THESIS REPORT

Master's Degree

Parametrically Optimal Control for the UH-60A (Black Hawk) Rotorcraft in Forward Flight

by P.J. Potter
Advisor: W.S. Levine

M.S. 95 -8

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Abstract

Title of Thesis: Parametrically Optimal Control for the UH-60A (Black Hawk) Rotorcraft in Forward Flight

Name of degree candidate: Patrick J. Potter

Degree and year: Master of Science, 1995

Thesis directed by: Professor William S. Levine
Department of Electrical Engineering

New techniques for the design of rotorcraft flight control systems (FCS) to meet hover flight condition handling qualities requirements (HQR) have recently been developed. These techniques are based on multicriterion optimization as implemented in the optimization package CONSOL-OPTCAD (C-O). Further development of these methodologies applicable to the design of a FCS to meet forward flight HQR is presented herein. The forward flight design methodology is applied to the design of a FCS for the Advanced Digital Optical Control System (ADOCS) UH-60A Black Hawk helicopter in forward flight at 80 knots. The controller parameters have been optimized to meet the ADS-33C forward flight specifications. Utilizing this approach an optimal controller based on minimum actuator control energy has been obtained and trade-off studies have been performed.

Parametrically Optimal Control for the UH-60A (Black Hawk) Rotorcraft in Forward Flight

by

Patrick J. Potter

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
1995

Advisory Committee:

Professor William S. Levine, Chairman/Advisor
Professor G. L. Blankenship
Professor W. P. Dayawansa

Dedication

To Michele. Without her understanding, love and constant support this would not have been possible.

Acknowledgements

Thanks to Professor Bill Levine, who responded favorably to my initial queries concerning application oriented research. It has been a pleasure and a great learning experience working with him and Dr. Mark Tischler of NASA-Ames Research Center on the Forward Flight ADOCS UH-60A project. Special thanks to Chujen Lin who provided significant insight and help throughout. Thanks also to Gil Yudilevitch, in Haifa, Isreal, for fielding questions and providing answers concerning his work on the Hover ADOCS UH-60A project. Finally, thanks to Professor Roberto Celi and his PhD. student Steve Turnour for providing the helicopter model and answering many related questions.

Table of Contents

List of Tables

List of Figures

Parametrically Optimal Control for the UH-60A
(Black Hawk) Rotorcraft in Forward Flight

Patrick J. Potter

April 26, 1995

This comment page is not part of the dissertation.

Typeset by LaTeX using the `dissertation` style by Pablo A. Straub, University of Maryland.

Chapter 1

Introduction

Multicriterion parametric optimization has been demonstrated as a new direct design technique for meeting rotorcraft hover handling qualities requirements [9]. The extension of this direct design technique to meet forward flight handling qualities is presented here.

Using the established hover technique as a benchmark, modifications were made to the existing simulation routines, thereby enhancing both the speed and accuracy of the optimization-based design. Clarifications to the methodology utilized in the hover problem setup are also presented. Using the established technique as a guide, modifications were made to facilitate forward flight controller design. The technique ensures that the controller-rotorcraft system meets the ADS-33C Level 1 [8] forward flight handling qualities requirements. Simply meeting the forward flight handling qualities requirements, that is, achieving a feasible solution, provides a stepping stone for achieving both optimal and alternative solutions. Optimal solutions, with respect to actuator energy cost functions, are obtained using the modified technique. Alternative solutions, resulting from implementation of different helicopter actuator hardware or from more stringent performance requirements, are also achieved.

The main objectives of this research are:

- Further development of the multicriterion optimization-based hover rotorcraft FCS design technique to facilitate design of a FCS for rotorcraft in forward flight.

- Demonstration of this technique by finding a set of FCS parameters such that the ADOCS UH-60A Black Hawk helicopter at 80 knots forward flight meets the ADS-33C Level 1 performance requirements [8].

- Further demonstration of this method by meeting the Level 1 performance requirements with minimum actuator energy ("optimal solution").

- Demonstration of the use of this method with the ADS-33C requirements to perform trade-off studies.

- Identification of ADS-33C requirements not directly implementable as CONSOL-OPTCAD constraints and interpretation of the spirit of the requirement to facilitate implementation.

The hover flight problem setup and solution were completed by Professor William Levine and Dr. Gil Yudilevitch, under the direction of Dr. Mark Tischler of the NASA Ames Research Center (ARC), prior to this research into the extension of the techniques to a forward flight problem. Their work, documented in [9] and [10], provides significant insight into the complexities involved in posing and solving the hover flight problem. In order to successfully pose and solve the forward flight problem, a better understanding of the hover problem was sought. Implementing the hover problem setup and solution in the model-based SIMULINK environment provided this necessary understanding. Additionally,

significant insight, simulation efficiency and comparably accurate results were achieved. Translation of the model-based parametric optimization technique to the forward flight problem follows from the understanding of the hover problem.

As with the hover problem, the forward flight handling qualities requirements which define Level 1 performance are given in ADS-33C Paragraph 3.4 - Forward Flight, Instrument Meterological Condition (IMC) and/or Divided Attention Mission-Task-Element (MTE) requirements, [8]. Definition of a more stringent performance level (Level 1 - 1), to be discussed in depth in Chapter 5, is based on the ADS-33C Aggressive Maneuvering Air Combat MTE requirements.

This document is organized into six chapters and two appendices. Chapter 2 presents a review of the hover problem setup and solution to clarify omissions of and deviations from the ADS-33C hover flight requirements. The more thorough an understanding the control engineer has of the hover problem setup and solution, the better able he or she is to pose and solve subsequent problems. Chapter 3 discusses an alternative setup and solution of the hover problem utilizing SIMULINK which enhances the understanding, accuracy and speed of the design process. Chapter 4 is an in-depth explanation of the technique modifications necessary to correctly setup and solve the eighty knot forward flight problem. Similarities and differences between the hover and forward flight problems are presented and discussed. Chapter 5 presents nominal (LEVEL 1) designs, optimal designs and the results of performance/specification and performance/hardware trade-off studies. Finally, some concluding remarks and suggestions for future application of these techniques are presented in Chapter 6.

Appendices A and B are listings of the computer code used in both design

processes. Appendix A contains the code necessary to run the hover problem utilizing SIMULINK as the simulation engine. Appendix B presents the code for the forward flight problem which also utilizes SIMULINK. These two appendices contain all the code necessary to run either problem and may serve as benchmarks for reference purposes.

Chapter 2

A Review of the Hover Solution

Converting the ADS-33C requirements on a rotorcraft control system into precisely defined mathematical specifications is a complex task. Converting those specifications into forms compatible with CONSOL-OPTCAD (C-O) is also difficult for three reasons:

- It must be possible to compute the specifications.
- They must be as smooth as possible. More precisely, they must be twice continuously differentiable and having more continuously differentiable derivatives is better.
- The computations must be reasonably fast. In order to minimize the computation time inactive constraints and constraints not affected by the control must be excluded.

The setup and solution of the ADOCS Rascal (UH-60A) in hover was the first attempt to formulate a realistic rotorcraft controller design problem as a CONSOL-OPTCAD problem [9]. As part of the setup and solution of the ADOCS Rascal at 80 knots forward flight we performed a critical review of the hover setup and solution.

The purpose of the information presented herein is to clarify aspects of the ADOCS Hover Flight Solution presented in reference [9]. That clarification is desirable was determined in the process of posing and solving the Eighty (80) Knot ADOCS Forward Flight problem. Using the hover flight problem setup and solution as a guide for the forward flight problem was an effective means to clarify both omissions of ADS-33C requirements and deviations from ADS-33C requirements in the setup of the hover flight problem in C-O.

The CONSOL-OPTCAD derived hover solution is defined in terms of five specifications derived from selected ADS-33C Hover and Low Speed Handling Qualities Requirements found in reference [8]. The rationale behind the omission of the ADS-33C Large-Amplitude Pitch and Roll Attitude Changes (ADS-33C - 3.3.4), the Large-Amplitude Heading Changes (ADS-33C - 3.3.8) and the Response to Collective Controller (ADS-33C - 3.3.10) requirements is not addressed in [9]. Also, the rationale behind deviations from the ADS-33C requirements by specifications 2 and 3 is not clarified.

Consider the omission of the Rotor RPM Governing and Torque Response requirements, a subset of the Response to Collective Controller requirement. The linearized and reduced 11 state helicopter model was derived from a full 224 state nonlinear model through model reduction techniques. The engine dynamics were eliminated during the model reduction. Thus, the linearized and reduced model contains no information about the engine dynamics. This precludes monitoring compliance with either of these requirements. Of course, if the engine dynamics were important to the controller design, the reduced model would have been made to include them. However, the engine dynamics do not influence the controller design and the omission of the engine related requirements is justified.

The remaining requirements under the general Response to Collective Controller requirement, Height Response Characteristics and Vertical Axis Control Power, were omitted because they are not functionally related to the ADOCS controller scheme. That is, the control system does not have an effect on the rotorcraft's ability to impart specified forces on a pilot or to achieve certain minimum vertical rates. The final omissions, the large-amplitude heading and attitude changes, also fall into the category that there is nothing the ADOCS control system can do to facilitate the rotorcraft meeting these requirements. Thus, the omitted ADS-33C specifications for hover performance were all those that were unaffected by the ADOCS controller.

Each of the five hover specifications are presented below, in order, with respect to the ADS-33C hover requirements they encompass. As applicable, information presented in [9] is either augmented or clarified.

Specification 1: Small Amplitude Changes, Short Term Response to Control Inputs is presented and calculated per the ADS-33C requirements (3.3.2.1 - Pitch & Roll, 3.3.5.1 - Yaw). That is, the CONSOL-OPTCAD specification directly implements both the letter and spirit of the ADS-33C requirements.

Specification 2: Small Amplitude Changes, Mid-Term Response to Control Inputs addresses the ADS-33C requirements pertaining to the mid-term response to control inputs (3.3.2.2 - Pitch & Roll, 3.3.5.2 - Yaw). The requirement is written in terms of an effective damping ratio while the CONSOL-OPTCAD specification is defined in terms of classical SISO stability margin criteria, that is, a gain margin $(GM) \geq 6$ db and a phase margin $(PM) \geq 45°$. Reference [9] indicates the reasons for not using the damping ratio parameter, ζ, or approximations for ζ based on model reduction techniques, system identification and

2^{nd} order approximations. The stability margin criteria were used in place of the damping ratio requirements because it was determined that the pole placement requirement does not accurately define the desired mid-term response and that the stability margin criteria are more appropriate. The pole placement requirement, as written in [8], does not accurately define the desired mid-term response because it allows undesirable, long term phugoid modes and may not prevent Pilot Induced Oscillations (PIO) [9]. A C-O hard constraint requiring asymptotic stability (i.e., $\Re\{\lambda_m(A - BH)\} < 0$) eliminates the possibility of any unstable modes. This enables the use of the GM/PM criteria, which apply to stable systems, in the specification. A significant digression into the relationship between gain margin, phase margin and control system stability is not mandated here. Of note though is the relationship between phase margin and damping ratio, from reference [2], as it applies to a second order system.

$$\zeta \cong \frac{PM}{100} \tag{2.1}$$

This relationship, coupled with the requirement on the phase margin ($\geq 45°$) in specification 2, indicates that an equivalent second order channel model meeting the requirement is stable with an effective damping ratio of at least 0.45. This exceeds the ADS-33C Mid-Term Response to Control Inputs damping ratio requirement by 28% for each of the pitch, roll and yaw channels. Thus a system meeting the stability margin requirements, as in specification 2, can be considered in compliance with the damping ratio requirements in ADS-33C 3.3.2.2 and 3.3.5.2.

Specification 3: Moderate-Amplitude Attitude Changes (Attitude Quickness) is presented in reference [9] per the ADS-33C requirements, but the actual calculation of the quickness ratio is not as specified. Consider just the yaw channel

because the quickness computations in the pitch and roll channels are similar. The ADS-33C requirement states that the ratio of peak yaw rate to change in heading, $r_{pk}/\triangle\psi_{pk}$, shall exceed specified limits. See reference [8] for details. The actual calculation of that ratio was done with the yaw channel command input, δ_ψ, as the denominator as opposed to $\triangle\psi_{pk}$. This was done because the $r_{pk}/\triangle\psi_{pk}$ calculation is essentially a max/max measurement, which C-O then tries to maximize. This calculation violates the smoothness requirement and causes the optimization to stick. Thus, in order to allow the optimization to continue, a smooth calculation, r_{pk}/δ_ψ, was used. An objective of the research is to identify ADS-33C requirements which are not smooth and therefore not directly implementable as specifications. The quickness requirement is one such requirement. Another objective is a feasible controller solution. Achievement of this objective is allowed through use of the smooth quickness measurement r_{pk}/δ_ψ.

Consequences of the use of δ_ψ instead of $\triangle\psi_{pk}$ in the quickness measurement can be seen in Figure (2.1). Note that the command input δ_ψ is smaller than $\triangle\psi_{pk}$ which, when used in the quickness measurement denominator, results in an optimistic measurement. The quantitative difference in the yaw channel quickness can be seen in Figures (2.2) and (2.3) (2^{nd} row, 1^{st} column). These plots are based on the flight tested design parameters from Dr. Tischler's AIAA paper [7]. Use of the r_{pk}/δ_ψ measurement, as in Figure (2.2), allows an optimistic yaw quickness which meets the specification while use of the prescribed quickness measurement $r_{pk}/\triangle\psi_{pk}$, as in Figure (2.3), provides a measurement out of specification. This out of specification measurement is consequently addressed by C-O in attempts to improve the measurement. These attempts fail in light

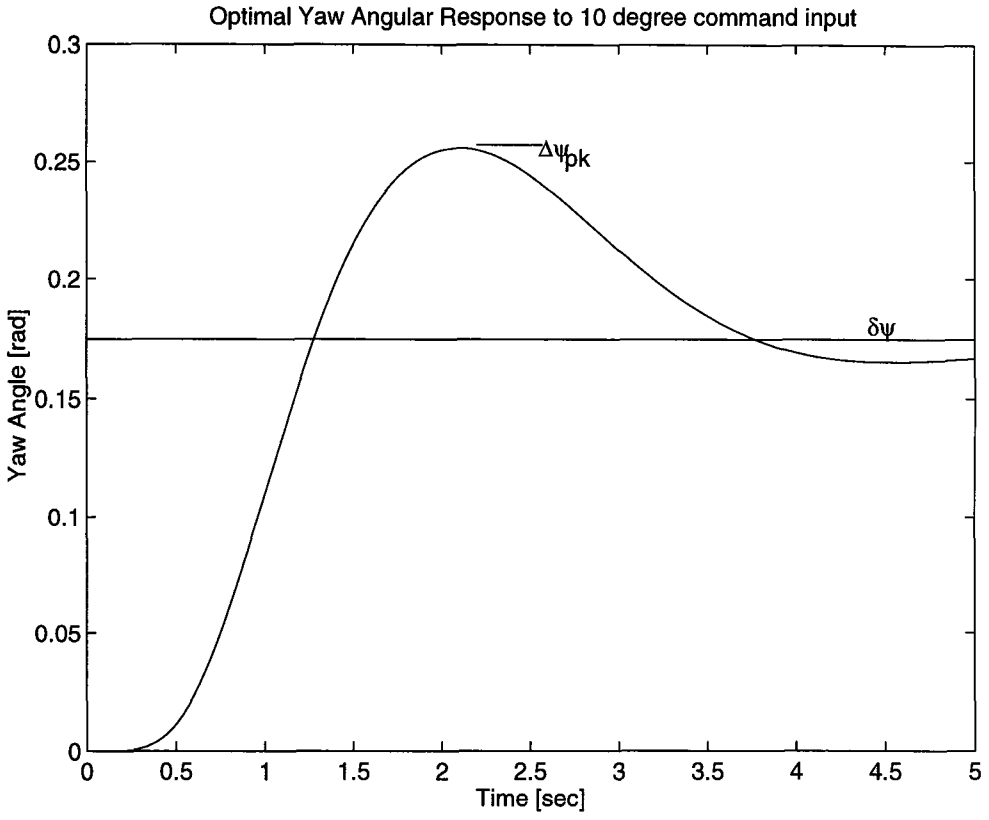

Figure 2.1: Yaw angular response, $\triangle\psi$, to pulse command input. $\triangle\psi_{pk}$ represents peak angular response. δ_ψ represents command input.

of the non-smoothness of the constraint. As a result, a feasible controller design is not achievable.

Consider also that the optimization objective, minimum actuator energy, has been observed to be directly related to both the channel quickness (Spec 3) and the channel bandwidth & phase delay (Spec 1). The connection between the channel quickness and actuator energy is that the quickness approaches the Level 1 - Level 2 boundary as the actuator energy is minimized. Figure (2.4), reproduced from reference [10], shows an optimal controller achieved using the optimistic quickness measurements for pitch - q_{pk}/δ_θ, roll - p_{pk}/δ_ϕ and yaw -

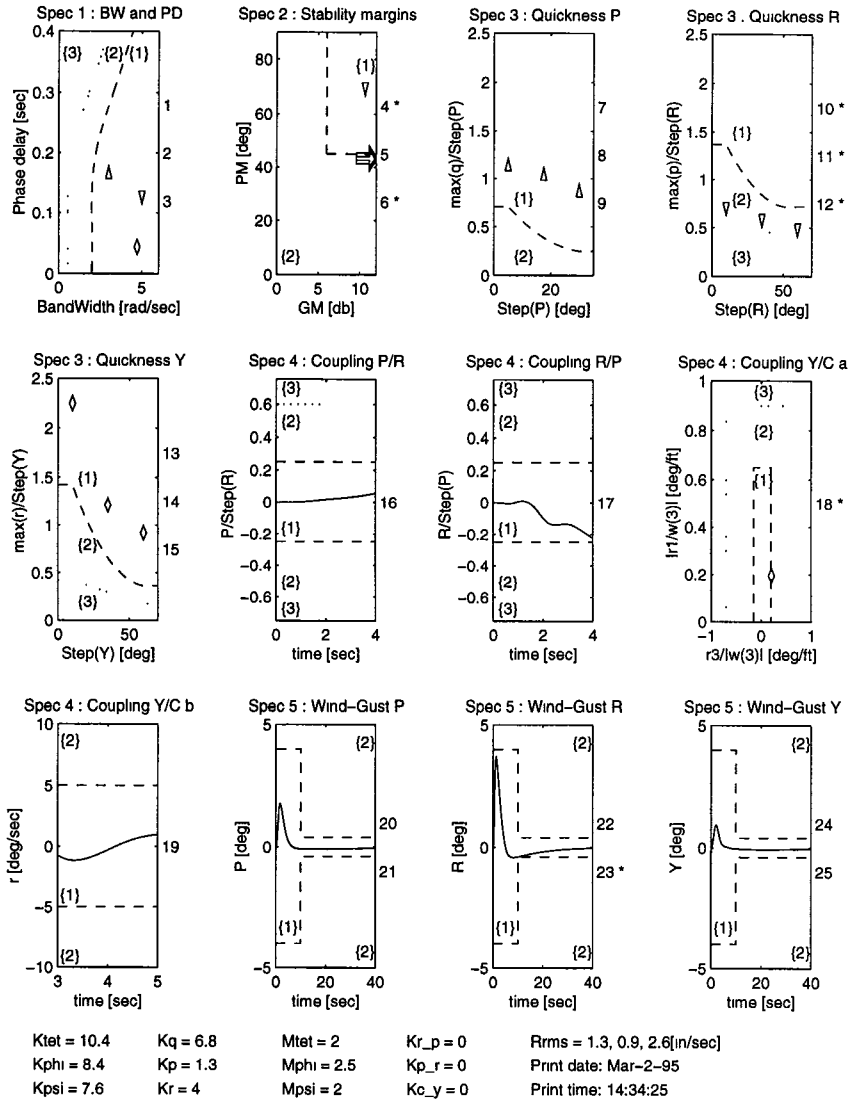

Figure 2.2: Initial design with quickness ratio calculation $\frac{r_{pk}}{\delta_\psi}$. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

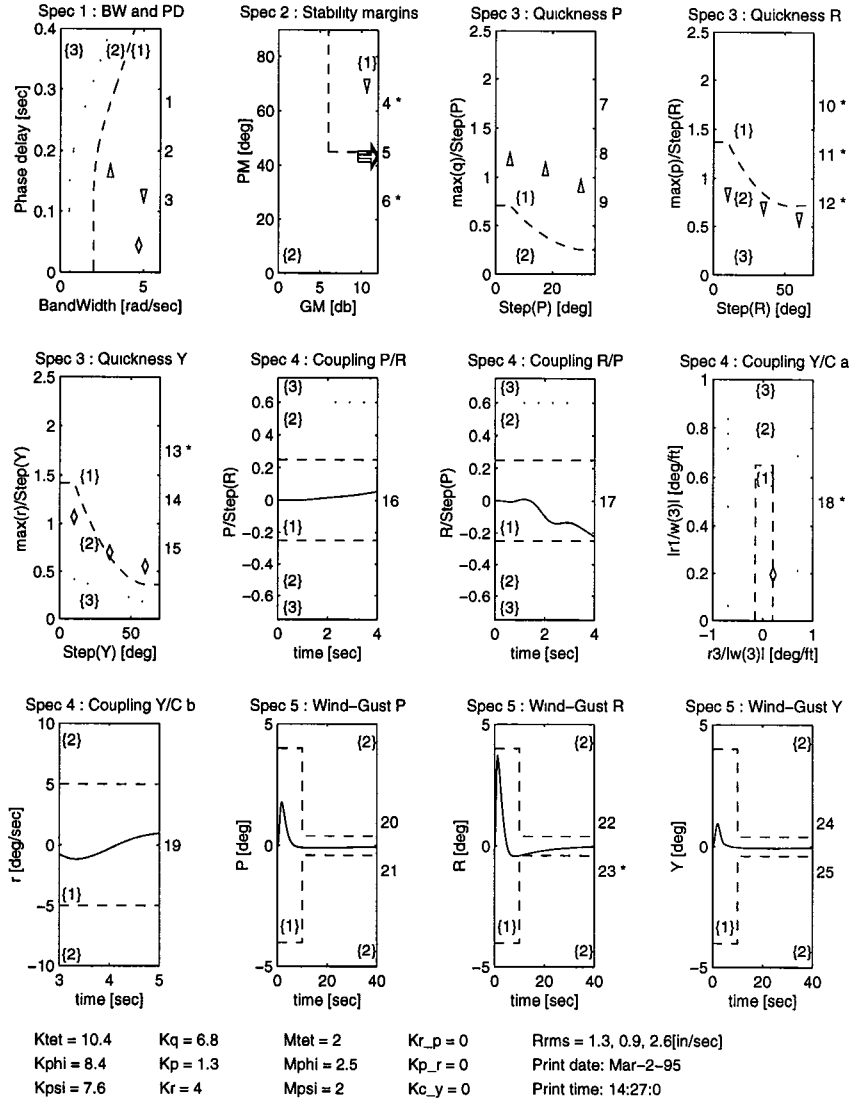

Figure 2.3: Initial design with quickness ratio calculation $\frac{r_{pk}}{\triangle\psi_{pk}}$, per ADS-33C. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

r_{pk}/δ_ψ. Notice that the pitch, roll and yaw quickness measures all have at least one component on the Level 1 - Level 2 boundary. This indicates that with the proper (per ADS-33C) quickness measurement, the optimal controller may not meet the quickness requirements. This is definitely true in the yaw channel due to the overshoot in the yaw response. Again, see Figure (2.1). The pitch and roll channels are not oscillatory and their compliance to the quickness requirement based on the original measurement is subject to interpretation and will be further discussed below. But, because the yaw angular response is oscillatory, i.e. $\triangle\psi_{pk} \neq \delta_\psi$ and in fact $\triangle\psi_{pk} > \delta_\psi$, the optimal hover controller achieved in references [9] and [10] through use of the optimistic measurement does not actually meet the ADS-33C yaw quickness requirements.

It has also been observed that the optimization has trouble increasing the yaw channel quickness independent of which quickness measurement is used. This is better understood by considering the effect that saturating the yaw channel actuator has on the channel bandwidth, which is effectively the channel quickness. Figure (2.5) shows the yaw channel actuator dynamics states for small, medium and large command inputs in the first three subplots and shows the command inputs in the fourth subplot. Notice that in each case, the actuator saturates. This saturation occurs because the command and or rate limits, either upper, lower or both, are exceeded by all three inputs. As a result of saturation, the effective loop gain is reduced. This results in an effectively lower channel bandwidth which explains the inability to meet the quickness requirement.

It is important to understand the flavor of what the quickness requirement is trying to capture. Reference [3] indicates that this requirement effectively allows decreasing bandwidth with increasingly large maneuvers. It is also clear

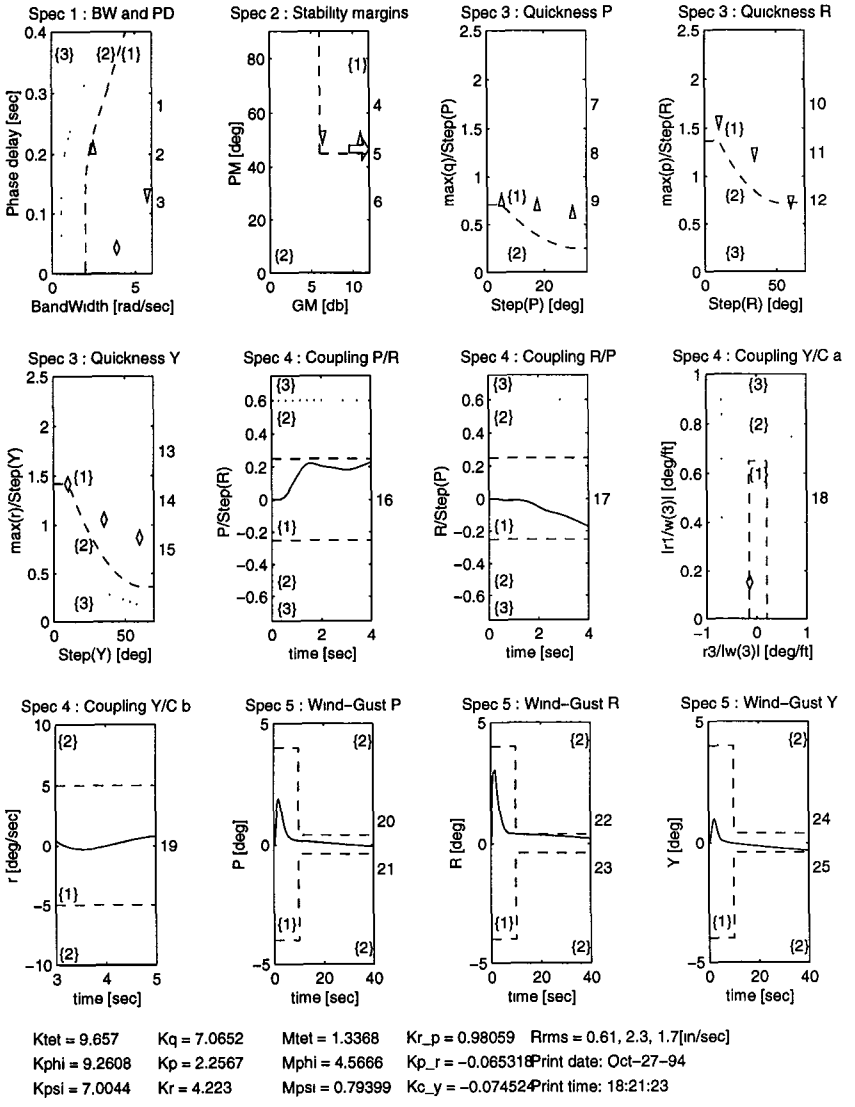

Figure 2.4: The optimal controller achieved using the quickness measurement $\frac{r_{pk}}{\delta_\psi}$. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

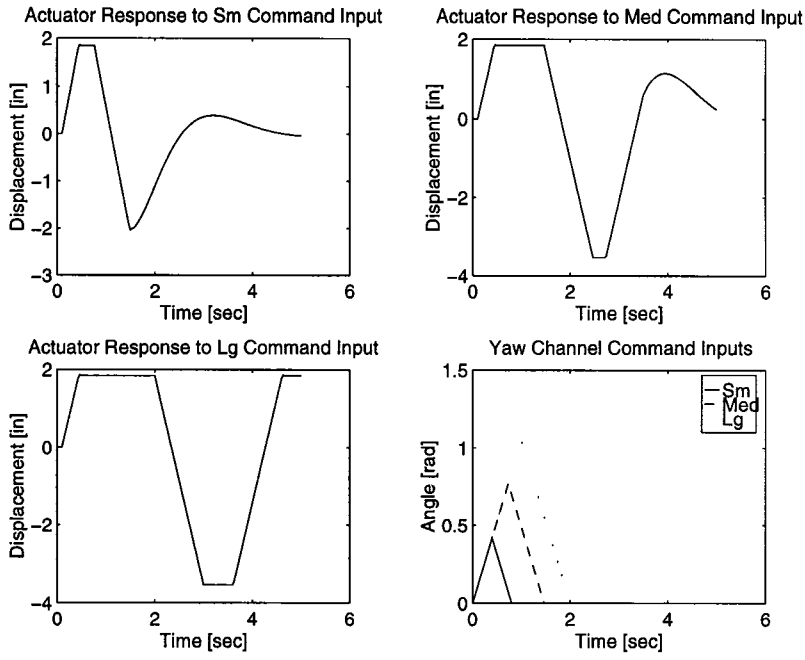

Figure 2.5: Yaw channel actuator state responses to pulse command inputs.

that there is meant to be a penalty for angular response overshoot or low angular rate. The Hover ADOCS controller has attitude command models for the pitch and roll channels and a rate command model for the yaw channel.

The rate command model in the yaw channel has an oscillatory angular response to a quickness requirement command input while the pitch and roll channel responses are not oscillatory. This is represented for the pitch channel at the optimal hover solution in Figure (2.6). The fact that the response does not achieve the commanded value results in more conservative pitch and roll channel quickness measurements when the original measurement is used. Since both the pitch and roll channels have qualitative first order responses and using δ_θ and δ_ϕ as representations for $\triangle\theta_{pk}$ and $\triangle\phi_{pk}$, respectively, produces more conservative quickness measures, it is reasonable to leave those measurements

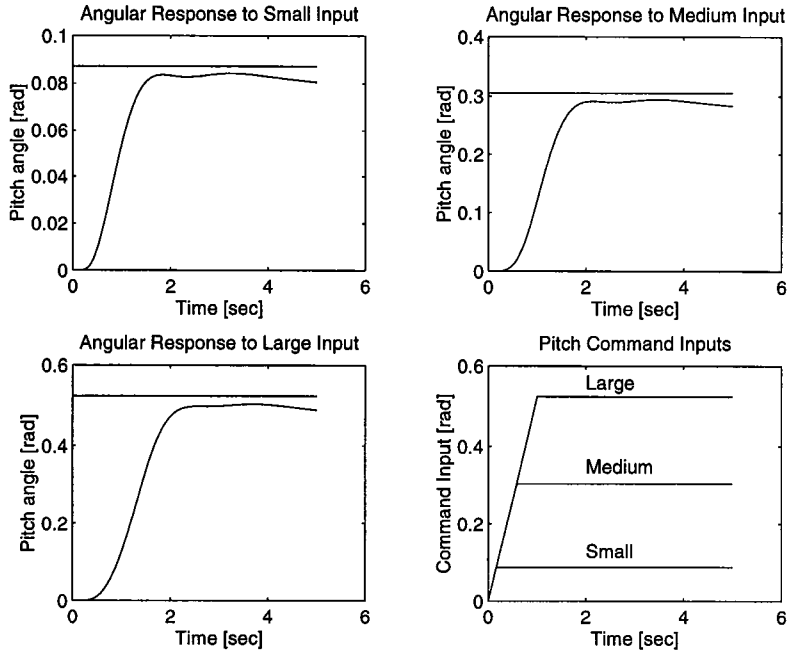

Figure 2.6: Pitch channel angular responses to small, medium and large step command inputs. Step command input co-plotted with response.

as they were. But, without altering the yaw channel actuator structure, the yaw channel quickness requirement will not be satisfied. Other NASA-Ames studies have also determined that the yaw channel cannot meet the quickness requirement because of limitations imposed by the yaw channel actuator[1]. A short term solution to this problem is to make the yaw channel saturation limits less restrictive, allowing the optimization to proceed by facilitating satisfaction of the correct quickness requirement. This was also effectively achieved by using δ_ψ instead of $\Delta\psi_{pk}$ in the quickness measurement and leaving the yaw channel saturation limits as specified.

Specification 4: Decoupling, addresses the pitch-to-roll, roll-to-pitch and yaw due to collective interaxis coupling requirements (3.3.9.1 Yaw, 3.3.9.2 Pitch & Roll). The pitch-to-roll and roll-to-pitch requirements presented in ADS-33C apply to Aggressive Maneuvering Mission-Task-Elements (MTE) such as Air Combat. Recall that Level 1 performance for the hover problem is defined in terms of IMC/Divided Attention MTEs and that Level 1 - 1 is in terms of Aggressive Maneuvering MTEs [9]. Inclusion of the Air Combat pitch-to-roll and roll-to-pitch requirements in the Level 1 hover specification may limit the size of the feasible set, but provides a feasible controller which meets both Level 1 and Level 1 - 1 interaxis coupling requirements. In general, the hover problem was not constrained by interaxis coupling because of the crossfeed design parameters. As a result, inclusion of these requirements in the definition of Level 1 performance minimally affects the solution. This is not the case for the forward flight problem, as discussed in Chapter 4.

[1]Phone conversation between Dr. Mark Tischler, Professor Bill Levine and P. J. Potter, March 1995

```
Mr1 = max(X41(:,16)); Jr1 = find(X41(:,16) == Mr1);  %for w(t) < 0
   if t(Jr1) >= t3, r1 =  X41(t1,16); else, r1 = Mr1; end; %& r(t) > 0
   r3 = X41(t3,16)-r1;
   y_y = -180/pi*r1/X41(t3,13);
   x_y =  180/pi*r3/X41(t3,13);
   a_y = (x_y-0.15)^2; b_y = (y_y-0.65)^2; c_y = (x_y+0.2)^2;
  if     (x_y >= 0.15) & (y_y < 0.65),
     yaw_d2 = a_y;
  elseif (x_y <  -0.2) & (y_y < 0.65),
     yaw_d2 = c_y;
  elseif (x_y >= -0.2) & (x_y < 0.15) & (y_y >= 0.65),
     yaw_d2 = b_y;
  elseif (x_y >= 0.15) & (y_y >= 0.65),
     yaw_d2 = a_y + b_y;
  elseif (x_y <  -0.2) & (y_y >= 0.65),
     yaw_d2 = b_y + c_y;
  else,
     yaw_d2 = 0;
  end;
```

Figure 2.7: Original Spec 4 Calculations.

The yaw due to collective requirement limits both peak yaw rate oscillations due to collective inputs and the yaw rate normalized by vertical rate. Figure (2.7) is the code which implements the Collective to Yaw Coupling Requirements of ADS-33C 3.3.9.1 for Specification 4 from reference [9]. X41(:,16) represents the yaw rate response and X41(:,13) represents the vertical rate response. The collective input command used simulates pulling the collective up to a new setting and imparts a negative vertical rate response (vertical positive down) and a positive yaw rate response on the system.

Based on the known rate responses, the calculations could be simplified as in the code presented in Figure (2.8). That the sign on both the yaw rate and vertical rate responses are known is accounted for in computation of both x_y

```
Mr1 = max(Y41(:,3)); Jr1 = find(Y41(:,3) == Mr1); % for w(t) < 0
   if t(Jr1) >= t3, r1 =  Y41(t1,3); else, r1 = Mr1; end; %& r(t) > 0
     % r1 > 0 for this collective input
     r3 = Y41(t3,3)-r1;
     % altitude rate, X41(:,13), is < 0 for this collective input
     y_y =-180/pi*r1/X41(t3,13);
     x_y =-180/pi*r3/X41(t3,13);
     a_y = (x_y-0.15)^2; b_y = (y_y-0.65)^2; c_y = (x_y+0.2)^2;
     yaw_d2=0;
  if     (y_y >= 0.65),
     yaw_d2 =b_y;
  end;
  if     (x_y <= -0.15),
     yaw_d2=yaw_d2 + a_y;
  elseif (x_y >= 0.20),
     yaw_d2=yaw_d2 + c_y;
  end;
end;
```

Figure 2.8: Updated Spec 4 Calculations.

and y_y. This avoids the use of the non-smooth abs(·) function. The method for checking compliance with the dimensional bounds of the requirement penalizes for both x_y and y_y individually out of spec as well as jointly out of spec. Both pieces of code perform the same calculation, but the code in Figure (2.8) more closely represents the ADS-33C requirement.

Specification 5: Wind-Gust Rejection, as described in [6] and [9], is based on an alternative requirement in light of the fact that use of the ADOCS controller precludes the ADS-33C disturbance rejection requirement (3.3.2.3 Pitch & Roll, 3.3.7.1 Yaw) from being directly related to system bandwidth.

Chapter 3

The Hover Setup and Solution using SIMULINK

The ADOCS Rascal Hover setup and solution were originally implemented entirely with MATLAB m-files. Alternatively they can be posed and solved using SIMULINK, an extension of MATLAB used both to define and analyze block diagram models of systems. For more information on SIMULINK, see reference [4]. Benefits of using SIMULINK to model and analyze the UH-60A with the ADOCS controller structure are fourfold:

- The ADOCS controller structure (feedforward control, delays, actuators, feedback stabilization and helicopter dynamics), shown in Figure (3.1), lends itself to direct implementation using SIMULINK.

- The overall dynamics of the controller are readily apparent, thereby facilitating both easy modification and understanding of the associated MATLAB m-file code used for simulation and calculations.

- Simulation times are greatly reduced, significantly reducing the overall controller design time.

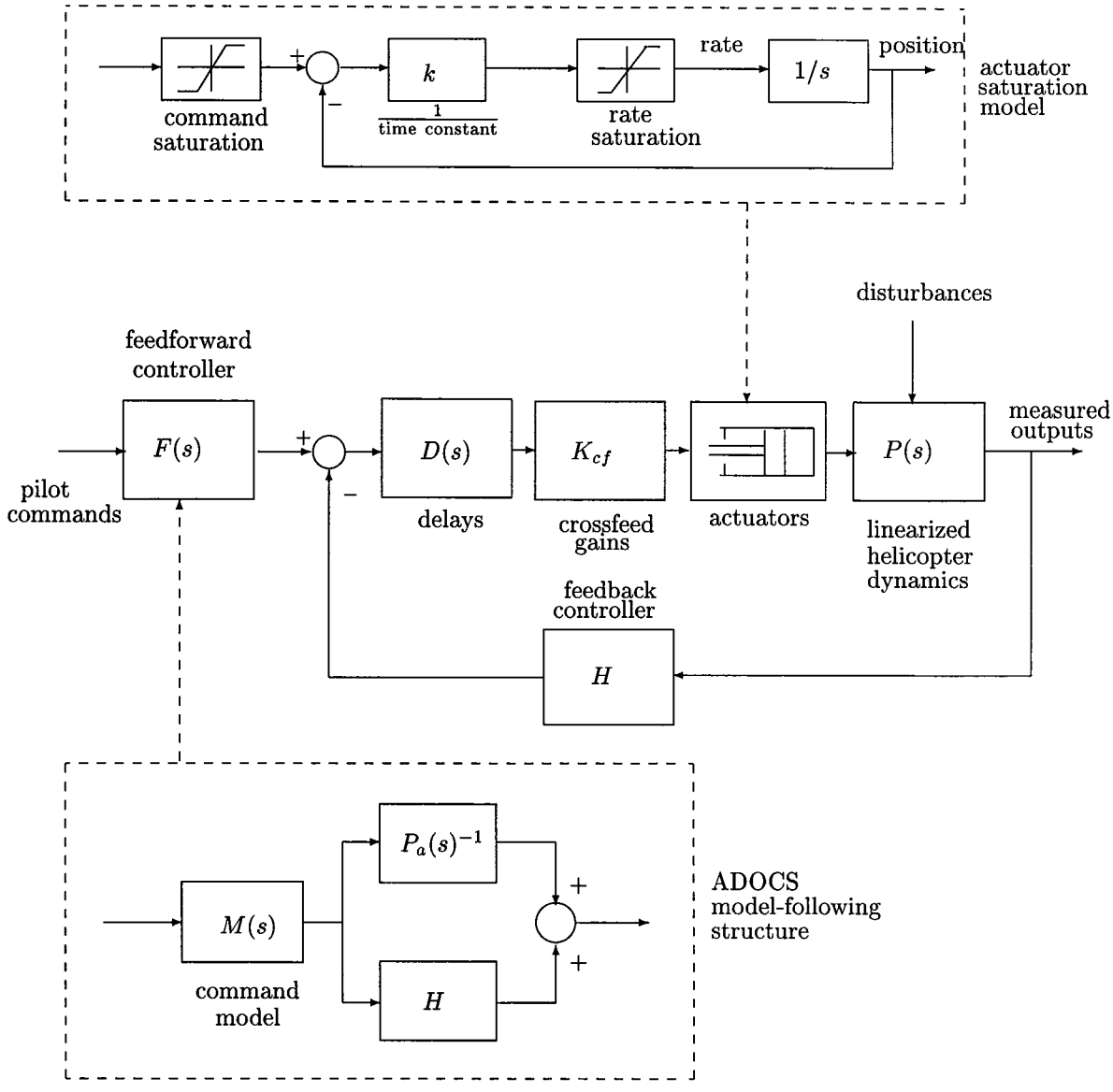

Figure 3.1: Model of the UH-60A with the ADOCS controller and crossfeed gains.

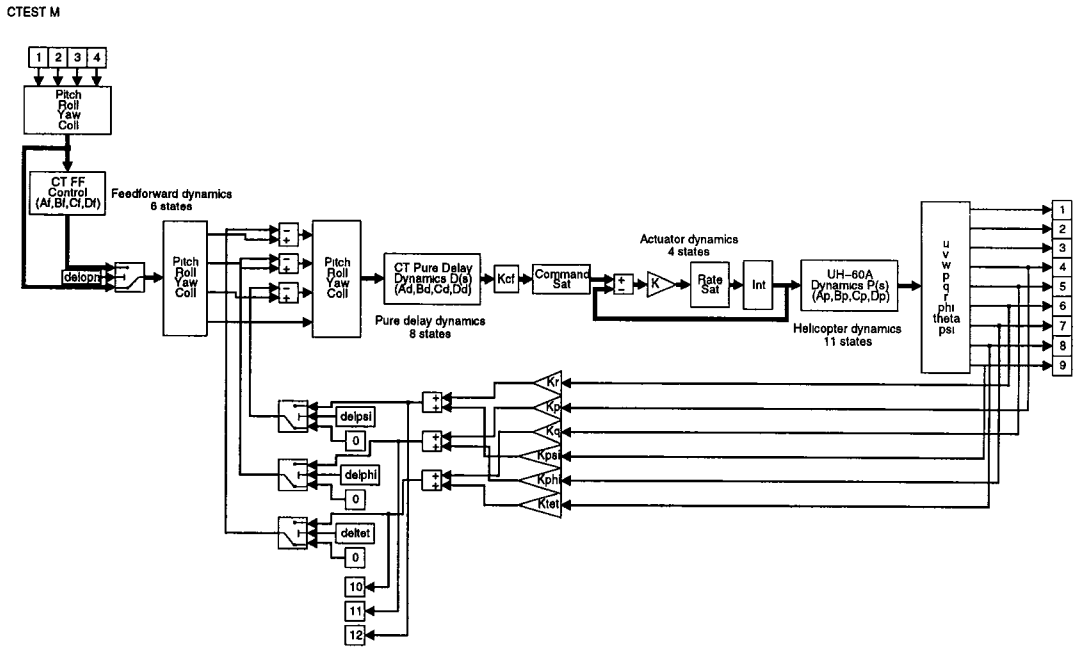

Figure 3.2: Continuous time ADOCS UH-60A model.

- The overall volume of code (number and size of m-files) required to run the simulations is significantly reduced.

The hover problem was initially posed in SIMULINK with separate block diagram representations for each of the five hover specifications. Each block diagram represented a vehicle from which either output time histories or static control channel (pitch, roll, yaw, collective) dynamics were obtained. This information was then used to determine compliance with the specifications. As a better understanding of both the hover problem and SIMULINK was achieved, it became apparent that only three block diagrams were necessary to complete the simulations efficiently and effectively .

Specification 1 (Bandwidth & Phase-Delay) and **Specification 2** (Stability Margins) are defined in terms of measurements taken from channel transfer

functions. Figure (3.2) is a continuous time SIMULINK representation of the ADOCS UH-60A dynamics used for checking compliance with Specs 1 & 2. The model's m-file name is 'ctest.m'.

Spec 1 is defined in terms of bandwidth and phase delay as measured from a frequency response (Bode) plot of angular attitude response to a cockpit controller input [9]. This spec is checked using a linear model. In order to extract the linear dynamics of the system presented in Figure (3.2), the command **linmod** is used. This command, which calls 'ctest.m', returns a linearized state space representation of the overall system. The actual command string is

$$[A1,B1,C1,D1] = \text{linmod('ctest')}; \tag{3.1}$$

The linearized state space dynamics (A1,B1,C1,D1) are then used to determine the bandwidth and phase delay (For more details see simu.m in Appendix A).

The square state space returned by the **linmod** command has a state for each of the states in the block diagram. The order of the states, which is non-deterministic (i.e. small changes to the block diagram can produce different ordering of the states), can be determined through use of the command string

$$[sizes,x0,states] = \text{ctest}([],[],[],0); \tag{3.2}$$

where ctest is the block diagram m-file name. Measurements based on system outputs, rather than states, are recommended because the outputs can be ordered and fixed by the designer in the block diagram model. The B, C and D matrices returned by the **linmod** command are dependent on the inputs and outputs specified by the designer.

Spec 2 is defined in terms of open-loop control system stability criteria independent of the feedforward control. The same block diagram, Figure (3.2),

is used for checking the open-loop specification. For open-loop calculations, the switch following the feedforward control dynamics is activated, by the **delopn** variable, to eliminate the dynamics and send the input directly into the forward loop. Unless activated, the switch sends the input through the feedforward control dynamics to the forward loop. Each of the three feedback switches, when activated by either **deltet**, **delphi** or **delpsi**, breaks the appropriate feedback loop (both position & rate feedback) for the open-loop calculations. Once the feedforward control is bypassed and the correct feedback loops are broken, the linearized open-loop dynamics are again extracted using the **linmod** command. The resultant linearized state space representation is then used to calculate the standard gain margin & phase margin through use of the MATLAB commands **bode** and **imargin**.

Specification 3 (Quickness) and **Specification 4** (Coupling) require output time histories of five seconds duration generated from nonlinear simulation. Figure (3.3) is a discrete time SIMULINK representation of the ADOCS UH-60A dynamics used for checking compliance with Specs 3 & 4. The model's m-file name is 'dtest.m'. All of the dynamics (feedforward, delay, actuator integrator and helicopter) are discretized representations of the continuous-time dynamics. The discretization is performed by bilinear transformation using the MATLAB command **c2dm** based on a sampling frequency of 30 Hz. The 30 Hz sampling frequency implies sample time spacing of $\triangle t = 0.0333$. 30 Hz was used for the following reasons:

- The ADOCS controller is a discrete-time system with a sampling frequency of 30 Hz.
- C-O requires fixed step sizes for its functional constraints.

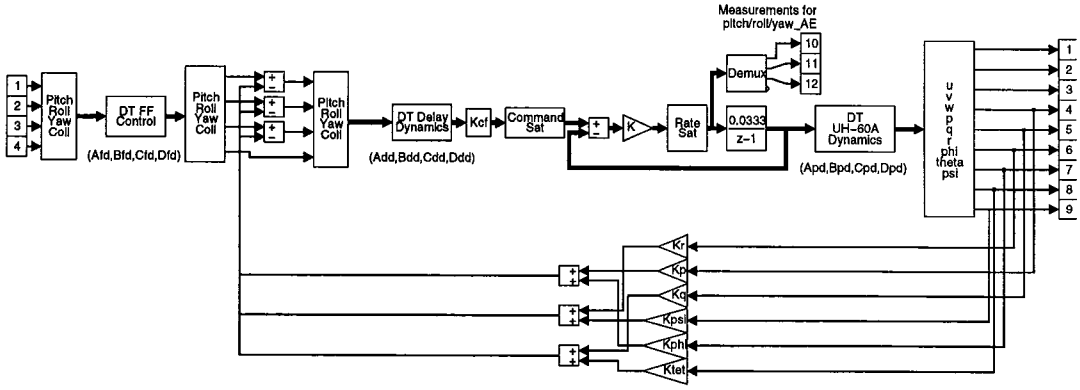

Figure 3.3: Discrete time ADOCS UH-60A model.

- Integration times are significantly reduced using fixed step size integration methods on discretized dynamics.

Spec 3 is defined in terms of the quickness ratio

$$\text{quickness} = \frac{\text{peak rate}}{\text{peak displacement}} \tag{3.3}$$

Measurement of the quickness ratio requires time histories for each of the pitch, roll and yaw channel rate and position responses to three separate inputs. These nine simulations make up the bulk of the overall simulation time. The time histories are generated by integrating the block diagram model 'dtest.m' in Figure (3.3). The integration is completed using the SIMULINK command **linsim**, which is the recommended integration method for systems which are relatively linear [4]. Other integration methods, such as Runge-Kutta, Adams, Gear and Euler are also available. Results from integration using the Adams and Euler methods are the same. The penalty for using these methods is increased

simulation time. The command string

$$[T, X, Y] = \text{linsim('dtest',tf,[],[],in);} \tag{3.4}$$

integrates the model 'dtest.m' from time 0 to time 'tf' based on the input specified in the input matrix 'in'. The first column of the input matrix 'in' must contain a time reference. Subsequent columns represent the specified inputs starting with the first. The two empty fields, represented by the '[]', are for initial condition and integration option arguments. The left hand arguments returned by the integration are T - the integration time vector based on the sample time specified in the model, X - the time oriented state vector and Y- the time oriented output vector.

Spec 4, defined in terms of limits on cross-coupling between the pitch & roll channels and the collective & yaw channels, is also checked from time histories generated by integration of the block diagram 'dtest.m' in Figure (3.3). The simulations necessary for checking the pitch-to-roll and roll-to-pitch coupling are accounted for in the Spec 3 simulations. Thus, only one additional simulation is necessary to determine the output response to a full collective input. This brings the total number of simulations involving model integration up to ten. These 10 simulations take a total of approximately 1.5 seconds when implemented in SIMULINK. The rest of the SIMULINK simulations and subsequent MATLAB calculations take approximately another 8.5 seconds, bringing the total time to just over 10 seconds per simulation. Before implementation in SIMULINK, the ten integrations took a total of approximately 8.5 seconds and the total simulation time was approximately 18 seconds. Considering that there can be as many as 25 simulations per C-O iteration and upwards of 25 iterations per optimization, the time savings resulting from SIMULINK implementation is at least an

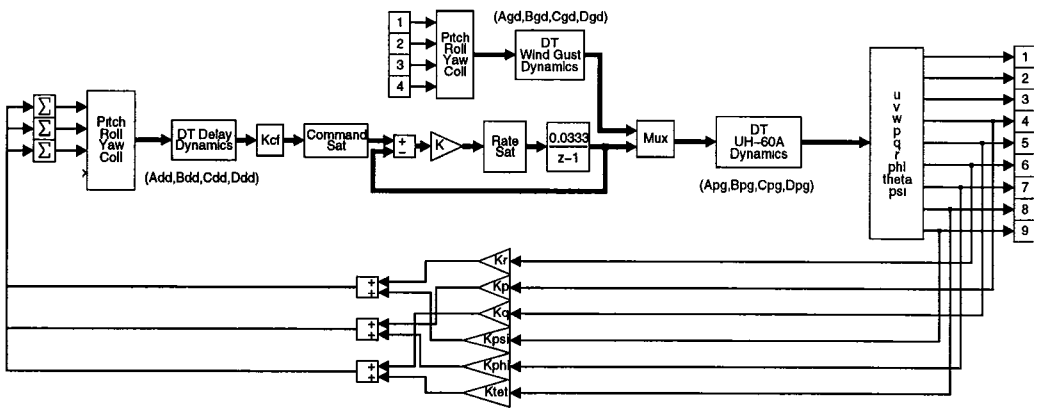

Figure 3.4: Discrete time wind-gust simulation model.

hour per optimization.

Specification 5 (Wind-Gust Rejection) was originally addressed separately from Specs 3 & 4 because, although it is also based time histories, it requires forty seconds of history and was the largest time consumer of the simulation. Thus, it was implemented using a linear simulation [9]. Ideally, the spec should be checked using the time history data from nonlinear simulations. Because of the speed of discrete time nonlinear simulation using SIMULINK, the time history data was determined using nonlinear simulation. Figure (3.4) is the discrete time SIMULINK model used for the wind-gust simulations. The model's m-file name is 'dwtest1.m'.

Both the everywhere feasible and optimal solutions to the hover ADOCS controller problem from Reference [10] have been reproduced using the SIMULINK based simulation. The feasible solution, shown in Figure (3.5), was achieved after five C-O iterations. The optimal solution, with respect to minimum actuator

energy, is shown in Figure (3.6). Both the feasible and optimal solutions, though not exact replications of those from [10], are only slightly different numerically. The performance maps for the feasible and optimal designs are essentially the same. Numerical differences between design parameter values are no larger than 5%. These differences are attributable to numerical differences between the integration methods used, not errors. The initial SIMULINK based hover design results were verified with previously obtained results from the initial hover design. Trade-off and robustness studies, although not specifically shown, can also be performed using the SIMULINK based ADOCS Hover problem setup.

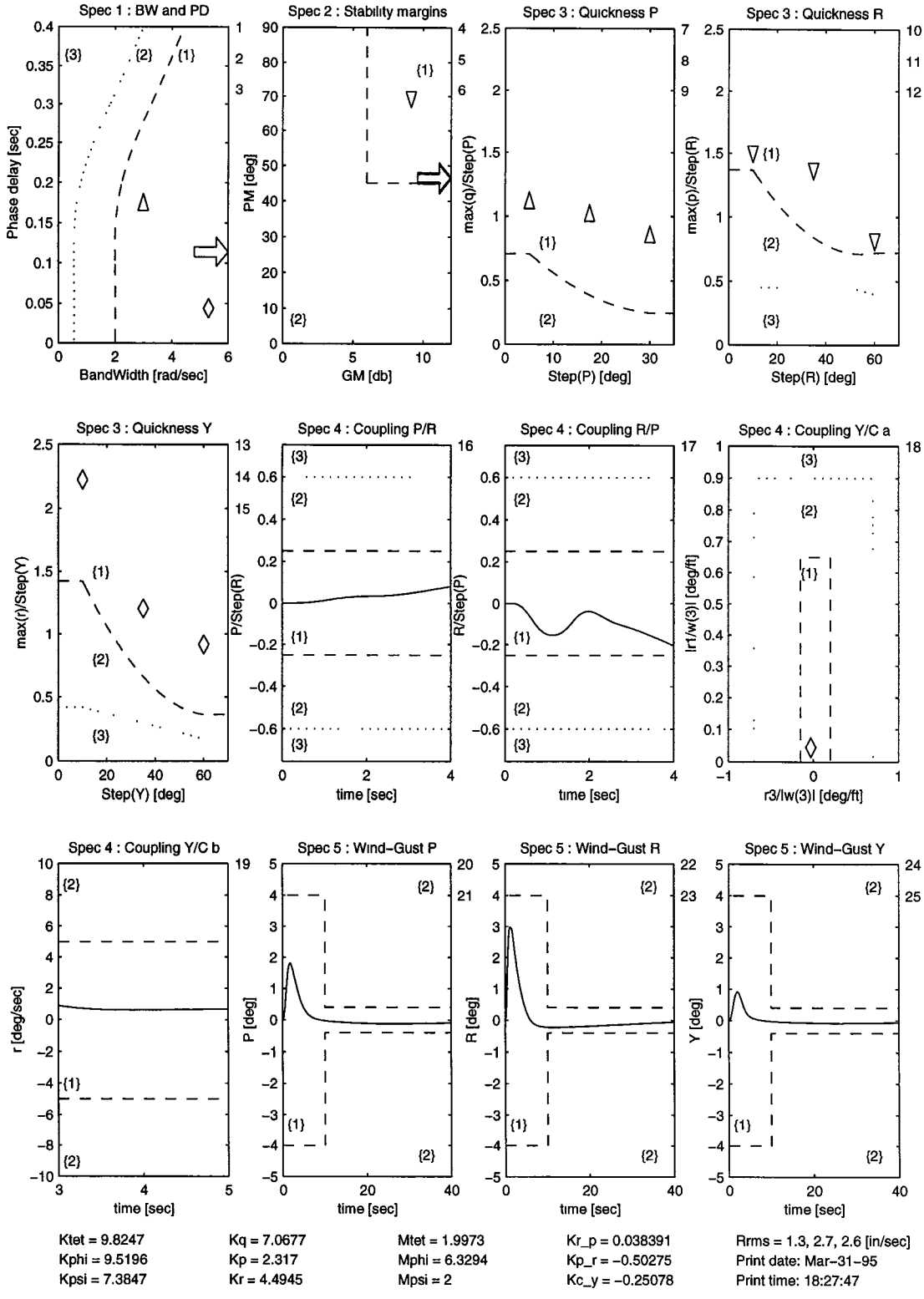

Figure 3.5: A feasible controller from SIMULINK using the ADOCS structure with crossfeed gains; $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

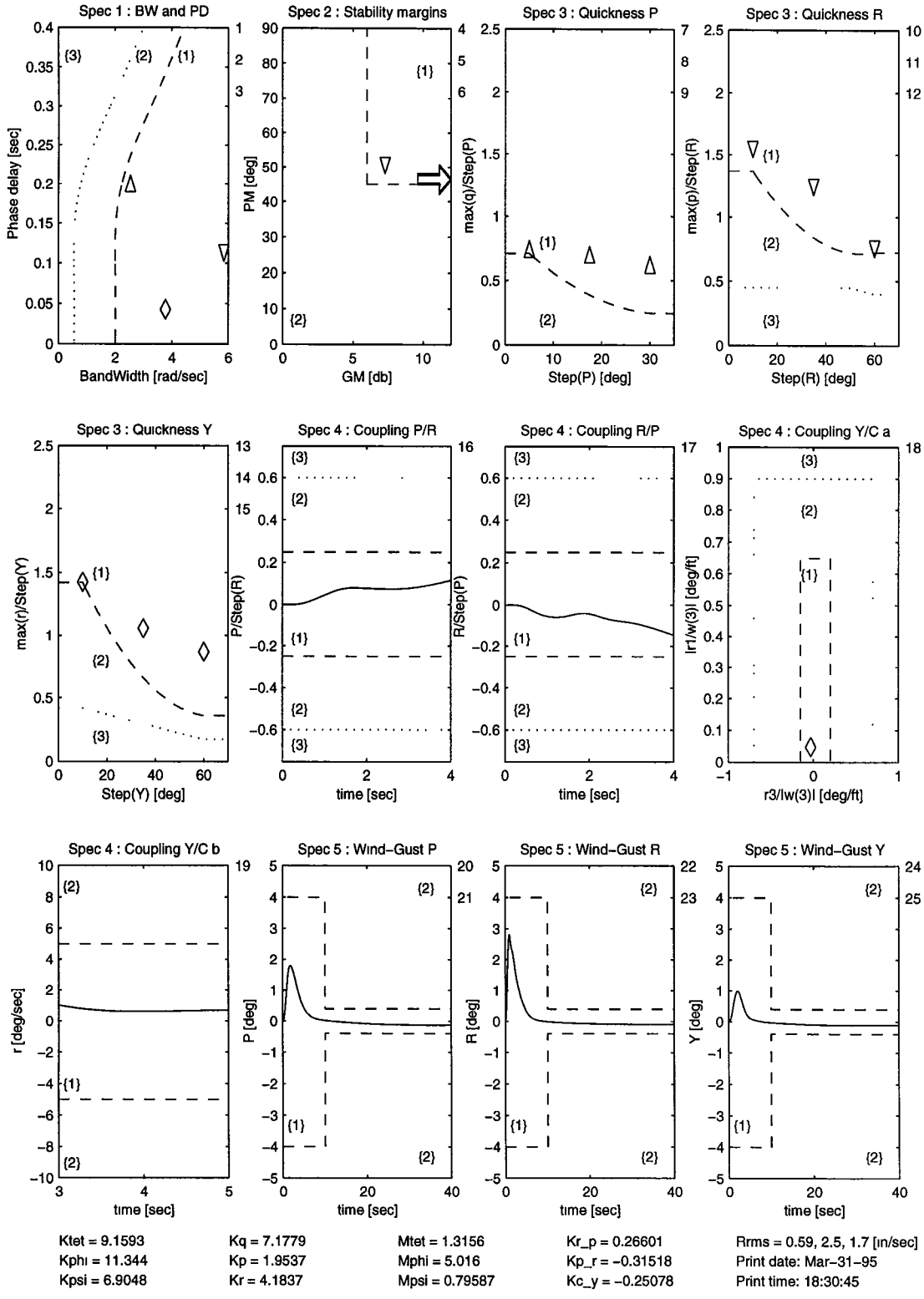

Figure 3.6: An optimal controller from SIMULINK using the ADOCS structure with crossfeed gains; $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

Chapter 4

The 80 Knot Forward Flight Problem Setup

Application of both the optimization and model-based simulation methodologies, proven by the solution of the hover problem using both MATLAB and SIMULINK, to a forward flight problem is the primary objective of this research. The 80 knot forward flight condition was chosen as a benchmark for the forward flight problem. Both system modeling and the translation of the ADS-33C requirements to CONSOL-OPTCAD specifications are required in the setup and solution of the 80 knot forward flight problem.

The general system model, shown in Figure (3.1) of Chapter 3, remains unchanged. Specific components of the general model, the command model $M(s)$ and the linearized helicopter dynamics $P(s)$, require modification for forward flight simulation. What follows is a discussion of the necessary changes to these two specific components of the overall model.

(*i*) **ADOCS Command Model,** $M(s)$ - For velocities $\geq 40kts$, the command model in the roll channel changes from a third order position command model to a second order rate command model. Neither the pitch nor the yaw channel command models are affected for velocities $\geq 40kts$.

The velocity dependent command models, reproduced from Reference [7], are presented in 4.1.

AXIS	HOVER	V $\geq$ 40 kts
Pitch, $\frac{\theta_m}{\theta_c}$	$\frac{4(s+0.2)}{s(s+2)(s+2)}$	Same as hover
Roll, $\frac{\phi_m}{\phi_c}$	$\frac{6.25(s+0.33)}{s(s+2.5)(s+2.5)}$	$\frac{5.08}{s(s+5.08)}$
Yaw, $\frac{\psi_m}{\psi_c}$	$\frac{2}{s(s+2)}$	Same as hover

(4.1)

The forward flight command model is implemented as $M(s) = \text{diag}(M_\theta(s), M_\phi(s), M_\psi(s))$, with

$$\begin{aligned} M_\theta(s) &= \frac{\alpha_\theta^2}{(s+\alpha_\theta)^2} \\ M_\phi(s) &= \frac{\alpha_\phi}{s(s+\alpha_\phi)} \\ M_\psi(s) &= \frac{\alpha_\psi}{s(s+\alpha_\psi)}. \end{aligned} \tag{4.2}$$

Each channel command model is specified in terms of a single parameter. Since the command model is not an ADS-33C requirement, the parameters α_θ, α_ϕ and α_ψ are free to be design parameters. The pole at $s = 0$ and the zero at $s = -0.2$ in the pitch command model are eliminated, thereby enabling specification in terms of a single design parameter. The elimination of the pole and zero was done for the following reasons[2]:

- For simplicity, to keep the number of design parameters to a minimum and to reduce the model complexity.
- For the frequency range in question, [0.1 10] rad/sec, removing the pole and zero reduces the low frequency gain to unity. The dynamics

[2]Phone conversation between Dr. Mark Tischler, Professor Bill Levine and Gil Yudilevitch, May 1992

still represent a lowpass filter with a cutoff frequency of about 0.5 rad/sec.

- Without the pole and zero, the command model produces an attitude command , as required. That is, the step response attains a steady state value. With the pole and zero present, the command model is a rate command model. The rate command model step response is monotone increasing.

This was also done for the ACAH pitch and roll channel command models in the hover flight simulation.

The actual series connection between the command model, $M(s)$, and the feedforward blocks $P_a^{-1}(s)$ and $H(s)$, is represented as $(P_a^{-1}(s)+H(s))M(s)$. $P_a(s)$ is a second-order approximation of the helicopter dynamics where $P_a(s) = \text{diag}(\tilde{p}_\theta(s), \tilde{p}_\phi(s), \tilde{p}_\psi(s))$, with

$$\begin{aligned} \tilde{p}_\theta(s) &= \frac{M_{\delta_\theta}}{s(s+\frac{1}{\tau_\theta})} \\ \tilde{p}_\phi(s) &= \frac{M_{\delta_\phi}}{s(s+\frac{1}{\tau_\phi})} \\ \tilde{p}_\psi(s) &= \frac{M_{\delta_\psi}}{s(s+\frac{1}{\tau_\psi})}. \end{aligned} \tag{4.3}$$

The "dynamic cancellation" properties of the feedforward controller must remain constant. Ideally, if $P_a(s) = P(s)$, the resulting closed loop transfer function is $M(s)$. Therefore the approximate parameters are fixed. $H(s)$ is a representation of the position/rate ("PD") feedback used. It is modeled as $H(s) = \text{diag}(h_\theta(s), h_\phi(s), h_\psi(s))$, with

$$\begin{aligned} h_\theta(s) &= K_q s + K_\theta \\ h_\phi(s) &= K_p s + K_\phi \\ h_\psi(s) &= K_r s + K_\psi. \end{aligned} \tag{4.4}$$

For the hover flight condition the following code in simu.m creates the state space representation of the series combination of the command model $M(s)$ and the feedforward blocks $P_a^{-1}(s) + H(s)$:

```
% Feedforward (Including Model Following & "Inverse Closed-Loop")
% -------------------------------------------------------------

% Continuous Time
% ---------------

[Af_tet,Bf_tet,Cf_tet,Df_tet] = ...
tf2ss(Mtet^2*[1/FGtet Kq+1/FGtet/FTtet Ktet],[1 2*Mtet Mtet^2]);
[Af_phi,Bf_phi,Cf_phi,Df_phi] = ...
tf2ss(Mphi^2*[1/FGphi Kp+1/FGphi/FTphi Kphi],[1 2*Mphi Mphi^2]);
[Af_psi,Bf_psi,Cf_psi,Df_psi] = ...
tf2ss(Mpsi*[1/FGpsi Kr+1/FGpsi/FTpsi Kpsi],[1 Mpsi 0]);
[Af,Bf,Cf,Df] = ...
append(Af_tet,Bf_tet,Cf_tet,Df_tet,Af_phi,Bf_phi,Cf_phi,Df_phi);
[Af,Bf,Cf,Df] = append(Af,Bf,Cf,Df,Af_psi,Bf_psi,Cf_psi,Df_psi);
[Af,Bf,Cf,Df] = append(Af,Bf,Cf,Df,[],[],[],1);
```

A simple change to the second **tf2ss** command line implements the rate command model for the roll channel. The change in the roll channel command model for the forward flight condition is reflected in the following code from simu.m:

```
% Feedforward (Including Model Following & "Inverse Closed-Loop")
% -------------------------------------------------------------

% Continuous Time
% ---------------

[Af_tet,Bf_tet,Cf_tet,Df_tet] = ...
tf2ss(Mtet^2*[1/FGtet Kq+1/FGtet/FTtet Ktet],[1 2*Mtet Mtet^2]);
[Af_phi,Bf_phi,Cf_phi,Df_phi] = ...
tf2ss(Mphi*[1/FGphi Kp+1/FGphi/FTphi Kphi],[1 Mphi 0]);
[Af_psi,Bf_psi,Cf_psi,Df_psi] = ...
```

```
tf2ss(Mpsi*[1/FGpsi Kr+1/FGpsi/FTpsi Kpsi],[1 Mpsi 0]);
[Af,Bf,Cf,Df] = ...
append(Af_tet,Bf_tet,Cf_tet,Df_tet,Af_phi,Bf_phi,Cf_phi,Df_phi);
[Af,Bf,Cf,Df] = append(Af,Bf,Cf,Df,Af_psi,Bf_psi,Cf_psi,Df_psi);
[Af,Bf,Cf,Df] = append(Af,Bf,Cf,Df,[],[],[],1);
```

(*ii*) **Linearized and reduced (UMGenhel) dynamic model** $P(s)$ - A bare airframe model consisting of 11 states, 9 for the 6 DOF fuselage dynamics and 2 for the main rotor flapping motions is used to represent the UH-60A in hover. Higher order dynamics are included in the model through the use of model reduction techniques. To simulate the UH-60A in forward flight at 80 knots, a similar state space representation of the 6 DOF fuselage dynamics and main rotor flapping motions is used. The linearized helicopter dynamics are derived from the 224 state fully nonlinear UMGenhel model trimmed for straight and level forward flight at 80 knots. The dynamics are linearized about the trim point. The dynamic response of the linearized and reduced helicopter model is then compared with that of the full order nonlinear model to ensure accurate model reduction.

As with the hover problem, there are five specifications essential to meeting the ADS-33C forward flight requirements. The five forward flight specifications are labeled in the same manner as the five hover flight specifications. They are based on the ADS-33C forward flight requirements for Instrument Meterological Conditions (IMC) and/or Divided Attention Mission-Task-Elements (MTEs). The IMC and/or Divided Attention MTE requirements were chosen both to parallel the Level 1 hover flight specifications (also defined in terms of IMC and/or Divided Attention MTE requirements and to facilitate definition of ultra-high performance based on the Aggressive Maneuvering Air Combat MTE require-

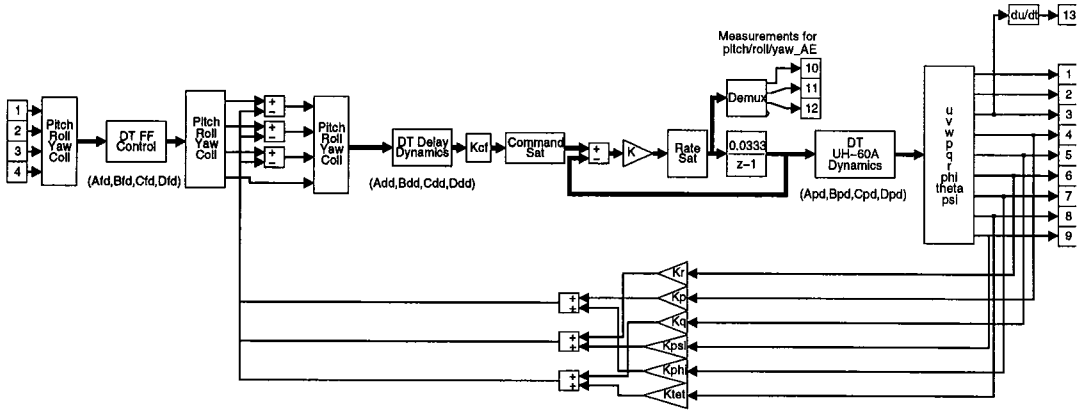

Figure 4.1: Discrete time ADOCS UH-60A model.

ments. A flight control system that meets these IMC and/or Divided Attention MTE requirements effectively enables the pilot to perform non-flight-control related tasks for short periods of time. That is, the pilot can devote attention to other cockpit matters such as communications or visual contact identification without compromising control of the rotorcraft. The five specifications naturally fall into two categories. Specs 1 & 2 are based on both open- and closed-loop linear system representations determined from continuous time models. Specs 3, 4 & 5 require nonlinear simulation to generate from five to 40 seconds of time history generated using discrete time models. SIMULINK models are again used to generate both the linear system representations and the nonlinear time histories. Two of the three models 'ctest.m' and 'dwtest1.m', shown in Figures (3.2) and (3.4), respectively, of Chapter 3 are general enough in their definition that they can be used for forward flight simulation without modification. The only modification to the model 'dtest.m' necessary is the addition of a differentiator to provide vertical acceleration, $\frac{d}{dt}$(vertical rate), as an output. The modified

model, also called 'dtest.m', is shown in Figure (4.1). As with the hover problem, compliance with Specs 1 & 2 is determined through use of 'ctest.m', with Specs 3 & 4 through 'dtest.m' and with Spec 5 through 'dwtest1.m'.

The five specifications are

- **Spec 1** - Short-Term Attitude Response to Control Inputs (Bandwidth & Phase Delay)
- **Spec 2** - Mid-Term Attitude Response to Control Inputs (Stability Margin Criteria)
- **Spec 3** - Moderate-Amplitude Attitude Changes (Attitude Quickness)
- **Spec 4** - Interaxis Coupling
- **Spec 5** - Wind-Gust Rejection

Naturally, there are differences between the dynamics of a helicopter in hover and those of a helicopter in forward flight at 80 knots. Thus, there are some differences between the ADS-33C hover and forward flight requirements which affect the definition of the five specifications. Each specification and its relationship to the ADS-33C forward flight requirements is now addressed.

(*i*) **Spec 1: Short-Term Attitude Response to Control Inputs (Bandwidth & Phase-Delay)** Spec 1 covers the ADS-33C requirements (3.4.1.1, 3.4.5.1 and 3.4.7.1) which address the pitch, roll and yaw short-term attitude response to control inputs. These requirements are defined in terms of a minimum bandwidth and phase delay which are obtained from a Stability Control Augmentation System-ON (SCAS-ON) frequency response (Bode) plot of angular attitude response to cockpit controller input as shown in

Figure (4.2)-a. SCAS-ON refers to the helicopter with all augmentation lops closed.

The level regions for the short-term response requirement are defined in the bandwidth-phase-delay plane. An example of these level regions is presented in the first subplot of Figure (4.3).

The forward flight requirements for the pitch and roll channels are exactly the same as their counterpart hover requirements, (Pitch & Roll - 3.3.2.1). There is no forward flight yaw channel short term response requirement for IMC and/or Divided Attention, it is only specified for Air Combat. The minimum bandwidth for the Air Combat requirement is 1.5 times more restrictive than the IMC and/or Divided Attention requirement for the pitch channel. A requirement on the yaw channel bandwidth and phase-delay, equivalent to the requirements on the pitch and roll channels for IMC/Divided Attention, has been implemented, for good reason. It is not unreasonable to require a satisfactory short term response in the yaw channel for IMC and/or Divided Attention operations when a much more stringent requirement exists for Air Combat operations. The addition of this yaw channel requirement will facilitate trade-off studies because the Air Combat requirement is used to define ultra-high performance. Thus, a feasible IMC/Divided Attention controller will always have a satisfactory yaw channel short-term response which precludes the expenditure of significant C-O effort to meet higher (Air Combat) yaw channel requirements. For ACAH response types, bandwidth is taken as the frequency of 45° of phase margin ($\omega_{BW_{\text{phase}}}$) and can be directly measured from the phase plot of the frequency response. For Rate response types, bandwidth is taken

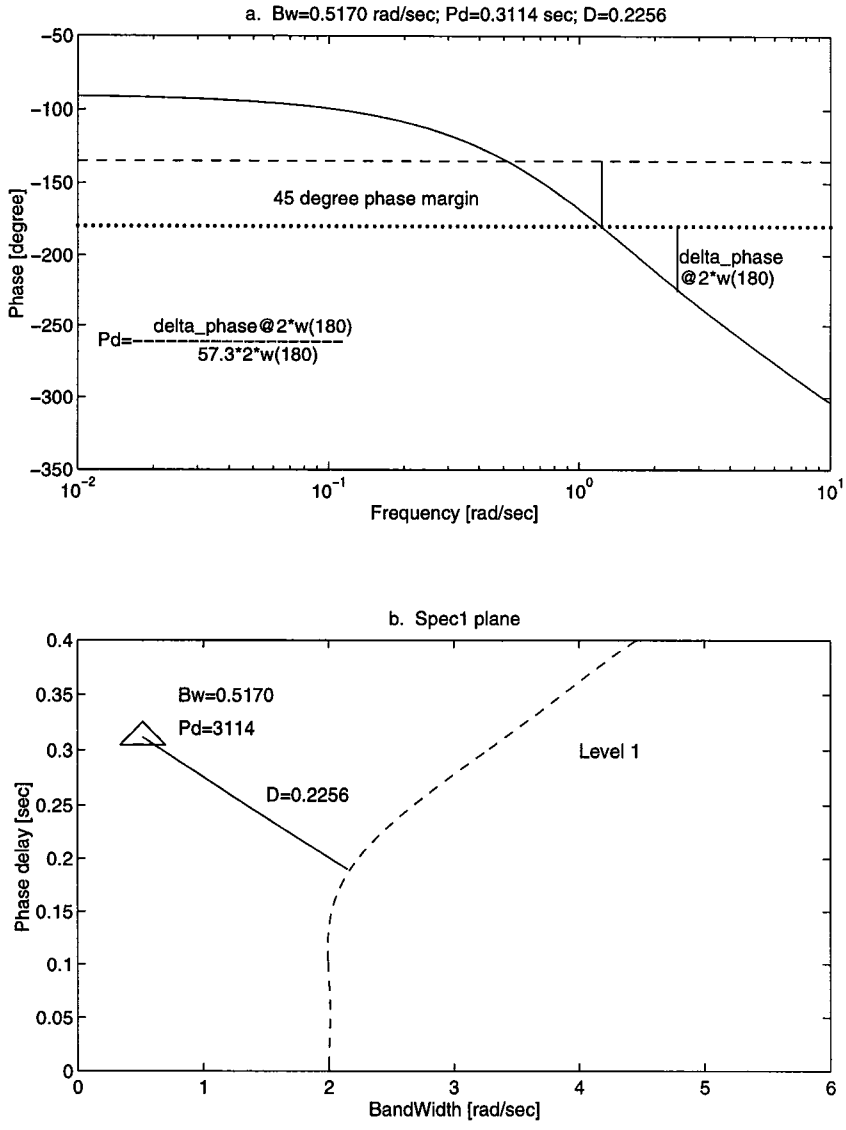

Figure 4.2: Spec 1: Bandwidth vs. Phase-delay (a) Bode phase plot, bandwidth and phase-delay of an example. (b) Its D measure in the Spec 1 plane.

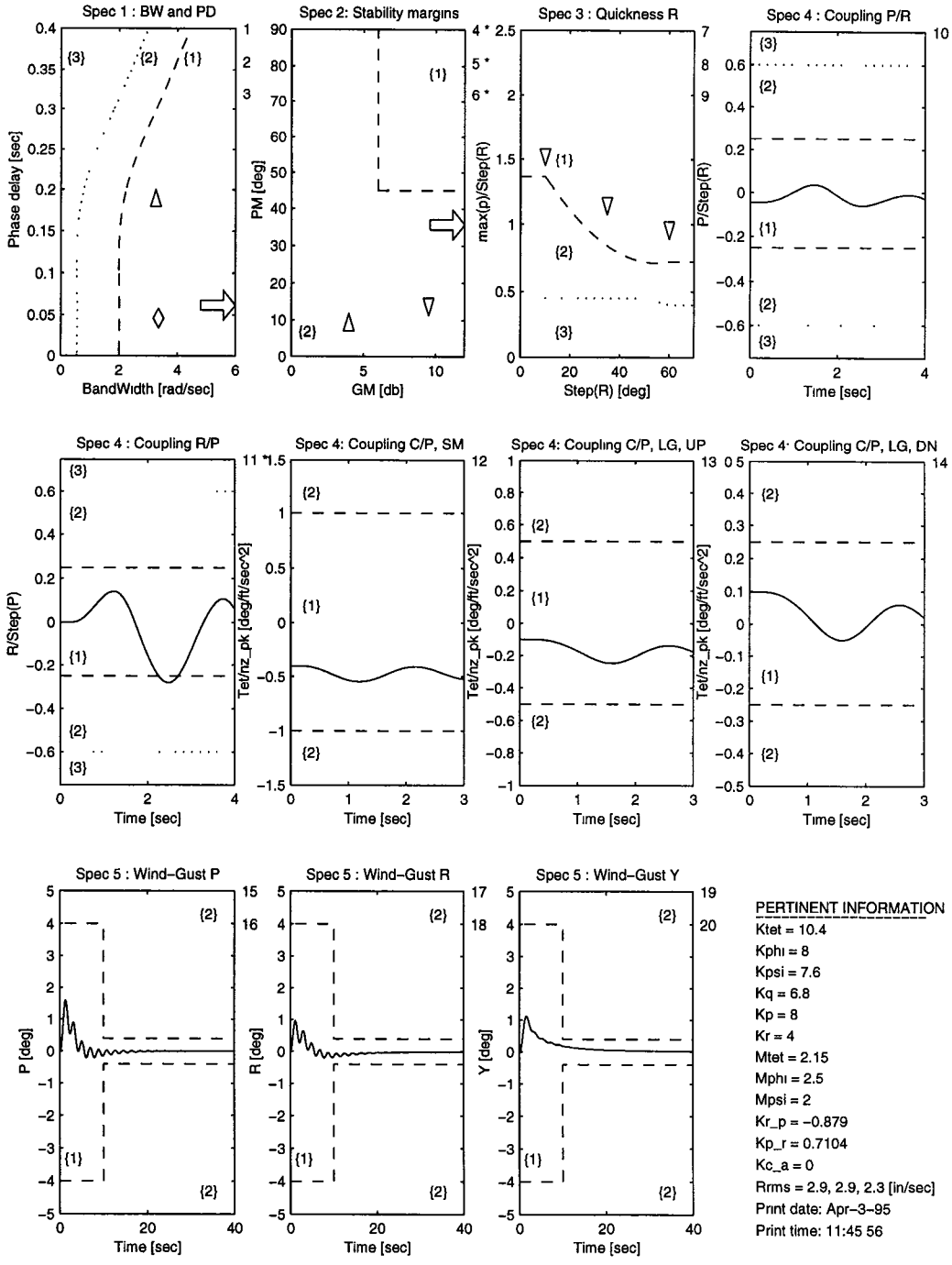

Figure 4.3: Performance map for the 80 knot forward flight ADOCS UH-60A design. All 5 specs are shown in eleven subplots. Dashed line - LEVEL 1/LEVEL 2 boundary. Dotted line - LEVEL 2/LEVEL 3 boundary. Specs are numbered 1, 2, ..., 20. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

as the lesser of $\omega_{BW_{\text{phase}}}$ or $\omega_{BW_{\text{gain}}}$, where $\omega_{BW_{\text{gain}}}$ is the frequency of 6 dB of gain margin. This calculation involves both the magnitude and phase plots. See Figure (2(3.3)), on page 19 of [8], for the ADS-33C definitions of bandwidth and phase delay. Spec 2, discussed in item (*ii*), requires at least 6 dB gain margin and 45° phase margin in each of the pitch, roll and yaw channels. Based on the ADS-33C definitions, a channel with exactly 6 dB gain margin and 45° phase margin has $\omega_{BW_{\text{phase}}} = \omega_{BW_{\text{gain}}}$. In general, the channel gain margin is $\gg 6$ dB and the phase margin is $\simeq 45°$ which makes $\omega_{BW_{\text{phase}}} < \omega_{BW_{\text{gain}}}$. Since the forward flight pitch channel has an attitude command model and attitude hold is required, see Reference [3], measurement of $\omega_{BW} = \omega_{BW_{\text{phase}}}$ can be accomplished directly from the phase plot. The forward flight roll and yaw channels are rate command channels which implies that measurement of ω_{BW} should be the lesser of $\omega_{BW_{\text{phase}}}$ or $\omega_{BW_{\text{gain}}}$. In light of the fact that $\omega_{BW_{\text{phase}}}$ is $< \omega_{BW_{\text{gain}}}$ because of the GM/PM properties of the channels, measurement of ω_{BW} for the roll and yaw channels can also be accomplished directly from the phase plot.

(*ii*) **Spec 2: Mid-Term Attitude Response to Control Inputs (Stability Criteria)** This specification covers the ADS-33C requirements (3.4.1.2, 3.4.6.1 and 3.4.8.1) which address the pitch channel mid-term response, bank angle oscillations and lateral-directional oscillations. Recall from Chapter 2 that the stability criteria, $(GM) \geq 6$ db and phase margin $(PM) \geq 45°$, better define the desired mid-term response, which is defined in ADS-33C in terms of a damping ratio, ζ. A desired mid-term response is only defined for the pitch channel in forward flight. It is exactly the same as

its counterpart hover requirement (Pitch - 3.3.2.2). Thus, GM/PM criteria are defined for the pitch channel in spec 2.

The ADS-33C requirement on bank angle oscillations (3.4.6.1) is defined in terms of the parameter $\frac{\phi_{OSC}}{\phi_{AV}} = \frac{\phi_1 - \phi_2}{\phi_1 + \phi_2}$ where ϕ_1 and ϕ_2 are the first max and min, respectively, of the roll angular response to a pulse lateral cyclic input. This requirement effectively limits the oscillations in the roll channel and can be addressed by the stability margin criteria. Thus, GM/PM criteria are defined for the roll channel in spec 2.

The ADS-33C requirement on lateral-directional oscillations (3.4.8.1) is defined in terms of a natural frequency, ω_n, and damping ratio, ζ. These requirements can not be directly implemented as C-O specifications because they violate the smoothness conditions required by C-O. It is known that small changes in system parameters can greatly affect the system eigenvalues. It has been shown that lateral oscillations are dominated by an oscillatory mode equivalent to the Dutch Roll mode [3]. A lightly damped low frequency Dutch Roll mode imparts a motion on the rotorcraft not unlike that of an ice skater's body skating in a straight line. The rotorcraft oscillates in both the roll and yaw axes about the trim point. By ensuring both stability and sufficient damping of both the yaw and roll channels, through stability margin criteria in spec 2, both the bank angle oscillation and lateral-directional oscillation requirements can be met. For this reason a GM/PM criteria has also been defined for the yaw channel in spec 2.

The level regions for the mid-term response requirement are defined in the gain-margin-phase-margin plane. An example of these level regions is presented in the second subplot of Figure (4.3). Measurement of the

stability criteria is as described in Chapter 3.

(*iii*) **Spec 3: Moderate-Amplitude Attitude Changes (Attitude Quickness)** This specification covers the ADS-33C requirement (3.4.5.2) which addresses the roll channel moderate-amplitude attitude changes (attitude quickness). The forward flight requirement for the roll channel quickness is exactly the same as its counterpart hover requirement, (Roll - 3.3.3). Thus, the implementation of the spec and simulation necessary for checking compliance is the same. Recall from Chapter 2 that in light of the inherent C-O smoothness requirement, the hover quickness calculations were completed with the channel command input, δ_ϕ in the case of the roll channel, as the denominator instead of the maximum angular excursion $\triangle\phi_{pk}$. The forward flight quickness measurement, $p_{pk}/\triangle\phi_{pk}$, is implemented per ADS-33C even though it violates the smoothness requirement. This is possible because the roll channel quickness is always in spec which precludes the non-smooth constraint from ever becoming active. There is no forward flight quickness requirement for either the pitch or yaw channel. Recall from Reference [9] that for nonlinear systems the quickness specification depends on the input signal. The ADS-33C requirement states that the required attitude changes shall be made as rapidly as possible. In light of the rate command model in the roll channel, the required aggressive maneuvers can again be interpreted as the rate limited pulse used in the hover setup. A rate limited pulse input is presented in Figure (4.4).

(*iv*) **Spec 4: Interaxis Coupling** This specification covers the ADS-33C requirements (3.4.4.1 and 3.4.4.2) which address collective-to-attitude, pitch-

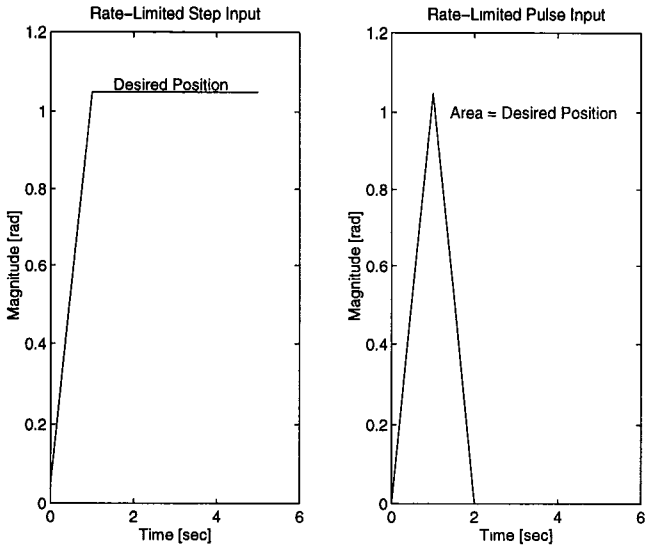

Figure 4.4: Rate limited input signals.

to-roll and roll-to-pitch coupling. The requirements on pitch-to-roll and roll-to-pitch coupling are exactly the same as their counterpart hover requirements (3.3.9.2). Thus, compliance with the specifications is determined through the same simulation and calculations. It is worthwhile to note that both the hover and forward flight pitch-to-roll and roll-to-pitch coupling requirements apply only to aggressive maneuvering. In the hover problem, they were implemented regardless of the MTE requirements. That is, a spot hover controller design meets aggressive maneuvering MTE pitch-to-roll and roll-to-pitch coupling requirements. These requirements did not significantly restrict either the feasible or optimal hover controller design. The same is not true for the forward flight controller design. Early attempts to achieve a feasible IMC and/or Divided Attention controller subject to the aggressive maneuvering pitch-to-roll and roll-to-pitch coupling requirements failed. Specifically, in order to meet the roll-to-pitch

coupling requirements, C-O increased the crossfeed $K_{\theta/\phi}$ to such a magnitude that the pitch channel bandwidth (Spec 1) was decreased below Level 1 and could not be restored. Rather than eliminate the pitch-to-roll and roll-to-pitch coupling requirements to allow a feasible controller solution, they were alternatively implemented. ADS-33C specifies both Level 1 and Level 2 performance levels for the coupling during aggressive maneuvering. The Level 2 performance requirement for aggressive maneuvering was implemented as the Level 1 performance requirement for IMC and/or Divided Attention operations. This alternative implementation, similar to the Spec 1 requirement on the yaw channel, will facilitate trade-off studies because the Level 1 aggressive maneuvering requirement is used to define ultra-high performance. The implementation also ensures there is not unreasonable coupling in routine forward flight.

In addition to the natural coupling between the pitch and roll channels discussed in Reference [9], helicopters in forward flight tend to experience coupling between collective controller inputs and pitch attitude. Excessive coupling will interfere with precision altitude control and could result in loss of attitude control in maneuvers, such as autorotation, where large collective changes are necessary [3].

The collective-to-attitude coupling is most significant for large collective commands, but the requirement is laid out for both small ($< 20\%$ of Full Control) and large ($\geq 20\%$ of Full Control) collective inputs. The limits on collective-to-attitude coupling are defined in terms of limits on the peak change in pitch attitude, θ_{pk}, occuring within the first three seconds following a step change in collective. The requirement is actually defined by

the ratio $\left|\frac{\theta_{pk}}{n_{z_{pk}}}\right|$, where $n_{z_{pk}}$ is the peak normal acceleration. The differentiator block used to derive normal acceleration from normal velocity can be seen in Figure (4.1). The rate limited step input of Figure (4.4) is used to represent step collective inputs. The collective-to-attitude requirements are shown in (4.5).

$$\begin{array}{lrcl} \text{Small Inputs} & \left|\frac{\theta_{pk}}{n_{z_{pk}}}\right| & \leq & 1.0 \text{ deg/ft/sec}^2 \\ \text{Lg Inputs UP} & \left|\frac{\theta_{pk}}{n_{z_{pk}}}\right| & \leq & 0.5 \text{ deg/ft/sec}^2 \\ \text{Lg Inputs DN} & \left|\frac{\theta_{pk}}{n_{z_{pk}}}\right| & \leq & 0.25 \text{ deg/ft/sec}^2 \end{array} \tag{4.5}$$

To avoid use of the non-differentiable functions max(·) and abs(·), suitable upper and lower limits are defined to bound the response over the relevant time interval based on the fact that

$$\left|\frac{\theta_{pk}}{n_{z_{pk}}}\right| \leq \gamma \Leftrightarrow -\gamma \leq \frac{\theta_{pk}}{n_{z_{pk}}} \leq \gamma \tag{4.6}$$

These upper and lower bounds are then used as C-O functional constraints [1].

Recall from Reference [9] that the static crossfeed gain matrix K_{cf} was used to improve the decoupling performance of the controller with $K_{cf} = \begin{bmatrix} 1 & K_{\theta/\phi} & 0 & 0 \\ K_{\phi/\theta} & 1 & 0 & 0 \\ 0 & 0 & 1 & K_{\psi/c} \\ 0 & 0 & 0 & 1 \end{bmatrix}$. The forward flight requirements do not limit the coupling between the collective and yaw channels, thereby eliminating the need for the crossfeed $K_{\psi/c}$. In light of the forward flight requirement limiting coupling between the collective and pitch channels, the crossfeed $K_{\theta/c}$ is used as a design parameter in place of $K_{\psi/c}$. The forward flight

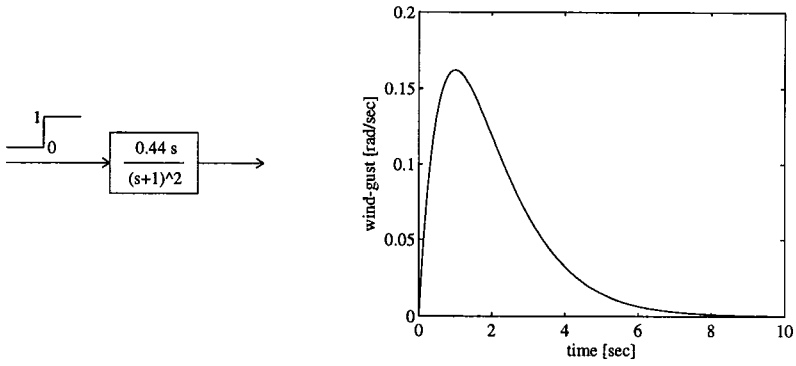

Figure 4.5: Spec 5: Wind-gust model and output waveform.

crossfeed gain matrix is now $K_{cf} = \begin{bmatrix} 1 & K_{\theta/\phi} & 0 & K_{\theta/c} \\ K_{\phi/\theta} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

(v) **Spec 5: Wind-Gust Rejection** As presented in Reference [9], the ADS-33C hover disturbance rejection requirements (Pitch & Roll - 3.3.2.3, Yaw - 3.3.7) are not applicable to the ADOCS configuration because the model-following concept prevents the disturbance rejection requirement (system stiffness) from being directly related to the overall system bandwidth. That is, the ADOCS configuration can give rise to a high-bandwidth, low-stiffness system. An alternative requirement, defined by using an approximate gust model given by the step response of $\frac{0.44s}{(s+1)^2}$, is used. The gust peak value is chosen to fit the disturbance input point. The wind-gust model and step response waveform are shown in Figure (4.5). The new wind-gust rejection requirement specifics are presented in Reference [6].

The ADS-33C forward flight disturbance rejection requirements (Pitch, Roll & Yaw - 3.4.10) are exactly the same as their counterpart hover requirements. Thus, they are not applicable to the forward flight ADOCS configuration. Again, the alternative requirement is utilized. Figure (3.4)

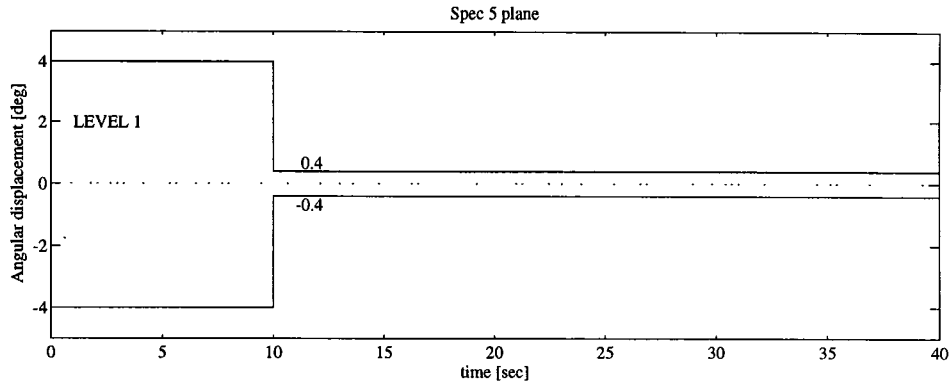

Figure 4.6: Spec 5: Wind-gust rejection - required envelope.

of Chapter 3 represents the discrete time model used to simulate gust inputs which are injected directly into the helicopter channel rate states. The response to gust inputs is required to lie between the two curves of Figure (4.6). These curves represent limits on the peak excursion and settling time, in accordance with the requirements for an ACAH response type from Reference [8].

Any of the ADS-33C forward flight requirements not directly addressed by the five forward flight specifications, with the exception of Turn Coordination - 3.4.6.2 and Spiral Stability - 3.4.8.2, have been intentionally excluded because there is nothing the ADOCS controller can do to ensure these requirements are met. The turn coordination requirement is not specifically addressed, though it should be, because measurement of the sideslip response to a pulse lateral cyclic input is limited by the helicopter model. The current helicopter model is trimmed to zero roll for forward flight at 80 knots with approximately 7.24° sideslip. This initial sideslip value is significantly higher than actual flight conditions and either validation or correction of the model has yet to occur. Once the sideslip trim value issue is cleared, the turn coordination requirement can be addressed by the specifications. The spiral stability requirement is not addressed by the

specifications because it is directly addressed by a C-O hard constraint requiring asymptotic stability (i.e., $\Re\{\lambda_m(A - BH)\} < 0$), used for numerical reasons, which ensures that there are no unstable modes.

Chapter 5

The 80 Knot Forward Flight Solution

The nominal forward flight design is defined in terms of the Level 1 ADS-33C forward flight requirements for IMC and/or Divided Attention MTEs (Mission Task Elements). For more information, see [8], Paragraph 3.4 - Forward Flight. These requirements are encompassed by the five CONSOL-OPTCAD specifications discussed in Chapter 4. A forward flight controller which meets the five C-O specifications meets or exceeds the ADS-33C forward flight Level 1 IMC/Divided Attention MTE requirements. IMC cruise, of which straight and level flight at 80 knots is a subset, is just one example of an IMC MTE.

Different initial design parameter configurations were used as starting points for the forward flight problem to observe their effects on both the resultant design and the ability and efficiency of the design technique. Examples of such configurations include:

(i) A configuration using the feasible solution to the hover problem, itself achieved from the ADOCS Current flight value design of [7], with the initial crossfeed gains set to zero ($K_{cf} = I$).

(ii) The feasible solution from (i) with the crossfeed gains as they were for

the hover solution. Since the crossfeed $K_{\theta/c}$ replaces $K_{\psi/c}$ for the forward flight simulation, $K_{\theta/c}$ was initialized to zero.

(*iii*) An open loop configuration in which the three crossfeed and six feedback gains were set to zero. The initial command model design parameters were as presented in (4.1), Chapter 4.

(*iv*) A unity feedback configuration in which the three crossfeed gains were set to zero and the six feedback gains were set to unity. The initial command model design parameters were as presented in (4.1), Chapter 4.

A feasible (Level 1) solution was obtained from the initial configurations described in items (*i*) and (*ii*) above. The performance map for the feasible Level 1 controller from item (*i*) is shown in Figure (5.1). A feasible design was not achieved after a considerable number of iterations for either of the configurations in items (*iii*) or (*iv*). It is difficult to prove that there is no feasible controller for either of these initial configurations, but finding such a controller is not easy. Engineering intuition and design experience can guide the engineer in determining the best infeasible design, if necessary. This was not necessary in light of the other initial configurations which yielded both feasible and optimal designs.

An objective function, based on an approximation for the actuator energy, allows determination of optimal solutions defined by minimum actuator energy. This is the same objective function used for the hover problem. An approximation for actuator energy was used because the simple actuator model of Figure (3.1) has only rate and position states. In order to calculate actuator energy ($E = \int_0^T (rate \cdot load) dt$), the actuator load is required in addition to the rate. Thus, a scaled RMS rate ($R = \int_0^T rate^2 dt$) is used to approximate actuator

energy. For more details see **qcost.m** in Appendix A.

In the course of optimizing the feasible solution presented in Figure (5.1) and the feasible solution obtained from item (*ii*), it was determined that optimal Level 1 solutions with very minimal channel actuator energy were achievable. An example of such an optimal solution is shown in Figure (5.2). Though all of the requirements are met, indicating Level 1 performance, something invisible to the optimization is allowing a significant reduction in actuator energy. After lengthy investigation, it was determined that the quickness measurement accounted for this discrepancy. The control system design goal is to meet the ADS-33C requirements and provide attitude and rate responses to control commands consistent with normal flight. When we consider the attitude responses to command inputs for the low actuator energy optimal controller, the discrepancy comes to light. Figure (5.3) shows the attitude responses to command inputs for the low actuator energy optimal controller. The poor attitude response in both the roll and yaw channels is directly related to the small feedforward design parameters. Recall from (4.1), Chapter 4, that the roll and yaw channels have rate command models and are completely specified in terms of their corresponding feedforward design parameter. The design parameter represents $1/\tau$ where τ is the desired time constant. As the design parameter is driven to an unrealistically low value by C-O in the reduction of the actuator energy the time constant is significantly increased. This high command model time constant manifests itself in a poor attitude response.

A study of the quickness measurement provides insight into the causes of this discrepancy. ADS-33C requires only the roll channel to meet a forward flight quickness specification. The roll channel quickness measurement is the

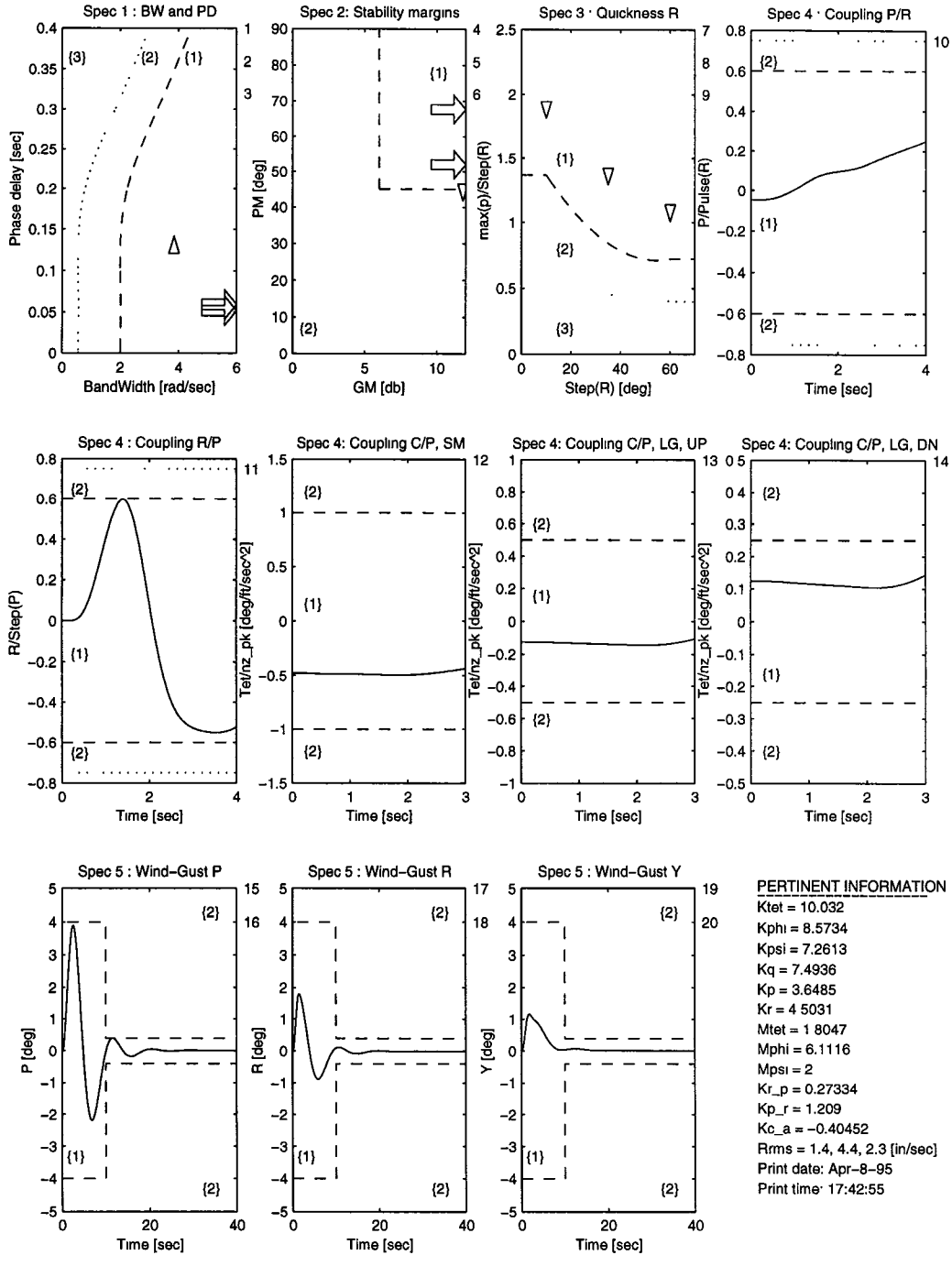

Figure 5.1: Performance map for feasible forward flight controller derived from feasible hover design with $K_{cf} = I$. Without constraints on attitude responses. $\triangle$ - pitch, $\triangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

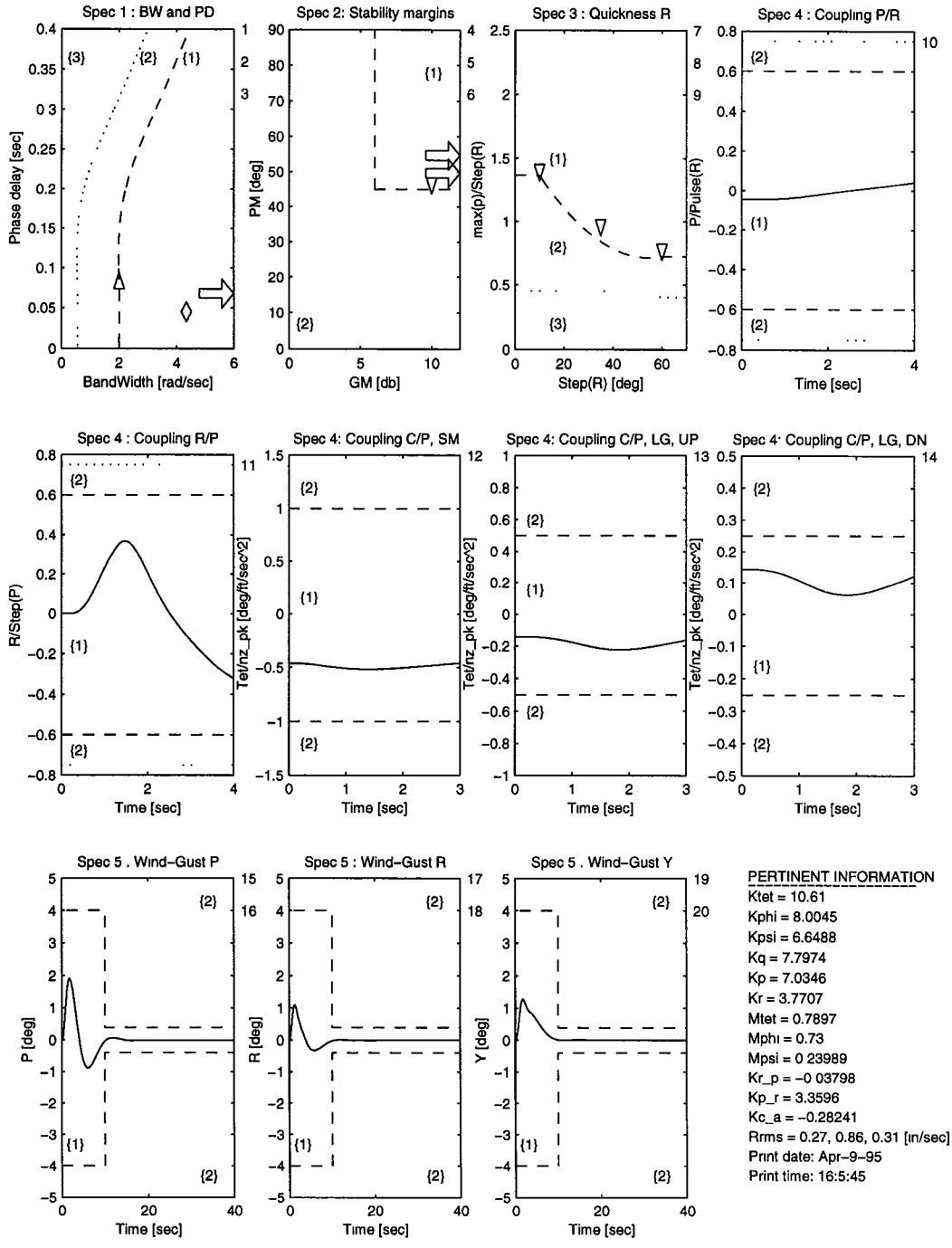

Figure 5.2: Performance map for low actuator energy optimal forward flight controller derived from feasible hover design with $K_{cf} = I$. $\triangle$ - pitch, $\triangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

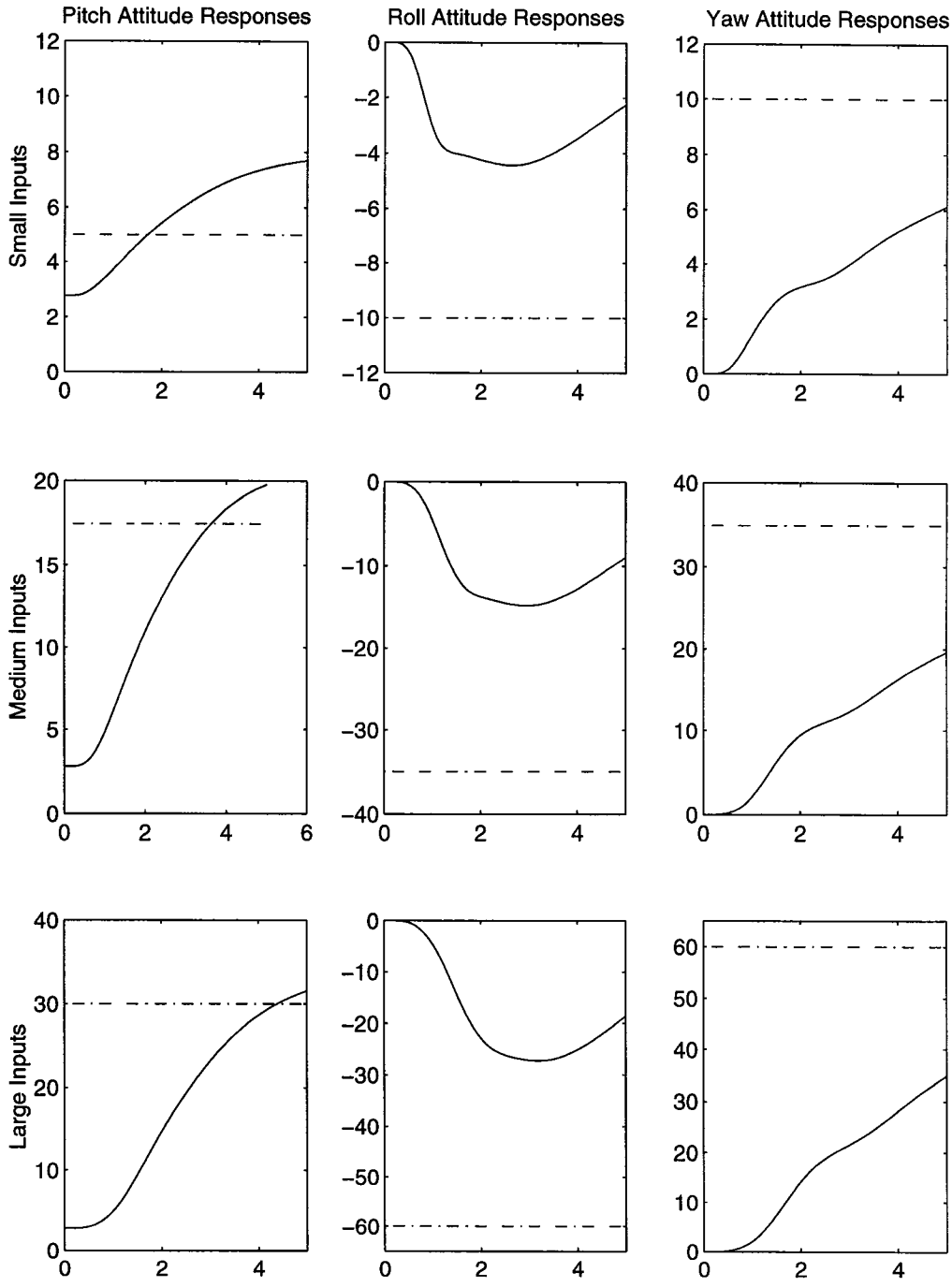

Figure 5.3: Attitude changes for the low actuator energy forward flight controller. Horizontal axis - Time [0 - 5 sec]. Vertical axis - Attitude [deg]. Desired response indicated by horizontal dashed line. For pitch channel (attitude command model), dashed line represents step input steady state value. For roll & yaw channels (rate command models), dashed line represents area under pulse input.

ratio of peak angular rate to peak angular attitude, measured in units of 1/sec. Specifically, the measurement is $p_{pk}/\triangle\phi_{pk}$ as described in both Chapters 2 and 4 and ADS-33C [8]. The actuator energy measurement, C-O objective, is directly related to the size of the desired command input, equivalently the magnitude of the signal in the control channel. The roll channel command model design parameter α_ϕ is related to the quickness and directly affects the size of the signal in the control channel. C-O is able to lower the actuator energy by reducing α_ϕ, which reduces the size of the signal in the control channel. A Level 1 quickness measurement is maintained because the ratio between peak rate and peak attitude is maintained. For the forward flight problem, the quickness measurement was implemented as a max/max with no constraints on either the angular rate or attitude. Since neither the rate nor attitude were constrained, C-O reduced both significantly by reducing the design parameter α_ϕ but preserved the ratio between them necessary to meet the quickness requirement. This reduction, from the decrease in the design parameter α_ϕ, served to lower the actuator energy almost to zero while maintaining quickness measurements at or above Level 1. The fact that this is occurring is not immediately apparent from the performance map because all the specs are at or above Level 1. Requirements on the minimum channel rates and/or attitudes were not implemented[3], although specified for the roll and yaw channels [8], because they are known not to drive the controller design as discussed in Chapters 2 and 4. In light of this discussion, they do limit the optimal design though. The roll rate limits and yaw attitude limits are only defined for large inputs and, as determined experimentally, do

[3]Phone conversation between Dr. Mark Tischler, Professor Bill Levine and P. J. Potter, March 1995

not limit the optimal design enough to prevent the lack of acceptable attitude response as presented in Figure (5.3).

Before discussing a method which prevents the optimal solution from being driven to zero actuator energy, a discussion of the optimal hover results and their connection to the quickness measurement is required. Recall from Chapter 2 that the hover quickness measurement applied to all three primary control channels and was defined in the same manner as the forward flight quickness measurement. For simplicity, consider only the yaw channel. The prescribed quickness measurement in the yaw channel is $r_{pk}/\triangle\psi_{pk}$. For the hover problem, the quickness measurement was implemented as r_{pk}/δ_{ψ}, where δ_{ψ} represented the yaw channel command input (desired position), a fixed parameter. This fixed parameter directly constrained the angular rates in each of the three channels through the quickness requirement. In order to meet the quickness requirement, the channel rates had to be above some reasonable level which kept the actuator energy above a reasonable level. There was no opportunity for C-O to maintain a satisfactory quickness ratio below certain values for the channel attitude rate. Based on these facts, a constraint on each of the channel attitude responses has been implemented to ensure an adequate optimal attitude response and reasonable actuator energy measurements. The max channel attitude response is required to be within 10% of the desired (command input) position. These constraints serve to limit the optimal solution by setting a lower bound on the command model design parameters. This lower bound limits the minimum Level 1 quickness achievable. Ideally, the command model is a second order representation of the closed loop dynamics in the presence of perfect cancellation by the feedforward control. Thus, a realistic model is required.

The feasible solutions for the initial configurations in items (i) and (ii) were recomputed based on the new constraints on the channel attitude responses. They are presented in Figures (5.4) and (5.5). This demonstrates that different feasible designs can be achieved by starting from different initial design configurations. Other methods to achieve different feasible designs include fixing specific design parameters such as in the command model or modifying the controller structure. Note that the feasible designs with and without the constraints on the attitude responses, Figures (5.1) and (5.4), are almost exactly the same. The attitude response constraints have little, if any, affect on the feasible solution. They do however, provide an optimal solution which can actually be flown because of the sufficient attitude response characteristics. Comparison of the two new feasible designs yields some interesting differences. The most significant differences between the two feasible designs can be seen in the roll-to-pitch coupling spec (2^{nd} row, 1^{st} col) and in both the pitch and roll wind gust response specs. The feasible design in Figure (5.5), resulting from the initial parameter configuration which left the crossfeed gains $K_{\theta/\phi}$ and $K_{\phi/\theta}$ as they were in the hover solution, has less oscillation in the pitch and roll channel wind-gust responses. The pitch-to-roll coupling and the roll-to-pitch coupling also have less overall deviation from trim. The most significant differences in the controller parameters between the two designs is in the roll channel feedforward design parameter (Mphi = α_ϕ) and the pitch-to-roll coupling design parameter (Kp_r = $K_{\phi/\theta}$). The decreased feedforward dp decreases the roll channel actuator energy. The increased crossfeed dp account for the smaller roll-to-pitch deviation from trim, but reduces the pitch channel bandwidth. The price paid for these qualitative differences can be seen in the pitch channel bandwidth and by comparing the

collective-to-attitude coupling responses (2^{nd} row, 2^{nd}, 3^{rd} & 4^{th} cols) of the two solutions. The flat collective-to-attitude response of the controller in Figure (5.4) indicates there is no pitch attitude response to any magnitude collective changes in either direction. For certain applications, this may be a desirable response.

Optimal solutions were obtained from both feasible designs. They are presented in Figures (5.6) and (5.7). The differences between the optimal solutions again lie with the interaxis coupling and wind-gust rejection responses. Also note the difference between the pitch channel bandwidth (Spec 1: BW & Phase Delay) for the two optimal solutions. The better roll-to-pitch coupling response is paid for through a reduced pitch channel bandwidth. Throughout the course of this research, the connection between the crossfeed design parameters and the channel bandwidth, channel coupling and channel stability margins has become more evident. It is not known to what extent these design parameters exactly affect the specs, but generalizations can be made. For example, a decrease in coupling from increasing a crossfeed gain is generally paid for through decreased channel bandwidth. Also, the feedback gains can be shown to affect channel bandwidth, gain margin and phase margin. Finally, the feedforward gains are known to directly affect the channel quickness.

To validate the use of the attitude response constraints as a method for both ensuring reasonable responses and reasonable actuator energies, the optimal attitude responses for the primary control channels are plotted in Figure (5.8). Compared to Figure (5.3), the improvement in attitude response at the cost of higher actuator energy is apparent.

There are many methods, most discussed in [9], for interpreting and guiding the design process. The following approach, applicable to either a new design

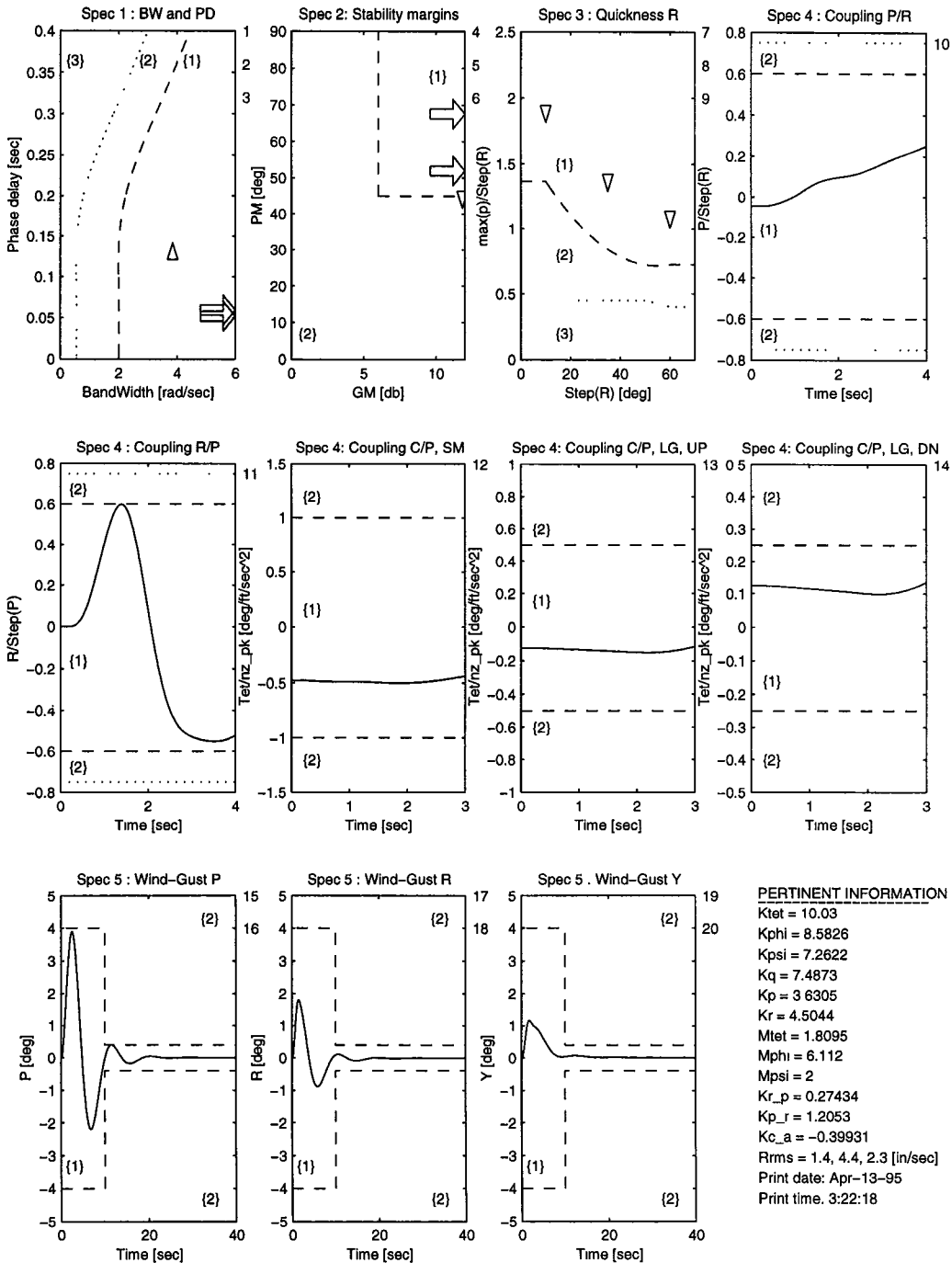

Figure 5.4: Performance map for feasible forward flight controller derived from feasible hover design parameters with $K_{cf} = I$. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

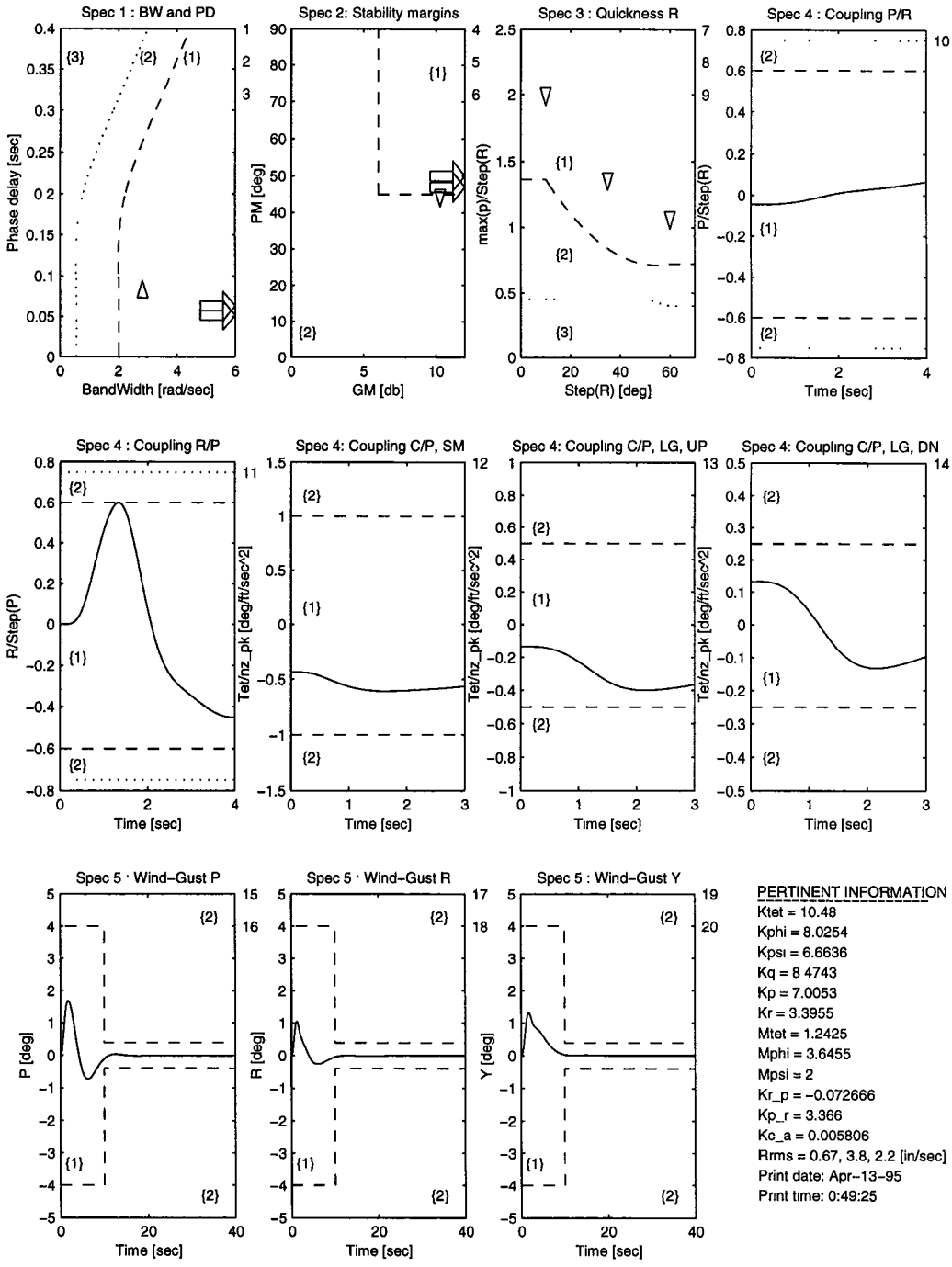

Figure 5.5: Performance map for feasible forward flight controller derived from feasible hover design parameters with crossfeeds intact. $\triangle$ - pitch, $\triangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

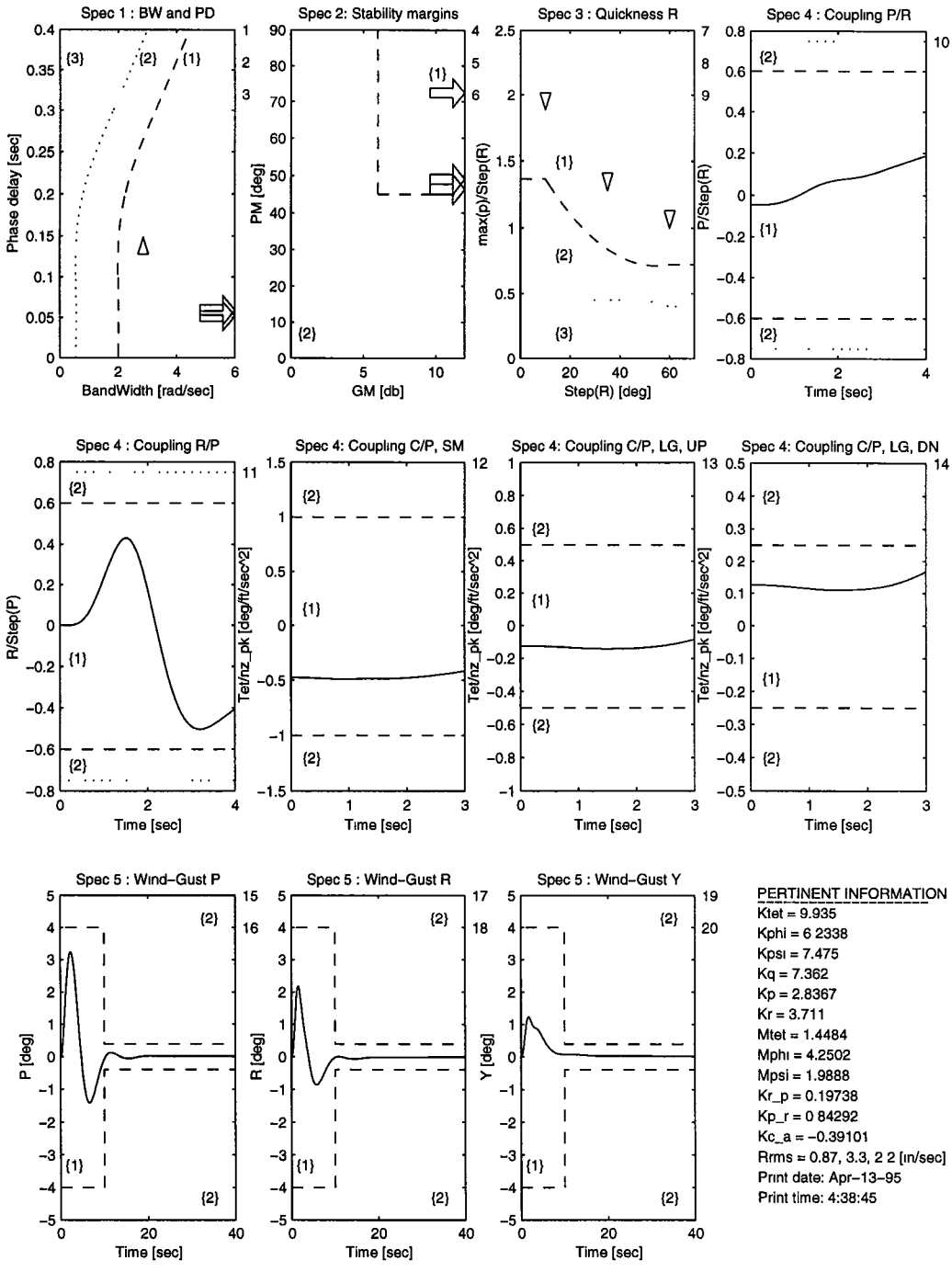

Figure 5.6: Performance map for optimal forward flight controller derived from feasible hover design parameters with $K_{cf} = I$. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

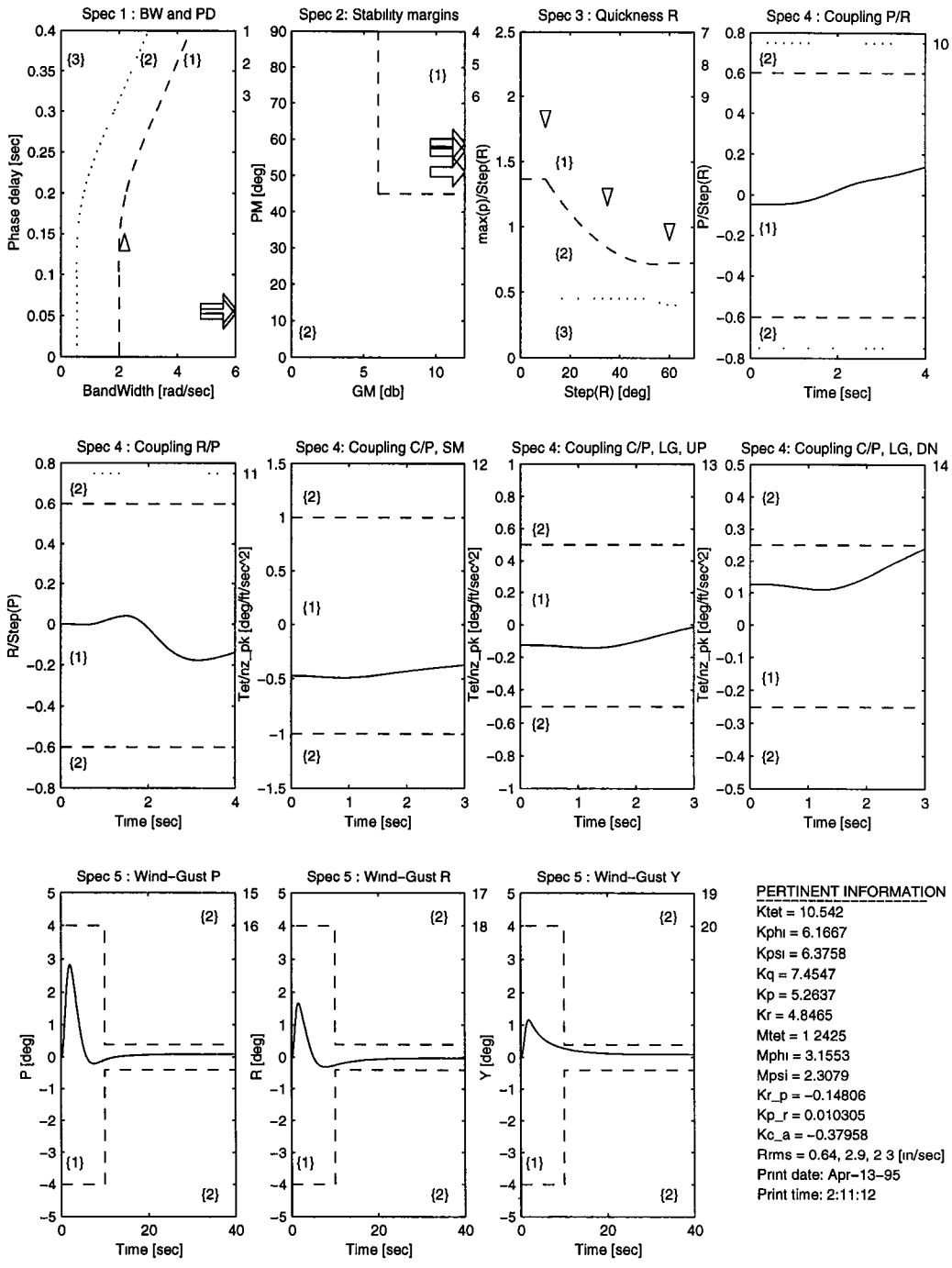

Figure 5.7: Performance map for optimal forward flight controller derived from feasible hover design with crossfeeds intact. $\triangle$ - pitch, $\triangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

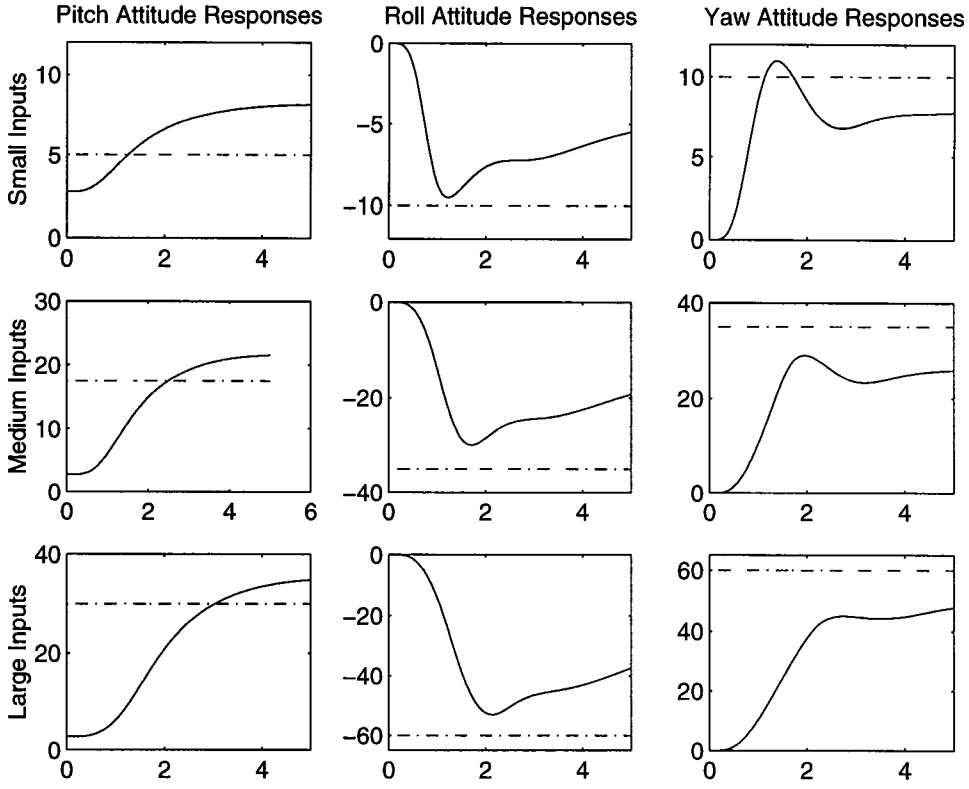

Figure 5.8: Optimal design attitude responses to command inputs. Horizontal axis - Time [0 - 5 sec]. Vertical axis - Attitude [deg]. Desired response indicated by horizontal dashed line. For pitch channel (attitude command model), dashed line represents step input steady state value. For roll & yaw channels (rate command models), dashed line represents area under pulse input.

(the forward flight problem) or one that is progressing slowly (open-loop or unity-feedback initial configuration), was determined in the course of this research:

- Rather than first defining specifications and then solving with respect to the entire family of specifications, implement each specification, one at a time, and solve the feasibility problem in a sequential manner. As the solution to the problem with respect to one spec is achieved, add the second spec to the family of constraints and solve with respect to the two specs. Once the solution with respect to two specs is achieved, add the third spec, and so on. This process is carried out until all specs are implemented and a feasible solution is achieved.

An example of an optimal controller, achieved through the use of this method, is shown in Figure (5.9). This optimal controller has comparatively robust pitch channel bandwidth, pitch-to-roll and roll-to-pitch coupling and reasonable wind-gust responses. The fact that this controller is achieved with reasonably low values for the design parameters Mphi and Kp_r may be explained by the increased roll rate feedback gain, Kp. Notice that as Kp increases and Mphi decreases (from Figure (5.6) to (5.7) to (5.9)) the roll channel actuator energy decreases, the oscillations in the wind-gust response improve, but the channel phase margins generally decrease.

Three types of trade-off studies were performed to determine the limits of the ADOCS design when applied to a forward flight condition. The first trade-off study was an attempt to push the helicopter performance as close as possible to the ultra-high performance level defined by the Level 1 Air Combat requirements. The second and third trade-off studies deal with simulated changes to the helicopter's actuator parameters. In the second study, the saturation limits are

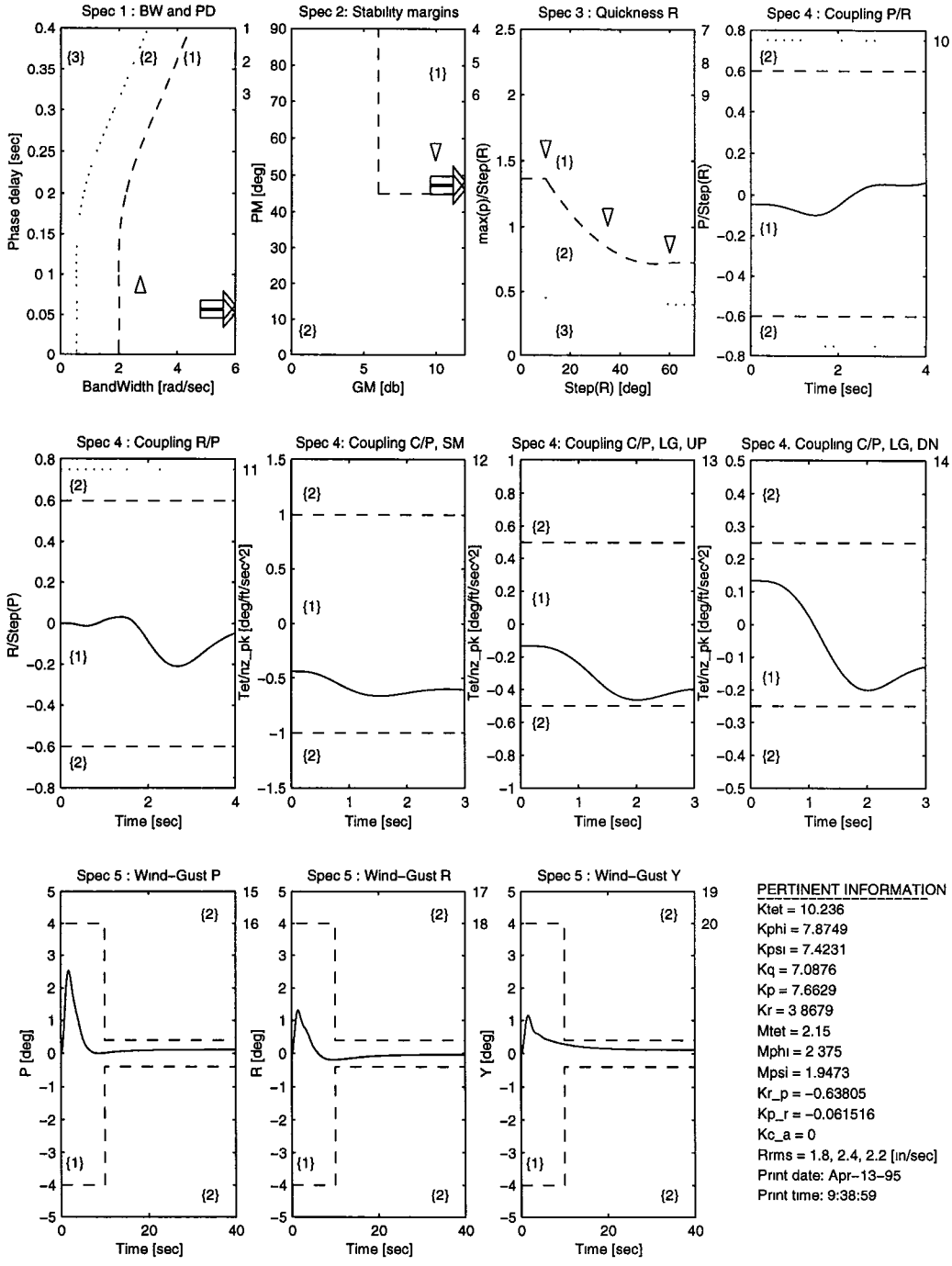

Figure 5.9: Performance map for the optimal forward flight controller achieved using sequential addition of specifications. Notice the design parameter $K_{\theta/c}$, represented as Kc_a, was not changed from its original setting of 0. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

varied from 50 % to 200 % of their nominal values through use of a multiplying factor $\mu \in [0.5\ 2.0]$. The variable μ corresponds to the declaration LIMIT in **init.m**. The third study deals with varying the actuator time constant. The multiplying factor $\nu \in [0.5\ 2.0]$ is used to manipulate the time constant. The variable ν corresponds to the declaration TIME in **init.m**. As with the hover problem, the trade-off criterion was the optimization objective.

Performance to Specification Trade-Off Study: For this trade-off study, an ultra-high performance level (Level 1 - 1) was defined based on the Forward Flight Air Combat requirements. Recall from ADS-33C that Spec 1 (Bandwidth & Phase Delay), Spec 3 (Quickness) and Spec 4 (Pitch-to-Roll and Roll-to-Pitch Coupling) are the only specs affected by the Air Combat requirements. The Air Combat requirement is simply a scalar factor of the Level 1 requirement. That is, each of the IMC-Divided Attention level curves need only be multiplied by a scalar factor to extend it to the Air Combat level curve. This makes quantification of intermediate levels a simple matter of adding a percentage of the difference to the Level 1 curve. A performance level between Level 1 and the ultra-high performance Level 1 - 1 is denoted by Level 1 - #, where $\# \in [0\ 1]$ represents the percentage difference added to the Level 1 curve. Simply stated, 1 - 0.5 represents performance halfway between Level 1 and Level 1 - 1. Note that Level 1 - 1 $\neq$ Level 0. See Figure (5.10) and [9] for more details. The first attempt to push the performance to a higher level was with $\# = 0.5$. The trade-off study was initiated from the optimal solution to the forward flight problem, as indicated in Table 5.1. Once the LEVEL variable in **init.m** is changed from 0 to 0.5, the level boundaries on Specs 1, 3 & 4 increase, as can be seen in Figure (5.10). An optimal Level 1 - 0.5 solution was achieved without much undue

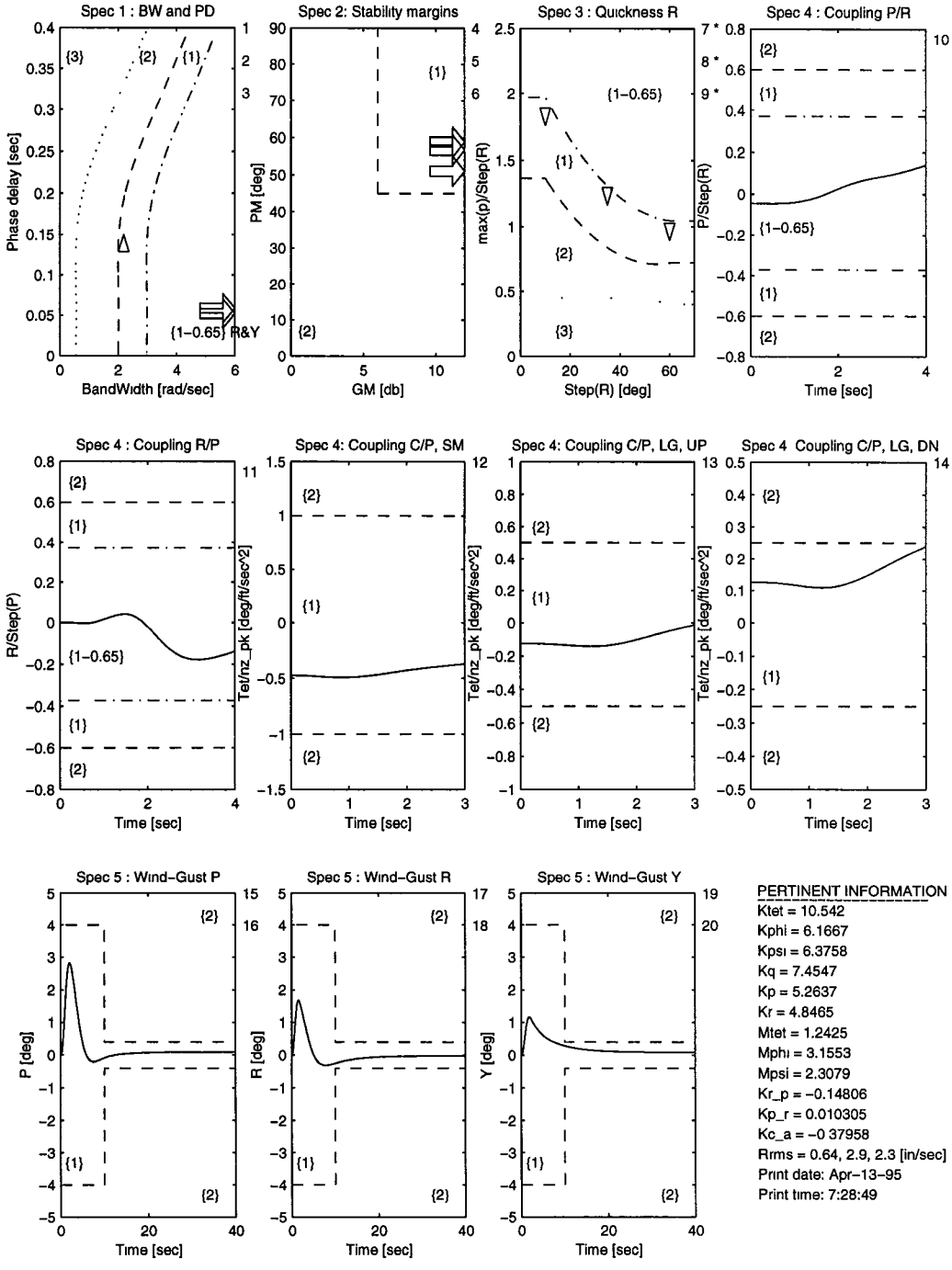

Figure 5.10: Performance map for the initial Level 1 - 0.65 design. $\triangle$ - pitch, $\triangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

hardship after approximately 10 iterations. Rather than somewhat randomly choosing a value between 0.5 and 1.0 for the second attempt, one dimensional analysis was performed. Experience has shown that the quickness requirement generally limits higher level performance. Thus, using the design parameters from the optimal Level 1 - 0.5 solution as a local solution, the design parameter directly related to the roll channel quickness (α_ϕ) was varied over the range [0 5] to see what affect it had on the quickness. The results of this one dimensional analysis are presented in Figure (5.11). From this you can clearly see that the maximum achievable quickness ratio, dependent only on α_ϕ, falls just below Level 1 - 0.7. The limiting quickness ratio is clearly the 10° command input response in subplot a. Based on the information gained from the one-dimensional analysis, the target performance Level 1 - 0.65 was chosen. After several iterations and much operator intervention, the optimal Level 1 - 0.65 solution was achieved, as shown in Figure (5.12).

Performance to Hardware Trade-Off Study - Saturation Limits: The actuator rate and command limits were considered as trade-off parameters for this study. The factor $\mu \in [0.5\ 2.0]$ was used as a scalar multiplier to either increase ($\mu > 1$) or decrease ($\mu < 1$) the actuator rate and command limits. The limits are all changed by the same factor and all at the same time. The variable μ represents the declaration LIMIT in **init.m**. The interesting objective of this study is to determine how low μ can be taken such that an optimal design is still achievable. Reference [9] indicates that designs for $\mu \in [0.75\ 2.0]$ are almost the same as for $\mu = 1$. This was verified for the forward flight problem for $\mu = 2$. Only slightly lower actuator energies are achieved. Again, turning first to one dimensional analysis of the quickness measurement with respect to α_ϕ, a target

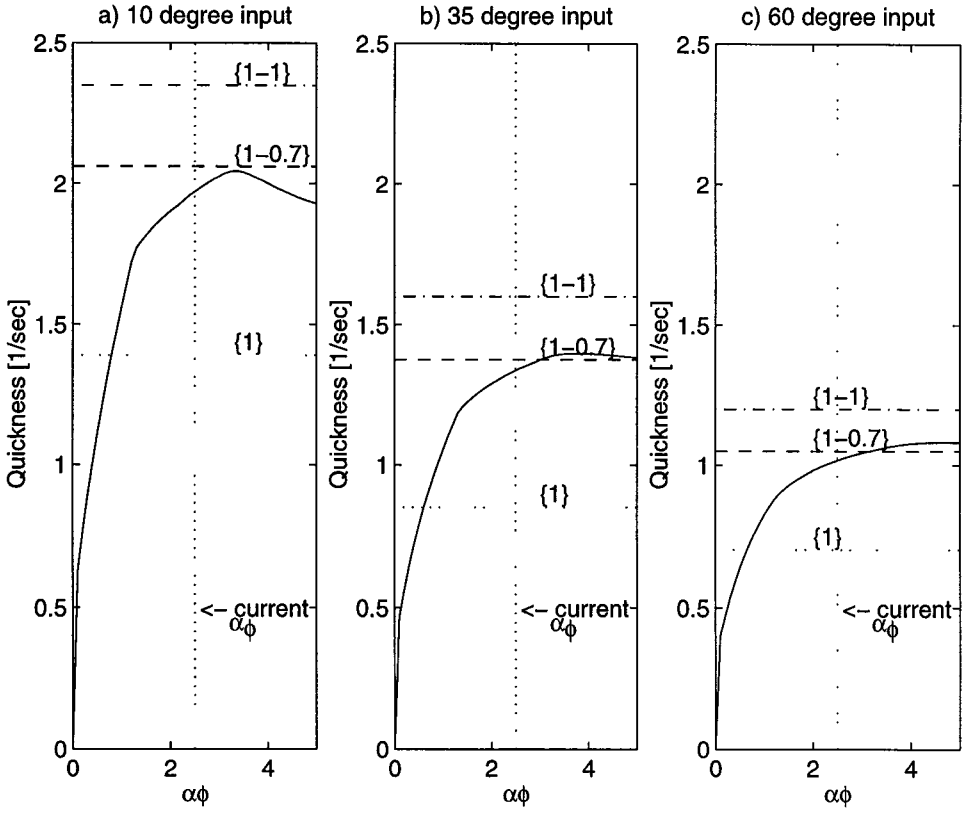

Figure 5.11: One dimensional analysis of the quickness ratio along the α_ϕ design parameter axis for performance level trade-off studies.

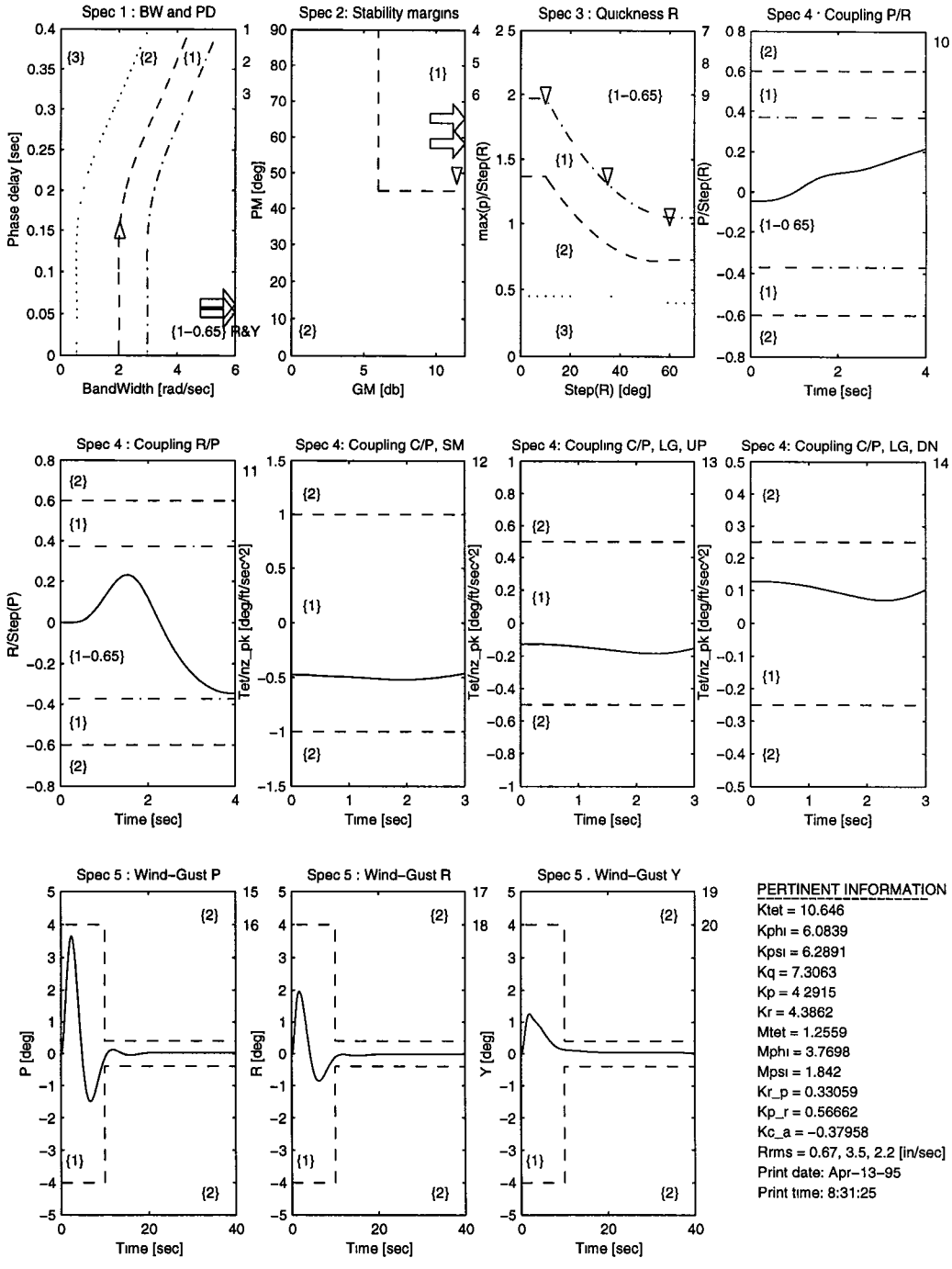

Figure 5.12: Performance map for the optimal Level 1 - 0.65 solution. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

minimum admissible limit factor is $\mu = 0.5$. The one dimensional analysis is shown in Figure (5.13). Although the feasible set of α_ϕ for $\mu = 0.40$ is not empty, the 60° quickness leaves very little room for α_ϕ to increase. That coupled with the fact that this is one dimensional analysis steers the designer toward choosing a target $\mu = 0.50$. As C-O progressed toward a feasible solution, more one dimensional analysis confirmed the target choice of $\mu = 0.50$ as the minimum admissible limits factor. The optimal solution is presented in Figure (5.14).

Performance to Hardware Trade-Off Study - Time Constant: Here the trade-off parameters are the actuator time constants. The nominal time constant value is 25 ms. Using the factor $\nu \in [0.5\ 2]$ as a scalar multiplier, faster ($\nu \in [0.5\ 1)$) or slower ($\nu \in (1\ 2]$) actuators can be simulated. Recall from [9] that below a certain time constant, the faster actuators begin to act almost as relays. This relay-like activity, though not detrimental to Level 1 helicopter performance, greatly increases the actuator energy measurements (objective functions) which limits the optimal solution. To determine the point at which this occurs, one dimensional analysis, with a specific set of design parameters, was performed with respect to the actuator rates as a function of time constant (a function of ν). As seen in Figure (5.15), the relay-like activity dies out for $\nu > 0.725$. After several optimization - one dimensional analysis iterations, each time using the newly optimized design parameters, it was determined that the minimum admissible time constant factor, ν, was 0.675. See Figure (5.16). This time constant factor allowed a low energy solution whose actuators did not exhibit relay-like behavior.

The lower limit on the time constant factor is determined by the Nyquist sampling rate [5]. To avoid aliasing, the sampling frequency, Ω_s, and the Nyquist

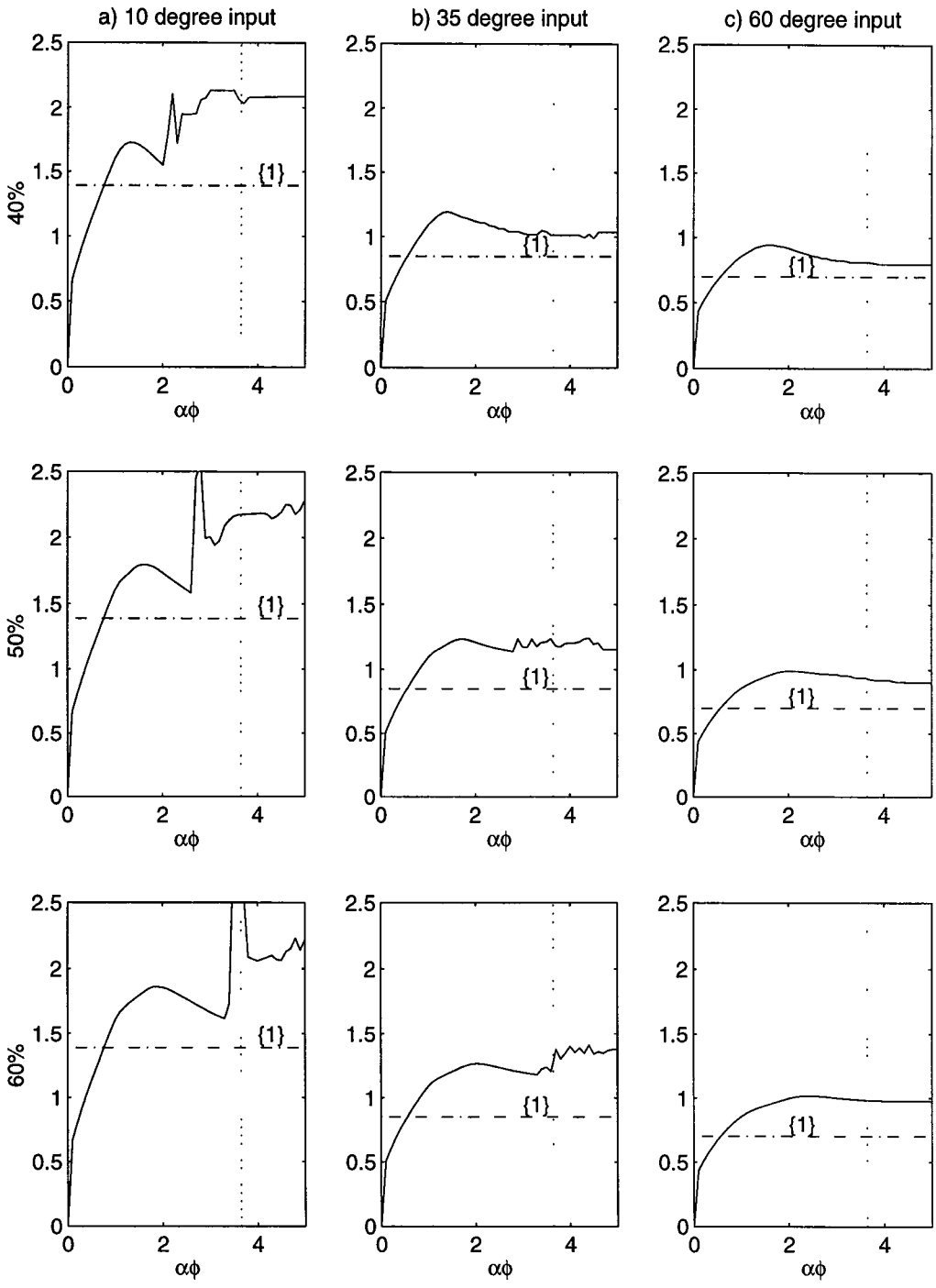

Figure 5.13: One dimensional analysis of the quickness ratio along the α_ϕ design parameter axis for actuator limit trade-off studies.

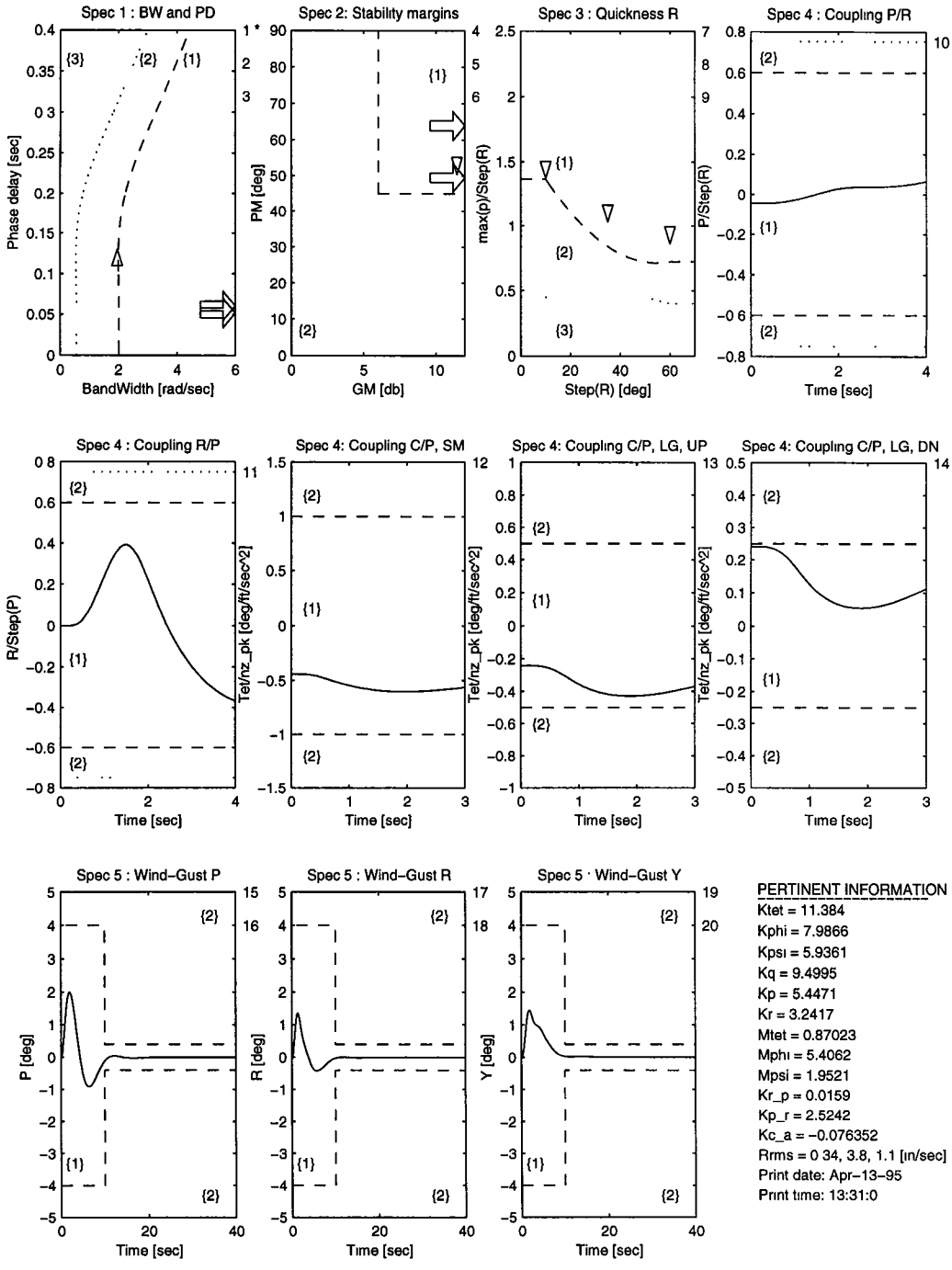

Figure 5.14: Performance map for the optimal design for $\mu = 0.50$. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

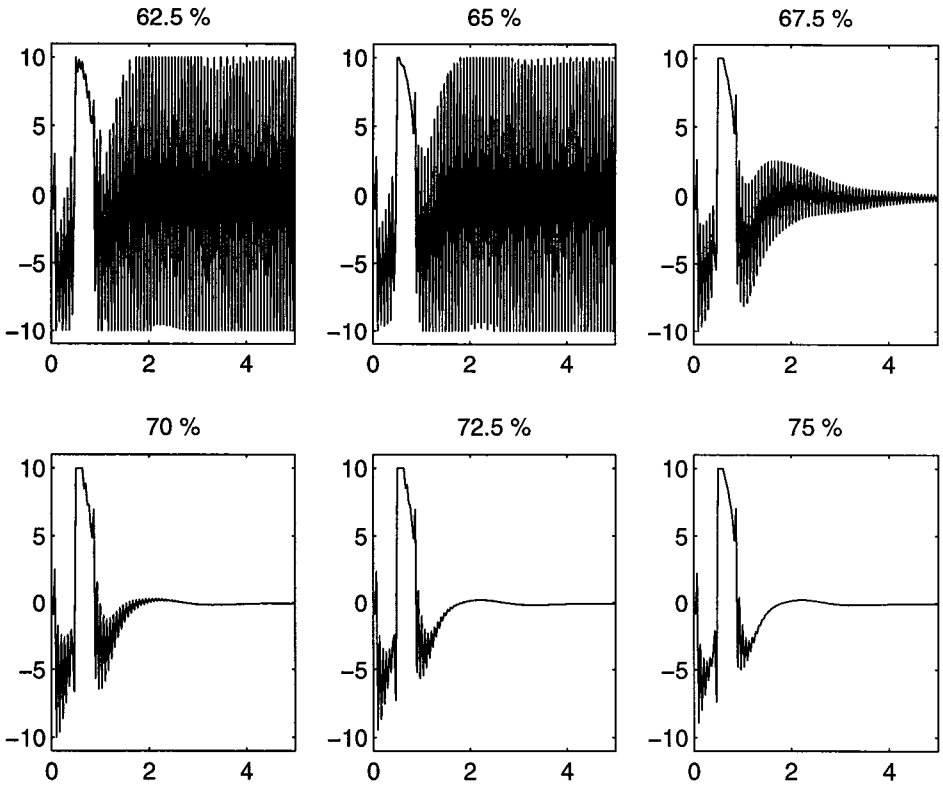

Figure 5.15: One dimensional analysis of actuator effort as a function of time constant.

rate, Ω_N, must be such that

$$\Omega_s = \frac{2\pi}{T} > 2\Omega_N \tag{5.1}$$

The ADOCS control structure uses a fixed sampling rate of 30 Hz, which sets $T = \triangle t = 0.0333$. Thus, for frequencies greater than $\pi/T = 94.25$ rad/sec, aliasing occurs. 67.5% of the nominal 25 ms time constant imparts a natural frequency on the discrete time ADOCS system of exactly 94.25 rad/sec, the aliasing frequency. Any actuator with time constant lower than or equal to 67.5% of the nominal 25 ms value introduces an equivalent s-plane natural frequency in excess of 94.25 rad/sec and causes aliasing.

A slower design with $\nu = 2$ was also completed without significant difficulty. In fact, after five iterations, a feasible controller was achieved. An optimal solution, with lower actuator energies than the nominal optimal solutions, was achieved. The optimal controller is presented in Figure (5.17).

Table 5.1 provides a summary of the initial, feasible and, where applicable, optimal design parameter configurations for each of the designs discussed in this chapter.

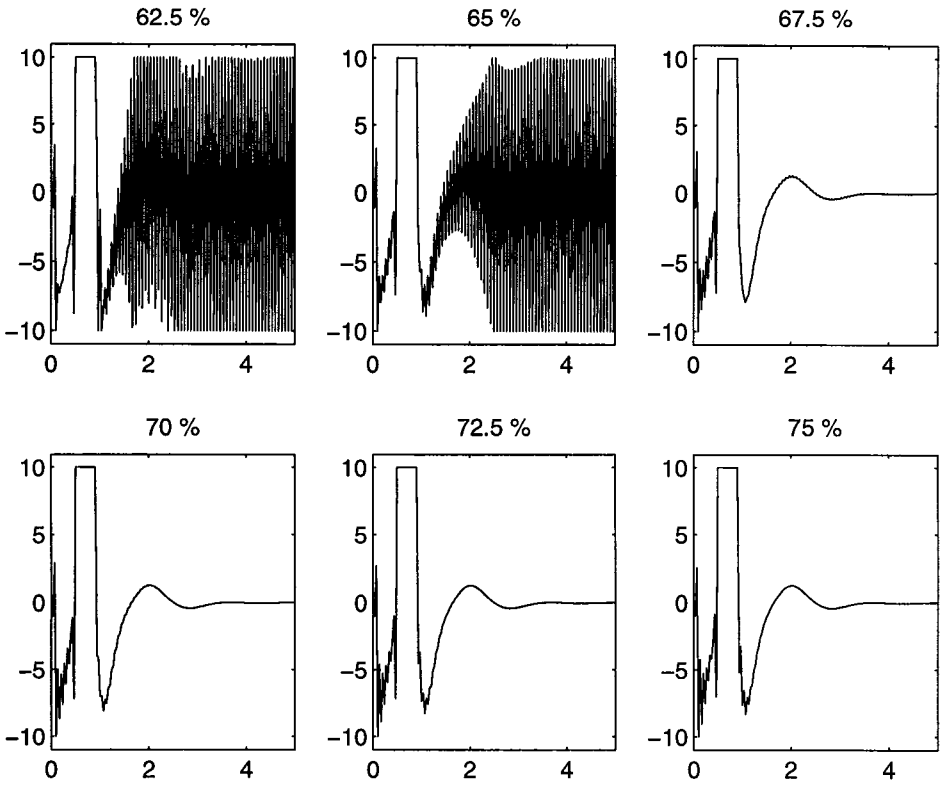

Figure 5.16: Further one dimensional analysis of actuator effort as a function of time constant.

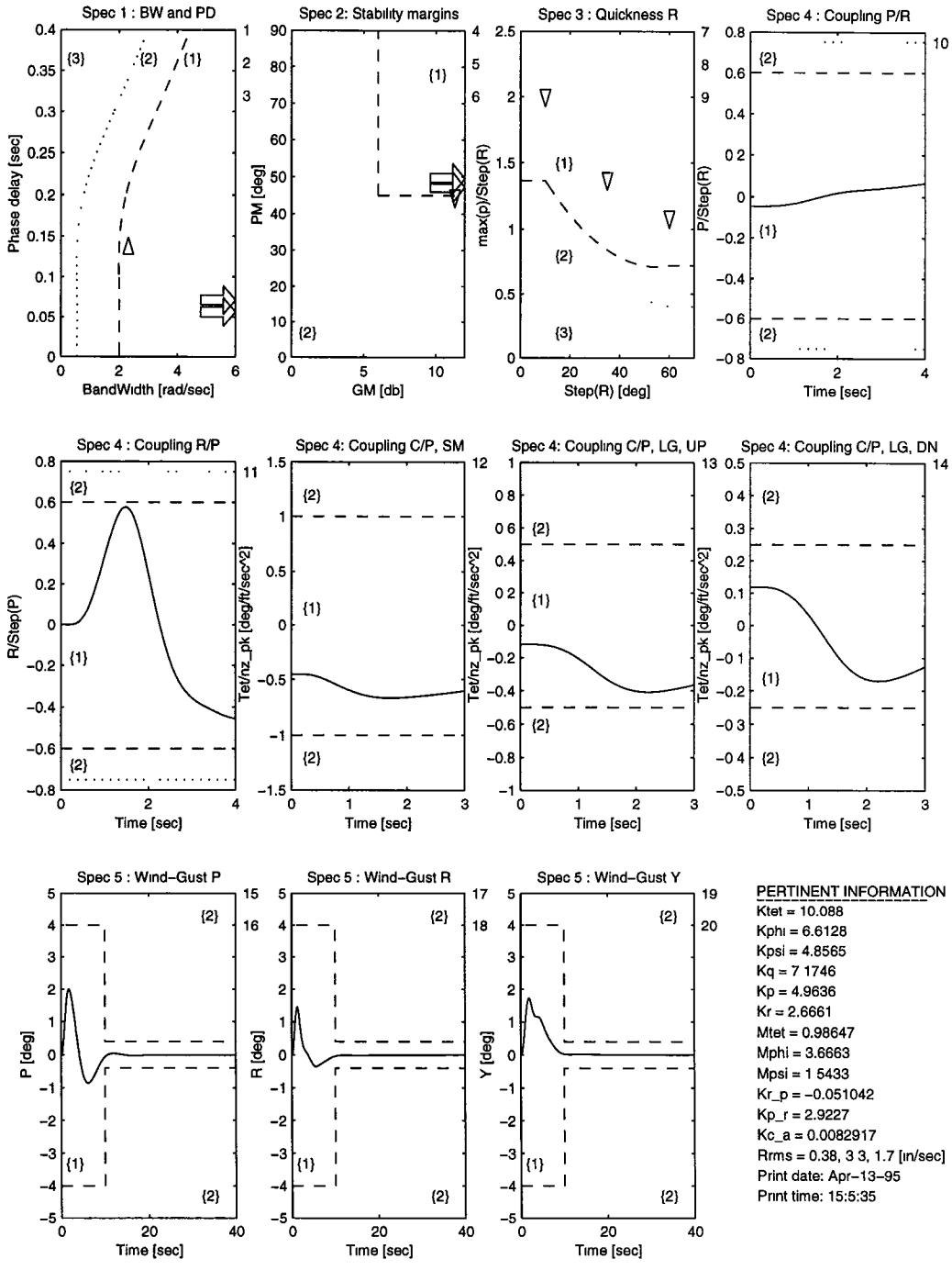

Figure 5.17: Performance map for the optimal design for $\nu = 2$. $\triangle$ - pitch, $\bigtriangledown$ - roll, $\diamond$ - yaw, $*$ - not in the desired level.

Flight Condition	Level	Initial DP's	Feasible Soln	Optimal Soln
Hover	1	TISCH	F-TISCH	O-TISCH
Fwd	1	F-TISCH, $K_{cf} = I$	Fig (5.4)	Fig (5.6)
Fwd	1	F-TISCH	Fig (5.5)	Fig (5.7)
Fwd	1	O-LOOP	–	–
Fwd	1	UNIT-FB	–	–
Fwd	1	SEQ	F-SEQ	Fig (5.9)
Fwd	1 - 0.65	Fig(5.7)	–	Fig(5.12)
Fwd	SAT T/O	Fig (5.5)	–	Fig(5.14)
Fwd	TC T/O	Fig (5.5)	–	Fig (5.17)

Table 5.1: Summary of forward flight feasible, optimal and trade-off solution design parameters and their initial design parameter configurations. TISCH represents the Current flight value design parameters from [7], 1 - 0.65 represents the performance level trade-off study, SAT T/O represents actuator saturation limits trade-off study and TC T/O represents actuator time constant trade-off study.

Chapter 6

Conclusions

Multicriterion parametric optimization has been successfully demonstrated as a technique for meeting rotorcraft forward flight handling qualities requirements. Model-based simulation, using SIMULINK, has been shown to be an intuitive and efficient alternative to strict MATLAB based simulation. Several feasible and optimal forward flight controllers were presented. The lack of a unique controller configuration, either feasible or optimal, demonstrates that different controllers can be achieved by varying the initial design parameter configurations. Designers may be able to more precisely steer the design toward a desired objective as familiarity with the design method, and the effects of different initial design parameter configurations, increases.

Three optimal controller designs, derived from three Level 1 feasible designs, were presented. The best optimal design, based on lowest actuator energy, was derived from the feasible hover solution design parameters with the crossfeed gains intact. The performance map for this optimal controller is presented in Figure (5.7). Consider also the optimal controller design presented in Figure (5.9), derived from the sequential addition of specifications to the forward flight problem. Notice that although the actuator energy is higher, none of the con-

straints are on the Level 1 boundary, indicating a certain robustness to slight variations in design parameters, and the design parameter $K_{\theta/c}$ is not used. Consequently, this controller design may be considered optimal because of the robustness and potential cost reduction (although slight) from not having to implement the crossfeed gain $K_{\theta/c}$.

Trade-off study results were generally consistent with those determined for the hover solution.

- Ultra-high performance can not be achieved with the current ADOCS controller structure. The design was limited, by the quickness specification, to Level 1 - 0.65. Improving the actuators may enable Level 1 - 1 performance.

- A minimum admissible saturation limit of 50%, limited by system quickness, was determined.

- Increasing actuator saturation levels beyond the nominal values does not improve performance other than to slightly reduce actuator energy.

- An upper limit on actuator speed, based on the actuator time constant, was determined. ADOCS control structures with quicker actuators, to 67.5% of the nominal (25 ms) time constant, provided low actuator energy optimal solutions to the forward flight problem. Below 67.5% of the nominal time constant, the actuators exhibited relay-like behavior, due to aliasing, which drove actuator energies exceptionally high.

- Lower than nominal actuator energy optimal solutions were achieved with slower, up to 50 ms time constant, actuators.

The hover controller design method, coupled with the 80 knot forward flight

design method, provide a framework which can be used to derive a design parameter map at 10 or 20 knot intervals which can be used to schedule controller configurations with respect to velocity. The generality of the model-based simulation lends itself to this task. With the appropriate velocity based helicopter models, feasible solutions at 10 or 20 knot intervals can be determined by simply adding the new helicopter dynamics to the model at each interval and running the optimization to achieve a feasible design.

Future design methodologies may include a robust optimal requirement. This will serve to limit the optimal solution and ensure Level 1 performance in the presence of design parameter perturbations. As the design technique is enhanced to include such constraints, the importance of the use of one dimensional analysis tools will increase. Tools which allow the one dimensional analysis of specifications with respect to the design parameters greatly increase the designer's efficiency and ability to understand the limits of the design. As the design technique is modified, these tools should be created to meet the needs of the design.

Appendix A

Hover ADOCS Code

Matlab m-file init.m for Hover ADOCS via SIMULINK

```
% ****************************************************************
%   System Initialization File for ADOCS
%   init.m CONSOL-MATLAB file
%   Simulink Model 'ctest' for spec 1 (Bandwidth-Phase Delay)
%   and spec 2 (Stability margins), 'dtest' for spec 3 (Quickness)
%   and spec 4 (Coupling), and 'dwtest1' for spec 5 (Wind gust)
%   Modified from original init.m by Gil Yudilevitch 04/09/94
%   P. J. Potter 11/07/94
% ****************************************************************

% Optimization Set-up
% -------------------

OPSYS  = 1; % Operating system: 0 - DOS (PC); 1 - UNIX (Workstation).
MATLAB = 0; % Matlab run type: 0 - C-O/MATLAB; 1 - autonomous MATLAB.
CONSOL = 0; % C-O run type: 0 - Background; 1 - interactive.
LEVEL  = 0; % Performance level (0,1): 0 - Level 1, 1 - Level 1-.
LIMIT  = 1; % Actuator limits: actual_limit = LIMIT*nominal_limit.
TIME   = 1; % Actuator time const. (TC): actual_TC = TIME*nominal_TC.
SPEC1  = 1; % Specs: 0 - Specs not included; 1 - Specs are included.
SPEC2  = 1;
SPEC3  = 1;
SPEC4  = 1;
SPEC5  = 1;
FLIGHT = 'h';
              %                                      f
              % Flight conditions (about hover)   l h r 10 kts
              %                                      b
              % Forward flight condition           f80
```

```
% General Information
% -------------------

dt = 1/30;                                      % Sampling rate 30 Hz
tf = 5; t =0:dt:tf; nt=length(t);               % Command Type Time Axis
t1= min(find(t==1));
t3= min(find(t==3));                            % Index for 1 & 3 sec.
tfg=40;
tg=0:dt:tfg;                                    % Wind-gust time vector
n10=max(find(tg<=10)); nfg=length(tg);
wo=logspace(-1,1,50);                           % Frequency vector
s=0;  % Counter for tracking dp's
Ktetv=[]; Kphiv=[]; Kpsi=[]; Kqv=[];            % Initial dp vectors
Kpv=[]; Krv=[]; Mtetv=[]; Mphiv=[];
Mpsiv=[]; Kr_pv=[]; Kp_rv=[]; Kc_yv=[];

%  Model Components
%  ----------------
%  1. Helicopter 6 DOF Dynamics + Rotor Aerodynamics
%     Linear Continuous Time Model

%  # of states    = 11 (9 for 6 DOF dynamics + 2 for rotor flapping)
%  # of inputs    = 4
%  # of outputs   = 7  ( 4 for the linear case --> C 4 x 11 )

%                                          State Output Input
%                                          ----- ------ -----
%  u     - linear velocity along X axis       1     1
%  v     - linear velocity along Y axis       2     2
%  w     - linear velocity along Z axis       3     3      4 del_c
%  p     - angular velocity about X axis      4     4
%  q     - angular velocity about Y axis      5     5
%  r     - angular velocity about Z axis      6     6
%  phi   - angle about X axis                 7     7      2 del_phi
%  theta - angle about Y axis                 8     8      1 del_tet
%  psi   - angle about Z axis                 9     9      3 del_psi
%  b1c   - longitudinal flap  (-a1s)         10
%  b1s   - lateral flap       (-b1s)         11
%
%  Units:  linear velocity  [ft /sec]
%  -----   angular velocity [rad/sec]
%          helicopter angle [rad    ]
%          flap angle       [rad    ]
%          del_#            [in     ]
```

```
B10to4 = zeros(10,4);          % Trans. 10 (UMGENHEL) --> 4 inputs
B10to4(1,2) = 1; B10to4(2,1) = 1; B10to4(4,3) = 1; B10to4(3,4) = 1;

% Continuous Time
% ---------------

if OPSYS == 1,
  eval(['load A11',FLIGHT,' -ascii;']); % Load A,B UMGENHEL matrices
  eval(['load B11',FLIGHT,' -ascii;']); % for Unix for the FLIGHT
  eval(['Ap = A11',FLIGHT,';']);        % conditions
  eval(['Bp = B11',FLIGHT,'*B10to4;']);
else,
  eval(['load a11',FLIGHT,'.;']);       % Load A,B UMGENHEL matrices
  eval(['load b11',FLIGHT,'.;']);       % for PC for the FLIGHT
  eval(['Ap = a11',FLIGHT,';']);        % conditions
  eval(['Bp = b11',FLIGHT,'*B10to4;']);
end;

%                                  Output
%State-> 1 2 3 4 5 6 7 8 9 10 11    \/
Cp =   [ 1 0 0 0 0 0 0 0 0 0  0  % u, longitudinal velocity
         0 1 0 0 0 0 0 0 0 0  0  % v, lateral velocity
         0 0 1 0 0 0 0 0 0 0  0  % w, vertical velocity
         0 0 0 1 0 0 0 0 0 0  0  % p, roll rate
         0 0 0 0 1 0 0 0 0 0  0  % q, pitch rate
         0 0 0 0 0 1 0 0 0 0  0  % r, yaw rate
         0 0 0 0 0 0 1 0 0 0  0  % phi, roll angle
         0 0 0 0 0 0 0 1 0 0  0  % theta, pitch angle
         0 0 0 0 0 0 0 0 1 0  0];% psi, yaw angle

Dp=zeros(9,4);

[Apm,Bpm,Cpm,Dpm]=modred(Ap,Bp,Cp,Dp,[10,11]);  % Reduced to 9 states
                                                % for inverse plant
                                                % parameters
% Discrete Time
% -------------

[Apd,Bpd,Cpd,Dpd]=c2dm(Ap,Bp,Cp,Dp,dt,'tustin');

% Pure Delay 2_nd Order Pade Approximation
% ----------------------------------------
%
% Total Delay was taken equal for all channels. Total delay
% from phoncon with Mark 10/94 is 100 ms ==> 75 ms "pure" and
% 25 ms Actuator Time-Constant (see later in SP-Actuator Model)
```

```
Td = [0.075;0.075;0.075;0.075];
[Ad1,Bd1,Cd1,Dd1] = pade(Td(1),2);
[Ad2,Bd2,Cd2,Dd2] = pade(Td(2),2);
[Ad3,Bd3,Cd3,Dd3] = pade(Td(3),2);
[Ad4,Bd4,Cd4,Dd4] = pade(Td(4),2);
[Ad,Bd,Cd,Dd]=append(Ad1,Bd1,Cd1,Dd1,Ad2,Bd2,Cd2,Dd2);
[Ad,Bd,Cd,Dd]=append(Ad,Bd,Cd,Dd,Ad3,Bd3,Cd3,Dd3);
[Ad,Bd,Cd,Dd]=append(Ad,Bd,Cd,Dd,Ad4,Bd4,Cd4,Dd4);

[Add,Bdd,Cdd,Ddd]=c2dm(Ad,Bd,Cd,Dd,dt,'tustin');

% Swashplate-Actuators Model
% --------------------------

% LINEAR PART :-

K   =  1/(TIME*0.025)*ones(1,4); % 1/(Time Constant) equal actuators
[as1,bs1,cs1,ds1] = tf2ss([0,K(1)],[1,K(1)]);
[as2,bs2,cs2,ds2] = tf2ss([0,K(2)],[1,K(2)]);
[as3,bs3,cs3,ds3] = tf2ss([0,K(3)],[1,K(3)]);
[as4,bs4,cs4,ds4] = tf2ss([0,K(4)],[1,K(4)]);
[As,Bs,Cs,Ds] = append(as1,bs1,cs1,ds1,as2,bs2,cs2,ds2);
[As,Bs,Cs,Ds] = append(As,Bs,Cs,Ds,as3,bs3,cs3,ds3);
[As,Bs,Cs,Ds] = append(As,Bs,Cs,Ds,as4,bs4,cs4,ds4);

% NONLINEAR SATURATIONS :-

% * All data is given at cockpit units (i.e., in's of stick or pedal)
% * The trim (hover) conditions set to be zero (see table below)
% * The swashplate mechanism ("limits coupling") is NOT accounted for
% * The rate limits are taken as: full stroke per 1 sec

% Command Limits Table, Ref. Sikorsky SER 70452 p 6.16 Fig. 6.3.1
% ..............................................................
% chn |     Pitch     |     Roll      |       Yaw       |   Collective  |
% act |-5.0 +0.2 +5.0|-5.0 -0.5 +5.0|-2.69 +0.85 +2.69| 0.0 +5.0 +10.0|
% sim |-5.2  0.0 +4.8|-4.5  0.0 +5.5|-3.54  0.00 +1.84|-5.0  0.0 + 5.0|

Csl = LIMIT*[ -5.2 -4.5 -3.54  -5];  % Displacement Lower Limits
Csu = LIMIT*[  4.8  5.5  1.84   5];  % Displacement Upper Limits
Rsl = LIMIT*[-10  -10   -5.38 -10];  % Rate Lower Limits
Rsu = LIMIT*[ 10   10    5.38  10];  % Rate Upper Limits

% Linear Continuous-Time Open-Loop
% --------------------------------
```

```
[Ao,Bo,Co,Do] = series(Ad,Bd,Cd,Dd,As,Bs,Cs,Ds); % Used for calc of
[Ao,Bo,Co,Do] = series(Ao,Bo,Co,Do,Ap,Bp,Cp,Dp); % eigen_A in simu.m

% Inverse plant (ff) constants
% ----------------------------

FGtet =  Bpm(5,1); FGphi =  Bpm(4,2); FGpsi =  Bpm(6,3);
FTtet = -Apm(5,5); FTphi = -Apm(4,4); FTpsi = -Apm(6,6);

% Time Response Inputs
% --------------------

DtR  = pi/180;
Stet = DtR*[ 5,17.5,30];
Sphi =-DtR*[10,35  ,60]; % Worst direction (nonsymmetric saturation)
Spsi = DtR*[10,35  ,60];
Scol = 5;

% Input Sequences for Simulation Time Histories
% ---------------------------------------------

aRps=pilot(t,Stet(3),Stet(1),'p');
aRrs=pilot(t,Sphi(3),Sphi(1),'p');
aRys=pilot(t,Spsi(3),Spsi(1),'r');

aRpm=pilot(t,Stet(3),Stet(2),'p');
aRrm=pilot(t,Sphi(3),Sphi(2),'p');
aRym=pilot(t,Spsi(3),Spsi(2),'r');

aRpl=pilot(t,Stet(3),Stet(3),'p');
aRrl=pilot(t,Sphi(3),Sphi(3),'p');
aRyl=pilot(t,Spsi(3),Spsi(3),'r');

aRc=pilot(t,Scol,Scol,'p');

% Linear Wind-Gust Model
% ----------------------

[ag,bg,cg,dg] = tf2ss([0 0.44 0],[1 2 1]);
[Ag,Bg,Cg,Dg]=append(ag,bg,cg,dg,ag,bg,cg,dg);
[Ag,Bg,Cg,Dg]=append(Ag,Bg,Cg,Dg,ag,bg,cg,dg);
[Ag,Bg,Cg,Dg]=append(Ag,Bg,Cg,Dg,[],[],[], 0);

[Agd,Bgd,Cgd,Dgd]=c2dm(Ag,Bg,Cg,Dg,dt,'tustin');
```

```
Bgust=[0      0      0      0
       0      0      0      0
       0      0      0      0
       0      FTphi  0      0
       FTtet  0      0      0
       0      0      FTpsi  0
       0      0      0      0
       0      0      0      0
       0      0      0      0
       0      0      0      0
       0      0      0      0];
Bgust=[Bgust Bp];
Dgust=[zeros(9,4) Dp];

[Apg,Bpg,Cpg,Dpg]=c2dm(Ap,Bgust,Cp,Dgust,dt,'tustin');

% Level 1 curve for spec 1
% ------------------------

Ts = [0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200 ...
      0.225 0.250 0.275 0.300 0.325 0.350 0.375 0.400];
Ws = [2.005 2.005 2.005 2.005 2.005 2.005 2.005 2.105 2.235 ...
      2.405 2.685 3.005 3.275 3.505 3.905 4.155 4.455];
PL1=polyfit(Ts,Ws,6);
Ws = Ws + 1.5*LEVEL;
PL =polyfit(Ts,Ws,6);

% LEVEL factors for spec 3
% ------------------------

d3p1 = 0.93; d3p2 = 0.78; d3p3 = 0.68;
d3r1 = 0.96; d3r2 = 0.50; d3r3 = 0.10;
d3y1 = 0.98; d3y2 = 0.76; d3y3 = 0.64;

% Switch variables for ctest.m
% ----------------------------

delopn=1; %Feedforward dynamics shutoff variable
deltet=1; %Pitch open loop calculation variable
delphi=1; % Roll open loop calculation variable
delpsi=1; %  Yaw open loop calculation variable

% Optimization Parameters (optional for interactive C-O/MATLAB run)
% -----------------------

if MATLAB == 0 & CONSOL == 1,
```

```
  CONS =[1:3,2,4,6,8:16,1,3,17,5:2:17];
  BAD  =[3;3;3;0.002;0.002;0.002;0.01;0.01;0.01;0.04;0.03;0.09;0.08;...
         0.07;0.04;0.05;0.05;0.002;0.5;0.3;0.03;0.3;0.03;0.3;0.03];
  Iter=-1; iter=[]; witer=[]; www=[]; kkk=[];
  perf=[]; dps=[]; grad=[]; bad=BAD;
end;

% SIMU.MAT saved variables, default values
% ----------------------------------------

pitch_dist=0; roll_dist=0; yaw_dist=0; pitch_bw  =0; roll_bw  =0;
pitch_pd  =0; roll_pd  =0; yaw_pd  =0; pitch_damp=0; roll_damp=0;
pitch_r1s =0; roll_r1s =0; yaw_r1s =0; pitch_r2s =0; roll_r2s =0;
pitch_r3s =0; roll_r3s =0; yaw_r3s =0; pitch_d   =0; roll_d   =0;
yaw_d2    =0; pitch_g  =0; roll_g  =0; yaw_g     =0; eigen_A  =0;
pitch_Wco =0; roll_Wco =0; yaw_Wco =0; roll_AE   =0; yaw_AE   =0;
yaw_bw    =0; yaw_damp =0; yaw_r2s =0; yaw_d1    =0; pitch_AE =0;

%*****************************end of init.m ****************************
```

Matlab m-file simu.m for Hover ADOCS via SIMULINK

```
% ************************************************************************
%   System Simulation File for ADOCS
%   simu.m CONSOL-MATLAB file
%   Simulink Model 'ctest' for spec 1 (Bandwidth-Phase
%   delay) and spec 2 (Stability margins), 'dtest' for
%   spec 3 (Quickness) and spec 4 (Coupling), and 'dwtest1' for spec 5
%   (Wind gust)
%   Modified from original simu.m by Gil Yudilevitch 04/09/94
%   P. J. Potter 11/07/94
% ************************************************************************
%========================================================================

% Feedforward (including Model Following & "Inverse Closed-Loop")
% ---------------------------------------------------------------

% Continuous Time
% ---------------

[Af_tet,Bf_tet,Cf_tet,Df_tet] = ...
tf2ss(Mtet^2*[1/FGtet Kq+1/FGtet/FTtet Ktet],[1 2*Mtet Mtet^2]);
[Af_phi,Bf_phi,Cf_phi,Df_phi] = ...
tf2ss(Mphi^2*[1/FGphi Kp+1/FGphi/FTphi Kphi],[1 2*Mphi Mphi^2]);
[Af_psi,Bf_psi,Cf_psi,Df_psi] = ...
tf2ss(Mpsi*[1/FGpsi Kr+1/FGpsi/FTpsi Kpsi],[1   Mpsi      0]);
```

```
[Af,Bf,Cf,Df] = ...
append(Af_tet,Bf_tet,Cf_tet,Df_tet,Af_phi,Bf_phi,Cf_phi,Df_phi);
[Af,Bf,Cf,Df] = append(Af,Bf,Cf,Df,Af_psi,Bf_psi,Cf_psi,Df_psi);
[Af,Bf,Cf,Df] = append(Af,Bf,Cf,Df,[],[],[],1);

% Discrete Time
% -------------

[Afd,Bfd,Cfd,Dfd]=c2dm(Af,Bf,Cf,Df,dt,'tustin');

% Crossfeed design
% ----------------

Kcf = [1    Kr_p 0 0
       Kp_r 1    0 0
       0    0    1 Kc_y
       0    0    0 1];

Bocf = Bo*Kcf; Docf = Do*Kcf;

% Spec 1
% ------
if SPEC1 == 1,
    % Linear Closed-Loop
    % ------------------
    % Simulink model: ctest.m (Continuous time (CT) model)
    % # states - order:
    %    11    - UH-60A Dynamics [states(1-11)]
    %     6    - Feedforward Control [states(12-17)]
    %     8    - Delay Dynamics [states(18-25)]
    %     4    - Linear Actuator Dynamics [states(26-29)]
    % Use [sizes,x0,states]=ctest([],[],[],0) to state order

%   u v w p  q  r  phi theta psi b1c b1s
H = [ ...
    0 0 0 0  Kq 0  0    Ktet 0    0   0     % del_theta
    0 0 0 Kp 0  0  Kphi 0    0    0   0     % del_phi
    0 0 0 0  0  Kr 0    0    Kpsi 0   0     % del_psi
    0 0 0 0  0  0  0    0    0    0   0];   % del_col
HH=[zeros(4,12),H];
Ac=Ao-Bocf*HH;
eigen_A=max(real(eig(Ac)));

[A1,B1,C1,D1]=linmod('ctest');

  [pitch_dist,pitch_bw,pitch_pd] = d_bw_pd(A1,B1,C1,D1,1,8,PL1);% angle
```

```
  [roll_dist ,roll_bw ,roll_pd ] = d_bw_pd(A1,B1,C1,D1,2,7,PL); % angle
  [yaw_dist  ,yaw_bw  ,yaw_pd  ] = d_bw_pd(A1,B1,C1,D1,3,6,PL); % rate
end;

% Spec 2
% ------
if SPEC2 == 1,
    % Linear Broken-Loop
    % ------------------
    % Simulink model: ctest.m (Continuous time (CT) model)
    % # states - order:
    %    11    - UH-60A Dynamics [states(1-11)]
    %     6    - Feedforward Control [states(12-17)]
    %     8    - Delay Dynamics [states(18-25)]
    %     4    - Linear Actuator Dynamics [states(26-29)]
    % Use [sizes,x0,states]=ctest([],[],[],0) to verify order of states

GMO=6; PMO=45; % Standard gain and phase margin
delopn=-1;     % Send the input around the feedforward control
deltet=-1;     % Break the pitch feedback loop
[A2p,B2p,C2p,D2p]=linmod('ctest');
[mo1,po1]=bode(A2p,B2p(:,1),C2p(10,:),D2p(10,1),1,wo);
[Gm,pitch_PM,Wg,pitch_Wco]=imargin(mo1,po1,wo);pitch_GM=20*log10(Gm);
d_g_p = (pitch_GM/GMO-1)^2;
d_p_p = (pitch_PM/PMO-1)^2;
    if pitch_GM >= GMO & pitch_PM >= PMO, pitch_damp = 0;
elseif pitch_GM >= GMO & pitch_PM <  PMO, pitch_damp = d_p_p;
elseif pitch_GM <  GMO & pitch_PM >= PMO, pitch_damp = d_g_p;
else,                                     pitch_damp = d_g_p+d_p_p;end;
deltet=1;      % Close the pitch feedback loop

delphi=-1;     % Break the roll feedback loop
[A2r,B2r,C2r,D2r]=linmod('ctest');
[mo2,po2]=bode(A2r,B2r(:,2),C2r(11,:),D2r(11,2),1,wo);
[Gm,roll_PM,Wg,roll_Wco]=imargin(mo2,po2,wo);roll_GM=20*log10(Gm);
d_g_r = (roll_GM/GMO-1)^2;
d_p_r = (roll_PM/PMO-1)^2;
    if roll_GM >= GMO & roll_PM >= PMO, roll_damp = 0;
elseif roll_GM >= GMO & roll_PM <  PMO, roll_damp = d_p_r;
elseif roll_GM <  GMO & roll_PM >= PMO, roll_damp = d_g_r;
else,                                   roll_damp = d_g_r+d_p_r; end;
delphi=1;      % Close the roll feedback loop

delpsi=-1;     % Break the yaw feedback loop
[A2y,B2y,C2y,D2y]=linmod('ctest');
[mo3,po3]=bode(A2y,B2y(:,3),C2y(12,:),D2y(12,3),1,wo);
```

```
[Gm,yaw_PM,Wg,yaw_Wco]=imargin(mo3,po3,wo);yaw_GM=20*log10(Gm);
d_g_y = (yaw_GM/GMO-1)^2;
d_p_y = (yaw_PM/PMO-1)^2;
    if yaw_GM >= GMO & yaw_PM >= PMO, yaw_damp = 0;
elseif yaw_GM >= GMO & yaw_PM <  PMO, yaw_damp = d_p_y;
elseif yaw_GM <  GMO & yaw_PM >= PMO, yaw_damp = d_g_y;
else,                                 yaw_damp = d_g_y+d_p_y; end;
delpsi=1;       % Close the yaw feedback loop
delopn=1;       % Reroute the input through the feedforward control
end;

% Spec 3 & Spec 4
% ---------------
if(SPEC3+SPEC4)>=1,
    % Nonlinear Closed-Loop Time Response
    % -----------------------------------
    % Simulink model: dtest.m (Discrete time (DT) model)
    % # states - order:
    %     4    - Actuator Dynamics [states(1-4)]
    %    11    - UH-60A Dynamics [states(5-15)]
    %     6    - Feedforward Control [states(16-21)]
    %     8    - Delay Dynamics [states(22-29)]
    % Use [sizes,x0,states]=dtest([],[],[],0) to verify state order

aR=zeros(5/dt+1,5);
[r,c]=size(aR);
aR4=zeros(r,4);

for k=1:r,
  aR(k,1)=(k-1)*dt;
end

% Pitch input
% -----------

aR(:,2)=aRps';
[t11,X11,Y11]=linsim('dtest',5,[],[],aR);

aR(:,2)=aRpm';
[t1,X12,Y12]=linsim('dtest',5,[],[],aR);

aR(:,2)=aRpl';
[t1,X13,Y13]=linsim('dtest',5,[],[],aR);

% Roll input
% ----------
```

```
aR(:,2:5)=aR4;
aR(:,3)=aRrs';
[t21,X21,Y21]=linsim('dtest',5,[],[],aR);

aR(:,3)=aRrm';
[t1,X22,Y22]=linsim('dtest',5,[],[],aR);

aR(:,3)=aRrl';
[t1,X23,Y23]=linsim('dtest',5,[],[],aR);

% Yaw input
% ---------

aR(:,2:5)=aR4;
aR(:,4)=aRys';
[t31,X31,Y31]=linsim('dtest',5,[],[],aR);

aR(:,4)=aRym';
[t1,X32,Y32]=linsim('dtest',5,[],[],aR);

aR(:,4)=aRyl';
[t1,X33,Y33]=linsim('dtest',5,[],[],aR);

% Collective input
% ----------------

aR(:,2:5)=aR4;
aR(:,5)=aRc';
[t1,X41,Y41]=linsim('dtest',5,[],[],aR);
end;

% Spec 3
% ------
if SPEC3 == 1,
  pitch_r1 = max(Y11(:,5))/Stet(1);   % [1/sec]
  pitch_r1s= pitch_r1-(0.70+d3p1*LEVEL);
  pitch_r2 = max(Y12(:,5))/Stet(2);
  pitch_r2s= pitch_r2-(0.40+d3p2*LEVEL);
  pitch_r3 = max(Y13(:,5))/Stet(3);
  pitch_r3s= pitch_r3-(0.25+d3p3*LEVEL);
  roll_r1  = min(Y21(:,4))/Sphi(1);
  roll_r1s = roll_r1-(1.39+d3r1*LEVEL);
  roll_r2  = min(Y22(:,4))/Sphi(2);
  roll_r2s = roll_r2-(0.85+d3r2*LEVEL);
  roll_r3  = min(Y23(:,4))/Sphi(3);
```

```
  roll_r3s = roll_r3-(0.75+d3r3*LEVEL);
  yaw_r1   = max(Y31(:,6))/Spsi(1);
  yaw_r1s  = yaw_r1-(1.42+d3y1*LEVEL);
  yaw_r2   = max(Y32(:,6))/Spsi(2);
  yaw_r2s  = yaw_r2-(0.67+d3y2*LEVEL);
  yaw_r3   = max(Y33(:,6))/Spsi(3);
  yaw_r3s  = yaw_r3-(0.36+d3y3*LEVEL);

% Quadratic objectives (actuator "energy")
% --------------------

pitch_AE = qcost(dt,Y11(:,10),1);
 roll_AE = qcost(dt,Y21(:,11),1);
  yaw_AE = qcost(dt,Y31(:,12),1);

end;

% Spec 4
% ------
if SPEC4 == 1,
  pitch_d = Y23(1:121,8)/Sphi(3); %Pitch angle out/Roll channel in
  roll_d  = Y13(1:121,7)/Stet(3); %Roll angle out/Pitch channel in
  yaw_d1 = 180/pi*Y41(t3:nt,6); % Yaw rate output
   Mr1 = max(Y41(:,6)); Jr1 = find(Y41(:,6) == Mr1); %for w(t) < 0
   if t(Jr1) >= t3, r1 =  Y41(t1,6); else, r1 = Mr1; end; %& r(t) > 0
     % r1 > 0 for this collective input
     r3 = Y41(t3,6)-r1;
     % altitude rate response, Y41(:,3), is < 0 for this col input
     y_y =-180/pi*r1/Y41(t3,3);
     x_y =-180/pi*r3/Y41(t3,3);
     a_y = (x_y-0.15)^2; b_y = (y_y-0.65)^2; c_y = (x_y+0.2)^2;
     yaw_d2=0;
  if (y_y >= 0.65),
    yaw_d2 =b_y;
  end;
  if (x_y <= -0.15),
    yaw_d2=yaw_d2 + a_y;
  elseif (x_y >= 0.20),
    yaw_d2=yaw_d2 + c_y;
  end;
end;

% Spec 5
% ------
if SPEC5 == 1,
    % Linear Wind Gust response
```

```
    % -------------------------
    % Simulink model: dwtest1.m (Discrete time nonlinear model)
    % # states - order:
    %     6    - Wind Gust Model [states(1-6)]
    %     4    - Linear Actuator Dynamics [states(7-10)]
    %    11    - UH-60A Dynamics [states(11-21)]
    %     8    - Delay Dynamics [states(22-29)]
    % Use [sizes,x0,states]=dwtest1([],[],[],0) to verify state order

tfin=40;
options=[];

ut=[0 1 0 0 0;tfin 1 0 0 0];
[tg1,x,y1]=linsim('dwtest1',tfin,[],options,ut);

ut=[0 0 1 0 0;tfin 0 1 0 0];
[tg2,x,y2]=linsim('dwtest1',tfin,[],options,ut);

ut=[0 0 0 1 0;tfin 0 0 1 0];
[tg3,x,y3]=linsim('dwtest1',tfin,[],options,ut);

pitch_g = 180/pi*y1(:,8);  % [deg]
roll_g  = 180/pi*y2(:,7);
yaw_g   = 180/pi*y3(:,9);

end;

if MATLAB==0,
 save simu pitch_dist roll_dist yaw_dist ...
           pitch_bw   roll_bw   yaw_bw   ...
           pitch_pd   roll_pd   yaw_pd   ...
           pitch_damp roll_damp yaw_damp ...
           pitch_Wco  roll_Wco  yaw_Wco  ...
           pitch_AE   roll_AE   yaw_AE   ...
           pitch_r1s  roll_r1s  yaw_r1s   ...
           pitch_r2s  roll_r2s  yaw_r2s   ...
           pitch_r3s  roll_r3s  yaw_r3s   ...
           pitch_d    roll_d    yaw_d1 yaw_d2 ...
           pitch_g    roll_g    yaw_g  eigen_A
end;
s=s+1;
Ktetv=[Ktetv Ktet];
Kphiv=[Kphiv Kphi];
Kpsiv=[Kpsiv Kpsi];
Kqv  =[Kqv Kq];
Kpv  =[Kpv Kp];
```

```
Krv  =[Krv Kr];
Mtetv=[Mtetv Mtet];
Mphiv=[Mphiv Mphi];
Mpsiv=[Mpsiv Mpsi];
Kr_pv=[Kr_pv Kr_p];
Kp_rv=[Kp_rv Kp_r];
Kc_yv=[Kc_yv Kc_y];

%**************************  end of simu.m  ************************
```

Matlab m-file pilot.m

```
function u=pilot(t,r,a,type)

%    type     u ^
%               |  _______ a
%    'p'osition | /
%               |/r
%               /         -> t
%
%             u ^
%               |   a
%    'r'ate     | /\
%               |/r \
%               /    \____-> t

if type == 'p',
  T=max(find(t<=a/r));
  u=[r*t(1:T),a*ones(1,length(t)-T)];
else,
  T=max(find(t<=sqrt(a/r))); %RMS, area == desired position
  u=[r*t(1:T),r*(t(T)-t(2:T)),zeros(1,length(t)-2*T+1)];
end;

%************************** end of pilot.m **************************
```

Matlab m-file d_bw_pd.m

```
function [d,Bw,Pd] = d_bw_pd(A,B,C,D,Iu,Iy,PL);
%
% MATLAB function for ADOCS
% By : Gil Yudilevitch
% Last updated  :  11/19/92
%
% Calculate Band Width [rad/sec] and Phase Delay [sec] for
```

```
% the SISO system (input Iu output Iy) of:
%    dx/dt = Ax + Bu
%      y   = Cx + Du
% Then calculate the distance of (Bw,Pd) to a given bound (spec)
%
%    [d,Bw,Pd] = d_bw_pd(A,B,C,D,Iu,Iy)

% Calculate Bw
% ------------

w=0.5; p=0;
for i = 1:3,
  epsilon = 0.1^(i-1);
  while p > -135,
    q=p;
    w = w+epsilon;
    p=angle(C(Iy,:)*inv(j*w*eye(size(A))-A)*B(:,Iu)+D(Iy,Iu))*180/pi;
    if p > 0, p=p-360; end;
  end;
  p=q;
  w = w-epsilon;
end;
Bw = w+epsilon/2;

% Calculate Pd
% ------------

p=0;
for i = 1:5,
  epsilon = 0.1^(i-1);
  while p > -180,
   q=p;
   w = w+epsilon;
   p=angle(C(Iy,:)*inv((j*w*eye(size(A))-A))*B(:,Iu)+D(Iy,Iu))*180/pi;
   if p > 0, p=p-360; end;
  end;
  p=q;
  w = w-epsilon;
end;
w = w+epsilon/2;
p=angle(C(Iy,:)*inv((j*2*w*eye(size(A))-A))*B(:,Iu)+D(Iy,Iu))*180/pi;
p=p-360;

% Assumption, at 2*w180 : -180 deg  >=  phase >= -540 deg !!!

Pd = -(180+p)*pi/360/w;
```

```
% Calculate d
% -----------

d=Bw-polyval(PL,Pd);
%pause
if d > 0, d=0;
else,
  dmin=100;
  d=(d/4.45)^2;
  Pd0=Pd;
  while d < dmin,
    dmin=d;
    Pd0=Pd0-0.005;
    d=((Pd-Pd0)/0.4)^2+((Bw-polyval(PL,Pd0))/4.45)^2;
  end;
  d=dmin;
end;

%************************ end of d_bw_pd.m ***************************
```

Matlab m-file qcost.m

```
function c=qcost(dt,y,w)
% Quadratic cost value
%           t1
% c=integral{y'*w'*w*y}dt
%           t0
% dt - time increment                                    n
% y  - vector valued function on the interval [t0,t1] : n x m
% w  - weighting matrix                                : m x m
%
% By : Gil Yudilevitch
% Updated : 10-30-92

[n,m]=size(y);
z=(y*w').^2*ones(m,1); % <=> z(t)=y(t)'*w'*w*y(t), t=1,2,...,n
c=dt*(sum(z)-(z(1)+z(n))/2); %trapezoid integration

%************************ end of qcost.m ****************************
```

The C-O code necessary to run the hover problem using SIMULINK as the simulation engine does not require modification from the original code presented in Reference [9] and is therefore not presented here.

Appendix B

Forward Flight ADOCS Code

The files **pilot.m**, **d_bw_pd.m** and **qcost.m** are not modified for the forward flight simulation from their condition as presented in Appendix A and are thus not reproduced here.

C-O file adocs for Forward Flight ADOCS

```
/*================================================*/
/*   80 Kts Forward Flight ADOCS - PDF File       */
/*   Based on original PDF file by G. Yudilevitch */
/*   P. J. Potter 2/27/95                         */
/*================================================*/

include "dp_adocs" /* Defines design parameters */

include "adocs.dp.restart"  /*Updates design parameters*/

dt  = 1/30
tf  = 5
dtg = 1/30
tfg = 40

global double getout();

global double geto(name, t, dt)
global char *name;
global double t, dt;
global {
global    double r;
```

```
global    int i;
global    i = t/dt + 1.5;
global    r = getout(name, i);
global    return r;
global }

include "stable.all"

include "object.sat"

include "spec1.pit"
include "spec1.rol"
include "spec1.yaw"

include "spec2.pit"
include "spec2.rol"
include "spec2.yaw"

include "spec3.rol"

include "spec4.pit"
include "spec4.rol"
include "spec4.col"

include "spec5.pit"
include "spec5.rol"
include "spec5.yaw"

include "spec.gen"

/*====================  adocs   ====================*/
```

C-O file spec1.pit for Forward Flight ADOCS

```
/*=============================================================*/
/*  SPECIFICATION 1 - Small Amplitude, Short Term Response     */
/*  Bandwidth & Phase Delay                                    */
/*  Pitch (ref. ADS-33C  3.4.1.1)                              */
/*  Included in adocs PDF                                      */
/*  Original: G. Yudilevitch                                   */
/*  Updated: P. J. Potter 2/27/95                              */
/*=============================================================*/

constraint  "pit bw pd" soft
      {
      return getout("pitch_dist", 1);
```

```
      }
   <=
   good_value = 0.000
   bad_value  = 0.002

constraint "pit Td" hard
      {
      return getout("pitch_pd", 1)-0.4;
      }
   <=
   good_value = 0.0000
   bad_value =  0.0001

/*==================    end of spec1.pit   ==================*/
```

C-O file spec1.rol for Forward Flight ADOCS

```
/*============================================================*/
/*  SPECIFICATION 1 - Small Amplitude, Short Term Response    */
/*  Bandwidth & Phasedelay                                    */
/*  Roll  (ref. ADS-33C  3.4.5.1)                             */
/*  Included in adocs PDF                                     */
/*  Original: G. Yudilevitch                                  */
/*  Updated: P. J. Potter 2/27/95                             */
/*============================================================*/

constraint "rol bw pd" soft
      {
      return getout("roll_dist", 1);
      }
   <=
   good_value = 0.000
   bad_value  = 0.002

constraint "rol Td" hard
      {
      return getout("roll_pd", 1)-0.4;
      }
   <=
   good_value = 0.0000
   bad_value =  0.0001

/*==================    end of spec1.rol   ==================*/
```

C-O file spec1.yaw for Forward Flight ADOCS

```
/*============================================================*/
/*  SPECIFICATION 1 - Small Amplitude, Short Term Response    */
/*  Bandwidth & Phasedelay                                    */
/*  Yaw   (ref. ADS-33C  3.4.7.1)                             */
/*  Included in adocs PDF                                     */
/*  Original: G. Yudilevitch                                  */
/*  Updated: P. J. Potter 2/27/95                             */
/*============================================================*/

constraint "yaw bw pd" soft
      {
      return getout("yaw_dist", 1);
      }
   <=
   good_value = 0.000
   bad_value  = 0.002

constraint "yaw Td" hard
      {
      return getout("yaw_pd", 1)-0.4;
      }
   <=
   good_value = 0.0000
   bad_value =  0.0001

/*==================    end of spec1.yaw   ==================*/
```

C-O file spec2.pit for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPEC 2 - Small Amplitude, Mid Term Response        */
/*  Stability Margins  (replaces: Damping Ratio)       */
/*  Pitch (ref. ADS-33C  3.4.1.2, briefing 10-93)      */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

constraint "pit damp" soft
      {
      return getout("pitch_damp", 1);
      }
   <=
   good_value = 0.000
   bad_value =  0.001
```

```
/*================   end of spec2.pit   ===============*/
```

C-O file spec2.rol for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPEC 2 - Small Amplitude, Mid Term Response        */
/*  Stability Margins (replaces: Damping Ratio)        */
/*  Roll  (ref. ADS-33C  3.4.6.1, phoncon 032795)      */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

constraint "rol damp" soft
      {
      return getout("roll_damp", 1);
      }
   <=
   good_value = 0.000
   bad_value =  0.001

/*================   end of spec2.rol   ===============*/
```

C-O file spec2.yaw for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPEC 2 - Small Amplitude, Mid Term Response        */
/*  Stability Margins (replaces: Damping Ratio)        */
/*  Yaw (ref. ADS-33C  3.4.8.1, phoncon 032795)        */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

constraint "yaw damp" soft
      {
      return getout("yaw_damp", 1);
      }
   <=
   good_value = 0.000
   bad_value =  0.001

/*================   end of spec2.yaw   ===============*/
```

C-O file spec3.rol for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPEC3 - Moderate Amplitude, Attitude Quickness     */
/*  p_pk/phi_pk                                        */
/*  Roll (ref. ADS-33C  3.4.5.2)                       */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

constraint  "rol quick1" soft
      {
      return getout("roll_r1s", 1);
      }
   >=
   good_value =  0.00
   bad_value  = -0.014

constraint  "rol quick2" soft
      {
      return getout("roll_r2s", 1);
      }
   >=
   good_value =  0.00
   bad_value  = -0.08

constraint  "rol quick3" soft
      {
      return getout("roll_r3s", 1);
      }
   >=
   good_value =  0.00
   bad_value  = -0.07

/*================   end of spec3.rol   ===============*/
```

C-O file spec4.pit for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION 4 - Decoupling                       */
/*  theta/delta_phi                                    */
/*  Pitch (ref. ADS-33C  3.4.4.2)                      */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/
```

```
functional_constraint "pit dec up" soft
   for t from 0 to 4 by dt
   {  import dt;
      return geto("pitch_du", t, dt)-0.25; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.05; }

functional_constraint "pit dec lo" soft
   for t from 0 to 4 by dt
   {  import dt;
      return geto("pitch_dl", t, dt)+0.25; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.05; }

/*================   end of spec4.pit   ===============*/
```

C-O file spec4.rol for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION 4 - Decoupling                       */
/*  phi/delta-theta                                    */
/*  Roll (ref. ADS-33C  3.4.4.2)                       */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

functional_constraint "rol dec up" soft
   for t from 0 to 4 by dt
   {  import dt;
      return geto("roll_du", t, dt)-0.25; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.05; }

functional_constraint "rol dec lo" soft
   for t from 0 to 4 by dt
   {  import dt;
      return geto("roll_dl", t, dt)+0.25; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.05; }

/*================   end of spec4.rol   ===============*/
```

C-O file spec4.col for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION 4 - Deoupling                        */
/*  theta/delta_c                                      */
/*  Collective (ref. ADS-33C  3.3.4.1)                 */
/*  Included in adocs PDF                              */
/*  By : P. J. Potter 2/27/95                          */
/*  Last update  :  03/09/95                           */
/*=====================================================*/

functional_constraint "col att su" soft
   for t from 0 to 3 by dt
   { import dt;
     return geto("col_att_s", t, dt)-1.00; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.20; }

functional_constraint "col att sl" soft
   for t from 0 to 3 by dt
   { import dt;
     return geto("col_att_s", t, dt)+1.00; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.20; }

functional_constraint "col att luu" soft
   for t from 0 to 3 by dt
   { import dt;
     return geto("col_att_lu", t, dt)-0.50; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.10; }

functional_constraint "col att lul" soft
   for t from 0 to 3 by dt
   { import dt;
     return geto("col_att_lu", t, dt)+0.50; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.10; }

functional_constraint "col att ldu" soft
   for t from 0 to 3 by dt
   { import dt;
     return geto("col_att_ld", t, dt)-0.25; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.05; }

functional_constraint "col att ldl" soft
   for t from 0 to 3 by dt
```

```
   { import dt;
     return geto("col_att_ld", t, dt)+0.25; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.05; }

/*================   end of spec4.col   ===============*/
```

C-O file spec5.pit for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION 5 - Wind-Gust Rejection              */
/*  theta/gust                                         */
/*  Pitch (ref. M. Tischler 9/26/90)                   */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

functional_constraint "p gust p u" soft
   for t from 0 to 10 by dtg
   {  import dtg;
      return geto("pitch_g", t, dtg)-4; }
 <= good_curve = { return  0.0; }
    bad_curve =  { return  0.3; }

functional_constraint "p gust p l" soft
   for t from 0 to 10 by dtg
   {  import dtg;
      return geto("pitch_g", t, dtg)+4; }
 >= good_curve = { return  0.0; }
    bad_curve =  { return -0.3; }

functional_constraint "p gust t u" soft
   for t from 10+dtg to tfg by dtg
   {  import dtg;
      return geto("pitch_g", t, dtg)-0.4; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.03; }

functional_constraint "p gust t l" soft
   for t from 10+dtg to tfg by dtg
   {  import dtg;
      return geto("pitch_g", t, dtg)+0.4; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.03; }
```

```
/*================   end of spec5.pit   ===============*/
```

C-O file spec5.rol for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION 5 - Wind-Gust Rejection              */
/*  phi/gust                                           */
/*  Roll  (ref. M. Tischler 9/26/90)                   */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

functional_constraint "r gust p u" soft
   for t from 0 to 10 by dtg
   {  import dtg;
      return geto("roll_g", t, dtg)-4; }
 <= good_curve = { return  0.0; }
    bad_curve =  { return  0.3; }

functional_constraint "r gust p l" soft
   for t from 0 to 10 by dtg
   {  import dtg;
      return geto("roll_g", t, dtg)+4; }
 >= good_curve = { return  0.0; }
    bad_curve =  { return -0.3; }

functional_constraint "r gust t u" soft
   for t from 10+dtg to tfg by dtg
   {  import dtg;
      return geto("roll_g", t, dtg)-0.4; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.03; }

functional_constraint "r gust t l" soft
   for t from 10+dtg to tfg by dtg
   {  import dtg;
      return geto("roll_g", t, dtg)+0.4; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.03; }

/*================   end of spec5.rol   ===============*/
```

C-O file spec5.yaw for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION 5 - Wind-Gust Rejection              */
/*  psi/gust                                           */
/*  Yaw   (ref. M. Tischler 9/26/90)                   */
/*  Included in adocs PDF                              */
/*  Original: G. Yudilevitch                           */
/*  Updated: P. J. Potter 2/27/95                      */
/*=====================================================*/

functional_constraint "y gust p u" soft
   for t from 0 to 10 by dtg
   {  import dtg;
      return geto("yaw_g", t, dtg)-4; }
 <= good_curve = { return  0.0; }
    bad_curve =  { return  0.3; }

functional_constraint "y gust p l" soft
   for t from 0 to 10 by dtg
   {  import dtg;
      return geto("yaw_g", t, dtg)+4; }
 >= good_curve = { return  0.0; }
    bad_curve =  { return -0.3; }

functional_constraint "y gust t u" soft
   for t from 10+dtg to tfg by dtg
   {  import dtg;
      return geto("yaw_g", t, dtg)-0.4; }
 <= good_curve = { return  0.00; }
    bad_curve =  { return  0.03; }

functional_constraint "y gust t l" soft
   for t from 10+dtg to tfg by dtg
   {  import dtg;
      return geto("yaw_g", t, dtg)+0.4; }
 >= good_curve = { return  0.00; }
    bad_curve =  { return -0.03; }

/*================   end of spec5.yaw   ===============*/
```

C-O file spec.gen for Forward Flight ADOCS

```
/*=====================================================*/
/*  SPECIFICATION Gen - Attitude Reqs                  */
/*  Pitch, Roll & Yaw                                  */
/*  Included in adocs PDF                              */
/*  P. J. Potter 04/12/95                              */
```

```
/*=====================================================*/

constraint "pit att" soft
{
return getout("pit_att", 1)-0.90;
}
     >=
     good_value =  0.000
     bad_value  = -0.001

constraint "rol att" soft
{
return getout("rol_att", 1)-0.90;
}
     >=
     good_value =  0.000
     bad_value  = -0.001

constraint "yaw att" soft
{
return getout("yaw_att", 1)-0.90;
}
     >=
     good_value =  0.000
     bad_value  = -0.001

/*================   end of spec.gen    ===============*/
```

Matlab m-file init.m for Forward Flight ADOCS

```
% ********************************************************************
%   System Initialization File for 80 Knot Forward Flight ADOCS
%   init.m CONSOL-MATLAB file
%   Simulink Model 'ctest' for Spec 1 (Bandwidth-Phase Delay)
%   and Spec 2 (Stability margins), 'dtest' for Spec 3 (Quickness)
%   and Spec 4 (Coupling), and 'dwtest1' for Spec 5 (Wind gust)
%   Modified from original simu.m by Gil Yudilevitch, 04/09/94
%   P. J. Potter, 03/14/95
% ********************************************************************

% Optimization Set-up
% -------------------

OPSYS  = 1; % Operating system: 0 - DOS (PC); 1 - UNIX (Workstation).
MATLAB = 0; % Matlab run type: 0 - C-O/MATLAB; 1 - autonomous MATLAB
CONSOL = 0; % C-O run type: 0 - Background; 1 - interactive.
```

```
LEVEL  = 0; % Performance level [0,1]: 0 - Level 1, 1 - Level 1-.
LIMIT  = 1; % Actuator limits: actual_limit = LIMIT*nominal_limit.
TIME   = 1; % Actuator time const. (TC): actual_TC = TIME*nominal_TC.
SPEC1  = 1; % Specs: 0 - Specs not included; 1 - Specs are included.
SPEC2  = 1;
SPEC3  = 1;
SPEC4  = 1;
SPEC5  = 1;
FLIGHT = 'f80';
              %                                      f
              % Flight conditions (about hover)   l h r 10 kts
              %                                      b
              % Forward flight condition           f80


% General Information
% -------------------

dt = 1/30;                                  % Sampling rate 30 Hz
tf = 5; t =0:dt:tf; nt=length(t);           % Command Type Time Axis
t1= min(find(t==1));
t3= min(find(t==3));
t4= min(find(t==4));                        % Index for 1, 3 & 4 sec.
tfg=40;
tg=0:dt:tfg;                                % Wind-gust time vector
n10=max(find(tg<=10)); nfg=length(tg);
wo=logspace(-1,1,50);                       % Frequency vector
s=0;  % Counter for tracking dp's
Ktetv=[]; Kphiv=[]; Kpsiv=[]; Kqv=[];       % Initial dp vectors
Kpv=[]; Krv=[]; Mtetv=[]; Mphiv=[];
Mpsiv=[]; Kr_pv=[]; Kp_rv=[]; Kc_av=[];

%  Model Components
%  ----------------
%  1. Helicopter 6 DOF Dynamics + Rotor Aerodynamics
%     Linear Continuous Time Model

%  # of states     = 11 ( 9 for 6 dof dynamics + 2 for rotor flapping)
%  # of inputs     = 4
%  # of outputs    = 9

%                                          State Output Input
%                                          ----- ------ -----
%  u     - linear velocity along X axis       1     1
%  v     - linear velocity along Y axis       2     2
%  w     - linear velocity along Z axis       3     3      4  delta_c
```

```
%  p      - angular velocity about X axis     4     4
%  q      - angular velocity about Y axis     5     5
%  r      - angular velocity about Z axis     6     6
%  phi    - angle about X axis                7     7      2  delta_phi
%  theta  - angle about Y axis                8     8      1  delta_tet
%  psi    - angle about Z axis                9     9      3  delta_psi
%  b1c    - longitudinal flap  (-a1s)        10
%  b1s    - lateral flap       (-b1s)        11
%
%  Units:  linear velocity  [ft /sec]
%  -----   angular velocity [rad/sec]
%          helicopter angle [rad    ]
%          flap angle       [rad    ]
%          delta_#          [in     ]

B10to4 = zeros(10,4);          % Trans. 10 (UMGENHEL) --> 4 inputs
B10to4(1,2) = 1; B10to4(2,1) = 1; B10to4(4,3) = 1; B10to4(3,4) = 1;

% Continuous Time
% ---------------

if OPSYS == 1,
  eval(['load A11',FLIGHT,' -ascii;']); % Load A,B UMGENHEL matrices
  eval(['load B11',FLIGHT,' -ascii;']); % for Unix for the FLIGHT
  eval(['Ap = A11',FLIGHT,';']);        % conditions
  eval(['Bp = B11',FLIGHT,'*B10to4;']);
else,
  eval(['load a11',FLIGHT,'.;']);       % Load A,B UMGENHEL matrices
  eval(['load b11',FLIGHT,'.;']);       % for PC for the FLIGHT
  eval(['Ap = a11',FLIGHT,';']);        % conditions
  eval(['Bp = b11',FLIGHT,'*B10to4;']);
end;

%                                  Output
%State-> 1 2 3 4 5 6 7 8 9 1011     \/
Cp =    [1 0 0 0 0 0 0 0 0 0 0   % u
         0 1 0 0 0 0 0 0 0 0 0   % v
         0 0 1 0 0 0 0 0 0 0 0   % w
         0 0 0 1 0 0 0 0 0 0 0   % p, roll rate
         0 0 0 0 1 0 0 0 0 0 0   % q, pitch rate
         0 0 0 0 0 1 0 0 0 0 0   % r, yaw rate
         0 0 0 0 0 0 1 0 0 0 0   % phi, roll angle
         0 0 0 0 0 0 0 1 0 0 0   % theta, pitch angle
         0 0 0 0 0 0 0 0 1 0 0]; % psi, yaw angle

Dp = zeros(9,4);
```

```
IC=[133.80837490639 17.000405177578 6.4889620671889 0 0 0 ...
 -4.9882265202173e-10 4.8456287012275e-02 0 0 0]; % Trim state values
                                                  % for first 9 states
  % Added to linear
  % solution.

[Apm,Bpm,Cpm,Dpm]=modred(Ap,Bp,Cp,Dp,[10,11]);  % Reduced to 9 states
                                                % for "inverse plant"
                                                % parameters
% Discrete Time Helicopter Dynamics
% ---------------------------------

[Apd,Bpd,Cpd,Dpd]=c2dm(Ap,Bp,Cp,Dp,dt,'tustin');

% Pure Delay 2_nd Order Pade Approximation
% ----------------------------------------
%
% Total Delay was taken equal for all channels. Total delay for Pitch
% from phoncon with Mark 10/94 is 100 ms ==> 75 ms "pure" and 25 ms
% Actuator Time-Constant (see later in SP-Actuator Model)

Td = [0.075;0.075;0.075;0.075];
[Ad1,Bd1,Cd1,Dd1] = pade(Td(1),2);
[Ad2,Bd2,Cd2,Dd2] = pade(Td(2),2);
[Ad3,Bd3,Cd3,Dd3] = pade(Td(3),2);
[Ad4,Bd4,Cd4,Dd4] = pade(Td(4),2);
[Ad,Bd,Cd,Dd]=append(Ad1,Bd1,Cd1,Dd1,Ad2,Bd2,Cd2,Dd2);
[Ad,Bd,Cd,Dd]=append(Ad,Bd,Cd,Dd,Ad3,Bd3,Cd3,Dd3);
[Ad,Bd,Cd,Dd]=append(Ad,Bd,Cd,Dd,Ad4,Bd4,Cd4,Dd4);

[Add,Bdd,Cdd,Ddd]=c2dm(Ad,Bd,Cd,Dd,dt,'tustin');

% Swashplate-Actuators Model
% --------------------------

% LINEAR PART:

K  = 1/(TIME*0.025)*ones(1,4);    % 1/(Time Constant) equal actuators
[as1,bs1,cs1,ds1] = tf2ss([0,K(1)],[1,K(1)]);
[as2,bs2,cs2,ds2] = tf2ss([0,K(2)],[1,K(2)]);
[as3,bs3,cs3,ds3] = tf2ss([0,K(3)],[1,K(3)]);
[as4,bs4,cs4,ds4] = tf2ss([0,K(4)],[1,K(4)]);
[As,Bs,Cs,Ds] = append(as1,bs1,cs1,ds1,as2,bs2,cs2,ds2);
[As,Bs,Cs,Ds] = append(As,Bs,Cs,Ds,as3,bs3,cs3,ds3);
[As,Bs,Cs,Ds] = append(As,Bs,Cs,Ds,as4,bs4,cs4,ds4);
```

```
% NONLINEAR SATURATIONS:

% *** All data is given at cockpit units (i.e., in's of stick or pedal)
% *** The trim (80 kts) conditions set to be zero (see table below)
% *** The swashplate mechanism ("limits coupling") is NOT accounted for
% *** The rate limits are taken as: full stroke per 1 sec

% Command Limits Table, Ref. Sikorsky SER 70452 p 6.16 Fig. 6.3.1
% ..............................................................
% chn |     Pitch     |     Roll      |       Yaw       |   Collective  |
% act |-5.0 +0.2 +5.0|-5.0 -0.5 +5.0|-2.69 +0.85 +2.69| 0.0 +5.0 +10.0|
% sim |-5.2  0.0 +4.8|-4.5  0.0 +5.5|-3.54  0.00 +1.84|-5.0  0.0 + 5.0|

Csl = LIMIT*[ -5.2 -4.5 -3.54  -5];  % Displacement Lower Limits
Csu = LIMIT*[  4.8  5.5  1.84   5];  % Displacement Upper Limits
Rsl = LIMIT*[-10  -10   -5.38 -10];  % Rate Lower Limits
Rsu = LIMIT*[ 10   10    5.38  10];  % Rate Upper Limits

% Linear Continuous-Time Open-Loop
% --------------------------------

[Ao,Bo,Co,Do] = series(Ad,Bd,Cd,Dd,As,Bs,Cs,Ds); % Used for calc of
[Ao,Bo,Co,Do] = series(Ao,Bo,Co,Do,Ap,Bp,Cp,Dp); % eigen_A in simu.m

% Inverse plant (ff) constants
% ----------------------------

FGtet =  Bpm(5,1); FGphi =  Bpm(4,2); FGpsi =  Bpm(6,3);
FTtet = -Apm(5,5); FTphi = -Apm(4,4); FTpsi = -Apm(6,6);

% Time Response Inputs
% --------------------

DtR  = pi/180;
Stet = DtR*[ 5,17.5,30];
Sphi =-DtR*[10,35  ,60]; % worst direction (nonsymmetric saturation)
Spsi = DtR*[10,35  ,60];
Scol = 5;

% Input Sequences for Simulation Time Histories
% ---------------------------------------------

aRps=pilot(t,Stet(3),Stet(1),'p');  % AE
aRrs=pilot(t,Sphi(3),Sphi(1),'r');  % AE & Quickness
aRys=pilot(t,Spsi(3),Spsi(1),'r');  % AE
```

```
aRrm=pilot(t,Sphi(3),Sphi(2),'r');  % Quickness

aRpl=pilot(t,Stet(3),Stet(3),'p');  % Decoupling
aRrl=pilot(t,Sphi(3),Sphi(3),'r');  % Decoupling & Quickness

aRcs=pilot(t,Scol,.95,'p');     %  < 20% of Full Control
aRc=pilot(t,Scol,Scol,'p');     % >= 20% of Full Control UP
aRcd=pilot(t,-Scol,-Scol,'p'); % >= 20% of Full Control DOWN

% Linear Wind-Gust Model
% ----------------------

[ag,bg,cg,dg]=tf2ss([0 0.44 0],[1 2 1]);
[Ag,Bg,Cg,Dg]=append(ag,bg,cg,dg,ag,bg,cg,dg);
[Ag,Bg,Cg,Dg]=append(Ag,Bg,Cg,Dg,ag,bg,cg,dg);
[Ag,Bg,Cg,Dg]=append(Ag,Bg,Cg,Dg,[],[],[], 0);

[Agd,Bgd,Cgd,Dgd]=c2dm(Ag,Bg,Cg,Dg,dt,'tustin');

Bgust=[0     0     0     0
       0     0     0     0
       0     0     0     0
       0     FTphi 0     0
       FTtet 0     0     0
       0     0     FTpsi 0
       0     0     0     0
       0     0     0     0
       0     0     0     0
       0     0     0     0
       0     0     0     0];
Bgust=[Bgust Bp];
Dgust=[zeros(9,4) Dp];

[Apg,Bpg,Cpg,Dpg]=c2dm(Ap,Bgust,Cp,Dgust,dt,'tustin');

% Level 1 curve for spec 1
% ------------------------

Ts = [0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200 ...
      0.225 0.250 0.275 0.300 0.325 0.350 0.375 0.400];
Ws = [2.005 2.005 2.005 2.005 2.005 2.005 2.005 2.105 2.235 ...
      2.405 2.685 3.005 3.275 3.505 3.905 4.155 4.455];
PL1=polyfit(Ts,Ws,6);
Ws = Ws + 1.5*LEVEL;
PL =polyfit(Ts,Ws,6);
```

```
% LEVEL factors for spec 3
% ------------------------

d3r1 = 0.96; d3r2 = 0.75; d3r3 = 0.50;

% Switch variables for ctest.m
% ---------------------------

delopn=1; %Feedforward dynamics shutoff variable
deltet=1; %Pitch open loop calculation variable
delphi=1; % Roll open loop calculation variable
delpsi=1; %  Yaw open loop calculation variable

% Optimization Parameters (optional for interactive C-O/MATLAB run)
% ----------------------

if MATLAB == 0 & CONSOL == 1,
  CONS = [1:3,2,4,6,8:16,1,3,17,5:2:17];
  BAD  = [3;3;3;0.002;0.002;0.002;0.01;0.01;0.01;0.04;0.03;0.09;0.08;...
          0.07;0.04;0.05;0.05;0.002;0.5;0.3;0.03;0.3;0.03;0.3;0.03];
  Iter=-1; iter=[]; witer=[]; www=[]; kkk=[];
  perf=[]; dps=[]; grad=[]; bad=BAD;
end;

% SIMU.MAT saved variables, default values
% ----------------------------------------

pitch_dist=0; roll_dist=0; yaw_dist =0;
pitch_bw  =0; pitch_pd =0; roll_bw  =0; roll_pd =0;
yaw_bw    =0; yaw_pd   =0; eigen_A  =0;
pitch_damp=0; roll_damp=0; yaw_damp =0;
roll_r1s  =0; roll_r2s =0; roll_r3s =0;
pitch_du  =0; roll_du  =0; pitch_dl =0; roll_dl =0;
pitch_Wco =0; roll_Wco =0; yaw_Wco  =0;
pitch_g   =0; roll_g   =0; yaw_g    =0; col_att_s  =0;
pitch_AE  =0; roll_AE  =0; yaw_AE   =0; col_att_lu =0;
pit_att   =0; rol_att  =0; yaw_att  =0; col_att_ld =0;

%*****************************  end of init.m  **********************
```

Matlab m-file simu.m for Forward Flight ADOCS

```
% ************************************************************************
%   System Simulation File for 80 Knot Forward Flight ADOCS
%   simu.m CONSOL-MATLAB file
```

```
%   Simulink Model 'ctest' for Spec 1 (Bandwidth-Phase delay)
%   and Spec 2 (Stability margins), 'dtest' for Spec 3 (Quickness)
%   and Spec 4 (Coupling) and 'dwtest1' for Spec 5 (Wind gust)
%   Modified from original simu.m by Gil Yudilevitch, 04/09/94
%   P. J. Potter, 03/14/95
% ****************************************************************

% Feedforward (including Model Following & "Inverse Closed-Loop")
% ---------------------------------------------------------------

% Continuous Time
% ---------------

[Af_tet,Bf_tet,Cf_tet,Df_tet] = ...
tf2ss(Mtet^2*[1/FGtet Kq+1/FGtet/FTtet Ktet],[1 2*Mtet Mtet^2]);
[Af_phi,Bf_phi,Cf_phi,Df_phi] = ...
tf2ss(Mphi*[1/FGphi Kp+1/FGphi/FTphi Kphi],[1 Mphi 0]);
[Af_psi,Bf_psi,Cf_psi,Df_psi] = ...
tf2ss(Mpsi*[1/FGpsi Kr+1/FGpsi/FTpsi Kpsi],[1 Mpsi 0]);
[Af,Bf,Cf,Df]= ...
append(Af_tet,Bf_tet,Cf_tet,Df_tet,Af_phi,Bf_phi,Cf_phi,Df_phi);
[Af,Bf,Cf,Df]=append(Af,Bf,Cf,Df,Af_psi,Bf_psi,Cf_psi,Df_psi);
[Af,Bf,Cf,Df]=append(Af,Bf,Cf,Df,[],[],[],1);

% Discrete Time
% -------------

[Afd,Bfd,Cfd,Dfd]=c2dm(Af,Bf,Cf,Df,dt,'tustin');

% Crossfeed design
% ----------------

Kcf = [1    Kr_p 0 Kc_a
       Kp_r 1    0 0
       0    0    1 0
       0    0    0 1];

Bocf = Bo*Kcf; Docf = Do*Kcf;

% Spec 1, Linear Closed Loop
% --------------------------
if SPEC1 == 1,
    % Simulink model: ctest.m (Continuous time (CT) model)
    % # states - order:
    %    11    - UH-60A Dynamics [states(1-11)]
    %     6    - Feedforward Control [states(12-17)]
```

```
    %     8    - Delay Dynamics [states(18-25)]
    %     4    - Actuator Dynamics [states(26-29)]
    % Use [sizes,x0,states]=ctest([],[],[],0) to state order

%   u v w p  q  r  phi theta psi b1c b1s
H = [ ...
    0 0 0 0  Kq 0  0    Ktet 0    0   0     % delta_theta
    0 0 0 Kp 0  0  Kphi 0    0    0   0     % delta_phi
    0 0 0 0  0  Kr 0    0    Kpsi 0   0     % delta_psi
    0 0 0 0  0  0  0    0    0    0   0];   % delta-collective
HH=[zeros(4,12),H];
Ac=Ao-Bocf*HH;
eigen_A=max(real(eig(Ac)));

[A1,B1,C1,D1]=linmod('ctest');

  [pitch_dist,pitch_bw,pitch_pd] = d_bw_pd(A1,B1,C1,D1,1,8,PL1);% angle
  [roll_dist ,roll_bw ,roll_pd ] = d_bw_pd(A1,B1,C1,D1,2,4,PL); % rate
  [yaw_dist  ,yaw_bw  ,yaw_pd  ] = d_bw_pd(A1,B1,C1,D1,3,6,PL); % rate
end;

% Spec 2, Linear Broken-Loop
% --------------------------
if SPEC2 == 1,
    % Simulink model: ctest.m (Continuous time (CT) model)
    % # states - order:
    %    11    - UH-60A Dynamics [states(1-11)]
    %     6    - Feedforward Control [states(12-17)]
    %     8    - Delay Dynamics [states(18-25)]
    %     4    - Actuator Dynamics [states(26-29)]
    % Use [sizes,x0,states]=ctest([],[],[],0) to verify state order

GMO=6; PMO=45; % Standard gain and phase margin
delopn=-1;     % Send input around Feedforward dynamics
deltet=-1;     % Break the pitch feedback loop
[A2p,B2p,C2p,D2p]=linmod('ctest');
[mo1,po1]=bode(A2p,B2p(:,1),C2p(10,:),D2p(10,1),1,wo);
[Gm,pitch_PM,Wg,pitch_Wco] = margin(mo1,po1,wo); pitch_GM=20*log10(Gm);
d_g_p = (pitch_GM/GMO-1)^2;
d_p_p = (pitch_PM/PMO-1)^2;
    if pitch_GM >= GMO & pitch_PM >= PMO, pitch_damp = 0;
elseif pitch_GM >= GMO & pitch_PM <  PMO, pitch_damp = d_p_p;
elseif pitch_GM <  GMO & pitch_PM >= PMO, pitch_damp = d_g_p;
else,                                     pitch_damp = d_g_p+d_p_p;end;
deltet=1;      % Close the pitch feedback loop
```

```
delphi=-1;     % Break the roll feedback loop
[A2r,B2r,C2r,D2r]=linmod('ctest');
[mo2,po2]=bode(A2r,B2r(:,2),C2r(11,:),D2r(11,2),1,wo);
[Gm,roll_PM,Wg,roll_Wco] = margin(mo2,po2,wo); roll_GM =20*log10(Gm);
d_g_r = (roll_GM/GMO-1)^2;
d_p_r = (roll_PM/PMO-1)^2;
    if roll_GM >= GMO & roll_PM >= PMO, roll_damp = 0;
elseif roll_GM >= GMO & roll_PM <  PMO, roll_damp = d_p_r;
elseif roll_GM <  GMO & roll_PM >= PMO, roll_damp = d_g_r;
else,                                   roll_damp = d_g_r+d_p_r; end;
delphi=1;      % Close the roll feedback loop

delpsi=-1;     % Break the yaw feedback loop
[A2y,B2y,C2y,D2y]=linmod('ctest');
[mo3,po3]=bode(A2y,B2y(:,3),C2y(12,:),D2y(12,3),1,wo);
[Gm,yaw_PM,Wg,yaw_Wco] = margin(mo3,po3,wo); yaw_GM=20*log10(Gm);
d_g_y = (yaw_GM/GMO-1)^2;
d_p_y = (yaw_PM/PMO-1)^2;
    if yaw_GM >= GMO & yaw_PM >= PMO, yaw_damp = 0;
elseif yaw_GM >= GMO & yaw_PM <  PMO, yaw_damp = d_p_y;
elseif yaw_GM <  GMO & yaw_PM >= PMO, yaw_damp = d_g_y;
else,                                 yaw_damp = d_g_y+d_p_y; end;
delpsi=1;      % Close the yaw feedback loop
delopn=1;      % Reroute the input through the feedforward control
end;

% Specs 3 & 4, Nonlinear Closed-Loop
% ----------------------------------
if(SPEC3+SPEC4)>=1,
    % Simulink model: dtest.m
    % # states - order:
    %     4    - Actuator Dynamics [states(1-4)]
    %    11    - UH-60A Dynamics [states(5-15)]
    %     6    - Feedforward Control [states(16-21)]
    %     8    - Delay Dynamics [states(22-29)]
    % Use [sizes,x0,states]=dtest([],[],[],0) to verify state order

aR=zeros(5/dt+1,5);
[r,c]=size(aR);
aR4=zeros(r,4);
for k=1:r,
  aR(k,1)=(k-1)*dt;
end
option=[];

% Pitch input
```

```
% -----------

aR(:,2)=aRps';
[t11,X11,Y11]=linsim('dtest',5,[],option,aR);

aR(:,2)=aRpl';
[t13,X13,Y13]=linsim('dtest',5,[],option,aR);

% Roll input
% ----------

aR(:,2:5)=aR4;
aR(:,3)=aRrs';
[t21,X21,Y21]=linsim('dtest',5,[],option,aR);

aR(:,3)=aRrm';
[t22,X22,Y22]=linsim('dtest',5,[],option,aR);

aR(:,3)=aRrl';
[t23,X23,Y23]=linsim('dtest',5,[],option,aR);

% Yaw input
% ---------

aR(:,2:5)=aR4;
aR(:,4)=aRys';
[t31,X31,Y31]=linsim('dtest',5,[],option,aR);

% Collective input
% ----------------

aR(:,2:5)=aR4;
aR(:,5)=aRc';
[t41,X41,Y41]=linsim('dtest',5,[],option,aR);

aR(:,5)=aRcs';
[t42,X42,Y42]=linsim('dtest',5,[],option,aR);

aR(:,5)=aRcd';
[t43,X43,Y43]=linsim('dtest',5,[],option,aR);
end;

% Spec 3
% ------
% Per phone conversation with Mark Tischler on 03/10/95, confirmed
% that we should be implementing the ADS-33C requirement.
```

```
if SPEC3 == 1,
  roll_r1  = min(Y21(:,4)+IC(4))/min(Y21(:,7)+IC(7));
  roll_r1s = roll_r1-(1.39+d3r1*LEVEL);
  roll_r2  = min(Y22(:,4)+IC(4))/min(Y22(:,7)+IC(7));
  roll_r2s = roll_r2-(0.85+d3r2*LEVEL);
  roll_r3  = min(Y23(:,4)+IC(4))/min(Y23(:,7)+IC(7));
  roll_r3s = roll_r3-(0.75+d3r3*LEVEL);
  pit_att=max(Y11(:,8)+IC(8))/Stet(1);
  rol_att=min(Y21(:,7)+IC(7))/Sphi(1);
  yaw_att=max(Y31(:,9)+IC(9))/Spsi(1);
% Quadratic objectives (actuator "energy")
% --------------------
pitch_AE = qcost(dt,Y11(:,10),1);
 roll_AE = qcost(dt,Y21(:,11),1);
  yaw_AE = qcost(dt,Y31(:,12),1);

end;

% Spec 4
% ------
if SPEC4 == 1,
  pitch_d = (Y23(1:t4,8)+IC(8))/Sphi(3); %Pitch out/Roll input
  pitch_du=pitch_d-(1-LEVEL)*.35;
  pitch_dl=pitch_d+(1-LEVEL)*.35;
  roll_d  = (Y13(1:t4,7)+IC(7))/Stet(3); %Roll out/Pitch input
  roll_du=roll_d-(1-LEVEL)*.35;
  roll_dl=roll_d+(1-LEVEL)*.35 ;
  col_att_s  = (Y42(1:t3,8)+IC(8))*(1/DtR)/min(Y42(1:t3,13));
  col_att_lu = (Y41(1:t3,8)+IC(8))*(1/DtR)/min(Y41(1:t3,13));
  col_att_ld = (Y43(1:t3,8)+IC(8))*(1/DtR)/max(Y43(1:t3,13));
end;

% Spec 5
% ------
if SPEC5 == 1,
    % Linear Wind Gust response
    % -------------------------
    % Simulink model: dwtest1.m (Discrete time nonlinear model)
    % # states - order:
    %     6    - Wind Gust Model [states(1-6)]
    %     4    - Linear Actuator Dynamics [states(7-10)]
    %    11    - UH-60A Dynamics [states(11-21)]
    %     8    - Delay Dynamics [states(22-29)]
    % Use [sizes,x0,states]=dwtest1([],[],[],0) to verify state order

tfin=40;
```

```
ut=[0 1 0 0 0;tfin 1 0 0 0];
options=[];

[tg1,x,y1]=linsim('dwtest1',tfin,[],options,ut);

ut=[0 0 1 0 0;tfin 0 1 0 0];
[tg2,x,y2]=linsim('dwtest1',tfin,[],options,ut);

ut=[0 0 0 1 0;tfin 0 0 1 0];
[tg3,x,y3]=linsim('dwtest1',tfin,[],options,ut);

pitch_g = 180/pi*y1(:,8);  % [deg]
roll_g  = 180/pi*y2(:,7);
yaw_g   = 180/pi*y3(:,9);
end;

if MATLAB==0,
 save simu pitch_dist roll_dist yaw_dist ...
           pitch_bw   roll_bw   yaw_bw   ...
           pitch_pd   roll_pd   yaw_pd   ...
           pitch_damp roll_damp yaw_damp ...
           pitch_Wco  roll_Wco  yaw_Wco  ...
           pitch_AE   roll_AE   yaw_AE   ...
           roll_r1s  ...
           roll_r2s  ...
           roll_r3s  ...
           pitch_du   pitch_dl  roll_du roll_dl ...
           pitch_g    roll_g    yaw_g   eigen_A ...
           pit_att    rol_att   yaw_att ...
           col_att_s  col_att_lu col_att_ld
end;
s=s+1;
Ktetv=[Ktetv Ktet];
Kphiv=[Kphiv Kphi];
Kpsiv=[Kpsiv Kpsi];
Kqv  =[Kqv Kq];
Kpv  =[Kpv Kp];
Krv  =[Krv Kr];
Mtetv=[Mtetv Mtet];
Mphiv=[Mphiv Mphi];
Mpsiv=[Mpsiv Mpsi];
Kr_pv=[Kr_pv Kr_p];
Kp_rv=[Kp_rv Kp_r];
Kc_av=[Kc_av Kc_a];

%****************************** end of simu.m ************************
```

Bibliography

[1] M. K. H. Fan, A. L. Tits, J. L. Zhou, L. S. Wang, and J. Koninckx. CONSOL - OPTCAD, User's Manual. Technical Report 87-212r2a, Department of Electrical Engineering and Institute for Systems Research, University of Maryland, College Park, MD, 1991.

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

[3] R. H. Hoh, D. G. Mitchell, B. L. Aponso, D. L. Key, and C. L. Blanken. Background Information and User's Guide for Handling Qualities Requirements for Military Rotorcraft. Technical Report 89-A-008, United States Army Aviation Systems Command, Directorate for Engineering, 1989.

[4] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Massachusetts 01760. *SIMULINK, A Program for Simulating Dynamic Systems*, 1992.

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

[6] M. B. Tischler. Sample Disturbance Response Specification, September 1990.

[7] M. B. Tischler, J. W. Fletcher, P. M. Morris, and G. E. Tucker. Flying Quality Analysis and Flight Evaluation of a Highly Augmented Combat Rotorcraft. *AIAA Journal of Guidance, Control and Dynamics*, 14(5), September - October 1991.

[8] United States Army Aviation Systems Command, Directorate for Engineering. *Aeronautical Design Standard, Handling Qualities Requirements for Military Rotorcraft, ADS-33C*, 1989.

[9] G. Yudilevitch and W. S. Levine. Techniques for Designing Rotorcraft Control Systems. Technical Report 94-54, Department of Electrical Engineering and Institute for Systems Research, University of Maryland, College Park, MD, 1994.

[10] G. Yudilevitch, M. B. Tischler, W. S. Levine, C. Lin, and P. J. Potter. Rotorcraft Flight Control System Design Based on Multi-criterion Parametric Optimization. To be presented at the 1995 AIAA Conference on Guidance, Control and Dynamics, 1995.