5-1999

Solving an Inverse Control Problem using Predictive Methods

Chad J. Davis

AFIT/GAE/ENY/99M-05

Solving an Inverse Control Problem using Predictive Methods

THESIS
Chad Jeffrey Davis
First Lieutenant, USAF

AFIT/GAE/ENY/99M-05

AFIT/GAE/ENY/99M-05

Solving an Inverse Control Problem using Predictive Methods

THESIS

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

Chad Jeffrey Davis, B.S. Mechanical Engineering
First Lieutenant, USAF

May, 1999

AFIT/GAE/ENY/99M-05

Solving an Inverse Control Problem Using Predictive Methods

Chad J. Davis, B.S.

First Lieutenant, USAF

Degree of Master of Science in Aeronautical Engineering

March 1999

Approved:	Date
[illegible]	6 MAR 99
Major Sharon Heise (Chairman)	
[illegible]	8 MAR 99
Dr. Brad Liebst	
[illegible]	8 Mar 99
Major Montgomery Hughson	

Acknowledgements

I would like to express my deepest gratitude to my thesis advisor, Major Sharon Heise. Her unending support and guidance were invaluable, especially through her transfer out of AFIT while my thesis was still being developed. I would also like to thank Dr. Brad Liebst for imparting to me some of his extensive knowledge in the field of systems control and dynamics. Thank you also to the Air Vehicles Directorate of the Air Force Research Laboratory for sponsoring my work.

Additionally, I cannot thank my wife, Courtney, enough for her support and understanding throughout my studies and while producing this thesis. Finally, I would like to thank my parents, Glenn and Lorene. They gave me the opportunity and motivation to succeed.

Chad Jeffrey Davis

Table of Contents

List of Figures

AFIT/GAE/ENY/99M-05

Abstract

Model Predictive Control (MPC) is the class of control methods that optimize a specified performance index in order to minimize weighted future output deviations from a setpoint trajectory. MPC operates on a receding horizon, calculating a set of inputs at each time step. The controller then implements the first input and the process begins again. The performance index can also include a weighted and/or constrained control sequence which can be of different length than the output horizon.

This thesis applies MPC in the inverse sense – known aircraft outputs are applied in the performance index as setpoints in an attempt to determine what control histories caused those outputs. Using this method, aircraft mishap investigators could then have a means of determining what the control surface deflections were throughout an incident. To accomplish these objectives, MATLAB and MATLAB routines are used in conjunction with SIMULINK to develop the controller and simulate the aircraft's response. The actual Flight Data Recorder data from aircraft mishaps are utilized as proof of concept.

Solving an Inverse Control Problem using Predictive Methods

I. Introduction

1.1 Inverse Control Problem

Recently, much interest has been placed in the area of reconstructing aircraft accidents based on the information available in the Cockpit Voice Recorder (CVR) and the Flight Data Recorder (FDR) post-crash. The CVR can provide audible clues of the accident's cause such as stall warnings, engine noise, landing gear extension/retraction, etc.; whereas, the FDR provides the investigator with the operating conditions leading up to and during the crash such as altitude, heading, airspeed, etc.. Neither the CVR nor the FDR provide definitive information regarding the control surface locations throughout the aircraft incident. Aircraft investigators currently use the information included in the CVR and FDR in an attempt to determine the cause of the accident – be it a control surface failure or other cause. However, of great value to the investigators would be the control histories which caused the aircraft response.

The effort herein focuses on the particular application of post-processing aircraft data from an incident. However, the derivation and algorithm are generic enough so any system could be used. In fact, a distillation process on which model predictive control had previously been performed was used in this thesis to validate the solution algorithm. Any plant model of the form discussed here could be used, with the methods described, to obtain the inputs from known outputs.

1.2 Model Predictive Control

This research effort applies predictive methods to the inverse control problem in an attempt to determine, analytically, the control surface deflections that caused the aircraft response obtained in an FDR. Model Predictive Control (MPC) is such a method, utilizing on-line optimization of a specified performance index in order to minimize weighted future output deviations from a setpoint trajectory along a prediction horizon. Typically, the performance index also includes a weighted (and possibly constrained) control sequence across a control horizon which can be, but does not have to be, the same length as the prediction horizon. MPC has proven to be beneficial as a flight controller (14), especially in the realm of aircraft control in the presence of actuator failures (2). This thesis details the usefulness of MPC for the inverse control problem by developing a performance index utilizing the FDR data from an aircraft accident as the setpoint trajectory.

1.3 Research Objectives

The primary objective of this thesis is to determine whether MPC is useful in determining the control histories that were involved in an aircraft incident. By applying the theory to be developed to actual aircraft crashes of unknown (or suspected) cause, this objective will be accomplished. Also to be explored are the effects of including a rate term in the performance index for improved setpoint following. These objectives will be accomplished using MATLAB (7) to develop the controller and SIMULINK (8) to implement the controller on-line with the aircraft model.

1.4 Thesis Overview

Chapter 2 reviews the available literature used in development of this thesis.

Chapter 3 details the methodology and necessary background information for the development of an MPC controller, including modifications for system constraints and control increments.

Chapter 4 delves into the first application, a distillation column, for validation purposes.

Chapter 5 presents the applications dealing with two aircraft mishaps, discussing the scenarios, models used, and simulation results.

Chapter 6 offers conclusions that can be made based on the results of the work presented and suggests areas for future research in the area of inverse control using predictive methods.

II. Literature Review: Inverse Simulation

In the context of this thesis, inverse simulation is the practice of determining a system's input when only the outputs and the system model are known. Typically, *inverse control* is used when the effort goes one step further – including feedback control for disturbance rejection and plant uncertainty robustness. Several methods have been studied in an attempt to perform inverse simulation. The most relevant for the discussion here are what have been called the differential and integration inverse methods and those utilizing optimization.

2.1 Differential Inverse Method

The differential inverse method entails modeling the desired plant trajectory as a set of dynamic constraints imposed on the equations of motion (1). This trajectory can be defined analytically, as a given equation, or numerically, as a series of specific points which are then smoothed to provide a continuous trajectory. The constrained equations of motion are then integrated until the control terms arise which can be solved for directly.

The shortcomings of this method of inverse simulation deal with the numbers of states, inputs, and outputs (3). If a typical plant is considered of the form

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t)\end{aligned} \tag{1}$$

the solution for the control to obtain the desired trajectory, y_D, resulting from this method is

$$u(t) = B^{-1}[\dot{x}(t) - Ax(t)] \tag{2}$$

where

$$\dot{x}(t) = C^{-1}\dot{y}_D(t) \tag{3}$$

initiates the state derivatives. The resulting input is used with Equation (1) to obtain the state derivatives at subsequent time steps. Letting n_x be the number of states, n_y the number of outputs, and n_u the number of inputs, if $n_x = n_y = n_u$, no problems arise in the solution. However, if $n_x > n_u$ and $n_x > n_y$, as is typical of aircraft models, Equation (3) can no longer provide all the state derivatives necessary to solve the problem. To avoid this problem, initial guesses of the states are typically made.

The differential inverse method has been shown to provide solutions to the problem at hand. However, it is desirable to develop an algorithm that does not contain shortcomings of this nature where the solution might depend on initial guesses of states.

2.2 Integration Inverse Method

The integration inverse method develops an error vector defined as the difference between the actual system output and the desired system output and attempts to drive that error to zero (3). If

$$y(kT) = G[u(kT)] \tag{4}$$

where G maps the input, u, to the output, y, then with the introduction of an error vector, F_E,

$$F_E[u_n(kT)] = G[u_n(kT) - Y_D(kT)] \tag{5}$$

where Y_D is the desired trajectory. The solution then becomes

$$u_{n+1}(kT) = u_n(kT) - (J\{G[u_n(kT)]\})^{-1} \cdot F_E[u_n(kT)] \tag{6}$$

where J{} is the Jacobian matrix found through partial derivatives of the output approximated by

$$\delta y_i(kT)/\delta u_j(kT) \approx [y_i(u_j + \Delta u_j) \mid_{(k+1)T} - y_i(u_j) \mid_{kT}]/\Delta u_j \tag{7}$$

Iteration on the desired input, u_{n+1}, is then performed until the actual output and the desired output differ only by the tolerance required.

In this manner, the integration method avoids the shortcomings of the differential method in that the only requirement imposed upon the number of states, inputs, and outputs is that $n_y \leq n_u$. Therefore, a solution of this type is of more interest for this effort.

2.3 Optimization Methods

The final methods of relevance in available literature are those involving optimization. Similar to the differential method, optimization methods formulate the solution by setting equality constraints on functions of the state variables. However, the solution is not found by differentiating the equations but as a general optimization problem.

In this type of solution, a cost function is defined and is augmented by any or all of: constraints on initial and final conditions, path constraints, a dynamic constraint equation, and input control constraints (5). The necessary and sufficient conditions for optimality are then found as well as the input solution to optimally drive the system to the desired trajectory. This method does not require time differentiation or output derivatives, and therefore the numerical difficulties of the previous methods can be avoided. Also avoided are the sensitivities of the results to initial guesses.

In this thesis, optimization is utilized via Model Predictive Control, which features on-line optimization of the control inputs and real-time simulation of the plant outputs.

III. MPC-State Space Formulation

A method of control which incorporates the advantages of optimization and eliminates the disadvantages of time differentiation and initial guess sensitivities is Model Predictive Control (MPC). This section presents the development of a state-space, constrained MPC controller which utilizes control increments.

It should also be noted that in several instances, users of MPC have included in the problem a stabilizing inner feedback loop (2). This is done in order to ensure that the controller in these forward loop applications produces only stabilizing inputs. This will not be the practice here as application of a stabilizing inner loop might prevent the algorithm from following an output trajectory which is in fact unstable. Whether the setpoint trajectory is stable or unstable, as could be the case in aircraft incidents, the inverse controller should be able to follow the given outputs.

In this thesis, the controller is based on a linear discrete-time plant model of the form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{8}$$

where $x \in R^n$, is the state vector, $u \in R^\xi$, is the input vector, and $y \in R^\eta$, is the output vector so $A \in R^{nxn}, B \in R^{nx\xi}, C \in R^{\eta xn}$, and $D \in R^{\eta x\xi}$.

The Model Predictive Controller uses this plant model (Equation (8)) to calculate the future plant outputs, y, along a specified prediction horizon, p, due to the inputs, u, implemented over the control horizon, q, and minimize the cost function involving the difference between these outputs and the desired setpoint output trajectory, y_d. The outputs are weighted by Q, and the inputs are weighted by R across their respective horizons. This gives the overall cost function

$$J = \sum_{\ell=1}^{p} \| C\hat{x}(k+\ell) - y_d(k+\ell) \|_Q^2 + \sum_{\ell=0}^{q-1} \| u(k+\ell) \|_R^2 \tag{9}$$

Noting that the actual plant states, x, are not assumed to be known at any time in the future, it is necessary to estimate the future states, $\hat{x}(k+\ell)$, based on past states and inputs and future inputs. Based on an estimator of the form

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L[y(k) - C\hat{x}(k)] \tag{10}$$

it is seen that

$$\hat{x}(k+\ell) = (A - LC)\hat{x}(k+\ell-1) + Bu(k+\ell-1) + Ly(k+\ell-1) \tag{11}$$

With this equation, it is then possible to express the state estimates as (using the notation of (6))

$$\hat{x}(k+\ell+1) = F(\ell)\hat{x}(k+\ell) + Gu(k+\ell) + H(\ell)y(k+\ell) \tag{12}$$

where F, G, and H are defined as $F(m) = (A - LC)^{m+1}$, $G = B$, $H(0) = -L$ and $H(m) = 0$ for $m = 1 \cdots \ell$. These terms dealing with output feedback in H are eliminated for all times greater than $k+1$ since the actual system output will not be available at any future time.

It is now possible to define a vector of future predicted states:

$$\begin{bmatrix} \hat{x}(k+1) \\ \vdots \\ \hat{x}(k+p) \end{bmatrix} = \mathcal{F}\hat{x}(k) + \mathcal{G}[u(k)^T \cdots u(k+q-1)^T]^T + \mathcal{H}(\ell)y(k) \tag{13}$$

where $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$ are matrix functions of $F(\ell)$, G, and $H(\ell)$ as follows:

$$\mathcal{F} = \overbrace{\begin{bmatrix} (A - L \cdot C) \\ (A - L \cdot C)^2 \\ \vdots \\ (A - L \cdot C)^p \end{bmatrix}}^{n} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \end{bmatrix}}\right\} pn \tag{14}$$

$$\mathcal{G} = \overbrace{\begin{bmatrix} B & 0 & 0 & 0 \\ (A - L \cdot C)B & B & 0 & 0 \\ (A - L \cdot C)^2 B & (A - L \cdot C)B & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ (A - L \cdot C)^{p-1} B & (A - L \cdot C)^{p-2} B & \cdots & B \end{bmatrix}}^{q\xi} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \\ a \end{bmatrix}}\right\} pn \tag{15}$$

$$\mathcal{H} = \overbrace{\begin{bmatrix} L \\ (A - L \cdot C)L \\ \vdots \\ (A - L \cdot C)^{p-1} L \end{bmatrix}}^{\eta} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \end{bmatrix}}\right\} pn \tag{16}$$

The state prediction vector can then be used to transform the performance indices developed in the next sections into the form of a quadratic program

$$J = \overset{min}{U} \frac{1}{2}\{U^T P U + f^T U\} \tag{17}$$
$$\textit{subject to } AU \leq b$$

where

$$U = [u(k)^T u(k+1)^T \cdots u(k+q-1)^T]^T \tag{18}$$

since the index is minimized over the control input, U. Also, P and f will be functions of the state prediction matrices. In this form, MPC is easily implementable using a quadratic program algorithm to determine the future inputs to drive the system to the desired trajectory.

3.1 Non-rate performance index

Using the state prediction matrices, Equation (13), and expanding the cost function, Equation (9), into the form of Equation (17) results in

$$J = U(k)^T(\mathcal{G}^T\mathcal{C}^T\mathcal{Q}\mathcal{C}\mathcal{G} + \mathcal{R})U(k) + 2\{[\mathcal{F}\hat{x}(k) + \mathcal{H}y(k)]^T\mathcal{C}^T - \mathcal{S}\}\mathcal{Q}\mathcal{C}\mathcal{G}U(k) + \kappa \quad (19)$$

and κ includes all terms from the expansion independent of U, and is therefore neglected. In addition, the matrices

$$\begin{aligned} \mathcal{C} &= diag(C \cdots C) \\ \mathcal{Q} &= diag(Q \cdots Q) \\ \mathcal{R} &= diag(R \cdots R) \\ \mathcal{S} &= \begin{bmatrix} y_d(k) \\ y_d(k+1) \\ \vdots \\ y_d(k+p) \end{bmatrix} \end{aligned} \quad (20)$$

are defined.

3.2 Rate performance index

It has been suggested that inclusion of an error rate term, in addition to the previously discussed output error term, in the performance index may improve the controller's tracking of the setpoints. In order to include an error rate term, define the error, e, as

$$e(k+\ell) = C\hat{x}(k+\ell) - y_d(k+\ell) \quad (21)$$

Then, define a performance index similar to Equation (9) as

$$J = \sum_{\ell=1}^{p} \| \Delta e(k+\ell) + Ee(k+\ell) \|_Q^2 + \sum_{\ell=0}^{q-1} \| u(k+\ell) \|_R^2 \tag{22}$$

where the notation is identical to that in Equation (9) with the addition of Δe, the error rate term, and E, which is a weight placed on the output error. By weighting the output error in this manner, the actual output approaches the setpoint arbitrarily fast by choice of E through $\Delta e = -Ee$, since the performance index attempts to drive $\Delta e(k+\ell) + Ee(k+\ell)$ to zero.

The state prediction equation will require some manipulation with the inclusion of the error rate term, $\Delta e(k) = e(k) - e(k-1)$ or

$$\Delta e(k+\ell) = C\hat{x}(k+\ell) - y_d(k+\ell) - C\hat{x}(k+\ell-1) + y_d(k+\ell-1) \tag{23}$$

which can be expanded into a vector, $\Delta\tilde{e}(k)$, as

$$\Delta\tilde{e}(k) = \left[\Delta e(k+1)^T \cdots \Delta e(k+p)^T\right]^T \tag{24}$$

$$\Delta\tilde{e}(k) = \mathcal{CF}\hat{x}(k) + \mathcal{CG}U(k) + \mathcal{CH}y(k) - \mathcal{Y} - \mathcal{C}\tilde{\mathcal{F}}\hat{x}(k) - \mathcal{C}\tilde{\mathcal{G}}U(k) - \mathcal{C}\tilde{\mathcal{H}}y(k) + \tilde{\mathcal{Y}} \tag{25}$$

Using the notation of Equation (13) and noting $\tilde{\mathcal{F}}$ is identical to $\mathcal{F}$, time-shifted with the first row as zeros due to the time shift between $e(k)$ and $e(k-1)$:

$$\tilde{\mathcal{F}} = \left.\overbrace{\begin{bmatrix} 0 \\ (A - L\cdot C) \\ (A - L\cdot C)^2 \\ \vdots \\ (A - L\cdot C)^{p-1} \end{bmatrix}}^{n}\right\} pn \tag{26}$$

And $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{H}}$ are defined similarly. Finally, we note

$$\mathcal{Y} = \begin{bmatrix} y_d(k+1)^T \\ y_d(k+2)^T \\ \vdots \\ y_d(k+p)^T \end{bmatrix} \tag{27}$$

and

$$\tilde{\mathcal{Y}} = \begin{bmatrix} y_d(k)^T \\ \vdots \\ y_d(k+p-1)^T \end{bmatrix} \tag{28}$$

We can then rewrite Equation (25) as

$$\Delta\tilde{e}(k) = \mathcal{C}[\hat{\mathcal{F}}\hat{x}(k) + \hat{\mathcal{G}}U(k) + \hat{\mathcal{H}}y(k)] + \hat{\mathcal{Y}} \tag{29}$$

if we let $\hat{\mathcal{F}}, \hat{\mathcal{G}}, \hat{\mathcal{H}}$, and $\hat{\mathcal{Y}}$ equal $\mathcal{F} - \tilde{\mathcal{F}}$, etc. Also, from direct substitution of Equation (13) into (21), across the horizon, we have

$$\tilde{e}(k) = \mathcal{C}[\mathcal{F}\hat{x}(k) + \mathcal{G}U(k) + \mathcal{H}y(k)] - \mathcal{Y} \tag{30}$$

Utilizing Equations (29) and (30) and expanding the cost function, Equation (22), into the general quadratic program form, we see

$$\begin{aligned} J = U(k)^T[\hat{\mathcal{G}}^T\mathcal{C}^T\mathcal{Q}(\mathcal{C}\hat{\mathcal{G}} + 2\mathcal{E}\mathcal{C}\mathcal{G}) + \mathcal{G}^T\mathcal{C}^T\mathcal{E}^T\mathcal{Q}\mathcal{E}\mathcal{C}\mathcal{G} + \mathcal{R}]U(k) \\ +2\{[\mathcal{C}\hat{\mathcal{F}}\hat{x}(k) + \mathcal{C}\hat{\mathcal{H}}y(k) + \hat{\mathcal{Y}}]^T\mathcal{Q}[\mathcal{C}\hat{\mathcal{G}} + \mathcal{E}\mathcal{C}\mathcal{G}] \\ + [\mathcal{C}\mathcal{F}\hat{x}(k) + \mathcal{C}\mathcal{H}y(k) - \mathcal{Y}]^T\mathcal{E}^T\mathcal{Q}[\mathcal{C}\hat{\mathcal{G}} + \mathcal{E}\mathcal{C}\mathcal{G}]\}U(k) + \kappa \end{aligned} \tag{31}$$

and κ once again includes the neglected terms independent in U. This formulation allows the MATLAB functions which will perform the quadratic program to calculate the original matrices, $\mathcal{F}$, etc. and manipulate those into the required matrices, $\hat{\mathcal{F}}$, etc. for the index including the error rate term.

3.3 System Constraints

Typically, the inputs determined from an MPC controller for an aerospace system will be constrained by rate and deflection limits based on either servo or physical limitations in both the maximum and minimum positions. This is in order to prevent the controller from attempting to drive the system with inputs that cannot be physically attained. With emphasis on the inverse control framework, the desire is to reconstruct inputs that have already been imposed on an actual system, and therefore the inputs will only be constrained to their physical deflection limits, both positive and negative. This restriction will ease calculations, decrease calculation time slightly, and provide no adverse effect on the results based on the above discussion. In fact, cases can be envisioned where the control deflection that actually occurred exceeded servo rate limits (implying some failure), and exclusion of those cases could prevent good setpoint tracking. Also, the output will only be constrained at the current time step instead of across the control horizon to allow the controller to be more aggressive, while still operating within limitations, in following the potentially volatile output trajectory.

Finally, the constraints pertinent for an aerospace system, written across the control horizon, are most easily expressed as

$$\begin{bmatrix} u_{min}(k) \\ \vdots \\ u_{min}(k+q-1) \end{bmatrix} \leq \begin{bmatrix} u(k) \\ \vdots \\ u(k+q-1) \end{bmatrix} \leq \begin{bmatrix} u_{max}(k) \\ \vdots \\ u_{max}(k+q-1) \end{bmatrix} \qquad (32)$$

These constraints must be expressed in the same framework as the quadratic program given in Equation (17) as $AU \leq b$, or

$$\begin{bmatrix} I \\ -I \end{bmatrix} \begin{bmatrix} u(k) \\ \vdots \\ u(k+q-1) \end{bmatrix} \leq \begin{bmatrix} u_{max}(k) \\ \vdots \\ u_{max}(k+q-1) \\ -u_{min}(k) \\ \vdots \\ -u_{min}(k+q-1) \end{bmatrix} \tag{33}$$

Constraints will only be applied in this thesis when working with the transport aircraft model. No constraints are placed on the distillation process since it is an unconstrained benchmark problem. The physical deflections limitations for the transport, then, are (4)

- Elevators $(\delta_e) = +19.33\,\text{deg}, -21.83\,\text{deg}$
- Ailerons $(\delta_a) = \pm 20\,\text{deg}$
- Rudder $(\delta_r) = \pm 26\,\text{deg}$
- Throttle $(\delta_T) = 0 \rightarrow 100\%$

3.4 Plant Modifications for Control Increments

The computer program used to solve the quadratic program utilizes an incremental control input instead of absolute inputs, so the aircraft model will need to be modified to accept these inputs. One method of achieving this is to introduce an additional state corresponding to each input so

$$\begin{bmatrix} x_p(k+1) \\ x_u(k+1) \end{bmatrix} = \hat{A} \begin{bmatrix} x_p(k) \\ x_u(k) \end{bmatrix} + \hat{B}\Delta u(k) \;\; y(k) = \hat{C} \begin{bmatrix} x_p(k) \\ x_u(k) \end{bmatrix} \tag{34}$$

where

$$\hat{A} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \hat{B} = \begin{bmatrix} B \\ I \end{bmatrix} \hat{C} = [C\ 0] \tag{35}$$

In this manner, the additional *input states*, $x_u(k) = u(k-1)$, allow absolute inputs to be carried from one time step to the next. The performance indices found in Equations (19) and (31) will not change with incremental control inputs – the derivation could have just as easily began by replacing u with Δu in Equations (9) and (22).

With the introduction of control increments, the constraint equation, Equation (33), must be modified to reflect that the predictive controller will be determining a control increment, not an absolute control input. This is easily accomplished by substituting the equation for control increments

$$u(k+\ell) = \Delta u(k+\ell) + \cdots + \Delta u(k+q) + u(k-1) \tag{36}$$

into Equation (33). After simple algebraic manipulation we see that the new constraint equation, involving Δu, and of the form $AU \leq b$ is

$$\begin{bmatrix} I \\ -I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ \vdots & \ddots & 0 \\ I & \cdots & I \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+q) \end{bmatrix} \leq \begin{bmatrix} u_{max} \\ \vdots \\ u_{min} \\ \vdots \end{bmatrix} - \begin{bmatrix} I \\ -I \end{bmatrix} \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \tag{37}$$

3.5 Solution Algorithm

MATLAB functions were developed (see Appendix A) to solve the quadratic programs given, forming the predictive controller. SIMULINK was then used to couple this controller with the system model to complete the system. Figure 1 shows the complete system that was developed to solve the inverse control problem.

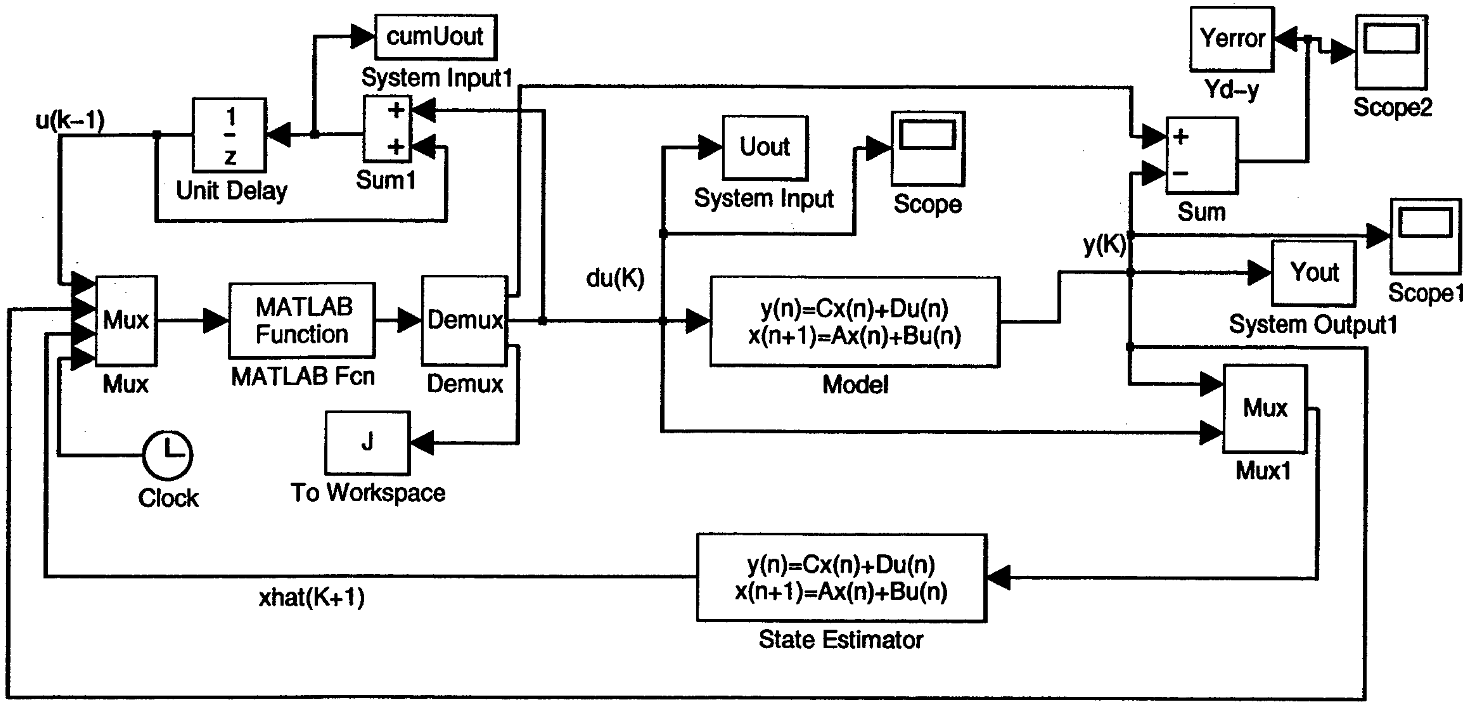

Figure 1. Inverse Control SIMULINK Diagram

The MPC-determined input is passed to the aircraft model where the actual system output is determined. This output, coupled with the previous input, is passed to the state estimator, and the process begins again with the previous inputs, outputs, and states being sent to the controller (MATLAB function), where the next input to perform is determined. The current simulation time is also an input to the MPC so the controller can determine where in time along the setpoint trajectory the simulation is, and the setpoints for the current calculation can be deduced from that. This allows for the prediction horizon used in the MPC calculations to be both less than and shifted along the given setpoint trajectory.

The MATLAB command *qp* was used to solve the quadratic problem of Equations (19) and (31) once the state prediction matrices were calculated. The matrices constant across the horizons were calculated prior to executing the SIMULINK program, while those requiring updates across time were calculated within the MATLAB functions (see Appendix A).

Once the MPC SIMULINK program was used to calculate the four input trajectories, $\delta_e, \delta_T, \delta_a$, and δ_r, that optimally minimized the error between the derived outputs and the setpoint trajectory, a second SIMULINK program was utilized to calculate all of the outputs which were contained in the FDR information (see Appendix B). This second model allows inclusion of the initial conditions of the flight since the model used in the MPC program was initialized to steady flight. In this manner, the inputs required to drive a selection of the outputs to the setpoints are determined, while the effect those inputs have on the entire set of known output profiles can also be found.

IV. Validation: Distillation Column

This section presents the results from applying the inverse model predictive control program described in the previous sections to a distillation system. This serves as a validation of the code and algorithm developed through comparison to results found under a previous study.

4.1 Scenario

In Morari's technical notes (9), predictive control, based on a step response model, is performed on a high-purity distillation column. His work provides a unique opportunity to validate the results obtained from the algorithm developed for this thesis. The algorithm developed for this thesis will be applied to the same system, under the same weighting and horizon conditions, to validate the program developed. Four simulations were run, each varying a combination of the input weights, the output weights, the prediction and control horizons, and the setpoints.

4.2 Model

The model used was a two-input, two-output transfer function system

$$\frac{1}{75s+1}\begin{bmatrix} 0.878 & 0.864 \\ 1.082 & 1.096 \end{bmatrix} \tag{38}$$

where the first state is reflux and the second state is buildup. This thesis utilizes state space models so the transfer function model given was transformed into a state space model utilizing MATLAB's *tf2ss* command. This results in a model in the form of Equation (8), for continuous time, where

$$A = \begin{bmatrix} -.0133 & 0 \\ 0 & -.0133 \end{bmatrix} \; B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \; C = \begin{bmatrix} .0177 & .0155 \\ .0144 & .0146 \end{bmatrix} \; D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

4.3 Simulation

The first set of results determine the effect of the control weight matrix, R. These effects were obtained by setting the setpoints to $[1\ 0]^T$ throughout the horizon, $q = 5, p = 20, Q = I$, and letting R, the input weight, equal 0 and I. For this 2-input, 2-output system then,

$$R = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{39}$$

The results obtained are seen in Figure 2. As expected, with the input weight set to 0, the controller utilizes large control inputs to drive the outputs to their setpoints very quickly; while, when the input weight is set to I, the response is much slower since less control power is being used.

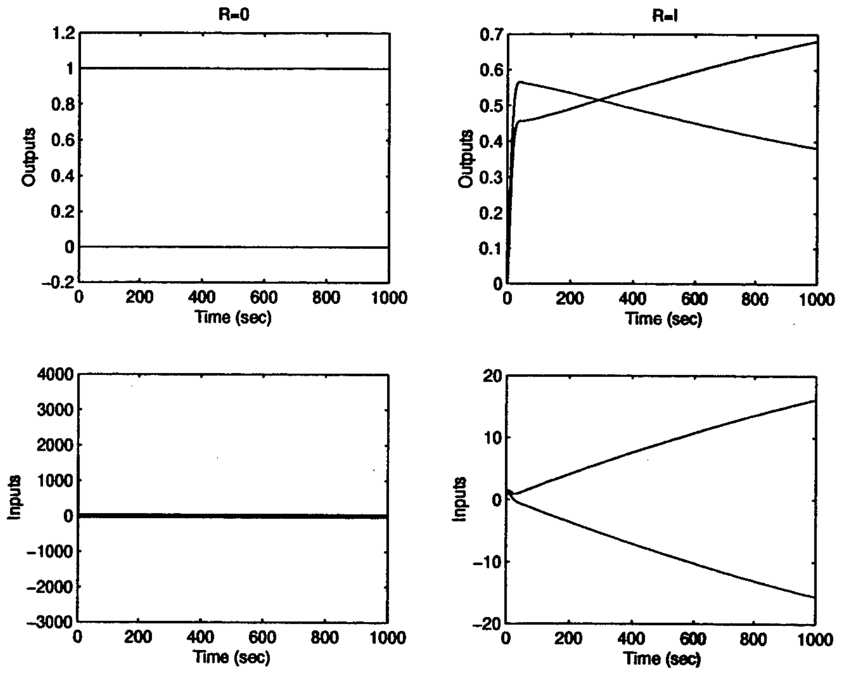

Figure 2. Distillation: Effect of control weight matrix

Next, the effects of the output weight matrix, Q, are determined. Both horizons and the setpoints are kept the same as the previous set of simulations, R is set to the identity, and two values are used to find the output weight effects:

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} \tag{40}$$

With these values, the validation results are found in Figure 3. From this figure, it is evident that the output associated with the higher weight (100) is being driven to its setpoint much faster than the other output, which has a weight of only one.

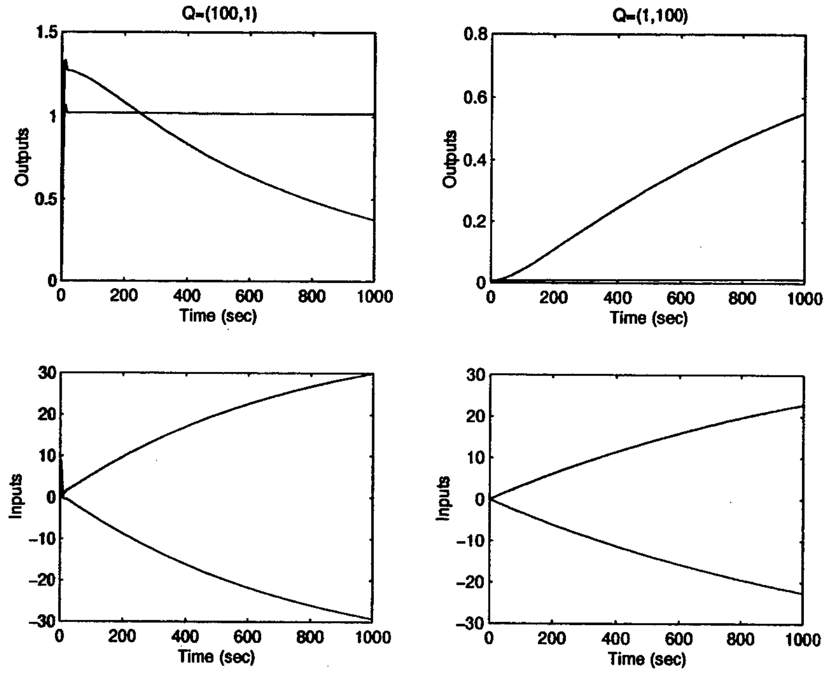

Figure 3. Distillation: Effect of output weight matrix

With the weighting matrices' effects found, it is then possible to determine the effects of varying the setpoints. This is done using four different setpoint values, r, remaining constant throughout the trajectory:

$$r = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, r = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, r = \begin{bmatrix} 0.88 \\ 1.12 \end{bmatrix}, r = \begin{bmatrix} 0.39 \\ 0.59 \end{bmatrix} \tag{41}$$

For all simulations, $q = 5, p = 20, Q = R = I$. These results are found in Figures 4 and 5. Obviously, as the setpoint changes, the controller drives the output to the desired setpoint.

From these runs, it is evident that the code developed is valid. The results are not exactly the same as the results found by Morari, since the exact format of the performance index used in the previous study is not known, and is most likely not identical to the index used in this thesis. Therefore, the equation being minimized is not the same, and slightly differing results can be expected. From the plots, however, it is obvious that the general trends obtained from the algorithm developed are the same as those found by Morari.

It is also important to note that the distillation model used for these validation runs is very different from the typical dynamic model for an aerospace system. The distillation model is smaller (two states versus the typical eight to ten) and responds much slower than expected from an aircraft model. These differences aside, the validation runs are useful for two reasons. First, as stated previously, knowing the results of a completed model predictive control study allowed validation of the algorithm. Second, utilizing this model shows the usefulness of applying model predictive control to a variety of models.

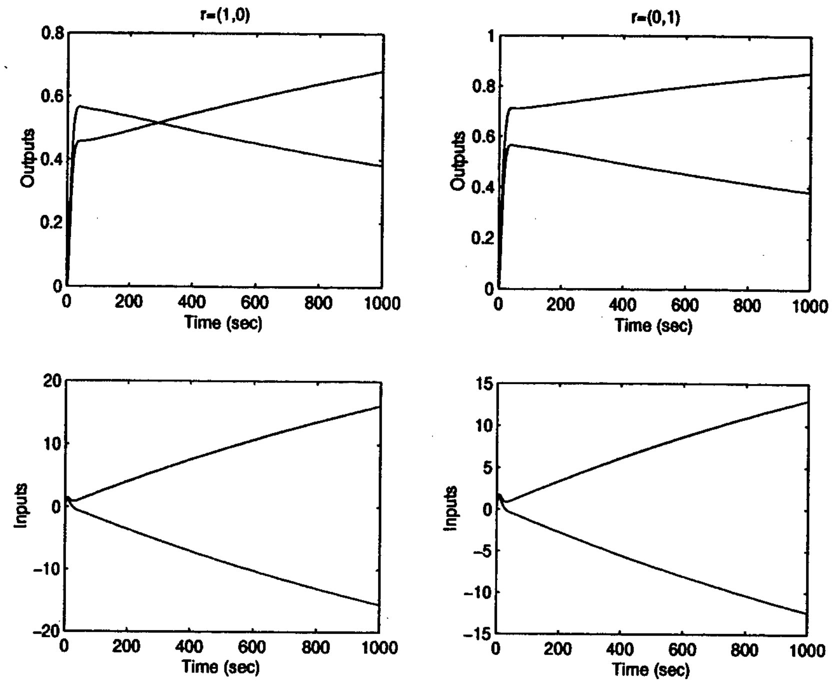

Figure 4. Distillation: Effect of setpoint changes ($r = [1,0]'$, $r = [0,1]'$)

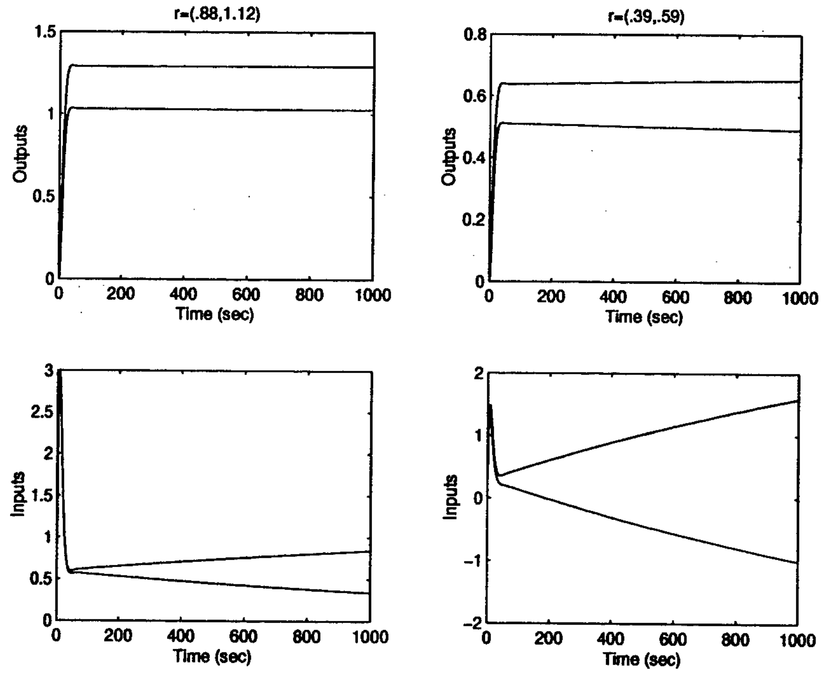

Figure 5. Distillation: Effect of setpoint changes ($r = [.88, 1.12]'$, $r = [.39, .59]'$)

V. Application: Transport Aircraft

This section presents the results from applying the inverse model predictive control program to a transport aircraft. Two actual aircraft mishaps, Flight 427 and N827AX, will be analyzed to find the degree to which the input histories can be determined from the FDR data.

5.1 Flight 427

5.1.1 Scenario. On 8 September, 1994, a Boeing 737-300 on Flight 427 crashed while on approach into Pittsburgh International Airport. The aircraft rolled, turned, and nosed over into the ground. The FDR data obtained from the crash contains all the relevant operating conditions of the aircraft (altitude, velocity, heading, etc.) but none of the control surface deflections.

In an attempt to determine these inputs, Parks, Bach, and Shin (12) utilized the Flight 427 information and an assumed set of control inputs to reproduce the outputs that were provided from the FDR in this aircraft accident. Napolitano (10) analyzed the same aircraft and used a neural network simulator and "Virtual" FDR to reconstruct control surface deflections.

5.1.2 Model. A detailed model of the Boeing 737 is not available in open literature, understandably, due to its proprietary status. However, an accurate model estimation can be made from aircraft data more widely available in the current literature.

The FDR data obtained for this particular incident suggested that the aircraft was in a steady descent at about 6700 ft 100 seconds before impacting the ground. Therefore, approximate stability derivatives (13) were used in the linearized equations of motion to provide for continuous-time aircraft dynamics, where in state-space form (11)

$$\dot{x} = Ax + Bu \tag{42}$$

where

$$A = \begin{bmatrix} A_{long} & 0 \\ 0 & A_{lat} \end{bmatrix} B = \begin{bmatrix} B_{long} & 0 \\ 0 & B_{lat} \end{bmatrix} \tag{43}$$

And the A and B matrices are defined in terms of conventional stability derivatives (see Appendix C) as

$$A_{long} = \begin{bmatrix} X_u & X_w & 0 & -g \\ Z_u & Z_w & u_0 & 0 \\ M_u + M_{\dot{w}} Z_u & M_w + M_{\dot{w}} Z_w & M_q + M_{\dot{w}} u_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{44}$$

$$B_{long} = \begin{bmatrix} X_{\delta_e} & X_{\delta_T} \\ Z_{\delta_e} & Z_{\delta_T} \\ M_{\delta_e} + M_{\dot{w}} Z_{\delta e} & M_{\delta_T} + M_{\dot{w}} Z_{\delta T} \\ 0 & 0 \end{bmatrix} \tag{45}$$

$$A_{lat} = \begin{bmatrix} Y_\beta/u_0 & Y_p/u_0 & -(1 - Y_r/u_0) & gcos(\theta_0)/u_0 & 0 \\ L_\beta & L_p & L_r & 0 & 0 \\ N_\beta & N_p & N_r & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{46}$$

$$B_{lat} = \begin{bmatrix} 0 & Y_{\delta_r}/u_0 \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{47}$$

The aircraft states are

$$\begin{aligned} x &= [x_{long}^T \ \ x_{lat}^T]^T \\ x_{long} &= [\Delta u \ \Delta w \ \Delta q \ \Delta\theta] \\ x_{lat} &= [\Delta\beta \ \Delta p \ \Delta r \ \Delta\phi \ \Delta\psi] \end{aligned} \tag{48}$$

where

$$\begin{aligned} \Delta u &= forward\ velocity\ change, ft/s \\ \Delta w &= vertical\ velocity\ change, ft/s \\ \Delta q &= pitch\ rate\ change, rad/s \\ \Delta\theta &= pitch\ angle\ change, rad \\ \Delta\beta &= sideslip\ angle\ change, rad \\ \Delta p &= roll\ rate\ change, rad/s \\ \Delta r &= yaw\ rate\ change, rad/s \\ \Delta\phi &= roll\ angle\ change, rad \\ \Delta\psi &= yaw\ angle\ change, rad \end{aligned} \tag{49}$$

and the inputs are

$$u = [\delta_e \ \delta_T \ \delta_a \ \delta_r]^T \tag{50}$$

where

$$\begin{aligned} \delta_e &= elevator\ angle, rad \\ \delta_T &= throttle\ position, percent \\ \delta_a &= aileron\ angle, rad \\ \delta_r &= rudder\ angle, rad \end{aligned} \tag{51}$$

The output equation $y = Cx$ can then be formed depending on which states, or linear combinations thereof, are of interest later.

Additionally, the following equations will be utilized to produce outputs included in the FDR that are not simple linear combinations of the states described above. The absolute velocity, rate of climb/descent, forward and vertical accelerations are, respectively,

$$\begin{aligned}
\Delta V_T &= \sqrt{\Delta u^2 + \Delta v^2 + \Delta w^2} \\
\Delta \dot{h} &= u_o \Delta\theta - \Delta w \\
\Delta g_x &= \Delta \dot{u}/g \\
\Delta g_z &= (\Delta \ddot{h} + \ell\ddot{\theta})/g
\end{aligned} \tag{52}$$

5.1.3 Simulation. For this application, both rate and non-rate performance indices were used to determine the input histories which produced the data contained in the FDR. The heading, pitch, and roll angles, ψ, θ, and ϕ, were chosen to be included in the performance index calculations since they carry most of the attitude and location information for the aircraft. The final value (at $t = 100$s) was extended for just enough time for the prediction horizon to allow for one final input calculation at a simulation time of $t = 99$s. Before being included in the setpoint vector passed to the MPC, the angles were resampled to the same discretization interval as the plant model, ΔT=2 sec, and linearized by subtracting the initial value of each angle throughout.

5.1.3.1 Non-rate Performance Index Results. Using the algorithm as explained in Section 3.5, and with the following parameter values:

$$\begin{aligned}
L &= observer\ gain\ matrix \\
Q &= diag(100 \cdots 100) \\
R &= diag(0.1 \cdots 0.1) \\
p &= 20 \\
q &= 15
\end{aligned} \tag{53}$$

the four inputs $(\delta_e, \delta_T, \delta_a, \delta_r)$ that optimally drive the outputs (ψ, θ, ϕ) to the setpoints are found. According to the developed algorithm, these inputs are next used in conjuction with the second SIMULINK program (see Appendix B) in order to determine the full set of outputs that are included in the FDR data. These calculated outputs are shown, along with their associated FDR output, in Figures 6 through 12.

These figures show the good agreement between the outputs obtained from the MPC-inverse control program and those from the FDR. The average deviation from the setpoints, across the entire trajectory, is only 0.4909, 0.0637, and 0.1786 radians for ψ, θ, and ϕ, respectively. However, the disadvantages of working with a linear model are evident in these plots. Especially when analyzing the acceleration deviations in Figures 8 and 9 it is apparent that the linear model used here is having difficulty matching the FDR data. In fact, Parks, Bach, and Shin (12) found that they had to change the lift and drag characteristics of the aircraft as it approached stall in order to match this data. The attempt here did not include this manipulation, without significant degradation of results. The poor matching with the accelerations is due to the lack of lift and drag manipulation in this study and due to uncertainty in the distance from the measurement accelerometer (in the right main wheel well) to the aircraft center of gravity. However, overall, the match between the results and the data is very good.

The optimal inputs which produced the above output results are shown in Figures 13, 14, 15, and 16. Upon analyzing the four inputs, elevator, throttle, aileron, and rudder, from the Flight 427 results, it appears that the cause of deviation from controlled flight was a rudder hard-over input. This is consistent with several other sources. Most notably, Parks, Bach, and Shin found that they needed to input a two or three-step rudder command, much as seen in Figure 16, in order to obtain output profiles similar to the FDR. Moreover, the results found here are analytical – not assumed input trajectories as others have been.

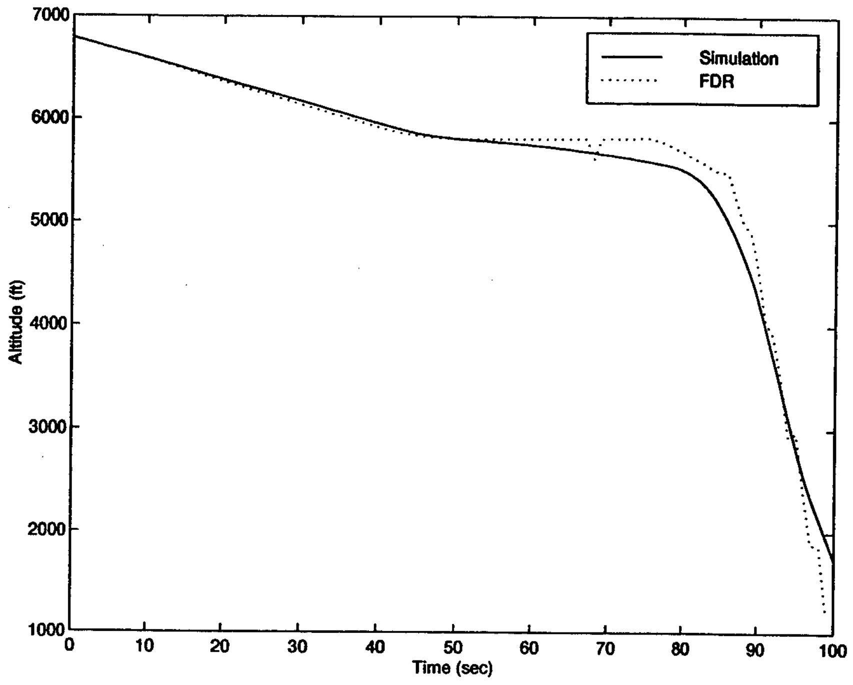

Figure 6. Output Response: h

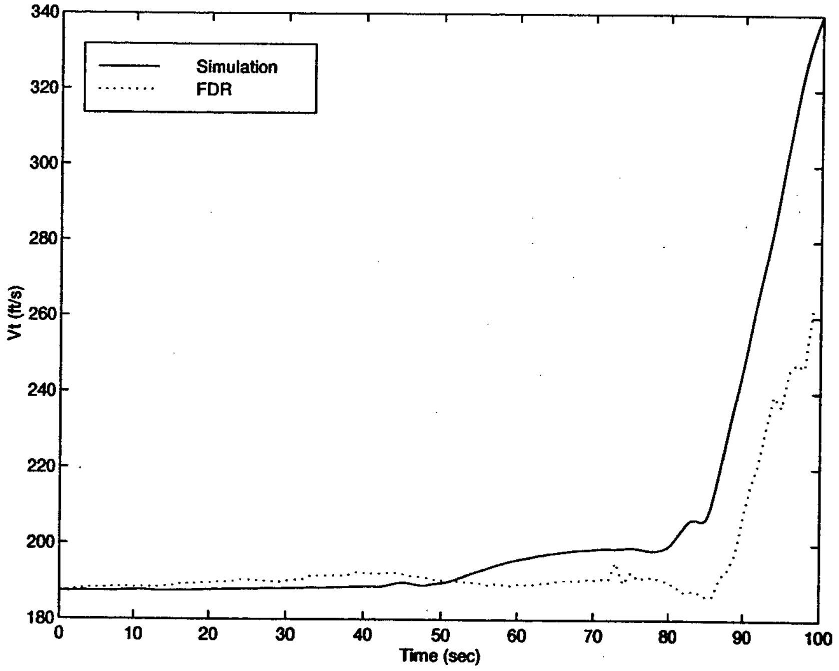

Figure 7. Output Response: V_T

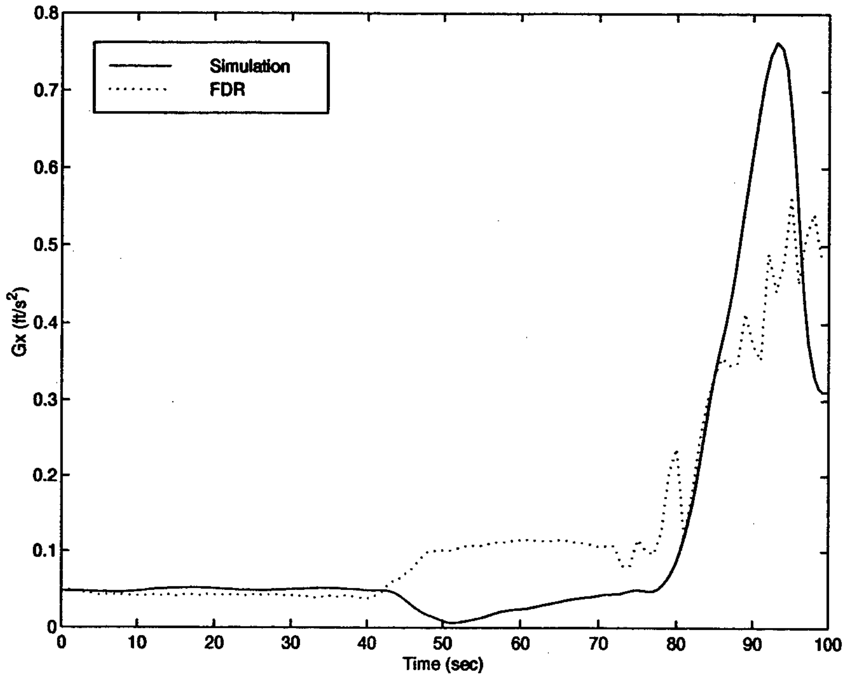

Figure 8. Output Response: g_x

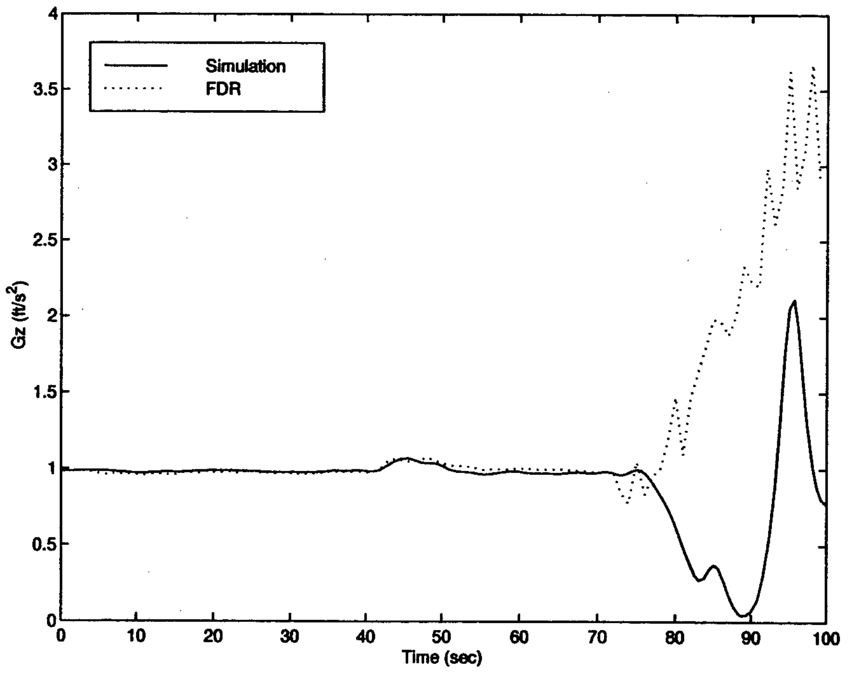

Figure 9. Output Response: g_z

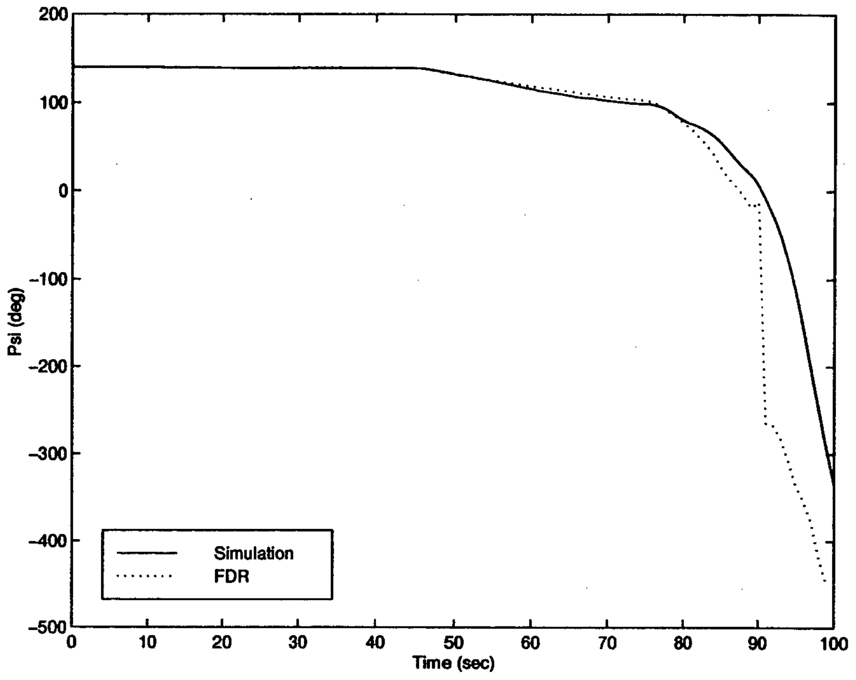

Figure 10. Output Response: ψ

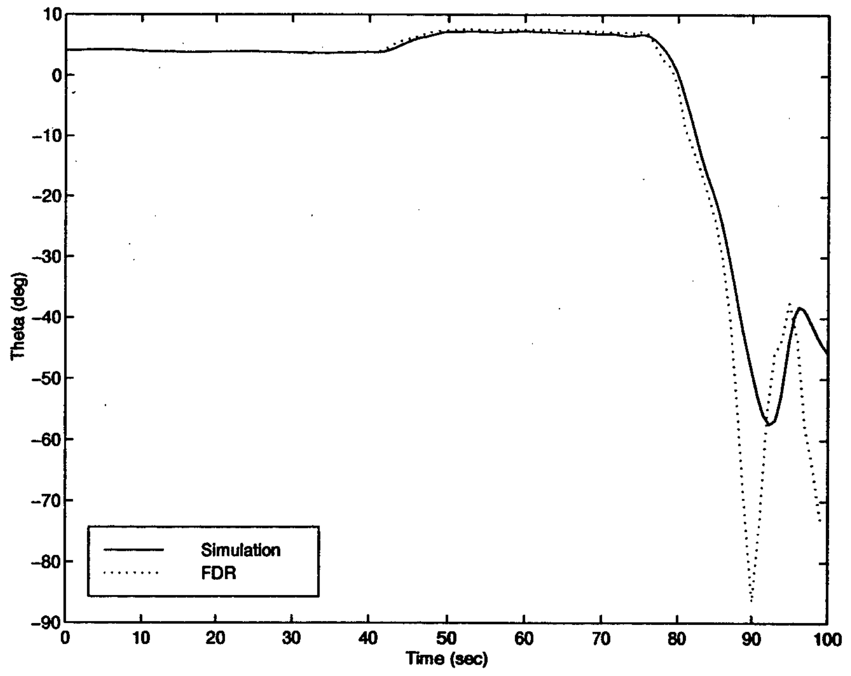

Figure 11. Output Response: θ

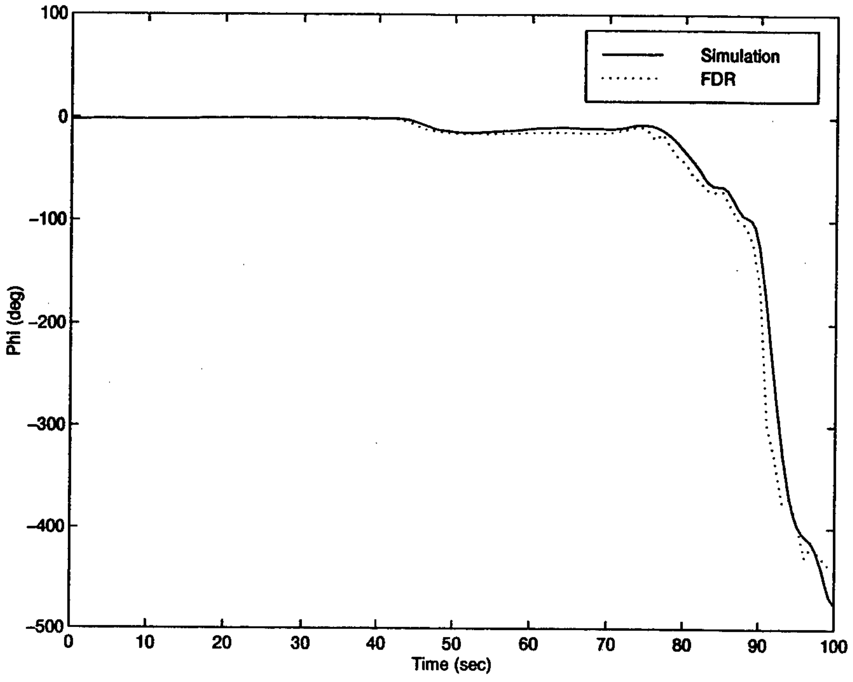

Figure 12. Output Response: ϕ

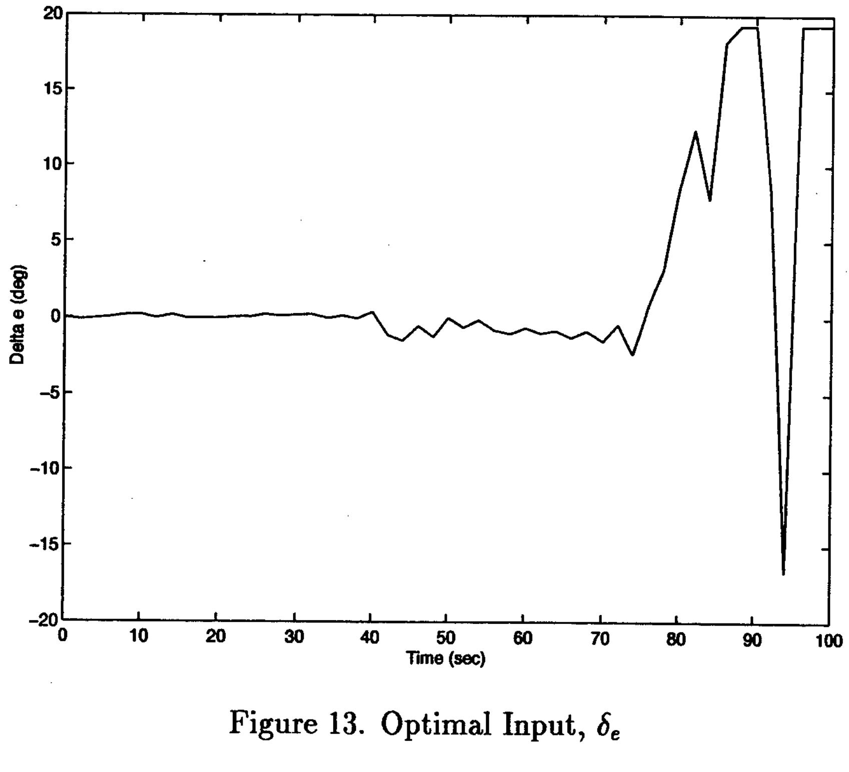

Figure 13. Optimal Input, δ_e

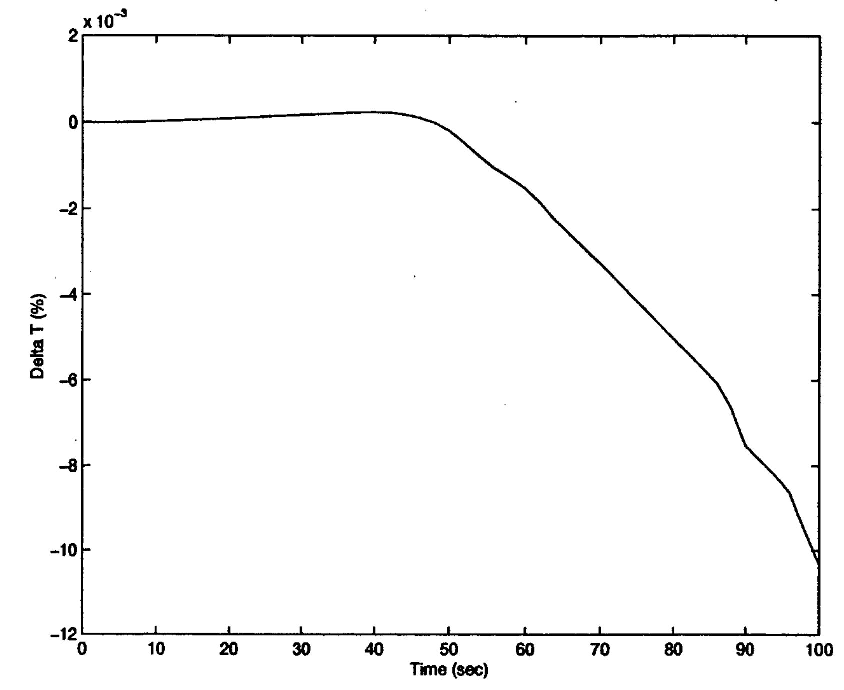

Figure 14. Optimal Input, δ_T

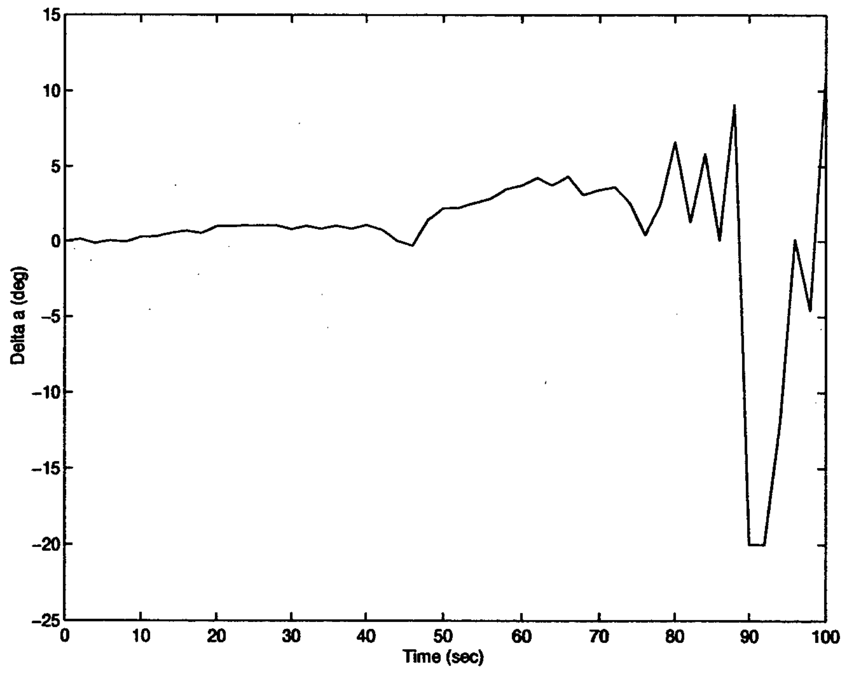

Figure 15. Optimal Input, δ_a

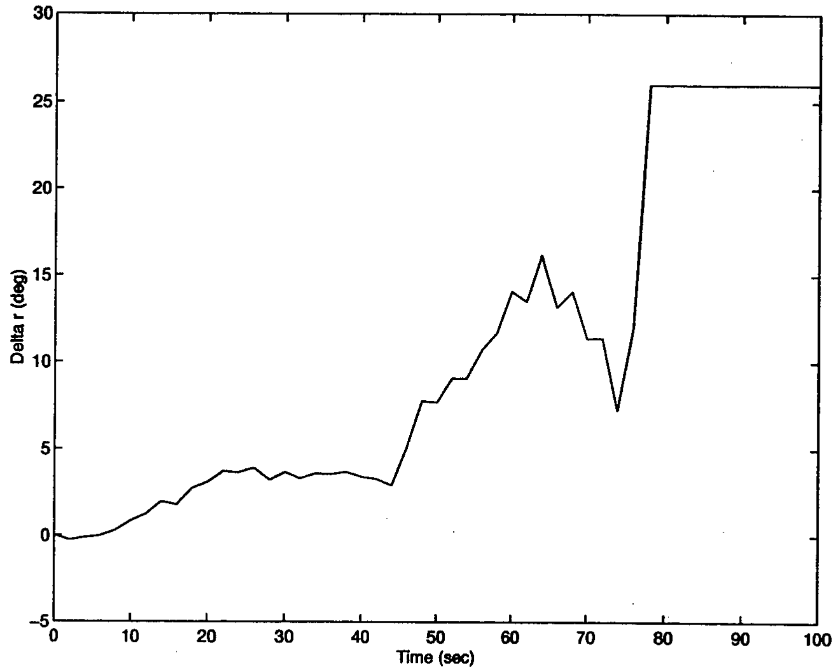

Figure 16. Optimal Input, δ_r

5.1.3.2 Rate Performance Index Results. Using the same solution algorithm and setpoints as in the previous section, but replacing the performance index with the terms including the rate term, Δe, in the MPC the results of the Flight 427 problem can be determined and compared to the results from the MPC without the rate term included. With the following parameter values (Note $term_R$ distinguishes parameters used in the rate-term calculations):

$$\begin{aligned} L_R &= observer\ gain\ matrix \\ Q_R &= diag(100 \cdots 100) \\ R_R &= diag(0.1 \cdots 0.1) \\ E &= diag(100 \cdots 100) \\ p_R &= 15 \\ q_R &= 15 \end{aligned} \tag{54}$$

the four inputs $(\delta_{e_R}, \delta_{T_R}, \delta_{a_R}, \delta_{r_R})$ that optimally drive the outputs (ψ, θ, ϕ) to the setpoints are found, just as in the solution for the non-rate performance index. Once again, these inputs are used in conjuction with the seven-output program to determine the full aircraft response. These calculated outputs are shown, along with their associated FDR output, in Figures 17 through 23. Plots of the four optimal inputs which produced these outputs are in Figures 24, 25, 26, and 27.

Only small differences can be seen between these plots and the results from the previous simulation. However, it is evident that there is improved setpoint tracking from using the rate-included performance index. Figures 21 through 23 show less deviation from the setpoint than Figures 10 through 12. In comparison to the non-rate index setpoint deviations, the average setpoint deviations across the entire trajectory here are slightly less. The following table details these differences. The improvement is small, however, and may not warrant the increased calculation time required for the additional terms. (Run times on a Sun platform doubled from approximately 45 to 90 seconds with inclusion of the rate term.)

Table 1. Rate and Non-rate Deviation Comparison

Setpoint	Rate deviation	Non-rate deviation
ψ	0.4828	0.4909
θ	0.0626	0.0637
ϕ	0.1539	0.1786

The lack of significantly differing results, compared to the non-rate performance index, may be due to the fact that the inclusion of the rate term amounts to changing the Q weight. The introduction of the error weight matrix, E, causes a direct increase in the Q weight, and its inclusion in the cross terms resulting from the performance index expansion, along with the time shifts between the $\mathcal{F}$ and $\tilde{\mathcal{F}}$, as well as the other matrices, introduce what is equivalent to a non-constant Q weight across time. So slightly better results (due to higher weighting) but not significantly better results can be expected.

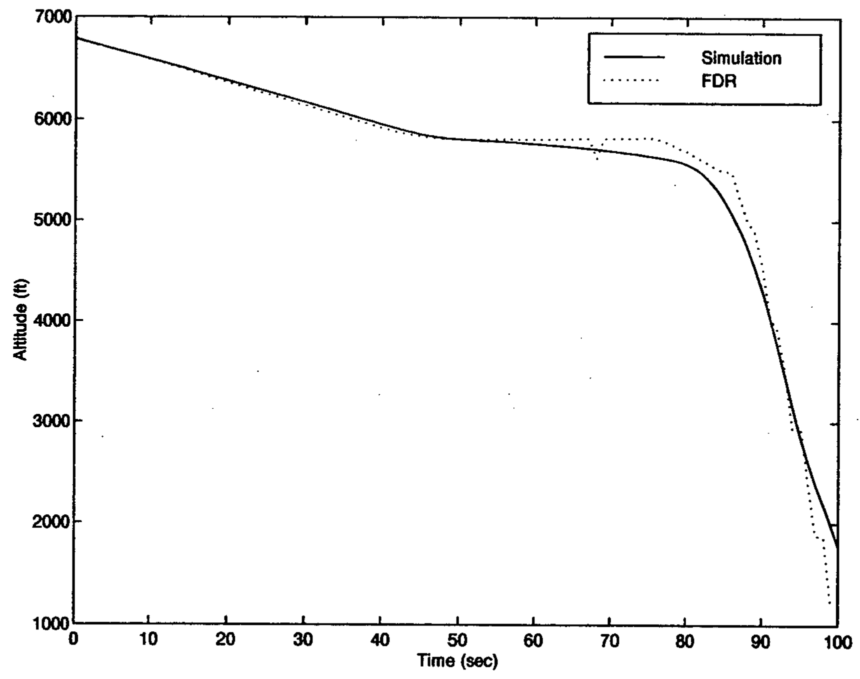

Figure 17. Output Response: h_R

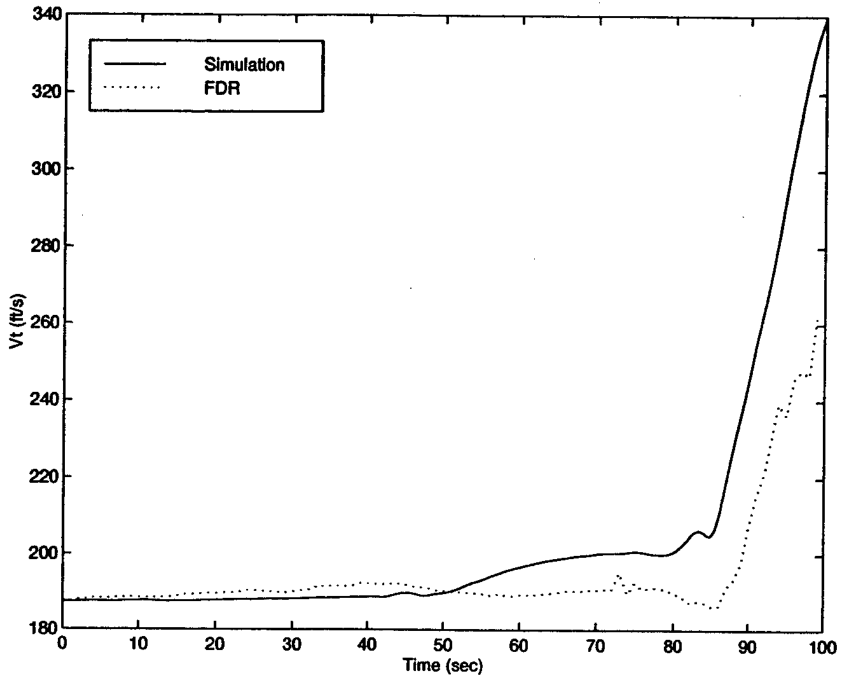

Figure 18. Output Response: V_{T_R}

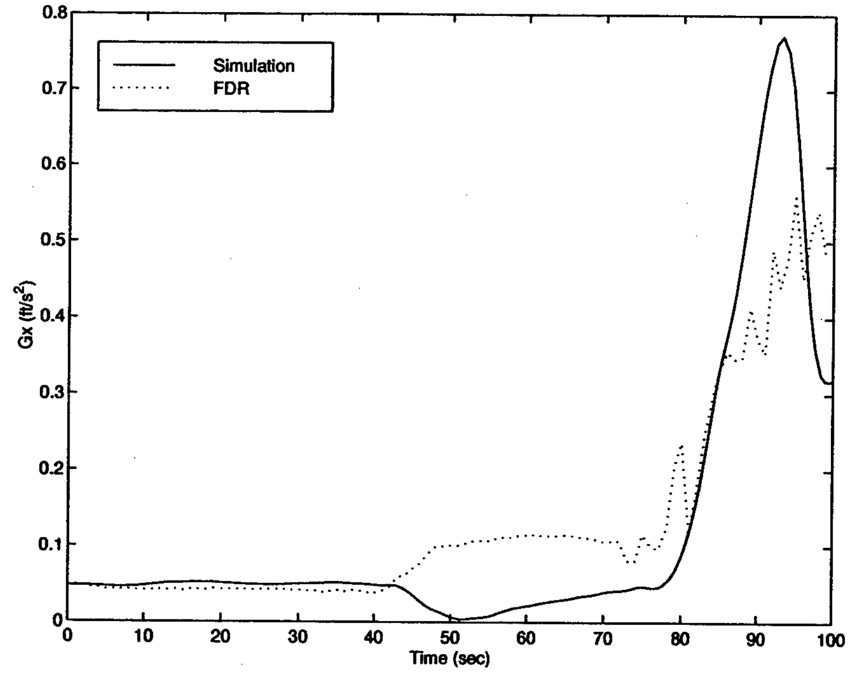

Figure 19. Output Response: g_{x_R}

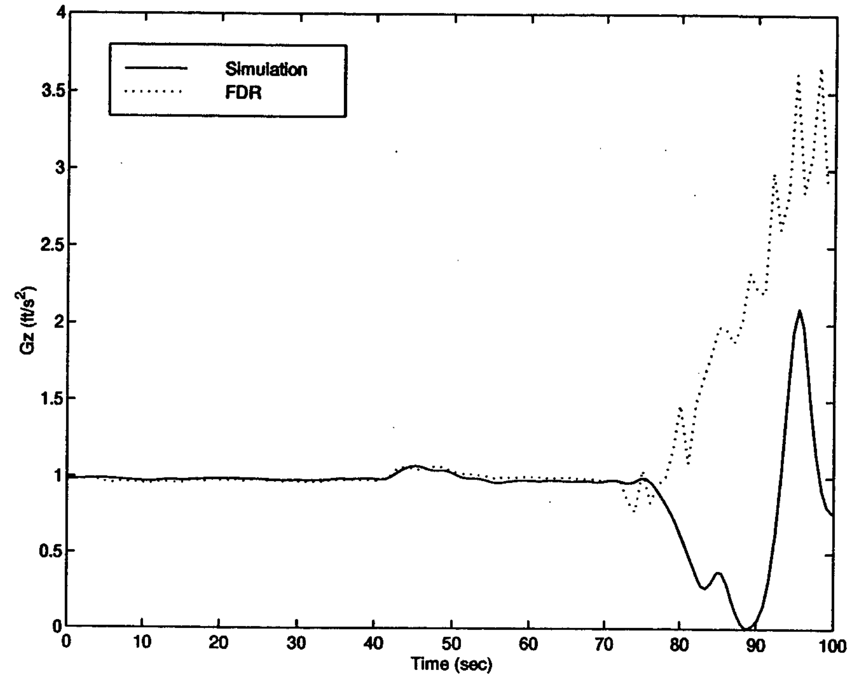

Figure 20. Output Response: g_{z_R}

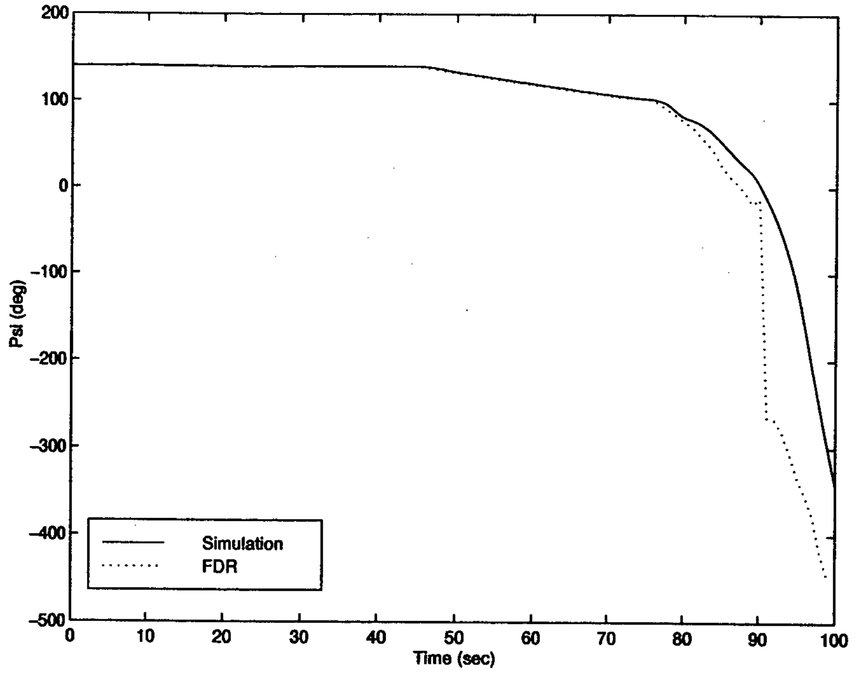

Figure 21. Output Response: ψ_R

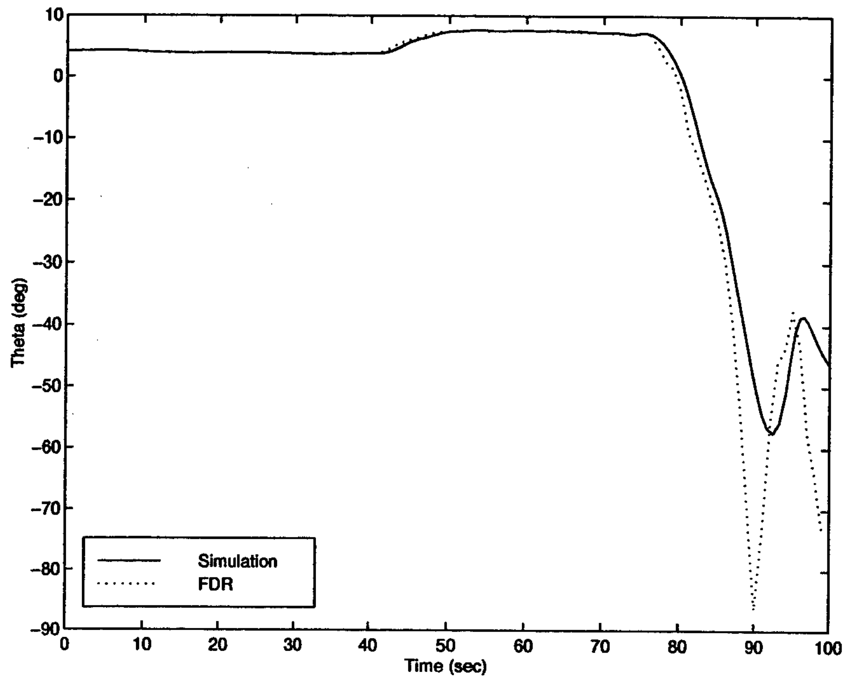

Figure 22. Output Response: θ_R

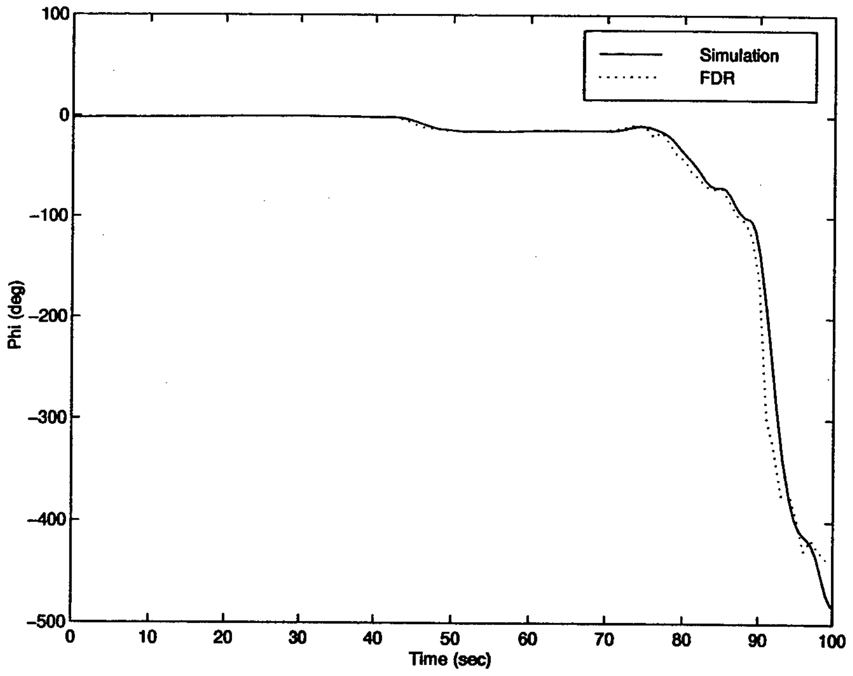

Figure 23. Output Response: ϕ_R

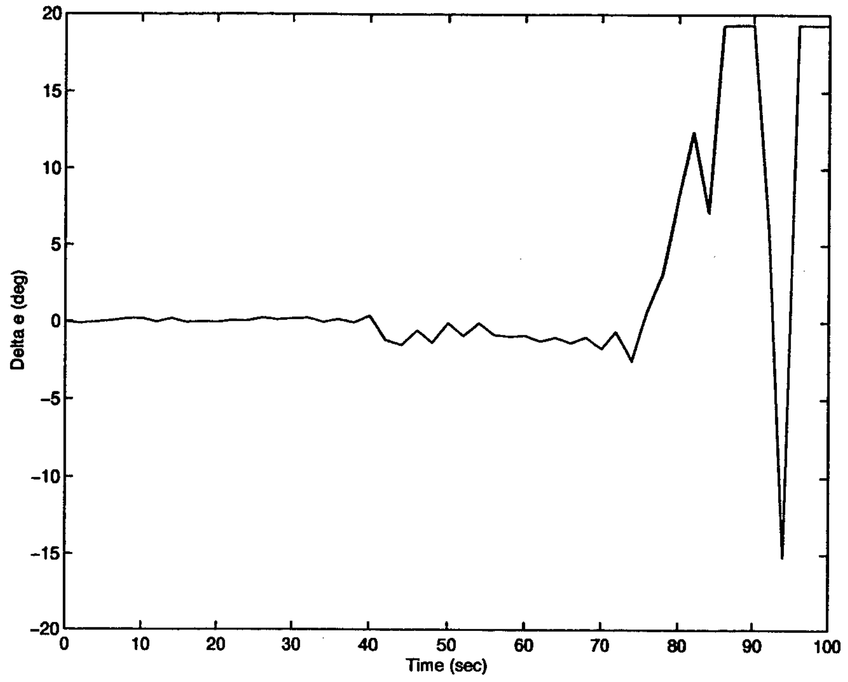

Figure 24. Optimal Input, δ_{e_R}

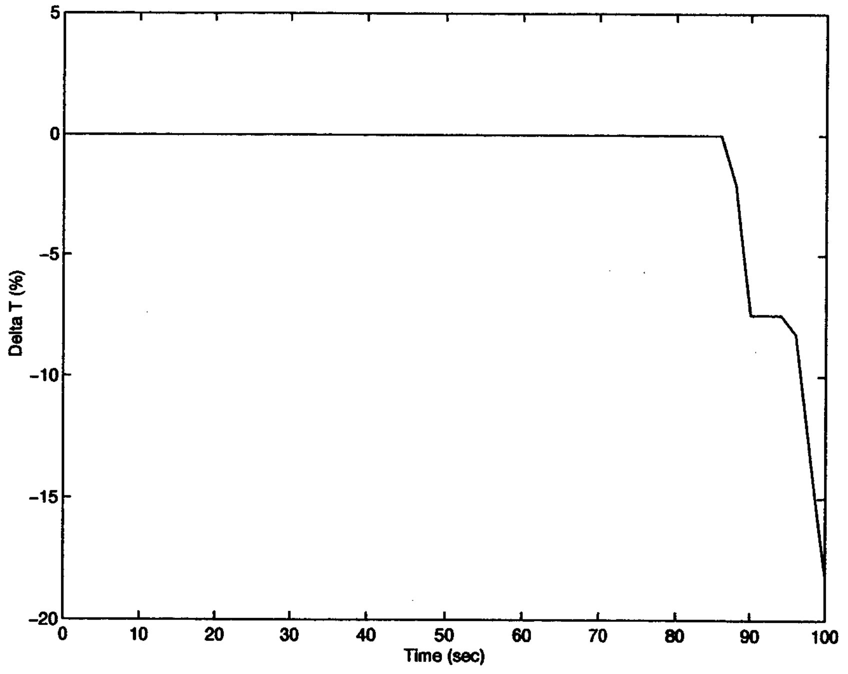

Figure 25. Optimal Input, δ_{T_R}

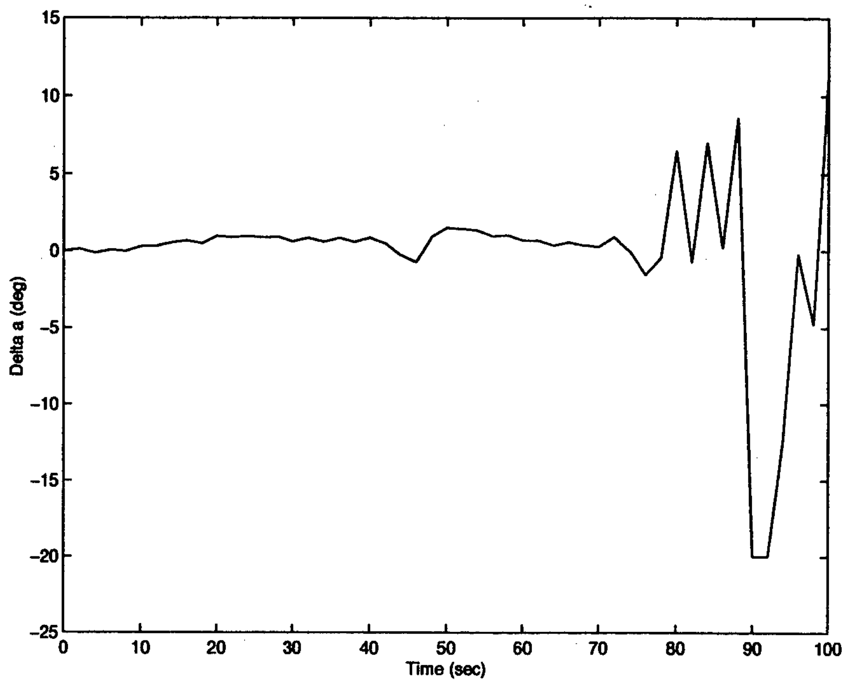

Figure 26. Optimal Input, δ_{a_R}

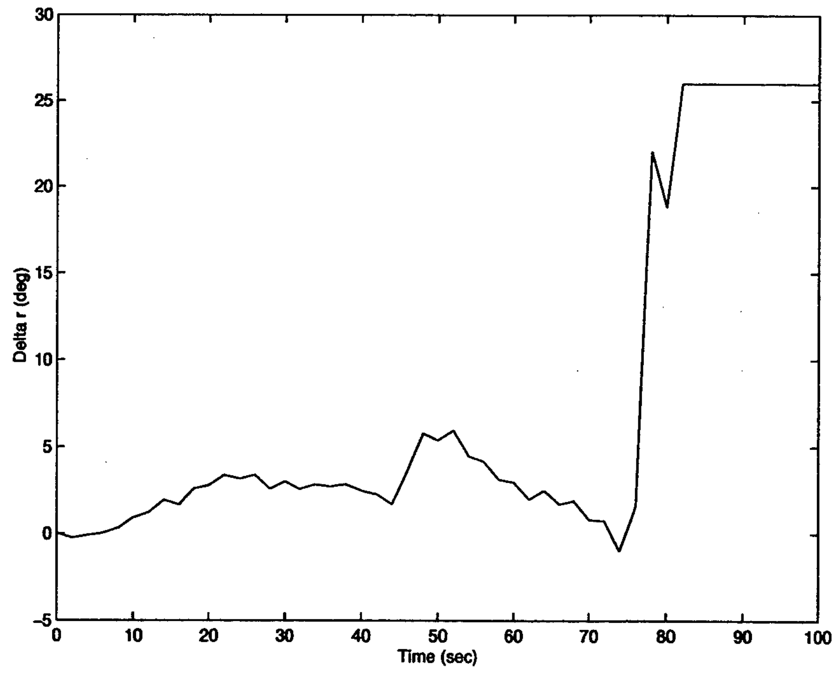

Figure 27. Optimal Input, δ_{r_R}

5.2 N827AX

5.2.1 Scenario. On 22 December, 1996, an Airborne Express Douglas DC-8-63F impacted mountainous terrain at about 3400 feet in Narrows, Virginia. The crew was performing stall maneuvers on a test flight when, after one of the planned stalls, the aircraft plunged unrecoverably into the mountains. As with Flight 427, the FDR data obtained from the crash includes all of the operating conditions of the flight except for the control surface deflections. This FDR data will be used to determine the control histories during the flight in an attempt to determine what might have caused the incident.

5.2.2 Model. The same model that was utilized for the Flight 427 calculations will be incorporated here. Although that model was based on flight at approximately 6700 feet and this flight is at approximately 13,500 feet, the flight characteristics (and therefore stability derivatives) at each altitude vary only slightly, so minimal differences can be expected. Additionally, although this incident involves a Douglas DC-8 and the previous calculations were for a Boeing 737, the flight characteristics for each can be well incorporated into a general transport model which is being utilized here.

5.2.3 Simulation. Only the non-rate performance index will be utilized for this flight since it was seen from previous results that the rate performance index did not provide greatly improved results. For this simulation, however, four FDR outputs will be used as the setpoints: the three flight angles, ψ, θ, and ϕ and the altitude, h. As before, the final value (at $t = 119$s) was extended just long enough in time for the prediction horizon to allow for a final calculation at a simulation time of $t = 118$s. Also, before being included in the MPC calculations, the angles were resampled to match the discretization interval of the plant, $\Delta T = 2sec$, and linearized by subtracting the initial value of each setpoint throughout time. It should be noted that while the downward spike in the altitude plot around $t = 118$s is not feasible, it was in the FDR data and will therefore be kept in the data for the calculation.

Using the algorithm as before, and with the following parameter values (Note $term_{AX}$ distinguishes terms used in the calculations for flight N827AX):

$$\begin{aligned} L_{AX} &= observer\ gain\ matrix \\ Q_{AX} &= diag(100 \cdots 100) \\ R_{AX} &= diag(0.1 \cdots 0.1) \\ p_{AX} &= 15 \\ q_{AX} &= 15 \end{aligned} \tag{55}$$

the four inputs $(\delta_e, \delta_T, \delta_a, \delta_r)$ that optimally drive the outputs (ψ, θ, ϕ, h) to the setpoints are found. Using these optimal inputs as the inputs to the seven-output SIMULINK program produces the outputs plotted in Figures 28 through 34, along with the associated FDR output.

Again, fairly good agreement is obtained between the outputs obtained from the MPC-inverse control program and the FDR data. For this simulation, average setpoint deviations across the entire trajectory are 0.2299, 0.3669, and 0.2513 radians for ψ, θ, and ϕ, respectively, and 868.9 feet for h. While the agreement is not as good as that seen from the Flight 427 results, several sources of error are entering the problem. As stated previously, the model used is the model initially developed for a Boeing 737 at a lower altitude. Stability derivatives more representative of a DC-8 aircraft at altitude would produce better results. These differences are minor, but are contributing factors to deviations from the setpoints. This compounds the linearization factors for error seen in the Flight 427 results. Here, after approximately $60s$, the aircraft begins to develop highly oscillatory flight. At these conditions, the linear assumptions at the root of the model begin to deteriorate. It is at this stage in the simulation that the outputs depart from the setpoints. Plots of the four optimal inputs which produced these results are shown in Figures 35, 36, 37, and 38.

Overall, results from the N827AX simulation show good setpoint following. The roll data is matched nearly identically, and in all other outputs, the trends that exist in the FDR data are also in the simulation results. Additionally, Figure 35 shows a stuck elevator. This is significant since a stuck elevator is the suspected cause of the incident involving N827AX. The results seen here support that preliminary conclusion.

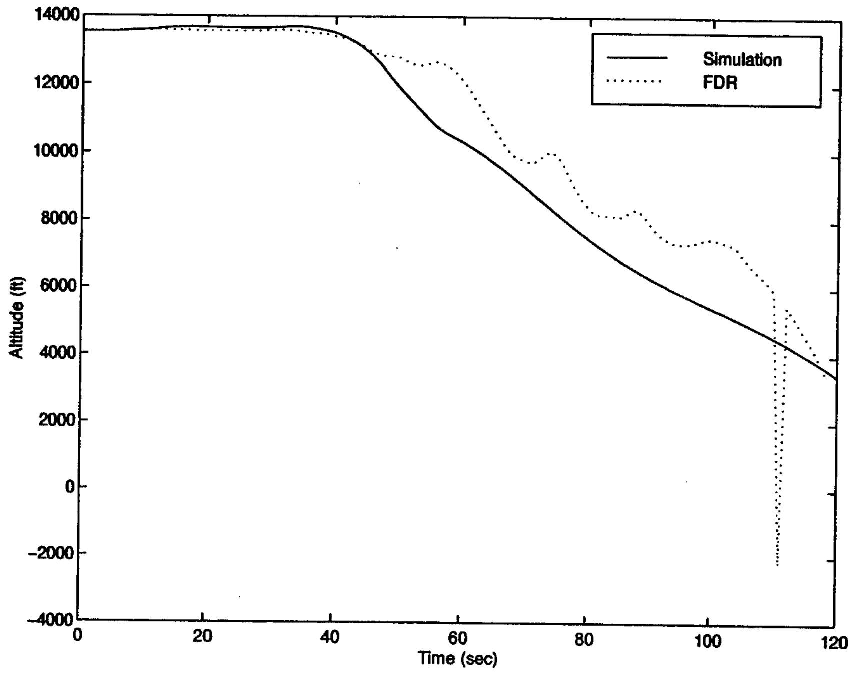

Figure 28. Output Response, h_{AX}

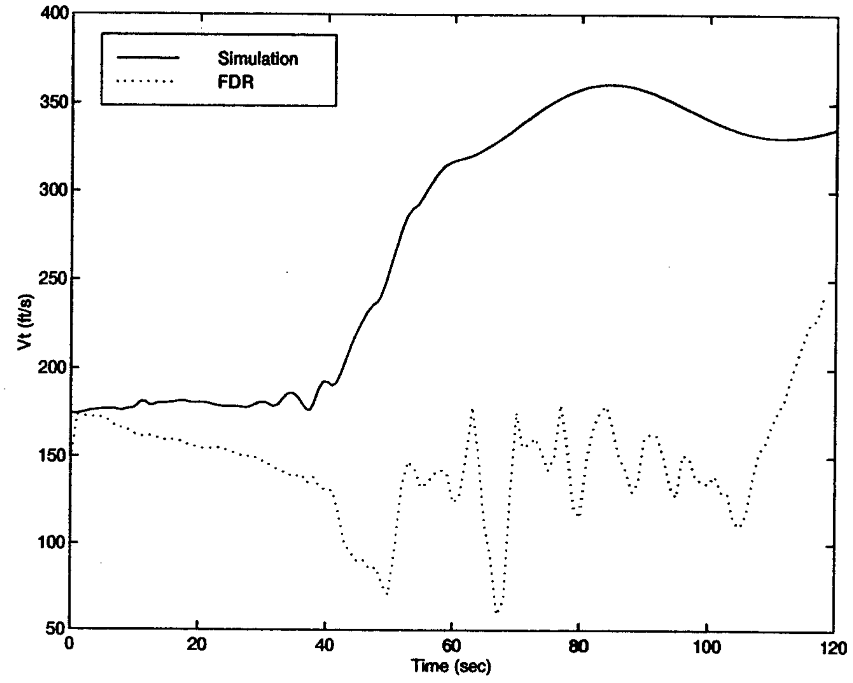

Figure 29. Output Response, $V_{T_{AX}}$

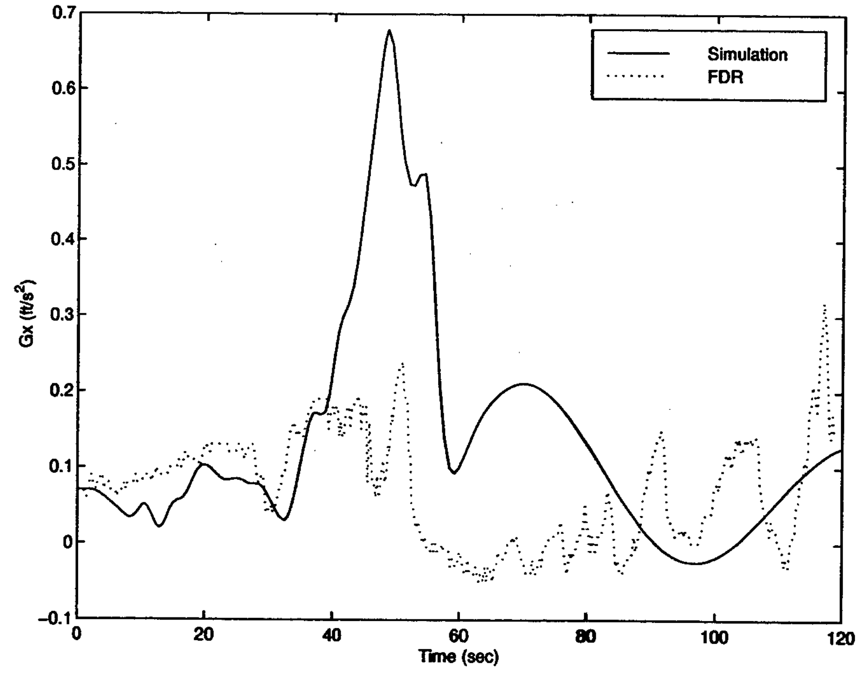

Figure 30. Output Response, $g_{x_{AX}}$

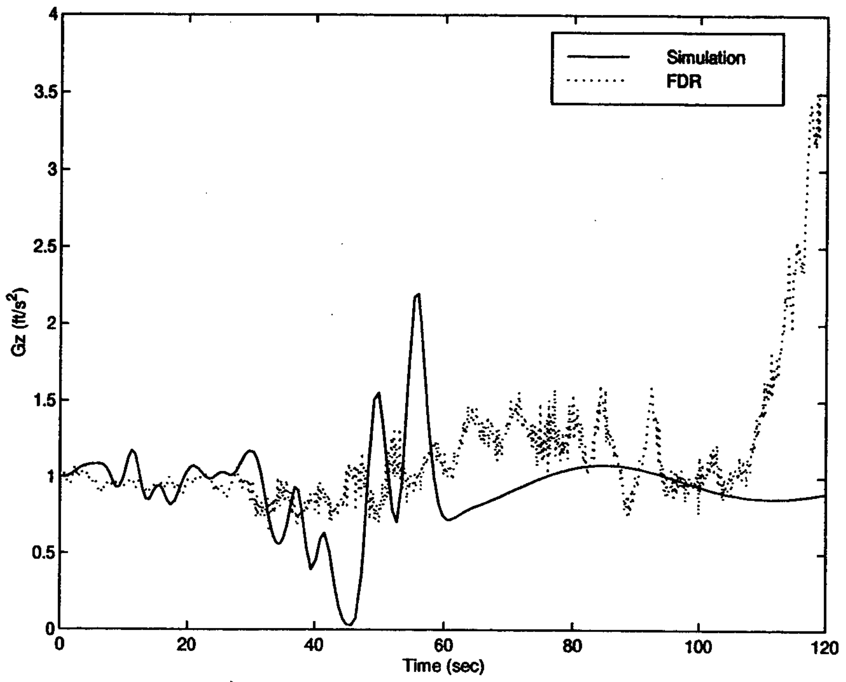

Figure 31. Output Response, $g_{z_{AX}}$

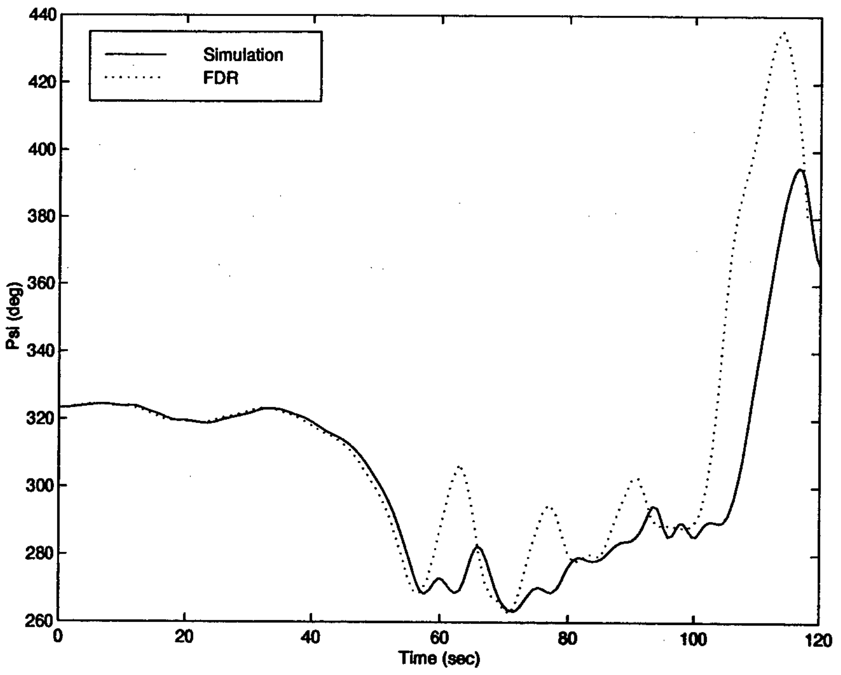

Figure 32. Output Response, ψ_{AX}

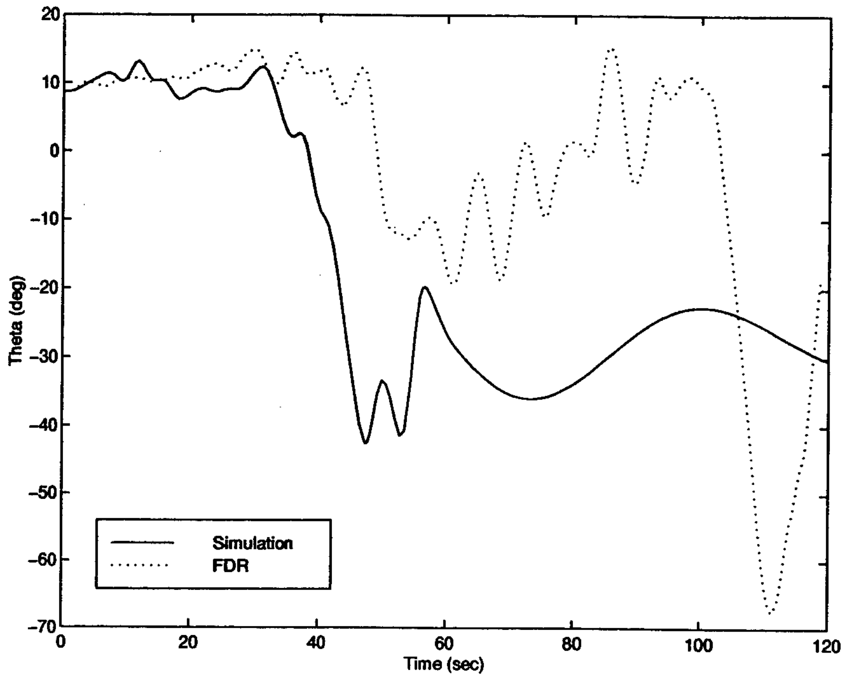

Figure 33. Output Response, θ_{AX}

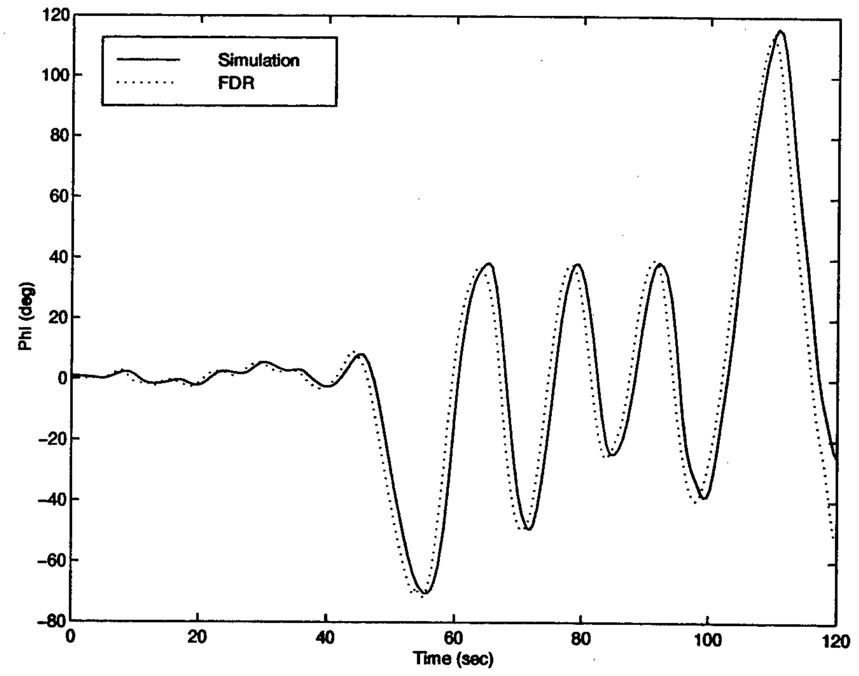

Figure 34. Output Response, ϕ_{AX}

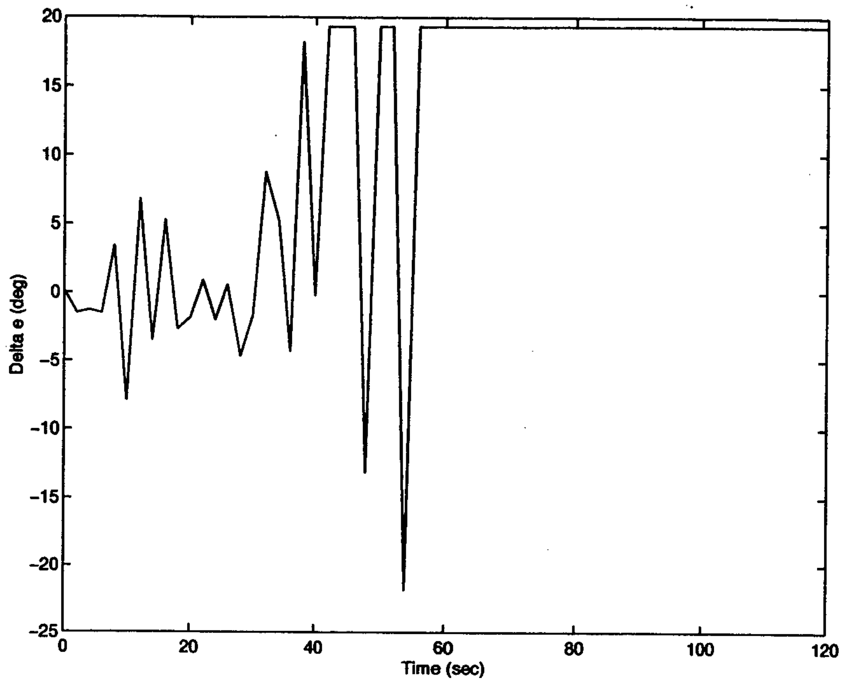

Figure 35. Optimal Input, $\delta_{e_{AX}}$

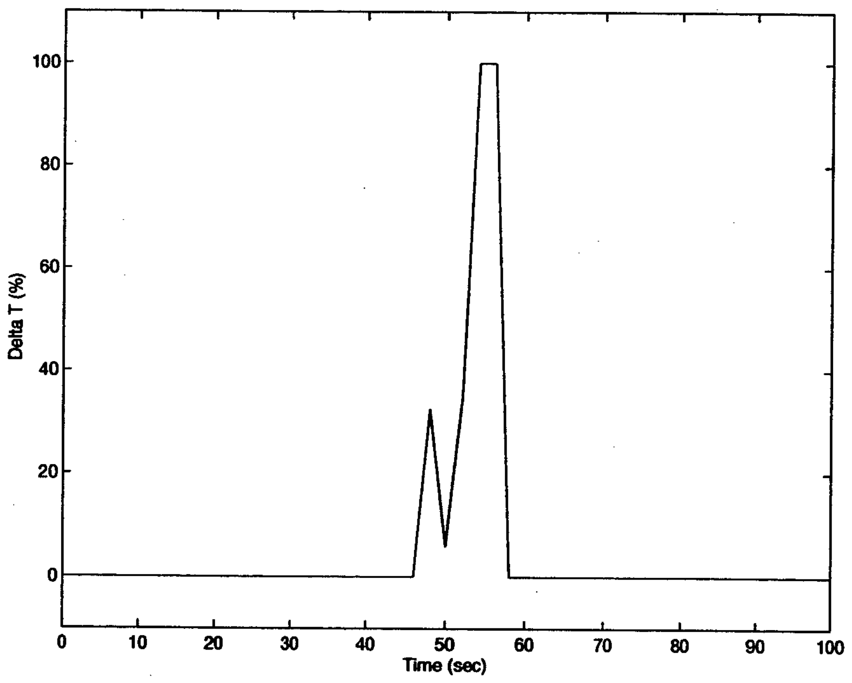

Figure 36. Optimal Input, $\delta_{T_{AX}}$

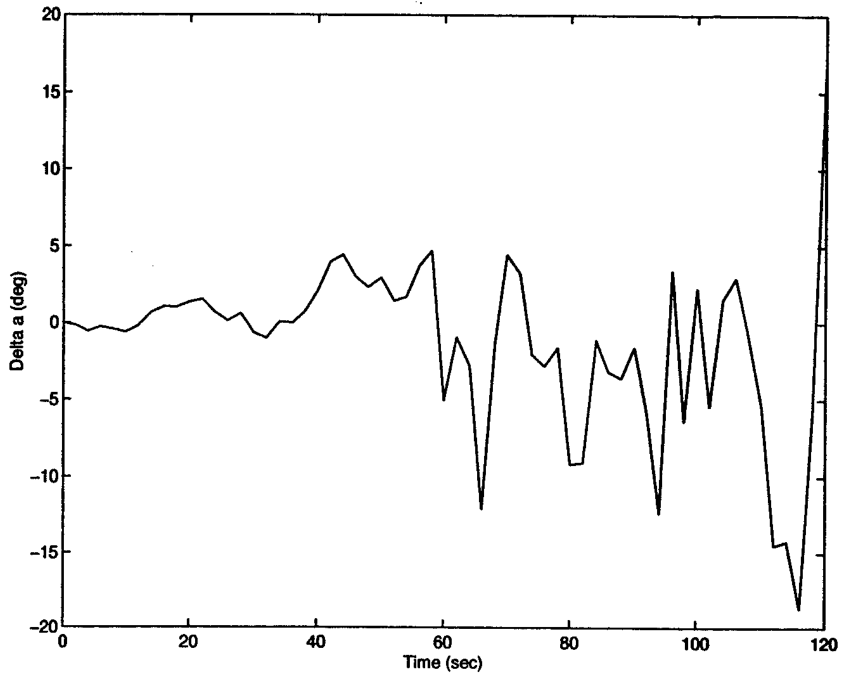

Figure 37. Optimal Input, $\delta_{a_{AX}}$

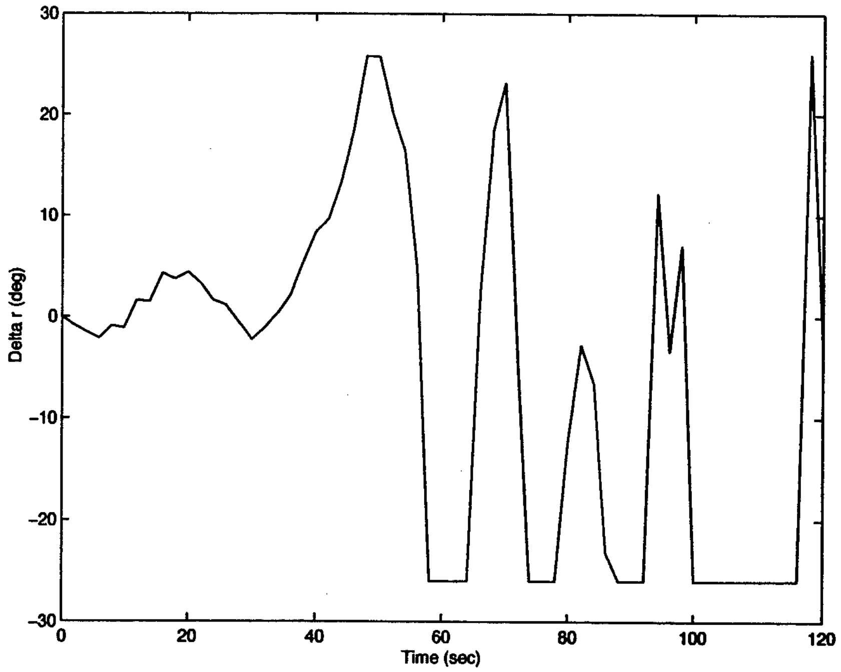

Figure 38. Optimal Input, $\delta_{r_{AX}}$

VI. Conclusions and Recommendations

6.1 Conclusions

This thesis proposed that a Model Predictive Controller, used in the inverse sense, might prove advantageous for use in aircraft mishap investigations. As a proof of concept, the FDR data obtained from Flights 427 and N827AX, in which the aircraft crashed for unknown reasons, were used as the setpoints for the MPC and the control surface inputs that may have caused the known outputs were found. Those inputs resulted in flight characteristics showing good agreement with the FDR data. Average deviations from the setpoints, across the entire output trajectory, ranged from 0.06 to 0.5 radians for the flight angles. Conclusions then are two-fold: first, MPC appears to be promising for future analysis of aircraft mishap investigations, and second, analysis of these flights using this method supports previous findings for the causes of the incidents.

Additionally, when the results from the MPC using non-rate and rate terms included in the performance index are compared, no advantages in the results can be seen for the inclusion of a rate term as developed for this application. However, other applications may warrant the use of a rate term in the performance index.

6.2 Recommendations

Two logical steps for exploration into this area of using MPC in the inverse sense become apparent. First, it might prove beneficial to replace the model used to find the actual outputs with a non-linear model. Some deficencies were seen when attempting to match the setpoints to the FDR data which might be eliminated with the aid of a non-linear model. Second, and possibly along those same lines, implementing this algorithm with a high-performance aircraft, as opposed to a transport aircraft, would be advantageous. This type of system is much more responsive, more dependent upon control usage, and would therefore provide an additional test for MPC in the inverse sense.

In the arena of including a rate term in the performance index, for inverse control or forward control schemes, additional studies need to be performed. In this application, its inclusion did not provide a dramatic improvement in results. However, before a definitive statement pro or con on the rate term can be made, it needs to be used with other applications. In the case of a more responsive aircraft, it could provide improved setpoint tracking, above and beyond what was seen in this thesis.

Appendix A. MATLAB Functions

Two MATLAB functions were developed for use in this thesis. The first is a *setup* file which calculates all time-constant matrices required and prepares the setpoint vector. This file is slightly different for the rate-inclusion performance index since the matrices are different than those required by the non-rate performance index. Both are listed below:

```
function [G,H,F,MC,MQ,MS,P,Cona,Conb,p]=setupcon(A,B,C,L,Q,R,p,q,yd)
% Setup.m to be run prior to running the Simulink program
% to perform the inverse problem. Calculates the matrices
% required for the non-rate performance index.  Of the form:
%
%   [     B                     0           0    ]
%   [(A-L*C)*B                  B           0    ]
% G=[(A-L*C)^2*B          (A-L*C)*B         0    ] (pn x qxi)
%   [     :                     :           0    ]
%   [(A-L*C)^(p-1)*B    (A-L*C)^(p-2)*B   ... B]
%
% H=[L (A-L*C)*L (A-L*C)^2*L....(A-L*C)^(p-1)*L]'   (pn x xi)
%
% F=[(A-L*C)  (A-L*C)^2  ...  (A-L*C)^p]'   (pn x n)
%
% MC=I*C    (peta x pn)
%
% MQ=I*Q    (peta x peta)   (Assumes same weights across time, but)
%                           (q can have varying weights on certain inputs)
%
% MR=I*R    (qxi x qxi)   (Assumes same weights across time, but)
%                         (r can have varying weights on certain inputs)
%
% Constraints: Cona(u) <= Conb
%   Cona=I*Con (p x qxi) matrix of constraint multipliers
%   Conb=I*Max (p x 1) matrix of constraint maximums
%
%         where p = prediction horizon, q = control horizon,
%               n = # states, xi = # inputs, eta = # outputs,
%               Q = output weighting, R = input weighting.
%

n=size(A);n=n(1);
xi=size(B);xi=xi(2);
```

```
eta=size(C);eta=eta(1);

for i=1:size(yd,1)
   ydt((i-1)*eta+1:i*eta,1)=yd(i,:)';
end
yd=ydt;

ALC=A-L*C;

for i=1:p
   F((i-1)*n+1:i*n,:)=[ALC^i];
   H((i-1)*n+1:i*n,:)=[ALC^(i-1)*L];
   MC((i-1)*eta+1:i*eta,(i-1)*n+1:i*n)=C;
   MQ((i-1)*eta+1:i*eta,(i-1)*eta+1:i*eta)=Q;
   for j=1:q
      if i>=j
         G((i-1)*n+1:i*n,(j-1)*xi+1:j*xi)=[ALC^(i-j)*B];
      end
      MR((j-1)*xi+1:j*xi,(j-1)*xi+1:j*xi)=R;
   end
end

P=2*(G'*MC'*MQ*MC*G+MR);

for i=1:q
   for j=1:q
      if i>=j
         a((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=eye(xi,xi);
      end
   end
end
Cona=[eye(q*xi,q*xi);-eye(q*xi,q*xi)]*a;

save datafile P F H MC MQ G Cona p q A B C yd

function [G,H,F,MC,MQ,MS,P,Cona,Conb,p]=setupconrate(A,B,C,L,Q,R,E,p,q,yd)
% Setup.m to be run prior to running the Simulink program
% to perform the inverse problem. Calculates the matrices
% required for the rate performance index.  Of the form:
%
%   [      B                    0           0     ]
%   [(A-L*C)*B                  B           0     ]
% G=[(A-L*C)^2*B          (A-L*C)*B        0     ] (pn x qxi)
%   [       :                     :         0     }
%   [(A-L*C)^(p-1)*B    (A-L*C)^(p-2)*B   ... B]
%
% H=[L (A-L*C)*L (A-L*C)^2*L....(A-L*C)^(p-1)*L]'   (pn x xi)
%
% F=[(A-L*C)  (A-L*C)^2  ...  (A-L*C)^p]'   (pn x n)
%
% MC=I*C   (peta x pn)
%
% MQ=I*Q   (peta x peta)   (Assumes same weights across time, but)
%                          (q can have varying weights on certain inputs)
%
% ME=I*E   (peta x peta)
%
% MR=I*R   (qxi x qxi)   (Assumes same weights across time, but)
%                        (r can have varying weights on certain inputs)
%
% Constraints: Cona(u) <= Conb
%   Cona=I*Con (p x qxi) matrix of constraint multipliers
%   Conb=I*Max (p x 1) matrix of constraint maximums
%
%        where p = prediction horizon, q = control horizon,
%              n = # states, xi = # inputs, eta = # outputs,
%              Q = output weighting, R = input weighting.
%

n=size(A);n=n(1);
xi=size(B);xi=xi(2);
eta=size(C);eta=eta(1);

for i=1:size(yd,1)
   ydt((i-1)*eta+1:i*eta,1)=yd(i,:)';
end
yd=ydt;

ALC=A-L*C;

for i=1:p
   F((i-1)*n+1:i*n,:)=[ALC^i];
   H((i-1)*n+1:i*n,:)=[ALC^(i-1)*L];
   MC((i-1)*eta+1:i*eta,(i-1)*n+1:i*n)=C;
   MQ((i-1)*eta+1:i*eta,(i-1)*eta+1:i*eta)=Q;
   ME((i-1)*eta+1:i*eta,(i-1)*eta+1:i*eta)=E;
   for j=1:q
      if i>=j
         G((i-1)*n+1:i*n,(j-1)*xi+1:j*xi)=[ALC^(i-j)*B];
      end
      MR((j-1)*xi+1:j*xi,(j-1)*xi+1:j*xi)=R;
   end
end

Gtil=[zeros(n,q*xi);G(1:(p-1)*n,:)];
Ghat=G-Gtil;

Ftil=[zeros(n,n);F(1:(p-1)*n,:)];
Fhat=F-Ftil;

Htil=[zeros(n,xi);H(1:(p-1)*n,:)];
Hhat=H-Htil;

P=2*(Ghat'*MC'*MQ*(MC*Ghat+2*ME*MC*G)+G'*MC'*ME'*MQ*ME*MC*G+MR);

for i=1:q
   for j=1:q
      if i>=j
         a((i-1)*xi+1:i*xi,(j-1)*xi+1:j*xi)=eye(xi,xi);
      end
   end
end
Cona=[eye(q*xi,q*xi);-eye(q*xi,q*xi)]*a;

save datafile P F Fhat H Hhat MC MQ G Ghat ME Cona p q A B C yd
```

Within the SIMULINK inverse control program is a MATLAB function which performs the inverse control calculation. Again, it is slightly different for each index, due to the differing matrices. These *inverse problem* functions are listed below:

```
function OUT=ip737con(IN)
% Performs the inverse problem (non-rate) and outputs the current
% J value and the next input to perform, driving the
% actual output to the desired output.
%
% MS=[yd(k)'  yd(k+1)'  yd(k+2)' ...  yd(k+p)']
```

```
%

% Load P F H MC MQ G Cona p q A B C yd from 'datafile.m'.
load datafile

n=size(A);n=n(1);
xi=size(B);xi=xi(2);
eta=size(C);eta=eta(1);

% Extract u(k-1),y,x,t from input vector {u(k-1),y,x;t}:
Uk_1=IN(1:xi,1);
y=IN(1+xi:xi+eta,1);
x=IN(1+xi+eta:xi+eta+n,1);
t=IN(xi+eta+n+1,1);

% Calculate where we are along the desired output vector:
% Round to ensure integer value, add one timestep since
%  t=0 is row 1, x by eta to step along yd vector for pmax
pmin=round(t/2)*eta+1;
pmax=pmin+p*eta-1;

% Store the current yd for (Yd-y) calculation:
curyd=yd(pmin:pmin+eta-1);

% Calculate the MS matrix for f calculation
MS=yd(pmin:pmax)';

f=2*((F*x+H*y)'*MC'-MS)*MQ*MC*G;f=f';

Umax=[19.33*pi/180 69.5 20*pi/180 26*pi/180]';
Umin=-[21.83*pi/180 30.5 20*pi/180 26*pi/180]';
for i=2:q
   Umax((i-1)*xi+1:i*xi,1)=[inf inf inf inf]';
   Umin((i-1)*xi+1:i*xi,1)=-[inf inf inf inf]';
   Uk_1((i-1)*xi+1:i*xi,1)=Uk_1(1:4);
end
Conb=([Umax;-Umin]-[eye(q*xi,q*xi);-eye(q*xi,q*xi)]*Uk_1);

du=qp(P,f,Cona,Conb);

J=.5*du'*P*du+f'*du;
du=du(1:xi);
```

```
% Establish the output vector: {curyd;du;J}:
OUT=[curyd;du;J];

function OUT=ip737conr(IN)
% Performs the inverse problem (rate) and outputs the current
% J value and the next input to perform, driving the
% actual output to the desired output.
%
% Y=[yd(k)'  yd(k+1)'  yd(k+2)' ...  yd(k+p)']
%

% Load P F Fhat H Hhat MC MQ G Ghat ME Cona Conb p A B C yd
% from 'datafile.m'.
load datafile

n=size(A);n=n(1);
xi=size(B);xi=xi(2);
eta=size(C);eta=eta(1);

% Extract u(k-1),y,x,t from input vector {u(k-1);y;x;t}:
Uk_1=IN(1:xi,1);
y=IN(1+xi:xi+eta,1);
x=IN(1+xi+eta:xi+eta+n,1);
t=IN(xi+eta+n+1,1);

% Calculate where we are along the desired output vector:
% Round to ensure integer value, add one timestep since
%  t=0 is row 1, x by eta to step along yd vector for pmax
pmin=round(t/2)*eta+1;
pmax=pmin+p*eta-1;

% Store the current yd for (Yd-y) calculation:
curyd=yd(pmin:pmin+eta-1);

% Calculate the Y matrices for f calculation
Y=yd(pmin:pmax);
if pmin-eta<0
   Ytil=[zeros(eta,1);yd(pmin:pmax-eta)];
else
   Ytil=yd(pmin-eta:pmax-eta);
end
```

```
Yhat=Ytil-Y;

f=2*((MC*Fhat*x+MC*Hhat*y+Yhat)'*MQ*(MC*G+ME*MC*G)+
(MC*F*x+MC*H*y-Y)'*ME'*MQ*(MC*Ghat+ME*MC*G));f=f';

Umax=[19.33*pi/180 69.5 20*pi/180 26*pi/180]';
Umin=-[21.83*pi/180 30.5 20*pi/180 26*pi/180]';
for i=2:q
   Umax((i-1)*xi+1:i*xi,1)=[inf inf inf inf]';
   Umin((i-1)*xi+1:i*xi,1)=-[inf inf inf inf]';
   Uk_1((i-1)*xi+1:i*xi,1)=Uk_1(1:4);
end
Conb=([Umax;-Umin]-[eye(q*xi,q*xi);-eye(q*xi,q*xi)]*Uk_1);

du=qp(P,f,Cona,Conb);

J=.5*du'*P*du+f'*du;
du=du(1:1+xi-1);

% Establish the output vector: {curyd;du;J}:
OUT=[curyd;du;J];
```

Appendix B. Seven-Output Program

A second SIMULINK diagram was utilized in the solution algorithm in order to determine the full seven outputs that were included in the FDR data. This program is found in Figure 39 below.

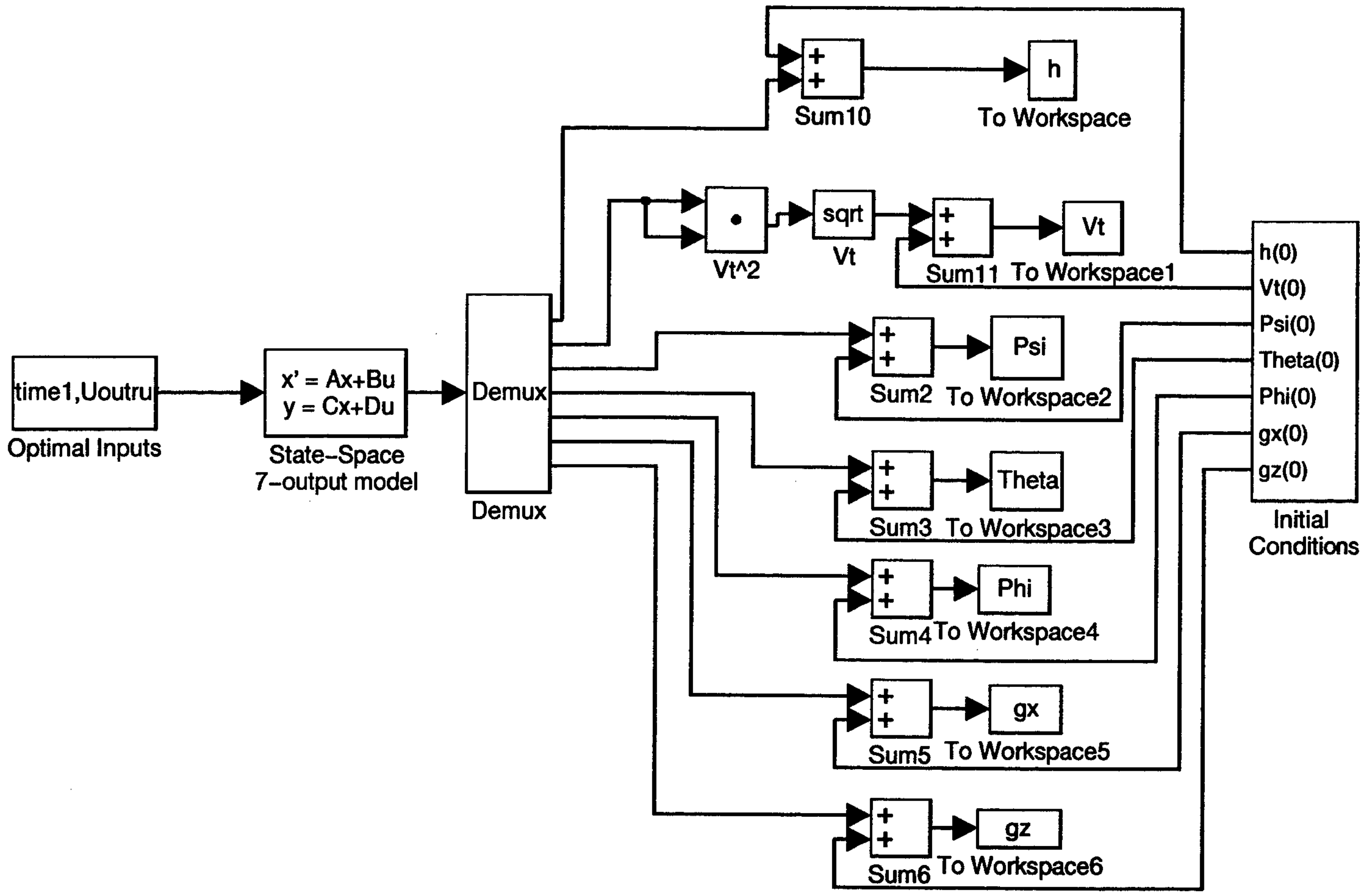

Figure 39. Seven-Output SIMULINK Program

Appendix C. Additional Transport Model Information

The following tables show the stability derivative values used to develop the transport aircraft model (13).

Derivative	Value (s^{-1})	Derivative	Value (s^{-2})
X_u	-.0292	M_{δ_e}	-.4430
Z_u	-.2260	L_β	-3.190
X_w	.1400	N_β	.4990
Z_w	-.6740	L_{δ_a}	3.840
Y_r	0	N_{δ_a}	.4010
L_r	.9800	L_{δ_r}	.3350
N_r	-.2150	N_{δ_r}	-.3270
L_p	-1.390	Y_{δ_r}	.0250 ft
N_p	-.1130	Y_p	0
M_q	-.4810	Y_β	-31.50 ft

Derivative	Value
M_w	-.0016 ft/s
M_u	$.894e^{-5}$ ft/s
$M_{\dot{w}}$	-.0007 1/ft
X_{δ_e}	.4500 ft/s^2
Z_{δ_e}	-4.950 ft/s^2
X_{δ_T}	.0003
Z_{δ_T}	$-.134e^{-4}$
M_{δ_T}	$.8160e^{-6}$

Additional values required for calculation of the A and B matrices are:

$$u_o = 316.26\ ft/s$$

$$\theta_o = .0723\ rad$$

$$\ell = 0\ ft$$

$$g = 32.2\ ft/s^2$$

where u_o and θ_o are taken from the FDR data, g is gravity, and ℓ is the distance from the aircraft center of gravity to the location of the measurement accelerometer (estimated here to be zero or approximately zero). The resulting ten-state A and B matrices are:

$$A = \begin{bmatrix} 0 & 0 & -1.0000 & 0 & 316.2612 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.0292 & 0.1400 & 0 & -32.2000 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2260 & -0.6740 & 316.2612 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0002 & -0.0011 & -0.7097 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.0996 & 0 & -1.0000 & 0.1015 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3.1900 & -1.3900 & 0.9800 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4990 & -0.1130 & -0.2150 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \end{bmatrix} \tag{56}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.4500 & 0.0003 & 0 & 0 \\ -4.9500 & 0.0000 & 0 & 0 \\ -0.4394 & 0.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 \\ 0 & 0 & 3.8400 & 0.3350 \\ 0 & 0 & 0.4010 & -0.3270 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{57}$$

where the states are

$$x^T = [\Delta h \; \Delta u \; \Delta w \; \Delta q \; \Delta\theta \; \Delta v \; \Delta p \; \Delta r \; \Delta\phi \; \Delta\psi] \tag{58}$$

as defined in the thesis text.

This model results in poles in the following table, along with two poles at the origin:

Pole(s)	Location
Phugoid	$-.0150 \pm .1171i$
Short Period	$-.6915 \pm .6047i$
Spiral	$-.0142$
Roll	-1.5891
Dutch Roll	$-.0506 \pm .9386i$

Bibliography

1. Abdelrahman, M.M. and A.M. Al-Bahl, "A Generalized Technique for the Inverse Simulation of Aircraft Motion Along Predetermined Trajectories." AIAA 94-3523, 1994.
2. Ebdon, D.W. *Model Predictive Control of Aerospace Systems.* MS thesis, Air Force Institute of Technology, 1996.
3. Hess, R.A., et al. "Generalized Technique for Inverse Simulation Applied to Aircraft Maneuvers," *AIAA Journal of Guidance, Control, and Dynamics* (1990).
4. Kerrigan, James W., "Personal communication." Boeing.
5. Lee, S. and Y. Kim. "Solution of the Inverse Simulation Problem by Optimization Technique and It's Application to Aircraft Nonlinear Large Angle Maneuvers." *AIAA Guidance Navigation and Control Conference.* Number AIAA 96-3701. 1996.
6. Maciejowski, J.M. and S.A. Heise, "Heuristic Robustness Analysis of Model-Based Predictive Controllers." Cambridge University Engineering Department.
7. Mathworks. *MATLAB User's Guide.* Prentice-Hall, Inc., 1997.
8. Mathworks. *SIMULINK User's Guide.* Prentice-Hall, Inc., 1998.
9. Morari, Manfred. *Model Predictive Control Draft Notes.*
10. Napolitano, Marcello. "Virtual Flight Data Recorder: A neural extension of existing capabilities," *AIAA Journal of Guidance, Control, and Dynamics*, (AIAA 97-3538) (1998).
11. Nelson, Robert C. *Flight Stability and Automatic Control.* Boston: The McGraw Hill Companies, 1998.
12. Parks, Edwin K., et al. "Reconstruction of the 1994 Pittsburgh Airplane Accident Using a Computer Simulation," *AIAA Journal of Guidance, Control, and Dynamics*, (AIAA 98-0503) (1998).
13. R.K., Heffley and W.F. Jewell. *Aircraft Handling Qualities Data CR-2144.* Technical Report, NASA, 1972.
14. Shearer, C.M. and S.A. Heise. "Constrained Model Predictive Control of a Nonlinear Aerospace System," *AIAA Journal of Guidance, Control, and Dynamics*, (AIAA 98-4235) (1998).

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE SOLVING AN INVERSE CONTROL PROBLEM USING PREDICTIVE METHODS	5. FUNDING NUMBERS
6. AUTHOR(S) Chad J. Davis, Lieutenant, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P Street WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GAE/ENY/99M-05
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Siva Banda AFRL/VAAD 2210 8th St. WPAFB OH 45433-7521	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Advisor: Sharon A. Heise, Major, USAF
DSN: 898-1583
e-mail: saheise@mindspring.com

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Model Predictive Control is the class of control methods that optimizes a specified performance index in order to minimize the weighted future output deviations from a setpoint trajectory. This thesis applies MPC in the inverse sense -- known aircraft outputs are applied in the performance index as setpoints in an attempt to determine what control histories caused those outputs. Using this method, aircraft mishap investigators could then have a means of determining what the control surface deflections were throughout an incident since Flight Data Recorder data does not include control surface deflections. The actual Flight Data Recorder data from aircraft mishaps are utilized as proof of concept.

14. SUBJECT TERMS Control theory, Jet transport aircraft, Flight simulation, Mathematical models, Model Predictive Control, Inverse simulation	15. NUMBER OF PAGES 74
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL