A Learning Enhanced Flight Control System for High Performance Aircraft

by

Noel F. Nistler

B.S.A.E., United States Air Force Academy

(1990)

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

MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1992

Signature of Author _____
Department of Aeronautics and Astronautics
June, 1992

Approved by _____
Walter L. Baker
Technical Supervisor, CSDL

Certified by _____
Dr. Milton B. Adams
Lecturer, Thesis Supervisor

Accepted by _____
Professor Harold Y. Wachman
Chairman, Departmental Graduate Committee

A Learning Enhanced Flight Control System for High Performance Aircraft

by

Noel F. Nistler

Submitted to the Department of Aeronautics and Astronautics
on May 8, 1992 in partial fulfillment of the requirements for the
Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

Numerous approaches to flight control system design have been proposed in an attempt to govern the complex behavior of high performance aircraft. Gain scheduled linear control and adaptive control have traditionally been the most widely used methodologies, but they are not without their limitations. Gain scheduling requires large amounts of *a priori* design information and costly manual tuning in conjunction with flight tests, while still lacking an ability to accommodate unmodeled dynamics and model uncertainty beyond a limited amount of robustness that can be incorporated into the design. Adaptive control is suitable for nonlinear systems with unmodeled dynamics, but has deficiencies in accounting for quasi-static state dependencies. Moreover, inherent time delays in adaptive control make it difficult to match the performance of a well-designed gain scheduled controller. An alternative approach that is able to compensate for the inadequacies experienced with traditional control techniques and to automate the tuning process is desired.

Recent learning techniques have demonstrated an ability to synthesize multivariable mappings and are thus able to learn a functional approximation of the initially unknown state dependent dynamic behavior of the vehicle. By combining a learning component with an adaptive controller, a new hybrid control system that is able to adapt to unmodeled dynamics and novel situations, as well as to learn to anticipate quasi-static state dependencies is formed.

This thesis explores the concept of augmenting an adaptive flight controller with a learning system. The goal is to examine the extent to which learning can be used to improve the performance of an adaptive flight control system architecture, as well as to highlight some of the difficulties introduced by learning augmentation. Performance of the control system is defined in terms of its ability to control a nonlinear, three-degree-of-freedom aircraft model reacting to altitude and velocity commands. This hybrid approach offers potential advantages over conventional techniques in terms of performance, model uncertainty accommodation, and tuning costs.

Thesis Supervisor: Dr. Milton B. Adams
Title: Lecturer, Department of Aeronautics and Astronautics

Technical Supervisor: Mr. Walter L. Baker
Title: Senior Member of Technical Staff, Draper Laboratory

ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife Kathy for all the support and encouragement she has given me over the last two years. Thank you to Walt Baker for the guidance and technical assistance in preparing this thesis. Walt has been a great supervisor and friend. I would also like to thank Pete Millington for all the technical help and providing that special type of humor that only Pete can deliver. Thanks also to Jeff Alexander and Leemon Baird for their support with Netsim and C.

A special thank you goes out to Milt Adams for all the time and effort required to be my thesis advisor.

Thank you: Steve A., for providing daily competition in Nerf hoops, Steve S., for interesting political discussions, and Whit, wherever you may be. Also Jay, Dino, Bill, Torsten, Mitch, and Dean. Thank you to everyone else who has made my stay at Draper enjoyable.

Finally, I would like to thank the Charles Stark Draper Laboratory for providing this educational opportunity.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc. with support provided by the U.S. Air Force Wright Laboratory under Contract F33615-88-C-1740. Publication of this report does not constitute approval by Draper Laboratory or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

TABLE OF CONTENTS

1	INTRODUCTION		1
	1.1	Problem Description	2
	1.2	Thesis Objectives	3
	1.3	Overview	3
2	BACKGROUND		5
	2.1	High Performance Aircraft Characteristics	5
	2.2	Traditional Control Techniques	7
		2.2.1 Robust Control	8
		2.2.2 Gain Scheduling	8
		2.2.3 Adaptive Control	9
	2.3	Connectionist Learning Systems	10
		2.3.1 Foundations of Connectionist Systems	11
		2.3.2 Early Connectionist Networks	13
		2.3.3 The Backpropagation Network	15
		2.3.4 Connectionist Learning Systems for Control	19
3	TECHNICAL APPROACH		22
	3.1	Adaptive Control Component	22
		3.1.1 Control Law Derivation	24
		3.1.2 Implementation Issues	28
	3.2	Learning Control System	29
		3.2.1 Incremental Learning and Fixation	29
		3.2.2 Spatially Localized Learning	31
		3.2.3 The Linear-Gaussian Network	32
	3.3	Hybrid Learning / Adaptive Control	38
		3.3.1 Hybrid Control System Architecture	39

3.3.2 Learning Versus Adaptation 40
3.3.3 Control Law Development 42

4 EXPERIMENTS 47

4.1 Aeroelastic Oscillator 48
4.1.1 Description 48
4.1.2 Open Loop Dynamics 50
4.1.3 Reference Model 50
4.1.4 Application of Hybrid Controller 51
4.1.5 Aeroelastic Oscillator Experiment 1 55
4.1.6 Aeroelastic Oscillator Experiment 2 62
4.2 High Performance Aircraft Model 66
4.2.1 Aircraft Description 66
4.2.2 General Aircraft Characteristics 71
4.3.3 Aircraft Reference Model 75
4.2.4 Application Issues 76
4.2.5 High Performance Aircraft Experiment 1 78
4.2.6 High Performance Aircraft Experiment 2 90

5 CONCLUSIONS AND RECOMMENDATIONS 96

5.1 Summary and Conclusions 96
5.2 Recommendations for Future Work 97

BIBLIOGRAPHY 99

1 INTRODUCTION

Numerous approaches to flight control system design have been proposed in an attempt to govern the behavior of high performance aircraft. This class of aircraft presents formidable challenges to the designer since by nature their dynamics are nonlinear, multivariable, and coupled (Etkin (1982)). Moreover, high performance aircraft tend to exhibit modes with relatively high natural frequencies and minimal damping as compared to typical aircraft. Gain scheduled linear control and adaptive control appear to be the most popular methodologies for flight control law design, but they are not without their limitations. Gain scheduling techniques combine multiple linear control laws to formulate a nonlinear controller (Lewis & Stevens (1992)). This process requires large amounts of *a priori* model information and potentially costly manual tuning, since a separate linear controller must be designed for each of a selected set of distinct regions of the operating envelope. In addition to this tedious design approach, gain scheduled controllers lack the ability to accommodate unmodeled dynamics and model uncertainty beyond a limited amount of robustness that can be incorporated into the design. Adaptive control is suitable for nonlinear systems with unmodeled dynamics but has deficiencies in effectively accounting for quasi-static state dependencies. Moreover, inherent time delays of adaptive control make it difficult to match the performance of an ideal gain scheduled controller (Stein (1980)). This thesis presents an alternative approach that compensates for some of the inadequacies experienced with these traditional control techniques.

By combining an adaptive component with a learning system, an innovative new hybrid controller is formed that allows each mechanism to focus on the control objective for which it is best suited. The primary role of the adaptive control component in the hybrid system is to accommodate unmodeled dynamics (i.e., dynamical behavior that is not

expected, based on the design model). Additionally, the adaptive component has the auxiliary task of providing estimates of any observed unmodeled state dependent dynamic behavior to the learning system (i.e., unknown dynamics that are a function of state in areas of the state space where learning has not occurred). These estimates are obtained by observing previous plant behavior, essentially providing delayed estimates. Moreover, since no use is made of past estimates, the adaptive component can be considered memoryless. Based on the estimates from the adaptive component, a learning system can be used to learn a functional approximation of these state dependencies and ultimately reduce model uncertainty in the system. Connectionist networks (which include artificial neural networks) have demonstrated the ability to synthesize highly nonlinear, multivariable mappings (Funahashi (1988), Hornik, *et al.* (1989)) More specifically, spatially localized connectionist networks have been proposed as an appropriate learning system for control applications (Baker & Farrell (1992)). Armed with a mapping from the learning system that represents the previously unknown state dependencies, the hybrid controller can anticipate vehicle behavior that is a function of state and compensate accordingly, effectively removing the delay in the estimates provided by an adaptive controller. The impact of a controller that has the ability to anticipate vehicle behavior can be seen in improved closed-loop system performance. Moreover, this ability to learn state dependencies offers advantages over conventional techniques in terms of model uncertainty accommodation and automation of the tuning process.

1.1 PROBLEM DESCRIPTION

This thesis presents the development and application of a hybrid control system to the problem of flight control for a high performance aircraft. Time Delay Control (TDC), a model reference adaptive controller, is augmented by a linear-Gaussian connectionist network, to form the hybrid flight control system. This hybrid system is applied to the

control of the longitudinal motion of a high performance aircraft during various altitude and velocity maneuvers. Due to nonlinearities, model uncertainty, unknown dynamics, and a host of other difficulties, high performance aircraft present a significant challenge to the development of flight control systems.

1.2 THESIS OBJECTIVES

This thesis explores the use of a learning system to augment an adaptive flight controller. The extent to which learning can be used to improve an adaptive flight control system architecture, as well as the difficulties introduced by learning augmentation, are examined. The primary objective of this thesis is to illustrate the advantages of a hybrid adaptive / learning control system in terms of its ability to accommodate unmodeled dynamics and reduce state dependent uncertainties in the system model. This hybrid approach offers advantages over conventional techniques in terms of performance, robustness, and design refinement costs.

1.3 OVERVIEW

In Chapter 2, the challenges associated with high performance aircraft control law design are outlined. Moreover, background information on traditional control techniques is provided to serve as a foundation for the hybrid control law development, and also as a basis for comparison of alternative designs. The theoretical concepts underlying connectionist learning systems, as well as some approaches in using learning systems for control, are also presented.

In Chapter 3, the technical aspects of the hybrid control law are developed. This is accomplished by first presenting the underlying theory of the adaptive component and the spatially localized learning system before moving on to the derivation of the hybrid system.

General characteristics of the hybrid controller are also presented.

In Chapter 4, two experiments are presented to illustrate the implementation and performance of the hybrid control law. The first experiment uses the hybrid system to control a relatively simple nonlinear aeroelastic oscillator. Due to the low dimensionality of the plant, and a known truth model, the analysis and evaluation of the hybrid control system for the aeroelastic oscillator is greatly simplified. In the second experiment, the hybrid system is applied to a realistic high performance aircraft model. Descriptions of the major components of the aircraft model as well as its significant characteristics are also provided. An evaluation of aircraft performance when controlled by the hybrid system is presented and compared with other designs for various simulations. Learning system characteristics are also described.

Chapter 5 summarizes the major contributions of this thesis. In addition, recommendations for future research are presented.

A bibliography of the works used in preparing this thesis follows Chapter 5.

2 BACKGROUND

The design of automatic flight control systems for high performance aircraft presents significant challenges for the control engineer. Although well-developed design methodologies exist for linear systems, similar methodologies and related theories for nonlinear systems have proven to be elusive. In this chapter, the formidable challenges inherent in high performance aircraft control system design are presented in Section 2.1, conventional control approaches for accommodating these difficulties are presented in Section 2.2, while the fundamentals of connectionist learning systems and some approaches for learning control are introduced in Section 2.3.

2.1 HIGH PERFORMANCE AIRCRAFT CHARACTERISTICS

Because the aerodynamic forces and moments that act on an aircraft are complicated, nonlinear functions of many variables, aircraft exhibit complex flight dynamics. This section discusses the major difficulties associated with high performance aircraft flight control design.

Due to the high cost and dangers involved in flight testing, the majority of the effort in flight control system design and development relies on a model of the aircraft instead of the actual vehicle. This approach guarantees the presence of *model uncertainty* since it is impossible to capture the complete dynamical behavior of complex aircraft in a model. Errors in the model can be attributed to two major factors: structural and parametric uncertainty (Baker & Farrell (1991)). Typically, the mathematical structure of an aircraft model is derived from the general equations-of-motion for a single, rigid body. These are the classical Euler equations. From this base set of equations, the designer determines

additional effects that must be included to obtain an effective flight control system design. Gyroscopic effects due to the presence of spinning rotors and aeroelastic effects due to inaccuracies in the rigid body assumption have historically been incorporated into the equations-of-motion. Beyond the difficulties associated with the selection and development of the proper model structure, the accuracy of the actual parameter values used in the model plays a large role in the quality of an aircraft model. Since values of the parameters are typically obtained from wind-tunnel testing or computational fluid dynamics (e.g., computer simulations of airflow over an aircraft model), large discrepancies are possible. Additional model uncertainty develops from the fact that not all flight conditions can be easily modeled by a single global model structure. For this reason, separate models are needed for post-stall flight, vertical take-off modes, and other extreme flight conditions. In general, all models contain a degree of uncertainty that must be addressed by the flight control system.

Nonlinearities present a major difficulty to the control engineer since no general theory for control design synthesis has been developed for nonlinear systems. Aircraft dynamical behavior is inherently nonlinear; this nonlinear behavior is caused primarily by the fact that the aerodynamics forces and moments that dictate aircraft motion are themselves complicated, nonlinear functions of many variables. Moreover, the full six-degree-of-freedom rigid body equations-of-motion include nonlinear terms. The effects of actuator rate limiting, control position limits, and other control linkages are further examples of nonlinearities.

Another complication experienced with flight control law design is that high performance aircraft are inherently *high dimensional, multivariable* systems. A six-degree-of-freedom aircraft requires twelve coupled state equations to fully characterize its rigid body dynamics. Moreover, multiple control effectors (e.g., stabilator, rudder, ailerons, and throttle) are employed to achieve the primary objective of simultaneously controlling a number of outputs (e.g., altitude, heading, and velocity). As a result, any

control system that attempts to decouple the dynamics and connect independently designed single-input / single-output controllers will generally sacrifice performance for ease of design.

The "high performance" qualifier on the aircraft model implies expanded flight regimes that also tend to exacerbate control difficulties. These regimes include high angle-of-attack, high Mach, and other regions of the aircraft envelope where large changes in the aircraft dynamics can be expected. For example, a dynamic mode that is stable and adequately damped in one region of the envelope may become lightly damped or unstable in another. This fact, combined with the general trend toward relaxed static stability, requires rapid control action to stabilize the aircraft.

The above discussion illustrates the major challenges in flight control law design. Additional difficulties confront the control engineer due to the design methods themselves (e.g., frequency domain methods do not easily lend themselves to multivariable control) and due to challenges in applying the control approach to the real vehicle (e.g., digital implementation issues).

2.2 TRADITIONAL CONTROL TECHNIQUES

Automatic flight control systems have evolved from the "Sperry Aeroplane Stabilizer," the first functional autopilot, to advanced multivariable digital systems capable of generating a large number of control actions per second (Lewis & Stevens (1992)). Of the multitude of design theories and methodologies developed for flight control law design, the majority can be classified into the two broad categories: fixed control (e.g., robust control and gain-scheduled control) and adaptive control. The following sections introduce these traditional control approaches. Each technique is critiqued in its ability to accommodate the design difficulties presented in the previous section.

2.2.1 Robust Control

Robust control has gained popularity for flight control due to its ability to accommodate a certain degree of uncertainty associated with the aircraft model. By explicitly incorporating uncertainty into the design process, robust controllers provide performance and stability guarantees. However, this resilience to uncertainty, or robustness, is usually obtained at the expense of a loss in system performance. Since typical robust control techniques (e.g., classical Bode gain / phase margin methods or H_∞ design) rely on a worst case estimate of the modeling error or margins to determine a fixed parameter control system, the resulting control law is often conservative when applied to the nominal plant and presents a tradeoff between stability robustness and high performance. Thus, a control system designed to account for modeling uncertainty results in suboptimal performance relative to the ideal case where no model uncertainty exists. To increase performance, the designer can exploit an improved model having less uncertainty. However, the added complexity and cost of a more refined model often prohibits this course of action. Beyond difficulties in achieving maximum performance, robust controllers are ill-adapted to handle highly nonlinear systems or unmodeled dynamics. In particular, although slight perturbations due to nonlinearities or unknown dynamics can be accommodated by further increasing the bounds on uncertainty, difficulties in achieving adequate performance are further exacerbated. For highly nonlinear aircraft with substantial unmodeled dynamics or model uncertainty, robust control is impractical from a performance point of view.

2.2.2 Gain Scheduling

Flight control systems for modern high performance aircraft are generally developed with a gain scheduling design methodology. Gain scheduling methods combine multiple linear control laws to formulate a nonlinear controller. This control approach can accommodate many of the difficulties associated with complex nonlinear systems, such as

high performance aircraft. To formulate this nonlinear control law, the operating envelope is separated into an *ad hoc* set of distinct regions where the dynamical behavior is approximately linear. By linearizing the dynamics in each distinct region, the designer is able to utilize the large class of linear control theories (e.g., robust or optimal approaches) to develop a control law best suited to realize local performance objectives. The combined nonlinear control law is achieved by transitioning among these linear control laws as flight conditions move among the prescribed linearized regions. Transitioning is accomplished by interpolating the control parameters (e.g., feedback gains) as a function of scheduling variables or operating condition. Mach number, angle-of-attack, and dynamic pressure are the most commonly used scheduling variables. As a result, highly nonlinear systems require numerous linearized regions, and subsequently a multitude of linear control laws, to approximate nonlinear behavior.

In addition to the subjective (and tedious) nature of defining a set of linearized operating regimes and designing a linear control law for each linearization point, gain scheduled flight control systems are also susceptible to model uncertainty and unmodeled dynamics. Differences between the observed and predicted vehicle behavior can only be corrected by on-line manual tuning during flight testing.

2.2.3 Adaptive Control

Adaptive control has been suggested as a viable method for aircraft flight control (Lewis & Stevens (1992), Stein (1980)). Adaptive techniques generally rely on differences between desired and observed vehicle behavior to adjust (adapt) variable internal parameters to ultimately achieve acceptable closed-loop performance. Using this approach, adaptive controllers have shown an ability to accommodate nonlinear plants with unmodeled dynamics. However, adaptive controllers encounter difficulties in systems with rapidly varying parameters and extensive nonlinearity. In an adaptive technique, the controller must wait until undesired plant behavior is observed before it can determine how

to adjust its parameters. Potentially, several control intervals might be required to accurately detect and compensate for variations in these parameters. Beyond this delay associated with determining the correct parameters, sensor noise causes additional delay due to the required filtering. For vehicles that regularly experience large parameter variations, the resulting control law may spend large portions of time in some suboptimal, partially adapted configuration. This dilemma is exacerbated by the reactive nature of adaptive controllers in that the parameters must be re-tuned whenever the vehicle enters a new region, even if the correct values had previously been determined for that region. Hence, adaptive controllers fail to make use of predictable behavior (e.g., state dependencies) that would reduce the time spent in partially adapted states and ultimately improve performance. For these reasons, it is difficult for an adaptive controller to match the performance of a well designed gain scheduled controller.

Although not as common as gain scheduling or adaptive approaches, multi-region adaptive controllers have also been suggested as a means for flight control (Athans, *et al.* (1977), Stein, *et al.* (1977)). Essentially, this approach schedules multiple local plant models within an indirect adaptive control framework.

2.3 CONNECTIONIST LEARNING SYSTEMS

Connectionist learning systems have received much attention in the research community due to its potential for solving problems in pattern recognition, associative memory, and database retrieval (Melsa (1989)). Moreover, recent attention has been given to the ability of connectionist networks to synthesize multivariable, nonlinear mappings and to how this information can be applied to improve automatic control systems. In this section, a brief history regarding the development of a class of connectionist learning systems that is relevant to the control problem described earlier is presented. Some alternative approaches for incorporating connectionist systems into control system designs

are also introduced.

2.3.1 Foundations of Connectionist Systems

Connectionist systems, which include what are often called "artificial neural networks," owe their foundations to biologists and research psychologists who originally studied the ability of neural models to mimic the behavior of the brain (Rosenblatt (1962), Klopf (1988)). Contemporary connectionist systems have advanced significantly from these early beginnings (Barto, *et al.* (1983), Rumelhart, *et al.* (1986)). Many of the recent connectionist learning systems emphasize the mathematical theory of function approximation, estimation, and optimization (Baker & Farrell (1992), Poggio & Girosi (1990)).

Connectionist learning systems typically contain a large number of simple processing units that are combined in a highly interconnected architecture. These processing units, also known as nodes or "artificial neurons," make up the basic building blocks of a connectionist system. Figure 2.1 illustrates the internal structure of a simple, 3-input node

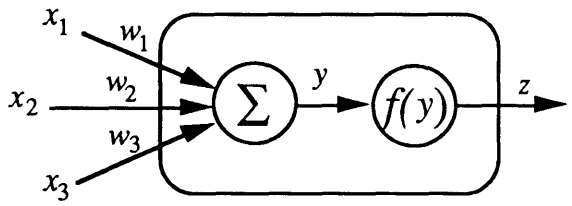

Figure 2.1 3-Input / 1-Output Simple Node

where x_1, x_2, and x_3 are the node inputs, w_1, w_2, and w_3 are weightings for the respective inputs, and y is the sum of the weighted inputs. The output of the network, z, is simply the value of the nodal function f evaluated at y. Nonlinear nodal functions are required to realize nonlinear mappings. Three examples of nodal functions are the threshold,

sigmoidal, and Gaussian functions.

If a large amount of *a priori* information is known about the desired mapping of the network, the weights between the nodes can be set to fixed values to realize the network mapping. However, typical connectionist networks use nodes with fixed functions and adaptable weights that are adjusted using an appropriate learning law. Under supervised learning, the amount of weight adjustment is determined by evaluating an error formed by the difference between the calculated output of the network and a known desired output (Melsa (1989)). This contrasts with the weight adaptation by unsupervised learning, where only inputs and a reinforcement signal that characterizes past performance (i.e., not a known desired output) are utilized in adjusting the weights (Barto (1989), Mendal & McLaren (1970)). Thus, the operation of adaptable connectionist networks consists of two distinct phases: output calculation and learning. The output calculation phase is characterized by the determination of the network output based upon the given inputs, weights, and nodal functions. The purpose of the learning phase is to adjust the weights (using either a supervised or unsupervised technique) to obtain desired input / output behavior.

Connectionist networks are frequently categorized by the nodal architecture and associated output calculation or by the learning technique. One common architecture dependent on a specific output calculation method is the feedforward connectionist network (Funahashi (1988), Hornik, *et al.* (1989)). In feedforward structures, the output for any given node is not connected back as an input to itself by any feedback loop. Because of this feature, present outputs do not impact future output values (present outputs can impact future outputs in the learning phase by adjustment of the weights). Moreover, the output of the entire system can be calculated in a single pass since each layer simply outputs computed values based on inputs from the previous layer. Figure 2.2 illustrates a simple feedforward network.

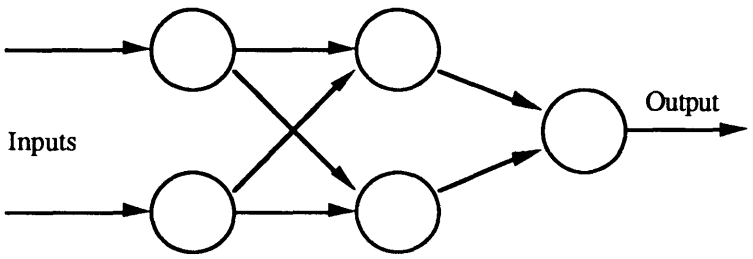

Figure 2.2 Simple 2-Input / 1-Output Feedforward Network

Another major class of connectionist systems consists of feedback (or recurrent) networks. The distinguishing feature of a feedback network is that nodes have the ability to influence themselves through feedback. The feedback can act directly from a given node to itself or indirectly through other nodes. Although feedback networks have an ability to learn dynamical mappings (e.g., mappings that change with time), the learning laws become complicated since the network output is no longer simply a function of network inputs and weights (it is also a function of the state of the network). Moreover, any feedback network representing a dynamical mapping can be expressed as an equivalent dynamic system of two static mappings separated by an integration or unit delay operator (Livstone, *et al.* (1992)).

By altering the nodal function, output calculation, learning approach, or a host of other variables, connectionist networks have been developed that display an array of different properties (Barto (1989), Melsa (1989), Minsky & Papert (1969)). Section 2.3.2 discusses some of the most popular early connectionist systems.

2.3.2 Early Connectionist Networks

One of the earliest uses of a connectionist methodology for learning was the *perceptron* network (Rosenblatt (1962)). A simple perceptron network is comprised of single or multiple layers of perceptron nodes connected in a feedforward configuration. A perceptron node is characterized by the binary threshold function used to formulate the output from the weighted sum of its inputs as shown in Figure 2.3. If the weighted sum is

greater than some prescribed threshold value, the perceptron node outputs an "on" signal or the value 1. For inputs below the threshold, the node is considered "off" and outputs –1.

$$z = f(y) = \begin{cases} 1 & \text{if } y > \text{ threshold} \\ -1 & \text{if } y < \text{ threshold} \end{cases}$$

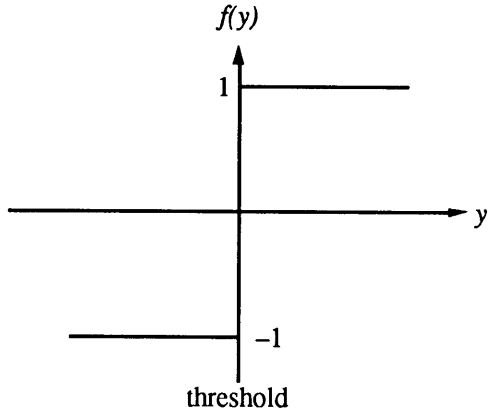

Figure 2.3 Binary Threshold Function

Perceptron networks have illustrated surprisingly powerful mapping capabilities. Minsky and Papert demonstrated the ability of single-layer perceptron networks to learn any discriminant function among classes that are linearly separable, using a simple learning rule (Minsky & Papert (1969)). The learning rule adjusts the weights incrementally depending on their impact on the error between the network output and the prescribed output. It was later shown that multi-layer perceptron networks are capable of discriminating a large class of nonlinearly separable problems. However, no general guarantee on the ability of any learning law to locate an optimal set of weights exists for multi-layer networks as in the single-layer case.

Another pioneering connectionist network is the adaptive linear element, or ADALINE (Widrow & Hoff (1960)). ADALINE networks consist of simple nodes connected in a feedforward architecture. The distinguishing features of an ADALINE network include a nodal function that simply outputs the weighted sum of the inputs (i.e., $f(y) = y$) and a normalized least mean square (LMS) learning law. Under supervised learning where the current inputs and desired output are known, the LMS learning law

attempts to minimize the mean squared value of the error. When the weights are changed in proportion to the error, an ADALINE network is guaranteed to converge to the minimum of the mean squared error for linearly separable problems. In an attempt to extend this result to nonlinearly separable problems, ADALINES can be connected in a hierarchical structure to form a network of multiple adaptive linear elements (MADALINES). Although MADALINES are capable of producing complicated nonlinear mappings, determining the optimal weights between layers of ADALINES is a difficult process. These difficulties are the result of LMS learning laws being limited to the determination of optimal ADALINE weights and not the weights associated with their connecting layers (Melsa (1989)).

2.3.3 The Backpropagation Network

Although the advent of perceptrons, ADALINES, MADALINES, and their variants played a large role in the development of connectionist networks, the latest resurgence of interest in learning systems can be attributed to the backpropagation sigmoidal network. Although backpropagation is strictly speaking a learning law, its extensive use has resulted in the name being generalized to denote the large class of feedforward multi-layer networks that employ this particular learning approach. Similar to the early architectures, backpropagation networks are constructed from the combination of simple nodes arranged in a hierarchical, feedforward fashion. However, instead of the threshold and identity functions associated with the simple perceptron and ADALINE networks respectively, the backpropagation node uses a nonlinear nodal function. One of the most commonly used nodal functions is the sigmoidal function illustrated in Fig 2.4.

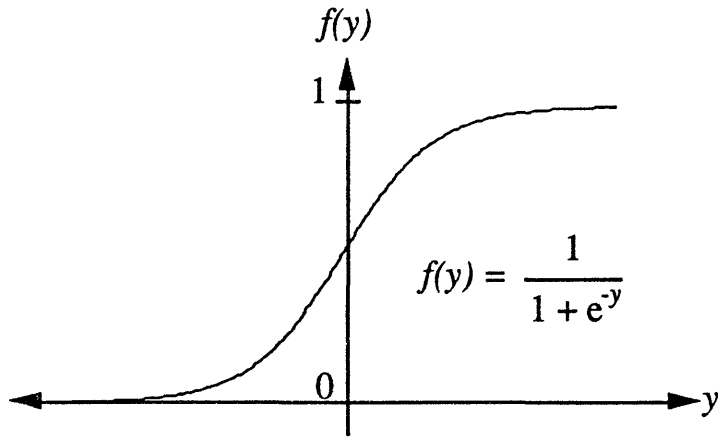

Figure 2.4 Sigmoidal Function

A sigmoidal function is a continuous, monotonically increasing function with finite asymptotic values. As a result, a sigmoid offers advantages over discontinuous nodal functions in that it is continuously differentiable, which plays a large role in the gradient learning algorithm described below.

A typical sigmoidal backpropagation network is shown in Figure 2.5. This network architecture is generally sub-divided into three distinct regions or layers: input layer, hidden layers, and output layer.

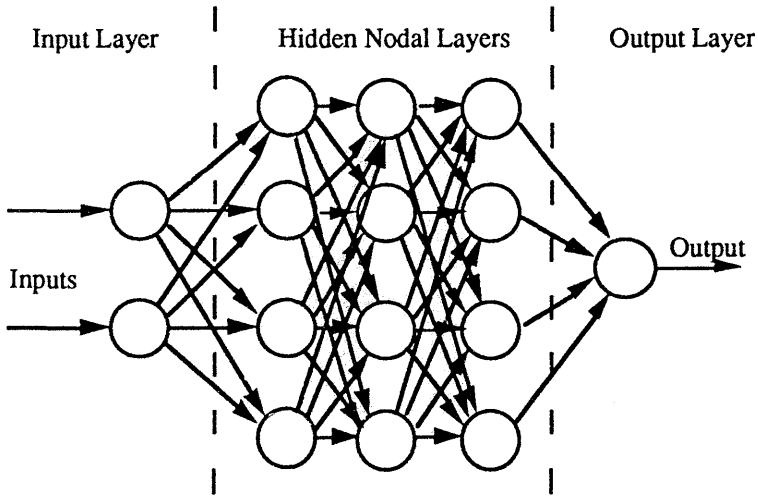

Figure 2.5 Typical 2-Input / 1-Output Feedforward Network with Three Hidden Layers

The first region, the input layer, is characterized by nodes that act as an interface between network inputs and the subsequent hidden layer by simply passing the input value to a set of nodes in the first hidden layer (although weighting is sometimes added to the signal). Moreover, there is the same number of nodes in the input layer as there are inputs, and each input layer neuron typically passes its value to each node in the subsequent layer. The second region contains the hidden nodal layers. In this region, the weighted sum of the outputs from the previous layer is used as the input to each sigmoid function to compute the output of the node. The output is subsequently passed to a following hidden or output layer. The final region is the output layer, which contains the same number of output nodes as there are network outputs. The function of the output layer is to compute the weighted sum of its inputs and pass this value, or the value of a sigmoid function evaluated at this weighted sum, as the network output. Typically, the number of processing nodes in backpropagation networks is large compared to the number of different kinds of nodal functions used in the network, with networks using a single nodal function being the most common.

Although the selection of the network architecture is significant, the performance of connectionist networks is ultimately determined by the ability of the learning law to find the optimal weights. For backpropagation networks, weights are adjusted using an "error backpropagation" algorithm (Rumelhart, *et al.* (1986)). Whereas the learning laws of early connectionist networks had difficulties in properly adjusting connecting layer weights, the error backpropagation algorithm provides a systematic method to adjust weights in all adaptable layers. The basic error backpropagation algorithm uses a supervised gradient descent method to incrementally adjust the weights in the negative direction of the gradient (with respect to the weights) of a cost function. The general form of the gradient rule is shown in Equation (2.1) below:

$$\Delta \mathbf{w} = -\alpha \frac{\partial J}{\partial \mathbf{w}} \tag{2.1}$$

where $\mathbf{w}$ is a vector whose elements are the input weights, α is the learning rate (i.e., the step size), and J is the cost function to be minimized. The most commonly used cost function to be minimized is a quadratic function of the error between the network output and some desired output. In many supervised learning applications, the network is trained on a finite number of (known) input / output sample points. In this case, known as batch-mode training, the quadratic error cost function takes the following form.

$$J = \frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{d}(\mathbf{x}_i) - \mathbf{f}_{net}(\mathbf{x}_i, \mathbf{w})\right]^{\mathrm{T}}\left[\mathbf{d}(\mathbf{x}_i) - \mathbf{f}_{net}(\mathbf{x}_i, \mathbf{w})\right] \tag{2.2}$$

Here, n is the number of training examples, $\mathbf{x}_i$ is the network input for the i^{th} training sample, $\mathbf{d}(\mathbf{x}_i)$ is the desired output at the i^{th} training sample, and $\mathbf{f}_{net}(\mathbf{x}_i, \mathbf{w})$ is the actual output of the network for the given input and weights. Using this technique, the weights are adjusted once per each pass or epoch through *all* the training examples. Recalling that the output of a layer is a function of the output of the previous layers, the partial derivatives of the cost function with respect to an individual weight can be found by forming a chain rule of partial derivatives and working backward along the same connections as the original forward path. Since the sigmoid is a continuous function, the partials always exist. Hence, propagation of the errors backward during the learning stage requires essentially the same amount of computation time as the forward calculation of the network output.

As with all gradient descent methods, the presence of local minima prevent any guarantees being placed on the ability of the learning algorithm to converge to the optimal solution. Moreover, simple gradient descent algorithms tend to converge slowly, especially if there are "troughs" in the error surface (Baird (1991)). Since the goal of learning is to follow the gradient in a downhill direction, a small learning rate results in slow convergence. If the learning rate is too high, the weight vector may completely bypass the trough to some possibly suboptimal plateau or oscillate across the bottom of the trough with little movement in the direction of the minimum.

If an acceptable learning rate is used, or if one of several techniques for speeding up

convergence is applied (e.g., adding momentum terms to the weight update equation (Rumelhart, *et al.* (1986)) or using second order derivative information on the cost (Jacobs (1988))), backpropagation networks have shown the ability to adequately map highly nonlinear functions. In fact, sigmoidal backpropagation networks with more than one hidden layer can represent any function to a desired degree of accuracy given enough neurons and training samples (Funahashi (1988), Hornik, *et al.* (1989)). This universal function approximation property has played a major role in the resurgence of the sigmoidal backpropagation network in applications ranging from pattern recognition to automatic control. However, one should recall that due to the presence of local minima in J, there is no guarantee that a given learning rule will actually yield the weights that represent the desired mapping.

Many variants of connectionist networks have been developed in an attempt at improved learning. However, the majority of all systems have one common characteristic. Learning is essentially a process of functional approximation, where inputs and desired outputs are synthesized to form a multivariable, nonlinear mapping. The type of learning system used and its associated details are dependent on the specific application. Section 3.2 presents one such specialized approach that is used for the learning augmented control of a high performance aircraft.

2.3.4 Connectionist Learning Systems for Control

Due to an ability to approximate smooth multivariable, nonlinear functions, connectionist learning systems have generated a large amount of interest among control engineers. However, a single, systematic approach for the application of connectionist learning systems to control has not yet materialized. This section introduces a small subset of commonly used approaches for learning control and lists references were further discussions may be found.

Copying an existing controller is perhaps one of the simplest techniques in

applying connectionist learning systems to control. Assuming there exists a controller that is able to control the plant, the objective of the connectionist learning system is to synthesize the mapping between the inputs and the desired output supplied by the existing controller. Using this approach, the learning system can replace an existing controller in situations where the existing controller is impractical (e.g., where it is dangerous for a human to control the plant) or the learning system offers a less costly representation. This approach was successfully applied to a pole balancing problem by Widrow and Smith (1964), where the existing control was supplied by a human.

Direct inverse control is another method of applying a connectionist learning system to control (Werbos (1989)). Using this approach, the objective of the learning system is to identify the system inverse. This is accomplished by providing the output of the plant as the network input and the input to the plant (e.g., control signals) as the desired network output. If the network has a plant inverse (e.g., a unique plant input produces a unique plant output), then when the desired plant output is provided as input to the network, the resulting network output is the control to be used as input to the plant (Barto (1989)). The drawbacks to this technique are that a desired reference trajectory must be known in order to supply the network with the desired plant output and the inverse of the plant must be well-defined (e.g., a 1-to-1 mapping between inputs and outputs).

In the *backpropagation through time* method developed by Jordan (1988), two connectionist learning systems are used. The objective of the first network is to identify the plant, form which one can efficiently compute the derivative of the model output with respect to its input by means of back-propagation. Subsequently, propagating errors between an actual and a desired plant outputs back through this network produces an error in the control signal, which can be used to train the second network (Barto (1989)). This approach offers an improvement over direct inverse control since it is able to accommodate systems with ill-defined inverses, although the desired trajectory must still be known.

Another approach for incorporating learning into a control system is to augment an

adaptive technique with a learning system to form a hybrid controller (Baker & Farrell (1990), Baird (1991)). Augmentation of the adaptive technique is typically implemented using a direct or indirect approach. Using a direct adaptive approach, the learning system generates a control action (or set of parameters) associated with a particular operating condition. This control action is then combined with a control action produced by the adaptive system to arrive at the control that is applied to the plant. In contrast, for the indirect approach, the objective of the learning system is to improve the model of the plant. Here, the learning system generates model parameters that are a function of the operating condition. The parameters are combined with adaptive component estimates to arrive at a model of the plant. Given a presumably improved plant model, an on-line control law design is used to form the closed-loop system. A particular learning augmented indirect approach is used in this thesis and is developed in Chapter 3.

Reinforcement learning has also been suggested as a method of applying connectionist learning systems for control (Mendal & McLaren (1970), Barto (1989), Millington (1991)). The major difference between reinforcement learning and the previously discussed approaches is that under reinforcement learning, the objective is to optimize the overall behavior of the plant, so that no reference / desired trajectory is required. As a result, reinforcement learning essentially involves two problems, the construction of a *critic* that is capable of evaluating plant performance in a manner that is consistent with the actual control objective and the determination of how to alter controller outputs to improve performance as measured by the critic (Barto (1989)). The latter of the problems can be addressed by one of the previously discussed techniques.

3 TECHNICAL APPROACH

As discussed in Chapter 2, control law design for high performance aircraft presents challenging and unique problems to the designer. Traditional techniques have proven to be either prohibitively costly in terms of the effort required in tuning and the complexity of developing a multi-region linearized gain scheduling design, or have simply sacrificed performance for ease of design. This chapter formally presents an innovative method of integrating an adaptive component with a learning component to form a new hybrid control law. The hybrid system is presented by introducing each component separately and then combining the components in a synergistic arrangement to form a superior flight control system.

3.1 ADAPTIVE CONTROL COMPONENT

Numerous adaptive control techniques have been developed for nonlinear systems with unmodeled dynamics or model uncertainty (Astrom & Wittenmark (1989), Slotine & Li (1991)). One major class of adaptive control, model reference adaptive control (MRAC), is considered in this thesis. The majority of MRACs can be grouped into two general categories, namely, direct and indirect adaptive control. Direct adaptive control approaches are characterized by the synthesis of control signals directly from observed plant behavior without the benefit of an explicit plant model. In contrast, the indirect adaptive control methods rely heavily on an explicit plant model. The control law for an indirect technique employs a local plant model that is updated from observed plant behavior. Although developing and periodically updating a plant model is not without its own costs, indirect techniques have the advantage that many different control design

techniques that are based on explicit plant models can be used. In either case, the adaptive control system reacts to differences between desired and predicted behavior by adjusting internal parameters to achieve desired closed-loop performance. These differences in behavior are typically attributed to nonlinearities, unmodeled dynamics, model uncertainty, or exogenous disturbances.

Although conventional adaptive control methods have the ability to stabilize and control some nonlinear systems, the closed-loop system is often unable to match the performance of a well-designed and well-tuned gain scheduled controller. This difference in performance stems from inherent time-delays or lags associated with adaptive controllers. Typically, the process of updating parameters of an adaptive control law requires several control intervals to accurately detect and compensate for variations in the plant behavior. Sensor noise exacerbates this dilemma since the required filtering creates additional lag. Adaptive control approaches also have performance limitations when presented with quasi-static state dependent disturbances. In particular, since adaptive controllers are reactive by nature, they are unable to learn and subsequently predict repeatable state dependent behavior (Baker & Farrell (1992)). Even if the plant repeatedly experiences the same disturbance at a particular location in the state space, the adaptive controller must wait until the effects of the discrepancy are observed before it can initiate changes in the parameters. Hence, adaptive controllers fail to make complete use of experientially gained knowledge. As will be discussed in a following section, this inadequacy of adaptive control can be overcome with the addition of a learning component.

The primary role of the adaptive control component in the hybrid system is to accommodate unmodeled dynamical behavior (i.e., behavior that is not expected based on the design model). Additionally, the adaptive component of the hybrid system has the auxiliary task of presenting estimates of any observed unmodeled state dependent dynamic behavior to the learning component (i.e., unknown dynamics that are a function of state in areas of the state space where the learned mapping can be improved).

3.1.1 Control Law Derivation

Consistent with the above discussion, any adaptive control approach that is applicable to nonlinear dynamic systems with model uncertainty and that develops estimates of unknown state dependent components of the plant dynamics is a candidate for the adaptive component in the hybrid control system. Adaptive techniques that require small amounts of on-line computation are especially appealing since extra computing power will be required to train the learning component. One such adaptive control technique is based on Time Delay Control (TDC). Developed by Youcef-Toumi (1990), TDC is an indirect adaptive technique designed for the class of systems with discrete nonlinear dynamics represented in the following form:

$$\mathbf{x}(k+1) = \mathbf{g}\{\mathbf{x}(k),k\} + \mathbf{h}\{\mathbf{x}(k),k\} + \Gamma\mathbf{u}(k) + \mathbf{d}(k) \tag{3.1}$$

Notationally, **g** and **h** represent known (modeled) and unknown nonlinear plant dynamics vectors, respectively. These vectors are functions of the state **x** and discrete time k. Furthermore, **d** is an unknown, possibly time-varying disturbance vector. The control vector is represented by **u** and acts *linearly* through the control input matrix Γ on the new state. It will be assumed here (in this subsection only) that Γ is known without error. Section 3.3 addresses the issue of uncertainty in Γ.

Overview of TDC

TDC uses a simple estimation technique to detect and compensate for unknown dynamics and unexpected disturbances. By examining the difference in the dynamical behavior between the current state of the plant and that expected (given knowledge of the state and control at the previous time step and the modeled terms of Equation 3.1), TDC constructs a combined estimate of the unknown dynamics **h** and disturbances **d** at the current time. Using this estimate (formed from state and control information at the previous time step), it is possible to form a control law that attempts to cancel the undesired

known dynamics, the estimated unknown dynamics, and the estimated disturbances. Desired state dynamics can then simply be "inserted" along with a proportional error term to achieve desired tracking error dynamics.

Critical to the TDC control law is the method of obtaining the estimates of the unknown dynamics and unexpected disturbances. By employing information from the previous time step, TDC is able to react rapidly to changes in the dynamical behavior of the plant. This characteristic is ideal for systems that operate in an environment with large variations in the unknown dynamics and unexpected disturbances. However, this beneficial feature is not without some cost. Since TDC basically "differentiates" the state in arriving at the control action, any sensor noise affecting the observed values of the state and control will be amplified, resulting in noisy control signals and possible rate or position saturation of the actuators. This effect translates into poor performance and possibly to instabilities. To counter the effects of noise, filters are used. Although filters can accommodate noise, they add additional lag to the control system which reduces the performance of the adaptive system.

Development of TDC

The full development of the TDC control law is contained in Youcef-Toumi & Osamu (1990). For the sake of completeness, the fundamental equations are summarized below.

Assume that the plant can be written in the following form:

$$\mathbf{x}(k+1) = \mathbf{\Phi x}(k) + \mathbf{h}\{\mathbf{x}(k), k\} + \mathbf{\Gamma u}(k) + \mathbf{d}(k) \tag{3.2}$$

where $\mathbf{x}$ is an n dimensional state vector, $\mathbf{u}$ is an m dimensional vector of control inputs, $\mathbf{\Phi}$ is an n by n state transition matrix, $\mathbf{\Gamma}$ is an n by m control weighting matrix, and $\mathbf{h}$ and $\mathbf{d}$ are n dimensional unknown state dynamics and disturbance vectors respectively. Notice that Equation (3.2) is a special case of Equation (3.1), since the current state acts linearly on the new state. Here, $\mathbf{\Phi x}(k)$ can be viewed as the best time-invariant, linear

approximation of the known function $\mathbf{g}\{\mathbf{x}(k),k\}$, linearized about a selected operating condition. This assumption essentially shifts plant nonlinearities and time dependencies to the unknown dynamics term **h**.

Define a desired n dimensional reference trajectory $\mathbf{x}_m$ to be the following linear, time-invariant system:

$$\mathbf{x}_m(k+1) = \mathbf{\Phi}_m \mathbf{x}_m(k) + \mathbf{\Gamma}_m \mathbf{r}(k) \tag{3.3}$$

where $\mathbf{\Phi}_m$ is an n by n reference state transition matrix, $\mathbf{\Gamma}_m$ is an n by m reference model command weighting matrix, and $\mathbf{r}(k)$ is an m dimensional vector of reference commands. There is no requirement that the reference model be a linear, time-invariant system. Moreover, it is assumed that the reference command $\mathbf{r}(k)$ is constrained in a way that the desired reference trajectory is achievable by the system described by Equation (3.2).

The difference between the desired reference state and plant state is the error vector:

$$\mathbf{e}(k) = \mathbf{x}_m(k) - \mathbf{x}(k) \tag{3.4}$$

The control objective of TDC is to force this error vector to zero with the following desired error dynamics defined in terms of an error dynamics transition matrix $\mathbf{\Phi}_e$:

$$\mathbf{e}(k+1) = \mathbf{\Phi}_e \mathbf{e}(k) \tag{3.5}$$

By expressing $\mathbf{\Phi}_e$ in terms of $\mathbf{\Phi}_m$, the error dynamics can be written as

$$\mathbf{\Phi}_e = \mathbf{\Phi}_m + \mathbf{K} \tag{3.6}$$

where **K** can be viewed as an error feedback matrix.

The control signal **u** that yields the desired error dynamics is obtained by incrementing Equation (3.4) one time step forward and substituting Equations (3.2) through (3.6) as follows:

$$\begin{aligned}
\mathbf{e}(k+1) &= \mathbf{x}_m(k+1) - \mathbf{x}(k+1) \\
\mathbf{\Phi}_e \mathbf{e}(k) &= \mathbf{\Phi}_m \mathbf{x}_m(k) + \mathbf{\Gamma}_m \mathbf{r}(k) - \mathbf{\Phi}\mathbf{x}(k) - \mathbf{h}\{\mathbf{x}(k),k\} - \mathbf{\Gamma}\mathbf{u}(k) - \mathbf{d}(k) \\
\mathbf{\Gamma}\mathbf{u}(k) &= \mathbf{\Phi}_m \mathbf{x}_m(k) + \mathbf{\Gamma}_m \mathbf{r}(k) - \mathbf{\Phi}_m \mathbf{x}(k) - \mathbf{h}\{\mathbf{x}(k),k\} - \mathbf{d}(k) - \mathbf{\Phi}_e \mathbf{e}(k) \\
\mathbf{\Gamma}\mathbf{u}(k) &= [\mathbf{\Phi}_m - \mathbf{\Phi}]\mathbf{x}(k) + \mathbf{\Gamma}_m \mathbf{r}(k) - \mathbf{h}\{\mathbf{x}(k),k\} - \mathbf{d}(k) - \mathbf{K}\mathbf{e}(k)
\end{aligned} \tag{3.7}$$

Notice that the terms **h** and **d** on the right hand side of Equation (3.7) are unknown. They will be replaced by estimates $\hat{\mathbf{h}}$ and $\hat{\mathbf{d}}$. In particular, if **h** and **d** change relatively slowly, then their estimated value can be obtained by solving Equation (3.2) at the previous time step, yielding an estimate of the sum of the two terms **h** and **d**:

$$\begin{aligned}\hat{\mathbf{h}}\{\mathbf{x}(k),k\}+\hat{\mathbf{d}}(k) &= \mathbf{h}\{\mathbf{x}(k-1),k-1\}+\mathbf{d}(k-1)\\ &= \mathbf{x}(k)-\Phi\mathbf{x}(k-1)-\Gamma\mathbf{u}(k-1)\end{aligned} \tag{3.8}$$

Here we assume full knowledge of the state and control values $\mathbf{x}(k)$, $\mathbf{x}(k\text{-}1)$, and $\mathbf{u}(k\text{-}1)$.

Unless Γ^{-1} exists, which implies that $n = m$ so that the number of inputs equals the number of states, Equation (3.7) will not have a general, exact solution. Nevertheless, an approximate solution can be generated as follows

$$\mathbf{u}(k)=\Gamma^{+}\left[\left[\Phi_m-\Phi\right]\mathbf{x}(k)+\Gamma_m\mathbf{r}(k)-\mathbf{h}\{\mathbf{x}(k),k\}-\mathbf{d}(k)-\mathbf{K}\mathbf{e}(k)\right] \tag{3.9}$$

where Γ^{+} is the pseudo-inverse of Γ. The use of the pseudo-inverse of the control weighting matrix is necessitated by the fact that the majority of control systems have more states than controls. The following pseudo-inverse

$$\Gamma^{+}=\left[\Gamma^{\mathrm{T}}\Gamma\right]^{-1}\Gamma^{\mathrm{T}} \tag{3.10}$$

results in the minimization of the L_2 norm $\left\|\Gamma\Gamma^{+}-\mathbf{I}\right\|_2$.

Substituting Equations (3.8) into Equation (3.9) results in the TDC control law

$$\begin{aligned}\mathbf{u}(k)=&-\Gamma^{+}\mathbf{K}\mathbf{e}(k) && \text{(error feedback)}\\ &+\Gamma^{+}\left[\Phi_m-\Phi\right]\mathbf{x}(k) && \text{(state feedback)}\\ &+\Gamma^{+}\Gamma_m\mathbf{r}(k) && \text{(command feedforward)}\\ &-\Gamma^{+}\left[\hat{\mathbf{h}}+\hat{\mathbf{d}}\right] && \text{(cancellation)}\end{aligned} \tag{3.11}$$

The first term in Equation (3.11), *error feedback*, represents proportional feedback of the error between the desired and actual state at time k. The *state feedback* term determines the contribution of the state at discrete time k to the control. This term is a function of the difference between the desired trajectory dynamics and the linearized approximation of the plant dynamics. Commands enter the control law through the *command feedforward*

term. As compared to the feedback terms, the command term is feedforward in the sense that it is an open-loop term that is not a function of plant state. The *cancellation* term attempts to cancel the unknown dynamics and disturbances at the present time k by using approximations based on observed behavior at the previous time k–1.

3.1.2 Implementation Issues

The design parameters of the TDC control law include those associated with the reference model dynamics $\{\mathbf{\Phi}_m, \mathbf{\Gamma}_m\}$ and desired tracking error dynamics $\mathbf{\Phi}_e$ (or equivalently the error feedback matrix $\mathbf{K} = \mathbf{\Phi}_e - \mathbf{\Phi}_m$). Of course, these parameters cannot be selected in an arbitrary manner. As alluded to in the previous section, TDC requires the use of a pseudo-inverse in the control law calculation due to the fact that the majority of systems have more states than controls. Hence, the control weighting matrix is singular and cannot be exactly inverted. By inserting Equation (3.11) into (3.2), the following constraint must be met in order for the plant state to track the model state with the desired error dynamics:

$$\left\{\mathbf{I} - \mathbf{\Gamma}\mathbf{\Gamma}^{+}\right\}\left\{\left[\mathbf{\Phi}_m - \mathbf{\Phi}\right]\mathbf{x}(k) + \mathbf{\Gamma}_m\mathbf{r}(k) - \mathbf{h}\left\{\mathbf{x}(k),k\right\} - \mathbf{d}(k) - \mathbf{K}\mathbf{e}(k)\right\} = 0 \qquad (3.12)$$

Notice that if $\mathbf{\Gamma}$ is square and invertible, then the first factor on the left hand side guarantees that the constraint is always met. If this is not the case, then values for the design parameters $\mathbf{\Phi}_m$, $\mathbf{\Gamma}_m$, and $\mathbf{K}$ must be selected to minimize the error of Equation (3.12) for arbitrary $\mathbf{r}$, $\mathbf{h}$, and $\mathbf{d}$. Alternatively, $\mathbf{\Gamma}^{+}$ can be selected so that the nonzero second factor on the left hand side of Equation (3.12) is in the nullspace of $\{\mathbf{I}–\mathbf{\Gamma}\mathbf{\Gamma}^{+}\}$. However, the approaches for meeting the constraint of Equation (3.12) when $\mathbf{\Gamma}$ is non-square are generally difficult.

Beyond this constraint issue, the error feedback matrix $\mathbf{K}$ is chosen to achieve the desired error dynamics $\mathbf{\Phi}_e$. Typically, error dynamics have been chosen as a function of the reference model dynamics (e.g., twice as fast). However, other selections can be

accommodated as long as the error dynamics are stable.

Selection of a desired reference model $\{\Phi_m, \Gamma_m\}$ is frequently application specific. Although there is no requirement on the method used to generate a reference model for a flight control application, typical design specifications are often stated in terms of characteristics of linear, time-invariant (LTI) systems. For example, military aircraft must meet MIL-F-8785C (1980) specifications for natural frequency and damping ratio of their characteristic modes. Thus, a LTI system is often employed in the role of a reference model. The reference model for the aircraft control problem addressed by this thesis is discussed in Section 4.3.3.

3.2 LEARNING CONTROL SYSTEM

The purpose of the learning system in the hybrid control law is to synthesize a mapping between the state and controls of the plant and an estimate of the unknown dynamics $\hat{\mathbf{h}}$ generated by an adaptive component. As discussed previously, connectionist networks have demonstrated an ability to learn highly nonlinear, multivariable mappings. In this section, the complete development of the learning system employed in the hybrid controller is presented.

3.2.1 Incremental Learning and Fixation

Since the objective of the network in control applications is to synthesize a mapping over a continuous input space, the training cost function in Equation (2.2) cannot be used directly (i.e., the number of training examples is not a finite set). As a result, one common approach for systems with a continuous input space is to use *incremental learning* (Baker & Farrell (1991), Rumelhart, *et al.* (1986)). Incremental learning algorithms seek to reduce a cost function defined in terms of the current input rather than a cost function defined over a fixed set of samples as in Equation (2.2). Using this approach, the cost

function defined in Equation (2.2) reduces to the single term

$$J = \frac{1}{2}\left[\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x},\mathbf{p})\right]^{T}\left[\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x},\mathbf{p})\right] \tag{3.13}$$

where $\mathbf{d}(\mathbf{x})$ is the desired network output at current state $\mathbf{x}$ and $\mathbf{f}_{net}(\mathbf{x},\mathbf{p})$ is the output of the network as a function of state and parameters $\mathbf{p}$. The general parameter vector $\mathbf{p}$ is used in Equation (3.13) to allow for nodal functions that are not simply a function of the state and weights. An incremental learning approach essentially provides a convenient, point-wise contribution to an aggregate cost function similar to Equation (2.2) since it can be computed quickly and efficiently.

During incremental learning, care must be taken to ensure that samples are sufficiently distributed throughout the input space so that over a finite period of time, the individual point-wise contributions of Equation (3.13) collectively provide an approximation to a batch-type cost function in Equation (2.2). Since parameters are updated at each sample, the network reacts to mapping errors at the current input. Unfortunately, sigmoidal networks possess a relatively high degree of "generalization," where parameter changes impact the network mapping in vast regions of the input space. As a consequence, the localized nature of incremental learning can result in "fixation" of the network, where the network attempts to achieve an accurate mapping at the current state while potentially degrading an acceptable mapping already learned in other regions of the input space.

The degree of fixation is determined by the rate of mapping degradation in outlying regions *relative* to the time required to receive samples from all regions of input space. This rate of outlying mapping degradation is in turn determined by the degree of generalization and the learning rate. A network with a high degree of generalization requires rapid and extensive distribution of sampling points or a very small learning rate to avoid fixation. For control problems, the former is generally not possible since the sampling process is constrained by the system dynamics. Furthermore, extensive

investigation of the state space is typically inconsistent with the control objectives. This point is most evident in regulation, where the goal is to keep the system near some operating point. Due to such constraints on the sampling process, an alternative approach to avoiding fixation during incremental training is to reduce the degree of global generalization in the network. Such spatially localized networks are discussed in the next section.

3.2.2 Spatially Localized Learning

The basic idea of spatially localized learning is that experience (learning) in a local region of the input space should only affect the mapping in that particular locality, with a marginal effect in all other areas. Spatially localized learning prevents knowledge that has already been collected in other regions of the mapping from becoming incorrectly perturbed (i.e., corrupted). This is accomplished by lessening the extent of generalization (i.e., where previous learning under *similar* situations is used to approximate the current situation) to include only a local region. Figure 3.1 illustrates the concept of spatially localized learning. Let $\mathbf{f}_{net}(\mathbf{x}; p^0, p^1, \ldots, p^N)$ represent the mapping to be learned where x is the input vector and $p^0, \ldots, p^N$ are a set of parameters to be learned that define the mapping

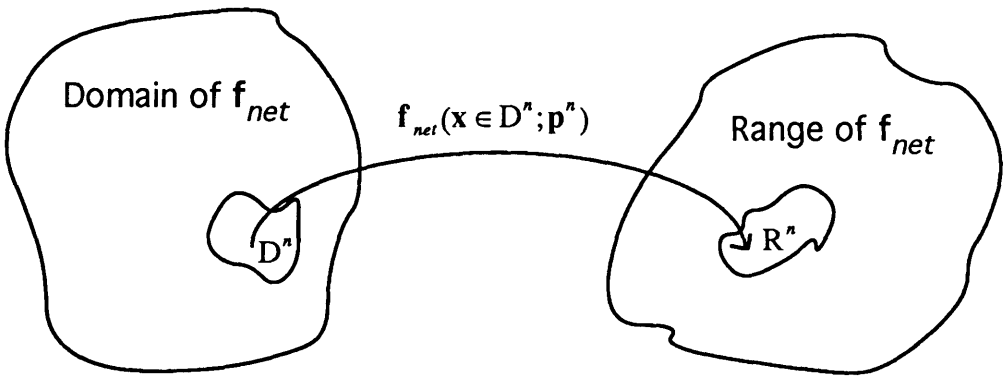

Figure 3.1 Spatially Localized Learning: the Ideal Case

Figure 3.1 shows a region of the domain D^n of the function to be learned being mapped to an associated region of the range R^n. The ideal situation for localized learning, as indicated

in the figure, is that this local mapping is a function of a distinct subset of the parameter set denoted $\mathbf{p}^n$. Thus, the learning based on samples in D^n will only cause the subset of parameters, $\mathbf{p}^n$, to change. Of course, this represents the ideal situation which is not practical for a variety of reasons. However, the objective that the results of learning from the input samples in one region of the domain should not significantly alter previously learned mappings in distant regions of the domain is desirable.

This local generalization property of spatially localized learning contrasts with typical structures (e.g., sigmoidal networks) that are characterized by a much larger, more global generalization due to its non-local parameters. The following discussion introduces and develops one example of a spatially localized learning system that is used in the hybrid control system. Learning is accomplished by an incremental gradient descent learning algorithm.

3.2.3 The Linear-Gaussian Network

One learning system design that exhibits spatially localized properties is the linear-Gaussian network. The linear-Gaussian network is an example of a local basis / influence function system (Baker & Farrell (1992), Millington (1991)). The network mapping is constructed from a set of hyperplanes that act as "basis" functions over a localized region of the input space. Although many functions could be used as a local basis, hyperplanes offer an attractive choice for the control problem due to their simple structure and similarity with conventional gain scheduled mappings. The influence function associated with each local basis function is an elliptic hyper-Gaussian. As the name suggests, the role of the influence function is to determine the region of applicability of a particular local basis function in the input space. For example, a basis function associated with a hyper-Gaussian whose center is very close to the current input plays a much larger role in the determination of the output of the mapping than a basis function whose Gaussian is centered far away from the current input. The following discussion details the linear-

Gaussian network.

Node Descriptions

The local basis function of a linear-Gaussian node is formed by adding the weighted sum of the inputs with a bias. Equation (3.14) shows the relationship between the i^{th} linear basis function $\mathbf{L}_i$ and its inputs, $\mathbf{x}$:

$$\mathbf{L}_i(\mathbf{x}) = \mathbf{W}_i(\mathbf{x} - \mathbf{x}_{io}) + \mathbf{b}_i \tag{3.14}$$

Here, if n is the number of node inputs and m is the number of node outputs, then $\mathbf{L}_i$ is an m dimensional vector, $\mathbf{x}$ is a n dimensional vector of node inputs, $\mathbf{W}_i$ is an mxn matrix whose elements are the weights on the input, $\mathbf{x}_{io}$ is a n dimensional vector that represents the center of the Gaussian nodal function described below, and $\mathbf{b}_i$ is a m dimensional bias vector.

The linear-Gaussian node uses a hyper-Gaussian as an influence function for the basis $\mathbf{L}_i$ in Equation (3.14). The value for the i^{th} Gaussian function G_i is given by:

$$G_i(\mathbf{x}) = \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{x}_{io})^T(\mathbf{D}_i)^2(\mathbf{x} - \mathbf{x}_{io})\right] \tag{3.15}$$

where $\mathbf{D}_i$ is a diagonal matrix containing values for the spatial decay of the Gaussian, $\mathbf{x}_{oi}$ is the Gaussian center, and $\mathbf{x}$ is the input vector. Figure 3.2 contains an illustration of a typical Gaussian function.

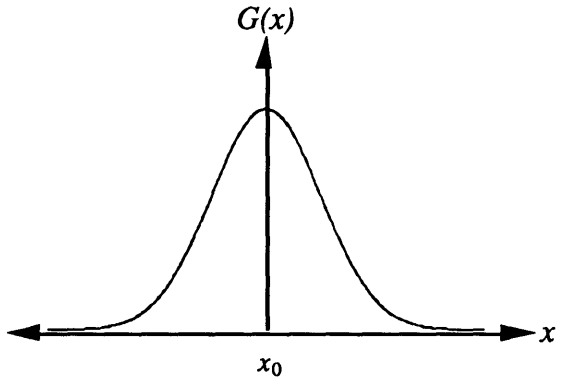

Figure 3.2 Gaussian Function

The Gaussian is a continuous function with finite asymptotic values. Moreover, a Gaussian is differentiable over the entire input space, which is important in any learning algorithm that relies on partial derivative information for training (e.g., gradient methods). The output of the linear-Gaussian node is simply the product of the linear basis function and the Gaussian influence function. The general structure is shown in Figure 3.3

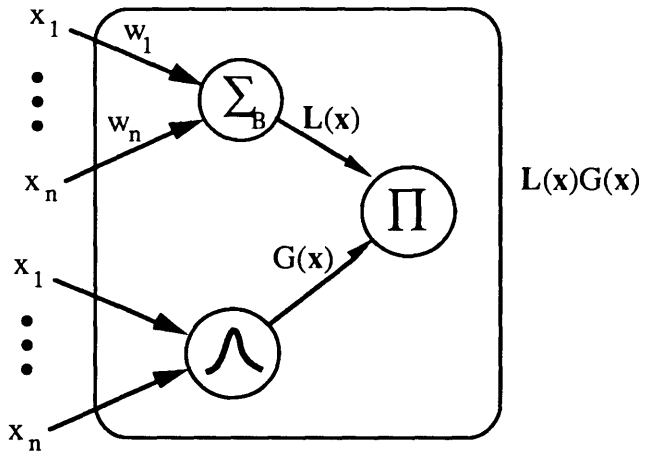

Figure 3.3 Linear Gaussian Node

where Σ_B represents a summing node with bias and Π a multiplication node. By dividing the i^{th} Gaussian function by the sum of all the Gaussians, the resulting quotient is the normalized i^{th} influence function, Γ_i. This relationship is shown in Equation 3.16 below

$$\Gamma_i(\mathbf{x}) = \frac{G_i(\mathbf{x})}{\sum_{i=1}^{n} G_i(\mathbf{x})} \tag{3.16}$$

where

$$\sum_{i=1}^{n} \Gamma_i(\mathbf{x}) = 1 \quad \text{and} \quad 0 < \Gamma_i(\mathbf{x}) \leq 1 \tag{3.17}$$

Combining Equations (3.14) through (3.16) yields the following equation for the m dimensional vector output of a linear-Gaussian network:

$$\mathbf{Y}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{L}_i(\mathbf{x})\Gamma_i(\mathbf{x}) \tag{3.18}$$

Network Architecture

Linear Gaussian networks use a feedforward architecture and consist of three main layers: input, hidden, and output as depicted in Figure 3.4. The first layer of the network, the input layer, simply passes the input values to the subsequent hidden layer. As one would expect, there are the same number of input nodes as there are network inputs. The hidden layer is not directly observable to the external environment. This hidden layer contains two elements, namely the linear-Gaussian nodes and nodes that normalize the Gaussian influence function. By adding enough linear-Gaussian nodes, a single hidden layer network can provide arbitrarily accurate function approximations. Furthermore, multiple hidden layers of linear-Gaussian nodes lead to non-localized mappings. For these reasons, only linear-Gaussian networks with a single hidden layer are investigated. The final layer is the output layer. It contains as many nodes as there are outputs. A typical linear-Gaussian network is shown in Figure 3.4

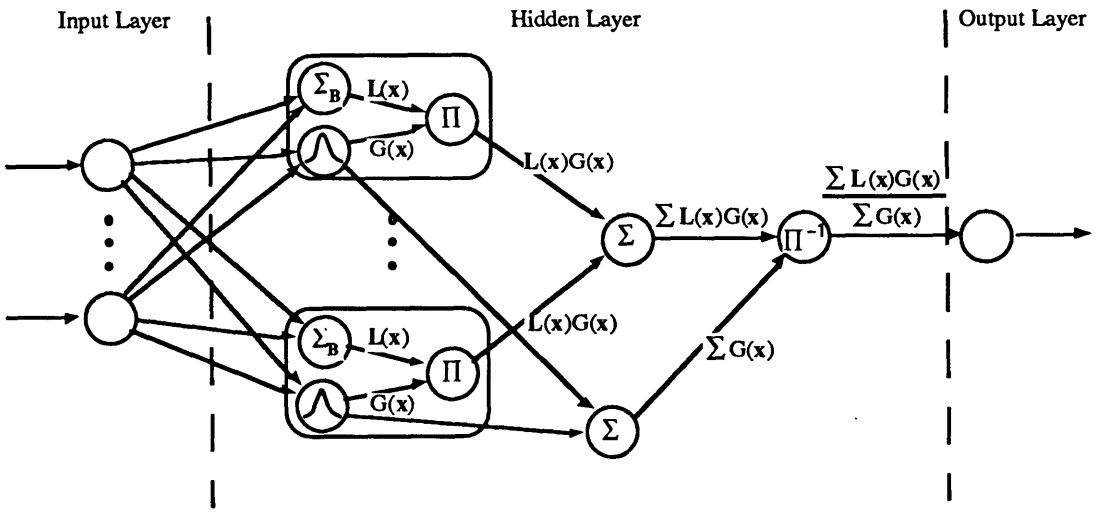

Figure 3.4 Multi-Input, Single Output Linear Gaussian Network

where the negative sign of the rightmost Π-node indicates that the argument is reciprocated prior to the multiplication.

Learning Algorithm

The linear-Gaussian network uses a supervised, incremental gradient descent algorithm to adjust the network parameters in the negative direction of their gradient with the cost function:

$$\mathbf{p}_i(k+1) = \mathbf{p}_i(k) - \alpha \left.\frac{\partial J}{\partial \mathbf{p}_i}\right|_k \quad ; \quad \alpha > 0 \tag{3.19}$$

where $\mathbf{p}_i$ is a vector of the adjustable parameters of the i^{th} node (e.g., weight matrix elements, bias, spatial decay, or center) and J is the cost at a particular training sample, and α is the learning rate. The typical cost used for linear-Gaussian networks is shown in Equation (3.20)

$$J = \frac{1}{2}\left[\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x},\mathbf{p})\right]^{\mathrm{T}}\left[\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x},\mathbf{p})\right] \tag{3.20}$$

where $\mathbf{d}$ is the desired output as a function of input state $\mathbf{x}$, and $\mathbf{f}$ is the output of the network as a function of input state and network parameters $\mathbf{p}$. In minimizing the cost at

each step (i.e., for each training sample), all of the parameters, or just a subset, can be adjusted using Equation (3.19). The local learning rate for each parameter can be adjusted independently in order to achieve a more rapid convergence.

Besides the basic nodal and architectural differences, the learning algorithm of the linear-Gaussian network also differs from that of the classic sigmoidal network. Since the normalized Gaussian influence functions represent a measure of the significance that each node has on a particular value of the input **x** (i.e., the influence of each node on the output for a given input), it is reasonable to eliminate insignificant nodes from the learning calculation. Due to the elimination of these nodes, the computational efficiency and hence the training time of the network are improved. For example, for a given input value **x**, the learning algorithm might first order the Gaussian nodes by their normalized influence and use only enough nodes so that the sum of the normalized influences equals or exceeds some threshold value (e.g., 99%). Since the remaining nodes have only a minor effect on the local region, their outputs need not be included in the parameter update. For large networks, this can result in a substantial reduction in computation.

Application Issues

The number of nodes needed by a linear-Gaussian network is dependent on the characteristics of the function it is attempting to approximate and on any requirements placed on the desired rate of convergence and the level of acceptable errors in the learned mapping. Although no set of strict rules has been developed for selecting the number of nodes, several guidelines do exist. For functions that are very smooth, the mapping can be realized with relatively few nodes spread evenly throughout the input space. A large number of nodes will not improve this mapping and will only serve to increase the network training time. However, more complex functions with large local variations will require a large number of nodes, each with a relatively small sphere of influence.

The sphere of influence of a Gaussian function is determined by the spatial decay

matrix. Hence, the spatial decay matrix is a factor in determining the size of the local regions in the input space. If the spatial decay is large, the transitions between the regional linear basis functions will be more abrupt if the density of basis functions is not high. This property is ideal for more complex functions. However, a large spatial decay will require many more nodes to sufficiently map the entire input space. In contrast, small spatial decay rates result in large regions of influence that are ideal for smooth, slowly varying functions.

Initial values for the weights, biases, and Gaussian centers must also be selected. The basis function described by a weight matrix and bias vector represent a best guess of the desired mapping based on *a priori* information. Hence, values for the weights and biases can be initialized from an existing gain scheduled controller or other linearizable control law. In cases with considerable *a priori* knowledge the adjustable parameters are presumably much closer to their optimal values, and training time is greatly reduced. If no *a priori* knowledge is available about the mapping, the weights and biases are set to zero. The initialization of the Gaussian centers effectively locates the influence functions in the input space, and generally, the centers are placed so that the entire input space is spanned.

In summary, a linear-Gaussian network is one example of a spatially localized learning system. This network combines linear basis functions with Gaussian functions to provide the properties of local learning and the generalization properties of typical connectionist networks. Networks that employ spatially localized learning are required for control systems that regularly encounter scenarios that might cause fixation, as described in Subsection 3.2.1. Although linear-Gaussian networks tend to require more nodes and thus more memory (due to localization), improved learning efficiency and, more importantly, better functional mappings can be obtained.

3.3 HYBRID LEARNING / ADAPTIVE CONTROL

The hybrid control law developed in this section represents one approach to

combining a learning system with an adaptive component with the objective of improving performance in the presence of unmodeled dynamics and model uncertainty. In augmenting an indirect adaptive controller with a connectionist learning system, the general goal is to develop a control law that combines the strengths of each component.

3.3.1 Hybrid Control System Architecture

Adaptive control systems are capable of controlling complex dynamic systems. However, traditional adaptive control techniques only react (after the fact) to differences between actual and expected behavior — they have no anticipatory capacity. Learning in connectionist systems is fundamentally a process of function approximation. Hence, given the vehicle state and the applied control as an input and the unknown dynamics as desired outputs, a connectionist learning system is capable of realizing a mapping of the state and control dependent dynamics. Thus, by augmenting an adaptive controller with a learning system, it is possible to anticipate the state dependent components of the plant dynamics by "looking up" the values of that component of the dynamics for the current situation from the network, instead of waiting for an adaptive component to react to observed differences between the actual and expected state values. *By incorporating a learning system into the control law, the hybrid controller is able to use experientially gained knowledge.* Figure 3.5 illustrates the control system architecture of the hybrid adaptive / learning controller (Baker & Farrell (1991)).

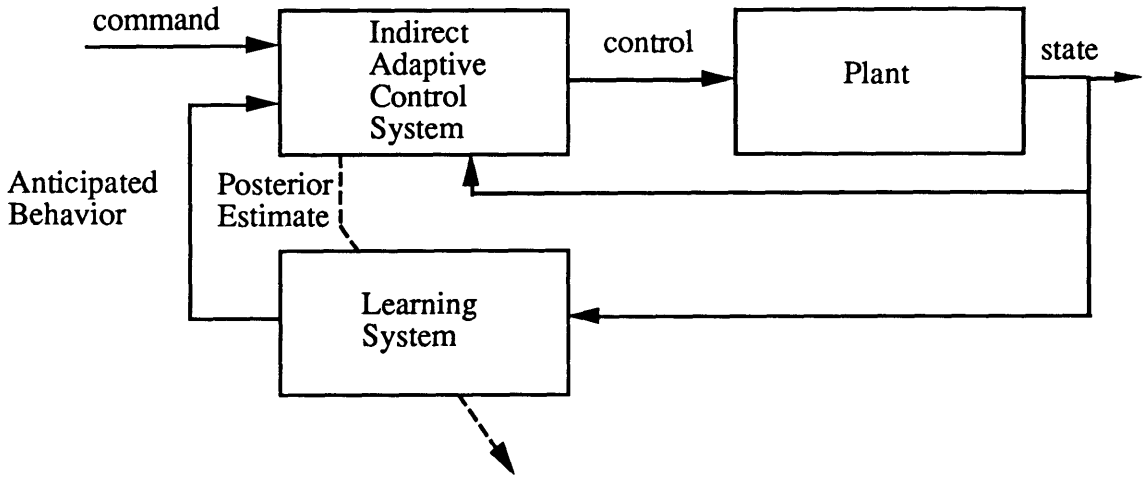

Figure 3.5 Hybrid System Architecture

The role of each component in the hybrid system is straightforward. The adaptive component provides an adaptive capability to accommodate unmodeled dynamic behavior that is not expected (based on the design model). Moreover, it provides a posterior estimate of any unmodeled state and control dependent behavior which can be used to train the learning system. The role of the learning system is to anticipate vehicle behavior that varies predictably with the current state and control.

3.3.2 Learning Versus Adaptation

Since both the adaptive component and the learning component in the hybrid control law derived in the previous section are based on parameter adjustment algorithms that use information gained by observing the closed-loop behavior of the plant, one might think it is difficult to distinguish between the two components. However, for the majority of systems, distinguishing qualities do exist. The following discussion presents the different goals and characteristics of the adaptive and learning components in an attempt to differentiate the two.

The adaptive component of the hybrid system can be characterized by its reactive approach to accommodating local disturbances and *apparent* time-varying dynamics. Nonlinearities that are a function of the operating condition of the plant appear to the

adaptive component as time-varying dynamics when they are actually changes in the local linearized behavior of a nonlinear, time-invariant plant. Since adaptive controllers typically lack the ability to associate the required changes in the control action as a function of the operating conditions, the controller must continually adapt to all unexpected effects, even those which are experienced repeatedly and are actually due to time-invariant nonlinearities. In other words, adaptive controllers have no "memory" and are unable to anticipate dynamics that are strictly a function of state. Thus, this lack of memory prevents any anticipatory action by the controller. Moreover, to prevent a situation where the adaptive controller is continuously in some suboptimal, partially adapted state, the generation of the unknown dynamics estimate must be relatively fast when compared to the plant dynamics. In summary, the adaptive component reacts to unexpected effects in order to maintain locally desired behavior; it is best at accommodating novel situations and slowly time-varying dynamics.

The reactive characteristics of the adaptive component directly contrasts with the constructional emphasis of the learning component. In particular, the objective of the learning component in the hybrid control law is to associate initially unknown state dependent dynamics with the state and control at the current operating condition. The association is essentially a memory function (or mapping) that stores experientially gained knowledge. This knowledge of originally unknown dynamics can be exploited by the hybrid control system as a means of anticipating transient behavior instead of waiting to react to errors observed in the output. Moreover, this allows the adaptive component to focus on accommodating slowly varying exogenous (non state dependent) disturbances. Since the objective of learning is to realize a mapping of state dependencies over the *entire* operating envelope, the learning system is characterized by a global optimization and relatively slow dynamics. Table 3.1 summarizes the major differences between adaptation and learning (Baker & Farrell (1991)).

Table 3.1 Adaptation vs. Learning

ADAPTATION	LEARNING
reactive: maintain desired behavior (local optimization)	*constructional*: synthesize desired behavior (global optimization)
temporal emphasis	spatial emphasis
no "memory" $\Rightarrow$ no anticipation	"memory" $\Rightarrow$ anticipation
fast dynamics	slow dynamics

The goal of the hybrid controller is to combine the different behavioral characteristics of the adaptive and learning components in a synergistic fashion. Ideally, the adaptive controller accommodates local unmodeled dynamics and novel state dependencies, while the learning system is responsible for reducing state and control dependent model uncertainty.

3.3.3 Control Law Development

As discussed previously, TDC is one example of a particularly simple indirect adaptive control approach. Recall that TDC calculates an estimate of the sum of the unknown dynamics **h** and disturbances **d** at the previous time step by examining the difference in the dynamical behavior between the current state of the plant and the expected state given the state and control at the previous time step. Assuming that **h** and **d** do not change significantly over a control time step, TDC uses this old value of the sum of **h** and **d** in formulating the control law. By integrating TDC with a learning component to form a hybrid controller, this delay in the estimated value of **h** can be eliminated. Although this delay is possibly insignificant with short control cycle times in the absence of sensor noise, such is not the case in a more realistic environment. If the control law is generated at a fast rate, the unknown dynamics and disturbances at the previous time will accurately reflect the current values (in the absence of noise). However, as the cycle time is increased, the

potential for error in the estimates grows. If sensor noise is present, it is still possible to predict the current state within a given tolerance. However, since **h** and **d** are essentially found by taking the derivative of the state, sensor noise can have a large impact on these estimates and subsequently the control generated by TDC.

The most common technique for offsetting the effects of sensor noise is to use a filter. Note, however, that filtering the noise only adds to the delay already associated with **h** and **d**. For this reason, a hybrid approach can offer significant advantages due to the use of a connectionist learning system. Since sensor noise can alter the estimates of **h** and **d** significantly, it is possible to have conflicting desired output values for the same input (over time). Given this contrasting information, connectionist systems tend to learn the average value. Thus, if the sensor noise is assumed zero mean, which is the assumed case, the correct mapping will still be realized by the learning system. Since the recall of the learned estimates of the state dependent dynamics is nearly instantaneous, *the hybrid system is essentially able to remove the delay associated with the adaptive component.*

As alluded to earlier, the hybrid control law can be derived by augmenting the TDC equations with a learning component. Assume the nonlinear plant can be written in the following form

$$\mathbf{x}(k+1) = \Phi\mathbf{x}(k) + \Gamma\mathbf{u}(k) + \mathbf{f}_{net}\{\mathbf{x}(k),\mathbf{u}(k)\} + \mathbf{h}\{\mathbf{x}(k),\mathbf{u}(k),k\} + \mathbf{d}(k) \qquad (3.21)$$

where $\mathbf{x}(k)$ is an n dimensional state vector at discrete time k, $\mathbf{u}(k)$ is an m dimensional control vector at k, Φ is an n by n state transition matrix, Γ is an n by m control weighting matrix, **h** and **d** are n dimensional unknown dynamics and disturbances vectors respectively, and $\mathbf{f}_{net}$ is the n dimensional learned component of the state dependent dynamics. Equation (3.21) differs from the form of the plant model used in the TDC derivation only by the learned dynamics term $\mathbf{f}_{net}$. Moreover, the unknown dynamics term is allowed to be a function of control as well as state, essentially accounting for errors in the assumed (but unlikely to be) perfectly known Γ.

As before, the desired reference trajectory is given by

$$\mathbf{x}_m(k+1) = \mathbf{\Phi}_m \mathbf{x}_m(k) + \mathbf{\Gamma}_m \mathbf{r}(k) \tag{3.22}$$

Here $\mathbf{x}_m(k)$ represents the n dimensional desired model state vector at time k, $\mathbf{\Phi}_m$ is the n by n state transition matrix defined by the linear relationship between the current and next state, $\mathbf{r}(k)$ is the m dimensional command vector and $\mathbf{\Gamma}$ is the n by m model command weighting matrix. As was the case with the derivation of the TDC control law in Section 3.1.1, there is no requirement that the reference model be linear. The only requirement on the reference model is that the desired trajectory is achievable, otherwise the control law may saturate the effectors and yield unsatisfactory performance.

If the difference between the desired reference state and plant state at discrete time k is represented by the error vector

$$\mathbf{e}(k) = \mathbf{x}_m(k) - \mathbf{x}(k) \tag{3.23}$$

then the control objective of the hybrid control law is to force this error vector to zero with the following dynamics

$$\mathbf{e}(k+1) = \left[\mathbf{\Phi}_m + \mathbf{K}\right]\mathbf{e}(k) = \mathbf{\Phi}_e \mathbf{e}(k) \tag{3.24}$$

where $\mathbf{K}$ is interpreted as the error feedback matrix and $\mathbf{\Phi}_e$ is the desired error dynamics matrix.

If Equations (3.21) through (3.23) are substituted into Equation (3.24), the control signal $\mathbf{u}$ that yields the desired error dynamics is obtained from:

$$\mathbf{\Gamma u}(k) = \left[\mathbf{\Phi}_m - \mathbf{\Phi}\right]\mathbf{x}(k) + \mathbf{\Gamma}_m \mathbf{r}(k) - \mathbf{f}_{net}\left\{\mathbf{x}(k), \mathbf{u}(k)\right\} - \mathbf{h}\left\{\mathbf{x}(k), k\right\} - \mathbf{d}(k) - \mathbf{K}\mathbf{e}(k) \tag{3.25}$$

To isolate the functions on the right hand side of Equation (3.25) that are dependent on $\mathbf{u}$ at the current time k, approximations for the unknown dynamics and disturbances as well as the output of the connectionist network are made.

If $\mathbf{h}$ and $\mathbf{d}$ change relatively slowly, then their estimated value can be obtained (as before) by solving Equation (3.21) at the previous time step, yielding

$$\begin{aligned}\hat{\mathbf{h}}\{\mathbf{x}(k),\mathbf{u}(k),k\}+\hat{\mathbf{d}}(k) &= \mathbf{h}\{\mathbf{x}(k-1),\mathbf{u}(k-1),k-1\}+\mathbf{d}(k-1)\\ &= \mathbf{x}(k)-\boldsymbol{\Phi}\mathbf{x}(k-1)-\boldsymbol{\Gamma}\mathbf{u}(k-1)-\mathbf{f}_{net}\{\mathbf{x}(k-1),\mathbf{u}(k-1)\}\end{aligned} \tag{3.26}$$

The network function $\mathbf{f}_{\text{net}}$ in Equation (3.25) can be approximated using the first-order Taylor series expansion shown in Equation (3.27) to isolate the linear dependence on $\mathbf{u}$ at time k. Since the network is continuously differentiable, the Jacobian in Equation (3.27) are known to exist. Moreover, this Jacobian information is already calculated since it is needed for the learning algorithm discussed in Section 3.2.

$$\mathbf{f}_{net}\{\mathbf{x}(k),\mathbf{u}(k)\} \approx \mathbf{f}_{net}\{\mathbf{x}(k),\mathbf{u}(k-1)\}+\left.\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right|_{\mathbf{x}(k),\mathbf{u}(k-1)} \bullet \left(\mathbf{u}(k)-\mathbf{u}(k-1)\right) \tag{3.27}$$

Substituting these approximations into Equation (3.25) and solving for $\mathbf{u}$ at the current time k (using a pseudo-inverse) yields the following hybrid control law:

$$\begin{aligned}\mathbf{u}(k) = &-\left[\boldsymbol{\Gamma}+\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right]^{+}\mathbf{K}\mathbf{e}(k)\\ &+\left[\boldsymbol{\Gamma}+\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right]^{+}\left[\boldsymbol{\Phi}_m-\boldsymbol{\Phi}\right]\mathbf{x}(k)\\ &+\left[\boldsymbol{\Gamma}+\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right]^{+}\boldsymbol{\Gamma}_m\mathbf{r}(k)\\ &-\left[\boldsymbol{\Gamma}+\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right]^{+}\left[\hat{\mathbf{h}}+\hat{\mathbf{d}}\right]\\ &-\left[\boldsymbol{\Gamma}+\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right]^{+}\left[\mathbf{f}_{net}\{\mathbf{x}(k),\mathbf{u}(k-1)\}-\left.\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right|_{\mathbf{x}(k),\mathbf{u}(k-1)} \bullet \mathbf{u}(\mathbf{k}-1)\right]\end{aligned} \tag{3.28}$$

The differences between the TDC control law in Equation (3.11) and the hybrid control law are the result of the added learning terms. The fifth term in Equation (3.28) represents the learned state dependent dynamics. The partial derivative of the network output with respect to the control used in the pseudo-inverse calculation is a linear correction for errors in $\boldsymbol{\Gamma}$ as discussed below.

Beyond removing the delay associated with a purely adaptive controller, the hybrid control system is able to reduce model uncertainty. This is accomplished by using partial

derivative information for the learned network term with respect to the control inputs. For example, if there are errors in the coefficients of the assumed linear control weighting matrix Γ, or the control actually affects the next state in a nonlinear fashion, the partial of the learned dynamics with respect to the control represents the locally linearized unmodeled effect of the controls. Assuming accurate derivative information can be obtained from the network, the actual manner that the controls impact the next state is thus the assumed linear control weighting matrix corrected by this learned effect. *The technique of using the partial information to improve the a priori design and ultimately reduce model uncertainty represents a potentially major improvement over the TDC control law.*

4 EXPERIMENTS

A learning enhanced hybrid flight control system is demonstrated using the realistic model of a high-performance, supersonic aircraft that is described in Section 4.2. However, because the complexity of this aircraft model makes control system analysis difficult, the hybrid controller is first applied to a relatively simple nonlinear aeroelastic oscillator, described in Section 4.1. For this simple example, an exact truth model of the nonlinear plant dynamics is known, and the mapping that is synthesized by the control system can be compared to the known dynamics.

The objective of the experiments detailed in this section is to illustrate some of the properties of the hybrid control system. In particular, the goal is to demonstrate the ability of the hybrid system to improve the control of a nonlinear plant with model uncertainty and unmodeled dynamics that are a function of state and control. Both the aeroelastic oscillator and the high performance aircraft fall into this category. A secondary objective is to illustrate the learning characteristics of spatially localized connectionist networks when applied to control systems.

Section 4.1 and Section 4.2 each begins with a description of the plant model of interest (i.e., the aeroelastic oscillator and the high performance aircraft) and the physical motion that the model represents. This description is followed by a brief discussion of the open-loop dynamics and other characteristics of that model. Next, the reference model, along with the motivation for its selection, is presented. Issues in applying the hybrid control law to each plant are also discussed. This development is followed by two experiments for each plant that highlight the capabilities of the hybrid controller.

4.1 AEROELASTIC OSCILLATOR

4.1.1 Description

The aeroelastic oscillator models the motion of a square prism in a steady wind with an external control force. If the aeroelastic oscillator is constrained to translational motion normal to the incident wind, the dynamics resemble a classic mass-spring-dashpot system with an additional aerodynamic lift force due to an effective angle-of-attack between the wind and the prism (Parkinson & Smith (1963)). Figure 4.1 illustrates the aeroelastic oscillator model, where V(t) is the incident wind, L(t) is the aerodynamic lift force, f(t) is the control force, m is the mass of the square prism, r is the damping coefficient, and k is the spring constant. The two state variables, position $x(t)$ and velocity $v(t)$, represent motion normal to the incident wind.

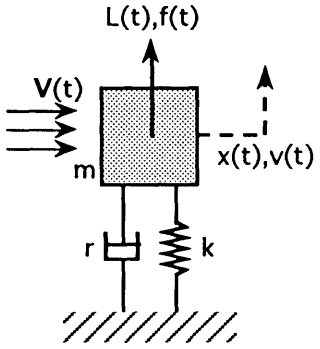

Figure 4.1 Aeroelastic Oscillator Model

The aerodynamic lift force is a nonlinear function of the effective angle-of-attack of the prism with respect to the incident wind. The effective angle-of-attack is due to the motion of the prism as illustrated in Figure 4.2, where α denotes the effective angle-of-attack and V_{REL} is the relative velocity.

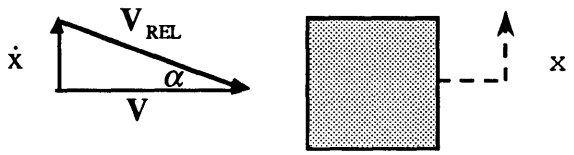

Figure 4.2 Effective Angle-of-Attack

Although current aerodynamic theory does not offer an analytic solution for the flow around a square prism, experimental data has been used to develop an approximation to the coefficient of lift (C_L) as a seventh-order polynomial of the effective angle-of-attack (Parkinson & Smith (1963)). Expressions for the coefficient of lift as well as for the resulting lift force L are given by:

$$C_L = A_1\left(\frac{\dot{x}}{V}\right) - A_3\left(\frac{\dot{x}}{V}\right)^3 + A_5\left(\frac{\dot{x}}{V}\right)^5 - A_7\left(\frac{\dot{x}}{V}\right)^7 \tag{4.1}$$

$$L = \frac{1}{2}\rho V^2 hlC_L \tag{4.2}$$

where the small angle approximation $\alpha = \dot{x}/V$ has been used, ρ is the air density, h is the side length of the prism, and l is the axial length of the prism. The differential equation governing the dynamics of the aeroelastic oscillator is:

$$m\frac{d^2x}{dt^2} + r\frac{dx}{dt} + kx = L + f \tag{4.3}$$

Equation (4.3) can be nondimensionalized by dividing through by kh and making the following change of variables:

$$X = \frac{x}{h};\quad n = \frac{\rho h^2 l}{2m};\quad \omega = \sqrt{\frac{k}{m}};\quad U = \frac{V}{\omega h};\quad b = \frac{r}{2m\omega};\quad \tau = \omega t$$

Applying this change of variables and substituting for the lift from Equation (4.2) yields:

$$\frac{d^2X}{d\tau^2} + 2b\frac{dX}{d\tau} + X = nA_1U\frac{dX}{d\tau} - \frac{nA_3}{U}\left(\frac{dX}{d\tau}\right)^3 + \frac{nA_5}{U^3}\left(\frac{dX}{d\tau}\right)^5 - \frac{nA_7}{U^7}\left(\frac{dX}{d\tau}\right)^7 + f \tag{4.4}$$

Equation (4.4) can be rewritten in a state space realization as:

$$x_1 = X; \quad x_2 = \frac{dX}{d\tau} \tag{4.5}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & (nA_1U - 2b) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f + \begin{bmatrix} 0 \\ -\frac{nA_3}{U} x_2^{\,3} + \frac{nA_5}{U^3} x_2^{\,5} - \frac{nA_7}{U^7} x_2^{\,7} \end{bmatrix} \tag{4.6}$$

where x_1 and x_2 are nondimensionalized position and velocity states, respectively.

Although the aeroelastic oscillator is a relatively simple second-order plant with a single control variable (force f), it still presents difficulties to conventional control design techniques due to the nonlinearities and uncertain parametric values (e.g., A_1, A_3, A_5, A_7) for the lift force. For these reasons, the aeroelastic oscillator has been selected to illustrate the properties of the hybrid controller.

4.1.2 Open Loop Dynamics

The nonlinearities in the open-loop dynamics of the aeroelastic oscillator in Equation (4.6) are a function of both mass velocity and incident wind velocity. For low incident wind velocities, the focus of the state trajectories in the phase plane is stable and the plant returns to the origin after exogenous disturbances. However, for higher wind velocities, the system tends to oscillate in a stable limit cycle. If the wind velocity is further increased, state trajectories in the phase plane are characterized by two stable limit cycles separated by an unstable limit cycle. Since the aeroelastic oscillator either returns to the origin or exhibits a stable limit cycle in face of disturbances for any value of incident wind, it is globally open-loop stable and a feedback loop is not required to provide nominal (bounded input / bounded output) stability.

4.1.3 Reference Model

As discussed in Section 3.3, the hybrid control law is designed to cause the plant state trajectory to follow a reference trajectory generated by a reference model. This reference model has a significant influence on the performance of the closed-loop system,

since by definition it represents the desired trajectory of the controlled vehicle states. As a result, if an unsatisfactory reference model is selected, the vehicle acting under the hybrid control law will also be unsatisfactory. Furthermore, if the reference model demands unrealistic state trajectories (e.g., reference trajectories that are chosen without regard to the limitations of the actual plant dynamics), control saturation leading to inadequate performance or even instabilities (in the general case) can occur. For these reasons, the reference model must be selected to yield satisfactory dynamics within the limitations of the vehicle or plant as required by specifications.

For the aeroelastic oscillator, the reference model was chosen to be the linear closed-loop system that results from applying an optimal linear quadratic control design to the aeroelastic oscillator dynamics linearized about the origin. The quadratic cost functional weights states and control equally. Thus, the objective of the hybrid control law is to force the true nonlinear model to behave identically to the linear reference model. Although not a requirement, a linear reference model is often used to achieve specifications (objectives) that have been stated in terms of natural frequency and damping ratio requirements. The reference model for the aeroelastic oscillator has been designed with a natural frequency of 1.12 radians per second and a damping ratio of 0.76.

4.1.4 Application of Hybrid Controller

To aid in the design, simulation, and analysis of the hybrid learning system, a custom-built software package developed at Draper Laboratory and coined "NetSim" was used. NetSim is a general-purpose simulation and design package that enables a variety of connectionist learning control systems to be developed interactively (Alexander, *et al.* (1991)). Through a graphical interface, pre-compiled code modules are connected in a block diagrammatic format to form the desired system. For dynamic systems, typical modules include plants, transforms (e.g., signal modifiers such as delays or switches), summing and gain objects and even dynamic compensators. NetSim also contains design

tools that allow the user to create connectionist networks by graphically specifying the network nodes and architecture. All of the code modules are automatically linked together at run time, resulting in a complete system in which the outputs can be viewed on-line while the simulation is in progress.

The closed-loop simulation for the aeroelastic oscillator system uses four main modules as illustrated in the block diagram in Figure 4.3. This figure is a screen dump of the actual simulation window. The main modules include the reference model, hybrid controller, aeroelastic oscillator, and linear-Gaussian network. In addition to these main components, supporting operators are needed to modify the signals passed between the main modules to deliver the expected variables in the proper time sequence. The arrows between modules represent exchanges of variables, and the number in the lower left corner of each block dictates the order of execution at each time step. Modules called more than once per time step are shown with multiple sequence numbers.

Each module in Figure 4.3 performs a specific function in modeling the closed-loop dynamic system. The first module in the sequence is *Random*. *Random* outputs a randomly generated commanded position at the current time k. This command is held constant for a user specified amount of time. Once that time has elapsed, a new command is issued. *AO Reference* outputs the desired (model) state trajectory of the aeroelastic oscillator for the given command. The reference trajectory is generated using a discrete version of the optimal linear design as discussed above.

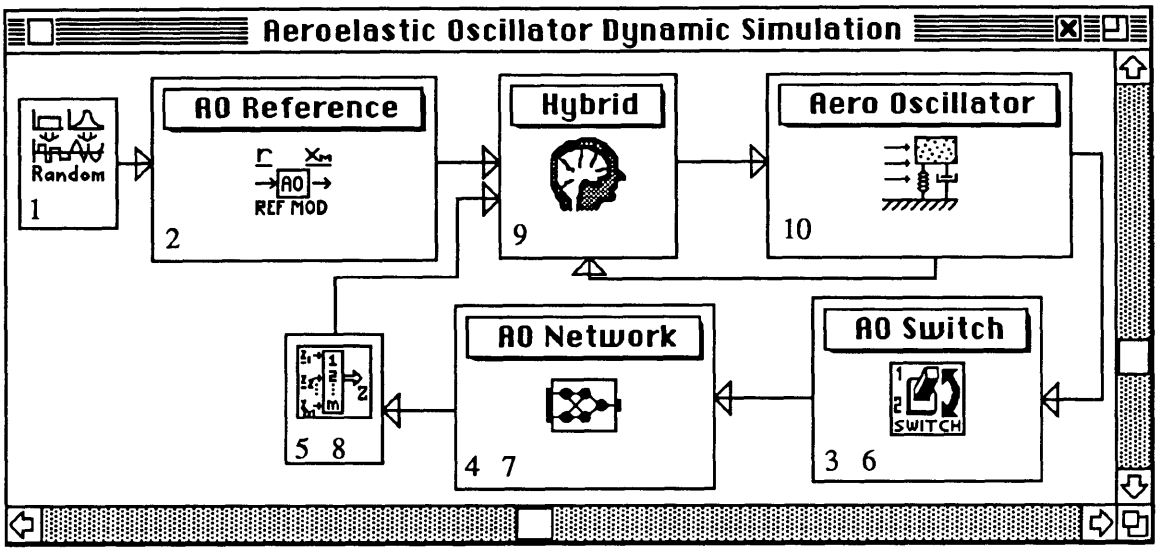

Figure 4.3 Block Diagram of the Aeroelastic Oscillator System

The *AO Switch* module supplies the network with the state and control at the appropriate time. It also sends a flag to the network to insure that learning only occurs with states and control at consistent times (e.g., learning occurs when the state, control, and desired output are all at the same time instant). *AO Network* is a linear-Gaussian network that serves as the learning system in the hybrid controller. The *Multiplexor* (shown with sequence numbers 5 and 8) gathers the outputs from the network that are needed for implementing the hybrid control law. *Hybrid* calculates the control signal based on the hybrid control law developed in Section 3.3. This control signal is passed to the *Aero Oscillator* module. The *Aero Oscillator* module contains the continuous nonlinear equations-of-motion of the plant. These equations are integrated using either an Euler or 4^{th} order Runge-Kutte technique. The type and rate of integration, as well as plant parameters and initial conditions, are selected by the user. Table 4.1 summarizes the output of each module for one time step.

Table 4.1 Module Execution Summary

Sequence #	Module	Module Output
1	Random	$\mathrm{r}(k)$
2	AO Reference	$\mathbf{x}_m(k+1) = \mathbf{\Phi}_m \mathbf{x}_m(k) + \mathbf{\Gamma}_m \mathrm{r}(k)$
3	Switch	$\mathbf{x}(k), \mathrm{u}(k-1)$; Don' t Learn Flag
4	AO Network	$\mathbf{f}_{net}(\mathbf{x}(k), \mathrm{u}(k-1))$; $\left.\frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right\|_{\mathbf{x}(k),\mathrm{u}(k-1)}$
5	Multiplexor	$\mathbf{f}_{net}(\mathbf{x}(k), \mathrm{u}(k-1))$; $\left.\frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right\|_{\mathbf{x}(k),\mathrm{u}(k-1)}$
6	Switch	$\mathbf{x}(k-1), \mathrm{u}(k-1)$; Learn Flag
7	AO Network	$\mathbf{f}_{net}(\mathbf{x}(k-1), \mathrm{u}(\mathrm{k}-1))$
8	Multiplexor	$\mathbf{f}_{net}(\mathbf{x}(k), \mathrm{u}(k-1))$, $\left.\frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right\|_{\mathbf{x}(k),\mathrm{u}(k-1)}$, $\mathbf{f}_{net}(\mathbf{x}(k-1), \mathrm{u}(k-1))$
9	Hybrid	$\mathrm{u}(k) = -\left[\mathbf{\Gamma} + \frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right]^{+} \mathbf{K}\mathbf{e}(k) + \left[\mathbf{\Gamma} + \frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right]^{+} [\mathbf{\Phi}_m - \mathbf{\Phi}]\mathbf{x}(k) + \left[\mathbf{\Gamma} + \frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right]^{+} \mathbf{\Gamma}_m \mathrm{r}(k) - \left[\mathbf{\Gamma} + \frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right]^{+} [\hat{\mathbf{h}} + \hat{\mathbf{d}}] - \left[\mathbf{\Gamma} + \frac{\partial \mathbf{f}_{net}}{\partial \mathrm{u}}\right]^{+} \left[\mathbf{f}_{net}\{\mathbf{x}(k), \mathrm{u}(k-1)\} - \left.\frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}}\right\|_{\mathbf{x}(k),\mathrm{u}(k-1)} \mathbf{u}(\mathbf{k}-1)\right]$
10	Aero Oscillator	$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & (nA_1 U - 2b) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -\frac{nA_3}{U} x_2^{\,3} + \frac{nA_5}{U^3} x_2^{\,5} - \frac{nA_7}{U^7} x_2^{\,7} \end{bmatrix}$

4.1.5 Aeroelastic Oscillator Experiment 1

In experiment 1, a selected reference trajectory was repeated continuously in order to learn the state dependent, previously unknown dynamics $\mathbf{f}_{net}$. The control objective was simply the regulation of both states about the origin given an initial position of -1 and velocity of 0.5. By using the geometry and velocity parameters for a particular incident wind velocity found in Parkinson & Smith (1963), the equations-of-motion used in experiment 1 become:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 1.2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ -26.1x_2^3 + 127.3x_2^5 - 158.9x_2^7 \end{bmatrix} \tag{4.7}$$

The nonlinear terms Equation (4.7) were not supplied to the control system and represent the unknown dynamics in Equation (3.21).

Figures 4.4 and 4.5 illustrate the reference trajectories for position and velocity (based on the linear model described above) for the selected initial conditions and command. These reference trajectories represent the desired states at each time step, and any deviation from the reference by the actual states can be considered an error. The position and velocity trajectories of the nonlinear aeroelastic oscillator controlled by TDC alone (TDC Position / Velocity) are also shown in Figures 4.4 and 4.5 and are almost indistinguishable from the reference. In this case, the TDC controlled trajectories were produced by integrating the aeroelastic oscillator equations-of-motion at 200 Hertz and generating a control signal at that same rate. Moreover, there was no noise in the observed state and control values used by TDC. Combining these facts, it is not surprising that the TDC controller does extremely well in generating a control law that drives the plant along the reference trajectory. Indeed, because of the extremely small time step, the unknown dynamics observed at the previous time provide an accurate estimate of the unknown dynamics at the current time that is required by the TDC control law. Also plotted in Figures 4.4 and 4.5 are the trajectories generated using the constant gains of the linear controller used to form the linear reference model and applied to the actual nonlinear

aeroelastic oscillator (labeled Linear Position / Velocity). Errors between trajectories under this linear control and the reference trajectory are due primarily to the nonlinear aerodynamic lift force. These plots show the degree of performance improvement (relative to a linear feedback law) that is possible with an adaptive controller operating under ideal conditions.

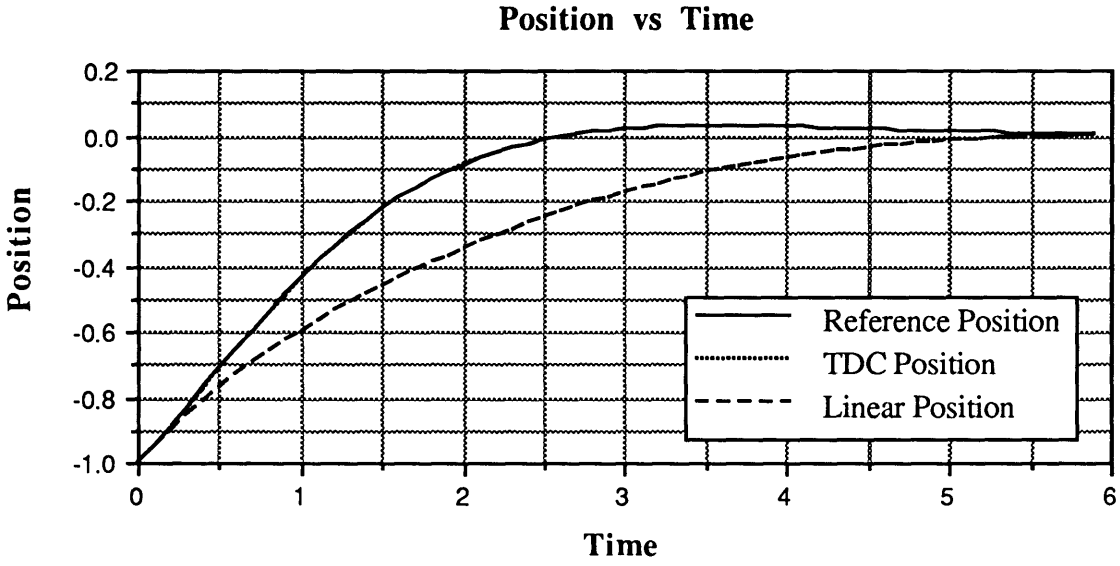

Figure 4.4 Position trajectories for the reference model, linear control law, and TDC at 200 Hertz controller rate.

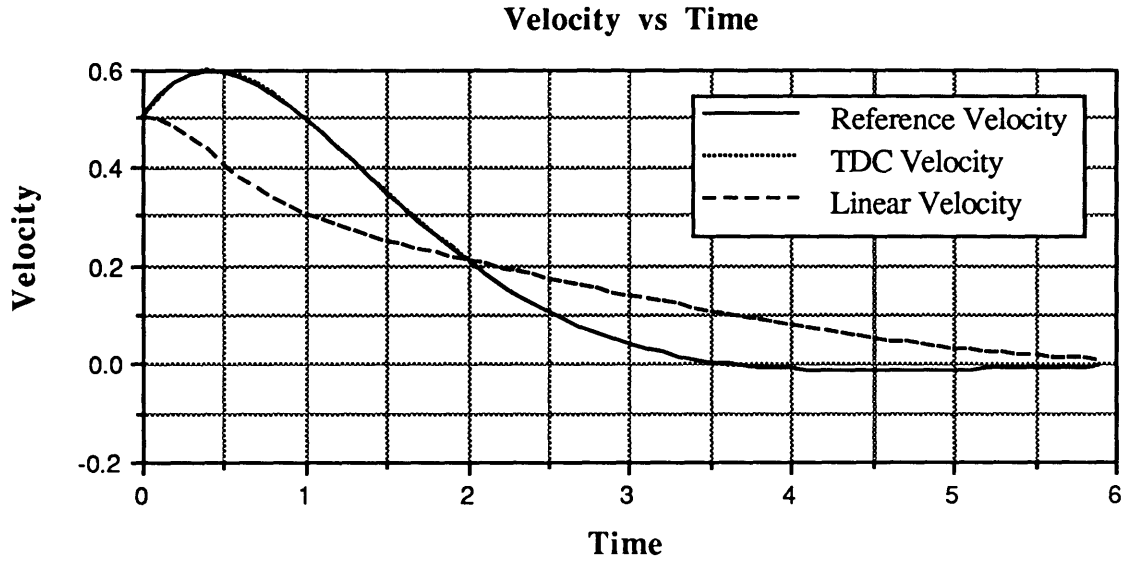

Figure 4.5 Velocity trajectories for the reference model, linear control law, and TDC at 200 Hertz controller rate.

Since the sensing and computational requirements associated with generating state information for the aeroelastic oscillator at the 200 Hertz integration rate may be unrealistic, the controller is slowed to calculate the control signal at a more moderate rate. For this experiment, the control was generated at 10 hertz. In order to produce unknown dynamics that are a function of control as well as state, an unknown external force equal to three times the control force was added to the unknown dynamics. In other words, the control form in Equation (4.7) was modified from the assumed known value

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} F$$

to the applied value

$$\left[\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} \right] F$$

where the added term is not known by the controller. A relatively large force error was used to highlight the ability of the hybrid control system to reduce large uncertainties.

Consistent with the hybrid control law developed in Section 3.3, the learning

system used a spatially localized network with 32 linear-Gaussian nodes. For training the network, a learning rate of 1 was used with the spatial decay of all the nodes fixed at 2. These values were selected based on the known mapping of the nonlinearities, the size of the input space (i.e., range of all possible position, velocity, and control combinations), and to a certain extent on trial and error. Initial values for the slopes and biases were set to zero while the Gaussian centers were placed randomly in the unit cube formed by scaling the state and control inputs. Figures 4.6 and 4.7 illustrate the hybrid controlled states for the first learning trial compared to the TDC controlled states and reference model. Since the slopes and biases of the learning system are initialized to zero, the learning system does not impact the states at start-up, and all of the unknown dynamics are incorporated into the TDC adaptive component. However, after a short time (within the first trial), the learning system begins to build a mapping of the unknown dynamics. This mapping is used to eliminate the delay associated with the unknown dynamics estimate in TDC and to improve the estimate of the local linearized behavior (i.e., using the derivative information as discussed in Section 3.3 to reduce model uncertainty). These features can be directly related to the improved performance seen in the state tracking of the reference trajectory.

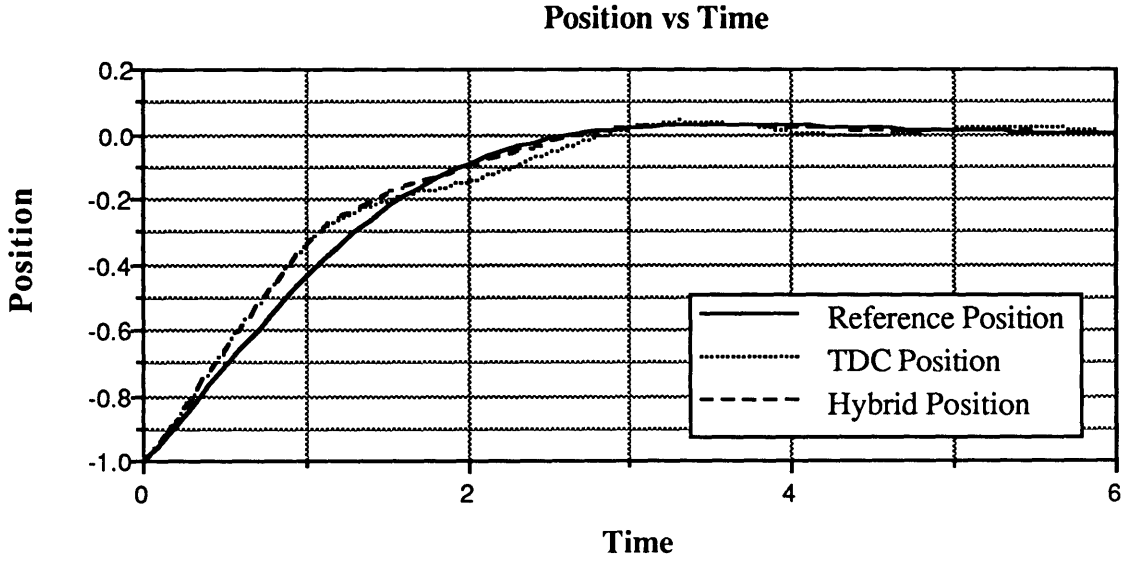

Figure 4.6 Position trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, first learning trial.

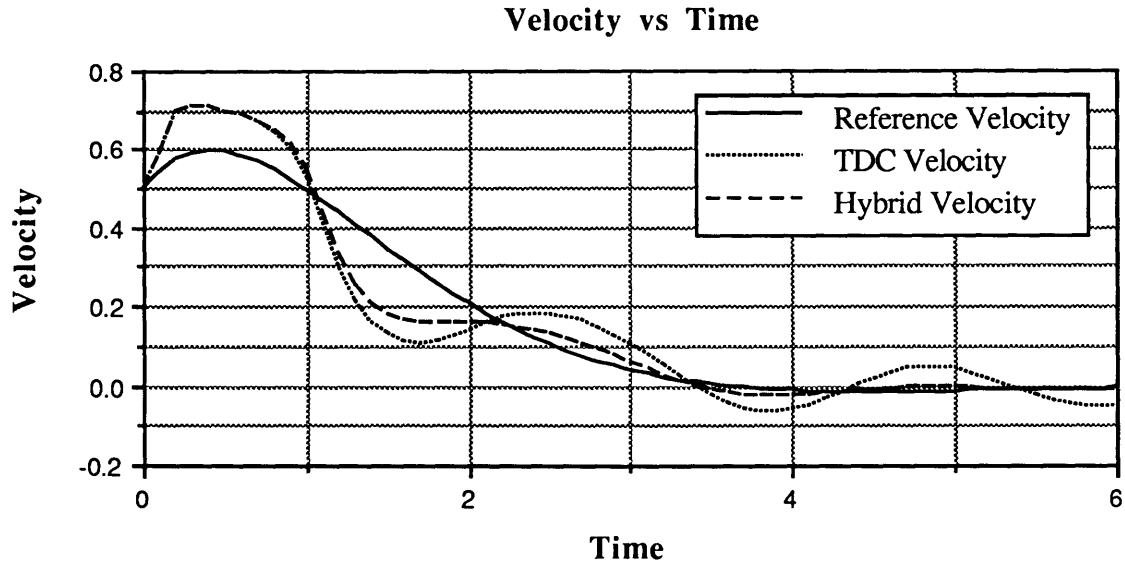

Figure 4.7 Velocity trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, first learning trial.

After the trajectory is repeated 10 times, the learning system has built a mapping of the previously unknown dynamics as a function of the state and control along that trajectory. Figure 4.8 compares the estimate of the unknown dynamics used by TDC to

that of the hybrid controller generated from the learned mapping (after 10 trials). Since the mapping used to generate these points represents a static function, the unknown dynamics can simply be looked up as a function of the current state and control. This can be contrasted with TDC which uses an estimate of the unknown dynamics based on the state and control at the previous time.

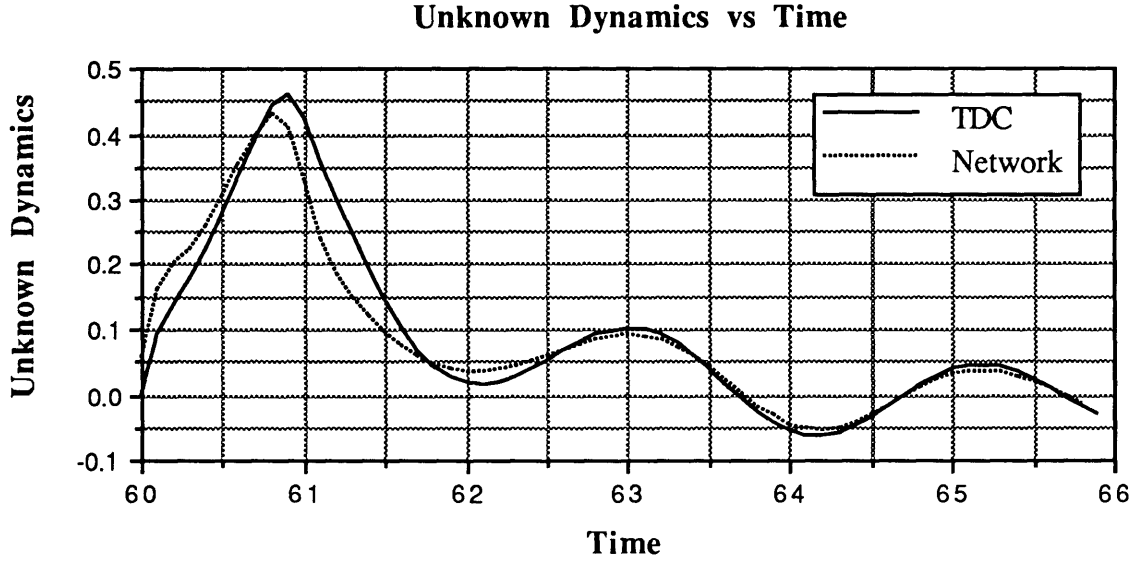

Figure 4.8 Unknown dynamics estimate from network and TDC after 10 trials.

As discussed in Section 3.3, the hybrid controller uses the output of the network ($\mathbf{f}_{net}$) as well as the derivative of the network output with respect to the control ($\partial \mathbf{f}_{net}/\partial u$) to formulate the control law. This derivative information provides local improvements to the linear control weighting vector, Γ. Since the truth model for the aeroelastic oscillator is known, it is possible to analyze the accuracy of the derivative information. For example, the partial of the unknown dynamics with respect to the control force is simply, in continuous time, the constant three (due to the added external control force). When converted to discrete time, this value is 0.3182. After 10 trials, the networks mean value of the $\partial \mathbf{f}_{net}/\partial u$ is 0.2285. Although it has not yet learned the correct value, it nonetheless provides some improvement to the control weighing matrix.

Figures 4.9 and 4.10 illustrate the state trajectories controlled by the TDC controller and the hybrid controller after 10 trials. Clearly, the hybrid controller uses experientially gained knowledge to improve the tracking of the reference states.

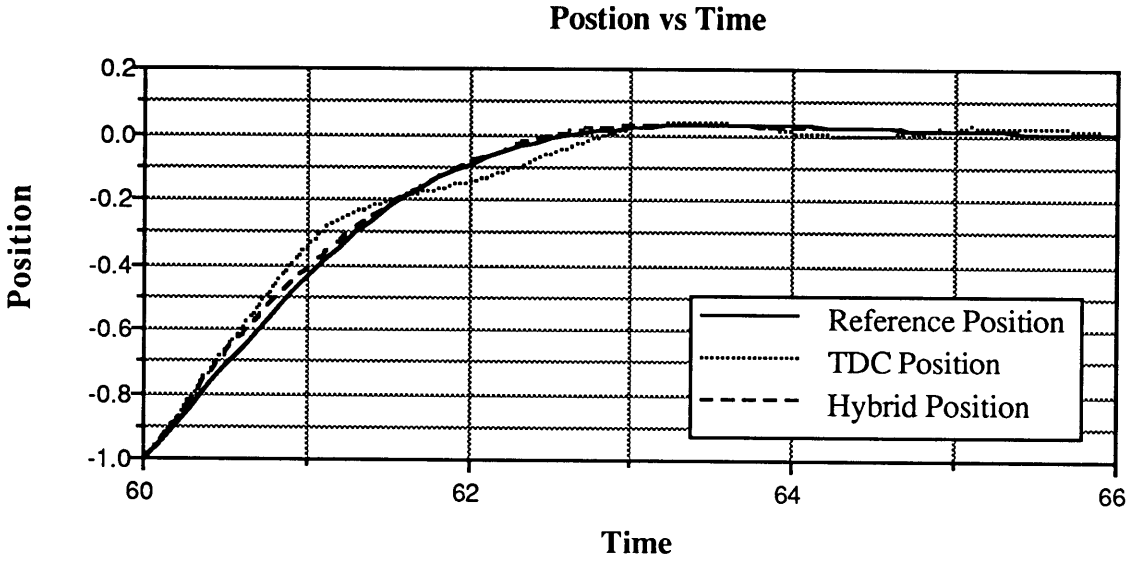

Figure 4.9 Position trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

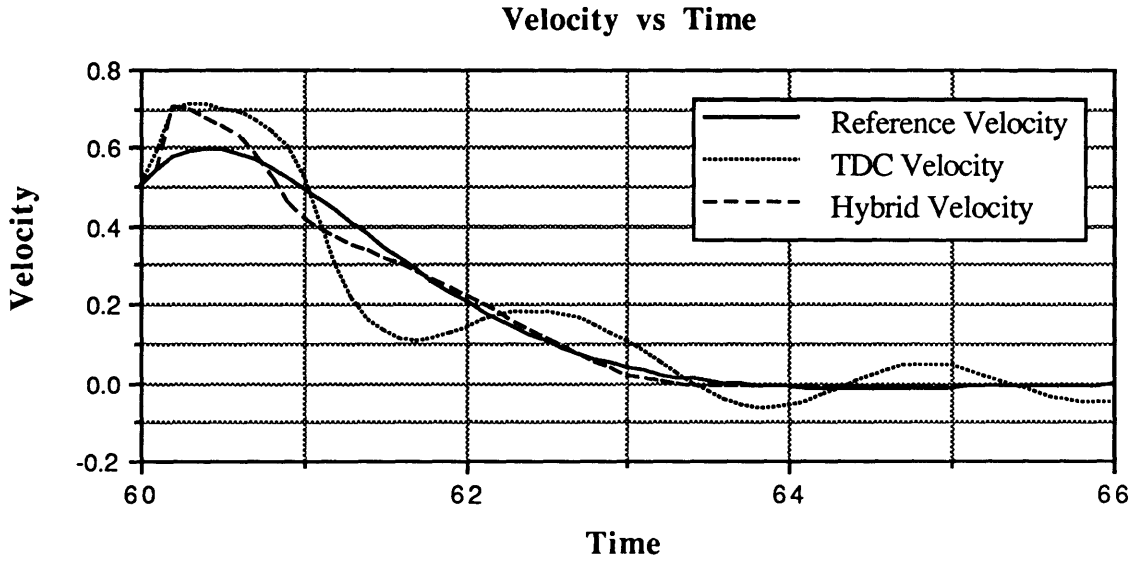

Figure 4.10 Velocity trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

This experiment shows that the hybrid controller has the ability to improve the controlled performance of the aeroelastic oscillator when a specific trajectory is repeated numerous times. This improved performance is realized by exploiting a learned functional mapping of the previously unknown model dynamics to improve the control law. The next experiment illustrates the ability to synthesize a mapping over a much larger input space, using randomly generated state trajectories.

4.1.6 Aeroelastic Oscillator Experiment 2

In experiment 2, the desired trajectory is selected in a random fashion in order to map the unknown dynamics over a much larger region of the state space than the single trajectory in experiment 1. By commanding a random position between –1 and 1, a large portion of the state space along with the associated controls is visited and subsequently mapped. As in experiment 1, the aeroelastic oscillator was integrated at 200 Hertz and the control signal issued once every 20 integrations (10 hertz). For this experiment, a spatially localized learning system with 99 linear-Gaussian nodes was used. The spatial decay for

each node was fixed to 1 and the initialization was the same as for experiment 1. The number of nodes, spatial decay, and other parameters were again selected based on the expected nonlinearities, size of the input space and trial and error.

Figure 4.11(a) shows the mapping synthesized by the learning system as a function of velocity and control. Learning was based on following the randomly generated reference trajectory for 60 seconds (10 trials). This mapping is compared to the nonlinear terms and extraneous control of the aeroelastic oscillator truth model shown in Figure 4.11(b). Comparing the two plots, the slope in the control direction (force) for the network mapping is nearly constant with a mean of 0.3120 and standard deviation of 0.0264 whereas the actual slope is 0.3182 (in the discrete time model). Moreover, the mappings in the velocity direction appear very similar. Hence, the network has synthesized the previously unknown dynamics of the system. (Note: the current version of the software does not allow a direct error surface plot to be generalized.)

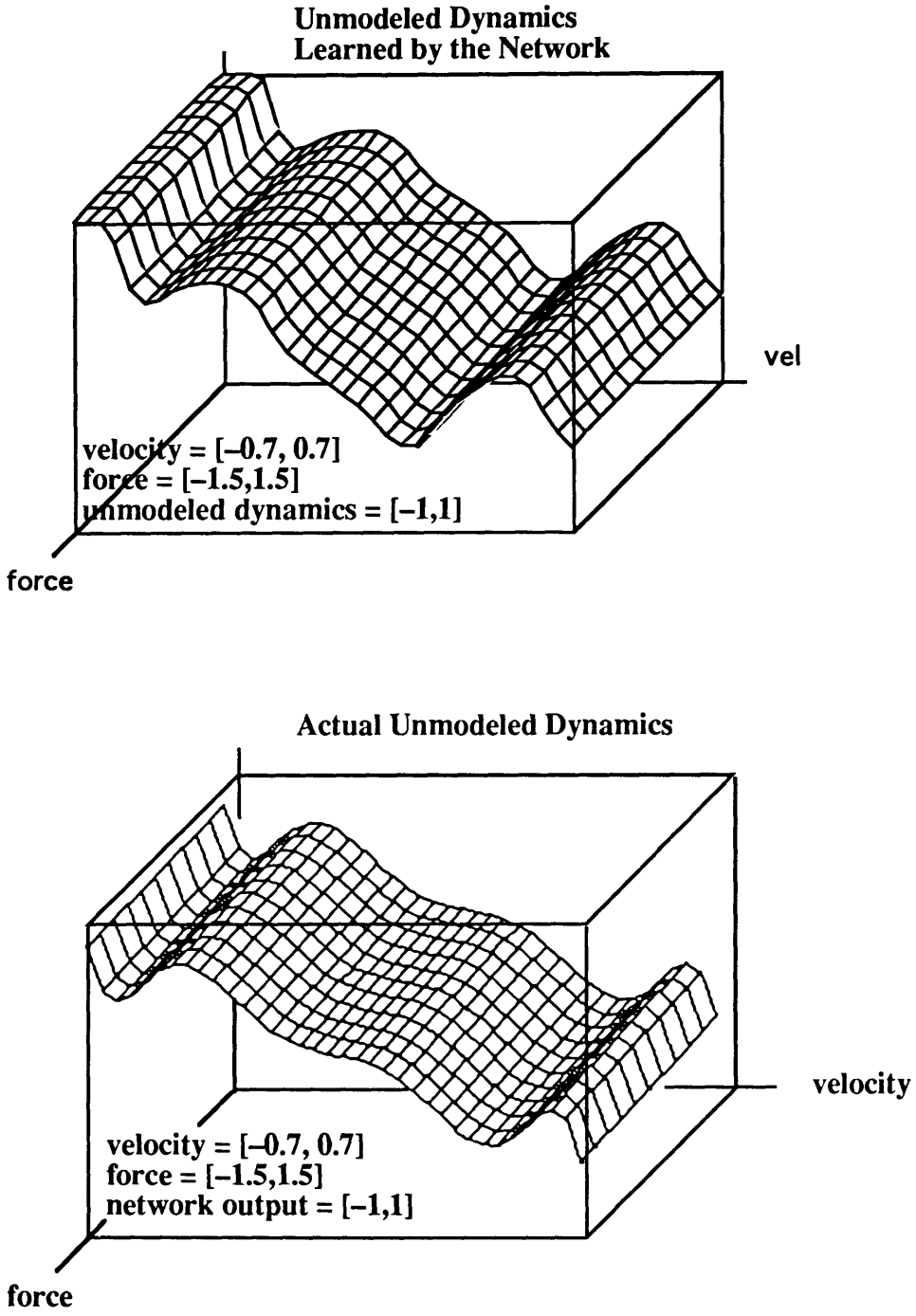

Figure 4.11 (a) Network Mapping of Unknown Dynamics (b) Actual Unknown Dynamics.

Figures 4.12 and 4.13 illustrate the position and velocity trajectories for the TDC and hybrid controlled states after 30 seconds of simulation. As predicted by the relatively accurate mapping of the unknown dynamics, the position and velocity show improved

performance for the hybrid controlled aeroelastic oscillator over that of TDC.

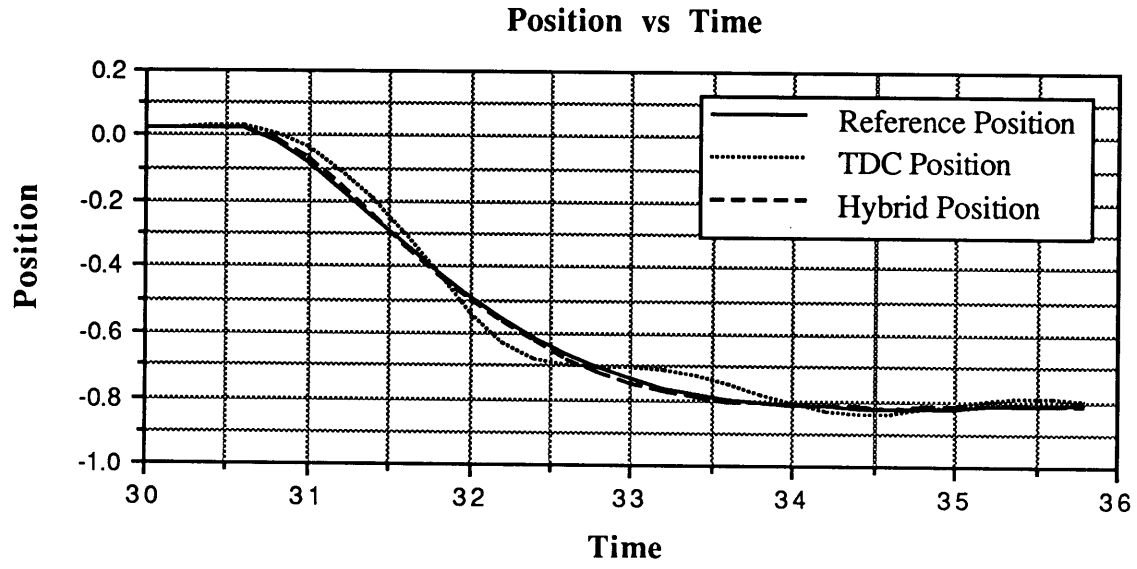

Figure 4.12 Position trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

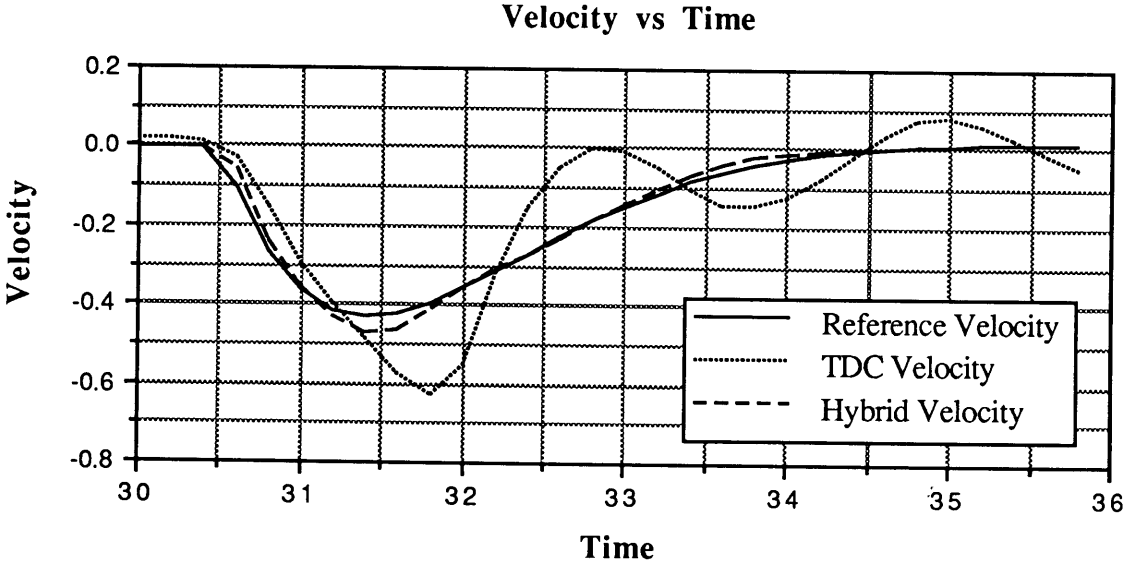

Figure 4.13 Velocity trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

4.2 HIGH PERFORMANCE AIRCRAFT MODEL

4.2.1 Aircraft Description

The high performance aircraft model that is used to illustrate the concept of learning enhanced flight control was developed by NASA to provide the aeronautical community with a common focus for research in flight control theory and design. This model is also being used to serve as a basis for the 1992 AIAA Control Design Challenge (Duke (1992)). A complete description of this generic, high performance, state-of-the-art aircraft model is found in Brumbaugh (1991). The following summarized the major characteristics of the aircraft model as well as its critical components.

The NASA model is the basis for the simulation of a high-performance, supersonic vehicle representative of modern fighter and attack aircraft. This model supports virtually all missions in nonterminal flight phases. These missions include flight phases that are normally accomplished using gradual maneuvers such as climb, cruise, or loiter as well as phases that require rapid maneuvering, precision tracking, or precise flight-path control (e.g., air-to-air combat, weapon delivery, or terrain following). The aircraft model includes full-envelope, nonlinear aerodynamics in addition to a full-envelope, nonlinear thrust model. An illustration of the basic configuration of the aircraft is shown below in Figure 4.14. Significant features of this aircraft configuration include a single vertical tail with rudder surface, a horizontal stabilator capable of symmetric and differential movement, and conventional trailing edge ailerons.

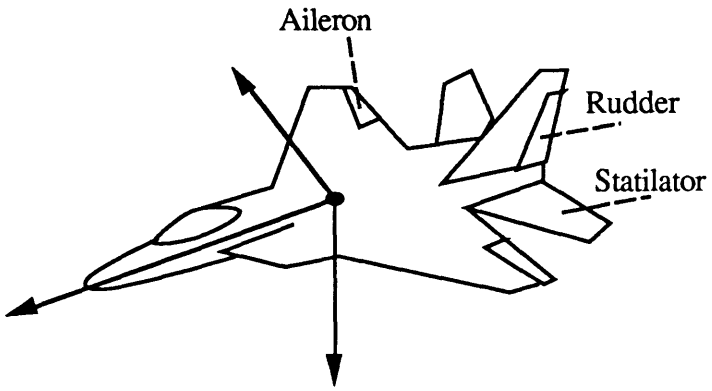

Figure 4.14 NASA High Performance Aircraft Model

The basic geometry and mass properties of the aircraft are summarized below in Table 4.2.

Table 4.2 Basic Aircraft Geometry and Mass Properties

Aircraft Geometry and Mass Properties	
Wing Area	608.00 ft^2
Wing Span	42.80 ft
Mean Chord	15.95 ft
Weight	45000.00 lb

To aide in the design and development of a competent flight control law, the model can be easily broken into separate components, each performing a specific function. The major components of the aircraft model are as follows: aerodynamics, propulsion, actuator dynamics, and equations-of-motion. Also included with the model is the standard atmosphere component, an environmental model, and the integration component that is used to simulate the aircraft in software. Of course, one element that is not in this list is the flight control law, which is to be determined by the designer. The function of each of the major components, as well as a brief discussion of its origins, are presented in the following paragraphs.

As alluded to previously, the NASA aircraft simulation contains a nonlinear, full-

envelope *aerodynamic model.* The primary function of this component is the calculation of aerodynamic forces and moments generated by the aircraft throughout its flight regime. In general, the aerodynamic forces and moments are complicated, nonlinear functions of many variables. The approach taken by the NASA model in calculating the complex force values is based on modeling the force terms as the product of dynamic pressure, a reference area (wing area), and an appropriate dimensionless aerodynamic coefficient. Similarly, the aerodynamics moment term is modeled as the product of dynamic pressure, a reference area, a dimensionless aerodynamic coefficient, and a reference length (mean chord). The aerodynamic coefficients are primarily functions of Mach number, angle-of-attack, and sideslip angle. The NASA aircraft model acquires coefficient values from multidimensional data tables or from direct calculation. The coefficients contained in the tabular data have been generated through a combination of wind-tunnel tests and computer programs that numerically integrate the theoretical aerodynamic pressure over the surface of the aircraft. For the tabular data, linear interpolation is employed to obtain intermediate values.

Vehicle thrust is generated by the *propulsion model.* Twin afterburning turbofan engines, each capable of generating 32,000 pounds of thrust, deliver power to the aircraft. Each engine thrust vector acts along the aircraft x-body axis at a point 10 feet behind the center of gravity and 4 feet laterally from the centerline. Engine dynamics are modeled by separating the powerplant into two separate sections. The first section, the engine core, is modeled as a first-order, closed-loop system that outputs thrust for a given throttle input. Moreover, rate limits that simulates spool-up effects and a gain scheduler that models changes in performance due to Mach number and altitude changes have been added to provide realism to the closed-loop system. Gains are obtained from tabular data and a linear interpolation routine based on Mach number and altitude. A second section, the afterburner, is modeled with similar first-order dynamics but has the added features of a rate limiter and sequencing logic to model fuel pump and pressure regulator effects. Together, these components comprise the full-envelope, nonlinear thrust model.

Atmospheric parameters required by the aircraft simulation are computed by the *standard atmosphere model.* For a given altitude, values for acceleration due to gravity, speed of sound, temperature, and other essential parameters are generated from tables based on the U.S. Standard Atmosphere of 1962. Linear interpolation is used between elements of the table.

The *actuator dynamics model* is a first-order system that outputs surface position for a given surface command. Furthermore, rate and position limits are included in the system. All actuators are considered to be identical.

The dynamics of the aircraft are simulated using the *equations-of-motion module.* The nonlinear equations-of-motion are derived from the general six-degree-of-freedom relations for a rigid aircraft. Beyond the rigid body assumption, it is also assumed that the vehicle is traveling with nonzero forward motion in an atmosphere that is stationary with respect to an Earth-fixed reference frame. The nonzero forward motion assumption mandates that only nonterminal flight phases be simulated by this model. Since each degree of freedom requires two state variables (the basic variable and its rate), a total of twelve first-order differential equations are required to completely describe the motion of the aircraft. Table 4.3 lists each state variable and its symbol. Note that if speed is assumed to be relatively constant, then angle-of-attack and sideslip angle may be supplemented for the y and z body axis velocity vector projections respectively. A detailed derivation of these equations-of-motion can be found in Etkin (1982) or Roskam (1979).

The state variables are propagated in time via the *integration module.* This module uses a second-order Runge-Kutta midpoint algorithm to arrive at a new state based on the state and control at the previous time. Running at 50 Hertz, this integration technique has been found to provide a balanced tradeoff between numerical stability and processing speed.

Table 4.3 The twelve aircraft state variables and symbols

State Variables and Symbols	
Displacement North	x
Displacement East	y
Altitude	h
Velocity	u
Angle of Attack	α
Side Slip Angle	β
Pitch Angle	θ
Roll Angle	ϕ
Yaw Angle	ψ
Roll Rate	p
Pitch Rate	q
Yaw Rate	r

An auxiliary component of the aircraft model that is not critical to the simulation but is invaluable to the control law designer is the *observation model.* The function of the observation model is to output a large class of aircraft measurements. States, state derivatives, accelerations, airdata parameters, force parameters, and a multitude of other important data are furnished for observation. Of these parameters, state information as well as vehicle body axis rates make up the set of parameters that have been traditionally used for feedback flight control.

By linking the previously described modules, a realistic, highly complex, nonlinear aircraft model that poses formidable challenges to the flight control designer is assembled. The high performance aircraft computer model received from NASA was written in the FORTRAN programming language. In order to produce a model compatible with the NetSim simulation and design package discussed in Section 4.1.4, this FORTRAN version

was transposed into the C programming language by the author.

4.2.2 General Aircraft Characteristics

The flight envelope of the NASA aircraft model is characteristic of a high performance fighter aircraft (Brumbaugh (1991)). Figure 4.15 below illustrates approximate bounds of aircraft operation in terms of altitude and Mach number.

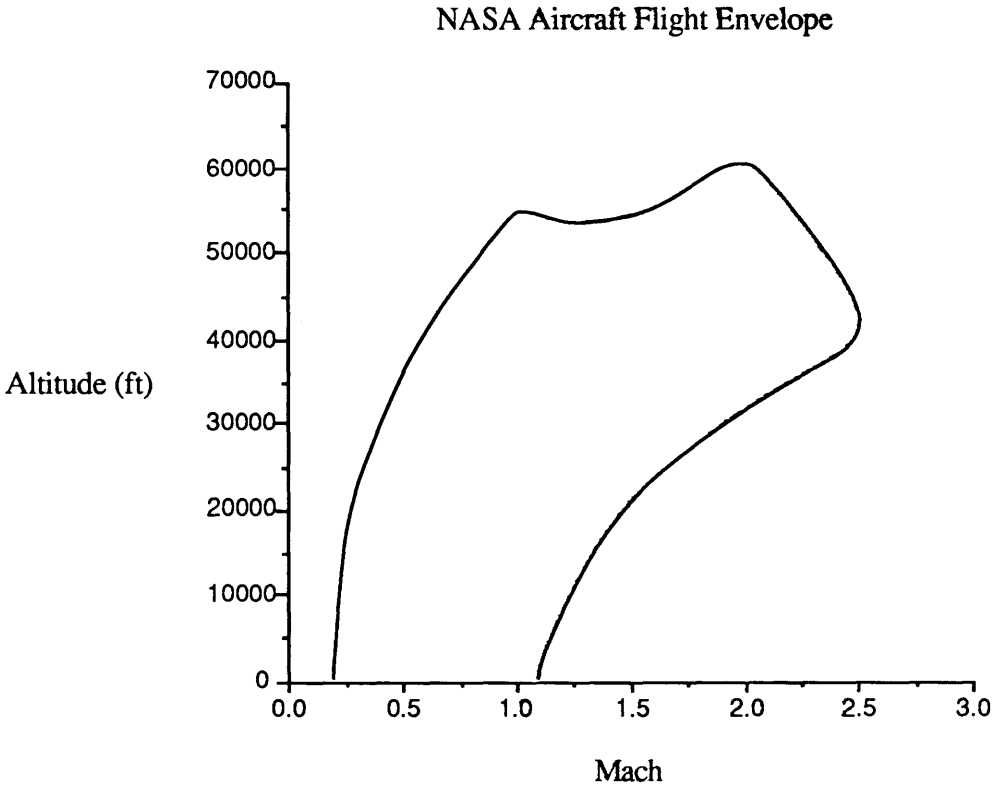

Figure 4.15 1-g Aircraft Flight Envelope.

To examine the nonlinear dynamics of a complex aircraft model, the equations-of-motion are frequently linearized about various operating conditions. By linearizing the dynamics at a sufficient number of operating conditions within the envelope, an improved overall picture of the actual nonlinear dynamics can be gained. Generally, operating conditions near the boundary of the envelope, as well as a few centrally located points, are

selected. The most common technique in obtaining linearized dynamics is by invoking the small perturbation theory based on a Taylor series expansion. This theory uses infinitesimal perturbations from an equilibrium or trimmed steady-state reference condition to predict aircraft response to perturbations that are not infinitesimal. A trim condition is classically defined as a constant velocity and altitude state with control surfaces and throttles set to maintain this condition. If it is assumed that all perturbations and their derivatives are small, the quadratic and higher order products of the perturbations will be negligible compared to the first-order quantities. In other words, a linear model is obtained by deriving relations of small deviations of all state and control variables about a steady-state equilibrium condition and retaining linear terms while ignoring quadratic and higher-order terms. A detailed version of the following short derivation of this theory can be found in (Athans (1990)).

Let $\mathbf{x}(t)$ and $\mathbf{u}(t)$ represent state and control variables, respectively, with

$$\mathbf{x}(t) \in \Re^n \tag{4.8}$$

$$\mathbf{u}(t) \in \Re^m \tag{4.9}$$

The nonlinear state dynamics in continuous time are given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}\{\mathbf{x}(t), \mathbf{u}(t)\} \tag{4.10}$$

The reference state and control values representing an equilibrium condition (e.g., $\dot{\mathbf{x}}(t) = 0$) for the nonlinear equation are denoted by a subscript zero.

$$0 = \mathbf{f}\{\mathbf{x}_0, \mathbf{u}_0\} \tag{4.11}$$

Small perturbations about the equilibrium condition are denoted with a lower case delta:

$$\mathbf{x}(t) = \mathbf{x}_0 + \delta\mathbf{x}(t) \tag{4.12}$$

$$\dot{\mathbf{x}}(t) = \delta\dot{\mathbf{x}}(t) \tag{4.13}$$

$$\mathbf{u}(t) = \mathbf{u}_0 + \delta\mathbf{u}(t) \tag{4.14}$$

Expanding the state dynamics in a Taylor series about the equilibrium condition and solving

for $\delta\dot{\mathbf{x}}(t)$ while retaining linear terms and disregarding higher-order terms yields the following state perturbation dynamics:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}_0\delta\mathbf{x}(t) + \mathbf{B}_0\delta\mathbf{u}(t) \tag{4.15}$$

where

$$(\mathbf{A}_0)_{ij} = \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j(t)}\right|_{\mathbf{x}(t)=\mathbf{x}_0,\mathbf{u}(t)=\mathbf{u}_0} \tag{4.16}$$

$$(\mathbf{B}_0)_{ij} = \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{u}_j(t)}\right|_{\mathbf{x}(t)=\mathbf{x}_0,\mathbf{u}(t)=\mathbf{u}_0} \tag{4.17}$$

$\mathbf{A}_0$ and $\mathbf{B}_0$ are the Jacobian matrices of the Taylor series expansion of $\mathbf{f}\{\mathbf{x}(t),\mathbf{u}(t)\}$ centered about $\mathbf{x}_0$ and $\mathbf{u}_0$. Although the Jacobian matrices can occasionally be found in closed form for relatively simple systems, more complex systems often require numerical differentiation. For this reason, numerical differentiation is used for the aircraft model to calculate the Jacobian matrices.

Using the small perturbation theory to linearize the equations-of-motion about an equilibrium condition can provide insight into the local behavior of the nonlinear aircraft dynamics in terms of stability, transient responses, and other system characteristics. However, this theory is not without its limitations. Large numbers of linear models must be computed to characterize the dynamics in highly nonlinear regions of the flight envelope. Moreover, the small perturbation theory is ill-suited to handle phases of flight where large deviations from the nominal trim condition are encountered (i.e., high angle-of-attack flight or spinning maneuvers).

Linearizing the equations-of-motion of the NASA aircraft has revealed that the longitudinal dynamics are only lightly coupled with the lateral dynamics at the majority of flight conditions. Moreover, control of the longitudinal or pitching motion is dominated by symmetric movement of the horizontal tail and engine thrust whereas the rolling and yawing motions associated with the lateral dynamics are most heavily influenced by the

ailerons and differential movements of the horizontal tail. For this reason, the aircraft flight control design problem can be separated into two distinct problems, each less complex than the whole. The existence of the lightly coupled modes and the ability to decompose the control system design is common to all but the most unconventional aircraft.

The uncoupled, linearized longitudinal dynamics of the aircraft can be described by a total of five coupled linear, time-invariant differential equations that are a function of pitch rate, velocity, angle-of-attack, pitch angle, and altitude. If the linearized equation for the dynamics of the total thrust in the longitudinal direction is added to this set of variables, the state of the aircraft for longitudinal motion is as follows (where T is total thrust):

$$\mathbf{x} = \begin{bmatrix} q & u & a & q & h & T \end{bmatrix}' \tag{4.18}$$

If the dynamics of the inertial altitude and thrust are temporarily neglected, the four remaining differential equations define the traditional natural modes associated with aircraft pitching motion, namely the short period and phugoid modes. The short period mode is characterized by a highly damped, high frequency oscillation. The short period oscillations represent changes in angle-of-attack and pitch angle with near constant trim speed. In contrast, the phugoid mode exhibits very lightly damped, low frequency oscillations when excited. Under the influence of the phugoid mode, the angle-of-attack remains essentially constant while the speed and pitch angle experience changes. This motion represents a continual exchange of kinetic and potential energy of a slowly rising and falling airplane. Table 4.4 contains the natural frequencies (ω_n) as well as the damping ratios (ξ) for the open-loop longitudinal modes (sp = short period, ph = phugoid) of the NASA aircraft model at four equilibrium points (trim conditions) near the subsonic boundary of the flight envelope. Trim condition 5, which is not on the boundary, is included since it will be used as the initial condition for experiments described in Sections 4.2.5 and 4.2.6.

Table 4.4 High performance aircraft longitudinal modes at various altitude and Mach number trim conditions.

Natural Frequency and Damping Ratio: Longitudinal Modes						
Trim Condition	Altitude	Mach	ω_{nsp}	ξ_{sp}	ω_{nph}	ξ_{ph}
1	5000	0.31	1.88	0.58	0.11	0.04
2	5000	0.90	4.68	0.28	**	**
3	35000	0.68	1.92	0.32	0.08	0.10
4	35000	0.90	2.11	0.21	0.02	0.12
5	9800	0.60	2.77	0.52	0.08	0.07
** at this trim condition, the aircraft does not exhibit a phugoid motion						

For purposes of comparison, the values of natural frequency and damping ratio for a high maneuverability aircraft in nonterminal flight phases can be found in the military specification regulation, MIL-F-8785C (1980). This regulation requires the phugoid mode to have a damping ratio greater than 0.04 and the short period damping ratio to be between 0.35 and 1.30. Moreover, the short period must have a natural frequency approximately bounded by 1 and 10 radians per second, depending on load factor and angle-of-attack. Examining Table 4.4 above, the NASA aircraft fails to meet the requirements for longitudinal motion in some areas of the flight envelope. However, through the use of a control system, the aircraft modes can be modified to meet the military specifications. For the hybrid control law, this is accomplished by selecting a reference model that meets these specifications.

4.3.3 Aircraft Reference Model

As discussed in Section 3.3, the reference model generates the desired state trajectory for the hybrid controlled aircraft states. During the process of selecting a reference model, close attention was paid to ensuring that following the reference trajectories did not require unrealistic control actions. Since the rate and position of the

horizontal stabilator is limited, unrealistic demands on control can translate into either rate of position saturation (e.g., an inability to exercise the control that has been calculated by the control law). Control saturation leads to inadequate performance (i.e., fails military or other specifications) and possibly to instabilities.

The reference model for the high performance aircraft was chosen to be the linear closed-loop system that results from applying an optimal linear control design to the open-loop dynamics linearized about a selected trim condition. For the experiments in Section 4.2.5 and 4.2.6, the linearized dynamics at trim condition 5 (see Table 4.4) were used. The state and control weights for the quadratic cost function used by the optimal control law were initially selected using guidelines suggested by Bryson & Ho (1975) and Kwakernaak & Sivan (1972). Trial and error (based on simulations of the linear dynamics) were used to arrive at the final cost function. The natural frequency and damping ratios for the modes of the closed-loop reference system are listed in Table 4.5.

Table 4.5 Reference Model Longitudinal Modes

Natural Frequency and Damping Ratio			
ω_{nsp}	ξ_{sp}	ω_{nph}	ξ_{ph}
1.46	0.96	0.75	0.96

Compared to the open-loop dynamics, the closed-loop reference model has modes that are heavily damped. Moreover, the natural frequency of the phugoid is much higher in the closed-loop system. This reference system meets military specification requirements.

4.2.4 Application Issues

In this section, the application of the hybrid flight control law to the high performance aircraft model is discussed. Figure 4.16 illustrates the block diagram representing the closed-loop simulation of the hybrid controlled aircraft model in the NetSim simulation and design package.

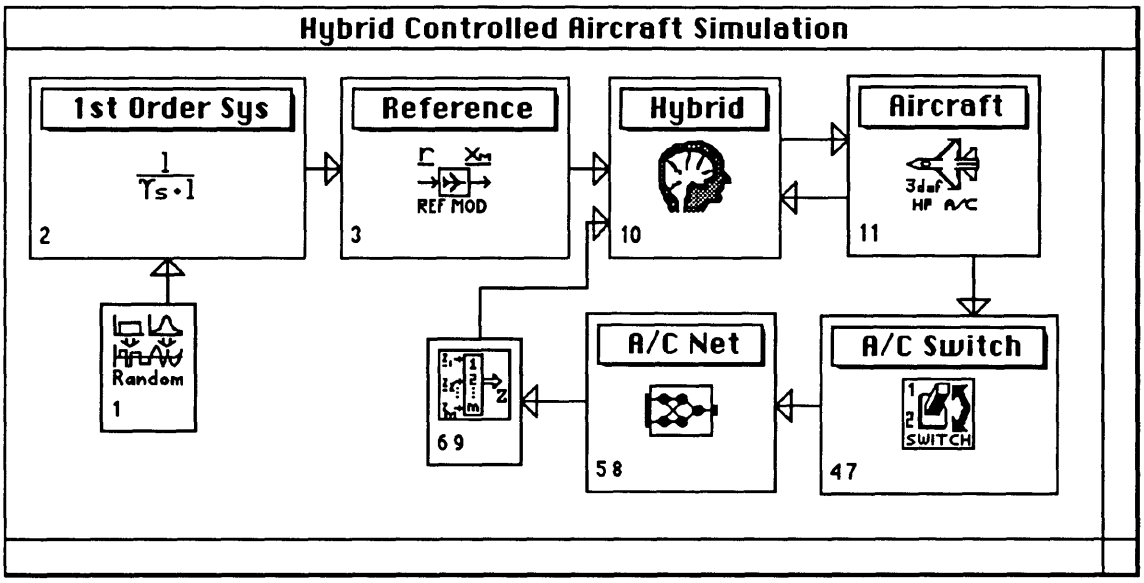

Figure 4.16 Block Diagram of the Hybrid Controlled Aircraft Simulation

The main modules in Figure 4.16 represent the reference model, hybrid controller, high performance aircraft model, and the linear-Gaussian network. The function of the remaining modules is to modify the output signals (represented by the connecting arrows) passed to the main modules. Again, the number in the lower left corner of each block dictates the order of execution at each time step. Modules that are called more than once per time step are shown with multiple sequence numbers. The following paragraph outlines the principal function of each module.

Random is the first module that is executed. It generates randomly selected reference commands for the altitude and velocity of the aircraft within a user-defined operating range. The length of time these commands are held constant before a new set of reference commands is generated is also determined by the user. The commands are supplied to the *1st Order Sys* module. This module processes the reference commands with a user-defined, rate-limited first-order filter. The purpose of this module is to smooth the step commands generated by the random module, effectively outputting a smoothed ramp to a step command. The function of *Reference* is to generate the desired state trajectory that is to be followed by the hybrid control law. The reference model that is used

is discussed in Section 4.2.1. The *A/C Switch* module supplies the state at the current time, as well as the state and control at the previous time step to the network. The switch also sends a flag to the network to ensure that learning only occurs with states and controls that are at consistent times. The linear-Gaussian network in the hybrid control architecture is contained in *A/C Net*. The role of the *Multiplexor* (shown with sequence numbers 6 and 9) is to store the output of the learned mapping for various inputs of state and control required by the hybrid control law. *Hybrid* executes the hybrid control law developed in Section 3.3. The complete high performance aircraft is model is contained in the *Aircraft* module.

4.2.5 High Performance Aircraft Experiment 1

In experiment 1, the aircraft was given random commands for altitude and velocity. More specifically, the random altitude commands were between ±500 feet and the random velocity commands were between ±10 feet per second. As discussed in Section 4.2.4, the commands are filtered by a rate limited, first-order system. The rate limits for altitude and velocity were set to 50 feet per second and 4 feet per second per second, respectively. The filtering and rate limiting is intended to result in a physically feasible reference trajectory. The initial condition for the aircraft was an equilibrium condition at an altitude of 9800 feet and velocity of 539 feet per second (trim condition 5 in Table 4.4). For each new randomly generated command, the aircraft was reinitialized to this same trim condition. By randomly selecting commands, the objective was to generate state trajectories that fully traverse a small region of the aircraft operating envelope.

Similar to the experiments involving the aeroelastic oscillator, the linearized dynamics of the aircraft supplied to the hybrid controller were perturbed from their actual values. The purpose of the perturbations was to increase model uncertainty, a feature the hybrid controller is able to accommodate. The perturbations to the dynamics can be viewed as a situation wherein the flight control system is provided linearized dynamics that

represent a trim condition other than that for which the maneuvers are actually to take place. The intent is to illustrate that the hybrid controller is able to adequately control the aircraft given an inaccurate linear representation, indicating that less accurate *a priori* design information is needed and thereby use of the hybrid controller can effectively reduce design costs.

The learning component used in the hybrid control law was again the spatially localized system developed in Section 3.2. For this case, the network consisted of 8 linear-Gaussian nodes. This relatively small number of nodes was considered to be sufficient due to the modest nonlinearities expected for the specified class of reference trajectories. Of the two largest factors in determining the nonlinearity of the system, angle-of-attack and Mach number, only angle-of-attack experiences significant changes during the maneuvers associated with these reference trajectories. This is due to the relatively small commanded changes in altitude and velocity when compared to the flight envelope, and thus small changes in Mach number. The centers of the linear-Gaussian nodes were arranged in a user-defined grid over the input space, with the highest density of nodes in the angle-of-attack dimension (due to expected nonlinearities). Moreover, the spatial decay of each node was varied as a function of the center location of its nearest neighbor. The closer the neighboring center, the higher the spatial decay, and conversely, the farther the neighboring center, the lower the spatial decay. This pattern ensures that each point in the input space can be adequately mapped to the desired output values. Initial values for the slopes and biases of the linear-Gaussian nodes were set to zero, since no *a priori* design information was assumed. Due to this initialization to zero, the learning system does not impact the states at start-up and all of the unknown dynamics are initially faced by the TDC adaptive component. After evaluating the magnitude of the unknown dynamics and disturbances supplied by the adaptive component, the cost function (Equation 3.20) was weighted to ensure all errors between the desired output and actual network output have the same significance. Equation (4.19) demonstrates how the cost function can be weighted for

specific errors between the desired and actual network output:

$$J = \frac{1}{2}\left[\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x},\mathbf{p})\right]^{T} \mathbf{C}\left[\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x},\mathbf{p})\right] \tag{4.19}$$

where **C** is a diagonal matrix with user-supplies weights along the diagonal. The global learning rate (α) was selected by trial and error in order to find the highest rate of convergence to the desired output with adequate accuracy while still maintaining a static mapping (i.e., one for which parameters are not in a continuous state of change).

The model of the aircraft dynamics, which is in a continuous time form, was integrated at 50 Hertz to provide a balanced tradeoff between numerical stability and processing speed. However, the control signal was calculated at a more moderate rate of 10 Hertz in order to reduce the real-time sensing and computation requirements in determining the complete state.

After running the simulation with randomly generated commands for 500 trials, of 20 seconds each, the learning system was able to build a mapping of a significant amount of the previously unknown dynamics. Since the true mapping of the unknown dynamics is not known (in contrast to the case with the aeroelastic oscillator), the cost, as defined in Equation (4.19), is used as a measure of performance of the learning system. Figure 4.17 illustrates both the initial cost and the cost after learning after 500 trials for a 500 foot climb and simultaneous 10 feet per second increase in velocity.

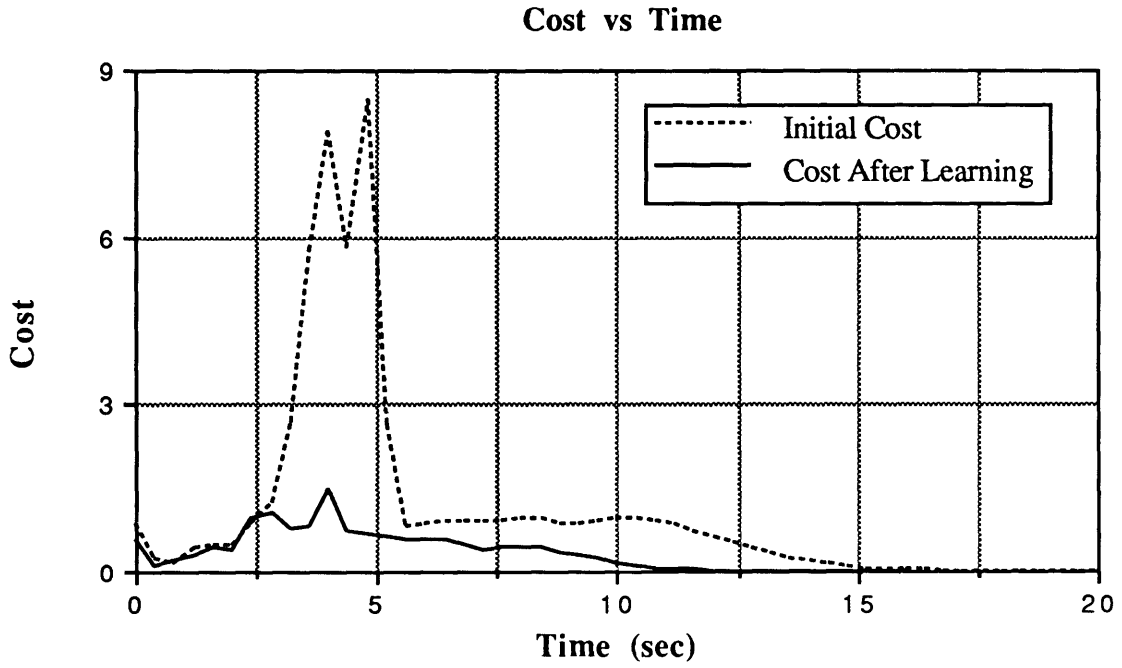

Figure 4.17 Comparison between initial cost and cost after 500 trials.

Since the cost is significantly less for the case after learning, this indicates that the learning component of the hybrid law has built a mapping of a significant amount of the unknown dynamics. If the simulation is allowed to run even longer, the cost will further decrease. However, since the true aircraft model dynamics are very high dimensional and contains states that are not included as inputs to the network (e.g., the state of the actuators), it is impossible to completely learn the initially unknown dynamics. For this reason, there will always be a finite, non-zero cost.

The state trajectories for the reference model, for the TDC controlled aircraft, and for the hybrid controlled aircraft for a commanded 500 foot climb and 10 feet per second increase in velocity are shown in Figures 4.18(a) through 4.23(a). Since the difference between these trajectories is typically small compared to the absolute initial trim values, the errors between the desired reference and the actual trajectory for both the TDC and hybrid controlled aircraft are shown in Figures 4.18(b) through 4.23(b). The horizontal stabilator deflection and throttle position for the TDC and hybrid controlled aircraft are shown in

Figures 4.24 and 4.25. These values represent the actual values used on the aircraft. Due to actuator dynamics, these actual values are generally not the commanded output calculated by the given control law.

As illustrated by the state trajectories, the hybrid control law offers improvements over the TDC controller. Although the errors in velocity and altitude are relatively small, errors in the vehicle rates and angles are significant in the sense that oscillations about the reference trajectory are reduced. This reduction in oscillations for the hybrid controlled aircraft has the potential to change a response that was formerly objectionable to the pilot to one that is satisfactory. Moreover, the horizontal stabilator deflection for the hybrid controlled aircraft is improved over that of the TDC controlled aircraft in the sense that the control signal is less oscillatory (and subsequently less taxing on the actuators).

The trajectories for the reference model and the hybrid controlled aircraft differ for two major reasons. The first, as previously discussed, is the inability of the learning system to map the unknown dynamics for states that are not given as inputs to the network (e.g., actuator states). Perhaps more significant are the difficulties associated with attempting to control more states than there are available control inputs. Since a pseudo-inverse must be used in the hybrid control law when the number of controls is less than the number of states as discussed in Section 3.3, the tracking of the complete state is not guaranteed even for a simulated case without any unknown dynamics (Anderson & Schmidt (1990)). Due to this inability to control all the state variables, it is almost certain that differences will exist between the reference and actual trajectories. As a result, errors between the reference trajectory and the hybrid controlled trajectory do not necessarily represent a failure of the learning system to map the unknown dynamics, but an inability to control all the states to a reference trajectory with a limited number of controls.

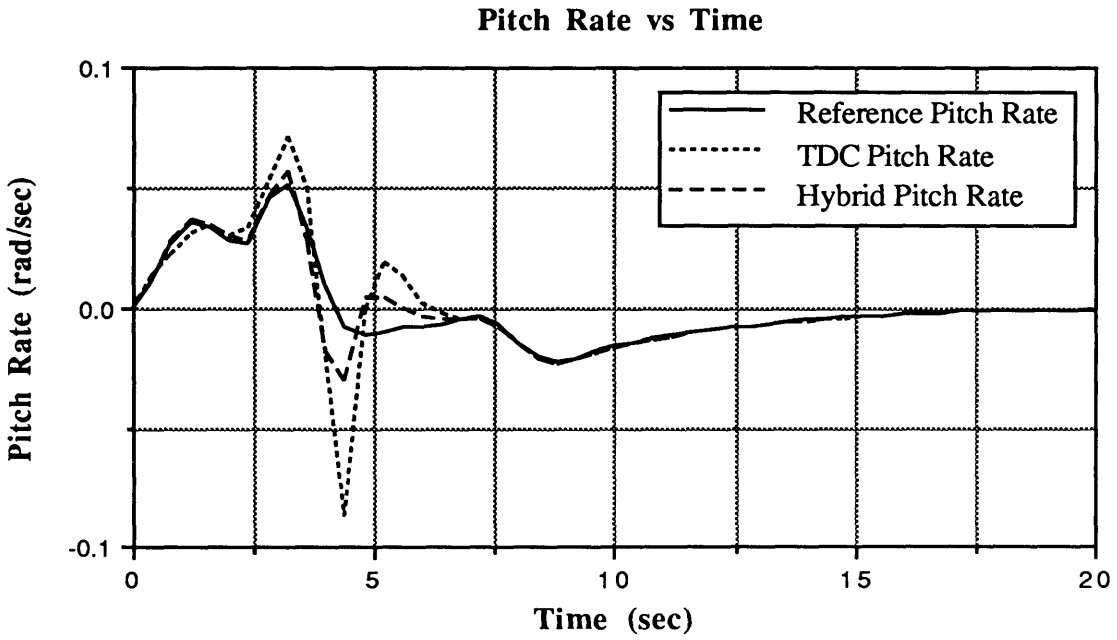

Figure 4.18(a) Pitch rate trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

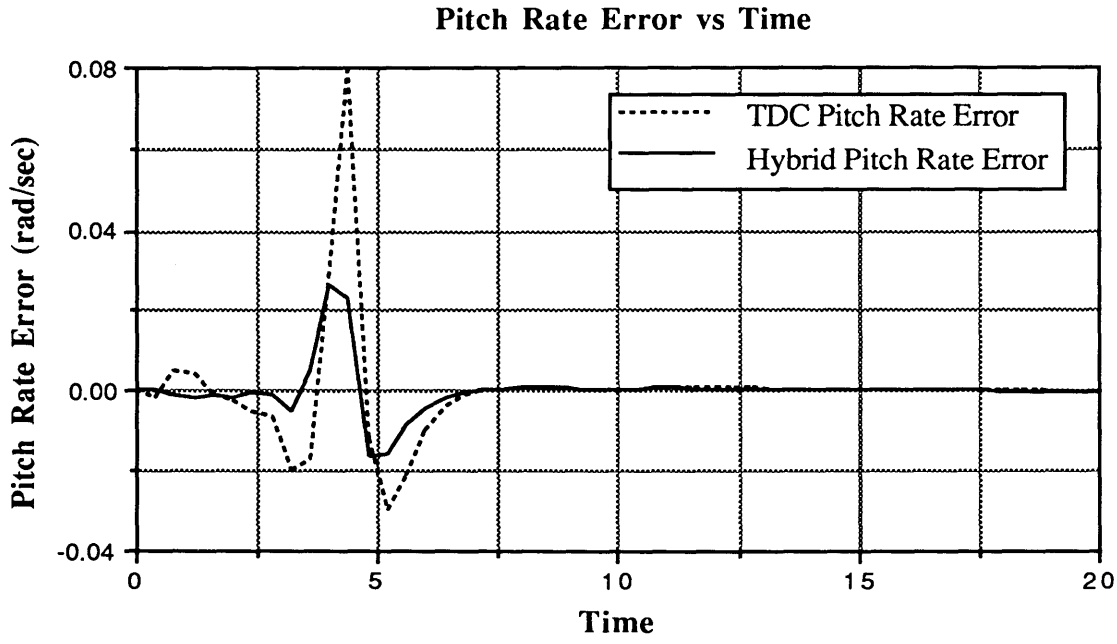

Figure 4.18(b) Error in pitch rate between reference trajectory and TDC or hybrid controlled aircraft.

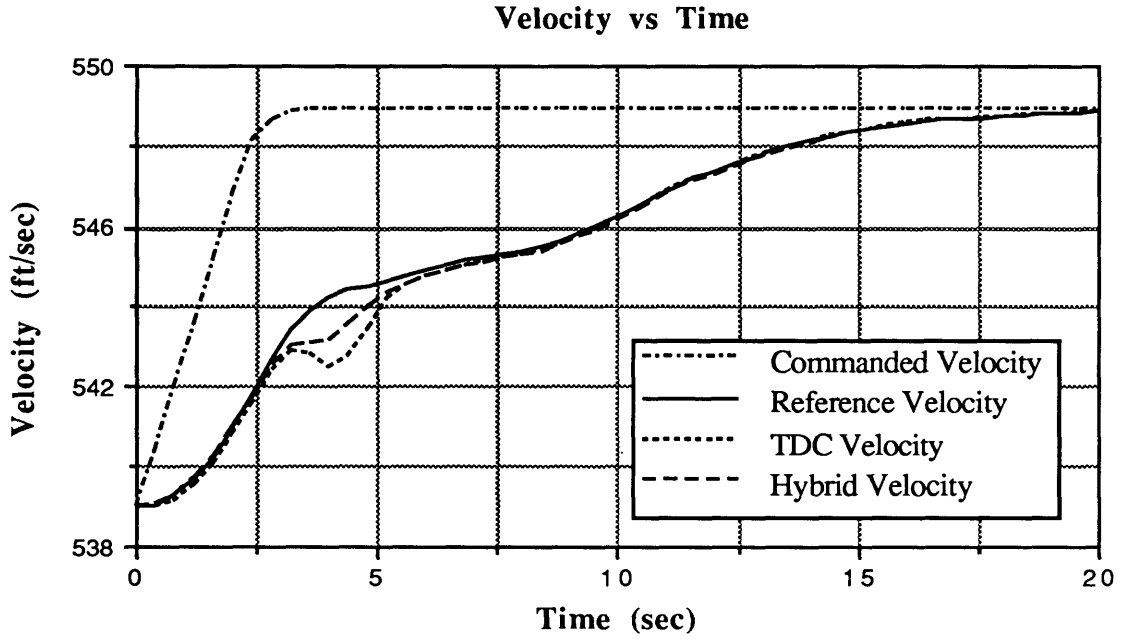

Figure 4.19(a) Velocity trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

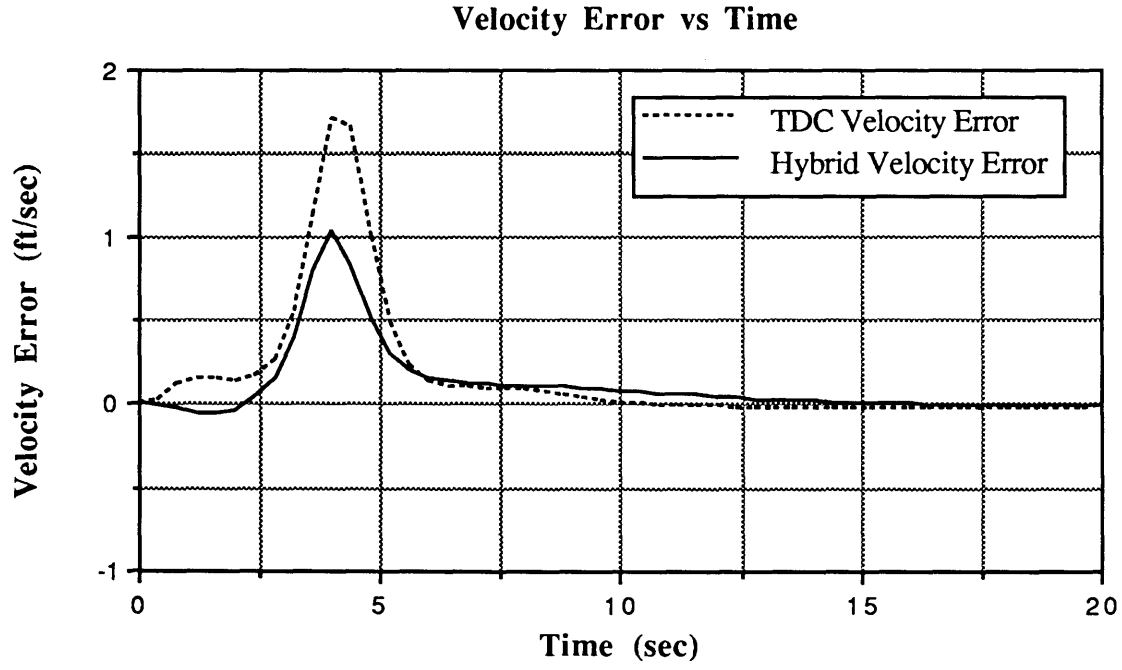

Figure 4.19(b) Error in velocity between reference trajectory and TDC or hybrid controlled aircraft.

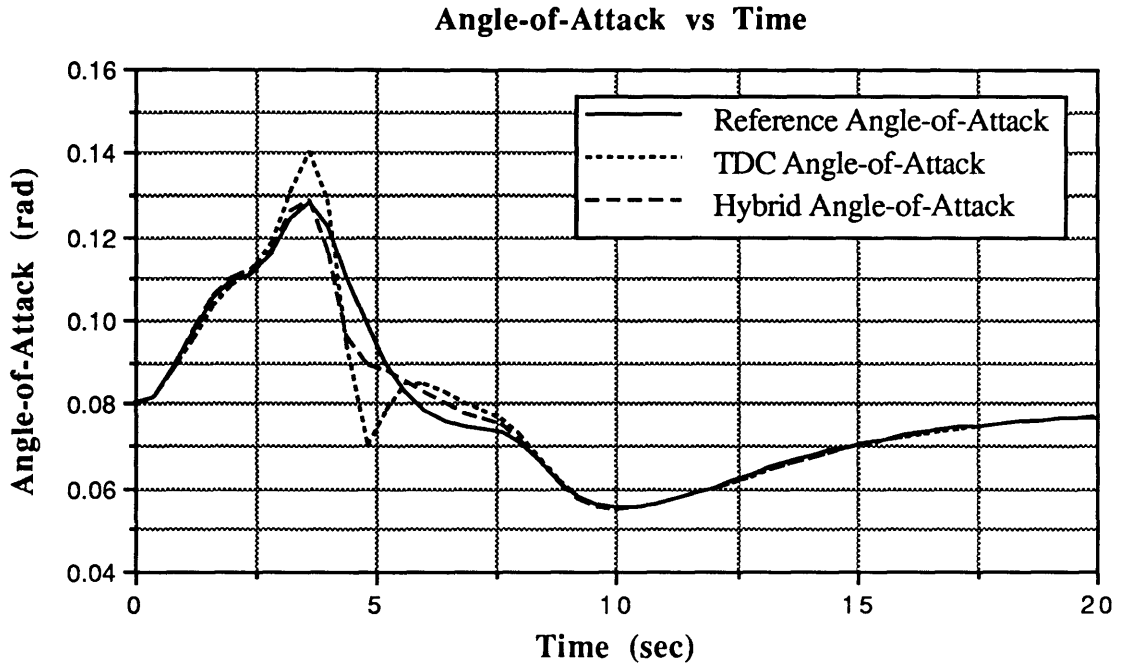

Figure 4.20(a) Angle-of-attack trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

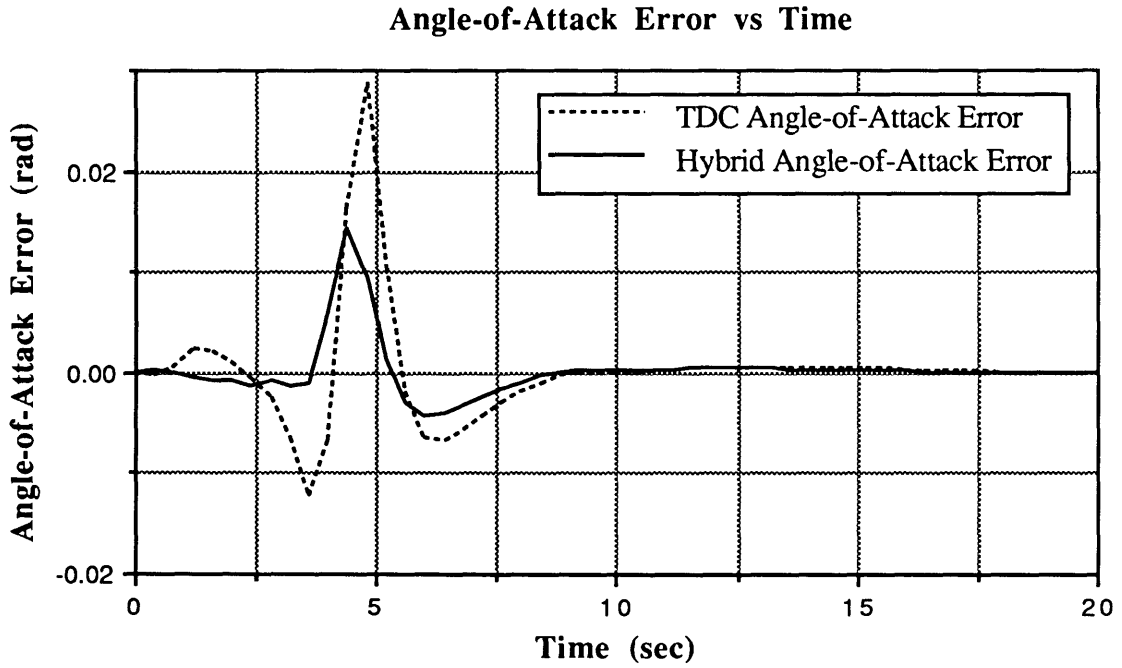

Figure 4.20(b) Error in angle-of-attack between reference trajectory and TDC or hybrid controlled aircraft.

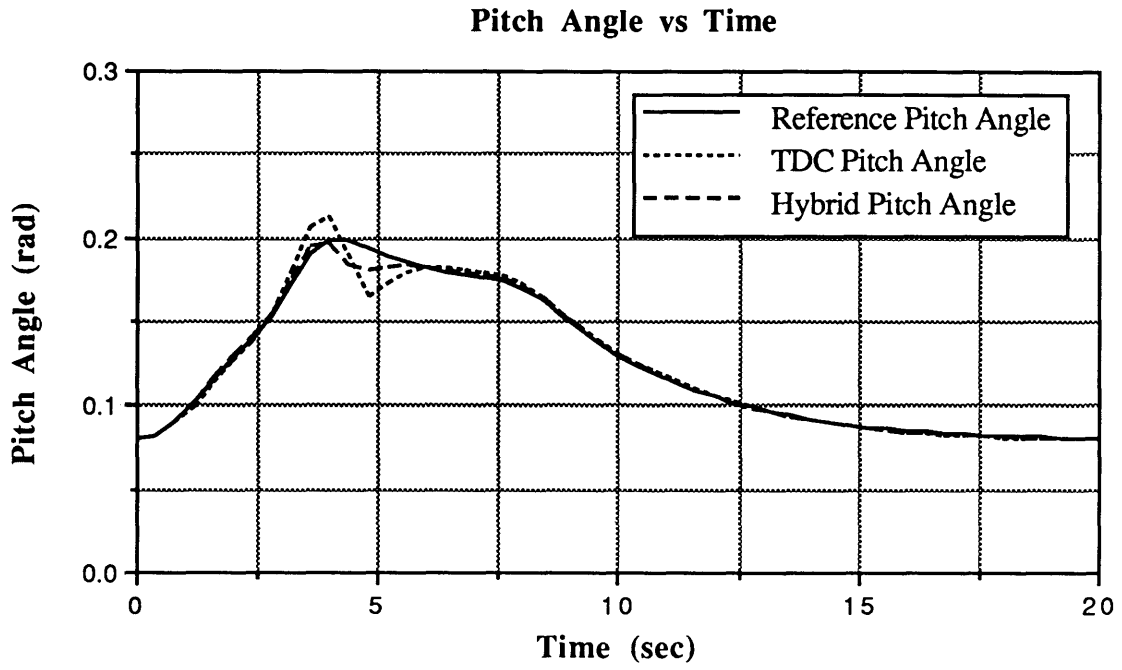

Figure 4.21(a) Pitch angle trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

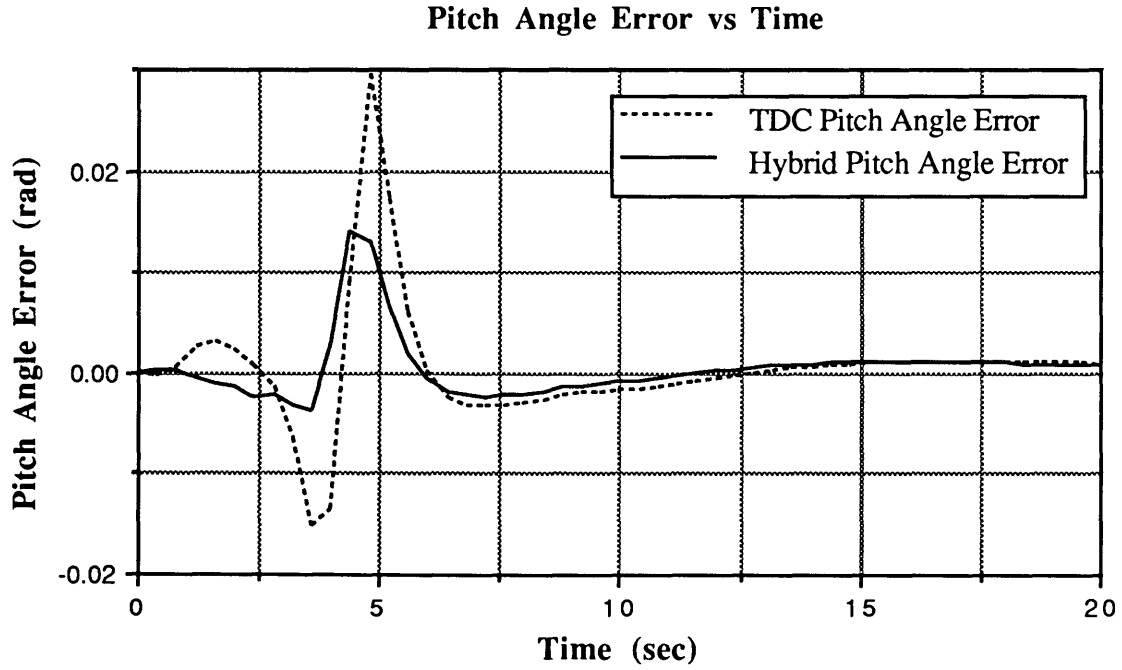

Figure 4.21(b) Error in pitch angle between reference trajectory and TDC or hybrid controlled aircraft.

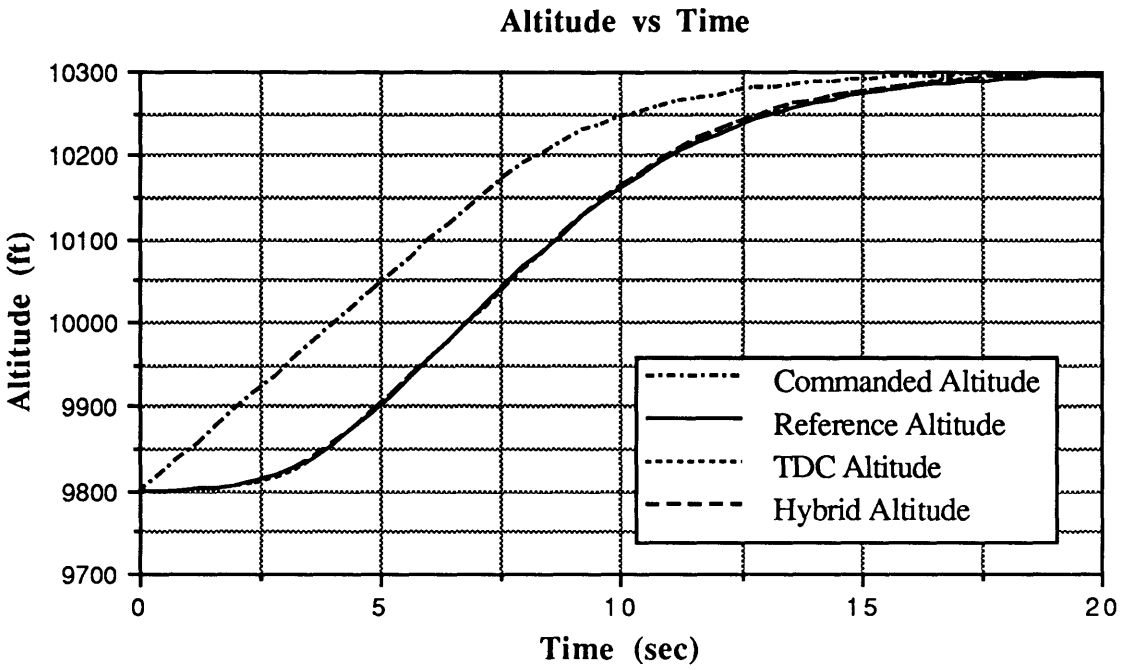

Figure 4.22(a) Altitude trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

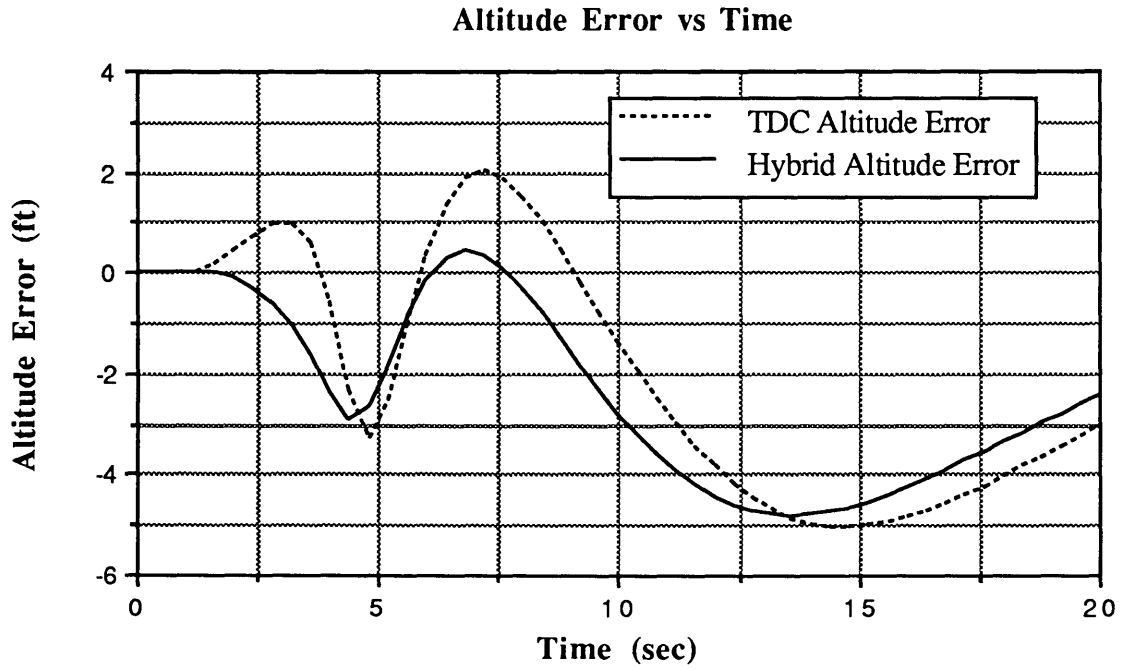

Figure 4.22(b) Error in altitude between reference trajectory and TDC or hybrid controlled aircraft.

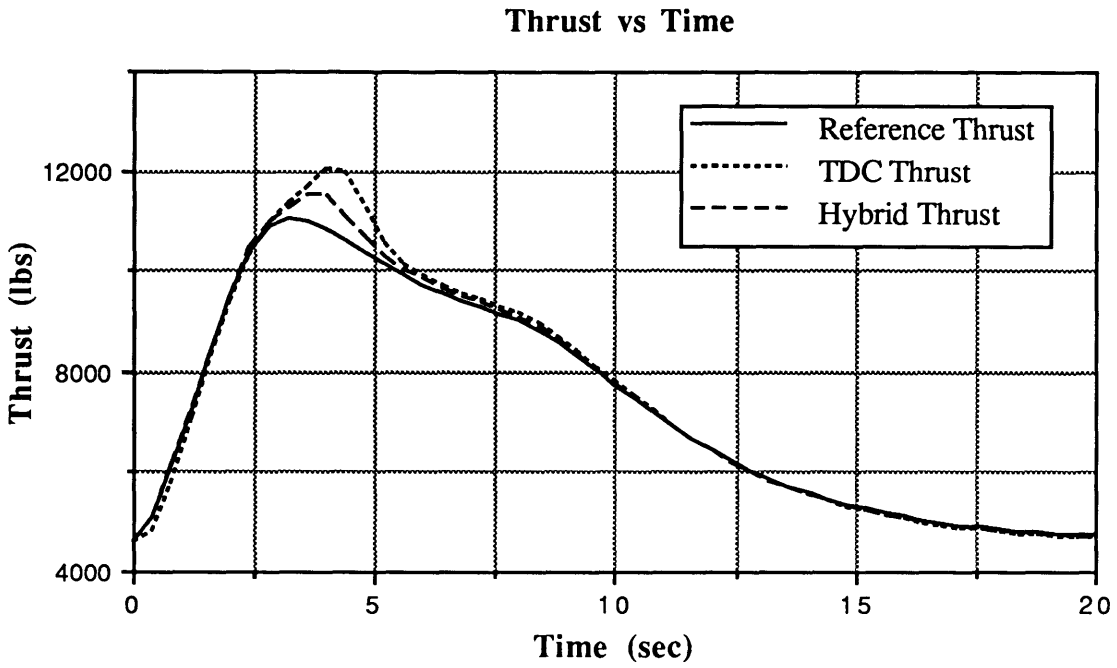

Figure 4.23(a) Thrust trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

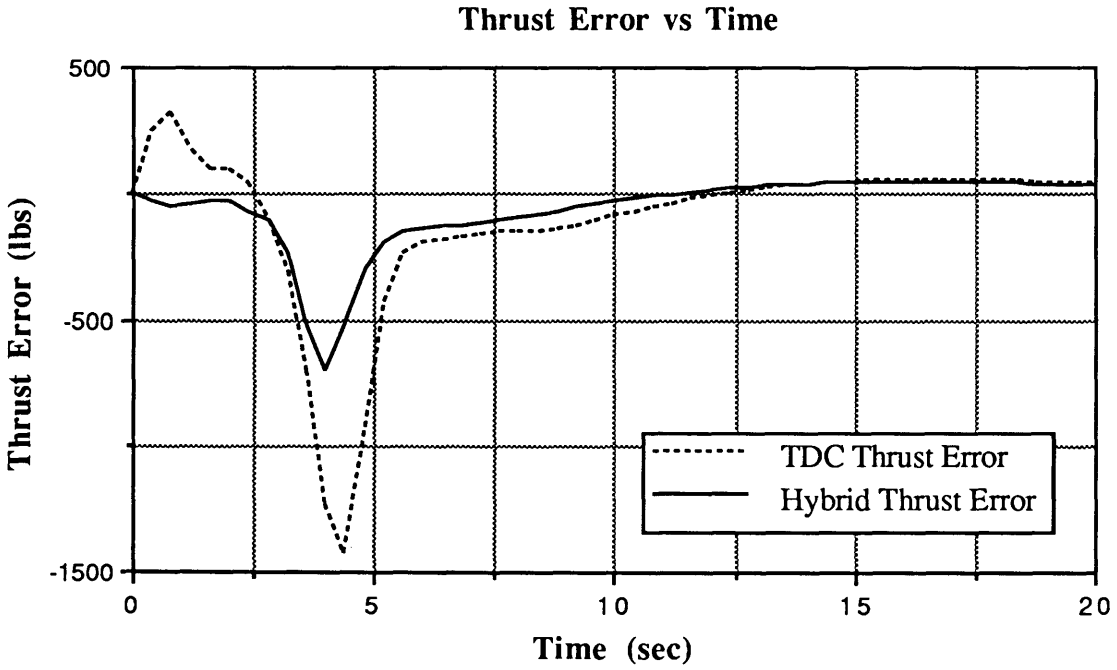

Figure 4.23(b) Error in thrust between reference trajectory and TDC or hybrid controlled aircraft.

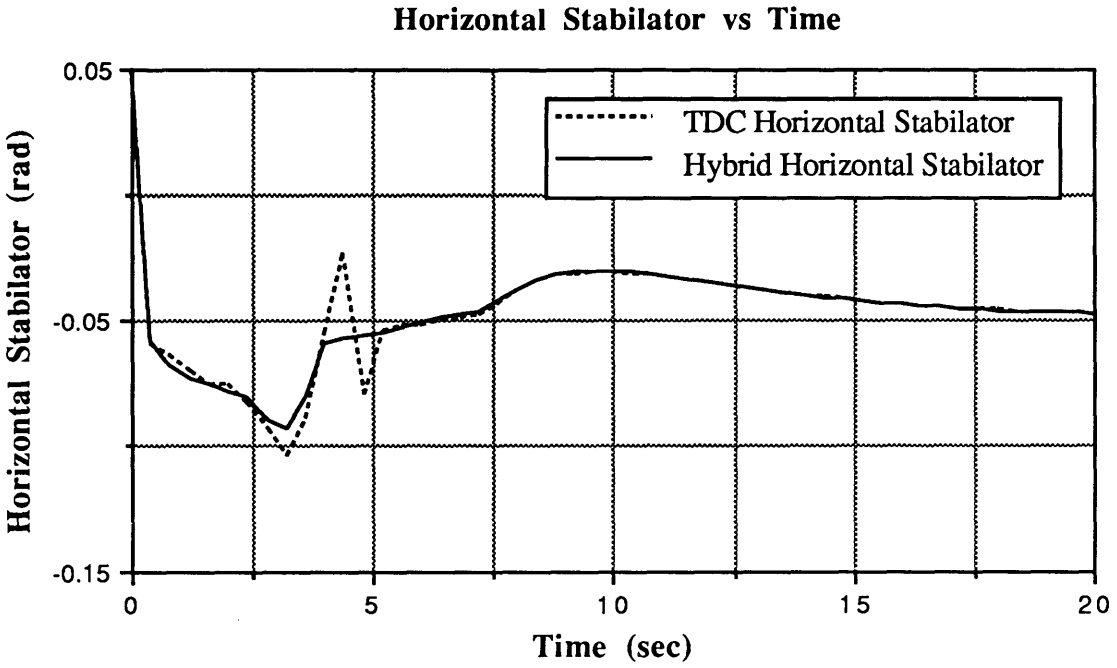

Figure 4.24 Horizontal stabilator deflection for the TDC and hybrid controlled aircraft.

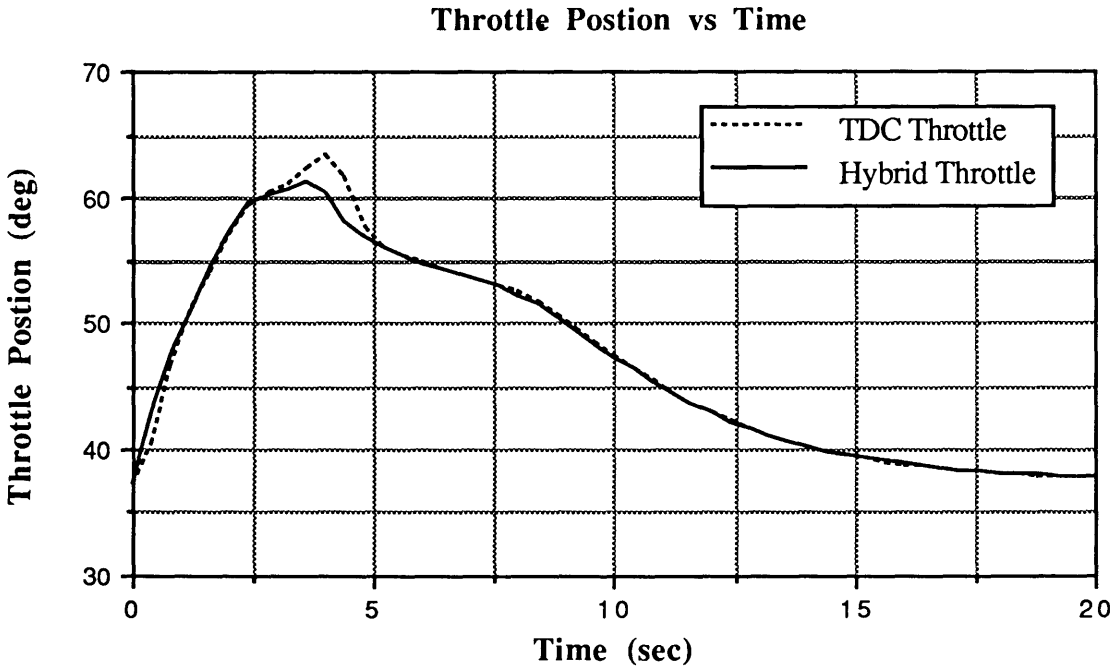

Figure 4.25 Throttle position for the TDC and hybrid controlled aircraft.

Nonetheless, experiment 1 demonstrates that the hybrid controller is able to improve the performance of the aircraft over the purely adaptive TDC controller. This improved performance is realized by exploiting the learned functional mapping of the previously unknown model dynamics to remove the delay associated with the adaptive component and reduce the model uncertainty to arrive at a superior nonlinear control law. The next experiment illustrates the ability to generalize the synthesized mapping to a larger input space generated by using a more demanding commanded altitude rate.

4.2.6 High Performance Aircraft Experiment 2

The objective of experiment 2 is to demonstrate the local generalization abilities of the learned functional mapping to areas of the input space that have not explicitly been trained. By increasing the rate limit on the randomly generated altitude command to 100 feet per second, the region of the input space for which controls must be computed is effectively increased. Moreover, the reference trajectory is more demanding in the sense that larger controls (resulting in larger angles and angular rates) are required to follow this trajectory.

Beyond the increased altitude rate limit, the setup of experiment 2 is identical to experiment 1 in terms of the learning system, initialization, and control calculation rate. Figures 4.26 through 4.31 contain the state trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft for a commanded 500 foot climb (at a 100 feet per second rate) and 10 feet per second increase in velocity using the *previously* trained network in experiment 1. The horizontal stabilator and throttle position applied to the aircraft for both the TDC and hybrid responses are shown in Figures 4.32 and 4.33.

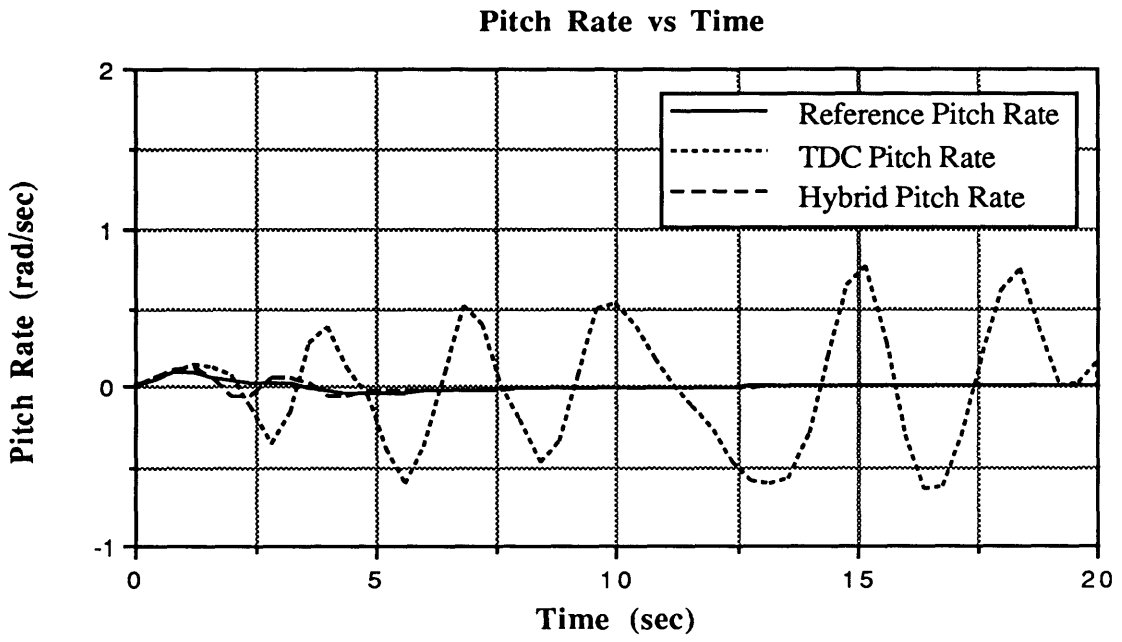

Figure 4.26 Pitch rate trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

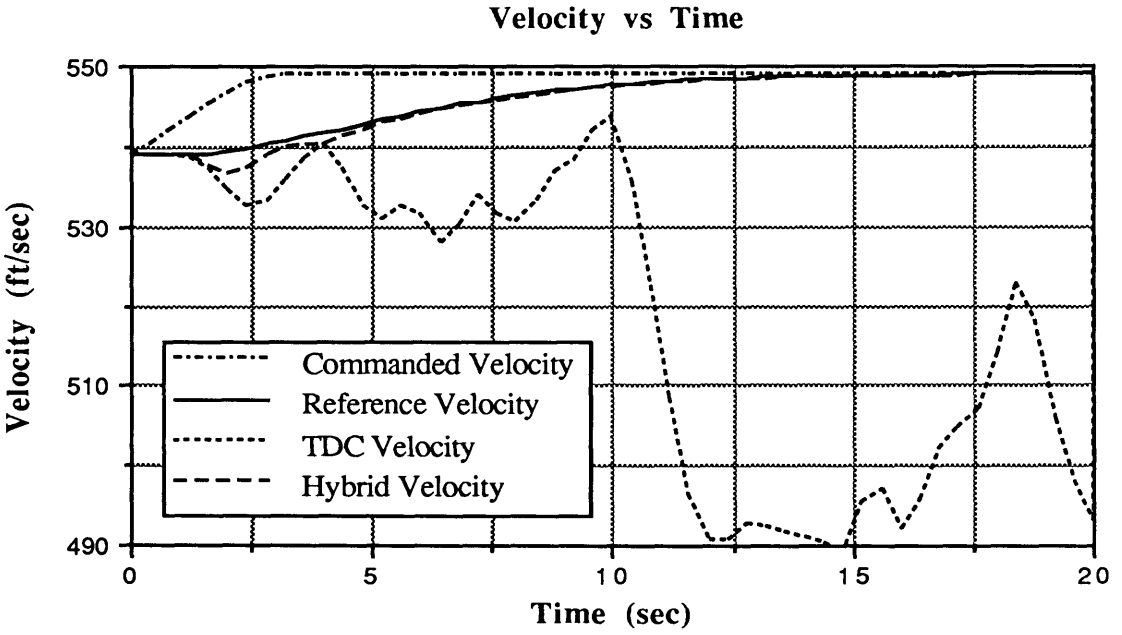

Figure 4.27 Velocity trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

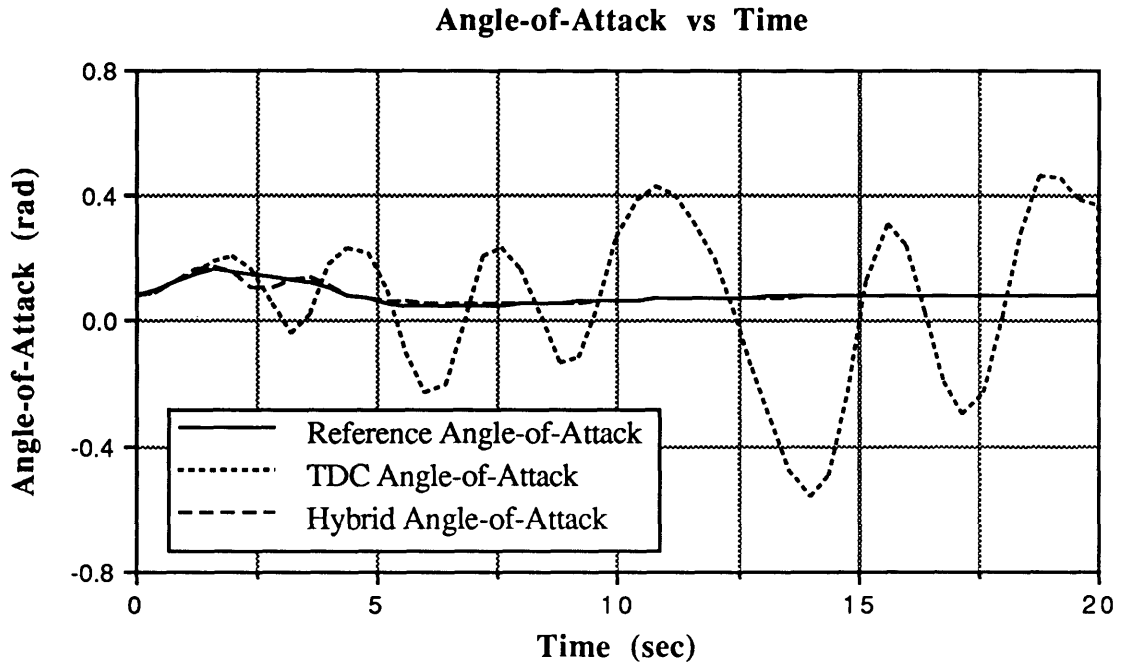

Figure 4.28 Angle-of-attack trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1

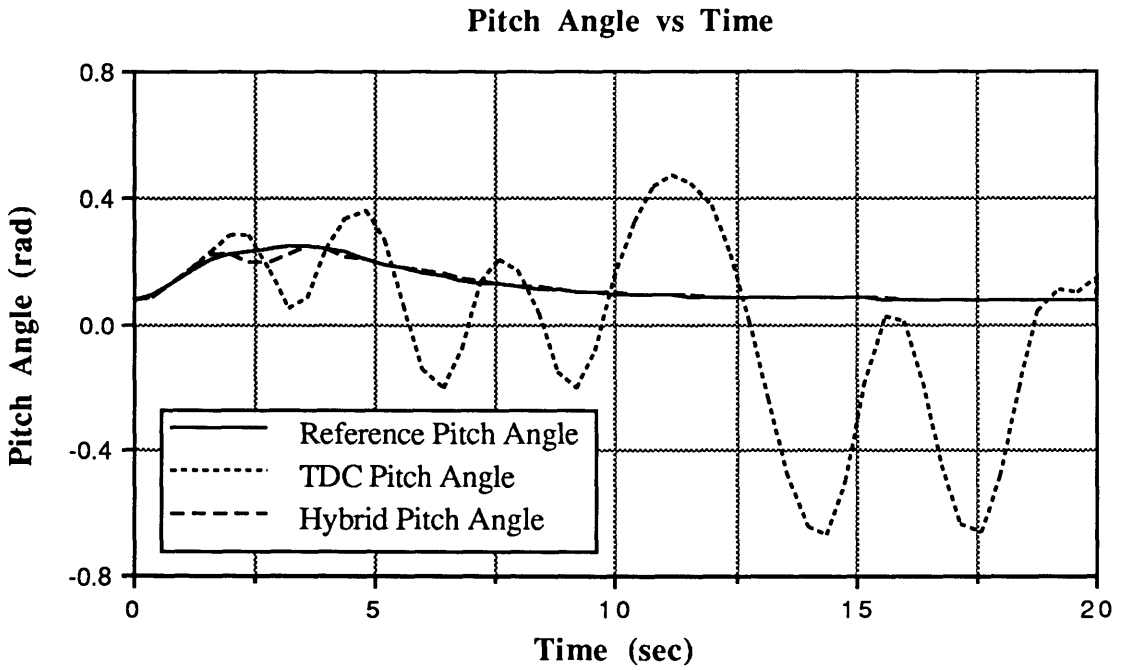

Figure 4.29 Pitch angle trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

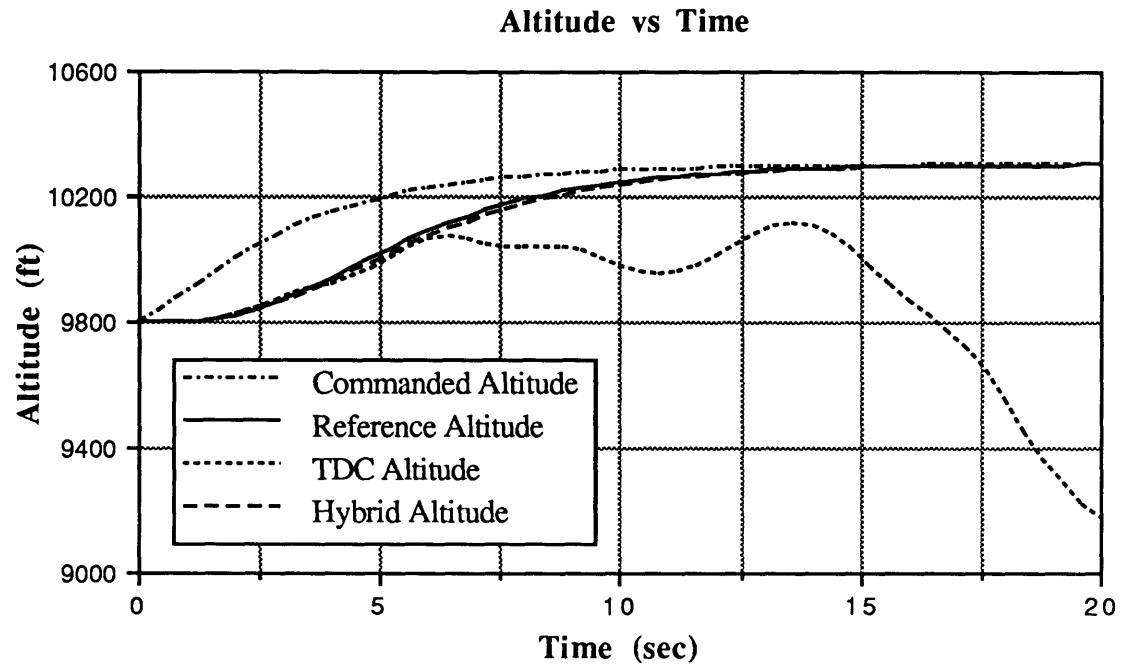

Figure 4.30 Altitude trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

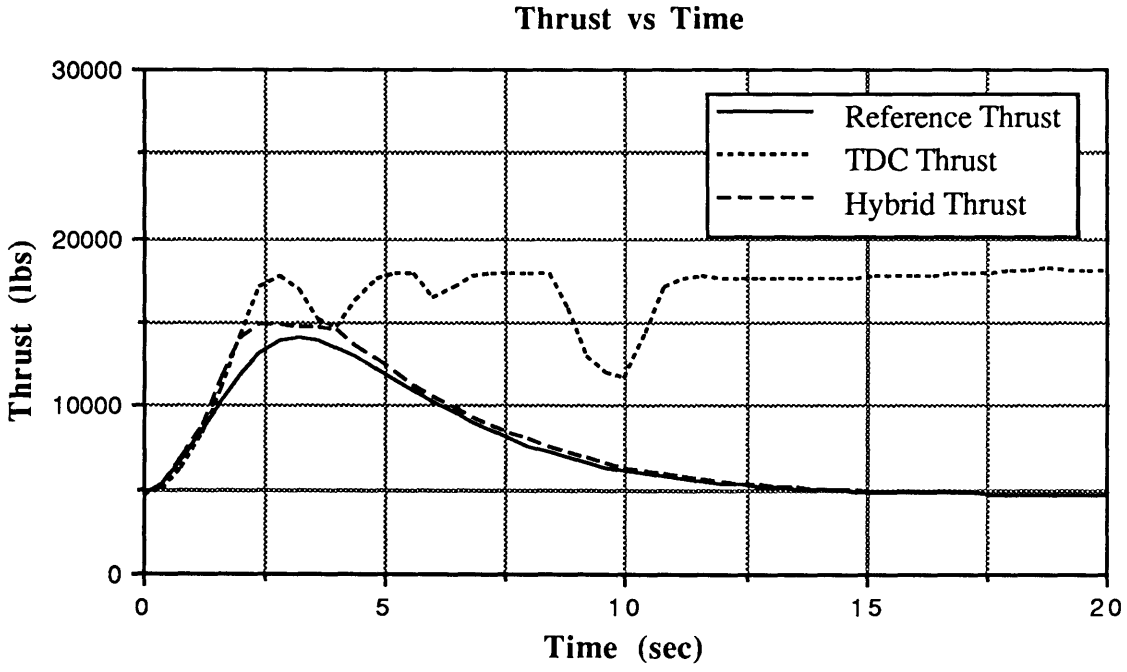

Figure 4.31 Thrust trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

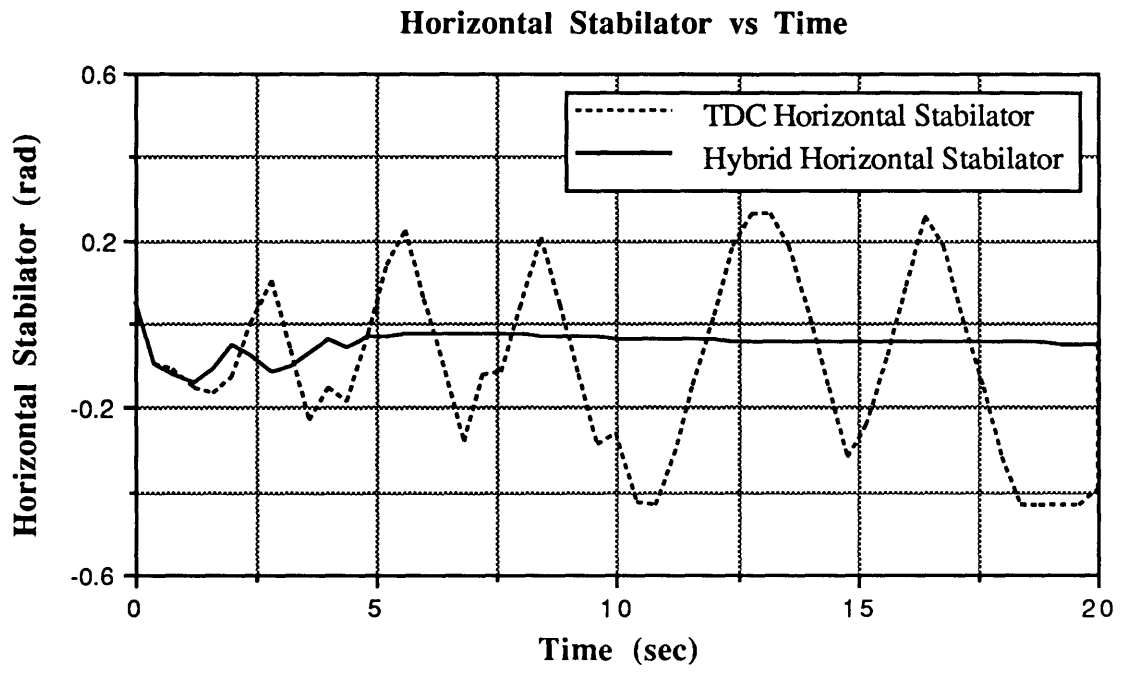

Figure 4.32 Horizontal stabilator deflection for the TDC and hybrid controlled aircraft.

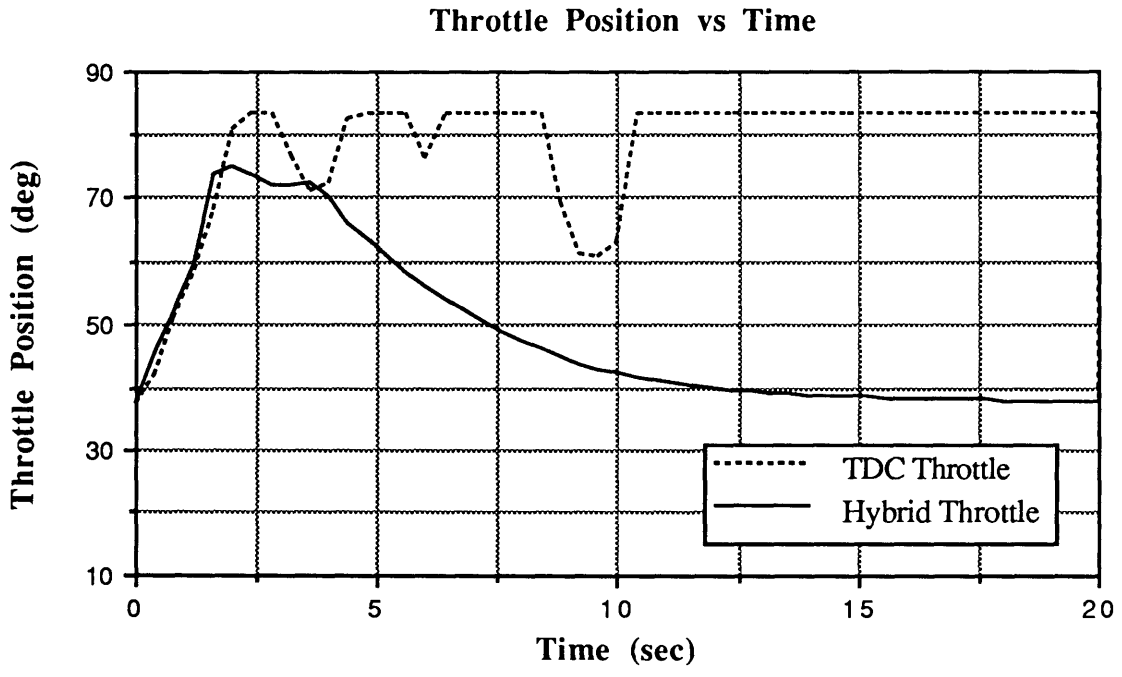

Figure 4.33 Throttle position for the TDC and hybrid controlled aircraft.

As illustrated by the state trajectories, the TDC control law is unable to provide the control necessary to reach the commanded states, whereas the hybrid controlled aircraft generally follows the reference trajectory. Moreover, the banging of the horizontal stabilator and throttle against their limits for the TDC controller illustrates a desperate attempt to regain the desired state trajectory. This failure of TDC demonstrates the consequences of not using experientially gained knowledge to remove the delay in the estimate in the unknown dynamics and an inability to accommodate model uncertainty (e.g., improve the *a priori* estimate of the control weighting matrix).

Experiment 2 also demonstrates the ability of the learning system to generalize to nearby regions of the input space for which it has not explicitly received training samples. This feature is especially important due to that fact that the hybrid control law uses a passive learning system. Under passive learning, the learning system does not guide the vehicle in an active search of the input space. Instead, the learning system is opportunistic in the sense that it learns for a given region of the input space presented by the adaptive controller for the state trajectories that have been flown. As a result, areas of the input space in which TDC in unable to traverse can not initially receive training information. However, due to generalization, the hybrid controller is able to stabilize and control the aircraft in areas the purely adaptive control law fails. Later excursions through these regions will provide additional inputs for the learning system to process. This, of course, suggests a conservative approach to flight testing / learning if the hybrid controller were to be employed. Since the hybrid controller is able to adequately control the aircraft given an inaccurate linear representation, less *a priori* design information is needed (i.e., fewer design point linearizations), effectively reducing design costs and automating the tuning process.

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 SUMMARY AND CONCLUSIONS

This thesis describes the development and application of a hybrid control system to the problem of flight control for a high performance aircraft. By combining an adaptive component based on the TDC approach with a learning system, an innovative new hybrid controller has been formed that allows each of these two mechanisms to focus on the control objective for which it is best suited. The adaptive component of the hybrid controller accommodates some of the unmodeled dynamics and provides estimates of any unmodeled state dependent dynamic behavior to the learning system. The connectionist learning system synthesizes a functional approximation of the state dependent dynamic behavior. Using this learned mapping, the hybrid control system is able to predict state dependent behavior, effectively removing the delay an adaptive controller experiences due to its reactive nature.

The impact of a controller that has the ability to anticipate vehicle behavior has been illustrated in terms of improved closed-loop aircraft performance. It has also been shown that by using derivative information from the learned mapping, model uncertainty could be reduced at each operating condition, essentially automating the tuning process normally associated with gain scheduled controllers. Due to its ability to reduce model uncertainty, the hybrid system adequately controls the aircraft even in situations where an inaccurate linear representation was used as the system model during the initial design of the control law. As a result, less *a priori* design information is needed (i.e., fewer design point linearizations), effectively reducing design costs.

This thesis has also demonstrated the ability of a spatially localized learning system to synthesize a nonlinear, multivariable mapping in a control environment. More specifically, it has been shown that a linear-Gaussian network is able to learn a functional approximation of the initially unknown dynamics, given state and control information, using an incremental learning approach.

5.2 RECOMMENDATIONS FOR FUTURE WORK

The major constraint to the amount of improvement the hybrid control system could realize was not a function of the unknown dynamics or the ability of the learning system to synthesize this mapping, but the requirement that *all* the states follow their given reference trajectory. Since aircraft have more states than controls, this requirement is unrealistic from the control standpoint. Moreover, in many cases, only a few of the states are of direct importance. Further research following (Anderson & Schmidt (1990)) should focus on reducing the number of controlled states to be the same as the number of control inputs. Using this approach, the pseudo-inverse required in the derivation of the hybrid control law would be replaced by a true inverse, essentially allowing perfect model following for the case where all of the initially unknown dynamics are learned and there is no state and control observation noise.

Another area for future work is the expansion of the hybrid control system to map the entire flight envelope, as compared to a small subset of trajectories. This research would require a much larger network than that used for the experiments in this thesis, due to the expected nonlinearities in Mach number as well as angle-of-attack. A thorough examination of the abilities of the hybrid control law trained over the entire flight envelope could further highlight the advantages of this learning enhanced controller over conventional techniques.

A future investigation into using different types of adaptive components (i.e., other

that TDC) in the hybrid control law is recommended (Astrom & Wittenmark (1989), Slotine & Li (1991)). Moreover, future research should examine areas of automatic flight control other than autopilots (e.g., stability augmentation systems and control augmentations systems) where the hybrid control law offers potential improvements.

BIBLIOGRAPHY

Alexander, J., Baird, L., Baker, W. & Farrell, J. (1991). "A Design & Simulation Tool for Connectionist Learning Control Systems: Application to Autonomous Underwater Vehicles" Draper Laboratory Report CSDL-P-3041, Cambridge, MA.

Albus, J. (1975). "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, pp. 220-227.

Anderson, M. & Schmidt, D. (1990). "Error Dynamics and Perfect Model Following with Application to Flight Control", *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, pp. 912-919.

Astrom, K. & Wittenmark, B. (1989). *Adaptive Control*, Addison-Wesley.

Athans, M. (1990). "Lecture Notes for Multivariable Control Systems I," School of Engineering, MIT, Cambridge, MA.

Athans, M., *et al.* (1977). "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method — Part I: Equilibrium Flight," *IEEE Transactions on Automatic Control*, Vol. 22, pp. 768-780.

Baird, L. (1991). "Learning and Adaptive Hybrid Systems for Nonlinear Control," M.S. Thesis, Northeastern University, Boston, MA.

Baker, W. & Farrell, J. (1990). "Connectionist Learning Systems for Control," Draper Laboratory Report CSDL-P-2922, Cambridge, MA.

Baker, W. & Farrell, J. (1991). "Learning Augmented Flight Control for High Performance Aircraft," Draper Laboratory Report CSDL-P-3080, Cambridge, MA.

Baker, W. & Farrell, J. (1992)."An Introduction to Connectionist Learning Control Systems," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, White, D. & Sofge, D. (eds.) Von Nostrand Reinhold.

Barto, A. (1989). "Connectionist Learning for Control: An Overview," COINS Technical Report 89-89, Department of Computer and Information Science, University of Massachusetts, Amherst.

Barto, A., Sutton, R., & Anderson, C. (1983). "Neuronlike Adaptive Elements That Can Solve Difficult Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, No. 5.

Brumbaugh, R. (1991). "An Aircraft Model for the AIAA Controls Design Challenge," *Proceedings, AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA.

Bryson, A. & Ho, Y. (1975). *Applied Optimal Control*, Hemisphere.

Duke, E. (1992). "AIAA Control Design Challenge."

Etkin, B. (1982). *Dynamics of Flight - Stability and Control*, John Wiley and Sons.

Farrell, J. & Baker, W. (1991). "Learning Control Systems," *Intelligent and Autonomous Control Systems,* Antsaklis, P. & Passino, K., (eds.), Kluiver Academic.

Funahashi, K. (1988). "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp. 183-192.

Hornik, K., Stinchcombe, M., & White, H. (1989). "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366.

Jacobs, R. (1988). "Increased Rates of Convergence Through Learning Rate Adaptation", *Neural Networks*, Vol. 1, No. 4, pp. 295-307.

Jordan, M. (1988). "Sequential Dependencies and Systems with Excess Degrees of Freedom," COINS Technical Report 88-27, University of Massachusetts, Amherst.

Klopf, H. (1988). "A Neuronal Model of Classical Conditioning," *Psychobiology*, Vol. 16, pp. 85-125.

Kwakernaak, H. & Sivan, R. (1972). *Linear Optimal Control Systems*, John Wiley and Sons.

Lewis, F. & Stevens, B. (1992). *Aircraft Control and Simulation*, John Wiley and Sons.

Livstone, M., Farrell, J., & Baker, W. (1992). "A Computationally Efficient Algorithm for Training Recurrent Connectionist Networks," *Proceedings, 1992 American Control Conference.*

Melsa, P. (1989). "Neural Networks: A Conceptual Overview," Tellabs Research Center, Mishawaka, IN.

Mendal, J. & McLaren, R. (1970). "Reinforcement Learning and Pattern Recognition Systems," *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*. Academic Press, New York.

Millington, P. (1991). "Associative Reinforcement Learning for Optimal Control," S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

MIL-F-8785C (1980). *U.S. Department of Defense Military Specification: Flying Qualities of Piloted Airplanes.*

Minsky, L. & Papert, S. (1969). *Perceptrons*, MIT Press, Cambridge, MA.

Poggio, T. & Girosi, F. (1990). "Networks for Approximation and Learning," *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481-1497.

Parkinson, G. & Smith, J. (1963). "The Square Prism as an Aeroelastic Non-Linear Oscillator," *Quarterly Journal of Mechanics and Applied Mathematics*, Vol. 17, pp. 225-239.

Rumelhart, D., Hinton, G., & Williams, R. (1986). "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognitions, Vol. 1: Foundations*, MIT Press, Cambridge.

Rosenblatt, F. (1962). *Principles of Neurodynamics*, Spartan Books, Washington.

Roskam, J. (1979). *Airplane Flight Dynamics and Automatic Flight Controls*, Roskam Aviation and Engineering Corp.

Slotine, J. & Li, W. (1991). *Applied Nonlinear Control*, Prentice-Hall.

Stein, G. (1980). "Adaptive Flight Control: A Pragmatic View," in *Applications of Adaptive Control*, Narendra, K. & Monopoli, R. (eds.), Academic Press.

Stein, G. , Hartmann, G., & Hendrick, R. (1977). "Adaptive Control Laws for F-8 Flight Test," *IEEE Transactions on Automatic Control*, Vol. 22, pp. 758-767.

Werbos, P. (1989). "Neural Networks for Control and System Identification," *Proceedings, IEEE Decision and Control Conference*, Tampa, FL.

Widrow, B. & Hoff, M. (1960). "Adaptive Switching Circuits," *IRE Convention Record.* New York.

Widrow, B. & Smith, F. (1964). "Pattern-Recognizing Control Systems," *Computer and Information Sciences Proceedings*, Washington, D.C.

Youcef-Toumi, K. & Osamu, I. (1990). "A Time Delay Controller for Systems with Unknown Dynamics," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 112.