KNOWLEDGE-BASED SYSTEMS AND FUZZY LOGIC FOR AUTOMATIC CONTROL

APPLICATION OF KNOWLEDGE-BASED SYSTEMS AND FUZZY LOGIC TO AUTOMATIC CONTROL

by

GREGORY P. FARISH, B.A.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Engineering

McMaster University

April 1989

MASTER OF ENGINEERING (1989)	McMaster University
(Electrical and Computer Engineering)	Hamilton, Ontario

TITLE	: Application of Knowledge-Based Systems and Fuzzy Logic to Automatic Control
AUTHOR	: Gregory Paul Farish B.A.Sc. (University of Waterloo)
SUPERVISOR	: Professor N.K. Sinha
NUMBER OF PAGES	: xii, 115

ABSTRACT

This thesis investigates the application of Knowledge Based systems and Fuzzy Logic to automatic control. The knowledge used by a human operator is put in a computer usable form and applied to a control problem. The idea is not to attempt to enhance the stability or response of the system but given a basically stable and controllable system we apply human type control methods via a computer controller.

A system can never be modelled exactly and therefore a controller design must allow for the uncertainty in the model. With fuzzy logic, the system inputs, outputs, parameters, reactions and cross coupling are represented in fuzzy or inexact variables, knowledge and reasoning. An exact (or nearly exact) model of the system is not necessary.

A simple aircraft is the process to which this control method is applied. Knowledge, reasoning and feedback similar to what a human pilot utilizes are applied in the control of the process.

ACKNOWLEDGEMENTS

The author is greatly appreciative of Dr. N.K. Sinha for his guidance, supervision, and assistance throughout the course of this work and in the preparation of this thesis.

The author also wishes to acknowledge with thanks the useful discussions he had with Dr. C. Weaver and colleagues R. Lingarkar and L. Liu.

The financial assistance provided by the Department of Electrical and Computer Engineering, McMaster University is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (cont'd)

LIST OF TABLES

LIST OF SYMBOLS

$\in$	down-wash factor
α	angle of attack
α_{wo}	angle between flight direction and zero lift line
σ	angle between thrust vector and zero lift line
τ	thrust coefficient
δ	flight angle
δ_e	elevator zero lift angle
a_w	wing slope-of-lift-curve
B	moment of inertia
bhp	brake horsepower
$\bar{c}$	mean aerodynamic chord
C_D	drag coefficient
C_L	lift coefficient
C_m	pitching moment coefficient
$Cm_{\alpha f}$	fuselage longitudinal static stability coefficient
Cm_{ac}	wing pitching moment coefficient
Cm_f	fuselage pitching moment coefficient
Cm_{it}	stabilizer effectivness
Cm_t	tail pitching moment coefficient
Cm_{ya}	total pitching moment coefficient
D	drag
damp	thrust damping, inversely proportional; to velocity
deltaT	simulation time step

e	propeller efficiency factor
exp	exponential to base e
fpm	feet per minute
g	gravitational constant
h	aircraft altitude
i_t	tail incidence
i_w	wing incidence
K	ratio of pitching moment to moment of inertia
K_f	fuselage moment factor
l	tail moment arm
m	mass
M_c	pitching moment
mK	throttle control input signal
M_{ya}	total pitching moment
n_t	tail efficiency
p	air density
pitch	propeller pitch in inches
q	pitch rate
rhp	rated horsepower
rpm	revolutions per minute
rpmDamp	rpm damping, propeller loading
S	wing area
S_t	tail surface area
T	thrust
t	time
thrustCoeff	thrust coefficient, thrust divided by aircraft weight

V	current velocity
V_{at}	tail volume
Vdiff	difference between V and Vnom
Vnom	nominal cruise velocity
Vol_f	fuselage volume
w_i	fuselage station width
$\underline{x}_a$	lift vector moment arm
z_T	distance of thrust vector from the centre of gravity in z-axis

CHAPTER 1

INTRODUCTION

Attempting to model a system accurately enough to be able to design an efficient and reliable controller can be a daunting task. As the number of inputs and outputs increase it becomes even more difficult to develop a model due to the cross coupling between inputs and outputs. Humans demonstrate the ability to cope with such complex systems and effectively control their operation without explicit knowledge of the inner workings and construction of the system. A human operator will have a collection of heuristics and facts gained through practice and experience which enable him or her to vary the system inputs to achieve the desired outputs. The largest failing of a human operator is wandering attention, boredom and inconsistent application of performance criteria.

A computer does not get bored or forget facts, yet it deals with the world in tedious detail. Much of the appearance of this tedious detail can be removed with a well designed man-machine interface. Still, the computer will present some small number as zero when all that is needed is to know if the result is close or not, as defined in the context of the problem.

This thesis addresses the application of a human operator's knowledge and methods in a computer based automatic controller. The controller is implemented using expert or knowledge-based systems techniques combined with fuzzy logic for reasoning with uncertainty and linguistic variables.

1.1 Solution and Direction

With a knowledge-based system, a knowledge base is navigated by an inference mechanism to find solutions to questions posed of the system. In applying knowledge-based systems to control, the operator's knowledge of how to control the system is entered into the knowledge base. This knowledge will be a collection of rules, heuristics and facts represented in a form usable by the computer through an inference mechanism. An accurate model of the system is not required, and in fact, if the knowledge base can lead to the control of a specific example of a system, then it should be able to control all examples within the same general type. For special situations within this class, new knowledge may have to be added to meet performance requirements, but basically the controller should be transferable.

If the knowledge-based system is combined with an implementation of fuzzy logic the computer can attempt to emulate the human operator with tireless repetition. The use of fuzzy logic allows the controller to cope with the uncertainty that the operator may express about the operation of the system as well as use some of the linguistic terms the human operator uses to express his heuristics.

With knowledge based control systems it is hoped that a more adaptable and flexible controller can be developed that mimics some of the more desirable traits of a human operator, possibly to include self-learning about the system operation, failures and development of new knowledge. Knowledge-based systems have found application where a system is large and complicated and the human operator, with many years of experience, is leaving the work force. With a knowledge based system, it will also be possible for the system to relax performance criteria if the system is highly stressed in order to prevent damage to the system. For example, it is at times necessary to turn off an aircraft autopilot in heavy turbulence in order to prevent damaging the aircraft. The controller is attempting to maintain the performance criteria in the presence of an extreme operating environment. The human operator, the pilot, relaxes the performance criteria in order to maintain the criterion of aircraft integrity.

1.2 Goal

The foremost goal of this thesis was to build a prototype knowledge-based controller incorporating fuzzy logic to investigate the application of such a controller (or system) to control applications, especially to a multi-variable system.

1.3 Scope

The system to be used in developing this prototype will be the control of power and the pitch axis of a light aircraft. An aircraft is a multi-variable system requiring a skilled operator, yet a pilot is licenced to fly a whole class of aircraft. This recognizes the fact that the expertise is transferable, even though the aircraft may be quite different in physical appearance.

1.4 Sequence of Presentation

Chapter 2 presents background information on knowledge-based systems, giving a short discussion on knowledge representation schemes and inferencing methods. In Chapter 3 a similar presentation of fuzzy logic is given.

Chapter 4 presents the criteria for selecting an example system for developing and testing the prototype controller. The aircraft parameters and the simulator derivation are given. Chapter 5 describes the criteria for selecting development tools for this thesis. It also discusses the tools evaluated and used in this thesis.

Chapter 6 presents the design of the controller, a discussion of the knowledge representation, the control strategy and the instrument reading scheme. In addition the fuzzy set and defuzzifier used in this thesis are presented.

Chapter 7 presents some results from test runs of the system. Neural networks exhibit some desirable properties even in conventional computers. Chapter 8 discusses these properties along with applying neural networks to automatic control. Their use is demonstrated with a few examples related to different aspects of the control application in this thesis. Chapter 9 summarizes the thesis, presents some conclusions and suggestions for future research.

Chapter 10 looks at the ethics and the engineers responsibility in researching and developing automated systems. The discussion is centred on the work done in this thesis and therefore its main theme is the development of sophisticated autopilots. Important questions are asked about the direction of this technology with emphasis on considering the impetus for it as well as the benefits and costs to society. Not all the questions are directly answered, but two main conclusions are drawn.

CHAPTER 2

KNOWLEDGE-BASED SYSTEMS

2.1 Introduction

A knowledge-based system is constructed of three major components. One is the group of application modules which provide an interface between the user and the system. The remaining two, the inference engine and the knowledge-base, are the main components for the system. The inference engine provides a means of navigating through the domain knowledge which is stored separately in the knowledge-base. It is important to note that the knowledge for the system is stored separately from the structure of the program. With a conventional program the knowledge of how to solve the problem is coded into the program's algorithm. In a knowledge-based system the program, essentially the inference engine, contains only enough information or knowledge in the algorithm for using the knowledge-base but no knowledge of the specific problem domain on which it is working.

2.2 Background

Knowledge-based or expert systems are computer programs which deal with problems usually requiring a certain amount of human expertise. Expert systems have recently become very popular for application to many tasks and may be regarded as the most recent and largest success of the artificial intelligence field. Artificial intelligence can be loosely defined as systems that exhibit the characteristics we associate with intelligence in human behaviour [JAC86].

Expert systems developed as researchers began to realize that knowledge representation was the pivotal problem in AI. This occurred in the early 1970's as a shift in attention away from general principles for problem solving toward "task specific" principles. Since the early 1970's work has concentrated on techniques and paradigms and their application to reasonable problem domains. The source of the reasoning power in an expert system is in the knowledge it contains, not in the reasoning mechanism used [ARC87].

The main points to emerge from this period about knowledge-based systems were that humans and why not the computer, deal with knowledge in an explicit, declarative and piecemeal fashion. In addition, there emerged the understanding that programming in this fashion allows for fast and incremental system prototyping and development. The program should not have to solve the whole problem, or indeed always be correct to be useful.

As already mentioned, a knowledge-based system is constructed of three major components. The system interface or application modules present information to and receive input from, the user or other sources such as files or processes. The second component, the inference engine, provides a means of working with the knowledge the system contains in the knowledge-base. This is a relatively simple program for tracing a path through the knowledge-base of the system to find a conclusion. The third section, the knowledge-base, contains the domain specific knowledge in a format which best suits the domain. Figure 2.1 summarizes the layout of a typical knowledge-based system.

2.3 Knowledge Representation Schemes

Knowledge representation is a set of syntactic and semantic conventions that make it possible to describe things. The syntax is made up of rules on how to combine symbols into expressions. It describes the how of creating the knowledge-base in the representation language, given that you already have the knowledge. The semantics is the interpretation of these expressions, or the meaning.

There are currently three main formalisms for knowledge representation in knowledge-based systems. These are:

1) Production Rules
2) Structured Objects
3) Predicate Logic.

All three representation schemes are an implementation

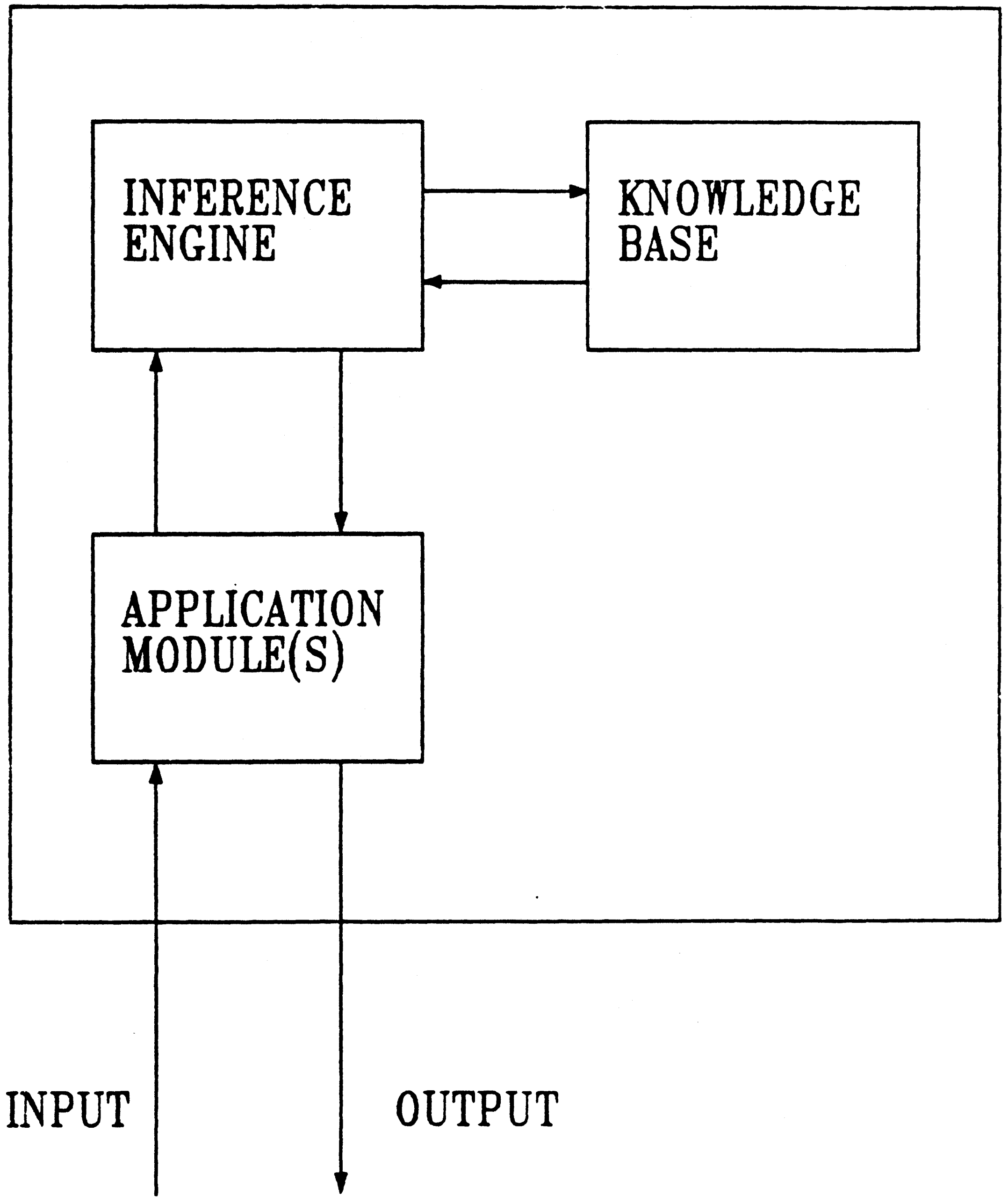


Figure 2.1 Knowledge-based System Block Diagram

of pattern directed inference.

2.3.1 Production Rules

A production system consists of a rule set, a rule interpreter and working memory. A rule has a premise or left hand side condition(s) and one or more action(s) or conclusion(s). For example consider the following:

Rule: if P1 and ... and Pn then C1 and ... and Cm.

Working memory is simply data storage. It holds data needed to evaluate the premise and may be updated by the conclusion. The interpreter (inference engine) attempts to match the premise of each rule against the information in working memory, provides conflict resolution if more than one rule is eligible to fire and fires the rule by executing the actions in the conclusion part of the rule. In a production system the order of the rules is unimportant as the conclusions are derived through the use of the inferencing mechanism or rule interpreter.

2.3.2 Structured Objects

Structured objects attempt to exploit some property in the knowledge being represented. Relationships within the knowledge are maintained and then exploited during inferencing to give more power to the system. The structures most commonly used are simple graphs, trees and networks. Simple graphs are used ,mostly for spatial and temporal relations and causal relationships. Trees allow for the exploitation of hierarchies in the domain while semantic networks are used to represent concepts and relationships between them.

Within these structures, further breakdowns of the knowledge can exist. A frame representation is a data structure to represent a stereotypical element, either real or abstract, or an event. Within the frame, there are slots which contain data to describe the element. Also within the frame there can be rules or instructions, sometimes called methods, on how the frame or object acts in the system. These frames can be linked together in a hierarchical fashion with each child being a more specific instance of the parent. The siblings inherit all the properties, data and rules, of the parent and implement new ones to describe their particular instance.

2.3.3 Predicate Logic

Predicate logic is derived from propositional logic. Propositional logic deals with simple statements such as "a computer is a tool" and "tools are useful", and is therefore not very expressive. For instance, statements such as "all animals are alive" cannot be expressed in propositional logic. Predicate logic is more expressive and inherits the qualities of propositional calculus. It is complete, meaning that for every well formed formula (wff), P which is true, P can be derived using only the rules of inference. It is sound, meaning that it is impossible to derive a contradiction, ie. for any wff P you cannot derive P and not P. It is decidable, meaning that for every wff there is a means with which to prove or disprove it.

Predicate logic introduces sets of variables, constants, predicates and functions. Predicates represent properties and relations of things. Functions represent operations with things, such as math operations where the things are numbers.

The language Prolog utilizes predicate logic with Horn clauses and a backward chaining inference engine. Horn clauses have a single conclusion and any number of premises. Programming in Logic (PROLOG) may not solve all your problems though. The inferencing method, pattern matching and conflict resolution provided may not fulfill your needs. These must be taken into consideration when writing rules for a rule based program, or the program may not perform anything like you expected it to.

2.4 Inferencing Methodologies

Rules for a knowledge-based system can be driven forward or backward, which relates to the terms forward chaining and backward chaining respectively. Both methods are equally powerful methods of reasoning, but one may be more useful than the other for the particular application at hand.

Forward chaining is a data driven method of reasoning. Here, the inferencing mechanism attempts to match the premise of each rule to the contents of working memory and thus determine if the premise is true, false or unknown. If the premise is unknown, the data in working memory is as yet incomplete for this rule, the next rule is tried. If the premise is false, the rule is ignored and may be removed from further inference. If the premise is true, the rule is fired and whatever actions are in the conclusion are taken, and the rule is removed from further inference. Inferencing continues until a conclusion is drawn or until no further rules can be fired and no conclusion is drawn. To come to a final conclusion it may not be necessary to fire all rules that are eligible, this depends on the setup of the system and whether more than one conclusion is desired.

Backward chaining on the other hand, is a goal driven inferencing technique. The system assumes a final goal and then attempts to prove it with the data available. The conclusions of rules are matched, then the conditions (premises) needed to be satisfied in order for the rule to fire are matched against the conclusions of other rules or data in working memory. The inferencing may stop with the first goal satisfied, or multiple goals may be derived depending again on the nature of the problem.

2.5 Suitability of a Knowledge-Based System

Knowledge-based systems have limits and will not solve all problems. Although a problem may be solvable by a knowledge-based system, its application to this problem may not be suitable or feasible. When considering their application to the domain of automatic control, a knowledge-based system may be suitable where the complete mathematical specification or modeling of the process is not feasible [SHI87]. Figures 2.2, 2.3 and 2.4 depict further items which should be satisfied for the development of a knowledge-based system to be possible, justified and appropriate [ARC87].

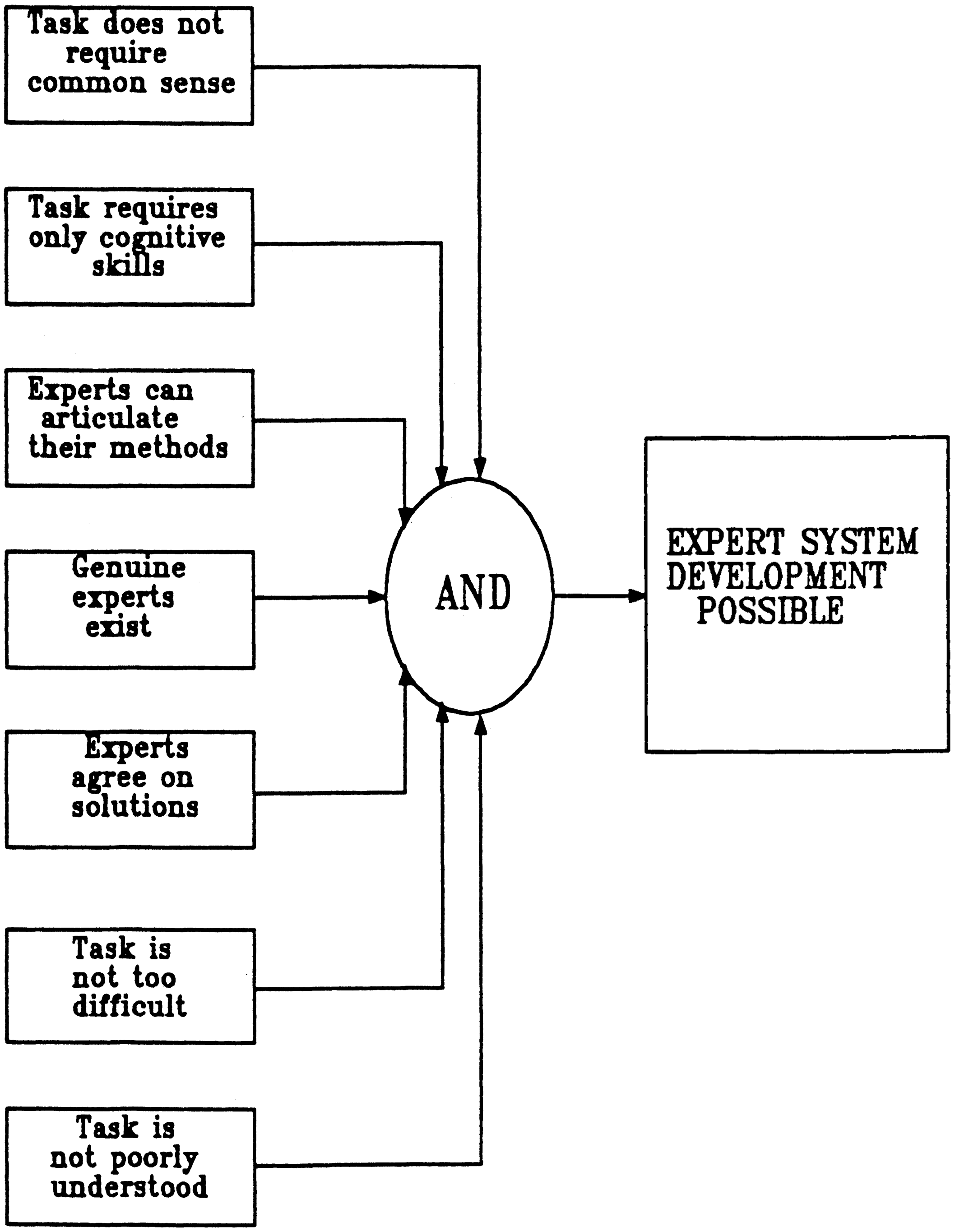

Task does not require common sense
Task requires only cognitive skills
Experts can articulate their methods
Genuine experts exist
Experts agree on solutions
Task is not too difficult
Task is not poorly understood
AND
EXPERT SYSTEM DEVELOPMENT POSSIBLE

Figure 2.2 When is Expert System Development Possible

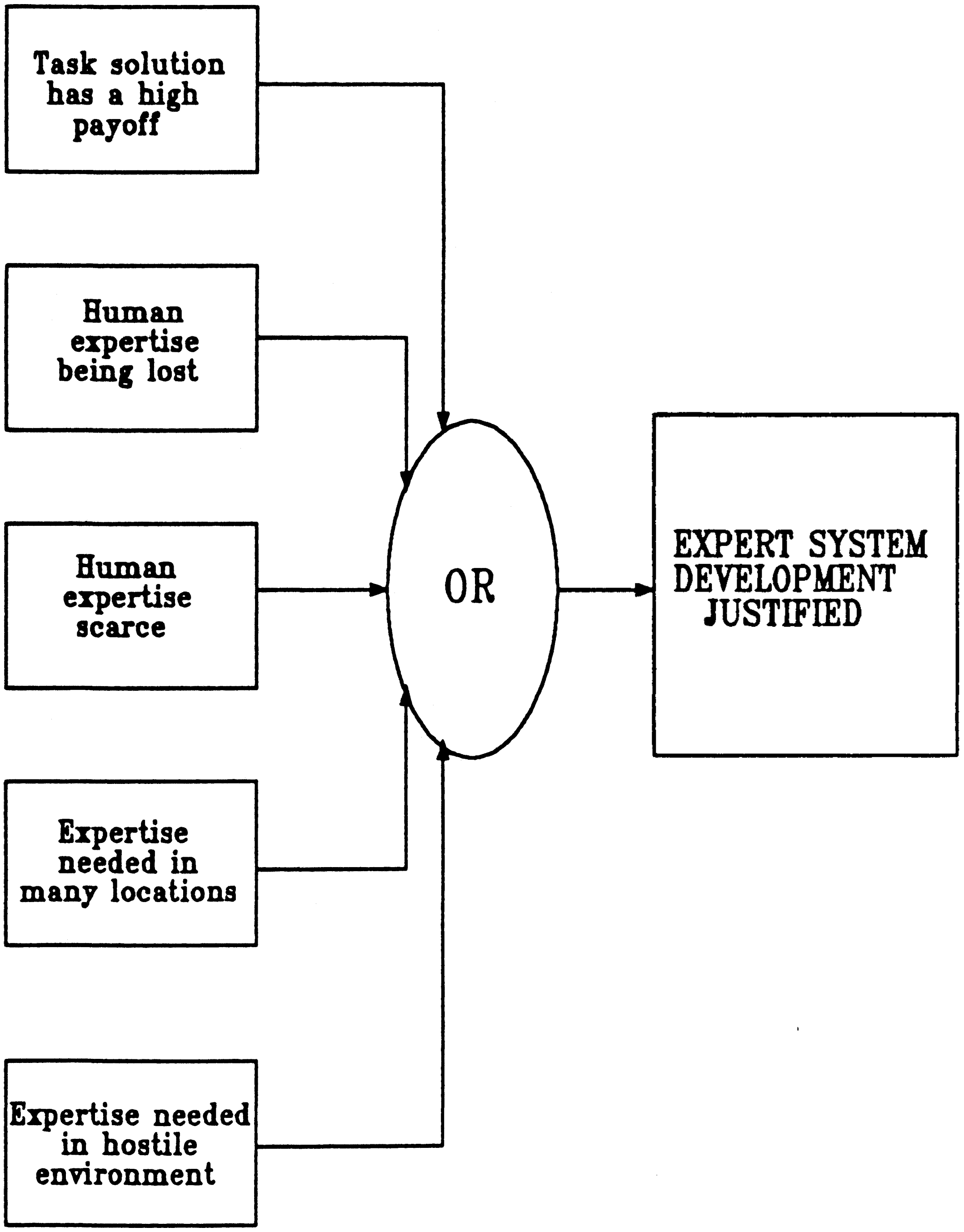

Task solution has a high payoff
Human expertise being lost
Human expertise scarce
Expertise needed in many locations
Expertise needed in hostile environment
OR
EXPERT SYSTEM DEVELOPMENT JUSTIFIED

Figure 2.3 When is Expert System Development Justified

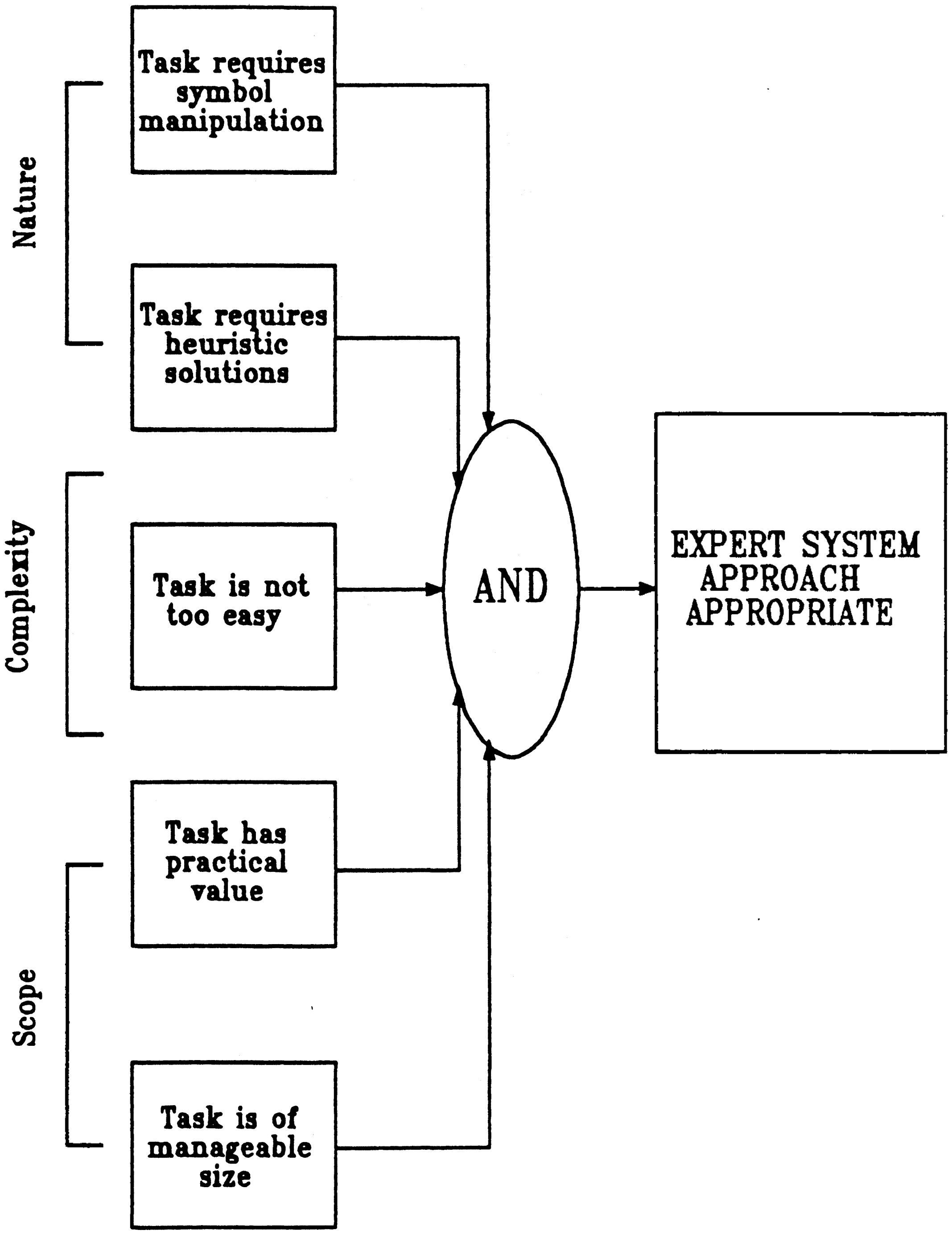

Nature
Complexity
Scope
Task requires symbol manipulation
Task requires heuristic solutions
Task is not too easy
Task has practical value
Task is of manageable size
AND
EXPERT SYSTEM APPROACH APPROPRIATE

Figure 2.4 When is Expert System Development Appropriate

CHAPTER 3
FUZZY LOGIC

3.1 Introduction

A knowledge-based system has the ability to emulate the method of reasoning a human uses. The variables and constants are all set to just one value, either a numerical value, a string or just true or false. The ability to deal with vagueness and uncertainty which may be expressed in the subject may also be needed. More than one statement may have some truth (or falsehood) to it, yet none is completely correct (or false). With fuzzy logic a partial truth or vague relation or membership can be expressed.

A classical example is the degree to which someone belongs to the set of "young" people. A person 20 years old may be considered 90 percent young, while a 60 year old person may be considered as only 30 percent young. There is no finite dividing line in the set, only some predefined gradual shift in the degree of membership in a class or set. Usually the degree of membership in a fuzzy set ranges between 0 and 1, 1 indicating absolute certainty. Depending on the use and definition, the range of -1 to 1 may be used. This allows for a degree of membership of 1 to indicate absolute certainty of membership, -1 for absolute certainty of non-membership and 0 to indicate unknown. The actual numerical values for the degree of membership may also be scaled depending on how they are to be dealt with, but they will still indicate the same idea. The advantage of using the scale of 0 to 1 is that fuzzy operators can be used in conjunction with probability operators. Probability operators can be used when the fuzzy operators do not provide a satisfactory means of combining evidence of, or confidence in, a value.

Fuzzy logic defines a minimum set of operations for dealing with certainty of membership in a set. These basic building blocks should fill the needs of most applications and using this as a standard, they can be built upon for unusual applications. Fuzzy logic is a super-set of predicate logic, therefore non-fuzzy information can be handled with the same operators as used to handle fuzzy information.

The three basic operators used in this thesis are the fuzzy AND, the fuzzy OR and the unary operator NOT.

3.2 Fuzzy Set Theory

Fuzzy set theory allows an element to exhibit a degree of membership in a given set. Conventional set theory only allows an element to be a member of a set or not. Because of this difference, the three basic set operators, union, intersection and complement are defined somewhat differently. The following paragraphs define these operations and how they are applied to fuzzy logic.

The union operator, when applied to two fuzzy sets with elements of differing degrees of membership, takes the respective elements with the greater degree of membership as the union of the two sets. If there are two sets, A and B, with the elements a_i and b_i respectively, then the union of the two sets can be defined as;

$$A \cup B = \{c_i \mid a_i \in A,\ b_i \in B,\ c_i = \max(a_i, b_i)\} \qquad (3.1)$$

where the max operator looks at the value of membership of the element.

With non-fuzzy values this is equivalent to the predicate logic OR. As an example, if a car is 80 percent slow and 20 percent accelerating, then it is 80 percent slow OR accelerating.

The intersection operator applied to two fuzzy sets with elements of differing degrees of membership, takes the respective elements with the lesser degree of membership as the intersection of the two sets. Again, using the two fuzzy sets A and B, the intersection of the two sets can be defined as;

$$A \cap B = \{c_i \mid a_i \in A,\ b_i \in B,\ c_i = \min(a_i, b_i)\} \qquad (3.2)$$

where the min operator looks at the value of membership of the element. This operation defines the fuzzy AND operator. Using the above example, the car would be 20 percent slow and accelerating.

The degree of membership of an element in the complement of a fuzzy set is 1 minus the degree of membership in the original set. Therefore, a car that is 80 percent slow, is only 20 percent not slow (or fast). This operation then, is equivalent to the predicate logic NOT and is the basis for the NOT operator in fuzzy logic. The complement of a set can be defined as;

$$\text{NOT } A = \{c_i \mid a_i \in A,\ c_i = 1 - a_i\} \tag{3.3}$$

where the minus operator looks at the value of membership of the element in the set.

3.3 Building a Fuzzy Set

Now that the basic operations of fuzzy logic have been defined, the fuzzy sets must be constructed. For the given domain the parameters to be used to characterize its state must be defined. Each parameter must first be identified, then the set of linguistic variables used for qualitative or quantitative description of its value must be defined.

Here it will be assumed that the parameters are identified by some conventional means, and will carry on from there to characterize them for use with fuzzy logic.

To characterize a parameter for fuzzy logic, it must be assigned membership values in a set, or sets. In general, these sets must be defined in advance and defined in the computer's vocabulary for the problem. Each parameter need not be assigned membership in all sets, as its absence will indicate a default membership value of zero. Using the range of -1 to 1 has an advantage here because non-membership indicates the parameter's value is unknown, equivalent to a value of zero.

For each parameter two things need to be known of it. First, as already mentioned, the range of values or linguistic terms it is to be mapped into and second, the nominal range of values it is to be mapped from, need to be specified. For example, if the height of a person is to be mapped to some fuzzy sets, both the fuzzy values and the range of heights that will be mapped into the sets must be defined. For the fuzzy sets the values short and tall might be selected. If the system is to deal with adults, tall may be people in the area of, or greater than, 6 feet, where as for children the height of 4 feet might be used.

The number of quantization levels selected, ie. the number of fuzzy sets, will depend on how accurately a designer wishes to describe a parameter. Some designers have used 15 or more levels, some as few as 3 [TON77]. The values to be mapped to these sets will also depend on the designer and the system they relate to.

Once these are known the method of quantization must be chosen and the rate at which each set rolls off between the nominal values of quantization. There are many examples of functions that can be used, some are shown in Figures 3.1 to 3.4 [KAU75]. In addition, a unique mapping may be selected for any particular set or parameter. The shape of the mapping function should first suit the assumptions of how the membership values for a set change between the nominal values for each set. Second, an appropriate function can be chosen for the convenience of coding and implementation. If, as in the case of this thesis the parameters are to be used in a knowledge-based system, the fuzzy sets should try to encompass the full flavour of the knowledge and linguistic terms to be encoded.

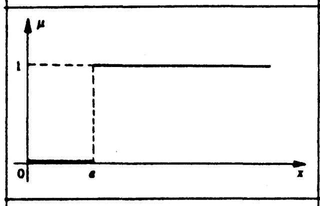
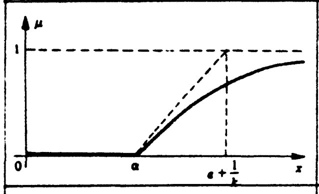
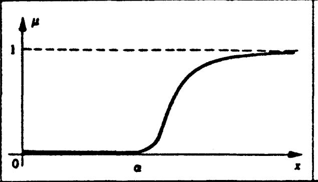
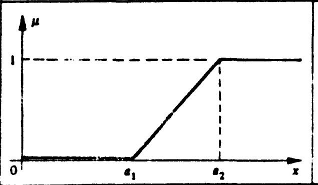
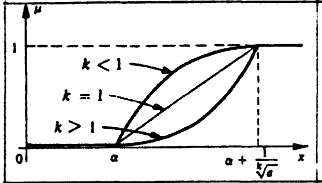
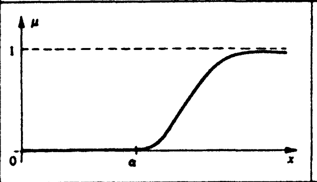
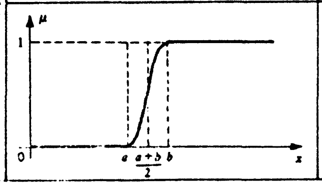
Curve	*Function*
	$\mu(x) = 0 \;,\; 0 \leq x < a \,,$ $= 1 \;,\; a \leq x \,.$
	$\mu(x) = 0 \;,\; 0 \leq x \leq \alpha \,,$ $= 1 - e^{-k(x-\alpha)}$ $\alpha \leq x \,.$ $k > 0 \,.$
	$\mu(x) = 0 \;,\; 0 \leq x \leq \alpha \,,$ $= 1 - e^{-k(x-\alpha)^2} \,,$ $\alpha \leq x \,,$ $k > 0 \,.$
	$\mu(x) = 0 \quad ,\; 0 \leq x \leq a_1 \,,$ $= \dfrac{x - a_1}{a_2 - a_1} \;,\; a_1 \leq x \leq a_2 \,,$ $= 1 \quad ,\; a_2 \leq x \,.$
	$\mu(x) = 0 \;,\; 0 \leq x \leq \alpha \;,$ $= a(x - \alpha)^k,$ $\alpha \leq x \leq \alpha + \dfrac{1}{\sqrt[k]{a}} \,,$ $= 1 \;,\; \alpha + \dfrac{1}{\sqrt[k]{a}} \leq x \,.$
	$\mu(x) = 0 \;,\; 0 \leq x \leq \alpha \,,$ $= \dfrac{k(x - \alpha)^2}{1 + k(x - \alpha)^2} \,,$ $\alpha \leq x < \infty \,.$
	$u(x) = 0 \;,\; 0 \leq x \leq a \,,$ $= \dfrac{1}{2} + \dfrac{1}{2} \sin \dfrac{\pi}{b - a}\left(x - \dfrac{a + b}{2}\right),$ $a \leq x \leq b \,,$ $= 1 \;,\; a \leq x \,.$

Figure 3.1 Membership Functions Corresponding to "x is Large"

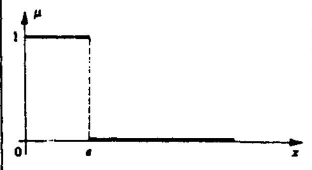
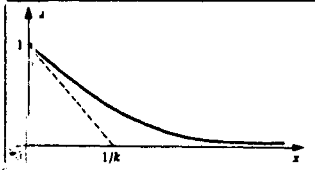
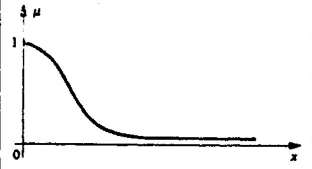
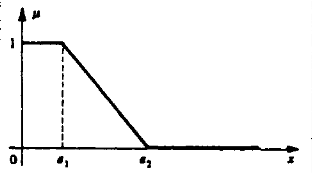
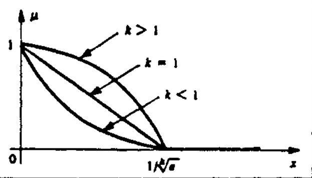
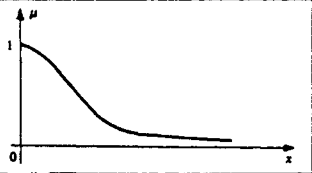
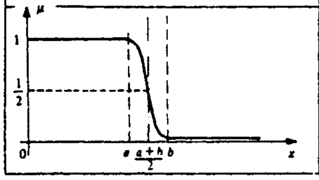
Curve	*Function*
	$\mu(x) = 1 \ , \ 0 \leqslant x \leqslant a ,$ $= 0 \ , \ x > a .$
	$\mu(x) = e^{-kx} \ , \ k > 0 .$
	$\mu(x) = e^{-kx^2} \ , \ k > 0 .$
	$\mu(x) = 1 \ , \ 0 \leqslant x \leqslant a_1 ,$ $= \frac{a_2 - x}{a_2 - a_1} \ , \ a_1 \leqslant x \leqslant a_2 ,$ $= 0 \ , \ a_2 \leqslant x .$
	$\mu(x) = 1 - ax^k \ , \ 0 \leqslant x \leqslant \frac{1}{\sqrt[k]{a}} ,$ $= 0 \ , \ \frac{1}{\sqrt[k]{a}} < x .$
	$\mu(x) = \frac{1}{1 + kx^2} ,$ $k > 1 .$
	$\mu(x) = 1 \ , \ 0 \leqslant x \leqslant a ,$ $= \frac{1}{2} - \frac{1}{2} \sin \frac{\pi}{b - a} \left(x - \frac{a + b}{2} \right) ,$ $a \leqslant x \leqslant b ,$ $= 0 \ , \ b \leqslant x .$

Figure 3.2 Membership Functions Corresponding to "x is Small"

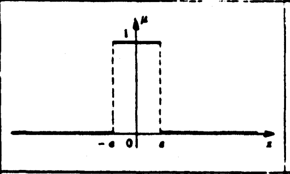
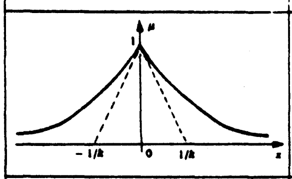
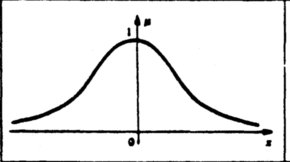
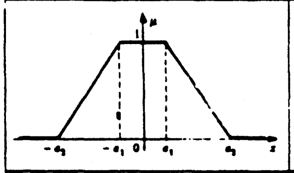
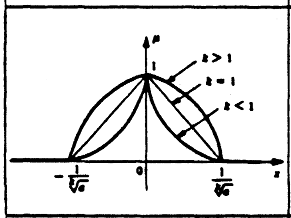
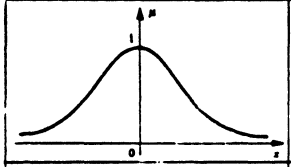
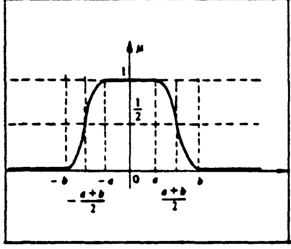
Curve	*Function*
	$\mu(x) = 0 , \; -\infty < x < a ,$ $= 1 , \; -a \leqslant x \leqslant a ,$ $= 0 , \; a \leqslant x .$
	$\mu(x) = e^{kx} , \; -\infty < x \leqslant 0 ,$ $= e^{-kx} , \; 0 \leqslant x < \infty ,$ $k > 1 .$
	$\mu(x) = e^{-kx^2} .$
	$\mu(x) = 0 , \; -\infty < x \leqslant -a_2 ,$ $= \frac{a_2 + x}{a_2 - a_1} , \; -a_2 \leqslant x \leqslant -a_1 ,$ $= 1 , \; -a_1 \leqslant x \leqslant a_1 ,$ $= \frac{a_2 - x}{a_2 - a_1} , \; a_1 \leqslant x \leqslant a_2 ,$ $= 0 , \; a_2 \leqslant x < \infty .$
	$\mu(x) = 0 \quad -\infty < x \leqslant -\frac{1}{\sqrt[k]{a}}$ $= 1 - a(-x)^k , \; -\frac{1}{\sqrt[k]{a}} \leqslant x \leqslant 0 ,$ $= 1 - a(x)^k , \; 0 \leqslant x \leqslant \frac{1}{\sqrt[k]{a}} ,$ $= 0 , \; \frac{1}{\sqrt[k]{a}} \leqslant x < \infty .$
	$\mu(x) = \frac{1}{1 + kx^2} ,$ $k > 1 .$
	$\mu(x) = 0 , \; -\infty < x \leqslant -b ,$ $= \frac{1}{2} + \frac{1}{2} \sin \frac{\pi}{b - a}\left(x + \frac{a + b}{2}\right),$ $-b \leqslant x \leqslant -a ,$ $= 1 , \; -a \leqslant x \leqslant a ,$ $= \frac{1}{2} - \frac{1}{2} \sin \frac{\pi}{b - a}\left(x - \frac{a + b}{2}\right),$ $a \leqslant x \leqslant b ,$ $= 0 , \; b \leqslant x < \infty .$

Figure 3.3 Membership Functions Corresponding to "|x| is Small"

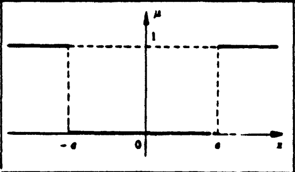
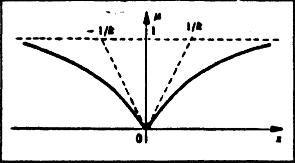
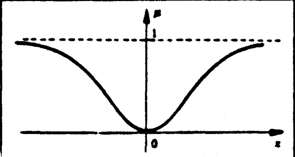
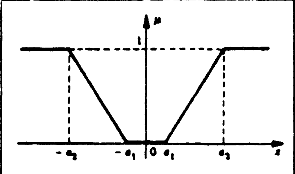
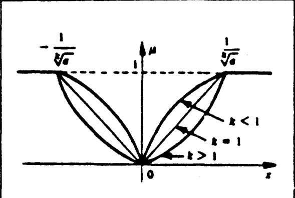
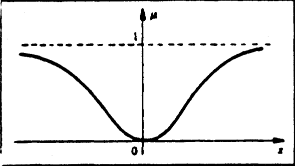
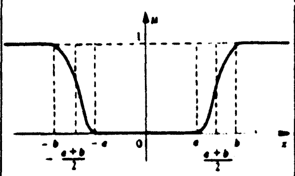
Curve	*Function*
	$\mu(x) = 1\ ,\ -\infty < x < -a\ ,$ $= 0\ ,\ -a < x < a\ ,$ $= 1\ ,\ a < x < \infty\ .$
	$\mu(x) = 1 - e^{kx}\ ,\ -\infty < x \leq 0\ ,$ $= 1 - e^{-kx}\ ,\ 0 \leq x < \infty\ ,$ $k > 1\ .$
	$\mu(x) = 1 - e^{-kx^2}\ ,$ $k > 1\ .$
	$\mu(x) = 1\ ,\ -\infty < x < -a_2\ ,$ $= -\frac{x + a_1}{a_2 - a_1}\ ,\ -a_2 \leq x \leq -a_1\ ,$ $= 0\ ,\ -a_1 \leq x \leq a_1\ ,$ $= \frac{x - a_1}{a_2 - a_1}\ ,\ a_1 \leq x \leq a_2\ ,$ $= 1\ ,\ a_2 \leq x < \infty\ .$
	$\mu(x) = 1\ ,\ -\infty < x < -\frac{1}{\sqrt[k]{a}}\ ,$ $= a(-x)^k\ ,\ -\frac{1}{\sqrt[k]{a}} \leq x \leq 0\ ,$ $= ax^k\ ,\ 0 \leq x \leq \frac{1}{\sqrt[k]{a}}\ ,$ $= 1\ ,\ \frac{1}{\sqrt[k]{a}} \leq x < \infty$
	$\mu(x) = \frac{kx^2}{1 + kx^2} = \frac{1}{1 + \frac{1}{kx^2}}\ ,$ $k > 1\ .$
	$\mu(x) = 1\ ,\ -\infty < x \leq -b\ ,$ $= \frac{1}{2} - \frac{1}{2} \sin \frac{\pi}{b - a}\left(x + \frac{a + b}{2}\right),$ $-b \leq x \leq -a$ $= 0\ ,\ -a \leq x \leq a\ ,$ $= \frac{1}{2} + \frac{1}{2} \sin \frac{\pi}{b - a}\left(x - \frac{a + b}{2}\right),$ $a \leq x \leq b\ ,$ $= 1\ ,\ b \leq x < \infty$

Figure 3.4 Membership Functions Corresponding to "|x| is Large"

CHAPTER 4

CHOICE OF PROCESS FOR TESTING

4.1 Introduction

The intent of this thesis has been to study the application of knowledge-based systems and fuzzy logic to control problems. In order to test the ideas, they must be applied to at least one system. This chapter describes the criteria for selecting a test process, the process that was selected and the model derivation for simulation.

4.2 Criteria

The process selected had to satisfy a few simple criteria. First, it had to be a system with some recognizable interest to the study of automatic control. Second, the problem of controlling the process had to be solvable with a knowledge-based system and reasonably appropriate for the application of a knowledge-based system as presented in Chapter 2. Finally, it had to be a problem to which the author could apply himself with a minimum of outside assistance in gathering the knowledge and building a model for simulation.

Multiple Input Multiple Output (MIMO) processes can be difficult to model and therefore designing controllers for them can be an arduous task. In part this is due to the increase in the size of the process, and therefore an increase in the complexity. It can also be due to cross coupling between the inputs of the process and its outputs. This provides an interesting area for the application of a knowledge-based controller where the heuristic's of an experienced human controller can be applied to handle this situation.

Also of interest to the author, for the field of automatic control, is the design of a generic controller. Of interest here is the ability to control a class of processes, not just a specific process, or even just one example of a process. For example any human experienced with one automatic coffee maker can operate pretty well any automatic coffee maker. There are a general set of rules for operating such a device and in addition some specific facts may need to be known, or they may be induced. All processes in a class are intended to achieve the same end result, yet internally they may go about it in slightly different ways.

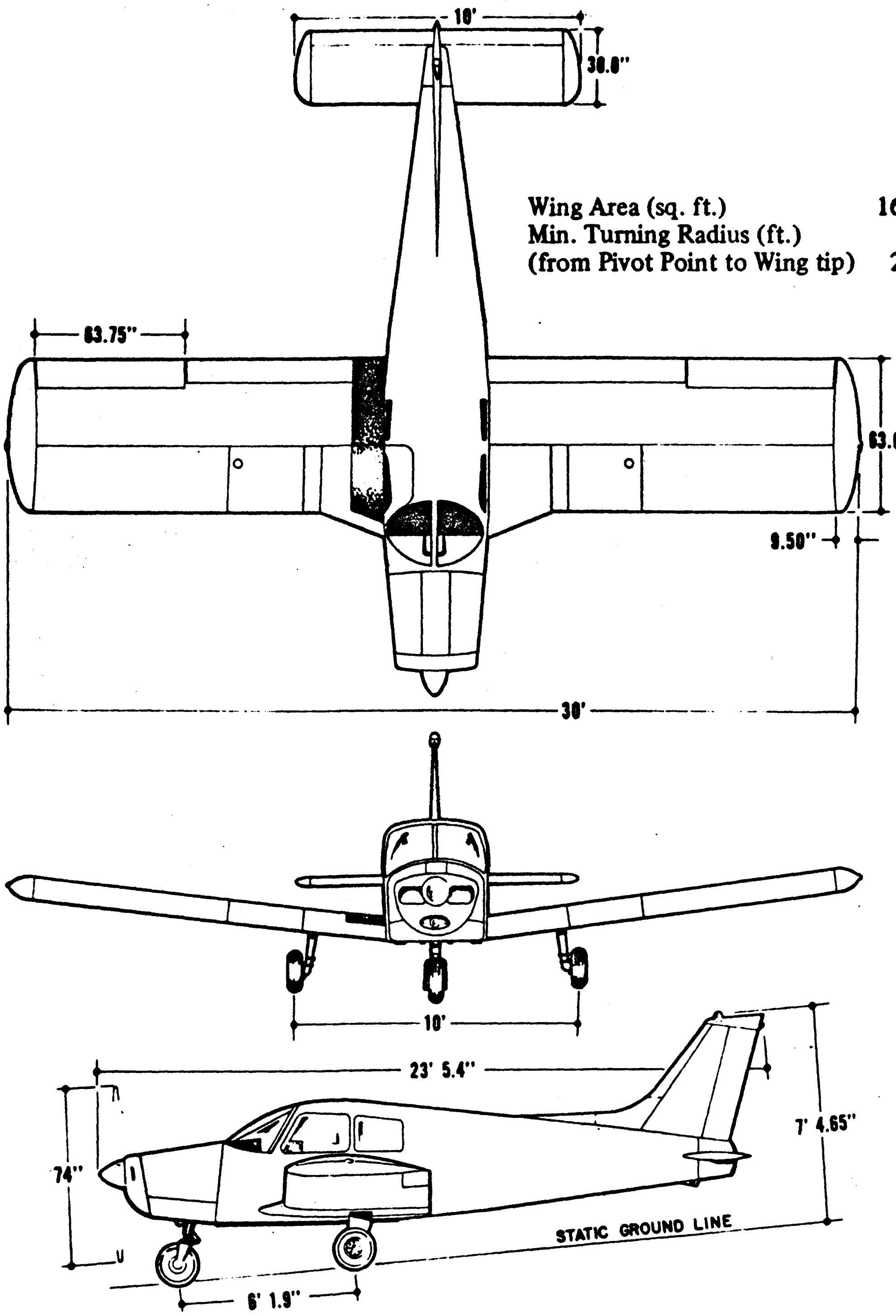
The criteria to be used when deciding whether or not a problem is a candidate for the application of a knowledge-based system were introduced in Chapter 2. For the purposes of study for this thesis, some of the criteria were considered to be more important than others. Most important was selecting a process for which recognized "experts" are needed to control it and for which a set of rules could be gathered which would be generally accepted as correct. It also had to be complex enough so that it wasn't just the implementation of a simple proportional controller. The interest is to encounter the problems and advantages associated with the development of a knowledge-based controller.

In order to make the acquisition of knowledge easier and faster, it would be advantageous if a process could be selected for which the author could act as the expert, as well as knowledge engineer, coder and tester. Possibly most important, was the selection of a process which would hold the interest of the author for as long as possible.

4.3 <u>Description of Process Chosen</u>

The process chosen for this application test was a light aircraft. The aircraft on which the simulation is based is a Piper Cherokee, Model PA-28, manufactured by the Piper Aircraft Corp, shown in Figure 4.1. This is a four place personal / training aircraft which is stable and not difficult to fly. It does require all the basic pilot skills and knowledge for flying (controlling) an aircraft and is easily modeled for the purpose of simulation to the requirements of this study. Configuration and performance data are available [CHE76] and the capabilities and performance of the aircraft are well understood by the author.

For this thesis, the problem considered is the



Wing Area (sq. ft.)	160.0
Min. Turning Radius (ft.) (from Pivot Point to Wing tip)	25.0

Figure 4.1 PA-28-140 Cherokee Cruiser
Piper Aircraft Corporation

control of power and the pitch axis of the aircraft. There are two inputs, throttle and elevator position, to control five aspects of the aircraft's flight. Normally two of the following are primary instruments for controlling the aircraft. These are the engine rpm or power output, airspeed, altitude, rate of climb and angle of pitch. Rpm and airspeed are candidates for power control while altitude, angle of climb, airspeed and angle of pitch are candidates for pitch control.

This problem satisfies all the criteria specified above to some extent. It is a MIMO process with some cross coupling of the inputs to the outputs. It also holds that any licensed pilot can fly any light airplane once he is given basic facts on the aircraft's performance and capabilities. Pilots are essentially recognized experts at a skill and the rules which apply to controlling an airplane can be expressed.

The author has been involved with aircraft for many years and is a licenced commercial pilot. This process then obviously should satisfy the criteria of interest. To supplement the author's flying knowledge many training manuals are available which express the knowledge required for flying [FRO63], [FLI79], [PRI80].

The process was limited to the control of power and pitch to limit the complexity of simulating the process, to keep to a time frame acceptable for a Master's thesis, to keep the problem tractable in a limited development environment, ie. hardware and software, and to leave room for experimentation.

4.4 Simulator Development

This section presents the development of the equations used for simulating the aircraft. Simplifying assumptions and the final equations are represented while the actual parameter values used are given in Appendix A.

4.4.1 Simulation Of Engine Thrust

A very simple equation was first developed to simulate the speed of rotation of the propeller, engine rpm, for an early prototype controller. The equation assumed a simple first order model with a one second time constant. The equation used was

$$\mathrm{RPM}\ (s) = \frac{1}{s + 1} \tag{4.1}$$

which in the discrete time domain is

$$\begin{aligned} \mathrm{rpm}(kt) = {} & \mathrm{deltaT} * \mathrm{mK}(kt) \\ & + \mathrm{rpm}((k-1)\ t) * \exp\ (-\mathrm{deltaT}). \end{aligned} \tag{4.2}$$

This model was used to test a simple knowledge-based controller and evaluate some concepts.

This model has remained, except for the addition of a factor to simulate induced and parasitic drag of the propeller on the engine rpm. The factor is multiplied by input signal (mK) to account for the observed relationship of decreased rpm for a given power setting when the airspeed of the aircraft is below the nominal cruise value and an increase in rpm (unloading of the engine) when the airspeed is high. The nominal cruise velocity is a simple equation relating rpm, propeller pitch and an efficiency factor for the propeller [AER86]. This was done instead of calculating the actual drag on the propeller and therefore the engine loading because the engine torque was not being modeled and rpm was a required value. These equations have provided a satisfactory simulation for the purposes of this study. For calculating engine rpm the following equations are used,

$$\text{nominal cruise} = Vnom(t) = pitch * rpm(t) * e \tag{4.3}$$

$$Vdiff(t) = V(t) - Vnom(t) \tag{4.4}$$

$$\begin{aligned} rpmDamp(kt) = 1.0 + (&\text{signof } Vdiff(t) * 0.075 \\ &* (1.0 - \exp(-0.01 * Vdiff(t))) \end{aligned} \tag{4.5}$$

$$\begin{aligned} rpm(kt) = deltaT &* mK(kt) * rpmDamp(kt) \\ &+ rpm((k-1)\ t) * \exp\ (-deltaT) \end{aligned} \tag{4.6}$$

Using the rpm value calculated above the brake horsepower (bhp) is calculated using a simple relationship between rpm and bhp. A quadratic relationship is used and the following equation was derived from two known values,

$$\begin{aligned} bhp(t) = (1.68E\text{-}7 &* (rpm(t))^2 \\ &- 8.4E\text{-}5 * rpm(t)) * rhp \end{aligned} \tag{4.7}$$

Using a relationship between bhp and thrust [AER86], the propeller thrust can be calculated in cruise flight. Adding a term to account for decreasing thrust with increasing airspeed for a constant rpm, the following equations are used to give a thrust coefficient,

```
  damp(t) = 1.0 - ((V(t) / 182) * 0.53) / V(t)          (4.8)
  thrustCoeff(t) =
            bhp(t) * 550 * 0.85 * damp / weight         (4.9)
```

The above equations provide an rpm value needed for the simulator output and in a "fuzzy way", a thrust coefficient needed in the following section in the equations of flight.

4.4.2 Simulation Of Longitudinal Motion Of Aircraft

For those readers feeling a little faint due to the last sections presentation, the equations of motion presented here have a much more rigorous basis than the equations for the derivation of the thrust and rpm. In a reference frame with its origin at the centre of mass related to the flight path, the motion equations for the longitudinal axis will be;

$$m \frac{dV}{dt} = - D - mg \sin \delta + T \cos (\alpha - \sigma) \qquad (4.10)$$

$$\frac{d\alpha}{dt} = -\frac{d\delta}{dt} + q \qquad (4.11)$$

$$m V \frac{d\delta}{dt} = L - mg \cos \delta + T \sin (\alpha - \sigma) \qquad (4.12)$$

$$B \frac{dq}{dt} = M - T z_r + M_c \qquad (4.13)$$

where m, D, L, M are known functions of the state variables and of t [HAC70], [SEC64].

The aerodynamic forces D and L and the longitudinal aerodynamic moment M, can be expressed in terms of non-dimensional coefficients which is the usual form,

$$D = pV^2 SC_D/2 \qquad (4.14)$$

$$L = pV^2 SC_L/2 \qquad (4.15)$$

$$M = pV^2 S1C_M/2 \qquad (4.16)$$

After substituting the non-dimensional terms into the equations of motion, the constants can be collected and the equations simplified. The following equations can be derived for an aircraft with the thrust vector fixed parallel to the zero-lift reference axis of the aircraft. Thus using $\ddot{Y}$ and K as the control variables, the following are obtained

note: $T = mg\tau$, and $M_c/B = K$,

$$\frac{dh}{dt} = V \sin \delta = f_1(V, \delta) \qquad (4.17)$$

$$\frac{dV}{dt} = -(k_2 + k_3\alpha^2)V^2 - g \sin \delta + g\tau \cos \alpha$$
$$= f_2(V, \delta, \alpha, \tau) \qquad (4.18)$$

$$\frac{d\delta}{dt} = k_5 V\alpha - \frac{g \cos \delta}{V} + \frac{g\tau \sin \alpha}{V} = f_3(V, \delta, \alpha, \tau) \qquad (4.19)$$

$$\frac{d\alpha}{dt} = -\frac{d\delta}{dt} + q = f_4(V, \delta, \alpha, q, \tau) \qquad (4.20)$$

$$\frac{dq}{dt} = V^2(k_8\alpha + k_9 V + k_{10}\alpha f_4) + K(V, \alpha, \delta_e, \delta_r)$$
$$= f_5(V, \delta, \alpha, q, \tau, \delta_e, \delta_r) \qquad (4.21)$$

$$k_2 = \frac{pSC_D}{2m}, \quad k_3 = \frac{pSC^2_{L\alpha}}{2\pi\lambda m}, \quad k_5 = \frac{pSC_{L\alpha}}{2m},$$

$k_8 = \frac{pS\bar{c}C_m}{8B}$, $k_9 = \frac{pS\bar{c}^2 C_{mq}}{8BV_{crit}}$, $k_{10} = \frac{pS\bar{c}^2 C_{m\dot{\alpha}}}{8BV_{crit}}$, [HAC70].

Data was unavailable in the form necessary for equation (4.21) so another expression for acceleration in pitch was used. This equation uses the sum of moments contributing to the overall pitching moment of the aircraft. These effects are from the displacement of the lift vector relative to the centre of gravity, the aerodynamic pitching moment of the wing, the pitching moment of the fuselage and the pitching moment due to elevator deflection. The new equation is as follows,

$$B \frac{dq}{dt} = M_{ya} = pV^2 C\bar{c}Cm_{ya} /2 \qquad (4.22)$$

where

$$Cm_{ya} = C_L \underline{x}_a + Cm_{ac} + Cm_t + Cm_f \qquad (4.23)$$

[SEC64].

C_L and Cm_{ac} are obvious coefficients and easy to determine. The coefficients Cm_f and Cm_t need further explanation. First, Cm_f the fuselage coefficient is expressed as follows

$$Cm_f = Cm_{\alpha f} * (\alpha - i_w) \qquad (4.24)$$

where

$$Cm_{\alpha f} = 2K_f \frac{Vol_f}{S\bar{c}} \qquad (4.25)$$

K_f is a fuselage moment factor, see Figure 4.2, and

Vol_f is the fuselage volume. An estimation for the fuselage volume is given in [SEC64] and is shown below,

$$Vol_f = 0.25\pi\ [w_3^2/24 * (l_f - l_3) + 2.3w_2^2 * (l_2 - l_1) + 1.2*\int_0^{l_1} w^2\ dl] \qquad (4.26)$$

The various dimensions are illustrated in Figure 4.3.

The pitching moment coefficient due to the tail is approximated with the following equation.

$$Cm_t = Cm_{it}\ (C_L/a_w + \alpha_{w0} + (i_t - i_w) - \epsilon + \tau\delta_e) \qquad (4.27)$$

The last term represents the change in the tails zero lift angle due to elevator deflection. The PA-28 aircraft has a flying stabilizer and therefore no elevator so this term is dropped. The pitching moment for the tail incidence is calculated as follows

$$Cm_{it} = - n_t\ V_{at} \qquad (4.28)$$

n_t is the tail efficiency, while V_{at} is the tail volume which is determined by

$$V_{at} = \frac{l_t}{\bar{c}} \frac{S_t}{S} \qquad (4.29)$$

ϵ is the down-wash effect of the airflow from the wing.

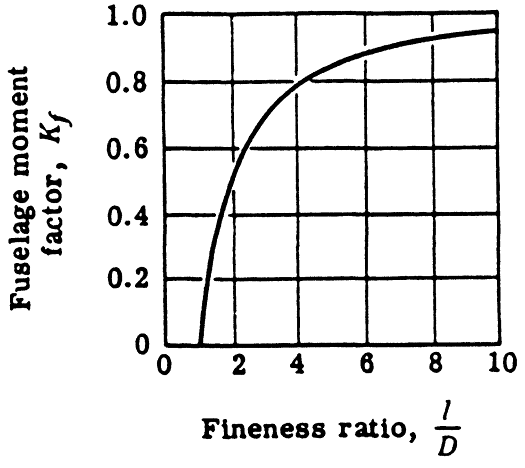


Figure 4.2 Fuselage Moment Factor

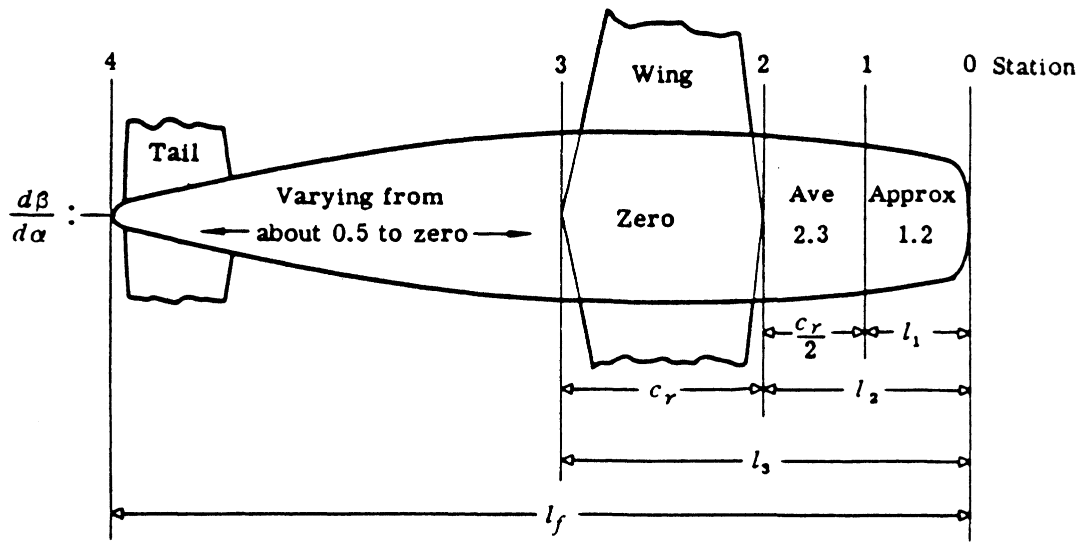


Figure 4.3 Approximate Interference Factors

An approximate and sufficient solution to the above differential equations are given below, assuming a sufficiently small time step is used.

$$h = h^1 + \frac{dh}{dt} * \text{deltaT} \qquad (4.30)$$

$$V = V^1 + \frac{dV}{dt} * \text{deltaT} \qquad (4.31)$$

$$\delta = \delta^1 + \frac{d\delta}{dt} * \text{deltaT} \qquad (4.32)$$

$$\alpha = \alpha^1 + \frac{d\alpha}{dt} * \text{deltaT} \qquad (4.33)$$

$$q = q^1 + \frac{dq}{dt} * \text{deltaT} \qquad (4.34)$$

4.5 Summary

This chapter has given a complete presentation of the process to be used in the development and evolution of the knowledge-based controller. The criteria for the process selection were presented, followed by the selection of a suitable process and the derivation of a set of equations such that a satisfactory simulation of the process could be done.

The simulator was coded as a separate class in Smalltalk/V. This allows for easy creation of an object, a simulated aircraft, to which messages can be sent to affect simulation parameters, process inputs and the length of time the simulation is to be run until the next sensor reading. A time step of 0.05 seconds was chosen. This is small enough that the simulator does not behave erratically and allows for reasonable execution time. Although the simulation is based on the PA-28-140, due to its simplicity it is not highly accurate, however its performance is representative of this class of aircraft as was desired.

CHAPTER 5

CHOICE OF DEVELOPMENT TOOLS

5.1 Introduction

This chapter presents the criteria for selecting the software development tools for this thesis. The software tools used are discussed based on their power, expressiveness and suitability to the tasks they were applied to. An expert systems shell by Texas Instruments called PC Plus was used for some early prototypes and later abandoned for PROLOG/V, a class in Smalltalk/V by digitalk Inc.

5.2 Software Criteria

The architecture the software was to run on was the first limiting factor. The software chosen had to run on an IBM compatible machine. An 80386 based machine with 4 megabytes of RAM was available. It was also intended that some development be done on a XT clone with 640 kilobytes of RAM. Therefore the first criterion was stipulated by the available hardware, it had to run on an IBM or compatible machine.

With hardware limitations in mind, a development environment or set of tools was needed conducive to building knowledge-based systems and of being able to perform tests using a simulator implemented in software. For the simulator a procedural language was considered to be desirable, yet for the knowledge-based controller a declarative language was needed. For the knowledge-based controller an expert systems shell would also suit and possibly offer other advantages.

5.2.1 Simulator Criteria

The selection of a tool for the simulator was not a large obstacle. It was preferred that a procedural language such as C, Pascal or Smalltalk be chosen as they are well suited to this type of task. In the simulator there is no requirement for manipulating symbolic data and the input and output would be straight forward and well defined. The main concern here was the ability for the language chosen to interface with the knowledge-based controller.

5.2.2 Knowledge-based System Criteria

For the knowledge-based system a language or tool that can deal with symbolic information and could incorporate fuzzy logic into it was required. The speed at which it executed was of secondary concern to these requirements as it would only be controlling a simulated process. Of greater concern than execution time was the development effort that would be required. Starting from scratch with a language such as LISP would require the development of the knowledge representation scheme and the inferencing mechanism. An expert system shell would provide these, but it may not suit the needs of the system correctly and could require that too many tradeoffs be made.

It was desirable to keep the inferencing method and the knowledge representation scheme flexible. For inferencing, both backward and forward chaining were desirable because each may fit a particular sub-problem better than the other. For the knowledge representation, frames allow the exploitation and use of any naturally occurring structure in the problem domain, yet rules are a simple and quite natural way to express knowledge.

The availability and support of the tool selected was the next concern. The purchase of a development tool for one specific project may not be justifiable. Also the purchase of a language compiler or interpreter can save money over a shell, but may increase the cost dramatically in development time. The availability of complete and informative support can greatly ease development, and can therefore be a major concern.

In summary, the criteria are the expressiveness and flexibility of the tool, the amount of development time anticipated, the support available and the availability of the tool itself.

5.3 Personal Consultant Plus

PC Plus is an expert system development shell for which McMaster University obtained a site licence from Texas Instruments. That solved the problem of availability and along with the site licence came support, both on campus and from Texas Instruments. This turned out to be difficult to get at times and the documentation was difficult to follow. PC Plus is written in TI Scheme, a dialect of LISP, and provides many aides to expert system developers as well as external language interface well suited to use with C. Knowledge is represented in PC Plus in a combination of frames and rules and PC Plus supposedly supports both backward and forward inferencing. All parameters in PC Plus have a certainty factor attached to them and they are used in a fashion not incompatible with the fuzzy logic described earlier.

The first prototype contained fewer than 30 rules in 3 frames and seemed to work well. It merely had questions posed to it, to which it would give an answer based on a simple subset of rules for controlling an aircraft. The second prototype had about 150 rules in 19 frames, incorporated a fuzzifier to create fuzzy sets and was therefore much more complex but presented many problems. Forward chaining was attempted in the majority of the system but it was found that PC Plus did not support this in an acceptable fashion. The inheritance in the frames was found to be very awkward. Instead of searching for and resolving parameters in the current frame, it would look to the parent frames first. These problems could have been dealt with, but there were additional problems related to hardware.

PC Plus required at least 2 megabytes of expensive memory and the speed of an 80386 based machine to be workable. There was only one such machine available which had to be time shared with other users and it was out of service too often to keep the work progressing smoothly. It was becoming apparent that the development of the knowledge-based controller was requiring the monopolization of the development system with the amount of prototype and test iteration required in entering and verifying the rules. The decision was made to look for an alternate tool.

5.4 Smalltalk/V and Prolog/V

After the trouble with PC Plus it was decided to try and stay away from shells and go with a more basic set of tools. The alternatives were to go with C as the procedural language, as it would have been with PC Plus, and then either LISP or Prolog for the knowledge-based controller. For these two languages, the original alternatives were TI Scheme or Turbo Prolog. Turbo Prolog was preferred in order to help save some development time but it had the draw backs of not supporting modular programming which helps to keep a large system organized nor did it easily support frames. TI

Scheme supported an object oriented environment which was interesting mainly due to its similarity to frames.

The other major concern was to be able to do most, if not all, of the development on an XT class computer to which there was unfettered access available. At this time a site licence was obtained from digitalk Inc. for Smalltalk/V. The shipped software contained an implementation of Prolog complete with source code and documentation. This combined two powerful programming paradigms, logic programming and object oriented programming. It appeared that the necessary development tools had been found.

There exists a very simple means of posing a Prolog question from Smalltalk/V and in Prolog/V a predicate is available so that messages may be sent to Smalltalk/V objects. The object oriented environment can be exploited in Prolog/V to provide a frame construct. The knowledge can be grouped by function into objects, which inherit the knowledge and data of their parents as well as contain their own.

This environment allowed more freedom in setting up the knowledge based controller and in the knowledge representation used. Fuzzy logic was easily implemented by defining special predicates in a parent class to the controller. A very big plus of this environment is the ease of testing. Any expression can be evaluated at any time in the Smalltalk/V environment, including Prolog questions.

This made testing and verification of the code very easy. The debugger and inspectors provided allow tracing of a problem with great ease. Prolog/V is not a full implementation of Prolog, but whatever is lacking can be easily added by modifying the class or whatever is needed may already be provided in Smalltalk/V.

CHAPTER 6
CONTROL SYSTEM DESIGN

6.1 Introduction

This chapter discusses the knowledge-based controller and its associated components. The ideology behind the control system is discussed introducing the methods of a human controller, a pilot, in controlling the process, an airplane. Next the instrument reading scheme is introduced which leads into the following section covering the implementation of fuzzy logic in this thesis. The fuzzification of the sensor readings is discussed as well as defuzzifying the output set. Lastly the structure used for the controller, including the knowledge representation and the layout of the objects or frames in the system and their interaction is discussed.

6.2 Control Method

This controller implements the knowledge used by a human operator to control a simulated process. The controller is designed to emulate what is perceived to be the methods employed by a human operator to enact this control. For this thesis the controller is essentially trying to control the flight path of an aircraft. The inputs to the controller specify a power set point and a pitch set point. The controller then decides, based on the aircraft's current state, how to achieve these set points and then proceeds to accomplish the task. Thus two main tasks are identified here, one is planning and the other is the execution of control.

There were originally five values being used to convey the state of the aircraft which corresponded to instruments a pilot uses. These five instruments can be grouped as to the type of information a pilot can get from them. The instruments which provide information about power are the tachometer and the airspeed indicator. The instruments which provide pitch information are the altimeter, the vertical speed indicator, pitch from the horizon (real or artificial) and the airspeed indicator as shown in Figure 6.1 [FLI79]. To these, acceleration and pitch rate were added because adequate control could not be realized without them. The pilot can sense this information either through feeling the motion or through the rate of indicator movement.

Within each of these groups one of the instruments will be considered primary for control purposes while the other(s) will provide backup information. In normal, steady state cruise flight the airspeed is primary for power control and altitude is primary for pitch control. The instrument that is primary depends on the flight condition

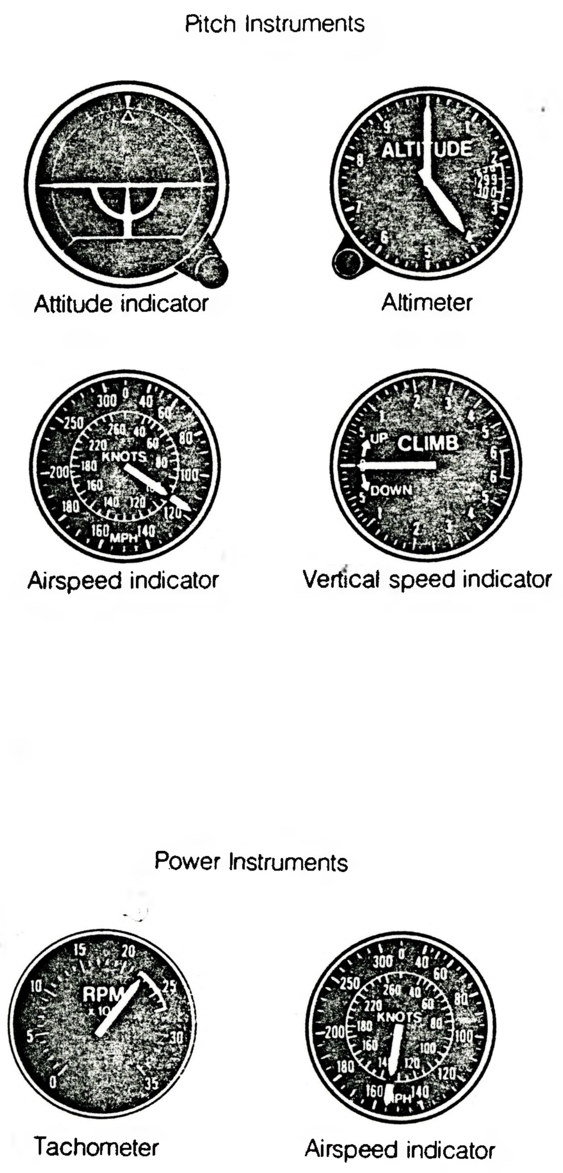


Figure 6.1 Flight Instruments

to be maintained, or achieved. If a fixed airspeed is to be maintained then the airspeed is normally primary for power, but it may also be used as the primary indicator for pitch control. As an example, in a descent it is often desirable to maintain a given airspeed for a given power setting, such as descending from altitude or on final approach to land.

Normally as long as the errors in the set points are not too large, the instrument related to the set point to be maintained can remain as the primary instrument. However, if the set point is changed by a large amount, or if the error should become large, then it is necessary to temporarily use another instrument as the primary control instrument to establish the aircraft at the desired set point. An example is in controlling power. If a large change in airspeed is necessary, the pilot will first make an estimate of the new power setting required and then establish the power at that value. The tachometer is now primary for power until the new power setting is established. When the tachometer is correct, the pilot will switch to the airspeed as primary and make corrections as necessary. This works similarly for control of pitch. It is obvious that there is a basic level of control below which the pilot cannot recurse, for power it is the engine speed from the tachometer and for pitch, the pitch reading from the horizon.

The pilot must plan how to achieve a new goal by creating sub-goals and deciding what instruments to use for deciding on and enacting control such that the sub-goal and ultimately the major goal is achieved. A knowledge-based controller is well suited to this type of control methodology and this controller exploits it.

6.3 Instrument Reading Scheme

While flying, a pilot scans the aircraft's instruments as opposed to taking a snapshot of all the instruments at one instant. The controller for this thesis attempts to emulate this procedure when getting the values which correspond to the instrument readings. Some values are read at a higher rate than others, corresponding to the speed at which they change as the aircraft reacts to inputs or the environment, or to how fast it is desired that the aircraft react.

Using fixed sample times in a scan fashion is only a partial implementation of what is ultimately desired. If the aircraft's state is at the desired set points and in its recent history the aircraft has been well behaved, then the sample time could be lengthened. A pilot does this when performing other duties in the aircraft, and if an error develops, it will receive the appropriate amount of attention for its correction. It is obvious then, that if the sample times are allowed to float, then the output gains may also have to be changed to suit. This is the type of task a knowledge-based controller can be applied to. It remains to be seen if this provides any real advantages, although it may for complicated multi-level controllers.

6.4 <u>Fuzzy Set</u>

Two problems present themselves when using fuzzy sets. First, how to create the fuzzy set and second how to convert a fuzzy set back to a discrete value for output. Within each of these, other problems exist such as the number of elements the set is to contain, how the mapping to the elements of the set is to be done and if the set is always going to be complete (ie. contain all the possible elements with a certainty factor). This section discusses how these problems were solved for this thesis.

6.4.1 <u>Fuzzifier</u>

The fuzzy set used for this thesis has the equivalent of seven quantization levels. This is realized by the use of four elements to indicate magnitude and two to indicate direction. The linguistic variables were chosen to relate as closely as possible to terms used by a pilot to describe the rules for flying. The name of each set created corresponded to the value read, such as altitude, pitch rate, airspeed, etc. The names of the sets elements indicate the error in the reading from the desired value and are as follows; correct, small, medium and large for magnitude and low or high for direction.

The fuzzy set used for the PC Plus prototypes used a trapezoidal shape with straight line segments for mapping the error value to the certainty factor for each element of the set (Figure 6.2). This fuzzy set and fuzzifier was sufficient but a smooth function was chosen for the Smalltalk/V implementation in order to simplify coding. The fuzzy ranges of the elements were also changed slightly. The correct and small ranges are narrower and the small error range is shifted towards correct slightly as is the range for medium relative to the original set. Figure 6.3 depicts the fuzzy set used. The direction elements correspond to the side of correct the error corresponds to. Table 6.1 lists the fuzzy set elements and the general functions that apply to them.

For each value, the range mapped into the fuzzy set varies. These ranges were picked based on the requirements to which the instrument readings must be maintained on an actual Department of Transport flight test [FLI85]. Those not stipulated were given an estimated value to begin with. Some of the mapping ranges were later modified to achieve better controller performance as shown in Table 6.2.

If one of the basic instruments, ie. the tachometer or pitch, are primary for control, and the error is beyond the current mapping range, the range can be modified by the system. This also necessitates changing either the sample time (extending it) or preferably modifying the output gain

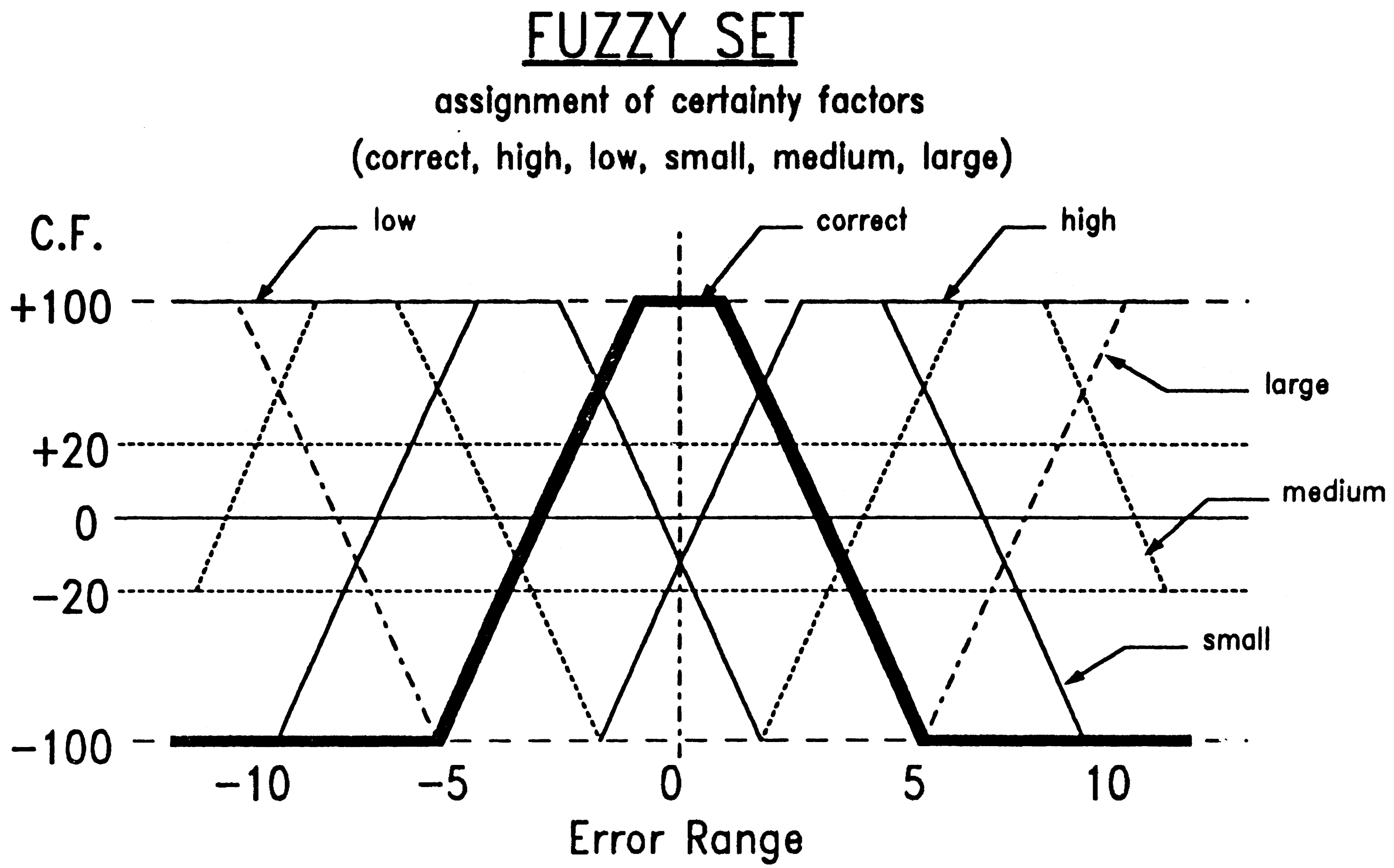


Figure 6.2 Fuzzy Set for PC Plus Prototypes

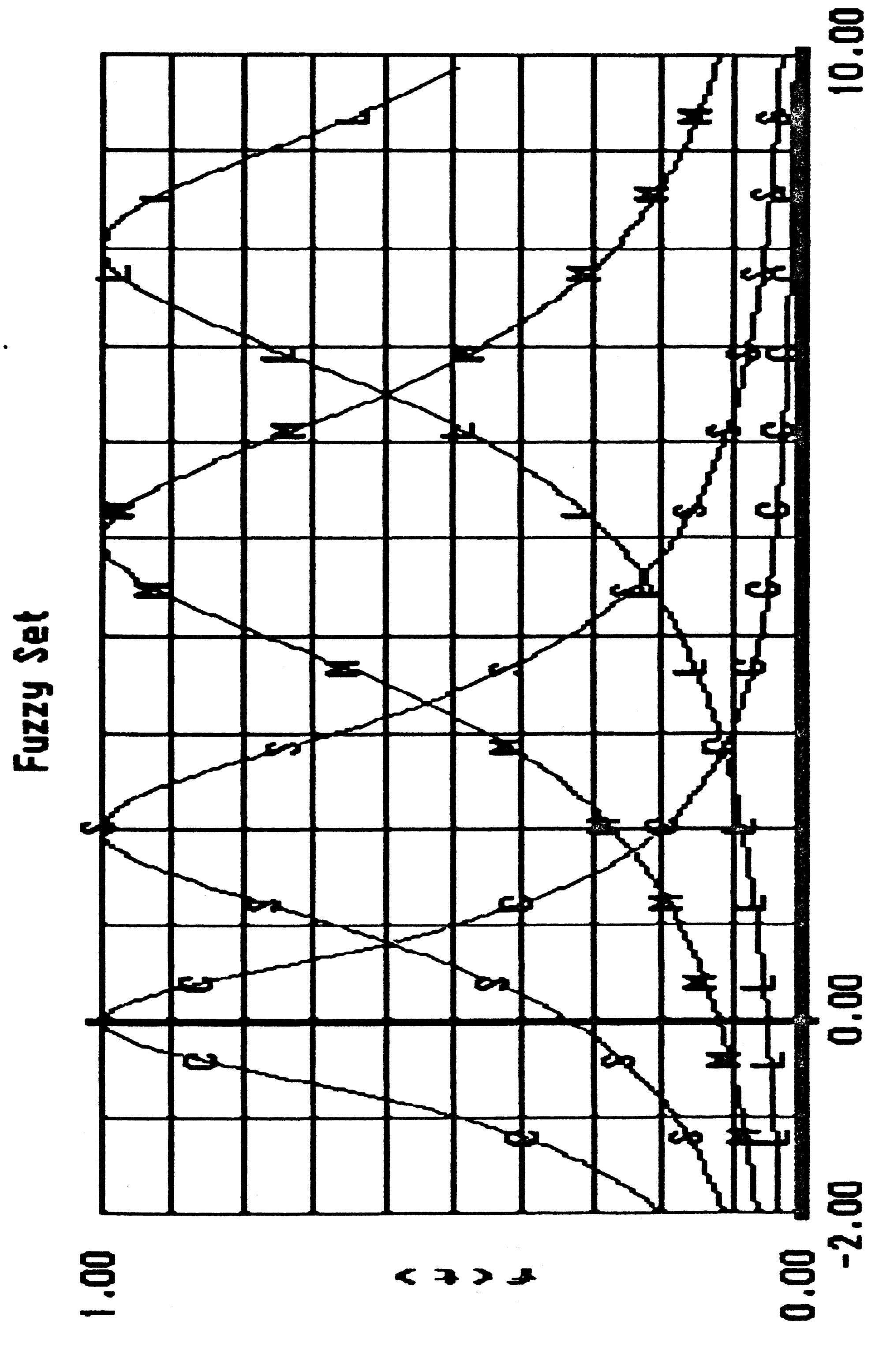


Figure 6.3 Fuzzy Set used with Prolog/V

Table of Fuzzy Set Functions

Element	Function;x is the scaled error value
Correct	$f(x) = 1/(1 + x^2)$
Small	$f(x) = 1/(1 + 0.5(x - 2)^2)$
Medium	$f(x) = 1/(1 + 0.3(x - 5)^2)$
Large	$f(x) = 1/(1 + 0.3(x - 8)^2);\ x \leqq 8$ $f(x) = 1;\ x > 8$
Low	$f(x) = 1 - 1/(1 + x^2);\ x \leqq 0$ $f(x) = 0;\ x > 0$
High	$f(x) = 1 - 1/(1 + x^2);\ x \geqq 0$ $f(x) = 0;\ x < 0$

Table 6.1 Fuzzy Set Functions - Prolog/V

Mapping Ranges for Fuzzy Set

Value	initial	final
tachometer (rpm)	100	500
airspeed (mph)	10	5
acceleration (mph/s)	n.a.	5.0 * 0.2 (airspeed/5)
altitude (ft)	100	100
pitch (degrees)	15	2
pitch rate (degrees/s)	n.a.	2.0 * 0.2 (pitch/5)
vertical speed (ft/m)	100	100

Table 6.2 Mapping Ranges for Fuzzy Set

by increasing it. The mapping range and gain can then be decreased or returned to the nominal values when the error has decreased sufficiently. It is only for the basic instruments that this is necessary. With this fuzzy set, the error is maintained within the correct range which is about 0.08 to 0.1 of the mapping or error range.

6.4.2 Defuzzification

The solution derived by the knowledge-based system is in the form of a fuzzy set. The question now is how to derive an output value which can be used to change the process inputs. The process will require a value which will indicate the direction and magnitude of change for the input. The defuzzifier must use some method of going from the fuzzy output set to a single discrete value.

A very simple conversion method was chosen for this thesis. Each fuzzy element is assigned a magnitude value for the output change. The output selected is the fuzzy element with highest certainty factor. The corresponding magnitude value is then multiplied by a gain which is related to the current error range mapping for the current primary instrument and given a positive or negative direction depending on which direction element has the maximum certainty factor. In this thesis the output has different magnitudes and gains assigned corresponding to the current primary instrument to allow for experimentation and tuning.

The output gain is also adjusted as the error range for the instrument changes. This is most valuable for the basic instruments, allowing the controller to operate satisfactorily for large errors. Table 6.3 shows the output magnitudes and gain calculations at the time of writing.

There are other schemes which could have been used to defuzzify the output set, as there are many choices to create the input fuzzy set. Some methods try to combine all the information contained in each set and the certainty factors. These allow for a more continuous range of output values. Thus the controller could distinguish between an output set with medium as the maximum element yet with small having a larger certainty factor than large and therefore the output value will reflect this in its magnitude. The current method allows the designer to vary the aggressiveness of the controller depending on the magnitude of the error and the individual primary instrument.

6.5 Structure of the Controller

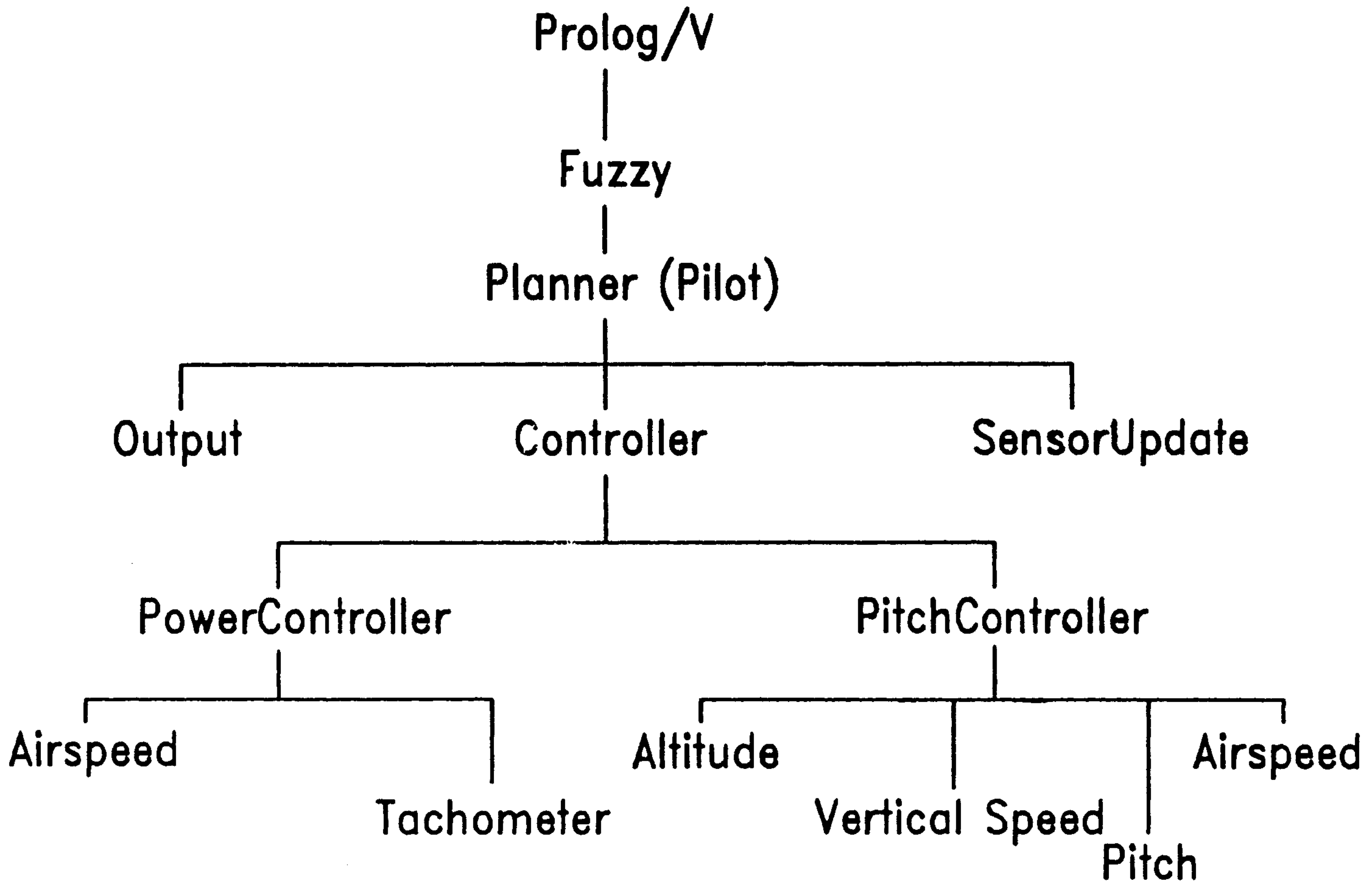
The control system was designed to exploit the natural hierarchy and segmentation inherent in the process, this is shown in Figure 6.4. Also note in the bottom leaves of the tree like structure, the figure is arranged to display the hierarchy of the instruments in the chain of control for the corresponding output. Control using the higher instruments as primary requires rules that refer to

Table 6.3 Table of Output Conversions

Magnitudes and Gains for Output Conversion

Controlled Par.	Primary Inst.	Correct	Small	Medium	Large	Gain
Power	tachometer	0	0.8	3.3	6.5	0.1
Power	airspeed	0	1.0	5.0	10.0	4.0
Pitch	vertical speed	0	1.0	4.0	8.0	8.33E-04
Pitch	altitude	0	0.8	2.5	4.7	5.56E-04
Pitch	pitch	0	1.0	4.0	8.0	2.25
Pitch	airspeed	0	1.0	4.0	8.0	-1.96E-04

Figure 6.4 System Structure



Structure and Inheritance of Knowledge-based Controller the lower instruments for adequate and correct control.

The Planner or Pilot frame oversees and coordinates the knowledge-based controller. It contains rules to initialize and clear the system databases as well as rules that are generic to the overall system and required in the child frames. This frame is responsible for coordinating sensor (instrument) reading by invoking the rules in the SensorUpdate frame, requesting a control action or solution from the Controller frame and then having the solution converted to a discrete output by the Output frame. The output value and time until the next sensor reading are then returned to the process.

The frame UpdateSensor, obtains the sensor reading, has it converted to a fuzzy set and modifies the output gains and fuzzy mapping if necessary.

The Controller frame invokes the appropriate child frame depending on whether the control action required is for the pitch axis or for power control. If the sensor just read is primary for pitch or power then the appropriate child frame is consulted, otherwise no action is taken and it returns to the parent frame. Rules generic to all the child frames for control are also present in this frame.

The PitchController and PowerController frame merely decide which specific child to consult based on the primary instrument. They also contain rules generic to Pitch or Power control respectively.

The bottom leaf frames decide what control action is best to correct an error in the primary instrument reading using information from other sensors as needed in the rules for effective control.

Figure 6.4 also shows the inheritance of predicates from Prolog/V and the Fuzzy frame.

6.6 Rules

The system rules are in the form of Prolog/V predicates. As the fuzzy set operations are an add on to Prolog/V, the certainty factors are calculated using the fuzzy predicates utilizing reverse polish notation [RIC88]. A predicate is then used to decide if the rule has succeeded or failed and therefore if backtracking to another rule and possible solution is necessary. This is unfortunate because it transfers part of the control mechanism into the rules and therefore the knowledge-base. This was necessary due to this implementation of fuzzy logic. Two solutions to this problem are possible. One, a different knowledge representation scheme can be used and create an inference engine in Prolog/V to deal with fuzzy logic. This however may make it very difficult to take advantage of the object oriented programming paradigm for implementation of frames. Second, a Fuzzy Prolog could be implemented as in [RIC86].

The following is an example predicate from the PitchAltitude frame for controlling the pitch axis with altitude as the primary information.

```
  checkClimb() :-
       errorIn (#pitchRate, #low), fNOT(),
       errorIn (#verticalSpeed, #medium), fAND(),
       errorIn (#altitude, #small), fAND(),
       fuzzy (cf),
       makeOutput (#pitch, [#small, cf]),
       goWith (cf), !.
```

which translates to;

```
  IF the pitch rate error is not low and the vertical speed
       error is medium and the altitude error is small
  THEN checkClimb succeeds by adding the control magnitude
       of small to the output set for pitch
```

This rule decides the magnitude of the control deflection and another rule would decide the direction in order to check or slow the climb of the aircraft. There are over one hundred rules in the system, the majority of which are in the bottom control frames.

CHAPTER 7
SIMULATION RESULTS

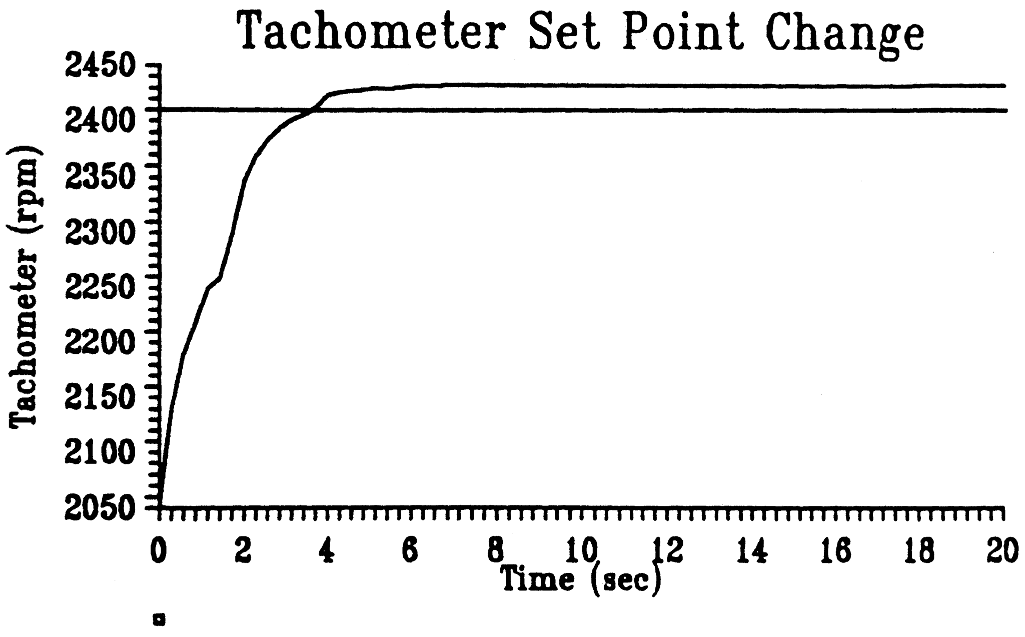
7.1 Introduction

This chapter contains a small number of simulation runs to demonstrate the knowledge-based controller's performance. Different sensors are selected as primary and a set point change, within the fuzzy error range, is made. For the control of power, an example of a large set point change of the airspeed is presented. Here the controller must switch to a new primary instrument for a short time.

7.2 Power Control

The response to a commanded change in the power output or engine speed is shown in Figure 7.1. The system is over-damped and quite easy to control. With the fuzzy error range set at 500 rpm the error can be quite large at plus or minus 40 rpm, but this relates to an approximate airspeed difference of plus or minus 2 mph.

Figure 7.2 (a & b) shows the aircraft response to a small commanded change in airspeed. The controller strives to achieve an acceptable acceleration for the magnitude of the airspeed change which it will maintain until the airspeed error decreases. The controller output depends on



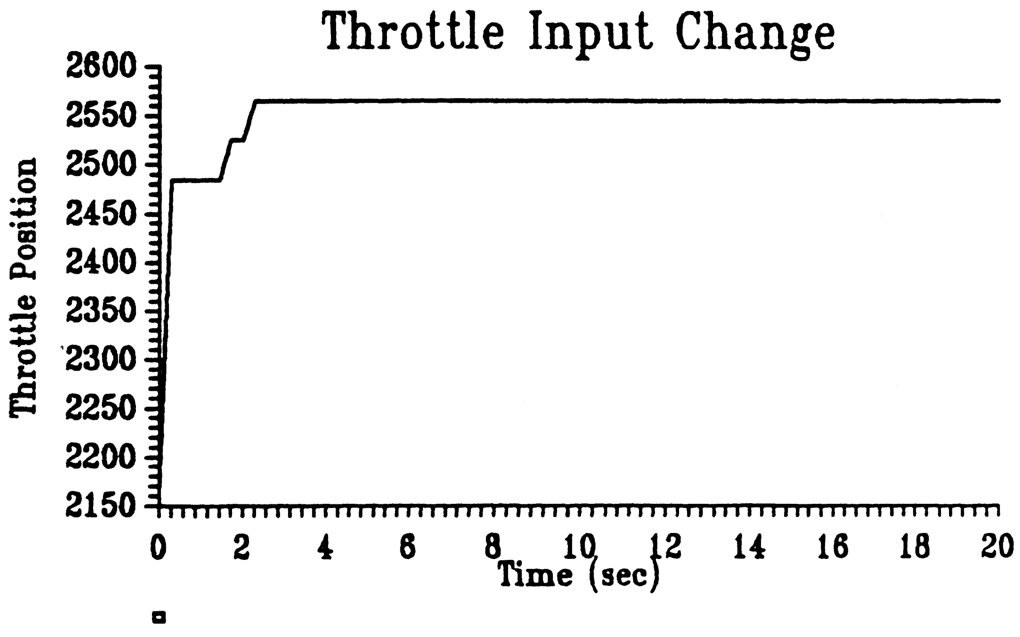
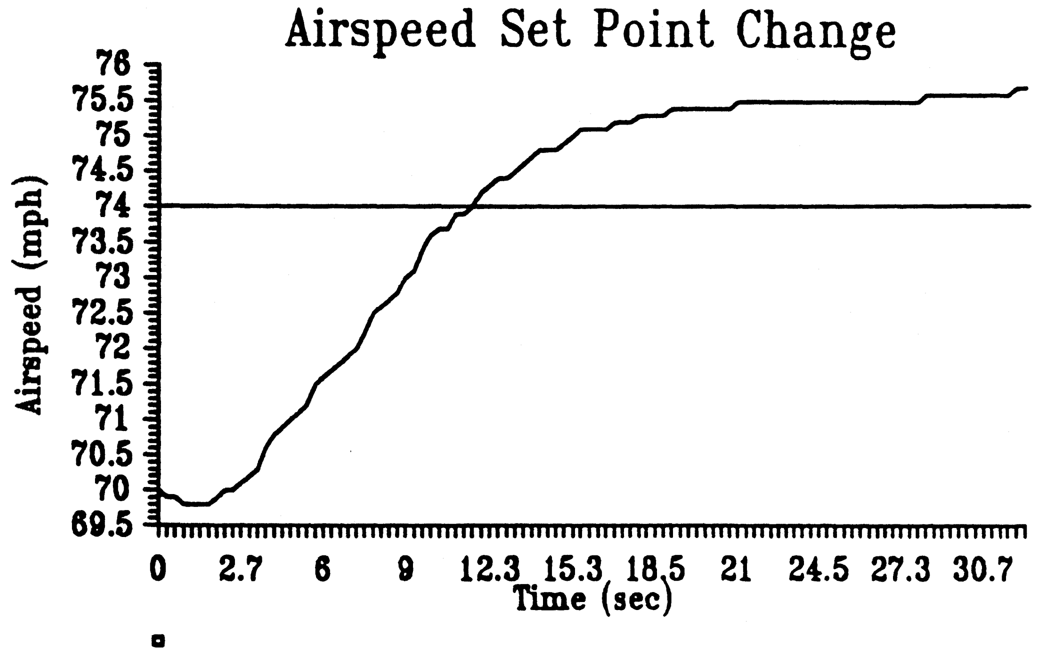


Figure 7.1 Tachometer Set Point Change
Vertical Speed constant at 0 fpm



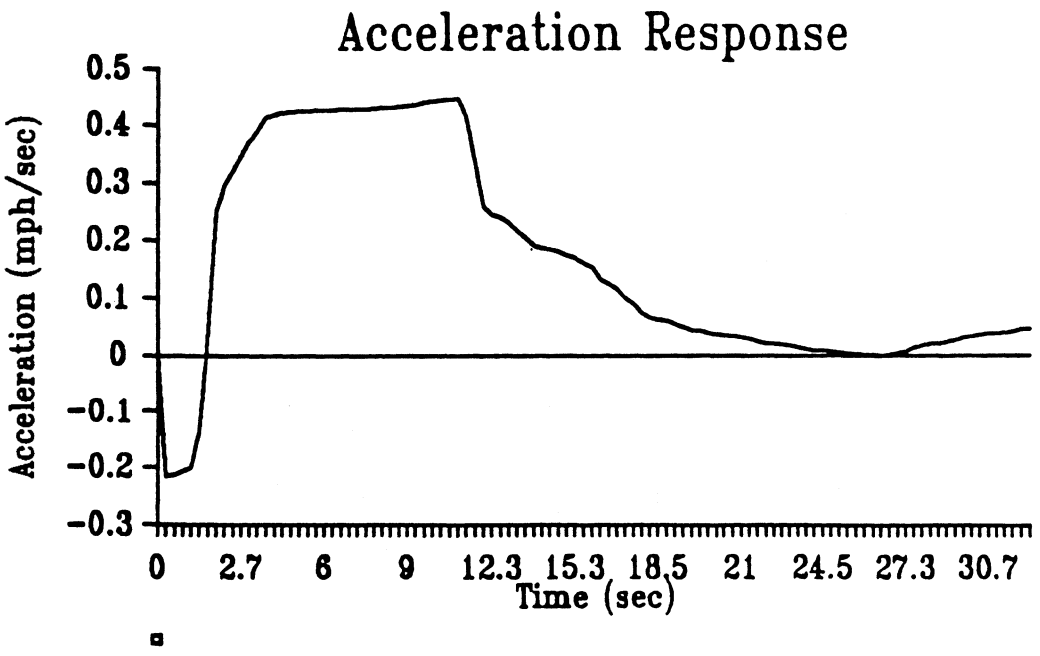
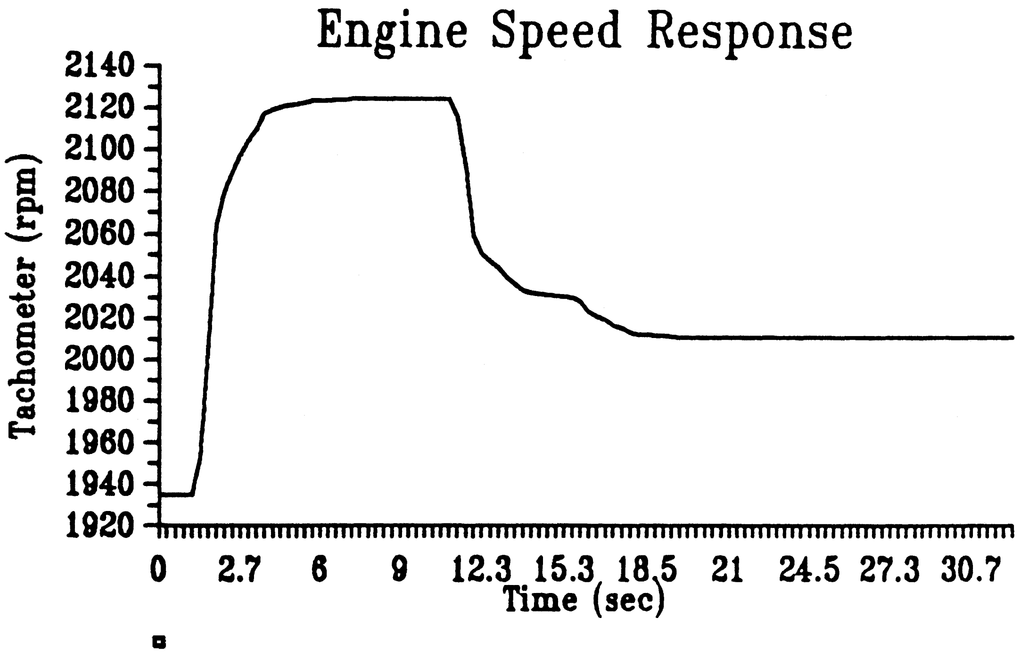


Figure 7.2-a Airspeed Set Point Change
Vertical Speed Constant at 0 fpm



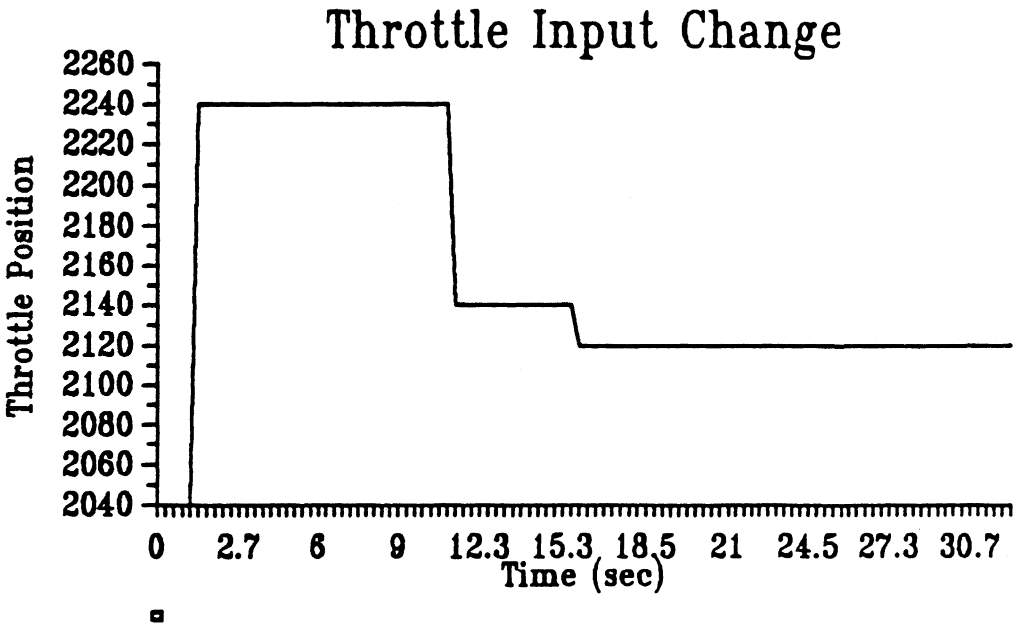


Figure 7.2-b Airspeed Set Point Change
Vertical Speed Constant at 0 fpm

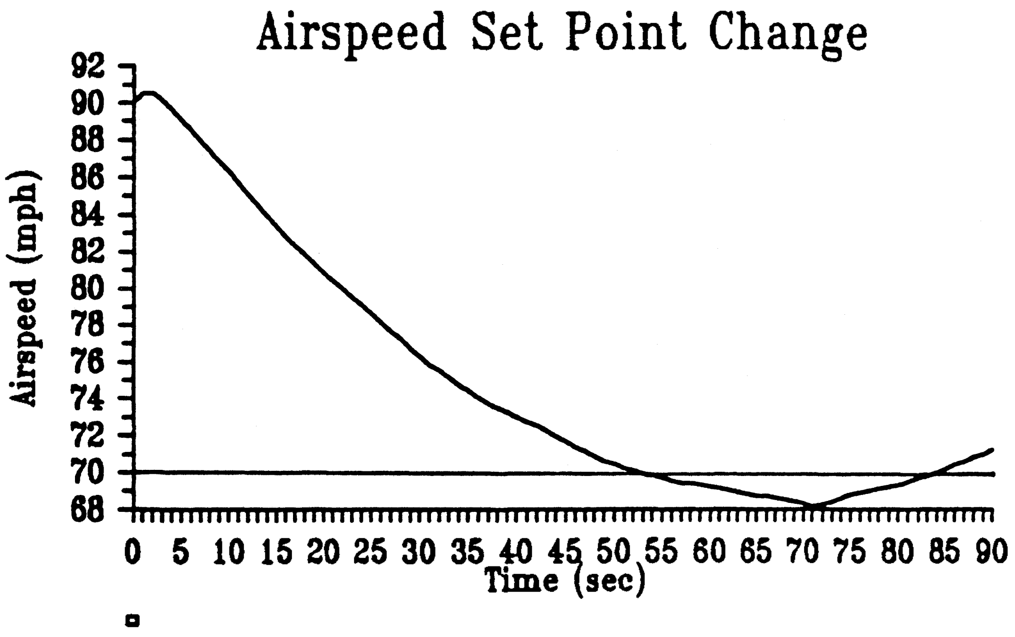
the airspeed error and the acceleration both in magnitude and direction.

A larger commanded change in the airspeed is shown in Figures 7.3 (a-c). Here a commanded deceleration of 20 mph is made. The commanded change is larger than can be dealt with by using the airspeed indicator alone. The controller therefore calculates an estimate for a new tachometer setting based on a "rule of thumb" and temporarily assigns the tachometer as the primary instrument for power control. When the needle movement (acceleration) is sufficiently slow the controller switches back to the airspeed as primary. This occurs at approximately 50 seconds in Figure 7.3. At this point, if the airspeed error was still too large, a new tachometer setting would have been found and this instrument again assigned as primary for power control. In this example, the airspeed error was within an acceptable range and the airspeed indicator remained as primary.

7.3 Pitch Control

The three main values controlled with pitch are presented in this section, beginning with the basic pitch control. Figure 7.4 shows the response of the system to a change of pitch angle command. Pitch position is controlled by a combination of position and rate error similar to airspeed control.

A set point change for vertical speed is shown in



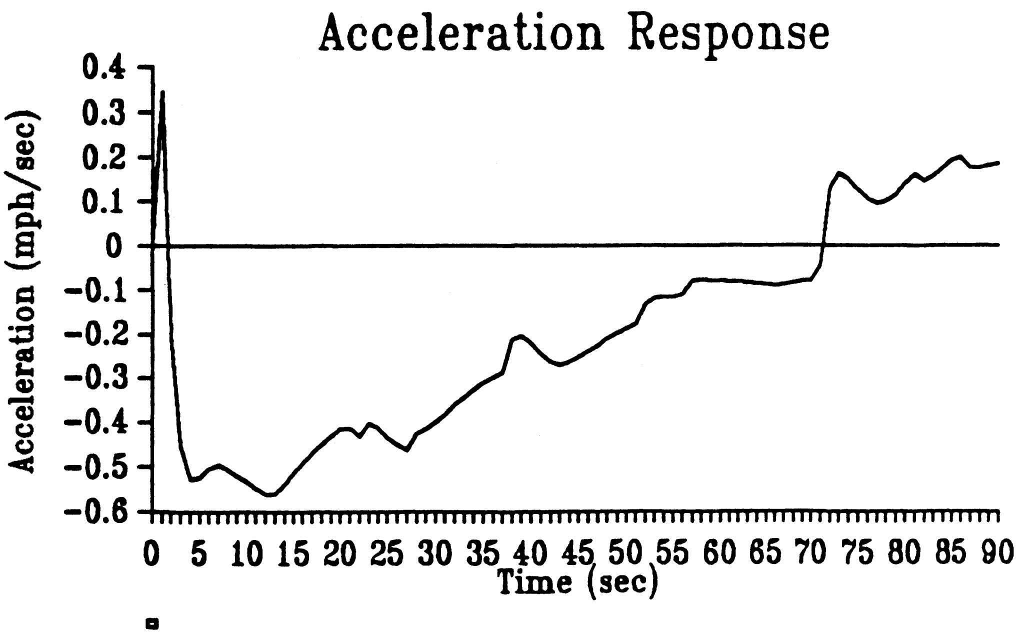
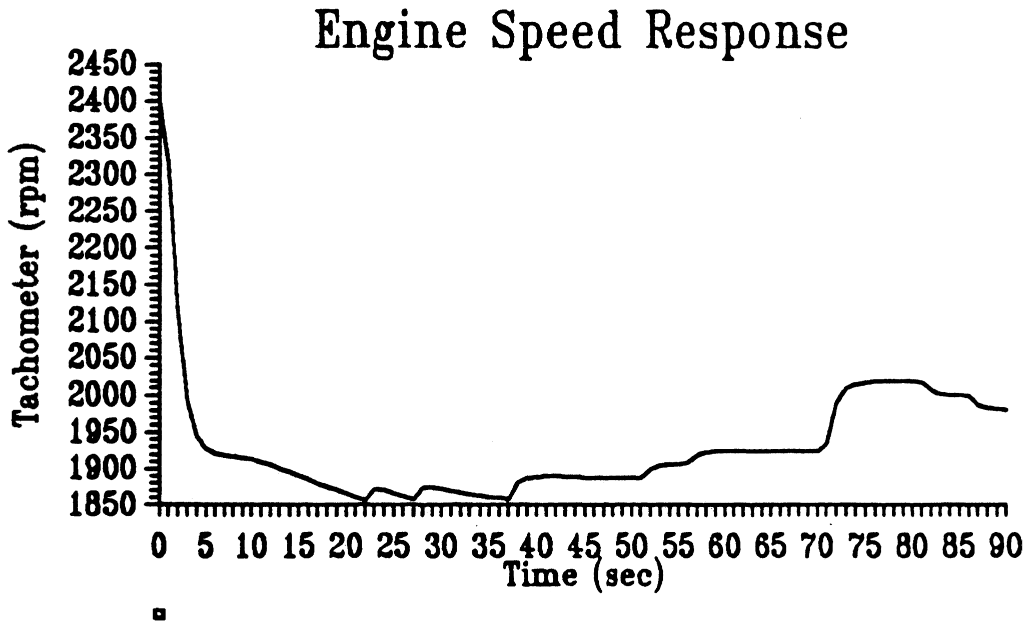


Figure 7.3-a Large Airspeed Set Point Change
Vertical Speed Constant at 0 fpm



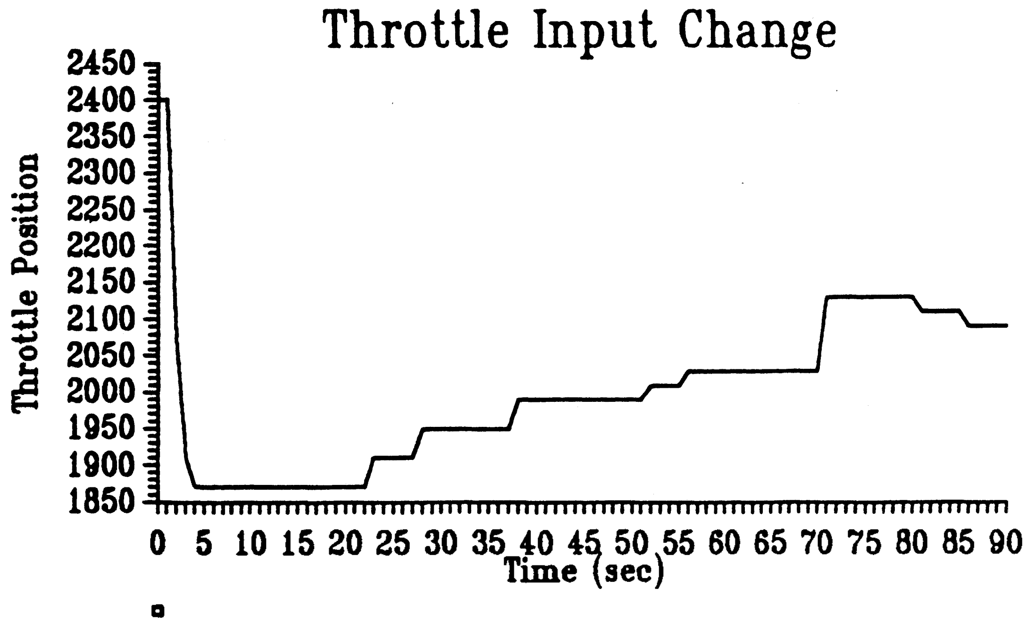


Figure 7.3-b Large Airspeed Set Point Change
Vertical Speed Constant at 0 fpm

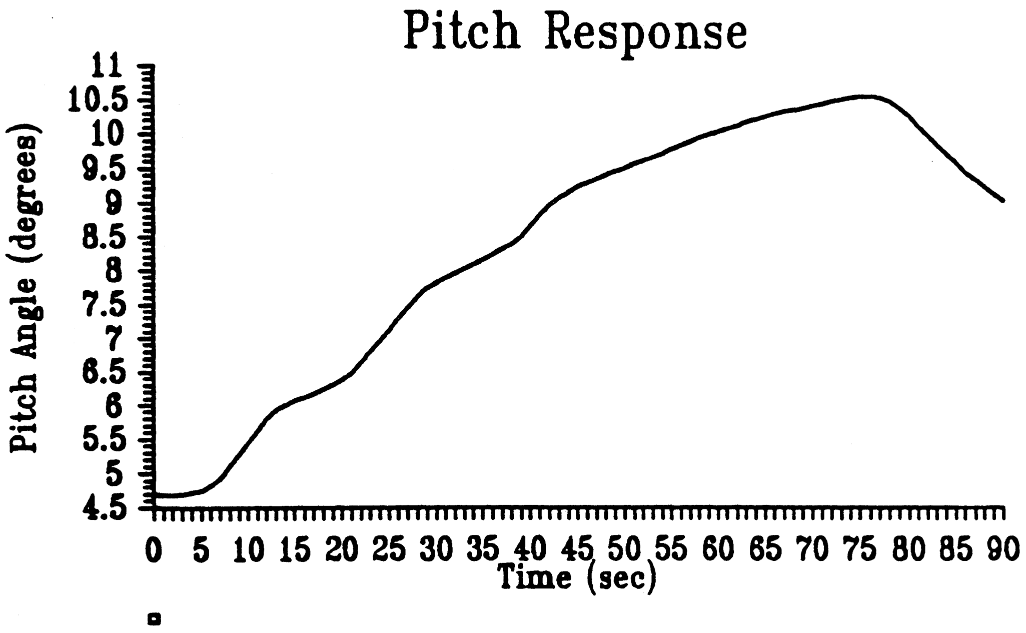
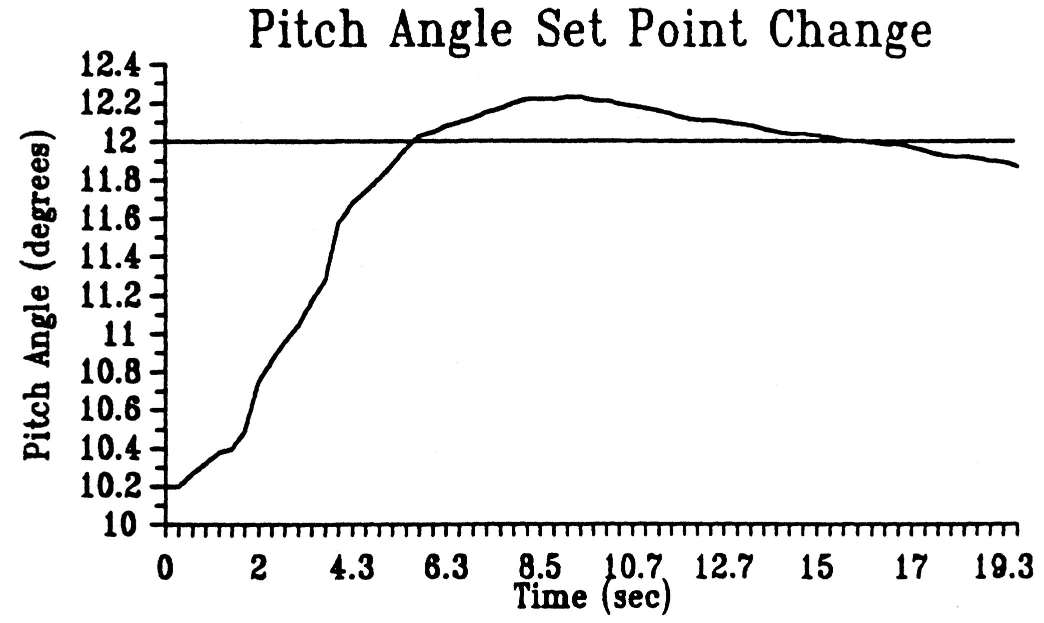


Figure 7.3-c Large Airspeed Set Point Change
Vertical Speed Constant at 0 fpm



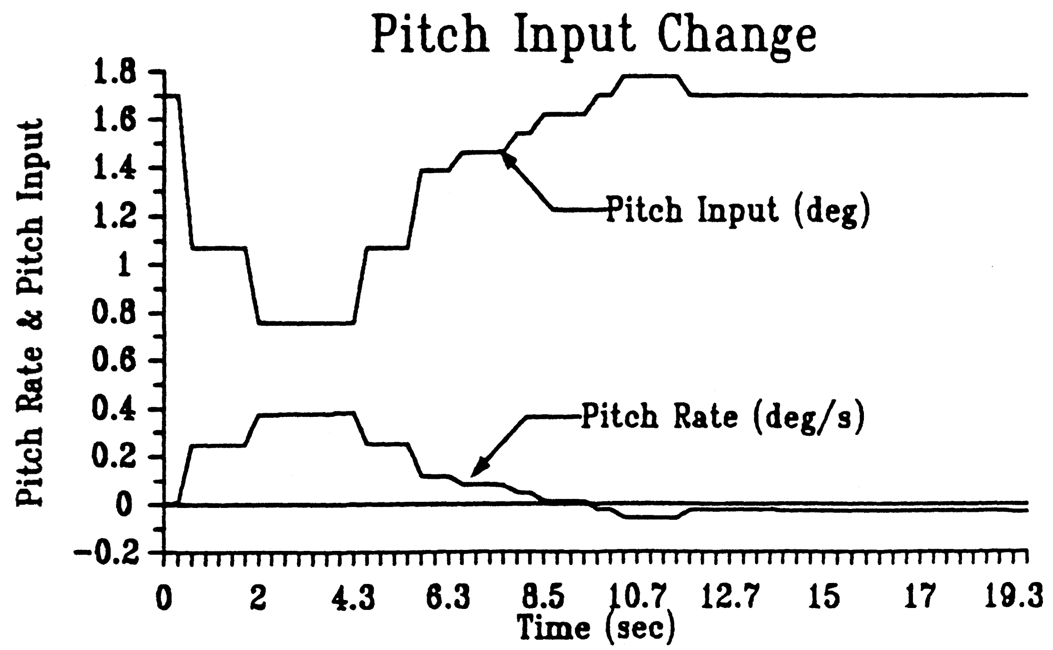
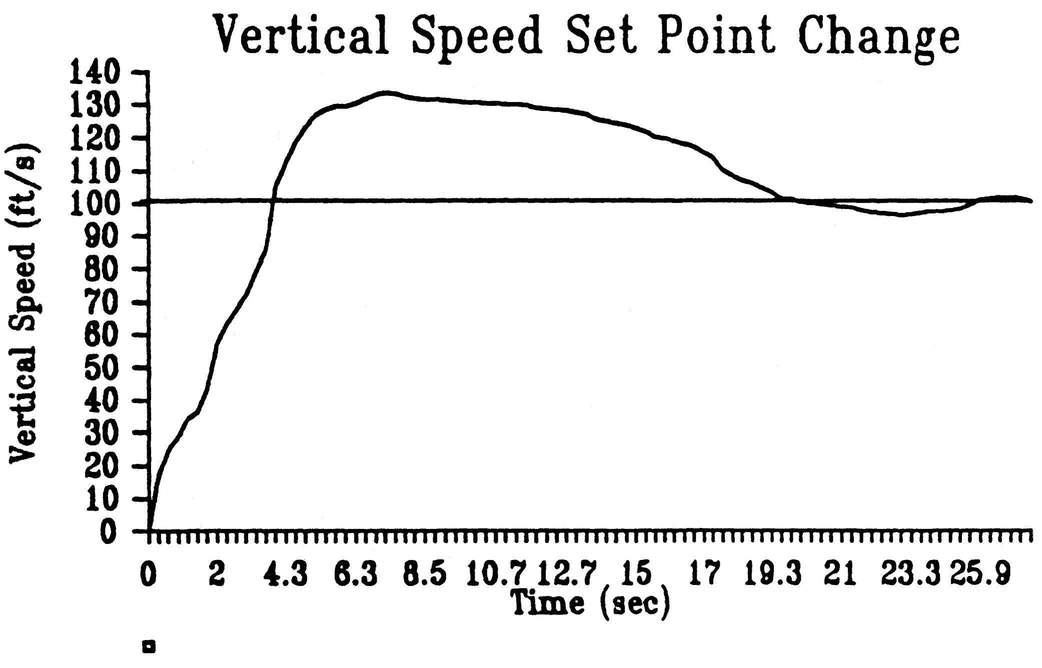


Figure 7.4 Pitch Angle Set Point Change
Airspeed Constant

Figure 7.5 (a & b). The control of vertical speed has been set to be quite aggressive with a resultant large overshoot. Part of this overshoot error is also attributed to simultaneous airspeed and power changes taking place. However, it can be seen that the response settles quite well to a small error for the vertical speed.

A set point change in the altitude is the slowest and most difficult to control. The plots in Figure 7.6 (a & b) show the response of the aircraft and the pitch control action. To control the altitude, a vertical speed that is commensurate with the altitude error must be achieved, but to do this care must be taken not to over control the pitch angle and pitch rate. This control is achieved with reference only to the fuzzy values, ie. for a medium error, a medium rate of climb is used. This is continued for the pitch angle and rate to achieve the desired climb. As the altitude error decreases, so must the rate of climb be decreased, ie. small altitude error, then use a small rate of climb. Therefore for altitude control there are rules for starting, checking and stopping a climb as well as controlling pitching of the aircraft. Similar rules are used elsewhere, but altitude is the most complicated and complete example of the interaction.



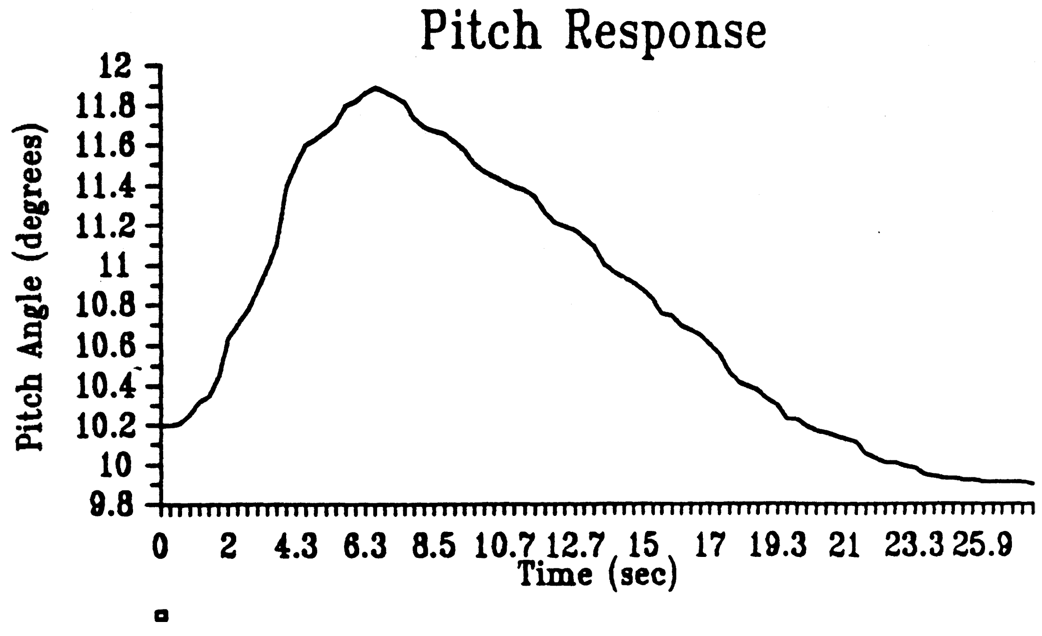
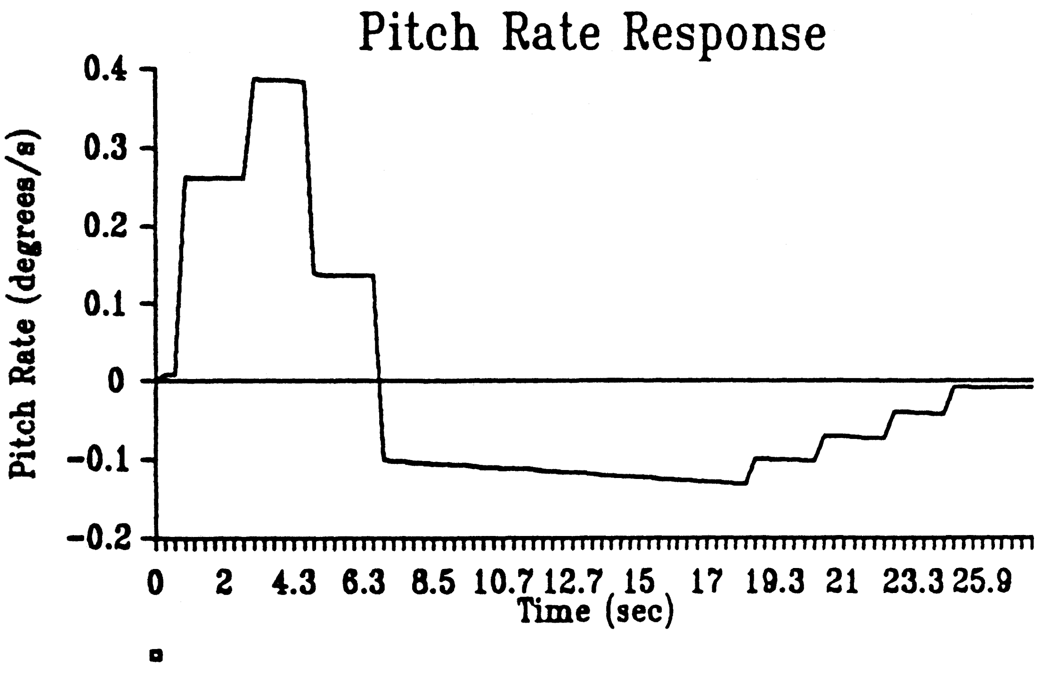


Figure 7.5-a Vertical Speed Set Point Change
Airspeed Constant



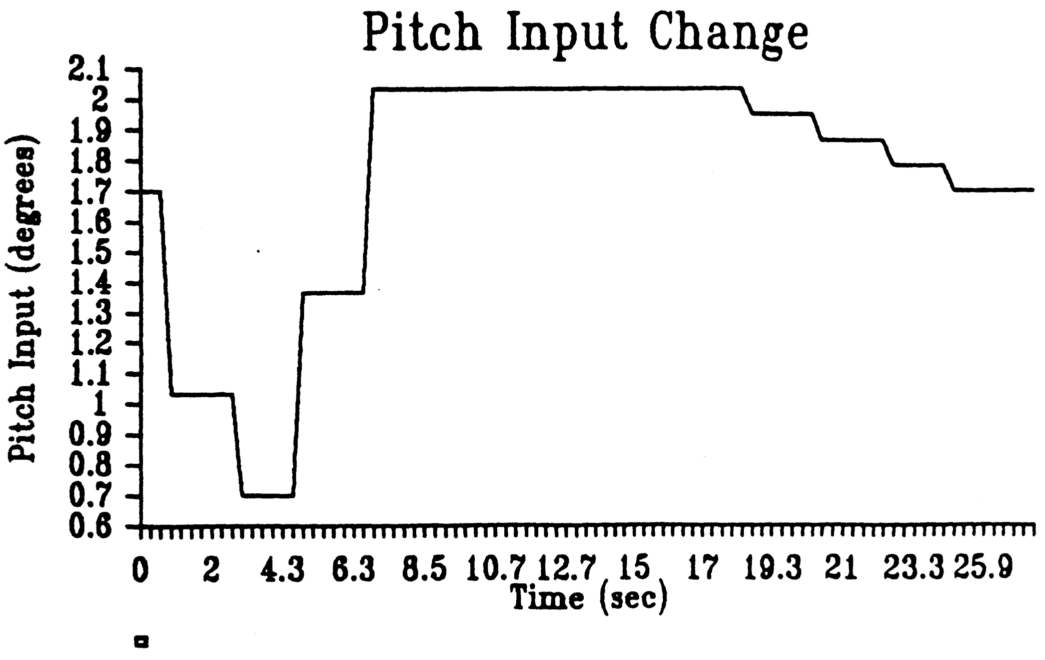
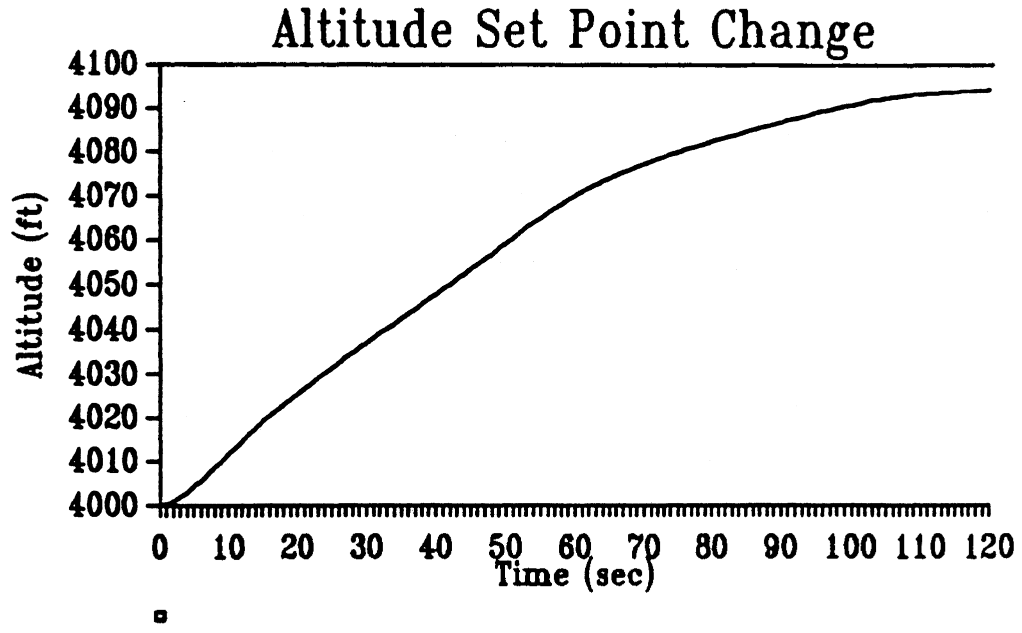


Figure 7.5-b Vertical Speed Set Point Change
Airspeed Constant



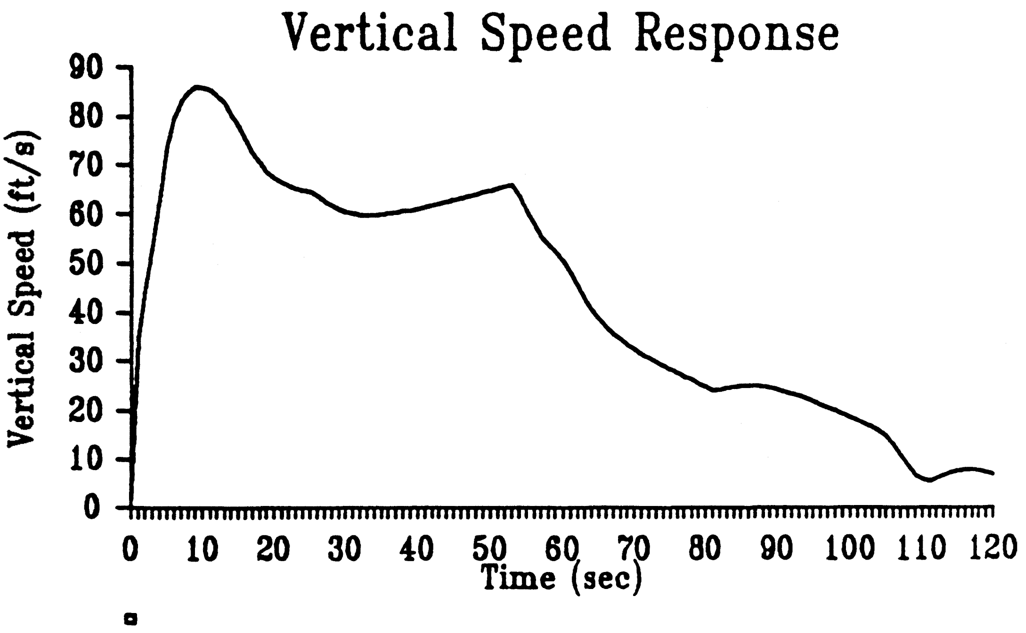
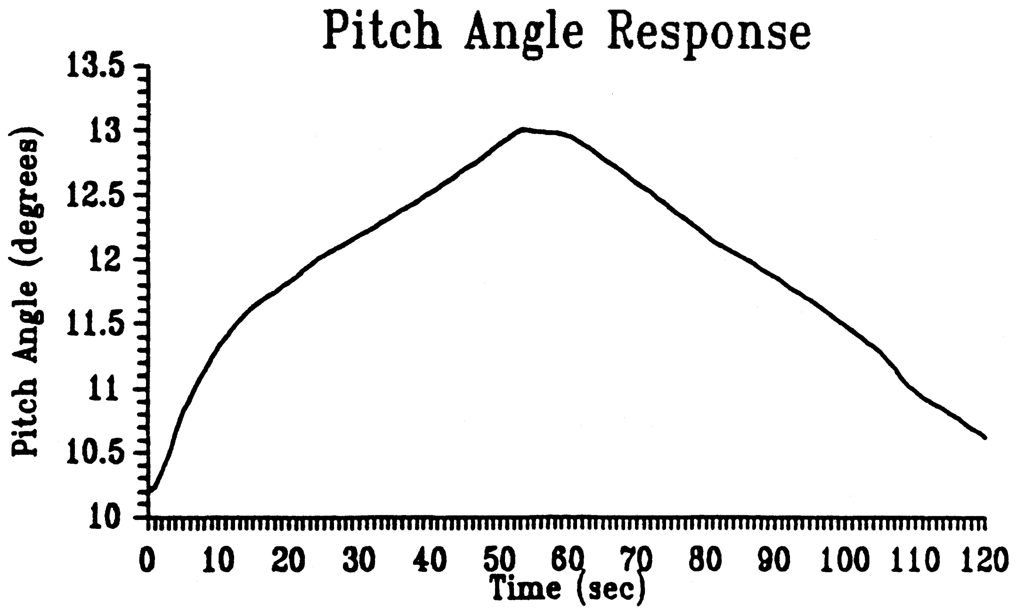


Figure 7.6-a Altitude Set Point Change
Engine Speed Constant



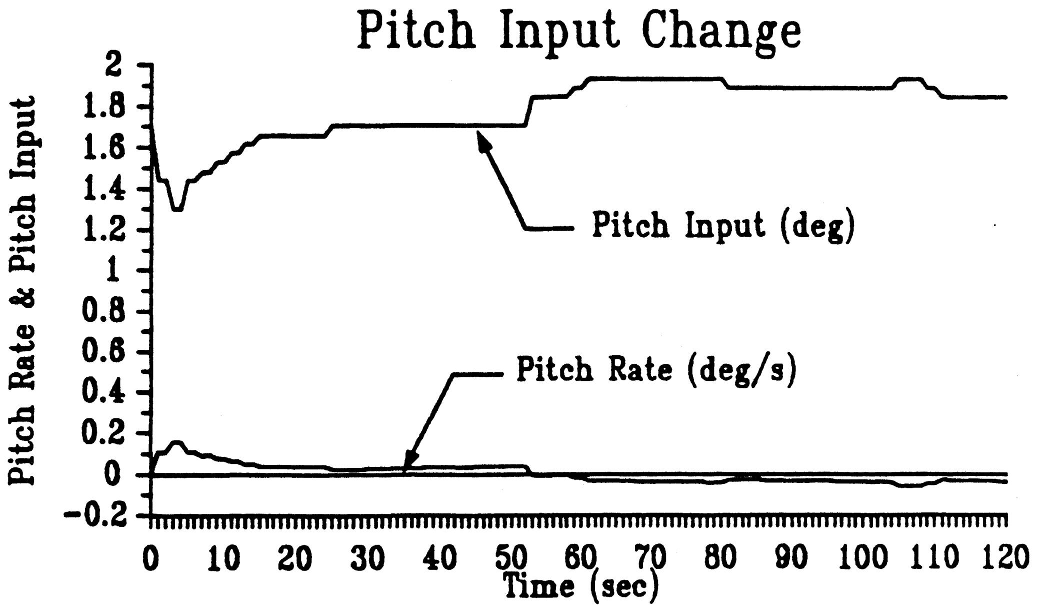


Figure 7.6-b Altitude Set Point Change
Engine Speed Constant

7.4 Conclusion

The controller uses a relatively small set of values to achieve control of the system and relies instead on the knowledge contained in the system to accomplish acceptable performance. The gains and sample times can also be juggled to tune the response but this is no substitute for a correct knowledge-base for the control of the process. In fact rules can also be written to adjust the gains, sample times and fuzzy mapping ranges as necessary for different situations and requirements.

This section has demonstrated that a process can be controlled with a knowledge-based controller utilizing fuzzy logic. The controller is realized with out explicit knowledge of the system, but using only operator knowledge and a knowledge representation scheme to best exploit that knowledge.

CHAPTER 8

APPLICATION OF NEURAL NETWORKS

8.1 Introduction

This thesis has so far presented knowledge-based systems as a method of emulating human thought and then using human rules and reasoning methods in a control application. A recent resurgence of interest in neural networks has researchers claiming their applicability to many problems including automatic control [JOS88]. The neural network is an attempt to model the basic structure and operation of the human brain and therefore emulate the way intelligent information processing occurs within the brain [TRE88]. Implementations of neural networks have existed since the beginning of the computer age. The current wave of excitement seems to be a product of recent hardware advances in the construction of massively parallel machines and theoretical advances which even increase the computational power of neural networks in conventional Von Neuman machines.

Neural systems are a pattern directed reasoning mechanism and have been shown to develop internal representations, through self-organization, of real or abstract classes in the external environment [JON87]. From this self-organization, collective properties emerge which include association, categories of generalization, differentiation, preferential learning, optimization, fault tolerance and hyperacuity [JOS88].

This chapter gives a short explanation of the construction of a neural network, then three implementation trials are discussed.

8.2 Neural Network Approach

Intelligent behaviour and computational power of the human brain seems to come from interactions involving large numbers of neurons that are connected together by a complex network of synapses. Each neuron is quite limited in its processing capabilities compared to a computer, but it is the combined ability of many connected neurons which is the power of a neural network. A simulated neuron has four important components; input connections or synapses through which it receives activation from other units, a summation function that combines the various input activations into a single activation, a threshold function that converts this summation of input activations into an output activation and finally, output connections or axonal paths by which a unit's output activation arrives as the input activation at other units in the system [JON87].

Inter-connections between computer simulated neurons are typically assigned a weighting factor to modulate the activation passing between the units. The absence of a connection can be represented by a weight of zero, and an inhibitory relation by a negative connection weight.

In a neural network the knowledge lies in the connection weights between the units. In contrast, the knowledge of a knowledge-based system lies in the rules and the relationship of the frames. Additionally, a neural network is driven by the activation that passes between units. An expert system is driven by the firing of rules using an inference engine.

Much of the current resurgence of interest in neural networks is due to the development of a powerful learning rule that can determine inter-connection weights for multi-layered systems. This learning rule is the back-propagation rule and is well explained in [JON87]. It takes the error from a desired output for a given set of inputs and propagates it back through the network to adjust the interconnection weights and thresholds at each neuron.

Neural networks can learn the behaviour of a system or domain expert from observation or a specified training set. This could provide substantial saving in effort required in comparison with an equivalent rule or frame based expert system. A set of example inputs and the required outputs from a system may already exist or be easily specified. These could be used to train a neural network quite easily, but it may take considerable effort and guess work to distill the knowledge from these and put it in a form usable by an expert system. The rules must be developed, entered into the knowledge base (which must be designed to suit the current problem) and checked for compatibility and consistency with the existing knowledge base.

The back-propagation learning rule can require large amounts of computational resources, the demand for which increases dramatically with increases in the size of the network (the number of neurons and the number of layers), the size of the training set and decreases in the error tolerance to which the network is to be trained. With current implementations a single new rule cannot be just added to the neural system. The network must be retrained with the complete training set with the new example(s) included. This process should be quicker than the original training as the network need only assimilate the new information.

8.3 Neural Network Implementation Trials

Three small neural networks were implemented to investigate their application directly or indirectly to the controller implemented for this thesis. The first example was a stand alone experiment for using a neural network to estimate intermediate goals and was not incorporated into the controller. The second implementation required the coding of a neural network simulation in Smalltalk/V so that the resulting trained network(s) could be incorporated into the controller. Two networks were created for the simple task of engine speed control.

8.3.1 Estimation of Intermediate Goals

The knowledge-based controller presented in this thesis must determine an intermediate goal for pitch angle when the vertical speed set point error becomes large. These intermediate goals do not need to be very accurate as the aircraft attitude will be modified when the controller returns to the super goal. Some intermediate goals can be inferred quickly by the knowledge-based system, yet others may require a rule base too complicated to be practical. One such case exists for the setting of a pitch attitude in order to establish a desired rate of climb.

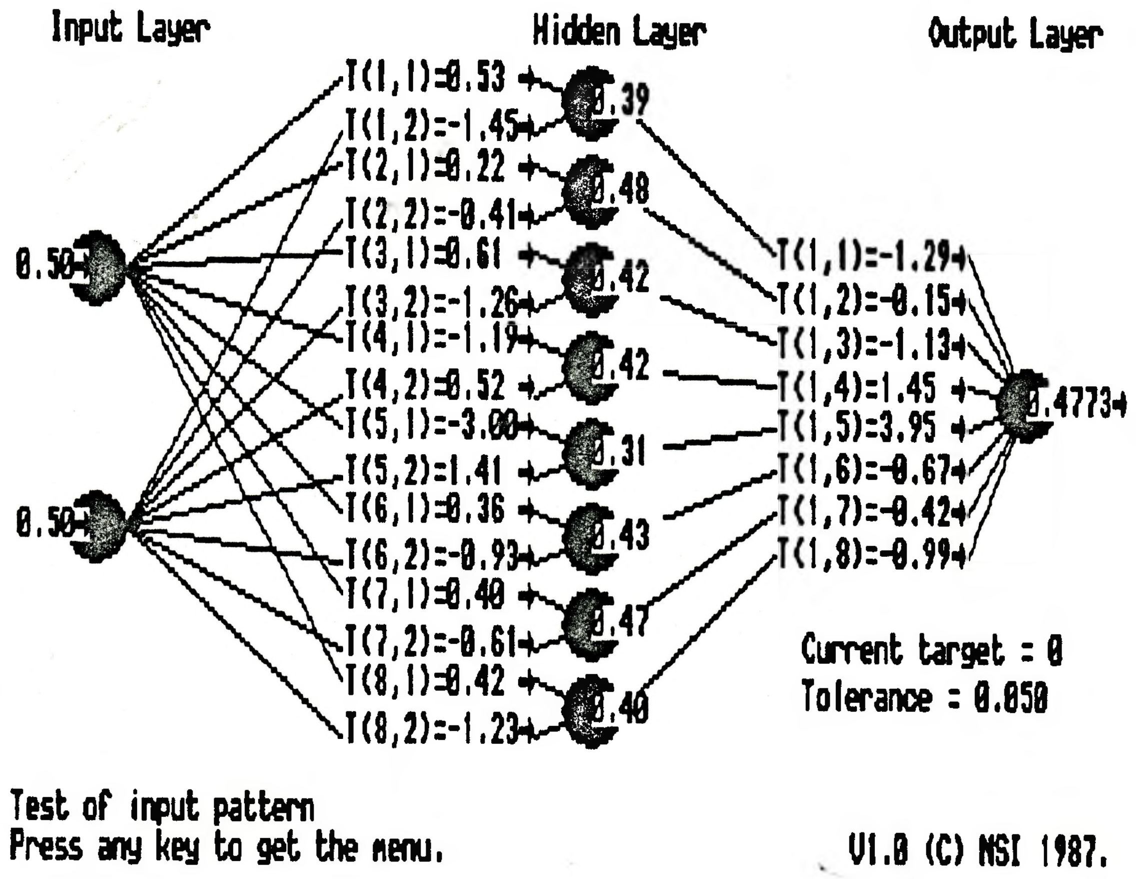
Of interest here is the ability of a neural network to learn a feature space from a relatively small training set and generalize to other features in the domain. The rate of climb of an aircraft is affected by a number of things, including air density, aircraft weight, airspeed and the angle of attack. The simulated aircraft in this thesis has a fixed weight and the air is a fixed density. Thus the only effects on the rate of climb to be considered are the airspeed and the angle of attack. The power level can affect the rate of climb as well but for normal climb attitudes the airspeed will suffice. For a given airspeed in a normal climb the pitch angle is proportional to the angle of attack of the wing.

A two input (airspeed and desired rate of climb), single output (pitch angle), neural network was required for testing. The range of rate of climb used was from -500 fpm to +500 fpm. The airspeed range is from 60 mph to 100 mph. A set of steady state data from test runs of the system was collected to be used for a training set and testing. Table 8.1 shows the data used and the results. The network was trained on a set of six values to a tolerance of 0.05. The network's inputs and output are all in the range from 0 to 1 therefore the aircraft data was mapped into this range with simple linear functions.

The example network used is from an instructional aid called "AWARENESS" and developed by Neural Systems Inc. of Vancouver B.C. The network is shown in Figure 8.1 with the training complete and an example input set. As can be seen from Table 8.1, the network performs quite well in this case predicting a pitch angle that should put the aircraft in a climb at a rate well within the error range for the vertical speed to be used as the primary instrument for pitch control.

8.3.2 Engine Speed Control

To further experiment with neural networks, the



```
Current target = 0
Tolerance = 0.050

Test of input pattern
Press any key to get the menu.
                                        V1.0 (C) NSI 1987.
```

Figure 8.1 Neural Network

Table 8.1 Neural Network Training and Test Data

	Example Data for Neural Network			Test of Network	
	Airspeed (mph)	Vertical Speed (ft/s)	Pitch Angle (degrees)	Network Output (degrees)	Difference (degrees)
*	70 (0.25)	500 (1.0)	14.3 (0.95)	13.5 (0.90)	-0.8 (-0.05)
	70 (0.25)	0 (0.5)	10.1 (0.68)	10.5 (0.70)	0.4 (0.02)
*	70 (0.25)	-500 (0.0)	5.8 (0.39)	5.4 (0.36)	-0.4 (-0.03)
*	80 (0.5)	500 (1.0)	11.0 (0.73)	11.7 (0.78)	0.7 (0.05)
	80 (0.5)	0 (0.5)	7.0 (0.47)	7.2 (0.48)	0.3 (0.01)
*	80 (0.5)	-500 (0.0)	2.9 (0.19)	3.0 (0.20)	0.1 (0.01)
*	87 (0.68)	460 (0.96)	8.7 (0.58)	9.4 (0.62)	0.7 (0.04)
	90 (0.75)	0 (0.5)	4.7 (0.31)	4.4 (0.30)	-0.3 (-0.01)
*	90 (0.75)	-500 (0.0)	1.3 (0.08)	1.9 (0.13)	0.6 (0.05)

Note:
1 Data is for a given altitude and aircraft weight.
2 Data marked with * is the training set.
3 Equivalent values for network are in brackets.

section of the knowledge-based controlling the engine speed with the tachometer as the primary instrument was replaced with a neural network. In addition, a neural network was trained to give a smoother, continuous defuzzification of the output fuzzy set. This provided insight into their function in the presence of incomplete information. A further test was planned that called for the application of a neural network to a more complicated task in the controller. A serious computer resource problem was encountered here and the plan had to be abandoned as it was beyond the scope of this thesis to try and broach this problem. A more detailed description of these experiments now follows.

The back-propagation algorithm was implemented in Smalltalk/V for these tests. The implementation allowed any combination of neurons and layers to be specified, trained and evaluated.

The engine speed controller required a network with six input neurons and six output neurons. Thus its inputs and outputs are a complete set of certainty factors gleaned from or applied to a fuzzy set. There was one hidden layer specified which contained six neurons as well. Each input and output neuron corresponds to a fuzzy value. For input, the certainty factor of a fuzzy value is applied to its corresponding neuron. Likewise, the potential of an output neuron is the certainty factor for its corresponding element in the output fuzzy set. The neural network implementation contains a class for 'fuzzy nets' so that the matching of the input and output neuron potentials to their corresponding fuzzy values in the input and output fuzzy sets is automatically handled. Therefore the neural network implementation is virtually transparent to the knowledge based controller.

The training set consisted of seven example input and output pairs. Training was done to a tolerance of 0.2 on all the output neurons for all the pairs in the training set. This required approximately 4,000 iterations to accomplish, which translated to approximately 4 hours on an 80386 based machine.

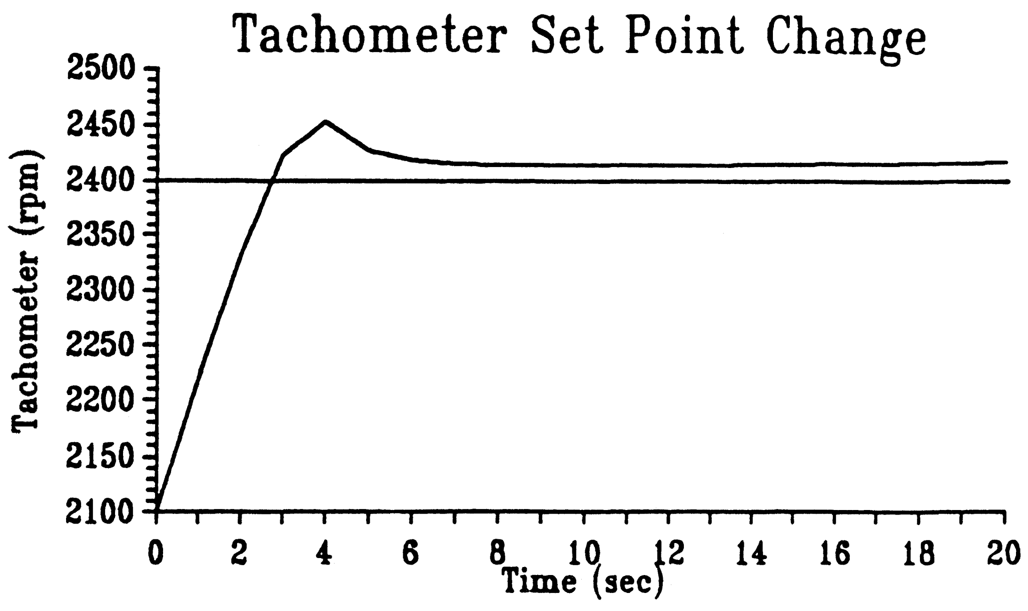
The performance of the control system utilizing the neural network was identical to those from the strictly rule-based controller. This was expected as the training set was based directly on the knowledge contained in the rules for controlling the engine speed with the tachometer as the primary instrument.

The strategy for defuzzifing the output set was briefly discussed in Chapter 6. The method selected for the controller for this thesis consisted of simply setting the final output to the value corresponding to the fuzzy value with the greatest certainty factor. This has worked well as the output set is often incomplete. A neural network with four input neurons, eight neurons in one hidden layer and one output neuron was created and trained to defuzzify the output set.

At first the network was trained assuming a complete fuzzy set for the output. The training set contained ten examples. Figure 8.2 shows the result for a tachometer set point change using the neural network controller and defuzzifier. Comparing this with the result shown in Figure 7.1, the reader will notice the presence of slightly more overshoot in the new result but also a smoother response. This is due to a continuous discreetization of the output set. The difference in steady state error is insignificant as both are within the range of the fuzzy value 'correct'.

A significant problem was encountered with defuzzifying output sets that were incomplete. All fuzzy values that were missing were given a certainty factor value of zero. It turned out that when the fuzzy output set contained values on the low end of the scale and therefore the fuzzy values on the high end of the scale were assigned certainty factors of zero, the output value was over estimated. Conversely, when the fuzzy values on the low end of the scale were missing, the output value was grossly underestimated. The result was under-control when large errors were present and over-control for small errors and therefore unstable response to a step input change.

Upon closer examination of the response of the network to various inputs it was decided that this was an



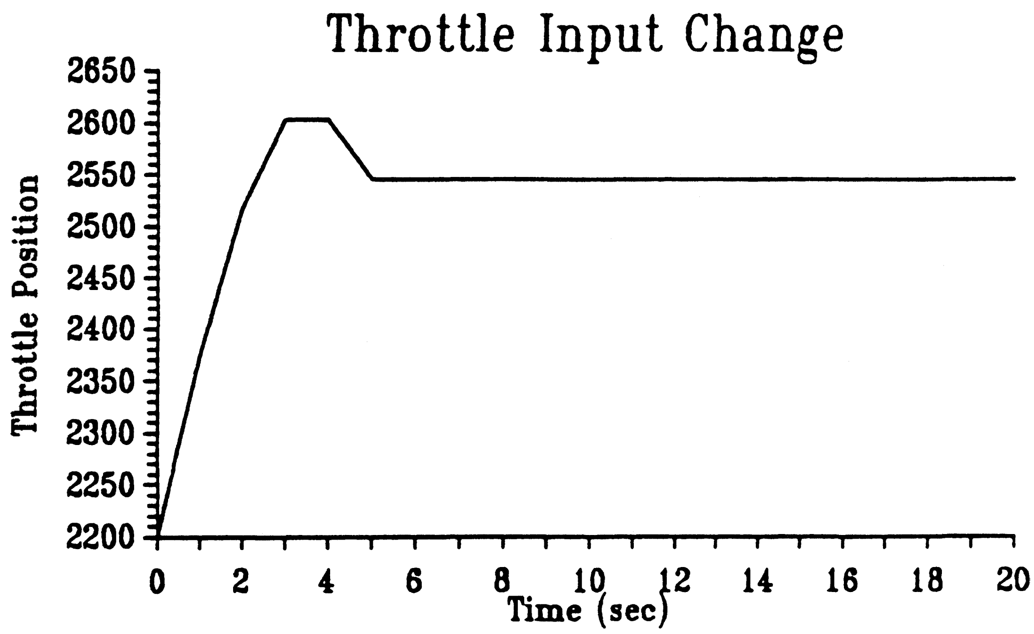


Figure 8.2 Tachometer Set Point Change
Vertical Speed constant at 0 fpm

example of over training. The network was trying to account for too many attributes of the output fuzzy set. A simplified training set was derived consisting of only four special cases. In each training pair one input neuron is set to one, the remaining three were set to zero, and an appropriate output example was provided. The resulting network performed better on incomplete sets however the performance with complete sets was degraded.

As long as the output fuzzy set was complete, the neural network defuzzifier gave a neat and continuous output, otherwise this implementation was useless. Two possibilities are now apparent, either a larger and more intelligently trained network is required, or this is simply a misapplication of the technology. For the knowledge-based controller implemented for this thesis, this has been a misapplication of a neural network. The original, simple defuzzifier gave a more satisfactory response (Figure 7.1) than did this example (Figure 8.2). Although some tuning may improve the response for the case when a complete output fuzzy set is available, there still exists the more serious problem of defuzzifying the incomplete sets.

With the success of the engine speed controller for implementing the knowledge-based control, an implementation of a neural network controller for the vertical speed was conceived. This would require a network with twelve input neurons, six neurons on one hidden layer and six neurons on the output layer. The training set consisted of forty-nine input / output pairs. This network has seven times as many training examples and 50% more weights to be adjusted than the engine speed controller network. From experience gained with the preceding tests, the number of training cycles that would be required would probably be at least 10,000 and each would take seven times longer than for the engine speed controller. It would therefore take on the order of twenty times as long to train this network. A conservative estimate then puts the training time on the 80386 based machine in the neighborhood of 100 hours (or one month on an 8 Mhz XT).

8.4 Conclusions

For the first trial the interest was in simplifying the specification of intermediate goals for the knowledge-based controller in the form of new set points. It appears that a neural network would function quite well in that role.

It is obvious that for the estimation of intermediate goals that the network could be extended to include further inputs and handle a greater range of conditions. The example used herein used purely numerical data to arrive at a numerical output. The network could also be extended to handle symbolic data [JON87] and even to mix the two. So it is possible to replace significant portions of the knowledge-based controller with a neural network.

For the second set of trials the purpose was to replace parts of the knowledge-based controller with a neural network. Neural networks can implement fuzzy logic with each input neuron representing a variable with a certainty factor assigned, and each output neuron likewise representing a variable with its potential representing the certainty factor. A major problem in utilizing a neural network is specifying the training set. It may not be possible to collect a complete training set which characterizes the particular problem domain. As demonstrated, the training time required quickly becomes prohibitively large as the network and the training set increase in size.

These trials have shown that simulating neural networks can provide some interesting alternatives to conventional programming and expert systems. The usefulness of neural network simulations is severely restricted in the micro-computer environment, however for small, specialized problems a micro-computer can be used to train them.

This investigation has only referred to the use of the back-propagation training algorithm. Other alternatives exist but this one appears to be the most general and easily realized.

CHAPTER 9

CONCLUSION AND DISCUSSIONS

This thesis has presented an application of knowledge-based systems to automatic control. The knowledge-based controller has incorporated fuzzy logic to be able to deal with some uncertainty, but mostly to allow the knowledge to incorporate linguistic terms. The controller was not based on a model of the system in a classical sense, but instead on the knowledge in the form of rules used by a human operator to make control action decisions.

Knowledge-based systems and fuzzy logic were briefly introduced to the reader. The knowledge-based system is a flexible and unique form of programming information for use by a computer. In their full realization including, explanation facilities, they are quite powerful and useful to human users. Explanation facilities were not incorporated into the controller in this thesis as the enquiring process did not require it. In addition, supporting such a facility would have made an already slow system even slower. The large amount of symbolic processing and logic tracing makes the knowledge-based system very slow. With the demand for ever faster controllers, knowledge-based controllers are at a large disadvantage.

The process used in this presentation was a simple training/personal aircraft. This aircraft is quite stable by industry standards and its response to control inputs and disturbances is quite slow. Even so, with the way the controller was setup, the controller was working much slower than real time on a 80386 based computer. The sample frequency could be lowered in the system to allow more time for decisions to be reached, but there is also a large amount of overhead in the Smalltalk/V-Prolog/V implementation. This environment allows dynamic typing and memory allocation. Implementation in a conventional, strongly typed software environment would greatly improve performance. The Smalltalk/V-Prolog/V environment's strong point is that it provides a very powerful and flexible development environment and was therefore the right tool for this study.

The knowledge-based system's strength in this controller was in the higher level decision making, the setting of intermediate goals, directing the overall operation plus the ability to adjust gains and sample times. These are the tasks more associated with conscience decision making for the human operator, in this case the pilot. The system seemed to get bogged down in the lower level control where the decision of output magnitude and direction is made. Many of the rules are repetitive yet with small variations for each primary instrument and require a fair amount of backtracking to find the most appropriate fuzzy output value. The lower level decisions become quite automatic with practice and require little or no conscience reasoning of an experienced pilot. Recognizing these reflex like control actions suggested that another solution may be more suitable here, such as neural networks.

In the knowledge-based controller a major problem encountered was incomplete specification of the controller rules. It must be clear that the rules were not necessarily incorrect, but incomplete, mostly due to the fact that most of the decision making in the reflex control actions was not recognized in the beginning. A problem also arose because some of the information a pilot uses in flying an aircraft comes from his own senses, flying by the seat of his pants if you will. The rules had to be modified and encoded to allow for this, as well a couple of sensors, pitch rate and acceleration, had to be added to the system to provide some of the additional information.

The pilots decision making and control is also much more fluid and flexible than could be captured in this knowledge-based controller. The computer's tireless attention and repetition compensates somewhat for this and therefore this failing is masked to some degree.

Neural networks were incorporated into the thesis at a very late date. The resurgence in interest had been gaining momentum over the past year and the author was introduced to them in a seminar just as the problem of finding set points for intermediate goals was being investigated. The problem of getting a pitch angle to establish a desired rate of climb was the most troublesome. A function could have been derived to provide the solution, but this would not have been in keeping with the philosophy of this thesis. At first the ability of a neural network to generalize after being trained on a relatively small training set was of the greatest interest. Subsequently, the ability of neural systems to deal with fuzzy logic was of interest and incorporated into the control of the engine speed. Neural networks also make the decision making seem more automatic as the solution is found quite quickly with a trained neural network, even in a machine with conventional architecture. The knowledge is stored implicitly in the system via the synapse or connection weights and does not have to be reasoned through for each solution. If neural networks do prove to be generally useful enough to warrant development of hardware to exploit them, they should provide some great advantages in speed of processing as well.

Knowledge-based controllers provide some powerful tools which used in the correct application will be advantageous for the control engineer. One clue to their use seems to be to apply them where a human would use conscience reasoning to solve the problem. Mixed with neural network technology and classical control, some powerful solutions are available to the control engineer.

With the above discussion in mind a number of areas where more specific research should be done can be selected. If knowledge-based systems are to be applied to automatic control then the practical aspects of their use should be approached. The methodology of selecting the most advantageous area for the application of a knowledge-based controller has not been addressed in detail. More importantly a systematic approach to designing such a controller has not been presented. It would be advantageous to an engineer considering the application of this type of controller to have a solid, acceptable approach to design which he could follow and be relatively certain of success. This would be a difficult area, as there still exists many questions concerning the design and construction of knowledge-based (expert) systems in general.

Neural computing is a field, which just being essentially reborn, is wide open for work. A major problem here is finding appropriate training sets for a given application. Another problem is weeding out the rhetoric and selecting appropriate and advantageous applications of the technology. Indeed, neural networks may only be a passing fad once again. In areas where it is not possible to put together a training set a knowledge-based system may be developed to use in the initial implementation of a controller and have it act as a teacher for a neural network. Eventually the knowledge-based controller would be discarded and replaced entirely by the neural network. This would be similar to a person practicing a task which he must think about, all the while the brain is learning to do the task in a more automatic and efficient fashion.

Assuming that practical hardware to take full advantage of the neural network structure is forth coming, there is the possibility of neural network controllers being trained or adapted on-line in real time control applications. Departing for a moment from the knowledge-based and fuzzy logic control, a neural network may be applied as a controller in a more conventional form. Future investigation could try to implement this technology by training a network to provide proper control values to a process based on some characterization of the process's current state and the desired state.

This thesis has broached the subject which might best be termed flexible intelligent control. The constraints put on a computer by a fixed sample time can be relaxed and the sample time is allowed to float. This is linked with using knowledge, reasoning and experience to create a controller. Situations of greater importance receive more attention from the controller in terms of computer resources needed to find an effective solution. Instead of relying on the speed of the computer, the emphasis is on intelligent control and the application of resources by creating a controller that can adapt itself to the overall situation, not just to parameter variations of the process. Work is therefore required in designing controllers with floating sample times and the knowledge and actions to accomplish this.

CHAPTER 10

EPILOGUE

10.1 Introduction

Should robotic pilots replace human pilots in the cockpit of tomorrow's aircraft? This chapter addresses the issues of developing sophisticated autopilots, in essence intelligent robots, to replace human operators. Directly related to the first question are the following:

1) For who's benefit would this technology be applied?
2) What costs would be associated with its implementation?

Aircraft accidents are very dramatic and news worthy items. Air travel is no less safe than any other means of travel, yet it has had a great amount of attention given to it for safety and concern over accidents [PER84]. Most often the probable cause of the accidents that occur is assessed to pilot error somewhere in the chain of events. Is it possible to create the perfect pilot, one that never forgets and always follows the correct procedure; whose attention never wanders from the task at hand and thereby eliminate most of the accidents?

10.2 Current Autopilots

Autopilots range in complexity in today's aircraft from simple wing levelers in small private aircraft to autopilots that can perform the complete flight from takeoff to landing in a jumbo jet. On long cross country flights these machines can save a pilot from much tedious work therefore allowing him to be more alert and rested for the final stage of flight, the landing. It has been argued, particularly by pilots, that the automation of the front office in an aircraft has gone the wrong route. The wrong assignments have been given to the machine and to the human. In today's aircraft, the machine does the flying while the human is there to monitor the aircraft's condition and progress. Continuing in the name of safety, more and more of the directing of an aircraft's flight is being transferred to the ground and away from the cockpit. The philosophy that is demonstrated by these actions is "To get rid of the human error, get rid of the human pilot".

With the increasing complexity of today's aircraft systems could you give a more ill-suited task to a human pilot than to monitor these systems? Flying the aircraft is a constantly changing task filled with new challenges which will keep a human pilot's attention. Since there currently exists autopilots that can fly the aircraft, might it not be best to replace the human completely with a computer? But what of the tasks a pilot has not seen before or has not been specifically trained to handle?

In today's cockpit, a phenomenon referred to as automation complacency has arisen. The pilot becomes so used to his automated environment always working correctly that errors in the system go unnoticed and have led to disastrous results [TV289]. Can it be expected that a human being can take over from a machine in an emergency after sitting for hours quietly monitoring a cockpit full of information? It is going to take time for him to just get into the pilot mode, interpret the error ennunciators and indications before taking action.

10.3 Robot Pilots

If we lose the man, in any environment, we lose the ability to be creative and to deal with ambiguous and new situations. Artificial intelligence promises to provide these to us in the future. Is this really possible, or is this more the pursuit of technology for its own benefit instead of an improved environment for human beings. If this direction is continued we will end up with an incompetent culture, one containing competence for a task (the machines) but no real expertise. An European Economic Community (EEC) sponsored program is pursuing "human-machine symbiosis" and the development of a human centred manufacturing program. They want to reintegrate the skilled craftsman into the manufacturing process and utilize the full skills of the human operator. This makes commercial sense as there is far more potential available than is being used. We have all around us the human mind which is only being used to a fraction of its ability, so perhaps much of the artificial intelligence and expert system research is wasted effort. A man has the ability to imagine what is in the material and understand and adapt to the qualities of the material [TV189].

For flying, the pilot must have an understanding of the aircraft, its flying qualities, its idiosyncracies and the general laws of flight. In an incident involving a three engine jet, pitch control was lost due to a jammed elevator. The pilot used the adverse characteristic of pitching moment changes due to differential power application and centre of gravity adjustment by shifting the passengers in order to control the aircraft and accomplish a safe landing. Up until this last year, no one had experienced the loss of such a substantial portion of a commercial aircraft's structure as in the Aloha incident in Hawaii. These are incidents for which a robot pilot would not have been specifically trained to handle and raises the question of would it have dealt with the situation adequately and saved the flight?

If the human is removed from the cockpit (or any operator position), human error can creep in, in other ways. The situation worth considering is an airplane which flies into the side of a mountain while the computer programmer is off quietly sipping a martini in some bar [TV289]. At least the human pilot's failings are offset by his will to survive. With robotic pilots you will have the situation of one pilot, repeated many times, flying many aircraft and potentially repeating the same mistake many times. Each of these examples of this pilot can only draw on the experience it was trained with. It cannot draw on a lifetime of diverse experiences, it cannot modify itself based on its own experiences nor those of others. Would a machine designed to fly airplanes be designed to interact with other pilots, human and machine, and learn from the experience of others?

10.4 Benefits and Costs

To who's benefit is this automation to be performed? For aircraft, it would be sold to the public as a safety feature. In manufacturing it is sold as cost cutting and efficiency. It can certainly be a boost to an engineers ego to design a complicated system, incorporating the latest technology and in the end save his employer money in salary and benefits. For the corporation comes the benefit of prestige, possible off shoot business and maybe increased difficulty of assessing responsibility for mistakes.

In Charles Perrow's book [PER84], he examines the occurrence of what he terms normal accidents. These are incidents caused by multiple system failures with unforeseen inter-dependencies and causal relationships which lead to catastrophic ends. When these failures do occur, technology is looked to for the solution. Thus a system prone to failure is put on top of another creating the potential for further unseen interactions and masking of other seemingly unconnected failures. In aviation, both industry and government officials are pressing for more automation in their respective systems, thus reducing the number of personnel. "Both of these, I would suspect, will lead to much tighter coupling - that is, less resources for recovery from incidents" [PER84, pg161]. In essence what is being said here is the reason there has not been more catastrophes (in aviation and elsewhere), is due to the fact that the human being is in the system.

Society today increasingly depends on experts, fast service and disposable items. How often has the story been told of people discarding, at times expensive items, when a simple repair would have returned the item to service. Increasing dependence on machines through making them the experts will continue the slide to an incompetent society.

The question was put to a group of machinists, what they would prefer in their home workshop - a NC machine (lathe) or a manual one - overwhelmingly the answer was a manual machine. One of the most well-liked aircraft is the venerable Piper J-3 Cub and that general class. These aircraft are simple flying machines without electrics of any kind. Obviously not all people will feel the same, but then automation to various extents has its support and uses. The engineering profession should make special effort to understand how the environment is to be oriented and how responsibility may be reassigned in the workplace.

10.5 Conclusion

The research from this thesis should not be continued to the implementation of a computerized pilot. Indeed it should not be implemented toward the robotic replacement of any skilled task where the long term costs to society may well outweigh the short term gains in profit and prestige. The skills of any trade or profession, if not continued to be practiced and improved upon will in the end be lost. The EEC project toward a human centred manufacturing program and the development of a "human-machine symbiosis" is a promising approach. Here the machine will continue as the tool with a human being directly involved in the process as opposed to the provision of a human material. The ability of human beings to think and reason for ourselves is a resource to be used instead of an attribute to be controlled and circumvented [TV189].

As a further recommendation, graduate study should require an analysis of the student's research based on its possible use and misuse. The ethics of their research and their responsibility for it should be an integral part of a graduate students research requirements. As technology grows and its application proliferates, it becomes harder for elected representatives of the society to implement safe guards to protect today's and tomorrow's society. Study of ethics and moral responsibility is becoming a requirement in undergraduate years and should be a requirement of every graduate students research as well in order to raise their awareness of this issue. It will be to the individual's and society's advantage for the future.

APPENDIX A

Parameter Values for PA-28

Calculation of the Moment of Inertia - the aircraft was modeled as three cubes. Moments of inertia were calculated for each and they were then combined.

I_T = B = 1387.06 lb·ft·s^2

Other values used;

C_{D0} = 0.037

S = 160 ft^2

ρ_{4000} = 0.002054

l_t = 58"

S_t = 25 ft^2

(a_1) = 7.0 /degree = 401 /rad.

efficiency factor = e = 3.375

$\tilde{c}$ = 5.25 ft

$Cm_\alpha \sim 0$

Cm_{ac} = -0.025 /rad [BERTIN & SMITH]

CG @ 0.33$\tilde{c}$

a_c @ 0.25$\tilde{c}$

$C_{L\alpha} = a_w = 2\pi\ A/(A + 2)$ = 4.635 /rad [SECKEL]

$a_t = \pi A_1\ /\ (1 + \sqrt{(1 + (A_1/2)^2)})$ = 3.88 /rad

$\frac{d\epsilon}{dt} = 4\ /(A_w + 2)$ = 0.5246 /rad

Vol_f = 113.0 cu.ft.

fineness ratio = l/D = 5.9

fuselage moment factor = K_f = 0.88

$Cm_{\alpha f}$ = 0.233 /rad

Cm_{it} = -1.15236 /rad

α_{w0} = 0 ; symmetrical wing, therefore zero lift @ α = 0

i_w = 3 degrees; a nominal value for most light aircraft

m = W /g = 1850 / 32.2 = 57.5 $lb \cdot s^2/ft$

k_2 = 1.0583E-4

k_3 = 5.7953E-3

k_5 = 1.3257E-2

propeller pitch = 58"

efficiency = e = 0.84

rated horsepower @ 2700 rpm = 160 hp

All aircraft data, ie. lengths, weights etc, were taken from the Cherokee manual and scaled from the diagram of the aircraft.

BIBLIOGRAPHY

[AER86] Aerodynamics, Class Notes, University of Waterloo, 1986.

[ARC87] Architecture of Expert Systems, The, Class Notes, McMaster University, 1987.

[CHE76] Cherokee Cruiser Information Manual, Piper Aircraft Corporation, 1976.

[FLI79] Flight Training Manual, 3rd Edition, Gage Publishing Ltd., Toronto, Canada, 1979.

[FLI85] Flight Test Guide, Private and Commercial Pilot Licences, Aeroplanes, Ninth Edition, Transport Canada, March 1985.

[FRO63] From The Ground Up, 22nd Edition, Aviation Publishers Co. Ltd., Ottawa, Canada, 1963.

[HAC70] Hacker, T., Flight Stability and Control, American Elsevier Publishing Company Inc., New York, 1970.

[JAC86] Jackson, P., Introduction to Expert Systems, Addison-Wesley Publishing Company, Great Britian, 1986.

[JON87] Jones, W.P., Hoskins, J., "Back-Propagation, A generalized delta learning rule", BYTE, October 1987, pp. 155-162. 10 85

[JOS88] Josin, G., "Neural Networks for Electrical and Computer Engineering", Proceedings Canadian Conference on Electrical and Computer Engineering, Vancouver, Canada, Nov. 1988, pp. 97-100.

[KAU75] Kaufmann, A., Introduction to the Theory of Fuzzy Subsets, Volume 1, Academic Press, New York, 1975.

[PER84] Perrow, C., Normal Accidents, Basic Books Inc., New York, 1984.

[PRI80] Private Pilot Manual, Jeppeson Sanderson Inc., U.S.A., 1980.

[RIC86] Richards, B.L., Programming in Fuzzy Logic: Fuzzy Prolog, Master of Science thesis, Air Force of Technology Air University, 1986.

[RIC88] Richards, B.L., "When Facts Get Fuzzy", BYTE, April 1988, pp. 285-290.

[SEC64] Seckel, E., Stability and Control of Airplanes and Helicopters, Academic Press, New York, 1964.

[SHI87] Shirley, R.S., "Some Lessons Learned Using Expert Systems for Process Control", IEEE Control Systems, Vol. 7, No. 6, Dec. 1987, pp. 11-15.

[TON77] Tong, R.M., "A Control Engineering Review of Fuzzy Systems", Automatica, Vol. 13, 1977, pp. 559-569.

[TRE88] Trelease, R.B., "Connectionism, Cybernetics and the Cerebellum", AI Expert, August 1988, pp. 30-36.

[TV189] Vista Presents, March 20, 1989, TVO Broadcasting Toronto.

[TV289] Vista Presents, March 27, 1989, TVO Broadcasting, Toronto.