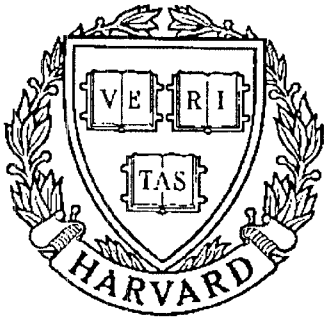
THESIS REPORT
Master's Degree

SYSTEMS
RESEARCH
CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Design of Flight Control Systems by Optimal Control Methods

by J. Reilly
Advisor: W.S. Levine

M.S. 90-4
Formerly TR 90-17r1

Design of Flight Control Systems by Optimal Control Methods

by

John Reilly

Thesis submitted to the Faculty of The Graduate School
of The University Of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1989

Advisory Committee:

Professor William S. Levine Chairman/Advisor
Associate Professor Eyad H. Abed
Associate Professor Andre L. Tits

ABSTRACT

Title of Thesis:	Design of Flight Control Systems by Optimal Control Methods
Name of degree candidate:	John Reilly
Degree and Year:	Master of Science, 1989
Thesis directed by:	William S. Levine, Professor, Department of Electrical Engineering

We outline two example applications of optimal control methodologies to aid in the design of control systems. The first method uses CONSOLE, a computer-aided design program based on feasible direction and multicriterion optimization, for controller parameter selection. The second method, based on linear quadratic regulator theory, selects a controller structure where the parameters are selected to minimize a performance function. Each of these methods is demonstrated using a realistic example from Grumman Aerospace Corporation. The CONSOLE method is used for a pitch control design for the X-29, a forward-swept wing aircraft, while the linear-quadratic-gaussian/loop transfer recovery method is used for a pitch control design for the F-14.

We show that the designs from these methods are at least comparable to ones done with other methods. In fact, the LQG/LTR design method leads to a stable robust design. Therefore, these design tools would be useful additions to a control designer's toolbox.

ACKNOWLEDGEMENTS

First, I wish to sincerely thank my advisor, Dr. William S. Levine for his guidance and patience during this project. I also wish to thank Drs. Eyad H. Abed and Andre L. Tits for their time and effort in serving in my advisory committee. For his help in understanding CONSOLE, I wish to thank Dr. M.K.H. Fan.

This work was supported by the Systems Research Center under grant NSFD CDR 8803012 from the Engineering Research Centers Program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern aircraft designs, in response to increasing demands, are relying more on the control system. Hence, the controller designs need to be more sophisticated to handle the additional requirements. Classical methods can no longer be used so easily to come up with a good design.

Control methodologies are being developed which can accommodate more sophisticated systems rather nicely. In this thesis, we will give example applications using two methods from optimal control theory. The first method uses a computer-aided-optimization package called CONSOLE, developed at the University of Maryland. Since a controller for a modern aircraft will tend to be larger than before, a tool of this type will be needed to adjust the parameters to move the design in a positive direction. The second method is based on linear quadratic regulator theory which has shown to promote a robust design.

These methods will be applied to realistic longitudinal control examples for fighter-type aircraft. The first method will be demonstrated on the X-29 forward-swept wing research aircraft. The objective here will be to control the pitch movement of the aircraft given a vertical acceleration command input. We will describe the design objectives to CONSOLE so that the parameters of the controller can be adjusted to give us a good design. Note that this method requires that the structure for the controller be known a priori. The second method will be demonstrated on the F-14 aircraft. Again, we will control the pitch movement of the aircraft, but this time the input is an angle-of-attack command. This method produces the controller structure which minimizes a given performance function. For both cases, we will compare our designs to others which used alternative design schemes.

This thesis is organized as follows: chapter 2 outlines the work necessary to describe a good design to CONSOLE; chapter 3 gives a detailed description of the steps needed to produce a linear quadratic Gaussian/loop transfer recovery (LQG/LTR) design; and finally, chapter 4 concludes with what we have learned from these two designs.

Chapter 2

Designing a Controller for the X-29 Using CONSOLE

This chapter explores the feasibility of using CONSOLE (a computer-aided design program based on feasible direction and multicriterion optimization) to aid in the design of aircraft control systems. The model used for discussion is the 1/2 scale model of the X-29 used for wind tunnel testing. In evaluating CONSOLE, we have attempted to show the following: 1) accuracy, 2) ease of use. For the first case, we mean to show that it can accurately and reliably reproduce a given design. Since CONSOLE was designed to be a broad-based optimization program, a simulator is needed to evaluate the system under study. In this case a continuous time linear system simulator, *MaryLin*, developed at the University of Maryland, was used. Therefore, the question of accuracy, in terms of system outputs, ultimately is passed down to the simulator. As for the question of being easy to use, CONSOLE requires a description of the optimization problem to be solved — a Problem Description File. The description has two basic types: constraints, functional constraints — restricting functions to be in a given range, and objectives, functional objectives — to minimize or maximize functions. Apart from the syntax of the problem description, e. g. the range for the constraint, the body of the function being optimized is written in C. This allows the designer to optimize any function of the signals returned from the simulator, in this case the system outputs. In this manner, the problem description is easy to read and understand; basically, a knowledge of C is all that is needed.

We begin this chapter with a description of the model for the X-29. The struc-

ture for the controller and the overall design by Grumman Aerospace Corporation are then discussed. Next, we state the design criteria and discuss the process of converting them into a form usable by CONSOLE [1]. Starting from Grumman's design, CONSOLE then attempts to optimize the controller gains such that all the design criteria are satisfied. We then compare the final design with that by Grumman, and make suggestions for further research.

2.1 Background

The X-29 is a forward-swept-wing (FSW) research aircraft, and as such, it has different control problems than conventional aircraft. Characteristic of the FSW configuration, a phenomenon known as Body Freedom Flutter (BFF) is encountered. BFF is a dynamic instability caused by coupling of aircraft pitch and wing bending motions. Because the aircraft has forward-swept-wings, a pitch up movement increases the force pushing up on the wings, hence the wings tend to bend upward. With increasing airspeed, another important design consideration is wing divergence—an upward and backward bend great enough to tear them off. For a more detailed discussion of these two design problems, see refs. [2,3,4]. In addition to these structural problems, the X-29 is statically unstable (the center of pressure is 25% forward of the center of gravity). Therefore, an active control system must be implemented to stabilize the aircraft.

The addition of wing-mounted stores reduces the speed at which BFF occurs. Research at Grumman [2] indicates that a good design approach is to use aeroelastic tailoring on the clean-wing configuration to stiffen the wing, and thereby increase the speed at which BFF occurs. Since the wing is made of layers of composite materials, a computer was used to analyze the amount and direction of material needed at each layer to make the wing stiff enough to avoid divergence. An active divergence/flutter suppression (ADFS) system can then be employed to enable stores to be carried without altering the flight envelope. This was the approach followed by Grumman to design the control system for the 1/2-scale model [3].

First, a canard-based stability augmentation system (SAS) was designed for longitudinal stability of the clean-wing configuration. The g-command control sys-

tem (a system with normal acceleration as the input) includes a pitch rate gyro and a N_z (vertical acceleration) accelerometer to sense motions, and a canard actuator driven by a proportional plus integral compensator (see Figure 2.1). The pitch rate feedback was used to increase short-period damping, while the PI compensator in the forward loop eliminates steady-state errors. Since an angle of attack (α) sensor would be too slow for this type of aircraft, N_z feedback was used to stabilize the short-period mode [5]. However, a major component of normal acceleration is proportional to angle of attack. With the SAS installed, the model exhibited Level 1 flying qualities [6] and had adequate flutter margin, gain margin, and phase margin as demonstrated in a wind tunnel test [4]. Level 1 flying qualities is a level of control over the aircraft by the pilot which is clearly adequate for the mission Flight Phase. In other words, the aircraft performs at its peak and its response is defined by what pilots like to fly. Our specifications are for category A Flight Phase which consist of those nonterminal (i.e. in-flight) missions that require rapid maneuvering, precision tracking, or precise flight-path control.

The next step was to develop an ADFS system to prevent BFF on the stores configuration. This system works by employing sensors at the wing tip and wing root to determine the relative wing bending. The measurement and its accompanying rate were fedback to an outboard wing flaperon to compensate for the wing bending, and hence aerodynamically stiffen the wing. This delays the coupling of the wing bending and short-period modes, thereby delaying the onset of BFF to higher speeds. The goal for the ADFS system was to achieve acceptable longitudinal flying qualities while providing the required flutter speed margin. Initial testing showed that wing bending feedback was not enough to ensure acceptable peak flap and canard deflections for a 1g step input [7]. Therefore cross-coupling between the flap and canard channels was required. In particular, the filtered N_z step command was also fed to the flap actuator in addition to pitch rate feedback. With the cross-coupling, the eight controller gains (4 SAS, 4 ADFS) were varied to obtain an acceptable design.

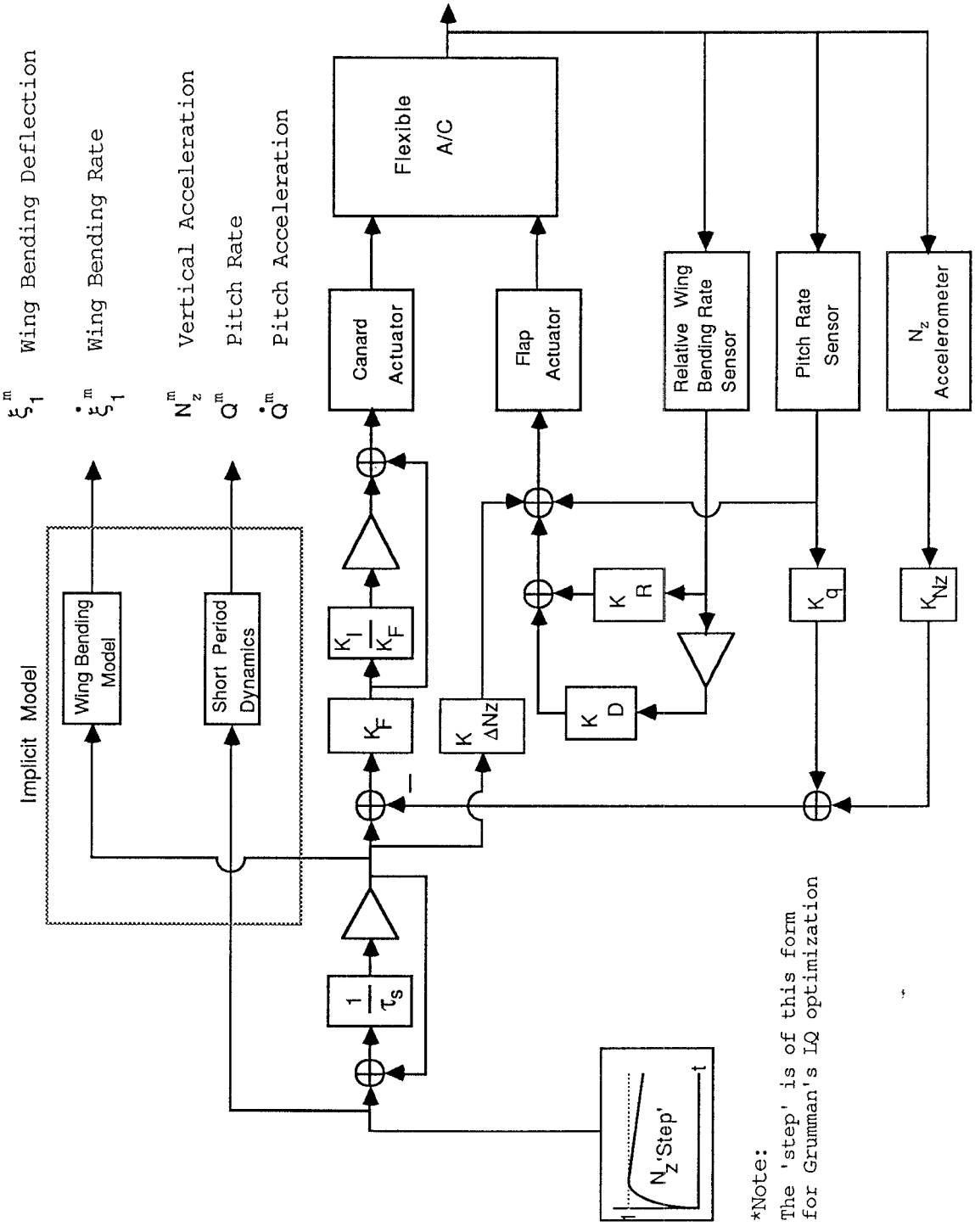


Figure 2.1: Block diagram of the longitudinal flight control system for Grumman's 1/2 scale model of the X-29 aircraft.

2.1.1 Design Goals

Based on a design point of sea-level and Mach 0.9, the following criteria must be satisfied:

- Level 1 flying qualities
- Control limits:

	Displacements		Rates	
	Deg	Rad	Deg/sec	Rad/sec
Canard	±8.0	±0.140	±60.0	±1.05
Flap	±10.0	±0.175	±100.0	±1.75

- Gain Margin : 6 dB
- Phase Margin: 45°
- Flutter Margin: 15° above V_d

2.2 Optimization of Grumman's Design

Our work with the model of the X-29 was to use CONSOLE to optimize the design by Grumman. The first step towards reaching a solution was to convert the design problem from a state variable feedback form, as given by Grumman's optimization program CASCADE, to the form required by *MaryLin*. This conversion required that the feedback matrices be incorporated into the system matrices {A,B,C,D}. The stated manipulations were handled quite readily by MACSYMA — a symbolic math program. Next, the design goals had to be described to CONSOLE.

2.2.1 Flying Qualities Assessment

The flying qualities criteria were evaluated using the following techniques: Control Anticipation Parameter (CAP) [6], Equivalent Systems Response [8], and Implicit Model Following. CAP is defined as the ratio of initial pitch acceleration to steady-state normal acceleration in response to a step longitudinal control input:

$$CAP = \frac{\dot{q}(t = 0^+)}{N_{z_{ss}}}$$

	Time delay		Damping ratio		$\frac{\omega_{sp}^2}{N_z/\alpha}$	
Level	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
1	0.0	0.1	0.35	1.3	0.28	3.6
2	0.1	0.2	0.25	2.0	0.16	10.0
3	0.2	0.25	0.15	$\cdots$	0.16	$\cdots$

Table 2.1: MIL-F-8785C short-period requirements

Since the system description includes actuators (hence $\dot{q}(t = 0^+) = 0$) , an investigation by Bischoff [8] showed that a better evaluation of the system can be achieved with an attenuated control anticipation parameter (CAP'). CAP' is defined as the ratio of the maximum pitch acceleration to steady-state normal acceleration:

$$CAP' = \frac{\dot{q}_{max}}{N_{z_{ss}}}$$

For level 1 flying qualities, pilot ratings of the aircraft response are favorable if

$$.25 < CAP' < 1.5$$

This design criterion was then constructed in the form of a constraint with the above upper and lower bounds.

The equivalent systems approach attempts to match the frequency response of the actual high-order system to that of a given low-order system. When restricted to the short-period mode, the desired low-order transfer function relating pitch rate to stick force is

$$\frac{q(s)}{\delta_{ST}(s)} = \frac{K_1(s + 1/\tau_{\theta_2})e^{-\tau_q s}}{s^2 + 2\xi_e\omega_e s + \omega_e^2} \tag{2.2.1}$$

The five parameters in the above equation are: pitch rate gain K_1, equivalent pitch rate time delay τ_q, numerator time constant τ_{θ_2}, equivalent short period damping ξ_e, and equivalent short-period frequency ω_e. Of the five, two parameters directly relate to the short-period characteristics of the aircraft, namely ξ_e and ω_e. For the category A flight phase [6], Table 2.1 indicates the requirements placed on the short-period characteristics. Of the remaining parameters in equation 2.2.1, K_1 is allowed to vary while τ_{θ_2} is set to the aircraft value.

In implementing the equivalent systems restrictions for CONSOLE, the requirements from Table 2.1 were used to calculate the upper and lower curves for the

magnitude of equation 2.2.1 as follows:

$$\left|\frac{q(\omega)}{\delta_{ST}(\omega)}\right|_{upper} = K\sqrt{\frac{1+(\omega\tau_{\theta_2})^2}{(1-\frac{\omega^2}{\omega^2_{sp_{max}}})^2+(\frac{2\xi_{sp_{min}}\omega}{\omega_{sp_{max}}})^2}} \tag{2.2.2}$$

where K is the DC value of the high order system

$$\omega_{sp_{max}} = 3.6 \times N_{z_{ss}}/\alpha_{ss}$$

$$\xi_{sp_{min}} = 0.35$$

The lower magnitude curve is given by (2.2.2) with $\omega_{sp_{max}}$ and $\xi_{sp_{min}}$ replaced by $\omega_{sp_{min}}$ and $\xi_{sp_{max}}$ respectively, with the obvious change of definitions. The design criterion was then described as two functional constraints, creating an envelope to restrict the magnitude of the actual pitch rate to stick force response (see Figure 2.2). The maximum allowable time delay was modeled as a lower bound on the phase response of equation 2.2.1

$$\phi(\omega)_{min} = \arctan(\omega\tau_{\theta_2}) - \arctan\left(\frac{\frac{2\xi_{sp_{max}}\omega}{\omega_{sp_{min}}}}{1-\frac{\omega^2}{\omega^2_{sp_{min}}}}\right) - \tau_{q_{max}}\omega \tag{2.2.3}$$

Equation 2.2.3 was implemented as the lower bound on the actual pitch rate to stick force phase response (see Figure 2.2).

The last technique used to evaluate flying qualities was Implicit Model Following. Low-order models of the wing-bending mode and short-period characteristics, based on the clean-wing design, were given by Grumman. These models were constructed from the Mil-specs as a reference to what the aircraft response should be to get favorable pilot ratings. The differences between the model and sensor signals were minimized via the objective function in CONSOLE. These differences were compared to an exponential function to force the steady-state errors to zero.

2.2.2 Design Goals Revisited

The remaining design goals were described to CONSOLE without involving any elaborate techniques. The control limits were constructed as simple functional constraints with the given displacement and rate restrictions. The gain and phase margin requirements were split into open loop requirements on the canard and flap loops separately. Each loop was opened in turn immediately before the actuator,

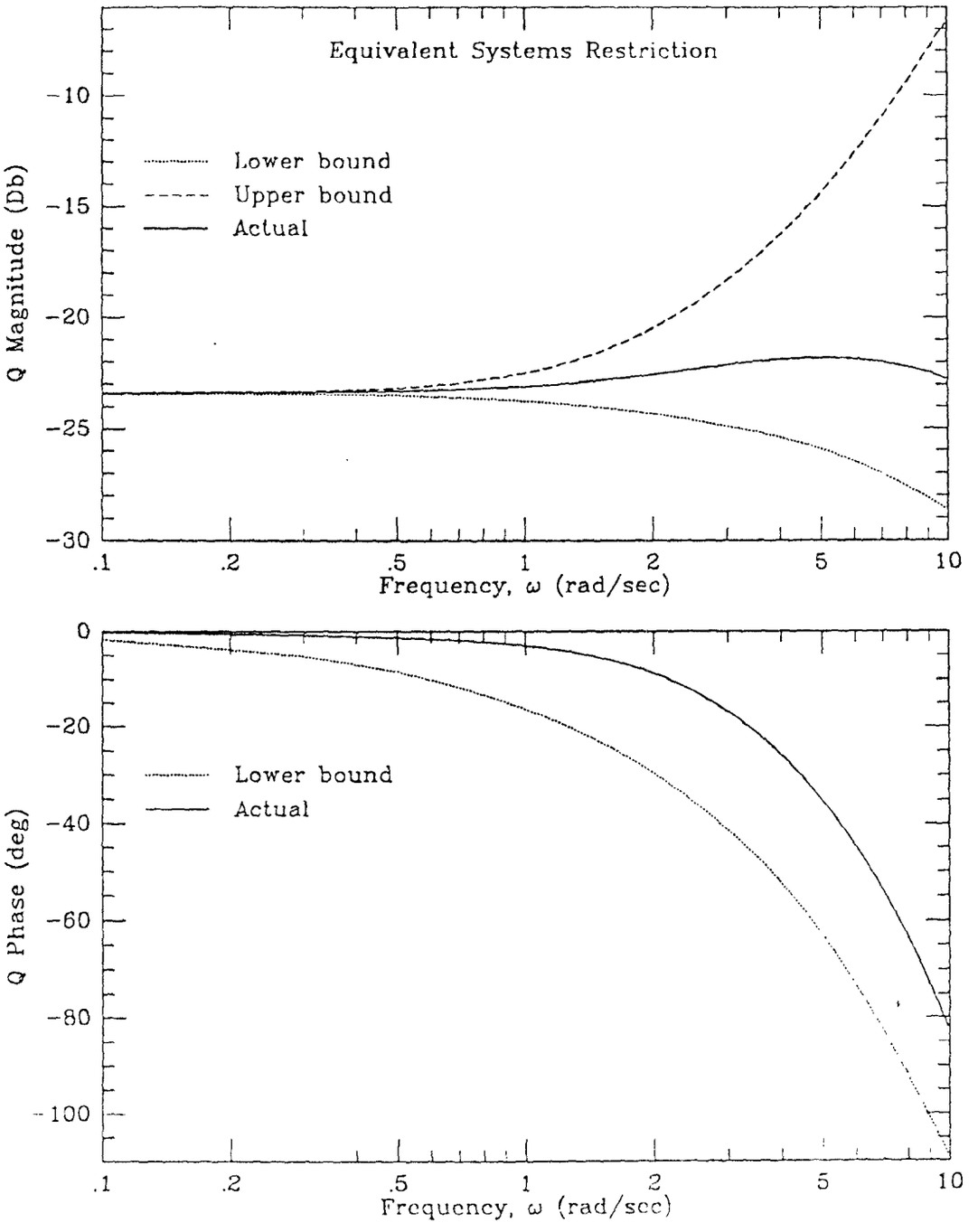


Figure 2.2: Restriction on the equivalent systems response

while the other loop remained intact. The gain margin was calculated by using a macro called "findroot" to locate the frequency where the open loop phase crosses the -180 degree mark. The phase margin was found in a similar fashion. The gain and phase margins were described to CONSOLE as constraints. The complete Problem Description File can be seen in appendix A. Table 2.2 lists the design criteria as stated in CONSOLE.

2.2.3 Running CONSOLE

With the Problem Description File completed, we were ready to run CONSOLE. The first step was to confirm the design by Grumman. This was necessary for two reasons: 1) to show the accuracy of the simulator, and 2) to determine, using CONSOLE, whether Grumman's design was indeed a good one. The 2nd reason was more for our sake, to confirm the description of the design goals in the PDF. We did indeed confirm Grumman's design.

Starting from the design by Grumman, CONSOLE was run for 50 iterations in an attempt to optimize the design even further. A comparison between the history plots after the initial simulation (i.e. Grumman's design), and after 50 iterations can be seen in appendix B. Upon comparison, there were no *significant* changes made to the system. There were, however, some improvements to be noted. The canard and flap displacements, along with the peak $\dot{q}$ error were decreased. As a result of the changes in the flap and canard deflections, there was a decrease in α_{ss}. In addition, the smaller control surface deflections and α_{ss} would lower the drag on the aircraft. Unfortunately these changes were at the expense of an increase in the peak value of the N_z error, although still below the design requirements.

2.3 Summary

We have shown that CONSOLE can be a useful design tool, eliminating the need to have one program for optimization, another to check the design, etc.. Furthermore, CONSOLE enables the designer to describe the design objectives in a manner analogous to the way he/she thinks, unlike LQ optimization techniques. Also, the designer can interactively change the design criteria to try alternative designs.

Design Criteria	Description	Good Value
CanDisp_I	Displacement limit on input to canard actuator	0.14 rad
CanRate_I	Rate limit	1.05 rad/sec
FlapDisp_I	Displacement limit on input to flaperon actuator	0.175 rad
FlapRate_I	Rate limit	1.75 rad/sec
CAPlowbd	Lower bound on CAP'	0.25
CAPupbd	Upper bound on CAP'	1.5
NzError	Minimize $(N_{z_{sensor}} - N_{z_{model}})$	$0.4e^{-t}$
QdotError	Minimize $(\dot{q}_{sensor} - \dot{q}_{model})$	$0.4e^{-t}$
GainMargCan	Lower bound on the gain margin for the canard channel	6 dB
PhaseMargCan	Lower bound on the phase margin for the canard channel	45 deg
GainMargFlap	Lower bound on the gain margin for the flaperon channel	6 dB
PhaseMargFlap	Lower bound on the phase margin for the flaperon channel	-100 deg
TopGnRatio	Upper bound on the magnitude equivalent systems restriction modeled as a ratio of the actual magnitude to equation 2.2.2	1
BotGnRatio	Lower bound on the magnitude equivalent systems restriction	1
Phase_diff	Lower bound on the phase equivalent systems restriction modeled as the difference between the actual phase and equation 2.2.3	0

Table 2.2: Design constraints as implemented in CONSOLE

This chapter described the work necessary to get an initial design from CONSOLE. Further research is needed to find a way to incorporate flutter margin calculations into the design description. In addition, we would like to try optimization with different initial values for some of the parameters to see if our solution is a global optimum. We would also like to replace the current controller with one obtained from LQG/LTR and H_∞ design methods to see if there are any advantages to these types of controllers.

Chapter 3

LQG/LTR Design for the F14 Aircraft

The design method presented in this chapter is based on the solution of an optimal control problem, the Linear Quadratic Gaussian problem. The LQG problem in turn is an extension of a design known as the Linear Quadratic Regulator. Given a quadratic index gauging the performance of the plant, the LQR is the full-state feedback control which minimizes the index and is found via the solution to a Riccati equation. The LQG controller advances the design a step further by introducing a Kalman filter in the feedback path to estimate the states, assuming now that the plant observations are corrupted by white Gaussian noise. But, with the addition of the Kalman filter, the system no longer has the robustness associated with the LQR design. Loop Transfer Recovery is the process of modifying the Kalman filter gains to restore the robustness.

We begin this chapter with a description of the LQR and proceed in a manner analogous with the above discussion. Once we have a design that meets all the specifications we will compare it to two other designs to gauge its performance.

3.1 Deterministic Model

We would like to design a pitch controller for the F14 aircraft. Assume that the system is disturbance free (this restriction will be relaxed later). Based on a design point of Mach 0.71 at an altitude of 35K ft. the following criteria must be satisfied:

- Level 1 flying qualities
- Time Domain Criteria:

Tail Rate	$<$	25°/sec
Model Following Error	$<$	10 % of the step input
$\dot{N}_{z_{pilot}}$	$>$	0 g's/sec

- Gain Margin : 6 dB
- Phase Margin: 45°

We start with the F14 system description

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t); \qquad x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^p \\ y(t) &= Cx(t); \qquad\qquad y(t) \in \mathbb{R}^r. \end{aligned}$$

We construct a command generator in the fashion of a zero input model (needed for our design method) to generate a step input for the system. This model is actually just an integrator with an initial condition and an extremely low time constant. We are then given an implicit model for the pitch(α) response, derived from the military specifications for good flying qualities. These models can be grouped together as

$$\begin{aligned} \dot{x}_m(t) &= A_m x_m(t); \quad x_m(t) \in \mathbb{R}^m \\ \tilde{y}(t) &= C_m x_m(t) \quad . \end{aligned}$$

Our design goal is for the α-response of the aircraft to imitate that of the implicit model. Therefore, a good candidate for a design methodology is linear quadratic (LQ) optimal control, since we conveniently have an error term we want to minimize.

An LQ regulator is designed by a model following scheme (see [9]) with the incorporation of the models in the performance index. The design procedure is as follows. Let

$$\widehat{x}(t) = \begin{bmatrix} x(t) \\ x_m(t) \end{bmatrix}.$$

Then from the above definitions we get

$$\dot{\widehat{x}}(t) = \widehat{A}\widehat{x}(t) + \widehat{B}u(t) \tag{3.1.1}$$

with

$$\widehat{A} = \begin{bmatrix} A & 0 \\ 0 & A_m \end{bmatrix} \quad \widehat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}. \tag{3.1.2}$$

The performance index we want to minimize is

$$J_1 = \int_0^\infty \{u'(t)Ru(t) + x'(t)Q_1x(t) + (y(t) - \widetilde{y}(t))'Q_2(y(t) - \widetilde{y}(t))\}dt \tag{3.1.3}$$

where $R = R' > 0$, $Q_1 = Q_1' \geq 0$, and $Q_2 = Q_2' \geq 0$. The first term is the control cost, the second is a "smoothness" cost, and the third is an error cost. We will combine the second and third cost terms into a more compact and agreeable form as

$$x'Q_1x + e'Q_2e = \widehat{x}'\widehat{Q}\widehat{x} \tag{3.1.4}$$

where

$$\widehat{Q} = \begin{bmatrix} Q_1 + C'Q_2C & -C'Q_2C_m \\ -C_m'Q_2C & C_m'Q_2C_m \end{bmatrix}.$$

The performance index then becomes

$$J_2 = \int_0^\infty \{u'Ru + \widehat{x}'\widehat{Q}\widehat{x}\}dt. \tag{3.1.5}$$

We need to show that $\widehat{Q} \geq 0$. But from the equality in (3.1.4), if $Q_1 \geq 0$ and $Q_2 \geq 0$ then $\widehat{Q}$ must be ≥ 0. If we factor $\widehat{Q}$ as $\widehat{Q} = \widehat{D}'\widehat{D}$, we have the following theorem from regulator theory (see [9]).

Theorem 3.1 *If*

$$\begin{gathered} (\widehat{A}, \widehat{B}) \text{ is stabilizable} \\ (\widehat{D}, \widehat{A}) \text{ is detectable} \\ R > 0 \end{gathered} \tag{3.1.6}$$

then there exists a positive semi-definite matrix K such that

1. *K is the unique positive semi-definite solution to*

$$0 = K\widehat{A} + \widehat{A}'K - K\widehat{B}R^{-1}\widehat{B}'K + \widehat{Q} \tag{3.1.7}$$

2. *the matrix $\widehat{A} - \widehat{B}R^{-1}\widehat{B}'K$ is stable.*

We need to see if the assumptions in (3.1.6) can be satisfied. Certainly we can pick $R > 0$. Now, let us look at the controllability of $(\widehat{A}, \widehat{B})$. From equation (3.1.2), we see that $(\widehat{A}, \widehat{B})$ are not controllable since A_m is not controllable by B. However, since A_m is stable, $(\widehat{A}, \widehat{B})$ will be stabilizable *if and only if*(A, B) is stabilizable. But, this assumption is true since we are dealing with a real engineering system, otherwise we would be wasting our time designing a controller for a system that can't be controlled. Next, let us look at the observability of $(\widehat{D}, \widehat{A})$. $(\widehat{D}, \widehat{A})$ will be observable *if and only if*

$$\text{rank} \begin{bmatrix} \widehat{D} \\ \hline sI - A & 0 \\ 0 & sI - A_m \end{bmatrix} = n + m \quad \forall s.$$

But, since the model is stable, and hence (C_m, A_m) is detectable, and (C, A) is at least detectable (again, because it is a real engineering system it should be detectable, but this should be verified), with the proper choice of Q_1, assumption (3.1.6) will be satisfied.

We now attempt to find values for Q_1, Q_2, and R such that the design goals are met. For this example, we varied the weighting matrices manually because the system was not too large. However, in light of the previous discussion of CONSOLE, we should be able to program a simulator that would allow CONSOLE to adjust the values for the weighting matrices. Regardless of the way the weighting matrices are adjusted, this is the point of the design where the designer has to come up with different designs and rate the performance of each. Tradeoffs have to be made between Q and R to achieve a good response. Basically, we would like a small α-error without a large tail rate.(There are other performance criteria, but this appears to be the dominant one.) Note that it is only the ratio of R to Q that is important — i.e. increasing R is the same as decreasing Q. Therefore, we will fix $R = 1$ and vary only Q. After various attempts(for summary see Table 3.1), it is apparent that a satisfactory design cannot be found. We cannot simultaneously achieve a good α-error and tail rate.

Since the tail rate is a function of both u and x_1, it seems likely that adding a cross-term N, coupling u with x, to the performance index should create more

Performance criteria	α-error	tail rate
increasing Q_2	decreases	increases
increasing q_{11} of Q_1 [a]	increases	increases
increasing q_{33} of Q_1	increases	increases
increasing off-diagonal terms of Q_1	decreases	increases

[a] It looks as if decreasing q_{11} is what we want, however it can only be decreased so much before it has a negligible effect on the performance criterion.

Table 3.1: The effect of changes in the performance criteria on the dominant design goals.

independence between the tail rate and α-error tradeoff. If we define

$$\widehat{N} = \begin{bmatrix} N \\ 0 \end{bmatrix}$$

then

$$u'N'x + x'Nu = u'\widehat{N}'\widehat{x} + \widehat{x}'\widehat{N}u.$$

The performance index in equation (3.1.5) now becomes

$$J_3 = \int_0^\infty \{u'Ru + \widehat{x}'\widehat{Q}\widehat{x} + u'\widehat{N}'\widehat{x} + \widehat{x}'\widehat{N}u\}dt. \tag{3.1.8}$$

Let

$$\bar{A} = \hat{A} - \hat{B}R^{-1}\widehat{N}', \quad \text{and} \quad \bar{Q} = \widehat{Q} - \widehat{N}R^{-1}\widehat{N}' \tag{3.1.9}$$

with N chosen such that $\bar{Q} \geq 0$. Note that

$$\widehat{B}R^{-1}\widehat{N}' = \begin{bmatrix} BR^{-1}N' & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \widehat{N}R^{-1}\widehat{N}' = \begin{bmatrix} NR^{-1}N' & 0 \\ 0 & 0 \end{bmatrix}.$$

Now, factor $\bar{Q} = \bar{D}'\bar{D}$. We get the following claim as an extension of Theorem 3.1.

Claim 3.2 *If*

$$\begin{gathered} (\bar{A}, \hat{B}) \text{ is stabilizable} \\ (\bar{D}, \bar{A}) \text{ is detectable} \\ R > 0 \end{gathered} \tag{3.1.10}$$

then there exists a positive semi-definite matrix K such that

1. K is the unique positive semi-definite solution to

$$\begin{aligned} 0 &= K\widehat{A} + \widehat{A}'K - (K\widehat{B} + \widehat{N})R^{-1}(\widehat{N}' + \widehat{B}'K) + \widehat{Q} \\ &= K\bar{A} + \bar{A}'K - K\widehat{B}R^{-1}\widehat{B}'K + \bar{Q} \end{aligned} \tag{3.1.11}$$

2. the matrix $\bar{A} - \widehat{B}R^{-1}\widehat{B}'K$ is stable.

Let us examine the assumptions needed for Claim 3.2. The last two assumptions fully depend on the choice of weighting matrices and can therefore be satisfied by an appropriate choice of $\bar{Q}$ and R. Now, let us look at the controllability of $(\bar{A}, \widehat{B})$. $(\bar{A}, \widehat{B})$ will be controllable *if and only if* $(\widehat{A}, \widehat{B})$ are controllable (see eq. 3.1.9). But, from the previous discussion of assumptions (3.1.6) $(\widehat{A}, \widehat{B})$ will be stabilizable as needed.

By examining (3.1.9) and (3.1.11) we see that we can rewrite the problem formulation — given the system description

$$\dot{\hat{x}} = \bar{A}\hat{x} + \widehat{B}u \tag{3.1.12}$$

find the control u which minimizes the cost function

$$J_4 = \int_0^\infty \{u'Ru + \hat{x}'\bar{Q}\hat{x}\}dt. \tag{3.1.13}$$

It can be shown that the u that minimizes (3.1.13) is

$$u^* = -R^{-1}\widehat{B}'K\hat{x}. \tag{3.1.14}$$

Note that (3.1.14) is the optimal control when $\bar{A}$ is the system matrix for the plant. We need to rewrite the control to absorb the adjustment to the plant's A matrix as shown in (3.1.9). The adjusted optimal control is

$$u^* = -R^{-1}(\widehat{B}'K + \widehat{N}')\hat{x}. \tag{3.1.15}$$

If we partition K into the following form

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{12}' & K_{22} \end{bmatrix}$$

we can expand (3.1.15) as

$$u^* = -\underbrace{R^{-1}(BK_{11}+N)}_{K_x}x - \underbrace{R^{-1}BK_{12}}_{K_f}x_m. \tag{3.1.16}$$

The term involving K_f is implemented as a reference input with x_m driven by an initial condition. This means that we need to add the implicit model to the controller.

The system is then simulated (assuming all system and model states are available for feedback) and the weights are varied to get a design that satisfies all the design criteria. The effects of the weighting matrices on the design are now a little more complicated, since a change in N causes a change in $\bar{A}$ and $\bar{Q}$ as seen from (3.1.9). Table 3.1 is still relevant with the addition of the effects of N. For fixed R, N must be bounded(above and below, since N can be negative) so that $\bar{Q}$ remains positive semi-definite. As N increases, so does the α-error and tail rate. Therefore, N should be set equal to its lower bound(defined by R and Q_1) to achieve the lowest value for $\bar{q}_{11}$, i.e. zero.

Note that LQ theory guarantees a good gain and phase margin. However, this does not remain true with N in the performance index. The guaranteed margins are for the augmented system $\bar{A}$. Therefore, we must check the stability margins to see if the design is satisfactory. We do this by finding the gain and phase crossover frequencies from the open loop frequency response shown in Figure 3.1.

A "good" design is achieved with the following values:

$$R = 1, \quad Q_2 = 2.22, \quad Q_1 = \text{diag}\{0.8, 0, 0\}, \quad N' = [-0.89\ 0\ 0]$$

In this case, good means that the design goals are met. However, if we look at Figure 3.1, the loop transfer function at low frequencies is not very high, hence steady state error and disturbance rejection are not very good. To remedy this situation, we will add an integral feedback path in the loop.

3.2 LQ Regulator with Integral Feedback

We have shown in the last section that the steady state performance for the LQR design is inadequate. For a way to improve this, consider the following closed loop

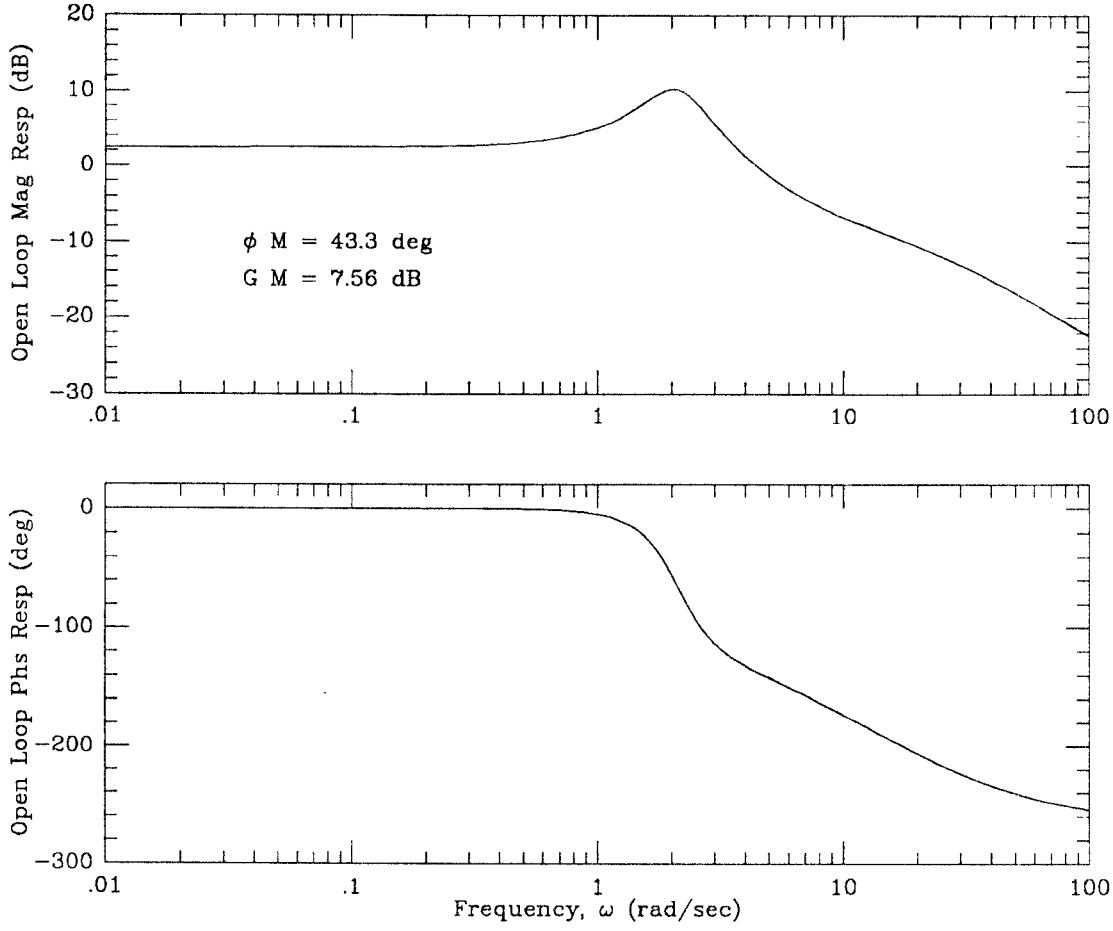
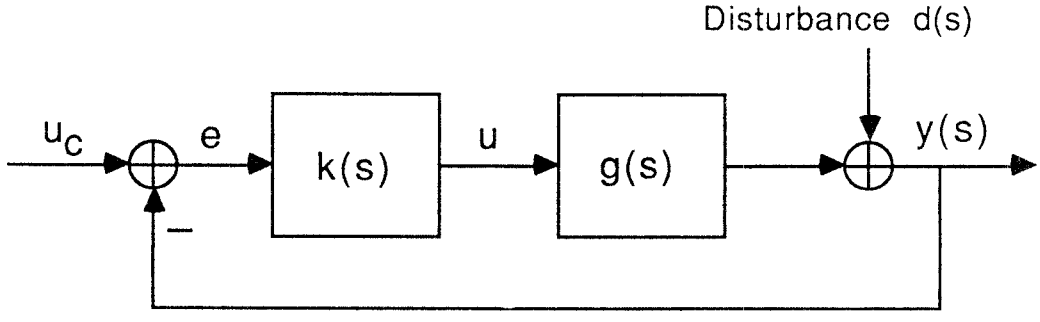


Figure 3.1: Open loop frequency response of the LQR.

feedback system:



where g(s) represents the plant, and k(s) represents the compensator. The response

to commands and disturbances is

$$\begin{aligned} y &= gke + d \\ &= gk(u_c - y) + d \\ &= (1+gk)^{-1}gk[u_c + (gk)^{-1}d] \end{aligned}$$

Consider the response to a sinusoidal command (assume $d = 0$):

$$\begin{aligned} u_c(t) &= \cos(\omega t) \\ y(t) &\rightarrow \left|\tfrac{gk(j\omega)}{1+gk(j\omega)}\right| \cos[\omega t + \angle(\tfrac{gk(j\omega)}{1+gk(j\omega)})] \\ \Rightarrow\ y(t) &\approx u_c(t) \quad \text{whenever the loop transfer function, } gk, \\ &\qquad \text{and hence the return difference, } (1+gk), \\ &\qquad \text{are large (i.e. } |gk| \gg 1). \end{aligned}$$

In particular, for the special case of $\omega = 0$ (i.e. constant commands) the steady state error will be zero if $gk(0) = \infty$. In addition, consider the response to a sinusoidal disturbance (assume $u_c = 0$):

$$\begin{aligned} d(t) &= \cos(\omega t) \\ y(t) &\rightarrow \left|\tfrac{1}{1+gk(j\omega)}\right| \cos[\omega t + \angle(\tfrac{1}{1+gk(j\omega)})] \\ \Rightarrow\ y(t) &\approx 0 \quad \text{whenever the return difference is large.} \end{aligned}$$

We can now see that adding high gain at low frequencies will improve the steady state performance and disturbance rejection. A way to add high gain to the low frequency end is to add an integral path around the loop. For a description of the overall system see Figure 3.2.

We will add an integral path as follows. Let

$$\begin{aligned} \dot{e}_i = e &= y - \tilde{y} \\ &= Cx - C_m x_m. \end{aligned} \tag{3.2.17}$$

Adding this to the performance index we get

$$J_5 = \int_0^\infty \{u'Ru + \hat{x}'\bar{Q}\hat{x} + e_i'Q_r e_i\}dt. \tag{3.2.18}$$

with $Q_r \geq 0$. If we now define

$$A_i = \begin{bmatrix} \bar{A} & 0 \\ C_i & 0 \end{bmatrix} \quad Q_i = \begin{bmatrix} \bar{Q} & 0 \\ 0 & Q_r \end{bmatrix} \quad B_i = \begin{bmatrix} \hat{B} \\ 0 \end{bmatrix} \quad C_i = \begin{bmatrix} C & -C_m \end{bmatrix} \tag{3.2.19}$$

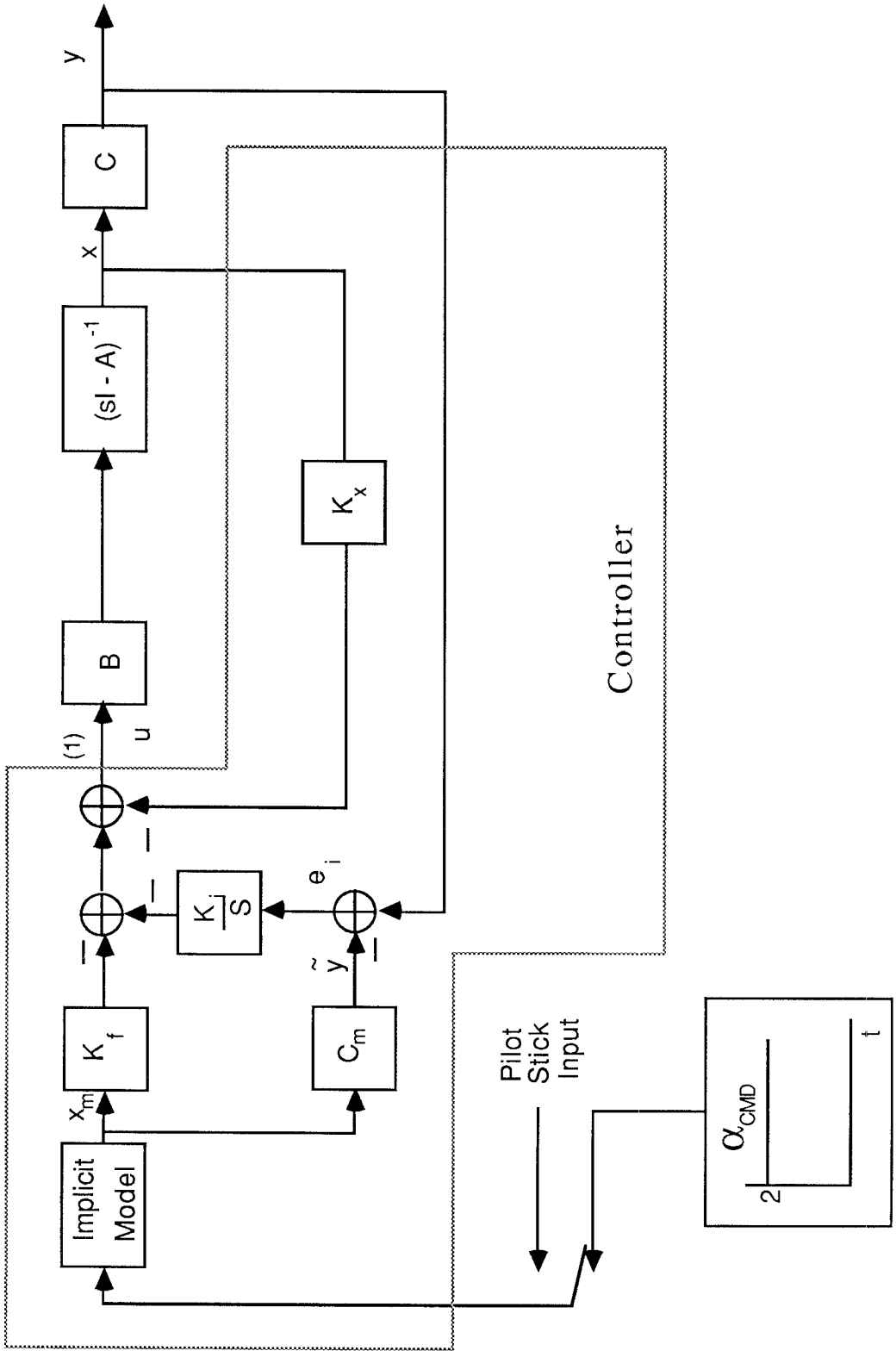


Figure 3.2: LQR design with integrator.

we can rewrite equation (3.2.18) to conform to Theorem 3.1 as

$$J_6 = \int_0^\infty \{u'Ru + x_i'Q_ix_i\}dt \tag{3.2.20}$$

subject to the constraint

$$\dot{x}_i = A_ix_i + B_iu \tag{3.2.21}$$

where

$$x_i = \begin{bmatrix} \widehat{x} \\ e_i \end{bmatrix}. \tag{3.2.22}$$

We need to check the assumptions for Theorem 3.1. Again, the last two assumptions will be satisfied by an appropriate choice for R and Q_i. Consider the controllability of (A_i, B_i). Again excluding the model, we need that

$$\text{rank} \left[\begin{array}{cc|c} sI - A & 0 & B \\ C & sI & 0 \end{array}\right] = n + p \quad \forall s \quad C \in \mathbb{R}^{p\times n}.$$

Since the original plant is controllable the only potential problem occurs at $s = 0$. At $s = 0$ we then need

$$\text{rank} \begin{bmatrix} -A & B \\ C & 0 \end{bmatrix} = n + p.$$

But this is equivalent to the condition that the original plant have no zeros at $s = 0$. Since the presence of a zero at the origin indicates zero DC gain in the plant, and since the purpose of adding integral action is to make the loop gain large at low frequencies, it is not surprising that this procedure fails in this case.

Therefore, since the original plant is stabilizable and has no zeros at the origin, the optimal control now becomes

$$u^* = -R^{-1}B_i'Kx_i. \tag{3.2.23}$$

If we partition K into the following form

$$K = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{12}' & K_{22} & K_{23} \\ K_{13}' & K_{23}' & K_{33} \end{bmatrix}$$

we can expand (3.2.23) along with the adjustment term (see eq. 3.1.15) as

$$u^* = -\underbrace{R^{-1}(BK_{11} + N)}_{K_x} x - \underbrace{R^{-1}BK_{12}}_{K_f} x_m - \underbrace{R^{-1}BK_{13}}_{K_i} e_i. \tag{3.2.24}$$

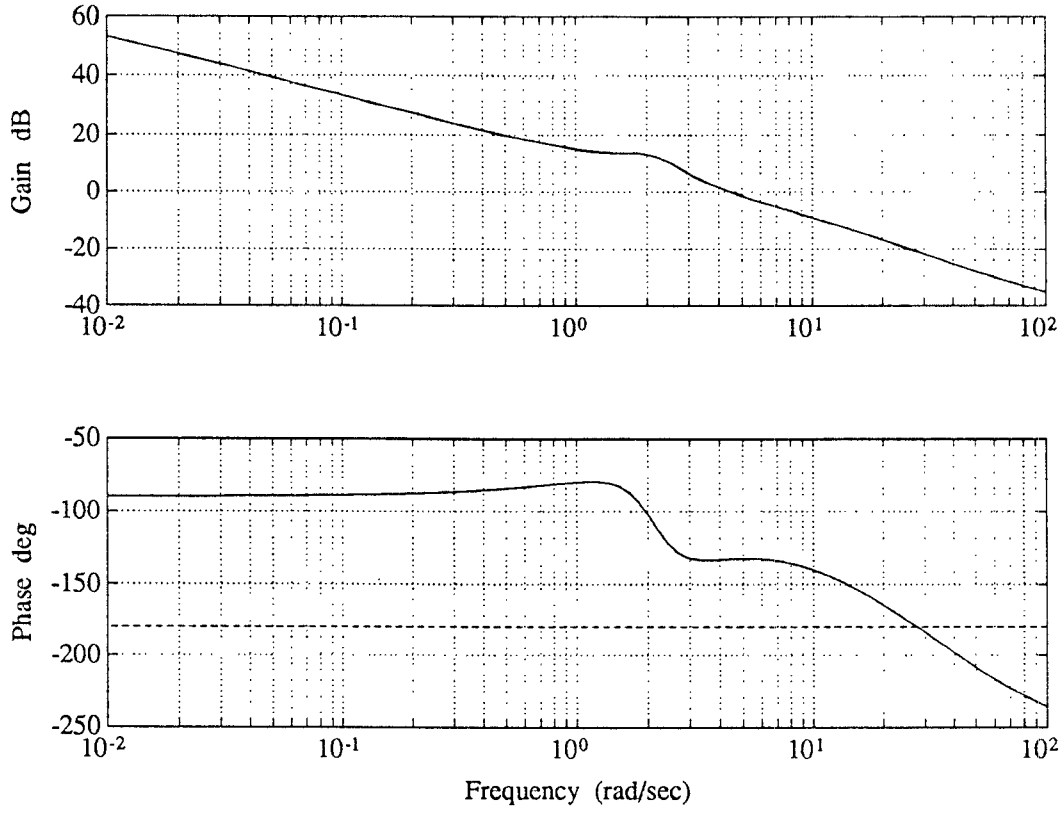


Figure 3.3: Open loop frequency response with an integral feedback path.

While the design goals are still achieved, our design has a better open loop frequency response (loop broken at point 1 in Figure 3.2), as shown in Figure 3.3, with the following values for the performance index:

$$R = 1, \quad Q_2 = 0.47, \quad Q_1 = \text{diag}\{0.15, 2.9e{-}7, 5.4e{-}8\}, \quad Q_r = 9.1, \quad N' = [-0.33\ 0\ 0]\ .$$

3.3 Stochastic model using full state feedback

The next step is to add the wind gust noise model to the system description. The system description now includes Gaussian process noise. i.e.

$$\begin{aligned} \dot{x}_g &= \begin{bmatrix} \dot{x} \\ \dot{x}_v \end{bmatrix} = \underbrace{\begin{bmatrix} A & B_v \\ 0 & A_v \end{bmatrix}}_{A_g} \begin{bmatrix} x \\ x_v \end{bmatrix} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{B_g} u + Gv \\ y &= \underbrace{[C \quad 0]}_{C_g} x_g \end{aligned} \tag{3.3.25}$$

where v is white noise with zero mean and unity covariance

G is the scaling vector with $g_5 = \frac{\sigma_{wg}}{\sqrt{a^3}}$

$\sigma_{wg} = 3.0$ feet — RMS gust velocity

$a = 2.5348$ seconds — gust correlation time.

In the above description, x_v is modeled as bandlimited white noise injected into the plant. This represents wind gust disturbances acting on the vertical direction of the aircraft.

The design goals must now include stochastic criteria for a 1σ disturbance as well:

Tail Rate	$<$	1°/sec
$N_{z_{pilot}}$	$<$	50 milli-g's
α	$<$	0.1°

If we replace A and B in eq. (3.1.2) with A_g and B_g respectively, we can follow the previous procedure to find the optimal gains with the obvious padding in Q_1 and N. Repartitioning K as

$$K = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K'_{12} & K_{22} & K_{23} & K_{24} \\ K'_{13} & K'_{23} & K_{33} & K_{34} \\ K'_{14} & K'_{24} & K'_{34} & K_{44} \end{bmatrix}$$

the optimal control now becomes

$$u^* = -K_x x - K_v x_v - K_f x_m - K_i e_i \tag{3.3.26}$$

where $K_v = R^{-1}B'K_{12}$, $K_f = R^{-1}B'K_{13}$ and $K_i = R^{-1}B'K_{14}$.

The system is simulated again to see if the covariance design criteria are satisfied. This is accomplished by checking the steady state covariance matrix of $y(t)$

$$\lim_{t\to\infty} C_{ov}(y(t)) = C_g(\lim_{t\to\infty} C_{ov}(x(t))C_g' \tag{3.3.27}$$

and

$$\begin{aligned} \tfrac{d}{dt}C_{ov}(x(t)) &= A_{cl}C_{ov}(x(t)) + C_{ov}(x(t))A_{cl}' + GG' \\ 0 &= A_{cl}(\lim_{t\to\infty} C_{ov}(x(t))) + (\lim_{t\to\infty} C_{ov}(x(t)))A_{cl}' + GG' \end{aligned}$$

where $A_{cl} = \bar{A} - \hat{B}R^{-1}\hat{B}'K$ is the closed loop system matrix. The dimension of $y(t)$ is increased, for computational purposes only, to include all the signals of interest. Simulation revealed that the stochastic criteria is not quite satisfied. At this point, we decided to see what would happen if we adjusted the off-diagonal terms in Q_1 and the terms corresponding to the disturbance model. Since we now have 36 entries in Q_1 to adjust, we decided to program CONSOLE in tandem with Pro-Matlab to adjust them. As expected, given the additional freedom in choosing Q_1, a better controller can be designed. Our design achieves the following values for wind gust effects due to a 1 σ disturbance:

$$\begin{aligned} \text{Tail rate} &= 0.632°/sec \\ N_{z_{pilot}} &= 50.7 \text{ milli-g's} \\ \alpha &= 0.055° \ . \end{aligned}$$

Notice that the $N_{z_{pilot}}$ criterion is not quite satisfied. This can most likely be corrected in the "robustness recovery" procedure in the next section. As seen in (3.3.26), the control law consists of a feedback term proportional to the the wind gust noise. But this signal is not available. Therefore we must find a way to estimate the wind gust noise so we can fully implement the control law. A solution to this problem is given in the next section in the form of a Kalman filter with a model for the wind gust noise.

3.4 LQG Regulator

The next step is to implement the Kalman-Bucy filter (KBF) to estimate the states for full state feedback. The input to the KBF is the measured angle-of-attack —

the output from the plant augmented with an α-sensor error model. The α-sensor error is modeled as band-limited white noise with

$$\begin{aligned}\omega_\alpha &= 10 \text{ rad/sec (half-variance density bandwidth)}\\ \sigma_\alpha &= 5.236\text{e-}3 \text{ rad}\end{aligned}$$

The model for the plant output now looks like

$$y = C_g x_g + x_w \tag{3.4.28}$$

where $\dot{x}_w = -x_w\omega_\alpha + Hw$

w is white noise with zero mean and unity covariance

H is a scaling factor $= \sqrt{2\omega_\alpha}\,\sigma_\alpha$.

As with the wind gust disturbance, we must also meet the same restrictions but with the 1 σ disturbance entering through the sensor.

From the system described by (3.3.25) and (3.4.28) and by theorem 3.1 we have the following claim.

Claim 3.3 *If*

$$\begin{gathered}(\tilde{A}, G) \text{ is stabilizable}\\ (C_g, \tilde{A}) \text{ is detectable}\\ HH' > 0\end{gathered} \tag{3.4.29}$$

then

1. *P is the unique positive semi-definite solution to the filter Riccati eq.*

$$0 = \tilde{A}P + P\tilde{A}' + GG' - PC_g'(HH')^{-1}C_gP \tag{3.4.30}$$

2. *the matrix $\tilde{A} - PC_g'\Theta^{-1}C_g$ is stable*

where $\Theta = HH'$

$\tilde{A} = A_g - B_gR^{-1}\,[N' \;\; 0]$.

If in addition, assumption (3.1.10) holds, then the optimal control is

$$\bar{u}^o = -\underbrace{[K_x \;\; K_v]}_{K_g} z_t - K_f x_m \tag{3.4.31}$$

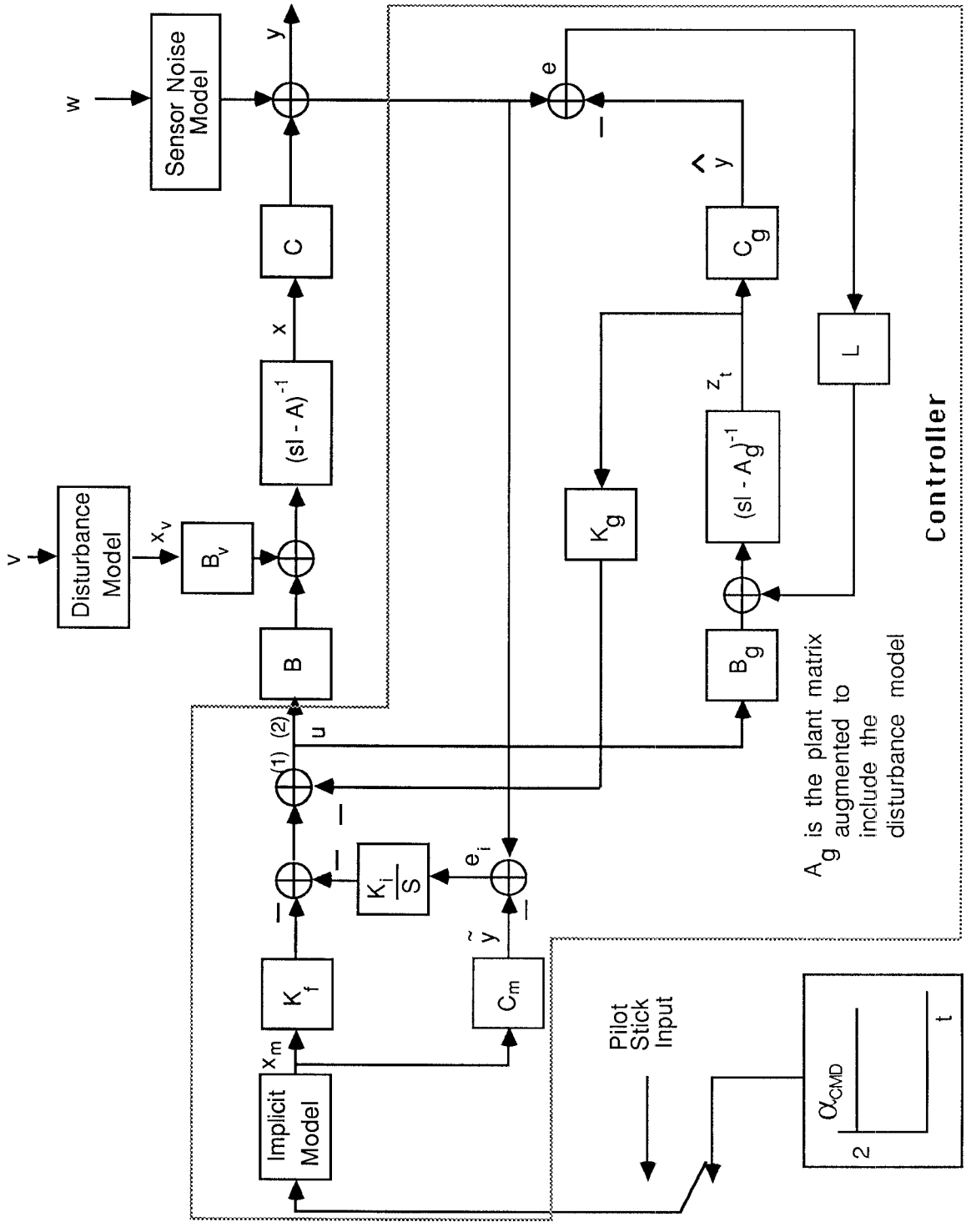


Figure 3.4: LQG/LTR design (in box) for the F14 aircraft.

where

$$\dot{z}_t = (\tilde{A} - B_g K_g) z_t + L(y - C_g z_t) \quad L = P C_g' \Theta^{-1}.$$

For a diagram of the complete system, see Figure 3.4.

A solution for the filter Riccati equation (3.4.30) is now attempted. The initial simulation reveals that the response to the wind gust disturbance is too high, while the sensor noise response is good by more than 2 orders of magnitude. We can approximate the LQR (with integral feedback) design by a procedure called "full-state loop transfer recovery" [10]. The procedure makes the loop transfer function obtained by breaking the LQG loop at point (2) in Figure 3.4 approach the LQR loop transfer function (obtained by breaking the loop at point (1)) pointwise in s. The required assumptions are 1) there are at least as many outputs as inputs (i.e. $r \geq p$), and 2) the plant $(C\Phi B)$ is minimum phase. In the following discussion, a slight abuse of notation will be used. Since

$$C_g \Phi_g B_g = C\Phi B \quad \text{where} \quad \Phi_g = (sI - A_g)^{-1},$$

and

$$K_g \Phi_g B_g = K_x \Phi B$$

we will use B to represent B or B_g and C to represent C or C_g. There are two steps to the procedure.

1) Append additional dummy columns to B and zero rows to K_x to make $C\Phi B$ and $K_x \Phi B$ square $(r \times r)$. $C\Phi B$ must remain minimum phase.

2) Design a modified KBF by replacing the filter Riccati eq. (3.4.30) by

$$0 = \tilde{A}P + P\tilde{A}' + (GG' + q^2 BB') - PC'\Theta^{-1}CP \tag{3.4.32}$$

where q is a scalar parameter. Under these conditions, it is known that

$$\frac{1}{q} L \underset{q \to \infty}{\longrightarrow} BW\Theta^{-1/2} \tag{3.4.33}$$

where W is an orthonormal matrix. When this L is used in the loop transfer expression for point (2) we get pointwise recovery as $q \to \infty$, i.e.,

$$K_{LQG}(s)G(s) = K_g(\Phi_g^{-1} + BK_g + LC)^{-1} LC\Phi B \tag{3.4.34}$$

$$= K_g \bar{\Phi}_g L(I_r + C\bar{\Phi}_g L)^{-1} C\Phi B \tag{3.4.35}$$

$$\underset{q\to\infty}{\longrightarrow} \quad K_g\bar{\Phi}_g B(C\bar{\Phi}_g B)^{-1}C\Phi B \tag{3.4.36}$$

$$= \quad \{K_g\Phi_g B(C\Phi_g B)^{-1}\}C\Phi B \tag{3.4.37}$$

$$= \quad K_x\Phi B. \tag{3.4.38}$$

In the steps above, $\bar{\Phi}_g$ was used to represent $(sI - A_g + BK_g)^{-1}$, the identity

$$(A+BD)^{-1} = A^{-1} - A^{-1}B(I+DA^{-1}B)^{-1}DA^{-1}$$

was used to get from (3.4.34) to (3.4.35), and the identity

$$\bar{\Phi}_g B = \Phi_g B(I + K_g\Phi_g B)^{-1}$$

was used to get from (3.4.36) to (3.4.37). Note that the above discussion intentionally ignores the dynamics due to the integral feedback path, since it will be identical in both cases. Eq. (3.4.37) shows explicitly that the asymptotic compensator (the term in brackets) inverts the nominal plant and substitutes the desired LQR dynamics. This is achieved at the expense of the sensor noise performance. Therefore we raise q until we reach a favorable tradeoff between the noise performance and the desired loop transfer properties.

The final design parameters can be found in Appendix C. Our design achieves the following values for the disturbance criteria:

- Effects of wind gust disturbance:

$$\begin{aligned} \text{Tail rate} &= 0.59°/sec \\ N_{z_{pilot}} &= 49.5 \text{ milli-g's} \\ \alpha &= 0.068° \ . \end{aligned}$$

- Effects of sensor noise:

$$\begin{aligned} \text{Tail rate} &= 0.73°/sec \\ N_{z_{pilot}} &= 1.7 \text{ milli-g's} \\ \alpha &= 0.0055° \ . \end{aligned}$$

We can see the change in the frequency response (with the loop broken at point 2 in Figure 3.4) of Figure 3.5 to approximate that of the original LQR design.

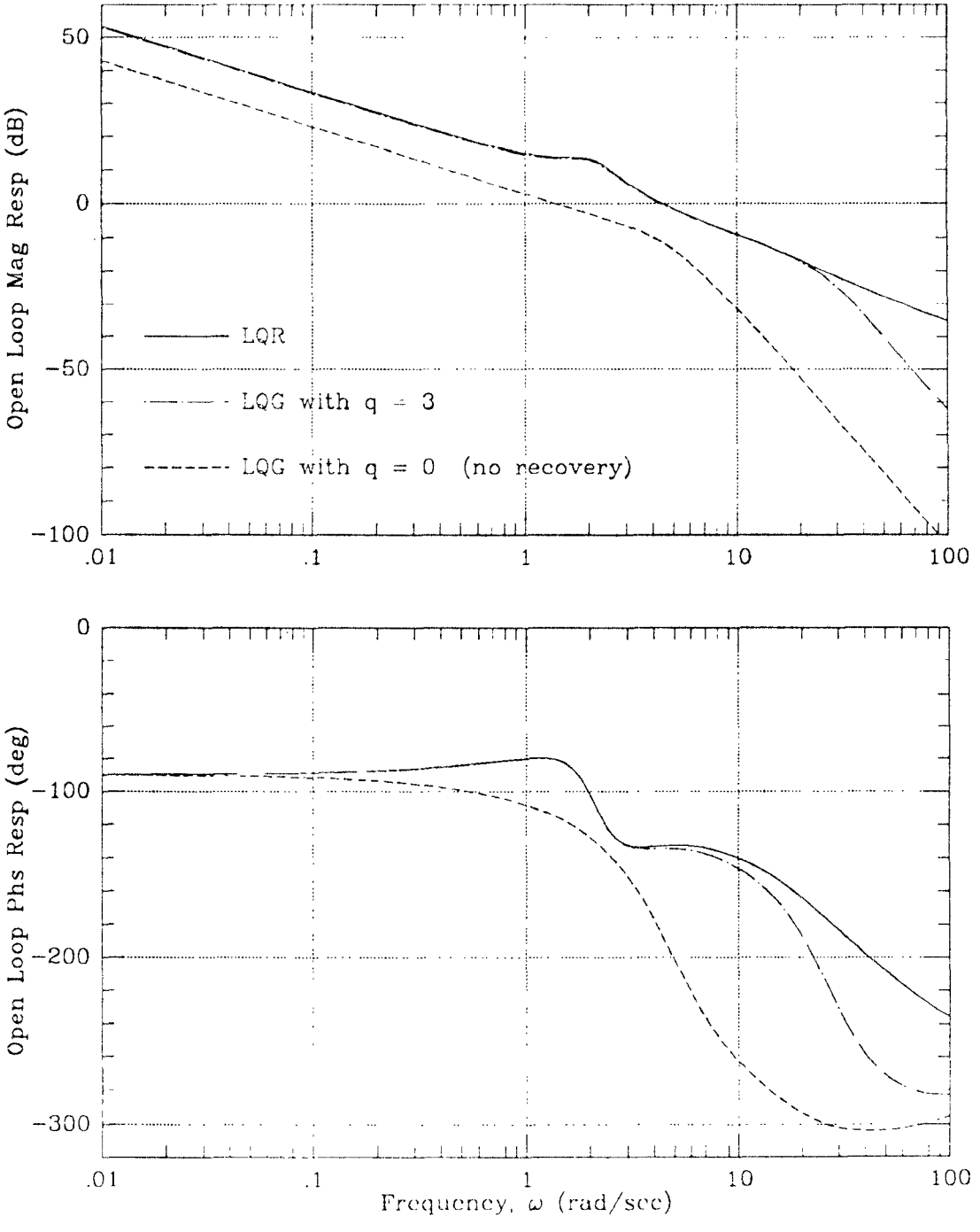


Figure 3.5: Open loop frequency response of the LQG/LTR design.

3.5 Comparison with Other Designs

Knowing that our LQG/LTR controller satisfies all the design criteria, we would like to compare our design to others done for the aircraft. We have at hand two other pitch controller designs — a design done at Grumman [11] using output feedback, and one done at the University of Maryland [12]. Maryland's design started from the design at Grumman and varied the gains using Delight — an optimization package similar to CONSOLE. During the course of the discussion, we will be referring to the figures in Appendix D which give a comparison between the three designs.

From the figures, we see that the three designs give similar results. Our design has a higher initial α-error than the other designs, but this is relatively small anyway so this is not a problem. The steady state error of our design compares favorably with the other designs. The vertical acceleration felt at the pilot station for our design has a smoother transition than the other designs, which means that the pilot may like the handling of the aircraft better with our controller (if this difference is enough to be felt). Note also that the Delight design will not be rated favorably by the pilot since the vertical acceleration is not monotonic. The smoother transition is a result of the difference in the tail rate. We decided to use the full range of the control surface under the assumption that the tail rate was limited only by the actuator, hence we should take advantage of all the control available. (This assumption may not be correct, the tail rate restriction may also be to limit the g force on the pilot. However, by looking at simulations with various large tail rates, the N_z at the pilot station does not seem to vary much.) The open loop frequency response of the designs are similar with the following stability margins.

Design	Gain Margin	Phase Margin
LQG/LTR	15.9 dB	45.6 deg
Grumman/Maryland	∞	63.5 deg

The following tables show the stochastic performance of each design.

- Wind gust disturbance effects:

Signal	Design		
	LQG/LTR	Grumman	Maryland
Tail rate	0.59°/sec	0.62°/sec	0.8°/sec
$N_{z_{pilot}}$	49.5 milli-g's	37 milli-g's	38 milli-g's
α	0.068°	0.09°	0.09°

- α-sensor noise effects:

Signal	Design		
	LQG/LTR	Grumman	Maryland
Tail rate	0.73°/sec	0.6°/sec	0.9°/sec
$N_{z_{pilot}}$	1.7 mill-g's	15 milli-g's	16 milli-g's
α	0.0055°	0.07°	0.08°

Another point of comparison is the implementation of each controller. Since Maryland's design is only an adjustment of the one done at Grumman, we need only compare our design with Grumman's. Grumman's design requires 8 gain terms and 4 integrators to install the controller. Our design requires: six states for the KBF along with six gains, 2 states for the model along with 3 gains, and an integrator with a gain. All totaled we will have 9 integrators and about 16 gain terms. This is about twice the size of the other design. However, in the age of the digital computer, the size of our controller is probably not a problem.

Chapter 4

Conclusions and Future Research

We have demonstrated two methods for designing aircraft controllers. The first, using CONSOLE, requires a structure for the controller while it varies the parameters to achieve a good design as stated by the designer. The second method, LQG/LTR, gives a form for the controller while it is up to the designer to vary the weights to achieve some criteria.

CONSOLE in itself is not a procedure for designing controllers. It is merely a tool for the designer to use to adjust the controller parameters to achieve (or attempt to achieve) a satisfactory design. Our design of a controller for the X-29 is a variation of a design done at Grumman. By showing that realistic design specifications can be described in CONSOLE, we have shown that CONSOLE is a viable tool for the designer to have in his/her toolbox of computer-aided design tools.

The LQG/LTR design process shown here takes the controller design from start to finish. The only prerequisites are the plant model and some criteria to gauge the design. The design procedure involves adjustment of the performance index weighting matrices to achieve a tradeoff between the various specifications. With this procedure we have achieved a relatively good design given realistic design specifications. Note that our design doesn't quite have the *nice* stability margins guaranteed by LQR theory. This is due to the specification on the tail rate and the need to include a coupling term in the performance index. In essence, this restriction changes the apparent dynamics of the tail surface actuator such that the tail rate is not excessive. This change brings the actuator dynamics into the range of the system dynamics, hence introducing a phase lag which decreases the

stability margins. An alternative way of dealing with the tail rate restriction might be to introduce a pre-filter to slow the response while maintaining the robustness of LQR. Although our design example is a single-input, single-output system, the procedure given applies equally to the multivariable case. The only difference is the definition of the stability that comes out of the design.

In designing the LQG/LTR controller, it was necessary to manipulate multiple weighting matrices in an attempt to satisfy the design criteria. This part of the process was trial and error. It seems that we have a tool that can do this job for us — CONSOLE. The next logical step would be to link a program that calculates the feedback gains to CONSOLE, and have it vary the weighting matrices to find a good design. Note that our design was not optimized to find the *best* LQG/LTR controller, only one which satisfied the criteria. We feel that a better design can be found if some numerical optimization is used. We should keep this in mind when making comparisons with similar results.

We have designed a relatively good controller for the F14 using the robust design method LQG/LTR. It would be interesting to see how this design compares with other robust methods such as H_∞.

Appendix A

Problem Description File

```
/*================
  c953  --  Specifies parameters and such for Grumman BFF design c953
 ================ */


/* Parameters and bounds for second order model for pitch rate */
 tautheta2  = .4713                    /* numerator time constant */
 taueff_max =  .1                      /* maximum delay allowed */
 wsp_squared_spec_min    = 0.28
 wsp_squared_spec_max    = 3.6         /* bounds on equivalent nat. freq */
 ksisp_min  =  .35
 ksisp_max =  1.3                      /* bounds on eq. dumping coeff. */

/* Conversion constants */
 PI = 2 * asin(1)
 TO_DEG = 180/PI

TIME_BEGIN = 0
TIME_END = 2
TIME_INC = 0.01
Nz_t_END = 3
Nz_INC = 2*TIME_INC

MATCH_START = 0.3
FREQ_START = 1.0
FREQ_END = 100
FREQ_INC = 1
FREQ_START2 = 0.1
FREQ_END2 = 10
FREQ_INC2 = 0.2

/* initial design parameters  */
design_parameter K1 init =  1.00    variation = 0.5
design_parameter K2 init =  0.2610  variation = 0.13
design_parameter K3 init = -0.2285  variation = 0.11
design_parameter K4 init = -2.562   variation = 1.25
design_parameter K5 init =  0.401   variation = 0.2
design_parameter K6 init =  7.471   variation = 3.7

/* where K1 =  K_Qf
        K2 =  K_deltaNz
        K3 = -K_F * K_Nz
        K4 = -K_Q * K_F
        K5 =  K_F
        K6 =  K_I / K_F        */

include "specs_1"
include "can_margin_2"
include "flap_margin_3"
include "eq_sys_4"
```

```
/*=================
  specs-  Specifies the constraints on the input and CAP specs
                and the Nzerror for Grumman BFF design c953
 ================= */


/*                        ************************
                            1-G INPUT SPECS
                          ************************

  ===============
  functional_constraint --canard displacement (input)
  =============== */

functional_constraint 'CanDisp_I' soft
     for TIME from TIME_BEGIN to TIME_END by TIME_INC
{

     double Ytr_1();

     return fabs(Ytr_1("9",TIME));
}
<=
     good_curve = { return 0.140; }      /* values in rad.  */
     bad_curve  = { return 0.1575; }

/*     good_curve = { return 8.0; }   values in degrees
       bad_curve  = { return 9.0; }                        */


/*===============
  functional_constraint --canard rate (input)
  =============== */

functional_constraint 'CanRate_I' soft
     for TIME from TIME_BEGIN to TIME_END by TIME_INC
{

     double Ytr_1();

     return fabs(Ytr_1("10",TIME));
}
<=
     good_curve = { return 1.05; }   /* values in rad. */
     bad_curve  = { return 1.13; }

/*     good_curve = { return 60; }    values in deg.
       bad_curve  = { return 65; }                    */

/* ===============
  functional_constraint --flap displacement (input)
  =============== */
```

```
functional_constraint 'FlapDisp_I' soft
     for TIME from TIME_BEGIN to TIME_END by TIME_INC
{

     double Ytr_1();

     return fabs(Ytr_1("5",TIME));                                    60
}
<=
     good_curve = { return 0.175; }
     bad_curve  = { return 0.210; }

/*     good_curve = { return 10; }  values in deg.
       bad_curve  = { return 12; }                          */

/* ================
  functional_constraint --flap rate (input)                         70
  ================ */

functional_constraint 'FlapRate_I' soft
     for TIME from TIME_BEGIN to TIME_END by TIME_INC
{

     double Ytr_1();

     return fabs(Ytr_1("6",TIME));
}                                                                  80
<=
     good_curve = { return 1.75; }
     bad_curve  = { return 2.10; }

/*     good_curve = { return 100; }   values in deg.
       bad_curve  = { return 120; }                         */


/*                  ========================
                        CAP SPECS                                  90
                    ========================
*/

/*================
   objective  -  Lower bound on CAP
  ================ */
constraint 'CAPlowbd' soft
 {
#include "macros.h"
#include "macros"                                                  100

        import TIME_BEGIN,TIME_END,TIME_INC;
        double Ytr_1(),Qdotmax,time,CAP;
```

```
        max(Ytr_1("2",time),time,TIME_BEGIN,TIME_END,TIME_INC);
                /* Get the time at which Qdot -- the first derivative
                   of the pitch rate -- has it's  maximum value. */
        Qdotmax=Ytr_1("2",time);

        CAP = Qdotmax/Ytr_1("1",TIME_END);                              110
                /*CAP = Qdotmax/Nzss (pitch rate/ steady
                        state value of normal acceleration) */
        return CAP;
    }
    >=
    good_value = 0.25
    bad_value  = 0.15


/*                                                                      120
 ====================
  constraint  -  Upper bound on CAP
 ==================== */

constraint 'CAPupbd' soft
{
#include "macros.h"
#include "macros"

        import TIME_BEGIN,TIME_END,TIME_INC;                            130
        double Ytr_1(),Qdotmax,time,CAP;

        max(Ytr_1("2",time),time,TIME_BEGIN,TIME_END,TIME_INC);
                /* Get the time at which Qdot -- the first derivative
                   of the pitch rate -- has it's  maximum value. */
        Qdotmax=Ytr_1("2",time);

        CAP = Qdotmax/Ytr_1("1",TIME_END);
                /*  CAP = Qdotmax/Nzss (pitch rate/ steady
                        state value of normal acceleration) */          140
        return CAP;
    }
    <=
     good_value = 1.5              /* As stated in Bischoff */
     bad_value  = 1.7


/*================
  Objective  -  Minimize Nz error, more precisely minimize Y(13) = NZS-NZM
 ================ */                                                   150

/* objective  'NzError' minimize Ytr(18) good=0.2 bad=0.4 */

  functional_objective 'NzError'
    for time from TIME_BEGIN to Nz_t_END by Nz_INC
    minimize {
```

```
        double Ytr_1(), Ytr_5();

        return fabs(Ytr_1("1",time) - Ytr_5("2",time));
    }                                                                    160
    good_curve = { return 0.4*exp(-time); }
    bad_curve  = { return 0.6*exp(-time); }


/*===============
  Objective  - Minimize Qdot error, more precisely minimize Y(15) = QdotS-QdotM
 =============== */

  functional_objective 'QdotError'
    for time from TIME_BEGIN to Nz_t_END  by Nz_INC                      170
    minimize {
     double Ytr_1(), Ytr_5();

     return fabs(Ytr_1("2",time) - Ytr_5("3",time));
    }
    good_curve = { return 0.4*exp(-time); }
    bad_curve  = { return 0.6*exp(-time); }


/*=================================================== 180
        Dummy Functional constraints used to plot signals of
        interest
  ================================================== */

   functional_constraint 'Qdot (A/C)' soft    /* in rad**2/s  */

        for TIME from TIME_BEGIN to TIME_END by TIME_INC

        {  double Ytr_1();
           return Ytr_1("2",TIME);                                       190
           }

        >=
        good_curve = { return -100; }
        bad_curve = { return -101; }

   functional_constraint 'Qdot_mod' soft    /* in rad**2/s  */

        for TIME from TIME_BEGIN to TIME_END by TIME_INC
                                                                         200
        {  double Ytr_5();
           return Ytr_5("3",TIME);
           }

        >=
        good_curve = { return -100; }
        bad_curve = { return -101; }
```

```
functional_constraint 'Nz (A/C)' soft    /* in g's  */
                                                                    210
     for TIME from TIME_BEGIN to TIME_END by TIME_INC

     {  double Ytr_1();
        return Ytr_1("1",TIME);
        }

     <=
     good_curve = { return 100; }
     bad_curve = { return 101; }
                                                                    220
functional_constraint 'Nz_model' soft    /* in g's  */

     for TIME from TIME_BEGIN to TIME_END by TIME_INC

     {  double Ytr_5();
        return Ytr_5("2",TIME);
        }

     <=
     good_curve = { return 100; }                                   230
     bad_curve = { return 101; }

functional_constraint 'Total WB' soft    /* in ft  */

     for TIME from TIME_BEGIN to TIME_END by TIME_INC

     {  double Ytr_1();
        return Ytr_1("3",TIME);
        }
                                                                    240
     >=
     good_curve = { return -100; }
     bad_curve = { return -101; }

functional_constraint 'Alpha(A/C)' soft    /* in rad  */

     for TIME from TIME_BEGIN to TIME_END by TIME_INC

     {  double Ytr_1();
        return Ytr_1("7",TIME);                                     250
        }

     <=
     good_curve = { return 100; }
     bad_curve = { return 101; }

functional_constraint 'WB1_rate' soft    /* in ft/sec  */

     for TIME from TIME_BEGIN to TIME_END by TIME_INC
                                                                    260
```

```
    { double Ytr_1();
      return Ytr_1("11",TIME);
      }

    >=
    good_curve = { return -100; }
    bad_curve = { return -101; }

functional_constraint 'Canard_D' soft    /* in rad  */

    for TIME from TIME_BEGIN to TIME_END by TIME_INC

    { double Ytr_1();
      return Ytr_1("9",TIME);
      }

    >=
    good_curve = { return -100; }
    bad_curve = { return -101; }

functional_constraint 'Canard_R' soft    /* in rad/sec  */

    for TIME from TIME_BEGIN to TIME_END by TIME_INC

    { double Ytr_1();
      return Ytr_1("10",TIME);
      }

    >=
    good_curve = { return -100; }
    bad_curve = { return -101; }

functional_constraint 'Flap_D' soft    /* in rad  */

    for TIME from TIME_BEGIN to TIME_END by TIME_INC

    { double Ytr_1();
      return Ytr_1("5",TIME);
      }

    >=
    good_curve = { return -100; }
    bad_curve = { return -101; }

functional_constraint 'Flap_R' soft    /* in rad/sec  */

    for TIME from TIME_BEGIN to TIME_END by TIME_INC

    { double Ytr_1();
      return Ytr_1("6",TIME);
      }
```

```
        >=
        good_curve = { return -100; }
        bad_curve = { return -101; }

functional_constraint 'Q (A/C)' soft    /* in rad/sec  */

        for TIME from TIME_BEGIN to TIME_END by TIME_INC
                                                                    320
        {  double Ytr_1();
           return Ytr_1("8",TIME);
           }

        <=
        good_curve = { return 100; }
        bad_curve = { return 101; }
```

```
/*================
  can_margin-  Specifies canard stability margin constraints for Grumman
  BFF design c953
 ================ */


/*
                    ***************************************
                    STABILITY MARGINS FOR CANARD LOOP
                    ***************************************                    10

        U(6)=1        # U(6) is set here to represent a unit impulse
                      # input to the proportional plus
                      # integral compensator right before the input to the
                      # canard actuator.  This is where the loop is cut.


 ===============
  constraint  - Lower bound on gain margin for loop cut at
 =============== canard actuator */                                          20

constraint 'GainMargCan' soft
{

#include "macros.h"
#include "macros"
#include "phase"

     import TO_DEG, FREQ_START, FREQ_END;
     double Ymagn_2(),Yphas_2(),freq,gmargin;                               30

     firstroot(phase(Yphas_2("1",freq))+180,freq,FREQ_START,FREQ_END);
     gmargin = to_Db(Ymagn_2("1",freq));

     return gmargin;
}
>=
     good_value = 6
     bad_value =  5
                                                                            40

/*================
  constraint  - Lower bound on phase margin for loop cut at
 =============== canard actuator */

constraint 'PhaseMargCan' soft
{
#include "macros.h"
#include "macros"
#include "phase"                                                            50

     import TO_DEG, FREQ_START, FREQ_END;
```

```
    double Ymagn_2(),Yphas_2(),freq,pmargin;

    firstroot(Ymagn_2("1",freq)-1,freq,FREQ_START,FREQ_END);
    pmargin = phase(Yphas_2("1",freq)) + 180 ;

    return pmargin;
}
>=                                                                    60
    good_value = 45
    bad_value =  42

/*=================================================
      The next two functional constraints are used to plot the open
      loop gain and phase plots.  They use the 2nd _t file, with
      U(6) = 1 and the feedback loop broken
  ================================================
*/
                                                                      70
functional_constraint "OLp_Gain_c" soft
    for freq from FREQ_START to FREQ_END by FREQ_INC
{
    double Ymagn_2(),gain;

    gain =  to_Db(Ymagn_2("1",freq));
    return gain;
}
>=
    good_curve = { return -100; }                                     80
    bad_curve = { return -150; }

functional_constraint "OLp_Phs_c" soft
    for freq from FREQ_START to FREQ_END by FREQ_INC
{
#include "phase"
    import TO_DEG;
    double Yphas_2(),theta;

    theta = phase(Yphas_2("1",freq));                                 90
    return theta;
}
<=
    good_curve = { return 200; }
    bad_curve = { return 250; }
```

```
/*================
  flap_marg-  Specifies flaperon stability margin constraints for Grumman
  BFF design c953
 ================ */


/*                  ****************************************
                    STABILITY MARGINS FOR FLAPERON LOOP
                    ****************************************                10

         U(5)=1          # U(5) is set to represent a unit impulse
                         # input to the flap actuator.



 ==============
  constraint  -- Lower bound on gain margin for loop cut at flap
 ==============   actuator */
                                                                             20
constraint 'GainMargFlap' soft
{

#include "macros.h"
#include "macros"
#include "phase"

     import TO_DEG, FREQ_START, FREQ_END;
     double Ymagn_3(),Yphas_3(),freq,gmargin;
     firstroot(phase(Yphas_3("1",freq))+180,freq,FREQ_START,FREQ_END);       30
     gmargin = to_Db(Ymagn_3("1",freq));

     return gmargin;
}
>=
     good_value = 6
     bad_value =  5

/*==============
  constraint  -- Lower bound on phase margin for loop cut at flap            40
 ==============   actuator */

constraint 'PhaseMargFlap' soft
{

#include "macros.h"
#include "macros"
#include "phase"

     import TO_DEG, FREQ_START, FREQ_END;                                    50
     double Ymagn_3(),Yphas_3(),freq,pmargin;
```

```
    firstroot(Ymagn_3("1",freq)-1,freq,FREQ_START,FREQ_END);
    pmargin = phase(Yphas_3("1",freq)) + 180 ;

    return pmargin;
}
>=
/*    good_value = 55
    bad_value =  50  */

      good_value = -100
      bad_value  = -110

/*================================================
      The next two functional constraints are used to plot the
      open loop gain and phase plots.
  ================================================
*/

functional_constraint "OLp_Gain_f" soft
    for freq from FREQ_START to FREQ_END by FREQ_INC
{
    double Ymagn_3(),gain;

    gain =  to_Db(Ymagn_3("1",freq));
    return gain;
}
>=
    good_curve = { return -100; }
    bad_curve = { return -150; }


functional_constraint "OLp_Phs_f" soft
    for freq from FREQ_START to FREQ_END by FREQ_INC
{
#include "phase"

    import TO_DEG;
    double Yphas_3(),theta;

    theta = phase(Yphas_3("1",freq));
    return theta;
}
<=
    good_curve = { return 200; }
    bad_curve = { return 250; }
```

```
/*========
  eq_sys -- Specifies equivalent system constraints for the c953 BFF problem
 ========  */


/*=============================================
  = The next 3 functional constraints aim at fitting between 2 given
  = second order models the transfer function from output of command
  = generator to pitch rate.  This is our method of achieving a
  = desirable CAP.
  =============================================  */

/*      U(8) = 1 */

/*================
   func constraint  --  Upper bound on the ratio of the magnitude to the
  ================       top given model */


   functional_constraint 'TopGnRatio' soft
        for freq from MATCH_START to FREQ_END2 by FREQ_INC2
   {

        import wsp_squared_spec_max, ksisp_min, tautheta2, TIME_END;
        double t1, t2, t3, t4, DC, top_ref, wsp_max, n_over_alpha;
        double Ymagn_4(), Ytr_4();

        n_over_alpha = Ytr_4("2",TIME_END)/Ytr_4("3",TIME_END);
        DC = Ymagn_4("1",0e0);
        wsp_max = sqrt(wsp_squared_spec_max * n_over_alpha);

        t1 = freq*tautheta2;
        t2 = freq/wsp_max;
        t3 = 1-t2*t2;
        t4 = 2*ksisp_min*freq/wsp_max;
        top_ref = DC*sqrt((1+t1*t1)/(t3*t3+t4*t4));

        return (Ymagn_4("1",freq)/top_ref);
      }
        <=
        good_curve = { return 1.0;}
        bad_curve  = { return 2.0;}


/*================
  func constraint  --  Lower bound on the ratio of the magnitude to the bottom
  ================       given model */


   functional_constraint 'BotGnRatio' soft
      for freq from MATCH_START to FREQ_END2  by FREQ_INC2
```

```
        {

        import wsp_squared_spec_min, ksisp_max, tautheta2, TIME_END;
        double t1, t2, t3, t4, DC, bottom_ref, wsp_min, n_over_alpha;
        double Ymagn_4(), Ytr_4();

        n_over_alpha = Ytr_4("2",TIME_END)/Ytr_4("3",TIME_END);                    60
        DC = Ymagn_4("1",0e0);
        wsp_min = sqrt(wsp_squared_spec_min * n_over_alpha);

        t1 = freq*tautheta2;
        t2 = freq/wsp_min;
        t3 = 1-t2*t2;
        t4 = 2*ksisp_max*freq/wsp_min;
        DC = Ymagn_4("1",0e0);
        bottom_ref = DC*sqrt((1+t1*t1)/(t3*t3+t4*t4));
                                                                                   70
        return (Ymagn_4("1",freq)/bottom_ref);
        }

      >=
      good_curve = { return 1.0;}
      bad_curve  = { return 0.5;}


/*================
  func constraint  -  Lower bound on the phase with the                            80
  ================     most-negative-phase model */


   functional_constraint  'Phase_diff' soft
      for freq from FREQ_START2 to FREQ_END2  by FREQ_INC2

   {

        import wsp_squared_spec_min, ksisp_max , tautheta2, TO_DEG, taueff_max;
        import TIME_END;                                                           90
        double denom_real, denom_imag, q_phase;
        double wsp_min, n_over_alpha;
        double Yphas_4(), Ytr_4();

        n_over_alpha = Ytr_4("2",TIME_END)/Ytr_4("3",TIME_END);
        wsp_min = sqrt(wsp_squared_spec_min * n_over_alpha);

        denom_real = 1-(freq/wsp_min)*(freq/wsp_min);
        denom_imag = 2*ksisp_max*freq/wsp_min;
        q_phase = TO_DEG * (atan2(freq*tautheta2,1.0) -                            100
                            atan2(denom_imag,denom_real) -
                            taueff_max*freq);
        return (TO_DEG*Yphas_4("1",freq) - q_phase);
        }
```

```
      >=
        good_curve = { return 0; }
        bad_curve  = { return -25; }

/*================================================== 110
        The next two functional constraints plot the upper bound
        and lower bound magnitude curves for q/x46  X46 = stick
        cmd output from gen.
  ==================================================
*/

 functional_constraint 'Top_curve' soft

      for freq from FREQ_START2 to FREQ_END2  by FREQ_INC2
        {                                                 120

        import wsp_squared_spec_max, ksisp_min, tautheta2, TIME_END;
        double t1, t2, t3, t4, DC, top_ref, wsp_max, n_over_alpha;
        double Ymagn_4(), Ytr_4();

        n_over_alpha = Ytr_4("2",TIME_END)/Ytr_4("3",TIME_END);
        DC = Ymagn_4("1",0e0);
        wsp_max = sqrt(wsp_squared_spec_max * n_over_alpha);

        t1 = freq*tautheta2;                              130
        t2 = freq/wsp_max;
        t3 = 1-t2*t2;
        t4 = 2*ksisp_min*freq/wsp_max;

        return (DC*sqrt((1+t1*t1)/(t3*t3+t4*t4)));
        }
   <=
      good_curve = {  return 200.0; }
      bad_curve = { return 300.0; }
                                                          140
functional_constraint 'Bot_curve' soft

      for freq from FREQ_START2 to FREQ_END2  by FREQ_INC2
        {

        import wsp_squared_spec_min, ksisp_max, tautheta2, TIME_END;
        double t1, t2, t3, t4, DC, bottom_ref, wsp_min, n_over_alpha;
        double Ymagn_4(), Ytr_4();

        n_over_alpha = Ytr_4("2",TIME_END)/Ytr_4("3",TIME_END);   150
        DC = Ymagn_4("1",0e0);
        wsp_min = sqrt(wsp_squared_spec_min * n_over_alpha);

        t1 = freq*tautheta2;
        t2 = freq/wsp_min;
        t3 = 1-t2*t2;
```

```
        t4 = 2*ksisp_max*freq/wsp_min;
        DC = Ymagn_4("1",0e0);

        return (DC*sqrt((1+t1*t1)/(t3*t3+t4*t4)));                    160
        }
    >=
        good_curve = {  return -200.0; }
        bad_curve = { return -300.0; }

/*=================================================
        maximum allowable time delay plotted in Freq domain as
        a lower bound on the q/x46 phase
  =================================================
*/                                                                    170

 functional_constraint 'Phase_min' soft
      for freq from FREQ_START2 to FREQ_END2  by FREQ_INC2

        {

        import wsp_squared_spec_min, ksisp_max , tautheta2, TO_DEG, taueff_max;
        import TIME_END;
        double denom_real, denom_imag, q_phase;
        double wsp_min, n_over_alpha;                                 180
        double Ytr_4();

        n_over_alpha = Ytr_4("2",TIME_END)/Ytr_4("3",TIME_END);
        wsp_min = sqrt(wsp_squared_spec_min * n_over_alpha);

        denom_real = 1-(freq/wsp_min)*(freq/wsp_min);
        denom_imag = 2*ksisp_max*freq/wsp_min;
        q_phase = TO_DEG * (atan2(freq*tautheta2,1.0) -
                            atan2(denom_imag,denom_real) -
                            taueff_max*freq);                         190
        return q_phase;
        }
        >=
        good_curve = { return -270.0; }
        bad_curve =  { return -300.0; }


 functional_constraint 'Q_magn' soft
      for freq from FREQ_START2 to FREQ_END2  by FREQ_INC2
                                                                      200
        {
        double Ymagn_4();

        return Ymagn_4("1",freq);
        }
        >=
      good_curve = { return -200.0; }
        bad_curve  = { return -300.0; }
```

(eq_sys_4)

```
                                                                    210
functional_constraint 'Q_phase' soft
    for freq from FREQ_START2 to FREQ_END2  by FREQ_INC2

      {
      import TO_DEG;
      double Yphas_4();

      return (TO_DEG * Yphas_4("1",freq));
      }
      >=                                                            220
      good_curve = { return -200.0; }
      bad_curve  = { return -300.0; }
```

Appendix B

Comparison of the Design for the X-29 using CONSOLE to that of Grumman

The following table is a key for the figures in this appendix. All time domain signals are a result of the step command input.

Page No.	Figure label	Description
56	Cnrd Defl	Canard deflection
	Cnrd rate	The deflection rate of the canard
	Flap defl	Flaperon deflection
	Flap rate	The deflection rate of the flaperon
57	Total wb	Wing bending — the deflection of the wing tips
	wb1 rate	The rate of the wing bending
	Q(A/C)	Pitch rate of the aircraft
	Alpha(A/C)	Angle of attack of the aircraft
58	Q_dot	Pitch acceleration of the aircraft
	Nz	Vertical acceleration of the aircraft
	Nz error	Error between the aircraft vertical acceleration and that of a given model
	Qdot error	Error between the aircraft pitch acceleration and that of a given model
59	Freq Resp	Open loop frequency response of the canard channel with the loop broken at point (1) in Figure 2.1
60	Freq Resp	Open loop frequency response of the flaperon channel with the loop broken at point (2) in Figure 2.1

Table B.1: Description of figures in this appendix.

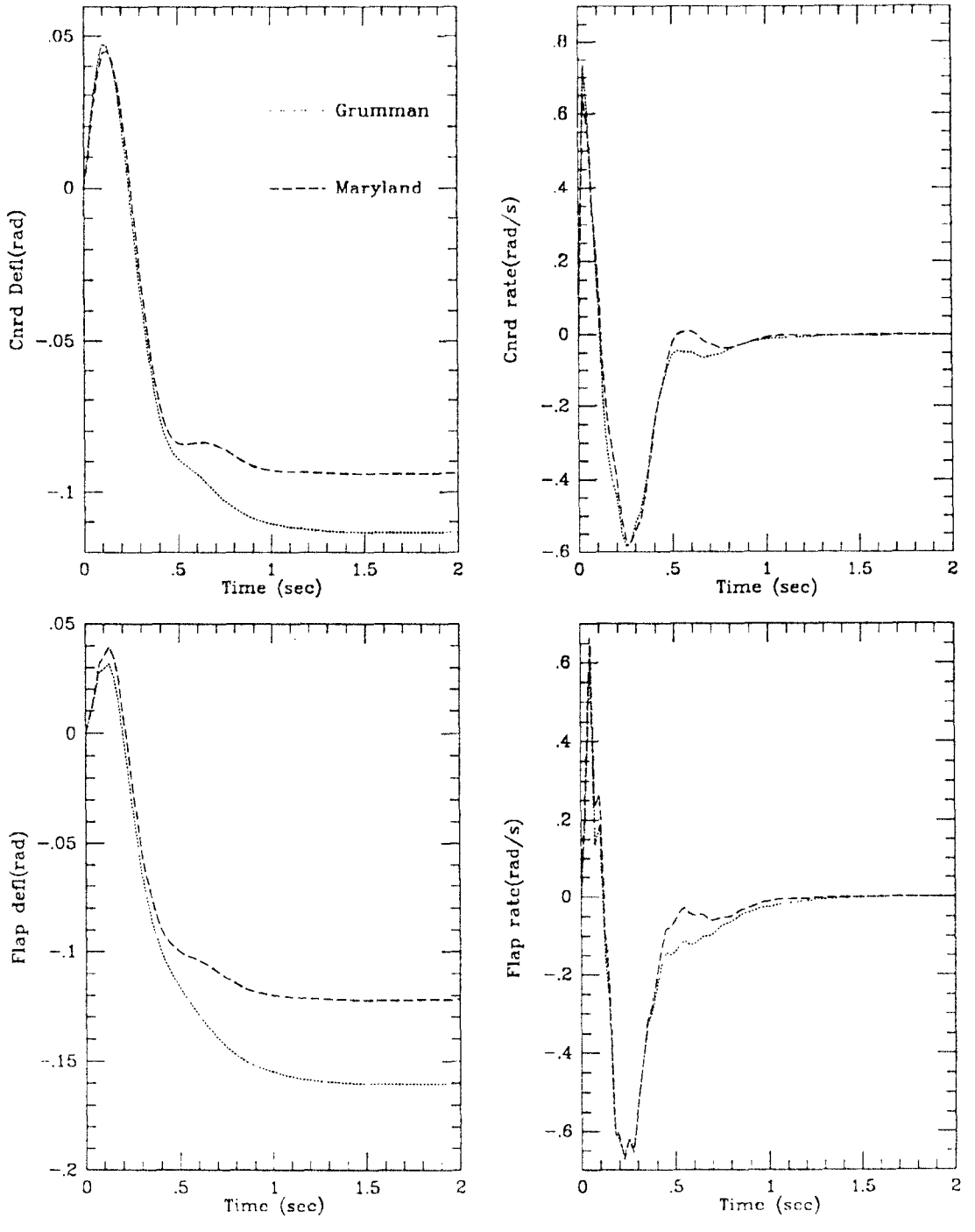


Figure B.1: Control surface responses.

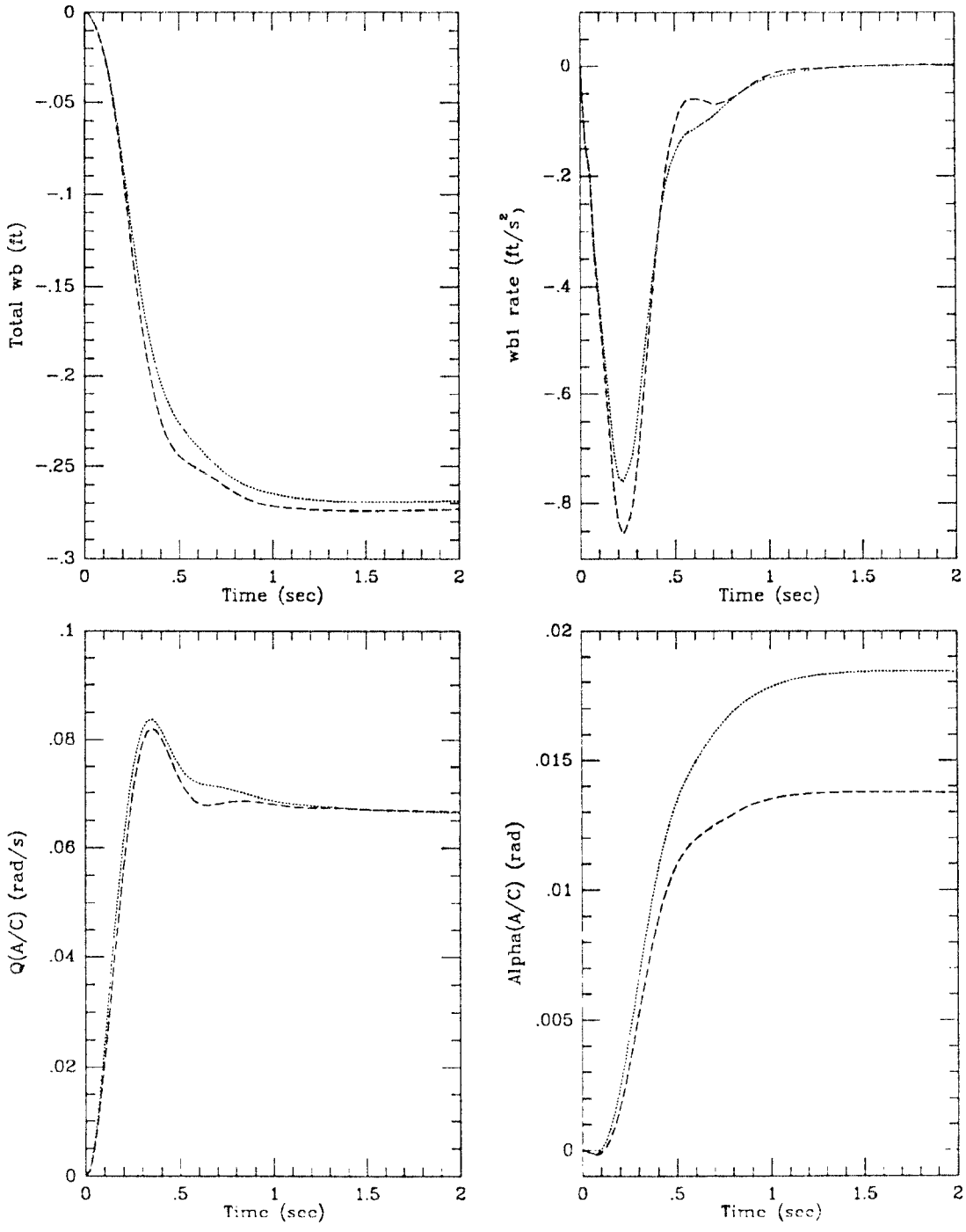


Figure B.2: Various time domain responses to the step command.

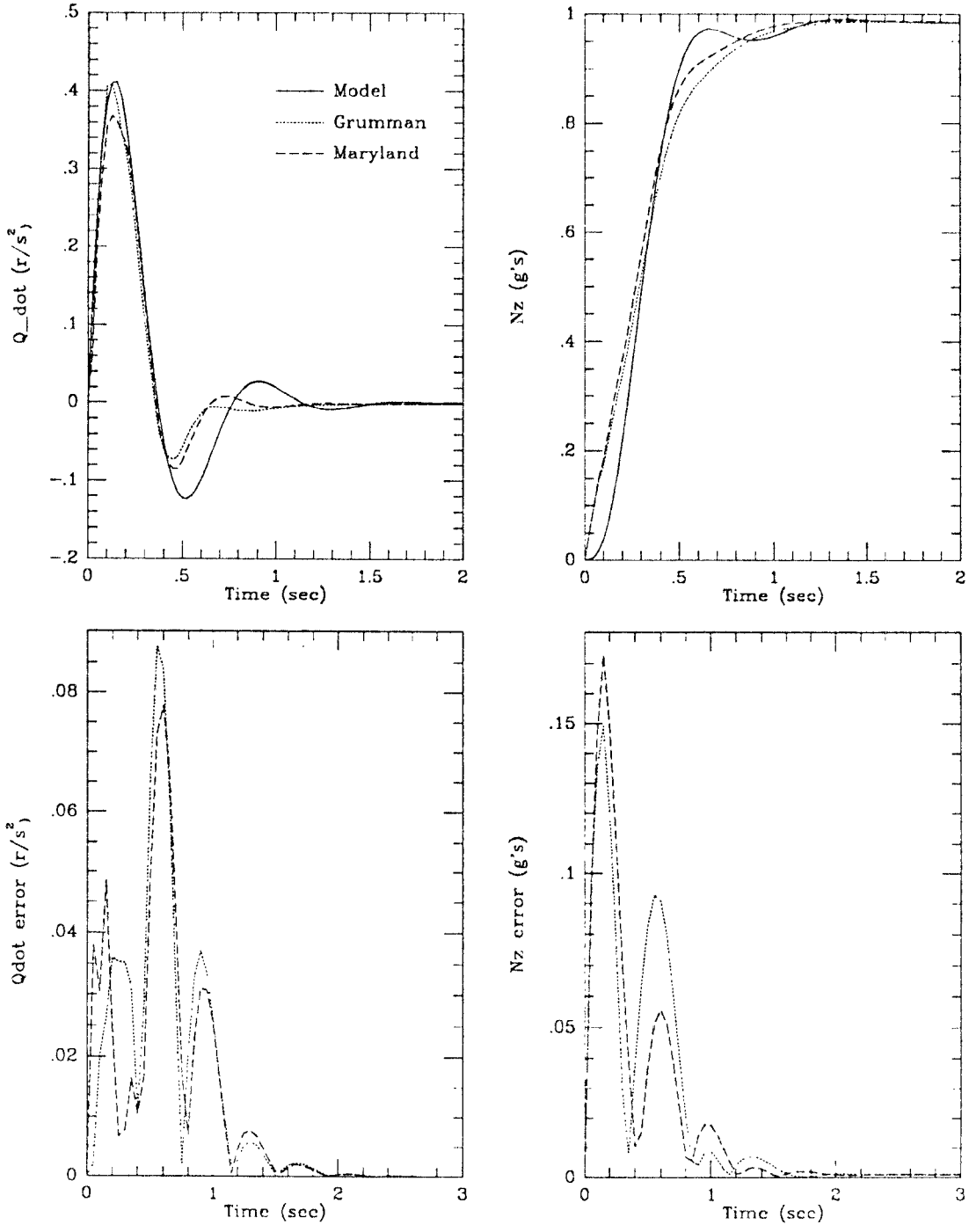


Figure B.3: Acceleration signals of the aircraft due to the step command.

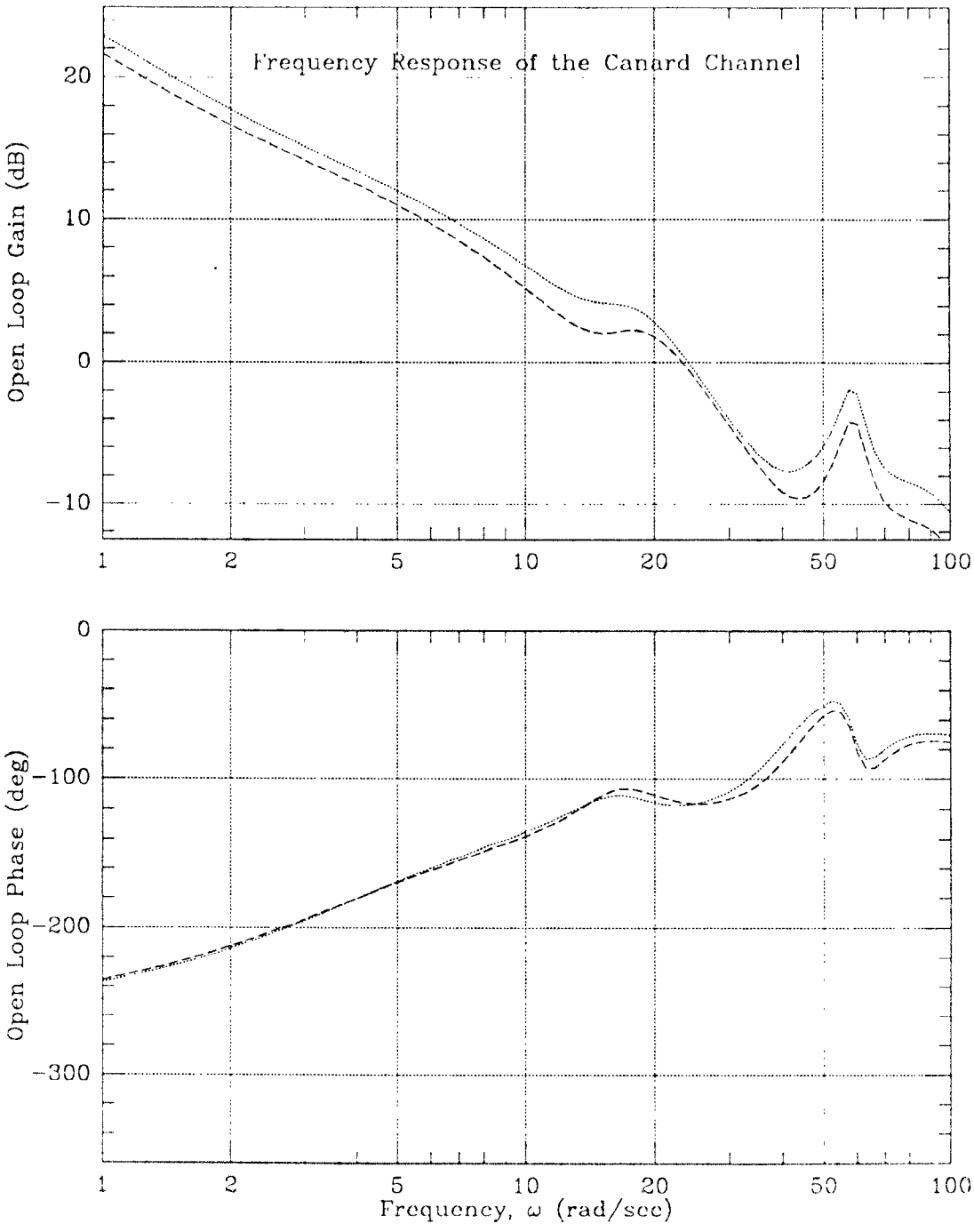


Figure B.4: Open loop frequency response of the canard channel.

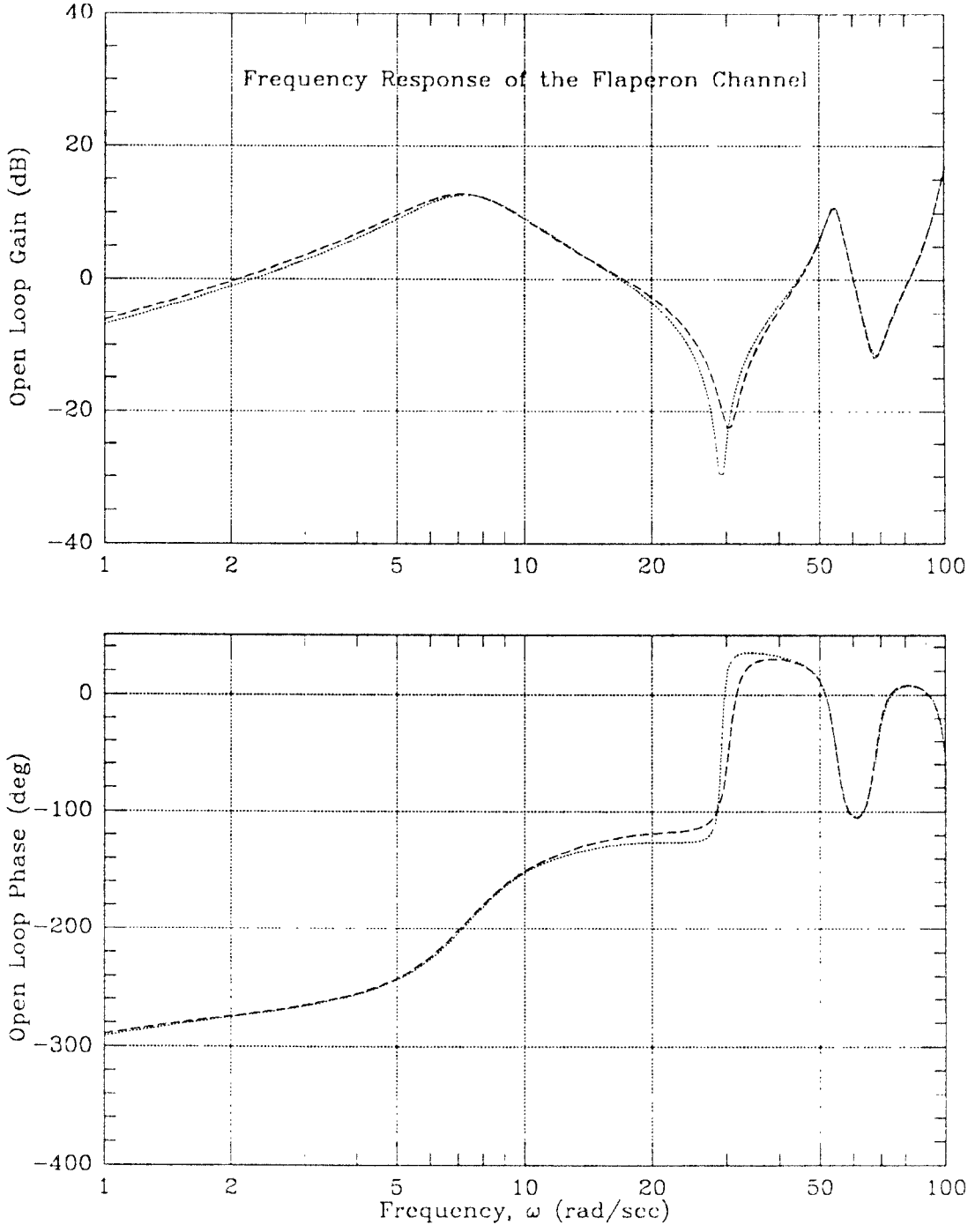


Figure B.5: Open loop frequency response of the flaperon channel.

Appendix C

Design values for the LQG/LTR design for the F14

Here are the values for the completed LQG/LTR design along with the state-space model:

$$A = \begin{bmatrix} -2.000e{+}01 & 0.000e{+}00 & 0.000e{+}00 \\ -6.400e{+}01 & -6.385e{-}01 & 6.894e{+}02 \\ -6.885e{+}00 & -5.920e{-}03 & -6.571e{-}01 \end{bmatrix} \quad B = \begin{bmatrix} 2.000e{+}01 \\ 0.000e{+}00 \\ 0.000e{+}00 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.000e{+}00 & 1.451e{-}03 & 0.000e{+}00 \end{bmatrix}$$

$$A_m = \begin{bmatrix} 0.000e{+}00 & 6.200e{+}00 & 0.000e{+}00 \\ -1.000e{+}00 & -3.521e{+}00 & 1.000e{+}00 \\ 0.000e{+}00 & 0.000e{+}00 & -1.000e{-}05 \end{bmatrix}$$

$$C_m = \begin{bmatrix} 1.000e{+}00 & 0.000e{+}00 & 0.000e{+}00 \end{bmatrix}$$

$$A_v = \begin{bmatrix} 0.000e{+}00 & 1.000e{+}00 & 0.000e{+}00 \\ -1.556e{-}01 & -7.890e{-}01 & 0.000e{+}00 \\ 1.225e{-}02 & 5.377e{-}02 & -8.455e{+}00 \end{bmatrix}$$

$$B_v = \begin{bmatrix} 0.000e{+}00 & 0.000e{+}00 & 0.000e{+}00 \\ 6.385e{-}01 & 2.803e{+}00 & 0.000e{+}00 \\ 1.397e{-}02 & 6.132e{-}02 & -5.556e{+}00 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 3.745e{-}02 & -1.027e{-}04 & -4.472e{-}05 & -2.801e{-}03 & 3.318e{-}03 & 1.398e{-}06 \\ -1.027e{-}04 & 2.922e{-}07 & 1.245e{-}07 & 7.874e{-}06 & -9.350e{-}06 & -5.014e{-}09 \\ -4.472e{-}05 & 1.245e{-}07 & 5.372e{-}08 & 3.378e{-}06 & -4.005e{-}06 & -1.880e{-}09 \\ -2.801e{-}03 & 7.874e{-}06 & 3.378e{-}06 & 2.129e{-}04 & -2.526e{-}04 & -1.259e{-}07 \\ 3.318e{-}03 & -9.350e{-}06 & -4.005e{-}06 & -2.526e{-}04 & 2.997e{-}04 & 1.519e{-}07 \\ 1.398e{-}06 & -5.014e{-}09 & -1.880e{-}09 & -1.259e{-}07 & 1.519e{-}07 & 1.888e{-}10 \end{bmatrix}$$

$$Q_2 = 4.668e{-}01 \qquad Q_r = 9.112e{+}00$$

Appendix D

Comparison between LQG/LTR design for the F14 and that of Grumman and Delight

The following table is a key for the figures in this appendix. All time domain signals are in response to the step command input.

Page No.	Figure label	Description
63	Alpha	Angle of attack of the aircraft superimposed on the desired (i.e. model) angle of attack
	Alpha error	Error between the actual aircraft angle of attack and that of the model
64	N_z CG	Vertical acceleration at the center of gravity of the aircraft
	N_z Pilot	Vertical acceleration at the pilot station of the aircraft
	Pitch rate	Pitch rate of the aircraft
	Tail rate	Tail rate of the aircraft
65	Freq Resp	Open loop frequency response of the aircraft with various controller designs

Table D.1: Description of figures in appendix.

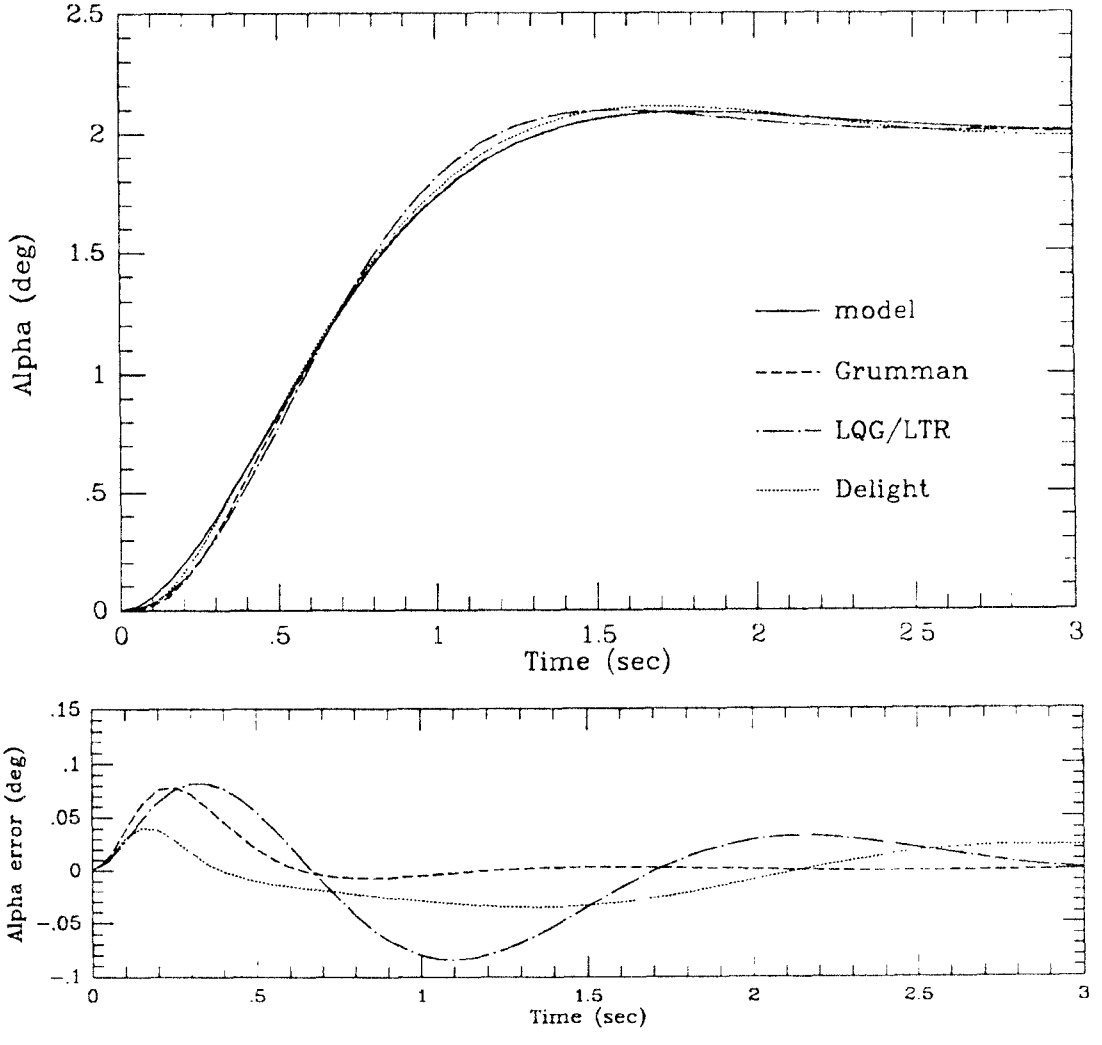


Figure D.1: Angle of attack data.

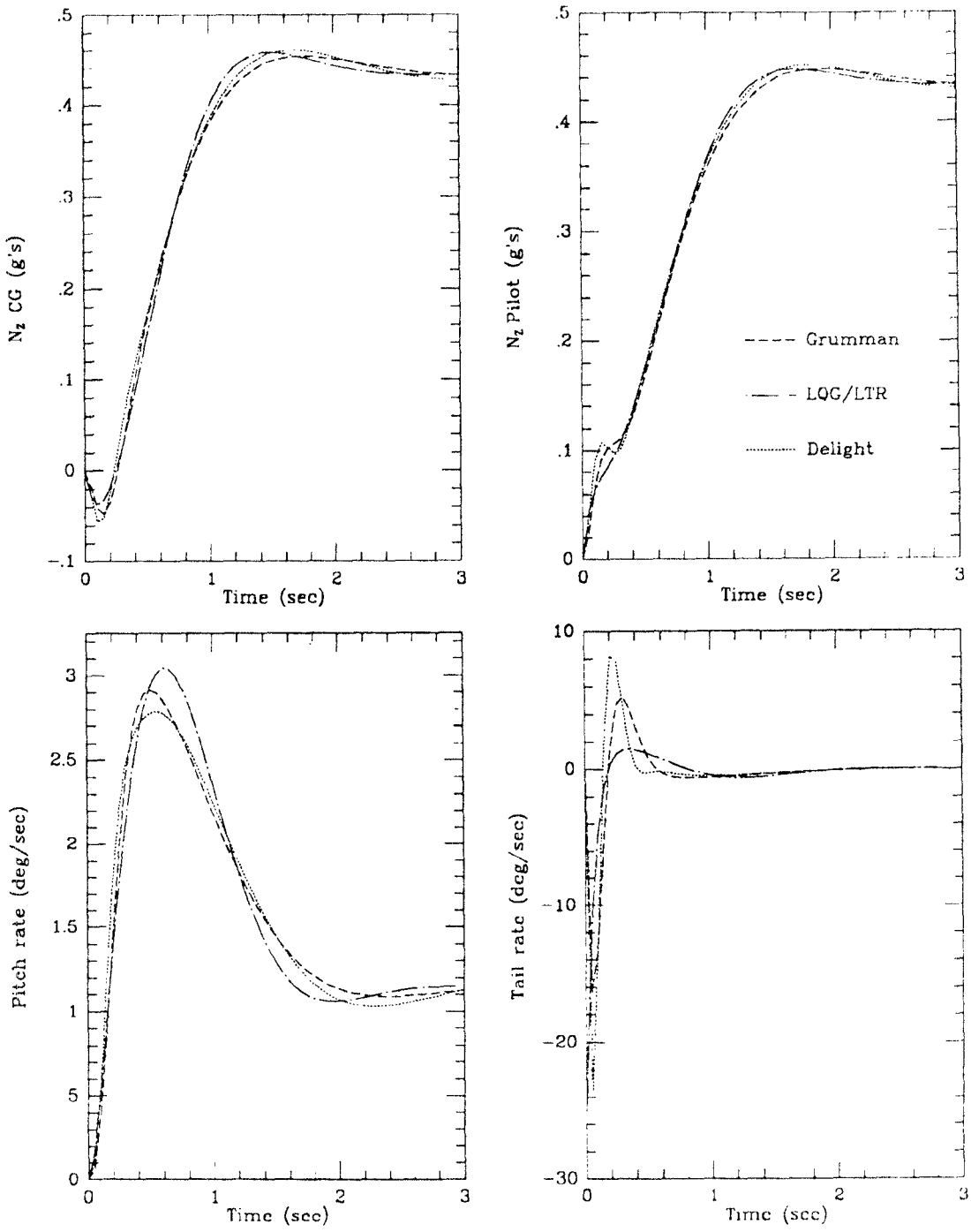


Figure D.2: Various time domain responses to the step command.

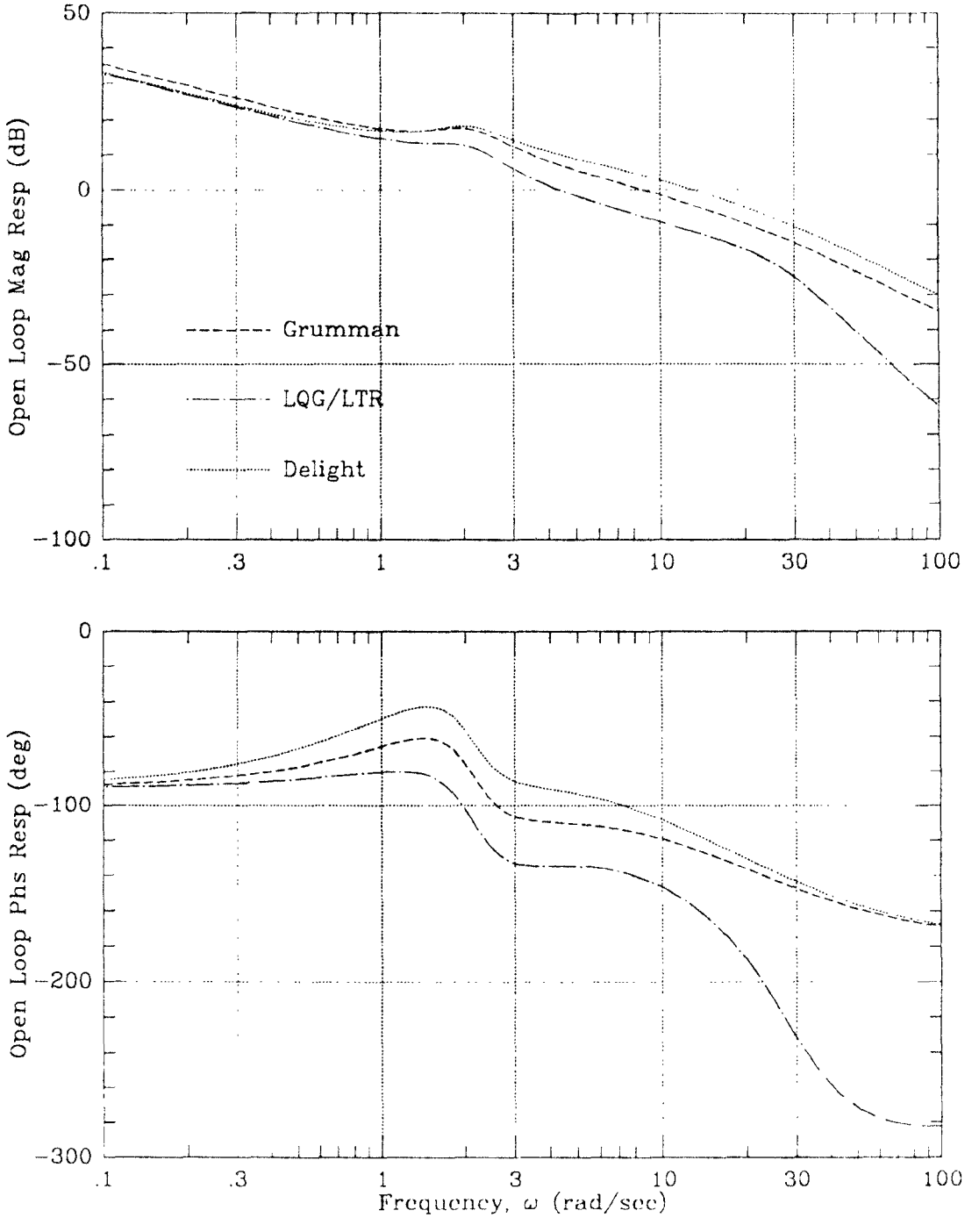


Figure D.3: Open loop frequency response of the aircraft with various controller designs.

Bibliography

[1] M. K. H. Fan, L. Wang, J. Koninckx, and A. Tits, "Software package for optimization-based design with user-supplied simulators," *IEEE Control Systems Magazine*, vol. 9, pp. 66–71, January 1989.

[2] M. Rimer, R. Chipman, and R. Mercadante, "Divergence/flutter suppression system for a forward swept-wing configuration with wing-mounted stores," *Journal of Aircraft*, vol. 21, pp. 631–638, August 1984.

[3] M. Rimer, R. Chipman, and B. Muniz, "Control of a forward swept-wing configuration dominated by flight dynamic/aeroelastic interactions," *Journal of Guidance, Control and Dynamics*, vol. 9, pp. 72–79, Jan.-Feb. 1986.

[4] R. Chipman, F. Rauch, M. Rimer, and B. Muniz, "Transonic test of a forward swept wing configuration exhibiting body freedom flutter," AIAA Paper 85-0689-CP, April 1985.

[5] D. McRuer, I. Ashkenas, and D. Graham, *Aircraft Dynamics and Automatic Control.* Princeton, New Jersey: Princeton University Press, 1973.

[6] "Military specification flying qualities of piloted airplanes," Military Spec MIL-F-8785C, November 1980.

[7] M. Stoughton, "Design refinements of an active divergence/flutter suppression (adfs) system for a 1/2-scale fsw model," Grumman Memorandum Number EG-LD-IOM-85-026, March 1985.

[8] D. E. Bischoff, "The definition of short-period flying qualities characteristics via equivalent systems," *Journal of Aircraft*, vol. 20, pp. 494–499, June 1983.

[9] B. D. O. Anderson and J. B. Moore, *Linear Optimal Control.* Englewood Cliffs, N.J.: Prentice-Hall, 1971.

[10] J. C. Doyle and G. Stein, "Multivariable feedback design: concepts for a classical/modern synthesis," *IEEE trans. Automat. Contr.*, vol. AC-26, pp. 4–16, Feb. 1981.

[11] M. Rimer and R. Grant, "Cascade/delight package for enhanced baseline flight control system design example," Note from Grumman, April 1984.

[12] C. D. Walrath, *Multivariable Optimization-Based Computer-Aided Design of Real Control Systems.* Master's thesis, University of Maryland, August 1984.