively solve for the minimum γ corresponding to the sub-optimal gain-scheduled controller.

Alternatively, γ can be solved for directly since minimizing γ subject to the feasibility of these LMI constraints is a convex optimization problem [GA94]. N_R

and N_S as defined in Theorem 3.4.1 can be written as

$$N_R = \begin{bmatrix} W_{R1} & 0 \\ W_{R2} & 0 \\ 0 & I \end{bmatrix}, N_S = \begin{bmatrix} W_{S1} & 0 \\ W_{S2} & 0 \\ 0 & I \end{bmatrix} \tag{3.42}$$

where $\begin{bmatrix} W_{R1} \\ W_{R2} \end{bmatrix}$ and $\begin{bmatrix} W_{S1} \\ W_{S2} \end{bmatrix}$ now denote the bases of the null spaces of $(B_u^T, D_{\theta u}^T, D_{eu}^T)$ and $(C_y, D_{y\theta}, D_{yd})$, respectively. Provided W_{R2} and W_{S2} have full column rank, this leads to the following Generalized Eigenvalue Problem (GEVP) formulation [SLBB96]

$$\text{minimize } \gamma \tag{3.43}$$

subject to

$$\begin{bmatrix} R & I \\ I & S \end{bmatrix} \geq 0 \tag{3.44}$$

$$L_3 \in L_\Delta, J_3 \in L_\Delta, \begin{bmatrix} L_3 & I \\ I & J_3 \end{bmatrix} \geq 0 \tag{3.45}$$

$$Z_R + \gamma \begin{bmatrix} W_{R2}^T \hat{J}_3 W_{R2} & 0 \\ 0 & \hat{J}_3 \end{bmatrix} < \gamma \begin{bmatrix} W_{R2}^T \hat{J}_3 W_{R2} & 0 \\ 0 & \hat{J}_3 \end{bmatrix} \tag{3.46}$$

$$Z_S + \gamma \begin{bmatrix} W_{S2}^T \hat{L}_3 W_{S2} & 0 \\ 0 & \hat{L}_3 \end{bmatrix} < \gamma \begin{bmatrix} W_{S2}^T \hat{L}_3 W_{S2} & 0 \\ 0 & \hat{L}_3 \end{bmatrix} \tag{3.47}$$

where $R, S, L_3, J_3, \hat{J}_3, \hat{L}_3, \hat{B}_d, \hat{C}_e, \hat{D}_{ed}, Z_R, Z_S$ are all as defined in Theorem 3.4.1. While not immediately apparent, the matrix sums on the left-hand sides of (3.46) and (3.47), when carried out, become independent of γ. Again, this problem can be solved using LMI solvers such as the LMI Control Toolbox in MATLAB. Often, W_{R2} and W_{S2} do not have full column rank; when this occurs, dummy variables are used to form a slightly modified GEVP [GNLC92]. This modified GEVP was

used to solve for the F-18 controller in Chapter IV. Note also that, depending on the problem, the number of dummy variables added can be extremely large. This significantly increases the computational effort required to solve the problem. In general, GEVP's require more computational effort than testing the LMI feasibility conditions, and adding dummy variables further increases the computer time and memory requirements.

Once values of R, S, L_3, J_3 have been found for some feasible performance γ (as explained in Chapter II, the chosen solution corresponds to the analytic center of the LMI), the LTI controller K must be determined. A systematic procedure to derive the state-space of K using a set of LMI constraints is given in [AG95]. This method offers the most flexibility since additional constraints on K can easily be incorporated in the form of more LMI's, and regularity of the plant is not required. Another method provides a series of analytical expressions to calculate the individual state-space matrices of K [Gah94]. A final method involves simply using the Riccati-based central H_∞ controller equations [DGKF89]. Regularity of the plant is required for this last approach; however, this can be addressed by incorporating small weights on all the measurements and control inputs of the system to ensure regularity. Since the aircraft design system developed in Chapter IV was regular, this last approach was used to determine the controllers in this thesis.

3.5 Solving the Gain-Scheduled H_∞ Problem for Uncertain LPV Systems

While the physical parameters θ are assumed to be measured, there may be an uncertainty associated with the measurements, or there may be other parameters which remain uncertain. This time-invariant unmeasured uncertainty can be handled by expanding the previous results to make use of classical robust control techniques such as μ-synthesis.

For such a problem, the basic LPV structure is shown in Figure 3.4, where Θ_t represents the block operator of time-varing parameters as defined in (3.5) and Θ_u

represents the constant uncertainty with the structure

$$\Theta_u = \text{blockdiag}(\delta_1 I_{r_1}, \ldots, \delta_S I_{r_S}, \Delta_1, \ldots, \Delta_F) \tag{3.48}$$

Δ_u denotes the corresponding structure of the uncertainty block

$$\Delta_u := \{\Theta_u : \delta_i \in C,\ \Delta_j \in C^{m_j \times m_j}\} \tag{3.49}$$

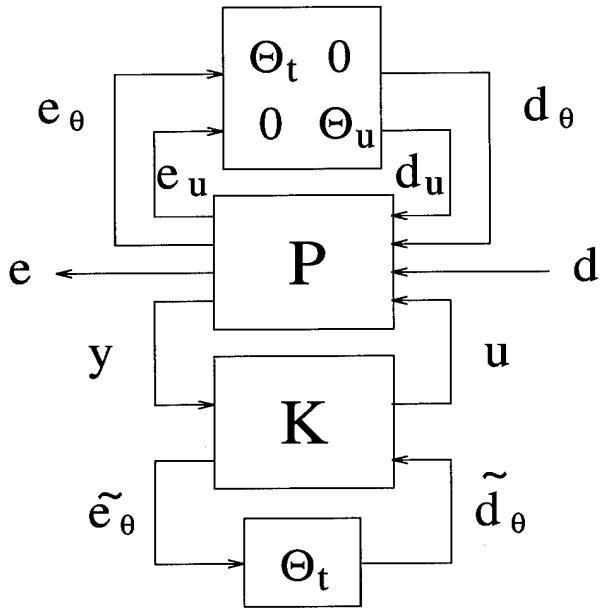


Figure 3.4 LPV control structure with added uncertainty

The total "uncertainty" structure then assumes the form

$$\begin{bmatrix} \Theta_t & 0 & 0 \\ 0 & \Theta_t & 0 \\ 0 & 0 & \Theta_u \end{bmatrix} \tag{3.50}$$

and the corresponding set of scalings is then

$$\left\{ \begin{bmatrix} L_t & 0 \\ 0 & L_u \end{bmatrix} > 0 : L_t \in L_{\Delta_t \oplus \Delta_t}, L_u \in L_{\Delta_u} \right\} \tag{3.51}$$

Figure 3.5 illustrates the transformed LPV control structure, taking into account the added time-invariant uncertainty block.

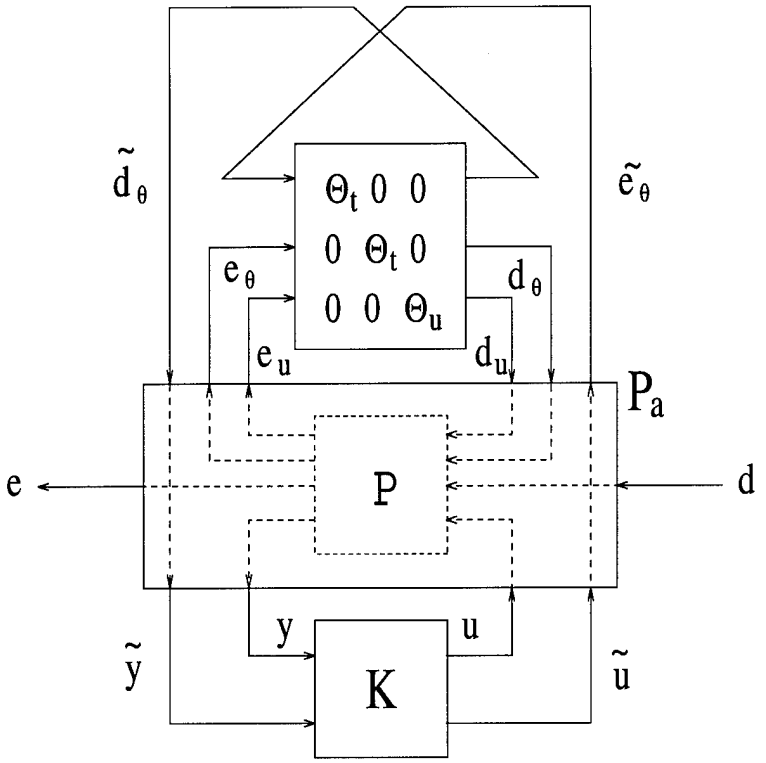


Figure 3.5 Modified LPV structure with uncertainty

Note that, in order to perform μ-synthesis, the performance channels must also be wrapped around to add a fictitious Θ_p block to the actual uncertainty block, so that all of the various objectives can be considered in order to obtain the optimum μ scales. This is shown in Figure 3.6. The corresponding set of scalings can then be written as

$$D = \left\{ \begin{bmatrix} L_t & 0 & 0 \\ 0 & L_u & 0 \\ 0 & 0 & L_p \end{bmatrix} > 0 : L_t \in L_{\Delta_t \oplus \Delta_t}, L_u \in L_{\Delta_u}, L_p \in L_{\Delta_p} \right\} \tag{3.52}$$

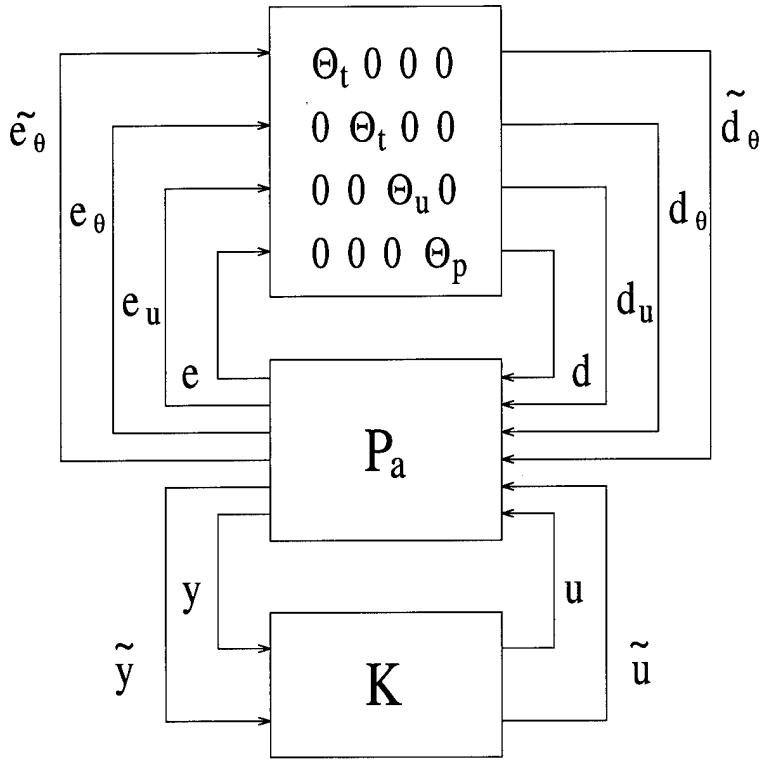


Figure 3.6 LPV/μ control structure

The D scalings defined in (3.52) could now be solved for in place of the L scalings of (3.26) to determine the γ-suboptimal gain-scheduled controller. Unfortunately, this augmented scaling resulting from the addition of the uncertainty block Θ_u and the performance block Θ_p ruins the convexity of the problem. However, the problem is convex as soon as the L_u and L_p blocks of the scaling matrix are fixed, since we then essentially restore the problem defined with no uncertainty. In addition, the performance γ can be optimized using μ-synthesis when L_t is fixed, by performing the usual D-K iterations [BDG+91]. This motivates a mixed LPV/μ ap-

proach to attempt to minimize μ and optimize γ, by alternating between solving the GEVP and performing D-K iterations. Specifically, the following "D-K-D" scheme is proposed [SLBB96]:

1. Set the D scale defined in (3.52) to I.
2. Calculate the scaled plant $\hat{P}_a(s)$ given by

$$\hat{P}_a = \left\{ \begin{bmatrix} D & 0 \\ 0 & I_{(r+p_2)} \end{bmatrix} P_a(s) \begin{bmatrix} D^{-1} & 0 \\ 0 & I_{(r+m_2)} \end{bmatrix} \right\} \tag{3.53}$$

3. Solve the LPV problem GEVP for L_t using the scaled plant $\hat{P}_a$.
4. Update the D scale with L_t, and recalculate $\hat{P}_a$.
5. Design an LTI controller K for $\hat{P}_a(s)$ using either the Riccati-based method (if the plant is regular) or the LMI-based methods outlined in Section 3.4.
6. Fix K and L_t, and find scalars m_{Θ_t} and matrices $M_{\Theta_u}, M_{\Theta_p}$ which minimize

$$\bar{\mu} = \left\| \begin{bmatrix} m_{\Theta_t} I & 0 & 0 \\ 0 & M_{\Theta_u} & 0 \\ 0 & 0 & M_{\Theta_p} \end{bmatrix} T_{ed} \begin{bmatrix} m_{\Theta_t} I & 0 & 0 \\ 0 & M_{\Theta_u} & 0 \\ 0 & 0 & M_{\Theta_p} \end{bmatrix}^{-1} \right\|_\infty , \tag{3.54}$$

   where $T_{ed} = DF_l(P_a, K)D^{-1}$. This is identical to performing the first D-K iteration. Performing additional iterations to further decrease $\bar{\mu}$ is not recommended, since this can quickly result in high order L_u, L_p scales, thereby requiring an inordinately large computational effort to solve the GEVP LMI's in follow-on iterations.
7. Normalize m_{Θ_t}, M_{Θ_u}, and M_{Θ_p} by m_{Θ_t} and obtain L_u and L_p by fitting transfer functions to the normalized M_Θ's. Note that, for m_{Θ_t} to be scalar, the two Θ_t parameter blocks must be combined and redefined as one full block prior to

step 6 [BDG+91]. This is necessary in order to be able to normalize the M_Θ matrices, although it does unfortunately introduce substantial conservativeness. Note also that L_t is now equal to I.

8. Fix L_u and L_p and update the D scale. Restore the actual uncertainty block structure (with the two original Θ_t scalar blocks).

9. Repeat steps 2 to 8 until $\bar{\mu}$ no longer decreases.

This is similar to μ-synthesis and, likewise, is not guaranteed to find the global minimum solution; however, it works well in practice [SLBB96],[BAG96] and will therefore be used to design the aircraft controller in the next chapter. Note also that, as mentioned previously, an alternative to solving the GEVP is to iterate on the feasibility problem to find the minimum γ and the corresponding L-scales. This is especially recommended when the number of dummy variables added to solve the GEVP is extremely large and would thus require an inordinate amount of computer memory.

Finally, μ-synthesis is strongly recommended even when uncertainty is not considered, since μ optimizes the scales on performance and robust stability to further decrease γ; whereas the LPV approach alone will optimize only the scales corresponding to the time-varying block. In fact, the standard LPV approach is simply steps 1 to 5 of the D-K-D iteration procedure.

IV. Longitudinal Aircraft Control

This chapter will describe the implementation of gain scheduling theory for LPV systems for the design of an aircraft pitch rate controller operating over a substantial portion of its flight envelope. The aircraft, namely an F-18 Supermanuever-able fighter, will be designed to be both robustly stable and meet a Level 1 performance requirement throughout the controller design envelope, thereby guaranteeing robust performance.

4.1 Longitudinal Equations of Motion

It seems worthwhile to first briefly review the derivations and assumptions involved in obtaining the aircraft longitudinal equations of motion and the resulting state-space dynamics.

The standard nonlinear equations describing motion of an aircraft are given below. Equation (4.1) describes the translational forces along the aircraft body axes, and (4.2) describes the rotational forces about the same axes. In order to derive these simplified equations, it is necessary to assume that the aircraft is a rigid body, that its mass remains constant over time, and that the earth provides a fixed inertial reference frame.

$$\begin{aligned} X &= m(\dot{U} + QW - RV + g\sin\Theta) \\ Y &= m(\dot{V} - PW + RU - g\cos\Theta\sin\Phi) \\ Z &= m(\dot{W} + PV - QU - g\cos\Theta\cos\Phi) \end{aligned} \tag{4.1}$$

$$\begin{aligned} L &= \dot{P}I_x - \dot{R}I_{xz} + QR(I_z - I_y) - PQI_{xz} \\ M &= \dot{Q}I_y + PR(I_x - I_z) - R^2I_{xz} + P^2I_{xz} \\ N &= \dot{R}I_z - \dot{P}Ixz + PQ(I_y - I_x) + QRI_{xz} \end{aligned} \tag{4.2}$$

In the above equations, U, V, W are the translational velocities along the aircraft body axes; P, Q, R are the rotational velocities about the same axes; I_x, I_y, I_z, I_{xz} are the moments of inertia; g is gravity; m is the aircraft mass; and X, Y, Z, L, M, N

are the external forces and moments due to the aircraft aerodynamics and propulsion. The Euler angles Θ, Φ, Ψ describe the orientation of the aircraft with respect to the earth

$$\begin{aligned} \dot{\Theta} &= Q\cos\Phi - R\sin\Phi \\ \dot{\Phi} &= P + Q\tan\Theta\sin\Phi + R\tan\Theta\cos\Phi \\ \dot{\Psi} &= \frac{R\cos\Phi}{\cos\Theta} + \frac{Q\sin\Phi}{\cos\Theta} \end{aligned} \tag{4.3}$$

where Θ is the pitch angle, Φ is the roll angle, and Ψ is the yaw angle.

The six equations of motion can be further decoupled into two sets of simultaneous equations: three equations to describe the longitudinal dynamics, and three to describe the lateral dynamics. By assuming that the aircraft is in straight and level flight with only longitudinal disturbances present, we can neglect all lateral forces and moments, resulting in the following simplified equations:

$$\begin{aligned} X &= m(\dot{U} + QW + g\sin\Theta) \\ Z &= m(\dot{W} - QU - g\cos\Theta) \\ M &= \dot{Q}I_y \end{aligned} \tag{4.4}$$

To further simplify these expressions, it is then assumed that the velocities, forces, moments and Euler angles can be considered as small perturbations from their equilibrium values; and that products of these perturbation values are negligible. The equations are then expanded to express the longitudinal forces and moments in terms of changes in them resulting from the translational and angular velocities perturbation variables. This involves determining the partial derivatives of the forces and moments with respect to the perturbation variables. A detailed development of this can be found in [Bla91] or any good text on aircraft stability and control. The resulting linear longitudinal equations of motion can be expressed in state space

form as:

$$\begin{bmatrix} \dot{u} \\ \dot{a} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_\alpha & X_q & -g\cos\Theta_0 \\ Z_u & Z_\alpha & Z_q & -g\sin\Theta_0/U_0 \\ M_u & M_\alpha & M_q & M_\theta \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} X_{\delta_e} \\ Z_{\delta_e} \\ M_{\delta_e} \\ 0 \end{bmatrix} [\delta_e] \qquad (4.5)$$

where state variable u is the perturbation velocity along the x-axis, α is the perturbation angle of attack, q is the perturbation pitch rate, θ is the perturbation pitch angle, U_0 is the equilibrium velocity, and Θ_0 is the equilibrium pitch angle (note: in this specific context, θ and Θ_0 indicate pitch angles, not time-varying parameters). The A matrix is composed of the longitudinal stability derivatives, while the B matrix is composed of the longitudinal control derivatives. For convenience, all these derivatives will henceforth be referred to simply as the stability derivatives. The control input is δ_e which is the elevator deflection. Other longitudinal control inputs are possible (such as trailing edge flaps or thrust vectoring), but will not be considered for the controller design in this thesis.

Using (4.1)–(4.2), an atmospheric model, and a precise nonlinear aircraft model, local equilibrium flight conditions (also called trimmed conditions) can be determined where all accelerations are zero. Since this could allow several solutions with non-zero velocities, an aircraft flight mode must also be specified; the most common of these being straight and level flight. Using these trimmed flight conditions, values for the stability derivatives can be then be obtained. In this way, the aircraft dynamics at a point in its flight envelope can be extremely well described by the resulting state space equations.

For the F-18 under study, such trimmed conditions were determined at 18 points in a subsonic flight envelope ranging from 5000 ft to 40000 ft, with Mach number values of 0.3 to 0.95, using the nonlinear F-18 aircraft model described in [ABSB92]. These points are shown in Figure 4.1, with the corresponding values of

the flight conditions given in Appendix A (the state space A and B matrices are given in program `f18longdat.m` of Appendix C). Note that X_q and M_θ are often neglected as a result of linearizing the equations of motion. However, these stability derivatives have been kept in this case since the trimmed conditions for the F-18 provided values for them.

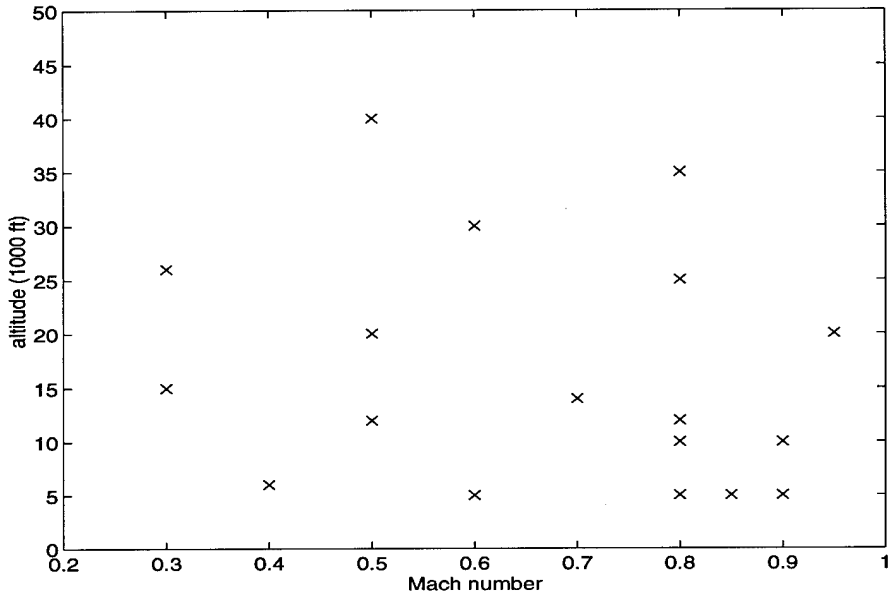


Figure 4.1 Flight envelope showing trimmed data points

Typically, an aircraft will exhibit two natural longitudinal modes of motion: the short period mode and the phugoid mode. The phugoid mode is characterized by a long time period (low natural frequency) and is lightly damped, while the short period mode has a shorter time period (higher natural frequency) and is more heavily damped. For a manual flight control system, the short period is the primary mode of interest since it dominates the response of the aircraft to pilot inputs, and the pilot can usually easily correct for the slight error caused by the slowly varying phugoid. As well, the pitch rate response is dominated by the short period and, for time frames of typically 10 seconds or less, exhibits very little phugoid motion. For these reasons, the longitudinal state space model can be reduced to a second order short period approximation, given in (4.6). Over a short time period, it provides a

relatively accurate measure of the aircraft pitch response.

$$\begin{bmatrix} \dot{a} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} Z_\alpha & Z_q \\ M_\alpha & M_q \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} Z_{\delta_e} \\ M_{\delta_e} \end{bmatrix} [\delta_e] \tag{4.6}$$

In this thesis two pitch rate controllers will be designed; one for the short period model, and another for the full longitudinal model. However, since it is the more accurate one, the full longitudinal state space model will always be used as the aircraft model for simulated tests of the controllers.

4.2 The LPV Aircraft Model

As explained in Chapter III, the aircraft must first be modelled as an LPV plant. The first step in doing this is to model each of the stability derivatives as a polynomial expression dependent on some measurable parameters such as Mach number, altitude, angle of attack, or dynamic pressure. For the full longitudinal model, this requires curve-fitting the 15 stability derivatives located in the first three rows of the A and B matrices of (4.5). Note that, since Θ_0 is unknown, the expressions in the last column of the A matrix are also being treated as stability derivatives; namely, X_θ and Z_θ respectively.

The method of least-squares was used to fit polynomial expressions of various parameters. Basically, for each stability derivative, this required determining which parameters it depended upon the most, and combining increasing orders of these parameters until a satisfactory curve fit was obtained. While not all of the stability derivatives depended as well on some parameters as others, it was decided that, in this case, the stability derivatives depended mainly on Mach number and altitude. These were therefore chosen as the time-varying parameters $\theta(t)$ on which the LPV model of the plant will depend. Combining various powers of Mach number and altitude to obtain as good a fit as possible while maintaining as low an order as possible resulted in the following polynomial functions for the stability derivatives.

Note that this was done for all stability derivatives, although for convenience, only the short period stability derivatives are listed here.

$$\begin{aligned} Z_\alpha &= (8.4931e+00)+(-5.8946e-04)h+(1.0695e-08)h^2+(-4.7974e+01)M \\ &\quad +(3.0496e-03)Mh+(-5.0945e-08)Mh^2+(7.9105e+01)M^2 \\ &\quad +(-4.8436e-03)M^2h+(7.5817e-08)M^2h^2+(-4.4981e+01)M^3 \\ &\quad +(2.5679e-03)M^3h+(-3.7016e-08)M^3h^2 \end{aligned} \tag{4.7}$$

$$Z_q = (9.8313e-01)+(3.9110e-07)h+(-3.9162e-04)M+(-5.3330e-08)Mh \tag{4.8}$$

$$\begin{aligned} Z_{\delta_e} &= (9.7204e-02)+(5.6449e-06)h+(-4.3877e-10)h^2+(-1.0410e+00)M \\ &\quad +(-5.5878e-06)Mh+(2.2222e-09)Mh^2+(1.0255e+00)M^2 \\ &\quad +(2.2785e-05)M^2h+(-4.1014e-09)M^2h^2+(-4.8758e-01)M^3 \\ &\quad +(-1.3154e-05)M^3h+(2.3785e-09)M^3h^2 \end{aligned} \tag{4.9}$$

$$\begin{aligned} M_\alpha &= (4.0000e+02)+(-3.5864e-02)h+(8.1789e-07)h^2+(-2.0713e+03)M \\ &\quad +(1.7638e-01)Mh+(-3.9293e-06)Mh^2+(3.4457e+03)M^2 \\ &\quad +(-2.7712e-01)M^2h+(6.0257e-06)M^2h^2+(-1.8966e+03)M^3 \\ &\quad +(1.4207e-01)M^3h+(-2.9944e-06)M^3h^2 \end{aligned} \tag{4.10}$$

$$\begin{aligned} M_q &= (-1.9283e+00)+(1.0426e-04)h+(-1.8301e-09)h^2+(7.3411e+00)M \\ &\quad +(-2.3624e-04)Mh+(8.0762e-10)Mh^2+(-1.4103e+01)M^2 \\ &\quad +(1.7893e-04)M^2h+(8.9252e-09)M^2h^2+(6.7100e+00)M^3 \\ &\quad +(3.4962e-05)M^3h+(-9.5971e-09)M^3h^2 \end{aligned} \tag{4.11}$$

$$\begin{aligned} M_{\delta_e} &= (6.2187e+01)+(-5.6127e-03)h+(1.0681e-07)h^2+(-3.1392e+02)M \\ &\quad +(2.8099e-02)Mh+(-5.0915e-07)Mh^2+(4.3654e+02)M^2 \\ &\quad +(-4.1827e-02)M^2h+(7.3379e-07)M^2h^2+(-2.5310e+02)M^3 \\ &\quad +(2.1593e-02)M^3h+(-3.4779e-07)M^3h^2 \end{aligned} \tag{4.12}$$

Figures comparing the polynomial curve fits to the actual values for all 15 stability derivatives are included in Appendix B.

As explained in Chapter III, the parameters will be normalized such that $\theta(t) \leq 1$ so that our eventual goal will be to design a controller which makes the robust performance objective $\gamma < 1$. The Mach number and altitude ranges for the modelled envelope are:

$$M \in [0.3, 0.95] \text{ and } h \in [5000, 40000] \text{ ft.} \tag{4.13}$$

Expressing these in terms of their normalized parameters such that $\theta(t) \in [-1, 1]$ gives:

$$M = 0.325\theta_M + 0.625 \tag{4.14}$$

$$h = 17500\theta_h + 22500 \tag{4.15}$$

These expressions can now be inserted into the polynomial functions given in (4.7)–(4.12) to produce normalized polynomial functions of the stability derivatives. For Z_α, this would result in the following equation:

$$\begin{aligned} Z_\alpha \;=\; & (-7.0152e-01) + (4.5365e-01)\theta_h + (-1.7379e-01)\theta_h^2 + (-4.8108e-01)\theta_M \\ & +(1.3941e-01)\theta_M\theta_h + (4.4512e-01)\theta_M\theta_h^2 + (-2.7859e+00)\theta_M^2 \\ & +(4.8005e+00)\theta_M^2\theta_h + (2.0740e+00)\theta_M^2\theta_h^2 + (-2.0404e+00)\theta_M^3 \\ & +(5.4195e+00)\theta_M^3\theta_h + (-3.8915e+01)\theta_M^3\theta_h^2 \end{aligned} \tag{4.16}$$

Once all the equations (4.7)–(4.12) have been normalized, we now have a non-linear parameter-varying model of the plant, where each of the elements of the state space model is given by a polynomial function of two normalized parameters: θ_M and θ_h. The dependence of the plant on these parameters is known, but since no information on Mach number and altitude are known *a priori* (except their range of values), this fits precisely into the LPV model defined in (3.2). We now need to express this nonlinear model as a linear fractional function of the parameters. In

other words, we need to model this as a nominal plant perturbed by the parameters, with added elements relating how the parameters will affect the nominal mapping.

As explained in Chapter II, an LFT can be thought of as a nominal mapping perturbed by some model uncertainty, with the other elements of the plant relating how the uncertainty affects the nominal plant. This is precisely what we wish to achieve, although in this case the "uncertainty" is composed of the time-varying parameters. Since the nominal mapping must be independent of the parameters, we can clearly see that the nominal plant will be composed of the constant terms in the normalized polynomial expressions. However, in order to determine the other terms, it is first necessary to convert each of the normalized polynomial equations into an LFT. Looking at (4.5) or (4.6), we can see that each of the stability derivatives is simply an input/output relation between specific inputs or states to specific outputs. Therefore, we can use the technique from Chapter II to convert these polynomial functions to LFT's. For example, the short period equation for $\dot{\alpha}$ is

$$\dot{\alpha} = Z_\alpha \alpha + Z_q q + Z_{\delta_e} \delta_e \tag{4.17}$$

where Z_α is given in (4.16). For convenience, we can rewrite this as

$$\dot{\alpha} = \dot{\alpha}_\alpha + \dot{\alpha}_q + \dot{\alpha}_{\delta_e} \tag{4.18}$$

where, for example,

$$\dot{\alpha}_\alpha := Z_\alpha \alpha. \tag{4.19}$$

The equation for $\dot{\alpha}_\alpha$ is now a simple input/output expression whose transfer function is given by the Z_α polynomial. For ease of notation, (4.16) will be generalized as:

$$Z_\alpha = a+b\theta_h+c\theta_h^2+d\theta_M+f\theta_M\theta_h+g\theta_M\theta_h^2+j\theta_M^2+k\theta_M^2\theta_h+m\theta_M^2\theta_h^2+n\theta_M^3+p\theta_M^3\theta_h+r\theta_M^3\theta_h^2 \tag{4.20}$$

Following the procedure outlined in Chapter II, the block diagram shown in Figure 4.2 can now be drawn to determine all the input/output relations. Note that there are several other realizations possible, depending on how the θ_M's and θ_h's are arranged; a continuing problem in control theory is how to guarantee that we always have the minimal number of θ to represent polynomial functions such as these [ZDG96].

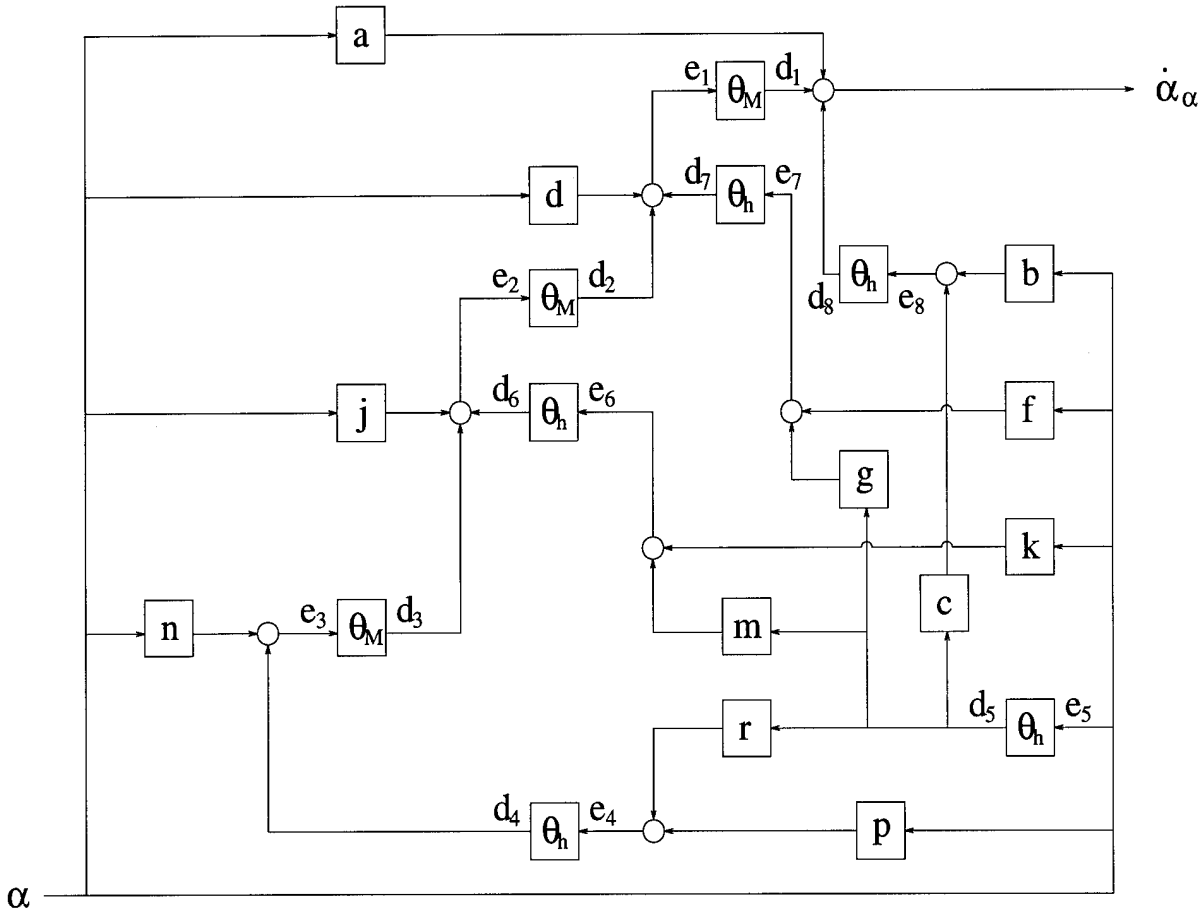


Figure 4.2 Block diagram of Z_α showing input/output relations

Pulling out the θ's into one block diagonal parameter structure and writing the outputs in terms of the inputs gives (for $\dot{\alpha}_\alpha$):

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ \dot{\alpha}_\alpha \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & d \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & j \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & n \\ 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & p \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & k \\ 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & f \\ 0 & 0 & 0 & 0 & c & 0 & 0 & 0 & b \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & a \end{bmatrix}}^{\bar{Z}_\alpha} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \\ \alpha \end{bmatrix} \tag{4.21}$$

or,

$$\begin{bmatrix} e_{\theta_1} \\ \dot{\alpha}_\alpha \end{bmatrix} = \overbrace{\begin{bmatrix} A_{Z_\alpha} & B_{Z_\alpha} \\ C_{Z_\alpha} & D_{Z_\alpha} \end{bmatrix}}^{\bar{Z}_\alpha} \begin{bmatrix} d_{\theta_1} \\ \alpha \end{bmatrix} \tag{4.22}$$

where $\bar{Z}_\alpha$ is a coefficient matrix (not a transfer matrix) and, as expected, the constant term $D_{Z_\alpha} = a$ is the nominal map for the upper LFT representation of Z_α. For completeness,

$$\dot{\alpha}_\alpha := Z_\alpha \alpha = F_u(\bar{Z}_\alpha, \Theta_{Z_\alpha})\alpha, \text{ where } \Theta_{Z_\alpha} = \begin{bmatrix} \theta_M I_3 & 0 \\ 0 & \theta_h I_5 \end{bmatrix}. \tag{4.23}$$

This LFT representation of Z_α is shown in Figure 4.3. Each of the stability derivative polynomials can be converted to LFT's in the same fashion. When this is completed, the LFT's can then be linearly interconnected to form the parameter-varying system shown in Figure 4.4 (for the short period model).

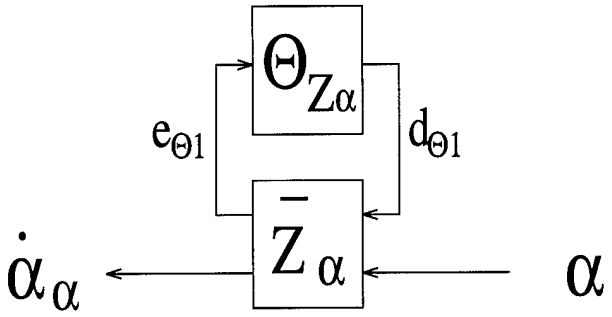


Figure 4.3 LFT representation of Z_α

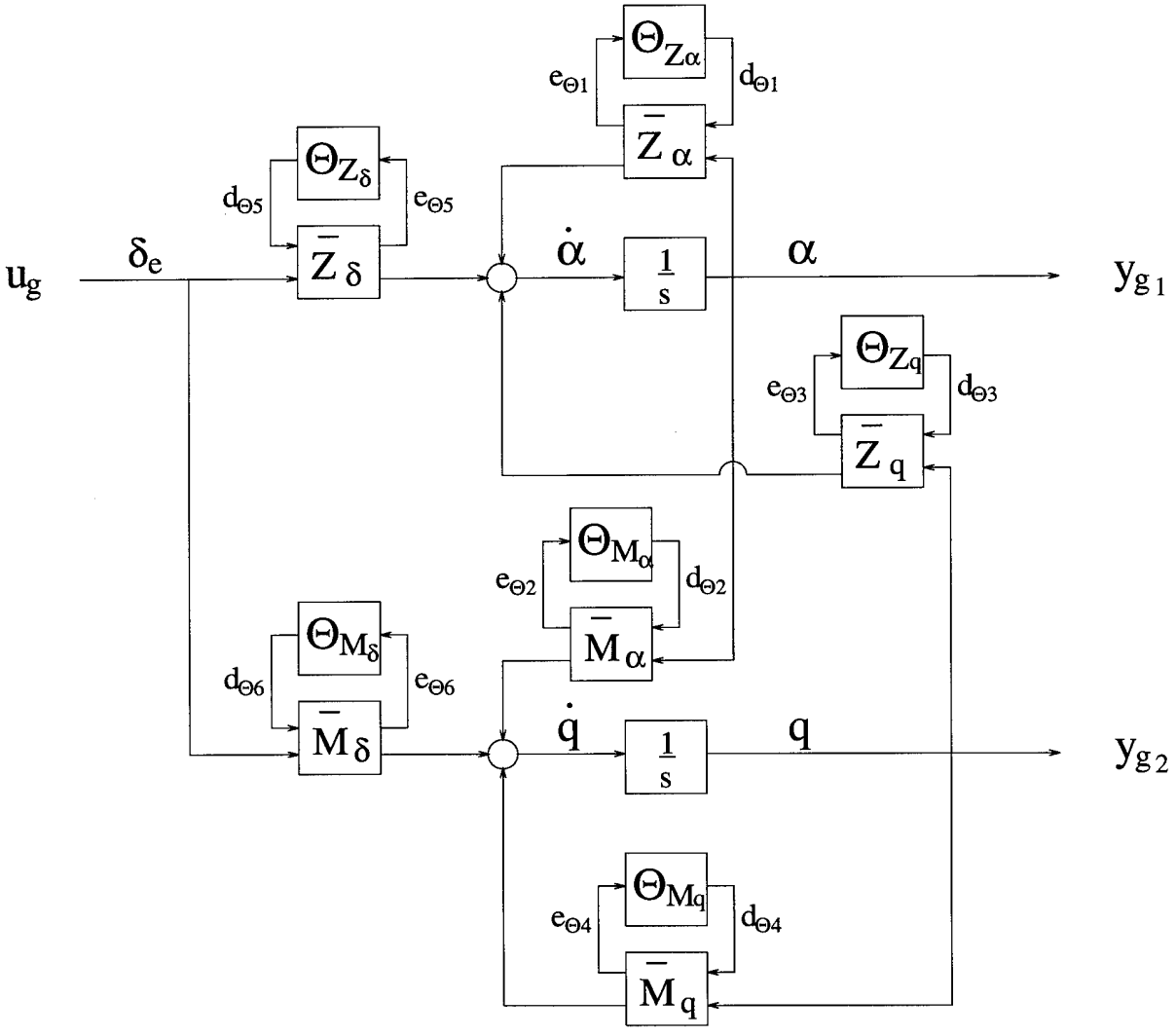


Figure 4.4 Short period model of LPV system

This linear interconnection of LFT's can now be grouped into one combined LFT by gathering the known systems together and by combining the varying parameter blocks together. Using the notation of (4.23) for each of the coefficient matrices in Figure 4.4, we can obtain the following state space equations for the LPV system. Note that, as shown, the two outputs for the plant have been chosen to be the states α and q.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ e_{\theta_1} \\ e_{\theta_2} \\ e_{\theta_3} \\ e_{\theta_4} \\ e_{\theta_5} \\ e_{\theta_6} \\ y_{g_1} \\ y_{g_2} \end{bmatrix} = \begin{bmatrix} D_{Z_\alpha} & D_{Z_q} & C_{Z_\alpha} & 0 & C_{Z_q} & 0 & C_{Z_\delta} & 0 & D_{Z_\delta} \\ D_{M_\alpha} & D_{M_q} & 0 & C_{M_\alpha} & 0 & C_{M_q} & 0 & C_{M_\delta} & D_{M_\delta} \\ B_{Z_\alpha} & 0 & A_{Z_\alpha} & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{M_\alpha} & 0 & 0 & A_{M_\alpha} & 0 & 0 & 0 & 0 & 0 \\ 0 & B_{Z_q} & 0 & 0 & A_{Z_q} & 0 & 0 & 0 & 0 \\ 0 & B_{M_q} & 0 & 0 & 0 & A_{M_q} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{Z_\delta} & 0 & B_{Z_\delta} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{M_\delta} & B_{M_\delta} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ d_{\theta_1} \\ d_{\theta_2} \\ d_{\theta_3} \\ d_{\theta_4} \\ d_{\theta_5} \\ d_{\theta_6} \\ u_g \end{bmatrix} \tag{4.24}$$

The combined parameter block Θ has a structured "uncertainty" form given by

$$\Theta = blockdiag(\Theta_{Z_\alpha}, \Theta_{M_\alpha}, \Theta_{Z_q}, \Theta_{M_q}, \Theta_{Z_\delta}, \Theta_{M_\delta}) \tag{4.25}$$

Equivalently, we can write this system as the transfer matrix $G(s)$ perturbed by the parameter structure Θ. $G(s)$ is then

$$G(s) = \left[\begin{array}{cc|ccccccc} D_{Z_\alpha} & D_{Z_q} & C_{Z_\alpha} & 0 & C_{Z_q} & 0 & C_{Z_\delta} & 0 & D_{Z_\delta} \\ D_{M_\alpha} & D_{M_q} & 0 & C_{M_\alpha} & 0 & C_{M_q} & 0 & C_{M_\delta} & D_{M_\delta} \\ \hline B_{Z_\alpha} & 0 & A_{Z_\alpha} & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{M_\alpha} & 0 & 0 & A_{M_\alpha} & 0 & 0 & 0 & 0 & 0 \\ 0 & B_{Z_q} & 0 & 0 & A_{Z_q} & 0 & 0 & 0 & 0 \\ 0 & B_{M_q} & 0 & 0 & 0 & A_{M_q} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{Z_\delta} & 0 & B_{Z_\delta} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{M_\delta} & B_{M_\delta} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{4.26}$$

To simplify the notation, the state space equations will be grouped together as follows

$$\begin{bmatrix} \dot{x}_g \\ e_\theta \\ y_g \end{bmatrix} = \begin{bmatrix} A_g & B_\theta & B_{u_g} \\ C_\theta & D_{\theta\theta} & D_{\theta u_g} \\ C_{y_g} & D_{y_g\theta} & D_{y_g u_g} \end{bmatrix} \begin{bmatrix} x_g \\ d_\theta \\ u_g \end{bmatrix} \tag{4.27}$$

Therefore,

$$G(s) = \left[\begin{array}{c|cc} A_g & B_\theta & B_{u_g} \\ \hline C_\theta & D_{\theta\theta} & D_{\theta u_g} \\ C_{y_g} & D_{y_g\theta} & D_{y_g u_g} \end{array}\right] \tag{4.28}$$

Further on, we will want to extract the pitch rate output q. This will be specified as

$$q = C_{y_{g_2}} x_g. \tag{4.29}$$

In addition, note that $D_{y_g\theta} = 0$; therefore, assumption 3 of Chapter III has been met, thereby ensuring that the LPV controller will be causal and well-posed.

It should also be pointed out that, while the θ_M's and θ_h's for each individual stability derivative were grouped together (as, for example, in (4.23)) these repeated scalar blocks of θ_M's and θ_h's were not combined together in (4.25) for the different stability derivatives. For practical reasons, it is usually convenient to combine them. This then requires rearranging the corresponding input/output d_θ, e_θ channels in (4.24) and (4.26). This will be assumed to have been performed in (4.27). Thus the time-varying parameter block can be expressed as

$$\Theta = \begin{bmatrix} \theta_M I_{r_M} & 0 \\ 0 & \theta_h I_{r_h} \end{bmatrix} \tag{4.30}$$

where r_M and r_h are the respective dimensions of θ_M and θ_h. Note that Θ is now a repeated scalar structured "uncertainty" block.

The combined LPV aircraft model is now an LFT, and can be simply drawn as in Figure 4.5. This is very nearly the form we need to make use of the gain scheduling theory outlined in Chapter III; all that is missing is the overall setup of the weighted aircraft/controller feedback design model.

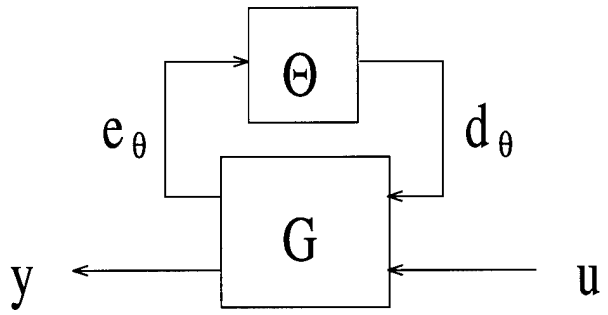


Figure 4.5 LPV Aircraft Model

4.3 The Design Model

The problem is to design a robustly performing pitch rate controller for the F-18 Supermaneuverable fighter aircraft. Frequency and time domain specifications for a predicted Level 1 handling qualities response can be found in MIL-STD-1797A. These

suggest that a damping factor ζ_{sp} of 0.5 and natural frequency ω_{sp} of 4 radians/second would meet Level 1 handling qualities over the chosen flight envelope. Thus the ideal pitch rate response can be approximated by the following second order system:

$$\begin{aligned} H(s) &= \frac{\omega_{sp}^2}{s^2 + 2\zeta_{sp}\omega_{sp}s + \omega_{sp}^2} \\ &= \frac{16}{s^2 + 4s + 16} \end{aligned} \tag{4.31}$$

The design approach will be to model-match the response of the plant to the reference model response given in (4.31). This should provide nearly the same pitch rate response over the entire envelope. Alternatively, an LPV reference model could be used if the desired response had to vary over the flight envelope. The system design model is shown in Figure 4.6.

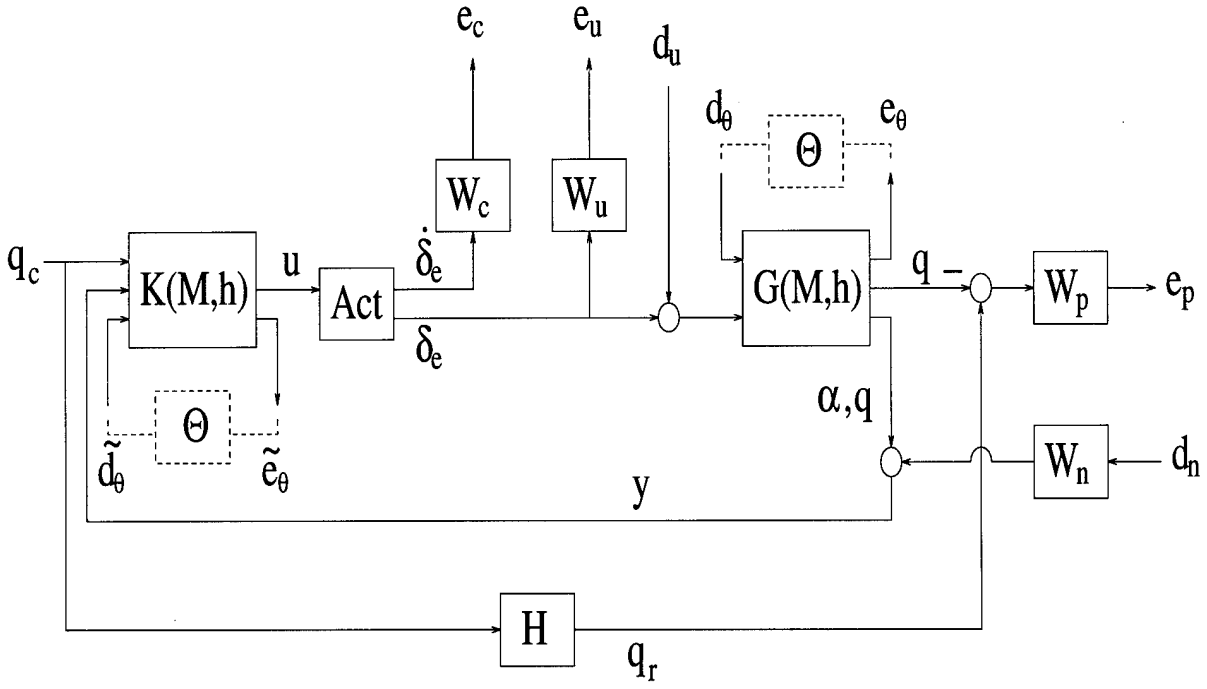


Figure 4.6 Design model

In state space, the chosen reference model can be written as

$$\begin{bmatrix} \dot{x}_h \\ q_r \end{bmatrix} = \begin{bmatrix} A_h & B_h \\ C_h & D_h \end{bmatrix} \begin{bmatrix} x_h \\ q_c \end{bmatrix} \tag{4.32}$$

Converting the transfer function in (4.31) to state space form gives

$$H(s) = \left[\begin{array}{c|c} A_h & B_h \\ \hline C_h & D_h \end{array}\right] = \left[\begin{array}{cc|c} -4 & -16 & 1 \\ 1 & 0 & 1 \\ \hline 0 & 16 & 0 \end{array}\right] \tag{4.33}$$

Note that this is only one of several possible state space forms since these are not unique.

The aircraft plant is specified in (4.27)–(4.29). Note that, as shown in Figure 4.6, both q and y_g are output. The output q is differenced with the reference pitch rate response q_r to produce an error signal which is then weighted by W_p to emphasize the frequency range of interest. For the short period design, W_p was chosen as

$$W_p = \frac{0.4(s+100)}{s+4} \tag{4.34}$$

whereas for the full longitudinal model, W_p was chosen to be

$$W_p = \frac{0.04(s+100)}{s+4} \tag{4.35}$$

For the short period, this effectively guarantees that the steady-state pitch rate error (to a unit step response) in the final design will be less than 0.1 degrees/second if an H_∞ norm (or peak μ value) less than one is achieved. It can also be thought of as a low-pass weight on sensitivity to provide nominal performance. Some iterating was required to obtain W_p, because the aircraft dynamics and the design problem formulation determines how small a performance error will be possible before robust

performance is no longer met. For instance, the full longitudinal model could only guarantee that the error to a step response would be less than 1 degree/second for a similar flight envelope. This is only a conservative guarantee based largely on the Small Gain Theorem ; in fact, the response will generally be much better. The state space equations for W_p can be generalized as

$$\begin{bmatrix} \dot{x}_p \\ e_p \end{bmatrix} = \begin{bmatrix} A_p & B_p \\ C_p & D_p \end{bmatrix} \begin{bmatrix} x_p \\ q_r - q \end{bmatrix} \tag{4.36}$$

The output y_g (composed of α and q) is added to a noise weighting W_n to produce a measured y_m. The noise weighting can be thought of as the error expected in the measurements or, alternatively, as a weight on complementary sensitivity. This should reduce the effect of the high frequency measurement noise in the design. In order to minimize the dimension of the eventual LMI problem and the number of controller states, W_n was chosen to be simply a static weight given by

$$W_n = 0.05 I_2 \tag{4.37}$$

which can be viewed as a 5% error expected in both measurements, or as a weight on complementary sensitivity between the output y_g and the input d_n. For the full longitudinal design, this weight was decreased to

$$W_n = 0.01 I_2 \tag{4.38}$$

in order to increase the relative importance of performance and stability, which, for this mode, are more difficult to attain.

Also included in the design model is a first order approximation of the actuator dynamics with a pole at $s = -20.2$. As shown in Figure 4.6, we would like not only to output the elevator deflection δ_e, but also the rate of elevator deflection $\dot{\delta}_e$. This

allows the designer to penalize either the elevator usage or the rate of elevator usage (or both). The state space equations can be written as

$$\begin{bmatrix} \dot{x}_a \\ \delta_e \\ \dot{\delta}_e \end{bmatrix} = \begin{bmatrix} A_a & B_a \\ C_a & D_a \end{bmatrix} \begin{bmatrix} x_a \\ u \end{bmatrix} \tag{4.39}$$

Numerically, this leads to

$$Act(s) = \left[\begin{array}{c|c} A_a & B_a \\ \hline C_a & D_a \end{array}\right] = \left[\begin{array}{c|c} -20.2 & 20.2 \\ \hline 1 & 0 \\ -20.2 & 20.2 \end{array}\right] \tag{4.40}$$

Note that, in this case, since $\delta_e = x_a$ then

$$\dot{\delta}_e = \dot{x}_a = A_a x_a + B_a u \tag{4.41}$$

For this design, penalizing the elevator deflection rate was sufficient. The control weight or penalty W_c was therefore selected to be

$$W_c = 1/70. \tag{4.42}$$

This ensures that the maximum allowable deflection rate of 70 degrees/second will not be violated if an H_∞ norm (or peak μ value) less than one is achieved. The control output e_c can then be written as

$$e_c = W_c \dot{\delta}_e. \tag{4.43}$$

To reflect some uncertainty in the measured parameters (and thus y_g) and the fact that we are not really modelling high frequency dynamics, an uncertainty

weighting W_u was added. An additive uncertainty "across" $G(s)$ could have been used, or even an uncertainty in each of the measured time-varying parameters, but this would have added several more input/output channels and increased the dimension of the LMI problem to be optimized. Since this gain scheduling approach is based on the Small Gain Theorem and thus quite conservative already, it was finally decided to add only a single input multiplicative uncertainty. After some iterations, the weighting W_u was chosen as

$$W_u = \frac{0.1(s+100)}{(s+10000)} \tag{4.44}$$

This can be viewed as a high-pass weight on complementary sensitivity to provide robust stability, or as increased uncertainty at higher frequencies. It can be generalized as

$$\begin{bmatrix} \dot{x}_u \\ e_u \end{bmatrix} = \begin{bmatrix} A_u & B_u \\ C_u & D_u \end{bmatrix} \begin{bmatrix} x_u \\ \delta_e \end{bmatrix} \tag{4.45}$$

The uncertainty input d_u is added to the elevator deflection δ_e to produce the input u_g of the aircraft plant $G(s)$. Specifically,

$$u_g = \delta_e + d_u. \tag{4.46}$$

Finally, to increase the flexibility of the controller, a 2 degree-of-freedom controller design was adopted. This is similar to the stability augmentation system (SAS) of classical control, and provides the controller with more information on the states α and q than it would have with only a standard error signal. The output of the controller is the commanded elevator deflection u and the inputs are q_c and y_m, where y_m is a "noisy" α and q given by

$$y_m = y_g + W_n d_n. \tag{4.47}$$

Therefore, the controller input y is defined as

$$y = \begin{bmatrix} q_c \\ y_m \end{bmatrix} \tag{4.48}$$

The linear interconnection of all these components as depicted in Figure 4.6 yields the following state space equations for the design model:

$$\begin{bmatrix} \dot{x}_g \\ \dot{x}_a \\ \dot{x}_p \\ \dot{x}_h \\ \dot{x}_u \\ e_\theta \\ e_u \\ e_p \\ e_c \\ q_c \\ y_m \end{bmatrix} = \begin{bmatrix} Ag & B_{u_g} & 0 & 0 & 0 & B_\theta & B_{u_g} & 0 & 0 & 0 \\ 0 & A_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_a \\ -B_p C_{y_{g_2}} & 0 & A_p & B_p C_h & 0 & 0 & 0 & B_p D_h & 0 & 0 \\ 0 & 0 & 0 & A_h & 0 & 0 & 0 & B_h & 0 & 0 \\ 0 & B_u & 0 & 0 & A_u & 0 & 0 & 0 & 0 & 0 \\ C_\theta & D_{\theta u_g} & 0 & 0 & 0 & D_{\theta\theta} & D_{\theta u_g} & 0 & 0 & 0 \\ 0 & D_u & 0 & 0 & C_u & 0 & 0 & 0 & 0 & 0 \\ -D_p C_{y_{g_2}} & 0 & C_p & D_p C_h & 0 & 0 & 0 & D_p D_h & 0 & 0 \\ 0 & W_c A_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W_c B_a \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ C_{y_g} & D_{y_g u_g} & 0 & 0 & 0 & D_{y_g \theta} & D_{y_g u_g} & 0 & W_n & 0 \end{bmatrix} \begin{bmatrix} x_g \\ x_a \\ x_p \\ x_h \\ x_u \\ d_\theta \\ d_u \\ q_c \\ d_n \\ u \end{bmatrix} \tag{4.49}$$

We can now write this as the design plant $P(s)$ given by

$$P(s) = \left[\begin{array}{ccccc|ccccc} Ag & B_{u_g} & 0 & 0 & 0 & B_\theta & B_{u_g} & 0 & 0 & 0 \\ 0 & A_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_a \\ -B_pC_{y_{g_2}} & 0 & A_p & B_pC_h & 0 & 0 & 0 & B_pD_h & 0 & 0 \\ 0 & 0 & 0 & A_h & 0 & 0 & 0 & B_h & 0 & 0 \\ 0 & B_u & 0 & 0 & A_u & 0 & 0 & 0 & 0 & 0 \\ \hline C_\theta & D_{\theta u_g} & 0 & 0 & 0 & D_{\theta\theta} & D_{\theta u_g} & 0 & 0 & 0 \\ 0 & D_u & 0 & 0 & C_u & 0 & 0 & 0 & 0 & 0 \\ -D_pC_{y_{g_2}} & 0 & C_p & D_pC_h & 0 & 0 & 0 & D_pD_h & 0 & 0 \\ 0 & W_cA_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W_cB_a \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ C_{y_g} & D_{y_gu_g} & 0 & 0 & 0 & D_{y_g\theta} & D_{y_gu_g} & 0 & W_n & 0 \end{array}\right] \tag{4.50}$$

The state space equations can then be grouped together to recover the notation of Chapter III as follows

$$\begin{bmatrix} \dot{x} \\ e_\theta \\ e \\ y \end{bmatrix} = \begin{bmatrix} A & B_\theta & B_d & B_u \\ C_\theta & D_{\theta\theta} & D_{\theta d} & D_{\theta u} \\ C_e & D_{e\theta} & D_{ed} & D_{eu} \\ C_y & D_{y\theta} & D_{yd} & D_{yu} \end{bmatrix} \begin{bmatrix} x \\ d_\theta \\ d \\ u \end{bmatrix} \tag{4.51}$$

$$\text{where } e = \begin{bmatrix} e_u \\ e_p \\ e_c \end{bmatrix} \text{ and } d = \begin{bmatrix} d_u \\ d_p \\ d_c \end{bmatrix} \tag{4.52}$$

Therefore,

$$P(s) = \left[\begin{array}{c|ccc} A & B_\theta & B_d & B_u \\ \hline C_\theta & D_{\theta\theta} & D_{\theta d} & D_{\theta u} \\ C_e & D_{e\theta} & D_{ed} & D_{eu} \\ C_y & D_{y\theta} & D_{yd} & D_{yu} \end{array}\right] \tag{4.53}$$

This is now exactly the LTI design plant $P(s)$ referred to in Chapter III. We can now add the time-varying parameter block of the controller and the resulting added plant interconnections to yield the augmented plant $P_a(s)$ given in (3.21) and reproduced below.

$$P_a(s) = \left[\begin{array}{c|ccccc} A & 0 & B_\theta & B_d & B_u & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I_r \\ C_\theta & 0 & D_{\theta\theta} & D_{\theta d} & D_{\theta u} & 0 \\ C_e & 0 & D_{e\theta} & D_{ed} & D_{eu} & 0 \\ C_y & 0 & D_{y\theta} & D_{yd} & D_{yu} & 0 \\ 0 & I_r & 0 & 0 & 0 & 0 \end{array}\right] \tag{4.54}$$

The resulting LPV design is then identical to the one shown in Figure 3.3 with the time-varying parameter block now given by

$$\Delta \oplus \Delta = \left[\begin{array}{cc} \Theta & 0 \\ 0 & \Theta \end{array}\right] \tag{4.55}$$

This plant P_a is the one used to solve the gain-scheduled robust performance H_∞ problem described in Chapter III. By solving either the GEVP or iterating on the feasibility problem, one can determine the L-scales required to find the corresponding controller. Since this particular design includes some uncertainty, the D-K-D iteration scheme will be used to incorporate the advantages of μ-synthesis. In fact, μ-synthesis is recommended even when uncertainty is not considered, since μ

optimizes the scales on all input/output channels; whereas the LPV approach alone will optimize only the scales corresponding to the time-varying block.

Further, for the purpose of this thesis, a robust stability subproblem will be defined where the only input/output channels in the design plant are d_θ and e_θ and, correspondingly, the only component of the uncertainty block is the time-varying parameter block Θ. This can also be achieved by making all the design weights extremely small. By solving the GEVP or simply testing the feasibility problem for $\gamma < 1$, we can then determine if the plant is at least robustly stabilizable in the given flight envelope. If not, the envelope can be gradually reduced until it is. Clearly, the robust performance problem will require an envelope at least as small, typically smaller.

4.4 Implementation

Most of the programs required to optimize this uncertain gain-scheduled H_∞ problem have been included in Appendix C. This section briefly describes these, along with some other implementation issues.

As mentioned previously, the full longitudinal state space A and B matrices for the 18 trimmed data points are available in program `f18longdat.m`. Using these data points, the program `f18poly.m` performs a least-squares curve fit to provide a polynomial expression for each of the 15 stability derivatives (for the short period, this is performed by `f18sppoly.m`). This program can also be modified to change the flight envelope, and/or the order of the curve fit, and/or the desired time-varying parameters. For the simulation aircraft model, the full longitudinal model was used, and its stability derivatives were fitted with polynomials of relatively high order (with as high as M^3h^2 terms) to minimize the model error. Six of these polynomials are given in (4.7)–(4.12), and the actual curve fits are illustrated in Appendix B.

Unfortunately, these high-order fits were extremely computationally demanding due to the resulting large number of variables to optimize when the LMI problem

was formulated, especially when performing D-K-D iterations which add additional weights to an already large problem. Computation was extremely slow and often ran out of available memory. In order to improve the LMI solver efficiency, the order of the polynomials was decreased so that the highest terms were of order M^2h. In addition, after several design iterations, it soon became clear that the envelope was too large for only one controller. For this reason, the envelope was iteratively decreased until robust stability was at least attainable (this will be demonstrated in Chapter V), and then further decreased until robust performance was possible. Thus, the design envelope for both the short period design and the full longitudinal design was finally chosen to be

$$M \in [0.5, 0.95] \text{ and } h \in [5000, 30000] \text{ ft.} \tag{4.56}$$

The resulting curve fits for these envelopes are also illustrated in Appendix B. Due to the reduced envelope (and thus less points to curve fit) the lower-order polynomials do model the aircraft dynamics relatively well. In addition, the conservativeness built into this gain-scheduling process and the modelled uncertainty may in large part offset the remaining curve-fit inaccuracies. The results, as we will see, tend to support this. Note that the simulation model retains the entire flight envelope and the original higher order curve fits in order to accurately test the resulting controllers beyond their chosen design envelopes.

The approach of the last section was then used to normalize the polynomials and convert both the short period and full longitudinal models into LFT's. As you may anticipate, this procedure can get very complex and intensive, especially considering the fact that the full longitudinal model has 11 additional stability derivatives. Fortunately, the Wright Laboratory Flight Dynamics Group has developed an algorithm to automate this process for the short period model of an aircraft, given two-parameter polynomial expressions for the stability derivatives [SLBB96]. A slightly

modified version of this program called `f18sppoly2lft.m` was used to convert the short period model into an LFT. This program was then adapted to convert the full longitudinal aircraft model into an LFT; it is included as program `f18poly2lft.m`. In addition, the programs also attempt to find the minimal realization of the resulting LFT. As mentioned previously, it is not guaranteed to find the absolute minimal realization, but it does manage to substantially reduce the dimension of the time-varying parameter block. This in turn reduces the number of input/output channels and thus the size of the plant; and, as a result, the number of variables which the LMI solver will eventually be attempting to optimize. This is very significant since, as mentioned, the LMI solver is very computationally expensive in terms of both memory and CPU requirements.

Program `f18ol.m` was then used to obtain $G(s)$, $P(s)$, and $P_a(s)$ for the full longitudinal model. For the short period, this is performed by `f18olsp.m`. For this design, the μ-Toolbox MATLAB program `sysic.m` was used to form the design plants; this automates the setup of the plants as developed in the last section.

We now have the design model necessary to perform the D-K-D iteration, implemented as program `dkdM18.m` for both design models. This program is basically an adaptation of the μ-Toolbox `dkit.m` program which performs the D-K iteration. Basically, once $P_a(s)$ has been created, the LMI Control Toolbox is used to solve either the GEVP or the feasibility problem for the L-scales corresponding to the time-varying parameter blocks. For a large problem, such as the full longitudinal one, it is often faster to first test for the feasibility of desired γ values; this helps to quickly eliminate problem formulations that will not lead to valid designs. The feasibility problem is set up in program `nshinflmi2.m` to test the LMI constraints given in (3.36)–(3.39) of Chapter III. To optimize γ, or to solve a smaller problem such as the short period design, the GEVP method should be used. Equations (3.36)–(3.47) of Chapter III are used to set up the GEVP in program `nshinflmi.m` and solve for the L-scales.

Specifically, the resulting scales are the L_t-scales of step 3 in the D-K-D iteration procedure. Since $P_a(s)$ for this design is regular, the LTI controller can then be found in step 4 using the Riccati-based H_∞ equations. Recall that the "uncertainty" block at this point is given by

$$\begin{bmatrix} \Theta_t & 0 & 0 \\ 0 & \Theta_t & 0 \\ 0 & 0 & \Theta_u \end{bmatrix} \tag{4.57}$$

where each Θ_t is the repeated scalar parameter block given in (4.30). Since we will be performing μ-synthesis, we must add a fictitious "uncertainty" block corresponding to the performance channels. Thus,

$$\Theta_{\text{LMI}} = \begin{bmatrix} \Theta_t & 0 & 0 & 0 \\ 0 & \Theta_t & 0 & 0 \\ 0 & 0 & \Theta_u & 0 \\ 0 & 0 & 0 & \Theta_p \end{bmatrix} \tag{4.58}$$

Prior to performing the μ-synthesis of step 6, the two Θ_t blocks must be combined into one full block. Using the notation of (4.30), Θ_t would then become a square full block of dimension $r_{2(M+h)}$. Thus,

$$\Theta_\mu = \begin{bmatrix} \Theta_t & 0 & 0 \\ 0 & \Theta_u & 0 \\ 0 & 0 & \Theta_p \end{bmatrix} \tag{4.59}$$

As explained in Chapter III, this is required in order for m_{Θ_t} to be scalar, since m_{Θ_t} must be used to normalize the M_Θ scales corresponding to the uncertainty and performance channels. Performing step 3 now gives the scales m_{Θ_t},M_{Θ_u}, and M_{Θ_p} respectively. Once normalized, the scales are fitted to transfer functions as in `dkit.m`. The resulting L_u and L_p scales then form a new D-scale. As indicated in step 2, P_a

is then scaled to form $\hat{P}_a$ and, using the original block structure, this scaled plant is used in the next iteration to solve the GEVP for the new L_t-scales.

This process is repeated in `dkdM18.m` until the scales converge and μ ceases to decrease or is less than one. As was mentioned in Chapter II, μ-synthesis is not yet implemented for real or even scalar blocks. However, we can perform *μ-analysis* on such structures. Since our time-varying parameters are known to be real, we can therefore take advantage of this to slightly reduce some of the conservativeness in the process. By combining and reorganizing the two original Θ_t blocks, we can produce two new repeated scalar real blocks defined as follows

$$\Theta_M = [\theta_M I_{r_M}] \text{ and } \Theta_h = [\theta_h I_{r_h}]. \tag{4.60}$$

Therefore,

$$\Theta_{\mu_{\text{analysis}}} = \begin{bmatrix} \Theta_M & 0 & 0 & 0 \\ 0 & \Theta_h & 0 & 0 \\ 0 & 0 & \Theta_u & 0 \\ 0 & 0 & 0 & \Theta_p \end{bmatrix} \tag{4.61}$$

This is necessary in order to guarantee that each identical real scalar block is treated identically (in other words, by grouping all the θ_M's and θ_h's together, we can ensure that they all represent the same real scalar values of θ_M and θ_h). Note that this also requires reorganizing the input/output channels of the scaled plant $\hat{P}_a$. While no D-scales are generated, this does produce a more realistic frequency response for μ. Even though the actual scales used are found using Θ_μ and the original $\hat{P}_a$, as the iterations proceed we can observe the more realistic value of μ decrease as well. The peak value of this more realistic μ is always less than or equal to that of the complex μ; as a result, we can focus on this more realistic response and stop iterating when it decreases below one (the complex μ at this point is usually still above one).

While there is no guarantee of finding the optimum, in practice the program usually converges to at least a local minimum and, as will be demonstrated in Chapter V, there is a substantial drop in γ and, therefore, highly improved robust perfomance levels compared to what could be achieved using standard H_∞ methods.

On occasion, the optimization does appear to diverge for a few iterations before resuming convergence. This may be due to the presence of a minimum within the iteration tolerance. One of the LMI solver options is the feasibility radius, and if this is too large, the resulting scales tend to be discontinuous and impossible to fit transfer functions to. This also leads to diverging behavior. If the radius is made too small, the LMI solver may not find an existing feasible solution. More details are given in [GNLC92]. As well, the LMI solver occasionally appears to search endlessly for a feasible solution or to simply "fall asleep", especially when trying to solve very large problems; whether this is due to the LMI solver, MATLAB, or the computer itself is unknown. Thankfully, this unexplained behavior is relatively uncommon, and only manifests itself when the design problem grows to about 1500 variables or more.

V. *Results and Simulations*

This chapter describes the approach taken to define the design envelope for each of the controllers, and describes the resulting short period and full longitudinal controllers. Finally, several simulations will be performed to assess their actual performance within and beyond their design envelopes.

5.1 The Short Period Controller

For the short period design, testing the feasibility of the robust stability sub-problem indicated that a stabilizing LPV controller could be found for the reduced flight envelope defined by

$$M \in [0.5, 0.95] \text{ and } h \in [5000, 35000] \text{ ft.} \tag{5.1}$$

Note that other parts of the flight envelope could also have been used to find a robustly stable envelope; the decision to focus on this portion was arbitrary. For comparison, the Riccati-based H_∞ method was used to check the feasibility of an LTI controller over the same envelope; the resulting H_∞ norm was 2.6938, indicating that no LTI controller could guarantee stability.

For the LPV controller, the norm of the gain-scheduled H_∞ problem was found to be approximately 0.93. While stability was ensured, this left very little room in which to accomodate the performance requirements and, as a result, no robustly performing controller could be found which met the objective $\gamma < 1$. Since the problem formulation is quite conservative, the solution can be expected to withstand larger deviations in M and h than it was actually designed for. For this reason, rather than decrease the weight on performance, the size of the design envelope was slightly reduced to

$$M \in [0.5, 0.95] \text{ and } h \in [5000, 30000] \text{ ft.} \tag{5.2}$$

Using this envelope, the plant $P_a(s)$ was generated as shown in Chapter IV. A check of the H_∞ norm of the unscaled problem gave $\gamma = 2.7325$, which exceeds the desired objective. (Also, testing for a robustly stabilizing LTI controller for this smaller envelope still gives an excessive norm of 2.2365.) Performing the *D-K-D* iterations gives the data shown in Table 5.1

Table 5.1 Short Period Design *D-K-D* Data

Iteration #	Number of variables	H_∞ norm	Controller order	Peak μ value	Peak μ_{analysis} value
1	480	1.5777	7	1.4920	1.4075
2	626	1.1301	13	1.1364	1.0647
3	824	1.0116	19	1.0212	0.9638
4	1436	0.9826	31	0.9805	0.9136

Note that the solution to the standard gain-scheduled H_∞ problem (without doing any μ-synthesis) is given in iteration 1 and would only reach $\gamma = 1.5777$, which points out the advantages of performing the *D-K-D* procedure.

The controller generated for either of iterations 3 or 4 can be used, since both indicate that the true value of μ is less than 1; however, the controller of iteration 3 is selected since it has a much lower order, further emphasizing the benefits of performing the μ_{analysis} step. Even so, the order of the controllers can be quite high depending on the order of the plant, the weights, and especially the order of the rational *D*-scales which are applied to each of the channels. For this design, the plant and weights contribute 7 states, whereas the L_p portion of the *D*-scale contributes 12 states (2nd order rational functions were used on the performance channels). As a result, the controller has 19 states. This is relatively high, but reasonable considering this single controller can operate over a large portion of the envelope. The order can also be reduced via model-order reduction, but that was not done for this thesis. Finally, it should be pointed out that the controller order

does not necessarily grow with each iteration; it only does so in this case since larger D-scales were required at each iteration to better approximate and decrement μ.

Due to the relatively high order of the controller, frequency response plots of the controller can be used to demonstrate that the resulting LPV controller does indeed adapt to changing values of M and h. The particular plots shown in Figure 5.1 represent the frequency response of the controller output u with respect to the commanded pitch rate q_c for various values of the parameters.

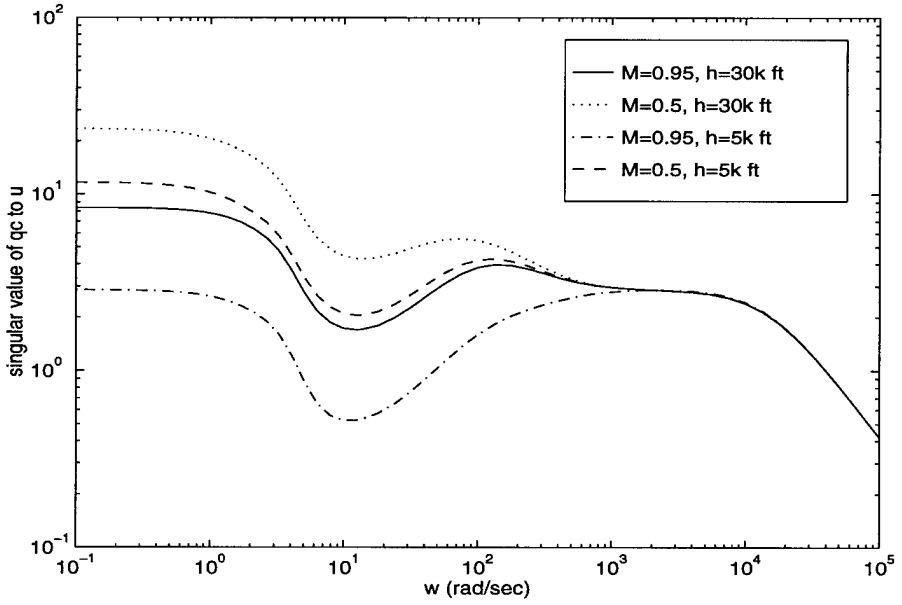


Figure 5.1 Frequency response of the short period LPV controller

5.2 The Full Longitudinal Controller

For the full longitudinal design, the feasibility of the robust stability subproblem was first tested to determine the envelope over which a stabilizing controller could be found. This resulted in a slightly smaller envelope than the one found for the short period design, indicating that the phugoid mode does increase the potential for instability. The stabilizable envelope chosen is defined by

$$M \in [0.5, 0.95] \text{ and } h \in [5000, 30000] \text{ ft.} \tag{5.3}$$

For comparison, the feasibility of an LTI controller over the same envelope was again tested; the resulting H_∞ norm was 157.41, indicating that no LTI controller could even come close to guaranteeing robust stability.

For the LPV controller, the norm of the gain-scheduled H_∞ problem was found to be approximately 0.84, thereby guaranteeing robust stability. In this case, decreasing the envelope to meet the robust performance objective was not sufficient; in fact, no sizable envelope was found which could provide robust performance using the same weights as the short period design. Also, for very small envelopes, the LPV plant becomes based on only a few trimmed points; in combination with the conservatism of the method, the results then become questionable. A better approach at that point may simply be to model the plant as LTI and add some general uncertainties, as in a standard H_∞/μ design.

In any event, in order to attempt to gain-schedule the full longitudinal model, the noise weight W_n and the performance weight W_p were decreased to the levels described in Chapter IV in order to direct more controller effort towards maintaining robust stability. Of course, as was mentioned, in doing so we are allowing some degraded performance in the form of greater error and more susceptibility to noise. After some initial trials, it was decided to attempt to gain-schedule the entire stabilizable envelope, since slightly smaller envelopes required just as large a decrease in the performance weight.

Using this envelope, the plant $P_a(s)$ was then generated for the full longitudinal design problem. A check of the H_∞ norm of the unscaled problem gave $\gamma = 157.43$, which far exceeds the desired objective. (Recall that the robustly stabilizing LTI controller for this envelope gives a norm of 157.41). Performing the D-K-D iterations gives the data shown in Table 5.2

As before, the solution to the standard gain-scheduled H_∞ problem is given in iteration 1 and reaches $\gamma = 1.1934$. The controller generated for any of the last three iterations can be used, since they all indicate that the true value of μ is less than 1;

Table 5.2 Full Longitudinal Design *D-K-D* Data

Iteration #	Number of variables	H_∞ norm	Controller order	Peak μ value	Peak μ_{analysis} value
1	934	1.1934	9	1.1934	1.0377
2	1178	1.0153	17	1.0199	0.9976
3	1112	1.0146	15	1.0210	0.9966
4	1112	1.0155	15	1.0144	0.9919

however, the controller of iteration 4 is selected since it has a lower order and slightly lower μ value. In this case, the plant and weights contribute 9 states, whereas the L_p portion of the D-scale contributes 3 states (1st order rational functions were used on the performance channels). As a result, the controller has 15 states. The order can then be further reduced via model-order reduction. Note that the order of this controller happens to be lower than that for the short period controller. This is due to the D-scales which are generated as rational approximations to the actual scales. These in turn depend on how "smooth" the LMI and $\mu_{\text{synthesis}}$ scales turn out to be as a result of the problem formulation, tolerances, feasibility radius of the LMI solver, and numerical conditioning of the problem to be solved.

Once again, frequency response plots of the controller can be used to demonstrate that the resulting LPV controller does indeed adapt to changing values of M and h. The particular plots shown in Figure 5.2 represent the frequency response of the controller output u (δ_{e_c}) with respect to the commanded pitch rate q_c for various values of the parameters.

5.3 Simulations

Several simulations were carried out to validate both the short period and full longitudinal controller designs. Unfortunately, the nonlinear aircraft/atmospheric model was not available to perform true dynamic simulations of the controllers;

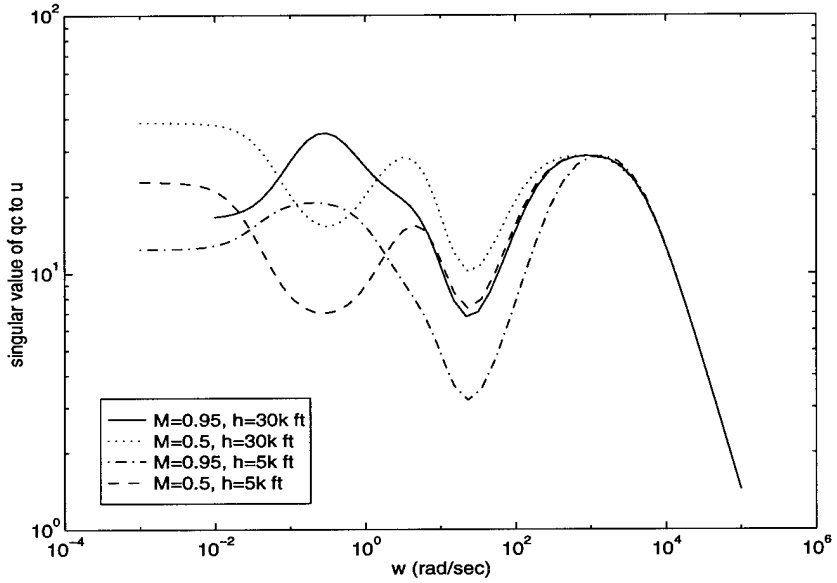


Figure 5.2 Frequency response of the full longitudinal LPV controller

therefore, MATLAB's simulation toolbox SIMULINK [SIM] was used. Two types of simulations were carried out; the first type is "static" and tests the controllers at arbitrary points within and outside their design envelopes using LTI aircraft models accurate at the selected test points. For small changes in pitch rate (which is an assumption of the linearized equations of motion), this should provide a realistic transient response as long as the state and input variables remain relatively small. The second type of simulation attempted is "dynamic", and tries to account for the changing aircraft model as the aircraft moves through the flight envelope. While this is potentially more realistic, it is still nevertheless limited by the assumptions that were made in order to linearize the equations of motion. Finally, simulations are also carried out for LTI controllers designed using standard μ-synthesis methods, in order to compare their behavior to that of the LPV controllers.

5.3.1 The "Static" Controller Simulations. Figure 5.3 illustrates the SIMULINK setup used to test the controllers at arbitrary test points in the flight envelope. At such test points, the values of M and h are known and thus the LTI

aircraft model can be generated by performing an upper LFT transformation of the high-order full longitudinal simulation LPV aircraft model with the known parameter block. A similar lower LFT transformation is performed on the controllers to convert them from LPV controllers to LTI controllers tuned to the test point. As illustrated, the simulation also includes wind and measurement noises, as well as rate and saturation limiters on elevator deflection.

For the static short period controller, the first four simulations performed correspond to the corners of the design envelope formed by the four combinations of minimum and maximum values of the parameters M and h. For clarity, these step responses are simulated without the added noises. The results are shown in Figure 5.4. A fifth test uses the parameter values at the center of the design envelope and adds the noises. The results for this simulation are shown in Figure 5.5.

The pitch-rate response displays predicted Level 1 flying qualities with excellent tracking, minimal overshoot and no oscillatory behavior. Even though the aircraft simulation model is full longitudinal, the phugoid mode does not noticeably affect the time response. These first five tests indicate that the controller should work well "statically" in at least the design envelope.

Three additional simulations were performed at arbitrary points outside the design envelope to examine the behavior of the controllers over areas in which robust performance is no longer guaranteed. For clarity, these tests were performed without the added noises. The results are shown in Figure 5.6. While all three cases are outside the robust performance envelope, note that the first one ($M = 0.45, h = 10000$ ft) is still within the robust stability envelope and does quite well, an indication that the robust performance envelope is conservative. The next case ($M = 0.35, h = 5000$ ft) is not as good, but still indicates that the robust stability envelope is conservative. Finally, the third case ($M = 0.3, h = 4000$ ft) is too far beyond the stability envelope for the controller to even maintain stability (because of this, only a few seconds are shown on the plots).

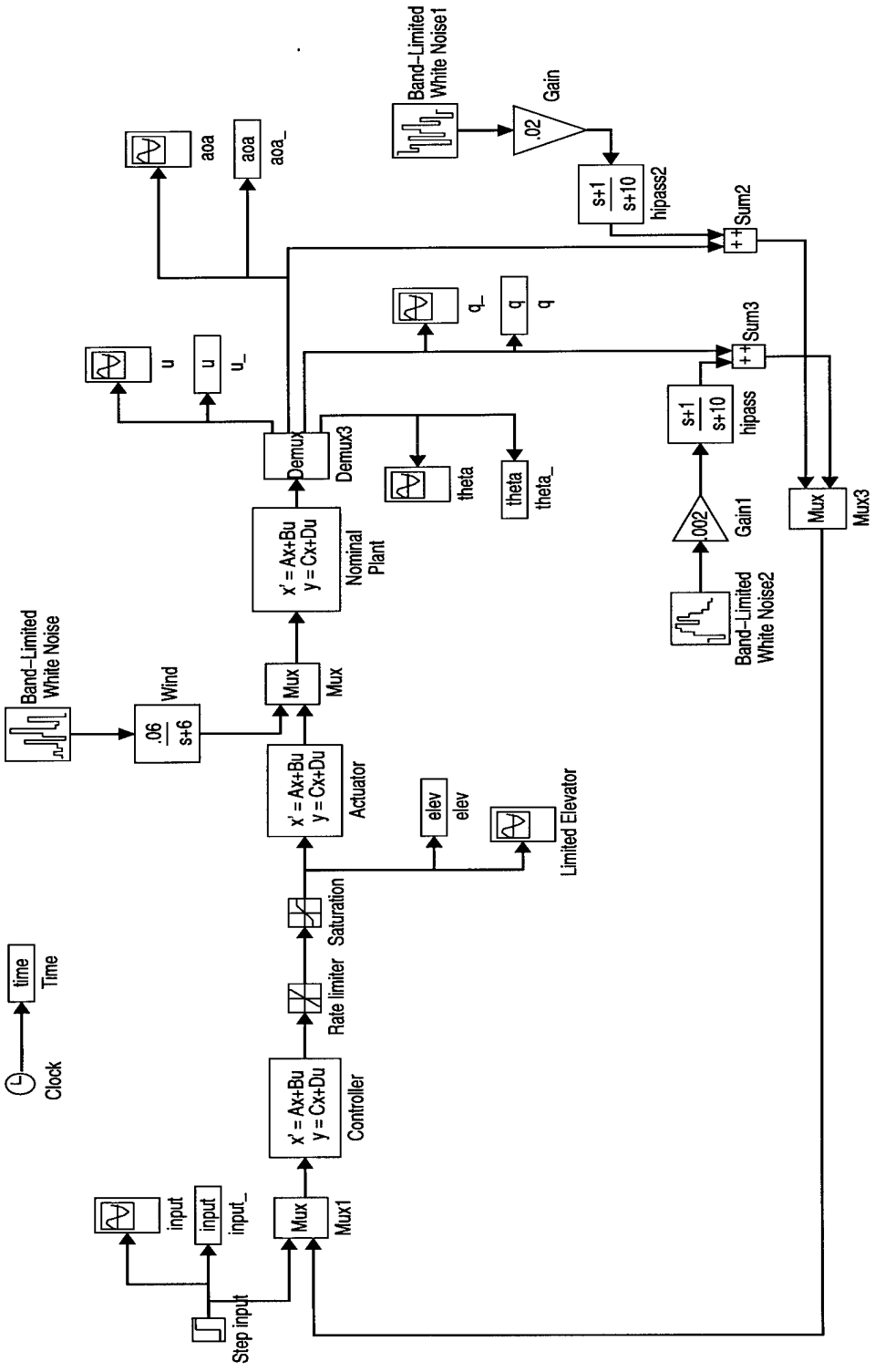


Figure 5.3 Static simulation model

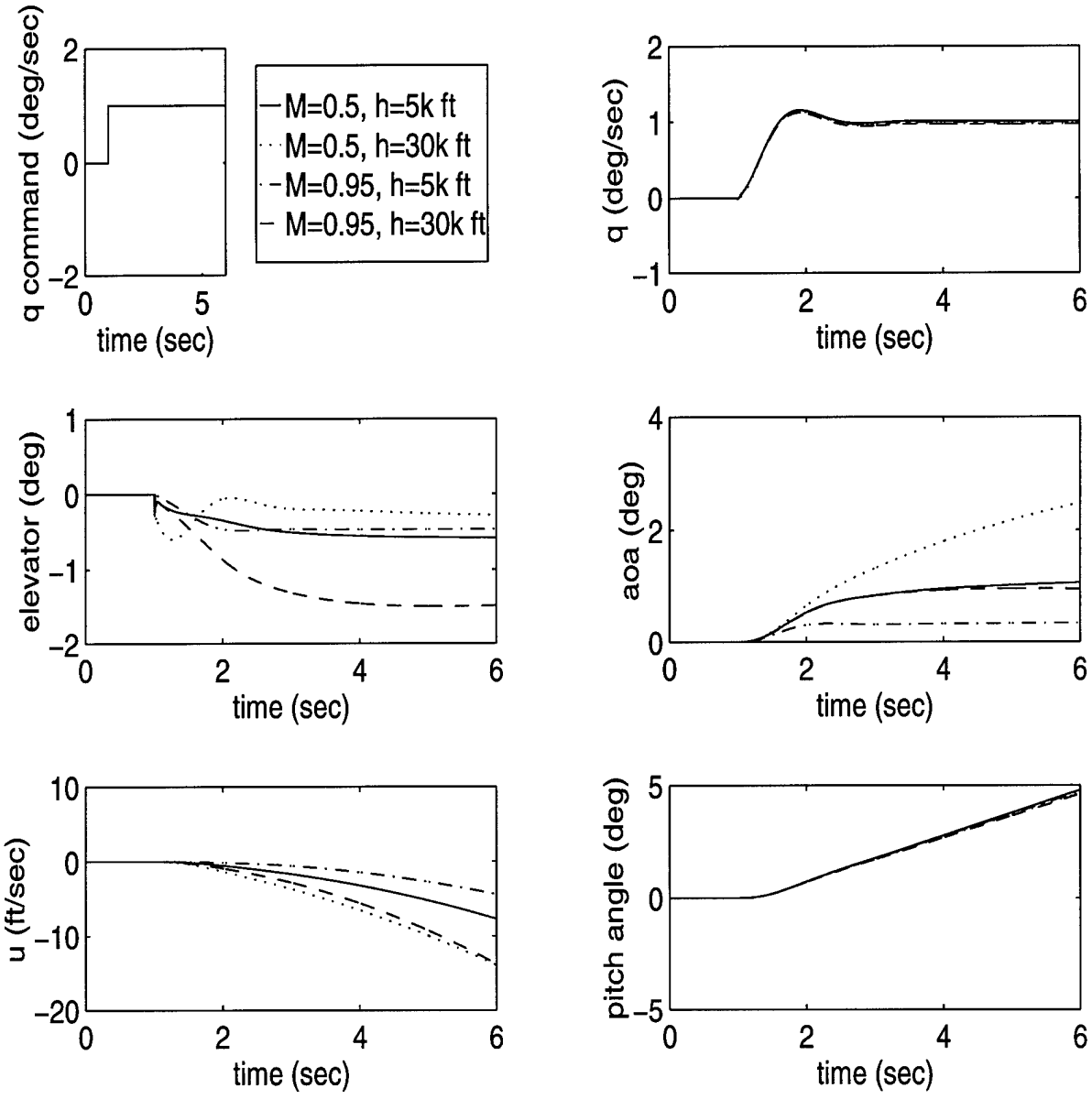


Figure 5.4 Static short period controller simulations at the corners of the design envelope

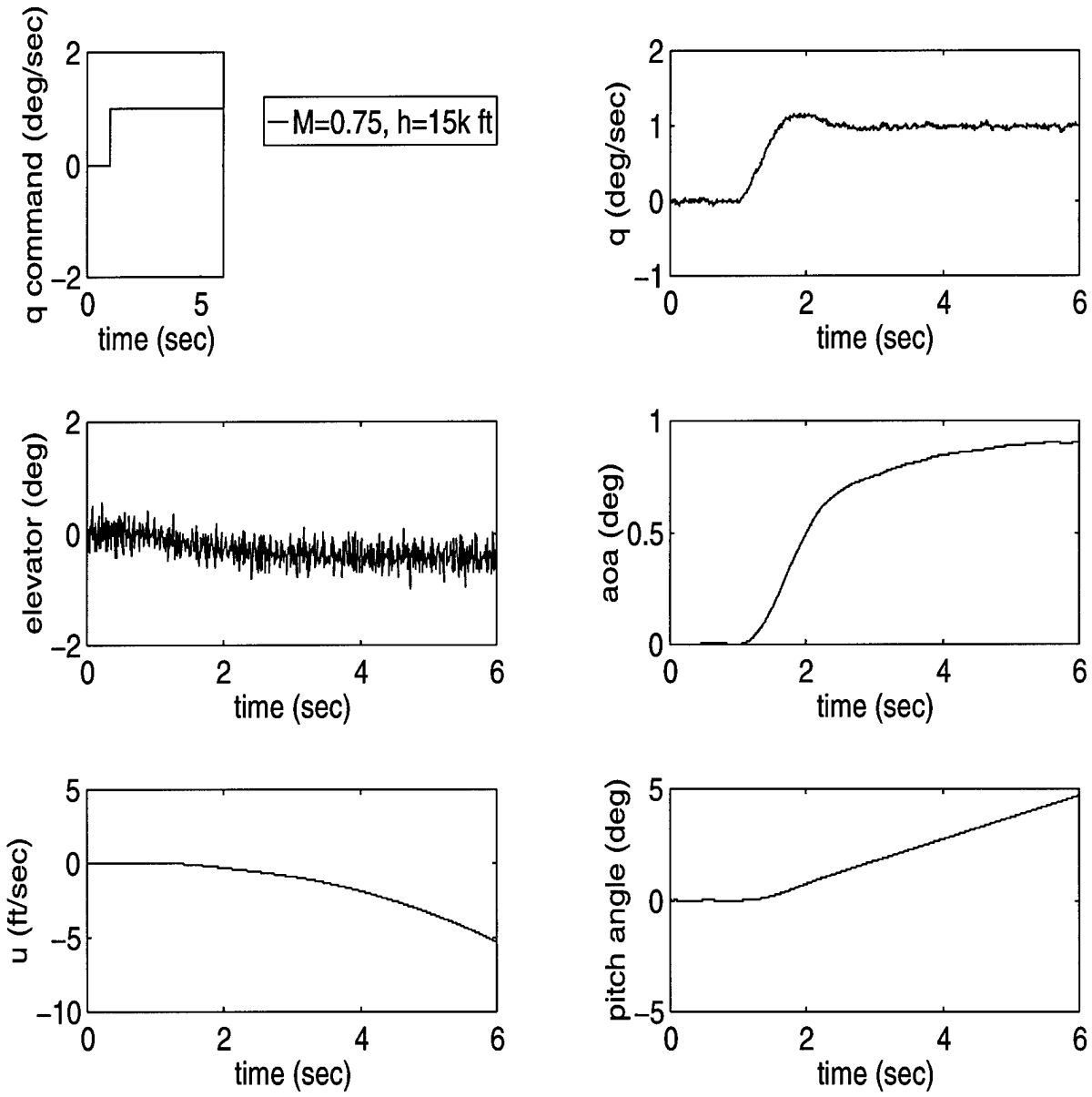


Figure 5.5 Static short period controller simulation at the center of the design envelope (with added noises)

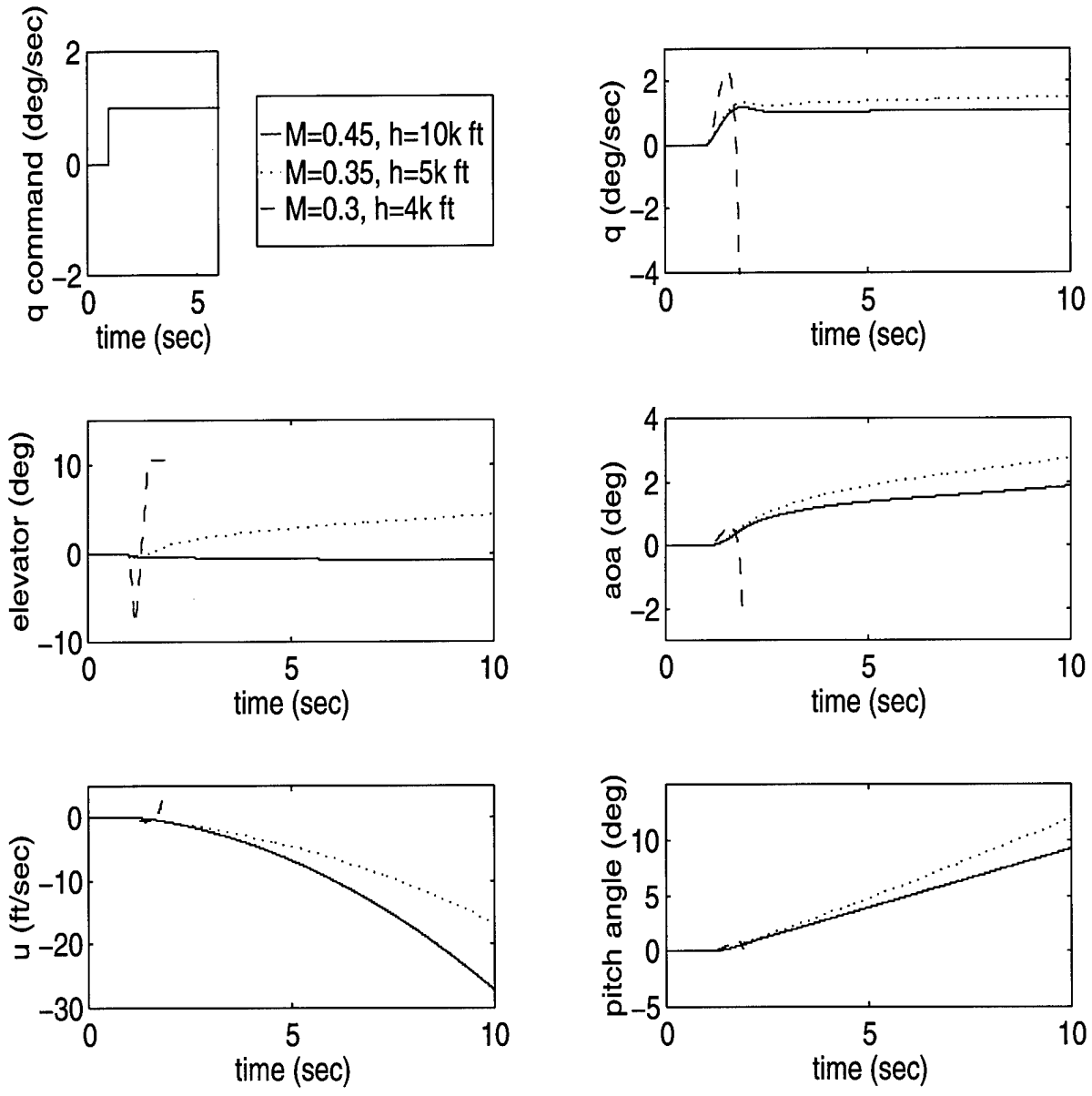


Figure 5.6 Static short period controller simulations beyond the design envelope

In the case of the full longitudinal controller design, the results are far different and are illustrated in Figure 5.7. By choosing to gain-schedule a fairly large envelope, we have had to decrease the weight on performance in order to maintain stability. Thus, the results are disappointing but not unexpected. The first three cases shown are all at the edge or within the design envelope, and stability is indeed maintained at the expense of any visible performance. As well, the last case ($M = 0.45, h = 10000$ ft) illustrates how easily stability is lost just outside the design envelope. As a result of these simulations, other designs were attempted with smaller envelopes; unfortunately, until the envelope becomes extremely small, the results are not much better. In retrospect, this appears to be largely due to the 9 additional stability derivatives and the correspondingly larger parameter block which undoubtedly increase the conservativeness of the problem. Specifically, for the previous short period design, the varying parameter blocks for the plant and controller each consisted of 3 θ_M's and 3 θ_h's, while for the full longitudinal design they consisted of 5 θ_M's and 5 θ_h's, even though both were curve-fitted with polynomials of order M^2h. Due to the Small Gain Theorem, these parameters are then treated as complex variables; this is not only very conservative, but due to the the greater number of them in the full longitudinal problem, this effectively adds much more demanding constraints to the problem. The LTI norms for the robust stability subproblems are a good indication of this; for the full longitudinal design the norm was 70 times greater than the one for the short period design, indicating how much more difficult the full longitudinal design is to control. Related to this is the fact that, in μ-synthesis alone, the parameter block forms a much larger complex full block than was the case for the short period. Unfortunately, there is little that can be done to avoid this in the current method, except trying to keep the order of the curve-fits to a minimum, and having as few parameter-dependent elements of the plant as possible.

Finally, it should also be pointed out that the ideal 2nd order reference model does not make any allowances for the phugoid mode. A more complex Level 1 performance reference model (or perhaps a parameter-varying reference model) might allow a greater weight on performance and thus improve the resulting level of performance.

5.3.2 The "Dynamic" Controller Simulations. For more realistic simulations, both the aircraft model and the controller should remain LPV and thus depend upon the changing operating conditions. As in a real implementation, this then requires updating the controller and the aircraft with the latest available values of the M and h parameters so that the LPV controller and the aircraft can adapt to the changing flight envelope. However, since M and h are not directly available from the "outside world" or a precise non-linear model of the aircraft/atmosphere, we can approximate them using the states of the aircraft model. This can be achieved at each simulation time-step i using the following relationships [Bla91],[BS89]:

$$\begin{aligned} U_i &= U_{i-1} + u_i \\ \Delta h_i &\approx \tfrac{U_i}{s}(\theta_i - \alpha_i)\tfrac{\pi}{180} \\ h_i &= h_{i-1} + \Delta h_i \\ T_i &= 518.67 - 0.003565 h_i \\ M_i &= \frac{U_i}{\sqrt{1.4 \times 1716.16 T_i}} \end{aligned} \tag{5.4}$$

where U is the total aircraft velocity, u is the perturbation velocity, θ is the perturbation pitch angle, α is the perturbation angle of attack, Δh is the change in altitude, and T is the ambient temperature in degrees Rankin. Note that u, θ, and α are available as state variables. Since the trimmed conditions were given in degrees, a conversion to radians is required to obtain Δh. Also, by choosing some M_0 and h_0 as the initial position in the flight envelope, we can calculate U_0 by using the last

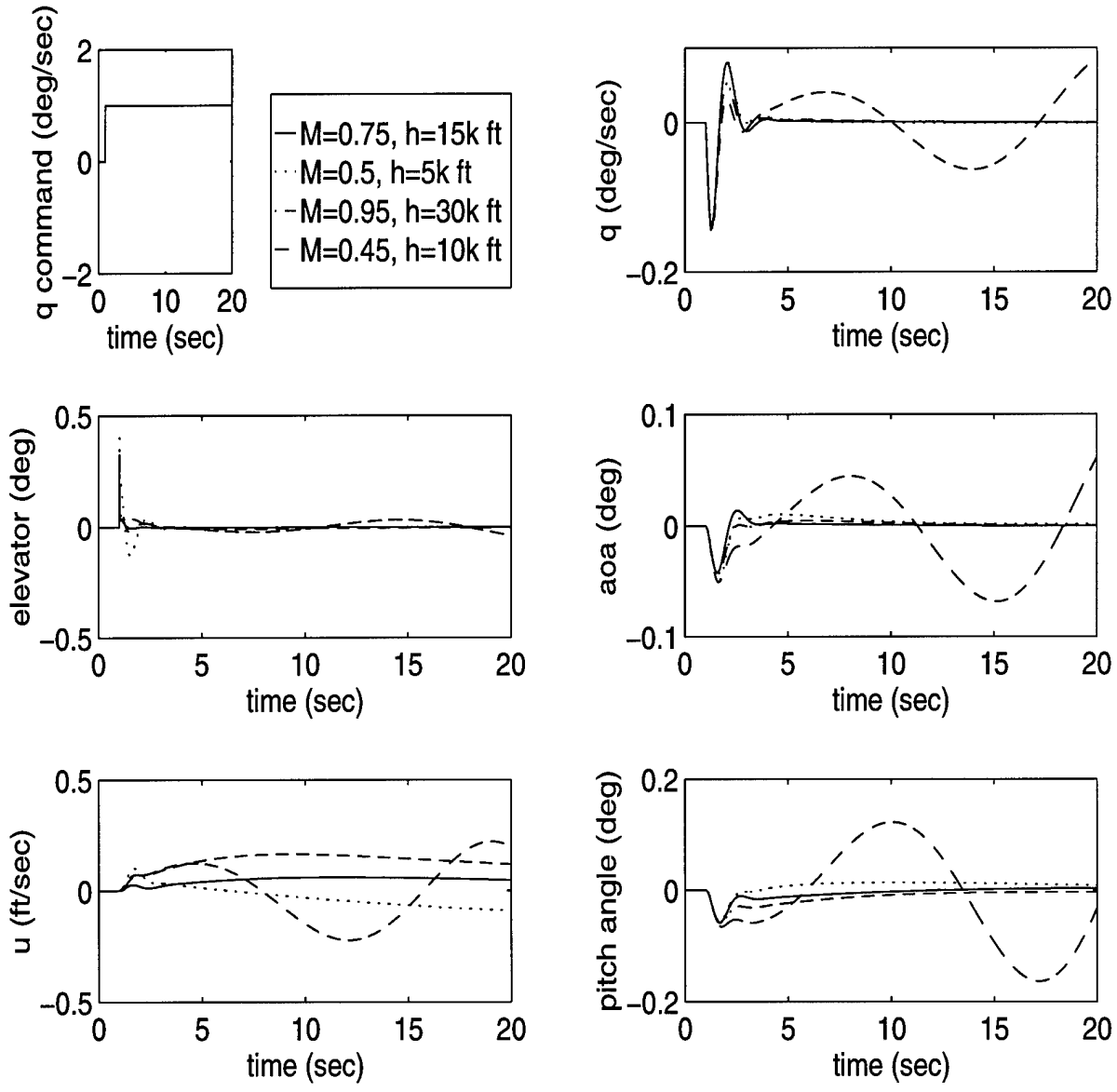


Figure 5.7 Static full longitudinal controller simulations

two equations. Finally, the equation for T_i is only valid for up to 36000 feet altitude; this is sufficient for all the test cases.

In a linear simulation, the state space models of the controller and plant are generally fixed, while the states and input/output channels vary with time. This leads naturally to think of M and h as the output of a block performing the functions given in (5.4), and whose inputs are the states u, θ, and α. The outputs would then be normalized and, since they form the parameter block, be fed back to both the aircraft model and the controller model. An example of this for the short period controller is illustrated in Figures 5.8 and 5.9. Unfortunately, this does not work in SIMULINK due to the unusually high number of algebraic loops which are created and would have to be resolved at each time-step iteration.

Fortunately, SIMULINK allows the user to define his own model blocks using what are called *S-functions*. This technique was used to define two new state-space blocks whose state-space matricies could be updated at each iteration. These new blocks, called `LPVG` and `LPVK`, are used to update the aircraft model and controller, respectively, at each time-step in the simulation. The details of the new state-space blocks are given in programs `LPVG.m` and `LPVK.m` of Appendix D. The final SIMULINK setup is given in Figure 5.10 (the `uat2Mh` functional block is identical to that in Figure 5.9).

Two tests using the short period controller will be performed using this dynamic simulation. It should be emphasized that this is only an approximation to a full nonlinear aircraft/atmospheric model and, as such, is not meant to exactly duplicate the behavior of the aircraft. There are several aircraft attributes (such as added drag and loss of lift) which are lost when only steady level flight conditions form the basis of a simulation model. Still, the aircraft simulation model appears to behave appropriately when compared to actual published flight trajectories of similar fighter aircraft.

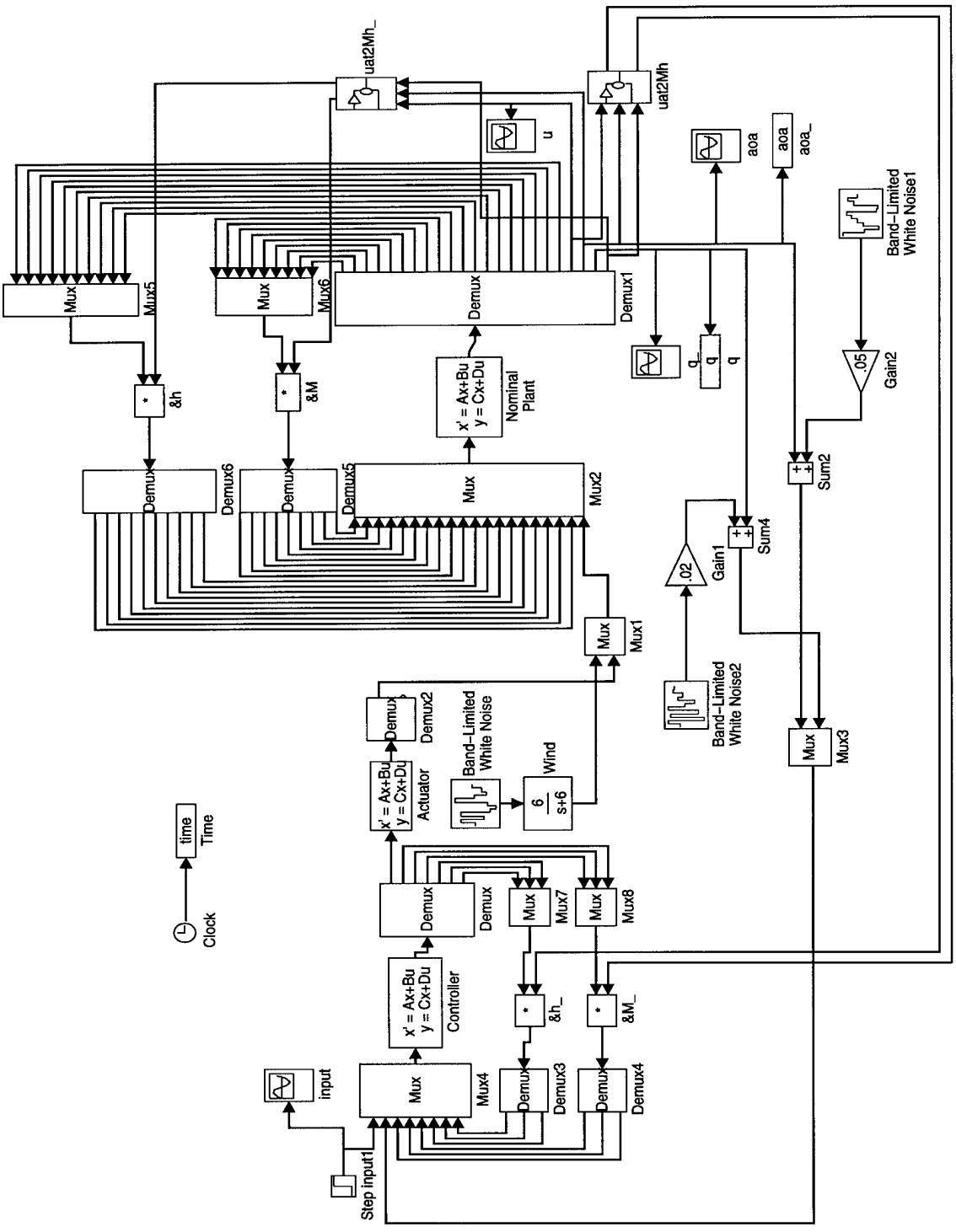


Figure 5.8 Possible dynamic LPV simulation model

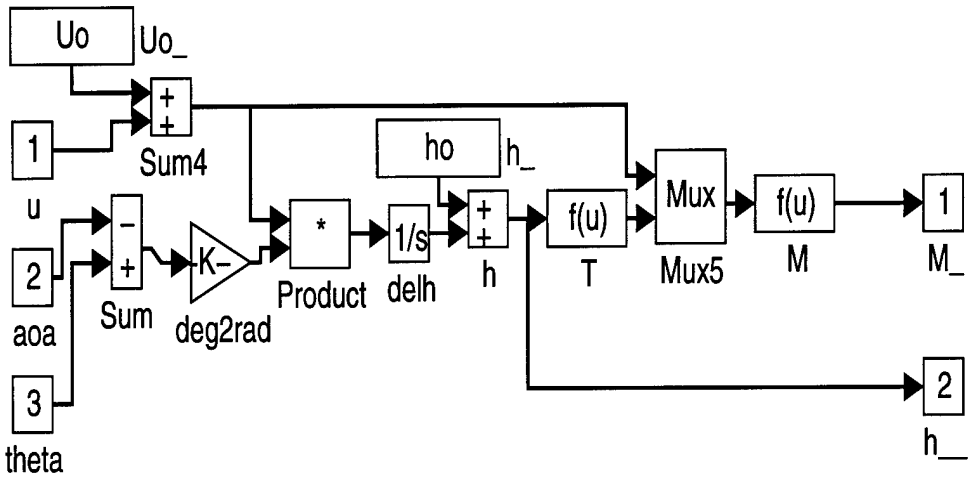


Figure 5.9 `uat2Mh` functional block (converts states u, α, θ to parameters M, h)

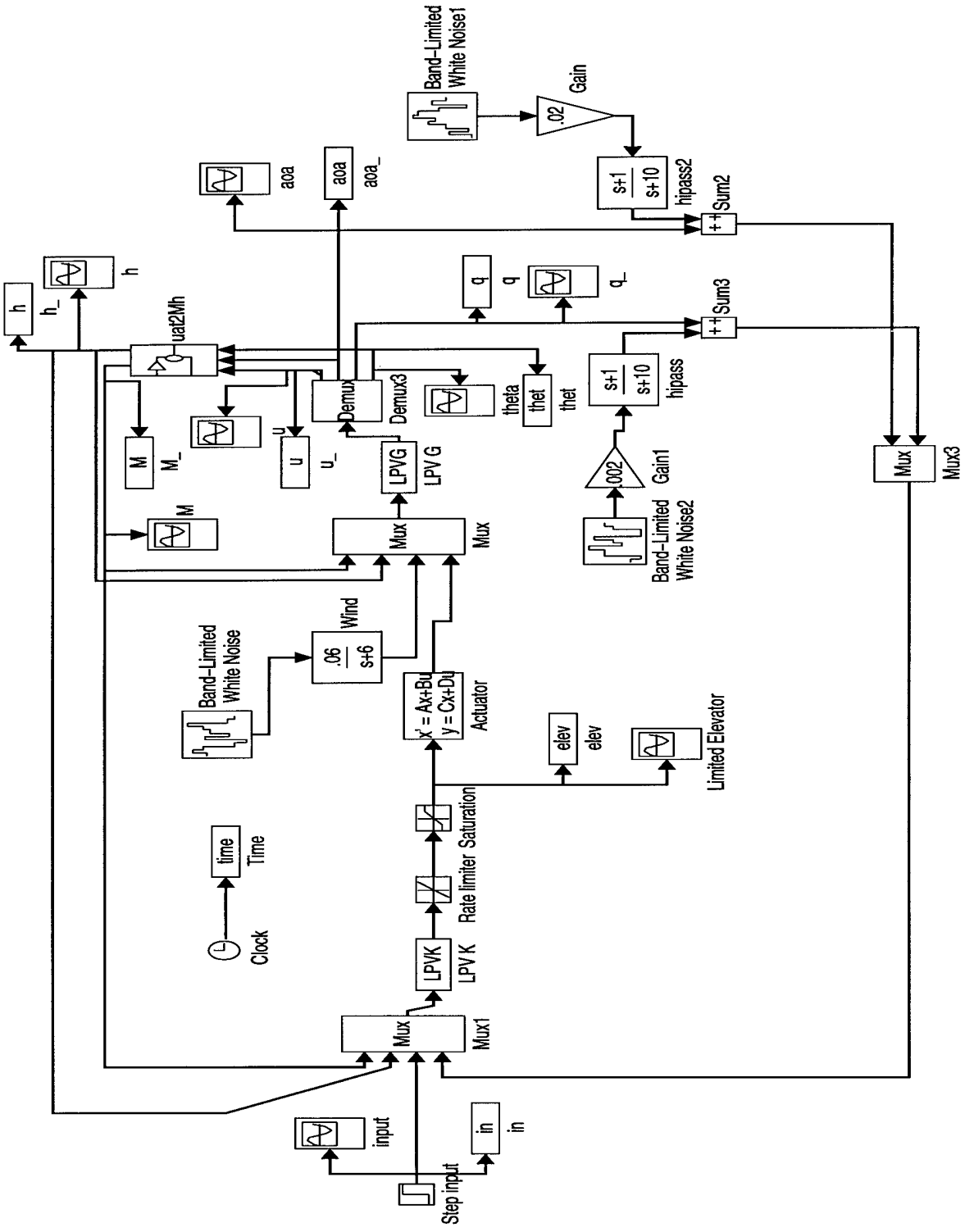


Figure 5.10 Dynamic LPV simulation model

Several pitch-rate commands were chosen to test as much of the envelope as possible without having access to a thrust input. The resulting dynamic simulations are illustrated in Figures 5.11 to 5.22. The first four figures illustrate how the controller reacts to a doublet pitch-rate command; while the next four illustrate how it reacts to the opposite pitch-rate command with added noises. The last four figures test the uppermost part of the envelope using basic pitch-rate command pulses. The results for these dynamic tests are once again quite good with excellent tracking and noise response, indicating that Level 1 flying qualities can be expected using this controller within the design envelope. As well, even though the commands are held for relatively long periods of time, the phugoid has no noticeable effect on the response. This suggests that this short period design may well be sufficient for adequate pitch-rate control, and that a full longitudinal design may not be necessary.

5.4 A Comparison with LTI μ-Synthesis Controllers

It has already been demonstrated in Chapter IV that the LPV controller stabilizes a much larger envelope than any LTI controller could. It would also be interesting to compare the performance of optimal LTI controllers to that of the LPV controllers. The best way to do this is by examining their H_∞ norms and time responses.

Using exactly the same weights and LPV design plant $P(s)$ developed in Chapter IV and specific parameter values of M and h, we can obtain a plant P_f by the upper LFT transformation

$$P_f(s) = F_u(P, \Theta). \tag{5.5}$$

This is now an LTI plant, and standard Riccati-based methods can then be used to find an optimal H_∞ controller. For a fair comparison, the LTI design model should also include some model uncertainty in order to account for the varying nature of the plant; otherwise, it will have excellent performance only at the design point.

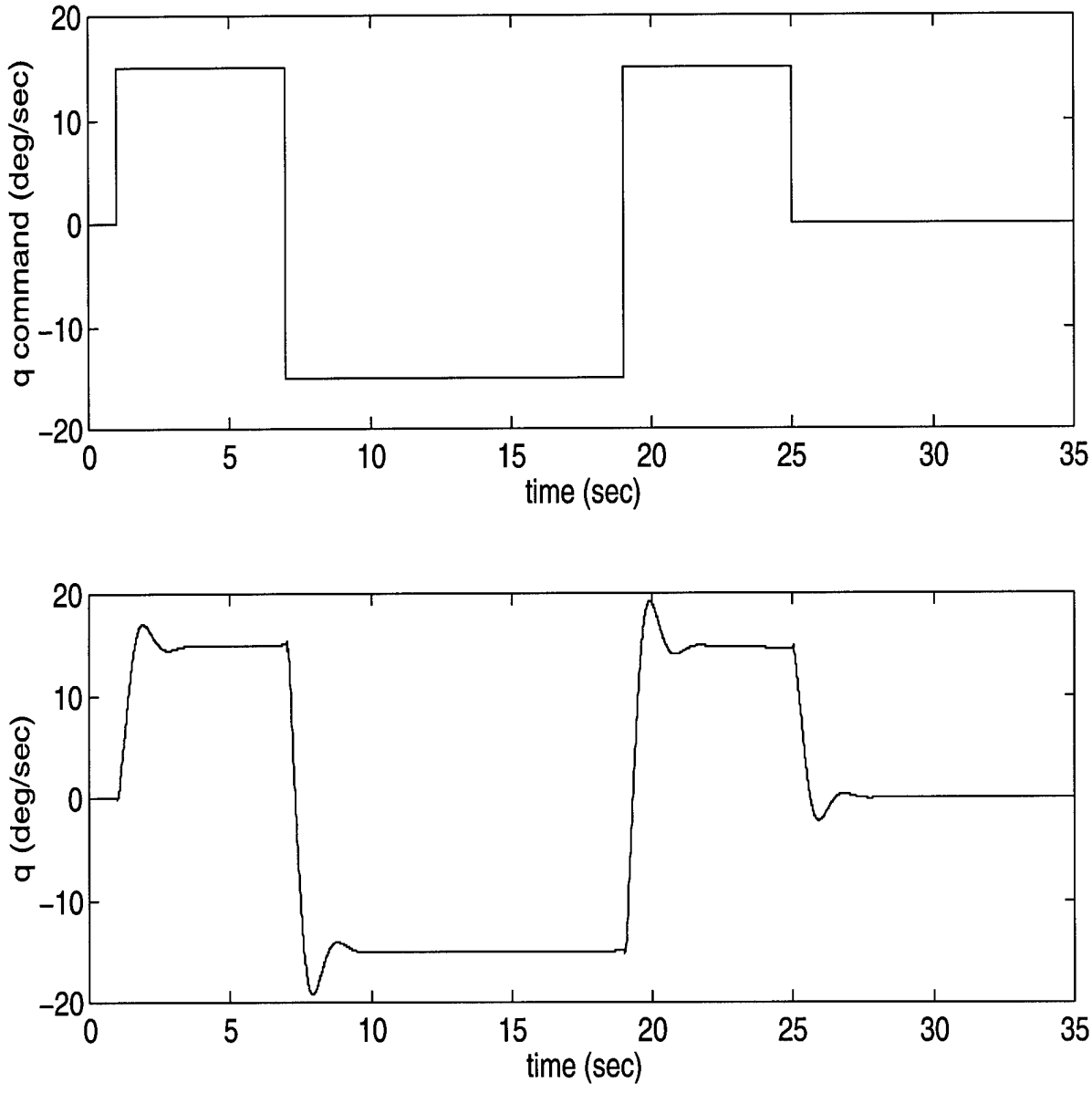


Figure 5.11 Dynamic response to a positive doublet pitch-rate command: q_c, q

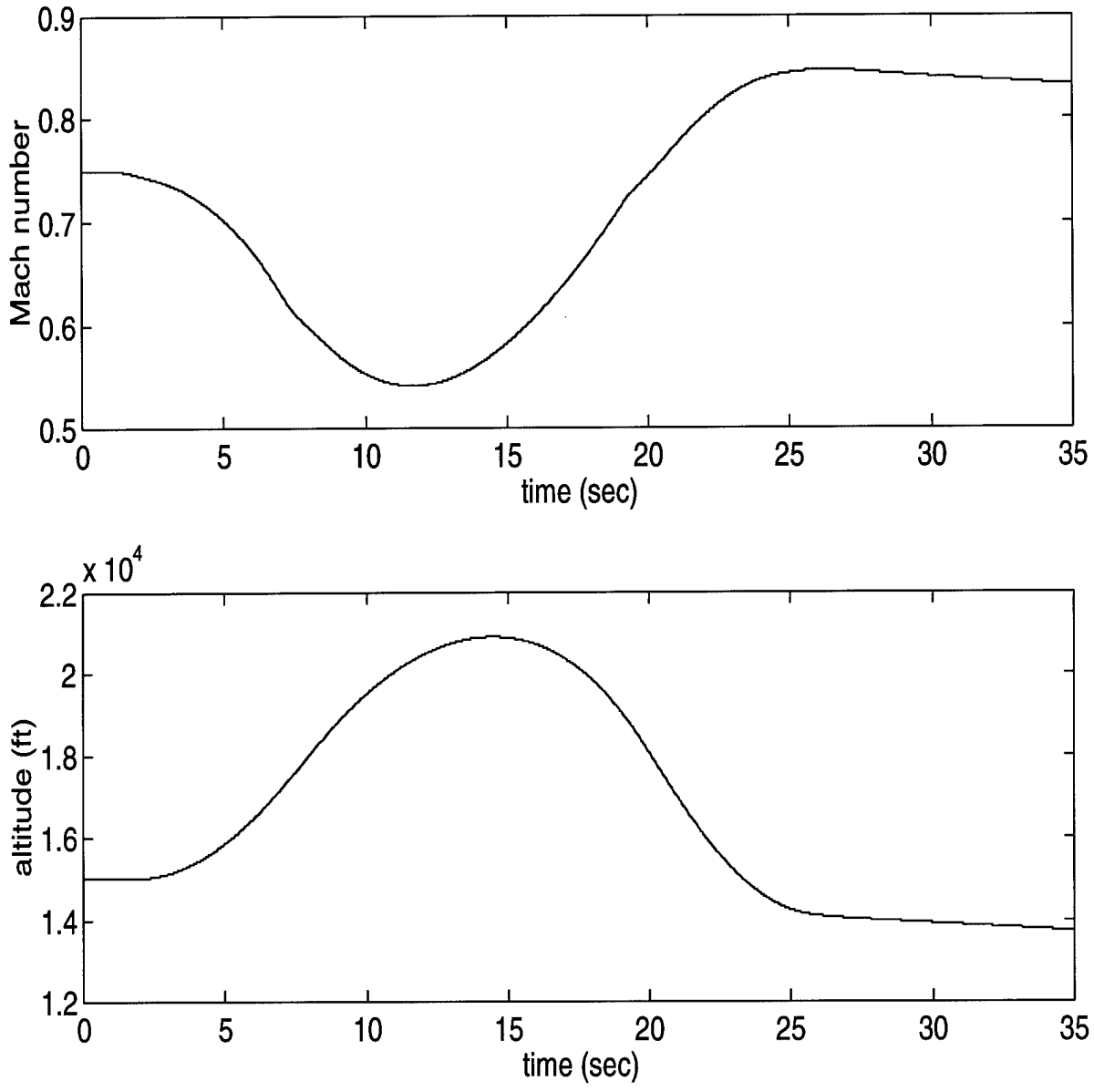


Figure 5.12 Dynamic response to a positive doublet pitch-rate command: M, h

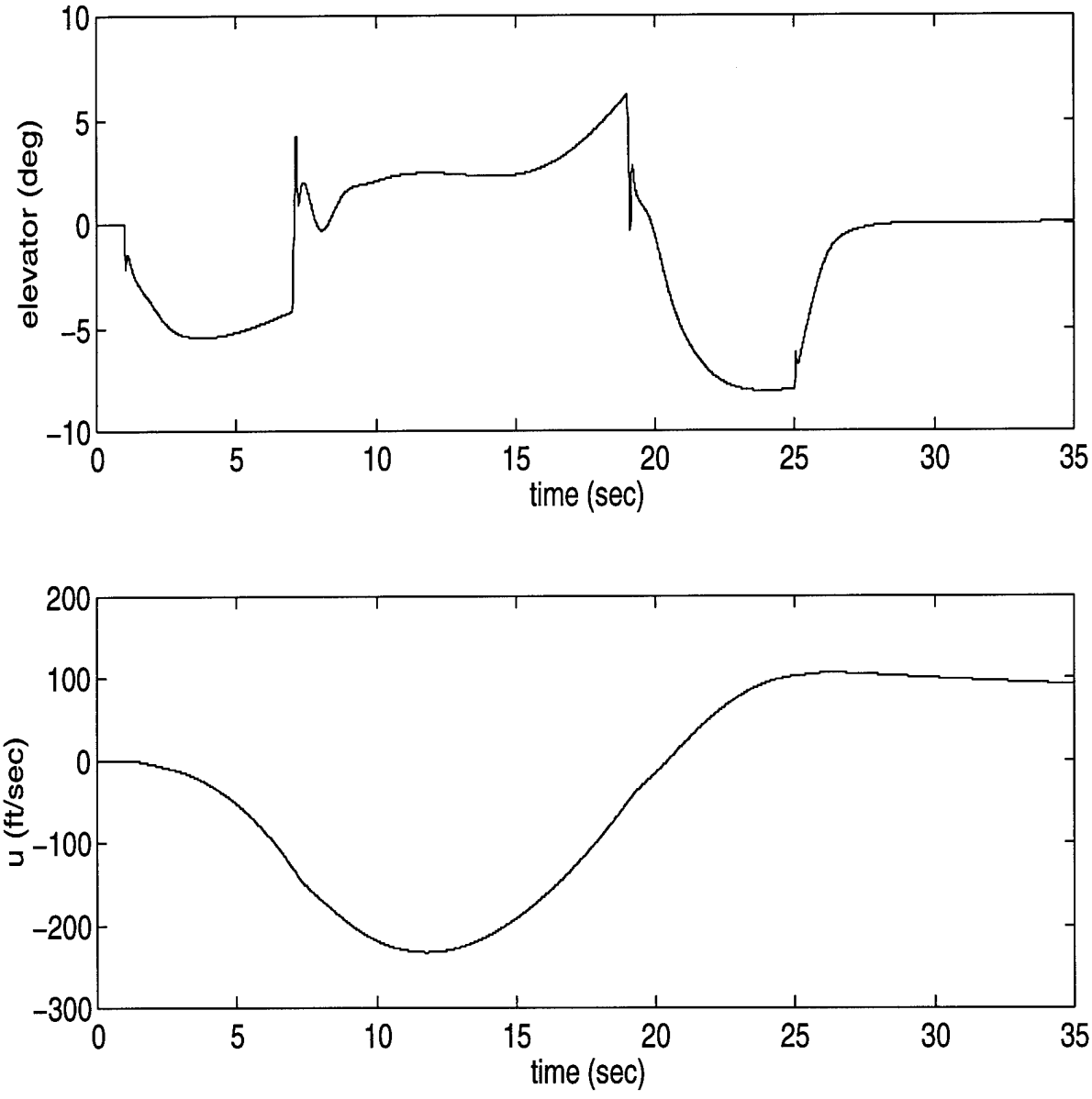


Figure 5.13 Dynamic response to a positive doublet pitch-rate command: δ_e, u

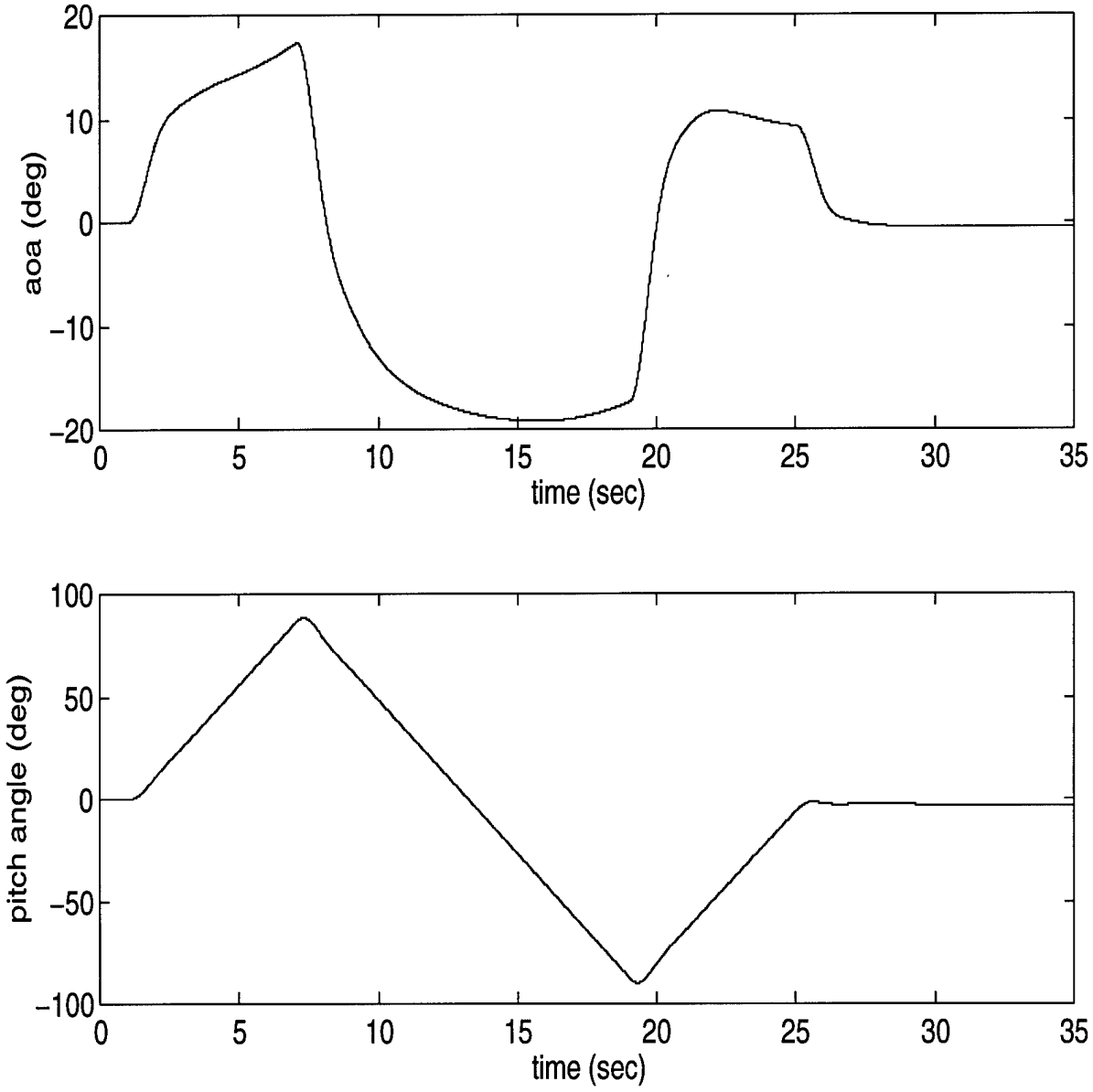


Figure 5.14 Dynamic response to a positive doublet pitch-rate command: α, θ

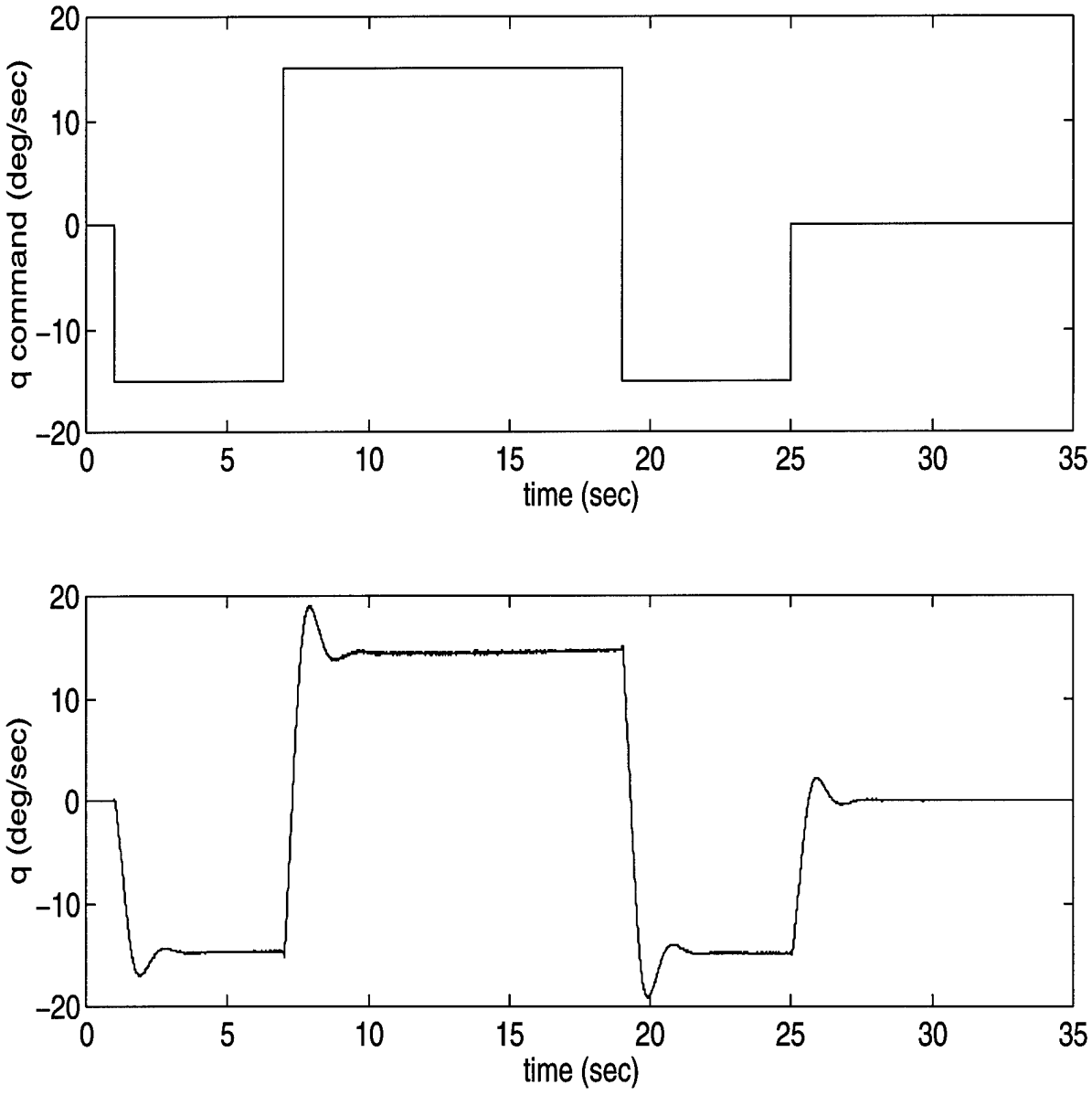


Figure 5.15 Dynamic response to a negative doublet pitch-rate command with added noises: q_c, q

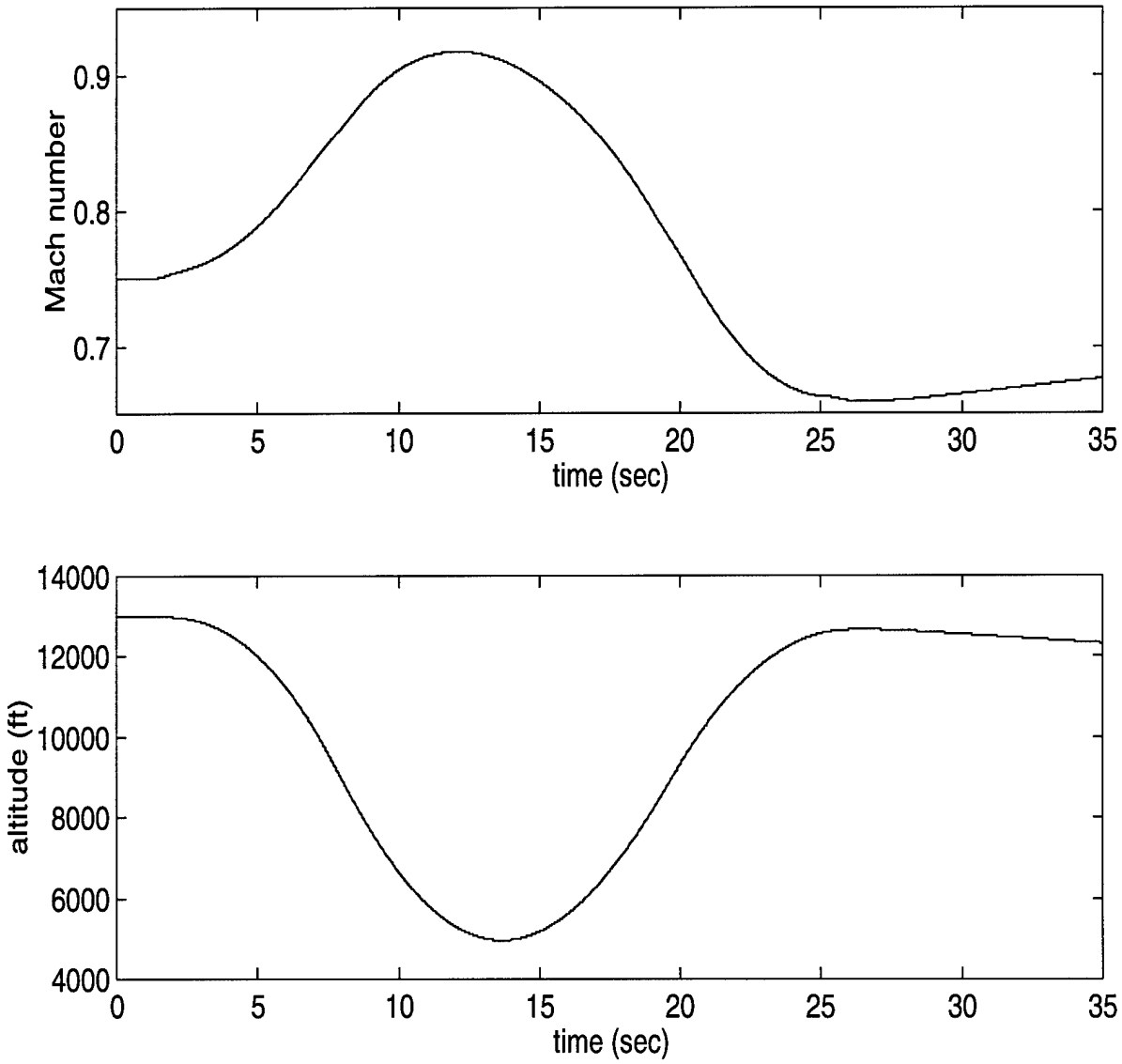


Figure 5.16 Dynamic response to a negative doublet pitch-rate command with added noises: M, h

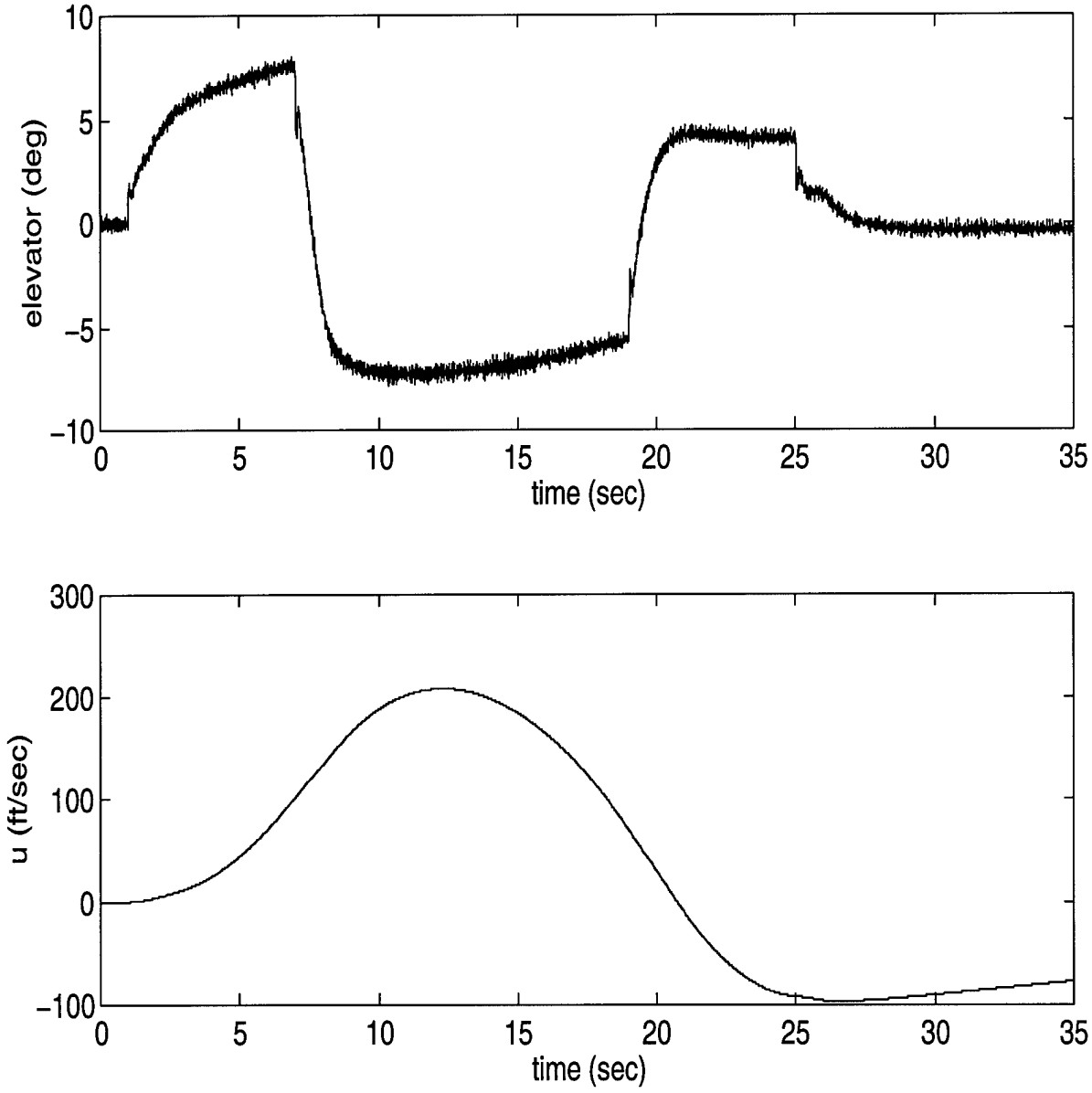


Figure 5.17 Dynamic response to a negative doublet pitch-rate command with added noises: δ_e, u

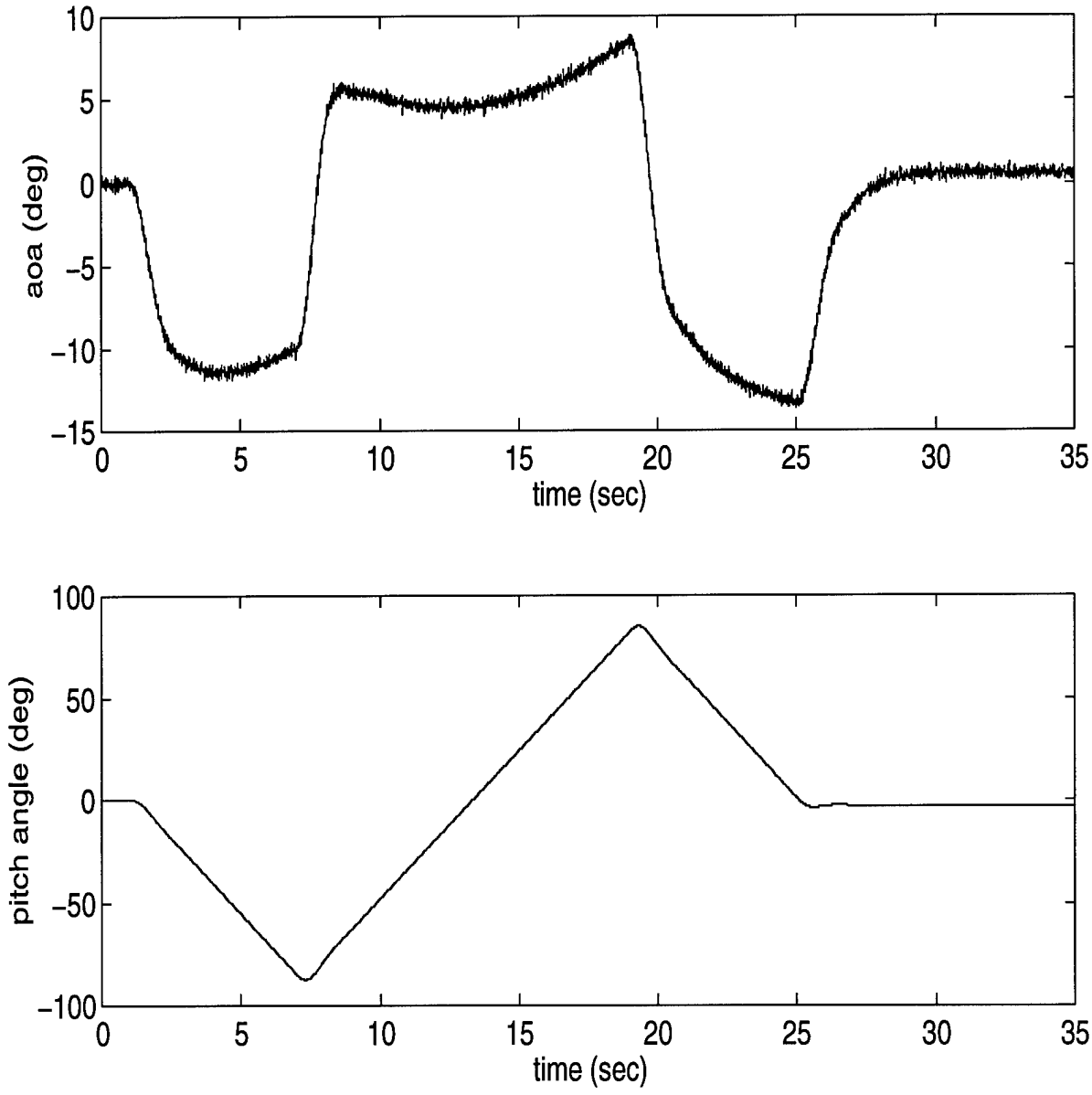


Figure 5.18 Dynamic response to a negative doublet pitch-rate command with added noises: α, θ

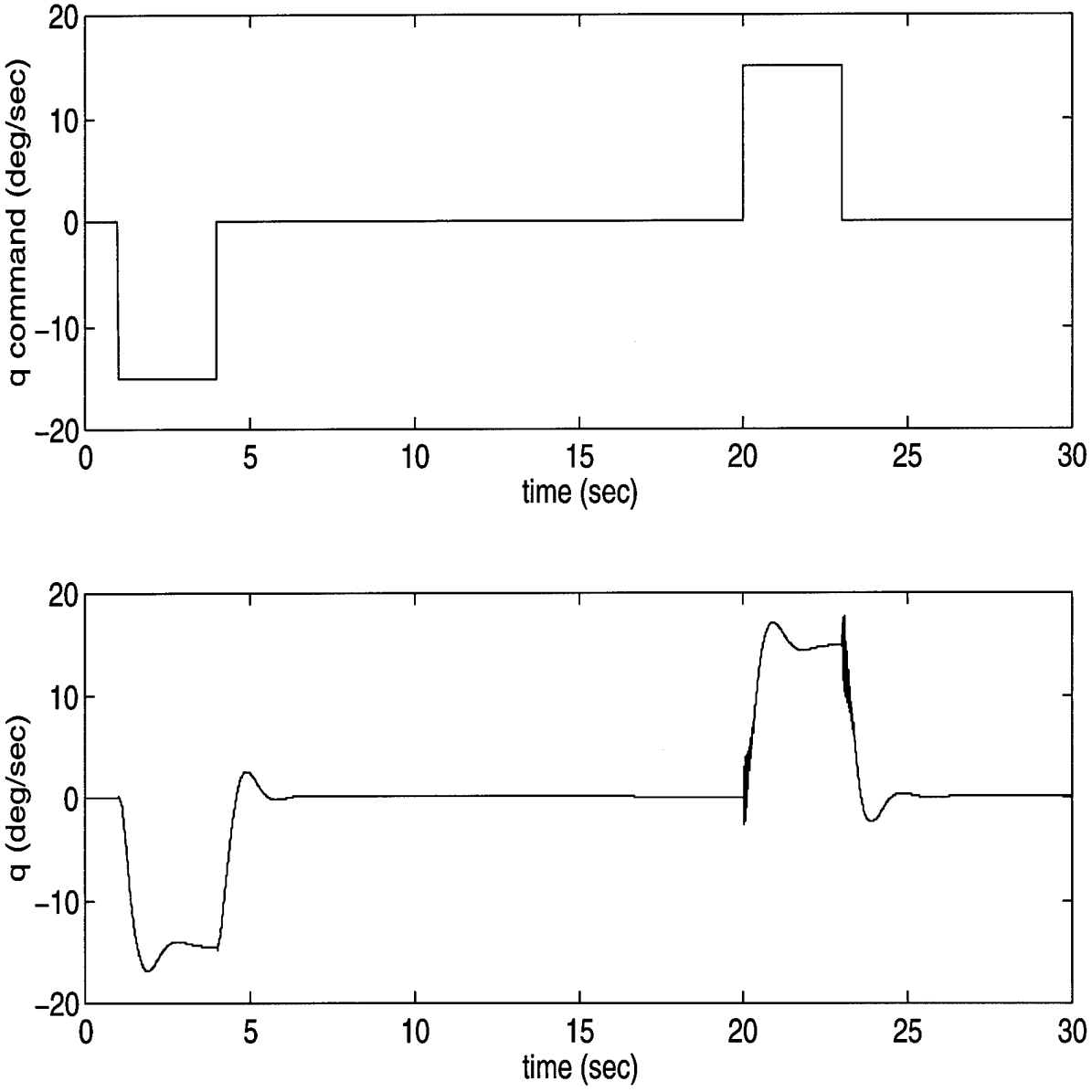


Figure 5.19 Dynamic response to pitch-rate command pulses: q_c, q

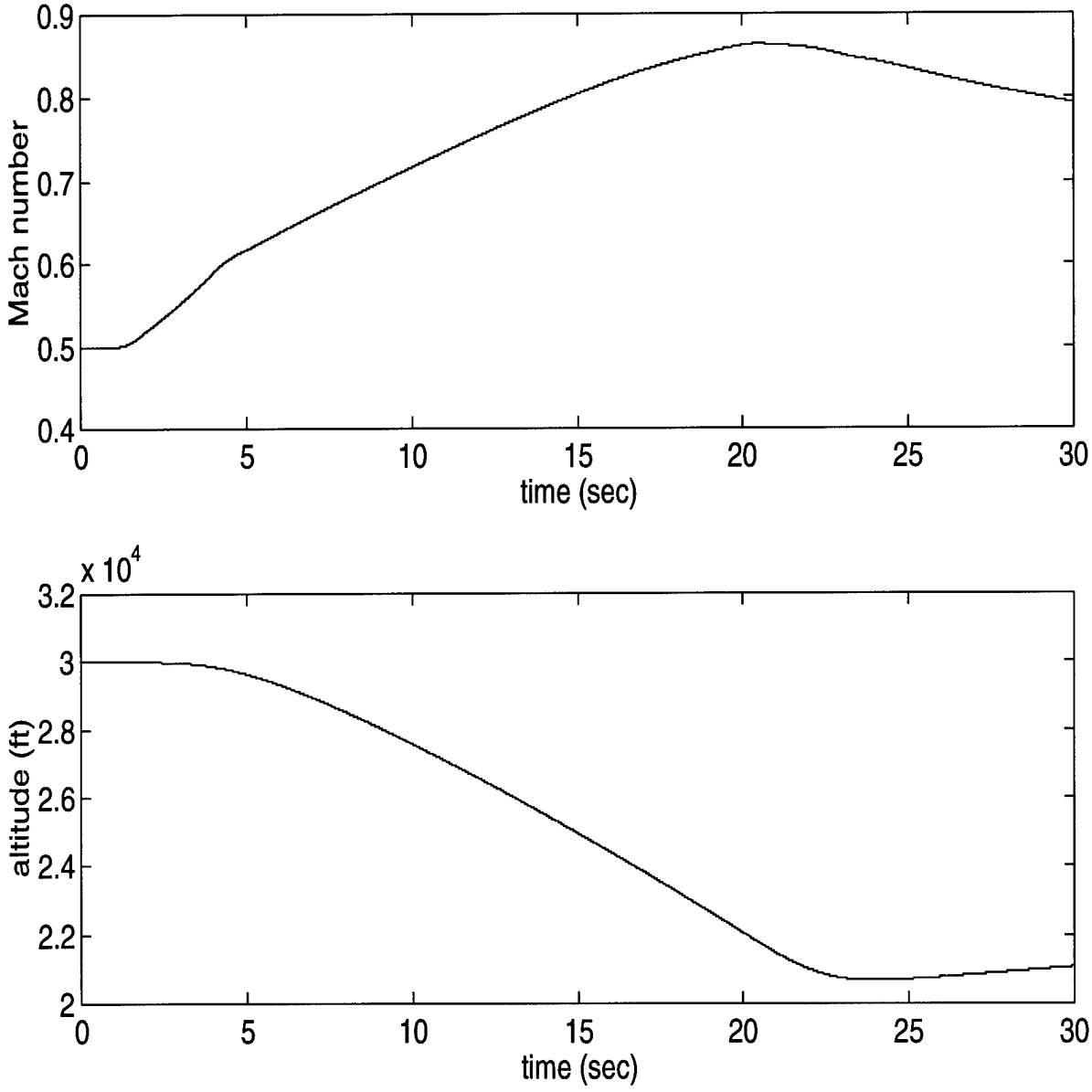


Figure 5.20 Dynamic response to pitch-rate command pulses: M, h

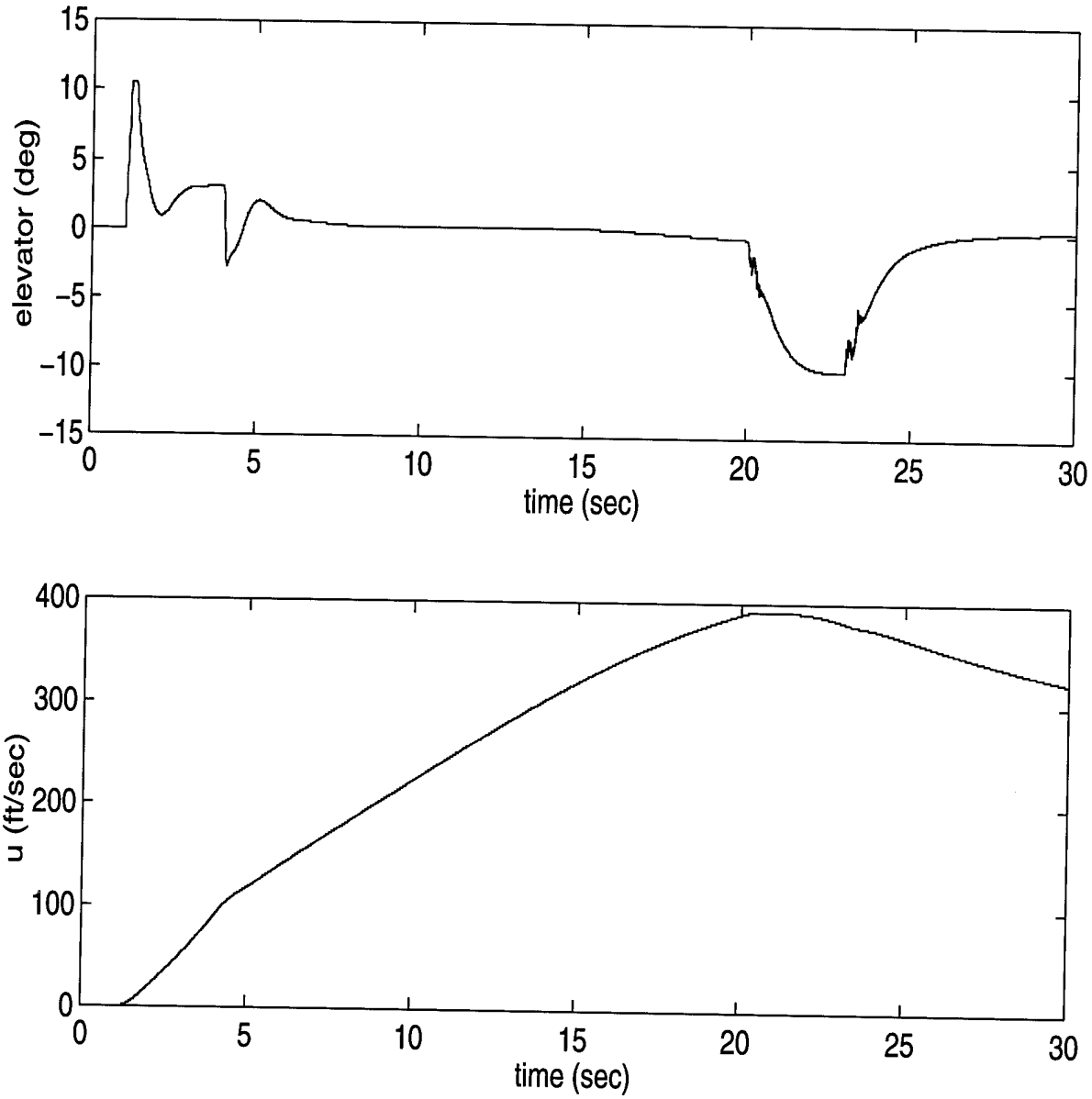


Figure 5.21 Dynamic response to pitch-rate command pulses: δ_e, u

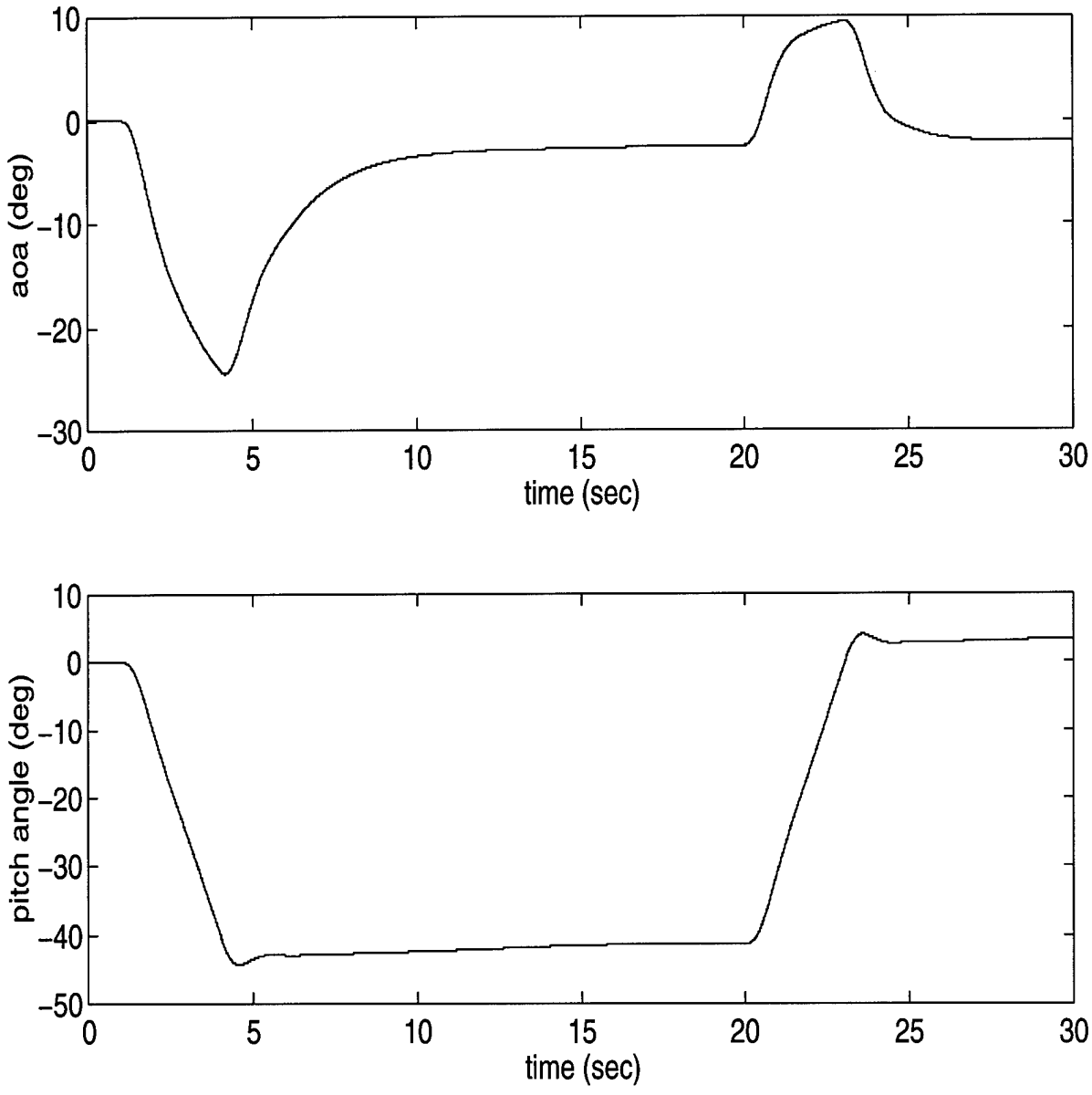


Figure 5.22 Dynamic response to pitch-rate command pulses: α, θ

The size of the uncertainty chosen will affect the tradeoff between performance and stability; as a result, several iterative designs should be performed until the right mix is achieved. Designs along these lines revealed that no LTI controller could guarantee robust performance for a sizable portion of the design envelope; therefore, for the purpose of this comparison, the multiplicative uncertainty already in the design model was used in combination with μ-synthesis to yield several controllers of different orders optimized for the center of the LPV design envelopes. The controllers which gave the most acceptable performance throughout the LPV design envelope when simulated against the full longitudinal simulation model were then used for comparisons.

For the short period design, the resulting robust performance controller resulted in a peak μ value of 0.326, while the nominal performance H_∞ norm was 0.2076; both indicate that robust performance for a sizable envelope should not be difficult to achieve and that a larger uncertainty could be addressed. On the other hand, the full longitudinal design had a peak μ value of 0.982 and nominal performance H_∞ norm of 0.986; this emphasizes the fact that, using the desired reference model, robust performance will likely not be achievable, since there is very little room to satisfy both nominal performance and robust stability for even the small multiplicative uncertainty. This was demonstrated in the full longitudinal LPV design, where we could find a controller to maintain stability only at the expense of performance.

Static step responses for the LTI controllers when tested on the full longitudinal simulation model are shown in Figures 5.23 and 5.24. As expected, the full longitudinal LTI controller, which is based on a plant similar to that of the simulation model, does well at the design point ($M = 0.75, h = 15000$ ft) but off-design points show degraded and/or slowly destabilizing performance. This also supports the fact that the LPV controller has to focus the majority of its efforts on stability, as was found earlier.

The short period LTI controller exhibits similar behavior when simulated with the full longitudinal simulation model; however, it was designed using the short period model. Figure 5.25 shows the response it would have at the center and at corners of the LPV design envelope when tested with its design plant. As expected, at the center of the envelope (its design point) the controller has excellent performance; elsewhere, it appears to be stable over a large portion of the envelope, though at the cost of some tracking error. Nevertheless, Figure 5.25 supports the fact that, unlike the full longitudinal LPV controller, the short period LPV controller does not have to spend as much effort to guarantee robust stability, and can thus devote more effort to achieving perfomance. Note also that several LTI/μ controllers could give much better performance when simulated with the short period plant; however, this controller was chosen since it gave better performance when tested with the full longitudinal simulation model.

Finally, this short period LTI controller was used in a dynamic simulation to compare its performance to that of the short period LPV controller. Its response to the doublet pitch-rate command is shown in Figures 5.26 to 5.29, while that of the LPV controller is given in Figures 5.11 to 5.14. It does surprisingly well, although it does exhibit some diverging behavior after extended periods of time, slightly more overshoot, and tracking is not quite as good. While the LPV controller definitely performs better, this does illustrate that a well-designed LTI controller may perform nearly or just as well as an LPV controller. However, since it does directly design a varying controller, the LPV design method should, in general, perform better. The extent to which it does, in all likelihood, depends largely on the extent the plant varies and the rate at which its parameters change. For linear systems, these are aspects which only this or similar LPV approaches to gain-scheduling can hope to address.

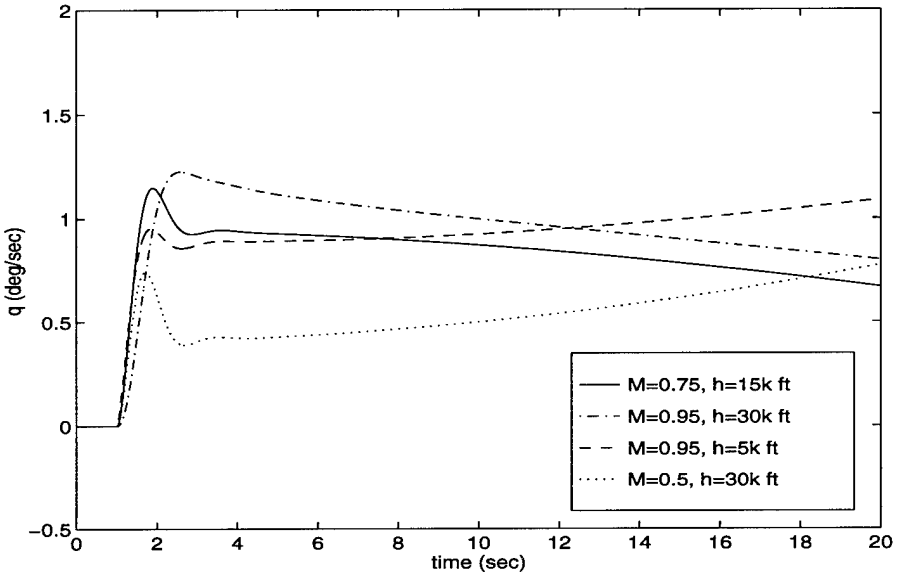


Figure 5.23 Step responses using short period LTI controller

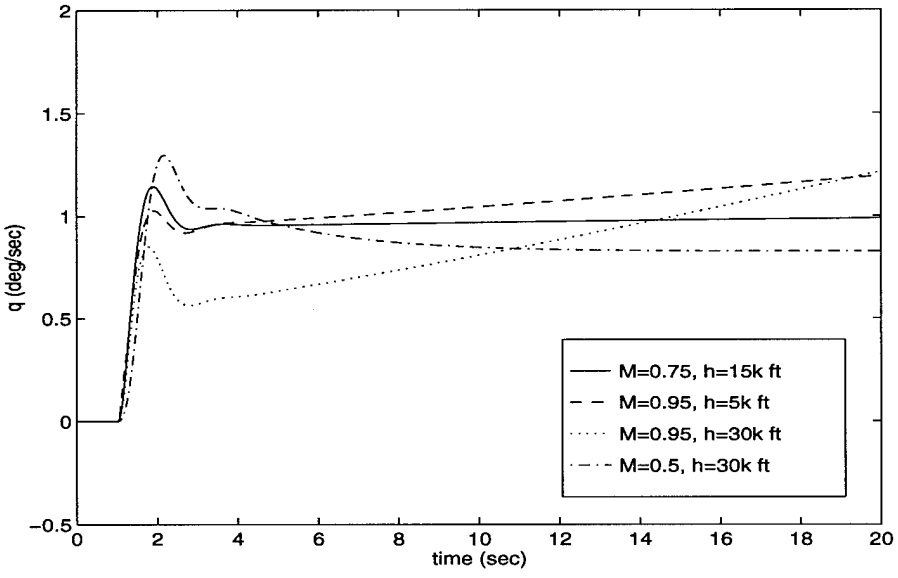


Figure 5.24 Step responses using full longitudinal LTI controller

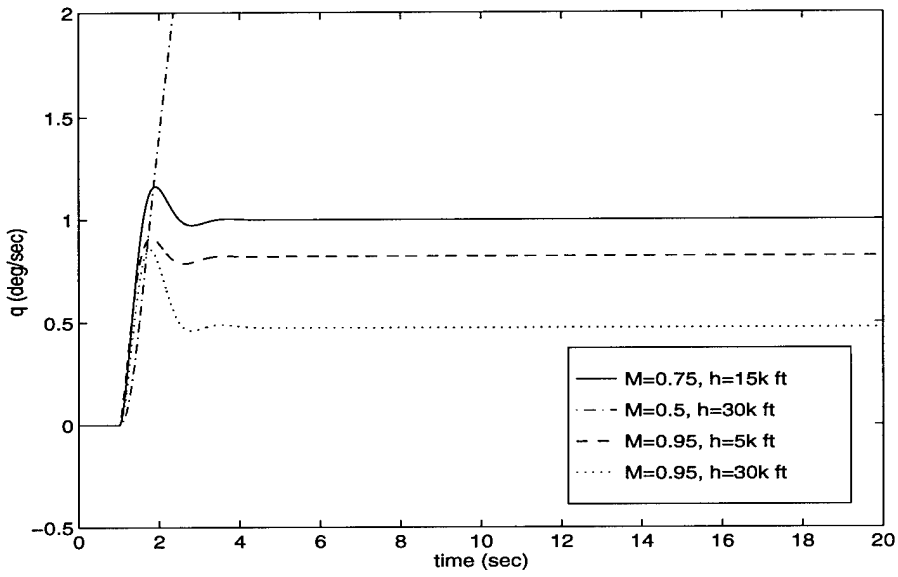


Figure 5.25 Step responses using short period LTI controller and short period plant model

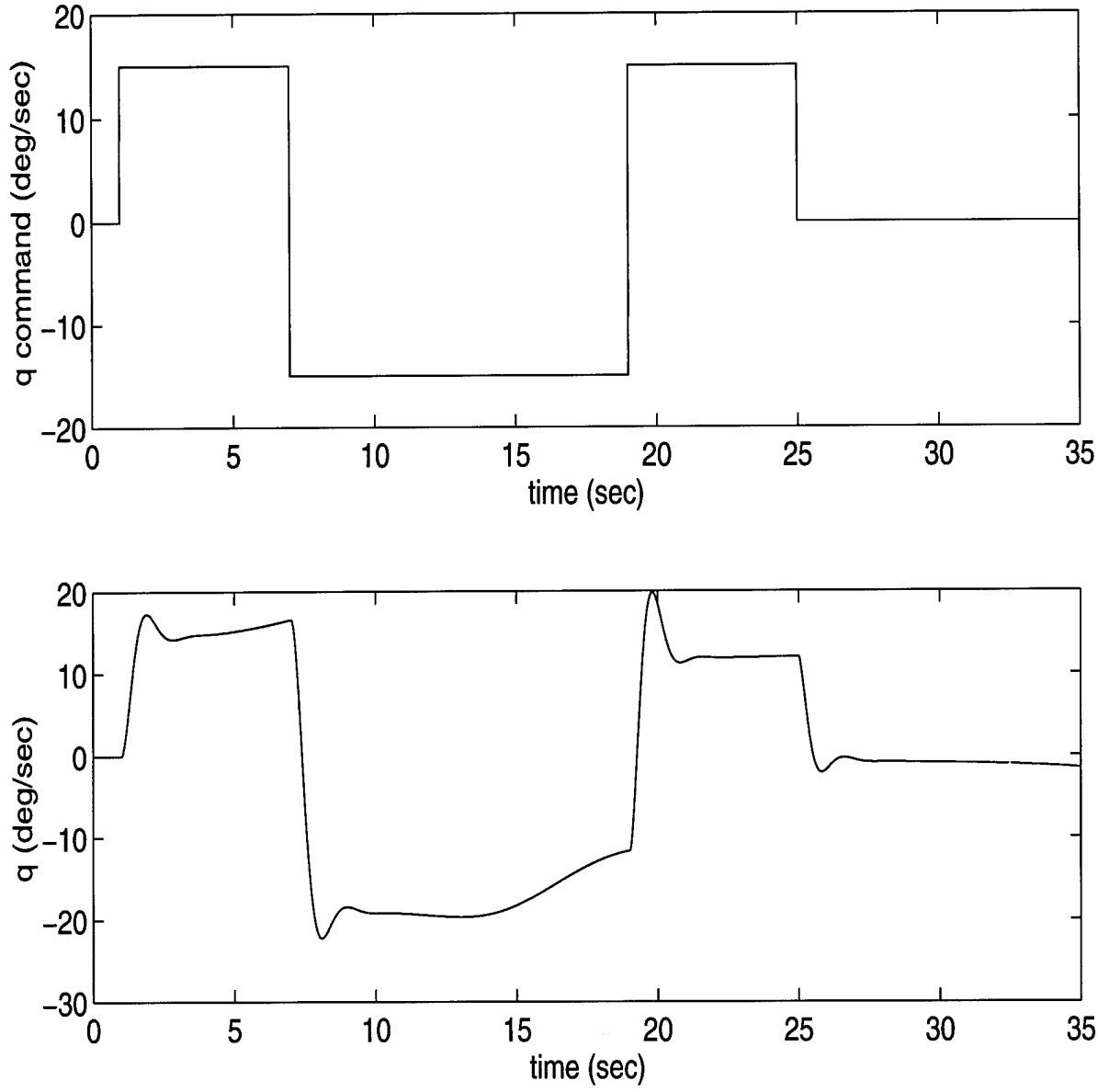


Figure 5.26 Dynamic response (using LTI controller) to a positive doublet pitch-rate command: q_c, q

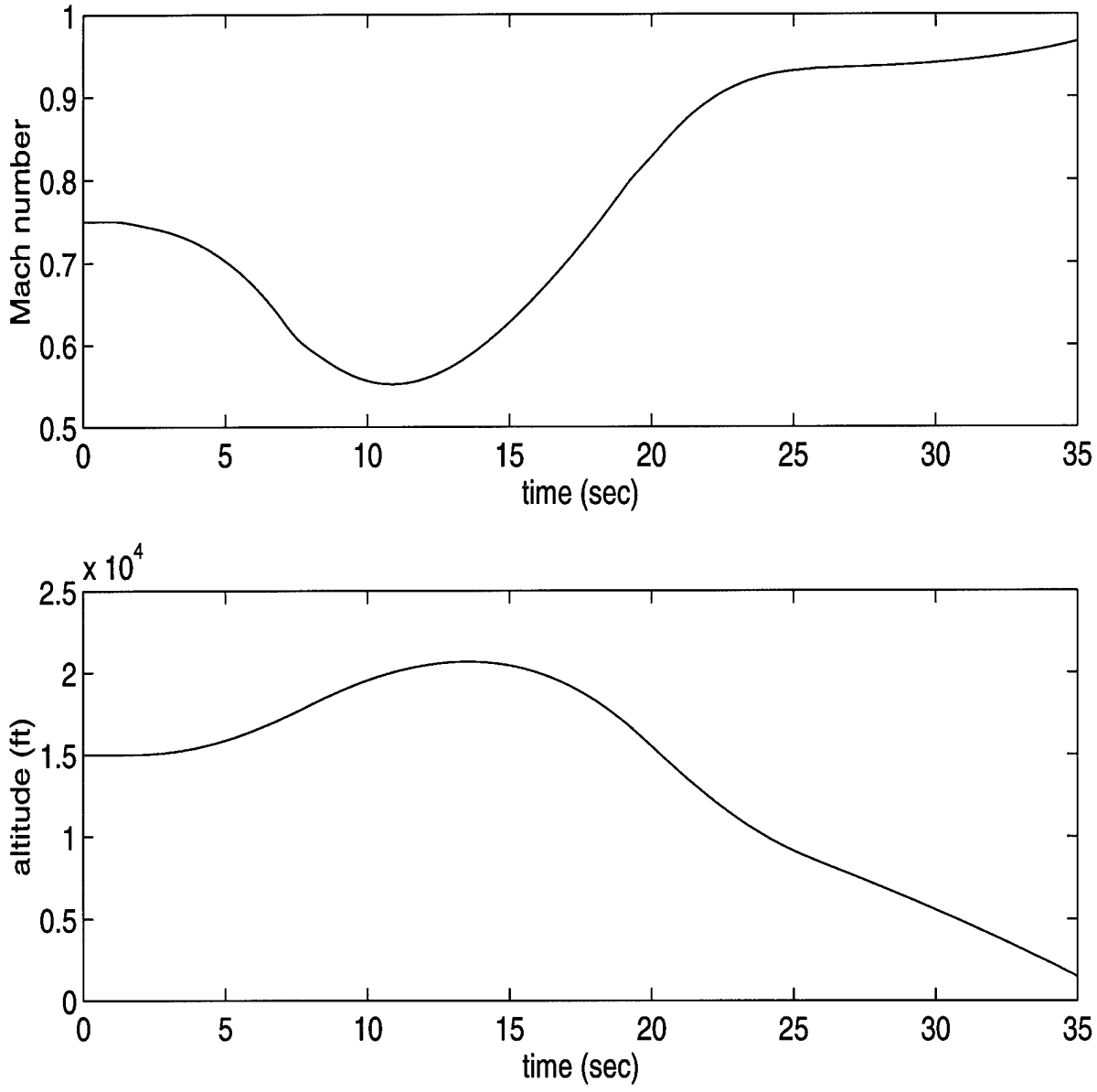
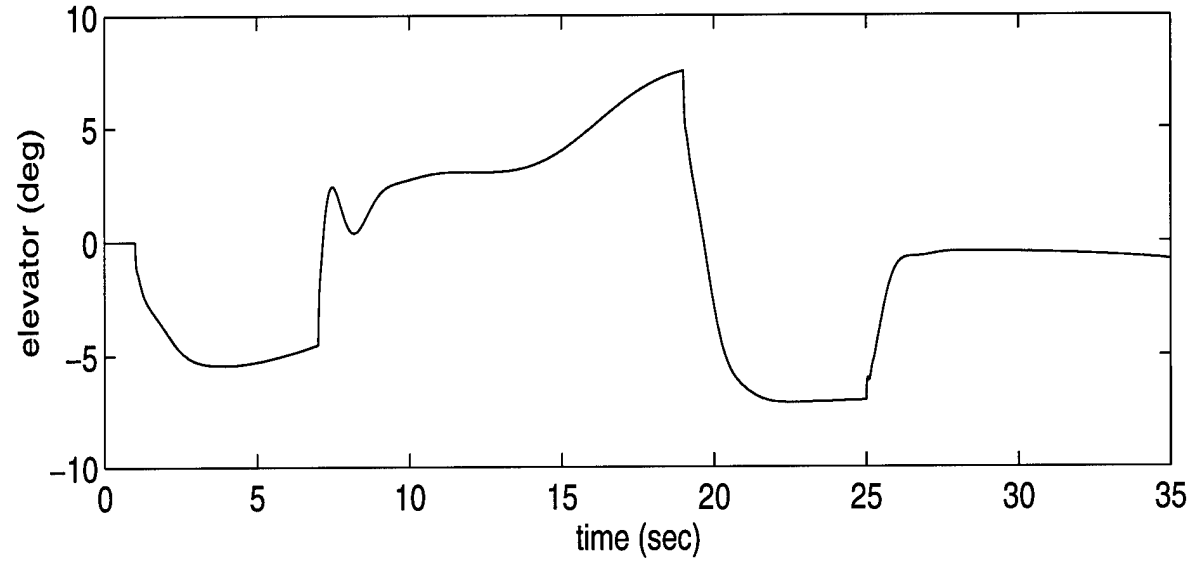


Figure 5.27 Dynamic response (using LTI controller) to a positive doublet pitch-rate command: M, h



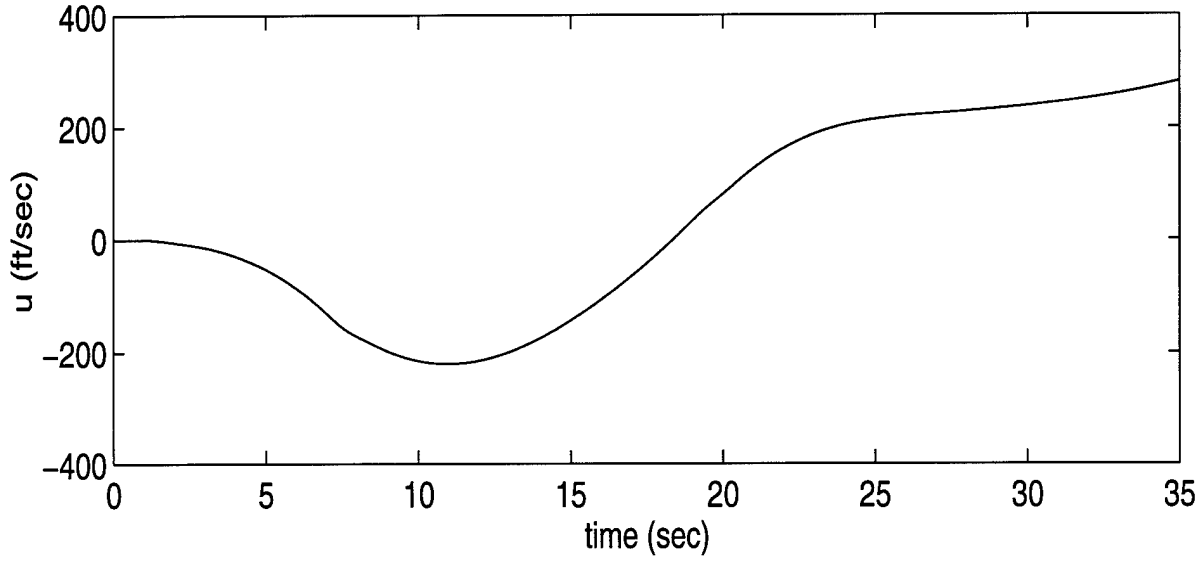


Figure 5.28 Dynamic response (using LTI controller) to a positive doublet pitch-rate command: δ_e, u

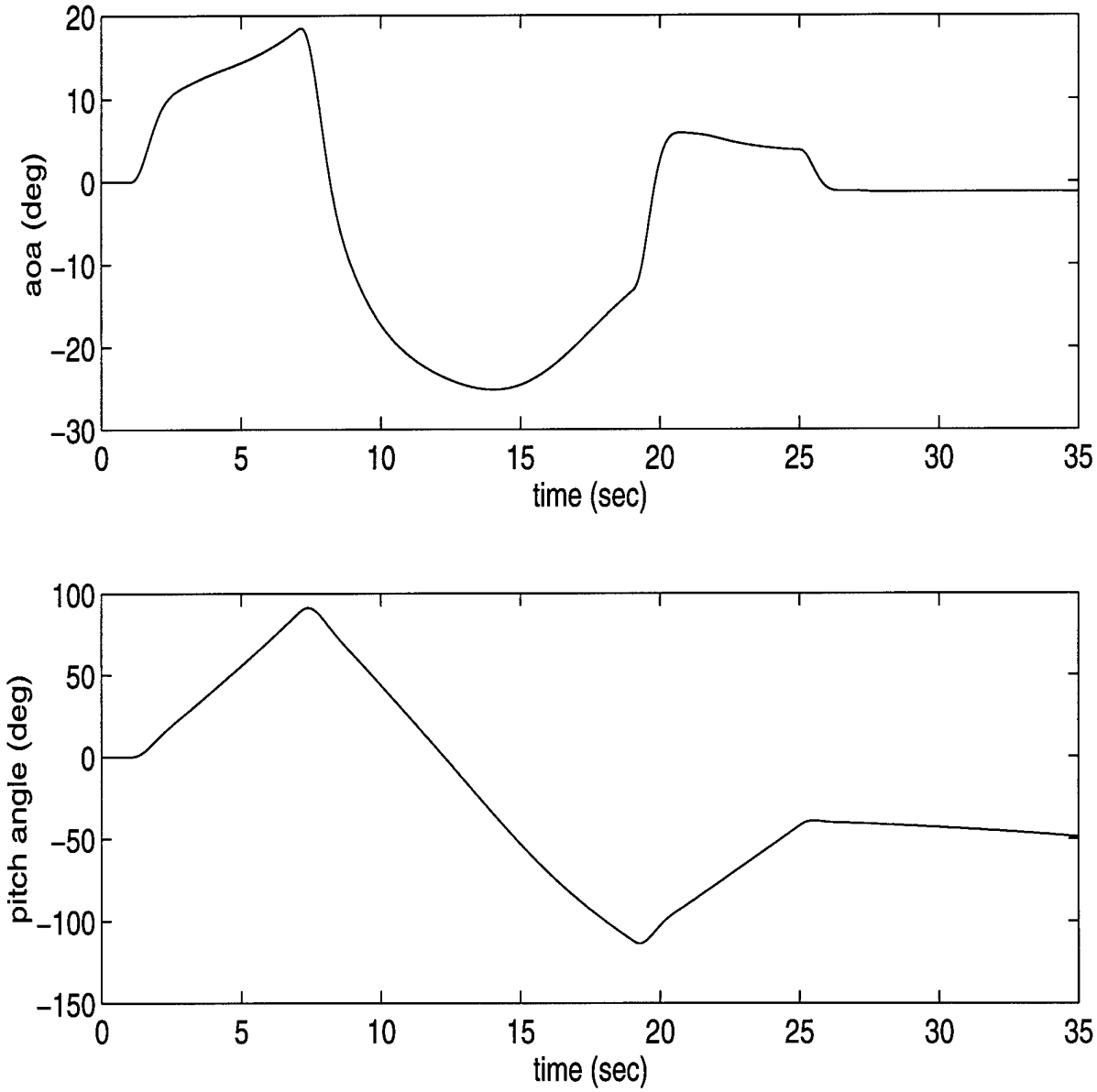


Figure 5.29 Dynamic response (using LTI controller) to a positive doublet pitch-rate command: α, θ

VI. Conclusions and Recommendations

The objectives of this thesis, set forth in Chapter I, were accomplished. The LMI formulation of a new gain-scheduling approach for LPV systems developed by Packard, Apkarian, and Gahinet was presented, along with the supporting theory and an extension to address LPV systems with modelled uncertainty. The method was tested in a realistic aircraft control problem in order to design two parameter-varying pitch-rate controllers, one for the short period aircraft model and the second for the full longitudinal model. The resulting short period LPV controller gives the desired robust performance and smoothly schedules itself based on measured values of the operating parameters. In the case of the full longitudinal design, the resulting LPV controller is robustly stabilizing, but does not provide acceptable performance. This emphasizes the conservatism of the approach which greatly increases when formulating large problems with many parameter-dependent elements, such as the full longitudinal design problem. Nevertheless, in the light of the simulation results, the short period LPV controller appears to be sufficient to provide the desired responses in the design envelope, although full non-linear aircraft models should be tested to confirm this.

6.1 Conclusions

Based on the presented theory and the subsequent implementations, some conclusions can readily be drawn.

First and foremost is the fact that this new method provides a relatively straightforward and numerically tractable way to design a practical gain-scheduled controller. As opposed to more classical methods, no point designs are required and no schedule must be determined; instead, the control law is directly designed by solving a convex optimization problem. In addition, unlike the other similar ap-

proaches mentioned in Chapter I where an infinite number of LMI's must be solved [BP94],[WYPB94], this method only requires solving 4 LMI's.

However, as has been explained, depending on the size of the design problem, even solving these four LMI's can prove to be a difficult task, since the LMI solvers are extremely computationally demanding. The size of the problem can be reduced by limiting the order of the parameters on which the elements of the state-space matricies depend, but this comes at the expense of additional model uncertainties.

In addition, using this method, it is unlikely that a single LPV controller can be found to gain-schedule an entire aircraft operating envelope. This is undoubtedly due to the excessive conservatism in the approach which stems from several sources. While it simplifies the formulation and solvability of the problem, the Small Gain Theorem forces the controller to handle complex, usually non-existing values of the parameters in addition to the real ones. Also, the problem formulation allows for infinitely fast parameter variation rates. Finally, if μ-synthesis is used, the fact that only complex full blocks can be used to represent the parameter blocks further increases the conservatism of the resulting controller designs. The result of this compounded conservatism is undoubtedly degraded robust performance, and results in guaranteed robust performance in a smaller envelope than is likely achievable in practice. For this reason, testing is recommended beyond the guaranteed envelope to further stretch the envelope over which the LPV controller is effective.

The last paragraph notwithstanding, the D-K-D procedure described in the thesis is strongly recommended for all gain-scheduling design problems, even when additional uncertainty is not modelled in the design, because the scales resulting from the gain-scheduling part of the problem only optimize the design on the input/output channels corresponding to the varying parameter block, whereas the scales derived in μ-synthesis are meant to optimize the design on all the input/output channels. In any case, the controller for the scaled H_∞ problem (the original gain-scheduling problem) is still available from the first D-K-D iteration.

It was also demonstrated, in the case of the short period design, that an LPV controller could guarantee robust stability and nominal performance over a much wider envelope than any optimal LTI controller. This was especially significant for a more complex plant like that of the full longitudinal design, for which the LTI norm for a robustly stabilizing LPV envelope reached a whopping 157.41. Having said that, the tradeoff between performance and stability is still present and, as the full longitudinal design demonstrated, may compete to the point that both requirements cannot be met simultaneously. This then requires either reducing the design envelope, and possibly simplifying the problem, or changing the performance requirements.

Specifically, if further full longitudinal designs are considered, it may be necessary to include a more complex, possibly time-varying, ideal reference model; alternatively, the model-matching design approach could be replaced by a design model in which performance is maximized by weighting the sensitivity. This last approach would avoid the need for a model and simply provide the best performance available. Another possibility would be to minimize the number of parameter-dependent stability derivatives by using a constant value for those that have the least effect on the overall aircraft model. Whatever the design methodology adopted, the designer should first develop the best LTI design possible. At best, this may eliminate the need to design an LPV controller if performance is acceptable. At worst, it will guide the proper choices of weights required to optimize the problem. In addition, the norm of the LTI nominal performance problem should be tested at the center of the robustly stabilizable design envelope, as an indicator of whether or not robust performance will likely be achievable.

Finally, it has been shown that, for at least some control problems, LTI controllers may perform as well or nearly as well as LPV controllers designed by this method. This undoubtedly depends upon the complexity of the design problem and the variability of the plant and its operating conditions. Since the LPV controllers

adapt to the changing operating conditions, their performance should in general surpass that of equivalent LTI controllers when the plant varies significantly throughout its operating envelope.

6.2 Recommendations

While the results were encouraging, some of the shortcomings of this gain-scheduling method highlight the need for further research in some areas.

Above all, there is the need to reduce the amount of conservatism inherent in the present method. This can be done by reducing the effect of or, preferably, entirely eliminating one or more of the three sources of conservatism mentioned in the last section. Becker's method [BP94], which considers only real values of the time-varying parameters, should be investigated and compared to the results obtained using the current method. Since it involves discretizing an infinite number of constraints, a small design problem such as the short period model is strongly recommended since the LMI solver is so computationally demanding. The method of Wu [WYPB94], which would further eliminate infinitely fast parameter variations, is only recommended if a more numerically tractable variation of the method can be presented; as is, even a small problem like the short period model would be extremely demanding. However, a possible alternative that should be investigated would be to include parameter rates as part of the time-varying parameter block by incorporating them in the curve-fits for the elements of the state-space matricies. This would prevent infinitely fast rates of change, although it would still allow complex values for them. Generally, though, it should help decrease some of the conservatism.

The advent of a practical method for μ-synthesis of real scalar uncertainties would eliminate the third source of conservatism. In fact, by combining the parameter block for the LPV controller and that of the LPV plant and by grouping like parameters to form real repeated scalar blocks, such real μ-synthesis techniques should entirely eliminate the need for the LMI L_t-scales, since these would effectively

already be produced by the μ-synthesis iteration. Certainly, when real μ-synthesis becomes practical, gain-scheduling should thus be revisited.

An observation can also be made that, while the more classical methods must determine control laws for the controller or its coefficients (depending on the method), the LPV method discussed in this thesis requires determining a curve-fit for many elements of the state-space matricies. However, the LPV method is able to guarantee robust performance over its entire design envelope, whereas the others cannot. This also emphasizes the importance of maintaining relatively good accuracy in modelling the LPV plant; otherwise, robust performance will be meaningless. This is an issue that warrants further investigation; specifically, to examine the relationship between curve-fitting the elements of the LPV state-space matricies, the resulting plant inaccuracies, and the effect on actual controller robust performance.

Finally, further controller designs should also be attempted in discrete-time and account for time delays and errors in the parameter measurements which would occur in an actual aircraft implementation. The discrete-time problem formulation is already provided in [AG95]. Similarly, incorporating the advantages of other norms such as H_2 and l_1, to reduce the effects of noise and add limits on control deflection or error magnitude should also be considered. An LMI approach for a general mixed H_2/H_∞ problem has already been presented in [CW94]; similarly, a mixed H_2/H_∞ approach has been applied to LPV systems in [Sch95]. These methods could be further extended to the gain-scheduled LPV H_∞ control method discussed in this thesis.

Appendix A. F-18 Design Flight Conditions

The following table lists the trimmed flight conditions that were used in the design and simulation of the longitudinal gain-scheduled controller. The data point numbers correspond to those given in all the figures of Appendix B, in order to easily cross-reference them with the stability derivative curve-fits.

Table A.1 Longitudinal Design Flight Conditions

Data Point #	Mach Number	Altitude (ft)	Dynamic Pressure (psf)	α (deg)
1	0.3	26000	47.4	25.2
2	0.5	40000	68.5	16.8
3	0.6	30000	158.4	5.2
4	0.4	6000	189.9	6.0
5	0.7	14000	426.4	2.6
6	0.8	12000	603.0	1.9
7	0.95	20000	614.4	1.6
8	0.8	10000	652.0	1.7
9	0.8	5000	789.1	1.5
10	0.9	10000	825.2	1.4
11	0.85	5000	890.8	1.4
12	0.9	5000	998.7	1.3
13	0.3	15000	75.0	15.0
14	0.5	12000	236.0	4.8
15	0.5	20000	170.0	6.5
16	0.6	5000	444.0	2.6
17	0.8	25000	198.0	3.0
18	0.8	35000	125.0	4.5

Appendix B. F-18 Stability Derivative Curve Fits

This appendix illustrates the curve-fits used to establish relationships between the stability derivatives and the varying parameters M and h for the simulation model, the short period design plant, and the full longitudinal design plant. For all the figures, the data point numbers correspond to those given in Table A.1 of Appendix A, in order to easily cross-reference the actual trimmed flight conditions.

B.1 Simulation Model Curve Fits

The following figures compare the trimmed data points to the polynomial curve fits used for the simulation model stability derivatives. The polynomials are as high as order M^3h^2 and cover the entire available flight envelope such that

$$M \in [0.3, 0.95] \text{ and } h \in [5000, 40000] \text{ ft.} \tag{B.1}$$

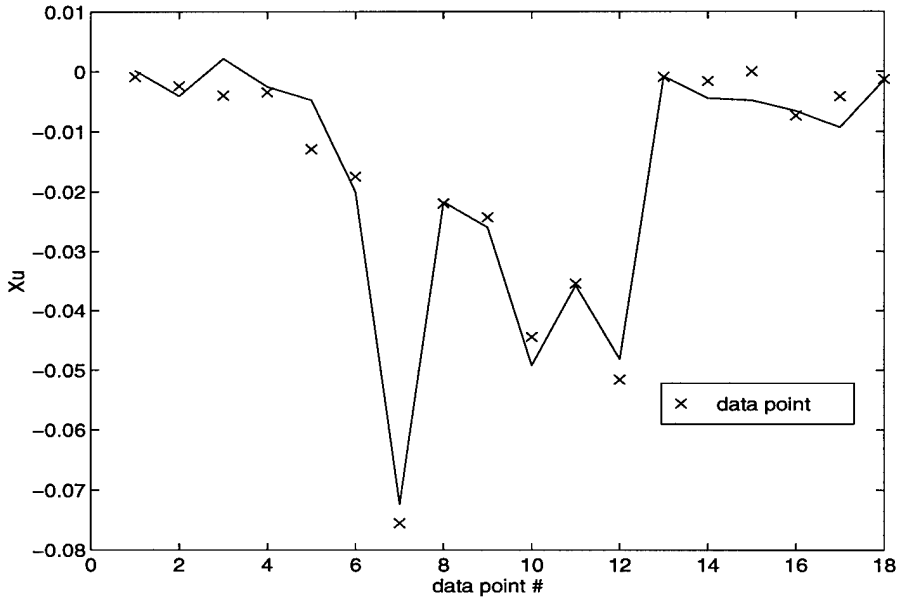


Figure B.1 Simulation model curve fit of X_u

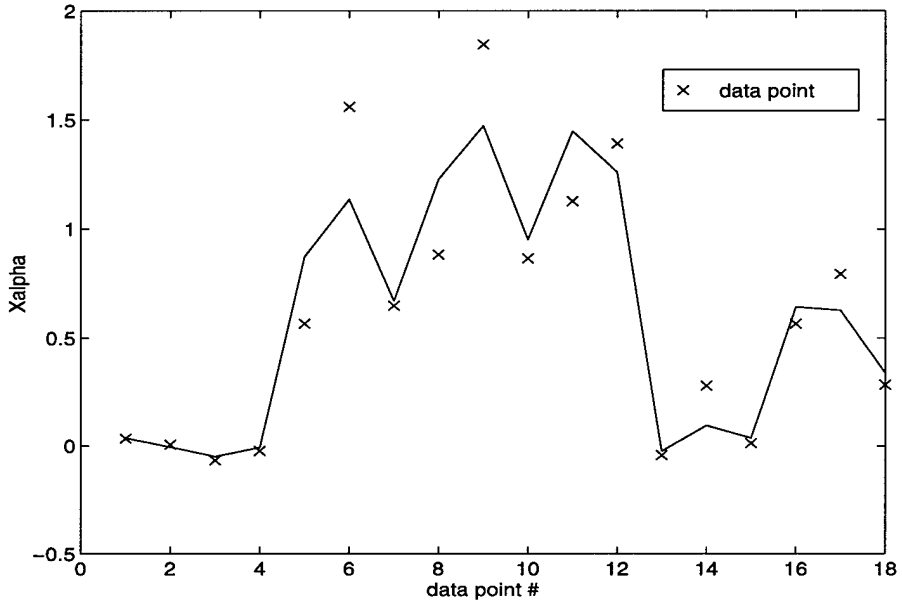


Figure B.2 Simulation model curve fit of X_α

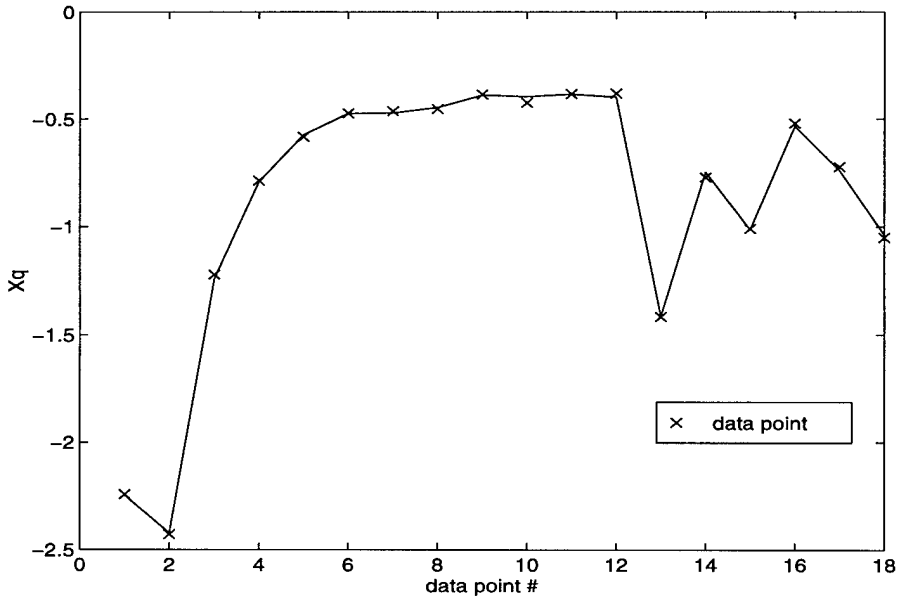


Figure B.3 Simulation model curve fit of X_q

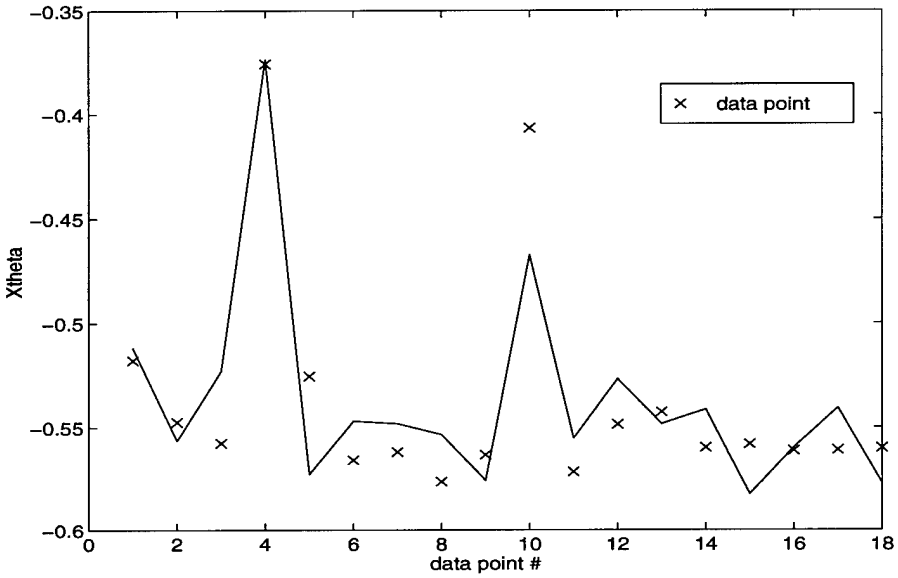


Figure B.4 Simulation model curve fit of X_θ

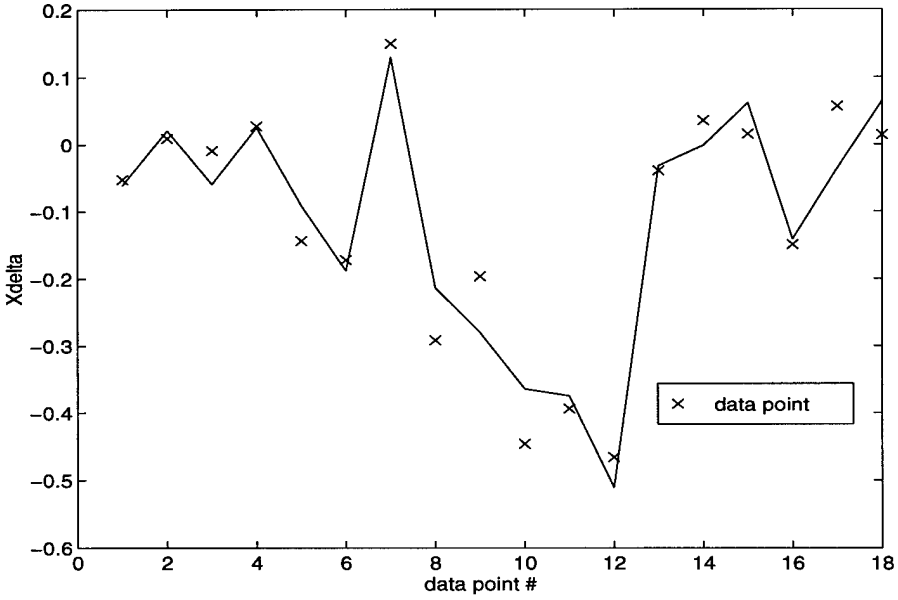


Figure B.5 Simulation model curve fit of X_δ

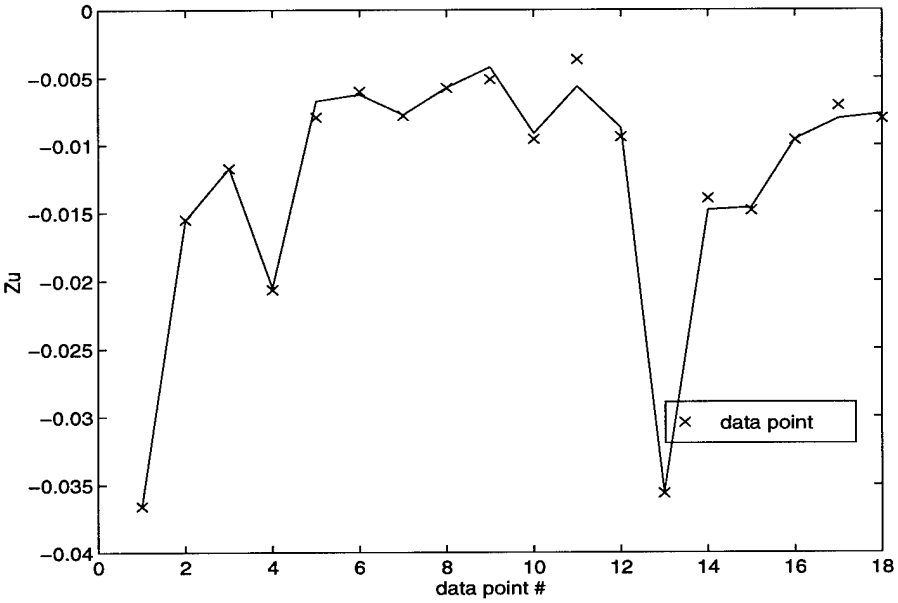


Figure B.6 Simulation model curve fit of Z_u

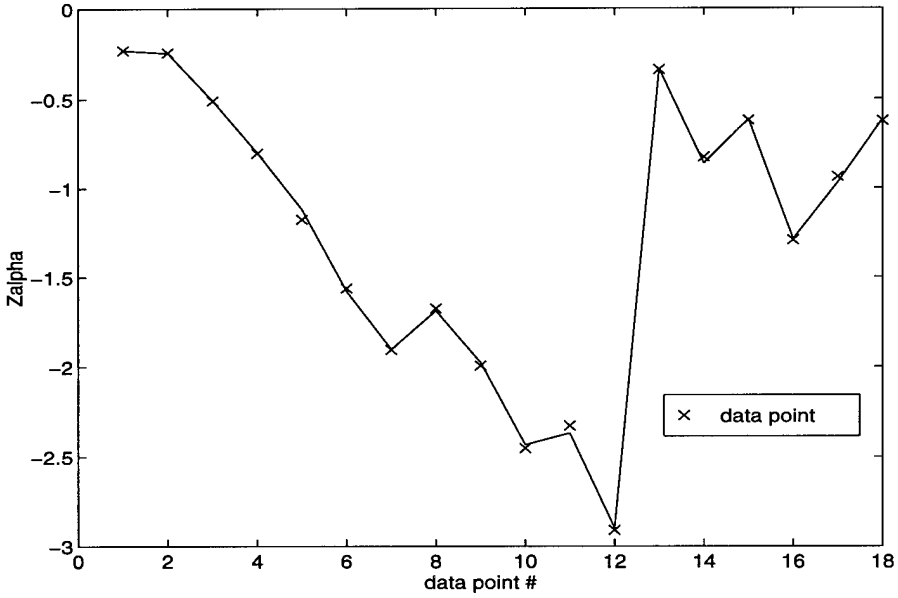


Figure B.7 Simulation model curve fit of Z_α

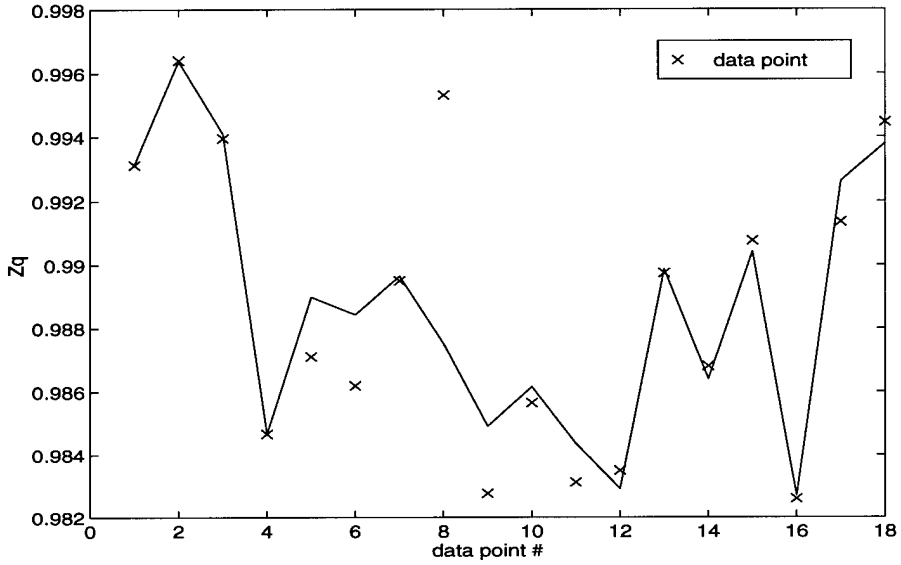


Figure B.8 Simulation model curve fit of Z_q

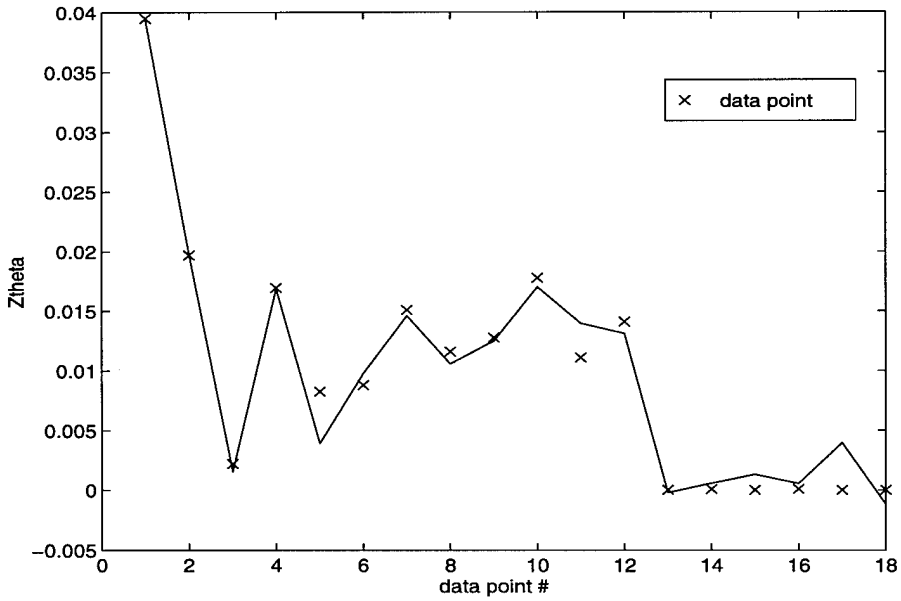


Figure B.9 Simulation model curve fit of Z_θ

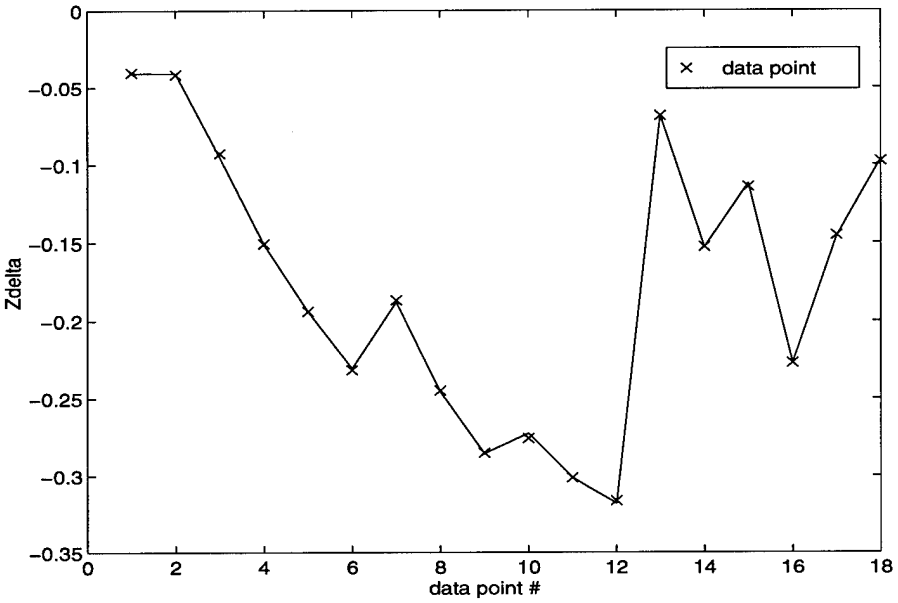


Figure B.10 Simulation model curve fit of Z_δ

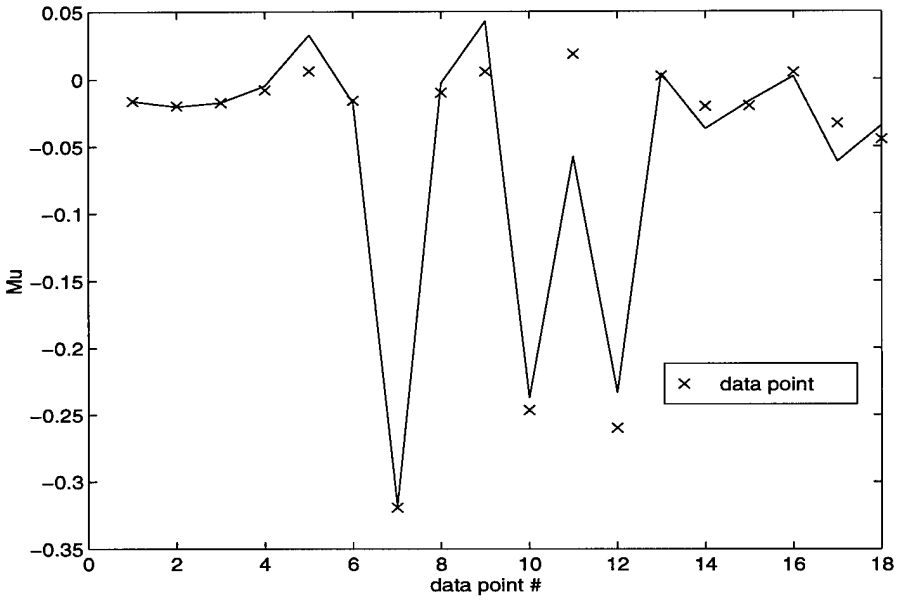


Figure B.11 Simulation model curve fit of M_u

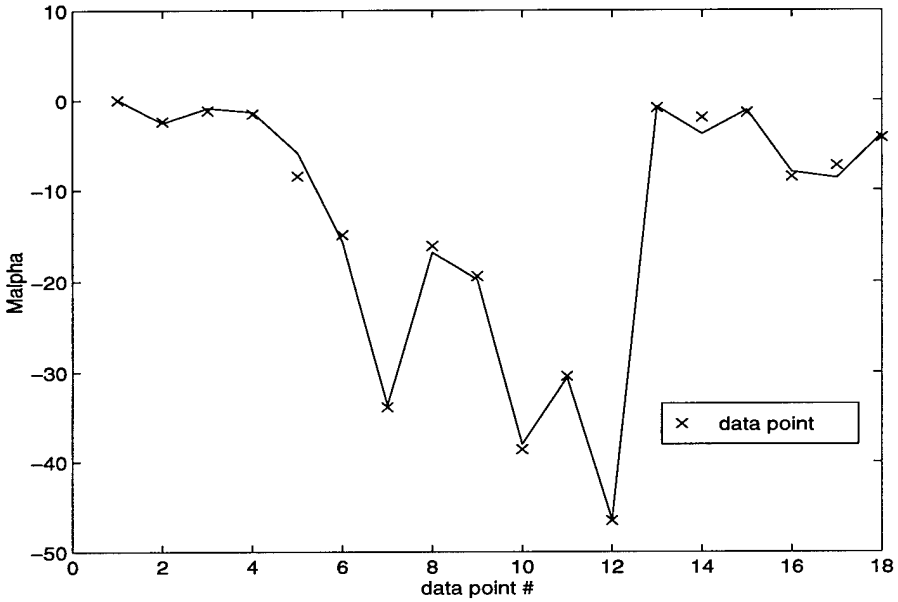


Figure B.12 Simulation model curve fit of M_α

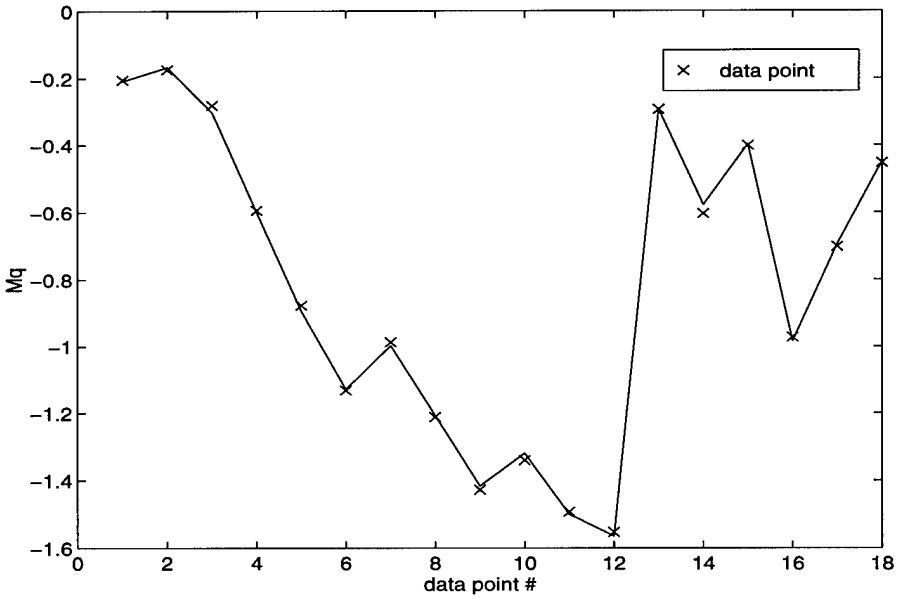


Figure B.13 Simulation model curve fit of M_q

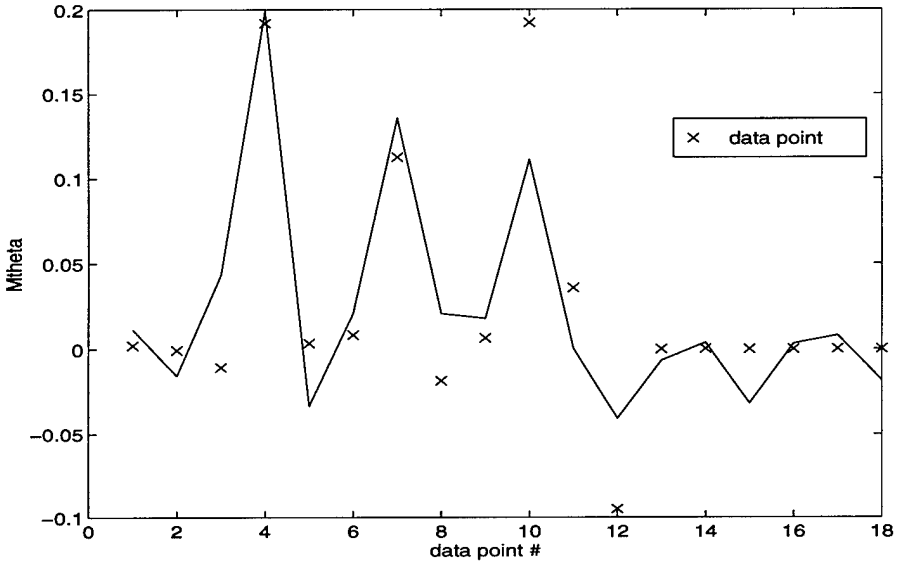


Figure B.14 Simulation model curve fit of M_θ

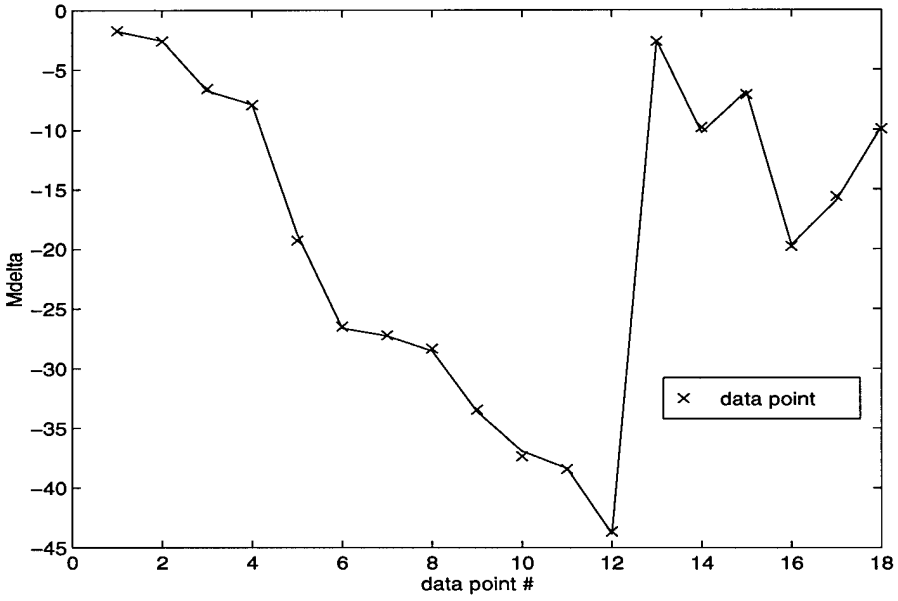


Figure B.15 Simulation model curve fit of M_δ

B.2 Short Period Design Model Curve Fits

The following figures compare the trimmed data points to the polynomial curve fits used for the short period design model stability derivatives. The polynomials are no higher than order M^2h and the design envelope is

$$M \in [0.5, 0.95] \text{ and } h \in [5000, 30000] \text{ ft.} \tag{B.2}$$

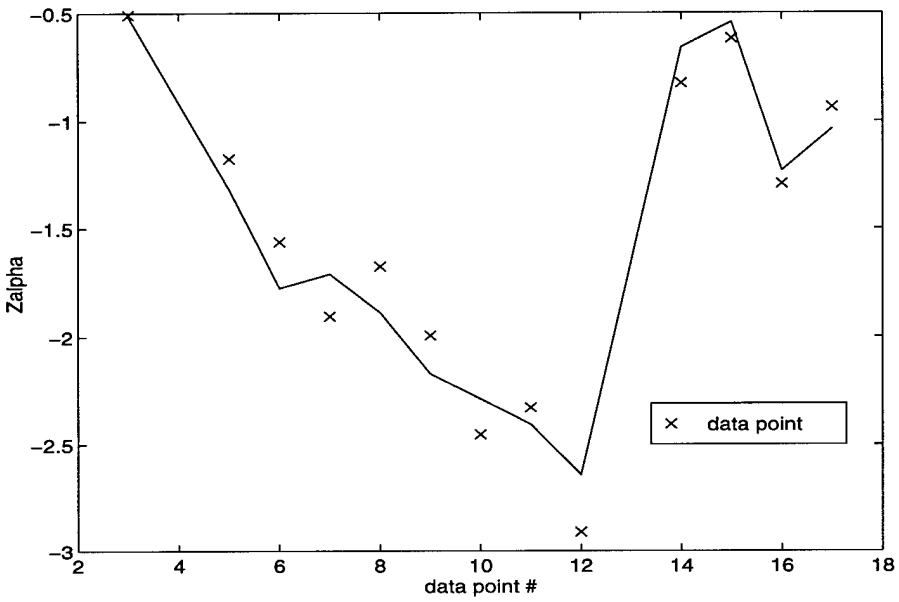


Figure B.16 Short period design model curve fit of Z_α

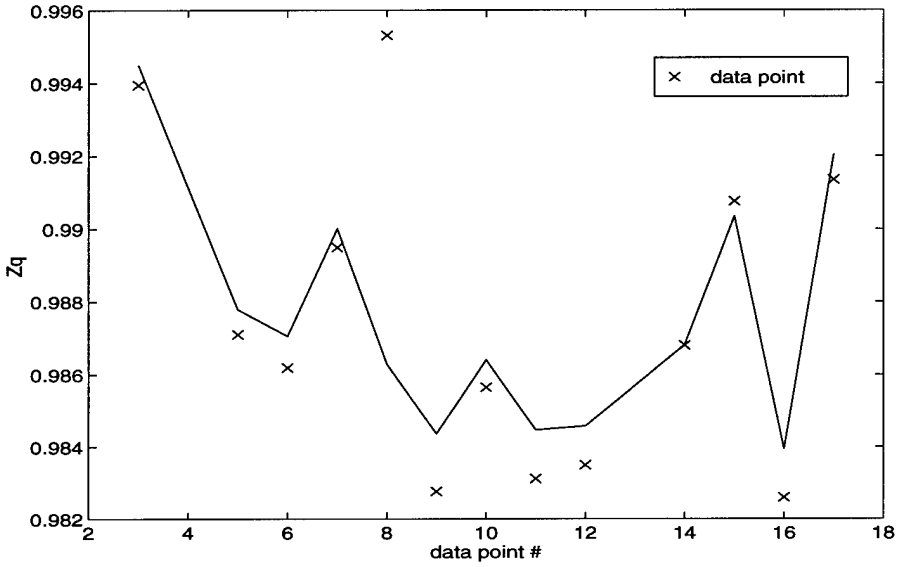


Figure B.17 Short period design model curve fit of Z_q

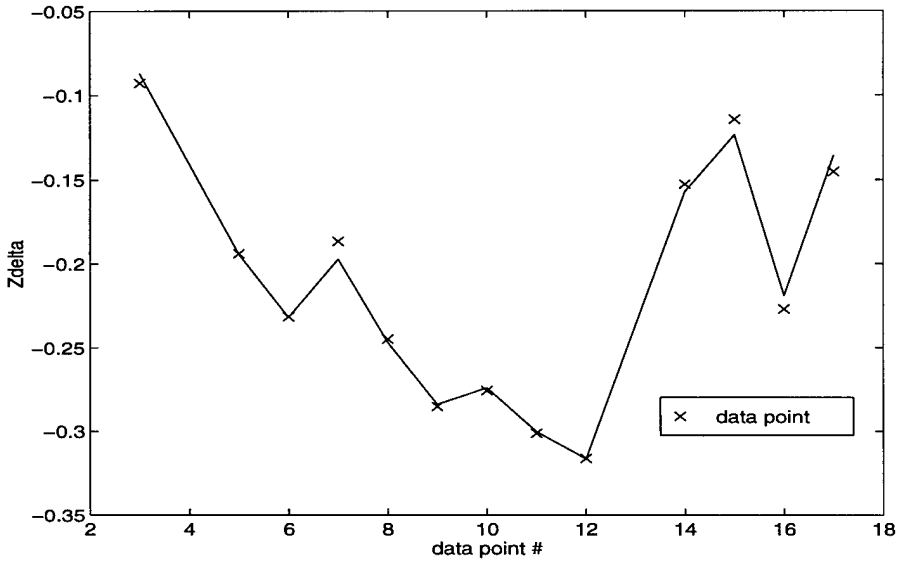


Figure B.18 Short period design model curve fit of Z_δ

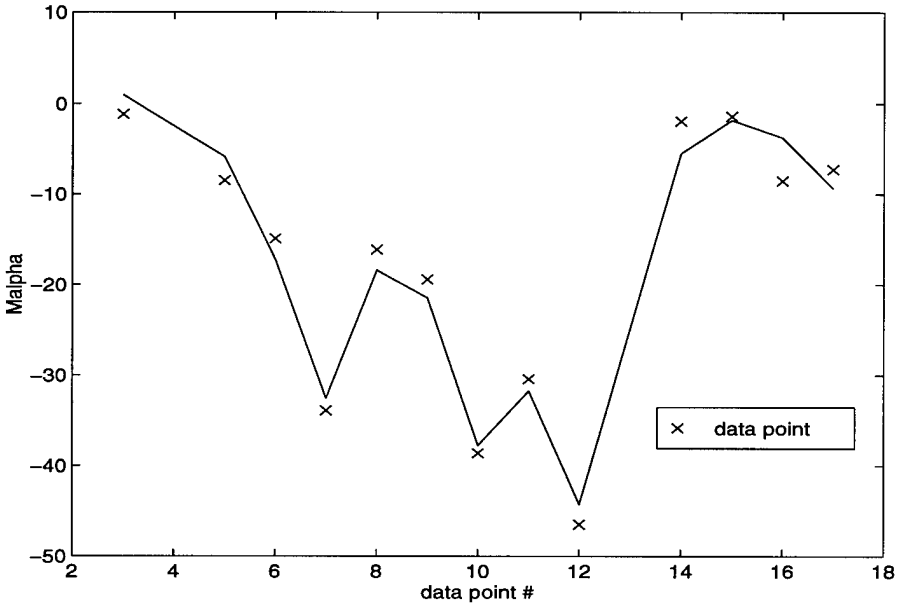


Figure B.19 Short period design model curve fit of M_α

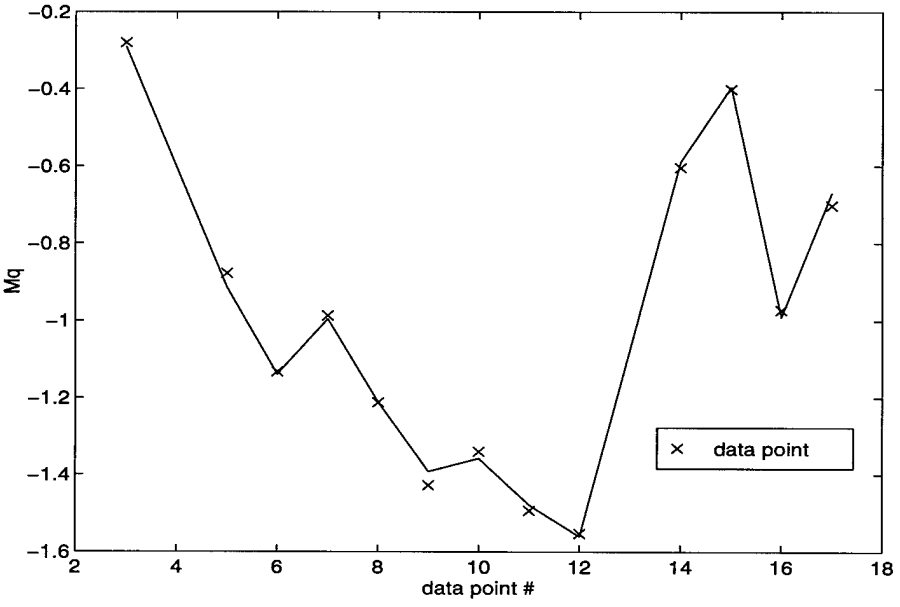


Figure B.20 Short period design model curve fit of M_q

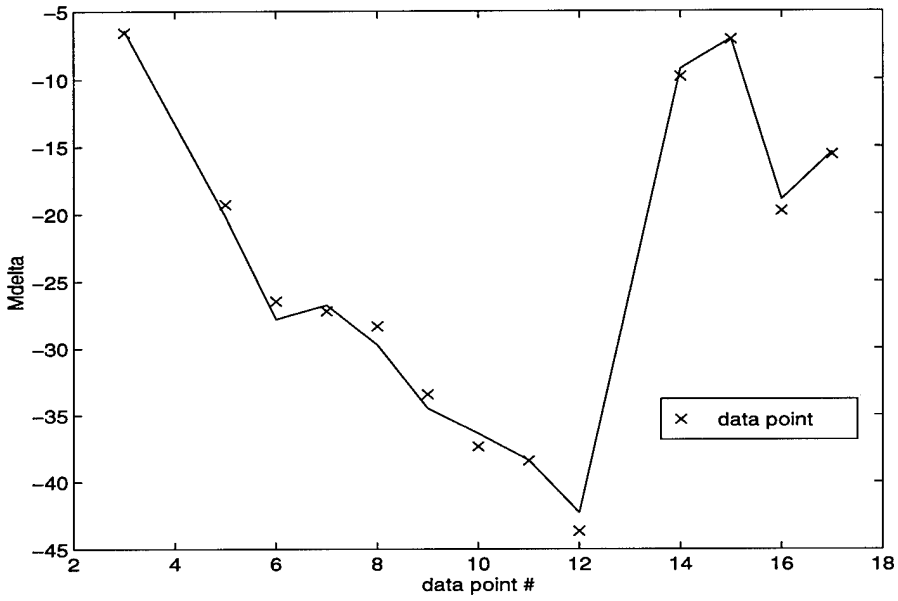


Figure B.21 Short period design model curve fit of M_δ

B.3 Full Longitudinal Design Model Curve Fits

The following figures compare the trimmed data points to the polynomial curve fits used for the full longitudinal model stability derivatives. The polynomials are no higher than order M^2h and the design envelope is

$$M \in [0.5, 0.95] \text{ and } h \in [5000, 30000] \text{ ft.} \tag{B.3}$$

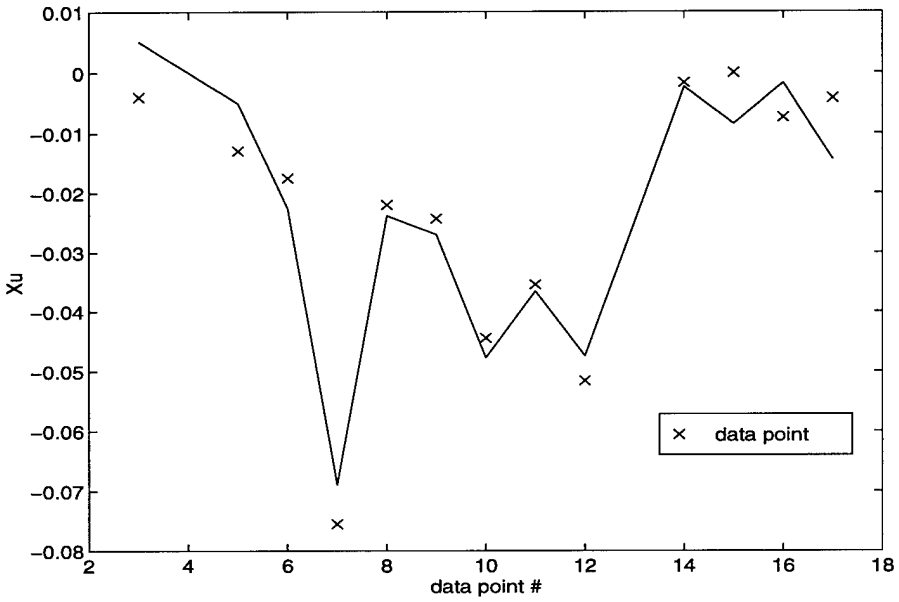


Figure B.22 Full longitudinal design model curve fit of X_u

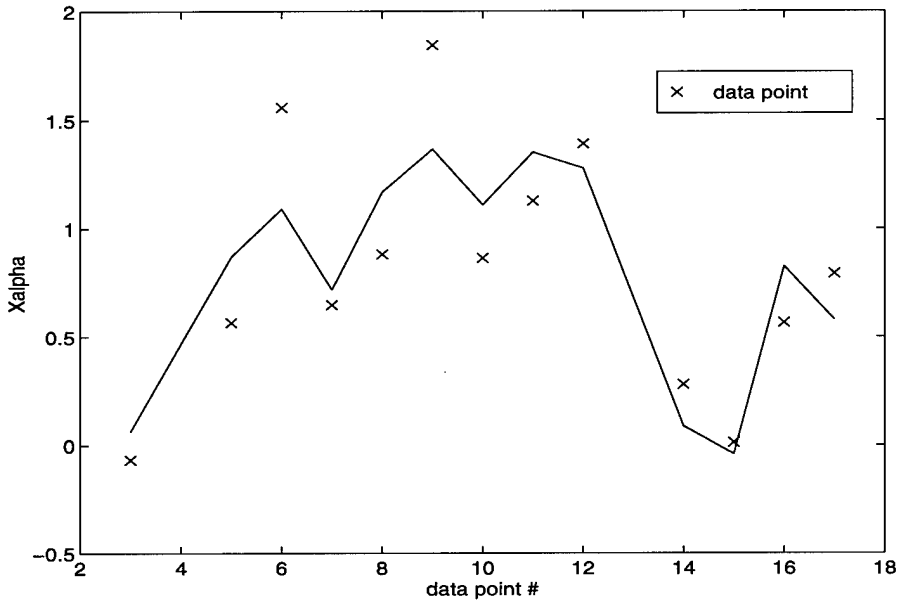


Figure B.23 Full longitudinal design model curve fit of X_α

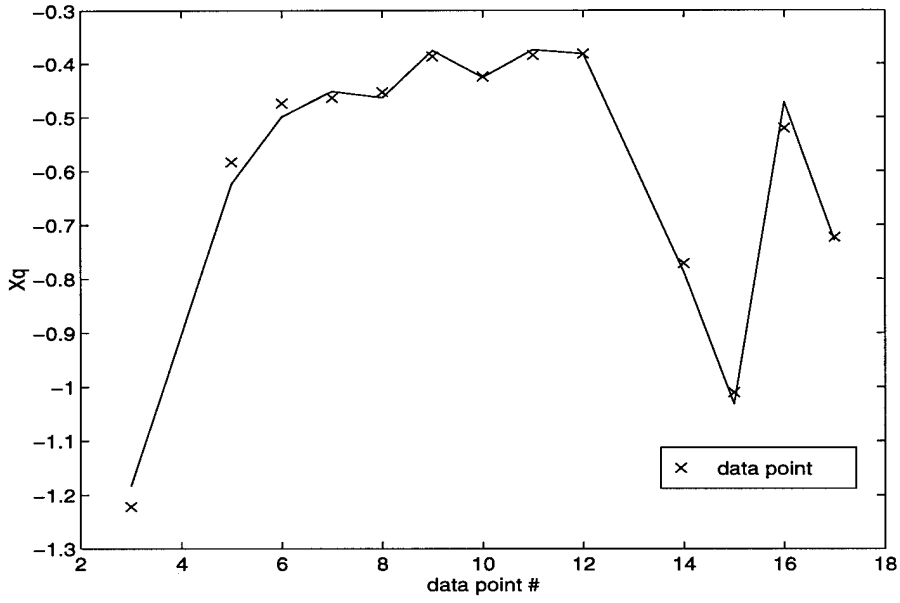


Figure B.24 Full longitudinal design model curve fit of X_q

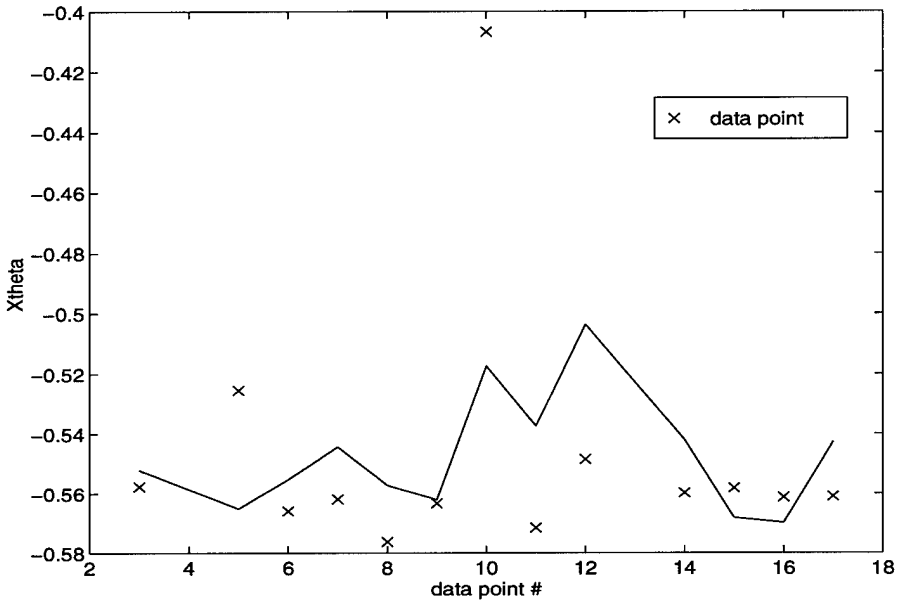


Figure B.25 Full longitudinal design model curve fit of X_θ

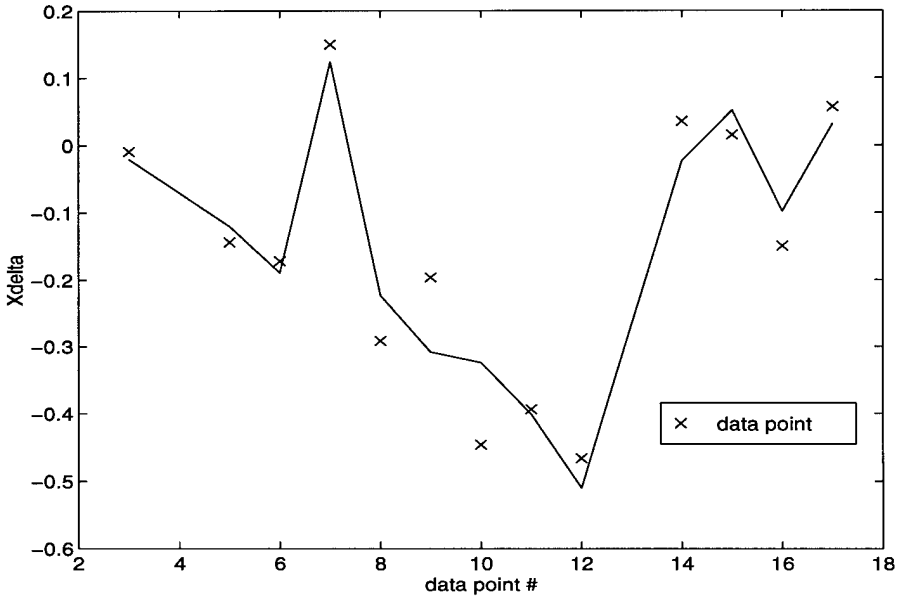


Figure B.26 Full longitudinal design model curve fit of X_δ

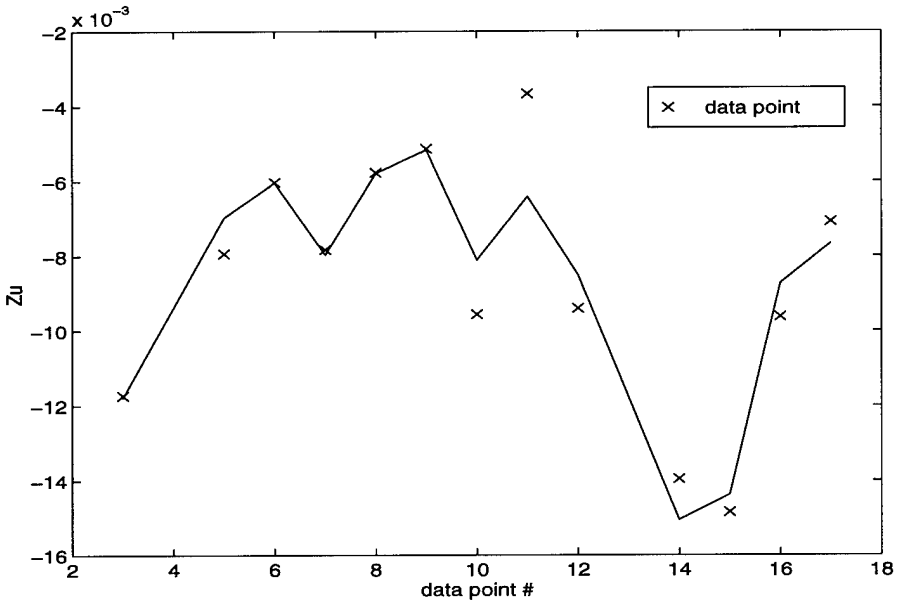


Figure B.27 Full longitudinal design model curve fit of Z_u

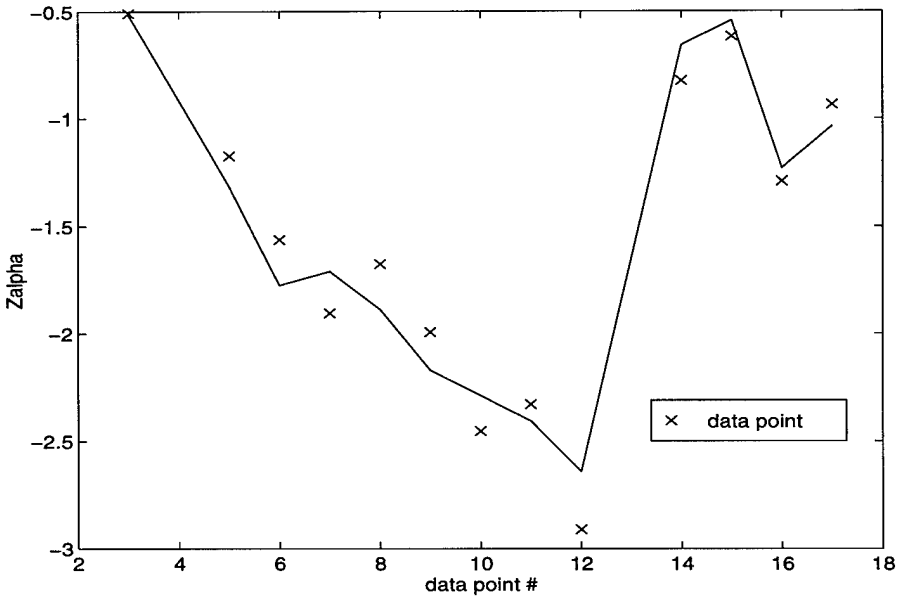


Figure B.28 Full longitudinal design model curve fit of Z_α

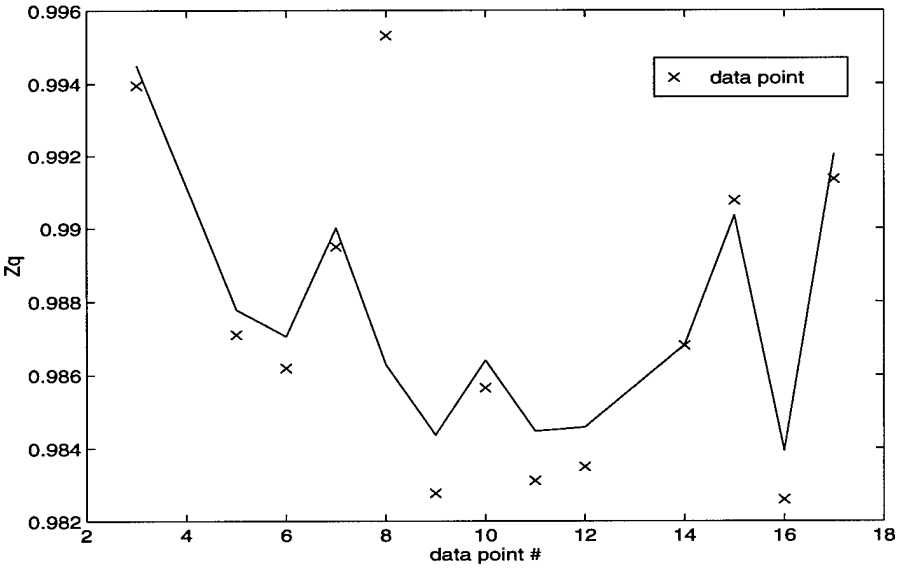


Figure B.29 Full longitudinal design model curve fit of Z_q

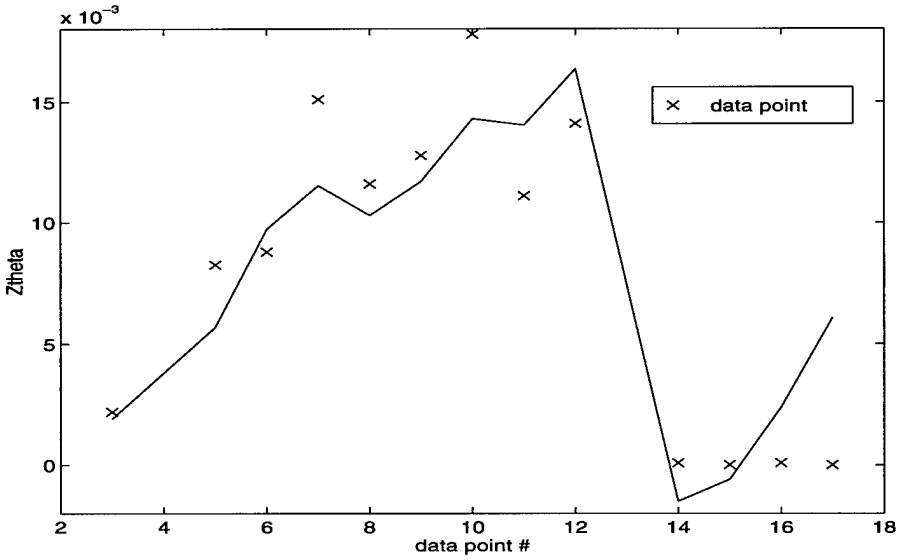


Figure B.30 Full longitudinal design model curve fit of Z_θ

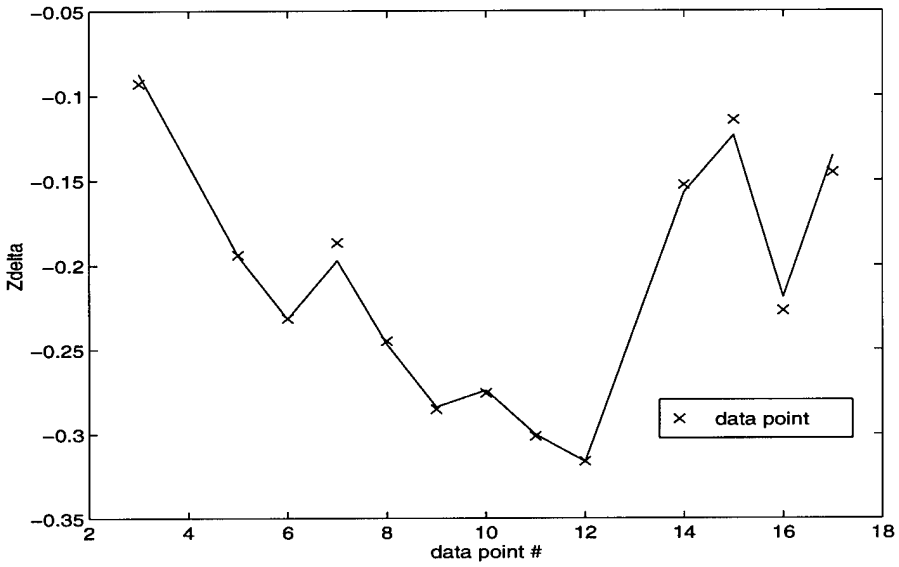


Figure B.31 Full longitudinal design model curve fit of Z_δ

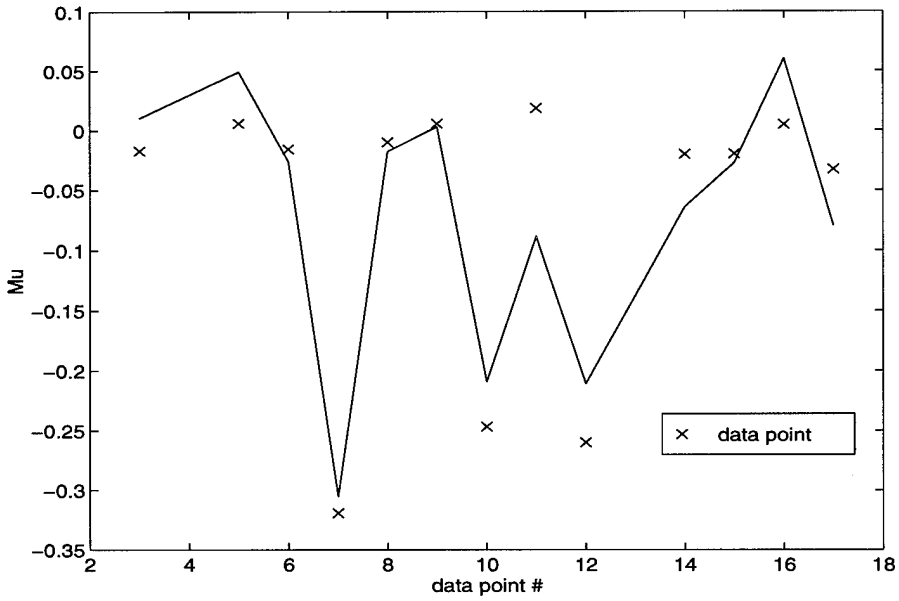


Figure B.32 Full longitudinal design model curve fit of M_u

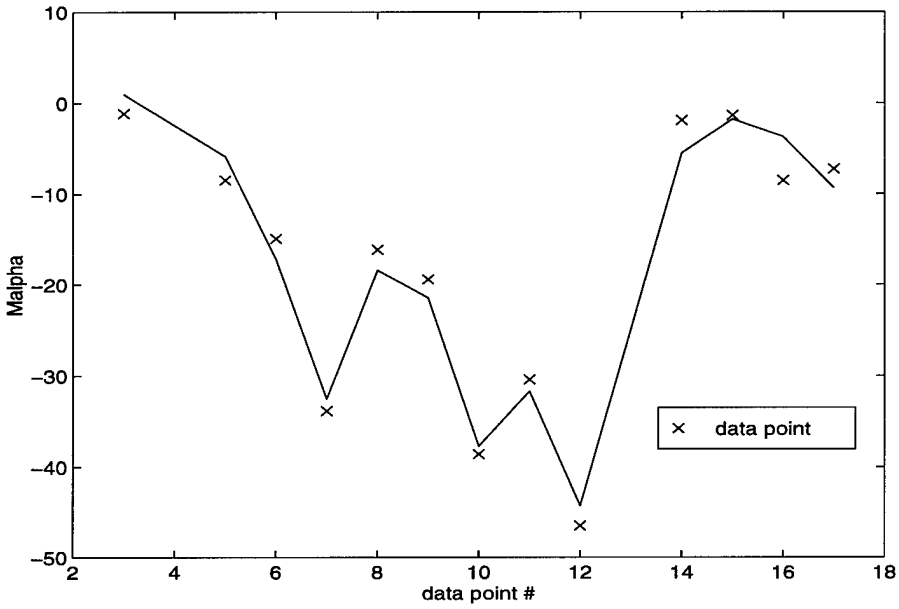


Figure B.33 Full longitudinal design model curve fit of M_α

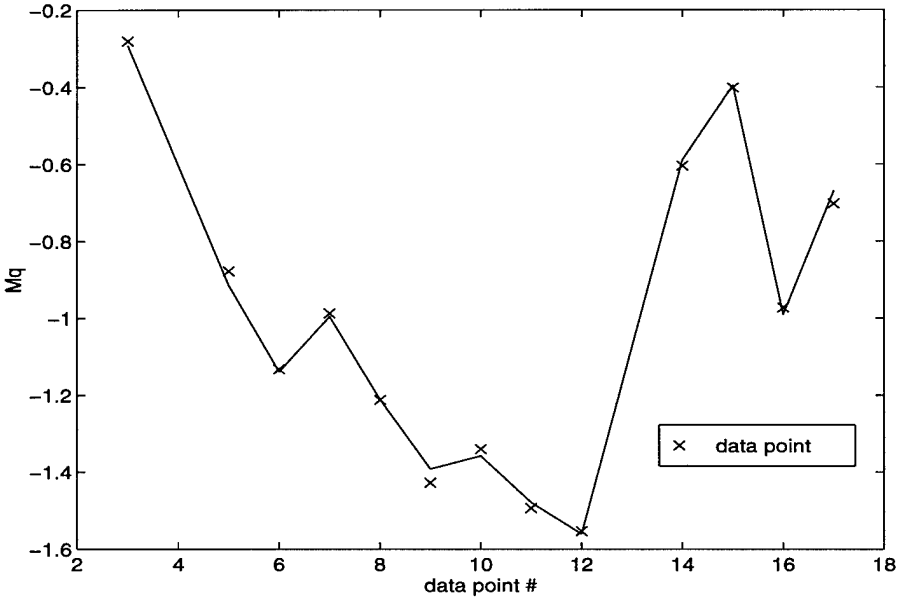


Figure B.34 Full longitudinal design model curve fit of M_q

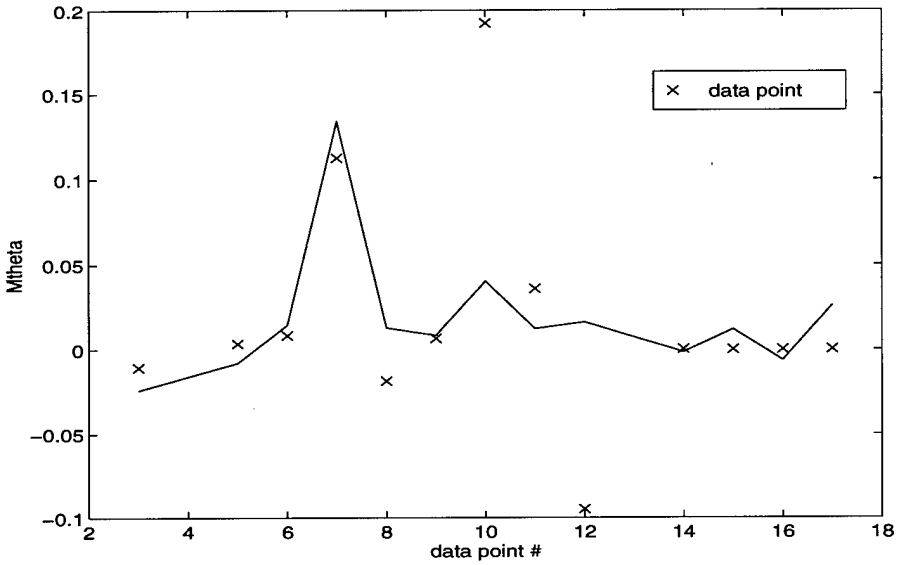


Figure B.35 Full longitudinal design model curve fit of M_θ

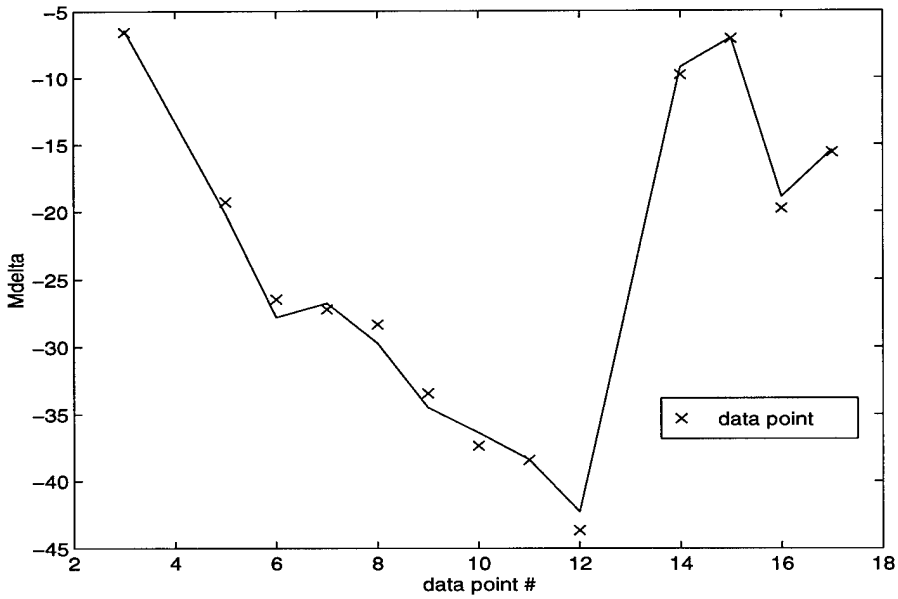


Figure B.36 Full longitudinal design model curve fit of M_δ

Appendix C. Implementation Programs

This appendix lists the main implementation programs, along with the main subprograms. They are included here for informational purposes only; the author should be contacted for access to the original set of programs (which are always evolving!).

C.1 `f18longdat.m`*: F-18 Trimmed Flight Condition Data*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18longdat.m

% F18 data for linear longitudinal models


% m3h26

a1=[-8.6469e-4 3.4079e-2 -2.242e0 -5.1786e-1;
-3.6574e-2 -2.2959e-1 9.9312e-1 3.9494e-2;
-1.6123e-2 2.4361e-2 -2.0464e-1 2.0935e-3;
0 0 9.998e-1 0];

b1=[-5.2165e-2;-4.0340e-2;-1.7302;0];

% m5h40

a2=[-2.4351e-3 6.204e-3 -2.4285e0 -5.4746e-1;
-1.5514e-2 -2.4232e-1 9.9641e-1 1.9699e-2;
-1.9554e-2 -2.3421e0 -1.7374e-1 -6.605e-4;
0 0 9.9998e-1 0];

b2=[9.3859e-3;-4.1597e-2;-2.5953;0];

% m6h30

b3=[-9.1427e-3;-9.2769e-2;-6.5735;0];

a3=[-4.0006e-3 -6.6825e-2 -1.2218e0 -5.5765e-1;
```

```
-1.1737e-2 -5.0877e-1 9.9395e-1 2.1962e-3;
-1.7155e-2 -1.1314e0 -2.8045e-1 -1.0661e-2;
0 0 1 0];

% m4h6

b4=[2.7435e-2;-1.5077e-1;-7.9255;0];

a4=[-3.4565e-3 -2.3318e-2 -7.8631e-1 -3.7581e-1;
-2.0661e-2 -8.0179e-1 9.8467e-1 1.6962e-2;
-7.9341e-3 -1.5211e0 -5.944e-1 1.9196e-1;
0 0 1 0];

% m7h14

b5=[-1.4391e-1;-1.9397e-1;-1.9292e1;0];

a5=[-1.2947e-2 5.6579e-1 -5.8325e-1 -5.2547e-1;
-7.9393e-3 -1.1751e0 9.8710e-1 8.2546e-3;
5.9347e-3 -8.4584e0 -8.7762e-1 3.4102e-3;
0 0 1 0];

% m8h12

b6=[-1.7201e-1;-2.3157e-1;-2.6479e1;0];

a6=[-1.7511e-2 1.5586e0 -4.7417e-1 -5.6586e-1;
-6.0265e-3 -1.5624e0 9.8619e-1 8.7944e-3;
-1.5683e-2 -1.4939e1 -1.1321e0 8.2664e-3;
0 0 1 0];

% m95h20

b7=[1.4969e-1;-1.8672e-1;-2.7216e1;0];

a7=[-7.5506e-2 6.4804e-1 -4.6379e-1 -5.6189e-1;
-7.8276e-3 -1.9054e0 9.895e-1 1.5097e-2;
-3.1925e-1 -3.3885e1 -9.8716e-1 1.1269e-1;
0 0 1 0];

% m8h10
```

```
b8=[-2.9175e-1;-2.4491e-1;-2.8335e1;0];

a8=[-2.1964e-2 8.823e-1 -4.5306e-1 -5.7631e-1;
-5.7642e-3 -1.6750e0  9.9531e-1 1.1595e-2;
-9.8837e-3 -1.6158e1 -1.2120e0 -1.8555e-2;
0 0 1 0];

% m8h5

b9=[-1.9651e-1;-2.8517e-1;-3.3445e1;0];

a9=[-2.4285e-2 1.8452e0 -3.8628e-1 -5.6327e-1;
-5.1311e-3 -1.9937e0 9.8277e-1 1.2773e-2;
5.6171e-3 -1.9439e1 -1.4272e0 6.5051e-3;
0 0 1 0];

% m9h10

b10=[-4.4621e-1;-2.7569e-1;-3.7363e1;0];

a10=[-4.4422e-2 8.6444e-1 -4.2435e-1 -4.0672e-1;
-9.5576e-3 -2.4524e0 9.8565e-1 1.7780e-2;
-2.4665e-1 -3.8613e1 -1.3401e0 1.9235e-1;
0 0 1 0];

% m85h5

b11=[-3.9377e-1;-3.0120e-1;-3.8430e1;0];

a11=[-3.5431e-2 1.1270e0 -3.8403e-1 -5.7154e-1;
-3.6673e-3 -2.3281e0 9.8312e-1 1.1098e-2;
1.8523e-2 -3.0436e1 -1.4930e0 3.5754e-2;
0 0 1 0];

% m9h5

a12=[-.0516 1.3906 -0.3817 -0.5485;
-0.0094 -2.9105 0.9835 0.0141;
-0.2600 -46.4719 -1.5527 -0.0948;
0 0 1 0];

b12=[-0.4662;-0.3161;-43.6515;0];
```

```
%m3h15

a13 =[-9.1762e-04  -4.2476e-02  -1.4162e+00  -5.4260e-01
  -3.5627e-02  -3.3711e-01   9.8974e-01   1.5817e-05
   2.2474e-03  -8.2586e-01  -2.9308e-01  -2.9711e-05
            0            0   9.9998e-01            0];


b13 =[-3.9377e-02
  -6.8546e-02
  -2.6358e+00
            0];

%m5h12

a14 =[-1.6397e-03   2.7965e-01  -7.7173e-01  -5.5977e-01
  -1.3957e-02  -8.2465e-01   9.8680e-01   8.2671e-05
  -2.0111e-02  -1.8828e+00  -6.0418e-01   1.5988e-04
            0            0   1.0000e+00            0];


b14 =[3.5500e-02
  -1.5267e-01
  -9.7970e+00
            0];


%m5h20

a15 =[-3.7742e-06   1.3294e-02  -1.0101e+00  -5.5813e-01
  -1.4848e-02  -6.1826e-01   9.9076e-01   4.7958e-06
  -1.9803e-02  -1.3457e+00  -4.0138e-01  -6.1538e-07
            0            0   1.0000e+00            0];


b15 =[1.5361e-02
  -1.1430e-01
  -7.0813e+00
            0];
```

```
%m6h5

a16 =[-7.3877e-03   5.6551e-01  -5.2056e-01  -5.6125e-01
  -9.6253e-03  -1.2925e+00   9.8260e-01   8.9218e-05
   4.8902e-03  -8.5064e+00  -9.7267e-01   1.1681e-05
            0            0   1.0000e+00            0];


b16 =[-1.4960e-01
  -2.2700e-01
  -1.9781e+01
            0];


%m8h25

a17 =[-4.1811e-03   7.9322e-01  -7.2318e-01  -5.6097e-01
  -7.0760e-03  -9.3598e-01   9.9135e-01   6.3784e-06
  -3.2723e-02  -7.2386e+00  -7.0232e-01   2.1230e-04
            0            0   1.0000e+00            0];


b17 =[5.6906e-02
  -1.4529e-01
  -1.5594e+01
            0];


%m8h35

a18 =[-1.3032e-03   2.8386e-01  -1.0511e+00  -5.5998e-01
  -8.0363e-03  -6.2303e-01   9.9446e-01   1.2041e-05
  -4.4622e-02  -4.1277e+00  -4.5340e-01   1.6097e-04
            0            0   1.0000e+00            0];


b18 =[1.4165e-02
  -9.7770e-02
  -9.9299e+00
            0];
```

```
% M,h,qbar(dyn press),aoa for all 12 points

Mhqa=[.3 26 47.4 25.2 ; .5 40 68.5 16.8
  .6 30 158.4 5.2 ; .4 6 189.9 6
  .7 14 426.4 2.6 ; .8 12 603 1.9
  .95 20 614.4 1.6 ; .8 10 652 1.7
  .8 5 789.1 1.5 ; .9 10 825.2 1.4
  .85 5 890.8 1.4 ; .9 5 998.7 1.3];

Mhqa2=[.3  15  75   15.0
.5      12  236   4.8
.5 20  170   6.5
.6      5   444   2.6
.8      25  198   3.0
.8      35  125   4.5];

Mhqa=[Mhqa;Mhqa2];

data=size(Mhqa,1);

%plot(Mhqa(:,1),Mhqa(:,2),'x') %plot M vs h
%axis([.2 1 0 50]);
%xlabel('Mach number');
%ylabel('altitude (1000 ft)');

% end of f18longdat.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.2 `f18poly.m`: *Curve-fit Trimmed Data to Polynomials*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18poly.m

% converts full longitudinal data to polynomial curve fit

f18longdat;
```

```
% choose design envelope
Mmin=.5;
Mmax=.95;
hmin=5;
hmax=30;

% build Adat for all pts (combines a,b)

Adat=[];
pts=0;
M=[];
h=[];
dindx=[];

for k=1:data,
Mdat=Mhqa(k,1);
hdat=Mhqa(k,2);
 if (Mdat>=Mmin & Mdat<=Mmax & hdat<=hmax & hdat>=hmin)
  pts=pts+1;
  M=[M;Mdat];
  h=[h;hdat];
  dindx=[dindx;k];
  eval(['a=a' int2str(k) ';']);
  eval(['b=b' int2str(k) ';']);
  Adat=[Adat; a(1,:) b(1) a(2,:) b(2) a(3,:) b(3)];
 end
end

clear a b

qbar=Mhqa(:,3);
qi=diag(minv(diag(qbar)));
qi2=qi.*qi;
q2=qbar.*qbar;
%h=1000*Mhqa(:,2);
h=1000*h;
h2=h.*h;
%M=Mhqa(:,1);
Mi=diag(minv(diag(M)));
Mi2=Mi.*Mi;
Mh=M.*h;
M2=M.*M;
```

```
M2h=M2.*h;
Mh2=M.*h2;
M2h2=M2.*h2;
M3=M2.*M;
M3h=diag(M3*h');
M3h2=diag(M3*h2');


% determine relationships between Adat and M,h

% names of stab. derivatives that depend on M,h
names='Xu Xa Xq Xt Xde Zu Za Zq Zt Zde Mu Ma Mq Mt Mde';

[num,names,namelen,maxlen]=namstuff(names);

% most accurate for generating only 10 terms
order=[2 1;2 1;2 1;2 1;2 1;1 1;1 1;1 1;1 1;1 1;2 1;2 1;2 1;2 1;1 1];

% Dr R.'s eyeball
%order=[2 1;3 2;1 1;3 2;3 2;2 1;1 1;0 0;2 1;1 1;3 2;2 1;1 1;3 2;1 1];

% <= 25% error
%order=[3 2;3 2;2 1;2 2;3 2;3 2;1 1;0 0;2 2;2 1;3 2;3 2;1 2;3 2;1 1];

% <= 10% error
%order=[3 2;3 2;2 2;2 2;3 2;3 2;2 1;0 0;3 2;2 1;3 2;3 2;1 2;3 2;2 1];

% <= 1% error
%order=[3 2;3 2;2 2;3 2;3 2;3 2;3 1;0 0;3 2;3 1;3 2;3 2;2 2;3 2;3 1];

% <= 1% error and delta s.v. < -20dB
%order=[3 2;3 2;2 2;3 2;3 2;3 2;3 2;1 1;3 2;3 1;3 2;3 2;2 2;3 2;3 2];

% MS order
%order=[2*ones(15,1) ones(15,1)];

% superduper -- used for simulation model
sim_order=[3*ones(15,1) 2*ones(15,1)];
%order=[3*ones(15,1) 2*ones(15,1)];

% q
%order=[ones(15,1) zeros(15,1)];
```

```
% constant nominal plant
%order=zeros(15,2);

maxfit=max([order(:,1);order(:,2)]);

aa00=[ones(pts,1)];
aa11=[aa00 h M Mh];
aa21=[aa11 M2 M2h];
aa22=[aa00 h h2 M Mh Mh2];
aa12=[aa00 h h2 M Mh Mh2];
aa22=[aa12 M2 M2h M2h2];
aa31=[aa21 M3 M3h];
aa32=[aa22 M3 M3h M3h2];

%aa10=[aa00 qbar];

for k=1:num;
  OM=order(k,1);
  Oh=order(k,2);
  eval(['aa=aa' int2str(OM) int2str(Oh) ';']);
  name=names(k,1:namelen(k));
  bb=Adat(:,k);
  cc=aa'*aa;
  dd=aa'*bb;
  xx=cc\dd;
  eval(['POLY_' name '=xx;']);
  err=[];
  for i=1:pts,
    er=(bb(i)-aa(i,:)*xx)/bb(i);
    err=[err;er];
  end
  err=max(abs(err));
  eval(['err_' name '=100*err']);
end

% plots
plt=0;

if plt==1

for k=1:num,
```

```
  OM=order(k,1);
  Oh=order(k,2);
  eval(['aa=aa' int2str(OM) int2str(Oh) ';']);
  name=names(k,1:namelen(k));
  t=1:pts;
  figure(k)
  plot(dindx(t),Adat(:,k),'x')
  legend('data point');
  hold;
  eval(['plot(dindx(t),aa*POLY_' name ')']);
%  legend('curve fit');
  hold;
%  title(name);
  xlabel('data point #');
  ylabel(name);
end

end

% end of f18poly.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.3 f18sppoly.m*: Curve-fit Trimmed Data to Polynomials*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18sppoly.m

% converts longitudinal data to short period polynomial curve fit

f18longdat;

% choose design envelope
Mmin=.4;
Mmax=.95;
hmin=5;
hmax=30;

% build Adat for all pts (combines a,b)
```

```
Adat=[];
pts=0;
M=[];
h=[];
dindx=[];

for k=1:data,
Mdat=Mhqa(k,1);
hdat=Mhqa(k,2);
 if (Mdat>=Mmin & Mdat<=Mmax & hdat<=hmax & hdat>=hmin)
  pts=pts+1;
  M=[M;Mdat];
  h=[h;hdat];
  dindx=[dindx;k];
  eval(['a=a' int2str(k) ';']);
  eval(['b=b' int2str(k) ';']);
  Adat=[Adat; a(2,2:3) b(2) a(3,2:3) b(3)];
 end
end

clear a b

% determine relationships between Adat and M,h

% names of stab. derivatives that depend on M,h
names='Xu Xa Xq Xt Xde Zu Za Zq Zt Zde Mu Ma Mq Mt Mde';

[num,names,namelen,maxlen]=namstuff(names);

qbar=Mhqa(:,3);
qi=diag(minv(diag(qbar)));
qi2=diag(qi*qi');
q2=diag(qbar*qbar');
%h=1000*Mhqa(:,2);
h=1000*h;
h=(h-22500)./35000;
h2=diag(h*h');
%M=Mhqa(:,1);
M=(M-.625)./.65;
Mi=diag(minv(diag(M)));
Mi2=diag(Mi*Mi');
```

```
Mh=diag(M*h');
M2=diag(M*M');
M2h=diag(M2*h');
Mh2=diag(M*h2');
M2h2=diag(M2*h2');
M3=diag(M2*M');
M3h=diag(M3*h');
M3h2=diag(M3*h2');


% determine relationships between Adat and M,h

% names of stab. derivatives that depend on M,h
names='Zalpha Zq Zdelta Malpha Mq Mdelta';

[num,names,namelen,maxlen]=namstuff(names);

% most accurate for generating only 6 terms
order=[1 1;1 1;1 1;2 1;2 1;1 1];

% <= 25% error -- 10 term theta block
%order=[1 1;0 0;2 1;3 2;1 2;1 1];

% <= 10% error -- 10 terms
%order=[2 1;0 0;2 1;3 2;1 2;2 1];

% <= 1% error -- 11 terms
%order=[3 1;0 0;3 1;3 2;2 2;3 1];

% <= 1% error and delta s.v. < -20dB  -- 12 terms
%order=[3 2;1 1;3 2;3 2;3 2;3 2];

% 7 terms
%order=[2*ones(6,1) ones(6,1)];

% 8 terms
%order=[1 1;0 0;1 1;3 2;1 1;2 1];

% modified MS -- 6 term theta
%order=[1 1;0 0;1 1;2 1;1 1;1 1];

% superduper -- 12 terms
```

```
%order=[3*ones(6,1) 2*ones(6,1)];

%order=[3 2;1 1;3 2;3 2;3 2;3 2];

% q
%order=[ones(6,1) zeros(6,1)];

% constant nominal plant
%order=zeros(6,2);

maxfit=max([order(:,1);order(:,2)]);

aa00=[ones(pts,1)];
aa11=[aa00 h M Mh];
aa21=[aa11 M2 M2h];
aa22=[aa00 h h2 M Mh Mh2];
aa12=[aa00 h h2 M Mh Mh2];
aa22=[aa12 M2 M2h M2h2];
aa31=[aa21 M3 M3h];
aa32=[aa22 M3 M3h M3h2];

%aa10=[aa00 qbar];

for k=1:num;
  OM=order(k,1);
  Oh=order(k,2);
  eval(['aa=aa' int2str(OM) int2str(Oh) ';']);
  name=names(k,1:namelen(k));
  bb=Adat(:,k);
  cc=aa'*aa;
  dd=aa'*bb;
  xx=cc\dd;
  eval(['POLY_' name '=xx;']);
  err=[];
  for i=1:pts,
    er=(bb(i)-aa(i,:)*xx)/bb(i);
    err=[err;er];
  end
  err=max(abs(err));
  eval(['err_' name '=100*err']);
end
```

```
% plots

plt=1;

if plt==1

for k=1:num,
  OM=order(k,1);
  Oh=order(k,2);
  eval(['aa=aa' int2str(OM) int2str(Oh) ';']);
  name=names(k,1:namelen(k));
  t=1:pts;
  figure(k)
  plot(dindx(t),Adat(:,k),'x')
  legend('data point');
  hold;
  eval(['plot(dindx(t),aa*POLY_' name ')']);
  hold;
%  title(name);
  xlabel('data point #');
  ylabel(name)
end

end

% end f18sppoly.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.4 f18poly2lft.m*: Convert Polynomials to LFT*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% f18poly2lft.m
%
% This script file defines the parameter normalization factors,
% calculates polynomial coefficients in the normalized deltas,
% and constructs LFTs (in deltas) for Za, Zde, Ma, Mq, Mde;
% these LFTs are combined to form lftab=[Za 1 Zde;Ma Mq Mde]
```

```
%
% Inputs: names = a string containing the names of the plant parameters
%                 (Za, Zq, etc)
%         POLY_xx = poly. coef.s for the parameters given in names, i.e.
%                   POLY_Za, POLY_Zq, etc.
%         maxfit = max. order of the poly's defined in POLY_xx
%         order = matrix defining the order (in M and h) of each poly. for
%                 each of the derivatives (i.e. Za, Zde, ...)
%                 Each row of order coeresponds to one of the derivatives,
%                 and the are listed in the order that they are given in
%                 names
% Outputs: LFTs for each of the derivatives and the A and B matrices.
%
% Written by: Capt Martin Breton, AFIT (for full longitudinal + schur redux)
%
% Based on a short period program written by: Capt Mark Spillman and Lawton Lee
%                     WL/FIGC-3 Bldg 146, WPAFB
%

% DEFINE inputs
names='Xu Xa Xq Xt Xde Zu Za Zq Zt Zde Mu Ma Mq Mt Mde';

flag=order(8,1);
if flag == 0,
  order=[order(1:7,:);order(9:15,:)];
  names='Xu Xa Xq Xt Xde Zu Za Zt Zde Mu Ma Mq Mt Mde';
  LFT_Zq=POLY_Zq;
end

tol=[];
%%%%%%%%% Parameter normalization factors and shortcut matrices to convert
%%%%%%%%% polynomial coefficients in parm to coefficients in delta_parm

% Design and spread Machs
Mbar=(Mmax+Mmin)/2;
Mtil=Mmax-Mbar;

% Design and spread altitudes
hbar = 500*(hmax+hmin);
htil = 1000*hmax-hbar;

% Create the shortcut matrices
```

```
Mmat=zeros(maxfit+1);
hmat=zeros(maxfit+1);
K=zeros(maxfit+1);kvec=1;K(1,1)=1;
for ii=1:maxfit+1;
 Mmat=Mmat + diag((Mtil.^(0:maxfit-ii+1))*(Mbar^(ii-1)),ii-1);
 hmat=hmat + diag((htil.^(0:maxfit-ii+1))*(hbar^(ii-1)),ii-1);
 if ii<=maxfit
  kvec=conv(kvec,[1 -1]);
  K(1:ii+1,ii+1)=abs(kvec');
 end
end
Mmat=Mmat.*K;
hmat=hmat.*K;

%%%%%%%%% LFT contruction

[num,names,namelen,maxlen] = namstuff(names);
blk=zeros(num,2);

for ii=1:num;
 OM=order(ii,1);Oh=order(ii,2);
 nM=OM*(Oh+1);nh=Oh;
 blk(ii,:)=[nM nh];
 name=names(ii,1:namelen(ii));
 eval(['c=kron(Mmat(1:OM+1,1:OM+1),hmat(1:Oh+1,1:Oh+1))*POLY_',...
        name ';'])
 eval(['LFT_' name  '=pss2sys([zeros(nM,nh+1) eye(nM);',...
        'zeros(nh,nM+1) eye(nh);c(nM+nh+1:-1:1)'']',nM+nh);'])
 eval(['[LFT_' name ',nrem]=lftmin(LFT_' name ',[nM nh],1,1,1,tol);'])
 blk(ii,:)=blk(ii,:)-[nrem 0];
end

%%%%%%%%% LFT construction for lftab

LFT_1=sbs(LFT_Xu,LFT_Xa,LFT_Xq,LFT_Xt,LFT_Xde);
LFT_2=sbs(LFT_Zu,LFT_Za,LFT_Zq,LFT_Zt,LFT_Zde);
LFT_3=sbs(LFT_Mu,LFT_Ma,LFT_Mq,LFT_Mt,LFT_Mde);
LFT_4=sbs(0,0,1,0,0);
LFT_AB=abv(LFT_1,LFT_2,LFT_3,LFT_4);

% Separate the M and h "states"
he=cumsum(sum(blk')');me=he-blk(:,2);
```

```
hs=he-blk(:,2)+ones(num,1);ms=me-blk(:,1)+ones(num,1);

mindx=[];
hindx=[];
for i=1:num,
 mindx=[mindx ms(i):me(i)];
 hindx=[hindx hs(i):he(i)];
end
indx=[mindx hindx];

LFT_AB=reordsys(LFT_AB,indx);

% Reduce the delta blocks one at a time
tblk=sum(blk)

[LFT_AB,nrem] = lftmin(LFT_AB,tblk,1,5,4,tol);
tblk = tblk - [nrem 0]
[LFT_AB,nrem] = lftmin(LFT_AB,tblk,2,5,4,tol);
tblk = tblk - [0 nrem]

% Now try a schur model reduction

[LFT_AB,nrem] = lftmin2(LFT_AB,tblk,1,5,4,tol);
tblk = tblk - [nrem 0]
[LFT_AB,nrem] = lftmin2(LFT_AB,tblk,2,5,4,tol);
tblk = tblk - [0 nrem]


minfo(LFT_AB)

%%%%%%%%%% Clean up the workspace

clear K c Mmat hmat ms hs me he indx kvec OM Oh blk
clear LFT_1 LFT_2 LFT_3 LFT_4 mindx hindx flag
clear nM nh name names namelen maxlen ii num nrem

% end of f18poly2lft.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% f18sppoly2lft.m
%
% This script file defines the parameter normalization factors,
% calculates polynomial coefficients in the normalized deltas,
% and constructs LFTs (in deltas) for Za, Zde, Ma, Mq, Mde;
% these LFTs are combined to form lftab=[Za 1 Zde;Ma Mq Mde]
%
% Inputs: names = a string containing the names of the plant parameters
%                 (Za, Zq, etc)
%         POLY_xx = poly. coef.s for the parameters given in names, i.e.
%                   POLY_Za, POLY_Zq, etc.
%         maxfit = max. order of the poly's defined in POLY_xx
%         order = matrix defining the order (in M and h) of each poly. for
%                 each of the derivatives (i.e. Za, Zde, ...)
%                 Each row of order coeresponds to one of the derivatives,
%                 and the are listed in the order that they are given in
%                 names
% Outputs: LFTs for each of the derivatives and the A and B matrices.
%
%
% Written by: Capt Mark Spillman and Lawton Lee
%             WL/FIGC-3 Bldg 146, WPAFB

% Modified by: Capt Martin Breton, AFIT (generalized + schur redux)


% DEFINE inputs
names='Za Zq Zde Ma Mq Mde';

flag=order(2,1);
if flag == 0,
  order=[order(1,:);order(3:6,:)];
  names='Za Zde Ma Mq Mde';
  LFT_Zq=POLY_Zq;
end

tol=[];
%%%%%%%%% Parameter normalization factors and shortcut matrices to convert
```

```
%%%%%%%%% polynomial coefficients in parm to coefficients in delta_parm

% Design and spread Machs
Mbar=(Mmax+Mmin)/2;
Mtil=Mmax-Mbar;

% Design and spread altitudes
hbar = 500*(hmax+hmin);
htil = 1000*hmax-hbar;

% Create the shortcut matrices
Mmat=zeros(maxfit+1);
hmat=zeros(maxfit+1);
K=zeros(maxfit+1);kvec=1;K(1,1)=1;
for ii=1:maxfit+1
 Mmat=Mmat + diag((Mtil.^(0:maxfit-ii+1))*(Mbar^(ii-1)),ii-1);
 hmat=hmat + diag((htil.^(0:maxfit-ii+1))*(hbar^(ii-1)),ii-1);
 if ii<=maxfit
  kvec=conv(kvec,[1 -1]);
  K(1:ii+1,ii+1)=abs(kvec');
 end
end
Mmat=Mmat.*K;
hmat=hmat.*K;

%%%%%%%%% LFT contruction for Za, Zde, Ma, Mq, Mde, qbar, Vt

[num,names,namelen,maxlen] = namstuff(names);
blk=zeros(num,2);

for ii=1:num;
 OM=order(ii,1);Oh=order(ii,2);
 nM=OM*(Oh+1);nh=Oh;
 blk(ii,:)=[nM nh];
 name=names(ii,1:namelen(ii));
 eval(['c=kron(Mmat(1:OM+1,1:OM+1),hmat(1:Oh+1,1:Oh+1))*POLY_',...
        name ';'])
 eval(['LFT_' name  '=pss2sys([zeros(nM,nh+1) eye(nM);',...
        'zeros(nh,nM+1) eye(nh);c(nM+nh+1:-1:1)''],nM+nh);'])
 eval(['[LFT_' name ',nrem]=lftmin(LFT_' name ',[nM nh],1,1,1,tol);'])
 blk(ii,:)=blk(ii,:)-[nrem 0];
end
```

```
%%%%%%%%% LFT construction for lftab

LFT_top=sbs(LFT_Za,LFT_Zq,LFT_Zde);
LFT_bot=sbs(LFT_Ma,LFT_Mq,LFT_Mde);
LFT_AB=abv(LFT_top,LFT_bot);

% Separate the M and h "states"
he=cumsum(sum(blk')');me=he-blk(:,2);
hs=he-blk(:,2)+ones(num,1);ms=me-blk(:,1)+ones(num,1);

mindx=[];
hindx=[];
for i=1:num,
 mindx=[mindx ms(i):me(i)];
 hindx=[hindx hs(i):he(i)];
end
indx=[mindx hindx];

LFT_AB=reordsys(LFT_AB,indx);

% Reduce the delta blocks one at a time
tblk=sum(blk)

[LFT_AB,nrem] = lftmin(LFT_AB,tblk,1,3,2,tol);
tblk = tblk - [nrem 0];
[LFT_AB,nrem] = lftmin(LFT_AB,tblk,2,3,2,tol);
tblk = tblk - [0 nrem]

% Now try a schur model reduction

[LFT_AB,nrem] = lftmin2(LFT_AB,tblk,1,3,2,tol);
tblk = tblk - [nrem 0]
[LFT_AB,nrem] = lftmin2(LFT_AB,tblk,2,3,2,tol);
tblk = tblk - [0 nrem]

minfo(LFT_AB)

%%%%%%%%%% Clean up the workspace

clear K c Mmat hmat ms hs me he indx kvec OM Oh blk
clear LFT_top LFT_bot
```

```
clear nM nh name names namelen maxlen ii num nrem

% end of f18sppoly2lft.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.5.1 `lftmin.m`*: Minimize the LFT Realization.* Note: this is a subprogram called by both `f18poly2lft.m` and `f18sppoly2lft.m`.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

% lftmin.m

function [LFTnew,nrem] = lftmin(LFTbig,blk,pnum,ninputs,noutputs,tol);

% function [LFTnew,nrem] = lftmin(LFTbig,blk,pnum,ninputs,noutputs,tol);
%
% Minimize the block structure of an LFT by truncating
% "uncontrollable" & "unobvservable" scalar deltas
% Inputs: LFTbig The LFT to reduce
%   blk the original block structure
%   pnum the choice of parameter to be reduced
%   ninputs number of external inputs, default = 1
%   noutputs number of external outputs, default = 1
%   tol tolerance for minimal2.m
% Outputs: LFTnew reduced LFT
%          nrem number of removed states

if nargin == 3
 ninputs = 1;
 noutputs = 1;
end

n = sum(blk);
nd = blk(pnum);

[a,b,c,d] = minfo(LFTbig);
if n ~= d
 error('block structure does not add up');
end
```

```
if (noutputs ~= b | ninputs ~= c)
 error('the input-output structure does not add up');
end
if a ~= 'syst'
 error('lft must be a system matrix');
end

% Re-order the delta_i's so that delta_pnum comes first
if pnum > 1
 nold = sum(blk(1:pnum-1));
 LFTmid = reordsys(LFTbig,[nold+1:n, 1:nold]);
else
 LFTmid = LFTbig;
end

% Partition & unpack the LFT realization, isolating delta_pnum
pss = sys2pss(LFTmid);
a = pss(1:nd,1:nd);
b = pss(1:nd,nd+1:n+ninputs);
c = pss(nd+1:n+noutputs,1:nd);
d = pss(nd+1:n+noutputs,nd+1:n+ninputs);

% Remove "uncontrollable, unobservable" delta channels
if isempty(tol)==0
 [am,bm,cm,dm] = minimal2(a,b,c,d,tol);
else
 [am,bm,cm,dm] = minimal2(a,b,c,d);
end

ndnew = max(size(am));
nrem = nd - ndnew;
disp([int2str(nrem) ' state(s) removed']);

%ashift= am+eye(ndnew); % shift to handle poles at origin
%[am,bm,cm,dm,errbnd,hsv] = schmr(ashift,bm,cm,dm,2,1e-3);
%ndnewer = max(size(am));
%am= am - eye(ndnewer); % shift back

%nrem= nd - ndnewer;

if nrem > 0
 pss = [am,bm;cm,dm];
```

```
 LFTnew = pss2sys(pss,max(size(pss))-max(ninputs,noutputs));

 % Shuffle the deltas back to their original order
 if pnum > 1
  LFTnew = reordsys(LFTnew,[n-nold-nrem+1:n-nrem, 1:n-nold-nrem]);
 end
else
 LFTnew = LFTbig;
end

% end of lftmin.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
```

C.6 f18ol.m*: Convert Full Longitudinal LFT to Design Model*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18ol.m

function [P,Pn,plant,qmax,err,Wp,Wh,Wr,Wu,Wn,Wc]=f18ol(LFT_AB,tblk,qmax,err,Wp,Wu,W

% f18ol
%[P,Pn,plant,qmax,err,Wh,Wp,Wr,Wu,Wn] = f18ol(LFT_AB,tblk,qmax,err)
%
% This function forms the open-loop interconnection for an F-18
% full longitudinal H_inf LPV design problem, using LFT parameter dependence.
% The user can specify the maximum pitch rate, maximum tracking error, and
% command, ideal and performance models if desired.
%
% Inputs:  LFT_AB Reduced LFT for [A,B] (system matrix)
% tblk [size of parameter1 size of parameter2]
%               qmax            Maximum pitch rate (deg/sec)
%               err             Maximum pitch rate tracking error
%                               example: if err=10, max error < .1*qmax
%
% Outputs: P      weighted open-loop interconnection w/ uncertainty (sys matrix)
%          Pn     weighted open-loop ic w/o uncertainty (sys matrix)
%                 NOTE: both P and Pn are LPV
%          plant  Plant only (sys matrix)
```

```
%          qmax   Maximum pitch rate (deg/sec)
%          err    Maximum pitch rate tracking error (%qmax)
%          W's    All the weightings
%

if nargin == 0
 disp('Usage: [P,Pn,plant,qmax,err,Wp,Wh,Wr,Wu,Wn,Wc] = ',...
  'f18ol(LFT_AB,tblk,qmax,err,Wp,Wu,Wc);')
 return;
end

if nargin<7;Wc=[];end
if nargin<6;Wu=[];end
if nargin<5;Wp=[];end


% Command Weight
% Need to be careful putting dynamic filters here.  They tend to cause a
% an unacceptable time delay in the pitch time response.
%wn1=4;
%Wr=nd2sys(wn1^2,conv([1 wn1],[1 wn1]),qmax*pi/180);
%Wr=qmax*pi/180;
Wr=1;

% Performance weight
if Wp==[]
 Wp=nd2sys([1 50],[1 4],4/50/(err/100*qmax*pi/180));% Less than err% ss error
 %Wp=nd2sys([1 50],[1 4],4/50*.1745);
 %Wp=err/100*qmax*pi/180;
end

% Ideal Model
% Overshoot and ss error are less critical than the speed of the response.
zeta=.5;
wn2=4;
Wh=nd2sys(wn2^2,[1 2*zeta*wn2 wn2^2]);

% Noise weight Wn(s)
% The small amount of noise below was added to make the weighted plant regular
sig_a = .01; % AOA noise variance (rad), realistic value =.01
sig_q = .01;           % rate gyro variance (rad), realistic value =.03
Wn = daug(sig_a,sig_q);
```

```
% Actuator model
au=20.2;                              % Elevator bandwidth
Act=pck(-au,au,[1;-au],[0;au]);  % output = [u;du/dt]


% Control Weight
if Wc==[]
 Wc=1/70;
end

% LPV plant with LFT parameter dependence
[ns,nx,nxu,n0] = minfo(LFT_AB);
nu = nxu - nx;
[d00,blft,b0,dlft] = unpck(LFT_AB); % Unpack LFT realization
a0 = dlft(:,1:nx);
b2 = dlft(:,nx+1:nxu);
c0 = blft(:,1:nx);
d02 = blft(:,nx+1:nxu);
%c2 = [0 1 0 0;0 0 1 0]; % form c2 and d22
c2=[0 1 0 0;0 0 1 0];
[ny dum] = size(c2);
d20 = zeros(ny,n0);
d22 = zeros(ny,nu);
plant = pck(a0,[b0 b2],[c0;c2],[d00 d02;d20 d22]);

% Dummy is used to make sure the number of controlled outputs equals
% the number of exogenous inputs
dummy=0;

msize=tblk(1);
hsize=tblk(2);
delsize=msize+hsize;
aoa=delsize+1;
q=delsize+2;

% Weighted plant without uncertainty
systemnames = 'plant Act Wp Wc Wh Wr Wn dummy';
inputvar = '[delm(5);delh(5);ref_cmd;noise(2);cont_cmd]';
outputvar = '[plant(1:10);Wp;Wc;dummy;Wr;plant(11:12)+Wn]';
input_to_Act = '[cont_cmd]';
input_to_Wr = '[ref_cmd]';
```

```
input_to_Wc = '[Act(2)]';
input_to_Wp = '[Wh-plant(12)]';
input_to_plant = '[delm;delh;Act(1)]';
input_to_Wh = '[Wr]';
input_to_Wn = '[noise]';
input_to_dummy = '[ref_cmd]';
cleanupsysic = 'yes';
sysoutname = 'f18ic_n';
sysic;
Pn=f18ic_n;

% Uncertainty weight
if Wu==[]
 Wu = nd2sys([1 40],[1 10000],10000/40*1e-2);
end

% Weighted plant with input additive uncertainty
systemnames = 'plant Act Wp Wc Wh Wr Wu Wn dummy';
inputvar = '[delm(5);delh(5);unc_in;ref_cmd;noise(2);cont_cmd]';
outputvar = '[plant(1:10);Wu;Wp;Wc;dummy;Wr;plant(11:12)+Wn]';
input_to_Wr = '[ref_cmd]';
input_to_Act = '[cont_cmd]';
input_to_Wc = '[Act(2)]';
input_to_Wp = '[Wh-plant(12)]';
input_to_plant = '[delm;delh;Act(1)+unc_in]';
input_to_Wh = '[Wr]';
input_to_Wu = '[Act(1)]';
input_to_Wn = '[noise]';
input_to_dummy = '[ref_cmd]';
cleanupsysic = 'yes';
sysoutname = 'f18ic';
sysic;
P=f18ic;

% end f18ol.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.7 `f18olsp.m`: *Convert Short Period LFT to Design Model*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18olsp.m

function [P,Pn,plant,qmax,err,Wp,Wh,Wr,Wu,Wn,Wc]=f18olsp(LFT_AB,tblk,qmax,err,Wp,Wu

% f18ol
%[P,Pn,plant,qmax,err,Wh,Wp,Wr,Wu,Wn] = f18olsp(LFT_AB,tblk,qmax,err)
%
% This function forms the open-loop interconnection for an F-18
% full longitudinal H_inf LPV design problem, using LFT parameter dependence.
% The user can specify the maximum pitch rate, maximum tracking error, and
% command, ideal and performance models if desired.
%
% Inputs:  LFT_AB Reduced LFT for [A,B] (system matrix)
% tblk [size of parameter1 size of parameter2]
%                qmax              Maximum pitch rate (deg/sec)
%                err               Maximum pitch rate tracking error
%                                  example: if err=10, max error < .1*qmax
%
% Outputs: P      weighted open-loop interconnection w/ uncertainty (sys matrix)
%          Pn     weighted open-loop ic w/o uncertainty (sys matrix)
%                NOTE: both P and Pn are LPV
%          plant  Plant only (sys matrix)
%          qmax   Maximum pitch rate (deg/sec)
%          err    Maximum pitch rate tracking error (%qmax)
%          W's    All the weightings
%

if nargin == 0
 disp('Usage: [P,Pn,plant,qmax,err,Wp,Wh,Wr,Wu,Wn,Wc] = ',...
  'f18olsp(LFT_AB,tblk,qmax,err,Wp,Wu,Wc);')
 return;
end

if nargin<7;Wc=[];end
if nargin<6;Wu=[];end
if nargin<5;Wp=[];end


% Command Weight
% Need to be careful putting dynamic filters here.  They tend to cause a
```

```
% an unacceptable time delay in the pitch time response.
%wn1=4;
%Wr=nd2sys(wn1^2,conv([1 wn1],[1 wn1]),qmax*pi/180);
%Wr=qmax*pi/180;
Wr=1;

% Performance weight
if Wp==[]
 Wp=nd2sys([1 50],[1 4],4/50/(err/100*qmax*pi/180));% Less than err% ss error
 %Wp=nd2sys([1 50],[1 4],4/50*.1745);
 %Wp=err/100*qmax*pi/180;
end

% Ideal Model
% Overshoot and ss error are less critical than the speed of the response.
zeta=.5;
wn2=4;
Wh=nd2sys(wn2^2,[1 2*zeta*wn2 wn2^2]);

% Noise weight Wn(s)
% The small amount of noise below was added to make the weighted plant regular
sig_a = .01; % AOA noise variance (rad), realistic value =.01
sig_q = .01;           % rate gyro variance (rad), realistic value =.03
Wn = daug(sig_a,sig_q);

% Actuator model
au=20.2;                              % Elevator bandwidth
Act=pck(-au,au,[1;-au],[0;au]);  % ouput = [u;du/dt]


% Control Weight
if Wc==[]
 Wc=1/70;
end

% LPV plant with LFT parameter dependence
[ns,nx,nxu,n0] = minfo(LFT_AB);
nu = nxu - nx;
[d00,blft,b0,dlft] = unpck(LFT_AB); % Unpack LFT realization
a0 = dlft(:,1:nx);
b2 = dlft(:,nx+1:nxu);
c0 = blft(:,1:nx);
```

```
d02 = blft(:,nx+1:nxu);
c2 = eye(2); % form c2 and d22
[ny dum] = size(c2);
d20 = zeros(ny,n0);
d22 = zeros(ny,nu);
plant = pck(a0,[b0 b2],[c0;c2],[d00 d02;d20 d22]);

% Dummy is used to make sure the number of controlled outputs equals
% the number of exogenous inputs
dummy=0;

msize=tblk(1);
hsize=tblk(2);
delsize=msize+hsize;
aoa=delsize+1;
q=delsize+2;

% Weighted plant without uncertainty
systemnames = 'plant Act Wp Wc Wh Wr Wn dummy';
inputvar = '[delm(3);delh(3);ref_cmd;noise(2);cont_cmd]';
outputvar = '[plant(1:6);Wp;Wc;dummy;Wr;plant(7:8)+Wn]';
input_to_Act = '[cont_cmd]';
input_to_Wr = '[ref_cmd]';
input_to_Wc = '[Act(2)]';
input_to_Wp = '[Wh-plant(8)]';
input_to_plant = '[delm;delh;Act(1)]';
input_to_Wh = '[Wr]';
input_to_Wn = '[noise]';
input_to_dummy = '[ref_cmd]';
cleanupsysic = 'yes';
sysoutname = 'f18ic_n';
sysic;
Pn=f18ic_n;

% Uncertainty weight
if Wu==[]
 Wu = nd2sys([1 40],[1 10000],10000/40*1e-2);
end

% Weighted plant with input additive uncertainty
systemnames = 'plant Act Wp Wc Wh Wr Wu Wn dummy';
inputvar = '[delm(3);delh(3);unc_in;ref_cmd;noise(2);cont_cmd]';
```

```
outputvar = '[plant(1:6);Wu;Wp;Wc;dummy;Wr;plant(7:8)+Wn]';
input_to_Wr = '[ref_cmd]';
input_to_Act = '[cont_cmd]';
input_to_Wc = '[Act(2)]';
input_to_Wp = '[Wh-plant(8)]';
input_to_plant = '[delm;delh;Act(1)+unc_in]';
input_to_Wh = '[Wr]';
input_to_Wu = '[Act(1)]';
input_to_Wn = '[noise]';
input_to_dummy = '[ref_cmd]';
cleanupsysic = 'yes';
sysoutname = 'f18ic';
sysic;
P=f18ic;

% end of f18olsp.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.8 dkdM18.m*: Perform D-K-D Iterations*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% dkdM18.m

% this is a modification/extension of MATLAB's dkit

%DEFINE SOME THINGS BELOW
clear
format short e;
f18poly % for full longitudinal design
f18poly2lft
w=logspace(-6,6,100);

%f18sppoly % for short period design
%f18sppoly2lft

% Pick some weights
Wp=nd2sys([1 100],[1 4],.4/10.2);
%Wp=nd2sys([1 10000],[1 4],4/10000*10);
```

```
%Wp=10;
%Wp=nd2sys([1 4],[1 50],50/4*20);
Wu=nd2sys([1 100],[1 10000],10000/100*1e-3);
%Wc=1;
Wc=1/70;
Wn=.01;

% Plot the frequency responses of the weights
Wpfrsp=frsp(Wp,w);
Wufrsp=frsp(Wu,w);
Wcfrsp=frsp(Wc,w);
Wnfrsp=frsp(Wn,w);
vplot('liv,lm',Wpfrsp,'r-',Wufrsp,'w-',Wcfrsp,'b-',Wnfrsp,'g-')
title('Weights')
legend('r-','Wp','w-','Wu','b-','Wc','g-','Wn')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')
axis([ 1e-1 1e6 1e-4 1e2])
pause(1)

% Form the weighted plant
[P,Pn,plant,qmax,err,Wp,Wh,Wr,Wu,Wn,Wc] = f18ol(LFT_AB,tblk,1,15,Wp,Wu,Wc);
%[P,Pn,plant,qmax,err,Wp,Wh,Wr,Wu,Wn,Wc] = f18olsp(LFT_AB,tblk,1,15,Wp,Wu,Wc);

msize=tblk(1); % tblk defined in f18poly2lft call
hsize=tblk(2);
delsize=msize+hsize;
aoa=delsize+1;
q=delsize+2;

%%%% THE FOLLOWING DEFINES THE DIMENSIONS OF THE SYSTEM %%%%%

nmtvb=4;  % # of measured time varying blocks (2 parameters for both P,K)
ntvb=nmtvb; % # of time varying blocks
       % NOTE: in general ntvb<=nmtvb, but currently let be =

% define dimension of parameter blocks (theata_p and theata_k)
nv_p=delsize;
nz_p=nv_p;
nv_k=nv_p;
nz_k=nv_k;
```

```
% define dimension of uncertain block (delta)
nv_u=1;
nz_u=1;

% define dimension of performance block (delta_perf)
nd=3;
ne=3;

% define dimension of the "performance" block used for lmi
% this includes d to e plus any unmeasured uncertainty (nv_u to nz_u)
nw=nd+nv_u;  % currently assume ne+nz_u=nd+nv_u;

% define dimension of measures and actuation of P
ny_p=3;
nu_p=1;

% calculate the number of I/O's of the controller
ny=ny_p+nv_k; % # inputs to K
nu=nu_p+nz_k; % # outputs from K


% define block strunctures
blk_p=[msize,0;hsize,0];
blk_k=blk_p;
blk_pk=[blk_p;blk_k];
blk_wu=[blk_k;blk_p;nv_u,nz_u;nd,ne];  % blk with uncertainty and d2e

% NOW GET SOME BLOCKS FOR A REORDERED PARAMETER BLOCK
% NOTE: THESE ARE USED FOR IN A MU CALCULATION FOR COMPARISON
blk_pkro=blk_p*2; % blk_pk reordered (i.e. after par_rord.m is run)
blk_wuro=[-blk_pkro;nv_u,nz_u;nd,ne]; % blk with uncertainty and d2e, used w/mu
% '-' for REAL parameters

% remove any zero dimension blks  in the blocks
z_ind=find(blk_wu(:,1)==0);
blk_wu(z_ind,:)=[];

z_ind=find(blk_wuro(:,1)==0);
blk_wuro(z_ind,:)=[];
```

```
%%%%% set some options %%%%%%%%

key_opt=0; % if 1 stops for input from keyboard once every iteration

%%%%%%%%%%%%%%%%%%%%%%%% END OF USER INPUTS %%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%% CALCULATE SOME STUFF %%%%%
nv=nv_p+nv_k+nv_u;
nz=nz_p+nz_k+nz_u;

nim=nv+nd; % # in M, where M=closed loop
nom=nz+ne; % # out M

blk_dim=dim_blk(blk_wu);

nb=ynum(blk_wu);

blk_c=[colsum(blk_dim(1:nmtvb,:)); % this blk is used w/ mu when const. D's
       blk_wu(nmtvb+1:nb,:)]; % are already wrapped in

% initilize D's
Dl=eye(nom);
Dr=eye(nim);
Dri=eye(nim);

% need a P that has proper inputs and outputs so L, J, and K
% can be folded into P
% i.e. want to pass parameters from DELTA thru P to K

Pk=param_k(P,nz_p,nv_p);

% now reorder the paramater channels so each parameter is grouped
% see ">>help par_rord" for more info
Pkro=par_rord(Pk,blk_p);


if(key_opt==1), keyboard; end


%%%%%%%%%%%%%%%%%%%%%%%% START LOOP %%%%
k_count=0;  % # of controllers found
```

```
dodkit=1;

while(dodkit==1);

  if(k_count>0)

    [Dsl,Dsr]=dbfit(DcMDci,blk_wu,nmtvb,w,mmb_DcMDci,rowd_DcMDci,sns_DcMDci);
% similar to musynfit

    Dsri=minv(Dsr);

    Dl=mmult(Dsl,Dcl);
    Dri=mmult(Dcri,Dsri);

    DMDi=mmult(Dl,M,Dri);
    DMDig=frsp(DMDi,w);
    msv_DMDi=sel(vsvd(DMDig),1,1);

    % determine what type of plot to use
    if(pkvnorm(mmb_DcMroDci)<3 & pkvnorm(mmb_DcMDci)<3 & pkvnorm(msv_DMDi)<3)
      plot_type='liv,m';
    else
      plot_type='liv,lm';
    end

    vplot(plot_type,mmb_DcMroDci,'r',mmb_DcMDci,'g',msv_DMDi,'b')
    title('mu_up(DcMroDci)=red, mu_up(DcMDci)=green, msv(DMDi)=blue')
    pos(['@ The red is the mu upper bound using dynamic D scales',...)
         '@ The green is an actual mu upper bound using constant and',...
         ' dynamic D scales ',...
         '@ The blue is the msv of the closed loop when TFs are used in',...
         ' place of the @ dynamic D scales of the green. ',...
    '@ @ NOTE: if the option "cmp_opt" is set to zero there is no red curve ']);

    %% the following is a check to see what the hinf norm of the weighted
    %% (by Ds, L, and J) closed loop system is using L, J, and K from the
    %% previous step, the hinf norm after finding new L, J, and K should
    %% be less than or equal to this norm; worst case it would be equal if
    %% the minimization returned the same L, J, and K as the previous step
    hinfnorm(DMDi)
    disp('The hinf norm after the next step should be <= that shown above')
    %pos('The hinf norm after the next step should be <= that shown above');
```

```
  else
    Dsl=eye(nom);
    Dsri=eye(nim);
  end

  % NOW need to scale P so can iterate
  % First scale Pk, then select P
  Dslp=daug(Dsl,eye(ny));
  Dsrip=daug(Dsri,eye(nu));
  DsPkDsi=mmult(Dslp,Pk,Dsrip);
  DsPDsi=sel(DsPkDsi,nz_k+1:nz+ne+ny_p,nv_k+1:nv+nd+nu_p);

%%%% temp
  if k_count==0
    [k,DwMDwi,gf]=hinfsyn(DsPDsi,ny,nu,.2,200,1e-2); % checks LTI controller
    [k,DwMDwi,gf]=hinfsyn(Pk,ny,nu,.2,1000,1e-2); % checks unscaled LPV
 end
%%%%%%% end temp

  % CALL LMI FUNCTION THAT RETURNS K AND Dc

  % do LMI iterations to determine gamma and find L,J

  gamma=input('ENTER VALUE OF TARGET GAMMA, OR 999 FOR GEVP     ');
  while looptst2(gamma,1)
    gamma=input('try again - ENTER VALUE OF TARGET GAMMA OR 999     ');
  end
  if gamma==999
    opt=[1e-2 200 1e7 5 0];
    [Ps,R,S,L,J,gamma]=nshinflmi2(DsPDsi,abs(blk_p),nw,opt,0);
  else
    disp('BEGINNING OF ITERATION LOOP');
    gtemp=-1;
    while gamma~=999
      disp(['target gamma set at ' num2str(gamma)]);
      opt=[0 200 1e7 30 0];
      [Ps,R,S,L,J,t]=nshinflmi(DsPDsi,abs(blk_p),nw,gamma,opt,-1e-8);
      if t<0
        gtemp=gamma;
        Ltemp=L;
        Jtemp=J;
```

```
      Ptemp=Ps;
    end
    gamma=input('ENTER NEW TARGET GAMMA, OR 999 TO MOVE ON     ');
    while looptst2(gamma,1)
      gamma=input('try again - ENTER VALUE OF TARGET GAMMA OR 999     ');
    end
  end
  if gtemp>-1
    disp(['the solution for gamma = ' num2str(gtemp) ' will be used']);
    gamma=gtemp;
    L=Ltemp;
    J=Jtemp;
    Ps=Ptemp;
  else
    error('Problem is not feasible.  Aborting dkdit.m');
  end
 end

[K]=hinfsyn(Ps,nv_p+ny_p,nv_p+nu_p,0,gamma,1e-2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k_count=k_count+1;
eval(['K_',num2str(k_count),'=K;']);

% reassign L to Dcl and J to Dcri (D constant left and right inverse)
Dcl=sqrtm(L);
Dcri=sqrtm(J);
%  Dcri=Dcl\eye(2*nv_p+nw)
%  Dcl=chol(L);
%  Dcri=Dcl\eye(2*nv_p+nw);

% form closed loop
M=starp(Pk,K);
Mg=frsp(M,w);
DcMDci=mmult(Dcl,M,Dcri);
DcMDcig=mmult(Dcl,Mg,Dcri);     % scale closed loop freq resp

%% CHECK THE HINF NORM WITH THAT OBTAINED BY LMI
%% The following checks the hinf norm of the weighted (by Ds, L, and J)
%% closed loop system using the L, J, and K just obtained and the Ds
%% from the previous step, this should be similar to the gamma obtained
%% by the LMI feasability search AND should be <= the norm checked using
```

```
  %% the L, J, and K from previous step, which was checked above
  DcDsMDsiDci=mmult(Dcl,starp(DsPkDsi,K),Dcri);
  hinfnorm(DcDsMDsiDci)
  disp('The hinf norm of the new closed loop system is shown above')
  %pos([' The hinf norm of the new closed loop system is shown above',...
  %   '@ Check this value with that of gfin given by the LMI']);

  [bnd_DcMDci,rowd_DcMDci,sns_DcMDci,rp]=mu2(DcMDcig,blk_c);
  mmb_DcMDci=sel(bnd_DcMDci,1,1);
  pk_mmb_DcMDci=pkvnorm(mmb_DcMDci);

  disp(' ')
  disp('The peak of the mu upper bound (green curve) is');
  disp(pk_mmb_DcMDci);


dodkit=input('Press return to do real mu, or 1 to skip it. ');
 if (isempty(dodkit))

  %% FORM CLOSED LOOP WITH REORDERED SYSTEM SO CAN DO A MU CALC FOR COMPARISON
  DcMroDci=par_rord(DcMDci,blk_p);
  DcMroDcig=frsp(DcMroDci,w);
  [bndro,d_DcMroDci,sns_DcMroDci,rp]=mu(DcMroDcig,blk_wuro);
  mmb_DcMroDci=sel(bndro,1,1);    % mu upper bound with full D's on theta block
  pk_mmb_DcMroDci=pkvnorm(mmb_DcMroDci);

  disp(' ')
  disp('The peak of the (real theta) mu upper bound (red curve) is');
  disp(pk_mmb_DcMroDci);

 else
    mmb_DcMroDci=mmb_DcMDci;
 end

  % determine what type of plot to use
  if(pkvnorm(mmb_DcMroDci)<3 & pkvnorm(mmb_DcMDci)<3)
    plot_type='liv,m';
  else
    plot_type='liv,lm';
  end

  vplot(plot_type,mmb_DcMroDci,'r',mmb_DcMDci,'g')
```

```
  eval(['plot_type_',num2str(k_count),'=plot_type;']);
  eval(['mmb_DcMroDci_',num2str(k_count),'=mmb_DcMroDci;']);
  eval(['mmb_DcMDci_',num2str(k_count),'=mmb_DcMDci;']);

  title('mmb_DcMroDci (red), mmb_DcMDci (green), NOTE: green is a true upper bound'
  disp([' @ The green plot is a true upper bound on mu for this system.',...
    '@ The red plot is the upper bound using dynamic scales for all blocks',...
    '@ @ NOTE: red curve scales can only be kept in last iteration '])

%%%%%%%%%%%%%%%%%%% DO SOME ANALYSIS %%%%%%%%
  do_m=input('Would you like to do an M analysis? y=yes, return=no ','s');
  if(~isempty(do_m))
    %%%%% look at some M analysis to see what is driving the problem %%%%%
    [dl,dr]=unwrapd2(rowd_DcMDci,blk_c);
    dmdig=mmult(dl,DcMDcig,vinv(dr));
    msv_DcMDci=vnorm(dmdig);
    blk_m=[nv,nz;nd,ne];
    msv=blknorm(dmdig,blk_m);
    msv_zv=sel(msv,1,1);
    msv_zd=sel(msv,1,2);
    msv_ev=sel(msv,2,1);
    msv_ed=sel(msv,2,2);

    subplot(2,2,1), vplot(plot_type,mmb_DcMDci,'r',msv_zv,'g')
    title('Robust Stability, mu(Mzv)')
    subplot(2,2,2), vplot(plot_type,mmb_DcMDci,'r',msv_zd,'g')
    title('msv(DMzdDi)')
    subplot(2,2,3), vplot(plot_type,mmb_DcMDci,'r',msv_ev,'g')
    title('msv(DMevDi)')
    subplot(2,2,4), vplot(plot_type,mmb_DcMDci,'r',msv_ed,'g')
    title('Nominal Performance, msv(Med)')
    pos('p')

    blk_m2=[nv-nv_u,nz-nz_u;nv_u,nz_u;nd,ne];
    msv2=blknorm(dmdig,blk_m2);
    % note: p denotes parameters and u denotes uncertainty
    msv_pp=sel(msv2,1,1);
    msv_pu=sel(msv2,1,2);
    msv_pd=sel(msv2,1,3);
    msv_up=sel(msv2,2,1);
    msv_uu=sel(msv2,2,2);
    msv_ud=sel(msv2,2,3);
```

```
   msv_ep=sel(msv2,3,1);
   msv_eu=sel(msv2,3,2);
   msv_ee=sel(msv2,3,3);

   subplot(3,3,1), vplot(plot_type,mmb_DcMDci,'r',msv_pp,'g')
   title('Parameter input')
   ylabel('Parameter output')
   subplot(3,3,2), vplot(plot_type,mmb_DcMDci,'r',msv_pu,'g')
   title('Uncertainty input')
   subplot(3,3,3), vplot(plot_type,mmb_DcMDci,'r',msv_pd,'g')
   title('Disturbance/Reference input')
   subplot(3,3,4), vplot(plot_type,mmb_DcMDci,'r',msv_up,'g')
   ylabel('Uncertainty output')
   subplot(3,3,5), vplot(plot_type,mmb_DcMDci,'r',msv_uu,'g')
   subplot(3,3,6), vplot(plot_type,mmb_DcMDci,'r',msv_ud,'g')
   subplot(3,3,7), vplot(plot_type,mmb_DcMDci,'r',msv_ep,'g')
   ylabel('Error/Performance output')
   subplot(3,3,8), vplot(plot_type,mmb_DcMDci,'r',msv_eu,'g')
   subplot(3,3,9), vplot(plot_type,mmb_DcMDci,'r',msv_ed,'g')
   pos('p')
   subplot(1,1,1)

end  % M analysis

%%%% Check convergence %%%%%
eval(['L_',num2str(k_count),'=L;']);
rd=rowd_DcMDci;
eval(['rd_',num2str(k_count),'=rd;']);

if(k_count>1)
  disp('CHECKING CONVERGENCE');
  eval(['L_dif=L_',num2str(k_count),'-L_',num2str(k_count-1),';']);
  max_L_dif=max(max(L_dif));
  disp(['The max difference between L(i) and L(i-1) is ',num2str(max_L_dif)]);

  [type,nr,rc,npts]=minfo(rd);
  eval(['rdi=rd_',num2str(k_count),';']);
  eval(['rdim1=rd_',num2str(k_count-1),';']);
  for k=1:rc;
    vplot('liv,lm',sel(rdi,1,k),'r',sel(rdim1,1,k),'g')
    title(['Dscale #',num2str(k),' from this iteration and the previous one'])
    pos('p')
```

```
    end
  end
%%%%%%%%%%%%%%%%%%%%%%%%% END OF ANALYSIS %%%%%%


  if(key_opt==1), keyboard; end

  dodkit=input('Press return to do another iteration, or 1 to quit. ');
  if (isempty(dodkit))
    dodkit=1;
  else
    dodkit=0;
  end

end  % while

% end dkdM18.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.8.1 param_k.m*: Add Controller Parameter Block to Design Model.*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% param_k.m

% this function converts an LPV plant into an LPV
% plant that "passes" the same parameters to the controller
%
% function sys_out=param_k(sys_in,nz,nv);
%
%   where, sys_in=LPV plant
%            where, the original LPV system is given by
%                   LPV_orig=starp(del,sys_in)
%          nz=# of inputs to the delta block (i.e. # of "uncertain"
%             (parameter) outputs from the original plant
%          nv=# of outputs from the delta block (i.e. # of "uncertain"
%             (parameter) inputs to the original plant
%
%  the new LPV system is given by LPV=starp(del_n,sys_out)
%      where,    del_n = [delk      |
```

```
%                          |      del]
%

function sys_out=param_k(sys_in,nz,nv);

[a,b,c,d]=unpck(sys_in);

[mtype,npo,npi,nx]=minfo(sys_in);

a_new=a;

b_new=[zeros(nx,nv),b,zeros(nx,nz)];

c_new=[zeros(nz,nx);c;zeros(nv,nx)];

d_new=[zeros(nz,nv),zeros(nz,npi),eye(nz);
       zeros(npo,nv),d,zeros(npo,nz);
       eye(nv),zeros(nv,npi),zeros(nv,nz)];

sys_out=pck(a_new,b_new,c_new,d_new);

% end of param_k.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.8.2 par_rord.m*: Reorder Parameter Block for μ-Analysis.*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

% par_rord.m

% this function modifies the LPV plant generated in param_k.m which
% "passes" the same parameters that the plant varies with to the controller
%
% The modification is simply a reordering of the "uncertainty"(/parameter) I/O's
% (i.e. inputs/outputs) so that the final delta (which includes the plant
% and the controller subdeltas) contains only scalar*Identities with each scalar
% appearing only once (i.e. converts del_k_p to del
%      where,    del_k_p = [delk     |
%                           |      delp]
%
```

```
%           where, delk=diag(d1*Ik1,d2*Ik2,...) and delp=diag(d1*Ip1,d2*Ip2,...)
%                                NOTE: assumed here that delk=delp i.e. Ik1=Ip1 etc.
%   then,
%               del=[d1*diag(Ik1,Ip1)                                  |
%                   |                  d2*diag(Ik2,Ip2)              |
%                   |                                            ... ]
%
% function sys_out=par_rord(sys_in,nB,B_dim)
%
%   where, sys_in=LPV plant generated by param_k.m
%          nB= # of blocks in each del (i.e. # of blocks in delp)
%          B_dim=a vector containing the dimension of each block of each
%                del(e.g. delp) =[dim1;dim2;...]
%
% **** OR ****
%
% function sys_out=par_rord(sys_in,blk_p)
%
%   where, sys_in=LPV plant generated by param_k.m
%          blk_p=an nB by 2 matrix that defines the structure of the
%                parameter block of P (i.e. delp)
%

function sys_out=par_rord(sys_in,nB_or_blk_p,B_dim)

if(nargin<2 | nargin>3)
  disp('useage: sys_out=par_rord(sys_in,nB,B_dim)');
  return
elseif(nargin==2)
  blk_dim=dim_blk(nB_or_blk_p);
  nB=ynum(nB_or_blk_p);
  B_dim=blk_dim(:,1);
elseif(nargin==3)
  nB=nB_or_blk_p;
end

[mtype,npo,npi,nx]=minfo(sys_in);

B_sum=sum(B_dim);

parm_order=[1:B_dim(1),B_sum+1:B_sum+B_dim(1)];
```

```
if(nB > 1)
  for blk_n=2:nB;
    sbdbnm1=sum(B_dim(1:blk_n-1));
    sbdbn=sum(B_dim(1:blk_n));

    parm_order=[parm_order,sbdbnm1+1:sbdbn,B_sum+sbdbnm1+1:B_sum+sbdbn];
  end
end

out_order=[parm_order,B_sum*2+1:npo];
in_order=[parm_order,B_sum*2+1:npi];

sys_out=sel(sys_in,out_order,in_order);

% end of par_rord.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.9 `nshinflmi.m`*: Solve LMI Feasability Problem*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% nshinflmi.m

function [Ps,R,S,L,J,t]=nshinflmi(P,blk,nw,gamma,opt,goal)
%
% NSHINFLMI  Find the solution to the Apkarian's scaled Hinf optimization
%            problem.  The problem is formulated as a Feasibility problem using
%            LMIs.  The t value displayed must be negative in order for a
%            solution to exist for the given gamma.
%
% usage:   [R,S,L,J]=nshinflmi(P,blk,nw,gamma,opt,goal)
%
% inputs:  P     - Open-loop plant in packed notation. i.e. P=pck(A, ...
%                  [Bt B1 B2], [Ct;C1;C2], [Dtt ...;D1t ...; D2t ...]),
%                  where t denotes the parameter influence
%          blk   - nm x 2 matrix describing the structure of the parameter
%                  block. The first element in each row denotes the size of
%                  the subblock.  The second element in each row is either
%                  0 (for repeated scalar) or 1 (for full).
```

```
%          nw    - Number of exogeneous inputs (must equal the number of
%                  controlled outputs)
%          gamma - Desired Hinf norm
%          opt   - Optional vector of parameter to pass to the LMI
%                  feasibility solver. opt=[ 0 (iters)  (feas. rad.)...
%                  (slow prog. tol)  (print)], default=[0 200 1e7 30 0].  See
%                  LMI Lab notes for more details.
%          goal  - Optional target for feasibility problem, default=-1e-8.
%                  See LMI Lab notes for more details.
%
% outputs: R, S - Solutions to the feasibility problem
%          L, J - The complete scaling matrices derived from L3 and J3
%    Ps    - The COMPLETE weighted plant scaled by L and J
%    t - t<=0 indicates the target gamma was feasible
%
% Written by: Capt Mark Spillman
%             WL/FIGC-3 Bldg 146, WPAFB
%
% Modified by: Capt Martin Breton, AFIT

if nargin<1
   disp('   usage:   [Ps,R,S,L,J,t]=nshinflmi(P,blk,nw,gamma,opt,goal);');
   disp(' ');
   return
end

% The equations described below are taken from "A Convex Characterization of
% Gain-Scheduled Hinf Controllers"

% Determine optional inputs
if nargin<6;goal=-1e-8;end
if nargin<5;opt=[0 200 1e7 30 0];end

% Get nm and change blk to denote the structure of L3 and J3
nm=sum(blk(:,1));
blk(:,2)=-(blk(:,2)-1);

% Set nz=nw, a current restriction of the code
nz=nw;

% Unpack P and spilt apart the elements in B, C and D
[A,B,C,D]=unpck(P);
```

```
na=size(A,1);nu=size(B,2)-nw-nm;ny=size(C,1)-nz-nm;
Bt=B(:,1:nm);B1=B(:,nm+1:nm+nw);B2=B(:,nm+nw+1:nm+nw+nu);
Ct=C(1:nm,:);C1=C(nm+1:nm+nz,:);C2=C(nm+nz+1:nm+nz+ny,:);
Dtt=D(1:nm,1:nm);Dt1=D(1:nm,nm+1:nm+nw);Dt2=D(1:nm,nm+nw+1:nm+nw+nu);
D1t=D(nm+1:nm+nz,1:nm);D11=D(nm+1:nm+nz,nm+1:nm+nw);
D12=D(nm+1:nm+nz,nm+nw+1:nm+nw+nu);
D2t=D(nm+nz+1:nm+nz+ny,1:nm);D21=D(nm+nz+1:nm+nz+ny,nm+1:nm+nw);
D22=D(nm+nz+1:nm+nz+ny,nm+nw+1:nm+nw+nu);

% tolerances (taken from Gahinet's routines)
macheps=mach_eps;
tolsing=sqrt(macheps);
toleig=macheps^(2/3);

% Make the appropriate substitutions to use a part of Gahinet's
% routine below
DD12=[Dt2;D12];DD21=[D2t D21];

% The following was taken from Gahinet's goptlmi.m:

%*************************************************************
% For numerical stability of the controller computation,
% zero the sing. values of DD12 s.t  || B2 DD12^+ || > 1/tolsing

[u,s,v]=svd(DD12);
abstol=max(toleig*norm(B2,1),tolsing*s(1,1));
ratio=max([s;zeros(1,size(s,2))])./...
      max([tolsing*abs(B2*v);abstol*ones(1,nu)]);
ind2=find(ratio < 1); l2=length(ind2);
if l2 > 0, s(:,ind2)=zeros(nz,length(ind2)); DD12=u*s*v'; end

[u,s,v]=svd(DD21');
abstol=max(toleig*norm(C2,1),tolsing*s(1,1));
ratio=max([s;zeros(1,size(s,2))])./...
      max([tolsing*abs(C2'*v);abstol*ones(1,ny)]);
ind2=find(ratio < 1); l2=length(ind2);
if l2 > 0, s(:,ind2)=zeros(nm+nw,length(ind2)); DD21=v*s'*u'; end

% compute the outer factors
Nr=lnull([B2;DD12],0,tolsing);
cnr=size(Nr,2);
Nr=[Nr zeros(na+nm+nz,nm+nw);zeros(nm+nw,cnr) eye(nm+nw)];
```

```
Ns=rnull([C2,DD21],0,tolsing);
cns=size(Ns,2);
Ns=[Ns zeros(na+nm+nw,nm+nz);zeros(nm+nz,cns) eye(nm+nz)];
%************************************************************
setlmis([]);

% Define the matrix variables: R,S,L3,J3
R=lmivar(1,[na 1]);
S=lmivar(1,[na 1]);
L3=lmivar(1,blk);
[J3,nvar]=lmivar(1,blk);

% Display variable info
disp(' ');
disp(['There are ' int2str(nvar) ' standard variables.'])
disp(' ');


% Define the individual terms in the first two equations
lmiterm([1,0,0,0],Nr);
lmiterm([2,0,0,0],Ns);
lmiterm([1,1,1,R],A,1,'s');
lmiterm([2,1,1,S],A',1,'s');
lmiterm([1,2,1,R],Ct,1);
lmiterm([2,2,1,S],Bt',1);
lmiterm([1,2,2,J3],-gamma,1);
lmiterm([2,2,2,L3],-gamma,1);
lmiterm([1,3,1,R],C1,1);
lmiterm([2,3,1,S],B1',1);
lmiterm([1,3,3,0],-gamma);
lmiterm([2,3,3,0],-gamma);
lmiterm([1,4,1,J3],1,Bt');
lmiterm([2,4,1,L3],1,Ct);
lmiterm([1,4,2,J3],1,Dtt');
lmiterm([2,4,2,L3],1,Dtt);
lmiterm([1,4,3,J3],1,D1t');
lmiterm([2,4,3,L3],1,Dt1);
lmiterm([1,4,4,J3],-gamma,1);
lmiterm([2,4,4,L3],-gamma,1);
lmiterm([1,5,1,0],B1');
lmiterm([2,5,1,0],C1);
```

```
lmiterm([1,5,2,0],Dt1');
lmiterm([2,5,2,0],D1t);
lmiterm([1,5,3,0],D11');
lmiterm([2,5,3,0],D11);
lmiterm([1,5,5,0],-gamma);
lmiterm([2,5,5,0],-gamma);

% Define individual terms in the last two equations
lmiterm([-3,1,1,R],1,1);
lmiterm([-4,1,1,L3],1,1);
lmiterm([-3,2,1,0],1);
lmiterm([-4,2,1,0],1);
lmiterm([-3,2,2,S],1,1);
lmiterm([-4,2,2,J3],1,1);

lsys=getlmis;

% Solve Feasibility problems
[t,xfeas]=feasp(lsys,opt,goal);
R=dec2mat(lsys,xfeas,R);
S=dec2mat(lsys,xfeas,S);
L3=dec2mat(lsys,xfeas,L3);
J3=dec2mat(lsys,xfeas,J3);

% Evaluate Feasibility
evals=evallmi(lsys,xfeas);
[lhs1]=showlmi(evals,1);
[lhs2]=showlmi(evals,2);
[lhs3,rhs3]=showlmi(evals,3);
[lhs4,rhs4]=showlmi(evals,4);

% Display Results
disp(' ')
disp('The following values should be negative.')
disp(['The maximum eig of the LHS of LMI1 is: ' num2str(max(eig(lhs1)))]);
disp(['The maximum eig of the LHS of LMI2 is: ' num2str(max(eig(lhs2)))]);
disp(' ')
disp('The following values should be positive.')
disp(['The minimum eig of the RHS of LMI3 is: ' num2str(max(eig(rhs3)))]);
disp(['The minimum eig of the RHS of LMI4 is: ' num2str(max(eig(rhs4)))]);
disp(['The minimum eig of R is: ' num2str(min(eig(R)))]);
disp(['The minimum eig of S is: ' num2str(min(eig(S)))]);
```

```
disp(['The minimum eig of L3 is: ' num2str(min(eig(L3)))]);
disp(['The minimum eig of J3 is: ' num2str(min(eig(J3)))]);

% Compute L and J (old way)
%N=(J3\eye(nm))-L3;
%L=[-N' N zeros(nm,nz);N' L3 zeros(nm,nz); zeros(nz,2*nm) eye(nz)];
%J1=(-N')\eye(nm)+J3;
%J=[ J1 J3 zeros(nm,nz); J3 J3 zeros(nm,nz);zeros(nz,2*nm) eye(nz)];

% Compute L and J (new way)
J3p=chol(J3);
J3pi=J3p\eye(nm);
J3i=J3pi*J3pi';
N=(J3i-L3);
Np=chol(-N);
Npi=Np\eye(nm);
Ni=Npi*Npi';
J1=J3+Ni;
L=[-N N zeros(nm,nz);N L3 zeros(nm,nz); zeros(nz,2*nm) eye(nz)];
J=[ J1 J3 zeros(nm,nz); J3 J3 zeros(nm,nz);zeros(nz,2*nm) eye(nz)];


% Form the elements of the true weighted plant
a=A;b1=[zeros(na,nm) Bt B1];b2=[B2 zeros(na,nm)];
c1=[zeros(nm,na);Ct;C1];c2=[C2;zeros(nm,na)];
d11=[zeros(nm,2*nm+nw);zeros(nm) Dtt Dt1;zeros(nz,nm) D1t D11];
d12=[zeros(nm,nu) eye(nm);Dt2 zeros(nm);D12 zeros(nz,nm)];
d21=[zeros(ny,nm) D2t D21; eye(nm) zeros(nm,nm+nw)];
d22=[D22 zeros(ny,nm);zeros(nm,nu+nm)];

% Scale the true plant so that L=J=I
condL=cond(L)
condJ=cond(J)
%F=chol(L);Fi=F\eye(2*nm+nz);
F=sqrtm(L);Fi=F\eye(2*nm+nz);
b1=b1*Fi;c1=F*c1;d11=F*d11*Fi;d21=d21*Fi;d12=F*d12;

% Form the outputs needed for Gahinet's klmi.m
Ps=pck(a,[b1 b2],[c1;c2],[d11 d12;d21 d22]);

end
```

```
% end nshinflmi.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.10 nshinflmi2.m*: Solve LMI GEVP Problem*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% nshinflmi2.m

function [Ps,R,S,L,J,gamma]=nshinflmi2(P,blk,nw,opt,target)
%
% NSHINFLMI2 Find the solution to the Apkarian's scaled Hinf optimization
%            problem.  The problem is formulated as a Generalized Eigenvalue
%            Problem using LMIs.  Gamma is displayed during the
%            minimization.
%
% usage:   [R,S,L,J]=nshinflmi2(P,blk,nw,opt,target)
%
% inputs:  P      - Open-loop plant in packed notation. i.e. P=pck(A, ...
%                   [Bt B1 B2], [Ct;C1;C2], [Dtt ...;D1t ...; D2t ...]),
%                   where t denotes the parameter influence
%          blk    - nm x 2 matrix describing the structure of the parameter
%                   block. The first element in each row denotes the size of
%                   the subblock.  The second element in each row is either
%                   0 (for repeated scalar) or 1 (for full).
%          nw     - Number of exogeneous inputs (must equal the number of
%                   controlled outputs)
%          opt    - Optional vector of parameter to pass to the LMI
%                   generalized eigenvalue solver. opt=[ (accuracy) (iters)
%                   (feas. rad.) (slow prog. tol)  (print)], default=[1e-2
%                   200 1e7 30 0].  See the LMI Lab manual for more details.
%          target- Optional target for the generalized eigenvalue
%                   minimization problem (gevp), i.e. the desired gamma.
%
% outputs: Ps     - The COMPLETE weighted plant scaled by L and J
%          R, S   - Solutions to the gevp problem
%          L, J   - The complete scaling matrices derived from L3 and J3
%          gamma - The optimal gamma found
%
```

```
% Written by: Capt Mark Spillman
%             WL/FIGC-3 Bldg 146, WPAFB
%
% Modified by: Capt Martin Breton, AFIT

if nargin<1
   disp('   usage:   [R,S,L,J,gopt]=nshinflmi2(P,blk,nw,opt,target);');
   disp(' ');
   return
end

if nargin<5;target=0;end
if nargin<4;opt=[1e-2 200 1e7 30 0];end

% The equations described below are derived from "A Convex Characterization
% of Gain-Scheduled Hinf Controllers"

% Get nm and change blk to denote the structure of L3 and J3
nm=sum(blk(:,1));
blk(:,2)=-(blk(:,2)-1);

% Set nz=nw, a current restriction of the code
nz=nw;

% Unpack P and spilt apart the elements in B, C and D
[A,B,C,D]=unpck(P);
na=size(A,1);nu=size(B,2)-nw-nm;ny=size(C,1)-nz-nm;
Bt=B(:,1:nm);B1=B(:,nm+1:nm+nw);B2=B(:,nm+nw+1:nm+nw+nu);
Ct=C(1:nm,:);C1=C(nm+1:nm+nz,:);C2=C(nm+nz+1:nm+nz+ny,:);
Dtt=D(1:nm,1:nm);Dt1=D(1:nm,nm+1:nm+nw);Dt2=D(1:nm,nm+nw+1:nm+nw+nu);
D1t=D(nm+1:nm+nz,1:nm);D11=D(nm+1:nm+nz,nm+1:nm+nw);
D12=D(nm+1:nm+nz,nm+nw+1:nm+nw+nu);
D2t=D(nm+nz+1:nm+nz+ny,1:nm);D21=D(nm+nz+1:nm+nz+ny,nm+1:nm+nw);
D22=D(nm+nz+1:nm+nz+ny,nm+nw+1:nm+nw+nu);

% tolerances (taken from Gahinet's routines)
macheps=mach_eps;
tolsing=sqrt(macheps);
toleig=macheps^(2/3);

% Make the appropriate substitutions to use a part of Gahinet's
% routine below
```

```
DD12=[Dt2;D12];DD21=[D2t D21];

% The following was taken from Gahinet's goptlmi.m:
%*************************************************************
% For numerical stability of the controller computation,
% zero the sing. values of DD12 s.t  || B2 DD12^+ || > 1/tolsing

[u,s,v]=svd(DD12);
abstol=max(toleig*norm(B2,1),tolsing*s(1,1));
ratio=max([s;zeros(1,size(s,2))])./...
      max([tolsing*abs(B2*v);abstol*ones(1,nu)]);
ind2=find(ratio < 1); l2=length(ind2);
if l2 > 0, s(:,ind2)=zeros(nz,length(ind2)); DD12=u*s*v'; end

[u,s,v]=svd(DD21');
abstol=max(toleig*norm(C2,1),tolsing*s(1,1));
ratio=max([s;zeros(1,size(s,2))])./...
      max([tolsing*abs(C2'*v);abstol*ones(1,ny)]);
ind2=find(ratio < 1); l2=length(ind2);
if l2 > 0, s(:,ind2)=zeros(nm+nw,length(ind2)); DD21=v*s'*u'; end
%*************************************************************

% Compute the outer factors for the standard feasibility problem
Nr=rnull([B2;DD12]',0,tolsing);
Wr1=Nr(1:na,:);Wr2=Nr(na+1:na+nm+nz,:);
Ns=rnull([C2,DD21],0,tolsing);
Ws1=Ns(1:na,:);Ws2=Ns(na+1:na+nm+nz,:);

% Cut-down Wr2 for the GEVP
[Wr2u,Wr2s,Wr2v]=svd(Wr2);
[rwr2,cwr2]=size(Wr2);Wr2sp=zeros(rwr2,cwr2);
cutJ=sum(diag(Wr2s)>max(size(Wr2))*max(diag(Wr2s))*eps);
Wr2sp(rwr2-cutJ+1:rwr2,cwr2-cutJ+1:cwr2)=Wr2s(1:cutJ,1:cutJ);
Wr2up=[Wr2u(:,cutJ+1:rwr2) Wr2u(:,1:cutJ)];
Wr2vp=[Wr2v(:,cutJ+1:cwr2) Wr2v(:,1:cutJ)];
Wr2p=Wr2up*Wr2sp;
Wr2pp=Wr2u(:,1:cutJ)*Wr2s(1:cutJ,1:cutJ);

% Cut-down Ws2 for the GEVP
[Ws2u,Ws2s,Ws2v]=svd(Ws2);
[rws2,cws2]=size(Ws2);Ws2sp=zeros(rws2,cws2);
cutL=sum(diag(Ws2s)>max(size(Ws2))*max(diag(Ws2s))*eps);
```

```
Ws2sp(rws2-cutL+1:rws2,cws2-cutL+1:cws2)=Ws2s(1:cutL,1:cutL);
Ws2up=[Ws2u(:,cutL+1:rws2) Ws2u(:,1:cutL)];
Ws2vp=[Ws2v(:,cutL+1:cws2) Ws2v(:,1:cutL)];
Ws2p=Ws2up*Ws2sp;
Ws2pp=Ws2u(:,1:cutL)*Ws2s(1:cutL,1:cutL);

% Compute the New outer factors for the GEVP
cnr=size(Nr,2);
Nrp=[Wr2vp zeros(cnr,nm+nz);zeros(nm+nz,cnr) eye(nm+nz)];
cns=size(Ns,2);
Nsp=[Ws2vp zeros(cns,nm+nz);zeros(nm+nz,cns) eye(nm+nz)];

% Define some additional shortcut matrices
B1h=[Bt B1];C1h=[Ct;C1];D11h=[Dtt Dt1;D1t D11];
I1=[eye(nm) zeros(nm,nz)];
I2=[zeros(nm,nm+nz);zeros(nz,nm) eye(nz)];
IY1=[zeros(cutJ,cwr2-cutJ) eye(cutJ)];
IZ1=[zeros(cutL,cws2-cutL) eye(cutL)];

% Define the matrix variables: R, S, L3, J3
setlmis([]);

R=lmivar(1,[na 1]);
S=lmivar(1,[na 1]);
L3=lmivar(1,blk);
[J3,nstandardvar]=lmivar(1,blk);

% Define the variable Y
[Y1,ndec]=lmivar(1,[cutJ 1]);
dy2=ndec+1:(nm+nz)*cutJ+ndec;
dy2=(reshape(dy2,cutJ,(nm+nz)))';
Y2=lmivar(3,[zeros(nm+nz,cwr2-cutJ) dy2]);
Y2p=lmivar(3,dy2);
Y3=lmivar(1,[(nm+nz) 1]);

% Define the variable Z
[Z1,ndec]=lmivar(1,[cutL 1]);
dz2=ndec+1:(nm+nz)*cutL+ndec;
dz2=(reshape(dz2,cutL,(nm+nz)))';
Z2=lmivar(3,[zeros(nm+nz,cws2-cutL) dz2]);
Z2p=lmivar(3,dz2);
[Z3,ntotalvar]=lmivar(1,[(nm+nz) 1]);
```

```
% Display the variable information
ndummyvar=ntotalvar-nstandardvar;
disp(' ');
disp('Attempting to solve the GEVP.')
disp(['There are ' int2str(nstandardvar) ' standard variables.'])
disp(['There are ' int2str(ndummyvar) ' additional dummy variables.'])
disp(' ');

% Define individual terms in the first equation
lmiterm([-1,1,1,R],1,1);
lmiterm([-1,2,1,0],1);
lmiterm([-1,2,2,S],1,1);

% Define the individual terms in the second equation
lmiterm([-2,1,1,L3],1,1);
lmiterm([-2,2,1,0],1);
lmiterm([-2,2,2,J3],1,1);

% Define the individual terms in the third equation
lmiterm([3,0,0,0],Nrp);
lmiterm([3,1,1,R],Wr1'*A,Wr1,'s');
lmiterm([3,1,1,R],Wr2'*C1h,Wr1,'s');
lmiterm([3,2,1,J3],I1',I1*(B1h'*Wr1+D11h'*Wr2));
lmiterm([3,2,1,0],I2*(B1h'*Wr1+D11h'*Wr2));
lmiterm([-3,1,1,Y1],IY1',IY1);
lmiterm([-3,2,1,Y2],1,1);
lmiterm([-3,2,2,Y3],1,1);

% Define the individual terms in the fourth equation
lmiterm([4,0,0,0],Nsp);
lmiterm([4,1,1,S],Ws1'*A',Ws1,'s');
lmiterm([4,1,1,S],Ws2'*B1h',Ws1,'s');
lmiterm([4,2,1,L3],I1',I1*(C1h*Ws1+D11h*Ws2));
lmiterm([4,2,1,0],I2*(C1h*Ws1+D11h*Ws2));
lmiterm([-4,1,1,Z1],IZ1',IZ1);
lmiterm([-4,2,1,Z2],1,1);
lmiterm([-4,2,2,Z3],1,1);

% Define the individual terms in the fifth equation
lmiterm([5,1,1,Y1],1,1);
lmiterm([5,2,1,Y2p],1,1);
```

```
lmiterm([5,2,2,Y3],1,1);
lmiterm([-5,1,1,J3],Wr2pp'*I1',I1*Wr2pp);
lmiterm([-5,1,1,0],Wr2pp'*I2*Wr2pp);
lmiterm([-5,2,2,J3],I1',I1);
lmiterm([-5,2,2,0],I2);

% Define the individual terms in the sixth equation
lmiterm([6,1,1,Z1],1,1);
lmiterm([6,2,1,Z2p],1,1);
lmiterm([6,2,2,Z3],1,1);
lmiterm([-6,1,1,L3],Ws2pp'*I1',I1*Ws2pp);
lmiterm([-6,1,1,0],Ws2pp'*I2*Ws2pp);
lmiterm([-6,2,2,L3],I1',I1);
lmiterm([-6,2,2,0],I2);

lgevp=getlmis;

% Solve the Generalized Eigenvalue Minimization Problem
[gamma,xopt]=gevp(lgevp,2,opt,[],[],target);
R=dec2mat(lgevp,xopt,R);
S=dec2mat(lgevp,xopt,S);
L3=dec2mat(lgevp,xopt,L3);
J3=dec2mat(lgevp,xopt,J3);

% Compute L and J (old way)
%N=(J3\eye(nm))-L3;
%L=[-N' N zeros(nm,nz);N' L3 zeros(nm,nz); zeros(nz,2*nm) eye(nz)];
%J1=(-N')\eye(nm)+J3;
%J=[ J1 J3 zeros(nm,nz); J3 J3 zeros(nm,nz);zeros(nz,2*nm) eye(nz)];

% Compute L and J (new way)
J3p=chol(J3);
J3pi=J3p\eye(nm);
J3i=J3pi*J3pi';
N=(J3i-L3);
Np=chol(-N);
Npi=Np\eye(nm);
Ni=Npi*Npi';
J1=J3+Ni;
L=[-N N zeros(nm,nz);N L3 zeros(nm,nz); zeros(nz,2*nm) eye(nz)];
J=[ J1 J3 zeros(nm,nz); J3 J3 zeros(nm,nz);zeros(nz,2*nm) eye(nz)];
```

```
% Form the elements of the true weighted plant
a=A;b1=[zeros(na,nm) Bt B1];b2=[B2 zeros(na,nm)];
c1=[zeros(nm,na);Ct;C1];c2=[C2;zeros(nm,na)];
d11=[zeros(nm,2*nm+nw);zeros(nm) Dtt Dt1;zeros(nz,nm) D1t D11];
d12=[zeros(nm,nu) eye(nm);Dt2 zeros(nm);D12 zeros(nz,nm)];
d21=[zeros(ny,nm) D2t D21; eye(nm) zeros(nm,nm+nw)];
d22=[D22 zeros(ny,nm);zeros(nm,nu+nm)];

% Scale the true plant so that L=J=I
condL=cond(L)
condJ=cond(J)
F=sqrtm(L);Fi=F\eye(2*nm+nz);
%F=chol(L);Fi=F\eye(2*nm+nz);
b1=b1*Fi;c1=F*c1;d11=F*d11*Fi;d21=d21*Fi;d12=F*d12;

% Form the outputs needed for Gahinet's klmi.m
Ps=pck(a,[b1 b2],[c1;c2],[d11 d12;d21 d22]);

% end nshinflmi2.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix D. Simulation Programs

This appendix lists the main programs and *S-functions* used to setup and run the simulations.

D.1 LPVG.m: Generate LPV Simulation Aircraft Model in SIMULINK

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LPVG.m

function [sys,x0]=LPVG(t,x,u,flag,G,Mbar1,Mtil1,hbar1,htil1,nM1,nh1,Mo,ho,Uo)

% M-file Simulink S-function to incorporate
% theta into G before converting to state-space

% nominal G must be in workspace

if flag == 0
  M=Mo;
  h=ho;
else
  M=u(1);
  h=u(2); % new ho; could find new Uo
  uu=[u(3);u(4)];
end

dM=(M-Mbar1)/Mtil1;
dh=(h-hbar1)/htil1;
theta=[dM*eye(nM1) zeros(nM1,nh1);zeros(nh1,nM1) dh*eye(nh1)];

GLTI=starp(theta,G);
[Ag,Bg,Cg,Dg]=unpck(GLTI);

% add wind noise input for simulink;
Bgx=[Ag(:,2) Bg];
Dgx=[zeros(4,1) Dg];

if abs(flag) == 1 % return state derivatives
  sys=Ag*x+Bgx*uu;
```

```
elseif flag == 3 % return system outputs y
  sys=Cg*x+Dgx*uu;   % y must be at least: u,aoa,theta
  x(1)=0;

elseif flag == 0   % parameter sizes and initial conditions
  [m,n]=size(Dgx);
  sys=[length(Ag) 0 m n+2 0 max(any(Dgx~=0))];
  x0=[0 0 0 0]';

else % flag=2 or 4 used for discrete only
  sys=[];
end

% end of LPVG.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D.2 LPVK.m*: Generate LPV Controller in SIMULINK*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LPVK.m

function [sys,x0]=LPVK(t,x,u,flag,K,Mbar,Mtil,hbar,htil,nM,nh,Mo,ho)

% M-file Simulink S-function to incorporate
% theta into K before converting to state-space
% use in dynamic simulations

% nominal K must be in workspace

if flag == 0
  M=Mo;
  h=ho;
else
  M=u(1);
  h=u(2);
  uu=[u(3);u(4);u(5)];
end
```

```
dM=(M-Mbar)/Mtil;
dh=(h-hbar)/htil;
theta=[dM*eye(nM) zeros(nM,nh);zeros(nh,nM) dh*eye(nh)];

KLTI=starp(K,theta);
[Ak,Bk,Ck,Dk]=unpck(KLTI);

if abs(flag) == 1 % return state derivatives
  sys=Ak*x+Bk*uu;

elseif flag == 3 % return system outputs y
  sys=Ck*x+Dk*uu;   % y must be at least: u,aoa,theta

elseif flag == 0   % parameter sizes and initial conditions
  [m,n]=size(Dk);
  sys=[length(Ak) 0 m n+2 0 max(any(Dk~=0))];
  x0=zeros(length(Ak),1);

else % flag=2 or 4 used for discrete only
  sys=[];
end

% end of LPVK.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D.3 f18simdat.m*: Setup Static Simulations*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18simdat.m

% f18 static simulation setup program

% this program converts the LPV plant
% and LPV controller into an LTI plant
% and LTI controller at a given M,h pt
% in the flight envelope in order to
% perform static tests
```

```
% load LPV K  % need controller in workspace or file
load fenv3RP

% enter given values for M,h test point
M=.75
h=15000
ao=0
% end of user inputs

K=K_3;
nM=tblk(1);
nh=tblk(2);

Act=[ -20.2000   20.2000    1.0000
    1.0000         0         0
   20.2000   20.2000         0
         0         0      -Inf];

[Ak,Bk,Ck,Dk]=unpck(K);
[Ade,Bde,Cde,Dde]=unpck(Act);

% LTI Gdel and Kdel (which take theta into account)
dM=(M-Mbar)/Mtil;
dh=(h-hbar)/htil;
theta=[dM*eye(nM) zeros(nM,nh);zeros(nh,nM) dh*eye(nh)];

Kdel=starp(K,theta); % gives K at test point
[Ak,Bk,Ck,Dk]=unpck(Kdel);

% dele actuator output
Cde=Cde(1,1);
Dde=Dde(1,1);


% get full longitudinal simulation model and find LTI G at test pt

load LPVF18; % get full longitudinal plant G
nM1=9;
nh1=10;
% LTI Gdel  (which take theta into account)
dM1=(M-Mbar1)/Mtil1;
```

```
dh1=(h-hbar1)/htil1;
thet1=[dM1*eye(nM1) zeros(nM1,nh1);zeros(nh1,nM1) dh1*eye(nh1)];

Gdel=starp(thet1,G);
[Ag,Bg,Cg,Dg]=unpck(Gdel);

s=size(Ag,1)/2; % s column of Ag is aoa
% add wind noise input for simulink;
Bgx=[Ag(:,s) Bg];
Dgx=[zeros(size(Dg,1),1) Dg];

% end of f18simdat.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D.4 `f18LPVsimdat.m`: *Setup Dynamic Simulation*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f18LPVsimdat.m

% f18 dynamic simulation setup program

% this program sets up LPV plant simulation model
% in order to perform dynamic tests
% controller must be available in workspace or file

load env1e5
K=K_3; % LPV controller
%load Kmusp
%load Kmuhih
%[Ak,Bk,Ck,Dk]=unpck(k_dk3gb); % to test LTI/mu controller
% need to also replace LPVK with state-space
nM=tblk(1);
nh=tblk(2);
hbar
htil
Mbar
Mtil
```

```
% user inputs -- starting pt of flight trajectory
ho=30000
Mo=.5
ao=0
% end user inputs

To=518.67-.003565*ho % good approx up to 36000 ft
Uo=Mo*sqrt(1.4*1716.16*To)

load LPVF18; % get full longitudinal simulation plant
nM1=9;
nh1=10;

Act=[ -20.2000   20.2000    1.0000
    1.0000         0         0
   20.2000   20.2000         0
         0         0      -Inf];

[Ade,Bde,Cde,Dde]=unpck(Act);
% dele actuator output
Cde=Cde(1,1);
Dde=Dde(1,1);

% end of f18LPVsimdat.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

ABSB92. R. J. Adams, J. M. Buffington, A. G. Sparks, and S. S. Banda. *An Introduction to Multivariable Flight Control System Design.* WL-TR-92-3110. USAF Wright Laboratory, Flight Dynamics Directorate, 1992.

AG95. P. Apkarian and P. Gahinet. A Convex Characterization of Gain-Scheduled H_∞ Controllers. In *IEEE Transactions on Automatic Control*, volume 40, no. 6, pages 853–864, 1995.

All95. J. B. Allison. Application of Mixed-Norm Optimal Control to a Multi-Objective Active Suspension Problem. Master's thesis, Air Force Institute of Technology, December 1995.

AW89. K. J. Astrom and B. Wittenmark. *Adaptive Control.* Addison-Wesley, 1989.

BAG96. J. Biannic, P. Apkarian, and W. L. Garrard. Parameter Varying Control of a High Performance Aircraft. Submitted to the *1996 GNC*, 1996.

BDG+91. G. J. Balas, J. C. Doyle, K. Glover, A. Packard, and R. Smith. *μ-Analysis and Synthesis Toolbox.* The MathWorks, Inc., 1991.

BEFB94. S. Boyd, L. ElGhaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory (SIAM Studies in Applied Mathematics, vol. 15).* SIAM Publications, 1994.

Bla91. J. Blakelock. *Automatic Control of Aircraft and Missiles.* John Wiley and Sons Inc., 1991.

BP94. G. Becker and A. Packard. Robust Performance of Linear Parametrically Varying Systems using Parametrically-Dependent Linear Feedback. *Systems and Control Letters*, 23:205–215, 1994.

BS89. J. J. Bertin and M. L. Smith. *Aerodynamics for Engineers.* Prentice Hall, 1989.

CW94. X. Chen and J. T. Wen. A Linear Matrix Inequality Approach to the General Mixed H_2/H_∞ Control Problem. In *IEEE Transactions on Automatic Control*, 1994.

DGKF89. J. C. Doyle, K. Glover, P. Khargonekar, and B. Francis. State Space Solutions to Standard H_2 and H_∞ Control Problems. In *IEEE Transactions on Automatic Control*, volume 34, pages 831–847, 1989.

Doy85. J. C. Doyle. Structured Uncertainty in Control System Design. In *IEEE Conference on Decision and Control*, pages 260–265, 1985.

GA94. P. Gahinet and P. Apkarian. A Linear Matrix Inequality Approach to H_∞ Control. *Journal of Robust and Nonlinear Control*, 4:421–448, 1994.

Gah94. P. Gahinet. Explicit Controller Formulas for LMI-based H_∞ Synthesis. In *American Control Conference*, pages 2396–2400, 1994.

GNLC92. P. Gahinet, A. Nemirovskii, A. J. Laub, and M. Chilali. *LMI Control Toolbox.* The MathWorks, Inc., 1992.

LR93. D. A. Lawrence and W. J. Rugh. Gain Scheduling Dynamic Linear Controllers for a Nonlinear Plant. In *IEEE Conference on Decision and Control*, pages 1024–1029, 1993.

MAT. MATLAB: High Performance Numeric Computation and Visualization Software. The MathWorks, Inc., Natick MA, 1994.

NN94. A. S. Nemirovskii and Y. E. Nesterov. *The Projective Method for Solving Linear Matrix Inequalities.* SIAM Publications, 1994.

Pac94. A. Packard. Gain Scheduling via Linear Fractional Transformations. *Systems and Control Letters*, 22:79–92, 1994.

PD93. A. Packard and J. Doyle. The Complex Structured Singular Value. *Automatica*, 29(1):71–109, 1993.

PZPB91. A. Packard, K. Zhou, P. Pandey, and G. Becker. A Collection of Robust Control Problems Leading to LMIs. In *IEEE Conference on Decision and Control*, volume 2, pages 1245–1250, 1991.

SA91a. J. F. Shamma and M. Athans. Gain Scheduling: Potential Hazards and Possible Remedies. In *American Control Conference*, pages 516–521, 1991.

SA91b. J. F. Shamma and M. Athans. Guaranteed Properties of Gain Scheduled Control for Linear Parameter-Varying Plants. *Automatica*, 27:559–564, 1991.

SC92. J. F. Shamma and J. R. Cloutier. A Linear Parameter-Varying Approach to Gain Scheduled Missile Autopilot Design. In *American Control Conference*, pages 1317–1321, 1992.

Sch95. C. W. Scherer. Mixed H_2/H_∞ Control for Linear Parametrically Varying Systems. In *IEEE Conference on Decision and Control*, pages 3182–3187, 1995.

SIM. SIMULINK: Dynamic System Simulation Software. The MathWorks, Inc., Natick MA, 1992.

SLBB96. M. Spillman, L. Lee, P. Blue, and S. Banda. A Robust Gain-Scheduling Example Using Linear Parameter-Varying Feedback. Submitted to the *1996 IFAC*, 1996.

WYPB94. F. Wu, X. H. Yang, A. Packard, and G. Becker. Induced L_2-Norm Control for LPV System with Bounded Parameter Variation Rates. Submitted to the *Int. Journal of Nonlinear and Robust Control*, 1994.

ZDG96. K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall, Inc., 1996.

ZPD82. K. Zhou, A. Packard, and J. C. Doyle. Review of LFTs, LMIs and μ. In *IEEE Conference on Decision and Control*, volume 2, pages 1227–1232, 1982.

Vita

Captain Martin R. Breton was born January 26, 1965, in Chicoutimi, Quebec, Canada. As one of the top students in his province, he was awarded a university scholarship in 1983 and went on to obtain his Bachelor of Engineering in Electrical Engineering from McMaster University in May 1987. Having joined the Canadian Forces in 1985, he completed qualification training for the Aeronautical Engineer officer classification in 1988. During his first assignment in Ottawa, Captain Breton was responsible for the maintenance and upgrade of the CF-18 mission computers, cockpit displays, data recorders, and associated systems, as well as a project officer on the CF-5 Avionics Upgrade program. In 1991, he was transferred to CFB Cold Lake, Alberta, where he maintained and improved software for the entire CF-18 automatic test equipment suite. Finally, in May 1994, Captain Breton began his Masters Degree Program at the Air Force Institute of Technology at Wright-Patterson AFB in Dayton, Ohio.

Permanent address: 85 Place St. Mathieu
Beloeil, Quebec, Canada J3G4S5
email: jbreton@afit.af.mil

June 1996 Master's Thesis

Gain-Scheduled Aircraft Control Using Linear Parameter-Varying Feedback

Martin R. Breton, Capt, Canada

Air Force Institute of Technology, WPAFB OH 45433-7655

Dr Marc Jacobs
AFOSR/NM
110 Duncan Ave, Suite B115
Bolling AFB DC 20332-0001

Systems which vary significantly over an operating envelope, such as fighter aircraft, generally cannot be controlled by a single linear time-invariant controller. As a result, gain-scheduling methods are employed to design control laws which can provide the desired performance. This thesis examines a relatively new approach to gain-scheduling, in which the varying controller is designed from the outset to guarantee robust performance, thereby avoiding the disadvantages of point designs. Specifically, the parameter-varying (LPV) aircraft model is linearized using linear fractional transformations (LFT's), and the resulting control problem is characterized as the solution to a set of four linear matrix inequalities (LMI's). The supporting theory is reviewed and two pitch-rate controllers are designed; one for the full longitudinal aircraft model, and another for the short period model. It is found that, even though the varying controllers are quite conservative, they can guarantee better robust performance over a large portion of an operating envelope when compared to time-invariant μ-synthesis controllers.

Control Theory, Gain Scheduling, Linear Parameter-Varying Systems
H_∞ Optimization, Linear Matrix Inequalities, Structured Singular Value
Aircraft Control

218

UNCLASSIFIED UNCLASSIFIED UNCLASSIFIED UL